

```
import AVFoundation
import Accelerate
import Combine
import CoreGraphics
import CoreImage
import CoreML
import CoreMedia
import CoreVideo
import CreateMLComponents
import Dispatch
import Foundation
import IOKit
import
import ImageIO
import MetalPerformanceShaders
import NaturalLanguage
import OSLog
import TabularData
import UniformTypeIdentifiers
import VideoToolbox
import Vision
import _Concurrency
import _StringProcessing
import _SwiftConcurrencyShims
import os
import

/// A model you train with videos to classify a person's body movements.
@available macOS 11.0
@available
@available
public struct MLActionClassifier : Sendable

    /// The underlying Core ML model of the action classifier stored in memory.
    public var model : MLModel get

        /// The model configuration parameters the action classifier used during its
        /// training session.
        public let modelParameters
    MLActionClassifier ModelParameters

    /// Measurements of the action classifier's performance on the training
    /// dataset.
    public var trainingMetrics : MLClassifierMetrics get

        /// Measurements of the action classifier's performance on the validation
        /// dataset.
    public var validationMetrics : MLClassifierMetrics get
```

```
    /// A collection of predictions, each paired with its confidence, for a
    /// range of video frames.
public struct Prediction Sendable

    /// The range of frame rates the action classifier used to make its
    /// prediction.
public var frameRange Range Int

    /// An array of prediction labels and their confidences for an action.
public var results String
Double

    /// Creates an action classifier with a training dataset represented by a
    /// data source.
    ///
    /// - Parameters:
    ///   - trainingData: A collection of labeled images represented by a
data
    ///   source.
    ///
    ///   - parameters: An ``MLActionClassifier/ModelParameters-
swift.struct``
    /// instance you use to configure the model for the training session.
public init MLActionClassifier DataSource
    MLActionClassifier ModelParameters
        throws

    /// Creates an action classifier from a training session checkpoint.
    ///
    /// - Parameters:
    ///   - checkpoint: A checkpoint from an action classifier training
session.
public init MLCheckpoint throws

    /// Begins an asynchronous action classifier training session.
    ///
    /// - Parameters:
    ///   - trainingData: A collection of labeled videos represented by a data
    ///   source.
    ///
    ///   - parameters: An ``MLActionClassifier/ModelParameters-
swift.struct``
    /// instance you use to configure the model for the training session.
    ///
    ///   - sessionParameters: An ``MLTrainingSessionParameters`` instance you use
    ///     to configure the training session.
    ///
    /// - Returns: An ``MLJob`` that represents the action classifier training
```

```

    /// session.
    public static func train
    MLActionClassifier DataSource
    MLActionClassifier ModelParameters
        MLTrainingSessionParameters
            throws MLJob MLActionClassifier

    /// Creates an asynchronous training session for an action classifier.
    ///
    /// - Parameters:
    /// - trainingData: A collection of labeled videos represented by a data
    /// source.
    ///
    /// - parameters: An ``MLActionClassifier/ModelParameters-
    swift.struct``
        /// instance you use to configure the model for the training session.
    ///
    /// - sessionParameters: An ``MLTrainingSessionParameters`` instance you use
    /// to configure the training session.
    ///
    /// - Returns: An ``MLTrainingSession`` that represents the action
    /// classifier training session.
    public static func makeTrainingSession
    MLActionClassifier DataSource
    MLActionClassifier ModelParameters
        MLTrainingSessionParameters
            throws
    MLTrainingSession MLActionClassifier

    /// Creates an asynchronous training session for an action classifier by
    /// restoring an existing training session's state from its parameters.
    ///
    /// - Parameters:
    /// - sessionParameters: The
    ``MLTrainingSessionParameters`` instance you
        /// used to create the training session using
    ///
    ``MLActionClassifier/makeTrainingSession(trainingData:parameters:sessionParameters:)``.
    ///
    /// - Returns: An ``MLTrainingSession`` that represents the action
    /// classifier training session.
    public static func
    restoreTrainingSession
    MLTrainingSessionParameters throws
    MLTrainingSession MLActionClassifier

    /// Begins or continues an asynchronous action classifier training session.

```

```
///  
/// - Parameters:  
/// - session: An ``MLTrainingSession`` instance that represents  
the  
/// training session.  
///  
/// - Returns: An ``MLJob`` that represents the action classifier training  
/// session.  
///  
///  
public static func resume _  
MLTrainingSession MLActionClassifier throws  
MLJob MLActionClassifier  
  
/// Generates a prediction for each action the classifier recognizes in the  
/// video.  
///  
/// - Parameters:  
/// - video: The location of a video you want the action classifier to  
/// analyze.  
///  
/// - Returns: An array of predictions.  
public func prediction URL throws  
MLActionClassifier Prediction  
  
/// Generates a sequence of predictions for each video input.  
///  
/// - Parameters:  
/// - videos: An array of locations to videos you want the action classifier  
/// to analyze.  
///  
/// - Returns: A array of prediction arrays. The index of each inner array  
/// corresponds to the video index in the input array.  
public func predictions URL throws  
MLActionClassifier Prediction  
  
/// Generates metrics describing the action classifier's performance on  
/// labeled videos represented by a data source.  
///  
/// - Parameters:  
/// - annotatedVideos: A collection of labeled videos represented by a  
data  
/// source.  
public func evaluation  
MLActionClassifier DataSource throws MLClassifierMetrics  
  
/// Exports the action classifier as a Core ML model file to a location in  
/// the file system.  
///  
/// - Parameters:
```

```
    /// - fileURL: The location URL in the file system where you want to
  save
    /// the model.
    ///
    /// - metadata: Descriptive information to include with the exported model
    /// file.
public func write URL
MLModelMetadata nil throws

    /// Exports the action classifier as a Core ML model file to the file path.
    ///
    /// - Parameters:
    /// - path : The location path in the file system where you want to save
  the
    /// model.
    ///
    /// - metadata: Descriptive information to include with the exported model
    /// file.
public func write String
MLModelMetadata nil throws

@available macOS 11.0
@available
@available
extension MLActionClassifier

    /// The video augmentations for an action classifier training session.
public struct VideoAugmentationOptions OptionSet
Codable Sendable

    /// The corresponding value of the raw type.
    ///
    /// A new instance initialized with `rawValue` will be equivalent to this
    /// instance. For example:
    ///
    ///     enum PaperSize: String {
    ///         case A4, A5, Letter, Legal
    ///     }
    ///
    ///     let selectedSize = PaperSize.Letter
    ///     print(selectedSize.rawValue)
    ///     // Prints "Letter"
    ///
    ///     print(selectedSize == PaperSize(rawValue:
selectedSize.rawValue)!)
    ///     // Prints "true"
public let rawValue Int

    /// A video augmentation that creates a horizontally flipped copy of a
```

```
    /// sample video.
    public static let horizontalFlip
MLActionClassifier VideoAugmentationOptions

    /// Creates a video augmentation option set from a raw value.
    ///
    /// - Parameters:
    ///   - rawValue: The underlying raw value of the option set.
    public init Int

    /// The type of the elements of an array literal.
    @available macOS 11.0
    @available
    @available
    public typealias ArrayLiteralElement
MLActionClassifier VideoAugmentationOptions

    /// The element type of the option set.
    ///
    /// To inherit all the default implementations from the `OptionSet` protocol,
    /// the `Element` type must be `Self`, the default.
    @available macOS 11.0
    @available
    @available
    public typealias Element
MLActionClassifier VideoAugmentationOptions

    /// The raw type that can be used to represent all values of the conforming
type.
    ///
    /// Every distinct value of the conforming type has a corresponding unique
value of the `RawValue` type, but there may be values of the `RawValue` type
that don't have a corresponding value of the conforming type.
    @available macOS 11.0
    @available
    @available
    public typealias RawValue Int

@available macOS 11.0
@available
@available
extension MLActionClassifier : CustomStringConvertible
CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible
```

```
/// A text representation of the action classifier.
public var description String get

/// A text representation of the action classifier that's suitable for
/// output during debugging.
public var debugDescription String get

/// A description of the action classifier shown in a playground.
public var playgroundDescription Any get

@available macOS 11.0
@available
@available
extension MLActionClassifier

/// Parameters that affect the training process of an action classifier.
public struct ModelParameters

    /// The action classifier's validation dataset.
    public var validation
    MLActionClassifier ModelParameters ValidationData

        /// The number of videos the training session uses for each of its
        /// training iterations.
        public var batchSize Int

        /// The largest number of training iterations the training session can
        /// use.
        public var maximumIterations Int

        /// The number of frames the training session uses to train an action
        /// classifier.
        public var predictionWindowSize Int

        /// The variations the training session uses to generate more variety in
        /// the training dataset.
        public var augmentationOptions
    MLActionClassifier VideoAugmentationOptions

        /// The algorithm the training session uses to train the action
        /// classifier.
        public var algorithm
    MLActionClassifier ModelParameters ModelAlgorithmType

        /// The number of frames the training session uses per second of video
        /// to train an action classifier.
        public var targetFrameRate Double
```

```

    /// Creates a new set of training parameters for an action classifier
    /// with the validation dataset.
    ///
    /// - Parameters:
    ///   - validation: A validation dataset represented by an
    ///     ``MLActionClassifier/ModelParameters-
    swift.struct/ValidationData``
        /// instance.
        ///
        /// - batchSize: The number of videos the training session uses for
each
        ///   of its training iterations.
        ///
        /// - maximumIterations: The largest number of training iterations the
        ///   training session can use.
        ///
        /// - predictionWindowSize: The number of frames the training session
        ///   uses to train an action classifier. For example, set to 60 to
        ///   capture actions that take 2 seconds from videos that have a frame
        ///   rate of 30 frames per second.
        ///
        /// - augmentationOptions: The variations the training session uses to
        ///   generate more variety in the training dataset.
        ///
        /// - algorithm: The algorithm the training session uses to train the
        ///   action classifier.
        ///
        /// - targetFrameRate: The number of frames the training session
uses
        ///   per second of video to train an action classifier.
public init
MLActionClassifier ModelParameters ValidationData
                        Int
Int
Int

MLActionClassifier VideoAugmentationOptions

MLActionClassifier ModelParameters ModelAlgorithmType
                        Double

    /// The action classifier training algorithm options.
public enum ModelAlgorithmType : Equatable, Sendable

    /// The spatial-temporal graph convolution neural-network
algorithm.

```

```

case stgcn

    /// Hashes the essential components of this value by feeding
them into the
    /// given hasher.
    ///
    /// Implement this method to conform to the `Hashable`
protocol. The
    /// components used for hashing must be the same as the
components compared
    /// in your type's `==` operator implementation. Call
`hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your implementation of `hash(into:)`,  

provided,  

    /// don't call `finalize()` on the `hasher` instance
    /// or replace it with a different instance.
    /// Doing so may become a compile-time error in the future.
    ///
    /// - Parameter hasher: The hasher to use when combining  

the components
    /// of this instance.
public func hash           inout Hasher

    /// Returns a Boolean value indicating whether two values are
equal.
    ///
    /// Equality is the inverse of inequality. For any values `a` and
`b`,
    /// `a == b` implies that `a != b` is `false`.
    ///
    /// - Parameters:
    /// - lhs: A value to compare.
    /// - rhs: Another value to compare.
public static func
MLActionClassifier ModelParameters ModelAlgorithmType
MLActionClassifier ModelParameters ModelAlgorithmType      Bool

    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different
executions of
    /// your program. Do not save hash values to use during a future
execution.
    ///
    /// - Important: `hashValue` is deprecated as a
`Hashable` requirement. To
    /// conform to `Hashable`, implement the `hash(into:)`
requirement instead.
    /// The compiler provides an implementation for `hashValue`

```

for you.

```
public var hashCode Int get

/// The source of a validation dataset for an action classifier.
public enum ValidationData

    /// A validation dataset derived by randomly selecting a portion of
    /// the action classifier's training dataset using the split
    /// strategy.
    case split MLValidationStrategy

    /// A validation dataset represented by a data source.
    case dataSource MLActionClassifier DataSource

    /// An empty validation dataset that skips the model validation
    /// phase after training.
    ///
    /// Use this case when you don't have validation data while
    /// preventing Create ML from using any of your training dataset
for
    /// validation.
    case none
```

```
@available macOS 11.0
@available
@available
extension MLActionClassifier
```

```
/// A data source for an action classifier.
public enum DataSource

    /// The location of a directory of video files, and the location of an
    /// action annotation file.
    ///
    /// - Parameters:
    ///   - at: The location of a directory that contains video files.
    ///   - annotationFile: The location of a JSON or CSV file with
    ///     object annotations for the images.
    ///   - videoColumn: The name of the column that contains the
    ///     URLs to the video files.
    ///   - labelColumn: The name of the column that contains the
    ///     labels of the action the person demonstrates in the video file.
    ///   - startTimeColumn: The name of the column that contains
    ///     the action's starting-time in the video file.
    ///   - endTimeColumn : The name of the column that contains the
    ///     action's ending-time in the video file.
```

```

case directoryWithVideosAndAnnotation      URL
      URL           String           String
      String     nil            String   nil

      /// The location of a folder with subfolders each of which contain sample videos of an action.
      /**
      /// The action classifier task uses each subfolder's name as the label for an action.
case labeledDirectories      URL

      /// The location of a folder that contains video files whose names you use to label corresponding actions.
      /**
      /// Use this case to create a data source from a folder of all your sample video files. Name each file using
          /// the action's label, followed by a period and an arbitrary string, followed by the file extension. For
          /// example, an exercise action classifier might have files named squat.3.mov, lunge.1.mov, lunge.2.mov, and so on.
case labeledFiles      URL

      /// A data table that contains the human body landmark movement data.
      /**
      // – Parameters:
      /// – table: A data table that contains the human body landmark locations and the action annotations.
      /// – sessionIdColumn: The name of the column that contains the action session's unique identifier.
      /// – labelColumn: The name of the column that contains the labels of the action the person demonstrates in
          /// the session.
      /// – featureColumn : The name of the column that contains the movement data.
@available                      11.0          14.0
@available
@available
case labeledKeypointsData      MLDataTable
      String
      String
      String

      /// A data table that contains the locations of the video files and the action annotations.
      /**
      // – Parameters:
      /// – table: A data table that contains the video file locations and the action annotations.
      /// – videoColumn: The name of the column that contains the URLs to the video files.

```

```

    /// - labelColumn: The name of the column that contains the
labels of the action the person demonstrates in
    /// the video file.
    /// - startTimeColumn: The name of the column that contains
the action's starting-time index in the video
    /// file.
    /// - endTimeColumn : The name of the column that contains the
action's ending-time index in the video file.
@available 11.0 14.0
@available
@available
case labeledVideoData MLDataTable
String String nil
String nil

    /// A data source made up of keypoints in a data frame.
    ///
    /// The data frame must contain a column of session identifiers, a
column of labels, and a column of keypoints.
    /// Each set of keypoints must be a multi-dimensional 1x3x18 array
containing the x, y, z coordinates of each
    /// of the 18 keypoints. See `VNRecognizedPointsObservation`  

for more details.
    ///
    /// - Parameters:
    /// - dataFrame: A `DataFrame` containing keypoints and
labels.
    /// - sessionIdColumn: The name of the column containing
session identifiers. Defaults to "session_id".
    /// - labelColumn: The name of the column containing the
labels. Defaults to "label".
    /// - featureColumn: The name of the column containing the
keypoints. Defaults to "keypoints".
@available macOS 12
@available
@available
case labeledKeypointsDataFrame DataFrame
String
String
String

    /// A data source made up of video references in a data frame.
    ///
    /// The data frame must contain a column of video file paths and a
column of labels. It can also contain
    /// columns with start and end times.
    ///
    /// - Parameters:
    /// - dataFrame: A `DataFrame` containing video paths and
labels.

```

```
    /// - videoColumn: The name of the column containing the video
paths. Defaults to "videoPath".
    /// - labelColumn: The name of the column containing the
labels. Defaults to "label".
    /// - startTimeColumn: The name of the column containing the
start time. If `nil` start time is 0.
    /// - endTimeColumn: The name of the column containing the
end time. If `nil` end time is the end of the video.
@available macOS 12
@available
@available
case labeledVideoDataFrame DataFrame
String                                         String
String                                         nil
String                                         nil

    /// Processes the data source and returns a data frame that contains
file URLs and annotations.
    ///
    /// This method collects file names from the filesystem if necessary. If
the data source is already in table
    /// format it renames the columns to the default column names. This
method returns nil if the data source
    /// contains key points.
@available macOS 14.0 iOS 17.0
@available
public func gatherAnnotatedFileNames    throws
DataFrame

    /// Generates a data table of the data source's video URL locations and
action annotations.
    ///
    /// The data table includes a column for the annotation's label, and if
applicable, the annotation's starting-
    /// and ending-time indices.
    ///
    /// - Returns: A data table.
@available                                     11.0          14.0
    "Use gatherAnnotatedFileNames()"
@available
@available
public func videosWithAnnotations    throws
MLDataTable

    /// Extracts key points from video files if necessary.
    ///
    /// If the data source already contains keypoints, this method just
renames the data frame columns to the
    /// defaults.
    ///
```

```

    /// - Parameters:
    ///   - targetFrameRate: The number of frames per second the
    method uses to extract body landmarks from the
    ///   data source.
    /// - Returns: A data frame that contains a column for hand joint
    locations and a column of hand action
    ///   annotations.
    @available macOS 14.0 iOS 17.0
    @available
    public func extractKeypoints           Double
                                            throws
    DataFrame

    /// Generates a data table with action annotations of the data source.
    ///
    /// - Parameters:
    ///   - targetFrameRate: The number of frames per second the
    method uses to extract body landmarks from the
    ///   data source. This no effect if the data source is an
    ///
``MLActionClassifier/DataSource/labeledKeypointsDataFrame(_:se
ssionIdColumn:labelColumn:featureColumn:)``
    ///   or an
    ///
``MLActionClassifier/DataSource/labeledKeypointsData(table:ses
sionIdColumn:labelColumn:featureColumn:)``.
    ///
    /// - Returns: A data table.
    @available                         11.0          14.0
    "Use extractKeypoints(targetFrameRate:)"
    @available
    @available
    public func keypointsWithAnnotations
Double
                                            throws
    MLDataTable

    /// Generates a data table by splitting the data source into strata.
    ///
    /// - Parameters:
    ///   - proportions: An array of proportions, each in the range
    ` [0.0, 1.0]`.
    ///   - seed: A seed number for the random-number generator.
    ///   - labelColumn: The name of the column that you want to
    stratify.
    ///
    /// - Returns: A new data table.
    @available                         11.0          14.0
    "Use DataFrame.stratifiedSplit(on:by:)"
    @available
    @available

```

```
    public func stratifiedSplit  
        Int  
        MLDDataTable  
        String  
        Double  
        throws  
  
  
@available macOS 11.0  
@available  
@available  
extension MLActionClassifier ModelParameters  
CustomStringConvertible CustomDebugStringConvertible  
CustomPlaygroundDisplayConvertible  
  
    /// A text representation of the model parameters.  
    public var description String get  
  
    /// A text representation of the model parameters that's suitable for output  
    /// during debugging.  
    public var debugDescription String get  
  
    /// A description of the model parameters shown in a playground.  
    public var playgroundDescription Any get  
  
@available macOS 11.0  
@available  
@available  
extension  
MLActionClassifier ModelParameters ModelAlgorithmType  
Hashable  
  
    /// A model you train to classify motion sensor data.  
    ///  
    /// An activity classifier is a machine-learning model that your app can use to  
    /// categorize user _activities_, based on  
    /// the motion of the user's device.  
    ///  
    /// You create an activity classifier by gathering a training dataset of a device's  
    /// motion sensors, such as the  
    /// accelerometer and gyroscope on an Apple Watch. For example, you can create  
    /// an activity classifier that recognizes a  
    /// person waving, shaking hands, or throwing a ball by gathering the motion-  
    /// sensor data from people performing those  
    /// activities.  
    ///  
    /// Evaluate your trained activity classifier by calling  
    ///  
    ``MLActivityClassifier/evaluation(on:featureColumns:labelColum-  
    n:recordingFileColumn:)``
```

```
/// ``MLSoundClassifier/evaluation(on:)`` with a dataset that's
// completely distinct from the training and
// validation datasets. Inspect the metrics the method returns and decide whether
// the activity classifier performs
// with enough accuracy. For example, you can assess how often the activity
// classifier confuses a person waving for
// shaking hands, or vice versa. If the classifier makes too many mistakes, you
// can train another classifier with
// different parameters, or with a training dataset that has more or better motion-
// sensor examples.
//
// When you're satisfied with an activity classifier, save it as a Core ML model file,
// and add it to your Xcode
// project. Use it to predict the user's activity based on the motion-sensor data
// your app captures from the user's
// device.
@available macOS 10.15
@available
@available
public struct MLActivityClassifier : Sendable

    /// The underlying Core ML model of the activity classifier stored in memory.
    public var model : MLMModel

    /// The model configuration parameters the activity classifier used during its
    // training session.
    public let modelParameters : MLActivityClassifier.ModelParameters

    /// Measurements of the activity classifier's performance on the training
    // dataset.
    public var trainingMetrics : MLClassifierMetrics { get }

    /// Measurements of the activity classifier's performance on the validation
    // dataset.
    public var validationMetrics : MLClassifierMetrics { get }

    /// The name of the label column the activity classifier used during its training
    // session.
    ///
    /// This property reflects the name of the data table column or annotation file
    // column the training session used to
    /// label each activity.
    ///
    /// - Note: The ``MLActivityClassifier`` instance provides a
    // default name if you trained it with a data source
    /// that's set to
    ``MLActivityClassifier/DataSource/labeledDirectories(at:)``.
    ///
    /// Changing the value of this property doesn't retrain the model or affect its
    behavior.
```

```
public var labelColumn String  
  
    /// The names of the feature columns the activity classifier used during its  
    training session.  
    ///  
    /// Changing the value of this property doesn't retrain the model or affect its  
    behavior.  
public var featureColumns String  
  
    /// The name of the column that contains the data files the activity classifier  
    used during its training session.  
    ///  
    /// This property reflects the name of the data table column or annotation file  
    column the training session used to  
    /// locate each activity's data files.  
    ///  
    /// – Note: The ``MLActivityClassifier`` instance provides a  
    default name if you trained it with a data source  
    /// that's set to  
``MLActivityClassifier/DataSource/labeledDirectories(at:)``.  
    ///  
    /// Changing the value of this property doesn't retrain the model or affect its  
    behavior.  
public var recordingFileColumn String  
  
    /// Creates an activity classifier with a training dataset represented by a data  
    source.  
    ///  
    /// Use this initializer to create an activity classifier with an  
``MLActivityClassifier/DataSource``. To configure  
    /// the training process, initialize the activity classifier with an  
    /// ``MLActivityClassifier/ModelParameters-swift.struct``  
instance. For example, you can explicitly define the  
    /// validation dataset instead of allowing the model to choose a random  
selection of your training data.  
    /// Alternatively, set ``MLActivityClassifier/ModelParameters-  
swift.struct/validationData`` to `nil` to allow the  
    /// activity classifier to choose the validation data for you from among your  
training data. This lets you set  
    /// other parameters — like  
``MLActivityClassifier/ModelParameters-swift.struct/maximumIte-  
rations`` and  
    ///  
``MLActivityClassifier/ModelParameters-swift.struct/batchSize``  
— to nondefault values.  
    ///  
    /// – Parameters:  
    /// – trainingData: An  
``MLActivityClassifier/DataSource`` instance.  
    /// – featureColumns: The names of the columns in an annotation file  
that contain sensor data.
```

```

    /// - labelColumn: The name of the column in an annotation file that
contains the activity labels if
    /// `trainingData` uses
    ///
``MLActivityClassifier/DataSource/directoryWithDataAndAnnotation(at:annotationFileName:timeStampColumn:labelStartTimeColumn:
labelEndTimeColumn:)``.
    ///
    /// The initializer ignores this parameter if `trainingData` uses
    ///
``MLActivityClassifier/DataSource/labeledDirectories(at:)``.
    ///
    /// - recordingFileColumn: The name of the column in an
annotation file that contains the data filenames if
    /// `trainingData` uses
    ///
``MLActivityClassifier/DataSource/directoryWithDataAndAnnotation(at:annotationFileName:timeStampColumn:labelStartTimeColumn:
labelEndTimeColumn:)``.
    ///
    /// The initializer ignores this parameter if `trainingData` uses
    ///
``MLActivityClassifier/DataSource/labeledDirectories(at:)``.
    ///
    /// - parameters: An
``MLActivityClassifier/ModelParameters-swift.struct`` instance
you use to configure the
    /// model for the training session.
public init MLActivityClassifier DataSource
String String nil
String nil
MLActivityClassifier ModelParameters
nil throws

    /// Creates an activity classifier with a training dataset represented by a data
table.
    ///
    /// Use this initializer to create an activity classifier with an
``MLDataTable``. To configure the training
    /// process, initialize the activity classifier with an
``MLActivityClassifier/ModelParameters-swift.struct``
    /// instance. For example, you can explicitly define the validation dataset
instead of allowing the model to choose
    /// a random selection of your training data. Alternatively, set
    ///
``MLActivityClassifier/ModelParameters-swift.struct/validation
Data`` to `nil` to allow the activity classifier
    /// to choose the validation data for you from among your training data. This
lets you set other parameters — like
    ///

```

```

``MLActivityClassifier/ModelParameters-swift.struct/maximumIterations`` and
    /**
``MLActivityClassifier/ModelParameters-swift.struct/batchSize`` to nondefault values.
    /**
    /// - Parameters:
    ///   - trainingData: An ``MLDataTable`` that contains a collection of sensor data that groups data entries by
    ///     activity label.
    ///   - featureColumns: The names of the columns in the data table that contain sensor data.
    ///   - labelColumn: The name of the column in the data table that contains activity labels.
    ///   - recordingFileColumn: The name of the column in the data table that contains data filenames.
    ///   - parameters: An
``MLActivityClassifier/ModelParameters-swift.struct`` instance you use to configure the
    /// model for the training session.

@available 10.15 14.0
"Use init(trainingData:parameters:)"
public init MLDataTable
String String String
MLActivityClassifier ModelParameters
nil throws

    /**
Creates an activity classifier from a training session checkpoint.

    /**
    /// - Parameters:
    ///   - checkpoint: A checkpoint from an activity classifier training session.

@available macOS 11.0
@available
@available
public init MLCheckpoint throws

    /**
Begins an asynchronous activity classifier training session with a
    /// training dataset represented by a data source.

    /**
    /// - Parameters:
    ///   - trainingData: An
``MLActivityClassifier/DataSource`` instance.
    ///   - featureColumns: The names of the columns in an annotation file that contain sensor data.
    ///   - labelColumn: The name of the column in an annotation file that contains the activity labels if `trainingData` uses
    /**
``MLActivityClassifier/DataSource/directoryWithDataAndAnnotation(at:annotationFileName:timeStampColumn:labelStartTimeColumn:`

```

```

labelEndTimeColumn:)``.
    /**
     /**
      * The initializer ignores this parameter if `trainingData` uses
     /**
````MLActivityClassifier/DataSource/labeledDirectories(at:)```.
    /**
     /**
      * - recordingFileColumn: The name of the column in an
annotation file that contains the data filenames if
      /**
       `trainingData` uses
     /**
````MLActivityClassifier/DataSource/directoryWithDataAndAnnotation(at:annotationFileName:timeStampColumn:labelStartTimeColumn:labelEndTimeColumn:)```.
    /**
     /**
      * The initializer ignores this parameter if `trainingData` uses
     /**
````MLActivityClassifier/DataSource/labeledDirectories(at:)```.
    /**
     /**
      * - parameters: An
````MLActivityClassifier/ModelParameters-swift.struct`` instance
you use to configure the
      /**
       model for the training session.
    /**
     /**
      * - sessionParameters: An
````MLTrainingSessionParameters`` instance you use to configure the training
session.
    /**
     /**
      * - Returns: An ``MLJob`` that represents the activity classifier
training
      /**
       session.
      @available macOS 11.0
      @available
      @available
      public static func train
      MLActivityClassifier DataSource           String
      String                               String
      MLActivityClassifier ModelParameters      nil
      MLTrainingSessionParameters
      throws      MLJob MLActivityClassifier

      /**
       Begins an asynchronous activity classifier training session with a
      /**
       training dataset represented by a data table.
    /**
     /**
      * - Parameters:
      /**
       - trainingData: An ``MLDataTable`` instance that contains a
collection of sensor data that groups data
      /**
       entries by activity label.
      /**
       - featureColumns: The names of the columns in the data table
that contain sensor data.

```

```

    /// - labelColumn: The name of the column in the data table that
contains activity labels.
    /// - recordingFileColumn: The name of the column in the data
table that contains data filenames.
    /// - parameters: An
``MLActivityClassifier/ModelParameters-swift.struct`` instance
you use to configure the
    /// model for the training session.
    /// - sessionParameters: An
``MLTrainingSessionParameters`` instance you use to configure the training
session.
    ///
    /// - Returns: An ``MLJob`` that represents the activity classifier
training
    /// session.

@available 11.0 14.0
"Use

train(trainingData:parameters:sessionParameters:)"
@available
@available
public static func train
    String String
    MLDataTable String
MLActivityClassifier ModelParameters
    nil
MLTrainingSessionParameters
throws MLJob MLActivityClassifier

    /// Creates an asynchronous training session for an activity classifier.
    ///
    /// - Parameters:
    ///   - trainingData: An ``MLDataTable`` instance that contains a
collection of sensor data that groups data
        /// entries by activity label.
        /// - featureColumns: The feature column names.
        /// - labelColumn: The label column name,
        /// - recordingFileColumn: The recording file column name.
        /// - parameters: An
``MLActivityClassifier/ModelParameters-swift.struct`` instance
you use to configure the
        /// model for the training session.
        /// - sessionParameters: An
``MLTrainingSessionParameters`` instance you use to configure the training
session.
    ///
    /// - Returns: An ``MLTrainingSession`` that represents the activity
classifier training session.

@available macOS 14.0
@available
@available
public static func makeTrainingSession

```

```

MLActivityClassifier DataSource           String
                      String
MLActivityClassifier ModelParameters
                      MLTrainingSessionParameters
                      throws
MLTrainingSession MLActivityClassifier

    /// Creates an asynchronous training session for an activity classifier.
    ///
    /// - Parameters:
    ///   - trainingData: An ``MLDataTable`` instance that contains a
    collection of sensor data that groups data
    ///   entries by activity label.
    ///   - featureColumns: The names of the columns in the data table
    that contain sensor data.
    ///   - labelColumn: The name of the column in the data table that
    contains activity labels.
    ///   - recordingFileColumn: The name of the column in the data
    table that contains data filenames.
    ///   - parameters: An
    ``MLActivityClassifier/ModelParameters-swift.struct`` instance
    you use to configure the
    ///   model for the training session.
    ///   - sessionParameters: An
    ``MLTrainingSessionParameters`` instance you use to configure the training
    session.
    ///
    /// - Returns: An ``MLTrainingSession`` that represents the activity
    /// classifier training session.
@available                               11.0          14.0
"Use
makeTrainingSession(trainingData:parameters:sessionParameters:
"
@available
@available
public static func makeTrainingSession
MLDataTable           String           String
                      String
MLActivityClassifier ModelParameters
                      nil
MLTrainingSessionParameters
throws      MLTrainingSession MLActivityClassifier

    /// Creates an asynchronous training session for an activity classifier by
    restoring an existing training session's
    /// state from its parameters.
    ///
    /// - Parameters:
    ///   - sessionParameters: The
    ``MLTrainingSessionParameters`` instance you used to create the training

```

```
session
    ///      using
    ///
``MLActivityClassifier/makeTrainingSession(trainingData:featureColumns:labelColumn:recordingFileColumn:parameters:sessionParameters:)``.
    ///
    /// - Returns: An ``MLTrainingSession`` that represents the activity classifier training session.
    @available macOS 11.0
@available
@available
public static func
restoreTrainingSession
MLTrainingSessionParameters throws
MLTrainingSession MLActivityClassifier

    /// Begins or continues an asynchronous activity classifier training session.
    ///
    /// - Parameters:
    ///   - session: An ``MLTrainingSession`` instance that represents the training session.
    ///
    /// - Returns: An ``MLJob`` that represents the activity classifier training session.
    @available macOS 11.0
@available
@available
public static func resume -
MLTrainingSession MLActivityClassifier throws
MLJob MLActivityClassifier

@available macOS 10.15
@available
@available
extension MLActivityClassifier : CustomStringConvertible

    /// A text representation of the activity classifier.
public var description : String get

@available macOS 10.15
@available
@available
extension MLActivityClassifier : CustomDebugStringConvertible

    /// A text representation of the activity classifier that's suitable for output during debugging.
```

```
public var debugDescription String get

@available macOS 10.15
@available
@available
extension MLActivityClassifier
CustomPlaygroundDisplayConvertible

    /// A description of the activity classifier shown in a playground.
public var playgroundDescription Any get

@available macOS 10.15
@available
@available
extension MLActivityClassifier

    /// Predict activities from new observations.
    ///
    /// - Parameters
    ///   - testingData: A data frame containing unlabeled sensor data
    samples. All samples are assumed to come from
    ///   the same recording. Feature column names used in the table should
    be consistent with those used in training.
    ///   - perWindowPrediction: A Boolean option to specify the prediction
    frequency. Default is false, and prediction
    ///   is made per sample, instead of per window.
    /// - Throws: `MLCreateError.type` if `testingData` format is
invalid.
    /// - Returns: An array of predicted class names.
@available macOS 13.0
@available
@available
public func predictions           DataFrame
                           Bool   false throws   String

    /// Predict activities from new observations.
    ///
    /// - Parameters:
    ///   - data: A MLDataTable containing unlabeled sensor data samples.
    All samples are assumed to come from
    ///   the same recording. Feature column names used in the table should
    be consistent with those used in training.
    ///   - perWindowPrediction: A Boolean option to specify the
    prediction frequency. Default is false, and prediction
    ///   is made per sample, instead of per window.
    ///
    /// - Throws: `MLCreateError.type` if `testingData` format is
invalid.
```

```
    /// - Returns: An array of predicted class names.
    @available(10.15, 14.0)
    public func predictions(Bool) throws [String] {
        return predictions(for: false)
    }

    @available(macOS 10.15)
    @available(iOS 13.0, tvOS 13.0, watchOS 6.0, *)
    extension MLActivityClassifier {
        /// A data source for an activity classifier.
        public enum DataSource {
            /// An activity classifier data source that uses a directory of directories
            /// that contain sensor data files.
            case labeledDirectories(at: URL)

            /// - Parameters:
            ///   - at: A <doc://com.apple.documentation/documentation/foundation/url> of a directory in the
            ///     file system
            ///     that contains directories, each named with an activity label for
            ///     the sensor data files.
            case annotationFile(at: URL)

            /// An activity classifier data source that uses a directory that contains
            /// sensor data files and one annotation
            /// file.
            case combined(annotationFileAt: URL, labeledDirectoriesAt: URL)
        }
    }
}
```

```

column names
    /// of the annotation file to
    ///
``MLActivityClassifier/DataSource/directoryWithDataAndAnnotation(at:annotationFileName:timeStampColumn:labelStartTimeColumn:labelEndTimeColumn:)``.
    ///
    /// - Parameters:
    /// - at: The location URL of a directory in the file system that contains sensor data files and an activity
    ///   annotation file.
    /// - annotationFileName: The name of the activity annotation file.
    /// - timeStampColumn: The name of the column that contains the timestamps for each sensor data sample.
    /// - labelStartTimeColumn: The name of the column that contains the activity's starting-time index in the data file.
    /// - labelEndTimeColumn: The name of the column that contains the activity's ending-time index in the data file.
case directoryWithDataAndAnnotation URL
        String String
        String String

    /// An activity classifier data source that uses a data frame containing sensor features and labels.
available macOS 14.0
case dataFrame DataFrame

    /// Processes the data source and returns a data frame that contains features, labels and file names.
    ///
    /// - Parameters:
    /// - featureColumns: The names of the feature columns.
    /// - labelColumn: The name of the column with the labels.
    /// - recordingFileColumn: The name of the column with the recording file names, if any.
available macOS 14.0
public func gatherAnnotatedFeatures
String String "label"
String nil throws DataFrame

    /// Generates a data table from the contents of the data source.
    ///
    /// The `labelColumn` and `recordingFileColumn` parameters are optional if the data source is
    ///
``MLActivityClassifier/DataSource/labeledDirectories(at:)``. If `nil`, the method names the data table's
    /// label column and data file column "label" and "recordingFile",

```

respectively.

```
///
/// - Parameters:
///   - featureColumns: The names of the feature columns the
method includes in the ``MLDataTable`` it
///   generates.
///   - labelColumn: The name of the label column. This
parameter must not be `nil` if the data source uses
///
``MLActivityClassifier/DataSource/directoryWithDataAndAnnotation(at:annotationFileName:timeStampColumn:labelStartTimeColumn:
labelEndTimeColumn:)``.
///   - recordingFileColumn: The name of the column with the
recording file names. This parameter must not be
///   `nil` if the data source uses
///
``MLActivityClassifier/DataSource/directoryWithDataAndAnnotation(at:annotationFileName:timeStampColumn:labelStartTimeColumn:
labelEndTimeColumn:)``.
///
/// - Returns: A new ``MLDataTable`` instance.
@available 10.15 14.0
"Use
gatherAnnotatedFeatures(featureColumns:labelColumn:recordingFi
leColumn:)"
public func labeledSensorData
String           String    nil
String    nil throws    MLDataTable

/// Generates a data table by splitting the data source into strata.
///
/// - Parameters:
///   - proportions: An array of proportions, each in the range
`[0.0, 1.0]`.
///   - seed: A seed number for the random-number generator.
///   - featureColumns: The names of the feature columns the
method includes in the data table.
///   - labelColumn: The name of the label column the methods
stratifies.
///   - recordingFileColumn: The name of the column with the
data file names.
///
/// - Returns: A new ``MLDataTable`` instance.
@available 10.15 14.0
"Use DataFrame.stratifiedSplit\(on:by:\)"
```

**public func stratifiedSplit** **Double**
Int String **String**
String String **throws**
MLDataTable

```
@available macOS 10.15
@available
@available
extension MLActivityClassifier

    /// Model training parameters that direct the training process for an
    /// activity classifier model.
    public struct ModelParameters

        /// The activity classifier's validation dataset.
        ///
        /// If you don't specify validation data, the training process
        automatically sets aside a random subset of the
        /// training data as the validation data.
        @available                         10.15          14.0
        "Use validationDataSource"
        public var validationData  MLDataTable

        /// The validation data source.
        ///
        /// If you don't specify validation data, the training process
        automatically sets aside a random subset of the
        /// training data as the validation data.
        @available macOS 14.0
        public var validation
MLActivityClassifier ModelParameters Validation

        /// The maximum number of iterations over the training data the training
        /// session uses.
        public var maximumIterations  Int

        /// The number of sequence chunks the training session uses per
        /// iteration.
        public var batchSize   Int

        /// The number of samples for each labeled activity.
        public var predictionWindowSize  Int

        /// Creates a set of activity classifier parameters that includes a
        /// validation dataset in a data source.
        ///
        /// - Parameters:
        ///   - validation: An
``MLActivityClassifier/DataSource`` instance that contains a validation
dataset.
        ///   - batchSize: The number of activity entries the training
        session uses for each of its training iterations.
        ///   - maximumIterations: The largest number of training
```



```
@available
```

```
@available
```

```
extension MLActivityClassifier
```

```
    /// Generates metrics describing the activity classifier's performance on  
    labeled activities in a data table.
```

```
    ///
```

```
    /// - Parameters:
```

```
    /// - testingData: The activity data that you provide to test this  
    model, contained in an ``MLDataTable``.
```

```
    /// - featureColumns: The names of the columns that contain the  
    sensor data.
```

```
    /// - labelColumn: The name of the column that contain the activity  
    labels.
```

```
    /// - recordingFileColumn: The name of the column that contain  
    the recording file names.
```

```
    ///
```

```
    /// - Returns: An ``MLClassifierMetrics`` instance.
```

```
@available
```

```
10.15
```

```
14.0
```

```
public func evaluation
```

```
MLDataTable
```

```
String
```

```
String
```

```
String
```

```
MLClassifierMetrics
```

```
    /// Generates metrics describing the activity classifier's performance on  
    /// labeled activities in a data source.
```

```
    ///
```

```
    /// - Parameters:
```

```
    /// - testingData: The activity data that you provide to test this  
    model, contained in an
```

```
    ///     ``MLActivityClassifier/DataSource``.
```

```
    /// - featureColumns: The names of the columns that contain sensor  
    data.
```

```
    /// - labelColumn: The name of the column that contain the activity  
    labels. The method ignores this parameter if
```

```
    ///     the data source uses a labeled directory.
```

```
    /// - recordingFileColumn: The name of the column that contain  
    the recording file names. The method ignores this
```

```
    ///     parameter if the data source uses a labeled directory.
```

```
    ///
```

```
    /// - Returns: An ``MLClassifierMetrics`` instance.
```

```
public func evaluation
```

```
MLActivityClassifier DataSource
```

```
String
```

```
String nil
```

```
String
```

```
nil MLClassifierMetrics
```

```
@available macOS 10.15
```

```
@available
```

```
@available
```

```
extension MLActivityClassifier
```

```
/// Exports the activity classifier as a Core ML model file.  
///  
/// - Parameters:  
///   - fileURL: A file-system URL.  
///   - metadata: The model's description, author, version, and license  
information.  
public func write URL  
MLModelMetadata throws  
  
/// Exports the activity classifier as a Core ML model file.  
///  
/// - Parameters:  
///   - path: A file-system path.  
///   - metadata: The model's description, author, version, and license  
information.  
public func write String  
MLModelMetadata throws  
  
@available macOS 10.15  
@available  
@available  
extension MLActivityClassifier ModelParameters  
CustomStringConvertible CustomDebugStringConvertible  
CustomPlaygroundDisplayConvertible  
  
/// A text representation of the activity-model parameters.  
public var description String get  
  
/// A text representation of the activity-model parameters that's suitable  
/// for output during debugging.  
public var debugDescription String get  
  
/// A description of the activity-model parameters shown in a playground.  
public var playgroundDescription Any get  
  
@available macOS 14.0  
@available  
@available  
extension MLActivityClassifier ModelParameters  
  
/// The source of a validation dataset for an activity classifier.  
public enum Validation  
  
    /// A validation dataset derived by randomly selecting a portion of the  
    training data.  
    case split MLSplitStrategy
```

```
    /// A validation dataset represented by a data source.  
    case dataSource MLActivityClassifier DataSource  
  
    /// An empty validation dataset that skips the model validation phase  
    after training.  
    case none  
  
  
    /// A classifier based on a collection of decision trees combined with gradient  
    boosting.  
    ///  
    /// A boosted tree classifier combines several  
    ``MLDecisionTreeClassifier`` models (a technique known as _ensemble  
    /// learning) by training each model to correct the errors of the preceding model.  
    ///  
    /// This model is useful for handling numerical and categorical features, but is less  
    suitable for sparse data such as  
    /// text.  
    @available macOS 10.14 iOS 15.0 tvOS 16.0  
    public struct MLBoostedTreeClassifier @unchecked Sendable  
  
        /// The Core ML model.  
        public var model MLModel  
  
        /// The name of the column you selected at initialization to define which  
        categories the classifier predicts.  
        ///  
        /// Changing the value of this property doesn't retrain the model or affect its  
        behavior.  
        public var targetColumn String  
  
        /// The names of the columns you selected at initialization to train the  
        classifier.  
        ///  
        /// Changing the value of this property doesn't retrain the model or affect its  
        behavior.  
        public var featureColumns String  
  
        /// The underlying parameters used when training the model.  
        public let modelParameters  
        MLBoostedTreeClassifier ModelParameters  
  
        /// Measurements of the classifier's performance on the training data set.  
        public var trainingMetrics MLClassifierMetrics get  
  
        /// Measurements of the classifier's performance on the validation data set.  
        public var validationMetrics MLClassifierMetrics get  
  
        /// Creates a boosted tree classifier.
```

```

    /**
     * - Parameters:
     *   - trainingData: The training data
     *   - targetColumn: Name of the column containing the class labels
     *   - featureColumns: Names of the columns containing feature
     * values. If `nil` all columns, other than the target
     *   - column, will be used as feature values.
     * - parameters: Model training parameters
@available macOS 12.0 iOS 15.0 tvOS 16.0
public init DataFrame String
String nil
MLBoostedTreeClassifier ModelParameters

throws

    /**
     * Creates a Boosted Tree Classifier from the feature columns in the training
     * data to predict the categories in
     *   - the target column.
    /**
     * - Parameters:
     *   - trainingData: A data table of training examples.
     *   - targetColumn: The column name for the values in the training
     * data that the classifier should predict.
     *   - featureColumns: The column names for the values in the
     * training data that the classifier uses to predict
     *   - the target value.
     * - parameters: The model parameters.
@available 10.14 13.0
    "Use DataFrame instead of MLDataTable when
initializing."
@available 15.0 16.0
    "Use DataFrame instead of MLDataTable when
initializing."
@available
public init MLDataTable
String String nil
MLBoostedTreeClassifier ModelParameters
nil throws

    /**
     * Creates a boosted tree classifier from a checkpoint.
    /**
     * - Parameter checkpoint: Training checkpoint.
     * - Throws: `MLCreateError` if the checkpoint can't be loaded.
@available macOS 12.0 iOS 15.0 tvOS 16.0
public init MLCheckpoint throws

    /**
     * Trains a boosted tree classifier.
    /**
     * If `sessionDirectory` is provided it will save training progress. If
     * there is progress already saved training

```

```

    /// will resume from the last checkpoint.
    ///
    /// - Parameters:
    ///   - trainingData: A DataTable specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
    trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
    columns to be
    ///                               used to predict the target, if not provided,
    default to use all the
    ///                               other columns in the trainingData, except the
    one specified by targetColumn
    ///   - parameters: Model training parameters. See
    `MLBoostedTreeClassifier.ModelParameters` for the defaults.
    ///   - sessionParameters: Training session parameters. See
    `MLTrainingSessionParameters` for the defaults.
    ///
    /// - Returns: A MLJob that can be used to observe training progress.
available           12.0          14.0
available           15.0          17.0
available           16.0          17.0
public static func train           MLDataTable
                    String           String      nil
                    MLBoostedTreeClassifier ModelParameters

MLTrainingSessionParameters
throws   MLJob MLBoostedTreeClassifier

    /// Trains a boosted tree classifier.
    ///
    /// If sessionDirectory is provided it will save training progress. If
    there is progress already saved training
    /// will resume from the last checkpoint.
    ///
    /// - Parameters:
    ///   - trainingData: A DataFrame specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
    trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
    columns to be
    ///                               used to predict the target, if not provided,
    default to use all the
    ///                               other columns in the trainingData, except the
    one specified by targetColumn
    ///   - parameters: Model training parameters. See
    `MLBoostedTreeClassifier.ModelParameters` for the defaults.
    ///   - sessionParameters: Training session parameters. See
    `MLTrainingSessionParameters` for the defaults.
    ///
    /// - Returns: A MLJob that can be used to observe training progress.
available macOS 12.0 iOS 15.0 tvOS 16.0

```

```

public static func train DataFrame
    String String nil
    MLBoostedTreeClassifier ModelParameters

MLTrainingSessionParameters
throws MLJob MLBoostedTreeClassifier

/// Creates or restores a training session.
///
/// - Parameters:
///   - trainingData: A DataTable specifying training data.
///   - targetColumn: A String specifying the target column name in the
trainingData
///   - featureColumns: An optional list of Strings specifying feature
columns to be
///           used to predict the target, if not provided, default to use
all the
///           other columns in the trainingData, except the one
specified by targetColumn
///   - parameters: Model training parameters. See
`MLBoostedTreeClassifier.ModelParameters` for the defaults.
///   - sessionParameters: Training session parameters. See
`MLTrainingSessionParameters` for the defaults.
///
/// - Returns: A `MLTrainingSession` that can be used to start or
resume training.
@available 12.0 14.0
@available 15.0 17.0
@available 16.0 17.0
public static func makeTrainingSession String
MLDataTable String String
nil MLBoostedTreeClassifier ModelParameters
MLTrainingSessionParameters
throws
MLTrainingSession MLBoostedTreeClassifier

/// Creates or restores a training session.
///
/// - Parameters:
///   - trainingData: A `DataFrame` specifying training data.
///   - targetColumn: A String specifying the target column name in the
trainingData
///   - featureColumns: An optional list of Strings specifying feature
columns to be used to predict the target, if
///           not provided, default to use all the other columns in the trainingData,
except the one specified by
///           targetColumn.
///   - parameters: Model training parameters. See
`MLBoostedTreeClassifier.ModelParameters` for the defaults.
///   - sessionParameters: Training session parameters. See

```

```

`MLTrainingSessionParameters` for the defaults.

///
/// - Returns: A `MLTrainingSession` that can be used to start or
resume training.
 @available macOS 12.0 iOS 15.0 tvOS 16.0
 public static func makeTrainingSession
 DataFrame String String
 nil MLBoostedTreeClassifier ModelParameters
 MLTrainingSessionParameters
 throws
 MLTrainingSession MLBoostedTreeClassifier

    /// Restores an existing training session.
    ///
    /// - Parameters:
    ///   - sessionParameters: Training session parameters. The
`sessionDirectory` parameter is required.
    ///
    /// - Returns: A `MLTrainingSession` that can be used to resume
training.
 @available macOS 12.0 iOS 15.0 tvOS 16.0
 public static func
 restoreTrainingSession
 MLTrainingSessionParameters throws
 MLTrainingSession MLBoostedTreeClassifier

    /// Resumes a training session from the last checkpoint if available.
    ///
    /// If there are no resumable checkpoints training starts over from the
beginning.
    ///
    /// - Parameter session: Loaded or new training session.
    ///
    /// - Returns: A `MLJob` that can be used to observe training progress.
 @available macOS 12.0 iOS 15.0 tvOS 16.0
 public static func resume
 _
 MLTrainingSession MLBoostedTreeClassifier throws
 MLJob MLBoostedTreeClassifier

    /// Predicts a column of labels for the given testing data.
 @available macOS 12.0 iOS 15.0 tvOS 16.0
 public func predictions
 DataFrame throws
 AnyColumn

    /// Classifies the provided data into the target categories.
    ///
    /// - Parameters:
    ///   - data: The data you want the model to classify.
    ///
    /// - Returns: A column of labels predicted by the classifier.

```

```
@available 10.14 13.0
    "Use DataFrame instead of MLDataTable."
@available 15.0 16.0
    "Use DataFrame instead of MLDataTable."
@available
public func predictions MLDataTable throws
MLUntypedColumn

    /// Evaluates the classifier on the provided labeled data.
    ///
    /// Evaluation should be done on a testing data set that the model has not
    seen as part of the training or
    /// validation data sets. The data should have feature columns with identical
    name and type to the
    /// training data, as well as a labels column with the same name.
    ///
    /// - Parameters:
    /// - labeledData: A `DataFrame` to evaluate the trained model on.
    ///
    /// - Returns: Metrics that describe the classification errors
    /// (` `MLClassifierMetrics/classificationError` `), the precision
    and recall
    /// percentages (` `MLClassifierMetrics/precisionRecall` `), and
    a table that
    /// describes how labels were misapplied
    (` `MLClassifierMetrics/confusion` `)
    /// on the provided data.
@available macOS 12.0 iOS 15.0 tvOS 16.0
public func evaluation DataFrame
MLClassifierMetrics

    /// Evaluates the classifier on the provided labeled data.
    ///
    /// Evaluation should be done on a testing data set that the model has not
    seen as part of the training or
    /// validation data sets. The data should have feature columns with identical
    name and type to the
    /// training data, as well as a labels column with the same name.
    ///
    /// - Parameters:
    /// - labeledData: An `MLDataTable` to evaluate the trained model
    on.
    ///
    /// - Returns: Metrics that describe the classification errors
    /// (` `MLClassifierMetrics/classificationError` `), the precision
    and recall
    /// percentages (` `MLClassifierMetrics/precisionRecall` `), and
    a table that
    /// describes how labels were misapplied
    (` `MLClassifierMetrics/confusion` `)
    /// on the provided data.
```

```

@available 10.14 13.0
    "Use DataFrame instead of MLDataTable."
@available 15.0 16.0
    "Use DataFrame instead of MLDataTable."
@available
public func evaluation MLDataTable
MLClassifierMetrics

/// Exports a Core ML model file for use in your app.
public func write URL
MLModelMetadata nil throws

/// Exports a Core ML model file for use in your app.
public func write String
MLModelMetadata nil throws

extension MLBoostedTreeClassifier

/// Parameters that affect the process of training a model.
@available macOS 10.14 iOS 15.0 tvOS 16.0
public struct ModelParameters

    /// Validation data represented as a `MLDataTable`.
    ///
    /// - Note: Setting this to `nil` means that the training data will be
    automatically split for
    /// validation. Setting it to an empty table means to not use a
    validation set.
    @available 10.14
10.15      "Use the validation property instead."
    @available 15.0 16.0
    "Use the validation property instead."
    @available
public var validationData MLDataTable

    /// Validation data.
    ///
    /// The default is `split(strategy: .automatic)`, which
    automatically generates the validation
    /// dataset from 0% to 10% of the training dataset.
    @available macOS 10.15 iOS 15.0 tvOS 16.0
    public var validation
MLBoostedTreeClassifier ModelParameters ValidationData

    public var maxDepth Int
    public var maxIterations Int
    public var minLossReduction Double

```

```

public var minChildWeight Double
public var randomSeed Int

/// Must be in the range (0, 1).
public var stepSize Double

/// Validation data must be specified for an early stop.
public var earlyStoppingRounds Int

/// Must be in the range (0, 1).
public var rowSubsample Double

/// Must be in the range (0, 1).
public var columnSubsample Double

@available macOS 10.15 iOS 15.0 tvOS 16.0
public init
MLBoostedTreeClassifier ModelParameters ValidationData
                           Int 6
                           Int 10
                           Double 0.1
                           Int 42
Double 0.3
Double 1.0
                           Int nil
                           Double 1.0

/// Creates a new set of parameters defining how a boosted tree
classifier should be built.
///
/// - Parameters:
/// - validationData: The dataset used to monitor how well the
model is generalizing.
///
/// The default value is `nil` which will use an automatically
sampled validation set.
///
/// - maxDepth: The maximum depth of the tree. Must be a value
of at least 1.
///
/// The default value is 6.
///
/// - maxIterations: The maximum number of passes through
the data. Each iteration creates an extra tree.
///
/// The default value is 10.
///
/// - minLossReduction: The minimum amount of reduction in
the loss function that is required to make another
/// split to the data. Larger values help prevent overfitting.
///

```

```

    /**
     *      The default value is 0.
    */
    /**
     *      - minChildWeight: Determines the minimum weight of each
leaf node of the tree. Larger values help prevent
     *          overfitting.
    */
    /**
     *      The default value is 0.1.
    */
    /**
     *      - randomSeed: A seed for internal random operations. Set this
value to ensure reproducible results.
    */
    /**
     *      The default value is 42.
    */
    /**
     *      - stepSize: The shrinkage used to decrease the prediction
weight of each learner. The smaller the step
     *          size the more conservative the _boosting_ process will be.
    */
    /**
     *      The default value is 0.3.
    */
    /**
     *      - earlyStoppingRounds: If the validation accuracy does not
improve after the specified number of rounds
     *          training will stop.
    */
    /**
     *      - rowSubsample: Select the specified ratio from the training
set to grow each tree. For example, a value
     *          of 0.5 means each tree is trained on half the data. This
technique is known as _bagging_.
    */
    /**
     *      The default value is 1.0.
    */
    /**
     *      - columnSubsample: Select the specified ratio of columns
from the training set to use when growing each
     *          tree. Similar to row subsampling, this can be used to prevent
overfitting.
    */
    /**
     *      The default value is 1.0.
    */
    /**
     *      @available 10.15
     *          "Use the validation property instead."
     *      @available 15.0
     *          "Use the validation property instead."
     *      @available
     *          public init
     *              MLDataTable
     *                  Int 6
     *                      Int 10
     *                          Double 0.1
     *                              Double 0.3
     *                                  Double 1.0
     *                                      Double 1.0
     *  Double 42
     *  Int nil
     *  Double 1.0
     */

```

`@available macOS 10.14 iOS 15.0 tvOS 16.0`

```
extension MLBoostedTreeClassifier CustomStringConvertible  
CustomDebugStringConvertible  
CustomPlaygroundDisplayConvertible
```

```
    /// A text representation of the boosted tree classifier.  
    public var description String get  
  
    /// A text representation of the boosted tree classifier that's suitable for  
    /// output during debugging.  
    public var debugDescription String get  
  
    /// A description of the boosted tree classifier shown in a playground.  
    public var playgroundDescription Any get
```

```
extension MLBoostedTreeClassifier ModelParameters
```

```
    /// Values for specifying validation data.  
    @available macOS 10.15 iOS 15.0 tvOS 16.0  
    public enum ValidationData  
  
        /// Generate validation data by splitting the training dataset. This is the  
        default.  
        case split MLSplitStrategy  
  
        /// Set validation data from the MLDataTable provided.  
        @available 10.15 14.0  
        @available 15.0 17.0  
        @available 16.0 17.0  
        case table MLDataTable  
  
        /// Validation data provided in a DataFrame.  
        @available macOS 12.0 iOS 15.0 tvOS 16.0  
        case dataFrame DataFrame  
  
        /// Do not set validation data.  
        case none
```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0  
extension MLBoostedTreeClassifier ModelParameters  
CustomStringConvertible CustomDebugStringConvertible  
CustomPlaygroundDisplayConvertible
```

```
    /// A text representation of the model parameters for a boosted tree classifier.  
    public var description String get  
  
    /// A text representation of the model parameters for a boosted tree classifier  
    that's suitable for output during
```

```
/// debugging.
public var debugDescription String get

/// A description of the model parameters for a boosted tree classifier shown
in a playground.
public var playgroundDescription Any get

/// A regressor based on a collection of decision trees combined with gradient
boosting.
@available macOS 10.14 iOS 15.0 tvOS 16.0
public struct MLBoostedTreeRegressor @unchecked Sendable

/// The Core ML model.
public var model MLModel

/// The name of the column you selected at initialization to define which
feature the regressor predicts.
///
/// Changing the value of this property doesn't retrain the model or affect its
behavior.
public var targetColumn String

/// The names of the columns you selected at initialization to train the
regressor.
///
/// Changing the value of this property doesn't retrain the model or affect its
behavior.
public var featureColumns String

/// The underlying parameters used when training the model.
public let modelParameters
MLBoostedTreeRegressor ModelParameters

/// Measurements of the regressor's performance on the training data set.
public var trainingMetrics MLRegressorMetrics get

/// Measurements of the regressor's performance on the validation data set.
public var validationMetrics MLRegressorMetrics get

/// Creates a boosted tree regressor.
///
/// - Parameters:
///   - trainingData: The training data
///   - targetColumn: Name of the column containing the target values
///   - featureColumns: Names of the columns containing feature
values. If `nil` all columns, other than the target
///   column, will be used as feature values.
///   - parameters: Model training parameters
@available macOS 12.0 iOS 15.0 tvOS 16.0
```

```
public init           DataFrame          String
                    String      nil
MLBoostedTreeRegressor ModelParameters

throws

    /// Creates a Boosted Tree Regressor from the feature columns in the
training data to predict the values in the
    /// target column.
    ///
    /// - Parameters:
    ///   - trainingData: A data table of training examples.
    ///   - targetColumn: The column name for the values in the training
data the regressor should predict.
    ///   - featureColumns: The column names for the values in the
training data that the regressor uses to predict the
    ///     target value.
    ///   - parameters: The model parameters.
@available           10.14           13.0
    "Use DataFrame instead of MLDataTable when
initializing."
@available           15.0           16.0
    "Use DataFrame instead of MLDataTable when
initializing."
@available
public init           MLDataTable
String                String      nil
MLBoostedTreeRegressor ModelParameters
                                nil    throws

    /// Creates a boosted tree regressor from a checkpoint.
    ///
    /// - Parameter checkpoint: Training checkpoint.
    /// - Throws: `MLCreateError` if the checkpoint can't be loaded.
@available macOS 12.0 iOS 15.0 tvOS 16.0
public init           MLCheckpoint throws

    /// Trains a boosted tree regressor.
    ///
    /// If `sessionDirectory` is provided it will save training progress. If
there is progress already saved training
    /// will resume from the last checkpoint.
    ///
    /// - Parameters:
    ///   - trainingData: A DataTable specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
columns to be
    ///
                                used to predict the target, if not provided,
```

default to use all the other columns in the trainingData, except the one specified by targetColumn

```

    /**
     * - parameters: Model training parameters. See
     * `MLBoostedTreeRegressor.ModelParameters` for the defaults.
     * - sessionParameters: Training session parameters. See
     * `MLTrainingSessionParameters` for the defaults.
    */
    /**
     * - Returns: A `MLJob` that can be used to observe training progress.
    
```

<b>@available</b>	12.0	14.0
<b>@available</b>	15.0	17.0
<b>@available</b>	16.0	17.0

```

public static func train
    String           DataFrame
    String           nil
    MLBoostedTreeRegressor ModelParameters

```

```

        MLTrainingSessionParameters
        throws

```

```

MLJob MLBoostedTreeRegressor

```

```

    /**
     * Trains a boosted tree regressor.
    */
    /**
     * If `sessionDirectory` is provided it will save training progress. If
     * there is progress already saved training
     * will resume from the last checkpoint.
    */
    /**
     * - Parameters:
     *   - trainingData: A `DataFrame` specifying training data.
     *   - targetColumn: A String specifying the target column name in the
     *     trainingData
     *   - featureColumns: An optional list of Strings specifying feature
     *     columns to be
     *     used to predict the target, if not provided,
     * default to use all the
     *   other columns in the trainingData, except the
     * one specified by targetColumn
     *   - parameters: Model training parameters. See
     * `MLBoostedTreeRegressor.ModelParameters` for the defaults.
     *   - sessionParameters: Training session parameters. See
     * `MLTrainingSessionParameters` for the defaults.
    */
    /**
     * - Returns: A `MLJob` that can be used to observe training progress.
    
```

<b>@available</b> macOS 12.0	iOS 15.0	tvOS 16.0
------------------------------	----------	-----------

```

public static func train
    String           DataFrame
    String           nil
    MLBoostedTreeRegressor ModelParameters

```

```

        MLTrainingSessionParameters
        throws

```

```

MLJob MLBoostedTreeRegressor

```

```

    /// Creates or restores a training session.
    ///
    /// - Parameters:
    ///   - trainingData: A DataTable specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
    trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
    columns to be
    ///           used to predict the target, if not provided, default to use
    all the
    ///           other columns in the trainingData, except the one
    specified by targetColumn
    ///   - parameters: Model training parameters. See
    `MLBoostedTreeRegressor.ModelParameters` for the defaults.
    ///   - sessionParameters: Training session parameters. See
    `MLTrainingSessionParameters` for the defaults.
    ///
    /// - Returns: A `MLTrainingSession` that can be used to start or
    resume training.
    @available(macOS 12.0, iOS 15.0, tvOS 16.0)
    @available(macOS 15.0, iOS 17.0, tvOS 17.0)
    @available(macOS 16.0, iOS 17.0, tvOS 17.0)
    public static func makeTrainingSession(
        MLDATAtable: String,
        nil: MLBoostedTreeRegressor,
        ModelParameters: MLTrainingSessionParameters
    ) throws
    MLTrainingSession

```

```

    /// Creates or restores a training session.
    ///
    /// - Parameters:
    ///   - trainingData: A `DataFrame` specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
    trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
    columns to be
    ///           used to predict the target, if not provided, default to use
    all the
    ///           other columns in the trainingData, except the one
    specified by targetColumn
    ///   - parameters: Model training parameters. See
    `MLBoostedTreeRegressor.ModelParameters` for the defaults.
    ///   - sessionParameters: Training session parameters. See
    `MLTrainingSessionParameters` for the defaults.
    ///
    /// - Returns: A `MLTrainingSession` that can be used to start or
    resume training.
    @available(macOS 12.0, iOS 15.0, tvOS 16.0)

```

```
public static func makeTrainingSession  
DataFrame String String  
nil MLBoostedTreeRegressor ModelParameters  
  
MLTrainingSessionParameters  
throws  
MLTrainingSession MLBoostedTreeRegressor  
  
/// Restores an existing training session.  
///  
/// - Parameters:  
///   - sessionParameters: Training session parameters. The  
`sessionDirectory` parameter is required.  
///  
/// - Returns: A `MLTrainingSession` that can be used to resume  
training.  
@available macOS 12.0 iOS 15.0 tvOS 16.0  
public static func  
restoreTrainingSession  
MLTrainingSessionParameters throws  
MLTrainingSession MLBoostedTreeRegressor  
  
/// Resumes a training session from the last checkpoint if available.  
///  
/// If there are no resumable checkpoints training starts over from the  
beginning.  
///  
/// - Parameter session: Loaded or new training session.  
///  
/// - Returns: A `MLJob` that can be used to observe training progress.  
@available macOS 12.0 iOS 15.0 tvOS 16.0  
public static func resume  
MLTrainingSession MLBoostedTreeRegressor throws  
MLJob MLBoostedTreeRegressor  
  
/// Predicts a column of labels for the given testing data.  
@available macOS 12.0 iOS 15.0 tvOS 16.0  
public func predictions DataFrame throws  
AnyColumn  
  
/// Predicts the target value from the provided data.  
///  
/// - Parameters:  
///   - data: The data you want the model to make predictions from.  
///  
/// - Returns: A column of values predicted by the regressor.  
@available 10.14 13.0  
"Use DataFrame instead of MLDataTable."  
@available 15.0 16.0  
"Use DataFrame instead of MLDataTable."
```

```
@available
public func predictions MLDataTable throws
MLUntypedColumn

    /// Evaluates the classifier on the provided labeled data.
    ///
    /// Evaluation should be done on a testing data set that the model has not
    seen as part of the training or
    /// validation data sets. The data should have feature columns with identical
    name and type to the
    /// training data, as well as a labels column with the same name.
    ///
    /// - Parameters:
    /// - labeledData: A `DataFrame` to evaluate the trained model on.
    ///
    /// - Returns: Metrics that describe the maximum error
    /// (` `MLRegressorMetrics/maximumError` `) or the average error
    /// (` `MLRegressorMetrics/rootMeanSquaredError` `).
@available macOS 12.0 iOS 15.0 tvOS 16.0
public func evaluation DataFrame
MLRegressorMetrics

    /// Evaluates the classifier on the provided labeled data.
    ///
    /// Evaluation should be done on a testing data set that the model has not
    seen as part of the training or
    /// validation data sets. The data should have feature columns with identical
    name and type to the
    /// training data, as well as a labels column with the same name.
    ///
    /// - Parameters:
    /// - labeledData: An `MLDataTable` to evaluate the trained model
    on.
    ///
    /// - Returns: Metrics that describe the maximum error
    /// (` `MLRegressorMetrics/maximumError` `) or the average error
    /// (` `MLRegressorMetrics/rootMeanSquaredError` `).
@available 10.14 13.0
    "Use DataFrame instead of MLDataTable."
@available 15.0 16.0
    "Use DataFrame instead of MLDataTable."
@available
public func evaluation MLDataTable
MLRegressorMetrics

    /// Exports a Core ML model file for use in your app.
public func write URL
MLModelMetadata nil throws

    /// Exports a Core ML model file for use in your app.
```

```
public func write          String
MLModelMetadata    nil  throws

extension MLBoostedTreeRegressor

    /// Parameters that affect the process of training a model.
    @available macOS 10.14  iOS 15.0  tvOS 16.0
    public struct ModelParameters

        /// Validation data represented as a `MLDataTable`.
        ///
        /// - Note: Setting this to `nil` means that the training data will be
        automatically split for
        /// validation. Setting it to an empty table means to not use a
        validation set.
        @available             10.14           11.0
        "Use the validation property instead."
        @available             15.0            16.0
        "Use the validation property instead."
        @available
        public var validationData  MLDataTable

        /// Validation data.
        ///
        /// The default is `split(strategy: .automatic)`, which
        automatically generates the validation
        /// dataset from 0% to 10% of the training dataset.
        @available macOS 11.0  iOS 15.0  tvOS 16.0
        public var validation
MLBoostedTreeRegressor ModelParameters ValidationData

        public var maxDepth  Int
        public var maxIterations  Int
        public var minLossReduction  Double
        public var minChildWeight  Double
        public var randomSeed  Int
        /// Must be in the range (0, 1).
        public var stepSize  Double
        /// Validation data must be specified for an early stop.
        public var earlyStoppingRounds  Int
        /// Must be in the range (0, 1).
        public var rowSubsample  Double
```

```

    /// Must be in the range (0, 1).
    public var columnSubsample Double

    @available(macOS 11.0, iOS 15.0, tvOS 16.0)
    public init
        MLBoostedTreeRegressor ModelParameters ValidationData
            Int 6                      Int 10
        Double 0                     Double 0.1
        42                         Double 1.0
   Int nil
   Double 1.0

    /// Creates a new set of parameters.
    ///
    /// - Parameters:
    ///   - validationData: The dataset used to monitor how well the
    model is generalizing.
    ///
    ///   The default value is `nil` which will use an automatically
    sampled validation set.
    ///
    ///   - maxDepth: The maximum depth of the tree. Must be a value
    of at least 1.
    ///
    ///   The default value is 6.
    ///
    ///   - maxIterations: The maximum number of passes through
    the data. Each iteration creates an extra tree.
    ///
    ///   The default value is 10.
    ///
    ///   - minLossReduction: The minimum amount of reduction in
    the loss function that is required to make another
    ///   split to the data. Larger values help prevent overfitting.
    ///
    ///   The default value is 0.
    ///
    ///   - minChildWeight: Determines the minimum weight of each
    leaf node of the tree. Larger values help prevent
    ///   overfitting.
    ///
    ///   The default value is 0.1.
    ///
    ///   - randomSeed: A seed for internal random operations. Set this
    value to ensure reproducible results.
    ///
    ///   The default value is 42.
    ///
    ///   - stepSize: The size used to decrease the prediction weight
    of each learner. The smaller the step size

```

```

    /**
     *      the more conservative the boosting process will be.
    */
    /**
     *      The default value is 0.3.
    */
    /**
     *      - earlyStoppingRounds: If the validation accuracy does not
     *      improve after the specified number of rounds
     *      training will stop.
    */
    /**
     *      - rowSubsample: Select the specified ratio from the training
     *      set to grow each tree. For example, a value
     *      of 0.5 means each tree is trained on half the data. This
     *      technique is known as _bagging_.
    */
    /**
     *      The default value is 1.0.
    */
    /**
     *      - columnSubsample: Select the specified ratio of columns
     *      from the training set to use when growing each
     *      tree. Similar to row subsampling, this can be used to prevent
     *      overfitting.
    */
    /**
     *      The default value is 1.0.
    */
    @available(macOS 10.14, iOS 11.0)
    "Use the validation property instead."
    @available(iOS 15.0, tvOS 16.0)
    "Use the validation property instead."
    @available
    public init(MLDataTable, nil)
        Int 6
        Double 0.1
        Double 0.3
        Double 1.0
Double 0
Int nil
Int 1.0

```

```

@available(macOS 10.14, iOS 15.0, tvOS 16.0)
extension MLBoostedTreeRegressor : CustomStringConvertible,
    CustomDebugStringConvertible, CustomPlaygroundDisplayConvertible

```

```

    /**
     * A text representation of the boosted tree regressor.
    */
    public var description: String { get }

```

```

    /**
     * A text representation of the boosted tree regressor that's suitable for
     * output during debugging.
    */
    public var debugDescription: String { get }

```

```

    /**
     * A description of the boosted tree regressor shown in a playground.
    */
    public var playgroundDescription: Any { get }

```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLBoostedTreeRegressor ModelParameters
CustomStringConvertible CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of the model parameters for a boosted tree
regressor.
public var description String get

    /// A text representation of the model parameters for a boosted tree
regressor that's suitable for output during
/// debugging.
public var debugDescription String get

    /// A description of the model parameters for a boosted tree regressor shown
in a playground.
public var playgroundDescription Any get

extension MLBoostedTreeRegressor ModelParameters

    /// Values for specifying validation data.
@available macOS 11.0 iOS 15.0 tvOS 16.0
public enum ValidationData

    /// Generate validation data by splitting the training dataset. This is the
default.
case split MLSplitStrategy

    /// Set validation data from the MLDataTable provided.
@available 11.0 14.0
@available 15.0 17.0
@available 16.0 17.0
case table MLDataTable

    /// Set validation data from the DataFrame provided.
@available macOS 13.0 iOS 16.0 tvOS 16.0
case dataFrame DataFrame

    /// Do not set validation data.
case none

/// A location within a bounding box that an annotation's coordinates use as
/// their reference point.
@available macOS 10.15
@available
@available
public enum MLBoundingBoxAnchor Sendable
```

```
/// An anchor at the bounding box's center point.
case center

/// An anchor at the bounding box's top-left corner.
case topLeft

/// An anchor at the bounding box's bottom-left corner.
case bottomLeft

/// Returns a Boolean value indicating whether two values are equal.
///
/// Equality is the inverse of inequality. For any values `a` and `b`,
/// `a == b` implies that `a != b` is `false`.
///
/// - Parameters:
///   - lhs: A value to compare.
///   - rhs: Another value to compare.
public static func MLBoundingBoxAnchor
MLBoundingBoxAnchor Bool

/// Hashes the essential components of this value by feeding them into the
/// given hasher.
///
/// Implement this method to conform to the `Hashable` protocol. The
/// components used for hashing must be the same as the components
/// compared
/// in your type's `==` operator implementation. Call
`hasher.combine(_:)`
/// with each of these components.
///
/// - Important: In your implementation of `hash(into:)`,  

/// don't call `finalize()` on the `hasher` instance provided,  

/// or replace it with a different instance.  

/// Doing so may become a compile-time error in the future.
///
/// - Parameter hasher: The hasher to use when combining the
/// components
/// of this instance.
public func hash inout Hasher

/// The hash value.
///
/// Hash values are not guaranteed to be equal across different executions of
/// your program. Do not save hash values to use during a future execution.
///
/// - Important: `hashValue` is deprecated as a `Hashable`  

/// requirement. To
/// conform to `Hashable`, implement the `hash(into:)` requirement  

instead.
```

```
/// The compiler provides an implementation for `hashValue` for you.
public var hashValue Int get

@available macOS 10.15
@available
@available
extension MLBoundingBoxAnchor : Equatable

@available macOS 10.15
@available
@available
extension MLBoundingBoxAnchor : Hashable

/// The location within an image that an annotation's coordinates use as their
/// origin.
@available macOS 10.15
@available
@available
public enum MLBoundingBoxCoordinatesOrigin : Sendable

    /// An origin at the image's top-left corner.
    case topLeft

    /// An origin at the image's bottom-left corner.
    case bottomLeft

    /// Returns a Boolean value indicating whether two values are equal.
    ///
    /// Equality is the inverse of inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is `false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to compare.
    public static func == (MLBoundingBoxCoordinatesOrigin lhs,
                           MLBoundingBoxCoordinatesOrigin rhs) Bool

    /// Hashes the essential components of this value by feeding them into the
    /// given hasher.
    ///
    /// Implement this method to conform to the `Hashable` protocol. The
    /// components used for hashing must be the same as the components
    compared
        /// in your type's `==` operator implementation. Call
        `hasher.combine(_:)`
        /// with each of these components.
```

```
///  
/// - Important: In your implementation of `hash(into:)`,  
///   don't call `finalize()` on the `hasher` instance provided,  
///   or replace it with a different instance.  
///   Doing so may become a compile-time error in the future.  
///  
/// - Parameter hasher: The hasher to use when combining the  
components  
///   of this instance.  
public func hash           inout Hasher  
  
/// The hash value.  
///  
/// Hash values are not guaranteed to be equal across different executions of  
/// your program. Do not save hash values to use during a future execution.  
///  
/// - Important: `hashValue` is deprecated as a `Hashable`  
requirement. To  
///   conform to `Hashable`, implement the `hash(into:)` requirement  
instead.  
///   The compiler provides an implementation for `hashValue` for you.  
public var hashValue Int get
```

```
@available macOS 10.15  
@available  
@available  
extension MLBoundingBoxCoordinatesOrigin Equatable
```

```
@available macOS 10.15  
@available  
@available  
extension MLBoundingBoxCoordinatesOrigin Hashable
```

```
/// The units a bounding box annotation uses to define its position and size.  
///  
/// All bounding box annotations in an annotation file must use the same units  
/// for their coordinates and size. See  
///  
``MLObjectDetector/AnnotationType/boundingBox(units:origin:anc  
hor:)``.  
available macOS 10.15  
available  
available  
public enum MLBoundingBoxUnits Sendable  
  
/// A unit of measurement in pixels for an image.  
case pixel
```

```

/// A unit of measurement as a portion of an image's overall width or
/// height.
case normalized

/// Returns a Boolean value indicating whether two values are equal.
///
/// Equality is the inverse of inequality. For any values `a` and `b`,
/// `a == b` implies that `a != b` is `false`.
///
/// - Parameters:
///   - lhs: A value to compare.
///   - rhs: Another value to compare.
public static func MLBoundingBoxUnits
MLBoundingBoxUnits Bool

/// Hashes the essential components of this value by feeding them into the
/// given hasher.
///
/// Implement this method to conform to the `Hashable` protocol. The
/// components used for hashing must be the same as the components
/// compared
/// in your type's `==` operator implementation. Call
`hasher.combine(_:)`
/// with each of these components.
///
/// - Important: In your implementation of `hash(into:)`,  

///   don't call `finalize()` on the `hasher` instance provided,  

///   or replace it with a different instance.  

/// Doing so may become a compile-time error in the future.
///
/// - Parameter hasher: The hasher to use when combining the
/// components
/// of this instance.
public func hash inout Hasher

/// The hash value.
///
/// Hash values are not guaranteed to be equal across different executions of
/// your program. Do not save hash values to use during a future execution.
///
/// - Important: `hashValue` is deprecated as a `Hashable`  

/// requirement. To
/// conform to `Hashable`, implement the `hash(into:)` requirement  

/// instead.
/// The compiler provides an implementation for `hashValue` for you.
public var hashValue Int get

```

**@available macOS 10.15**

```
@available
@available
extension MLBoundingBoxUnits Equatable

@available macOS 10.15
@available
@available
extension MLBoundingBoxUnits Hashable

/// The state of a model's asynchronous training session at a specific point in
/// time during the feature extraction or training phase.
@available macOS 11.0 iOS 15.0 tvOS 16.0
public struct MLCheckpoint Codable

    /// The location of the checkpoint in the file system.
    public var url URL

    /// The training session's phase when it created the checkpoint.
    public var phase MLPhase

    /// The iteration number of a training session's phase when it created the
    /// checkpoint.
    public var iteration Int

    /// The time when the training session created the checkpoint.
    public var date Date

    /// Measurements of the model's performance at the time the session saved
    /// the checkpoint.
    public var metrics MLProgress Metric Any

    /// Creates a new checkpoint by decoding from the decoder.
    ///
    /// - Parameters:
    ///   - decoder: The decoder to read data from.
    public init any Decoder throws

    /// Encodes the checkpoint into the encoder.
    ///
    /// - Parameters:
    ///   - encoder: The encoder to write data to.
    public func encode any Encoder throws

/// A model you train to classify data into discrete categories.
///
/// Use an ``MLClassifier`` to train a general-purpose model to recognize
```

```
categories.  
///  
/// For example, you can create a classifier that predicts whether a sports team is  
likely to win or lose its next game  
/// by training it with these inputs:  
///  
/// - The team's win-loss ratio  
/// - The team's game locations  
///  
/// - Important: When working with image or natural language data, don't use  
``MLClassifier``. Instead, use the  
/// `MLImageClassifierBuilder` or one of the Natural Language models  
(``MLTextClassifier`` or ``MLWordTagger``).  
///  
/// When you create an ``MLClassifier``, Create ML inspects your data and  
automatically chooses a specific classifier  
/// (see _Supporting Classifier Types_).  
@available macOS 10.14  
@available  
@available  
public enum MLClassifier : Sendable  
  
    /// A classifier based on a collection of decision trees combined with gradient  
boosting.  
    ///  
    /// Don't create an ``MLClassifier`` using one of its enumeration cases.  
Use the classifier's initializer instead.  
    case boostedTree MLBoostedTreeClassifier  
  
    /// A classifier that predicts the target by creating rules to split the data.  
    ///  
    /// Don't create an ``MLClassifier`` using one of its enumeration cases.  
Use the classifier's initializer instead.  
    case decisionTree MLDecisionTreeClassifier  
  
    /// A classifier based on a collection of decision trees trained on subsets of  
the data.  
    ///  
    /// Don't create an ``MLClassifier`` using one of its enumeration cases.  
Use the classifier's initializer instead.  
    case randomForest MLRandomForestClassifier  
  
    /// A classifier that predicts a discrete target value as a function of data  
features.  
    ///  
    /// Don't create an ``MLClassifier`` using one of its enumeration cases.  
Use the classifier's initializer instead.  
    case logisticRegression MLLogisticRegressionClassifier  
  
    /// A classifier that predicts a binary target value by maximizing the  
separation between categories.
```

```
///  
/// Don't create an ``MLClassifier`` using one of its enumeration cases.  
Use the classifier's initializer instead.  
@available 10.14 14.0  
case supportVector MLSupportVectorClassifier  
  
/// Creates a classifier.  
///  
/// - Parameters:  
///   - trainingData: The training data  
///   - targetColumn: Name of the column containing the class labels  
///   - featureColumns: Names of the columns containing feature  
values. If `nil` all columns, other than the target  
///   column, will be used as feature values.  
@available macOS 12.0  
@available  
@available  
public init DataFrame String  
      String nil throws  
  
/// Creates a classifier from the feature columns in the training data to predict  
the categories in the target  
/// column.  
///  
/// To view details about the supporting model chosen by the  
``MLClassifier``, print the model's description:  
///  
/// ````swift  
/// print(model)  
/// ````  
///  
/// - Parameters:  
///   - trainingData: A data table of training examples.  
///   - targetColumn: The column name for the values in the training  
data that the classifier should predict.  
///   - featureColumns: The column names for the values in the  
training data that the classifier uses to predict  
///   the target value.  
@available 10.14 13.0  
"Use DataFrame instead of MLDataTable when  
initializing."  
@available  
@available  
public init MLDataTable  
      String nil throws  
  
@available macOS 12.0  
@available  
@available  
public func predictions DataFrame throws
```

## AnyColumn

```
    /// Classifies the provided data into the target categories.  
    ///  
    /// If the supplied data doesn't match the expected columns (noted by the  
``MLClassifier/featureColumns`` property),  
    /// this method throws an ``MLCreateError/type(reason:)``.  
    ///  
    /// - Parameters:  
    ///   - data: The data you want the model to classify.  
    ///  
    /// - Returns: A column of labels predicted by the classifier.  
@available 10.14 13.0  
    "Use DataFrame instead of MLDataTable."  
@available  
@available  
public func predictions MLDataTable throws  
MLUntypedColumn  
  
    /// Evaluates the classifier on the provided labeled data.  
    ///  
    /// Evaluation should be done on a testing data set that the model has not  
seen as part of the training or  
    /// validation data sets. The data should have feature columns with identical  
name and type to the  
    /// training data, as well as a labels column with the same name.  
    ///  
    /// - Parameters:  
    ///   - labeledData: A `DataFrame` to evaluate the trained model on.  
    ///  
    /// - Returns: Metrics that describe the classification errors  
    ///(``MLClassifierMetrics/classificationError``), the precision  
and recall  
    /// percentages(``MLClassifierMetrics/precisionRecall``), and  
a table that  
    /// describes how labels were misapplied  
(``MLClassifierMetrics/confusion``)  
    /// on the provided data.  
@available macOS 12.0  
@available  
@available  
public func evaluation DataFrame  
MLClassifierMetrics  
  
    /// Evaluates the classifier on the provided labeled data.  
    ///  
    /// Evaluation should be done on a testing data set that the model has not  
seen as part of the training or  
    /// validation data sets. The data should have feature columns with identical  
name and type to the
```

```
    /// training data, as well as a labels column with the same name.  
    ///  
    /// - Parameters:  
    ///   - labeledData: An `MLDataTable` to evaluate the trained model  
on.  
    ///  
    /// - Returns: Metrics that describe the classification errors  
(``MLClassifierMetrics/classificationError``), the  
    /// precision and recall percentages  
(``MLClassifierMetrics/precisionRecall``), and a table that describes  
how  
    /// labels were misapplied (``MLClassifierMetrics/confusion``)  
on the provided data.  
    @available 10.14 13.0  
    "Use DataFrame instead of MLDataTable."  
    @available  
    @available  
    public func evaluation MLDataTable  
MLClassifierMetrics  
  
    /// Exports a Core ML model file for use in your app.  
    public func write URL  
MLModelMetadata throws  
  
    /// Exports a Core ML model file for use in your app.  
    public func write String  
MLModelMetadata throws  
  
@available macOS 10.14  
@available  
@available  
extension MLClassifier  
  
    /// The underlying Core ML model stored in memory.  
    public var model MLModel get  
  
    /// Measurements of the classifier's performance on the training data set.  
    public var trainingMetrics MLClassifierMetrics get  
  
    /// Measurements of the classifier's performance on the validation data set.  
    public var validationMetrics MLClassifierMetrics get  
  
    /// The name of the column you selected at initialization to define which  
    /// categories the classifier predicts.  
    public var targetColumn String get  
  
    /// The names of the columns you selected at initialization to train the  
    /// classifier.  
    public var featureColumns String get
```

```
@available macOS 10.14
@available
@available
extension MLClassifier <CustomStringConvertible>

    /// A text representation of the classifier.
    public var description String get

@available macOS 10.14
@available
@available
extension MLClassifier <CustomDebugStringConvertible>

    /// A text representation of the classifier that's suitable for output
    /// during debugging.
    public var debugDescription String get

@available macOS 10.14
@available
@available
extension MLClassifier <CustomPlaygroundDisplayConvertible>

    /// A description of the classifier shown in a playground.
    public var playgroundDescription Any get

    /// Metrics you use to evaluate a classifier's performance.
    ///
    /// Use ``MLClassifierMetrics`` to evaluate your model's ability to
    /// distinguish between different categories when it's
    /// classifying data.
    ///
    /// You can determine the model's accuracy using the
    /// ``MLClassifierMetrics/classificationError`` metric. For
    /// information about how your model is mislabeling or missing a certain category,
    /// use the
    /// ``MLClassifierMetrics/precisionRecall`` metric. To determine
    /// specific cases where your model is mistaking one label
    /// for another, use the ``MLClassifierMetrics/confusion`` property.
    ///
    /// Accuracy can be a misleading metric if you use unbalanced data, which means
    /// the number of examples for some
    /// categories are much larger than others. Instead, use
    /// ``MLClassifierMetrics/precisionRecall`` or
    /// ``MLClassifierMetrics/confusion``.
    ///
```

```

/// - Note: Each trained model contains different metrics for its various data
sets (training, validation, and testing).
/// <doc:improving-your-model-s-accuracy> compares these metrics between
different data sets.
@available macOS 10.14 iOS 15.0 tvOS 16.0
public struct MLClassifierMetrics

    /// Creates empty classifier metrics.
    ///
    /// You typically don't initialize metrics directly. Instead you get metrics about
your model after training. For
    /// example, when you train an ``MLClassifier``, you can look at its
``MLClassifier/trainingMetrics`` and
    /// ``MLClassifier/validationMetrics`` properties. Additionally,
you can check the performance on a test set with
    /// the ``MLClassifier/evaluation(on:)`` method.
    ///
    /// - Warning: This initializer should not be used, it creates an empty
instance.
    ///
    /// - Parameters:
    ///   - classificationError: The fraction of incorrectly labeled
examples.
    ///   - confusion: A confusion matrix describing the classifications for
each category.
    ///   - precisionRecall: A two-dimensional table describing the
precision and recall for each category.
    @available 10.14 14.0
    @available 15.0 17.0
    @available 16.0 17.0
    public init
        Double
    MLDataTable
        MLDataTable

    /// The underlying error present when the metrics are invalid.
    public var error: Any Error {
        get
            /// A Boolean value indicating whether the classifier model was able to
calculate metrics.
            ///
            /// Your metrics may be invalid if you attempt to perform evaluation on data
that doesn't match the structure of
            /// your training examples.
        public var isValid: Bool {
            get
                /// The fraction of incorrectly labeled examples.
                ///
                /// The classification error describes how many examples were incorrectly
labeled divided by the total number of
                /// examples. Accuracy as a percentage may be more intuitive. You can
calculate it as follows:
                ///

```

```

    /// swift
    /// let accuracy = (1 - metrics.classificationError) * 100
    ///
    ///
    /// - Important: This is a useful metric only when the data is well-balanced between categories. For example,
    /// suppose you build a classifier to detect a rare disease with very few examples of sick patients compared
    /// to the number of healthy patients. Predicting that a new patient will always be healthy would be highly
    /// accurate (low classification error), but a poor disease detector. The ``MLClassifierMetrics/precisionRecall``
    /// and ``MLClassifierMetrics/confusion`` properties provide more detail in these cases.
    public var classificationError Double get

    /// A table comparing the actual and predicted labels for each classification category.
    ///
    /// The confusion data table describes how examples were mislabeled between categories. Each row contains the true
    /// label, the predicted label, and the count for each possible combination of categories. For example, the table
    /// below lists that “business” was labeled correctly with “business” 113 times, while “business” was confused with
    /// “entertainment” 2 times.
    ///
    /// ! [A table showing the format of the confusion table containing rows for the true label the label predicted by
    /// the classifier and a count for how many times those labels were combined.] (MLClassifierMetrics-confusion-1)
    ///
    /// To gain insight into the performance of your model, you can use this data table to determine what categories
    /// your model is most confused about (making the most mistakes on) for a given data set. For example, the code
    /// listing below shows how to find the mistake that happens most frequently.
    ///
    /// swift
    /// let confusion = model.validationMetrics.confusion
    ///
    /// // Filter for rows which contain mistakes.
    /// let errors = confusion[confusion["True Label"] != confusion["Predicted"]]
    /// let mostCommonError = errors.rows.max { row1, row2 in
    ///     row1["Count", Int.self]! < row2["Count", Int.self]!
    /// }
    /// print(mostCommonError ?? "The confusion table is empty.")
    /// // ["Predicted": "tech", "True Label": "business",

```

```

"Count": 9]
////
////
/// Another useful view into this data is to compare the actual and predicated
labels using a matrix. Printing the
/// ``MLClassifierMetrics`` directly displays the matrix format.
////
/// ``swift
/// print(model.validationMetrics)
/// // ...
/// // *****CONFUSION MATRIX*****
/// // -----
/// // True\Pred business entertainment politics sport
tech
/// // business 113 2 3 0 9
/// // entertainment 1 183 3 2 3
/// // politics 6 8 116 0 3
/// // sport 0 6 1 135 3
/// // tech 2 7 3 0 129
/// // ...
////
/// In this example, the upper left hand count shows that 113 business
examples were correctly labeled as
/// "business". The second column shows that "entertainment" was predicted
for 2 "business" examples. The second
/// row shows that 1 "entertainment" example was mislabeled as "business".
@available 10.14 14.0
@available 15.0 17.0
@available 16.0 17.0
public var confusion MLDataTable get

/// A data frame comparing the actual and predicted labels for each class.
///
/// The confusion data frame describes how examples were mislabeled
between categories. Each row contains the true
/// label, the predicted label, and the number of instances of that
combination. For example, the table below lists
/// that "business" was labeled correctly with "business" 113 times, while
"business" was confused with
/// "entertainment" 2 times.
///
/// ! [A table showing the format of the confusion matrix containing rows for
the true label the label predicted by
/// the classifier and a count for how many times those labels were
combined.] (MLClassifierMetrics-confusion-1)
@available macOS 14 iOS 17.0 tvOS 17.0
public var confusionDataFrame DataFrame get

/// A data table listing the precision and recall percentages for each class.

```

```

    /**
     * Precision and recall are metrics calculated for each class. Together they
     * describe the tradeoff between
     *   - misapplying a label too liberally and missing examples of that label.
     */
     /**
      * Precision describes how effective the model was at applying a label only
      * when appropriate for a given category
      *   - (few false positives).
      */
      /**
       * Recall describes how effective the model was at finding all the relevant
       * examples of a category (few false
       *   - negatives).
       */
       /**
        * 
        */
        /**
         * The figure below shows how each example contributes to the precision
         * and recall percentages for the category
         *   - "Elephant".
         */
         /**
          * ![[A table of actual and predicted labels for the Elephant category.]
          * (MLClassifierMetrics-precisionRecall-2)
          */
          /**
           * "Elephant" appears as the true or correct label only once, but it's predicted
           * twice. This second prediction is
           *   - an error in precision. Precision and recall can give you a much better idea
           * of how your model is making
           *   - mistakes than ``MLClassifierMetrics/classificationError``.
           */
           /**
            * To determine what other categories "Elephant" examples may have been
            * labeled with, see the
            *   - ``MLClassifierMetrics/confusion`` property.
            */
            @available
            @available
            @available
            public var precisionRecall MLDataTable get

    /**
     * A data frame listing the precision and recall percentages for each class.
     */
     /**
      * Precision and recall are metrics calculated for each class. Together they
      * describe the tradeoff between
      *   - misapplying a label too liberally and missing examples of that label.
      */
      /**
       * Precision describes how effective the model was at applying a label only
       * when appropriate for a given category
       *   - (few false positives).
       */
       /**
        * Recall describes how effective the model was at finding all the relevant
        * examples of a category (few false
        *   - negatives).
        */
        /**
         * 

```

```
///  
/// The figure below shows how each example contributes to the precision  
and recall percentages for the category  
/// "Elephant".  
///  
/// ! [A table of actual and predicted labels for the Elephant category.]  
(MLClassifierMetrics-precisionRecall-2)  
///  
/// "Elephant" appears as the true or correct label only once, but it's predicted  
twice. This second prediction is  
/// an error in precision. Precision and recall can give you a much better idea  
of how your model is making  
/// mistakes than ``MLClassifierMetrics/classificationError``.  
///  
/// To determine what other categories "Elephant" examples may have been  
labeled with, see the  
/// ``MLClassifierMetrics/confusion`` property.  
@available macOS 14.0 iOS 17.0 tvOS 17.0  
public var precisionRecallDataFrame DataFrame get  
  
@available macOS 10.14 iOS 15.0 tvOS 16.0  
extension MLClassifierMetrics CustomStringConvertible  
CustomDebugStringConvertible  
CustomPlaygroundDisplayConvertible  
  
/// A text representation of the classifier metrics.  
public var description String get  
  
/// A text representation of the classifier metrics that's suitable for output  
during debugging.  
public var debugDescription String get  
  
/// A description of the classifier metrics shown in a playground.  
public var playgroundDescription Any get  
  
/// The errors Create ML throws while performing various operations, such as  
/// training models, making predictions, writing models to a file system, and so  
/// on.  
@available macOS 10.14 iOS 15.0 tvOS 16.0  
public enum MLCreateError Error Sendable  
  
/// An error that indicates a failure not covered by one of the other  
/// errors.  
case generic String  
  
/// An error that indicates a missing or incorrect type.  
case type String
```

```
    /// An error that indicates an I/O failure.  
    case io          String  
  
    /// An error that indicates you canceled the training session.  
    @available macOS 11.0  iOS 15.0  tvOS 16.0  
    case cancelled  
  
    /// An error that indicates the training session parameters are  
    /// incompatible.  
    @available macOS 11.0  iOS 15.0  tvOS 16.0  
    case incompatibleParameters      String  
                                  String  
  
    /// An error that indicates the training data is different from the data  
    /// when you created the session.  
    @available macOS 11.0  iOS 15.0  tvOS 16.0  
    case modifiedTrainingData  
  
  
@available macOS 10.14  iOS 15.0  tvOS 16.0  
extension MLCreateError  CustomNSError  LocalizedError  
  
    /// The domain of the error.  
    public static var errorDomain  String  get  
  
    /// A localized, human-readable reason behind the failure, if applicable.  
    public var failureReason  String  get  
  
    /// A localized, human-readable description of the error and why it  
    /// occurred, if applicable.  
    public var errorDescription  String  get  
  
    /// The numeric code of this error.  
    @available macOS 12.0  iOS 15.0  tvOS 16.0  
    public var errorCode  Int  get  
  
    /// A dictionary that provides additional information about the error.  
    @available macOS 12.0  iOS 15.0  tvOS 16.0  
    public var errorUserInfo  String  Any  get  
  
  
@available macOS 10.14  iOS 15.0  tvOS 16.0  
extension MLCreateError  CustomStringConvertible  
CustomDebugStringConvertible  
  
    /// A human-readable description of the error.  
    public var description  String  get  
  
    /// A human-readable description of the error that's suitable for output
```

```
    /// during debugging.
    public var debugDescription String get

    /// A global constant that defines the domain for Create ML errors.
    @available macOS 10.14 iOS 15.0 tvOS 16.0
    public let MLCreateErrorDomain String

    /// A column of typed values in a data table.
    ///
    /// A column is a homogenous collection of data values, similar to an
    /// <doc://com.apple.documentation/documentation/swift/array>. Columns are the
    main components of an ``MLDataTable``
    /// and are designed to efficiently scale with large data sets.
    ///
    /// Typically you use ``MLDataColumn``, the typed equivalent to
    ``MLUntypedColumn``, to work directly with the column's
    /// element type. A data column has extra math and statistics functionality when its
    element type is
    /// <doc://com.apple.documentation/documentation/swift/int> or
    /// <doc://com.apple.documentation/documentation/swift/double>.
    @available macOS 10.14 iOS 15.0 tvOS 16.0
    public struct MLDataColumn<Element : MLDataValueConvertible>

        /// The number of elements in the column.
        public var count Int get

        @available macOS 11.0 iOS 15.0 tvOS 16.0
        public var isEmpty Bool get

        /// The underlying error present when the column is invalid.
        public var error any Error get

        /// A Boolean value that indicates whether the column is valid.
        ///
        /// Check ``MLDataColumn/isValid`` after you create or mutate a data
        column to ensure it's valid. If the value is
        /// <doc://com.apple.documentation/documentation/swift/false>, the data
        column encountered an error and you can't
        /// use it for subsequent operations. For example, comparing two columns of
        different sizes creates an invalid
        /// column.
        public var isValid Bool get

        /// Creates a new column from a given sequence of elements.
        ///
        /// Use this initializer to create a column from a sequence of any type that
        conforms to ``MLDataValueConvertible``.
        ///
```

```

/// ````swift
/// let sequenceColumn = ML DataColumn([2, 3, 5, 7, 11])
/// print(sequenceColumn) // Prints [2, 3, 5, 7, 11]
///
///
/// - Parameters:
///   - source: A sequence of elements for the new column.
public init S _           S where Element      S Element
S Sequence

/// Constructs a new Column containing the specified number of a single,
repeated ML DataValue.
public init                               ML DataValue
Int

/// Creates a new column with a repeating element.
///
/// Use this initializer to create a column of repeating elements with any type
that conforms to
/// ``ML DataValueConvertible``, including integers, doubles, strings,
arrays, and dictionaries.
///
/// ````swift
/// let three5s = ML DataColumn(repeating: 5, count: 3)
/// print(three5s) // Prints [5, 5, 5]
///
///
/// - Parameters:
///   - repeatedValue: An initial value for every element in the new
column.
///   - count: A number of elements to create for the new column.
public init                           Element      Int
Element

/// Constructs an invalid Column.
///
/// Assigning an invalid Column to a column name in a DataTable will remove
any Column
/// previously stored under that name.
public init

/// Appends the elements of the given column to the end of this column.
///
/// - Parameters:
///   - newColumn: A column to append.
///
/// - Note: The type of `newColumn` must be the same type or
convertible to the same type as the column. See
/// ``ML DataValueConvertible``.
public mutating func append
ML DataColumn Element

```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0
extension ML DataColumn
```

```
    /// Creates a column of Booleans by testing whether each element in the
    /// first column is equal to the corresponding element in the second column.
    ///
    /// - Parameters:
    ///   - a: A column.
    ///
    ///   - b: A column.
    ///
    /// - Returns: A new column of Booleans if the columns are the same
size;
    /// otherwise an invalid column.
    public static func      ML DataColumn Element
ML DataColumn Element      ML DataColumn Bool

    /// Creates a column of Booleans by testing whether each element in the
    /// first column is not equal to the corresponding element in the second
    /// column.
    ///
    /// - Parameters:
    ///   - a: A column.
    ///
    ///   - b: A column.
    ///
    /// - Returns: A new column of Booleans if the columns are the same
size;
    /// otherwise an invalid column.
    public static func      ML DataColumn Element
ML DataColumn Element      ML DataColumn Bool

    /// Creates a column of Booleans by testing whether each element in the
    /// first column is greater than the corresponding element in the second
    /// column.
    ///
    /// - Parameters:
    ///   - a: A column.
    ///
    ///   - b: A column.
    ///
    /// - Returns: A new column of Booleans if the columns are the same
size;
    /// otherwise an invalid column.
    public static func      ML DataColumn Element
ML DataColumn Element      ML DataColumn Bool

    /// Creates a column of Booleans by testing whether each element in the
```

```
    /// first column is less than the corresponding element in the second
    /// column.
    ///
    /// - Parameters:
    ///   - a: A column.
    ///
    ///   - b: A column.
    ///
    /// - Returns: A new column of Booleans if the columns are the same
size;
    /// otherwise an invalid column.
public static func      MLDataColumn Element
MLDataColumn Element      MLDataColumn Bool

    /// Creates a column of Booleans by testing whether each element in the
    /// first column is greater than or equal to the corresponding element in
    /// the second column.
    ///
    /// - Parameters:
    ///   - a: A column.
    ///
    ///   - b: A column.
    ///
    /// - Returns: A new column of Booleans if the columns are the same
size;
    /// otherwise an invalid column.
public static func      MLDataColumn Element
MLDataColumn Element      MLDataColumn Bool

    /// Creates a column of Booleans by testing whether each element in the
    /// first column is less than or equal to the corresponding element in the
    /// second column.
    ///
    /// - Parameters:
    ///   - a: A column.
    ///
    ///   - b: A column.
    ///
    /// - Returns: A new column of Booleans if the columns are the same
size;
    /// otherwise an invalid column.
public static func      MLDataColumn Element
MLDataColumn Element      MLDataColumn Bool

    /// Creates a column of Booleans by testing whether each element in the
    /// given column is equal to the given value.
    ///
    /// - Parameters:
    ///   - a: A column.
    ///
```

```
/// - b: A value of the same type as the elements of the column.  
///  
/// - Returns: A new column of Booleans.  
public static func MLDataColumn Element  
Element MLDataColumn Bool  
  
/// Creates a column of Booleans by testing whether each element in the  
/// given column is not equal to the given value.  
///  
/// - Parameters:  
/// - a: A column.  
///  
/// - b: A value of the same type as the elements of the column.  
///  
/// - Returns: A new column of Booleans.  
public static func MLDataColumn Element  
Element MLDataColumn Bool  
  
/// Creates a column of Booleans by testing whether each element in the  
/// given column is greater than the given value.  
///  
/// - Parameters:  
/// - a: A column.  
///  
/// - b: A value of the same type as the elements of the column.  
///  
/// - Returns: A new column of Booleans.  
public static func MLDataColumn Element  
Element MLDataColumn Bool  
  
/// Creates a column of Booleans by testing whether each element in the  
/// given column is less than the given value.  
///  
/// - Parameters:  
/// - a: A column.  
///  
/// - b: A value of the same type as the elements of the column.  
///  
/// - Returns: A new column of Booleans.  
public static func MLDataColumn Element  
Element MLDataColumn Bool  
  
/// Creates a column of Booleans by testing whether each element in the  
/// given column is greater than or equal to the given value.  
///  
/// - Parameters:  
/// - a: A column.  
///  
/// - b: A value of the same type as the elements of the column.
```

```
///  
/// – Returns: A new column of Booleans.  
public static func MLDataColumn Element  
Element MLDataColumn Bool  
  
/// Creates a column of Booleans by testing whether each element in the  
/// given column is less than or equal to the given value.  
///  
/// – Parameters:  
/// – a: A column.  
///  
/// – b: A value of the same type as the elements of the column.  
///  
/// – Returns: A new column of Booleans.  
public static func MLDataColumn Element  
Element MLDataColumn Bool  
  
/// Creates a column of Booleans by testing whether the given value is equal  
/// to each element in the given column.  
///  
/// – Parameters:  
/// – a: A value of the same type as the elements of the column.  
///  
/// – b: A column.  
///  
/// – Returns: A new column of Booleans.  
public static func Element  
MLDataColumn Element MLDataColumn Bool  
  
/// Creates a column of Booleans by testing whether the given value is not  
/// equal to each element in the given column.  
///  
/// – Parameters:  
/// – a: A value of the same type as the elements of the column.  
///  
/// – b: A column.  
///  
/// – Returns: A new column of Booleans.  
public static func Element  
MLDataColumn Element MLDataColumn Bool  
  
/// Creates a column of Booleans by testing whether the given value is  
/// greater than each element in the given column.  
///  
/// – Parameters:  
/// – a: A value of the same type as the elements of the column.  
///  
/// – b: A column.  
///
```

```
    /// - Returns: A new column of Booleans.  
    public static func      Element  
    MLDataColumn Element      MLDataColumn Bool  
  
    /// Creates a column of Booleans by testing whether the given value is less  
    /// than each element in the given column.  
    ///  
    /// - Parameters:  
    ///   - a: A value of the same type as the elements of the column.  
    ///  
    ///   - b: A column.  
    ///  
    /// - Returns: A new column of Booleans.  
    public static func      Element  
    MLDataColumn Element      MLDataColumn Bool  
  
    /// Creates a column of Booleans by testing whether the given value is  
    /// greater than or equal to each element in the given column.  
    ///  
    /// - Parameters:  
    ///   - a: A value of the same type as the elements of the column.  
    ///  
    ///   - b: A column.  
    ///  
    /// - Returns: A new column of Booleans.  
    public static func      Element  
    MLDataColumn Element      MLDataColumn Bool  
  
    /// Creates a column of Booleans by testing whether the given value is less  
    /// than or equal to each element in the given column.  
    ///  
    /// - Parameters:  
    ///   - a: A value of the same type as the elements of the column.  
    ///  
    ///   - b: A column.  
    ///  
    /// - Returns: A new column of Booleans.  
    public static func      Element  
    MLDataColumn Element      MLDataColumn Bool
```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0  
extension MLDataColumn where Element == Int
```

```
    /// Creates a column of integers by adding each element in the first column  
    /// to the corresponding element in the second column.  
    ///  
    /// - Parameters:  
    ///   - a: A column of integers.
```

```
///  
/// - b: A column of integers.  
///  
/// - Returns: A new column of integers if the columns are the same size;  
/// otherwise an invalid column.  
public static func MLDataColumn Int  
MLDataColumn Int MLDataColumn Int  
  
/// Creates a column of integers by subtracting each element in the second  
/// column from the corresponding element in the first column.  
///  
/// - Parameters:  
/// - a: A column of integers.  
///  
/// - b: A column of integers.  
///  
/// - Returns: A new column of integers if the columns are the same size;  
/// otherwise an invalid column.  
public static func MLDataColumn Int  
MLDataColumn Int MLDataColumn Int  
  
/// Creates a column of integers by multiplying each element in the first  
/// column by the corresponding element in the second column.  
///  
/// - Parameters:  
/// - a: A column of integers.  
///  
/// - b: A column of integers.  
///  
/// - Returns: A new column of integers if the columns are the same size;  
/// otherwise an invalid column.  
public static func MLDataColumn Int  
MLDataColumn Int MLDataColumn Int  
  
/// Creates a column of integers by dividing each element in the first  
/// column by the corresponding element in the second column.  
///  
/// - Parameters:  
/// - a: A column of integers.  
///  
/// - b: A column of integers.  
///  
/// - Returns: A new column of integers if the columns are the same size;  
/// otherwise an invalid column.  
public static func MLDataColumn Int  
MLDataColumn Int MLDataColumn Int  
  
/// Creates a column of integers by adding the given integer to each element  
/// of the given column.
```

```
///  
/// - Parameters:  
///   - a: An integer.  
///  
///   - b: A column of integers.  
///  
/// - Returns: A new column of integers.  
public static func      Int      MLDataColumn Int  
MLDataColumn Int  
  
/// Creates a column of integers by subtracting each element of the given  
/// column from the given integer.  
///  
/// - Parameters:  
///   - a: An integer.  
///  
///   - b: A column of integers.  
///  
/// - Returns: A new column of integers.  
public static func      Int      MLDataColumn Int  
MLDataColumn Int  
  
/// Creates a column of integers by multiplying the given integer by each  
/// element of the given column.  
///  
/// - Parameters:  
///   - a: An integer.  
///  
///   - b: A column of integers.  
///  
/// - Returns: A new column of integers.  
public static func      Int      MLDataColumn Int  
MLDataColumn Int  
  
/// Creates a column of integers by dividing the given integer by each  
/// element of the given column.  
///  
/// - Parameters:  
///   - a: An integer.  
///  
///   - b: A column of integers.  
///  
/// - Returns: A new column of integers.  
public static func      Int      MLDataColumn Int  
MLDataColumn Int  
  
/// Creates a column of integers by adding each element of the given column  
/// to the given integer.  
///
```

```
/// - Parameters:  
///   - a: A column of integers.  
///  
///   - b: An integer.  
///  
/// - Returns: A new column of integers.  
public static func      MLDataColumn Int      Int  
MLDataColumn Int  
  
/// Creates a column of integers by subtracting the given integer from each  
/// element of the given column.  
///  
/// - Parameters:  
///   - a: A column of integers.  
///  
///   - b: An integer.  
///  
/// - Returns: A new column of integers.  
public static func      MLDataColumn Int      Int  
MLDataColumn Int  
  
/// Creates a column of integers by multiplying each element of the given  
/// column by the given integer.  
///  
/// - Parameters:  
///   - a: A column of integers.  
///  
///   - b: An integer.  
///  
/// - Returns: A new column of integers.  
public static func      MLDataColumn Int      Int  
MLDataColumn Int  
  
/// Creates a column of integers by dividing each element of the given  
/// column by the given integer.  
///  
/// - Parameters:  
///   - a: A column of integers.  
///  
///   - b: An integer.  
///  
/// - Returns: A new column of integers.  
public static func      MLDataColumn Int      Int  
MLDataColumn Int  
  
@available macOS 10.14 iOS 15.0 tvOS 16.0  
extension MLDataColumn where Element      Double
```

```
    /// Creates a column of doubles by adding each element in the first column
    /// to the corresponding element in the second column.
    ///
    /// - Parameters:
    ///   - a: A column of doubles.
    ///
    ///   - b: A column of doubles.
    ///
    /// - Returns: A new column of doubles if the columns are the same size;
    /// otherwise an invalid column.
    public static func      MLDataColumn Double
MLDataColumn Double      MLDataColumn Double

    /// Creates a column of doubles by subtracting each element in the second
    /// column from the corresponding element in the first column.
    ///
    /// - Parameters:
    ///   - a: A column of doubles.
    ///
    ///   - b: A column of doubles.
    ///
    /// - Returns: A new column of doubles if the columns are the same size;
    /// otherwise an invalid column.
    public static func      MLDataColumn Double
MLDataColumn Double      MLDataColumn Double

    /// Creates a column of doubles by multiplying each element in the first
    /// column by the corresponding element in the second column.
    ///
    /// - Parameters:
    ///   - a: A column of doubles.
    ///
    ///   - b: A column of doubles.
    ///
    /// - Returns: A new column of doubles if the columns are the same size;
    /// otherwise an invalid column.
    public static func      MLDataColumn Double
MLDataColumn Double      MLDataColumn Double

    /// Creates a column of doubles by dividing each element in the first column
    /// by the corresponding element in the second column.
    ///
    /// - Parameters:
    ///   - a: A column of doubles.
    ///
    ///   - b: A column of doubles.
    ///
    /// - Returns: A new column of doubles if the columns are the same size;
    /// otherwise an invalid column.
```

```
public static func      MLDataColumn Double
MLDataColumn Double      MLDataColumn Double

    /// Creates a column of doubles by adding the given double to each element
    /// of the given column.
    ///
    /// - Parameters:
    ///   - a: A double.
    ///
    ///   - b: A column of doubles.
    ///
    /// - Returns: A new column of doubles.
public static func      Double      MLDataColumn Double
MLDataColumn Double

    /// Creates a column of doubles by subtracting each element of the given
    /// column from the given double.
    ///
    /// - Parameters:
    ///   - a: A double.
    ///
    ///   - b: A column of doubles.
    ///
    /// - Returns: A new column of doubles.
public static func      Double      MLDataColumn Double
MLDataColumn Double

    /// Creates a column of doubles by multiplying the given double by each
    /// element of the given column.
    ///
    /// - Parameters:
    ///   - a: A double.
    ///
    ///   - b: A column of doubles.
    ///
    /// - Returns: A new column of doubles.
public static func      Double      MLDataColumn Double
MLDataColumn Double

    /// Creates a column of doubles by dividing the given double by each
element
    /// of the given column.
    ///
    /// - Parameters:
    ///   - a: A double.
    ///
    ///   - b: A column of doubles.
    ///
    /// - Returns: A new column of doubles.
```

```
public static func      Double      MLDataColumn Double
MLDataColumn Double

    /// Creates a column of doubles by adding each element of the given column
    /// to the given double.
    ///
    /// - Parameters:
    ///   - a: A column of doubles.
    ///
    ///   - b: A double.
    ///
    /// - Returns: A new column of doubles.
public static func      MLDataColumn Double      Double
MLDataColumn Double

    /// Creates a column of doubles by subtracting the given double from each
    /// element of the given column.
    ///
    /// - Parameters:
    ///   - a: A column of doubles.
    ///
    ///   - b: A double.
    ///
    /// - Returns: A new column of doubles.
public static func      MLDataColumn Double      Double
MLDataColumn Double

    /// Creates a column of doubles by multiplying each element of the given
    /// column by the given double.
    ///
    /// - Parameters:
    ///   - a: A column of doubles.
    ///
    ///   - b: A double.
    ///
    /// - Returns: A new column of doubles.
public static func      MLDataColumn Double      Double
MLDataColumn Double

    /// Creates a column of doubles by dividing each element of the given
    column
    /// by the given double.
    ///
    /// - Parameters:
    ///   - a: A column of doubles.
    ///
    ///   - b: A double.
    ///
    /// - Returns: A new column of doubles.
```

```

public static func      MLDataColumn Double      Double
MLDataColumn Double

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataColumn where Element Bool

    /// Creates a column of Booleans by performing a logical OR operation on
    /// each element in the first column with the corresponding element in the
    /// second column.
    ///
    /// - Parameters:
    ///   - a: A column of Booleans.
    ///
    ///   - b: A column of Booleans.
    ///
    /// - Returns: A new column of Booleans if the columns are the same
size;
    /// otherwise an invalid column.
public static func      MLDataColumn Bool
MLDataColumn Bool      MLDataColumn Bool

    /// Creates a column of Booleans by performing a logical AND operation on
    /// each element in the first column with the corresponding element in the
    /// second column.
    ///
    /// - Parameters:
    ///   - a: A column of Booleans.
    ///
    ///   - b: A column of Booleans.
    ///
    /// - Returns: A new column of Booleans if the columns are the same
size;
    /// otherwise an invalid column.
public static func      MLDataColumn Bool
MLDataColumn Bool      MLDataColumn Bool

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataColumn

    /// Returns a `MLDataColumn` containing only the elements for which the
    corresponding mask has a nonzero or
    /// non-default value.
    ///
    /// - Parameter mask: a MLUntypedColumn with the same element
count as this MLUntypedColumn.
    /// - Returns: A MLUntypedColumn containing the subsequence of this
MLUntypedColumn's elements indicated by the
    /// mask MLUntypedColumn.

```

```

public subscript      MLUntypedColumn
MLDataColumn Element   get

    /// Creates a subset of the column by masking its elements with a column of
Booleans.
    ///
    /// - Parameters:
    /// - mask: A Boolean column indicating whether elements should be
kept (`true`) or removed (`false`) in the
/// derived column.
    ///
    /// - Returns: A new column.
public subscript      MLDataColumn Bool
MLDataColumn Element   get

    /// Creates a new column, potentially with missing values, by applying the
given thread-safe transform to every
/// non-missing element of this column.
    ///
    /// - Parameters:
    /// - lazyTransform: A thread-safe element transformation function.
The implementation of the transform you
/// provide should accept an `Element` of the column and return a
transformed value of a type that conforms to
/// ``MLDataValueConvertible``. If the transform returns `nil`
for a given element, the corresponding element
/// in the new column will have a missing value.
    ///
    /// - Returns: A new column typed to the return type of
`lazyTransform`.
public func map T      @escaping Element
T      MLDataColumn T where T : MLDataValueConvertible

    /// Creates a new column by applying the given thread-safe transform to
every non-missing element of this column.
    ///
    /// - Parameters:
    /// - lazyTransform: A thread-safe element transformation function.
The implementation of the transform you
/// provide should accept an `Element` of the column and return a
transformed value of a type that conforms to
/// ``MLDataValueConvertible``.
    ///
    /// - Returns: A new column typed to the return type of
`lazyTransform`.
public func map T      @escaping Element
T      MLDataColumn T where T : MLDataValueConvertible

    /// Creates a new column, potentially with missing elements, by applying the
given thread-safe transform to every
/// element of the column, including missing elements.

```

```

    /**
     * - Parameters:
     *   - lazyTransform: A thread-safe element transformation function.
     * The implementation of the transform you
     *   provide should accept an `Element` of the column and return a
     * transformed value of a type that conforms to
     *   ``MLDataValueConvertible``. If the transform returns `nil`
     * for a given element, the corresponding element
     *   in the new column will have a missing value.
     */
    /**
     * - Returns: A new column.
     public func mapMissing T _ @escaping
     Element T ML DataColumn T where T
     MLDataValueConvertible

     /// Creates a new column by converting this column to the given type.
     /**
     * This method is functionally equivalent to the initializers of
     * ``ML DataColumn`` that have one parameter
     *   `column`, such as ``ML DataColumn/init(column:)``.
     /**
     * - Parameters:
     *   - type: A type of ``ML DataColumn`` to convert the contents of
     * the column to, using ``MLDataValueConvertible``.
     /**
     * - Returns: A new column.
     public func map T T ML DataColumn T
     where T MLDataValueConvertible

     /// Creates a subset of the column by removing all elements without a value.
     /**
     * - Returns: A new column.
     public func dropMissing ML DataColumn Element

     /// Creates a modified copy of the column such that every missing element is
     replaced with the given value.
     /**
     * - Parameters:
     *   - value: A value used to replace every undefined element.
     /**
     * - Returns: A new column.
     public func fillMissing Element
     ML DataColumn Element

     /// Creates a subset of the column by removing all duplicate elements.
     /**
     * - Note: The new column may not preserve the order of the original
     column.
     /**
     * - Returns: A new column.

```

```
public func dropDuplicates      MLDataColumn Element
    /// Creates a subset of the column, given a number of initial elements.
    ///
    /// - Parameters:
    ///   - maxLength: An integer that limits the number of elements to use
    // from the beginning of the column. The
    ///   default value is `10`.
    ///
    /// - Returns: A new column.
    public func prefix _           Int     10
MLDataColumn Element

    /// Creates a subset of the column, given a number of final elements.
    ///
    /// - Parameters:
    ///   - maxLength: An integer that limits the number of elements to use
    // from the end of the column. The default
    ///   value is `10`.
    ///
    /// - Returns: A new column.
    public func suffix _           Int     10
MLDataColumn Element

    /// Returns a new MLDataColumn containing values sorted by the specified
order.
    ///
    /// - Parameter byIncreasingOrder: A boolean indicating whether to
sort values in ascending or descending order.
    ///   The default is true, sorted by ascending order.
    ///
    /// - Returns: A MLDataColumn sorted by the specified order.
    public func sort                Bool    true
MLDataColumn Element

    /// Returns a new MLDataColumn by copying the original MLDataColumn
    ///
    /// - Returns: A MLDataColumn which is a copy of the original
MLDataColumn.
    public func copy      MLDataColumn Element

    /// Creates a new column by immediately evaluating any lazily applied data
    /// processing operations stored in the column.
    ///
    /// - Returns: A new column.
    public func materialize throws  MLDataColumn Element
```

@available macOS 10.14 iOS 15.0 tvOS 16.0  
extension MLDataColumn

```

    /// Accesses the element at the given index.
    ///
    /// - Parameters:
    ///   - index: A row number of the column, beginning with `0`.
    ///
    /// - Returns: The `Element` at the given index; otherwise, `nil`.
public func element Int Element

    /// Accesses the element at the given row.
    ///
    /// - Parameters:
    ///   - index: A row number of the column, beginning with `0`.
    ///
    /// - Returns: The `Element` at the given index.
public subscript Int Element get

/// Range-Based Slicing Support
@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataColumn

    /// Creates a subset of the column, given a range of elements.
    ///
    /// - Parameters:
    ///   - slice: A range of integers indicating which elements to include in
    ///     the new column.
    ///
    /// - Returns: A new column.
public subscript Range Int MLDataColumn Element get

    /// Creates a subset of the column, given a range expression of elements.
    ///
    /// - Parameters:
    ///   - slice: An interger range expression indicating which elements to
    ///     include in the new column.
    ///
    /// - Returns: A new column.
public subscript R R MLDataColumn Element
where R RangeExpression R Bound Int get

@available 10.15 15.0
@available
@available
extension MLDataColumn

    /// Provides a visualization for the data in the column.
public func show any MLStreamingVisualizable

```

```

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension ML DataColumn where Element == Int

    /// Creates a new column of integers from a given column whose elements
    can be converted to integers.
    ///
    /// Use this initializer to create a column of integers from another column.
    /// Start by creating a column that is convertible to a column of integers.
    ///
    /// ````swift
    /// let stringsColumn = ML DataColumn(["1", "2", "3", "4",
    "5"])
    /// print(stringsColumn) // Prints ["1", "2", "3", "4",
    "5"]
    /// ````
    ///
    /// Then use ``ML DataColumn/init(column:) -5rg9u`` to convert the
    column to a column of integers.
    ///
    /// ````swift
    /// let intsColumn = ML DataColumn<Int>(column:
    stringsColumn)
    /// print(intsColumn) // Prints [1, 2, 3, 4, 5]
    /// ````
    ///
    /// - Parameters:
    ///   - column: An ``ML DataColumn`` of elements convertible to
    ///     <doc://com.apple.documentation/documentation/swift/int>.
    public init T == ML DataColumn T where T : ML DataColumnConvertible

        /// Returns the sum of the elements in a column of integers.
        ///
        /// - Returns: An
        <doc://com.apple.documentation/documentation/swift/int>; otherwise `nil` if the
        column is invalid.
        public func sum == Int

            /// Returns the element with the lowest value in a column of integers.
            ///
            /// - Returns: An
            <doc://com.apple.documentation/documentation/swift/int>; otherwise `nil` if the
            column is empty
            /// or invalid.
            public func min == Int

                /// Returns the element with the highest value in a column of integers.
                ///

```

```

    /// - Returns: An
<doc://com.apple.documentation/documentation/swift/int>; otherwise `nil` if the
column is empty
    /// or invalid.
public func max      Int

    /// Returns the standard deviation of the elements in a column of integers.
    ///
    /// - Returns: A
<doc://com.apple.documentation/documentation/swift/double>; otherwise `nil` if
the column is
    /// invalid.
@available          10.14          10.15
    "stdev"
@available          15.0           16.0
    "stdev"
@available
public func std      Double

    /// Returns the arithmetic mean of the elements in a column of integers.
    ///
    /// - Returns: A
<doc://com.apple.documentation/documentation/swift/double>; otherwise `nil` if
the column is
    /// invalid.
public func mean      Double

```

```

@available macOS 10.14  iOS 15.0  tvOS 16.0
extension MLDataColumn where Element      Double

```

```

    /// Creates a new column of doubles from a given column whose elements
can be converted to doubles.
    ///
    /// Use this initializer to create a column of doubles from another column.
    /// Start by creating a column that is convertible to a column of doubles.
    ///
    /// ````swift
    /// let stringsColumn = MLDataColumn(["1.0", "2.0", "3.0",
    "4.0", "5.0"])
    /// print(stringsColumn) // Prints ["1.0", "2.0", "3.0",
    "4.0", "5.0"]
    /// ````

    /// Then use ``MLDataColumn/init(column:)`` to convert the
column to a column of doubles.
    ///
    /// ````swift
    /// let doublesColumn = MLDataColumn<Double>(column:
stringsColumn)

```

```

    /// print(doublesColumn) // Prints [1.0, 2.0, 3.0, 4.0,
5.0]
    /**
     */
    /**
     - Parameters:
     - column: An ``MLDataColumn`` of elements convertible to
      <doc://com.apple.documentation/documentation/swift/double>.
public init T          MLDataColumn T  where T
MLDataValueConvertible

    /// Returns the sum of the elements in a column of doubles.
    /**
     */
    /**
     - Returns: A
      <doc://com.apple.documentation/documentation/swift/double>; otherwise `nil` if
      the column is
      /// invalid.
public func sum      Double

    /// Returns the element with the lowest value in a column of doubles.
    /**
     */
    /**
     - Returns: A
      <doc://com.apple.documentation/documentation/swift/double>; otherwise `nil` if
      the column is empty
      /// or invalid.
public func min      Double

    /// Returns the element with the highest value in a column of doubles.
    /**
     */
    /**
     - Returns: A
      <doc://com.apple.documentation/documentation/swift/double>; otherwise `nil` if
      the column is
      /// empty or invalid.
public func max      Double

    /// Returns the standard deviation of the elements in a column of doubles.
    /**
     */
    /**
     - Returns: A
      <doc://com.apple.documentation/documentation/swift/double>; otherwise `nil` if
      the column is
      /// invalid.
@available           10.14           10.15
      "stdev"
@available           15.0            16.0
      "stdev"
@available
public func std      Double

    /// Returns the arithmetic mean of the elements in a column of doubles.
    /**
     */
    /**
     - Returns: A

```

```
<doc://com.apple.documentation/documentation/swift/double>; otherwise `nil` if  
the column is
```

```
    /// invalid.  
public func mean      Double
```

```
@available macOS 10.15  iOS 15.0  tvOS 16.0  
extension MLDataColumn where Element  Int
```

```
    /// Standard deviation of the Elements in the MLDataColumn.
```

```
    ///
```

```
    /// - Returns: Standard deviation of the Element values in the  
MLDataColumn. Returns nil if the MLDataColumn is
```

```
    /// invalid, 0 if the MLDataColumn is empty.
```

```
public func stdev      Double
```

```
@available macOS 10.15  iOS 15.0  tvOS 16.0  
extension MLDataColumn where Element  Double
```

```
    /// Returns the standard deviation of the elements in a column of doubles.
```

```
    ///
```

```
    /// - Returns: A
```

```
<doc://com.apple.documentation/documentation/swift/double>; otherwise `nil` if  
the column is
```

```
    /// invalid.
```

```
public func stdev      Double
```

```
@available macOS 10.14  iOS 15.0  tvOS 16.0  
extension MLDataColumn where Element  String
```

```
    /// Creates a new column of strings from a given column whose elements  
can be converted to strings.
```

```
    ///
```

```
    /// Use this initializer to create a column of strings from another column.
```

```
    /// Start by creating a column that is convertible to a column of strings.
```

```
    ///
```

```
    /// ````swift
```

```
    /// let doublesColumn = MLDataColumn([1.0, 2.718, 3.14,  
4.2, 5.1])
```

```
    /// print(doublesColumn) // Prints [1.0, 2.718, 3.14, 4.2,  
5.1]
```

```
    /// ````
```

```
    ///
```

```
    /// Then use ``MLDataColumn/init(column:)-ztkv`` to convert the  
column to a column of strings.
```

```
    ///
```

```
    /// ````swift
```

```
    /// let stringsColumn = MLDataColumn<String>(column:
```

```
doublesColumn)
    /// print(stringsColumn) // Prints ["1", "2.718", "3.14",
"4.2", "5.1"]
    ///
    ///
    /// - Parameters:
    ///   - column: An ``MLDataColumn`` of elements convertible to
    ///     <doc://com.apple.documentation/documentation/swift/string>.
public init T      MLDataColumn T  where T
MLDataValueConvertible

@available macOS 10.14  iOS 15.0  tvOS 16.0
extension MLDataColumn where Element
MLDataValue SequenceType

    /// Creates a new column of machine learning sequences from a given
column whose elements can be converted to
    /// sequences.
    ///
    /// Use this initializer to create a column of sequences from another column.
Start by creating a column that is
    /// convertible to a column of sequences.
    ///
    /// ````swift
    /// let intSequenceString = "[1, 2, 3]"
    /// let intSequenceString2 = "[4, 5, 6]"
    /// let stringsColumn = MLDataColumn([intSequenceString,
intSequenceString2])
    ///
    /// print(stringsColumn) // Prints "[1, 2, 3]", "[4, 5,
6]"
    /// ``
    ///
    /// Then use ``MLDataColumn/init(column:)`` to convert the
column to a column of sequences.
    ///
    /// ````swift
    /// let sequenceColumn =
MLDataColumn<MLDataValue.SequenceType>(column: stringsColumn)
    /// print(sequenceColumn) // Prints [[DataValue(1),
DataValue(2), DataValue(3)],
    ///                                // [DataValue(4), DataValue(5),
DataValue(6)]]
    /// ``
    ///
    /// - Parameters:
    ///   - column: An ``MLDataColumn`` of elements convertible to
``MLDataValue/SequenceType``.
public init T      MLDataColumn T  where T
```

## MLDataValueConvertible

```
@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataColumn where Element == Int

    /// Creates a new column of arrays of integers from a given column whose
elements can be converted to an array of
    /// integers.
    ///
    /// Use this initializer to create a column of arrays of integers from another
column. Start by creating a column
    /// that is convertible to a column of arrays of integers.
    ///
    /// Then use ``MLDataColumn/init(column:) -2rxtu`` to convert the
column to a column of arrays of integers.
    ///
    /// ````swift
    /// let intsColumn = MLDataColumn<Int>(column:
stringsColumn)
    /// print(intsColumn) // Prints [1, 2, 3, 4, 5]
    /// ````

    /// - Parameters:
    ///   - column: An ``MLDataColumn`` of elements convertible to an
    ///     <doc://com.apple.documentation/documentation/swift/array> of
    ///     <doc://com.apple.documentation/documentation/swift/int>.
public init T == MLDataColumn T where T : MLDataValueConvertible
```

## @available macOS 10.14 iOS 15.0 tvOS 16.0

```
extension MLDataColumn where Element == Double
```

```
    /// Creates a new column of arrays of doubles from a given column whose
elements can be converted to an array of
    /// doubles.
    ///
    /// Use this initializer to create a column of arrays of doubles from another
column. Start by creating a column
    /// that is convertible to a column of arrays of doubles.
    ///
    /// ````swift
    /// let doubleArrayString = "[1.0, 2.0, 3.0]"
    /// let doubleArrayString2 = "[4.0, 5.0, 6.0]"
    /// let stringsColumn = MLDataColumn([doubleArrayString,
doubleArrayString2])
    /// print(stringsColumn) // Prints "[[1.0, 2.0, 3.0],"
    /// "[4.0, 5.0, 6.0]]"
    /// ````
```

```

    /**
     /// Then use ``MLDataColumn/init(column:)--23pmx`` to convert the
     column to a column of arrays of doubles.
    /**
     /// ``swift
     /// let doubleArrayColumn = MLDataColumn<[Double]>(column:
     stringsColumn)
     /// print(doubleArrayColumn) // Prints [[1.0, 2.0, 3.0],
     [4.0, 5.0, 6.0]]
    /**
    /**
     /// - Parameters:
     ///   - column: An ``MLDataColumn`` of elements convertible to an
     ///     <doc://com.apple.documentation/documentation/swift/array> of
     ///     <doc://com.apple.documentation/documentation/swift/double>.
    public init T MLDataColumn T where T
MLDataValueConvertible

```

**@available macOS 10.14 iOS 15.0 tvOS 16.0**  
**extension** MLDataColumn **where** Element **String**

```

    /// Creates a new column of arrays of strings from a given column whose
    elements can be converted to an array of
    /// strings.
    /**
    /// Use this initializer to create a column of arrays of strings from another
    column. Start by creating a column
    /// that is convertible to a column of arrays of strings.
    /**
    /// ``swift
    /// let stringArrayString = "[\"Array\", \"of\",
    \"strings\", \"1\"]"
    /// let stringArrayString2 = "[\"Array\", \"of\",
    \"strings\", \"2\"]"
    /// let stringsColumn = MLDataColumn([stringArrayString,
    stringArrayString2])
    /**
    /// print(stringsColumn) // Prints [[["Array", "of",
    "strings", "1"]], [[["Array", "of", "strings", "2"]]]]
    /**
    /**
     /// Then use ``MLDataColumn/init(column:)--8uzuq`` to convert the
     column to a column of arrays of strings.
    /**
    /// ``swift
    /// let stringArrayColumn = MLDataColumn<[String]>(column:
    stringsColumn)
     /// print(stringArrayColumn) // Prints [[["Array", "of",
     "strings", "1"], ["Array", "of", "strings", "2"]]]

```

```

////
////
/// - Parameters:
///   - column: An ``MLDataColumn`` of elements convertible to an
///     <doc://com.apple.documentation/documentation/swift/array> of
///     <doc://com.apple.documentation/documentation/swift/string>.
public init T      MLDataColumn T  where T
MLDataValueConvertible

@available macOS 10.14  iOS 15.0  tvOS 16.0
extension MLDataColumn where Element
MLDataValue DictionaryType

    /// Creates a new column of machine learning dictionaries from a given
    column whose elements can be converted to
    /// dictionaries.
    ///
    /// Use this initializer to create a column of dictionaries from another column.
Start by creating a column that
    /// is convertible to a column of dictionaries.
    ///
    /// swift
    /// let intDictionaryString = "{1:\"one\", 2:\"two\",
3:\"three\"}"
    /// let intDictionaryString2 = "{4:\"four\", 5:\"five\",
6:\"six\"}"
    /// let stringsColumn = MLDataColumn([intDictionaryString,
intDictionaryString2])
    ///
    /// print(stringsColumn) // Prints [{"1:"one", 2:"two",
3:"three"}, {"4:"four", 5:"five", 6:"six"}]
    ///
    ///
    /// Then use ``MLDataColumn/init(column:)>-s8g5`` to convert the
column to a column of dictionaries.
    ///
    /// swift
    /// let dictionaryColumn =
MLDataColumn<MLDataValue.DictionaryType>(column:
stringsColumn)
    /// print(dictionaryColumn) // Prints [[1: one, 2: two, 3:
three], [5: five, 4: four, 6: six]]
    ///
    ///
/// - Parameters:
///   - column: An ``MLDataColumn`` of elements convertible to
``MLDataValue/DictionaryType``.
public init T      MLDataColumn T  where T
MLDataValueConvertible

```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataTable CustomReflectable

    /// The custom mirror for this instance.
    ///
    /// If this type has value semantics, the mirror should be unaffected by
    /// subsequent mutations of the instance.
    public var customMirror Mirror get

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataTable CustomStringConvertible
CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of the column.
    public var description String get

    /// A text representation of the column for debugging.
    public var debugDescription String get

    /// A description of the column shown in a playground.
    public var playgroundDescription Any get

    /// A table of data for training or evaluating a machine learning model.
    ///
    /// ``MLDataTable`` is Create ML's version of a spreadsheet in which each row
    /// represents an entity (such as a book, in the example below) with observable
    /// features. Each column (``MLDataTableColumn`` or ``MLUntypedColumn``) in
    /// the table
    /// represents an observable feature of that entity, such as a book's title or
    /// author.
    ///
    /// ![A table of information about a book. Columns named "Title", "Author",
    /// "Pages", and "Genre". The first row is "Alice in Wonderland", "Lewis
    /// Carroll", "124", and "Fantasy".](MLDataTable-1)
    ///
    /// In most cases you interact with columns using the typed ``MLDataTableColumn``,
    /// especially when you need to directly access the contents of a column. You
    /// can also interact with columns using ``MLUntypedColumn``, if the
    /// underlying
    /// type of the column isn't important.
    ///
    /// After you create a data table, you can modify it with methods like
    /// ``MLDataTable/append(contentsOf:)``,
    /// ``MLDataTable/addColumn(_:_named:)``, and
    /// ``MLDataTable/removeColumn(named:)``. You can also filter or map the
```

```

/// contents of the data table to derive new data tables or new columns by using
/// various subscripts and methods like
``MLDataTable/dropDuplicates()`` or
/// ``MLDataTable/map(_:)``-92wrj``.
///
/// - Note: For a demonstration that creates and uses data tables, see
///
<doc://com.apple.documentation/documentation/createlm/creating_a_model_from_t
abular_data>.
///
/// Finally, when your data table is ready, use it to train and evaluate a model
/// from these groups:
///
/// - Regressors like ``MLRegressor`` and its supporting types
/// - Classifiers like ``MLClassifier`` and its supporting types
/// - Natural language processing types like ``MLTextClassifier`` and
``MLWordTagger``.
///
/// - Note: It's easier to train an ``MLTextClassifier`` from folders and
files
/// with
``MLTextClassifier/init(trainingData:parameters:)``-8n8vs`` if your
data
/// is ready to use, as-is. Otherwise, use a data table to prepare your data
/// before training a text classifier.
@available macOS 10.14 iOS 15.0 tvOS 16.0
public struct MLDataTable

    /// The underlying error present when the data table is invalid.
public var error any Error get

    /// A Boolean value that indicates whether the data table is valid.
    ///
    /// Check ``MLDataTable/isValid`` after you create or mutate a data
table to
    /// ensure the data table is still valid. If it's
    /// <doc://com.apple.documentation/documentation/swift/false>, the data
    /// table encountered an error and you can't use it for training or
    /// subsequent operations. The following are some of the actions that can
    /// invalidate a data table:
    ///
    /// - Adding a column with the name of a column that already exists in the
data table
    /// - Adding an invalid column
    /// - Appending another data table with different column names
    /// - Appending another data table with different column types (for a given
column name)
public var isValid Bool get

    /// The number of rows and columns in the data table.

```

```

////
/// - Parameters:
///   - rows: The number of rows in the data table.
///   - columns: The number of columns in the data table.
public var size           Int           Int      get

/// The type of the data in each column.
///
/// The keys in the dictionary provided by this column correspond to the
/// names of the columns in the data table.
public var columnTypes  String  MLDataTable.ValueType
get

/// Creates an empty table containing no rows or columns.
///
/// Use this initializer to create an empty data table. Then, you add data
/// columns with ``MLDataTable/addColumn(_:named:)``-kkbw``, untyped
/// columns
/// with ``MLDataTable/addColumn(_:named:)``-9cb24``, or another
/// table with
/// ``MLDataTable/append(contentsOf:)``.
public init

/// Creates a data table from a dictionary of column names and untyped
/// columns.
///
/// Use this initializer to create a data table from untyped columns.
///
/// ![A table of information about a book. Columns named "Title", "Author",
/// "Pages", and "Genre". The first row is "Alice in Wonderland", "Lewis
/// Carroll", "124", and "Fantasy".](MLDataTable-init(namedColumns:)-1)
///
/// For example, to create a data table as shown above, first create your
/// untyped columns.
///
/// ````swift
/// let pages = MLUntypedColumn([124, 98, 280, 94])
/// let genre = MLUntypedColumn(["Fantasy", "Drama",
/// "Adventure", "Fantasy"])
/// let title = MLUntypedColumn(["Alice in Wonderland",
/// "Hamlet", "Treasure Island", "Peter Pan"])
/// let author = MLUntypedColumn(["Lewis Carroll",
/// "William Shakespeare", "Robert L. Stevenson", "J. M. Barrie"])
/// ````

/// Then, use ``MLDataTable/init(namedColumns:)`` to create a
/// data table
/// from the columns paired with their names.
///
/// ````swift

```

```

    /// let bookTable = try MLDataTable(namedColumns:
    ["Title": title,
     /**
     "Author": author,
     /**
     "Pages": pages,
     /**
     "Genre": genre])
     /**
     /**
     /**
     - Parameters:
     - namedColumns: The dictionary of each column name and its
       associated untyped column data.
public init           String      MLUntypedColumn
throws

    /**
     Creates a data table from a dictionary of column names and data values.
    /**
    /**
     Use this initializer to create a data table from an in-memory
     <doc://com.apple.documentation/documentation/swift/dictionary>.
    /**
    /**
     ! [A table of information about a book. Columns named "Title", "Author",
     "Pages", and "Genre". The first row is "Alice in Wonderland", "Lewis
     Carroll", "124", and "Fantasy".](MLDataTable-init(dictionary:)-1)
    /**
    /**
     For example, to create a data table as shown above, first create a
     dictionary.
    /**
    /**
     swift
    /**
    /**
     let data: [String: MLDataTableValueConvertible] = [
     "Title": ["Alice in Wonderland", "Hamlet",
     "Treasure Island", "Peter Pan"],
     "Author": ["Lewis Carroll", "William Shakespeare",
     "Robert L. Stevenson", "J. M. Barrie"],
     "Pages": [124, 98, 280, 94],
     "Genre": ["Fantasy", "Drama", "Adventure",
     "Fantasy"]
     ]
    /**
    /**
    /**
     Then, use ``MLDataTable/init(dictionary:)`` to create a data
     table from
    /**
     the dictionary.
    /**
    /**
     swift
    /**
    /**
     let bookTable = try MLDataTable(dictionary: data)
    /**
    /**
    /**
     The keys of the dictionary become the column names, and the value of
     each key becomes the element(s) of the corresponding column in the

```

```

data
    /// table.
    ///
    /// - Parameters:
    ///   - dictionary: The dictionary of each column name and its
associated data values.
public init           String     any
MLDataValueConvertible throws

    /// Retrieves or adds an untyped column with the specified name.
    ///
    /// Use this subscript to retrieve an ``MLUntypedColumn`` or add a new
one
    /// to the data table.
    ///
    /// - Important: The number of elements in the column must equal the
number
    ///   of rows in the data table. Otherwise, the data table will be
    ///   invalidated.
    ///
    /// ! [A table on the left with two columns plus a third semi-opaque column .
    /// The first two columns are named "Planet" and "Distance (AU)" (for
    /// Astronomical Units). The third column has a dash where its name should
    /// be. The first column has values like "Mercury", "Venus", and "Earth";
    /// the second column has values like "0.39", "0.72", and "1.00"; and the
    /// third column has dashes where its values should be. An arrow points to
    /// the right from the table to a second table with a same "Planet" and
    /// "Distance (AU)" columns along with a third, fully opaque column named
    /// "Distance (km)" (for kilometers). The third column contains values such
    /// as "5834170".] (MLDataTable-subscript(_):-3wjk-1)
    ///
    /// For example, to extract, convert, and add a column as shown above,
begin
    /// by creating a data table.
    ///
    /// ````swift
    /// let data: [String: MLDataValueConvertible] = [
    ///   "Planet": ["Mercury", "Venus", "Earth", "Mars",
    "Jupiter", "Saturn", "Uranus", "Neptune"],
    ///   "Distance (AU)": [0.39, 0.72, 1.00, 1.52, 5.20,
9.54, 19.22, 30.06]
    /// ]
    ///
    /// var table = try MLDataTable(dictionary: data)
    ///
    ///
    /// After creating the table, extract a column.
    ///
    /// ````swift
    /// let distanceInAU = table["Distance (AU)"]

```

```

////
////
/// - Note: Without a type annotation, the compiler uses this version of
///   `subscript(_:)` instead of the equivalent
///   ``MLDataTable/subscript(_:)--5tl9r`` from
``ML DataColumn``.
///
/// Use the untyped column's multiplication operator to create a new column
/// of data.
///
/// ``swift
/// // Create a new column of Doubles by multiplying all
the the values in
/// // `distanceInAU` by 149,597,870.7 (the number of km
in an astronomical unit)
/// let distanceInKm = (distanceInAU * 149_597_870.700)
///
///
/// Finally, use the `subscript(_:)` to add the new column to a table with
a
/// different name.
///
/// ``swift
/// table["Distance (km)"] = distanceInKm
///
///
/// - Important: To replace a column, use
///   ``MLDataTable/removeColumn(named:)`` before adding its
replacement.
/// Adding a column with the same name as one already in the data table
may
/// invalidate the data table.
///
/// - Parameters:
///   - columnName: The name of the column to extract from or add to
the datatable.
///
/// - Returns: A new ``MLUntypedColumn`` with the specified name,
if it
/// exists; otherwise an invalid column.
public subscript String MLUntypedColumn

/// Retrieves or adds a typed column with the specified name.
///
/// Use this subscript to retrieve an ``ML DataColumn`` or add a new one
to
/// the data table.
///
/// - Important: The number of elements in the column must equal the
number

```

```

    /// of rows in the data table. Otherwise, the data table will be
    /// invalidated.
    ///
    /// ! [A table on the left with two columns plus a third semi-opaque column .
    /// The first two columns are named "Planet" and "Distance (AU)" (for
    /// Astronomical Units). The third column has a dash where its name should
    /// be. The first column has values like "Mercury", "Venus", and "Earth";
    /// the second column has values like "0.39", "0.72", and "1.00"; and the
    /// third column has dashes where its values should be. An arrow points to
    /// the right from the table to a second table with a same "Planet" and
    /// "Distance (AU)" columns along with a third, fully opaque column named
    /// "Distance (km)" (for kilometers). The third column contains values such
    /// as "5834170".] (MLDataTable-subscript(_):-5t/9r-1)
    ///
    /// For example, to extract, convert, and add a column as shown above,
begin
    /// by creating a data table.
    ///
    /// ````swift
    /// let data: [String: MLDataTableValueConvertible] = [
    ///     "Planet": ["Mercury", "Venus", "Earth", "Mars",
    "Jupiter", "Saturn", "Uranus", "Neptune"],
    ///     "Distance (AU)": [0.39, 0.72, 1.00, 1.52, 5.20,
9.54, 19.22, 30.06]
    /// ]
    ///
    /// var table = try MLDataTable(dictionary: data)
    /// ````

    /// After creating the table, extract a column.
    ///
    /// ````swift
    /// let distanceInAU: MLDataTableColumn<Double> =
table["Distance (AU)"]
    /// ````

    /// - Note: By explicitly adding the type annotation
`MLDataTableColumn<Double>`,
    /// the compiler uses this version of `subscript(_:)` instead of the
    /// equivalent ``MLDataTable/subscript(_:)`` from
``MLUntypedColumn``.
    ///
    /// Use the column's multiplication operator to create a new column of data.
    ///
    /// ````swift
    /// // Create a new column of Doubles by multiplying all
the the values in
    /// // `distanceInAU` by 149,597,870.7 (the number of km
in an astronomical unit)

```

```

    /// let distanceInKm = (distanceInAU * 149_597_870.700)
    ///
    ///
    /// Finally, use the `subscript(_:)` to add the new column to the table
with
    /// a different name.
    ///
    /// ````swift
    /// table["Distance (km)"] = distanceInKm
    ///
    ///
    /// - Important: To replace a column, use
    ///   ``MLDataTable/removeColumn(named:)`` before adding its
replacement.
    /// Adding a column with the same name as one already in the data table
may
    /// invalidate the data table.
    ///
    /// - Parameters:
    ///   - columnName: The name of the column to extract from or add to
the data table.
    ///
    /// - Returns: A new ``ML DataColumn`` with the specified name and
inferred
    /// type, if it exists; otherwise an invalid column.
public subscript Element String
ML DataColumn Element where Element ML DataValueConvertible

    /// Retrieves a column with the specified name and type.
    ///
    /// Use this subscript to get a typed ``ML DataColumn``, which is easier to
    /// work with than a ``ML Untyped Column`` returned from
    /// ``ML DataTable/subscript(_:)``.
    ///
    /// - Parameters:
    ///   - columnName: The name of the column to extract.
    ///   - columnType: The underlying type of the column's content.
    ///
    /// - Returns: A new ``ML DataColumn`` with the specified name and
type, if
    /// it exists; otherwise `nil`.
public subscript T String
T ML DataColumn T where T ML DataValueConvertible
get

    /// Creates a subset of the table given a sequence of column names.
    ///
    /// - Parameters:
    ///   - columnNames: A sequence of names indicating which columns to
include in the new data table.

```

```

////
/// - Returns: A new data table.
public subscript S           S      MLDataTable where S
Sequence S Element   String  get

/// Creates a subset of the table given a range of rows.
///
/// - Parameters:
///   - slice: A range of integers indicating which rows to include in the
new data table.
///
/// - Returns: A new data table.
public subscript      Range Int      MLDataTable  get

/// Creates a subset of the table given a range expression of rows.
///
/// - Parameters:
///   - slice: An interger range expression indicating which rows to
include in the new data table.
///
/// - Returns: A new data table.
public subscript R           R      MLDataTable where R
RangeExpression R Bound    Int  get

/// Adds an untyped column to the table.
///
/// Use this method to add an untyped column to a data table.
///
/// - Important: The number of elements in the column must equal the
number
///   of rows in the data table. Otherwise, the data table will be
///   invalidated.
///
/// As an example, start with a data table variable.
///
/// ````swift
/// let data: [String: MLDataTableValueConvertible] = [
///   "Title": ["Alice in Wonderland", "Hamlet",
/// "Treasure Island", "Peter Pan"],
///   "Author": ["Lewis Carroll", "William Shakespeare",
/// "Robert L. Stevenson", "J. M. Barrie"],
///   "Pages": [124, 98, 280, 94],
/// ]
///
/// var bookTable = try MLDataTable(dictionary: data)
/// ````

/// Then use ``MLDataTableaddColumn(_:named:)`` to add
a column to
/// the table.

```

```

////
//// ````swift
//// let pagesColumn = MLUntypedColumn([124, 98, 280, 94])
//// bookTable.addColumn(pagesColumn, named: "Pages")
////
////
//// - Parameters:
////   - newColumn: A column to add to the data table.
////   - named: The name of the new column.
public mutating func addColumn -
MLUntypedColumn           String

/// Adds a data column to the table.
///
/// Use this method to add a data column to a data table.
///
/// - Important: The number of elements in the column must equal the
number
///   of rows in the data table. Otherwise, the data table will be
///   invalidated.
///
/// As an example, start with a data table variable.
///
/// ````swift
/// let data: [String: MLDataValueConvertible] = [
///   "Title": ["Alice in Wonderland", "Hamlet",
/// "Treasure Island", "Peter Pan"],
///   "Author": ["Lewis Carroll", "William Shakespeare",
/// "Robert L. Stevenson", "J. M. Barrie"],
///   "Pages": [124, 98, 280, 94],
/// ]
///
/// var bookTable = try MLDataTable(dictionary: data)
/// ````

/// Then use ``MLDataTable.addColumn(_:named:)`` to add a
column to the
/// table.
///
/// ````swift
/// let pagesColumn = ML DataColumn([124, 98, 280, 94])
/// bookTable.addColumn(pagesColumn, named: "Pages")
/// ````

/// - Parameters:
///   - newColumn: A column to add to the data table.
///   - named: The name of the new column.
public mutating func addColumn Element
ML DataColumn Element           String where Element
ML DataValueConvertible

```

```

    /// Removes the column with the specified name.
    ///
    /// - Parameters:
    ///   - named: The name of the column to remove.
    public mutating func removeColumn      String

    /// Changes the name of an existing column.
    ///
    /// - Parameters:
    ///   - named: The name of the column to rename.
    ///   - to: The new name for the column.
    public mutating func renameColumn      String
String

    /// Appends the contents of the given data table to the end of this data
    /// table.
    ///
    /// - Important: The columns of both data tables must have the same
names
    ///   and types. Otherwise, the data table will be invalidated.
    ///
    /// - Parameters:
    ///   - newTable: Another data table to append to the data table.
    public mutating func append
MLDataTable

    /// Creates a subset of the table by randomly selecting the given proportion
    /// of rows.
    ///
    /// - Parameters:
    ///   - proportion: The fraction of rows to sample from the original
table.
    ///     The value must be in the range `(0.0, 1.0)` .
    ///
    ///   - seed: A number that seeds a random number generator.
    ///
    /// - Returns: A new data table.
    public func randomSample      Double      Int
42      MLDataTable

    /// Creates a subset of the table by including the rows that contain any of
    /// the given values in the given column.
    ///
    /// - Parameters:
    ///   - values: The values to include from the new table.
    ///   - columnNamed: The name of the column to search for included
values.
    ///
    /// - Returns: A new data table.

```

```

public func intersect T T  

String      MLDataTable where T MLDataTableConvertible

    /// Creates a subset of the table by excluding the rows that contain any of
    /// the given values in the given column.
    ///
    /// - Parameters:
    ///   - values: The values to exclude from the new table.
    ///   - columnNamed: The name of the column to search for excluded
values.
    ///
    /// - Returns: A new data table.
public func exclude T - T  

String      MLDataTable where T MLDataTableConvertible

    /// Creates a new data table by merging two data tables by the given
    /// columns.
    ///
    /// - Parameters:
    ///   - with: Another data table to merge with this data table.
    ///   - columnsNamed: The name of the columns to perform the `join`  

operation on. The method merges all rows with
    ///           matching values in these columns.
    ///
    ///           If you do not provide any column names, the method uses all the
columns present in both tables.
    ///
    ///   - type: The type of `join` operation, which are equivalent to SQL
`join` types.
    ///
    /// - Returns: A new data table.
public func join      MLDataTable  

String          MLDataTable JoinType           MLDataTable

    /// Join types available for ``MLDataTable`` join operations.
public enum JoinType  Sendable

    /// An operation that joins the rows of the data tables whose values
    /// match exactly.
case inner

    /// An operation that is the union between a left join and a right join.
case outer

    /// An operation that is the union between an inner join and the
    /// remaining rows from the original data table.
    ///
    /// The `.left` join type merges missing values.
case left

```

```

    /// An operation that is the union between an inner join and the
    /// remaining rows from the secondary data table.
    ///
    /// The `right` join type merges missing values.
case right

    /// Returns a Boolean value indicating whether two values are equal.
    ///
    /// Equality is the inverse of inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is `false`.
    ///
    /// - Parameters:
    ///   - `lhs`: A value to compare.
    ///   - `rhs`: Another value to compare.
public static func      MLDataTable JoinType
MLDataTable JoinType      Bool

the
    /// Hashes the essential components of this value by feeding them into
    /// given hasher.
    ///
    /// Implement this method to conform to the `Hashable` protocol. The
    /// components used for hashing must be the same as the components
    compared
        /// in your type's `==` operator implementation. Call
        `hasher.combine(_:)`
        /// with each of these components.
        ///
        /// - Important: In your implementation of `hash(into:)`,  

        ///   don't call `finalize()` on the `hasher` instance provided,  

        ///   or replace it with a different instance.  

        ///   Doing so may become a compile-time error in the future.
        ///
        /// - Parameter hasher: The hasher to use when combining the
        components
            /// of this instance.
public func hash           inout Hasher

        /// The hash value.
        ///
        /// Hash values are not guaranteed to be equal across different
        executions of
            /// your program. Do not save hash values to use during a future
        execution.
        ///
        /// - Important: `hashValue` is deprecated as a `Hashable`  

        requirement. To
            /// conform to `Hashable`, implement the `hash(into:)`  

        requirement instead.
        /// The compiler provides an implementation for `hashValue` for

```

you.

```
public var hashValue  Int    get

    /// Creates a subset of the table by masking the rows with the given untyped
    /// column.
    ///
    /// Use this untyped column-based subscript to create a new table by
masking
    /// a subset of the table rows. The derived table will not include rows
    /// where `mask` contains a default value for its underlying type, such as:
    ///
    /// - `0` in untyped `Int` columns
    /// - `0.0` in untyped `Double` columns
    /// - An empty string in untyped `String` columns
    ///
    /// The derived data table includes rows where the masking column has any
    /// other (nondefault) value.
    ///
    /// ! [A table of book information on the left with columns named "Title",
    /// "Genre", and "Pages". One row of the table has the values: "Hamlet",
    /// "Drama", and "98", respectively. The pages column of the table is
    /// highlighted. Two of its rows show a zero for the books Dracula and The
    /// Great Gatsby. All other rows have a nonzero number of pages and have
    /// arrows leading from them to all the rows in a table on the right hand
    /// side of the image. The right side table is a subset of the left side
    /// table where the rows with zero in the pages column have been
    /// removed.] (MLDataTable-subscript(_):-10r4l-1)
    ///
    /// For example, to filter the values in a data table as shown above, begin
    /// by creating a table with the original data.
    ///
    /// ````swift
    /// let data: [String: MLDataTableConvertible] = [
    ///     "Title": ["Alice in Wonderland", "Hamlet",
    "Dracula", "Peter Pan", "The Great Gatsby"],
    ///     "Genre": ["Fantasy", "Drama", "Adventure",
    "Fantasy", "Drama"],
    ///     "Pages": [124, 98, 0, 94, 0]
    /// ]
    ///
    /// guard let table = try? MLDataTable(dictionary: data)
else {
    ///     fatalError("Invalid dictionary data")
    /// }
    ///
    ///
    /// After you create the table, use the `String`-based
    /// ``MLDataTable/subscript(_):-3wjk`` to extract a column.
    ///
```

```

    /// swift
    /// // Create a new untyped column "pagesColumn" copied
    from the table's "Pages" column
    /// let pagesColumn = table["Pages"]
    ///
    ///
    /// Use `subscript(mask: MLUntypedColumn)` with an untyped
    column-row mask
    /// to create a filtered table. The subscript uses the untyped values to
    /// determine whether to keep a row.
    ///
    /// In this example, the untyped column `pagesColumn` has an underlying
    type
    /// of `Int`. The subscript removes any row with a value of `0` (the default
    /// value for `Int`) and keeps all other rows.
    ///
    /// swift
    /// // Filter the table using pagesColumn.
    /// // Each row with a value of 0 is removed.
    /// let validPageCountTable = table[pagesColumn]
    ///
    ///
    /// - Parameters:
    ///   - mask: An untyped column indicating whether rows should be
    removed (a
    ///     default value) or included (any nondefault value) in the derived table.
    ///
    /// - Returns: A new data table.
    public subscript MLUntypedColumn MLDataTable
get

    /// Creates a subset of the table by masking the rows with the given column
    /// of Booleans.
    ///
    /// Use this Boolean column-based subscript to create a new table by
    masking
    /// a subset of the table rows.
    ///
    /// ! [A table of book titles and genres such as "Hamlet" and "Drama" on the
    /// left side of the image. A one-column table titled "mask" in the middle
    /// of the image with true and false cells lined up with genres, where rows
    /// with "Fantasy" have a mask of false and all other rows have a mask of
    /// true. Arrows from the true cells in the mask column to all the rows in a
    /// table on the right hand side showing a subset of the rows in the table
    /// on the left side of the image. None of the genres in this table are
    /// "Fantasy".] (MLDataTable-subscript(_):-3opgl-1)
    ///
    /// For example, to filter the values in a data table as shown above, begin
    /// by creating a table with the original data.
    ///

```

```

    /// swift
    /// let data: [String: MLDataValueConvertible] = [
    ///     "Title": ["Alice in Wonderland", "Hamlet",
    "Treasure Island", "Peter Pan"],
    ///     "Genre": ["Fantasy", "Drama", "Adventure",
    "Fantasy"]
    /// ]
    ///
    /// let table = try? MLDataTable(dictionary: data) else {
    ///     fatalError("Invalid dictionary data")
    /// }
    ///
    ///
    /// After you create the table, use column arithmetic operators or the
    /// `MLDataTable/map(_:)` method to build a row mask. The
    subscript
    /// uses the Boolean values in the row mask to determine whether to keep a
    /// row.
    ///
    /// swift
    /// // Retrieve the "Genre" column from the table.
    /// guard let genreColumn = table["Genre", String.self]
else {
    ///     fatalError("Missing or invalid 'genre' column in
table.")
    /// }
    ///
    /// // Create a new column of Booleans by comparing all of
the values
    /// // in `Genre` with `Fantasy` using the
    /// // `!=(ML DataColumn<String>, String) ->
ML DataColumn<Bool>` operator.
    /// let noFantasyMask = genreColumn != "Fantasy"
    /// ```
    ///
    /// Use `subscript(mask: ML DataColumn<Bool>)` with the Boolean
column-row
    /// mask to create a filtered table.
    ///
    /// swift
    /// let noFantasyTable = table[noFantasyMask]
    /// ```
    ///
    /// - Parameters:
    ///   - mask: A Boolean column indicating whether rows should be kept
(`true`)
    ///       or removed (`false`) in the derived table.
    ///
    /// - Returns: A new data table.
public subscript      ML DataColumn Bool      ML DataTable

```

## get

```
    /// Creates a new column, potentially with missing values, by applying a
    /// given thread-safe transform to every row in the data table.
    ///
    /// This mapping function is the same as
``MLDataTable/map(_:)--92wrj`` with
    /// the exception that this version's `lazyTransform` parameter returns
an
    /// optional (`T?`). Use this version of `map()` to create empty values by
    /// returning `nil` from your `lazyTransform`.
    ///
    /// - Parameters:
    ///   - lazyTransform: A thread-safe row transformation function.
    ///
    ///   The implementation of your transform must accept a row from the
data table and return
    ///   an optional type that conforms to
``MLDataTableValueConvertible``. If the
    ///   transform returns `nil` for a given row, the corresponding element
in
    ///   the new column will have a missing value.
    ///
    /// - Returns: A new ``MLDataColumn``.
public func map T _ @escaping
MLDataTable Row T MLDataColumn T where T
MLDataTableValueConvertible

    /// Creates a new column by applying a given thread-safe transform to every
    /// row in the data table.
    ///
    /// Use this method to create a new column derived from the existing data in
    /// the table. The closure you pass evaluates lazily only when the
    /// transformed values are needed for a subsequent operation. Your
    /// implementation should accept a data table row and must be thread-safe
    /// because the framework may invoke the closure concurrently on
unspecified
    /// threads.
    ///
    /// ![A table on the left with "Day" and "Temperature" columns. The first
    /// column has values like "Monday" and "Tuesday", and the second column
has
    /// values like "91.3" and "85.8". An arrow points to the right from the
    /// table to a second table with a single column, "Description". It contains
    /// values such as "Monday 91.3°".](MLDataTable-map(_:)--92wrj-1)
    ///
    /// For example, to perform the column derivation operation shown above,
    /// begin by creating a table of data.
    ///
    /// ````swift
```

```

    ///> let data: [String: MLDataValueConvertible] = [
    ///>     "Day": ["Monday", "Tuesday", "Wednesday",
    "Thursday", "Friday"],
    ///>     "Temperature": [91.3, 85.8, 79.5, 83.4, 72.2]
    ///> ]
    ///>
    ///> var table = try MLDataTable(dictionary: data)
    ///> ...
    ///>
    ///> After creating the table, use ``MLDataTable/map(_:)--92wrj`` to
create a
    ///> new column of data from the original data. The example closure
    ///> implementation below is stateless and safe to invoke concurrently on any
    ///> thread, so no synchronization is necessary.
    ///>
    ///> ````swift
    ///> let derivedColumn = table.map { row -> String in
    ///>     guard let day = row["Day"]?.stringValue,
    ///>             let temperature =
    row["Temperature"]?.doubleValue else {
    ///>         fatalError("Missing or invalid columns in
row.")
    ///>     }
    ///>     return String(format: "%@ %.1f°", day,
temperature)
    ///> }
    ///> ...
    ///>
    ///> Then use ``MLDataTable.addColumn(_:named:)--kkbw`` to add
the derived
    ///> column to a table.
    ///>
    ///> ````swift
    ///> table.addColumn(derivedColumn, named: "Description")
    ///> ...
    ///>
    ///> - Parameters:
    ///> - lazyTransform: A thread-safe row transformation function.
    ///>
    ///> The implementation of your transform must accept a row from the
data table and return
    ///> a type that conforms to ``MLDataValueConvertible``.
    ///>
    ///> - Returns: A new ``MLDataTable``.
public func map T -> MLDataTable T where T
MLDataValueConvertible

    ///> Creates a subset of the table by removing any row missing one or more
    ///> values.

```

```
///  
/// - Returns: A new data table.  
public func dropMissing      MLDataTable  
  
/// Creates a modified copy of the table by filling in the missing values in  
/// the named column.  
///  
/// - Parameters:  
///   - columnNamed: The name of the column with missing values.  
///   - value: An `MLDataTable` to put in place for every missing  
value in the column.  
///  
/// - Returns: A new data table.  
public func fillMissing      String  
MLDataTable      MLDataTable  
  
/// Creates a subset of the table by removing all duplicate rows.  
///  
/// - Returns: A new data table.  
///  
/// - Note: The order of the rows may not be preserved.  
public func dropDuplicates  MLDataTable  
  
/// Creates a subset of the table given a number of initial rows.  
///  
/// - Parameters:  
///   - maxLength: The largest number of rows to use from the beginning  
of the  
///   data table. The default value is `10`.  
///  
/// - Returns: A new data table.  
public func prefix _          Int    10      MLDataTable  
  
/// Creates a subset of the table given a number of final rows.  
///  
/// - Parameters:  
///   - maxLength: The largest number of rows to use from the end of  
the data  
///   table. The default value is `10`.  
///  
/// - Returns: A new data table.  
public func suffix _          Int    10      MLDataTable  
  
/// Creates a new data table by sorting the table by the given column.  
///  
/// - Parameters:  
///   - columnNamed: The name of the column to sort the rows of data  
table.  
///  
///   - byIncreasingOrder: Set this value to
```

```

<doc://com.apple.documentation/documentation/swift/true> to sort the
    ///      table in ascending order.
    ///
    /// - Returns: A new data table.
public func sort           String
Bool   true     MLDataTable

    /// Creates a new data table where duplicate row values in the given column
    /// are condensed into a new sequence-type column.
    ///
    /// This function performs the inverse of
``MLDataTable/condense(columnNamed:to:)``.
    ///
    /// - Parameters:
    ///   - columnNamed: The name of the column to expand.
    ///   - to: The name of the new expanded column.
    ///
    /// - Returns: A new data table.
public func expand        String      String
MLDataTable

    /// Creates a new data table where duplicate row values in the given column
    /// are condensed into a new sequence-type column.
    ///
    /// This function performs the inverse of
``MLDataTable/expand(columnNamed:to:)``.
    ///
    /// - Parameters:
    ///   - columnNamed: The name of the column to condense.
    ///   - to: The name of the new condensed column.
    ///
    /// - Returns: A new data table.
public func condense      String      String
MLDataTable

```

## **extension** MLDataTable

```

    /// A collection of the names of the columns in a data table.
@available macOS 10.14  iOS 15.0  tvOS 16.0
public struct ColumnNames  RandomAccessCollection

    /// The position of the first element in a nonempty collection.
    ///
    /// If the collection is empty, `startIndex` is equal to `endIndex`.
public var startIndex  Int  get

    /// The collection's "past the end" position---that is, the position one
    /// greater than the last valid subscript argument.

```

```
///  
/// When you need a range that includes the last element of a  
collection, use  
/// the half-open range operator (`..  
` <`) with `endIndex`. The  
`..  
` operator  
/// creates a range that doesn't include the upper bound, so it's always  
/// safe to use with `endIndex`. For example:  
///  
///     let numbers = [10, 20, 30, 40, 50]  
///     if let index = numbers.firstIndex(of: 30) {  
///         print(numbers[index ..< numbers.endIndex])  
///     }  
///     // Prints "[30, 40, 50]"  
///  
/// If the collection is empty, `endIndex` is equal to `startIndex`.  
public var endIndex Int get  
  
/// Accesses the element at the specified position.  
///  
/// The following example accesses an element of an array through its  
/// subscript to print its value:  
///  
///     var streets = ["Adams", "Bryant", "Channing",  
"Douglas", "Evarts"]  
///     print(streets[1])  
///     // Prints "Bryant"  
///  
/// You can subscript a collection with any valid index other than the  
/// collection's end index. The end index refers to the position one past  
/// the last element of a collection, so it doesn't correspond with an  
/// element.  
///  
/// - Parameter position: The position of the element to access.  
`position`  
/// must be a valid index of the collection that is not equal to the  
/// `endIndex` property.  
///  
/// - Complexity: O(1)  
public subscript Int String get  
  
/// A type representing the sequence's elements.  
@available iOS 15.0 tvOS 16.0 macOS 10.14  
public typealias Element String  
  
/// A type that represents a position in the collection.  
///  
/// Valid indices consist of the position of every element and a  
/// "past the end" position that's not valid for use as a subscript  
/// argument.
```

```
@available iOS 15.0 tvOS 16.0 macOS 10.14
public typealias Index Int

    /// A type that represents the indices that are valid for subscripting the
    /// collection, in ascending order.
@available iOS 15.0 tvOS 16.0 macOS 10.14
public typealias Indices Range Int

    /// A type that provides the collection's iteration interface and
    /// encapsulates its iteration state.
///
    /// By default, a collection conforms to the `Sequence` protocol by
    /// supplying `IndexingIterator` as its associated `Iterator`
    /// type.
@available iOS 15.0 tvOS 16.0 macOS 10.14
public typealias Iterator
IndexingIterator MLDataTable ColumnNames

    /// A collection representing a contiguous subrange of this collection's
    /// elements. The subsequence shares indices with the original
collection.
///
    /// The default subsequence type for collections that don't define their
own
    /// is `Slice`.
@available iOS 15.0 tvOS 16.0 macOS 10.14
public typealias SubSequence
Slice MLDataTable ColumnNames

    /// The names of the columns in the data table.
@available macOS 10.14 iOS 15.0 tvOS 16.0
public var columnNames MLDataTable ColumnNames get

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataTable

    /// Exports a binary file of the data table to the given directory URL.
///
    /// - Parameters:
    /// - directoryURL: The directory location in the file system to which the
    /// data table file should be written.
public func write URL throws

    /// Exports a binary file of the data table to the given directory path.
///
    /// - Parameters:
    /// - path: A file system directory path where the data table file should be
    /// written.
```

```

public func write String throws
    /// Exports a CSV file of the data table to the given directory URL.
    ///
    /// - Parameters:
    /// - fileURL: The location in the file system to which the data table file
    /// should be written.
public func writeCSV URL throws
    /// Exports a CSV file of the data table to the given directory path.
    ///
    /// - Parameters:
    /// - path: A file system path where the data table file should be written.
public func writeCSV String throws

```

## **extension** MLDataTable

```

    /// Creates a data table from an imported JSON or CSV file.
    ///
    /// Use this initializer to create a data table from either a JavaScript
    /// Object Notation (JSON) file or a comma-separated values (CSV) file. Use
    /// `options` to customize how the data table imports information from
your
    /// CSV file. The data table ignores the `options` parameter when
importing
    /// JSON files.
    ///
    /// ## Create a Table from a JSON File
    ///
    /// This initializer imports data from a JSON file and creates a row from
    /// each dictionary in the root JSON array. The data table names its columns
    /// with the dictionary's keys.
    ///
    /// ! [A table of information about a book. Columns named "Title", "Author",
    /// "Pages", and "Genre". The first row is "Alice in Wonderland", "Lewis
    /// Carroll", "124", and
    /// "Fantasy"].(MLDataTable-init(contentsOf:options:-1)
    ///
    /// For example, to create a data table as shown above, build a JSON file
    /// with a root array of dictionaries that all use the same set of keys.
    ///
    /// ````swift
    /// /*
    /// books.json file:
    /// [
    ///     {
    ///         "Title": "Alice in Wonderland",
    ///         "Author": "Lewis Carroll",
    ///         "Pages": 124,

```

```

    /**
     *      "Genre": "Fantasy"
     */
    },
    {
        "Title": "Hamlet",
        "Author": "William Shakespeare",
        "Pages": 98,
        "Genre": "Drama"
    },
    {
        "Title": "Treasure Island",
        "Author": "Robert L. Stevenson",
        "Pages": 280,
        "Genre": "Adventure"
    },
    {
        "Title": "Peter Pan",
        "Author": "J. M. Barrie",
        "Pages": 94,
        "Genre": "Fantasy"
    }
]
*/
```
/// Then, use ``MLDataTable/init(contentsOf:options:)`` to
create the data
/// table.
```
swift
let bookTable = try MLDataTable(contentsOf: jsonUrl)
```
/// Each entry in JSON array becomes a row in the data table. The set of
/// keys in the JSON dictionaries become the column names in the data
table.
```
## Create a Table from a CSV File
```
/// This initializer imports data from a CSV file and creates a row from
/// each line in the CSV file. You can create a CSV file programmatically or
/// use an app like Numbers to export a spreadsheet.
```
For best results, provide your own parsing options configured to match
your CSV file, especially if it has a custom format. See
```
MLDataTable/ParsingOptions
```
for the full list of CSV import
customization options.
```
If you don't provide parsing options, it is assumed that your CSV file

```

```

/// has the following formatting attributes:
///
/// - The first entry is a header row with the names of the columns. - Data
/// fields are separated by a comma (` , `). - Each row ends with a newline
/// character (`\n`). - Special characters are escaped with a leading
/// backslash (`\ `). - Every quote literal (`"`) is represented by two
/// consecutive quotes (`""`).
///
/// ! [A table of information about a book. Columns named "Title", "Author",
/// "Pages", and "Genre". The first row is "Alice in Wonderland", "Lewis
/// Carroll", "124", and
/// "Fantasy".](MLDataTable-init(contentsOf:options:)-2)
///
/// For example, to create a data table as shown above, first create a CSV
/// file programmatically or use a spreadsheet app like Numbers to export
/// one, and add it to your project.
///
/// ````swift
/// /*
/// books.csv file:
/// This example skips these first 3 rows, starting with
this one.
/// Use skipRows to ignore any lines above the data.
///
/// Title,Author,Pages,Genre
/// Alice in Wonderland,Lewis Carroll,124,Fantasy
/// Hamlet,William Shakespeare,98,Drama
/// Treasure Island,Robert L. Stevenson,280,Adventure
/// Peter Pan,J. M. Barrie,94,Fantasy
/// */
///
/// let csvURL = URL(fileURLWithPath: "books.csv")
/// ````

///
/// Then, configure your parsing options to match the format of your CSV
/// file. In this example, the initializer must ignore the first three lines
/// of the CSV file beginning with "This example...".
///
/// ````swift
/// // Configure CSV file parsing options
/// var parsingOptions = MLDataTable.ParsingOptions()
/// parsingOptions.skipRows = 3
/// parsingOptions.containsHeader = true
/// parsingOptions.delimiter = ","
/// parsingOptions.lineTerminator = "\n"
/// ````

///
/// Lastly, use ``MLDataTable/init(contentsOf:options:)`` to
create the data

```

```

    /// table.
    ///
    /// `` swift
    /// let bookTable = try MLDataTable(contentsOf: csvURL,
options: parsingOptions)
    ///
    ///
    /// The header row becomes the column names in the data table. If the CSV
    /// file doesn't have a header row, the column names become "X1", "X2", etc.
    /// Each subsequent line of the CSV file becomes a row in the data table.
    ///
    /// - Parameters:
    ///   - url: The URL of the JSON or CSV file to import.
    ///   - options: The parsing options to use when importing a CSV file;
ignored
    ///      when importing JSON files.
@available macOS 10.14 iOS 15.0 tvOS 16.0
public init URL
MLDataTable ParsingOptions throws

    /// The options for parsing a comma-separated values (CSV) file into a data
    /// table for a machine learning model.
    ///
    /// Use `ParsingOptions` only when importing a CSV file with
    /// ``MLDataTable/init(contentsOf:options:)``, especially if your
CSV file
    /// has special formatting or your data table only needs to import specific
    /// rows or columns.
@available macOS 10.14 iOS 15.0 tvOS 16.0
public struct ParsingOptions : Sendable

    /// A Boolean value indicating whether an input CSV file contains a
    /// header.
    ///
    /// Set `containsHeader` to `false` when the first row in a CSV
contains
    /// usable data. Because every column in a data table needs a name,
    /// `MLDataTable` names the columns `X1`, `X2`, ... `X_n` in
the same
    /// order as they appear in the CSV file.
public var containsHeader Bool

    /// The character that separates the data fields in a CSV file.
public var delimiter String

    /// The character that marks the beginning of a comment, or text to
    /// ignore, in a CSV file.
public var comment String

    /// The character that marks a C escape sequence in a CSV file.

```

```

public var escape String

/// A Boolean value indicating whether two consecutive quotes (`""`)
/// represent a single quote (`"`) in a CSV file.
public var doubleQuote Bool

/// The character that represents a quote (`"`) in a CSV file.
public var quote String

/// A Boolean value indicating whether to ignore leading spaces of a
/// data field.
public var skipInitialSpaces Bool

/// A list of strings that represent missing values in a CSV file.
public var missingValues String

/// The character that represents the end of a line in a CSV file.
public var lineTerminator String

/// The list of column names to import from a CSV file; otherwise `nil`
/// to import all columns.
public var selectColumns String

/// The maximum number of rows to import form a CSV file; otherwise
/// `nil` to import all rows.
public var maxRows Int

/// The number of starting rows to skip from the start of a CSV file.
public var skipRows Int

/// Creates CSV parsing options.
public init String ","
String Bool Bool String Bool String
String Bool Bool String String
String String String String
String nil Int nil Int

```

**@available macOS 10.14 iOS 15.0 tvOS 16.0**  
**extension** MLDataTable

```

/// The rows of data in the table.
public var rows MLDataTable.Rows get

/// A collection of rows in a data table.
public struct Rows

```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataTable

    /// A row of untyped values in a data table.
public struct Row

        public typealias Keys MLDataTable ColumnNames
        public typealias Key MLDataTable Row Keys Element
        public typealias Value
MLDataTable Row Values Element

        public var keys MLDataTable Row Keys get
        public var values MLDataTable Row Values get
        public subscript MLDataTable Row Key
MLDataTable Row Value get
        public subscript T MLDataTable Row Key
T where T MLAttributeValueConvertible get
```

## **extension** MLDataTable

```
    /// Creates a new data table with the given columns and adds a new column
    /// for each of the given aggregators.
    ///
    /// - Parameters:
    /// - columnsNamed: The name of the columns to include in the new data
    /// table.
    ///
    /// - aggregators: A sequence of aggregators, each of which adds a column
in
    /// the new data table.
    ///
    /// - Returns: A new data table.
@available macOS 10.14 iOS 15.0 tvOS 16.0
public func group S String
S MLDataTable where S Sequence S Element
MLDataTable Aggregator

    /// A collection of column operations you can use with a data table's
    /// `group` method.
    ///
```

```
    /// Use one or more ``MLDataTable/Aggregator`` instances when you
use the
    /// ``MLDataTable/group(columnsNamed:aggregators:)`` method.
Each aggregator
    /// performs a sequence of one or more
    /// ``MLDataTable/Aggregator/Operations-swift.enum`` for a
single column.
    /// Each operation makes a new column in the new data table generated by
the
    /// method.
@available macOS 10.14 iOS 15.0 tvOS 16.0
public struct Aggregator : Sendable

    /// An array of operations to perform on a column.
public var operations
MLDataTable Aggregator Operations

    /// The name of the column on which the aggregator performs
operations
    /// on.
public var columnName : String

    /// Creates an aggregator with the given operations to perform on the
    /// given column.
public init
MLDataTable Aggregator Operations : String

    /// The operations that an aggregator can perform on a column in a
data
    /// table.
public enum Operations : Sendable

        /// An operation that identifies the smallest value in a column.
case min

        /// An operation that identifies the largest value in a column.
case max

        /// An operation that adds the values in a column.
case sum

        /// An operation that computes the mean of the values in a
column.
case mean

        /// An operation that computes the standard deviation of the
values
        /// in a column.
case stdev
```

```
    /// An operation that counts the number of values in a column.  
    case count  
  
    /// An operation that counts the number of distinct values in a  
    /// column.  
    case distinctCount  
  
    /// An operation that computes the variance in a column.  
    case variance  
  
    /// An operation that combines the values in a column into a  
    /// sequence.  
    case sequenceMerge  
  
    /// An operation that selects a random value from a column.  
    case randomlySelectOne  
  
    /// An operation that combines two columns into a dictionary,  
using  
    /// the given column as the values for that dictionary.  
    ///  
    /// The dictionary keys are the values in the column that you  
    /// specified with the aggregator's  
    /// ``MLDataTable/Aggregator/columnName`` property.  
    ///  
    /// - Parameters:  
    /// - valueColumn: The name of the column that contains the  
values  
    /// of the dictionary.  
    case dictionaryMerge String  
  
    /// An operation that retrieves the value in the given column that's  
    /// in the same row as the minimum value of the aggregator's  
column.  
    ///  
    /// Use this operation to find a value in the given  
`outputColumn`  
    /// that's in the same row as the minimum value in the  
aggregator's  
    /// column. For example, take the following data table of drink  
    /// ratings.  
    ///  
    /// ````swift  
    /// let teaVsCoffee = try! MLDataTable(dictionary:  
[  
    ///     "userName" : ["Sara", "Sara", "James",  
    "James"],  
    ///     "drink"     : ["tea", "coffee", "tea",  
    "coffee"],  
    ///     "rating"   : [5.0, 3.5, 3.1, 4.9]
```

```

    ////
    ///
    /// print(teaVsCoffee)
    ///
    /// Prints ...
    /// Columns:
    ///   ratingfloat
    ///   drinkstring
    ///   userNamestring
    /// Rows: 4
    /// Data:
    /// +-----+
+-----+           /// | drink          | userName      | rating
|           |           |             |             |
+-----+           /// +-----+
+-----+           /// | tea            | Sara         | 5
|           |           |             |             |
|           /// | coffee         | Sara         | 3.5
|           |           |             |             |
|           /// | tea            | James        | 3.1
|           |           |             |             |
|           /// | coffee         | James        | 4.9
|           |           |             |             |
+-----+           /// +-----+
+-----+           /// [4 rows x 3 columns]
/// ``
///
/// To find out which person gave the lowest rating for any
/// particular beverage, use the `argmin` operation with the
/// `'"userName"'` string. Then use the
///
``MLDataTable/group(columnsNamed:aggregators:)`` function group
/// the `'"drink"'` column.
///
/// ``swift
/// var lowestRater =
MLDataTable.Aggregator(operations: .argmin(outputColumn:
"userName"),
///                                     of:
"rating")
/// let lowestDrinkRatings =
teaVsCoffee.group(columnsNamed: "drink",
/// aggregators: [lowestRater])
/// print(lowestDrinkRatings)
///

```

```

    /// Prints ...
    /// +-----+-----+
    /// | drink      | ratingArgmin |
    /// +-----+-----+
    /// | tea        | James       |
    /// | coffee     | Sara        |
    /// +-----+-----+
    /// [2 rows x 2 columns]
    ///
    ///
    /// In this example, the drinks column only has two distinct drinks,
    /// tea and coffee, which result in a data table with two rows.
    ///
    /// - Parameters:
    ///   - outputColumn: The name of the column that holds the
values
    ///   associated with the minimum values of the aggregator's
    ///   designated column.
    case argmin           String

    /// An operation that retrieves the value in the given column that's
    /// in the same row as the maximum value of the aggregator's
column.
    ///
    /// Use this operation to find a value in the given
`outputColumn`
    /// that's in the same row as the maximum value in the
aggregator's
    /// column. For example, take the following data table of drink
    /// ratings.
    ///
    /// `` swift
    /// let teaVsCoffee = try! MLDataTable(dictionary:
[
    ///   "userName" : ["Sara", "Sara", "James",
"James"],
    ///   "drink"    : ["tea", "coffee", "tea",
"coffee"],
    ///   "rating"   : [5.0, 3.5, 3.1, 4.9]
    /// ])
    ///
    /// print(teaVsCoffee)
    ///
    /// Prints ...
    /// Columns:
    ///   ratingfloat
    ///   drinkstring
    ///   userNamestring
    /// Rows: 4 Data:
    /// +-----+

```

```

+-----+
|     | drink      | userName    | rating
|     |
|     +-----+
+-----+
|     | tea       | Sara        | 5
|     |
|     | coffee    | Sara        | 3.5
|     |
|     | tea       | James       | 3.1
|     |
|     | coffee    | James       | 4.9
|     |
|     +-----+
+-----+
|     [4 rows x 3 columns]
|     ``
|     ``
|     ``
|     /// To find out which person gave the highest rating for any
|     /// particular beverage, use the `argmax` operation with the
|     /// `'"userName"'` string. Then use the
|     ///
``MLDataTable/group(columnsNamed:aggregators:)`` function group
    /// the `'"drink"'` column.
    ///
    /// ``swift
    /// var highestRater =
MLDataTable.Aggregator(operations: .argmax(outputColumn:
"userName"),
    ///                                         of:
"rating")
    ///
    /// let highestDrinkRatings =
teaVsCoffee.group(columnsNamed: "drink",
    ///
aggregators: [highestRater])
    /// print(highestDrinkRatings)
    ///
    /// Prints ...
    /// +-----+
    /// | drink      | ratingArgmax |
    /// +-----+
    /// | tea       | Sara        |
    /// | coffee    | James       |
    /// +-----+
    /// [2 rows x 2 columns]
    ///
    ///
    /// In this example, the drinks column only has two distinct drinks,

```

```
    /// tea and coffee, which result in a data table with two rows.  
    ///  
    /// - Parameters:  
    /// - outputColumn: The name of the column that holds the  
values  
    /// associated with the maximum values of the aggregator's  
    /// designated column.  
    case argmax String
```

## extension MLDataTable

```
@available macOS 11.0 iOS 15.0 tvOS 16.0  
public func randomSplitBySequence Double  
String String Int  
1 MLDataTable MLDataTable
```

```
    /// Creates two mutually exclusive, randomly divided subsets of the table.  
    ///  
    /// - Parameters:  
    /// - proportion: A value between `0.0` and `1.0` indicating the  
fraction of  
    /// rows to go into one subset. The remaining rows go into the other subset.  
    ///  
    /// - seed: A random number generator seed. The default value is `1`.  
    ///  
    /// - Returns: Two new data tables.
```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0  
public func randomSplit Double Int  
1 MLDataTable MLDataTable
```

```
    /// Randomly split a MLDataTable into a number partitions while stratifying on  
a user-defined label column.
```

```
    ///  
    /// The proportions specified will be applied uniformly to each label being  
partitioned on.  
    ///  
    /// - Parameter proportions: An array of values on [0,1] specifying  
the proportions in each partition. Automatically normalized to 1.  
    /// - Parameter column: The column in an MLDataTable being stratified  
on.  
    /// - Parameter seed: Seed for the random number generator used for  
splitting. The default seed is the current epoch time in milliseconds.
```

```
    ///  
    /// - Returns: A new MLDataTable with an additional partition column  
with the index of the partition for each row.
```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0  
public func stratifiedSplit Double
```

```

String           Int           throws
MLDataTable

    /// Randomly split a MLDataTable into a number partitions while stratifying on
    // a user-define label column.
    /**
     /// The proportions specified will be applied uniformly to each label being
     // partitioned on.
     /**
      /// - Parameter proportions: An array of values on [0,1] specifying
      // the proportions in each partition. Automatically normalized to 1.
      /// - Parameter column: The column in an MLDataTable being stratified
      // on.
      /// - Parameter generator: User-defined RandomNumberGenerator to
      // use in stratification.
      /**
      /// - Returns: A new MLDataTable with an additional partition column
      // with the index of the partition for each row.
      @available macOS 10.14 iOS 15.0 tvOS 16.0
      public func stratifiedSplit RNG           Double
          String           inout RNG   throws   MLDataTable
      where RNG   RandomNumberGenerator

    /// Randomly split a MLDataTable into partitions on a user-define label
    // column, while keeping rows from the same sequence in the original order.
    /**
     /// The proportions specified will be applied uniformly to each label being
     // partitioned on.
     /**
      /// - Parameter proportions: An array of values on [0,1] specifying
      // the proportions in each partition. Automatically normalized to 1.
      /// - Parameter sequenceIdentifierColumn: The sequence
      // identifier column in an MLDataTable to identify rows of a sequence.
      /// - Parameter column: The column in an MLDataTable being stratified
      // on.
      /// - Parameter generator: User-defined RandomNumberGenerator to
      // use in stratification.
      /**
      /// - Returns: A new MLDataTable with an additional partition column
      // with the index of the partition for each row.
      @available macOS 11.0 iOS 15.0 tvOS 16.0
      public func stratifiedSplitBySequence RNG
          Double           String
          String           inout RNG   throws   MLDataTable where
      RNG   RandomNumberGenerator

    /// Randomly split a MLDataTable into partitions on a user-define label
    // column, while keeping rows from the same sequence in the original order.
    /**
     /// The proportions specified will be applied uniformly to each label being
     // partitioned on.

```

```

    /**
     * - Parameter proportions: An array of values on [0,1] specifying
     * the proportions in each partition. Automatically normalized to 1.
     * - Parameter sequenceIdentifierColumn: The sequence
     * identifier column in an MLDataTable to identify rows of a sequence.
     * - Parameter column: The column in an MLDataTable being stratified
     * on.
     * - Parameter seed: Seed for the random number generator used for
     * splitting. The default seed is the current epoch time in milliseconds.
     */
    /**
     * - Returns: A new MLDataTable with an additional partition column
     * with the index of the partition for each row.
     */
    @available(macOS 11.0, iOS 15.0, tvOS 16.0)
    public func stratifiedSplitBySequence(
        Double, String
        String, throws MLDataTable
        String
        Int
    ) MLDataTable

    @available(10.15, 15.0)
    @available(10.14, 15.0)
    @available(10.14, 15.0)
    extension MLDataTable

    /// Generates a visualization for the data in the table.
    public func show() Any MLStreamingVisualizable

    @available(macOS 10.14, iOS 15.0, tvOS 16.0)
    extension MLDataTable : CustomStringConvertible, CustomPlaygroundDisplayConvertible

    /// A description of the data table shown in a playground.
    public var playgroundDescription: Any { get }

    /// A text representation of the data table.
    public var description: String { get }

    extension MLDataTable

    /// Creates a new data table with an additional column that contains the
    /// combined values of the given columns.
    /**
     * This function performs the inverse of
     */
    ``MLDataTable/unpack(columnNamed:valueTypes:indexSubset:keySubset:)``.

    /**
     * - Parameters:
     *   - columnsNamed: The name of the columns to compact.

```

```


    /**
     * - to: The name of the new condensed column.
     */
    /**
     * - type: The collection type for the new column. Typically, the type is
     * a sequence or a dictionary.
     */
    /**
     * - filling: The value to fill in any missing values with.
     */
    /**
     * - Returns: A new data table.
     */

public func pack String String  

MLDataTable PackType MLDataTable MLDataValue


    /**
     * Creates a new data table with additional columns that contain the
     * unpacked collections in the given column.
     */
    /**
     * This function performs the inverse of
     * ``MLDataTable/pack(columnsNamed:to:type:filling:)``.
     */
    /**
     * - Parameters:
     * - columnNamed: The name of the column to unpack. The underlying
     * type of
     *   the column must be either ``MLDataValue/SequenceType`` or
     *   ``MLDataValue/DictionaryType``.
     */
    /**
     * - valueTypes: An array of the underlying types for the new, unpacked
     * columns. If `nil`, the method infers the underlying types in the
     * sequence or dictionary.
     */
    /**
     * > Note: If not `nil`, the method unpacks
     * `valueTypes.count` columns.
     */
    /**
     * - indexSubset: The subset of indices to unpack from a specified
     * sequence-typed column. If `nil`, the method unpacks all indices.
     */
    /**
     * - keySubset: The subset of keys to unpack from a specified
     * dictionary-typed column. If `nil`, the method unpacks all keys.
     */
    /**
     * - Returns: A new data table.
     */

public func unpack String  

MLDataValue ValueType nil Int nil  

String nil MLDataTable


    /**
     * The storage operations for combining multiple columns into one.
     */

public enum PackType Sendable



```

```
    /// A packing type that combines the values of several columns into a sequence type.
```

```
    case sequence
```

```
    /// A packing type that combines the values of several columns into a dictionary type.
```

```
    case dictionary
```

```
    /// Returns a Boolean value indicating whether two values are equal.
```

```
    ///
```

```
    /// Equality is the inverse of inequality. For any values `a` and `b`,  
    /// `a == b` implies that `a != b` is `false`.
```

```
    ///
```

```
    /// - Parameters:
```

```
    ///   - lhs: A value to compare.
```

```
    ///   - rhs: Another value to compare.
```

```
    public static func      MLDataTable PackType
```

```
MLDataTable PackType      Bool
```

```
the    /// Hashes the essential components of this value by feeding them into
```

```
    /// given hasher.
```

```
    ///
```

```
    /// Implement this method to conform to the `Hashable` protocol. The  
    /// components used for hashing must be the same as the components  
compared
```

```
    /// in your type's `==` operator implementation. Call  
`hasher.combine(_:)`
```

```
    /// with each of these components.
```

```
    ///
```

```
    /// - Important: In your implementation of `hash(into:)`,  
    ///   don't call `finalize()` on the `hasher` instance provided,  
    ///   or replace it with a different instance.  
    /// Doing so may become a compile-time error in the future.
```

```
    ///
```

```
    /// - Parameter hasher: The hasher to use when combining the  
components
```

```
    ///   of this instance.
```

```
    public func hash          inout Hasher
```

```
    /// The hash value.
```

```
    ///
```

```
    /// Hash values are not guaranteed to be equal across different  
executions of
```

```
    /// your program. Do not save hash values to use during a future  
execution.
```

```
    ///
```

```
    /// - Important: `hashValue` is deprecated as a `Hashable`  
requirement. To
```

```
    ///   conform to `Hashable`, implement the `hash(into:)`  
requirement instead.
```

```
    /// The compiler provides an implementation for `hashValue` for
you.  
    public var hashValue Int get
```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataTable JoinType Equatable
```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataTable JoinType Hashable
```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataTable ColumnNames Equatable
```

```
/// Returns a Boolean value indicating whether two values are equal.
///
/// Equality is the inverse of inequality. For any values `a` and `b`,
/// `a == b` implies that `a != b` is `false`.
///
/// - Parameters:
///   - lhs: A value to compare.
///   - rhs: Another value to compare.
public static func MLDataTable ColumnNames
MLDataTable ColumnNames Bool
```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataTable ColumnNames CustomStringConvertible
CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible
```

```
/// A textual representation of this instance.
///
/// Calling this property directly is discouraged. Instead, convert an
/// instance of any type to a string by using the `String(describing:)`
/// initializer. This initializer works with any type, and uses the custom
/// `description` property for types that conform to
/// `CustomStringConvertible`:
///
///     struct Point: CustomStringConvertible {
///         let x: Int, y: Int
///
///         var description: String {
///             return "(\(x), \(y))"
///         }
///     }
```

```

////
////    let p = Point(x: 21, y: 30)
////    let s = String(describing: p)
////    print(s)
////    // Prints "(21, 30)"
////
//// The conversion of `p` to a string in the assignment to `s` uses the
//// `Point` type's `description` property.
public var description String get

/// A textual representation of this instance, suitable for debugging.
///
/// Calling this property directly is discouraged. Instead, convert an
/// instance of any type to a string by using the `String(reflecting:)` initializer.
/// This initializer works with any type, and uses the custom
/// `debugDescription` property for types that conform to
/// `CustomDebugStringConvertible`:
///
/// struct Point: CustomDebugStringConvertible {
///     let x: Int, y: Int
///
///     var debugDescription: String {
///         return "(\(x), \(y))"
///     }
/// }
///
/// let p = Point(x: 21, y: 30)
/// let s = String(reflecting: p)
/// print(s)
/// // Prints "(21, 30)"
///
/// The conversion of `p` to a string in the assignment to `s` uses the
/// `Point` type's `debugDescription` property.
public var debugDescription String get

/// A custom playground description for this instance.
public var playgroundDescription Any get

```

```

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataTable Rows RandomAccessCollection

```

```

/// The Element of a DataTable is a Row. This is represented as a Dictionary-
like type
/// containing all Column names and their corresponding values.
public typealias Element MLDataTable Row

/// Subscript by index. This returns a row in the data table.
public subscript Int MLDataTable Rows Element

```

```
get
```

```
    /// The position of the first row in a nonempty DataTable. If the DataTable is  
empty, `startIndex`
```

```
    /// is equal to `endIndex`.
```

```
public var startIndex : Int { get
```

```
    /// The DataTable's "past the end" position---that is, the position one greater  
than the last valid
```

```
    /// subscript argument.
```

```
public var endIndex : Int { get
```

```
    /// A type that represents a position in the collection.
```

```
    ///
```

```
    /// Valid indices consist of the position of every element and a
```

```
    /// "past the end" position that's not valid for use as a subscript
```

```
    /// argument.
```

```
@available iOS 15.0 tvOS 16.0 macOS 10.14
```

```
public typealias Index = Int
```

```
    /// A type that represents the indices that are valid for subscripting the  
    /// collection, in ascending order.
```

```
@available iOS 15.0 tvOS 16.0 macOS 10.14
```

```
public typealias Indices = Range<Int>
```

```
    /// A type that provides the collection's iteration interface and  
    /// encapsulates its iteration state.
```

```
    ///
```

```
    /// By default, a collection conforms to the `Sequence` protocol by  
    /// supplying `IndexingIterator` as its associated `Iterator`  
    /// type.
```

```
@available iOS 15.0 tvOS 16.0 macOS 10.14
```

```
public typealias Iterator
```

```
IndexingIterator MLDataTable.Rows
```

```
    /// A collection representing a contiguous subrange of this collection's  
    /// elements. The subsequence shares indices with the original collection.
```

```
    ///
```

```
    /// The default subsequence type for collections that don't define their own  
    /// is `Slice`.
```

```
@available iOS 15.0 tvOS 16.0 macOS 10.14
```

```
public typealias SubSequence = Slice<MLDataTable.Rows>
```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0
```

```
extension MLDataTable.Rows : CustomStringConvertible
```

```
CustomDebugStringConvertible
```

```
CustomPlaygroundDisplayConvertible
```

```
/// A textual representation of this instance.  
///  
/// Calling this property directly is discouraged. Instead, convert an  
/// instance of any type to a string by using the `String(describing:)`  
/// initializer. This initializer works with any type, and uses the custom  
/// `description` property for types that conform to  
/// `CustomStringConvertible`:  
///  
///     struct Point: CustomStringConvertible {  
///         let x: Int, y: Int  
///  
///         var description: String {  
///             return "(\(x), \(y))"  
///         }  
///     }  
///  
///     let p = Point(x: 21, y: 30)  
///     let s = String(describing: p)  
///     print(s)  
///     // Prints "(21, 30)"  
///  
/// The conversion of `p` to a string in the assignment to `s` uses the  
/// `Point` type's `description` property.  
public var description String get  
  
/// A textual representation of this instance, suitable for debugging.  
///  
/// Calling this property directly is discouraged. Instead, convert an  
/// instance of any type to a string by using the `String(reflecting:)`  
/// initializer. This initializer works with any type, and uses the custom  
/// `debugDescription` property for types that conform to  
/// `CustomDebugStringConvertible`:  
///  
///     struct Point: CustomDebugStringConvertible {  
///         let x: Int, y: Int  
///  
///         var debugDescription: String {  
///             return "(\(x), \(y))"  
///         }  
///     }  
///  
///     let p = Point(x: 21, y: 30)  
///     let s = String(reflecting: p)  
///     print(s)  
///     // Prints "(21, 30)"  
///  
/// The conversion of `p` to a string in the assignment to `s` uses the  
/// `Point` type's `debugDescription` property.  
public var debugDescription String get
```

```

/// A custom playground description for this instance.
public var playgroundDescription Any get

@available(macOS 10.14, iOS 15.0, tvOS 16.0)
extension MLDataTable.Row : Equatable

    /// Returns a Boolean value indicating whether two values are equal.
    ///
    /// Equality is the inverse of inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is `false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to compare.
    public static func MLDataTable.Row
        MLDataTable.Row Bool

@available(macOS 10.14, iOS 15.0, tvOS 16.0)
extension MLDataTable.Row : Collection

    /// A type representing the sequence's elements.
    public typealias Element MLDataTable.Row.Key
    MLDataTable.Row.Value

    /// A type that represents a position in the collection.
    ///
    /// Valid indices consist of the position of every element and a
    /// "past the end" position that's not valid for use as a subscript
    /// argument.
    public typealias Index Int

    /// The number of elements in the collection.
    ///
    /// To check whether a collection is empty, use its `isEmpty` property
    /// instead of comparing `count` to zero. Unless the collection guarantees
    /// random-access performance, calculating `count` can be an O(*n*)
    /// operation.
    ///
    /// - Complexity: O(1) if the collection conforms to
    ///   `RandomAccessCollection`; otherwise, O(*n*), where *n* is the
    length
    /// of the collection.
    public var count Int get

    /// A Boolean value indicating whether the collection is empty.
    ///

```

```

    /// When you need to check whether your collection is empty, use the
    /// `isEmpty` property instead of checking that the `count` property is
    /// equal to zero. For collections that don't conform to
    /// `RandomAccessCollection`, accessing the `count` property
iterates
    /// through the elements of the collection.
    ///
    /// let horseName = "Silver"
    /// if horseName.isEmpty {
    ///     print("My horse has no name.")
    /// } else {
    ///     print("Hi ho, \(horseName)!")
    /// }
    // Prints "Hi ho, Silver!"
    ///
    /// - Complexity: O(1)
public var isEmpty Bool get

    /// The position of the first element in a nonempty collection.
    ///
    /// If the collection is empty, `startIndex` is equal to `endIndex`.
public var startIndex Int get

    /// The collection's "past the end" position---that is, the position one
    /// greater than the last valid subscript argument.
    ///
    /// When you need a range that includes the last element of a collection, use
    /// the half-open range operator (`..<`) with `endIndex`. The `..<` operator
creates a range that doesn't include the upper bound, so it's always
safe to use with `endIndex`. For example:
    ///
    /// let numbers = [10, 20, 30, 40, 50]
    /// if let index = numbers.firstIndex(of: 30) {
    ///     print(numbers[index ..< numbers.endIndex])
    /// }
    // Prints "[30, 40, 50]"
    ///
    /// If the collection is empty, `endIndex` is equal to `startIndex`.
public var endIndex Int get

    /// Returns the position immediately after the given index.
    ///
    /// The successor of an index must be well defined. For an index `i` into a
    /// collection `c`, calling `c.index(after: i)` returns the same index
every
    /// time.
    ///
    /// - Parameter i: A valid index of the collection. `i` must be less than
    /// `endIndex`.

```

```
    /// - Returns: The index value immediately after `i`.
public func index           Int      Int

public func index           MLDataTable Row Key
MLDataTable Row Index

    /// Accesses the element at the specified position.
    ///
    /// The following example accesses an element of an array through its
    /// subscript to print its value:
    ///
    ///     var streets = ["Adams", "Bryant", "Channing",
    "Douglas", "Evarts"]
    ///     print(streets[1])
    ///     // Prints "Bryant"
    ///
    /// You can subscript a collection with any valid index other than the
    /// collection's end index. The end index refers to the position one past
    /// the last element of a collection, so it doesn't correspond with an
    /// element.
    ///
    /// - Parameter position: The position of the element to access.
`position`
    ///     must be a valid index of the collection that is not equal to the
    ///     `endIndex` property.
    ///
    /// - Complexity: O(1)
public subscript          Int      MLDataTable Row Key
MLDataTable Row Value     get

    /// A type that represents the indices that are valid for subscripting the
    /// collection, in ascending order.
available iOS 15.0 tvOS 16.0 macOS 10.14
public typealias Indices DefaultIndices MLDataTable Row

    /// A type that provides the collection's iteration interface and
    /// encapsulates its iteration state.
    ///
    /// By default, a collection conforms to the `Sequence` protocol by
    /// supplying `IndexingIterator` as its associated `Iterator`-
    /// type.
available iOS 15.0 tvOS 16.0 macOS 10.14
public typealias Iterator
IndexingIterator MLDataTable Row

    /// A collection representing a contiguous subrange of this collection's
    /// elements. The subsequence shares indices with the original collection.
    ///
    /// The default subsequence type for collections that don't define their own
```

```
/// is `Slice`.  
@available iOS 15.0 tvOS 16.0 macOS 10.14  
public typealias SubSequence Slice MLDataTable Row  
  
@available macOS 10.14 iOS 15.0 tvOS 16.0  
extension MLDataTable Row : CustomStringConvertible,  
    CustomDebugStringConvertible,  
    CustomPlaygroundDisplayConvertible  
  
    /// A textual representation of this instance.  
    ///  
    /// Calling this property directly is discouraged. Instead, convert an  
    /// instance of any type to a string by using the `String(describing:)`  
    /// initializer. This initializer works with any type, and uses the custom  
    /// `description` property for types that conform to  
    /// `CustomStringConvertible`:  
    ///  
    ///     struct Point: CustomStringConvertible {  
    ///         let x: Int, y: Int  
    ///  
    ///         var description: String {  
    ///             return "(\(x), \(y))"  
    ///         }  
    ///     }  
    ///  
    ///     let p = Point(x: 21, y: 30)  
    ///     let s = String(describing: p)  
    ///     print(s)  
    ///     // Prints "(21, 30)"  
    ///  
    /// The conversion of `p` to a string in the assignment to `s` uses the  
    /// `Point` type's `description` property.  
public var description String get  
  
    /// A textual representation of this instance, suitable for debugging.  
    ///  
    /// Calling this property directly is discouraged. Instead, convert an  
    /// instance of any type to a string by using the `String(reflecting:)`  
    /// initializer. This initializer works with any type, and uses the custom  
    /// `debugDescription` property for types that conform to  
    /// `CustomDebugStringConvertible`:  
    ///  
    ///     struct Point: CustomDebugStringConvertible {  
    ///         let x: Int, y: Int  
    ///  
    ///         var debugDescription: String {  
    ///             return "(\(x), \(y))"  
    ///         }  
    ///     }
```

```

    /**
     */
    let p = Point(x: 21, y: 30)
    let s = String(describing: p)
    print(s)
    // Prints "(21, 30)"

    /**
     * The conversion of `p` to a string in the assignment to `s` uses the
     * `Point` type's `debugDescription` property.
    public var debugDescription String get

    /**
     * A custom playground description for this instance.
    public var playgroundDescription Any get

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataTable Row

public struct Values RandomAccessCollection

    /**
     * The position of the first element in a nonempty collection.
    /**
     * If the collection is empty, `startIndex` is equal to `endIndex`.
    public var startIndex Int get

    /**
     * The collection's "past the end" position---that is, the position one
     * greater than the last valid subscript argument.
    /**
     * When you need a range that includes the last element of a
collection, use
        /**
         * the half-open range operator (`..<`) with `endIndex`. The
`..<` operator
            /**
             * creates a range that doesn't include the upper bound, so it's always
             * safe to use with `endIndex`. For example:
            /**
            let numbers = [10, 20, 30, 40, 50]
            if let index = numbers.firstIndex(of: 30) {
                print(numbers[index ..< numbers.endIndex])
            }
            // Prints "[30, 40, 50]"
            /**
             * If the collection is empty, `endIndex` is equal to `startIndex`.
    public var endIndex Int get

    /**
     * Accesses the element at the specified position.
    /**
     * The following example accesses an element of an array through its
     * subscript to print its value:
    /**

```

```

    ///      var streets = ["Adams", "Bryant", "Channing",
"Douglas", "Evarts"]
    ///      print(streets[1])
    ///      // Prints "Bryant"
    ///
    /// You can subscript a collection with any valid index other than the
    /// collection's end index. The end index refers to the position one past
    /// the last element of a collection, so it doesn't correspond with an
    /// element.
    ///
    /// - Parameter position: The position of the element to access.
`position`
    ///      must be a valid index of the collection that is not equal to the
    ///      `endIndex` property.
    ///
    /// - Complexity: O(1)
public subscript Int MLDataTableValue get

    /// A type representing the sequence's elements.
@available iOS 15.0 tvOS 16.0 macOS 10.14
public typealias Element MLDataTableValue

    /// A type that represents a position in the collection.
    ///
    /// Valid indices consist of the position of every element and a
    /// "past the end" position that's not valid for use as a subscript
    /// argument.
@available iOS 15.0 tvOS 16.0 macOS 10.14
public typealias Index Int

    /// A type that represents the indices that are valid for subscripting the
    /// collection, in ascending order.
@available iOS 15.0 tvOS 16.0 macOS 10.14
public typealias Indices Range Int

    /// A type that provides the collection's iteration interface and
    /// encapsulates its iteration state.
    ///
    /// By default, a collection conforms to the `Sequence` protocol by
    /// supplying `IndexingIterator` as its associated `Iterator`
    /// type.
@available iOS 15.0 tvOS 16.0 macOS 10.14
public typealias Iterator
IndexingIterator MLDataTable Row Values

    /// A collection representing a contiguous subrange of this collection's
    /// elements. The subsequence shares indices with the original
collection.
    ///

```

```
    /// The default subsequence type for collections that don't define their
own
    /// is `Slice`.
@available ios 15.0 tvOS 16.0 macOS 10.14
public typealias SubSequence
Slice MLDataTable Row Values
```

```
@available macOS 11.0 iOS 15.0 tvOS 16.0
extension MLDataTable PackType Equatable
```

```
@available macOS 11.0 iOS 15.0 tvOS 16.0
extension MLDataTable PackType Hashable
```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataTable Row Values Equatable
```

```
    /// Returns a Boolean value indicating whether two values are equal.
    ///
    /// Equality is the inverse of inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is `false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to compare.
public static func MLDataTable Row Values
MLDataTable Row Values Bool
```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataTable Row Values CustomStringConvertible
CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible
```

```
    /// A textual representation of this instance.
    ///
    /// Calling this property directly is discouraged. Instead, convert an
    /// instance of any type to a string by using the `String(describing:)`
    /// initializer. This initializer works with any type, and uses the custom
    /// `description` property for types that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point: CustomStringConvertible {
    ///         let x: Int, y: Int
    ///         var description: String {
```

```

    /**
     *      }
     */
    let p = Point(x: 21, y: 30)
    let s = String(describing: p)
    print(s)
    // Prints "(21, 30)"
    /**
     * The conversion of `p` to a string in the assignment to `s` uses the
     * `Point` type's `description` property.
public var description String get

    /**
     * A textual representation of this instance, suitable for debugging.
    /**
     * Calling this property directly is discouraged. Instead, convert an
     * instance of any type to a string by using the `String(reflecting:)` initializer.
     * This initializer works with any type, and uses the custom
     * `debugDescription` property for types that conform to
     * `CustomDebugStringConvertible`:
    /**
     * struct Point: CustomDebugStringConvertible {
     *     let x: Int, y: Int
     *
     *     var debugDescription: String {
     *         return "(\(x), \(y))"
     *     }
     * }
    /**
     * let p = Point(x: 21, y: 30)
     * let s = String(reflecting: p)
     * print(s)
     * // Prints "(21, 30)"
    /**
     * The conversion of `p` to a string in the assignment to `s` uses the
     * `Point` type's `debugDescription` property.
public var debugDescription String get

    /**
     * A custom playground description for this instance.
public var playgroundDescription Any get

    /**
     * The value of a cell in a data table.
    /**
     * The ``MLDataValue`` enumeration is the fundamental type that you use to
     * store training data in a table. Classifiers use data values to store
     * information like evaluation metrics. Data values wrap all of the possible
     * data types you can use with Create ML.
    /**

```

```
/// To access the underlying information in a data value, you can use the
/// properties that correspond to the type's enumeration cases. If you aren't
/// sure which kind of value a data value wrapper contains, use a switch
/// statement to unwrap the value, or check the value of the
/// ``MLDataValue/type`` property.
@available(macOS 10.14, iOS 15.0, tvOS 16.0)
public enum MLDataValue

    /// An integer value.
    case int Int

    /// A double value.
    case double Double

    /// A string value.
    case string String

    /// A sequence of data values.
    case sequence MLDataValue SequenceType

    /// A dictionary of named data values.
    case dictionary MLDataValue DictionaryType

    /// A multidimensional array of data values.
    case multiArray MLDataValue MultiArrayType

    /// An invalid value.
    case invalid

    /// The kind of the underlying value that the data value wraps.
    public var type MLDataValue ValueType get

    /// A Boolean value indicating whether the data value is valid.
    public var isValid Bool get

    /// The underlying integer value.
    public var intValue Int get

    /// The underlying double value.
    public var doubleValue Double get

    /// The underlying string value.
    public var stringValue String get

    /// The underlying sequence.
    public var sequenceValue MLDataValue SequenceType
get

    /// The underlying dictionary.
```

```
    public var dictionaryValue MLDataValue DictionaryType
get

    /// The underlying multidimensional array.
    public var multiArrayValue MLDataValue MultiArrayType
get

    /// A sequence of data values.
    public struct SequenceType

    /// A dictionary of named data values.
    public struct DictionaryType

    /// A multidimensional array of data values.
    public struct MultiArrayType

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataValue

    /// An enumeration describing the supported underlying types that an
    /// `MLDataValue` wraps.
    public enum ValueType

        /// An integer type.
        case int

        /// A double type.
        case double

        /// A string type.
        case string

        /// A sequence type.
        case sequence

        /// A dictionary type.
        case dictionary

        /// A multidimensional type.
        case multiArray

        /// An invalid type.
        case invalid

    /// Returns a Boolean value indicating whether two values are equal.
```

```

    /**
     * Equality is the inverse of inequality. For any values `a` and `b`,
     * `a == b` implies that `a != b` is `false`.
     */
    /**
     * - Parameters:
     *   - `lhs`: A value to compare.
     *   - `rhs`: Another value to compare.
     */
    public static func hash(MLDataValue.ValueType,
                           MLDataValue.ValueType) Bool
    {
        /**
         * Hashes the essential components of this value by feeding them into
         * the given hasher.
         */
        /**
         * Implement this method to conform to the `Hashable` protocol. The
         * components used for hashing must be the same as the components
         * compared
         *   - in your type's `==` operator implementation. Call
         *     `hasher.combine(_:)`
         *   - with each of these components.
         */
        /**
         * - Important: In your implementation of `hash(into:)`, don't call
         *   `finalize()` on the `hasher` instance provided, or replace it with a
         *   different instance. Doing so may become a compile-time error in the
         *   future.
         */
        /**
         * - Parameter hasher: The hasher to use when combining the
         * components
         *   - of this instance.
         */
        public func hash(inout Hasher)
        {
            /**
             * The hash value.
             */
            /**
             * Hash values are not guaranteed to be equal across different
             * executions of
             *   - your program. Do not save hash values to use during a future
             *     execution.
             */
            /**
             * - Important: `hashValue` is deprecated as a `Hashable` requirement. To
             *   conform to `Hashable`, implement the `hash(into:)` requirement instead.
             */
            /**
             * The compiler provides an implementation for `hashValue` for
             * you.
             */
            public var hashValue: Int { get }
        }
    }

```

**@available macOS 10.14 iOS 15.0 tvOS 16.0**  
**extension** MLDataValue **CustomStringConvertible**

## CustomDebugStringConvertible

```
/// A text representation of the data value.
public var description String get

/// A text representation of the data value that's suitable for output
/// during debugging.
public var debugDescription String get

@available(macOS 10.14, iOS 15.0, tvOS 16.0)
extension MLDataValue : Hashable

    /// Returns a Boolean value indicating whether two data values wrap the
    same
    /// underlying value.
    ///
    /// - Parameters:
    ///   - left: A data value to compare.
    ///
    ///   - right: A data value to compare.
    public static func == (MLDataValue, MLDataValue) Bool

    /// Hashes the essential components of this value by feeding them into the
    /// given hasher.
    ///
    /// Implement this method to conform to the `Hashable` protocol. The
    /// components used for hashing must be the same as the components
    compared
    /// in your type's `==` operator implementation. Call
    `hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your implementation of `hash(into:)`,
    /// don't call `finalize()` on the `hasher` instance provided,
    /// or replace it with a different instance.
    /// Doing so may become a compile-time error in the future.
    ///
    /// - Parameter hasher: The hasher to use when combining the
    components
    /// of this instance.
    public func hash(inout Hasher)

    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different executions of
    /// your program. Do not save hash values to use during a future execution.
    ///
    /// - Important: `hashValue` is deprecated as a `Hashable`'
```

requirement. To

    /// conform to `Hashable`, implement the `hash(into:)` requirement instead.

    /// The compiler provides an implementation for `hashValue` for you.  
    **public var** hashValue **Int** **get**

**@available** macOS 10.14 iOS 15.0 tvOS 16.0  
**extension** MLDataValue SequenceType

    /// Creates a new default instance of the conforming type when a data value  
    /// is missing or invalid.

**public init**

**public init** **S** \_ **S** **where** **S** **Sequence**  
        **S** **Element** **MLDataValue**

**public init** **S** \_ **S** **where** **S** **Sequence**  
        **S** **Element** **MLDataValueConvertible**

**@available** macOS 10.14 iOS 15.0 tvOS 16.0  
**extension** MLDataValue SequenceType RandomAccessCollection

    /// A type representing the sequence's elements.  
    **public typealias** Element **MLDataValue**

    /// A type that represents a position in the collection.

    ///

    /// Valid indices consist of the position of every element and a  
    /// "past the end" position that's not valid for use as a subscript  
    /// argument.

**public typealias** Index **Int**

    /// The position of the first element in a nonempty collection.

    ///

    /// If the collection is empty, `startIndex` is equal to `endIndex`.

**public var** startIndex **MLDataValue** SequenceType Index  
    **get**

    /// The collection's "past the end" position---that is, the position one  
    /// greater than the last valid subscript argument.

    ///

    /// When you need a range that includes the last element of a collection, use  
    /// the half-open range operator (`..<`) with `endIndex`. The `..<`  
operator

    /// creates a range that doesn't include the upper bound, so it's always  
    /// safe to use with `endIndex`. For example:

    ///

    /// let numbers = [10, 20, 30, 40, 50]

```

    /**
     *      if let index = numbers.firstIndex(of: 30) {
     *          print(numbers[index ..< numbers.endIndex])
     *      }
     *      // Prints "[30, 40, 50]"
     *
     *      // If the collection is empty, `endIndex` is equal to `startIndex`.
public var endIndex MLDataValue SequenceType Index
get

    /// Accesses the element at the specified position.
    ///
    /// The following example accesses an element of an array through its
    /// subscript to print its value:
    ///
    ///     var streets = ["Adams", "Bryant", "Channing",
    "Douglas", "Evarts"]
    ///     print(streets[1])
    ///     // Prints "Bryant"
    ///
    /// You can subscript a collection with any valid index other than the
    /// collection's end index. The end index refers to the position one past
    /// the last element of a collection, so it doesn't correspond with an
    /// element.
    ///
    /// - Parameter position: The position of the element to access.
`position`
    /// must be a valid index of the collection that is not equal to the
    /// `endIndex` property.
    ///
    /// - Complexity: O(1)
public subscript MLDataValue SequenceType Index
MLDataValue SequenceType Element get

    /// A type that represents the indices that are valid for subscripting the
    /// collection, in ascending order.
available iOS 15.0 tvOS 16.0 macOS 10.14
public typealias Indices
Range MLDataValue SequenceType Index

    /// A type that provides the collection's iteration interface and
    /// encapsulates its iteration state.
    ///
    /// By default, a collection conforms to the `Sequence` protocol by
    /// supplying `IndexingIterator` as its associated `Iterator` type.
    ///
available iOS 15.0 tvOS 16.0 macOS 10.14
public typealias Iterator
IndexingIterator MLDataValue SequenceType

```

```
/// A collection representing a contiguous subrange of this collection's
/// elements. The subsequence shares indices with the original collection.
///
/// The default subsequence type for collections that don't define their own
/// is `Slice`.
@available iOS 15.0 tvOS 16.0 macOS 10.14
public typealias SubSequence
Slice MLDataValue SequenceType

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataValue SequenceType ExpressibleByArrayLiteral

    /// Creates an instance initialized with the given elements.
    public init
    MLDataValue SequenceType Element

    /// The type of the elements of an array literal.
    @available iOS 15.0 tvOS 16.0 macOS 10.14
    public typealias ArrayLiteralElement
    MLDataValue SequenceType Element

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataValue SequenceType Equatable

    /// Returns a Boolean value indicating whether two values are equal.
    ///
    /// Equality is the inverse of inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is `false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to compare.
    public static func == (MLDataValue SequenceType
    MLDataValue SequenceType) Bool

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataValue SequenceType CustomStringConvertible
CustomDebugStringConvertible

    /// A textual representation of this instance.
    ///
    /// Calling this property directly is discouraged. Instead, convert an
    /// instance of any type to a string by using the `String(describing:)`
    /// initializer. This initializer works with any type, and uses the custom
    /// `description` property for types that conform to
```

```

/// `CustomStringConvertible`:
///
///     struct Point: CustomStringConvertible {
///         let x: Int, y: Int
///
///         var description: String {
///             return "(\(x), \(y))"
///         }
///     }
///
///     let p = Point(x: 21, y: 30)
///     let s = String(describing: p)
///     print(s)
///     // Prints "(21, 30)"
///
/// The conversion of `p` to a string in the assignment to `s` uses the
/// `Point` type's `description` property.
public var description String get

/// A textual representation of this instance, suitable for debugging.
///
/// Calling this property directly is discouraged. Instead, convert an
/// instance of any type to a string by using the `String(reflecting:)` initializer.
/// This initializer works with any type, and uses the custom
/// `debugDescription` property for types that conform to
/// `CustomDebugStringConvertible`:
///
///     struct Point: CustomDebugStringConvertible {
///         let x: Int, y: Int
///
///         var debugDescription: String {
///             return "(\(x), \(y))"
///         }
///     }
///
///     let p = Point(x: 21, y: 30)
///     let s = String(reflecting: p)
///     print(s)
///     // Prints "(21, 30)"
///
/// The conversion of `p` to a string in the assignment to `s` uses the
/// `Point` type's `debugDescription` property.
public var debugDescription String get

```

**@available macOS 10.14 iOS 15.0 tvOS 16.0**  
**extension** MLDataValue SequenceType **MLDataValueConvertible**

/// The underlying type a machine learning sequence uses when it wraps

```
    /// itself in a data value.  
    ///  
    /// A machine learning sequence is an  
``MLDataValue/ValueType/sequence``.  
    public static var dataValueType MLDataValue ValueType  
get  
  
    /// Creates a data-value sequence from another sequence.  
    ///  
    /// Use this initializer to create an ``MLDataValue/SequenceType`` from  
    /// another data-value sequence instance. You can confirm the data value's  
    /// underlying type by retrieving a non-`nil` value from  
    /// ``MLDataValue/sequenceValue`` or by inspecting the  
``MLDataValue/type``  
    /// property.  
    public init MLDataValue  
  
    /// The sequence wrapped in a data value.  
    public var dataValue MLDataValue get  
  
    /// Represents a dictionary of MLDataValues keyed by other MLDataValues.  
@available macOS 10.14 iOS 15.0 tvOS 16.0  
extension MLDataValue DictionaryType  
  
    public typealias Key MLDataValue  
  
    public typealias Value MLDataValue  
  
    /// Creates a new default instance of the conforming type when a data value  
    /// is missing or invalid.  
    public init  
  
    public init _ MLDataValue MLDataValue  
  
    public init S S  
where S Sequence S Element MLDataValue MLDataValue  
  
    /// The number of elements in the collection.  
    ///  
    /// To check whether a collection is empty, use its `isEmpty` property  
    /// instead of comparing `count` to zero. Unless the collection guarantees  
    /// random-access performance, calculating `count` can be an O(*n*)  
    /// operation.  
    ///  
    /// - Complexity: O(1) if the collection conforms to  
    /// `RandomAccessCollection`; otherwise, O(*n*), where *n* is the  
length  
    /// of the collection.
```

```

public var count Int get

    /// A Boolean value indicating whether the collection is empty.
    ///
    /// When you need to check whether your collection is empty, use the
    /// `isEmpty` property instead of checking that the `count` property is
    /// equal to zero. For collections that don't conform to
    /// `RandomAccessCollection`, accessing the `count` property
iterates
    /// through the elements of the collection.
    ///
    /// let horseName = "Silver"
    /// if horseName.isEmpty {
    ///     print("My horse has no name.")
    /// } else {
    ///     print("Hi ho, \u{horseName}!")
    /// }
    /// // Prints "Hi ho, Silver!"
    ///
    /// - Complexity: O(1)
public var isEmpty Bool get

    public subscript MLDataValue DictionaryType Key
    MLDataValue DictionaryType Value get

```

**@available macOS 10.14 iOS 15.0 tvOS 16.0**  
**extension** MLDataValue DictionaryType Collection

```

    /// A type that represents a position in the collection.
    ///
    /// Valid indices consist of the position of every element and a
    /// "past the end" position that's not valid for use as a subscript
    /// argument.
public struct Index Comparable

    /// Returns a Boolean value indicating whether two values are equal.
    ///
    /// Equality is the inverse of inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is `false`.
    ///
    /// - Parameters:
    ///     - lhs: A value to compare.
    ///     - rhs: Another value to compare.
public static func
    MLDataValue DictionaryType Index
    MLDataValue DictionaryType Index Bool

    /// Returns a Boolean value indicating whether the value of the first

```

```

    /// argument is less than that of the second argument.
    ///
    /// This function is the only requirement of the `Comparable` protocol. The
    /// remainder of the relational operator functions are implemented by
    /// the standard library for any type that conforms to `Comparable`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to compare.
public static func
MLDataValue DictionaryType Index
MLDataValue DictionaryType Index      Bool

    /// A type representing the sequence's elements.
public typealias Element      MLDataValue
MLDataValue

    /// The position of the first element in a nonempty collection.
    ///
    /// If the collection is empty, `startIndex` is equal to `endIndex`.
public var startIndex  MLDataValue DictionaryType Index
get

    /// The collection's "past the end" position---that is, the position one
    /// greater than the last valid subscript argument.
    ///
    /// When you need a range that includes the last element of a collection, use
    /// the half-open range operator (`..<`) with `endIndex`. The `..<` operator
    /// creates a range that doesn't include the upper bound, so it's always
    /// safe to use with `endIndex`. For example:
    ///
    ///     let numbers = [10, 20, 30, 40, 50]
    ///     if let index = numbers.firstIndex(of: 30) {
    ///         print(numbers[index ..< numbers.endIndex])
    ///     }
    ///     // Prints "[30, 40, 50]"
    ///
    /// If the collection is empty, `endIndex` is equal to `startIndex`.
public var endIndex  MLDataValue DictionaryType Index
get

    /// Accesses the element at the specified position.
    ///
    /// The following example accesses an element of an array through its
    /// subscript to print its value:
    ///

```

```

    ///      var streets = ["Adams", "Bryant", "Channing",
"Douglas", "Evarts"]
    ///      print(streets[1])
    ///      // Prints "Bryant"
    ///
    /// You can subscript a collection with any valid index other than the
    /// collection's end index. The end index refers to the position one past
    /// the last element of a collection, so it doesn't correspond with an
    /// element.
    ///
    /// - Parameter position: The position of the element to access.
`position`
    ///      must be a valid index of the collection that is not equal to the
    ///      `endIndex` property.
    ///
    /// - Complexity: O(1)
public subscript MLDataValue DictionaryType Index
MLDataValue DictionaryType Element get

    /// Returns the position immediately after the given index.
    ///
    /// The successor of an index must be well defined. For an index `i` into a
    /// collection `c`, calling `c.index(after: i)` returns the same index
every
    /// time.
    ///
    /// - Parameter i: A valid index of the collection. `i` must be less than
    ///      `endIndex`.
    /// - Returns: The index value immediately after `i`.
public func index
MLDataValue DictionaryType Index
MLDataValue DictionaryType Index

    /// A type that represents the indices that are valid for subscripting the
    /// collection, in ascending order.
available iOS 15.0 tvOS 16.0 macOS 10.14
public typealias Indices
DefaultIndices MLDataValue DictionaryType

    /// A type that provides the collection's iteration interface and
    /// encapsulates its iteration state.
    ///
    /// By default, a collection conforms to the `Sequence` protocol by
    /// supplying `IndexingIterator` as its associated `Iterator`
    /// type.
available iOS 15.0 tvOS 16.0 macOS 10.14
public typealias Iterator
IndexingIterator MLDataValue DictionaryType

```

```

    /// A collection representing a contiguous subrange of this collection's
    /// elements. The subsequence shares indices with the original collection.
    ///
    /// The default subsequence type for collections that don't define their own
    /// is `Slice`.
    @available iOS 15.0 tvOS 16.0 macOS 10.14
    public typealias SubSequence
    Slice MLKeyValue DictionaryType

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLKeyValue DictionaryType
CustomStringConvertible CustomDebugStringConvertible

    /// A textual representation of this instance.
    ///
    /// Calling this property directly is discouraged. Instead, convert an
    /// instance of any type to a string by using the `String(describing:)` initializer.
    /// This initializer works with any type, and uses the custom
    /// `description` property for types that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point: CustomStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var description: String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(describing: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string in the assignment to `s` uses the
    /// `Point` type's `description` property.
public var description String get

    /// A textual representation of this instance, suitable for debugging.
    ///
    /// Calling this property directly is discouraged. Instead, convert an
    /// instance of any type to a string by using the `String(reflecting:)` initializer.
    /// This initializer works with any type, and uses the custom
    /// `debugDescription` property for types that conform to
    /// `CustomDebugStringConvertible`:
    ///
    ///     struct Point: CustomDebugStringConvertible {
    ///         let x: Int, y: Int

```

```
///  
///     var debugDescription: String {  
///         return "( \(x), \(y))"  
///     }  
/// }  
  
/// let p = Point(x: 21, y: 30)  
/// let s = String(reflecting: p)  
/// print(s)  
/// // Prints "(21, 30)"  
///  
/// The conversion of `p` to a string in the assignment to `s` uses the  
/// `Point` type's `debugDescription` property.  
public var debugDescription String get
```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0  
extension MLDataValue DictionaryType Equatable  
  
/// Returns a Boolean value indicating whether two values are equal.  
///  
/// Equality is the inverse of inequality. For any values `a` and `b`,  
/// `a == b` implies that `a != b` is `false`.  
///  
/// - Parameters:  
/// - lhs: A value to compare.  
/// - rhs: Another value to compare.  
public static func MLDataValue DictionaryType  
MLDataValue DictionaryType Bool
```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0  
extension MLDataValue DictionaryType MLDataValueConvertible
```

```
/// The underlying type a machine learning dictionary uses when it wraps  
/// itself in a data value.  
///  
/// A machine learning dictionary is an  
/// ``MLDataValue/ValueType/dictionary``.  
public static var dataValueType MLDataValue ValueType  
get  
  
/// Creates a data-value dictionary from another dictionary.  
///  
/// Use this initializer to create an ``MLDataValue/DictionaryType``  
from  
/// another data-value dictionary instance. You can confirm the data value's  
/// underlying type by retrieving a non-`nil` value from
```

```

    /// ``MLDataValue/dictionaryValue`` or by inspecting the
    /// ``MLDataValue/type`` property.
public init MLDataValue

    /// The dictionary wrapped in a data value.
public var dataValue MLDataValue get

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataValue MultiArrayType Equatable

public init Int

public init MLMultiArray

public subscript Int Double get

public subscript Int Double get

public var mlMultiArray MLMultiArray get

    /// Returns a Boolean value indicating whether two values are equal.
    ///
    /// Equality is the inverse of inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is `false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to compare.
public static func MLDataValue MultiArrayType
    MLDataValue MultiArrayType Bool

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataValue MultiArrayType
CustomStringConvertible CustomDebugStringConvertible

    /// A textual representation of this instance.
    ///
    /// Calling this property directly is discouraged. Instead, convert an
    /// instance of any type to a string by using the `String(describing:)`
    /// initializer. This initializer works with any type, and uses the custom
    /// `description` property for types that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point: CustomStringConvertible {
    ///         let x: Int, y: Int
    ///         var description: String {

```

```

    /**
     *      }
     */
    let p = Point(x: 21, y: 30)
    let s = String(describing: p)
    print(s)
    // Prints "(21, 30)"
    /**
     * The conversion of `p` to a string in the assignment to `s` uses the
     * `Point` type's `description` property.
public var description String get

    /**
     * A textual representation of this instance, suitable for debugging.
    /**
     * Calling this property directly is discouraged. Instead, convert an
     * instance of any type to a string by using the `String(reflecting:)` initializer.
     * This initializer works with any type, and uses the custom
     * `debugDescription` property for types that conform to
     * `CustomDebugStringConvertible`:
    /**
     struct Point: CustomDebugStringConvertible {
    let x: Int, y: Int

    var debugDescription: String {
        return "(\(x), \(y))"
    }
}

let p = Point(x: 21, y: 30)
let s = String(reflecting: p)
print(s)
// Prints "(21, 30)"

    * The conversion of `p` to a string in the assignment to `s` uses the
     * `Point` type's `debugDescription` property.
public var debugDescription String get

```

**@available macOS 10.14 iOS 15.0 tvOS 16.0**  
**extension** **MLDataValue** **MultiArrayType** **MLDataValueConvertible**

```

    /**
     * The underlying type a machine learning multidimensional array uses
when
    /**
     * it wraps itself in a data value.
    /**
     * A machine learning multidimensional array is an
     * ``MLDataValue/ValueType/multiArray``.

```

```
public static var dataValueType MLDataValue.ValueType
get

    /// Creates a data-value multidimensional array from another instance.
    ///
    /// Use this initializer to create an ``MLDataValue/MultiArrayType`` from
    /// another multiarray instance. You can confirm the data value's underlying
    /// type by retrieving a non-`nil` value from
    /// ``MLDataValue/multiArrayValue`` or by inspecting the
    /// ``MLDataValue/type`` property.
public init MLDataValue

    /// Creates a new default instance of the conforming type when a data value
    /// is missing or invalid.
public init

    /// The multidimensional array wrapped in a data value.
public var dataValue MLDataValue get

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataValue.ValueType : CustomStringConvertible
CustomDebugStringConvertible

    /// A text representation of the data value type.
public var description String get

    /// A text representation of the data value type that's suitable for output
    /// during debugging.
public var debugDescription String get

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataValue.ValueType : Equatable

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDataValue.ValueType : Hashable

    /// A type that can convert itself to and from a data value.
    ///
    /// You can use any type that conforms to the ``MLDataValueConvertible`` protocol
    /// in an ``MLDataValue`` or an ``MLDataTable``. For example, you can
    /// create a data table by using its ``MLDataTable/init(dictionary:)`` initializer
    /// with a ``[<doc://com.apple.documentation/documentation/swift/string>]``:
    /// ``MLDataValueConvertible`` dictionary.
```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0
public protocol MLDataValueConvertible

    /// The underlying type the conforming type uses when it wraps itself in a
    /// data value.
    ///
    /// See ``MLDataValue/ValueType`` for a list of available options.
static var dataType MLDataValue ValueType get

    /// Creates an instance of the conforming type from a data value.
init MLDataValue

    /// Creates a new default instance of the conforming type when a data value
    /// is missing or invalid.
init

    /// The value of the conforming type's instance wrapped in a data value.
var dataValue MLDataValue get

    /// A classifier that predicts the target by creating rules to split the data.
@available macOS 10.14 iOS 15.0 tvOS 16.0
public struct MLDecisionTreeClassifier @unchecked Sendable

    /// The Core ML model.
public var model MLModel

    /// The name of the column you selected at initialization to define which
    categories the classifier predicts.
    ///
    /// Changing the value of this property doesn't retrain the model or affect its
    behavior.
public var targetColumn String

    /// The names of the columns you selected at initialization to train the
    classifier.
    ///
    /// Changing the value of this property doesn't retrain the model or affect its
    behavior.
public var featureColumns String

    /// The underlying parameters used when training the model.
public let modelParameters
MLDecisionTreeClassifier ModelParameters

    /// Measurements of the classifier's performance on the training data set.
public var trainingMetrics MLClassifierMetrics get

    /// Measurements of the classifier's performance on the validation data set.
public var validationMetrics MLClassifierMetrics get
```

```

/// Creates a decision tree classifier.
///
/// - Parameters:
///   - trainingData: The training data
///   - targetColumn: Name of the column containing the class labels
///   - featureColumns: Names of the columns containing feature
values. If `nil` all columns, other than the target
///   column, will be used as feature values.
///   - parameters: Model training parameters
@available macOS 12.0  iOS 15.0  tvOS 16.0
public init           DataFrame          String
                      String      nil
MLDecisionTreeClassifier ModelParameters

throws

/// Creates a Decision Tree Classifier from the feature columns in the training
data to predict the categories in
/// the target column.
///
/// - Parameters:
///   - trainingData: A data table of training examples.
///   - targetColumn: The column name for the values in the training
data that the classifier should predict.
///   - featureColumns: The column names for the values in the
training data that the classifier uses to predict
///   the target value.
///   - parameters: The model parameters.
@available                         10.14          13.0
"Use DataFrame instead of MLDataTable when
initializing."
@available                         15.0          16.0
"Use DataFrame instead of MLDataTable when
initializing."
@available
public init           MLDataTable
String                  String      nil
MLDecisionTreeClassifier ModelParameters
nil    throws

/// Creates a decision tree classifier classifier from a checkpoint.
///
/// - Parameter checkpoint: Training checkpoint.
/// - Throws: `MLCreateError` if the checkpoint can't be loaded.
@available macOS 12.0  iOS 15.0  tvOS 16.0
public init           MLCheckpoint  throws

/// Trains a decision tree classifier.
///

```

```

    /// If `sessionDirectory` is provided it will save training progress. If
    there is progress already saved training
    /// will resume from the last checkpoint.
    ///
    /// - Parameters:
    ///   - trainingData: A DataTable specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
    trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
    columns to be
    ///                               used to predict the target, if not provided,
    default to use all the
    ///                               other columns in the trainingData, except the
    one specified by targetColumn
    ///   - parameters: Model training parameters. See
    `MLDecisionTreeClassifier.ModelParameters` for the defaults.
    ///   - sessionParameters: Training session parameters. See
    `MLTrainingSessionParameters` for the defaults.
    ///
    /// - Returns: A `MLJob` that can be used to observe training progress.
@available           12.0          14.0
@available           15.0          17.0
@available           16.0          17.0
public static func train           MLDataTable
                    String           String      nil
                    MLDecisionTreeClassifier ModelParameters

MLTrainingSessionParameters
throws MLJob MLDecisionTreeClassifier

```

```

    /// Trains a decision tree classifier.
    ///
    /// If `sessionDirectory` is provided it will save training progress. If
    there is progress already saved training
    /// will resume from the last checkpoint.
    ///
    /// - Parameters:
    ///   - trainingData: A `DataFrame` specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
    trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
    columns to be
    ///                               used to predict the target, if not provided,
    default to use all the
    ///                               other columns in the trainingData, except the
    one specified by targetColumn
    ///   - parameters: Model training parameters. See
    `MLDecisionTreeClassifier.ModelParameters` for the defaults.
    ///   - sessionParameters: Training session parameters. See
    `MLTrainingSessionParameters` for the defaults.
    ///

```

```
    /// - Returns: A `MLJob` that can be used to observe training progress.
    @available macos 12.0  iOS 15.0  tvOS 16.0
    public static func train                DataFrame
                                String      String    nil
                                MLDecisionTreeClassifier ModelParameters

    MLTrainingSessionParameters
    throws    MLJob MLDecisionTreeClassifier

    /// Creates or restores a training session.
    ///
    /// - Parameters:
    ///   - trainingData: A DataTable specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
    trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
    columns to be
    ///                   used to predict the target, if not provided, default to use
    all the
    ///                   other columns in the trainingData, except the one
    specified by targetColumn
    ///   - parameters: Model training parameters. See
    `MLDecisionTreeClassifier.ModelParameters` for the defaults.
    ///   - sessionParameters: Training session parameters. See
    `MLTrainingSessionParameters` for the defaults.
    ///
    /// - Returns: A `MLTrainingSession` that can be used to start or
    resume training.
    @available             12.0          14.0
    @available             15.0          17.0
    @available             16.0          17.0
    public static func makeTrainingSession
    MLDataTable           String         String
    nil                  MLDecisionTreeClassifier ModelParameters
                                MLTrainingSessionParameters
    throws
    MLTrainingSession MLDecisionTreeClassifier

    /// Creates or restores a training session.
    ///
    /// - Parameters:
    ///   - trainingData: A `DataFrame` specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
    trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
    columns to be
    ///                   used to predict the target, if not provided, default to use
    all the
    ///                   other columns in the trainingData, except the one
    specified by targetColumn
    ///   - parameters: Model training parameters. See
```

```
`MLDecisionTreeClassifier.ModelParameters` for the defaults.  
    /// - sessionParameters: Training session parameters. See  
`MLTrainingSessionParameters` for the defaults.  
    ///  
    /// - Returns: A `MLTrainingSession` that can be used to start or  
resume training.  
    @available macOS 12.0 iOS 15.0 tvOS 16.0  
    public static func makeTrainingSession  
DataFrame String String  
nil MLDecisionTreeClassifier ModelParameters  
MLTrainingSessionParameters  
throws  
MLTrainingSession MLDecisionTreeClassifier  
  
    /// Restores an existing training session.  
    ///  
    /// - Parameters:  
    /// - sessionParameters: Training session parameters. The  
`sessionDirectory` parameter is required.  
    ///  
    /// - Returns: A `MLTrainingSession` that can be used to resume  
training.  
    @available macOS 12.0 iOS 15.0 tvOS 16.0  
    public static func  
restoreTrainingSession  
MLTrainingSessionParameters throws  
MLTrainingSession MLDecisionTreeClassifier  
  
    /// Resumes a training session from the last checkpoint if available.  
    ///  
    /// If there are no resumable checkpoints training starts over from the  
beginning.  
    ///  
    /// - Parameter session: Loaded or new training session.  
    ///  
    /// - Returns: A `MLJob` that can be used to observe training progress.  
    @available macOS 12.0 iOS 15.0 tvOS 16.0  
    public static func resume  
MLTrainingSession MLDecisionTreeClassifier throws  
MLJob MLDecisionTreeClassifier  
  
    /// Predicts a column of labels for the given testing data.  
    @available macOS 12.0 iOS 15.0 tvOS 16.0  
    public func predictions DataFrame throws  
AnyColumn  
  
    /// Classifies the provided data into the target categories.  
    ///  
    /// - Parameters:  
    /// - data: The data you want the model to classify.
```

```
///  
/// – Returns: A column of labels predicted by the classifier.  
@available 10.14 13.0  
    "Use DataFrame instead of MLDataTable."  
@available 15.0 16.0  
    "Use DataFrame instead of MLDataTable."  
@available  
public func predictions MLDataTable throws  
MLUntypedColumn  
  
/// Evaluates the classifier on the provided labeled data.  
///  
/// Evaluation should be done on a testing data set that the model has not  
seen as part of the training or  
/// validation data sets. The data should have feature columns with identical  
name and type to the  
/// training data, as well as a labels column with the same name.  
///  
/// – Parameters:  
/// – labeledData: A `DataFrame` to evaluate the trained model on.  
///  
/// – Returns: Metrics that describe the classification errors  
/// (` `MLClassifierMetrics/classificationError` `), the precision  
and recall  
/// percentages (` `MLClassifierMetrics/precisionRecall` `), and  
a table that  
/// describes how labels were misapplied  
(` `MLClassifierMetrics/confusion` `)  
/// on the provided data.  
@available macOS 12.0 iOS 15.0 tvOS 16.0  
public func evaluation DataFrame  
MLClassifierMetrics  
  
/// Evaluates the classifier on the provided labeled data.  
///  
/// Evaluation should be done on a testing data set that the model has not  
seen as part of the training or  
/// validation data sets. The data should have feature columns with identical  
name and type to the  
/// training data, as well as a labels column with the same name.  
///  
/// – Parameters:  
/// – labeledData: An `MLDataTable` to evaluate the trained model  
on.  
///  
/// – Returns: Metrics that describe the classification errors  
/// (` `MLClassifierMetrics/classificationError` `), the precision  
and recall  
/// percentages (` `MLClassifierMetrics/precisionRecall` `), and  
a table that  
/// describes how labels were misapplied
```

```

(``MLClassifierMetrics/confusion``)
    /// on the provided data.
    @available(macOS 10.14, iOS 13.0)
        "Use DataFrame instead of MLDataTable."
    @available(macOS 15.0, iOS 16.0)
        "Use DataFrame instead of MLDataTable."
    @available(macOS 16.0, iOS 17.0)
        public func evaluation() -> MLDataTable
MLClassifierMetrics

    /// Exports a Core ML model file for use in your app.
    public func write(to url: URL) throws
MLModelMetadata nil throws

    /// Exports a Core ML model file for use in your app.
    public func write(to string: String) throws
MLModelMetadata nil throws

extension MLDecisionTreeClassifier
    /// Parameters that affect the process of training a model.
    @available(macOS 10.14, iOS 15.0, tvOS 16.0)
    public struct ModelParameters {
        /// The maximum depth of the tree. Must be greater than 0.
        public var maxDepth: Int
        /// The data used for the validation set to inform the model training
process.
        @available(macOS 10.14, iOS 15.0, tvOS 16.0)
            "Use the validation property instead."
        @available(macOS 10.15, iOS 15.0, tvOS 16.0)
            "Use the validation property instead."
        @available(macOS 10.15, iOS 15.0, tvOS 16.0)
            public var validationData: MLDataTable
        /// Validation data.
        ///
        /// The default is `split(strategy: .automatic)`, which
automatically generates the validation
        /// dataset from 0% to 10% of the training dataset.
        @available(macOS 10.15, iOS 15.0, tvOS 16.0)
            public var validation: ValidationData
MLDecisionTreeClassifier ModelParameters ValidationData

        /// The minimum amount that the loss needs to be reduced to create a
new node.
        /// split.
        public var minLossReduction: Double

```

```

    /// The minimum weight of each leaf node.
    ///
    /// The minimum child weight controls when the tree building should
    terminate based comparing the sum of the
    /// instance weights to the
``MLDecisionTreeClassifier/ModelParameters-swift.struct/minChi
ldWeight``.
public var minChildWeight Double

    /// The seed value for random operations during tree building process.
    ///
    /// Set the random seed value to ensure results are reproducible.
public var randomSeed Int

@available macOS 10.15 iOS 15.0 tvOS 16.0
public init

MLDecisionTreeClassifier ModelParameters ValidationData
    Int 6
    Double 0
    Int 42

    /// Creates a new set of parameters.
    ///
    /// - Parameters:
    ///   - validationData: The dataset used to monitor how well the
    model is generalizing.
    ///
    ///   The default value is `nil` which will use an automatically
    sampled validation set.
    ///
    ///   - maxDepth: The maximum depth of the tree. Must be a value
    of at least 1.
    ///
    ///   The default value is 6.
    ///
    ///   - minLossReduction: The minimum amount of reduction to
    the loss function that is required to make another
    ///   node to split the data. Larger values help prevent overfitting.
    ///
    ///   The default value is 0.
    ///
    ///   - minChildWeight: Determines the minimum weight of each
    leaf node of the tree. Larger values help prevent
    ///   overfitting.
    ///
    ///   The default value is 0.1.
    ///
    ///   - randomSeed: A seed for internal random operations. Set this
    value to ensure reproducible results.

```

```
///  
///      The default value is 42.  
@available(macOS 10.15, iOS 15.0, tvOS 16.0)  
"Use the validation property instead."  
@available(15.0, 16.0)  
"Use the validation property instead."  
@available  
public init  
MLDataTable  
Int 6 Double 0 Double  
0.1 Int 42
```

```
@available(macOS 10.14, iOS 15.0, tvOS 16.0)  
extension MLDecisionTreeClassifier : CustomStringConvertible,  
CustomDebugStringConvertible,  
CustomPlaygroundDisplayConvertible
```

```
/// A text representation of the decision tree classifier.  
public var description String get  
  
/// A text representation of the decision tree classifier that's suitable  
/// for output during debugging.  
public var debugDescription String get  
  
/// A description of the decision tree classifier shown in a playground.  
public var playgroundDescription Any get
```

```
extension MLDecisionTreeClassifier : ModelParameters
```

```
/// Values for specifying validation data.  
@available(macOS 10.15, iOS 15.0, tvOS 16.0)  
public enum ValidationData  
  
    /// Generate validation data by splitting the training dataset. This is the  
    default.  
    case split MLSplitStrategy  
  
    /// Set validation data from the MLDataTable provided.  
    @available(macOS 10.15, iOS 15.0, tvOS 16.0)  
    @available(14.0, 17.0)  
    @available(15.0, 17.0)  
    @available(16.0, 17.0)  
    case table MLDataTable  
  
    /// Validation data provided in a DataFrame.  
    @available(macOS 12.0, iOS 15.0, tvOS 16.0)  
    case dataFrame DataFrame
```

```
    /// Do not set validation data.  
    case none  
  
  
@available macOS 10.14 iOS 15.0 tvOS 16.0  
extension MLDecisionTreeClassifier ModelParameters  
CustomStringConvertible CustomDebugStringConvertible  
CustomPlaygroundDisplayConvertible  
  
    /// A text representation of the model parameters for a decision tree  
    classifier.  
    public var description String get  
  
    /// A text representation of the model parameters for a decision tree classifier  
    that's suitable for output during  
    /// debugging.  
    public var debugDescription String get  
  
    /// A description of the model parameters for a decision tree classifier shown  
    in a playground.  
    public var playgroundDescription Any get  
  
  
    /// A regressor that estimates the target by learning rules to split the data.  
@available macOS 10.14 iOS 15.0 tvOS 16.0  
public struct MLDecisionTreeRegressor @unchecked Sendable  
  
    /// The Core ML model.  
    public var model MLModel  
  
    /// The name of the column you selected at initialization to define which  
    feature the regressor predicts.  
    ///  
    /// Changing the value of this property doesn't retrain the model or affect its  
    behavior.  
    public var targetColumn String  
  
    /// The names of the columns you selected at initialization to train the  
    regressor.  
    ///  
    /// Changing the value of this property doesn't retrain the model or affect its  
    behavior.  
    public var featureColumns String  
  
    /// The underlying parameters used when training the model.  
    public let modelParameters  
MLDecisionTreeRegressor ModelParameters  
  
    /// Measurements of the regressor's performance on the training data set.
```

```
public var trainingMetrics MLRegressorMetrics get
    /// Measurements of the regressor's performance on the validation data set.
public var validationMetrics MLRegressorMetrics get
    /// Creates a decision tree regressor.
    ///
    /// - Parameters:
    ///   - trainingData: The training data
    ///   - targetColumn: Name of the column containing the target values
    ///   - featureColumns: Names of the columns containing feature
    ///   values. If `nil` all columns, other than the target
    ///   column, will be used as feature values.
    ///   - parameters: Model training parameters
@available macOS 12.0 iOS 15.0 tvOS 16.0
public init DataFrame String
String nil
MLDecisionTreeRegressor ModelParameters

throws

    /// Creates a Decision Tree Regressor from the feature columns in the
    /// training data to predict the values in the
    /// target column.
    ///
    /// - Parameters:
    ///   - trainingData: A data table of training examples.
    ///   - targetColumn: The column name for the values in the training
    ///   data the regressor should predict.
    ///   - featureColumns: The column names for the values in the
    ///   training data that the regressor uses to predict
    ///   the target value.
    ///   - parameters: The model parameters.
@available 10.14 13.0
    "Use DataFrame instead of MLDataTable when
initializing."
@available 15.0 16.0
    "Use DataFrame instead of MLDataTable when
initializing."
@available
public init MLDataTable
String String nil
MLDecisionTreeRegressor ModelParameters
nil throws

    /// Creates a decision tree regressor from a checkpoint.
    ///
    /// - Parameter checkpoint: Training checkpoint.
    /// - Throws: `MLCreateError` if the checkpoint can't be loaded.
@available macOS 12.0 iOS 15.0 tvOS 16.0
```

```

public init           MLCheckpoint throws

    /// Trains a decision tree regressor.
    ///
    /// If `sessionDirectory` is provided it will save training progress. If
    there is progress already saved training
    /// will resume from the last checkpoint.
    ///
    /// - Parameters:
    ///   - trainingData: A DataTable specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
    trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
    columns to be
    ///
    default to use all the
    ///
    one specified by targetColumn
    ///   - parameters: Model training parameters. See
    `MLDecisionTreeRegressor.ModelParameters` for the defaults.
    ///   - sessionParameters: Training session parameters. See
    `MLTrainingSessionParameters` for the defaults.
    ///
    /// - Returns: A `MLJob` that can be used to observe training progress.
@available           12.0           14.0
@available           15.0           17.0
@available           16.0           17.0
public static func train           MLDataTable
String                 String           nil
MLDecisionTreeRegressor ModelParameters

MLTrainingSessionParameters
throws
MLJob MLDecisionTreeRegressor

    /// Trains a decision tree regressor.
    ///
    /// If `sessionDirectory` is provided it will save training progress. If
    there is progress already saved training
    /// will resume from the last checkpoint.
    ///
    /// - Parameters:
    ///   - trainingData: A `DataFrame` specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
    trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
    columns to be
    ///
    default to use all the
    ///
    one specified by targetColumn

```

```

    /**
     * - parameters: Model training parameters. See
     * `MLDecisionTreeRegressor.ModelParameters` for the defaults.
     * - sessionParameters: Training session parameters. See
     * `MLTrainingSessionParameters` for the defaults.
     */
    /**
     * - Returns: A `MLJob` that can be used to observe training progress.
     * @available macOS 12.0 iOS 15.0 tvOS 16.0
     * public static func train
     *   String           DataFrame
     *   String           nil
     *   MLDecisionTreeRegressor ModelParameters
     *
     *   MLTrainingSessionParameters
     *   throws
     * MLJob MLDecisionTreeRegressor
     */
    /**
     * Creates or restores a training session.
     */
    /**
     * - Parameters:
     * - trainingData: A DataTable specifying training data.
     * - targetColumn: A String specifying the target column name in the
     * trainingData
     * - featureColumns: An optional list of Strings specifying feature
     * columns to be
     *   used to predict the target, if not provided, default to use
     * all the
     *   other columns in the trainingData, except the one
     * specified by targetColumn
     * - parameters: Model training parameters. See
     * `MLDecisionTreeRegressor.ModelParameters` for the defaults.
     * - sessionParameters: Training session parameters. See
     * `MLTrainingSessionParameters` for the defaults.
     */
    /**
     * - Returns: A `MLTrainingSession` that can be used to start or
     * resume training.
     * @available           12.0          14.0
     * @available           15.0          17.0
     * @available           16.0          17.0
     * public static func makeTrainingSession
     *   MLDataTable        String         String
     *   nil                MLDecisionTreeRegressor ModelParameters
     *
     *   MLTrainingSessionParameters
     *   throws
     * MLTrainingSession MLDecisionTreeRegressor
     */
    /**
     * Creates or restores a training session.
     */
    /**
     * - Parameters:
     * - trainingData: A `DataFrame` specifying training data.
     * - targetColumn: A String specifying the target column name in the

```

```
trainingData
    /// - featureColumns: An optional list of Strings specifying feature
columns to be
    /// used to predict the target, if not provided, default to use
all the
    /// other columns in the trainingData, except the one
specified by targetColumn
    /// - parameters: Model training parameters. See
`MLDecisionTreeRegressor.ModelParameters` for the defaults.
    /// - sessionParameters: Training session parameters. See
`MLTrainingSessionParameters` for the defaults.
    ///
    /// - Returns: A `MLTrainingSession` that can be used to start or
resume training.
@available macOS 12.0 iOS 15.0 tvOS 16.0
public static func makeTrainingSession
DataFrame String String
nil MLDecisionTreeRegressor ModelParameters

MLTrainingSessionParameters
throws
MLTrainingSession MLDecisionTreeRegressor

    /// Restores an existing training session.
    ///
    /// - Parameters:
    /// - sessionParameters: Training session parameters. The
`sessionDirectory` parameter is required.
    ///
    /// - Returns: A `MLTrainingSession` that can be used to resume
training.
@available macOS 12.0 iOS 15.0 tvOS 16.0
public static func
restoreTrainingSession
MLTrainingSessionParameters throws
MLTrainingSession MLDecisionTreeRegressor

    /// Resumes a training session from the last checkpoint if available.
    ///
    /// If there are no resumable checkpoints training starts over from the
beginning.
    ///
    /// - Parameter session: Loaded or new training session.
    ///
    /// - Returns: A `MLJob` that can be used to observe training progress.
@available macOS 12.0 iOS 15.0 tvOS 16.0
public static func resume
MLTrainingSession MLDecisionTreeRegressor throws
MLJob MLDecisionTreeRegressor
```

```
    /// Predicts a column of labels for the given testing data.
    @available macOS 12.0  iOS 15.0  tvOS 16.0
    public func predictions           DataFrame  throws
AnyColumn

    /// Predicts the target value from the provided data.
    ///
    /// - Parameters:
    ///   - data: The data you want the model to make predictions from.
    ///
    /// - Returns: A column of values predicted by the regressor.
    @available 10.14          13.0
        "Use DataFrame instead of MLDataTable."
    @available 15.0           16.0
        "Use DataFrame instead of MLDataTable."
    @available
    public func predictions       MLDataTable  throws
MLUntypedColumn

    /// Evaluates the classifier on the provided labeled data.
    ///
    /// Evaluation should be done on a testing data set that the model has not
    seen as part of the training or
    /// validation data sets. The data should have feature columns with identical
    name and type to the
    /// training data, as well as a labels column with the same name.
    ///
    /// - Parameters:
    ///   - labeledData: A `DataFrame` to evaluate the trained model on.
    ///
    /// - Returns: Metrics that describe the maximum error
    ///   (`MLRegressorMetrics/maximumError`) or the average error
    ///   (`MLRegressorMetrics/rootMeanSquaredError`).
    @available macOS 12.0  iOS 15.0  tvOS 16.0
    public func evaluation        DataFrame
MLRegressorMetrics

    /// Evaluates the classifier on the provided labeled data.
    ///
    /// Evaluation should be done on a testing data set that the model has not
    seen as part of the training or
    /// validation data sets. The data should have feature columns with identical
    name and type to the
    /// training data, as well as a labels column with the same name.
    ///
    /// - Parameters:
    ///   - labeledData: An `MLDataTable` to evaluate the trained model
    on.
    ///
    /// - Returns: Metrics that describe the maximum error
```

```

    ///(``MLRegressorMetrics/maximumError``) or the average error
    ///(``MLRegressorMetrics/rootMeanSquaredError``).
    @available(10.14, 13.0)
        "Use DataFrame instead of MLDataTable."
    @available(15.0, 16.0)
        "Use DataFrame instead of MLDataTable."
    @available
        public func evaluation MLDataTable
    MLRegressorMetrics

    /// Exports a Core ML model file for use in your app.
    public func write URL
    MLModelMetadata nil throws

    /// Exports a Core ML model file for use in your app.
    public func write String
    MLModelMetadata nil throws

extension MLDecisionTreeRegressor

    /// Parameters that affect the process of training a model.
    @available(macOS 10.14, iOS 15.0, tvOS 16.0)
    public struct ModelParameters

        /// Validation data represented as a `MLDataTable`.
        ///
        /// - Note: Setting this to `nil` means that the training data will be
        automatically split for
        /// validation. Setting it to an empty table means to not use a
        validation set.
        @available(10.14, 11.0)
            "Use the validation property instead."
        @available(15.0, 16.0)
            "Use the validation property instead."
        @available
            public var validationData MLDataTable

        /// Validation data.
        ///
        /// The default is `.split(strategy: .automatic)`, which
        automatically generates the validation
        /// dataset from 0% to 10% of the training dataset.
        @available(macOS 11.0, iOS 15.0, tvOS 16.0)
        public var validation
    MLDecisionTreeRegressor ModelParameters ValidationData

        public var maxDepth Int
        public var minLossReduction Double

```

```

public var minChildWeight Double
public var randomSeed Int

@available macOS 11.0 iOS 15.0 tvOS 16.0
public init
MLDecisionTreeRegressor ModelParameters ValidationData
    Int 6                      Double 0
    Double 0.1                  Int 42

    /// Creates a new set of parameters.
    ///
    /// - Parameters:
    ///     - validationData: The dataset used to monitor how well the
model is generalizing.
    ///
    ///     The default value is `nil` which will use an automatically
sampled validation set.
    ///
    ///     - maxDepth: The maximum depth of the tree. Must be a value
of at least 1.
    ///
    ///     The default value is 6.
    ///
    ///     - minLossReduction: The minimum amount of reduction to
the loss function that is required to make another
    ///         node to split the data. Larger values help prevent overfitting.
    ///
    ///     The default value is 0.
    ///
    ///     - minChildWeight: Determines the minimum weight of each
leaf node of the tree. Larger values help prevent
    ///         overfitting.
    ///
    ///     The default value is 0.1.
    ///
    ///     - randomSeed: A seed for internal random operations. Set this
value to ensure reproducible results.
    ///
    ///     The default value is 42.
@available 10.14 11.0
    "Use the validation property instead."
@available 15.0 16.0
    "Use the validation property instead."
public init MLDataTable nil
    Int 6                      Double 0
    Double 0.1                  Int 42

```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDecisionTreeRegressor CustomStringConvertible
CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of the decision tree regressor.
public var description String get

    /// A text representation of the decision tree regressor that's suitable for
    /// output during debugging.
public var debugDescription String get

    /// A description of the decision tree regressor shown in a playground.
public var playgroundDescription Any get

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLDecisionTreeRegressor ModelParameters
CustomStringConvertible CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of the model parameters for a decision tree
    regressor.
public var description String get

    /// A text representation of the model parameters for a decision tree
    regressor that's suitable for output during
    /// debugging.
public var debugDescription String get

    /// A text representation of the model parameters for a decision tree
    regressor.
public var playgroundDescription Any get

extension MLDecisionTreeRegressor ModelParameters

    /// Values for specifying validation data.
@available macOS 11.0 iOS 15.0 tvOS 16.0
public enum ValidationData

        /// Generate validation data by splitting the training dataset. This is the
        default.
case split MLSplitStrategy

        /// Set validation data from the MLDataTable provided.
@available 11.0 14.0
@available 15.0 17.0
```

|   |      |      |
|---|------|------|
| <b>@available</b>                           | 16.0 | 17.0 |
| <b>case</b> <b>table</b> <b>MLDataTable</b> |      |      |

```
/// Set validation data from the DataFrame provided.  
@available macOS 13.0 iOS 16.0 tvOS 16.0  
case dataFrame DataFrame  
  
/// Do not set validation data.  
case none
```

```
/// A collection of terms and their labels, which augments a tagger that analyzes  
natural language text.  
///  
/// Use an ``MLGazetteer`` to configure a gazetteer and save it to a file, which  
you then add to your app in Xcode.  
/// Your app uses the gazetteer file at runtime to create an instance of  
/// <doc://com.apple.documentation/documentation/naturallanguage/nlgazetteer>,  
/// which augments an  
/// <doc://com.apple.documentation/documentation/naturallanguage/nltagger> to  
/// tag specific terms with a label.  
///  
/// You configure a gazetteer with a dictionary, keyed by labels. Each value in the  
dictionary is an array  
/// of terms (words or phrases) for each label. For example, you can store the  
names of real and fictional planets in a  
/// gazetteer.  
///  
/// ````swift  
/// let planets = [  
///     "real planet": ["Mercury", "Venus", "Earth", "Mars",  
///     "Jupiter", "Saturn", "Uranus", "Neptune"],  
///     "fictional planet" : ["Arrakis", "Hoth", "Vulcan",  
///     "Pandora", "Tatooine", "Bajor", "Alderaan", "Romulus"]  
/// ]  
///  
/// let parameters =  
MLGazetteer.ModelParameters(language: .english)  
/// let planetGazetteer = try! MLGazetteer(dictionary:  
planets, parameters: parameters)  
/// ````  
///  
/// Once you've configured an ``MLGazetteer``, save it to an ` `.mlmodel` file  
to include in your app.  
///  
/// ````swift  
/// try planetGazetteer.write(toFile:  
/// "~/Desktop/PlanetGazetteer.mlmodel")  
/// ````
```

```
///  
/// A gazetteer file can efficiently store many labels, and many terms for each  
label.  
@available macOS 10.15  
@available  
@available  
public struct MLGazetteer  
  
    /// The gazetteer contained within a Core ML model file stored in memory.  
public var model MLModel  
  
    /// The model configuration parameters.  
public let modelParameters MLGazetteer ModelParameters  
  
    /// Creates a gazetteer from a dictionary of labels and terms.  
    ///  
    /// - Parameters:  
    ///   - dictionary: A dictionary of labels and terms.  
    ///   - parameters: The model parameters.  
    public init String String  
MLGazetteer ModelParameters throws  
  
    /// Creates a gazetteer from a table of labels and terms.  
    ///  
    /// - Parameters:  
    ///   - labeledData: A table of labels and terms.  
    ///   - textColumn: The name of the column containing the terms.  
    ///   - labelColumn: The name of the column containing the labels.  
    ///   - parameters: The model parameters.  
@available 10.15 14.0  
@available  
@available  
public init MLDDataTable String  
String MLGazetteer ModelParameters  
throws  
  
@available macOS 10.15  
@available  
@available  
extension MLGazetteer  
  
    /// Exports the gazetteer as a Core ML model file at the specified URL.  
    ///  
    /// - Parameters:  
    ///   - fileURL: The location in the file system to which the file should be  
written.  
    ///   - metadata: Descriptive information to include with the exported  
model file.  
public func write URL
```

```
MLModelMetadata    nil    throws

    /// Exports the gazetteer as a Core ML model file at the specified file path.
    ///
    /// - Parameters:
    ///   - path: A file system path where the model file should be written.
    ///   - metadata: Descriptive information to include with the exported
model file.
public func write           String
MLModelMetadata    nil    throws

@available macOS 10.15
@available
@available
extension MLGazetteer  CustomStringConvertible
CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of the gazetteer.
public var description  String  get

    /// A text representation of the gazetteer that's suitable for output during
debugging.
public var debugDescription  String  get

    /// A description of the gazetteer shown in a playground.
public var playgroundDescription  Any  get

@available macOS 10.15
@available
@available
extension MLGazetteer

    /// The model configuration parameters.
public struct ModelParameters  Sendable

        /// The language setting.
public var language  NLLanguage

        /// Creates model parameters.
///
/// - Parameter language: The language of the text in the
gazetteer.
public init          NLLanguage  nil

@available macOS 10.15
```

```
@available
@available
extension MLGazetteer

    /// Predicts the label for the given term.
    ///
    /// - Parameter text: The input term.
    /// - Returns: A label.
    public func prediction String throws String

    /// Predicts the labels for the given terms.
    ///
    /// - Parameter texts: The array of input terms.
    /// - Returns: An array of labels.
    public func predictions String throws
String

    /// Predicts the labels for the given terms in the table column.
    ///
    /// - Parameter texts: The column of terms.
    /// - Returns: A column of labels.
    @available 10.15 14.0
    @available
    @available
    public func predictions MLDataColumn String
throws MLDataColumn String

@available macOS 10.15
@available
@available
extension MLGazetteer ModelParameters
CustomStringConvertible CustomDebugStringConvertible
CustomPlaygroundConvertible

    /// A text representation of the gazetteer settings.
    public var description String get

    /// A text representation of the gazetteer settings that's suitable for output
during debugging.
    public var debugDescription String get

    /// A description of the gazetteer settings shown in a playground.
    public var playgroundDescription Any get

    /// A task that creates a hand action classification model by training with
/// videos of people's hand movements that you provide.
@available macOS 12.0 iOS 15.0
```

```
@available
public struct MLHandActionClassifier

    /// The underlying Core ML model of the hand action classifier stored in
    /// memory.
    public var model MLModel

    /// The hand action model's configuration parameters.
    ///
    /// The property reflects the model parameters you provide to one of these
    /// methods that train a hand action classifier:
    ///
    /// -
    ///
``MLHandActionClassifier/train(trainingData:parameters:sessionParameters:)``
    /// -
    ///
``MLHandActionClassifier/makeTrainingSession(trainingData:parameters:sessionParameters:)``
    /// -
``MLHandActionClassifier/init(trainingData:parameters:)``
    public let modelParameters
MLHandActionClassifier ModelParameters

    /// Measurements of the hand action classifier's performance on the training
    /// dataset.
public var trainingMetrics MLClassifierMetrics

    /// Measurements of the hand action classifier's performance on the
    /// validation dataset.
public var validationMetrics MLClassifierMetrics

    /// A collection of predictions, each paired with its confidence, for a
    /// range of video frames.
public struct Prediction Sendable

    /// The range of frame rates the hand action classifier used to make its
    /// prediction.
public var frameRange Range Int

    /// An array of prediction labels and their confidences for a hand
    /// action.
public var results String
Double

    /// Creates a hand action classifier by starting a synchronous training
    /// session.
```

```
    /**
     * - Parameters:
     *   - trainingData: An
     *     ``MLHandActionClassifier/DataSource`` instance.
     *   - parameters: An
     *     ``MLHandActionClassifier/ModelParameters-swift.struct``
     *       instance you use to configure the model for the training session.
     */
    public init
        MLHandActionClassifier DataSource
        MLHandActionClassifier ModelParameters
    throws

    /**
     * Creates a hand action classifier from a training session checkpoint.
     */
    /**
     * - Parameters:
     *   - checkpoint: An ``MLCheckpoint`` instance from a hand action
     *     training
     *   - session.
     */
    public init MLCheckpoint throws

    /**
     * Begins an asynchronous hand action classifier's training session.
     */
    /**
     * - Parameters:
     *   - trainingData: An
     *     ``MLHandActionClassifier/DataSource`` instance.
     */
    /**
     *   - parameters: An
     *     ``MLHandActionClassifier/ModelParameters-swift.struct``
     *       instance you use to configure the model for the training session.
     */
    /**
     *   - sessionParameters: An
     *     ``MLTrainingSessionParameters`` instance you use
     *       to configure the training session.
     */
    /**
     * - Returns: An ``MLJob`` that represents the hand action classifier's
     *   training session.
     */
    public static func train
        MLHandActionClassifier DataSource
        MLHandActionClassifier ModelParameters
            MLTrainingSessionParameters
        throws
    MLJob MLHandActionClassifier

    /**
     * Creates an asynchronous hand action classifier's training session.
     */
    /**
     * - Parameters:
     *   - trainingData: An
     *     ``MLHandActionClassifier/DataSource`` instance.
     */

```

```
    /// - parameters: An
``MLHandActionClassifier/ModelParameters-swift.struct``
    /// instance you use to configure the model for the training session.
    ///
    /// - sessionParameters: An ``MLTrainingSessionParameters``
instance you use
    /// to configure the training session.
    ///
    /// - Returns: An ``MLTrainingSession`` that represents the action
    /// classifier training session.
public static func makeTrainingSession
MLHandActionClassifier DataSource
MLHandActionClassifier ModelParameters
    MLTrainingSessionParameters
throws
MLTrainingSession MLHandActionClassifier

    /// Recreates an asynchronous hand action classifier's training session by
    /// restoring its saved state from the file system.
    ///
    /// - Parameters:
    ///   - sessionParameters: The same
``MLTrainingSessionParameters`` instance
    /// that created an existing training session.
    ///
    /// - Returns: An ``MLTrainingSession`` that represents the hand
action
    /// classifier training session.
public static func
restoreTrainingSession
MLTrainingSessionParameters throws
MLTrainingSession MLHandActionClassifier

    /// Begins or continues an asynchronous hand action classifier's training
    /// session.
    ///
    /// - Parameters:
    ///   - session: An ``MLTrainingSession`` instance that represents
the
    ///   training session.
    ///
    /// - Returns: An ``MLJob`` that represents the hand action training
    /// session.
    ///
    ///
public static func resume _
MLTrainingSession MLHandActionClassifier throws
MLJob MLHandActionClassifier

    /// Generates a hand action prediction for a video.
```

```
///  
/// - Parameters:  
///   - video : A video file URL.  
///  
/// - Returns: An array of predictions.  
public func prediction URL throws  
MLHandActionClassifier Prediction  
  
/// Generates an array of hand action predictions for each video in a URL  
/// array.  
///  
/// - Parameters:  
///   - videos: An array of video file URLs.  
///  
/// - Returns: A array of prediction arrays. The index of each inner array  
/// corresponds to the video URL index in the input array.  
public func predictions URL throws  
MLHandActionClassifier Prediction  
  
/// Generates metrics describing the hand action classifier's performance on  
/// labeled videos.  
///  
/// - Parameters:  
///   - annotatedVideos : An  
``MLHandActionClassifier/DataSource`` instance.  
public func evaluation  
MLHandActionClassifier DataSource throws  
MLClassifierMetrics  
  
/// Exports the hand action classifier as a CoreML model file.  
///  
/// - Parameters:  
///   - fileURL: A file-system URL.  
///   - metadata: The model's description, author, version, and license  
information.  
public func write URL  
MLModelMetadata nil throws  
  
/// Exports the hand action classifier as a Core ML model file.  
///  
/// - Parameters:  
///   - path: A file-system path.  
///   - metadata: The model's description, author, version, and license  
information.  
public func write String  
MLModelMetadata nil throws
```

**@available** macOS 12.0 iOS 15.0  
**@available**

```
extension MLHandActionClassifier

    /// A set of parameters that affect the training process of a hand action
    /// classifier task.
public struct ModelParameters

    /// A dataset the hand action classifier task uses to evaluate the model
    /// that's distinct from the training dataset.
    ///
    /// The task produces
``MLHandActionClassifier/validationMetrics`` by
    /// evaluating the model with the
    /// ``MLHandActionClassifier/ModelParameters-
swift.struct/validation``
    /// dataset.
public var validation
MLHandActionClassifier ModelParameters ValidationData

    /// The number of videos the model training session uses for each
    /// training iteration.
public var batchSize Int

    /// The largest number of training iterations you allow the training
    /// session to use.
public var maximumIterations Int

    /// The number of video frames the model training session uses to train
    /// a hand action classifier.
    ///
    /// Set to 60 to capture hand actions with a 2 second duration from
    /// videos that have a frame rate of 30 frames per second.
public var predictionWindowSize Int

    /// The variations the training session uses to add more variety to its
    /// training dataset.
public var augmentationOptions
MLHandActionClassifier VideoAugmentationOptions

    /// The algorithm the training session uses to create the hand action
    /// classifier.
public var algorithm
MLHandActionClassifier ModelParameters ModelAlgorithmType

    /// The number of video frames per second the hand action classifier
    /// model expects as its input at runtime.
public var targetFrameRate Double

    /// Creates a set of training session parameters for a hand action
    /// classifier task.
```

```

    /**
     * - Parameters:
     *   - validation: An
     *   - ``MLHandActionClassifier/ModelParameters-
     swift.struct/ValidationData``
     *   - instance.
     *
     *   - batchSize: The number of videos the training session uses for
each
     *   - of its training iterations.
     *
     *   - maximumIterations: The largest number of training iterations the
     *   - training session can use.
     *
     *   - predictionWindowSize: The number of frames the training session
     *   - uses to train a hand action classifier. For example, set to 60 to
     *   - capture hand actions that take 2 seconds from videos that have a
     *   - frame rate of 30 frames per second.
     *
     *   - augmentationOptions: The variations the training session uses to
     *   - add more variety to its training dataset.
     *
     *   - algorithm: The algorithm the training session uses to train the
     *   - hand action classifier.
     *
     *   - targetFrameRate: The number of frames the training session
uses
     *   - per second of video to train a hand action classifier.
public init
MLHandActionClassifier ModelParameters ValidationData
                        Int
                        Int
                        Int

```

```

MLHandActionClassifier VideoAugmentationOptions
MLHandActionClassifier ModelParameters ModelAlgorithmType
                        Double

```

```

/// The hand action classifier training algorithm options.
public enum ModelAlgorithmType : Equatable, Sendable

/// Selects the graph convolutional neural-network algorithm for a
/// hand action classifier.
case gcn

```

```

/// Hashes the essential components of this value by feeding
them into the
/// given hasher.
///
/// Implement this method to conform to the `Hashable`
protocol. The
/// components used for hashing must be the same as the
components compared
/// in your type's `==` operator implementation. Call
`hasher.combine(_:)`
/// with each of these components.
///
/// - Important: In your implementation of `hash(into:)`,  

///   don't call `finalize()` on the `hasher` instance
provided,  

/// or replace it with a different instance.  

/// Doing so may become a compile-time error in the future.
///
/// - Parameter hasher: The hasher to use when combining
the components
/// of this instance.
public func hash inout Hasher

/// Returns a Boolean value indicating whether two values are
equal.
///
/// Equality is the inverse of inequality. For any values `a` and
`b`,
/// `a == b` implies that `a != b` is `false`.
///
/// - Parameters:
///   - lhs: A value to compare.
///   - rhs: Another value to compare.
public static func
MLHandActionClassifier ModelParameters ModelAlgorithmType
MLHandActionClassifier ModelParameters ModelAlgorithmType
Bool

/// The hash value.
///
/// Hash values are not guaranteed to be equal across different
executions of
/// your program. Do not save hash values to use during a future
execution.
///
/// - Important: `hashValue` is deprecated as a
`Hashable` requirement. To
/// conform to `Hashable`, implement the `hash(into:)`  

requirement instead.
/// The compiler provides an implementation for `hashValue`  

for you.

```

```
public var hashCode Int get

    /// A dataset a hand action classifier task uses to validate the model
    /// during a training session.
public enum ValidationData

    /// Creates a validation dataset by randomly selecting a portion of
    /// the hand action classifier task's training dataset with a split
    /// strategy.
    ///
    /// - Parameters:
    ///   - strategy: An ``MLSplitStrategy`` instance.
    case split MLSplitStrategy

    /// Creates a validation dataset from a data source.
    ///
    /// - Parameters:
    ///   - dataSource: An
``MLHandActionClassifier/DataSource`` instance.
    case dataSource MLHandActionClassifier DataSource

    /// Creates an empty validation dataset, which tells the task to
    /// skip the model validation phase in a training session.
    ///
    /// This case prevents Create ML from creating a validation
dataset
    /// from your training dataset. You typically use it when your
    /// training dataset is small and can't spare any data to validate
    /// the model.
    case none
```

```
@available macOS 12.0 iOS 15.0
@available
extension MLHandActionClassifier : CustomStringConvertible
: CustomDebugStringConvertible
: CustomPlaygroundDisplayConvertible

    /// A text representation of the hand action classifier.
    public var description String get

    /// A text representation of the hand action classifier suitable for
    /// debugging.
    public var debugDescription String get

    /// A description of the hand action classifier that's viewable in a
```

```
/// playground.
public var playgroundDescription Any get

@available(macOS 12.0, iOS 15.0)
@available
extension MLHandActionClassifier

    /// Options a hand action classification training session can use to
    /// generate additional training data from the videos you provide.
    public struct VideoAugmentationOptions : OptionSet
Codable Sendable

    /// The corresponding value of the raw type.
    ///
    /// A new instance initialized with `rawValue` will be equivalent to this
    /// instance. For example:
    ///
    ///     enum PaperSize: String {
    ///         case A4, A5, Letter, Legal
    ///     }
    ///
    ///     let selectedSize = PaperSize.Letter
    ///     print(selectedSize.rawValue)
    ///     // Prints "Letter"
    ///
    ///     print(selectedSize == PaperSize(rawValue:
selectedSize.rawValue)!)
    ///     // Prints "true"
    public let rawValue Int

    /// Apply left-right flips to the pose in a video.
    public static let horizontallyFlip
MLHandActionClassifier VideoAugmentationOptions

    /// Randomly rotate the pose in a video.
    public static let rotate
MLHandActionClassifier VideoAugmentationOptions

    /// Randomly translate the pose in a video.
    public static let translate
MLHandActionClassifier VideoAugmentationOptions

    /// Randomly scale the pose in a video.
    public static let scale
MLHandActionClassifier VideoAugmentationOptions

    /// Random time interpolation through a video.
    public static let interpolateFrames
```

```
MLHandActionClassifier VideoAugmentationOptions

    /// Randomly drop frames from a video.
    public static let dropFrames
MLHandActionClassifier VideoAugmentationOptions

    /// Creates an option set from an integer.
    ///
    /// - Parameters:
    ///   - rawValue: An integer that represents the option set's
underlying
    /// raw value.
    public init Int

    /// The type of the elements of an array literal.
    @available iOS 15.0 macOS 12.0
    @available
    public typealias ArrayLiteralElement
MLHandActionClassifier VideoAugmentationOptions

    /// The element type of the option set.
    ///
    /// To inherit all the default implementations from the `OptionSet`'
protocol,
    /// the `Element` type must be `Self`, the default.
    @available iOS 15.0 macOS 12.0
    @available
    public typealias Element
MLHandActionClassifier VideoAugmentationOptions

    /// The raw type that can be used to represent all values of the
conforming
    /// type.
    ///
    /// Every distinct value of the conforming type has a corresponding
unique
    /// value of the `RawValue` type, but there may be values of the
`RawValue`
    /// type that don't have a corresponding value of the conforming type.
    @available iOS 15.0 macOS 12.0
    @available
    public typealias RawValue Int

@available macOS 12.0 iOS 15.0
@available
extension MLHandActionClassifier

    /// A hand action classifier dataset that contains annotated videos or hand
```

joint location data.

```
public enum DataSource
```

```
    /// Creates a data source from a folder that contains videos and an annotation file.
```

```
    ///
```

```
    /// - Parameters:
```

```
        /// - at: A URL to a folder in the file system that contains hand action videos.
```

```
        /// - annotationFile: A URL to a JSON or CSV file that contains annotations for the hand action videos.
```

```
        /// - videoColumn: The name of the column or key name in the annotation file that contains the video
```

```
        /// filenames.
```

```
        /// - labelColumn: The name of the column or key name in the annotation file that contains the hand action
```

```
        /// label names.
```

```
        /// - startTimeColumn: The name of the column or key name in the annotation file that contains the hand
```

```
        /// action's starting-time index in the video file. Otherwise `nil`, if every video example starts at the
```

```
        /// beginning of the video file.
```

```
        /// - endTimeColumn: The name of the column or key name in the annotation file that contains the hand
```

```
        /// action's ending-time in the video file. Otherwise `nil`, if every video example ends at the end of the
```

```
        /// video file.
```

```
case directoryWithVideosAndAnnotation      URL  
          URL           String           String  
          String         nil            String         nil
```

```
    /// Creates a data source from a folder with subfolders that each contain videos of a hand action.
```

```
    ///
```

```
    /// - Parameters:
```

```
        /// - at: The URL to a folder in the file system that contains folders of hand action videos. The data source
```

```
        /// uses the name of each folder as the classification label for the hand action videos it contains.
```

```
case labeledDirectories      URL
```

```
    /// Creates a data source from a folder that contains videos, each named after the hand action they represent.
```

```
    ///
```

```
    /// Create a hand action data source from a directory of videos with the `labeledFiles` case. You must name
```

```
        /// each video file with a hand action label, followed by a period and an arbitrary string, ending with the
```

```
        /// video file's extension. For example, you can name a hand action classifier's training files `Wave.3.mov`,
```

```
        /// `MoveCloser.1.mov`, `MoveCloser.2.mov`, and so on.
```

```

    /**
     * In this example, a hand action classifier would have at least two
     * class labels:
     */
    /**
     * - `Wave`
     * - `MoveCloser`
     */
    /**
     * - Parameters:
     *   - at: The URL to a folder in the file system that contains
     *     folders of hand action videos. The data source
     *       uses each file's name before the first period as its classification
     *     label.
case labeledFiles URL

    /**
     * Creates a data source from a data table of hand action observations
     * that each contain the locations of each
     * hand joint and an annotation.
    /**
    /**
     * - Parameters:
     *   - table: A data table that contains the hand-joint locations
     *     and annotations for a set of hand actions.
     *   - sessionIdColumn: The name of the column in the data
     *     table that contains the session identifiers.
     *   - labelColumn: The name of the column in the data table that
     *     contains the hand action label names.
     *   - featureColumn: The name of the column in the data table
     *     that contains the hand-joint location data.
     *   Each entry in the column must be an
<doc://com.apple.documentation/documentation/coreml/mlmultiarray>
    /**
     * instance — which you must wrap in an
``MLDataValue/MultiArrayType`` — that contains three dimensions:
    /**
    /**
     *   - The first dimension has a size of one.
     *   - The second dimension has three channels: the x-
     *     coordinate, the y-coordinate, and the confidence
     *   value, respectively.
     *   - The third dimension has 21 channels, one for each hand
joint.
@available 12.0 14.0
@available 15.0 17.0
@available
case labeledKeypointsData MLDataTable
    String
    String
    String

    /**
     * Creates a data source from a data table that contains the location
     * and annotation for a set of video files.
    /**
    /**
     * - Parameters:
     *   - table: A data table that contains the locations of annotations

```



```

    /// Creates a data source from a data frame that contains the location
and annotation for a set of video files.

    /**
     * - Parameters:
     *   - dataFrame: A data frame that contains the locations of
annotations for each hand action video file.
     *   - videoColumn: The name of the column in the data frame
that contains the video filenames.
     *   - labelColumn: The name of the column in the data frame
that contains the hand action label names.
     *   - startTimeColumn: The name of the column in the data
frame that contains the hand action's starting-time
     *   - index: index in the video file. Otherwise `nil`, if every video
example starts at the beginning of the video
     *   - file.
     *   - endTimeColumn: The name of the column in the data frame
that contains the hand action's ending-time in
     *   - the video file. Otherwise `nil`, if every video example ends
at the end of the video file.
  
```

`case labeledVideoDataFrame DataFrame  
String String String String String  
String String String String String  
String String String String String`

```

    /// Processes the data source and returns a data frame that contains
file URLs and annotations.
    /**
     * This method collects file names from the filesystem if necessary. If
the data source is already in table
     * format it renames the columns to the default column names. This
method returns nil if the data source
     * contains key points.
    @available macOS 14.0 iOS 17.0  

@available  

public func gatherAnnotatedFileNames throws  

DataFrame

```

```

    /// Generates a data table that contains a column for the data source's
    /// video file URLs and a column of annotations.
@available 12.0 14.0  

"Use gatherAnnotatedFileNames()"  

@available 15.0 17.0  

"Use gatherAnnotatedFileNames()"  

@available  

public func videosWithAnnotations throws  

MLDataTable

```

```

    /// Extracts key points from video files if necessary.
    /**
     * If the data source already contains keypoints, this method just
renames the data frame columns to the

```

```

    /// defaults.
    ///
    /// - Parameters:
    ///   - targetFrameRate: The number of frames per second the
method uses to extract body landmarks from the
    ///   data source.
    /// - Returns: A data frame that contains a column for hand joint
locations and a column of hand action
    ///   annotations.
@available macOS 14.0 iOS 17.0
@available
public func extractKeypoints Double  
throws
DataFrame

    /// Generates a data table that contains a column for hand joint
locations and a column of hand action
    /// annotations.
    ///
    /// - Parameters:
    ///   - targetFrameRate: The number of frames per second the
method uses to extract body landmarks from the
    ///   data source.
    ///
    ///   This parameter has no effect if the data source is either:
    ///
    ///
``MLHandActionClassifier/DataSource/labeledKeypointsDataFrame(
:_sessionIdColumn:labelColumn:featureColumn:)``
    ///
    -
``MLHandActionClassifier/DataSource/labeledKeypointsData(table
:_sessionIdColumn:labelColumn:featureColumn:)``
@available 12.0 14.0
  "Use extractKeypoints(targetFrameRate:)"
@available 15.0 17.0
  "Use extractKeypoints(targetFrameRate:)"
@available
public func keypointsWithAnnotations Double  
throws
MLDataTable

    /// Generates a data table by splitting the data source into strata.
    ///
    /// - Parameters:
    ///   - proportions: An array of proportions, each in the range
`[0.0, 1.0]`.
    ///   - seed: A seed number for the random-number generator.
    ///   - labelColumn: The name of the column that you want to
stratify.
    ///

```

```
/// - Returns: A new data table.  
@available(12.0, 14.0)  
"Use DataFrame.stratifiedSplit(on:by:)"  
@available(15.0, 17.0)  
"Use DataFrame.stratifiedSplit(on:by:)"  
@available  
public func stratifiedSplit(Double  
    Int, String) throws  
MLDataTable
```

```
@available(macOS 12.0, iOS 15.0)  
@available  
extension MLHandActionClassifier : ModelParameters,  
    CustomStringConvertible, CustomDebugStringConvertible,  
    CustomPlaygroundDisplayConvertible
```

```
/// A text representation of the hand action parameters.  
public var description : String { get }  
  
/// A text representation of the hand action parameters suitable for  
/// debugging.  
public var debugDescription : String { get }  
  
/// A description of the hand action parameters that's viewable in a  
/// playground.  
public var playgroundDescription : Any { get }
```

```
@available(macOS 12.0, iOS 15.0)  
@available  
extension MLHandActionClassifier : DataSource  
  
/// Generates a dictionary that contains the data source's classification  
/// labels paired with an array of URLs to the label's video files.  
public func labeledMedia() throws : String, URL
```

```
@available(macOS 12.0, iOS 15.0)  
@available  
extension MLHandActionClassifier : ModelParameters, ModelAlgorithmType,  
    Hashable
```

```
/// A task that creates a hand pose classification model by training with images  
/// of people's hands that you provide.  
@available(macOS 12.0, iOS 15.0)
```

```
@available
public struct MLHandPoseClassifier

    /// The underlying Core ML model of the hand pose classifier stored in
    /// memory.
    public var model MLModel

    /// The hand pose model's configuration parameters.
    ///
    /// The property reflects the model parameters you provide to one of these
    /// methods that train a hand pose classifier:
    ///
    /// -
    ///
``MLHandPoseClassifier/train(trainingData:parameters:sessionParameters:)``
    /// -
    ///
``MLHandPoseClassifier/makeTrainingSession(trainingData:parameters:sessionParameters:)``
    /// -
``MLHandPoseClassifier/init(trainingData:parameters:)``
    public let modelParameters
MLHandPoseClassifier ModelParameters

    /// Measurements of the hand pose classifier's performance on the training
    /// dataset.
    public var trainingMetrics MLClassifierMetrics

    /// Measurements of the hand pose classifier's performance on the validation
    /// dataset.
    public var validationMetrics MLClassifierMetrics

    /// Creates a hand pose classifier by starting a synchronous training
    /// session.
    ///
    /// - Parameters:
    ///   - trainingData: An
``MLHandPoseClassifier/DataSource`` instance.
    ///
    /// - parameters: An
``MLHandPoseClassifier/ModelParameters-swift.struct``
    /// instance you use to configure the model for the training session.
    public init MLHandPoseClassifier DataSource
        MLHandPoseClassifier ModelParameters
        throws

    /// Creates a hand pose classifier from a training session checkpoint.
    ///
```

```
    /// - Parameters:  
    /// - checkpoint: An ``MLCheckpoint`` instance from a hand pose  
    training  
    /// session.  
    public init          MLCheckpoint  throws  
  
    /// Begins an asynchronous hand pose classifier's training session.  
    ///  
    /// - Parameters:  
    /// - trainingData: An  
``MLHandPoseClassifier/DataSource`` instance.  
    ///  
    /// - parameters: An  
``MLHandPoseClassifier/ModelParameters-swift.struct``  
    /// instance you use to configure the model for the training session.  
    ///  
    /// - sessionParameters: An  
``MLTrainingSessionParameters`` instance you use  
    /// to configure the training session.  
    ///  
    /// - Returns: An ``MLJob`` that represents the hand pose classifier's  
    /// training session.  
    public static func train  
MLHandPoseClassifier DataSource  
MLHandPoseClassifier ModelParameters  
                    MLTrainingSessionParameters  
                    throws  
MLJob MLHandPoseClassifier  
  
    /// Creates an asynchronous hand pose classifier's training session.  
    ///  
    /// - Parameters:  
    /// - trainingData: An  
``MLHandPoseClassifier/DataSource`` instance.  
    ///  
    /// - parameters: An  
``MLHandPoseClassifier/ModelParameters-swift.struct``  
    /// instance you use to configure the model for the training session.  
    ///  
    /// - sessionParameters: An  
``MLTrainingSessionParameters`` instance you use  
    /// to configure the training session.  
    ///  
    /// - Returns: An ``MLTrainingSession`` that represents the action  
    /// classifier training session.  
    public static func makeTrainingSession  
MLHandPoseClassifier DataSource  
MLHandPoseClassifier ModelParameters  
                    MLTrainingSessionParameters
```

```
    throws
MLTrainingSession MLHandPoseClassifier

    /// Recreates an asynchronous hand pose classifier's training session by
    /// restoring its saved state from the file system.
    ///
    /// - Parameters:
    ///   - sessionParameters: The same
``MLTrainingSessionParameters`` instance
        /// that created an existing training session.
    ///
    /// - Returns: An ``MLTrainingSession`` that represents the hand
pose
    /// classifier training session.
public static func
restoreTrainingSession
MLTrainingSessionParameters throws
MLTrainingSession MLHandPoseClassifier

    /// Begins or continues an asynchronous hand pose classifier's training
    /// session.
    ///
    /// - Parameters:
    ///   - session: An ``MLTrainingSession`` instance that
represents the
        /// training session.
    ///
    /// - Returns: An ``MLJob`` that represents the hand pose training
session.
public static func resume -
MLTrainingSession MLHandPoseClassifier throws
MLJob MLHandPoseClassifier

    /// Generates a hand pose prediction for an image.
    ///
    /// Each prediction consists of an array of tuples that pair a
    /// classification label with the model's confidence in that label.
    ///
    /// - Parameters:
    ///   - image : An image file URL.
    ///
    /// - Returns: An array of prediction tuples.
public func prediction URL throws
String Double

    /// Generates an array of hand pose predictions for each image in a URL
    /// array.
    ///
    /// Each prediction consists of an array of tuples that pair a
    /// classification label with the model's confidence for that label. The
```

```

    /// method returns an array of prediction arrays, where each element of the
    /// outer array is the prediction for the corresponding URL element in
    /// `images`.
    ///
    /// - Parameters:
    ///   - images: An array of image file URLs.
    ///
    /// - Returns: An array of a prediction tuple arrays.
public func predictions URL throws  

    String Double

    /// Generates metrics that describe the hand pose classifier's performance
    /// with a dataset of labeled images.
    ///
    /// - Parameters:
    ///   - annotatedImages : An
``MLHandPoseClassifier/DataSource`` instance.
public func evaluation  

MLHandPoseClassifier DataSource throws MLClassifierMetrics

    /// Exports the hand pose classifier as a CoreML model file.
    ///
    /// - Parameters:
    ///   - fileURL: A file-system URL.
    ///   - metadata: The model's description, author, version, and license
information.
public func write URL  

MLModelMetadata nil throws

    /// Exports the hand pose classifier as a Core ML model file.
    ///
    /// - Parameters:
    ///   - path: A file-system path.
    ///   - metadata: The model's description, author, version, and license
information.
public func write String  

MLModelMetadata nil throws

@available macOS 12.0 iOS 15.0
@available
extension MLHandPoseClassifier CustomStringConvertible  

CustomDebugStringConvertible  

CustomPlaygroundDisplayConvertible

    /// A text representation of the hand pose classifier.
public var description String get

    /// A text representation of the hand pose classifier suitable for
    /// debugging.

```

```
public var debugDescription String get
    /// A description of the hand pose classifier that's viewable in a
    /// playground.
public var playgroundDescription Any get

@available macOS 12.0 iOS 15.0
@available
extension MLHandPoseClassifier

    /// A set of parameters that affect the training process of a hand pose
    /// classifier task.
public struct ModelParameters

    /// A dataset the hand pose classifier task uses to evaluate the model
    /// that's distinct from the training dataset.
    ///
    /// The task produces
``MLHandPoseClassifier/validationMetrics`` by
    /// evaluating the model with the
    /// ``MLHandPoseClassifier/ModelParameters-
swift.struct/validation``
    /// dataset.
    public var validation
MLHandPoseClassifier ModelParameters ValidationData

    /// The number of images the model training session uses for each
    /// training iteration.
    public var batchSize Int

    /// The largest number of training iterations you allow the training
    /// session to use.
    public var maximumIterations Int

    /// The variations the training session uses to add more variety to its
    /// training dataset.
    public var augmentationOptions
MLHandPoseClassifier ImageAugmentationOptions

    /// The algorithm the training session uses to create the hand pose
    /// classifier.
    public var algorithm
MLHandPoseClassifier ModelParameters ModelAlgorithmType

    /// Creates a set of training session parameters for a hand pose
    /// classifier task.
    ///
    /// - Parameters:
```

```

    /// - validation: An
    /// ``MLHandPoseClassifier/ModelParameters-
swift.struct/ValidationData``
    /// instance.
    ///
    /// - batchSize: The number of images the training session uses for
each
    /// of its training iterations.
    ///
    /// - maximumIterations: The largest number of training iterations the
    /// training session can use.
    ///
    /// - augmentationOptions: The variations the training session uses to
    /// add more variety to its training dataset.
    ///
    /// - algorithm: The algorithm the training session uses to train the
    /// hand pose classifier.
public init
MLHandPoseClassifier ModelParameters ValidationData
                    Int

Int

MLHandPoseClassifier ImageAugmentationOptions
MLHandPoseClassifier ModelParameters ModelAlgorithmType

    /// The hand pose classifier training algorithm options.
public enum ModelAlgorithmType : Equatable, Sendable

    /// Selects the graph convolutional neural-network algorithm for a
    /// hand pose classifier.
case gcn

    /// Hashes the essential components of this value by feeding
them into the
    /// given hasher.
    ///
    /// Implement this method to conform to the `Hashable`  

protocol. The
    /// components used for hashing must be the same as the
components compared
    /// in your type's `==` operator implementation. Call
`hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your implementation of `hash(into:)`,  

provided,
    /// don't call `finalize()` on the `hasher` instance
    /// or replace it with a different instance.

```

```

    /// Doing so may become a compile-time error in the future.
    ///
    /// - Parameter hasher: The hasher to use when combining
the components
    /// of this instance.
public func hash inout Hasher

    /// Returns a Boolean value indicating whether two values are
equal.
    ///
    /// Equality is the inverse of inequality. For any values `a` and
`b`,
    /// `a == b` implies that `a != b` is `false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to compare.
public static func
MLHandPoseClassifier ModelParameters ModelAlgorithmType
MLHandPoseClassifier ModelParameters ModelAlgorithmType
Bool

    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different
executions of
    /// your program. Do not save hash values to use during a future
execution.
    ///
    /// - Important: `hashValue` is deprecated as a
`Hashable` requirement. To
    /// conform to `Hashable`, implement the `hash(into:)` requirement instead.
    /// The compiler provides an implementation for `hashValue`
for you.
public var hashValue Int get

    /// A dataset a hand pose classifier task uses to validate the model
    /// during a training session.
public enum ValidationData

    /// Creates a validation dataset by randomly selecting a portion of
    /// the hand pose classifier task's training dataset with a split
    /// strategy.
    ///
    /// - Parameters:
    ///   - strategy: An ``MLSplitStrategy`` instance.
case split MLSplitStrategy

```

```
    /// Creates a validation dataset from a data source.  
    ///  
    /// - Parameters:  
    ///   - dataSource: An  
``MLHandPoseClassifier/DataSource`` instance.  
    case dataSource MLHandPoseClassifier DataSource  
  
    /// Creates an empty validation dataset, which tells the task to  
    /// skip the model validation phase in a training session.  
    ///  
    /// This case prevents Create ML from creating a validation  
dataset  
    /// from your training dataset. You typically use it when your  
    /// training dataset is small and can't spare any data to validate  
    /// the model.  
    case none
```

```
@available macOS 12.0 iOS 15.0  
@available  
extension MLHandPoseClassifier  
  
    /// A hand pose classifier dataset that contains annotated images or hand  
    /// joint location data.  
    public enum DataSource  
  
    /// Creates a data source from a folder that contains images and an  
    /// annotation file.  
    ///  
    /// - Parameters:  
    ///   - at: A URL to a folder in the file system that contains images.  
    ///  
    ///   - annotationFile: A URL to a JSON or CSV file that contains  
    ///     annotations for the hand pose images.  
    ///  
    ///   - imageColumn: The name of the column or key name in the  
annotation  
    ///     file that contains the image filenames.  
    ///  
    ///   - labelColumn: The name of the column or key name in the  
annotation  
    ///     file that contains the hand pose label names.  
    case directoryWithImagesAndAnnotation URL String String  
  
    /// Creates a data source from a folder with subfolders that each  
    /// contain images of a hand pose.  
    ///
```

```

    /// - Parameters:
    /// - at: The URL to a folder in the file system that contains folders
    /// of hand pose images. The data source uses the name of each folder
as
    /// the classification label for the hand pose images it contains.
case labeledDirectories URL

    /// Creates a data source from a folder that contains images, each
named
    /// after the hand pose it represents.
    ///
    /// Create a hand-pose data source from a directory of images with the
    /// `labeledFiles` case. You must name each image file with a
hand-pose
    /// label, followed by a period and an arbitrary string, ending with the
    /// image file's extension. For example, you can name a hand-pose
    /// classifier's training files as `Peace.3.png`, `ThumbsUp.1.jpg`,
    /// `ThumbsUp.2.jpg`, and so on.
    ///
    /// In this example, these image names give a hand-pose classifier at
    /// least two class labels:
    ///
    /// - `Peace` - `ThumbsUp`
    ///
/// - Parameters:
/// - at: The URL to a folder in the file system that contains folders
    /// of hand pose images. The data source uses each file's name before
    /// the first period as its classification label.
case labeledFiles URL

    /// Creates a data source from a data table of hand pose observations
    /// that each contain the locations of each hand joint and an
    /// annotation.
    ///
/// - Parameters:
/// - table : A data table that contains the hand-joint locations and
    /// annotations for a set of hand poses.
    ///
/// - sessionIdColumn: The name of the column in the data table that
    /// contains the session identifiers.
    ///
/// - labelColumn: The name of the column in the data table that
    /// contains the hand pose label names.
    ///
/// - featureColumn: The name of the column in the data table that
    /// contains the hand-joint location data. Each entry in the column must
    /// be an
    ///
<doc://com.apple.documentation/documentation/coreml/mlmultiarray>
    /// instance — which you must wrap in an

```

```

``MLDataValue/MultiArrayType``
    /// — that contains three dimensions:
    ///
    ///     — The first dimension has a size of one.
    ///     — The second dimension has three channels: the x-coordinate, the
    ///       y-coordinate, and the confidence value, respectively. - The third
    ///       dimension has 21 channels, one for each hand joint.
    @available           12.0          14.0
    @available           15.0          17.0
    @available
    case labeledKeypointsData      MLDataTable
        String
        String
        String

    /// Creates a data source from a data table that contains the hand joint
    /// locations and annotation for a set of image files.
    ///
    /// — Parameters:
    ///     — table: A data table that contains the locations of annotations
for
    ///   each hand pose image file.
    ///
    ///     — imageColumn: The name of the column in the data table that
    ///       contains the image filenames.
    ///
    ///     — labelColumn: The name of the column in the data table that
    ///       contains the hand pose label names.
    @available           12.0          14.0
    @available           15.0          17.0
    @available
    case labeledImageData      MLDataTable
String                               String

    /// Creates a data source from a data frame of hand pose observations
    /// that each contain the locations of each hand joint and an
    /// annotation.
    ///
    /// — Parameters:
    ///     — dataFrame: A data frame that contains the hand-joint locations
and
    ///   annotations for a set of hand poses.
    ///
    ///     — sessionIdColumn: The name of the column in the data frame that
    ///       contains the hand pose session identifiers.
    ///
    ///     — labelColumn: The name of the column in the data frame that
    ///       contains the hand pose label names.
    ///

```

```

    /// - featureColumn: The name of the column in the data frame that
    /// contains the hand-joint location data. Each entry in the column must
    /// be a
    ///
<doc://com.apple.documentation/documentation/tabulardata/shapeddata>
    /// instance that contains three dimensions:
    ///
    /// - The first dimension has a size of one.
    /// - The second dimension has three channels: the x-coordinate, the
    /// y-coordinate, and the confidence value, respectively. - The third
    /// dimension has 21 channels, one for each hand joint.
case labeledKeypointsDataFrame DataFrame
    String
    String
    String

    /// Creates a data source from a data frame that contains the location
    /// and annotation for a set of image files.
    ///
    /// - Parameters:
    /// - dataFrame: A data frame that contains the locations of
annotations
    /// for each hand pose image file.
    ///
    /// - imageColumn: The name of the column in the data frame that
    /// contains the image filenames.
    ///
    /// - labelColumn: The name of the column in the data frame that
    /// contains the hand pose label names.
case labeledImageDataFrame DataFrame
String                                         String

    /// Processes the data source and returns a data frame that contains
file URLs and annotations.
    ///
    /// This method collects file names from the filesystem if necessary. If
the data source is already in table
    /// format it renames the columns to the default column names. This
method returns nil if the data source
    /// contains key points.
available macOS 14.0 iOS 17.0
available
public func gatherAnnotatedFileNames throws
DataFrame

    /// Generates a data table that contains a column for the data source's
    /// image file URLs and a column of annotations.
available 12.0 14.0
    "Use gatherAnnotatedFileNames()"

```

```

@available 15.0 17.0
    "Use gatherAnnotatedFileNames()"
@available
public func imagesWithAnnotations throws
MLDataTable

    /// Extracts key points from video files if necessary.
    ///
    /// If the data source already contains keypoints, this method just
    renames the data frame columns to the
    /// defaults.
    ///
    /// – Returns: A data frame that contains a column for hand joint
    locations and a column of hand action
    /// annotations.
@available macOS 14.0 iOS 17.0
@available
public func extractKeypoints throws DataFrame

    /// Generates a data table that contains a column for hand joint
    /// locations and hand pose annotations.
@available 12.0 14.0
    "Use extractKeypoints()"
@available 15.0 17.0
    "Use extractKeypoints()"
@available
public func keypointsWithAnnotations throws
MLDataTable

    /// Generates a data table by splitting the data source into strata.
    ///
    /// – Parameters:
    /// – proportions: An array of stratum proportions, each in the
range
    /// ` [0.0, 1.0]`.
    ///
    /// – seed: A seed number for the random-number generator.
    ///
    /// – labelColumn: The name of the column or category the method
uses to
    /// stratify the contents of the data source.
@available 12.0 14.0
    "Use DataFrame.stratifiedSplit(on:by:)"
@available 15.0 17.0
    "Use DataFrame.stratifiedSplit(on:by:)"
@available
public func stratifiedSplit throws
Int String Double
MLDataTable

```

```
@available macOS 12.0 iOS 15.0
@available
extension MLHandPoseClassifier

    /// Options a hand pose classification training session can use to generate
    /// additional training data from the images you provide.
    public struct ImageAugmentationOptions : OptionSet
Codable Sendable

    /// The corresponding value of the raw type.
    ///
    /// A new instance initialized with `rawValue` will be equivalent to this
    /// instance. For example:
    ///
    ///     enum PaperSize: String {
    ///         case A4, A5, Letter, Legal
    ///     }
    ///
    ///     let selectedSize = PaperSize.Letter
    ///     print(selectedSize.rawValue)
    ///     // Prints "Letter"
    ///
    ///     print(selectedSize == PaperSize(rawValue:
selectedSize.rawValue)!)
    ///     // Prints "true"
public let rawValue : Int

    /// Apply left-right flips to the pose in an image.
public static let horizontallyFlip
MLHandPoseClassifier ImageAugmentationOptions

    /// Randomly rotate the pose in an image.
public static let rotate
MLHandPoseClassifier ImageAugmentationOptions

    /// Randomly translate the pose in an image.
public static let translate
MLHandPoseClassifier ImageAugmentationOptions

    /// Randomly scale the pose in an image.
public static let scale
MLHandPoseClassifier ImageAugmentationOptions

    /// Creates an option set from an integer.
    ///
    /// - Parameters:
    /// - rawValue: An integer that represents the option set's
```

```

underlying
    /// raw value.
    public init Int

    /// The type of the elements of an array literal.
    @available iOS 15.0 macOS 12.0
    @available
    public typealias ArrayLiteralElement
MLHandPoseClassifier ImageAugmentationOptions

    /// The element type of the option set.
    ///
    /// To inherit all the default implementations from the `OptionSet` protocol,
    /// the `Element` type must be `Self`, the default.
    @available iOS 15.0 macOS 12.0
    @available
    public typealias Element
MLHandPoseClassifier ImageAugmentationOptions

    /// The raw type that can be used to represent all values of the conforming
unique
    /// type.
    ///
    /// Every distinct value of the conforming type has a corresponding
    /// value of the `RawValue` type, but there may be values of the
`RawValue` type that don't have a corresponding value of the conforming type.
    @available iOS 15.0 macOS 12.0
    @available
    public typealias RawValue Int

```

```

@available macOS 12.0 iOS 15.0
@available
extension MLHandPoseClassifier ModelParameters
CustomStringConvertible CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of the hand pose parameters.
    public var description String get

    /// A text representation of the hand pose parameters suitable for
    /// debugging.
    public var debugDescription String get

    /// A description of the hand pose parameters that's viewable in a
    /// playground.

```

```
public var playgroundDescription Any get

@available macOS 12.0 iOS 15.0
@available
extension MLHandPoseClassifier DataSource

/// Generates a dictionary that contains the data source's classification
/// labels paired with an array of URLs to the label's image files.
public func labeledMedia throws String URL

@available macOS 12.0 iOS 15.0
@available
extension MLHandPoseClassifier ModelParameters ModelAlgorithmType
Hashable

/// A type the Create ML framework can use as a machine learning identifier.
///
/// You can use any type that conforms to the ``MLIdentifier`` protocol,
/// typically <doc://com.apple.documentation/documentation/swift/int> or
/// <doc://com.apple.documentation/documentation/swift/string>, to uniquely
/// identify users and items in these ``MLRecommender`` methods:
///
/// -
///
``MLRecommender/recommendations(fromUsers:maxCount:restricting
ToItems:excluding:excludingObserved:)``-7an46``-
/// - ``MLRecommender/getSimilarItems(fromItems:maxCount:)``-kq37``
@available 10.15 14.0
@available
@available
public protocol MLIdentifier

/// The value of the unique identifier wrapped in a data value.
var identifierValue MLDataValue get

/// A model you train to classify images.
///
/// Use an image classifier to train a machine learning model that you can include
/// in your app to categorize images.
///
/// When you create the model, you give it a training dataset made up of labeled
/// images, along with parameters that
/// control the training process. For example, you can provide the model with
/// images of elephants and giraffes, in
```

```
/// two folders labeled `Elephant` and `Giraffe`, to train it to recognize these
animals.
///
/// After training completes, you evaluate the trained model by showing it a testing
dataset containing labeled images
/// that the model hasn't seen before. The metrics that come from this evaluation
tell you whether the model performs
/// well enough. For example, you can see how often the elephant and giraffe
classifier mistakes a giraffe for an
/// elephant. When the model makes too many mistakes, you can add more or
better training data, or change the
/// parameters, and try again.
///
/// When your model does perform well enough, you save it as a Core ML model
file with the `mlmodel` extension. You can
/// then import this model file into an app—like the
///
<doc://com.apple.documentation/documentation/vision/classifying_images_with_visi
on_and_core_ml> sample code
/// project—that uses a Core ML model file to classify images.
@available macOS 10.14 iOS 15.0
@available
public struct MLImageClassifier @unchecked Sendable

    /// The underlying Core ML model of the image classifier stored in memory.
    public var model MLMModel

    /// The model configuration parameters the image classifier used during its
    training session.
    public let modelParameters
    MLImageClassifier ModelParameters

    /// Measurements of the classifier's performance on the training data set.
    public var trainingMetrics MLClassifierMetrics get

    /// Measurements of the image classifier's performance on the validation
    dataset.
    public var validationMetrics MLClassifierMetrics get

    /// Creates an image classifier with a training dataset represented by a data
    source.
    ///
    /// When you create an ``MLImageClassifier`` instance, initialize it
    with an
    /// ``MLImageClassifier/ModelParameters-swift.struct`` structure. This allows you to configure the image classifier
    /// training process. For example, you can explicitly define the validation
    dataset instead of allowing the model
    /// to choose a random selection of your training data. Alternatively, as
    shown in the following example, set
    /// `validationData` to `nil` to allow the classifier to choose the
```

```
validation data for you from among your
    /// training data. This lets you set other parameters—like maximum iterations
and augmentation options—to values
    /// other than the default.
    ///
    /// ````swift
    /// let parameters = MLImageClassifier.ModelParameters(
    ///     featureExtractor: .scenePrint(revision: 1),
    ///     validationData: nil,
    ///     maxIterations: 20,
    ///     augmentationOptions: [.crop]
    /// )
    /// ``
    ///
    /// Use the parameter structure and your training data to build a classifier.
The following example uses training
    /// data from labeled directories within a directory called `Training`, which
resides in the `Downloads` directory:
    ///
    /// ````swift
    /// if let downloads =
FileManager.default.urls(for: .downloadsDirectory,
in: .userDomainMask).first {
    ///     let trainingURL =
downloads.appendingPathComponent("Training")
    ///     let classifier = try MLImageClassifier(
    ///         trainingData: .labeledDirectories(at:
trainingURL),
    ///         parameters: parameters
    ///     )
    /// }
    /// ``
    ///
    /// Training begins immediately.
    ///
    /// - Note: If you represent your training data with a dictionary of strings
and corresponding URL arrays, use
    ///
````MLImageClassifier/init(trainingData:parameters:)`-7j4w6```
instead.
    ///
    /// - Parameters:
    ///   - trainingData: A set of labeled images the task uses to train the
image classifier model, contained in a
    ///   data source.
    ///   - parameters: An ``MLImageClassifier/ModelParameters-
swift.struct`` instance you use to configure the model
    ///   for the training session.
@available macOS 11.0 iOS 15.0
@available
```

```
public init MLImageClassifier DataSource  
MLImageClassifier ModelParameters
```

1

### **throws**

```
/// Creates an image classifier with a training dataset represented by a  
dictionary.  
///  
/// When you create an ``MLImageClassifier`` instance, initialize it  
with an  
/// ``MLImageClassifier/ModelParameters-swift.struct``  
structure. This allows you to configure the image classifier  
/// training process. For example, you can explicitly define the validation  
dataset instead of allowing the model  
/// to choose a random selection of your training data. Alternatively, as  
shown in the following example, set  
/// `validationData` to `nil` to allow the classifier to choose the  
validation data for you from among your  
/// training data. This lets you set other parameters—like maximum iterations  
and augmentation options—to values  
/// other than the default:  
///  
/// ````swift  
/// let parameters = MLImageClassifier.ModelParameters(  
///     featureExtractor: .scenePrint(revision: 1),  
///     validationData: nil,  
///     maxIterations: 20,  
///     augmentationOptions: [.crop]  
/// )  
/// ````  
///  
/// For this particular initialization method—there's another with the same  
signature but a different training  
/// data type—represent your training data with a dictionary that uses labels  
as keys. The corresponding values are  
/// arrays of URLs that indicate the images associated with that label. In this  
example, if you have elephant and  
/// giraffe images stored in a directory called `Training` within your  
`Downloads` directory, you can construct a  
/// dictionary with the URL of each image:  
///  
/// ````swift  
/// // Get the URL of the directory that holds the  
validation data.  
/// guard let downloadsURL = FileManager.default.urls(for:
```

```

.downloadsDirectory, in: .userDomainMask).first
    /// else { fatalError("Can't find Downloads directory") }
    /// let url =
downloadsURL.appendingPathComponent("Training")
    ///
    /// // For a real classifier, use at least 10 images per
label. More is better.
    /// let trainingData = [
    ///     "Elephant": [
    ///         url.appendingPathComponent("Elephant.1.jpg"),
    ///         url.appendingPathComponent("Elephant.2.jpg")
    ///     ],
    ///     "Giraffe": [
    ///         url.appendingPathComponent("Giraffe.1.jpg"),
    ///         url.appendingPathComponent("Giraffe.2.jpg")
    ///     ]
    /// ]
    ///
    /// - Note: If you have training data in an
``MLImageClassifier/DataSource`` instance, use the similarly named
    ///
``MLImageClassifier/init(trainingData:parameters:)`` instead. That initialization method takes a data
        /// source instead of a dictionary as its training data.
    ///
    /// Use the parameter structure and training data to initialize the classifier.
Training begins immediately.
    ///
    /// ``swift
    /// let classifier = try MLImageClassifier(trainingData:
trainingData, parameters: parameters)
    ///
    ///
    /// - Parameters:
    ///     - trainingData: A set of labeled images the task uses to train the
image classifier model, contained in a
        ///     dictionary whose keys are the image labels. Each dictionary value is
an array of URLs to images.
    ///     - parameters: An ``MLImageClassifier/ModelParameters-
swift.struct`` instance you use to configure the model
        ///     for the training session.
    @available               10.14          11.0
        "Use DataSource.filesByLabel to provide dictionary
training data instead."
    @available               15.0           16.0
        "Use DataSource.filesByLabel to provide dictionary
training data instead."
    @available
    public init             String        URL

```

```
MLImageClassifier ModelParameters  
nil throws
```

```
extension MLImageClassifier
```

```
    /// The underlying base model that extracts image features for image  
    classifier training session.  
    ///  
    /// Create ML has one built-in feature extractor: `scenePrint`.  
    Alternatively, you can provide your own custom  
    /// feature extractor from an `mlmodel` file or a specific layer within a  
    model file.  
    @available macOS 10.14 iOS 15.0  
    @available  
    public enum FeatureExtractorType : Sendable  
  
        /// A feature extractor trained on millions of images.  
        /// - Parameters:  
        ///     - revision: the sceneprint version. The supported versions  
        include 1 and 2. If `nil` defaults to the latest version.  
        ///  
        /// - Note: The case's associated value indicates which revision of  
        the feature extractor to use. Set this to  
        ///     1 when creating a scene print feature extractor.  
        ///  
        /// The scene print feature extractor works best with images of real  
        world objects because it trained on  
        /// millions of such images. Scene print is not suitable for character  
        recognition, because the input images  
        /// are highly binary in nature (pixels are either on or off).  
        ///  
        /// When you train an image classifier using scene print, or make  
        predictions with the resulting model, use  
        /// images with 299x299 pixels or more. The model upscales smaller  
        images, to 299x299 before it feeds them to  
        /// the feature extractor, which may result in poor accuracy.  
        ///  
        /// Typically, scene print works best if the source of your training data  
        matches the source of the images you  
        /// want to classify. For example, if your app classifies images captured  
        with an iPhone camera, train your  
        /// model using images captured in the same way, if possible.  
        case scenePrint : Int  
  
        /// A feature extractor that you provide as a Core ML model file or a  
        layer within that file.  
        /// - Parameters:  
        ///     - _: custom feature extractor to be used for extracting features.  
        ///  
        /// A custom feature extractor is a neural network model you provide to
```

an ``MLImageClassifier``. The feature  
    /// extractor can be a layer within the model or the model itself. In either  
case, the neural network must take  
    /// an image as input and output an  
`<doc://com.apple.documentation/documentation/coreml/mlmultiarray>`.  
    **@available macOS 10.15 iOS 15.0**  
    **@available**  
    **case** custom MLImageClassifier CustomFeatureExtractor

    /// A custom feature extractor a training session uses to train an image  
classifier.  
    **@available macOS 10.15 iOS 15.0**  
    **@available**  
    **public struct** CustomFeatureExtractor Sendable

        /// The location of a neural network ` `.mlmodel` file that takes an  
image as an input.  
        **public var** modelPath URL

        /// The name of the output from a feature extraction layer within the  
model.  
        ///  
        /// When training an image classifier with a custom feature extractor,  
use  
        ///  
        ``MLImageClassifier/CustomFeatureExtractor/outputName`` to select  
a layer within the model that has an  
        /// output of  
`<doc://com.apple.documentation/documentation/coreml/mlmultiarray>`.  
        ///  
        /// Set  
        ``MLImageClassifier/CustomFeatureExtractor/outputName`` to ` nil`  
if the model (specified by  
        ///  
        ``MLImageClassifier/CustomFeatureExtractor/modelPath``) has only  
one output of type  
        ///  
`<doc://com.apple.documentation/documentation/coreml/mlmultiarray>`.  
        **public var** outputName String

        /// Creates a custom feature extractor given a model file and an  
optional output layer name.  
        ///  
        /// - Parameters:  
        ///     - modelPath: The location of the neural network ` `.mlmodel`  
which contains the feature extractor.  
        ///     - outputName: The name of a feature extraction layer within  
the model has one output type of  
        ///  
`<doc://com.apple.documentation/documentation/coreml/mlmultiarray>`. Set this value

```

to `nil` if the
    /// model, as a whole, accepts an input image and has exactly
one output of type
    ///
<doc://com.apple.documentation/documentation/coreml/mlmultiarray>
public init URL String nil

@available macOS 10.14 iOS 15.0
available
extension MLImageClassifier

    /// Parameters that affect the process of training an image classifier model.
    ///
    /// Use this structure to configure the model training session. With it you can:
    /// – Set a limit to the number of training iterations the session can use
    /// – Provide your own validation dataset. See
``MLImageClassifier/ModelParameters-swift.struct/ValidationDat
a-swift.enum``.
    /// – Enable specific image augmentations. See
``MLImageClassifier/ImageAugmentationOptions``.
    /// – Designate a custom feature extractor. See
``MLImageClassifier/FeatureExtractorType/custom(_:)``.
    ///
    /// Once you configure an ``MLImageClassifier/ModelParameters-
swift.struct`` instance, use it to configure a
    /// training session with one of the applicable ``MLImageClassifier`` asynchronous type methods or synchronous
    /// initializers.
public struct ModelParameters

    /// The maximum number of iterations the training session can use.
public var maxIterations Int

    /// The variations the training session uses to generate more variety in
the training dataset.
public var augmentationOptions
MLImageClassifier ImageAugmentationOptions

    /// Model algorithm to be used
available macOS 11.0 iOS 15.0
available
public var algorithm
MLImageClassifier ModelParameters ModelAlgorithmType

    /// A set of images that the training process uses for validation.
available 10.14
10.15 "Use the validation property instead."
available 15.0 16.0

```

```
"Use the validation property instead."
@available
public var validationData    String    URL

/// The image classifier's validation dataset.
@available macOS 10.15 iOS 15.0
@available
public var validation
MLImageClassifier ModelParameters ValidationData

/// The underlying base model the training session uses to extract
image features as it trains an image
/// classifier.
///
/// When you train an image classifier with Create ML, you don't train a
complete model that converts images to
/// labels. Instead, Create ML leverages a _feature extractor_, which is
another model that identifies a set of
/// general but distinguishing image characteristics. You can either use
Create ML's `scenePrint` feature
/// extractor or provide your own custom feature extractor.
///
/// In either case, you train your model to map the feature extractor's
output to labels in a process known as
/// _transfer learning_. Your model effectively borrows the knowledge of
the feature extractor to accelerate
/// its own training process while requiring fewer training images.
@available 10.14 11.0
"Use featureExtractor in ModelAlgorithmType
instead."
@available 15.0 16.0
"Use featureExtractor in ModelAlgorithmType
instead."
@available
public var featureExtractor
MLImageClassifier FeatureExtractorType

/// Creates model training parameters.
///
/// - Parameters:
///   - validation: Labeled data that the model evaluates on for
validation. The default is
///     `split(strategy: .automatic)`.
///   - maxIterations: The maximum number of training
iterations to use during training. The default is 25.
///   - augmentation: The image augmentation options to use to
increase the training data variety. If no data
///     augmentation needs to be applied, use `[]` as input.
Otherwise, inputs take the form `[.crop, .blur]`.
///   - algorithm: The type of model algorithm to use for training.
The default is a logistic regression
```

```
    /// classifier with a `sceneprint(revision: 1)` feature
extractor.
    @available macOS 11.0 iOS 15.0
    @available
    public init
MLImageClassifier ModelParameters ValidationData
    Int

MLImageClassifier ImageAugmentationOptions
MLImageClassifier ModelParameters ModelAlgorithmType

    /// Creates a new set of training parameters for an image classifier with
a validation dataset.
    ///
    /// - Parameters:
    ///   - featureExtractor: The feature extractor you want Create
ML to use during the training session.
    ///   - validation: A validation dataset.
    ///   - maxIterations: The maximum number of training
iterations to use during training. The default is 25.
    ///   - augmentationOptions: The image augmentation options
to use to increase the training data variety.
    @available 10.15 11.0
    "Use featureExtractor in ModelAlgorithm Type
instead."
    @available 15.0 16.0
    "Use featureExtractor in ModelAlgorithm Type
instead."
    @available
    public init
MLImageClassifier FeatureExtractorType
1
MLImageClassifier ModelParameters ValidationData
    Int 25

MLImageClassifier ImageAugmentationOptions

    /// Creates a new set of image classifier parameters with validation
data represented by a dictionary.
    ///
    /// - Parameters:
    ///   - featureExtractor: A versioned feature extractor.
    ///   - validationData: Validation data represented by a
dictionary.
    ///   - maxIterations: The maximum number of training
iterations to use during training. The default is 25.
    ///   - augmentationOptions: The image augmentation options
to use to increase the training data variety.
    @available 10.14
    "Use the validation property instead."
10.15
```

```

@available 15.0 16.0
    "Use the validation property instead."
@available
public init
MLImageClassifier FeatureExtractorType
1           String   URL           Int
25
MLImageClassifier ImageAugmentationOptions

    /// Creates a new set of image classifier parameters with validation
data represented by a data source.
    ///
    /// - Parameters:
    ///   - featureExtractor: A versioned feature extractor.
    ///   - validationData: The validation datasource.
    ///   - maxIterations: The maximum number of training
iterations to use during training. The default is 25.
    ///   - augmentationOptions: The image augmentation options
to use to increase the training data variety.
@available 10.14
10.15      "Use the validation property instead."
@available 15.0 16.0
    "Use the validation property instead."
@available
public init
MLImageClassifier FeatureExtractorType
1           MLImageClassifier DataSource
           Int     25
MLImageClassifier ImageAugmentationOptions

```

## **extension** MLImageClassifier

```

    /// Exports the image classifier as a Core ML model file to a location in the
file system.
    ///
    /// - Parameters:
    ///   - fileURL: The location URL in the file system where you want to
save the model.
    ///   - metadata: Descriptive information to include with the exported
model file.
@available macOS 10.14 iOS 15.0
@available
public func write           URL
MLModelMetadata   nil  throws

    /// Exports the image classifier as a Core ML model file to the file path.
    ///
    /// - Parameters:

```

```
    /// - path: The location path in the file system where you want to save  
the model.  
    /// - metadata: Descriptive information to include with the exported  
model file.  
    @available macOS 10.14 iOS 15.0  
    @available  
    public func write String  
MLModelMetadata nil throws  
  
@available macOS 10.14 iOS 15.0  
@available  
extension MLImageClassifier  
  
    /// The variations that the training process can use to generate more training  
data from the training data you  
    /// provide.  
    ///  
    /// Augmentation generates new images from the training data you supply to  
increase the amount of training data  
    /// available to the model.  
    ///  
    /// See <doc:improving-your-model-s-accuracy> for a discussion about when  
to use augmentation.  
    public struct ImageAugmentationOptions OptionSet  
Sendable  
  
    /// The corresponding value of the raw type.  
    ///  
    /// A new instance initialized with `rawValue` will be equivalent to this  
/// instance. For example:  
    ///  
    ///     enum PaperSize: String {  
    ///         case A4, A5, Letter, Legal  
    ///     }  
    ///  
    ///     let selectedSize = PaperSize.Letter  
    ///     print(selectedSize.rawValue)  
    ///     // Prints "Letter"  
    ///  
    ///     print(selectedSize == PaperSize(rawValue:  
selectedSize.rawValue))  
    ///     // Prints "true"  
    public let rawValue Int  
  
    /// An option for augmenting training data by blurring each image.  
    ///  
    /// Use this option to tell the image classifier to augment your training  
data set by creating blurred versions  
    /// of the original images.
```

```
///  
/// ! [Diagram showing how the original image results in four blurred  
variants.] (MLImageClassifier–ImageAugmentationOptions–blur–1)  
///  
/// The classifier creates four new images with random amounts of blur  
for each original.  
public static let blur  
MLImageClassifier ImageAugmentationOptions  
  
    /// An option for augmenting training data by flipping each image along  
the horizontal and vertical axes.  
    ///  
    /// Use this option to tell the image classifier to augment your training  
data set by flipping the original  
    /// images.  
    ///  
    /// ! [Diagram showing how the original image results in three flipped  
variants.] (MLImageClassifier–ImageAugmentationOptions–flip–1)  
    ///  
    /// The classifier creates three new images by flipping the original  
horizontally, vertically, and both  
    /// horizontally and vertically.  
public static let flip  
MLImageClassifier ImageAugmentationOptions  
  
    /// An option for augmenting training data by lightening or darkening  
each image.  
    ///  
    /// Use this option to tell the image classifier to augment your training  
data set by creating darker or  
    /// lighter versions of the original images.  
    ///  
    /// ! [Diagram showing how the original image results in four variants  
with different exposures.] (MLImageClassifier–  
ImageAugmentationOptions–exposure–1)  
    ///  
    /// The classifier creates four new images with random exposure  
variations for each original.  
public static let exposure  
MLImageClassifier ImageAugmentationOptions  
  
    /// An option for augmenting training data by adding random amounts  
of noise to each image.  
    ///  
    /// Use this option to tell the image classifier to augment your training  
data set by adding noise to the  
    /// original images.  
    ///  
    /// ! [Diagram showing how the original image results in four variants  
with random amounts of noise added.] (MLImageClassifier–  
ImageAugmentationOptions–noise–1)
```

```
///  
/// The classifier creates four new images with random amounts of  
noise for each original.  
public static let noise  
MLImageClassifier ImageAugmentationOptions  
  
    /// An option for augmenting training data by rotating each image.  
    ///  
    /// Use this option to tell the image classifier to augment your training  
data set by rotating the original  
    /// images.  
    ///  
    /// ! [Diagram showing how the original image results in four rotated  
variants.] (MLImageClassifier–ImageAugmentationOptions–rotation–1)  
    ///  
    /// The classifier creates four new images with random rotation angles  
for each original.  
public static let rotation  
MLImageClassifier ImageAugmentationOptions  
  
    /// An option for augmenting training data by creating cropped versions  
of each image.  
    ///  
    /// Use this option to tell the image classifier to augment your training  
data set by creating cropped versions  
    /// of the original images.  
    ///  
    /// ! [Diagram showing how the original image results in four cropped  
variants.] (MLImageClassifier–ImageAugmentationOptions–crop–1)  
    ///  
    /// The classifier creates four new images with random amounts of crop  
for each original.  
public static let crop  
MLImageClassifier ImageAugmentationOptions  
  
    /// Creates an augmentation set with the given raw value.  
public init Int  
  
    /// The type of the elements of an array literal.  
@available iOS 15.0 macOS 10.14  
@available  
    public typealias ArrayLiteralElement  
MLImageClassifier ImageAugmentationOptions  
  
    /// The element type of the option set.  
    ///  
    /// To inherit all the default implementations from the `OptionSet`  
protocol,  
    /// the `Element` type must be `Self`, the default.  
@available iOS 15.0 macOS 10.14
```

```
@available
public typealias Element
MLImageClassifier ImageAugmentationOptions

    /// The raw type that can be used to represent all values of the
conforming
    /// type.
    ///
    /// Every distinct value of the conforming type has a corresponding
unique
    /// value of the `RawValue` type, but there may be values of the
`RawValue`
    /// type that don't have a corresponding value of the conforming type.
@available iOS 15.0 macOS 10.14
@available
public typealias RawValue Int
```

```
@available macOS 10.14 iOS 15.0
@available
extension MLImageClassifier CustomStringConvertible

    /// A text representation of the image classifier.
public var description String get

@available macOS 10.14 iOS 15.0
@available
extension MLImageClassifier CustomDebugStringConvertible

    /// A text representation of the image classifier that's suitable for output
    /// during debugging.
public var debugDescription String get

@available macOS 10.14 iOS 15.0
@available
extension MLImageClassifier
CustomPlaygroundDisplayConvertible

    /// A description of the image classifier shown in a playground.
public var playgroundDescription Any get
```

```
extension MLImageClassifier

    /// Generates metrics describing the image classifier's performance on
labeled images represented by a data source.
    ///
```

```

    /// - Parameter labeledImages: A set of labeled images in a data
source.
    /// - Returns: The metrics that indicate the performance of the classifier
when operating on the input dataset.
    @available macOS 10.14 iOS 15.0
    @available
    public func evaluation
MLImageClassifier DataSource MLClassifierMetrics

    /// Generates metrics describing the image classifier's performance on
labeled images represented by a dictionary.
    ///
    /// - Parameter labeledImages: A dictionary with keys that are labels,
and values that are arrays of image URLs
    /// corresponding to those labels.
    /// - Returns: The metrics that indicate the performance of the classifier
when operating on the input dataset.
    @available 10.14 11.0
    "Use DataSource.filesByLabel to provide dictionary
training data instead."
    @available 15.0 16.0
    "Use DataSource.filesByLabel to provide dictionary
training data instead."
    @available
    public func evaluation String URL
MLClassifierMetrics

```

## `extension MLImageClassifier`

```

    /// Generates a prediction for an image.
    ///
    /// - Parameter image: The image that you want the model to classify.
    /// - Returns: A prediction label for the image.
    @available macOS 10.14 iOS 15.0
    @available
    public func prediction CGImage throws
String

    /// Generates a prediction for an image at the URL.
    ///
    /// - Parameter image: The URL of the image you want the model to
classify.
    /// - Returns: A prediction label for the image.
    @available macOS 10.14 iOS 15.0
    @available
    public func prediction URL throws String

    /// Generates predictions for an array of images.
    ///

```

```
    /// - Parameter images: An array of images you want the model to
    // classify.
    /// - Returns: An array of prediction labels for the images. Each label's
    // index corresponds to the image's index in
    /// the input array.
    @available(macOS 10.14, iOS 15.0)
    @available
    public func predictions(url: URL) throws
    String

@available(macOS 10.14, iOS 15.0)
@available
extension MLImageClassifier

    /// A data source for an image classifier.
    ///
    /// Use a data source to provide training or testing data to an image
    // classifier.
    ///
    /// To train a model programmatically with an ``MLImageClassifier`` instance,
    // initialize a data source with the URL
    /// of the directory that contains the data. Use either
    ``MLImageClassifier/DataSource/labeledDirectories(at:)`` or
    /// ``MLImageClassifier/DataSource/labeledFiles(at:)`` to do
    // this, depending on whether your images are grouped by
    /// directory or by file name. See the respective creation methods for details
    // about how to arrange your image
    /// files in each case.
    ///
    /// When you train a model using ``MLImageClassifierBuilder``, you
    // don't initialize a data source directly. Instead,
    /// you drag the directory containing your data from a Finder window into the
    // live view. The builder automatically
    /// chooses the correct kind of data source based on how your images are
    // arranged inside that directory, looking
    /// for either labeled directories or labeled files.
    public enum DataSource : Sendable

        /// An image classifier data source that uses the directory structure to
        // label images.
        ///
        /// Use this case to create a data source based on a directory structure
        // on disk that contains your training
        /// images. The associated value is a URL that represents a directory
        // with subdirectories that are named as the
        /// labels for your images. Each subdirectory should contain the images
        // associated with the label for which
        /// that subdirectory is named.
        ///
        /// For example, you might initialize a data source with a file URL that
```

indicates a directory called  
    /// `Training Data` that contains subdirectories called  
    `Elephant` and `Cheetah`. The subdirectory `Cheetah`,  
        /// in turn, contains images of cheetahs, with names like  
    `Image001.png`, `Image02.jpg`, and so on, while the  
        /// subdirectory `Elephant` contains images of elephants, called  
    `Image545.jpg`, `Image657.png`, and so on. The  
        /// individual image names aren't important in this case because the  
    directory that an image appears in  
        /// indicates the image's label.  
    ///  
        /// ! [Diagram showing a directory called "Training Data" containing  
    subdirectories, that are named using the  
        /// label that applies to all the images inside that subdirectory. For  
    example, the Cheetah subdirectory  
        /// contains images of cheetahs, but the individual file names aren't  
    important.] (MLImageClassifier-DataSource-labeledDirectories(at):-1)  
    ///  
        /// If you want to create a data source with labels based on file names,  
use  
        /// ``MLImageClassifier/DataSource/labeledFiles(at:)``  
instead.  
**case** **labeledDirectories**       **URL**  
  
    /// An image classifier data source that uses file names to label images.  
    ///  
        /// Use this case to create a data source based on files on disk that  
    contain your training images. The  
        /// associated value is a URL that represents a directory containing files  
    with names based on each image's  
        /// associated label. The image's label is taken as the part of the file  
    name up to the first period.  
    ///  
        /// For example, you can initialize a data source with a file URL that  
    indicates a directory called `Training  
        /// Data` that contains the files `buffalo.1.jpg`,  
    `buffalo.2.jpg`, `cheetah.1.jpg`, and `cheetah.2.jpg`. The  
        /// first two images receive the label `buffalo`, while the second two  
    images receive the label `cheetah`. The  
        /// part of the name after the first period can be any arbitrary string.  
    ///  
        /// ! [Diagram showing a directory called "Training Data" containing  
    images with names that use the label in the  
        /// part of the name that comes before the first period. For example, all  
    the images of a buffalo start with  
        /// the word "buffalo".] (MLImageClassifier-DataSource-  
    labeledFiles(at):-1)  
    ///  
        /// If you want to create a data source with labels based on how you  
    sort your images into directories, use  
    ///

```

``MLImageClassifier/DataSource/labeledDirectories(at:)`` instead.
case labeledFiles URL

    /// Dictionary of labels to file URLs.
@available macOS 11.0 iOS 15.0
@available
case filesByLabel String URL

    /// Returns the labeled images represented by the data source.
    ///
    /// - Returns: The labeled images that the data source represents,
    as a dictionary. The dictionary keys are the
        /// labels, while the dictionary values are arrays of images,
    represented as URLs, that correspond to the
        /// label.
public func labeledImages throws String URL

    /// Generates an array of labeled image dictionaries by splitting the
data source into strata.
    ///
    /// - Parameters:
    ///   - proportions: An array of proportions, each in the range
`[0.0, 1.0]`.
    ///   - seed: A seed number for the random-number generator.
    /// - Returns: An array of dictionaries of labeled images.
public func stratifiedSplit Double
Int throws String URL

    /// Generates an array of labeled image dictionaries by splitting the
data source into strata using the
    /// random-number generator.
    ///
    /// - Parameters:
    ///   - proportions: An array of proportions, each in the range
`[0.0, 1.0]`.
    ///   - generator: A random-number generator.
    /// - Returns: An array of dictionaries of labeled images.
public func stratifiedSplit RNG
Double inout RNG throws String URL
where RNG RandomNumberGenerator

```

## **extension** MLImageClassifier

```

    /// Creates an image classifier from a training session checkpoint.
    ///
    /// You can only use a checkpoint from an image classifier's training session
if its ``MLCheckpoint/phase``
    /// property is ``MLPhase/training``.

```

```
///  
/// - Parameter checkpoint: A checkpoint from an image classifier  
training session.  
@available macOS 11.0 iOS 15.0  
@available  
public init          MLCheckpoint throws  
  
/// Begins an asynchronous image classifier training session with a training  
dataset represented by a data source.  
///  
/// - Parameters:  
///   - trainingData: Data source for training.  
///   - parameters: Model training parameters. See  
`MLImageClassifier.ModelParameters` for the defaults.  
///   - sessionParameters: Training session parameters. See  
`MLTrainingSessionParameters` for the defaults.  
/// - Returns: An ``MLJob`` that represents the image classifier  
training session.  
@available macOS 11.0 iOS 15.0  
@available  
public static func train  
MLImageClassifier DataSource  
MLImageClassifier ModelParameters
```

1

```
MLTrainingSessionParameters  
throws      MLJob MLImageClassifier  
  
/// Creates or restores a training session.  
///  
/// - Parameters:  
///   - trainingData: Data source for training.  
///   - parameters: Model training parameters. See  
`MLImageClassifier.ModelParameters` for the defaults.  
///   - sessionParameters: Training session parameters. See  
`MLTrainingSessionParameters` for the defaults.  
/// - Returns: A `MLTrainingSession` that can be used to start or  
resume training.  
@available macOS 11.0 iOS 15.0  
@available  
public static func makeTrainingSession  
MLImageClassifier DataSource  
MLImageClassifier ModelParameters
```

```

    MLTrainingSessionParameters
    throws
MLTrainingSession MLImageClassifier

    /// Creates an asynchronous training session for an image classifier by
restoring an existing training session's
    /// state from its parameters.
    ///
    /// - Parameter sessionParameters: Training session parameters.
See `MLTrainingSessionParameters` for the defaults.
    /// - Returns: An ``MLTrainingSession`` that represents the image
classifier training session.
    @available macOS 11.0 iOS 15.0
    @available
    public static func
restoreTrainingSession
MLTrainingSessionParameters throws
MLTrainingSession MLImageClassifier

    /// Begins or continues an asynchronous image classifier training session.
    ///
    /// - Parameter session: An ``MLTrainingSession`` instance that
represents the training session.
    /// - Returns: An ``MLJob`` that represents the image classifier
training session.
    @available macOS 11.0 iOS 15.0
    @available
    public static func resume
MLTrainingSession MLImageClassifier throws
MLJob MLImageClassifier

@available macOS 10.14 iOS 15.0
@available
extension MLImageClassifier FeatureExtractorType
CustomStringConvertible CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of the feature extractor.
public var description String get

    /// A text representation of the feature extractor that's suitable for output
during debugging.
public var debugDescription String get

    /// A description of the feature extractor shown in a playground.
public var playgroundDescription Any get

```

```
extension MLImageClassifier ModelParameters

    /// The source of a validation dataset for an image classifier.
@available macOS 10.15 iOS 15.0
@available
public enum ValidationData : Sendable

    /// A validation dataset derived by randomly selecting a portion of the
image classifier's training dataset
    /// using the split strategy.
case split : MLSplitStrategy

    /// A validation dataset represented by a data source.
case dataSource : MLImageClassifier.DataSource

    /// A validation dataset represented by a dictionary.
@available 10.15 11.0
    "Use DataSource.filesByLabel to provide dictionary
validation data instead."
@available 15.0 16.0
    "Use DataSource.filesByLabel to provide dictionary
validation data instead."
@available
case dictionary : String : URL

    /// An empty validation dataset that skips the model validation phase
after training.
case none
```

```
@available macOS 10.14 iOS 15.0
@available
extension MLImageClassifier ModelParameters
CustomStringConvertible CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of the model parameters.
public var description : String : get

    /// A text representation of the model parameters that's suitable for output
/// during debugging.
public var debugDescription : String : get

    /// A description of the model parameters shown in a playground.
public var playgroundDescription : Any : get
```

```
extension MLImageClassifier ModelParameters

    /// Type of classifier to be used.
    @available macOS 11.0 iOS 15.0
    @available
    public enum ClassifierType : Equatable, Hashable
    CustomStringConvertible, Sendable

        /// Logistic regression is a statistical model that classifies input feature
        vector into different categories.
        case logisticRegressor

            /// Multilayer perceptron, layerSizes holds a list of positive integers that
            represent the number of hidden
            /// units in each layer. An additional fully connected layer with a
            Softmax activation output will be added
            /// that maps to probabilities of sound categories.
            @available macOS 11.0
            @available
            @available
            case multilayerPerceptron Int

            /// A textual representation of this instance.
            ///
            /// Calling this property directly is discouraged. Instead, convert an
            /// instance of any type to a string by using the
`String(describing:)`  

            /// initializer. This initializer works with any type, and uses the custom
            /// `description` property for types that conform to
            /// `CustomStringConvertible`:
            ///
            /// struct Point: CustomStringConvertible {
            ///     let x: Int, y: Int
            ///
            ///     var description: String {
            ///         return "(\(x), \(y))"
            ///     }
            /// }
            ///
            /// let p = Point(x: 21, y: 30)
            /// let s = String(describing: p)
            /// print(s)
            /// // Prints "(21, 30)"
            ///
            /// The conversion of `p` to a string in the assignment to `s` uses the
            /// `Point` type's `description` property.
public var description : String {
    get
        /// Returns a Boolean value indicating whether two values are equal.
        ///
```

```

    /// Equality is the inverse of inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is `false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to compare.
public static func
MLImageClassifier ModelParameters ClassifierType
MLImageClassifier ModelParameters ClassifierType      Bool

    /// Hashes the essential components of this value by feeding them into
the
    /// given hasher.
    ///
    /// Implement this method to conform to the `Hashable` protocol. The
components
compared
    /// in your type's `==` operator implementation. Call
`hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your implementation of `hash(into:)`,  

    /// don't call `finalize()` on the `hasher` instance provided,  

    /// or replace it with a different instance.  

    /// Doing so may become a compile-time error in the future.
    ///
    /// - Parameter hasher: The hasher to use when combining the
components
    /// of this instance.
public func hash           inout Hasher

    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different
executions of
    /// your program. Do not save hash values to use during a future
execution.
    ///
    /// - Important: `hashValue` is deprecated as a `Hashable`  

requirement. To
    /// conform to `Hashable`, implement the `hash(into:)`  

requirement instead.
    /// The compiler provides an implementation for `hashValue` for
you.
public var hashValue Int get

```

```
extension MLImageClassifier ModelParameters
```

```

    /// The model algorithm to use for training an image classifier.
    @available(macOS 11.0, iOS 15.0)
    @available
    public enum ModelAlgorithmType : CustomStringConvertible
    Sendable

        /// Train using a transfer-learning algorithm with a specified feature
        extractor and classifier.
        case transferLearning
        MLImageClassifier FeatureExtractorType
        MLImageClassifier ModelParameters ClassifierType

        /// A textual representation of this instance.
        ///
        /// Calling this property directly is discouraged. Instead, convert an
        /// instance of any type to a string by using the
        `String(describing:)`
        /// initializer. This initializer works with any type, and uses the custom
        /// `description` property for types that conform to
        /// `CustomStringConvertible`:
        ///
        ///     struct Point: CustomStringConvertible {
        ///         let x: Int, y: Int
        ///
        ///         var description: String {
        ///             return "(\(x), \(y))"
        ///         }
        ///     }
        ///
        ///     let p = Point(x: 21, y: 30)
        ///     let s = String(describing: p)
        ///     print(s)
        ///     // Prints "(21, 30)"
        ///
        /// The conversion of `p` to a string in the assignment to `s` uses the
        /// `Point` type's `description` property.
    public var description: String { get

```

```

    /// The representation of a model's asynchronous training session you use to
    /// monitor the session's progress or terminate its execution.

```

```

    @available(macOS 11.0, iOS 15.0, tvOS 16.0)
    final public class MLJob.Result : Cancellable

```

```

        /// The date and time when the training session began.
        final public let startDate: Date

```

```

        /// The training session's current progress.

```

```
final public let progress Progress

/// A Boolean value that indicates whether you canceled the job.
final public var isCanceled Bool get

/// A publisher that sends a checkpoint for each of the session's checkpoint
/// intervals.
final public var checkpoints AnyPublisher MLCheckpoint
Never get

/// A publisher that provides a result when the training session has
/// finished.
final public var result AnyPublisher Result any Error
get

/// Phase publisher.
@available macOS 12.0 iOS 15.0 tvOS 16.0
final public var phase AnyPublisher MLPhase Never get

/// Stops the training session's execution.
final public func cancel

/// A regressor that estimates the target as a linear function of the features.
@available macOS 10.14 iOS 15.0 tvOS 16.0
public struct MLLinearRegressor @unchecked Sendable

/// The Core ML model.
public var model MLModel

/// The name of the column you selected at initialization to define which
feature the regressor predicts.
///
/// Changing the value of this property doesn't retrain the model or affect its
behavior.
public var targetColumn String

/// The names of the columns you selected at initialization to train the
regressor.
///
/// Changing the value of this property doesn't retrain the model or affect its
behavior.
public var featureColumns String

/// The underlying parameters used when training the model.
public let modelParameters
MLLinearRegressor ModelParameters

/// Measurements of the regressor's performance on the training data set.
```

```
public var trainingMetrics MLRegressorMetrics get
    /// Measurements of the regressor's performance on the validation data set.
public var validationMetrics MLRegressorMetrics get
    /// Creates a linear regressor.
    ///
    /// - Parameters:
    ///   - trainingData: The training data
    ///   - targetColumn: Name of the column containing the target values
    ///   - featureColumns: Names of the columns containing feature
    ///   values. If `nil` all columns, other than the target
    ///   column, will be used as feature values.
    ///   - parameters: Model training parameters
@available macOS 12.0 iOS 15.0 tvOS 16.0
public init DataFrame String
String nil
MLLinearRegressor ModelParameters
throws

    /// Creates a Linear Regressor from the feature columns in the training data
    /// to predict the values in the target
    /// column.
    ///
    /// - Parameters:
    ///   - trainingData: A data table of training examples.
    ///   - targetColumn: The column name for the values in the training
    ///   data the regressor should predict.
    ///   - featureColumns: The column names for the values in the
    ///   training data that the regressor uses to predict the
    ///   target value.
    ///   - parameters: The model parameters.
@available 10.14 13.0
    "Use DataFrame instead of MLDataTable when
initializing."
@available 15.0 16.0
    "Use DataFrame instead of MLDataTable when
initializing."
@available
public init MLDataTable
String String nil
MLLinearRegressor ModelParameters
nil throws

    /// Creates a linear regressor from a checkpoint.
    ///
    /// - Parameter checkpoint: Training checkpoint.
    /// - Throws: `MLCreateError` if the checkpoint can't be loaded.
@available macOS 12.0 iOS 15.0 tvOS 16.0
```

```

public init          MLCheckpoint throws

    /// Trains a linear regressor.
    ///
    /// If `sessionDirectory` is provided it will save training progress. If
    there is progress already saved training
    /// will resume from the last checkpoint.
    ///
    /// - Parameters:
    ///   - trainingData: A DataTable specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
    trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
    columns to be
    ///
    default to use all the
    ///
    one specified by targetColumn
    ///   - parameters: Model training parameters. See
    `MLLinearRegressor.ModelParameters` for the defaults.
    ///   - sessionParameters: Training session parameters. See
    `MLTrainingSessionParameters` for the defaults.
    ///
    /// - Returns: A `MLJob` that can be used to observe training progress.
@available           12.0           14.0
@available           15.0           17.0
@available           16.0           17.0
public static func train           MLDataTable
String                String           nil
MLLinearRegressor ModelParameters

MLTrainingSessionParameters
throws      MLJob MLLinearRegressor

    /// Trains a linear regressor.
    ///
    /// If `sessionDirectory` is provided it will save training progress. If
    there is progress already saved training
    /// will resume from the last checkpoint.
    ///
    /// - Parameters:
    ///   - trainingData: A `DataFrame` specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
    trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
    columns to be
    ///
    default to use all the
    ///
    one specified by targetColumn
    ///   - parameters: Model training parameters. See

```

```

`MLLinearRegressor.ModelParameters` for the defaults.
    /// - sessionParameters: Training session parameters. See
`MLTrainingSessionParameters` for the defaults.
    ///
    /// - Returns: A `MLJob` that can be used to observe training progress.
@available macOS 12.0 iOS 15.0 tvOS 16.0
public static func train MLTrainingSessionParameters throws MLJob MLLinearRegressor

    /// Creates or restores a training session.
    ///
    /// - Parameters:
    ///   - trainingData: A DataTable specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
    trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
    columns to be
        /// used to predict the target, if not provided, default to use
    all the
        /// other columns in the trainingData, except the one
    specified by targetColumn
    ///   - parameters: Model training parameters. See
`MLLinearRegressor.ModelParameters` for the defaults.
    ///   - sessionParameters: Training session parameters. See
`MLTrainingSessionParameters` for the defaults.
    ///
    /// - Returns: A `MLTrainingSession` that can be used to start or
resume training.
@available 12.0 14.0
@available 15.0 17.0
@available 16.0 17.0
public static func makeTrainingSession
MLDataTable String String
nil MLLinearRegressor ModelParameters

    MLTrainingSessionParameters
    throws
MLTrainingSession MLLinearRegressor

    /// Creates or restores a training session.
    ///
    /// - Parameters:
    ///   - trainingData: A `DataFrame` specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
    trainingData
    ///   - featureColumns: An optional list of Strings specifying feature

```

```
columns to be
    /**
     * used to predict the target, if not provided, default to use
all the
    /**
     * other columns in the trainingData, except the one
specified by targetColumn
    /**
     * - parameters: Model training parameters. See
`MLLinearRegressor.ModelParameters` for the defaults.
    /**
     * - sessionParameters: Training session parameters. See
`MLTrainingSessionParameters` for the defaults.
    /**
    /**
     * - Returns: A `MLTrainingSession` that can be used to start or
resume training.
@available macOS 12.0 iOS 15.0 tvOS 16.0
public static func makeTrainingSession
DataFrame String String
nil MLLinearRegressor ModelParameters

MLTrainingSessionParameters
throws
MLTrainingSession MLLinearRegressor

    /**
     * Restores an existing training session.
    /**
    /**
     * - Parameters:
    /**
     * - sessionParameters: Training session parameters. The
`sessionDirectory` parameter is required.
    /**
    /**
     * - Returns: A `MLTrainingSession` that can be used to resume
training.
@available macOS 12.0 iOS 15.0 tvOS 16.0
public static func
restoreTrainingSession
MLTrainingSessionParameters throws
MLTrainingSession MLLinearRegressor

    /**
     * Resumes a training session from the last checkpoint if available.
    /**
    /**
     * If there are no resumable checkpoints training starts over from the
beginning.
    /**
    /**
     * - Parameter session: Loaded or new training session.
    /**
    /**
     * - Returns: A `MLJob` that can be used to observe training progress.
@available macOS 12.0 iOS 15.0 tvOS 16.0
public static func resume
MLTrainingSession MLLinearRegressor throws
MLJob MLLinearRegressor

    /**
     * Predicts a column of labels for the given testing data.
@available macOS 12.0 iOS 15.0 tvOS 16.0
```

```
public func predictions      DataFrame throws
AnyColumn

    /// Predicts the target value from the provided data.
    ///
    /// - Parameters:
    ///   - data: The data you want the model to make predictions from.
    ///
    /// - Returns: A column of values predicted by the regressor.
@available(macOS 10.14, iOS 13.0)
    "Use DataFrame instead of MLDataTable."
@available(iOS 15.0, tvOS 16.0)
    "Use DataFrame instead of MLDataTable."
@available
public func predictions      MLDataTable throws
MLUntypedColumn

    /// Evaluates the classifier on the provided labeled data.
    ///
    /// Evaluation should be done on a testing data set that the model has not
    seen as part of the training or
    /// validation data sets. The data should have feature columns with identical
    name and type to the
    /// training data, as well as a labels column with the same name.
    ///
    /// - Parameters:
    ///   - labeledData: A `DataFrame` to evaluate the trained model on.
    ///
    /// - Returns: Metrics that describe the maximum error
    ///(``MLRegressorMetrics/maximumError``) or the average error
    ///(``MLRegressorMetrics/rootMeanSquaredError``).
@available(macOS 12.0, iOS 15.0, tvOS 16.0)
public func evaluation      DataFrame
MLRegressorMetrics

    /// Evaluates the classifier on the provided labeled data.
    ///
    /// Evaluation should be done on a testing data set that the model has not
    seen as part of the training or
    /// validation data sets. The data should have feature columns with identical
    name and type to the
    /// training data, as well as a labels column with the same name.
    ///
    /// - Parameters:
    ///   - labeledData: An `MLDataTable` to evaluate the trained model
    on.
    ///
    /// - Returns: Metrics that describe the maximum error
    ///(``MLRegressorMetrics/maximumError``) or the average error
    ///(``MLRegressorMetrics/rootMeanSquaredError``).
```

```

@available 10.14 13.0
    "Use DataFrame instead of MLDataTable."
@available 15.0 16.0
    "Use DataFrame instead of MLDataTable."
@available
public func evaluation MLDataTable
MLRegressorMetrics

/// Exports a Core ML model file for use in your app.
public func write URL
MLModelMetadata nil throws

/// Exports a Core ML model file for use in your app.
public func write String
MLModelMetadata nil throws

extension MLLinearRegressor

/// Parameters that affect the process of training a model.
@available macOS 10.14 iOS 15.0 tvOS 16.0
public struct ModelParameters

    /// Validation data represented as a `MLDataTable`.
    ///
    /// - Note: Setting this to `nil` means that the training data will be
    automatically split for
    /// validation. Setting it to an empty table means to not use a
    validation set.
    @available 10.14 11.0
    "Use the validation property instead."
    @available 15.0 16.0
    "Use the validation property instead."
    @available
    public var validationData MLDataTable

    /// Validation data.
    ///
    /// The default is `split(strategy: .automatic)`, which
    automatically generates the validation
    /// dataset from 0% to 10% of the training dataset.
    @available macOS 11.0 iOS 15.0 tvOS 16.0
    public var validation
MLLinearRegressor ModelParameters ValidationData

    public var maxIterations Int
    public var l1Penalty Double
    public var l2Penalty Double

```

```

public var stepSize Double

public var convergenceThreshold Double

public var featureRescaling Bool

@available macOS 11.0 iOS 15.0 tvOS 16.0
public init

MLLinearRegressor ModelParameters ValidationData
    Int 10           Double 0
Double 0.01           Double 1.0
Double 0.01           Bool true

    /// Creates a new set of parameters.
    ///
    /// - Parameters:
    ///     - validationData: The dataset used to monitor how well the
model is generalizing.
    ///
    ///     The default value is `nil` which will use an automatically
sampled validation set.
    ///
    ///     - maxIterations: The maximum number of passes through
the data.
    ///
    ///     The default value is 10.
    ///
    ///     - l1Penalty: Weight on the l1-regularizer. The l1Penalty
zeros out small coefficients, indicating
    ///         features that are not useful for the model.
    ///
    ///     The default value is 0 which prevents any values from being
discarded.
    ///
    ///     - l2Penalty: Weight of the l2-regularizer. The larger the
l2Penalty the less variance in the model.
    ///
    ///     The default value is 0.01.
    ///
    ///     - stepSize: The adjustment size that should be made by the
underlying solver. Values close to 1.0 take an
    ///         aggressive step based off feedback from each training
iteration.
    ///
    ///     The default value is 1.0.
    ///
    ///     - convergenceThreshold: The threshold with which to
determine if the model has converged. Consider
    ///         reducing this value for higher training accuracy, but beware of
overfitting.

```

```
///  
///      The default value is 0.01.  
///  
///      - featureRescaling: Determines if the features should be  
// preprocessed to ensure all features are on the  
///      same scale.  
///  
///      The default value is true.  
@available 10.14 11.0  
    "Use the validation property instead."  
@available 15.0 16.0  
    "Use the validation property instead."  
@available  
public init MLDataTable nil  
    Int 10 Double 0  
Double 0.01 Double 1.0  
Double 0.01 Bool true
```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0  
extension MLLinearRegressor CustomStringConvertible
```

```
/// A text representation of the linear regressor.  
public var description String get
```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0  
extension MLLinearRegressor CustomDebugStringConvertible
```

```
/// A text representation of the linear regressor that's suitable for output  
// during debugging.  
public var debugDescription String get
```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0  
extension MLLinearRegressor  
CustomPlaygroundDisplayConvertible
```

```
/// A description of the linear regressor shown in a playground.  
public var playgroundDescription Any get
```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0  
extension MLLinearRegressor ModelParameters  
CustomStringConvertible
```

```
/// A text representation of the model parameters for a linear regressor.  
public var description String get
```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLLinearRegressor ModelParameters
CustomDebugStringConvertible

    /// A text representation of the model parameters for a linear regressor that's
    suitable for output during
    /// debugging.
public var debugDescription String get

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLLinearRegressor ModelParameters
CustomPlaygroundDisplayConvertible

    /// A description of the model parameters for a linear regressor shown in a
    playground.
public var playgroundDescription Any get

extension MLLinearRegressor ModelParameters

    /// Values for specifying validation data.
@available macOS 11.0 iOS 15.0 tvOS 16.0
public enum ValidationData

        /// Generate validation data by splitting the training dataset. This is the
        default.
case split MLSplitStrategy

        /// Set validation data from the MLDataTable provided.
@available macOS 11.0 iOS 14.0
@available macOS 15.0 iOS 17.0
@available macOS 16.0 iOS 17.0
case table MLDataTable

        /// Set validation data from the DataFrame provided.
@available macOS 13.0 iOS 16.0 tvOS 16.0
case dataFrame DataFrame

        /// Do not set validation data.
case none

/// A classifier that predicts a discrete target value as a function of data features.
@available macOS 10.14 iOS 15.0 tvOS 16.0
public struct MLLogisticRegressionClassifier @unchecked
Sendable
```

```
    ///> The Core ML model.
    public var model  MLModel

    ///> The name of the column you selected at initialization to define which
    categories the classifier predicts.
    ///
    ///> Changing the value of this property doesn't retrain the model or affect its
    behavior.
    public var targetColumn  String

    ///> The names of the columns you selected at initialization to train the
    classifier.
    ///
    ///> Changing the value of this property doesn't retrain the model or affect its
    behavior.
    public var featureColumns  String

    ///> The underlying parameters used when training the model.
    public let modelParameters
MLLogisticRegressionClassifier ModelParameters

    ///> Measurements of the classifier's performance on the training data set.
    public var trainingMetrics  MLClassifierMetrics  get

    ///> Measurements of the classifier's performance on the validation data set.
    public var validationMetrics  MLClassifierMetrics  get

    ///> Creates a logistic regression classifier.
    ///
    ///> - Parameters:
    ///>   - trainingData: The training data
    ///>   - targetColumn: Name of the column containing the class labels
    ///>   - featureColumns: Names of the columns containing feature
    values. If `nil` all columns, other than the target
    ///>   column, will be used as feature values.
    ///>   - parameters: Model training parameters
    @available macOS 12.0  iOS 15.0  tvOS 16.0
    public init           DataFrame           String
                           String      nil
MLLogisticRegressionClassifier ModelParameters

throws

    ///> Creates a Logistic Regression Classifier from the feature columns in the
    training data to predict the
    ///> categories in the target column.
    ///
    ///> - Parameters:
    ///>   - trainingData: A data table of training examples.
    ///>   - targetColumn: The column name for the values in the training
```

```

data that the classifier should predict.

    /// - featureColumns: The column names for the values in the
    training data that the classifier uses to predict
    /// the target value.
    /// - parameters: The model parameters.
    @available(10.14, 13.0)
        "Use DataFrame instead of MLDataTable when
initializing."
    @available(15.0, 16.0)
        "Use DataFrame instead of MLDataTable when
initializing."
    @available
    public init(MLDataTable)
String nil
MLLogisticRegressionClassifier(ModelParameters)
nil throws

    /// Creates a logistic regression classifier from a checkpoint.
    ///
    /// - Parameter checkpoint: Training checkpoint.
    /// - Throws: `MLCreateError` if the checkpoint can't be loaded.
    @available(macOS 12.0, iOS 15.0, tvOS 16.0)
    public init(MLCheckpoint) throws

    /// Trains a logistic regression classifier.
    ///
    /// If `sessionDirectory` is provided it will save training progress. If
    there is progress already saved training
    /// will resume from the last checkpoint.
    ///
    /// - Parameters:
    ///   - trainingData: A DataTable specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
    trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
    columns to be
    ///                               used to predict the target, if not provided,
    default to use all the
    ///                               other columns in the trainingData, except the
    one specified by targetColumn
    ///   - parameters: Model training parameters. See
    `MLLogisticRegressionClassifier.ModelParameters` for the defaults.
    ///   - sessionParameters: Training session parameters. See
    `MLTrainingSessionParameters` for the defaults.
    ///
    /// - Returns: A `MLJob` that can be used to observe training progress.
    @available(12.0, 14.0)
    @available(15.0, 17.0)
    @available(16.0, 17.0)
    public static func train(MLDataTable)

```

```
        String           String     nil
        MLLogisticRegressionClassifier ModelParameters

MLTrainingSessionParameters
throws    MLJob  MLLogisticRegressionClassifier

    /// Trains a logistic regression classifier.
    ///
    /// If `sessionDirectory` is provided it will save training progress. If
    there is progress already saved training
    /// will resume from the last checkpoint.
    ///
    /// - Parameters:
    ///   - trainingData: A `DataFrame` specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
    trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
    columns to be
    ///                               used to predict the target, if not provided,
    default to use all the
    ///                               other columns in the trainingData, except the
    one specified by targetColumn
    ///   - parameters: Model training parameters. See
    `MLLogisticRegressionClassifier.ModelParameters` for the defaults.
    ///   - sessionParameters: Training session parameters. See
    `MLTrainingSessionParameters` for the defaults.
    ///
    /// - Returns: A `MLJob` that can be used to observe training progress.
    @available macOS 12.0  iOS 15.0  tvOS 16.0
public static func train           DataFrame
        String           String     nil
        MLLogisticRegressionClassifier ModelParameters

MLTrainingSessionParameters
throws    MLJob  MLLogisticRegressionClassifier

    /// Creates or restores a training session.
    ///
    /// - Parameters:
    ///   - trainingData: A DataTable specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
    trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
    columns to be
    ///                               used to predict the target, if not provided, default to use
    all the
    ///                               other columns in the trainingData, except the one
    specified by targetColumn
    ///   - parameters: Model training parameters. See
    `MLLogisticRegressionClassifier.ModelParameters` for the defaults.
    ///   - sessionParameters: Training session parameters. See
```

```

`MLTrainingSessionParameters` for the defaults.


    /**
     * - Returns: A `MLTrainingSession` that can be used to start or
     * resume training.
     */


@available(macOS 12.0, iOS 15.0, tvOS 16.0)
public static func makeTrainingSession(
    _ trainingData: DataFrame,
    _ targetColumn: String,
    _ featureColumns: [String] = [],
    _ parameters: ModelParameters = ModelParameters(),
    _ sessionParameters: MLTrainingSessionParameters = MLTrainingSessionParameters()
) throws
    MLTrainingSession


/// Creates or restores a training session.


    /**
     * - Parameters:
     *   - trainingData: A `DataFrame` specifying training data.
     *   - targetColumn: A String specifying the target column name in the
     *     trainingData.
     *   - featureColumns: An optional list of Strings specifying feature
     *     columns to be
     *       used to predict the target, if not provided, default to use
     *     all the
     *       other columns in the trainingData, except the one
     *     specified by targetColumn.
     *   - parameters: Model training parameters. See
     *     `MLLogisticRegressionClassifier.ModelParameters` for the defaults.
     *   - sessionParameters: Training session parameters. See
     *     `MLTrainingSessionParameters` for the defaults.
     */


    /**
     * - Returns: A `MLTrainingSession` that can be used to start or
     * resume training.
     */


@available(macOS 12.0, iOS 15.0, tvOS 16.0)
public static func makeTrainingSession(
    _ sessionDirectory: URL,
    _ parameters: ModelParameters = ModelParameters(),
    _ sessionParameters: MLTrainingSessionParameters = MLTrainingSessionParameters()
) throws
    MLTrainingSession


/// Restores an existing training session.


    /**
     * - Parameters:
     *   - sessionParameters: Training session parameters. The
     *     `sessionDirectory` parameter is required.
     */


    /**
     * - Returns: A `MLTrainingSession` that can be used to resume
     */


```

training.

```
    @available macOS 12.0  iOS 15.0  tvOS 16.0
    public static func
    restoreTrainingSession
    MLTrainingSessionParameters  throws
    MLTrainingSession MLLogisticRegressionClassifier

    /// Resumes a training session from the last checkpoint if available.
    ///
    /// If there are no resumable checkpoints training starts over from the
    beginning.
    ///
    /// - Parameter session: Loaded or new training session.
    ///
    /// - Returns: A `MLJob` that can be used to observe training progress.
    @available macOS 12.0  iOS 15.0  tvOS 16.0
    public static func resume
    MLTrainingSession MLLogisticRegressionClassifier  throws
    MLJob MLLogisticRegressionClassifier

    /// Predicts a column of labels for the given testing data.
    @available macOS 12.0  iOS 15.0  tvOS 16.0
    public func predictions      DataFrame  throws
    AnyColumn

    /// Classifies the provided data into the target categories.
    ///
    /// - Parameters:
    ///   - data: The data you want the model to classify.
    ///
    /// - Returns: A column of labels predicted by the classifier.
    @available           10.14          13.0
    "Use DataFrame instead of MLDataTable."
    @available           15.0          16.0
    "Use DataFrame instead of MLDataTable."
    @available
    public func predictions      MLDataTable  throws
    MLUntypedColumn

    /// Evaluates the classifier on the provided labeled data.
    ///
    /// Evaluation should be done on a testing data set that the model has not
    seen as part of the training or
    /// validation data sets. The data should have feature columns with identical
    name and type to the
    /// training data, as well as a labels column with the same name.
    ///
    /// - Parameters:
    ///   - labeledData: A `DataFrame` to evaluate the trained model on.
    ///
```

```

    /// - Returns: Metrics that describe the classification errors
    /// (` `MLClassifierMetrics/classificationError` `), the precision
and recall
    /// percentages (` `MLClassifierMetrics/precisionRecall` `), and
a table that
    /// describes how labels were misapplied
(` `MLClassifierMetrics/confusion` `)
    /// on the provided data.
@available macOS 12.0 iOS 15.0 tvOS 16.0
public func evaluation DataFrame
MLClassifierMetrics

    /// Evaluates the classifier on the provided labeled data.
    ///
    /// Evaluation should be done on a testing data set that the model has not
seen as part of the training or
    /// validation data sets. The data should have feature columns with identical
name and type to the
    /// training data, as well as a labels column with the same name.
    ///
    /// - Parameters:
    ///   - labeledData: An `MLDataTable` to evaluate the trained model
on.
    ///
    /// - Returns: Metrics that describe the classification errors
    /// (` `MLClassifierMetrics/classificationError` `), the precision
and recall
    /// percentages (` `MLClassifierMetrics/precisionRecall` `), and
a table that
    /// describes how labels were misapplied
(` `MLClassifierMetrics/confusion` `)
    /// on the provided data.
@available 10.14 13.0
    "Use DataFrame instead of MLDataTable."
@available 15.0 16.0
    "Use DataFrame instead of MLDataTable."
available
public func evaluation MLDataTable
MLClassifierMetrics

    /// Exports a Core ML model file for use in your app.
public func write URL
MLModelMetadata nil throws

    /// Exports a Core ML model file for use in your app.
public func write String
MLModelMetadata nil throws

```

**extension** **MLLogisticRegressionClassifier**

```

/// Parameters that affect the process of training a model.
@available(macOS 10.14, iOS 15.0, tvOS 16.0)
public struct ModelParameters

    public var maxIterations: Int

        /// Validation data represented as a `MLDataTable`.
        ///
        /// - Note: Setting this to `nil` means that the training data will be
        automatically split for
        /// validation. Setting it to an empty table means to not use a
validation set.
        @available(10.15, 10.14)
        "Use the validation property instead."
        @available(15.0, 16.0)
        "Use the validation property instead."
        @available
        public var validationData: MLDataTable

            /// Validation data.
            ///
            /// The default is `split(strategy: .automatic)`, which
automatically generates the validation
            /// dataset from 0% to 10% of the training dataset.
            @available(macOS 10.15, iOS 15.0, tvOS 16.0)
            public var validation
MLLogisticRegressionClassifier ModelParameters ValidationData

        public var l1Penalty: Double
        public var l2Penalty: Double
        public var stepSize: Double
        public var convergenceThreshold: Double
        public var featureRescaling: Bool

        @available(macOS 10.15, iOS 15.0, tvOS 16.0)
        public init
MLLogisticRegressionClassifier ModelParameters ValidationData
   Int 10
   Double 0           Double 0.01
Double 1.0                      Double 0.01
   Bool true

        /// Creates a new set of parameters.
        ///
        /// - Parameters:

```

```
    /**
     * - validationData: The dataset used to monitor how well the
     * model is generalizing.
     */
    /**
     * The default value is `nil` which will use an automatically
     * sampled validation set.
     */
    /**
     * - maxIterations: The maximum number of passes through
     * the data.
     */
    /**
     * The default value is 10.
     */
    /**
     * - l1Penalty: Weight on the L1-regularizer. The L1Penalty
     * zeros out small coefficients, indicating
     * features that are not useful for the model.
     */
    /**
     * The default value is 0 which prevents any values from being
     * discarded.
     */
    /**
     * - l2Penalty: Weight of the L2-regularizer. The larger the
     * L2Penalty the less variance in the model.
     */
    /**
     * The default value is 0.01.
     */
    /**
     * - stepSize: The adjustment size that should be made by the
     * underlying solver. Values close to 1.0 take an
     * aggressive step based off feedback from each training
     * iteration.
     */
    /**
     * The default value is 1.0.
     */
    /**
     * - convergenceThreshold: The threshold with which to
     * determine if the model has converged. Consider
     * reducing this value for higher training accuracy, but beware of
     * overfitting.
     */
    /**
     * The default value is 0.01.
     */
    /**
     * - featureRescaling: Determines if the features should be
     * preprocessed to ensure all features are on the
     * same scale.
     */
    /**
     * The default value is true.
     */
@available 10.14 "Use the validation property instead."
@available 15.0 16.0 "Use the validation property instead."
@available
public init MLDataTable
    Int 10 Double 0
Double 0.01 Double 1.0
Double 0.01 Bool true
```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLLogisticRegressionClassifier
CustomStringConvertible

    /// A text representation of the logistic regression classifier.
public var description String get

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLLogisticRegressionClassifier
CustomDebugStringConvertible

    /// A text representation of the logistic regression classifier that's suitable for
output during debugging.
public var debugDescription String get

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLLogisticRegressionClassifier
CustomPlaygroundDisplayConvertible

    /// A description of the logistic regression classifier shown in a playground.
public var playgroundDescription Any get

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLLogisticRegressionClassifier ModelParameters
CustomStringConvertible

    /// A text representation of the model parameters for a logistic regression
classifier.
public var description String get

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLLogisticRegressionClassifier ModelParameters
CustomDebugStringConvertible

    /// A text representation of the model parameters for a logistic regression
classifier that's suitable for output
/// during debugging.
public var debugDescription String get

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLLogisticRegressionClassifier ModelParameters
CustomPlaygroundDisplayConvertible
```

```
    /// A description of the model parameters for a logistic regression classifier  
    shown in a playground.  
    public var playgroundDescription Any get  
  
extension MLLogisticRegressionClassifier ModelParameters  
  
    /// Values for specifying validation data.  
    @available macOS 10.15 iOS 15.0 tvOS 16.0  
    public enum ValidationData  
  
        /// Generate validation data by splitting the training dataset. This is the  
        default.  
        case split MLSplitStrategy  
  
        /// Set validation data from the MLDataTable provided.  
        @available macOS 10.15 14.0  
        @available iOS 15.0 17.0  
        @available tvOS 16.0 17.0  
        case table MLDataTable  
  
        /// Validation data provided in a DataFrame.  
        @available macOS 12.0 iOS 15.0 tvOS 16.0  
        case dataFrame DataFrame  
  
        /// Do not set validation data.  
        case none
```

```
    /// Information about a model that's stored in a Core ML model file.  
    ///  
    /// Create a metadata instance and store it as part of your model when you export  
    a Core ML model. You can examine this  
    /// metadata in Xcode when you import the model into your app.  
    @available macOS 10.14 iOS 15.0 tvOS 16.0  
    public struct MLModelMetadata Sendable  
  
        /// The author of the model.  
        public var author String  
  
        /// A short text description of the model.  
        public var shortDescription String  
  
        /// The license governing the use of the model.  
        public var license String  
  
        /// The model version.  
        public var version String
```

```

    /// A dictionary that encodes key value pairs to hold additional information
    // about the model.
    public var additional String String

    /// Creates a new metadata instance for a machine learning model.
    ///
    /// - Parameters:
    ///   - author: The author of the model.
    ///   - shortDescription: A short description of the model.
    ///   - license: The license governing the use of the model.
    ///   - version: The model version.
    ///   - additional: A dictionary that you can use to store arbitrary key
    value
    /// pairs.
    public init String
                    String "A model trained using CreateML for
use with CoreML." String nil String
"1"           String String nil

    /// A model you train to classify one or more objects within an image.
    ///
    /// Use an ``MLObjectDetector`` task to train a machine learning model that
    can
    /// identify items, or _objects_, within an image. For example, you can train an
    /// object detector to recognize breakfast items on a table, such as bananas,
    /// croissants, and beverages.
    ///
    /// You create an object detector training it with a combination of images and
    /// annotations for each object within an image. Then save it as a Core ML model
    /// and use it in your app to recognize similar items.
    @available macOS 10.15
    @available
    @available
    public struct MLObjectDetector @unchecked Sendable

        /// An array of annotations that represent the items an object detector
        /// found in an image.
        public typealias DetectedObjects
MLObjectDetector ObjectAnnotation

        /// The object detector's underlying Core ML model instance.
        public var model MLMModel get

            /// The model configuration parameters the object detector used during its
            training session.
            public let modelParameters
MLObjectDetector ModelParameters

```

```
    /// Measurements of the object detector's performance on the training
    dataset.
    public var trainingMetrics  MLObjectDetectorMetrics
get

    /// Measurements of the object detector's performance on the validation
    dataset.
    public var validationMetrics  MLObjectDetectorMetrics
get

    /// Creates an object detector with a data source.
    ///
    /// Use this initializer to create an object detector with an
    /// ``MLObjectDetector/DataSource``.
    ///
    /// - Parameters:
    /// - trainingData: A data source that contains the annotated images
    /// the
    /// task uses to train the object detector.
    ///
    /// - parameters: An ``MLObjectDetector/ModelParameters-
    swift.struct``
    /// instance you use to set the model configuration settings for the
    /// training session.
    ///
    /// - annotationType : The format your data source uses for its image
    /// annotations.
public init          MLObjectDetector DataSource
                    MLObjectDetector ModelParameters
                    MLObjectDetector AnnotationType throws

    /// Creates an object detector with a data table.
    ///
    /// Use this initializer to create an object detector with an
    /// ``MLDataTable``.
    ///
    /// - Parameters:
    /// - trainingData: An ``MLDataTable`` that contains the annotated
    images
    /// the task uses to train the object detector.
    ///
    /// - imageColumn: The name of the column in the data table that contains
    /// the image file URLs.
    ///
    /// - annotationColumn: The name of the column in the data table that
    /// contains the image annotations.
    ///
    /// - annotationType : The format your data table uses for its image
    /// annotations.
    ///
```

```
    /// - parameters: An ``MLObjectDetector/ModelParameters``  
    /// instance you use to set the model configuration settings for the  
    /// training session.  
    @available(10.15, 13.0)  
    "Use DataSource instead of MLDataTable when  
initializing."  
    @available  
    @available  
    public init  
        String  
        String  
        MLObjectDetector  
        AnnotationType  
        MLObjectDetector  
        ModelParameters  
throws  
  
@available(macOS 10.15)  
@available  
@available  
extension MLObjectDetector  
  
    /// Generates metrics by evaluating the object detector's performance using  
    /// annotated images in a data source.  
    ///  
    /// - Parameters:  
    ///     - annotatedImages: An ``MLObjectDetector/DataSource``  
instance that  
    ///         contains a set of images with object annotations.  
    ///  
    /// - Returns: An ``MLObjectDetectorMetrics`` instance that  
represents the  
    ///     object detector's performance on the annotated images.  
    public func evaluation  
        MLObjectDetector  
        DataSource  
        MLObjectDetectorMetrics  
  
    /// Generates metrics by evaluating the object detector's performance using  
    /// annotated images in a data table.  
    ///  
    /// - Parameters:  
    ///     - annotatedImages: An ``MLDataTable`` instance that  
contains a set of  
    ///         images with object annotations.  
    ///  
    ///     - imageColumn: The name of the column in the data table that  
contains  
    ///         the image file URLs.  
    ///  
    ///     - annotationColumn: The name of the column in the data table  
that  
    ///         contains the object annotations.  
    ///
```

```

    /// - Returns: An ``MLObjectDetectorMetrics`` instance that
    represents the
    /// object detector's performance on the annotated images.
    @available(10.15, 13.0)
    "Use DataSource instead of MLDataTable."
    @available
    @available
    public func evaluation
        String
    MLObjectDetectorMetrics
        String
        MLDataTable

@available(macOS 10.15)
@available
@available
extension MLObjectDetector

    /// A data source for an object detector.
    ///
    /// You use a data source to specify the training dataset for an
    /// ``MLObjectDetector`` training session. An object-detector data
    source
    /// represents a set of images and an annotation for each object in an
    /// image.
    ///
    /// Each object annotation consists of the object's name, or _label_, and
    /// its location in the image. A single image can have multiple objects and,
    /// therefore, multiple annotations. For example, you can train an object
    /// detector with images of dining tables, along with annotations for
    /// bananas, croissants, and beverages. Each image can have one or more
    /// instances of an object, or any combination of objects.
    public enum DataSource

        /// An object-detector data source you create by selecting a directory
        /// that contains image files and exactly one JSON annotation file.
        ///
        /// - Parameters:
        ///   - directoryWithImagesAndJsonAnnotation: The location
        of a directory
        /// that contains exactly one JSON annotation file and all the image
        /// files the JSON file's annotations refer to.
        case directoryWithImagesAndJsonAnnotation URL

        /// An object-detector data source you create by selecting the location
        /// of a directory of image files, and the location of a JSON annotation
        /// file.
        ///
        /// - Parameters:
        ///   - directoryWithImages: The location of a directory that
contains

```

```
    /// image files.  
    ///  
    /// – annotationFile: The location of a JSON file with object  
    /// annotations for the images.  
case directoryWithImages URL URL  
  
    /// An object-detector data source you create with a data table.  
    ///  
    /// – Parameters:  
    /// – table: An ``MLDataTable`` instance that contains a column  
of image  
    /// file locations and a column of object annotations.  
    ///  
    /// – imageColumn: The name of the column in the data table that  
    /// contains the URL for an image.  
    ///  
    /// – annotationColumn: The name of the column in the data table that  
    /// contains the object annotations for an image.  
@available 11.0 14.0  
@available  
@available  
case table MLDataTable String  
String  
  
    /// Data specified by a `DataFrame` containing a column for image  
file paths and a column with annotations.  
@available macOS 12  
@available  
@available  
case frame DataFrame String  
String  
  
    /// Processes the data source and returns a data frame that contains  
file URLs and annotations.  
    ///  
    /// This method collects file names from the filesystem if necessary. If  
the data source is already in table  
    /// format it renames the columns to the default column names.  
@available macOS 14.0  
public func gatherAnnotatedFileNames throws  
DataFrame  
  
    /// Generates a data table where each row represents an image, and  
its  
    /// columns are the image file URLs and its annotations.  
    ///  
    /// – Returns: An ``MLDataTable`` containing the contents of  
the data  
    /// source.  
@available 10.15 14.0
```

```

    "Use gatherAnnotatedFileNames()"
    @available
    @available
    public func imagesWithObjectAnnotations    throws
MLDataTable

        /// Generates a new data table by splitting the data source using the
        /// specified proportions.
        ///
        /// - Parameters:
        ///   - proportions: An array of doubles, each representing a
portion of
        ///   the data source. If these values don't add up to `1.0`, the method
        ///   normalizes the numbers so that they do.
        ///
        ///   - seed: The value the method uses to initialize the random-number
        ///   generator, which affects how the method splits the data.
        ///
        ///   - annotationColumn: The name of the column the method uses to
split
        ///   the data.
        ///
        /// - Returns: An ``MLDataTable`` containing the data source's
split
        ///   contents.
@available                         10.15          14.0
    "Use DataFrame.stratifiedSplit(on:by:)"
@available
@available
    public func stratifiedSplit
Int                               Double
MLDataTable                        String    throws

```

```

@available macOS 10.15
@available
@available
extension MLObjectDetector

    /// The label, location, and confidence score of an item the object detector
found in an image.
    public struct ObjectAnnotation    Sendable

        /// The name of the item the object detector found in an image.
        public var label    String

        /// A rectangular region that encloses an item the object detector found
in the image.
        public var boundingBox    CGRect

```

```

    /// The object detector's confidence score for its prediction's accuracy.
    ///
    /// The confidence range is `[0.0, 1.0]`, where `1.0` is the
    highest possible confidence score.
    public var confidence Double

    /// Creates an annotation for an item an object detector finds in an
image.
    ///
    /// You don't use this initializer to create an object annotation yourself.
    /// The object detector uses it to create object annotations when it
makes predictions on your images, such as when you use
    /// ``MLObjectDetector/prediction(from:)``.

    ///
    /// - Parameters:
    ///   - label: The name of the item.
    ///   - boundingBox: The location of the item in an image.
    ///   - confidence: The confidence score of the item in the image.
The value must be in the range `[0.0, 1.0]`, where `1.0` is the most confident.
    public init String CGRect
                    Double

```

```

@available macOS 10.15
@available
@available
extension MLObjectDetector

```

```

    /// Locates objects in an image and generates an annotation for each object
it detects.
    ///
    /// - Parameters:
    ///   - image: The URL for the image file where you want the object
detector to look for objects.
    ///
    /// - Returns: An ``MLObjectDetector/DetectedObjects``  

instance — which is an ``MLObjectDetector/ObjectAnnotation``  

    /// array — where each annotation represents an item the object detector
found in the image.
    public func prediction URL throws
MLObjectDetector DetectedObjects

```

```

    /// Locates objects in an array of images and generates an array of
annotation collections, one for each input image.
    ///
    /// - Parameters:
    ///   - images: An array of URLs for the image files where you want the
object detector to look for objects.

```

```
    /**
     * - Returns: An ``MLObjectDetector/DetectedObjects`` array,
     where each element represents the annotations of the items the object detector
     found in the corresponding input image.
     */
    public func predictions URL throws
    MLObjectDetector DetectedObjects
```

```
@available macOS 10.15
```

```
@available
```

```
@available
```

```
extension MLObjectDetector
```

```
    /**
     * Parameters that affect the process of training an object detection model.
     */

```

```
    /**
     */

```

```
    /**
     * Customize the training process of an object detector by creating an
     ``MLObjectDetector/ModelParameters-swift.struct`` instance and
     passing it to an object detector's initializer. You can explicitly set values
     for
     ``MLObjectDetector/ModelParameters-swift.struct/maxIterations``
     
```

```
    /**
     * and
     ``MLObjectDetector/ModelParameters-swift.struct/batchSize``.
```

```
You can also explicitly define the validation dataset to override the
```

```
    /**
     * default behavior, which uses a random selection of your training dataset
     for validation.
```

```
public struct ModelParameters
```

```
    /**
     * The object detector's validation dataset for the training session.
```

```
    public var validation
```

```
MLObjectDetector ModelParameters ValidationData
```

```
    /**
     * The number of images the training session can use in a training
     iteration.
```

```
    /**
     */

```

```
    /**
     * If you don't have a preference, set this property to `nil` to tell
     Create ML to use an appropriate value when it trains the model.
```

```
    public var batchSize Int
```

```
    /**
     * The maximum number of iterations the training session can use.
```

```
    /**
     */

```

```
    /**
     * If you don't have a preference, set this property to `nil` to tell
     Create ML to use an appropriate value when it trains the model.
```

```
    public var maxIterations Int
```

```
    /**
     * The number of rectangles, vertically and horizontally, the training
     algorithm uses to analyze each input image.
```

```
@available macOS 11.0
```

```
@available
```

```
@available
```

```
public var gridSize CGSize
    /// The algorithm the training session uses to train the object detector.
    @available macos 11.0
    @available
    @available
    public var algorithm
MLObjectDetector ModelParameters ModelAlgorithmType

    /// Creates a model parameters instance for an object-detector training
session.
    /**
     /// - Parameters:
     /// - validation: An
``MLObjectDetector/ModelParameters-swift.struct/ValidationData
`` instance that contains your validation
    /// dataset.
    /**
     /// - batchSize: The number of images the object detector uses for
each training iteration. If you don't have a preference, set this parameter to `nil` to
    /// tell Create ML to use an appropriate value when it trains the model.
    /**
     /// - maxIterations: The largest number of training iterations the object
detector can use. If you don't have a preference, set this parameter to `nil` to
    /// tell Create ML to use an appropriate value when it trains the model.
    /**
     /// - gridSize: The number of rectangles, vertically and horizontally, the
training algorithm uses to analyze each input image.
    /**
     /// - algorithm: The algorithm the training session uses to train the
object detector.
    @available macos 11.0
    @available
    @available
    public init
MLObjectDetector ModelParameters ValidationData
                                Int      nil
                                Int      nil
                                CGSize
13          13
MLObjectDetector ModelParameters ModelAlgorithmType

    /// Creates a model parameters instance for an object-detector training
session set to use the full network algorithm.
    /**
     /// - Parameters:
     /// - validation: An
``MLObjectDetector/ModelParameters-swift.struct/ValidationData
`` instance that contains your validation
    /// dataset.
```

```
    /**
     * - batchSize: The number of images the object detector uses for
     * each training iteration. If you don't have a preference, set this parameter to `nil` to
     * tell Create ML to use an appropriate value when it trains the model.
     */
    /**
     * - maxIterations: The largest number of training iterations the object
     * detector can use. If you don't have a preference, set this parameter to `nil` to
     * tell Create ML to use an appropriate value when it trains the model.
     */
public init
MLObjectDetector ModelParameters ValidationData
    Int      nil
    Int      nil

    /**
     * Creates a model parameters instance for an object-detector training
     * session set to use the full network algorithm.
     */
    /**
     * - Parameters:
     * - validationData: An
``MLObjectDetector/ModelParameters-swift.struct/ValidationData
`` instance that contains your
    validation dataset.
    /**
     * - batchSize: The number of images the object detector uses for
     * each training iteration. If you don't have a preference, set this parameter to `nil` to
     * tell Create ML to use an appropriate value when it trains the model.
     */
    /**
     * - maxIterations: The largest number of training iterations the object
     * detector can use. If you don't have a preference, set this parameter to `nil` to
     * tell Create ML to use an appropriate value when it trains the model.
     */
@available                                     10.15          11.0
@available
@available
public init
MLObjectDetector DataSource
    Int      nil
    Int      nil throws

@available macOS 10.15
@available
@available
extension MLObjectDetector

    /**
     * The available types of image annotations.
     */
    /**
     * Use ``MLObjectDetector/AnnotationType`` to tell Create ML
     * how to interpret your object annotations.
     */
public enum AnnotationType   Sendable

    /**
     * An annotation type that defines a rectangle around an object within
```

an image.

```
///
/// To use bounding box annotations, you must tell Create ML how to
interpret your annotations.
///
/// - Use ``MLBoundingBoxUnits`` to specify the coordinate units
of your bounding boxes. - Use ``MLBoundingBoxAnchor`` to specify the
/// location in the bounding box its coordinates refer to. - Use
``MLBoundingBoxCoordinatesOrigin`` to specify the part of the image the
/// annotations use as their origin.
///
/// ## Formatting a Bounding Box JSON Annotation File
///
/// The base of the JSON file must contain an array of the following
JSON object structure.
///
/// | Name      | Type   | Value
/// |-----|-----|-----
/// | `imagefilename` | String | The name of the image's file.
/// | `annotation` | Array | An array of annotation JSON objects.
///
/// Each JSON object in the `annotation` array must have the
following JSON object structure.
///
/// | Name      | Type   | Value
|-----|
/// |-----|-----|
/// | `label` | String | The name of the annotated object.
|-----|
/// | `coordinates` | JSON object | The location of the object and the
area it occupies in the image.
///
/// The `coordinate` JSON object must have the following structure.
The origin of the image is the upper-left corner. The `x`-values increase from left
/// to right, and the `y`-values increase from the top to the bottom.
///
/// | Name      | Type   | Value
|-----|
/// |-----|-----|
/// | `x` | Number | The `x`-coordinate of the annotation's origin,
which `MLBoundingBoxCoordinatesOrigin` defines.
/// | `y` | Number | The `y`-coordinate of the annotation's origin,
which `MLBoundingBoxCoordinatesOrigin` defines.
/// | `width` | Number | The width of the annotation's bounding box.
|-----|
/// | `height` | Number | The height of the annotation's bounding box.
|-----|
/// As an example, the following JSON file has correct structure with
```

one image file (`"cat\_and\_dog.png"`) in its base array, which has two annotations.

```
///  
/// ` ```  
/// // JSON file  
/// [{  
///   "imagefilename": "cat_and_dog.png",  
///   "annotation": [  
///     {  
///       "label": "cat",  
///       "coordinates": {  
///         "y": 2.0,  
///         "x": 3.9,  
///         "height": 40.1,  
///         "width": 20.0  
///       }  
///     }, {  
///       "label": "dog",  
///       "coordinates": {  
///         "y": 40.0,  
///         "x": 38.9,  
///         "height": 100.1,  
///         "width": 70.0  
///       }  
///     }  
///   ],  
///   ... ]  
/// ` ```  
/// A typical annotation JSON file has many more objects in its base array, one for each image file.
```

```
case boundingBox      MLBoundingBoxUnits  
MLBoundingBoxCoordinatesOrigin  
MLBoundingBoxAnchor
```

## extension MLObjectDetector

```
/// Creates an object detector from a training session checkpoint.  
///  
/// - Parameters:  
///   - checkpoint: A checkpoint from an object-detector training session.  
@available macOS 11.0  
@available
```

```
@available
public init MLCheckpoint throws

    /// Begins an asynchronous object-detector training session.
    ///
    /// - Parameters:
    /// - trainingData: The annotated images the task uses to train the
object detector.
    ///
    /// - annotationType: The format your data source uses for its image
annotations.
    ///
    /// - parameters: An ``MLObjectDetector/ModelParameters-
swift.struct`` instance you use to set the model configuration settings for the
/// training session.
    ///
    /// - sessionParameters: An ``MLTrainingSessionParameters``
instance you use to configure the training session.
    ///
    /// - Returns: An ``MLJob`` that represents the object-detector training
session.

@available macOS 11.0
@available
@available
public static func train
MLObjectDetector DataSource
MLObjectDetector AnnotationType
MLObjectDetector ModelParameters
MLTrainingSessionParameters
throws
MLJob MLObjectDetector

    /// Creates an asynchronous object-detector training session.
    ///
    /// Use ``MLObjectDetector/resume(_:)`` to start the
``MLTrainingSession`` instance you get from this method.
    ///
    /// - Parameters:
    /// - trainingData: The annotated images the task uses to train the
object detector.
    ///
    /// - annotationType: The format type of the image annotations in the data
source.
    ///
    /// - parameters: An ``MLObjectDetector/ModelParameters-
swift.struct`` instance you use to set the model configuration settings for the
/// training session.
    ///
    /// - sessionParameters: An ``MLTrainingSessionParameters``
instance you use to configure the training session.
```

```
///  
/// - Returns: An ``MLTrainingSession`` that represents the object-  
detector training session.  
@available macOS 11.0  
@available  
@available  
public static func makeTrainingSession  
MLObjectDetector DataSource  
MLObjectDetector AnnotationType  
MLObjectDetector ModelParameters  
    MLTrainingSessionParameters  
    throws  
MLTrainingSession MLObjectDetector  
  
    /// Creates an asynchronous training session for an object detector by  
restoring an existing training session's state from its parameters.  
///  
/// Use ``MLObjectDetector/resume(_:)`` to start the  
``MLTrainingSession`` instance you get from this method.  
///  
/// - Parameters:  
/// - sessionParameters: The  
``MLTrainingSessionParameters`` instance you used to create the training  
session with  
///  
``MLObjectDetector/makeTrainingSession(trainingData:annotation  
Type:parameters:sessionParameters:)``.  
///  
/// - Returns: An ``MLTrainingSession`` that represents the object-  
detector training session.  
@available macOS 11.0  
@available  
@available  
public static func  
restoreTrainingSession  
MLTrainingSessionParameters throws  
MLTrainingSession MLObjectDetector  
  
    /// Begins or continues an asynchronous object-detector training session.  
///  
/// Use this method to start or resume a training session you get from  
///  
``MLObjectDetector/makeTrainingSession(trainingData:annotation  
Type:parameters:sessionParameters:)``  
    /// or  
``MLObjectDetector/restoreTrainingSession(sessionParameters:)``  
.  
///  
/// - Parameters:  
/// - session: An ``MLTrainingSession`` instance that represents
```

the training session.

```
///  
/// - Returns: An ``MLJob`` that represents the object-detector training  
session.  
@available macOS 11.0  
@available  
@available  
public static func resume _  
MLTrainingSession MLObjectDetector throws  
MLJob MLObjectDetector
```

**@available** macOS 10.15  
**@available**  
**@available**  
**extension** MLObjectDetector

```
/// Exports the object detector as a Core ML model file.  
///  
/// - Parameters:  
///   - fileURL: A file-system URL.  
///   - metadata: The model's description, author, version, and license  
information.  
public func write URL throws  
MLModelMetadata
```

/// Exports the object detector as a Core ML model file.  
///  
/// - Parameters:  
/// - path: A file-system path.  
/// - metadata: The model's description, author, version, and license  
information.  
**public func** write String **throws**  
MLModelMetadata

**@available** macOS 10.15  
**@available**  
**@available**  
**extension** MLObjectDetector CustomStringConvertible  
CustomDebugStringConvertible  
CustomPlaygroundDisplayConvertible

```
/// A text representation of the object detector.  
public var description String get
```

/// A text representation of the object detector that's suitable for output  
/// during debugging.  
**public var** debugDescription String **get**

```
/// A description of the object detector within a playground.  
public var playgroundDescription Any get  
  
@available macOS 10.15  
@available  
@available  
extension MLObjectDetector ObjectAnnotation  
CustomStringConvertible CustomDebugStringConvertible  
CustomPlaygroundDisplayConvertible  
  
/// A text representation of the object annotation.  
public var description String get  
  
/// A text representation of the object annotation that's suitable for  
/// output during debugging.  
public var debugDescription String get  
  
/// A description of the object annotation within a playground.  
public var playgroundDescription Any get
```

### **extension** MLObjectDetector ModelParameters

```
/// The underlying base model that extracts image features for an object-  
detector training session.  
@available macOS 11.0  
@available  
@available  
public enum FeatureExtractorType Sendable  
  
/// A feature extractor you use with a transfer-learning algorithm for  
object detectors.  
case objectPrint Int 1
```

### **available** macOS 10.15 @available @available **extension** MLObjectDetector ModelParameters

```
/// A validation dataset for an object detector.  
public enum ValidationData  
  
/// A validation dataset Create ML derives by randomly selecting a  
portion of the object detector's training dataset using the split strategy.  
///  
/// - Parameters:  
/// - strategy: An ``MLSplitStrategy`` instance the
```

enumeration case uses to select a portion of the object detector's training dataset as its

```
    /// associated value.  
    case split          MLSplitStrategy  
  
    /// A validation dataset you provide as a data source.  
    ///  
    /// - Parameters:  
    /// - dataSource : An ``MLObjectDetector/DataSource``  
instance the enumeration case uses as its associated value.  
    case dataSource MLObjectDetector DataSource  
  
    /// A validation dataset you provide as a data table.  
    ///  
    /// - Parameters:  
    /// - table : An ``MLDataTable`` instance the enumeration case  
uses as its associated value.  
    ///  
    /// - imageColumn: The name of the column in the data table that  
contains the image file URLs.  
    ///  
    /// - annotationColumn : The name of the column in the data table that  
contains the image annotations.  
    @available           10.15           14.0  
    @available  
    @available  
    case table MLDataTable           String  
                           String  
  
    /// Set validation data from the MLDataTable provided.  
    @available macOS 13.0  
    @available  
    @available  
    case DataFrame DataFrame           String  
                           String  
  
    /// An empty validation dataset that skips the model validation phase  
after training.  
    ///  
    /// Use this case when you don't have validation data while preventing  
Create ML from using any of your training dataset for validation.  
    case none
```

**extension** MLObjectDetector ModelParameters

```
    /// An object-detector training algorithm.  
    @available macOS 11.0  
    @available
```

```

@available
public enum ModelAlgorithmType : Equatable, Sendable

    /// An algorithm that trains a full neural network with your training data.
    ///
    /// Use this algorithm when your training dataset has a significant
number of examples.
    case darknetYolo

        /// An algorithm that leverages the knowledge of a general purpose
model built into the operating system.
        ///
        /// You typically use this transfer-learning algorithm to train an object
detector in these situations:
        ///
        /// - Your training dataset has a limited number of examples. - You
prefer your object detector's Core ML model file to be as small as possible.
    case
transferLearning MLObjectDetector ModelParameters FeatureExtra
ctorType

    /// Returns a Boolean value that indicates whether two algorithm are
equal.
    ///
    /// - Parameters:
    /// - lhs: An algorithm instance.
    ///
    /// - rhs: Another algorithm instance.
public static func
MLObjectDetector ModelParameters ModelAlgorithmType
MLObjectDetector ModelParameters ModelAlgorithmType Bool

```

```

@available macOS 10.15
@available
@available
extension MLObjectDetector ModelParameters
CustomStringConvertible CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of the model parameters.
    public var description : String { get }

    /// A text representation of the model parameters that's suitable for output
during debugging.
    public var debugDescription : String { get }

    /// A description of the model parameters within a playground.
    public var playgroundDescription : Any { get }

```

```

/// Metrics you use to evaluate an object detector's performance.
///
/// An object detector generates intersection-over-union (IoU) metrics, which is
/// a way to measure the similarity of two bounding boxes. The IoU metric is the
/// overlapping area divided by the area of the union of the bounding boxes.
///
/// For example, two bounding boxes that overlap perfectly have an IoU of `1.0`,
/// because their overlap is the same area as the union. Two bounding boxes that
/// have no overlap have an IoU of `0.0`. Anything between `0.0` and `1.0`
/// either means the two bounding boxes partially overlap or one box completely
/// encases the other.
@available(macOS 10.15)
@available
@available
public struct MLObjectDetectorMetrics Sendable

    /// Two dictionaries of average precisions at different thresholds.
    ///
    /// - Parameters:
    ///   - variedIoU: The average precision values of all classes at various
    ///     thresholds, varying from 50% to 95%, for the intersection-over-union
    ///     metric. The keys of the dictionary are each object's label.
    ///
    ///   - IoU50: The average precision values of all classes at a 50%
threshold
    ///     of intersection-over-union metric. The keys of the dictionary are each
    ///     object's label.
    public var averagePrecision String
Double String Double get

    /// Two mean-average precisions at different thresholds.
    ///
    /// - Parameters:
    ///   - variedIoU: The mean of the average precision values across all
classes
    ///     at various thresholds, varying from 50% to 95%, of the
    ///     intersection-over-union metric.
    ///
    ///   - IoU50: The mean of the average precision values across all
classes at
    ///     a 50% threshold of intersection-over-union metric.
    public var meanAveragePrecision Double
Double get

    /// Creates metrics for an object detector given an average precision and a
    /// mean average precision.
    ///
    /// You do not use this initializer. Create ML uses this initializer to

```

```
    /// generate metrics for you when train an object detector or when you use
    /// an evaluation method.
    ///
    /// - Parameters:
    ///   - averagePrecision: The `averagePrecision` for this
    `MLObjectDetectorMetrics`.
    ///
    ///   - meanAveragePrecision: The `meanAveragePrecision` for
    this `MLObjectDetectorMetrics`.
    public init
    Double      String    Double
                  Double      Double
   String

    /// The underlying error present when the metrics are invalid.
    public var error  Any Error    get

    /// A Boolean value indicating whether the object detector model was able to
    /// calculate metrics.
    ///
    /// Your metrics may be invalid if you attempt to perform evaluation on
    /// images with annotations that don't match the annotations of your
    /// training examples.
    public var isValid Bool     get

@available macOS 10.15
@available
@available
extension MLObjectDetectorMetrics : CustomStringConvertible
CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of the object detector metrics.
    public var description String    get

    /// A text representation of the object detector metrics that's suitable for
    /// output during debugging.
    public var debugDescription String    get

    /// A description of the object detector metrics shown in a playground.
    public var playgroundDescription Any     get

    /// The possible states of a training session.
@available macOS 11.0 iOS 15.0 tvOS 16.0
public enum MLPhase String Codable Sendable

    /// The training session is in the initial idle state before extracting
    /// features and training.
```

```
case initialized

/// The training session is extracting features from the training dataset.
case extractingFeatures

/// The training session is training a model from the features it extracted
/// from the training dataset.
case training

/// The training session is evaluating the model it trained.
case evaluating

/// The training session is using the model to make a prediction.
case inferencing

/// Creates a new instance with the specified raw value.
///
/// If there is no value of the type that corresponds with the specified raw
/// value, this initializer returns `nil`. For example:
///
///     enum PaperSize: String {
///         case A4, A5, Letter, Legal
///     }
///
///     print(PaperSize(rawValue: "Legal"))
///     // Prints "Optional("PaperSize.Legal")"
///
///     print(PaperSize(rawValue: "Tabloid"))
///     // Prints "nil"
///
/// - Parameter rawValue: The raw value to use for the new instance.
public init String

/// The raw type that can be used to represent all values of the conforming
/// type.
///
/// Every distinct value of the conforming type has a corresponding unique
/// value of the `RawValue` type, but there may be values of the
`RawValue`
/// type that don't have a corresponding value of the conforming type.
@available iOS 15.0 tvOS 16.0 macOS 11.0
public typealias RawValue String

/// The corresponding value of the raw type.
///
/// A new instance initialized with `rawValue` will be equivalent to this
/// instance. For example:
///
///     enum PaperSize: String {
```

```
    /**
     *      case A4, A5, Letter, Legal
     */
    /**
     *      let selectedSize = PaperSize.Letter
     *      print(selectedSize.rawValue)
     *      // Prints "Letter"
     */
    /**
     *      print(selectedSize == PaperSize(rawValue:
     selectedSize.rawValue)!)
     *      // Prints "true"
public var rawValue String get
```

```
@available macOS 11.0 iOS 15.0 tvOS 16.0
extension MLPhase Equatable
```

```
@available macOS 11.0 iOS 15.0 tvOS 16.0
extension MLPhase Hashable
```

```
@available macOS 11.0 iOS 15.0 tvOS 16.0
extension MLPhase RawRepresentable
```

```
    /**
     * A convenience type that exposes information about the progress of a training
     * session.
     */

```

```
    /**
     * Create ML uses this type to exposes specific values within a _Progress_
     * instance as properties.
     */

```

```
@available macOS 11.0 iOS 15.0 tvOS 16.0
public struct MLProgress Codable
```

```
    /**
     * The time, in seconds, since the training session started.
     */
public var elapsedTime TimeInterval
```

```
    /**
     * The current phase of the training session.
     */
public var phase MLPhase
```

```
    /**
     * The current number of files processed during a feature extraction phase,
     * or the completed iterations during a training phase.
     */

```

```
public var itemCount Int
```

```
    /**
     * The total number of files during a feature extraction phase, or total
     * iterations during a training phase.
     */

```

```
public var totalItemCount Int
```

```
    /**
     * Measurements of the model's performance during the training or
     * evaluation phases of a training session.
     */

```

```
public var metrics    MLProgress Metric    Any
    /// Creates a training session progress instance from a training phase.
    ///
    /// - Parameters:
    ///   - phase: A training session's phase.
public init        MLPhase

    /// Creates a training session progress instance from a foundation progress
    /// object.
    ///
    /// - Parameters:
    ///   - progress: A foundation progress object.
public init        Progress

    /// Metrics you use to evaluate a model's performance during a training
    /// session.
public enum Metric    String    Codable    Hashable
CaseIterable    Sendable

    /// The metric for the model's loss.
    case loss

    /// The metric for the style transfer model's content loss.
    case contentLoss

    /// The metric for the style transfer model's style loss.
    case styleLoss

    /// The metric for the model's accuracy.
    case accuracy

    /// The metric for the model's validation loss.
    case validationLoss

    /// The metric for the model's validation accuracy.
    case validationAccuracy

    /// The location of the stylized image content in the file system.
    case stylizedImageURL

    /// The metric for the model's root mean squared error (RMSE).
@available macOS 14.0  iOS 17.0  tvOS 17.0
    case rootMeanSquaredError

    /// The metric for the model's maximum error.
@available macOS 14.0  iOS 17.0  tvOS 17.0
    case maximumError
```

```
    /// The metric for the model's validation root mean squared error
(RMSE).
@available macOS 14.0 iOS 17.0 tvOS 17.0
case validationRootMeanSquaredError

    /// The metric for the model's validation maximum error.
@available macOS 14.0 iOS 17.0 tvOS 17.0
case validationMaximumError

    /// A collection of all values of this type.
public static var allCases MLProgress Metric
```

**get**

```
    /// Creates a new instance with the specified raw value.
    ///
    /// If there is no value of the type that corresponds with the specified
raw
    /// value, this initializer returns `nil`. For example:
    ///
    ///     enum PaperSize: String {
    ///         case A4, A5, Letter, Legal
    ///     }
    ///
    ///     print(PaperSize(rawValue: "Legal"))
    ///     // Prints "Optional("PaperSize.Legal")"
    ///
    ///     print(PaperSize(rawValue: "Tabloid"))
    ///     // Prints "nil"
    ///
    /// - Parameter rawValue: The raw value to use for the new
instance.
```

**public init String**

```
    /// A type that can represent a collection of all values of this type.
@available iOS 15.0 tvOS 16.0 macOS 11.0
public typealias AllCases MLProgress Metric
```

**conforming**

```
    /// The raw type that can be used to represent all values of the
unique
    /// type.
    ///
    /// Every distinct value of the conforming type has a corresponding
`RawValue`
    /// value of the `RawValue` type, but there may be values of the
    /// type that don't have a corresponding value of the conforming type.
@available iOS 15.0 tvOS 16.0 macOS 11.0
public typealias RawValue String
```

**/// The corresponding value of the raw type.**

```
///  
/// A new instance initialized with `rawValue` will be equivalent to this  
/// instance. For example:  
///  
///     enum PaperSize: String {  
///         case A4, A5, Letter, Legal  
///     }  
///  
///     let selectedSize = PaperSize.Letter  
///     print(selectedSize.rawValue)  
///     // Prints "Letter"  
///  
///     print(selectedSize == PaperSize(rawValue:  
selectedSize.rawValue)!)  
///     // Prints "true"  
public var rawValue String get
```

```
/// Creates a progress instance by decoding from the given decoder.
```

```
///  
/// - Parameters:  
///   - decoder: The decoder to read data from.  
public init any Decoder throws
```

```
/// Encodes the progress value into the given encoder.
```

```
///  
/// - Parameters:  
///   - encoder: The encoder to write data to.  
public func encode any Encoder throws
```

```
@available macOS 11.0 iOS 15.0 tvOS 16.0  
extension MLProgress
```

```
/// The key that accesses the elapsed time value.
```

```
public static let elapsedTimeKey ProgressUserInfoKey
```

```
/// The key that accesses the current phase value.
```

```
public static let phaseKey ProgressUserInfoKey
```

```
/// The key that accesses the current item count value.
```

```
public static let itemCountKey ProgressUserInfoKey
```

```
/// The key that accesses the total item count value.
```

```
public static let totalItemCountKey ProgressUserInfoKey
```

```
/// The key that accesses the training loss value.
```

```
public static let lossKey ProgressUserInfoKey
```

```
    /// The key that accesses the content image loss value.  
    public static let contentLossKey ProgressUserInfoKey  
  
    /// The key that accesses the style image loss value.  
    public static let styleLossKey ProgressUserInfoKey  
  
    /// The key that accesses the training accuracy value.  
    public static let accuracyKey ProgressUserInfoKey  
  
    /// The key that accesses the validation loss value.  
    public static let validationLossKey ProgressUserInfoKey  
  
    /// The key that accesses the validation accuracy value.  
    public static let validationAccuracyKey  
ProgressUserInfoKey  
  
    /// The key that accesses the stylized image value.  
    public static let stylizedImageKey ProgressUserInfoKey  
  
    /// They key that accesses the root-mean-squared error value.  
    @available macOS 14.0 iOS 17.0 tvOS 17.0  
    public static let rootMeanSquaredErrorKey  
ProgressUserInfoKey  
  
    /// They key that accesses the maximum error value.  
    @available macOS 14.0 iOS 17.0 tvOS 17.0  
    public static let maximumErrorKey ProgressUserInfoKey  
  
    /// They key that accesses the validation root-mean-squared error value.  
    @available macOS 14.0 iOS 17.0 tvOS 17.0  
    public static let validationRootMeanSquaredErrorKey  
ProgressUserInfoKey  
  
    /// They key that accesses the validation maximum error value.  
    @available macOS 14.0 iOS 17.0 tvOS 17.0  
    public static let validationMaximumErrorKey  
ProgressUserInfoKey  
  
@available macOS 11.0 iOS 15.0 tvOS 16.0  
extension MLProgress Metric RawRepresentable  
  
    /// A classifier based on a collection of decision trees trained on subsets of the  
    data.  
    @available macOS 10.14 iOS 15.0 tvOS 16.0  
    public struct MLRandomForestClassifier @unchecked Sendable  
  
    /// The Core ML model.
```

```
public var model MLModel
    /// The name of the column you selected at initialization to define which categories the classifier predicts.
    ///
    /// Changing the value of this property doesn't retrain the model or affect its behavior.
public var targetColumn String
    /// The names of the columns you selected at initialization to train the classifier.
    ///
    /// Changing the value of this property doesn't retrain the model or affect its behavior.
public var featureColumns String
    /// The underlying parameters used when training the model.
public let modelParameters
MLRandomForestClassifier ModelParameters
    /// Measurements of the classifier's performance on the training data set.
public var trainingMetrics MLClassifierMetrics get
    /// Measurements of the classifier's performance on the validation data set.
public var validationMetrics MLClassifierMetrics get
    /// Creates a random forest classifier.
    ///
    /// - Parameters:
    /// - trainingData: The training data
    /// - targetColumn: Name of the column containing the class labels
    /// - featureColumns: Names of the columns containing feature values. If `nil` all columns, other than the target
    /// column, will be used as feature values.
    /// - parameters: Model training parameters
@available macOS 12.0 iOS 15.0 tvOS 16.0
public init DataFrame String
    String nil
MLRandomForestClassifier ModelParameters

throws

    /// Creates a Random Forest Classifier from the feature columns in the training data to predict the categories in
    /// the target column.
    ///
    /// - Parameters:
    /// - trainingData: A data table of training examples.
    /// - targetColumn: The column name for the values in the training data that the classifier should predict.
```

```

    /// - featureColumns: The column names for the values in the
    // training data that the classifier uses to predict
    /// the target value.
    /// - parameters: The model parameters.
    @available(10.14, 13.0)
        "Use DataFrame instead of MLDataTable when
initializing."
    @available(15.0, 16.0)
        "Use DataFrame instead of MLDataTable when
initializing."
    @available
    public init MLDataTable
String String nil
MLRandomForestClassifier ModelParameters
nil throws

    /// Creates a random forest classifier classifier from a checkpoint.
    ///
    /// - Parameter checkpoint: Training checkpoint.
    /// - Throws: `MLCreateError` if the checkpoint can't be loaded.
    @available(macOS 12.0, iOS 15.0, tvOS 16.0)
    public init MLCheckpoint throws

    /// Trains a random forest classifier.
    ///
    /// If `sessionDirectory` is provided it will save training progress. If
    // there is progress already saved training
    /// will resume from the last checkpoint.
    ///
    /// - Parameters:
    ///   - trainingData: A DataTable specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
    // trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
    // columns to be
    ///   used to predict the target, if not provided,
    // default to use all the
    ///   other columns in the trainingData, except the
    // one specified by targetColumn
    ///   - parameters: Model training parameters. See
    `MLRandomForestClassifier.ModelParameters` for the defaults.
    ///   - sessionParameters: Training session parameters. See
    `MLTrainingSessionParameters` for the defaults.
    ///
    /// - Returns: A `MLJob` that can be used to observe training progress.
    @available(12.0, 14.0)
    @available(15.0, 17.0)
    @available(16.0, 17.0)
    public static func train MLDataTable
String String nil

```

## MLRandomForestClassifier ModelParameters

```
MLTrainingSessionParameters
throws MLJob MLRandomForestClassifier

    /// Trains a random forest classifier.
    ///
    /// If `sessionDirectory` is provided it will save training progress. If
    there is progress already saved training
    /// will resume from the last checkpoint.
    ///
    /// - Parameters:
    ///   - trainingData: A `DataFrame` specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
    trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
    columns to be
        ///                               used to predict the target, if not provided,
    default to use all the
        ///                               other columns in the trainingData, except the
    one specified by targetColumn
    ///   - parameters: Model training parameters. See
    `MLRandomForestClassifier.ModelParameters` for the defaults.
    ///   - sessionParameters: Training session parameters. See
    `MLTrainingSessionParameters` for the defaults.
    ///
    /// - Returns: A `MLJob` that can be used to observe training progress.
@available macOS 12.0 iOS 15.0 tvOS 16.0
public static func train
    String           DataFrame
    String           nil
    MLRandomForestClassifier ModelParameters
```

```
MLTrainingSessionParameters
throws MLJob MLRandomForestClassifier
```

```
    /// Creates or restores a training session.
    ///
    /// - Parameters:
    ///   - trainingData: A DataTable specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
    trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
    columns to be
        ///                               used to predict the target, if not provided, default to use
    all the
        ///                               other columns in the trainingData, except the one
    specified by targetColumn
    ///   - parameters: Model training parameters. See
    `MLRandomForestClassifier.ModelParameters` for the defaults.
    ///   - sessionParameters: Training session parameters. See
    `MLTrainingSessionParameters` for the defaults.
```

```

    /**
     * - Returns: A `MLTrainingSession` that can be used to start or
     * resume training.
     */
    @available(macOS 12.0, iOS 15.0, tvOS 16.0)
    public static func makeTrainingSession(
        MLDDataTable? nil,
        MLRandomForestClassifier? nil,
        ModelParameters? nil,
        MLTrainingSessionParameters? nil
    ) throws MLTrainingSession, MLRandomForestClassifier

    /**
     * Creates or restores a training session.
     */
    /**
     * - Parameters:
     *   - trainingData: A `DataFrame` specifying training data.
     *   - targetColumn: A String specifying the target column name in the
     *     trainingData
     *   - featureColumns: An optional list of Strings specifying feature
     *     columns to be
     *     used to predict the target, if not provided, default to use
     *     all the
     *     other columns in the trainingData, except the one
     *     specified by targetColumn
     *   - parameters: Model training parameters. See
     *     `MLRandomForestClassifier.ModelParameters` for the defaults.
     *   - sessionParameters: Training session parameters. See
     *     `MLTrainingSessionParameters` for the defaults.
     */
    /**
     * - Returns: A `MLTrainingSession` that can be used to start or
     * resume training.
     */
    @available(macOS 12.0, iOS 15.0, tvOS 16.0)
    public static func makeTrainingSession(
        DataFrame? nil,
        String? nil,
        String? nil,
        ModelParameters? nil,
        MLTrainingSessionParameters? nil
    ) throws MLTrainingSession, MLRandomForestClassifier

    /**
     * Restores an existing training session.
     */
    /**
     * - Parameters:
     *   - sessionParameters: Training session parameters. The
     *     `sessionDirectory` parameter is required.
     */
    /**
     * - Returns: A `MLTrainingSession` that can be used to resume
     * training.
     */
    @available(macOS 12.0, iOS 15.0, tvOS 16.0)
    public static func

```

```
restoreTrainingSession
MLTrainingSessionParameters  throws
MLTrainingSession MLRandomForestClassifier

    /// Resumes a training session from the last checkpoint if available.
    ///
    /// If there are no resumable checkpoints training starts over from the
beginning.
    ///
    /// - Parameter session: Loaded or new training session.
    ///
    /// - Returns: A `MLJob` that can be used to observe training progress.
    @available macOS 12.0  iOS 15.0  tvOS 16.0
    public static func resume -
MLTrainingSession MLRandomForestClassifier  throws
MLJob MLRandomForestClassifier

    /// Predicts a column of labels for the given testing data.
    @available macOS 12.0  iOS 15.0  tvOS 16.0
    public func predictions           DataFrame  throws
AnyColumn

    /// Classifies the provided data into the target categories.
    ///
    /// - Parameters:
    ///   - data: The data you want the model to classify.
    ///
    /// - Returns: A column of labels predicted by the classifier.
    @available                         10.14          13.0
    "Use DataFrame instead of MLDataTable."
    @available                         15.0           16.0
    "Use DataFrame instead of MLDataTable."
    @available
    public func predictions           MLDataTable  throws
MLUntypedColumn

    /// Evaluates the classifier on the provided labeled data.
    ///
    /// Evaluation should be done on a testing data set that the model has not
seen as part of the training or
    /// validation data sets. The data should have feature columns with identical
name and type to the
    /// training data, as well as a labels column with the same name.
    ///
    /// - Parameters:
    ///   - labeledData: A `DataFrame` to evaluate the trained model on.
    ///
    /// - Returns: Metrics that describe the classification errors
    /// (`MLClassifierMetrics/classificationError`), the precision
and recall
```

```

    /// percentages(``MLClassifierMetrics/precisionRecall``), and
    a table that
    /// describes how labels were misapplied
    (``MLClassifierMetrics/confusion``)
    /// on the provided data.
    @available macOS 12.0  iOS 15.0  tvOS 16.0
    public func evaluation           DataFrame
MLClassifierMetrics

    /// Evaluates the classifier on the provided labeled data.
    ///
    /// Evaluation should be done on a testing data set that the model has not
    seen as part of the training or
    /// validation data sets. The data should have feature columns with identical
    name and type to the
    /// training data, as well as a labels column with the same name.
    ///
    /// - Parameters:
    ///   - labeledData: An `MLDataTable` to evaluate the trained model
    on.
    ///
    /// - Returns: Metrics that describe the classification errors
    /// (``MLClassifierMetrics/classificationError``), the precision
    and recall
    /// percentages(``MLClassifierMetrics/precisionRecall``), and
    a table that
    /// describes how labels were misapplied
    (``MLClassifierMetrics/confusion``)
    /// on the provided data.
    @available                         10.14          13.0
        "Use DataFrame instead of MLDataTable."
    @available                         15.0          16.0
        "Use DataFrame instead of MLDataTable."
    @available
    public func evaluation           MLDataTable
MLClassifierMetrics

    /// Exports a Core ML model file for use in your app.
    public func write                  URL
MLModelMetadata    nil  throws

    /// Exports a Core ML model file for use in your app.
    public func write                  String
MLModelMetadata    nil  throws

extension MLRandomForestClassifier

    /// Parameters that affect the process of training a model.
    @available macOS 10.14  iOS 15.0  tvOS 16.0

```

```

public struct ModelParameters

    /// Validation data represented as a `MLDataTable`.
    ///
    /// - Note: Setting this to `nil` means that the training data will be
automatically split for
    /// validation. Setting it to an empty table means to not use a
validation set.
    @available 10.14
10.15 "Use the validation property instead."
    @available 15.0 16.0 "Use the validation property instead."
    @available
    public var validationData MLDataTable

    /// Validation data.
    ///
    /// The default is `split(strategy: .automatic)`, which
automatically generates the validation
    /// dataset from 0% to 10% of the training dataset.
    @available macOS 10.15 iOS 15.0 tvOS 16.0
    public var validation
MLRandomForestClassifier ModelParameters ValidationData

    public var maxDepth Int
    public var maxIterations Int
    public var minLossReduction Double
    public var minChildWeight Double
    public var randomSeed Int

    /// Must be in the range (0, 1).
    public var rowSubsample Double

    /// Must be in the range (0, 1).
    public var columnSubsample Double

    @available macOS 10.15 iOS 15.0 tvOS 16.0
    public init
MLRandomForestClassifier ModelParameters ValidationData
    Int 6
    Int 10 Double 0
    Double 0.1 Int 42
    Double 0.8 Double 0.8

    /// Creates a new set of parameters.
    ///

```

```

    /**
     * - Parameters:
     *   - validationData: The dataset used to monitor how well the
     *     model is generalizing.
     */
    /**
     *   The default value is `nil` which will use an automatically
     *     sampled validation set.
     */
    /**
     *   - maxDepth: The maximum depth of the tree. Must be a value
     *     of at least 1.
     */
    /**
     *   The default value is 6.
     */
    /**
     *   - maxIterations: The maximum number of passes through
     *     the data.
     */
    /**
     *   The default value is 10.
     */
    /**
     *   - minLossReduction: The minimum amount of reduction in
     *     the loss function that is required to make another
     *       split to the data. Larger values help prevent overfitting.
     */
    /**
     *   The default value is 0.
     */
    /**
     *   - minChildWeight: Determines the minimum weight of each
     *     leaf node of the tree. Larger values help prevent
     *       overfitting.
     */
    /**
     *   The default value is 0.1.
     */
    /**
     *   - randomSeed: A seed for internal random operations. Set this
     *     value to ensure reproducible results.
     */
    /**
     *   The default value is 42.
     */
    /**
     *   - rowSubsample: Select the specified ratio from the training
     *     set to grow each tree. This technique is
     *       known as bagging.
     */
    /**
     *   The default value is 0.8.
     */
    /**
     *   - columnSubsample: Select the specified ratio of columns
     *     from the training set to use when growing each
     *       tree. Similar to row subsampling, this can be used to prevent
     *     overfitting.
     */
    /**
     *   The default value is 0.8
@available           10.14
10.15      "Use the validation property instead."
@available           15.0          16.0
      "Use the validation property instead."
@available

```

```
    public init
```

|       |            |             |
|-------|------------|-------------|
| Int 6 | Int 10     | MLDataTable |
| 0     | Double 0.1 | Double      |
|       | Double 0.8 | Int 42      |
|       |            | Double 0.8  |

```
@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLRandomForestClassifier : CustomStringConvertible
: CustomDebugStringConvertible
: CustomPlaygroundDisplayConvertible

    /// A text representation of the random forest classifier.
    public var description String get

    /// A text representation of the random forest classifier that's suitable
    /// for output during debugging.
    public var debugDescription String get

    /// A description of the random forest classifier shown in a playground.
    public var playgroundDescription Any get

extension MLRandomForestClassifier ModelParameters

    /// Values for specifying validation data.
    @available macOS 10.15 iOS 15.0 tvOS 16.0
    public enum ValidationData

        /// Generate validation data by splitting the training dataset. This is the
        default.
        case split MLSplitStrategy

        /// Set validation data from the MLDataTable provided.
        @available 10.15 14.0
        @available 15.0 17.0
        @available 16.0 17.0
        case table MLDataTable

        /// Validation data provided in a DataFrame.
        @available macOS 12.0 iOS 15.0 tvOS 16.0
        case dataFrame DataFrame

        /// Do not set validation data.
        case none

@available macOS 10.14 iOS 15.0 tvOS 16.0
```

```
extension MLRandomForestClassifier ModelParameters
CustomStringConvertible CustomDebugStringConvertible
CustomPlaygroundConvertible

    /// A text representation of the model parameters for a random forest
classifier.
public var description String get

    /// A text representation of the model parameters for a random forest
classifier that's suitable for output during
/// debugging.
public var debugDescription String get

    /// A description of the model parameters for a random forest classifier
shown in a playground.
public var playgroundDescription Any get

/// A regressor based on a collection of decision trees trained on subsets of the
data.
@available macOS 10.14 iOS 15.0 tvOS 16.0
public struct MLRandomForestRegressor @unchecked Sendable

    /// The Core ML model.
public var model MLModel

    /// The name of the column you selected at initialization to define which
feature the regressor predicts.
///
/// Changing the value of this property doesn't retrain the model or affect its
behavior.
public var targetColumn String

    /// The names of the columns you selected at initialization to train the
regressor.
///
/// Changing the value of this property doesn't retrain the model or affect its
behavior.
public var featureColumns String

    /// The underlying parameters used when training the model.
public let modelParameters
MLRandomForestRegressor ModelParameters

    /// Measurements of the regressor's performance on the training data set.
public var trainingMetrics MLRegressorMetrics get

    /// Measurements of the regressor's performance on the validation data set.
public var validationMetrics MLRegressorMetrics get
```

```

    /// Creates a random tree regressor.
    ///
    /// - Parameters:
    ///   - trainingData: The training data
    ///   - targetColumn: Name of the column containing the target values
    ///   - featureColumns: Names of the columns containing feature
    values. If `nil` all columns, other than the target
    ///   column, will be used as feature values.
    ///   - parameters: Model training parameters
    @available macOS 12.0 iOS 15.0 tvOS 16.0
    public init           DataFrame          String
                           String      nil
    MLRandomForestRegressor ModelParameters

throws

    /// Creates a Random Forest Regressor from the feature columns in the
    training data to predict the values in the
    /// target column.
    ///
    /// - Parameters:
    ///   - trainingData: A data table of training examples.
    ///   - targetColumn: The column name for the values in the training
    data the regressor should predict.
    ///   - featureColumns: The column names for the values in the
    training data that the regressor uses to predict
    ///   the target value.
    ///   - parameters: The model parameters.
    @available                      10.14                  13.0
        "Use DataFrame instead of MLDataTable when
initializing."
    @available                      15.0                  16.0
        "Use DataFrame instead of MLDataTable when
initializing."
    @available
    public init           MLDataTable
                           String      nil
    MLRandomForestRegressor ModelParameters
                           nil    throws

    /// Creates a random forest regressor from a checkpoint.
    ///
    /// - Parameter checkpoint: Training checkpoint.
    /// - Throws: `MLCreateError` if the checkpoint can't be loaded.
    @available macOS 12.0 iOS 15.0 tvOS 16.0
    public init           MLCheckpoint  throws

    /// Trains a random forest regressor.
    ///
    /// If `sessionDirectory` is provided it will save training progress. If

```

```

there is progress already saved training
    /// will resume from the last checkpoint.
    ///
    /// - Parameters:
    ///   - trainingData: A DataTable specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
  trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
  columns to be
    ///                               used to predict the target, if not provided,
  default to use all the
    ///                               other columns in the trainingData, except the
  one specified by targetColumn
    ///   - parameters: Model training parameters. See
`MLRandomForestRegressor.ModelParameters` for the defaults.
    ///   - sessionParameters: Training session parameters. See
`MLTrainingSessionParameters` for the defaults.
    ///
    /// - Returns: A MLJob that can be used to observe training progress.
@available           12.0          14.0
@available           15.0          17.0
@available           16.0          17.0
public static func train           MLDataTable
  String                String      nil
  MLRandomForestRegressor ModelParameters

  MLTrainingSessionParameters
  throws
MLJob MLRandomForestRegressor

    /// Trains a random forest regressor.
    ///
    /// If sessionDirectory is provided it will save training progress. If
  there is progress already saved training
    /// will resume from the last checkpoint.
    ///
    /// - Parameters:
    ///   - trainingData: A DataFrame specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
  trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
  columns to be
    ///                               used to predict the target, if not provided,
  default to use all the
    ///                               other columns in the trainingData, except the
  one specified by targetColumn
    ///   - parameters: Model training parameters. See
`MLRandomForestRegressor.ModelParameters` for the defaults.
    ///   - sessionParameters: Training session parameters. See
`MLTrainingSessionParameters` for the defaults.
    ///

```

```

    /// - Returns: A `MLJob` that can be used to observe training progress.
    @available macos 12.0  iOS 15.0  tvOS 16.0
    public static func train                DataFrame
        String           String      nil
        MLRandomForestRegressor ModelParameters

        MLTrainingSessionParameters
        throws
    MLJob MLRandomForestRegressor

    /// Creates or restores a training session.
    ///
    /// - Parameters:
    ///   - trainingData: A DataTable specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
    trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
    columns to be
        ///           used to predict the target, if not provided, default to use
    all the
        ///           other columns in the trainingData, except the one
    specified by targetColumn
    ///   - parameters: Model training parameters. See
    `MLRandomForestRegressor.ModelParameters` for the defaults.
    ///   - sessionParameters: Training session parameters. See
    `MLTrainingSessionParameters` for the defaults.
    ///
    /// - Returns: A `MLTrainingSession` that can be used to start or
    resume training.
    @available             12.0          14.0
    @available             15.0          17.0
    @available             16.0          17.0
    public static func makeTrainingSession
    MLDataTable            String         String
    nil                  MLRandomForestRegressor ModelParameters

        MLTrainingSessionParameters
        throws
    MLTrainingSession MLRandomForestRegressor

    /// Creates or restores a training session.
    ///
    /// - Parameters:
    ///   - trainingData: A `DataFrame` specifying training data.
    ///   - targetColumn: A String specifying the target column name in the
    trainingData
    ///   - featureColumns: An optional list of Strings specifying feature
    columns to be
        ///           used to predict the target, if not provided, default to use
    all the

```

```
    /**
     * - parameters: Model training parameters. See
     * `MLRandomForestRegressor.ModelParameters` for the defaults.
     * - sessionParameters: Training session parameters. See
     * `MLTrainingSessionParameters` for the defaults.
     */
    /**
     * - Returns: A `MLTrainingSession` that can be used to start or
     * resume training.
     *
     * @available macOS 12.0 iOS 15.0 tvOS 16.0
     * public static func makeTrainingSession(
     *   DataFrame           String          String
     *   nil                MLRandomForestRegressor ModelParameters
     *
     *   MLTrainingSessionParameters
     *   throws
     * ) MLTrainingSession MLRandomForestRegressor
     */
    /**
     * Restores an existing training session.
     */
    /**
     * - Parameters:
     * - sessionParameters: Training session parameters. The
     * `sessionDirectory` parameter is required.
     */
    /**
     * - Returns: A `MLTrainingSession` that can be used to resume
     * training.
     *
     * @available macOS 12.0 iOS 15.0 tvOS 16.0
     * public static func
     * restoreTrainingSession(
     *   MLTrainingSessionParameters throws
     * ) MLTrainingSession MLRandomForestRegressor
     */
    /**
     * Resumes a training session from the last checkpoint if available.
     */
    /**
     * If there are no resumable checkpoints training starts over from the
     * beginning.
     */
    /**
     * - Parameter session: Loaded or new training session.
     */
    /**
     * - Returns: A `MLJob` that can be used to observe training progress.
     *
     * @available macOS 12.0 iOS 15.0 tvOS 16.0
     * public static func resume(
     *   MLTrainingSession MLRandomForestRegressor throws
     * ) MLJob MLRandomForestRegressor
     */
    /**
     * Predicts a column of labels for the given testing data.
     *
     * @available macOS 12.0 iOS 15.0 tvOS 16.0
     * public func predictions DataFrame throws
     * AnyColumn
```

```

    /// Predicts the target value from the provided data.
    ///
    /// - Parameters:
    ///   - data: The data you want the model to make predictions from.
    ///
    /// - Returns: A column of values predicted by the regressor.
    @available(macOS 13.0, iOS 16.0, tvOS 16.0)
    public func predictions(MLUntypedColumn) throws DataFrame
}

/// Evaluates the classifier on the provided labeled data.
///
/// Evaluation should be done on a testing data set that the model has not
seen as part of the training or
/// validation data sets. The data should have feature columns with identical
name and type to the
/// training data, as well as a labels column with the same name.
///
/// - Parameters:
///   - labeledData: A `DataFrame` to evaluate the trained model on.
///
/// - Returns: Metrics that describe the maximum error
/// (`MLRegressorMetrics/maximumError`) or the average error
/// (`MLRegressorMetrics/rootMeanSquaredError`).
@available(macOS 12.0, iOS 15.0, tvOS 16.0)
public func evaluation() throws DataFrame
MLRegressorMetrics

/// Evaluates the classifier on the provided labeled data.
///
/// Evaluation should be done on a testing data set that the model has not
seen as part of the training or
/// validation data sets. The data should have feature columns with identical
name and type to the
/// training data, as well as a labels column with the same name.
///
/// - Parameters:
///   - labeledData: An `MLDataTable` to evaluate the trained model
on.
///
/// - Returns: Metrics that describe the maximum error
/// (`MLRegressorMetrics/maximumError`) or the average error
/// (`MLRegressorMetrics/rootMeanSquaredError`).
@available(macOS 10.14, iOS 13.0)
"Use DataFrame instead of MLDataTable."

```

```
@available 15.0 16.0
    "Use DataFrame instead of MLDataTable."
@available
public func evaluation MLDataTable
MLRegressorMetrics

    /// Exports a Core ML model file for use in your app.
public func write URL
MLModelMetadata nil throws

    /// Exports a Core ML model file for use in your app.
public func write String
MLModelMetadata nil throws

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLRandomForestRegressor CustomStringConvertible
CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of the random forest regressor.
public var description String get

    /// A text representation of the random forest regressor that's suitable for
    /// output during debugging.
public var debugDescription String get

    /// A description of the random forest regressor shown in a playground.
public var playgroundDescription Any get

extension MLRandomForestRegressor

    /// Parameters that affect the process of training a model.
@available macOS 10.14 iOS 15.0 tvOS 16.0
public struct ModelParameters

        /// Validation data represented as a `MLDataTable`.
        ///
        /// - Note: Setting this to `nil` means that the training data will be
automatically split for
        /// validation. Setting it to an empty table means to not use a
validation set.
@available 10.14 11.0
    "Use the validation property instead."
@available 15.0 16.0
    "Use the validation property instead."
@available
public var validationData MLDataTable
```

```

    /// Validation data.
    ///
    /// The default is `split(strategy: .automatic)`, which
    automatically generates the validation
    /// dataset from 0% to 10% of the training dataset.
    @available(macOS 11.0, iOS 15.0, tvOS 16.0)
    public var validation
MLRandomForestRegressor ModelParameters ValidationData

    public var maxDepth Int
    public var maxIterations Int
    public var minLossReduction Double
    public var minChildWeight Double
    public var randomSeed Int
    /// Must be in the range (0, 1).
    public var rowSubsample Double
    /// Must be in the range (0, 1).
    public var columnSubsample Double
    @available(macOS 11.0, iOS 15.0, tvOS 16.0)
    public init
MLRandomForestRegressor ModelParameters ValidationData
    Int 6           Int 10
Double 0           Double 0.1           Int
42           Double 0.8           Double 0.8

    /// Creates a new set of parameters.
    ///
    /// - Parameters:
    ///   - validationData: The dataset used to monitor how well the
    model is generalizing.
    ///
    ///   The default value is `nil` which will use an automatically
    sampled validation set.
    ///
    ///   - maxDepth: The maximum depth of the tree. Must be a value
    of at least 1.
    ///
    ///   The default value is 6.
    ///
    ///   - maxIterations: The maximum number of passes through
    the data.
    ///
    ///   The default value is 10.

```

```

    /**
     * - minLossReduction: The minimum amount of reduction in
     * the loss function that is required to make another
     * split to the data. Larger values help prevent overfitting.
     */
    /**
     * The default value is 0.
     */
    /**
     * - minChildWeight: Determines the minimum weight of each
     * leaf node of the tree. Larger values help prevent
     * overfitting.
     */
    /**
     * The default value is 0.1.
     */
    /**
     * - randomSeed: A seed for internal random operations. Set this
     * value to ensure reproducible results.
     */
    /**
     * The default value is 42.
     */
    /**
     * - rowSubsample: Select the specified ratio from the training
     * set to grow each tree. For example, a value
     * of 0.5 means each tree is trained on half the data. This
     * technique is known as _bagging.
     */
    /**
     * The default value is 0.8.
     */
    /**
     * - columnSubsample: Select the specified ratio of columns
     * from the training set to use when growing each
     * tree. Similar to row subsampling, this can be used to prevent
     * overfitting.
     */
    /**
     * The default value is 0.8.
@available 10.14 11.0
    "Use the validation property instead."
@available 15.0 16.0
    "Use the validation property instead."
@available
public init MLDDataTable nil
    Int 6 Int 10
Double 0 Double 0.1 Int
42 Double 0.8 Double 0.8

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLRandomForestRegressor ModelParameters
CustomStringConvertible CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /**
     * A text representation of the model parameters for a random forest
     * regressor.
     */

```

```
public var description String get
    /// A text representation of the model parameters for a random forest
    regressor that's suitable for output during
    /// debugging.
public var debugDescription String get
    /// A description of the model parameters for a random forest regressor
    shown in a playground.
public var playgroundDescription Any get

extension MLRandomForestRegressor ModelParameters

    /// Values for specifying validation data.
    @available macOS 11.0 iOS 15.0 tvOS 16.0
    public enum ValidationData
        /// Generate validation data by splitting the training dataset. This is the
        default.
        case split MLSplitStrategy
        /// Set validation data from the MLDataTable provided.
        @available           11.0          14.0
        @available           15.0          17.0
        @available           16.0          17.0
        case table MLDataTable
        /// Set validation data from the DataFrame provided.
        @available macOS 13.0 iOS 16.0 tvOS 16.0
        case dataFrame DataFrame
        /// Do not set validation data.
        case none

    /// A model you train to make recommendations based on item similarity,
    /// grouping, and, optionally, item ratings.
    ///
    /// Use an ``MLRecommender`` to train a machine learning model that you
    include
    /// in your app to make recommendations for the user, while keeping their data
    /// on-device.
    ///
    /// You create a recommender model by training it with tabular data that
    /// includes columns for the recommendation items and the groups the items
    /// belong to. You also have the option to include an item rating column, which
    /// gives higher-rated items more weight than those with lesser or negative
    /// ratings. The recommender uses the training information to find similarity
```

```
/// patterns by looking at items that occur in groups or have similar ratings
/// within groups.
///
/// After you train a recommender, you save it as a Core ML model file with the
/// ` `.mlmodel` extension. Import this model file into your Xcode project by
/// dragging it into the Project navigator. At runtime, use the recommender to
/// make item suggestions to the user based on the patterns in training data and
/// the user's item history. For example, a hiking app can recommend trails
/// based on the trails a user has previously hiked and their ratings of those
/// trails.
@available macOS 10.15
@available
@available
public struct MLRecommender

    /// The Core ML model.
public var model MLModel

    /// The name of the column you selected at initialization to define the user
    identifiers.
    ///
    /// Changing the value of this property doesn't retrain the model or affect its
    behavior.
public var userIdentifierColumn String

    /// The name of the column you selected at initialization to define the item
    identifiers.
    ///
    /// Changing the value of this property doesn't retrain the model or affect its
    behavior.
public var itemIdentifierColumn String

    /// The name of the column you selected at initialization to define the ratings.
    ///
    /// Changing the value of this property doesn't retrain the model or affect its
    behavior.
public var ratingColumn String

    /// The configuration parameters that the recommender used for training
    during initialization.
public let modelParameters MLRecommender.ModelParameters

    /// Creates an instance given a table and the names of the item and user
    columns contained therein.
    ///
    /// - Parameters:
    ///   - trainingData: A data frame containing training data.
    ///   - itemColumn: Name of the Int or String typed column in the
    training data containing item identifiers.
    ///   - userColumn: Name of the Int or String typed column in the
    training data containing user identifiers.
```

```

    /// - ratingColumn: Name of an Int or Double typed column optionally
in the training data containing scores or
    /// ratings. The default is nil, which corresponds to no rating column.
    /// - parameters: Model training parameters.
@available macOS 13.0
@available
@available
public init           DataFrame          String
String             String      nil
MLRecommender ModelParameters
nil   throws

    /// Creates an instance given a table and the names of the item and user
columns contained therein.
    ///
    /// - Parameters:
    /// - trainingData: A MLDataTable containing training data.
    /// - itemColumn: Name of the Int or String typed column in the
training data containing item identifiers.
    /// - userColumn: Name of the Int or String typed column in the
training data containing user identifiers.
    /// - ratingColumn: Name of an Int or Double typed column optionally
in the training data containing scores or
    /// ratings. The default is nil, which corresponds to no rating column.
    /// - parameters: Model training parameters.
@available           10.15          13.0
                    "Use DataFrame instead of MLDataTable when
initializing."
@available
@available
public init           MLDataTable        String
String             String      nil
MLRecommender ModelParameters
nil   throws

    /// Retrieves the highest scored item for the given array of users, based on
item similarity and the rating column.
    ///
    /// - Parameters:
    /// - fromUsers: An array of user identifiers.
    /// - maxCount: The maximum number of recommendations per user.
The default is `10`.
    /// - restrictingToItems: An array of item identifiers that defines
the only values the recommender can use this
    /// set of recommendations. By default, the parameter is `nil`,
meaning there are no restrictions.
    /// - userItemObservations: A data table of user-item observations
to exclude from recommendations. The default
    /// is `nil`, meaning no observations are excluded. The column
names for the user identifiers and item
    /// identifiers must be the same as those provided in the training data.

```

```

    /// - excludingObserved: Set this value to `true` to omit training
    // data from the recommendations, or `false` to
    /// include them. The default is `true`.
    ///
    /// - Returns: An ``MLDataTable`` containing columns with user
    // identifiers, item identifiers, scores and ranks
    /// (numbered between `1` and the `maxCount`).
    @available(10.15, 14.0)
    @available
    @available
    public func recommendations<T>(
        Int 10,
        any MLIdentifier,
        any MLIdentifier,
        MLDDataTable<T> nil,
        Bool true throws MLDDataTable<T>
    )

    /// Retrieves the highest scored items for the given column of users, based
    // on item similarity and the rating
    /// column.
    ///
    /// - Parameters:
    ///   - fromUsers: A column of user identifiers.
    ///   - maxCount: The maximum number of recommendations per user.
    // The default is `10`.
    ///   - restrictingToItems: An array of item identifiers that defines
    // the only values the recommender can use this
    ///     set of recommendations. By default, the parameter is `nil`,
    // meaning there are no restrictions.
    ///   - userItemObservations: A data table of user-item observations
    // to exclude from recommendations.
    ///
    ///     The default is `nil`, meaning no observations are excluded.
    ///
    ///     The column names for the user identifiers and item identifiers must
    // be the same as those provided in the
    ///     training data.
    ///   - excludingObserved: Set this value to `true` to omit training
    // data from the recommendations, or `false` to include them.
    ///
    ///     The default is `true`.
    ///
    /// - Returns: An ``MLDataTable`` containing columns with user
    // identifiers, item identifiers, scores and ranks
    /// (numbered between `1` and the `maxCount`).
    @available(10.15, 14.0)
    @available
    @available
    public func recommendations<T>(
        Int 10,
        MLDDataTable<T> nil,
        Bool true throws MLDDataTable<T> where
    )

```

```
T MLDataValueConvertible T MLIdentifier
```

```
    /// Computes the metrics for the given testing data.  
    ///  
    /// Let Pk be a vector of the first k items recommended by the model for a  
    particular user and  
    /// let A be the set of items in the provided testingData for the same user.  
    ///  
    /// The "precision at cutoff k" for this user is:  
    ///           precision(k) = IA ∩ Pk / k  
    ///  
    /// while "recall at cutoff k" for this user is:  
    ///           recall(k) = IA ∩ Pk / |A|  
    ///  
    /// where IA ∩ Pk is the number of elements in the intersection of A and Pk  
    and |A| is the  
    /// number of elements in A.  
    ///  
    /// - Parameters:  
    ///   - testingData: A MLDataTable containing testing data.  
    ///   - userColumn: Name of the Int or String typed column in the testing  
    data containing user identifiers.  
    ///   - itemColumn: Name of the Int or String typed column in the testing  
    data containing item identifiers.  
    ///   - ratingColumn: Name of an Int or Double typed column optionally  
    in the testing data containing  
    ///     scores or ratings. The default is nil, which corresponds to no rating  
    column.  
    ///     - cutoffs: A list of Ints corresponding to each value at which the  
    precision and recall will be evaluated.  
    ///       The default is [1,2,3,4,5].  
    ///     - excludingObserved: Specifies whether user-item interactions  
    observed in the training data are excluded when  
    ///       generating evaluation result. The default is true.  
@available macOS 13.0  
@available  
@available  
public func evaluation DataFrame  
    String String String  
nil Int 1 2 3 4 5 String  
Bool true MLRecommenderMetrics
```

  

```
    /// Computes the metrics for the given testing data.  
    ///  
    /// - Parameters:  
    ///   - testingData: A MLDataTable containing testing data.  
    ///   - userColumn: Name of the Int or String typed column in the testing  
    data containing user identifiers.  
    ///   - itemColumn: Name of the Int or String typed column in the testing  
    data containing item identifiers.  
    ///   - ratingColumn: Name of an Int or Double typed column optionally
```

```

in the testing data containing scores or
    ///      ratings. The default is nil, which corresponds to no rating column.
    ///      - cutoffs: A list of Ints corresponding to each value at which the
precision and recall will be evaluated.
    ///      The default is [1,2,3,4,5].
    ///
    ///      Definition: Let  $P_k$  be a vector of the first  $k$  items recommended by
the model for a particular user and
    ///      let  $A$  be the set of items in the provided testingData for the same
user.
    ///
    ///      The "precision at cutoff  $k$ " for this user is: `precision( $k$ ) = |A
 $\cap P_k| / k` while "recall at cutoff  $k$ " for
    ///      this user is: `recall( $k$ ) = |A \cap P_k| / |A|` where `|A \cap
 $P_k|` is the number of elements in the intersection
    ///      of  $A$  and  $P_k$  and  $|A|$  is the number of elements in  $A$ .
    ///      - excludingObserved: Specifies whether user-item interactions
observed in the training data are excluded when
    ///      generating evaluation result. The default is true.
@available                               10.15           13.0
    "Use DataFrame instead of MLDataTable."
@available
@available
public func evaluation          MLDataTable
    String           String           String
nil        Int       1 2 3 4 5
Bool     true      MLRecommenderMetrics$$ 
```

## **extension** **MLRecommender**

```

    /// Exports the recommender as a Core ML model file at the given URL.
    ///
    /// - Parameters:
    ///   - fileURL: The location in the file system to which the file should be
written.
    ///   - metadata: Descriptive information to include with the exported
model file.
@available macOS 10.15
@available
@available
public func write           URL
MLModelMetadata   nil   throws

    /// Exports the recommender as a Core ML model file at the given file path.
    ///
    /// - Parameters:
    ///   - path: A file system path where the model file should be written.
    ///   - metadata: Descriptive information to include with the exported
model file.

```

```

@available macOS 10.15
@available
@available
public func write           String
MLModelMetadata nil throws

extension MLRecommender

    /// Returns the top ranked similar items based on the model's similarity
    /// type.
    ///
    /// - Parameters:
    ///   - fromItems: An array of item identifiers.
    ///
    ///   - maxCount: The maximum number of similar items per item in the
    ///     `fromItems` column. The default is `10`.
    @available 10.15          14.0
    @available
    @available
    public func getSimilarItems      any MLIdentifier
        Int 10 throws MLDataTable

    /// Returns the top ranked similar items based on the model's similarity
    /// type.
    ///
    /// - Parameters:
    ///   - fromItems: A column of item identifiers.
    ///
    ///   - maxCount: The maximum number of similar items per item in the
    ///     `fromItems` column. The default is `10`.
    @available 10.15          14.0
    @available
    @available
    public func getSimilarItems T      ML DataColumn T
        Int 10 throws MLDataTable where T
MLDataTableConvertible

```

**extension** MLRecommender

```

    /// The algorithms a recommender can use to make recommendations.
    @available macOS 10.15
    @available
    @available
    public enum ModelAlgorithmType  Sendable

        /// An algorithm that compares the similarity from item to item.
        ///

```

```
    /// The default value is
``MLRecommender/SimilarityType/jaccard``.
case itemSimilarity MLRecommender SimilarityType

    /// The metric by which the recommender computes item similarity.
@available macOS 10.15
@available
@available
public enum SimilarityType Sendable

        /// The cosine similarity measure.
        ///
        /// Use cosine similarity for comparing item ratings.
case cosine

        /// The Jaccard similarity measure.
        ///
        /// Use Jaccard similarity when the presence or absence of a user's
        /// rating is more important than the rating itself.
case jaccard

        /// The Pearson correlation similarity measure.
        ///
        /// Use Pearson correlation similarity for comparing item ratings.
case pearson

        /// Returns a Boolean value indicating whether two values are equal.
        ///
        /// Equality is the inverse of inequality. For any values `a` and `b`,
        /// `a == b` implies that `a != b` is `false`.
        ///
        /// - Parameters:
        ///   - `lhs`: A value to compare.
        ///   - `rhs`: Another value to compare.
public static func
MLRecommender SimilarityType MLRecommender SimilarityType
Bool

    /// Hashes the essential components of this value by feeding them into
the
    /// given hasher.
    ///
    /// Implement this method to conform to the `Hashable` protocol. The
    /// components used for hashing must be the same as the components
compared
    /// in your type's `==` operator implementation. Call
`hasher.combine(_:)`
    /// with each of these components.
    ///
```

```

    /// - Important: In your implementation of `hash(into:)`,
    /// don't call `finalize()` on the `hasher` instance provided,
    /// or replace it with a different instance.
    /// Doing so may become a compile-time error in the future.
    ///
    /// - Parameter hasher: The hasher to use when combining the
components
    ///   of this instance.
public func hash           inout Hasher

    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different
executions of
    /// your program. Do not save hash values to use during a future
execution.
    ///
    /// - Important: `hashValue` is deprecated as a `Hashable`
requirement. To
    /// conform to `Hashable`, implement the `hash(into:)`
requirement instead.
    /// The compiler provides an implementation for `hashValue` for
you.
public var hashValue  Int  get

```

## **extension** MLRecommender

```

    /// Parameters that affect the process of training a recommender model.
@available macOS 10.15
@available
@available
public struct ModelParameters

    /// The algorithm the recommender uses to make recommendations.
    ///
    /// The default is
    ///
``MLRecommender/ModelAlgorithmType/itemSimilarity(_:)``.
public var algorithm  MLRecommender ModelAlgorithmType

    /// The item confidence value cutoff, below which the recommender
omits
    /// those items from its recommendations.
    ///
    /// The default value is `0.001`.
public var threshold  Double

    /// The largest number of similar items the model stores for each item.

```

```
///  
/// The memory Create ML requires to train this model is proportional to  
/// this number. A lower value reduces its demand for memory but  
/// decreases the recommender's accuracy. The default value is `64`.  
public var maxCount Int  
  
/// A data frame that lists each item's nearest items.  
@available macOS 14.0  
public var nearestItemsDataFrame DataFrame  
  
/// A data table that lists each item's nearest items.  
@available 10.15 14.0  
"Use nearestItemsDataFrame."  
public var nearestItems MLDataTable  
  
/// The largest number of iterations the recommender uses to build its  
/// lookup table.  
///  
/// This value limits the number of iterations the recommender can use  
/// to construct a lookup table. The default value is `1024`.  
public var maxSimilarityIterations Int  
  
/// Creates a new set of recommender configuration parameters.  
@available macOS 14.0  
public init  
MLRecommender ModelAlgorithmType  
    Double 0.001           Int 64  
        DataFrame  
Int 1024  
  
/// Creates a new set of recommender configuration parameters.  
@available 10.15 14.0  
public init  
MLRecommender ModelAlgorithmType  
    Double 0.001           Int 64  
    MLDataTable           Int 1024  
  
  
@available macOS 10.15  
@available  
@available  
extension MLRecommender ModelAlgorithmType  
CustomStringConvertible CustomDebugStringConvertible  
CustomPlaygroundDisplayConvertible  
  
/// A text representation of the recommender algorithm.  
public var description String get
```

```
    /// A text representation of the recommender algorithm that's suitable for
    output during debugging.
    public var debugDescription String get

    /// A description of the recommender algorithm shown in a playground.
    public var playgroundDescription Any get

@available macOS 10.15
@available
@available
extension MLRecommender SimilarityType
CustomStringConvertible CustomDebugStringConvertible

    /// A text representation of the similarity type.
    public var description String get

    /// A text representation of the similarity type that's suitable for output during
    debugging.
    public var debugDescription String get

@available macOS 10.15
@available
@available
extension MLRecommender SimilarityType
CustomPlaygroundDisplayConvertible

    /// A description of the similarity type shown in a playground.
    public var playgroundDescription Any get

@available macOS 10.15
@available
@available
extension MLRecommender SimilarityType Equatable

@available macOS 10.15
@available
@available
extension MLRecommender SimilarityType Hashable

    /// Metrics you use to evaluate a recommender's performance.
    @available macOS 10.15
    @available
    @available
    public struct MLRecommenderMetrics
```

```
    /// A Boolean value that indicates whether the recommender omitted training
    data from the recommendations.
    public let excludingObserved Bool

    /// The underlying error present when the metrics are invalid.
    public let error any Error

    /// A Boolean value indicating whether the recommender model was able to
    calculate metrics.
    public var isValid Bool get

    /// A data table with the recall and precision for each item.
    @available macOS 14.0
    public var precisionRecallDataFrame DataFrame get

    /// A data table with the recall and precision for each item.
    @available 10.15 14.0
    @available
    @available
    public var precisionRecall MLDataTable get

    /// Creates metrics for a recommender, given a data table with precision and
    recall metric columns, and whether the
    /// recommender omitted training data.
    ///
    /// Do not use this initializer. ``MLRecommender`` generates metrics for
    you when you call its
    ///
    ``MLRecommender/evaluation(on:userColumn:itemColumn:ratingColu-
    mn:cutoffs:excludingObserved:)``
    /// method.
    @available 10.15 14.0
    @available
    @available
    public init
        MLDataTable
        Bool

    /// A model you train to estimate continuous values.
    ///
    /// Use an ``MLRegressor`` to estimate continuous values like price, time, or
    temperature.
    ///
    /// A regressor differs from a classifier because it can predict output values not
    seen during the training process. By
    /// contrast, a classifier can only classify input into the categories you provide in
    the training data.
    ///
    /// For example, when estimating housing prices on Mars, a regressor can
    interpolate between the examples to estimate
```

```
/// prices not seen during training. The figure below shows a linear regressor for
/// Mars real-estate prices similar to
/// the
<doc://com.apple.documentation/documentation/coreml/integrating_a_core_ml_mod
el_into_your_app> sample.
///
/// ! [A graph showing housing prices for mars with a linear regressor used to
/// create a continuous estimation between
/// points.] (MLRegressor-1)
///
/// In this case, there are no data points with three solar panels, but the regressor
/// can make an informed prediction
/// about the housing price.
///
/// When you create an ``MLRegressor``, Create ML inspects your data and
/// automatically chooses a specific regressor
/// (see Supporting Regressor Types).
@available macos 10.14
@available
@available
public enum MLRegressor : Sendable

    /// A regressor based on a collection of decision trees combined with
    gradient boosting.
    ///
    /// Don't create an ``MLRegressor`` using one of its enumeration cases.
    Use the regressor's initializer instead.
    case boostedTree : MLBoostedTreeRegressor

    /// A regressor that estimates the target by learning rules to split the data.
    ///
    /// Don't create an ``MLRegressor`` using one of its enumeration cases.
    Use the regressor's initializer instead.
    case decisionTree : MLDecisionTreeRegressor

    /// A regressor based on a collection of decision trees trained on subsets of
    the data.
    ///
    /// Don't create an ``MLRegressor`` using one of its enumeration cases.
    Use the regressor's initializer instead.
    case randomForest : MLRandomForestRegressor

    /// A regressor that estimates the target as a linear function of the features.
    ///
    /// Don't create an ``MLRegressor`` using one of its enumeration cases.
    Use the regressor's initializer instead.
    case linear : MLLinearRegressor

    /// Creates a regressor.
    ///
    /// - Parameters:
```

```

    ///> - trainingData: The training data
    ///> - targetColumn: Name of the column containing the target values
    ///> - featureColumns: Names of the columns containing feature
values. If `nil` all columns, other than the target
    ///>     column, will be used as feature values.
@available macOS 12.0
@available
@available
public init           DataFrame          String
String      nil throws

    ///> Creates a regressor from the feature columns in the training data to
predict the values in the target column.
    ///
    ///> To view details about the supporting model picked by the
``MLRegressor```, print the model's description:
    ///
    ///> ````swift
    ///> print(model)
    ///
    ///
    ///> - Parameters:
    ///> - trainingData: A data table of training examples.
    ///> - targetColumn: The column name for the values in the training
data the regressor should predict.
    ///> - featureColumns: The column names for the values in the
training data that the regressor uses to predict the
    ///>     target value.
@available                  10.14          13.0
    "Use DataFrame instead of MLDataTable when
initializing."
@available
@available
public init           MLDataTable
String      String      nil throws

@available macOS 12.0
@available
@available
public func predictions      DataFrame  throws
AnyColumn

    ///> Predicts the target value from the provided data.
    ///
    ///> If the supplied data doesn't match the expected columns (noted by the
``MLRegressor/featureColumns`` property),
    ///> this method throws an ``MLCreateError/type(reason:)``.
    ///
    ///> - Parameters:
    ///> - data: The data you want the model to make predictions from.

```

```
///  
/// – Returns: A column of values predicted by the regressor.  
@available 10.14 13.0  
    "Use DataFrame instead of MLDataTable."  
@available  
@available  
public func predictions  
MLDataTable throws  
MLUntypedColumn  
  
/// Evaluates the classifier on the provided labeled data.  
///  
/// Evaluation should be done on a testing data set that the model has not  
seen as part of the training or  
/// validation data sets. The data should have feature columns with identical  
name and type to the  
/// training data, as well as a labels column with the same name.  
///  
/// – Parameters:  
///     – labeledData: A `DataFrame` to evaluate the trained model on.  
///  
/// – Returns: Metrics that describe the maximum error  
/// (`^MLRegressorMetrics/maximumError`) or the average error  
/// (`^MLRegressorMetrics/rootMeanSquaredError`).  
@available macOS 12.0  
@available  
@available  
public func evaluation  
DataFrame  
MLRegressorMetrics  
  
/// Evaluates the classifier on the provided labeled data.  
///  
/// Evaluation should be done on a testing data set that the model has not  
seen as part of the training or  
/// validation data sets. The data should have feature columns with identical  
name and type to the  
/// training data, as well as a labels column with the same name.  
///  
/// – Parameters:  
///     – labeledData: An `MLDataTable` to evaluate the trained model  
on.  
///  
/// – Returns: Metrics that describe the maximum error  
(`^MLRegressorMetrics/maximumError`) or the average error  
(`^MLRegressorMetrics/rootMeanSquaredError`).  
@available 10.14 13.0  
    "Use DataFrame instead of MLDataTable."  
@available  
@available  
public func evaluation  
MLDataTable  
MLRegressorMetrics
```

```
    /// Exports a Core ML model file for use in your app.
    public func write URL
MLModelMetadata throws

    /// Exports a Core ML model file for use in your app.
    public func write String
MLModelMetadata throws

@available macOS 10.14
@available
@available
extension MLRegressor

    /// The underlying Core ML model stored in memory.
    public var model MLModel get

    /// Measurements of the regressor's performance on the training data set.
    public var trainingMetrics MLRegressorMetrics get

    /// Measurements of the regressor's performance on the validation data set.
    public var validationMetrics MLRegressorMetrics get

    /// The name of the column you selected at initialization to define which
    feature the regressor predicts.
    public var targetColumn String get

    /// The names of the columns you selected at initialization to train the
    regressor.
    public var featureColumns String get

@available macOS 10.14
@available
@available
extension MLRegressor CustomStringConvertible

    /// A text representation of the regressor.
    public var description String get

@available macOS 10.14
@available
@available
extension MLRegressor CustomDebugStringConvertible

    /// A text representation of the regressor that's suitable for output during
    debugging.
    public var debugDescription String get
```

```
@available macOS 10.14
@available
@available
extension MLRegressor CustomPlaygroundDisplayConvertible

    /// A description of the regressor shown in a playground.
public var playgroundDescription Any get

    /// Metrics you use to evaluate a regressor's performance.
    ///
    /// To understand what performance you can expect from the regressor, you start
    /// by looking at its ``MLRegressorMetrics/maximumError``. This high-
    level
    /// metric indicates your model's worst-case performance. To get a sense for how
    /// your model performs on average, look at the
    /// ``MLRegressorMetrics/rootMeanSquaredError``. In both cases, you
    want to
    /// minimize the value and therefore the error.
    ///
    /// - Note: Each trained model contains different metrics for its various data
    /// sets (training, validation, and testing).
    /// <doc:improving-your-model-s-accuracy> compares these metrics among
    different
    /// data sets.
@available macOS 10.14 iOS 15.0 tvOS 16.0
public struct MLRegressorMetrics Sendable

    /// The underlying error present when the metrics are invalid.
public var error any Error get

    /// A Boolean value indicating whether the regressor model was able to
    /// calculate metrics.
    ///
    /// Your metrics may be invalid if you attempt to perform evaluation on data
    /// that doesn't match the structure of your training examples.
public var isValid Bool get

    /// The largest absolute difference between the expected values and the
    /// model's predicted values during testing or training.
public var maximumError Double get

    /// A common metric used to determine the deviation between correct and
    /// predicted values.
    ///
    /// The ``MLRegressorMetrics/rootMeanSquaredError`` is
    calculated by taking
    /// the square-root of the average squared distance between the correct and
```

```
    /// predicted value.
    public var rootMeanSquaredError Double get

    /// Creates regressor metrics describing the quality of your model.
    ///
    /// You typically don't initialize metrics directly. Instead you get metrics
    /// about your model after training. For example, when you train an
    /// ``MLRegressor``, you can look at its
    ``MLRegressor/trainingMetrics`` and
    ``MLRegressor/validationMetrics`` properties. Additionally, you
can
    /// check the performance on a test set with the
    /// ``MLRegressor/evaluation(on:)`` method.
    ///
    /// - Parameters:
    ///   - maximumError: The maximum error of the model for the training
data.
    ///   - rootMeanSquaredError: The root mean squared error of the
model for the training data.
    public init Double
Double

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLRegressorMetrics CustomStringConvertible
CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of the regressor metrics.
    public var description String get

    /// A text representation of the regressor metrics that's suitable for
    /// output during debugging.
    public var debugDescription String get

    /// A description of the regressor metrics shown in a playground.
    public var playgroundDescription Any get

    /// A machine learning model you train with audio files to recognize and identify
sounds on a device.
    ///
    /// A sound classifier is a machine learning model that identifies and categorizes
sounds in an app. Create a sound
    /// classifier by gathering a dataset of audio files and use them to train a model
with ``MLSoundClassifier``.
    ///
    /// Assemble an audio dataset by recording or gathering audio files that best
represent the sounds you want your app to
    /// identify. Additionally, create a _negative class_ — a group of related noises the
```

sound classifier might hear but  
/// aren't relevant — by collecting or recording example sounds.  
///  
/// For example, say you're creating a sound classifier to identify laughter and applause. In addition to gathering  
/// audio examples of people laughing and clapping, you can add an additional category for background noise. By adding  
/// recordings from various settings, such as theaters and amphitheaters, your sound classifier can distinguish the  
/// sounds of interest from environmental noises. In other words, the sound classifier won't predict "Applause" when  
/// there isn't any. Like any classifier, when you request a prediction, a sound classifier always returns one of the  
/// categories it learned from a training dataset.  
///  
/// Gather at least 10 audio examples of each sound category you want the sound classifier to learn, plus at least one  
/// negative class for background noise. The audio examples can be in any file format that Core Audio supports,  
/// including:  
///  
/// – M4A  
/// – MP3  
/// – AIFF  
/// – WAV  
///  
/// – Tip: Use single-channel audio files with a sample rate of 16 kHz or higher for best results.  
///  
/// Reduce a sound classifier's bias — which can adversely affect its performance  
– by gathering audio files that use a  
/// consistent bit depth and sample rate.  
///  
/// Train, evaluate, and export your sound classifier by following similar steps to creating any other Create ML model  
/// type. For more information about the Create ML training workflow, see:  
///  
/// – <doc:creating-an-image-classifier-model>  
/// – <doc:creating-an-action-classifier-model>  
///  
/// Add the sound classifier's Core ML model to an Xcode project and use it to create an  
/// <https://com.apple.documentation/documentation/documentation/soundanalysis/snclassifysoundrequest> at runtime. Your app uses the  
/// sound request to identify sounds in an audio file or audio stream by following the steps in the following articles,  
/// respectively:  
///  
/// –  
[https://com.apple.documentation/documentation/documentation/soundanalysis/classifying\\_sounds\\_in\\_an\\_audio\\_file](https://com.apple.documentation/documentation/documentation/soundanalysis/classifying_sounds_in_an_audio_file)

```
n_an_audio_file>
/// -
<doc://com.apple.documentation/documentation/soundanalysis/classifying_sounds_i
n_an_audio_stream>
@available macOS 10.15 iOS 15.0
@available
public struct MLSoundClassifier @unchecked Sendable

    /// The underlying model instance of the sound classifier stored in memory.
    public var model MLModel

    /// The model configuration parameters the sound classifier used during its
    training session.
    public let modelParameters
    MLSoundClassifier ModelParameters

    /// Measurements of the classifier's performance on the training data set.
    public var trainingMetrics MLClassifierMetrics get

    /// Measurements of the image classifier's performance on the validation
    dataset.
    public var validationMetrics MLClassifierMetrics get

    /// Creates a sound classifier with a training dataset represented by a data
    source.
    ///
    /// Use this initializer to train a sound classifier with an
    ``MLSoundClassifier/DataSource``. For example, you can
    /// organize your audio files into labeled directories. See
    ///
    ``MLSoundClassifier/DataSource/labeledDirectories(at:)``.
    ///
    /// ``swift
    /// // Get the Documents directory URL.
    /// guard let documentsURL = FileManager.default.urls(for:
    .documentDirectory,
    ///
    in: .userDomainMask).first else {
        ///     fatalError("Can't find Documents directory.")
        /// }
    ///
    /// // Build a URL to the ~/Documents/Sounds directory,
    which contains the training data.
    /// let soundsURL =
    documentsURL.appendingPathComponent("Sounds")
    ///
    /// // The Sounds directory contains subdirectories, one
    for each class of sound.
    /// // Each subdirectory's name is the label for audio
    files it contains.
```

```

    /**
     * Sounds
     * -- Laughter
     * -- Recording1.wav
     * -- Recording4.wav
     * ...
     * -- Applause
     * -- Recording2.wav
     * -- Recording5.wav
     * ...
     */
    // Create a data source from the Sounds directory.
    let trainingData =
MLSoundClassifier.DataSource.labeledDirectories(at: soundsURL)
    /*
     * Train a sound classifier with the data source.
     * let soundClassifier = try
MLSoundClassifier(trainingData: trainingData)
    /**
     */
    /**
     - Parameters:
     - trainingData: An ``MLSoundClassifier/DataSource`` instance that contains a collection of labeled audio files.
     - parameters: An ``MLSoundClassifier/ModelParameters-swift.struct`` instance you use to configure the model for the training session.
public init          MLSoundClassifier DataSource
                    MLSoundClassifier ModelParameters
                    throws

     * Creates a sound classifier with a training dataset represented by a dictionary.
     */
    /**
     - Parameters:
     - trainingData: A dictionary that contains a collection of labeled audio files. Each of the dictionary's keys is a label, and each key's value is an array of audio-file URLs.
     - parameters: An ``MLSoundClassifier/ModelParameters-swift.struct`` instance you use to configure the model for the training session.
@available           10.15          11.0
                    "Use DataSource.filesByLabel to provide dictionary training data instead."
@available           15.0          16.0
                    "Use DataSource.filesByLabel to provide dictionary training data instead."
@available
public init          String      URL
                    MLSoundClassifier ModelParameters
                    throws

```

```
    /// Creates a sound classifier from a training session checkpoint.  
    ///  
    /// - Parameters:  
    ///   - checkpoint: A checkpoint from a sound classifier training  
session.  
    @available macOS 11.0 iOS 15.0  
    @available  
    public init          MLCheckpoint throws  
  
    /// Begins an asynchronous sound classifier training session with a training  
dataset represented by a data source.  
    ///  
    /// - Parameters:  
    ///   - trainingData: A collection of labeled audio files represented by  
an ``MLSoundClassifier/DataSource``.  
    ///   - parameters: An ``MLSoundClassifier/ModelParameters-  
swift.struct`` instance you use to configure the model  
    ///     for the training session.  
    ///   - sessionParameters: An  
``MLTrainingSessionParameters`` instance you use to configure the training  
session.  
    ///  
    /// - Returns: An ``MLJob`` that represents the sound classifier  
training session.  
    @available macOS 11.0 iOS 15.0  
    @available  
    public static func train  
        MLSoundClassifier DataSource  
        MLSoundClassifier ModelParameters  
        MLTrainingSessionParameters  
        throws      MLJob MLSoundClassifier  
  
    /// Begins an asynchronous sound classifier training session with a training  
dataset represented by a dictionary.  
    ///  
    /// - Parameters:  
    ///   - trainingData: A collection of labeled audio files represented by  
a dictionary. The keys of this dictionary  
    ///     are the labels, and the values are the associated audio-file URLs.  
    ///   - parameters: An ``MLSoundClassifier/ModelParameters-  
swift.struct`` instance you use to configure the model  
    ///     for the training session.  
    ///   - sessionParameters: An  
``MLTrainingSessionParameters`` instance you use to configure the training  
session.  
    ///  
    /// - Returns: An ``MLJob`` that represents the sound classifier  
training session.  
    @available macOS 11.0 iOS 15.0
```

```

@available
public static func train String URL
    MLSoundClassifier ModelParameters
    MLTrainingSessionParameters
throws MLJob MLSoundClassifier

    /// Begins an asynchronous session that extracts sound features from a data
    source of sound files.
    ///
    /// Use this method to reduce the training time for multiple sound classifiers
    that use the same training data. Use
    /// the ``MLJob`` instance this method returns to save the audio features
    as an ``MLSoundClassifier/DataSource``.
    /// Then use the audio features data source to train one or more sound
    classifiers.
    ///
    /// You can also create a data source from a
<doc://com.apple.documentation/documentation/tabulardata/frame> or
    /// an ``MLDataTable`` that contains audio features by using
    ///
``MLSoundClassifier/DataSource/featuresDataFrame(_:featureColu
mn:labelColumn:parameters:)`` or
    ///
``MLSoundClassifier/DataSource/features(table:featureColumn:la
belColumn:parameters:)``, respectively.
    ///
/// - Parameters:
    /// - trainingData: An ``MLSoundClassifier/DataSource`` instance that contains a collection of labeled audio
        /// files.
    /// - parameters: An
``MLSoundClassifier/FeatureExtractionParameters`` instance you use
to configure the feature
    /// extraction session.
    /// - sessionParameters: An
``MLTrainingSessionParameters`` instance you use to configure the feature
extraction
    /// session.
    ///
/// - Returns: An ``MLJob`` that represents the sound feature
extraction session.
@available macOS 11.0 iOS 15.0
@available
public static func extractFeatures
MLSoundClassifier DataSource
MLSoundClassifier FeatureExtractionParameters

MLTrainingSessionParameters
throws MLJob MLSoundClassifier DataSource

```

```
    /// Creates an asynchronous training session for a sound classifier.  
    ///  
    /// - Parameters:  
    ///   - trainingData: A collection of labeled audio files represented by  
    ///     an ``MLSoundClassifier/DataSource``.  
    ///   - parameters: An ``MLSoundClassifier/ModelParameters-  
    ///     swift.struct`` instance you use to configure the model  
    ///     for the training session.  
    ///   - sessionParameters: An  
    ///     ``MLTrainingSessionParameters`` instance you use to configure the training  
    ///     session.  
    ///  
    /// - Returns: An ``MLTrainingSession`` that represents the sound  
    ///     classifier training session.  
    @available macOS 11.0 iOS 15.0  
    @available  
    public static func makeTrainingSession  
        MLSoundClassifier DataSource  
        MLSoundClassifier ModelParameters  
            MLTrainingSessionParameters  
                throws  
        MLTrainingSession MLSoundClassifier  
  
    /// Creates an asynchronous training session for a sound classifier by  
    /// restoring an existing training session's  
    /// state from its parameters.  
    ///  
    /// - Parameters:  
    ///   - sessionParameters: The  
    ///     ``MLTrainingSessionParameters`` instance you used to create the training  
    ///     session  
    ///       using  
    ``MLSoundClassifier/makeTrainingSession(trainingData:parameters:  
sessionParameters:)``.  
    ///  
    /// - Returns: An ``MLTrainingSession`` that represents the sound  
    ///     classifier training session.  
    @available macOS 11.0 iOS 15.0  
    @available  
    public static func  
        restoreTrainingSession  
            MLTrainingSessionParameters throws  
        MLTrainingSession MLSoundClassifier  
  
    /// Begins or continues an asynchronous training session for a sound  
    /// classifier.  
    ///  
    /// - Parameters:  
    ///   - session: An ``MLTrainingSession`` instance that  
    ///     represents the training session.
```

```

    /**
     * - Returns: An ``MLJob`` that represents the sound classifier
     * training session.
     */
    @available(macOS 11.0, iOS 15.0)
    @available
    public static func resume(
        MLTrainingSession MLSoundClassifier) throws
        MLJob MLSoundClassifier

        /**
         * Generates metrics by evaluating the sound classifier's performance on a
         * dataset represented by a data source.
         */
        /**
         * - Parameters:
         *   - testingData: A collection of labeled audio files represented by
         *     an ``MLSoundClassifier/DataSource``.
         */
        /**
         * - Returns: An ``MLClassifierMetrics`` instance that contains
         * the evaluation results.
         */
        public func evaluation(
            MLSoundClassifier DataSource) MLClassifierMetrics

        /**
         * Generates metrics by evaluating the sound classifier's performance on a
         * dataset represented by a dictionary.
         */
        /**
         * - Parameters:
         *   - testingData: A collection of labeled audio files represented by a
         *     dictionary. Each key of the dictionary is
         *       a label, and its value is an array of audio-file URLs.
         */
        /**
         * - Returns: An ``MLClassifierMetrics`` instance that contains
         * the evaluation results.
         */
        @available(10.15, 11.0)
        "Use DataSource.filesByLabel to provide dictionary
        testing data instead."
        @available(15.0, 16.0)
        "Use DataSource.filesByLabel to provide dictionary
        testing data instead."
        @available
        public func evaluation(
            MLClassifierMetrics) String URL

        /**
         * Exports the sound classifier as a model file to a location in the file system.
         */
        /**
         * Use this method to save the sound classifier as a Core ML model to a
         * URL.
         */
        /**
         * This method:
         */
        /**
         * - Uses the name `SoundClassifier.mlmodel` if the URL's location
         * is a directory
         */

```

```
    /// - Appends `mlmodel` as the extension if you don't provide one
    /// - Creates intermediate directories if none exist
    ///
    /// - Parameters:
    ///   - fileURL: The location URL in the file system where you want to
    save the model.
    ///   - metadata: Descriptive information to include with the exported
    model file.
public func write URL
MLModelMetadata nil throws

    /// Exports the sound classifier as a model file to a path in the file system.
    ///
    /// Use this method to save the sound classifier as a Core ML model to a
path.
    ///
    /// This method:
    ///
    /// - Uses the name `SoundClassifier.mlmodel` if the path's location
is a directory
    /// - Appends `mlmodel` as the extension if you don't provide one
    /// - Replaces the tilde (~) with the path to your home directory
    /// - Creates intermediate directories if none exist
    ///
    /// - Parameters:
    ///   - path: The location path in the file system where you want to save
the model.
    ///   - metadata: Descriptive information to include with the exported
model file.
public func write String
MLModelMetadata nil throws

    /// Generates predictions for an array of audio files.
    ///
    /// - Parameters:
    ///   - audioFiles: An array of audio-file URLs you want the sound
classifier to categorize.
    ///
    /// - Returns: An array of prediction labels for the audio files.
public func predictions URL throws
String

    /// Generates predictions that use an overlap factor and time window size for
an array of audio files.
    ///
    /// - Parameters:
    ///   - audioFiles: An array of audio-file URLs you want the sound
classifier to categorize.
    ///   - overlapFactor: The amount of overlap between successive
analysis windows when the model analyzes a block of
    ///     audio data.
```

```
    /// - predictionTimeWindowSize: The duration of the audio buffer
the method sends to the model for each
    /// prediction.
    ///
    /// - Returns: An array of prediction labels for the audio files.
@available macOS 12.0 iOS 15.0
@available
public func predictions URL TimeInterval
throws String

@available macOS 10.15 iOS 15.0
@available
extension MLSoundClassifier

    /// Parameters that affect the process of training a sound-classifier model.
    ///
    /// By default, a sound-classifier training session's transfer-learning algorithm
uses the
    ///
``MLSoundClassifier/ModelParameters-swift.struct/FeatureExtractorType/audioFeaturePrint(type:revision:)``
    /// feature extractor. See ``MLSoundClassifier/ModelParameters-
swift.struct/FeatureExtractorType`` for more
    /// information.
public struct ModelParameters

    /// The sound classifier's validation dataset.
public var validation MLSoundClassifier ModelParameters ValidationData

    /// The largest number of iterations the training session can use.
public var maxIterations Int

    /// The proportion of overlap that the training session uses to analyze
two consecutive windows in the audio
    /// data.
    ///
    /// The overlap factor — which must be in the range ` [0.0, 1.0)` —
affects how much audio data the training
    /// session analyzes in each file. Sessions with smaller overlap factors
read fewer audio data samples and
    /// finish in less time but may compromise the model's prediction
accuracy. Sessions with larger overlap
    /// factors read more audio samples for each file, which increases the
session's training data and processing
    /// time. The additional training data can improve a sound classifier's
accuracy; however, it may only be a
    /// modest improvement that isn't worth the extra processing time.
```

```

    /**
     * The training session uses the expression
     `(featureExtractionTimeWindowSize * (1.0 - overlapFactor))` to
     determine the how much to _step_ (advance) in time between
samples.

    /**
     * | Window size | Overlap factor | Step time |
     * |-----|-----|-----|
     * | `1.0` | `0.0` | `1.0` |
     * | `2.0` | `0.0` | `2.0` |
     * | `5.0` | `0.0` | `5.0` |
     * | `1.0` | `0.5` | `0.5` |
     * | `2.0` | `0.5` | `1.0` |
     * | `5.0` | `0.5` | `2.5` |
     * | `1.0` | `0.75` | `0.25` |
     * | `2.0` | `0.75` | `0.5` |
     * | `5.0` | `0.75` | `1.25` |

    /**
     * For example, a session that's analyzing a 5-second audio file with a
window size of `1.0` and an overlap
     * factor of `0.0` samples the audio five times. The time offsets for
those samples are: `0.0`, `1.0`, `2.0`,
     * `3.0`, and `4.0`.

    /**
     * Another session with a window size of `1.0` and an overlap factor
of `0.5` samples the same audio file 10
     * times at half-second intervals. Unlike the first session, this session
samples each portion of audio data
     * twice, except for the first and final half-second.

    /**
     * A third session with a window size of `1.0` and an overlap factor of
`0.75` samples the same 5-second audio
     * file 20 times at quarter-second intervals. This third session samples
most of the audio portions four times;
     * the session samples the intervals near the ends one, two, or three
times.

public var overlapFactor Double

     * The algorithm the training session uses to train the sound classifier.
@available macOS 11.0 iOS 15.0
@available
public var algorithm
MLSoundClassifier ModelParameters ModelAlgorithmType

     * A time duration, in seconds, the training session uses for each audio
sample it reads from an audio file in
     * a dataset.
    /**
     * The time-window size value defaults to `0.975` seconds and must
be in the range `[0.5, 15.0]`.

```

```

    /**
     * Training sessions that use
     */
``MLSoundClassifier/ModelParameters-swift.struct/FeatureExtractorType/vggish(revision:)`` ignore this value
    /// and always use a time-window size of `0.975` seconds.
    @available macOS 12.0 iOS 15.0
@available
public var featureExtractionTimeWindowSize
TimeInterval

    /// Creates a new set of training parameters for a sound classifier with
a validation dataset and a training
    /// algorithm.
    /**
     * - Parameters:
     *   - validation: A validation dataset represented by an
     *     ``MLSoundClassifier/ModelParameters-
``swift.struct/ValidationData`` instance.
    /**
     *   - maxIterations: The largest number of iterations the
training session can use to train the sound
     *     classifier.
    /**
     *   - overlapFactor: A proportion of overlap the training session
uses to analyze two consecutive windows in
     *     the audio data. The proportion must be in the range `[0.0,
1.0)`. Higher proportions generate more
     *     training data, but also increases the training time.
    /**
     *     The default value is `0.5`, which represents a 50% overlap.
    /**
     *   - algorithm: The algorithm the training session uses to train
the sound classifier.
    @available macOS 11.0 iOS 15.0
@available
public init
MLSoundClassifier ModelParameters ValidationData
Int
Double

MLSoundClassifier ModelParameters ModelAlgorithmType

```

```

    /// Creates a new set of training parameters for a sound classifier with
a validation dataset.
    /**
     * - Parameters:
     *   - validation: A validation dataset represented by an
     *     ``MLSoundClassifier/ModelParameters-

```

```
swift.struct/ValidationData`` instance.  
///  
/// - maxIterations: The largest number of iterations the  
training session can use to train the sound  
/// classifier.  
///  
/// - overlapFactor: A proportion of overlap the training session  
uses to analyze two consecutive windows in  
/// the audio data. The proportion must be in the range ` [0.0,  
1.0)`. Higher proportions generate more  
/// training data, but also increases the training time.  
///  
/// The default value is `0.5`, which represents a 50% overlap.  
public init  
MLSoundClassifier ModelParameters ValidationData  
Int 25  
Double 0.5  
  
/// Creates a new set of training parameters for a sound classifier with  
a validation dataset, a training  
/// algorithm, and a time-window size.  
///  
/// - Parameters:  
/// - validation: A validation dataset represented by an  
/// ``MLSoundClassifier/ModelParameters-  
swift.struct/ValidationData`` instance.  
///  
/// - maxIterations: The largest number of iterations the  
training session can use to train the sound  
/// classifier.  
///  
/// - overlapFactor: A proportion of overlap the training session  
uses to analyze two consecutive windows in  
/// the audio data. The proportion must be in the range ` [0.0,  
1.0)`. Higher proportions generate more  
/// training data but also increase the training time.  
///  
/// The default value is `0.5`, which represents a 50% overlap.  
///  
/// - algorithm: An algorithm the training session uses to train  
the sound classifier.  
///  
/// - featureExtractionTimeWindowSize: A time duration, in  
seconds, the feature-extraction session uses for  
/// each audio sample it reads from an audio file in a dataset. The  
value must be in the range  
/// ` [0.5, 15.0]`.  
@available macOS 12.0 iOS 15.0  
@available  
public init
```

```
MLSoundClassifier ModelParameters ValidationData
                    Int
                    Double
```

```
MLSoundClassifier ModelParameters ModelAlgorithmType
```

```
TimeInterval
```

```
/// The type options for an Audio Feature Print feature extractor.
@available macOS 12.0 iOS 15.0
@available
public enum FeaturePrintType : Equatable
CustomStringConvertible Sendable

/// Creates a feature print type for a sound classifier.
case sound

/// A text representation of the feature-print type.
public var description : String { get }

/// Hashes the essential components of this value by feeding
them into the
/// given hasher.
///
/// Implement this method to conform to the `Hashable`
protocol. The
/// components used for hashing must be the same as the
components compared
/// in your type's `==` operator implementation. Call
`hasher.combine(_:)`
/// with each of these components.
///
/// - Important: In your implementation of `hash(into:)`,
/// don't call `finalize()` on the `hasher` instance
provided,
/// or replace it with a different instance.
/// Doing so may become a compile-time error in the future.
///
/// - Parameter hasher: The hasher to use when combining
the components
/// of this instance.
public func hash(inout Hasher)

/// Returns a Boolean value indicating whether two values are
equal.
///
/// Equality is the inverse of inequality. For any values `a` and
`b`,
/// `a == b` implies that `a != b` is `false`.
///
```

```

    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to compare.
public static func
MLSoundClassifier ModelParameters FeaturePrintType
MLSoundClassifier ModelParameters FeaturePrintType      Bool

    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different
executions of
    /// your program. Do not save hash values to use during a future
execution.
    ///
    /// - Important: `hashValue` is deprecated as a
`Hashable` requirement. To
        /// conform to `Hashable`, implement the `hash(into:)` requirement instead.
    /// The compiler provides an implementation for `hashValue` for you.
public var hashValue Int get

```

## **extension** MLSoundClassifier

```

    /// Parameters that affect the process of extracting sound features from
audio files.
@available macOS 11.0 iOS 15.0
@available
public struct FeatureExtractionParameters Sendable

    /// The proportion of overlap that the feature-extraction session uses to
analyze two consecutive windows in
    /// the audio data.
    ///
    /// The overlap factor — which must be in the range `[0.0, 1.0)` —
affects how much audio data the feature-
    /// extraction session analyzes in each file. Sessions with smaller
overlap factors read fewer audio data
    /// samples and finish in less time but may compromise the model's
prediction accuracy. Sessions with larger
    /// overlap factors read more audio samples for each file, which
increases the session's training data and
    /// processing time. The additional training data can improve a sound
classifier's accuracy; however, it may
    /// only be a modest improvement that isn't worth the extra processing
time.
    ///
    /// The feature-extraction session uses the expression

```

```

    ///`(featureExtractionTimeWindowSize * (1.0 -
overlapFactor))` to determine the how much to _step_ (advance)
    /// in time between samples.
    ///
    ///| Window size | Overlap factor | Step time |
    ///|-----|-----|-----|
    ///| `1.0` | `0.0` | `1.0` |
    ///| `2.0` | `0.0` | `2.0` |
    ///| `5.0` | `0.0` | `5.0` |
    ///| `1.0` | `0.5` | `0.5` |
    ///| `2.0` | `0.5` | `1.0` |
    ///| `5.0` | `0.5` | `2.5` |
    ///| `1.0` | `0.75` | `0.25` |
    ///| `2.0` | `0.75` | `0.5` |
    ///| `5.0` | `0.75` | `1.25` |
    ///
    /// For example, a session that's analyzing a 5-second audio file with a
window size of `1.0` and an overlap
    /// factor of `0.0` samples the audio five times. The time offsets for
those samples are: `0.0`, `1.0`, `2.0`,
    /// `3.0`, and `4.0`.
    ///
    /// Another session with a window size of `1.0` and an overlap factor
of `0.5` samples the same audio file 10
    /// times at half-second intervals. Unlike the first session, this session
samples each portion of audio data
    /// twice, except for the first and final half-second.
    ///
    /// A third session with a window size of `1.0` and an overlap factor of
`0.75` samples the same 5-second audio
    /// file 20 times at quarter-second intervals. This third session samples
most of the audio portions four times;
    /// the session samples the intervals near the ends one, two, or three
times.
public var overlapFactor Double

    /// A time duration, in seconds, that determines how much audio data
the feature-extraction session reads each
    /// time it samples an audio file.
    ///
    /// The time-window size defaults to `0.975` seconds and must be in
the range `[0.5, 15.0]`.
    ///
    /// Feature-extraction sessions that use
    ///
``MLSoundClassifier/ModelParameters-swift.struct/FeatureExtractorType/vggish(revision:)``
    /// ignore this value and always use a time-window size of `0.975`  

seconds.

```

**@available macOS 12.0 iOS 15.0**

```
@available
public var featureExtractionTimeWindowSize
TimeInterval

    /// The algorithm type the session uses to extract features from audio
files.
    public var featureExtractor
MLSoundClassifier ModelParameters FeatureExtractorType

    /// Creates the parameters for a feature-extraction session with a
default time window size.
    ///
    /// The initializer sets
``MLSoundClassifier/FeatureExtractionParameters/featureExtract
ionTimeWindowSize`` to a
    /// default value.
    ///
    /// - Parameters:
    /// - overlapFactor: A portion of overlap between consecutive
audio analysis windows. The value must be in
    /// the range `[0.0, 1.0)`.
    /// - featureExtractor: An algorithm type the session uses to
extract features from audio files.
public init Double
MLSoundClassifier ModelParameters FeatureExtractorType

    /// Creates the parameters for a feature-extraction session.
    ///
    /// - Parameters:
    /// - overlapFactor: A portion of overlap between consecutive
audio analysis windows. The value must be in
    /// the range `[0.0, 1.0)`.
    /// - featureExtractor: An algorithm type the session uses to
extract features from audio files.
    /// - featureExtractionTimeWindowSize: A time duration, in
seconds, the feature-extraction session uses for
    /// each audio sample it reads from an audio file in a dataset. The
value must be in the range
    /// ` [0.5, 15.0]`.
@available macOS 12.0 iOS 15.0
@available
public init Double
MLSoundClassifier ModelParameters FeatureExtractorType
TimeInterval
```

```
@available macOS 10.15 iOS 15.0
@available
extension MLSoundClassifier

    /// A representation of a sound-classifier dataset located in the file system or
    in a data table.
    ///
    /// Use a data source to represent a dataset for training, validating, or testing
    a sound classifier.
public enum DataSource

    /// Creates a data source from a folder with subfolders that each
    contain audio files.
    ///
    /// This data source uses each subdirectory's name as the label for the
    audio files contained within them. For
    /// example, for a directory that contains a `Laughter` subdirectory,
    the data source applies the label
    /// `""Laughter"` to each audio file in that subdirectory.
    ///
    /// swift
    /// // Build a URL to the directory that contains the
    labeled directories.
    /// let home =
    FileManager.default.homeDirectoryForCurrentUser
    /// let documents =
    home.appendingPathComponent("Documents")
    /// let labeledDirectories =
    documents.appendingPathComponent("Labeled Audio Directories")
    ///
    /// // Labeled Audio Directories/
    /// // | Laughter/
    /// // | | Laughter1.m4a
    /// // | | 20190229164259.m4a
    /// // | | .
    /// // | | .
    /// // | | .
    /// // | | AudienceLaughing.mp3
    /// // | | Applause
    /// // | | | misc-clapping.mp3
    /// // | | | 20190229164211.m4a
    /// // | | .
    /// // | | .
    /// // | | .
    /// // | | AudienceClapping.m4a
    ///
    /// // Create a data source from the labeled
    directories.
    /// let soundDataSource =
```

```

MLSoundClassifier.DataSource.labeledDirectories(at:
labeledDirectories)
    /**
     /// // Train a sound classifier with the data source.
     /// let soundClassifier = try
MLSoundClassifier(trainingData: soundDataSource)
    /**
    /**
    /// - Parameters:
    /// - at: URL : The URL to a folder in the file system that contains
folders of audio files. The data source
        /// uses the name of each folder as the classification label for the
audio content it contains.
case labeledDirectories URL

    /// Creates a data source from a folder that contains audio files, each
named after the sound they represent.
    /**
    /// Create a sound classifier data source from a directory of audio files
with the `labeledFiles` case. You
        /// must name each file with a sound classification label, followed by a
period and an arbitrary string, ending
            /// with the file extension. For example, you can name a sound
classifier's training files as `Laughter.3.png`,
        /// `Applause.1.jpg`, `Applause.2.jpg`, and so on.
    /**
    /// In this example, these audio file names give a sound classifier at
least two class labels:
    /**
    /// - `Laughter`
    /// - `Applause`
    /**
    /// - Parameters:
    /// - at: URL: The URL to a folder in the file system that contains
audio files. The data source uses the
        /// first component of each audio file's name as its classification
label.
case labeledFiles URL

    /// Creates a data source from a dictionary.
    /**
    /// This data source uses each key in the dictionary to label the audio
files in its associated URL array. The
        /// following code demonstrates how to create a dictionary with two
labels, `@"Laughter"` and `@"Applause"`.
    /**
    /// `` swift
    /// // Get the Documents directory URL.
    /// guard let documentsURL =
FileManager.default.urls(for: .documentDirectory,

```

```

    /**
in: .userDomainMask).first else {
    /**
     fatalError("Can't find Documents directory.")
    /**
    /**
     // Build a URL to the ~/Documents/Sounds
directory, which contains the training data.
    /**
    let url =
documentsURL.appendingPathComponent("Sounds")
    /**
    /**
     // Create a dictionary of arrays of audio file
URLs keyed by a label.
    /**
    let trainingData = [
    /**
        "Laughter": [
    /**
url.appendingPathComponent("Laughter.1.aif"),
    /**
url.appendingPathComponent("Laughter.2.wav")
    /**
        ],
    /**
        "Applause": [
    /**
url.appendingPathComponent("Applause.1.mp3"),
    /**
url.appendingPathComponent("Applause.2.caf")
    /**
        ]
    /**
    /**
    /**
        let soundClassifier = try
MLSoundClassifier(trainingData: trainingData)
    /**
    /**
    /**
        The value for each label key is an array of URLs to audio files of
laughter and applause, respectively.
    /**
    /**
        - Note: Use a minimum of 10 sound files per label to train a
sound classifier.
    /**
    /**
        - Parameters:
    /**
        - dictionary: A dictionary that contains a collection of
labeled audio files. Each of the dictionary's
    /**
        keys is a label, and each key's value is an array of audio-file
URLs.
    /**
        @available macOS 11.0 iOS 15.0
    /**
        @available
    /**
        case filesByLabel String URL
    /**
        /**
            Creates a data source from a data table of audio features.
    /**
    /**
        /**
            Use
````MLSoundClassifier/extractFeatures(trainingData:parameters:se

```

```
ssionParameters:``` to create an
    /// ``MLDataTable`` of audio features.
    ///
    /// - Parameters:
    ///   - table: An ``MLDataTable`` instance that contains
labeled audio data.
    ///   - featureColumn: The name of the column that contains the
audio features.
    ///   - labelColumn: The name of the column that contains the
audio labels.
    ///   - parameters: An
``MLSoundClassifier/FeatureExtractionParameters`` instance you use
to configure the
    ///     feature-extraction phase.
@available           11.0          14.0
@available           15.0          17.0
@available
case features      MLDataTable
String                               String
```

### MLSoundClassifier FeatureExtractionParameters

```
    /// Creates a data source from a data frame of audio features.
    ///
    /// Use
``MLSoundClassifier/extractFeatures(trainingData:parameters:se
ssionParameters:``` to create a
    ///
<doc://com.apple.documentation/documentation/tabulardata/dataframe> of audio
features.
    ///
    /// - Parameters:
    ///   - dataFrame: A data frame that contains labeled audio data.
    ///   - featureColumn: The name of the column that contains the
audio features.
    ///   - labelColumn: The name of the column that contains the
audio labels.
    ///   - parameters: An
``MLSoundClassifier/FeatureExtractionParameters`` instance you use
to configure the
    ///     feature-extraction phase.
@available macOS 12.0 iOS 15.0
@available
case featuresDataFrame DataFrame
String                               String
```

### MLSoundClassifier FeatureExtractionParameters

```

    /// Generates a dictionary of the data source's labeled audio files.
    ///
    /// - Returns: A dictionary of labeled audio files. Each dictionary
key is a label string and its value is an
    /// array of audio-file URLs.
public func labeledSounds throws String URL

    /// Generates an array of labeled audio dictionaries by splitting the data
source into strata.
    ///
    /// - Parameters:
    ///   - proportions: An array of proportions, each in the range
`[0.0, 1.0]`.
    ///   - seed: A seed for the random-number generator.
    ///
    /// - Returns: An array of dictionaries of labeled audio files. Each
dictionary key is a label string and its
    /// value is an array of audio-file URLs.
public func stratifiedSplit throws Double
Int String URL

    /// Generates an array of labeled audio dictionaries by splitting the data
source into strata using the
    /// random-number generator.
    ///
    /// - Parameters:
    ///   - proportions: An array of proportions, each in the range
`[0.0, 1.0]`.
    ///   - generator: A random-number generator.
    ///
    /// - Returns: An array of dictionaries of labeled audio files. Each
dictionary key is a label string and its
    /// value is an array of audio-file URLs.
public func stratifiedSplit RNG
Double inout RNG throws String URL
where RNG RandomNumberGenerator

```

**@available** macOS 10.15 iOS 15.0

**@available**

**extension** MLSoundClassifier CustomStringConvertible

/// A text representation of the sound classifier.

**public var** description String **get**

**@available** macOS 10.15 iOS 15.0

**@available**

**extension** MLSoundClassifier CustomDebugStringConvertible

```
    /// A text representation of the sound classifier that's suitable for output
    // during debugging.
    public var debugDescription String get

    @available macOS 10.15 iOS 15.0
    @available
    extension MLSoundClassifier
    CustomPlaygroundDisplayConvertible

    /// A description of the sound classifier in a playground.
    public var playgroundDescription Any get

    @available macOS 10.15 iOS 15.0
    @available
    extension MLSoundClassifier ModelParameters
    CustomStringConvertible

    /// A text representation of the model parameters.
    public var description String get

    @available macOS 10.15 iOS 15.0
    @available
    extension MLSoundClassifier ModelParameters
    CustomDebugStringConvertible

    /// A text representation of the model parameters that's suitable for output
    // during debugging.
    public var debugDescription String get

    @available macOS 10.15 iOS 15.0
    @available
    extension MLSoundClassifier ModelParameters
    CustomPlaygroundDisplayConvertible

    /// A description of the parameters in a playground.
    public var playgroundDescription Any get

extension MLSoundClassifier ModelParameters

    /// The classifier options for a sound classifier training algorithm.
    @available macOS 11.0 iOS 15.0
    public enum ClassifierType : Equatable, Hashable
    CustomStringConvertible, Sendable
```

```

    /// A statistical model that uses logistic regression to classify an input
vector into a category.
case logisticRegressor

    /// A neural network model that uses three or more layers to classify an
input into a category.
    /**
     * The neural network has a minimum of three layers:
     */
    /**
     * - An input layer
     * - One or more hidden layers
     * - An output layer
     */
    /**
     * The number of integers in your `layerSizes` array determines
the number of hidden layers in the neural
     * network. Each integer in the array determines the size of that hidden
layer.
    /**
     * - Parameters:
     * - layerSizes: An array of positive integers. Each element
represents the number of units for that hidden
     * layer.
@available macOS 11.0
@available
@available
case multilayerPerceptron           Int

    /// A text representation of the classifier type.
public var description String get

    /// Returns a Boolean value indicating whether two values are equal.
    /**
     * Equality is the inverse of inequality. For any values `a` and `b`,
     * `a == b` implies that `a != b` is `false`.
    /**
     * - Parameters:
     * - lhs: A value to compare.
     * - rhs: Another value to compare.
public static func
MLSoundClassifier ModelParameters ClassifierType
MLSoundClassifier ModelParameters ClassifierType      Bool

    /// Hashes the essential components of this value by feeding them into
the
    /// given hasher.
    /**
     * Implement this method to conform to the `Hashable` protocol. The
     * components used for hashing must be the same as the components
compared
     * in your type's `==` operator implementation. Call

```

```

`hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your implementation of `hash(into:)`,  

    /// don't call `finalize()` on the `hasher` instance provided,  

    /// or replace it with a different instance.  

    /// Doing so may become a compile-time error in the future.
    ///
    /// - Parameter hasher: The hasher to use when combining the  

components
    /// of this instance.
public func hash           inout Hasher

    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different  

executions of
    /// your program. Do not save hash values to use during a future  

execution.
    ///
    /// - Important: `hashValue` is deprecated as a `Hashable`  

requirement. To
    /// conform to `Hashable`, implement the `hash(into:)`  

requirement instead.
    /// The compiler provides an implementation for `hashValue` for  

you.
public var hashValue Int   get

```

## **extension** MLSoundClassifier ModelParameters

```

    /// The feature-extractor options for a sound-classifier training algorithm.
    ///
    /// Use the ``MLSoundClassifier/ModelParameters-  

    /// swift.struct/FeatureExtractorType/  

    /// audioFeaturePrint(type:revision)``
    /// feature extractor to create a model with the following advantages over  

`vggish`:
    ///
    /// - More accurate predictions
    /// - Lower latency
    /// - Smaller model file size
    /// - Less training time
    ///
    /// Use
``MLSoundClassifier/ModelParameters-swift.struct/FeatureExtractorType/vggish(revision)``
    /// to support older
    /// OS versions.
@available macOS 11.0  iOS 15.0

```

```
@available
public enum FeatureExtractorType : Equatable
CustomStringConvertible, Sendable

    /// Represents the VGGish feature extractor, which is compatible with
    older OS versions.
    ///
    /// The case uses the newest version if you don't provide an associated
    value for `revision`.
    ///
    /// - Parameters:
    ///   - revision: A version of the VGGish feature extractor.
case vggish : Int = 1

    /// Represents the Audio Feature Print extractor.
    ///
    /// The case uses the newest version of
    ``MLSoundClassifier/ModelParameters-swift.struct/FeaturePrintT
    ype/sound``
    /// if you don't provide associated values for `type` and
    `revision`.
    ///
    /// - Parameters:
    ///   - type: An Audio Feature Print extractor type.
    ///   - revision: A version of the extractor you pass to `type`.
@available macOS 12.0 iOS 15.0
@available
case audioFeaturePrint
MLSoundClassifier ModelParameters FeaturePrintType
Int = 1

    /// A text representation of the feature-extractor type.
public var description : String { get }

    /// Returns a Boolean value indicating whether two values are equal.
    ///
    /// Equality is the inverse of inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is `false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to compare.
public static func
MLSoundClassifier ModelParameters FeatureExtractorType
MLSoundClassifier ModelParameters FeatureExtractorType
Bool
```

```

extension MLSoundClassifier ModelParameters

    /// The algorithm options to train a sound classifier.
    @available macOS 11.0 iOS 15.0
    @available
    public enum ModelAlgorithmType : Equatable
    CustomStringConvertible Sendable

        /// An algorithm that leverages the knowledge of a general-purpose
model built into the operating system.
        ///
        /// You typically use this transfer-learning algorithm to train an object
detector in these situations:
        ///
        /// – Your training dataset has a limited number of examples.
        /// – You prefer your object detector's Core ML model file to be as
small as possible.
        ///
        /// – Parameters:
        /// – featureExtractor: The extractor type the algorithm uses
to detect features from the audio data.
        /// – classifier: The model type the algorithm uses to classify
audio data.
        case transferLearning
    MLSoundClassifier ModelParameters FeatureExtractorType
    MLSoundClassifier ModelParameters ClassifierType

        /// A text representation of the training algorithm.
        public var description : String get

            /// Returns a Boolean value indicating whether two values are equal.
            ///
            /// Equality is the inverse of inequality. For any values `a` and `b`,
            /// `a == b` implies that `a != b` is `false`.
            ///
            /// – Parameters:
            /// – lhs: A value to compare.
            /// – rhs: Another value to compare.
        public static func
    MLSoundClassifier ModelParameters ModelAlgorithmType
    MLSoundClassifier ModelParameters ModelAlgorithmType : Bool

```

**extension** MLSoundClassifier ModelParameters

```

    /// The source of a validation dataset for a sound classifier.
    @available macOS 10.15 iOS 15.0
    @available
    public enum ValidationData

```

```

    /// A validation dataset derived by randomly selecting a portion of the
sound classifier's training dataset
    /// using the split strategy.
    ///
    /// - Parameters:
    /// - strategy: The partition method this case uses to create the
validation dataset from the training dataset.
case split      MLSplitStrategy

    /// A validation dataset represented by a data source.
    ///
    /// - Parameters:
    /// - dataSource: An ``MLSoundClassifier/DataSource`` instance that contains a validation dataset.
@available macOS 11.0 iOS 15.0
@available
case dataSource MLSoundClassifier DataSource

    /// A validation dataset represented by a dictionary.
    ///
    /// - Parameters:
    /// - dictionary: A validation dataset that uses a collection of labeled audio files represented by a
    /// dictionary. Each key of the dictionary is a label, and its value is an array of audio-file URLs.
@available                      10.15          11.0
    "Use DataSource.filesByLabel to provide dictionary validation data instead."
@available                      15.0          16.0
    "Use DataSource.filesByLabel to provide dictionary validation data instead."
@available
case dictionary String      URL

    /// An empty validation dataset that skips the model validation
    /// phase after training.
case none

```

```

@available macOS 12.0 iOS 15.0
@available
extension MLSoundClassifier ModelParameters FeaturePrintType
Hashable

```

```

    /// Data partitioning approaches, typically for creating a validation dataset
    /// from a training dataset.
@available macOS 10.15 iOS 15.0 tvOS 16.0

```

```
public enum MLSplitStrategy : Sendable

    /// Create ML automatically decides how much of your training dataset it
    /// uses for a validation dataset.
    ///
    /// Create ML typically creates a validation dataset by partitioning up to
    /// 10% from the training dataset, depending on its size:
    ///
    /// | Training samples | % used for validation |
    /// | ----- | ----- |
    /// | < 50 | None |
    /// | 50 to 199 | 10% |
    /// | ≥ 200 | 5% |
    case automatic

    /// Create ML uses a portion of your training dataset to create a validation
    /// dataset based on the ratio.
    ///
    /// Use a value in the range `[0.0, 1.0]` for `ratio`. Use any value for
    /// `seed`, including the return value from ``timestampSeed()`` , to tell
    /// Create ML how to pseudorandomly select data from your training dataset.
    /// Otherwise, set `seed` to `nil` to tell Create ML to choose a number
for
    /// you.
    case fixed Double Int

    /// Resolves this split strategy for a specific element count.
    ///
    /// - Parameter count: The number of elements in the collection being
split.
    /// - Returns: The split fraction and the random seed to use.
    @available(macOS 11.0, iOS 15.0, tvOS 16.0)
    public func resolve Int Double
Int

/// A sequence of image visualizations for machine learning types.
@available(macOS 10.15)
@available
@available
public protocol MLStreamingVisualizable : MLVisualizable

    /// Advances the visualization stream to the next iteration.
    mutating func nextIteration

    /// A Boolean value that indicates whether the stream has provided its final
    /// iteration.
    ///
    /// This method updates ``MLVisualizable/cgImage`` to the next
iteration.
```

```
var hasFinishedStreaming Bool get

/// A model you train to apply an image's style to other images or videos.
@available macOS 11.0 iOS 15.0
@available
public struct MLStyleTransfer Sendable

/// Creates a style transfer model with a training dataset represented by a
/// data source.
///
/// - Parameters:
/// - trainingData: A style image and a content image, represented by
a data
/// source.
///
/// - parameters: An ``MLStyleTransfer/ModelParameters``
instance you use to
/// configure the model for the training session.
public init MLStyleTransfer DataSource
MLStyleTransfer ModelParameters throws

/// Creates a style transfer model from a training session checkpoint.
///
/// You can only use a checkpoint from a style transfer model's training
/// session if its ``MLCheckpoint/phase`` property is
``MLPhase/training``.
///
/// - Parameters:
/// - checkpoint: A checkpoint from a style transfer training session.
public init MLCheckpoint throws

/// Begins an asynchronous style transfer model-training session.
///
/// - Parameters:
/// - trainingData: The style image and content images represented by
a data
/// source.
///
/// - parameters: An ``MLStyleTransfer/ModelParameters``
instance you use to
/// configure the model for the training session.
///
/// - sessionParameters: An ``MLTrainingSessionParameters``
instance you use
/// to configure the training session.
///
/// - Returns: An ``MLJob`` that represents the style transfer
/// model-training session.
public static func train
```

```
MLStyleTransfer DataSource
MLStyleTransfer ModelParameters
MLTrainingSessionParameters           throws
MLJob MLStyleTransfer

    /// Creates an asynchronous training session for a style transfer model.
    ///
    /// - Parameters:
    /// - trainingData: The style image and content images represented by
    a data
    /// source.
    ///
    /// - parameters: An ``MLStyleTransfer/ModelParameters`` instance you use to
    /// configure the model for the training session.
    ///
    /// - sessionParameters: An ``MLTrainingSessionParameters`` instance you use
    /// to configure the training session.
    ///
    /// - Returns: An ``MLTrainingSession`` that represents the style
    transfer
    /// model-training session.
public static func makeTrainingSession
MLStyleTransfer DataSource
MLStyleTransfer ModelParameters
MLTrainingSessionParameters           throws
MLTrainingSession MLStyleTransfer

    /// Creates an asynchronous training session for a style transfer model by
    /// restoring an existing training session's state from its parameters.
    ///
    /// - Parameters:
    /// - sessionParameters: The
    ``MLTrainingSessionParameters`` instance you
    /// used to create the training session using
    ///
    ``MLStyleTransfer/makeTrainingSession(trainingData:parameters:
    sessionParameters)``.
    ///
    /// - Returns: An ``MLTrainingSession`` that represents the style
    transfer
    /// model-training session.
public static func
restoreTrainingSession
MLTrainingSessionParameters  throws
MLTrainingSession MLStyleTransfer

    /// Begins or continues an asynchronous style transfer model-training
    /// session.
```

```
///  
/// - Parameters:  
/// - session: An ``MLTrainingSession`` instance that represents  
the  
/// training session.  
///  
/// - Returns: An ``MLJob`` that represents the style transfer  
/// model-training session.  
public static func resume _  
MLTrainingSession MLStyleTransfer throws  
MLJob MLStyleTransfer  
  
/// Applies the style the model learned to an image.  
///  
/// - Parameters:  
/// - image: An input image the model applies its style to.  
///  
/// - Throws: `MLCreateError.generic` if stylization fails.  
///  
/// - Returns: An image of type `CGImage` stylized using style of the  
model.  
public func stylize CGImage throws CGImage  
  
/// Exports the style transfer model as a Core ML model file to a location  
/// in the file system.  
///  
/// - Parameters:  
/// - fileURL: The location URL in the file system where you want to  
save  
/// the model.  
///  
/// - metadata: Descriptive information to include with the exported model  
/// file.  
public func write URL  
MLModelMetadata throws  
  
/// Exports the style transfer model as a Core ML model file to the file  
/// path.  
///  
/// - Parameters:  
/// - path: The location path in the file system where you want to save the  
/// model.  
///  
/// - metadata: Descriptive information to include with the exported model  
/// file.  
public func write String  
MLModelMetadata throws  
  
/// Initiates a download of the mlmodel assets required for Style Transfer  
training. This will be performed
```

```

    /// automatically if needed at training time, but can be run independently prior
    to training.
    @available macOS 12.0  iOS 15.0
    @available
    public static func downloadAssets    throws

    /// A data source for a style transfer model.
    public enum DataSource    Sendable

        /// The locations of a style-image file and content-image directory in
        /// the file system.
        case images          URL          URL
                           VNImageCropAndScaleOption

        /// Converts the content images to square images and saves them to a
        /// destination directory.
        ///
        /// - Parameters:
        /// - textelDensity: The length of a side, in pixels, of the
destination
        /// image. The value must be a multiple of `4` in the range `[64,
        /// 1024]`.
        ///
        /// - styleImageDestination: A location in the file system to save the
        /// converted style image to. If `nil`, Create ML saves the image to a
        /// temporary file location.
        ///
        /// - contentImagesDestination: A location in the file system to save
        /// the converted content image to. If `nil`, Create ML saves the
image
        /// to a temporary file location.
        ///
        /// - Returns: A tuple of two URLs:
        ///
        /// - term `processedStyleImage` : The URL to the converted
style image.
        /// - term `processedContentImages` : The URL to the converted
content
        /// image.
        ///
        ///

        public func processImages           Int
                           URL      nil
URL      nil    throws
                           URL
                           URL

    /// Parameters that affect the training process of a style transfer model.
    public struct ModelParameters    Sendable

```

```
    /// The style transfer task's training algorithm that prioritizes either
    /// speed or quality.
    ///
    /// You choose the task's training algorithm based on your intended use
    /// for this model.
    ///
    /// Use
``MLStyleTransfer/ModelParameters/ModelAlgorithmType/cnnLite``
    /// to train a model that prioritizes speed to stylize images with low
    /// latency, typically for frames of a video stream. Otherwise, select
    ///
``MLStyleTransfer/ModelParameters/ModelAlgorithmType/cnn`` to
train
    /// a model that applies a higher-quality stylization, typically for a
    /// still image.
public var algorithm
MLStyleTransfer ModelParameters ModelAlgorithmType

    /// The style transfer model's validation dataset.
public var validation
MLStyleTransfer ModelParameters ValidationData

    /// The largest number of iterations the style transfer model can use
    /// during training.
public var maxIterations Int

    /// The amount of detail the task applies from the input style image to
    /// the stylized image output.
    ///
    /// The value must be a multiple of `4` in the range `[64, 1024]`.
The
    /// style transfer task converts all other values to the nearest
    /// multiple of `4` less than the value.
public var textelDensity Int

    /// The amount of influence the input style image has in the stylized
    /// image output.
    ///
    /// This parameter's value must be in the range `[1, 10]`. Higher
    /// numbers apply more of the style image than the content image.
public var styleStrength Int

    /// Creates a new set of training parameters for a style transfer model.
    ///
    /// - Parameters:
    ///   - algorithm: The style transfer task's training algorithm that
    ///     prioritizes either speed or quality.
    ///
    ///   - validation: The style transfer model's validation dataset.
    /// 
```

```
    /// - maxIterations: The largest number of training iterations the style
    /// transfer model can use.
    ///
    /// - textureDensity: The amount of detail the task applies from the
    /// input style image to the stylized image output.
    ///
    /// - styleStrength: The amount of influence the style image has in the
    /// stylized image output.
public init
MLStyleTransfer ModelParameters ModelAlgorithmType
    MLStyleTransfer ModelParameters ValidationData
        Int
        Int
        Int
        Int

    /// The source of a validation dataset for a style transfer model.
public enum ValidationData Sendable

    /// The location of a validation image you use to validate the
    /// model.
case content URL

    /// An empty validation dataset you use to skip model validation.
case none

    /// The style transfer training algorithm options.
    ///
    /// This object uses a convolutional neural network for training the style
    transfer model, giving more detailed style
    /// It's ideal for cases where latency is not the main
    /// concern, such as single images.
public enum ModelAlgorithmType String Sendable

    /// A style-transfer training algorithm that generates a model that
    /// prioritizes image quality over speed.
case cnn

    /// A style-transfer training algorithm that generates a model that
    /// prioritizes speed over image quality.
case cnnLite

    /// Creates a new instance with the specified raw value.
    ///
    /// If there is no value of the type that corresponds with the
specified raw
    /// value, this initializer returns `nil`. For example:
    ///
    ///     enum PaperSize: String {
```

```
        ///      case A4, A5, Letter, Legal
        /// }
        ///
        /// print(PaperSize(rawValue: "Legal"))
        // Prints "Optional("PaperSize.Legal")"
        ///
        /// print(PaperSize(rawValue: "Tabloid"))
        // Prints "nil"
        ///
        /// - Parameter rawValue: The raw value to use for the new
instance.
```

```
public init String

/// The raw type that can be used to represent all values of the
conforming
/// type.
///
/// Every distinct value of the conforming type has a
corresponding unique
/// value of the `RawValue` type, but there may be values of the
`RawValue`
/// type that don't have a corresponding value of the conforming
type.
@available iOS 15.0 macOS 11.0
@available
public typealias RawValue String
```

```
/// The corresponding value of the raw type.
///
/// A new instance initialized with `rawValue` will be equivalent
to this
/// instance. For example:
///
/// enum PaperSize: String {
///     case A4, A5, Letter, Legal
/// }
///
/// let selectedSize = PaperSize.Letter
/// print(selectedSize.rawValue)
// Prints "Letter"
///
/// print(selectedSize == PaperSize(rawValue:
selectedSize.rawValue!))
// Prints "true"
public var rawValue String get
```

```
@available macOS 11.0 iOS 15.0
```

```
@available
extension MLStyleTransfer : CustomStringConvertible
CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of the style transfer model.
public var description : String { get }

    /// A text representation of the style transfer model that's suitable for
    /// output during debugging.
public var debugDescription : String { get }

    /// A description of the style transfer model shown in a playground.
public var playgroundDescription : Any { get }

@available macOS 11.0 iOS 15.0
@available
extension MLStyleTransfer : ModelParameters
CustomStringConvertible : CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of the style transfer model parameters.
public var description : String { get }

    /// A text representation of the style transfer model parameters that's
    /// suitable for output during debugging.
public var debugDescription : String { get }

    /// A description of the style transfer model parameters shown in a
    /// playground.
public var playgroundDescription : Any { get }

@available macOS 11.0 iOS 15.0
@available
extension MLStyleTransfer : ModelParameters, ValidationData, Equatable

    /// Returns a Boolean value indicating whether two values are equal.
    ///
    /// Equality is the inverse of inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is `false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to compare.
public static func
MLStyleTransfer(ModelParameters, ValidationData)
```

```
MLStyleTransfer ModelParameters ValidationData      Bool

@available macOS 11.0  iOS 15.0
@available
extension MLStyleTransfer ModelParameters ModelAlgorithmType
CustomStringConvertible CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of the model parameters.
public var description String get

    /// A text representation of the model parameters that's suitable for output
    /// during debugging.
public var debugDescription String get

    /// A description of the model parameters shown in a playground.
public var playgroundDescription Any get

@available macOS 11.0  iOS 15.0
@available
extension MLStyleTransfer ModelParameters ModelAlgorithmType
Equatable

@available macOS 11.0  iOS 15.0
@available
extension MLStyleTransfer ModelParameters ModelAlgorithmType
Hashable

@available macOS 11.0  iOS 15.0
@available
extension MLStyleTransfer ModelParameters ModelAlgorithmType
RawRepresentable

    /// A classifier that predicts a binary target value by maximizing the separation
    /// between categories.
@available                         10.14          14.0
@available
@available
public struct MLSupportVectorClassifier @unchecked Sendable

    /// The Core ML model.
public var model MLModel
```

```

    ///> The name of the column you selected at initialization to define which
    categories the classifier predicts.
    /**
     * Changing the value of this property doesn't retrain the model or affect its
     * behavior.
    */
    public var targetColumn String

    /**
     * The names of the columns you selected at initialization to train the
     * classifier.
    */
    /**
     * Changing the value of this property doesn't retrain the model or affect its
     * behavior.
    */
    public var featureColumns String

    /**
     * The underlying parameters used when training the model.
    */
    public let modelParameters
    MLSupportVectorClassifier ModelParameters

    /**
     * Measurements of the classifier's performance on the training data set.
    */
    public var trainingMetrics MLClassifierMetrics get

    /**
     * Measurements of the classifier's performance on the validation data set.
    */
    public var validationMetrics MLClassifierMetrics get

    /**
     * Creates a support vector classifier.
    */
    /**
     * - Parameters:
     *   - trainingData: The training data
     *   - targetColumn: Name of the column containing the class labels
     *   - featureColumns: Names of the columns containing feature
     * values. If `nil` all columns, other than the target
     *   column, will be used as feature values.
     *   - parameters: Model training parameters
    */
    @available(12.0, 14.0)
    @available
    @available
    public init DataFrame String
        String nil
    MLSupportVectorClassifier ModelParameters

    throws

    /**
     * Creates a Support Vector Classifier from the feature columns in the
     * training data to predict the categories in
     * the target column.
    */
    /**
     * - Parameters:
     *   - trainingData: A data table of training examples.
     *   - targetColumn: The column name for the values in the training
     * data that the classifier should predict.
    */

```

```

    ///> - featureColumns: The column names for the values in the
    // training data that the classifier uses to predict
    ///> the target value.
    ///> - parameters: The model parameters.
    @available(10.14, 13.0)
    "Use DataFrame instead of MLDataTable when
initializing."
    @available
    @available
    public init
    String
    String
    MLDataTable
    ModelParameters
    nil
    throws

    ///> Predicts a column of labels for the given testing data.
    @available(12.0, 14.0)
    @available
    @available
    public func predictions
    AnyColumn
    DataFrame
    throws

    ///> Classifies the provided data into the target categories.
    ///
    ///> - Parameters:
    ///> - data: The data you want the model to classify.
    ///
    ///> - Returns: A column of labels predicted by the classifier.
    @available(10.14, 13.0)
    "Use DataFrame instead of MLDataTable."
    @available
    @available
    public func predictions
    MLUntypedColumn
    MLDataTable
    throws

    ///> Evaluates the classifier on the provided labeled data.
    ///
    ///> Evaluation should be done on a testing data set that the model has not
    // seen as part of the training or
    ///> validation data sets. The data should have feature columns with identical
    // name and type to the
    ///> training data, as well as a labels column with the same name.
    ///
    ///> - Parameters:
    ///> - labeledData: A `DataFrame` to evaluate the trained model on.
    ///
    ///> - Returns: Metrics that describe the classification errors
    ///(``MLClassifierMetrics/classificationError``), the precision
    // and recall
    ///(``MLClassifierMetrics/precisionRecall``), and
    // a table that

```

```

    ///>     /// describes how labels were misapplied
    ///>     (``MLClassifierMetrics/confusion``)
    ///>     /// on the provided data.
    @available(12.0, 14.0)
    @available
    @available
    public func evaluation(DataFrame) MLClassifierMetrics

    ///>     /// Evaluates the classifier on the provided labeled data.
    ///>
    ///>     /// Evaluation should be done on a testing data set that the model has not
    seen as part of the training or
    ///>     /// validation data sets. The data should have feature columns with identical
name and type to the
    ///>     /// training data, as well as a labels column with the same name.
    ///>
    ///>     /// - Parameters:
    ///>     ///   - labeledData: An `MLDataTable` to evaluate the trained model
on.
    ///>
    ///>     /// - Returns: Metrics that describe the classification errors
    ///>     /// (``MLClassifierMetrics/classificationError``), the precision
and recall
    ///>     /// percentages (``MLClassifierMetrics/precisionRecall``), and
a table that
    ///>     /// describes how labels were misapplied
    ///>     (``MLClassifierMetrics/confusion``)
    ///>     /// on the provided data.
    @available(10.14, 13.0)
    "Use DataFrame instead of MLDataTable."
    @available
    @available
    public func evaluation(MLDataTable) MLClassifierMetrics

    ///>     /// Exports a Core ML model file for use in your app.
    public func write(URL) MLModelMetadata nil throws

    ///>     /// Exports a Core ML model file for use in your app.
    public func write(String) MLModelMetadata nil throws

@available(10.14, 14.0)
@available
@available
extension MLSupportVectorClassifier

```

```
/// Parameters that affect the process of training a model.
@available(10.14, 14.0)
@available
@available
public struct ModelParameters

    public var maxIterations Int

    /// Validation data represented as a `MLDataTable`.
    ///
    /// - Note: Setting this to `nil` means that the training data will be
    automatically split for
    /// validation. Setting it to an empty table means to not use a
validation set.
    @available(10.14, 14.0)
    @available("Use the validation property instead.")
    @available
    @available
    public var validationData MLDataTable

    /// Validation data.
    ///
    /// The default is `split(strategy: .automatic)`, which
    automatically generates the validation
    /// dataset from 0% to 10% of the training dataset.
    @available(10.15, 14.0)
    @available
    @available
    @available
    public var validation
MLSupportVectorClassifier ModelParameters ValidationData

    public var penalty Double

    public var convergenceThreshold Double

    public var featureRescaling Bool

    /// Creates a new set of parameters.
    ///
    /// - Parameters:
    ///   - validation: The data used to monitor how well the model
    is generalizing.
    ///
    ///   The default is to automatically split off some data from the
    training set for validation.
    ///
    ///   - maxIterations: The maximum number of passes through
    the data.
    ///
    ///   The default value is 11.
```

```

    /**
     * - penalty: Weight of the regularizer. The larger the penalty
     * the less variance in the model.
     */
    /**
     * The default value is 1.0.
     */
    /**
     * - convergenceThreshold: The threshold with which to
     * determine if the model has converged. Consider
     * reducing this value for higher training accuracy, but beware of
     * overfitting.
     */
    /**
     * The default value is 0.01.
     */
    /**
     * - featureRescaling: Determines if the features should be
     * preprocessed to ensure all features are on the
     * same scale.
     */
    /**
     * The default value is true.
     */
@available 10.15 14.0
@available
@available
public init
MLSupportVectorClassifier ModelParameters ValidationData
Double 1.0 Int 11
Bool true Double 0.01

    /**
     * Creates a new set of parameters.
     */
    /**
     * - Parameters:
     * - validationData: The dataset used to monitor how well the
     * model is generalizing.
     */
    /**
     * The default value is `nil` which will use an automatically
     * sampled validation set.
     */
    /**
     * - maxIterations: The maximum number of passes through
     * the data.
     */
    /**
     * The default value is 11.
     */
    /**
     * - penalty: Weight of the regularizer. The larger the penalty
     * the less variance in the model.
     */
    /**
     * The default value is 1.0.
     */
    /**
     * - convergenceThreshold: The threshold with which to
     * determine if the model has converged. Consider
     * reducing this value for higher training accuracy, but beware of
     * overfitting.
     */

```

```
    ///      The default value is 0.01.  
    ///  
    ///      - featureRescaling: Determines if the features should be  
    // preprocessed to ensure all features are on the  
    ///      same scale.  
    ///  
    ///      The default value is true.  
10.15  @available          10.14  
        "Use the validation property instead."  
@available  
@available  
public init          MLDataTable  
           Int   11           Double  1.0  
           Double  0.01           Bool  
true
```

```
@available          10.14          14.0  
@available  
@available  
extension MLSupportVectorClassifier  CustomStringConvertible
```

```
    /// A text representation of the support vector classifier.  
public var description  String  get
```

```
@available          10.14          14.0  
@available  
@available  
extension MLSupportVectorClassifier  CustomDebugStringConvertible
```

```
    /// A text representation of the support vector classifier that's suitable for  
    // output during debugging.  
public var debugDescription  String  get
```

```
@available          10.14          14.0  
@available  
@available  
extension MLSupportVectorClassifier  CustomPlaygroundDisplayConvertible
```

```
    /// A description of the support vector classifier shown in a playground.  
public var playgroundDescription  Any  get
```

```
@available          10.14          14.0
```

```
@available
@available
extension MLSupportVectorClassifier ModelParameters
CustomStringConvertible CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of the model parameters for a support vector
    /// classifier.
public var description String get

    /// A text representation of the model parameters for a support vector
    /// classifier that's suitable for output during debugging.
public var debugDescription String get

    /// A description of the model parameters for a support vector classifier
    /// shown in a playground.
public var playgroundDescription Any get
```

|  |                 |             |
|--|-----------------|-------------|
| <b>@available</b>  | <b>10.15</b>    | <b>14.0</b> |
| <b>@available</b>  |                 |             |
| <b>@available</b>  |                 |             |
| <b>extension</b> MLSupportVectorClassifier ModelParameters                         |                 |             |
| <br>   |                 |             |
| <b>/// Values for specifying validation data.</b>                                  |                 |             |
| <b>@available</b>  | <b>10.15</b>    | <b>14.0</b> |
| <b>@available</b>  |                 |             |
| <b>@available</b>  |                 |             |
| <b>public enum</b> ValidationData  |                 |             |
| <br>   |                 |             |
| <b>/// Generate validation data by splitting the training dataset. This is the</b> |                 |             |
| <b>default.</b>  |                 |             |
| <b>case</b> split  | MLSplitStrategy |             |
| <br>   |                 |             |
| <b>/// Set validation data from the MLDataTable provided.</b>                      |                 |             |
| <b>@available</b>  | <b>10.15</b>    | <b>14.0</b> |
| <b>@available</b>  |                 |             |
| <b>@available</b>  |                 |             |
| <b>case</b> table MLDataTable  |                 |             |
| <br>   |                 |             |
| <b>/// Validation data provided in a DataFrame.</b>                                |                 |             |
| <b>@available macOS 12.0</b>   |                 |             |
| <b>case</b> dataFrame DataFrame  |                 |             |
| <br>   |                 |             |
| <b>/// Do not set validation data.</b>   |                 |             |
| <b>case</b> none   |                 |             |

```
/// A model you train to classify natural language text.  
///  
/// Use a text classifier to train a machine learning model you can include in  
/// your app to classify natural language text. The model learns to associate  
/// labels with features of the input text, which can be sentences, paragraphs,  
/// or even entire documents.  
///  
/// After you train a text classifier, you save it to a Core ML model file. You  
/// then use an instance of the  
/// <doc://com.apple.documentation/documentation/naturallanguage/nlmodel>  
class  
/// from the <doc://com.apple.documentation/documentation/naturallanguage>  
/// framework to read the model file into your app.  
@available macOS 10.14 iOS 15.0  
@available  
public struct MLTextClassifier @unchecked Sendable  
  
/// The underlying Core ML model of the text classifier.  
public var model MLMModel  
  
/// The configuration parameters that the text classifier used for training  
during initialization.  
public let modelParameters  
MLTextClassifier ModelParameters  
  
/// Measurements of the classifier's performance on the training data set.  
public let trainingMetrics MLClassifierMetrics  
  
/// Measurements of the classifier's performance on the validation data set.  
public let validationMetrics MLClassifierMetrics  
  
/// Creates a text classifier.  
///  
/// – Parameter trainingData: A data source specifying the training  
data.  
/// – Parameter parameters: Model training parameters.  
@available macOS 10.15 iOS 15.0  
@available  
public init MLTextClassifier DataSource  
MLTextClassifier ModelParameters  
  
throws  
  
/// Creates a text classifier.  
///  
/// – Parameter trainingData: A dictionary specifying the training  
data.  
/// – Parameter parameters: Model training parameters.  
@available macOS 10.15 iOS 15.0  
@available
```

```
public init String String
MLTextClassifier ModelParameters
throws

/// Creates a text classifier.
///
/// - Parameter trainingData: A data frame specifying the training
data.
/// - Parameter textColumn: The name of the text column in the
provided trainingData.
/// - Parameter labelColumn: The name of the label column in the
provided trainingData.
/// - Parameter parameters: Model training parameters.
@available macOS 13.0 iOS 16.0
@available
public init DataFrame String
String
MLTextClassifier ModelParameters
throws

/// Creates a text classifier.
///
/// - Parameter trainingData: A table specifying training data
/// - Parameter textColumn: name of the text column in the provided
trainingData
/// - Parameter labelColumn: name of the label column in the
provided trainingData
/// - Parameter parameters: Model training parameters.
@available 10.14 13.0
    "Use DataSource instead of MLDataTable when
initializing."
@available 15.0 16.0
    "Use DataFrame instead of MLDataTable when
initializing."
@available
public init MLDataTable String
String
MLTextClassifier ModelParameters
nil throws

@available macOS 10.14 iOS 15.0
@available
extension MLTextClassifier

/// Exports the text classifier as a Core ML model file at the specified URL.
///
/// - Parameters:
///   - fileURL: The location in the file system to which the file should be
written.
```

```
    /// - metadata: Descriptive information to include with the exported
model file.
public func write URL
MLModelMetadata nil throws

    /// Exports the text classifier as a Core ML model file at the specified file
path.
    ///
    /// - Parameters:
    ///   - path: A file system path where the model file should be written.
    ///   - metadata: Descriptive information to include with the exported
model file.
public func write String
MLModelMetadata nil throws

@available macOS 10.14 iOS 15.0
@available
extension MLTextClassifier

    /// A data source for a text classifier.
public enum DataSource

    /// A root directory of labeled directories for your data set.
    ///
    /// Labeled directories can be used to organize textual inputs. Just
    /// like the data source used with the image classifier, place the
    /// collected text inputs of the same kind in a directory. Name that
    /// directory with the label appropriate for the textual inputs.
case labeledDirectories URL

    /// Fetches the labeled data from the data source.
    ///
    /// - Returns: A dictionary that contains each label with their related
text entries.
public func labeledTexts throws String
String

@available macOS 10.14 iOS 15.0
@available
extension MLTextClassifier

    /// The text feature extractor type.
@available macOS 10.15 iOS 15.0
@available
public enum FeatureExtractorType Sendable

    /// A feature extractor that uses the standard, built-in word embeddings.
```

```

    /**
     * The standard word embeddings are the same as those that
     */
<doc://com.apple.documentation/documentation/naturallanguage/nlembdding>
    /**
     * provides with its
     */
<doc://com.apple.documentation/documentation/naturallanguage/nlembdding/
3074352-wordembedding>
    /**
     * method.
case staticEmbedding

    /**
     * A feature extractor that provides embeddings for words, based on
their in-context use.
    /**
     * Dynamic embedding requires certain downloadable assets to be
present on device at training time.
     * Training will throw an error if the specified language is unavailable at
runtime. Asset downloads are
     * managed in the background automatically by the OS when a new
language is configured in device
     * settings, such as when adding a new keyboard language or
changing the preferred language.

@available macOS 14.0 iOS 17.0
"elmoEmbedding"
@available macOS 14.0 iOS 17.0
"elmoEmbedding"
case dynamicEmbedding

    /**
     * A feature extractor that provides ELMo contextual word embeddings.
    /**
     * ELMo embeddings requires certain downloadable assets to be
present on device at training
     * time. Training will throw an error if the specified language is
unavailable at runtime. Asset downloads are
     * managed in the background automatically by the OS when a new
language is configured in device
     * settings, such as when adding a new keyboard language or
changing the preferred language.

