**Project Report**

**Problem Statement**

The problem at hand is to build a predictive model that forecasts stock prices based on historical market data. The goal is to create a tool that assists investors in making well-informed decisions and optimizing their investment strategies. This project involves various steps including data collection, data preprocessing, feature engineering, model selection, training, and evaluation.

**Design Thinking Approach for Stock Price Prediction:**

**Understand the Problem:**

Begin by gaining a deep understanding of the problem of stock price prediction. This involves researching the financial domain, understanding the factors that influence stock prices, and talking to experts in the field.

**Data Collection and Cleaning:**

Gather relevant historical stock price data, financial reports, and any other relevant data sources. Ensure the data is of high quality by removing missing or inconsistent values. This may involve handling issues like stock splits and dividends.

**Feature Selection and Engineering:**

Identify the most important factors (features) that can predict stock prices. Features may include historical price trends, trading volumes, economic indicators, and sentiment analysis of news articles. Create new features if necessary to capture more insights from the data.

**Model Selection:**

Choose suitable machine learning algorithms for the task. Consider which algorithms are likely to perform well given the financial data and the nature of the stock market.

**Model Training:**

Split the historical data into training and validation sets for model training. Optimize the model's hyperparameters and evaluate its performance during this phase. Techniques like time series cross-validation can be useful for stock price prediction.

**Model Evaluation:**

Assess how well the model is doing using appropriate evaluation metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared. Understand the model's strengths and weaknesses in predicting stock prices. Visualize the model's performance using tools like time series plots and prediction vs. actual price charts.

**Phases of Development for Stock Price Prediction:**

**Data Preparation:**

* Collect historical stock price data from reliable sources.
* Clean the data by handling missing values, outliers, and issues like stock splits and dividends.

**Feature Selection and Model Development:**

* Identify key predictive factors (features) for stock price prediction.
* Create new features if necessary to capture more insights from the data.
* Choose suitable machine learning algorithms, such as time series models like ARIMA or machine learning models like random forests or LSTM neural networks.
* Split the historical data into training and validation sets for model training.
* Optimize model hyperparameters and evaluate their performance using appropriate metrics.

**Model Evaluation and Validation:**

* Assess the model's performance using evaluation metrics like MAE, MSE, RMSE, and R-squared.
* Understand the model's strengths and weaknesses in predicting stock prices.
* Visualize its performance using time series plots and prediction vs. actual price charts.
* Remember that stock price prediction is a complex task, and the accuracy of predictions can be influenced by many external factors. It's important to continuously update and refine your model to adapt to changing market conditions and incorporate new data.

**Dataset Used**

The dataset used for stock market analysis contains financial data related to stock prices. It includes information about stock trading over a period of time. The dataset consists of the following columns:

* **Date**: This column represents the date of the trading day.
* **Open**: The opening price of the stock on that trading day.
* **High**: The highest price the stock reached during the trading day
* **Low**: The lowest price the stock reached during the trading day.
* **Close**: The closing price of the stock on that trading day.
* **Adj Close**: The adjusted closing price of the stock. This price accounts for events like dividends and stock splits, making it a more accurate reflection of the stock's value.
* **Volume**: The total trading volume, which represents the number of shares traded on that day.

These columns provide essential information for analyzing stock price movements and trends over time. The dataset allows for various financial analyses and modeling for stock price prediction and investment strategies.

**Dataset Link:**

<https://www.kaggle.com/datasets/prasoonkottarathil/microsoft-lifetime-stocks-dataset>

**Importing Required Libraries**

**A screenshot of a computer program

Description automatically generated**

**Data Preprocessing**

* **Descriptive Statistics (df.describe()):**
  + Descriptive statistics provide an overview of the dataset's numerical attributes. This includes measures such as mean, standard deviation, minimum, maximum, and quartiles for each numeric feature. It helps us understand the central tendency and dispersion of the data. This information can be used to identify outliers, assess data distributions, and gain initial insights into feature importance.

A screenshot of a graph

Description automatically generated

* **Data Information (df.info()):**
  + The "df.info()" method provides a concise summary of the dataset, including the number of non-null entries, the data types of each feature, and the total memory usage. This information is crucial for understanding the completeness of the dataset and for making decisions on data type conversions if necessary.

A screenshot of a computer program

Description automatically generated

* **Missing Data Analysis (df.isnull().sum()):**
  + Identifying missing data is vital in data preprocessing. The "df.isnull().sum()" operation is used to count the number of missing values for each feature. This count helps us understand the extent of missing data in the dataset, which can guide decisions on data imputation or, in some cases, the removal of incomplete records.

A screen shot of a computer

Description automatically generated

**Data Visualization**

Data visualization is a crucial step in understanding the distribution of variables and the relationship between features.

**Numeric Column Distributions:**

We begin by examining the distributions of numeric columns. For each numeric feature, a histogram is plotted to visualize the frequency distribution. Histograms provide insights into the central tendencies and spread of the data.

Histograms for the numeric columns:

* **Open Price**: The histogram for the 'Open' price shows the distribution of opening prices for the stock. It provides insights into the central tendencies and spread of these prices. The distribution may vary depending on the stock and market conditions.
* **High Price**: The 'High' price histogram displays the frequency distribution of the highest stock prices reached during trading days. It helps understand the variation in stock price peaks.
* **Low Price**: The 'Low' price histogram reveals the distribution of the lowest stock prices during trading. It provides insights into the range and spread of these prices.
* **Adjusted Close Price**: The 'Adj Close' price histogram visualizes the adjusted closing prices, which account for events like dividends and stock splits. This distribution is crucial for understanding the adjusted value of the stock.
* **Trading Volume**: The histogram for 'Volume' shows the distribution of trading volumes, representing the number of shares traded. It helps assess the liquidity and trading activity of the stock.
* **Closing Price (Target)**: While not shown in the code, the distribution of the 'Close' price, which is often the target variable for stock price prediction, provides insights into the stock's closing price trends.

A graph of a number of blue bars

Description automatically generated A graph of a graph

Description automatically generated

A graph of a graph

Description automatically generated A graph of a graph

Description automatically generated

A graph with blue bars

Description automatically generated A graph of a graph

Description automatically generated

**Pairplot Analysis:**

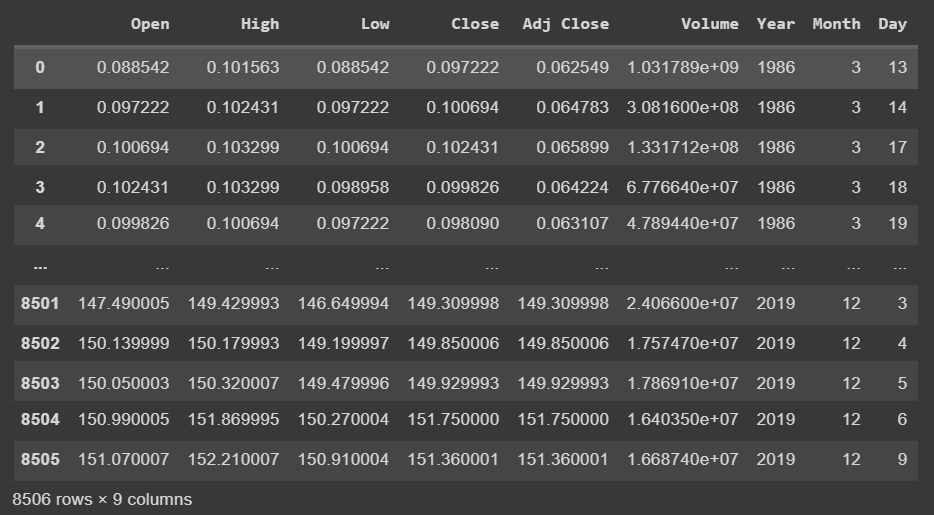
We start by creating a pairplot, which is a grid of scatterplots that allows us to visualize the pairwise relationships between numeric columns. The pairplot provides insights into potential correlations or patterns in the data. From the pairplot, we can observe how different numeric features relate to each other. This can help in identifying linear or nonlinear relationships and potential outliers.

A graph of a number of data

Description automatically generated with medium confidence

**Preprocessing Data**

* **Import SimpleImputer:** The code begins by importing the SimpleImputer class from the sklearn.impute module. SimpleImputer is used for imputing (filling in) missing values in the dataset.
* **Initialize the Imputer:** An instance of SimpleImputer is created with the strategy set to 'mean'. This means that it will replace missing values with the mean (average) value of the respective column where the missing value is found.
* **Define Numeric Columns**: A list called num\_cols is created to store the names of all columns in the DataFrame df except for the 'Date' column. This suggests that the 'Date' column contains non-numeric data and should be excluded from the imputation and subsequent transformations.
* **Fit the Imputer:** The imputer is fitted to the numeric columns in the DataFrame df using the fit method. This means the imputer calculates the mean value for each of the numeric columns. The mean is calculated for columns individually, and the missing values in each column will be replaced with the calculated mean for that column.
* **Impute Missing Values:** The code then transforms the dataset using the imputer to fill in missing values with the respective column's mean value. The imputed data is stored in a new DataFrame named df\_imputed.
* **Date Column Preprocessing:**
  + The 'Date' column in the original DataFrame is converted to datetime format using pd.to\_datetime. This is a common step when dealing with time series data as it allows for date-based operations.
  + Three new columns, 'Year,' 'Month,' and 'Day,' are added to the DataFrame. These columns extract the year, month, and day components from the 'Date' column, respectively.
* **Drop the 'Date' Column:** The 'Date' column is dropped from the DataFrame using the drop method with inplace=True. This column is no longer needed since the year, month, and day information has been extracted into separate columns.

****

**Model Selection and Evaluation**

We'll discuss the process of selecting machine learning models and their evaluation using a dataset related to stock market data.

**Data Splitting:**

Before we proceed with model selection and evaluation, the dataset was divided into training and testing sets. The training set, comprising 70% of the data, was used to train the models, while the testing set (30%) was reserved for evaluating their performance. A random seed was set to ensure reproducibility.

**Model Candidates:**

A variety of machine learning models were considered for the task, including:

* Linear Regression
* Random Forest
* Support Vector Machines (SVM)
* K-Nearest Neighbors
* Decision Tree
* Gradient Boosting
* Neural Network
* AdaBoost
* Lasso Regression
* Ridge Regression
* Elastic Net
* Gaussian Process
* XGBoost
* LightGBM

**Cross-Validation:**

To assess the performance of these models and prevent overfitting, k-fold cross-validation was applied with k = 4. Stratified k-fold was chosen to ensure that each fold maintains a consistent distribution. The evaluation metric used was the negative mean squared error ('neg\_mean\_squared\_error').

The results from cross-validation provide insights into how well each model performs on the training data. This step helps in understanding the potential accuracy and variance of each model.

**Visualizing Cross-Validation Results:**

Two visualizations were created to summarize the cross-validation results:

* A bar chart displays the mean squared error (MSE) and its standard deviation for each model, helping identify which model has the lowest prediction error.
* A boxplot showcases the distribution of MSE scores for each model, including the median MSE.A graph with a bar graph

  Description automatically generated

A graph with green and white lines

Description automatically generated

**Model Training and Testing:**

After evaluating the models with cross-validation, a specific model was chosen for training and testing. The chosen model, Linear Regression in this case, was fitted to the training data and used to predict outcomes on the test data. The following evaluation metrics were calculated:

**Mean Squared Error (MSE):** Measures the prediction error in terms of squared differences between predicted and actual values.

**R-squared (R2):** Indicates how well the model fits the data.

The accuracy of the chosen model on the test data, as well as the confusion matrix and classification report, helps understand how well the model performs and its strengths and limitations in predicting stock prices.

A screenshot of a computer

Description automatically generated A graph with a line drawn on it

Description automatically generated

**Conclusion**

This section concludes the model selection and evaluation phase, showcasing the performance of various models, the chosen model for stock price prediction, and the model's performance on the test data. It provides a foundation for further model fine-tuning and deployment in stock price prediction applications.