Stock Price prediction

Phase 2

TEAM: TG-06

TEAM LEADER:

- SUPRIYA.A

TEAM MEMBERS:

- PRIYADHARSHINI.Y
- GETSY JACINTH.S
- SRINIDHI.E
- SARANYA.C

Dataset Link:

https://www.kaggle.com/datasets/prasoonkottarathil/microsoft-lifetime-stocks-dataset

Steps involved in Predicting this stock price are:

Data Collection and Preprocessing:

- Gather historical stock price data. This data should include relevant features like open price, close price, volume, and any other indicators you plan to use.
- Clean the data to handle missing values, outliers, and any inconsistencies. You may need to normalize or standardize the data.

Feature Engineering:

- Create new features or transform existing ones to improve model performance. You can include technical indicators like Moving Averages, Relative Strength Index (RSI), and Bollinger Bands.
- Consider including external data like news sentiment or macroeconomic indicators if they can impact stock prices.

Data Splitting:

Divide your dataset into training, validation, and testing sets. A common split might be 70% for training, 15% for validation, and 15% for testing. **Model Selection and Design Innovation:**

Revisit your LSTM model architecture. Innovation in this step might involve experimenting with:

- Different LSTM variants (e.g., Bidirectional LSTM, stacked LSTM).
- Attention mechanisms to focus on important time steps or features.
- Incorporating other types of neural networks like CNNs for feature extraction.
- Hybrid models that combine LSTMs with other architectures like Transformer models.

- Reinforcement Learning for dynamic trading strategies.

Hyperparameter Tuning:

Optimize hyperparameters such as learning rate, batch size, number of hidden units, and dropout rates using techniques like grid search or random search.

Training:

- Train the LSTM model on the training data using the selected architecture and hyperparameters.
- Implement early stopping and model checkpointing to prevent overfitting and save the best model.

Validation:

- Evaluate the model's performance on the validation set, using appropriate evaluation metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE).
- Continuously monitor validation metrics during training to detect overfitting or underfitting.

Model Interpretability:

Innovate by incorporating interpretability techniques like SHAP (SHapley Additive exPlanations) values or LIME (Local Interpretable Model-agnostic Explanations) to understand how the model makes predictions.

Back testing and Simulation:

- Test your model's performance on historical data by implementing a back testing framework.
- Simulate trading strategies based on model predictions to assess their profitability.

Ensemble Models:

Combine multiple models, such as LSTM, CNN, or other models you've experimented with, using techniques like stacking or blending, to improve prediction accuracy and robustness.

Deployment:

- Once you have a well-performing model, deploy it in a production environment. This might involve containerizing your model using Docker and deploying it on a cloud platform like AWS, GCP, or Azure.
- Set up a data pipeline to feed new data to the model and automate the prediction process.

Monitoring and Maintenance:

- Continuously monitor the deployed model's performance in real-time.
- Implement model retraining strategies to keep it up-to-date with the latest data. *Feedback Loop:*

Innovate by implementing a feedback loop where the model learns from its predictions and user feedback, potentially improving its performance over time.

Security and Compliance:

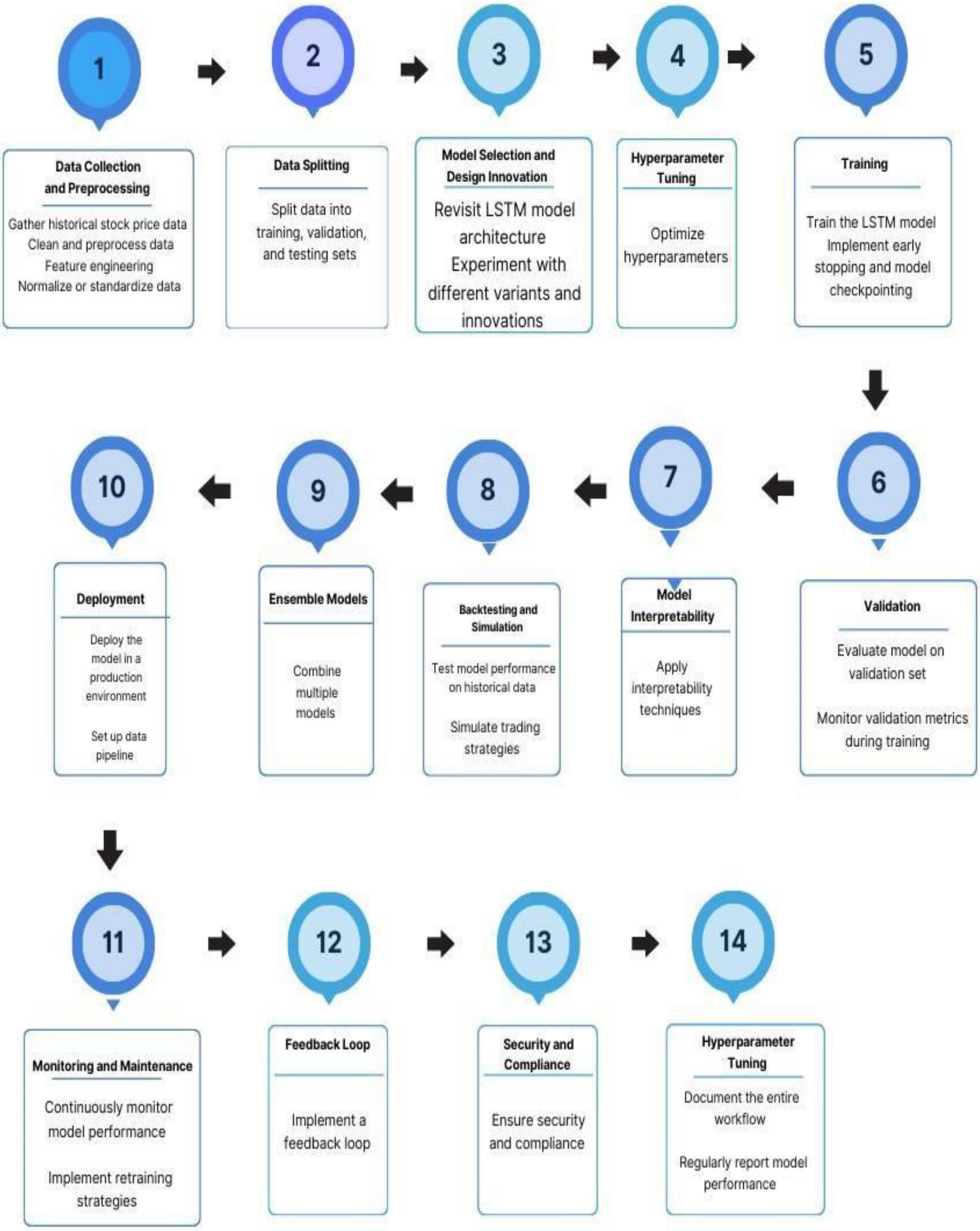
Ensure that your deployed model complies with security and privacy standards, especially if handling sensitive financial data.

Documentation and Reporting:

- Document your entire workflow, including data sources, preprocessing steps, model architecture, and deployment process.
- Regularly report the model's performance and insights to stakeholders.

The workflow of this predictive model :

WorkFlow


1
Data Collection and Preprocessing
Gather historical stock price data
Clean and preprocess data
Feature engineering
Normalize or standardize data
2
Data Splitting
Split data into training, validation, and testing sets
3
Model Selection and Design Innovation
Revisit LSTM model architecture
Experiment with different variants and innovations
4
Hyperparameter Tuning
Optimize hyperparameters
5
Training
Train the LSTM model
Implement early stopping and model checkpointing
6
Validation
Evaluate model on validation set
Monitor validation metrics during training
7
Model Interpretability
Apply interpretability techniques
8
Backtesting and Simulation
Test model performance on historical data
Simulate trading strategies
9
Ensemble Models
Combine multiple models
10
Deployment
Deploy the model in a production environment
Set up data pipeline
11
Monitoring and Maintenance
Continuously monitor model performance
Implement retraining strategies
12
Feedback Loop
Implement a feedback loop
13
Security and Compliance
Ensure security and compliance
14
Hyperparameter Tuning
Document the entire workflow
Regularly report model performance

For this Project We have taken LSTM as our prediction model for stock analysis

<u>To state why to choose LSTM model over ARIMA for this prediction</u>

- ✞ unlike ARIMA and Prophet, are not restricted to time series. The main idea behind LSTM cells is to learn the important parts of the sequence seen so far and forget the less important ones.

- ✞ The LSTM model provides better results when the data set is large and has fewer Nan values. Whereas, despite providing better accuracy than LSTM, the ARIMA model requires more time in terms of processing and works well when all the attributes of the data set provide legitimate values.

- ✞ LSTM works better if we are dealing with huge amount of data and enough training data is available, while ARIMA is better for smaller datasets , ARIMA requires a series of parameters (p,q,d) which must be calculated based on data, while LSTM does not require setting such parameters.

- ✟ Innovative approaches to solving stock price prediction often involve advanced machine learning and data analysis techniques. Below, I'll provide you with an example code implementing one such innovative approach using Long ShortTerm Memory (LSTM) networks, a type of recurrent neural network (RNN) known for its ability to capture sequential dependencies in time series data. We'll also use Python and the Keras library for this example.

- ✟ Please note that while this example is innovative and incorporates deep learning, it's important to understand that predicting stock prices remains a challenging and uncertain task due to the many factors that influence market behavior.

Libraries can be installed in terminal by this commands:

- To install numpy ,pandas,matplotlib:

  Pip install numpy or pandas or matplotlib

- This goes same for sklearn and keras

Pip install keras

Pip install sklearn

- The MinMax scalar package is preinstalled in sklearn.preprocessing
- For running the model prediction LSTM it is already installed in keras.layers
- We have added Mean Squared Error Model*MSE+ for the LSTM model inorder to find the loss.

Code for lstm: import numpy

as np import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential from keras.layers

import LSTM, Dense, Dropout

Load historical stock price data (e.g., CSV file with 'Date' and 'Close' columns) data = pd.read_csv('MSFT.csv')

Extract the 'Close' prices as the target variable prices

```
= data*'Close'+.values.reshape(-1, 1)

# Normalize the data using Min-Max scaling scaler
= MinMaxScaler(feature_range=(0, 1)) prices_scaled
= scaler.fit_transform(prices)

# Define a function to create sequences of data for training
the LSTM model  def create_sequences(data, seq_length):
   X, y = *+, *+     for i in
range(len(data) - seq_length):
X.append(data*i:i+seq_length+)
      y.append(data*i+seq_length+)
return np.array(X), np.array(y)

# Set the sequence length and split the data into training and
testing sets sequence_length = 10
X, y = create_sequences(prices_scaled, sequence_length) train_size
= int(len(X) * 0.8)
X_train, X_test = X*:train_size+, X*train_size:+ y_train,
y_test = y*:train_size+, y*train_size:+  # Create an
LSTM model model
= Sequential()
```

```
model.add(LSTM(units=50, return_sequences=True,
input_shape=(X_train.shape*1+, 1)))
model.add(LSTM(units=50)) model.add(Dense(1))

# Compile the model model.compile(optimizer='adam',
loss='mean_squared_error')

# Train the model

model.fit(X_train, y_train, epochs=50, batch_size=64)

# Make predictions on the test set predictions

= model.predict(X_test)

# Inverse transform the predictions to get actual price values

predictions_actual = scaler.inverse_transform(predictions)

y_test_actual = scaler.inverse_transform(y_test)

# Plot the actual vs. predicted prices plt.figure(figsize=(12,

6))

plt.plot(predictions_actual, label='Predicted Prices', color='red')

plt.plot(y_test_actual, label='Actual Prices', color='blue') plt.title('Stock

Price Prediction with LSTM') plt.xlabel('Time') plt.ylabel('Price')

plt.legend() plt.show()
```

Output:

Epoch 1/50

107/107 *================================+ - 7s
20ms/step - loss: 0.0011

Epoch 2/50

107/107 *================================+ - 2s
19ms/step - loss: 3.8592e-05

Epoch 3/50

107/107 *================================+ - 2s
20ms/step - loss: 3.8419e-05

Epoch 4/50

107/107 *================================+ - 2s
14ms/step - loss: 3.7421e-05

Epoch 5/50

107/107 *================================+ - 1s
13ms/step - loss: 3.6841e-05

Epoch 6/50

107/107 *================================+ - 2s
15ms/step - loss: 3.6274e-05

Epoch 7/50

107/107 *================================+ -

- -05

2s

22ms/step loss: 3.6100e

Epoch 8/50

107/107 *================================+ -
2s 15ms/step - loss: 3.6038e-05

Epoch 9/50

107/107 *================================+ -
2s 21ms/step - loss: 3.4980e-05

Epoch 10/50

107/107 *================================+ -
2s 21ms/step - loss: 3.3884e-05

Epoch 11/50

107/107 *================================+ -
2s 21ms/step - loss: 3.1855e-05

Epoch 12/50

107/107 *================================+ -
2s 21ms/step - loss: 3.1442e-05

Epoch 13/50

107/107 *================================+ -
2s 16ms/step - loss: 3.2507e-05

Epoch 14/50

- 2s

```
107/107 *==================================+ -
2s
              -                   -05


107/107 *==================================+
18ms/step - loss: 3.1582e-05

Epoch 15/50

14ms/step loss: 2.8938e

Epoch 16/50

107/107 *==================================+ -
2s 23ms/step - loss: 2.6421e-05

Epoch 17/50

107/107 *==================================+ -
2s 22ms/step - loss: 2.6747e-05

Epoch 18/50

107/107 *==================================+ -
2s 19ms/step - loss: 2.4096e-05

Epoch 19/50

107/107 *==================================+ -
2s 18ms/step - loss: 2.5314e-05

Epoch 20/50

107/107 *==================================+ -
2s 20ms/step - loss: 2.5337e-05

Epoch 21/50



                                              - 2s
```

107/107 *==================================+ -

- -05

107/107 *==================================+ -
2s 23ms/step - loss: 2.2405e-05

Epoch 22/50

107/107 *==================================+
22ms/step - loss: 2.4915e-05

Epoch 23/50

1s

14ms/step loss: 2.1624e

Epoch 24/50

107/107 *==================================+ -
2s 19ms/step - loss: 2.1545e-05

Epoch 25/50

107/107 *==================================+ -
2s 22ms/step - loss: 2.2694e-05

Epoch 26/50

107/107 *==================================+ -
2s 22ms/step - loss: 2.0566e-05

Epoch 27/50

107/107 *==================================+ -
2s 19ms/step - loss: 2.2009e-05

Epoch 28/50

- 2s

107/107 *================================+ -
2s

- -05

107/107 *================================+ -
2s 18ms/step - loss: 2.2940e-05

Epoch 29/50

107/107 *================================+ -
2s 15ms/step - loss: 2.0115e-05

Epoch 30/50

107/107 *================================+
22ms/step - loss: 1.8910e-05

Epoch 31/50

23ms/step loss: 2.3294e

Epoch 32/50

107/107 *================================+ -
2s 19ms/step - loss: 1.8463e-05

Epoch 33/50

107/107 *================================+ -
2s 19ms/step - loss: 2.0214e-05

Epoch 34/50

107/107 *================================+ -
2s 22ms/step - loss: 1.8284e-05

Epoch 35/50

- 2s

107/107 *================================+ -

- -05

107/107 *================================+ -
2s 23ms/step - loss: 1.7490e-05

Epoch 36/50

107/107 *================================+ -
2s 21ms/step - loss: 1.8360e-05

Epoch 37/50

107/107 *================================+ -
2s 19ms/step - loss: 1.7240e-05

Epoch 38/50

107/107 *================================+
21ms/step - loss: 1.6853e-05

Epoch 39/50

3s

24ms/step loss: 1.5736e

Epoch 40/50

107/107 *================================+ -
2s 22ms/step - loss: 1.5684e-05

Epoch 41/50

107/107 *================================+ -
2s 15ms/step - loss: 1.7331e-05

Epoch 42/50

- 2s

107/107 *================================+ -
2s

- -05

107/107 *================================+ -
2s 23ms/step - loss: 1.6515e-05

Epoch 43/50

107/107 *================================+ -
2s 21ms/step - loss: 1.6822e-05

Epoch 44/50

107/107 *================================+ -
2s 16ms/step - loss: 1.4114e-05

Epoch 45/50

107/107 *================================+ -
2s 23ms/step - loss: 1.4346e-05

Epoch 46/50

107/107 *================================+
21ms/step - loss: 1.5537e-05

Epoch 47/50

- 2s

2s

- -05

19ms/step loss: 1.4485e

Epoch 48/50

107/107 *================================+ - 2s
23ms/step - loss: 1.4945e-05

Epoch 49/50

107/107 *================================+ - 2s
22ms/step - loss: 1.3325e-05

Epoch 50/50

107/107 *================================+ - 2s
19ms/step - loss: 1.2995e-05

54/54 *================================+ - 1s 3ms/step

Process finished with exit code 0

107/107 *==============================+ -

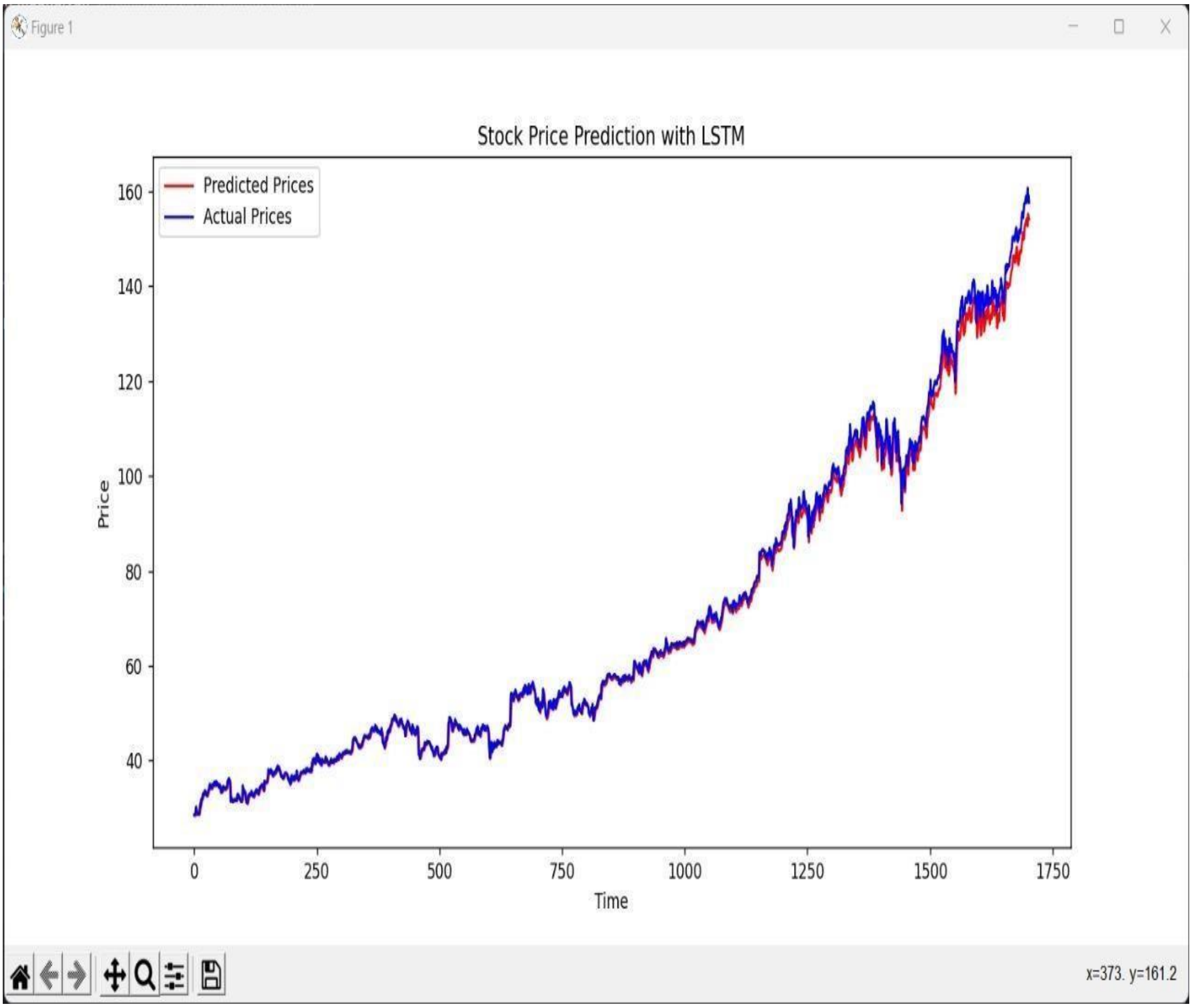


Figure 1
Stock Price Prediction with LSTM
Predicted Prices
Actual Prices
160
140
120
100
80
60
40
Price
0
250
500
750
1000
1250
1500
1750
Time
x=373. y=161.2