LABORATORY REPORT

Application Development Lab
(CS33002)

B.Tech Program in ECSc

Submitted By

Name:-Pruthibiraj Nayak

Roll No: 2230183

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

Spring 2024-2025

Table of Content

Experiment Number	3
Experiment Title	Regression Analysis for Stock Prediction
Date of Experiment	22.01.2025
Date of Submission	27.01.2025

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

2. Procedure:- (Steps Followed)
1. Collect historical stock price data.
2. Preprocess the data for analysis (missing data, scaling, splitting into train/test).
3. Implement Linear Regression to predict future stock prices.
4. Design and train an LSTM model for time-series prediction.
5. Compare the accuracy of both models.
6. Create a Flask backend for model predictions.
7. Build a frontend to visualize predictions using charts and graphs.

Code:-

GOOGLE COLAB (Training & Testing)

```
import numpy as np   #Linear algera Library
import pandas as pd
import matplotlib.pyplot as plt  #to plot graphs
import seaborn as sns  #to plot graphs
from sklearn.linear_model import LinearRegression   #for linear regression
model
sns.set()  #setting seaborn as default
import math

import warnings
warnings.filterwarnings('ignore')

data=pd.read_csv("/content/NSE-TATAGLOBAL.csv")   #reads the input data
data.head()   #displays the first five rows

data.info()

data.describe(include ='all')   #parameter include=all will display NaN
values as well

data.isnull().sum() # No null values
data.head()
sns.pairplot(data)
plt.show()
```

```
#  we use open,high,low,last to predict  close price
x=data[['High','Low','Last','Open','Total Trade Quantity','Turnover
(Lacs)']].values   #input
y=data[['Close']].values   #output

from sklearn.model_selection import train_test_split
#split to train and test data
x_train, x_test, y_train, y_test = train_test_split(x,y,
test_size=0.2,random_state=0)

#using linear regression
lm=LinearRegression()
lm.fit(x_train,y_train)

#values from 0 to 1
#0 model explain None of the variability
#1 model explain Entire of the variability
lm.score(x_train,y_train)

#predict the output(predictions) using the test data
predictions = lm.predict(x_test)

from sklearn.metrics import r2_score
r2_score(y_test, predictions)

#load actual and predecited values side by side
dframe=pd.DataFrame({'actual':y_test.flatten(),'Predicted':predictions.flatte
n()})
#flatten toget single axis of data (1 dimension only)
dframe.head(15)

graph =dframe.head(10)
graph.plot(kind='bar')
plt.title('Actual vs Predicted')
plt.ylabel('Closing price')

#using scatter plot compare the actual and predicted data
fig = plt.figure()
plt.scatter(y_test,predictions)
plt.title('Actual versus Prediction ')
plt.xlabel('Actual', fontsize=20)
plt.ylabel('Predicted', fontsize=20)

sns.regplot(x=y_test, y=predictions)
plt.title('Actual versus Prediction')
plt.xlabel('Actual', fontsize=20)
plt.ylabel('Predicted', fontsize=20)
plt.show()
```

```
import math
from sklearn import metrics
#metrics to find accuracy of continous variables
print('Mean Abs value:' ,metrics.mean_absolute_error(y_test,predictions))
print('Mean squared value:',metrics.mean_squared_error(y_test,predictions))
print('root mean squared error
value:',math.sqrt(metrics.mean_squared_error(y_test,predictions)))

from sklearn.preprocessing import MinMaxScaler
# Scale the data between 0 and 1
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(data[['Close']].values)

# Prepare the data for time-series prediction
def create_dataset(dataset, time_step=1):
    X, Y = [], []
    for i in range(len(dataset) - time_step - 1):
        X.append(dataset[i:(i + time_step), 0])
        Y.append(dataset[i + time_step, 0])
    return np.array(X), np.array(Y)

# Define time step
time_step = 60  # 60 days
X, Y = create_dataset(scaled_data, time_step)

# Reshape X to fit LSTM input (samples, time_steps, features)
X = X.reshape(X.shape[0], X.shape[1], 1)

# Split into train and test sets
train_size = int(len(X) * 0.8)
test_size = len(X) - train_size
X_train, X_test = X[:train_size], X[train_size:]
Y_train, Y_test = Y[:train_size], Y[train_size:]

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
# Build the LSTM model
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(time_step, 1)))
model.add(LSTM(units=50, return_sequences=False))
model.add(Dense(units=25))
model.add(Dense(units=1))

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
```

```
model.fit(X_train, Y_train, batch_size=32, epochs=50)

# Predict using the LSTM model
predicted_prices = model.predict(X_test)

# Reverse scaling to get actual prices
predicted_prices = scaler.inverse_transform(predicted_prices.reshape(-1, 1))
Y_test_actual = scaler.inverse_transform(Y_test.reshape(-1, 1))

# Plot actual vs predicted
plt.figure(figsize=(10, 6))
plt.plot(Y_test_actual, label='Actual Prices')
plt.plot(predicted_prices, label='Predicted Prices')
plt.title('LSTM Model: Actual vs Predicted Prices')
plt.xlabel('Time')
plt.ylabel('Stock Price')
plt.legend()
plt.show()

# Calculate metrics for LSTM
lstm_mae = metrics.mean_absolute_error(Y_test_actual, predicted_prices)
lstm_mse = metrics.mean_squared_error(Y_test_actual, predicted_prices)
lstm_rmse = math.sqrt(lstm_mse)

print("LSTM Model Performance:")
print(f"MAE: {lstm_mae}")
print(f"MSE: {lstm_mse}")
print(f"RMSE: {lstm_rmse}")

# Compare with Linear Regression
print("\nLinear Regression Performance:")
print('Mean Abs value:', metrics.mean_absolute_error(y_test, predictions))
print('Mean squared value:', metrics.mean_squared_error(y_test, predictions))
print('Root mean squared error value:',
math.sqrt(metrics.mean_squared_error(y_test, predictions)))

model.save('lstm_model.keras')
import joblib
joblib.dump(lm, 'linear_model.pkl')
```

FLASK CODE

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

# Initialize Flask app
```

```
app = Flask(__name__)

# Load models
linear_model = joblib.load('linear_model.pkl')
lstm_model = load_model('lstm_model.keras')

# Initialize scaler
scaler = MinMaxScaler(feature_range=(0, 1))

@app.route('/')
def home():
    return render_template('index.html', linear_result='', lstm_result='')

@app.route('/predict-linear', methods=['POST'])
def predict_linear():
    features = request.form['features']
    features = np.array([float(x) for x in features.split(',')]).reshape(1, -
1)
    prediction = linear_model.predict(features)[0][0]
    return render_template('index.html', linear_result=f'Predicted Price:
{prediction}', lstm_result='')

@app.route('/predict-lstm', methods=['POST'])
def predict_lstm():
    prices = request.form['prices']
    prices = np.array([float(x) for x in prices.split(',')]).reshape(-1, 1)
    scaled_prices = scaler.fit_transform(prices)
    scaled_prices = scaled_prices.reshape(1, len(scaled_prices), 1)
    prediction = lstm_model.predict(scaled_prices)
    predicted_price = scaler.inverse_transform(prediction)[0][0]
    return render_template('index.html', linear_result='',
lstm_result=f'Predicted Price: {predicted_price}')

if __name__ == '__main__':
    app.run(debug=True)
```

HTML CODE

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Stock Price Prediction</title>
    <link rel="stylesheet" href="/static/style.css">
</head>
<body>
    <div class="container">
        <header>
```

```
            <h1>  Stock Price Prediction</h1>
        </header>

        <main>
            <section class="form-section">
                <h2>Linear Regression Prediction</h2>
                <form action="/predict-linear" method="POST">
                    <label for="features">Enter Features (comma-
separated):</label>
                    <input type="text" id="features" name="features"
placeholder="e.g., 100, 98, 102, 101, 50000, 500">
                    <button type="submit">Predict</button>
                </form>
                <p class="result">{{ linear_result }}</p>
            </section>
            <section class="form-section">
                <h2>LSTM Prediction</h2>
                <form action="/predict-lstm" method="POST">
                    <label for="prices">Enter Recent Prices (comma-
separated):</label>
                    <input type="text" id="prices" name="prices"
placeholder="e.g., 100, 101, 102, 103, 104, 105">
                    <button type="submit">Predict</button>
                </form>
                <p class="result">{{ lstm_result }}</p>
            </section>
        </main>
        <footer>
            <p>© 2025 Stock Prediction App</p>
        </footer>
    </div>
</body>
</html>
```

CSS SHEET

```
/* General Reset */
body {
    margin: 0;
    font-family: Arial, sans-serif;
    background-color: #f4f4f9;
    color: #333;
}

/* Container */
.container {
    max-width: 800px;
    margin: 20px auto;
    padding: 20px;
```

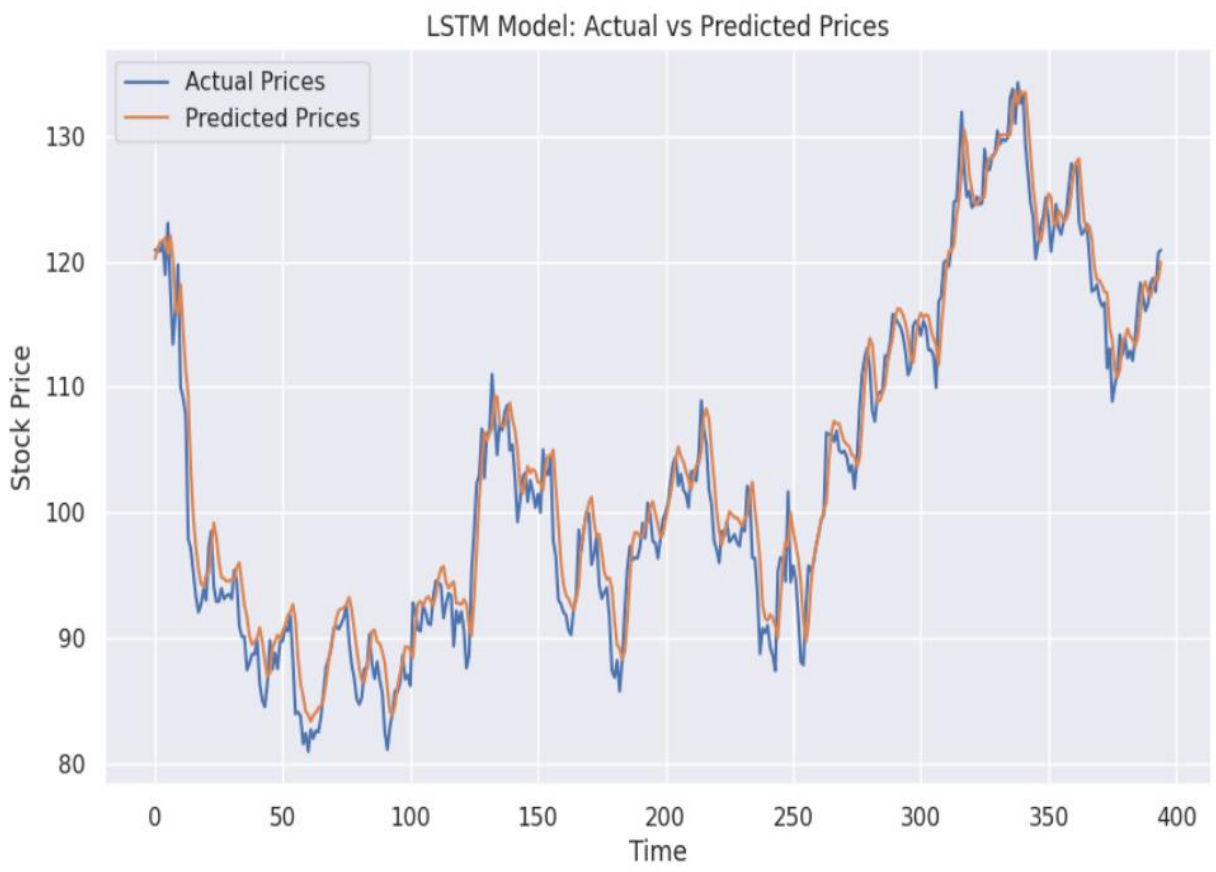
```
    background: #ffffff;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    border-radius: 10px;
}
/* Header */
header {
    text-align: center;
    margin-bottom: 20px;
}
header h1 {
    color: #0078d7;
    font-size: 2.5rem;
}
/* Form Section */
.form-section {
    margin-bottom: 30px;
}
.form-section h2 {
    color: #0078d7;
    font-size: 1.5rem;
    margin-bottom: 10px;
}
label {
    display: block;
    font-weight: bold;
    margin-bottom: 5px;
}
input[type="text"] {
    width: 100%;
    padding: 10px;
    margin-bottom: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
}
button {
    background-color: #0078d7;
    color: white;
    border: none;
    padding: 10px 20px;
    border-radius: 5px;
    cursor: pointer;
    font-size: 1rem;
}
button:hover {
    background-color: #005bb5;
}
.result {
    font-size: 1.2rem;
    margin-top: 10px;
```

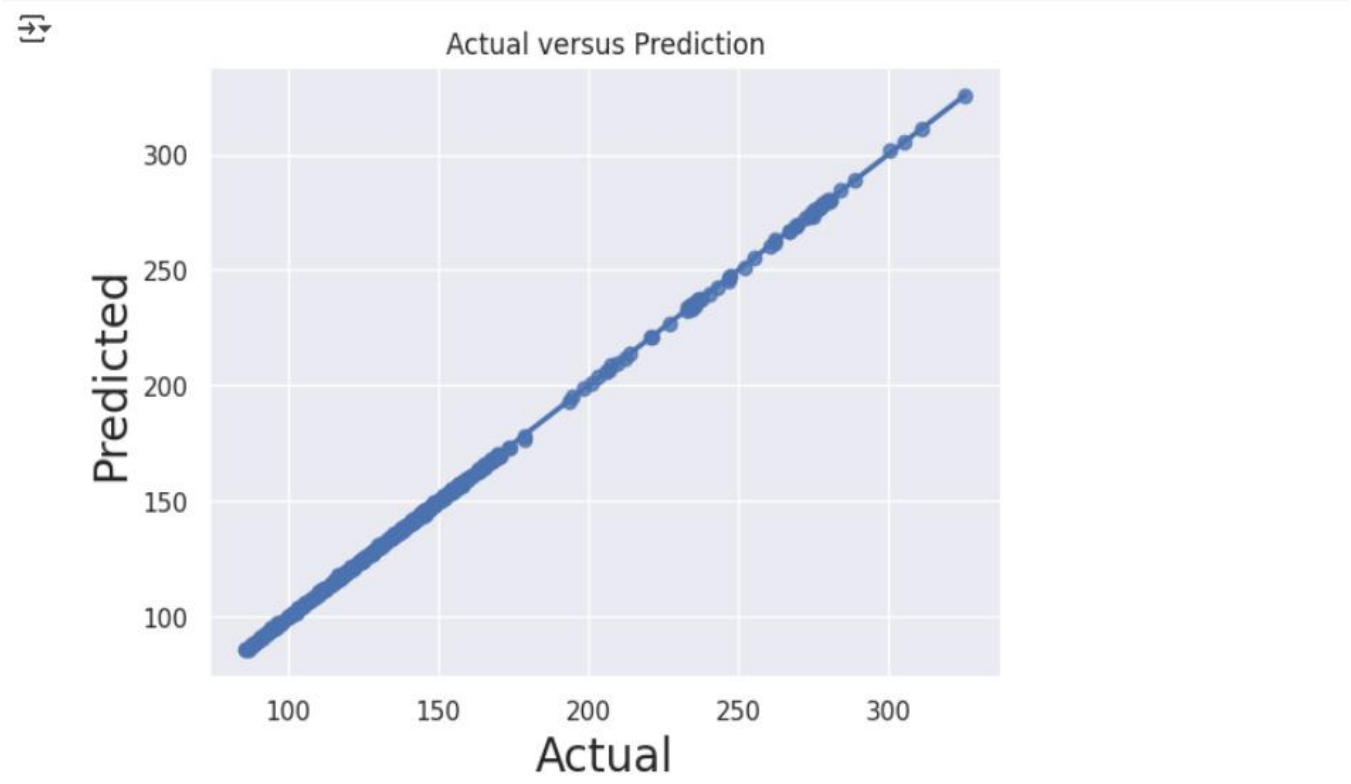
```
    color: #333;
}
/* Footer */
footer {
    text-align: center;
    margin-top: 20px;
    font-size: 0.9rem;
    color: #666;
}
```

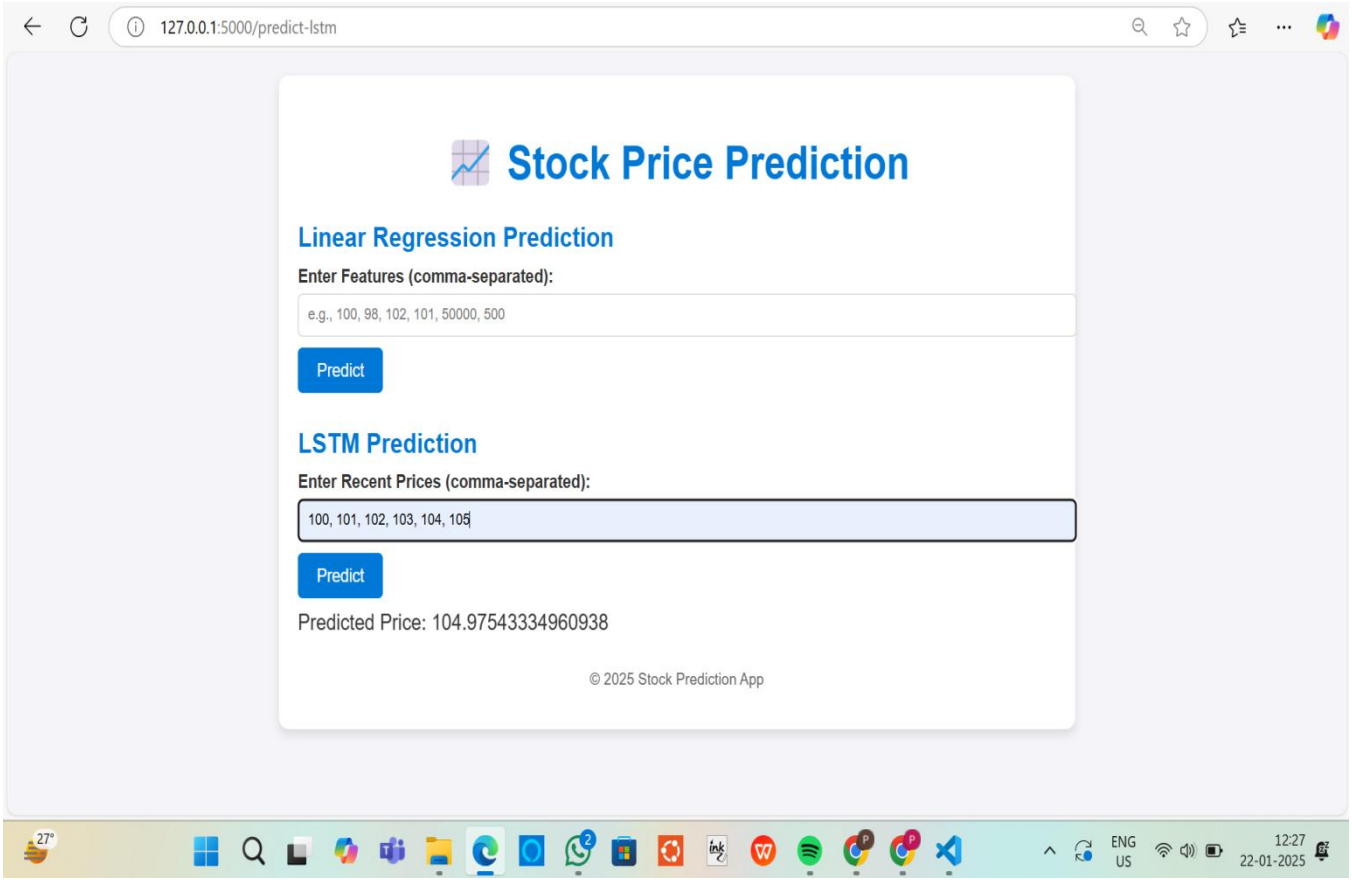
3. Results/Output:- Entire Screen Shot including Date & Time

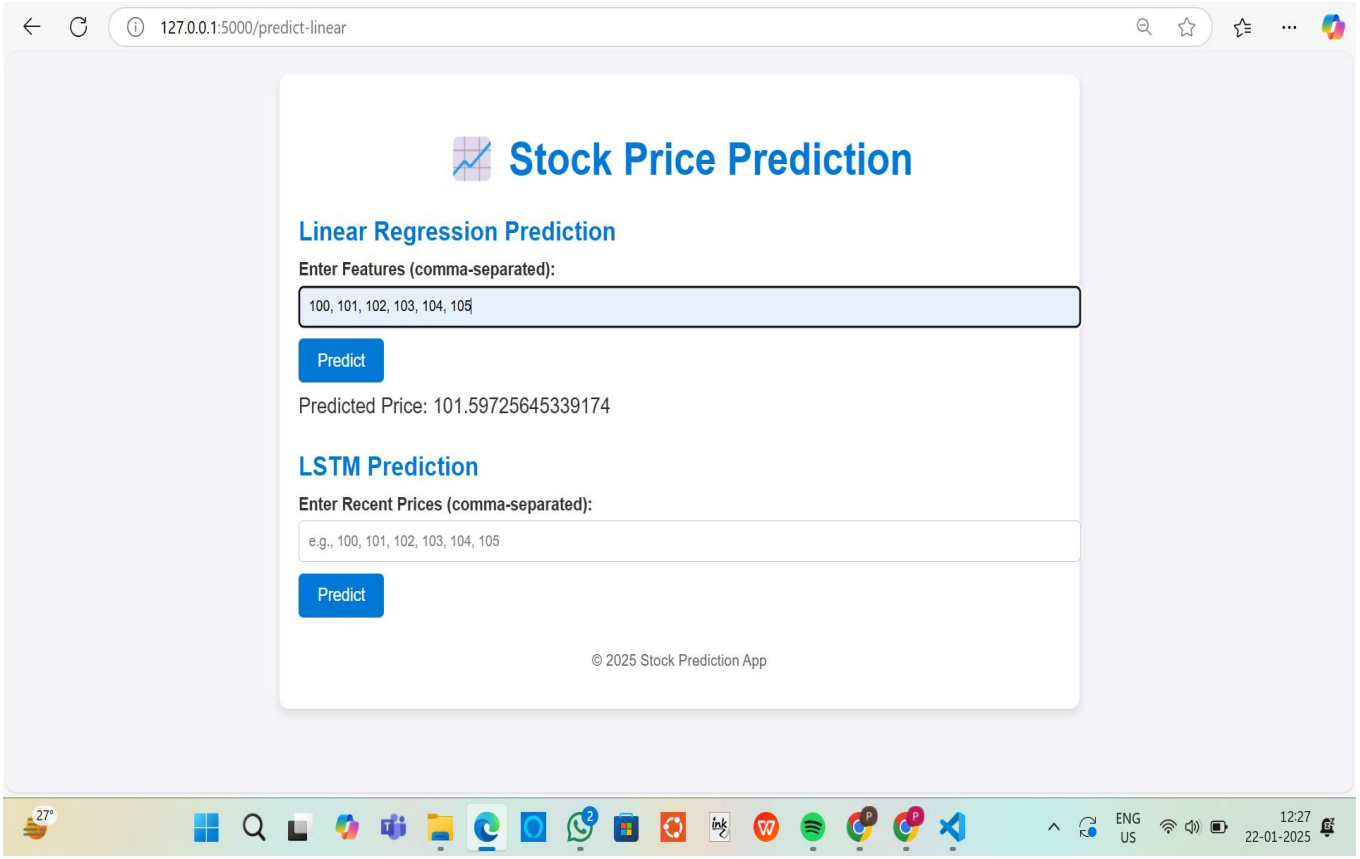
```
LSTM Model Performance:
MAE: 2.1359249993819223
MSE: 7.618938543688009
RMSE: 2.7602424791470783

Linear Regression Performance:
Mean Abs value: 0.2773168169894746
Mean squared value: 0.15226644841085718
Root mean squared error value: 0.390213337048924
```




Actual versus Prediction
Predicted
Actual
100
150
200
250
300


127.0.0.1:5000/predict-lstm
Stock Price Prediction
Linear Regression Prediction
Enter Features (comma-separated):
e.g., 100, 98, 102, 101, 50000, 500
Predict
LSTM Prediction
Enter Recent Prices (comma-separated):
100, 101, 102, 103, 104, 105
Predict
Predicted Price: 104.97543334960938
© 2025 Stock Prediction App



4. Remarks:-

This project integrates Linear Regression and LSTM models for stock price prediction into a Flask web application, providing an intuitive interface for users to input data and receive predictions. The Linear Regression model offers a simple approach based on selected features, while the LSTM model leverages its strength in time-series data to predict future prices. The website is styled for a clean, user-friendly experience, making machine learning accessible without requiring technical expertise. This project demonstrates the practical use of predictive models in real-world applications, with potential for further enhancements like real-time data and advanced visualizations.

Pruthibiraj Nayak (2230183)

(Name of the Student) (Name of the Coordinator)