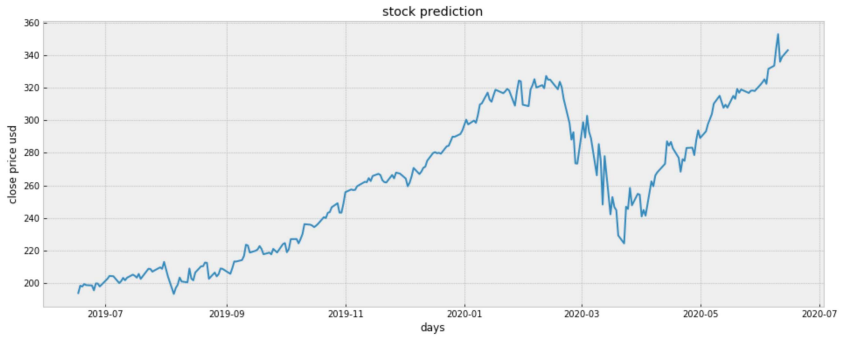
```
import numpy as np
import pandas as pd
import datetime
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
plt.style.use('bmh')
```

```
df=pd.read_csv(r'downloads/AAPL.csv',index_col='Date',parse_dates=True)
df.head(7)
```

Date	Open	High	Low	Close	Adj Close	Volume
2019-06-17	192.899994	194.960007	192.169998	193.889999	191.603317	14669100
2019-06-18	196.050003	200.289993	195.210007	198.449997	196.109528	26551000
2019-06-19	199.679993	199.880005	197.309998	197.869995	195.536377	21124200
2019-06-20	200.369995	200.610001	198.029999	199.460007	197.107620	21514000
2019-06-21	198.800003	200.850006	198.149994	198.779999	196.435623	47800600
2019-06-24	198.539993	200.160004	198.169998	198.580002	196.237991	18220400
2019-06-25	198.429993	199.259995	195.289993	195.570007	193.263504	21070300

```
plt.figure(figsize=(16,6))
plt.title('stock prediction')
plt.xlabel('days')
plt.ylabel('close price usd')
plt.plot(df['Close'])
plt.show()
```




```
df.shape
```

```
(252, 6)
```

```
future_days=25
df['prediction']=df['Close'].shift(-future_days)
df.head(7)
```

Date	Open	High	Low	Close	Adj Close	Volume	predict
2019-06-17	192.899994	194.960007	192.169998	193.889999	191.603317	14669100	208.839

```
X=np.array(df.drop(['prediction'],1))[:-future_days]
print(X)
```

```
[[1.92899994e+02 1.94960007e+02 1.92169998e+02 1.93889999e+02
  1.91603317e+02 1.46691000e+07]
 [1.96050003e+02 2.00289993e+02 1.95210007e+02 1.98449997e+02
  1.96109528e+02 2.65510000e+07]
 [1.99679993e+02 1.99880005e+02 1.97309998e+02 1.97869995e+02
  1.95536377e+02 2.11242000e+07]
 ...
 [3.00459991e+02 3.03239990e+02 2.98869995e+02 3.00630005e+02
  2.99818390e+02 3.55834000e+07]
 [3.03220001e+02 3.05170013e+02 3.01970001e+02 3.03739990e+02
  3.02919983e+02 2.88038000e+07]
 [3.05640015e+02 3.10350006e+02 3.04290009e+02 3.10130005e+02
  3.10130005e+02 3.35120000e+07]]
```

```
y=np.array(df['prediction'])[:-future_days]
print(y)
```

```
[208.839996 208.669998 207.020004 207.740005 209.679993 208.779999
 213.039993 208.429993 204.020004 193.339996 197.       199.039993
 203.429993 200.990005 200.479996 208.970001 202.75     201.740005
 206.5      210.350006 210.360001 212.639999 212.460007 202.639999
 206.490005 204.160004 205.529999 209.009995 208.740005 205.699997
 209.190002 213.279999 213.259995 214.169998 216.699997 223.589996
 223.089996 218.75     219.899994 220.699997 222.770004 220.960007
 217.729996 218.720001 217.679993 221.029999 219.889999 218.820007
 223.970001 224.589996 218.960007 220.820007 227.009995 227.059998
 224.399994 227.029999 230.089996 236.210007 235.869995 235.320007
 234.369995 235.279999 236.410004 240.509995 239.960007 243.179993
 243.580002 246.580002 249.050003 243.289993 243.259995 248.759995
 255.820007 257.5      257.130005 257.23999  259.429993 260.140015
 262.200012 261.959991 264.470001 262.640015 265.76001  267.100006
 266.290009 263.190002 262.01001  261.779999 266.369995 264.290009
 267.839996 267.25     264.160004 259.450012 261.73999  265.579987
 270.709991 266.920013 268.480011 270.769989 271.459991 275.149994
 279.859985 280.410004 279.73999  280.019989 279.440002 284.
 284.269989 289.910004 289.799988 291.519989 293.649994 300.350006
 297.429993 299.799988 298.390015 303.190002 309.630005 310.329987
 316.959991 312.679993 311.339996 315.23999  318.730011 316.570007
 317.700012 319.230011 318.309998 308.950012 317.690002 324.339996
 323.869995 309.51001  308.660004 318.850006 321.450012 325.209991
 320.029999 321.549988 319.609985 327.200012 324.869995 324.950012
 319.       323.619995 320.299988 313.049988 298.179993 288.079987
 292.649994 273.519989 273.359985 298.809998 289.320007 302.73999
 292.920013 289.029999 266.170013 285.339996 275.429993 248.229996
 277.970001 242.210007 252.860001 246.669998 244.779999 229.240005
 224.369995 246.880005 245.520004 258.440002 247.740005 254.809998
 254.289993 240.910004 244.929993 241.410004 262.470001 259.429993
 266.070007 267.98999  273.25     287.049988 284.429993 286.690002
 282.799988 276.929993 268.369995 276.100006 275.029999 282.970001
 283.170013 278.579987 287.730011 293.799988 289.070007 293.160004
 297.559998 300.630005 303.73999  310.130005 315.01001  311.410004
 307.649994 309.540009 307.709991 314.959991 313.140015 319.230011
 316.850006 318.890015 316.730011 318.109985 318.25     317.940002
 321.850006 323.339996 325.119995 322.320007 331.5      333.459991
 343.98999  352.839996 335.899994 338.799988 342.98999 ]
```

```
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.25)
```

```
tree= DecisionTreeRegressor().fit(x_train, y_train)
```

```
lr= LinearRegression().fit(x_train,y_train)
```

```
x_future=df.drop(['prediction'],1)[:-future_days]
x_future=x_future.tail(future_days)
x_future=np.array(x_future)
x_future
```

```
array([[2.42800003e+02, 2.45699997e+02, 2.38970001e+02, 2.41410004e+02,
        2.40758270e+02, 3.24700000e+07],
       [2.50899994e+02, 2.63109985e+02, 2.49380005e+02, 2.62470001e+02,
        2.61761414e+02, 5.04551000e+07],
       [2.70799988e+02, 2.71700012e+02, 2.59000000e+02, 2.59429993e+02,
        2.58729614e+02, 5.07218000e+07],
       [2.62739990e+02, 2.67369995e+02, 2.61230011e+02, 2.66070007e+02,
        2.65351715e+02, 4.22238000e+07],
       [2.68700012e+02, 2.70070007e+02, 2.64700012e+02, 2.67989990e+02,
        2.67266510e+02, 4.05291000e+07],
       [2.68309998e+02, 2.73700012e+02, 2.65829987e+02, 2.73250000e+02,
        2.72512329e+02, 3.27557000e+07],
       [2.80000000e+02, 2.88250000e+02, 2.78049988e+02, 2.87049988e+02,
        2.86275055e+02, 4.87487000e+07],
       [2.82399994e+02, 2.86329987e+02, 2.80630005e+02, 2.84429993e+02,
        2.83662140e+02, 3.27886000e+07],
       [2.87380005e+02, 2.88200012e+02, 2.82350006e+02, 2.86690002e+02,
        2.85916046e+02, 3.92813000e+07],
       [2.84690002e+02, 2.86950012e+02, 2.76859985e+02, 2.82799988e+02,
        2.82036530e+02, 5.38125000e+07],
       [2.77950012e+02, 2.81679993e+02, 2.76850006e+02, 2.76929993e+02,
        2.76182373e+02, 3.25038000e+07],
       [2.76279999e+02, 2.77250000e+02, 2.65429993e+02, 2.68369995e+02,
        2.67645477e+02, 4.52479000e+07],
       [2.73609985e+02, 2.77899994e+02, 2.72200012e+02, 2.76100006e+02,
        2.75354614e+02, 2.92643000e+07],
       [2.75869995e+02, 2.81750000e+02, 2.74869995e+02, 2.75029999e+02,
        2.74287506e+02, 3.12036000e+07],
       [2.77200012e+02, 2.83010010e+02, 2.77000000e+02, 2.82970001e+02,
        2.82206085e+02, 3.16272000e+07],
       [2.81799988e+02, 2.84540009e+02, 2.79950012e+02, 2.83170013e+02,
        2.82405548e+02, 2.92719000e+07],
       [2.85079987e+02, 2.85829987e+02, 2.78200012e+02, 2.78579987e+02,
        2.77827911e+02, 2.80012000e+07],
       [2.84730011e+02, 2.89670013e+02, 2.83890015e+02, 2.87730011e+02,
        2.86953247e+02, 3.43202000e+07],
       [2.89959991e+02, 2.94529999e+02, 2.88350006e+02, 2.93799988e+02,
        2.93006836e+02, 4.54576000e+07],
       [2.86250000e+02, 2.99000000e+02, 2.85850006e+02, 2.89070007e+02,
        2.88289612e+02, 6.01542000e+07],
       [2.89170013e+02, 2.93690002e+02, 2.86320007e+02, 2.93160004e+02,
        2.92368561e+02, 3.33920000e+07],
       [2.95059998e+02, 3.01000000e+02, 2.94459991e+02, 2.97559998e+02,
        2.96756683e+02, 3.69378000e+07],
       [3.00459991e+02, 3.03239990e+02, 2.98869995e+02, 3.00630005e+02,
        2.99818390e+02, 3.55834000e+07],
       [3.03220001e+02, 3.05170013e+02, 3.01970001e+02, 3.03739990e+02,
        3.02919983e+02, 2.88038000e+07],
       [3.05640015e+02, 3.10350006e+02, 3.04290009e+02, 3.10130005e+02,
        3.10130005e+02, 3.35120000e+07]])
```

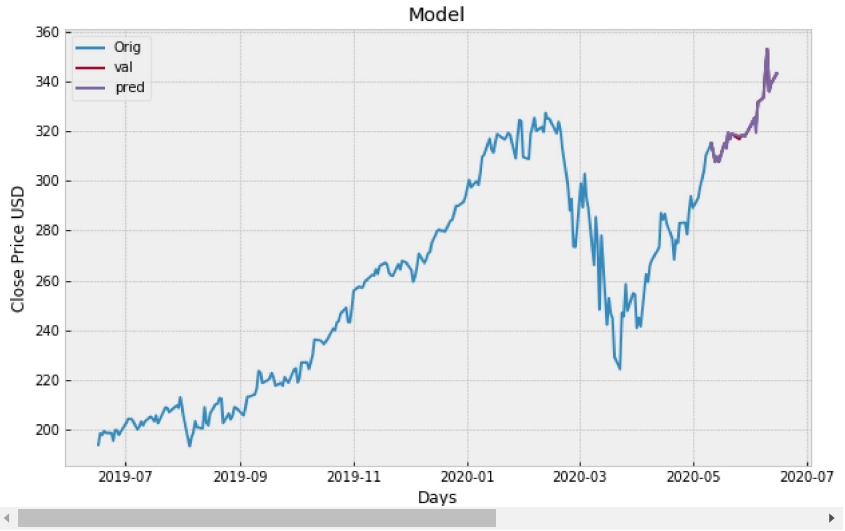
```
tree_prediction=tree.predict(x_future)
print(tree_prediction)
print()
lr_prediction=lr.predict(x_future)
print(lr_prediction)
```

```
[315.01001  311.410004 307.649994 309.540009 307.709991 314.959991
 313.140015 319.230011 316.850006 318.890015 317.940002 318.109985
 318.25     317.940002 321.850006 323.339996 325.119995 319.230011
 331.5      333.459991 343.98999  352.839996 335.899994 338.799988
 342.98999 ]

[290.91327548 282.28317136 287.40717183 304.77101713 307.46878997
 297.96451591 301.90085199 309.27789636 312.07727975 303.43386454
 309.90196686 290.50830137 304.87163617 302.62761912 306.8570739
 310.87437505 300.44520392 310.81089846 315.77971222 299.6903581
 307.01824516 313.30228144 320.62522893 322.56556812 348.91986667]
```

```
prediction=tree_prediction
valid=df[X.shape[0]:]
valid['prediction']=prediction
plt.figure(figsize=(10,6))
plt.title('Model')
plt.xlabel('Days')
plt.ylabel('Close Price USD')
plt.plot(df['Close'])
plt.plot(valid[['Close','prediction']])
```

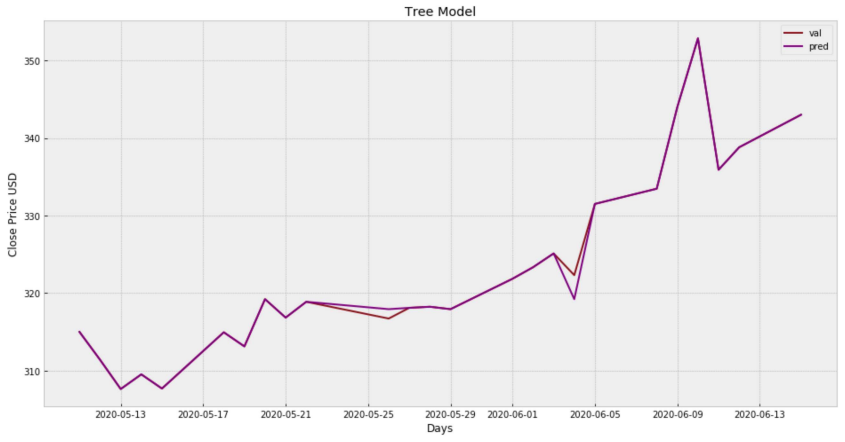
```
plt.legend(['Orig','val','pred'])
plt.show()
```

```
C:\Users\vigne\anaconda3\lib\site-packages\ipykernel_launcher.py:3: SettingWithCop
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable
  This is separate from the ipykernel package so we can avoid doing imports until
```



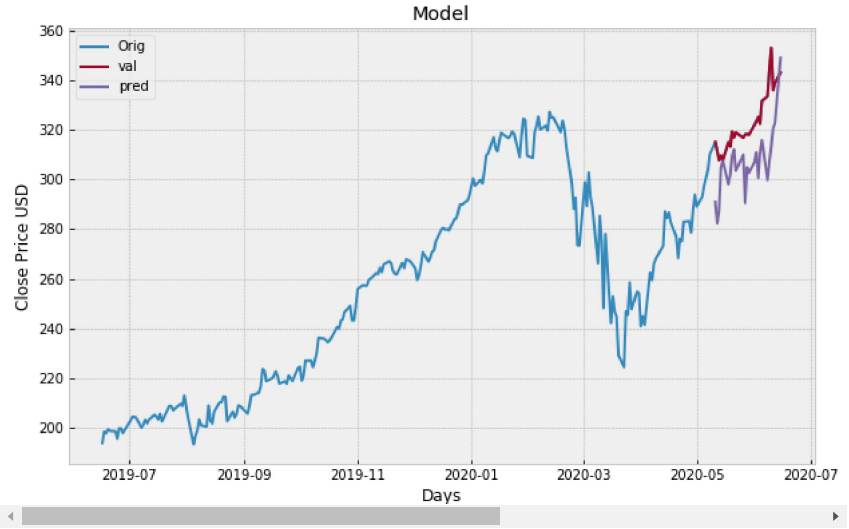

```
prediction=tree_prediction
valid=df[X.shape[0]:]
valid['prediction']=prediction
plt.figure(figsize=(16,8))
plt.title('Tree Model')
plt.xlabel('Days')
plt.ylabel('Close Price USD')
plt.plot(valid[['Close']],color= "#8a121c")
plt.plot(valid['prediction'],color="purple")
plt.legend(['val','pred'])
```

```
C:\Users\vigne\anaconda3\lib\site-packages\ipykernel_launcher.py:3: SettingWithCop
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable
  This is separate from the ipykernel package so we can avoid doing imports until
<matplotlib.legend.Legend at 0x14f92228e88>
```



```
prediction=lr_prediction
valid=df[X.shape[0]:]
valid['prediction']=prediction
plt.figure(figsize=(10,6))
plt.title('Model')
plt.xlabel('Days')
plt.ylabel('Close Price USD')
plt.plot(df['Close'])
plt.plot(valid[['Close','prediction']])
plt.legend(['Orig','val','pred'])
plt.show()
```

```
C:\Users\vigne\anaconda3\lib\site-packages\ipykernel_launcher.py:3: SettingWithCop
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable
  This is separate from the ipykernel package so we can avoid doing imports until
```




```
prediction=lr_prediction
valid=df[X.shape[0]:]
valid['prediction']=prediction
plt.figure(figsize=(16,8))
plt.title('Linear Model')
plt.xlabel('Days')
plt.ylabel('Close Price USD')
plt.plot(valid[['Close']],color= "#8a121c")
plt.plot(valid['prediction'],color="purple")
plt.legend(['val','pred'])
```

```
C:\Users\vigne\anaconda3\lib\site-packages\ipykernel_launcher.py:3: SettingWithCop
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable
  This is separate from the ipykernel package so we can avoid doing imports until
<matplotlib.legend.Legend at 0x14f9197d5c8>
```