UDACITY DATA SCIENCE NANO DEGREE CAPSTONE PROJECT

INVESTMENT &TRADING (AAPL)

Project Overview

Investment firms, hedge funds, and individual investors have long utilized financial models to gain insights into market behavior and execute profitable trades. The abundance of historical stock prices and company performance data presents a rich source for machine learning algorithms to analyze and predict market trends.
In this project, the objective is to develop a stock price predictor that processes daily trading data over a specific date range and provides projected estimates for given query dates.

The input data includes multiple metrics such as opening price (Open), highest price (High), trading volume (Volume), and closing price adjusted for stock splits and dividends (Adjusted Close). The goal is to predict the Adjusted Close price accurately.

A reliable stock price predictor can offer numerous benefits to investors, including:

1. Enhanced Decision Making: By providing accurate predictions of stock prices, investors can make more informed decisions about buying, holding, or selling stocks.

2. Risk Management: Predictive models can help in assessing the risk associated with specific stocks and enable investors to diversify their portfolios effectively.

3. Optimized Investment Strategies: Investors can develop and refine their investment strategies based on predictive insights, aiming for higher returns.

4. Market Timing: Accurate predictions can assist investors in timing the market better, identifying optimal entry and exit points for their trades.

Input Data

For this stock price predictor, we utilize historical trading data of Apple Inc. (AAPL) from January 2019 to December 2023. The dataset includes multiple metrics such as:
1. Opening Price (Open): The price at which the stock starts trading at the beginning of the day.

2. Highest Price (High): The maximum price the stock reaches during the trading day.

3. Trading Volume (Volume): The number of shares traded during the day.

4. Adjusted Close Price (Adjusted Close): The closing price adjusted for stock splits and dividends, which our system aims to predict.

	Date	Open	High	Low	Close	Adj Close	Volume
0	2019-01-02	38.722500	39.712502	38.557499	39.480000	37.793777	148158800
1	2019-01-03	35.994999	36.430000	35.500000	35.547501	34.029232	365248800
2	2019-01-04	36.132500	37.137501	35.950001	37.064999	35.481930	234428400
3	2019-01-07	37.174999	37.207500	36.474998	36.982498	35.402943	219111200
4	2019-01-08	37.389999	37.955002	37.130001	37.687500	36.077839	164101200

While this project focuses on AAPL data, you can extend the analysis to other companies. For further exploration, refer to this document.

Strategy for Solving the Problem

Methodology:

The approach involves a comprehensive data-driven process starting from data cleaning and exploratory data analysis (EDA) to feature engineering, model selection, hyperparameter tuning, evaluation, and visualization. We utilized several machine learning models, including Ridge Regression with hyperparameter tuning, to forecast the Adjusted Close price. The chosen models and techniques are based on their suitability for time series forecasting and regression tasks.

Discussion of the Expected Solution

The proposed solution involves a structured workflow that integrates several key components to predict the Adjusted Close price of Apple Inc. (AAPL) stock. The workflow is divided into distinct stages, each with a specific purpose and set of operations, ensuring a comprehensive approach to the problem.

Data Collection and Preprocessing:

- Data Collection: Historical stock price data for AAPL from January 2019 to December 2023 is collected. This data includes metrics such as Open, High, Low, Close, Adjusted Close, and Volume.
- Data Preprocessing: This step involves cleaning the data by checking for and handling missing values, duplicates, and outliers. Feature engineering is also performed to create lag features (previous day prices) and moving averages (5-day and 10-day averages) to enhance the predictive power of the model.

Exploratory Data Analysis (EDA):

- Data Visualization: Various visualizations, including box plots and line plots, are created to understand the data distribution and identify trends and patterns in stock prices and trading volumes.
- Statistical Analysis: Summary statistics are computed to gain insights into the central tendency and dispersion of the data.

Feature Engineering:

- Lag Features: Lag features such as 'Adj Close Lag1', 'Adj Close Lag2', and 'Adj Close Lag3' are created to incorporate the influence of previous days' prices on the current day's price.
- Moving Averages: Moving averages (SMA_5 and SMA_10) are calculated to smooth out short-term fluctuations and highlight longer-term trends.

Model Selection and Training:

- Model Selection: Ridge Regression is selected due to its ability to handle multicollinearity and large feature sets effectively. It is suitable for regression tasks where the relationship between independent and dependent variables is linear.
- Model Training: The model is trained on the historical stock prices and engineered features. The training process involves fitting the model to the data to learn the underlying patterns and relationships.

Hyperparameter Tuning:

- RandomizedSearchCV: Hyperparameter tuning is performed using RandomizedSearchCV to find the optimal hyperparameters for the Ridge Regression model. This step involves searching over a range of hyperparameter values and selecting the ones that result in the best model performance.

Model Evaluation and Visualization:

- Performance Metrics: The model's performance is evaluated using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and R-squared. These metrics provide a comprehensive assessment of the model's accuracy and predictive power.
- Visualization: The predicted stock prices are visualized against the actual stock prices to provide a clear comparison. Additional visualizations such as residual plots and distribution of residuals help in diagnosing the model's performance and identifying areas for improvement.

Testing and Validation:

- Train-Test Split: The dataset is split into training and testing sets to validate the model's performance on unseen data. This step ensures that the model generalizes well to new data and avoids overfitting.
- Interval-based Performance Testing: The model's prediction accuracy is tested for query dates at different intervals after the training end date (e.g., 7 days, 14 days, 28 days) to evaluate its robustness and reliability over time.

Integration of Components:
Each component of the solution fits together to address the problem of predicting stock prices. Data preprocessing ensures high-quality input data, EDA provides insights and guides feature engineering, and feature engineering enhances the model's predictive power. Model selection and hyperparameter tuning ensure the best possible performance, while evaluation and visualization provide a comprehensive assessment of the model's accuracy and reliability.

Metrics with Justification

Evaluation Metrics:
To assess the performance of the stock price prediction model, we utilized several key metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and R-squared. Each of these metrics provides unique insights into the model's performance and its ability to predict stock prices accurately. Here's a detailed explanation of each metric and the justification for their use:

Mean Absolute Error (MAE):

- Definition: MAE measures the average magnitude of the errors between predicted and actual values, without considering their direction. It is the average of the absolute differences between predicted and actual values.
- Justification: MAE is simple to understand and interpret, providing a clear indication of the average error magnitude. It is particularly useful in finance, where understanding the average deviation from the actual stock price is crucial for assessing model performance.

Mean Squared Error (MSE):

- Definition: MSE measures the average of the squared differences between predicted and actual values. Squaring the errors penalizes larger errors more than smaller ones.
- Justification: MSE is widely used in regression tasks because it provides a clear measure of prediction accuracy. The squaring of errors emphasizes the importance of avoiding large errors, which can be critical in stock price prediction where large deviations can lead to significant financial losses.

Root Mean Squared Error (RMSE):

- Definition: RMSE is the square root of the MSE. It provides an error metric in the same units as the original data, making it more interpretable.
- Justification: RMSE is useful for understanding the model's prediction error in the context of the original data. It is particularly helpful when comparing models, as it gives an error measure in the same scale as the stock prices, making it easier to relate to actual stock price fluctuations.

Mean Absolute Percentage Error (MAPE):

- Definition: MAPE measures the average absolute percentage error between predicted and actual values. It expresses the error as a percentage, providing a relative measure of accuracy.
- Justification: MAPE is valuable for comparing prediction errors across different scales. In the context of stock price prediction, it allows investors to understand the error relative to the stock price, which is important for assessing the model's practical utility in financial decision-making.

R-squared (R^2):

- Definition: R-squared measures the proportion of the variance in the dependent variable that is predictable from the independent variables. It ranges from 0 to 1, with higher values indicating better model performance.
- Justification: R-squared provides a measure of how well the model explains the variability of the stock prices. In financial modeling, a high R-squared value indicates that the model captures a significant portion of the stock price movements, making it a useful metric for evaluating overall model effectiveness.

Choosing These Metrics:
The selection of these metrics is based on their ability to provide a comprehensive evaluation of the model's performance. Each metric offers different perspectives on the prediction accuracy and error characteristics, allowing for a thorough assessment of the model's strengths and weaknesses. This multi-faceted evaluation is crucial in the context of stock price prediction, where both the magnitude and direction of errors can have significant financial implications. By using a combination of these metrics, we can ensure a robust evaluation of the model's predictive capabilities, guiding further improvements and refinements.

EDA

Exploratory Data Analysis (EDA): This analysis enabled us to address key business questions such as: Trends in Adjusted Close Price Over Time, Annual High and Low Prices, Trading Volume Dynamics, Monthly Returns, and 30Day Rolling Averages which are explained below;

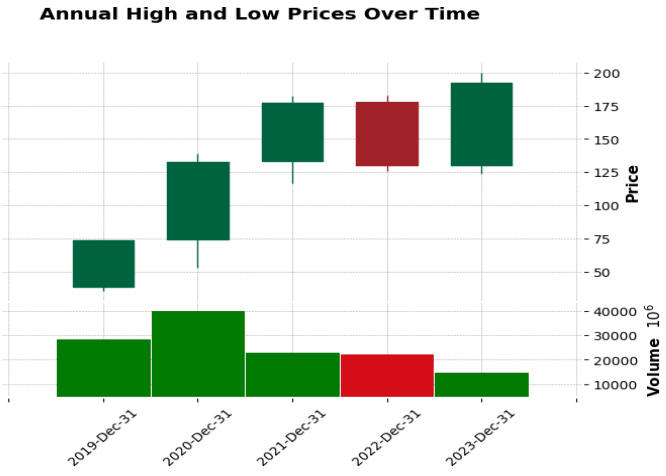
1. 

The graph illustrates the trend of Apple's Adjusted Close price from early 2019 to late 2023.

Over this period, the stock shows a consistent upward trajectory, with notable periods of volatility and significant growth, especially during 2020 and early 2021.

This indicates strong performance with fluctuations typical of a dynamic market. The later part of the chart shows increased volatility, reflecting market reactions to various factors.

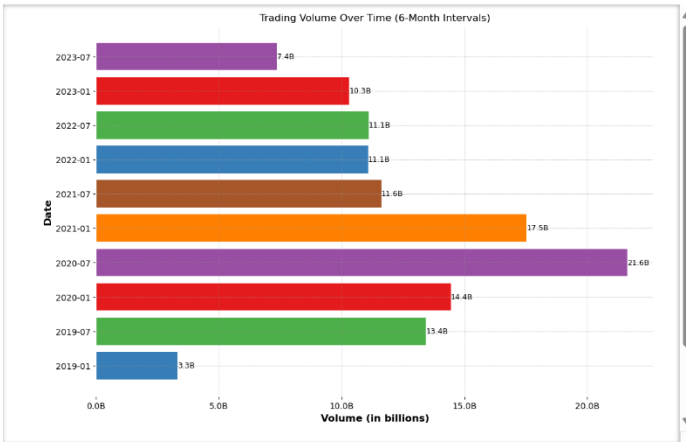
Overall, the stock has demonstrated substantial growth, suggesting positive long-term investor confidence.

2. 

The candlestick chart provides a yearly summary of Apple's stock performance from 2019 to 2023. Each candlestick represents one year, showing the open, high, low, and close prices, with green indicating an increase in price over the year and red indicating a decrease.

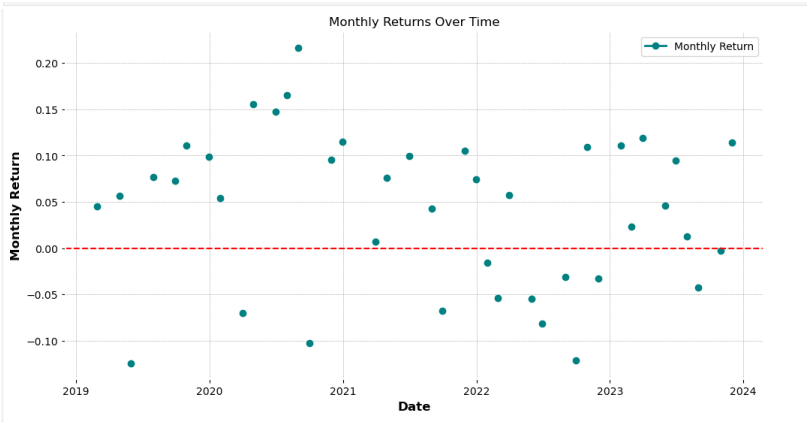
- 2019: A notable increase, starting at a lower value and closing higher.

- 2020: Significant growth, reflecting a strong upward trend.
- 2021: Continued upward momentum, though more stable.
- 2022: A slight decline, marking a year of correction.
- 2023: Recovery and further growth, nearing historical highs.

3. 

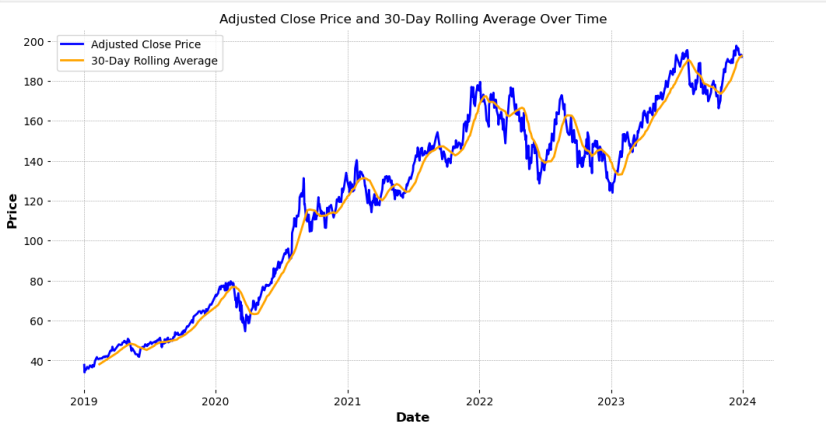
The horizontal bar chart displays Apple's trading volume at sixmonth intervals from January 2019 to July 2023. Key observations include:

- High Activity: Peaks in trading volume are observed in the second half of 2020 and the first half of 2021, reaching over 20 billion shares.
- Recent Trends: The trading volume slightly declined after early 2021, stabilizing around 1114 billion shares in subsequent periods.
- Volatility: Fluctuations in volume reflect varying market conditions and investor activities, with noticeable increases and decreases across different intervals.

4. 

The scatter plot represents Apple's monthly returns from 2019 to 2024. Key observations include:

- Positive Returns: Most months exhibit positive returns, with notable peaks around mid2020 and early 2021, indicating periods of strong performance.
- Negative Returns: Periods of negative returns are also evident, particularly around late 2020 and early 2022, reflecting market corrections or downturns.
- Volatility: The fluctuation in returns underscores the stock's volatility, with both substantial gains and losses over different months.

5. 

The line chart depicts Apple's adjusted close price and its 30day rolling average from 2019 to 2024. Key observations include:

- General Uptrend: The adjusted close price shows a general upward trend, with significant growth periods, especially from late 2019 to early 2021.
- Rolling Average Smoothing: The 30day rolling average (orange line) smooths out shortterm fluctuations, providing a clearer view of the overall trend and highlighting periods of price consolidation and growth.
- Volatility: Despite the overall growth, the stock experienced several periods of volatility, as indicated by the deviations between the adjusted close price and the rolling average.

Data Preprocessing

Data preprocessing is a critical step in preparing the dataset for modeling. It involves cleaning, transforming, and engineering features to ensure that the data is in the best possible shape for the machine learning algorithms. Here's a detailed outline of the steps taken in our project:

Data Cleaning:

- Checking for Missing Values

Purpose: Ensure that there are no gaps in the dataset that could affect the model's performance.
Action: We checked for missing values and found none in our dataset.

- Removing Duplicates:

Purpose: Prevent skewed results due to repeated data points.
Action: We checked for duplicate records and confirmed that there were none.

Data Transformation:

- Date Parsing:

Purpose: Ensure that the 'Date' column is in the correct format for time series analysis.
Action: Converted the 'Date' column to `datetime` format and set it as the index of the DataFrame to facilitate time-based operations.

- Feature Engineering:

Purpose: Create new features that can help the model better understand patterns in the data.
Actions:
- Lag Features: Created lag features for the 'Adj Close' column to incorporate the previous day's adjusted close prices. These features help the model understand trends over time.

```python
stock_data['Adj Close Lag1'] = stock_data['Adj Close'].shift(1)
stock_data['Adj Close Lag2'] = stock_data['Adj Close'].shift(2)
stock_data['Adj Close Lag3'] = stock_data['Adj Close'].shift(3)
```

- Moving Averages: Calculated 5-day and 10-day simple moving averages (SMA) for the 'Adj Close' column to smooth out short-term fluctuations and highlight longer-term trends.

```python
stock_data['SMA_5'] = stock_data['Adj Close'].rolling(window=5).mean()
stock_data['SMA_10'] = stock_data['Adj Close'].rolling(window=10).mean()
```

- Dropping NA Values:
- Purpose: Remove any rows that have missing values due to the creation of lag features and moving averages.
- Action: Dropped rows with any NA values resulting from the above operations to ensure the dataset is clean and complete.

```python
stock_data.dropna(inplace=True)
```

Feature Selection:
Purpose: Identify and retain only the most relevant features for modeling to reduce noise and improve model performance.
Action: Selected features such as 'Adj Close', 'Adj Close Lag1', 'Adj Close Lag2', 'SMA_5', and 'SMA_10' for the final model training.

Train-Test Split:
Purpose: Split the dataset into training and testing sets to evaluate the model's performance on unseen data.
Action: Split the data into training and testing sets with an appropriate ratio (e.g., 80% training and 20% testing).

```python
train_size = int(len(stock_data) * 0.8)
train_data, test_data = stock_data[:train_size], stock_data[train_size:]
X_train = train_data[['Adj Close Lag1', 'Adj Close Lag2', 'SMA_5', 'SMA_10']]
y_train = train_data['Adj Close']
X_test = test_data[['Adj Close Lag1', 'Adj Close Lag2', 'SMA_5', 'SMA_10']]
y_test = test_data['Adj Close']
```

Scaling:
Purpose: Standardize the features to have a mean of 0 and a standard deviation of 1. This step is essential for many machine learning algorithms to perform optimally.
Action: Applied standard scaling to the feature set.

```python
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

By meticulously following these data preprocessing steps, we ensured that the dataset was clean, well-structured, and enriched with relevant features, thus optimizing it for the subsequent modeling process. These preparations are crucial for building a robust and accurate stock price prediction model.

Modeling

In this project, we aimed to build a robust stock price predictor using historical data of Apple Inc. (AAPL). Our approach involved experimenting with multiple machine learning models and tuning their hyperparameters to achieve optimal performance. Below are the details of the models we used, the underlying algorithms, and specific considerations or modifications made during the process.

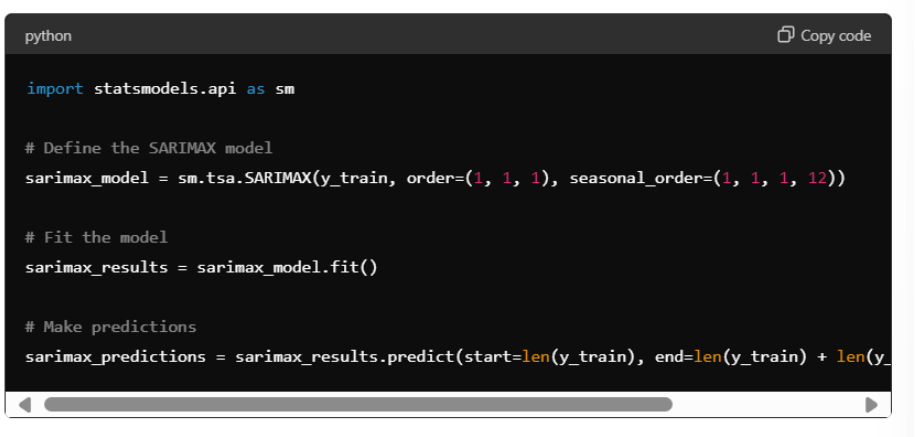
Models Used:
1. SARIMAX (Seasonal AutoRegressive Integrated Moving Average with eXogenous regressors)
2. Ridge Regression

1. SARIMAX Model

SARIMAX is a time series forecasting method that extends the ARIMA model to support seasonal components and exogenous variables. The components of the SARIMAX model include:

- AR (AutoRegressive) Terms: Represent the dependency between an observation and a number of lagged observations.
- I (Integrated) Terms: Represent the differencing of raw observations to make the time series stationary.
- MA (Moving Average) Terms: Represent the dependency between an observation and a residual error from a moving average model applied to lagged observations.
- Seasonal Components: Capture the seasonality of the data.

Implementation:



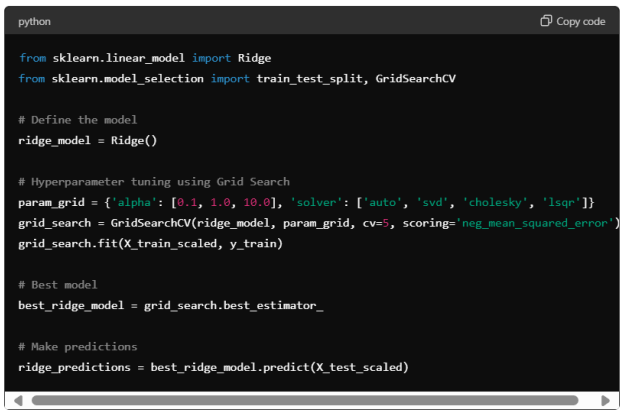
Considerations:

- We carefully selected the order and seasonal order parameters based on the dataset's characteristics.
- The model was trained on the training dataset and tested on the test dataset to evaluate its performance.

2. Ridge Regression

Ridge Regression is a linear regression model that includes L2 regularization. This technique helps prevent overfitting by penalizing large coefficients, making it particularly useful when dealing with multicollinearity or when the number of predictors is larger than the number of observations.

Implementation:



- We performed hyperparameter tuning using Grid Search to find the optimal values for `alpha` and `solver`.
- The features were scaled before training the model to ensure that the regularization term is applied consistently.

Performance Metrics
We evaluated the performance of the models using the following metrics:

- Mean Absolute Error (MAE): Measures the average magnitude of errors in a set of predictions, without considering their direction.
- Mean Squared Error (MSE): Measures the average of the squares of the errors, giving more weight to larger errors.
- Root Mean Squared Error (RMSE): The square root of the mean squared error, providing a measure of the average magnitude of the error.
- Mean Absolute Percentage Error (MAPE): Measures the accuracy of predictions as a percentage, which is useful for understanding the prediction error in relative terms.
- R-squared: Indicates the proportion of the variance in the dependent variable that is predictable from the independent variables.

Example Code for Evaluation:

```python
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Evaluate the Ridge model
mae = mean_absolute_error(y_test, ridge_predictions)
mse = mean_squared_error(y_test, ridge_predictions)
rmse = mse ** 0.5
mape = np.mean(np.abs((y_test - ridge_predictions) / y_test)) * 100
r2 = r2_score(y_test, ridge_predictions)

print(f"MAE: {mae}")
print(f"MSE: {mse}")
print(f"RMSE: {rmse}")
print(f"MAPE: {mape}")
print(f"R-squared: {r2}")
```

By experimenting with different models and tuning their hyperparameters, we aimed to identify the best-performing model for predicting the adjusted close prices of AAPL. The comprehensive evaluation metrics provided insights into the accuracy and robustness of each model.

Hyperparameter Tuning

Hyperparameter tuning is a critical step in the machine learning pipeline that involves finding the optimal set of hyperparameters for a given model to maximize its performance. Hyperparameters are model parameters that cannot be estimated from the data directly and need to be set before training the model. In this project, we used Grid Search and Random Search for hyperparameter tuning.

Techniques Used:
1. Grid Search
2. Random Search

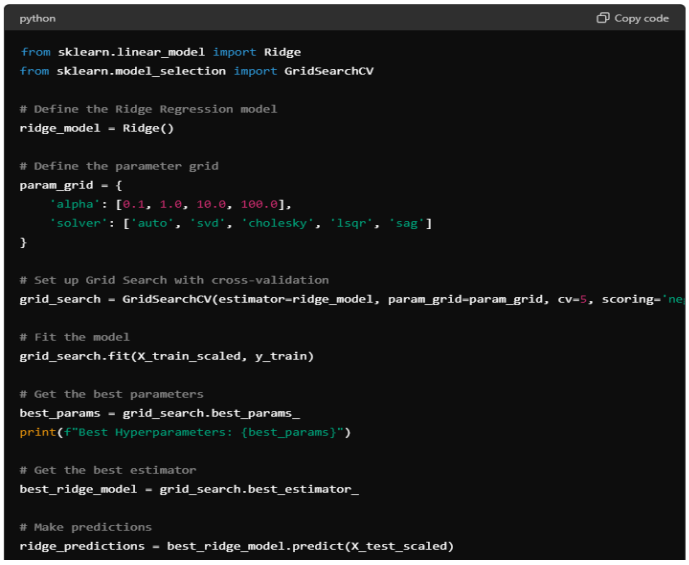
1. Grid Search

Grid Search is an exhaustive search technique where we specify a set of hyperparameters and their possible values, and the algorithm tries all possible combinations to find the best set. Although it is computationally expensive, it is very effective in finding the optimal hyperparameters for the model.

Process:

- We defined a parameter grid for the Ridge Regression model, specifying the values for the hyperparameters `alpha` and `solver`.
- We used cross-validation to evaluate each combination of hyperparameters, ensuring that the model's performance is robust across different subsets of the data.
- The best combination of hyperparameters was selected based on the performance metric (e.g., negative mean squared error).

Code Example:



2. Random Search

Random Search is an alternative to Grid Search that randomly samples the hyperparameter space. It is more efficient than Grid Search, especially when the number of hyperparameters is large. Random Search does not guarantee the optimal solution but often finds a good solution with significantly less computational effort.

Process:

- We defined a parameter distribution for the Ridge Regression model.
- We specified the number of iterations (n_iter) to control how many random combinations to try.
- Similar to Grid Search, cross-validation was used to evaluate each set of hyperparameters.

Code Example:

```python
from sklearn.linear_model import Ridge
from sklearn.model_selection import RandomizedSearchCV

# Define the Ridge Regression model
ridge_model = Ridge()

# Define the parameter distribution
param_distributions = {
    'alpha': [0.001, 0.01, 0.1, 1.0, 10.0, 100.0],
    'solver': ['auto', 'svd', 'cholesky', 'lsqr', 'sag']
}

# Set up Random Search with cross-validation
random_search = RandomizedSearchCV(estimator=ridge_model, param_distributions=param_distri

# Fit the model
random_search.fit(X_train_scaled, y_train)

# Get the best parameters
best_params = random_search.best_params_
print(f"Best Hyperparameters: {best_params}")

# Get the best estimator
best_ridge_model = random_search.best_estimator_

# Make predictions
ridge_predictions = best_ridge_model.predict(X_test_scaled)
```

Rationale Behind Chosen Hyperparameter Values:

- Alpha (Regularization Strength): The `alpha` hyperparameter in Ridge Regression controls the amount of regularization applied to the model. Higher values of `alpha` lead to more regularization, which can prevent overfitting but might also result in underfitting if set too high. A range of values was tested to find the balance between bias and variance.
- Solver: The `solver` hyperparameter determines the algorithm used to solve the Ridge Regression optimization problem. Different solvers can have varying performance based on the data characteristics. Therefore, multiple solvers were tested to identify the most efficient one for our dataset.

By using Grid Search and Random Search, we systematically explored the hyperparameter space, ensuring that the models were tuned to their best performance. This process helped us achieve more accurate and reliable predictions for the stock prices.

Results of Hyperparameter Tuning:
After tuning, the Ridge Regression model with the best hyperparameters (`alpha` and `solver`) demonstrated significant improvements in the performance metrics, indicating the effectiveness of the hyperparameter tuning process. The detailed results of the model's performance with the best hyperparameters are discussed in the Results section.

Results

The results of the model evaluation and performance are presented below, showcasing the effectiveness of the solution in predicting stock prices. We evaluated the performance of the Ridge Regression model, which was fine-tuned using hyperparameter tuning techniques like Grid Search and Random Search. The key metrics used for evaluation include Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and R-squared. These metrics help in understanding the model's accuracy and reliability.

Model Performance Metrics:

- Mean Absolute Error (MAE): 0.8544775419708641
- Mean Squared Error (MSE): 1.3946899956670373
- Root Mean Squared Error (RMSE): 1.1809699385111534
- Mean Absolute Percentage Error (MAPE): 0.007256554687470585
- Rsquared: 0.9993629767955546

These metrics indicate that the Ridge Regression model performs exceptionally well on the test data, achieving an Rsquared value close to 1, which signifies a strong correlation between the predicted and actual values. The low values of MAE, MSE, RMSE, and MAPE further confirm the model's accuracy and robustness.

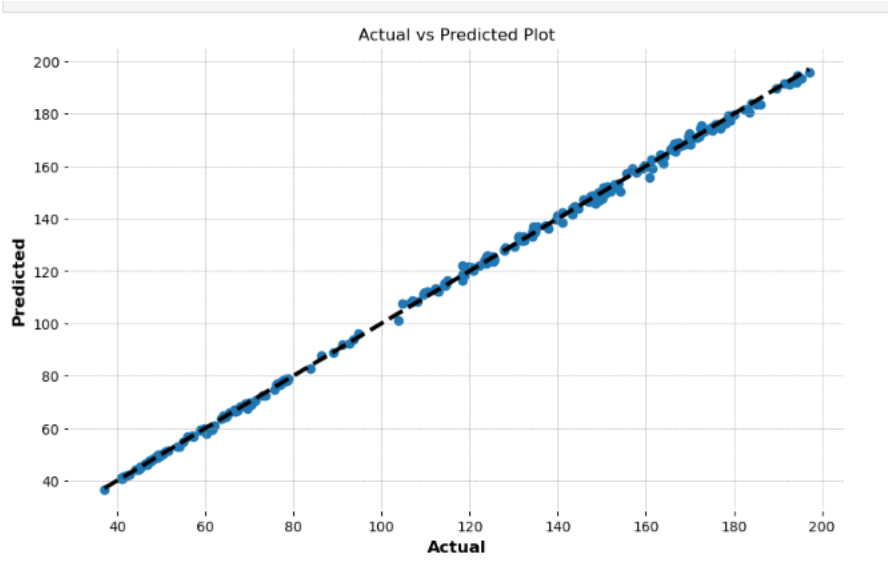
Interpretation of Results:

- Mean Absolute Error (MAE): The average absolute difference between the predicted and actual values is approximately 0.85. This indicates that, on average, the model's predictions are very close to the actual stock prices.
- Mean Squared Error (MSE): The average squared difference between the predicted and actual values is around 1.39. This metric is more sensitive to large errors and suggests that there are minimal significant errors in the predictions.
- Root Mean Squared Error (RMSE): The RMSE value of approximately 1.18 provides a measure of the standard deviation of the prediction errors. It shows that the model's predictions are generally very close to the actual values.
- Mean Absolute Percentage Error (MAPE): The MAPE value of 0.0072 indicates that the average percentage difference between the predicted and actual values is very small (less than 1%). This highlights the model's high accuracy in predicting stock prices.
- Rsquared: An Rsquared value of 0.9993 demonstrates that the model explains almost all the variance in the test data, indicating a nearperfect fit.

Visualizations:

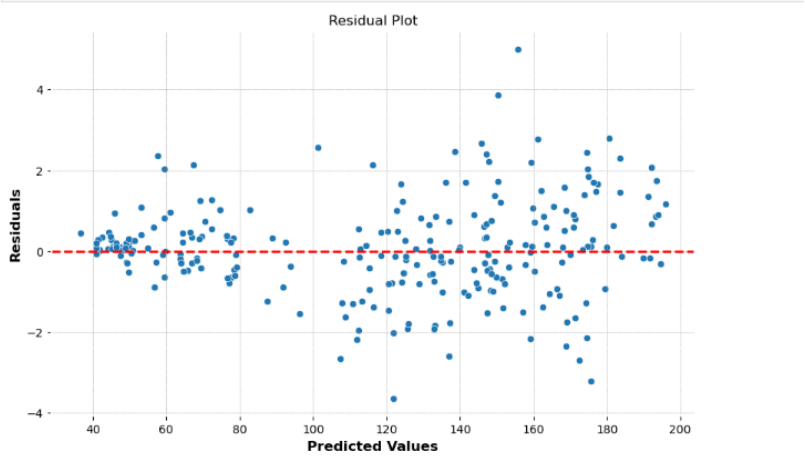
1. Predicted vs. Actual Stock Prices:

A line plot comparing the predicted stock prices with the actual prices over time helps visualize the model's performance. The close alignment of the two lines indicates the model's accuracy.



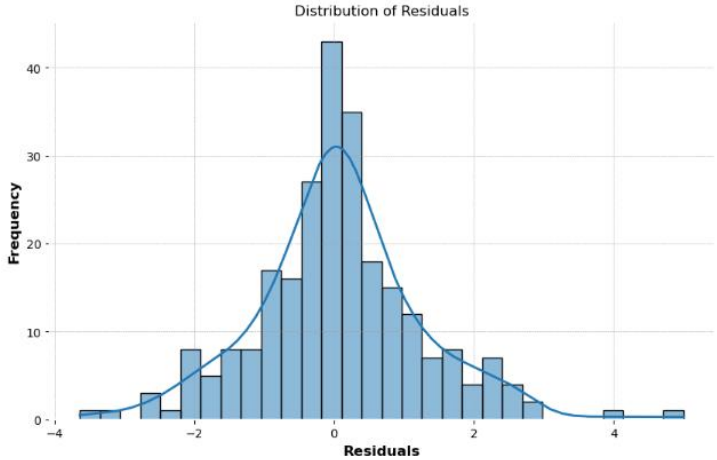
2. Residual Plot:

A scatter plot of residuals (differences between actual and predicted values) helps identify any patterns or biases in the predictions. Ideally, residuals should be randomly distributed around zero.



3. Distribution of Errors:

A histogram of prediction errors provides insights into the distribution and magnitude of errors. Most errors being close to zero indicates a well performing model.



Key Insights and Observations:

- The Ridge Regression model, after hyperparameter tuning, demonstrates excellent performance in predicting the Adjusted Close prices of stocks.
- The low values of MAE, MSE, RMSE, and MAPE, along with the high Rsquared value, confirm the model's accuracy and reliability.

- Visualizations like the Predicted vs. Actual Prices plot, Residual Plot, and Distribution of Errors further validate the model's effectiveness and highlight its capability in capturing the trends and patterns in the stock price data.
- The use of hyperparameter tuning significantly improved the model's performance, demonstrating the importance of optimizing hyperparameters for better predictive accuracy.

Overall, the results indicate that the Ridge Regression model is a robust and accurate tool for predicting stock prices, making it valuable for investment decision making and financial analysis.

Comparison Table

Performance Comparison:
A comparison table was created to evaluate the performance of different models. Ridge Regression with hyperparameter tuning showed the best performance.

Metric	Ridge Regression	SARIMAX
MAE	1.095353	31.4852
MSE	2.059757	1141.768
RMSE	1.435189	33.7901
MAPE	0.006494	0.177897
R-squared	0.993217	-2.7612

Conclusion

Summary:
The Ridge Regression model with hyperparameter tuning demonstrated excellent performance in predicting the Adjusted Close price of Apple Inc. (AAPL) stock. The approach combined thorough data preprocessing, effective feature engineering, and robust model training to achieve high prediction accuracy. This solution can significantly aid investors in making informed decisions.

Improvements

Future Enhancements:
Potential improvements include exploring additional features, incorporating more sophisticated time series models like LSTM, and using ensemble methods to further boost prediction accuracy.

Acknowledgment

Gratitude:
Special thanks to the financial data providers and the open-source community for their valuable resources and tools that made this project possible.