MScFE690 Capstone Project: Draft Project (Module 6)

A Capstone Project Submitted for the Award of Master of Science in Financial Engineering at WorldQuant University

- **Calvin Emmy Okello (Uganda):** okellocalvinemmy@gmail.com
- **Kyaw Lin Oo (Country):** klo.kyawlinoo99@gmail.com
- **Mohammad Ilyas Zewar (Afghanistan):** m.ilyas.zewar@gmail.com


```
import warnings
warnings.filterwarnings("ignore")

try:
  import diagrams
except:
  !pip install diagrams --quiet
  import diagrams

try:
  from pmdarima.arima import auto_arima
except:
  !pip install pmdarima --quiet
  from pmdarima.arima import auto_arima

try:
  from arch import arch_model
except:
  !pip install arch --quiet
  from arch import arch_model

import pandas as pd, numpy as np, matplotlib.pyplot as plt,
from dateutil.relativedelta import relativedelta

import statsmodels.api as sm
from scipy import stats
from statsmodels.stats.diagnostic import acorr_ljungbox
from sklearn.decomposition import PCA

# Setting Figures Size
plt.rcParams["figure.figsize"] = (10, 4)
```


3 Study Materials and Methods

3.1 Scope of the Study

3.1.1 Time Scope:

The study shall be conducted on daily price data for 10 years. The first 6 years shall be used as in sample datasets for the time series modeling of the sub-indices whereas the remaining 4 years data shall be used as out of sample dataset for the time series modeling of the sub-indices. This shall be referred to as Phase 1 modeling. The phase 1 out of sample predictions for sub-indices shall be recombined with index fund actual prices to form raw data for Phase 2 modeling. The first 3 years of phase 2 raw data shall constitute the in-sample data set whereas the last 1 year shall constitute the out of the sample data set.

```
# web scraping list of Dow 30 subindices
djia_indices = pd.read_html('https://www.investopedia.com/terms/d/dow-30.asp')
djia_indices = pd.DataFrame(np.array(djia_indices)[0], columns = ["Company", "Symbol", "Y
djia_indices.sort_values("Year Added", ascending = False, inplace = True)
djia_indices.reset_index(inplace = True, drop = True)
djia_indices
```

	Company	Symbol	Year Added
0	Amazon.com	AMZN	2024
1	Salesforce	CRM	2020
2	Amgen	AMGN	2020
3	Honeywell	HON	2020
4	Dow	DOW	2019
5	Apple	AAPL	2015
6	NIKE	NKE	2013
7	Visa	V	2013
8	Goldman Sachs	GS	2013
9	UnitedHealth Group	UNH	2012
10	Cisco Systems	CSCO	2009
11	The Travelers Companies	TRV	2009
12	Chevron	CVX	2008
13	Verizon	VZ	2004
14	Microsoft	MSFT	1999
15	The Home Depot	HD	1999
16	Intel	INTC	1999
17	Johnson & Johnson	JNJ	1997
18	Walmart	WMT	1997
19	The Walt Disney Company	DIS	1991
20	JPMorgan Chase	JPM	1991
21	Caterpillar	CAT	1991
22	The Coca-Cola Company	KO	1987
23	Boeing	BA	1987
24	McDonald's	MCD	1985
25	American Express	AXP	1982
26	Merck & Co.	MRK	1979
27	IBM	IBM	1979
28	3M	MMM	1976
29	Procter & Gamble	PG	1932

Next steps: View recommended plots New interactive sheet

3.1.2 Content Scope:

The study shall involve Dow Jones Industrial Average sub-indices whose prices are highly correlated with the price of the index fund (Dow Jones Industrial Average) within the training set. The study shall consider prices for top 5 subindices which are most correlated with the index fund price. The study shall exclude subindices that have not spent 10 years by 31st December 2023 as a sub-index of Dow Jones Industrial Average.

```
# Subindices that had spent at least 10 years by December 31st, 2023
djia_indices_10yrs = djia_indices[djia_indices["Year Added"] < 2014]
djia_indices_10yrs.reset_index(inplace = True, drop = True)
djia_indices_10yrs
```

	Company	Symbol	Year Added
0	NIKE	NKE	2013
1	Visa	V	2013
2	Goldman Sachs	GS	2013
3	UnitedHealth Group	UNH	2012
4	Cisco Systems	CSCO	2009
5	The Travelers Companies	TRV	2009
6	Chevron	CVX	2008
7	Verizon	VZ	2004
8	Microsoft	MSFT	1999
9	The Home Depot	HD	1999
10	Intel	INTC	1999
11	Johnson & Johnson	JNJ	1997
12	Walmart	WMT	1997
13	The Walt Disney Company	DIS	1991
14	JPMorgan Chase	JPM	1991
15	Caterpillar	CAT	1991
16	The Coca-Cola Company	KO	1987
17	Boeing	BA	1987
18	McDonald's	MCD	1985
19	American Express	AXP	1982
20	Merck & Co.	MRK	1979
21	IBM	IBM	1979
22	3M	MMM	1976
23	Procter & Gamble	PG	1932

Next steps: View recommended plots New interactive sheet

```
# Adding index fund (Dow Jone Industrial Avearage)
djia_indices_10yrs.loc[len(djia_indices_10yrs.index)] = ["Dow Jones Industrial Average",
djia_indices_10yrs.dropna(inplace = True)
djia_indices_10yrs.drop_duplicates(subset = "Symbol", inplace = True)
```

```
5 djia_indices_10yrs.reset_index(drop = True, inplace = True)
```

	Company	Symbol	Year Added
0	NIKE	NKE	2013
1	Visa	V	2013
2	Goldman Sachs	GS	2013
3	UnitedHealth Group	UNH	2012
4	Cisco Systems	CSCO	2009
5	The Travelers Companies	TRV	2009
6	Chevron	CVX	2008
7	Verizon	VZ	2004
8	Microsoft	MSFT	1999
9	The Home Depot	HD	1999
10	Intel	INTC	1999
11	Johnson & Johnson	JNJ	1997
12	Walmart	WMT	1997
13	The Walt Disney Company	DIS	1991
14	JPMorgan Chase	JPM	1991
15	Caterpillar	CAT	1991
16	The Coca-Cola Company	KO	1987
17	Boeing	BA	1987
18	McDonald's	MCD	1985
19	American Express	AXP	1982
20	Merck & Co.	MRK	1979
21	IBM	IBM	1979
22	3M	MMM	1976
23	Procter & Gamble	PG	1932
24	Dow Jones Industrial Average	^DJI	1885

Next steps: View recommended plots New interactive sheet

3.2 Data Retrieval

The 30 companies that make up Dow Jones index fund as of July 2024 shall be extracted through web scraping with python panda's html reader (pandas.read_html, 2024). The web page that will be scrapped shall be for Investopedia and the content will be the Dow Jones companies posted on 7th July 2024 (Chen, 2024). The list will comprise of company name, symbol/ticker and the year the company was added to Dow Jones index fund. Only companies which were added within a period not less than 10 years will be used in sub-indices selection for model training. Adjusted closing prices for the subindices and for index fund (Dow Jones Industrial Average) shall be downloaded through yahoo finance module in python ("yfinance", 2024).

```
# required dates for modeling both phase 1 and phase 2
start_date = "2014-01-01"
end_date = "2023-12-31"

# downloading adjusted closing prices for all subindices that have taken at least 10 yrs
djia_indices_10yrs_data = pd.DataFrame()
for ticker in djia_indices_10yrs["Symbol"]:
  ticker_data = pd.DataFrame(yf.download(ticker, start = start_date, end = end_date)["Adj
  ticker_data.rename(columns = {"Adj Close": ticker}, inplace = True)
  djia_indices_10yrs_data[ticker] = ticker_data
```

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

```
# Scaling the data
djia_indices_10yrs_data_scaled = StandardScaler().fit_transform(djia_indices_10yrs_data[c
djia_indices_10yrs_data_scaled = pd.DataFrame(djia_indices_10yrs_data_scaled, columns = c
```

```

# Fitting PCA to the scaled data
pca = PCA().fit(djia_indices_10yrs_data_scaled)
print(f"Percentage of PCA Explained Variances: \n", np.round(pca.explained_variance_ratio
```

```
Percentage of PCA Explained Variances:
 [73. 14.  4.  3.  2.  1.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
  0.  0.  0.  0.  0.  0.]
```

```
loadings = pca.components_.T * np.sqrt(pca.explained_variance_)
loadings_df = pd.DataFrame(loadings, index = djia_indices_10yrs_data_scaled.columns, colu
loadings_df = loadings_df.iloc[:, : 5] # We shall Focus on Explained Variance at 96% (fir

# Explained variance updated to 100 percent
explained_variance_100percent = pca.explained_variance_ratio_[: 5] / np.sum(pca.explained

explained_variance_magnitude = pd.DataFrame(loadings_df * explained_variance_100percent).
explained_variance_magnitude
```

	PC1	PC2	PC3	PC4	PC5
NKE	0.697056	0.032660	0.011929	0.002836	0.000461
V	0.745333	0.013592	0.002953	0.003337	0.000834
GS	0.691948	0.020661	0.000973	0.010309	0.001011
UNH	0.738206	0.024755	0.000752	0.000558	0.001869
CSCO	0.722604	0.026953	0.005511	0.002525	0.000528
TRV	0.719478	0.021930	0.007093	0.000806	0.002485
CVX	0.590203	0.073439	0.013315	0.001865	0.001611
VZ	0.453513	0.103548	0.008084	0.004962	0.001056
MSFT	0.738027	0.016358	0.007128	0.000174	0.000674
HD	0.747121	0.004269	0.004836	0.001095	0.000834
INTC	0.455471	0.105614	0.000138	0.002632	0.003558
JNJ	0.740769	0.002412	0.000710	0.000104	0.002205
WMT	0.734672	0.006386	0.006012	0.002777	0.001047
DIS	0.407945	0.103179	0.011500	0.007369	0.001183
JPM	0.740405	0.015468	0.004024	0.002758	0.000800
CAT	0.725813	0.025018	0.003825	0.002691	0.000693
KO	0.733228	0.022449	0.000849	0.002994	0.001558
BA	0.276156	0.084665	0.029723	0.006923	0.001362
MCD	0.747502	0.011721	0.001110	0.003186	0.001060
AXP	0.727628	0.016813	0.000789	0.002722	0.000989
MRK	0.690992	0.042841	0.001274	0.007001	0.000085
IBM	0.417386	0.086414	0.012251	0.009144	0.005907
MMM	0.096280	0.120363	0.015151	0.009317	0.003784
PG	0.733036	0.015261	0.008877	0.002202	0.000176

Next steps: View recommended plots New interactive sheet

3.3 Variables/Sub-Indices Selection

Pearson product-moment correlation coefficient (r) shall be used to determine the sub-indices which are most correlated with the index fund (Dow Jones Industrial Average). Pearson product-moment correlation coefficient (r) formula that shall be used to determine the correlation is shown below:

Where xi is the ith independent variable (sub index), y is the dependent variable (Dow Jones Industrial Average), x and y are means for independent and dependent variables respectively (NLC, 2024). The study shall consider the top 5 most correlated variables as independent variables and the index fund price as the dependent variable.

```
# Getting correlation with Dow Jones Industrial Average and selecting top 5 indices
top5_pca_drivers = explained_variance_magnitude.sum(axis = 1).sort_values(ascending = Fal
top5_pca_drivers.rename("Magnitude of Explained Variance", inplace = True)
top5_pca_drivers
```

	Magnitude of Explained Variance
UNH	0.766140
V	0.766050
MCD	0.764580
JPM	0.763454
MSFT	0.762361

dtype: float64

3.4 Data Cleaning Process

The data shall be checked for missing values and in case there are any, the study shall consider dropping the missing rows provided they are less than 5% of the rows of the entire dataset when all variables are combined into a single data frame. In case the columns with null values are greater than or equals to 5%, the nulls will be replaced through filling them with the most preceding record. This is majorly aimed at reducing huge variations/swings within a short duration of time which are associated with replacing nulls with means.

```
# Getting the top 5 indices with higher correlation
selected_indices = list(top5_pca_drivers.index)
selected_indices.append("^DJI")
selected_indices
```

```
['UNH', 'V', 'MCD', 'JPM', 'MSFT', '^DJI']
```

```
# Extracting Adj. Price data for the selected indices
study_data = djia_indices_10yrs_data[selected_indices]

# Looking for null values if any
study_data.isna().sum()
```

	0
UNH	0
V	0
MCD	0
JPM	0
MSFT	0
^DJI	0

dtype: int64

```
study_data.head(3)
```

Date	UNH	V	MCD	JPM	MSFT	^DJI
2014-01-02 00:00:00+00:00	63.254967	51.229275	72.703751	43.285358	31.120729	16441.349609
2014-01-03 00:00:00+00:00	63.704544	51.264046	72.801826	43.619984	30.911367	16469.990234

Next steps: View recommended plots | New interactive sheet

3.5 Phase 1: Data Preprocessing and Time Series Analysis

3.5.1 Input Variables:

The sub-index daily prices shall be converted into a daily log return data. The log return shall be used majorly because it compresses large amounts of variations/jumps into a smaller range as per the advice by (Rawle, 2023). This is expected to smoothen data during a regime switch since the scope of this study does not incorporate regime switch in the time series modeling.

3.5.2 Data Training, Validation and Testing:

Several time series analysis techniques such as Auto-Regressive Integrated Moving Average (ARIMA), Generalized Autoregressive Conditional Heteroskedasticity (GARCH), Long Short-Term Memory (LSTM) among others shall be used and the model technique that output the lowest mean squared error shall be considered for sub-index modeling. Walk-Forward Optimization shall be used during modeling for better convergence of hyperparameters and improvement of model performance.

```
1 phase1_data = study_data.iloc[:, : -1]
2 phase1_data.head(3)
```

Date	UNH	V	MCD	JPM	MSFT
2014-01-02 00:00:00+00:00	63.254967	51.229275	72.703751	43.285358	31.120729
2014-01-03 00:00:00+00:00	63.704544	51.264046	72.801826	43.619984	30.911367
2014-01-06 00:00:00+00:00	62.975025	50.955757	72.281471	43.872814	30.258137

Next steps: View recommended plots New interactive sheet

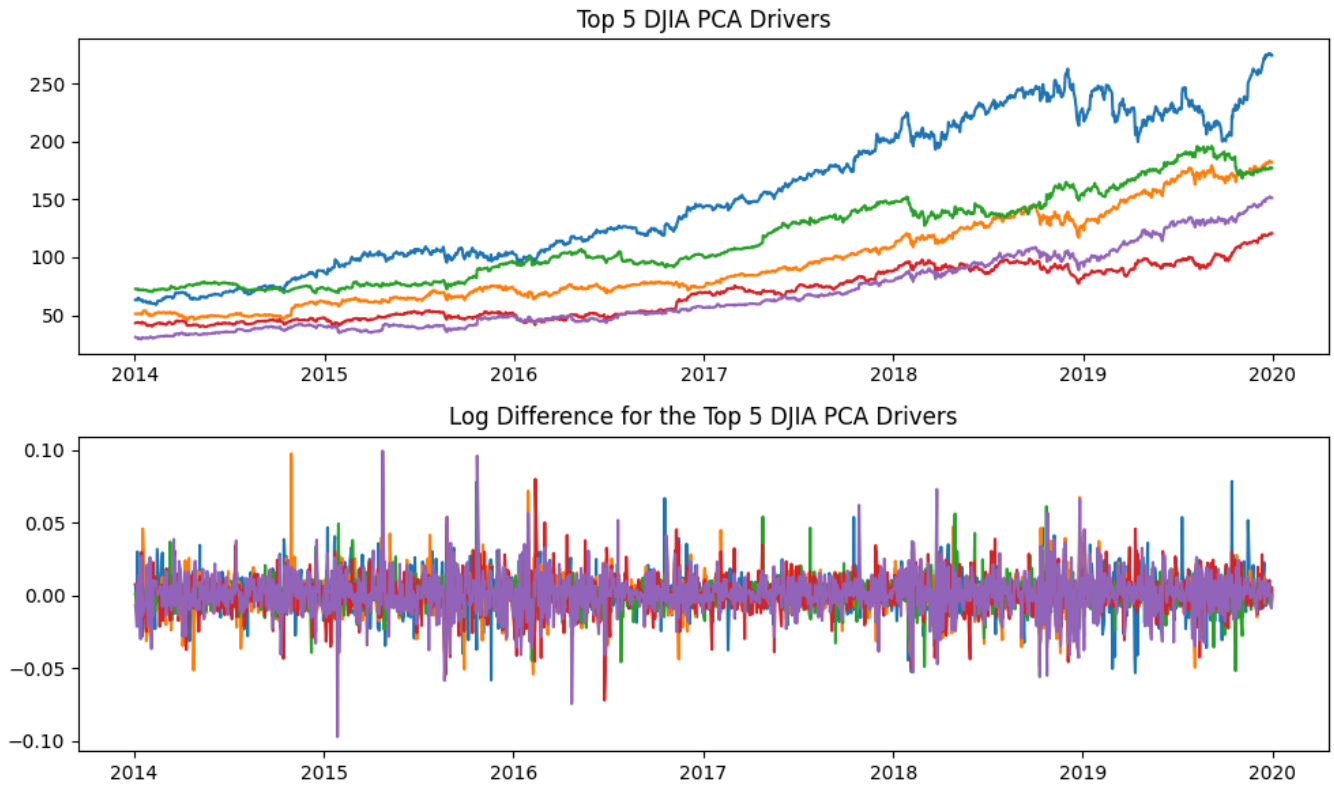
```
1 start_date = "2014-01-01"
2 end_date = "2023-12-31"
3
4 # setting time periods for phase 1
5 train_start = pd.to_datetime(dt.datetime.strptime(start_date, "%Y-%m-%d"))
6 train_end = pd.to_datetime(dt.datetime.strptime(end_date, "%Y-%m-%d"))
7 test_start = pd.to_datetime(train_end - relativedelta(years = 4) + dt.timedelta(days = 1)
```

```
1 # splitting data into training and testing datasets
2 phase1_data.index = pd.to_datetime(phase1_data.index).tz_localize(None)
3 phase1_train_set = phase1_data.loc[train_start: test_start]
4 phase1_test_set = phase1_data.loc[test_start: ]
5 phase1_train_set.head(3)
```

Date	UNH	V	MCD	JPM	MSFT
2014-01-02	63.254967	51.229275	72.703751	43.285358	31.120729
2014-01-03	63.704544	51.264046	72.801826	43.619984	30.911367
2014-01-06	62.975025	50.955757	72.281471	43.872814	30.258137

Next steps: View recommended plots New interactive sheet

```
fig, (ax1, ax2) = plt.subplots(2, 1, figsize = (10, 6))

ax1.plot(phase1_train_set)
ax1.set_title("Top 5 DJIA PCA Drivers")

ax2.plot(pd.DataFrame(np.log(phase1_train_set).diff().dropna()))
ax2.set_title("Log Difference for the Top 5 DJIA PCA Drivers")
fig.tight_layout()
```



3.5.3 Modeling Individual Sub-Indices:

For modeling purposes, we have started with defining various functions where the first function looks at the trend, log differences, ACF, PACF plots…etc for the selected sub-index and the second function is looking at the GARCH diagnostics for that particular sub-index. The third function will model the data for the selected sub-index while the fourth one will provide the model predictions for the specified sub-index.

```
def price_logdiff_acf_pacf(data, ticker):
  # Plotting stock prices
  data[f'{ticker}'].plot(figsize = (10, 1.5))
  plt.title(f"{ticker} Stock Price")
  plt.ylabel("Stock Price")
  plt.xlabel("Time")
```

```

  # ACF and PACF Plots for Stock Price
  fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (10, 1.5))
  sm.graphics.tsa.plot_acf(data[f'{ticker}'], title = f"{ticker} Stock Price ACF", lags =
  sm.graphics.tsa.plot_pacf(data[f'{ticker}'], title = f"{ticker} Stock Price PACF", lags

  #Plotting log differences
  log_diff_data = np.log(data[f'{ticker}']).diff().dropna()
  fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (10, 1.5))
  ax1.plot(log_diff_data)
  ax1.set_title(f"First Difference of Log {ticker} Stock Price")

  x = np.linspace(min(log_diff_data), max(log_diff_data), len(log_diff_data))
  values, bins, _ = ax2.hist(log_diff_data, bins = 50)  # Histogram

  (mu, sigma) = stats.norm.fit(log_diff_data)
  ax2.plot(x, stats.norm.pdf(x, mu, sigma) * sum(values * np.diff(bins)), "r")  # Density
  ax2.set_title(f"Histogram for {ticker} Prices Log Difference")

  # Normal QQ Plot f0r MCD stock prices Log difference
  qq = sm.qqplot(log_diff_data, stats.norm, fit = True, line = "q")
  qq.set_size_inches((10, 1.5))
  plt.title(f"Normal QQ Plot for {ticker} Prices Log Difference")

  # ACF and PACF Plots for log difference of Stock Prices
  fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (10, 1.5))
  sm.graphics.tsa.plot_acf(log_diff_data, title = f"ACF for Log Difference of {ticker} Pr
  sm.graphics.tsa.plot_pacf(log_diff_data, title=f"PACF for Log Difference of {ticker} Pr
```

```
# garch Diagnostic
def garch_diagnostic(data, p, q, ticker, dist = 'StudentsT'):
  # Log Return
  data = np.log(data[f'{ticker}']).diff().dropna()

  # GARCH(p, q) Model with Student's t White Noise
  garchpq_t_spec = arch_model(data, vol = "GARCH", p = p, q = q, mean = "AR", dist = dist
  garchpq_t_fit = garchpq_t_spec.fit()

  # Model Diagnostic Plots for the GARCH(p,q) Model with Student's t White Noise
  fig, ax = plt.subplots(3, 3, figsize = (10, 6))

  # Figure Row 1 Column 1
  ax[0, 0].plot(data)
  ax[0, 0].plot(2.0 * garchpq_t_fit.conditional_volatility / 100.0, c = "r")
  ax[0, 0].plot(-2.0 * garchpq_t_fit.conditional_volatility / 100.0, c = "r")
  ax[0, 0].tick_params(labelrotation = 45)
  ax[0, 0].set_title("Series with 2 Conditional SD")
  ax[0, 0].set_ylabel("Returns")

  # Figure Row 1 Column 2
  VaR_1 = stats.t(df = len(data) - 1).ppf(0.99)
```

```
  # VaR_1 = stats.norm.ppf(0.99)
  ax[0, 1].plot(data)
  ax[0, 1].plot(VaR_1 * garchpq_t_fit.conditional_volatility / 100.0, c = "r")
  ax[0, 1].plot(-VaR_1 * garchpq_t_fit.conditional_volatility / 100.0, c = "r")
  ax[0, 1].tick_params(labelrotation = 45)
  ax[0, 1].set_title("Series with 1% VaR Limits")
  ax[0, 1].set_ylabel("Returns")

  # Figure Row 1 Column 3
  sm.graphics.tsa.plot_acf(garchpq_t_fit.resid / 100.0, lags = 20, ax = ax[0, 2])
  ax[0, 2].set_title("ACF of Observations")
  ax[0, 2].set_ylim([-0.2, 0.2])

  # Figure Row 2 Column 1
  sm.graphics.tsa.plot_acf(garchpq_t_fit.resid ** 2, lags = 20, ax = ax[1, 0])
  ax[1, 0].set_title("ACF of Squared Observations")
  ax[1, 0].set_ylim([-0.2, 0.2])

  # Figure Row 2 Column 2
  sm.graphics.tsa.plot_pacf(garchpq_t_fit.resid ** 2, lags = 20, ax = ax[1, 1])
  ax[1, 1].set_title("PACF of Squared Observations")
  ax[1, 1].set_ylim([-0.2, 0.2])

  # Figure Row 2 Column 3
  ax[1, 2].xcorr(garchpq_t_fit.resid ** 2, garchpq_t_fit.resid, usevlines = True, maxlags
  ax[1, 2].set_title("Cross-Correlation of Squared Observations \n vs Actual Observation"

  # Figure Row 3 Column 1
  standaraized_residuals = garchpq_t_fit.std_resid
  min_val = np.min(standaraized_residuals)
  max_val = np.max(standaraized_residuals)
  empirical_density = np.linspace(min_val, max_val, len(standaraized_residuals))
  ax[2, 0].plot(empirical_density, stats.norm.pdf(empirical_density), lw = 1)
  ax[2, 0].set_title("Empirical Density of \n Standarized Residuals")

  # Figure Row 3 Column 2
  sm.qqplot(garchpq_t_fit.resid, stats.t, fit = True, line = "q", ax = ax[2, 1])
  ax[2, 1].set_title("StudentsT QQ-Plot")

  # Figure Row 3 Column 3
  sm.graphics.tsa.plot_acf((garchpq_t_fit.std_resid) ** 2, lags = 20, ax = ax[2, 2])
  ax[2, 2].set_title("ACF of Squared Standardized Residuals")
  ax[2, 2].set_ylim([-0.2, 0.2])
  fig.tight_layout()
```

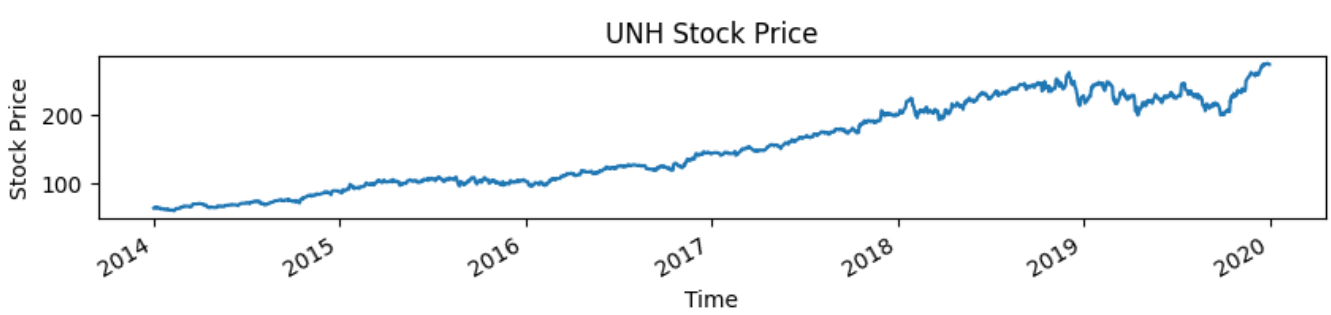
```
# GARCH(p, q) Model with Student's t and Normal White Noise
def p1_garch_modeling(price_data, ticker, p, q, dist, test_data = None, seed = 42):
  """
  dist can be 'StudentsT', or 'skewstudent' or 'normal'
  """
  #get log return
```

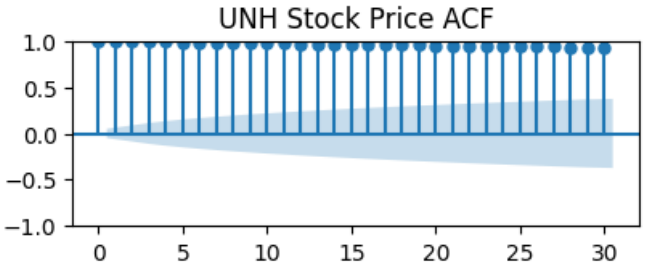
```
  data = np.log(price_data[f'{ticker}']).diff().dropna()
  test_data = np.log(test_data[f'{ticker}']).diff().dropna()

  # GARCH(p,q) Model with Student's t White Noise
  garchpq_t_spec = arch_model(data, vol = "GARCH", p = p, q = q, mean = "AR", dist = dist
  garchpq_t_fit = garchpq_t_spec.fit()
  print('Model Summary')
  print(garchpq_t_fit.summary())

  if test_data.empty:
    np.random.seed(seed)
    predicted_volatility = np.empty()
    z_t = np.empty()
  else:
    np.random.seed(seed)
    time_steps = len(test_data)
    forecasts = garchpq_t_fit.forecast(horizon = time_steps)
    predicted_variance = forecasts.variance[-1: ]  # Extract the last row with the x-step
    predicted_volatility = np.sqrt(predicted_variance.values.flatten())  # Convert variar
    z_t = np.random.normal(size = time_steps)
  return predicted_volatility, z_t, predicted_volatility * z_t
```

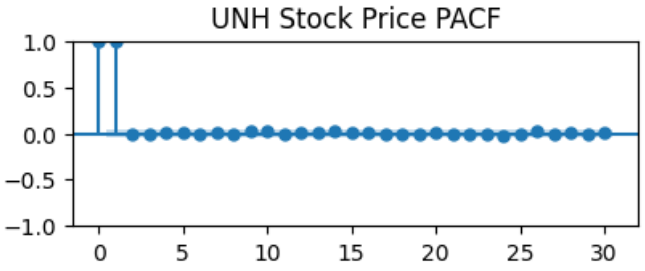
```
# price prediction
def price_prediction(price_data, ticker, p, q, dist, test_data, seed = 42):
  _, _, predictions = p1_garch_modeling(price_data, ticker, p, q, dist, test_data = test_

  predictions[0] = predictions[0] + price_data[f'{ticker}'][-1]
  predictions = pd.DataFrame(np.cumsum(predictions))
  predictions.index = test_data.index[1: ]

  predictions_log_diff = np.log(predictions).diff().dropna()
  test_log_diff = np.log(test_data[f'{ticker}']).diff().dropna()[1:]

  plt.figure(figsize = (10, 2.5))
  plt.plot(test_log_diff, c = 'r', label = 'Actual Log Difference')
  plt.plot(predictions_log_diff, c = 'b', label = 'Predicted Log Difference')
  plt.title(f'{ticker} Predicted Log Differences Vs {ticker} Actual Lof Differences')
  plt.legend()
  return predictions
```

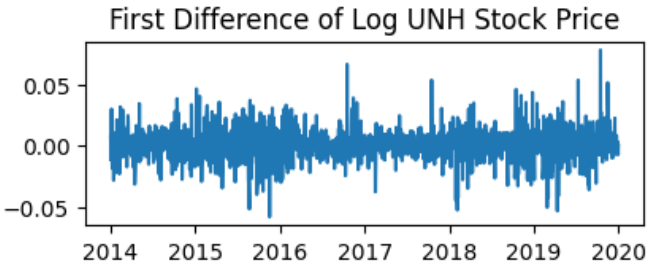
3.5.3.1 UNH Modeling Results

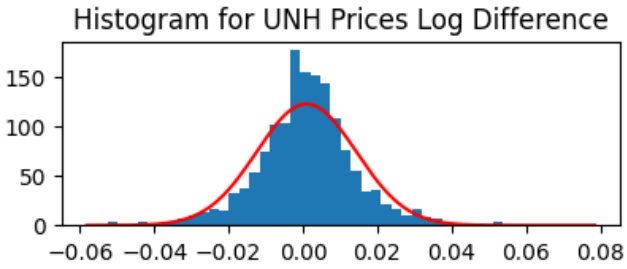
```
price_logdiff_acf_pacf(phase1_train_set, "UNH")
```

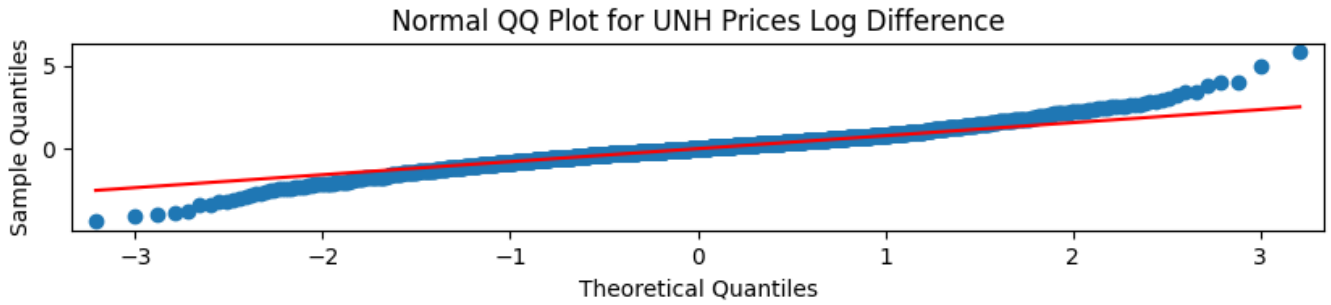

UNH Stock Price
Stock Price
200
100
2014
2015
2016
2017
2018
2019
2020
Time

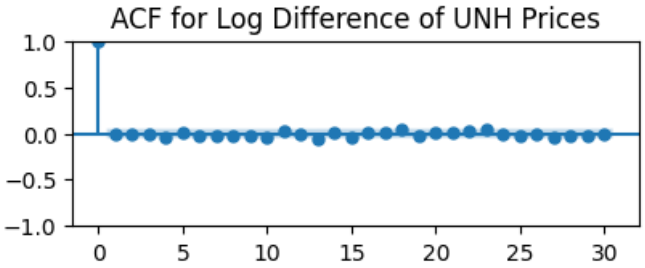

UNH Stock Price ACF
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30

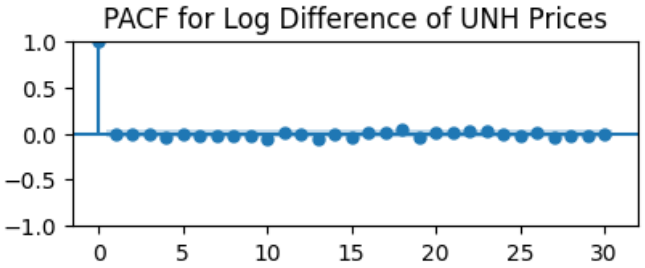

UNH Stock Price PACF
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30


First Difference of Log UNH Stock Price
0.05
0.00
−0.05
2014
2015
2016
2017
2018
2019
2020


Histogram for UNH Prices Log Difference
150
100
50
0
−0.06
−0.04
−0.02
0.00
0.02
0.04
0.06
0.08

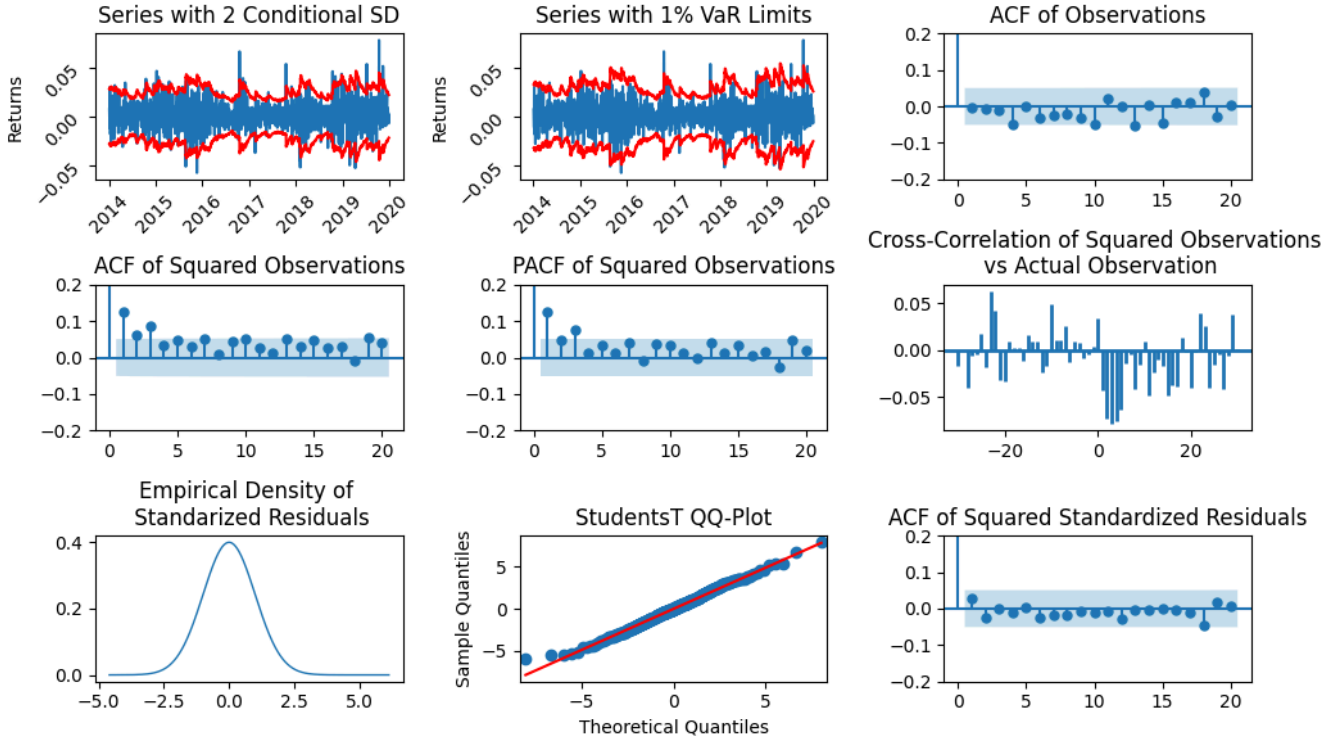

Normal QQ Plot for UNH Prices Log Difference
Sample Quantiles
5
0
−3
−2
−1
0
1
2
3
Theoretical Quantiles


ACF for Log Difference of UNH Prices
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30


PACF for Log Difference of UNH Prices
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30

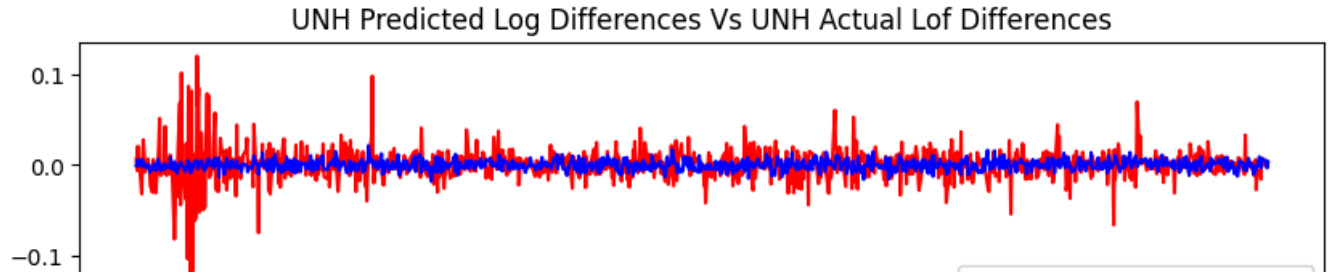
```
garch_diagnostic(data = phase1_train_set,  p = 1, q = 1, ticker = "UNH")
```

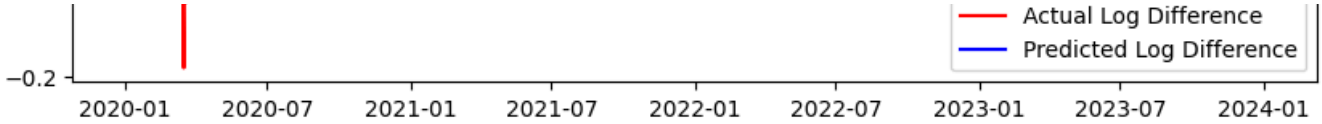
```
Iteration:      1,   Func. Count:      7,   Neg. LLF: 50394.90417388448
Iteration:      2,   Func. Count:     16,   Neg. LLF: 61593.334239160235
Iteration:      3,   Func. Count:     25,   Neg. LLF: 4545.03831649127
Iteration:      4,   Func. Count:     33,   Neg. LLF: 3771.4490790477475
Iteration:      5,   Func. Count:     40,   Neg. LLF: 2456.836816903225
Iteration:      6,   Func. Count:     47,   Neg. LLF: 3776.425510108596
Iteration:      7,   Func. Count:     54,   Neg. LLF: 2462.06009150593
Iteration:      8,   Func. Count:     61,   Neg. LLF: 2453.5279187496035
Iteration:      9,   Func. Count:     68,   Neg. LLF: 2453.3623711566843
Iteration:     10,   Func. Count:     74,   Neg. LLF: 2453.361110251927
Iteration:     11,   Func. Count:     80,   Neg. LLF: 2453.3610466044033
Iteration:     12,   Func. Count:     85,   Neg. LLF: 2453.3610466044047
Optimization terminated successfully    (Exit mode 0)
            Current function value: 2453.3610466044033
            Iterations: 12
            Function evaluations: 85
            Gradient evaluations: 12
```



```
predicted_unh = price_prediction(phase1_train_set, ticker = "UNH", p = 1, q = 1, dist = '
```

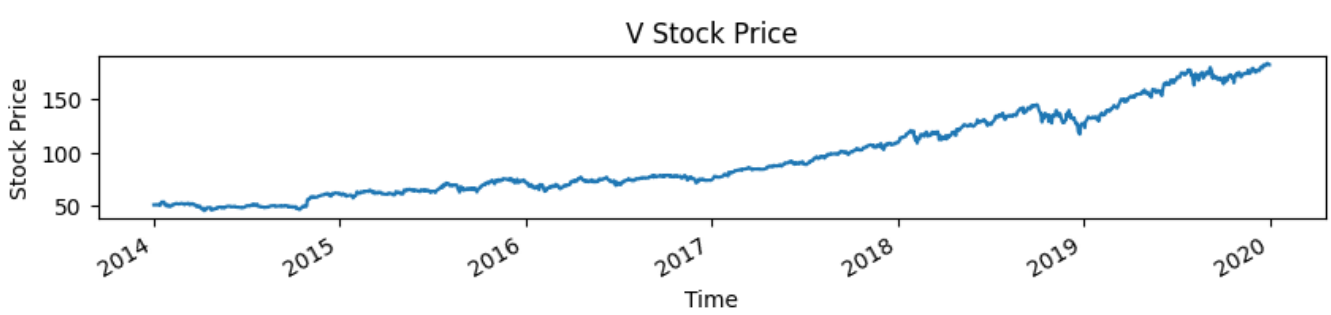
```
Iteration:      1,   Func. Count:      7,   Neg. LLF: 50394.90417388448
Iteration:      2,   Func. Count:     16,   Neg. LLF: 61593.334239160235
Iteration:      3,   Func. Count:     25,   Neg. LLF: 4545.03831649127
Iteration:      4,   Func. Count:     33,   Neg. LLF: 3771.4490790477475
Iteration:      5,   Func. Count:     40,   Neg. LLF: 2456.836816903225
Iteration:      6,   Func. Count:     47,   Neg. LLF: 3776.425510108596
Iteration:      7,   Func. Count:     54,   Neg. LLF: 2462.06009150593
Iteration:      8,   Func. Count:     61,   Neg. LLF: 2453.5279187496035
Iteration:      9,   Func. Count:     68,   Neg. LLF: 2453.3623711566843
Iteration:     10,   Func. Count:     74,   Neg. LLF: 2453.361110251927
Iteration:     11,   Func. Count:     80,   Neg. LLF: 2453.3610466044033
Iteration:     12,   Func. Count:     85,   Neg. LLF: 2453.3610466044047
Optimization terminated successfully    (Exit mode 0)
            Current function value: 2453.3610466044033
            Iterations: 12
            Function evaluations: 85
            Gradient evaluations: 12
Model Summary
                              AR - GARCH Model Results
==============================================================================
Dep. Variable:                    UNH   R-squared:                       0.000
Mean Model:                        AR   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:               -2453.36
Distribution:      Standardized Student's t   AIC:                     4916.72
Method:                  Maximum Likelihood   BIC:                     4943.32
                                              No. Observations:             1509
Date:                      Tue, Oct 29 2024   Df Residuals:                 1508
Time:                             17:48:33   Df Model:                        1
                                Mean Model
============================================================================
                 coef    std err          t      P>|t|    95.0% Conf. Int.
----------------------------------------------------------------------------
Const          0.1070  2.682e-02      3.990  6.609e-05 [5.444e-02,  0.160]
                            Volatility Model
============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
----------------------------------------------------------------------------
omega          0.0216  1.382e-02      1.566      0.117 [-5.442e-03,4.873e-02]
alpha[1]       0.0586  1.814e-02      3.229  1.243e-03  [2.302e-02,9.415e-02]
beta[1]        0.9327  2.195e-02     42.501      0.000      [  0.890,  0.976]
                            Distribution
============================================================================
                 coef    std err          t      P>|t|  95.0% Conf. Int.
----------------------------------------------------------------------------
nu             4.7985      0.594      8.073  6.858e-16 [  3.634,  5.963]
============================================================================

Covariance estimator: robust
```

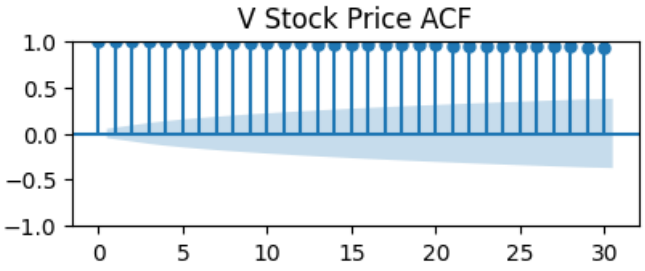


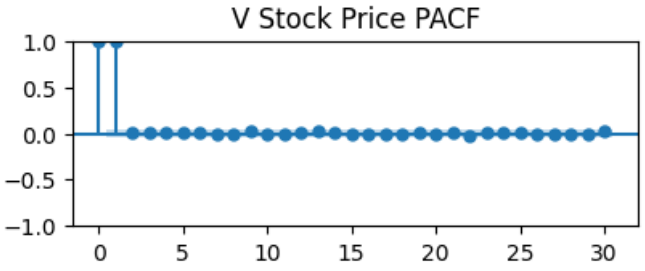


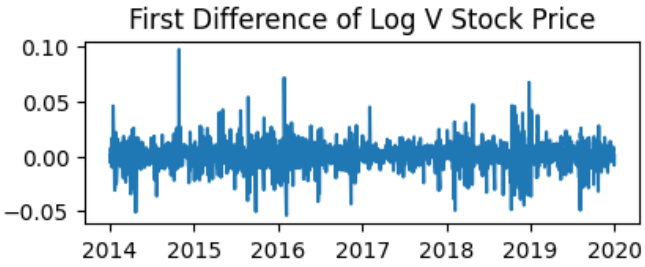
3.5.3.2 V Modeling Results

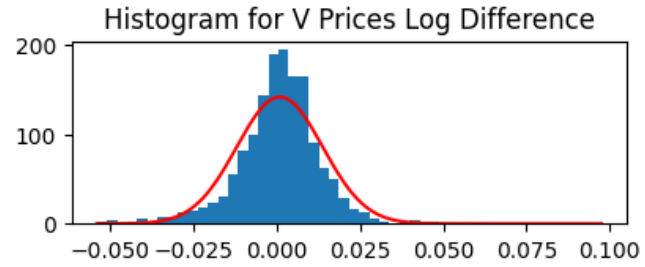
```
1 price_logdiff_acf_pacf(phase1_train_set, "V")
```

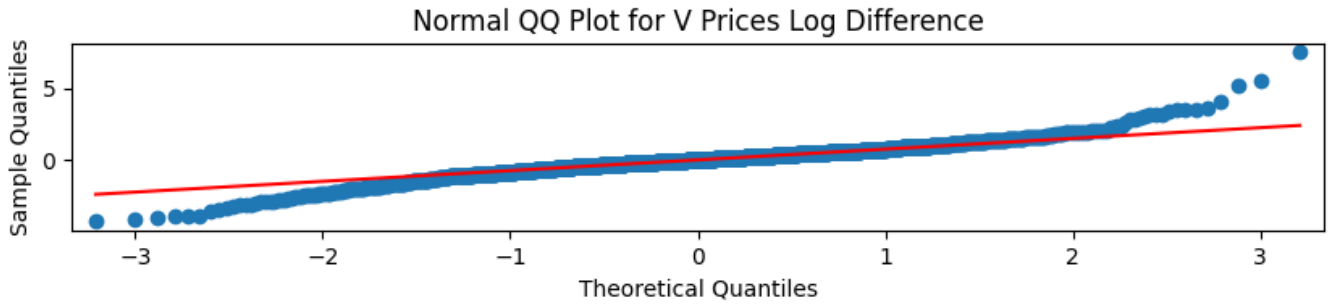

V Stock Price
Stock Price
150
100
50
2014
2015
2016
2017
2018
2019
2020
Time

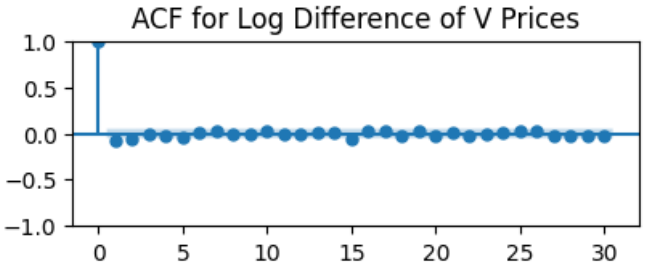

V Stock Price ACF
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30

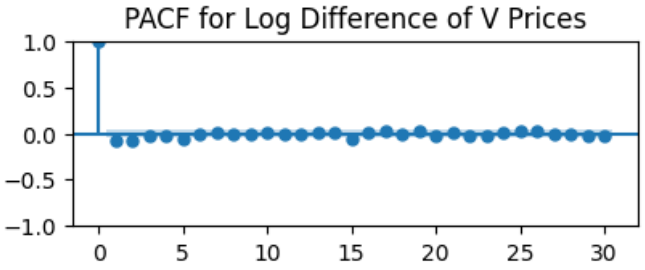

V Stock Price PACF
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30


First Difference of Log V Stock Price
0.10
0.05
0.00
−0.05
2014
2015
2016
2017
2018
2019
2020


Histogram for V Prices Log Difference
200
100
0
−0.050
−0.025
0.000
0.025
0.050
0.075
0.100

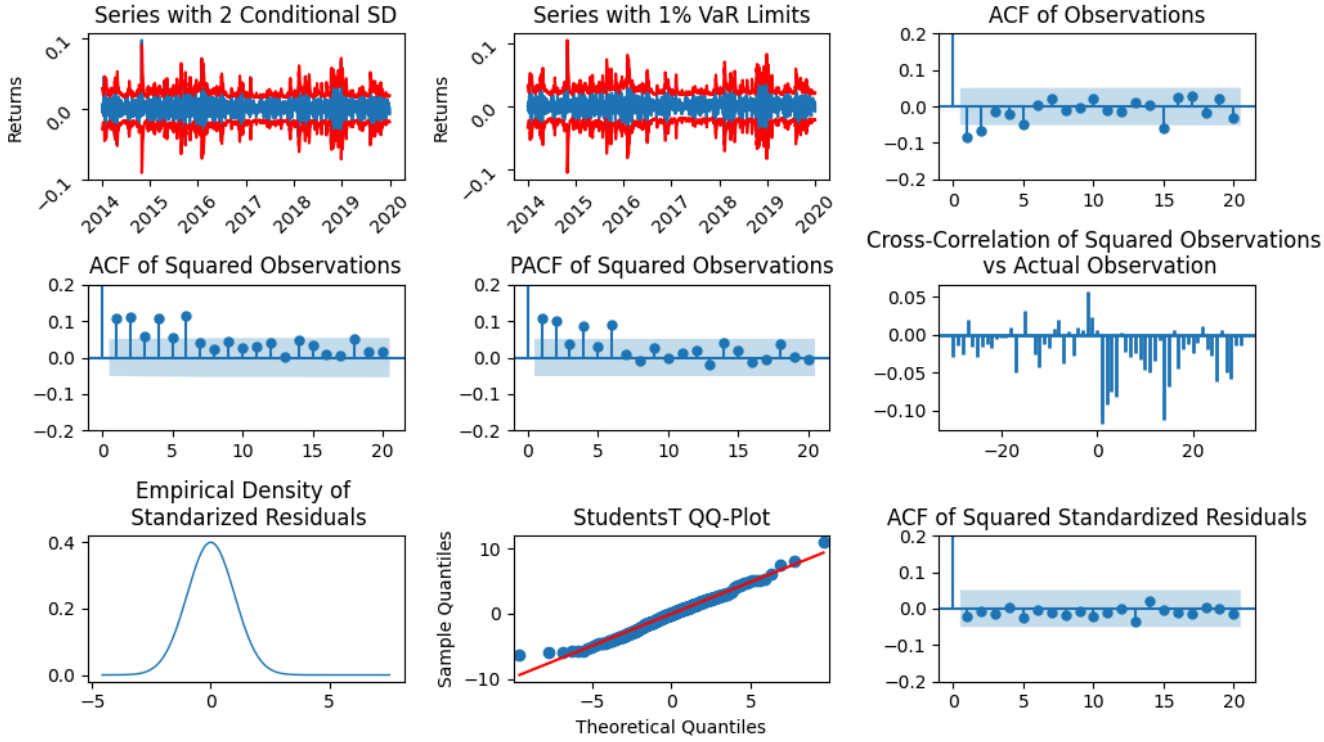

Normal QQ Plot for V Prices Log Difference
Sample Quantiles
5
0
−3
−2
−1
0
1
2
3
Theoretical Quantiles


ACF for Log Difference of V Prices
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30


PACF for Log Difference of V Prices
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30

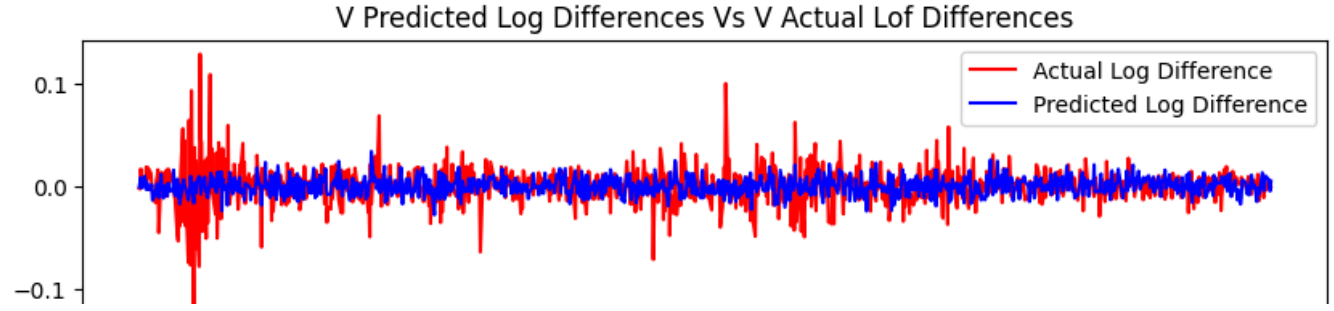
```
garch_diagnostic(data = phase1_train_set,  p = 1, q = 1, ticker = "V")
```

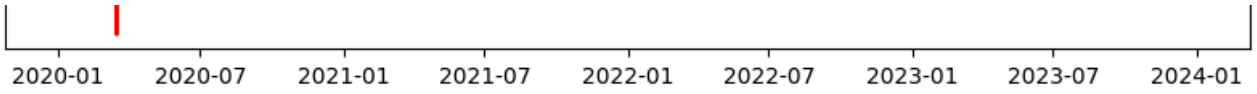
```
Iteration:      1,   Func. Count:      7,   Neg. LLF: 9557.124256502097
Iteration:      2,   Func. Count:     17,   Neg. LLF: 4177.30959687375
Iteration:      3,   Func. Count:     25,   Neg. LLF: 3765.1353250004886
Iteration:      4,   Func. Count:     32,   Neg. LLF: 3498.3473565876125
Iteration:      5,   Func. Count:     39,   Neg. LLF: 2339.1855485393407
Iteration:      6,   Func. Count:     46,   Neg. LLF: 3660.320773131
Iteration:      7,   Func. Count:     54,   Neg. LLF: 2347.5080830312013
Iteration:      8,   Func. Count:     61,   Neg. LLF: 2333.6643361334623
Iteration:      9,   Func. Count:     67,   Neg. LLF: 2333.6635720409913
Iteration:     10,   Func. Count:     73,   Neg. LLF: 2333.6635014926164
Iteration:     11,   Func. Count:     79,   Neg. LLF: 2333.6635008934527
Optimization terminated successfully    (Exit mode 0)
            Current function value: 2333.6635008934527
            Iterations: 11
            Function evaluations: 79
            Gradient evaluations: 11
```



```
predicted_v = price_prediction(phase1_train_set, ticker = "V", p = 1, q = 1, dist = "Stud
```

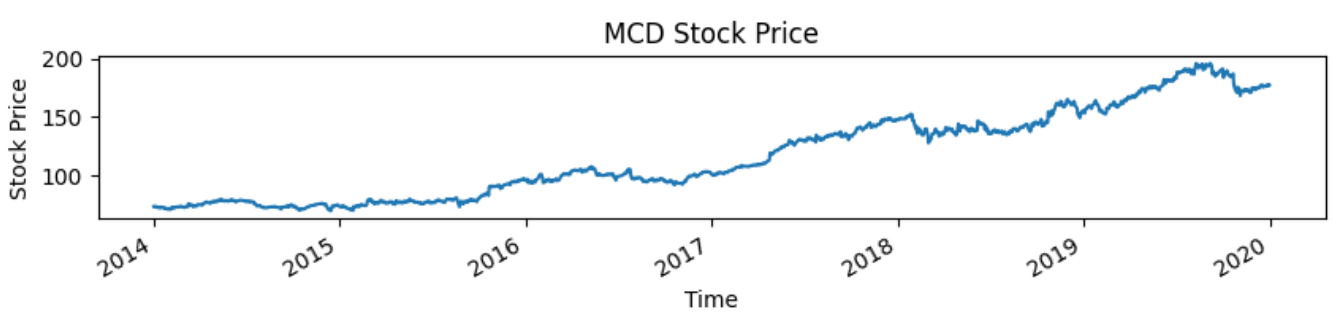
```
Iteration:      1,   Func. Count:      7,   Neg. LLF: 9557.124256502097
Iteration:      2,   Func. Count:     17,   Neg. LLF: 4177.30959687375
Iteration:      3,   Func. Count:     25,   Neg. LLF: 3765.1353250004886
Iteration:      4,   Func. Count:     32,   Neg. LLF: 3498.3473565876125
Iteration:      5,   Func. Count:     39,   Neg. LLF: 2339.1855485393407
Iteration:      6,   Func. Count:     46,   Neg. LLF: 3660.320773131
Iteration:      7,   Func. Count:     54,   Neg. LLF: 2347.5080830312013
Iteration:      8,   Func. Count:     61,   Neg. LLF: 2333.6643361334623
Iteration:      9,   Func. Count:     67,   Neg. LLF: 2333.6635720409913
Iteration:     10,   Func. Count:     73,   Neg. LLF: 2333.6635014926164
Iteration:     11,   Func. Count:     79,   Neg. LLF: 2333.6635008934527
Optimization terminated successfully    (Exit mode 0)
            Current function value: 2333.6635008934527
            Iterations: 11
            Function evaluations: 79
            Gradient evaluations: 11
Model Summary
                           AR - GARCH Model Results
==============================================================================
Dep. Variable:                              V   R-squared:                       0.000
Mean Model:                                AR   Adj. R-squared:                  0.000
Vol Model:                              GARCH   Log-Likelihood:               -2333.66
Distribution:      Standardized Student's t   AIC:                            4677.33
Method:                  Maximum Likelihood   BIC:                            4703.92
                                              No. Observations:                 1509
Date:                      Tue, Oct 29 2024   Df Residuals:                     1508
Time:                             17:48:39   Df Model:                            1
                              Mean Model
============================================================================
                 coef    std err          t      P>|t|    95.0% Conf. Int.
----------------------------------------------------------------------------
Const          0.1300  2.339e-02      5.559  2.718e-08 [8.418e-02,  0.176]
                           Volatility Model
============================================================================
                 coef    std err          t      P>|t|    95.0% Conf. Int.
----------------------------------------------------------------------------
omega          0.1512  4.436e-02      3.408  6.541e-04 [6.424e-02,  0.238]
alpha[1]       0.2099  4.167e-02      5.037  4.738e-07   [  0.128,  0.292]
beta[1]        0.7280  4.876e-02     14.930  2.111e-50   [  0.632,  0.824]
                           Distribution
============================================================================
                 coef    std err          t      P>|t|  95.0% Conf. Int.
----------------------------------------------------------------------------
nu             4.2688      0.465      9.188  4.008e-20 [  3.358,  5.179]
============================================================================

Covariance estimator: robust
```

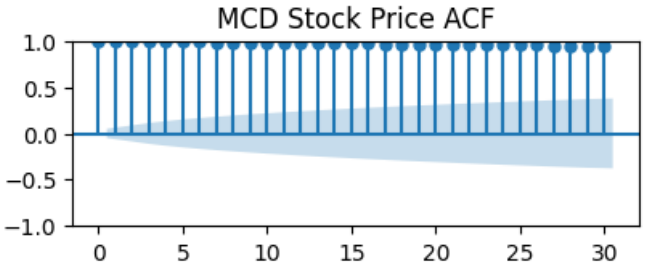


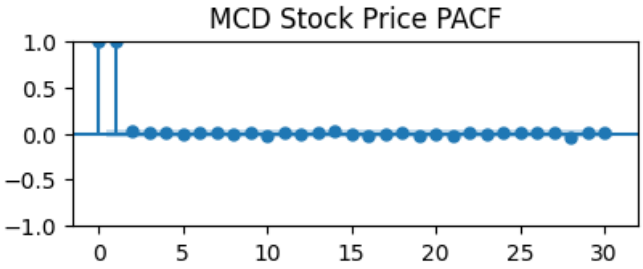


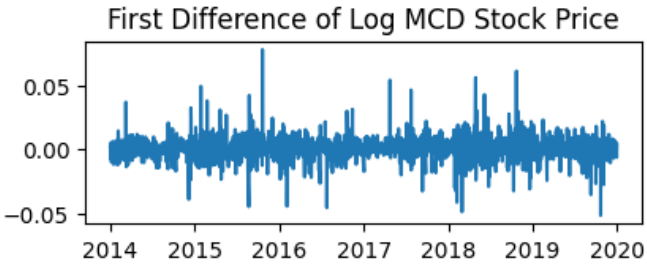
3.5.3.3 MCD Modeling Results

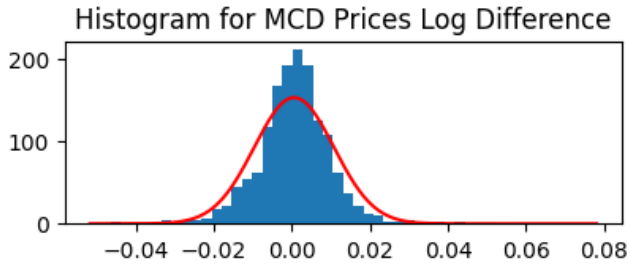
```
price_logdiff_acf_pacf(phase1_train_set, "MCD")
```

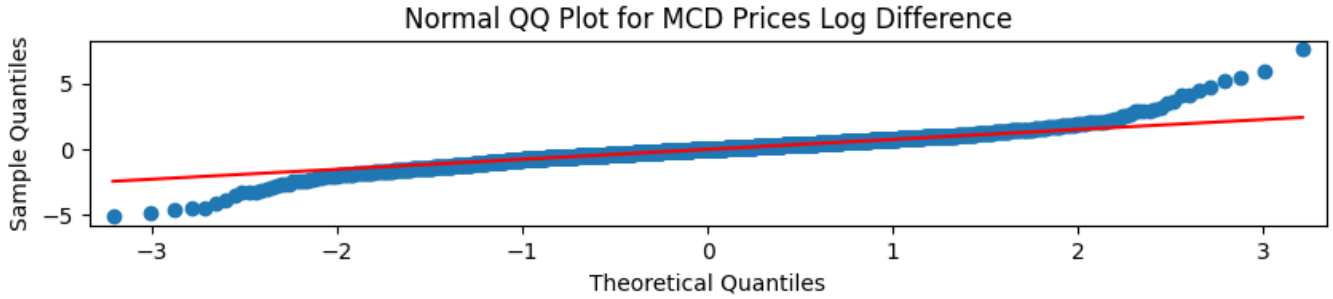

MCD Stock Price
Stock Price
200
150
100
2014
2015
2016
2017
2018
2019
2020
Time

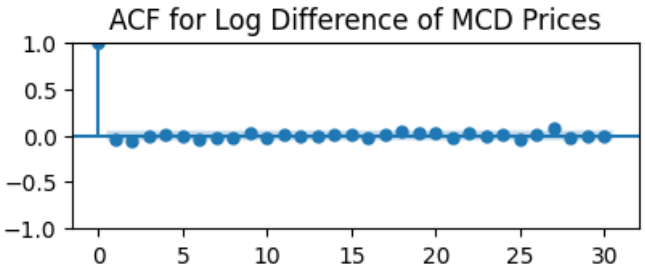

MCD Stock Price ACF
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30

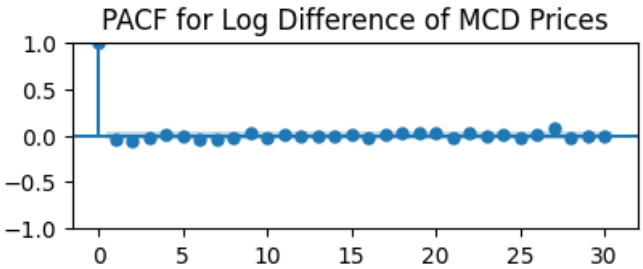

MCD Stock Price PACF
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30


First Difference of Log MCD Stock Price
0.05
0.00
−0.05
2014
2015
2016
2017
2018
2019
2020


Histogram for MCD Prices Log Difference
200
100
0
−0.04
−0.02
0.00
0.02
0.04
0.06
0.08

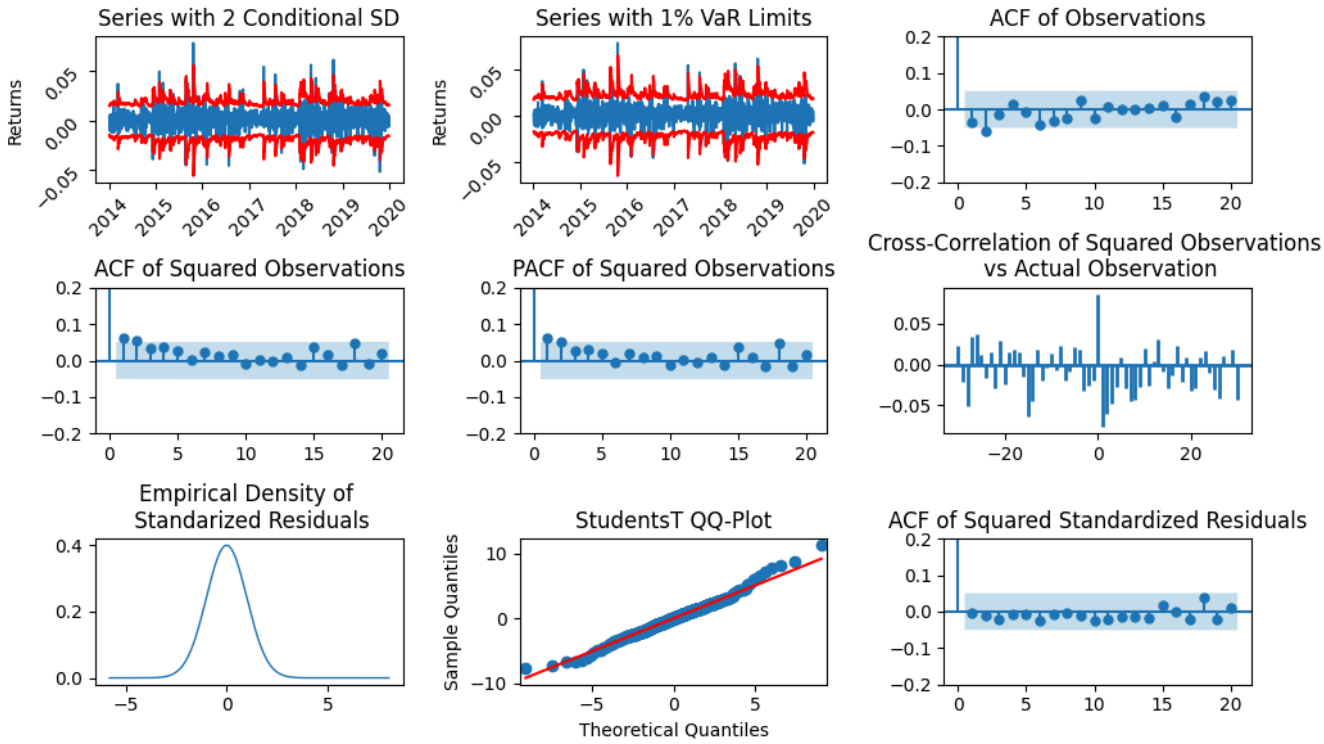

Normal QQ Plot for MCD Prices Log Difference
Sample Quantiles
5
0
−5
−3
−2
−1
0
1
2
3
Theoretical Quantiles


ACF for Log Difference of MCD Prices
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30


PACF for Log Difference of MCD Prices
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30

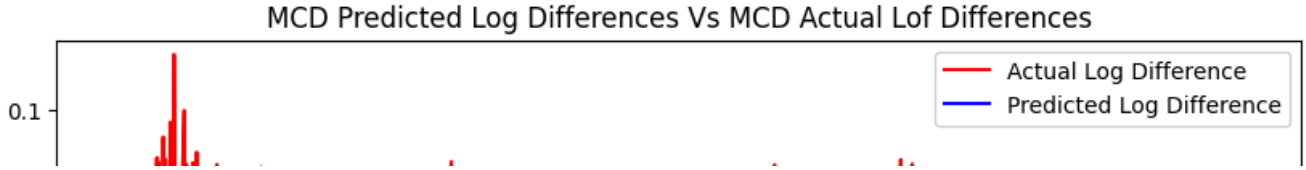
```
garch_diagnostic(data = phase1_train_set,  p = 1, q = 1, ticker = "MCD")
```

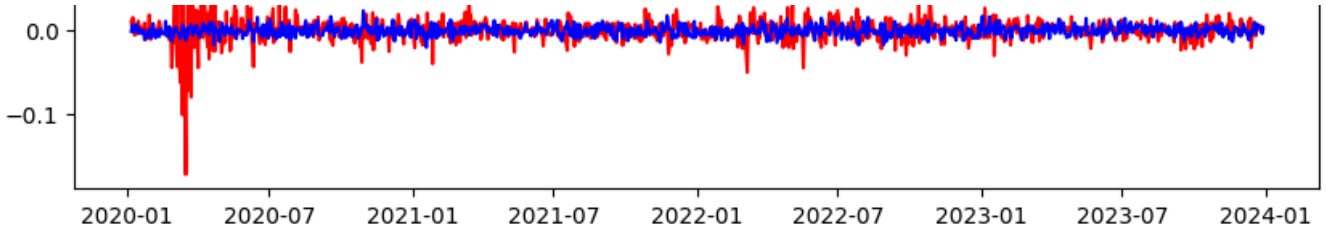
```
Iteration:      1,   Func. Count:      7,   Neg. LLF: 46191.39779881113
Iteration:      2,   Func. Count:     17,   Neg. LLF: 23308.1382965706
Iteration:      3,   Func. Count:     25,   Neg. LLF: 2140.847635119466
Iteration:      4,   Func. Count:     32,   Neg. LLF: 3219.1489690845974
Iteration:      5,   Func. Count:     39,   Neg. LLF: 2003.8900191457774
Iteration:      6,   Func. Count:     46,   Neg. LLF: 2773.32298640647
Iteration:      7,   Func. Count:     53,   Neg. LLF: 1996.6071910609478
Iteration:      8,   Func. Count:     60,   Neg. LLF: 1989.5901521164171
Iteration:      9,   Func. Count:     66,   Neg. LLF: 1989.5261282474435
Iteration:     10,   Func. Count:     73,   Neg. LLF: 1989.4157714215798
Iteration:     11,   Func. Count:     80,   Neg. LLF: 1989.3157471885734
Iteration:     12,   Func. Count:     86,   Neg. LLF: 1989.3126594514108
Iteration:     13,   Func. Count:     92,   Neg. LLF: 1989.312448570466
Iteration:     14,   Func. Count:     98,   Neg. LLF: 1989.3124436615915
Iteration:     15,   Func. Count:    103,   Neg. LLF: 1989.31244366158
Optimization terminated successfully    (Exit mode 0)
            Current function value: 1989.3124436615915
            Iterations: 15
            Function evaluations: 103
            Gradient evaluations: 15
```



```
predicted_mcd = price_prediction(phase1_train_set, ticker = "MCD", p = 1, q = 1, dist = '
```

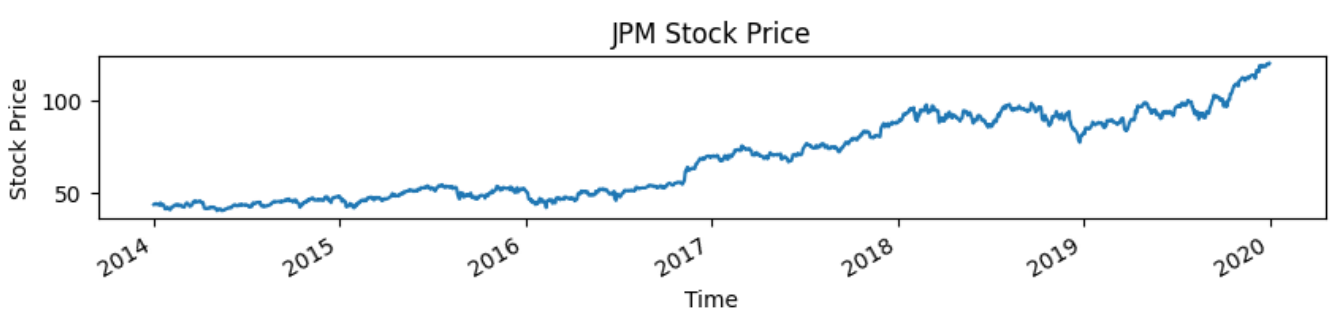
```
Iteration:      1,   Func. Count:      7,   Neg. LLF: 46191.39779881113
Iteration:      2,   Func. Count:     17,   Neg. LLF: 23308.1382965706
Iteration:      3,   Func. Count:     25,   Neg. LLF: 2140.847635119466
Iteration:      4,   Func. Count:     32,   Neg. LLF: 3219.1489690845974
Iteration:      5,   Func. Count:     39,   Neg. LLF: 2003.8900191457774
Iteration:      6,   Func. Count:     46,   Neg. LLF: 2773.3229640647
Iteration:      7,   Func. Count:     53,   Neg. LLF: 1996.6071910609478
Iteration:      8,   Func. Count:     60,   Neg. LLF: 1989.5901521164171
Iteration:      9,   Func. Count:     66,   Neg. LLF: 1989.5261282474435
Iteration:     10,   Func. Count:     73,   Neg. LLF: 1989.4157714215798
Iteration:     11,   Func. Count:     80,   Neg. LLF: 1989.3157471885734
Iteration:     12,   Func. Count:     86,   Neg. LLF: 1989.3126594514108
Iteration:     13,   Func. Count:     92,   Neg. LLF: 1989.312448570466
Iteration:     14,   Func. Count:     98,   Neg. LLF: 1989.3124436615915
Iteration:     15,   Func. Count:    103,   Neg. LLF: 1989.31244366158
Optimization terminated successfully    (Exit mode 0)
            Current function value: 1989.3124436615915
            Iterations: 15
            Function evaluations: 103
            Gradient evaluations: 15
Model Summary
                              AR - GARCH Model Results
==============================================================================
Dep. Variable:                          MCD   R-squared:                       0.000
Mean Model:                              AR   Adj. R-squared:                  0.000
Vol Model:                            GARCH   Log-Likelihood:              -1989.31
Distribution:      Standardized Student's t   AIC:                          3988.62
Method:                  Maximum Likelihood   BIC:                          4015.22
                                              No. Observations:                1509
Date:                      Tue, Oct 29 2024   Df Residuals:                    1508
Time:                             17:48:45   Df Model:                           1
                                Mean Model
============================================================================
                 coef    std err          t      P>|t|    95.0% Conf. Int.
----------------------------------------------------------------------------
Const          0.0833  2.408e-02      3.458  5.443e-04 [3.607e-02,  0.130]
                            Volatility Model
============================================================================
                 coef    std err          t      P>|t|    95.0% Conf. Int.
----------------------------------------------------------------------------
omega          0.0916      0.228      0.401      0.688   [ -0.356,  0.539]
alpha[1]       0.1167      0.177      0.659      0.510   [ -0.231,  0.464]
beta[1]        0.8047      0.371      2.172  2.986e-02 [7.856e-02,  1.531]
                              Distribution
============================================================================
                 coef    std err          t      P>|t|  95.0% Conf. Int.
----------------------------------------------------------------------------
nu             3.9700      0.423      9.393  5.853e-21 [  3.142,  4.798]
============================================================================

Covariance estimator: robust
```

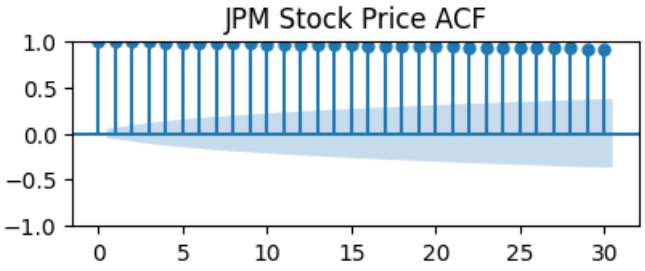




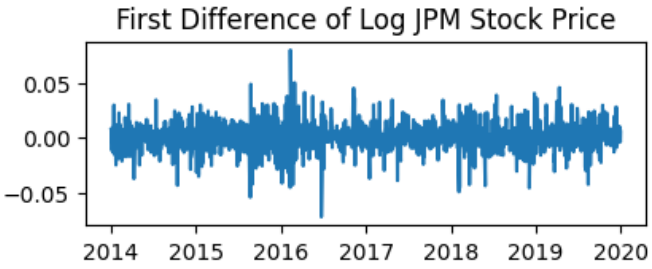
3.5.3.4 JPM Modeling Results

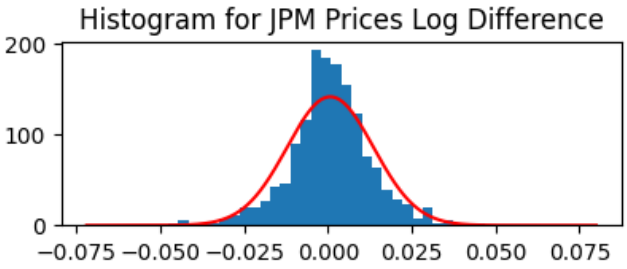
```
1 price_logdiff_acf_pacf(phase1_train_set, "JPM")
```

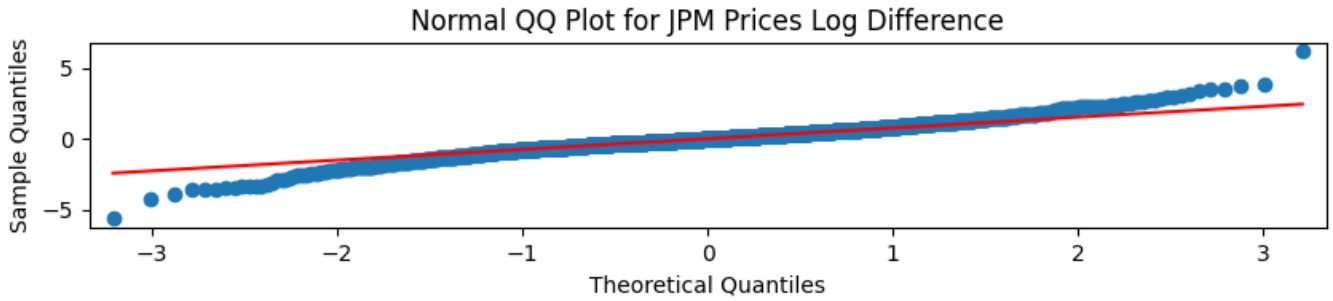

JPM Stock Price
Stock Price
100
50
2014
2015
2016
2017
2018
2019
2020
Time

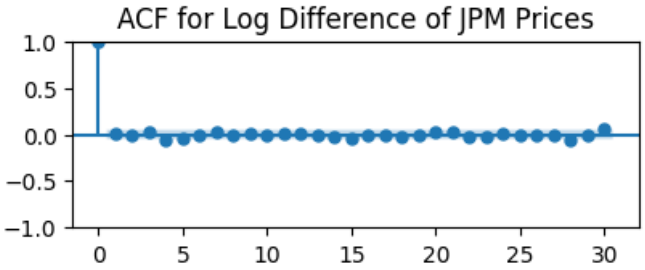

JPM Stock Price ACF
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30

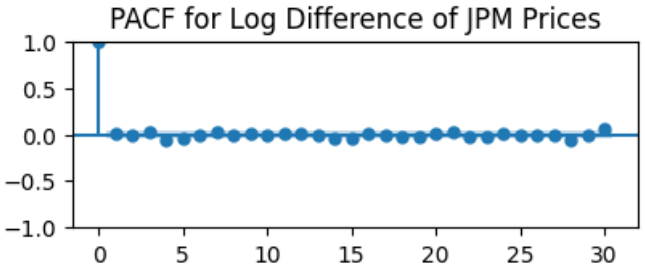

JPM Stock Price PACF
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30


First Difference of Log JPM Stock Price
0.05
0.00
−0.05
2014
2015
2016
2017
2018
2019
2020


Histogram for JPM Prices Log Difference
200
100
0
−0.075
−0.050
−0.025
0.000
0.025
0.050
0.075

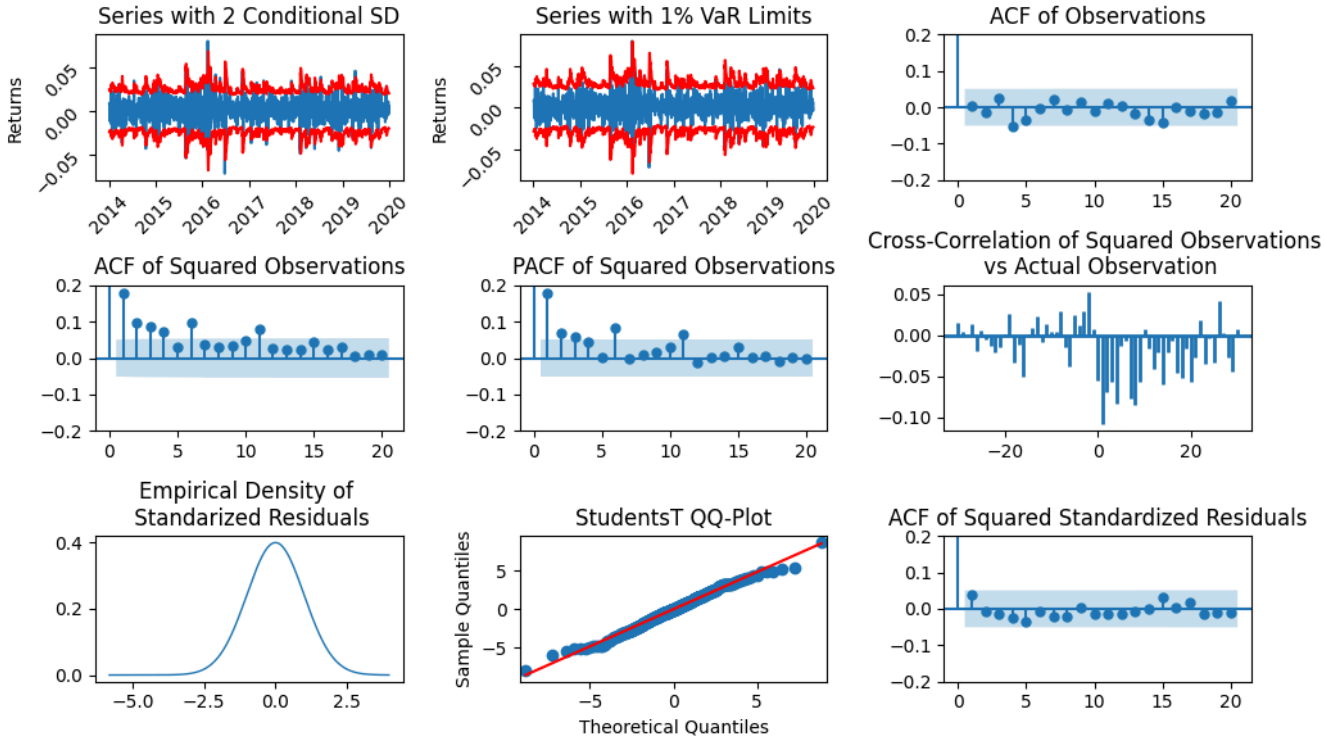

Normal QQ Plot for JPM Prices Log Difference
Sample Quantiles
5
0
−5
−3
−2
−1
0
1
2
3
Theoretical Quantiles


ACF for Log Difference of JPM Prices
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30


PACF for Log Difference of JPM Prices
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30

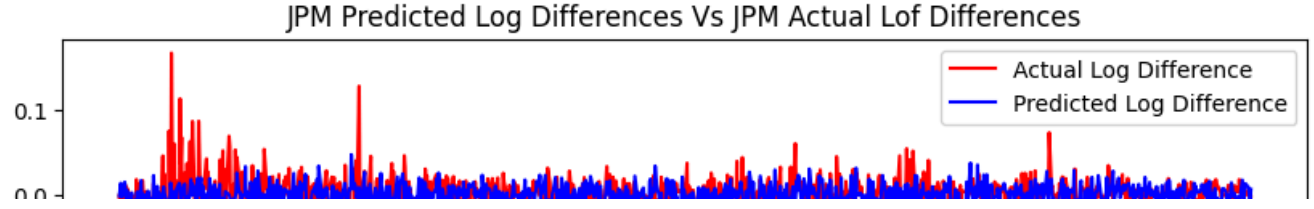
```
garch_diagnostic(data = phase1_train_set,  p = 1, q = 1, ticker = "JPM")
```

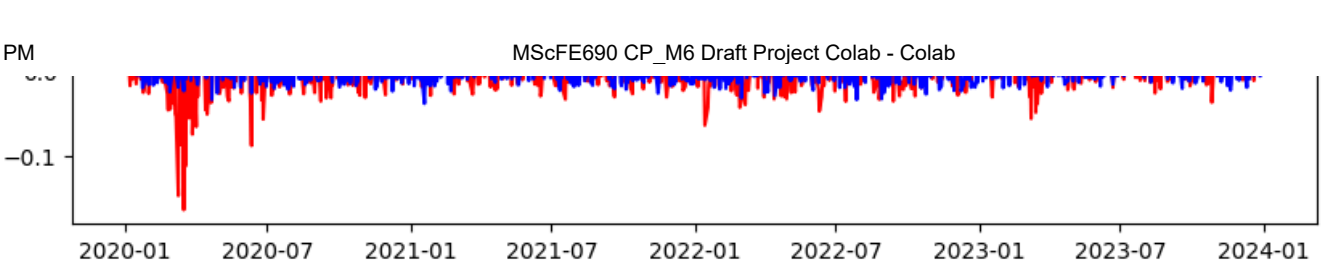
```
Iteration:      1,   Func. Count:      7,   Neg. LLF: 47595.18004773867
Iteration:      2,   Func. Count:     16,   Neg. LLF: 26406.639398726627
Iteration:      3,   Func. Count:     24,   Neg. LLF: 4284.012855192017
Iteration:      4,   Func. Count:     32,   Neg. LLF: 3691.576310680292
Iteration:      5,   Func. Count:     39,   Neg. LLF: 3712.5026808226266
Iteration:      6,   Func. Count:     46,   Neg. LLF: 3707.90700808806
Iteration:      7,   Func. Count:     53,   Neg. LLF: 2554.2455057906836
Iteration:      8,   Func. Count:     60,   Neg. LLF: 2403.508838980282
Iteration:      9,   Func. Count:     67,   Neg. LLF: 2402.2678688617466
Iteration:     10,   Func. Count:     73,   Neg. LLF: 2402.222017984779
Iteration:     11,   Func. Count:     79,   Neg. LLF: 2402.214086607139
Iteration:     12,   Func. Count:     85,   Neg. LLF: 2402.2121734205466
Iteration:     13,   Func. Count:     91,   Neg. LLF: 2402.2121229638965
Iteration:     14,   Func. Count:     96,   Neg. LLF: 2402.2121229638797
Optimization terminated successfully    (Exit mode 0)
            Current function value: 2402.2121229638965
            Iterations: 14
            Function evaluations: 96
            Gradient evaluations: 14
```



```
predicted_jpm = price_prediction(phase1_train_set, ticker = "JPM", p = 1, q = 1, dist = '
```

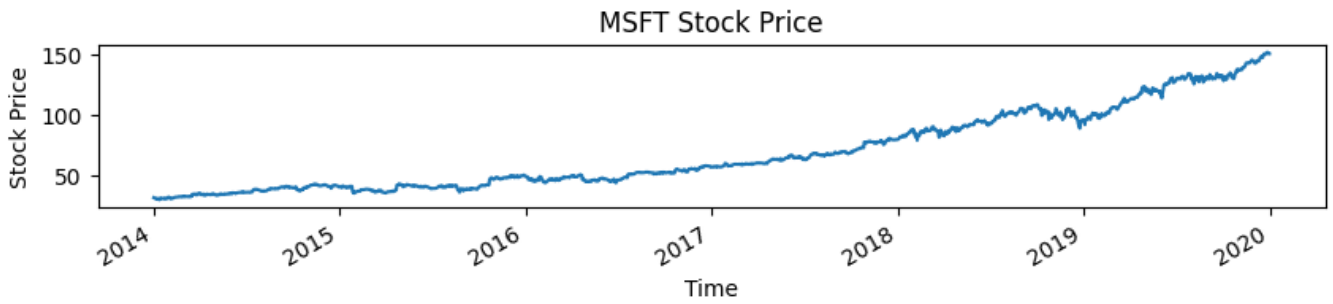
```
Iteration:      1,   Func. Count:      7,   Neg. LLF: 47595.18004773867
Iteration:      2,   Func. Count:     16,   Neg. LLF: 26406.639398726627
Iteration:      3,   Func. Count:     24,   Neg. LLF: 4284.012855192017
Iteration:      4,   Func. Count:     32,   Neg. LLF: 3691.576310680292
Iteration:      5,   Func. Count:     39,   Neg. LLF: 3712.5026808226266
Iteration:      6,   Func. Count:     46,   Neg. LLF: 3707.90700808806
Iteration:      7,   Func. Count:     53,   Neg. LLF: 2554.2455057906836
Iteration:      8,   Func. Count:     60,   Neg. LLF: 2403.508838980282
Iteration:      9,   Func. Count:     67,   Neg. LLF: 2402.2678688617466
Iteration:     10,   Func. Count:     73,   Neg. LLF: 2402.222017984779
Iteration:     11,   Func. Count:     79,   Neg. LLF: 2402.214086607139
Iteration:     12,   Func. Count:     85,   Neg. LLF: 2402.2121734205466
Iteration:     13,   Func. Count:     91,   Neg. LLF: 2402.2121229638965
Iteration:     14,   Func. Count:     96,   Neg. LLF: 2402.2121229638797
Optimization terminated successfully    (Exit mode 0)
            Current function value: 2402.2121229638965
            Iterations: 14
            Function evaluations: 96
            Gradient evaluations: 14
Model Summary
                              AR - GARCH Model Results
==============================================================================
Dep. Variable:                            JPM   R-squared:                       0.000
Mean Model:                                AR   Adj. R-squared:                  0.000
Vol Model:                              GARCH   Log-Likelihood:               -2402.21
Distribution:      Standardized Student's t   AIC:                            4814.42
Method:                  Maximum Likelihood   BIC:                            4841.02
                                              No. Observations:                  1509
Date:                      Tue, Oct 29 2024   Df Residuals:                      1508
Time:                              17:48:52   Df Model:                             1
                                 Mean Model
============================================================================
                 coef    std err          t      P>|t|    95.0% Conf. Int.
----------------------------------------------------------------------------
Const          0.0855  2.622e-02      3.262  1.106e-03 [3.414e-02,  0.137]
                            Volatility Model
============================================================================
                 coef    std err          t      P>|t|    95.0% Conf. Int.
----------------------------------------------------------------------------
omega          0.1592  7.031e-02      2.264  2.355e-02 [2.140e-02,  0.297]
alpha[1]       0.1226  3.503e-02      3.498  4.685e-04 [5.389e-02,  0.191]
beta[1]        0.7968  6.318e-02     12.610  1.862e-36   [  0.673,  0.921]
                            Distribution
============================================================================
                 coef    std err          t      P>|t|  95.0% Conf. Int.
----------------------------------------------------------------------------
nu             4.2893      0.449      9.543  1.386e-21 [  3.408,  5.170]
============================================================================

Covariance estimator: robust
```

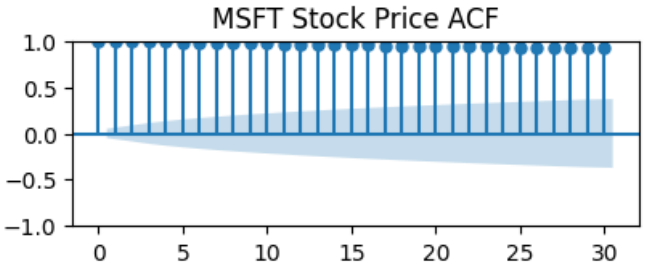


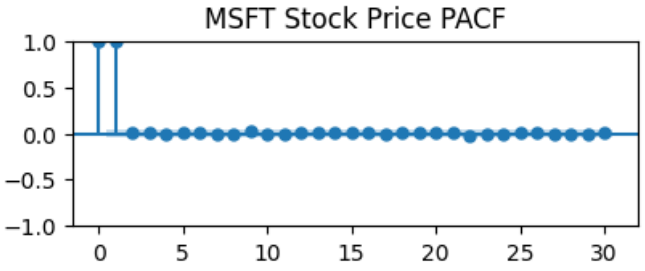


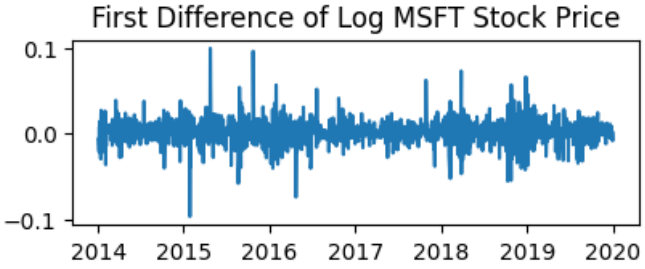
3.5.3.4 MSFT Modeling Results

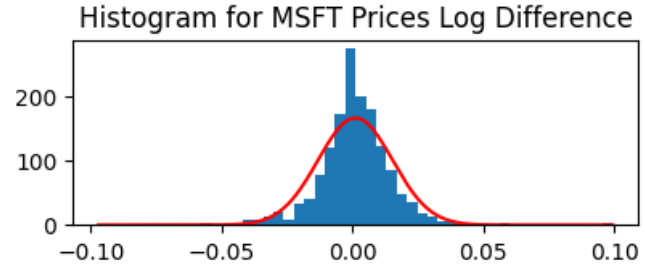
```
price_logdiff_acf_pacf(phase1_train_set, "MSFT")
```

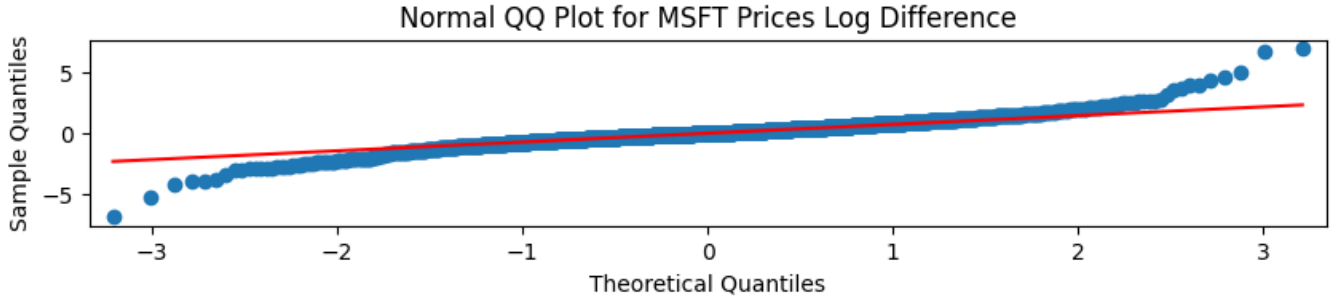

MSFT Stock Price
Stock Price
150
100
50
2014
2015
2016
2017
2018
2019
2020
Time

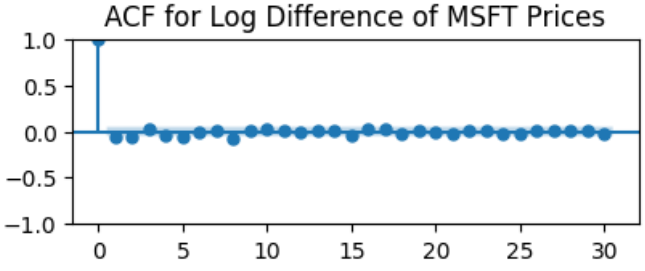

MSFT Stock Price ACF
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30

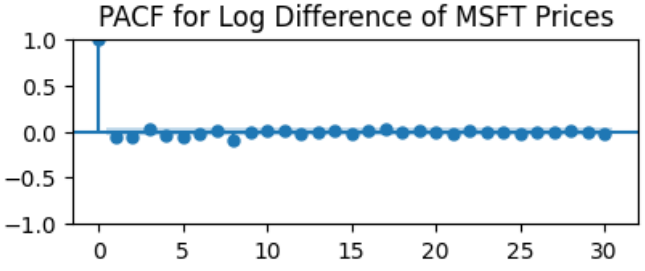

MSFT Stock Price PACF
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30


First Difference of Log MSFT Stock Price
0.1
0.0
−0.1
2014
2015
2016
2017
2018
2019
2020


Histogram for MSFT Prices Log Difference
200
100
0
−0.10
−0.05
0.00
0.05
0.10


Normal QQ Plot for MSFT Prices Log Difference
Sample Quantiles
5
0
−5
−3
−2
−1
0
1
2
3
Theoretical Quantiles


ACF for Log Difference of MSFT Prices
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30


PACF for Log Difference of MSFT Prices
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30

```
1 garch_diagnostic(data = phase1_train_set,  p = 1, q = 1, ticker = "MSFT")
```

```
Iteration:      1,   Func. Count:      7,   Neg. LLF: 41693.66022752198
Iteration:      2,   Func. Count:     16,   Neg. LLF: 24003.44636556912
Iteration:      3,   Func. Count:     24,   Neg. LLF: 4175.042882047764
Iteration:      4,   Func. Count:     31,   Neg. LLF: 3568.2804382910435
Iteration:      5,   Func. Count:     38,   Neg. LLF: 2486.9312709968644
Iteration:      6,   Func. Count:     45,   Neg. LLF: 2540.3468579147197
Iteration:      7,   Func. Count:     52,   Neg. LLF: 2466.878891159432
Iteration:      8,   Func. Count:     59,   Neg. LLF: 2463.7772642597083
Iteration:      9,   Func. Count:     65,   Neg. LLF: 2463.7584906152624
Iteration:     10,   Func. Count:     71,   Neg. LLF: 2463.758077687742
Iteration:     11,   Func. Count:     77,   Neg. LLF: 2463.7580479664566
Iteration:     12,   Func. Count:     83,   Neg. LLF: 2463.758044961342
Iteration:     13,   Func. Count:     88,   Neg. LLF: 2463.7580449613297
Optimization terminated successfully    (Exit mode 0)
            Current function value: 2463.758044961342
            Iterations: 13
            Function evaluations: 88
            Gradient evaluations: 13
```