Regressão Linear - Bolsa De Valores Americana

Com Split de Dados - Versão 02

fonte de dados: https://finance.yahoo.com/ (https://finance.yahoo.com/)

Ativos mais comercializados: https://finance.yahoo.com/screener/predefined/most_actives (https://finance.yahoo.com/screener/predefined/most_actives) - Bolsa de Nova York

Ativos da Bovespa

https://www.infomoney.com.br/cotacoes/ (https://www.infomoney.com.br/cotacoes/)

Nome: João Emanuel da Silva Lins - Matricula: 162080263

In [1]:

```
import numpy, pandas as pd
import matplotlib, matplotlib.pyplot as plt
import sklearn.linear_model as lm
import warnings
warnings.filterwarnings('ignore')
```

Leitura de dados

In [2]:

```
fb = pd.read_csv("fb2018.csv").set_index("Date")
fb.head()
```

Out[2]:

Date	Open	High	Low	Close	Adj Close	Volume
2012-05-18	42.049999	45.000000	38.000000	38.230000	38.230000	573576400
2012-05-21	36.529999	36.660000	33.000000	34.029999	34.029999	168192700
2012-05-22	32.610001	33.590000	30.940001	31.000000	31.000000	101786600
2012-05-23	31.370001	32.500000	31.360001	32.000000	32.000000	73600000
2012-05-24	32.950001	33.209999	31.770000	33.029999	33.029999	50237200

In [3]:

```
fb.tail()
```

Out[3]:

Date	Open	High	Low	Close	Adj Close	Volume
2018-04-02	157.809998	159.199997	154.110001	155.389999	155.389999	36796000
2018-04-03	156.550003	157.389999	150.809998	156.110001	156.110001	42034000
2018-04-04	152.029999	155.559998	150.509995	155.100006	155.100006	49885600
2018-04-05	161.559998	161.570007	156.649994	159.339996	159.339996	41449600
2018-04-06	157.729996	161.419998	156.809998	157.199997	157.199997	41323600

In [4]:

```
print("Numero de Registros ou Tuplas: ", len(fb))
```

```
Numero de Registros ou Tuplas:  1480
```

In [5]:

```
1480/330
```

Out[5]:

```
4.484848484848484
```

In [6]:

```
# converte o index para o tipo data
fb.index = pd.to_datetime(fb.index)
fb.head()
```

Out[6]:

Date	Open	High	Low	Close	Adj Close	Volume
2012-05-18	42.049999	45.000000	38.000000	38.230000	38.230000	573576400
2012-05-21	36.529999	36.660000	33.000000	34.029999	34.029999	168192700
2012-05-22	32.610001	33.590000	30.940001	31.000000	31.000000	101786600
2012-05-23	31.370001	32.500000	31.360001	32.000000	32.000000	73600000
2012-05-24	32.950001	33.209999	31.770000	33.029999	33.029999	50237200

In [7]:

```
fb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1480 entries, 2012-05-18 to 2018-04-06
Data columns (total 6 columns):
Open         1480 non-null float64
High         1480 non-null float64
Low          1480 non-null float64
Close        1480 non-null float64
Adj Close    1480 non-null float64
Volume       1480 non-null int64
dtypes: float64(5), int64(1)
memory usage: 80.9 KB
```

In [8]:

```
fb.Close.describe()
```

Out[8]:

```
count    1480.000000
mean       89.860723
std        48.678628
min        17.730000
25%        49.395002
50%        81.410000
75%       124.352499
max       193.089996
Name: Close, dtype: float64
```

In [9]:

```
fb.head()
```

Out[9]:

Date	Open	High	Low	Close	Adj Close	Volume
2012-05-18	42.049999	45.000000	38.000000	38.230000	38.230000	573576400
2012-05-21	36.529999	36.660000	33.000000	34.029999	34.029999	168192700
2012-05-22	32.610001	33.590000	30.940001	31.000000	31.000000	101786600
2012-05-23	31.370001	32.500000	31.360001	32.000000	32.000000	73600000
2012-05-24	32.950001	33.209999	31.770000	33.029999	33.029999	50237200

