

Tarefa-01. MO432.

Patrick de Carvalho Tavares Rezende Ferreira - 175480

In [19]:

```
from matplotlib import pyplot as plt
from numpy import hstack, add
from pandas import read_csv, get_dummies
from sklearn.decomposition import PCA
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, make_scorer
from sklearn.model_selection import ShuffleSplit, cross_validate
from sklearn.preprocessing import StandardScaler
```

Item 1 - Input de dados CSV

Primeiramente, lemos o arquivo CSV com os dados categóricos e não categóricos a serem trabalhados.

In [20]:

```
data = read_csv("car-data.csv")
```

Item 2 - Dados categóricos para "dummy variables"

Abaixo fazemos a impressão de alguns dos atributos originais do arquivo CSV e dos atributos após a conversão de atributos categóricos em "dummy variables". O tamanho original também é exibido abaixo de cada impressão, sendo de 301 amostras e 9 features para o arquivo original, e 301 amostras com 110 features após a conversão das variáveis categóricas em "dummy variables".

In [21]:

```
print(data)
print("\n\n===== \n\n")
print(get_dummies(data))
get_dummies(data).to_csv("car-with-dummies.csv")

# OneHot encoding para converter vriaveis ctegoricas em dummy variables.
data = get_dummies(data)

# Separamos dados de entrada dos dados de saida.
# A coluna a remover ("i") eh a do selling_price.
i = 1
X_data = hstack((data.to_numpy()[:, 0:i], data.to_numpy()[:, i + 1:]))
# X_data = X_data.T
y_data = data.to_numpy()[:, i]
```

```

Car_Name Year Selling_Price Present_Price Kms_Driven Fuel_Ty
\
0   ritz 2014      3.35      5.59      27000   Petr
1   sx4  2013      4.75      9.54      43000   Dies
2   ciaz 2017      7.25      9.85       6900   Petr
3   wagon r 2011      2.85      4.15       5200   Petr
4   swift 2014      4.60      6.87      42450   Dies
..   ...   ...      ...      ...      ...
...
296  city 2016      9.50     11.60     33988   Dies
297  brio 2015      4.00      5.90     60000   Petr
298  city 2009      3.35     11.00     87934   Petr
299  city 2017     11.50     12.50       9000   Dies
300  brio 2016      5.30      5.90       5464   Petr
ol

```

```

Seller_Type Transmission Owner
0   Dealer      Manual      0
1   Dealer      Manual      0
2   Dealer      Manual      0
3   Dealer      Manual      0
4   Dealer      Manual      0
..   ...      ...      ...
296  Dealer      Manual      0
297  Dealer      Manual      0
298  Dealer      Manual      0
299  Dealer      Manual      0
300  Dealer      Manual      0

```

[301 rows x 9 columns]

=====

```

Year Selling_Price Present_Price Kms_Driven Owner Car_Name
\
_800 2014      3.35      5.59      27000      0
0
1   2013      4.75      9.54      43000      0
0
2   2017      7.25      9.85       6900      0
0
3   2011      2.85      4.15       5200      0
0
4   2014      4.60      6.87      42450      0
0
..   ...      ...      ...      ...
...
296 2016      9.50     11.60     33988      0
0
297 2015      4.00      5.90     60000      0

```

```

0
298 2009          3.35          11.00          87934          0
0
299 2017          11.50          12.50          9000          0
0
300 2016          5.30           5.90          5464          0
0

```

```

Car_Name_Activa 3g  Car_Name_Activa 4g  Car_Name_Bajaj  ct 100
\
0          0          0          0
1          0          0          0
2          0          0          0
3          0          0          0
4          0          0          0
..          ...          ...          ...
296        0          0          0
297        0          0          0
298        0          0          0
299        0          0          0
300        0          0          0

```

```

Car_Name_Bajaj Avenger 150  ...  Car_Name_vitara brezza  \
0          0  ...          0
1          0  ...          0
2          0  ...          0
3          0  ...          0
4          0  ...          0
..          ...  ...          ...
296        0  ...          0
297        0  ...          0
298        0  ...          0
299        0  ...          0
300        0  ...          0

```

```

Car_Name_wagon r  Car_Name_xcent  Fuel_Type_CNG  Fuel_Type_Dies
el \
0          0          0          0
0
1          0          0          0
1
2          0          0          0
0
3          1          0          0
0
4          0          0          0
1
..          ...          ...          ...
...
296        0          0          0
1
297        0          0          0
0
298        0          0          0
0
299        0          0          0
1
300        0          0          0
0

```

```

Fuel_Type_Petrol  Seller_Type_Dealer  Seller_Type_Individual  \

```

0	1	1	0
1	0	1	0
2	1	1	0
3	1	1	0
4	0	1	0
..
296	0	1	0
297	1	1	0
298	1	1	0
299	0	1	0
300	1	1	0

	Transmission_Automatic	Transmission_Manual
0	0	1
1	0	1
2	0	1
3	0	1
4	0	1
..
296	0	1
297	0	1
298	0	1
299	0	1
300	0	1

[301 rows x 110 columns]



Item 3 - Centering e Scaling dos dados

Abaixo fazemos a centralização dos dados em função da média de cada coluna de features e o escalamento de cada coluna também em função do próprio desvio padrão, de forma que todas as entradas estejam em função de quantos desvios padrão elas representam para cada features. Isto ajuda a regressão a ponderar o peso que dá sobre as variações em cada entrada.

In [22]:

```

scaler = StandardScaler()
scaler.fit(X_data)

print("\nMédias do Scaler em cada coluna: \n", scaler.mean_)
print('\nDados centrados e "escalados": \n', scaler.transform(X_data))

X_data_scaled = scaler.transform(X_data)

```

Médias do Scaler em cada coluna:

```

[2.01362791e+03 7.62847176e+00 3.69472060e+04 4.31893688e-02
 3.32225914e-03 6.64451827e-03 3.32225914e-03 3.32225914e-03
 3.32225914e-03 3.32225914e-03 9.96677741e-03 6.64451827e-03
 3.32225914e-03 3.32225914e-03 6.64451827e-03 3.32225914e-03
 3.32225914e-03 3.32225914e-03 1.32890365e-02 6.64451827e-03
 9.96677741e-03 3.32225914e-03 3.32225914e-03 3.32225914e-03
 6.64451827e-03 3.32225914e-03 3.32225914e-03 3.32225914e-03
 3.32225914e-03 6.64451827e-03 3.32225914e-03 3.32225914e-03
 6.64451827e-03 3.32225914e-03 3.32225914e-03 6.64451827e-03
 9.96677741e-03 6.64451827e-03 3.32225914e-03 3.32225914e-03
 6.64451827e-03 6.64451827e-03 3.32225914e-03 6.64451827e-03
 3.32225914e-03 3.32225914e-03 6.64451827e-03 3.32225914e-03
 3.32225914e-03 3.32225914e-03 2.32558140e-02 6.64451827e-03
 1.32890365e-02 9.96677741e-03 3.32225914e-03 9.96677741e-03
 6.64451827e-03 3.32225914e-03 3.32225914e-03 3.32225914e-03
 3.32225914e-03 6.64451827e-03 3.32225914e-03 3.32225914e-03
 9.96677741e-03 3.32225914e-03 3.32225914e-03 1.66112957e-02
 2.32558140e-02 3.32225914e-03 3.32225914e-02 3.32225914e-03
 2.99003322e-02 8.63787375e-02 3.32225914e-03 5.31561462e-02
 9.96677741e-03 1.32890365e-02 6.64451827e-03 1.99335548e-02
 1.99335548e-02 9.96677741e-03 9.96677741e-03 3.32225914e-03
 1.32890365e-02 3.65448505e-02 2.65780731e-02 1.66112957e-02
 2.99003322e-02 3.32225914e-03 2.99003322e-02 2.32558140e-02
 3.32225914e-03 3.32225914e-03 1.32890365e-02 3.32225914e-03
 1.66112957e-02 1.99335548e-02 4.65116279e-02 3.32225914e-03
 1.32890365e-02 9.96677741e-03 6.64451827e-03 1.99335548e-01
 7.94019934e-01 6.47840532e-01 3.52159468e-01 1.32890365e-01
 8.67109635e-01]

```

Dados centrados e "escalados":

```

[[ 0.128897 -0.23621461 -0.25622446 ... -0.73728539 -0.39148015
  0.39148015]
 [-0.21751369 0.22150462 0.1559105 ... -0.73728539 -0.39148015
  0.39148015]
 [ 1.16812909 0.25742689 -0.77396901 ... -0.73728539 -0.39148015
  0.39148015]
 ...
 [-1.60315648 0.39068691 1.31334003 ... -0.73728539 -0.39148015
  0.39148015]
 [ 1.16812909 0.56450434 -0.7198763 ... -0.73728539 -0.39148015
  0.39148015]
 [ 0.8217184 -0.20029235 -0.81095812 ... -0.73728539 -0.39148015
  0.39148015]]

```

Item 4 - PCA

Abaixo realizamos a redução dos atributos de entrada através da técnica de "principal components analysis" (PCA). Fazemos este processo devido ao fato que a maior parte dos dados de entrada geralmente não contribui com informações de alta relevância para os dados de saída, e para datasets de tamanho relativamente pequeno como o que estamos lidando, uma quantidade elevada de coeficientes a ajustar poderia produzir um modelo sub ajustado, sem informação suficientemente grande para ajustar cada coeficiente.

Para explicar 90% da variância, o PCA exige que utilizemos 89 componentes de entrada, mas este ainda é um valor muito alto que reduz coeficientes lineares altos ao final da regressão devido à pequena quantidade de dados para esta quantidade de atributos.

Graficamente porém é possível perceber que a curva de relevância para a variância em relação a cada componente tem seu "joelho" por volta de 10 componentes no "scree plot" (gráficos exibidos abaixo). Realizamos o PCA então com 15 componentes, para utilizarmos apenas as componentes de maior relevância.

In [23]:

```
pca = PCA(n_components=0.9, svd_solver="full")
pca.fit(X_data_scaled)

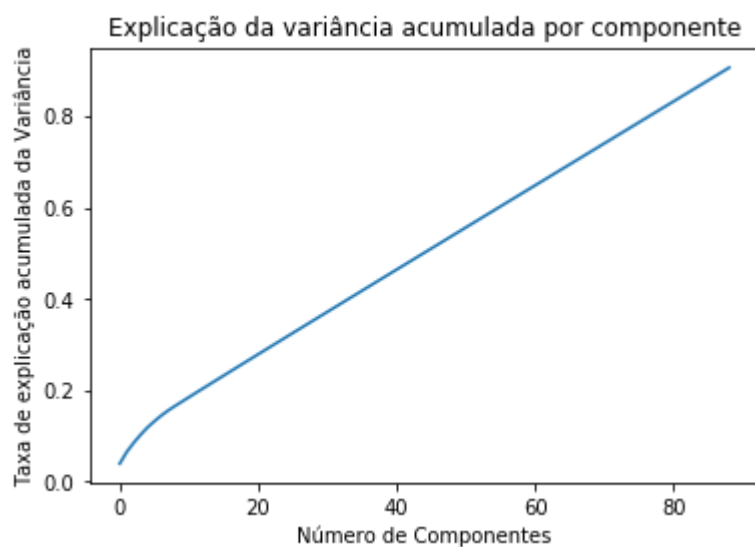
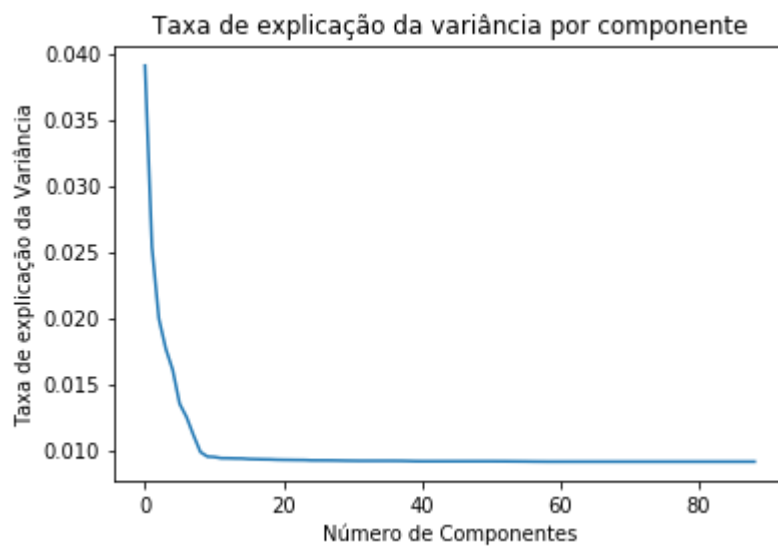
print("Números de componentes do PCA para 90% de explicação da variância: ", pca
      .n_components_)

plt.plot(pca.explained_variance_ratio_)
plt.title('Taxa de explicação da variância por componente')
plt.ylabel('Taxa de explicação da Variância')
plt.xlabel('Número de Componentes')
plt.show()

plt.plot(add.accumulate(pca.explained_variance_ratio_))
plt.title('Explicação da variância acumulada por componente')
plt.ylabel('Taxa de explicação acumulada da Variância')
plt.xlabel('Número de Componentes')
plt.show()

# Obtido graficamente
pca = PCA(n_components=15)
pca.fit(X_data_scaled)
X_data_pca = pca.transform(X_data_scaled)
```


Números de componentes do PCA para 90% de explicação da variância:
89



Item 5 - Regressão Linear

Fazemos agora a regressão linear em função dos dados anteriormente tratados. Utilizamos o shuffle splitter do sklearn para repartir os dados em arranjos de 70/30 para treino e teste, respectivamente, e em 5 embaralhamentos diferentes. A validação cruzada também do sklearn é configurada para avaliar a RMSE e a MAE da regressão, cujos valores individuais para cada uma das 5 repetições e também as médias são exibidas abaixo.

Utilizando o PCA com 15 componentes, o erro RMSE médio foi de 2,47 dólares e o MAE foi de 1,60 dólares (supondo que o valor de saída em "y" eram dólares). Isto é um resultado que demonstra a utilidade do PCA, uma vez que a regressão com 89 componentes retornava erros acima de 10^6 .

In [24]:

```
shuffle_splitter = ShuffleSplit(n_splits=5, test_size=0.3, random_state=1234)
regressor = LinearRegression()
cv_results = cross_validate(regressor, X_data_pca, y_data,
                             cv=shuffle_splitter,
                             scoring={"MSE": make_scorer(mean_squared_error, greater_is_better=False),
                                       "MAE": make_scorer(mean_absolute_error, greater_is_better=False)})

# print(cv_results.keys())
print("\nRMSE para cada repetição: \n", (-cv_results["test_MSE"]) ** (1 / 2))
print("\nMAE para cada repetição: \n", -cv_results["test_MAE"])

print("\nRMSE médio: ", ((-cv_results["test_MSE"]) ** (1 / 2)).mean())
print("\nMAE médio: ", (-cv_results["test_MAE"]).mean())
```

RMSE para cada repetição:
[2.56438532 2.69903945 2.32235582 1.93230384 2.84091571]

MAE para cada repetição:
[1.74302273 1.66889463 1.65069842 1.29538618 1.62432251]

RMSE médio: 2.471800027566019

MAE médio: 1.596464895253393