Predict Stock Prices using Time Series Analysis by Patrick BENIE

April 6, 2021

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
df=pd.read_csv('/Users/patrickslearningprogams/Desktop/Python Projects/Stock Price Predictor/BAJFINANCE.csv')
df.head()
```

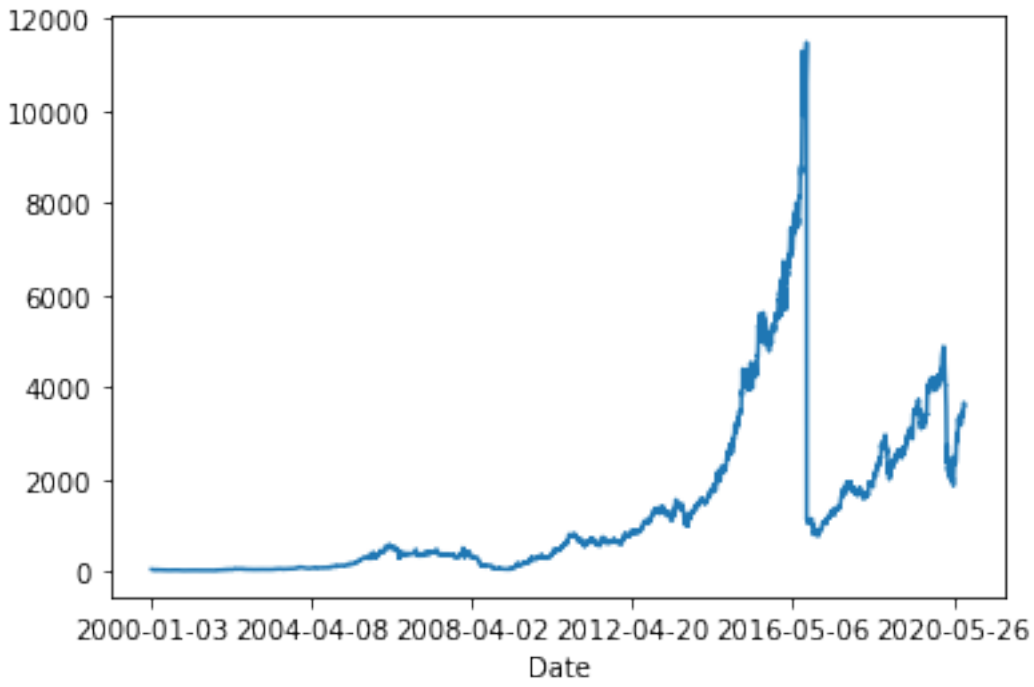
[2]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trades	Deliverable Volume	%Deliverble
0	2000-01-03	BAJAUTOFIN	EQ	46.95	49.45	50.75	46.5	50.75	50.75	50.05	7600	3.803800e+10	NaN	NaN	NaN
1	2000-01-04	BAJAUTOFIN	EQ	50.75	53.20	53.20	47.9	48.00	48.10	48.56	5000	2.428000e+10	NaN	NaN	NaN
2	2000-01-05	BAJAUTOFIN	EQ	48.10	46.55	47.40	44.6	44.60	44.60	45.47	3500	1.591450e+10	NaN	NaN	NaN
3	2000-01-06	BAJAUTOFIN	EQ	44.60	43.50	46.00	42.1	46.00	45.25	44.43	6200	2.754750e+10	NaN	NaN	NaN
4	2000-01-07	BAJAUTOFIN	EQ	45.25	48.00	48.00	42.0	42.90	42.90	44.44	3500	1.555550e+10	NaN	NaN	NaN

[3]:
```
#since it's a time series use case, everything that happens in the data must be related to the time series index
#assign the 'Date' column to the time series index
df.set_index('Date',inplace=True) #inplace parameter used to update dataframe
```

[4]:
```
#VWAP volume weigthed average price
#check the trend of VWAP
#Plotting the target variable VWAP over time
df['VWAP'].plot()
```

[4]:
```
<AxesSubplot:xlabel='Date'>
```



[ ]:

[5]:
```
#Data processing
```

[6]:
```
df.shape
```

[6]:
```
(5070, 14)
```

[7]:
```
#checking for missing values
df.isna().sum()
```

[7]:
```
Symbol                0
Series                0
Prev Close            0
Open                  0
High                  0
Low                   0
Last                  0
Close                 0
VWAP                  0
Volume                0
Turnover              0
Trades             2779
Deliverable Volume  446
```

```
%Deliverble            446
dtype: int64
```

[8]:
```
df.dropna(inplace=True)
```

[9]:
```
df.isna().sum()
```

[9]:
```
Symbol                0
Series                0
Prev Close            0
Open                  0
High                  0
Low                   0
Last                  0
Close                 0
VWAP                  0
Volume                0
Turnover              0
Trades                0
Deliverable Volume    0
%Deliverble           0
dtype: int64
```

[10]:
```
df.shape
```

[10]:
```
(2291, 14)
```

[11]:
```
#create a copy of df
data=df.copy()
```

[12]:
```
data.dtypes
```

[12]:
```
Symbol                object
Series                object
Prev Close           float64
Open                 float64
High                 float64
Low                  float64
Last                 float64
Close                float64
VWAP                 float64
Volume                 int64
Turnover             float64
Trades               float64
Deliverable Volume   float64
%Deliverble          float64
dtype: object
```

[13]:
```
data.columns
```

[13]:
```
Index(['Symbol', 'Series', 'Prev Close', 'Open', 'High', 'Low', 'Last',
       'Close', 'VWAP', 'Volume', 'Turnover', 'Trades', 'Deliverable Volume',
       '%Deliverble'],
      dtype='object')
```

[14]:
```
#we choose 5 features(columns) for training purposes
lag_features=['High','Low','Volume','Turnover','Trades']
window1=3
window2=7
```

[15]:
```
#mean
for feature in lag_features:
    data[feature+'rolling_mean_3']=data[feature].rolling(window=window1).mean()
    data[feature+'rolling_mean_7']=data[feature].rolling(window=window2).mean()
```

[16]:
```
#standardisation
for feature in lag_features:
    data[feature+'rolling_std_3']=data[feature].rolling(window=window1).std()
    data[feature+'rolling_std_7']=data[feature].rolling(window=window2).std()
```

[17]:
```
data.head()
```

[17]:

Date	Symbol	Series	Prev Close	Open	High	Low	Last
2011-06-01	BAJFINANCE	EQ	616.70	617.00	636.50	616.00	627.00
2011-06-02	BAJFINANCE	EQ	631.85	625.00	638.90	620.00	634.00
2011-06-03	BAJFINANCE	EQ	633.45	625.15	637.80	620.00	623.00
2011-06-06	BAJFINANCE	EQ	625.00	620.00	641.00	611.35	611.35
2011-06-07	BAJFINANCE	EQ	614.00	604.00	623.95	604.00	619.90

Date	Close	VWAP	Volume	…	Highrolling_std_3	Highrolling_std_7
2011-06-01	631.85	627.01	6894	…	NaN	NaN
2011-06-02	633.45	636.04	2769	…	NaN	NaN
2011-06-03	625.00	625.09	51427	…	1.201388	NaN
2011-06-06	614.00	616.03	5446	…	1.625833	NaN
2011-06-07	619.15	617.73	5991	…	9.062422	NaN

Date	Lowrolling_std_3	Lowrolling_std_7	Volumerolling_std_3
2011-06-01	NaN	NaN	NaN
2011-06-02	NaN	NaN	NaN
2011-06-03	2.309401	NaN	26980.871860
2011-06-06	4.994080	NaN	27352.695339
2011-06-07	8.008797	NaN	26391.221653

```
            Volumerolling_std_7  Turnoverrolling_std_3  Turnoverrolling_std_7  \
Date
2011-06-01                  NaN                    NaN                    NaN
2011-06-02                  NaN                    NaN                    NaN
2011-06-03                  NaN           1.685217e+12                    NaN
2011-06-06                  NaN           1.710136e+12                    NaN
2011-06-07                  NaN           1.652377e+12                    NaN

            Tradesrolling_std_3  Tradesrolling_std_7
Date
2011-06-01                  NaN                 NaN
2011-06-02                  NaN                 NaN
2011-06-03           670.500559                 NaN
2011-06-06           148.769396                 NaN
2011-06-07            78.270897                 NaN

[5 rows x 34 columns]
```

[18]:
```
data.columns
```

[18]:
```
Index(['Symbol', 'Series', 'Prev Close', 'Open', 'High', 'Low', 'Last',
       'Close', 'VWAP', 'Volume', 'Turnover', 'Trades', 'Deliverable Volume',
       '%Deliverble', 'Highrolling_mean_3', 'Highrolling_mean_7',
       'Lowrolling_mean_3', 'Lowrolling_mean_7', 'Volumerolling_mean_3',
       'Volumerolling_mean_7', 'Turnoverrolling_mean_3',
       'Turnoverrolling_mean_7', 'Tradesrolling_mean_3',
       'Tradesrolling_mean_7', 'Highrolling_std_3', 'Highrolling_std_7',
       'Lowrolling_std_3', 'Lowrolling_std_7', 'Volumerolling_std_3',
       'Volumerolling_std_7', 'Turnoverrolling_std_3', 'Turnoverrolling_std_7',
       'Tradesrolling_std_3', 'Tradesrolling_std_7'],
      dtype='object')
```

[19]:
```
data.shape
```

[19]:
```
(2291, 34)
```

[20]:
```
data.isna().sum()
```

[20]:
```
Symbol                0
Series                0
Prev Close            0
Open                  0
High                  0
Low                   0
Last                  0
Close                 0
```

```
VWAP                    0
Volume                  0
Turnover                0
Trades                  0
Deliverable Volume      0
%Deliverble             0
Highrolling_mean_3      2
Highrolling_mean_7      6
Lowrolling_mean_3       2
Lowrolling_mean_7       6
Volumerolling_mean_3    2
Volumerolling_mean_7    6
Turnoverrolling_mean_3  2
Turnoverrolling_mean_7  6
Tradesrolling_mean_3    2
Tradesrolling_mean_7    6
Highrolling_std_3       2
Highrolling_std_7       6
Lowrolling_std_3        2
Lowrolling_std_7        6
Volumerolling_std_3     2
Volumerolling_std_7     6
Turnoverrolling_std_3   2
Turnoverrolling_std_7   6
Tradesrolling_std_3     2
Tradesrolling_std_7     6
dtype: int64
```

[21]:
```
data.dropna(inplace=True)
```

[22]:
```
data.isna().sum()
```

[22]:
```
Symbol                  0
Series                  0
Prev Close              0
Open                    0
High                    0
Low                     0
Last                    0
Close                   0
VWAP                    0
Volume                  0
Turnover                0
Trades                  0
Deliverable Volume      0
%Deliverble             0
Highrolling_mean_3      0
```

```
Highrolling_mean_7        0
Lowrolling_mean_3         0
Lowrolling_mean_7         0
Volumerolling_mean_3      0
Volumerolling_mean_7      0
Turnoverrolling_mean_3    0
Turnoverrolling_mean_7    0
Tradesrolling_mean_3      0
Tradesrolling_mean_7      0
Highrolling_std_3         0
Highrolling_std_7         0
Lowrolling_std_3          0
Lowrolling_std_7          0
Volumerolling_std_3       0
Volumerolling_std_7       0
Turnoverrolling_std_3     0
Turnoverrolling_std_7     0
Tradesrolling_std_3       0
Tradesrolling_std_7       0
dtype: int64
```

[23]:
```
data.columns
```

[23]:
```
Index(['Symbol', 'Series', 'Prev Close', 'Open', 'High', 'Low', 'Last',
       'Close', 'VWAP', 'Volume', 'Turnover', 'Trades', 'Deliverable Volume',
       '%Deliverble', 'Highrolling_mean_3', 'Highrolling_mean_7',
       'Lowrolling_mean_3', 'Lowrolling_mean_7', 'Volumerolling_mean_3',
       'Volumerolling_mean_7', 'Turnoverrolling_mean_3',
       'Turnoverrolling_mean_7', 'Tradesrolling_mean_3',
       'Tradesrolling_mean_7', 'Highrolling_std_3', 'Highrolling_std_7',
       'Lowrolling_std_3', 'Lowrolling_std_7', 'Volumerolling_std_3',
       'Volumerolling_std_7', 'Turnoverrolling_std_3', 'Turnoverrolling_std_7',
       'Tradesrolling_std_3', 'Tradesrolling_std_7'],
      dtype='object')
```

[24]:
```
#independent features
ind_features=['Highrolling_mean_3', 'Highrolling_mean_7',
       'Lowrolling_mean_3', 'Lowrolling_mean_7', 'Volumerolling_mean_3',
       'Volumerolling_mean_7', 'Turnoverrolling_mean_3',
       'Turnoverrolling_mean_7', 'Tradesrolling_mean_3',
       'Tradesrolling_mean_7', 'Highrolling_std_3', 'Highrolling_std_7',
       'Lowrolling_std_3', 'Lowrolling_std_7', 'Volumerolling_std_3',
       'Volumerolling_std_7', 'Turnoverrolling_std_3', 'Turnoverrolling_std_7',
       'Tradesrolling_std_3', 'Tradesrolling_std_7']
```

[25]:
```
training_data=data[0:1800] # about 80% of the dataset
test_data=data[1800:]
```

[26]:
```
training_data
```

[26]:

```
               Symbol Series  Prev Close     Open     High      Low     Last  \
Date
2011-06-09  BAJFINANCE     EQ      635.60   639.80   647.00   630.00   630.00
2011-06-10  BAJFINANCE     EQ      631.10   641.85   648.25   618.55   621.10
2011-06-13  BAJFINANCE     EQ      622.20   616.00   627.85   616.00   622.75
2011-06-14  BAJFINANCE     EQ      624.95   625.00   628.95   619.95   621.20
2011-06-15  BAJFINANCE     EQ      622.10   612.00   623.00   598.10   605.00
…                  …    …          …        …        …        …      …
2018-09-04  BAJFINANCE     EQ     2724.05  2724.00  2777.65  2683.50  2748.00
2018-09-05  BAJFINANCE     EQ     2746.30  2740.15  2764.80  2668.00  2704.45
2018-09-06  BAJFINANCE     EQ     2716.90  2729.00  2731.50  2671.40  2672.20
2018-09-07  BAJFINANCE     EQ     2684.10  2698.40  2751.40  2672.60  2745.00
2018-09-10  BAJFINANCE     EQ     2744.20  2732.00  2738.00  2596.00  2607.60

              Close     VWAP   Volume  …  Highrolling_std_3  \
Date                                   …
2011-06-09   631.10   638.27    31252  …          12.769789
2011-06-10   622.20   634.16    30885  …           1.639360
2011-06-13   624.95   622.92     3981  …          11.434196
2011-06-14   622.10   625.35     5597  …          11.473593
2011-06-15   601.70   606.90    12590  …           3.165833
…              …       …      …  …                …
2018-09-04  2746.30  2726.23  2606992  …          88.954937
2018-09-05  2716.90  2712.53  1728455  …          63.129081
2018-09-06  2684.10  2695.89  1147879  …          23.818183
2018-09-07  2744.20  2716.32  1264436  …          16.755397
2018-09-10  2615.65  2655.39  1570179  …          10.147413

            Highrolling_std_7  Lowrolling_std_3  Lowrolling_std_7  \
Date
2011-06-09           7.494911         15.011107          9.410145
2011-06-10           8.227994         13.030765          9.501961
2011-06-13           9.497080          7.456597          9.298317
2011-06-14          10.198891          2.002707          9.293713
2011-06-15          11.352292         11.643560         11.262712
…                         …                 …                 …
2018-09-04          79.489416         83.341306        107.041856
2018-09-05         100.594924         24.113551        119.854378
2018-09-06         113.135709          8.146778        118.187686
2018-09-07         106.101111          2.386071        100.988340
2018-09-10          84.670766         43.882722         74.650851

            Volumerolling_std_3  Volumerolling_std_7  Turnoverrolling_std_3  \
Date
2011-06-09         13497.047986         18373.894011           8.665128e+11
```

```
2011-06-10          2434.970705         18047.331029          1.579562e+11
2011-06-13         15640.051929         17835.642665          9.982094e+11
2011-06-14         15088.183102         13139.472184          9.595224e+11
2011-06-15          4575.812970         12469.137006          2.733211e+11
…                           …                    …                     …
2018-09-04        696998.737787        642979.884199          1.827289e+14
2018-09-05        483305.554092        570427.041296          1.375623e+14
2018-09-06        734609.476628        584307.816124          2.020435e+14
2018-09-07        307128.544854        599913.723573          8.394860e+13
2018-09-10        218098.451125        592650.415369          5.493903e+13

            Turnoverrolling_std_7  Tradesrolling_std_3  Tradesrolling_std_7
Date
2011-06-09           1.155773e+12           354.841279           463.942320
2011-06-10           1.136003e+12           272.875429           448.020620
2011-06-13           1.123423e+12           557.373603           464.778596
2011-06-14           8.397765e+11           656.385050           455.234163
2011-06-15           7.983186e+11           150.639747           436.941971
…                             …                    …                     …
2018-09-04           1.673270e+14          2794.744413         21708.856675
2018-09-05           1.467744e+14         21770.352164         17267.163548
2018-09-06           1.561692e+14         30923.312263         24013.596897
2018-09-07           1.641199e+14         12123.301421         25604.694191
2018-09-10           1.647685e+14         15501.359822         25567.783116

[1800 rows x 34 columns]
```

[ ]:

[27]:
```
!pip install pmdarima
```

```
Requirement already satisfied: pmdarima in
/Applications/anaconda3/lib/python3.8/site-packages (1.8.0)
Requirement already satisfied: statsmodels!=0.12.0,>=0.11 in
/Applications/anaconda3/lib/python3.8/site-packages (from pmdarima) (0.12.2)
Requirement already satisfied: scikit-learn>=0.22 in
/Applications/anaconda3/lib/python3.8/site-packages (from pmdarima) (0.23.2)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in
/Applications/anaconda3/lib/python3.8/site-packages (from pmdarima)
(50.3.1.post20201107)
Requirement already satisfied: pandas>=0.19 in
/Applications/anaconda3/lib/python3.8/site-packages (from pmdarima) (1.1.3)
Requirement already satisfied: numpy>=1.17.3 in
/Applications/anaconda3/lib/python3.8/site-packages (from pmdarima) (1.19.2)
Requirement already satisfied: joblib>=0.11 in
/Applications/anaconda3/lib/python3.8/site-packages (from pmdarima) (0.17.0)
Requirement already satisfied: scipy>=1.3.2 in
```

```
/Applications/anaconda3/lib/python3.8/site-packages (from pmdarima) (1.5.2)
Requirement already satisfied: urllib3 in
/Applications/anaconda3/lib/python3.8/site-packages (from pmdarima) (1.25.11)
Requirement already satisfied: Cython<0.29.18,>=0.29 in
/Applications/anaconda3/lib/python3.8/site-packages (from pmdarima) (0.29.17)
Requirement already satisfied: patsy>=0.5 in
/Applications/anaconda3/lib/python3.8/site-packages (from
statsmodels!=0.12.0,>=0.11->pmdarima) (0.5.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/Applications/anaconda3/lib/python3.8/site-packages (from scikit-
learn>=0.22->pmdarima) (2.1.0)
Requirement already satisfied: pytz>=2017.2 in
/Applications/anaconda3/lib/python3.8/site-packages (from
pandas>=0.19->pmdarima) (2020.1)
Requirement already satisfied: python-dateutil>=2.7.3 in
/Applications/anaconda3/lib/python3.8/site-packages (from
pandas>=0.19->pmdarima) (2.8.1)
Requirement already satisfied: six in
/Applications/anaconda3/lib/python3.8/site-packages (from
patsy>=0.5->statsmodels!=0.12.0,>=0.11->pmdarima) (1.15.0)
```

[28]:
```
from pmdarima import auto_arima
```

[29]:
```
import warnings
warnings.filterwarnings('ignore')
```

[30]:
```
#find the best arima model
model=auto_arima(y=training_data['VWAP'],exogenous=training_data[ind_features],trace=True)
```