@available macOS 14.0 iOS 17.0
case elmoEmbedding

    /**
     * A feature extractor that provides BERT contextual word embeddings.
    /**
     * The embeddings consider the context from left-to-right and right-to-
left simultaneously.
    /**
     * BERT embedding requires certain downloadable assets to be
present on device at training
     * time. Training will throw an error if the specified language is
unavailable at runtime. Asset downloads are
     * managed in the background automatically by the OS when a new

```

```
language is configured in device
    /// settings, such as when adding a new keyboard language or
    changing the preferred language.
    @available macOS 14.0 iOS 17.0
    case bertEmbedding

        /// A feature extractor that uses a custom embedding contained in a
CoreML model file.
        ///
        /// - Parameters:
        /// - URL: The path to a model file (ending in ` `.mlmodel` ,
` `.mlmodelc` , or ` `.dat` ).
        case customEmbedding URL

@available macOS 10.14 iOS 15.0
@available
extension MLTextClassifier

    /// The type of algorithm that a text classifier uses.
    ///
    /// Typically, `maxEnt` is the fastest training algorithm.
    /// If the text classifier's performance isn't good enough, consider the
`transferLearning` algorithm.
    public enum ModelAlgorithmType : Sendable

        /// A text classification algorithm that uses multinomial logistic
regression based on the frequencies of words, independent of context.
        /// - Parameters:
        /// - revision: The algorithm version. The only supported
version is 1. If `nil` defaults to the latest version.
        case maxEnt : Int

        /// A text classification algorithm that uses a statistical model of
transition probabilities between words.
        /// - Parameters:
        /// - revision: The algorithm version. The only supported
version is 1. If `nil` defaults to the latest version.
        case crf : Int

        /// A text classification algorithm that uses transfer learning by
leveraging a feature extractor to generate embeddings.
        /// - Parameters:
        /// - _: Feature extractor to be used by the transfer learning
algorithm.
        /// - revision: The algorithm version. The only supported
version is 1. If `nil` defaults to the latest version.
    @available macOS 10.15 iOS 15.0
    @available
```

```
    case
transferLearning MLTextClassifier FeatureExtractorType
    Int

@available macOS 10.14 iOS 15.0
@available
extension MLTextClassifier

    /// Classifies a string with a label.
    ///
    /// - Parameter text: The string to be classified.
    /// - Returns: The label for the given string.
    public func prediction           String throws      String

    /// Classifies an array of strings with labels.
    ///
    /// - Parameter texts: The array of strings to be classified.
    /// - Returns: An array of labels for the given strings.
    public func predictions          String throws      String
String

    /// Predicts multiple possible labels and their confidence scores for the
    specified string.
    ///
    /// - Parameter text: The string to classify.
    /// - Returns: A dictionary of label predictions and confidence scores.
    @available macOS 11.0 iOS 15.0
    @available
    public func predictionWithConfidence      String
throws      String Double

    /// Predicts multiple possible labels and their confidence scores for each
    string in the specified array.
    ///
    /// - Parameter texts: The array of strings to classify.
    /// - Returns: An array of dictionaries. Each dictionary corresponds to a
    string in the input array.
    /// A dictionary entry contains a label prediction with its associated
    confidence score.
    @available macOS 11.0 iOS 15.0
    @available
    public func predictionsWithConfidence     String
throws      String Double

@available macOS 10.14 iOS 15.0
@available
extension MLTextClassifier
```

```

/// Classifies a data column with labels.
///
/// - Parameter texts: The data column of strings to be classified.
/// - Returns: An array of labels for the given strings.
@available 10.14 13.0
"Use DataSource instead of MLDataTable."
@available 15.0 16.0
"Use DataFrame instead of MLDataTable."
@available
public func predictions MLDataColumn String
throws MLDataColumn String

/// Predicts multiple possible labels and their confidence scores for each
/// string in the specified data column.
///
/// - Parameter texts: The data column of strings to classify.
/// - Returns: A data column of dictionaries. Each dictionary corresponds
to
/// a string in the input data column. A dictionary entry contains a label
/// prediction with its associated confidence score.
@available 11.0 13.0
"Use DataFrame instead of MLDataTable."
@available 15.0 16.0
"Use DataFrame instead of MLDataTable."
@available
public func predictionsWithConfidence
MLDataColumn String throws MLDataColumn String
Double

@available macOS 10.14 iOS 15.0
@available
extension MLTextClassifier CustomStringConvertible
CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

/// A text representation of the text classifier.
public var description String get

/// A text representation of the text classifier that's suitable for output
/// during debugging.
public var debugDescription String get

/// A description of the text classifier in a playground.
public var playgroundDescription Any get

@available macOS 10.14 iOS 15.0

```

```

@available
extension MLTextClassifier

    /// Parameters that specify model training parameters and validation data.
public struct ModelParameters

        /// The parameter's algorithm setting.
public var algorithm
MLTextClassifier ModelAlgorithmType

        /// The parameter's language setting.
public var language NLLanguage

        /// The validation dataset.
///
/// The default value is
///
``MLTextClassifier/ModelParameters-swift.struct/ValidationData
-swift.enum/split(strategy:)``
        /// with the ``MLSplitStrategy/automatic`` split strategy``,
which
        /// automatically generates the validation dataset by partitioning up to
10% of the training dataset.
@available macOS 10.15 iOS 15.0
@available
public var validation
MLTextClassifier ModelParameters ValidationData

        /// The maximum number of training iterations.
@available macOS 11.0 iOS 15.0
@available
public var maxIterations Int

        /// Creates model parameters for a text classifier with the specified
validation data, algorithm, and language.
///
/// - Parameters:
///     - validation: The validation data to use during text classifier
training.
///     - algorithm: An algorithm type for the classifier.
///     - language: The language of the text to classify.
@available macOS 10.15 iOS 15.0
@available
public init
MLTextClassifier ModelParameters ValidationData

MLTextClassifier ModelAlgorithmType
NLLanguage nil

```

```
@available macOS 10.14 iOS 15.0
@available
extension MLTextClassifier

    /// Computes evaluation metrics.
    ///
    /// - Parameter labeledTexts: A data source of labeled texts to evaluate.
    /// - Returns: Classifier metrics.
    public func evaluation MLTextClassifier DataSource MLClassifierMetrics

    /// Computes evaluation metrics.
    ///
    /// - Parameter labeledTexts: A dictionary of labeled texts to evaluate.
    /// - Returns: Classifier metrics.
    public func evaluation String
String MLClassifierMetrics

    /// Computes evaluation metrics.
    ///
    /// - Parameters:
    ///     - dataFrame: A data frame containing labeled examples.
    ///     - textColumn: The name of the text column.
    ///     - labelColumn: The name of the label column.
    /// - Returns: The computed metrics or an error message.
@available macOS 13.0 iOS 16.0
@available
public func evaluation DataFrame
String String
MLClassifierMetrics
```

```
@available macOS 10.14 iOS 15.0
@available
extension MLTextClassifier

    /// Computes evaluation metrics.
    ///
    /// - Parameter labeledTexts: the data table for the data to be evaluated
    /// - Parameter textColumn: name of the text column in the provided trainingData
    /// - Parameter labelColumn: name of the label column in the provided trainingData
    /// - Returns: A `MLClassifierMetrics` struct containing the computed metrics or an error message.
```

**@available**

**10.14**

**13.0**

```
        "Use DataFrame instead of MLDataTable."
@available 15.0 16.0
        "Use DataFrame instead of MLDataTable."
@available
public func evaluation String String
MLClassifierMetrics

@available macOS 10.14 iOS 15.0
@available
extension MLTextClassifier DataSource

    /// Generates a data table by splitting the data source into strata.
    ///
    /// - Parameters:
    ///   - proportions: An array of proportions, each in the range ` [0.0, 1.0]`.
    ///   - seed: A seed number for the random-number generator. The default value is the current epoch time in milliseconds.
    ///   - labelColumn: The name of the column with the labels.
    ///   - textColumn: The name of the column with the text data.
    /// - Returns: A new data table.
    @available 10.14 14.0
    @available 15.0 17.0
    @available
    public func stratifiedSplit Double
Int String
String throws MLDataTable

@available macOS 10.15 iOS 15.0
@available
extension MLTextClassifier FeatureExtractorType
CustomStringConvertible CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of a feature extractor type.
    public var description String get

    /// A text representation of the feature extractor type that's suitable for output during debugging.
    public var debugDescription String get

    /// A description of the feature extractor type shown in a playground.
    public var playgroundDescription Any get

@available macOS 10.14 iOS 15.0
```

```
@available
extension MLTextClassifier ModelAlgorithmType
CustomStringConvertible CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of the algorithm type.
public var description String get

    /// A text representation of the algorithm type that's suitable for output
/// during debugging.
public var debugDescription String get

    /// A description of the algorithm type shown in a playground.
public var playgroundDescription Any get

@available macOS 10.14 iOS 15.0
@available
extension MLTextClassifier ModelParameters

    /// The validation data that a text classifier uses.
@available macOS 10.15 iOS 15.0
@available
public enum ValidationData

    /// Generates the validation data by splitting the training dataset.
///
/// By default, model parameters use this approach to specify the
validation data.
case split MLSplitStrategy

    /// Sets the validation data from the provided data table.
@available 10.15 14.0
@available 15.0 17.0
@available
case table MLDataTable String
String

    /// Set validation data from the MLDataTable provided.
@available macOS 13.0 iOS 16.0
@available
case dataFrame DataFrame String
String

    /// Sets the validation data from the provided data source.
case dataSource MLTextClassifier DataSource

    /// Sets the validation data from the provided dictionary.
case dictionary String String
```

```
    /// Doesn't set validation data.
    case none

@available macOS 10.14 iOS 15.0
@available
extension MLTextClassifier ModelParameters
CustomStringConvertible CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of the model parameters.
    public var description String get

    /// A text representation of the model parameters that's suitable for output
    // during debugging.
    public var debugDescription String get

    /// A description of the model parameters in a playground.
    public var playgroundDescription Any get

@available macOS 10.14 iOS 15.0
@available
extension MLTextClassifier ModelParameters

    /// The validation data.
    @available 10.14 10.15
    "Use the validation property instead."
    @available 15.0 16.0
    "Use the validation property instead."
    @available
    public var validationData MLDataTable

    /// The name of the text column in the validation data table.
    @available 10.14 10.15
    "Use the validation property instead."
    @available 15.0 16.0
    "Use the validation property instead."
    @available
    public var textColumnValidationData String

    /// Creates parameters for a text classifier with validation data in a
    /// set of labeled directories.
    ///
    /// - Parameters:
    ///   - validationData: A data source of the labeled directories the text
    ///     classifier uses for validation data during training.
```

```
///  
/// – algorithm: An algorithm type for the text classifier.  
///  
/// – language: The language of the text to classify.  
@available 10.14 10.15  
    "Use the validation property instead."  
@available 15.0 16.0  
    "Use the validation property instead."  
@available  
public init MLTextClassifier DataSource  
    MLTextClassifier ModelAlgorithmType  
        1 NLLanguage nil  
  
/// Creates parameters for a text classifier with validation data in a  
/// dictionary.  
///  
/// – Parameters:  
/// – validationData: A dictionary the text classifier uses for  
/// validation data during training.  
///  
/// – algorithm: An algorithm type for the classifier.  
///  
/// – language: The language of the text to classify.  
@available 10.14 10.15  
    "Use the validation property instead."  
@available 15.0 16.0  
    "Use the validation property instead."  
@available  
public init String String  
    MLTextClassifier ModelAlgorithmType  
        1 NLLanguage nil  
  
/// Creates parameters for a text classifier with validation data in a  
/// data table.  
///  
/// – Parameters:  
/// – validationData: A data table the text classifier uses for  
/// validation data during training.  
///  
/// – algorithm: An algorithm type for the classifier.  
///  
/// – language: The language of the text to classify.  
///  
/// – textColumnValidationData: The name of the text column in the  
/// validation data table.  
///  
/// – labelColumnValidationData: The name of the label column in the  
/// validation data table.  
@available 10.14 10.15
```

```
        "Use the validation property instead."
@available 15.0 16.0
        "Use the validation property instead."
@available
public init MLDataTable nil
MLTextClassifier ModelAlgorithmType 1
NLLanguage nil String
nil String nil

/// The name of the label column in the validation data table.
@available 10.14 10.15
        "Use the validation property instead."
@available 15.0 16.0
        "Use the validation property instead."
@available
public var labelColumnValidationData String

/// The current state of a model's asynchronous training session.
@available macOS 11.0 iOS 15.0 tvOS 16.0
final public class MLTrainingSession Task

/// The parameters you used to create the training session.
final public let parameters MLTrainingSessionParameters

/// The time when you created this training session.
final public var date Date get

/// The training session's current state.
final public var phase MLPhase get

/// The iteration number of a training session's phase.
final public var iteration Int get

/// An array of checkpoints the training session has created so far.
final public var checkpoints MLCheckpoint get

/// Removes the checkpoints that satisfy your closure from the training
/// session.
///
/// - Parameters:
///   - predicate: A closure that returns a Boolean indicating whether to
///     remove a checkpoint.
final public func removeCheckpoints _  
MLCheckpoint Bool throws

/// Uses the features another session has already extracted from its
/// dataset.
///
```

```
    /// You can only use this method for a new training session that doesn't
    /// already have its own checkpoints.
    ///
    /// - Note: This method throws an error if the training session has one or
    /// more checkpoints.
    ///
    /// - Parameters:
    ///   - session: Another training session that's already completed its
feature
    /// extraction phase.
    final public func reuseExtractedFeatures
MLTrainingSession Task throws
```

```
@available macOS 14.0 iOS 17.0 tvOS 17.0
extension MLTrainingSession @unchecked Sendable
```

```
    /// The configuration settings for a training session.
    @available macOS 11.0 iOS 15.0 tvOS 16.0
    public struct MLTrainingSessionParameters Sendable

    /// The location in the file system where the session stores its progress.
    public let sessionDirectory URL

    /// The number of iterations the session completes before it reports its
    /// progress.
    public var reportInterval Int

    /// The number of iterations the session completes before it saves a
    /// checkpoint.
    public var checkpointInterval Int

    /// The maximum number of iterations for the training session.
    ///
    /// Each iteration represents a full pass over the training data, also known as
an epoch. Training may stop with
    /// fewer iterations if training converges. This limit also affects resumed
training sessions. To extend training
    /// beyond the original limit, increase the limit before resuming.
    public var iterations Int

    /// Creates a set of parameters for a training session.
    ///
    /// - Parameters:
    ///   - sessionDirectory: The location in the file system where the
session stores its progress.
    ///   - reportInterval: The number of iterations the session
completes before it reports its progress.
    ///   - checkpointInterval: The number of iterations the session
```

completes before it saves a checkpoint.

/// - iterations: The total number of iterations for the session.

|                    |            |            |
|--------------------|------------|------------|
| <b>public init</b> | <b>URL</b> | <b>nil</b> |
| Int 5              | Int 10     | Int 1000   |

/// A column of untyped values in a data table.

///

/// A column is a homogenous collection of data values, similar to an  
/// <doc://com.apple.documentation/documentation/swift/array>. Columns are the  
/// main components of an ``MLDataTable`` and are designed to efficiently  
scale

/// with large data sets.

///

/// Typically you use ``MLDataColumn``, the typed equivalent to  
/// ``MLUntypedColumn``, for its type-specific functionality.

///

/// Untyped columns are especially useful when:

///

/// - You're initializing a data table with columns by using  
``MLDataTable/init(namedColumns:)``.

/// - You're using columns of a non-Boolean type to filter a data table with  
``MLDataTable/subscript(\_:)10r4l``.

/// - You don't need to work directly with the underlying type.

///

/// Each element of an untyped column is an ``MLDataValue``, and has an  
/// *\_underlying type\_* that conforms to ``MLDataValueConvertible``. The  
/// underlying type is hidden from the Swift compiler and is what makes an  
/// ``MLUntypedColumn`` untyped. Using an untyped column allows you to  
quickly

/// write type-agnostic code with Create ML.

///

/// ``swift

/// let column = MLUntypedColumn([2, 3, 5, 7, 11])  
/// let columnOver2 = column / 2 print(columnOver2)

/// /\* Prints...

/// Value Type: Double

/// Values: [1.0, 1.5, 2.5, 3.5, 5.5]

/// \*/

/// ``

///

/// However, by avoiding type safety at compile time, you expose your code to  
/// errors at runtime. When an error occurs during an operation, Create ML marks  
/// the product of that operation *\_invalid\_* by setting  
/// ``MLUntypedColumn/isValid`` to `false` and by setting  
/// ``MLUntypedColumn/error`` with a value. For example, using a slash  
(` `/)`)  
/// operator to divide a column of integers with a string produces an invalid  
/// column.

///

```

/// ````swift
/// let column = MLUntypedColumn([2, 3, 5, 7, 11])
/// let invalidColumn = column / "foo"
/// print(invalidColumn.isValid) // Prints "false"
///
///
/// - Important: A mismatch between the underlying types of two columns, or
///   between the underlying type of a column and the type of a value, will result
///   in an invalid column.
///
/// Once a column becomes invalid, you can't use it for any subsequent operation
/// because it will only produce further invalid columns or invalid tables.
///
/// Each comparison operator of ``MLUntypedColumn`` returns a column of
/// Booleans. However, ``MLUntypedColumn`` uses integers as its underlying
/// type
/// for columns of Booleans, because ``MLDataValue`` does not have a case
/// for
/// <doc://com.apple.documentation/documentation/swift/bool>.
///
/// For example, create an untyped column of Booleans using the less-than
/// comparison operator(``MLUntypedColumn/<(_:_)-7zms0``).
///
/// ````swift
/// let column = MLUntypedColumn([2, 3, 5, 7, 11])
/// let lessThan5 = column < 5
///
///
/// Then print the column to see that its underlying `ValueType` is `Int`, and
/// each Boolean value of `true` or `false` is represented in the column by an
/// integer value of `1` or `0`, respectively.
///
/// ````swift
/// print(lessThan5)
/// /* Prints...
///  ValueType: Int
///  Values:      [1, 1, 0, 0, 0]
/// */
///
///
/// Use these untyped columns of Booleans just as you would with a typed column
/// of Booleans
///
(``ML DataColumn````<`<doc://com.apple.documentation/documentation/swift/
bool>`>`)
/// to:
///
/// - Filter another untyped column with
/// ``MLUntypedColumn/subscript(_:_)-9hr32``
/// - Logically combine with another untyped column of Booleans with the

```

```
/// ``MLUntypedColumn/&&(_:_:)`` and ``MLUntypedColumn//|(_:_:)`` operators
/// - Mask rows of an ``MLDataTable`` with its
/// ``MLDataTable/subscript(_:)``-3opgl``
@available macOS 10.14 iOS 15.0 tvOS 16.0
public struct MLUntypedColumn

    /// The number of elements in the column.
public var count Int get

@available macOS 11.0 iOS 15.0 tvOS 16.0
public var isEmpty Bool get

    /// The underlying type of the column.
    ///
    /// Use this to determine the underlying type of the column. Then use a
    /// corresponding property or method to create an ``ML DataColumn`` of
    the
    /// untyped column. For example, if ``MLUntypedColumn/type`` is equal
    to
    /// ``MLDataValue/ValueType/double``, then use the
    ``MLUntypedColumn``
    /// ``MLUntypedColumn/doubles`` property.
public var type MLDataValue ValueType get

    /// The underlying error present when the column is invalid.
public var error any Error get

    /// A Boolean value that indicates whether the column is valid.
    ///
    /// Check ``MLUntypedColumn/isValid`` after you create or mutate an
    untyped
    /// column to ensure it's valid. If the value is
    /// <doc://com.apple.documentation/documentation/swift/false>, the untyped
    /// column encountered an error and you can't use it for subsequent
    /// operations. For example, comparing two columns of different sizes
    /// creates an invalid column.
public var isValid Bool get

    /// Clones the column to a data column of the given type.
    ///
    /// Use this method to create a typed copy of the column. For example, to
    /// create a data column of integers from an untyped column of integers, use
    /// ``MLUntypedColumn/column(type:)`` with
    /// <doc://com.apple.documentation/documentation/swift/int>`.self` as the
    /// argument for the `type` parameter.
    ///
    /// - Parameters:
    ///   - type: A metatype used to create a new data column of that type.
    /// 
```

```

the    /// - Returns: A new data column if the underlying type of the column is
      /// same as `type`; otherwise `nil`.
      public func column<T>() -> MLDataColumn<T>
      where T : MLDataValueConvertible

      /// A cloned data column of integers.
      ///
      /// This property is functionally equivalent to passing
      /// <doc://com.apple.documentation/documentation/swift/int>`.self` to
      /// ``MLUntypedColumn/column(type:)``. Typically you ensure
      /// ``MLUntypedColumn/type`` is equal to
      ``MLDataValue/ValueType/int``
      /// before getting this property.
      ///
      /// - Returns: A new data column if the underlying type of the column is
      /// <doc://com.apple.documentation/documentation/swift/int>; otherwise
      `nil`.

      public var ints: MLDataColumn<Int> { get }

      /// A cloned data column of doubles.
      ///
      /// This property is functionally equivalent to passing
      /// <doc://com.apple.documentation/documentation/swift/double>`.self` to
      to
      /// ``MLUntypedColumn/column(type:)``. Typically you ensure
      /// ``MLUntypedColumn/type`` is equal to
      ``MLDataValue/ValueType/double``
      /// before getting this property.
      ///
      /// - Returns: A new data column if the underlying type of the column is
      /// <doc://com.apple.documentation/documentation/swift/double>; otherwise
      `nil`.

      public var doubles: MLDataColumn<Double> { get }

      /// A cloned data column of strings.
      ///
      /// This property is functionally equivalent to passing
      /// <doc://com.apple.documentation/documentation/swift/string>`.self` to
      /// ``MLUntypedColumn/column(type:)``. Typically you ensure
      /// ``MLUntypedColumn/type`` is equal to
      ``MLDataValue/ValueType/string``
      /// before getting this property.
      ///
      /// - Returns: A new data column if the underlying type of the column is
      /// <doc://com.apple.documentation/documentation/swift/string>; otherwise
      `nil`.

      public var strings: MLDataColumn<String> { get }

      /// A cloned data column of machine learning sequences.

```

```

    /**
     * This property is functionally equivalent to passing
     * ``MLDataValue/SequenceType``'`self` to
     * ``MLUntypedColumn/column(type:)``. Typically you ensure
     * ``MLUntypedColumn/type`` is equal to
     * ``MLDataValue/ValueType/sequence`` before getting this property.
     */
    /**
     * - Returns: A new data column if the underlying type of the column is
     *   ``MLDataValue/SequenceType``; otherwise `nil`.
     */
    public var sequences: ML DataColumn MLDataValue SequenceType get

    /**
     * A cloned data column of machine learning dictionaries.
     */
    /**
     * This property is functionally equivalent to passing
     * ``MLDataValue/DictionaryType``'`self` to
     * ``MLUntypedColumn/column(type:)``. Typically you ensure
     * ``MLUntypedColumn/type`` is equal to
     * ``MLDataValue/ValueType/dictionary`` before getting this
     * property.
     */
    /**
     * - Returns: A new data column if the underlying type of the column is
     *   ``MLDataValue/DictionaryType``; otherwise `nil`.
     */
    public var dictionaries: ML DataColumn MLDataValue DictionaryType get

    /**
     * A cloned data column of machine learning multi-arrays.
     */
    /**
     * This property is functionally equivalent to passing
     * ``MLDataValue/MultiArrayType``'`self` to
     * ``MLUntypedColumn/column(type:)``. Typically you ensure
     * ``MLUntypedColumn/type`` is equal to
     * ``MLDataValue/ValueType/multiArray`` before getting this
     * property.
     */
    /**
     * - Returns: A new data column if the underlying type of the column is
     *   ``MLDataValue/MultiArrayType``; otherwise `nil`.
     */
    public var multiArrays: ML DataColumn MLDataValue MultiArrayType get

    /**
     * Creates a new column from a given sequence of machine learning data
     * values.
     */
    /**
     * Use this initializer to create a column from a sequence of data values.
     */
    /**
     * swift
     * let dataValue0f2 = MLDataValue.int(2)
     * let dataValue0f3 = MLDataValue.int(3)
     * let dataValue0f5 = MLDataValue.int(5)
     */

```

```

    ///> let dataValue0f7 = MLDataValue.int(7)
    ///> let dataValue0f11 = MLDataValue.int(11)
    ///
    ///> let sequence = [dataValue0f2,
    ///>                  dataValue0f3,
    ///>                  dataValue0f5,
    ///>                  dataValue0f7,
    ///>                  dataValue0f11
    ///> ]
    ///
    ///> let sequenceColumn = MLUntypedColumn(sequence)
    ///> print(sequenceColumn)
    ///> /*
    ///> Prints...
    ///> Value: Int
    ///> Values: [2, 3, 5, 7, 11]
    ///> */
    ///> ```

    ///> - Parameters:
    ///>   - source: A sequence of data value elements for the new
column.
public init S _           S where S Sequence S Element
MLDataValue

    ///> Creates a new column from a given sequence of elements that can be
    ///> converted to machine learning data values.
    ///
    ///> Use this initializer to create a column from a sequence of any type that
    ///> conforms to ``MLDataValueConvertible``.
    ///
    ///> ````swift
    ///> let sequenceColumn = MLUntypedColumn([2, 3, 5, 7, 11])
    ///> print(sequenceColumn)
    ///> /* Prints...
    ///> Value: Int
    ///> Values: [2, 3, 5, 7, 11]
    ///> */
    ///> ````

    ///> - Parameters:
    ///>   - source: A sequence of convertible elements for the new column.
public init S _           S where S Sequence
S Element MLDataValueConvertible

    ///> Creates a new column with a repeating value.
    ///
    ///> Use this initializer to create a column of repeating elements of a data
    ///> value.

```

```

////
//// ````swift
/// let dataValueOf5 = MLDataValue.int(5)
/// let three5s = MLUntypedColumn(repeating: dataValueOf5,
count: 3)
/// print(three5s)
/// /*
/// Prints...
/// ValueType: Int
/// Values:      [5, 5, 5]
/// */
/// ``
/// - Parameters:
///   - repeatedValue: An initial value for every element in the new
column.
///   - count: The number of elements in the new column.
public init MLDataValue
Int

/// Creates a new column with a repeating value.
///
/// Use this initializer to create a column of repeating elements with any
/// type that conforms to ``MLDataValueConvertible``, including
integers,
/// doubles, strings, arrays, and dictionaries.
///
/// ````swift
/// let three5s = MLUntypedColumn(repeating: 5, count: 3)
/// print(three5s)
/// /* Prints...
/// ValueType: Int
/// Values:      [5, 5, 5]
/// */
/// ``
/// - Parameters:
///   - repeatedValue: An initial value for every element in the new
column.
///   - count: The number of elements in the new column.
public init T T Int
where T MLDataValueConvertible

/// Creates a new column of integers from a given range.
///
/// Use this initializer to create a column of incrementing values from a
/// range.
///
/// ````swift
/// let rangeColumn = MLUntypedColumn(3..<7)

```

```
////
/// print(rangeColumn)
/// /* Prints...
///   ValueType: Int
///   Values:      [3, 4, 5, 6]
///   */
/// ``
///
/// - Parameters:
///   - range: A range of integer elements for the new column.
public init _ Range Int

/// Creates a new column of integers from a given closed range.
///
/// Use this initializer to create a column of incrementing values from a
/// closed range.
///
/// ````swift
/// let closedRangeColumn = MLUntypedColumn(3...7)
/// print(closedRangeColumn)
/// /* Prints...
///   ValueType: Int
///   Values:      [3, 4, 5, 6, 7]
///   */
/// ``
///
/// - Parameters:
///   - range: A closed range of integer elements for the new column.
public init _ ClosedRange Int

/// Creates an empty, invalid column used to remove an existing column from
/// a data table.
///
/// Use this initializer to create an empty column that, when assigned to
/// the name of an existing column within a data table, will remove that
/// column from the table.
///
/// For example, to remove a column from a data table, first start with a
/// variable data table.
///
/// ````swift
/// let data: [String: MLDataValueConvertible] = [
///   "Day": ["Monday", "Tuesday", "Wednesday",
/// "Thursday", "Friday"],
///   "Temperature": [91.3, 85.8, 79.5, 83.4, 72.2]
/// ]
///
/// var table = try MLDataTable(dictionary: data)
/// 
```

```

/// print(table)
/// /* Prints...
/// Columns:
/// Day    string
/// Temperature   float
/// Rows: 5
/// Data:
/// +-----+-----+
/// | Day        | Temperature |
/// +-----+-----+
/// | Monday      | 91.3       |
/// | Tuesday     | 85.8       |
/// | Wednesday   | 79.5       |
/// | Thursday    | 83.4       |
/// | Friday      | 72.2       |
/// +-----+-----+
/// [5 rows x 2 columns]
/// */
/// ```` swift
/// let emptyColumn = MLUntypedColumn()
/// ````

/// Finally, use ``MLUntypedColumn/subscript(_:_)-9hr32`` to
remove the
/// existing column from the data table by setting the empty column to the
/// name of that existing column.
/// ```` swift
/// // Remove the "Temperature" column from the data table
/// table["Temperature"] = emptyColumn
/// ````

/// print(table)
/// /* Prints...
/// Columns:
/// Day    string
/// Rows: 5
/// Data:
/// +-----+
/// | Day        |
/// +-----+
/// | Monday      |
/// | Tuesday     |
/// | Wednesday   |
/// | Thursday    |
/// | Friday      |

```

```
/// +-----+
/// [5 rows x 1 columns]
/// */
///
///
/// - Returns: An empty, invalid column.
public init

/// Appends the elements of the given column to the end of this column.
///
/// - Parameters:
///   - newColumn: Another column to append to the column.
///
/// - Note: The type of `newColumn` must be the same type or
convertible to the same
///   type as this column. See ``MLDataValueConvertible``.
public mutating func append
MLUntypedColumn
```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLUntypedColumn
```

```
/// Creates a column of Booleans by testing whether each element in the
/// given column is equal to the given value.
///
/// - Parameters:
///   - a: A column.
///   - b: A value.
///
/// - Returns: A new column of Booleans if the column and value have
the
///   same underlying type; otherwise an invalid column.
public static func MLUntypedColumn any
MLDataValueConvertible MLUntypedColumn

/// Creates a column of Booleans by testing whether each element in the
/// given column is not equal to the given value.
///
/// - Parameters:
///   - a: A column.
///   - b: A value.
///
/// - Returns: A new column of Booleans if the column and value have
the
///   same underlying type; otherwise an invalid column.
public static func MLUntypedColumn any
MLDataValueConvertible MLUntypedColumn

/// Creates a column of Booleans by testing whether each element in the
```

```
    /// given column is greater than the given value.  
    ///  
    /// - Parameters:  
    ///   - a: A column.  
    ///   - b: A value.  
    ///  
    /// - Returns: A new column of Booleans if the column and value have  
the same underlying type; otherwise an invalid column.  
    public static func      MLUntypedColumn      any  
MLDataValueConvertible      MLUntypedColumn  
  
    /// Creates a column of Booleans by testing whether each element in the  
    /// given column is less than the given value.  
    ///  
    /// - Parameters:  
    ///   - a: A column.  
    ///   - b: A value.  
    ///  
    /// - Returns: A new column of Booleans if the column and value have  
the same underlying type; otherwise an invalid column.  
    public static func      MLUntypedColumn      any  
MLDataValueConvertible      MLUntypedColumn  
  
    /// Creates a column of Booleans by testing whether each element in the  
    /// given column is greater than or equal to the given value.  
    ///  
    /// - Parameters:  
    ///   - a: A column.  
    ///   - b: A value.  
    ///  
    /// - Returns: A new column of Booleans if the column and value have  
the same underlying type; otherwise an invalid column.  
    public static func      MLUntypedColumn      any  
MLDataValueConvertible      MLUntypedColumn  
  
    /// Creates a column of Booleans by testing whether each element in the  
    /// given column is less than or equal to the given value.  
    ///  
    /// - Parameters:  
    ///   - a: A column.  
    ///   - b: A value.  
    ///  
    /// - Returns: A new column of Booleans if the column and value have  
the same underlying type; otherwise an invalid column.  
    public static func      MLUntypedColumn      any  
MLDataValueConvertible      MLUntypedColumn
```

```
    /// Creates a column by adding each element of the given column to the
given
    /// value.
    ///
    /// - Parameters:
    ///   - a: An column.
    ///   - b: A value.
    ///
    /// - Returns: A new column if the column and value have the same
underlying
    /// type; otherwise an invalid column.
public static func      MLUntypedColumn      any
MLDataValueConvertible  MLUntypedColumn

    /// Creates a column by subtracting the given value from each element of the
    /// given column.
    ///
    /// - Parameters:
    ///   - a: An column.
    ///   - b: A value.
    ///
    /// - Returns: A new column if the column and value have the same
underlying
    /// type; otherwise an invalid column.
public static func      MLUntypedColumn      any
MLDataValueConvertible  MLUntypedColumn

    /// Creates a column by multiplying each element of the given column by the
    /// given value.
    ///
    /// - Parameters:
    ///   - a: An column.
    ///   - b: A value.
    ///
    /// - Returns: A new column if the column and value have the same
underlying
    /// type; otherwise an invalid column.
public static func      MLUntypedColumn      any
MLDataValueConvertible  MLUntypedColumn

    /// Creates a column by dividing each element of the given column by the
    /// given value.
    ///
    /// - Parameters:
    ///   - a: An column.
    ///   - b: A value.
    ///
    /// - Returns: A new column if the column and value have the same
underlying
```

```
    /// type; otherwise an invalid column.
    public static func      MLUntypedColumn      any
MLDataValueConvertible      MLUntypedColumn

    /// Creates a column of Booleans by testing whether the given value is equal
    /// to each element in the given column.
    ///
    /// - Parameters:
    ///   - a: A value.
    ///   - b: A column.
    ///
    /// - Returns: A new column of Booleans if the column and value have
the
    /// same underlying type; otherwise an invalid column.
    public static func      any MLDataValueConvertible
MLUntypedColumn      MLUntypedColumn

    /// Creates a column of Booleans by testing whether the given value is not
    /// equal to each element in the given column.
    ///
    /// - Parameters:
    ///   - a: A value.
    ///   - b: A column.
    ///
    /// - Returns: A new column of Booleans if the column and value have
the
    /// same underlying type; otherwise an invalid column.
    public static func      any MLDataValueConvertible
MLUntypedColumn      MLUntypedColumn

    /// Creates a column of Booleans by testing whether the given value is
    /// greater than each element in the given column.
    ///
    /// - Parameters:
    ///   - a: A value.
    ///   - b: A column.
    ///
    /// - Returns: A new column of Booleans if the column and value have
the
    /// same underlying type; otherwise an invalid column.
    public static func      any MLDataValueConvertible
MLUntypedColumn      MLUntypedColumn

    /// Creates a column of Booleans by testing whether the given value is less
    /// than each element in the given column.
    ///
    /// - Parameters:
    ///   - a: A value.
    ///   - b: A column.
    ///
```

```
the    /// - Returns: A new column of Booleans if the column and value have
       /// same underlying type; otherwise an invalid column.
       public static func      any MLDataValueConvertible
       MLUntypedColumn      MLUntypedColumn

       /// Creates a column of Booleans by testing whether the given value is
       /// greater than or equal to each element in the given column.
       ///
       /// - Parameters:
       ///   - a: A value.
       ///   - b: A column.
       ///
       the    /// - Returns: A new column of Booleans if the column and value have
              /// same underlying type; otherwise an invalid column.
              public static func      any MLDataValueConvertible
              MLUntypedColumn      MLUntypedColumn

              /// Creates a column of Booleans by testing whether the given value is less
              /// than or equal to each element in the given column.
              ///
              /// - Parameters:
              ///   - a: A value.
              ///   - b: A column.
              ///
              the    /// - Returns: A new column of Booleans if the column and value have
                     /// same underlying type; otherwise an invalid column.
                     public static func      any MLDataValueConvertible
                     MLUntypedColumn      MLUntypedColumn

                     /// Creates a column by adding the given value to each element of the given
                     /// column.
                     ///
                     /// - Parameters:
                     ///   - a: An value.
                     ///   - b: A column.
                     ///
                     the    /// - Returns: A new column if the column and value have the same
                            /// underlying
                            ///   type; otherwise an invalid column.
                            public static func      any MLDataValueConvertible
                            MLUntypedColumn      MLUntypedColumn

                            /// Creates a column by subtracting each element of the given column from
                            /// the given value.
                            ///
                            /// - Parameters:
                            ///   - a: An value.
```

```
    /// - b: A column.  
    ///  
    /// - Returns: A new column if the column and value have the same  
underlying  
    /// type; otherwise an invalid column.  
    public static func      any MLDataValueConvertible  
    MLUntypedColumn      MLUntypedColumn  
  
    /// Creates a column by multiplying the given value by each element of the  
    /// given column.  
    ///  
    /// - Parameters:  
    ///   - a: An value.  
    ///   - b: A column.  
    ///  
    /// - Returns: A new column if the column and value have the same  
underlying  
    /// type; otherwise an invalid column.  
    public static func      any MLDataValueConvertible  
    MLUntypedColumn      MLUntypedColumn  
  
    /// Creates a column by dividing the given value by each element of the  
    /// given column.  
    ///  
    /// - Parameters:  
    ///   - a: An value.  
    ///   - b: A column.  
    ///  
    /// - Returns: A new column if the column and value have the same  
underlying  
    /// type; otherwise an invalid column.  
    public static func      any MLDataValueConvertible  
    MLUntypedColumn      MLUntypedColumn  
  
    /// Creates a column of Booleans by testing whether each element in the  
    /// first column is equal to the corresponding element in the second column.  
    ///  
    /// - Parameters:  
    ///   - a: A column.  
    ///   - b: A column.  
    ///  
    /// - Returns: A new column of Booleans if the columns have the same  
size  
    /// and underlying type; otherwise an invalid column.  
    public static func      MLUntypedColumn  
    MLUntypedColumn      MLUntypedColumn  
  
    /// Creates a column of Booleans by testing whether each element in the  
    /// first column is not equal to the corresponding element in the second  
    /// column.
```

```
///  
/// - Parameters:  
///   - a: A column.  
///   - b: A column.  
///  
/// - Returns: A new column of Booleans if the columns have the same  
size  
/// and underlying type; otherwise an invalid column.  
public static func      MLUntypedColumn  
MLUntypedColumn      MLUntypedColumn  
  
/// Creates a column of Booleans by testing whether each element in the  
/// first column is greater than the corresponding element in the second  
/// column.  
///  
/// - Parameters:  
///   - a: A column.  
///   - b: A column.  
///  
/// - Returns: A new column of Booleans if the columns have the same  
size  
/// and underlying type; otherwise an invalid column.  
public static func      MLUntypedColumn  
MLUntypedColumn      MLUntypedColumn  
  
/// Creates a column of Booleans by testing whether each element in the  
/// first column is less than the corresponding element in the second  
/// column.  
///  
/// - Parameters:  
///   - a: A column.  
///   - b: A column.  
///  
/// - Returns: A new column of Booleans if the columns have the same  
size  
/// and underlying type; otherwise an invalid column.  
public static func      MLUntypedColumn  
MLUntypedColumn      MLUntypedColumn  
  
/// Creates a column of Booleans by testing whether each element in the  
/// first column is greater than or equal to the corresponding element in  
/// the second column.  
///  
/// - Parameters:  
///   - a: A column.  
///   - b: A column.  
///  
/// - Returns: A new column of Booleans if the columns have the same  
size  
/// and underlying type; otherwise an invalid column.
```

```
public static func          MLUntypedColumn
MLUntypedColumn      MLUntypedColumn

    Creates a column of Booleans by testing whether each element in the
first column is less than or equal to the corresponding element in the
second column.

- Parameters:
    - a: A column.
    - b: A column.

- Returns: A new column of Booleans if the columns have the same
size
    and underlying type; otherwise an invalid column.

public static func          MLUntypedColumn
MLUntypedColumn      MLUntypedColumn

    Creates a column of Booleans by performing a logical AND operation on
each row of two columns of Booleans.

- Parameters:
    - a: A column.
    - b: A column.

- Returns: A new column of Booleans if the column and value have
the
    same Boolean/integer underlying type; otherwise an invalid column.

public static func          MLUntypedColumn
MLUntypedColumn      MLUntypedColumn

    Creates a column of Booleans by performing a logical OR operation on
each row of two columns of Booleans.

- Parameters:
    - a: A column.
    - b: A column.

- Returns: A new column of Booleans if the column and value have
the
    same Boolean/integer underlying type; otherwise an invalid column.

public static func          MLUntypedColumn
MLUntypedColumn      MLUntypedColumn

    Creates a column by adding each element in the first column to the
corresponding element in the second column.

- Parameters:
    - a: A column.
    - b: A column.
```

```
    /// - Returns: A new column if the columns have the same size and
underlying
    /// type; otherwise an invalid column.
public static func      MLUntypedColumn
MLUntypedColumn      MLUntypedColumn

    /// Creates a column by subtracting each element in the second column from
    /// the corresponding element in the first column.
    ///
    /// - Parameters:
    ///   - a: A column.
    ///   - b: A column.
    ///
    /// - Returns: A new column if the columns have the same size and
underlying
    /// type; otherwise an invalid column.
public static func      MLUntypedColumn
MLUntypedColumn      MLUntypedColumn

    /// Creates a column by multiplying each element in the first column by the
    /// corresponding element in the second column.
    ///
    /// - Parameters:
    ///   - a: A column.
    ///   - b: A column.
    ///
    /// - Returns: A new column if the columns have the same size and
underlying
    /// type; otherwise an invalid column.
public static func      MLUntypedColumn
MLUntypedColumn      MLUntypedColumn

    /// Creates a column by dividing each element in the first column by the
    /// corresponding element in the second column.
    ///
    /// - Parameters:
    ///   - a: A column.
    ///   - b: A column.
    ///
    /// - Returns: A new column if the columns have the same size and
underlying
    /// type; otherwise an invalid column.
public static func      MLUntypedColumn
MLUntypedColumn      MLUntypedColumn
```

**@available macOS 10.14 iOS 15.0 tvOS 16.0**  
**extension** MLUntypedColumn

```
    /// Creates a subset of the column by masking its elements with another
```

```

    //// untyped column.
    ///
    //// Use this untyped column-based subscript to create a new column by
    //// masking a subset of the elements. The derived column will not include
    //// elements where `mask` contains a default value for its underlying type,
    //// such as:
    ///
    //// - `0` in untyped `Int` columns
    //// - `0.0` in untyped `Double` columns
    //// - An empty string in untyped `String` columns
    ///
    //// The derived column includes elements where the masking column has
any
    //// other (nondefault) value.
    ///
    //// See ``MLDataTable/subscript(_:)--10r4l`` from
``MLDataTable`` for an
    /// example.
    ///
    //// - Parameters:
    ////   - mask: An untyped column indicating whether elements should be
removed
    ////     (a default value) or included (any nondefault value) in the derived
    ////     column.
    ///
    //// - Returns: A new column.
public subscript      MLUntypedColumn      MLUntypedColumn
get

    //// Creates a subset of the column by masking its elements with a data
    //// column of Booleans.
    ///
    //// - Parameters:
    ////   - mask: A Boolean column indicating whether elements should be
kept
    ////     (`true`) or removed (`false`) in the derived column.
    ///
    //// - Returns: A new column.
public subscript      ML DataColumn Bool
MLUntypedColumn      get

    //// Creates a new column of typed values, potentially with missing values,
    //// by applying the given thread-safe transform to every non-missing element
    //// of this untyped column.
    ///
    //// - Parameters:
    ////   - lazyTransform: A thread-safe element transformation function.
The
    ////     implementation of the transform you provide should accept an
`Element`
```

/// of the column and return a transformed value of a type that  
    /// conforms to  
    /// `MLDataValueConvertible`. If the transform returns `nil`  
    /// for a given  
    /// element, the corresponding element in the new column will have a  
    /// missing  
    /// value.

    /// - **Returns:** A new `MLDataColumn` typed to the return type of  
    `lazyTransform`.

**public func** `map T` @escaping  
`MLDataValue T` MLDataColumn T where T  
`MLDataValueConvertible`

    /// Creates a new column of typed values by applying the given thread-safe  
    /// transform to every non-missing element of this untyped column.

    ///

    /// - **Parameters:**

    /// - `lazyTransform`: A thread-safe element transformation function.

The

    /// implementation of the transform you provide should accept an  
    `Element`  
    /// of the column and return a transformed value of a type that  
    /// conforms to  
    /// `MLDataValueConvertible`.

    ///

    /// - **Returns:** A new `MLDataColumn` typed to the return type of  
    `lazyTransform`.

**public func** `map T` @escaping  
`MLDataValue T` MLDataColumn T where T  
`MLDataValueConvertible`

    /// Creates a new column of typed values by applying the given thread-safe  
    /// transform to every element of this untyped column, including missing  
    /// elements.

    ///

    /// - **Parameters:**

    /// - `lazyTransform`: A thread-safe element transformation function.

The

    /// implementation of the transform you provide should accept an  
    element of  
    /// the column and return a transformed value of a type that conforms  
    to  
    /// to  
    /// `MLDataValueConvertible`. If the transform returns `nil`  
    for a given  
    /// element, the corresponding element in the new column will have a  
    /// missing  
    /// value.

    ///

    /// - **Returns:** A new `MLDataColumn` column.

**public func** `mapMissing T` @escaping

```
MLDataValue      T      ML DataColumn T  where T
MLDataValueConvertible

    /// Creates a new column of typed values by converting this untyped column
    /// to the given type.
    ///
    /// Use this method to convert the elements of the column to a data column
    /// of the given type via ``MLDataValueConvertible``. Unlike
    /// ``MLUntypedColumn/column(type:)`` , which doesn't alter its
elements,
    /// ``MLUntypedColumn/map(to:)`` converts the elements to the
destination
    /// type. For example, you can use ``MLUntypedColumn/map(to:)`` to
convert
    /// an untyped column of integers to a data column of strings.
    ///
    /// - Parameters:
    ///   - type: A metatype used to create a new data column of that type.
    ///
    /// - Returns: A new data column if the column's underlying type is
    /// convertible to given type; otherwise `nil`.
public func map T      T      ML DataColumn T
where T  MLDataValueConvertible

    /// Creates a subset of the column by removing all elements without a value.
    ///
    /// - Returns: A new column.
public func dropMissing      MLUntypedColumn

    /// Creates a modified copy of the column such that every missing element is
    /// replaced with the given value.
    ///
    /// - Parameters:
    ///   - value: A value to replace every undefined element.
    ///
    /// - Returns: A new `ML DataColumn` column.
public func fillMissing      MLDataValue
MLUntypedColumn

    /// Creates a subset of the column by removing all duplicate elements.
    ///
    /// - Returns: A new column.
    ///
    /// - Note: The new column may not preserve the order of the original
    /// column.
public func dropDuplicates      MLUntypedColumn

    /// Creates a subset of the column, given a number of initial elements.
    ///
    /// - Parameters:
```

```
    /// - maxLength: The largest number of elements to use from the
beginning of
    ///      the column. The default value is `10`.
    ///
    /// - Returns: A new column.
public func prefix _           Int 10
MLUntypedColumn

    /// Creates a subset of the column, given a number of final elements.
    ///
    /// - Parameters:
    ///   - maxLength: The largest number of elements to use from the end
of the
    ///      column. The default value is `10`.
    ///
    /// - Returns: A new column.
public func suffix _           Int 10
MLUntypedColumn

    /// Returns a new MLUntypedColumn containing values sorted by the
specified order.
    ///
    /// - Parameter byIncreasingOrder: A boolean indicating whether to
sort values in ascending or descending order.
    ///      The default is true, sorted by ascending order.
    ///
    /// - Returns: A MLUntypedColumn sorted by the specified order.
public func sort           Bool true
MLUntypedColumn

    /// Returns a new MLUntypedColumn by copying the original
MLUntypedColumn
    ///
    /// - Returns: A MLUntypedColumn which is a copy of the original
MLUntypedColumn.
public func copy           MLUntypedColumn

    /// Creates a new column by immediately evaluating any lazily applied data
    /// processing operations stored in the column.
    ///
    /// - Returns: A new column.
public func materialize throws MLUntypedColumn
```

```
@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLUntypedColumn
```

```
    /// Creates a new column of integers by converting the elements of another
    /// column.
    ///
```

```

    /// Use this initializer to create a column of integers from another column.
    /// As an example, to create a column with this initializer, first start
    /// with a column that is convertible to integers.
    ///
    /// ````swift
    /// let stringColumn = MLUntypedColumn(["1", "2", "3",
    "4", "5"])
    /// print(stringColumn)
    /// /* Prints...
    ///  ValueType: String
    ///  Values:      [1, 2, 3, 4, 5]
    /// */
    ///
    ///
    /// Then use ``MLUntypedColumn/init(ints:)`` to convert the
column to a
    /// column of integers.
    ///
    /// ````swift
    /// let intsColumn = MLUntypedColumn(ints: stringColumn)
    /// print(intsColumn)
    /// /* Prints...
    ///  ValueType: Int
    ///  Values:      [1, 2, 3, 4, 5]
    /// */
    ///
    ///
    /// - Parameters:
    ///   - ints: A column with elements that are convertible to integers.
    ///
    /// - Returns: A new untyped column of integers; otherwise an invalid
column
    ///   if any element of the given column cannot be converted to
    ///   <doc://com.apple.documentation/documentation/swift/int>.
public init      MLUntypedColumn

    /// Creates a new column of doubles by converting the elements of another
    /// column.
    ///
    /// Use this initializer to create a column of doubles from another column.
    /// As an example, to create a column with this initializer, first start
    /// with a column that is convertible to doubles.
    ///
    /// ````swift
    /// let stringColumn = MLUntypedColumn(["1.0", "2.0",
    "3.0", "4.0", "5.0"])
    /// print(stringColumn)
    /// /* Prints...
    ///  ValueType: String
    ///  Values:      [1.0, 2.0, 3.0, 4.0, 5.0]

```

```

////*/
///```
///
/// Then use ``MLUntypedColumn/init(doubles:)`` to convert the
column to a
/// column of doubles.
///
/// ``swift
/// let doublesColumn = MLUntypedColumn(doubles:
stringColumn)
/// print(doublesColumn)
/// /* Prints...
///   ValueType: Double
///   Values:      [1.0, 2.0, 3.0, 4.0, 5.0]
/// */
/// ```
///
/// - Parameters:
///   - doubles: A column with elements that are convertible to doubles.
///
/// - Returns: A new untyped column of doubles; otherwise an invalid
column
///   if any element of the given column cannot be converted to
///   <doc://com.apple.documentation/documentation/swift/double>.
public init MLUntypedColumn

/// Creates a new column of strings by converting the elements of another
/// column.
///
/// Use this initializer to create a column of strings from another column.
/// As an example, to create a column with this initializer, first start
/// with a column that is convertible to strings.
///
/// ``swift
/// let doublesColumn = MLUntypedColumn([1.0, 2.718, 3.14,
4.2, 5.1])
/// print(doublesColumn)
/// /* Prints...
///   ValueType: Double
///   Values:      [1.0, 2.718, 3.14, 4.2, 5.1]
/// */
/// ```
///
/// Then use ``MLUntypedColumn/init(strings:)`` to convert the
column to a
/// column of strings.
///
/// ``swift
/// let stringsColumn = MLUntypedColumn(strings:
doublesColumn)

```

```

    ///> print(stringsColumn)
    ///> /* Prints...
    ///>   ValueType: String
    ///>   Values:      [1, 2.718, 3.14, 4.2, 5.1]
    ///> */
    ///> ```
    ///>
    ///> - Parameters:
    ///>   - strings: A column with elements that are convertible to string.
    ///>
    ///> - Returns: A new untyped column of doubles; otherwise an invalid
column
    ///>     if any element of the given column cannot be converted to
    ///>     <doc://com.apple.documentation/documentation/swift/string>.
public init          MLUntypedColumn

    ///> Creates a new column of machine learning sequences by converting the
    ///> elements of another column.
    ///>
    ///> Use this initializer to create a column of sequences from another
    ///> column. As an example, to create a column with this initializer, first
    ///> start with a column that is convertible to sequences.
    ///>
    ///> ````swift
    ///> let intSequenceString = "[1, 2, 3]"
    ///> let intSequenceString2 = "[4, 5, 6]"
    ///> let stringsColumn =
MLUntypedColumn([intSequenceString, intSequenceString2])
    ///>
    ///> print(stringsColumn)
    ///> /* Prints...
    ///>   ValueType: String
    ///>   Values:      [[1, 2, 3], [4, 5, 6]]
    ///> */
    ///> ```
    ///>
    ///> Then use ``MLUntypedColumn/init(sequences:)`` to convert the
column to a
    ///> column of sequences.
    ///>
    ///> ````swift
    ///> let sequenceColumn = MLUntypedColumn(sequences:
stringsColumn)
    ///> print(sequenceColumn)
    ///> /* Prints...
    ///>   ValueType: Sequence
    ///>   Values:      [[DataValue(1), DataValue(2),
DataValue(3)],
    ///>                  [DataValue(4), DataValue(5),
DataValue(6)]]
```

```

/// *\
/// ``
///
/// - Parameters:
///   - sequences: A column with elements that are convertible to
Create ML sequences.
///
/// - Returns: A new untyped column of doubles; otherwise an invalid
column
///   if any element of the given column cannot be converted to
///   ``MLDataValue/SequenceType``.
public init MLUntypedColumn

/// Creates a new column of machine learning dictionaries by converting the
/// elements of another column.
///
/// Use this initializer to create a column of dictionaries from another
/// column. As an example, to create a column with this initializer, first
/// start with a column that is convertible to dictionaries.
///
/// ``swift
/// let intDictionaryString = "{1:\"one\", 2:\"two\",
3:\"three\"}"
/// let intDictionaryString2 = "{4:\"four\", 5:\"five\",
6:\"six\"}"
/// let stringsColumn =
MLUntypedColumn([intDictionaryString, intDictionaryString2])
///
/// print(stringsColumn)
/// /* Prints...
///   Value: String
///   Values: [{1:"one", 2:"two", 3:"three"}, {4:"four",
5:"five", 6:"six"}]
/// */
///
/// Then use ``MLUntypedColumn/init(dictionaries:)`` to convert
the column
/// to a column of dictionaries.
///
/// ``swift
/// let dictionaryColumn = MLUntypedColumn(dictionaries:
stringsColumn)
/// print(dictionaryColumn)
/// /* Prints...
///   Value: Dictionary
///   Values: [[1: one, 3: three, 2: two], [4: four,
5: five, 6: six]]
/// */
///

```

```

    /**
     * - Parameters:
     *   - dictionaries: A column with elements that are convertible to
     * Create ML dictionaries.
     */
    /**
     * - Returns: A new untyped column of doubles; otherwise an invalid
     * column
     *   if any element of the given column cannot be converted to
     *   ``MLDataValue/DictionaryType``.
public init MLUntypedColumn

    /// Creates a MLUntypedColumn of type MLMultiArray from the specified
    MLUntypedColumn if the values of
    /// the given MLUntypedColumn are convertible to
    MLDataValue.MultiArrayType.
    /**
     * - Parameter multiArrays: a MLUntypedColumn from which to
     * create a MLUntypedColumn with type
     *   MLDataValue.DictionaryType
     /**
     * - Returns: a MLUntypedColumn of type MultiArray from the specified
     * MLUntypedColumn if the given
     *   MLUntypedColumn's values are convertible to
     *   MLDataValue.MultiArrayType. Returns an invalid MLUntypedColumn
     *   if the elements in the given MLUntypedColumn are not convertible to
     *   MLDataValue.MultiArrayType.
public init MLUntypedColumn

```

```

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLUntypedColumn

    /// Accesses the element at the given position.
    /**
     * - Parameters:
     *   - index: The index of an element in the column.
    /**
     * - Returns: The value at the given index.
public subscript Int MLDataValue get

```

```

    /// Range-Based Slicing Support
@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLUntypedColumn

    /// Creates a subset of the column, given a range of elements.
    /**
     * - Parameters:
     *   - slice: A range of integers indicating which elements to include in
     * the new column.
    /**

```

```
    /// - Returns: A new column.
    public subscript Range Int MLUntypedColumn
get

    /// Creates a subset of the column, given a range expression of elements.
    ///
    /// - Parameters:
    ///   - slice: An integer range expression indicating which elements to
    include in the new column.
    ///
    /// - Returns: A new column.
    public subscript R R MLUntypedColumn where R
RangeExpression R Bound Int get

@available 10.15 15.0
@available
@available
extension MLUntypedColumn

    /// Provides a visualization for the data in the column.
    public func show any MLStreamingVisualizable

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLUntypedColumn CustomReflectable

    /// A view of the column for Xcode Playgrounds and lldb.
    public var customMirror Mirror get

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension MLUntypedColumn CustomStringConvertible
CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of the column.
    public var description String get

    /// A text representation of the column for debugging.
    public var debugDescription String get

    /// A description of the column shown in a playground.
    public var playgroundDescription Any get

    /// An image visualization of machine learning types.
@available macOS 10.15
@available
```

```
@available
public protocol MLVisualizable
CustomPlaygroundDisplayConvertible

    /// The visualization for a machine learning type as an image.
    var cgImage CGImage get

    /// A map of strings in a vector space that enable your app to find similar
    /// strings by looking at a string's neighbors.
    ///
    /// Use an ``MLWordEmbedding`` to configure and save a word embedding to a
    file,
    /// which you then add to your project in Xcode. Your project uses that word
    /// embedding file at runtime to create an
    ///
    <doc://com.apple.documentation/documentation/naturallanguage/nlembbeding>
    /// instance, which finds similar strings based on the proximity of their
    /// vectors.
    ///
    /// You configure a word embedding with a dictionary, keyed by strings which
    /// make up the _vocabulary_ of the word embedding. The value for each string is
    /// an array of doubles, which represents a vector. The length of the arrays is
    /// arbitrary but all arrays in a word embedding must be the same length. The
    /// length of the arrays determine the number of dimensions in the vector space.
    /// For example, the following listing creates a word embedding with four
    /// dimensions and a vocabulary of two strings.
    ///
    /// ````swift
    let wordEmbedding = try! MLWordEmbedding(dictionary: [
        "Hello" : [0.0, 1.2, 5.0, 0.0],
        "Goodbye" : [0.0, 1.3, -6.2, 0.1]
    ])
    ````
    ///
    /// Once you've configured an ``MLWordEmbedding``, save it to an
    ``.mlmodel`` file
    /// to include in your app.
    ///
    /// ````swift
    try wordEmbedding.write(toFile:
    "~/Desktop/WordEmbedding.mlmodel")
    ````
    ///
    /// A word embedding file can efficiently store many strings and their vectors.
@available macOS 10.15
@available
@available
public struct MLWordEmbedding
```

```

/// The word embedding contained within a Core ML model file.
public var model MLModel

/// The model configuration parameters.
public let modelParameters
MLWordEmbedding ModelParameters

/// The number of dimensions in the vocabulary embedding space.
public let dimension Int

/// The number of strings in the vocabulary.
public let vocabularySize Int

/// Creates a word embedding.
///
/// - Parameters:
///   - dictionary: A dictionary of strings and their embeddings.
///   - parameters: The model parameters.
public init String Double
MLWordEmbedding ModelParameters throws

@available macOS 10.15
@available
@available
extension MLWordEmbedding

/// Predicts neighbors.
///
/// The distance values are calculated with a formula determined by
///
<doc://com.apple.documentation/documentation/naturallanguage/nldistancetype>,
/// such as
///
<doc://com.apple.documentation/documentation/naturallanguage/nldistancetype/>
cosine.

///
/// - Parameters:
///   - text: A string in the embedding vocabulary.
///   - maxCount: The largest number of neighboring strings.
///   - maxDistance: The maximum allowed neighbor distance.
///   - distanceType: The type of distance formula to use for evaluating
a neighbor's distance from the input string.
///
/// - Returns: An array of neighboring strings and their distances to the
input string.
public func prediction String Int
10           Double 2.0          NLDistanceType
throws        String           Double

```

```
@available macOS 10.15
@available
@available
extension MLWordEmbedding

    /// Exports the word embedding as a Core ML model file at the specified URL.
    ///
    /// - Parameters:
    ///   - fileURL: The location in the file system to which the file should be written.
    ///   - metadata: Descriptive information to include with the exported model file.
    public func write URL
MLModelMetadata nil throws

    /// Exports the word embedding as a Core ML model file at the specified file path.
    ///
    /// - Parameters:
    ///   - path: A file system path where the model file should be written.
    ///   - metadata: Descriptive information to include with the exported model file.
    public func write String
MLModelMetadata nil throws
```

```
@available macOS 10.15
@available
@available
extension MLWordEmbedding

    /// Returns a Boolean value indicating whether the vocabulary contains the given string.
    ///
    /// - Parameter text: The string to find in the vocabulary.
    /// - Returns: `true` if the string was found in the vocabulary; otherwise, false.
    public func contains _ String Bool

    /// Calculates the distance between two strings in the vocabulary space.
    ///
    /// - Parameters:
    ///   - first: A string in the embedding vocabulary.
    ///   - second: Another string in the embedding vocabulary.
    ///   - distanceType: The metric to use to calculate the distance between the first and second strings.
    /// - Returns: The distance
```

```
public func distance String NLDistanceType Double
String NLDistancetypenumber
String NLDistanceType Double

    /// Accesses the vector associated with the given string in the vocabulary.
    ///
    /// - Parameter text: A string in the vocabulary.
    /// - Returns: The vector associated with the string if present in the word embedding; otherwise, `nil`.
public func vector String Double
String Double

@available macos 10.15
@available
@available
extension MLWordEmbedding : CustomStringConvertible
CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of the word embedding.
public var description String get
String get

    /// A text representation of the word embedding that's suitable for output during debugging.
public var debugDescription String get
String get

    /// A description of the word embedding shown in a playground.
public var playgroundDescription Any get
Any get

@available macos 10.15
@available
@available
extension MLWordEmbedding

    /// The model configuration parameters.
public struct ModelParameters : Sendable
ModelParameters Sendable

    /// The language of the word embedding.
public var language NLLanguage
NLLanguage

    /// The revision of the word embedding.
public var revision Int
Int

    /// Creates model parameters.
    ///
    /// - Parameters:
    ///   - language: The language of the text in the word embedding.
    ///   - revision: The revision of the word embedding.
public init NLLanguage nil Int
NLLanguage nil Int
```

1

```
@available macOS 10.15
@available
@available
extension MLWordEmbedding ModelParameters
CustomStringConvertible CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of the word embedding parameters.
    public var description String get

    /// A text representation of the word embedding parameters that's suitable for
    output during debugging.
    public var debugDescription String get

    /// A description of the word embedding parameters shown in a playground.
    public var playgroundDescription Any get

    /// A word-tagging model you train to classify natural language text at the word
    level.
    ///
    /// Use an ``MLWordTagger`` to create a custom word tagger to identify
    content
    /// that's relevant for your app, like product names and points of interest.
    ///
    /// To use your custom word tagger in the
    /// <doc://com.apple.documentation/documentation/naturallanguage> framework,
    /// save it to a model file and import it into an
    /// <doc://com.apple.documentation/documentation/naturallanguage/nlmodel>.
    Then
    /// add your custom
    /// <doc://com.apple.documentation/documentation/naturallanguage/nlmodel> to
    an
    /// <doc://com.apple.documentation/documentation/naturallanguage/nltagger>
    with
    /// its
    /// <doc://com.apple.documentation/documentation/naturallanguage/nltagger/
    2976627-setmodels>
    /// method.

@available macOS 10.14
@available
@available
public struct MLWordTagger @unchecked Sendable

    /// The token type of a word tagger, which is a string.
    public typealias Token String
```

```
/// The underlying Core ML model of the word tagger.  
public var model MLModel  
  
/// The configuration parameters that the word tagger used for training during  
initialization.  
public let modelParameters MLWordTagger ModelParameters  
  
/// Measurements of the tagger's performance on the training data set.  
public let trainingMetrics MLWordTaggerMetrics  
  
/// Measurements of the tagger's performance on the validation data set.  
public let validationMetrics MLWordTaggerMetrics  
  
/// Creates a word tagger.  
///  
/// - Parameters:  
///   - trainingData: An array of tuples containing the tokens and their  
labels.  
///   - parameters: The model parameters.  
@available macOS 10.15  
@available  
@available  
public init MLWordTagger Token  
String MLWordTagger ModelParameters
```

### throws

```
/// Creates a word tagger.  
///  
/// - Parameters:  
///   - trainingData: A data frame specifying training data.  
///   - tokenColumn: The name of the token column in the training data  
frame.  
///   - labelColumn: The name of the label column in the training data  
frame.  
///   - parameters: The model parameters.  
@available macOS 13.0  
@available  
@available  
public init DataFrame String  
String MLWordTagger ModelParameters
```

### throws

```
/// Creates a word tagger.  
///  
/// - Parameters:  
///   - trainingData: A table specifying training data.  
///   - tokenColumn: The name of the token column in the training data
```

```
frame.  
    /// - labelColumn: The name of the label column in the training data  
frame.  
    /// - parameters: The model parameters.  
    @available 10.14 13.0  
        "Use DataFrame instead of MLDataTable when  
initializing."  
    @available  
    @available  
    public init  
        MLDataTable  
String String  
MLWordTagger ModelParameters  
nil throws  
  
@available macOS 10.14  
@available  
@available  
extension MLWordTagger  
  
    /// Computes evaluation metrics.  
    ///  
    /// - Parameter labeledTokens: An array of token and label pairs.  
    /// - Returns: Word tagger metrics.  
    public func evaluation  
        MLWordTagger Token String  
MLWordTaggerMetrics  
  
    /// Computes evaluation metrics.  
    ///  
    /// - Parameters:  
    ///     - labeledTokens: An data frame containing tokens and labels.  
    ///     - tokenColumn: The name of the token column in the data frame.  
    ///     - labelColumn: The name of the label column in the data frame.  
    /// - Returns: Word tagger metrics.  
    @available macOS 13.0  
    @available  
    @available  
    public func evaluation  
        String DataFrame  
        String  
MLWordTaggerMetrics  
  
    /// Computes evaluation metrics.  
    ///  
    /// - Parameters:  
    ///     - labeledTokens: A table containing tokens and labels.  
    ///     - tokenColumn: The name of the token column in the data frame.  
    ///     - labelColumn: The name of the label column in the data frame.  
    /// - Returns: Word tagger metrics.  
    @available 10.14 13.0
```

```
@available  
@available  
public func evaluation  
    String          String  
MLWordTaggerMetrics
```

```
@available macOS 10.14  
@available  
@available  
extension MLWordTagger  CustomStringConvertible  
CustomDebugStringConvertible  
CustomPlaygroundDisplayConvertible  
  
/// A text representation of the word tagger.  
public var description  String  get  
  
/// A text representation of the word tagger that's suitable for output  
/// during debugging.  
public var debugDescription  String  get  
  
/// A description of the word tagger in a playground.  
public var playgroundDescription  Any  get
```

```
@available macOS 10.14  
@available  
@available  
extension MLWordTagger  
  
/// The algorithm type.  
public enum ModelAlgorithmType  Sendable  
  
/// A conditional random field algorithm.  
case crf           Int  
  
/// A transfer-learning algorithm.  
@available macOS 11.0  
@available  
@available  
case  
transferLearning MLWordTagger FeatureExtractorType  
Int  1
```

```
@available macOS 10.14  
@available  
@available
```

```
extension MLWordTagger

    /// Parameters that specify model training parameters and validation data.
    public struct ModelParameters

        /// The algorithm type.
        public var algorithm MLWordTagger.ModelAlgorithmType

        /// The language setting.
        public var language NLLanguage

        /// The word tagger's validation dataset as a data table.
        @available(10.15, 16.0)
        public var validationData MLDataTable

        /// The name of the column containing the tokens in the validation data
        table.
        @available(10.15, 16.0)
        public var tokenColumnValidationData String

        /// The name of the column containing the token labels in the validation
        data table.
        @available(10.15, 16.0)
        public var labelColumnValidationData String

        /// The maximum training iterations.
        @available(macOS 11.0)
        @available
        @available
        public var maxIterations Int

        /// The validation dataset.
        ///
        /// The default value is an
        ///

``MLWordTagger/ModelParameters-swift.struct/ValidationData-
swift.enum/split(strategy)``
        /// instance with the ``MLSplitStrategy/automatic`` split
```

```

strategy```,
    /// which automatically generates the validation dataset by partitioning
up to 10% of the training dataset.
@available macOS 10.15
@available
@available
public var validation
MLWordTagger ModelParameters ValidationData

    /// Creates model parameters.
    ///
    /// - Parameters:
    ///   - validation: The validation data source.
    ///   - algorithm: The algorithm type.
    ///   - language: The language of the text to tag.
@available macOS 10.15
@available
@available
public init
MLWordTagger ModelParameters ValidationData
    MLWordTagger ModelAlgorithmType
        1           NLLanguage      nil

    /// Creates model parameters.
    ///
    /// - Parameters:
    ///   - validationData: The validation data table.
    ///   - algorithm: The algorithm type.
    ///   - language: The language of the text to tag.
    ///   - tokenColumnValidationData: The name of the column
containing the tokens in the validation data table.
    ///   - labelColumnValidationData: The optional name of the
column containing the token labels in the validation data table.
@available 10.14
10.15          "Use the validation property instead."
@available 15.0          16.0
    "Use the validation property instead."
@available
public init          MLDataTable
MLWordTagger ModelAlgorithmType
    1           String      nil
NLLanguage      nil
    String      nil

    /// Creates model parameters.
    ///
    /// - Parameters:
    ///   - validationData: The validation data of token and label
pairs.
    ///   - algorithm: The algorithm type.
    ///   - language: The language of the text to tag.

```

```
@available 10.14
10.15      "Use the validation property instead."
@available 15.0 16.0
      "Use the validation property instead."
@available
public init
MLWordTagger Token String
MLWordTagger ModelAlgorithmType 1
NLLanguage nil
```

```
@available macOS 10.14
@available
@available
extension MLWordTagger

    /// Exports the word tagger as a Core ML model file at the specified URL.
    ///
    /// - Parameters:
    ///   - fileURL: The location in the file system to which the file should be
written.
    ///   - metadata: Descriptive information to include with the exported
model file.
public func write URL
MLModelMetadata nil throws

    /// Exports the word tagger as a Core ML model file at the specified file path.
    ///
    /// - Parameters:
    ///   - path: A file system path where the model file should be written.
    ///   - metadata: Descriptive information to include with the exported
model file.
public func write String
MLModelMetadata nil throws
```

```
@available macOS 10.14
@available
@available
extension MLWordTagger

    /// The feature extractors that are available to train a word tagger using with
the transfer-learning algorithm option.
@available macOS 11.0
@available
@available
public enum FeatureExtractorType Sendable

    /// A feature extractor that provides embeddings for words, based on
```

their in-context use.

///

/// Dynamic embedding requires certain downloadable assets to be present on device at training time.

/// Training will throw an error if the specified language is unavailable at runtime. Asset downloads are

/// managed in the background automatically by the OS when a new language is configured in device

/// settings, such as when adding a new keyboard language or changing the preferred language.

**@available**

"elmoEmbedding"

**case** dynamicEmbedding

11.0

14.0

/// A feature extractor that provides ELMo contextual word embeddings.

///

/// ELMo embeddings requires certain downloadable assets to be present on device at training

/// time. Training will throw an error if the specified language is unavailable at runtime. Asset downloads are

/// managed in the background automatically by the OS when a new language is configured in device

/// settings, such as when adding a new keyboard language or changing the preferred language.

**@available macOS 14.0**

**case** elmoEmbedding

/// A feature extractor that provides BERT contextual word embeddings.

///

/// The embeddings consider the context from left-to-right and right-to-left simultaneously.

///

/// BERT embedding requires certain downloadable assets to be present on device at training

/// time. Training will throw an error if the specified language is unavailable at runtime. Asset downloads are

/// managed in the background automatically by the OS when a new language is configured in device

/// settings, such as when adding a new keyboard language or changing the preferred language.

**@available macOS 14.0**

**case** bertEmbedding

/// Returns a Boolean value indicating whether two values are equal.

///

/// Equality is the inverse of inequality. For any values `a` and `b`,  
/// `a == b` implies that `a != b` is `false`.

///

/// - **Parameters:**

/// - `lhs`: A value to compare.

/// - `rhs`: Another value to compare.

```

public static func
MLWordTagger FeatureExtractorType
MLWordTagger FeatureExtractorType      Bool

    /// Hashes the essential components of this value by feeding them into
the
    /// given hasher.
///
/// Implement this method to conform to the `Hashable` protocol. The
/// components used for hashing must be the same as the components
compared
    /// in your type's `==` operator implementation. Call
`hasher.combine(_:)`
    /// with each of these components.
///
/// - Important: In your implementation of `hash(into:)`,
/// don't call `finalize()` on the `hasher` instance provided,
/// or replace it with a different instance.
/// Doing so may become a compile-time error in the future.
///
/// - Parameter hasher: The hasher to use when combining the
components
    /// of this instance.
public func hash           inout Hasher

    /// The hash value.
///
/// Hash values are not guaranteed to be equal across different
executions of
    /// your program. Do not save hash values to use during a future
execution.
///
/// - Important: `hashValue` is deprecated as a `Hashable`
requirement. To
    /// conform to `Hashable`, implement the `hash(into:)`
requirement instead.
    /// The compiler provides an implementation for `hashValue` for
you.
public var hashValue Int get

```

```

@available macOS 10.14
@available
@available
extension MLWordTagger

    /// Predicts a tag for the input string.
///
/// - Parameter text: The string to tag.

```

```

    /// - Returns: An array of tags for the tokens in the string.
    public func prediction           String throws
String

    /// Predicts a tag for each token in the specified array.
    ///
    /// - Parameter tokens: An array of tokens to tag.
    /// - Returns: An array of tags for the tokens.
    public func prediction           MLWordTagger Token
throws String

    /// Predicts sequences of labels, token locations, and token lengths from the
input strings.
    ///
    /// - Parameter texts: A sequence of strings to tokenize and tag.
    /// - Returns: A `DataFrame` containing predicted labels, token
locations, and token lengths.
@available macOS 13.0
@available
@available
public func predictions S           S throws
DataFrame where S Sequence S Element String

    /// Predicts sequences of labels, token locations, and token lengths from the
input column containing strings.
    ///
    /// - Parameter texts: A column of strings to tokenize and tag.
    /// - Returns: A table containing predicted labels, token locations, and
token lengths.
@available 10.14
@available
@available
public func predictions           MLDataTable String
throws MLDataTable

    /// Predicts tags and confidence scores for the input string. Predicts tags and
confidence scores for the input string.
    ///
    /// - Parameter text: The string to tag.
    /// - Returns: An array of dictionaries. Each dictionary corresponds to a
token in the input text string.
    /// A dictionary entry contains a tag prediction with its associated confidence
score.
@available macOS 11.0
@available
@available
public func predictionWithConfidence String
throws String Double

    /// Predicts tags and confidence scores for each token in the specified token

```

```
array.

    /**
     * - Parameter tokens: An array of tokens to tag.
     * - Returns: An array of dictionaries. Each dictionary corresponds to a
     * token in the input token array.
     * A dictionary entry contains a tag prediction with its associated confidence
     * score.
     */
    @available macOS 11.0
    @available
    @available
    public func predictionWithConfidence
        MLWordTagger Token throws String Double

@available macOS 10.14
@available
@available
extension MLWordTagger ModelAlgorithmType
CustomStringConvertible CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /**
     * A text representation of the model algorithm type.
     */
    public var description String get

    /**
     * A text representation of the algorithm type that's suitable for output during
     * debugging.
     */
    public var debugDescription String get

    /**
     * A description of the algorithm type in a playground.
     */
    public var playgroundDescription Any get

@available macOS 10.14
@available
@available
extension MLWordTagger ModelParameters

    /**
     * The validation data.
     */
    @available macOS 10.15
    @available
    @available
    public enum ValidationData

        /**
         * Generates the validation data by splitting the training dataset.
         */
        /**
         * By default, model parameters use this approach to specify the
         * validation data.
         */
        case split MLSplitStrategy

        /**
         * Sets the validation data from the provided data table.
         */

```

```
@available 10.15 14.0
@available
@available
case table MLDataTable String
String

/// Set validation data from the DataFrame provided.
@available macOS 13.0 iOS 16.0 tvOS 16.0
case dataFrame DataFrame String
String

/// Sets the validation data from a list of tokens and labels.
case tuples MLWordTagger Token
String

/// Doesn't set validation data.
case none
```

```
@available macOS 10.14
@available
@available
extension MLWordTagger ModelParameters
CustomStringConvertible CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

/// A text representation of the model parameters.
public var description String get

/// A text representation of the model parameters that's suitable for output
during debugging.
public var debugDescription String get

/// A description of the model parameters in a playground.
public var playgroundDescription Any get

@available macOS 11.0
@available
@available
extension MLWordTagger FeatureExtractorType
CustomStringConvertible CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

/// A text representation of a feature extractor type.
public var description String get

/// A text representation of the feature extractor that's suitable for output
during debugging.
```



```
@available
@available
public var precisionRecall MLDataTable get

    /// A data frame listing the precision and recall percentages for each class.
@available macOS 15.0
public var precisionRecallDataFrame DataFrame get

@available macOS 10.14
@available
@available
extension MLWordTaggerMetrics CustomStringConvertible
CustomDebugStringConvertible
CustomPlaygroundDisplayConvertible

    /// A text representation of the word tagger metrics.
public var description String get

    /// A text representation of the word tagger metrics that's suitable for
    /// output during debugging.
public var debugDescription String get

    /// A description of the word tagger metrics shown in a playground.
public var playgroundDescription Any get

    /// Generates a streaming plot visualization of the two untyped columns.
@available 10.15 14.0
@available
@available
public func show _ MLUntypedColumn _ MLUntypedColumn
any MLStreamingVisualizable

    /// Generates a streaming plot visualization of the two data columns.
@available 10.15 14.0
@available
@available
public func show ElementX ElementY
ML DataColumn ElementX _ ML DataColumn ElementY any
MLStreamingVisualizable where ElementX
MLDataValueConvertible ElementY MLDataValueConvertible

    /// Generates a streaming visualization of the untyped column.
@available 10.15 14.0
@available
@available
public func show _ MLUntypedColumn any
MLStreamingVisualizable
```

```
/// Generates a streaming visualization of the data column.  
@available 10.15 14.0  
@available  
@available  
public func show Element _ MLDataTable Element  
any MLStreamingVisualizable where Element  
MLDataValueConvertible  
  
/// Generates a streaming visualization of the data table.  
@available 10.15 14.0  
@available  
@available  
public func show _ MLDataTable any  
MLStreamingVisualizable  
  
/// Creates a deterministic number from the current system time to seed  
/// random-number generators.  
@available macOS 10.14 iOS 15.0 tvOS 16.0  
public func timestampSeed Int  
  
@available 10.15 14.0  
@available  
@available  
extension Int MLIdentifier  
  
/// The value of the unique identifier wrapped in a data value.  
public var identifierValue MLDataValue get  
  
@available 10.15 14.0  
@available  
@available  
extension String MLIdentifier  
  
/// The value of the unique identifier wrapped in a data value.  
public var identifierValue MLDataValue get  
  
@available macOS 10.14 iOS 15.0 tvOS 16.0  
extension Array where Element MLDataValueConvertible  
  
/// Constructs an Array with the elements of a DataColumn.  
public init _ MLDataTable Element  
  
@available macOS 10.14 iOS 15.0 tvOS 16.0  
extension Array MLDataValueConvertible where Element  
MLDataValueConvertible
```

```
    /// The underlying type the conforming type uses when it wraps itself in a
    /// data value.
    ///
    /// See ``MLDataValue/ValueType`` for a list of available options.
    public static var dataType MLDataValue ValueType
get

    /// Creates an instance of the conforming type from a data value.
    public init MLDataValue

    /// The value of the conforming type's instance wrapped in a data value.
    public var value MLDataValue get

extension Int32

    @available macOS 12.0 iOS 15.0 tvOS 16.0
    public static var mlMultiArrayType
MLMultiArrayType get

extension Float

    @available macOS 12.0 iOS 15.0 tvOS 16.0
    public static var mlMultiArrayType
MLMultiArrayType get

extension Double

    @available macOS 12.0 iOS 15.0 tvOS 16.0
    public static var mlMultiArrayType
MLMultiArrayType get

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension Dictionary : MLDataValueConvertible where Key
MLDataValueConvertible Value : MLDataValueConvertible

    /// The underlying type the conforming type uses when it wraps itself in a
    /// data value.
    ///
    /// See ``MLDataValue/ValueType`` for a list of available options.
    public static var dataType MLDataValue ValueType
get

    /// Creates an instance of the conforming type from a data value.
    public init MLDataValue
```

```
public init
MLDataValue DictionaryType

/// The value of the conforming type's instance wrapped in a data value.
public var dataValue MLDataValue get

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension Int MLDataValueConvertible

/// The underlying type the conforming type uses when it wraps itself in a
/// data value.
///
/// See ``MLDataValue/ValueType`` for a list of available options.
public static var dataValueType MLDataValue ValueType
get

/// Creates an instance of the conforming type from a data value.
public init MLDataValue

/// The value of the conforming type's instance wrapped in a data value.
public var dataValue MLDataValue get

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension Bool MLDataValueConvertible

/// The underlying type the conforming type uses when it wraps itself in a
/// data value.
///
/// See ``MLDataValue/ValueType`` for a list of available options.
public static var dataValueType MLDataValue ValueType
get

/// Creates an instance of the conforming type from a data value.
public init MLDataValue

/// The value of the conforming type's instance wrapped in a data value.
public var dataValue MLDataValue get

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension Int64 MLDataValueConvertible

/// The underlying type the conforming type uses when it wraps itself in a
/// data value.
///
/// See ``MLDataValue/ValueType`` for a list of available options.
```

```
    public static var dataValueType MLDataValue ValueType
get

    /// Creates an instance of the conforming type from a data value.
public init MLDataValue

    /// The value of the conforming type's instance wrapped in a data value.
public var dataValue MLDataValue get

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension Double MLDataValueConvertible

    /// The underlying type the conforming type uses when it wraps itself in a
    /// data value.
    ///
    /// See ``MLDataValue/ValueType`` for a list of available options.
public static var dataValueType MLDataValue ValueType
get

    /// Creates an instance of the conforming type from a data value.
public init MLDataValue

    /// The value of the conforming type's instance wrapped in a data value.
public var dataValue MLDataValue get

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension String MLDataValueConvertible

    /// The underlying type the conforming type uses when it wraps itself in a
    /// data value.
    ///
    /// See ``MLDataValue/ValueType`` for a list of available options.
public static var dataValueType MLDataValue ValueType
get

    /// Creates an instance of the conforming type from a data value.
public init MLDataValue

    /// The value of the conforming type's instance wrapped in a data value.
public var dataValue MLDataValue get

@available macOS 10.14 iOS 15.0 tvOS 16.0
extension Array where Element MLDataValue

    /// Constructs an Array with the elements of an MLUntypedColumn.
public init _ MLUntypedColumn
```

```
/// A global constant that defines the domain for Create ML errors.  
@available macOS 10.14 iOS 15.0 tvOS 16.0  
public var MLCreateErrorDomain String
```