time-series-revision-notebook

January 18, 2023

[109]:

```
import os
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima_model import ARIMA
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import mean_squared_error, mean_absolute_error
import math
import yfinance as yf
from pmdarima.arima import auto_arima
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
```

[11]:

```
stock_data=pd.read_csv(r"C:\Users\hrush\Downloads\8th Jan␣
  ↪FSDSbootcamp(time-series-main)\time-series-main\TSLA.CSV")
```

[12]:

```
stock_data
```

[12]:

```
                          Date        Open        High         Low  \
0    2020-01-13 00:00:00-05:00   32.900002   35.042000   32.799999
1    2020-01-14 00:00:00-05:00   36.284000   36.493999   34.993332
2    2020-01-15 00:00:00-05:00   35.317333   35.855999   34.452667
3    2020-01-16 00:00:00-05:00   32.916668   34.297333   32.811333
4    2020-01-17 00:00:00-05:00   33.840668   34.377998   33.543999
..                         …           …           …           …
753  2023-01-09 00:00:00-05:00  118.959999  123.519997  117.110001
754  2023-01-10 00:00:00-05:00  121.070000  122.760002  114.919998
755  2023-01-11 00:00:00-05:00  122.089996  125.949997  120.510002
756  2023-01-12 00:00:00-05:00  122.559998  124.129997  117.000000
757  2023-01-13 00:00:00-05:00  116.550003  121.650002  115.599998

          Close     Volume  Dividends  Stock Splits
```

```
0     34.990665  397764000        0.0           0.0
1     35.861332  434943000        0.0           0.0
2     34.566666  260532000        0.0           0.0
3     34.232666  326050500        0.0           0.0
4     34.033333  204436500        0.0           0.0
..          …          …        …           …
753  119.769997  190284000        0.0           0.0
754  118.849998  167642500        0.0           0.0
755  123.220001  183810800        0.0           0.0
756  123.559998  169089400        0.0           0.0
757  119.029999   92498442        0.0           0.0

[758 rows x 8 columns]
```

[14]:
```
stock_data=stock_data[["Date", "Close"]]
```

[15]:
```
stock_data
```

[15]:
```
                          Date       Close
0    2020-01-13 00:00:00-05:00   34.990665
1    2020-01-14 00:00:00-05:00   35.861332
2    2020-01-15 00:00:00-05:00   34.566666
3    2020-01-16 00:00:00-05:00   34.232666
4    2020-01-17 00:00:00-05:00   34.033333
..                         …         …
753  2023-01-09 00:00:00-05:00  119.769997
754  2023-01-10 00:00:00-05:00  118.849998
755  2023-01-11 00:00:00-05:00  123.220001
756  2023-01-12 00:00:00-05:00  123.559998
757  2023-01-13 00:00:00-05:00  119.029999

[758 rows x 2 columns]
```

[16]:
```
stock_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 758 entries, 0 to 757
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    758 non-null    object
 1   Close   758 non-null    float64
dtypes: float64(1), object(1)
memory usage: 12.0+ KB
```

[19]:
```
stock_data.Date=pd.to_datetime(stock_data.Date)
```

[20]:

```
stock_data
```

[20]:

```
                          Date       Close
0    2020-01-13 00:00:00-05:00   34.990665
1    2020-01-14 00:00:00-05:00   35.861332
2    2020-01-15 00:00:00-05:00   34.566666
3    2020-01-16 00:00:00-05:00   34.232666
4    2020-01-17 00:00:00-05:00   34.033333
..                         …          …
753  2023-01-09 00:00:00-05:00  119.769997
754  2023-01-10 00:00:00-05:00  118.849998
755  2023-01-11 00:00:00-05:00  123.220001
756  2023-01-12 00:00:00-05:00  123.559998
757  2023-01-13 00:00:00-05:00  119.029999

[758 rows x 2 columns]
```

[21]:

```
stock_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 758 entries, 0 to 757
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    758 non-null    object
 1   Close   758 non-null    float64
dtypes: float64(1), object(1)
memory usage: 12.0+ KB
```

[23]:

```
stock_data=stock_data.set_index("Date")
```

[24]:

```
stock_data
```

[24]:

```
                                Close
Date
2020-01-13 00:00:00-05:00   34.990665
2020-01-14 00:00:00-05:00   35.861332
2020-01-15 00:00:00-05:00   34.566666
2020-01-16 00:00:00-05:00   34.232666
2020-01-17 00:00:00-05:00   34.033333
…                                …
2023-01-09 00:00:00-05:00  119.769997
2023-01-10 00:00:00-05:00  118.849998
2023-01-11 00:00:00-05:00  123.220001
2023-01-12 00:00:00-05:00  123.559998
2023-01-13 00:00:00-05:00  119.029999
```

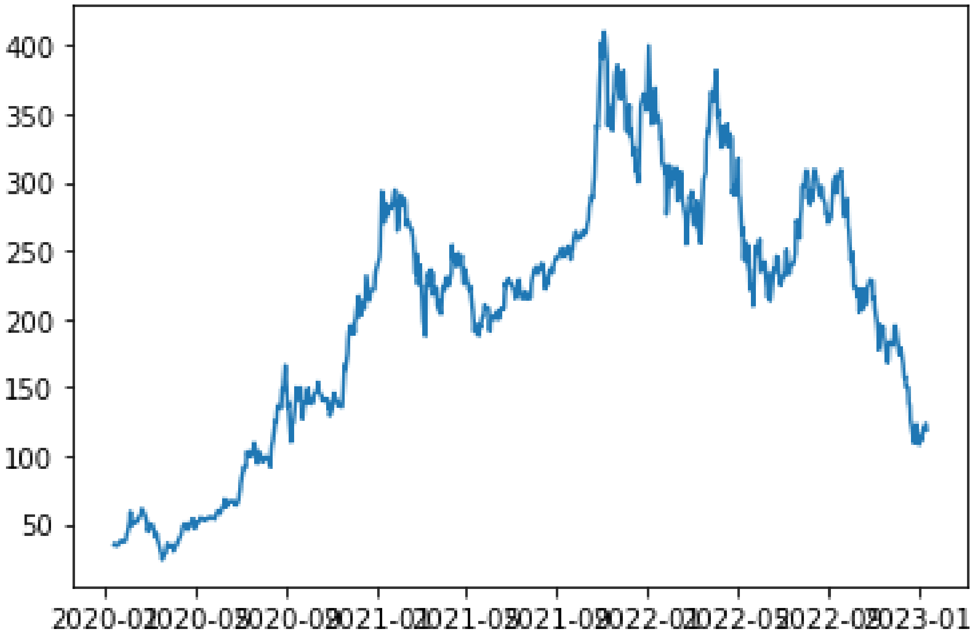
```
[758 rows x 1 columns]
```

[25]:

```python
plt.plot(stock_data['Close'])
```

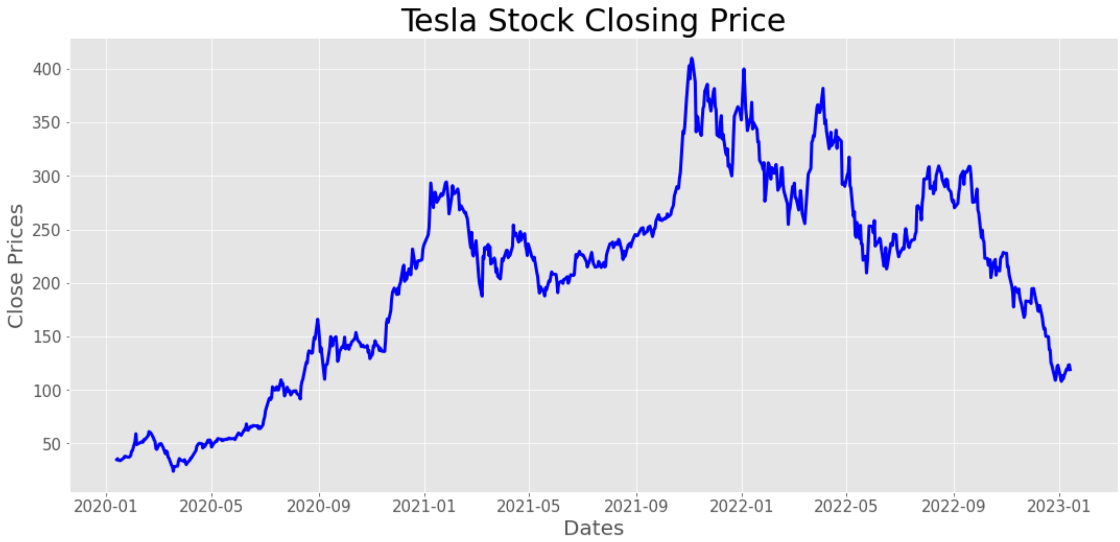
[25]:

```
[<matplotlib.lines.Line2D at 0x1be5805b430>]
```



