Home

References

Alairas.Common

- `BaseLineExtraction`
- `BasicMetadataExtraction`
- `SimpleRenderingTransformation`

SigStat.Common

- `ArrayExtension`
- `Baseline`
- `Configuration`
- `DataSet`
- `FeatureAttribute`
- `FeatureDescriptor`
- `FeatureDescriptor<T>`
- `Features`
- `IClassification`
- `IClassificationMethods`

- ITransformation
- ITransformationMethods
- Loop
- MathHelper
- Matrix
- Origin
- PipelineBase
- Signature
- Signer
- Vector

SigStat.Common.Algorithms

- Dtw
- DtwPy
- HSCPThinningStep
- PatternMatching3x3

SigStat.Common.Helpers

- ConfigurationHelper
- ILogger
- IProgress
- LogEntry
- Logger
- LogLevel

SigStat.Common.Loaders

- DataSetLoader
- IDataSetLoader
- ImageLoader
- ImageSaver
- Svc2004
- Svc2004Loader

SigStat.Common.Model

- ApproximateLimit
- BenchmarkResults
- Result
- Sampler
- ThresholdResult
- Verifier
- VerifierBenchmark

SigStat.Common.Pipeline

- IClassificationModel
- IClassifier
- IPipelineIO
- ParallelTransformPipeline
- SequentialTransformPipeline

SigStat.Common.PipelineItems.Classifiers

- DTWClassifier
- WeightedClassifier

SigStat.Common.PipelineItems.Markers

- LogMarker
- TimeMarkerStart
- TimeMarkerStop

SigStat.Common.Transforms

- AddConst
- AddVector
- ApproximateOnlineFeatures
- Binarization
- BinaryRasterizer
- CentroidExtraction
- CentroidTranslate

- ComponentExtraction
- ComponentSorter
- ComponentsToFeatures
- EndpointExtraction
- Extrema
- HSCPThinning
- ImageGenerator
- Map
- Multiply
- Normalize
- OnePixelThinning
- PrepareForThinning
- RealisticImageGenerator
- Resize
- TangentExtraction
- TimeReset
- Translate
- Trim

Alairas.Common

BaseLineExtraction

```
public class Alairas.Common.BaseLineExtraction
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
void	Run(Signature input)	
void	Transform(Signature signature)	

Static Methods

Type	Name	Summary
	GetComponentLowerEnvelopes(Image<Rgba32>	Extracts lower envelope for

`List<List<Point>>`	image)	each component
`Baseline`	GetLineOfBestFit(`List<Point>` points)	Megkeresi a megadott pontokra legjobban illeszkedő egyenest. Képes még ennek az egyenesnek különböző hibamértékeinek kiszámítására, azonban jelenleg ezzel nem foglalkozom, hiszen ebben a speciális esetben szinte biztosan elég jól illeszkedő egyenest kapunk eredményül. Az algorimus kimenete nem egy egyenes, hanem egy egy vektor, mely az egyenes egy szakasza. Felteszem, hogy a pontok X koordináta szerint rendezettek, így az első és utolsó pont X koordinátája közötti szakaszt adom vissza az egyenesből. Azaz: Paraméterként egy előzőleg megtalált komponenst kap, kimenete pedig az adott komponens alapvonala.

BasicMetadataExtraction

```
public class Alairas.Common.BasicMetadataExtraction
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
`void`	Run(`Signature` input)	
`void`	Transform(`Signature` signature)	

Static Properties

Type	Name	Summary
`Double`	Trim	

SimpleRenderingTransformation

Renders an image of the signature based on the available online information (X,Y,Dpi)

```
public class Alairas.Common.SimpleRenderingTransformation
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
`void`	Run(`Signature` input)	
`void`	Transform(`Signature` signature)	

SigStat.Common.Algorithms

Dtw

Dynamic Time Warping algorithm

```
public class SigStat.Common.Algorithms.Dtw
```

Properties

Type	Name	Summary
`List<ValueTuple<Int32, Int32>>`	ForwardPath	Gets the list of points representing the shortest path.

Methods

Type	Name	Summary
`Double`	Compute(`Double[][]` signature1, `Double[][]` signature2)	Generate shortest path between the two sequences.

DtwPy

```
public static class SigStat.Common.Algorithms.DtwPy
```

Static Methods

Type	Name	Summary
`Double`	Dtw(`IEnumerable<P>` sequence1, `IEnumerable<P>` sequence2, `Func<P, P, Double>` distance)	
`Double`	EuclideanDistance(`Double[]` p1, `Double[]` p2)	

HSCPThinningStep

HSCP thinning algorithm http://www.ppgia.pucpr.br/~facon/Afinamento/1987holt.pdf (http://www.ppgia.pucpr.br/~facon/Afinamento/1987holt.pdf)

public class SigStat.Common.Algorithms.HSCPThinningStep

Properties

Type	Name	Summary
`Nullable<Boolean>`	ResultChanged	Gets whether the last `SigStat.Common.Algorithms.HSCPThinningStep.Scan(System.Boolean[0:` call was effective.

Methods

Type	Name	Summary
`Boolean[,]`	Scan(`Boolean[,]` b)	Does one step of the thinning. Call it iteratively while ResultChanged.

PatternMatching3x3

Binary 3x3 pattern matcher with rotating option.

public class SigStat.Common.Algorithms.PatternMatching3x3

Methods

Type	Name	Summary
`Boolean`	Match(`Boolean[,]` input)	Match the 3x3 input with the 3x3 pattern.
`Boolean`	RotMatch(`Boolean[,]` input)	Match the 3x3 input with the 3x3 pattern from all 4 directions.

SigStat.Common.Helpers

ConfigurationHelper

public class SigStat.Common.Helpers.ConfigurationHelper

Static Methods

Type	Name	Summary
`Configuration`	Load()	

ILogger

Enables logging by exposing a `SigStat.Common.Helpers.Logger` property.

public interface SigStat.Common.Helpers.ILogger

Properties

Type	Name	Summary
`Logger`	Logger	Gets or sets the attached `SigStat.Common.Helpers.Logger` object used to log messages.

IProgress

Enables progress tracking by expsoing the `SigStat.Common.Helpers.IProgress.Progress` property and the `SigStat.Common.Helpers.IProgress.ProgressChanged` event.

```
public interface SigStat.Common.Helpers.IProgress
```

Properties

Type	Name	Summary
`Int32`	Progress	Gets the current progress in percentage.

Events

Type	Name	Summary
`EventHandler<Int32>`	ProgressChanged	Invoked whenever the `SigStat.Common.Helpers.IProgress.Progress` property is changed.

LogEntry

Represents a single entry of the log, consisting of a timestamp, a level, a sender and the message.

```
public class SigStat.Common.Helpers.LogEntry
```

Fields

Type	Name	Summary
`LogLevel`	Level	Log level of the entry.

Methods

Type	Name	Summary
`String`	ToString()	Format the contained data to string, divided by tab characters. Use this to display the entry in the console.

Logger

A easy-to-use class to log pipeline messages, complete with filtering levels and multi-thread support.

```
public class SigStat.Common.Helpers.Logger
```

Properties

Type	Name	Summary
`List<LogEntry>`	Entries	
`LogLevel`	FilteringLevel	
`IReadOnlyDictionary<String, Object>`	ObjectEntries	
`Boolean`	StoreEntries	Enable or disable the storing of log entries. This can come useful for filtering by certain type of entries.

Methods

Type	Name	Summary
`void`	Debug(`Object` sender, `String` message)	Enqueue a debug level log entry.
`void`	EnqueueEntry(`LogLevel` messageLevel, `Object` sender, `String` message)	Enqueue a new log entry with specified level. The entry is filtered through `SigStat.Common.Helpers.Logger.FilteringLevel` .
`void`	Error(`Object` sender, `String` message)	Enqueue an error level log entry.
`void`	Fatal(`Object` sender, `String` message)	Enqueue a fatal level log entry.
`void`	Info(`Object` sender, `String` message)	Enqueue an information level log entry.
`void`	Info(`Object` sender, `String` key, `Object` infoObject)	Enqueue an information level log entry.
`void`	Stop()	Stop accepting entries, flush the queue and stop the consuming thread.
`void`	Warn(`Object` sender, `String` message)	Enqueue a warning level log entry.

LogLevel

Represents the level of log. Lowest level: Off. Highest level: Debug.

```
public enum SigStat.Common.Helpers.LogLevel
    : Enum, IComparable, IFormattable, IConvertible
```

Enum

Value	Name	Summary

0	Off	Completely turn off logging.
1	Fatal	Represents a fatal error level log.
2	Error	Represents an error level log.
3	Warn	Represents a warning level log.
4	Info	Represents an information level log.
5	Debug	Represents a debug level log.

SigStat.Common.Loaders

DataSetLoader

Abstract loader class to inherit from. Implements ILogger.

```
public abstract class SigStat.Common.Loaders.DataSetLoader
    : IDataSetLoader, ILogger
```

Properties

Type	Name	Summary
Logger	Logger	

Methods

Type	Name	Summary
IEnumerable<Signer>	EnumerateSigners()	
IEnumerable<Signer>	EnumerateSigners(Predicate<Signer> signerFilter)	
void	Log(LogLevel level, String message)	

IDataSetLoader

Exposes a function to enable loading collections of SigStat.Common.Signer s. Base abstract class: SigStat.Common.Loaders.DataSetLoader .

```
public interface SigStat.Common.Loaders.IDataSetLoader
```

Methods

Type	Name	Summary
IEnumerable<Signer>	EnumerateSigners()	Loads the database and returns the collection of SigStat.Common.Signer s that match the .

IEnumerable<Signer>	EnumerateSigners(Predicate<Signer> signerFilter)	Loads the database and returns the collection of SigStat.Common.Signer s that match the .

ImageLoader

DataSetLoader for Image type databases. Similar format to Svc2004Loader, but finds png images.

```
public class SigStat.Common.Loaders.ImageLoader
    : DataSetLoader, IDataSetLoader, ILogger
```

Methods

Type	Name	Summary
IEnumerable<Signer>	EnumerateSigners(Predicate<Signer> signerFilter)	

Static Methods

Type	Name	Summary
void	LoadImage(Signature signature, String file)	Load one image.
Signature	LoadSignature(String file)	

ImageSaver

Get the SigStat.Common.Features.Image of a SigStat.Common.Signature and save it as png file.

```
public static class SigStat.Common.Loaders.ImageSaver
```

Static Methods

Type	Name	Summary
void	Save(Signature signature, String path)	Saves a png image file to the specified .

Svc2004

Set of features containing raw data loaded from SVC2004-format database.

