

Machine Learning in iOS

Using Core ML & Create ML



Jeriel Ng (@jerielng)

Mobile Software Engineer

Introducing Apple's Machine Learning



Core ML

Integrating a trained model
into your app



Create ML

Training your own model to be
used in your app via Core ML

How much machine learning do I need to know?

- Practically
 - Cleaning a dataset
- Conceptually
 - High-level approach of machine learning
 - Common ML vocabulary
 - Types of ML algorithms

Short Answer: Not much

Most work is done under the hood by Apple

Overview: What can Core ML do?

- Audio Classification
- Computer Vision
 - Image Classification
 - Object Detection
- Natural Language Processing
- Tabular Data Analysis
 - Regression
 - Classification
 - Recommendations



Package a model into .mlmodel file



Drag it into your Xcode project

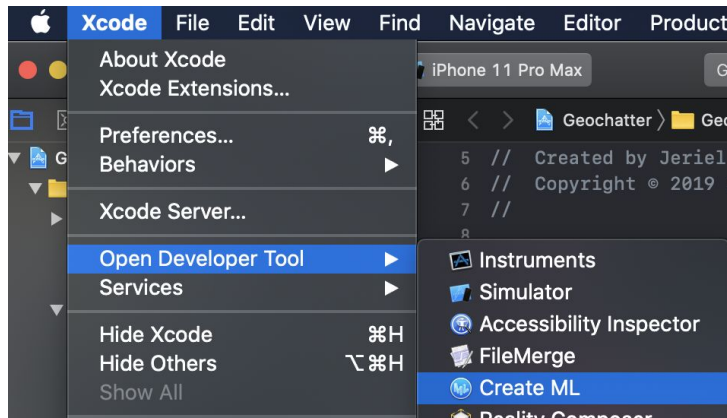
Getting a model - What are my options?

```
import Foundation
import CreateML

let trainingDataUrl = Bundle.main.url(forResource: "MOCK_DATA", withExtension: "csv")!
let trainingDataTable = try MLDataTable(contentsOf: trainingDataUrl)

let testingDataUrl = Bundle.main.url(forResource: "MOCK_DATA-2", withExtension: "csv")!
let testingDataTable = try MLDataTable(contentsOf: testingDataUrl)

if #available(OSX 10.15, *) {
    let recommender = try MLRecommender.init(trainingData: trainingDataTable, userColumn:
    let _ = recommender.evaluation(on: testingDataTable, userColumn: "id", itemColumn: "i
    print("Success")
} else {
    print("Failure")
}
```



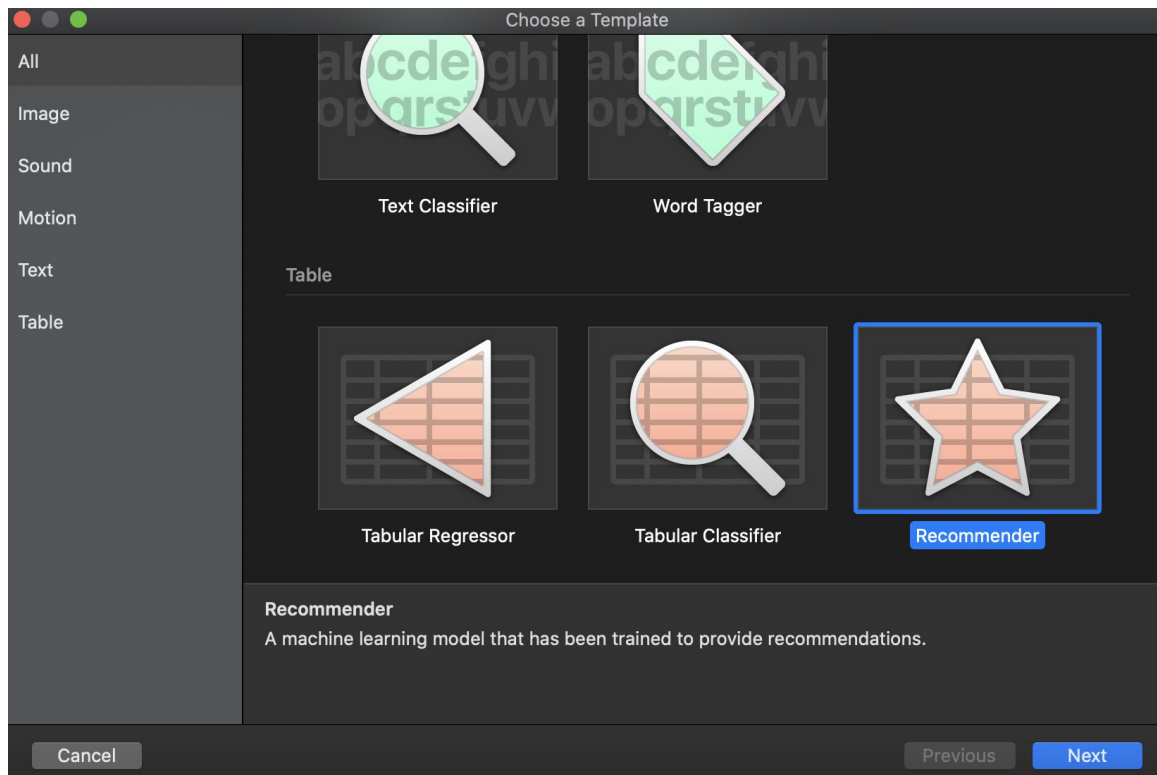
1. Use Create ML in Xcode **playground**

2. Use **Create ML app** (Introduced 2019)

3. Generate using **CoreMLTools** in Python

4. Download a pre-trained model

Creating your own model with Create ML



Today, we'll build a recommendation engine in the **Create ML app**

Creating your own model with Create ML

The screenshot displays the Create ML application interface for a project named "SongRecommender.mlproj". The interface is divided into several sections:

- Project:** Shows the project name "SongRecommender".
- Model Sources:** Lists "SongRecommender 1" as the selected model source.
- Data Inputs:** Contains fields for "Training Data" and "Testing Data". The "Training Data" field has a "Choose" button, which is highlighted by a yellow box with the text "JSON or CSV file".
- Output:** Displays the generated model file "UpsellRecommender.mlmodel" with a size of 84 KB. This section is highlighted by a red box.

The "Output" section details the model's metadata:

- File Name: UpsellRecommender.mlmodel
- Size: 84 KB
- Model Name: UpsellRecommender
- Author: Jeriel Ng
- License: License
- Description: A machine learning model that has been trained to provide recommendations.

A status bar at the bottom indicates "Training data required".

Creating your own model with Create ML

▼ Model Class

 UpsellRecommender_1

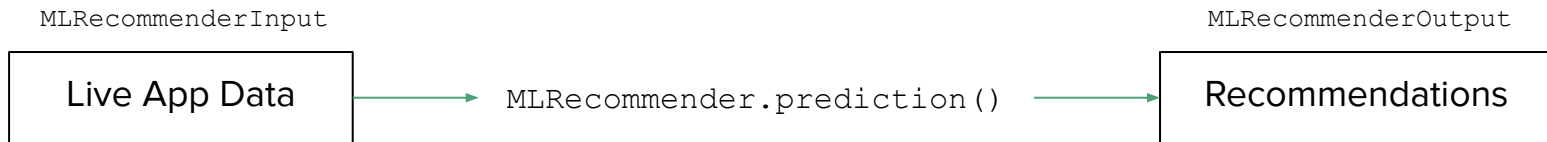
Model class can be viewed by adding model to a workspace and target.

Each MLModel comes with a generated Swift class

▼ Prediction

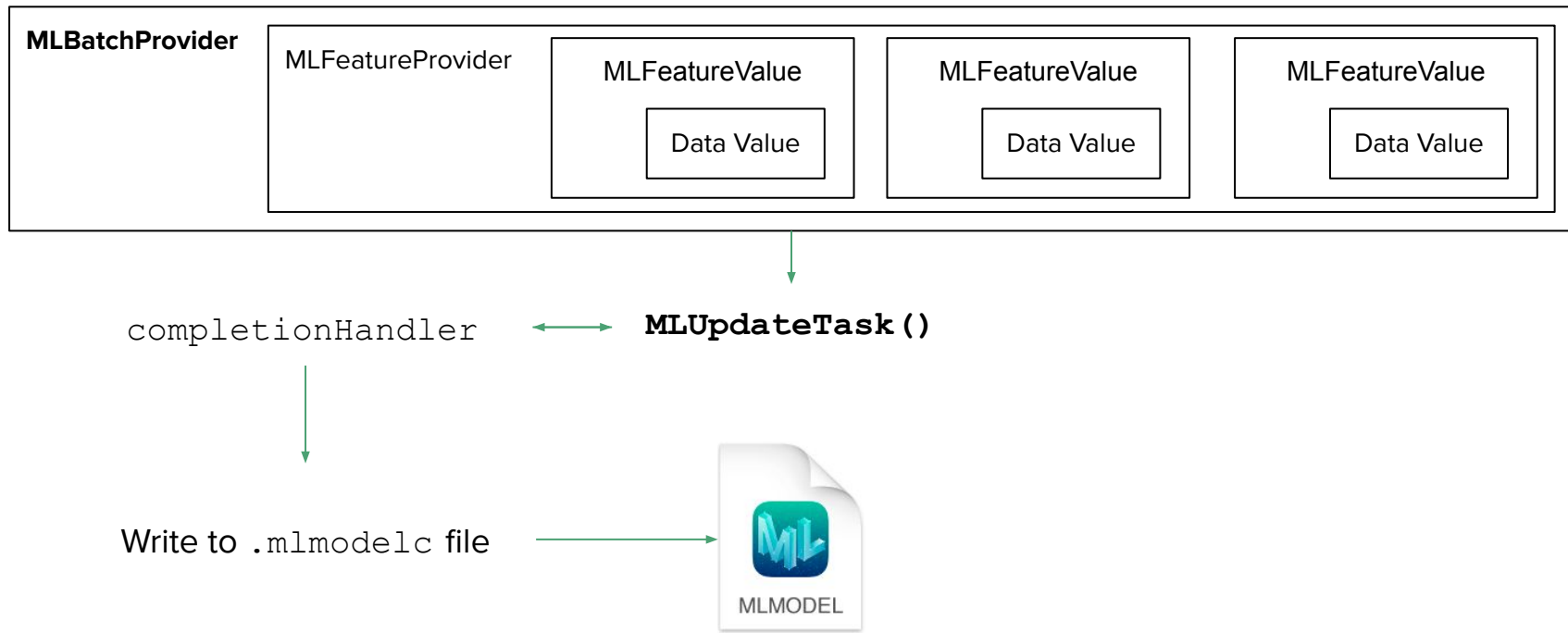
| Name | Type | Description |
|-----------------|-----------------------------|--|
| ▼ Inputs | | |
| items | Dictionary (Int64 → Double) | The list of items used to generate the recommendations. |
| k | Int64 | The number of items to return on a recommendation. |
| restrict | Sequence (Int64 0) | A sequence of items from which to generate recommendations. |
| exclude | Sequence (Int64 0) | A sequence of items to exclude from recommendations. Defaults to the input item list if not given. |
| ▼ Outputs | | |
| recommendations | Sequence (Int64 0) | The recommended items in order from most relevant to least relevant. |
| scores | Dictionary (Int64 → Double) | The scores for the recommended items, given as a dictionary of items and the corresponding scores. |

