LABORATORY REPORT

Application Development Lab
(CS33002)

B.Tech Program in ECSc

Submitted By

Name:-ARUNODOY GHOSH

Roll No: 2230071

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	21/01/25
Date of Submission	28/01/25

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.

3. Code:-

app.py

```
from flask import Flask, render_template, request, jsonify
import numpy as np
import pandas as pd
import yfinance as yf
from tensorflow.keras.models import load_model
from tensorflow.keras.losses import MeanSquaredError
import joblib
from datetime import datetime, timedelta
app = Flask(__name__)
```

```
STOCKS = ['AAPL', 'GOOGL', 'MSFT', 'TSLA']
def load_models(stock):
    lstm_model = load_model(f'C:/Users/aruno/OneDrive/Desktop/CODE MATERIALS/AD LAB/STOCK PREDICTION/models/{stock}_lstm_model.h5', custom_objects={'mse': MeanSquaredError()})
    linear_model = joblib.load(f'C:/Users/aruno/OneDrive/Desktop/CODE MATERIALS/AD LAB/STOCK PREDICTION/models/{stock}_linear_model.pkl')
        scaler = joblib.load(f'C:/Users/aruno/OneDrive/Desktop/CODE MATERIALS/AD LAB/STOCK PREDICTION/models/{stock}_scaler.pkl')
    return lstm_model, linear_model, scaler

def prepare_data(data, lookback=60):
    X = []
    for i in range(lookback, len(data)):
        X.append(data[i-lookback:i])
    return np.array(X)

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

@app.route('/predict', methods=['POST'])
def predict():
    try:
        stock = request.json['stock']
        model_type = request.json['model_type']
```

```
end_date = datetime.now()
start_date = end_date - timedelta(days=100)
df = yf.download(stock, start=start_date, end=end_date)

if df.empty:
    return jsonify({'error': 'No data found for the given stock'})

lstm_model, linear_model, scaler = load_models(stock)
data = df['Close'].values.reshape(-1, 1)
scaled_data = scaler.transform(data)
X = prepare_data(scaled_data)

predictions = {}
historical = df['Close'].tolist()
dates = [d.strftime('%Y-%m-%d') for d in df.index]

if model_type in ['lstm', 'comparison']:
    lstm_pred = lstm_model.predict(X)
    lstm_pred = scaler.inverse_transform(lstm_pred)
    predictions['lstm'] = lstm_pred.reshape(-1).tolist()

if model_type in ['linear', 'comparison']:
    X_2d = X.reshape(X.shape[0], -1)
    linear_pred = linear_model.predict(X_2d)
    linear_pred = scaler.inverse_transform(linear_pred.reshape(-1, 1))
    predictions['linear'] = linear_pred.reshape(-1).tolist()

return jsonify({
```

```
            'dates': dates[60:],
            'historical': historical[60:],
            'predictions': predictions
        })

    except Exception as e:
        return jsonify({'error': str(e)})

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

model.py

```
  # Install required libraries
# Import libraries
import yfinance as yf
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# Download stock price dataset
ticker = 'AAPL'  # Example: Apple stock
data = yf.download(ticker, start="2015-01-01", end="2023-01-01")
data.reset_index(inplace=True)

# Preprocessing
```

```
if 'Close' not in data.columns:
    raise KeyError("'Close' column not found in the dataset.")

data['Date'] = pd.to_datetime(data['Date'])
data['Close'] = data['Close'].fillna(method='ffill')  # Fill missing values

# Ensure there is enough data for processing
if len(data) < 11:  # LSTM requires a sequence length of at least 10
    raise ValueError("Dataset has insufficient rows for processing.")

# Features and labels
X = np.array(range(len(data['Close']))).reshape(-1, 1)
y = data['Close'].values

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Linear Regression Model
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_predictions = lr_model.predict(X_test)

# Prepare data for LSTM
seq_length = 10
X_lstm = []
y_lstm = []
for i in range(len(y) - seq_length):
    X_lstm.append(y[i:i + seq_length])
```

```
    y_lstm.append(y[i + seq_length])

X_lstm, y_lstm = np.array(X_lstm), np.array(y_lstm)
X_lstm = X_lstm.reshape((X_lstm.shape[0], X_lstm.shape[1], 1))

# Check if LSTM data is sufficient
if len(X_lstm) < 1:
    raise ValueError("Insufficient data for LSTM training.")

# Train-test split for LSTM
split = int(0.8 * len(X_lstm))
X_train_lstm, X_test_lstm = X_lstm[:split], X_lstm[split:]
y_train_lstm, y_test_lstm = y_lstm[:split], y_lstm[split:]

# LSTM Model
lstm_model = Sequential([
    LSTM(50, return_sequences=True, input_shape=(seq_length, 1)),
    LSTM(50),
    Dense(1)
])
lstm_model.compile(optimizer='adam', loss='mse')
lstm_model.fit(X_train_lstm, y_train_lstm, epochs=20, batch_size=32)

# Predictions
lstm_predictions = lstm_model.predict(X_test_lstm)

# Compare models
lr_mse = mean_squared_error(y_test, lr_predictions[:len(y_test)])
lstm_mse = mean_squared_error(y_test_lstm, lstm_predictions)
```

```
print(f"Linear Regression MSE: {lr_mse}")
print(f"LSTM MSE: {lstm_mse}")
```

index.html

```
<!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>
                                                                    <script
src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/3.7.0/chart.min.js"></
script>
    <style>
        :root {
            --primary-color: #007bff;
            --secondary-color: #333;
            --background-color: #f5f5f5;
            --text-color: #333;
            --border-color: #ddd;
        }

        [data-theme="dark"] {
            --primary-color: #0d6efd;
            --secondary-color: #ffffff;
            --background-color: #121212;
            --text-color: #ffffff;
            --border-color: #555;
```

```
}

body {
    font-family: 'Arial', sans-serif;
    margin: 0;
    padding: 20px;
    background-color: var(--background-color);
    color: var(--text-color);
    transition: all 0.3s ease-in-out;
}

.container {
    max-width: 1200px;
    margin: 0 auto;
    background-color: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    background-color: var(--background-color);
    color: var(--text-color);
}

h1 {
    color: var(--secondary-color);
    text-align: center;
    margin-bottom: 30px;
}

.controls {
```

```
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(200px,1fr));
    gap: 20px;
    margin-bottom: 30px;
}

select, button {
    padding: 10px;
    font-size: 16px;
    border: 1px solid var(--border-color);
    border-radius: 5px;
    background-color: var(--background-color);
    color: var(--text-color);
    transition: all 0.3s ease;
}

button {
    background-color: var(--primary-color);
    color: white;
    border: none;
    cursor: pointer;
}

button:hover {
    background-color: #0056b3;
}

button:disabled {
    background-color: #ccc;
```

```
    cursor: not-allowed;
}

.chart-container {
    position: relative;
    height: 60vh;
    margin-top: 20px;
}

.loading {
    text-align: center;
    margin: 20px 0;
    color: #666;
    display: none;
}

.error {
    color: #dc3545;
    text-align: center;
    margin: 20px 0;
    display: none;
}

.empty-state {
    text-align: center;
    color: var(--text-color);
    margin-top: 30px;
}
```

```
        .theme-toggle {
            position: fixed;
            top: 20px;
            right: 20px;
            background-color: var(--background-color);
            border: 1px solid var(--border-color);
            border-radius: 50%;
            width: 40px;
            height: 40px;
            display: flex;
            align-items: center;
            justify-content: center;
            cursor: pointer;
            transition: background-color 0.3s ease;
        }

        .theme-toggle:hover {
            background-color: var(--primary-color);
            color: white;
        }

        @media (max-width: 768px) {
            .controls {
                grid-template-columns: 1fr;
            }
        }
    </style>
</head>
<body data-theme="light">
```

```
    <div class="theme-toggle" onclick="toggleTheme()" title="Toggle Dark Mode">
        🌓
    </div>

    <div class="container">
        <h1>Stock Price Prediction</h1>

        <div class="controls">
            <select id="stockSelect" title="Choose a stock">
                <option value="">Select Stock</option>
                {% for stock in stocks %}
                <option value="{{ stock }}">{{ stock }}</option>
                {% endfor %}
            </select>

            <select id="modelSelect" title="Choose a prediction model">
                <option value="">Select Model</option>
                <option value="lstm">LSTM Prediction</option>
                <option value="linear">Linear Regression Prediction</option>
                <option value="comparison">Compare Both Models</option>
            </select>

                    <button id="predictBtn" onclick="getPrediction()">Get Prediction</button>
        </div>

        <div id="loading" class="loading">Loading predictions...</div>
        <div id="error" class="error"></div>
```

```
        <div class="chart-container">
            <canvas id="predictionChart"></canvas>
                <div id="emptyState" class="empty-state" style="display: none;">
                Select a stock and model to view predictions.
            </div>
        </div>
    </div>

    <script>
        let chart;

        function toggleTheme() {
            const body = document.body;
            const currentTheme = body.getAttribute("data-theme");
                body.setAttribute("data-theme", currentTheme === "light" ? "dark" : "light");
        }

        async function getPrediction() {
            const stock = document.getElementById('stockSelect').value;
                                                const modelType = document.getElementById('modelSelect').value;
            const loadingDiv = document.getElementById('loading');
            const errorDiv = document.getElementById('error');
            const emptyState = document.getElementById('emptyState');
            const predictBtn = document.getElementById('predictBtn');
```

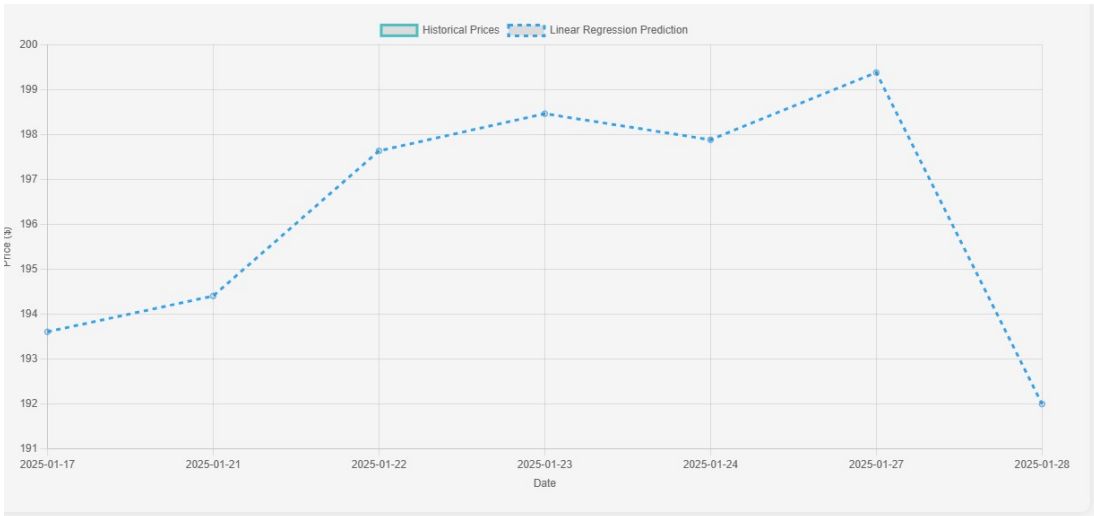
```
        if (!stock || !modelType) {
                errorDiv.textContent = 'Please select both stock and model
type';
            errorDiv.style.display = 'block';
            return;
        }

        predictBtn.disabled = true;
        loadingDiv.style.display = 'block';
        errorDiv.style.display = 'none';
        emptyState.style.display = 'none';
        try {
            const response = await fetch('/predict', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                },
                body: JSON.stringify({ stock, model_type: modelType }),
            });
            const data = await response.json();
            if (data.error) {
                throw new Error(data.error);
            }
            updateChart(data, modelType);
        } catch (error) {
            errorDiv.textContent = error.message;
            errorDiv.style.display = 'block';
        } finally {
            loadingDiv.style.display = 'none';
```

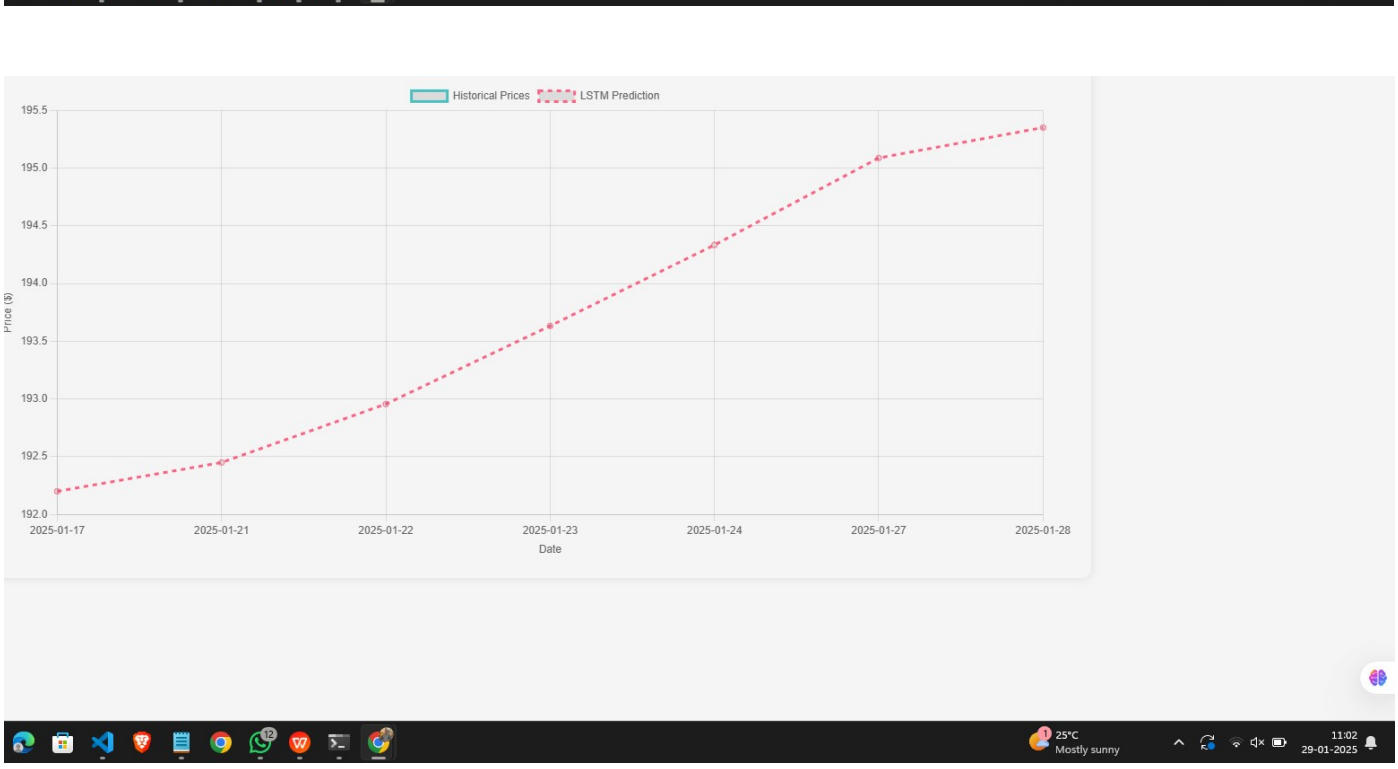
```
            predictBtn.disabled = false;
        }
    }
    function updateChart(data, modelType) {
                                                    const ctx =
document.getElementById('predictionChart').getContext('2d');
        if (chart) {
            chart.destroy();
        }
        if (!data.historical || data.historical.length === 0) {
            document.getElementById('emptyState').style.display = 'block';
            return;
        }
        const datasets = [{
            label: 'Historical Prices',
            data: data.historical,
            borderColor: 'rgb(75, 192, 192)',
            tension: 0.1,
        }]
        if (modelType === 'lstm' || modelType === 'comparison') {
            datasets.push({
                label: 'LSTM Prediction',
                data: data.predictions.lstm,
                borderColor: 'rgb(255, 99, 132)',
                borderDash: [5, 5],
            });
        }
        if (modelType === 'linear' || modelType === 'comparison') {
            datasets.push({
```

```
            label: 'Linear Regression Prediction',
            data: data.predictions.linear,
            borderColor: 'rgb(54, 162, 235)',
            borderDash: [5, 5],
        });
    }
    chart = new Chart(ctx, {
        type: 'line',
        data: {
            labels: data.dates,
            datasets: datasets,
        },
        options: {
            responsive: true,
            maintainAspectRatio: false,
            interaction: {
                intersect: false,
                mode: 'index',
            },
            plugins: {
                title: {
                    display: true,
                    text: `Stock Price Prediction - ${document.getElementById('stockSelect').value}`,
                },
                legend: {
                    position: 'top',
                },
            },
```

```
                scales: {
                    y: {
                        beginAtZero: false,
                        title: {
                            display: true,
                            text: 'Price ($)',
                        },
                    },
                    x: {
                        title: {
                            display: true,
                            text: 'Date',
                        },
                    },
                },
            },
        });
    }
  </script>
</body>
</html>
```

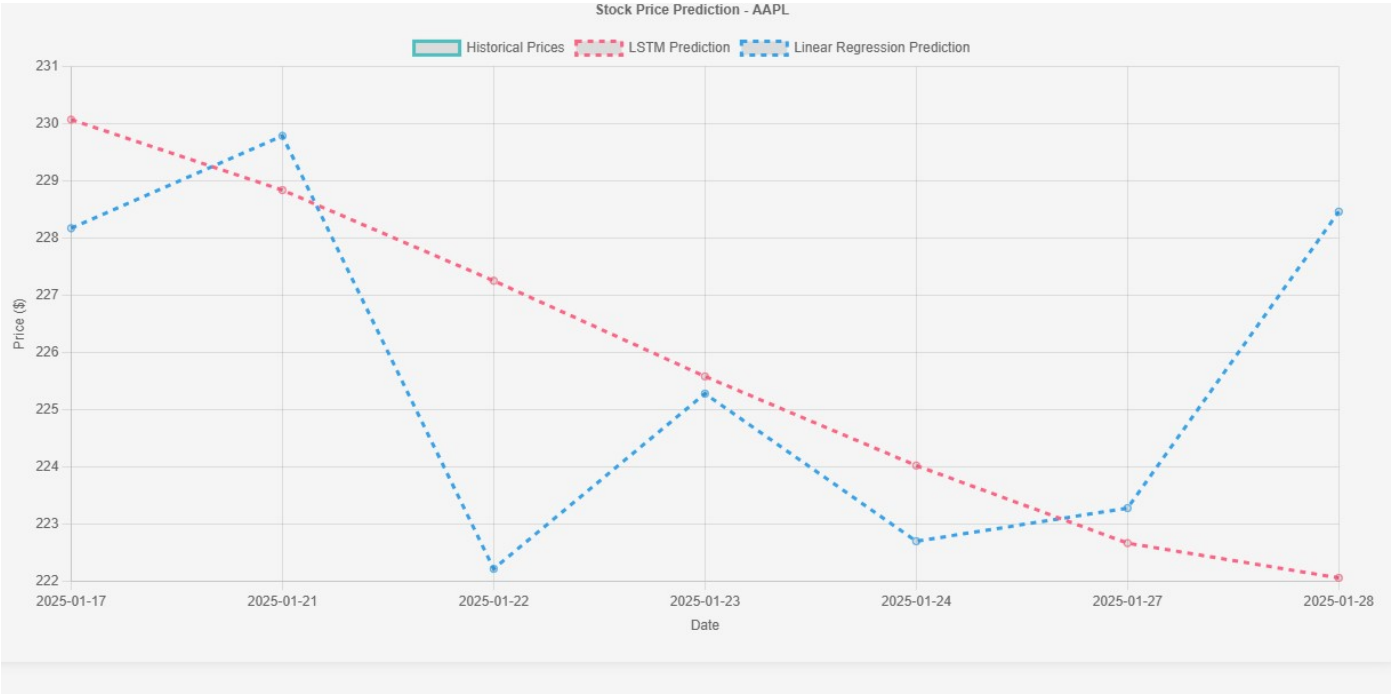
4. Results/Output:-


Historical Prices
Linear Regression Prediction
200
199
198
197
196
195
194
193
192
191
2025-01-17
2025-01-21
2025-01-22
2025-01-23
2025-01-24
2025-01-27
2025-01-28
Date


25°C
Mostly sunny
11:01
29-01-2025

Historical Prices
LSTM Prediction
195.5
195.0
194.5
194.0
193.5
193.0
192.5
192.0
2025-01-17
2025-01-21
2025-01-22
2025-01-23
2025-01-24
2025-01-27
2025-01-28
Date

25°C
Mostly sunny
11:02
29-01-2025


Stock Price Prediction - AAPL
Historical Prices
LSTM Prediction
Linear Regression Prediction
231
230
229
228
227
226
225
224
223
222
Price ($)
2025-01-17
2025-01-21
2025-01-22
2025-01-23
2025-01-24
2025-01-27
2025-01-28
Date

5. Remarks:-

In this project, we successfully implemented and trained predictive models to forecast stock prices for different companies using two distinct techniques:Linear Regression and Long Short-Term Memory (LSTM) networks. The models were trained on historical stock price data, fetched from Yahoo Finance, and evaluated using key performance metrics such as Mean Squared Error (MSE) and R2 score.Both models were deployed using a Flask backend, enabling users to interact with the system via a web interface, where they could select a stock, choose the prediction method (LSTM, Linear Regression, or comparison of both), and view the predictions alongside historical data.

ARUNODOY GHOSH

(Name of the Student)

Signature of the Lab Coordinator

(Name of the Coordinator)