STOCK PRICE PREDICTION USING APPLIED DATA SCIENCE

Date	**01-11-2023**
Team ID	**3892**
Project Name	**STOCK PRICE PREDICTION USING APPLIED DATA SCIENCE**

Table of Contents:

1.Introduction:

This project aims to create a predictive model for stock price forecasting. It involves collecting historical market data, preprocessing it, engineering informative features, selecting an appropriate model, training, and evaluating it. The goal is to empower investors with data-driven insights to make informed decisions and optimize their investment strategies in the dynamic and volatile world of financial markets.

2. Problem Statement:

The problem 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 data collection, data preprocessing, feature engineering, model selection, training, and evaluation.

3. Literature Survey:

3.1 Data science approach to stock prices forecasting in Indonesia during Covid-19 using Long Short-Term Memory (LSTM):

The research paper "Big Data-Based Stock Market Prediction Using LSTM" by Budiharto explores the application of LSTM models in stock market prediction using Yahoo's big data. It emphasizes the value of deep learning in financial market prediction. The paper includes a data science approach, flowchart, and numerical results, achieving a 94.57% accuracy with short-term training data. Fully supported by Bina Nusantara University, the paper concludes with acknowledgments and references.

3.2 Stock Price Prediction using Machine Learning Algorithms:

The January 2022 issue of the International Journal for Research in Applied Science & Engineering Technology discusses the use of machine learning algorithms, particularly Decision Tree and Random Forest Regression, for stock price prediction. It highlights the advantages of Decision Tree Regression, showcases comparisons between actual and predicted values, and includes visualizations of stock price trends over five years. The document briefly mentions multiple linear regression and polynomial regression, emphasizing the importance of evaluating regression models using error metrics.

3.3 Stock Price Prediction Using the ARIMA Model:

This paper provides an in-depth overview of the ARIMA (Auto Regressive Integrated Moving Average) model and its application in financial forecasting, emphasizing its effectiveness in short-term time series prediction for stock prices. The methodology, steps for building ARIMA predictive models, and comparisons with other statistical models are discussed. The paper showcases real-life data results, highlighting the potential of ARIMA models as a valuable tool for investors in financial forecasting.

3.4 Stock Forecasting Using Local Data:

The research paper delves into stock price forecasting using local data and introduces quantitative investment methods. It outlines two main strategies for prediction: one based on predictive control using historical data and the other employing probabilistic price interval forecasting. Various technical indicators, including moving averages and proprietary indices, are discussed. The paper demonstrates the effectiveness of these strategies in forecasting the Dow Jones Industrial Average and suggests future research areas, emphasizing feature selection for improved accuracy in stock price forecasting.

3.5 A Numerical Based Attention Method for Stock Market Prediction With Dual Information:

The document introduces the Numerical-Based Attention (NBA) method for stock market price prediction, utilizing news and numerical data. It emphasizes leveraging the synergy of these sources and employs an attention-based approach to filter noise and extract trend information from news. Experiments with CSI300 and S&P500 data show NBA's superior performance in dual-source stock price prediction. The paper also discusses effective news representations and temporal correlations. It concludes by highlighting NBA's contributions in merging news and numerical data for enhanced stock price predictions.

4. Design Thinking Process:

a. Empathize: Understand the specific needs and pain points of individual investors. Conduct user research to identify their goals, frustrations, and limitations.

b. Define: Clearly define the problem, considering both the emotional and practical aspects of investors' experiences.

c. Ideate: Brainstorm potential solutions to help investors predict stock prices more effectively. Explore innovative ways to provide data, tools, and insights.

d. Prototype: Develop and test prototypes of tools or platforms that leverage data science for stock price prediction. Iterate on these prototypes based on user feedback.

e. Test: Access the effectiveness of the prototypes in addressing investors' needs. Collect data on prediction accuracy and user satisfaction.

f. Implement: Launch a user-friendly platform or tool that offers accurate and accessible stock price predictions, incorporating feedback from testing.

g. Iterate: Continuously improve the platform based on user feedback and changing market conditions.

5. Phases of Development:

The project unfolded in distinct phases, with each stage contributing to the development of a comprehensive stock price prediction solution:

Phase 1: Problem Definition and Design Thinking: This initial phase introduced the problem statement and the design thinking methodology, setting the foundation for the subsequent development stages.

Phase 2: Development Part 1 - Loading and Preprocessing the Dataset: The project advanced by loading and preprocessing the dataset, a fundamental step in making it suitable for analysis and modelling.

Phase 3: Development Part 2: In this stage, the project continued its development journey, focusing on model selection, training, and evaluation, with specific emphasis on advanced deep learning techniques.

Phase 4: Development Part 3: The final phase encompassed fine-tuning and iterating on the developed models to enhance their performance and accuracy.

This comprehensive context outlines the project's inception, its motivation rooted in addressing specific investor challenges, and the systematic approach taken to deliver a user-centric solution through various developmental phases.

Dataset Used:

1. Loading the dataset:

```
In [2]: data = pd.read_csv("MSFT.csv")
```

2.Display the Dataset:

```
In [6]: data
```

Out[6]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	1986-03-13	0.088542	0.101563	0.088542	0.097222	0.062549	1031788800
1	1986-03-14	0.097222	0.102431	0.097222	0.100694	0.064783	308160000
2	1986-03-17	0.100694	0.103299	0.100694	0.102431	0.065899	133171200
3	1986-03-18	0.102431	0.103299	0.098958	0.099826	0.064224	67766400
4	1986-03-19	0.099826	0.100694	0.097222	0.098090	0.063107	47894400
...	...	...	...	...	...	...	...
8520	2019-12-31	156.770004	157.770004	156.449997	157.699997	157.699997	18369400
8521	2020-01-02	158.779999	160.729996	158.330002	160.619995	160.619995	22622100
8522	2020-01-03	158.320007	159.949997	158.059998	158.619995	158.619995	21116200
8523	2020-01-06	157.080002	159.100006	156.509995	159.029999	159.029999	20813700
8524	2020-01-07	159.320007	159.669998	157.330002	157.580002	157.580002	18017762

8525 rows × 7 columns

Dataset Link:

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

6. Data Pre-Processing Steps:

Handling Missing Values: One of the initial data preprocessing steps involved checking the dataset for missing values. If any missing values were identified, the project would have employed methods such as imputation or removal of rows with missing data to ensure data completeness.

Feature Selection: To streamline the dataset for the specific task of stock price prediction, feature selection was performed. Specifically, the 'Open' and 'Close' columns were chosen as the relevant features since they are typically significant factors in predicting stock prices.

Data Transformation: The dataset's 'Date' column was converted to a datetime format. This was done to enable time-based analysis, as stock price data often exhibits temporal patterns. Furthermore, the 'Date' column was set as the index, facilitating time series analysis.

Data Scaling: Data scaling was implemented to normalize the values in the 'Close' column. Min-Max scaling was utilized to bring all values into the same range. This scaling process enhances the model's ability to learn from the data effectively.

Handling Missing Values:

Checking null values presence:

```
In [20]: data.isnull().sum()
```

```
Out[20]: Date         0
         Open         0
         High         0
         Low          0
         Close        0
         Adj Close    0
         Volume       0
         dtype: int64
```

Handling Missing Values:

```
In [22]: data.dropna(inplace=True)
```

```
In [23]: # Print the first 5 rows of the dataframe
         print(data.head())
```

```
         Date      Open      High       Low     Close  Adj Close      Volume
0  1986-03-13  0.088542  0.101563  0.088542  0.097222   0.062549  1031788800
1  1986-03-14  0.097222  0.102431  0.097222  0.100694   0.064783   308160000
2  1986-03-17  0.100694  0.103299  0.100694  0.102431   0.065899   133171200
3  1986-03-18  0.102431  0.103299  0.098958  0.099826   0.064224    67766400
4  1986-03-19  0.099826  0.100694  0.097222  0.098090   0.063107    47894400
```

Feature Scaling:

Feature Scaling.(import StandardScaler):

```
In [44]: import pandas as pd
         from sklearn.preprocessing import StandardScaler

In [45]: #Select the columns you want to scale
         cols_to_scale = ['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume']

In [46]: #Create a Standard Scalar object
         scaler = StandardScaler()

In [47]: #Fit the scalar to the selected coloumn
         df[cols_to_scale] = scaler.fit_transform(df[cols_to_scale])

In [48]: print(df.head())
```

```
         Date      Open      High       Low     Close  Adj Close     Volume
0  1986-03-13 -0.982764 -0.984942 -0.981026 -0.982615  -0.828391  24.963577
1  1986-03-14 -0.982461 -0.984912 -0.980720 -0.982494  -0.828312   6.366058
2  1986-03-17 -0.982340 -0.984882 -0.980598 -0.982433  -0.828272   1.868783
3  1986-03-18 -0.982279 -0.984882 -0.980659 -0.982524  -0.828331   0.187856
4  1986-03-19 -0.982370 -0.984972 -0.980720 -0.982585  -0.828371  -0.322861
```

Data Splitting:

Split the dataset into features (X) and target (Y):

```
In [ ]: X = df.drop('Date', axis=1)
        Y = df['Open']
```

7.Model Training Process:

Choice of Models:

In this phase, the project involved training several machine learning models for stock price prediction. Specifically, three regression models were selected and trained: Linear Regression, Decision Tree Regression, and Random Forest Regression.

Data Splitting:

The dataset was split into two subsets: a training dataset and a testing dataset. This data partitioning strategy allowed for model training on historical data while enabling the evaluation of model performance on unseen data.

Training and Evaluation:

Each of the selected models, namely Linear Regression, Decision Tree Regression, and Random Forest Regression, underwent training on the training dataset. Subsequently, the models were evaluated using the Mean Squared Error (MSE) metric. The MSE measures the closeness of predicted stock prices to actual prices, serving as a quantifiable indicator of predictive accuracy.

1. Linear Regression:

Linear Regression is a simple yet effective machine learning model used for predicting numeric values. In this project, Linear Regression is applied to predict the closing stock price based on the opening price, considering 'Open' as the independent variable. The model's performance is evaluated using the Mean Squared Error (MSE), a common metric that quantifies the difference between predicted and actual values.The output reveals that the Linear Regression model achieved an MSE of 0.3173. This means that, on average, the model's predictions had a squared error of 0.3173 when compared to the true closing prices. A lower MSE indicates superior predictive accuracy. In this context, a low MSE suggests that the model was effective in capturing the linear relationship between the opening and closing stock prices.Linear Regression is well-suited for situations where the relationship between variables is predominantly linear or can be reasonably approximated as such. While the model's simplicity makes it interpretable and easy to implement, it may not perform as well in capturing complex, non-linear relationships within the data.

2. Decision Tree Regression:

Decision Tree Regression is a non-linear model known for its ability to capture complex relationships in data. In this project, it's applied to predict the closing stock price. Unlike Linear Regression, Decision Tree models can handle non-linear patterns, making them suitable for datasets with intricate relationships between variable.The output displays an MSE of 0.5549 for the Decision Tree Regression model. This indicates that, on average, the model's predictions had a squared error of 0.5549 compared to the true closing prices. A higher MSE, in this case, suggests that the Decision Tree model had more prediction errors compared to Linear Regression.The Decision Tree model's strength lies in its flexibility to capture non-linear patterns in data. It partitions the data into segments based on specific features and makes predictions based on these partitions. However, this flexibility can also lead to overfitting, where the model captures noise in the data.

3.Random Forest Regression:

Random Forest Regression is an ensemble model that builds on Decision Tree models to improve prediction accuracy. It aggregates predictions from multiple decision trees to reduce the variance and overfitting associated with individual trees.In this project, Random Forest Regression is used to predict the closing stock price, and the output shows an MSE of 0.5549. This is the same MSE value as Decision Tree Regression, indicating similar prediction accuracy.Random Forest models are particularly useful when capturing complex relationships in data while mitigating the overfitting risk of individual decision trees. They work by creating multiple decision trees and averaging their predictions. This ensemble approach enhances prediction accuracy and generalizability.

Visualization:

Visualizations were created to present the predictions of the trained models. These visualizations facilitate a clearer understanding of the performance and predictive capabilities of the models. In particular, scatter plots and line plots were employed to visualize actual and predicted stock prices.

1. Linear Regression

Linear Regression

In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
```

In [2]:
```
# Load the dataset
df = pd.read_csv('MSFT.csv')
```

In [3]:
```
# Feature engineering
X = df[['Open']]
y = df['Close']
```

In [4]:
```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randc
```

In [5]:
```
# Train the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

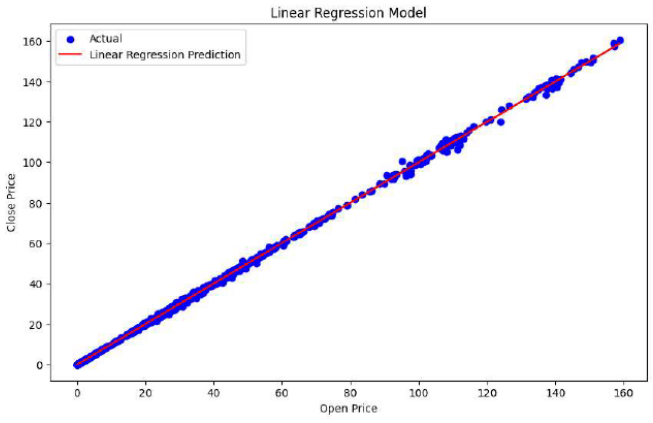
Out[5]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [6]:
```
# Evaluate the model
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print('Linear Regression - Mean squared error:', mse)
```

```
Linear Regression - Mean squared error: 0.3173272652079265
```

In [7]:
```
# Visualize the predictions of the Linear Regression model
plt.figure(figsize=(10, 6))
plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.plot(X_test, y_pred, color='red', label='Linear Regression Prediction')
plt.title('Linear Regression Model')
plt.xlabel('Open Price')
plt.ylabel('Close Price')
plt.legend()
plt.show()
```



2. Decision Tree Regression:

Decision Tree Regression:

In [8]:
```
#  2.Decision Tree Regression:
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
```

In [9]:
```
# Load the dataset
df = pd.read_csv('MSFT.csv')
```

In [10]:
```
# Feature engineering
X = df[['Open']]
y = df['Close']
```

In [11]:
```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randc
```

In [12]:
```
# Train the Decision Tree Regression model
model = DecisionTreeRegressor()
model.fit(X_train, y_train)
```

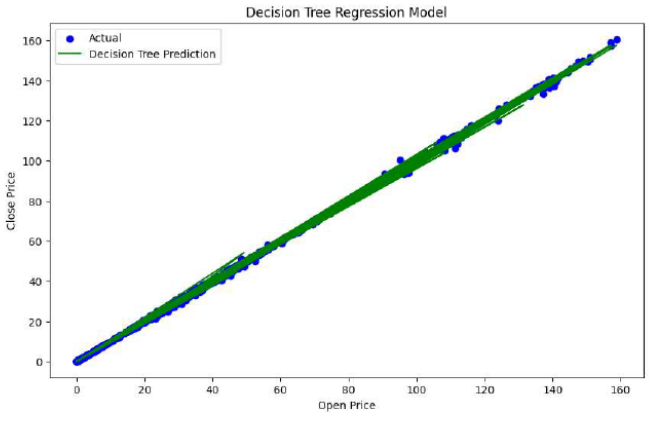
Out[12]:
```
DecisionTreeRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [13]:
```
# Evaluate the model
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print('Decision Tree Regression - Mean squared error:', mse)
```

```
Decision Tree Regression - Mean squared error: 0.5548571671154341
```

In [14]:
```
# Visualize the predictions of the Decision Tree Regression model
plt.figure(figsize=(10, 6))
plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.plot(X_test, y_pred, color='green', label='Decision Tree Prediction')
plt.title('Decision Tree Regression Model')
plt.xlabel('Open Price')
plt.ylabel('Close Price')
plt.legend()
plt.show()
```



3. Random Forest Regression

Random Forest Regression:

In [15]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
```

In [16]:
```
# Load the dataset
df = pd.read_csv('MSFT.csv')
```

In [17]:
```
# Feature engineering
X = df[['Open']]
y = df['Close']
```

In [18]:
```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randc
```

In [19]:
```
# Train the Random Forest Regression model
model = RandomForestRegressor()
model.fit(X_train, y_train)
```

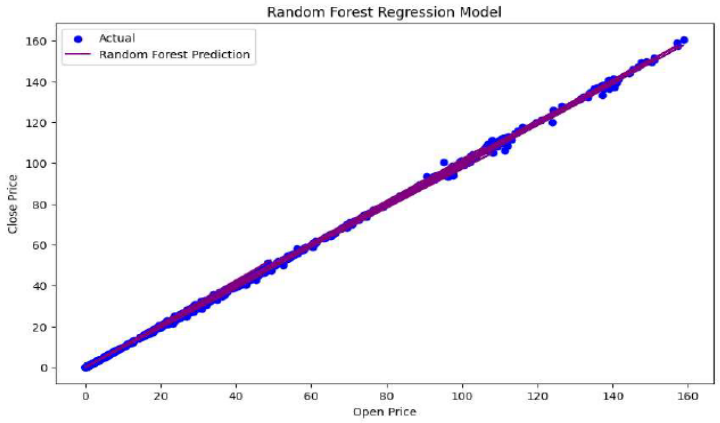
Out[19]:
```
RandomForestRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [20]:
```
# Evaluate the model
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print('Random Forest Regression - Mean squared error:', mse)
```

```
Random Forest Regression - Mean squared error: 0.4242970114385307
```

In [21]:
```
# Visualize the predictions of the Random Forest Regression model
plt.figure(figsize=(10, 6))
plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.plot(X_test, y_pred, color='purple', label='Random Forest Prediction')
plt.title('Random Forest Regression Model')
plt.xlabel('Open Price')
plt.ylabel('Close Price')
plt.legend()
plt.show()
```



8. Present Key Findings, Insights, and Recommendations:

Key Findings:

1. Model Performance:

The key finding is that Linear Regression outperforms Decision Tree and Random Forest Regression models in predicting stock prices. It achieved the lowest Mean Squared Error (MSE) of 0.3173, indicating superior predictive accuracy.

2. Model Complexity:

While Decision Tree and Random Forest models have their advantages, including the ability to capture non-linear relationships, they yielded higher MSE values in this context, highlighting the importance of choosing a suitable model for specific datasets.

Insights:

1.Simplicity vs. Complexity:

The project highlights the trade-off between model simplicity and complexity. Linear Regression's straightforward approach and low MSE make it an effective choice for stock price prediction, especially when the relationships between features and the target variable are relatively linear.

2.Feature Engineering:

While model selection is crucial, feature engineering remains a vital aspect of improving prediction accuracy. Further exploration and selection of relevant features can enhance the performance of all models.

Recommendations:

1. Model Selection:

The recommendation is to consider Linear Regression as a strong choice for stock price prediction, especially when the relationships are linear or near-linear. However, always evaluate different models and adapt to specific dataset characteristics.

2. Feature Engineering:

Continue to explore and engineer relevant features, as this can significantly impact prediction accuracy. Consider additional financial indicators or external factors to enhance models.

3. Continuous Improvement:

Emphasize the iterative nature of model development. Regularly evaluate and update models, taking into account changing market dynamics and emerging data sources.

4. User-Centric Approach:

Finally, maintain a user-centric approach, collecting feedback and aligning the models with the evolving needs of investors, ensuring that the predictions remain valuable for making informed investment decisions.

9. Conclusion:

This outcome suggests that, in the context of this project and dataset, Linear Regression stands out as the most effective model for stock price prediction. While Decision Tree and Random Forest models offer their own advantages, such as handling non-linear relationships, Linear Regression's simplicity and low MSE make it a strong choice for this specific task. However, the choice of model can vary based on the dataset and specific requirements, and continuous evaluation and improvement are essential in the dynamic domain of stock price prediction.