**Exploração de dados**

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

teste\_csv = pd.read\_csv('car\_rental\_data.csv')

teste\_df = pd.DataFrame(teste\_csv)

remov\_dupli = teste\_df.drop\_duplicates(keep="last")

remov\_vlr\_null = teste\_df.dropna(thresh=2)

duplicatas = teste\_csv.duplicated()

X\_simple = remov\_vlr\_null[['purchase\_value']]

y = remov\_vlr\_null['selling\_value']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_simple, y, test\_size=0.2, random\_state=42)

model\_regression = LinearRegression()

model\_regression.fit(X\_train, y\_train)

y\_pred = model\_regression.predict(X\_test)

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

mae\_percentage = (mae / y\_test.mean()) \* 100

mse\_percentage = (mse / y\_test.mean()) \* 100

print("Erro Médio Absoluto (MAE):", mae)

print("Erro Quadrático Médio (MSE):", mse)

print("Porcentagem de Erro MAE:", mae\_percentage, "%")

print("Porcentagem de Erro MSE:", mse\_percentage, "%")

plt.figure(figsize=(10, 6))

plt.scatter(X\_test, y\_test, color='blue', label='Valor Real', alpha=0.5)

plt.plot(X\_test, y\_pred, color='red', label='Valor Previsto', alpha=0.5)

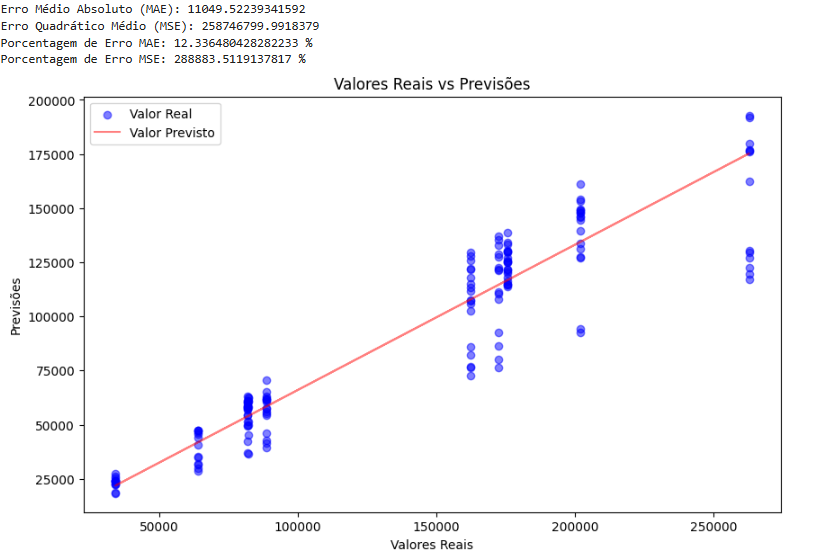
plt.xlabel('Valores Reais')

plt.ylabel('Previsões')

plt.title('Valores Reais vs Previsões')

plt.legend()

plt.show()



**Implementação e aprendizado**

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix

import seaborn as sns

import matplotlib.pyplot as plt

teste\_csv = pd.read\_csv('car\_rental\_data.csv')

teste\_df = pd.DataFrame(teste\_csv)

remov\_dupli = teste\_df.drop\_duplicates(keep="last")

remov\_vlr\_null = teste\_df.dropna(thresh=2)

duplicatas = teste\_csv.duplicated()

y\_classification = pd.cut(remov\_vlr\_null['purchase\_value'], bins=3, labels=['baixo', 'médio', 'alto'])

X\_simple =remov\_vlr\_null [['selling\_value']]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_simple, y\_classification, test\_size=0.2, random\_state=42)

model\_knn = KNeighborsClassifier(n\_neighbors=5)

model\_knn.fit(X\_train, y\_train)

y\_pred = model\_knn.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average='weighted')

recall = recall\_score(y\_test, y\_pred, average='weighted')

f1 = f1\_score(y\_test, y\_pred, average='weighted')

print("Acurácia:", accuracy)

print("Precisão:", precision)

print("Recall:", recall)

print("F1-score:", f1)

cm = confusion\_matrix(y\_test, y\_pred)

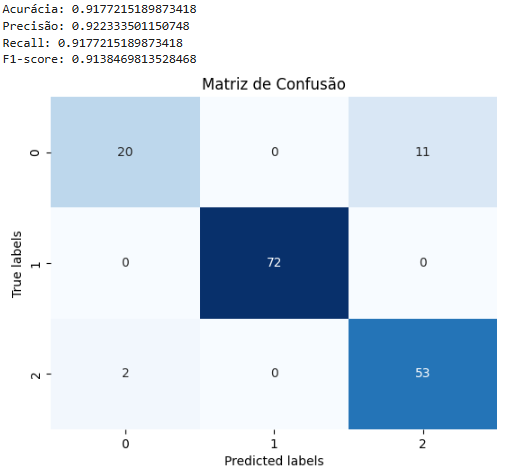
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)

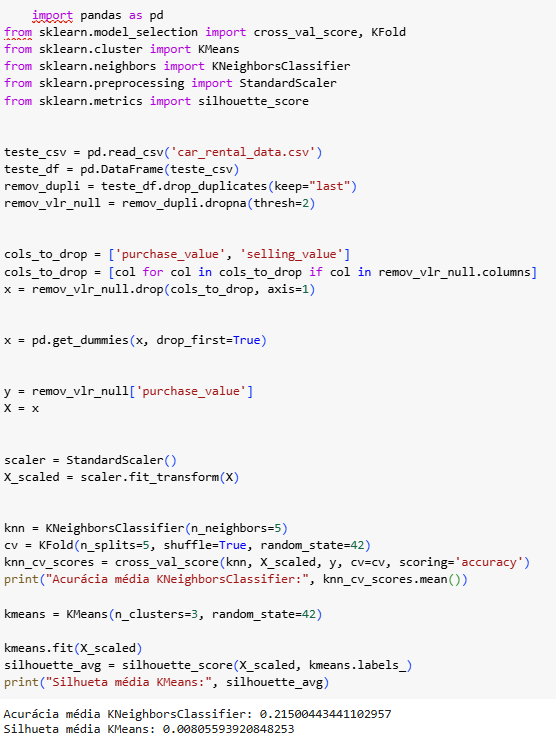
plt.xlabel('Predicted labels')

plt.ylabel('True labels')

plt.title('Matriz de Confusão')

plt.show()





**Otimização**

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

import numpy as np

teste\_csv = pd.read\_csv('car\_rental\_data.csv')

teste\_df = pd.DataFrame(teste\_csv)

remov\_dupli = teste\_df.drop\_duplicates(keep="last")

remov\_vlr\_null = remov\_dupli.dropna(thresh=2)

print("Colunas disponíveis:", remov\_vlr\_null.columns)

cols\_to\_drop = ['purchase\_value', 'selling\_value']

cols\_to\_drop = [col for col in cols\_to\_drop if col in remov\_vlr\_null.columns]

x = remov\_vlr\_null.drop(cols\_to\_drop, axis=1)

x = pd.get\_dummies(x, drop\_first=True)

scaler = StandardScaler()

x\_scaled = scaler.fit\_transform(x)

n\_cluster = 3

kmeans = KMeans(n\_clusters=n\_cluster, random\_state=42)

kmeans.fit(x\_scaled)

remov\_vlr\_null['cluster'] = kmeans.labels\_

print('Centro dos Clusters')

print(scaler.inverse\_transform(kmeans.cluster\_centers\_))

plt.figure(figsize=(10,6))

for cluster in range (n\_cluster):

    cluster\_data = remov\_vlr\_null[remov\_vlr\_null['cluster'] == cluster]

    plt.scatter(cluster\_data['purchase\_value'],cluster\_data['segment'],label=f'cluster {cluster}')

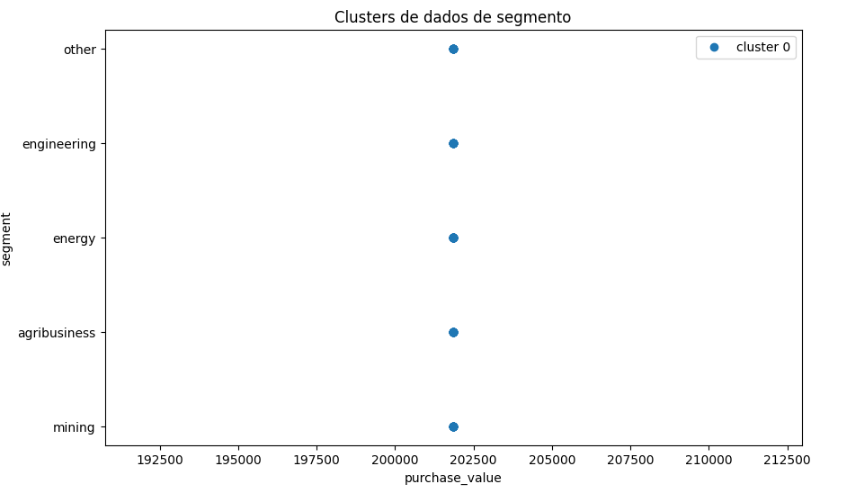
plt.title ('Clusters de dados de segmento')

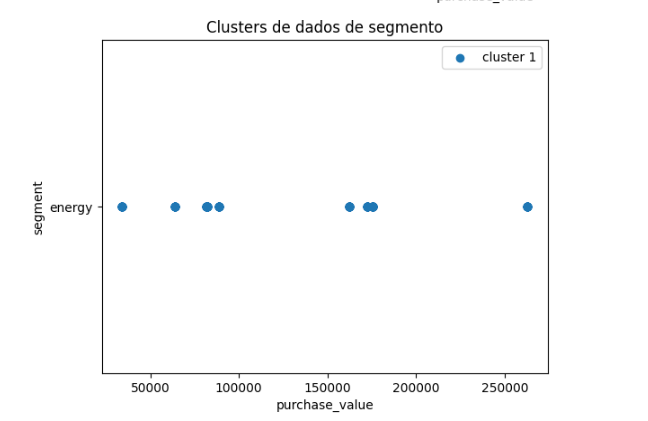
    plt.xlabel('purchase\_value')

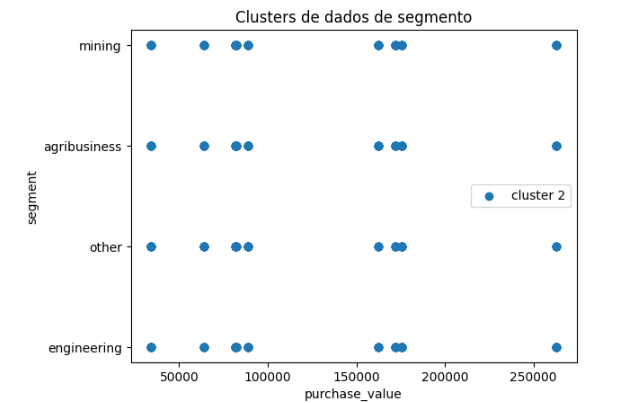
    plt.ylabel('segment')

    plt.legend()

    plt.show()

****

****

****