```
public static class SigStat.Common.Loaders.Svc2004
```

Static Fields

Type	Name	Summary
FeatureDescriptor<List<Int32>>	Altitude	
FeatureDescriptor<List<Int32>>	Azimuth	

`FeatureDescriptor<List<Int32>>`	Button	
`FeatureDescriptor<List<Int32>>`	Pressure	
`FeatureDescriptor<List<Int32>>`	T	
`FeatureDescriptor<List<Int32>>`	X	
`FeatureDescriptor<List<Int32>>`	Y	

Svc2004Loader

Loads SVC2004-format database from .zip

```
public class SigStat.Common.Loaders.Svc2004Loader
    : DataSetLoader, IDataSetLoader, ILogger
```

Properties

Type	Name	Summary
`Predicate<Signer>`	SignerFilter	

Methods

Type	Name	Summary
`IEnumerable<Signer>`	EnumerateSigners(`Predicate<Signer>` signerFilter)	

Static Methods

Type	Name	Summary
`void`	LoadSignature(`Signature` signature, `String` path, `Boolean` standardFeatures)	Loads one signature from specified file path.
`void`	LoadSignature(`Signature` signature, `Stream` stream, `Boolean` standardFeatures)	Loads one signature from specified file path.

SigStat.Common

ArrayExtension

```
public static class SigStat.Common.ArrayExtension
```

Static Methods

Type	Name	Summary
`T[]`	Clone(this `T[]` array)	
`T[][]`	CreateNested(`Int32` length1, `Int32` length2)	

`T[]`	ForEach(this `T[]` array, `Action<T>` action)	Performs a given action on all items of the array and returns the original array.
`IEnumerable<T>`	GetColumn(this `T[,]` array, `Int32` colIndex)	
`T[,]`	GetPart(this `T[,]` source, `Int32` startIndex1, `Int32` startIndex2, `Int32` length1, `Int32` length2)	
`IEnumerable<T>`	GetRow(this `T[,]` array, `Int32` rowIndex)	
`Tuple<Int32, Int32>`	IndexOf(this `Int32[,]` array, `Int32` value)	
`Tuple<Int32, Int32>`	IndexOf(this `Double[,]` array, `Double` value)	
`Int32`	IndexOf(this `T[]` array, `T` value)	
`Int32`	Max(this `Int32[,]` array)	
`Byte`	Max(this `Byte[,]` array)	
`Double`	Max(this `Double[,]` array)	
`void`	SetColumn(this `T[,]` array, `Int32` x, `T` value)	
`void`	SetRow(this `T[,]` array, `Int32` y, `T` value)	
`T[,]`	SetValues(this `T[,]` array, `T` value)	
`T[]`	Shuffle(this `T[]` array)	

Baseline

`public class SigStat.Common.Baseline`

Properties

Type	Name	Summary
`PointF`	End	
`PointF`	Start	

Methods

Type	Name	Summary
`String`	ToString()	

Configuration

```
public class SigStat.Common.Configuration
```

Properties

Type	Name	Summary
`String`	DatabaseFolder	
`Lazy<Configuration>`	Default	

DataSet

```
public class SigStat.Common.DataSet
```

Properties

Type	Name	Summary
`List<Signer>`	Signers	

FeatureAttribute

```
public class SigStat.Common.FeatureAttribute
    : Attribute, _Attribute
```

Properties

Type	Name	Summary
`String`	FeatureKey	

FeatureDescriptor

Represents a feature with name and type.

```
public class SigStat.Common.FeatureDescriptor
```

Properties

Type	Name	Summary
`Type`	FeatureType	Gets or sets the type of the feature.
`Boolean`	IsCollection	Gets whether the type of the feature is List.
`String`	Key	Gets the unique key of the feature.
`String`	Name	Gets or sets a human readable name of the feature.

Methods

Type	Name	Summary
`String`	ToString()	Returns a string represenatation of the FeatureDescriptor

Static Fields

Type	Name	Summary
`Dictionary<String, FeatureDescriptor>`	descriptors	The static dictionary of all descriptors.

Static Methods

Type	Name	Summary
`FeatureDescriptor`	Get(`String` key)	Gets the `SigStat.Common.FeatureDescriptor` specified by . Throws `System.Collections.Generic.KeyNotFoundException` exception if there is no descriptor registered with the given key.
`FeatureDescriptor<T>`	Get(`String` key)	Gets the `SigStat.Common.FeatureDescriptor` specified by . Throws `System.Collections.Generic.KeyNotFoundException` exception if there is no descriptor registered with the given key.
`Boolean`	IsRegistered(`String` featureKey)	
`FeatureDescriptor`	Register(`String` featureKey, `Type` type)	

FeatureDescriptor<T>

Represents a feature with the type `type`

```
public class SigStat.Common.FeatureDescriptor<T>
    : FeatureDescriptor
```

Static Methods

Type	Name	Summary
`FeatureDescriptor<T>`	Get(`String` key)	Gets the `SigStat.Common.FeatureDescriptor` 1 `specified by ` . If the key is not registered yet, a new `SigStat.Common.FeatureDescriptor` 1` is automatically created with the given key and type.

Features

Standard set of features.

