Regressão Linear (Lab)

Regressão Linear

Caso: **Como estimar a emissão de gases CO2 de um motor?**

Empregue os dados da URL:
https://meusite.mackenzie.br/rogerio/TIC/FuelConsumptionCo2.csv

Neste Lab você vai estimar as emissões de CO2 de modelos de veículos a partir de suas características empregando modelos de regressão linear e avaliar esses modelos.

`FuelConsumption.csv`:

`FuelConsumption.csv`, contém informações do consumo de combustível, outras características dos modelos e as emissões de dióxido de carbono para vários novos veículos comerciais leves no Canadá.

- **MODELYEAR** ex. 2014
- **MAKE** ex. Acura
- **MODEL** ex. ILX
- **VEHICLE CLASS** ex. SUV
- **ENGINE SIZE** ex. 4.7
- **CYLINDERS** ex. 6
- **TRANSMISSION** ex. A6
- **FUEL CONSUMPTION in CITY(L/100 km)** ex. 9.9
- **FUEL CONSUMPTION in HWY (L/100 km)** ex. 8.9
- **FUEL CONSUMPTION COMB (L/100 km)** ex. 9.2
- **CO2 EMISSIONS (g/km)** ex. 182

Regressão Linear. Vamos lembrar?

Um modelo linear aproxima o valor de variável objetivo Y a partir de uma combinação linear das variáveis preditoras X.

$$\widehat{Y} = a_0 + a_1 X_1 + a_2 X_2 + \ldots + a_n X_n$$

A cada variável preditora corresponde um coeficiente a_n, havendo um coeficiente independente que corresponte ao valor de $\widehat{Y}$ para $X = 0$ (*intercept*).

Uma regressão linear simples

Vamos começar com uma regressão simples de valores aleatórios apenas para você se familiarizar com a construção do modelo.

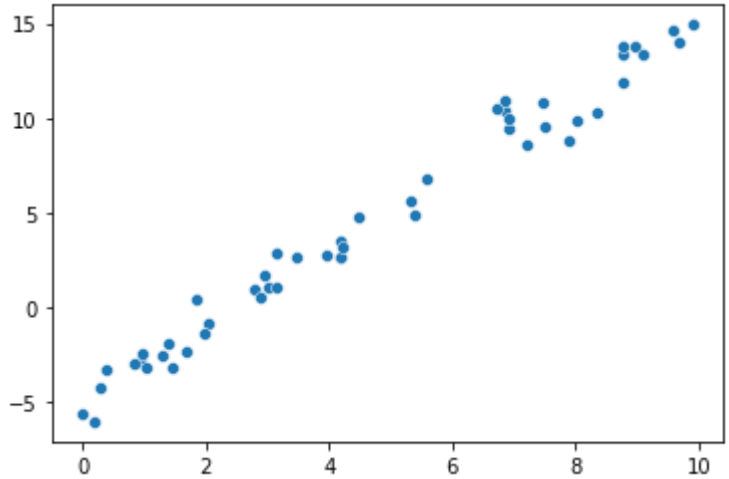
```
import pandas                   as pd
import numpy                    as np
import matplotlib.pyplot        as plt
import seaborn                  as sns
import statsmodels.formula.api  as sm
import warnings
warnings.filterwarnings("ignore")
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarnir
  import pandas.util.testing as tm
```

Gerando uma amostra de 50 valores "aleatórios" a partir de uma função linear.

```
rng = np.random.RandomState(1)
x = 10 * rng.rand(50)
y = 2 * x - 5 + rng.randn(50)
sns.scatterplot(x, y)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9d00d543c8>
```



Podemos ainda traçar uma linha de aproximação linear com a função `sns.lmplot()`.

```
df = pd.DataFrame({'x':x,'y':y})
sns.lmplot('x','y',data=df)
plt.show()

print(df)
```

Construindo o modelo linear, `sm.ols(formula = , data= )`

Um conjunto de dados é informado e o parâmetro `formula` indica as variáveis objetivo e preditoras.

```
formula = 'y ~ x'
```

significa

$$y \leftarrow x$$

para um modelo

$$\hat{y} = a_0 + b_1 x$$

```
     2   0.001144   -5.668959
model = sm.ols(formula='y ~ x', data=df)

     6   1.862602    0.385006
result = model.fit()
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.979
Model:                            OLS   Adj. R-squared:                  0.979
Method:                 Least Squares   F-statistic:                     2246.
Date:                Tue, 20 Oct 2020   Prob (F-statistic):           5.71e-42
Time:                        19:20:51   Log-Likelihood:                -65.935
No. Observations:                  50   AIC:                             135.9
Df Residuals:                      48   BIC:                             139.7
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -4.9986      0.239    -20.948      0.000      -5.478      -4.519
x              2.0272      0.043     47.397      0.000       1.941       2.113
==============================================================================
Omnibus:                        0.058   Durbin-Watson:                   1.590
Prob(Omnibus):                  0.971   Jarque-Bera (JB):                0.057
Skew:                           0.048   Prob(JB):                        0.972
Kurtosis:                       2.865   Cond. No.                         10.4
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
```

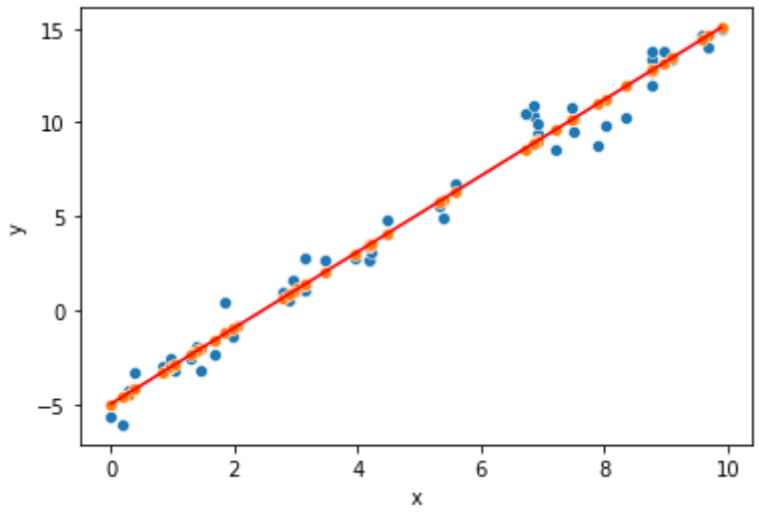
E desse modo a aproximação linear é dada por:

$$y = -4.9986 + 2.0272x$$

Empregamos então o modelo para estimar os valores de y, isto é $\bar{y}$, a partir do modelo linear. Podemos então comparar os valores de y e $\bar{y}$ (`predicted`).

```
df['predicted'] = result.predict(df.x)
```

```
sns.scatterplot('x','y',data=df)
sns.scatterplot('x','predicted',data=df)
sns.lineplot('x','predicted',data=df,color='red')
plt.show()
```



Caso: **Como estimar a emissão de gases CO2 de um motor?**

Empregue os dados da URL:
https://meusite.mackenzie.br/rogerio/TIC/FuelConsumptionCo2.csv

Exercício. Explorando os dados.

Acesse e explore os dados **FuelConsumptionCo2.csv**. Você pode querer verificar os imports, mas eles já devem estar feitos se você executou os código anteriores. Caso contrário refaça aqui.

```
# Insira aqui seu código

import pandas                 as pd
import numpy                  as np
import matplotlib.pyplot      as plt
import seaborn                as sns
import statsmodels.formula.api as sm
import warnings
```

```
warnings.filterwarnings("ignore")

df = pd.read_csv("https://meusite.mackenzie.br/rogerio/TIC/FuelConsumptionCo2.csv")
df.head()
```

	MODELYEAR	MAKE	MODEL	VEHICLECLASS	ENGINESIZE	CYLINDERS	TRANSMISSION
0	2014	ACURA	ILX	COMPACT	2.0	4	AS5
1	2014	ACURA	ILX	COMPACT	2.4	4	M6
2	2014	ACURA	ILX HYBRID	COMPACT	1.5	4	AV7
3	2014	ACURA	MDX 4WD	SUV - SMALL	3.5	6	AS6