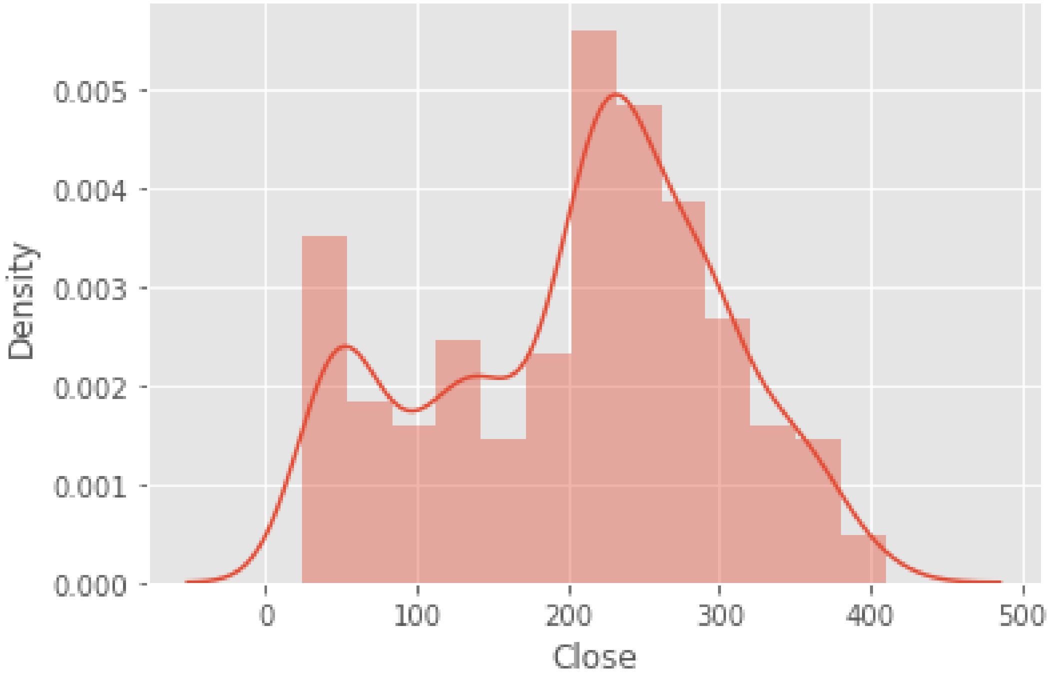
[26]:

```python
# plotting close price
plt.style.use('ggplot')
plt.figure(figsize=(18,8))
plt.grid(True)
plt.xlabel('Dates', fontsize = 20)
plt.xticks(fontsize = 15)
plt.ylabel('Close Prices', fontsize = 20)
plt.yticks(fontsize = 15)
plt.plot(stock_data['Close'], linewidth = 3, color = 'blue')
plt.title('Tesla Stock Closing Price', fontsize = 30)
plt.show()
```



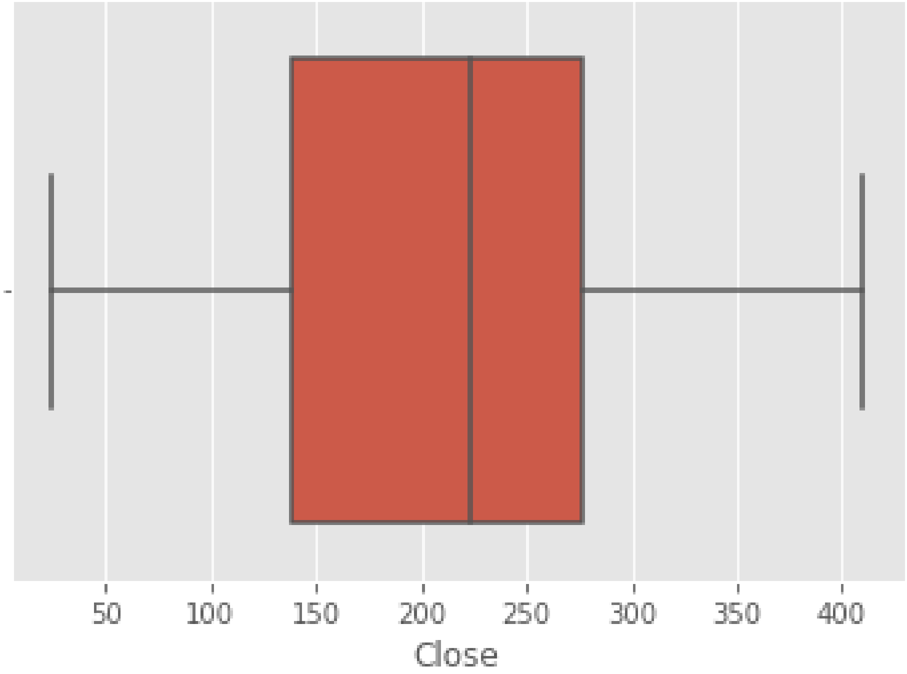
[27]:
```
import seaborn as sns
sns.distplot(stock_data['Close'])
```

[27]:
```
<AxesSubplot:xlabel='Close', ylabel='Density'>
```



[28]:
```
sns.boxplot(stock_data['Close'])
```

```
[28]: <AxesSubplot:xlabel='Close'>
```



```
[57]: stock_data
```

```
[57]:                                    Close
      Date
      2020-01-13 00:00:00-05:00   34.990665
      2020-01-14 00:00:00-05:00   35.861332
      2020-01-15 00:00:00-05:00   34.566666
      2020-01-16 00:00:00-05:00   34.232666
      2020-01-17 00:00:00-05:00   34.033333
      …                                 …
      2023-01-09 00:00:00-05:00  119.769997
      2023-01-10 00:00:00-05:00  118.849998
      2023-01-11 00:00:00-05:00  123.220001
      2023-01-12 00:00:00-05:00  123.559998
      2023-01-13 00:00:00-05:00  119.029999

      [758 rows x 1 columns]
```

```
[65]: result.trend.isnull().sum()
```

```
[65]: 20
```

[66]:
```
result.resid.isnull().sum()
```

[66]:
```
20
```

[60]:
```
result.observed
```

[60]:
```
Date
2020-01-13 00:00:00-05:00     34.990665
2020-01-14 00:00:00-05:00     35.861332
2020-01-15 00:00:00-05:00     34.566666
2020-01-16 00:00:00-05:00     34.232666
2020-01-17 00:00:00-05:00     34.033333
                                 …
2023-01-09 00:00:00-05:00    119.769997
2023-01-10 00:00:00-05:00    118.849998
2023-01-11 00:00:00-05:00    123.220001
2023-01-12 00:00:00-05:00    123.559998
2023-01-13 00:00:00-05:00    119.029999
Length: 758, dtype: float64
```

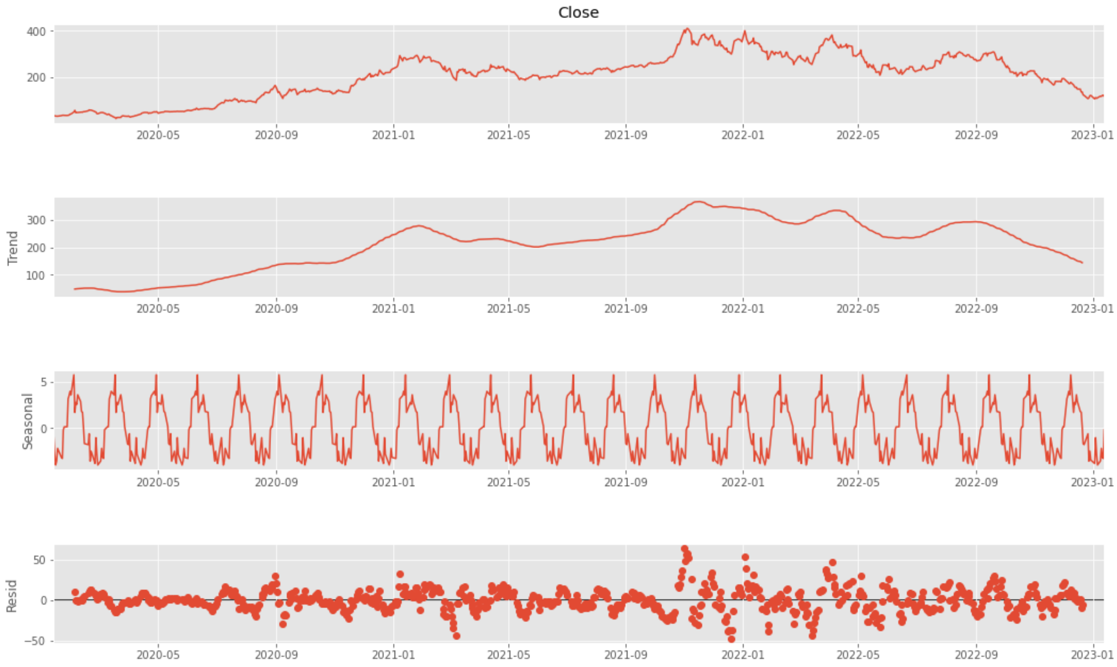
[67]:
```
result=seasonal_decompose(stock_data["Close"],period=30)
```

[64]:
```
result.seasonal
```

[64]:
```
Date
2020-01-13 00:00:00-05:00    1.918348
2020-01-14 00:00:00-05:00    0.516842
2020-01-15 00:00:00-05:00    0.353797
2020-01-16 00:00:00-05:00   -1.262323
2020-01-17 00:00:00-05:00   -1.596447
                               …
2023-01-09 00:00:00-05:00   -0.043307
2023-01-10 00:00:00-05:00    2.751346
2023-01-11 00:00:00-05:00    1.703326
2023-01-12 00:00:00-05:00    1.405505
2023-01-13 00:00:00-05:00    3.281316
Name: seasonal, Length: 758, dtype: float64
```

[54]:
```
fig=plt.figure(figsize=(20,10))
fig=result.plot()
fig.set_size_inches(17,10)
```

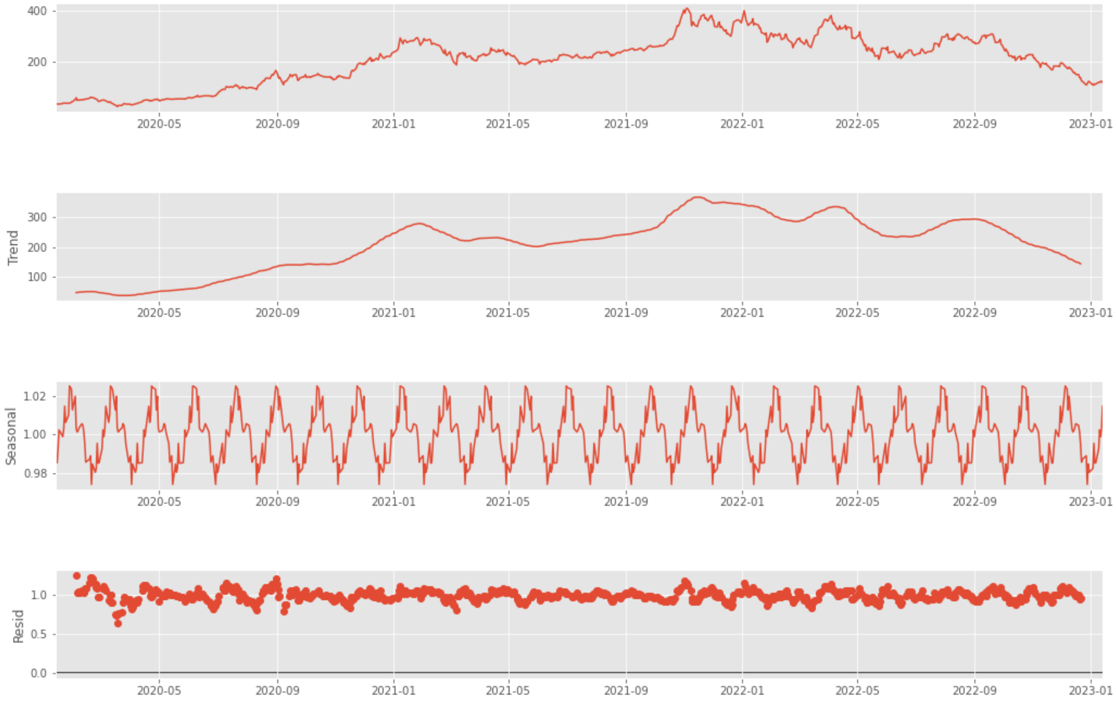
```
<Figure size 1440x720 with 0 Axes>
```



[55]:

```
result=seasonal_decompose(stock_data[["Close"]],model="multiplicative",period=30)
fig=plt.figure(figsize=(20,10))
fig=result.plot()
fig.set_size_inches(17,10)
```

```
<Figure size 1440x720 with 0 Axes>
```

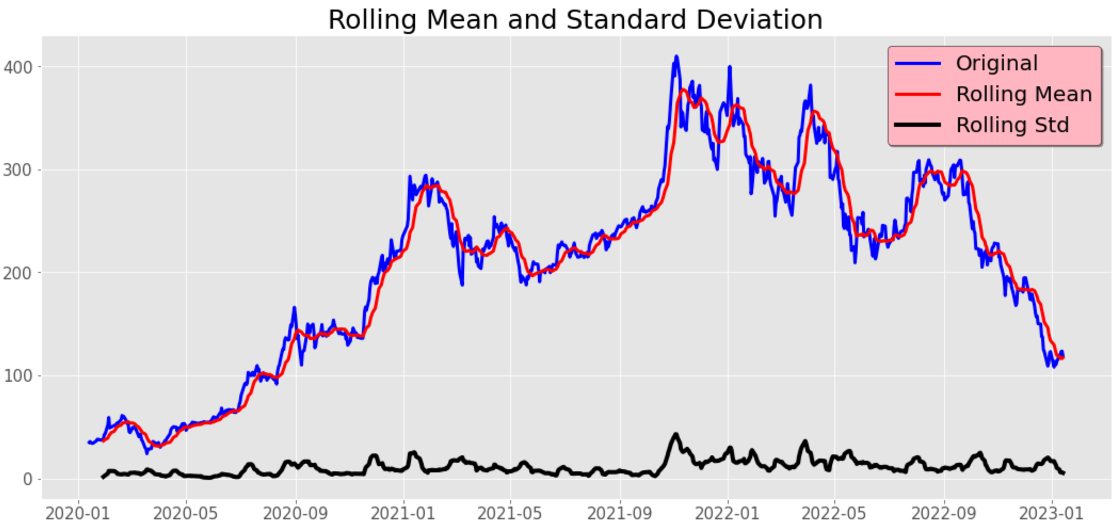


[68]:
```
#Test for staionarity
def test_stationarity(timeseries):
    # Determing rolling statistics
    rolmean = timeseries.rolling(12).mean() # rolling mean
    rolstd = timeseries.rolling(12).std() # rolling standard deviation
    # Plot rolling statistics:
    plt.figure(figsize = (18,8))
    plt.grid('both')
    plt.plot(timeseries, color='blue',label='Original', linewidth = 3)
    plt.plot(rolmean, color='red', label='Rolling Mean',linewidth = 3)
    plt.plot(rolstd, color='black', label = 'Rolling Std',linewidth = 4)
    plt.legend(loc='best', fontsize = 20,␣
 ↪shadow=True,facecolor='lightpink',edgecolor = 'k')
    plt.title('Rolling Mean and Standard Deviation', fontsize = 25)
    plt.xticks(fontsize = 15)
    plt.yticks(fontsize = 15)
    plt.show(block=False)

    print("Results of dickey fuller test")
    adft = adfuller(timeseries,autolag='AIC')
    # output for dft will give us without defining what the values are.
    # hence we manually write what values does it explains using a for loop
    output = pd.Series(adft[0:4],index=['Test Statistics','p-value','No. of␣
 ↪lags used','Number of observations used'])
```

```
    for key,values in adft[4].items():
        output['critical value (%s)'%key] =  values
    print(output)
```

[69]:
```
test_stationarity(stock_data['Close'])
```



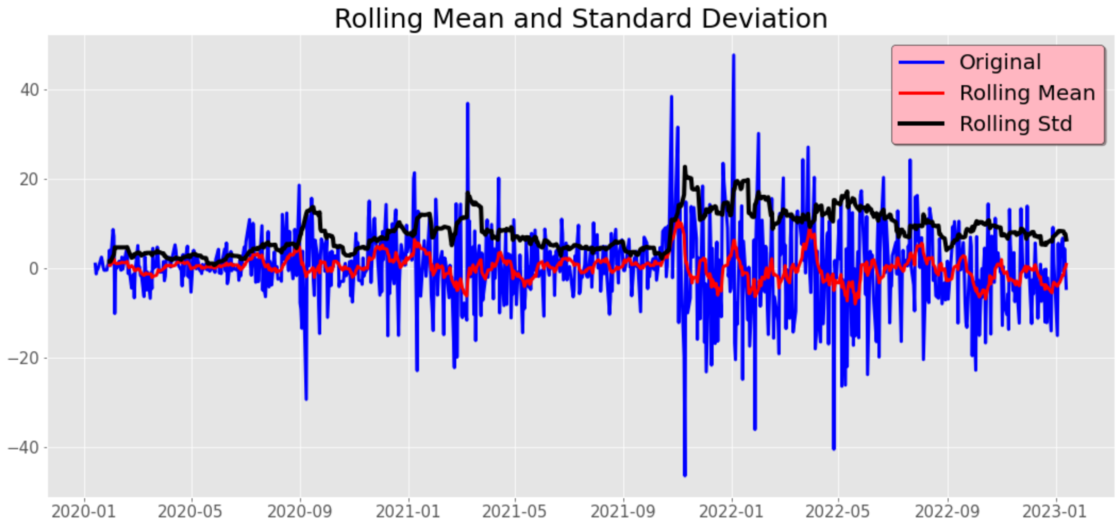
```
Results of dickey fuller test
Test Statistics                 -1.881719
p-value                          0.340709
No. of lags used                 9.000000
Number of observations used    748.000000
critical value (1%)             -3.439123
critical value (5%)             -2.865412
critical value (10%)            -2.568832
dtype: float64
```

[81]:
```
df_close=stock_data["Close"]
```

[82]:
```
tesla_close_diff_1=df_close.diff()
```

[83]:
```
tesla_close_diff_1.dropna(inplace=True)
```

[84]:
```
test_stationarity(tesla_close_diff_1)
```

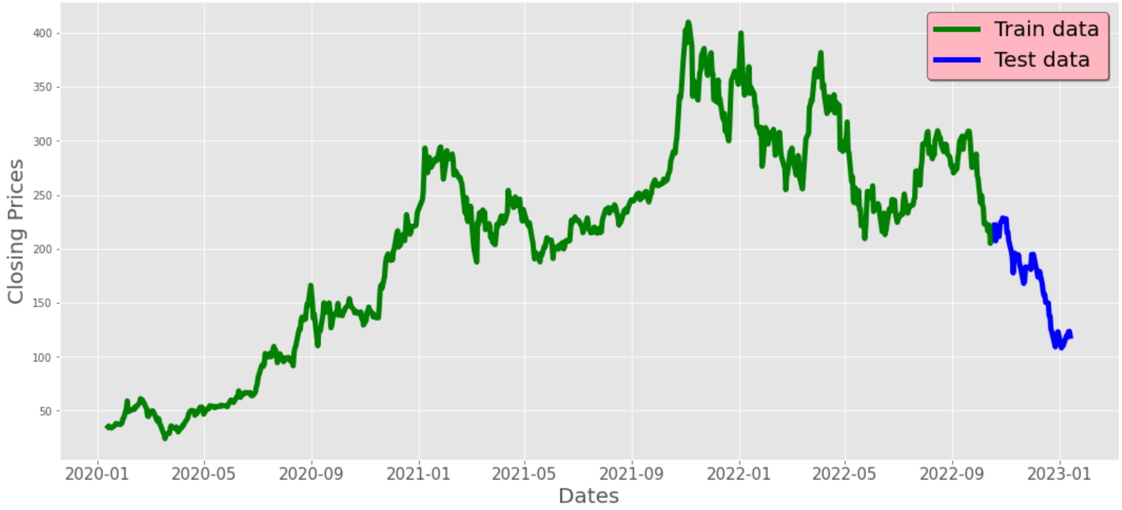


```
Results of dickey fuller test
Test Statistics                -8.163375e+00
p-value                         9.017932e-13
No. of lags used                8.000000e+00
Number of observations used     7.480000e+02
critical value (1%)            -3.439123e+00
critical value (5%)            -2.865412e+00
critical value (10%)           -2.568832e+00
dtype: float64
```

1 ARIMA

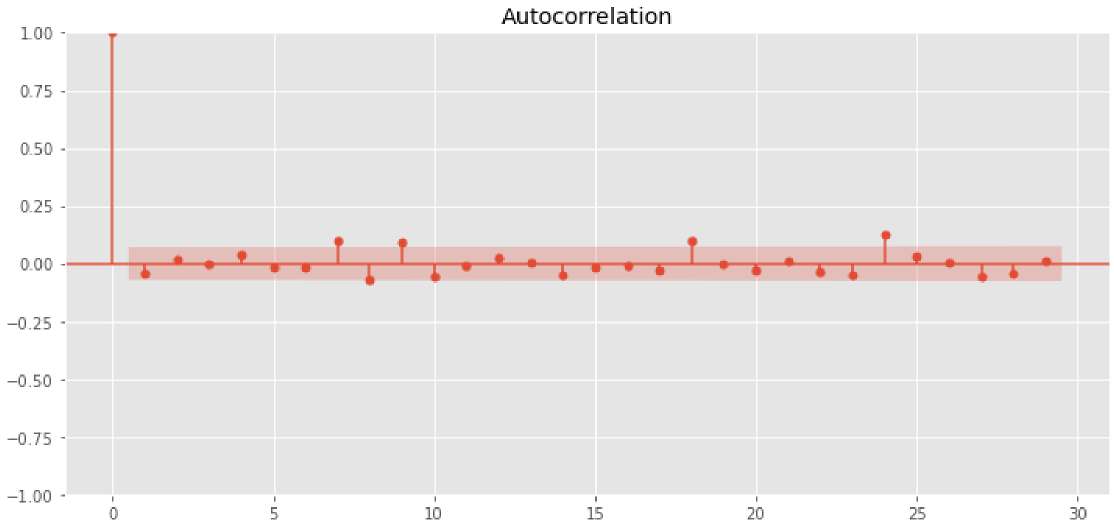
[87]:
```
#split data into train and training set
train_data=df_close[0:-60]
test_data=df_close[-60:]
plt.figure(figsize=(18,8))
plt.grid(True)
plt.xlabel('Dates', fontsize = 20)
plt.ylabel('Closing Prices', fontsize = 20)
plt.xticks(fontsize = 15)
plt.xticks(fontsize = 15)
plt.plot(train_data, 'green', label='Train data', linewidth = 5)
plt.plot(test_data, 'blue', label='Test data', linewidth = 5)
plt.legend(fontsize = 20, shadow=True,facecolor='lightpink',edgecolor = 'k')
```

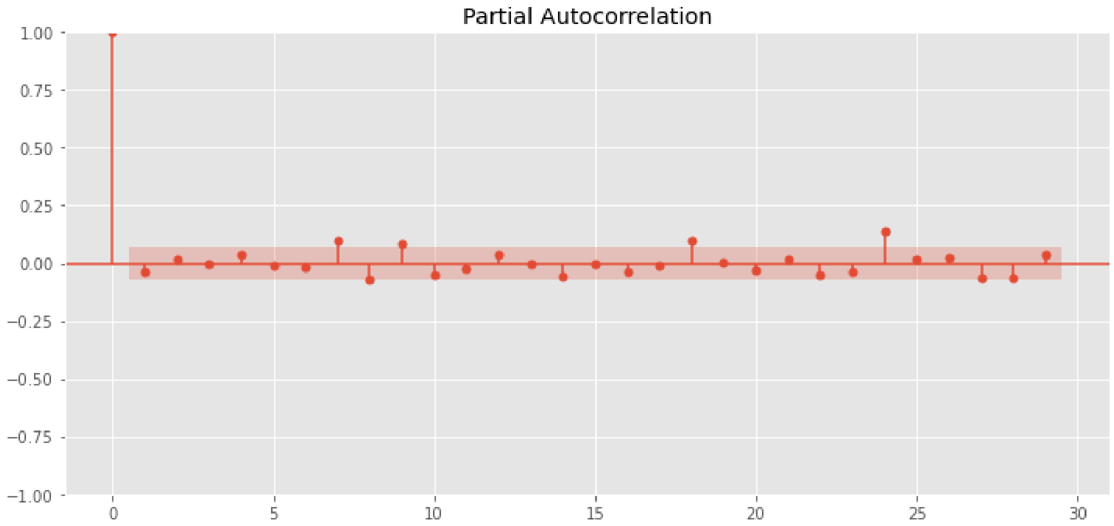
[87]:
```
<matplotlib.legend.Legend at 0x1be5ada6d90>
```



[88]:

```
fig = plt.figure(figsize=(12,12))
ax1 = fig.add_subplot(211)#function
fig = plot_acf(tesla_close_diff_1, ax=ax1)
ax2 = fig.add_subplot(212)
fig = plot_pacf(tesla_close_diff_1, ax=ax2)
```


Autocorrelation


Partial Autocorrelation

[89]:
```
df_close
```

[89]:
```
Date
2020-01-13 00:00:00-05:00     34.990665
2020-01-14 00:00:00-05:00     35.861332
2020-01-15 00:00:00-05:00     34.566666
2020-01-16 00:00:00-05:00     34.232666
2020-01-17 00:00:00-05:00     34.033333
                                 …
2023-01-09 00:00:00-05:00    119.769997
2023-01-10 00:00:00-05:00    118.849998
2023-01-11 00:00:00-05:00    123.220001
2023-01-12 00:00:00-05:00    123.559998
2023-01-13 00:00:00-05:00    119.029999
```

```
Name: Close, Length: 758, dtype: float64
```

```
[91]: df_close_diff=df_close.diff(2)
```

```
[93]: df=pd.concat([df_close,df_close_diff],axis=1)
```

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

```
[97]: df
```

```
[97]:                                 Close     Close
      Date
      2020-01-14 00:00:00-05:00   35.861332  0.870667
      2020-01-15 00:00:00-05:00   34.566666 -1.294666
      2020-01-16 00:00:00-05:00   34.232666 -0.334000
      2020-01-17 00:00:00-05:00   34.033333 -0.199333
      2020-01-21 00:00:00-05:00   36.480000  2.446667
      …                                 …         …
      2023-01-09 00:00:00-05:00  119.769997  6.709999
      2023-01-10 00:00:00-05:00  118.849998 -0.919998
      2023-01-11 00:00:00-05:00  123.220001  4.370003
      2023-01-12 00:00:00-05:00  123.559998  0.339996
      2023-01-13 00:00:00-05:00  119.029999 -4.529999

      [757 rows x 2 columns]
```

```
[96]: df.corr()
```

```
[96]:           Close     Close
      Close  1.000000  0.030537
      Close  0.030537  1.000000
```

```
[98]: df_close
```

```
[98]: Date
      2020-01-13 00:00:00-05:00     34.990665
      2020-01-14 00:00:00-05:00     35.861332
      2020-01-15 00:00:00-05:00     34.566666
      2020-01-16 00:00:00-05:00     34.232666
      2020-01-17 00:00:00-05:00     34.033333
                                      …
      2023-01-09 00:00:00-05:00    119.769997
      2023-01-10 00:00:00-05:00    118.849998
      2023-01-11 00:00:00-05:00    123.220001
      2023-01-12 00:00:00-05:00    123.559998
      2023-01-13 00:00:00-05:00    119.029999
      Name: Close, Length: 758, dtype: float64
```

[99]:

```
train_data
```

[99]:

```
Date
2020-01-13 00:00:00-05:00     34.990665
2020-01-14 00:00:00-05:00     35.861332
2020-01-15 00:00:00-05:00     34.566666
2020-01-16 00:00:00-05:00     34.232666
2020-01-17 00:00:00-05:00     34.033333
                                …
2022-10-12 00:00:00-04:00    217.240005
2022-10-13 00:00:00-04:00    221.720001
2022-10-14 00:00:00-04:00    204.990005
2022-10-17 00:00:00-04:00    219.350006
2022-10-18 00:00:00-04:00    220.190002
Name: Close, Length: 698, dtype: float64
```

[100]:

```
test_data
```

[100]:

```
Date
2022-10-19 00:00:00-04:00    222.039993
2022-10-20 00:00:00-04:00    207.279999
2022-10-21 00:00:00-04:00    214.440002
2022-10-24 00:00:00-04:00    211.250000
2022-10-25 00:00:00-04:00    222.419998
2022-10-26 00:00:00-04:00    224.639999
2022-10-27 00:00:00-04:00    225.089996
2022-10-28 00:00:00-04:00    228.520004
2022-10-31 00:00:00-04:00    227.539993
2022-11-01 00:00:00-04:00    227.820007
2022-11-02 00:00:00-04:00    214.979996
2022-11-03 00:00:00-04:00    215.309998
2022-11-04 00:00:00-04:00    207.470001
2022-11-07 00:00:00-05:00    197.080002
2022-11-08 00:00:00-05:00    191.300003
2022-11-09 00:00:00-05:00    177.589996
2022-11-10 00:00:00-05:00    190.720001
2022-11-11 00:00:00-05:00    195.970001
2022-11-14 00:00:00-05:00    190.949997
2022-11-15 00:00:00-05:00    194.419998
2022-11-16 00:00:00-05:00    186.919998
2022-11-17 00:00:00-05:00    183.169998
2022-11-18 00:00:00-05:00    180.190002
2022-11-21 00:00:00-05:00    167.869995
2022-11-22 00:00:00-05:00    169.910004
2022-11-23 00:00:00-05:00    183.199997
2022-11-25 00:00:00-05:00    182.860001
2022-11-28 00:00:00-05:00    182.919998
```

```
2022-11-29 00:00:00-05:00    180.830002
2022-11-30 00:00:00-05:00    194.699997
2022-12-01 00:00:00-05:00    194.699997
2022-12-02 00:00:00-05:00    194.860001
2022-12-05 00:00:00-05:00    182.449997
2022-12-06 00:00:00-05:00    179.820007
2022-12-07 00:00:00-05:00    174.039993
2022-12-08 00:00:00-05:00    173.440002
2022-12-09 00:00:00-05:00    179.050003
2022-12-12 00:00:00-05:00    167.820007
2022-12-13 00:00:00-05:00    160.949997
2022-12-14 00:00:00-05:00    156.800003
2022-12-15 00:00:00-05:00    157.669998
2022-12-16 00:00:00-05:00    150.229996
2022-12-19 00:00:00-05:00    149.869995
2022-12-20 00:00:00-05:00    137.800003
2022-12-21 00:00:00-05:00    137.570007
2022-12-22 00:00:00-05:00    125.349998
2022-12-23 00:00:00-05:00    123.150002
2022-12-27 00:00:00-05:00    109.099998
2022-12-28 00:00:00-05:00    112.709999
2022-12-29 00:00:00-05:00    121.820000
2022-12-30 00:00:00-05:00    123.180000
2023-01-03 00:00:00-05:00    108.099998
2023-01-04 00:00:00-05:00    113.639999
2023-01-05 00:00:00-05:00    110.339996
2023-01-06 00:00:00-05:00    113.059998
2023-01-09 00:00:00-05:00    119.769997
2023-01-10 00:00:00-05:00    118.849998
2023-01-11 00:00:00-05:00    123.220001
2023-01-12 00:00:00-05:00    123.559998
2023-01-13 00:00:00-05:00    119.029999
Name: Close, dtype: float64
```

[141]:

```python
model = sm.tsa.arima.ARIMA(train_data, order=(2,2,0))
```

[142]:

```python
model_fit=model.fit()
```

[143]:

```python
model_fit.summary()
```

[143]:

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                               SARIMAX Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                  698
Model:                 ARIMA(2, 2, 0)   Log Likelihood               -2626.473
Date:                Thu, 19 Jan 2023   AIC                           5258.947
```

```
Time:                        00:24:01   BIC                           5272.583
Sample:                             0   HQIC                          5264.219
                                - 698
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.7027      0.028    -25.451      0.000      -0.757      -0.649
ar.L2         -0.3397      0.031    -11.068      0.000      -0.400      -0.280
sigma2       110.9011      3.866     28.687      0.000     103.324     118.478
================================================================================
===
Ljung-Box (L1) (Q):                   6.24   Jarque-Bera (JB):
230.13
Prob(Q):                              0.01   Prob(JB):
0.00
Heteroskedasticity (H):               5.59   Skew:
0.01
Prob(H) (two-sided):                  0.00   Kurtosis:
5.82
================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
"""
```

```
[173]: y=np.array(list(test_data))
```

```
[179]: predictions=model_fit.forecast(step=60)
```

```
[181]: predictions
```

```
[181]: 698    219.969144
       dtype: float64
```

```
[148]: orignal_value=test_data[0]
```

```
[149]: orignal_value
```

```
[149]: 222.0399932861328
```

```
[150]: org=np.array(orignal_value)
```

```
[151]: org
```

[151]:
```
array(222.03999329)
```

[133]:
```
model_fit.predict()
```

[133]:
```
Date
2020-01-13 00:00:00-05:00      0.000000
2020-01-14 00:00:00-05:00     34.990548
2020-01-15 00:00:00-05:00     35.826054
2020-01-16 00:00:00-05:00     34.627762
2020-01-17 00:00:00-05:00     34.229571
                                …
2022-10-12 00:00:00-04:00    216.812013
2022-10-13 00:00:00-04:00    217.125816
2022-10-14 00:00:00-04:00    221.576266
2022-10-17 00:00:00-04:00    205.695151
2022-10-18 00:00:00-04:00    218.589481
Name: predicted_mean, Length: 698, dtype: float64
```

[136]:
```
6# evaluate an ARIMA model for a given order (p,d,q)

def evaluate_arima_model(X, y, arima_order):
    # prepare training dataset
    # make predictions list
    history = [x for x in X]
    predictions = list()
    for t in range(len(y)):
        model = ARIMA(history, order=arima_order)
        model_fit = model.fit()
        yhat = model_fit.forecast()[0]
        predictions.append(yhat)
        history.append(y[t])
    # calculate out of sample error
    rmse = np.sqrt(mean_squared_error(y, predictions))
    return rmse
```

[137]:
```
# evaluate different combinations of p, d and q values for an ARIMA model to␣
 ↪get the best order for ARIMA Model
def evaluate_models(dataset, test, p_values, d_values, q_values):
    dataset = dataset.astype('float32')
    best_score, best_cfg = float("inf"), None
    for p in p_values:
        for d in d_values:
            for q in q_values:
                order = (p,d,q)
                try:
                    rmse = evaluate_arima_model(dataset, test, order)
                    if rmse < best_score:
                        best_score, best_cfg = rmse, order
                    print('ARIMA%s RMSE=%.3f' % (order,rmse))
                except:
                    continue
    print('Best ARIMA%s RMSE=%.3f' % (best_cfg, best_score))
```

[138]:
```
# evaluate parameters
p_values = range(0, 3)
d_values = range(0, 3)
q_values = range(0, 3)
```

[140]:
```
warnings.filterwarnings("ignore")
evaluate_models(train_data, test_data, p_values, d_values, q_values)
```

```
Best ARIMANone RMSE=inf
```

[135]:
```
for p in p_values:
        for d in d_values:
            for q in q_values:
                order = (p,d,q)
                print(order)
```

```
(0, 0, 0)
(0, 0, 1)
(0, 0, 2)
(0, 1, 0)
(0, 1, 1)
(0, 1, 2)
(0, 2, 0)
(0, 2, 1)
(0, 2, 2)
(1, 0, 0)
(1, 0, 1)
(1, 0, 2)
(1, 1, 0)
(1, 1, 1)
(1, 1, 2)
(1, 2, 0)
(1, 2, 1)
(1, 2, 2)
(2, 0, 0)
(2, 0, 1)
(2, 0, 2)
(2, 1, 0)
(2, 1, 1)
(2, 1, 2)
(2, 2, 0)
```

```
(2, 2, 1)
(2, 2, 2)
```

[192]:

```
history=[x for x in train_data]
predictions=list()
for t in range(len(test_data)):
    model=sm.tsa.arima.ARIMA(history,order=(2,2,0))
    model_fit=model.fit()
    fc=model_fit.forecast(alpha=0.05)[0]
    predictions.append(fc)
    history.append(test_data[t])
print(np.sqrt(mean_squared_error(test_data,predictions)))
```

```
8.551872703262262
```

[200]:

```
len(df_close)+60
```

[200]:

```
818
```

[211]:

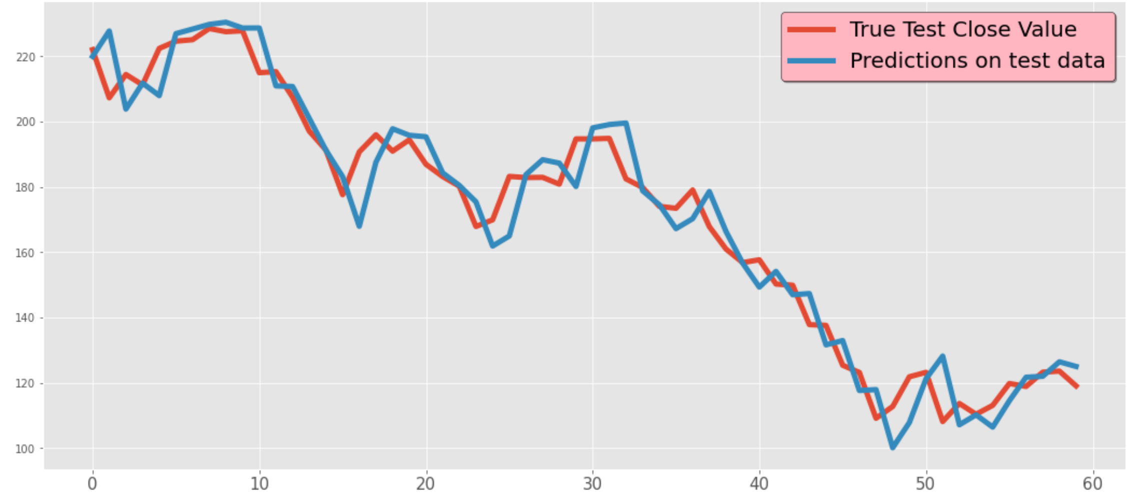
```
model_fit.predict(start=1,end=818)
```

[211]:

```
array([ 52.48591678,  36.73495358,  34.41598384,  33.94168106,
        33.42082873,  37.02285076,  39.25565691,  39.56339606,
        38.06795844,  36.94233324,  37.63802512,  39.08132689,
        44.4514095 ,  45.35599933,  56.12971376,  64.6770751 ,
        51.46894694,  48.81129803,  46.83832594,  52.17201556,
        52.24123238,  51.60187727,  54.22091148,  54.00795278,
        59.09714409,  63.68745071,  62.35387384,  60.95740091,
        53.89478422,  51.02832407,  49.1776924 ,  42.01703873,
        41.37909508,  48.59546636,  51.36552245,  51.76440953,
        47.94831916,  45.95195567,  37.56777077,  40.93471508,
        40.8746404 ,  36.45997719,  34.09725955,  25.68478921,
        25.56534196,  20.10112586,  27.79108387,  28.61789214,
        30.55225373,  35.23324967,  38.52750496,  37.40114336,
        34.49990493,  32.6463667 ,  34.75931388,  31.53325788,
        29.19462329,  30.89870693,  35.1671438 ,  38.40318438,
        38.1799794 ,  39.40853963,  45.61455448,  50.95670971,
        52.23174656,  51.78174232,  51.25264754,  50.16275221,
        44.60461425,  48.06546732,  46.32139296,  49.07870446,
        54.60656344,  52.93933514,  54.88371773,  51.87745163,
        45.3906783 ,  49.52833774,  51.05323248,  53.95402442,
        52.46295235,  55.66352756,  54.84277247,  54.59283531,
        52.13867704,  53.30262638,  53.09365298,  54.69921681,
        54.02163727,  54.70098206,  55.47453911,  54.71087002,
        54.63177201,  54.52120672,  53.51243026,  55.9187227 ,
        61.52753505,  60.64888151,  59.87752101,  56.92598403,
        59.03722177,  64.71757534,  64.58790394,  71.20120877,
        65.70369169,  62.18124891,  65.07939749,  65.84697225,
        67.32344866,  67.21700672,  67.18058768,  66.35089811,
        66.70281681,  63.28431103,  65.38266755,  63.17927514,
        68.17996027,  74.03125754,  78.27099138,  84.88489981,
        97.7509954 ,  99.01780957,  94.63698865,  93.32595841,
       106.13008955, 103.20789523, 103.65826704, 103.08265723,
       100.30213018,  99.58341882, 111.35359053, 106.5984999 ,
       107.92686074,  98.24294402,  91.13143619, 100.9419852 ,
        98.09041806, 101.53823631,  98.10004261,  94.47196305,
        98.41514751,  99.26056948, 100.21059308,  99.39097258,
        96.18542923,  93.08331516,  89.09060943, 105.37889335,
       112.86480482, 116.26211847, 128.2413816 , 132.05715555,
       130.42639586, 136.82533797, 140.48058692, 137.48739699,
       135.24988538, 145.5536183 , 154.3600992 , 152.04438653,
       172.88667764, 162.36454236, 149.65153436, 125.66957384,
       132.50964309,  98.2741826 , 116.02251876, 119.00379493,
       128.98945501, 145.25217418, 158.86066171, 155.37193617,
       141.66836341, 146.08762481, 150.80355891, 141.88064808,
       120.12661078, 121.79305184, 133.78779934, 145.01301351,
       143.35394502, 145.25712994, 152.27882478, 138.55246242,
       140.9696293 , 134.48809608, 142.61221074, 142.14162299,
       146.79547902, 149.32391634, 151.23414623, 156.67422417,
       150.68826523, 145.7269301 , 140.22187388, 137.65414948,
       138.87278825, 141.34618393, 140.16549432, 139.76863244,
       141.38251348, 133.99585968, 135.59858705, 125.62004055,
       132.45239143, 142.64177525, 144.30840076, 149.95017019,
       144.30320622, 140.44305448, 133.73173062, 137.40578035,
       136.35355643, 135.92766766, 134.99261234, 150.11490785,
       170.77725745, 176.94954391, 168.78508774, 177.34698669,
       191.36994943, 200.8894818 , 202.42802363, 190.91247703,
       195.67477766, 188.1417533 , 200.15306116, 201.51672947,
       221.56919621, 223.34072799, 202.71930406, 206.54498895,
       199.41279429, 216.61895704, 212.23382198, 209.0356169 ,
       219.89382266, 238.48151951, 220.64333247, 211.24189328,
       209.71891788, 221.80044654, 223.99660081, 224.20740826,
       234.94377608, 240.11772723, 250.17458472, 249.75033697,
       257.38627453, 281.13809065, 309.41685107, 278.12551962,
       285.4970425 , 282.41888294, 285.5764001 , 272.90469109,
       279.96931547, 284.26022884, 283.88449244, 282.35371044,
       296.58662141, 298.99760159, 290.23591609, 273.54070891,
       254.71109397, 276.16637327, 295.29170838, 292.29685223,
       284.27081492, 281.71679267, 288.67340572, 283.40112477,
       263.35877455, 264.16729186, 268.39397107, 264.7417689 ,
       264.27303713, 259.43156524, 258.71194104, 229.60286769,
       222.46059958, 242.29984965, 225.0886597 , 221.87726469,
       236.29115056, 230.20918265, 215.22707666, 196.35153269,
       189.38615857, 177.78663082, 228.56873753, 231.9529074 ,
       247.86613418, 233.95219334, 240.14868255, 223.65656777,
       234.13696099, 212.54790168, 215.19802854, 219.66722246,
       222.00315275, 207.63165007, 209.65576284, 201.81447874,
       201.46504949, 210.97099112, 228.03496995, 226.65604176,
       236.06122801, 233.53721852, 224.91882355, 226.71410541,
       224.3056325 , 237.04701076, 262.42539819, 251.31570769,
       249.88746095, 244.17840405, 236.56380246, 237.08776164,
       248.29034107, 241.06506571, 243.85197247, 245.43960645,
       233.75830479, 227.27608643, 218.96472285, 236.40063782,
       227.95644608, 223.96378245, 219.27299673, 218.86849214,
       223.78935205, 205.65914844, 200.17668502, 187.66928668,
       184.0729347 , 193.09960885, 191.21774319, 193.12480687,
       185.09331797, 196.23762037, 194.33328697, 206.52812316,
       203.8989861 , 210.41673218, 213.03756482, 210.89418497,
       208.43492314, 199.05958495, 185.27838195, 196.21321259,
       201.98637509, 204.69300723, 199.59811896, 203.73227599,
       204.14473042, 207.8947999 , 198.90901719, 200.76460627,
       205.35149751, 210.49490578, 208.82391738, 208.62374547,
       222.19593897, 233.29372211, 229.67622673, 232.83014756,
       227.33625193, 227.33955366, 224.78596052, 226.05983276,
       217.86940478, 211.12226043, 214.43018515, 218.75270477,
       232.83041917, 225.15909148, 217.34605465, 212.81941816,
       212.08812547, 214.50855627, 221.12432076, 219.90076109,
       216.7562985 , 212.55551099, 219.22444389, 214.76519843,
       215.86399071, 227.6465883 , 234.04233797, 243.35903175,
       240.45474362, 239.58257879, 238.73143209, 232.08323844,
       237.86247321, 236.38622664, 236.87901573, 241.49526091,
       240.09796927, 226.61718066, 215.47972082, 226.01397477,
       223.56941148, 228.08002601, 237.13884242, 240.35152243,
       240.48437381, 233.30203953, 237.43103591, 245.74008178,
       249.25477254, 247.22897963, 244.29155149, 244.2498732 ,
       252.84939456, 253.90777822, 253.9743948 , 243.80809281,
       246.14400081, 247.08861644, 253.99553576, 254.01219817,
       254.80802684, 240.9211265 , 244.02298026, 249.78997521,
       253.95324471, 261.78533082, 268.22305429, 262.2111368 ,
