What's going to happen to apple stock!

An attempt to predict market behavior, utilizing Machine Learning

Bita &

Parmis

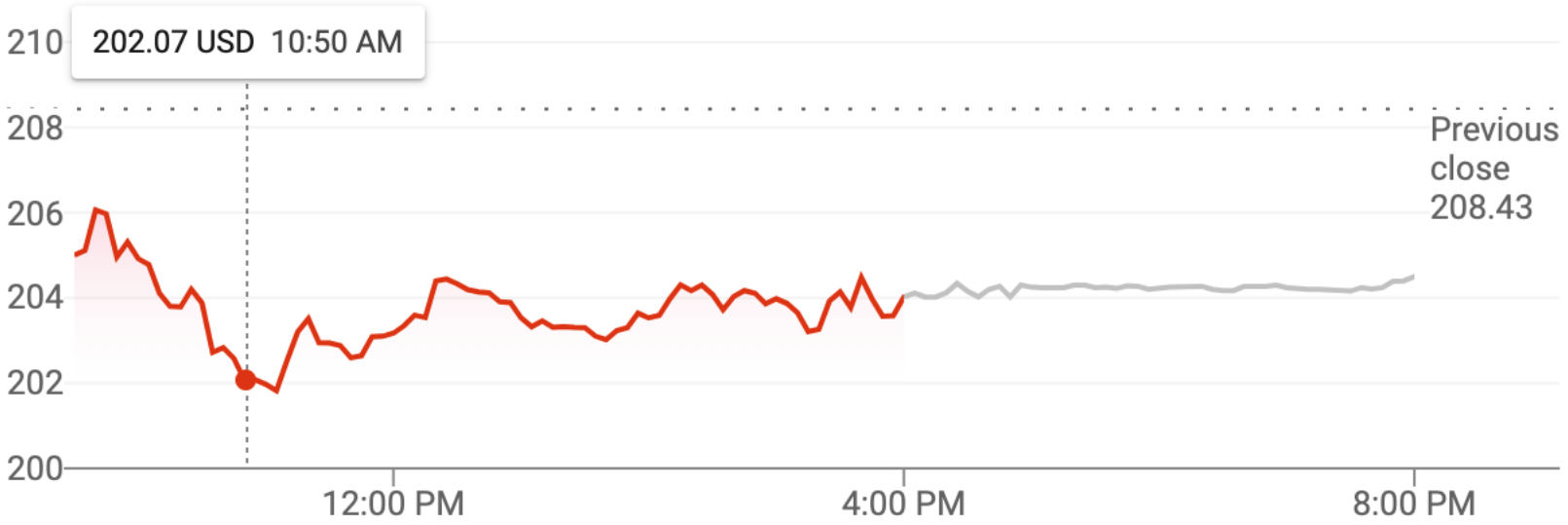
Apple Stock today

Market Summary > Apple Inc.
NASDAQ: AAPL

+ Follow

204.02 USD −4.41 (2.12%) ↓
Closed: Aug 2, 7:58 PM EDT · Disclaimer
After hours 204.50 +0.48 (0.24%)

1 day | 5 days | 1 month | 6 months | YTD | 1 year | 5 years | Max



Open	205.53	Div yield	1.51%
High	206.43	Prev close	208.43
Low	201.63	52-wk high	233.47
Mkt cap	922.00B	52-wk low	142.00
P/E ratio	17.38		

[7]: [<matplotlib.lines.Line2D at 0x224196c3b00>]

Code

[77]:
```
short_rolling = df['Close'].rolling(window=10).mean()
short_rolling.head(30)
```

[77]:
```
0            NaN
1            NaN
2            NaN
3            NaN
4            NaN
5            NaN
6            NaN
7            NaN
8            NaN
9     103.518000
10    104.621001
11    105.835001
12    106.340001
13    106.699001
14    107.096001
15    107.439001
16    107.929001
17    108.272001
18    108.593001
19    108.781001
20    108.795000
21    108.799000
22    108.802000
23    108.766000
24    108.642000
25    108.376000
26    108.038000
27    107.726000
28    107.491000
29    107.328000
Name: Close, dtype: float64
```

[81]:
```
short_rolling.to_csv(r'Short.csv')
```

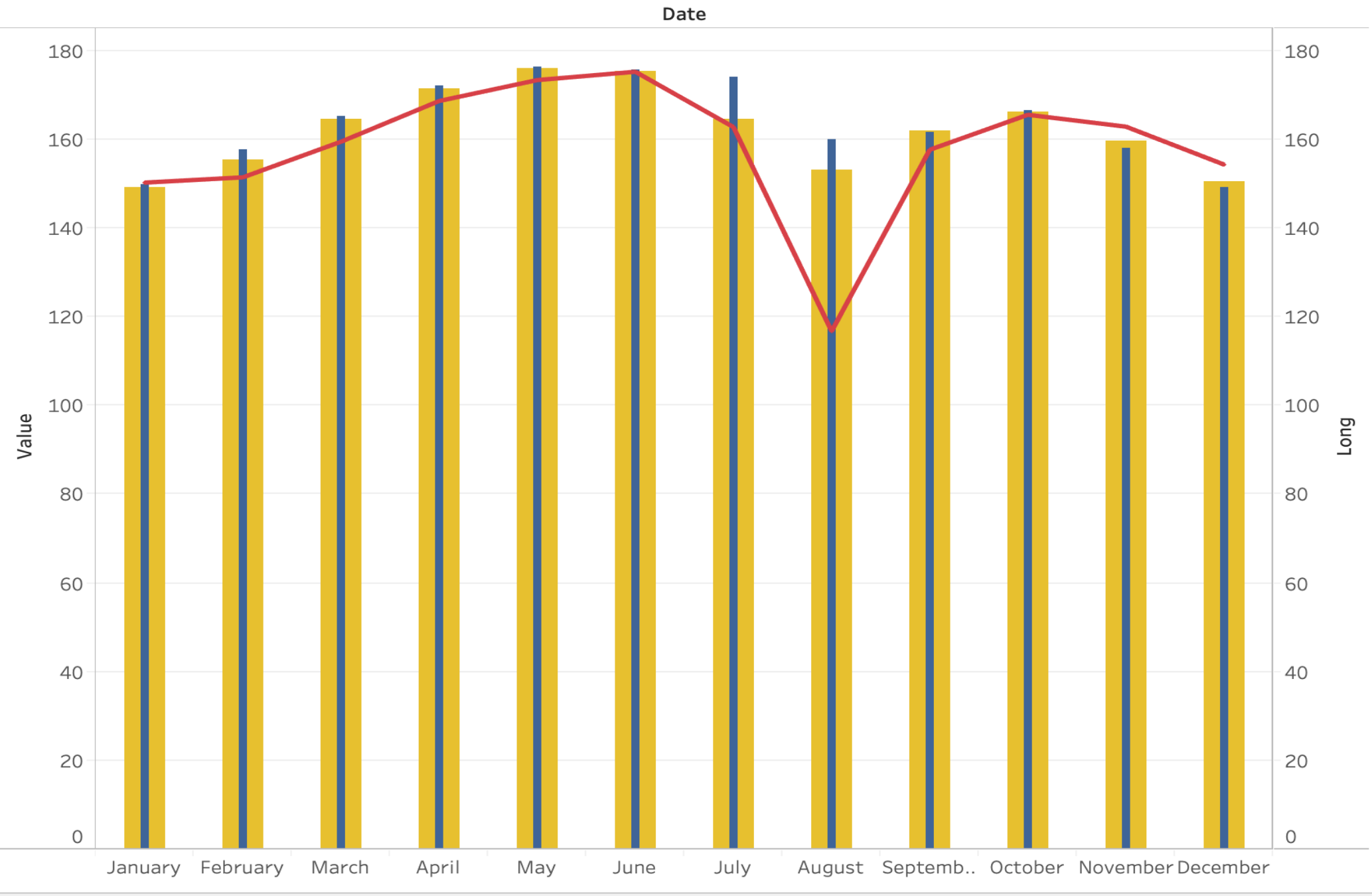
[78]:
```
long_rolling = df['Close'].rolling(window=30).mean()
long_rolling.tail()
```

[78]:
```
751    199.415001
752    199.984001
753    200.526001
754    200.988001
755    201.415668
Name: Close, dtype: float64
```

[82]:
```
long_rolling.to_csv(r'long.csv')
```

- Moving Average model was build to replicate the performance of historical data.
- Two moving averages were built, short term and long term windows to show the efficiency of the model.

Moving Averages and Actual Data



Measure Names

- Avg. Close
- Avg. Short
- Long

Measure Names

- Avg. Close
- Avg. Short

[1]:
```
#import packages
import pandas as pd
import numpy as np
import datetime
#to plot within notebook
import matplotlib.pyplot as plt
%matplotlib inline

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

#for normalizing data
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))

#read the file
df = pd.read_csv('AAPL.csv')

#print the head
df.head()
```

[1]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2016-07-25	98.250000	98.839996	96.919998	97.339996	92.638840	40382900
1	2016-07-26	96.820000	97.970001	96.419998	96.669998	92.001190	56239800
2	2016-07-27	104.269997	104.349998	102.750000	102.949997	97.977890	92344800
3	2016-07-28	102.830002	104.449997	102.820000	104.339996	99.300751	39869800
4	2016-07-29	104.190002	104.550003	103.680000	104.209999	99.177032	27733700

[2]:
```
#creating dataframe with date and the target variable
data = df.sort_index(ascending=True, axis=0)
new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close'])

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

[3]:
```
#splitting into train and validation
train = new_data[:700]
valid = new_data[700:]
```

[4]:
```
new_data.shape, train.shape, valid.shape
```

[4]:
```
((756, 2), (700, 2), (56, 2))
```

Code

[5]:
```
#setting index as date values
df['Date'] = pd.to_datetime(df.Date,format='%Y-%m-%d')
df.index = df['Date']

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

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

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

[6]:
```
train['Date'].min(), train['Date'].max(), valid['Date'].min(), valid['Date'].max()
```

[6]:
```
('2016-07-25', '2019-05-06', '2019-05-07', '2019-07-25')
```

[7]:
```
preds = []
for i in range(0,56):
    a = train['Close'][len(train)-56+i:].sum() + sum(preds)
    b = a/56
    preds.append(b)
```

[8]:
```
rms=np.sqrt(np.mean(np.power((np.array(valid['Close'])-preds),2)))
rms
```

[8]:
```
9.158918318956124
```

[9]:
```
valid['Predictions'] = 0
valid['Predictions'] = preds
plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
```

Code

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

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

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

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

[6]:
```
train['Date'].min(), train['Date'].max(), valid['Date'].min(), valid['Date'].max()
```

[6]:
```
('2016-07-25', '2019-05-06', '2019-05-07', '2019-07-25')
```

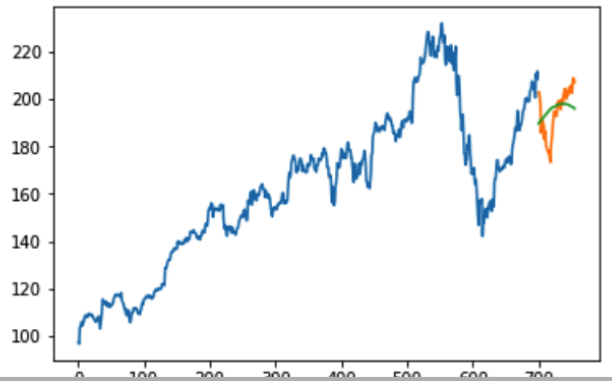
[7]:
```
preds = []
for i in range(0,56):
    a = train['Close'][len(train)-56+i:].sum() + sum(preds)
    b = a/56
    preds.append(b)
```

[8]:
```
rms=np.sqrt(np.mean(np.power((np.array(valid['Close'])-preds),2)))
rms
```

[8]:
```
9.158918318956124
```

[9]:
```
valid['Predictions'] = 0
valid['Predictions'] = preds
plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
```

[9]:
```
[<matplotlib.lines.Line2D at 0x1a2389c198>,
 <matplotlib.lines.Line2D at 0x1a2389c2e8>]
```



MovingAverages.ipynb | LinearRegression.ipynb | Untitled1.ipynb | Stock Analysis .ipynb | APPL Stock Analysis.i

Code

Linear Regression

[10]:
```
#setting index as date values
df['Date'] = pd.to_datetime(df.Date,format='%Y-%m-%d')
df.index = df['Date']

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

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

for i in range(0,len(data)):
    new_data['Date'][i] = data['Date'][i]
    new_data['Close'][i] = data['Close'][i]
new_data['Date']= pd.to_datetime(new_data['Date'])
new_data=new_data.set_index('Date')
```

[11]:
```
new_data['Dayofweek'] = 1
for i in range(len(new_data['Date'])):
    new_data['Dayofweek'][i] = new_data['Date'][i].weekday()


new_data['mon_fri'] = 0
for i in range(0,len(new_data)):
    if (new_data['Dayofweek'][i] == 0 or new_data['Dayofweek'][i] == 4):
        new_data['mon_fri'][i] = 1
    else:
        new_data['mon_fri'][i] = 0
```

```
new_data.head()
```

[12]:

Date	Close	Dayofweek
2016-07-25	97.34	1
2016-07-26	96.67	1
2016-07-27	102.95	1
2016-07-28	104.34	1
2016-07-29	104.21	1

[ ]:

[13]:
```
#split into train and validation
train = new_data[:700]
valid = new_data[700:]

x_train = train.drop('Close', axis=1)
y_train = train['Close']
x_valid = valid.drop('Close', axis=1)
y_valid = valid['Close']

#implement linear regression
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train,y_train)
```

[13]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

[14]:
```
#make predictions and find the rmse
preds = model.predict(x_valid)
rms=np.sqrt(np.mean(np.power((np.array(y_valid)-np.array(preds)),2)))
rms
```

[14]:
```
34.51278666092229
```

[15]:
```
#plot
valid['Predictions'] = 0
valid['Predictions'] = preds

valid.index = new_data[700:].index
train.index = new_data[:700].index

plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
```

```
/Users/prmis/anaconda3/envs/PythonData/lib/python3.6/site-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
```

[15]:
```
#plot
valid['Predictions'] = 0
valid['Predictions'] = preds

valid.index = new_data[700:].index
train.index = new_data[:700].index

plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
```

```
/Users/prmis/anaconda3/envs/PythonData/lib/python3.6/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/indexing.html#indexing-view-versus-copy

/Users/prmis/anaconda3/envs/PythonData/lib/python3.6/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/indexing.html#indexing-view-versus-copy
  This is separate from the ipykernel package so we can avoid doing imports until
```

[15]:
```
[<matplotlib.lines.Line2D at 0x1a2416a9e8>,
 <matplotlib.lines.Line2D at 0x1a2416ab38>]
```

[ ]:

Take Aways

- Utilizing Sklearn, Keras and LSTM gave us the most efficient model that was able to at least replicate the behavior of our data with a better fit. LSTM has the most efficiency when it comes to non-linear data!!!
- Our second best model with a better RMSE was Moving averages due to the nature of our data set.
- Fastai was utilized heavily in future predictions, but due to it's conflict with Python 3.7 we were unable to utilize it.
- Over all, stock prices our dependent on many factors and in this trial we simplified our variables, therefore our models were not able to provide the best fir possible.