Data Normalization

Lesson Objectives

- After completing this lesson, you should be able to:
  - Normalize a dataset to have unit p-norm
  - Normalize a dataset to have unit standard deviation and zero mean
  - Normalize a dataset to have given minimum and maximum values

Normalizer

- A Transformer which transforms a dataset of Vector rows, normalizing each Vector to have unit norm
- Takes a parameter P, which specifies the p-norm used for normalization (p=2 by default)
- Standardize input data and improve the behavior of learning algorithms

A Simple Normalizer

```
from pyspark.ml.feature import Normalizer
```

```
scaler1 = Normalizer().setInputCol("features").setOutputCol("scaledFeat").setP(1.0)
```

```
scaler1.transform(dfVec.select("id","features")).show(5)
```

```
+---+--------------------+--------------------+
| id|            features|          scaledFeat|
+---+--------------------+--------------------+
|  0|[0.41371264720975...|[0.32886636983701...|
|  1|[0.19829196382083...|[0.20619308493718...|
|  2|[0.12030715258495...|[0.15654175655718...|
|  3|[0.44292918521277...|[0.33788519635286...|
|  4|[0.88987842538862...|[0.32390519197407...|
+---+--------------------+--------------------+
only showing top 5 rows
```

Standard Scaler

- A Model which can be fit on a dataset to produce a StandardScalerModel
- A Transformer which transforms a dataset of Vector rows, normalizing each feature to have unit standard deviation and/or zero mean
- Takes two parameters:
  - withStd: scales the data to unit standard deviation (default: true)
  - withMean: centers the data with mean before scaling (default: false)
- It builds a dense output, sparse inputs will raise an exception
- If the standard deviation of a feature is zero, it returns 0.0 in the Vector for that feature

A Simple Standard Scaler

```
from pyspark.ml.feature import StandardScaler
```

```
scaler2 = StandardScaler().setInputCol("features").setOutputCol("scaledFeat").setWithStd(True).setWithMean(True)
```

```
scaler2Model = scaler2.fit(dfVec.select("id","features"))
```

```
scaler2Model.transform(dfVec.select("id","features")).toPandas()[:5]
```

	id	features	scaledFeat
0	0	[0.41371264721, -0.587748239674, -0.256535324205]	[-0.0321173330608, -0.400475290492, 0.01770640...
1	1	[0.198291963821, -0.256535324205, -0.506853671...	[-0.795199539636, 0.116559162465, -0.38815037517]
2	2	[0.120307152585, -0.506853671746, -0.141369919...	[-1.07144423862, -0.274196160897, 0.204431268542]
3	3	[0.442929185213, -0.141369919356, -0.726587521...	[0.0713760731298, 0.296336216182, -0.744418595...
4	4	[0.889878425389, 0.965766508876, 0.891697335754]	[1.65459922076, 2.02461325728, 1.87940775681]

MinMax Scaler

- A Model which can be fit on a dataset to produce a MinMaxScalerModel
- A Transformer which transforms a dataset of Vector rows, rescaling each feature to a specific range (often [0, 1])
- Takes two parameters:
  - min: lower bound after transformation, shared by all features (default: 0.0)
  - max: upper bound after transformation, shared by all features (default: 1.0)
- Since zero values are likely to be transformed to non-zero values, sparse inputs may result in dense outputs

A Simple MinMax Scaler

```
from pyspark.ml.feature import MinMaxScaler
```

```
scaler3 = MinMaxScaler().setInputCol("features").setOutputCol("scaledFeat").setMin(-1.0).setMax(1.0)
```

```
scaler3Model = scaler3.fit(dfVec.select("id","features"))
```

```
scaler3Model.transform(dfVec.select("id","features")).toPandas()[:5]
```

	id	features	scaledFeat
0	0	[0.41371264721, -0.587748239674, -0.256535324205]	[-0.237483245559, -0.435578353707, -0.09866323...
1	1	[0.198291963821, -0.256535324205, -0.506853671...	[-0.797329203956, -0.12950974872, -0.338175289...
2	2	[0.120307152585, -0.506853671746, -0.141369919...	[-1.0, -0.360824965779, 0.0115304584941]
3	3	[0.442929185213, -0.141369919356, -0.726587521...	[-0.161553857247, -0.0230872236347, -0.5484231...
4	4	[0.889878425389, 0.965766508876, 0.891697335754]	[1.0, 1.0, 1.0]

Lesson Summary

- Having completed this lesson, you should be able to:
  - Normalize a dataset to have unit p-norm
  - Normalize a dataset to have unit standard deviation and zero mean
  - Normalize a dataset to have given minimum and maximum values