       260.9773234 , 256.8539299 , 258.07434627, 260.45892441,
       260.8551506 , 261.71194606, 265.77640796, 262.6070132 ,
       264.81544922, 269.83946659, 273.30851056, 275.67703333,
       284.94722911, 296.57730494, 293.59156906, 291.02255199,
       300.3389946 , 308.43777091, 358.09324901, 354.81780181,
       359.81499789, 364.58935704, 381.99031184, 421.14118444,
       402.82379278, 414.69897276, 412.69381509, 413.20008421,
       382.59038967, 319.16494774, 336.60560204, 344.14833407,
       345.85467885, 331.59895441, 349.87123484, 369.30023458,
       374.99960383, 387.75283043, 393.43283661, 371.88110715,
       368.96368912, 352.83615489, 381.01079197, 385.40098517,
       367.11555799, 355.2069812 , 324.63510962, 325.98458292,
       346.38647212, 362.6532091 , 334.99376156, 334.16469808,
       311.55312684, 313.92192437, 320.4905082 , 305.39808043,
       307.279839  , 292.02066897, 313.25732394, 344.42849078,
       374.39366825, 382.2812711 , 372.03821251, 364.13850093,
       354.32230388, 348.70357588, 410.67226053, 394.4090136 ,
       366.2120209 , 339.42841378, 328.88820248, 348.53904752,
       355.10489028, 377.10616027, 342.31920959, 347.04767192,
       335.5902506 , 328.06070824, 325.72885048, 305.66583144,
       302.21363161, 297.47585718, 311.3832712 , 266.69963662,
       272.53202501, 311.35960279, 322.95801032, 308.66224813,
       291.83783718, 306.33882266, 303.23776208, 310.36697998,
       311.70473629, 301.65291283, 279.96102884, 284.93821072,
       309.12776036, 315.40590254, 292.73112644, 278.02822856,
       262.7343655 , 242.45598718, 259.53548335, 269.04700868,
       301.33152077, 296.01848372, 300.77930474, 276.98862446,
       276.02492994, 260.24065554, 272.48484034, 288.4753472 ,
       283.85076125, 262.12393855, 244.98737653, 262.48598476,
       285.06693988, 302.31073812, 313.2636062 , 316.26300116,
       344.23006641, 344.29459191, 348.12467279, 338.95359866,
       373.2267345 , 376.96979381, 374.04137433, 357.73966515,
       359.57899833, 386.90223199, 366.68157929, 344.10129869,
       341.94284729, 335.07400178, 317.73926792, 320.43455458,
       340.18598878, 330.22789245, 335.96852773, 343.33698463,
       325.76614661, 335.72612271, 332.92117378, 335.00574712,
       278.80490608, 278.52476781, 279.29579759, 289.65212318,
       302.89283371, 306.94273427, 326.17628256, 289.29031621,
       282.77467692, 244.8716685 , 257.40452803, 231.03866559,
       235.33331755, 252.610991  , 241.45495653, 256.56553534,
       231.07636872, 234.14424964, 211.01418685, 220.38402063,
       201.08276963, 218.07220541, 239.38341879, 267.7927916 ,
       264.44476043, 250.58379181, 259.40192469, 229.73993984,
       234.38989534, 232.56598114, 244.22582184, 240.37120712,
       230.19336615, 207.34168478, 213.82357931, 233.04415511,
       213.38444109, 214.48029573, 237.82562538, 244.54121289,
       241.13879596, 248.17744661, 248.29555454, 232.25373008,
       222.44044402, 217.66620677, 225.21881417, 234.49569503,
       234.4152494 , 249.78225997, 256.92384747, 236.01609624,
       228.66142118, 232.30333808, 239.74278171, 242.3769293 ,
       241.76218044, 247.77137677, 250.09398859, 281.28836524,
       282.03777201, 275.52621856, 254.77746543, 274.7505771 ,
       285.4553219 , 309.53037726, 305.35979061, 307.00634609,
       310.6918916 , 312.6254165 , 284.80184859, 283.91080873,
       275.27557887, 295.75118148, 286.07141417, 304.80311289,
       314.49237471, 313.63385901, 305.29109601, 300.66853444,
       293.55285149, 285.2298258 , 293.83541231, 297.4752274 ,
       298.17203012, 285.54937973, 280.54628847, 271.72334084,
       271.26238193, 274.48451299, 268.02535002, 273.60720004,
       285.71710248, 295.75262722, 307.91872558, 311.53941468,
       293.68925411, 302.73810424, 303.90219895, 307.12913473,
       311.00056705, 310.61860961, 300.21240271, 281.90404045,
       264.22695853, 267.24351466, 280.85615992, 292.05496581,
       266.48644263, 258.7122967 , 228.00761666, 242.0737504 ,
       233.27891454, 236.46541328, 214.73560176, 216.45079751,
       209.5443925 , 215.02667657, 221.18469262, 201.92264974,
       218.89829677, 220.22035271, 227.66346642, 203.85578501,
       211.70609075, 208.0637181 , 226.93220446, 228.38687204,
       229.75140704, 230.44308525, 228.67044743, 228.6770193 ,
       210.94070331, 210.7368476 , 201.00341627, 191.19082543,
       183.12643832, 167.92321476, 187.62092001, 197.85678194,
       195.75896677, 195.33011895, 184.31263474, 180.4227056 ,
       175.42550513, 161.85723995, 164.95730274, 183.82372693,
       188.36644154, 187.21855359, 180.11802238, 198.06893261,
       199.15326534, 199.50677075, 178.81901177, 174.48640187,
       167.23030162, 170.24489295, 178.57901409, 166.3631238 ,
       156.60054738, 149.29310217, 154.11085652, 146.96426166,
       147.29090206, 131.61013451, 132.90381368, 117.62848528,
       117.88543697, 100.0536115 , 107.84091311, 121.20960142,
       128.16162957, 107.14109292, 110.1431367 , 106.41380886,
       114.48143957, 121.68031042, 121.96801711, 126.40313815,
       124.97747459, 126.97420143, 128.2066447 , 129.78402212,
       131.37246908, 132.83876057, 134.38721168, 135.91844078,
       137.43452706, 138.96696384, 140.49293351, 142.01802546,
       143.54587857, 145.07208265, 146.59852981, 148.12535298,
       149.65183135, 151.1784273 , 152.70505497, 154.23162137,
       155.7582203 , 157.28481669, 158.81140407, 160.33799863,
       161.86459113, 163.3911827 , 164.9177756 , 166.44436788,
       167.97096016, 169.49755264, 171.02414498, 172.55073735,
       174.07732974, 175.60392211, 177.13051449, 178.65710687,
       180.18369925, 181.71029163, 183.236884  , 184.76347638,
       186.29006876, 187.81666114, 189.34325352, 190.86984589,
       192.39643827, 193.92303065, 195.44962303, 196.97621541,
       198.50280778, 200.02940016, 201.55599254, 203.08258492,
       204.6091773 , 206.13576967, 207.66236205, 209.18895443,
       210.71554681, 212.24213919, 213.76873156, 215.29532394,
       216.82191632, 218.3485087 ])
```

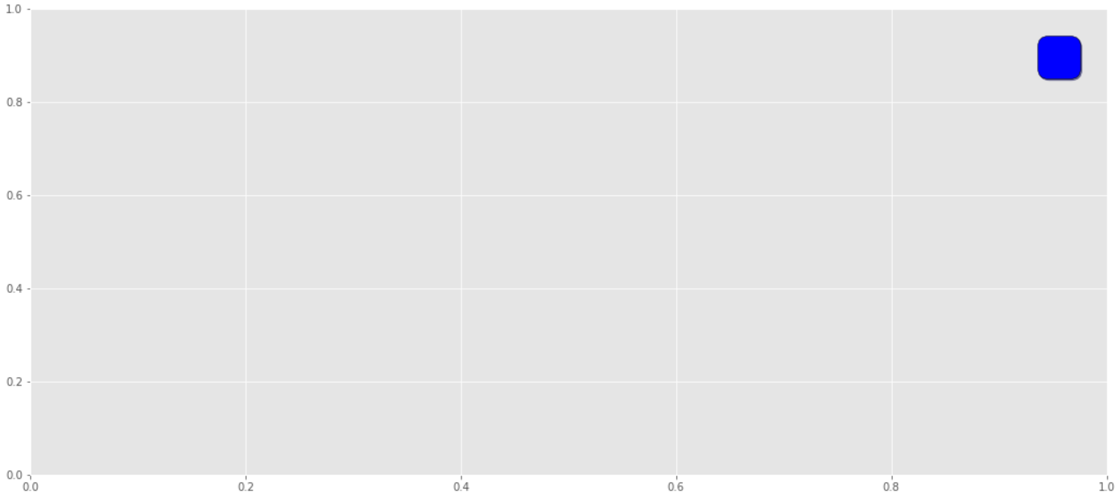
[193]:

```
plt.figure(figsize=(18,8))
plt.grid(True)
plt.plot(range(len(test_data)),test_data, label = 'True Test Close Value',␣
 ↪linewidth = 5)
plt.plot(range(len(predictions)), predictions, label = 'Predictions on test␣
 ↪data', linewidth = 5)
plt.xticks(fontsize = 15)
plt.xticks(fontsize = 15)
plt.legend(fontsize = 20, shadow=True,facecolor='lightpink',edgecolor = 'k')
plt.show()
```



[217]:

```
fig = plt.figure(figsize=(18,8))
ax1 = fig.add_subplot(111)
model_fit.predict(start=1, end=len(df_close)+60, ax = ax1)
plt.grid("both")
plt.legend(['Forecast','Close','95% confidence interval'],fontsize = 50,␣
 ↪shadow=True,facecolor='blue',edgecolor = 'k')
plt.show()
```



[ ]:

```

```