In [10]:

```
fb.index
```

Out[10]:

```
DatetimeIndex(['2012-05-18', '2012-05-21', '2012-05-22', '2012-05-2
3',
               '2012-05-24', '2012-05-25', '2012-05-29', '2012-05-3
0',
               '2012-05-31', '2012-06-01',
               ...
               '2018-03-23', '2018-03-26', '2018-03-27', '2018-03-2
8',
               '2018-03-29', '2018-04-02', '2018-04-03', '2018-04-0
4',
               '2018-04-05', '2018-04-06'],
              dtype='datetime64[ns]', name='Date', length=1480, freq
=None)
```

Preparacao do modelo

In [11]:

```
olm = lm.LinearRegression()
X = numpy.array([x.toordinal() for x in fb.index])[:, numpy.newaxis]
y = fb['Close']
print(y[:5])
```

```
Date
2012-05-18    38.230000
2012-05-21    34.029999
2012-05-22    31.000000
2012-05-23    32.000000
2012-05-24    33.029999
Name: Close, dtype: float64
```

In [12]:

```
print(X[:5])
```

```
[[734641]
 [734644]
 [734645]
 [734646]
 [734647]]
```

Aplicacao do modelo de regressao

In [13]:

```
olm.fit(X, y)
olm
```

Out[13]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, norma
lize=False)
```

Predição

In [14]:

```
import numpy as np

yp = [olm.predict( np.array( x.toordinal() ).reshape(-1, 1)  )[0] for x in fb.in
dex]
yp[:3]
```

Out[14]:

```
[6.836301229246601, 7.067829500949301, 7.145005591519293]
```

Métrica para avaliar o modelo

R^2 - coeficiente de determinação. É uma métrica que mede os o quanto dos futuros examplos são previstos corretamente.
Varia entre 0 e 1. Quanto mais o R^2 se aproximar de 1, melhor a previsão.
Um R^2 próximo de 0, não reflete o modelo.

In [15]:

```
from sklearn.metrics import r2_score
# aplicando a predicao - datas
y_pred = [olm.predict( np.array( x.toordinal() ).reshape(-1, 1)  )[0] for x in f
b.index]

# Evaluate the model
r2 = r2_score(y, y_pred)
print('r2 = ', r2 )
```

```
r2 =  0.9664639352112534
```

a reta de regressao

In [16]:

```
a = olm.coef_[0]
b = olm.intercept_
print(' y = {0} * x + {1}'.format(a, b))
```

```
 y = 0.07717609056774749 * x + -56689.884049551336
```

Plota os modelo (Dados Observados x Dados Previstos)

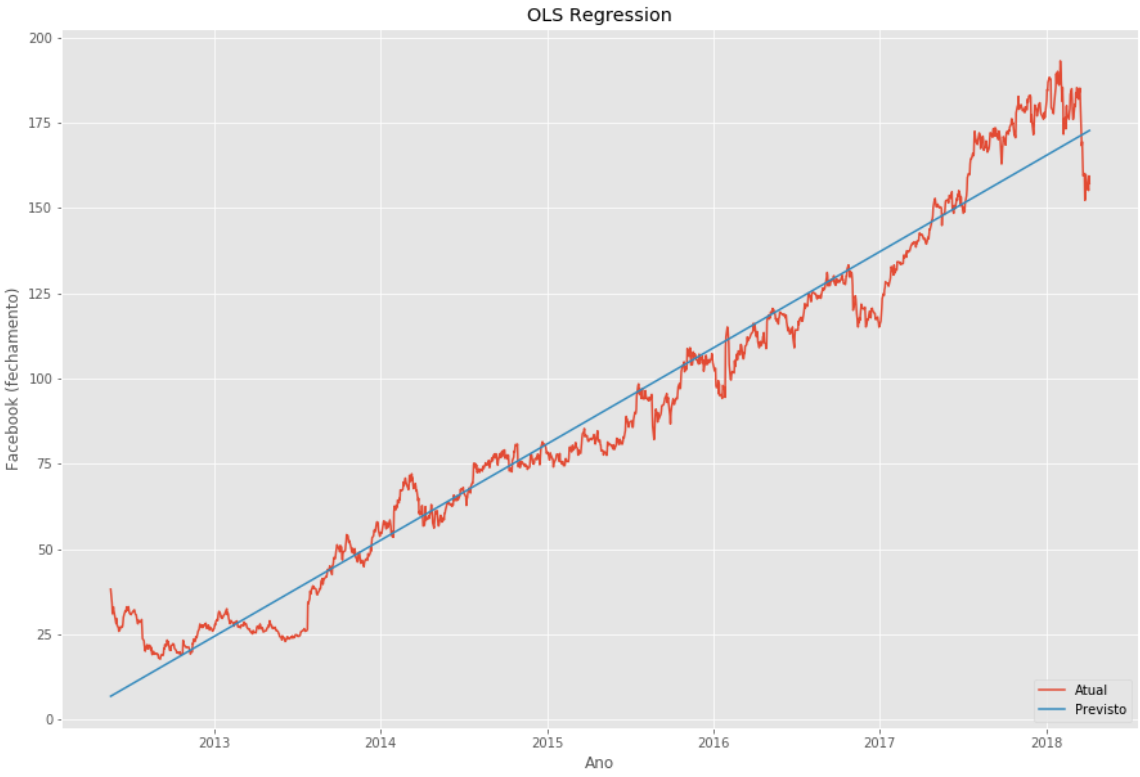
In [17]:

```
# Plot do modelo
matplotlib.style.use("ggplot")
plt.figure(figsize=(15,10))

# Plot o Y e o YPrevisto - datas
plt.plot(fb.index, y)
plt.plot(fb.index, y_pred)

# adcionando textos
plt.title("OLS Regression")
plt.xlabel("Ano")
plt.ylabel("Facebook (fechamento)")
plt.legend(["Atual", "Previsto"], loc="lower right")
plt.savefig("fabebook-linregr.pdf")
print('Reta de Regressão: y = {0} * x + {1}'.format(a, b))
plt.show()
```

```
Reta de Regressão: y = 0.07717609056774749 * x + -56689.884049551336
```



previsao para os anos de 2019, 2020, ...

In [18]:

```
from datetime import datetime

data = "2019-01-30"
datetime_object = datetime.strptime(data, '%Y-%m-%d')
yp2019 = olm.predict(np.array (datetime_object.toordinal()).reshape(-1, 1)  )
print ("previsao para ", data, " (fechamento) = us$ ", yp2019[0])
```

```
previsao para  2019-01-30  (fechamento) = us$  195.76337093909387
```

In [19]:

```
data = "2020-01-30"
datetime_object = datetime.strptime(data, '%Y-%m-%d')
yp2020 = olm.predict(np.array (datetime_object.toordinal()).reshape(-1, 1)  )
print ("previsao para ", data, " (fechamento) = us$ ", yp2020[0])
```

```
previsao para  2020-01-30  (fechamento) = us$  223.9326439963188
```

In [20]:

```
data = "2021-01-30"
datetime_object = datetime.strptime(data, '%Y-%m-%d')
yp2025 = olm.predict(np.array (datetime_object.toordinal()).reshape(-1, 1)  )
print ("previsao para ", data, " (fechamento) = us$ ", yp2025[0])
```

```
previsao para  2021-01-30  (fechamento) = us$  252.179093144121
```

In [21]:

```
lucro = yp2025[0] / yp2019[0] * 100
print("diferenca ano 2021 - 2019 - Lucro da ação do Facebook (%) ->", "{:3.0f}%"
.format(lucro)  )
```

```
diferenca ano 2021 - 2019 - Lucro da ação do Facebook (%) -> 129%
```

Regressão Linear - Usando conjuntos de treino/teste

Split aleatório de treino e teste

In [33]:

```
from IPython.display import Image
Image(url = 'images/split-de-dados-machine-learning.png')
```

Out[33]:

Este é o método mais utilizado para avaliar performance de um algoritmo de Machine Learning. Dividimos nossos dados originais em dados de treino e de teste. Treinamos o algoritmo nos dados de treino e fazemos as previsões nos dados de teste e avaliamos o resultado. A divisão dos dados vai depender do seu dataset, mas utiliza-se com frequência tamanhos entre 70/30 (treino/teste) e 65/35 (treino/teste). Este método é bem veloz e ideal para conjuntos de dados muito grandes. O ponto negativo é a possibilidade de alta variância.

In [43]:

```
from sklearn import model_selection
# Definindo o tamanho das amostras
teste_size = 0.33

# Garante que os resultados podem ser reproduzidos
# Isso é importante para comparar a acurácia com outros algoritmos de Machine Le
arning.
seed = 7

Y = y
# Criando os conjuntos de dados de treino e de teste
X_treino, X_teste, Y_treino, Y_teste = model_selection.train_test_split(X, Y,
                                                                        test_si
ze = teste_size,
                                                                        random_
state = seed)
# Criação do modelo
modelo = lm.LinearRegression()
modelo.fit(X_treino, Y_treino)
olm = modelo

# aplicando a predicao - datas
from sklearn.metrics import r2_score
y_pred = [olm.predict(x.reshape(-1, 1)) for x in X_teste.ravel()]
# Evaluate the model
r2 = r2_score(Y_teste, y_pred)
print('r2 = ', r2 )
```

```
r2 =  0.9661838204844923
```

In [44]:

```
X_teste[:3].ravel()
```

Out[44]:

```
array([736629, 736263, 735467])
```

a reta de regressao

In [45]:

```
a = olm.coef_[0]
b = olm.intercept_
print(' y = {0} * x + {1}'.format(a, b))
```

```
 y = 0.07731167411063893 * x + -56789.451002988564
```

Plota o modelo

In [51]:

```
# Plot do modelo
matplotlib.style.use("ggplot")
plt.figure(figsize=(15,10))

y_pred = [olm.predict(x.toordinal())[0] for x in fb.index]

# Plot o Y e o YPrevisto - datas
plt.plot(fb.index, y)
plt.plot(fb.index, y_pred)

# adcionando textos
plt.title("OLS Regression")
plt.xlabel("Ano")
plt.ylabel("Facebook (fechamento)")
plt.legend(["Atual", "Previsto"], loc="lower right")
plt.savefig("fabebook-linregr.pdf")

print('Reta de Regressão Linear: y = {0} * x + {1}'.format(a, b))

plt.show()
```

```
---------------------------------------------------------------------
-------
ValueError                                Traceback (most recent cal
l last)
<ipython-input-51-9d1217141aec> in <module>
      3 plt.figure(figsize=(15,10))
      4
----> 5 y_pred = [olm.predict(x.toordinal())[0] for x in fb.index]
      6
      7 # Plot o Y e o YPrevisto - datas

<ipython-input-51-9d1217141aec> in <listcomp>(.0)
      3 plt.figure(figsize=(15,10))
      4
----> 5 y_pred = [olm.predict(x.toordinal())[0] for x in fb.index]
      6
      7 # Plot o Y e o YPrevisto - datas

~\Anaconda3\lib\site-packages\sklearn\linear_model\base.py in predic
t(self, X)
    219             Returns predicted values.
    220         """
--> 221         return self._decision_function(X)
    222
    223     _preprocess_data = staticmethod(_preprocess_data)

~\Anaconda3\lib\site-packages\sklearn\linear_model\base.py in _decis
ion_function(self, X)
    202         check_is_fitted(self, "coef_")
    203
--> 204         X = check_array(X, accept_sparse=['csr', 'csc', 'co
o'])
    205         return safe_sparse_dot(X, self.coef_.T,
    206                                dense_output=True) + self.int
ercept_

~\Anaconda3\lib\site-packages\sklearn\utils\validation.py in check_a
rray(array, accept_sparse, accept_large_sparse, dtype, order, copy,
 force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_m
in_features, warn_on_dtype, estimator)
    512                     "Reshape your data either using array.re
shape(-1, 1) if "
    513                     "your data has a single feature or arra
y.reshape(1, -1) "
--> 514                     "if it contains a single sample.".format
(array))
    515
    516             # If input is 1D raise error
    517             if array.ndim == 1:

ValueError: Expected 2D array, got scalar array instead:
array=734641.
Reshape your data either using array.reshape(-1, 1) if your data has
a single feature or array.reshape(1, -1) if it contains a single sam
ple.


<Figure size 1080x720 with 0 Axes>
```

Usando Validação Cruzada para Avaliar o modelo

In [52]:

```
from IPython.display import Image
Image(url = 'images/cross-validation.jpg')
```

Out[52]:

Cross Validation é uma técnica que pode ser utilizada para avaliar a performance de um modelo com menos variância que a técnica de dividir os dados em treino/teste. Com esta técnica dividimos os dados em partes normalmente chamadas de k-folds (por exemplo k = 5, k = 10). Cada parte é chamada fold. O algoritmo é treinado em k-1 folds. Cada fold é usado no treinamento de forma repetida e um fold por vez. Após executar o processo em k-1 folds, podemos sumarizar a performance em cada fold usando a média e o desvio padrão (Eu disse que Estatística era importante no processo de Big Data Analytics). O resultado é normalmente mais confiável e oferece maior acurácia ao modelo. A chave deste processo está em definir o correto valor de k, de modo que o número de folds represente adequadamente o número de repetições necessárias.

In [53]:

```
# Definindo os valores para os folds
num_folds = 10
num_instances = len(X)
seed = 7

# Separando os dados em folds
kfold = model_selection.KFold(num_folds, True, random_state = seed)

Y = y
# Criando o modelo
modelo = lm.LinearRegression()
modelo.fit(X,Y)
resultado = model_selection.cross_val_score(modelo, X, Y, cv = kfold)

# Usamos a média e o desvio padrão
print("Acurácia: %.3f%% (%.3f%%)" % (resultado.mean()*100.0, resultado.std() * 1
00.0))
print(resultado.mean())
```

```
Acurácia: 96.596% (0.294%)
0.9659633164998389
```

In [54]:

```
# aplicando a predicao - datas
olm = modelo
yp = [olm.predict(x.toordinal())[0] for x in fb.index]
```

```
---------------------------------------------------------------------
-------
ValueError                                Traceback (most recent cal
l last)
<ipython-input-54-b5b33b594d5e> in <module>
      1 # aplicando a predicao - datas
      2 olm = modelo
----> 3 yp = [olm.predict(x.toordinal())[0] for x in fb.index]

<ipython-input-54-b5b33b594d5e> in <listcomp>(.0)
      1 # aplicando a predicao - datas
      2 olm = modelo
----> 3 yp = [olm.predict(x.toordinal())[0] for x in fb.index]

~\Anaconda3\lib\site-packages\sklearn\linear_model\base.py in predic
t(self, X)
    219             Returns predicted values.
    220         """
--> 221         return self._decision_function(X)
    222
    223     _preprocess_data = staticmethod(_preprocess_data)

~\Anaconda3\lib\site-packages\sklearn\linear_model\base.py in _decis
ion_function(self, X)
    202         check_is_fitted(self, "coef_")
    203
--> 204         X = check_array(X, accept_sparse=['csr', 'csc', 'co
o'])
    205         return safe_sparse_dot(X, self.coef_.T,
    206                                dense_output=True) + self.int
ercept_

~\Anaconda3\lib\site-packages\sklearn\utils\validation.py in check_a
rray(array, accept_sparse, accept_large_sparse, dtype, order, copy,
 force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_m
in_features, warn_on_dtype, estimator)
    512                     "Reshape your data either using array.re
shape(-1, 1) if "
    513                     "your data has a single feature or arra
y.reshape(1, -1) "
--> 514                     "if it contains a single sample.".format
(array))
    515
    516             # If input is 1D raise error
    517             if array.ndim == 1:

ValueError: Expected 2D array, got scalar array instead:
array=734641.
Reshape your data either using array.reshape(-1, 1) if your data has
a single feature or array.reshape(1, -1) if it contains a single sam
ple.
```

a reta de regressao

In [55]:

```
a = olm.coef_[0]
b = olm.intercept_
print("Modelo Cros-Validatinon")
print(' y = {0} * x + {1}'.format(a, b))
```

```
Modelo Cros-Validatinon
 y = 0.07717609056774749 * x + -56689.884049551336
```

In [56]:

```
print("Modelo de Split: ")
print(" y = 0.07118755997540646 * x + -52286.73808925971")
```

```
Modelo de Split:
 y = 0.07118755997540646 * x + -52286.73808925971
```

In [57]:

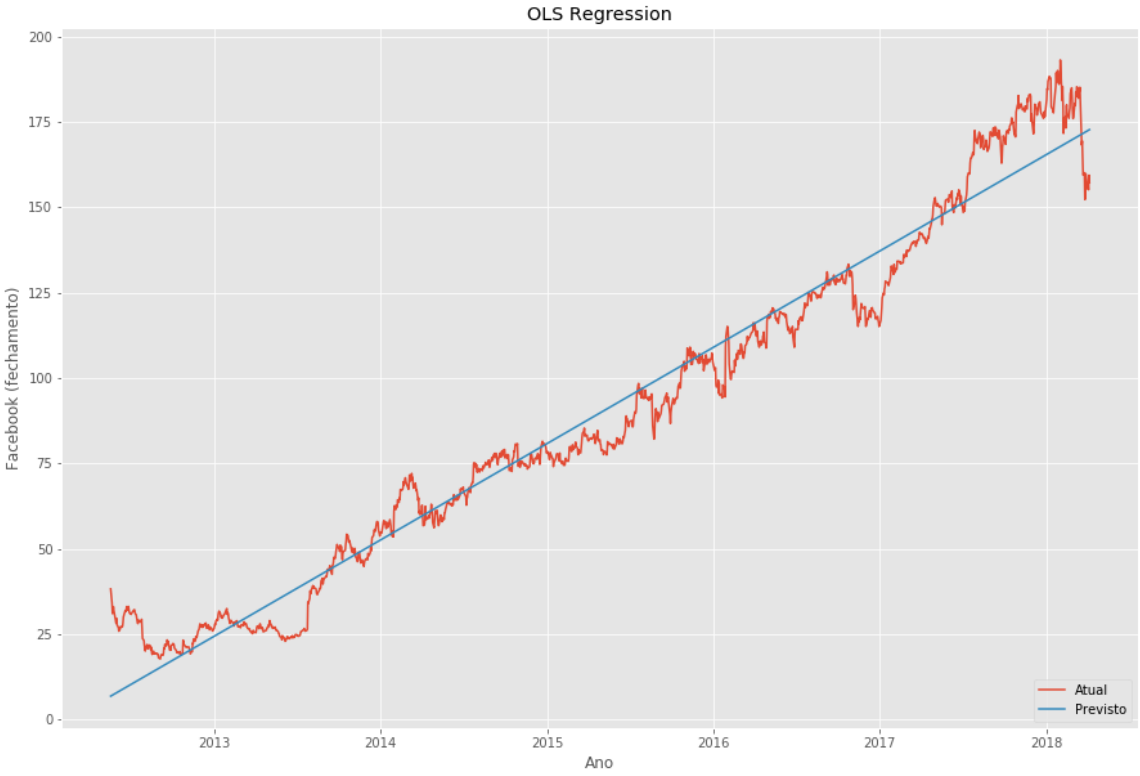
```
# Plot do modelo
matplotlib.style.use("ggplot")
plt.figure(figsize=(15,10))

# Plot o Y e o YPrevisto - datas
plt.plot(fb.index, y)
plt.plot(fb.index, yp)

# adcionando textos
plt.title("OLS Regression")
plt.xlabel("Ano")
plt.ylabel("Facebook (fechamento)")
plt.legend(["Atual", "Previsto"], loc="lower right")
plt.savefig("fabebook-linregr.pdf")

print('Reta de Regressão: y = {0} * x + {1}'.format(a, b))
plt.show()
```

```
Reta de Regressão: y = 0.07717609056774749 * x + -56689.884049551336
```



In [ ]:

In [ ]: