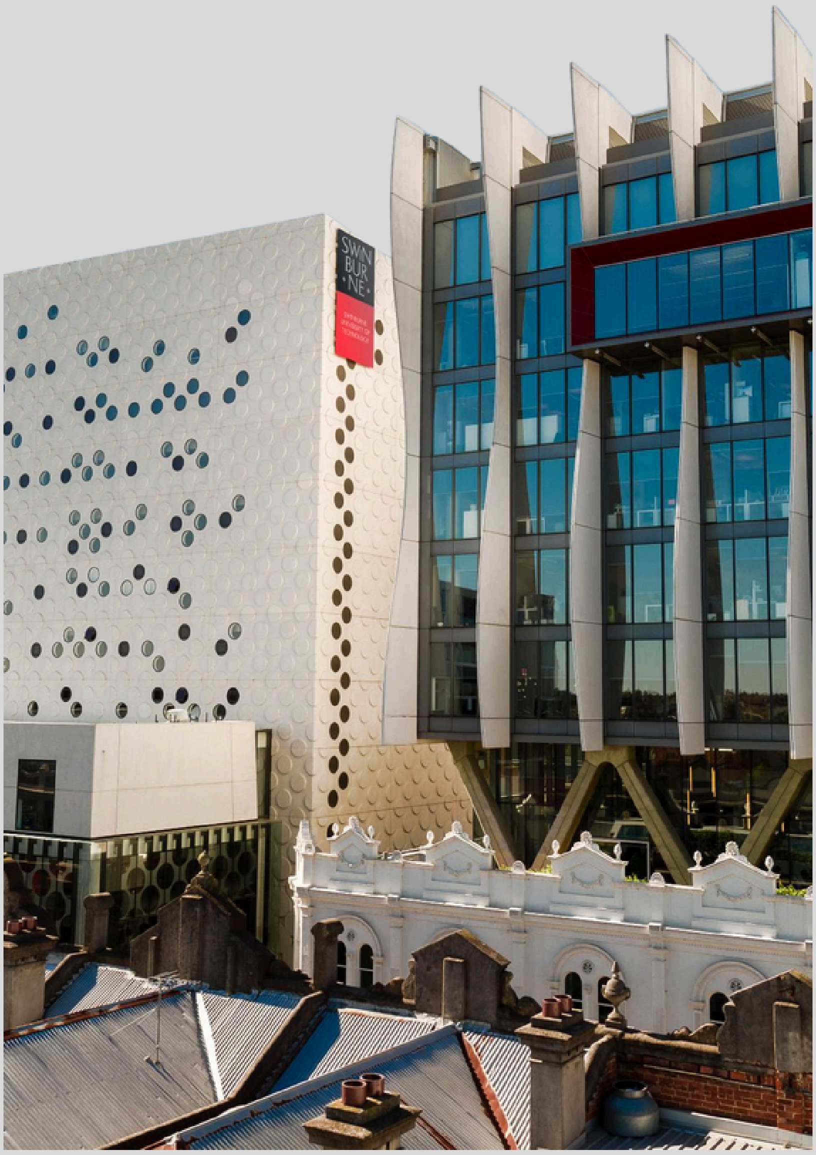
Swinburne University of Technology

TASK B.4 REPORT

COS30018

Intelligent Systems



Instructor:

Mr Aiden Nguyen

Group 4:

Le Hoang Triet Thong

104171146

Table of Contents

Task B4

This report analyzes the v0.4_codebase_stock_prediction.ipynb notebook, confirming its successful implementation of the dynamic model creation and experimentation framework required by "Task B.4 - Machine Learning 1". The project has evolved from using a hard-coded model to a versatile system capable of building, training, and evaluating various Deep Learning architectures programmatically.

1. Dynamic Model Creation Function

The central requirement was to create a flexible function that could build different types of neural networks based on specified parameters.

Requirement Fulfillment: This is masterfully achieved in **Cell 9** with the build_model function.

a. Flexible Architecture:

Instead of taking simple numbers for layers and sizes, the function accepts a layers_config parameter. This is a list of dictionaries, where each dictionary defines a layer's type (e.g., 'lstm', 'gru', 'rnn', 'dense'), its units, and other relevant parameters like return_sequences. This is a robust and extensible method that directly fulfills the requirement to specify layer names, sizes, and counts.

b. Support for Multiple Layer Types:

The function's logic uses if/elif statements to correctly add LSTM, GRU, SimpleRNN, Dense, and Dropout layers from TensorFlow/Keras based on the type specified in the configuration. This directly addresses the need to create different DL networks.

c. Detailed Code Explanation:

The function is well-documented with a clear docstring explaining its purpose, arguments, and return value. Inline comments and print statements are used effectively to show the model-building process step-by-step, satisfying the need for detailed explanation.

2. Experimentation with Models and Hyperparameters

The assignment required using the model-building function to experiment with different network types and hyperparameter configurations.

Requirement Fulfillment: This is demonstrated methodically in **Cells 10, 11, and 12**.

a. Experiment 1: LSTM Model (Cell 10):

A specific lstm_layers configuration is defined and passed to the build_model function. The model is then trained with a defined number of epochs (50) and a batch_size (32).

b. Experiment 2: GRU Model (Cell 11):

A new gru_layers configuration is created, swapping 'lstm' layers for 'gru' layers. This model is also trained for 50 epochs, allowing for a direct comparison with the LSTM.

c. Experiment 3: Simple RNN Model (Cell 12):

A third rnn_layers configuration is defined. The number of epochs is increased to 75, demonstrating experimentation with training parameters as required.

d. Evaluation:

After each experiment, the model's performance on the test set is evaluated, and the final Test Loss (MSE) and Mean Absolute Error (MAE) are printed, providing clear results for each configuration.

3. Summary of Experimental Results

The final part of the task was to summarize the results of the experiments.

Requirement Fulfillment: This is comprehensively covered in **Cell 15**.

- All three trained models (LSTM, GRU, RNN) are loaded from their saved files.
- Predictions are made on the test data for each model.
- The predictions are correctly de-scaled using the saved close_scaler to convert them back to dollar values.
- A pandas DataFrame (metrics_df) is created to present a clear, side-by-side comparison of the **MAE (**),MSE,andRMSE(**)** for each model. This table serves as a perfect summary of the experimental results.
- The notebook goes further by providing detailed visualizations and statistical summaries of the prediction errors for each model, offering deep insight into their respective performance.

Conclusion

The code in v0.4_codebase_stock_prediction.ipynb fully satisfies all requirements of Task B.4. It successfully implements a dynamic model-building function, uses it to conduct and evaluate a series of well-structured experiments with different architectures and hyperparameters, and presents the findings in a clear, comparative format.