📈 Stock Market Prediction

📌 1. Introduction

--> This notebook aims to predict stock price trends using historical stock market data.

--> We will use **LSTM (Long Short-Term Memory)** and **XGBoost** models for prediction.

--> The dataset includes stock prices, technical indicators, and market trends.

🔹 2. Installing & Importing Required Libraries

```
#Installing Required Libraries

!pip install tensorflow pandas numpy matplotlib scikit-learn
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.11/dist-packages (2.18.0)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (1.26.4)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from tensorflow) (24.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in /usr/local/
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from tensorflow) (75.1.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.2)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.71.0)
Requirement already satisfied: tensorboard<2.19,>=2.18 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.18.0)
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.8.0)
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.12.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.11/dist-packages (from tensorflo
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4.56.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.1)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.11/dist-packages (from astunparse>=1.6.0->tensorfl
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (0.14.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensor
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensor
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2.18->tenso
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from tensorbo
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2.18->tenso
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.11/dist-packages (from werkzeug>=1.0.1->tensorboard
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tens
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->te
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich->keras>
```

```
#Importing Required Libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
```

```
from sklearn.preprocessing import MinMaxScaler
from tensorflow import keras
from tensorflow.keras.layers import LSTM
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

📂 3. Loading the Dataset

```
# Load stock market data from a CSV file

df=pd.read_csv('/content/stock_details_5_years.csv')
```

🔄 4. Data Preprocessing

Printing the Dataset

```
print(df.head(10))
```

```
                        Date        Open        High         Low       Close  \
0  2018-11-29 00:00:00-05:00   43.829761   43.863354   42.639594   43.083508
1  2018-11-29 00:00:00-05:00  104.769074  105.519257  103.534595  104.636131
2  2018-11-29 00:00:00-05:00   54.176498   55.007500   54.099998   54.729000
3  2018-11-29 00:00:00-05:00   83.749496   84.499496   82.616501   83.678497
4  2018-11-29 00:00:00-05:00   39.692784   40.064904   38.735195   39.037853
5  2018-11-29 00:00:00-05:00  135.919998  139.990005  135.660004  138.679993
6  2018-11-29 00:00:00-05:00   23.133333   23.166668   22.636667   22.744667
7  2018-11-29 00:00:00-05:00  106.370278  108.796588  106.065834  107.938614
8  2018-11-29 00:00:00-05:00  135.973059  135.982718  134.059447  134.436371
9  2018-11-29 00:00:00-05:00   33.520714   33.891693   33.450050   33.503048

      Volume  Dividends  Stock Splits Company
0  167080000       0.00           0.0    AAPL
1   28123200       0.00           0.0    MSFT
2   31004000       0.00           0.0   GOOGL
3  132264000       0.00           0.0    AMZN
4   54917200       0.04           0.0    NVDA
5   24238700       0.00           0.0    META
6   46210500       0.00           0.0    TSLA
7    4688300       0.00           0.0     LLY
8    8751500       0.00           0.0       V
9    7056600       0.00           0.0     TSM
```

Checking for missing values and handling them

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 602962 entries, 0 to 602961
Data columns (total 9 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   Date          602962 non-null  object
 1   Open          602962 non-null  float64
 2   High          602962 non-null  float64
 3   Low           602962 non-null  float64
 4   Close         602962 non-null  float64
 5   Volume        602962 non-null  int64
 6   Dividends     602962 non-null  float64
 7   Stock Splits  602962 non-null  float64
 8   Company       602962 non-null  object
dtypes: float64(6), int64(1), object(2)
memory usage: 41.4+ MB
```

📊 5. Feature Engineering - Creating Technical Indicators

Simple Moving Averages (SMA)

```
df['SMA_10'] = df['Close'].rolling(window=10).mean()  # Short-Term Trend
df['SMA_50'] = df['Close'].rolling(window=50).mean()  # Long-Term Trend
```

Exponential Moving Averages (EMA)

```
df['EMA_10'] = df['Close'].ewm(span=10, adjust=False).mean()
df['EMA_50'] = df['Close'].ewm(span=50, adjust=False).mean()
```

Relative Strength Index (RSI)

```
# Relative Strength Index (RSI)
def compute_RSI(data, window=14):
    delta = data.diff()
    gain = (delta.where(delta > 0, 0)).rolling(window=window).mean()
    loss = (-delta.where(delta < 0, 0)).rolling(window=window).mean()
    RS = gain / loss
    # Avoid division by zero
    RS = RS.replace([np.inf, -np.inf], np.nan).fillna(0)
    return 100 - (100 / (1 + RS))

df['RSI'] = compute_RSI(df['Close'])
```

Moving Average Convergence Divergence (MACD)

```
# Moving Average Convergence Divergence (MACD)
df['MACD'] = df['EMA_10'] - df['EMA_50']
```

Target Variable (Price Increase or Decrease)

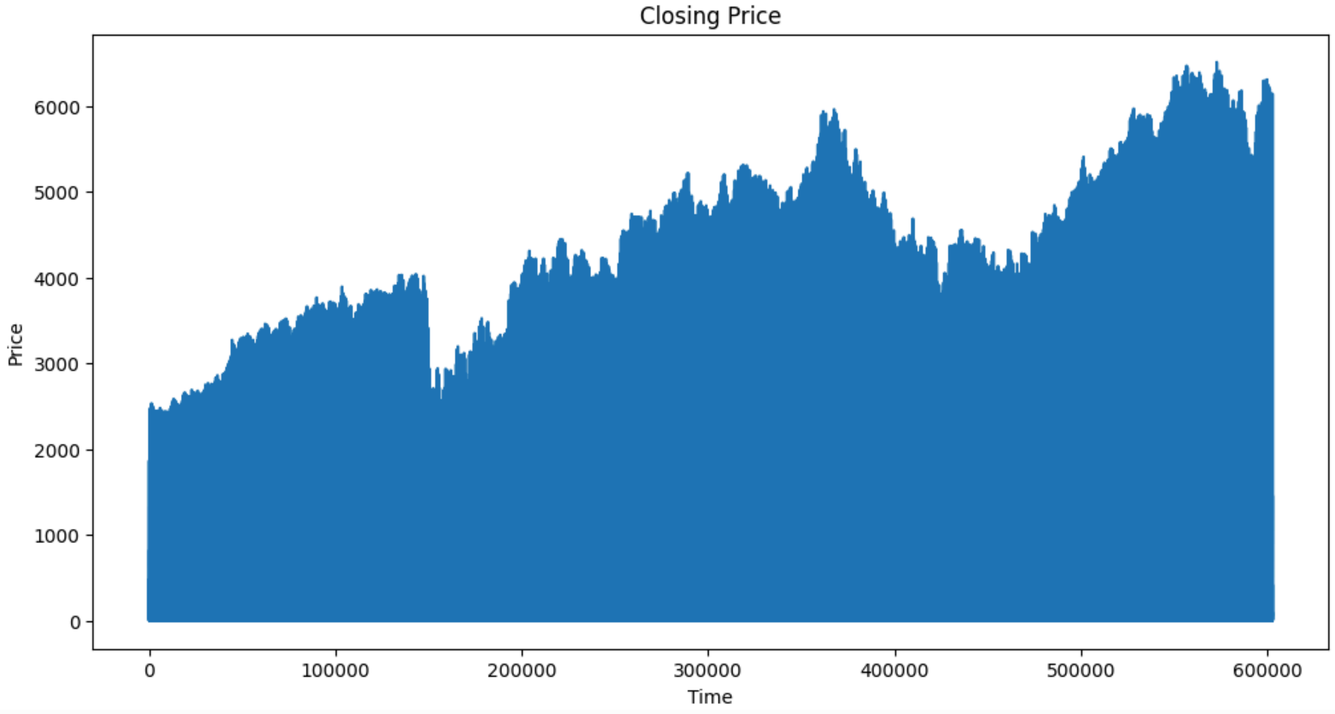
```
df['Target'] = np.where(df['Close'].shift(-1) > df['Close'], 1, 0)
df.dropna(inplace=True)  # Remove last row (shift created NaN)
```

```
print(df["Target"].value_counts())
```

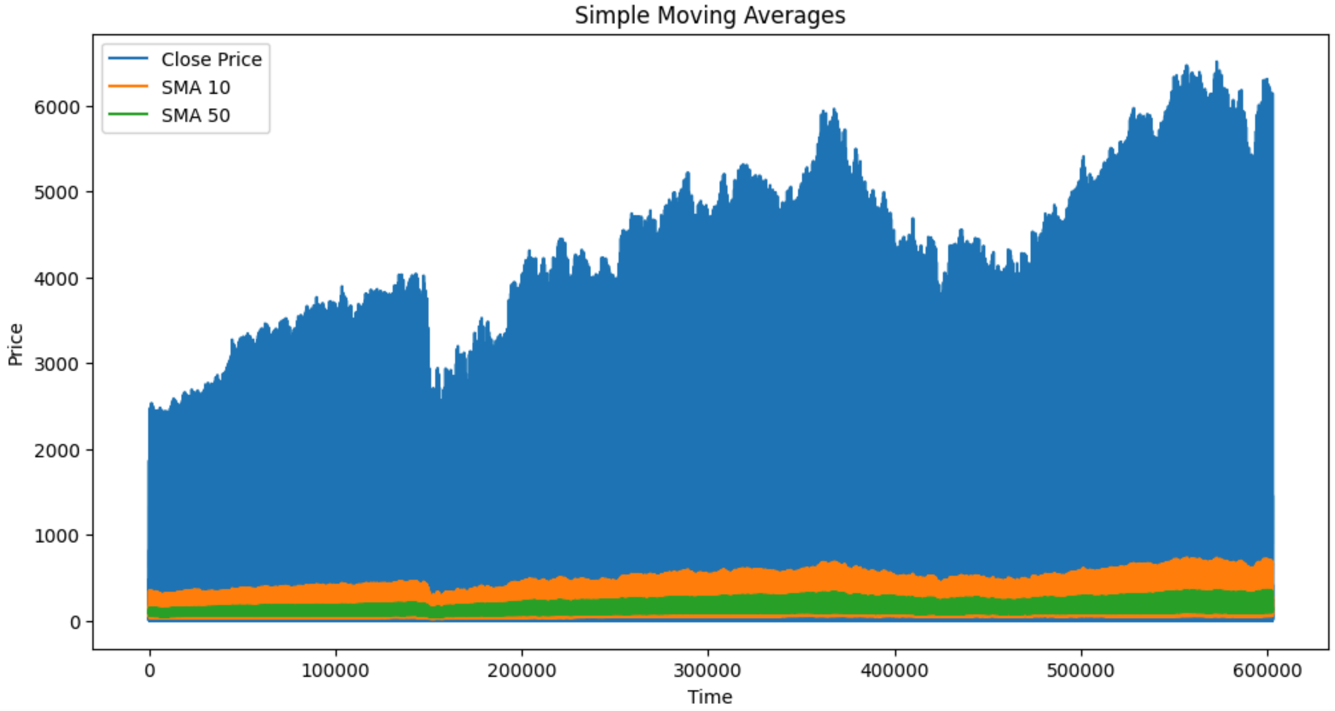
```
Target
0    312350
1    290563
Name: count, dtype: int64
```

📉 6. Data Visualization

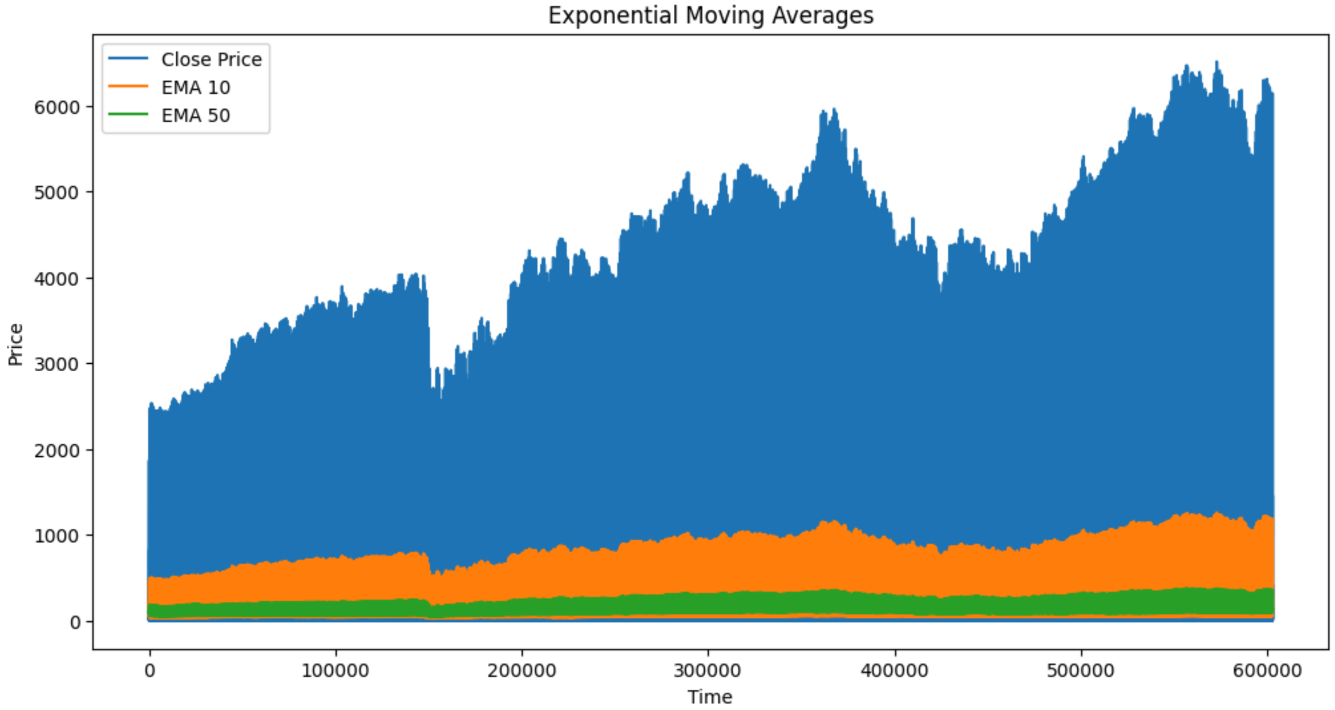
```
# Plotting the closing price
plt.figure(figsize=(12, 6))
plt.plot(df['Close'])
plt.title('Closing Price')
plt.xlabel('Time')
plt.ylabel('Price')
plt.show()
```



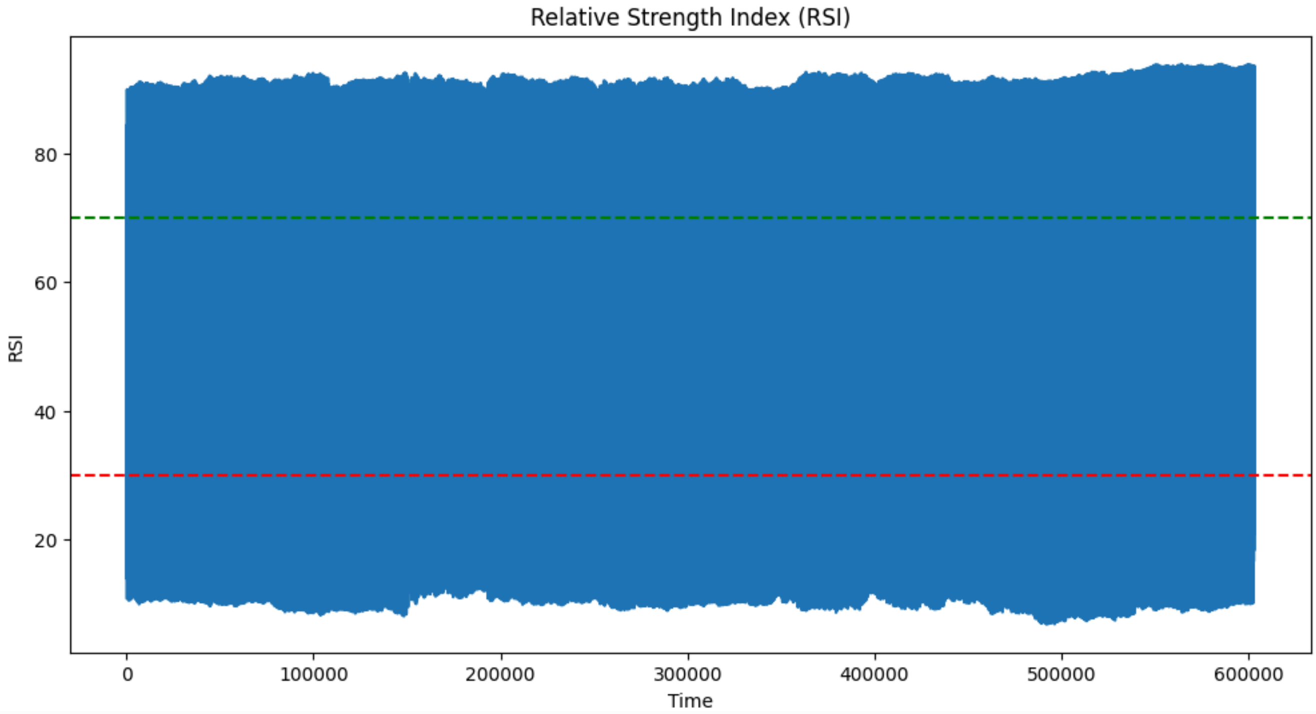
```
# Plotting SMA
plt.figure(figsize=(12, 6))
plt.plot(df['Close'], label='Close Price')
plt.plot(df['SMA_10'], label='SMA 10')
plt.plot(df['SMA_50'], label='SMA 50')
plt.title('Simple Moving Averages')
plt.xlabel('Time')
plt.ylabel('Price')
plt.legend()
plt.show()
```



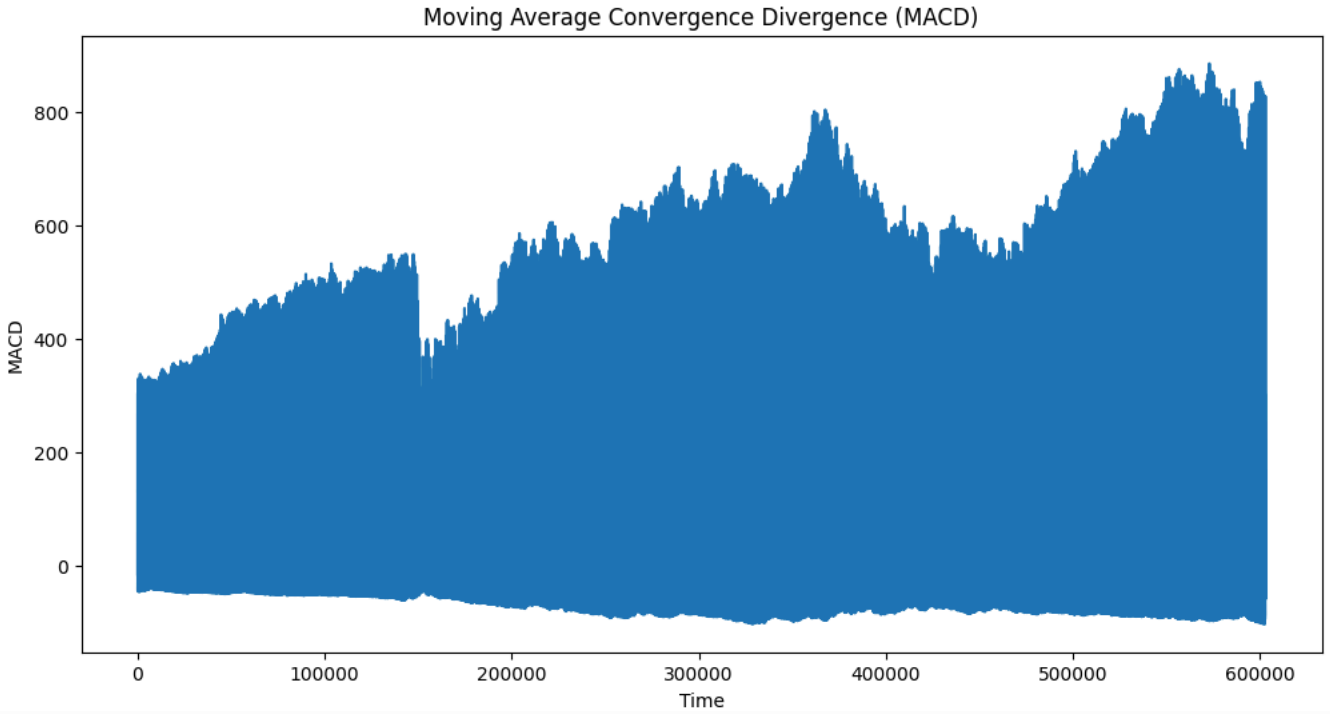
```
# Plotting EMA
plt.figure(figsize=(12, 6))
plt.plot(df['Close'], label='Close Price')
plt.plot(df['EMA_10'], label='EMA 10')
plt.plot(df['EMA_50'], label='EMA 50')
plt.title('Exponential Moving Averages')
plt.xlabel('Time')
plt.ylabel('Price')
plt.legend()
plt.show()
```



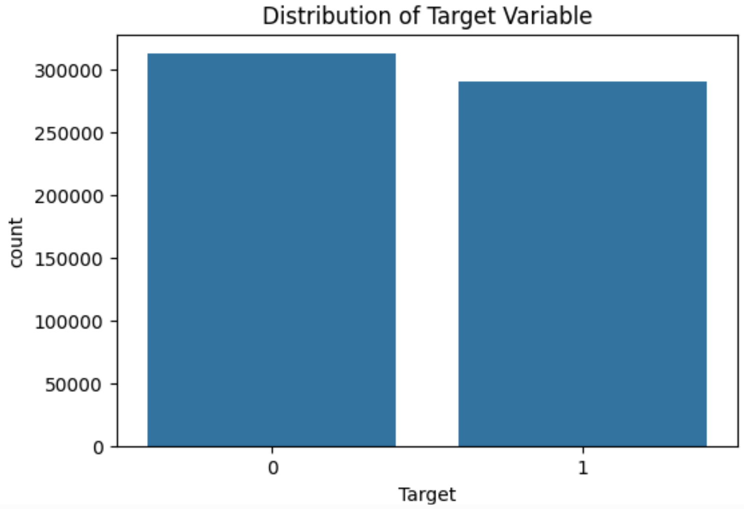
```
# Plotting RSI
plt.figure(figsize=(12, 6))
plt.plot(df['RSI'])
plt.title('Relative Strength Index (RSI)')
plt.xlabel('Time')
plt.ylabel('RSI')
plt.axhline(y=30, color='r', linestyle='--')  # Oversold line
plt.axhline(y=70, color='g', linestyle='--')  # Overbought line
plt.show()
```



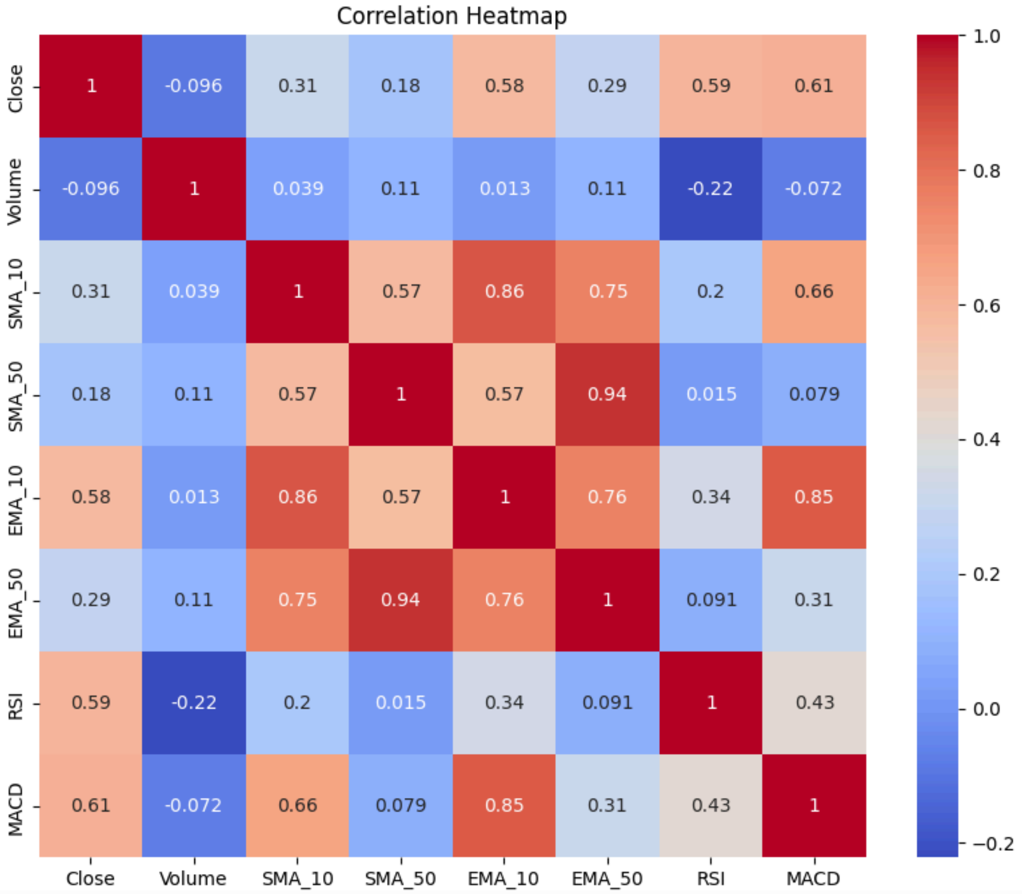
```
# Plotting MACD
plt.figure(figsize=(12, 6))
plt.plot(df['MACD'])
plt.title('Moving Average Convergence Divergence (MACD)')
plt.xlabel('Time')
plt.ylabel('MACD')
plt.show()
```



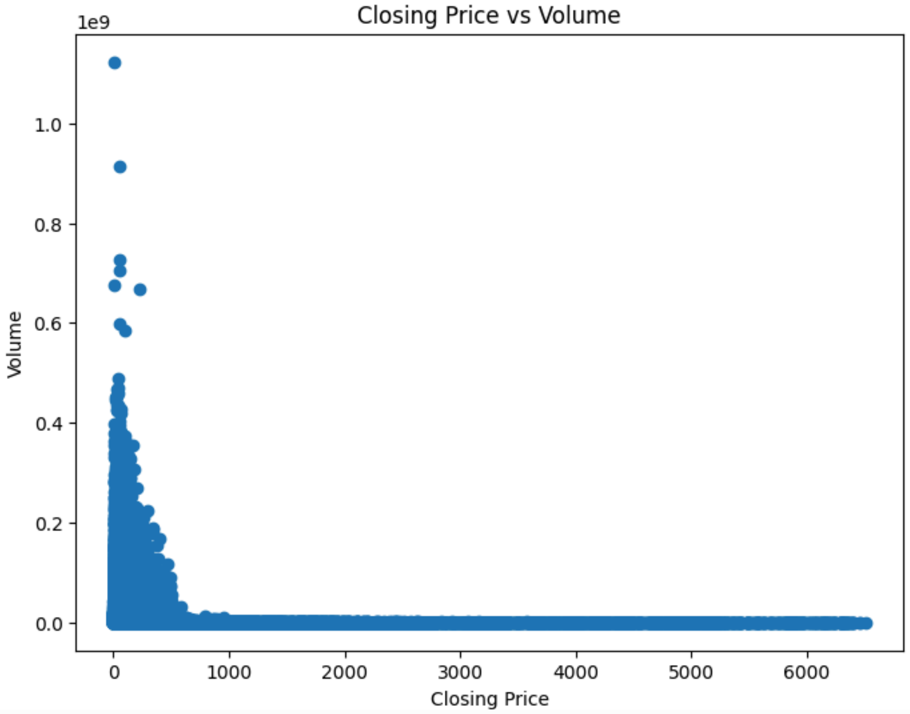
```
# Plot the distribution of the target variable
plt.figure(figsize=(6, 4))
sns.countplot(x='Target', data=df)
plt.title('Distribution of Target Variable')
plt.show()
```



```
# Correlation Heatmap
plt.figure(figsize=(10, 8))
correlation_matrix = df[['Close', 'Volume', 'SMA_10', 'SMA_50', 'EMA_10', 'EMA_50', 'RSI', 'MACD']].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

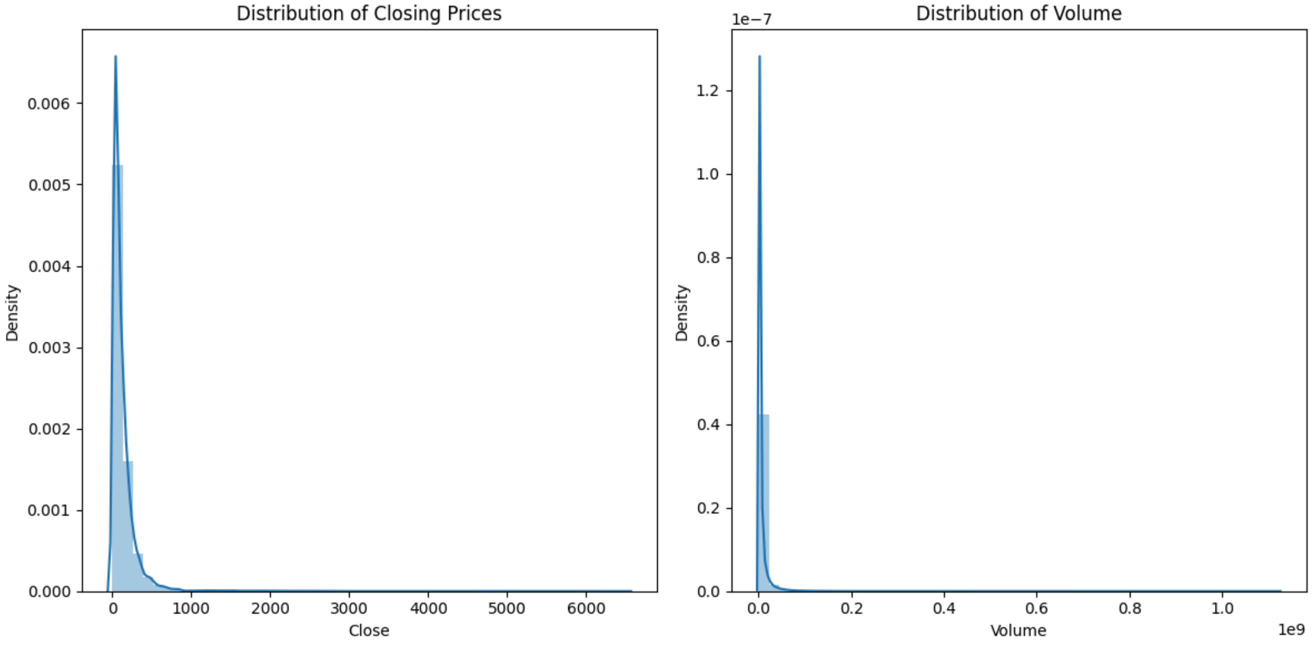


```
# Scatter plot of Closing Price vs Volume
plt.figure(figsize=(8,6))
plt.scatter(df['Close'],df['Volume'])
plt.xlabel("Closing Price")
plt.ylabel("Volume")
plt.title("Closing Price vs Volume")
plt.show()
```



```
# Distribution plots of key features
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.distplot(df['Close'])
plt.title('Distribution of Closing Prices')
plt.subplot(1, 2, 2)
sns.distplot(df['Volume'])
plt.title('Distribution of Volume')
plt.tight_layout()
plt.show()
```

```
<ipython-input-44-fdde5e33e9d9>:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Close'])
<ipython-input-44-fdde5e33e9d9>:8: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Volume'])
```



🔄 7. Data Preparation for Training

Selecting Features and Target

```
#Make a training dataset
Data= df[['Close', 'Open', 'High', 'Low', 'SMA_10', 'SMA_50', 'EMA_10', 'EMA_50', 'RSI', 'MACD','Target']]


#Separating The Feature and Target Variable
X = Data[['Close', 'Open', 'High', 'Low', 'SMA_10', 'SMA_50', 'EMA_10', 'EMA_50', 'RSI', 'MACD']]  # Features
y = Data["Target"]  # Target variable

# Now you can split the data:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
#Normalizing Data
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

🔥 8. Training LSTM Model

```
# Reshape the input data to be 3-dimensional
# [samples, timesteps, features]
if len(X_train.shape) != 3:
    X_train = X_train.reshape(X_train.shape[0], 1, X_train.shape[1])

if len(X_test.shape) != 3:
    X_test = X_test.reshape(X_test.shape[0], 1, X_test.shape[1])


# Build LSTM Model
model = Sequential([
    LSTM(128, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])), # Use correct input shape
    Dropout(0.3),
    LSTM(64, return_sequences=False),
    Dropout(0.2),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')
])

# Compile Model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=['accuracy'])
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_dim`
  super().__init__(**kwargs)
```

```
history = model.fit(X_train, y_train, epochs=50, batch_size=16, validation_data=(X_test, y_test))
```

```
Epoch 1/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 203s 7ms/step - accuracy: 0.7392 - loss: 0.5171 - val_accuracy: 0.7600 - val_loss: 0.4856
Epoch 2/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 187s 6ms/step - accuracy: 0.7584 - loss: 0.4829 - val_accuracy: 0.7680 - val_loss: 0.4631
Epoch 3/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 207s 6ms/step - accuracy: 0.7641 - loss: 0.4673 - val_accuracy: 0.7835 - val_loss: 0.4390
Epoch 4/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 202s 6ms/step - accuracy: 0.7753 - loss: 0.4484 - val_accuracy: 0.7989 - val_loss: 0.4114
Epoch 5/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 190s 6ms/step - accuracy: 0.7894 - loss: 0.4280 - val_accuracy: 0.8131 - val_loss: 0.3853
Epoch 6/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 190s 6ms/step - accuracy: 0.7985 - loss: 0.4103 - val_accuracy: 0.8236 - val_loss: 0.3686
Epoch 7/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 186s 6ms/step - accuracy: 0.8075 - loss: 0.3955 - val_accuracy: 0.8331 - val_loss: 0.3537
Epoch 8/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 202s 6ms/step - accuracy: 0.8141 - loss: 0.3840 - val_accuracy: 0.8424 - val_loss: 0.3394
Epoch 9/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 186s 6ms/step - accuracy: 0.8194 - loss: 0.3742 - val_accuracy: 0.8503 - val_loss: 0.3195
Epoch 10/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 190s 6ms/step - accuracy: 0.8234 - loss: 0.3645 - val_accuracy: 0.8579 - val_loss: 0.3075
Epoch 11/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 191s 6ms/step - accuracy: 0.8287 - loss: 0.3569 - val_accuracy: 0.8604 - val_loss: 0.2966
Epoch 12/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 188s 6ms/step - accuracy: 0.8337 - loss: 0.3478 - val_accuracy: 0.8697 - val_loss: 0.2865
Epoch 13/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 205s 6ms/step - accuracy: 0.8376 - loss: 0.3415 - val_accuracy: 0.8711 - val_loss: 0.2788
Epoch 14/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 198s 6ms/step - accuracy: 0.8401 - loss: 0.3356 - val_accuracy: 0.8731 - val_loss: 0.2805
Epoch 15/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 202s 6ms/step - accuracy: 0.8438 - loss: 0.3302 - val_accuracy: 0.8790 - val_loss: 0.2689
Epoch 16/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 191s 6ms/step - accuracy: 0.8458 - loss: 0.3264 - val_accuracy: 0.8789 - val_loss: 0.2592
Epoch 17/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 190s 6ms/step - accuracy: 0.8474 - loss: 0.3232 - val_accuracy: 0.8868 - val_loss: 0.2541
Epoch 18/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 200s 6ms/step - accuracy: 0.8512 - loss: 0.3163 - val_accuracy: 0.8851 - val_loss: 0.2569
Epoch 19/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 204s 6ms/step - accuracy: 0.8527 - loss: 0.3145 - val_accuracy: 0.8926 - val_loss: 0.2433
Epoch 20/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 206s 6ms/step - accuracy: 0.8547 - loss: 0.3109 - val_accuracy: 0.8958 - val_loss: 0.2372
Epoch 21/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 193s 6ms/step - accuracy: 0.8567 - loss: 0.3068 - val_accuracy: 0.8971 - val_loss: 0.2335
Epoch 22/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 201s 6ms/step - accuracy: 0.8589 - loss: 0.3042 - val_accuracy: 0.9031 - val_loss: 0.2278
Epoch 23/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 203s 6ms/step - accuracy: 0.8603 - loss: 0.3020 - val_accuracy: 0.9039 - val_loss: 0.2233
Epoch 24/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 201s 6ms/step - accuracy: 0.8623 - loss: 0.2991 - val_accuracy: 0.9044 - val_loss: 0.2218
Epoch 25/50
```

```
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 201s 6ms/step - accuracy: 0.8634 - loss: 0.2953 - val_accuracy: 0.9024 - val_loss: 0.2222
Epoch 26/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 199s 6ms/step - accuracy: 0.8643 - loss: 0.2948 - val_accuracy: 0.9081 - val_loss: 0.2174
Epoch 27/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 199s 6ms/step - accuracy: 0.8662 - loss: 0.2923 - val_accuracy: 0.9082 - val_loss: 0.2164
Epoch 28/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 185s 6ms/step - accuracy: 0.8670 - loss: 0.2895 - val_accuracy: 0.9092 - val_loss: 0.2164
Epoch 29/50
30146/30146 ━━━━━━━━━━━━━━━━━━━━ 208s 6ms/step - accuracy: 0.8685 - loss: 0.2867 - val_accuracy: 0.9065 - val_loss: 0.2175
```

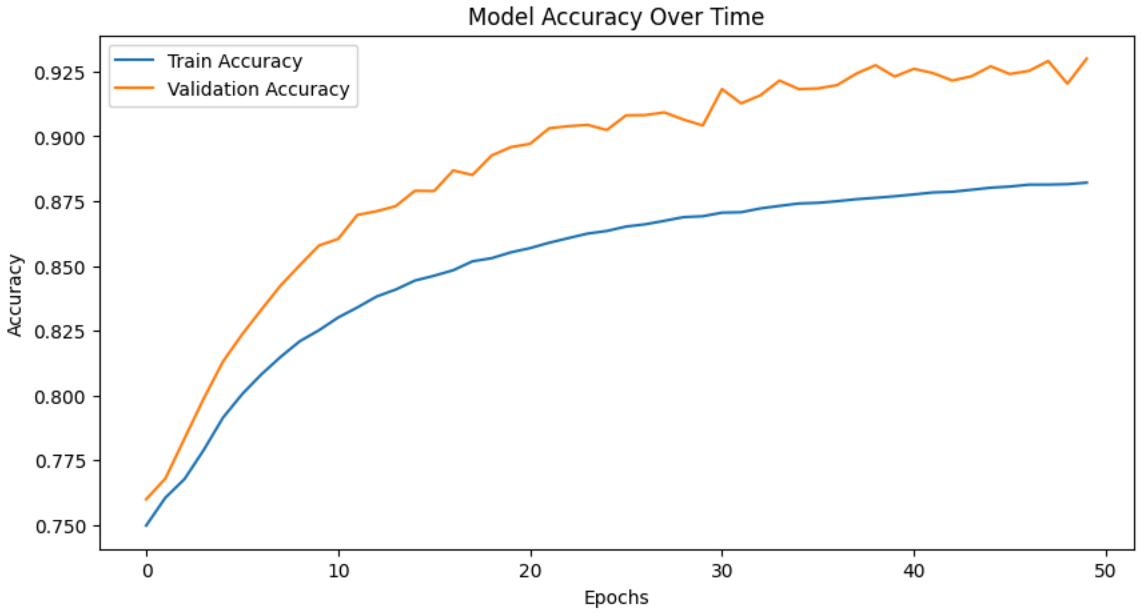
Finding accuracy Of the model

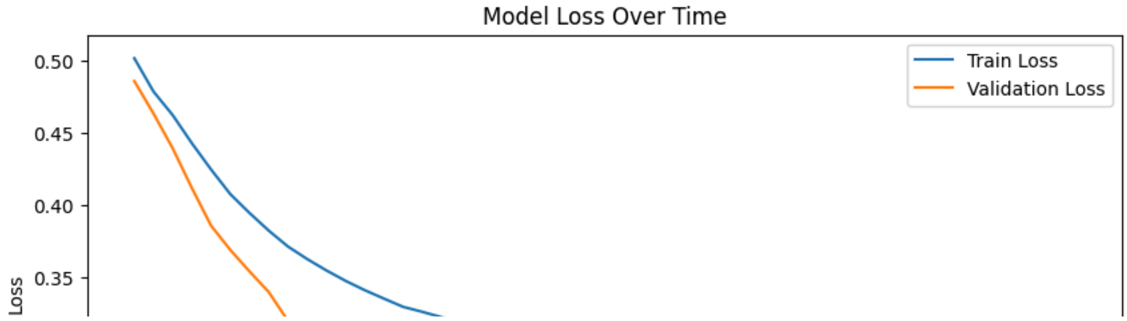
```
# Evaluate Model Performance
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Neural Network Accuracy: {accuracy * 100:.2f}%")
```

```
3769/3769 ━━━━━━━━━━━━━━━━━━━━ 12s 3ms/step - accuracy: 0.9301 - loss: 0.1782
Neural Network Accuracy: 93.00%
```

📊 10. Model Performance Visualization

```
# Plot Training vs Validation Accuracy
plt.figure(figsize=(10,5))
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Model Accuracy Over Time')
plt.show()

# Plot Training vs Validation Loss
plt.figure(figsize=(10,5))
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Model Loss Over Time')
plt.show()
```


Model Accuracy Over Time
Train Accuracy
Validation Accuracy
Accuracy
Epochs


Model Loss Over Time
Train Loss
Validation Loss
Loss