

**Assessment Report**

on

**“Stock Price Prediction”**

submitted as partial fulfillment for the award of

**BACHELOR OF TECHNOLOGY**

**DEGREE**

SESSION 2024-25

in

**CSE(AIML)**

By

Name : Anushika Sadhwani Roll\_No :202401100400042

Name : Akeel Ahmad Roll\_No :202401100400020

Name : Abhilash Raj Roll\_No :202401100400007

Name : Akash Bharti Roll\_No :202401100400019

Name : Ayush Chaudhary Roll\_No :202401100400063

Section: A

**Under the supervision of**

“Bikki Kumar”

**KIET Group of Institutions, Ghaziabad**

**May, 2025**

**1. Introduction**

Stock price prediction has long been a major area of interest in financial markets, aiming to help investors make informed decisions. By leveraging historical stock data and machine learning models, we can estimate future prices and detect patterns or trends.

**2. Problem Statement**

The goal is to develop a regression-based model that predicts the next-day stock closing price based on historical data. We will analyze trends and assess the model’s predictive performance.

**3. Objectives**

* Build a predictive model using historical stock data.
* Visualize stock trends over time.
* Evaluate the accuracy of the regression model.
* Provide insights into the model’s strengths and weaknesses.

**4. Methodology**

**Data Collection: Use NIFTY 50 historical data (Kaggle or Yahoo Finance).**

**Data Preprocessing: Handle missing values, generate lag features.**

**Modeling: Apply Linear Regression for price prediction.**

**Evaluation: Use Mean Squared Error (MSE) and R-squared to assess performance.**

**Visualization: Plot trends and predictions for better analysis.**

.  
**5. Data Preprocessing**

Removed rows with missing values (due to lag features).

Created a new feature Prev\_Close (previous day's close).

Split data chronologically into training and testing sets (80/20 split).

**6. Model Implementation**

We used a simple Linear Regression model from Scikit-learn:

Input: Previous day’s closing price.

Output: Next day’s closing price.

Model was trained on historical data and evaluated on unseen (test) data.

**7. Evaluation Metrics**

Mean Squared Error (MSE): Measures the average squared difference between actual and predicted prices.

R-squared (R²): Indicates how well the model explains variance in the data.

**8. Results and Analysis**

:- MSE: (example) ~150.45

:- R²: (example) 0.87

The model shows a reasonable ability to capture price trends using just the previous day’s close, but more sophisticated models (like LSTM, ARIMA) could improve accuracy.

Visualization of actual vs predicted prices shows the general trend is followed, though exact prices may deviate due to market volatility.

**9. Conclusion**

This project demonstrated the feasibility of using regression models for next-day stock price prediction. While the Linear Regression model provides a baseline, future work could explore more advanced models, include additional features (like volume, indicators), and apply hyperparameter tuning.

**10. References**

* Kaggle Dataset: <https://www.kaggle.com/datasets/rohanrao/nifty50-stock-market-data>
* Scikit-learn Documentation: [https://scikit-learn.org](https://scikit-learn.org/)
* Yahoo Finance API (yfinance): <https://pypi.org/project/yfinance>
* Seaborn & Matplotlib for Visualization

**CODE**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

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

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

df1 = pd.read\_csv("/content/ADANIPORTS.csv")

df2 = pd.read\_csv("/content/ONGC.csv")

print("DF1 Columns:", df1.columns)

print("DF2 Columns:", df2.columns)

def preprocess(df):

df.columns = df.columns.str.strip().str.lower()

if 'date' not in df.columns or 'close' not in df.columns:

raise ValueError("Missing 'date' or 'close' column. Found columns: " + str(df.columns))

df['date'] = pd.to\_datetime(df['date'])

df = df.sort\_values('date')

df.set\_index('date', inplace=True)

df = df[['close']]

df['next\_close'] = df['close'].shift(-1)

return df.dropna()

# --- Apply Preprocessing ---

df1 = preprocess(df1)

df2 = preprocess(df2)

# --- Combine Datasets ---

data = pd.concat([df1, df2]).dropna()

# --- Feature Engineering (only 1 lag feature to avoid overfitting) ---

data['lag1\_close'] = data['close'].shift(1)

data = data.dropna()

X = data[['lag1\_close']]

y = data['next\_close']

# --- Train-Test Split (no shuffling to keep time sequence) ---

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, shuffle=False)

# --- Model Training ---

model = LinearRegression()

model.fit(X\_train, y\_train)

# --- Predictions ---

y\_pred = model.predict(X\_test)

# --- Evaluation Metrics ---

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

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

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

print("Model Evaluation Metrics:")

print(f"R² Score: {r2:.2f}")

print(f"MAE: {mae:.2f}")

print(f"RMSE: {rmse:.2f}")

# --- Visualization ---

plt.figure(figsize=(14, 6))

plt.plot(y\_test.index, y\_test.values, label="Actual Price", color='blue', marker='o', alpha=0.6)

plt.plot(y\_test.index, y\_pred, label="Predicted Price", color='red', marker='x', alpha=0.6)

# Add error lines

for i in range(len(y\_test)):

plt.plot([y\_test.index[i], y\_test.index[i]],

[y\_test.values[i], y\_pred[i]],

color='gray', linestyle='--', linewidth=0.5, alpha=0.3)

plt.title("Next-Day Stock Price Prediction (Avoiding Overfitting)")

plt.xlabel("Date")

plt.ylabel("Stock Price")

plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()

**OUTPUT/RESULT:**



