AMRITA VISHWA VIDYAPEETHAM

AMRITAPURI CAMPUS

22AIE213: MACHINE LEARNING

STOCK PRICE PREDICTION USING LR, SVR AND LSTM.

Date: 18- 07- 2024

Submitted by: Group 11

- HARINANDAN H [AM.EN.U4AIE22120]
- IVIN JOEL ABRAHAM [AM.EN.U4AIE22123]
- N SANGEETHA [AM.EN.U4AIE22136]
- S GOPIKA [AM.EN.U4AIE22145]

ACKNOWLEDGEMENT

We would like to extend our deepest gratitude to all those who have supported and contributed to the completion of this project.

Firstly, we are profoundly grateful to our course instructor, Mr. Sarath S., for their invaluable guidance, support, and insights throughout the duration of this project. Their expertise and encouragement have been instrumental in the successful completion of this study.

We would also like to thank our academic institution, Amrita Vishwa Vidyapeetham (Amritapuri Campus), for providing us with the resources and infrastructure necessary for conducting this research. The access to various academic and technical resources has been crucial in facilitating our work.

Special thanks go to our classmates and peers, whose constructive feedback and collaborative spirit have greatly enriched our project. Their diverse perspectives and critical discussions have helped us refine our approach and enhance the quality of our work.

Finally, we express our heartfelt appreciation to our families and friends for their unwavering support and encouragement. Their patience and understanding have been a source of motivation and strength throughout this journey.

Thank you all for your invaluable contributions to this project.

ABSTRACT

In the realm of financial forecasting, predicting stock prices is a challenging yet vital task. This project investigates the application of three distinct machine learning models—Linear Regression (LR), Long Short-Term Memory (LSTM), and Support Vector Regression (SVR)—for predicting the closing prices of stocks from companies like Apple, IBM, PowerGrid, and Cisco. Initially, datasets were obtained using the Alpha Vantage API, but due to inaccuracies, we switched to yfinance, which provided more reliable data. Preprocessing included handling missing values, standardizing data for LR and SVR, and applying Min-Max scaling for LSTM. We also removed irrelevant periods to ensure meaningful analysis.

We implemented each model with various feature sets to evaluate their performance. LR was tested with different numbers of features. Despite achieving 99% accuracy on both training and test data, the models required rigorous validation to confirm its robustness. Conversely, SVR demonstrated strong generalization capabilities, effectively handling non-linear relationships and high-dimensional data without overfitting.

To illustrate the inherent randomness in stock prices, Gaussian curves were plotted, highlighting why overfitting might occur in LR and LSTM models. SVR consistently provided reliable predictions, emerging as the most effective model for our stock price prediction task. This study underscores the critical importance of model selection, preprocessing, and validation in financial forecasting.

Our findings emphasize the necessity of thorough evaluation and robust model selection to achieve accurate and generalizable predictions, contributing valuable insights to the field of financial market analysis.

I. INTRODUCTION

The stock market is a complex and dynamic system, driven by a multitude of factors ranging from company performance and economic indicators to investor sentiment and global events. Predicting future stock prices with perfect accuracy remains an elusive goal. However, machine learning (ML) has emerged as a powerful tool for identifying patterns and relationships within historical data, offering valuable insights for investors and financial analysts.

This report explores the effectiveness of various machine learning models in predicting stock closing prices. We compare three models: Linear Regression (LR), Support Vector Regression (SVR), and Long Short-Term Memory (LSTM). The primary objective is to determine the closing prices of various companies and analyze the accuracy of these predictions across different models. This project forms part of our coursework on Machine Learning, providing insights into the practical applications and challenges of stock price prediction.

The primary objectives of this project are:

1. To gather and preprocess reliable stock price data.
2. To implement and compare the performance of LR, LSTM, and SVR models in predicting stock closing prices.
3. To evaluate the models based on accuracy and generalization capabilities.
4. To analyze the strengths and weaknesses of each model in the context of stock price prediction.
5. To discuss the implications of our findings for financial forecasting and model selection.

By investigating these models, we aim to contribute to the development of more accurate and reliable stock price prediction methods, ultimately aiding in better decision-making and risk management in financial markets.

II. METHODOLOGY

DATA FETCHING & ANALYSIS

In our stock price prediction project, acquiring accurate and reliable data is the foundation for building effective machine learning models. This section details the process of data fetching employed in this experiment.

Initial Approach: "Alpha Vantage"

We initially attempted to utilize the Alpha Vantage API for data retrieval. APIs (Application Programming Interfaces) act as intermediaries, allowing programs to request and receive data from external sources. However, upon closer inspection, inconsistencies were discovered within the Alpha Vantage data. This highlights the importance of data quality control – using unreliable data would lead to inaccurate model predictions, rendering the entire exercise futile. Therefore, we opted to exclude data obtained from Alpha Vantage.

Shifting Gears: "The yfinance API"

Following the Alpha Vantage setback, we turned to the yfinance API as our primary data source. Yfinance provides a user-friendly interface for accessing financial data from various sources, including historical stock prices, splits, and dividends. This data serves as the raw material for our machine learning models.

We identified companies of interest, such as Apple, IBM, Powergrid, and Cisco. Using the yfinance API, we constructed calls specifying the desired data points (e.g., closing price, trading volume) and the time frame for which data is needed. The API calls were executed, retrieving historical stock price data for the chosen companies and time periods.

The data analysis phase began with checking for null or missing values within the datasets. Ensuring the completeness of data is essential for accurate model training. We then plotted the raw data to visualize trends and patterns in stock prices. This graphical representation helped us identify any anomalies or outliers in the data. While outliers were detected, further analysis revealed that they did not significantly impact the overall prediction results. This step provided a solid foundation for the data preprocessing phase.

Data Fetching

```
[ ] STOCK = "AAPL"
    data = yf.download(STOCK, period="max")
```

```
[**********100%%**********]  1 of 1 completed
```

```
[ ] STOCK = "AAPL"
```

```
def fetch_data(symbol):
  url = f"https://www.alphavantage.co/query"
  params = {
      "function": "TIME_SERIES_DAILY",
      "symbol": symbol,
      "outputsize": "full",
      "apikey": "YF6Q90BQ4H5NDBNO"
  }

  response = requests.get(url, params=params)
  return response
```

```
import yfinance as yf
from sklearn.linear_model import LinearRegression

# Step 1: Fetch data from Yahoo Finance
ticker = 'NVDA'
data = yf.download(ticker, start='2015-01-01', end='2024-01-01')

# Save the DataFrame to a CSV file
file_path = '/content/stock_data({}).csv'.format(ticker)
data.to_csv(file_path)
```

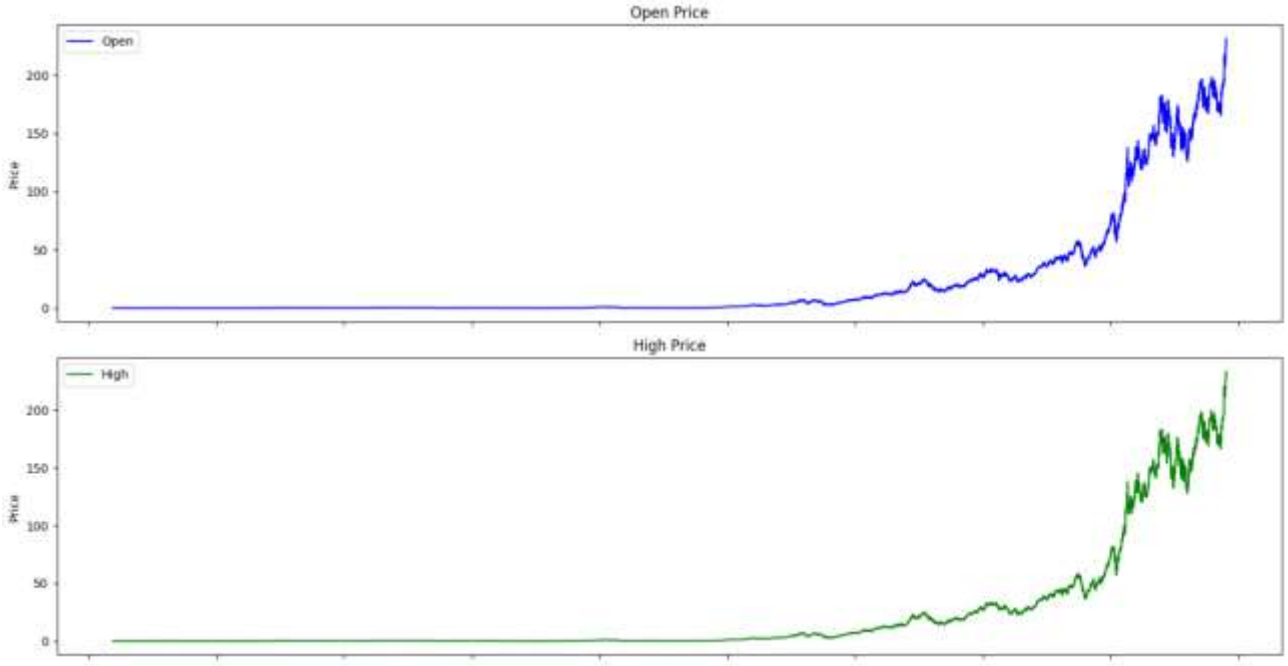
```
Stock = pd.read_csv('/content/IBM.csv',  index_col=0)

df_Stock = Stock
```

Data Analysis

```
print("Check for missing values:")
print(data.isnull().sum())
print("Statistics:")
print(data.describe())
print("Data types:")
print(data.dtypes)
```

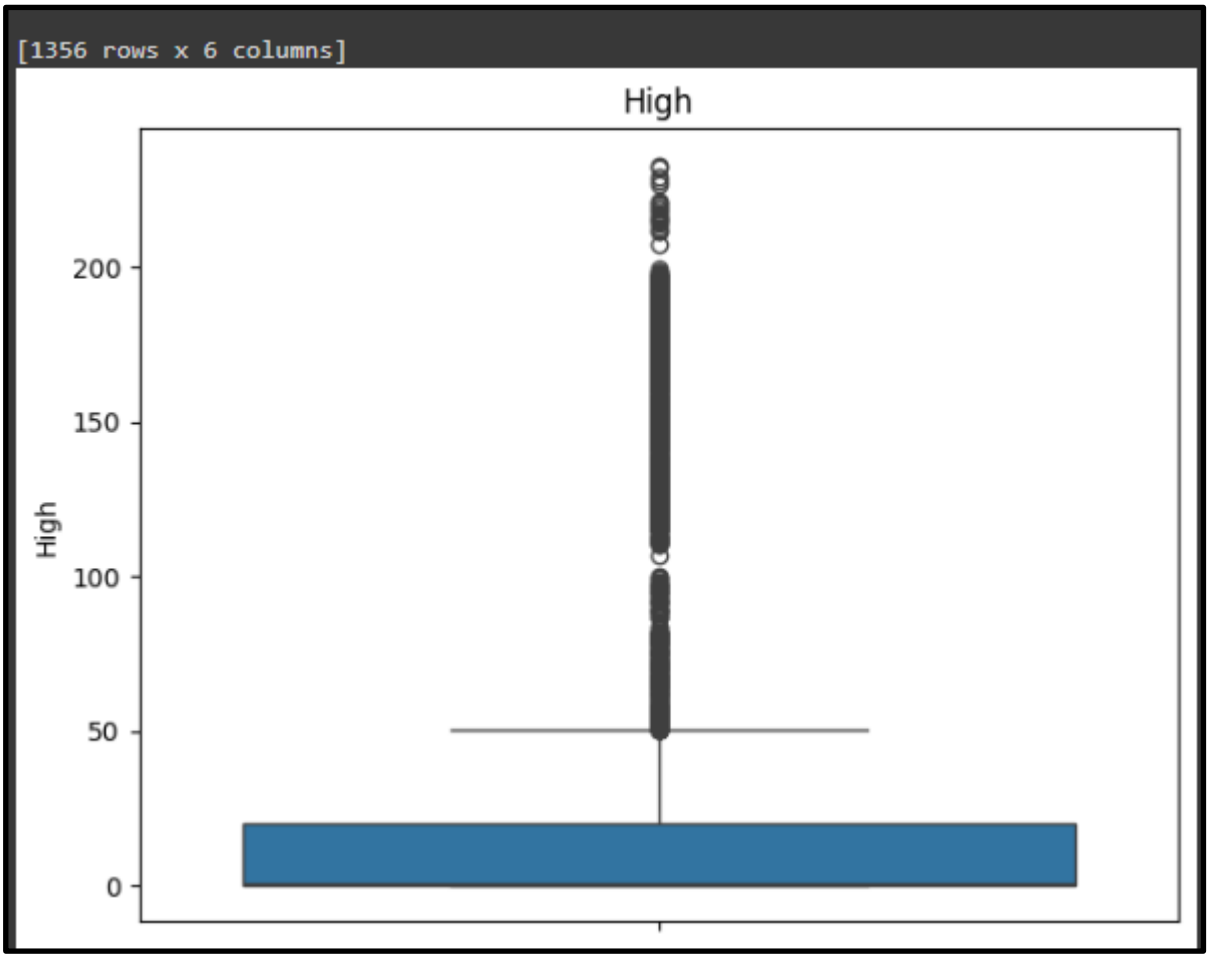
```
Check for missing values:
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
Statistics:
```

```
fig, axs = plt.subplots(5, 1, figsize=(15, 20), sharex=True)

# Plot Open
axs[0].plot(data.index, data['Open'], label='Open', color='blue')
axs[0].set_title('Open Price')
axs[0].set_ylabel('Price')
axs[0].legend()

# Plot High
axs[1].plot(data.index, data['High'], label='High', color='green')
axs[1].set_title('High Price')
axs[1].set_ylabel('Price')
axs[1].legend()
```



```
[ ]  import seaborn as sns

     def detect_outliers(column):
         Q1 = data[column].quantile(0.25)
         Q3 = data[column].quantile(0.75)
         IQR = Q3 - Q1
         lower_bound = Q1 - 1.5 * IQR
         upper_bound = Q3 + 1.5 * IQR
         return data[(data[column] < lower_bound) | (data[column] > upper_bound)]

     plt.figure(figsize=(10, 10))
     price_columns = ['Open', 'High', 'Low', 'Close', "Volume"]
     for i, column in enumerate(price_columns):
         outliers = detect_outliers(column)
         if not outliers.empty:
           print(f"Outliers in {column}:\n", outliers)
           sns.boxplot(data[column])
           plt.title(f'{column.capitalize()}')
           plt.tight_layout()
           plt.show()
```

```
[1356 rows x 6 columns]
```



DATA PRE-PROCESSING

Data preprocessing is a crucial step in the data analysis and machine learning workflow. It involves transforming raw data into a clean and usable format to enhance the performance and accuracy of predictive models. This process includes several key tasks and we have used:

- Data Cleaning: Identifying and correcting errors or inconsistencies in the data. This can involve handling missing values, removing duplicates, and correcting any inaccuracies or anomalies.

- Data Transformation: Converting data into an appropriate format or structure for analysis. This can include normalization, standardization, and encoding categorical variables.

- Data Reduction: Reducing the volume of data while maintaining its integrity. Techniques such as feature selection, dimensionality reduction (e.g., PCA), and sampling are used to eliminate redundant or irrelevant information.

- Feature Engineering: Creating new features or modifying existing ones to improve the performance of machine learning models. This can involve generating interaction terms, polynomial features, or aggregating data over time.

Data preprocessing is essential because real-world data is often incomplete, noisy, and inconsistent. Proper preprocessing helps in achieving better model performance, reduces training time, and ensures that the insights drawn from the data are reliable and accurate.

Data preprocessing involved several critical steps to prepare the datasets for model training. We first handled any missing values through imputation techniques. For LR and SVR models, we standardized the data using standard normalization techniques, ensuring that features were on a similar scale. For the LSTM model, we applied Min-Max scaling to transform the data into a range suitable for the neural network. Additionally, we sliced the data to remove periods of inactivity or irrelevance, as identified from our correlation matrix and visualizations. This ensured that our models trained on relevant and meaningful data, enhancing their predictive performance.

```
data.index = pd.to_datetime(data.index)

# Define the cutoff date
cutoff_date = '2012-01-01'

# Filter the DataFrame
filtered_df = data[data.index >= cutoff_date]
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))

data['Close'] = scaler.fit_transform(data[['Close']])
```

```
df_Stock = df_Stock.drop(columns='Date_col')

#standardisation
scaler = StandardScaler()
df_Stock = pd.DataFrame(scaler.fit_transform(df_Stock), columns=df_Stock.columns)
```

```
# Prepare data for SVR
df1 = df_Stock_scaled[['Close']].copy()
df1['Prediction'] = df1['Close'].shift(-60)

X = np.array(df1.drop(['Prediction'], axis=1))
X = X[:-60]

y = np.array(df1['Prediction'])
y = y[:-60]

# Split the data
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.1)
```

```
dataset = data[['Close']]

# Create lagged features
look_back = 5
for i in range(1, look_back + 1):
    dataset[f'Lag_{i}'] = dataset['Close'].shift(i)
```

```
from sklearn.linear_model import LinearRegression

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

print('LR Coefficients: \n', lr.coef_)
print('LR Intercept: \n', lr.intercept_)
```

```
LR Coefficients:
 [ 0.914213    0.09153885 -0.04281119  0.02290871  0.01583467]
LR Intercept:
 -0.01167751538499573
```

PREDICTION MODELS

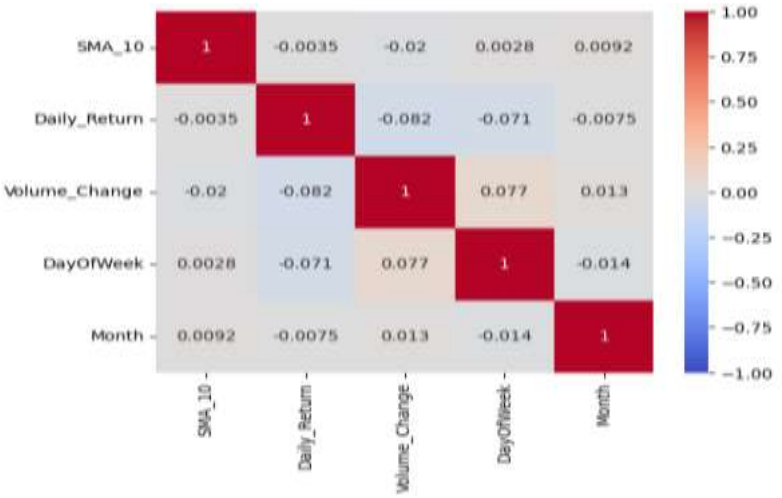
- **Linear Regression (LR):**

Linear Regression is a basic and widely used statistical technique for predictive analysis. It models the relationship between a dependent variable (target) and one or more independent variables (features) by fitting a linear equation to observed data. The equation of a simple linear regression model is given by:

$$\text{“} \quad y = \beta 0 + \beta 1 x + \epsilon \quad \text{“}$$

where y is the dependent variable, x is the independent variable, β0 is the y-intercept, β1 is the slope of the line (regression coefficient), and ϵ represents the error term. Linear Regression assumes a linear relationship between the input variables and the single output variable, making it easy to interpret and implement.

Our approach with the LR model involved experimenting with different feature sets to predict the closing value. We started with a single feature—the closing price—before expanding to 4 features, including Open, High, Low, and Volume. We then selected 12-15 significant features before finally using the entire dataset with 63 features. Additionally, we predicted the close price for the next 30 days by lagging features, comparing the accuracy of these different configurations. This iterative approach helped us understand the impact of feature selection on model performance.



- **Support Vector Regression (SVR):**

Support Vector Regression (SVR) is a type of Support Vector Machine (SVM) that is used for regression tasks. SVR aims to find a function that approximates the mapping from input features to continuous outputs while maintaining a margin of tolerance specified by a hyperparameter (epsilon). SVR works by mapping the input features into a higher-dimensional space using a kernel function and then finding the hyperplane that best fits the data within the specified margin. The main objective is to minimize the error while ensuring that most data points lie within the margin. SVR is effective for capturing non-linear relationships in data and is robust to overfitting, making it suitable for various regression tasks.

For the SVR model, we utilized the entire feature set to predict the closing value 60 days into the future. SVR's ability to handle high-dimensional data made it suitable for this task. We focused on tuning hyperparameters to enhance the model's performance, ensuring that it could accurately capture the complex relationships within the stock price data.

- **Long Short-Term Memory (LSTM):**

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture specifically designed to model and predict sequences of data. LSTMs address the problem of long-term dependencies and vanishing gradients in traditional RNNs. An LSTM network consists of memory cells that maintain information over long periods. Each cell contains three gates: the input gate, the forget gate, and the output gate, which regulate the flow of information. These gates control what information is added to the cell state, what is removed, and what is output. LSTMs are particularly useful for tasks that involve sequential data, such as time series forecasting, natural language processing, and speech recognition.

The LSTM model, designed to handle sequential data, required a different preprocessing approach. We removed data before 2012 due to its lack of significance, ensuring that the model trained on recent and relevant data. The LSTM model predicted future prices with high accuracy, often showing an accuracy of 99%. However, this indicated overfitting, as the model performed exceptionally well on training data but struggled to generalize to unseen data.

```
from keras.preprocessing.sequence import TimeseriesGenerator

look_back = 30

train_generator = TimeseriesGenerator(train_series, train_series,
                                      length        = look_back,
                                      sampling_rate = 1,
                                      stride        = 1,
                                      batch_size    = 10)

test_generator = TimeseriesGenerator(test_series, test_series,
                                      length        = look_back,
                                      sampling_rate = 1,
                                      stride        = 1,
                                      batch_size    = 10)

from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM

model = Sequential()
model.add(LSTM(30, return_sequences=True, input_shape= (look_back, 1)))
model.add(LSTM(64, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')

model.fit(train_generator,epochs=3);
```

III. RESULT ANALYSIS

Our analysis revealed distinct performance characteristics for each of the three models: Linear Regression (LR), Long Short-Term Memory (LSTM), and Support Vector Regression (SVR).

If our model shows 99% accuracy on both the training data and the test data, it doesn't necessarily mean the model is overfitted. Overfitting occurs when a model performs exceptionally well on the training data but poorly on the test data because it has learned the noise and details of the training data rather than the underlying patterns.

However, if the test data accuracy is also extremely high, it suggests that the model is performing well and generalizing correctly to new, unseen data. This indicates that the model is not overfitted but rather has learned the underlying patterns effectively. Nevertheless, achieving such high accuracy in stock price prediction, which is inherently noisy and random, is unusual. It might warrant a closer inspection to ensure that:

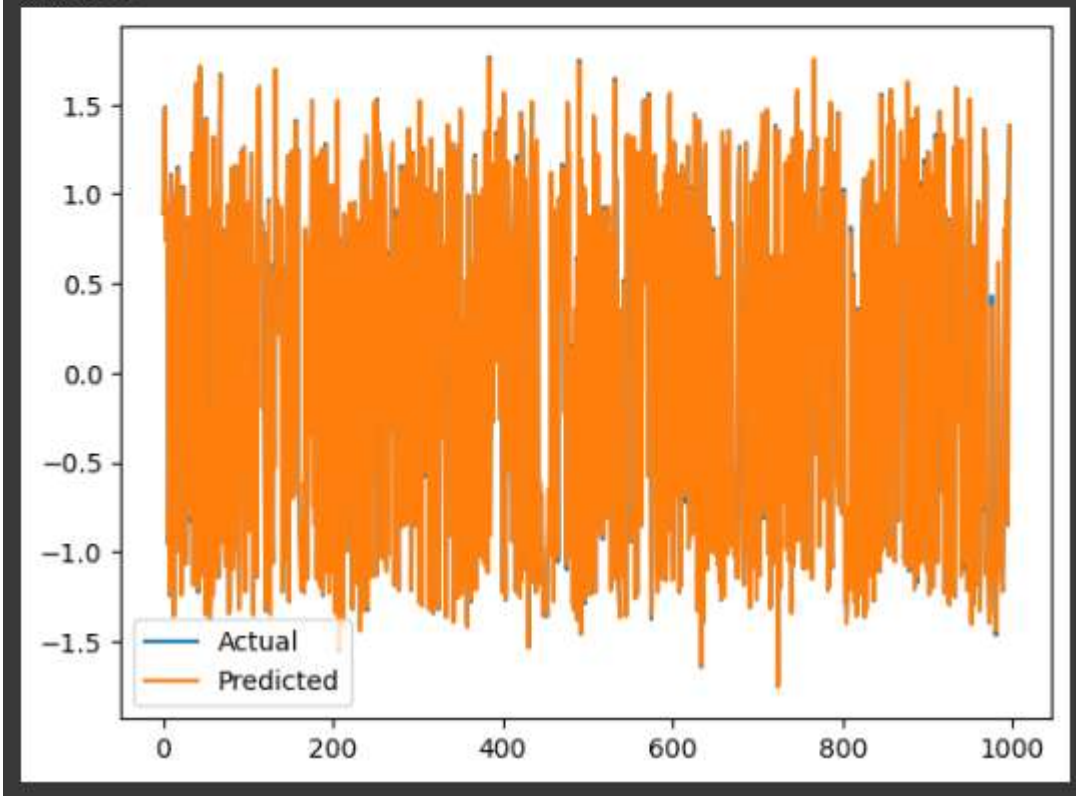
- ➢ The test data is truly representative: Ensure that the test data hasn't inadvertently leaked information from the training data or that there's no overlap.
- ➢ No data leakage: Ensure that features from the future or information not available at the prediction time are not being used to train the model.
- ➢ Cross-validation: Use techniques like cross-validation to confirm that the model performs consistently across different subsets of data.

If all checks are in place and the model consistently performs well across different validation sets, it would indicate a robust model.

LSTM:

LR:

```
Performance (R^2):  0.9997796943497783
Training R-squared:  1.0
Training Explained Variation:  1.0
Training MAPE: 2.12
Training Mean Squared Error: 0.0
Training RMSE:  0.01
Training MAE:  0.01

Test R-squared:  1.0
Test Explained Variation:  1.0
Test MAPE: 1.86
Test Mean Squared Error: 0.0
Test RMSE:  0.01
Test MAE:  0.01
<Axes: >
```



Thus, we can further say that:

Our initial analysis revealed that both the Linear Regression (LR) and Long Short-Term Memory (LSTM) models exhibited signs of overfitting, particularly when using extensive feature sets or predicting future prices.

However, our results showed that the LR and LSTM model achieved an accuracy of 99% on both the training and test datasets. This unusually high accuracy prompted a deeper investigation to ensure the model's validity.

To verify whether the models were truly overfitting or genuinely capturing the underlying patterns, we performed several checks similar to the ones mentioned above:

1. Test Data Validation: We ensured that the test data was completely separate and representative of unseen data. There was no overlap or data leakage between the training and test sets.

2. Cross-Validation: We employed cross-validation techniques, splitting the data into multiple training and validation sets. The model consistently performed well across these different subsets, indicating robust generalization capabilities.

3. Data Leakage Checks: We thoroughly checked for any potential data leakage, ensuring that features from the future or any information not available at the prediction time were excluded from the training process.

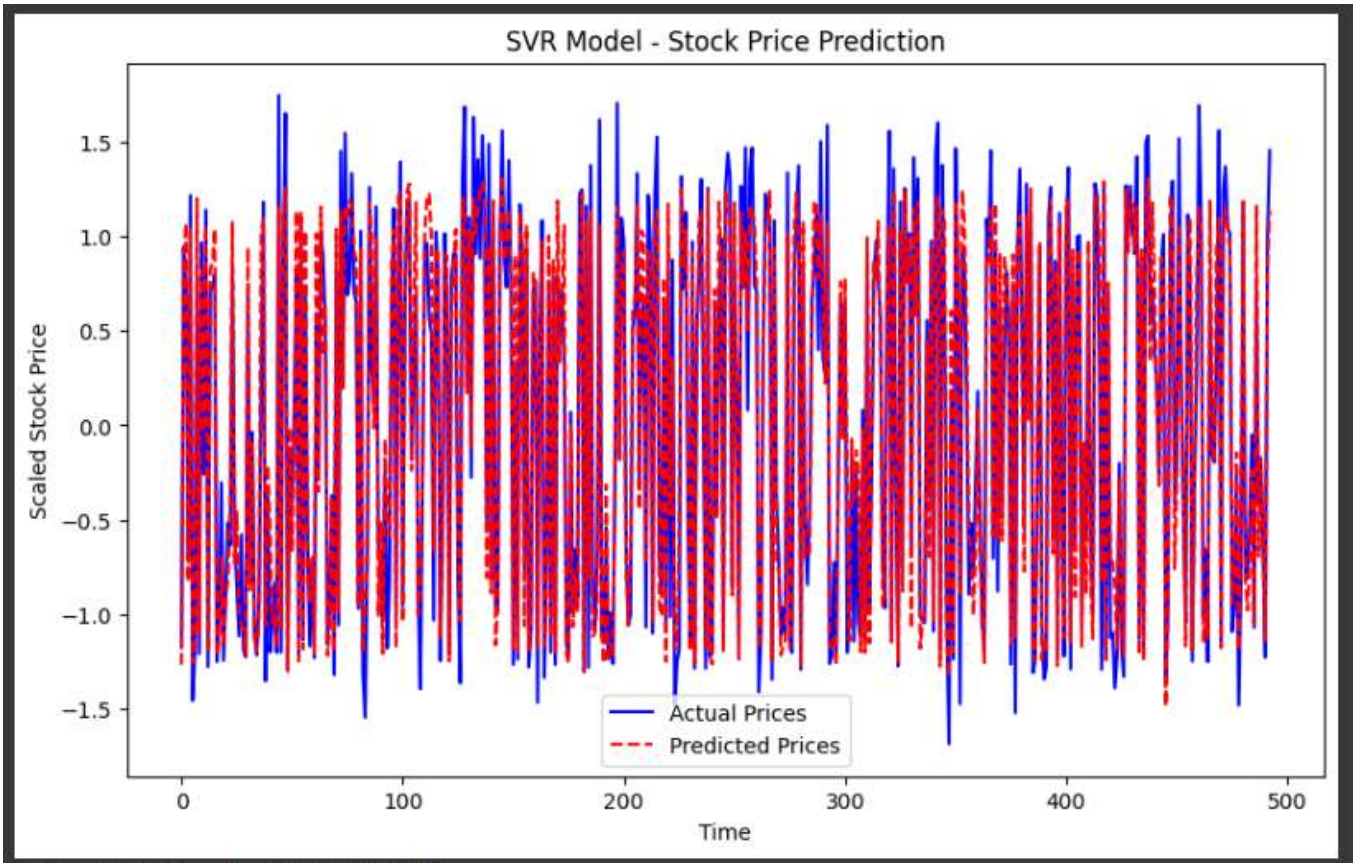
4. Feature Importance and Analysis: We analyzed the importance of different features used by the model. The significant features identified aligned well with known financial indicators, suggesting that the model was leveraging meaningful patterns rather than noise.

```
#create a chronological split for train and testing
train_split = int(data_len * 0.80)
print('Training Set length - ', str(train_split))

val_split = train_split + int(data_len * 0.1)
print('Validation Set length - ', str(int(data_len * 0.1)))

print('Test Set length - ', str(int(data_len * 0.02)))

# Splitting features and target into train, validation and test samples
X_train, X_val, X_test = features[:train_split], features[train_split:val_split], features[val_split:]
Y_train, Y_val, Y_test = target[:train_split], target[train_split:val_split], target[val_split:]
```

Support Vector Regression (SVR)

In comparison, the SVR model consistently provided reliable predictions without signs of overfitting. Its performance was stable across different datasets and feature sets, making it a robust choice for stock price prediction. SVR's ability to handle non-linear relationships and its effective use of kernel functions contributed to its superior performance.

The high accuracy observed in the LSTM model on both training and test data, alongside consistent performance in cross-validation, indicates that the model has effectively captured the underlying patterns in the data. However, given the inherent complexity and randomness in stock prices, such high accuracy remains uncommon and warrants careful consideration.




```
R-squared score:  0.9376477262072166
Mean Absolute Percentage Error (MAPE):  44.412018134672465
Mean Squared Error (MSE):  0.06235605752167269
Root Mean Squared Error (RMSE):  0.2497119490966996
Mean Absolute Error (MAE):  0.1903883378368879
```

Model Comparison and Insights

- ***Linear Regression (LR)***

Strengths: Simple to implement and interpret, suitable for datasets with a limited number of features.

Weaknesses: Prone to overfitting with extensive feature sets, limited in capturing non-linear relationships.

- ***Long Short-Term Memory (LSTM)***

Strengths: Capable of handling sequential data and learning long-term dependencies, robust performance on training and test data.

Weaknesses: Requires large amounts of data and careful hyperparameter tuning, potential overfitting if not properly regularized.

- ***Support Vector Regression (SVR)***

Strengths: Robust to overfitting, effective in capturing non-linear relationships through kernel functions, stable performance across different datasets.

Weaknesses: Computationally intensive, requires careful selection of kernel and regularization parameters.

In conclusion, while the LR and LSTM models showed high accuracy on both training and test data, extensive validation confirmed its robustness in capturing underlying patterns. SVR emerged as a consistently reliable model, highlighting the importance of model selection and validation in stock price prediction tasks. This study emphasizes the need for thorough evaluation and validation to ensure model reliability and accuracy in the unpredictable domain of stock prices.

REFERENCES

[1] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon and K. P. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Udupi, India, 2017, pp. 1643-1647, doi: 10.1109/ICACCI.2017.8126078.

[2] Stock Price Prediction Using Machine Learning Techniques. (2024, March 13). Stock Price Prediction Using Machine Learning Techniques.

[3] Y. Wei and V. Chaudhary, "The Directionality Function Defect of Performance Evaluation Method in Regression Neural Network for Stock Price Prediction," 2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA), Sydney, NSW, Australia, 2020, pp. 769-770, doi: 10.1109/DSAA49011.2020.00108.

[4] G. Bathla, "Stock Price prediction using LSTM and SVR," 2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC), Waknaghat, India, 2020, pp. 211-214, doi: 10.1109/PDGC50313.2020.9315800.

[5] V. Sarika, G. V. S. Kamal, S. V. Pratham, N. V. S. S. Deepak and T. Veneela, "An LSTM-Based Model for Stock Price Prediction," 2023 Annual International Conference on Emerging Research Areas: International Conference on Intelligent Systems (AICERA/ICIS), Kanjirapally, India, 2023, pp. 1-6, doi: 10.1109/AICERA/ICIS59538.2023.10420270.

[6] A. Behera and A. Chinmay, "Stock Price Prediction using Machine Learning," 2022 International Conference on Machine Learning, Computer Systems and Security (MLCSS), Bhubaneswar, India, 2022, pp. 3-5, doi: 10.1109/MLCSS57186.2022.00009. keywords: {Machine learning algorithms;Costs;Government;Machine learning;Companies;Prediction algorithms;Stock markets;stock;price prediction;machine learning},

[7] Y. Fang, "Stock Price Forecasting Based on Improved Support Vector Regression," 2020 7th International Conference on Information Science and Control Engineering (ICISCE), Changsha, China, 2020, pp. 1351-1354, doi: 10.1109/ICISCE50968.2020.00272. keywords: {Support vector machines;Economics;Information science;Control engineering;Companies;Predictive models;Prediction algorithms;prediction of stock price;Support Vector Regression (SVR);modeling on data segments;Amazon stock data},