```
Performing stepwise search to minimize aic
 ARIMA(2,0,2)(0,0,0)[0] intercept   : AIC=20931.536, Time=4.79 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=20925.224, Time=2.57 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=20926.348, Time=2.35 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=20926.320, Time=3.12 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=32616.913, Time=2.13 sec
 ARIMA(1,0,1)(0,0,0)[0] intercept   : AIC=20929.234, Time=3.25 sec

Best model:  ARIMA(0,0,0)(0,0,0)[0] intercept
Total fit time: 18.224 seconds
```

[31]:
```
#Arima model
model.fit(training_data['VWAP'],training_data[ind_features])
```

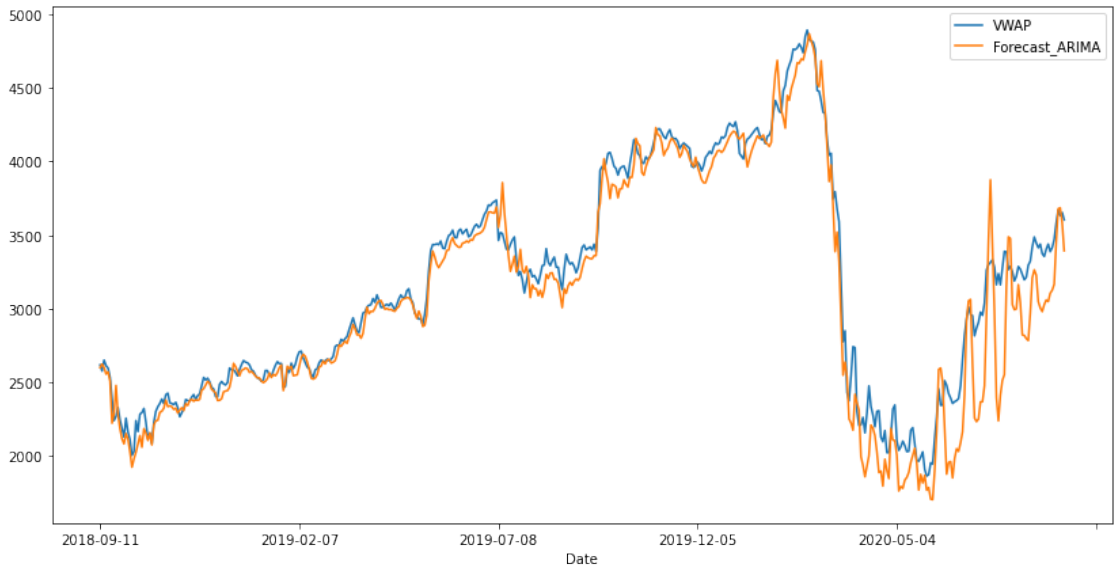
[31]:
```
ARIMA(order=(0, 0, 0), scoring_args={}, suppress_warnings=True)
```

[32]:
```
forecast=model.predict(n_periods=len(test_data),␣
 ↪exogenous=test_data[ind_features])
```

[33]:
```
test_data['Forecast_ARIMA']=forecast
```

[34]:
```
#blue line is the actual data and orange line is the prediction. we can see␣
 ↪that they're very close to each other
test_data[['VWAP','Forecast_ARIMA']].plot(figsize=(14,7))
```

[34]: <AxesSubplot:xlabel='Date'>



[ ]:

[35]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

[36]:
```
np.sqrt(mean_squared_error(test_data['VWAP'],test_data['Forecast_ARIMA']))
```

[36]: 187.77545472730867

[37]:
```
mean_absolute_error(test_data['VWAP'],test_data['Forecast_ARIMA'])
```

[37]: 124.6480740444013

[ ]: