Deborah Karoline de Matos Vaz RM: 334964

Análise de dos atributos dos Diamantes utilizando K-Means e Árvore de Decisão

BARUERI

2020

SUMÁRIO

CAPÍTULO 1 – BASE DE DADOS

1.1. DEFINIÇÃO

Para a análise de negócio, utilizou-se a base de dados com atributos do Diamante, contendo cor, preço, claridade, corte etc. que pode ser encontrada no Kaggle (https://www.kaggle.com/shivam2503/diamonds#diamonds.csv).

Descrição dos campos:

- Carat: Peso em quilates do diamante;
- Cut: Descreve a qualidade do corte do diamante;
- Color: Cor do diamante, sendo D o melhor e J o pior;
- Clarity: Clareza;
- Depth: Altura do diamante;
- Table: Largura da tabela do diamante expressa em porcentagem do seu diâmetro médio;
- Price: Preço do diamante
- X: Altura em mm
- Y: Largura em mm
- Z: Comprimento em mm

1.2. PREPARAÇÃO

A. Realiza-se a importação da Base:

```
import pandas as pd
diamonds = pd.read_csv('/content/drive/My Drive/Interligencia_Artificial/TRAB/diamonds.csv')
diamonds
```

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...	...	...
53935	53936	0.72	Ideal	D	SI1	60.8	57.0	2757	5.75	5.76	3.50
53936	53937	0.72	Good	D	SI1	63.1	55.0	2757	5.69	5.75	3.61
53937	53938	0.70	Very Good	D	SI1	62.8	60.0	2757	5.66	5.68	3.56
53938	53939	0.86	Premium	H	SI2	61.0	58.0	2757	6.15	6.12	3.74
53939	53940	0.75	Ideal	D	SI2	62.2	55.0	2757	5.83	5.87	3.64

53940 rows × 11 columns

B. Transformação dos dados para numéricos:

```
diamonds['color'] = diamonds['color'].replace(['J','I','H','G','F','E','D'], [0,1,2,3,4,5,6])
diamonds['cut'] = diamonds['cut'].replace(['Fair','Good','Premium','Very Good','Ideal'], [0,1,2,3,4])
diamonds['clarity'] = diamonds['clarity'].replace(['I1','SI2','SI1','VS2','VS1','VVS2','VVS1','IF'], [0,1,2,3,4,5,6,7])
```

C. Normalização dos dados:

As colunas depth e table estavam em percentual, sendo assim bastou dividir por 100. Os demais dados foram normalizados [0,1], por meio da fórmula:

$$col_{norm} = \frac{datatable['column'] - datatable['column'].\min()}{datatable['column'].\max() - datatable['column'].\min()}$$

Normalização dos dados

```
[ ] diamonds['depth_norm'] = diamonds['depth']/100
    diamonds['table_norm'] = diamonds['table']/100
    diamonds['carat_norm'] = (diamonds['carat'] - diamonds['carat'].min())/(diamonds['carat'].max() - diamonds['carat'].min())
    diamonds['price_norm'] = (diamonds['price'] - diamonds['price'].min())/(diamonds['price'].max() - diamonds['price'].min())
    diamonds['x_norm'] = (diamonds['x'] - diamonds['x'].min())/(diamonds['x'].max() - diamonds['x'].min())
    diamonds['y_norm'] = (diamonds['y'] - diamonds['y'].min())/(diamonds['y'].max() - diamonds['y'].min())
    diamonds['z_norm'] = (diamonds['z'] - diamonds['z'].min())/(diamonds['z'].max() - diamonds['z'].min())
```

D. Criação de um novo datatable somente com as colunas alteradas:

Criar outra base com as novas colunas

```
[ ] diamonds2 = diamonds[['carat_norm','color','clarity','depth_norm','table_norm','price_norm','x_norm','y_norm','z_norm','cut']]
    diamonds2
```

	carat_norm	color	clarity	depth_norm	table_norm	price_norm	x_norm	y_norm	z_norm	cut
0	0.006237	5	1	0.615	0.55	0.000000	0.367784	0.067572	0.076415	4
1	0.002079	5	2	0.598	0.61	0.000000	0.362197	0.065195	0.072642	2
2	0.006237	5	4	0.569	0.65	0.000054	0.377095	0.069100	0.072642	1
3	0.018711	1	3	0.624	0.58	0.000433	0.391061	0.071817	0.082704	2
4	0.022869	0	1	0.633	0.58	0.000487	0.404097	0.073854	0.086478	1
...	...	...	...	...	...	...	...	...	...	...
53935	0.108108	6	2	0.608	0.57	0.131427	0.535382	0.097793	0.110063	4
53936	0.108108	6	2	0.631	0.55	0.131427	0.529795	0.097623	0.113522	1
53937	0.103950	6	2	0.628	0.60	0.131427	0.527002	0.096435	0.111950	3
53938	0.137214	2	1	0.610	0.58	0.131427	0.572626	0.103905	0.117610	2
53939	0.114345	6	1	0.622	0.55	0.131427	0.542831	0.099660	0.114465	4

53940 rows × 10 columns

CAPÍTULO 2 – APLICANDO OS MODELOS

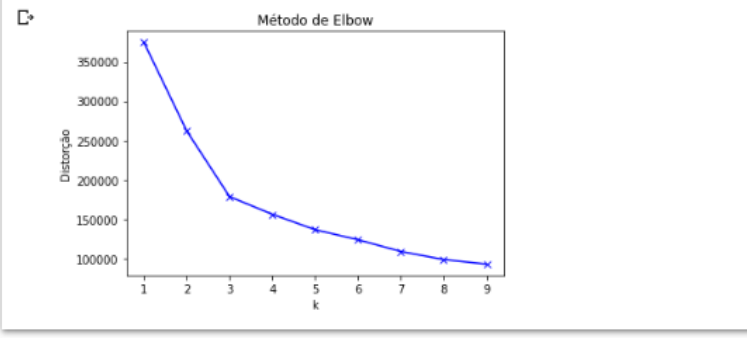
2.1. K-MEANS

A. Definição da quantidade de grupos de acordo com o método elbow (Observa-se de 3 é a quantidade ideal de grupos a ser criada):

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# k means determine k
distortions = []
K = range(1,10)
for k in K:
    kmeanModel = KMeans(n_clusters=k)
    kmeanModel.fit(diamonds2)
    distortions.append(kmeanModel.inertia_)

# Plot the elbow
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distorção')
plt.title('Método de Elbow')
plt.show()
```



B. Treinamento da base, definindo em 3 clusters:

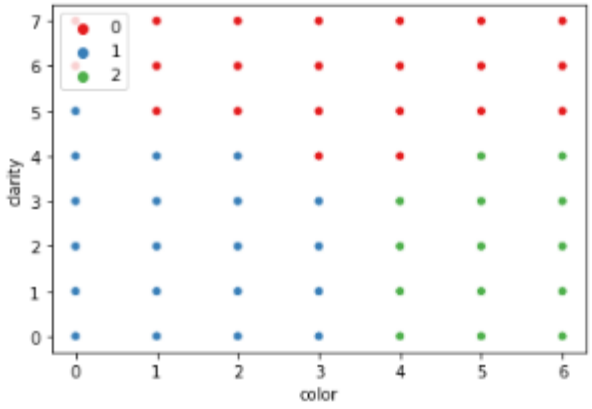
```
k = KMeans(n_clusters=3)
k.fit(diamonds2)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

C. Definiremos os clusters com base na cor e na clareza do diamante:

```
[73] import seaborn as sns
     sns.scatterplot(diamonds2['color'],diamonds2['clarity'],
                     hue=k.labels_, palette=sns.color_palette('Set1',3))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f20b0e12c88>
```

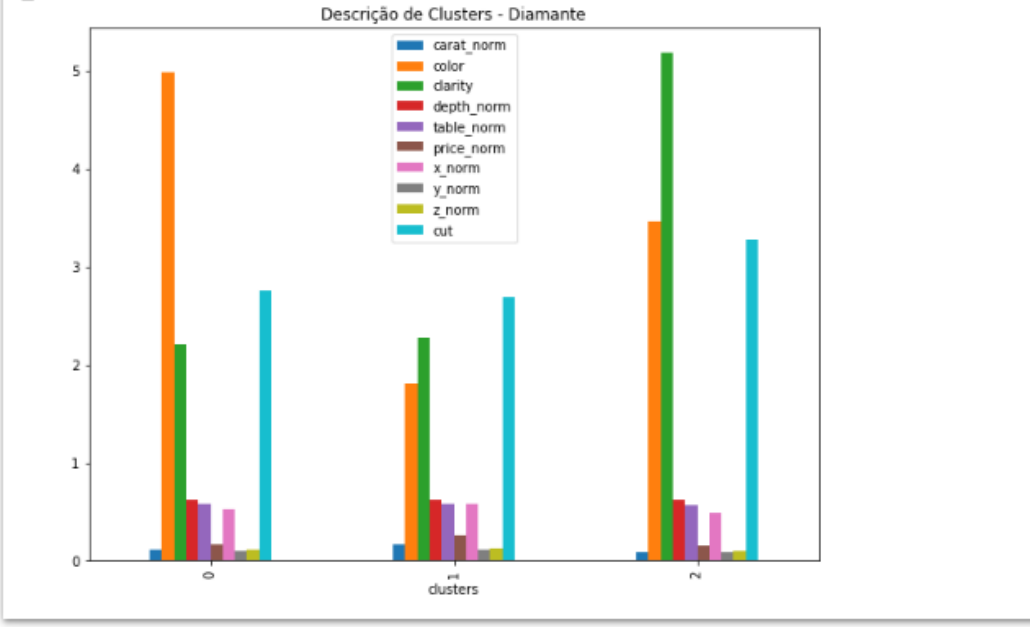


Neste caso temos 3 clusters onde, o primeiro tem maior clareza, independente da cor, o segundo tem cor de mediana a ruim e o último é composto pelas melhores cores.

D. Descrição das características de cada grupo de acordo com os atributos da base:

```
diamonds2['clusters'] = k.fit_predict(diamonds2)
diamonds2.groupby("clusters").aggregate("mean").plot.bar(figsize=(10,7.5))
plt.title("Descrição de Clusters - Diamante")
```

```
Text(0.5, 1.0, 'Descrição de Clusters - Diamante')
```



2.2. ÁRVORE DE DECISÃO

A. Divisão da base entre treino e teste:

Neste ponto dividimos a base em 2:

- diamonds2_train: 70% da base, que será utilizada para treino do modelo (fit)
- diamonds2_test: 30% da base que será utilizada para validação da árvore gerada.

```
from sklearn.model_selection import train_test_split

diamonds2_train, diamonds2_test = train_test_split(diamonds2)
```

B. Treino do modelo:

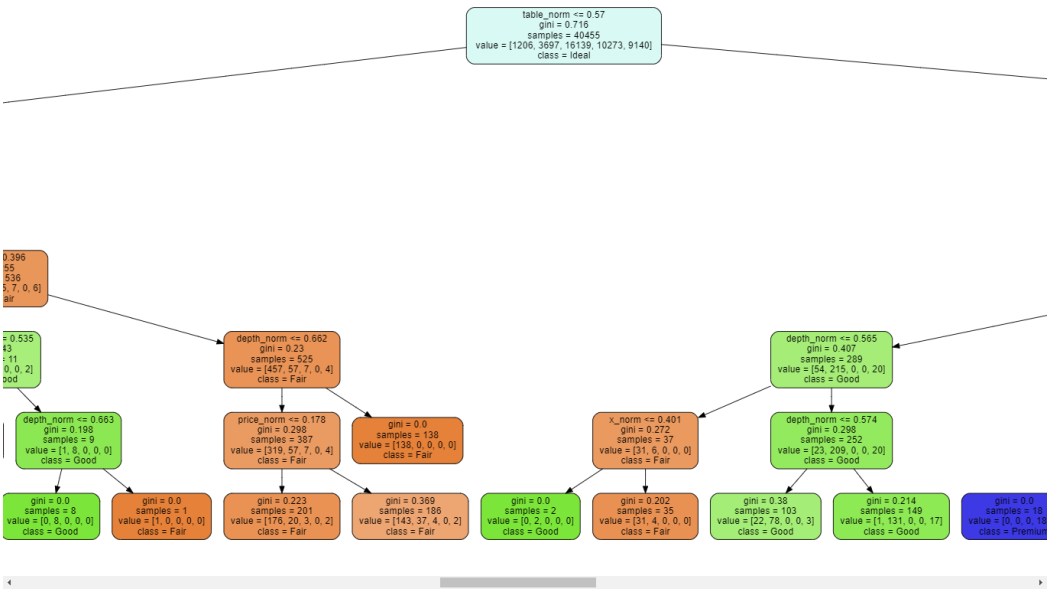
```
[19] from sklearn.tree import DecisionTreeClassifier

     arv = DecisionTreeClassifier(max_depth=6)
     arv.fit(diamonds2_train[['carat_norm','color','clarity','depth_norm','table_norm','price_norm','x_norm','y_norm','z_norm']], diamonds2_train['cut'])
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=6, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

C. Com base no treino anterior, plotamos a árvore:

```
from sklearn.tree import export_graphviz
from graphviz import Source
dot_data = export_graphviz(arv, out_file=None,
                           filled=True, rounded = True,
                           feature_names=['carat_norm','color','clarity','depth_norm','table_norm','price_norm','x_norm','y_norm','z_norm'],
                           class_names=['Fair','Good','Ideal','Premium','Very Good'])
grafico = Source(dot_data)
grafico
```



D. Aplicamos o resultado da árvore decisão à base de teste para medir a acuracidade do modelo:

Taxa de Acerto (Acuracidade): [Acertos/ Total]

```
from sklearn.model_selection import cross_val_score

score_dm = cross_val_score(arv, diamonds2_test[['carat_norm','color','clarity','depth_norm','table_norm','price_norm','x_norm','y_norm','z_norm']],
                           diamonds2_test['cut'], scoring = 'accuracy')
print(score_dm.mean())
```

```
0.7250278086763069
```

Esse script realiza o seguinte cálculo:

$$Acuracidade = \frac{Quantidade\ de\ acertos}{Total\ de\ registros}$$

Com isso, verificamos que o modelo tem uma assertividade de 72,5%.