In [63]:
```python
import math
import pandas as pd
import numpy as np
import pandas_datareader as web
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense,LSTM
import matplotlib.pyplot as plt
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.arima_model import ARMA,ARMAResults,ARIMA,ARIMAResults
from statsmodels.stats.diagnostic import acorr_ljungbox
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf,plot_predict
from pmdarima import auto_arima
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
from statsmodels.tsa.holtwinters import ExponentialSmoothing
plt.style.use('fivethirtyeight')
from datetime import datetime , date
import warnings
warnings.filterwarnings("ignore")
from sklearn.metrics import mean_squared_error
import pylab
import scipy.stats as stats
from scipy.stats import norm
from math import log
from pmdarima.utils import diff_inv
```

In [199…
```python
def create_dataset(df):
    x = []
    y = []
    for i in range(30, df.shape[0]):
        x.append(df[i-30:i, 0])
        y.append(df[i, 0])
    x = np.array(x)
    y = np.array(y)
    return x,y
```

In [65]:
```python
# reading data from csv file
NFLX=pd.read_csv('D:\\R-documantary\\NFLX.csv',index_col='date',parse_dates=True)
dataset=NFLX[NFLX.index>='2018-01-01']
dataset=dataset.dropna()
```

In [67]:
```python
# take a look at the dataset
dataset
```

Out[67]:

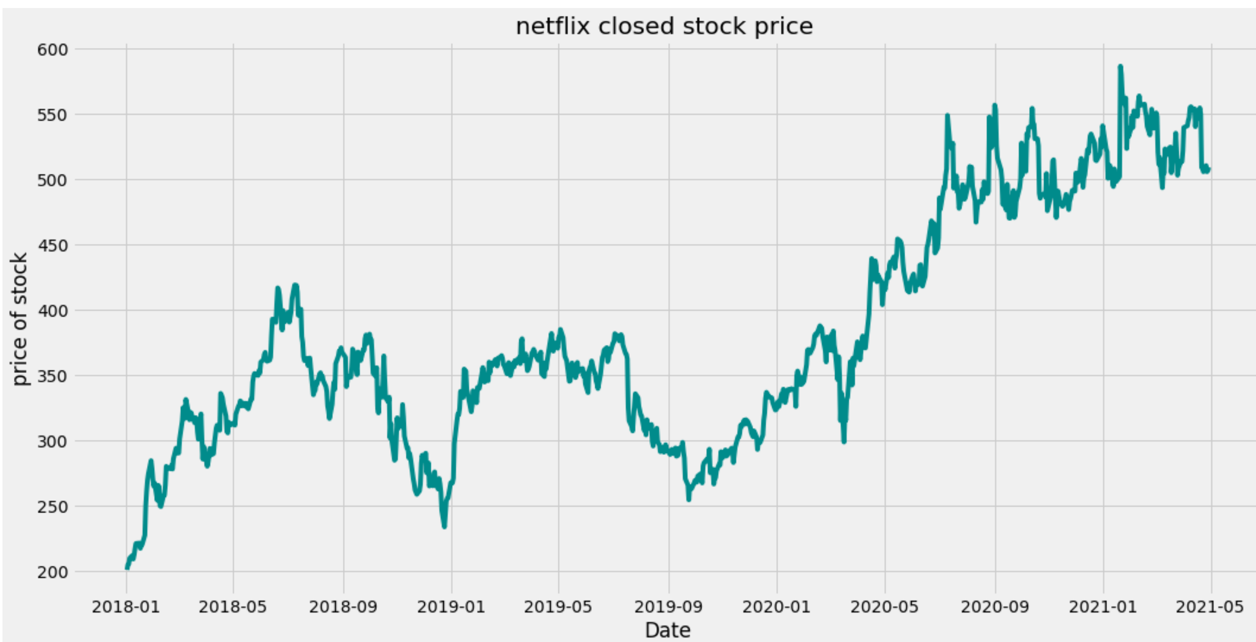
	close
date	
2018-01-02	201.070007
2018-01-03	205.050003
2018-01-04	205.630005
2018-01-05	209.990005

	close
date	
2018-01-08	212.050003
...	...
2021-04-23	505.549988
2021-04-26	510.299988
2021-04-27	505.549988
2021-04-28	506.519989
2021-04-29	509.000000

837 rows × 1 columns

In [71]:

```
#plotting our data to see the patterns and see if it needs transformation for stationar
plt.figure(figsize=(16,8))
plt.plot(dataset,color='darkcyan')
plt.xlabel('Date')
plt.ylabel('price of stock')
plt.title('netflix closed stock price ')
plt.show()
```



In [72]:

```
dataset.describe()
```

Out[72]:

	close
count	837.000000
mean	381.247312
std	88.776410
min	201.070007
25%	315.339996

	close
50%	359.970001
75%	466.929993
max	586.340027

In [74]:
```
round(0.9 * len(dataset))
```

Out[74]: 753

In [73]:
```
#defining train and test datasets
train_size=round(0.9 * len(dataset))
train=dataset[:train_size]
test=dataset[train_size:]
test.head()
```

Out[73]:

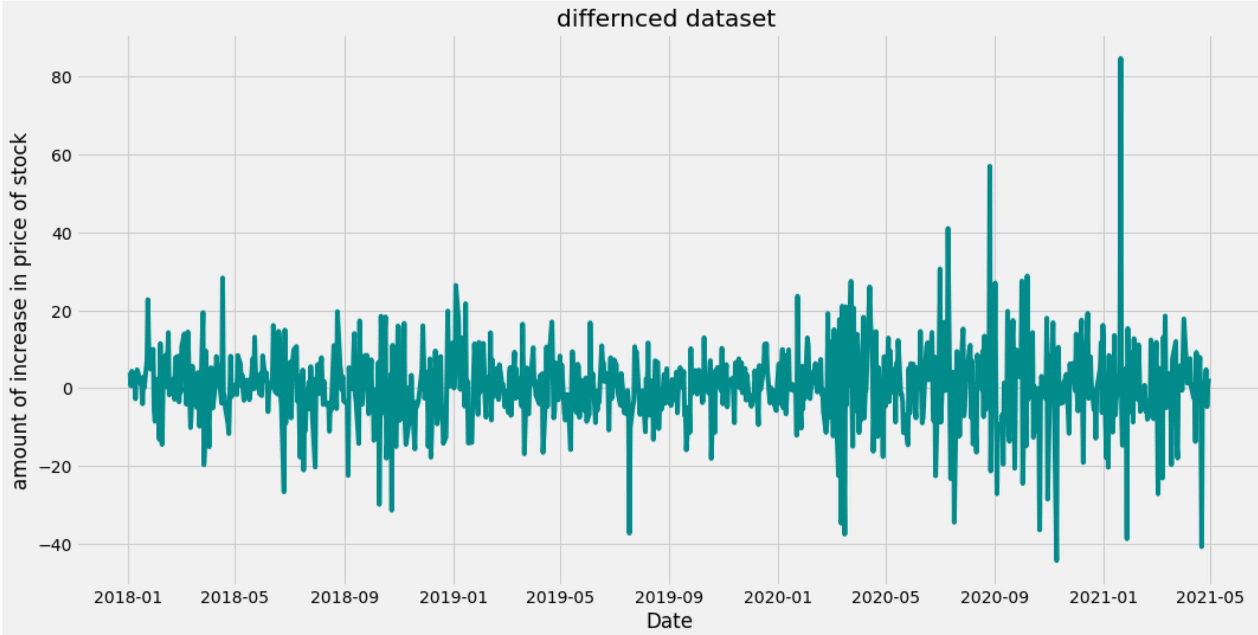
	close
date	
2020-12-29	530.869995
2020-12-30	524.590027
2020-12-31	540.729980
2021-01-04	522.859985
2021-01-05	520.799988

In [75]:
```
#differenced datasets
differenced_train=dataset.diff().dropna()[:train_size]
differenced_test=dataset.diff().dropna()[train_size:]
differenced_test.head()
```

Out[75]:

	close
date	
2020-12-30	-6.279968
2020-12-31	16.139953
2021-01-04	-17.869995
2021-01-05	-2.059997
2021-01-06	-20.309998

In [78]:
```
plt.figure(figsize=(16,8))
plt.plot(dataset.diff().dropna(),color='darkcyan')
plt.xlabel('Date')
plt.ylabel('amount of increase in price of stock')
plt.title('differnced dataset ')
plt.show()
```

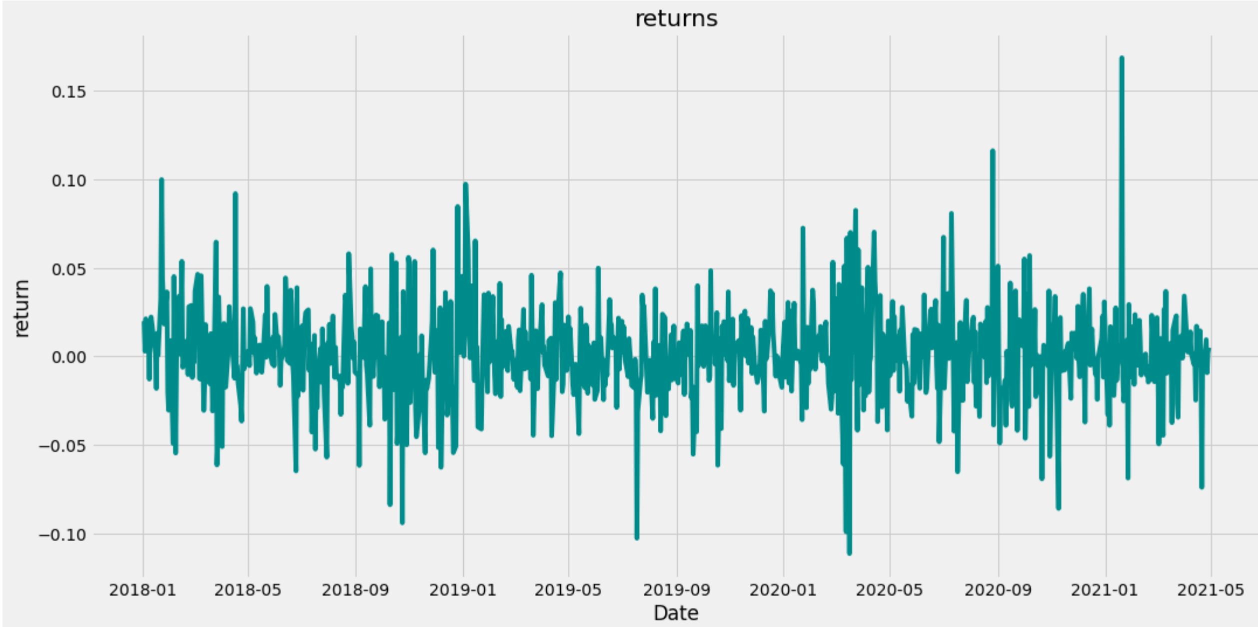


In [83]:
```
# stationarity test
result=adfuller(differenced_train['close'])
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

```
ADF Statistic: -11.273619
p-value: 0.000000
Critical Values:
	1%: -3.439
	5%: -2.865
	10%: -2.569
```

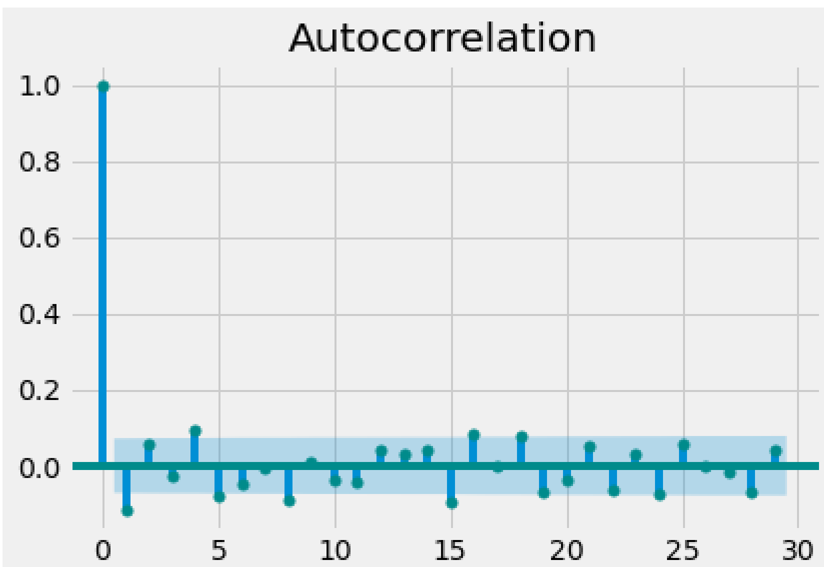
In [80]:
```
returns=dataset.pct_change().dropna()
returns.head()
train_returns=returns[:train_size]
test_returns=returns[train_size:]
```

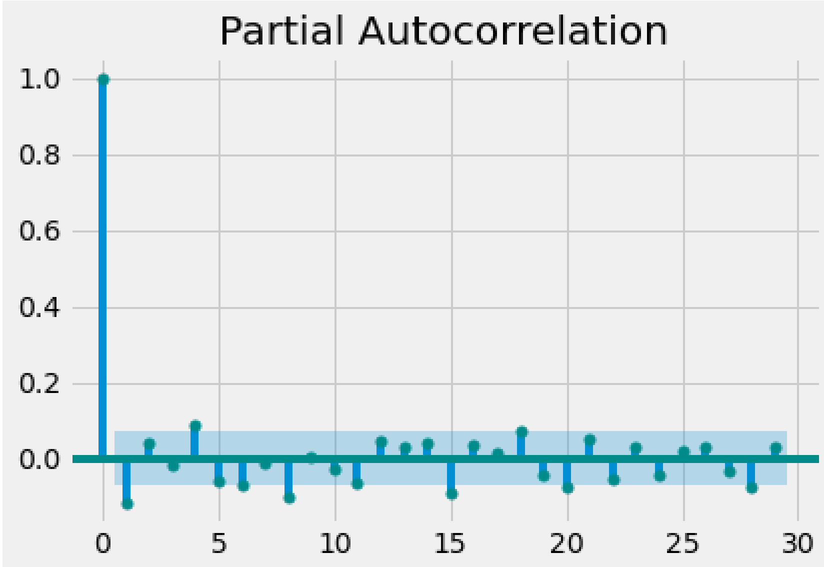
In [79]:
```
plt.figure(figsize=(16,8))
plt.plot(returns,color='darkcyan')
plt.xlabel('Date')
plt.ylabel('return')
plt.title('returns')
plt.show()
```



ARIMA MODEL

In [82]:
```
plot_acf(differenced_train)
plot_pacf(differenced_train)
plt.show()
```





In [84]:
```
auto_arima(train['close'],trace=True,stepwise=False,max_p=10,max_q=10)
```

```
 ARIMA(0,1,0)(0,0,0)[1] intercept   : AIC=5585.505, Time=0.75 sec
 ARIMA(0,1,1)(0,0,0)[1] intercept   : AIC=5578.246, Time=0.21 sec
 ARIMA(0,1,2)(0,0,0)[1] intercept   : AIC=5578.312, Time=0.20 sec
 ARIMA(0,1,3)(0,0,0)[1] intercept   : AIC=5580.307, Time=0.22 sec
 ARIMA(0,1,4)(0,0,0)[1] intercept   : AIC=5576.322, Time=0.37 sec
 ARIMA(0,1,5)(0,0,0)[1] intercept   : AIC=5571.827, Time=0.51 sec
 ARIMA(1,1,0)(0,0,0)[1] intercept   : AIC=5577.354, Time=0.09 sec
 ARIMA(1,1,1)(0,0,0)[1] intercept   : AIC=5576.300, Time=0.27 sec
 ARIMA(1,1,2)(0,0,0)[1] intercept   : AIC=5578.114, Time=0.24 sec
 ARIMA(1,1,3)(0,0,0)[1] intercept   : AIC=5579.956, Time=0.36 sec
 ARIMA(1,1,4)(0,0,0)[1] intercept   : AIC=5575.240, Time=0.64 sec
 ARIMA(2,1,0)(0,0,0)[1] intercept   : AIC=5577.983, Time=0.29 sec
 ARIMA(2,1,1)(0,0,0)[1] intercept   : AIC=5578.125, Time=0.38 sec
 ARIMA(2,1,2)(0,0,0)[1] intercept   : AIC=5580.100, Time=0.56 sec
 ARIMA(2,1,3)(0,0,0)[1] intercept   : AIC=5582.013, Time=0.74 sec
 ARIMA(3,1,0)(0,0,0)[1] intercept   : AIC=5579.862, Time=0.23 sec
 ARIMA(3,1,1)(0,0,0)[1] intercept   : AIC=5579.921, Time=0.50 sec
 ARIMA(3,1,2)(0,0,0)[1] intercept   : AIC=5573.182, Time=0.86 sec
 ARIMA(4,1,0)(0,0,0)[1] intercept   : AIC=5576.153, Time=0.29 sec
 ARIMA(4,1,1)(0,0,0)[1] intercept   : AIC=5577.227, Time=0.55 sec
 ARIMA(5,1,0)(0,0,0)[1] intercept   : AIC=5575.651, Time=0.24 sec

Best model:  ARIMA(0,1,5)(0,0,0)[1] intercept
Total fit time: 8.863 seconds
```

Out[84]:
```
ARIMA(order=(0, 1, 5), scoring_args={}, seasonal_order=(0, 0, 0, 1),
      suppress_warnings=True)
```

In [85]:
```
arima_model=ARIMA(train['close'],order=(0,1,5))
arima_model=arima_model.fit()
print(arima_model.summary())
```

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:                D.close   No. Observations:                  752
Model:                 ARIMA(0, 1, 5)   Log Likelihood               -2778.913
Method:                       css-mle   S.D. of innovations              9.741
Date:                Fri, 09 Jul 2021   AIC                           5571.827
Time:                        17:12:04   BIC                           5604.186
Sample:                             1   HQIC                          5584.294

==================================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
```

```
--------------------------------------------------------------------------------
const              0.4232      0.327      1.292      0.196      -0.219       1.065
ma.L1.D.close     -0.1099      0.037     -3.006      0.003      -0.181      -0.038
ma.L2.D.close      0.0533      0.036      1.475      0.140      -0.018       0.124
ma.L3.D.close     -0.0256      0.038     -0.674      0.500      -0.100       0.049
ma.L4.D.close      0.1059      0.040      2.678      0.007       0.028       0.183
ma.L5.D.close     -0.1019      0.040     -2.574      0.010      -0.180      -0.024
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
MA.1           -1.0847          -0.9931j            1.4706           -0.3820
MA.2           -1.0847          +0.9931j            1.4706            0.3820
MA.3            0.7022          -1.4220j            1.5859           -0.1770
MA.4            0.7022          +1.4220j            1.5859            0.1770
MA.5            1.8034          -0.0000j            1.8034           -0.0000
-----------------------------------------------------------------------------
```

In [87]:

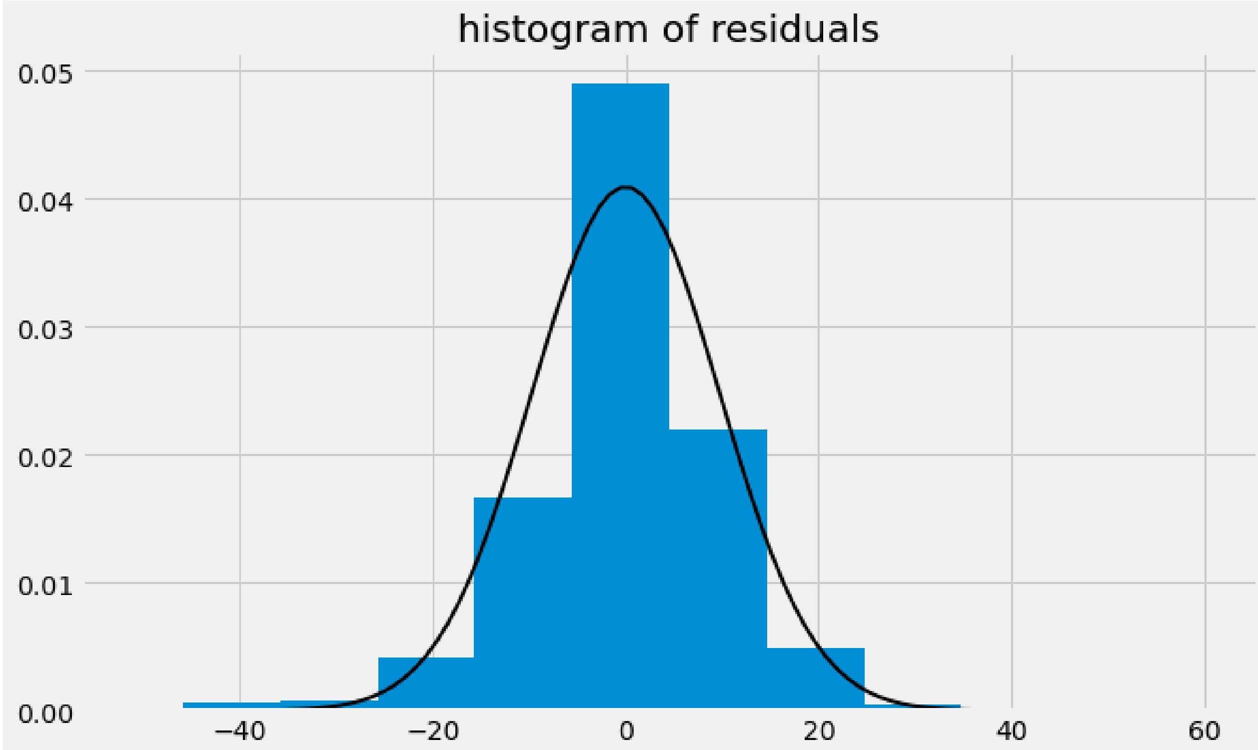
```
train.head()
```

Out[87]:

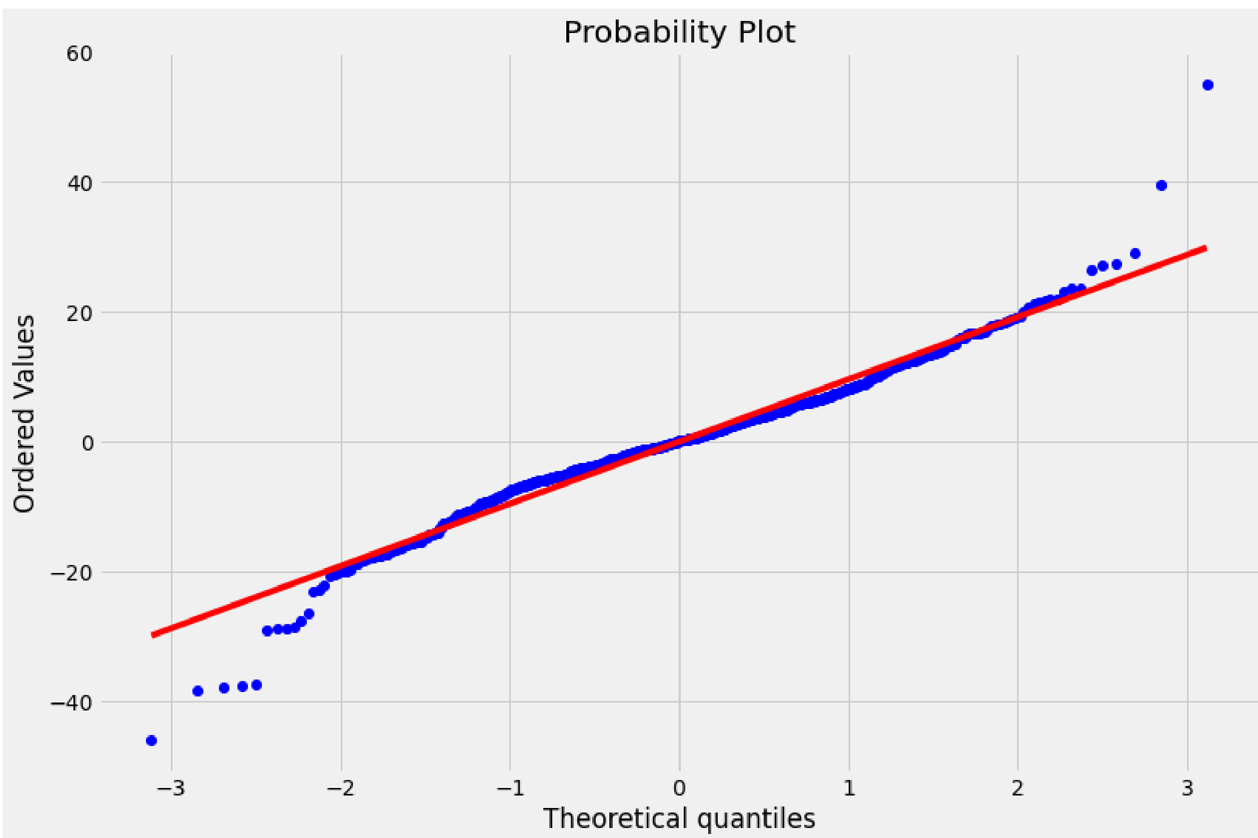
	close
date	
2018-01-02	201.070007
2018-01-03	205.050003
2018-01-04	205.630005
2018-01-05	209.990005
2018-01-08	212.050003

In [89]:

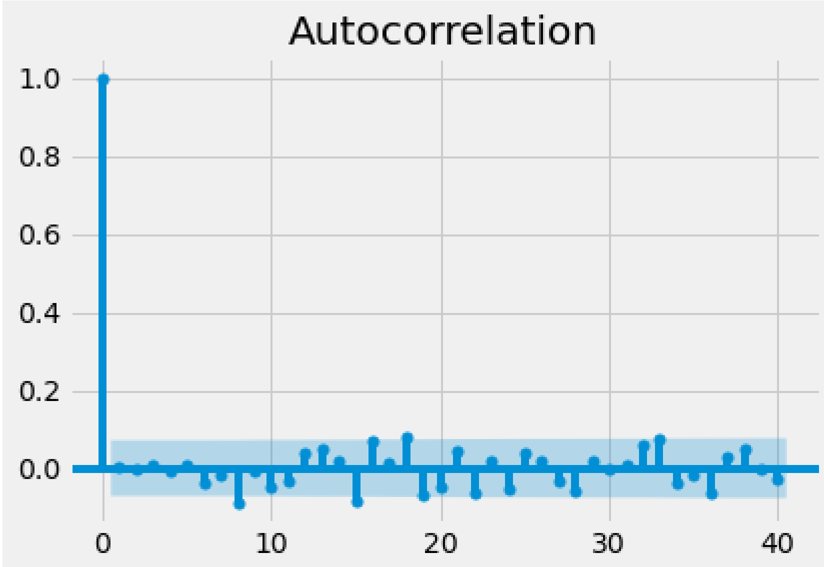
```
plt.figure(figsize=(10,6))
plt.hist(arima_model.resid,density=True)
plt.title('histogram of residuals')
mu, std = norm.fit(arima_model.resid)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, mu, std)
plt.plot(x, p, 'k', linewidth=2)
plt.show()
```

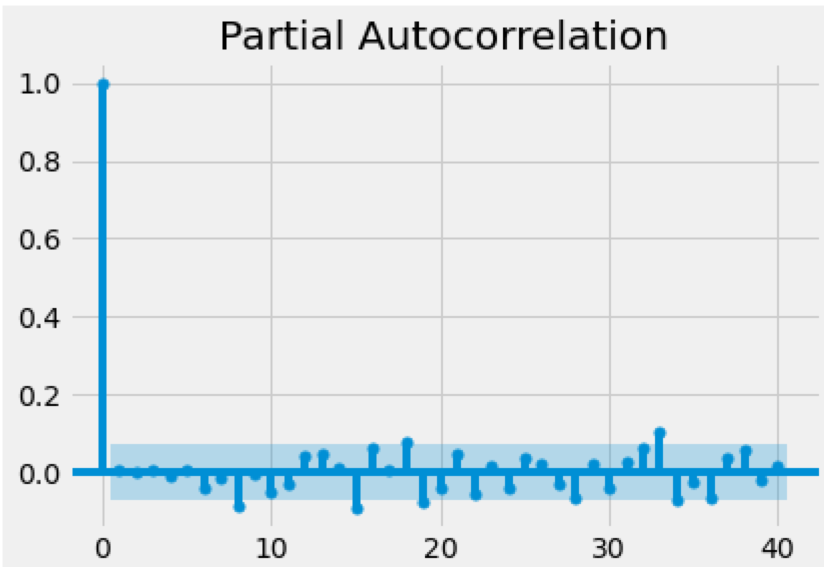


In [91]:
```
plt.figure(figsize=(12,8))
stats.probplot(arima_model.resid, dist="norm", plot=pylab)
plt.show()
```



In [95]:
```
plot_acf(arima_model.resid,lags=40)
plot_pacf(arima_model.resid,lags=40)
plt.show()
```





In [97]:
```
acorr_ljungbox(arima_model.resid,return_df=True,lags=20,boxpierce=True)
```

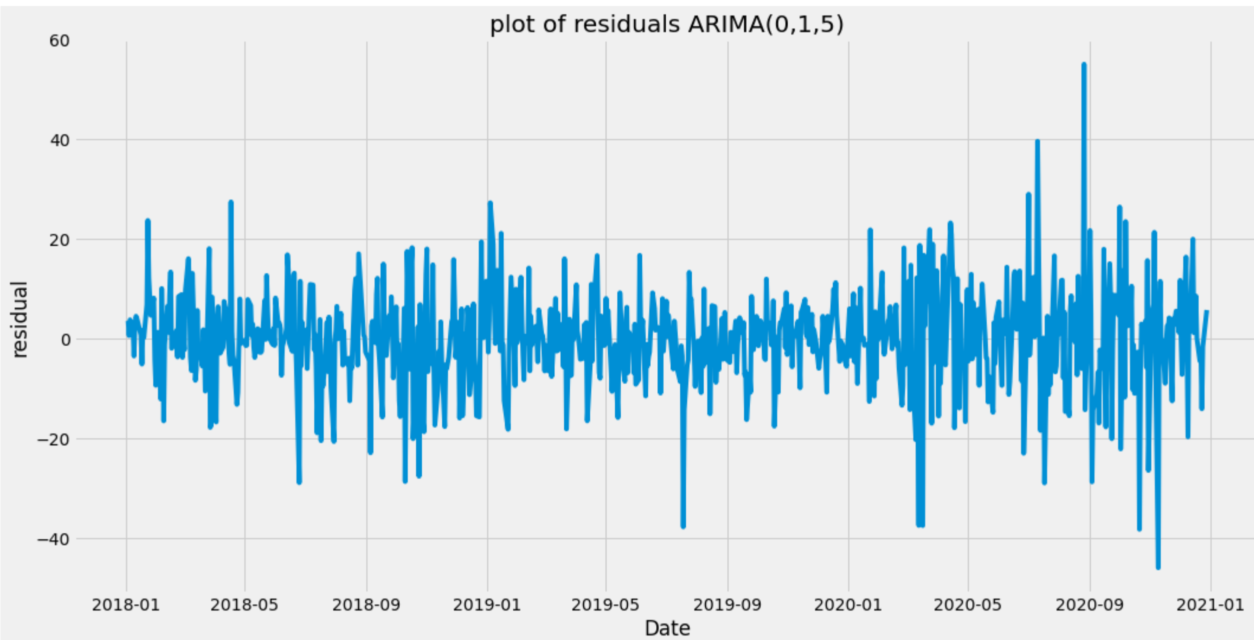
Out[97]:

	lb_stat	lb_pvalue	bp_stat	bp_pvalue
1	0.007739	0.929899	0.007708	0.930038
2	0.008730	0.995645	0.008694	0.995663
3	0.048189	0.997227	0.047891	0.997252
4	0.094221	0.998925	0.093557	0.998939
5	0.139683	0.999631	0.138597	0.999638
6	1.275105	0.973013	1.261972	0.973713
7	1.521556	0.981554	1.505482	0.982119
8	7.398120	0.494349	7.304107	0.504205
9	7.445984	0.590790	7.351273	0.600597
10	9.158638	0.517117	9.036670	0.528627
11	10.029214	0.527761	9.892236	0.540106

	lb_stat	lb_pvalue	bp_stat	bp_pvalue
12	11.333266	0.500594	11.172074	0.514232
13	13.035429	0.445079	12.840375	0.460216
14	13.263981	0.505851	13.064077	0.521484
15	18.726336	0.226404	18.403275	0.242058
16	22.436012	0.129663	22.024391	0.142406
17	22.589018	0.163114	22.173541	0.178137
18	27.293636	0.073650	26.753369	0.083739
19	30.960230	0.040780	30.317843	0.047890
20	32.791281	0.035563	32.095468	0.042292

In [98]:

```
plt.figure(figsize=(16,8))
plt.xlabel('Date')
plt.ylabel('residual')
plt.plot(arima_model.resid)
plt.title('plot of residuals ARIMA(0,1,5)')
plt.show()
```



In [99]:

```
arima_pred=[]
for i in range(len(test)):
    model=ARIMA(dataset[:train_size+i],order=(0,1,5))
    model=model.fit()
    start=len(train)+i
    arima_pred.append(model.predict(start,start,typ='levels'))
```

In [103…

```
test['arima(1 step ahead)']=arima_pred
```

In [113…

```
arima_pred2=arima_model.forecast(steps=84,alpha=0.05)
test['upper_interval']=arima_pred2[2][:,1]
test['lower_interval']=arima_pred2[2][:,0]
test['prediction(without updating model)']=arima_pred2[0]
```

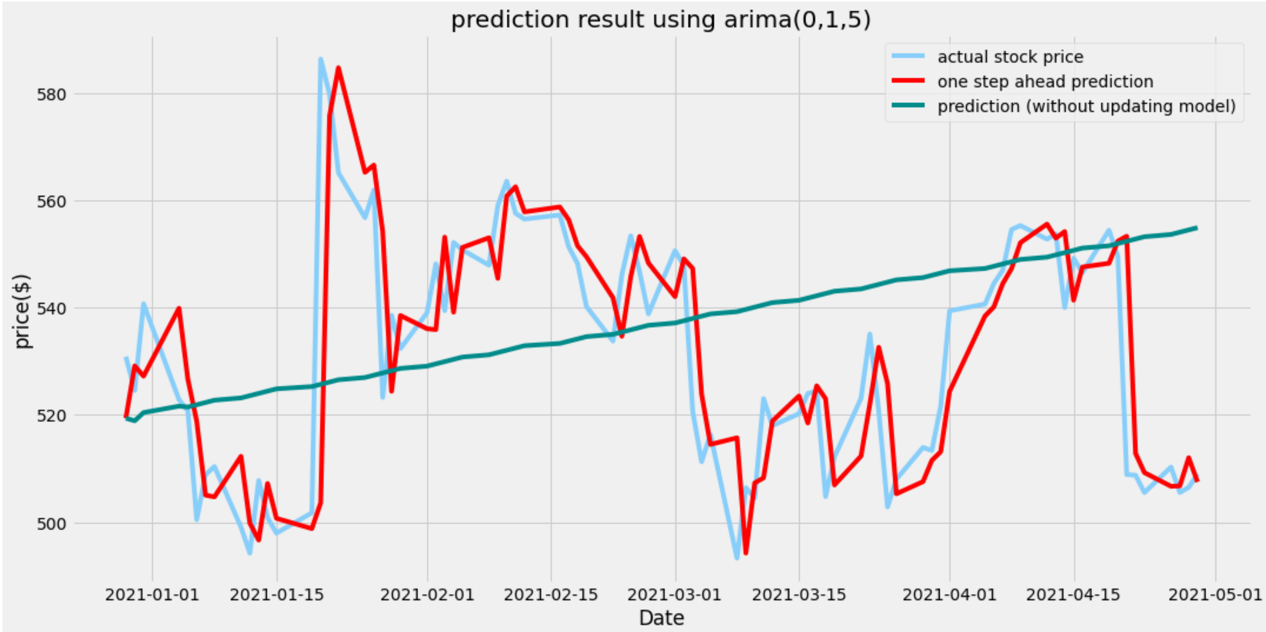
In [114...
```
test
```

Out[114...

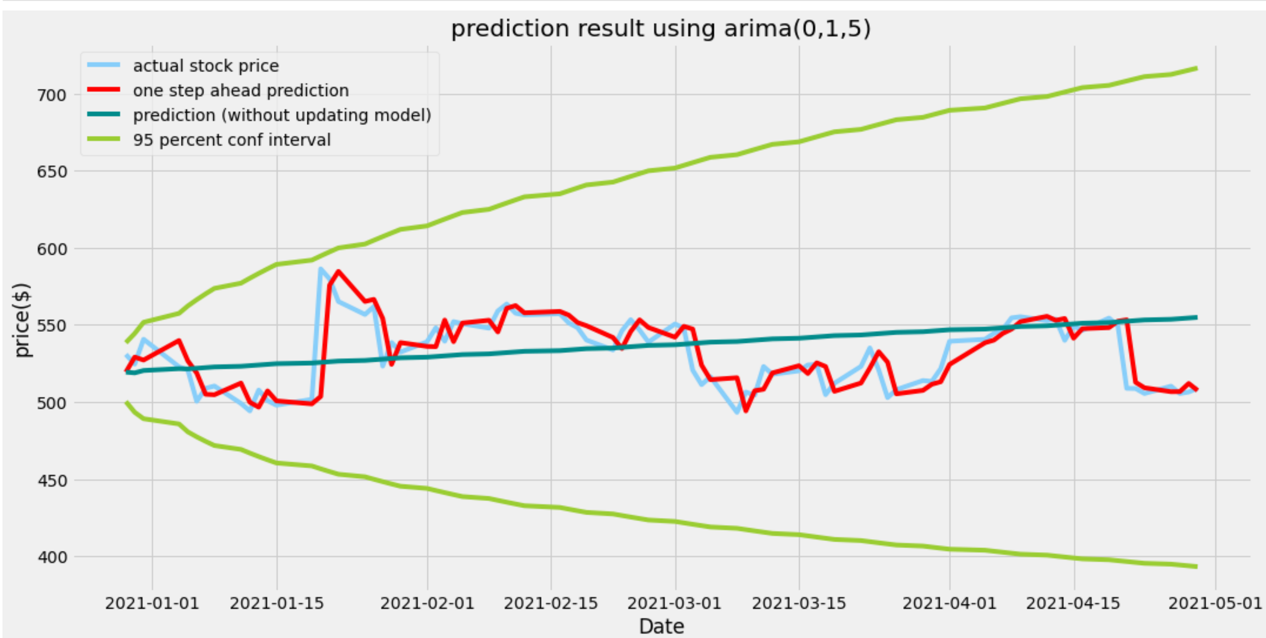
date	close	arima(1 step ahead)	upper_interval	lower_interval	prediction(without updating model)
2020-12-29	530.869995	752 519.402626 dtype: float64	538.494883	500.310368	519.402626
2020-12-30	524.590027	753 529.157691 dtype: float64	544.477397	493.356263	518.916830
2020-12-31	540.729980	754 527.239778 dtype: float64	551.726140	489.186979	520.456559
2021-01-04	522.859985	755 539.908122 dtype: float64	557.493992	485.804806	521.649399
2021-01-05	520.799988	756 526.684286 dtype: float64	562.315842	480.662902	521.489372
...	...	...	...	...	...
2021-04-23	505.549988	831 509.263792 dtype: float64	711.004728	395.455594	553.230161
2021-04-26	510.299988	832 506.715746 dtype: float64	712.406306	394.900437	553.653372
2021-04-27	505.549988	833 506.785502 dtype: float64	713.801891	394.351273	554.076582
2021-04-28	506.519989	834 512.049115 dtype: float64	715.191593	393.807992	554.499793
2021-04-29	509.000000	835 507.523816 dtype: float64	716.575516	393.270491	554.923003

84 rows × 5 columns

In [148...
```
plt.figure(figsize=(16,8))
plt.plot(test['close'],label='actual stock price',color='lightskyblue')
plt.plot(test['arima(1 step ahead)'],label='one step ahead prediction',color='red')
plt.plot(test['prediction(without updating model)'],label='prediction (without updating
plt.title('prediction result using arima(0,1,5)')
plt.xlabel('Date')
plt.ylabel('price($)')
plt.legend()
plt.show()
```



In [153…
```
plt.figure(figsize=(16,8))
plt.plot(test['close'],label='actual stock price',color='lightskyblue')
plt.plot(test['arima(1 step ahead)'],label='one step ahead prediction',color='red')
plt.plot(test['prediction(without updating model)'],label='prediction (without updating
plt.plot(test['upper_interval'],label='95 percent conf interval',color='yellowgreen')
plt.plot(test['lower_interval'],color='yellowgreen')
plt.title('prediction result using arima(0,1,5)')
plt.xlabel('Date')
plt.ylabel('price($)')
plt.legend()
plt.show()
```



calculate MSE of one step ahead prediction

In [119…
```
mean_squared_error(test['close'],test['arima(1 step ahead)'])
```

Out[119… 205.05949777109194

exponential smoothing models (simple model)

In [120…

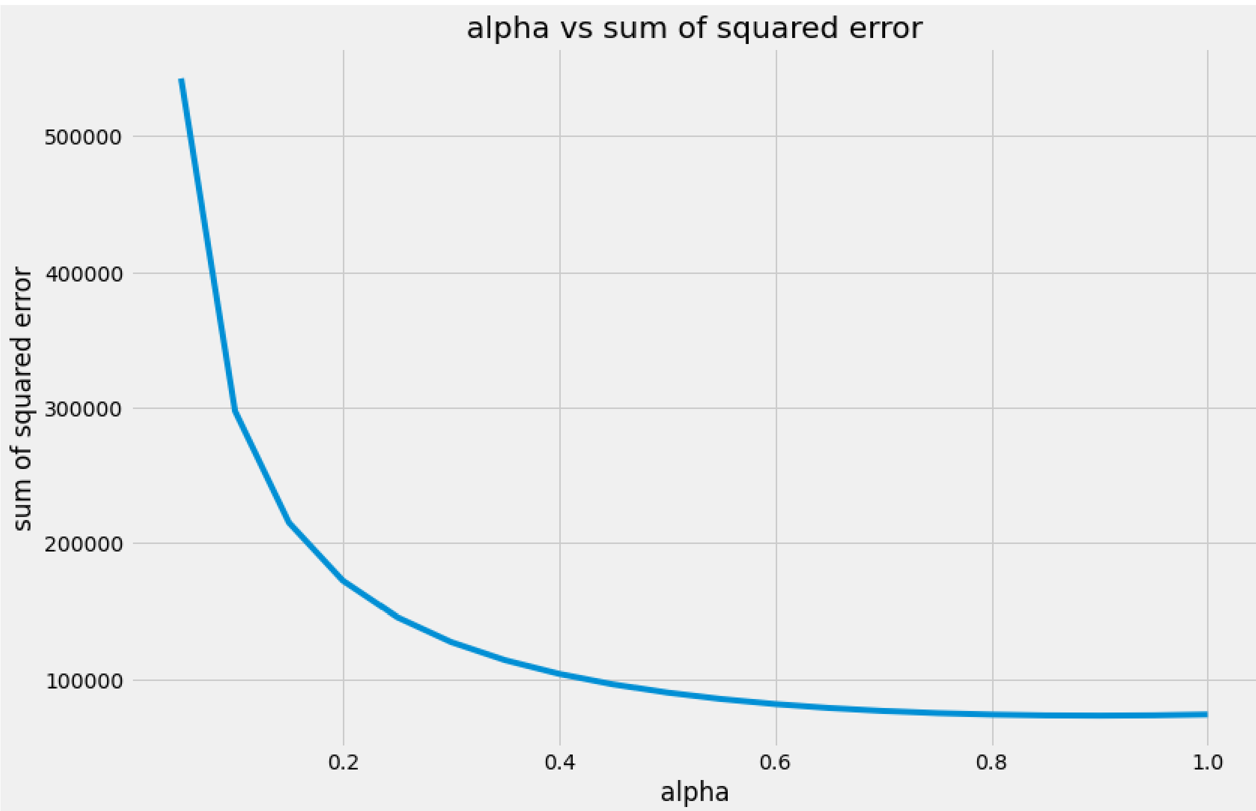
```
alpha=(0.05, 0.1 , 0.15, 0.2 , 0.25, 0.3 , 0.35, 0.4 , 0.45, 0.5 ,
       0.55, 0.6 , 0.65, 0.7 , 0.75, 0.8 , 0.85, 0.9 , 0.95,1.00)
```

In [124…

```
sum_of_squared_error=[]
for i in alpha:
    model=SimpleExpSmoothing(train['close']).fit(smoothing_level=i,initial_level=train[
    sum_of_squared_error.append(model.sse)
```

In [130…

```
plt.figure(figsize=(12,8))
plt.plot(alpha,sum_of_squared_error)
plt.title('alpha vs sum of squared error')
plt.ylabel('sum of squared error')
plt.xlabel('alpha')
plt.show()
```



In [132…

```
simple_model=SimpleExpSmoothing(train['close']).fit()
print(simple_model.summary())
```

```
                       SimpleExpSmoothing Model Results
==============================================================================
Dep. Variable:                    close   No. Observations:                  753
Model:               SimpleExpSmoothing   SSE                          72914.115
Optimized:                         True   AIC                           3447.448
Trend:                             None   BIC                           3456.696
Seasonal:                          None   AICC                          3447.502
Seasonal Periods:                  None   Date:                 Fri, 09 Jul 2021
Box-Cox:                          False   Time:                         17:31:23
Box-Cox Coeff.:                    None
==============================================================================
                       coeff                 code              optimized
```

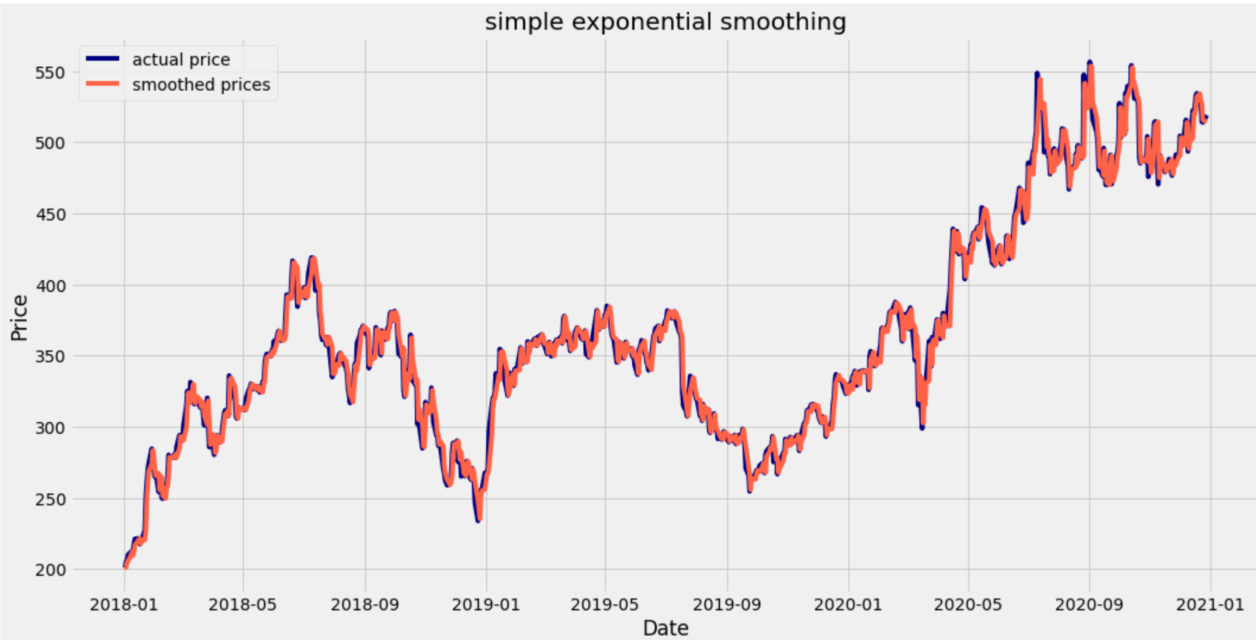
```
==============================================================================
smoothing_level          0.8963960              alpha                True
initial_level            201.49371                1.0                True
------------------------------------------------------------------------------
```

In [136…

```
train['SES fitted values']=simple_model.fittedvalues
```

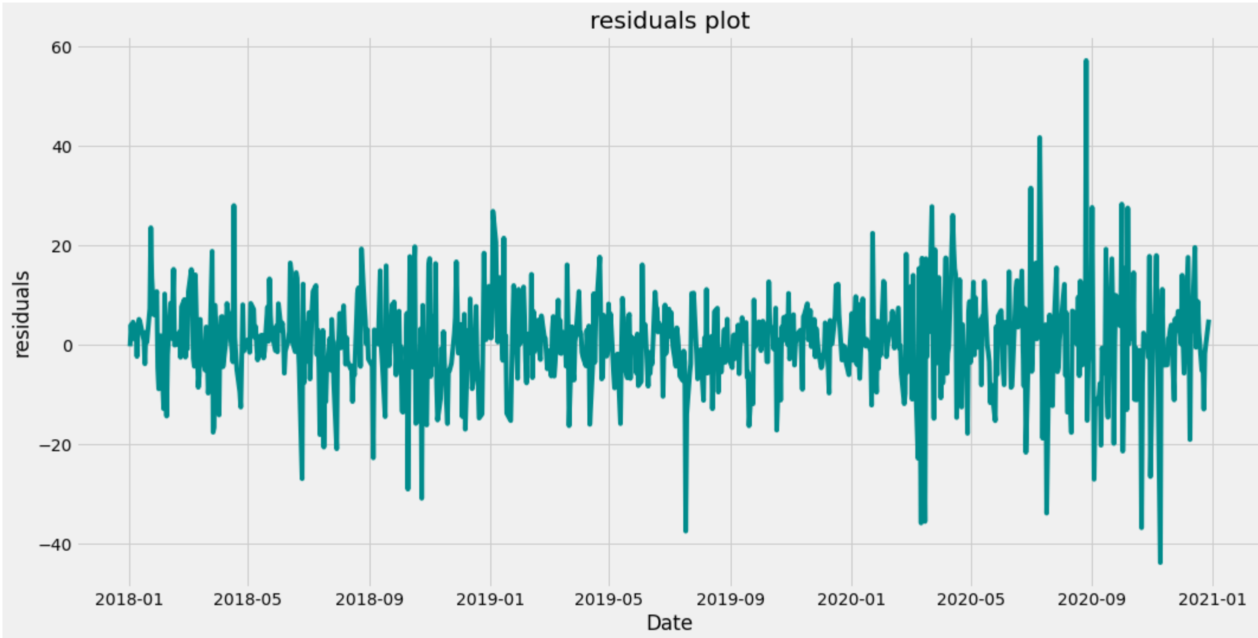
In [145…

```
plt.figure(figsize=(16,8))
plt.plot(train['close'],label='actual price',color='navy')
plt.plot(train['SES fitted values'],label='smoothed prices',color='tomato')
plt.xlabel('Date')
plt.ylabel('Price')
plt.title('simple exponential smoothing')
plt.legend()
plt.show()
```



In [147…

```
plt.figure(figsize=(16,8))
plt.plot(simple_model.resid,color='darkcyan')
plt.xlabel('Date')
plt.ylabel('residuals')
plt.title('residuals plot')
plt.show()
```



In [154…

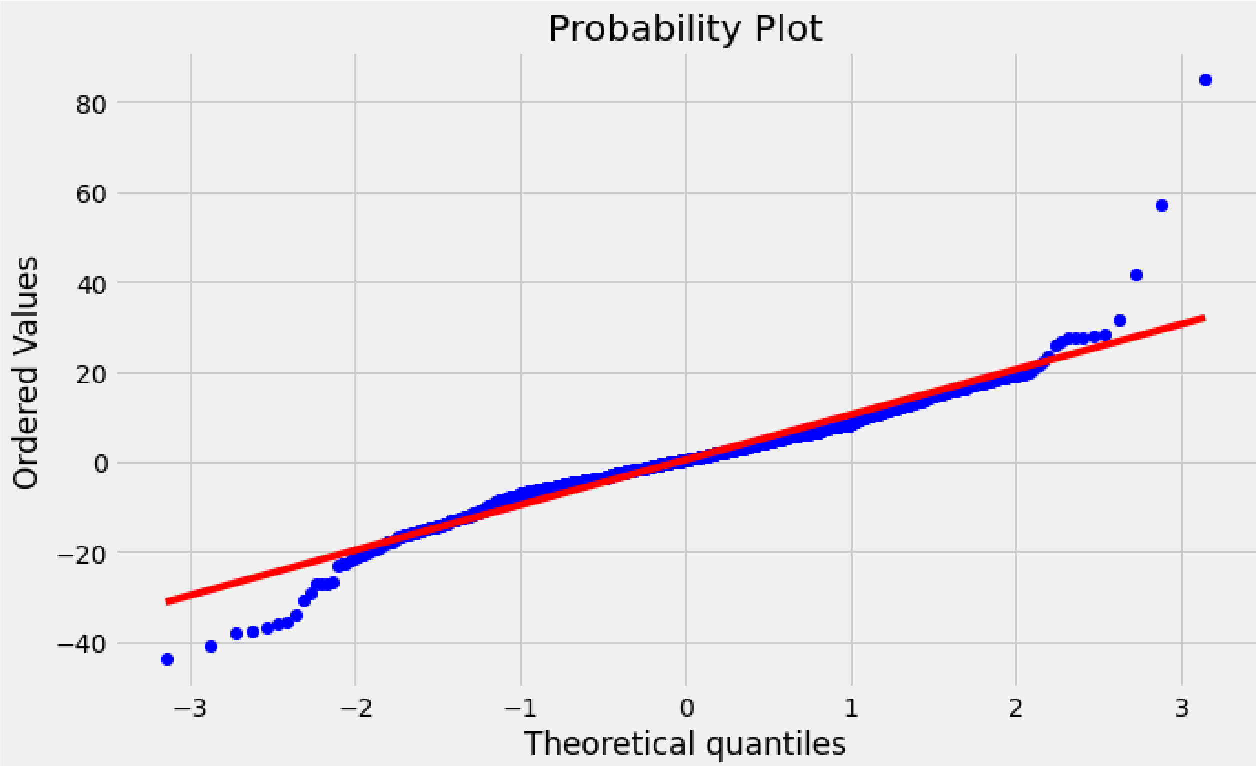
```
acorr_ljungbox(simple_model.resid,lags=20,return_df=True,boxpierce=True)
```

Out[154…

	lb_stat	lb_pvalue	bp_stat	bp_pvalue
1	0.049345	0.824207	0.049149	0.824551
2	2.244913	0.325479	2.233085	0.327410
3	2.315222	0.509611	2.302928	0.511960
4	7.619350	0.106560	7.564904	0.108881
5	11.890964	0.036313	11.796914	0.037679
6	14.272591	0.026735	14.153305	0.027970
7	14.647647	0.040791	14.523890	0.042610
8	21.321387	0.006341	21.109237	0.006863
9	21.329868	0.011264	21.117593	0.012138
10	22.971521	0.010852	22.733154	0.011776
11	24.382267	0.011214	24.119609	0.012238
12	25.586356	0.012276	25.301370	0.013458
13	26.873072	0.012948	26.562523	0.014273
14	27.848150	0.014903	27.516937	0.016479
15	33.756972	0.003683	33.292713	0.004275
16	38.283402	0.001379	37.711228	0.001664
17	38.441499	0.002136	37.865346	0.002565
18	42.294589	0.001006	41.616368	0.001251
19	46.112667	0.000478	45.328248	0.000616
20	47.505523	0.000499	46.680517	0.000650

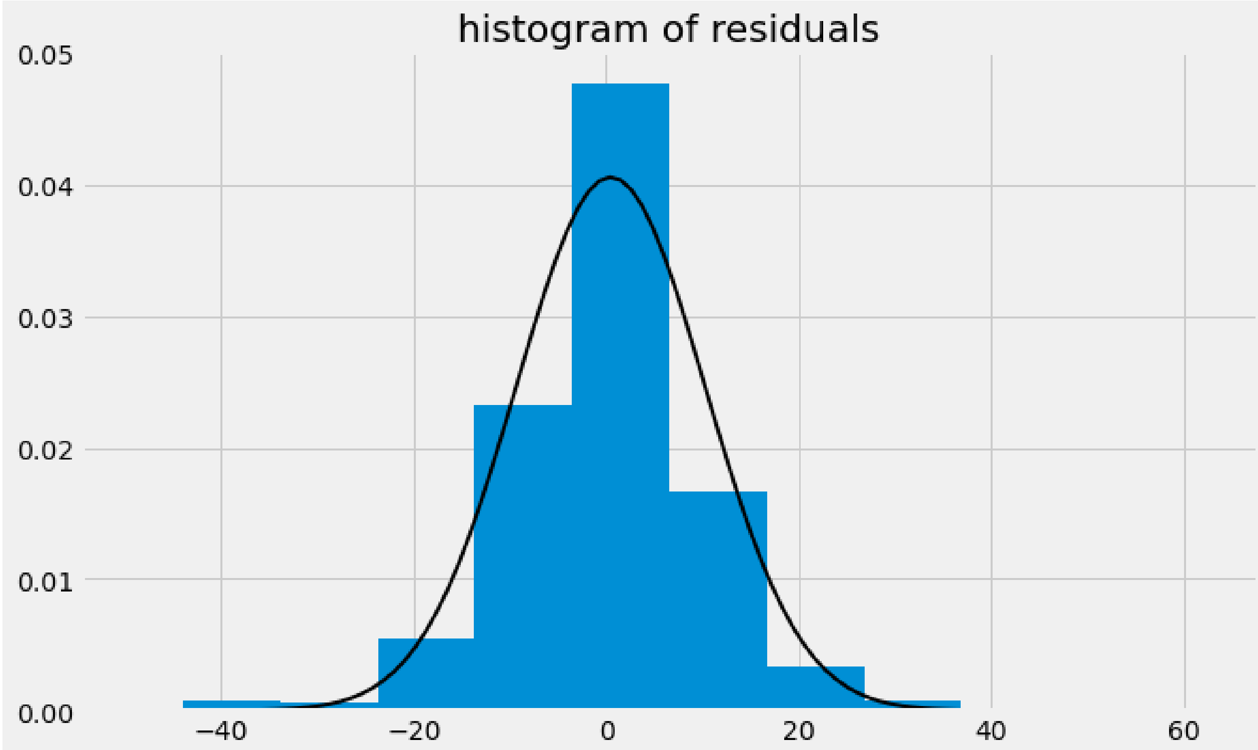
In [283…

```
plt.figure(figsize=(10,6))
stats.probplot(simple_model.resid, dist="norm", plot=pylab)
plt.show()
```



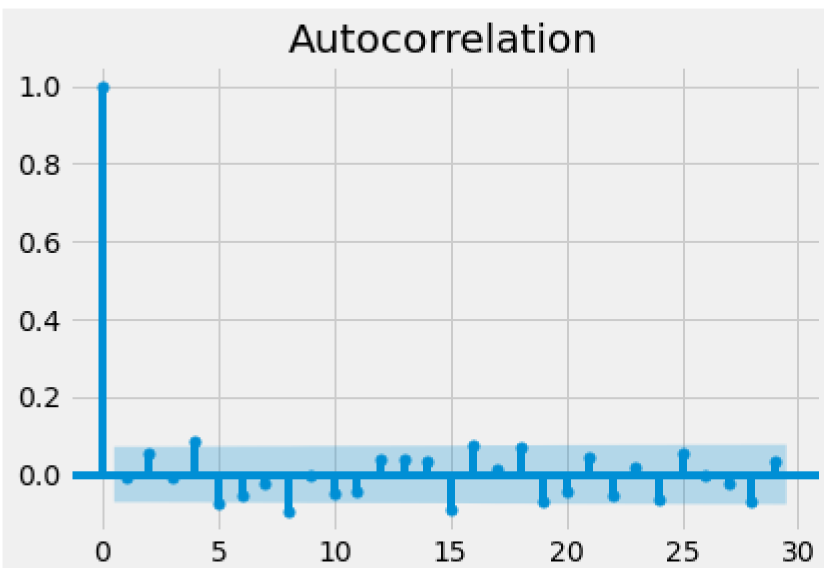
In [156…

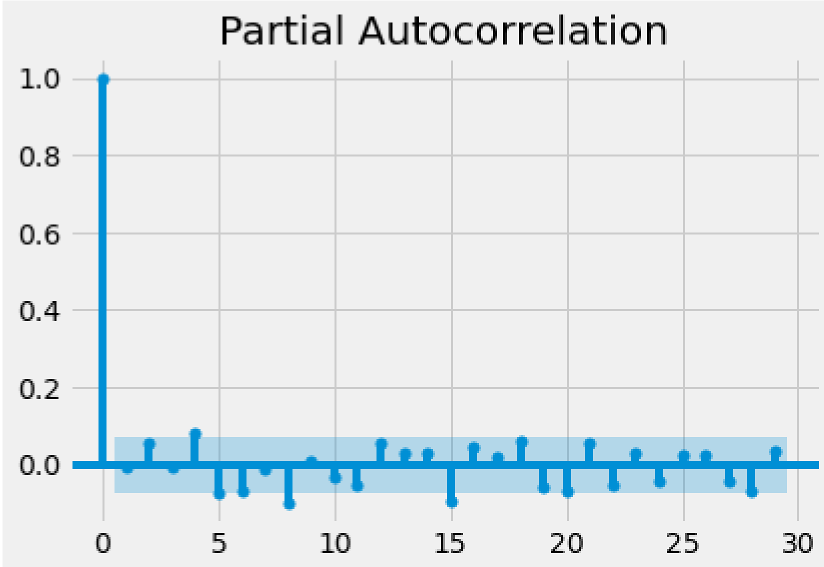
```
plt.figure(figsize=(10,6))
plt.hist(simple_model.resid,density=True)
plt.title('histogram of residuals')
mu, std = norm.fit(simple_model.resid)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, mu, std)
plt.plot(x, p, 'k', linewidth=2)
plt.show()
```



In [157…

```
plot_acf(simple_model.resid)
plot_pacf(simple_model.resid)
plt.show()
```





In [158…

```
acorr_ljungbox(simple_model.resid,return_df=True,lags=10)
```

Out[158…

	lb_stat	lb_pvalue
1	0.049345	0.824207
2	2.244913	0.325479
3	2.315222	0.509611
4	7.619350	0.106560
5	11.890964	0.036313
6	14.272591	0.026735
7	14.647647	0.040791
8	21.321387	0.006341
9	21.329868	0.011264
10	22.971521	0.010852

In [159…

```
pred=simple_model.forecast(84)
pred.index=test.index
```

In [162…

```
pred_simple_model=simple_model.forecast()
for i in range(83):
    simple_model=SimpleExpSmoothing(dataset[:train_size+i+1]['close']).fit()
    pred_simple_model=pred_simple_model.append(simple_model.forecast())
```

In [163…

```
pred_simple_model.index=test.index
```

In [164…

```
test['simple exp smoothing prediction']=pred_simple_model
test
```

Out[164…

	close	arima(1 step ahead)	upper_interval	lower_interval	prediction(without updating model)	simple exp smoothing prediction

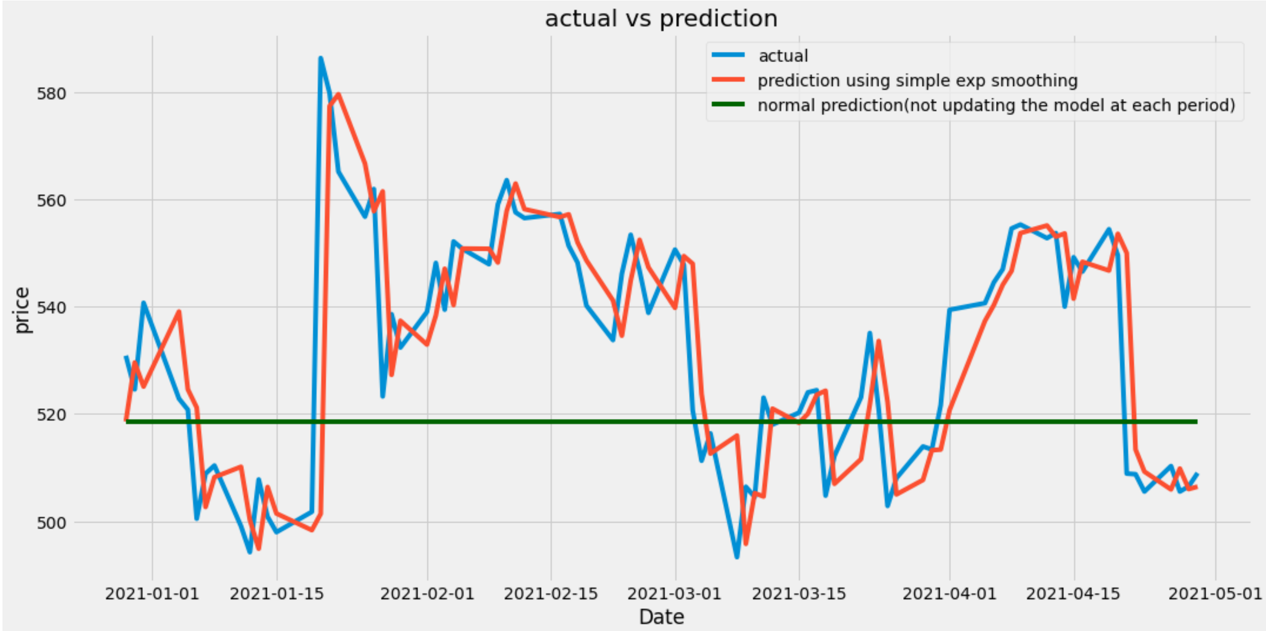
date	close	arima(1 step ahead)	upper_interval	lower_interval	prediction(without updating model)	simple exp smoothing prediction
date						
2020-12-29	530.869995	752 519.402626 dtype: float64	538.494883	500.310368	519.402626	518.606441
2020-12-30	524.590027	753 529.157691 dtype: float64	544.477397	493.356263	518.916830	529.608261
2020-12-31	540.729980	754 527.239778 dtype: float64	551.726140	489.186979	520.456559	525.109345
2021-01-04	522.859985	755 539.908122 dtype: float64	557.493992	485.804806	521.649399	539.099610
2021-01-05	520.799988	756 526.684286 dtype: float64	562.315842	480.662902	521.489372	524.598856
...	...	...	...	...	...	...
2021-04-23	505.549988	831 509.263792 dtype: float64	711.004728	395.455594	553.230161	509.262370
2021-04-26	510.299988	832 506.715746 dtype: float64	712.406306	394.900437	553.653372	505.945167
2021-04-27	505.549988	833 506.785502 dtype: float64	713.801891	394.351273	554.076582	509.835224
2021-04-28	506.519989	834 512.049115 dtype: float64	715.191593	393.807992	554.499793	506.008094
2021-04-29	509.000000	835 507.523816 dtype: float64	716.575516	393.270491	554.923003	506.465255

84 rows × 6 columns

In [165…
```
mean_squared_error(test['close'],pred_simple_model)
```

Out[165… 210.4021667310662

In [166…
```
plt.figure(figsize=(16,8))
plt.plot(test['close'],label='actual')
plt.plot(pred_simple_model,label='prediction using simple exp smoothing')
plt.plot(pred,label='normal prediction(not updating the model at each period)',color='d
plt.title('actual vs prediction')
plt.xlabel('Date')
plt.ylabel('price')
plt.legend()
plt.show()
```



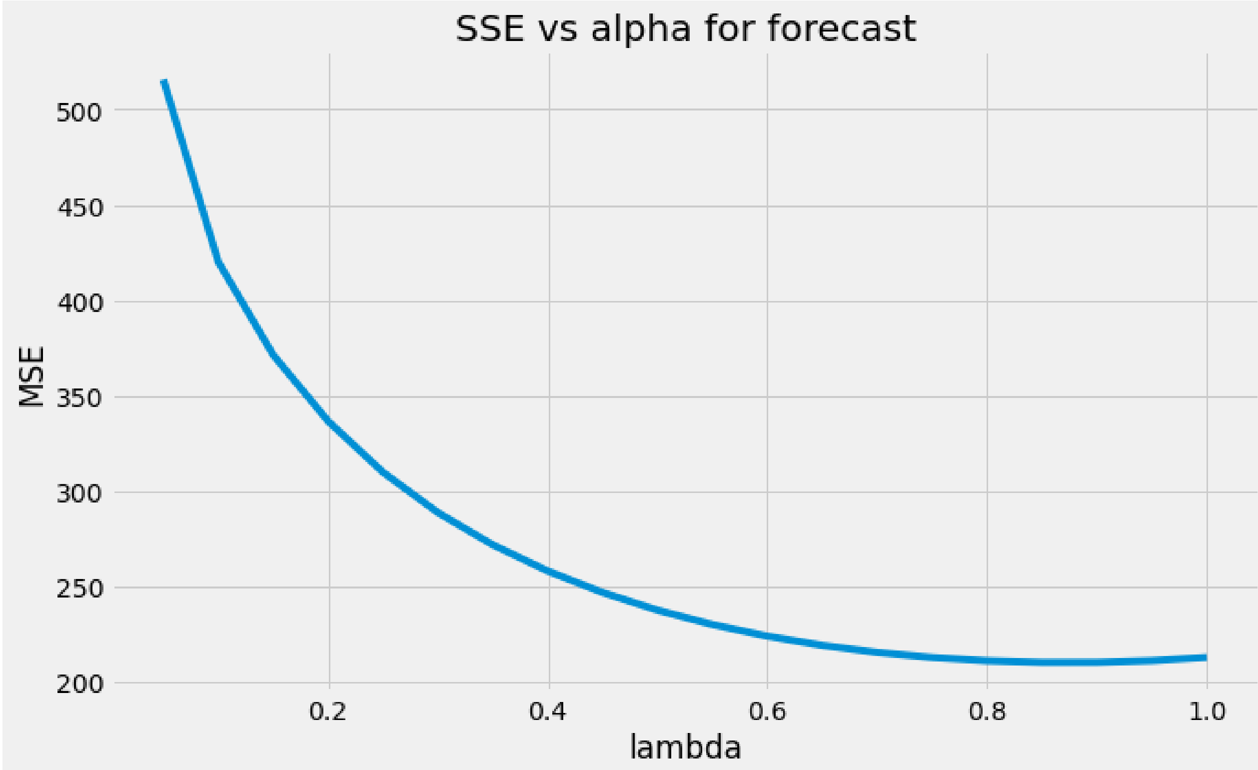
In [168…

```
mse=[]
for i in alpha:
    smodel=SimpleExpSmoothing(train['close']).fit(smoothing_level=i,optimized=True)
    pred_simple_model=smodel.forecast()
    for j in range(83):
        smodel=SimpleExpSmoothing(dataset[:train_size+j+1]['close']).fit(smoothing_leve
        pred_simple_model=pred_simple_model.append(smodel.forecast())
    mse.append(mean_squared_error(pred_simple_model,test['close']))
    print('simple exponential model with landa =',i,'  MSE will be ',mean_squared_error
```

```
simple exponential model with landa = 0.05   MSE will be  515.5250111950622
simple exponential model with landa = 0.1   MSE will be  420.0820990263477
simple exponential model with landa = 0.15   MSE will be  371.3731597541392
simple exponential model with landa = 0.2   MSE will be  336.6228319316936
simple exponential model with landa = 0.25   MSE will be  309.86983271605186
simple exponential model with landa = 0.3   MSE will be  288.7618963838183
simple exponential model with landa = 0.35   MSE will be  271.82909878496383
simple exponential model with landa = 0.4   MSE will be  258.07734949829694
simple exponential model with landa = 0.45   MSE will be  246.8274782332964
simple exponential model with landa = 0.5   MSE will be  237.6078249115878
simple exponential model with landa = 0.55   MSE will be  230.08269168454007
simple exponential model with landa = 0.6   MSE will be  224.00680391500347
simple exponential model with landa = 0.65   MSE will be  219.1968671662775
simple exponential model with landa = 0.7   MSE will be  215.51385272273794
simple exponential model with landa = 0.75   MSE will be  212.85204934239167
simple exponential model with landa = 0.8   MSE will be  211.13255905207743
simple exponential model with landa = 0.85   MSE will be  210.2998900819452
simple exponential model with landa = 0.9   MSE will be  210.3208560228981
simple exponential model with landa = 0.95   MSE will be  211.18532288706658
simple exponential model with landa = 1.0   MSE will be  212.9085792450042
```

In [171…

```
plt.figure(figsize=(10,6))
plt.plot(alpha,mse)
plt.title('SSE vs alpha for forecast')
plt.ylabel('MSE')
plt.xlabel('lambda')
plt.show()
```



double exponential model

In [175…

```
double_model=ExponentialSmoothing(train['close'],trend='additive',damped=False,seasonal
double_model.summary()
```

Out[175…

ExponentialSmoothing Model Results

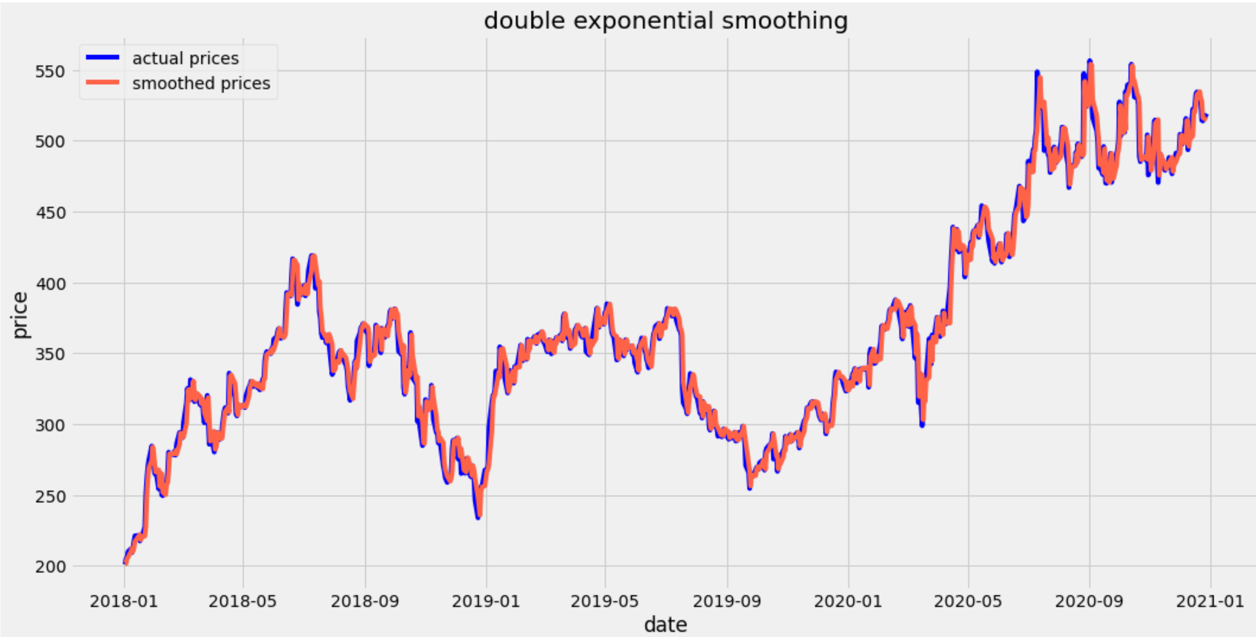
Dep. Variable:	close	**No. Observations:**	753
Model:	ExponentialSmoothing	**SSE**	72747.238
Optimized:	True	**AIC**	3449.723
Trend:	Additive	**BIC**	3468.219
Seasonal:	None	**AICC**	3449.835
Seasonal Periods:	None	**Date:**	Fri, 09 Jul 2021
Box-Cox:	False	**Time:**	17:47:22
Box-Cox Coeff.:	None		

	coeff	code	optimized
smoothing_level	0.8941686	alpha	True
smoothing_trend	1.1208e-13	beta	True
initial_level	201.03135	l.0	True
initial_trend	0.4217996	b.0	True

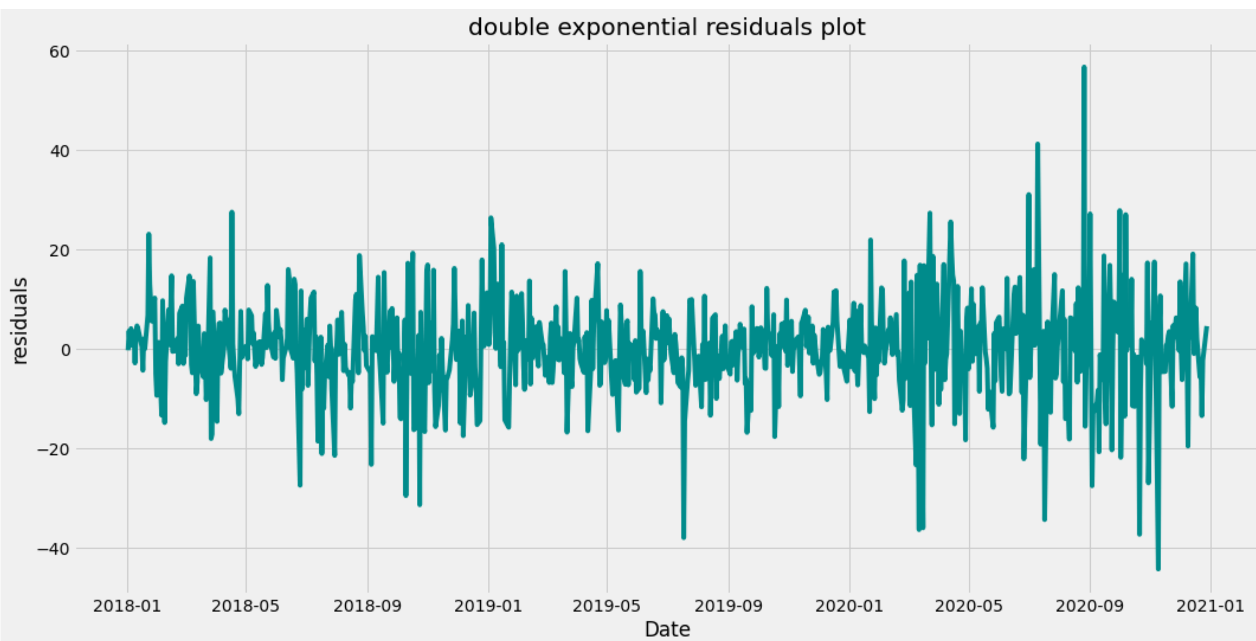
In [178…

```
train['double_exp_fitted']=double_model.fittedvalues
```

In [179…
```
plt.figure(figsize=(16,8))
plt.plot(train['close'],label='actual prices',color='blue')
plt.plot(train['double_exp_fitted'],label='smoothed prices',color='tomato')
plt.xlabel('date')
plt.ylabel('price')
plt.title('double exponential smoothing')
plt.legend()
plt.show()
```



In [180…
```
plt.figure(figsize=(16,8))
plt.plot(double_model.resid,color='darkcyan')
plt.xlabel('Date')
plt.ylabel('residuals')
plt.title('double exponential residuals plot')
plt.show()
```



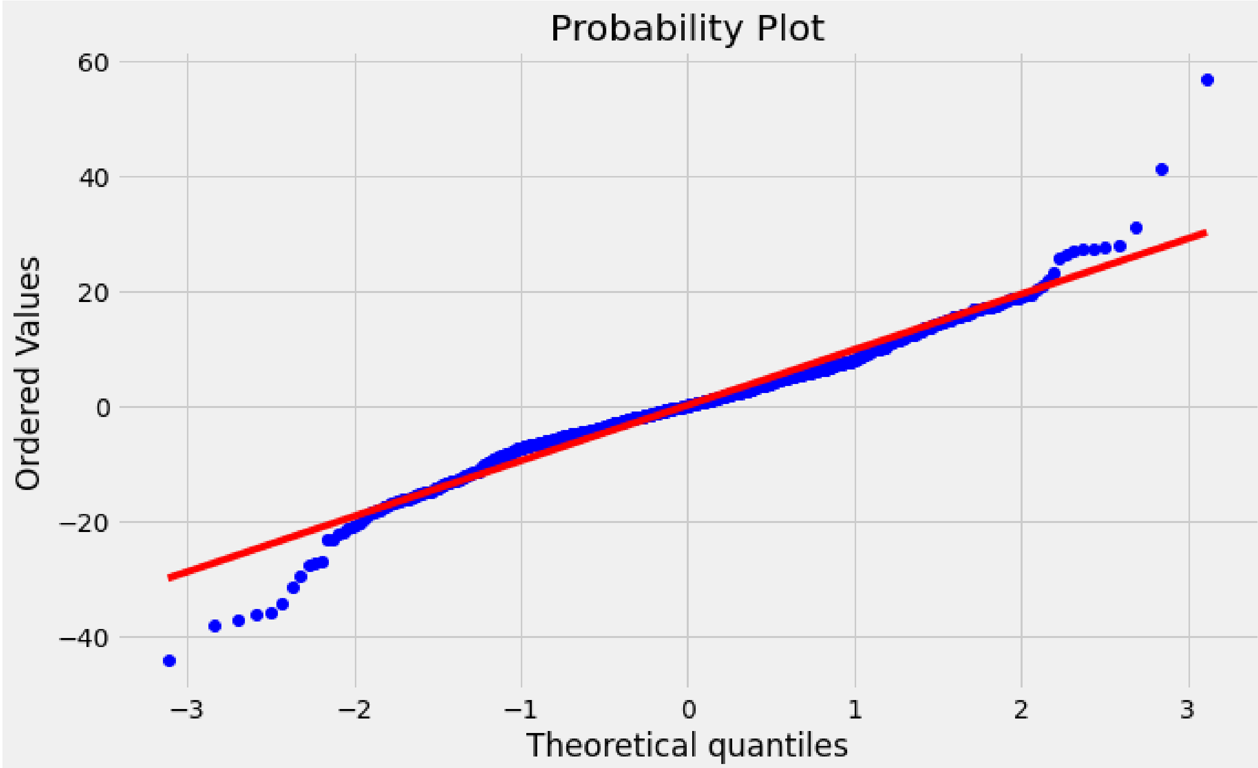
In [52]:
```
acorr_ljungbox(double_model.resid,lags=20,return_df=True,boxpierce=True)
```

Out[52]:

	lb_stat	lb_pvalue	bp_stat	bp_pvalue
1	0.024628	0.875297	0.024530	0.875543
2	2.238320	0.326554	2.226494	0.328491
3	2.303738	0.511805	2.291478	0.514155
4	7.585912	0.107980	7.531676	0.110321
5	11.853595	0.036850	11.759790	0.038232
6	14.252202	0.026942	14.132981	0.028185
7	14.638050	0.040929	14.514230	0.042756
8	21.319948	0.006344	21.107626	0.006867
9	21.330181	0.011262	21.117710	0.012138
10	22.982315	0.010812	22.743585	0.011734
11	24.392689	0.011175	24.129674	0.012197
12	25.596613	0.012235	25.311274	0.013415
13	26.896311	0.012854	26.585150	0.014172
14	27.876656	0.014774	27.544720	0.016341
15	33.736775	0.003707	33.272889	0.004302
16	38.255321	0.001391	37.683708	0.001679
17	38.420975	0.002150	37.845194	0.002581
18	42.268713	0.001014	41.591004	0.001261
19	46.083411	0.000482	45.299599	0.000622
20	47.484874	0.000502	46.660224	0.000654

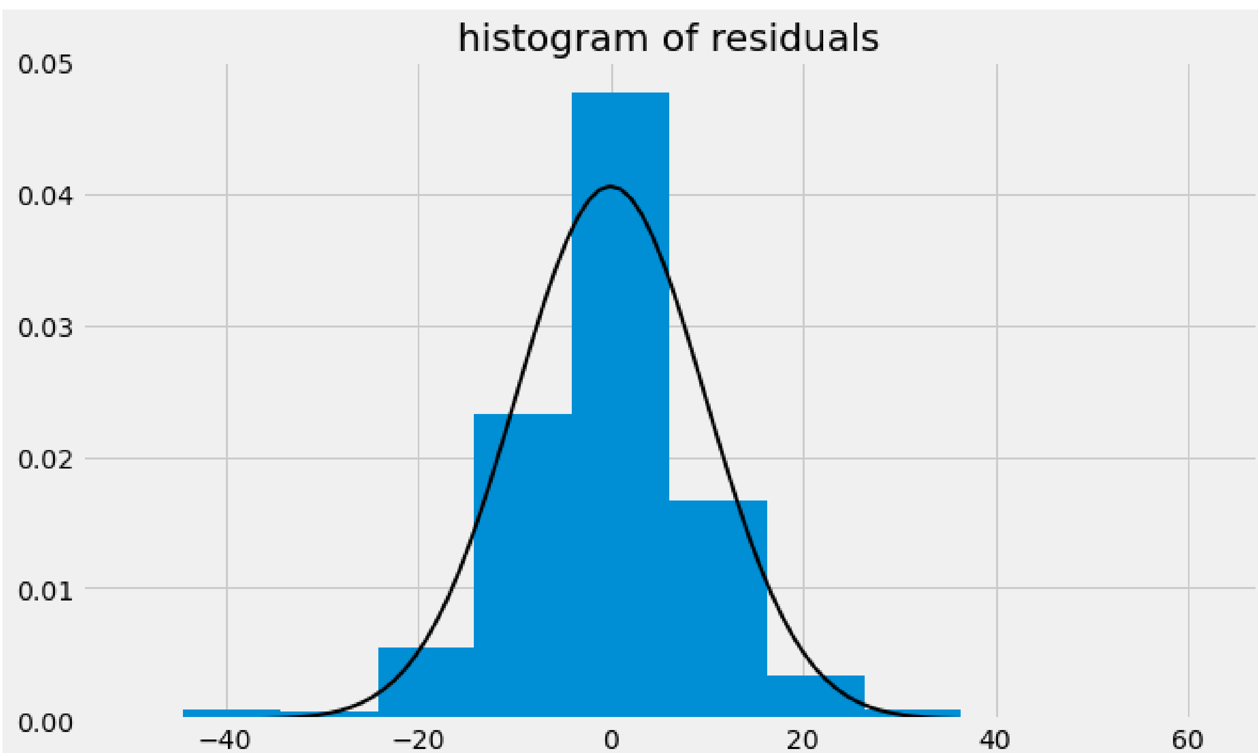
In [181…

```
plt.figure(figsize=(10,6))
stats.probplot(double_model.resid, dist="norm", plot=pylab)
plt.show()
```



In [184…

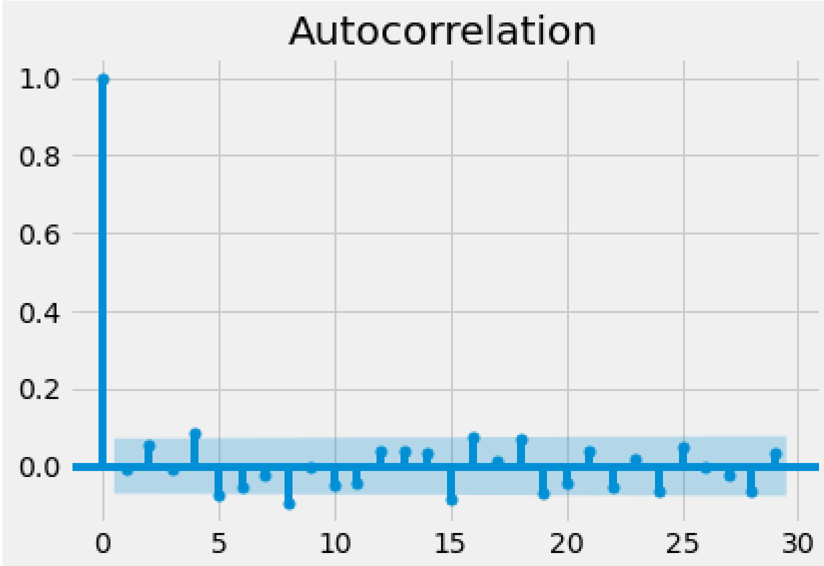
```
plt.figure(figsize=(10,6))
plt.hist(double_model.resid,density=True)
plt.title('histogram of residuals')
mu, std = norm.fit(double_model.resid)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, mu, std)
plt.plot(x, p, 'k', linewidth=2)
plt.show()
```

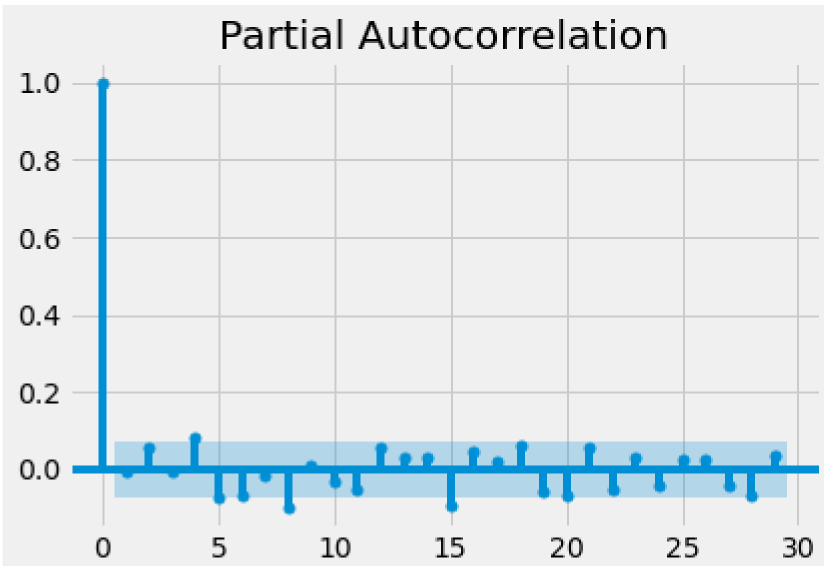


In [185…

```
plot_acf(double_model.resid)
```

```
plot_pacf(double_model.resid)
plt.show()
```





In [186…

```
mean_squared_error(train['close'],train['double_exp'])
```

Out[186…

```
96.60987757125862
```

In [188…

```
mse_double=[]
for i in alpha:
    dmodel=ExponentialSmoothing(train['close'],trend='additive',damped=False,seasonal=N
        smoothing_level=i)
    pred_double_model=dmodel.forecast()
    for j in range(83):
        dmodel=ExponentialSmoothing(dataset[:train_size+j+1]['close'],trend='additive',
            smoothing_level=i)
        pred_double_model=pred_double_model.append(dmodel.forecast())
    mse_double.append(mean_squared_error(pred_double_model,test['close']))
    print('double exponential model with landa =',i,'and beta= ',dmodel.params['smoothi
```

```
double exponential model with landa = 0.05 and beta=  0.044227655221363626   MSE will be
607.5110559926981
double exponential model with landa = 0.1 and beta=  0.019702540107021323   MSE will be
445.23460066873287
double exponential model with landa = 0.15 and beta=  8.426995741527164e-05   MSE will b
```

```
e  377.6576636489258
double exponential model with landa = 0.2 and beta=  0.0   MSE will be  341.655450246395
05
double exponential model with landa = 0.25 and beta=  7.056550550992811e-17   MSE will b
e  313.7264248140947
double exponential model with landa = 0.3 and beta=  0.0   MSE will be  291.562057164767
1
double exponential model with landa = 0.35 and beta=  0.0   MSE will be  274.16084222219
86
double exponential model with landa = 0.4 and beta=  0.0   MSE will be  259.948413382837
03
double exponential model with landa = 0.45 and beta=  0.0   MSE will be  248.37371542158
84
double exponential model with landa = 0.5 and beta=  0.0   MSE will be  238.891288559455
7
double exponential model with landa = 0.55 and beta=  0.0   MSE will be  231.20353263650
594
double exponential model with landa = 0.6 and beta=  0.003045025604773771   MSE will be
225.04698499381686
double exponential model with landa = 0.65 and beta=  0.0   MSE will be  220.21407341825
008
double exponential model with landa = 0.7 and beta=  0.0   MSE will be  216.285348015001
12
double exponential model with landa = 0.75 and beta=  0.00021911680003832722   MSE will
be  213.52664031251055
double exponential model with landa = 0.8 and beta=  1.1245406485645362e-05   MSE will b
e  211.7811896325025
double exponential model with landa = 0.85 and beta=  0.005558985266252279   MSE will be
211.1165786352363
double exponential model with landa = 0.9 and beta=  0.0   MSE will be  211.734885434066
77
double exponential model with landa = 0.95 and beta=  1.9880349739069687e-14   MSE will
be  213.37739240842504
double exponential model with landa = 1.0 and beta=  0.0   MSE will be  214.280000352774
4
```

In [189…

```
pred_double=double_model.forecast(84)
pred_double.index=test.index
```

In [193…

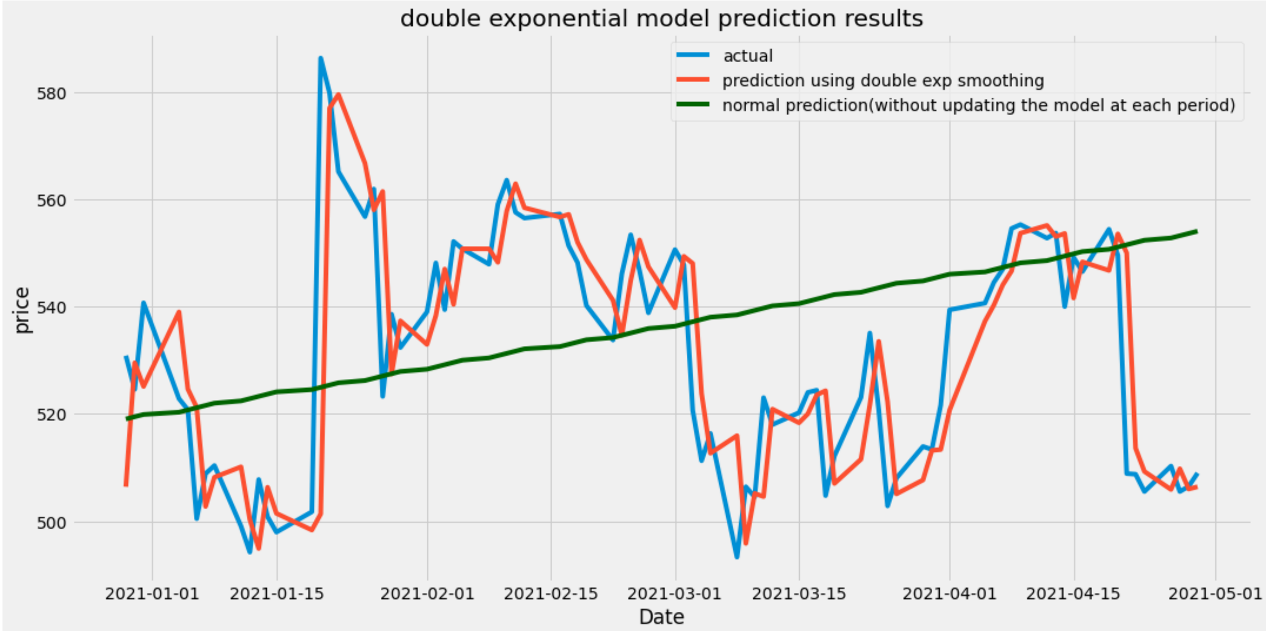
```
pred_double_1step=double_model.forecast()
for i in range(83):
    double_model=ExponentialSmoothing(dataset[:train_size+i+1]['close'],trend='additive
    pred_double_1step=pred_double_1step.append(double_model.forecast())
```

In [195…

```
pred_double_1step.index=test.index
```

In [196…

```
plt.figure(figsize=(16,8))
plt.plot(test['close'],label='actual')
plt.xlabel('Date')
plt.ylabel('price')
plt.plot(pred_double_1step,label='prediction using double exp smoothing')
plt.plot(pred_double,label='normal prediction(without updating the model at each period
plt.title('double exponential model prediction results')
plt.legend()
plt.show()
```



mean squared error of one step ahead prediction

In [265…

```
mean_squared_error(pred_double_1step,test['close'])
```

Out[265…

```
215.68763384098105
```

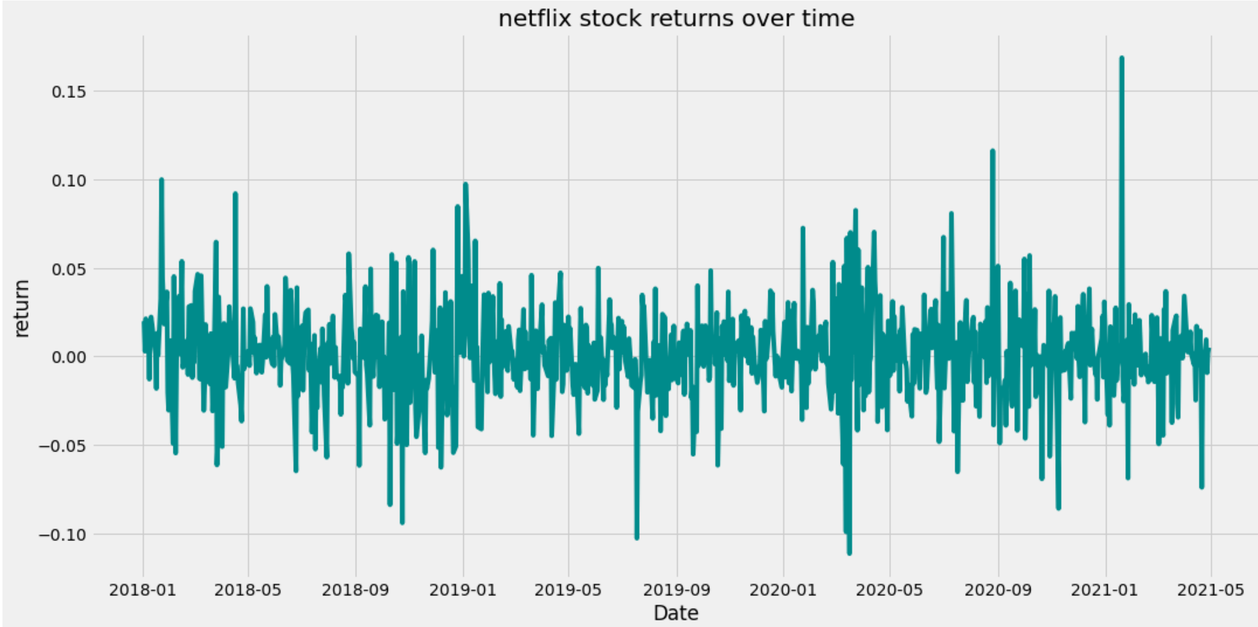
GARCH MODEL (PREDICTING VOLATILITY)

In [197…

```
from arch import arch_model
```

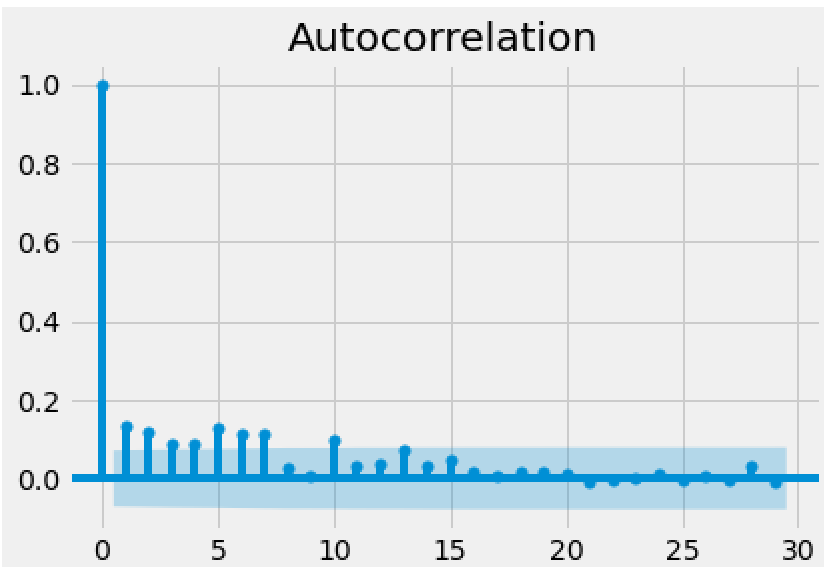
In [198…

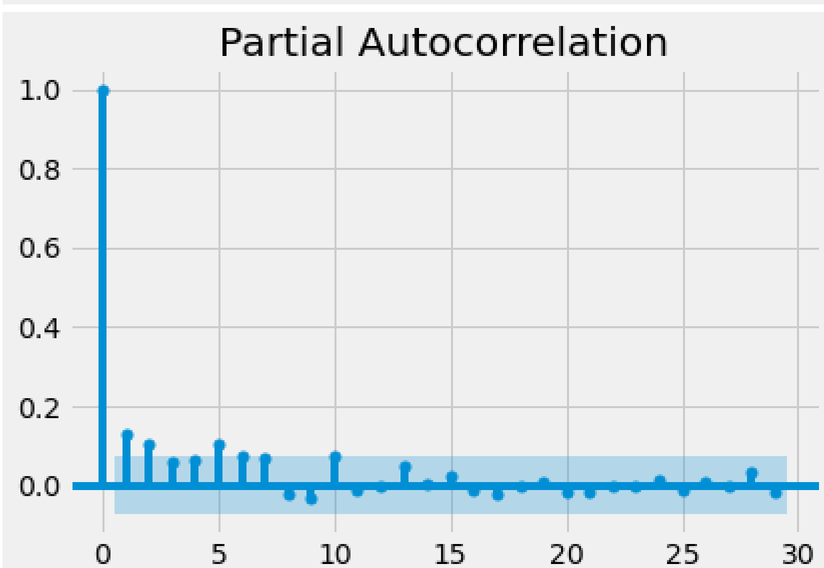
```
plt.figure(figsize=(16,8))
plt.plot(returns,color='darkcyan')
plt.title('netflix stock returns over time')
plt.xlabel('Date')
plt.ylabel('return')
plt.show()
```



In [21]:

```
plot_acf(train_returns**2)
plot_pacf(train_returns**2)
plt.show()
```





In [271…

```
gmodel=arch_model(train_returns,p=1,q=1,mean='constant')
gmodel=gmodel.fit(disp='off')
gmodel.summary()
```

Out[271…

Constant Mean - GARCH Model Results

Dep. Variable:	close	**R-squared:**	0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	0.000
Vol Model:	GARCH	**Log-Likelihood:**	1676.72
Distribution:	Normal	**AIC:**	-3345.45
Method:	Maximum Likelihood	**BIC:**	-3326.95
		No. Observations:	753
Date:	Fri, Jul 09 2021	**Df Residuals:**	752
Time:	19:35:28	**Df Model:**	1

Mean Model

	coef	**std err**	**t**	**P>|t|**	**95.0% Conf. Int.**
mu	1.8623e-03	8.775e-04	2.122	3.380e-02	[1.426e-04,3.582e-03]

Volatility Model

	coef	**std err**	**t**	**P>|t|**	**95.0% Conf. Int.**
omega	6.8092e-05	1.426e-05	4.776	1.792e-06	[4.015e-05,9.604e-05]
alpha[1]	0.0904	3.056e-02	2.958	3.101e-03	[3.049e-02, 0.150]
beta[1]	0.8196	3.292e-02	24.898	7.877e-137	[0.755, 0.884]

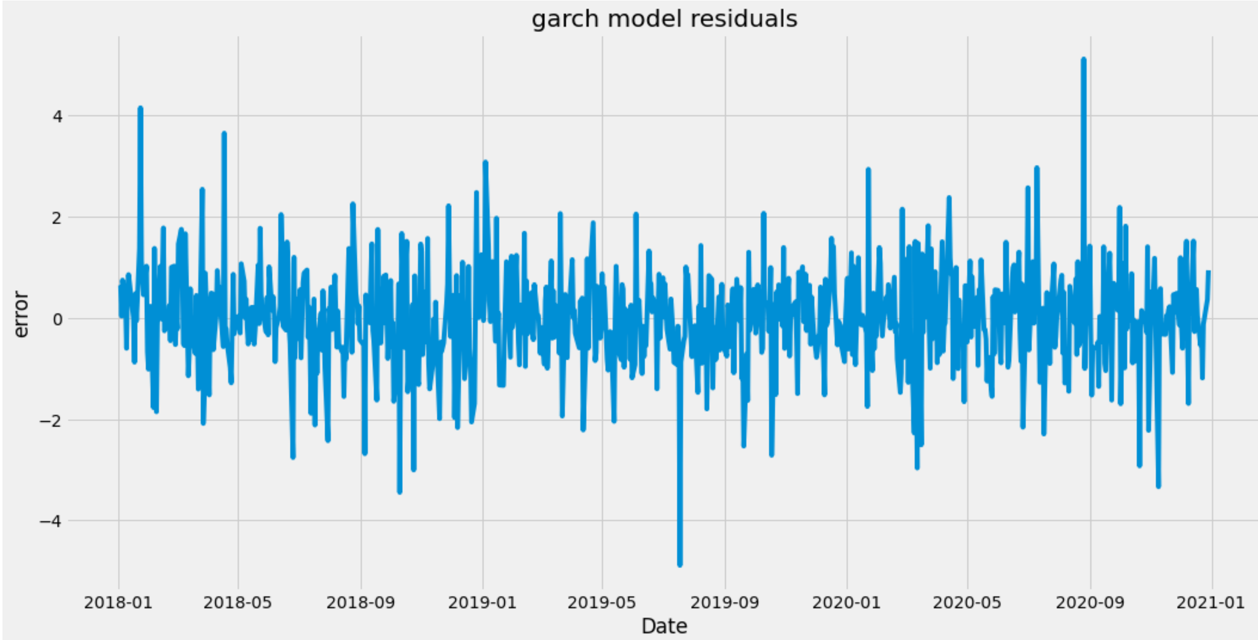
Covariance estimator: robust

In [272…

```
plt.figure(figsize=(16,8))
plt.title('garch model residuals')
plt.xlabel('Date')
plt.ylabel('error')
plt.plot(gmodel.std_resid)
```

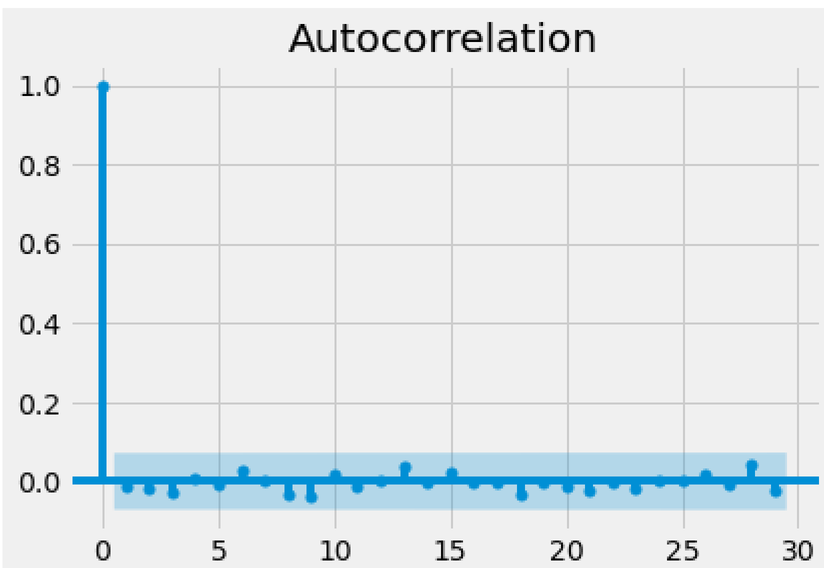
Out[272…

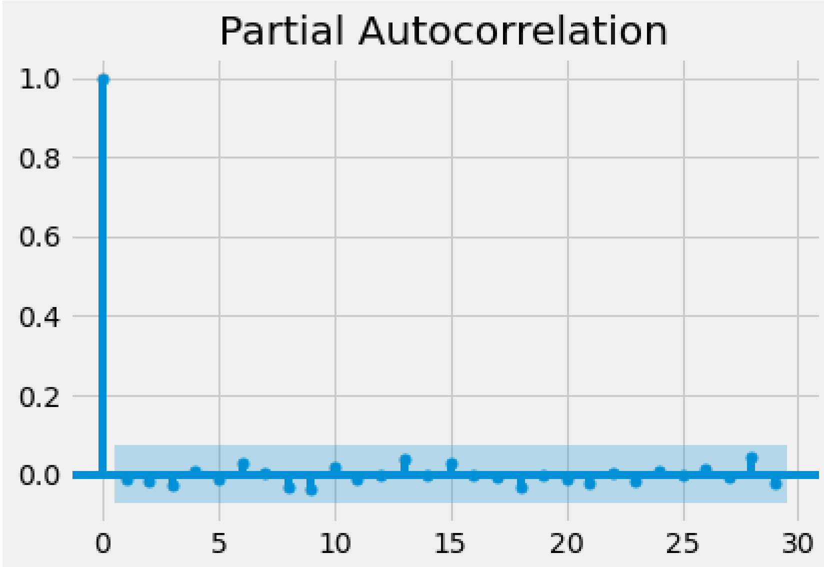
```
[<matplotlib.lines.Line2D at 0x25b9070e940>]
```



In [273…

```
plot_acf(gmodel.std_resid**2)
plot_pacf(gmodel.std_resid**2)
plt.show()
```





In [274...
```
acorr_ljungbox(gmodel.std_resid**2,lags=30,return_df=True)
```

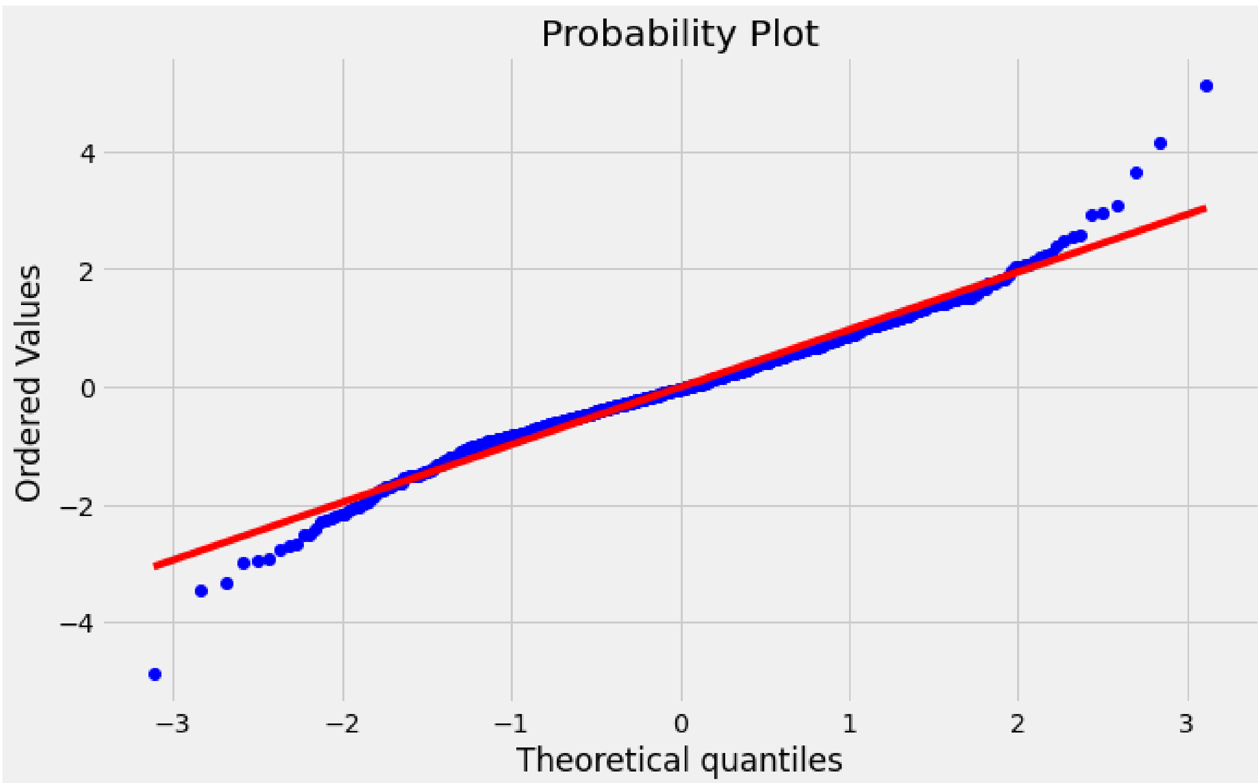
Out[274...

	lb_stat	lb_pvalue
1	0.089164	0.765242
2	0.338895	0.844131
3	0.894277	0.826809
4	0.958525	0.916018
5	1.030785	0.960045
6	1.682806	0.946441
7	1.694705	0.974794
8	2.589744	0.957415
9	3.698184	0.930132
10	3.978198	0.948325
11	4.081291	0.967478
12	4.081304	0.981924
13	5.026737	0.974600
14	5.027133	0.985431
15	5.412351	0.988022
16	5.422944	0.993217
17	5.429618	0.996271
18	6.360562	0.994502
19	6.362259	0.996915
20	6.457166	0.998122
21	6.854146	0.998411
22	6.855217	0.999136

	lb_stat	lb_pvalue
23	7.076043	0.999400
24	7.080089	0.999680
25	7.080090	0.999834
26	7.278277	0.999889
27	7.311909	0.999941
28	8.800810	0.999799
29	9.149508	0.999837
30	9.698585	0.999836

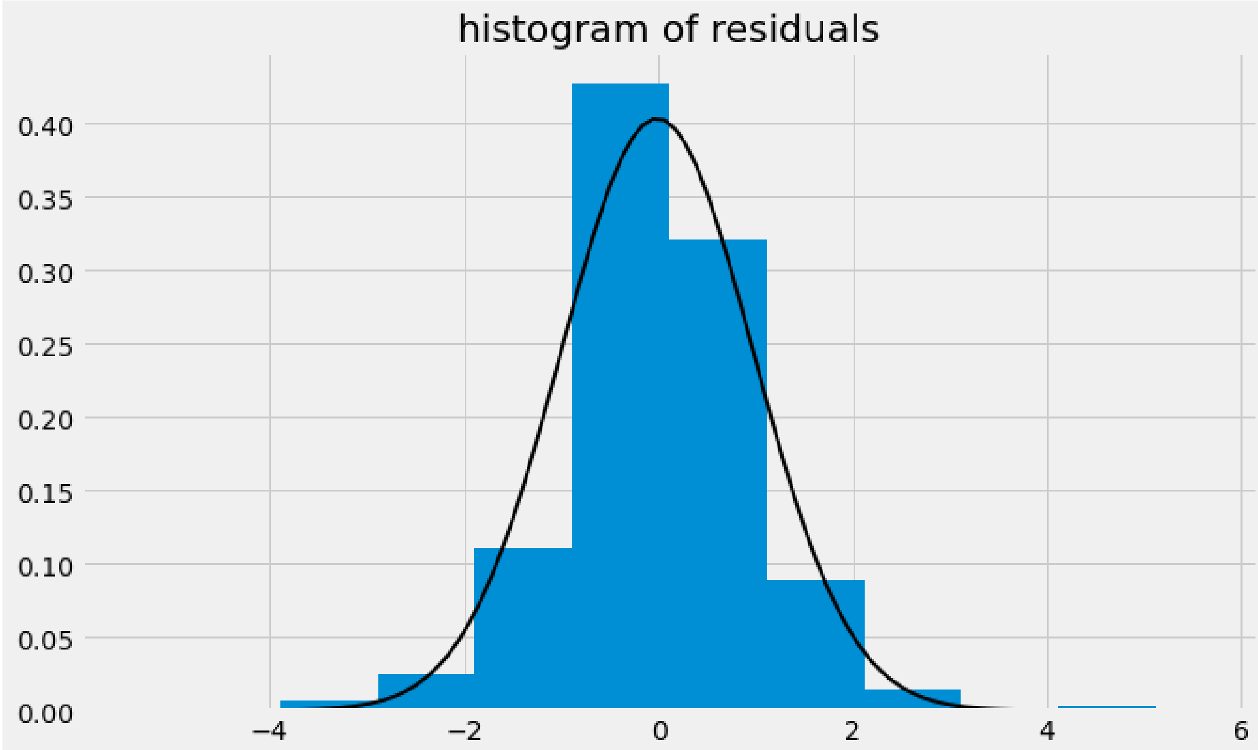
In [275...

```
plt.figure(figsize=(10,6))
stats.probplot(gmodel.std_resid, dist="norm", plot=pylab)
plt.show()
```



In [276...

```
plt.figure(figsize=(10,6))
plt.hist(gmodel.std_resid,density=True)
plt.title('histogram of residuals')
mu, std = norm.fit(gmodel.std_resid)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, mu, std)
plt.plot(x, p, 'k', linewidth=2)
plt.show()
```



predict future volatility

In [277…

```
garch_prediction=[]
test_size=len(test_returns)
for i in range(test_size):
    gtrain=returns[:-(test_size-i)]
    gmodel=arch_model(gtrain,p=1,q=1)
    gmodel_fit=gmodel.fit(disp='off')
    pred=gmodel_fit.forecast(horizon=1)
    garch_prediction.append(np.sqrt(pred.variance.values[-1,:][0]))
```

In [278…

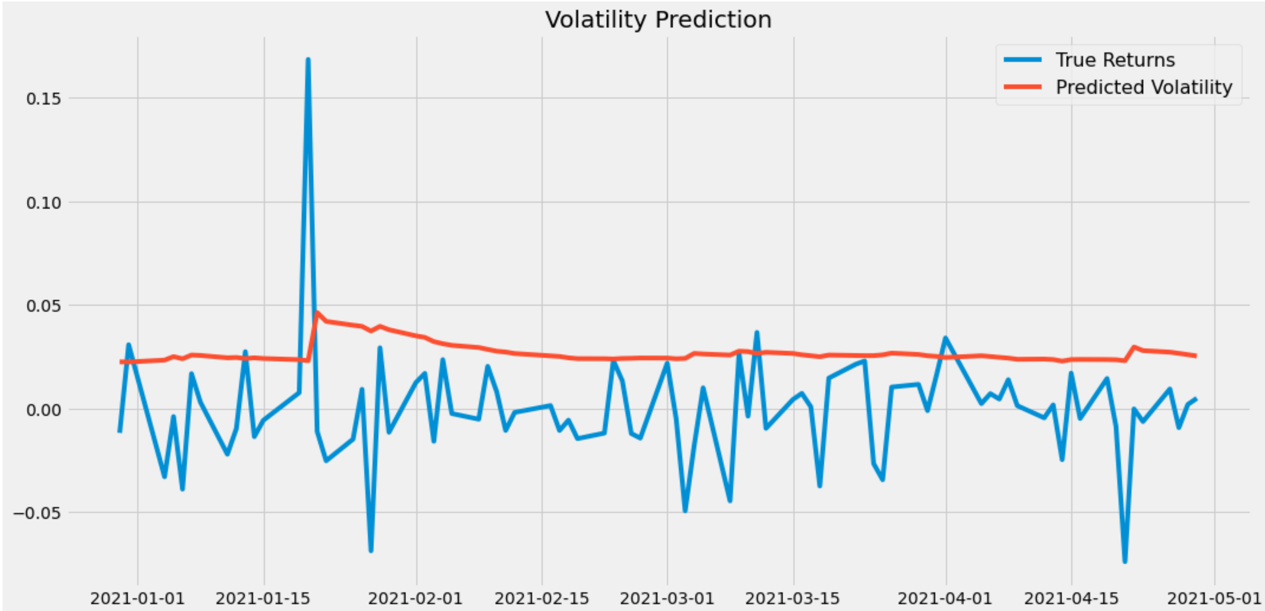
```
garch_prediction = pd.Series(garch_prediction, index=returns.index[-test_size:])
```

In [279…

```
plt.figure(figsize=(16,8))
true, = plt.plot(returns[-test_size:])
preds, = plt.plot(garch_prediction)
plt.title('Volatility Prediction', fontsize=20)
plt.legend(['True Returns', 'Predicted Volatility'], fontsize=16)
```

Out[279…

```
<matplotlib.legend.Legend at 0x25b8fbe7dc0>
```



RECURRENT NEURAL NETWORK

In [206…
```
from sklearn.preprocessing import MinMaxScaler
from statsmodels.tsa.seasonal import seasonal_decompose
from keras.preprocessing.sequence import TimeseriesGenerator
```

In [209…
```
dataset_test = np.array(dataset[int(dataset.shape[0]*0.9):])
```

In [210…
```
scaler=MinMaxScaler(feature_range=(0,1))
scaled_dataset=scaler.fit_transform(dataset)
```

In [211…
```
dataset_test=scaler.transform(dataset_test)
x_test, y_test = create_dataset(dataset_test)
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))
```

In [212…
```
scaled_dataset.shape
```

Out[212…
```
(837, 1)
```

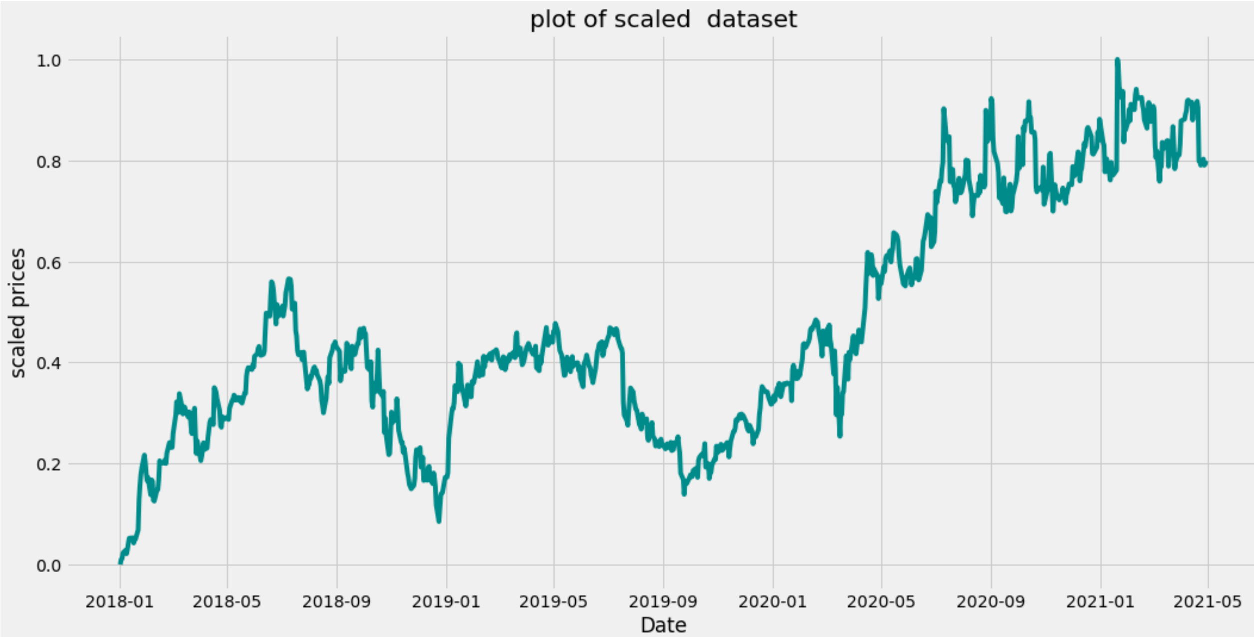
In [213…
```
test_size=84
train_rnn=dataset[:-test_size]
test_rnn=dataset[len(train_rnn):]
scaled_train=scaled_dataset[:-test_size]
scaled_test=scaled_dataset[len(scaled_train):]
```

In [214…
```
scaled_test[:10]
```

Out[214…
```
array([[0.85602297],
       [0.8397228 ],
       [0.88161537],
       [0.83523233],
       [0.82988544],
       [0.77716917],
       [0.79897213],
       [0.8028914 ],
       [0.77356136],
       [0.76097277]])
```

In [216...

```
plt.figure(figsize=(16,8))
plt.plot(RNN_dataset.index,scaled_dataset,color='darkcyan')
plt.title('plot of scaled  dataset')
plt.xlabel('Date')
plt.ylabel('scaled prices')
plt.show()
```



In [217...

```
#define generator
generator=TimeseriesGenerator(scaled_train,scaled_train,length=30,batch_size=1)
```

In [218...

```
x,y=generator[0]
print(f'given the array : \n{x.flatten()}')
print(f'predict this y : \n{y}')
```

```
given the array :
[0.         0.01033041 0.01183585 0.02315259 0.02849948 0.02138757
 0.02971941 0.04197056 0.05232691 0.05310559 0.0426454  0.0499909
 0.05032834 0.06880887 0.12775452 0.15633187 0.17813482 0.19085316
 0.216783   0.20175455 0.17969211 0.16611726 0.1722428  0.13805899
 0.16780437 0.16479349 0.12726139 0.12562616 0.14763673 0.14846725]
predict this y :
[[0.16853113]]
```

In [219...

```
del model
```

In [220...

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
```

In [221...

```
model=Sequential()
model.add(LSTM(units=50,return_sequences=True,activation='relu',input_shape=(30,1)))
model.add(LSTM(units=50,return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
```

In [222...

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

In [253…

```
LSTM_model=model.fit(generator,epochs=20,validation_data=(x_test,y_test))
```

```
723/723 [==============================] - 10s 14ms/step - loss: 9.6535e-04 - accuracy:
0.0000e+00 - val_loss: 7.6871e-04 - val_accuracy: 0.0000e+00
```

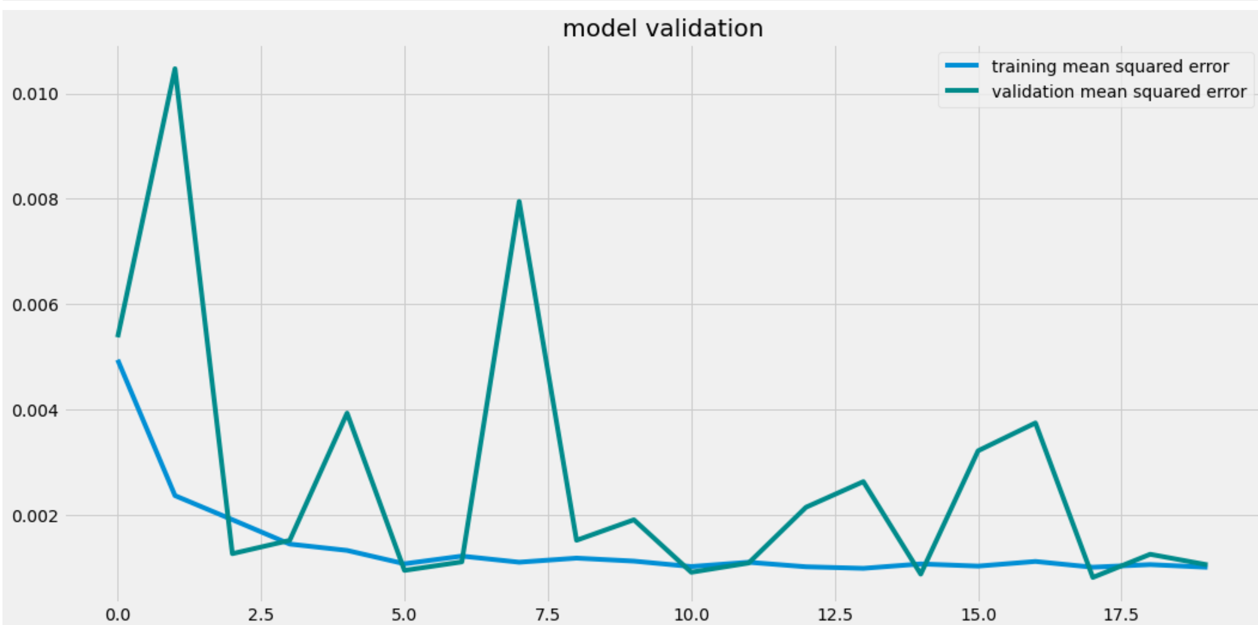
In [224…

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 30, 50)            10400
_________________________________________________________________
lstm_1 (LSTM)                (None, 50)                20200
_________________________________________________________________
dense (Dense)                (None, 25)                1275
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 26
=================================================================
Total params: 31,901
Trainable params: 31,901
Non-trainable params: 0
_________________________________________________________________
```

In [227…

```
plt.figure(figsize=(16,8))
plt.plot(LSTM_model.history['loss'],label='training mean squared error')
plt.plot(LSTM_model.history['val_loss'],label='validation mean squared error',color='da
plt.title('model validation')
plt.legend()
plt.show()
```



In [226…

```
fitted=[]
for i in range(len(generator)):
    fitted.append(model.predict(generator[i][0])[0])
```

In [228…

```
scaler.inverse_transform(fitted)
```

Out[228…

```
array([[264.03021046],
       [269.57678038],
       [281.63975388],
       [282.00633782],
       [280.8337216 ],
```

```
[281.65294663],
[279.13733366],
[284.40191858],
[291.94148944],
[290.36509927],
[289.95163456],
[288.8115926 ],
[296.48667333],
[309.32470083],
[319.95440656],
[317.75343323],
[313.20625148],
[324.09147633],
[317.92972695],
[312.29830549],
[315.80585485],
[316.62146308],
[314.84188847],
[310.7185511 ],
[313.02190891],
[312.97092909],
[305.55888816],
[299.56750783],
[313.79561953],
[301.87251904],
[288.22662214],
[292.03526245],
[282.79429753],
[282.79990072],
[287.41717972],
[292.40324719],
[289.1449478 ],
[288.74240242],
[294.68046062],
[300.05673039],
[305.12541043],
[307.62485683],
[304.81310162],
[327.42105561],
[330.12780846],
[328.29330425],
[323.36817143],
[315.16781486],
[304.945431  ],
[302.31626162],
[308.91629966],
[309.14628295],
[309.62940859],
[310.23140678],
[310.24782595],
[308.82594827],
[315.10317153],
[321.48699313],
[322.89347353],
[325.59212014],
[325.241875  ],
[322.40638662],
[323.55412147],
[321.99020069],
[323.50474913],
[321.50497385],
[320.30695957],
[326.59328801],
[327.68446282],
[338.90304288],
```

```
[344.58664977],
[346.76837937],
[345.31725718],
[348.04483827],
[346.86477027],
[353.84225417],
[356.7800122 ],
[360.60109894],
[362.52570195],
[357.50083699],
[355.81722845],
[356.50067953],
[358.94620673],
[373.42662261],
[386.62289817],
[387.82651564],
[386.00533047],
[397.52094835],
[409.22646605],
[410.15468916],
[406.46641403],
[382.95183447],
[391.82375832],
[386.44576626],
[390.41402853],
[387.8952695 ],
[393.39828842],
[387.66133643],
[393.40034369],
[402.53357355],
[412.85512694],
[411.66079837],
[413.55070295],
[409.36147072],
[393.36991654],
[394.87165113],
[377.76303036],
[371.5122615 ],
[361.94540463],
[358.23152478],
[359.63259719],
[355.59484621],
[359.66654929],
[360.46883846],
[353.53028982],
[335.28584097],
[333.56482426],
[334.8840534 ],
[340.83498285],
[340.6522822 ],
[346.98494027],
[348.51630975],
[344.67325805],
[345.13146789],
[342.18925486],
[337.80781494],
[333.9138638 ],
[324.52181949],
[319.75065951],
[314.89309793],
[323.07613974],
[333.71126494],
[340.74241544],
[336.71092212],
[352.1496323 ],
```

```
[359.44796757],
[363.44698996],
[363.16544127],
[365.27941594],
[362.74672932],
[358.84514269],
[339.60979081],
[340.64927393],
[343.93665707],
[344.8487136 ],
[351.58436485],
[364.70992313],
[364.90569023],
[361.15784679],
[348.0038133 ],
[360.45324598],
[362.54842471],
[361.46849072],
[357.5180025 ],
[364.07453546],
[365.17594068],
[372.5442469 ],
[376.05954658],
[370.70184311],
[375.81848322],
[373.03675329],
[372.47224365],
[361.01403548],
[348.66860622],
[345.04537629],
[350.99042699],
[328.25855989],
[319.65235113],
[333.99058613],
[331.75159609],
[342.32216982],
[359.53503513],
[346.32834545],
[331.2952004 ],
[325.58245234],
[328.14743766],
[306.3028261 ],
[308.88806557],
[300.64356093],
[288.55106735],
[286.00697916],
[298.45597553],
[313.71066464],
[309.15722524],
[311.96931342],
[308.51868003],
[321.06161017],
[315.45838835],
[302.99918453],
[293.27511692],
[291.84067799],
[286.44097358],
[288.27345697],
[286.13169601],
[275.59416731],
[270.29768555],
[266.61404913],
[264.37411757],
[265.19954859],
[269.19977084],
```

```
[282.63420482],
[290.11153207],
[288.26524739],
[289.8220953 ],
[278.40963118],
[280.67075841],
[270.33902054],
[269.71367383],
[267.57824515],
[274.44738545],
[277.23545341],
[270.87429716],
[266.64127281],
[271.36868659],
[269.72603988],
[265.51822413],
[258.78725751],
[252.64600914],
[259.16406038],
[262.35085024],
[263.22478672],
[271.40201867],
[272.57414117],
[274.65147696],
[295.92607902],
[313.39657614],
[318.65389299],
[317.49566364],
[319.9775886 ],
[330.92370219],
[328.65169008],
[346.46593356],
[346.68669684],
[347.75498815],
[335.77649877],
[322.26994671],
[317.59088338],
[321.43162721],
[332.69498683],
[332.76457888],
[326.69502949],
[335.17794517],
[335.73297073],
[335.8178682 ],
[345.51013083],
[350.92576069],
[348.37160284],
[341.17211561],
[342.21747748],
[341.2055625 ],
[353.57490865],
[348.78761654],
[353.88625297],
[352.81086582],
[356.82179498],
[355.74570744],
[352.91877311],
[357.65213453],
[359.3854484 ],
[360.54613489],
[358.84130772],
[354.37131906],
[352.98206157],
[347.63017944],
[349.69874303],
```

```
[354.97943712],
[349.9119512 ],
[346.24977453],
[353.63624518],
[352.82237073],
[356.73581821],
[355.20219826],
[357.00111994],
[358.94993836],
[355.20148638],
[368.66602602],
[373.24245227],
[359.29448846],
[360.85773181],
[356.08178386],
[349.89329305],
[350.1810305 ],
[352.37565432],
[361.94677098],
[364.11977431],
[365.80373879],
[364.09159763],
[361.4896749 ],
[357.4876098 ],
[359.78519219],
[359.76544325],
[363.06761513],
[349.59796603],
[345.08705574],
[353.90995169],
[351.77519471],
[356.1056663 ],
[371.37412226],
[377.52441266],
[371.32849302],
[364.57676706],
[368.94601316],
[367.42247436],
[366.19070318],
[373.26533578],
[374.77697929],
[380.03689106],
[375.24905931],
[367.15873417],
[360.5894907 ],
[358.450738  ],
[357.15519611],
[343.90340537],
[341.96570147],
[350.35449967],
[355.80330086],
[352.23077518],
[345.72058333],
[349.64011624],
[355.17969366],
[349.57663259],
[350.17662143],
[350.89247455],
[346.01852823],
[347.41506527],
[340.55816472],
[333.74359809],
[347.22457987],
[351.92564259],
[353.56183071],
```

```
[356.57377357],
[349.17158414],
[346.92714018],
[341.87243366],
[339.40869609],
[336.2073701 ],
[345.3185776 ],
[352.72437236],
[359.18247064],
[361.2035564 ],
[364.44790523],
[366.21962619],
[357.10519226],
[356.99318592],
[364.10569745],
[363.26885912],
[369.3428635 ],
[371.02101812],
[376.53198254],
[376.25641595],
[372.20983536],
[374.68694939],
[376.17876358],
[375.21284526],
[369.67357786],
[363.07943005],
[361.68669353],
[358.82516411],
[329.372756  ],
[314.89298311],
[309.49831927],
[307.02381165],
[315.34204183],
[324.14378805],
[333.00396587],
[330.96985959],
[324.02842899],
[319.81239791],
[316.2692086 ],
[315.24855587],
[306.79006228],
[306.91550249],
[302.97233975],
[311.13022463],
[307.62962183],
[307.96298852],
[309.03575778],
[299.28540801],
[294.40838454],
[298.92040858],
[305.33562181],
[298.83096426],
[296.09887076],
[294.89436918],
[290.62400556],
[292.44884198],
[290.06359497],
[290.07898077],
[293.95681126],
[292.53762035],
[288.52367144],
[289.28852948],
[290.96491588],
[288.9614549 ],
[291.65042222],
```

```
[287.3862933 ],
[286.53161203],
[286.97703098],
[291.29248419],
[292.33202471],
[295.52232804],
[290.79566058],
[285.63199044],
[274.59510063],
[268.84818805],
[263.19047868],
[266.31540383],
[266.81501181],
[266.93901103],
[270.20871199],
[272.3328941 ],
[271.32341904],
[270.83357072],
[274.11080382],
[276.01388967],
[273.23709698],
[269.99688165],
[279.21334411],
[283.29348641],
[285.73599387],
[284.58043981],
[285.4386001 ],
[290.81675291],
[278.59770538],
[276.82691446],
[269.78970724],
[270.8569135 ],
[272.01678477],
[276.73211955],
[281.74812044],
[282.00758936],
[289.80076185],
[287.76161499],
[286.11145335],
[290.15173035],
[287.19948211],
[286.7447513 ],
[287.46898624],
[289.24659743],
[291.63637981],
[290.40718058],
[283.18278902],
[286.53873083],
[291.87027844],
[298.64652983],
[299.86275447],
[301.59444938],
[306.73005766],
[306.71951724],
[310.43195037],
[308.76961786],
[310.94448126],
[310.52954687],
[306.89857811],
[303.3095414 ],
[301.36419053],
[300.14527912],
[303.4961574 ],
[300.74936702],
[293.05040385],
```

```
[295.63607963],
[296.22767518],
[296.40634567],
[300.6445943 ],
[310.39860681],
[316.43518006],
[326.58288537],
[331.84163747],
[328.89275344],
[327.94924787],
[327.31046155],
[324.47507651],
[319.51867837],
[319.02577011],
[324.59173303],
[322.65021712],
[330.54094019],
[327.66008666],
[334.01544453],
[331.98581621],
[325.87884717],
[333.01818052],
[334.39910212],
[334.86079098],
[334.43239975],
[335.19823376],
[334.02107068],
[323.95983587],
[342.04368681],
[348.69348759],
[340.63505928],
[343.40694919],
[339.36025378],
[342.56157978],
[341.05013146],
[351.94639046],
[363.34024235],
[365.44578927],
[362.88097618],
[361.88321844],
[365.3946487 ],
[368.39708451],
[374.39526215],
[376.54137477],
[375.86759149],
[381.98482537],
[381.61452128],
[381.29004162],
[376.09532431],
[365.56470774],
[356.65095518],
[372.32963794],
[369.01655821],
[365.96476154],
[375.68617681],
[366.6817284 ],
[377.83382801],
[370.20349241],
[365.38080148],
[345.68082137],
[357.38802692],
[348.20483913],
[320.55509581],
[330.20371557],
[306.28124005],
```

```
[315.34290298],
[314.83139397],
[328.17009153],
[330.84470643],
[353.96120708],
[354.52577414],
[340.84161941],
[355.14330739],
[352.8176861 ],
[364.13846691],
[369.91705242],
[360.83558315],
[364.09632819],
[357.73726165],
[372.56400732],
[368.72327498],
[366.87426907],
[366.15219275],
[388.1611683 ],
[405.97280537],
[419.55242851],
[431.63025964],
[419.98261092],
[429.30415611],
[427.89431149],
[417.44425222],
[420.14069429],
[419.62464992],
[416.90448616],
[401.57382104],
[406.25094392],
[414.12721122],
[411.94621646],
[422.07069381],
[420.84189645],
[428.34658516],
[431.33827387],
[430.90878037],
[434.75539147],
[428.16845433],
[432.29235431],
[436.15621128],
[447.11795179],
[447.61739327],
[446.30211393],
[443.06397683],
[432.99805738],
[425.52944493],
[412.0212165 ],
[414.6918242 ],
[410.30981018],
[415.12956174],
[421.17618169],
[423.3014431 ],
[419.01280023],
[411.46916476],
[414.50046617],
[415.08498884],
[427.35220312],
[429.84867569],
[422.75081512],
[414.79986927],
[419.5807889 ],
[429.53779065],
[441.09140227],
```

```
[444.69021011],
[448.20405158],
[460.35231294],
[460.70294847],
[453.68493332],
[458.95585627],
[441.2599342 ],
[441.48252312],
[448.3367369 ],
[473.52131889],
[472.03711723],
[485.54261295],
[486.91146703],
[495.00504155],
[500.06609759],
[530.3115519 ],
[519.17099164],
[517.75395933],
[515.66732317],
[518.92807969],
[491.01116308],
[494.76566604],
[485.28165138],
[484.20882471],
[474.080765  ],
[475.12918107],
[488.00277963],
[484.22526685],
[480.2751805 ],
[480.44194422],
[482.94466297],
[491.45319486],
[501.43603097],
[496.95795903],
[501.94759739],
[490.49824179],
[479.31308567],
[463.82158152],
[468.75447612],
[474.94958054],
[477.40324844],
[477.50493251],
[485.26617373],
[480.17299122],
[490.40080604],
[487.08688813],
[483.68710826],
[484.44932546],
[526.15903893],
[519.59501974],
[517.71340512],
[521.42719312],
[541.172458  ],
[542.81281407],
[521.66167732],
[510.42345167],
[501.16184221],
[494.5781889 ],
[477.80045472],
[476.74101599],
[472.15664424],
[487.84249173],
[480.57768373],
[468.14131749],
[465.56084303],
```

```
[479.69717957],
[485.1685313 ],
[468.99580357],
[468.08204771],
[475.95443412],
[483.84452568],
[487.44668624],
[493.29005287],
[514.1812155 ],
[499.11886039],
[511.2485899 ],
[500.695928  ],
[521.22607543],
[523.44955337],
[530.06501166],
[531.61137656],
[542.08377969],
[534.2837984 ],
[533.87269898],
[524.35933648],
[523.1540082 ],
[518.66133123],
[488.17739699],
[480.33192425],
[482.11670018],
[483.04563516],
[484.08934363],
[482.00277636],
[495.89762471],
[474.75914106],
[478.31209002],
[481.21888126],
[489.98738726],
[504.03462394],
[507.89806756],
[472.01332664],
[473.66417721],
[482.86176335],
[481.54402687],
[478.2803769 ],
[474.55320096],
[475.20369886],
[476.30791716],
[478.9292903 ],
[482.28385428],
[473.09049349],
[476.72863846],
[478.99921532],
[484.76241503],
[485.05860319],
[496.35722386],
[497.0956964 ],
[492.32933587],
[492.0370975 ],
[505.6821677 ],
[505.87568434],
[490.28653776],
[493.93446534],
[496.24832913],
[511.53834817],
[512.80535743],
[517.34972611],
[523.92823551],
[526.44668452],
[522.15427557],
```

```
       [520.13850595],
       [509.10282174],
       [507.19649798]])
```

In [232...

```
train2=train_rnn[30:]
train2
```

Out[232...

	close
date	
2018-02-14	266.000000
2018-02-15	280.269989
2018-02-16	278.519989
2018-02-20	278.549988
2018-02-21	281.040009
...	...
2020-12-21	528.909973
2020-12-22	527.330017
2020-12-23	514.479980
2020-12-24	513.969971
2020-12-28	519.119995

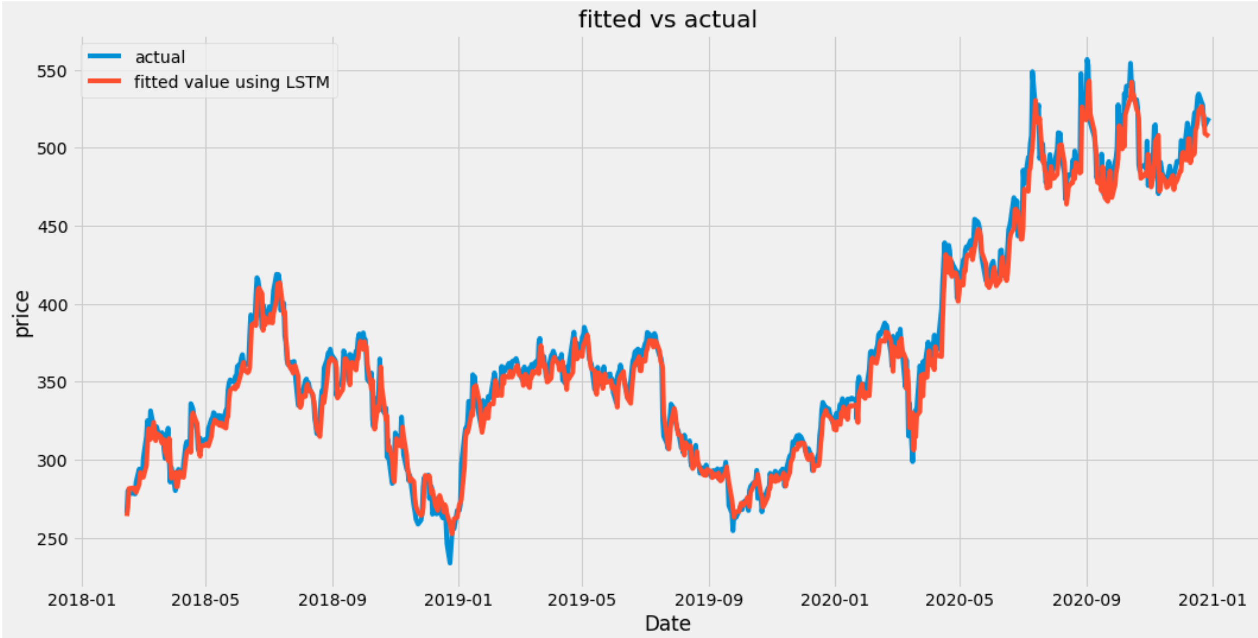
723 rows × 1 columns

In [233...

```
train2['fitted']=scaler.inverse_transform(fitted)
resid=train2['close']-train2['fitted']
```

In [234...

```
plt.figure(figsize=(16,8))
plt.plot(train2['close'],label='actual')
plt.plot(train2['fitted'],label='fitted value using LSTM')
plt.xlabel('Date')
plt.ylabel('price')
plt.title('fitted vs actual')
plt.legend()
plt.show()
```



In [235…

```
mean_squared_error(train2['fitted'],train2['close'])
```

Out[235…

```
115.5039161489512
```

In [236…

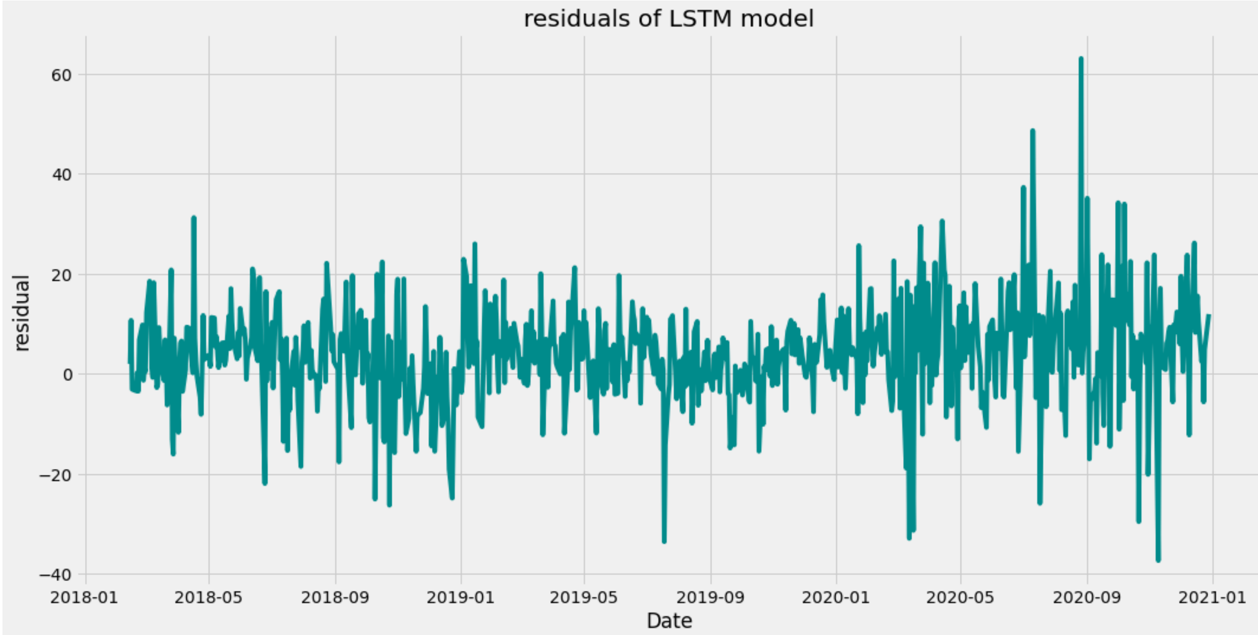
```
train2.head()
```

Out[236…

	close	fitted
date		
2018-02-14	266.000000	264.030210
2018-02-15	280.269989	269.576780
2018-02-16	278.519989	281.639754
2018-02-20	278.549988	282.006338
2018-02-21	281.040009	280.833722

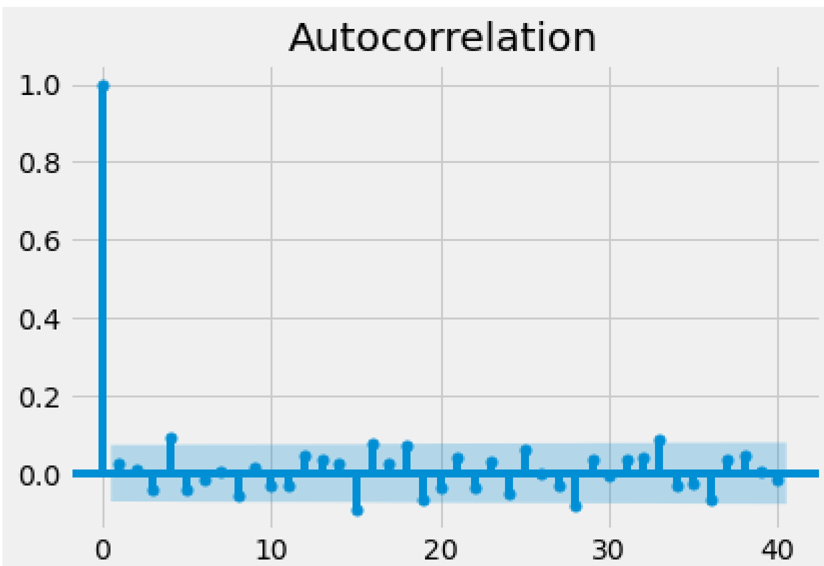
In [238…

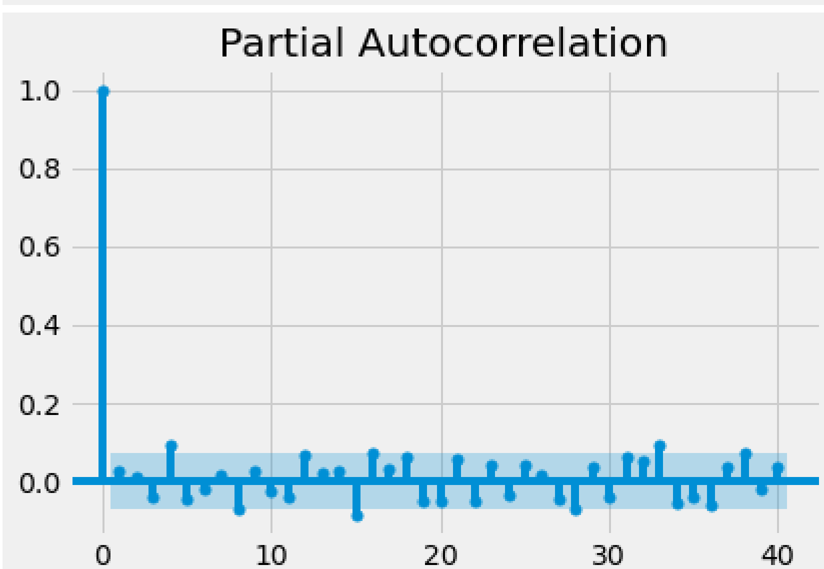
```
plt.figure(figsize=(16,8))
plt.plot(resid,color='darkcyan')
plt.title('residuals of LSTM model')
plt.xlabel('Date')
plt.ylabel('residual')
plt.show()
```



In [356…

```
plot_acf(resid,lags=40)
plot_pacf(resid,lags=40)
plt.show()
```





In [357...

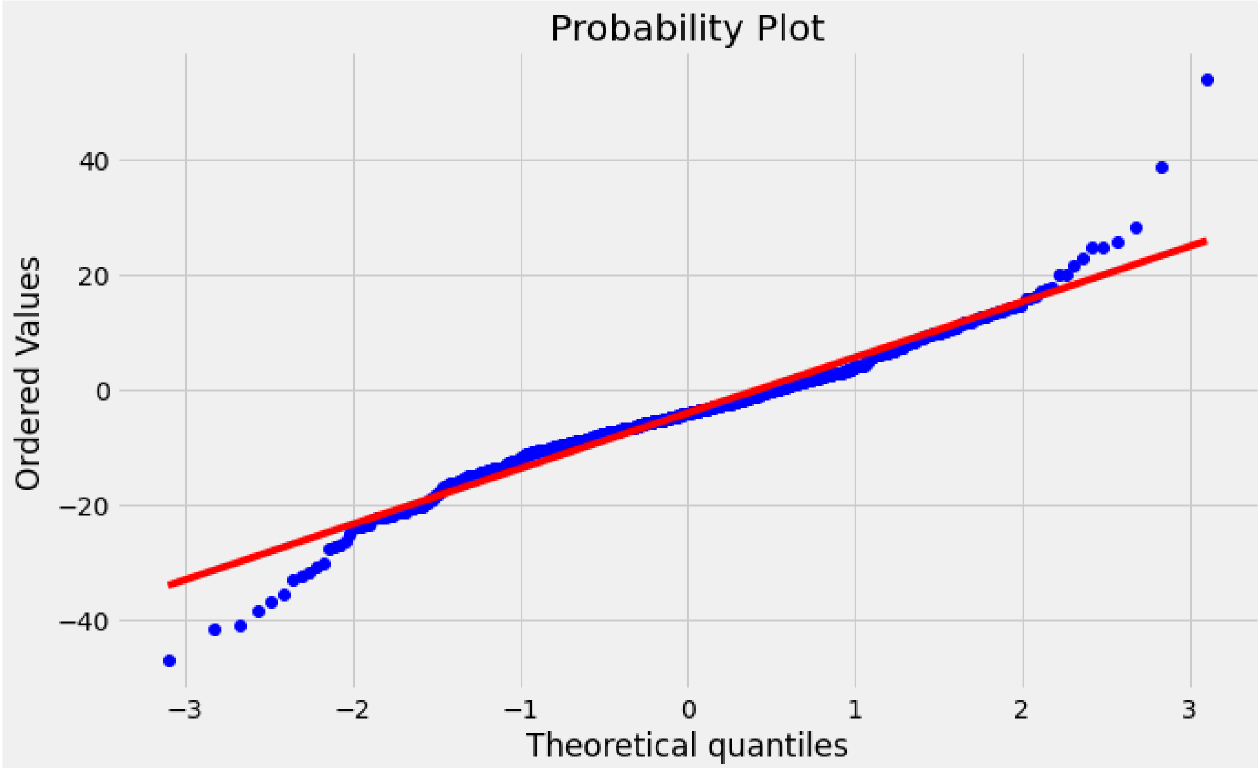
```
acorr_ljungbox(resid,return_df=True,lags=20,boxpierce=True)
```

Out[357...

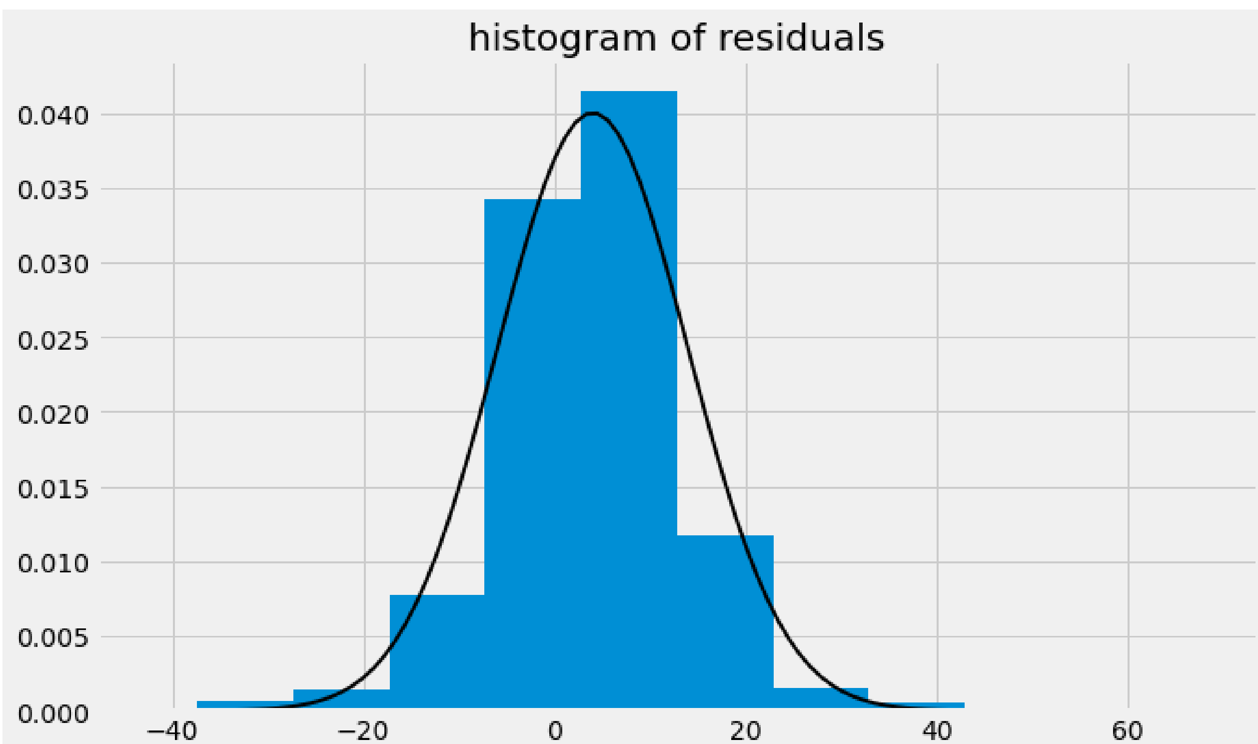
	lb_stat	lb_pvalue	bp_stat	bp_pvalue
1	0.377331	0.539035	0.375769	0.539876
2	0.478677	0.787148	0.476557	0.787983
3	1.741805	0.627680	1.730973	0.630070
4	7.707041	0.102919	7.646841	0.105406
5	9.132438	0.103897	9.058476	0.106754
6	9.382151	0.153199	9.305434	0.157115
7	9.399951	0.225202	9.323013	0.230289
8	11.732072	0.163564	11.622966	0.168835
9	11.872079	0.220617	11.760850	0.227130
10	12.496205	0.253218	12.374645	0.260765
11	13.249130	0.277355	13.114070	0.285935
12	14.827904	0.250983	14.662356	0.260422
13	15.820572	0.258955	15.634487	0.269432
14	16.276995	0.296753	16.080837	0.308463
15	22.534048	0.094546	22.191173	0.102888
16	27.087014	0.040526	26.631100	0.045770
17	27.599663	0.049839	27.130314	0.056184
18	31.651792	0.024168	31.070660	0.028250
19	34.823451	0.014666	34.150450	0.017641
20	35.927967	0.015684	35.221450	0.018956

In [358...

```
plt.figure(figsize=(10,6))
stats.probplot(resid, dist="norm", plot=pylab)
plt.show()
```



In [239…
```
plt.figure(figsize=(10,6))
plt.hist(resid,density=True)
plt.title('histogram of residuals')
mu, std = norm.fit(resid)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, mu, std)
plt.plot(x, p, 'k', linewidth=2)
plt.show()
```



In [254…
```
last_train_batch=scaled_train[-30:]
```

```
last_train_batch=last_train_batch.reshape((1,30,1))
```

In [255…

```
model.predict(last_train_batch)
```

Out[255…

```
array([[0.82432914]], dtype=float32)
```

In [256…

```
scaled_test[0]
```

Out[256…

```
array([0.85602297])
```

In [257…

```
test_prediction=[]
current_batch=last_train_batch
for i in range(len(test_rnn)):
    current_pred=model.predict(current_batch)[0]
    test_prediction.append(current_pred)
    #update current batch
    current_batch=np.append(current_batch[:,1:],scaled_test[i].reshape((1,1,1)),axis=1)
```

In [258…

```
test_prediction[:5]
```

Out[258…

```
[array([0.82432914], dtype=float32),
 array([0.85044825], dtype=float32),
 array([0.84130585], dtype=float32),
 array([0.87288827], dtype=float32),
 array([0.83907336], dtype=float32)]
```

In [259…

```
test_prediction=scaler.inverse_transform(test_prediction)
```

In [260…

```
test_rnn['prediction']=test_prediction
```

In [261…

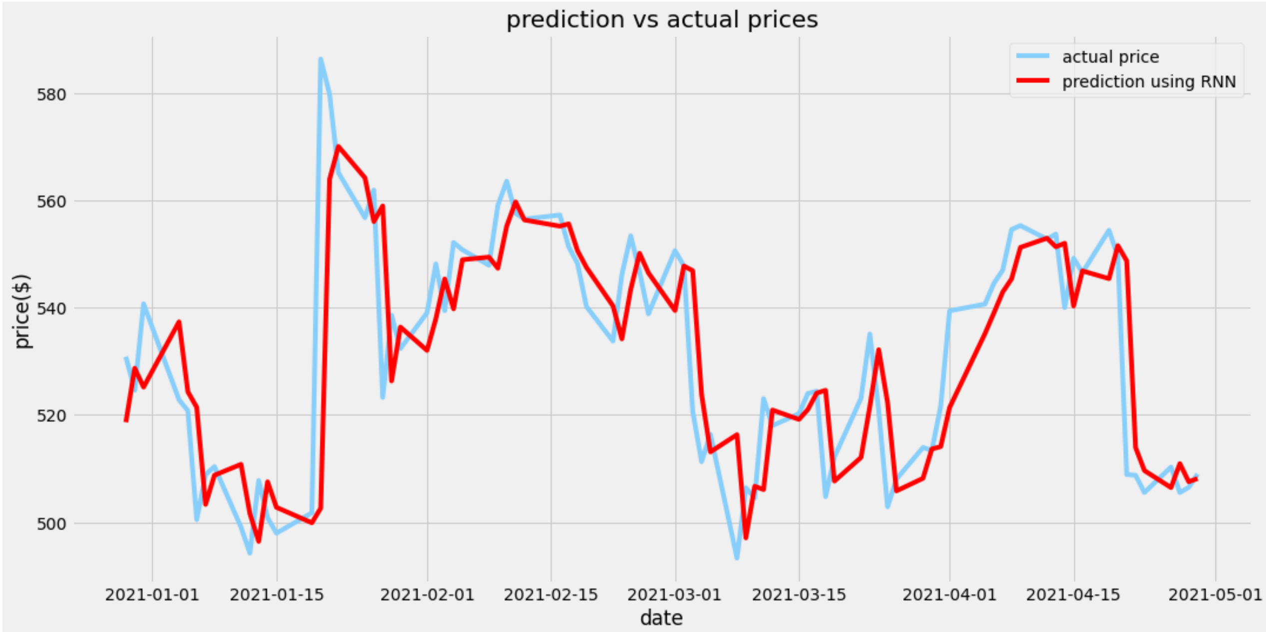
```
test_rnn.head()
```

Out[261…

date	close	prediction
2020-12-29	530.869995	518.659310
2020-12-30	524.590027	528.722222
2020-12-31	540.729980	525.199929
2021-01-04	522.859985	537.367687
2021-01-05	520.799988	524.339817

In [263…

```
plt.figure(figsize=(16,8))
plt.plot(test_rnn['close'],label='actual price',color='lightskyblue')
plt.plot(test_rnn['prediction'],label='prediction using RNN',color='red')
plt.title('prediction vs actual prices')
plt.xlabel('date')
plt.ylabel('price($)')
plt.legend()
plt.show()
```



mean squared error of one step ahead prediction

In [264…

```
mean_squared_error(test_rnn['close'],test_rnn['prediction'])
```

Out[264…

```
202.67323890317698
```