Data Normalization

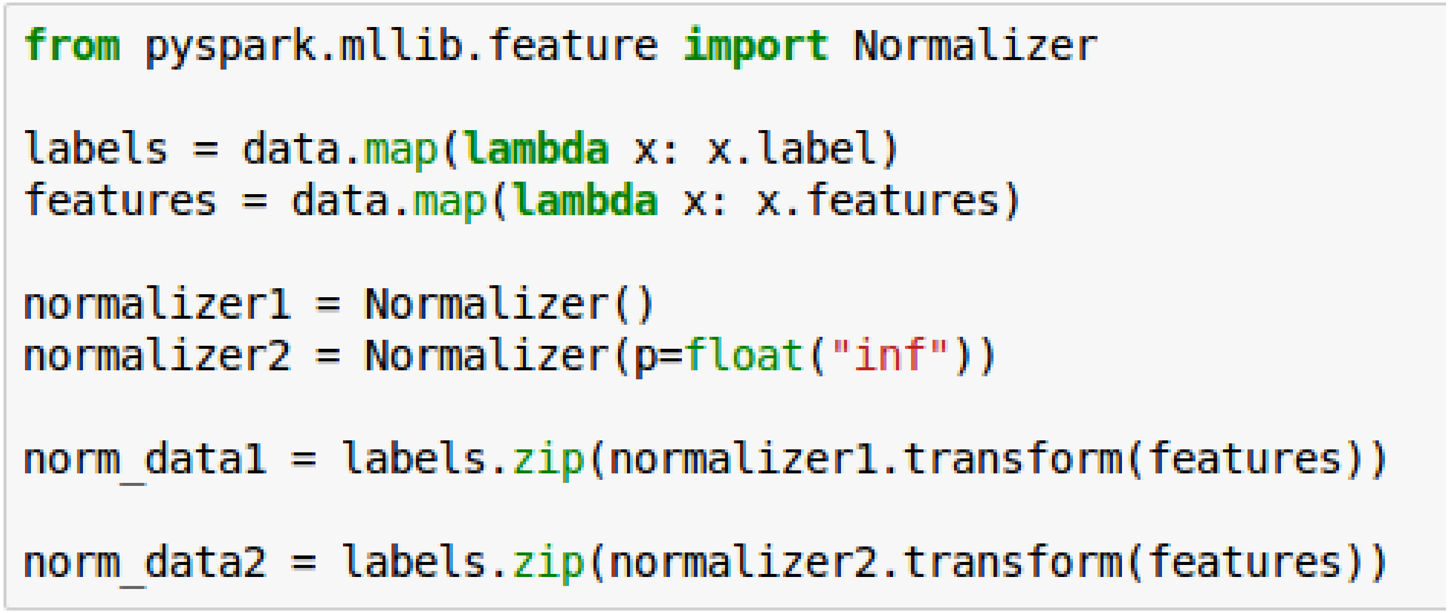
Lesson Objectives

- 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

Normalizer

- Transforms an RDD of Labeled Points, normalizing each feature 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



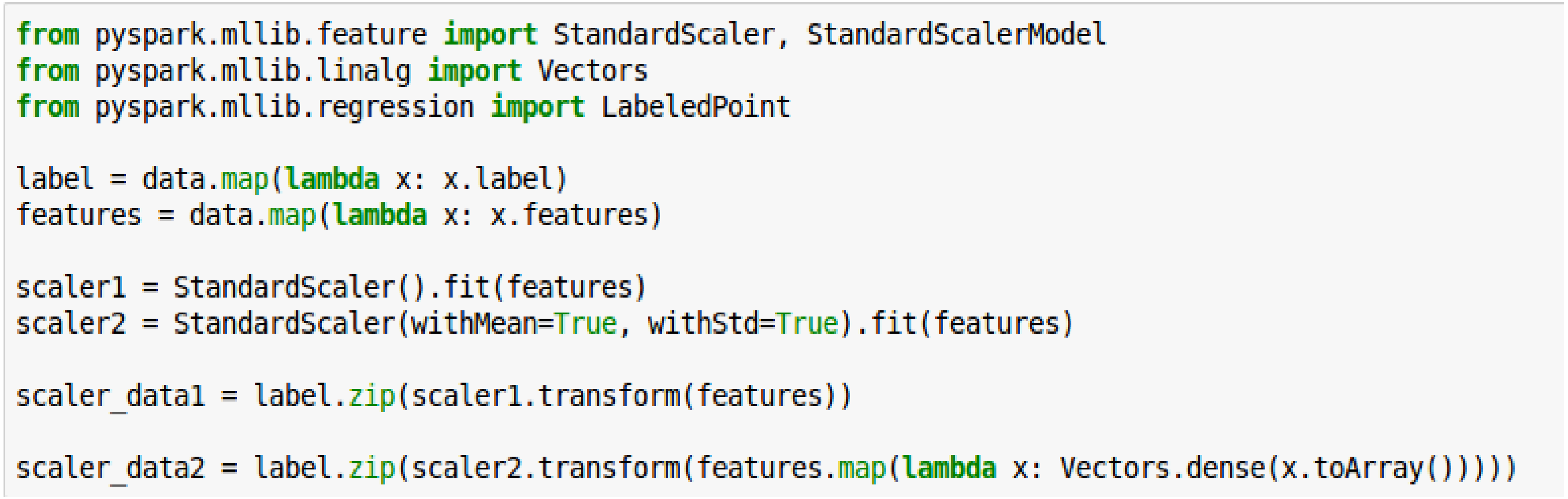
```python
from pyspark.mllib.feature import Normalizer

labels = data.map(lambda x: x.label)
features = data.map(lambda x: x.features)

normalizer1 = Normalizer()
normalizer2 = Normalizer(p=float("inf"))

norm_data1 = labels.zip(normalizer1.transform(features))

norm_data2 = labels.zip(normalizer2.transform(features))
```

Standard Scaler

Transforms an RDD of LabeledPoints, 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



```python
from pyspark.mllib.feature import StandardScaler, StandardScalerModel
from pyspark.mllib.linalg import Vectors
from pyspark.mllib.regression import LabeledPoint

label = data.map(lambda x: x.label)
features = data.map(lambda x: x.features)

scaler1 = StandardScaler().fit(features)
scaler2 = StandardScaler(withMean=True, withStd=True).fit(features)

scaler_data1 = label.zip(scaler1.transform(features))

scaler_data2 = label.zip(scaler2.transform(features.map(lambda x: Vectors.dense(x.toArray()))))
```

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