In [153]:

```
#Pré-Processamento de dados
import pandas as pd
dados = pd.read_csv('iris.csv')
print(dados.head())
```

```
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
```

In [154]:

```
#pegando os dados unicos
dados['species'].unique()
```

Out[154]:

```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

In [179]:

```
#Substituindo os valores texto do rotulo por numero
dados = dados.replace({'species':{'setosa':1.1}})
dados = dados.replace({'species':{'versicolor':2.2}})
dados = dados.replace({'species':{'virginica':3.3}})
```

In [180]:

```
print(dados.head())
```

```
   sepal_length  sepal_width  petal_length  petal_width  species
0           5.1          3.5           1.4          0.2      1.1
1           4.9          3.0           1.4          0.2      1.1
2           4.7          3.2           1.3          0.2      1.1
3           4.6          3.1           1.5          0.2      1.1
4           5.0          3.6           1.4          0.2      1.1
```

In [183]:

```
X = dados.iloc[:, :-1].values
y = dados.iloc[:, -1].values
```

In [184]:

```
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
labelencoder = LabelEncoder()
X[:, -1] = labelencoder.fit_transform(X[:, -1])
onehotencoder = OneHotEncoder(categorical_features = [-1])
X = onehotencoder.fit_transform(X).toarray()
```

```
C:\Users\Daniel\Anaconda3\lib\site-packages\sklearn\preprocessing\_encoder
s.py:415: FutureWarning: The handling of integer data will change in versi
on 0.22. Currently, the categories are determined based on the range [0, m
ax(values)], while in the future they will be determined based on the uniq
ue values.
If you want the future behaviour and silence this warning, you can specify
"categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the c
ategories to integers, then you can now use the OneHotEncoder directly.
  warnings.warn(msg, FutureWarning)
C:\Users\Daniel\Anaconda3\lib\site-packages\sklearn\preprocessing\_encoder
s.py:451: DeprecationWarning: The 'categorical_features' keyword is deprec
ated in version 0.20 and will be removed in 0.22. You can use the ColumnTr
ansformer instead.
  "use the ColumnTransformer instead.", DeprecationWarning)
```

In [186]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size = 0.2, random_state
= 0)
```

In [187]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train[:,3:] = scaler.fit_transform(X_train[:,3:])
X_test[:,3:] = scaler.transform(X_test[:,3:])
```

In [188]:

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train,Y_train)
print(model.score(X_test,Y_test))
```

```
0.9445672250090675
```

In [189]:

```
import numpy as np
X_train = np.append (arr=np.ones([X_train.shape[0],1]).astype(int), values = X_train, a
xis = 1)
```

In [191]:

```
import statsmodels.api as sm
X_opt = [0,1,2,3,4,5,6]
regressor = sm.OLS(Y_train, X_train[:,X_opt]).fit()
print(regressor.summary())
```

```
                            OLS Regression Results
==========================================================================
====
Dep. Variable:                      y   R-squared:
0.757
Model:                            OLS   Adj. R-squared:
0.744
Method:                 Least Squares   F-statistic:                     5
8.75
Date:                Tue, 12 Nov 2019   Prob (F-statistic):           1.79
e-32
Time:                        10:16:47   Log-Likelihood:                -7
4.500
No. Observations:                 120   AIC:                             1
63.0
Df Residuals:                     113   BIC:                             1
82.5
Df Model:                           6
Covariance Type:            nonrobust
==========================================================================
====
                 coef    std err          t      P>|t|      [0.025      0.
975]
--------------------------------------------------------------------------
----
const          2.7127      0.050     54.388      0.000       2.614
2.811
x1            -1.6975      0.214     -7.941      0.000      -2.121      -
1.274
x2            -1.6975      0.110    -15.487      0.000      -1.915      -
1.480
x3            -1.6975      0.214     -7.941      0.000      -2.121      -
1.274
x4            -0.3047      0.043     -7.144      0.000      -0.389      -
0.220
x5            -0.1543      0.042     -3.637      0.000      -0.238      -
0.070
x6            -0.1543      0.042     -3.637      0.000      -0.238      -
0.070
==========================================================================
====
Omnibus:                      278.011   Durbin-Watson:
1.963
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               1
1.754
Skew:                          -0.211   Prob(JB):                      0.0
0280
Kurtosis:                       1.526   Cond. No.
5.35
==========================================================================
====

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
```

In [192]:

```
#Essa linha é usada pra retirar o modelo que o valor de P está acima de  0,05
#X_opt = [0,1,2,3,4,6]
#regressor_OLS = sm.OLS(Y_train, X_train[:,X_opt]).fit()
#print(regressor_OLS.summary())
```

In [200]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X[:,[0,1,2,3,4,5,6]], y, test_size
= 0.2, random_state = 0)
scaler = StandardScaler()
X_train[:,3:] = scaler.fit_transform(X_train[:,3:])
X_test[:,3:] = scaler.transform(X_test[:,3:])
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train,Y_train)
print('Model score: '+str(model.score(X_test,Y_test)))
```

```
Model score: 0.706142730673673
```

In [ ]: