Transformers and Estimators

Lesson Objectives

- After completing this lesson, you should be able to:
  - Understand, create, and use a Transformer
  - Understand, create, and use an Estimator
  - Set parameters of Transformers and Estimators
  - Create a feature Vector with `VectorAssembler`

Transformers

- Algorithm which can transform one DataFrame into another DataFrame
- Abstraction that includes feature transformers and learned models.
- Implements a method transform(), which converts one DataFrame into another, generally by appending one or more columns
- Input and output columns set with setInputCol and setOutputCol methods
- Examples
  - –read one or more columns and map them into a new column of feature vectors
  - –read a column containing feature vectors and make a prediction for each vector

General Purpose Transformers

Transformer	Description	scikit-learn
`Binarizer`	Threshold numerical feature to binary	`Binarizer`
`Bucketizer`	Bucket numerical features into ranges	
`ElementwiseProduct`	Scale each feature/column separately	
`Normalizer`	Scale each row to unit norm	`Normalizer`
`OneHotEncoder`	Encode k-category feature as binary features	`OneHotEncoder`
`PolynomialExpansion`	Create higher-order features	`PolynomialFeatures`
`StandardScaler`	Scale features to 0 mean and/or unit variance	`StandardScaler`
`StringIndexer`	Convert String feature to 0-based indices	`LabelEncoder`
`VectorAssembler`	Concatenate feature vectors	`FeatureUnion`
`VectorIndexer`	Identify categorical features, and index	

Transformers for Natural Language Processing

Transformer	Description	scikit-learn
`HashingTF`	Hash text/data to vector. Scale by term frequency	`FeatureHasher`
`IDF`	Scale features by inverse document frequency	`TfidfTransformer`
`RegexTokenizer`	Tokenize text using regular expressions	(part of text methods)
`Tokenizer`	Tokenize text on whitespace	(part of text methods)
`Word2Vec`	Learn vector representation of words	

Transformer Example

```
from pyspark.ml.feature import Tokenizer, RegexTokenizer
```

```
sentenceDataFrame = sqlc.createDataFrame([(0, "Hi I heard about Spark"),
                                          (1, "I wish Java could use case classes"),
                                          (2, "Logistic,regression,models,are,neat")]) \
                    .toDF("label", "sentence")
```

```
tokenizer = Tokenizer().setInputCol("sentence").setOutputCol("words")
tokenized = tokenizer.transform(sentenceDataFrame)
```

```
tokenized.toPandas()
```

	label	sentence	words
0	0	Hi I heard about Spark	[hi, i, heard, about, spark]
1	1	I wish Java could use case classes	[i, wish, java, could, use, case, classes]
2	2	Logistic,regression,models,are,neat	[logistic,regression,models,are,neat]

Estimators

- Algorithm which can be fit on a DataFrame to produce a Transformer
- Abstracts the concept of a learning algorithm or any algorithm that fits or trains on data
- Implements a method fit(), which accepts a DataFrame and produces a Model, which is a Transformer
- Example: LogisticRegression
    - It is a learning algorithm and therefore an Estimator
    - By calling the method fit() to train the logistic regression, a Model is returned

A Simple Example of an Estimator

```
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.linalg import Vectors
```

```
training = sqlc.createDataFrame([(1.0, Vectors.dense(0.0, 1.1, 0.1)),
                                 (0.0, Vectors.dense(2.0, 1.0, -1.0)),
                                 (0.0, Vectors.dense(2.0, 1.3, 1.0)),
                                 (1.0, Vectors.dense(0.0, 1.2, -0.5))]) \
            .toDF("label", "features")
```

```
lr = LogisticRegression()
```

```
lr.setMaxIter(10).setRegParam(0.01)
```

```
LogisticRegression_441ba6efd2c160f50102
```

```
model1 = lr.fit(training)
model1.coefficients
```

```
DenseVector([-3.1009, 2.6082, -0.3802])
```

A Simple Example of an Estimator

```
model1.transform(training).toPandas()
```

	label	features	rawPrediction	probability	prediction
0	1.0	[0.0, 1.1, 0.1]	[-2.89919489464, 2.89919489464]	[0.052193376663, 0.947806623337]	1.0
1	0.0	[2.0, 1.0, -1.0]	[3.14530074644, -3.14530074644]	[0.95872315829, 0.04127684171]	0.0
2	0.0	[2.0, 1.3, 1.0]	[3.12319457003, -3.12319457003]	[0.95783942353, 0.0421605764704]	0.0
3	1.0	[0.0, 1.2, -0.5]	[-3.388123842, 3.388123842]	[0.0326686926626, 0.967331307337]	1.0

Parameters

- Transformers and Estimators use a uniform API for specifying parameters
- A param map is a dict of {parameter: value} pairs
- Parameters are specific to a given instance
- There are two main ways to pass parameters to an algorithm:
  - Setting parameters for an instance using an appropriate method, for instance, setMaxIter(10)
  - Passing a dict to fit() or transform()
    - In this case, the parameter MaxIter is being specified to two different instances of models, lr1 and lr2

A Simple Example of Parameter Setting

```
model1 = lr.fit(training, {'maxIter': 10, 'regParam': 0.01})
model1.coefficients
```

```
DenseVector([-3.1009, 2.6082, -0.3802])
```

```
model1.transform(training).toPandas()
```

	label	features	rawPrediction	probability	prediction
0	1.0	[0.0, 1.1, 0.1]	[-2.89919489464, 2.89919489464]	[0.052193376663, 0.947806623337]	1.0
1	0.0	[2.0, 1.0, -1.0]	[3.14530074644, -3.14530074644]	[0.95872315829, 0.04127684171]	0.0
2	0.0	[2.0, 1.3, 1.0]	[3.12319457003, -3.12319457003]	[0.95783942353, 0.0421605764704]	0.0
3	1.0	[0.0, 1.2, -0.5]	[-3.388123842, 3.388123842]	[0.0326686926626, 0.967331307337]	1.0

Vector Assembler

- Transformer that combines a given list of columns into a single vector column
- Useful for combining raw features and features generated by other transformers into a single feature vector
- Accepts the following input column types:
  - all numeric types
  - boolean
  - vector

An Example of VectorAssembler

```
dfRandom = sqlc.range(0, 10).select("id") \
            .withColumn("uniform", rand(10)) \
            .withColumn("normal1", randn(10)) \
            .withColumn("normal2", randn(11))
```

```
from pyspark.ml.feature import VectorAssembler
assembler = VectorAssembler(inputCols = ["uniform","normal1","normal2"], outputCol = "features")
```

```
dfVec = assembler.transform(dfRandom)
```

An Example of VectorAssembler

```
dfVec.select("id","features").toPandas()
```

	id	features
0	0	[0.41371264721, -0.587748239674, -0.256535324205]
1	1	[0.198291963821, -0.256535324205, -0.506853671...
2	2	[0.120307152585, -0.506853671746, -0.141369919...
3	3	[0.442929185213, -0.141369919356, -0.726587521...
4	4	[0.889878425389, 0.965766508876, 0.891697335754]
5	5	[0.273107306848, -0.726587521995, -1.19853855262]
6	6	[0.870793547001, -1.19853855262, -0.117110926001]
7	7	[0.271493317932, -0.117110926001, 0.304945613282]
8	8	[0.603714357844, 0.304945613282, 0.0393394905131]
9	9	[0.143566883898, -1.04800065723, -0.963546696012]

Lesson Summary

- Having completed this lesson, you should be able to:
  - Understand, create, and use a Transformer
  - Understand, create, and use an Estimator
  - Set parameters of Transformers and Estimators
  - Create a feature Vector with VectorAssembler