Core ML: Integrating your model



```
private func createSuggestionsMapFromRecommender() -> [Int: [Int64]] {
    var suggestionsMap = [Int: [Int64]]()
    if #available(iOS 13.0, *) {
        var itemsInCart: [Int64: Double] = [:]
        var itemsToExclude: [Int64] = []
        for itemId in cartButler.salesItemIds {
            itemsInCart.updateValue(0, forKey: itemId)
            itemsToExclude.append(itemId)
        }
        let modelInput = UpsellRecommender_1Input(items: itemsInCart, k: 4, restrict_: [], exclude: itemsToExclude)
        guard let output: UpsellRecommender_1Output = try? upsellRecommender.prediction(input: modelInput) else { return [:] }
        suggestionsMap.updateValue(output.recommendations, forKey: 0)
    }
    return suggestionsMap
}
```

Core ML: Updating your model



Core ML: Updating your model

```
private func updateRecommenderModel() {
    guard #available(iOS 13.0, *),
        let order = self.order else { return }
    var orderItemsDictionary: [AnyHashable: NSNumber] = [:]
    for orderItem in order.lineItems {
        orderItemsDictionary.updateValue(0, forKey: orderItem.menuItemId)
    }
    guard let updateValues = try? MLFeatureValue(dictionary: orderItemsDictionary) else { return }
    let dataPointFeatures: [String: MLFeatureValue] = ["items": updateValues]
    guard let featureProvider = try? MLDictionaryFeatureProvider(dictionary: dataPointFeatures) else { return }
    let trainingData = MLArrayBatchProvider(arrav: [featureProvider])
    guard let updateTask = try? MLUpdateTask(forModelAt: UpsellRecommender_1.urlOfModelInThisBundle, trainingData: trainingData, configuration: nil,
        completionHandler: { context in
            let updatedModel = context.model
            do {
                _ = try updatedModel.write(to: UpsellRecommender_1.urlOfModelInThisBundle)
            } catch { return }
        }) else { return }
    updateTask.resume()
}
```

Restrictions:

- MLModel must have `isUpdatable = true`
- Only on k-Nearest Neighbors and Neural Network models (Not available through Create ML)

Core ML: How to tell if your model is updatable?

▼ Prediction

| Name | Type | Description |
|-------------------|------------------------------|--|
| ▼ Inputs | | |
| items | MultiArray (Float32 128) | Input vector to classify |
| ▼ Outputs | | |
| recommendations | String | Predicted label. Defaults to 'defaultLabel' |
| recommendation... | Dictionary (String → Double) | Probabilities / score for each possible label. |

▼ Update

| Name | Type | Description |
|-----------------|--------------------------|--|
| ▼ Inputs | | |
| items | MultiArray (Float32 128) | Example input vector |
| recommendations | String | Associated true label of each example vector |

▼ Parameters

| Name | Type | Default | Description |
|-------------------|-------|---------|---------------------------------------|
| ▼ Model | | | |
| numberOfNeighb... | Int64 | 4 | Number of neighbors to use for pre... |

Drawbacks: Framework is still very new

- Core ML features limited to certain devices and iOS versions
- Some features limited to only certain types of models
- Structure of data can be highly specific
 - Must follow specific columns
 - Must follow specific data types
- Create ML tooling limited to certain versions of macOS

Further Resources

- [Pre-trained open source models](#)
- [CoreMLTools](#): Train a model using Python
- Core ML Tutorials
 - [Using your model to make predictions](#)
 - [Updating your existing model](#)

Questions
