Time_Series_Nifty_Bajaj_Deploy

Reading the market data of BAJAJFINSV stock and preparing a training dataset and validation dataset.

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

In [2]:
```
df=pd.read_csv('BAJFINANCE.csv')
df.head()
```

Out[2]:

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

In [3]:
```
df.set_index('Date',inplace=True)
```

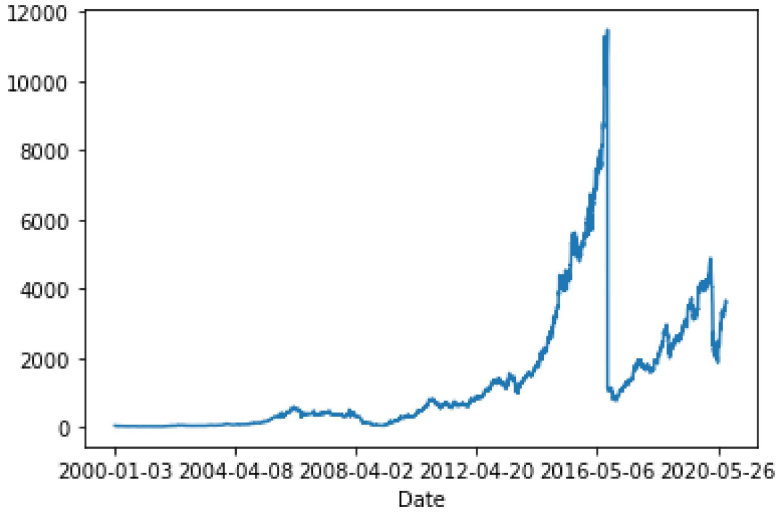
In [ ]:
```

```

Plotting the target variable VWAP over time

In [4]:
```
df['VWAP'].plot()
```

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



so u can observe here some kind of Seasonality

Feature Engineering Almost every time series problem will have some external features or some internal feature engineering to help the model. Let's add some basic features like lag values of available numeric features that are widely used for time series problems. Since we need to predict the price of the stock for a day, we cannot use the feature values of the same day since they will be unavailable at actual inference time. We need to use statistics like mean, standard deviation of their lagged values. We will use three sets of lagged values, one previous day, one looking back 7 days and another looking back 30 days as a proxy for last week and last month metrics.

Data Pre-Processing

In [5]:
```
df.shape
```

Out[5]:
```
(5070, 14)
```

In [6]:
```
df.isna().sum()
```

Out[6]:
```
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
```

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

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

Out[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
```

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

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

In [11]:
```
data=df.copy()
```

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

Out[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
```

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

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

In [14]:
```
lag_features=['High','Low','Volume','Turnover','Trades']
window1=3
window2=7
```

In [15]:
```
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()
```

In [16]:
```
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()
```

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

Out[17]:

Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	...	Highrolli
2011-06-01	BAJFINANCE	EQ	616.70	617.00	636.50	616.00	627.00	631.85	627.01	6894	...	
2011-06-02	BAJFINANCE	EQ	631.85	625.00	638.90	620.00	634.00	633.45	636.04	2769	...	
2011-06-03	BAJFINANCE	EQ	633.45	625.15	637.80	620.00	623.00	625.00	625.09	51427	...	
2011-06-06	BAJFINANCE	EQ	625.00	620.00	641.00	611.35	611.35	614.00	616.03	5446	...	
2011-06-07	BAJFINANCE	EQ	614.00	604.00	623.95	604.00	619.90	619.15	617.73	5991	...	

5 rows × 34 columns

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

Out[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')
```

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

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

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

Out[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
```

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

In [22]:
```
data.columns
```

Out[22]:
```
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')
```

In [23]:
```
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']
```

In [24]:
```
training_data=data[0:1800]
test_data=data[1800:]
```

In [25]:
```
training_data
```

Out[25]:

Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	...
2011-06-09	BAJFINANCE	EQ	635.60	639.80	647.00	630.00	630.00	631.10	638.27	31252	...
2011-06-10	BAJFINANCE	EQ	631.10	641.85	648.25	618.55	621.10	622.20	634.16	30885	...
2011-06-13	BAJFINANCE	EQ	622.20	616.00	627.85	616.00	622.75	624.95	622.92	3981	...
2011-06-14	BAJFINANCE	EQ	624.95	625.00	628.95	619.95	621.20	622.10	625.35	5597	...
2011-06-15	BAJFINANCE	EQ	622.10	612.00	623.00	598.10	605.00	601.70	606.90	12590	...
...	...	...	...	...	...	...	...	...	...	...	...
2018-09-04	BAJFINANCE	EQ	2724.05	2724.00	2777.65	2683.50	2748.00	2746.30	2726.23	2606992	...
2018-09-05	BAJFINANCE	EQ	2746.30	2740.15	2764.80	2668.00	2704.45	2716.90	2712.53	1728455	...
2018-09-06	BAJFINANCE	EQ	2716.90	2729.00	2731.50	2671.40	2672.20	2684.10	2695.89	1147879	...
2018-09-07	BAJFINANCE	EQ	2684.10	2698.40	2751.40	2672.60	2745.00	2744.20	2716.32	1264436	...
2018-09-10	BAJFINANCE	EQ	2744.20	2732.00	2738.00	2596.00	2607.60	2615.65	2655.39	1570179	...

1800 rows × 34 columns

In [ ]:

In [26]:
```
!pip install pmdarima
```

```
Collecting pmdarima
  Downloading pmdarima-1.8.5-cp38-cp38-win_amd64.whl (602 kB)
Requirement already satisfied: numpy>=1.19.3 in c:\anaconda\lib\site-packages (from pmda
rima) (1.20.1)
Requirement already satisfied: joblib>=0.11 in c:\anaconda\lib\site-packages (from pmdar
ima) (1.0.1)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in c:\anaconda\lib\site-packa
ges (from pmdarima) (52.0.0.post20210125)
Requirement already satisfied: scipy>=1.3.2 in c:\anaconda\lib\site-packages (from pmdar
ima) (1.6.2)
Requirement already satisfied: scikit-learn>=0.22 in c:\anaconda\lib\site-packages (from
pmdarima) (0.24.1)
Requirement already satisfied: Cython!=0.29.18,>=0.29 in c:\anaconda\lib\site-packages
(from pmdarima) (0.29.23)
Requirement already satisfied: urllib3 in c:\anaconda\lib\site-packages (from pmdarima)
(1.26.4)
```

```
Requirement already satisfied: pandas>=0.19 in c:\anaconda\lib\site-packages (from pmdar
ima) (1.2.4)
Requirement already satisfied: statsmodels!=0.12.0,>=0.11 in c:\anaconda\lib\site-packag
es (from pmdarima) (0.12.2)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\anaconda\lib\site-packages
(from pandas>=0.19->pmdarima) (2.8.1)
Requirement already satisfied: pytz>=2017.3 in c:\anaconda\lib\site-packages (from panda
s>=0.19->pmdarima) (2021.1)
Requirement already satisfied: six>=1.5 in c:\anaconda\lib\site-packages (from python-da
teutil>=2.7.3->pandas>=0.19->pmdarima) (1.15.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\anaconda\lib\site-packages (fr
om scikit-learn>=0.22->pmdarima) (2.1.0)
Requirement already satisfied: patsy>=0.5 in c:\anaconda\lib\site-packages (from statsmo
dels!=0.12.0,>=0.11->pmdarima) (0.5.1)
Installing collected packages: pmdarima
Successfully installed pmdarima-1.8.5
```

In [27]:
```
from pmdarima import auto_arima
```

In [28]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [29]:
```
model=auto_arima(y=training_data['VWAP'],exogenous=training_data[ind_features],trace=Tr
```

```
Performing stepwise search to minimize aic
 ARIMA(2,0,2)(0,0,0)[0] intercept   : AIC=20931.563, Time=3.41 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=20925.246, Time=2.45 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=20926.371, Time=3.41 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=20926.343, Time=2.96 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=32616.914, Time=2.44 sec
 ARIMA(1,0,1)(0,0,0)[0] intercept   : AIC=20929.256, Time=4.29 sec

Best model:  ARIMA(0,0,0)(0,0,0)[0] intercept
Total fit time: 19.104 seconds
```

In [30]:
```
model.fit(training_data['VWAP'],training_data[ind_features])
```

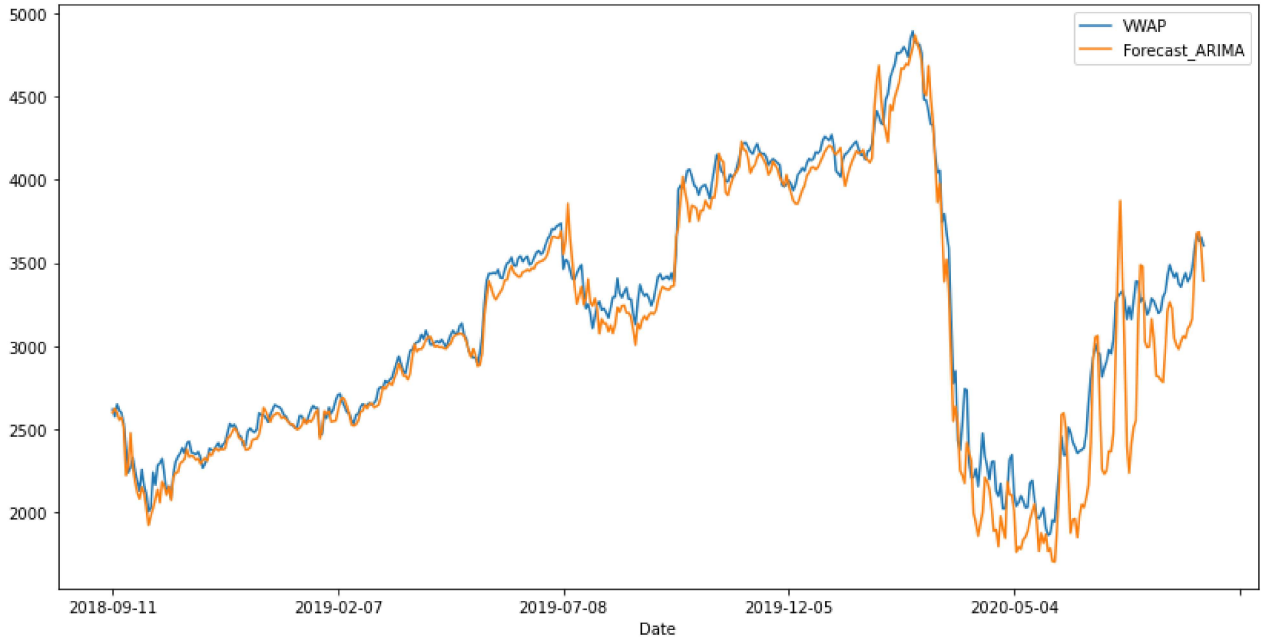
Out[30]:
```
ARIMA(order=(0, 0, 0), scoring_args={}, suppress_warnings=True)
```

In [31]:
```
forecast=model.predict(n_periods=len(test_data), exogenous=test_data[ind_features])
```

In [32]:
```
test_data['Forecast_ARIMA']=forecast
```

In [34]:
```
test_data[['VWAP','Forecast_ARIMA']].plot(figsize=(14,7))
```

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



The Auto ARIMA model seems to do a fairly good job in predicting the stock price

In [ ]:

Checking Accuracy of our model

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

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

Out[36]: 187.91287668231973

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

Out[37]: 124.74418035631113

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: