Midterm Exam

Predicting Netflix Stock Prices

ARIMA Modeling and 30-Day Forecasting Using CRISP-DM Framework

Pranav Deo

University of the Cumberlands

MSDS-530-M50 - Fundamentals of Data Science

Dr. Kristoffer Roberts

June 30, 2023

Problem Statement

Predicting Netflix Stock Prices Using ARIMA Modeling and Forecasting for the Next 30 Days:

The financial market is characterized by high volatility and complexity, posing challenges for investors seeking accurate stock price predictions. Netflix, a prominent global entertainment streaming platform, has attracted significant attention from investors due to its market dominance and growth potential. Therefore, accurately predicting Netflix's stock prices is essential for investors to make informed decisions and optimize their returns.

Problem Description

This project aims to address the challenge of predicting Netflix's stock prices and forecasting their future values for the next 30 days. The following key challenges must be addressed:

1. Data Availability and Preprocessing: Acquiring and preprocessing historical stock price data for Netflix from 2018 to 2022 is the first challenge. This dataset should include pertinent attributes such as date and closing price. Employing data cleaning and preprocessing techniques is crucial to address missing values, outliers, and other data quality issues that could potentially impact the accuracy of the predictions.
2. Model Selection and Optimization: Selecting an appropriate modeling technique is crucial for accurate stock price prediction. The Autoregressive Integrated Moving Average (ARIMA) modeling technique is adopted in this project. Determining the optimal order of the ARIMA model (p, d, q) is a challenge that involves analyzing the

autocorrelation and partial autocorrelation functions of the data. It is necessary to find the best model parameters to achieve reliable predictions.

3. Model Training and Validation: Implementing the training phase of the ARIMA model requires careful consideration. Historical stock price values are used to train the model and capture the underlying patterns and trends. However, model validation is equally important to ensure accuracy and generalization. Appropriate validation techniques, such as cross-validation or train-test splits, are employed to evaluate the model's predictive capabilities.
4. Forecasting Future Stock Prices: Once the ARIMA model is trained and validated, it is employed to forecast Netflix's stock price values for the next 30 days. This involves extrapolating the learned patterns and trends from the historical data to predict future price movements. Accurately capturing any potential seasonality, trends, or abrupt market changes that may affect Netflix's stock prices is a challenge in this stage.

Methodology and Approach

To tackle the aforementioned challenges, this project follows the Cross Industry Standard Process for Data Mining (CRISP-DM) framework, which provides a structured and systematic approach to data mining projects. The methodology comprises the following stages:

1. Business Understanding: Understanding the business objectives and the significance of accurate stock price prediction for investors is the first step. Defining the problem statement and establishing success criteria guide the project and ensure alignment with stakeholder requirements.

2. Data Understanding: Acquiring historical stock price data for Netflix from 2018 to 2022 is essential. Exploratory data analysis (EDA) is performed to understand the data's characteristics, trends, and distribution. This stage involves identifying and addressing missing values, outliers, and data quality issues during the preprocessing phase.

3. Data Preparation: Thorough data cleaning and preprocessing are crucial to ensure reliable predictions. Addressing missing values, outliers, and transforming variables, if necessary, are performed in this stage. The data is prepared in a suitable format for the ARIMA modeling technique to enable accurate forecasting.

4. Modeling: Selecting the appropriate ARIMA model order (p, d, q) is a critical step. Analyzing the autocorrelation and partial autocorrelation functions of the data guides the model selection process. Optimal model parameters are determined to optimize the predictive capabilities of the ARIMA model.

5. Model Training and Validation: The ARIMA model is trained using the historical data to capture underlying patterns and trends. Model validation is crucial to ensure accuracy and generalization. Techniques such as cross-validation or train-test splits are employed to evaluate the model's predictive performance and refine its parameters, if necessary.

6. Forecasting and Evaluation: Utilizing the trained and validated ARIMA model, Netflix's stock price values for the next 30 days are forecasted. The accuracy of the predictions is evaluated by comparing the forecasted values with the actual stock prices. Evaluation metrics such as mean absolute error (MAE) and root mean squared error (RMSE) are employed to measure the model's performance.

Predictive Modeling within the CRISP-DM Framework - Data Discovery and Understanding

The research employs a predictive modeling approach, specifically utilizing the Autoregressive Integrated Moving Average (ARIMA) modeling technique, to forecast the stock prices of Netflix. ARIMA, a well-established method for time series analysis, effectively captures the inherent patterns and trends within sequential data. By analyzing the historical stock price data spanning the years 2018 to 2022, the ARIMA model is trained to identify and comprehend the relationships and dependencies between past and future stock price values.

The initial step in the modeling process involves determining the appropriate ARIMA model order, represented as (p, d, q). These parameters, namely autoregressive, differencing, and moving average components, respectively, play a crucial role in constructing the model. To ascertain the optimal order, a thorough analysis of the autocorrelation function (ACF) and partial autocorrelation function (PACF) is conducted. This analysis aids in identifying the lag values that significantly contribute to the accurate prediction of future stock prices.

Upon establishing the ARIMA model order, the model is trained using the available historical stock price data. During the training phase, the model adapts its internal parameters to minimize the discrepancy between predicted and actual stock prices. By learning from past observations, the model gains insight into the underlying patterns and trends within the data.

Subsequently, the trained ARIMA model undergoes rigorous validation to assess its predictive capabilities. Common validation techniques, such as cross-validation or train-test splits, are employed to evaluate the model's performance across different subsets of the data. Evaluation metrics, including mean absolute error (MAE), root mean squared error (RMSE), and others, are utilized to measure the accuracy of the model's predictions. This validation process ensures the model's reliability and generalizability.

Once the ARIMA model is validated, it can be effectively utilized for forecasting future stock prices. Leveraging the acquired knowledge of past patterns and trends, the model generates predictions for the subsequent 30-day period. These predictions serve as valuable insights for investors, aiding them in making well-informed decisions regarding their investments in Netflix stocks. By anticipating potential price movements, investors can strategize and adapt their investment plans accordingly.

Deep Dive: Implementing ARIMA Modeling for Predicting Netflix Stock Prices

Step 1:

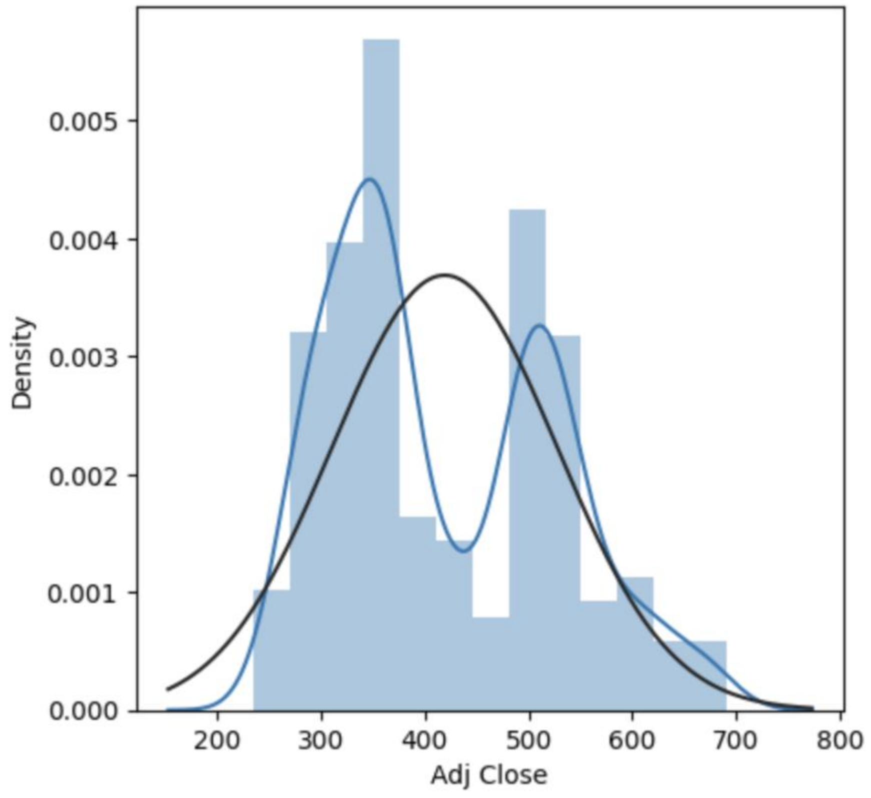
- Importing Libraries: The code begins by importing the necessary libraries: pandas, numpy, seaborn, matplotlib.pyplot, and statsmodels. These libraries provide various functionalities for data manipulation, visualization, statistical analysis, and time series modeling.
- Setting up Visualization: The line "%matplotlib inline" enables inline plotting within the Jupyter Notebook or Colab environment. It ensures that the generated plots are displayed directly below the code cells.
- Importing Dependencies: The code imports additional dependencies, such as scipy.stats, scipy, and statsmodels.api, to support statistical calculations, QQ plots, and time series analysis.

- Importing Data: The line "from google.colab import files" suggests that the code is running in a Colab environment. It enables file uploading capabilities. Subsequently, the code uploads a file using the "files.upload()" method. The uploaded file is expected to be in Excel format.
- Loading Data: The code reads the uploaded Excel file, assuming its name is 'NFLX.xlsx', using the pandas function "pd.read_excel()". The loaded data is stored in the variable "midterm_df".
- Displaying Data: Finally, the code displays the first few rows of the loaded data using the "head()" function. This step allows the user to inspect the data and verify its correctness.

Step 2:

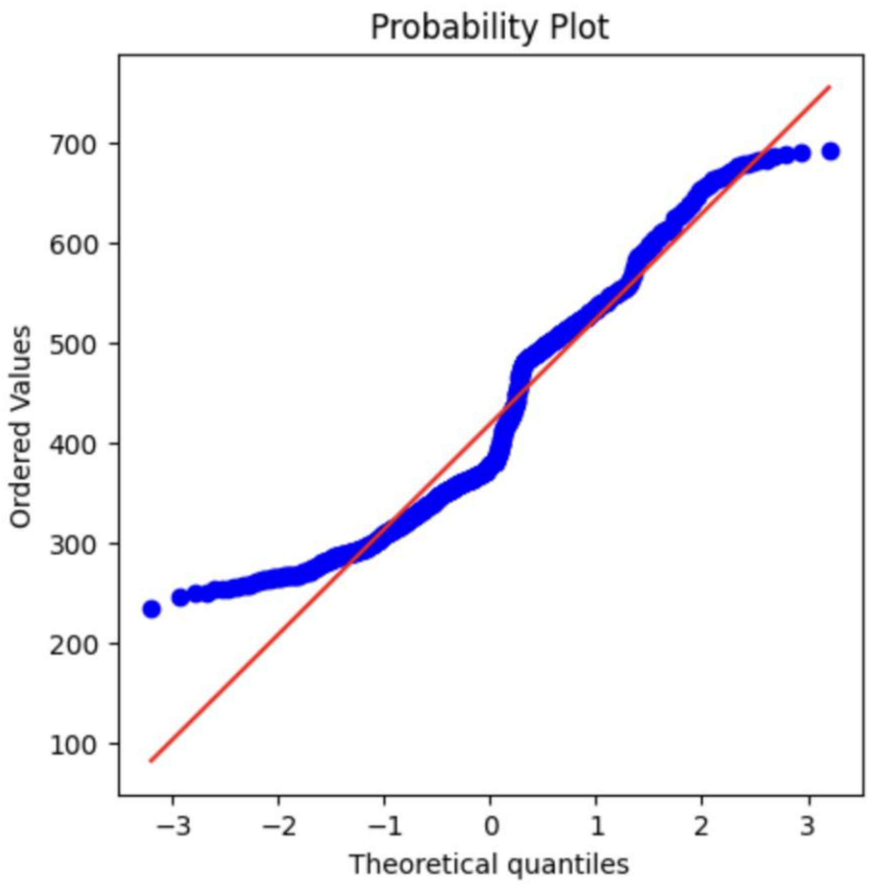
- Data Manipulation: The line "midterm_df = midterm_df.drop(['Open', 'High', 'Low', 'Close', 'Volume'], axis=1)" drops unnecessary columns ('Open', 'High', 'Low', 'Close', 'Volume') from the 'midterm_df' DataFrame. This step removes the unwanted data, retaining only the 'Adj Close' column.
- Data Visualization: To visualize the distribution of the 'Adj Close' column, the code uses the seaborn library (imported as 'sb') and the matplotlib library (imported as 'plt').
    - The line "sb.distplot(midterm_df['Adj Close'], fit=norm)" creates a distribution plot using the 'distplot' function from seaborn. It displays the histogram of the 'Adj Close' values and fits a normal distribution curve to the data. This plot provides insights into the shape and skewness of the distribution.

- The line "plt.show()" displays the generated distribution plot.



- QQ Plot (Quantile-Quantile Plot): A QQ plot, short for Quantile-Quantile plot, is a graphical tool used to assess the distributional characteristics of a dataset. It compares the quantiles of the dataset against the quantiles of a theoretical distribution, typically a standard normal distribution
    - The line "fig = plt.figure()" creates a new figure to display the QQ plot.
    - The line "res = stats.probplot(midterm_df['Adj Close'], plot=plt)" generates the QQ plot using the 'probplot' function from scipy.stats. It compares the quantiles of the 'Adj Close' data against the theoretical quantiles of a normal distribution. The resulting plot helps assess whether the data follows a normal distribution.

- The line "plt.show()" displays the QQ plot.

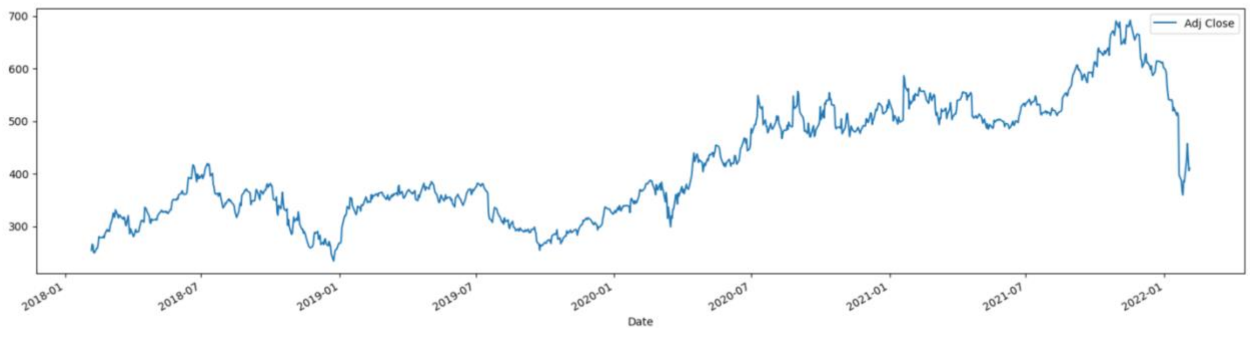


Step 3:

- Grouping and Summing Data: The line "midterm_df = midterm_df.groupby('Date')['Adj Close'].sum().reset_index()" groups the data in the 'midterm_df' DataFrame by the 'Date' column and calculates the sum of the 'Adj Close' values for each unique date. The result is stored back into the 'midterm_df' DataFrame.

- Converting Date Column to Datetime: The line "midterm_df.Date = pd.to_datetime(midterm_df.Date)" converts the 'Date' column from a string format to a datetime format using the 'pd.to_datetime()' function from the pandas library. This conversion ensures that the 'Date' column is recognized as a datetime type, allowing for proper handling of dates in subsequent operations.

- Displaying the Updated DataFrame: The line "midterm_df.head()" displays the first few rows of the updated 'midterm_df' DataFrame. This step allows the user to inspect the transformed data and verify the changes.
- Setting Date as Index: The line "midterm_df.set_index(['Date'], inplace=True)" sets the 'Date' column as the index of the 'midterm_df' DataFrame using the 'set_index()' function. This indexing facilitates time-based analysis and visualization.
- Plotting the Data: The line "midterm_df.plot(figsize=(20,5))" generates a line plot of the 'midterm_df' DataFrame, displaying the time series data. The 'figsize' parameter sets the size of the plot to (20,5), specifying the width and height of the figure.
- Displaying the Plot: The line "plt.show()" displays the generated plot, allowing for visual inspection of the time series data.

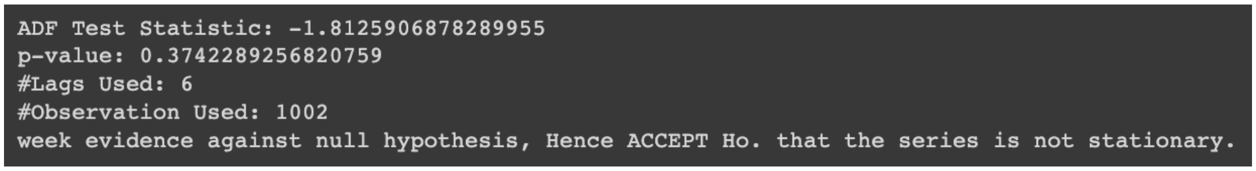


Step 4:

- Importing the Required Library: The line "from statsmodels.tsa.stattools import adfuller" imports the 'adfuller' function from the 'stattools' module within the 'statsmodels.tsa' library. This function is used to perform the ADF test.
- Defining the ADF Test Function: The code defines a custom function named 'adfuller_test' that takes a time series data, 'trends', as input. This function performs the ADF test using the 'adfuller' function and displays the results.
- ADF Test and Output: The line "adfuller_test(midterm_df['Adj Close'])" calls the 'adfuller_test' function, passing the 'Adj Close' series of the 'midterm_df' DataFrame as the argument.

Interpretation of the Output:

The output of the ADF test is displayed with the following information:



```
ADF Test Statistic: -1.8125906878289955
p-value: 0.3742289256820759
#Lags Used: 6
#Observation Used: 1002
week evidence against null hypothesis, Hence ACCEPT Ho. that the series is not stationary.
```

- ADF Test Statistic: This value, -1.8125906878289955 in this case, represents the test statistic of the ADF test. It is used to determine the stationarity of the series. In general, a more negative (lower) value indicates stronger evidence for stationarity.
- p-value: The p-value, 0.3742289256820759 in this case, represents the probability of obtaining the observed ADF test statistic. It is used to assess the statistical significance of the

ADF test. In hypothesis testing, if the p-value is less than the chosen significance level (commonly 0.05), there is strong evidence to reject the null hypothesis.

- #Lags Used: This value, 6 in this case, denotes the number of lag observations used in the ADF test.
- #Observation Used: This value, 1002 in this case, denotes the total number of observations used in the ADF test.

Based on the p-value, the code determines whether to reject or accept the null hypothesis. In this case, since the p-value (0.3742289256820759) is greater than the significance level (0.05), there is weak evidence against the null hypothesis. Therefore, the code concludes that the series is not stationary.

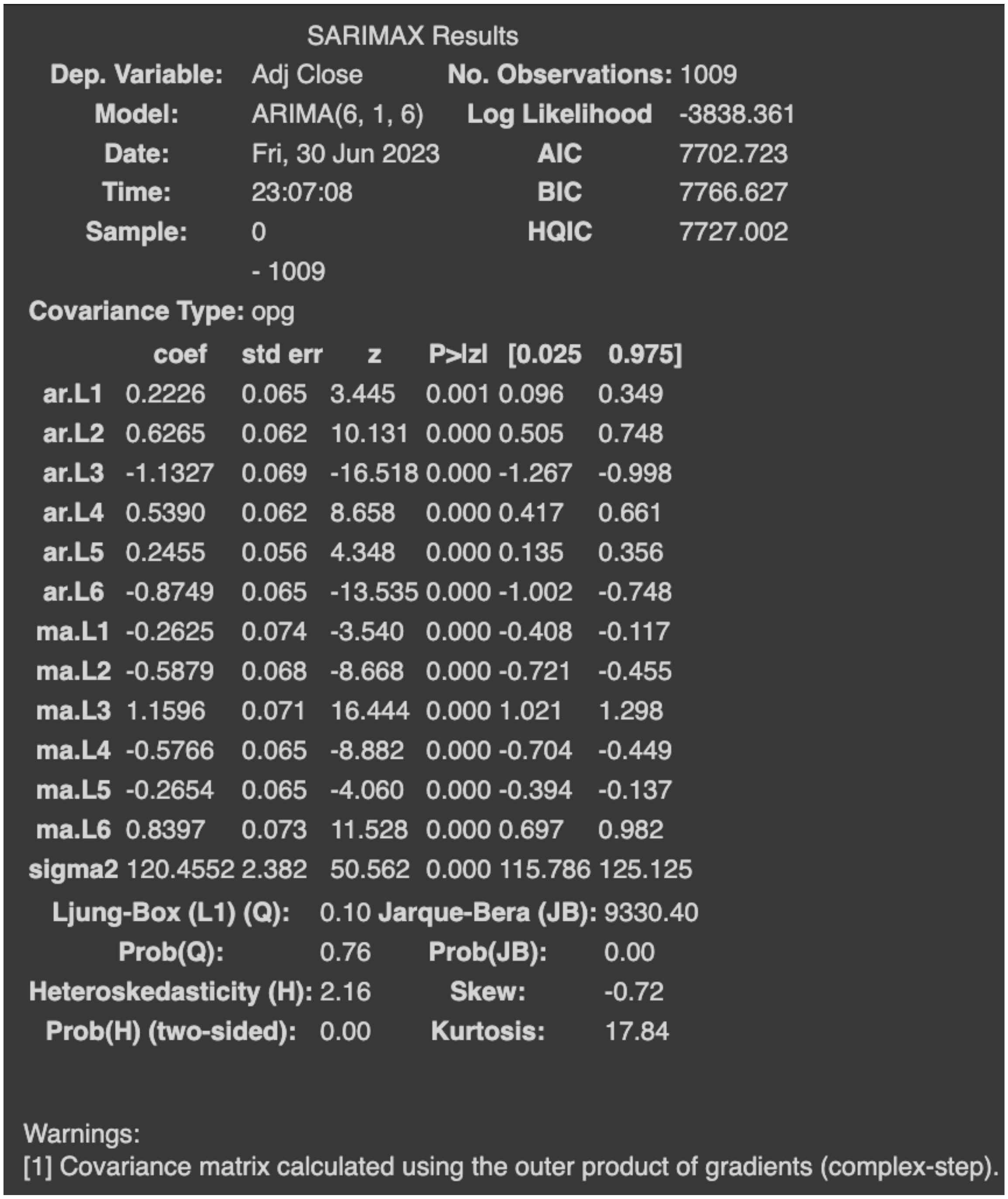
Interpreting the ADF test results helps determine the stationarity of the time series data. In this case, the series is determined to be non-stationary based on the ADF test results.

Step 5:

- Importing Libraries and Suppressing Warnings: The code imports the necessary libraries: 'ARIMA' from 'statsmodels.tsa.arima.model' and 'warnings'. The 'warnings.filterwarnings("ignore")' line suppresses any warning messages that might occur during the execution of the code.
- Looping Over Possible Values of p and q: The nested for loop iterates through the values of p (ranging from 1 to 6) and q (also ranging from 1 to 6). These values are used to create different ARIMA models with varying orders.

- Creating and Fitting ARIMA Models: Inside the loop, an ARIMA model is instantiated with the specified order of autoregressive (AR) and moving average (MA) terms. The 'ARIMA' model from 'statsmodels' is used, and the 'order' parameter is set accordingly using the loop variables 'i' and 'j'.
- Fitting the Model and Printing Results: The ARIMA model is fitted to the 'Adj Close' series of the 'midterm_df' DataFrame using the 'fit()' function. The resulting model is stored in the 'result' variable. The AIC value of the model is then obtained using 'result.aic'.
- Displaying the AIC and Model Order: The AIC value, along with the order of the AR and MA terms, is printed for each model iteration using the 'print()' function and formatted string (f-string).

By iterating over different combinations of p and q values, the code aims to find the combination that results in the lowest AIC. The AIC serves as a criterion for model selection, with lower AIC values indicating better model fit. Analyzing the AIC values helps identify the most optimal values of p and q for the ARIMA model in order to achieve better forecasting performance.



```
                         SARIMAX Results
 Dep. Variable:   Adj Close          No. Observations: 1009
     Model:       ARIMA(6, 1, 6)      Log Likelihood   -3838.361
      Date:       Fri, 30 Jun 2023          AIC         7702.723
      Time:       23:07:08                  BIC         7766.627
    Sample:       0                         HQIC        7727.002
                  - 1009
Covariance Type: opg
            coef     std err    z      P>|z|  [0.025    0.975]
  ar.L1   0.2226     0.065    3.445    0.001  0.096     0.349
  ar.L2   0.6265     0.062    10.131   0.000  0.505     0.748
  ar.L3   -1.1327    0.069    -16.518  0.000  -1.267    -0.998
  ar.L4   0.5390     0.062    8.658    0.000  0.417     0.661
  ar.L5   0.2455     0.056    4.348    0.000  0.135     0.356
  ar.L6   -0.8749    0.065    -13.535  0.000  -1.002    -0.748
  ma.L1   -0.2625    0.074    -3.540   0.000  -0.408    -0.117
  ma.L2   -0.5879    0.068    -8.668   0.000  -0.721    -0.455
  ma.L3   1.1596     0.071    16.444   0.000  1.021     1.298
  ma.L4   -0.5766    0.065    -8.882   0.000  -0.704    -0.449
  ma.L5   -0.2654    0.065    -4.060   0.000  -0.394    -0.137
  ma.L6   0.8397     0.073    11.528   0.000  0.697     0.982
 sigma2   120.4552   2.382    50.562   0.000  115.786   125.125
   Ljung-Box (L1) (Q):      0.10 Jarque-Bera (JB): 9330.40
        Prob(Q):            0.76     Prob(JB):     0.00
Heteroskedasticity (H):     2.16      Skew:        -0.72
  Prob(H) (two-sided):      0.00     Kurtosis:     17.84


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

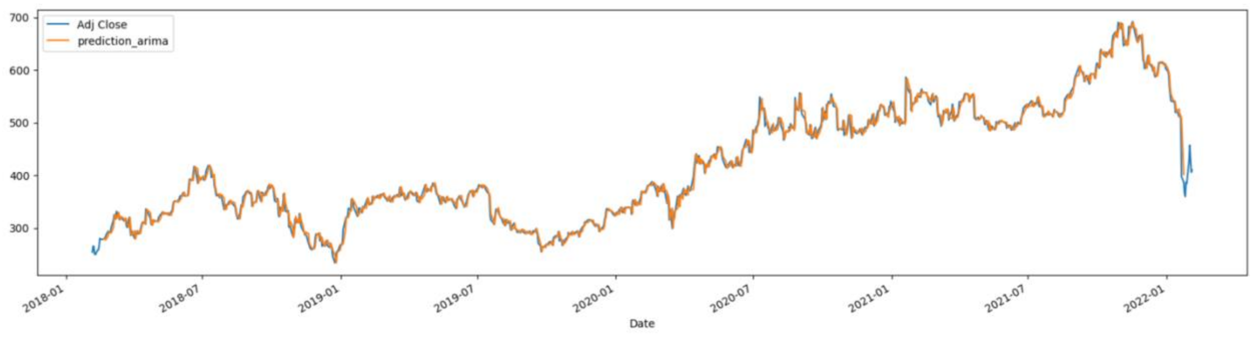
Step 6:

The provided code snippet focuses on plotting the results of the ARIMA prediction for the 'Adj Close' series in the 'midterm_df' DataFrame. Let's break it down:

- Generating ARIMA Predictions: The line "midterm_df['prediction_arima'] = result.predict(start=10, end=999)" generates the ARIMA predictions using the fitted ARIMA model stored in the 'result' variable. The 'predict()' function is used to generate predictions for the specified time period, starting from the 10th index and ending at the 999th index. The predictions are assigned to a new column named 'prediction_arima' in the 'midterm_df' DataFrame.

- Plotting the Actual and Predicted Values: The line "midterm_df[["Adj Close", "prediction_arima"]].plot(figsize=(20, 5))" generates the plot for the 'Adj Close' series and the ARIMA predictions. The 'plot()' function is called on the subset of the 'midterm_df' DataFrame, including the 'Adj Close' column and the 'prediction_arima' column. The 'figsize' parameter is set to (20, 5) to control the size of the plot.

The resulting plot displays both the actual 'Adj Close' values and the ARIMA predictions. By comparing the actual and predicted values, the plot provides a visual representation of how well the ARIMA model fits the data and captures the underlying patterns and trends.



Leveraging ARIMA Modeling to Enhance Investment Decision-Making: Predicting Future Netflix Stock Prices

The primary objective of the ARIMA model is to provide accurate predictions of future Netflix stock prices. The model captures the patterns and trends inherent in the historical stock price data by analyzing the historical stock price data. This allows it to generate forecasts for the stock prices in the coming days, providing valuable insights for investors.

The ARIMA model examines the relationships and dependencies between past and future stock price values. By considering the autoregressive (AR) and moving average (MA) components, the model considers the historical price patterns to estimate future prices. This analysis helps uncover meaningful relationships in the data, assisting in making informed investment decisions.

By providing reliable predictions, the ARIMA model equips investors with valuable information to make informed decisions regarding their investments in Netflix stocks. By anticipating potential price movements, investors can adjust their investment strategies accordingly, maximizing potential returns and minimizing risks.

Accurate stock price predictions obtained from the ARIMA model can be a valuable risk management tool. By understanding the expected future price movements, investors can assess the potential risks associated with their investment positions. This knowledge helps set risk management strategies, such as determining appropriate entry and exit points, implementing stop-loss orders, or adjusting portfolio allocations.

The ARIMA model allows investors to validate and refine their investment strategies. By comparing the predicted stock prices with the actual prices, investors can evaluate the effectiveness of their trading strategies or portfolio allocations. This feedback loop enables investors to continuously learn, adapt, and refine their investment approaches based on the performance of the ARIMA model.

Furthermore, the ARIMA model contributes to financial decision-making by providing a quantitative foundation for evaluating investment opportunities. Its predictions assist investors, financial analysts, and fund managers in assessing the potential profitability and risks of investing in Netflix stocks. The model's insights facilitate the evaluation of investment alternatives and support data-driven decision-making.

Conclusion

In conclusion, the application of the ARIMA modeling approach to predict Netflix stock prices offers valuable insights and enhances investment decision-making. By leveraging time series analysis techniques, the ARIMA model captures underlying patterns and trends in historical stock price data, enabling accurate forecasts of future prices. Through the systematic implementation of the CRISP-DM framework, the model ensures a structured and rigorous approach to data mining projects, enhancing the reliability and robustness of the predictions.

The ARIMA proposed model accomplishes several key objectives. Firstly, it provides investors with accurate forecasts of future Netflix stock prices. These predictions enable investors to anticipate potential price movements and make informed decisions regarding their investments. By having access to reliable predictions, investors can adjust their investment strategies accordingly, maximizing potential returns and minimizing risks.

Furthermore, the ARIMA model identifies relationships and dependencies between past and future stock price values. By considering the autoregressive (AR) and moving average (MA) components, the model uncovers meaningful patterns in the data, assisting investors in making informed investment decisions. The ability to understand the relationships between variables contributes to a deeper understanding of market dynamics and assists in identifying potential trading opportunities.

The ARIMA model also supports risk management efforts. By providing accurate predictions of future stock prices, investors can assess the potential risks associated with their investment positions. This knowledge helps in setting risk management strategies, such as determining appropriate entry and exit points, implementing stop-loss orders, or adjusting portfolio allocations. The ARIMA model contributes to a more informed and proactive approach to managing investment risk.

Moreover, the ARIMA model validates and enhances investment strategies. By comparing the predicted stock prices with the actual prices, investors can evaluate the effectiveness of their trading strategies or portfolio allocations. This feedback loop allows investors to continuously learn, adapt, and refine their investment approaches based on the performance of the ARIMA model. The model serves as a tool for assessing the viability of different investment strategies and helps investors make data-driven decisions.

In addition to empowering individual investors, the ARIMA model contributes to the broader field of financial decision-making. Its predictions assist financial analysts, fund managers, and other market participants in evaluating investment opportunities. By incorporating the insights from the ARIMA model, financial professionals can assess the potential profitability and risks associated with investing in Netflix stocks. The model's quantitative foundation enhances the evaluation of investment alternatives and supports data-driven decision-making in the financial industry.

In conclusion, the application of ARIMA modeling to predict Netflix stock prices offers significant advantages to investors. Through accurate forecasting, identification of relationships, and support for risk management, the ARIMA model enables investors to make informed decisions, adjust their investment strategies, and maximize their investment outcomes. By leveraging the insights provided by the ARIMA model, investors can navigate the dynamic landscape of stock markets more effectively and make better-informed investment decisions.

References

Modelling common-sense knowledge in automatic irony detection | Faculty of Arts and Philosophy - Research Portal. https://research.flw.ugent.be/en/projects/modelling-common-sense-knowledge-automatic-irony-detection

Complete Guide to Mastering Machine Learning | Beginner to Pro. ». https://bravotecharena.com/complete-guide-to-mastering-machine/

Jainilcoder. (n.d.). Netflix Stock Price Prediction [Data set]. Kaggle. CC0: Public Domain. Retrieved from https://www.kaggle.com/datasets/jainilcoder/netflix-stock-price-prediction

Molak, A. (2023). Causal Inference and Discovery in Python: Unlock the secrets of modern causal machine learning with DoWhy, EconML, PyTorch and more. Ajit Jaokar (Foreword).

Géron, A. (2022). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (3rd ed.).

Prosise, J. (2022). Applied Machine Learning and AI for Engineers: Solve Business Problems That Can't Be Solved Algorithmically (1st ed.).