BTC-Time-Series-Model

September 29, 2021

1 Modeling BTC

1.1 Importing Necessary Libraries

[282]:
```
import pandas as pd
import numpy as np
import itertools
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.statespace.sarimax import SARIMAX
import statsmodels.api as sm
import matplotlib.pyplot as plt
from matplotlib.pylab import rcParams
from tqdm import tqdm_notebook as tqdm
import _pickle as pickle
plt.style.use('ggplot')
```

1.2 Loading in and formatting the Data

[283]:
```
bc = pd.read_csv('BTC-USD.csv')
bc.tail()
```

[283]:
```
            Date          Open          High          Low         Close  \
3354  2019-09-22   9988.379883  10096.660156  9871.969727  10036.980469
3355  2019-09-23  10036.980469  10073.790039  9636.200195   9694.599609
3356  2019-09-24   9694.599609   9787.580078  8107.009766   8542.540039
3357  2019-09-25   8542.540039   8752.200195  8246.240234   8446.969727
3358  2019-09-26   8447.000000   8471.019531  7753.709961   7937.390137

         Adj Close      Volume
3354  10036.980469   178007946
3355   9694.599609   327752087
3356   8542.540039  1084866143
3357   8446.969727   631898763
3358   7937.390137   514440800
```

1.2.1 Converting Dates into a Datetime Format

[284]:
```
bc['Date'] = pd.to_datetime(bc.Date)
bc.dtypes
```

[284]:
```
Date         datetime64[ns]
Open                float64
High                float64
Low                 float64
Close               float64
Adj Close           float64
Volume                int64
dtype: object
```

Setting dates as the index

[285]:
```
bc.set_index('Date', inplace=True)
bc.head()
```

[285]:
```
               Open     High      Low    Close  Adj Close  Volume
Date
2010-07-17  0.04951  0.04951  0.04951  0.04951    0.04951       0
2010-07-18  0.04951  0.08585  0.05941  0.08584    0.08584       5
2010-07-19  0.08584  0.09307  0.07723  0.08080    0.08080      49
2010-07-20  0.08080  0.08181  0.07426  0.07474    0.07474      20
2010-07-21  0.07474  0.07921  0.06634  0.07921    0.07921      42
```

Selecting only the Closing Price as well as the dates starting from January 2017. This is the time when Bitcoin and Cryptocurrency in general started to become popular to trade and is probably a better representation of current crypto trading trends.

[286]:
```
bc = bc[['Close']].loc['2017-01-01':]
bc.head()
```

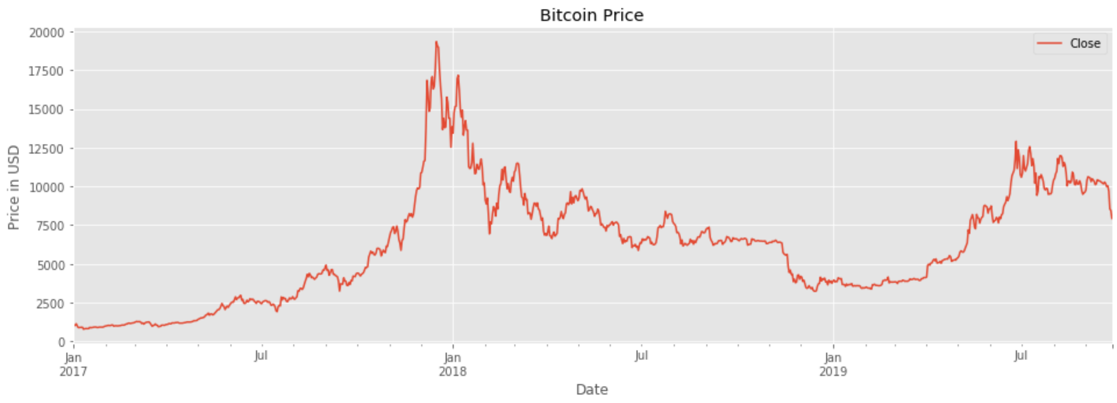
[286]:
```
                  Close
Date
2017-01-01   995.440002
2017-01-02  1017.049988
2017-01-03  1033.300049
2017-01-04  1135.410034
2017-01-05   989.349976
```

1.2.2 Exporting this data for later use

[287]:
```
with open("curr_bitcoin.pickle", 'wb') as fp:
    pickle.dump(bc, fp)
```

1.3 Plotting Bitcoin's Historical Prices

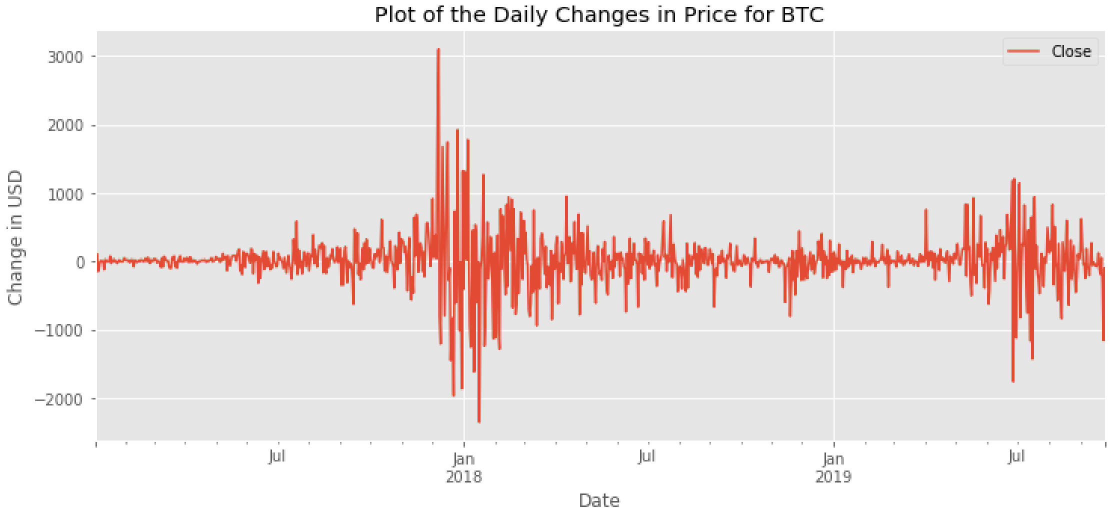
[410]:
```
bc.plot(figsize=(16,5))

plt.xlabel('Date')
plt.ylabel('Price in USD')
plt.title('Bitcoin Price')
plt.savefig('btcprice.png')
plt.show()
```



1.4 Detrending

1.4.1 Method #1 - Differencing the Data

[290]:
```
# Differencing the price
bc_diff = bc.diff(1).dropna()

# Plotting the differences daily
bc_diff.plot(figsize=(12,5))
plt.title('Plot of the Daily Changes in Price for BTC')
plt.ylabel('Change in USD')
plt.show()
```



Testing for Stationarity

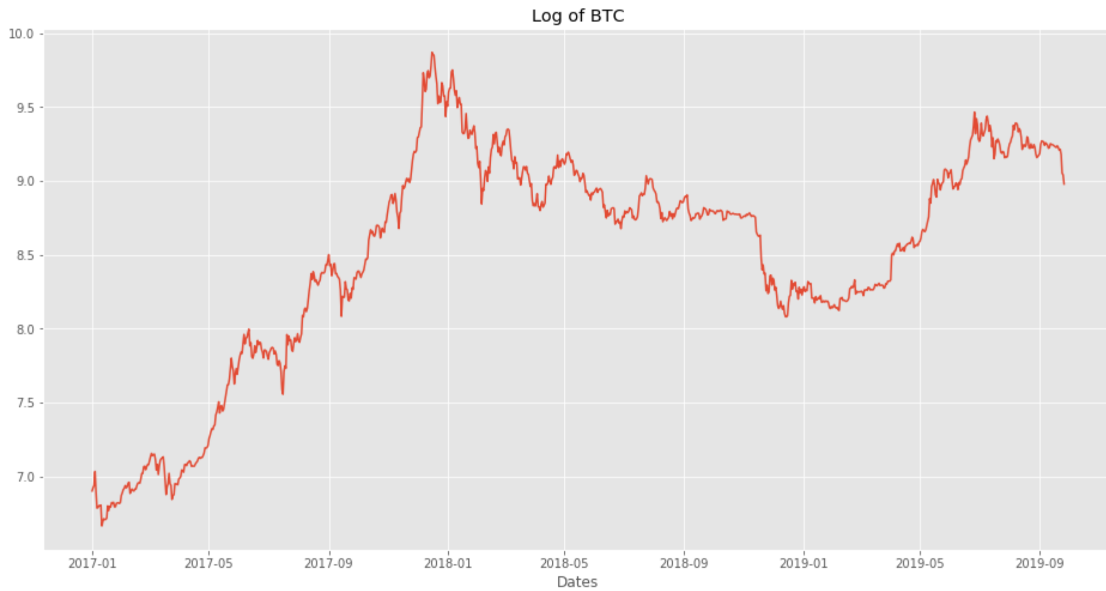
[336]:
```
results = adfuller(bc_diff.Close)
print(f"P-value: {results[1]}")
```

```
P-value: 0.2639720972728658
```

1.4.2 Method #2 - Taking the Log then differencing

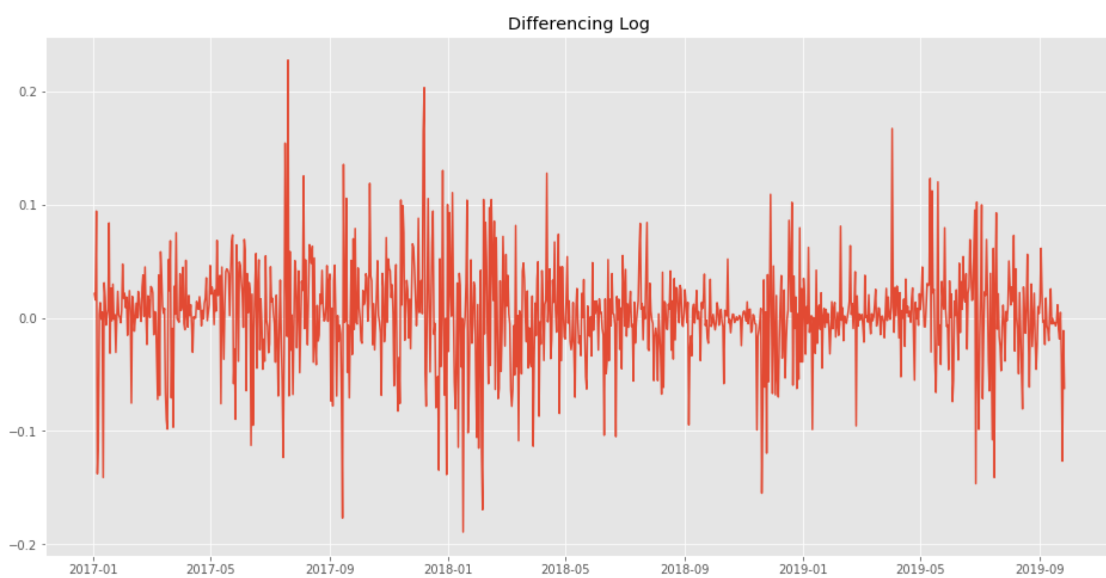
[292]:
```
# Converting the data to a logarithmic scale
bc_log = pd.DataFrame(np.log(bc.Close))
```

[411]:
```
# Plotting the log of the data
plt.figure(figsize=(16,8))
plt.plot(bc_log)

plt.title('Log of BTC')
plt.xlabel('Dates')

plt.savefig('btc_log.png')
plt.show()
```



[294]:
```
# Differencing the log values
log_diff = bc_log.diff().dropna()
```

[412]:
```
# Plotting the daily log difference
plt.figure(figsize=(16,8))
plt.plot(log_diff)
plt.title('Differencing Log')
plt.savefig('logdiff.png')
plt.show()
```



Testing for Stationarity

[296]:
```
results = adfuller(log_diff.Close)
print(f"P-value: {results[1]}")
```

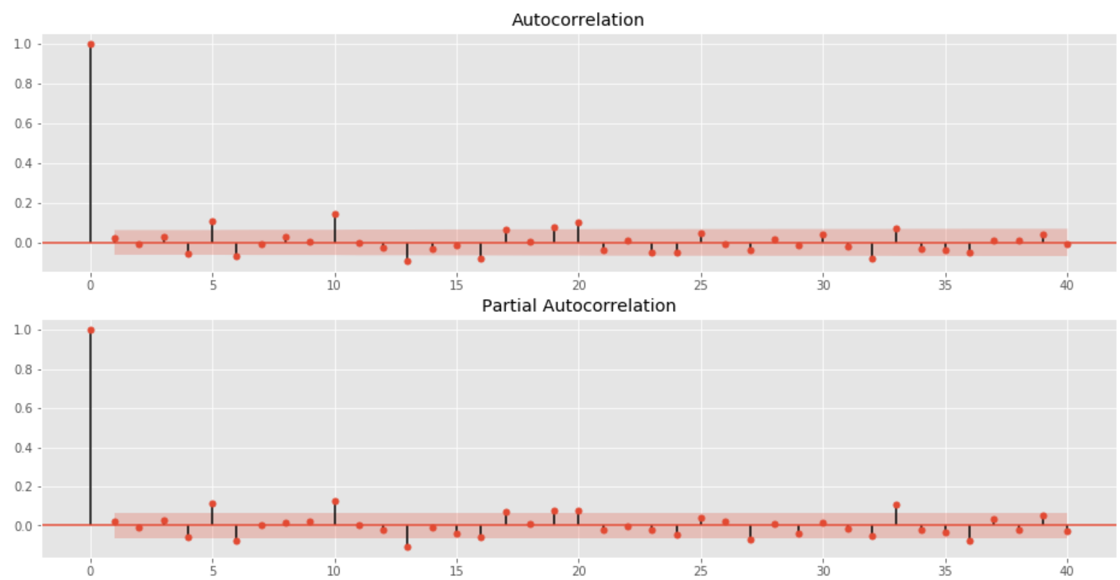
```
P-value: 5.879487529183016e-25
```

Since the p-values for both are less than .05, we can reject the null hypothesis and accept that our data is stationary.

1.5 PACF and ACF

ACF and PACF for the Differencing

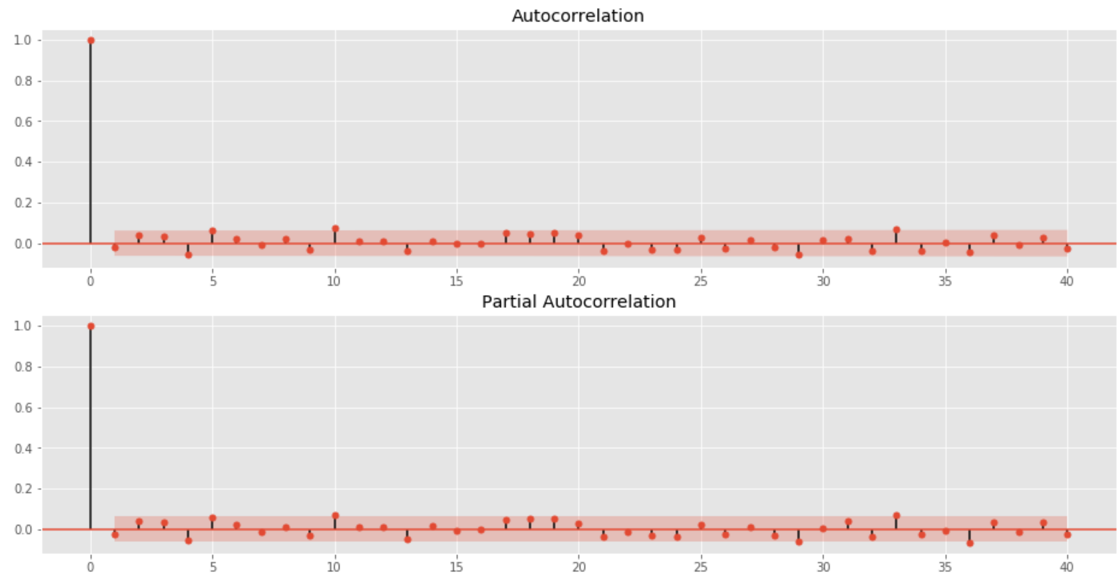
[297]:
```
fig, (ax1, ax2) = plt.subplots(2,1,figsize=(16,8))
plot_acf(bc_diff, ax=ax1, lags=40)
plot_pacf(bc_diff, ax=ax2, lags=40)
plt.show()
```



Appears to be some correlation at day 5 and 10 mostly.

ACF and PACF for the Log Difference

[413]:
```
fig, (ax1, ax2) = plt.subplots(2,1,figsize=(16,8))
plot_acf(log_diff, ax=ax1, lags=40)
plot_pacf(log_diff, ax=ax2, lags=40)
plt.savefig('acfpacf.png')
plt.show()
```



Some correlation at day 5 and 10 again but not as much as before.

1.6 Modeling

1.7 SARIMA Model for Differencing

1.7.1 Finding the Best Parameters for ARIMA

[337]:
```
def best_param(model, data, pdq, pdqs):
    """
    Loops through each possible combo for pdq and pdqs
    Runs the model for each combo
    Retrieves the model with lowest AIC score
    """
    ans = []
    for comb in tqdm(pdq):
        for combs in tqdm(pdqs):
            try:
                mod = model(data,
                            order=comb,
                            seasonal_order=combs,
                            enforce_stationarity=False,
                            enforce_invertibility=False,
                            freq='D')

                output = mod.fit()
                ans.append([comb, combs, output.aic])
            except:
                continue
```

```python
    ans_df = pd.DataFrame(ans, columns=['pdq', 'pdqs', 'aic'])
    return ans_df.loc[ans_df.aic.idxmin()]
```

[393]:
```python
# Assigning variables for p, d, q.
p = d = q = range(0,6)
d = range(2)

# Creating a list of all possible combinations of p, d, and q.
pdq = list(itertools.product(p, d, q))

# Keeping seasonality at zeroes
pdqs = [(0,0,0,0)]
```

[394]:
```python
# Finding the best parameters
best_param(SARIMAX, bc_log, pdq, pdqs)
```

```
HBox(children=(IntProgress(value=0, max=72), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))
```

```
HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))
```

```
/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))
```

```
HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))
```

```
/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))

/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)

HBox(children=(IntProgress(value=0, max=1), HTML(value='')))
```

```
/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py:508:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)
```

[394]:
```
pdq         (1, 0, 0)
pdqs     (0, 0, 0, 0)
aic          -3368.06
Name: 12, dtype: object
```

Best Parameters according to the function

[301]:
```
# pdq         (1, 0, 0)
# pdqs     (0, 0, 0, 0)
# aic          -3368.06
```

1.7.2 Fitting and Training SARIMAX

Train, test, split

[395]:
```
# Splitting 80/20
index = round(len(bc)*.80)

train = bc_log.iloc[:index]
test = bc_log.iloc[index:]
```

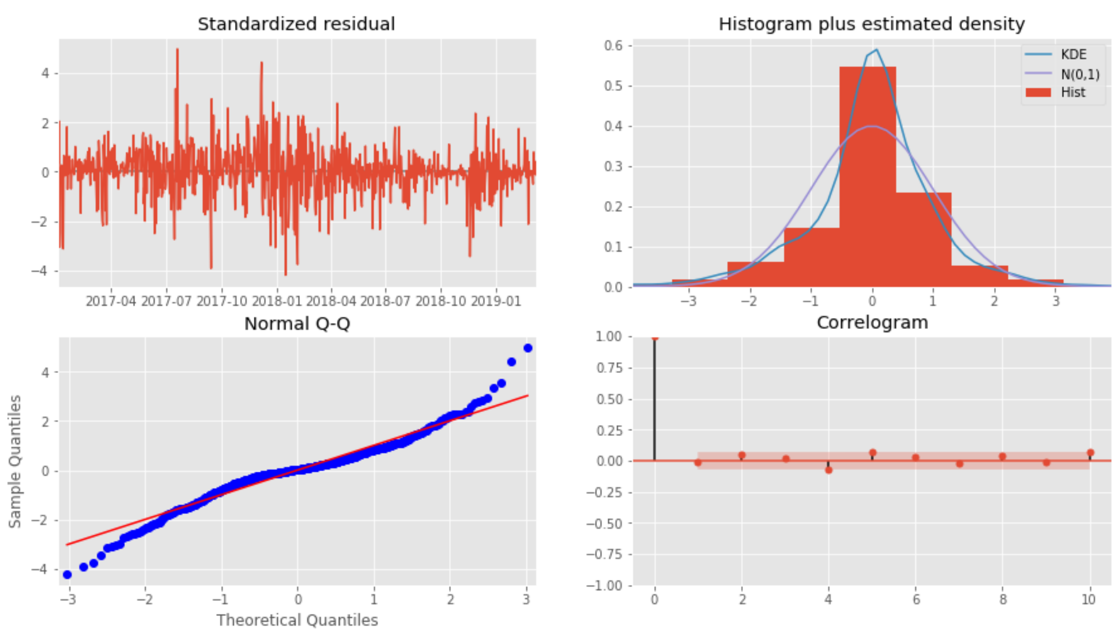
[396]:
```
# Fitting the model to the training set
model = SARIMAX(train,
                order=(1, 0, 0),
                seasonal_order=(0,0,0,0),
                freq='D',
                enforce_stationarity=False,
                enforce_invertibility=False)
output = model.fit()
```

1.7.3 Summary and Diagnostics from fitting the model

[397]:
```
print(output.summary())
output.plot_diagnostics(figsize=(15,8))
plt.show()
```

```
                           Statespace Model Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                  799
Model:               SARIMAX(1, 0, 0)   Log Likelihood                1334.461
Date:                Thu, 26 Sep 2019   AIC                          -2664.921
Time:                        16:07:08   BIC                          -2655.557
Sample:                    01-01-2017   HQIC                         -2661.324
```

```
                    - 03-10-2019
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          1.0002      0.000   5322.016      0.000       1.000       1.001
sigma2         0.0021   6.74e-05     30.664      0.000       0.002       0.002
================================================================================
===
Ljung-Box (Q):                       55.46   Jarque-Bera (JB):
250.31
Prob(Q):                              0.05   Prob(JB):
0.00
Heteroskedasticity (H):               0.53   Skew:
-0.13
Prob(H) (two-sided):                  0.00   Kurtosis:
5.73
================================================================================
===

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



1.7.4 Predictions with ARIMA

1.7.5 Transforming the Data back to its original price

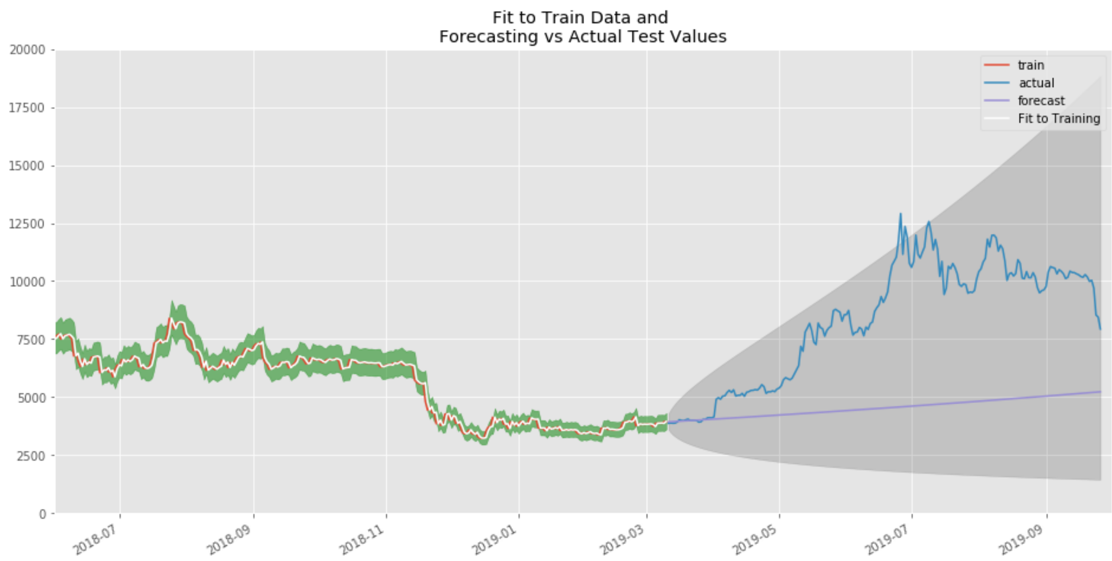
[398]:
```python
# Values to test against the test set
fc   = output.get_forecast(len(test))
conf = fc.conf_int()

# Transforming the values back to normal
fc_series    = np.exp(pd.Series(fc.predicted_mean, index=test.index))
lower_series = np.exp(pd.Series(conf.iloc[:, 0], index=test.index))
upper_series = np.exp(pd.Series(conf.iloc[:, 1], index=test.index))

etrain = np.exp(train)
etest  = np.exp(test)

# Values to test against the train set, see how the model fits
predictions = output.get_prediction(start=pd.to_datetime('2018'), dynamic=False)
pred        = np.exp(predictions.predicted_mean)

# Confidence interval for the training set
conf_int   = np.exp(predictions.conf_int())
low_conf   = np.exp(pd.Series(conf_int.iloc[:,0], index=train.index))
upper_conf = np.exp(pd.Series(conf_int.iloc[:,1], index=train.index))
```

```
/anaconda3/lib/python3.7/site-packages/statsmodels/tsa/base/tsa_model.py:320:
FutureWarning: Creating a DatetimeIndex by passing range endpoints is
deprecated.  Use `pandas.date_range` instead.
  freq=base_index.freq)
```

1.7.6 Plotting the Fitted Model and Testing against the Test set

[414]:
```python
rcParams['figure.figsize'] = 16, 8

# Plotting the training set, test set,forecast, and confidence interval.
plt.plot(etrain, label='train')
plt.plot(etest, label='actual')
plt.plot(fc_series, label='forecast')
plt.fill_between(lower_series.index, lower_series, upper_series, color='k',␣
 ↪alpha=.15)

# Plotting against the training data
pred.plot(label='Fit to Training', color='w')

# Confidence interval for the fitted data
plt.fill_between(conf_int.index, conf_int.iloc[:,0], conf_int.iloc[:,1],␣
 ↪color='g',alpha=.5)
```

```
# Limiting the viewing size
plt.xlim(['2018-06', '2019-10'])
plt.ylim([0, 20000])

plt.title('Fit to Train Data and \nForecasting vs Actual Test Values')
plt.legend()
plt.savefig('btc_fit_fc.png')
plt.show()
```



1.7.7 Calculating the RMSE for SARIMA

[401]:
```
forecast = pred
actual_val = etrain.Close

# Calculating our errors
rmse = np.sqrt(((forecast - actual_val) ** 2).mean())

print("The Root Mean Squared Error: ", rmse)
```

```
The Root Mean Squared Error:  358.9843097214424
```

On average, the SARIMA model is off the mark by $358.

1.7.8 Forecasting Future Values

Fitting the model to the entire dataset

[402]:
```
model = SARIMAX(bc_log,
                order=(1, 0, 0),
                seasonal_order=(0,0,0,0),
```

```
                freq='D',
                enforce_stationarity=False,
                enforce_invertibility=False)
output = model.fit()
```

```
/anaconda3/lib/python3.7/site-packages/statsmodels/tsa/base/tsa_model.py:191:
FutureWarning: Creating a DatetimeIndex by passing range endpoints is
deprecated.  Use `pandas.date_range` instead.
  start=index[0], end=index[-1], freq=freq)
```

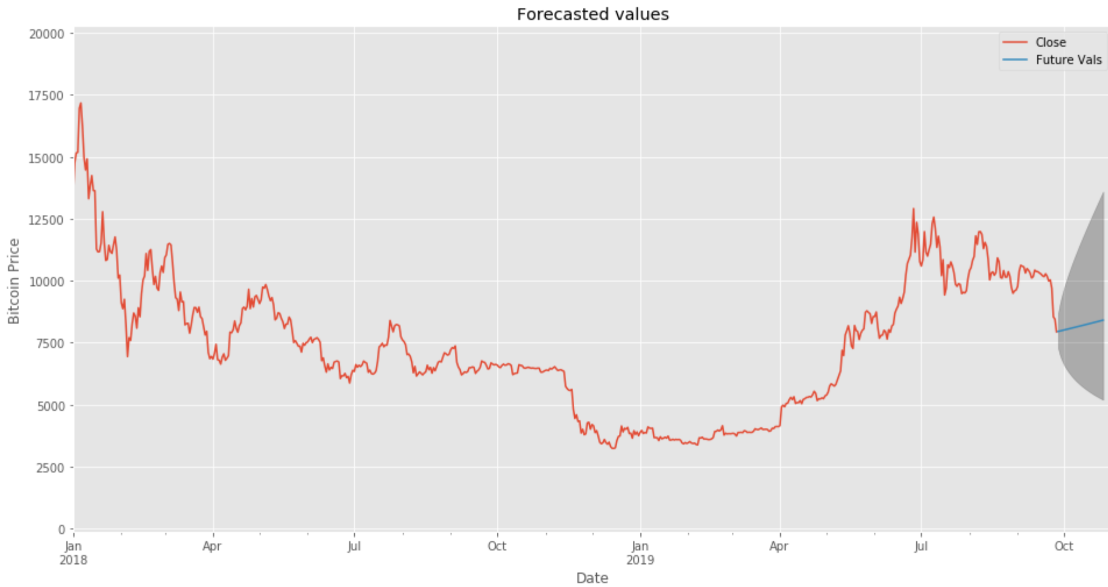
[403]:
```
# Getting the forecast of future values
future = output.get_forecast(steps=30)

# Transforming values back
pred_fut = np.exp(future.predicted_mean)

# Confidence interval for our forecasted values
pred_conf = future.conf_int()

# Transforming value back
pred_conf = np.exp(pred_conf)
```

```
/anaconda3/lib/python3.7/site-packages/statsmodels/tsa/base/tsa_model.py:320:
FutureWarning: Creating a DatetimeIndex by passing range endpoints is
deprecated.  Use `pandas.date_range` instead.
  freq=base_index.freq)
```

1.7.9 Plotting the forecasted values

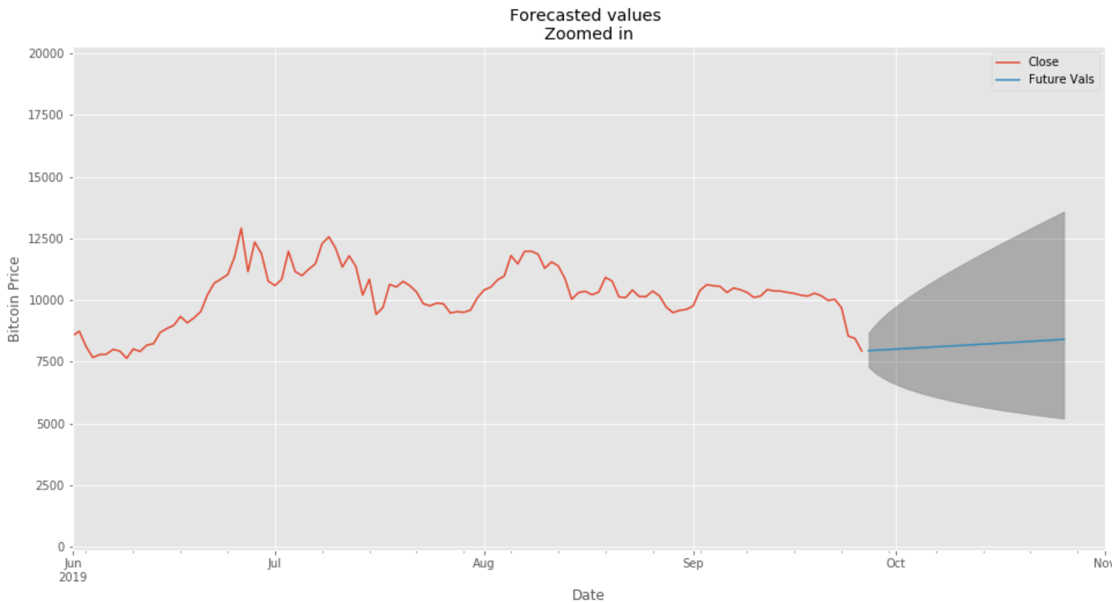
[415]:
```
# Plotting the prices up to the most recent
ax = np.exp(bc_log).plot(label='Actual', figsize=(16,8))

# Plottting the forecast
pred_fut.plot(ax=ax, label='Future Vals')

# Shading in the confidence interval
ax.fill_between(pred_conf.index,
                pred_conf.iloc[:, 0],
                pred_conf.iloc[:, 1], color='k', alpha=.25)

ax.set_xlabel('Date')
ax.set_ylabel('Bitcoin Price')
ax.set_xlim(['2018-01', '2019-11'])

plt.title('Forecasted values')
plt.legend()
plt.savefig('fc_val.png')
```

```
plt.show()
```



1.7.10 Zooming in on the Graph above

[416]:
```
ax = np.exp(bc_log).plot(label='Actual', figsize=(16,8))
pred_fut.plot(ax=ax, label='Future Vals')

ax.fill_between(pred_conf.index,
                pred_conf.iloc[:, 0],
                pred_conf.iloc[:, 1], color='k', alpha=.25)

ax.set_xlabel('Date')
ax.set_ylabel('Bitcoin Price')
ax.set_xlim(['2019-06','2019-11'])

plt.title('Forecasted values \nZoomed in')
plt.legend()
plt.savefig('fc_zoom.png')
plt.show()
```


Forecasted values
Zoomed in
Close
Future Vals
Bitcoin Price
20000
17500
15000
12500
10000
7500
5000
2500
0
Jun
2019
Jul
Aug
Sep
Oct
Nov
Date

[ ]: