LARSON AND TUBRO STOCK PRICE PREDICTION

In [1]:

```
import pandas as pd
import numpy as np
import scipy
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
from arch import arch_model
from pmdarima.arima import auto_arima
import yfinance
import statsmodels.graphics.tsaplots as sgt
import warnings
warnings.filterwarnings("ignore")
sns.set()
```

Importing the data

In [95]:

```
tickers= " LT.NS ", interval= "1d",start = "2013-01-18", end = "2023-02-06", group_by= "tic
```

```
[*********************100%***********************]  1 of 1 completed
```

In [96]:

```
df = raw_data[["Close"]].copy()
```

In [97]:

```
df['Return'] = df.Close.div(df.Close[1])*100
```

In [98]:

```
start_date = "2013-01-18"
end_date = "2023-01-18"
```

In [99]:

```
df = df.asfreq('b')
df = df.fillna(method='bfill')
```

In [100]:

```
size = int(len(df)*0.9)
training_data = df[0:size]
testing_data = df[size:]
```

In [101]:

```
df.head()
```

Out[101]:

	Close	Return
Date		
2013-01-18	580.318481	98.076368
2013-01-21	591.700623	100.000000
2013-01-22	586.434387	99.109983
2013-01-23	589.341187	99.601245
2013-01-24	598.363831	101.126111

In [102]:

```
df.isnull().sum()
```

Out[102]:

```
Close     0
Return    0
dtype: int64
```

In [103]:

```
training_data.head()
```

Out[103]:

	Close	Return
Date		
2013-01-18	580.318481	98.076368
2013-01-21	591.700623	100.000000
2013-01-22	586.434387	99.109983
2013-01-23	589.341187	99.601245
2013-01-24	598.363831	101.126111

In [104]:

```
testing_data.tail()
```

Out[104]:

Date	Close	Return
2023-01-30	2112.899902	357.089349
2023-01-31	2124.399902	359.032900
2023-02-01	2145.550049	362.607367
2023-02-02	2144.899902	362.497490
2023-02-03	2166.550049	366.156459

Plotting the data

In [105]:

```
plt.figure(figsize=(20,10))
plt.grid(True)
plt.xlabel("Date")
plt.ylabel("Price")
plt.plot(df[0:size]["Close"],'green', label = "Train data")
plt.plot(df[size:]["Close"], 'blue', label = "Test data")
plt.legend()
plt.show()
```

AdFuller

In [106]:

```
from statsmodels.tsa.stattools import adfuller
```

In [107]:

```
result = adfuller(df.Close.dropna())
print("ADF Statistics:",result[0])
print("p-value =",result[1])
```

```
ADF Statistics: -0.13556733014594946
p-value = 0.9457786243823475
```

Decompose the ETS

In [108]:

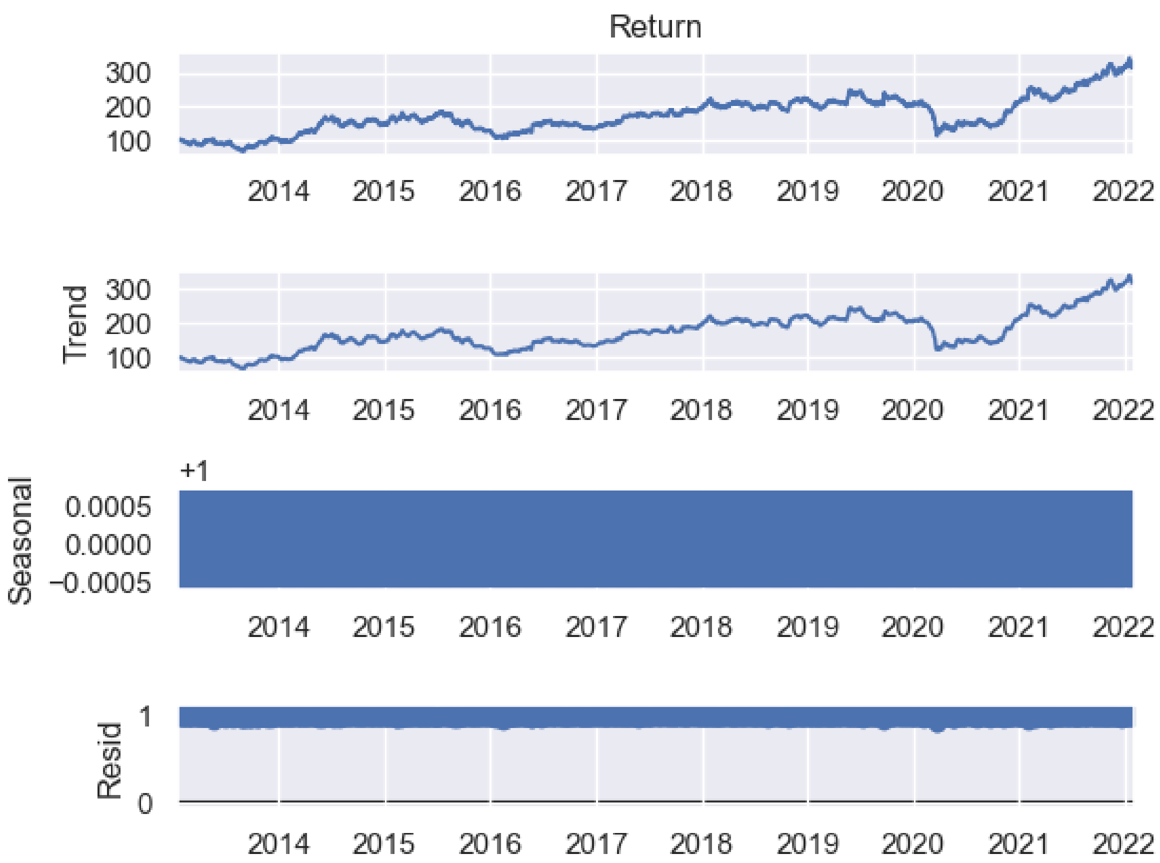
```
from statsmodels.tsa.seasonal import seasonal_decompose
```

In [109]:

```
decompose_data = seasonal_decompose(training_data.Return, model= 'multiplicative')
```

In [110]:

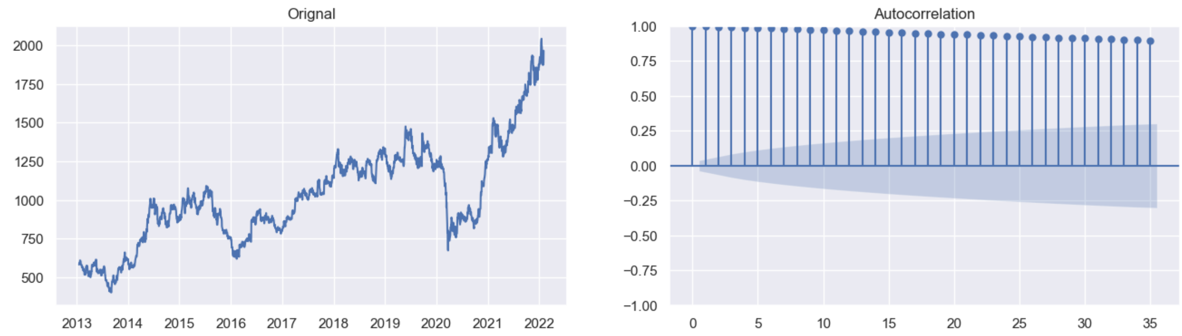
```
decompose_data.plot();
```



Autocorrelation ACF

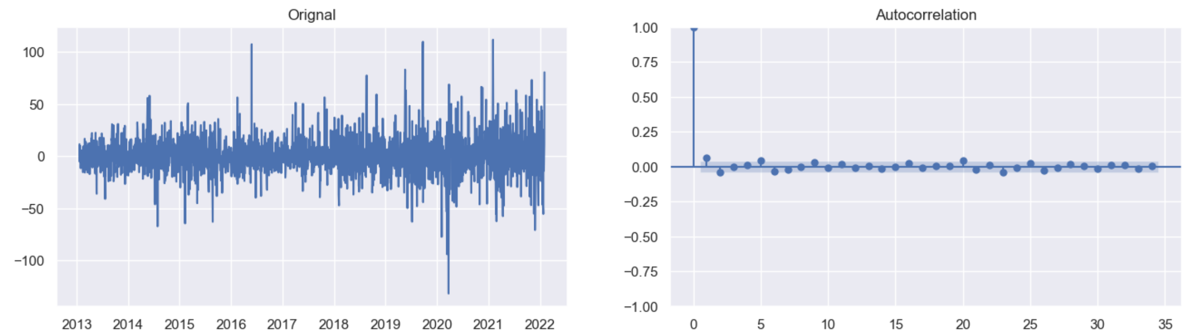
In [111]:

```
fig, (axis1, axis2) = plt.subplots(1, 2, figsize = (16,4))

axis1.plot(training_data.Close)
axis1.set_title("Orignal")

sgt.plot_acf(df.Return, ax= axis2);
```



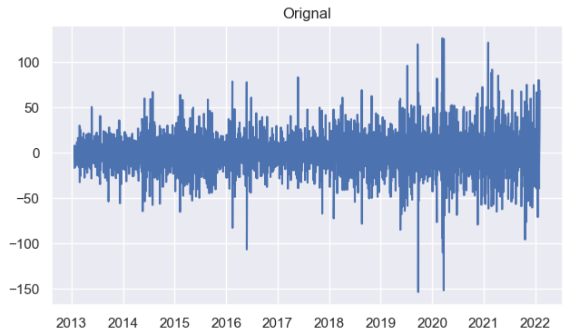
In [112]:

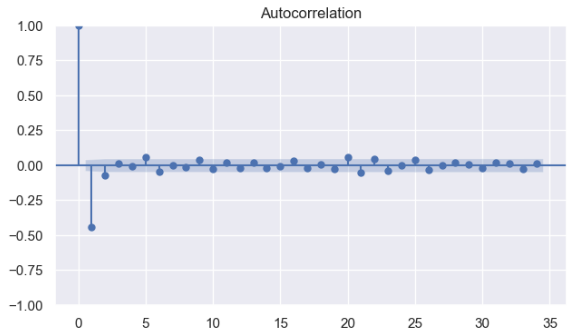
```
diff = training_data['Close'].diff().dropna()

fig, (axis1, axis2) = plt.subplots(1, 2, figsize = (16,4))

axis1.plot(diff)
axis1.set_title("Orignal")

sgt.plot_acf(diff, ax= axis2);
```



In [113]:

```
diff = training_data["Close"].diff().diff().dropna()


fig, (axis1, axis2) = plt.subplots(1, 2, figsize = (16,4))

axis1.plot(diff)
axis1.set_title("Orignal")

sgt.plot_acf(diff, ax= axis2);
```





In [114]:

```
from pmdarima.arima.utils import ndiffs
```

In [115]:

```
ndiffs(training_data['Close'], test='adf')
```

Out[115]:

1

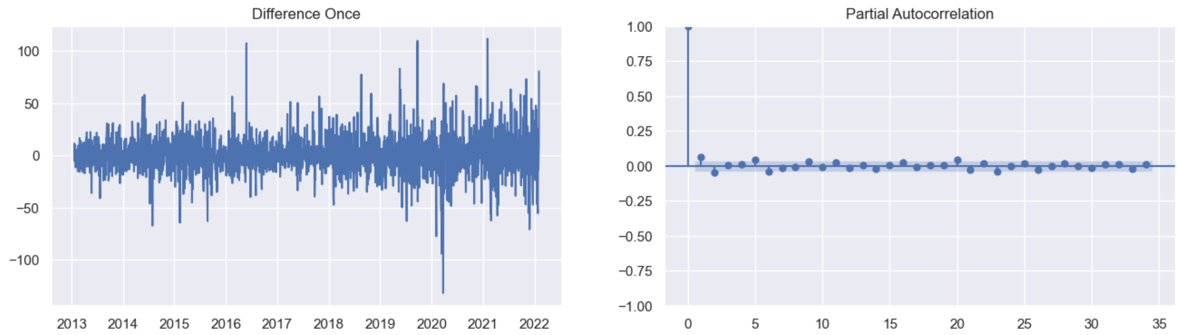
Plotting the Partial AutoCorrelation

In [116]:

```
from statsmodels.graphics.tsaplots import plot_pacf
```

In [117]:

```
diff = training_data["Close"].diff().dropna()

fig, (ax1,ax2) = plt.subplots(1, 2, figsize = (16, 4))

ax1.plot(diff)
ax1.set_title("Difference Once")
ax2.set_ylim(0, 1)
plot_pacf(diff, ax = ax2);
```



Fitting the ARIMA model

In [118]:

```
# ARIMA MODEL
model = ARIMA(training_data['Close'], order = (3, 1, 3))
result = model.fit()
```

In [119]:

```
print(result.summary())
```

```
                               SARIMAX Results                                
==============================================================================
==
Dep. Variable:                  Close   No. Observations:                   23
58
Model:                 ARIMA(3, 1, 3)   Log Likelihood              -10080.2
20
Date:                Tue, 07 Feb 2023   AIC                          20174.4
40
Time:                        00:14:49   BIC                          20214.7
96
Sample:                    01-18-2013   HQIC                         20189.1
35
                         - 02-01-2022                                         
Covariance Type:                  opg                                         
==============================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
ar.L1         -0.4897      0.091     -5.394      0.000      -0.668      -0.3
12
ar.L2         -0.7881      0.056    -14.154      0.000      -0.897      -0.6
79
ar.L3         -0.7216      0.079     -9.115      0.000      -0.877      -0.5
66
ma.L1          0.5449      0.087      6.288      0.000       0.375       0.7
15
ma.L2          0.7705      0.060     12.742      0.000       0.652       0.8
89
ma.L3          0.7673      0.074     10.313      0.000       0.621       0.9
13
sigma2       303.4678      4.518     67.175      0.000     294.614     312.3
22
==============================================================================
=======
Ljung-Box (L1) (Q):                   0.38   Jarque-Bera (JB):              
3467.29
Prob(Q):                              0.54   Prob(JB):                      
0.00
Heteroskedasticity (H):               2.65   Skew:                          
0.30
Prob(H) (two-sided):                  0.00   Kurtosis:                      
8.91
==============================================================================
=======

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

In [120]:

```
training_data.tail()
```

Out[120]:

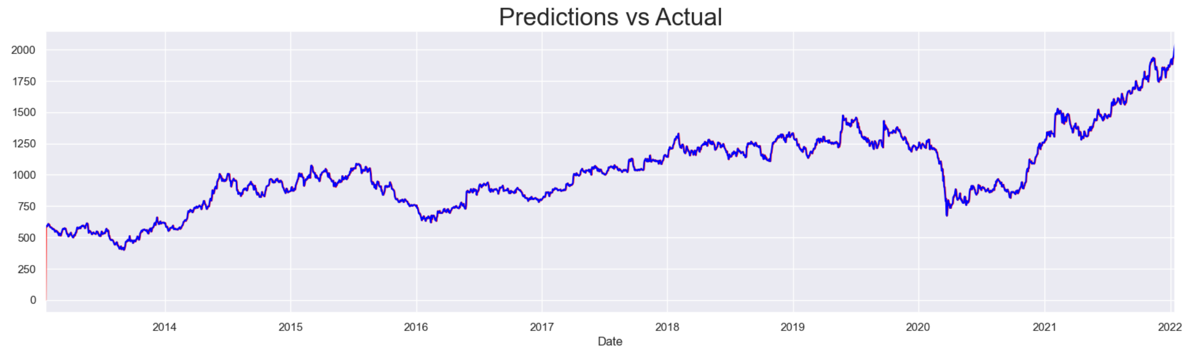
Date	Close	Return
2022-01-26	1886.689087	318.858729
2022-01-27	1886.689087	318.858729
2022-01-28	1873.557373	316.639412
2022-01-31	1885.059937	318.583396
2022-02-01	1965.332031	332.149732

In [121]:

```
end_date_train = '2022-01-17'
```

In [122]:

```
df_pred = result.predict(start = start_date, end = end_date_train)
df_pred[start_date:end_date_train].plot(figsize = (20,5), color = "red")
df.Close[start_date:end_date_train].plot(color = "blue")
plt.title("Predictions vs Actual", size = 24)
plt.show()
```



In [53]:

```
df_pred, training_data
```

Out[53]:

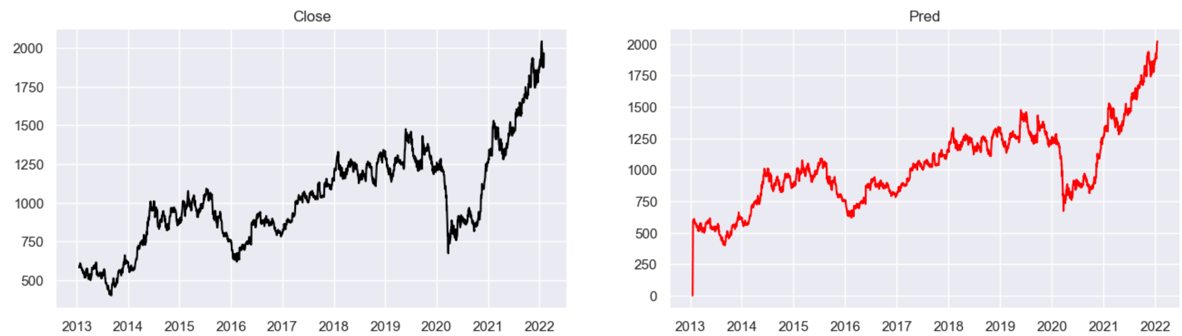
```
(Date
 2013-01-18       0.000000
 2013-01-21     580.330039
 2013-01-22     592.442470
 2013-01-23     585.568575
 2013-01-24     589.768374
                   ...
 2022-01-11    1930.762984
 2022-01-12    1934.968528
 2022-01-13    1950.378425
 2022-01-14    1994.992399
 2022-01-17    2019.964924
 Freq: B, Name: predicted_mean, Length: 2347, dtype: float64,
                   Close      Return
 Date
 2013-01-18   580.318542   98.076379
 2013-01-21   591.700623  100.000000
 2013-01-22   586.434204   99.109952
 2013-01-23   589.341187   99.601245
 2013-01-24   598.363892  101.126122
 ...                 ...         ...
 2022-01-11  1936.353149  327.252174
 2022-01-12  1949.238037  329.429776
 2022-01-13  1992.879272  336.805336
 2022-01-14  2018.896118  341.202297
 2022-01-17  2043.234253  345.315549

 [2347 rows x 2 columns])
```

In [123]:

```
fig, (axis1, axis2) = plt.subplots(1, 2, figsize = (16,4))

axis1.plot(training_data.Close, color = "Black")
axis1.set_title("Close")

axis2.plot(df_pred, color = "Red")
axis2.set_title("Pred")
```

Out[123]:

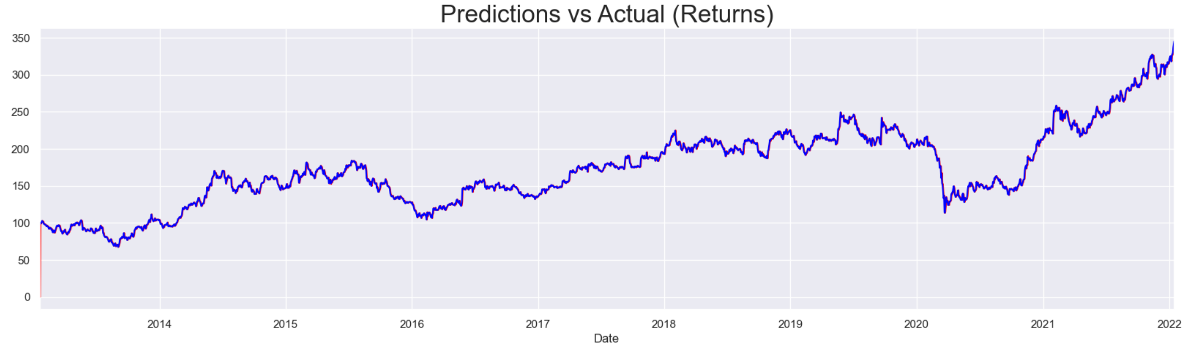
```
Text(0.5, 1.0, 'Pred')
```



Fitting the model with the help of Return

In [124]:

```
model_ret = ARIMA(training_data.Return[1:], order = (3,1,3))
results_ret = model_ret.fit()

df_pred_ar = results_ret.predict(start_date = start_date, end = end_date_train)

df_pred_ar[start_date:end_date_train].plot(figsize = (20,5), color = "red")
df['Return'][start_date:end_date_train].plot(color = "blue")
plt.title("Predictions vs Actual (Returns)", size = 24)
plt.show()
```

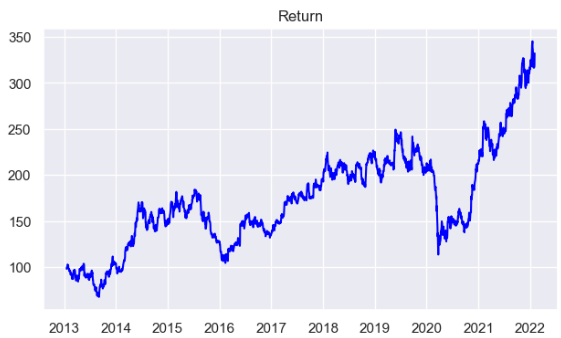


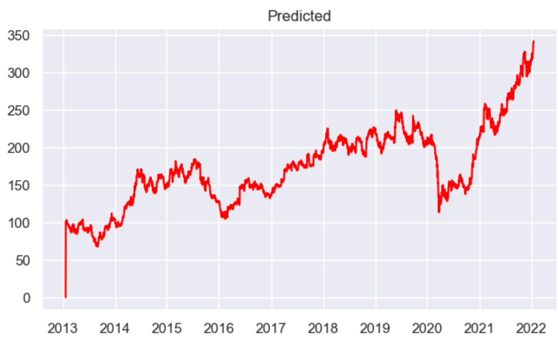
In [125]:

```
fig, (axis1, axis2) = plt.subplots(1, 2, figsize = (16,4))

axis1.plot(training_data.Return, color = "Blue")
axis1.set_title("Return")

axis2.plot(df_pred_ar, color = "Red")
axis2.set_title("Predicted")
```

Out[125]:

```
Text(0.5, 1.0, 'Predicted')
```





In [141]:

```
df_pred_ar.tail(), training_data['Return'][:-10]
```

Out[141]:

```
(Date
 2022-01-11    326.348336
 2022-01-12    326.678815
 2022-01-13    329.648008
 2022-01-14    337.319585
 2022-01-17    341.413065
 Freq: B, Name: predicted_mean, dtype: float64,
 Date
 2013-01-18     98.076368
 2013-01-21    100.000000
 2013-01-22     99.109983
 2013-01-23     99.601245
 2013-01-24    101.126111
                  ...
 2022-01-12    329.429776
 2022-01-13    336.805336
 2022-01-14    341.202297
 2022-01-17    345.315549
 2022-01-18    337.439412
 Freq: B, Name: Return, Length: 2348, dtype: float64)
```

Mean Absolute Error

In [142]:

```
from sklearn.metrics import mean_absolute_error
```

In [153]:

```
mean_absolute_error(df_pred, training_data['Close'][:-11])
```

Out[153]:

```
12.211602708983543
```

In [159]:

```
mean_absolute_error(df_pred_ar, training_data['Return'][:-12])
```

Out[159]:

```
0.22757514010962202
```

Analysing the residual plot

In [160]:

```
df['residual'] = result.resid.iloc[:]
```

In [161]:

```
df.residual.mean()
```

Out[161]:

```
0.8183479228415355
```

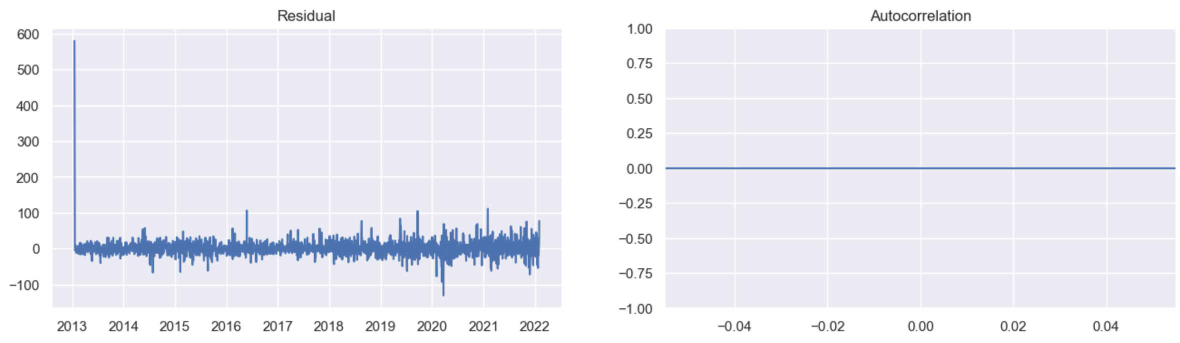
In [162]:

```
result = adfuller(df.residual.dropna())
print(result[0])
print("p-value =",result[1])
```

```
-57.82269464208276
p-value = 0.0
```

In [163]:

```
fig, (axis1, axis2) = plt.subplots(1, 2, figsize = (16,4))

axis1.plot(df.residual)
axis1.set_title("Residual")

sgt.plot_acf(df.residual, ax= axis2);
```



Auto ARIMA

In [164]:

```
model_auto = auto_arima(training_data.Return[1:], m = 5, max_p = 5, max_q = 5, max_P = 5, m
```

In [165]:

```
model_auto
```

Out[165]:

```
ARIMA(order=(0, 1, 2), scoring_args={}, seasonal_order=(0, 0, 1, 5),
      suppress_warnings=True)
```

In [166]:

```
model_auto.summary()
```

Out[166]:

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	2357
Model:	SARIMAX(0, 1, 2)x(0, 0, [1], 5)	**Log Likelihood**	-5885.283
Date:	Tue, 07 Feb 2023	**AIC**	11780.566
Time:	00:25:28	**BIC**	11809.390
Sample:	01-21-2013	**HQIC**	11791.062
	- 02-01-2022		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	0.0989	0.068	1.465	0.143	-0.033	0.231
ma.L1	0.0682	0.015	4.614	0.000	0.039	0.097
ma.L2	-0.0393	0.016	-2.457	0.014	-0.071	-0.008
ma.S.L5	0.0484	0.016	3.085	0.002	0.018	0.079
sigma2	8.6547	0.134	64.515	0.000	8.392	8.918

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	3388.49
Prob(Q):	1.00	**Prob(JB):**	0.00
Heteroskedasticity (H):	2.64	**Skew:**	0.30
Prob(H) (two-sided):	0.00	**Kurtosis:**	8.84

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

SARIMAX

In [167]:

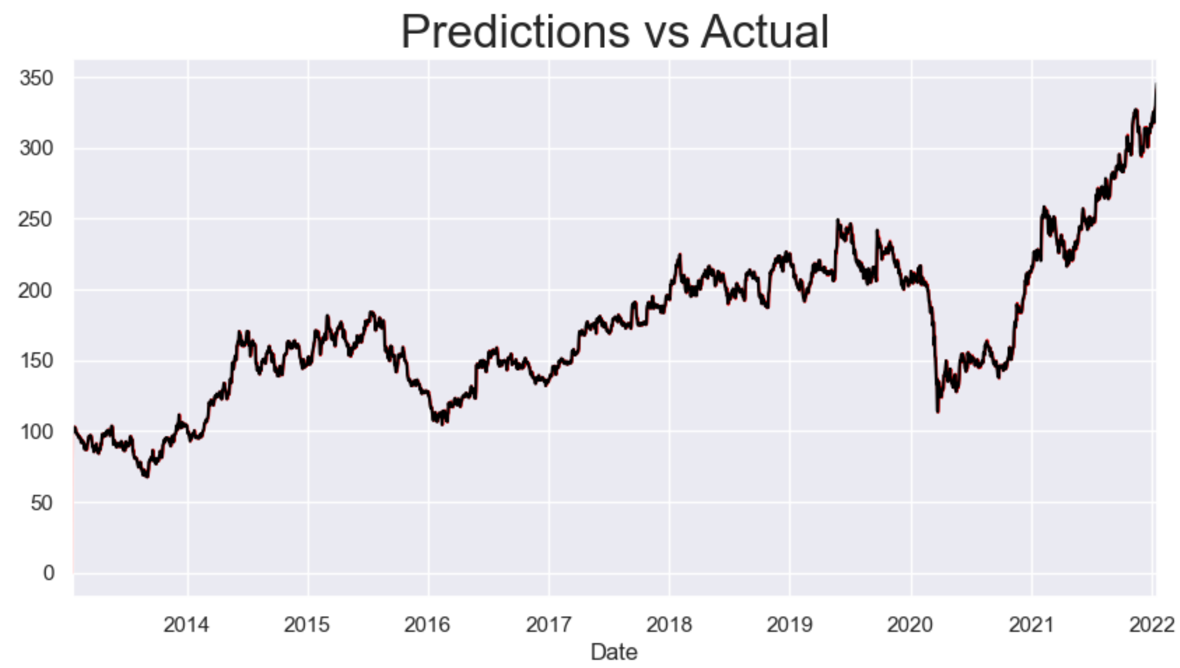
```
SAR_model= SARIMAX(training_data['Return'],order=(3, 1, 3),seasonal_order=(0,0,1,5))
results= SAR_model.fit()
```

In [168]:

```
training_data['forecast']= results.predict(start= start_date,end= end_date_train)
```

In [169]:

```
training_data['forecast'][start_date:end_date_train].plot(figsize = (10,5), color = "red")
training_data.Return[start_date:end_date_train].plot(color = "Black")
plt.title("Predictions vs Actual", size = 24)
plt.show()
```



In [94]:

```
training_data.forecast
```

Out[94]:

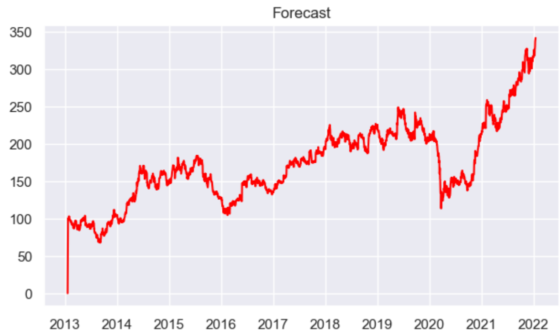
```
Date
2013-01-18      0.000000
2013-01-21     98.076434
2013-01-22    100.127092
2013-01-23     98.959355
2013-01-24     99.682036
                 ...
2022-01-11    326.436076
2022-01-12    327.016823
2022-01-13    329.500388
2022-01-14    337.126755
2022-01-17    341.558877
Freq: B, Name: forecast, Length: 2347, dtype: float64
```

In [170]:

```
fig, (axis1, axis2) = plt.subplots(1, 2, figsize = (16,4))

axis1.plot(training_data.Return, color = "Black")
axis1.set_title("Return")

axis2.plot(training_data.forecast, color = "Red")
axis2.set_title("Forecast")
```

Out[170]:

```
Text(0.5, 1.0, 'Forecast')
```



Predicting for Future

In [185]:

```
s_date_for = '2023-02-07'
e_date_for = '2023-02-28'
```

In [172]:

```
future_date = pd.DataFrame(pd.date_range(start= '2023-02-07', end= '2023-02-28'), columns=[
#future_date.set_index("Dates", inplace= True)
future_date
```

Out[172]:

	Dates
0	2023-02-07
1	2023-02-08
2	2023-02-09
3	2023-02-10
4	2023-02-11
5	2023-02-12
6	2023-02-13
7	2023-02-14
8	2023-02-15
9	2023-02-16
10	2023-02-17
11	2023-02-18
12	2023-02-19
13	2023-02-20
14	2023-02-21
15	2023-02-22
16	2023-02-23
17	2023-02-24
18	2023-02-25
19	2023-02-26
20	2023-02-27
21	2023-02-28

In [184]:

```
results.predict(start= future_date.Dates[0],end= future_date.Dates[21])
```

Out[184]:

```
2023-02-07    332.191949
2023-02-08    332.245119
2023-02-09    332.318916
2023-02-10    332.268893
2023-02-13    332.193876
2023-02-14    332.240733
2023-02-15    332.316845
2023-02-16    332.273167
2023-02-17    332.196087
2023-02-20    332.236576
2023-02-21    332.314499
2023-02-22    332.277204
2023-02-23    332.198561
2023-02-24    332.232663
2023-02-27    332.311903
2023-02-28    332.280989
Freq: B, Name: predicted_mean, dtype: float64
```