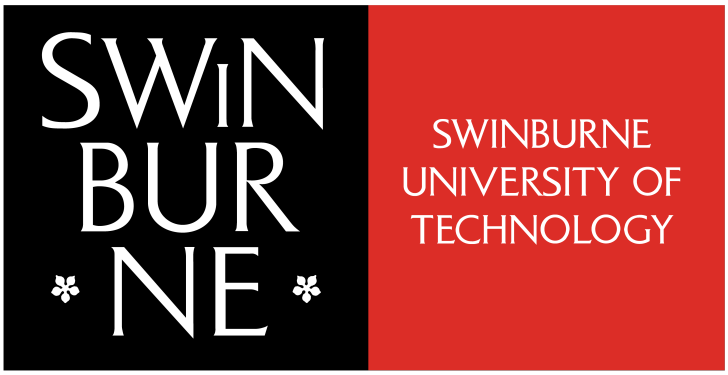

Assignment 2B

Machine Learning and Software Integration

COS30019 - Introduction to Artificial Intelligence

Student:	***ID:***
Hoang Trang Anh Pham	105006870
Syed Haider Bukhari	104083199
Farzana Moietry	103804690
Parav Sharma	105321434

23/05/2025

Contents

1 Introduction

This project is focused on the development of a Traffic-Based Route Guidance System (TBRGS), an intelligent transport tool that integrates machine learning and optimisation techniques to enhance urban mobility. The goal is to support commuters in identifying the most time-efficient travel paths by leveraging both real-time and forecasted traffic conditions within Melbourne's Boroondara area.

To achieve this, we utilise historical SCATS (Sydney Coordinated Adaptive Traffic System) traffic flow data as the foundation for training our machine learning models. These models include Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Bidirectional LSTM (BiLSTM), each selected for their strengths in predicting time series. By learning patterns in traffic volume, the models provide accurate forecasts of traffic flow at specific intersections over time.

The predicted flow data is then used to estimate travel time between SCATS sites, which serves as input for the routing component of the system. A graph is constructed, where intersections are represented as nodes and predicted travel times determine the weight of the edges. Search algorithms such as A* and Greedy Best First Search are applied to find the optimal routes, prioritising speed and traffic avoidance.

Ultimately, this project bridges artificial intelligence and transportation planning, offering a data-driven decision-making tool for efficient urban navigation. It lays the groundwork for future enhancements such as real-time updates, broader geographic coverage, and multimodal transport integration.

2 Problem Statement

Given the SCATS site data from VicRoads, the challenge is to:

- Process and structure the traffic volume data.
- Train and evaluate multiple ML models (LSTM, GRU, BiLSTM).
- Estimate travel time between intersections.
- Integrate prediction results into a search-based routing algorithm from Part A.
- Display results visually in a user-friendly interface.

3 Summary of Implemented Algorithms

The styling for the website ensured design consistency across all pages because management was through a single external CSS file. The following principles were a focus within the design strategy:

- **LSTM (Long Short-Term Memory):** a type of recurrent neural network (RNN) designed to address the vanishing gradient problem in the traditional RNN. LSTMs excel at learning and retaining information over long sequences of data, making them suitable for various applications like natural language processing, time series forecasting, etc.
- **GRU (Gated Recurrent Unit):** also a type of RNN. It is similar to an LSTM model, but with a simpler architecture, using a gating mechanism to control the flow of information, selectively remembering and forgetting information using update and reset gates.
- **BiLSTM (Bidirectional LSTM):** another type of RNN. However, this model processes data in both forward and backward directions, allowing the model to consider the context of a sequence from both the past and the future. It consists of two standard LSTMs working in parallel.

4 Implemented Features

- Data Processing (data_processing.py): During our analysis of the Scats_Data_October_2006.xlsx file, specifically the Data Summary sheet, we observed that each SCATS Site corresponds to multiple Locations. To simplify the scope, we performed data transformation on the original dataset. This process began with loading Scats_Data_October_2006.xlsx and grouping the data by SCATS Number and Date, calculating mean traffic volume while retaining first occurrences for other columns, resulting in grouped_scats_data.csv. This transformed dataset then underwent comprehensive processing and feature engineering. The processing phase included providing an untouched data overview, converting the data structure from wide to long format for improved accessibility, eliminating duplicate entries, removing SCATS sites with insufficient date records, and identifying outliers. The feature engineering stage involved extracting relevant features from the dataset, creating sequences, implementing an 80-20 train-test split, normalizing the data, preparing inputs, and finally storing all processed data in a designated 'processed_data' folder.
- Model Training (model_training.py): In our implementation, we utilize Keras for time-series prediction tasks, implementing three distinct neural network architectures. The first architecture, create_lstm_model(), takes features and categorical SCATS input, consisting of an

Embedding layer followed by three LSTM layers with decreasing units, an optional Attention layer, and Dense layers, ultimately producing a single regression value output. The second architecture, create_gru_model(), follows a similar structure but replaces LSTM layers with GRU layers, while the third architecture, create_bidirectional_lstm_model(), employs bidirectional LSTM layers to capture better contextual information. All these functions return compiled Keras Models, and they are organized in a Model_List dictionary that maps model type strings ('lstm', 'gru', 'bilstm') to their corresponding creation functions. During the training process, each model is compiled using Mean Squared Error (MSE) loss and Mean Absolute Error (MAE) metric, trained with the provided inputs and callbacks, and saves three outputs: the final model (final_model.keras), the best-performing model (best.keras), and TensorBoard logs for monitoring and analysis.

- Predicting (prediction_making.py): This script serves two purposes. Firstly, it identifies and addresses missing rows in the October dataset, where each SCATS location should contain 2,976 rows (calculated from 96 intervals per day multiplied by 31 days). Using three different models, it predicts both these missing values and traffic volumes for November 1st and 2nd. These predictions are stored as both .csv and .pkl files in the 'predicted_csv' folder. The predicted data is then merged with the original dataset (cleaned_data.csv from the processed_data folder) to create a comprehensive traffic volume dataset spanning from October 1st to November 2nd, 2006, with the complete data stored in both .csv and .pkl formats within the 'complete_oct_nov_csv' folder. The prediction range was intentionally limited to maintain accuracy. Secondly, the script generates predictions for traffic volumes from October 1st to October 3rd, 2006 (using data the models were trained on), saving these predictions as .csv and .pkl files in the 'oct3_csv' folder. These predictions serve as the foundation for the subsequent model performance evaluation phase.

- Model evaluating (model_evaluating.py): This script performs a comprehensive evaluation of model predictions by first combining predicted and actual data for each model to enable direct comparison with ground truth values. The evaluation process calculates three key performance metrics: Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and the Coefficient of Determination (R^2) for each model. Following the calculations, the script generates a detailed comparison report and preserves the metrics for all models in both CSV and Markdown formats. To visualize the results, it creates two types of plots: a bar chart that compares metrics across different models, and scatter plots that illustrate the relationship between predicted and actual traffic volumes for each model. All evaluation outputs, including the metrics and visualizations, are stored in the evaluation_results directory for further

analysis and reference.

- GUI (main.py): This script implements functionality for calculating inter-SCATS travel time based on traffic flow data from the originating SCATS at specific timestamps. Given the superior performance demonstrated during evaluation, the system utilizes BiLSTM model predictions (stored in complete_oct_nov_csv_model_model_complete_data.csv) to calculate estimated speeds and travel times. The user interface allows input of Origin SCATS, Destination SCATS, Date, Start time, and preferred Algorithm selection. Upon processing these inputs, the system calculates the travel time and displays a detailed route breakdown in the window's right panel. Additionally, it generates a folium map visualization to display the calculated route. It's important to note that when using Depth Limited Search, the system implements a 10-node expansion limit; searches exceeding this threshold will result in a "No path found" message in the GUI window.

5 Known Bugs and Missing Features

- Use of external data: In the original data (Scats_Data_October_2006.xlsx), there are only LATITUDE and LONGITUDE that represent the SCATS site coordinates. We struggled with extracting the relationship between SCATS sites (i.e., which site is connected to which) from the original data. Through research, we found this dataset that already has neighbour nodes for each SCATS site ("Neighbours" column), so we decided to use it for our map.

