In [4]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
import statsmodels.api as sm
from sklearn import linear_model
df = pd.read_csv(r'C:\Users\Berk\Desktop\aapl_e.csv')
```

In [5]:
```
df.tail(2)
```

Out[5]:

	#RIC	Date-Time	Price	Volume	Numberoftrades	Bidprice	Ask Price	N
24072	AAPL.OQ	2019-12-31T22:00:00.000000000Z	293.675	2716.0	NaN	293.50	293.85	
24073	AAPL.OQ	2019-12-31T23:00:00.000000000Z	293.810	848.0	NaN	293.77	293.85	

In [6]:
```
df = df.dropna(subset=['Numberoftrades', 'Depth','Volume','Spread','Price'])
```

In [7]:
```
x = df[['Numberoftrades', 'Depth','Volume','Spread']]
y = df[['Price']]
```

In [8]:
```
x_normalized = StandardScaler().fit_transform(x)
y_normalized = StandardScaler().fit_transform(y)
```

In [9]:
```
#To check if the data is normalized
x_normalized
```

Out[9]:
```
array([[-8.75524028e-01, -9.51987624e-01, -7.68765782e-01,
         1.91725535e-01],
       [-8.79798866e-01, -9.52366910e-01, -7.70541302e-01,
         1.57982502e-01],
       [-8.79086393e-01, -9.52499659e-01, -7.69804553e-01,
         7.76419457e-02],
       ...,
       [-4.48277732e-01, -3.18827468e-01, -4.78221197e-01,
        -1.89520475e-03],
       [ 7.10203332e-01,  5.81216890e-01,  4.55488938e-01,
        -1.09179919e-03],
       [-8.81223812e-01, -9.43870916e-01,  1.36051802e+00,
         1.17626898e-02]])
```

In [10]:
```
x_e = pd.DataFrame(x_normalized, columns = ['Numberoftrades', 'Depth', 'Volum
e','Spread'])
```

In [11]:

```
reg = linear_model.LinearRegression()
reg.fit(x_normalized, y_normalized)
```

Out[11]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fals
e)
```

In [12]:

```
model = sm.OLS(y_normalized, x_normalized).fit()
predictions = model.predict(x_normalized)

print_model = model.summary()
print(print_model)
```

```
                                 OLS Regression Results
=============================================================================
==========
Dep. Variable:                      y   R-squared (uncentered):
0.144
Model:                            OLS   Adj. R-squared (uncentered):
0.144
Method:                 Least Squares   F-statistic:
673.1
Date:                Wed, 03 Jun 2020   Prob (F-statistic):
0.00
Time:                        02:49:04   Log-Likelihood:
-21463.
No. Observations:               16003   AIC:
4.293e+04
Df Residuals:                   15999   BIC:
4.296e+04
Df Model:                           4
Covariance Type:            nonrobust
=============================================================================
=
                 coef    std err          t      P>|t|      [0.025      0.97
5]
-----------------------------------------------------------------------------
-
x1             0.3988      0.013     29.733      0.000       0.372       0.42
5
x2            -0.5490      0.012    -45.031      0.000      -0.573      -0.52
5
x3            -0.1995      0.009    -23.088      0.000      -0.216      -0.18
3
x4            -0.0151      0.007     -2.067      0.039      -0.029      -0.00
1
=============================================================================
=
Omnibus:                     4614.893   Durbin-Watson:                   0.10
7
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            10705.47
9
Skew:                           1.651   Prob(JB):                         0.0
0
Kurtosis:                       5.269   Cond. No.                         3.4
6
=============================================================================
=

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

In [48]:

```
from sklearn.tree import DecisionTreeRegressor
df = pd.read_csv(r'C:\Users\Berk\Desktop\aapl_e.csv')
df.describe()
df = df.dropna(subset=['Numberoftrades', 'Depth','Volume','Spread','Price'])
x = df[['Numberoftrades', 'Depth','Volume','Spread']].values
y = df[['Price']].values
```

In [49]:

```
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, rando
m_state=0)
regr = DecisionTreeRegressor()
regr.fit(x_train,y_train)
```

Out[49]:

```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=None, splitter='best')
```

In [50]:

```
y_pred = regr.predict(x_test)
```

In [51]:

```
df2=pd.DataFrame({'y_pred':y_pred})
df2.assign(y=y_test)
```

Out[51]:

```
      y_pred       y
   0  518.10  528.75
   1  223.64  186.70
   2  112.55  114.95
   3  265.98  215.61
   4   98.20  175.32
 ...     ...     ...
3196  131.77   97.36
3197  110.67  120.25
3198   93.39  511.83
3199  131.50  114.61
3200  104.97  127.15

3201 rows × 2 columns
```

In [52]:

```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y
_pred)))
```

```
Mean Absolute Error: 40.269442674164324
Mean Squared Error: 6366.101425572007
Root Mean Squared Error: 79.78785261912998
```

In [ ]: