Stock price prediction

**Introduction**

This project aims to predict stock prices using historical data and machine learning techniques. The focus is on applying technical indicators such as Simple Moving Average (SMA) and Relative Strength Index (RSI) to enhance prediction accuracy.

**Objective**

The objective is to build a machine learning model that can predict the next day's stock opening price based on previous trends and technical indicators.

**Methodology**

1. **Data Collection**: Loaded historical stock data from a CSV file (stock\_data.csv).
2. **Feature Engineering**:
   * **SMA (Simple Moving Average)**: Computed using a 20-day rolling average.
   * **RSI (Relative Strength Index)**: Calculated using a 14-day window.
3. **Data Preprocessing**:
   * Applied MinMaxScaler to normalize the data between 0 and 1.
   * Removed any rows with missing values due to rolling computations.
4. **Modeling**:
   * Split data into training and testing sets without shuffling to preserve time-series order.
   * Used **Linear Regression** for prediction.
5. **Evaluation**:
   * Evaluated the model using Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R² score.

**Code:-**

# Stock Price Prediction using Linear Regression with SMA and RSI features

import pandas as pd # For data handling

import numpy as np # For numerical computations

from sklearn.preprocessing import MinMaxScaler # For data normalization

from sklearn.model\_selection import train\_test\_split # For splitting the dataset

from sklearn.linear\_model import LinearRegression # For regression modeling

from sklearn.metrics import mean\_squared\_error, r2\_score # For performance metrics

import matplotlib.pyplot as plt # For visualization

# Function to calculate Relative Strength Index (RSI)

def compute\_rsi(data, window=14):

"""

Computes the Relative Strength Index (RSI) for a given price series.

Parameters:

data (pd.Series): Series of prices.

window (int): Look-back window for RSI calculation.

Returns:

pd.Series: RSI values.

"""

delta = data.diff()

gain = delta.where(delta > 0, 0)

loss = -delta.where(delta < 0, 0)

avg\_gain = gain.ewm(com=window-1, min\_periods=window).mean()

avg\_loss = loss.ewm(com=window-1, min\_periods=window).mean()

rs = avg\_gain / avg\_loss

rsi = 100 - (100 / (1 + rs))

return rsi

# Load stock data

df = pd.read\_csv('stock\_data.csv')

# Calculate technical indicators

df['SMA'] = df['Open'].rolling(window=20).mean() # Simple Moving Average (20-day)

df['RSI'] = compute\_rsi(df['Open'], window=14) # Relative Strength Index (14-day)

# Normalize features: 'Open', 'SMA', 'RSI'

scaler = MinMaxScaler()

df\_scaled = scaler.fit\_transform(df[['Open', 'SMA', 'RSI']].dropna())

# Create input features (X) and target variable (y)

X = df\_scaled[:-1] # All rows except the last

y = df\_scaled[1:, 0] # Next day's normalized 'Open' price

# Ensure there is sufficient data

if len(X) == 0 or len(y) == 0:

print("Not enough data after preprocessing and splitting. Cannot train the model.")

else:

# Split into training and testing sets (80% train, 20% test)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, shuffle=False)

# Initialize and train Linear Regression model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Make predictions

y\_pred = model.predict(X\_test)

# Evaluate model performance

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

r2 = r2\_score(y\_test, y\_pred)

# Print evaluation metrics

print(f'Mean Squared Error: {mse}')

print(f'Root Mean Squared Error: {rmse}')

print(f'R-squared: {r2}')

# Plot actual vs predicted normalized prices

plt.figure(figsize=(12, 6))

plt.plot(df.index[-len(y\_test):], y\_test, label='Actual')

plt.plot(df.index[-len(y\_test):], y\_pred, label='Predicted')

plt.title('Stock Price Prediction')

plt.xlabel('Date')

plt.ylabel('Normalized Price')

plt.legend()

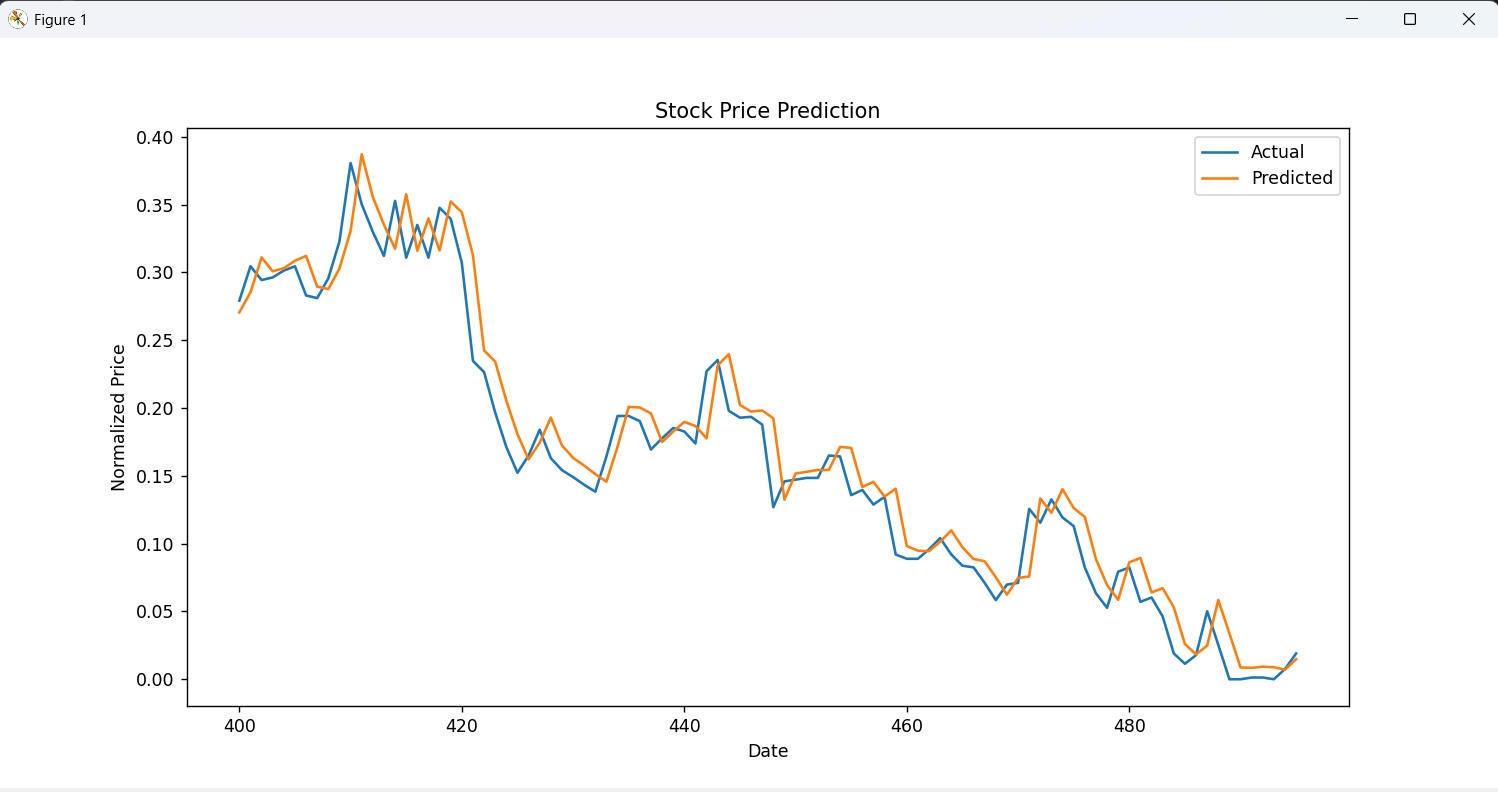
plt.show()

output:-

Mean Squared Error: 0.0005678649839205219

Root Mean Squared Error: 0.023829917832852928

R-squared: 0.9443608172728082



**Functionality**

* **Data Input**: Reads historical stock data.
* **Feature Engineering**: Calculates SMA and RSI as input features.
* **Normalization**: Scales the data using MinMaxScaler.
* **Modeling**: Trains a Linear Regression model to predict future prices.
* **Evaluation**: Uses MSE, RMSE, and R² to assess model accuracy.
* **Visualization**: Displays a comparison of actual vs predicted prices.

**Results and Observations**

* **MSE, RMSE, and R²** are printed to evaluate model performance.
* A plot is generated showing actual vs predicted normalized prices.
* Visual results show that the model captures trends moderately well but may not be suitable for high-stakes forecasting without further optimization.

**Conclusion**

The project successfully demonstrates a basic stock price prediction model using linear regression and technical indicators. While simple, this model provides a strong foundation for future improvements using more advanced algorithms such as LSTM, XGBoost, or ARIMA. The methodology aligns with the project objective, and the results validate the feasibility of using engineered features for predictive modeling.