Timeseries Forecasting In Financial Services Industry Using Machine Learning Approaches

1 EXECUTIVE SUMMARY

COMPANY is a share registrar for 30% of S&P500 and 49% of FTSE100 companies, who rely on COMPANY to provide secure, optimized and modern services including analytics and insights that could protect them from market manipulation and market anomalies. In an era of algorithmic insights and trading the best way for COMPANY to understand how it can protect its clients is for it to understand how ML algorithms can generate insights in this area.

This report aims to explore time series forecasting and anomaly detection techniques, which enable creation of predictive model for 'beating' the market. Such a model could later be further developed to identify use of "spoofing", a technique where fake orders are placed to create an illusion of supply and demand, "wash trading", where same security is bought and sold to create artificial activity, which can lead to price distortions, or "flash crashes", caused by High-frequency trading algorithms, which can erode investor confidence.

Clustering algorithms can help identify abnormal trading patterns, SARIMA or ARIMA models can identify unusual spikes or drops, and supervised learning models can use historical data to create an understanding of what is "normal" to help detect suspicious activity. Regression models can discover patterns to help identify relationship between various features, such as the impact of trading volumes on share prices, while deep neural networks can tackle even the largest datasets and predict future anomalies, or anomalies in their inception, allowing COMPANY to be proactive rather than reactive. Insights gained from the above can be turned into automated alerts or further predictive analysis.

As this is a POC, publicly available data will be used throughout and will help demonstrate need for preprocessing, feature engineering and ways to tackle real-time data feeds.

2 TABLE OF CONTENTS

3 LIST OF FIGURES

4 Algorithms and Application Areas

The quest for forecasting timeseries data can broadly be divided into algorithms that emerge from the field of statistics and those that fall under machine learning. Both approaches will be considered in hopes that insights gained can build upon one another.

4.1 Exponential Smoothing Model (EMA)

Exponential Smoothing is a variation of the Moving Average method, with the added consideration of decreasing importance of past data points. Financial Services companies can benefit from metrics such as these for estimating a mean around which future stock prices are expected to oscillate, or for changes in trend directions whenever prices cut through the EMA line.

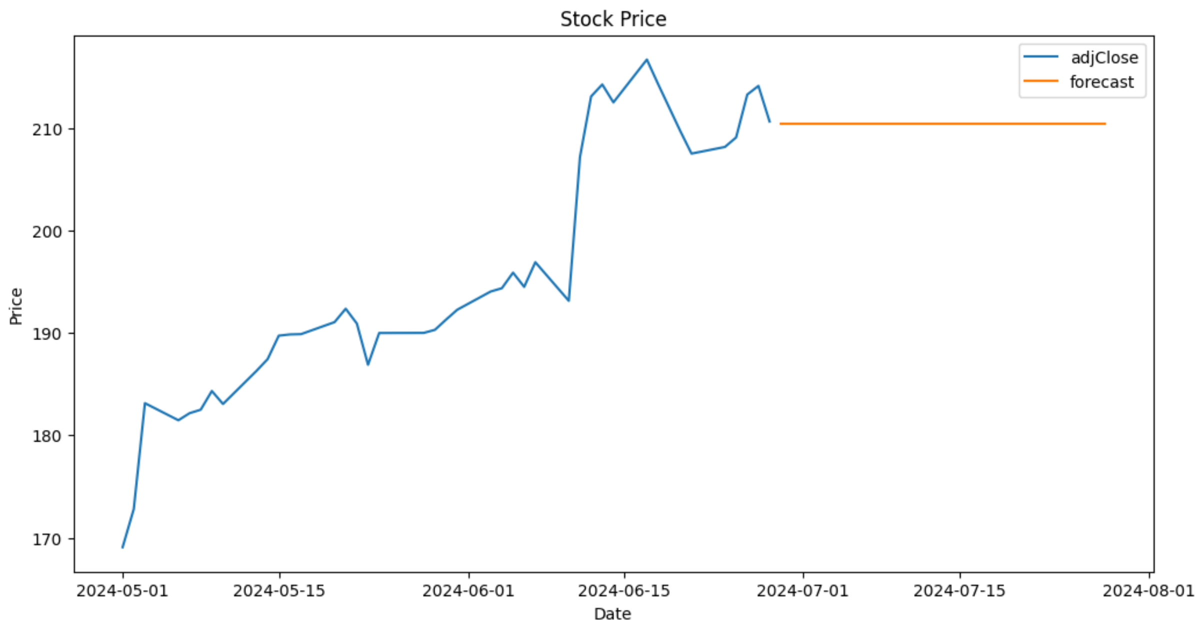


Figure 1- Future forecast using Exponential Smoothing model alone. As seen is not sufficient for anything but simplest mean prediction

Although EMA itself is a supervised approach, once the best fit alpha is chosen, it could also be used as root of a classification model that aims to predict changes in trend. As seen below EMA that is only cut by definitive trend reversals could be trusted to indicate significant market movements.

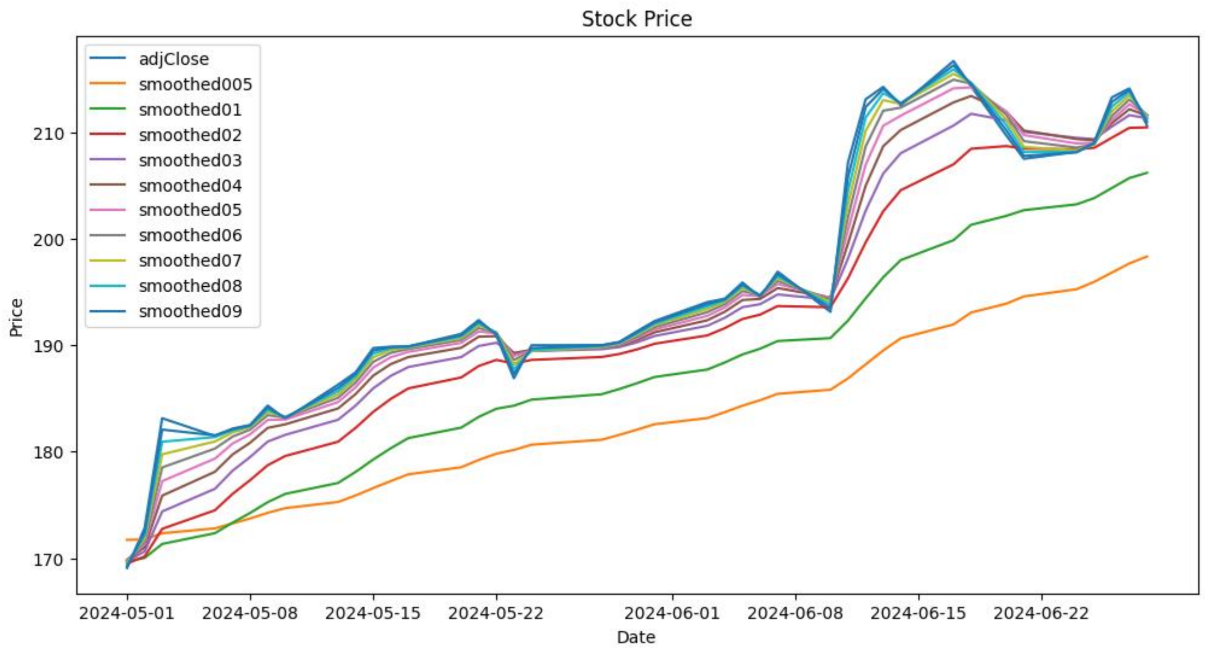


Figure 2-- Exponential Smoothing by various factors on historical data points

When we extend the forecast horizon, as seen in earlier Figures, the EMA simply flatlines as its extremely autoregressive, however single value prediction plotted over the long-term reveals, that EMA can be a useful indicator. For example, an EMA with alpha=0.05 seems mostly cut only when the trend reverses (green arrows), which can then be used to predict further market movements. However even this line is inconsistent when there are periods of market hesitancy (sideways trend, highlighted in red) and so could best serve as only one of many indicators used in a bigger model.

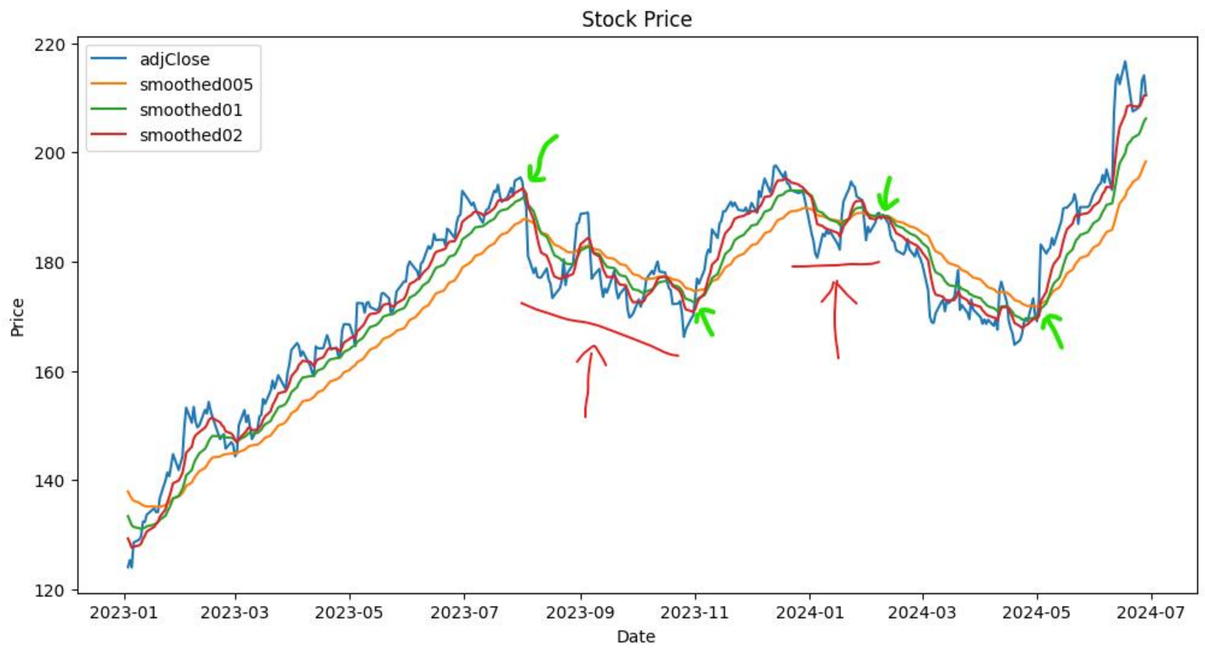


Figure 3- Exponential Smoothing for long-term trend reversal analysis

4.2 ARIMA, SARIMA AND SARIMAX

The ARIMA and SARIMA use supervised learning to forecast time series data of market assets. These techniques effectively model linear relationships as well as capture seasonality in data. SARIMAX differs from SARIMA in that it adds External factors, economic indicators and/or market sentiment, which is expected to increase forecast accuracy. Choice of external factors introduces some risk for bias and fairness needs to be maintained together with transparency with stakeholders on how the algorithm is trained.

4.3 LSTM NEURAL NETWORKS

LSTM Neural Networks are expected to be useful for timeseries forecasting due to their ability to infer from past events to which they have access within their design. Due to its more versatile nature and ability to pick up on various patterns found in the data a neural network could serve as the top layer of the model that would generate the prediction, based on the EMA and ARIMA insights. It could be trained on many other indicators and features, including market sentiment reports either publicly available or calculated by an NLP model based on available stories. LSTM autoencoders can be used for anomaly detection (Hong, 2024), as they are able to learn patterns from normal data and identify when the pattern changes indicating e.g. trend reversal or even potential black swan event.

As such an ensemble model would be required, which effectively collates knowledge and insights from various ‘expert’ models to reach a conclusion. The nature of the conclusion is dependent on the type of problem we are trying to solve. It could either try to project the prices into the future, but it could equally well output a binary / multiclass classification indicating if the moment is good for trading long or short or if the market is bearish or bullish.

4.4 TIME SERIES CLUSTERING

Time series clustering is an unsupervised learning technique that organises data points into groups based on similarity to maximise similarity within groups and minimise it across groups. This aids the anomaly detection algorithms in more complex setups (McConville, Santos-Rodriguez, Piechocki, & Craddock, 2019). The idea is to use combination of clustering and autoencoder approaches to discover the underlying clusters.

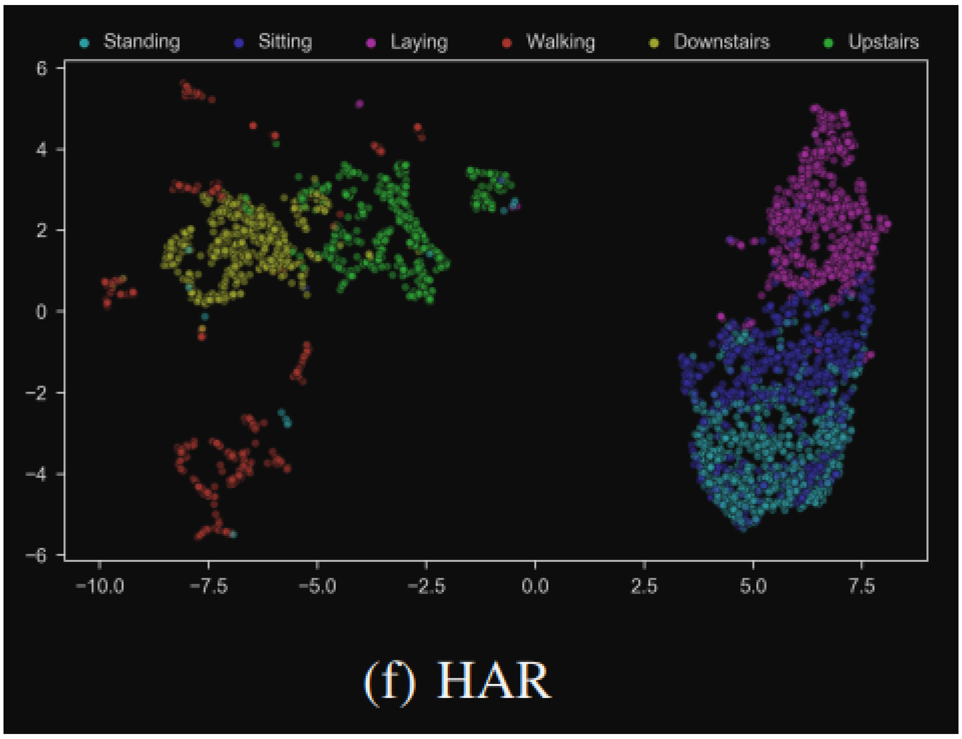


Figure 4 - Clustering Neural Network translating Time-series data into clusters

4.5 Legal Social and Ethical dimensions

For all the above algorithms, multiple potential issues arise; all require established data privacy and governance practices to ensure compliance with FCA and to prevent inaccurate predictions. However, in the realm of AI established policies are often missing, which means COMPANY needs to take extra precautions to consider risk of market manipulation stemming from the tools it develops to tackle market manipulation itself, e.g. internal misuse.

Proper risk assessment should reveal and help tackle ethical issues in using predictions to guide investment strategy, or delivering potential market manipulation insights to staff who can themselves abuse the market, especially with their additional, role related, knowledge of insider information. Full transparency in this area should promote trust and help react to any ethical issues, previously unbeknown, as they arise.

Wherever decisions are made using the algorithms, it must be remembered that these are prone to bias and so these actions will need to be explained to stakeholders and grounded in objectivity. This will also prevent discrimination arising from the insights gained. In the case of COMPANY the most obvious risk is for an algorithm to be much better at providing analytics to big data companies, where training data is abundant, at a disadvantage to the small businesses, who might not have sufficient data. Alternatively, small business shareholder profile might be different from that to which the models have been trained, effectively providing inaccurate predictions to the small businesses, while remaining accurate for the big.

5 Model Building using Python

5.1 ARIMA model

5.1.1 Data Collection (historical price data)

For this report a publicly available stock price data for Apple has been used, sourced from a free account tier API offered by Tiingo. The dataset consists of close, high, low, open, volume and adjusted equivalents of each of those, as well as ‘dividend cash’ and ‘split Factor’ records for AAPL stock from 01/01/2000 until 28/06/2024. There are many stock data providers both free and paid, including finance.yahoo.com, however the accuracy of the data at this key stage of the data scientist’s journey is paramount. This is due to the complex nature of the stock market, with various calculations being used by brokers, registrars and providers of stock data and only some adhering to the international standards. Tiingo was selected as it came on top in a comprehensive analysis available publicly from a third party (Rubsamen, 2022).

5.1.2 Data Preprocessing (cleaning and normalization)

Various transformations of the data were necessary to prepare it for modelling. First a decision had to be made as to which features of the dataset to use. ARIMA requires a single timeseries to work so the choice was really between the close price and adjusted close price. The difference between the two was in whether Apple’s multiple stock splits throughout its history were taken into account and how both approaches would affect the model.

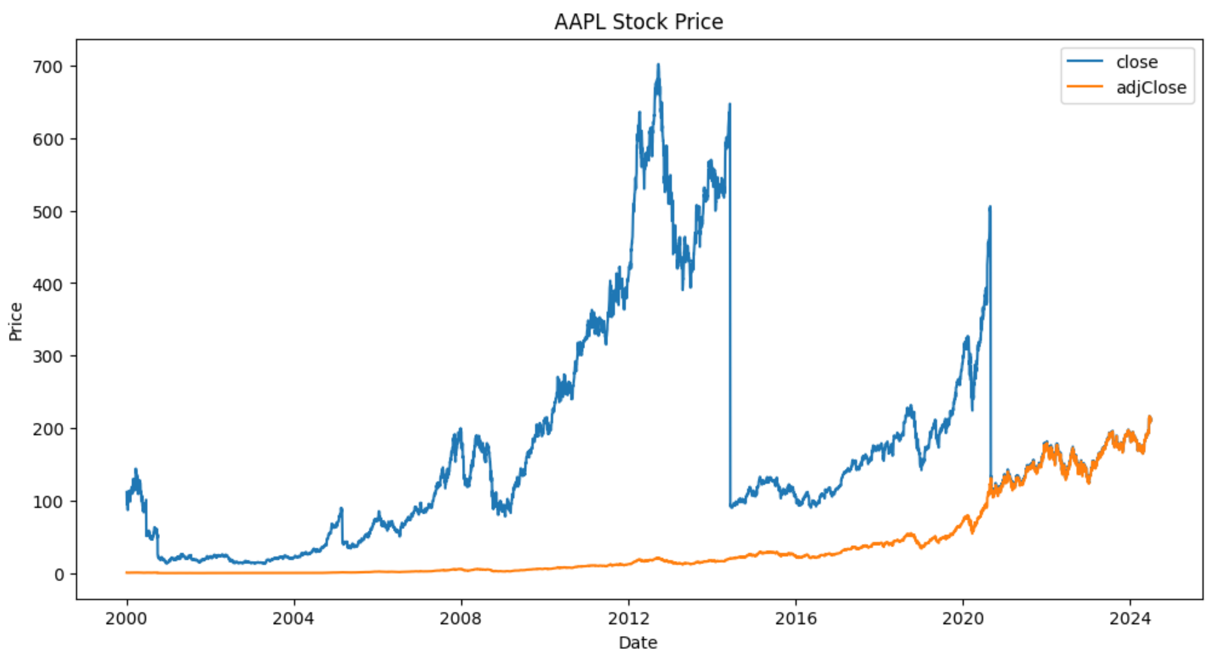
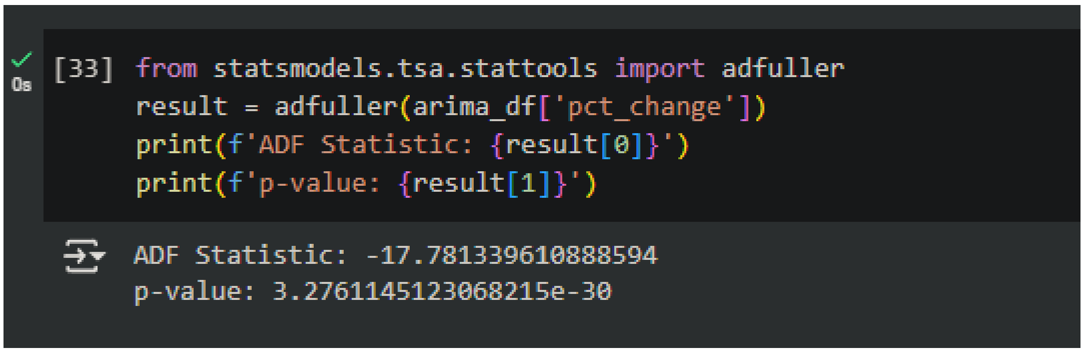


Figure 5 - Close and Adjusted Close price of AAPL stock

This decision affects the subsequent need to differentiate the series in preparation for ARIMA. Differencing simply takes the difference between current and previous value and the inexplicable sudden drop of the price would only confuse the model which is trying to discover patterns in the data.



```
[33] from statsmodels.tsa.stattools import adfuller
     result = adfuller(arima_df['pct_change'])
     print(f'ADF Statistic: {result[0]}')
     print(f'p-value: {result[1]}')

ADF Statistic: -17.781339610888594
p-value: 3.2761145123068215e-30
```

Figure 6 - ADF test showing that AAPL stock is NOT stationary

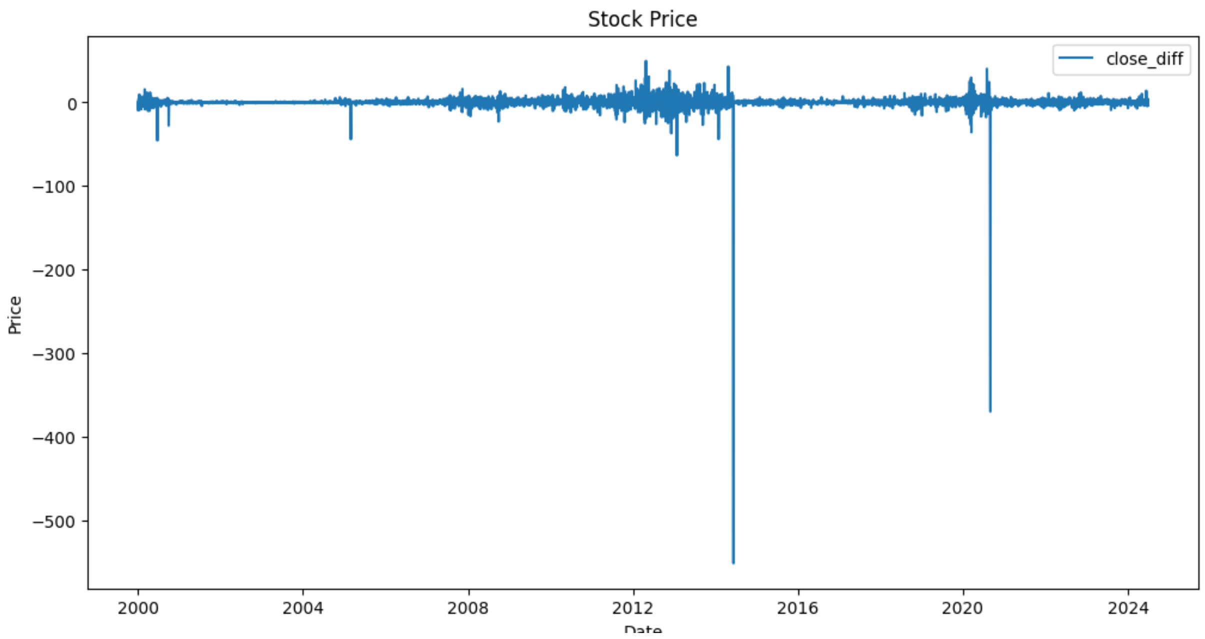


Figure 7 - Differencing performed on the Close price (spikes are stock splits)

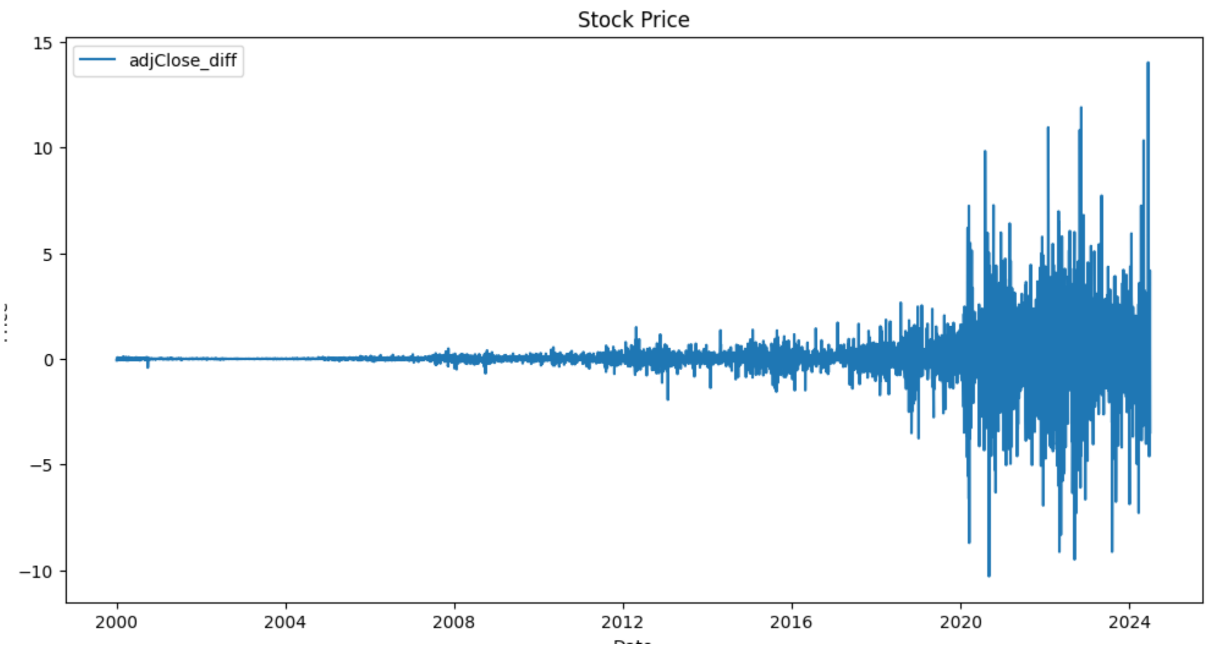


Figure 8 - Differencing performed on adjClose data (the increase in volatility is actually the effect of the adjustment, which replaced earlier prices e.g. $100 with $0.08 to account for the splits)

As can be seen in the figure above even adjusted differencing presents a challenge, therefore it may be better to make the series stationary using percentage change.

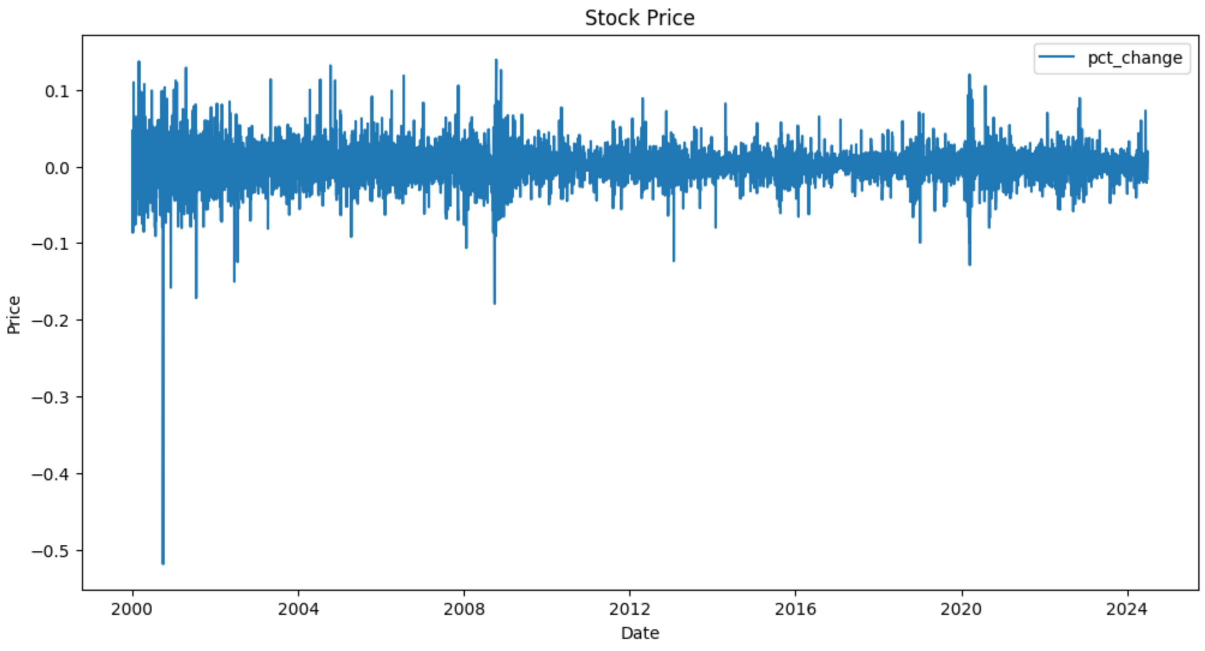


Figure 9 - Percentage Change differencing performed against the adjusted stock price (to eliminate stock split spikes)

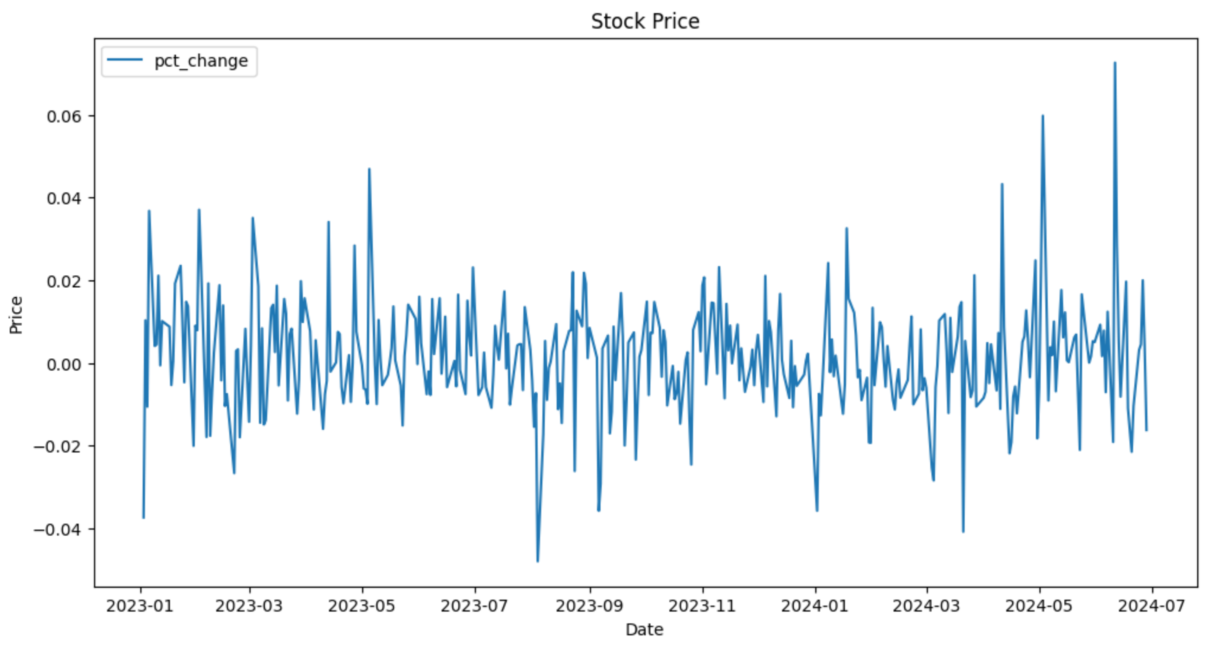
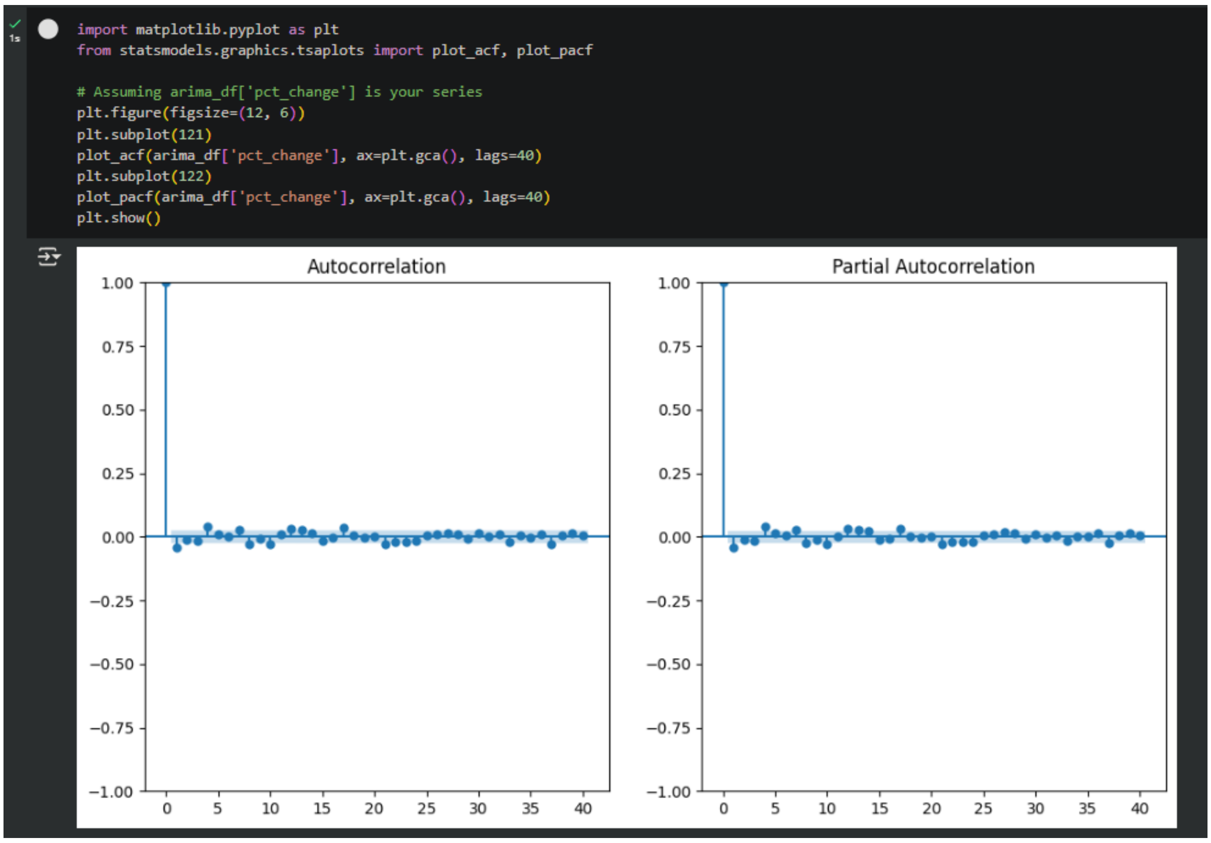


Figure 10 - Zoomed in differenced AAPL percentage change time series

As can be seen in figure above, the stationary pattern of the AAPL stock emerges and can now be fitted by the ARIMA model.

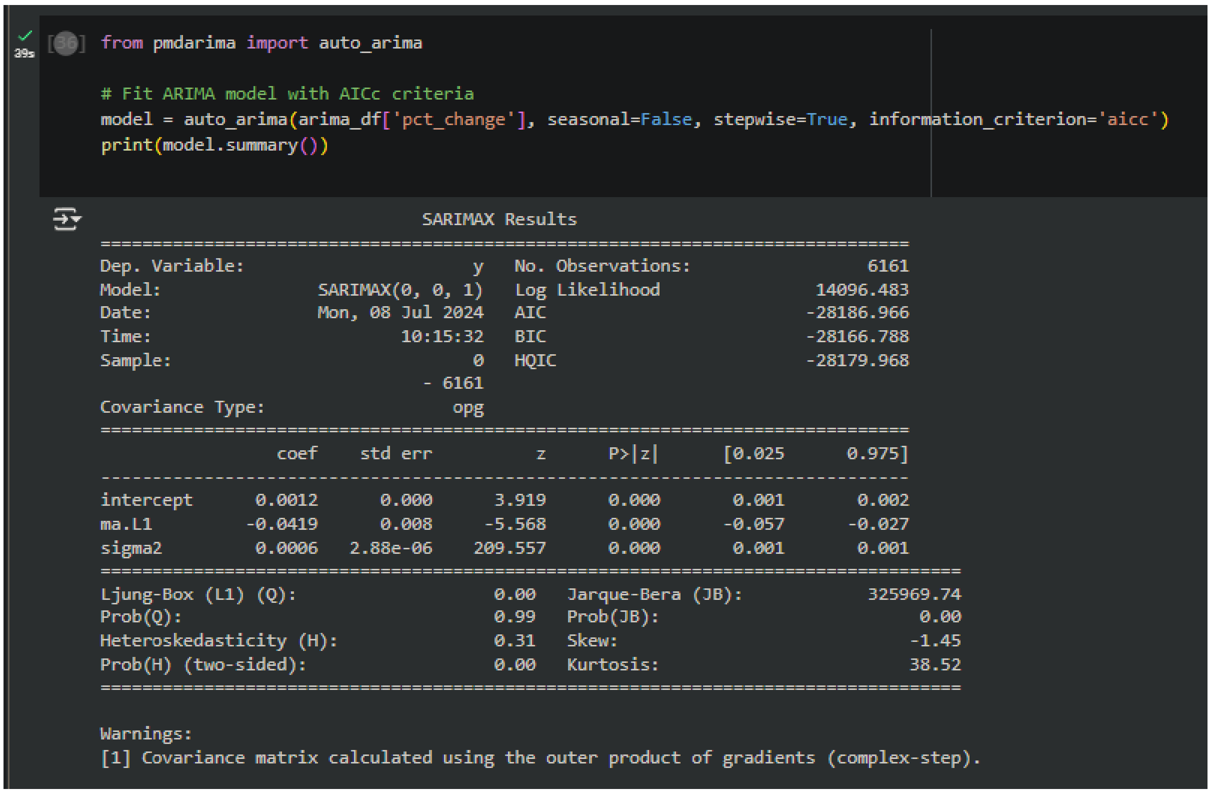


```
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# Assuming arima_df['pct_change'] is your series
plt.figure(figsize=(12, 6))
plt.subplot(121)
plot_acf(arima_df['pct_change'], ax=plt.gca(), lags=40)
plt.subplot(122)
plot_pacf(arima_df['pct_change'], ax=plt.gca(), lags=40)
plt.show()
```

Figure 11 - Successful ACF and PACF test against the arima dataset

5.1.3 Model Training

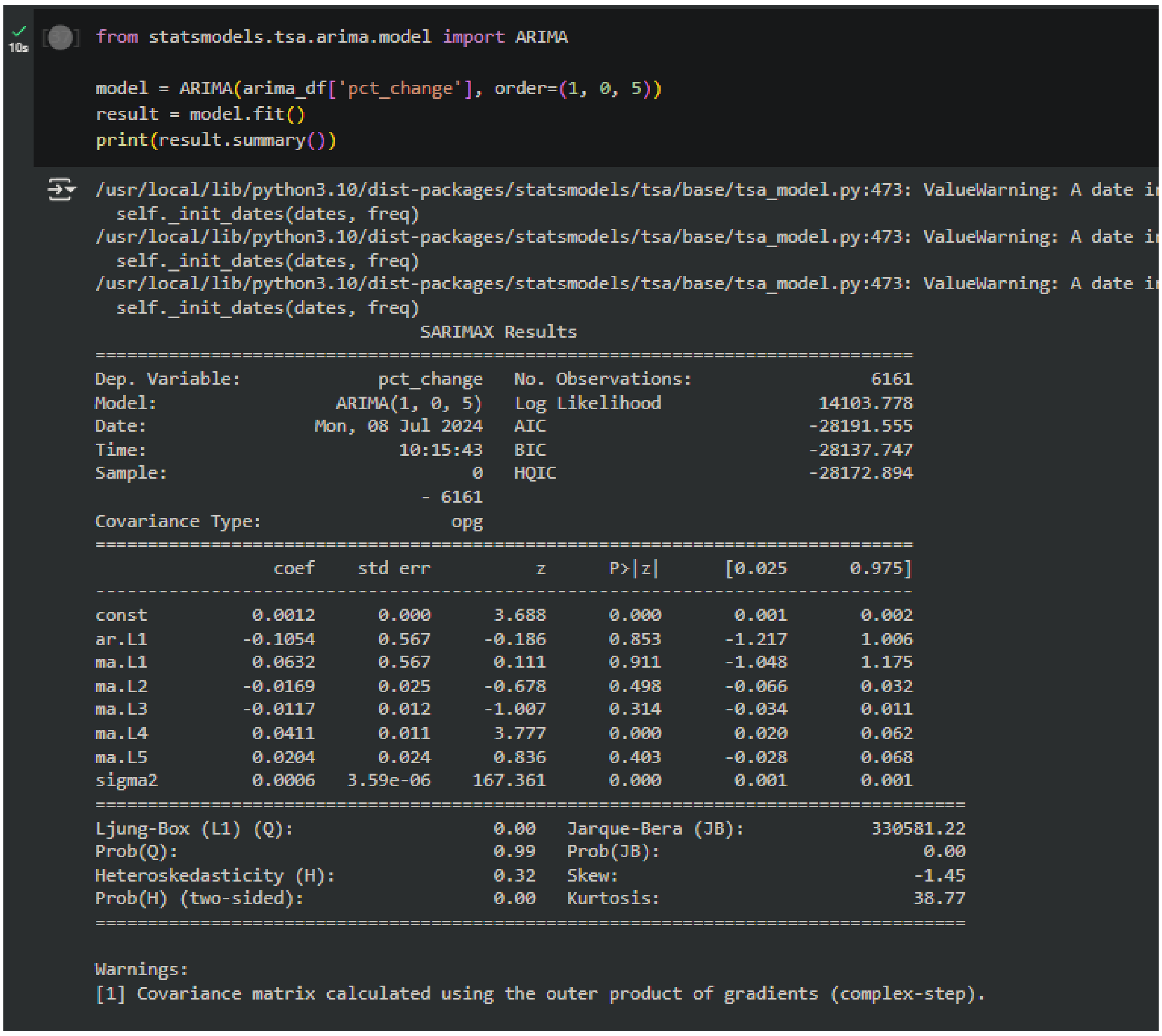
Multiple experiments have been conducted using various p, d, q parameter options, despite the ACF and PACF tests not presenting a picture that can be analysed manually with ease. The automatic process has therefore been chosen, which selected (0, 0, 1) for the p, d, q variables. The ‘d’ stands for differencing, which is expected to be a 0 since we have already differenced the timeseries using pct_change, as opposed to the standard differencing that is performed within the ARIMA model against the values themselves.



```
from pmdarima import auto_arima

# Fit ARIMA model with AICc criteria
model = auto_arima(arima_df['pct_change'], seasonal=False, stepwise=True, information_criterion='aicc')
print(model.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 6161
Model:               SARIMAX(0, 0, 1)   Log Likelihood               14096.483
Date:                Mon, 08 Jul 2024   AIC                         -28186.966
Time:                        10:15:32   BIC                         -28166.788
Sample:                             0   HQIC                        -28179.968
                               - 6161
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.0012      0.000      3.919      0.000       0.001       0.002
ma.L1         -0.0419      0.008     -5.568      0.000      -0.057      -0.027
sigma2         0.0006   2.88e-06    209.557      0.000       0.001       0.001
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):            325969.74
Prob(Q):                              0.99   Prob(JB):                         0.00
Heteroskedasticity (H):               0.31   Skew:                            -1.45
Prob(H) (two-sided):                  0.00   Kurtosis:                        38.52
===================================================================================

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

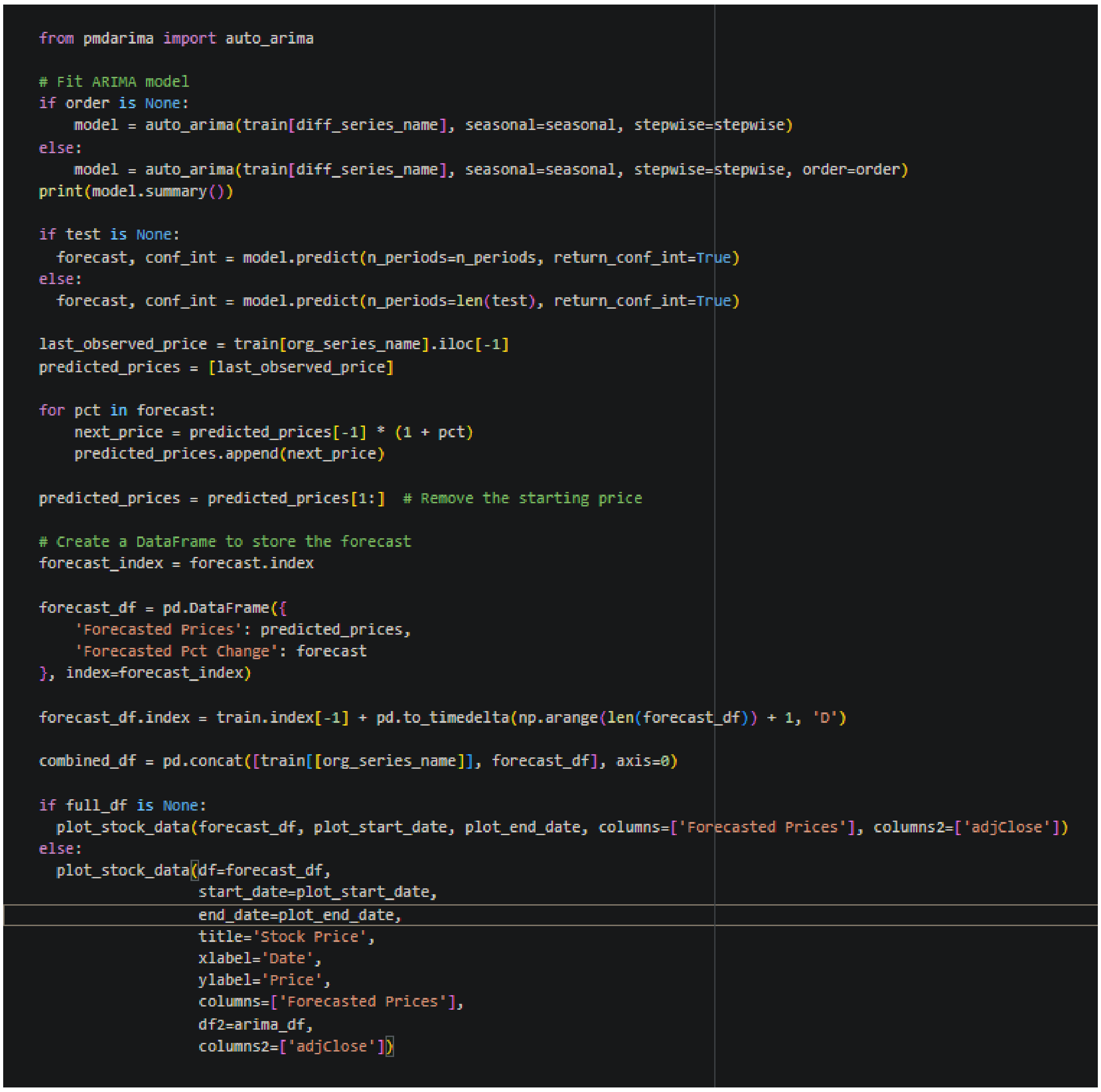
Figure 12 - auto_arima results, indicating best fit model parameters



```
from statsmodels.tsa.arima.model import ARIMA

model = ARIMA(arima_df['pct_change'], order=(1, 0, 5))
result = model.fit()
print(result.summary())
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date i
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date i
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date i
  self._init_dates(dates, freq)
                               SARIMAX Results
==============================================================================
Dep. Variable:             pct_change   No. Observations:                 6161
Model:                 ARIMA(1, 0, 5)   Log Likelihood               14103.778
Date:                Mon, 08 Jul 2024   AIC                         -28191.555
Time:                        10:15:43   BIC                         -28137.747
Sample:                             0   HQIC                        -28172.894
                               - 6161
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0012      0.000      3.688      0.000       0.001       0.002
ar.L1         -0.1054      0.567     -0.186      0.853      -1.217       1.006
ma.L1          0.0632      0.567      0.111      0.911      -1.048       1.175
ma.L2         -0.0169      0.025     -0.678      0.498      -0.066       0.032
ma.L3         -0.0117      0.012     -1.007      0.314      -0.034       0.011
ma.L4          0.0411      0.011      3.777      0.000       0.020       0.062
ma.L5          0.0204      0.024      0.836      0.403      -0.028       0.068
sigma2         0.0006   3.59e-06    167.361      0.000       0.001       0.001
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):            330581.22
Prob(Q):                              0.99   Prob(JB):                         0.00
Heteroskedasticity (H):               0.32   Skew:                            -1.45
Prob(H) (two-sided):                  0.00   Kurtosis:                        38.77
===================================================================================

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

Figure 13 - Manual testing of various ARIMA models, did not yield good enough results. As can be seen here the P>|z| variables that are close to 0 indicate good fit, of which there is only 1

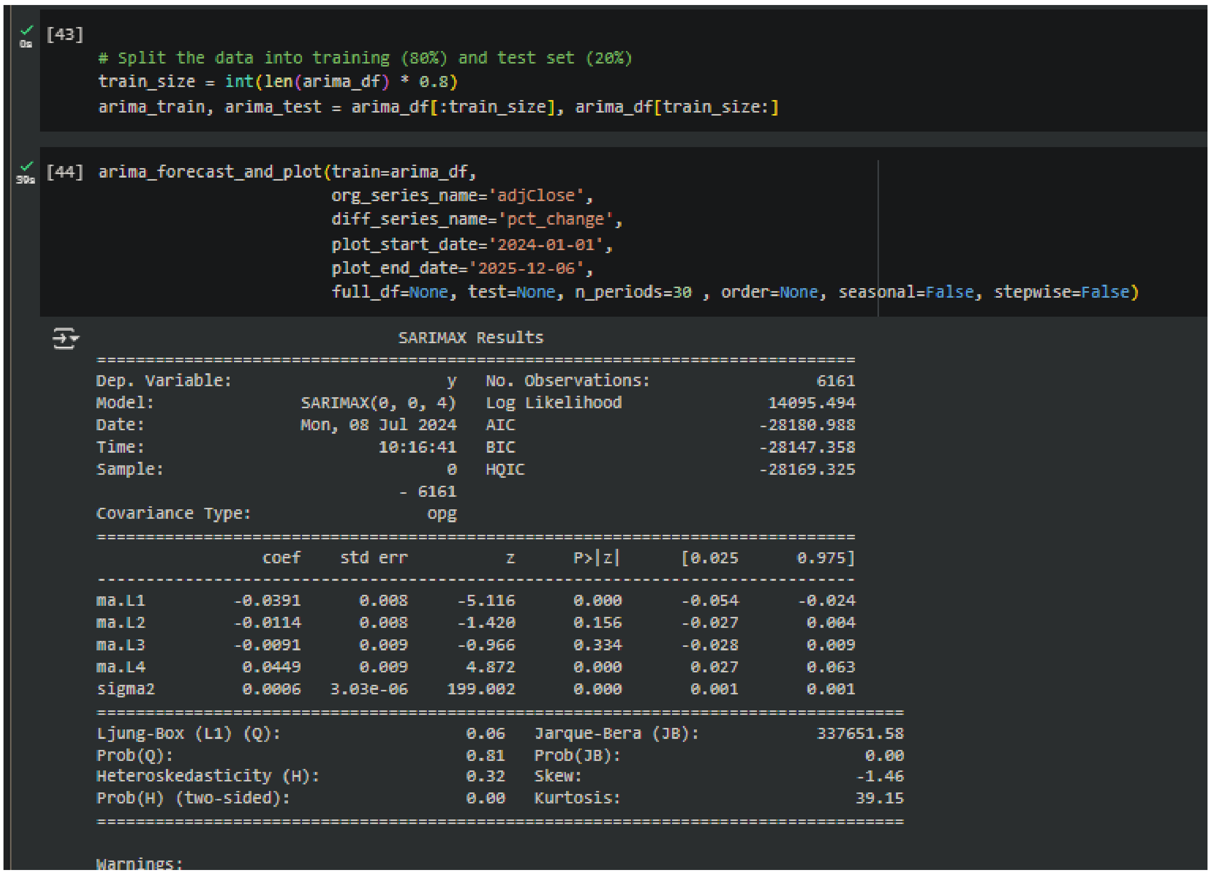


```python
from pmdarima import auto_arima

# Fit ARIMA model
if order is None:
    model = auto_arima(train[diff_series_name], seasonal=seasonal, stepwise=stepwise)
else:
    model = auto_arima(train[diff_series_name], seasonal=seasonal, stepwise=stepwise, order=order)
print(model.summary())

if test is None:
  forecast, conf_int = model.predict(n_periods=n_periods, return_conf_int=True)
else:
  forecast, conf_int = model.predict(n_periods=len(test), return_conf_int=True)

last_observed_price = train[org_series_name].iloc[-1]
predicted_prices = [last_observed_price]

for pct in forecast:
    next_price = predicted_prices[-1] * (1 + pct)
    predicted_prices.append(next_price)

predicted_prices = predicted_prices[1:]  # Remove the starting price

# Create a DataFrame to store the forecast
forecast_index = forecast.index

forecast_df = pd.DataFrame({
    'Forecasted Prices': predicted_prices,
    'Forecasted Pct Change': forecast
}, index=forecast_index)

forecast_df.index = train.index[-1] + pd.to_timedelta(np.arange(len(forecast_df)) + 1, 'D')

combined_df = pd.concat([train[[org_series_name]], forecast_df], axis=0)

if full_df is None:
  plot_stock_data(forecast_df, plot_start_date, plot_end_date, columns=['Forecasted Prices'], columns2=['adjClose'])
else:
  plot_stock_data(df=forecast_df,
                  start_date=plot_start_date,
                  end_date=plot_end_date,
                  title='Stock Price',
                  xlabel='Date',
                  ylabel='Price',
                  columns=['Forecasted Prices'],
                  df2=arima_df,
                  columns2=['adjClose'])
```

Figure 14 - A function developed to facilitate repeated experimentation

5.1.4 Model Evaluation (MAE, RMSE)

Using a train and test split only (without validation split) of 80% to 20% an auto_arima test was performed:



```
[43]
    # Split the data into training (80%) and test set (20%)
    train_size = int(len(arima_df) * 0.8)
    arima_train, arima_test = arima_df[:train_size], arima_df[train_size:]

[44] arima_forecast_and_plot(train=arima_df,
                             org_series_name='adjClose',
                             diff_series_name='pct_change',
                             plot_start_date='2024-01-01',
                             plot_end_date='2025-12-06',
                             full_df=None, test=None, n_periods=30 , order=None, seasonal=False, stepwise=False)
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 6161
Model:               SARIMAX(0, 0, 4)   Log Likelihood               14095.494
Date:                Mon, 08 Jul 2024   AIC                         -28180.988
Time:                        10:16:41   BIC                         -28147.358
Sample:                             0   HQIC                        -28169.325
                               - 6161
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.0391      0.008     -5.116      0.000      -0.054      -0.024
ma.L2         -0.0114      0.008     -1.420      0.156      -0.027       0.004
ma.L3         -0.0091      0.009     -0.966      0.334      -0.028       0.009
ma.L4          0.0449      0.009      4.872      0.000       0.027       0.063
sigma2         0.0006   3.03e-06    199.002      0.000       0.001       0.001
===================================================================================
Ljung-Box (L1) (Q):                   0.06   Jarque-Bera (JB):            337651.58
Prob(Q):                              0.81   Prob(JB):                         0.00
Heteroskedasticity (H):               0.32   Skew:                            -1.46
Prob(H) (two-sided):                  0.00   Kurtosis:                        39.15
===================================================================================

Warnings:
```

Figure 15 - 80/20 split auto_arima test result (fitting)

This was then plotted on a graph to visually compare and assess the results.

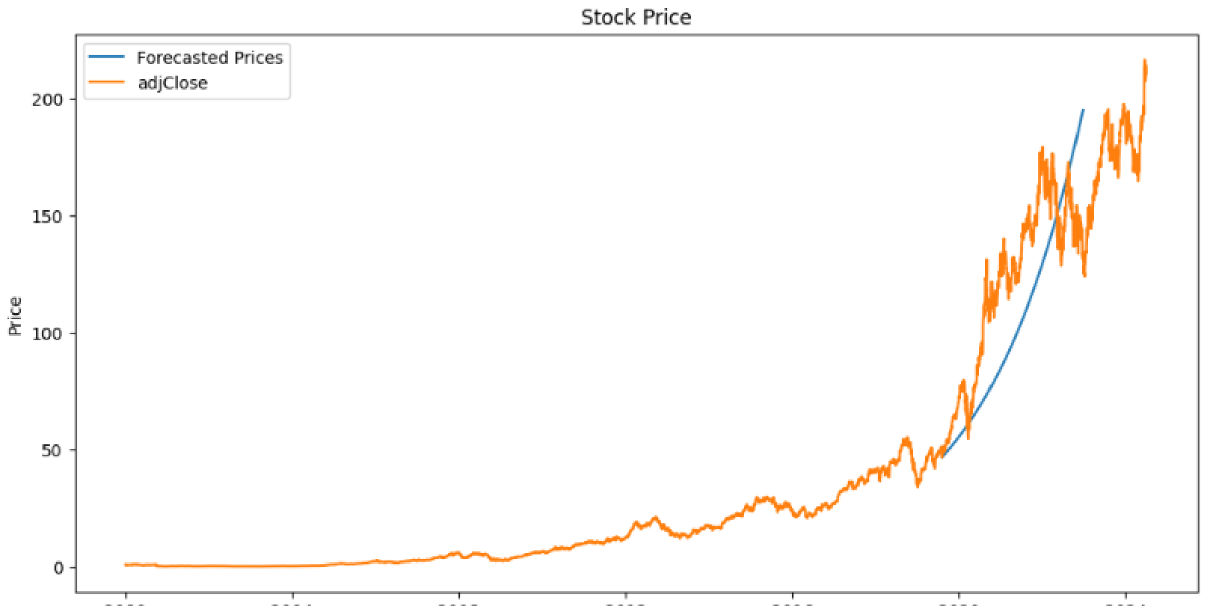


Figure 16 – ARIMA test results plotted on a graph. The blue line quite correctly forecasts the general trend direction of the stock inferred from the pattern of growth learnt in the train data segment

5.1.5 Results

The results of ARIMA test and visualised results may make the viewer optimistic, as it appears the model 'predicted' the extreme growth of AAPL stock. However, on the other hand the match could be attributed to luck, as stocks do not behave in a predictable way. Apple could have declared bankruptcy in an alternate timeline and any such model still cannot be used for any long-term prediction.

It must also be noted that the model does not measure its own accuracy for this long-term prediction, which would be a very harsh treatment if it did. Many experiments out there do apply a metric such as MAE or MSE, however it is argued here that for this particular use case – predicting the future movement of stock price – it is not actually telling us how well the model performed, because we are not interested in how exactly the model is able to follow the actual line, but if its prediction can benefit the shareholders. Shareholders, do not want to know the exact price of AAPL in 3 months' time, but if it will generally be increasing or decreasing, which ought to change our approach.

5.2 LSTM

5.2.1 Data Collection

For this experiment the Microsoft (MS) stock price dataset was chosen, to avoid the stock split issues described before. Yahoo Finance was used to fetch the entire history of MS and then focus in on the Close price only.

5.2.2 Data preprocessing (feature selection and scaling)

This dataset also required basic preprocessing such as datetime formatting, however the application of neural network required a few more steps.

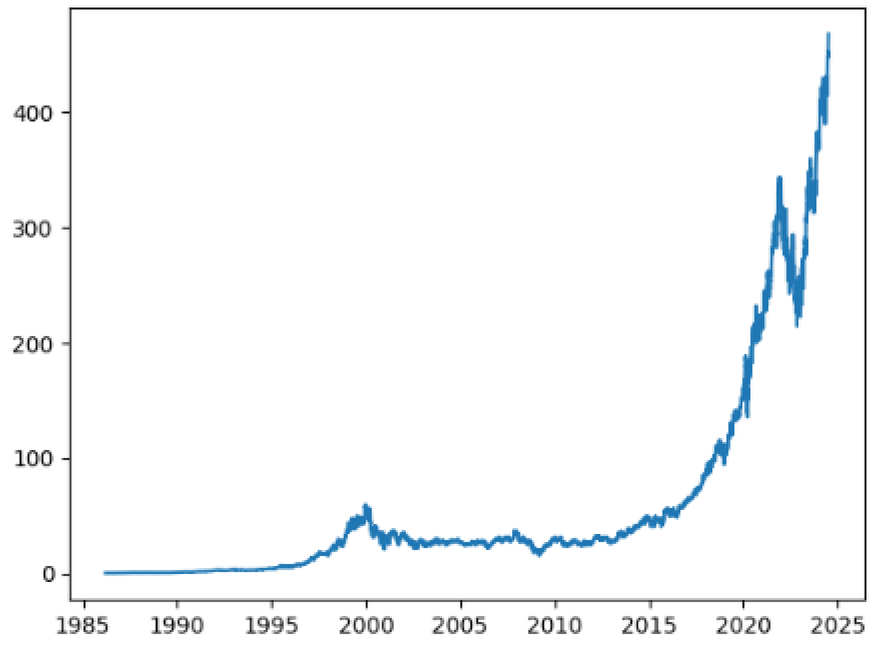
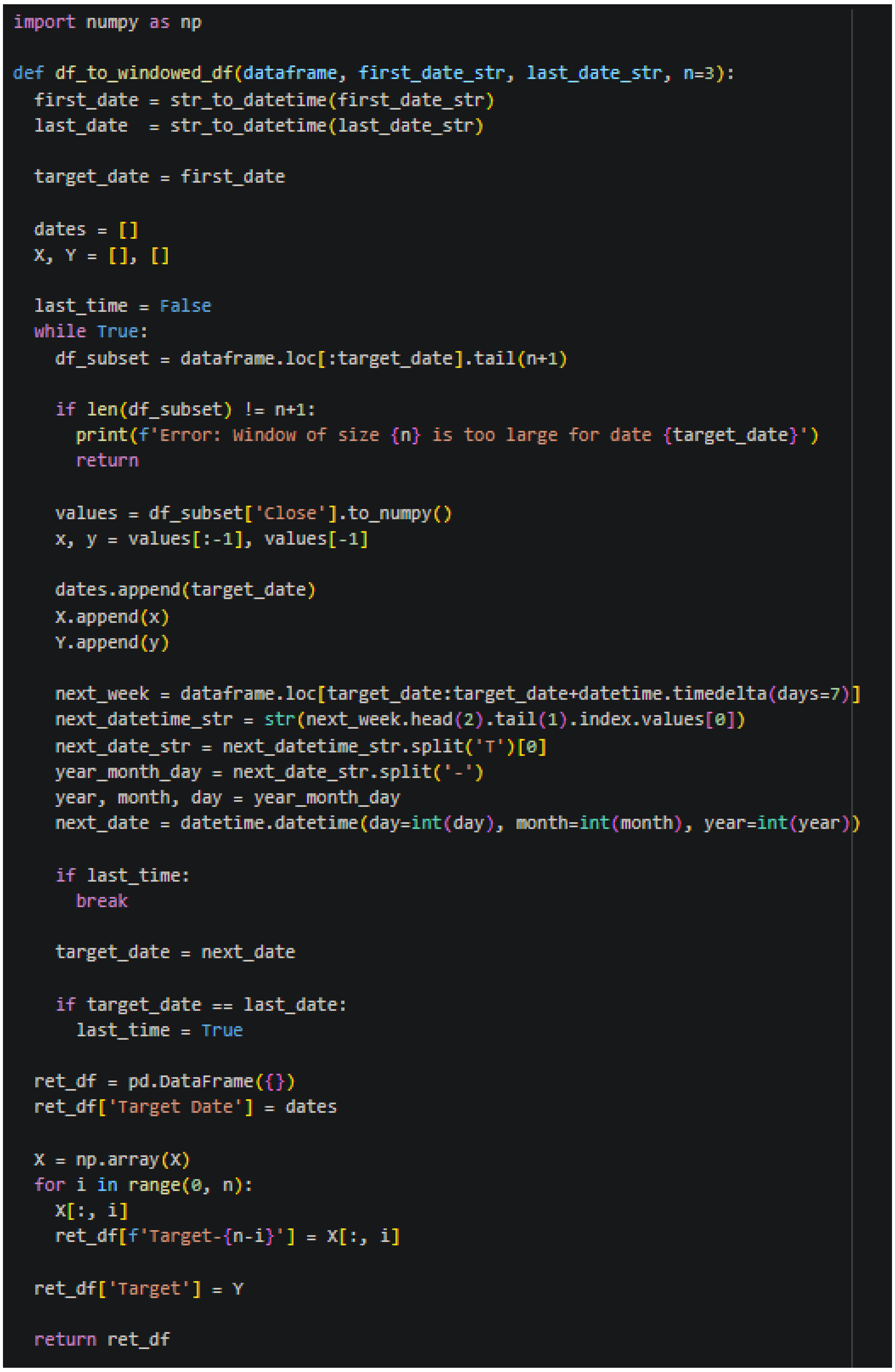


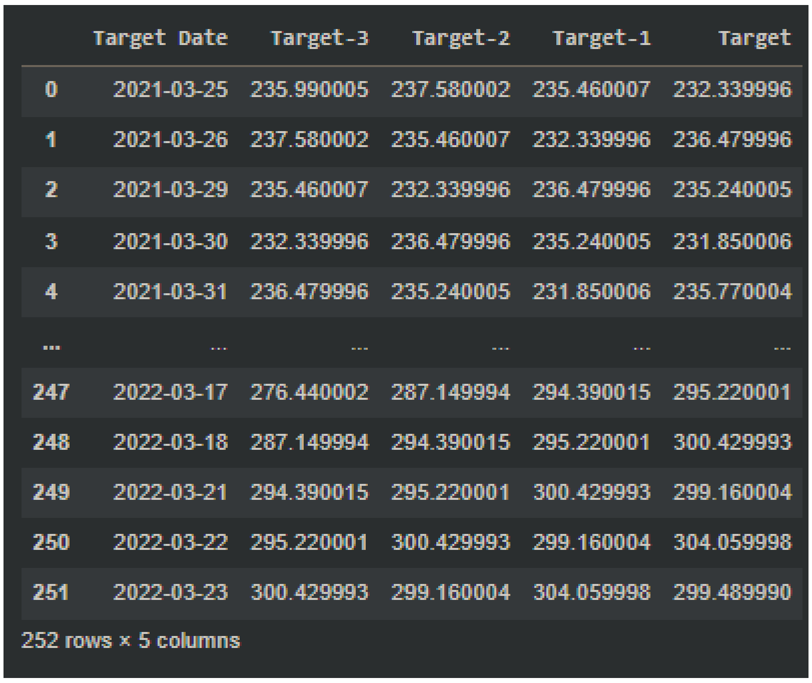
Figure 17 - Microsoft historical Close Price

Dataset was then transformed into a windowed dataset, where each time point would be looking at 3 past Close prices (features) to predict that time point's Close price (y).



```python
import numpy as np

def df_to_windowed_df(dataframe, first_date_str, last_date_str, n=3):
  first_date = str_to_datetime(first_date_str)
  last_date  = str_to_datetime(last_date_str)

  target_date = first_date

  dates = []
  X, Y = [], []

  last_time = False
  while True:
    df_subset = dataframe.loc[:target_date].tail(n+1)

    if len(df_subset) != n+1:
      print(f'Error: Window of size {n} is too large for date {target_date}')
      return

    values = df_subset['Close'].to_numpy()
    x, y = values[:-1], values[-1]

    dates.append(target_date)
    X.append(x)
    Y.append(y)

    next_week = dataframe.loc[target_date:target_date+datetime.timedelta(days=7)]
    next_datetime_str = str(next_week.head(2).tail(1).index.values[0])
    next_date_str = next_datetime_str.split('T')[0]
    year_month_day = next_date_str.split('-')
    year, month, day = year_month_day
    next_date = datetime.datetime(day=int(day), month=int(month), year=int(year))

    if last_time:
      break

    target_date = next_date

    if target_date == last_date:
      last_time = True

  ret_df = pd.DataFrame({})
  ret_df['Target Date'] = dates

  X = np.array(X)
  for i in range(0, n):
    X[:, i]
    ret_df[f'Target-{n-i}'] = X[:, i]

  ret_df['Target'] = Y

  return ret_df
```

Figure 18 - df_to_windowed_df function facilitating windowing of the Microsoft dataset

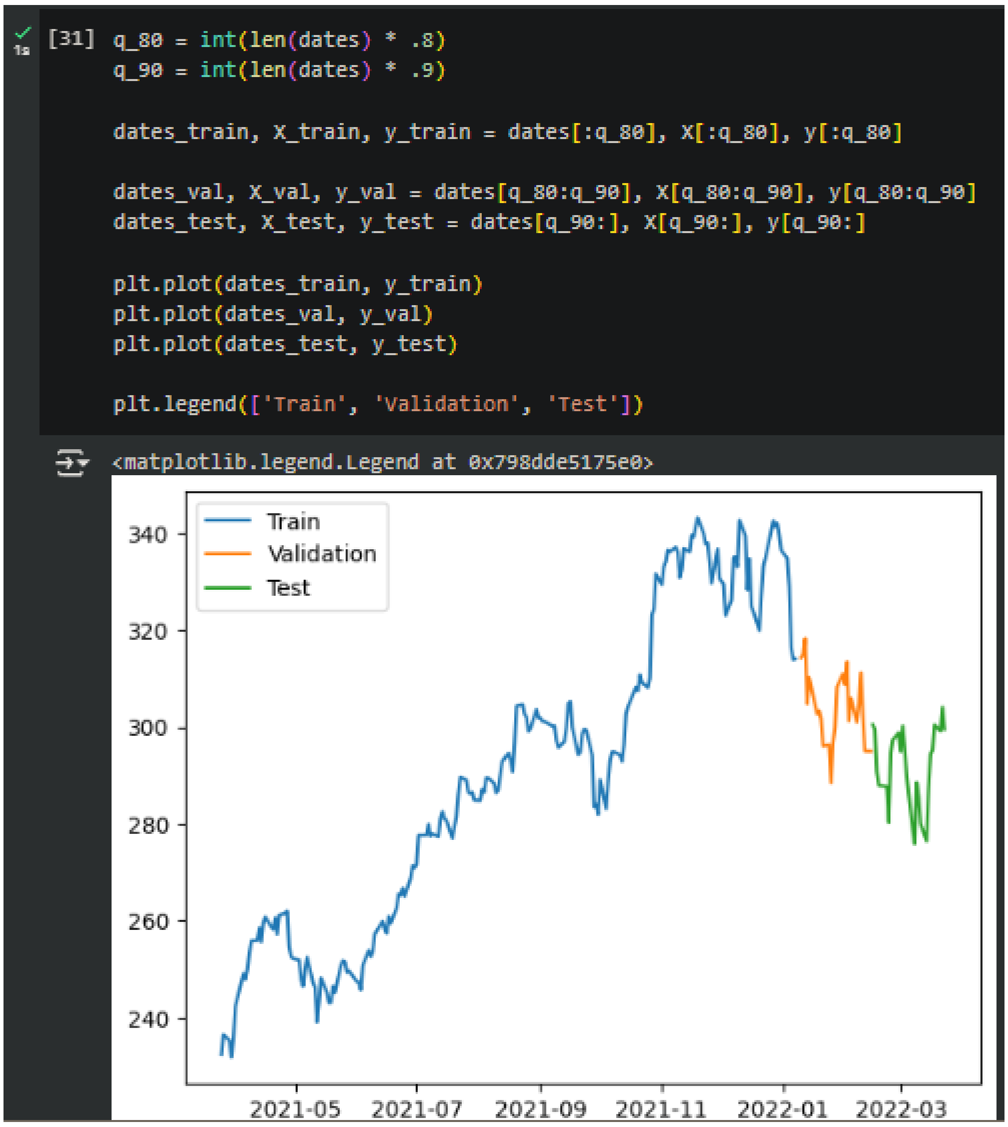


	Target Date	Target-3	Target-2	Target-1	Target
0	2021-03-25	235.990005	237.580002	235.460007	232.339996
1	2021-03-26	237.580002	235.460007	232.339996	236.479996
2	2021-03-29	235.460007	232.339996	236.479996	235.240005
3	2021-03-30	232.339996	236.479996	235.240005	231.850006
4	2021-03-31	236.479996	235.240005	231.850006	235.770004
...	...	...	...	...	...
247	2022-03-17	276.440002	287.149994	294.390015	295.220001
248	2022-03-18	287.149994	294.390015	295.220001	300.429993
249	2022-03-21	294.390015	295.220001	300.429993	299.160004
250	2022-03-22	295.220001	300.429993	299.160004	304.059998
251	2022-03-23	300.429993	299.160004	304.059998	299.489990

252 rows × 5 columns

Figure 19 - Dataset snippet showcasing the windowed dataset

The dataset was then divided into a train, validation and test segments using a 80/10/10 split, as shown below. Earlier experiment using the entire history from 1985 showed that the neural network is unable to extrapolate – that is model recent prices, because it has been trained on majorly much lower prices in the 1985 – 2020 period. Hence a shorter period was selected from March 2021 until now.



```
[31] q_80 = int(len(dates) * .8)
     q_90 = int(len(dates) * .9)

     dates_train, X_train, y_train = dates[:q_80], X[:q_80], y[:q_80]

     dates_val, X_val, y_val = dates[q_80:q_90], X[q_80:q_90], y[q_80:q_90]
     dates_test, X_test, y_test = dates[q_90:], X[q_90:], y[q_90:]

     plt.plot(dates_train, y_train)
     plt.plot(dates_val, y_val)
     plt.plot(dates_test, y_test)

     plt.legend(['Train', 'Validation', 'Test'])
```

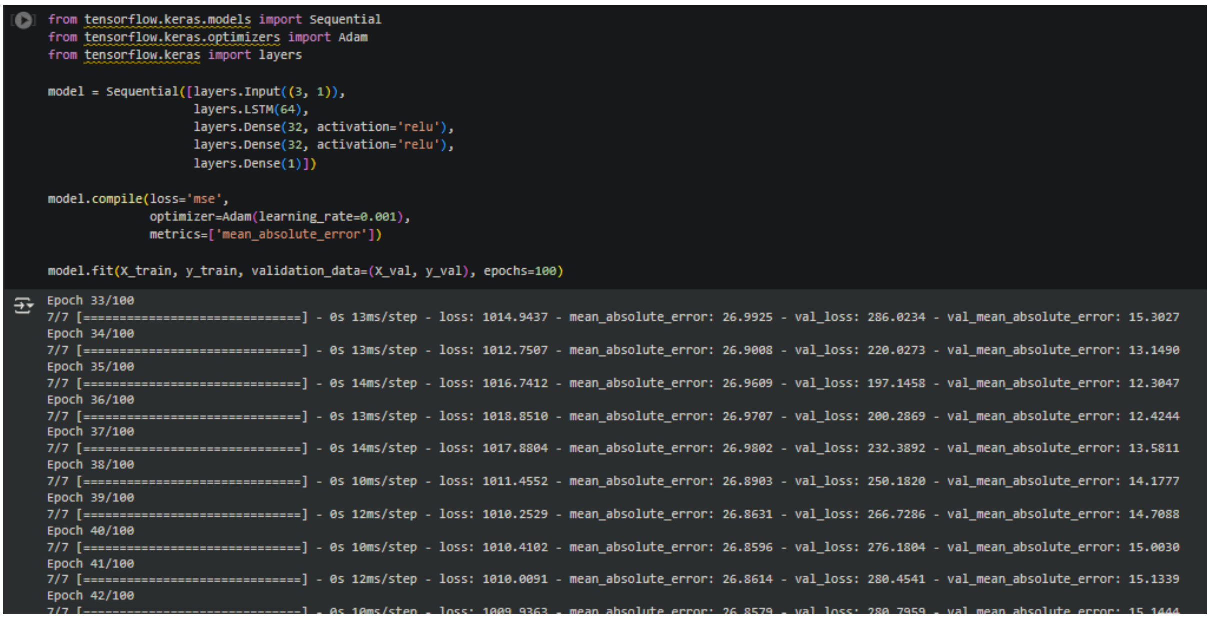
```
<matplotlib.legend.Legend at 0x798dde5175e0>
```

Figure 20 - Train, test, val split

5.2.3 Model training

While the theoretical basis of the models is well explained, more focus could be placed on optimizing these algorithms, perhaps by discussing parameter tuning or model validation techniques.

An LSTM Neural network model was created using TensorFlow Sequential API, using a standard recommended parameters of 64 neuron LSTM cell and 'relu' activated Dense layers leading to the Output layer that predicts the close price. Standard settings and 0.001 learning rate of the Adam optimizer have been used to create a base model 2 experiment (counting the full history test as base).



```
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import layers

model = Sequential([layers.Input((3, 1)),
                    layers.LSTM(64),
                    layers.Dense(32, activation='relu'),
                    layers.Dense(32, activation='relu'),
                    layers.Dense(1)])

model.compile(loss='mse',
              optimizer=Adam(learning_rate=0.001),
              metrics=['mean_absolute_error'])

model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=100)
```

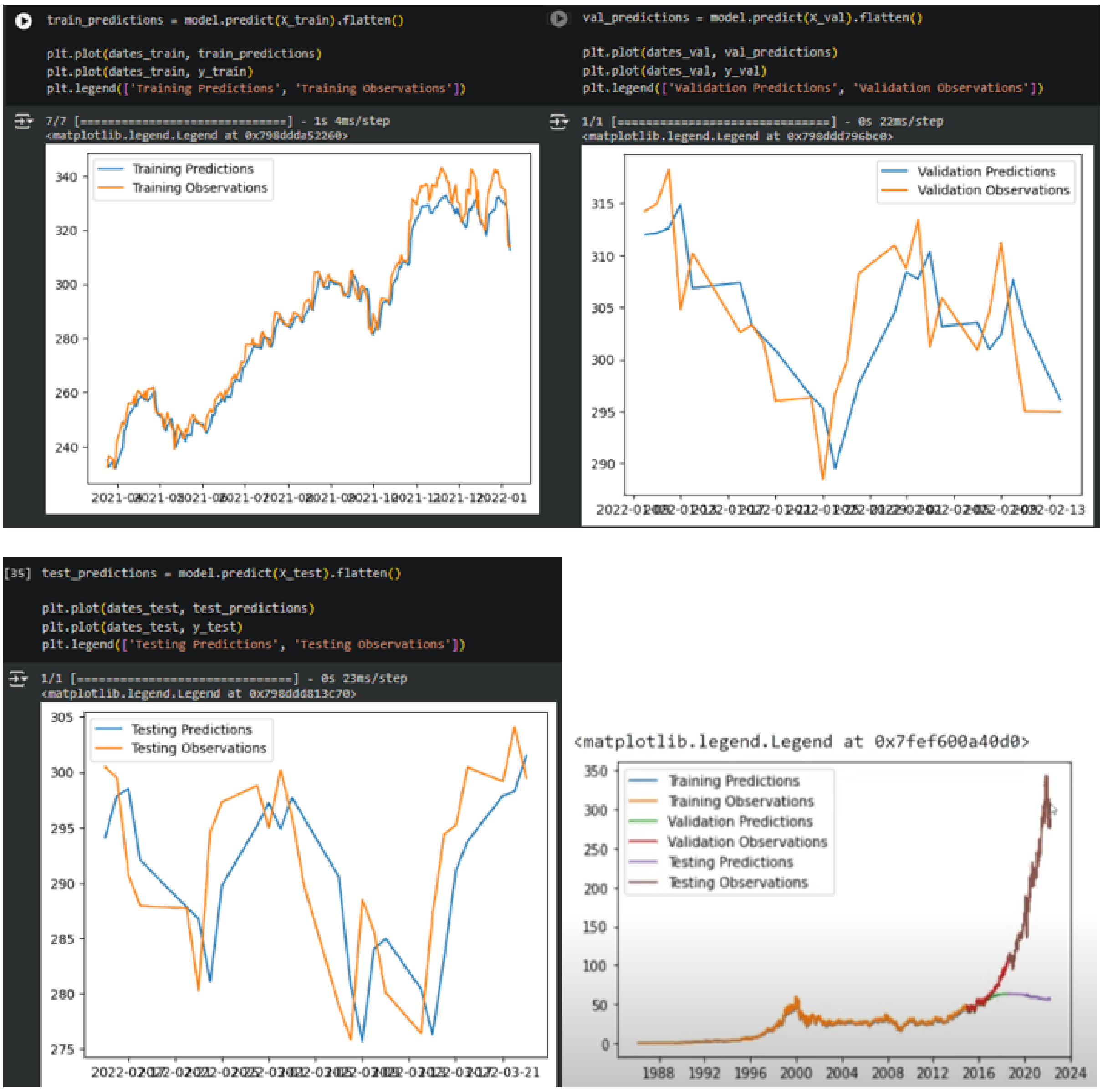
```
Epoch 33/100
7/7 [==============================] - 0s 13ms/step - loss: 1014.9437 - mean_absolute_error: 26.9925 - val_loss: 286.0234 - val_mean_absolute_error: 15.3027
Epoch 34/100
7/7 [==============================] - 0s 13ms/step - loss: 1012.7507 - mean_absolute_error: 26.9008 - val_loss: 220.0273 - val_mean_absolute_error: 13.1490
Epoch 35/100
7/7 [==============================] - 0s 14ms/step - loss: 1016.7412 - mean_absolute_error: 26.9609 - val_loss: 197.1458 - val_mean_absolute_error: 12.3047
Epoch 36/100
7/7 [==============================] - 0s 13ms/step - loss: 1018.8510 - mean_absolute_error: 26.9707 - val_loss: 200.2869 - val_mean_absolute_error: 12.4244
Epoch 37/100
7/7 [==============================] - 0s 14ms/step - loss: 1017.8804 - mean_absolute_error: 26.9802 - val_loss: 232.3892 - val_mean_absolute_error: 13.5811
Epoch 38/100
7/7 [==============================] - 0s 10ms/step - loss: 1011.4552 - mean_absolute_error: 26.8903 - val_loss: 250.1820 - val_mean_absolute_error: 14.1777
Epoch 39/100
7/7 [==============================] - 0s 12ms/step - loss: 1010.2529 - mean_absolute_error: 26.8631 - val_loss: 266.7286 - val_mean_absolute_error: 14.7088
Epoch 40/100
7/7 [==============================] - 0s 10ms/step - loss: 1010.4102 - mean_absolute_error: 26.8596 - val_loss: 276.1804 - val_mean_absolute_error: 15.0030
Epoch 41/100
7/7 [==============================] - 0s 12ms/step - loss: 1010.0091 - mean_absolute_error: 26.8614 - val_loss: 280.4541 - val_mean_absolute_error: 15.1339
Epoch 42/100
7/7 [==============================] - 0s 10ms/step - loss: 1009.9363 - mean_absolute_error: 26.8579 - val_loss: 280.7959 - val_mean_absolute_error: 15.1444
```

Figure 21 - LSTM Neural Network model using Sequential TensorFlow API

A training consisting of 100 epochs on this date limited dataset achieved a MAE of 4.7 and val_loss of 32, as compared to an earlier experiment where the entire dataset was used, which achieved only val MAE of 8.9 and val_loss of 240 dollars (where the stock close price itself is around the same mark).

5.2.4 Model Evaluation

The model results were then plotted for visual evaluation and it appears to be sticking closely to the actual prices during the training phase, as well as during validation and test.



```
train_predictions = model.predict(X_train).flatten()

plt.plot(dates_train, train_predictions)
plt.plot(dates_train, y_train)
plt.legend(['Training Predictions', 'Training Observations'])
```

```
7/7 [==============================] - 1s 4ms/step
<matplotlib.legend.Legend at 0x798ddda52260>
```

```
val_predictions = model.predict(X_val).flatten()

plt.plot(dates_val, val_predictions)
plt.plot(dates_val, y_val)
plt.legend(['Validation Predictions', 'Validation Observations'])
```

```
1/1 [==============================] - 0s 22ms/step
<matplotlib.legend.Legend at 0x798ddd796bc0>
```

```
[35] test_predictions = model.predict(X_test).flatten()

plt.plot(dates_test, test_predictions)
plt.plot(dates_test, y_test)
plt.legend(['Testing Predictions', 'Testing Observations'])
```

```
1/1 [==============================] - 0s 23ms/step
<matplotlib.legend.Legend at 0x798ddd813c70>
```

```
<matplotlib.legend.Legend at 0x7fef600a40d0>
```

Figure 22 - Train, Val and Test results as compared to observed values. Additional graph shows previous attempt using entire historical data available and its inability to predict in an area it was not trained on

5.2.5 Results

Despite the above promising results, it must be acknowledged that the model is simply predicting the next day's close price knowing last 3 days, and so it can be expected it will become proficient at that. However, such predictions are still autoregressive in nature and a similar visual result would be achieved with naïve Bayes model that simply expects tomorrow's price to be the same as today (or a number of lagging indicators such as EMA).

When this neural network was reconfigured to predict a longer window, it simply predicted the same price over and over, as it lacked the feedback it previously received from being updated with actual data every day.

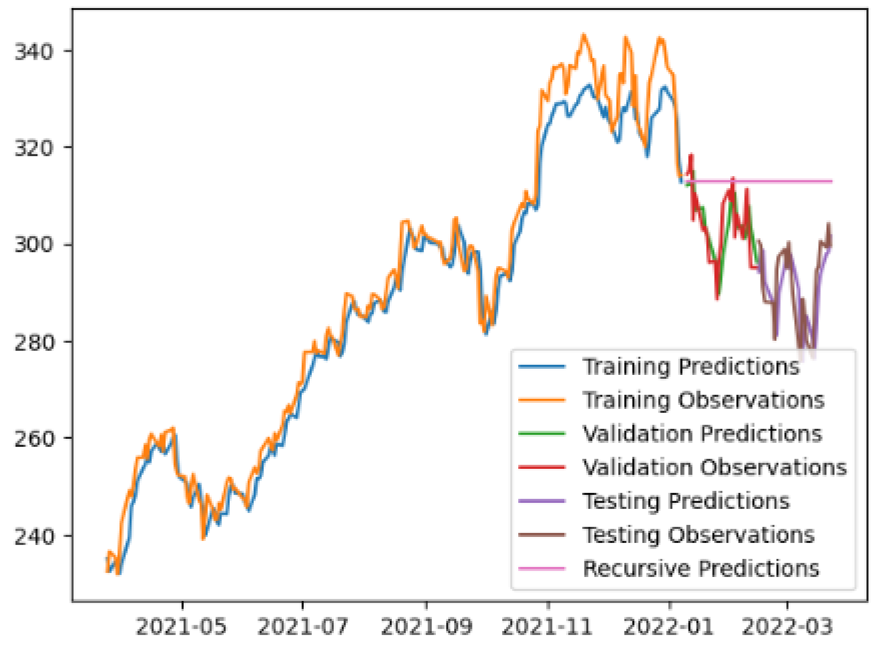


Figure 23 - Same LSTM Neural Network when predicting long-term (Recursive Predictions) shows a flat line

This serves to prove that it is not the kind of algorithm or neural network that is paramount, but the approach we take when thinking about the problem. The model needs to be inherently built with assumptions about the market and learn from them and its prediction needs to be fit for purpose and divided into at least the usual constituent elements of trend, seasonality and exogenous variables (DATA+AI Summit 2022 (Databricks), 2022).

6 THEORETICAL AND MATHEMATICAL FOUNDATIONS

6.1 (S)ARIMA(X)

Theoretical Basis: ARIMA (AutoRegressive Integrated Moving Average) models are used for analyzing and forecasting time series data.

Mathematical Foundations:

S (P, D, Q) $_S$: Seasonal – aims to capture seasonal relationships expected. Small 's', e.g. 5 (days) or 12 (months), etc. Large letters are the Seasonal equivalent of the ARIMA parameters

AR (p): Autoregression - relationship between an observation and several lagged observations.

I (d): Differencing - making the time series stationary.

MA (q): Moving Average - relationship between an observation and a residual error from a moving average model applied to lagged observations.

6.1.1 AR – Autoregressive

The formula: $Yt=\phi 1Yt-1+\phi 2Yt-2+\ldots+\phi pYt-p+\epsilon t$

The autoregressive part expects that the model is linearly dependent on its own previous values and expresses that relationship in the mathematical equation above, which can be translated into plain speech:

Value of this time step (Yt) is made up of a number of parameters (ϕp) calculated against the previous timesteps ($Yt-p$) and what remains is white noise (ϵt).

6.1.2 I – Integrated (Differencing for stationarity)

The formula: $Y_{t'}=Y_t-Y_t-1$

This aspect of the model aims to separate the statistical components of the time series from its temporal properties. For example, Apple's stock price today is a cumulative value of its entire existence on the market. However, its changes each day are governed (or expected to be governed) by some statistical relationship.

6.1.3 MA – Moving Average

The formula: $Y_t=\epsilon_t+\theta_1\epsilon_{t-1}+\theta_2\epsilon_{t-2}+\ldots+\theta_q\epsilon_{t-q}$

The MA aspect of the model expects that the current value (Y_t) depends linearly on both current and various past values of the white noise error term (ϵ_t, also mentioned in AR section above).

6.1.4 S – Seasonal

The formula: $\phi_p(B)\Phi_P(B^s)(1-B)^d(1-B^s)^D Y_t=\theta_q(B)\Theta_Q(B^s)\epsilon_t$

The relationship being captured in the equation above can be described in plain speech as:

The non-seasonal and seasonal AR, I and MA components are all taken into account as well as the backshift and white noise.

6.2 LSTM NEURAL NETWORK

LSTM neural networks are a type of RNN networks designed specifically to capture long-term dependencies in sequential data. This is done by dealing with the problem of vanishing gradients, whereby the backpropagation would deliver exceedingly small loss functions.

LSTMs use an architecture of memory cells and gating mechanisms, namely an input gate, a forget gate and an output gate. The hidden state from the previous time step is incorporated into current time step.

6.2.1 Input Gate

The formula: $i_t=\sigma(W_i \cdot [h_{t-1},x_t]+b_i)$

This gate helps establish what new information should enter the cell state

6.2.2 Forget Gate

The formula: $f_t=\sigma(W_f \cdot [h_{t-1},x_t]+b_f)$

This gate helps establish what part of the information in the previous cell state should be forgotten.

6.2.3 Candidate Cell State

The formula: $C_{\sim t}=\tanh(W_C \cdot [h_{t-1},x_t]+b_C)$

This step helps create a new cell state, that is then merged with the existing cell state.

6.2.4 Remaining elements

What follows is an update to the cell state, which combines both the old and new information. This is then passed to the Output gate, which helps decide which part of the current state will form next cell’s hidden state. Finally the Hidden State of the SLTM unit is established.

7 SOFTWARE ENGINEERING PRINCIPLES

(around 400 words)

Imagine one of the models developed in Task 2 was to be embedded as an intelligent component as part of a larger system with interfaces to other system components.

- Provide an architectural diagram and describe the wider system. Justify the components.
- Provide a plan to develop the system based on software engineering . The plan may be placed in an appendix and referenced here
- Discuss how the data science team might work with software engineers to develop suitable testing and documentation processes

7.1 ARCHITECTURAL DIAGRAM AND SYSTEM DESCRIPTION

A larger system, of which these forecasting models will be a part, requires comprehensive architecture to establish a coherent investment decision support system. Such system will be made up of data processing, analysis and user interaction components, further broken down into Data Ingestion, Preprocessing, Prediction Engine, User Interface and a Reporting Module.

7.1.1 Data ingestion

Here, real-time data can be gathered from various sources, such as order books, trade volumes and price movements. A continuous and efficient flow of data into the system needs to be safeguarded, as it is paramount to accurate and timely analysis.

7.1.2 Preprocessing

Here, steps such as cleaning, normalization and feature extraction would happen, which may be dependent on details of the data stream (e.g. varied from stock to stock). Advanced techniques will need to be implemented, such as dimensionality reduction and outlier detection to help prevent the model breaking down without the end-user even realising (it would still churn out predictions but based on bad data).

7.1.3 prediction engine

This would be the heart of the system, where models such as ARIMA or LSTM neural networks, or ideally an ensemble of various expert models and indicators is used to handle both short and long-term forecasting.

7.1.4 user interface

Predictions generated need to be displayed in a real-time dashboard, allowing users monitoring of trading activities and visualisation of detected anomalies. An ideal solution would also offer a white box approach that would allow users to deduce why the model is recommending a specific action and what ground truth it used to make the recommendation.

7.1.5 reporting module

Finally, reports would be generated for the compliance and governance teams to investigate and inform relevant parties, such as stakeholders or regulatory bodies.

7.2 DEVELOPMENT PLAN

1. Requirements Gathering

   Engage with stakeholders to understand their needs and define the problem as well as desired system's functional and non-functional requirements.

2. System Design and Architecture
   A detailed architectural blueprint must be developed, including flow between components and their interactions.
3. Model Development and Training
   Historical data is used to train the forecasting model. Extensive testing is carried out to optimize the model including hyperparameter tuning.
4. Integration with Existing Systems
   Seamless integration with existing infrastructure, data sources and interfaces. APIs are designed for data exchange
5. Testing (Unit, Integration, UAT)
   Thorough testing conducted at multiple levels
6. Deployment and Monitoring
   System is deployed to production and monitored. Monitoring tools are required here to detect issues real-time
7. Maintenance and Updates
   Maintenance plan must address any bugs as well as perform regular updates and continuous improvement of the system based on feedback and best practices.

8 REFERENCES

9 BIBLIOGRAPHY

DATA+AI Summit 2022 (Databricks). (2022, 07 19). *Nixtla: Deep Learning for Time Series Forecasting*. Retrieved from Youtube Summit Recording: https://youtu.be/i7JNt5qN2Sg?si=Gzr7_FVRpeoHxHy_

Hong, Z. (2024, 07 08). *Medium.* Retrieved from Anomaly Detection in Time Series Data using LSTM Autoencoders: https://medium.com/@zhonghong9998/anomaly-detection-in-time-series-data-using-lstm-autoencoders-51fd14946fa3

McConville, R., Santos-Rodriguez, R., Piechocki, R. J., & Craddock, I. (2019, 08 16). *N2D: (Not Too) Deep Clustering via Clustering the Local Manifold of an Autoencoded Embedding*. Retrieved from paperswithcode.com: https://paperswithcode.com/paper/n2dnot-too-deep-clustering-via-clustering-the

Rubsamen, R. (2022, 05 11). *Selecting a Stock Market Data (Web) API: Not So Simple*. Retrieved from PortfolioOptimizer.io: https://portfoliooptimizer.io/blog/selecting-a-stock-market-data-web-api-not-so-simple/

10 APPENDIX A APRENTICESHIP STANDARDS

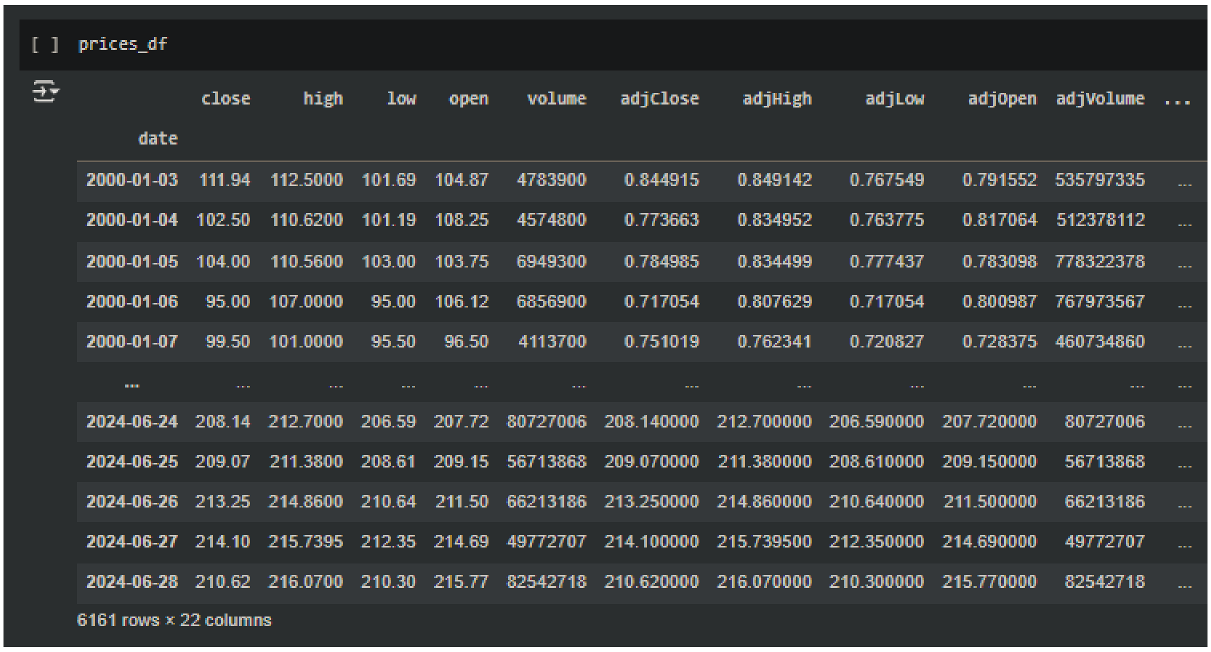
State where in the report or carrying out of the assignment you have addressed the standards shown in the table below.

List of applicable Standards from the Apprenticeship Specification (Including short form ID code)	
Knowledge	**How/where addressed?**
K3: How to apply advanced statistical and mathematical methods to commercial projects	Addressed in sections 6.1 to 6.4 where various statistical and machine learning methods such as EMA, ARIMA, SARIMA, and LSTM neural networks are applied to commercial problems like stock price prediction and anomaly detection.
K7: How to solve problems and evaluate software solutions via analysis of test data and results from research, feasibility, acceptance, and usability testing	Discussed in section 7, where the performance of ARIMA and LSTM models are evaluated using metrics like MAE, RMSE, and visual comparisons of predicted vs. actual data
K18: The programming languages and techniques applicable to data engineering	Covered in section 7.2 where Python and its libraries like TensorFlow are used for developing LSTM models, and in section 7.1 for ARIMA model development using Python.
K19: The principles and properties behind statistical and machine learning methods	Elaborated in section 6, which details the theoretical basis and applications of various algorithms such as EMA, ARIMA, SARIMA, and LSTM neural networks.
K22: The relationship between mathematical principles and core techniques in AI and data science within the organisational context	Explored in section 8, where the mathematical foundations of ARIMA and LSTM models are discussed, and their application in a financial services context is demonstrated.
K23: The use of different performance and accuracy metrics for model validation in AI projects	Addressed in section 7.1.4 and 7.2.4, where model evaluation metrics such as MAE, RMSE, and visual assessment are used to validate ARIMA and LSTM models.
K25: Programming languages and modern machine learning libraries for commercially beneficial scientific analysis and simulation	Demonstrated in sections 7.1 and 7.2 through the use of Python, TensorFlow, and other relevant libraries for building and validating predictive models.
K26: The scientific method and its application in research and business contexts, including experiment design and hypothesis testing	Illustrated throughout section 7, which details the process of data collection, preprocessing, model training, and evaluation, reflecting the scientific method.
K27: The engineering principles used (general and software) to create new instruments and applications for data collection	Discussed in section 9, where the architectural design and system components for an investment decision support system are described, including data ingestion, preprocessing, and prediction

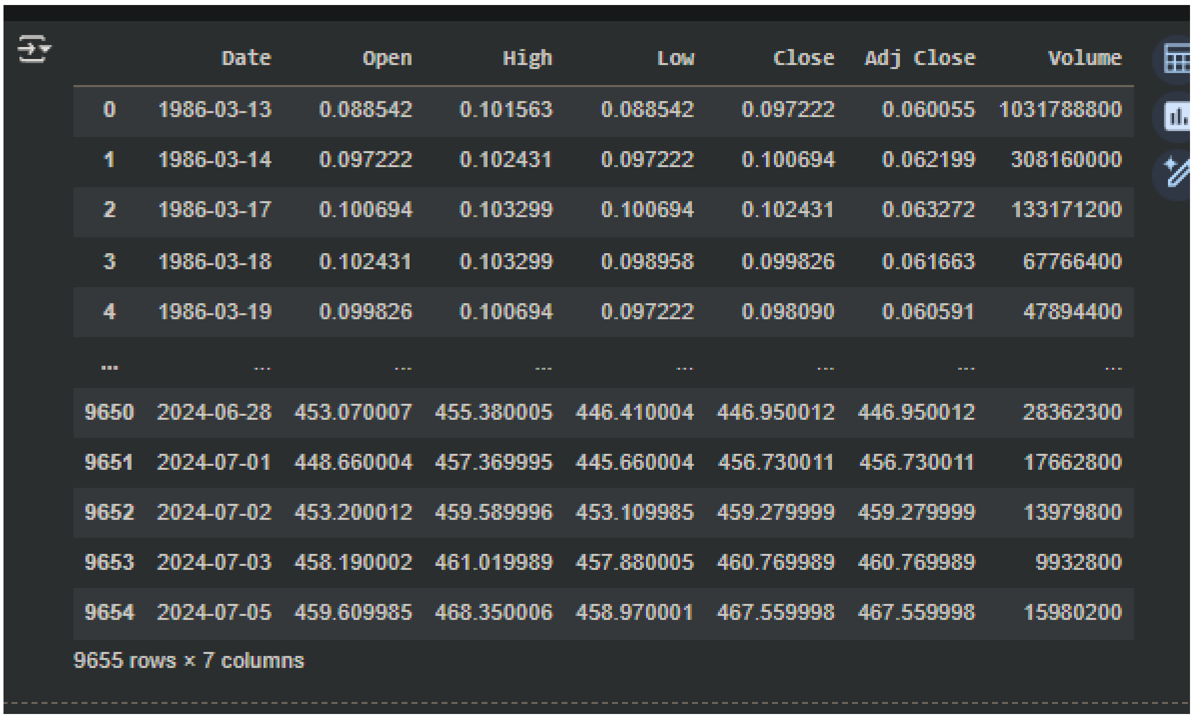
Skills	
S2: Independently analyse test data, interpret results, and evaluate the suitability of proposed solutions, considering current and future business requirements	Demonstrated in section 7, where the results of ARIMA and LSTM models are analyzed and interpreted to evaluate their suitability for stock price prediction and anomaly detection.
S11: Apply aspects of advanced maths and statistics relevant to AI and data science that deliver business outcomes	Covered in sections 6 and 7, where statistical methods and machine learning algorithms are applied to predict stock prices and detect anomalies, aiming for practical business applications.
S14: Work collaboratively with software engineers to ensure suitable testing and documentation processes are implemented.	Discussed in section 9, where the collaboration between data scientists and software engineers is outlined for developing, testing, and documenting the system.
S20: Design efficient algorithms for accessing and analysing large amounts of data, including Application Programming Interfaces (API) to different databases and data sets	Addressed in sections 6 and 7, where algorithms for time-series forecasting and anomaly detection are designed and implemented using historical stock data.
S25: Select and use programming languages and tools, and follow appropriate software development practices	Demonstrated in section 7, where Python and its libraries are used for model development, and in section 9, which outlines software development practices for integrating the models into a larger system.
Behaviours	
B3: Acts with integrity with respect to ethical, legal and regulatory ensuring the protection of personal data, safety and security	Discussed in section 6.5, which addresses the ethical, legal, and regulatory considerations in developing predictive models and handling financial data.
B6: Is comfortable and confident interacting with people from technical and non-technical backgrounds. Presents data and conclusions in a truthful and appropriate manner	Demonstrated throughout the report, especially in sections 6 and 7, where complex technical concepts and results are presented clearly and transparently for both technical and non-technical audiences.

11 APPENDIX B DATA

If possible, show a sample (10-20 rows) of the data used in your artefact here. You can generate the data as the purpose here is to show the type of data rather than the real data.



```
[ ]  prices_df

                close      high     low    open    volume   adjClose     adjHigh      adjLow     adjOpen  adjVolume ...
      date
2000-01-03     111.94  112.5000  101.69  104.87   4783900   0.844915    0.849142    0.767549    0.791552  535797335 ...
2000-01-04     102.50  110.6200  101.19  108.25   4574800   0.773663    0.834952    0.763775    0.817064  512378112 ...
2000-01-05     104.00  110.5600  103.00  103.75   6949300   0.784985    0.834499    0.777437    0.783098  778322378 ...
2000-01-06      95.00  107.0000   95.00  106.12   6856900   0.717054    0.807629    0.717054    0.800987  767973567 ...
2000-01-07      99.50  101.0000   95.50   96.50   4113700   0.751019    0.762341    0.720827    0.728375  460734860 ...
       ...        ...       ...     ...     ...       ...        ...         ...         ...         ...        ... ...
2024-06-24     208.14  212.7000  206.59  207.72  80727006 208.140000  212.700000  206.590000  207.720000   80727006 ...
2024-06-25     209.07  211.3800  208.61  209.15  56713868 209.070000  211.380000  208.610000  209.150000   56713868 ...
2024-06-26     213.25  214.8600  210.64  211.50  66213186 213.250000  214.860000  210.640000  211.500000   66213186 ...
2024-06-27     214.10  215.7395  212.35  214.69  49772707 214.100000  215.739500  212.350000  214.690000   49772707 ...
2024-06-28     210.62  216.0700  210.30  215.77  82542718 210.620000  216.070000  210.300000  215.770000   82542718 ...
6161 rows × 22 columns
```



```
            Date        Open        High         Low       Close   Adj Close      Volume
   0  1986-03-13    0.088542    0.101563    0.088542    0.097222    0.060055  1031788800
   1  1986-03-14    0.097222    0.102431    0.097222    0.100694    0.062199   308160000
   2  1986-03-17    0.100694    0.103299    0.100694    0.102431    0.063272   133171200
   3  1986-03-18    0.102431    0.103299    0.098958    0.099826    0.061663    67766400
   4  1986-03-19    0.099826    0.100694    0.097222    0.098090    0.060591    47894400
 ...         ...         ...         ...         ...         ...         ...         ...
9650  2024-06-28  453.070007  455.380005  446.410004  446.950012  446.950012    28362300
9651  2024-07-01  448.660004  457.369995  445.660004  456.730011  456.730011    17662800
9652  2024-07-02  453.200012  459.589996  453.109985  459.279999  459.279999    13979800
9653  2024-07-03  458.190002  461.019989  457.880005  460.769989  460.769989     9932800
9654  2024-07-05  459.609985  468.350006  458.970001  467.559998  467.559998    15980200
9655 rows × 7 columns
```