ANSHUL VATS (104491909)

COS30018 – Intelligent Systems

SUMMARY REPORT – STOCK PREDICTION

Task B.1: Setup Project

Objective: The objective of Task B.1 was to establish an initial project environment and familiarize myself with the base code provided to lay the ground for subsequent steps of development.

Methods:

Environment Setup: Was done on a virtual environment to isolate the project's dependencies. Base code was downloaded from Canvas with one extra reference project - P1 - from GitHub to review other solutions and structures.
Installation of Libraries: The libraries were installed by creating a requirements.txt file for easy setup on other systems. It contained essential Python packages like machine learning, handling data, and extracting finance data, such as yahoo_fin and tensorflow. Testing: Tests were run and performed to ensure proper functionality in both versions, v0.1 and P1, with some sample stock data; the training can be excellently run without any errors.
Challenges and Solutions: Early testing showed that a few of the library dependencies were outdated. I researched compatible versions and bumped some of the libraries in order to maintain functionality. Some comments in the code also presented further modification and improvements that could be made.

Results: I successfully tested setups for both v0.1 and P1. Screenshots and documentation of this showed that the project had been initialized correctly and was ready for the rest of the development that would occur in later tasks.

Task B.2: Data Processing 1

Objective: The objective of this work is to enhance data manipulation to be flexible regarding different time ranges and to include several stock features such as Open, High, Low, and Volume to make the data more robust for the model.

Approach:

Data Loading: A load_data function was implemented that enables users to define both start and end dates for loading the data. This enables the project to manipulate historical data for particular

periods.
Data Cleaning and Feature Scaling: MinMaxScaler was used for feature normalization, hence optimizing the features for model ingestion. A copy of the original dataset was saved so that the unscaled values can be kept for evaluation.
Data Splitting: Added options for date or random splitting in order to allow flexibility depending on what type of analysis is in focus.
Handling Missing Data: Created options in order to efficiently cope with NaN values without distorting the dataset using various interpolation methods and threshold-based row exclusion.
Challenges and Solutions: It was a challenge to ensure all the required columns were present in the dataset and scaled properly. Testing with different stocks also helped to test the robustness of the function against different data format types.

Results: With a better data loading function, better processing of data was carried out, and better selection of features was enabled, which prepared the project for the smooth progress of the task into machine learning. Consistent formatting of data maintains a high model training accuracy.

Task B.3: Data Processing 2

Objective: The activity originally contributed the data visualization techniques in this activity to candlestick and boxplot charts for further analysis of stock price trends.

Procedure:

Candlestick Chart: The plot_candlestick_chart function resamples the stock data to aggregate it over a specified number of trading days; each candlestick visualizes the Open, High, Low, and Close prices for a customizable period using the mplfinance library.
Boxplot Chart: The plot_boxplot_chart function shows the distribution of the data across Open, High, Low, and Close stock features. This chart gave insight into the dispersion and anomalous nature of the data during specific trading periods.
Challenges and Solutions: It was difficult to aggregate data for the candlestick chart properly because slight changes would result in incorrect period groupings. However, it was checked against various stocks, confirming that visualizations would work.

Outcomes: These visualizations gave enlightening insights into the stock trends and allowed the identification of patterns in prices before training the models. Example charts to illustrate the task were included in the report.

Task B.4: Machine Learning 1

Objective: Introduce automated model construction to make experimentation with deep learning architectures and configurations for optimal predictive performance straightforward.

Methods:

Model Creation: In that regard, a function, create_model, was created that builds a sequential model; its parameters come from the user. These are the kind of layer to use, either LSTM, RNN, or GRU; how many layers could be used; whether the layer is bidirectional; and the dropout rate. Several combinations have been tried in finding which ones returned the highest accuracy.
Hyperparameter Tuning: It also allows for variable hyperparameters such as units per layer, dropout, optimizer, and loss function in said function for quick hyperparameter testing.
Challenges and Solutions: The challenge was to maintain a balance of model complexity concerning training time. EarlyStopping helped to avoid overfitting, and it also provided a real-time training loss with a custom callback function for visualizing learning in progress.

Results: Tests run with multiple configurations made it easier to understand how model architecture influences accuracy. Documented the best configuration selected for subsequent tasks, thus setting a good foundation for further deep learning tasks.

Task B.5: Machine Learning 2

Objective: This exercise made the model predict on both multivariate and multistep data, hence more robust for real-world usage.

Methods:

Multistep Prediction: This gives a function get_multistep_predictions, which uses recent stock data in forecasting upto many days into the future. This is useful in projecting price trends over a period rather than just a single day.
Multivariate Prediction: Increased the input features from just closing prices to include Open, High, Low, and Volume. With these features fed into the model, the model would have more context upon which it could draw and hopefully make better predictions.
Combined Multivariate and Multistep Prediction: The functionalities were put together in order to solve complex prediction scenarios. It simply enables the practitioner to make multistep predictions using many feature inputs.
Solution and Challenges: Due to the nature of input sequences, construction of various features for multistep forecasting needed to be handled very precisely as the sequences could neither overlap nor gap out completely. Different datasets tested ensured robustness in every respect.

Results: Multistep and multivariate predictions showed a great gain in predictions by the model. This extended functionality of the model prepared it for advanced use cases that give a much better forecast of the future price sequence.

B.6 Task: Machine Learning 3 (Ensemble Modeling)

Objective: The project had a specific task included to introduce one of the approaches for ensemble modeling, where the predictions through ARIMA were combined with LSTM to develop a more resilient prediction model.

Approach:

ARIMA Model Development: The integrated ARIMA was used to predict stock prices based on historical time series patterns. Several different configurations of (P, D, Q) were tested for optimal tuning in the stock price data.
Ensemble Methodology: The final prediction was obtained by combining the predictions of ARIMA and LSTM models through the weighted average function. As there was a need to strike an optimal balance, weights for the ensembling were chosen as 0.3 for ARIMA and 0.7 for LSTM because this set of values gave the best results.
Metrics Calculation: Calculated various metrics like MAE and RMSE for both individual models and their combination to validate if the prediction accuracy has improved.
Challenges and Solutions: The parameters of ARIMA and the weights in the ensemble method needed several tries to achieve the best performance. Comments and references were added inside the code to make it easier for readers to understand and reproduce the code.

Results: The model developed through the ensemble performed better when compared to each individual model, ARIMA, and LSTM, as illustrated by the measure of accuracy. The visualization in predictions showed that the improvement achieved great consistency in forecasting.

Task B.7: Project Extensions

Objective: This assignment, B.7, was to enhance the project using one additional advanced technique that needed to be of an independent choice.

Approach:

Research and Idea Selection: Made a study on some research related to sentiment analysis in stock price prediction. An approach has been selected which merges sentiment data such as Twitter mentions with historical price data, at least attempting to determine how market sentiment informs short-term pricing.
Integration and Implementation: Extended the model pipeline to include sentiment scores as input features. These were then combined with historical prices to form a multivariate input, preprocessed similar to previous feature configurations.
Performance Evaluation: The model was assessed in terms of predictions integrated with sentiment through MAE and RMSE, among other metrics. Such results were put up against the earlier model so that the improvement level would be visible.
Challenges and Solutions: Integrating real-time data sources, such as Twitter, proved challenging, and for this reason, static sentiment scores derived from previous research were utilized. This had the added benefit of testing without the potential compromise of data integrity.

Results: The extended model performed better in volatile markets, therefore confirming that sentiment data contributes to better short-term predictions. That result showed the potential of this project for further development with external data sources integration.

Code Overview: stock_prediction.py

The above functionalities are integrated into the stock_prediction.py script, and form the core for all the tasks listed below:

Data Loading and Preprocessing: The function `load_data` fetches data, scales, and prepares sequences. It is highly flexible and enables one to pass custom date inputs, feature scaling, and a variety of options regarding how to split the data.

Model Construction: The `create_model` function constructs a tunable deep learning model, LSTM/RNN/GRU, which can easily experiment with architecture.
Visualization: Functions such as plot_candlestick_chart and plot_boxplot_chart enable detailed analysis of the stock data in order to validate the model.
ARIMA and Ensemble Models: The integration of the ARIMA model with ensemble functionality will ensure robust forecasting, while weights will make sure there is a balanced prediction based on the strength of each individual model.
Performance Evaluation: The error metrics will be computed for various models, and plots of predictions were provided for easy comparison.
Conclusion
This report summarizes key objectives, approaches, and major results of all the tasks involved in the COS30018 stock prediction project. The current project has evolved step by step from basic data processing to advanced ensemble and extension techniques. This thus eventually finalizes a flexible and insightful stock prediction model. The incorporation of external sentiment data sets into the project can show its real-world application and easily adapt to future enhancements