databricks (http://databricks.com)

Predicting_IBM_Employee_Attrition_Part_5_k-Means_Clusters

I. Use k-Means Clustering to Group Employees into Clusters

Initiate new Spark session.

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('Employee_Attrition_Part_5').getOrCreate()
```

Import numpy, pandas, and data visualization libraries.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Import churn modeling CSV file into PySpark dataframe called churn_model2.

```
churn_model2 = spark.read.csv('/FileStore/tables/churn_modeling_data.csv',
inferSchema=True, header=True)
```

Check number of rows and columns in churn_model2 dataframe.

```
print(churn_model2.count(), len(churn_model2.columns))
```

```
1470 11
```

View structure of churn_model2 dataframe.

```
churn_model2.printSchema()
```

```
root
 |-- Churn: integer (nullable = true)
 |-- Age: integer (nullable = true)
 |-- DistanceFromHome: integer (nullable = true)
 |-- EnvironmentSatisfaction: integer (nullable = true)
 |-- JobInvolvement: integer (nullable = true)
 |-- MonthlyIncome: integer (nullable = true)
 |-- StockOptionLevel: integer (nullable = true)
 |-- Sales_Rep: integer (nullable = true)
 |-- Single: integer (nullable = true)
 |-- BusTravLevel: integer (nullable = true)
 |-- Overtime_Dum: integer (nullable = true)
```

View first five rows of churn_model2 dataframe.

```
display(churn_model2.head(5))
```

Churn	Age	DistanceFromHome	EnvironmentSatisfaction	JobInvolve
1	41	1	1	2
0	49	8	2	1
1	37	2	3	1
0	33	3	3	2
0	27	2	0	2

Drop Churn target variable and include only predictor features in churn_model2 dataframe for k-means clustering.

```
churn_pred_feat = churn_model2.drop(*['Churn'])
```

Convert predictor features into vector column using VectorAssembler.

```
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
```

```
assembler = VectorAssembler(inputCols=['Age', 'DistanceFromHome',
'EnvironmentSatisfaction', 'JobInvolvement', 'MonthlyIncome',
'StockOptionLevel', 'Sales_Rep', 'Single', 'BusTravLevel', 'Overtime_Dum'],
outputCol='features')
```

```
churn_pred_feat_vect = assembler.transform(churn_pred_feat)
display(churn_pred_feat_vect.head(5))
```

Age	DistanceFromHome	EnvironmentSatisfaction	JobInvolvement	MonthlyI
41	1	1	2	5993
49	8	2	1	5130
37	2	3	1	2090
33	3	3	2	2909
27	2	0	2	3468

Center and scale all predictor features.

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

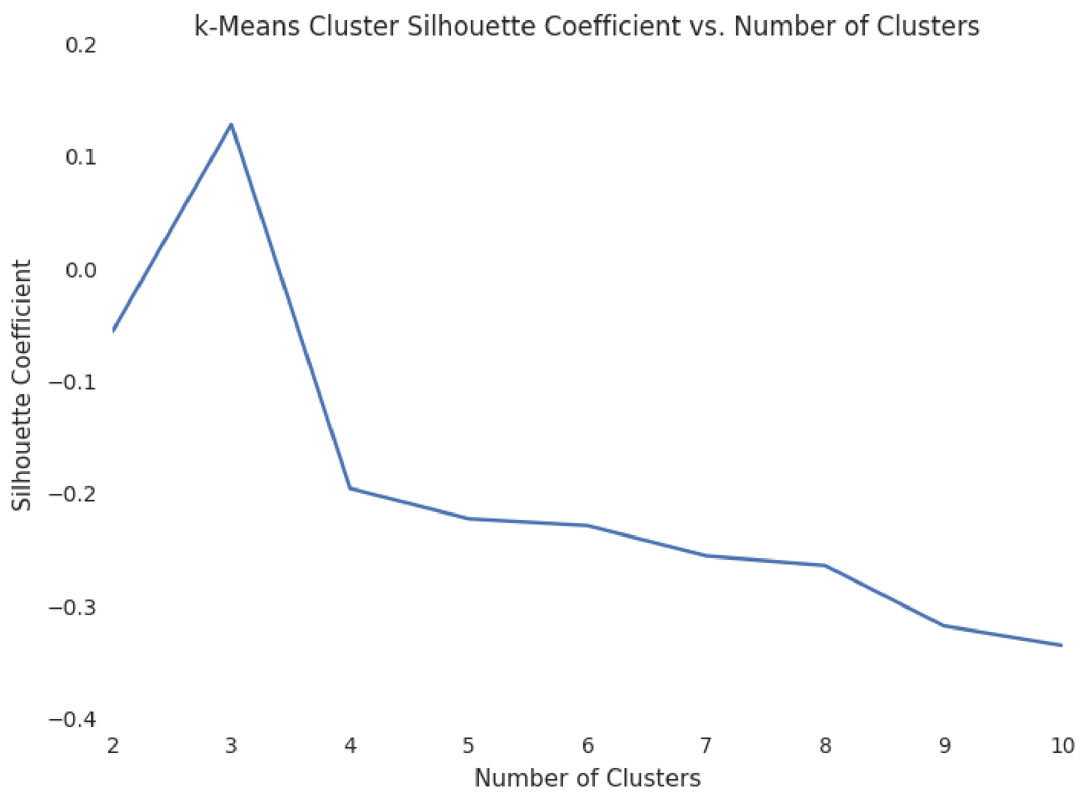
```
scaler = StandardScaler(inputCol='features', outputCol='scaled_features',
withStd=True, withMean=True)
scaler_model = scaler.fit(churn_pred_feat_vect)
churn_pred_feat_scaled = scaler_model.transform(churn_pred_feat_vect)
```

Using scaled predictor features, determine how many clusters will yield the highest silhouette score and produce the lowest inertia (which is sum of squared error or distance between a data point and the cluster centroid).

```
from pyspark.ml.clustering import KMeans
from pyspark.ml.evaluation import ClusteringEvaluator
```

```
clus_eval = ClusteringEvaluator()
```

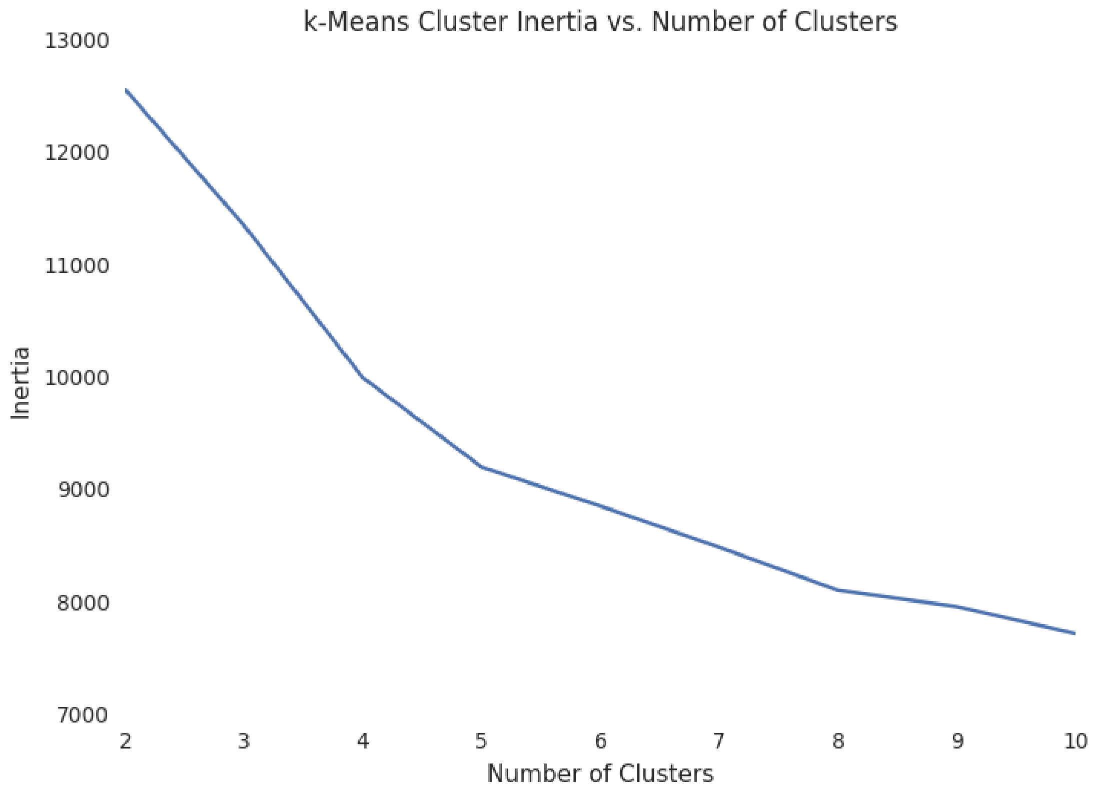
```
k_range = range(2, 11)


sil_scores = []


inertia_values = []


for k_value in k_range:
  kmc1 = KMeans(featuresCol='scaled_features', k=k_value).setSeed(123)
  kmc1_model = kmc1.fit(churn_pred_feat_scaled)
  inertia_values.append(kmc1_model.computeCost(churn_pred_feat_scaled))
  kmc1_predictions = kmc1_model.transform(churn_pred_feat_scaled)
  sil_scores.append(clus_eval.evaluate(kmc1_predictions))


clus_sil_score, ax = plt.subplots()
ax.plot(k_range, sil_scores)
ax.set_xlabel('Number of Clusters')
ax.set_ylabel('Silhouette Coefficient')
ax.set_title('k-Means Cluster Silhouette Coefficient vs. Number of Clusters')
display(clus_sil_score)
```



```
dict(zip(k_range, sil_scores))
```

```
Out[20]:
{2: -0.05339519820395932,
 3: 0.13052274927074242,
 4: -0.1932975231466738,
 5: -0.2203365278439531,
 6: -0.22640454950944017,
 7: -0.2534142285461967,
 8: -0.2618607909548427,
 9: -0.3156176700871847,
 10: -0.333179263985305}
```

```
clus_inertias, ax = plt.subplots()
ax.plot(k_range, inertia_values)
ax.set_xlabel('Number of Clusters')
ax.set_ylabel('Inertia')
ax.set_title('k-Means Cluster Inertia vs. Number of Clusters')
display(clus_inertias)
```



```
dict(zip(k_range, inertia_values))
```

```
Out[22]:
{2: 12575.875971955697,
 3: 11366.65913206184,
 4: 10012.298230300212,
 5: 9216.848432554125,
 6: 8869.376481685269,
 7: 8503.43906691617,
 8: 8121.02485210504,
 9: 7975.112045996482,
 10: 7736.043697767086}
```

- The employees in the churn_model2 dataframe can be grouped into 3 clusters using k-means clustering and features for predicting churn. Even though the data points in each cluster are less closely related to the centroid, as indicated by the high inertia value, the clusters overlap each other the least from having the highest silhouette score.

Assign employees in churn_model2 dataframe into 3 clusters.

```
kmc2 = KMeans(featuresCol='scaled_features', k=3).setSeed(123)
kmc2_model = kmc2.fit(churn_pred_feat_scaled)
kmc2_predictions = kmc2_model.transform(churn_pred_feat_scaled)
display(kmc2_predictions.head(5))
```

Age	DistanceFromHome	EnvironmentSatisfaction	JobInvolvement	MonthlyIncom
41	1	1	2	5993
49	8	2	1	5130
37	2	3	1	2090
33	3	3	2	2909
27	2	0	2	3468

Inspect cluster traits by calculating cluster centers as mean of features for predicting churn.

```
from pyspark.sql import functions as F


clus_center_means = kmc2_predictions.withColumn('Cluster',
kmc2_predictions['prediction']).groupBy('Cluster').agg(F.mean('Age').alias('Age
'), F.mean('DistanceFromHome').alias('DistanceFromHome'),
F.mean('EnvironmentSatisfaction').alias('EnvironmentSatisfaction'),
F.mean('JobInvolvement').alias('JobInvolvement'),
F.mean('MonthlyIncome').alias('MonthlyIncome'),
F.mean('StockOptionLevel').alias('StockOptionLevel'),
F.mean('Sales_Rep').alias('Sales_Rep'), F.mean('Single').alias('Single'),
F.mean('BusTravLevel').alias('BusTravLevel'),
F.mean('Overtime_Dum').alias('Overtime_Dum')).orderBy('Cluster')
display(clus_center_means)
```

Cluster	Age	DistanceFromHome	EnvironmentSatisfaction	JobInvolv
0	47.844106463878326	8.939163498098859	1.8022813688212929	1.661596
1	34.96173469387755	9.501275510204081	1.6964285714285714	1.783163
2	33.77068557919622	8.777777777777779	1.718676122931442	1.673758

Obtain number of employees for each cluster.

Cluster
0
1
2