In [19]:
```
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\aapl1.csv')
```

In [20]:
```
df.tail(2)
```

Out[20]:

	#RIC	Date-Time	Price	Volume	Numberoftrades	Bidprice	Ask Price	Nu
22346	AAPL.OQ	2019-12-31T22:00:00.000000000Z	293.675	2716.0	NaN	293.50	293.85	
22347	AAPL.OQ	2019-12-31T23:00:00.000000000Z	293.810	848.0	NaN	293.77	293.85	

In [21]:
```
df = df.dropna(subset=['Numberoftrades', 'Depth','Volume','Spread','Price'])
```

In [22]:
```
x = df[['Numberoftrades', 'Depth','Volume','Spread']]
y = df[['Price']]
```

In [23]:
```
x_normalized = StandardScaler().fit_transform(x)
y_normalized = StandardScaler().fit_transform(y)
```

In [24]:
```
#To check if the data is normalized
x_normalized
```

Out[24]:
```
array([[-0.90541615, -1.04209567, -0.81497772,  0.00374322],
       [-0.90727053, -1.04521684, -0.81509221,  0.00192592],
       [-0.90564795, -1.04258453, -0.81411772,  0.00737782],
       ...,
       [-0.48586371, -0.41707072, -0.5324871 , -0.0307855 ],
       [ 0.64484171,  0.47528485,  0.37295597, -0.0271509 ],
       [-0.90842951, -1.03677463,  1.25058628,  0.03100274]])
```

In [25]:
```
x_e = pd.DataFrame(x_normalized, columns = ['Numberoftrades', 'Depth', 'Volum
e','Spread'])
```

In [26]:
```
reg = linear_model.LinearRegression()
reg.fit(x_normalized, y_normalized)
```

Out[26]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fals
e)
```

In [27]:

```
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):	0.011
Model:	OLS	Adj. R-squared (uncentered):	0.011
Method:	Least Squares	F-statistic:	39.91
Date:	Wed, 03 Jun 2020	Prob (F-statistic):	2.66e-33
Time:	03:31:06	Log-Likelihood:	-20243.
No. Observations:	14322	AIC:	4.049e+04
Df Residuals:	14318	BIC:	4.052e+04
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
x1	0.0470	0.016	2.947	0.003	0.016	0.078
x2	-0.1208	0.014	-8.381	0.000	-0.149	-0.093
x3	0.0706	0.010	7.161	0.000	0.051	0.090
x4	-0.0125	0.008	-1.497	0.134	-0.029	0.004

Omnibus:	1848.193	Durbin-Watson:	0.014
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2641.151
Skew:	1.030	Prob(JB):	0.00
Kurtosis:	3.424	Cond. No.	3.63

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

In [28]:
```
from sklearn.tree import DecisionTreeRegressor
df = pd.read_csv(r'C:\Users\Berk\Desktop\aapl1.csv')
df.describe()
df = df.dropna(subset=['Numberoftrades', 'Depth','Volume','Spread','Price'])
x = df[['Numberoftrades', 'Depth','Volume','Spread']].values
y = df[['Price']].values
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[28]:
```
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 [29]:
```
y_pred = regr.predict(x_test)
```

In [30]:
```
df2=pd.DataFrame({'y_pred':y_pred})
df2.assign(y=y_test)
```

Out[30]:

	y_pred	y
0	180.84	100.67
1	125.45	128.60
2	105.34	94.04
3	186.02	166.57
4	223.11	155.47
...	...	...
2860	101.88	101.82
2861	175.42	142.83
2862	118.52	113.40
2863	165.24	151.68
2864	157.51	143.81

2865 rows × 2 columns

In [31]:
```
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: 30.67673996509599
Mean Squared Error: 1898.073335445026
Root Mean Squared Error: 43.566883471795705
```

In [ ]: