PROJECT SUMMARY REPORT
GIA NGUYEN NGUYEN – 104308290

COS 30018 – Intelligent Systems

Contents

Introduction

In This unit I'm work on Project B – Stock Price Predict System. In this project, using machine learning models to predict prices on a given stock over a given timeframe.

What' Included

- Project Report
- Task Report each week
- Final code
- Library install requirement.txt

Outline of Capabilities

I have implemented the following capabilities after completing the weekly tasks:

1. Loading different datasets with adjustable timeframes, with data saving
2. Displaying box plot charts and candlestick charts
3. LTSM Model hyper configuration adjustability with alternative layer types
4. Multivariate and multistep predictions
5. Ensemble predictions with ARIMA model

How to run

1. Install python newest
2. Open command terminal
3. Install virtual environment package (pip install virtualenv)
4. Create a python virtual environment (python -m venv [name virtual environment])
5. Activate the virtual environment (.\[venv name]\Scripts\actiivate)
6. Install library (pip install – r requirements.txt)
7. Run python program (python .\stock_prediction.py)

Overall System Architecture

The Python-based stock price prediction program leverages multiple libraries for data processing, visualization, and machine learning. Stock price data is retrieved from Yahoo Finance using the yfinance library. For data manipulation and mathematical operations, the program utilizes pandas and numpy. Visualization is facilitated by matplotlib and mplfinance.

In the machine learning component, the program employs the tensorflow library to design and train an LSTM (Long Short-Term Memory) model. Additionally, the sklearn library is used for data preprocessing and for splitting the dataset into training and testing subsets.

Implemented Data Processing Techniques

The program incorporates various data processing methods to prepare the data for machine learning:

- **Data Fetching:** Stock price data is retrieved using the yfinance library and stored as a CSV file for future reference.
- **Data Cleaning:** Missing values in the dataset are addressed through methods such as row deletion, forward fill, backward fill, or replacing them with the column mean.
- **Feature Scaling:** The MinMaxScaler from the sklearn library is used to normalize features within a specified range.
- **Data Splitting:** The dataset is divided into training and testing subsets, either randomly or based on a designated date.
- **Data Sequencing:** Sequences of a predefined length are created from the data to serve as input for the LSTM model.
- **Data Reshaping:** The data is reshaped to align with the input format required by the LSTM model.

Experimented Machine Learning Techniques

The program explores multiple machine learning techniques for stock price prediction:

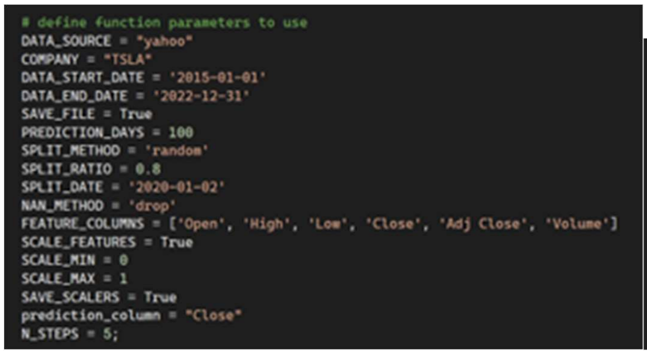
- **LSTM Model:** Leveraging a Long Short-Term Memory (LSTM) model, the program addresses long-term dependencies in time-series data. Built using the tensorflow library, the model architecture includes LSTM layers, dropout layers to mitigate overfitting, and a dense output layer.
- **ARIMA Model:** The program employs the AutoRegressive Integrated Moving Average (ARIMA) model for time-series forecasting. This model predicts future values by analyzing past data points and previous prediction errors.
- **Random Forest Model:** Experimentation extends to the Random Forest model, an ensemble learning approach. This model creates multiple decision trees and combines their predictions by averaging the outputs to enhance accuracy.
- **Ensemble Prediction:** The program generates ensemble predictions by combining the strengths of different models. It calculates average predictions by pairing the LSTM model with either the ARIMA model or the Random Forest model.

Scenarios / Examples

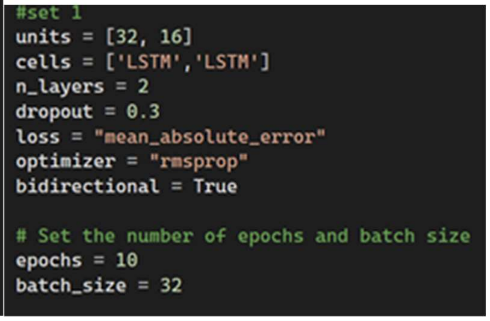
In this, I will detail some screenshots to help show basic usage and adjustment areas.

Parameter Adjustments

The main areas requiring adjustments are data processing and the hyperparameter configuration of the LSTM model.



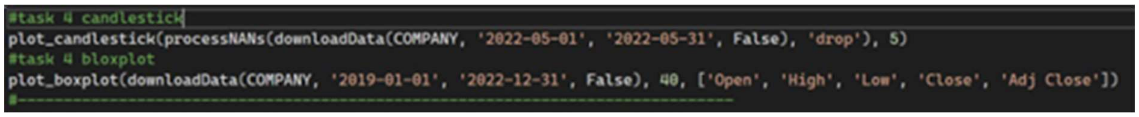
```
# define function parameters to use
DATA_SOURCE = "yahoo"
COMPANY = "TSLA"
DATA_START_DATE = '2015-01-01'
DATA_END_DATE = '2022-12-31'
SAVE_FILE = True
PREDICTION_DAYS = 100
SPLIT_METHOD = 'random'
SPLIT_RATIO = 0.8
SPLIT_DATE = '2020-01-02'
NAN_METHOD = 'drop'
FEATURE_COLUMNS = ['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume']
SCALE_FEATURES = True
SCALE_MIN = 0
SCALE_MAX = 1
SAVE_SCALERS = True
prediction_column = "Close"
N_STEPS = 5;
```



```
#set 1
units = [32, 16]
cells = ['LSTM','LSTM']
n_layers = 2
dropout = 0.3
loss = "mean_absolute_error"
optimizer = "rmsprop"
bidirectional = True

# Set the number of epochs and batch size
epochs = 10
batch_size = 32
```

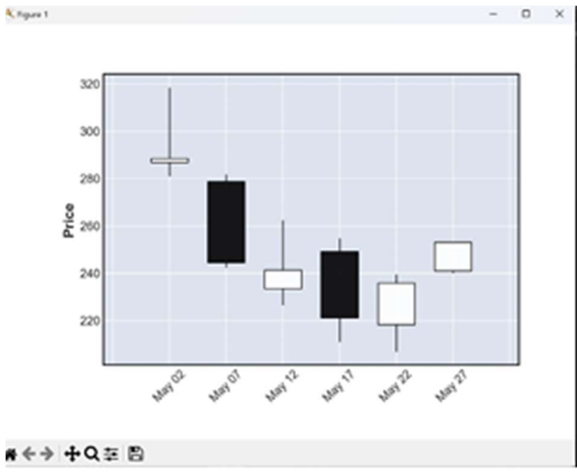
The settings for the candlestick and boxplot charts can also be adjusted on these lines:

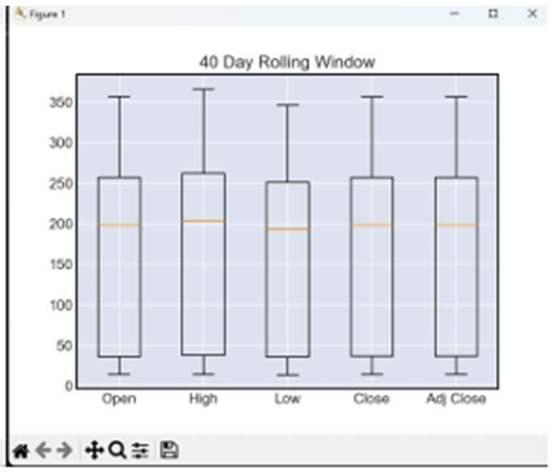


```
#task 4 candlestick
plot_candlestick(processNANs(downloadData(COMPANY, '2022-05-01', '2022-05-31', False), 'drop'), 5)
#task 4 bloxplot
plot_boxplot(downloadData(COMPANY, '2019-01-01', '2022-12-31', False), 40, ['Open', 'High', 'Low', 'Close', 'Adj Close'])
#------------------------------------------------------------------------
```

Charts

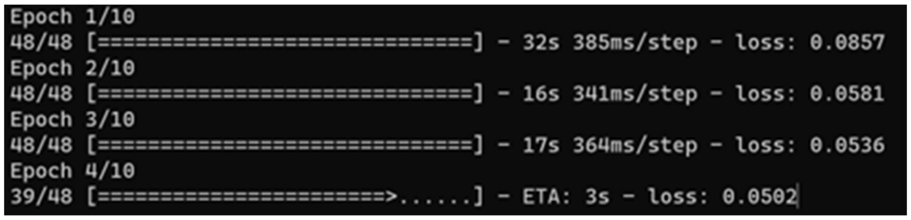
When the program runs, candlestick and boxplot charts for the data will appear first. These charts must be closed before moving on to the next stages of the prediction process.



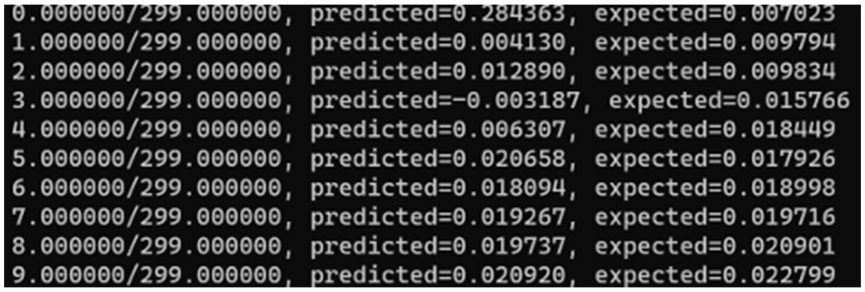


Predictions

Following the initial charts, the models will proceed through their training and prediction phases. This process may take some time, depending on the dataset and configuration settings.

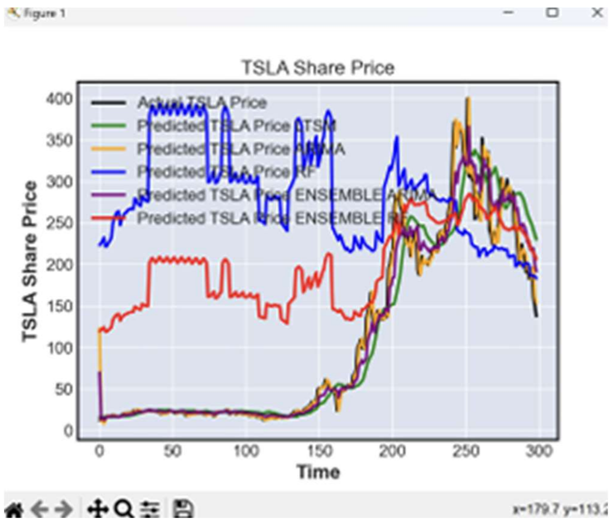


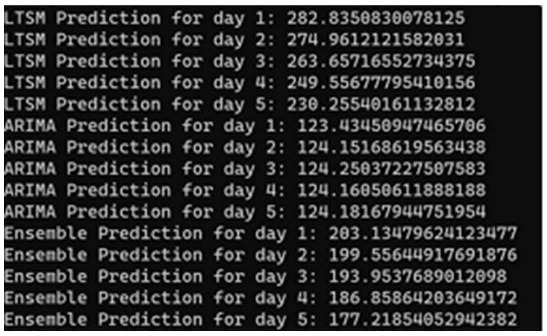
```
Epoch 1/10
48/48 [==============================] - 32s 385ms/step - loss: 0.0857
Epoch 2/10
48/48 [==============================] - 16s 341ms/step - loss: 0.0581
Epoch 3/10
48/48 [==============================] - 17s 364ms/step - loss: 0.0536
Epoch 4/10
39/48 [=======================>......] - ETA: 3s - loss: 0.0502
```



```
0.000000/299.000000, predicted=0.284363, expected=0.007023
1.000000/299.000000, predicted=0.004130, expected=0.009794
2.000000/299.000000, predicted=0.012890, expected=0.009834
3.000000/299.000000, predicted=-0.003187, expected=0.015766
4.000000/299.000000, predicted=0.006307, expected=0.018449
5.000000/299.000000, predicted=0.020658, expected=0.017926
6.000000/299.000000, predicted=0.018094, expected=0.018998
7.000000/299.000000, predicted=0.019267, expected=0.019716
8.000000/299.000000, predicted=0.019737, expected=0.020901
9.000000/299.000000, predicted=0.020920, expected=0.022799
```

A line chart displaying predicted prices alongside actual prices will then appear. Once this chart is closed, the next n-day predictions for the models will be output to the console.





```
LTSM Prediction for day 1: 282.8350830078125
LTSM Prediction for day 2: 274.9612121582031
LTSM Prediction for day 3: 263.65716552734375
LTSM Prediction for day 4: 249.55677795410156
LTSM Prediction for day 5: 230.25540161132812
ARIMA Prediction for day 1: 123.43450947465706
ARIMA Prediction for day 2: 124.15168619563438
ARIMA Prediction for day 3: 124.25037227507583
ARIMA Prediction for day 4: 124.16050611888188
ARIMA Prediction for day 5: 124.18167944751954
Ensemble Prediction for day 1: 203.13479624123477
Ensemble Prediction for day 2: 199.55644917691876
Ensemble Prediction for day 3: 193.9537689012098
Ensemble Prediction for day 4: 186.85864203649172
Ensemble Prediction for day 5: 177.21854052942382
```

Critical Analysis

1. As I progress in implementing the system, I have identified several strengths and areas for improvement based on a detailed analysis of my work:
2. **Strengths:**
3. **Model Versatility:** The system utilizes a diverse range of predictive models, including LSTM, ARIMA, and Random Forest. This multi-model approach enables the system to capture various patterns and dependencies in the data.
4. **Robust Data Preprocessing:** Comprehensive data preprocessing steps, such as handling missing values and scaling features, are implemented. These processes enhance the performance of the machine learning models.
5. **Ensemble Predictions:** The inclusion of ensemble methods that average predictions from multiple models adds robustness to the system. While ensemble methods typically improve results, this hasn't been the case here, presenting an opportunity for further refinement.
6. **Areas for Improvement:**
7. **Random Forest Optimization:** The Random Forest model, while included, is highly inaccurate. It requires adjustments and tuning to align its predictions more closely with those of the LSTM and ARIMA models.
8. **Enhanced Model Evaluation:** Although the system provides visual comparisons between predicted and actual prices, the ARIMA and Random Forest models lack quantitative performance metrics. Metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R-squared should be incorporated to objectively evaluate model performance.
9. **Hyperparameter Tuning:** The system does not currently include a mechanism for hyperparameter optimization. Using techniques like grid search or random search could help identify optimal hyperparameters, potentially improving model accuracy.
10. **Outlier Management:** Outliers in the dataset are not addressed, which can significantly impact model performance. Methods like the Z-score or Interquartile Range (IQR) could be implemented to detect and manage outliers effectively.

Conclusion

In summary, this unit proved to be quite challenging, requiring me to learn numerous new concepts and techniques to successfully complete the project. It provided me with a solid introduction to creating neural network models in a relatively straightforward context of stock price prediction. While there remain several areas for improvement, I feel I have achieved significant success in this unit, both in terms of understanding the material and developing a functional system that meets the given specifications.