In [110]:
```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
```

In [111]:
```
df = pd.read_csv(r'C:\Users\Berk\Desktop\aapl1.csv')
df = df.dropna(subset=['Price','Volume'])
df = df[['Price','Volume']]
```

In [112]:
```
#RSI
rsi_period = 14
chg = df['Price'].diff(1)
gain = chg.mask(chg<0,0)
loss = chg.mask(chg>0,0)
avg_gain = gain.ewm(com = rsi_period-1, min_periods = rsi_period).mean()
avg_loss = loss.ewm(com = rsi_period-1, min_periods = rsi_period).mean()

rs = abs(avg_gain / avg_loss)
rsi = 100 - (100/(rs+1))
df['RSI'] = rsi
```

In [128]:
```
#SMA
df['SMA12'] = df['Price'].rolling(window = 12).mean()
df['SMA26'] = df['Price'].rolling(window = 26).mean()
df['SMA_diff'] = df['SMA12'] - df['SMA26']
#EMA
df['EMA12'] = df['Price'].ewm(span = 12).mean()
df['EMA26'] = df['Price'].ewm(span = 26).mean()
df['EMA_diff'] = df['EMA12'] - df['EMA26']
#MACD and its Signal
df['MACD'] = df['EMA12'] - df['EMA26']
df['Signal'] = df['MACD'].ewm(span = 9).mean() #signal line for MACD
df['MACD-SIGNAL'] = df['MACD'] - df['Signal']
```

In [129]:
```
#On Balance Volume - OBV
#df['OBV'] = (df.Volume * (~df['Price'].diff().le(0) * 2 - 1)).cumsum()
```

In [137]:
```
df = df.dropna(subset=['Price','RSI','MACD-SIGNAL','SMA_diff','EMA_diff'])
```

In [138]:
```
df = df[['Price','RSI','MACD-SIGNAL','SMA_diff','EMA_diff']]
```

In [139]:
```
regr = DecisionTreeRegressor()
regr.fit(df[['RSI','MACD-SIGNAL','SMA_diff','EMA_diff']], df['Price'])
```

Out[139]:
```
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 [144]:
```
#Prediction with 67 for rsi, 0.5 for macd-signal, 10 for SMA diff,30 for EMA d
iff
y_predicted1 = regr.predict([[17,-0.5,-10,-30]])
print(y_predicted1)
```

```
[193.6]
```

In [ ]: