

Welcome to Week 3

This document provides a running example of completing the Week 3 assignment :

- You can then copy paste the commands below in pySpark.
- To open pySpark, type pyspark and hit enter

Finally, make sure that your working directory contains the data file(s) (.csv) for the fastest support.

We recommend working in your home directory (type `cd ~` in your terminal). Please run any scripts using your terminal for proper settings

To run these commands in Cloudera VM: first run the setup script:
`setupWeek3.sh`

```
In [2]: from pyspark.sql import SQLContext
        from pyspark.ml.clustering import KMeans
        from pyspark.ml.feature import VectorAssembler
        from pyspark.ml.feature import StandardScaler
```

Import Data

First let us read the contents of the file `w3_clustering.csv`. The following commands read in the CSV file in a table format.

If you want to run this command on your machine: Note that you must change the path in the following command to `'w3_clustering.csv'` to point to the location on your local machine.

```
In [4]: sqlContext = SQLContext(sc)
data = sqlContext.read.load('file:///home/cloudera/coursera/courseraData
Simulation/capstone/readings/w3_clustering.csv',
                             format='com.databricks.spark.csv',
                             header='true',inferSchema='true')
```

Let us display the number of lines in the data:

```
In [5]: data.count()
```

```
Out[5]: 709
```

What does the data contain ?

```
In [6]: data
```

```
Out[6]: DataFrame[totalAdClicks: int, revenue: double]
```

Statistics about the data:

```
In [7]: data.describe().toPandas().transpose()
```

```
Out[7]:
```

	0	1	2	3	4
summary	count	mean	stddev	min	max
totalAdClicks	709	32.208744710860366	16.38412081704256	1	73
revenue	709	44.64880112834979	44.9445287652853	1.0	278.0

Drop all rows with NULL or NaN values

```
In [8]: data = data.na.drop()
```

Let us look at the column names:

```
In [9]: data.columns
```

```
Out[9]: ['totalAdClicks', 'revenue']
```

Use VectorAssembler to gather all the features:

```
In [10]: featuresUsed = ['totalAdClicks', 'revenue']  
assembler = VectorAssembler(inputCols=featuresUsed, outputCol="features_  
unscaled")  
assembled = assembler.transform(data)
```

Scale the features using StandardScaler:

```
In [11]: scaler = StandardScaler(inputCol="features_unscaled", outputCol="feature  
s", withStd=True, withMean=True)  
scalerModel = scaler.fit(assembled)  
scaledData = scalerModel.transform(assembled)
```

Select the features column make the data persist:

```
In [12]: scaledData = scaledData.select("features")  
scaledData.persist()
```

```
Out[12]: DataFrame[features: vector]
```

We can now perform KMeans clustering to generate 2 clusters:

```
In [13]: kmeans = KMeans(k=2, seed=1)
         model = kmeans.fit(scaledData)
         transformed = model.transform(scaledData)
```

Print the center of these two clusters:

```
In [14]: centers = model.clusterCenters()
         centers
```

```
Out[14]: [array([ 0.84174521,  0.51884657]), array([-0.79780796, -0.49176392])]
```

Analyze center of these two clusters:

Each array denotes the center for a cluster:

One Cluster is centered at ... `array([0.84174521, 0.51884657])`

Other Cluster is centered at ... `array([-0.79780796, -0.49176392])`

First number (field1) in each array refers to scaled version of the number of ad-clicks and the second number (field2) is the scaled version of the revenue per user. Compare the 1st number of each cluster to see how differently users in each cluster behave when it comes to clicking ads. Compare the 2nd number of each cluster to see how differently users in each cluster behave when it comes to buying stuff.

In one cluster, in general, players click on ads much more often and spend more money on in-app purchases. Assuming that Eglenec Inc. gets paid for showing ads and for hosting in-app purchase items, we can use this information to increase game's revenue by increasing the prices for ads we show to the frequent-clickers, and charge higher fees for hosting the in-app purchase items shown to the higher revenue generating buyers.

Note: This analysis requires you to compare the cluster centers and find any 'significant' differences in the corresponding feature values of the centers. The answer to this question will depend on the features you have chosen.

Some features help distinguish the clusters remarkably while others may not tell you much. At this point, if you don't find clear distinguishing patterns, perhaps re-running the clustering model with different numbers of clusters and revising the features you picked would be a good idea.

In []: