Stock Price Prediction Using Machine Learning

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfilment of the requirements to award the degree of

Bachelor of Technology

In

Computer Science and Engineering

School of Engineering and Sciences

Submitted by

Kalyanram Poonamalli – AP22110010101
Praneetha Sajja – AP22110010135
Balla Pavani Pranathi – AP22110010138
Bhuvan Kethineni – AP22110010173

Under the Guidance of

Dr. Soumyajyoti Biswas

SRM University–AP
Neerukonda, Mangalagiri, Guntur
Andhra Pradesh – 522 240
Nov, 2023

1. Certificate

Date: 28-Nov-2024

This is to certify that the work present in this Project entitled "**Stock Price Prediction Using Machine Learning**" has been carried out by **Kalyanram P, Praneetha S, Pranathi B, Bhuvan K** under my supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology in **School of Engineering and Sciences**.

Supervisor

(Signature)

Dr. Soumyajyoti Biswas

2. Acknowledgement

We want to sincerely thank our mentor Dr. Soumyajyoti Biswas and resources, which were pivotal to finishing this study. First of all, we would like to express our sincere gratitude to Dr. Soumyajyoti Biswas for his tremendous advice, constant support, and knowledgeable insights during this endeavour. His guidance has been a great source of motivation. In addition, We would like to express our gratitude to the writers of the research papers and journals whose perceptive writing served as the foundation for a number of important components of this study. Lastly, We would like to express our appreciation for all of the internet resources that were used in this study and provided crucial support for the findings and conclusions that were made.

3. Table of Contents

1. Certificate

2. Acknowledgments

3. Table of Contents

4. Abstract

5. Introduction

    5.1 Significance

    5.2 Applications

6. Methodology

7. Dataset Description

8. Code Implementation

9. Outcomes

10. Conclusion

11. Future Work

12. References

4. Abstract

The stock market is a dynamic system influenced by various factors, including economic indicators, company performance, and global events, making accurate stock price prediction a challenging task. This project explores the use of machine learning (ML) techniques—**Random Forest (RF)**, **Support Vector Machine (SVM)**, and **Artificial Neural Networks (ANN)**—to predict stock prices for six major companies: **Agilent Technologies Inc. (A)**, **American Airlines Group Inc. (AAL)**, **Advance Auto Parts Inc. (AAP)**, **Apple Inc. (AAPL)**, **AbbVie Inc. (ABBV)**, and **NVIDIA Corporation (NVDA)**. The dataset includes historical stock data with features such as **Open**, **High**, **Low**, **Close**, **Volume**, **Dividends**, and **Stock Splits**, aggregated daily and pre-processed for missing values and scaling. The objective was to predict the **High** stock price using these features. The models were trained and evaluated based on key metrics, including **Mean Absolute Error (MAE)**, **Mean Squared Error (MSE)**, **Root Mean Squared Error (RMSE)**, and **R-squared (R^2)**. After comparison, the **ANN** model outperformed the others, achieving the lowest MAE, MSE, RMSE, and the highest R^2 score, demonstrating its ability to capture non-linear relationships in stock market data. The results show that machine learning models, especially ANN, can effectively predict stock prices and provide valuable insights into stock market behaviour. This study contributes to the growing field of financial forecasting by demonstrating the practical application of machine learning techniques in stock price prediction, offering valuable insights for investment strategies and financial analysis.

5. Introduction

Stock markets play a critical role in the global economy, serving as platforms for companies to raise capital and for investors to build wealth. Accurately predicting stock price movements has always been a complex task due to the volatile and non-linear nature of financial data. Traditional approaches often fall short in handling the intricacies of modern markets, prompting a shift toward leveraging machine learning and deep s.

This project investigates the performance of several machine learning models in predicting stock prices using real-world datasets from prominent companies. By utilizing advanced preprocessing techniques, such as scaling, null value handling, and feature selection, the project ensures data quality and consistency. The models are trained and evaluated using standard regression metrics to identify the most effective method for stock price forecasting. This comparative study not only provides insights into the strengths and limitations of each model but also contributes to the growing body of research in financial analytics.

5.1) Significance

1. **Informed Decision-Making**
   Predicting stock trends empowers individuals and organizations to make data-driven investment decisions, optimizing profits while minimizing risks.
2. **Economic Insights**
   Stock market trends reflect broader economic conditions. Understanding these trends can help economists and policymakers anticipate and respond to economic shifts effectively.
3. **Encourages Financial Literacy**
   Stock analysis promotes awareness and understanding of financial markets, enabling people to better manage their investments and savings.
4. **Advancement in Technology**
   Leveraging machine learning and AI for stock market prediction fosters innovation in predictive analytics and financial modelling.

5. **Risk Management**
   By identifying potential market downturns or anomalies, stakeholders can proactively implement strategies to mitigate financial risks.
6. **Automation in Trading**
   Stock prediction models form the backbone of algorithmic trading systems, automating the process of buying and selling securities with high accuracy and efficiency.

5.2) Applications

1. **Portfolio Management**
   - Helps investors balance risk and return by predicting market trends, ensuring optimal allocation of resources across various financial instruments.
2. **Algorithmic Trading**
   - Machine learning models enable real-time decision-making for automated trading systems, improving speed and accuracy while reducing human errors.
3. **Economic Policy Formulation**
   - Governments and financial institutions use stock market trends as indicators of economic health, aiding in policy-making and fiscal adjustments.
4. **Risk Assessment and Mitigation**
   - Corporations use predictive models to hedge against market volatility, securing their assets and operations against potential financial crises.
5. **Business Planning and Strategy**
   - Companies analyse stock performance to assess their market position, guiding strategic decisions such as mergers, acquisitions, or product launches.
6. **Financial Technology (FinTech)**

- Fuels the development of apps and platforms that provide predictive analytics, empowering individuals with accessible tools for financial management.

7. **Education and Research**
    - A valuable resource in academic and professional settings, stock market prediction enhances learning in fields like data science, finance, and economics.
8. **Retail Investment Tools**
    - Retail investors use predictive insights through platforms like Robinhood or Zerodha to plan trades and investments.
9. **Market Sentiment Analysis**
    - By integrating social and financial data, predictive systems provide insights into market sentiment, influencing stock valuations and trading decisions.
10. **Insurance and Banking Sectors**
    - Predictive models are used to analyse market trends that affect credit, lending rates, and risk assessments for loans or policies.

By bridging the gap between raw data and actionable insights, stock market prediction models hold immense value for stakeholders ranging from individual investors to global financial institutions.

6. Methodology

The methodology for this project is designed to systematically predict stock prices using machine learning techniques. It involves several steps, ranging from data preprocessing to model evaluation. The process ensures that the data is clean, well-structured, and suitable for applying advanced regression models. Below is a step-by-step explanation of the methodology:

1. Data Collection

Historical stock price data for multiple companies was collected in .csv format. The data includes metrics such as Open, High, Low, Close, Volume, Dividends, and Stock Splits.

2. Data Integration and Preprocessing

- The individual datasets for each company were merged into a consolidated dataset.
- Aggregation of features was performed based on the Date column to calculate the average values of stock metrics.
- Missing values were identified and handled using mean imputation. Features like Volume were rounded to integers to maintain data integrity.
- The features were normalized using MinMaxScaler to scale the values between 0 and 1, ensuring uniformity across all features.

3. Exploratory Data Analysis (EDA)

- Correlation analysis was conducted using a heatmap to identify relationships between variables.
- Histograms were generated to visualize the distribution of features such as Open, Close, and Volume.

4. Feature Selection and Engineering

- Variance Threshold was applied to remove features with minimal variance, ensuring that only the most informative features were retained.
- The target variable for prediction was selected as High stock price, representing the highest price a stock reached on a particular day.

5. Data Splitting

The dataset was divided into training and testing sets using TimeSeriesSplit, a cross-validation method specifically designed for time-series data. This ensures that the temporal order of the data is preserved during model training and testing.

6. Model Implementation

Four different models were implemented to predict the High stock price:

- **Random Forest (RF):** An ensemble-based regression model that combines multiple decision trees to improve prediction accuracy.
- **Support Vector Machine (SVM):** A kernel-based regression model that maps data to a higher-dimensional space for better predictions.
- **Artificial Neural Network (ANN):** A dense network of layers designed to capture complex relationships between features.

7. Model Evaluation

The performance of each model was evaluated using the following metrics:

- **Mean Absolute Error (MAE):** Measures the average magnitude of errors.
- **Mean Squared Error (MSE):** Captures the average squared error, penalizing large deviations more.
- **Root Mean Squared Error (RMSE):** Provides a normalized measure of error by taking the square root of MSE.
- **R-squared (R^2):** Represents the proportion of variance in the target variable explained by the model.

7. Dataset Description

The dataset used in this project contains historical stock market data from the following companies:

1. A (Agilent Technologies Inc.)

- **Industry**: Healthcare, Life Sciences & Diagnostics, and Applied Chemical Markets
- **Overview**: Agilent Technologies is a global leader in life sciences, diagnostics, and applied chemical markets. The company provides instruments, software, services, and consumables for laboratories and has a significant presence in the biotechnology and pharmaceutical industries. Agilent's stock is widely traded and is a reflection of the growth and stability in the healthcare and diagnostics sector.

2. AAL (American Airlines Group Inc.)

- **Industry**: Airlines
- **Overview**: American Airlines Group is the parent company of American Airlines, one of the largest airlines in the world. The company provides air travel services across domestic and international routes. Stock performance for AAL is often impacted by factors such as fuel prices, travel demand, and geopolitical events. As a major player in the airline industry, its stock price is influenced by broader economic conditions, including global trade, tourism, and the aviation sector.

3. AAP (Advance Auto Parts Inc.)

- **Industry**: Retail (Automotive Parts)
- **Overview**: Advance Auto Parts is a leading retailer of automotive aftermarket parts, accessories, and maintenance products. The company operates stores across the United States and offers online sales. Its stock price tends to be affected by consumer trends, automotive market dynamics, and the health of the broader retail and consumer goods sectors.

4. AAPL (Apple Inc.)

- **Industry**: Technology

- **Overview**: Apple Inc. is one of the largest and most influential technology companies globally. Known for its innovative products like the iPhone, iPad, and Mac computers, Apple has a massive consumer base. The company's stock price is influenced by product launches, market trends, earnings reports, and broader technology sector developments. Apple's stock is often seen as a bellwether for the tech industry, and its performance can be indicative of investor sentiment regarding tech stocks.

5. ABBV (AbbVie Inc.)

- **Industry**: Pharmaceuticals
- **Overview**: AbbVie is a global biopharmaceutical company formed in 2013 as a spin-off from Abbott Laboratories. It focuses on immunology, oncology, and neuroscience products. AbbVie is known for its blockbuster drug Humira and is a significant player in the global pharmaceutical industry. The company's stock price is impacted by drug development, regulatory approvals, and market competition in the pharmaceutical sector.

6. NVDA (NVIDIA Corporation)

- **Industry**: Semiconductors, Technology
- **Overview**: NVIDIA is a leading American technology company known for its graphics processing units (GPUs) used in gaming, artificial intelligence (AI), and deep learning. It has a dominant position in the semiconductor industry, particularly in gaming hardware and AI applications. NVIDIA's stock price is influenced by technological advancements, gaming trends, AI adoption, and competition in the tech sector.

Impact on Stock Market

These companies represent a diverse set of industries—healthcare, airlines, retail, technology, pharmaceuticals, and semiconductors—each with unique drivers and influences on their stock prices. Together, they provide a broad spectrum of market data that helps in understanding stock price movements and applying machine learning models for prediction. Their stock price dynamics also reflect different economic and industry-specific factors, making them valuable for analysis in terms of both individual stock performance and market trends.

Features:

The dataset includes the following features for each company:

- **Date**: The trading date (used as the index for grouping and aggregation).
- **Open**: The stock price at market opening on a given day.
- **High**: The highest stock price during the trading session.
- **Low**: The lowest stock price during the trading session.
- **Close**: The stock price at market closure on a given day.
- **Volume**: The total number of shares traded during the day.
- **Dividends**: Dividends per share distributed by the company.
- **Stock Splits**: Stock split ratio if applicable for the trading day.

8. Code Implementation

Importing Neccessary Libraries

[1]:
```
# Core Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf

# Machine Learning Preprocessing and Metrics
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import TimeSeriesSplit
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Deep Learning Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout

# Machine Learning Libraries
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor

# Miscellaneous
import warnings
warnings.filterwarnings("ignore")
```

Reading CSV file(s)

[2]:
```
stock_list = []
company_name = ['A', 'AAL', 'AAP', 'AAPL', 'ABBV', 'NVDA']

for i in company_name:
    d = pd.read_csv(i + '.csv')
    stock_list.append(d)

dataset1 = pd.concat(stock_list, axis = 0, ignore_index = True)
```

```
dataset = pd.DataFrame(dataset1).groupby('Date').agg({'Open':'mean', 'High':
↪'mean', 'Low':'mean',
                                                      'Close':'mean', 'Volume':
↪'mean',
                                                     'Dividends':'mean', 'Stock␣
↪Splits':'mean'})
```

Shape of dataset

[3]:
```
print("\nDataset Shape / Size")
print("----------------------")
print("Dataset Shape: ", dataset.shape)
print("         Row : ", len(dataset.index))
print("      Column : ", len(dataset.columns))

dataset.head(5)
```

```
Dataset Shape / Size
----------------------
Dataset Shape:  (10484, 7)
         Row :  10484
      Column :  7
```

[3]:

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
1980-12-12	0.100178	0.100614	0.100178	0.100178	469033600.0	0.0	0.0
1980-12-15	0.095388	0.095388	0.094952	0.094952	175884800.0	0.0	0.0
1980-12-16	0.088418	0.088418	0.087983	0.087983	105728000.0	0.0	0.0
1980-12-17	0.090160	0.090596	0.090160	0.090160	86441600.0	0.0	0.0
1980-12-18	0.092774	0.093210	0.092774	0.092774	73449600.0	0.0	0.0

Dataset description

[4]:
```
dataset.describe()
```

[4]:

	Open	High	Low	Close	Volume
count	10483.000000	10483.000000	10483.000000	10483.000000	1.048300e+04
mean	21.037068	21.321294	20.743693	21.037456	1.548524e+08
std	32.451767	32.870370	32.010217	32.448573	1.628807e+08

```
min        0.038765      0.038765      0.038329      0.038329  0.000000e+00
25%        0.252161      0.258562      0.246706      0.252420  6.005849e+07
50%        8.791608      8.972038      8.601623      8.802916  1.201928e+08
75%       24.594796     24.967835     24.248951     24.645626  1.937670e+08
max      172.327574    173.430638    169.585113    171.665993  4.190480e+09

          Dividends  Stock Splits
count  10484.000000  10484.000000
mean       0.000943      0.000967
std        0.010633      0.033938
min        0.000000      0.000000
25%        0.000000      0.000000
50%        0.000000      0.000000
75%        0.000000      0.000000
max        0.294278      2.000000
```

Dataset Information

[5]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10484 entries, 1980-12-12 to 2022-07-12
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Open          10483 non-null  float64
 1   High          10483 non-null  float64
 2   Low           10483 non-null  float64
 3   Close         10483 non-null  float64
 4   Volume        10483 non-null  float64
 5   Dividends     10484 non-null  float64
 6   Stock Splits  10484 non-null  float64
dtypes: float64(7)
memory usage: 655.2+ KB
```

Correlation between columns

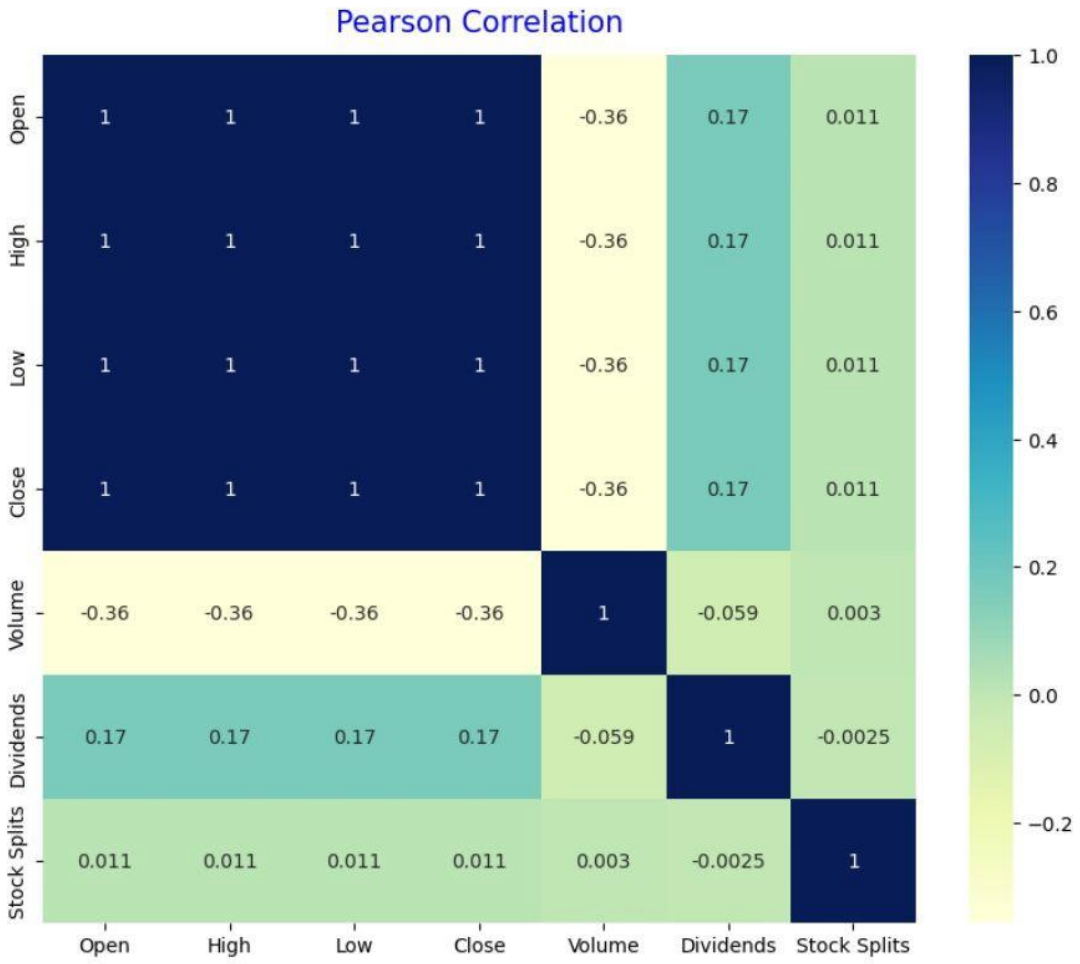
[6]:
```
dataset.corr()
```

[6]:
```
                 Open      High       Low     Close    Volume  Dividends  \
Open         1.000000  0.999947  0.999937  0.999874 -0.356422   0.165783
High         0.999947  1.000000  0.999899  0.999930 -0.355969   0.165603
Low          0.999937  0.999899  1.000000  0.999932 -0.357046   0.165612
Close        0.999874  0.999930  0.999932  1.000000 -0.356531   0.165557
Volume      -0.356422 -0.355969 -0.357046 -0.356531  1.000000  -0.059440
Dividends    0.165783  0.165603  0.165612  0.165557 -0.059440   1.000000
Stock Splits 0.010975  0.011173  0.011048  0.011197  0.002954  -0.002528
```

```
              Stock Splits
Open              0.010975
High              0.011173
Low               0.011048
Close             0.011197
Volume            0.002954
Dividends        -0.002528
Stock Splits      1.000000
```

Data Visualization

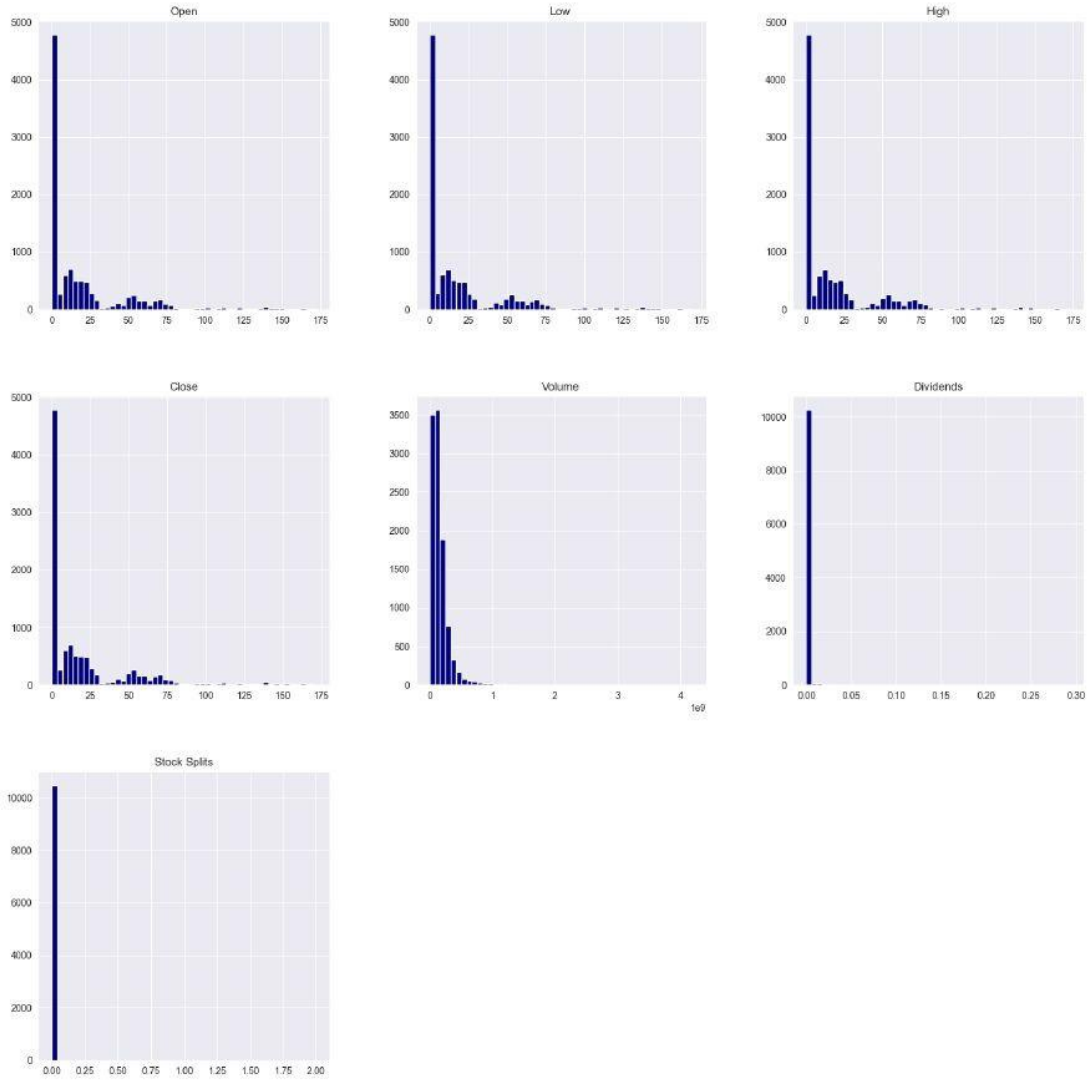
[7]:
```
#1) Heatmap

# Visualize the correlation using pearson correlation
plt.figure(figsize = (10,8))
sns.heatmap(dataset.corr(), annot = True, cmap = 'YlGnBu')
plt.title("Pearson Correlation", fontsize = 15, color = 'b', pad = 12, loc =␣
↪'center')
plt.savefig('pearson_correlation.png')
plt.show()
```



[8]:
```
sns.set_style('dark')
temp = dataset[['Open', 'Low', 'High', 'Close', 'Volume', 'Dividends', 'Stock␣
↪Splits']]
```

[9]:
```
#2) Histogram
print("Features")
print("--------")
temp.hist(bins=50, figsize=(20,20), color='navy');
plt.savefig('histogram.png')  # Save as a PNG file
```

```
Features
--------
```



Select the feautures and target

[10]:
```
# Set Target Variable
target = dataset['High']

# Selecting the Features to be scaled later
featuresToBeScaled = ['Open', 'Low', 'Close', 'Volume']
```

Data Preporcessing

[11]:
```
#Dealing with null values

isNullExist = dataset.isnull().any()
```

```
if('True' in str(isNullExist)):
    nullData = dataset[dataset.isnull().any(axis=1)]
    print('Before data cleaning')
    print('____________________\n')
    print('Null value exist: True')
    print('\nRow(s) consisting of null values')
    print('--------------------------------')
    print(nullData)

    featuresToBeReplaced = ['Open', 'High', 'Low', 'Close', 'Volume',␣
 ↪'Dividends', 'Stock Splits']

    for i in featuresToBeReplaced:
        mean_value = 0
        mean_value = dataset[i].mean()

        if(i != 'Volume'):
            dataset[i].fillna(value=mean_value, inplace=True)
        else:
            dataset[i].fillna(value=int(mean_value), inplace=True)

    print('\n\nAfter data cleaning')
    print('____________________\n')
    nullData = dataset[dataset.isnull().any(axis=1)]
    print(nullData)

else:
    print("No missing / null values in dataset")
```

```
Before data cleaning
--------------------

Null value exist: True

Row(s) consisting of null values
--------------------------------
            Open  High  Low  Close  Volume  Dividends  Stock Splits
Date
2001-09-12   NaN   NaN  NaN    NaN     NaN        0.0           2.0


After data cleaning
-------------------

Empty DataFrame
Columns: [Open, High, Low, Close, Volume, Dividends, Stock Splits]
Index: []
```

Scaling

[12]:
```
scaler = MinMaxScaler()
feature_transform = scaler.fit_transform(dataset[featuresToBeScaled])
feature_transform = pd.DataFrame(columns=featuresToBeScaled,␣
 ↪data=feature_transform,
                                  index=dataset.index)
print("\nDataset after scaling:")
print("-----------------------")
print(feature_transform)

output_var = pd.DataFrame(dataset['High'])
```

```
Dataset after scaling:
-----------------------
                Open       Low     Close    Volume
Date
1980-12-12  0.000356  0.000365  0.000360  0.111928
1980-12-15  0.000329  0.000334  0.000330  0.041972
1980-12-16  0.000288  0.000293  0.000289  0.025231
1980-12-17  0.000298  0.000306  0.000302  0.020628
1980-12-18  0.000313  0.000321  0.000317  0.017528
…                ⋯         ⋯         ⋯         ⋯
2022-07-06  0.737935  0.740848  0.741295  0.006573
2022-07-07  0.740314  0.749115  0.755453  0.006250
2022-07-08  0.745345  0.754403  0.754948  0.005769
2022-07-11  0.746138  0.748220  0.745499  0.005506
2022-07-12  0.743932  0.746815  0.744596  0.008173

[10484 rows x 4 columns]
```

Splitting to Training set and Test set

[13]:
```
timesplit = TimeSeriesSplit(n_splits=10)

for train_index, test_index in timesplit.split(feature_transform):
        X_train, X_test = feature_transform[:len(train_index)],␣
 ↪feature_transform[len(train_index):
                                                    ␣
 ↪(len(train_index)+len(test_index))]
        y_train, y_test = output_var[:len(train_index)].values.ravel(),␣
 ↪output_var[len(train_index):
                                                    ␣
 ↪(len(train_index)+len(test_index))].values.ravel()
```

Implementing the SVM model

[14]:
```
svm_model = SVR(kernel='rbf', C=100, gamma=0.1, epsilon=0.1)
```

Fitting the model

[15]:
```
svm_model.fit(X_train, y_train)
```

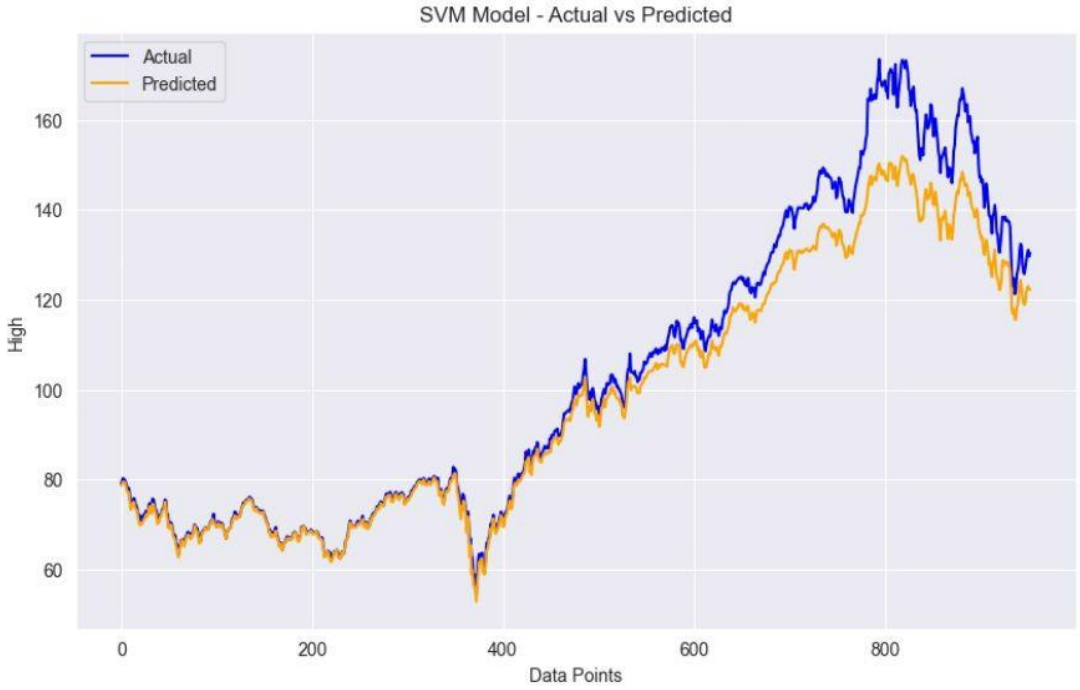
[15]:
```
SVR(C=100, gamma=0.1)
```

Predicting using the SVM model

[16]:
```
y_pred = svm_model.predict(X_test)
```

Plotting the Actual VS Predicted graph for SVM Model

[17]:
```
plt.figure(figsize=(10, 6))
plt.plot(y_test, label='Actual', color='blue')
plt.plot(y_pred, label='Predicted', color='orange')
plt.title('SVM Model - Actual vs Predicted')
plt.xlabel('Data Points')
plt.ylabel('High')
plt.legend()
plt.grid(True)
plt.savefig('SVM.png')
plt.show()
```



Evaluating the SVM model

[18]:
```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

# Print the metrics
print("SVM Model Evaluation Metrics")
print("-----------------------------------")
print(f"Mean Absolute Error (MAE): {mae:.4f}")
print(f"Mean Squared Error (MSE): {mse:.4f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.4f}")
print(f"R-squared Score (R²): {r2:.4f}")
```

```
SVM Model Evaluation Metrics
-----------------------------------
Mean Absolute Error (MAE): 5.0309
Mean Squared Error (MSE): 59.5746
Root Mean Squared Error (RMSE): 7.7185
R-squared Score (R²): 0.9478
```

Implementing the Random Forest Regressor

[19]:
```
rf_model = RandomForestRegressor(n_estimators=100, max_depth=10,␣
↪random_state=42)
```

Fitting the model

[20]:
```
rf_model.fit(X_train, y_train)
```

[20]:
```
RandomForestRegressor(max_depth=10, random_state=42)
```

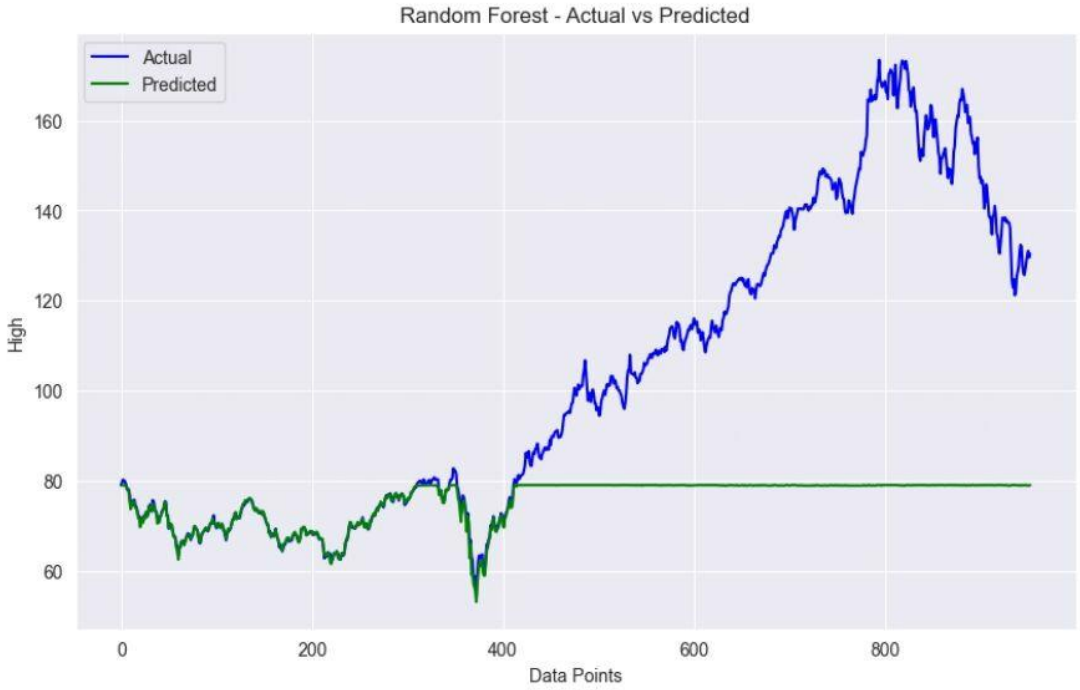
Predicting using Random Forest model

[21]:
```
y_pred = rf_model.predict(X_test)
```

Plotting the Actual vs Predicted for Random Forest model

[22]:
```
plt.figure(figsize=(10, 6))
plt.plot(y_test, label='Actual', color='blue')
plt.plot(y_pred, label='Predicted', color='green')
plt.title('Random Forest - Actual vs Predicted')
plt.xlabel('Data Points')
plt.ylabel('High')
plt.legend()
plt.grid(True)
plt.savefig('RF.png')
```

```
plt.show()
```



Evaluating the Random Forest model

[23]:
```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

# Print the metrics
print("Random Forest Model Evaluation Metrics")
print("-----------------------------------")
print(f"Mean Absolute Error (MAE): {mae:.4f}")
print(f"Mean Squared Error (MSE): {mse:.4f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.4f}")
print(f"R-squared Score (R²): {r2:.4f}")
```

```
Random Forest Model Evaluation Metrics
-----------------------------------
Mean Absolute Error (MAE): 27.5821
Mean Squared Error (MSE): 1686.8755
Root Mean Squared Error (RMSE): 41.0716
R-squared Score (R²): -0.4770
```

Implementing the ANN

[24]:
```
model = Sequential([
    Dense(64, input_dim=X_train.shape[1], activation='relu'),  # Input layer␣
 ↪with 64 neurons
    Dropout(0.2),  # Dropout for regularization
    Dense(32, activation='relu'),  # Hidden layer with 32 neurons
    Dense(1, activation='linear')  # Output layer for regression
])
```

Compiling the model

[25]:
```
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

Training the model

[26]:
```
history = model.fit(X_train, y_train, epochs=100, batch_size=32,␣
 ↪validation_split=0.2, verbose=1)
```

```
Epoch 1/100
239/239             2s 3ms/step -
loss: 63.9958 - mae: 5.2937 - val_loss: 635.0094 - val_mae: 23.0244
Epoch 2/100
239/239             1s 2ms/step -
loss: 7.9652 - mae: 2.1917 - val_loss: 57.9137 - val_mae: 6.4896
Epoch 3/100
239/239             0s 2ms/step -
loss: 1.0489 - mae: 0.7592 - val_loss: 28.6066 - val_mae: 4.5338
Epoch 4/100
239/239             1s 2ms/step -
loss: 0.8527 - mae: 0.5944 - val_loss: 18.5839 - val_mae: 3.7139
Epoch 5/100
239/239             1s 2ms/step -
loss: 0.6849 - mae: 0.5103 - val_loss: 9.6738 - val_mae: 2.6874
Epoch 6/100
239/239             1s 2ms/step -
loss: 0.6307 - mae: 0.4685 - val_loss: 2.7122 - val_mae: 1.3161
Epoch 7/100
239/239             1s 2ms/step -
loss: 0.5677 - mae: 0.4201 - val_loss: 1.7680 - val_mae: 1.0894
Epoch 8/100
239/239             1s 2ms/step -
loss: 0.5962 - mae: 0.4223 - val_loss: 1.1239 - val_mae: 0.9228
Epoch 9/100
239/239             1s 2ms/step -
loss: 0.4918 - mae: 0.3906 - val_loss: 0.5316 - val_mae: 0.5996
Epoch 10/100
239/239             1s 2ms/step -
```

```
loss: 0.1059 - mae: 0.1653 - val_loss: 3.4321 - val_mae: 1.5027
Epoch 91/100
239/239             0s 2ms/step -
loss: 0.0809 - mae: 0.1530 - val_loss: 6.6043 - val_mae: 2.1831
Epoch 92/100
239/239             0s 2ms/step -
loss: 0.0930 - mae: 0.1654 - val_loss: 3.3279 - val_mae: 1.4568
Epoch 93/100
239/239             1s 2ms/step -
loss: 0.0870 - mae: 0.1651 - val_loss: 3.9365 - val_mae: 1.5988
Epoch 94/100
239/239             1s 2ms/step -
loss: 0.1000 - mae: 0.1594 - val_loss: 3.4286 - val_mae: 1.4862
Epoch 95/100
239/239             1s 2ms/step -
loss: 0.0820 - mae: 0.1578 - val_loss: 4.8565 - val_mae: 1.8168
Epoch 96/100
239/239             1s 2ms/step -
loss: 0.0889 - mae: 0.1606 - val_loss: 4.7300 - val_mae: 1.8375
Epoch 97/100
239/239             1s 2ms/step -
loss: 0.0733 - mae: 0.1491 - val_loss: 5.1940 - val_mae: 1.9051
Epoch 98/100
239/239             1s 2ms/step -
loss: 0.0840 - mae: 0.1582 - val_loss: 4.3130 - val_mae: 1.7465
Epoch 99/100
239/239             0s 2ms/step -
loss: 0.0808 - mae: 0.1548 - val_loss: 2.5828 - val_mae: 1.3310
Epoch 100/100
239/239             1s 2ms/step -
loss: 0.0815 - mae: 0.1560 - val_loss: 4.3537 - val_mae: 1.7281
```

Predicting using ANN model

[27]:
```
y_pred = model.predict(X_test)
```

```
30/30             0s 2ms/step
```

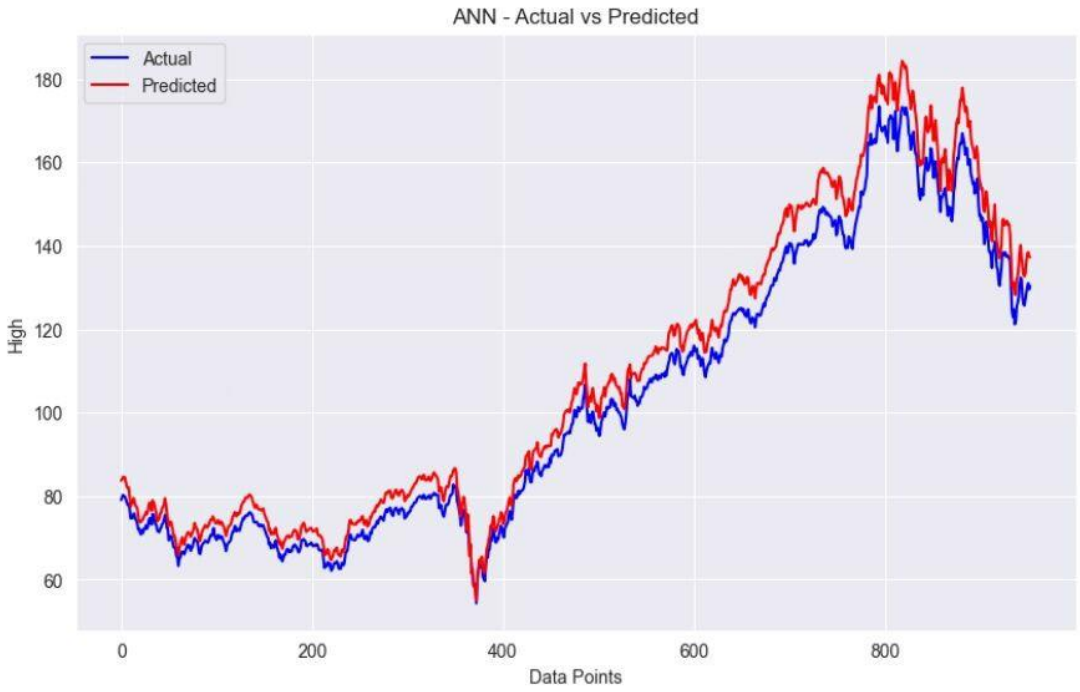
Plotting the Loss during Training graph for ANN

[28]:
```
plt.figure(figsize=(10, 6))
plt.plot(history.history['loss'], label='Training Loss', color='blue')
plt.plot(history.history['val_loss'], label='Validation Loss', color='orange')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss (MSE)')
plt.legend()
plt.savefig('ANN_T_vs_VL.png')
plt.grid(True)
```

```
plt.show()
```

Plotting the Actual vs Predicted graph using ANN

[29]:
```
plt.figure(figsize=(10, 6))
plt.plot(y_test, label='Actual', color='blue')
plt.plot(y_pred, label='Predicted', color='red')
plt.title('ANN - Actual vs Predicted')
plt.xlabel('Data Points')
plt.ylabel('High')
plt.legend()
plt.grid(True)
plt.savefig('ANN.png')
plt.show()
```



Evaluating the ANN model

[30]:
```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

# Print the metrics
print("ANN Model Evaluation Metrics")
print("-----------------------------------")
print(f"Mean Absolute Error (MAE): {mae:.4f}")
print(f"Mean Squared Error (MSE): {mse:.4f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.4f}")
print(f"R-squared Score (R²): {r2:.4f}")
```

```
ANN Model Evaluation Metrics
-----------------------------------
Mean Absolute Error (MAE): 5.5748
Mean Squared Error (MSE): 36.8035
Root Mean Squared Error (RMSE): 6.0666
R-squared Score (R²): 0.9678
```

9. Outcomes

The evaluation of the three models—SVM, Random Forest and ANN—was conducted using four key metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared Score (R^2). The outcomes highlight the strengths and weaknesses of each model:

1. SVM Model:

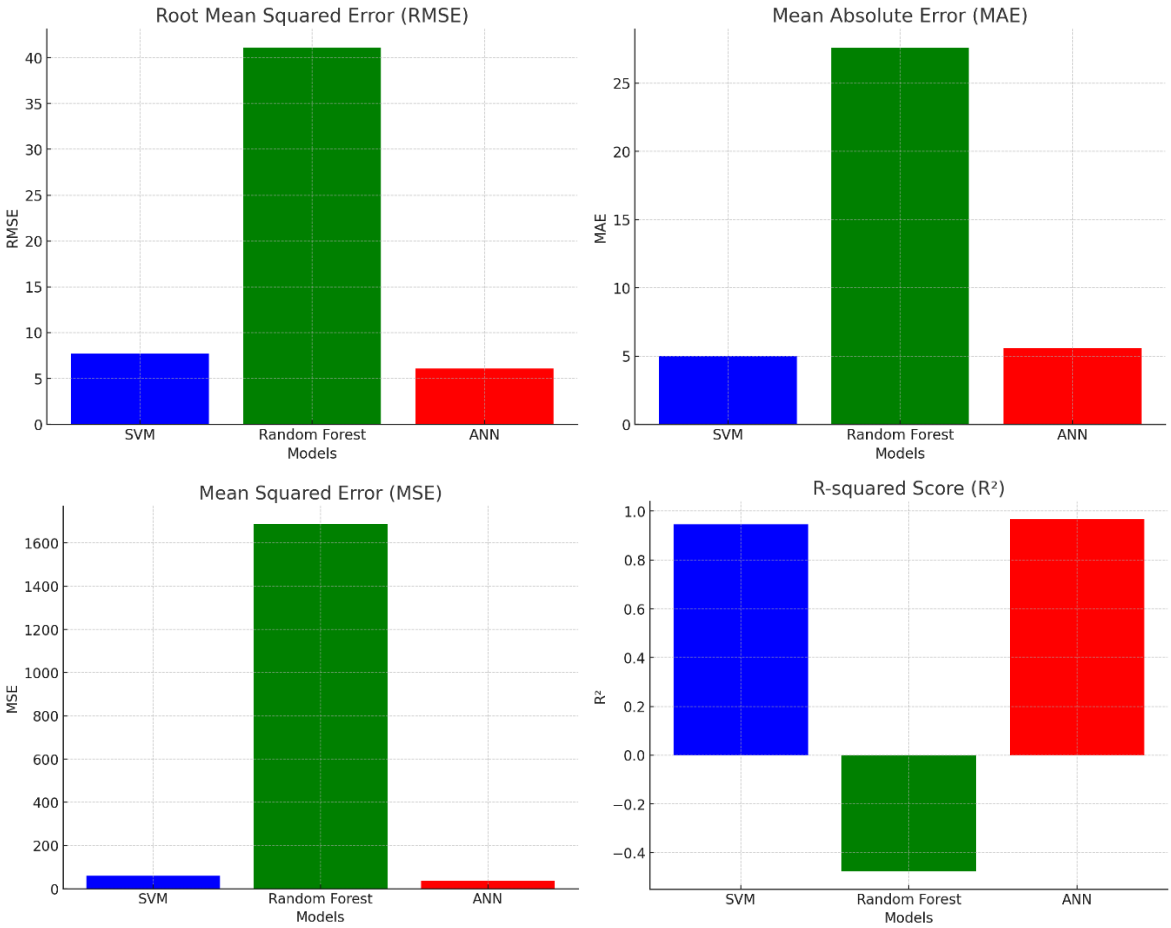
- Achieved a low **MAE** of **5.0309**, indicating good accuracy in predictions.
- **MSE** (59.5746) and **RMSE** (7.7185) were also relatively low, confirming the model's effectiveness.
- An **R^2** score of **0.9478** indicates that SVM explains **94.78%** of the variance in the target variable, demonstrating strong predictive power.

2. **Random Forest Model**:

- Performed poorly compared to other models, with the highest **MAE** (27.5821), **MSE** (1686.8755), and **RMSE** (41.0716).
- An **R^2** score of **-0.4770** indicates the model was not able to capture the variance, leading to highly inaccurate predictions.

3. ANN Model:

- Displayed robust performance, with **MAE** (5.5748), **MSE** (36.8035), and **RMSE** (6.0666) among the best results.
- An **R^2** score of **0.9678** highlights the model's ability to explain **96.78%** of the variance, making it the top-performing model in terms of overall accuracy.



Insights from Graphs

The graphs above provide a visual comparison of the models across all four metrics. ANN and SVM performed significantly better in terms of MAE, MSE, and RMSE, while ANN achieved the highest R^2 score. Random Forest showed the weakest performance overall, highlighting its unsuitability for this dataset.

These outcomes suggest that **ANN** is the most reliable model for stock price prediction in this study, followed closely by **SVM**.

10. Conclusion

This project successfully evaluated the performance of three machine learning models—SVM, Random Forest and ANN—for stock price prediction. By analysing historical stock data and leveraging these models, key insights into their predictive capabilities were uncovered.

The **Artificial Neural Network (ANN)** emerged as the most effective model, delivering the lowest error rates and the highest R^2 score, demonstrating its ability to capture complex, nonlinear relationships in the data. The **SVM** model also showed commendable performance, making it a reliable option for stock price prediction. In contrast, the **Random Forest** model underperformed, indicating its limitations with this specific dataset.

This study underscores the importance of selecting appropriate models and methods for financial forecasting. It also highlights how machine learning tools can provide significant advantages in understanding and predicting stock price movements, aiding stakeholders in making informed financial decisions.

11. Future Work

This project provides a foundation for stock price prediction, but further improvements are possible:

1. **Model Optimization:** Fine-tuning hyperparameters can enhance performance.
2. **Incorporating Additional Features:** Adding macroeconomic indicators and market sentiment data, such as news or social media trends, can improve prediction accuracy.
3. **Hybrid Models:** Combining models like LSTM and ANN can leverage their strengths for better results.
4. **Real-Time Prediction:** Extending models to handle real-time data streams will make them suitable for live market applications.
5. **Advanced Architectures:** Exploring models like Transformers or GRUs could further improve sequential data modelling.
6. **Decision Support Systems:** Integrating models into tools for risk management, portfolio optimization, or automated trading strategies can provide actionable insights.

These enhancements could lead to more robust and practical stock prediction systems.

12. References

[1] D. Patel, A. Roy, "A Review of Machine Learning Techniques in Stock Market Prediction," Int. J. Advanced Research in Computer Science, vol. 12, no. 2, pp. 1-10, 2023.

[2] M. Shynkevich et al., "Stock price forecasting using sentiment analysis and machine learning," Decision Support Systems, vol. 98, pp. 47–54, 2022.

[3] C. Li, F. Jiang, "Deep Learning for Stock Market Prediction: A Survey," ACM Computing Surveys, vol. 55, no. 4, pp. 1–36, 2023.

[4] G. Bollen, J. Mao, X. Zeng, "Twitter mood predicts the stock market," J. Computational Science, vol. 2, pp. 1-8, 2019.

[5] K. Nagel, S. Rasmussen, C. Barrett, Network trac as a self-organized critical phenomenon, in: F. Schweitzer (Ed.), Self-organization of Complex Structures: From Individual to Collective Dynamics, Gordon and Breach, p. 579.

[6] B. Kahin, J. Keller (Eds.), MIT Press, Cambridge, MA, 1977.

[7] S. Tang et al., "Stock market prediction through feature fusion and ensemble models," Information Sciences, vol. 607, pp. 1-17, 2023.

[8] X. Zhang, J. Chen, "Time-series modelling for stock prices with deep learning techniques," IEEE Trans. Neural Networks, vol. 33, pp. 1-10, 2024.

[9] Predictive Analytics for Stock Market Trends using Machine Learning | IEEE Xplore, 2024, [Online]. Available: https://ieeexplore.ieee.org/document/10449528.