# Title: Stock Market Prediction

Team Number: 19

Team Members:

Shubham Sharma 22BAI1258

Aryan Jewalia 22BAI1194

Shreya Mittal 22BAI1131

**Introduction:**

Stock market prediction projects hold significant value in the financial domain due to their potential to provide insights into market trends, inform investment decisions, and mitigate financial risks. With the volatility and unpredictability inherent in the stock market, accurate forecasting of stock prices becomes imperative for investors, traders, and financial institutions alike. Leveraging advanced machine learning techniques like Random Forest regression offers a promising approach to tackle this challenge by harnessing the power of data-driven models to uncover underlying patterns and relationships in financial data.

In this project, we focus on predicting stock prices using Random Forest regression, a powerful ensemble learning technique known for its robustness and predictive accuracy. By employing Random Forest regression, we aim to capitalize on its capability to capture complex relationships and nonlinear dependencies in financial data, thereby enhancing our ability to forecast stock prices effectively.

The primary objective of this study is to assess the effectiveness of Random Forest regression in stock market prediction. By evaluating the model's performance metrics, such as Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R²) score, we seek to quantify its predictive accuracy and reliability. Additionally, we aim to understand the implications of employing Random Forest regression for investors, traders, and financial institutions, highlighting its potential to inform investment strategies, optimize portfolio management, and mitigate financial risks.

Furthermore, this project focuses specifically on predicting the closing price of stocks, a key metric that encapsulates the market sentiment, supply-demand dynamics, and investor behavior. By utilizing features such as previous closing prices, trading volume, and technical indicators derived from historical stock data, we aim to develop a robust predictive model capable of accurately forecasting future closing prices.

Through comprehensive data analysis, feature engineering, model training, and evaluation, we endeavor to provide valuable insights into the application of Random Forest regression for stock market prediction. By elucidating the strengths, limitations, and practical implications of this approach, we aim to contribute to the ongoing discourse on leveraging machine learning for financial forecasting and decision-making.

**Abstract:**

This report delves into the utilization of Random Forest (RF) regression as a means to enhance stock market predictions. Random Forest, renowned for its accuracy and resilience in managing intricate datasets, is chosen for its potential to effectively capture the complexities of financial data. The report commences by elucidating the principles of RF and delineating its superiority over traditional models, setting the stage for its integration into stock market analysis. A comprehensive case study is presented to showcase the practical application of RF, encompassing crucial stages such as data preprocessing, model training, and evaluation. Through this exploration, the report aims to shed light on the efficacy of RF regression in augmenting stock market predictions, offering valuable insights for investors, traders, and financial institutions alike.

* **Methodology**

The methodology involved several key steps:

* Data collection and preprocessing: Historical stock market data was sourced and preprocessed to ensure data quality and relevance.
* Feature engineering: Relevant attributes such as previous closing prices and trading volume were selected, and technical indicators like Simple Moving Average (SMA) and Exponential Moving Average (EMA) were derived to capture important aspects of market dynamics.
* Model training: A Random Forest Regression model was trained using the preprocessed data to predict stock prices.
* Model evaluation: The performance of the model was evaluated using metrics such as Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R² Score.

**LITERATURE SURVEY**

**1.Survey of stock market prediction using machine learning approach Authors: Ashish Sharma ; Dinesh Bhuriya ; Upendra Singh 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)**

Stock market is basically nonlinear in nature and the research on stock market is one of the most important issues in recent years. People invest in stock market based on some prediction. For predict, the stock market prices people search such methods and tools which will increase their profits, while minimize their risks. Prediction plays a very important role in stock market business which is very complicated and challenging process. Employing traditional methods like fundamental and technical analysis may not ensure the reliability of the prediction. To make predictions regression analysis is used mostly. In this paper we survey of well-known efficient regression approach to predict the stock market price from stock market data based. In future the results of multiple regression approach could be improved using more number of variables

2.**Stock market prediction using an improved training algorithm of neural network Authors: Mustain Billah ; Sajjad Waheed ; Abu Hanifa,2016 2nd International Conference on Electrical, Computer & Telecommunication Engineering (ICECTE)**

Predicting closing stock price accurately is an challenging task. Computer aided systems have been proved to be helpful tool for stock prediction such as Artificial Neural Net-work(ANN), Adaptive Neuro Fuzzy Inference System (ANFIS) etc. Latest research works prove that Adaptive Neuro Fuzzy Inference System shows better results than Neural Network for stock prediction. In this paper, an improved Levenberg Marquardt(LM) training algorithm of artificial neural network has been proposed. Improved Levenberg Marquardt algorithm of neural network can predict the possible day-end closing stock price with less memory and time needed, provided previous historical stock market data of Dhaka Stock Exchange such as opening price, highest price, lowest price, total share traded. Morever, improved LM algorithm can predict day-end stock price with 53% less error than ANFIS and traditional LM algorithm. It also requires 30% less time, 54% less memory than traditional LM and 47% less time, 59% less memory than ANFIS.

3. . **Stock Market Movement Prediction using LDA-Online Learning Model Authors:Tanapon Tantisripreecha ; Nuanwan Soonthomphisaj, 2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)**

In this paper, an online learning method namely LDA-Online algorithm is proposed to predict the stock movement. The feature set which are the opening price, the closing price, the highest price and the lowest price are applied to fit the Linear Discriminant 7 Analysis (LDA). Experiments on the four well known NASDAQ stocks (APPLE, FACBOOK GOOGLE, and AMAZON) show that our model provide the best performance in stock prediction. We compare LDA-online to ANN, KNN and Decision Tree in both Batch and Online learning scheme. We found that LDA-Online provided the best performance. The highest performances measured on GOOGLE, AMAZON, APPLE FACEBOOK stocks are 97.81%, 97.64%, 95.58% and 95.18% respectively.

**4. Stock Market Prediction Analysis by Incorporating Social and News Opinion and Sentiment Authors: Zhaoxia Wang ; Seng-Beng Ho ; Zhiping Lin, 2018 IEEE International Conference on Data Mining Workshops (ICDMW)**

The price of the stocks is an important indicator for a company and many factors can affect their values. Different events may affect public sentiments and emotions differently, which may have an effect on the trend of stock market prices. Because of dependency on various factors, the stock prices are not static, but are instead dynamic, highly noisy and nonlinear time series data. Due to its great learning capability for solving the nonlinear time series prediction problems, machine learning has been applied to this research area. Learning-based methods for stock price prediction are very popular and a lot of enhanced strategies have been used to improve the performance of the learning based predictors. However, performing successful stock market prediction is still a challenge. News articles and social media data are also very useful and important in financial prediction, but currently no good method exists that can take these social media into consideration to provide better analysis of the financial market. This paper aims to successfully predict stock price through analyzing the relationship between the stock price and the news sentiments. A novel enhanced learning-based method for stock price prediction is proposed that considers the effect of news sentiments. Compared with existing learning-based methods, the effectiveness of this new enhanced learning-based method is demonstrated by using the real stock price data set with an improvement of performance in terms of reducing the Mean Square Error (MSE). The research work and findings

**Proposed System**

**Data Loading and Preprocessing**:

* + Loads stock market data from a CSV file into a Pandas DataFrame.
  + Converts the 'Date' column to datetime type and sets it as the index.
  + Handles missing values by filling missing 'Trades' values with 0 and forward filling other missing values.

1. **Feature Engineering**:
   * Creates lag features for 'Close' price to use the previous day's close as a feature.
   * Calculates additional features such as previous volume, 5-day Simple Moving Average (SMA), and 10-day Exponential Moving Average (EMA).
   * Drops rows with NaN values created by rolling functions.
2. **Model Training**:
   * Defines the features (inputs) and the target (output) for the regression model.
   * Splits the data into training and testing sets (70% train, 30% test) using **train\_test\_split**.
   * Initializes a **RandomForestRegressor** model with 100 estimators.
   * Trains the model on the training data using the **fit** method.
3. **Model Evaluation**:
   * Makes predictions on the testing set using the trained model.
   * Calculates performance metrics including Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R² Score.
   * Prints the calculated metrics and visualizes the relationship between actual and predicted stock prices using a scatter plot.
4. **Feature Importance**:
   * Calculates feature importances using the trained RandomForestRegressor model.
   * Displays the feature importances as a dictionary.
5. **Model Saving**:
   * Saves the trained RandomForestRegressor model to disk using **joblib.dump**.

**Python code:**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_squared\_error

from sklearn.preprocessing import StandardScaler

import numpy as np

from sklearn.metrics import mean\_absolute\_error, r2\_score

import matplotlib.pyplot as plt

import joblib

# Load the data

data\_path = "TCS.csv"

data = pd.read\_csv(data\_path)

# Convert 'Date' to datetime type and set as index

data['Date'] = pd.to\_datetime(data['Date'])

data.set\_index('Date', inplace=True)

# Handling missing values: fill missing 'Trades' with 0

data['Trades'] = data['Trades'].fillna(0)

# Check for other missing values and fill or drop

data.ffill(inplace=True)  # forward fill

# Creating lag features for 'Close' to use previous day's close as a feature

data['Prev\_Close'] = data['Close'].shift(1)

# Drop the first row as it now contains NaN (due to lag feature)

data.dropna(inplace=True)

# Define additional features

data['Prev\_Volume'] = data['Volume'].shift(1)

data['SMA\_5'] = data['Close'].rolling(window=5).mean().shift(1)

data['EMA\_10'] = data['Close'].ewm(span=10, adjust=False).mean().shift(1)

#print("sma", data['Close'].rolling(window=5).mean())

#print("ema", data['Close'].ewm(span=10, adjust=False).mean())

#print('prev\_close', data['Close'].shift(1))

#print('prev\_volume', data['Volume'].shift(1))

# Drop any additional rows with NaN values created by rolling functions

data.dropna(inplace=True)

# Define the features (inputs) and the target (output)

features = ['Prev\_Close', 'Prev\_Volume', 'SMA\_5', 'EMA\_10']

target = 'Close'

# Select the features and target from the DataFrame

X = data[features]  # Features

y = data[target]    # Target

# Split the data into training and testing sets (70% train, 30% test)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.35, random\_state=42)

print("Training set shape:", X\_train.shape)

print("Testing set shape:", X\_test.shape)

# Initialize the RandomForestRegressor

rf\_model = RandomForestRegressor(n\_estimators=100, random\_state=42)

# Train the model on the training data

rf\_model.fit(X\_train, y\_train)

# Make predictions on the testing set

y\_pred = rf\_model.predict(X\_test)

# Calculate the performance metrics, such as Root Mean Squared Error (RMSE)

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

print(f"Root Mean Squared Error: {rmse}")

# Calculate MAE

mae = mean\_absolute\_error(y\_test, y\_pred)

print(f"Mean Absolute Error: {mae}")

# Calculate R² Score

r2 = r2\_score(y\_test, y\_pred)

print(f"R² Score: {r2}")

importances = rf\_model.feature\_importances\_

feature\_names = X\_train.columns

feature\_importance\_dict = dict(zip(feature\_names, importances))

print("Feature Importances:", feature\_importance\_dict)

plt.figure(figsize=(10, 6))

plt.scatter(y\_test, y\_pred, alpha=0.5)

plt.xlabel('Actual Prices')

plt.ylabel('Predicted Prices')

plt.title('Actual vs Predicted Stock Prices')

plt.plot([min(y\_test), max(y\_test)], [min(y\_test), max(y\_test)], 'k--')  # a reference line

plt.show()

# Save the model to disk

joblib.dump(rf\_model, 'random\_forest\_stock\_model.pkl')

**Output Screenshots**

**A graph showing a blue line

Description automatically generated with medium confidence**

**A computer screen with white text

Description automatically generated**