ML_for_Stock

December 16, 2019

1 Can We Trust ML IN Stock Market?

2 Introduction

The stock market is one of the most well-known infrastructures through which anyone can potentially make a fortune by predicting the future stock prices. There's just one problem. How to predict the future of the stock market accurately? Here, we are going to use machine learning to possibly predict the stock market. In this case, we would focus on the predict of a famous company APPLE INC.

3 Background of APPLE INC.

Apple Inc. is an American multinational technology company headquartered in Cupertino, California, that designs, develops, and sells consumer electronics, computer software, and online services. It is considered one of the Big Four tech companies along with Amazon, Google, and Facebook. Shares of technology giant Apple Inc. (AAPL), which had become the first U.S.-based public company with a market cap of more than $1 trillion. Apple exerts a major influence on the performance of leading capitalization-weighted market indexes, index funds, and index-linked ETFs. Moreover, the S&P Information Technology Sector Index has never outperformed the entire S&P 500 Index (SPX) when Apple has dropped by more than 30%, according to Bank of America Merrill Lynch.

4 Importing Data using stocker

[11]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

[12]:
```
import stocker
from stocker import Stocker
```

[23]:
```
apple = Stocker('AAPL')
apple_dataset = apple.stock
print("the shape of the dataset is:", apple_dataset.shape)
```

```
AAPL Stocker Initialized. Data covers 1980-12-12 00:00:00 to 2018-03-27
00:00:00.
```

```
the shape of the dataset is: (9400, 16)
```

The Data Frame has 9400 rows with 16 columns, covering the period from Dec 12, 1980 through Mar 27, 2018. Now, to preview the first five and last five rows of our dataset.

[32]:
```
print(apple_dataset.head(5))
print(apple_dataset.tail(5))
```

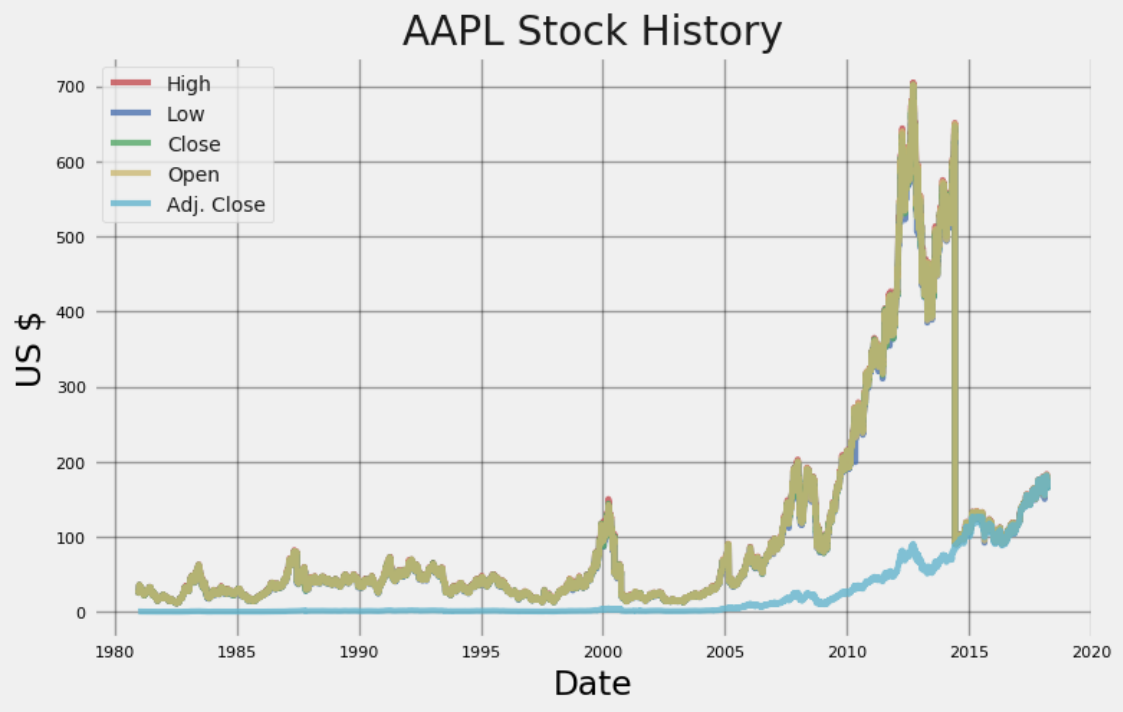
```
        Date   Open   High    Low  Close     Volume  Ex-Dividend  Split Ratio  \
0 1980-12-12  28.75  28.87  28.75  28.75  2093900.0          0.0          1.0
1 1980-12-15  27.38  27.38  27.25  27.25   785200.0          0.0          1.0
2 1980-12-16  25.37  25.37  25.25  25.25   472000.0          0.0          1.0
3 1980-12-17  25.87  26.00  25.87  25.87   385900.0          0.0          1.0
4 1980-12-18  26.63  26.75  26.63  26.63   327900.0          0.0          1.0

   Adj. Open  Adj. High  Adj. Low  Adj. Close  Adj. Volume         ds  \
0   0.422706   0.424470  0.422706    0.422706  117258400.0 1980-12-12
1   0.402563   0.402563  0.400652    0.400652   43971200.0 1980-12-15
2   0.373010   0.373010  0.371246    0.371246   26432000.0 1980-12-16
3   0.380362   0.382273  0.380362    0.380362   21610400.0 1980-12-17
4   0.391536   0.393300  0.391536    0.391536   18362400.0 1980-12-18

          y  Daily Change
0  0.422706      0.000000
1  0.400652     -0.001911
2  0.371246     -0.001764
3  0.380362      0.000000
4  0.391536      0.000000
           Date    Open    High     Low    Close      Volume  Ex-Dividend  \
9395 2018-03-21  175.04  175.09  171.26  171.270  35247358.0          0.0
9396 2018-03-22  170.00  172.68  168.60  168.845  41051076.0          0.0
9397 2018-03-23  168.39  169.92  164.94  164.940  40248954.0          0.0
9398 2018-03-26  168.07  173.10  166.44  172.770  36272617.0          0.0
9399 2018-03-27  173.68  175.15  166.92  168.340  38962839.0          0.0

      Split Ratio  Adj. Open  Adj. High  Adj. Low  Adj. Close  Adj. Volume  \
9395          1.0     175.04     175.09    171.26     171.270   35247358.0
9396          1.0     170.00     172.68    168.60     168.845   41051076.0
9397          1.0     168.39     169.92    164.94     164.940   40248954.0
9398          1.0     168.07     173.10    166.44     172.770   36272617.0
9399          1.0     173.68     175.15    166.92     168.340   38962839.0

             ds        y  Daily Change
9395 2018-03-21  171.270        -3.770
9396 2018-03-22  168.845        -1.155
9397 2018-03-23  164.940        -3.450
9398 2018-03-26  172.770         4.700
9399 2018-03-27  168.340        -5.340
```

5 Relationship Among Prices' Parameters

We filtered the unnecessary attributes out, and visualized the filtered attributes: High, Low, Close, Open, Adj.Close to see the difference.

[92]:
```
apple.plot_stock(stats = ['High','Low', 'Close','Open','Adj. Close'])
```

```
Maximum High = 705.07 on 2012-09-21 00:00:00.
Minimum High = 11.12 on 1982-07-08 00:00:00.
Current High = 175.15 on 2018-03-27 00:00:00.

Maximum Low = 699.57 on 2012-09-19 00:00:00.
Minimum Low = 11.00 on 1982-07-08 00:00:00.
Current Low = 166.92 on 2018-03-27 00:00:00.

Maximum Close = 702.10 on 2012-09-19 00:00:00.
Minimum Close = 11.00 on 1982-07-08 00:00:00.
Current Close = 168.34 on 2018-03-27 00:00:00.

Maximum Open = 702.41 on 2012-09-21 00:00:00.
Minimum Open = 11.12 on 1982-07-08 00:00:00.
Current Open = 173.68 on 2018-03-27 00:00:00.

Maximum Adj. Close = 181.72 on 2018-03-12 00:00:00.
Minimum Adj. Close = 0.16 on 1982-07-08 00:00:00.
Current Adj. Close = 168.34 on 2018-03-27 00:00:00.
```



5.1 Inference

We can already see some interesting patterns emerge:

1. Fours series "Close, High, Low and Open" perform in a similar pattern. On the other hand, Adjusted Close price perform in a extremely different pattern. This difference is hightlighed from year 1980 to year 2016 where Adj. Close price were almost zero.

Investigation of the difference The closing price is simply the cash value of that specific piece of stock at day's end while the adjusted closing price reflects the closing price of the stock in relation to other stock attributes(dividends/split ratio). In general, the adjusted closing price is considered to be a more technically accurate reflection of the true value of the stock. Since the close price for year 1980 to year 2016 were not very high, its adjusted close price after dividend were then close to zero.

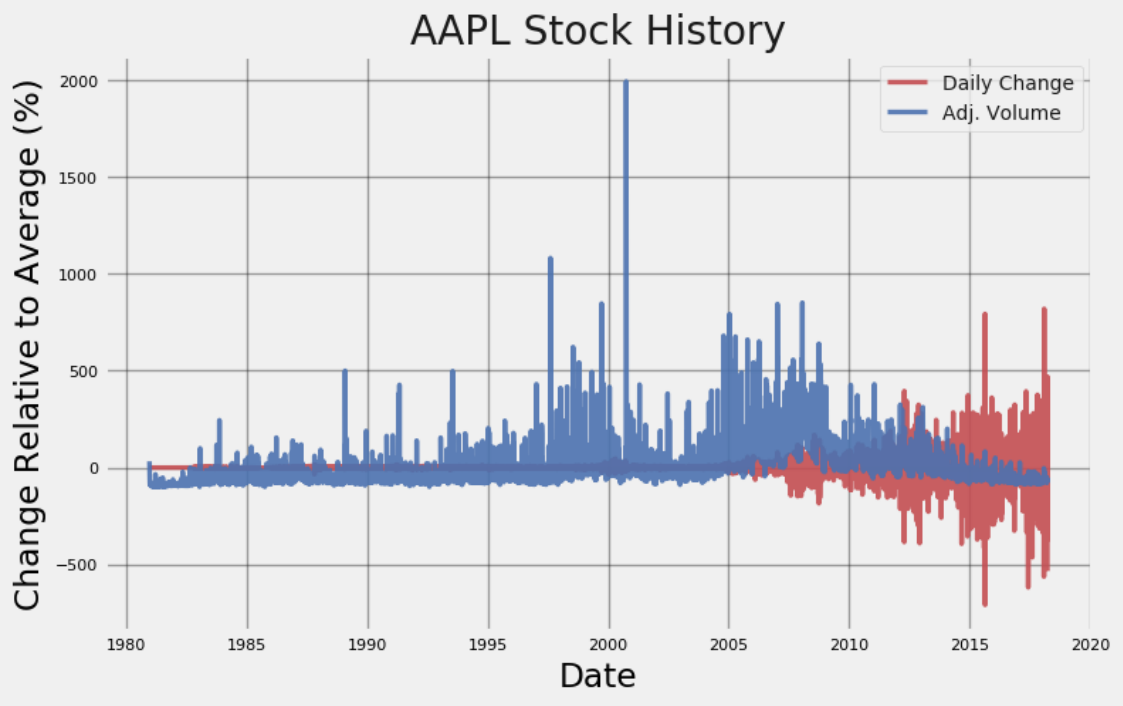
2. The series does not exhibit clear seasonality(no pattern repeats again and again at regular time intervals) Since the data has no seasonality, we assume that an MA model might not fit the dataset well, but we can do a MA model to confirm our assumption. Besides, the data has a clear increasing trend from year 2007, so probably a linear regression model might work.

Investigation of the increasing trend from year 2007 The first generation of IPhone(IPhone 1) was announced and released in 2007. As the first smartphone in the market, the release of IPhone improved Apple's brand awareness so that affect Apple's stock price in a extremely good way. Apple continuely released new products from year 2007 and was popular among the world. Therefore, there is an increasing trend from year 2007.

[ ]:
```
We also tried to figure out if volume would affect the price change.
```

[67]:
```
apple.plot_stock(stats = ['Daily Change', 'Adj. Volume'],  plot_type='pct')
```

```
Maximum Daily Change = 8.20 on 2018-02-06 00:00:00.
Minimum Daily Change = -7.09 on 2015-08-25 00:00:00.
Current Daily Change = -5.34 on 2018-03-27 00:00:00.

Maximum Adj. Volume = 1855410200.00 on 2000-09-29 00:00:00.
Minimum Adj. Volume = 250376.00 on 1985-09-27 00:00:00.
Current Adj. Volume = 38962839.00 on 2018-03-27 00:00:00.
```



The y-axis is in the percentage, because the daily volume is originally in shares, with a range in the hundreds of millions, while daily price change typically is a few dollars! By converting to percentage change we can look at both datasets on a similar scale.

5.1.1 Inference:

So, from this chart we can see there is no relationship between the daily trade volume with daily price change. Therefore, we might not use Volume to do further analysis.

5.2 Summary of analyzing the relationship among parameters

1). Using only Adj. Close price that reflects the true value of the stock to do further analysis. 2). Determing to use linear regression models by the visualized plot of Adj. Close price(no seasonality, increasing trend). But we also tried Additive Model, Moving Average Model and other models to compare with our assumption.

5.3 Insights at Apple's Adjusted Close Price

[30]:
```
apple.plot_stock()
```

```
Maximum Adj. Close = 181.72 on 2018-03-12 00:00:00.
Minimum Adj. Close = 0.16 on 1982-07-08 00:00:00.
Current Adj. Close = 168.34 on 2018-03-27 00:00:00.
```

Incredicle increasing, right?

6 An Interesting Fact

6.1 Scenario 1

6.1.1 If bought and hold Apple's share for a long term

Assuming my mom bought me 100 shares apple stock at the day I came to the world, that's $1,662 in close price, do you know how much I will earn when I sold it at my 21st birthday , that's the close price in 2018 Jan 31st?

[93]:
```
apple_dataset[apple_dataset['Date'] == '1997-1-31']
```

[93]:
```
                Date   Open   High   Low  Close     Volume  Ex-Dividend  \
Date
1997-01-31 1997-01-31  16.62  16.62  16.5  16.62  1782400.0          0.0

            Split Ratio  Adj. Open  Adj. High  Adj. Low  Adj. Close  \
Date
1997-01-31          1.0   0.533975   0.533975   0.53012    0.533975

            Adj. Volume         ds         y  Daily Change
Date
1997-01-31   49907200.0 1997-01-31  0.533975           0.0
```

[94]:
```
apple.buy_and_hold(start_date='1997-01-31', end_date='2018-01-31', nshares=100)
```

```
AAPL Total buy and hold profit from 1997-01-31 00:00:00 to 2018-01-31 00:00:00
for 100 shares = $16689.60
```

6.1.2 Inference

If bought 100 shares($16.62 close price per share) of Apple and hold for 21 years, this $1662 would become $16689.

6.2 Scenario 2

6.2.1 If bought Apple's share and sold it in short term using Additive Model

Assuming that we use Additive model to predict Apple's stock price in next 90 days, we want to know how the model could help us predict the future stock price correctly to earn benefit.

[ ]:
```
model, model_data = apple.create_prophet_model(days=90)
```

[ ]:
```
apple.evaluate_prediction()
```

Result shows that: When the model predicted an increase, the price increased

48.59% of the time. When the model predicted a decrease, the price decreased

43.66% of the time.

6.3 How Bad It is!! Just as bad as a coin flip!

6.4 Summary

We can see that if we bought and hold the share in a long term, we could increase our share value about 300 times; but if we want to bought and sold the share in a short term, the chance we could increase our share value is like fliping a coin. However, we want to figure out if there are any other ML models could perform better in a short term forecasting.

7 Moving Average

In summary, a moving average is a commonly used indicator in technical analysis. It's a lagging indicator, which means that it uses past prices to predict future prices. It's effective in smoothing out any short-term fluctuations and finding the overall trend. We'll use moving averages to see if we can do a better job of predicting stock prices.

[195]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
%matplotlib inline

import matplotlib.style
import matplotlib as mpl
mpl.style.use('ggplot')

from matplotlib.pylab import rcParams
rcParams['figure.figsize'] = 20, 10

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))

# Creating copy of goog_data dataframe for moving averages
df = apple_dataset

df['Date'] = pd.to_datetime(df.Date, format='%Y-%m-%d')
df.index = df['Date']
```

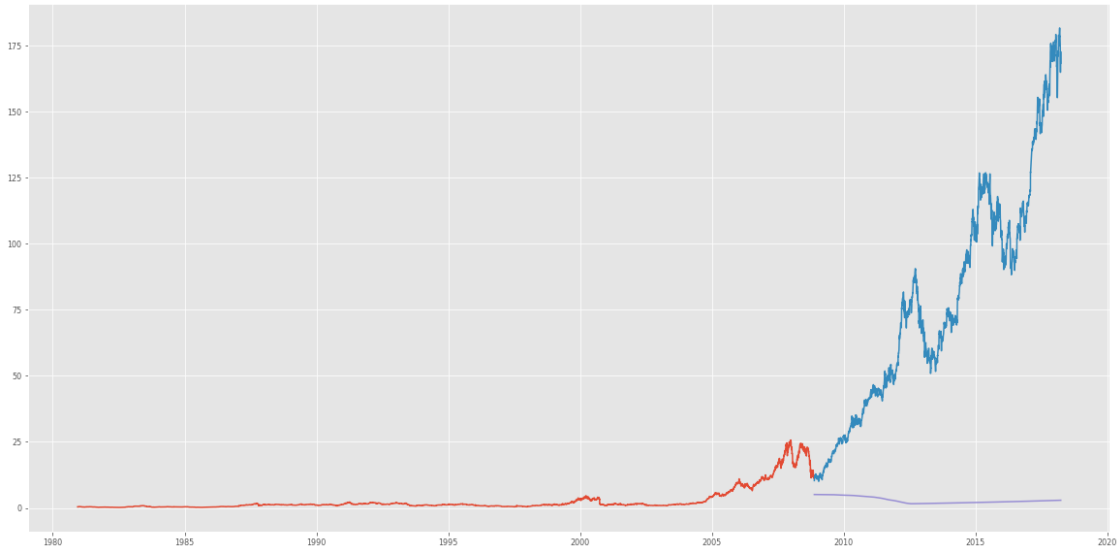
[216]:
```
# Sorting the dataset in ascending order
data = df.sort_index(ascending=True, axis=0)

# Creating a separate dataset
new_data = pd.DataFrame(index=range(0, len(df)), columns=['Date', 'Adj. Close'])
for i in range(0, len(data)):
    new_data['Date'][i] = data['Date'][i]
    new_data['Adj. Close'][i] = data['Adj. Close'][i]
```

```
 #Train-test split
# We split our data into valid set(75% of the datasest) and test set(25% of the␣
 ↪dataset)
train = new_data[:7050]
valid = new_data[7050:]

num = valid.shape[0]

train['Date'].min(), train['Date'].max(), valid['Date'].min(), valid['Date'].
 ↪max()

# Making predictions
pred = []
for i in range(0, num):
    a = train['Adj. Close'][len(train)-924+i:].sum() + sum(pred)
    b = a/num
    pred.append(b)
```

[258]:
```
# Plot
valid['Predictions']=0
valid['Predictions']=pred
plt.plot(train['Date'],train['Adj. Close'])
plt.plot(valid['Date'],valid[['Adj. Close','Predictions']])
```

```
/Users/yanlanLIANG/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:2: SettingWithCopyWarning:


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: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/Users/yanlanLIANG/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:3: SettingWithCopyWarning:


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: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[258]:
```
[<matplotlib.lines.Line2D at 0x1a4a05bda0>,
 <matplotlib.lines.Line2D at 0x1a4a05bc88>]
```



The red line is the Adj. Close price for the train set, the blue line is the Adj. Close price for the test set, the purple line is the Adj. Close price prediction for the test set.

7.0.1 Root Mean Square Error of MA Model

[259]:
```
rms_MA = np.sqrt(np.mean(np.power(np.array(test['Adj. Close'])-np.
↪array(pred),2)))
```

[260]:
```
rms_MA
```

[260]: 86.93413258927612

7.1 Summary

The moving average method failed to capture the generao trend of the stock data and the full extent of the increase and the up-and-down in price with a relatively high rms error rate(86.934)

8 Linear Regression

The most basic machine learning algorithm that can be implemented on this data is linear regression. The linear regression model returns an equation that determines the relationship between the independent variables and the dependent variable. The equation for linear regression can be written as: Y=theta1*x1+ theta2*x2 +... +thetan*xn Here, x1, x2,....xn represent the independent variables while the coefficients 1, 2, ... n represent the weights. You can refer to the following article to study linear regression in more detail: For our problem statement, we do not have a set of independent variables. We have only the dates instead. Hence, we create new feature Year and month as our independent variables.

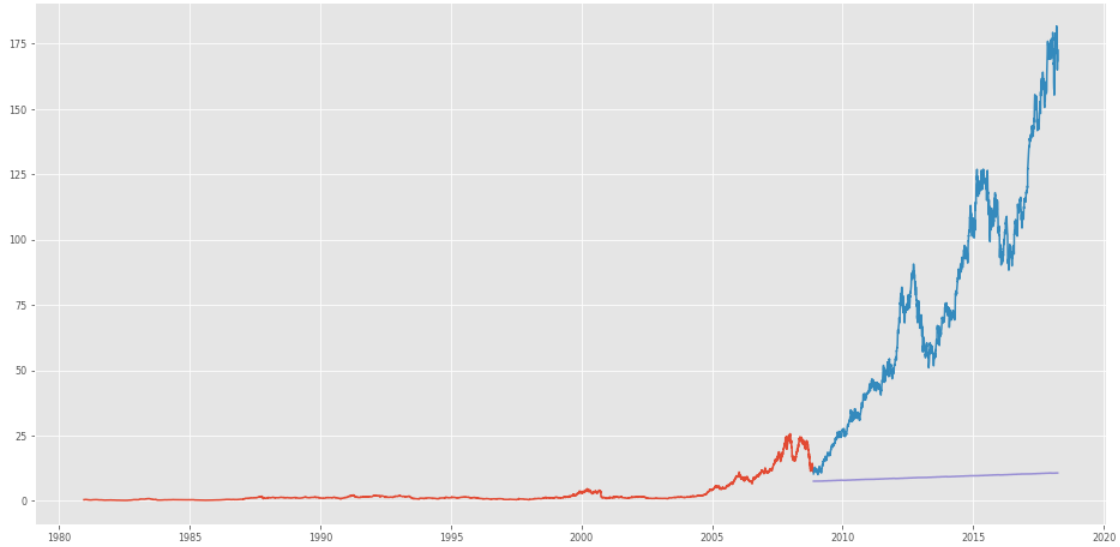
[327]:
```
# split the data into train and vaild as before
new_data2=apple_dataset

train = new_data2[:7050]
valid = new_data2[7050:]


x_train = train[['year','month']]

x_test = valid[['year','month']]

y_train = train['Adj. Close']

y_test = valid['Adj. Close']

from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train,y_train)
preds = model.predict(x_test)

# Plot
plt.figure(figsize=(16,8))
valid['Predictions'] = preds
plt.plot(train['Adj. Close'])#x is the data point at the dataset
plt.plot(valid[['Adj. Close','Predictions']])
```

	Date	Open	High	Low	Close	Volume	\
Date							
1980-12-12	1980-12-12	28.75	28.87	28.75	28.750	2093900.0	
1980-12-15	1980-12-15	27.38	27.38	27.25	27.250	785200.0	
1980-12-16	1980-12-16	25.37	25.37	25.25	25.250	472000.0	
1980-12-17	1980-12-17	25.87	26.00	25.87	25.870	385900.0	
1980-12-18	1980-12-18	26.63	26.75	26.63	26.630	327900.0	
…	…	…	…	…	…	…	
2018-03-21	2018-03-21	175.04	175.09	171.26	171.270	35247358.0	
2018-03-22	2018-03-22	170.00	172.68	168.60	168.845	41051076.0	
2018-03-23	2018-03-23	168.39	169.92	164.94	164.940	40248954.0	
2018-03-26	2018-03-26	168.07	173.10	166.44	172.770	36272617.0	
2018-03-27	2018-03-27	173.68	175.15	166.92	168.340	38962839.0	

	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	\
Date						
1980-12-12	0.0	1.0	0.422706	0.424470	0.422706	
1980-12-15	0.0	1.0	0.402563	0.402563	0.400652	
1980-12-16	0.0	1.0	0.373010	0.373010	0.371246	
1980-12-17	0.0	1.0	0.380362	0.382273	0.380362	
1980-12-18	0.0	1.0	0.391536	0.393300	0.391536	

```
…                  …           …          …           …           …
2018-03-21         0.0         1.0  175.040000  175.090000  171.260000
2018-03-22         0.0         1.0  170.000000  172.680000  168.600000
2018-03-23         0.0         1.0  168.390000  169.920000  164.940000
2018-03-26         0.0         1.0  168.070000  173.100000  166.440000
2018-03-27         0.0         1.0  173.680000  175.150000  166.920000

            Adj. Close  Adj. Volume         ds           y  Daily Change  \
Date
1980-12-12    0.422706  117258400.0 1980-12-12    0.422706      0.000000
1980-12-15    0.400652   43971200.0 1980-12-15    0.400652     -0.001911
1980-12-16    0.371246   26432000.0 1980-12-16    0.371246     -0.001764
1980-12-17    0.380362   21610400.0 1980-12-17    0.380362      0.000000
1980-12-18    0.391536   18362400.0 1980-12-18    0.391536      0.000000
…                  …          …        …          …           …
2018-03-21  171.270000   35247358.0 2018-03-21  171.270000     -3.770000
2018-03-22  168.845000   41051076.0 2018-03-22  168.845000     -1.155000
2018-03-23  164.940000   40248954.0 2018-03-23  164.940000     -3.450000
2018-03-26  172.770000   36272617.0 2018-03-26  172.770000      4.700000
2018-03-27  168.340000   38962839.0 2018-03-27  168.340000     -5.340000

            year  month
Date
1980-12-12  1980     12
1980-12-15  1980     12
1980-12-16  1980     12
1980-12-17  1980     12
1980-12-18  1980     12
…           …    …
2018-03-21  2018      3
2018-03-22  2018      3
2018-03-23  2018      3
2018-03-26  2018      3
2018-03-27  2018      3

[9400 rows x 18 columns]

/Users/yanlanLIANG/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:24: SettingWithCopyWarning:


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: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[327]:
```
[<matplotlib.lines.Line2D at 0x1c4d799358>,
 <matplotlib.lines.Line2D at 0x1c4d7994a8>]
```



The red line is the Adj. Close price for the train set, the blue line is the Adj. Close price for the test set, the purple line is the Adj. Close price prediction for the test set.

8.0.1 Root Mean Square Error of LR Model

[314]:
```
rms_LR = np.sqrt(np.mean(np.power(np.array(y_test)-np.array(preds),2)))
```

[315]:
```
rms_LR
```

[315]:
```
80.82799706132346
```

8.1 Summary

The Linear Regression Method capture the general increasing trend of the stock data but it failed to capture the full extent of the increase and the up-and-down in price Hence, although it has a relatively lower rms error rate(80.828) compared with the MA model, this models still does not fit the dataset well.

9 Auto ARIMA

ARIMA is a very popular statistical method for time series forecasting. ARIMA models take into account the past values to predict the future values. There are three important parameters in ARIMA:

p (past values used for forecasting the next value) q (past forecast errors used to predict the future values) d (order of differencing) Parameter tuning for ARIMA consumes a lot of time. So we will

use auto ARIMA which automatically selects the best combination of (p,q,d) that provides the least error.

[174]:
```
!pip install pmdarima
from pmdarima import auto_arima

data = df.sort_index(ascending=True, axis=0)

# split the train and validation set as before
train = data[:7050]
valid = data[7050:]

training = train['Adj. Close']
validation = valid['Adj. Close']

model = auto_arima(training, start_p=1, start_q=1,max_p=3, max_q=3,␣
 ↪m=12,start_P=0, seasonal=True,d=1, D=1,␣
 ↪trace=True,error_action='ignore',suppress_warnings=True)
model.fit(training)
# make prediction
forecast = model.predict(n_periods=2350)
forecast = pd.DataFrame(forecast,index = valid.index,columns=['Prediction'])
```

```
Requirement already satisfied: pmdarima in
/Users/yanlanLIANG/anaconda3/lib/python3.7/site-packages (1.5.1)
Requirement already satisfied: Cython>=0.29 in
/Users/yanlanLIANG/anaconda3/lib/python3.7/site-packages (from pmdarima)
(0.29.14)
Requirement already satisfied: scipy>=1.3 in
/Users/yanlanLIANG/anaconda3/lib/python3.7/site-packages (from pmdarima) (1.3.1)
Requirement already satisfied: six>=1.5 in
/Users/yanlanLIANG/anaconda3/lib/python3.7/site-packages (from pmdarima)
(1.13.0)
Requirement already satisfied: numpy>=1.16 in
/Users/yanlanLIANG/anaconda3/lib/python3.7/site-packages (from pmdarima)
(1.17.4)
Requirement already satisfied: pandas>=0.19 in
/Users/yanlanLIANG/anaconda3/lib/python3.7/site-packages (from pmdarima)
(0.25.3)
Requirement already satisfied: scikit-learn>=0.19 in
/Users/yanlanLIANG/anaconda3/lib/python3.7/site-packages (from pmdarima)
(0.21.3)
Requirement already satisfied: joblib>=0.11 in
/Users/yanlanLIANG/anaconda3/lib/python3.7/site-packages (from pmdarima)
(0.14.0)
Requirement already satisfied: statsmodels>=0.10.0 in
/Users/yanlanLIANG/anaconda3/lib/python3.7/site-packages (from pmdarima)
(0.10.1)
```

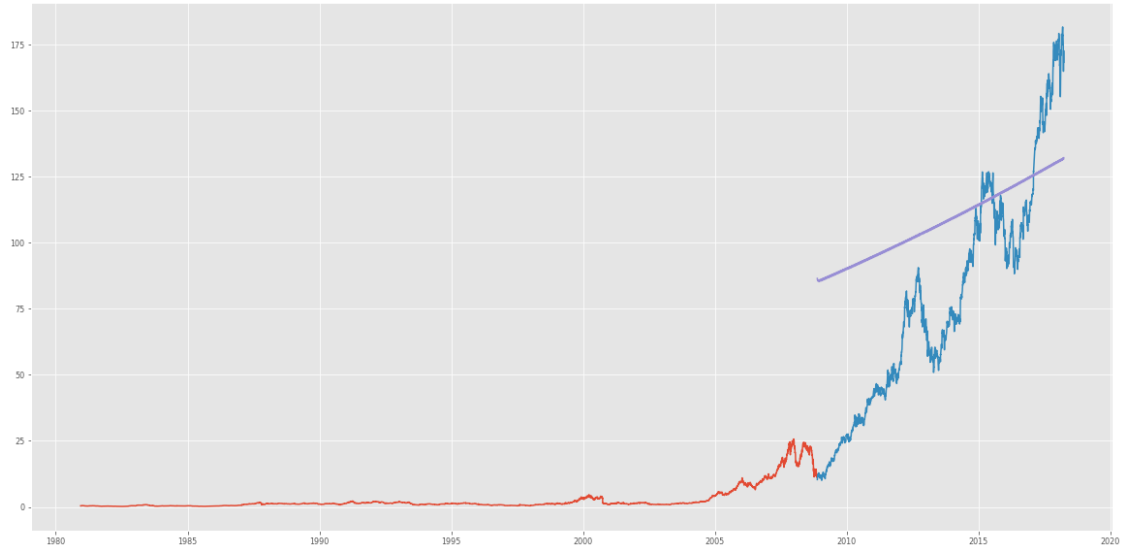
```
Requirement already satisfied: pathlib in
/Users/yanlanLIANG/anaconda3/lib/python3.7/site-packages (from pmdarima) (1.0.1)
Requirement already satisfied: python-dateutil>=2.6.1 in
/Users/yanlanLIANG/anaconda3/lib/python3.7/site-packages (from
pandas>=0.19->pmdarima) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in
/Users/yanlanLIANG/anaconda3/lib/python3.7/site-packages (from
pandas>=0.19->pmdarima) (2019.3)
Requirement already satisfied: patsy>=0.4.0 in
/Users/yanlanLIANG/anaconda3/lib/python3.7/site-packages (from
statsmodels>=0.10.0->pmdarima) (0.5.1)
Fit ARIMA: order=(1, 1, 1) seasonal_order=(0, 1, 1, 12); AIC=29247.746,
BIC=29282.041, Fit time=37.507 seconds
Fit ARIMA: order=(0, 1, 0) seasonal_order=(0, 1, 0, 12); AIC=33772.463,
BIC=33786.180, Fit time=0.582 seconds
Fit ARIMA: order=(1, 1, 0) seasonal_order=(1, 1, 0, 12); AIC=31827.931,
BIC=31855.367, Fit time=8.302 seconds
Fit ARIMA: order=(0, 1, 1) seasonal_order=(0, 1, 1, 12); AIC=29245.277,
BIC=29272.713, Fit time=25.206 seconds
Near non-invertible roots for order (0, 1, 1)(0, 1, 1, 12); setting score to inf
(at least one inverse root too close to the border of the unit circle: 1.000)
Fit ARIMA: order=(0, 1, 0) seasonal_order=(0, 1, 0, 12); AIC=33770.476,
BIC=33777.335, Fit time=0.665 seconds
Fit ARIMA: order=(1, 1, 1) seasonal_order=(0, 1, 0, 12); AIC=33763.728,
BIC=33791.164, Fit time=6.174 seconds
Fit ARIMA: order=(1, 1, 1) seasonal_order=(1, 1, 1, 12); AIC=29241.640,
BIC=29282.794, Fit time=38.743 seconds
Near non-invertible roots for order (1, 1, 1)(1, 1, 1, 12); setting score to inf
(at least one inverse root too close to the border of the unit circle: 1.000)
Fit ARIMA: order=(1, 1, 1) seasonal_order=(0, 1, 2, 12); AIC=29241.885,
BIC=29283.039, Fit time=80.613 seconds
Near non-invertible roots for order (1, 1, 1)(0, 1, 2, 12); setting score to inf
(at least one inverse root too close to the border of the unit circle: 1.000)
Fit ARIMA: order=(1, 1, 1) seasonal_order=(1, 1, 0, 12); AIC=31827.857,
BIC=31862.151, Fit time=24.812 seconds
Fit ARIMA: order=(1, 1, 1) seasonal_order=(1, 1, 2, 12); AIC=29251.591,
BIC=29299.603, Fit time=113.828 seconds
Near non-invertible roots for order (1, 1, 1)(1, 1, 2, 12); setting score to inf
(at least one inverse root too close to the border of the unit circle: 1.000)
Fit ARIMA: order=(1, 1, 0) seasonal_order=(0, 1, 1, 12); AIC=29245.339,
BIC=29272.775, Fit time=21.744 seconds
Near non-invertible roots for order (1, 1, 0)(0, 1, 1, 12); setting score to inf
(at least one inverse root too close to the border of the unit circle: 1.000)
Fit ARIMA: order=(2, 1, 1) seasonal_order=(0, 1, 1, 12); AIC=29248.990,
BIC=29290.144, Fit time=42.218 seconds
Near non-invertible roots for order (2, 1, 1)(0, 1, 1, 12); setting score to inf
(at least one inverse root too close to the border of the unit circle: 1.000)
Fit ARIMA: order=(1, 1, 2) seasonal_order=(0, 1, 1, 12); AIC=29249.266,
```

```
BIC=29290.420, Fit time=27.814 seconds
Near non-invertible roots for order (1, 1, 2)(0, 1, 1, 12); setting score to inf
(at least one inverse root too close to the border of the unit circle: 1.000)
Fit ARIMA: order=(0, 1, 0) seasonal_order=(0, 1, 1, 12); AIC=29246.479,
BIC=29267.056, Fit time=16.105 seconds
Near non-invertible roots for order (0, 1, 0)(0, 1, 1, 12); setting score to inf
(at least one inverse root too close to the border of the unit circle: 1.000)
Fit ARIMA: order=(0, 1, 2) seasonal_order=(0, 1, 1, 12); AIC=29246.706,
BIC=29281.001, Fit time=66.547 seconds
Near non-invertible roots for order (0, 1, 2)(0, 1, 1, 12); setting score to inf
(at least one inverse root too close to the border of the unit circle: 1.000)
Fit ARIMA: order=(2, 1, 0) seasonal_order=(0, 1, 1, 12); AIC=29246.720,
BIC=29281.014, Fit time=74.883 seconds
Near non-invertible roots for order (2, 1, 0)(0, 1, 1, 12); setting score to inf
(at least one inverse root too close to the border of the unit circle: 1.000)
Fit ARIMA: order=(2, 1, 2) seasonal_order=(0, 1, 1, 12); AIC=29251.232,
BIC=29299.244, Fit time=62.066 seconds
Near non-invertible roots for order (2, 1, 2)(0, 1, 1, 12); setting score to inf
(at least one inverse root too close to the border of the unit circle: 1.000)
Total fit time: 648.082 seconds
```

[322]:
```
# Plot
valid['Predictions']=0
valid['Predictions']=forecast
plt.plot(train['Adj. Close'])
plt.plot(valid[['Adj. Close','Predictions']])
```

```
/Users/yanlanLIANG/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:1: SettingWithCopyWarning:


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: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/Users/yanlanLIANG/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:2: SettingWithCopyWarning:


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: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[322]:
```
[<matplotlib.lines.Line2D at 0x1c4d8074a8>,
 <matplotlib.lines.Line2D at 0x1c4c2b2198>]
```



The red line is the Adj. Close price for the train set, the blue line is the Adj. Close price for the test set, the purple line is the Adj. Close price prediction for the test set.

9.0.1 Root Mean Square Error of AA Model

[325]:
```
rms_AA=np.sqrt(np.mean(np.power((np.array(valid['Adj. Close'])-np.
↪array(forecast['Prediction'])),2)))
```

[ ]:
```
rms_AA
```

9.1 Summary

The Auto Arima method captures the increasing trend better than linear regression with a smaller root mean of square error 42.258

10 Long Short Term Memory (LSTM)

LSTMs are widely used for sequence prediction problems and have proven to be extremely effective. The reason they work so well is because LSTM is able to store past information that is important, and forget the information that is not. LSTM has three gates:

The input gate: The input gate adds information to the cell state The forget gate: It removes the information that is no longer required by the model The output gate: Output Gate at LSTM selects the information to be shown as output

For now, let us implement LSTM as a black box and check it's performance on our particular data.

[353]:
```
# importing required libraries
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM


df = apple_dataset
# Sorting the dataset in ascending order
data = df.sort_index(ascending=True, axis=0)

# Creating a separate dataset
new_data = pd.DataFrame(index=range(0, len(df)), columns=['Date', 'Adj. Close'])

for i in range(0, len(data)):
    new_data['Date'][i] = data['Date'][i]
    new_data['Adj. Close'][i] = data['Adj. Close'][i]


# setting index
new_data.index = new_data.Date
new_data.drop('Date', axis=1, inplace=True)

# creating train and test sets as before
dataset = new_data.values
train = dataset[0:7050,:]
valid = dataset[7050:,:]

# converting dataset into x_train and y_train
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(dataset)

x_train, y_train = [], []
for i in range(30,len(train)):
    x_train.append(scaled_data[i-30:i,0])
    y_train.append(scaled_data[i,0])
x_train, y_train = np.array(x_train), np.array(y_train)

x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))

# create and fit the LSTM network
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.
 ↪shape[1],1)))
model.add(LSTM(units=50))
model.add(Dense(1))

model.compile(loss='mean_squared_error', optimizer='adam')
```

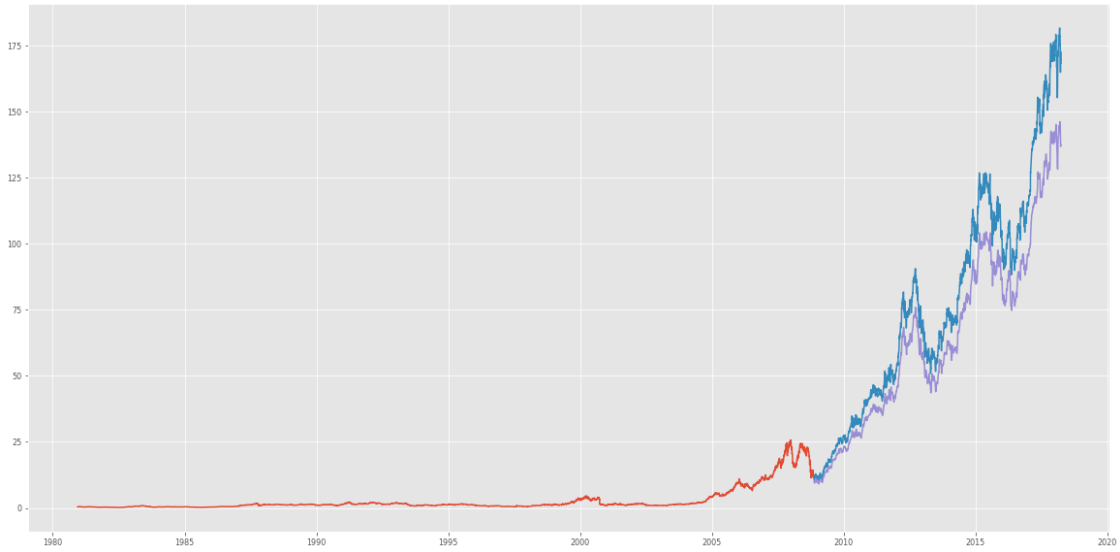
```
model.fit(x_train, y_train, epochs=1, batch_size=1, verbose=2)

# predicting 90 values, using past 30 from the train data
inputs = new_data[len(new_data) - len(valid) - 30:].values
inputs = inputs.reshape(-1,1)
inputs  = scaler.transform(inputs)

X_test = []
for i in range(30,inputs.shape[0]):
    X_test.append(inputs[i-30:i,0])
X_test = np.array(X_test)

X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
closing_price = model.predict(X_test)
closing_price = scaler.inverse_transform(closing_price)
```

```
Epoch 1/1
 - 219s - loss: 1.9082e-05
[[  9.913641]
 [  9.592852]
 [  9.391266]
 …
 [139.31696 ]
 [136.86789 ]
 [137.4223  ]]
```

[357]:
```
# plot
train = new_data[:7050]
valid = new_data[7050:]
valid['Predictions']=closing_price
plt.plot(train['Adj. Close'])
plt.plot(valid[['Adj. Close','Predictions']])
```

```
/Users/yanlanLIANG/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:4: SettingWithCopyWarning:


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: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[357]:
```
[<matplotlib.lines.Line2D at 0x1a48e7f2b0>,
 <matplotlib.lines.Line2D at 0x1c54c28198>]
```



The red line is the Adj. Close price for the train set, the blue line is the Adj. Close price for the test set, the purple line is the Adj. Close price prediction for the test set.

10.0.1 Root Mean Square Error of LSTM Model

[348]:
```
rms_LSTM=np.sqrt(np.mean(np.power((valid-closing_price),2)))
```

[349]:
```
rms_LSTM
```

[349]:
```
6.755877678410655
```

10.1 Summary

The LSTM Model method could be tuned for various parameters such as changing the number of LSTM layers, adding dropout value or increasing the number of epochs. Hence, LSTM has the smallest RMSE rate so far, which is 6.756.

11 Conclusion:

In this project, we applied different machine learnign to predict Apple's future stock in a short term. Here is the summary of the models' perfomance based on RSME. Model, RSME: Moving Average Model, 83.834 Linear Regression Model, 80.828 Auto Arima Model, 42.258 Long Short Term Memory Model, 6.756

11.1 To conclude:

We could observe that the LSTM Model fitted the dataset pretty well with a low RSME rate. But are the predictions from LSTM enough to identify whether the stock price will increase or decrease? Certainly not! As we mentioned before, the stock price is affected by the news about

the company and other factors like demonetization or merger/demerger of the companies. Besides, politics, public protest, and the stock market itself could affect the stock price. There are certain intangible factors as well, which can often be impossible to predict.

However, if investors still want to make short-term investments in Apple, we recommend them to use Long Short Term Memory Model to make the prediction. Base on the forecast from LSTM Model, the investors, could decide the number of shares to buy and sell shares to increase their chances to have a return on their investments. However, as we mentioned above, many factors affect the stock price to fluctuate in the short term. Therefore, instead of trading frequently, we recommend the investors to hold their shares for the long run, they would have a higher chance to have a high return.

12 Improvement:

1. Try out different machine learning algorithms:There are MANY machine learning algorithms out there that are very good. We only used a small subset of them.
2. Some model like Linear Regression model could tune some features to fit the dataset better.
3. We took Apple as the example. It can't represent the whole market. When we tried to measure the accuracy of the integrated dataset, We only took very small amount of data(Dec 12, 1980 through Mar 27, 2018)