LABORATORY REPORT

Application Development Lab (CS33002)

B.Tech Program in CSE

Submitted By

Name:-Madhurima Ghosh

Roll No: 2305941

Kalinga Institute of Industrial Technology (Deemed to be University) Bhubaneswar, India

Spring 2025-2026

Experiment Number	**6**
Experiment Title	**Regression Analysis for Stock Prediction**
Date of Experiment	**21-01-2026**
Date of Submission	**28-01-2026**

1. Objective:- To perform stock price prediction using Linear Regression and LSTM models.

2. Code:-

a. train_linear.py

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import MinMaxScaler
import joblib
import os

df = pd.read_csv("data/stock.csv")

df['Close'] = pd.to_numeric(df['Close'], errors='coerce')
df = df.dropna()

data = df[['Close']].values

scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data)

X = []
y = []

for i in range(60, len(scaled_data)):
    X.append(scaled_data[i-60:i])
    y.append(scaled_data[i])

X, y = np.array(X), np.array(y)
X = X.reshape(X.shape[0], -1)

model = LinearRegression()
model.fit(X, y)

os.makedirs("models", exist_ok=True)
joblib.dump(model, "models/linear_model.pkl")
joblib.dump(scaler, "models/linear_scaler.pkl")

print("✅ Linear Regression model trained & saved")
```

b. train_lstm.py

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
import os

df = pd.read_csv("data/stock.csv")
df['Close'] = pd.to_numeric(df['Close'], errors='coerce')
df = df.dropna()

data = df[['Close']].values

scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data)

X, y = [], []

for i in range(60, len(scaled_data)):
    X.append(scaled_data[i-60:i])
    y.append(scaled_data[i])

X, y = np.array(X), np.array(y)

model = Sequential([
    LSTM(50, return_sequences=True, input_shape=(X.shape[1], 1)),
    LSTM(50),
    Dense(1)])

model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X, y, epochs=10, batch_size=32)

os.makedirs("models", exist_ok=True)
model.save("models/lstm_model.h5")

print("✅ LSTM model trained & saved")
```

c. app.py

```
from flask import Flask, render_template, request
import numpy as np
import pandas as pd
import joblib
from tensorflow.keras.models import load_model
from sklearn.preprocessing import MinMaxScaler

app = Flask(__name__)

df = pd.read_csv("data/stock.csv")
df["Close"] = pd.to_numeric(df["Close"], errors="coerce")
df.dropna(inplace=True)

prices = df["Close"].values.reshape(-1, 1)

scaler = MinMaxScaler()
scaler.fit(prices)

linear_model = joblib.load("models/linear_model.pkl")
lstm_model = load_model("models/lstm_model.h5")

SEQ_LEN = 60
@app.route("/", methods=["GET"])
def index():
```

```
return render_template("index.html")

@app.route("/predict", methods=["POST"])
def predict():
    try:
        prices_input = request.form["prices"]
        model_type = request.form["model"]

        prices_list = np.array(
            [float(x.strip()) for x in prices_input.split(",")]
        ).reshape(-1, 1)

        if len(prices_list) < 60:
            return render_template(
                "index.html",
                error="Please enter at least 60 closing prices."
            )

        scaled_input = scaler.transform(prices_list)
        last_60 = scaled_input[-SEQ_LEN:]

        if model_type == "linear":
            X = last_60.reshape(1, -1)
            pred_scaled = linear_model.predict(X)
            prediction = scaler.inverse_transform(
                pred_scaled.reshape(-1, 1)
            )[0][0]
        else:
            X = last_60.reshape(1, SEQ_LEN, 1)
            pred_scaled = lstm_model.predict(X)
            prediction = scaler.inverse_transform(pred_scaled)[0][0]

        return render_template(
            "index.html",
            prediction=round(prediction, 2),
            model_used=model_type.upper()
        )

    except Exception as e:
        return render_template(
            "index.html",
            error=f"Error: {e}"
        )
if __name__ == "__main__":
    app.run(debug=True)
```

d. templates/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Stock Price Prediction</title>

    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f6f8;
            text-align: center;
            padding: 40px;
        }

        textarea {
            width: 60%;
            height: 100px;
            padding: 10px;
        }

        select, button {
            padding: 10px;
            margin-top: 15px;
        }

        button {
            background-color: #3498db;
            color: white;
            border: none;
            cursor: pointer;
        }

        button:hover {
            background-color: #2980b9;
        }

        .error {
            color: red;
            margin-top: 20px;
        }
    </style>
</head>

<body>

    <h1>Stock Price Prediction</h1>

    <form method="post" action="/predict">
        <p>Enter last 60 closing prices (comma separated):</p>

        <textarea name="prices" required></textarea>
        <br>

        <select name="model">
            <option value="linear">Linear Regression</option>
            <option value="lstm">LSTM</option>
        </select>
        <br>

        <button type="submit">Predict</button>
    </form>

    {% if prediction %}
        <h2>Predicted Next Closing Price ({{ model_used }}): {{ prediction }}</h2>
    {% endif %}

    {% if error %}
        <div class="error">{{ error }}</div>
    {% endif %}

</body>
```

```
</html>
```

e. static/style.css

```
body {
   font-family: Arial;
   text-align: center;
   margin-top: 50px;
}

textarea {
   width: 300px;
}
```

f. download_data.py

```
import yfinance as yf
import pandas as pd
import os
df = yf.download("AAPL", start="2020-01-01", end="2024-01-01")
os.makedirs("data", exist_ok=True)
df.to_csv("data/stock.csv")
print("✅ Stock data downloaded and saved as data/stock.csv")
print(df.head())
```

g. evaluate_models.py

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from tensorflow.keras.models import load_model

data = pd.read_csv("data/stock.csv")
prices = data["Close"].values.reshape(-1, 1)

scaler = MinMaxScaler()
scaled_prices = scaler.fit_transform(prices)

X_lr = np.arange(len(prices)).reshape(-1, 1)
y_lr = prices

lr_model = LinearRegression()
lr_model.fit(X_lr, y_lr)
lr_pred = lr_model.predict(X_lr)
```

```
lr_mse = mean_squared_error(y_lr, lr_pred)
lr_rmse = np.sqrt(lr_mse)

SEQ_LEN = 60

def create_sequences(data, seq_len):
    X, y = [], []
    for i in range(seq_len, len(data)):
        X.append(data[i-seq_len:i])
        y.append(data[i])
    return np.array(X), np.array(y)

X_lstm, y_lstm = create_sequences(scaled_prices, SEQ_LEN)

lstm_model = load_model("models/lstm_model.h5")
lstm_pred = lstm_model.predict(X_lstm)
lstm_pred = scaler.inverse_transform(lstm_pred)
y_lstm_actual = scaler.inverse_transform(y_lstm)

lstm_mse = mean_squared_error(y_lstm_actual, lstm_pred)
lstm_rmse = np.sqrt(lstm_mse)

print("MODEL PERFORMANCE COMPARISON")
print("-----------------------------")
print(f"Linear Regression MSE  : {lr_mse:.4f}")
print(f"Linear Regression RMSE : {lr_rmse:.4f}")
print()
print(f"LSTM MSE               : {lstm_mse:.4f}")
print(f"LSTM RMSE              : {lstm_rmse:.4f}")
```

h. plot_comparison.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
from tensorflow.keras.models import load_model
data = pd.read_csv("data/stock.csv")
prices = data["Close"].values.reshape(-1, 1)

scaler = MinMaxScaler()
scaled_prices = scaler.fit_transform(prices)

X = []
```

```
y = []
for i in range(60, len(scaled_prices)):
    X.append(scaled_prices[i-60:i, 0])
    y.append(scaled_prices[i, 0])

X, y = np.array(X), np.array(y)

split = int(0.8 * len(X))
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]

lr = LinearRegression()
lr.fit(X_train, y_train)
lr_pred = lr.predict(X_test)

X_test_lstm = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))
lstm_model = load_model("models/lstm_model.h5")
lstm_pred = lstm_model.predict(X_test_lstm)

actual = scaler.inverse_transform(y_test.reshape(-1, 1))
lr_pred = scaler.inverse_transform(lr_pred.reshape(-1, 1))
lstm_pred = scaler.inverse_transform(lstm_pred)

plt.figure()
plt.plot(actual, label="Actual Price")
plt.plot(lr_pred, label="Linear Regression")
plt.plot(lstm_pred, label="LSTM")
plt.title("Stock Price Prediction Comparison")
plt.xlabel("Days")
plt.ylabel("Price")
plt.legend()
plt.savefig("static/comparison.png")
plt.show()
plt.close()
```

i. requirements.txt

```
flask
pandas
Numpy
scikit-learn
tensorflow
matplotlib
```

j. **Folder data: stock.csv**

k. **Folder models: linear_model.pkl, linear_scaler.pkl, lstm_model.h5**

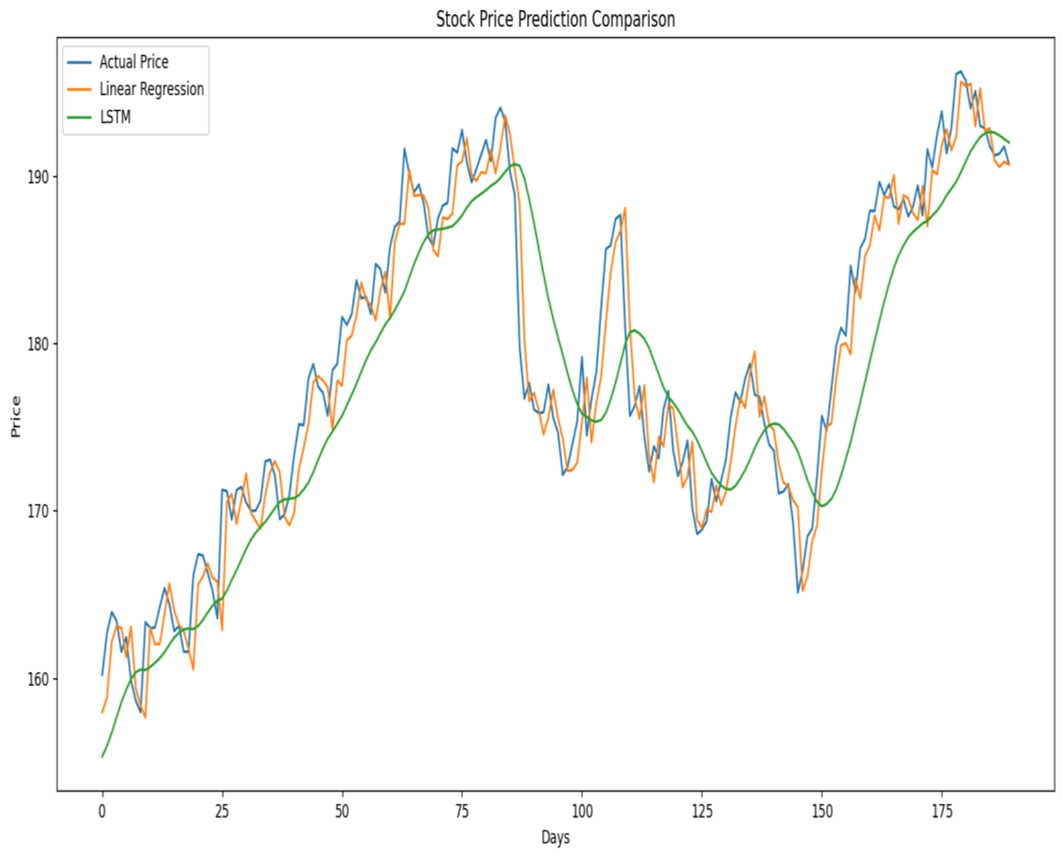
l. **Folder static: comparison.png**

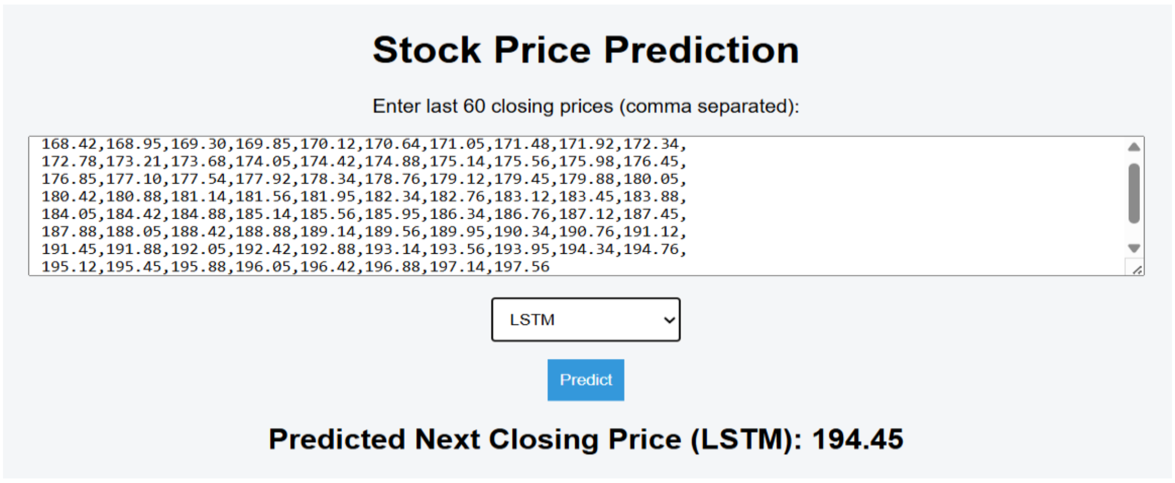
3. Results/Output:-

```
MODEL PERFORMANCE COMPARISON
-----------------------------
Linear Regression MSE  : 257.0627
Linear Regression RMSE : 16.0332

LSTM MSE               : 26.0991
LSTM RMSE              : 5.1087
```

```python
plt.figure()
plt.plot(actual, label="Actual Price")
plt.plot(lr_pred, label="Linear Regression")
plt.plot(lstm_pred, label="LSTM")
plt.title("Stock Price Prediction Comparison")
plt.xlabel("Days")
plt.ylabel("Price")
plt.legend()
plt.savefig("static/comparison.png")
plt.show()
plt.close()
```





Stock Price Prediction

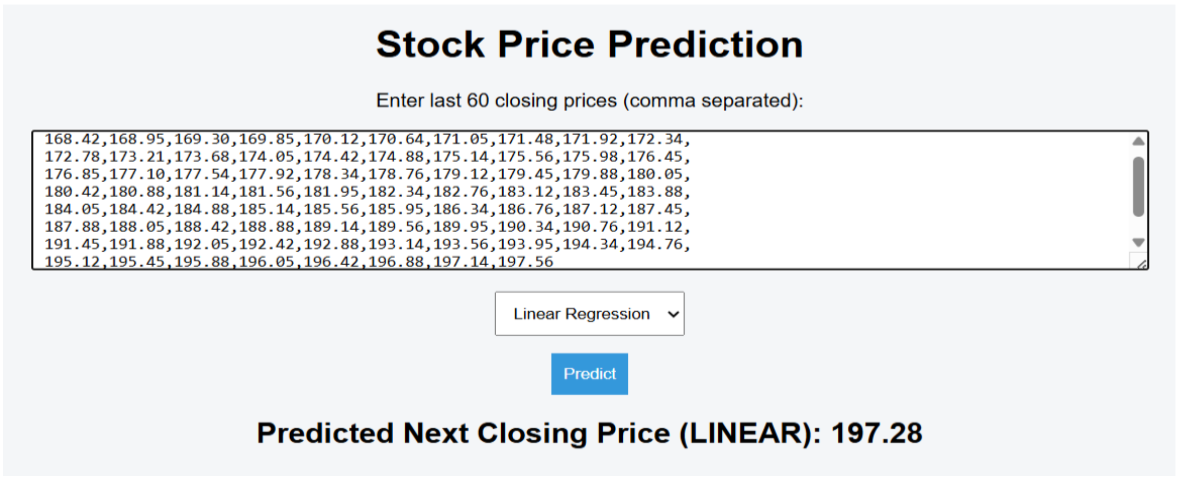
Enter last 60 closing prices (comma separated):

```
168.42,168.95,169.30,169.85,170.12,170.64,171.05,171.48,171.92,172.34,
172.78,173.21,173.68,174.05,174.42,174.88,175.14,175.56,175.98,176.45,
176.85,177.10,177.54,177.92,178.34,178.76,179.12,179.45,179.88,180.05,
180.42,180.88,181.14,181.56,181.95,182.34,182.76,183.12,183.45,183.88,
184.05,184.42,184.88,185.14,185.56,185.95,186.34,186.76,187.12,187.45,
187.88,188.05,188.42,188.88,189.14,189.56,189.95,190.34,190.76,191.12,
191.45,191.88,192.05,192.42,192.88,193.14,193.56,193.95,194.34,194.76,
195.12,195.45,195.88,196.05,196.42,196.88,197.14,197.56
```

LSTM

Predict

Predicted Next Closing Price (LSTM): 194.45



Stock Price Prediction

Enter last 60 closing prices (comma separated):

```
168.42,168.95,169.30,169.85,170.12,170.64,171.05,171.48,171.92,172.34,
172.78,173.21,173.68,174.05,174.42,174.88,175.14,175.56,175.98,176.45,
176.85,177.10,177.54,177.92,178.34,178.76,179.12,179.45,179.88,180.05,
180.42,180.88,181.14,181.56,181.95,182.34,182.76,183.12,183.45,183.88,
184.05,184.42,184.88,185.14,185.56,185.95,186.34,186.76,187.12,187.45,
187.88,188.05,188.42,188.88,189.14,189.56,189.95,190.34,190.76,191.12,
191.45,191.88,192.05,192.42,192.88,193.14,193.56,193.95,194.34,194.76,
195.12,195.45,195.88,196.05,196.42,196.88,197.14,197.56
```

Linear Regression

Predict

Predicted Next Closing Price (LINEAR): 197.28

Roll number: 2305941

Signature of the Student

(Name of the Student)

Signature of the Lab Coordinator

(Name of the Coordinator)