LABORATORY REPORT

Application Development Lab (CS33002)

B.Tech Program in ECSc

Submitted By

Name: Shreyaa Venkateswaran

Roll No: 2230120

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.

3. Code:

Model Training Code:

```
import numpy as np

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')

from google.colab import drive

drive.mount('/content/drive')

data=pd.read_csv("/content/drive/MyDrive/Stock/dataset/INDIAVIX.csv") #reads the input data

data.head() #displays the first five rows

data.info()

data.describe(include ='all')

data.isnull().sum()

data.head()

sns.pairplot(data)

plt.show()

x=data[['Date','Open','High','Low','Close','Previous','Change','%Change']].values

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

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)

data['Date'] = pd.to_datetime(data['Date'])

# Extract year, month, day, day of the week from the Date

data['Year'] = data['Date'].dt.year

data['Month'] = data['Date'].dt.month
```

```
data['Day'] = data['Date'].dt.day

data['Day_of_week'] = data['Date'].dt.dayofweek # Monday=0, Sunday=6

X = data[['Open', 'High', 'Low', 'Previous', 'Change', '%Change', 'Year', 'Month', 'Day', 'Day_of_week']]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print(f"Training data size: {len(X_train)}")

print(f"Test data size: {len(X_test)}")

#using linear regression

lm=LinearRegression()

lm.fit(X_train,y_train)

lm.coef_

#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)

from sklearn.impute import SimpleImputer

# Create an imputer instance

imputer = SimpleImputer(strategy='mean') # Use 'median' or other strategies if more appropriate

# Fit and transform the training and test data

X_train = imputer.fit_transform(X_train)

X_test = imputer.transform(X_test)

#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.flatten()})
#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)
import matplotlib.pyplot as plt
# Calculate residuals
residuals = y_test - predictions
# Create the residual plot
plt.figure(figsize=(8, 6))
plt.scatter(predictions, residuals, alpha=0.7)
plt.axhline(0, color='red', linestyle='--', linewidth=1) # Horizontal line at residual = 0
plt.title("Residual Plot", fontsize=16)
plt.xlabel("Predicted Values", fontsize=14)

plt.ylabel("Residuals (Actual - Predicted)", fontsize=14)

plt.grid(alpha=0.3)

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)))

import os

folder_path = '/content/drive/My Drive/your_folder_name'

os.makedirs(folder_path, exist_ok=True) # This creates the folder if it doesn't exist

# Save the model

model_path=os.path.join(folder_path,'linear_regression_model.pkl)

dump(lm, model_path)

print(f"Model saved to {model_path}")
```

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>
</head>
<body>
<h1>Enter Stock Data for Prediction</h1>
<form action="/predict" method="post">
<label for="Open">Open:</label>
<input type="text" name="Open" required><br><br>
<label for="High">High:</label>
<input type="text" name="High" required><br><br>
<label for="Low">Low:</label>
<input type="text" name="Low" required><br><br>
<label for="Previous">Previous:</label>
<input type="text" name="Previous" required><br><br>
<label for="Change">Change:</label>
<input type="text" name="Change" required><br><br>
<label for="%Change">% Change:</label>
<input type="text" name="%Change" required><br><br>
<label for="Year">Year:</label>
<input type="text" name="Year" required><br><br>
<label for="Month">Month:</label>
<input type="text" name="Month" required><br><br>
```

```
<label for="Day">Day:</label>
<input type="text" name="Day" required><br><br>
<label for="Day_of_week">Day of the week:</label>
<input type="text" name="Day_of_week" required><br><br>
<button type="submit">Predict</button>
</form>
<h2>{{ prediction_text }}</h2>
</body>
</html>
```

app.py (Flask code):

```
from flask import Flask, render_template, request
from joblib import load
import numpy as np
# Initialize Flask app
app = Flask(__name__)
# Load the trained model
model_path = 'model/linear_regression_model.pkl' # Path to model in your folder
model = load(model_path)
@app.route('/')
def home():
return render_template('index.html')
@app.route('/predict', methods=['POST'])
def predict():
```

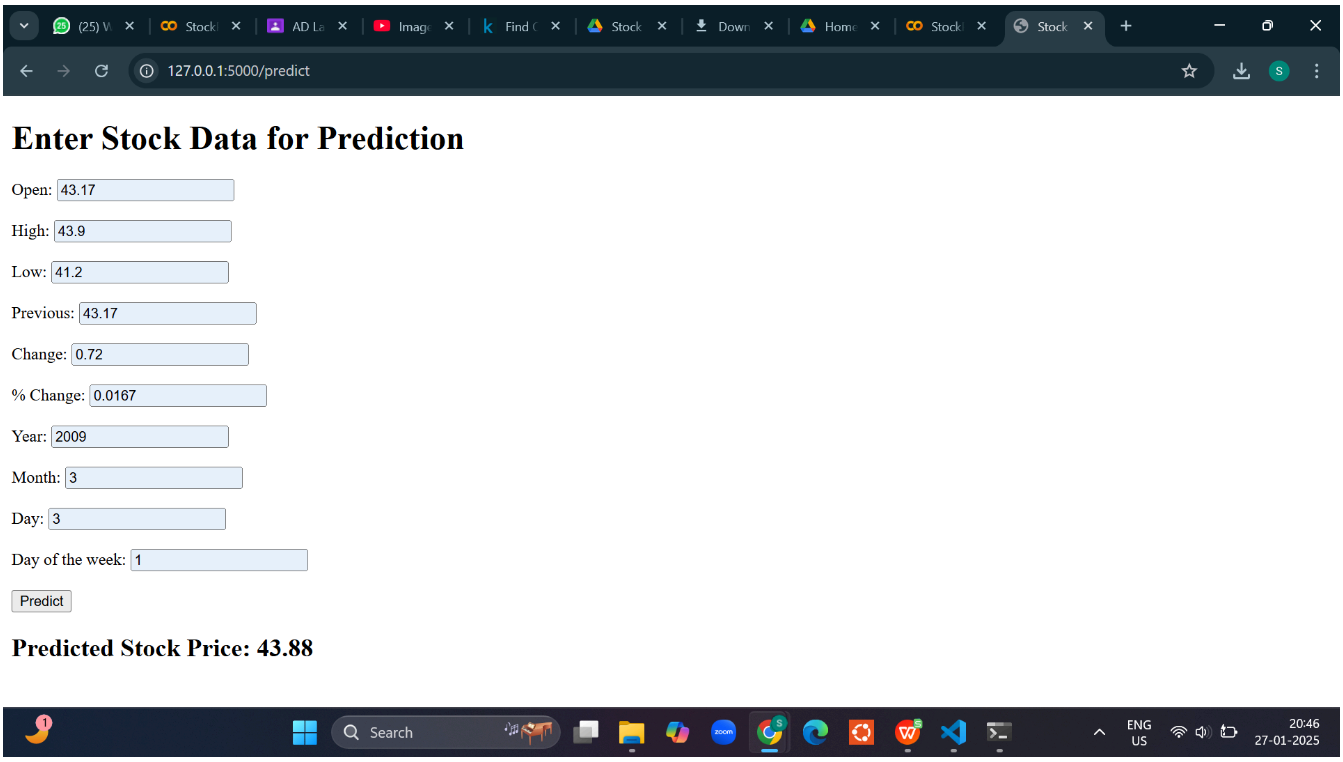
```
try:
# Get the form data (input values from the user)
features = [
float(request.form['Open']),
float(request.form['High']),
float(request.form['Low']),
float(request.form['Previous']),
float(request.form['Change']),
float(request.form['%Change']),
int(request.form['Year']),
int(request.form['Month']),
int(request.form['Day']),
int(request.form['Day_of_week'])
]
# Convert features to a 2D array (model expects 2D array)
features = np.array(features).reshape(1, -1)
# Predict the stock price using the trained model
prediction = model.predict(features)
# Ensure the prediction is a scalar value
prediction_value = prediction.item() # This converts the numpy array to a scalar
# Return the predicted value to the user
return render_template('index.html', prediction_text='Predicted Stock Price: {:.2f}'.format(prediction_value))
except Exception as e:
```

```
return render_template('index.html', prediction_text="Error: " + str(e))

if __name__ == '__main__':

app.run(debug=True)
```

4. Results/Output:



Enter Stock Data for Prediction

Open: 43.17

High: 43.9

Low: 41.2

Previous: 43.17

Change: 0.72

% Change: 0.0167

Year: 2009

Month: 3

Day: 3

Day of the week: 1

Predicted Stock Price: 43.88

5. Remarks:

Built a linear regression model for stock data spanning over a few years. The frontend is done using HTML while the backend is done using flask. The model shows pretty accurate results.

Initially there was a problem in integrating the flask interface along with the model however it was later rectified.

Website link: Stock Price Prediction

GitHub link: https://github.com/ShreyaaVenkateswaran

Shreyaa Venkateswaran

Signature of the Lab Coordinator