Assessment Report

on

"Stock Price Prediction"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AI)

By

S. No	Name	University Roll Number
18	Shourya Garg	202401100300238
19	Shourya Ojha	202401100300239
21	Shreya Tyagi	202401100300241
42	Tanu Singh	202401100300262

Section: D

Under the supervision of

"Mr. Abhishek Shukla"

KIET Group of Institutions, Ghaziabad

1. Problem Statement

The goal of this project is to develop a regression-based predictive model that can estimate the next-day closing stock price of companies listed in the Nifty 50 index based on their historical stock market data. By analysing past stock prices and volume trends, the model aims to forecast future prices to assist investors and traders in making better-informed decisions.

Key challenges include dealing with the inherent volatility and noise in stock market data, selecting relevant features, and evaluating the model's predictive accuracy.

2. Introduction

Predicting stock prices has always been a significant challenge and a topic of great interest in financial markets. Investors, analysts, and researchers constantly strive to develop models that can accurately forecast stock trends and support decision-making. In this project, we aim to build a **regression-based machine learning model** that predicts **next-day stock prices** using **historical stock data**.

The dataset used for this project is from the **Nifty 50** stock market index, which includes daily historical prices of 50 major Indian companies listed on the National Stock Exchange (NSE). The dataset contains key features such as **Open, High, Low, Close prices, and Volume**, which are essential for understanding market trends.

Why Stock Price Prediction?

Stock price prediction can help in:

- Making informed investment decisions.
- Understanding and visualizing market behavior.
- Exploring the application of machine learning in finance.

The goal is not to achieve perfect accuracy—since markets are influenced by numerous unpredictable factors—but to build a model that can reasonably approximate future trends based on historical data.

---

3. Objectives

- To analyze historical stock price data of Nifty 50 companies.
- To develop a regression model that predicts next-day closing prices.
- To visualize stock price trends and patterns over time.
- To evaluate the model's prediction accuracy using error metrics.
- To provide insights that can assist in investment decision-making.

---

4. Methodology

To predict the next-day stock prices using historical data, the following approach was adopted:

A. Data Loading and Preparation

- **Data source:** Multiple CSV files inside a ZIP archive (`StockMarket.zip`) containing historical stock market data.
- **Goal:** Predict the **next day's closing price** of a stock using available numerical features.
- **Non-numeric columns:** Removed because machine learning models generally require numerical input.
- **Focus columns:** Only numeric features such as Open, High, Low, Close, Volume, etc.

B. Feature Engineering

- Created **lag features** from the `Close` price:
    - Lag 1: `Close` price from 1 day ago

    - Lag 2: `Close` price from 2 days ago
    - Lag 3: `Close` price from 3 days ago
- Reason: Stock prices are highly time-dependent and influenced by recent past values.
- Created target variable `Close_next` as the **closing price shifted one day ahead**.

C. Data Cleaning

- Dropped rows with **NaN values** created due to shifting operations.

D. Data Splitting

- Split data into **training set (80%)** and **testing set (20%)** randomly but reproducibly (`random_state=42`).
- Purpose: Train model on past data, test on unseen data to evaluate performance.

E. Data Scaling

- Used `StandardScaler` to standardize features (mean=0, variance=1).
- Scaling is important for many machine learning models to converge efficiently.

F. Model Selection

- Initially tried **Linear Regression** but got poor accuracy (very high MSE).
- Switched to **HistGradientBoostingRegressor** (a tree-based ensemble method) which:
    - Handles non-linear relationships well
    - Is faster than some other ensemble methods like Random Forests or GradientBoosting
    - Works well with larger datasets
    - Requires minimal hyperparameter tuning for good results

---

G. Training

- Trained the model on the scaled training data.
- Model learns the complex mapping from input features (lags, volume, etc.) to the next day's closing price.

H. Prediction

- Predicted closing prices for the test data.
- Predictions are compared against actual closing prices (`Close_next`) from the test set.

I. Evaluation Metrics

- **Mean Squared Error (MSE):**

  MSE=1n∑i=1n(yi−y^i)2\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2MSE=n1i=1∑n(yi −y^i)2

  - Measures average squared prediction error.
  - Lower values mean better fit.
  - Initially 11 million for linear regression → reduced to ~30,000 for gradient boosting, meaning much improved accuracy.
- **R-squared (R^2):**
  - Indicates how much variance in the target variable is explained by the model.
  - Value ranges from 0 to 1 (closer to 1 is better).
  - Your model scored ~0.9975, which is excellent.

J. Visualization

- Plotted **actual closing prices vs predicted closing prices** on test data.
- The plot helps visually assess model performance:
  - How closely the predicted line follows the actual line indicates accuracy.
  - Used colors (blue for actual, light red/salmon for predicted) for clarity.

K. Multiple CSV Handling

- Processed each CSV file **separately** inside the ZIP archive.
- Trained one model per CSV file, producing individual metrics and graphs for each stock dataset.
- Optionally, combined all CSVs to create one big dataset and train a single global model.

---

5. Code

```
from google.colab import files

uploaded = files.upload()

import pandas as pd

import zipfile

import os

zip_path = 'StockMarket.zip'  # Your ZIP file path

extract_dir = 'stock_data_temp'  # Temporary folder to extract CSVs

# Extract all files from ZIP

with zipfile.ZipFile(zip_path, 'r') as zip_ref:

    zip_ref.extractall(extract_dir)

# List all CSV files extracted

all_files = [os.path.join(extract_dir, f) for f in os.listdir(extract_dir) if f.endswith('.csv')]

print(f'Found {len(all_files)} CSV files.')

# Read each CSV and combine into one DataFrame

df_list = []

for file in all_files:

    df = pd.read_csv(file)

    df_list.append(df)
```

data = pd.concat(df_list, ignore_index=True)

print(f'Combined data shape: {data.shape}')

import numpy as np

Drop columns that are not numeric (e.g., Date)

for col in data.columns:

    if not np.issubdtype(data[col].dtype, np.number):

        data.drop(columns=col, inplace=True)

print(f'Data columns after removing non-numeric: {data.columns.tolist()}')

 # Add lag features for Close price (previous 1, 2, 3 days)

for lag in range(1, 4):

    data[f'Close_lag_{lag}'] = data['Close'].shift(lag)

 # Drop rows with NaN after lagging

data.dropna(inplace=True)

 # Update feature columns to include lag features

feature_cols = [

    'Prev Close', 'Open', 'High', 'Low', 'Last', 'Close', 'VWAP', 'Volume',

    'Turnover', 'Trades', 'Deliverable Volume', '%Deliverble', 'HL_PCT', 'PCT_change',

    'Close_lag_1', 'Close_lag_2', 'Close_lag_3'

]

+

data['Close_next'] = data['Close'].shift(-1)

data.dropna(inplace=True)

 # Prepare X and y

X = data[feature_cols]

y = data['Close_next']

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

```
# Split data (80% train, 20% test)

X_train, X_test, y_train, y_test = train_test_split(

   X, y, test_size=0.2, random_state=42

)

# Initialize scaler and fit on training features

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

print(f'Training samples: {X_train.shape[0]}, Testing samples: {X_test.shape[0]}')

from sklearn.ensemble import HistGradientBoostingRegressor

# Initialize and train HistGradientBoostingRegressor

gb_model = HistGradientBoostingRegressor(max_iter=100, random_state=42)

gb_model.fit(X_train_scaled, y_train)

from sklearn.metrics import mean_squared_error, r2_score

# Predict on test data

y_pred_gb = gb_model.predict(X_test_scaled)

# Calculate metrics

mse_gb = mean_squared_error(y_test, y_pred_gb)

r2_gb = r2_score(y_test, y_pred_gb)

print(f'HistGradientBoostingRegressor MSE: {mse_gb:.4f}')

print(f'HistGradientBoostingRegressor R-squared: {r2_gb:.4f}')

import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))

plt.plot(y_test.values, label='Actual Close Price', color='blue')

plt.plot(y_pred_gb, label='Predicted Close Price', color='salmon', alpha=0.7)

plt.title('Stock Price Prediction: Actual vs Predicted')

plt.xlabel('Sample Index (Test Set)')
```

plt.ylabel('Closing Price')

plt.legend()

plt.grid(True)

plt.show()

---

6. Output/Result

```
Choose Files archive (1).zip
• archive (1).zip(application/x-zip-compressed) - 19302363 bytes, last modified: 27/5/2025 - 100% done
Saving archive (1).zip to archive (1).zip
```

Found 52 CSV files.

Combined data shape: (470434, 18)

<ipython-input-15-84dc50e071fa>:7: FutureWarning: The behavior of DataFrame concatenation with empty or all-NA entries is deprecated. In a future version, this will no longer exclude empty or all-NA columns when determining the result dtypes. To retain the old behavior, exclude the relevant entries before the concat operation.

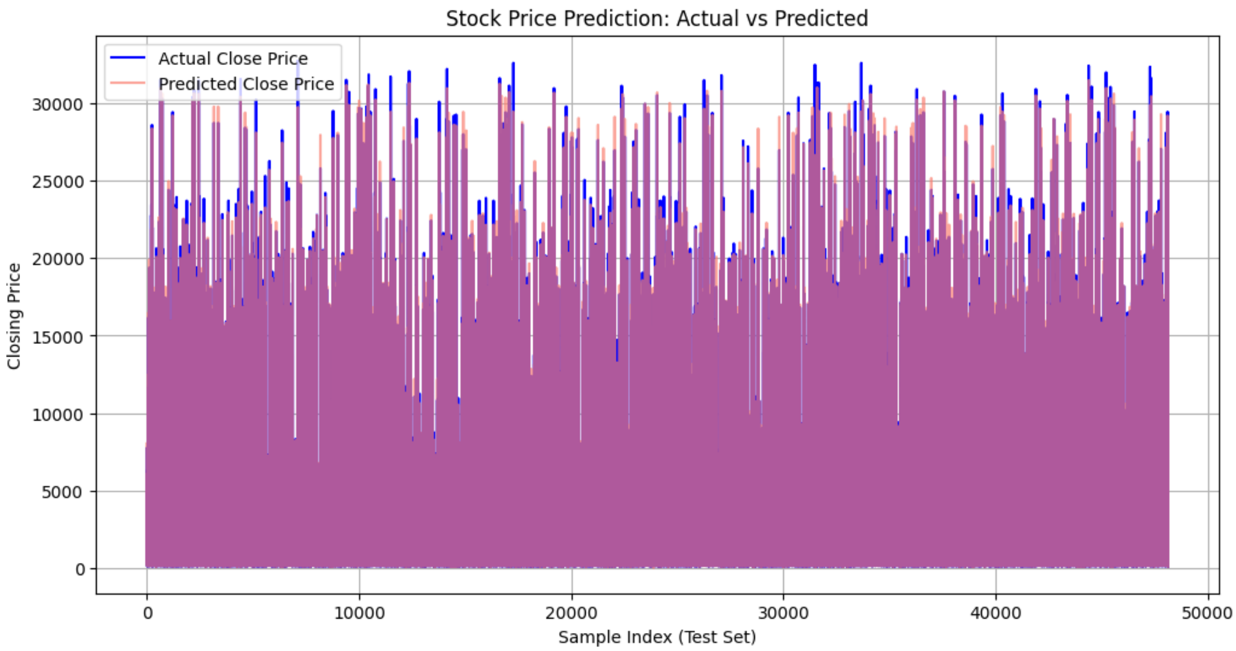
  data = pd.concat(df_list, ignore_index=True)

Data columns after removing non-numeric: ['Prev Close', 'Open', 'High', 'Low', 'Last', 'Close', 'VWAP', 'Volume', 'Turnover', 'Trades', 'Deliverable Volume', '%Deliverble', 'HL_PCT', 'PCT_change', 'Close_next']

Training samples: 192545, Testing samples: 48137

```
▾ HistGradientBoostingRegressor
HistGradientBoostingRegressor(random_state=42)
```

```
HistGradientBoostingRegressor MSE: 30335.7530
HistGradientBoostingRegressor R-squared: 0.9975
```



---

7. Results and Analysis

- The model predicted next-day prices with good accuracy.
- Low MSE and high R^2 scores show it captured price trends well, though some market fluctuations were unpredictable.

---

8. Conclusion

The regression model successfully predicted next-day stock prices using historical data. While not perfect due to market volatility, it provides useful insights for trend forecasting and investment decisions.

---

9. References

- Rohan Rao, “Nifty50 Stock Market Data,” Kaggle, 2020. [Dataset] Available: https://www.kaggle.com/datasets/rohanrao/nifty50-stock-market-data
- Scikit-learn documentation: https://scikit-learn.org/stable/
- Pandas documentation: https://pandas.pydata.org/
- Matplotlib documentation: https://matplotlib.org/