Exercício. `CO2EMISSIONS`

Explore a relação das demais variáveis com `CO2EMISSIONS`

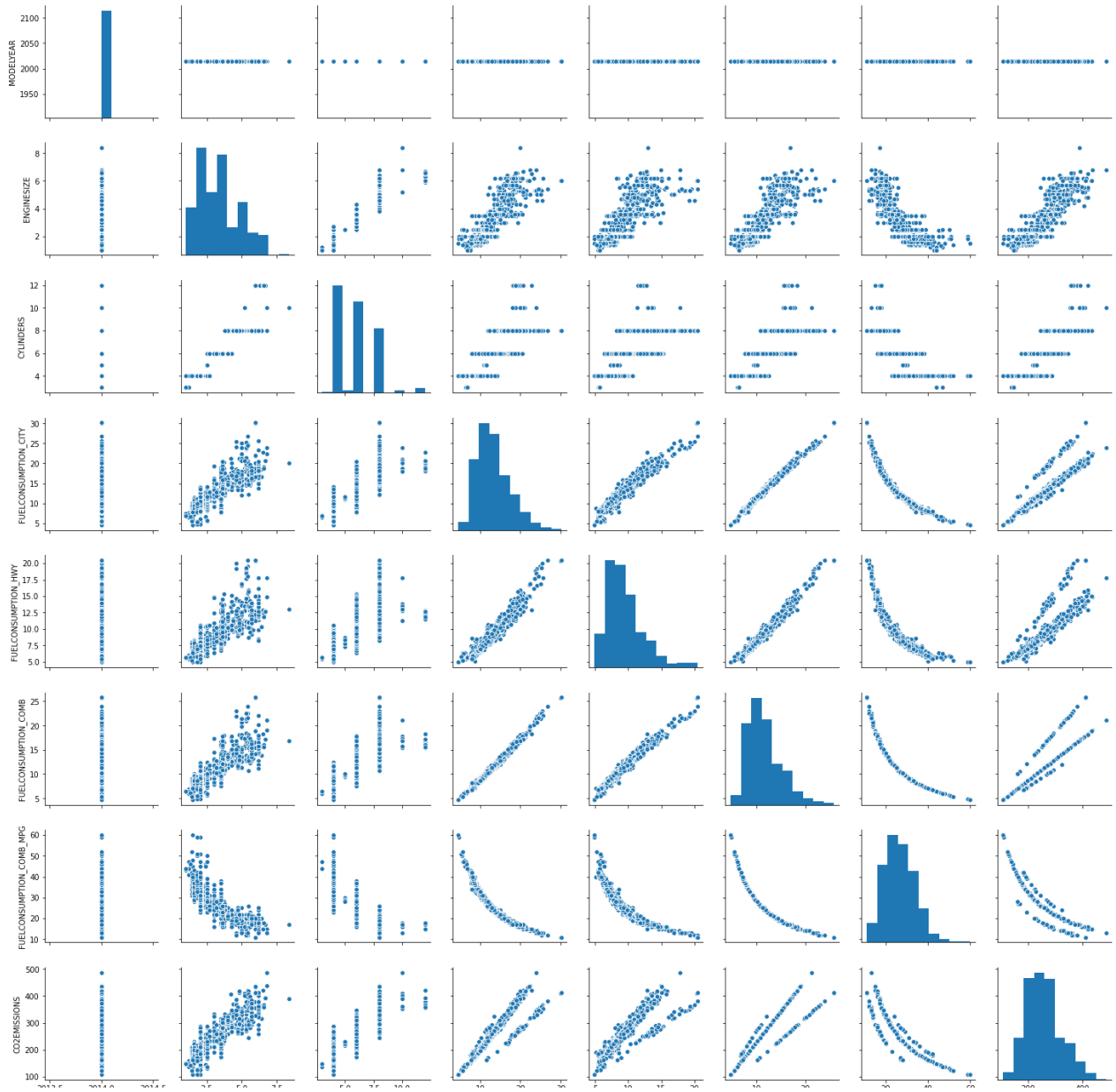
DICA:

Embora o `sns.scatterplot(x,y)` possa ser usado para verificar a relação de cada par de variáveis, o `sns.pairplot(df)` permite você criar vários `scatterplots` simultâneos.

```
# Insira aqui seu código

sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7fbf4d0f2ba8>
```



Exercício. FUELCONSUMPTION_COM X CO2EMISSIONS

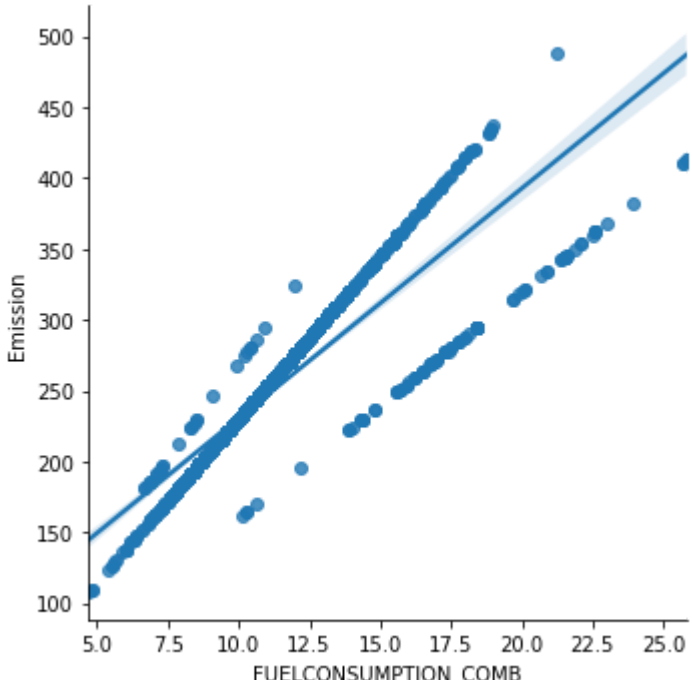
Explore a relação de FUELCONSUMPTION_COM com CO2EMISSIONS

DICA:

Embora o sns.scatterplot(x,y) possa ser usado para verificar a relação de cada par de variáveis, o sns.lmplot('x','y',data=df) vai ainda permitir você visualizar a linha de tendência dos dados.

```
# Insira aqui seu código

sns.lmplot('FUELCONSUMPTION_COMB', 'CO2EMISSIONS', data=df)
plt.xlabel("FUELCONSUMPTION_COMB")
plt.ylabel("Emission")
plt.show()
```



Exercício. Regressão Simples

Crie um modelo regressão para obter `CO2EMISSIONS` a partir de `FUELCONSUMPTION_COMB`

DICA:

Lembre-se

```
model = sm.ols(formula='y ~ x', data=df)
```

```
# Insira aqui seu código

model = sm.ols(formula='CO2EMISSIONS ~ FUELCONSUMPTION_COMB', data=df)
```

Exercício. Resultados

Verifique os resultados do modelo obtido

DICA:

Lembre-se

```
result = model.fit()
print(result.summary())
```

```
# Insira aqui seu código

result = model.fit()
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           CO2EMISSIONS   R-squared:                       0.796
Model:                            OLS   Adj. R-squared:                  0.796
Method:                 Least Squares   F-statistic:                     4153.
Date:                Tue, 20 Oct 2020   Prob (F-statistic):               0.00
Time:                        19:40:20   Log-Likelihood:                -5092.7
No. Observations:                1067   AIC:                         1.019e+04
Df Residuals:                    1065   BIC:                         1.020e+04
Df Model:                           1
Covariance Type:            nonrobust
============================================================================================
                           coef    std err          t      P>|t|      [0.025      0.9
--------------------------------------------------------------------------------------------
Intercept               68.3871      3.044     22.467      0.000      62.414      74
FUELCONSUMPTION_COMB    16.2200      0.252     64.443      0.000      15.726      16
==============================================================================
Omnibus:                      152.161   Durbin-Watson:                   2.195
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              240.073
Skew:                          -0.954   Prob(JB):                     7.39e-53
Kurtosis:                       4.325   Cond. No.                         42.2
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
```

Exercício. Modelo final

...insira no texto abaixo os coeficientes obtidos.

$$\widehat{CO2EMISSIONS} = 68.38 + 16.22 \times FUELCONSUMPTIONCOMB$$

Exercício. Predição

A partir do seu modelo empregue a função `result.predict(x)` para estimar a emissão de gases por veículos novos que apresentam consumo de combustível no valor de 4 e 28, e que não existem nos dados originais.

DICA:

A entrada x deve ser um dataframe no mesmo formato e nome dos atributos de entrada no modelo.

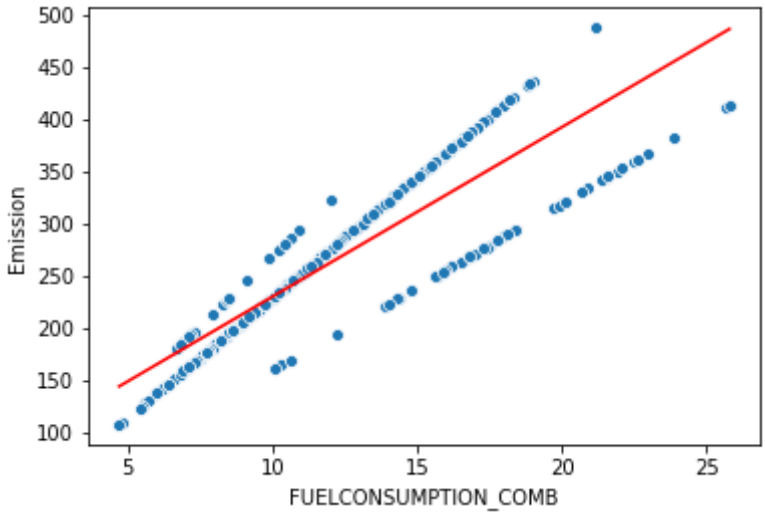
```
x = pd.DataFrame({'FUELCONSUMPTION_COMB': [4,28]})
result.predict(x)
```

```
0    133.267015
1    522.546301
dtype: float64
```

Exercício. Verificação Visual (**RESOLVIDO**)

```
df['predicted'] = result.predict()

sns.scatterplot('FUELCONSUMPTION_COMB', 'CO2EMISSIONS', data=df)
sns.lineplot('FUELCONSUMPTION_COMB', 'predicted', data=df, color='red')
plt.xlabel("FUELCONSUMPTION_COMB")
plt.ylabel("Emission")
plt.show()
```



Exercício. Tudo junto...

Coloque todo o código empregado para estimar as emissões de `FUELCONSUMPTION_COMB` aqui.

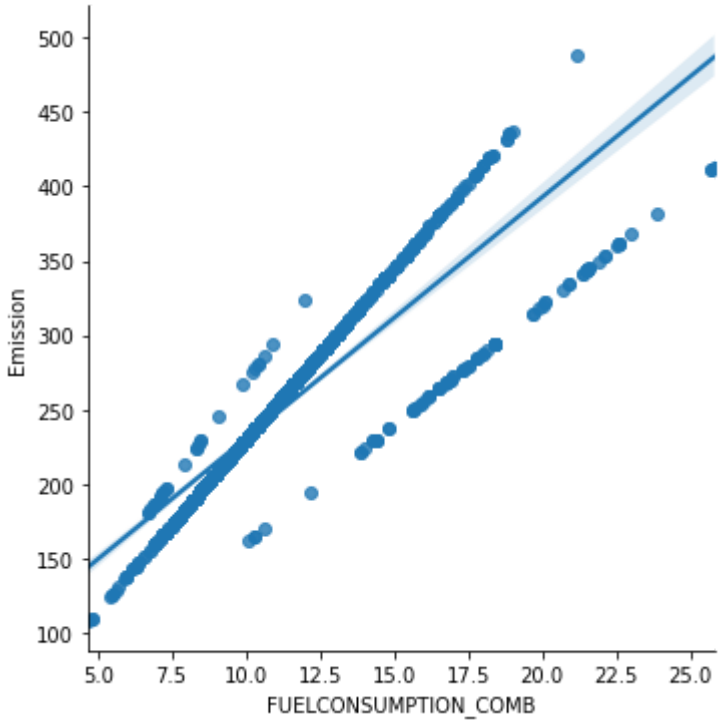
```
# Insira aqui seu código

# explora a relação linear (ou outra) dos dados
sns.lmplot('FUELCONSUMPTION_COMB', 'CO2EMISSIONS', data=df)
plt.xlabel("FUELCONSUMPTION_COMB")
plt.ylabel("Emission")
plt.show()

# define o modelo
model = sm.ols(formula='CO2EMISSIONS ~ FUELCONSUMPTION_COMB', data=df)

# calcula o modelo e mostra os resultados
result = model.fit()
print(result.summary())

# faz uma previsão
x = pd.DataFrame({'FUELCONSUMPTION_COMB': [4,28]})
print(result.predict(x))
```




```
                            OLS Regression Results
==============================================================================
Dep. Variable:           CO2EMISSIONS   R-squared:                       0.796
Model:                            OLS   Adj. R-squared:                  0.796
Method:                 Least Squares   F-statistic:                     4153.
Date:                Tue, 20 Oct 2020   Prob (F-statistic):               0.00
Time:                        19:43:24   Log-Likelihood:                -5092.7
No. Observations:                1067   AIC:                         1.019e+04
Df Residuals:                    1065   BIC:                         1.020e+04
Df Model:                           1
Covariance Type:            nonrobust
===========================================================================================
                              coef    std err          t      P>|t|      [0.025
-------------------------------------------------------------------------------------------
Intercept                  68.3871      3.044     22.467      0.000      62.414
FUELCONSUMPTION_COMB       16.2200      0.252     64.443      0.000      15.726
==============================================================================
Omnibus:                      152.161   Durbin-Watson:                   2.195
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              240.073
Skew:                          -0.954   Prob(JB):                     7.39e-53
Kurtosis:                       4.325   Cond. No.                         42.2
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
0    133.267015
1    522.546301
dtype: float64
```

Exercício. Regressão Múltipla

Faça agora um modelo de regressão múltipla para estimar as emissões de CO2 a partir de `FUELCONSUMPTION_COMB` e `ENGINESIZE`.

Faça então uma predição de emissões para um veículo com `FUELCONSUMPTION_COMB` = 10 e `ENGINESIZE` = 2.

```
# Insira aqui seu código

# define o modelo
model = sm.ols(formula='CO2EMISSIONS ~ FUELCONSUMPTION_COMB + ENGINESIZE', data=df)

# calcula o modelo e mostra os resultados
result = model.fit()
print(result.summary())

# faz uma previsão
x = pd.DataFrame({'FUELCONSUMPTION_COMB': [10], 'ENGINESIZE': [2]})
print(result.predict(x))
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           CO2EMISSIONS   R-squared:                       0.858
Model:                            OLS   Adj. R-squared:                  0.858
Method:                 Least Squares   F-statistic:                     3220.
Date:                Tue, 20 Oct 2020   Prob (F-statistic):               0.00
Time:                        19:50:57   Log-Likelihood:                -4898.4
No. Observations:                1067   AIC:                             9803.
Df Residuals:                    1064   BIC:                             9818.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                             coef    std err          t      P>|t|      [0.025      0.9
------------------------------------------------------------------------------
Intercept                 78.3068      2.579     30.360      0.000      73.246      83
FUELCONSUMPTION_COMB       9.7300      0.366     26.569      0.000       9.011      10
ENGINESIZE                19.4963      0.902     21.626      0.000      17.727      21
==============================================================================
Omnibus:                       60.372   Durbin-Watson:                   1.740
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               91.765
Skew:                          -0.462   Prob(JB):                     1.18e-20
Kurtosis:                       4.101   Cond. No.                         44.9
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
0    214.598964
dtype: float64
```

Solução: O modelo obtido é

$$\widehat{CO2EMISSIONS} = 78.30 + 19.49 \times ENGINESIZE + 9.73 \times FUELCONSU$$

E apresenta Coeficiente de Determinação e p-values

$$R^2 = 0.858$$
$$p - values < 0.05$$

melhor que os modelos unidimensionais.

Exercício. Regressão com Atributos Categóricos

Faça agora um modelo de Regressão Múltipla empregando o atributo categórico `VEHICLECLASS`. Sendo um atributo categórico você deve fazer o `hot encode` antes.

DICA: (IMPORTANTE) ajuste o nome das colunas, não são aceitos para fórmula do statsmodel atributos com ' ' (brancos) no nome das colunas.

```
pd_dummies = pd.get_dummies(df['VEHICLECLASS'], prefix='VEHICLECLASS')
df = pd.concat([df,pd_dummies],axis=1)
df.iloc[0]
```

```
MODELYEAR                                2014
MAKE                                    ACURA
MODEL                                     ILX
VEHICLECLASS                          COMPACT
ENGINESIZE                                  2
CYLINDERS                                   4
TRANSMISSION                              AS5
FUELTYPE                                    Z
FUELCONSUMPTION_CITY                      9.9
FUELCONSUMPTION_HWY                       6.7
FUELCONSUMPTION_COMB                      8.5
FUELCONSUMPTION_COMB_MPG                   33
CO2EMISSIONS                              196
VEHICLECLASS_COMPACT                        1
VEHICLECLASS_FULL-SIZE                      0
VEHICLECLASS_MID-SIZE                       0
VEHICLECLASS_MINICOMPACT                    0
VEHICLECLASS_MINIVAN                        0
VEHICLECLASS_PICKUP TRUCK - SMALL           0
VEHICLECLASS_PICKUP TRUCK - STANDARD        0
VEHICLECLASS_SPECIAL PURPOSE VEHICLE        0
VEHICLECLASS_STATION WAGON - MID-SIZE       0
VEHICLECLASS_STATION WAGON - SMALL          0
VEHICLECLASS_SUBCOMPACT                     0
VEHICLECLASS_SUV - SMALL                    0
VEHICLECLASS_SUV - STANDARD                 0
VEHICLECLASS_TWO-SEATER                     0
VEHICLECLASS_VAN - CARGO                    0
VEHICLECLASS_VAN - PASSENGER                0
Name: 0, dtype: object
```

```
df.columns = df.columns.str.replace("-", "_")
df.columns = df.columns.str.replace(" ", "")
```

```
formula = 'CO2EMISSIONS ~ FUELCONSUMPTION_COMB + ENGINESIZE'
for c in [x for x in df.columns if x.find('VEHICLECLASS_') >= 0]:
```

```
    formula = formula + ' + ' + c
print(formula)
```

```
CO2EMISSIONS ~ FUELCONSUMPTION_COMB + ENGINESIZE + VEHICLECLASS_COMPACT + VEHICLECLAS
```

```
model = sm.ols(formula='CO2EMISSIONS ~ FUELCONSUMPTION_COMB + ENGINESIZE + VEHICLECLASS',

result = model.fit()
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           CO2EMISSIONS   R-squared:                       0.870
Model:                            OLS   Adj. R-squared:                  0.868
Method:                 Least Squares   F-statistic:                     414.5
Date:                Tue, 20 Oct 2020   Prob (F-statistic):               0.00
Time:                        20:02:26   Log-Likelihood:                -4850.3
No. Observations:                1067   AIC:                             9737.
Df Residuals:                    1049   BIC:                             9826.
Df Model:                          17
Covariance Type:            nonrobust
================================================================================================
                                                 coef    std err          t      P>|t|
------------------------------------------------------------------------------------------------
Intercept                                     85.1547      3.314     25.694      0.000
VEHICLECLASS[T.FULL-SIZE]                     -1.1773      3.158     -0.373      0.709
VEHICLECLASS[T.MID-SIZE]                      -4.5891      2.482     -1.849      0.065
VEHICLECLASS[T.MINICOMPACT]                    0.7377      3.801      0.194      0.846
VEHICLECLASS[T.MINIVAN]                        0.8707      6.444      0.135      0.893
VEHICLECLASS[T.PICKUP TRUCK - SMALL]          27.1642      6.916      3.928      0.000
VEHICLECLASS[T.PICKUP TRUCK - STANDARD]        1.4902      3.745      0.398      0.691
VEHICLECLASS[T.SPECIAL PURPOSE VEHICLE]       18.1171      8.881      2.040      0.042
VEHICLECLASS[T.STATION WAGON - MID-SIZE]      -5.8249      9.569     -0.609      0.543
VEHICLECLASS[T.STATION WAGON - SMALL]          7.4700      4.217      1.771      0.077
VEHICLECLASS[T.SUBCOMPACT]                     7.6220      3.381      2.255      0.024
VEHICLECLASS[T.SUV - SMALL]                   11.4515      2.580      4.439      0.000
VEHICLECLASS[T.SUV - STANDARD]                 9.9109      3.148      3.148      0.002
VEHICLECLASS[T.TWO-SEATER]                    10.3299      3.306      3.125      0.002
VEHICLECLASS[T.VAN - CARGO]                   13.0886      5.854      2.236      0.026
VEHICLECLASS[T.VAN - PASSENGER]               33.0287      5.860      5.636      0.000
FUELCONSUMPTION_COMB                           8.0833      0.435     18.600      0.000
ENGINESIZE                                    21.7192      0.924     23.495      0.000
==============================================================================
Omnibus:                       44.735   Durbin-Watson:                   1.679
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               59.488
Skew:                          -0.408   Prob(JB):                     1.21e-13
Kurtosis:                       3.820   Cond. No.                         180.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
```

Exercício. Atributos Categóricos diretos

A biblioteca `statsmodel` permite empregar atributos categóricos diretamente para a regressão linear. Ela faz internamente o `hot encode`. Faça a regressão linear anterior informando o atributo `VEHICLECLASS` diretamente, sem o `hot encode`. Existe diferenças com relação ao resultado anterior? Qual?

```
model = sm.ols(formula=formula, data=df)

result = model.fit()
print(result.summary())
```

O modelo é o mesmo, mas emprega uma atributo 'hot encode' a menos (veja que isso faz sentido, o atributos `sexo` (M e F) pode ser representado com um único `hot encode`, uma a menos do que os valores do atributo).