LABORATORY REPORT

Application Development Lab
(CS33002)

B.Tech Program in ECSc

Submitted By

Name:-Rohit Kumar Satpathy

Roll No: 2230038

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

Spring 2024-2025

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:- (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.

3. Code:-

Pre-processing the data for analysis

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
import matplotlib.pyplot as plt
```

```
stock_prices = pd.read_csv(r"C:\\Users\\KIIT\\OneDrive\\Documents\\AD Lab\\Lab3\\historical_stock_prices.csv")
stock_prices['date'] = pd.to_datetime(stock_prices['date'])


stock_data = stock_prices[stock_prices['ticker'] == 'AAPL']
stock_data = stock_data[['date', 'close']]

stock_prices = pd.read_csv(r"C:\\Users\\KIIT\\OneDrive\\Documents\\AD Lab\\Lab3\\historical_stock_prices.csv")
stock_prices['date'] = pd.to_datetime(stock_prices['date'])

stock_data = stock_prices[stock_prices['ticker'] == 'AAPL']
stock_data = stock_data[['date', 'close']]

 stock_data = stock_data.sort_values(by='date')

scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(stock_data[['close']])

def create_dataset(data, time_step=1):
    X, y = [], []
    for i in range(len(data) - time_step - 1):
        X.append(data[i:(i + time_step), 0])
        y.append(data[i + time_step, 0])
    return np.array(X), np.array(y)

time_step = 60
X, y = create_dataset(scaled_data, time_step)

X_lstm = X.reshape(X.shape[0], X.shape[1], 1)

X_lstm = X.reshape(X.shape[0], X.shape[1], 1)

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

Implementation of Linear regression model

```
X_lr = np.array([scaled_data[i:i + time_step].flatten() for i in range(len(scaled_data) - time_step)])
y_lr = scaled_data[time_step:]

X_train_lr, X_test_lr, y_train_lr, y_test_lr = train_test_split(X_lr, y_lr, test_size=0.2, random_state=42)

model_lr = LinearRegression()
model_lr.fit(X_train_lr, y_train_lr)
```

Output:

```
▾ LinearRegression
LinearRegression()
```

```
print("Coefficients:", model_lr.coef_)
print("Intercept:", model_lr.intercept_)
```

Output:

```
Coefficients: [[-0.04155551  0.01485497  0.00309308  0.0365469  -0.00973016 -0.02230525
  -0.00183749  0.03856867 -0.0245641   0.0169666  -0.04750607  0.06538593
  -0.06362138  0.0308973   0.00373014  0.0121101  -0.03428811  0.0332388
  -0.01432668  0.0465431  -0.06178462  0.00572276  0.03518186 -0.05383799
   0.08357324 -0.05933193  0.0168304  -0.01289363  0.03553011 -0.00572944
  -0.01208151 -0.00822791  0.05597744 -0.11092924  0.06333797 -0.01388794
   0.025737    0.00811379 -0.06196126  0.01580123 -0.02320361  0.05917857
  -0.05521831  0.02500307  0.0163871  -0.05022486  0.0562716  -0.01138555
   0.03637093 -0.0465588  -0.01413227  0.10314029 -0.12403161  0.04289825
   0.02123028 -0.08276648  0.0841412  -0.00443508 -0.08673359  1.06756557]]
Intercept: [9.87802521e-06]
```

```
y_pred_lr = model_lr.predict(X_test_lr)

mse_lr = mean_squared_error(y_test_lr, y_pred_lr)
r2_lr = r2_score(y_test_lr, y_pred_lr)

print(f"Mean Squared Error: {mse_lr}")
print(f"R-squared: {r2_lr}")
```

Output:

```
Mean Squared Error: 1.2319795719945155e-05
R-squared: 0.9997302820973807
```

Training of LSTM model

```
model_lstm = Sequential()
model_lstm.add(LSTM(units=50, return_sequences=True,
input_shape=(X_train.shape[1], 1)))
model_lstm.add(LSTM(units=50, return_sequences=False))
model_lstm.add(Dense(units=1))  # Output layer

model_lstm.compile(optimizer='adam', loss='mean_squared_error')
model_lstm.fit(X_train, y_train, epochs=20, batch_size=16,
validation_data=(X_test, y_test))
```

Output:

```
Epoch 1/20
c:\Users\KIIT\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\rnn\rnn.py:200: UserW
  super().__init__(**kwargs)
473/473 ━━━━━━━━━━━━━━━━━━━━ 11s 20ms/step - loss: 0.0021 - val_loss: 1.3639e-04
Epoch 2/20
473/473 ━━━━━━━━━━━━━━━━━━━━ 9s 19ms/step - loss: 8.3760e-05 - val_loss: 6.4508e-05
Epoch 3/20
473/473 ━━━━━━━━━━━━━━━━━━━━ 11s 24ms/step - loss: 9.2812e-05 - val_loss: 5.3449e-05
Epoch 4/20
473/473 ━━━━━━━━━━━━━━━━━━━━ 11s 23ms/step - loss: 9.1172e-05 - val_loss: 5.0351e-05
Epoch 5/20
473/473 ━━━━━━━━━━━━━━━━━━━━ 11s 23ms/step - loss: 5.4766e-05 - val_loss: 4.7512e-05
Epoch 6/20
473/473 ━━━━━━━━━━━━━━━━━━━━ 11s 23ms/step - loss: 6.1570e-05 - val_loss: 3.0769e-05
Epoch 7/20
473/473 ━━━━━━━━━━━━━━━━━━━━ 11s 23ms/step - loss: 4.3846e-05 - val_loss: 2.2939e-05
Epoch 8/20
473/473 ━━━━━━━━━━━━━━━━━━━━ 11s 23ms/step - loss: 4.2318e-05 - val_loss: 3.7291e-05
Epoch 9/20
473/473 ━━━━━━━━━━━━━━━━━━━━ 11s 23ms/step - loss: 3.5982e-05 - val_loss: 2.3886e-05
Epoch 10/20
473/473 ━━━━━━━━━━━━━━━━━━━━ 11s 23ms/step - loss: 3.8374e-05 - val_loss: 1.6380e-05
Epoch 11/20
473/473 ━━━━━━━━━━━━━━━━━━━━ 12s 25ms/step - loss: 2.2714e-05 - val_loss: 4.0233e-05
Epoch 12/20
473/473 ━━━━━━━━━━━━━━━━━━━━ 11s 24ms/step - loss: 3.2889e-05 - val_loss: 2.7470e-05
Epoch 13/20
473/473 ━━━━━━━━━━━━━━━━━━━━ 11s 23ms/step - loss: 3.2598e-05 - val_loss: 3.2901e-05
```

```python
predicted_prices_lstm = model_lstm.predict(X_test)
predicted_prices_lstm = scaler.inverse_transform(predicted_prices_lstm)

mse_lstm = mean_squared_error(y_test, predicted_prices_lstm)
r2_lstm = r2_score(y_test, predicted_prices_lstm)

print(f"Mean Squared Error: {mse_lstm}")
print(f"R-squared: {r2_lstm}")
```

Output:

```
Mean Squared Error: 2568.387619480227
R-squared: -60227.56472880524
```

Comparison between both the models

```python
print(f"Linear Regression Model MSE: {mse_lr}, R2: {r2_lr}")
print(f"LSTM Model MSE: {mse_lstm}, R2: {r2_lstm}")
```

Output:

```
Linear Regression Model MSE: 1.2319795719945155e-05, R2: 0.9997302820973807
LSTM Model MSE: 2568.387619480227, R2: -60227.56472880524
```

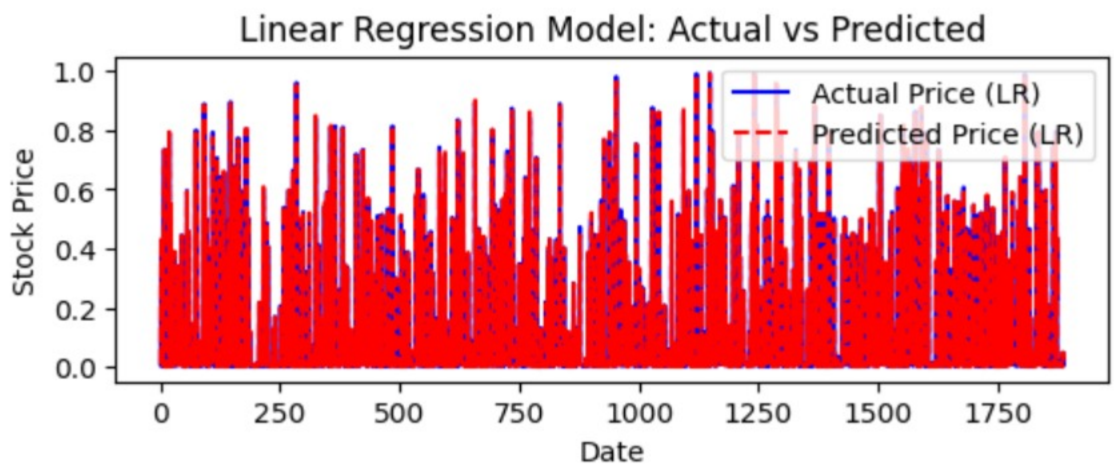
```python
plt.figure(figsize=(10, 6))
plt.subplot(2, 1, 1)
```

```
plt.plot(y_test_lr, color='blue', label='Actual Price (LR)')
plt.plot(y_pred_lr, color='red', linestyle='dashed', label='Predicted
Price (LR)')
plt.title('Linear Regression Model: Actual vs Predicted')
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.legend()
```

Output:

```
<matplotlib.legend.Legend at 0x295a0ddf0b0>
```

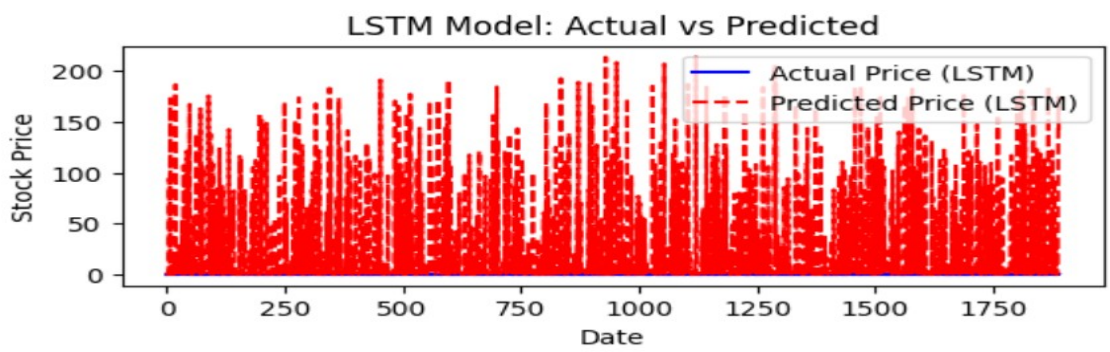
```
# Actual vs Predicted for LSTM
```




```
plt.subplot(2, 1, 2)
plt.plot(y_test, color='blue', label='Actual Price (LSTM)')
plt.plot(predicted_prices_lstm, color='red', linestyle='dashed',
label='Predicted Price (LSTM)')
plt.title('LSTM Model: Actual vs Predicted')
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.legend()
```

Output:

```
<matplotlib.legend.Legend at 0x295a01cd0d0>
```



Flask backend for model predictions:

app.py

```
from flask import Flask, request, jsonify, render_template
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import joblib
import os

app = Flask(__name__)

def train_model():

    stock_prices = pd.read_csv(r"C:\Users\KIIT\OneDrive\Documents\AD
Lab\Lab3\historical_stock_prices.csv")
    stocks = pd.read_csv(r"C:\Users\KIIT\OneDrive\Documents\AD
Lab\Lab3\historical_stocks.csv")

    merged_data = pd.merge(stock_prices, stocks, on='ticker')
    stock_data = merged_data[merged_data['ticker'] == 'AAPL']
    stock_data['date'] = pd.to_datetime(stock_data['date'])
    stock_data = stock_data.sort_values(by='date')
    stock_data['Target'] = stock_data['close'].shift(-1)
    stock_data = stock_data.dropna()

    features = ['close', 'volume']
    X = stock_data[features]
    y = stock_data['Target']

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

    model = LinearRegression()
    model.fit(X_train, y_train)

    joblib.dump(model, 'linear_regression_model.pkl')
    return model

if os.path.exists('linear_regression_model.pkl'):
    model = joblib.load('linear_regression_model.pkl')
else:
    model = train_model()

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
```

```
def predict():
    try:

        data = request.get_json()
        close = float(data['close'])
        volume = float(data['volume'])

        prediction = model.predict([[close, volume]])[0]
        return jsonify({'predicted_price': prediction})
    except Exception as e:
        print(f"Prediction Error: {e}")
        return jsonify({'error': 'Prediction failed. Please check your
input and try again.'}), 400

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

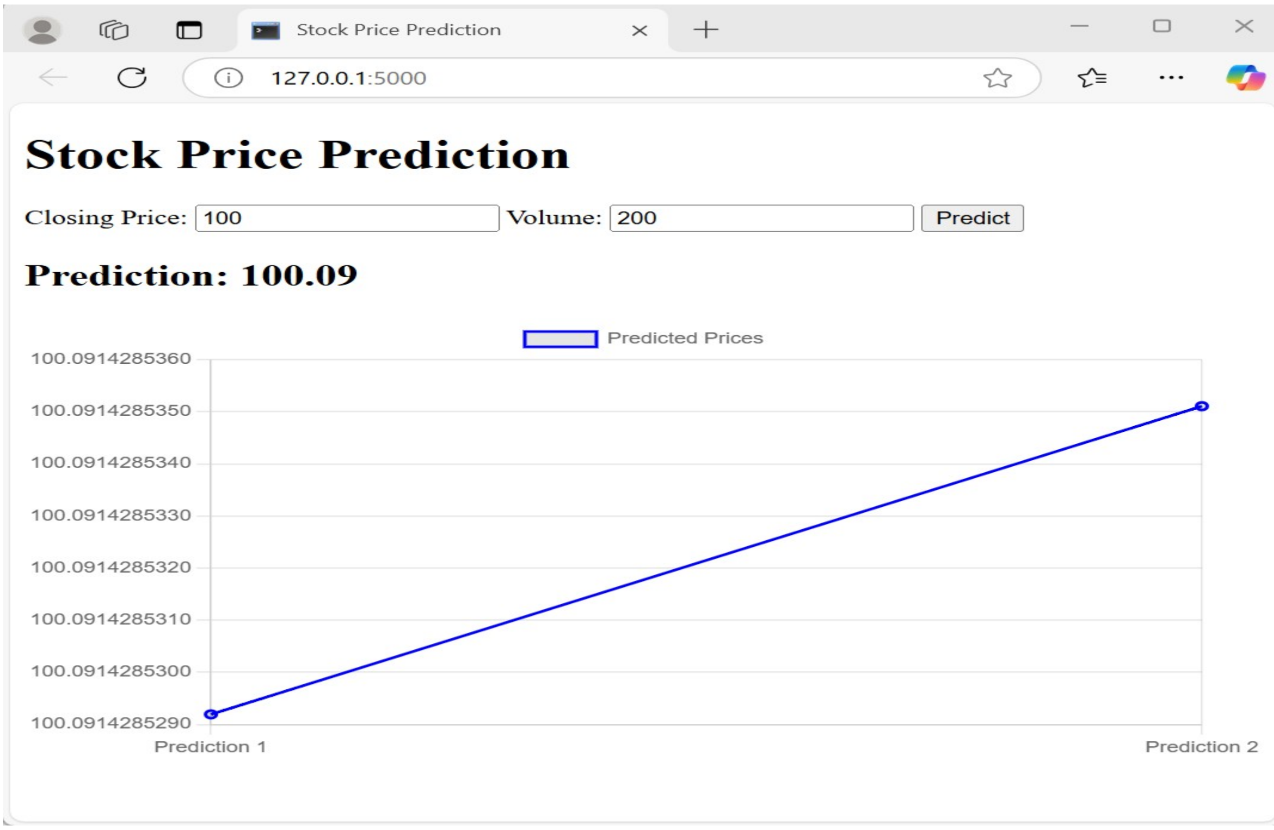
HTMLCSS code for uploading images and selecting models

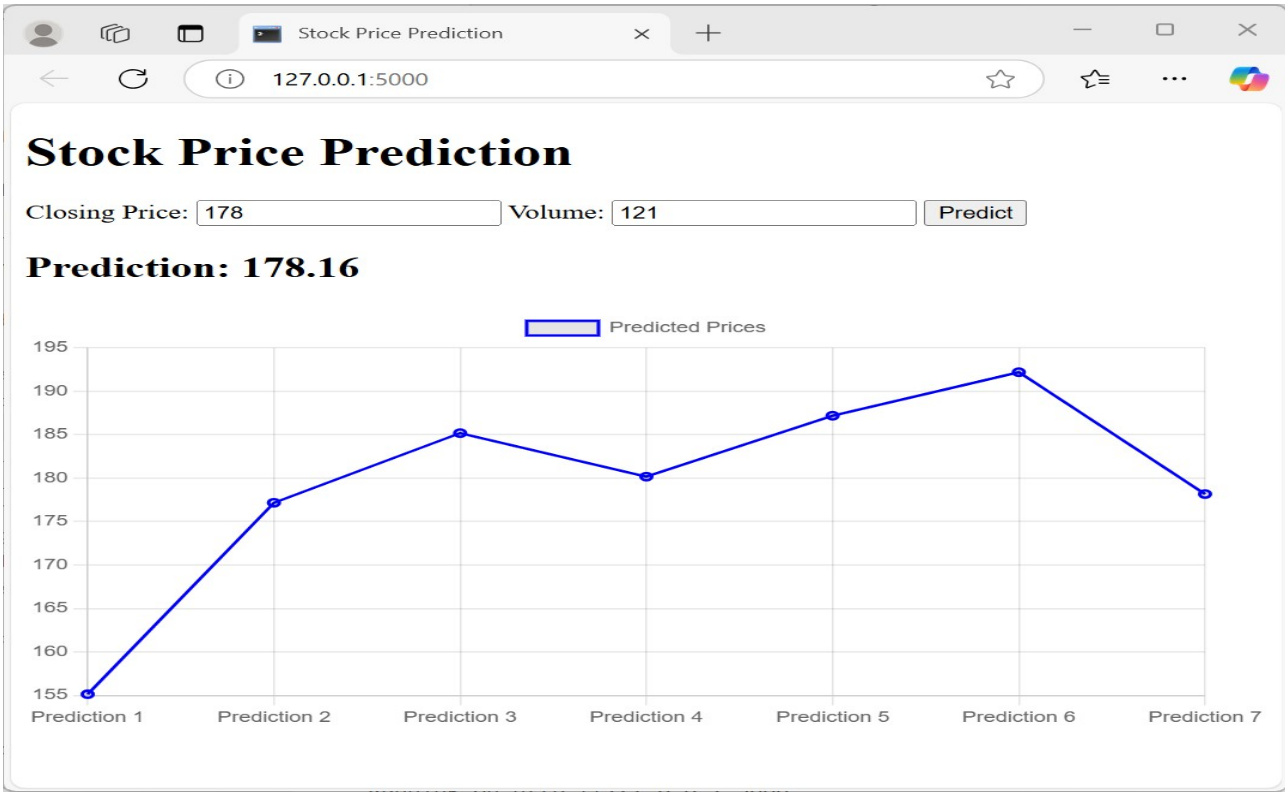
index.html

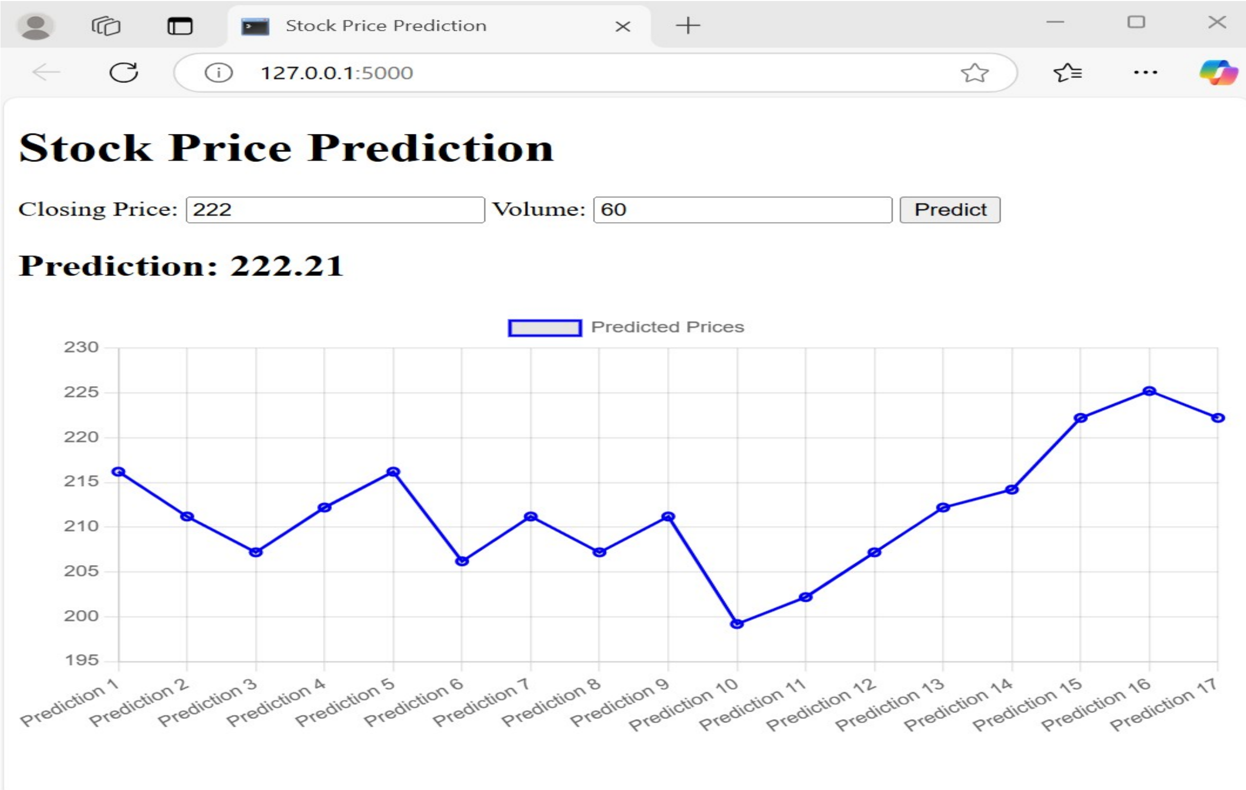
```
from flask import Flask, request, jsonify, render_template
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import joblib
import os

app = Flask(__name__)

def train_model():

    stock_prices = pd.read_csv(r"C:\Users\KIIT\OneDrive\Documents\AD
Lab\Lab3\historical_stock_prices.csv")
    stocks = pd.read_csv(r"C:\Users\KIIT\OneDrive\Documents\AD
Lab\Lab3\historical_stocks.csv")


    merged_data = pd.merge(stock_prices, stocks, on='ticker')
    stock_data = merged_data[merged_data['ticker'] == 'AAPL']
    stock_data['date'] = pd.to_datetime(stock_data['date'])
    stock_data = stock_data.sort_values(by='date')
    stock_data['Target'] = stock_data['close'].shift(-1)
    stock_data = stock_data.dropna()

    features = ['close', 'volume']
    X = stock_data[features]
    y = stock_data['Target']
```

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


    model = LinearRegression()
    model.fit(X_train, y_train)

    joblib.dump(model, 'linear_regression_model.pkl')
    return model

if os.path.exists('linear_regression_model.pkl'):
    model = joblib.load('linear_regression_model.pkl')
else:
    model = train_model()

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    try:

        data = request.get_json()
        close = float(data['close'])
        volume = float(data['volume'])


        prediction = model.predict([[close, volume]])[0]
        return jsonify({'predicted_price': prediction})
    except Exception as e:
        print(f"Prediction Error: {e}")
        return jsonify({'error': 'Prediction failed.'}), 400

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

4. Results/Output:-


Stock Price Prediction
127.0.0.1:5000
Stock Price Prediction
Closing Price: 100
Volume: 200
Predict
Prediction: 100.09
Predicted Prices
100.0914285360
100.0914285350
100.0914285340
100.0914285330
100.0914285320
100.0914285310
100.0914285300
100.0914285290
Prediction 1
Prediction 2


Stock Price Prediction
127.0.0.1:5000
Stock Price Prediction
Closing Price: 178
Volume: 121
Predict
Prediction: 178.16
Predicted Prices
195
190
185
180
175
170
165
160
155
Prediction 1
Prediction 2
Prediction 3
Prediction 4
Prediction 5
Prediction 6
Prediction 7



5. Remarks:-

The experiment demonstrated the use of Linear Regression and LSTM models for stock price prediction.Linear Regression served as a simple baseline, performing well for short-term predictions but struggled with capturing complex patterns.The Flask backend enabled real-time predictions, while the frontend visualization provided an intuitive way to analyze results. Overall, the LSTM model proved more effective for stock prediction, with Linear Regression offering a faster but less precise alternative.

Signature of the Student

____

(Rohit Kumar Satpathy)

Signature of the Lab Coordinator

____

(Name of the Coordinator)