Clustering_Task_:_Saad_Lahlali

November 8, 2021

[106]:
```
%%latex
\tableofcontents
```

Contents

1 Settings

[5]:
```
import pyspark
sc = pyspark.SparkContext(appName="Clustering Task")
```

[6]:
```
%matplotlib inline
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
```

[7]:
```
from pyspark.sql.types import StructType, StructField, DoubleType, IntegerType,␣
 ↪StringType
from pyspark.ml.feature import VectorAssembler
from pyspark.sql import SQLContext
from pyspark.ml.feature import StringIndexer
from pyspark.ml.feature import StandardScaler
from pyspark.sql.functions import col
```

2 Data Preparation

[8]:
```
sqlContext = SQLContext(sc)
#indexer needed so that labels are 0 and 1

schema = StructType([ StructField("c"+str(0),StringType())]+[␣
↪StructField("c"+str(i),DoubleType()) for i in range(1,8)])
# schema to cast data, can use inferschema also

raw_data = sqlContext.read.csv("/content/drive/MyDrive/exo2_tocluster.
↪csv",schema=schema)
#droping rows with nulls
raw_data_nn = raw_data.dropna()

assembled_data = VectorAssembler(inputCols=["c"+str(i) for i in␣
↪range(1,8)],outputCol="features").transform(raw_data)

#standardizing the data
scale=StandardScaler(inputCol='features',outputCol='standardized_features')
data_scale=scale.fit(assembled_data)
data_scale_output=data_scale.transform(assembled_data)
```

```
/usr/local/lib/python3.7/dist-packages/pyspark/sql/context.py:79: FutureWarning:
Deprecated in 3.0.0. Use SparkSession.builder.getOrCreate() instead.
  FutureWarning
```

[ ]:
```
data_scale_output.show()
```

3 Cluster Selection

3.1 K-Means

[10]:
```
from pyspark.ml.clustering import KMeans
from pyspark.ml.evaluation import ClusteringEvaluator
```

3.1.1 Unstandardized data

[ ]:
```
silhouette_score=[]
evaluator_silhouette = ClusteringEvaluator(predictionCol='prediction',␣
↪featuresCol='features', metricName='silhouette',␣
↪distanceMeasure='squaredEuclidean')

cost = []
X = range(2,27)
```

```
# we train the Kmeans models 10 times in order to soften the curves and avoid␣
↪rare cases since kmeans is initialized with random centroids
for _ in range(10):
  S, I = [], []

  for i in X: # ie with cluster numbers ranging from 2 to 26
      print('For', i, 'clusters :')
      #training the model
      KMeans_algo=KMeans(featuresCol='features', k=i)
      KMeans_fit=KMeans_algo.fit(data_scale_output)

      #predictions of clusters on data
      output=KMeans_fit.transform(data_scale_output)

      #evaluating the models
      score=evaluator_silhouette.evaluate(output)
      S.append(score)
      print("Silhouette Score:",score)

      I.append(KMeans_fit.summary.trainingCost)
      print("Costcore:",KMeans_fit.summary.trainingCost)
      print('\n')

  silhouette_score.append(S)
  cost.append(I)
```

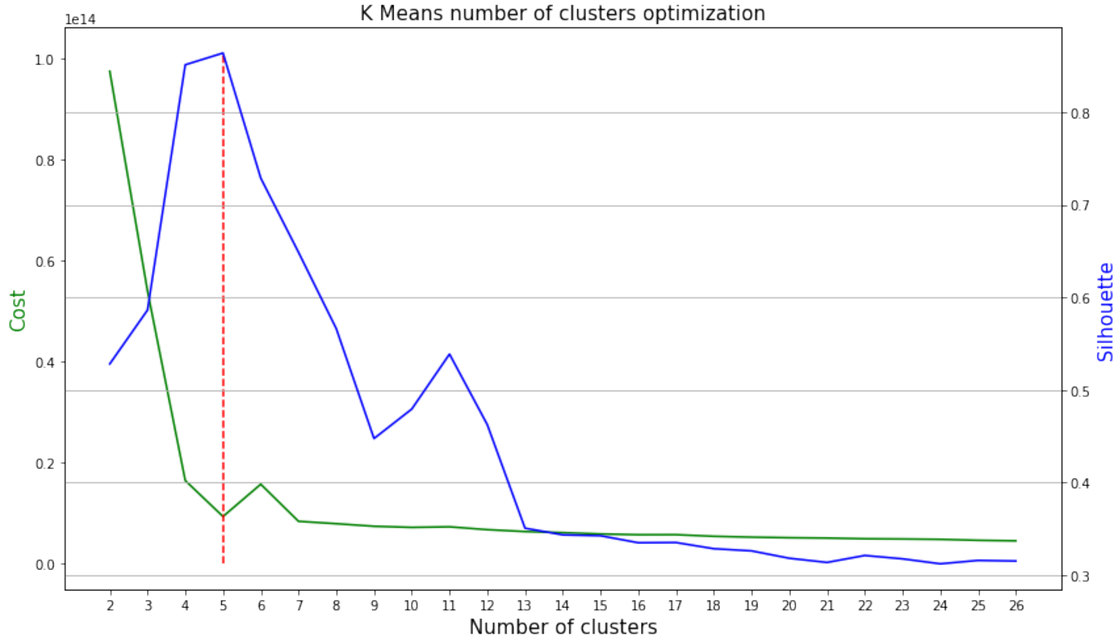
[14]:
```
mean_Sil, mean_Cost = np.mean(silhouette_score, axis=0), np.mean(cost, axis=0)
```

[25]:
```
fig, ax1 = plt.subplots(figsize=(12,7))

ax1.set_xlabel('Number of clusters', fontsize=15)
ax1.set_ylabel('Cost', fontsize=15, color='green')
ax1.plot(X, mean_Cost, color='green')
ax1.plot([5,5],[0,1e14+1e12], linestyle='dashed', color='red')
ax1.set_xticks(X)

ax2 = ax1.twinx()
ax2.set_ylabel('Silhouette', fontsize=15, color='blue')
ax2.plot(X, mean_Sil, color='blue')


plt.grid()
plt.title('K Means number of clusters optimization', fontsize=15)
fig.tight_layout()
plt.show()
```



After analyzing the previous graph where we've calculated the sum of squares and the silhouette score, we can deduce the optimal number of clusters.

Indeed, by applying the elbow to the cost curve we can deduce that the optimal number of clusters is between 5 and 7.

Also, by looking at the silhouette curve we can deduce that the optimal number of clusters is the one for which we get maximum silhouette score therefor the number of clusters 4 and 5 would be good candidates.

From the candidates obtained from the cost and silhouette scores we can confidently say that the optimal number of clusters is 5.

[46]:
```
optimal_K = 5
KMeans_algo=KMeans(featuresCol='features', k=optimal_K)
KMeans_fit=KMeans_algo.fit(data_scale_output)
output=KMeans_fit.transform(data_scale_output)
```

3.1.2 Standardized data

[ ]:
```
silhouette_score=[]
evaluator_silhouette = ClusteringEvaluator(predictionCol='prediction',␣
 ↪featuresCol='standardized_features', metricName='silhouette',␣
 ↪distanceMeasure='squaredEuclidean')

cost = []
X = range(2,27)

for _ in range(10):
  S, I = [], []
```

```
for i in X:
    print('For', i, 'clusters :')
    #training the model
    KMeans_algo=KMeans(featuresCol='standardized_features', k=i)
    KMeans_fit=KMeans_algo.fit(data_scale_output)

    #predictions of clusters on data
    output=KMeans_fit.transform(data_scale_output)

    #evaluating the models
    score=evaluator_silhouette.evaluate(output)
    S.append(score)
    print("Silhouette Score:",score)

    I.append(KMeans_fit.summary.trainingCost)
    print("Costcore:",KMeans_fit.summary.trainingCost)
    print('\n')

silhouette_score.append(S)
cost.append(I)
```

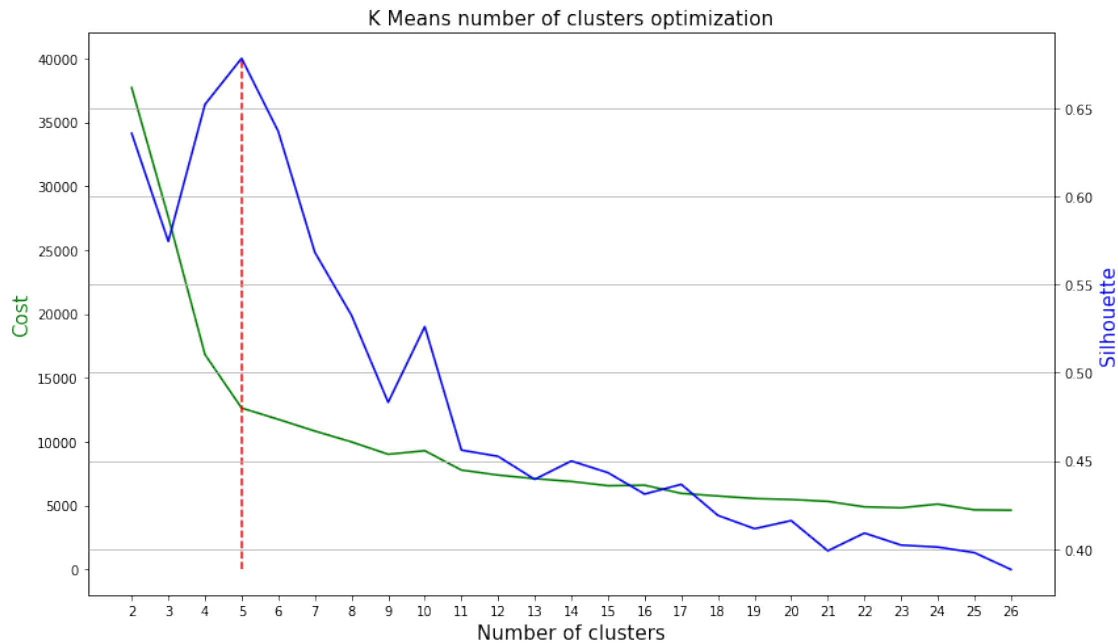
[40]:
```
mean_Sil, mean_Cost = np.mean(silhouette_score, axis=0), np.mean(cost, axis=0)
```

[45]:
```
fig, ax1 = plt.subplots(figsize=(12,7))

ax1.set_xlabel('Number of clusters', fontsize=15)
ax1.set_ylabel('Cost', fontsize=15, color='green')
ax1.plot(X, mean_Cost, color='green')
ax1.plot([5,5],[0,40000], linestyle='dashed', color='red')
ax1.set_xticks(X)

ax2 = ax1.twinx()
ax2.set_ylabel('Silhouette', fontsize=15, color='blue')
ax2.plot(X, mean_Sil, color='blue')


plt.grid()
plt.title('K Means number of clusters optimization', fontsize=15)
fig.tight_layout()
plt.show()
```



By standadizing the data, we can detect that 5 is a candidate with more confidence than before. However, we are less confident with elbow method since the curve descent is much smoother than before therefor increasing the number of candidates and descresing our confidence.

[63]:
```
optimal_K = 5
KMeans_algo=KMeans(featuresCol='features', k=optimal_K)
KMeans_fit=KMeans_algo.fit(data_scale_output)
output_non_stand=KMeans_fit.transform(data_scale_output)
```

[64]:
```
KMeans_algo=KMeans(featuresCol='standardized_features', k=optimal_K)
KMeans_fit=KMeans_algo.fit(data_scale_output)

#predictions of clusters on data
output_stand=KMeans_fit.transform(data_scale_output)
```

3.2 Comparing the predictions made by each of the models

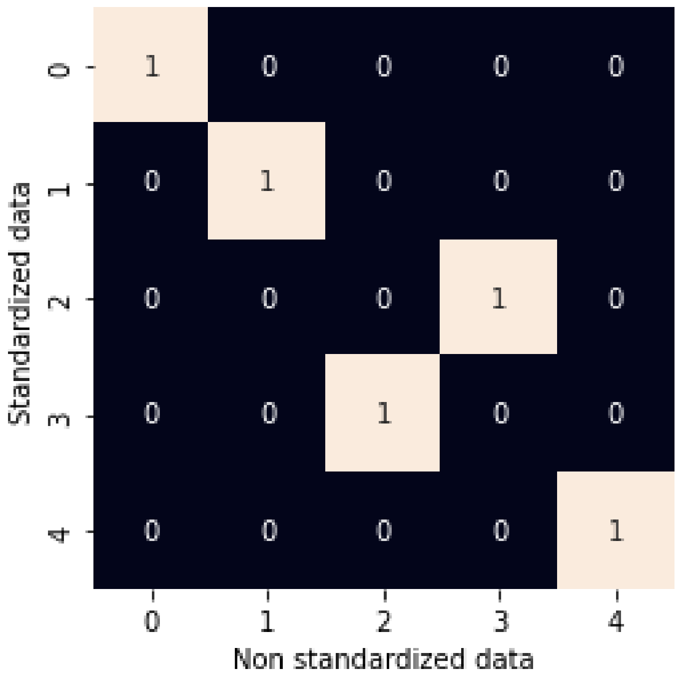
[65]:
```
pred_non_stand = [w.prediction for w in output_non_stand.select('prediction').
  ↪collect()]
pred_stand = [w.prediction for w in output_stand.select('prediction').collect()]
```

[69]:
```
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import normalize
import seaborn as sn

cm = confusion_matrix(pred_non_stand, pred_stand, labels=range(5))
```

```
normed_cm = normalize(cm, axis=1, norm='l1')

sn.heatmap(normed_cm.T, square=True, annot=True, cbar=False,
            xticklabels=range(5),
            yticklabels=range(5))
plt.xlabel('Non standardized data')
plt.ylabel('Standardized data')
plt.show()
```



We get perfect similarity to the predictions made by the kmeans on standardized data and not standardized one.

4 Testing

[83]:
```
pred = [w.prediction for w in output_stand.select('prediction').collect()]
ind = [w.c0 for w in output_stand.select('c0').collect()]
```

[101]:
```
dic = dict()
for i in range(optimal_K):
  dic[i] = chr(ord('a') + i)
```

[104]:
```
with open('exo2.csv','wb') as file:
  for p, i in zip(pred, ind):
    file.write((str(i)+','+str(dic[p])+'\n').encode())
```

```
file.write((opti_letter).encode())
```

5 To PDF

[ ]:
```
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
!pip install pypandoc
```

[108]:
```
!jupyter nbconvert --to PDF --TemplateExporter.exclude_input=False␣
↪"Clustering_Task_:_Saad_Lahlali.ipynb"
```

```
[NbConvertApp] Converting notebook Clustering_Task_:_Saad_Lahlali.ipynb to PDF
[NbConvertApp] Support files will be in Clustering_Task_:_Saad_Lahlali_files/
[NbConvertApp] Making directory ./Clustering_Task_:_Saad_Lahlali_files
[NbConvertApp] Making directory ./Clustering_Task_:_Saad_Lahlali_files
[NbConvertApp] Making directory ./Clustering_Task_:_Saad_Lahlali_files
[NbConvertApp] Writing 94601 bytes to ./notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: [u'xelatex', u'./notebook.tex',
'-quiet']
[NbConvertApp] Running bibtex 1 time: [u'bibtex', u'./notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 149116 bytes to Clustering_Task_:_Saad_Lahlali.pdf
```