LABORATORY REPORT

Application Development Lab
(CS33002)

B.Tech Program in ECSc

Submitted By

Name:- ABHISHEK ADARSH

Roll No: 2230141

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	21/01/2025
Date of Submission	27/01/2025

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

2. Procedure:-

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:-

Frontend code (html):

```
<!DOCTYPE html>
<html lang="en">

<head>
   <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Stock Trend Prediction</title>
```

```
                <link     href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>
        body {
            background-color: #181818;
            color: #f1f1f1;
        }

        .container {
            max-width: 1200px;
            margin-top: 50px;
        }

        h1 {
            color: #00b300;
            font-weight: bold;
        }

        .form-label {
            color: #ddd;
        }

        .form-control {
            background-color: #333;
            border-color: #555;
            color: #ddd;
        }

        .form-control:focus {
```

```
    background-color: #444;
    border-color: #00b300;
    color: #fff;
}

.btn-primary {
    background-color: #00b300;
    border-color: #00b300;
}

.btn-primary:hover {
    background-color: #008c00;
    border-color: #008c00;
}

.chart-container {
    margin-top: 30px;
    padding: 20px;
    background-color: #222;
    border-radius: 8px;
    box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.2);
    overflow: hidden;
}

.chart-container h3 {
    color: #00b300;
}

.img-fluid {
```

```
    display: block;
    max-width: 100%;
    height: auto;
    border-radius: 8px;
    box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.5);
    margin-bottom: 20px;
}

.table-responsive {
    margin-top: 20px;
    background-color: #222;
    border-radius: 8px;
    padding: 15px;
}

.table {
    color: #ddd;
}

.table th,
.table td {
    border-top: 1px solid #444;
}

.download-link {
    margin-top: 20px;
    margin-bottom: 40px;
}
```

```
        .download-link a {
            background-color: #00b300;
            color: #fff;
            padding: 10px 20px;
            border-radius: 5px;
            text-decoration: none;
        }

        .download-link a:hover {
            background-color: #008c00;
        }
    </style>
</head>

<body>
    <div class="container">
        <h1 class="text-center">Stock Trend Prediction</h1>
        <form method="POST">
            <div class="mb-3">
                        <label for="stock" class="form-label">Enter Stock Ticker:</label>
                        <input type="text" class="form-control" id="stock" name="stock" value="TSLA">
            </div>
            <button type="submit" class="btn btn-primary">Submit</button>
        </form>

        {% if plot_path_ema_20_50 %}
        <div class="chart-container">
```

```
        <h3>Closing Price vs Time 100MA</h3>
            <img src="{{ url_for('static', filename='ema_20_50.png') }}"
class="img-fluid" alt="100MA">
    </div>
    {% endif %}

    {% if plot_path_ema_100_200 %}
    <div class="chart-container">
        <h3>Closing Price vs Time 100 & 200MA</h3>
          <img src="{{ url_for('static', filename='ema_100_200.png') }}"
class="img-fluid" alt="100 & 200MA">
    </div>
    {% endif %}

    {% if plot_path_prediction %}
    <div class="chart-container">
        <h3>Prediction vs Original Trend</h3>
        <img src="{{ url_for('static', filename='stock_prediction.png') }}"
class="img-fluid"
            alt="Prediction vs Original">
    </div>
    {% endif %}

    {% if data_desc %}
    <div class="table-responsive">
            <h3 class="mt-4">Descriptive Data from Jan 2000 to Nov
2024</h3>
        {{ data_desc | safe }}
    </div>
```

```
        {% endif %}

        {% if dataset_link %}
        <div class="download-link">
                                        <a    href="{{    url_for('download_file',
filename=dataset_link.split('/')[-1]) }}" class="btn btn-success"
              download>Download Dataset (CSV)</a>
        </div>
        {% endif %}
    </div>

                <script     src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.bundle.min.js"></script>
</body>

</html>
```

Backend Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from keras.models import load_model
from flask import Flask, render_template, request, send_file
import datetime as dt
import yfinance as yf
from sklearn.preprocessing import MinMaxScaler
import os
plt.style.use("fivethirtyeight")
```

```
app = Flask(__name__)

# Load the model (ensure the model file is in the same directory)
model = load_model('keras_model.h5')

@app.route('/', methods=['GET', 'POST'])
def index():
    # Ensure the 'static' directory exists
    if not os.path.exists('static'):
        os.makedirs('static')

    if request.method == 'POST':
        stock = request.form.get('stock', 'TSLA')  # Default to 'TSLA' if no
stock is entered

        # Define the start and end dates for stock data
        start = dt.datetime(2010, 1, 1)
        end = dt.datetime(2019, 12, 31)

        # Download stock data
        try:
            df = yf.download(stock, start=start, end=end)

            # Check if data is fetched successfully
            if df.empty:
                return render_template(
                    'index.html',
```

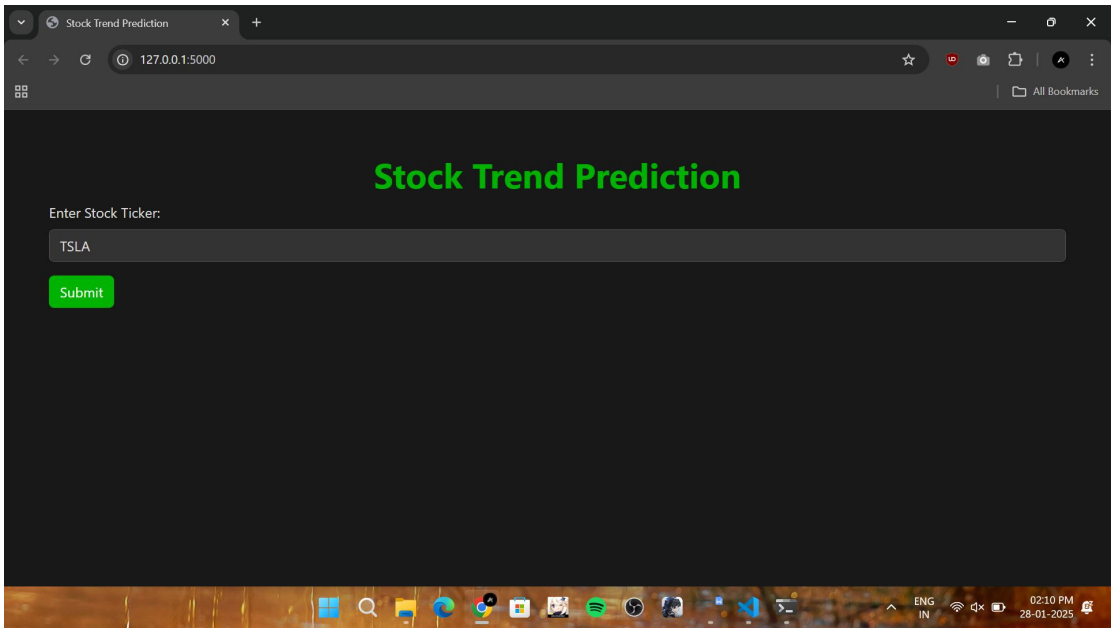
```
                error=f"No data available for stock '{stock}' in the specified
date range. Please try again.",
            )

    except Exception as e:
        return render_template(
            'index.html',
                error=f"Error fetching data for stock '{stock}': {e}. Please
check the stock symbol or try later.",
            )

    # Save dataset as CSV
    csv_file_path = f"static/{stock}_data.csv"
    df.to_csv(csv_file_path)

    # Descriptive Statistics
    data_desc = df.describe()

    # Exponential Moving Averages
    ma100 = df['Close'].ewm(span=100, adjust=False).mean()
    ma200 = df['Close'].ewm(span=200, adjust=False).mean()

    # Data Splitting
    data_training = pd.DataFrame(df['Close'][0:int(len(df) * 0.70)])
    data_testing = pd.DataFrame(df['Close'][int(len(df) * 0.70):])

    # Check if training data is empty
    if data_training.empty:
        return render_template(
```

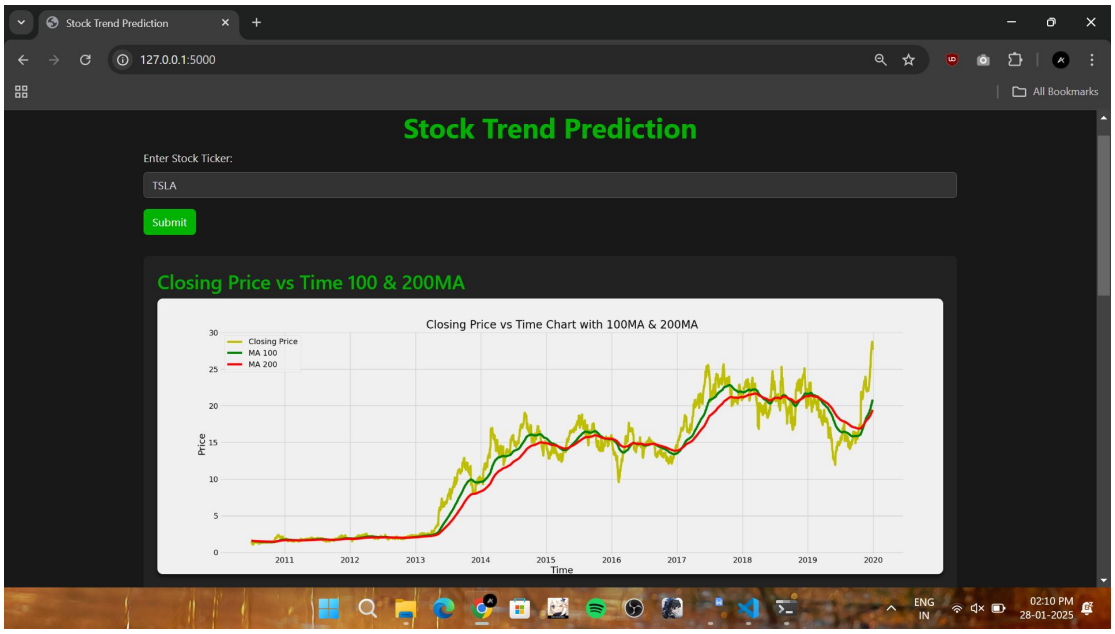
```
            'index.html',
             error="Insufficient data for training. Please try another stock or date range.",
        )

    # Scaling Data
    scaler = MinMaxScaler(feature_range=(0, 1))
    data_training_array = scaler.fit_transform(data_training)

    # Prepare Data for Prediction
    past_100_days = data_training.tail(100)
                    final_df = pd.concat([past_100_days, data_testing], ignore_index=True)
    input_data = scaler.fit_transform(final_df)

    x_test, y_test = [], []
    for i in range(100, input_data.shape[0]):
        x_test.append(input_data[i - 100:i])
        y_test.append(input_data[i, 0])
    x_test, y_test = np.array(x_test), np.array(y_test)

    # Make Predictions
    y_predicted = model.predict(x_test)

    # Inverse Scaling for Predictions
    scale_factor = 1 / scaler.scale_[0]
    y_predicted = y_predicted * scale_factor
    y_test = y_test * scale_factor
```

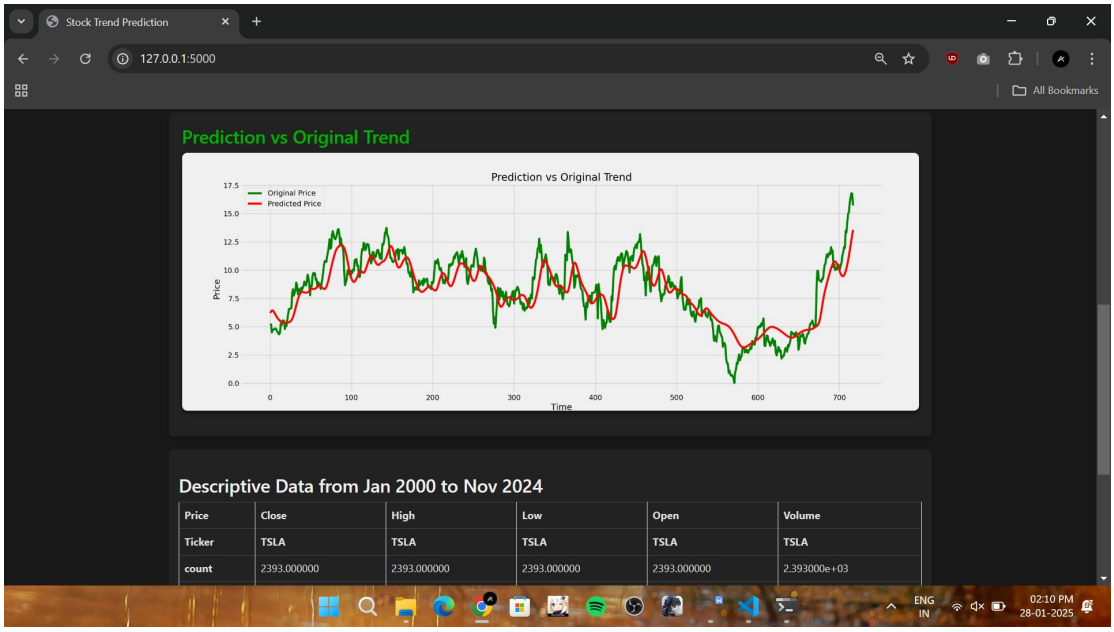
```
# Plot 1: Closing Price vs Time Chart with 100MA
fig1, ax1 = plt.subplots(figsize=(20, 7))
ax1.plot(df['Close'], 'y', label='Closing Price')
ax1.plot(ma100, 'r', label='100MA')
ax1.set_title("Closing Price vs Time Chart with 100MA")
ax1.set_xlabel("Time")
ax1.set_ylabel("Price")
ax1.legend()
ema_chart_path = "static/ema_100.png"
fig1.savefig(ema_chart_path)
plt.close(fig1)

# Plot 2: Closing Price vs Time Chart with 100 & 200MA
fig2, ax2 = plt.subplots(figsize=(20, 7))
ax2.plot(df['Close'], 'y', label='Closing Price')
ax2.plot(ma100, 'g', label='MA 100')
ax2.plot(ma200, 'r', label='MA 200')
ax2.set_title("Closing Price vs Time Chart with 100MA & 200MA")
ax2.set_xlabel("Time")
ax2.set_ylabel("Price")
ax2.legend()
ema_chart_path_100_200 = "static/ema_100_200.png"
fig2.savefig(ema_chart_path_100_200)
plt.close(fig2)

# Plot 3: Prediction vs Original Trend
fig3, ax3 = plt.subplots(figsize=(20, 7))
ax3.plot(y_test, 'g', label="Original Price")
ax3.plot(y_predicted, 'r', label="Predicted Price")
```

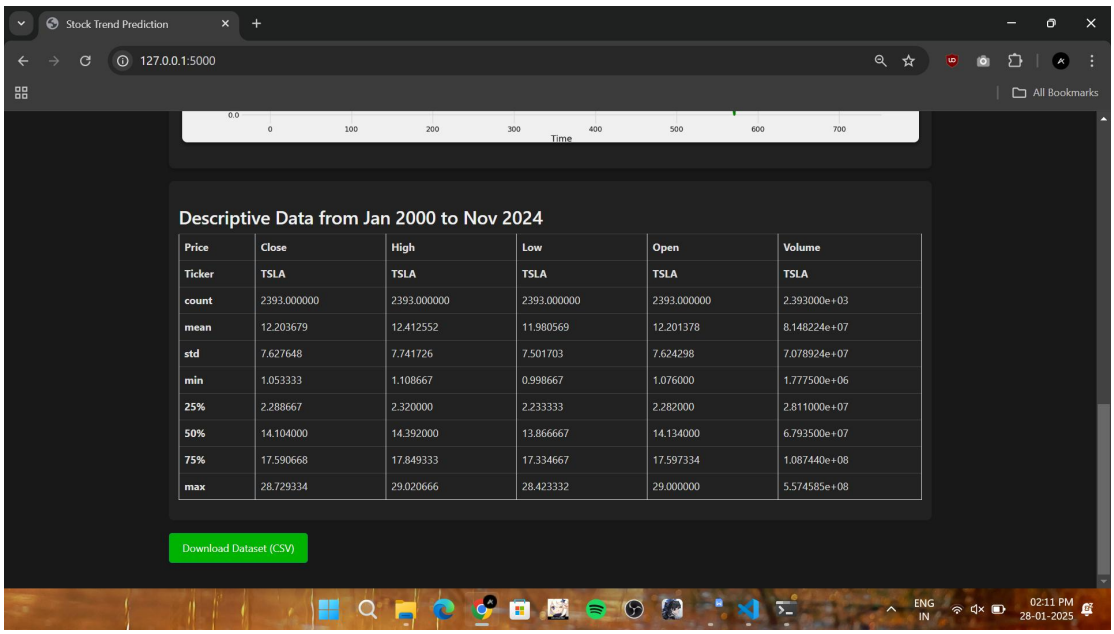
```
        ax3.set_title("Prediction vs Original Trend")
        ax3.set_xlabel("Time")
        ax3.set_ylabel("Price")
        ax3.legend()
        prediction_chart_path = "static/stock_prediction.png"
        fig3.savefig(prediction_chart_path)
        plt.close(fig3)

        # Return the rendered template with charts and dataset
        return render_template(
            'index.html',
            plot_path_ema_100=ema_chart_path,
            plot_path_ema_100_200=ema_chart_path_100_200,
            plot_path_prediction=prediction_chart_path,
            data_desc=data_desc.to_html(classes='table table-bordered'),
            dataset_link=csv_file_path,
        )

    return render_template('index.html')

@app.route('/download/<filename>')
def download_file(filename):
    return send_file(f"static/{filename}", as_attachment=True)

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

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









4. Remarks:-

Abhishek Adarsh

(Name of the Student)

Signature of the Lab Coordinator

(Name of the Coordinator)