Introduction to Financial Econometrics

Empirical Analysis – Forecasting 30 Day returns of the SP500 using the ARIMA model

Student: Immanuel Berger

Student ID: 515532

Overview

I will be investigating the SP500 Returns over time and predict a 30-day forecast using the best-fit ARIMA model. This is done with the help of econometric analysis using Python to process the data, as well as using Yahoo Finance as the source of historical data. **To bypass any bias within fitting** the ARIMA model I ensured that I only fit the ARIMA model to a **training data** and then evaluate the models forecasted values based on the **test data** (Actual values) .

ARIMA model

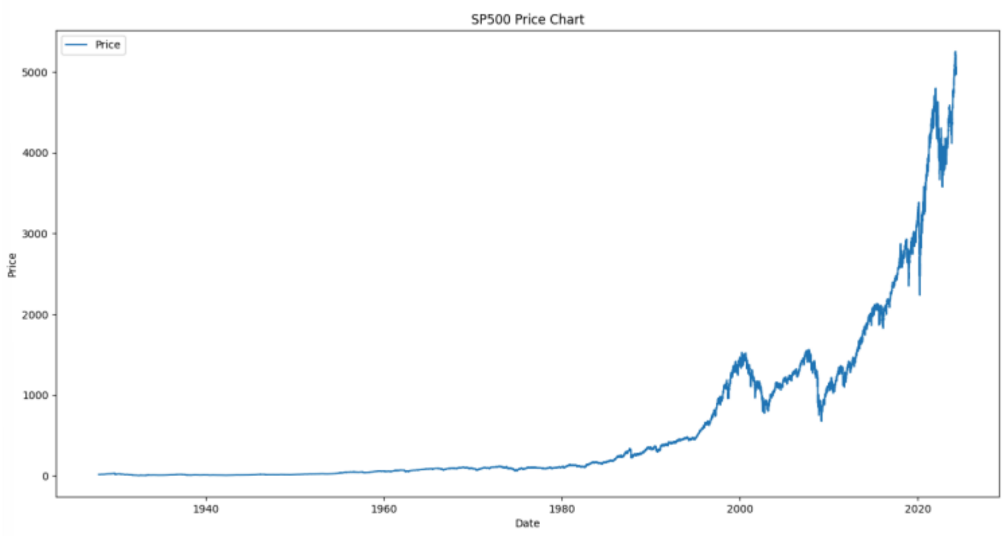
A ARIMA model is a popular statistical/econometric model used for analyzing and forecasting time series data. It is composed of three components:

- AR, also known as autoregressive, which is denoted by the parameter *p*, which predicts the future value of its data based on its past value by regressing its own prior lagged values.
- I, presenting the term Integrated, denoted as *d*, provides an overview of how much data is differentiated, implying removing trends and seasonality to achieve a stationary time series.
- MA, also known as moving average, that is denoted as q, computes the relationship between an observed value and the residual error on lagged observations.

Through the processing of data and relevant statistical analysis the best-fitted parameters ARIMA(*p,d,q)*, are found to forecast 30 days of the log returns of the SP500.

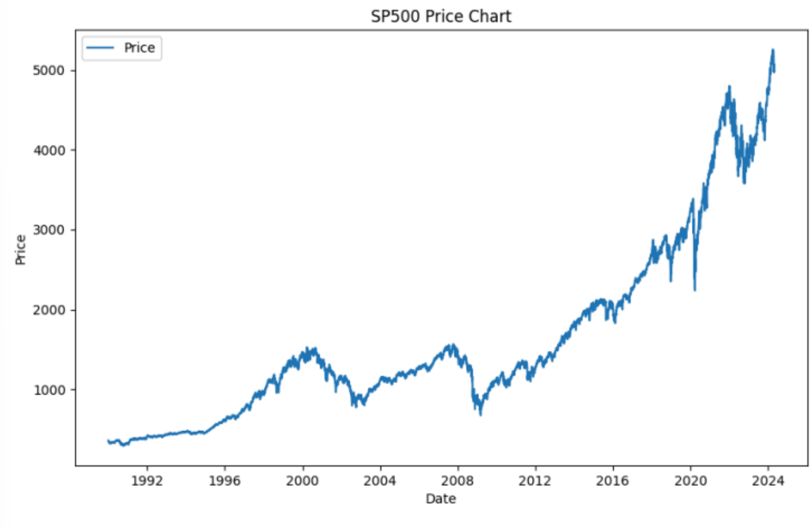
Processing of Historical Data

SP500 Historical Price Chart



At first, I took a brief overview of the historical price action of the SP500 and concluded that removing the data between 1930-1980 would provide a more appropriate modeling fit, to encapsulate better actual current market dynamics.

SP500 Historical Price Chart (1980-Present)



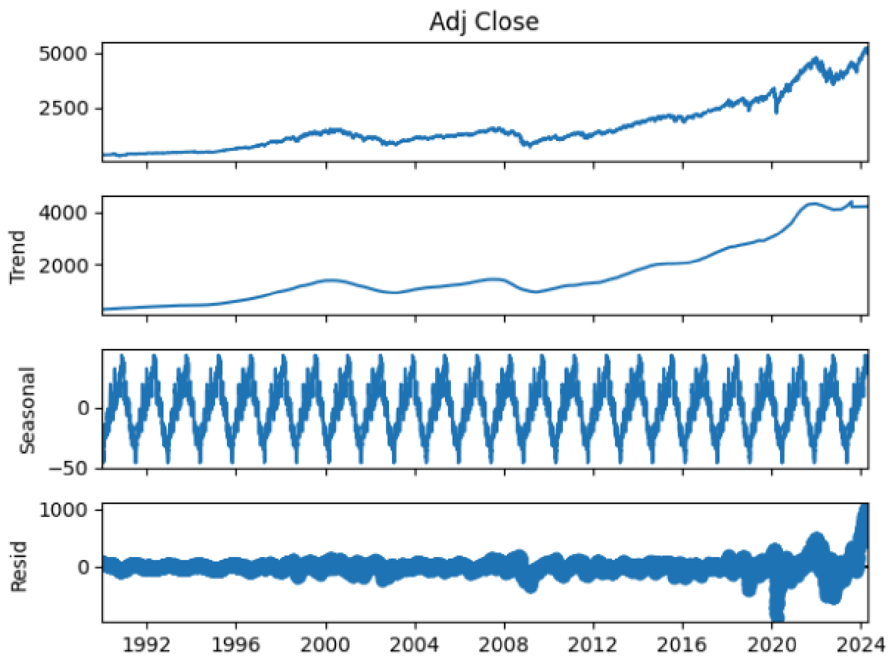
Basic Descriptive Statistics (1980-Present)

count	8643.000000
mean	1614.918244
std	1132.355586
min	295.459991
25%	900.875000
50%	1272.869995
75%	2068.209961
max	5254.350098

Seasonal Decomposition of SP500

To investigate further the dynamics of the data I seasonally decomposed the data to Trend, Seasonality and Residuals. This is done to identify any seasonality within the data, as well as see if time-series data is stationary or not to further improve the forecasting accuracy to identify how to approach data-processing to fit the ARIMA model.

Seasonal Decomposition



Trend: Overall trend is positive and increasing with minor deviations

Seasonality: Seasonality pattern is present within the data with a full cycle approximately lasting every 2 years

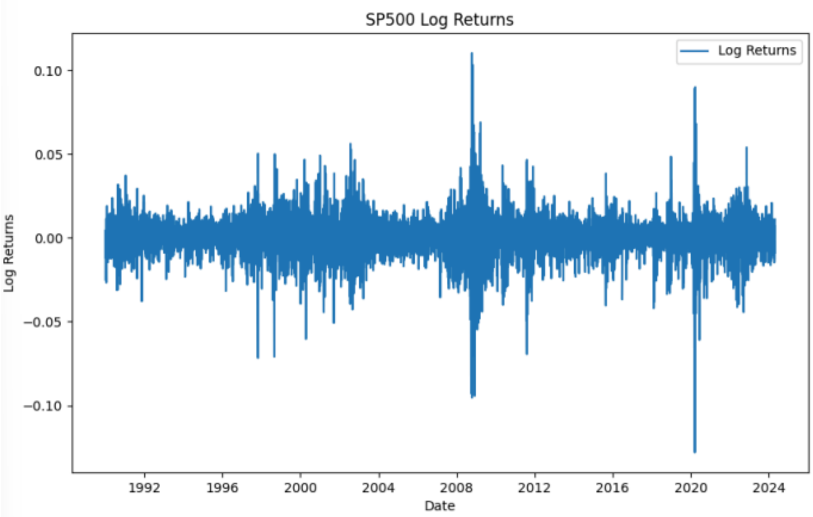
Residuals: Show major deviations from 2020-2024, while staying rather stable during the times before

From the seasonal decomposition to explore underlying patterns the price chart has revealed clear evidence of seasonality and residuals deviating out both to the upside and downside implying possible anomalies or outliers within the data. As well as non-stationary assumptions can be made since statistical properties, such as mean and variance change over time, based on visual interpretation.

Compute Logarithmic Returns

Since the aim of this empirical analysis is to forecast the returns of the sp500 I computed not only normal returns, but the logarithmic returns, since this is a common method to ensure stationarity and tends to provide a better fit to model ARIMA. Logarithmic returns mimic closer a stationary time series data. That's why usually returns especially also in this case logarithmic returns are forecasted instead of prices, since they provide a more consistent statistical properties to model.

SP500 Log Returns



Basic Descriptive Statistics

```
count    8642.000000
mean        7.147637
std         0.699112
min         5.669035
25%         6.795845
50%         7.143414
75%         7.630817
max         8.565180
Name: Log Returns, dtype: float64
```

Based on the graph the data looks more stationary and provides a more interpretable scale for changes in value and a relative reduced sensitivity to extreme values.

Augmented Dickey Fuller Test (Stationarity Test)

To ensure stationarity of the data the ADF test also known as the Augmented Dickey Fuller Test is used which tests for the following hypothesis:

H0: Data = Non-stationary/Unit Root **H1: Data ≠ non-stationary/unit root**

The p-value is used in this case to interpret, whether to reject/accept the null hypothesis. The p-value reports the smallest significance level that leads to a rejection. In this case the critical region of 0.05 is used, implying that if the ADF test of the test-stationary data results in a p-value lower than 0.05, the null hypothesis is rejected, meaning that the data is stationary. The Integrated component (*d*) is

I first computed the ADF test on the SP500 logarithmic returns without any order of differencing:

ADF on SP500 Logarithmic Returns (d=0)

```
1. ADF :  -0.6095522210071702
2. P-Value :  0.868808974930416
3. Num Of Lags :  34
4. Num Of Observations Used For ADF Regression and Critical Values Calculation
: 8607
5. Critical Values :
	1% :  -3.4311099920233863
	5% :  -2.8618758652800826
	10% :  -2.566948776154556
```

p-value = 0.8688 > 0.05

Since the p-value is above the rejecting value of 0.05 I cannot reject the null hypothesis meaning that the data is non-stationary, and I need change the order of differencing to get a stationary time series data. After computing the logarithmic returns of the SP500 with first order of differencing (d=1) I got the following results:

ADF Test on SP500 Logarithmic Returns with First order of Differencing (d=1)

```
1. ADF :  -17.031821567461638
2. P-Value :  8.314721454703682e-30
3. Num Of Lags :  33
4. Num Of Observations Used For ADF Regression and Critical Values Calculation
: 8607
5. Critical Values :
        1% :  -3.4311099920233863
        5% :  -2.8618758652800826
        10% :  -2.566948776154556
```

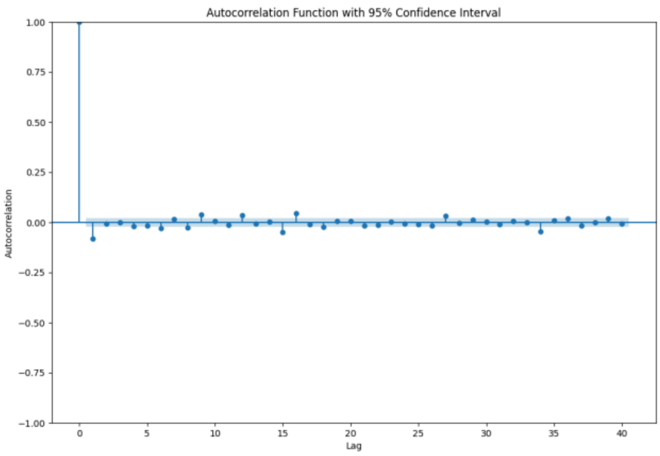
p-value = 8.3147e-30 < 0.05

Since the p-value is below the rejection we can reject the null hypothesis and therefore conclude that the data is stationary to fit an ARIMA model with first order of differencing (d=1). This would imply an ARIMA model with d = 1, still having to find the parameters for p and q.

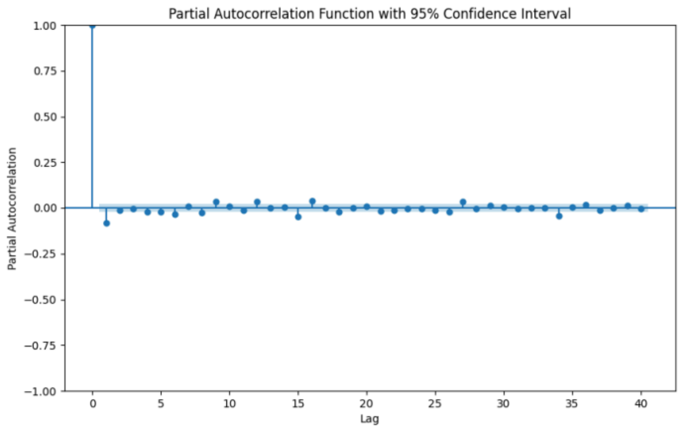
ACF and PACF Plots

To investigate further the time-series data the ACF Plot showing autocorrelation of the series with its lags as well as PACF plot that shows partial correlation of the series with its lags are used to find any autocorrelation within the time-series data as well as any more seasonality or trend in data and interpret possibly how the MA component, looking at the ACF plot, and AR component, looking at the PACF plot, can be interpreted.

ACF Plot



PACF Plot



From both the ACF and PACF plot I was able to conclude that there are minor deviations above the confidence interval of 95% after lag 0 implying a stationary time series with minimal seasonality. Since it is usually a more challenging approach to base AR and MA orders based on visual interpretation of the ACF and PACF since there is no clear pattern within the correlation coefficients I will be using Python that provides automation methods for model selection and parameter estimation, ensuring greater accuracy in estimating an ARIMA model for forecasting.

Parameter Estimation

Now since the data is processed to fit an ARIMA model I used a grid parameter estimation method to find the best parameters ranging for each parameter from 0-3. I fitted the training data to models with different parameters and determined their measure of goodness of fit based on the AIC and BIC. The AIC, also known as Akaike Information Criterion, which is a statistical measure that evaluates the quality of a given set of data and its goodness of fit. BIC, also known as Bayesian Information Criterion, does the same evaluations, like the AIC, but penalizes more complex models reducing any overfitting. The lower the AIC and BIC is, the better the fit of the model.

Grid Parameter Estimation test for ARIMA (0-3, 0-3, 0-3):

```
Parameters: (1, 1, 1) AIC: -52498.428083179 BIC : -52477.24534760355
Parameters: (1, 1, 2) AIC: -52499.094028239306 BIC : -52470.85038080537
Parameters: (1, 2, 1) AIC: -52483.38192815294 BIC : -52462.19954094885
Parameters: (2, 1, 1) AIC: -52496.3275963396 BIC : -52468.08394890567
Parameters: (2, 2, 1) AIC: -52483.09138238907 BIC : -52454.84819945029
Parameters: (1, 2, 2) AIC: -52426.422936838106 BIC : -52398.17975389932
Parameters: (2, 1, 2) AIC: -52498.65905302033 BIC : -52463.35449372791
Parameters: (2, 2, 2) AIC: -52472.52087009975 BIC : -52437.216891426266
Parameters: (1, 1, 3) AIC: -52494.602132528984 BIC : -52459.29757323656
Parameters: (1, 3, 1) AIC: -48667.795221265595 BIC : -48646.61318247333
Parameters: (3, 1, 1) AIC: -52494.303190916726 BIC : -52458.9986316243
Parameters: (1, 2, 3) AIC: -52483.37837053227 BIC : -52448.07439185879
Parameters: (2, 1, 3) AIC: -52492.22115972352 BIC : -52449.855688572614
Parameters: (2, 2, 3) AIC: -52469.24933104872 BIC : -52426.8845564054
Parameters: (1, 3, 1) AIC: -48667.795221265595 BIC : -48646.61318247333
Parameters: (1, 3, 2) AIC: -52391.783775342 BIC : -52363.541056952316
Parameters: (2, 3, 1) AIC: -49827.01046193038 BIC : -49798.767743540695
Parameters: (3, 1, 2) AIC: -52492.21667936981 BIC : -52449.851208218905
Parameters: (3, 2, 1) AIC: -52459.98698850088 BIC : -52424.6830098274
Parameters: (3, 2, 2) AIC: -52447.072515771004 BIC : -52404.7077413628З
Parameters: (1, 3, 3) AIC: -51932.03652483568 BIC : -51896.73312684857
Parameters: (3, 3, 1) AIC: -50323.995829398 BIC : -50288.69243141089
Parameters: (3, 1, 3) AIC: -52490.21561120897 BIC : -52440.78922819958
Parameters: (2, 3, 3) AIC: -52434.86440795286 BIC : -52392.500330368326
Parameters: (3, 2, 3) AIC: -52456.858718622505 BIC : -52407.43314847963
Parameters: (3, 3, 2) AIC: -49778.00047842631 BIC : -49735.63640084177
Parameters: (3, 3, 3) AIC: -49019.83194447708 BIC : -48970.40718729513
```

Out of the following computation, I was able to conclude that the best parameter estimated based on the AIC and BIC and the range of 0-3 on each parameter is: ARIMA (1,1,1) AIC: -52498 BIC:52477

Model Summary of ARIMA(1,1,1)

```
                               SARIMAX Results
==============================================================================
Dep. Variable:            Log Returns   No. Observations:                 8613
Model:                 ARIMA(1, 1, 1)   Log Likelihood               26252.214
Date:                Wed, 24 Apr 2024   AIC                         -52498.428
Time:                        17:06:58   BIC                         -52477.245
Sample:                             0   HQIC                        -52491.204
                               - 8613
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.0843      0.052      1.606      0.108      -0.019       0.187
ma.L1         -0.1654      0.052     -3.162      0.002      -0.268      -0.063
sigma2         0.0001   8.37e-07    157.346      0.000       0.000       0.000
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):             37270.48
Prob(Q):                              0.93   Prob(JB):                         0.00
Heteroskedasticity (H):               1.19   Skew:                            -0.50
Prob(H) (two-sided):                  0.00   Kurtosis:                        13.14
===================================================================================
```

Log Likelihood: Shows how well the parameters explain observed data. A high value usually indicates a good fit for the model, which seems to be the case with these parameters.

AIC and BIC: From the parameter estimation this model had the lowest AIC and BIC indicating from the parameters used the best trade-off between goodness of fit and simplicity of model.

Ljung Box Test(L1) (p): Looks at whether data is independently distributed or serially correlated. A p-value above the critical region would accept the null hypothesis implying data is independently distributed. The p-value is 0.93 > 0.05. This means that we cannot reject the Null hypothesis. This implies that, there is no evidence for serial correlation within data.

Heteroskedasticity (p): With a p-value of 0 < 0.05 we reject the H0 indicating that there is heteroskedasticity within data

Skew: Tests for asymmetry of the distribution. If its skewed to the lest meaning that the value is below 0 would mean that the distribution is skewed to the left, while if the value is above 0 it would mean that the distribution it skewed to the right. Distribution is skewed given a value of -5 to the left.

Kurtosis: Assesses the tails of a distribution. Usually when the value of Kurtosis is greater than 3 the distribution has fat tails(extremes), which can be explained by the complexities/volatilities in financial markets, such as the occurrence of any extreme events influencing financial markets

Finding the Best-fit Parameter

By computing the auto_arima function using a library in python I found the best fitting ARIMA model for the training data, which resulted in the ARIMA model with parameters (0, 1, 5). In order to ensure that this is a better fit than the first proposed parameters I provided a statistical summary of the model on the data and compared it to the previous research.

Model Summary of ARIMA (0,1,5)

```
                               SARIMAX Results
==============================================================================
Dep. Variable:            Log Returns   No. Observations:                 8612
Model:                 ARIMA(0, 1, 5)   Log Likelihood               26253.140
Date:                Tue, 23 Apr 2024   AIC                         -52494.280
Time:                        23:30:34   BIC                         -52451.915
Sample:                             0   HQIC                        -52479.833
                               - 8612
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.0810      0.006    -14.667      0.000      -0.092      -0.070
ma.L2         -0.0087      0.004     -1.962      0.050      -0.017   -7.76e-06
ma.L3         -0.0032      0.006     -0.567      0.571      -0.014       0.008
ma.L4         -0.0202      0.005     -3.890      0.000      -0.030      -0.010
ma.L5         -0.0181      0.005     -3.465      0.001      -0.028      -0.008
sigma2         0.0001   8.56e-07    153.692      0.000       0.000       0.000
===================================================================================
Ljung-Box (L1) (Q):                   0.04   Jarque-Bera (JB):             37277.81
Prob(Q):                              0.84   Prob(JB):                         0.00
Heteroskedasticity (H):               1.19   Skew:                            -0.54
Prob(H) (two-sided):                  0.00   Kurtosis:                        13.13
===================================================================================
```

Log Likelihood: With these parameters the log-likelihood value is even higher than the first proposed model fit concluding a better fit

AIC and BIC: AIC and BIC values are also lower meaning that these parameters are a better fit

Ljung-Box Test(L1) (p) p-value is still very close to 1 implying no evidence for serial correlation

Heteroskedasticity (p): Same heteroskedasticity p-value indicating no heteroskedasticity

Skew: This distribution is skewed a bit more to the left in comparison to the other model

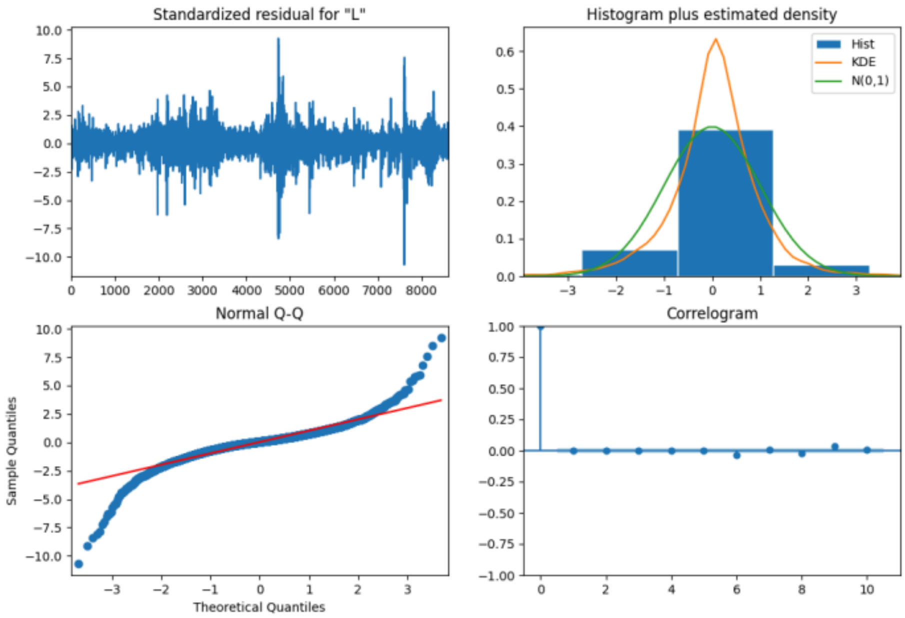
Kurt: Has the same values of the first proposed parameters, indicating heavy tails within

Overall: Proposes a better fit, if not the most-suited parameters to use for ARIMA modeling this training data to forecast.

Residual Analysis on ARIMA(0,1,5)

Before forecasting using the ARIMA model to further strengthen and acknowledge a good-fit model residuals must be analyzed to ensure that the underlying patterns and dynamics of the data are still encapsulated and if there are any possible model improvements to be made. Analyzing residuals ensures validity and reliability and accuracy of the ARIMA model by detecting any misspecifications.

Residual Analysis ARIMA (0,1,5)



Standardized Residuals: Act very similar to white noise showing randomness and no serial correlation.

Histogram: Looks very similar to a normal distribution

Normal Q-Q: Follows the straight line, however, deviates on both ends confirming heavier tails within the distribution.

Correlogram: Minor deviations above confidence intervals confirming stationarity.

Ljung-Box Test

The Ljung-Box test with 20 lags to test is used to further acknowledge a good-fit based on residual analysis. The Ljung-Box test essentially tests if data is independently distributed or serially correlated. This is done by testing the following hypothesis:

H0: Residuals = Independently Distributed **H1: Residuals ≠ Independently Distributed**

Ljung-Box Test summary (20 lags, lb_stat = Ljung-Box test value, lb_pvalue = p-value)

	lb_stat	lb_pvalue
1	0.021497	0.883433
2	0.049856	0.975380
3	0.052597	0.996842
4	0.090288	0.999011
5	0.106315	0.999811
6	0.109439	0.999974
7	0.240672	0.999953
8	0.297678	0.999982
9	0.363220	0.999992
10	0.380559	0.999998
11	0.380803	1.000000
12	0.399104	1.000000
13	0.576521	1.000000
14	0.576891	1.000000
15	0.619841	1.000000
16	0.627704	1.000000
17	0.632330	1.000000
18	0.646003	1.000000
19	0.656297	1.000000
20	0.738970	1.000000

Using the critical region of 0.05 we can conclude that the residuals are independently distributed. Most of the lags' p-value is closer to 1 and above 0.05, suggesting that residuals act accordingly to white noise, since we fail to reject the null hypothesis.

Model Evaluation

Statistical Evaluation

To evaluate the model, I computed the Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE) to see any significance within errors of the model. The Mean Absolute Error evaluates the average absolute differences between the forecasted values and the actual values in this case the forecasted log returns to the actual log returns. The Root Mean Squared Error is also looked at, which is the square root of the squared differences between the forecasted values and actual values decreasing the significance of any large errors/outliers. A base model is also used to compare MAE and RMSE values. In this case, the first proposed ARIMA(1,1,1) model is used and compared to the new ARIMA(0,1,5) model.

Base Model: ARIMA(1,1,1)

```
MAE: 0.013038768442033976
RMSE: 0.016170888538900093
```

ARIMA (0,1,5)

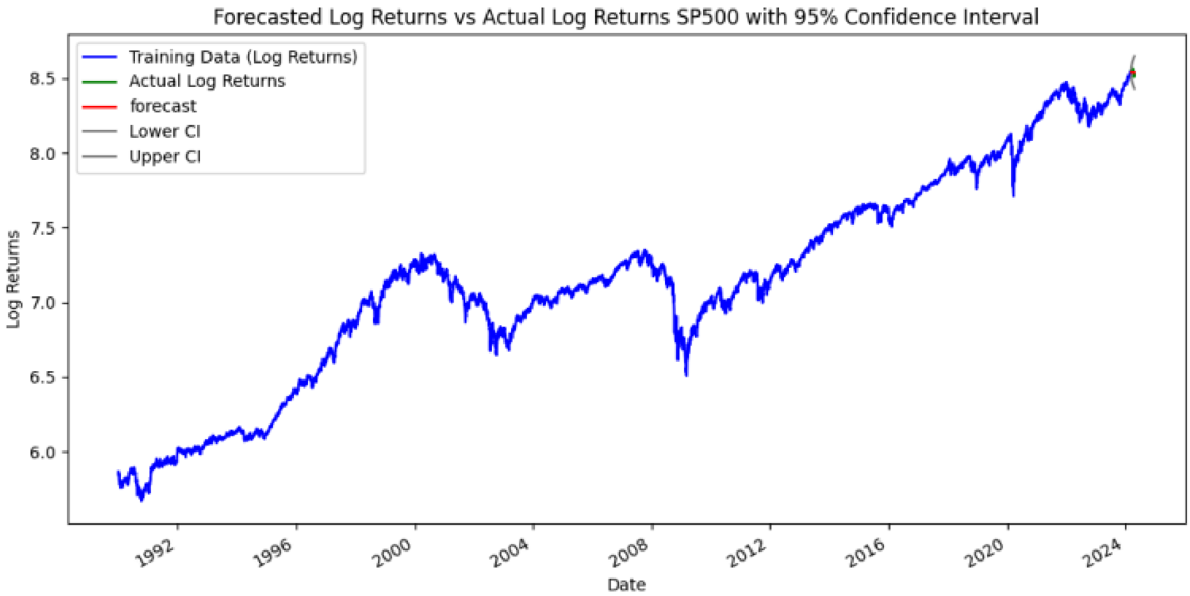
```
MAE: 0.012440100102111865
RMSE: 0.015804848351589417
```

Comparing the Base model with the ARIMA (0,1,5) model, we can conclude that ARIMA(0,1,5) provides a more accurate and valid forecasts with a lower MAE and RMSE.

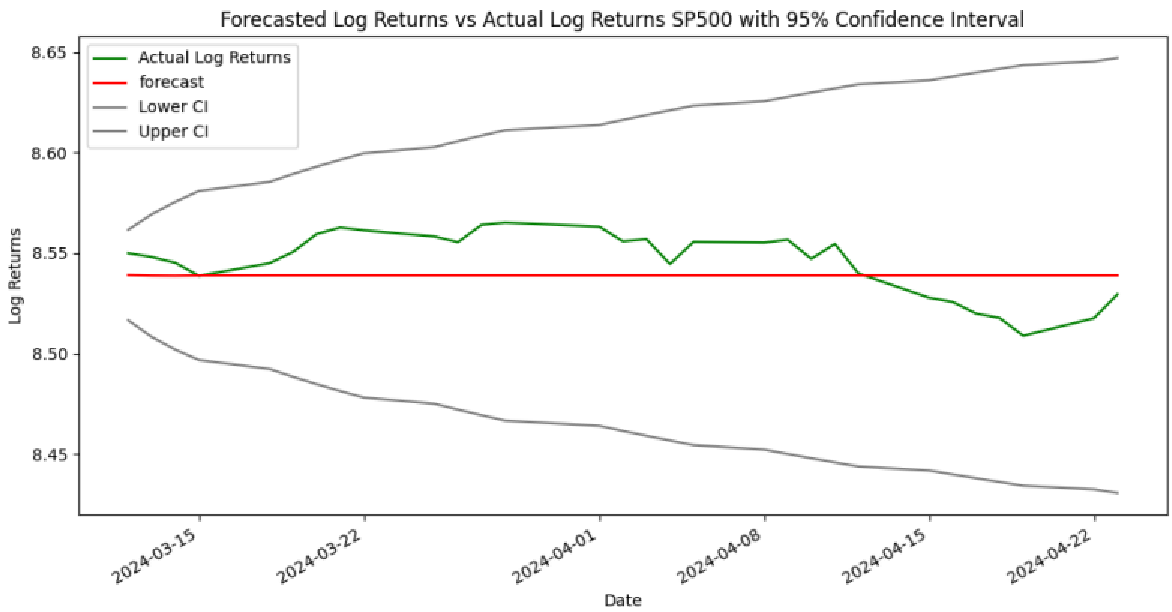
Visual Evaluation

I forecasted 30 days of logarithmic returns with 95% confidence intervals to provide a confidence range of values in which the population parameter lies, using the ARIMA (0,1,5) model and compared them to the actual log returns:

Forecast



(Zoomed in)



The forecasted log returns followed a relatively similar direction to the Actual Log Returns, while the actual log returns had minor deviations above or below the forecasted line they followed the line very accurately and never deviated above or below the confidence intervals meaning that the model fit very well within a 30-day forecast.

Conclusion

The best fit ARIMA model for the returns of the SP500 was ARIMA (0,1,5). With extensive analysis and investigation, I was able to conclude the best-fit model and provide relatively accurate results. Nonetheless there are significant challenges posed especially in modeling and forecasting financial markets, due to their nonlinearity or fat tails that can come from extreme events like any black swan events making it very difficult to incorporate that within the model.

Appendix

https://github.com/JVNI/ARIMA/blob/main/EmpiricalAnalysis/main.py

```
import pandas as pd
import numpy as np
import yfinance as yf
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.stats.diagnostic import acorr_ljungbox
from sklearn.metrics import mean_absolute_error, mean_squared_error
import pmdarima as pm
import warnings

#Download and Process Historical Data to Log Returns
sp500 = yf.download('^GSPC', start="1990-01-01", end="2024-04-25")
sp500['Returns'] = (sp500['Adj Close'] - sp500['Adj Close'].shift(1))  /
sp500['Adj Close']
print(sp500['Returns'])
sp500['Log Returns'] = np.log(sp500['Adj Close'] - np.log(sp500['Adj
Close'].shift(1)))
sp500['Log Returns'].dropna(inplace=True)
sp500.dropna(inplace=True)
```

```
# Plotting Price Function
def plot_price(ticker):
    plt.figure(figsize=(10,6))
    plt.plot(ticker.index, ticker['Returns'], label='Price')
    plt.ylabel('Returns')
    plt.xlabel('Date')
    plt.title('SP500 Price Chart')
    plt.legend()
    plt.show()

#Plotting Logarithmic Returns
def plot_logs(ticker):
    plt.figure(figsize=(10,6))
    plt.plot(ticker.index, ticker['Log Returns'], label='Log Returns')
    plt.ylabel('Log Returns')
    plt.xlabel('Date')
    plt.title('SP500 Log Returns')
    plt.legend()
    plt.show()

#Seasonal Decomposition of Data
def decompose_data(ticker):
    decompose_data = seasonal_decompose(ticker['Log Returns'].dropna(), model
= 'additive', period=365, extrapolate_trend='freq')
    daily_frequency = ticker.asfreq(freq='D')
    decompose_data.plot()
    plt.show()

# ADF Test and Test Results
def stationarity_test(ticker):
    ticker['Log Returns'] = ticker['Log Returns'].diff().diff()
    dftest = adfuller(ticker['Log Returns'].dropna(), autolag = 'AIC')
    print("1. ADF : ", dftest[0])
    print("2. P-Value : ", dftest[1])
    print("3. Num Of Lags : ", dftest[2])
    print("4. Num Of Observations Used For ADF Regression and Critical Values
Calculation :", dftest[3])
    print("5. Critical Values :")
    for key, val in dftest[4].items():
        print("\t",key, ": ", val)

#Plotting ACF
def acf_plot(ticker):
    fig, ax = plt.subplots(figsize=(12,8))
    plot_acf(ticker['Log Returns'].diff().dropna(), lags=40, alpha=0.05,
ax=ax)
    ax.set_title('Autocorrelation Function with 95% Confidence Interval')
    ax.set_xlabel('Lag')
    ax.set_ylabel('Autocorrelation')
```

```
    plt.show()

#Plotting PACF
def pacf_plot(ticker):
    fig, ax = plt.subplots(figsize=(10, 6))
    plot_pacf(ticker['Log Returns'].diff().dropna(), lags=40, alpha=0.05,
ax=ax)
    ax.set_title('Partial Autocorrelation Function with 95% Confidence
Interval')
    ax.set_xlabel('Lag')
    ax.set_ylabel('Partial Autocorrelation')
    plt.show()

#ARIMA Modeling and Plotting forecast
def arima_model(ticker, train, test, params):
    model = ARIMA(train, order=params)
    model_fit = model.fit()
    ticker['forecast log returns'] = model_fit.predict(start=8613, end=8643)
    forecast = model_fit.get_forecast(steps=30)
    ci = forecast.conf_int(alpha=0.05)
    plt.figure(figsize=(10, 6))
    plt.plot(ticker.index, ticker['Log Returns'], label='Log Returns')
    plt.plot(test.index, test, label='Actual Log Returns')
    plt.plot(ci.iloc[:, 0], label='Lower CI', color='gray')
    plt.plot(ci.iloc[:, 1], label='Upper CI', color='gray')
    plt.plot(ticker['forecast log returns'], label='Forecasted Log Returns')
    plt.fill_between(ci.index, ci.iloc[:, 0], ci.iloc[:, 1], color='gray',
alpha=0.25)
    plt.title('ARIMA Forecasted Log Returns vs Actual Log Returns with 95%
Confidence Interval')
    plt.legend()
    plt.xlabel('Date')
    plt.ylabel('Log Returns')
    plt.show()

train_data, test_data = sp500['Log Returns'][:8613], sp500['Log
Returns'][8613:]
train_data.dropna(inplace=True)
test_data.dropna(inplace=True)


def model_arima(ticker, train, test, params):
    model = ARIMA(train, order=params)
    model_fit = model.fit()
    forecast_series = model_fit.forecast(30, alpha=0.05)
    forecast = model_fit.get_forecast(30)
    ci = forecast.conf_int(alpha=0.05)

    plt.figure(figsize=(12,6))
```

```
    train.plot(color='blue', label='Training Data (Log Returns)')
    test.plot(color='green', label='Actual Log Returns')
    plt.plot(test.index, forecast_series, label='forecast', color='red')
    plt.plot(test.index, ci.iloc[:, 0], label='Lower CI', color='gray')
    plt.plot(test.index, ci.iloc[:, 1], label='Upper CI', color='gray')
    plt.legend()
    plt.xlabel('Date')
    plt.ylabel('Log Returns')
    plt.title('Forecasted Log Returns vs Actual Log Returns SP500 with 95%
Confidence Interval')
    plt.show()

# Close-up of Just Forecasts and Actual Log Returns
def model_close_arima(ticker, train, test, params):
    model = ARIMA(train, order=params)
    model_fit = model.fit()
    forecast = model_fit.get_forecast(30)
    ci = forecast.conf_int(alpha=0.05)
    forecast_series = model_fit.forecast(30, alpha=0.05)
    plt.figure(figsize=(12,6))
    test.plot(color='green', label='Actual Log Returns')
    plt.plot(test.index, forecast_series, label='forecast', color='red')
    plt.plot(test.index, ci.iloc[:, 0], label='Lower CI', color='gray')
    plt.plot(test.index, ci.iloc[:, 1], label='Upper CI', color='gray')
    plt.legend()
    plt.xlabel('Date')
    plt.ylabel('Log Returns')
    plt.title('Forecasted Log Returns vs Actual Log Returns SP500 with 95%
Confidence Interval')
    plt.show()

# Residual Analysis with Ljung-Box Test and Test Results
def arima_diagnostics(ticker, train, test, params):
    arima_model = ARIMA(train, order=params)
    arima_model_fit = arima_model.fit()
    arima_model_fit.plot_diagnostics(figsize=(12, 8))
    plt.show()
    residuals = arima_model_fit.resid
    Btest = acorr_ljungbox(residuals, lags=20, return_df=True)

    print(Btest)

# MAE and RMSE Test
def mae_rmse(train, test, params):
    model = ARIMA(train, order=params)
    model_fit = model.fit()
    forecast = model_fit.get_forecast(30)
    rmse = mean_squared_error(test, forecast.predicted_mean, squared=False)
    mae = mean_absolute_error(test, forecast.predicted_mean)
```

```
    print(f'MAE: {mae}\nRMSE: {rmse}')


mae_rmse(train_data, test_data, (1,1,1))

# param_grid =  [(1,1,1), (1,1,2), (1,2,1), (2,1,1), (2,2,1), (1,2,2),
(2,1,2), (2,2,2), (1,1,3), (1,3,1), (3,1,1), (1,2,3), (2,1,3), (2,2,3),
(1,3,1), (1,3,2), (2,3,1), (3,1,2), (3,2,1), (3,2,2), (1,3,3),(3,3,1),
(3,1,3), (2,3,3), (3,2,3), (3,3,2), (3,3,3)]
# for params in param_grid:
#     warnings.filterwarnings('ignore')
#     model = ARIMA(train_data, order=params)
#     predictions = model.fit()
#     print(f'Parameters: {params} AIC: {predictions.aic} BIC :
{predictions.bic}')


# model = ARIMA(train_data, order=(1,1,1))
# model_fit = model.fit()
# print(model_fit.summary())


# mae_rmse(train_data, test_data, (0,1,5))

# model_close_arima(sp500, train_data, test_data, (0,1,5))
# model_arima(sp500, train_data, test_data, (0,1,5))
# arima_diagnostics(sp500, train_data, test_data, (0,1,5))

# model_arima(sp500, train_data, test_data, (0,1,5))
# model_arima_forecast(sp500, (0,1,5))
```