clustering_titanic_Rafael

December 1, 2021

[21]:
```
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats
from collections import Counter
from sklearn.cluster import KMeans,DBSCAN
from impyute.imputation.cs import mice
from yellowbrick.cluster import KElbowVisualizer
from sklearn.metrics import silhouette_score, homogeneity_completeness_v_measure
```

0.1 Carregando os dados

[4]:
```
train = pd.read_csv('../Datasets/train_titanic.csv')
```

[5]:
```
train
```

[5]:
```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           …       …     …
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                 Name     Sex   Age  SibSp  \
0                             Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                              Heikkinen, Miss. Laina  female  26.0      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                            Allen, Mr. William Henry    male  35.0      0
..                                                 …     …   …    …
886                            Montvila, Rev. Juozas    male  27.0      0
```

```
887                      Graham, Miss. Margaret Edith  female  19.0      0
888          Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                             Behr, Mr. Karl Howell    male  26.0      0
890                               Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Cabin Embarked
0        0         A/5 21171   7.2500   NaN        S
1        0          PC 17599  71.2833   C85        C
2        0  STON/O2. 3101282   7.9250   NaN        S
3        0            113803  53.1000  C123        S
4        0            373450   8.0500   NaN        S
..     …               …       …   …      …
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q

[891 rows x 12 columns]
```

0.2 Remoção de atributos

Remoção dos atributos **Name, Ticket** e **Cabin** por terem alta cardinalidade. Após a remoção o conjunto é dividido: + Atributos -> **X** + Alvo (***Survived***) -> **y**

[6]:
```
#removendo o atributo Survived
df = train.drop(['Name','Ticket','Cabin'], axis=1)
X, y = df.drop(['Survived'],axis=1), df[['Survived']]
```

0.3 Tratamento de atributos categóricos com *one hot encoding*

Utilizando dummies do pandas para codificar os atributos **Sex** e ** Embarked**

[7]:
```
df=pd.get_dummies(X)
```

[8]:
```
df
```

[8]:
```
     PassengerId  Pclass   Age  SibSp  Parch     Fare  Sex_female  Sex_male  \
0              1       3  22.0      1      0   7.2500           0         1
1              2       1  38.0      1      0  71.2833           1         0
2              3       3  26.0      0      0   7.9250           1         0
3              4       1  35.0      1      0  53.1000           1         0
4              5       3  35.0      0      0   8.0500           0         1
..           …     …   …    …      …      …        …         …
886          887       2  27.0      0      0  13.0000           0         1
887          888       1  19.0      0      0  30.0000           1         0
888          889       3   NaN      1      2  23.4500           1         0
889          890       1  26.0      0      0  30.0000           0         1
```

```
890          891       3  32.0      0      0   7.7500           0          1

     Embarked_C  Embarked_Q  Embarked_S
0             0           0           1
1             1           0           0
2             0           0           1
3             0           0           1
4             0           0           1
..          ...         ...         ...
886           0           0           1
887           0           0           1
888           0           0           1
889           1           0           0
890           0           1           0

[891 rows x 11 columns]
```

0.4 Tratamento de valores ausentes

Utilizarei a técnica de imputação **Multiple imputation by chained equations (MICE)** para o atributo **Age**. Para os atributos **Embarked** e **Fare** será utilizado a remoção dos valores ausentes, uma vez que a presença é bem pequena.

A técnica MICE, em vez de utilizar apenas uma imputação, preenche os valores ausentes utilizando várias imputações de forma iterativa até uma conversão.

Mais informações sobre essa técnica pode ser encontrada neste artigo.

Abaixo segue uma animação do funcionamento da técnica para três variáveis. Referência: medium

[9]:
```
#verificando valores nulos
X.isnull().sum()
```

[9]:
```
PassengerId      0
Pclass           0
Sex              0
Age            177
SibSp            0
Parch            0
Fare             0
Embarked         2
dtype: int64
```

[10]:
```
#aplicando o MICE
imputed_values = mice(df.values)

#convertendo o resultado em dataframe
df_ = pd.DataFrame(imputed_values, columns=df.columns)
```

[11]:
```
#verificando novamente a presença de valores nulos
df_.isnull().sum()
```

[11]:
```
PassengerId    0
Pclass         0
Age            0
SibSp          0
Parch          0
Fare           0
Sex_female     0
Sex_male       0
Embarked_C     0
Embarked_Q     0
Embarked_S     0
dtype: int64
```

[12]:
```
#dataframe após o one hot encoding
df_
```

[12]:

	PassengerId	Pclass	Age	SibSp	Parch	Fare	Sex_female	Sex_male	Embarked_C	Embarked_Q	Embarked_S
0	1.0	3.0	22.000000	1.0	0.0	7.2500	0.0	1.0	0.0	0.0	1.0
1	2.0	1.0	38.000000	1.0	0.0	71.2833	1.0	0.0	1.0	0.0	0.0
2	3.0	3.0	26.000000	0.0	0.0	7.9250	1.0	0.0	0.0	0.0	1.0
3	4.0	1.0	35.000000	1.0	0.0	53.1000	1.0	0.0	0.0	0.0	1.0
4	5.0	3.0	35.000000	0.0	0.0	8.0500	0.0	1.0	0.0	0.0	1.0
..	…	…	…	…	…	…	…	…	…	…	…
886	887.0	2.0	27.000000	0.0	0.0	13.0000	0.0	1.0	0.0	0.0	1.0
887	888.0	1.0	19.000000	0.0	0.0	30.0000	1.0	0.0	0.0	0.0	1.0
888	889.0	3.0	19.462726	1.0	2.0	23.4500	1.0	0.0	0.0	0.0	1.0
889	890.0	1.0	26.000000	0.0	0.0	30.0000	0.0	1.0	1.0	0.0	0.0
890	891.0	3.0	32.000000	0.0	0.0	7.7500	0.0	1.0	0.0	1.0	0.0

[891 rows x 11 columns]

0.5 Normalização dos dados

Utilizarei o Zcore de acordo com a fórmula abaixo:

$$X' = \frac{X - \mu}{\sigma}$$

onde X é o valor original, μ é a média dos valores e σ é o desvio padrão dos valores.

[13]:
```
#aplicando o zscore para normalização
df_scaled = stats.zscore(df_)
```

[14]:
```
df_scaled
```

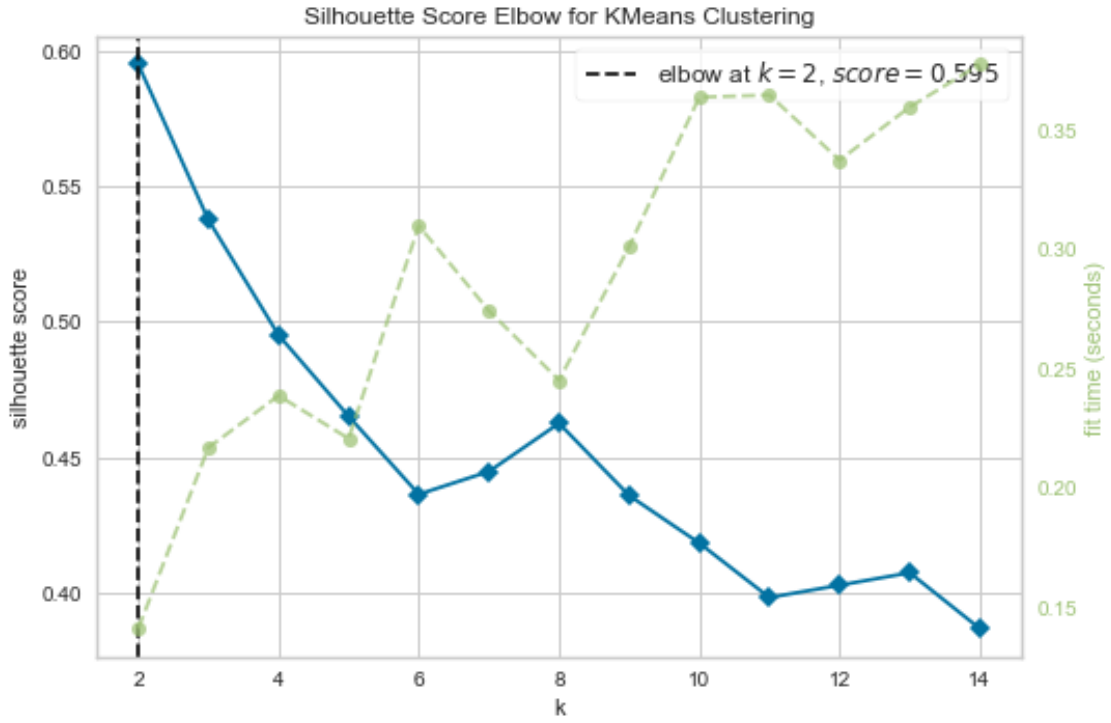
[14]:
```
     PassengerId    Pclass       Age     SibSp     Parch      Fare  \
0      -1.730108  0.827377 -0.540144  0.432793 -0.473674 -0.502445
1      -1.726220 -1.566107  0.631506  0.432793 -0.473674  0.786845
2      -1.722332  0.827377 -0.247232 -0.474545 -0.473674 -0.488854
3      -1.718444 -1.566107  0.411821  0.432793 -0.473674  0.420730
4      -1.714556  0.827377  0.411821 -0.474545 -0.473674 -0.486337
..           …       …       …       …       …       …
886     1.714556 -0.369365 -0.174004 -0.474545 -0.473674 -0.386671
887     1.718444 -1.566107 -0.759829 -0.474545 -0.473674 -0.044381
888     1.722332  0.827377 -0.725944  0.432793  2.008933 -0.176263
889     1.726220 -1.566107 -0.247232 -0.474545 -0.473674 -0.044381
890     1.730108  0.827377  0.192137 -0.474545 -0.473674 -0.492378

     Sex_female  Sex_male  Embarked_C  Embarked_Q  Embarked_S
0     -0.737695  0.737695   -0.482043   -0.307562    0.619306
1      1.355574 -1.355574    2.074505   -0.307562   -1.614710
2      1.355574 -1.355574   -0.482043   -0.307562    0.619306
3      1.355574 -1.355574   -0.482043   -0.307562    0.619306
4     -0.737695  0.737695   -0.482043   -0.307562    0.619306
..          …       …          …          …         …
886   -0.737695  0.737695   -0.482043   -0.307562    0.619306
887    1.355574 -1.355574   -0.482043   -0.307562    0.619306
888    1.355574 -1.355574   -0.482043   -0.307562    0.619306
889   -0.737695  0.737695    2.074505   -0.307562   -1.614710
890   -0.737695  0.737695   -0.482043    3.251373   -1.614710

[891 rows x 11 columns]
```

0.6 Método Elbow

Encontrando o melhor k (número de clusters) de acordo com o conjunto de dados utilizando como métrica de avaliação o **silhouette**. A avaliação é realizada no intervalo de [2,15]. Como observamos abaixo, o melhor k é 2.

[15]:
```
model = KMeans()
visualizer_s = KElbowVisualizer(model, k=(2,15), metric='silhouette')
```

```
visualizer_s.fit(df_)
visualizer_s.show()
```



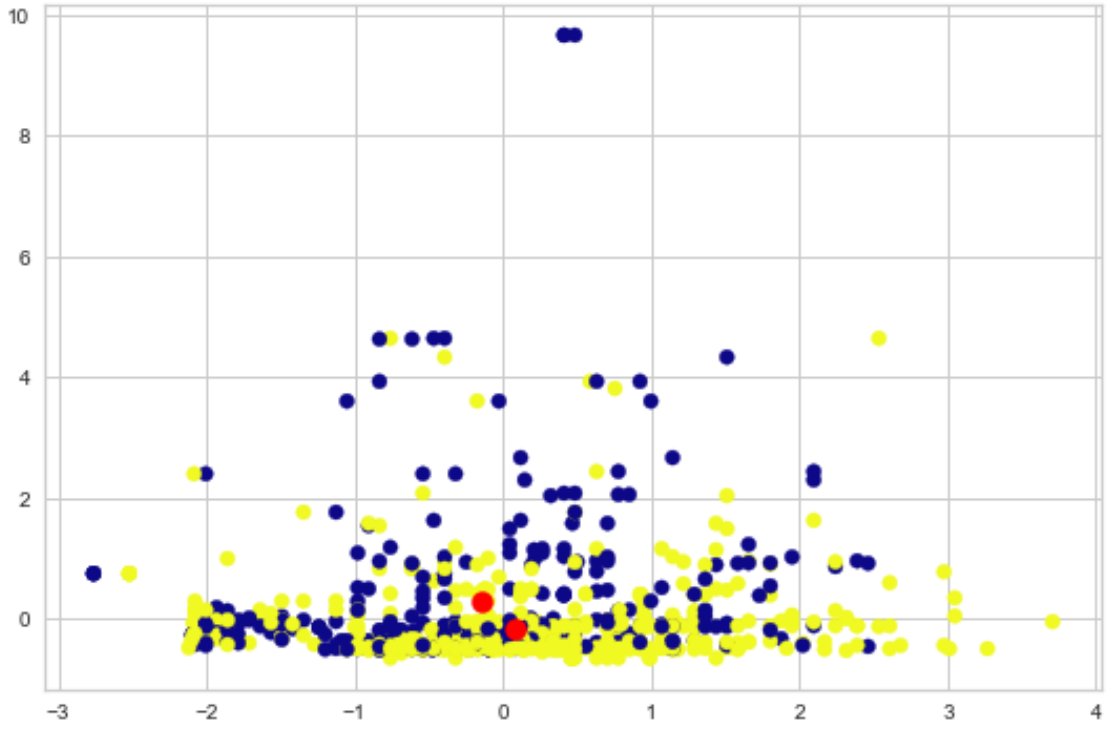
[15]:
```
<AxesSubplot:title={'center':'Silhouette Score Elbow for KMeans Clustering'},
xlabel='k', ylabel='silhouette score'>
```

1 Aplicando a 1ª técnica: *KMeans*

Essa técnica basea-se na divisão do conjunto de dados em k grupos (clusters) onde cada amostra pertence ao grupo em relação a média das amostras pertencente ao grupo (centróides). O ponto principal é a escolha do melhor valor de k. Para isso, foi utilizado o método ''joelho'' (**elbow**) para a escolha do melhor valor de k indicando $k = 2$

[16]:
```
#configurando o numero de clusters e aplicando no conjunto de dados
kmeans = KMeans(n_clusters=2,init='k-means++',random_state=1, n_init=50).
 ↪fit(df_scaled)

#plotando o resultado
plt.figure(figsize=(9,6))
plt.scatter(df_scaled['Age'], df_scaled['Fare'], c=kmeans.labels_,␣
 ↪cmap='plasma')
plt.scatter(kmeans.cluster_centers_[:,2], kmeans.cluster_centers_[:,5], s=100,␣
 ↪c='red')
```

[16]: <matplotlib.collections.PathCollection at 0x24373f29970>

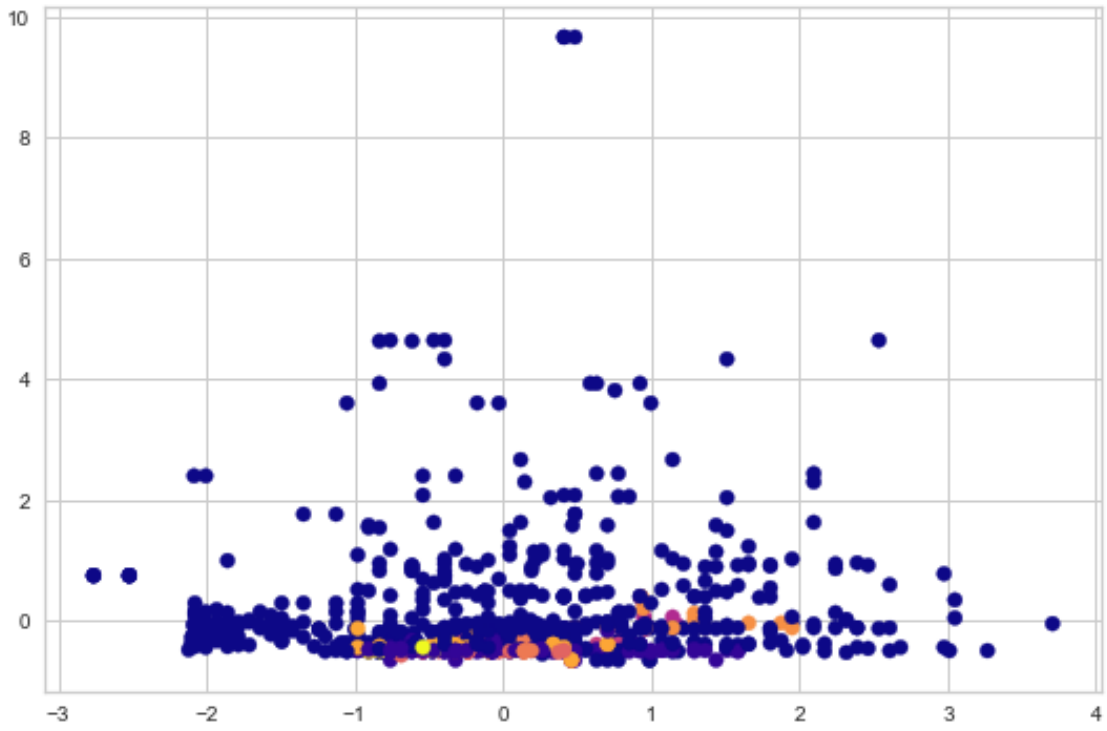


2 Aplicando a 2ª técnica: *DBSCAN*

Está técnica basea-se no agrupamento por densidade espacial, expandindo o agrupamento pelas amostras de maior densidade

[47]:
```
dbscan = DBSCAN().fit(df_scaled)

#plotando o resultado
plt.figure(figsize=(9,6))
plt.scatter(df_scaled['Age'], df_scaled['Fare'], c=dbscan.labels_,␣
 ↪cmap='plasma')
#plt.scatter(dbscan.cluster_centers_[:,2], dbscan.cluster_centers_[:,5], s=100,␣
 ↪c='red')
```

[47]: <matplotlib.collections.PathCollection at 0x243757f82e0>



3 Avaliando os resultados

3.1 Silhouette coefficient

O silhouete coeficiente varia de -1 a 1. A função **silhouette_score** retorna o valor médio do coeficiente para todas as amostras.

[48]:
```
silhouette_k = silhouette_score(df_scaled, kmeans.labels_, random_state=1)
silhouette_d = silhouette_score(df_scaled, dbscan.labels_, random_state=1)

print(f'Coeficiente de silhouette utilizando KMeans: {silhouette_k:.4f}')
print(f'Coeficiente de silhouette utilizando DBSCAN: {silhouette_d:.4f}')
```

```
Coeficiente de silhouette utilizando KMeans: 0.2623
Coeficiente de silhouette utilizando DBSCAN: -0.1816
```

Logo, para o conjunto de dados do Titanic utilizando o KMeans, o coeficiente de silhouette ficou em 0.26, positivo, indicando um bom agrupamento de forma geral. Enquanto que para o DBSCAN, o coeficiente de silhouette ficou em -0.18, indicando um mau agrupamento dos dados ou criação de mais de dois clusters.

3.2 Homogeneity, Completeness e V_measure

- ***Homogeneity***: indica se os grupos/clusters contém membros pertencentes a uma mesma classe

- ***Completeness***: indica se membros de uma mesma classe estão no mesmo grupo/cluster.
- ***V__measure***: é a média harmônica entre homogeneity e completeness

Para ambas as métrica, a variação é de 0 a 1, e quanto maior o valor, melhor o agrupamento.

[49]:
```
homo_k, compl_k, v_meas_k = homogeneity_completeness_v_measure(y.values.
 ↪squeeze(),kmeans.labels_)
homo_d, compl_d, v_meas_d = homogeneity_completeness_v_measure(y.values.
 ↪squeeze(),dbscan.labels_)

print(f'Métricas utilizando KMeans:')
print(f'+ Homogeneity: {homo_k:.4f}')
print(f'+ Completeness: {compl_k:.4f}')
print(f'+ V_measure: {v_meas_k:.4f}')
print('\n')
print(f'Métricas utilizando DBSCAN:')
print(f'+ Homogeneity: {homo_d:.4f}')
print(f'+ Completeness: {compl_d:.4f}')
print(f'+ V_measure: {v_meas_d:.4f}')
```

```
Métricas utilizando KMeans:
+ Homogeneity: 0.2312
+ Completeness: 0.2368
+ V_measure: 0.2339


Métricas utilizando DBSCAN:
+ Homogeneity: 0.1473
+ Completeness: 0.0669
+ V_measure: 0.0920
```

Os resultados mostram que as métricas para o KMeans não foram tão altas, ambas tendo valores de 0.23. Já para o DBSCAN, as métricas também ficaram baixas, principalmente a completeness, indicando que o agrupamento não está separando bem membros de uma mesma classe, o que consequentemente influencia no valor de V-measure. Logo, juntamente com as métricas analisadas, o KMeans apresentou o melhor desempenho de agrupamento em relação ao DBSCAN.

Realizando a análise das métricas para o DBSCAN é possível observar que as mesmas são influenciadas pelo parâmetro *eps*, o qual controla a distância máxima entre amostras para serem consideradas vizinhas uma da outra. Assim, outra análise será levada em conta variando esse parâmetro no intervalo de 0.1 a 5 e retirando a média.

[51]:
```
dbscan_silhouette = []
homo_d = []
compl_d = []
v_meas_d = []

values = np.arange(0.1,5,0.1)
#calculando a variação para o DBSCAN
```

```
for i in values:
    dbscan_ = DBSCAN(eps=i).fit(df_scaled)
    silhouette_d = silhouette_score(df_scaled, dbscan_.labels_, random_state=1)
    dbscan_silhouette.append(silhouette_d)

    homo, compl, v_meas = homogeneity_completeness_v_measure(y.values.
 ↪squeeze(),dbscan_.labels_)
    homo_d.append(homo)
    compl_d.append(compl)
    v_meas_d.append(v_meas)

print('Métricas de avaliação do DBSCAN variando eps')
print(f'Coeficiente de silhouette: {np.array(dbscan_silhouette).mean():.4f}')
print(f'Homogeneity: {np.array(homo_d).mean():.4f}')
print(f'Completeness: {np.array(compl_d).mean():.4f}')
print(f'V-measure: {np.array(v_meas_d).mean():.4f}')
```

```
Métricas de avaliação do DBSCAN variando eps
Coeficiente de silhouette: 0.2598
Homogeneity: 0.1250
Completeness: 0.0745
V-measure: 0.0758
```

Observa-se que o coefficiente de silhouette melhorou, indo de -0,18 para 0,25. Já as outras métricas, não houve melhoras de resultado. Logo a variação do *eps* interferiu apenas no coeficiente de silhouette.