```
public static class SigStat.Common.Features
```

Static Fields

Type	Name	Summary
`IReadOnlyList<FeatureDescriptor>`	All	
`FeatureDescriptor<List<Double>>`	Altitude	
`FeatureDescriptor<List<Double>>`	Azimuth	
`FeatureDescriptor<RectangleF>`	Bounds	
`FeatureDescriptor<List<Boolean>>`	Button	
`FeatureDescriptor<Point>`	Cog	
`FeatureDescriptor<Int32>`	Dpi	
`FeatureDescriptor<Image<Rgba32>>`	Image	
`FeatureDescriptor<List<Double>>`	Pressure	
`FeatureDescriptor<List<Double>>`	T	
`FeatureDescriptor<Rectangle>`	TrimmedBounds	
`FeatureDescriptor<List<Double>>`	X	
`FeatureDescriptor<List<Double>>`	Y	

IClassification

Allows implementing a pipeline classifier item capable of logging, progress tracking and IO rewiring.

```
public interface SigStat.Common.IClassification
    : ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
`Double`	Pair(`Signature` signature1, `Signature` signature2)	Executes the classification by pairing the parameters. This function gets called by the pipeline.

IClassificationMethods

Extension methods for `SigStat.Common.IClassification` for convenient IO rewiring.

```
public static class SigStat.Common.IClassificationMethods
```

Static Methods

Type	Name	Summary

`IClassification`	Input(this `IClassification` caller, `FeatureDescriptor[]` inputFeatures)	Sets the InputFeatures in a convenient way.
`IClassification`	Output(this `IClassification` caller, `FeatureDescriptor[]` outputFeatures)	Sets the OutputFeatures in a convenient way.

ITransformation

Allows implementing a pipeline transform item capable of logging, progress tracking and IO rewiring.

```
public interface SigStat.Common.ITransformation
    : ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
`void`	Transform(`Signature` signature)	Executes the transform on the parameter. This function gets called by the pipeline.

ITransformationMethods

Extension methods for `SigStat.Common.ITransformation` for convenient IO rewiring.

```
public static class SigStat.Common.ITransformationMethods
```

Static Methods

Type	Name	Summary
`ITransformation`	Input(this `ITransformation` caller, `FeatureDescriptor[]` inputFeatures)	Sets the InputFeatures in a convenient way.
`ITransformation`	Output(this `ITransformation` caller, `FeatureDescriptor[]` outputFeatures)	Sets the OutputFeatures in a convenient way.

Loop

```
public class SigStat.Common.Loop
```

Properties

Type	Name	Summary
`RectangleF`	Bounds	
`PointF`	Center	
`List<PointF>`	Points	

Methods

Type	Name	Summary
`String`	ToString()	

MathHelper

`public static class SigStat.Common.MathHelper`

Static Methods

Type	Name	Summary
`Double`	Min(`Double` d1, `Double` d2, `Double` d3)	Returns the smallest of the three double parameters

Matrix

`public static class SigStat.Common.Matrix`

Static Methods

Type	Name	Summary
`E[,]`	Evaluate(`T[,]` matrix, `ItemEvaluator<E, T>` evaluator)	
`T[,]`	FromTableRows(`IEnumerable<DataRow>` rows, `Int32` ignoreColumns, `Int32` ignoreRows)	Egy DataRow gyüjteményt átalakít egy kétdimenziós tömbbé. Az átalakítás során ignoreColumns oszlopot és ignoreRows sort figyelmen kívül hagy.
`Point`	GetCog(`Double[,]` weightMartix)	
`IEnumerable<Point>`	GetNeighbourPixels(this `Point` p)	
`IEnumerable<Point>`	GetNeighbours(this `Point` p, `Point` start, `Int32` offset)	
`Double`	GetSum(`Double[,]` matrix, `Int32` x1, `Int32` y1, `Int32` x2, `Int32` y2)	
`Double`	GetSumCol(`Double[,]` matrix, `Int32` col)	
`Double`	GetSumRow(`Double[,]` matrix, `Int32` row)	
`Boolean[,]`	Invert(this `Boolean[,]` array)	returns a copy of the array with inverted values
`Byte[,]`	Neighbours(`T[,]` matrix, `T` emptyValue)	returns a same sized matrix with each item showing the neighbour

		count for the given position.
`T[]`	SetValues(this `T[]` array, `T` value)	
`T[]`	SetValues(this `T[]` array, `Func<T, T>` transformation)	

Origin

Represents our knowledge on the origin of a signature.

```
public enum SigStat.Common.Origin
    : Enum, IComparable, IFormattable, IConvertible
```

Enum

Value	Name	Summary
`0`	Unknown	Use this in practice before a signature is verified.
`1`	Genuine	The `SigStat.Common.Signature` 's origin is verified to be from `SigStat.Common.Signature.Signer`
`2`	Forged	The `SigStat.Common.Signature` is a forgery.

PipelineBase

TODO: Ideiglenes osztaly, C# 8.0 ban ezt atalakitani default implementacios interface be. IProgress, ILogger, IPipelineIO default implementacioja.

```
public abstract class SigStat.Common.PipelineBase
```

Properties

Type	Name	Summary
`List<FeatureDescriptor>`	InputFeatures	
`Logger`	Logger	
`List<FeatureDescriptor>`	OutputFeatures	
`Int32`	Progress	

Events

Type	Name	Summary
`EventHandler<Int32>`	ProgressChanged	

Methods

Type	Name	Summary

`void`	Log(`LogLevel` level, `String` message)	Enqueues a new log entry to be consumed by the attached `SigStat.Common.Helpers.Logger` . Use this when developing new pipeline items.
`void`	OnProgressChanged(`Int32` v)	Used to raise base class event in derived classes. See explanation: Event docs link.

Signature

Represents a signature as a collection of features, containing the data that flows in the pipeline.

```
public class SigStat.Common.Signature
```

Properties

Type	Name	Summary
`String`	ID	An identifier for the Signature. Keep it unique to be useful for logs.
`Object`	Item	Gets or sets the specified feature.
`Object`	Item	Gets or sets the specified feature.
`Origin`	Origin	Represents our knowledge on the origin of the signature. `SigStat.Common.Origin.Unknown` may be used in practice before it is verified.
`Signer`	Signer	A reference to the `SigStat.Common.Signer` who this signature belongs to. (The origin is not constrained to be genuine.)

Methods

Type	Name	Summary
`List<Double[]>`	GetAggregateFeature(`List<FeatureDescriptor>` fs)	Aggregate multiple featu Example: X, Y features - feature. Use this for exa algorithm input.
`T`	GetFeature(`String` featureKey)	Gets the specified featu
`T`	GetFeature(`FeatureDescriptor<T>` featureDescriptor)	Gets the specified featu
`T`	GetFeature(`FeatureDescriptor` featureDescriptor)	Gets the specified featu
`IEnumerable<FeatureDescriptor>`	GetFeatureDescriptors()	Gets a collection of `SigStat.Common.Featur` that are used in this sig
`Boolean`	HasFeature(`FeatureDescriptor` featureDescriptor)	Returns true if the signa the specified feature
`Boolean`	HasFeature(`String` featureKey)	Returns true if the signa the specified feature

`void`	SetFeature(`FeatureDescriptor` featureDescriptor, `T` feature)	Sets the specified featu
`void`	SetFeature(`String` featureKey, `T` feature)	Sets the specified featu
`String`	ToString()	Returns a string represe signature

Signer

Represents a person as a `SigStat.Common.Signer.ID` and a list of `SigStat.Common.Signer.Signatures` .

```
public class SigStat.Common.Signer
```

Properties

Type	Name	Summary
`String`	ID	An identifier for the Signer. Keep it unique to be useful for logs.
`List<Signature>`	Signatures	List of signatures that belong to the signer. (Their origin is not constrained to be genuine.)

Vector

```
public class SigStat.Common.Vector
```

Properties

Type	Name	Summary
`Double`	Angle	
`Double`	B	
`Rectangle`	BoundingRectangle	
`Rectangle`	Bounds	
`Point`	COG	
`Point`	End	
`Double`	Length	
`Point`	Location	
`Double`	M	
`Point`	Start	
`Int32`	Vx	
`Int32`	Vy	
`Int32`	X	

`Int32`	X2	
`Int32`	Y	
`Int32`	Y2	

Methods

Type	Name	Summary
`void`	Add(`Point` p)	
`Vector`	Clone()	
`Boolean`	Equals(`Object` obj)	Két vektor akkor egyenlõ, ha ugyanabból a pontból indulnak ki és ugyanabban az irányba mutatnak és hosszuk is megegyezik.
`IEnumerator<VectorPoint>`	GetEnumerator()	
`Int32`	GetHashCode()	
`Double`	GetLength()	
`Vector`	GetNormal()	Elofrgatja a vektort 90 fokkal a kezdõpontja körül az óramutató járásával megegyezõ irányba
`Vector`	GetNormal(`Double` length)	Elofrgatja a vektort 90 fokkal a kezdõpontja körül az óramutató járásával megegyezõ irányba
`String`	ToMatlabString()	
`String`	ToString()	

SigStat.Common.Model

ApproximateLimit

Used to approximate the classification limit in the training process.

```
public class SigStat.Common.Model.ApproximateLimit
```

Methods

Type	Name	Summary
`Double`	Calculate(`List<Signature>` sigs)	Calculate the limit by pairing each signature. Limit = AverageCost + StdDeviation.

BenchmarkResults

Contains the benchmark results of every `SigStat.Common.Signer` and the summarized final results.

```
public struct SigStat.Common.Model.BenchmarkResults
```

Fields

Type	Name	Summary
`Result`	FinalResult	Summarized, final result of the benchmark execution.
`List<Result>`	SignerResults	List that contains the `SigStat.Common.Model.Result` s for each `SigStat.Common.Signer`

Result

Contains the benchmark results of a single `SigStat.Common.Signer`

```
public class SigStat.Common.Model.Result
```

Fields

Type	Name	Summary
`Double`	Aer	Average Error Rate
`Double`	Far	False Acceptance Rate
`Double`	Frr	False Rejection Rate
`String`	Signer	Identifier of the `SigStat.Common.Model.Result.Signer`

Sampler

Takes samples from a set of `SigStat.Common.Signature` s by given sampling strategies. Use this to fine-tune the `SigStat.Common.Model.VerifierBenchmark`

```
public class SigStat.Common.Model.Sampler
```

Methods

Type	Name	Summary
`void`	Init(`Signer` s)	Initialize the Sampler with a Signer's Signatures.
`void`	Init(`List<Signature>` s)	Initialize the Sampler with a Signer's Signatures.
`List<Signature>`	SampleForgeryTests()	Samples a batch of forged signatures to test on.
`List<Signature>`	SampleGenuineTests()	Samples a batch of genuine signatures to test on.
`List<Signature>`	SampleReferences()	Samples a batch of genuine reference signatures to train on.

Static Properties

Type	Name	Summary
`Sampler`	BasicSampler	Default sampler for SVC2004 database. 10 references, 10 genuine tests, 10 forged tests

ThresholdResult

```
public class SigStat.Common.Model.ThresholdResult
    : Result
```

Verifier

Uses pipelines to transform, train on, and classify `SigStat.Common.Signature` objects.

```
public class SigStat.Common.Model.Verifier
    : ILogger, IProgress
```

Properties

Type	Name	Summary
`IClassification`	ClassifierPipeline	Gets or sets the classifier pipeline. Hands over the Logger object.
`Logger`	Logger	Gets or sets the attached `SigStat.Common.Helpers.Logger` object used to log messages. Hands it over to the pipelines.
`Int32`	Progress	
`ITransformation`	TransformPipeline	Gets or sets the transform pipeline. Hands over the Logger object.

Events

Type	Name	Summary
`EventHandler<Int32>`	ProgressChanged	

Methods

Type	Name	Summary
`void`	Log(`LogLevel` level, `String` message)	Enqueues a new log entry to be consumed by the attached `SigStat.Common.Helpers.Logger` . Use this when developing new pipeline items.
`void`	LoggerChanged(`Logger` oldLogger, `Logger` newLogger)	
`Boolean`	Test(`Signature` sig)	Verifies the genuinity of .
`void`	Train(`Signer` signer)	Trains the verifier with `SigStat.Common.Signer.Signatures` having `SigStat.Common.Origin.Genuine` property.

void	Train(List<Signature> sigs)	Trains the verifier with SigStat.Common.Signer.Signatures having SigStat.Common.Origin.Genuine property.

Static Properties

Type	Name	Summary
Verifier	BasicVerifier	Basic SigStat.Common.Model.Verifier model with DTW classification of tangent features.

VerifierBenchmark

Benchmarking class to test error rates of a SigStat.Common.Model.Verifier

```
public class SigStat.Common.Model.VerifierBenchmark
    : ILogger, IProgress
```

Properties

Type	Name	Summary
IDataSetLoader	Loader	
Logger	Logger	Gets or sets the attached SigStat.Common.Helpers.Logger object used to log messages. Hands it over to the verifier.
Int32	Progress	
Sampler	Sampler	
Verifier	Verifier	Gets or sets the SigStat.Common.Model.Verifier to be benchmarked.

Events

Type	Name	Summary
EventHandler<Int32>	ProgressChanged	

Methods

Type	Name	Summary
BenchmarkResults	Execute()	Synchronously execute the benchmarking process.
BenchmarkResults	ExecuteParallel()	Parallel execute the benchmarking process.
void	Log(LogLevel level, String message)	

Static Methods

Type	Name	Summary

`Task<Int32>`	ExecuteAsync()	Asynchronously execute the benchmarking process.

SigStat.Common.Pipeline

IClassificationModel

Analyzes signatures based on their similiarity to the trained model

```
public interface SigStat.Common.Pipeline.IClassificationModel
```

Methods

Type	Name	Summary
`Double`	Test(`Signature` signature)	Returns a double value in the range [0..1], representing the probability of the given signature belonging to the trained model. 0: non-match0.5: inconclusive1: match

IClassifier

Trains classification models based on reference signatures

```
public interface SigStat.Common.Pipeline.IClassifier
```

Methods

Type	Name	Summary
`IClassificationModel`	Train(`List<Signature>` signatures)	Trains a model based on the signatures and returns the trained model

IPipelineIO

Gives ability to get or set (rewire) a pipeline item's default input and output features.

```
public interface SigStat.Common.Pipeline.IPipelineIO
```

Properties

Type	Name	Summary
`List<FeatureDescriptor>`	InputFeatures	List of features to be used as input.
`List<FeatureDescriptor>`	OutputFeatures	List of features to be used as output.

ParallelTransformPipeline

Runs pipeline items in parallel. Default Pipeline Output: Range of all the Item outputs.

```
public class SigStat.Common.Pipeline.ParallelTransformPipeline
    : PipelineBase, IEnumerable, ITransformation, ILogger, IProgress, IPipelineIO
```

Properties

Type	Name	Summary
`List<ITransformation>`	Items	
`Logger`	Logger	Passes Logger to child items as well.
`Int32`	Progress	Gets the minimum progess of all the child items.

Methods

Type	Name	Summary
`void`	Add(`ITransformation` newItem)	Add new transform to the list. Pass `SigStat.Common.Pipeline.ParallelTransformPipeline.Logger` and set up Progress event.
`IEnumerator`	GetEnumerator()	
`void`	Transform(`Signature` signature)	Executes transform `SigStat.Common.Pipeline.ParallelTransformPipeline.Items` parallel. Passes input features for each. Output is a range of all the Item outputs.

SequentialTransformPipeline

Runs pipeline items in a sequence. Default Pipeline Output: Output of the last Item in the sequence.

```
public class SigStat.Common.Pipeline.SequentialTransformPipeline
    : PipelineBase, IEnumerable, ITransformation, ILogger, IProgress, IPipelineIO
```

Properties

Type	Name	Summary
`List<ITransformation>`	Items	
`Logger`	Logger	Passes Logger to child items as well.

Methods

Type	Name	Summary
`void`	Add(`ITransformation` newItem)	Add new transform to the list. Pass `SigStat.Common.Pipeline.SequentialTransformPipeline.Logger` and set up Progress event.
`IEnumerator`	GetEnumerator()	
		Executes transform

void	Transform(Signature signature)	SigStat.Common.Pipeline.SequentialTransformPipeline.Items in sequence. Passes input features for each. Output is the output of the last Item in the sequence.

SigStat.Common.PipelineItems.Classifiers

DTWClassifier

Classifies Signatures with the SigStat.Common.Algorithms.Dtw algorithm.

```
public class SigStat.Common.PipelineItems.Classifiers.DTWClassifier
    : PipelineBase, IClassification, ILogger, IProgress, IPipelineIO, IEnumerable
```

Methods

Type	Name	Summary
void	Add(FeatureDescriptor f)	
IEnumerator	GetEnumerator()	
Double	Pair(Signature signature1, Signature signature2)	Aggregates the input features and executes the SigStat.Common.Algorithms.Dtw algorithm.

WeightedClassifier

Classifies Signatures by weighing other Classifier results.

```
public class SigStat.Common.PipelineItems.Classifiers.WeightedClassifier
    : PipelineBase, IEnumerable, IClassification, ILogger, IProgress, IPipelineIO
```

Fields

Type	Name	Summary
List<ValueTuple<IClassification, Double>>	Items	List of classifiers and belonging weights.

Properties

Type	Name	Summary
Logger	Logger	Gets or sets the Logger. Passes it to child Items as well.

Methods

Type	Name	Summary
void	Add(ValueTuple<IClassification, Double> newItem)	Add a new classifier with given weight to the list of items.
IEnumerator	GetEnumerator()	

`Double`	Pair(`Signature` signature1, `Signature` signature2)	Execute each classifier in the list and weigh returned costs.

SigStat.Common.PipelineItems.Markers

LogMarker

Logs the Pipeline Input. Useful for logging TimeMarker results. Default Pipeline Output: -

```
public class SigStat.Common.PipelineItems.Markers.LogMarker
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
`void`	Transform(`Signature` signature)	

TimeMarkerStart

Starts a timer to measure completion time of following transforms. Default Pipeline Output: (`System.DateTime`) DefaultTimer

```
public class SigStat.Common.PipelineItems.Markers.TimeMarkerStart
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
`void`	Transform(`Signature` signature)	

TimeMarkerStop

Stops a timer to measure completion time of previous transforms. Default Pipeline Output: (`System.DateTime`) DefaultTimer

```
public class SigStat.Common.PipelineItems.Markers.TimeMarkerStop
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
`void`	Transform(`Signature` signature)	

SigStat.Common.Transforms

AddConst

Adds a constant value to a feature. Works with collection features too. Default Pipeline Output: Pipeline Input

```
public class SigStat.Common.Transforms.AddConst
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
`void`	Transform(`Signature` signature)	

AddVector

Adds a vector feature's elements to other features. Default Pipeline Output: Pipeline Input

```
public class SigStat.Common.Transforms.AddVector
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
`void`	Transform(`Signature` signature)	

ApproximateOnlineFeatures

init Pressure, Altitude, Azimuth features with default values. Default Pipeline Output: Features.Pressure, Features.Altitude, Features.Azimuth

```
public class SigStat.Common.Transforms.ApproximateOnlineFeatures
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
`void`	Transform(`Signature` signature)	

Binarization

Generates a binary raster version of the input image with the iterative threshold method. Pipeline Input type: Image{Rgba32}Default Pipeline Output: (bool[,]) Binarized

```
public class SigStat.Common.Transforms.Binarization
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
`void`	Transform(`Signature` signature)	

BinaryRasterizer

Converts standard features to a binary raster. Default Pipeline Input: Standard `SigStat.Common.Features` Default Pipeline Output: (bool[,]) Binarized

```
public class SigStat.Common.Transforms.BinaryRasterizer
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
`void`	Transform(`Signature` signature)	

CentroidExtraction

Extracts the Centroid (aka. Center Of Gravity) of the input features. Default Pipeline Output: (List{double}) Centroid.

```
public class SigStat.Common.Transforms.CentroidExtraction
    : PipelineBase, IEnumerable, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
`void`	Add(`FeatureDescriptor<List<Double>>` newitem)	
`IEnumerator`	GetEnumerator()	
`void`	Transform(`Signature` signature)	

CentroidTranslate

Sequential pipeline to translate X and Y `SigStat.Common.Features` to Centroid. The following Transforms are called: `SigStat.Common.Transforms.CentroidExtraction` , `SigStat.Common.Transforms.Multiply` (-1), `SigStat.Common.Transforms.Translate` Default Pipeline Input: `SigStat.Common.Features.X` , `SigStat.Common.Features.Y` Default Pipeline Output: (List{double}) Centroid

```
public class SigStat.Common.Transforms.CentroidTranslate
    : SequentialTransformPipeline, IEnumerable, ITransformation, ILogger, IProgress, IPipelineIO
```

ComponentExtraction

Extracts unsorted components by tracing through the binary Skeleton raster. Default Pipeline Input: (bool[,]) Skeleton, (List{Point}) EndPoints, (List{Point}) CrossingPointsDefault Pipeline Output: (List{List{PointF}}) Components

```
public class SigStat.Common.Transforms.ComponentExtraction
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
`void`	Transform(`Signature` signature)	

ComponentSorter

Sorts Component order by comparing each starting X value, and finding nearest components. Default Pipeline Input: (bool[,]) ComponentsDefault Pipeline Output: (bool[,]) Components

```
public class SigStat.Common.Transforms.ComponentSorter
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
`void`	Transform(`Signature` signature)	

ComponentsToFeatures

Extracts standard `SigStat.Common.Features` from sorted Components. Default Pipeline Input: (List{List{PointF}}) ComponentsDefault Pipeline Output: X, Y, Button `SigStat.Common.Features`

```
public class SigStat.Common.Transforms.ComponentsToFeatures
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
`void`	Transform(`Signature` signature)	

EndpointExtraction

Extracts EndPoints and CrossingPoints from Skeleton. Default Pipeline Input: (bool[,]) SkeletonDefault Pipeline Output: (List{Point}) EndPoints, (List{Point}) CrossingPoints

```
public class SigStat.Common.Transforms.EndpointExtraction
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
`void`	Transform(`Signature` signature)	

Extrema

Extracts minimum and maximum values of given feature. Default Pipeline Output: (List{double}) Min, (List{double}) Max

```
public class SigStat.Common.Transforms.Extrema
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
`void`	Transform(`Signature` signature)	

HSCPThinning

Iteratively thins the input binary raster with the `SigStat.Common.Algorithms.HSCPThinningStep` algorithm. Pipeline Input type: bool[,]Default Pipeline Output: (bool[,]) HSCPThinningResult

```
public class SigStat.Common.Transforms.HSCPThinning
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
`void`	Transform(`Signature` signature)	

ImageGenerator

Generates an image feature out of a binary raster. Optionally, saves the image to a png file. Useful for debugging pipeline steps. Pipeline Input type: bool[,]Default Pipeline Output: (bool[,]) Input, (Image{Rgba32}) InputImage

```
public class SigStat.Common.Transforms.ImageGenerator
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
`void`	Transform(`Signature` signature)	

Map

Maps values of a feature to a specified range. Pipeline Input type: List{double}Default Pipeline Output: (List{double}) MapResult

```
public class SigStat.Common.Transforms.Map
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
void	Transform(Signature signature)	

Multiply

Multiplies the values of a feature with a given constant. Pipeline Input type: List{double}Default Pipeline Output: (List{double}) Input

```
public class SigStat.Common.Transforms.Multiply
    : PipelineBase, IEnumerable, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
void	Add(FeatureDescriptor newItem)	
IEnumerator	GetEnumerator()	
void	Transform(Signature signature)	

Normalize

Maps values of a feature to 0.0 - 1.0 range. Pipeline Input type: List{double}Default Pipeline Output: (List{double}) NormalizationResult

```
public class SigStat.Common.Transforms.Normalize
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
void	Transform(Signature signature)	

OnePixelThinning

Iteratively thins the input binary raster with the SigStat.Common.Algorithms.OnePixelThinningStep algorithm. Pipeline Input type: bool[,]Default Pipeline Output: (bool[,]) OnePixelThinningResult

```
public class SigStat.Common.Transforms.OnePixelThinning
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
void	Transform(Signature signature)	

PrepareForThinning

```
public class SigStat.Common.Transforms.PrepareForThinning
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
void	Transform(Signature signature)	

RealisticImageGenerator

Generates a realistic looking image of the Signature based on standard features. Uses blue ink and white paper. It does NOT save the image to file. Default Pipeline Input: X, Y, Button, Pressure, Azimuth, Altitude SigStat.Common.Features Default Pipeline Output: SigStat.Common.Features.Image

```
public class SigStat.Common.Transforms.RealisticImageGenerator
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
void	Transform(Signature signature)	

Resize

Resizes the image to a specified width and height

```
public class SigStat.Common.Transforms.Resize
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Properties

Type	Name	Summary
Nullable<Int32>	Height	The new height. Leave it as null, if you do not want to explicitly specify a given height
Func<Image<Rgba32>, Size>	ResizeFunction	Set a resize function if you want to dynamically calculate the new width and height of the image
Nullable<Int32>	Width	The new width. Leave it as null, if you do not want to explicitly specify a given width

Methods

Type	Name	Summary
void	Transform(Signature signature)	

TangentExtraction

Extracts tangent values of the standard X, Y `SigStat.Common.Features` Default Pipeline Input: X, Y `SigStat.Common.Features` Default Pipeline Output: (List{double}) Tangent

```
public class SigStat.Common.Transforms.TangentExtraction
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
`void`	Transform(`Signature` signature)	

TimeReset

Sequential pipeline to reset time values to begin at 0. The following Transforms are called: Extrema, Multiply, AddVector. Default Pipeline Input: `SigStat.Common.Features.T` Default Pipeline Output: `SigStat.Common.Features.T`

```
public class SigStat.Common.Transforms.TimeReset
    : SequentialTransformPipeline, IEnumerable, ITransformation, ILogger, IProgress, IPipelineIO
```

Translate

Sequential pipeline to translate X and Y `SigStat.Common.Features` by specified vector (constant or feature). The following Transforms are called: `SigStat.Common.Transforms.AddConst` twice, or `SigStat.Common.Transforms.AddVector` . Default Pipeline Input: `SigStat.Common.Features.X` , `SigStat.Common.Features.Y` Default Pipeline Output: `SigStat.Common.Features.X` , `SigStat.Common.Features.Y`

```
public class SigStat.Common.Transforms.Translate
    : SequentialTransformPipeline, IEnumerable, ITransformation, ILogger, IProgress, IPipelineIO
```

Trim

Trims unnecessary empty space from a binary raster. Pipeline Input type: bool[,]Default Pipeline Output: (bool[,]) Trimmed

```
public class SigStat.Common.Transforms.Trim
    : PipelineBase, ITransformation, ILogger, IProgress, IPipelineIO
```

Methods

Type	Name	Summary
`void`	Transform(`Signature` signature)	