6 Test Cases

Outcomes Achieved:

- Test case 1: Compare prediction between GRU, LSTM, BiLSTM
  - Input: The same input for all models
  - Expected Output: Different predictions with GRU are generally faster, and BiLSTM is more accurate

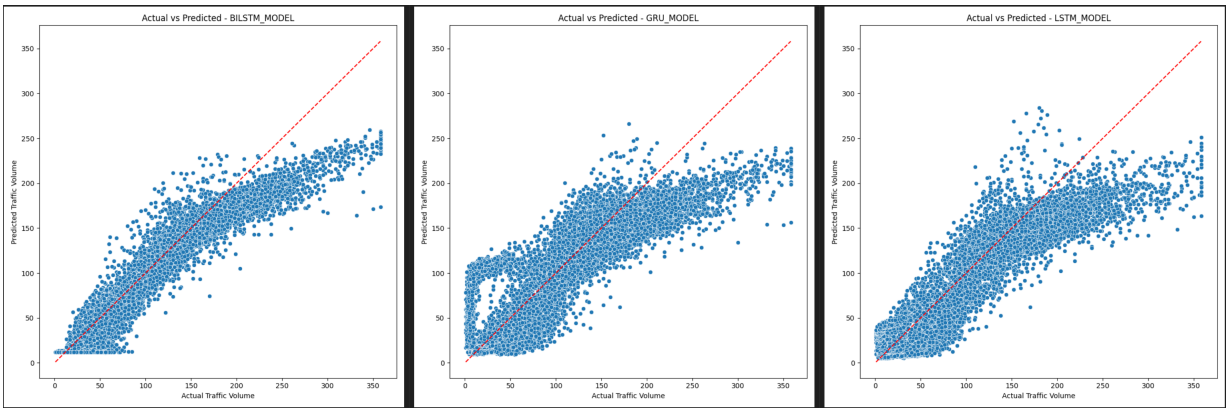

Figure 1: Predictions made by three models

- Test case 2: Handle missing values in the input
  - Input: October traffic data with some missing values
  - Expected Output: Predictor detects and fills in with predictions

main.py | traffic_predictions_oct_nov.csv | data_processing.py | requirements.txt

predicted_csv > bilstm_model > traffic_predictions_oct_nov.csv > data

```
SCATS Number,Date,interval_id,time_of_day,predicted_traffic
2000,2006-10-13,0,00:00,56.12852
3682,2006-10-13,0,00:00,21.095705
2000,2006-10-13,1,00:15,52.85022
3682,2006-10-13,1,00:15,13.583793
2000,2006-10-13,2,00:30,50.634476
3682,2006-10-13,2,00:30,11.879091
2000,2006-10-13,3,00:45,47.945343
3682,2006-10-13,3,00:45,11.8167715
2000,2006-10-13,4,01:00,44.457245
3682,2006-10-13,4,01:00,11.782212
2000,2006-10-13,5,01:15,39.928074
3682,2006-10-13,5,01:15,11.766288
2000,2006-10-13,6,01:30,35.372242
3682,2006-10-13,6,01:30,11.763126
2000,2006-10-13,7,01:45,30.367987
3682,2006-10-13,7,01:45,11.780416
2000,2006-10-13,8,02:00,23.489027
3682,2006-10-13,8,02:00,11.801507
2000,2006-10-13,9,02:15,14.619547
```

Figure 2: Prediction for missing values and November 1st and 2nd saved as a csv file

- Test case 3: Optimal path using A* Search
  - Input: Origin, destination, predicted traffic volumes
  - Expected Output: Fastest time route avoiding high-traffic edges

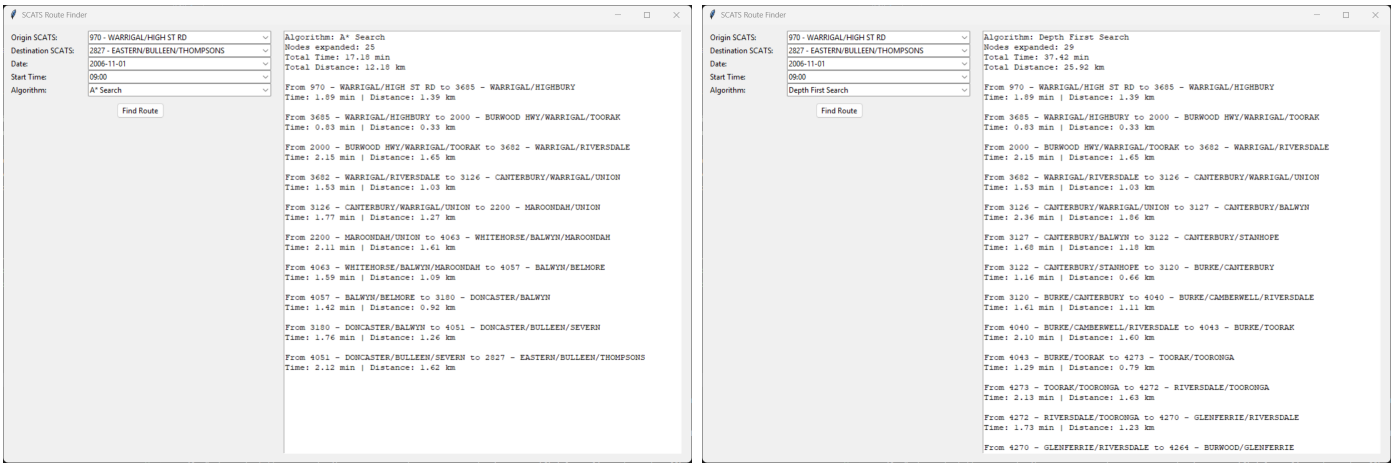

Figure 3: Pathfinding logic by A* versus Depth-First Search

- Test case 4: Depth Limited Search limit = 10
  - Input: Origin: SCATS 970, destination: SCATS 2827 (10+ nodes away)
  - Expected Output: No path found message

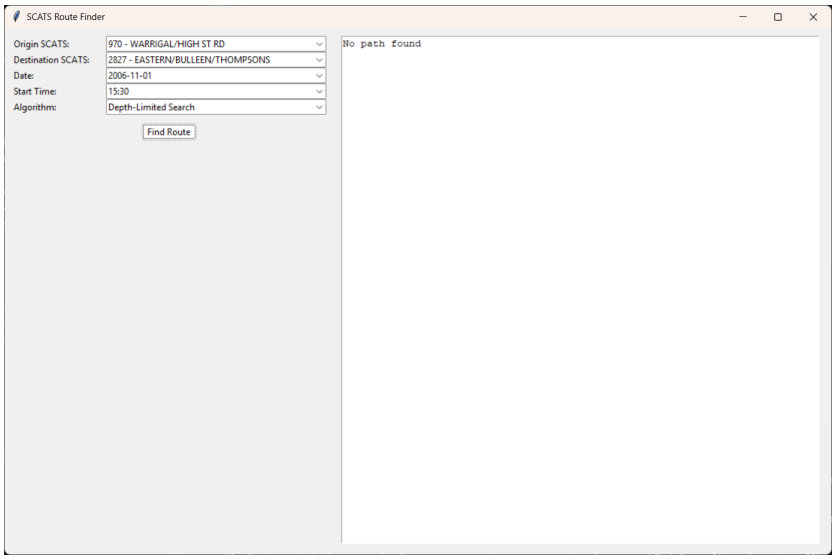

Figure 4: When nodes expanded excess 10, Depth Limited Search stops searching, resulting in a "No path found"

- Test case 5: Algorithm dropdown selection test
  - Input: Select Breadth-First Search
  - Expected Output: The correct algorithm used in pathfinding logic

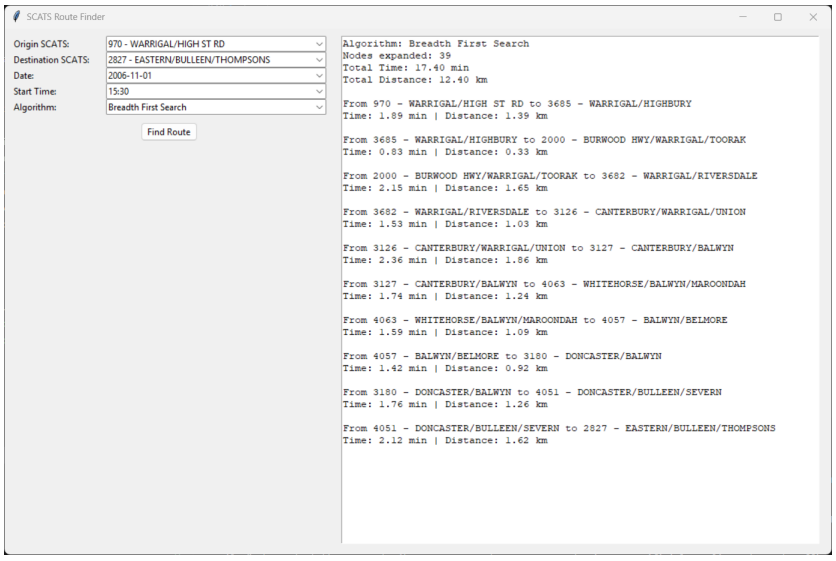

Figure 5: Pathfinding logic by Breadth-First Search

- Test case 6: Real-time display responsiveness
  - Input: Different inputs
  - Expected Output: Multiple map generations quickly
- Test case 7: Empty origin or destination
  - Input: Blank field
  - Expected Output: Error popup: "Please fill in all fields"

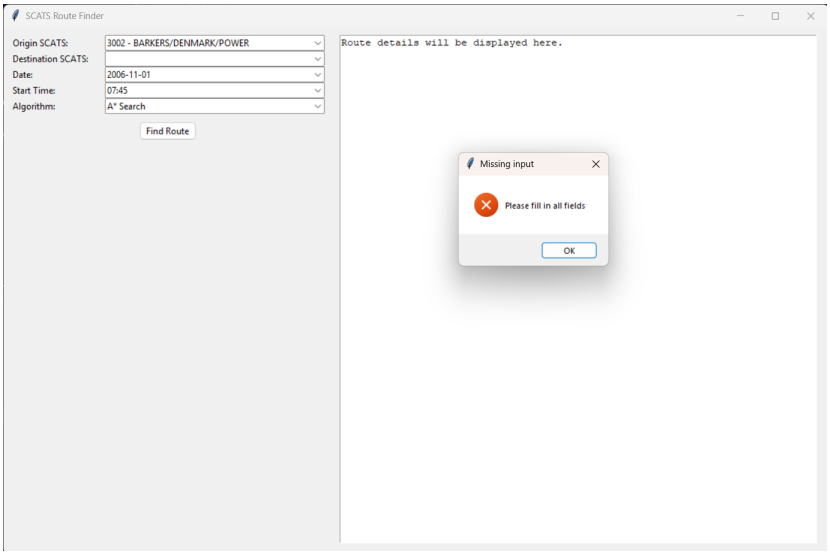

Figure 6: Error window pops up

- Test case 8: Show markers for the route found

  - Input: Any valid path
  - Expected Output: Folium map shows colored path with labels

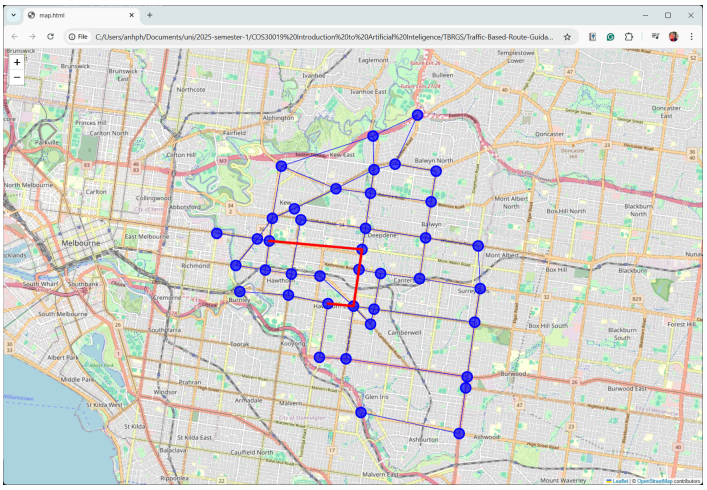

Figure 7: Route found by Greedy Best First Search

- Test case 9: Algorithm comparison (user switches options)
  - Input: A* Search, Greedy Best First Search
  - Expected Output: Different routes shown

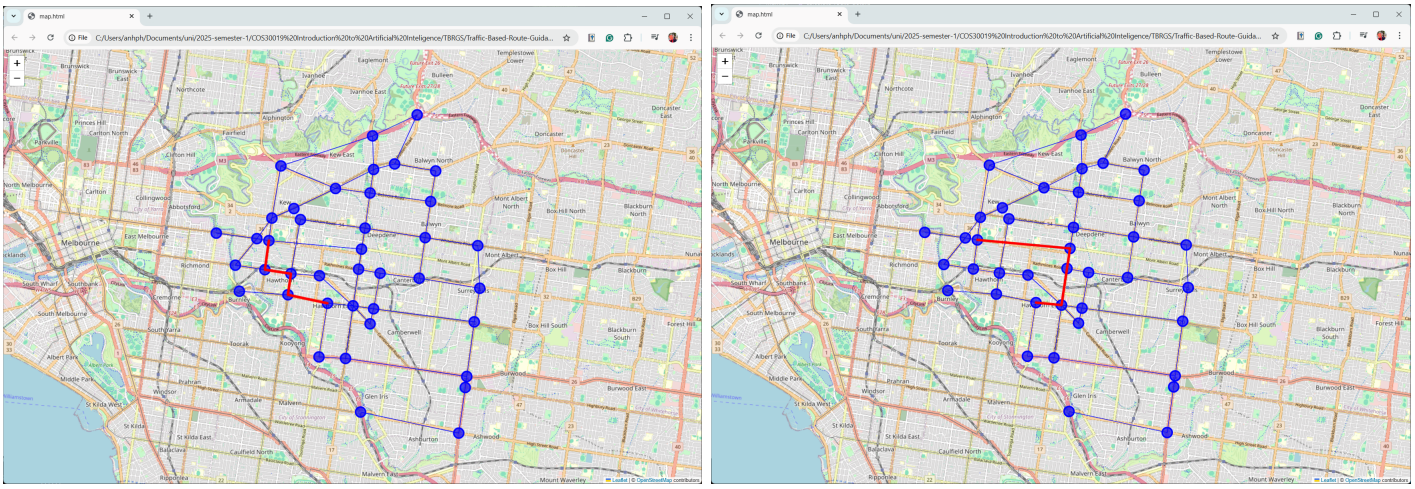

Figure 8: Route found by Greedy Best First Search versus A*

- Test case 10: Log all prediction outputs
  - Input: Any model run
  - Expected Output: Different predictions with GRU are generally faster, and BiLSTM is more accurate

Figure 9: Visualisations of BiLSTM training progress

7 Model Evaluation

7.1 ML Model Evaluation Metrics

- Mean Absolute Error (MAE): Shows average prediction error in the same units.
- Mean Absolute Percentage Error (MAPE): Indicates the average percentage difference between predicted values and actual observed values.
- Shows how much variance in the data is explained by the model. Useful for overall model quality.

7.2 Evaluation Result Summary

Model	MAE	MAPE	R2
LSTM	25.98	0.72	0.79
GRU	30.50	1.30	0.72
BiLSTM	17.20	0.34	0.89

7.3 Comparative Analysis and Insights

Among the three models implemented—BiLSTM, LSTM, and GRU—the BiLSTM (Bidirectional Long Short-Term Memory) model consistently outperforms the others across all key evaluation metrics: mean absolute error (MAE), mean absolute percentage error (MAPE), and R^2 score. This superior performance is demonstrated by BiLSTM achieving the lowest MAE of approximately 17, along with the most favourable MAPE and R^2 scores among all models.

The LSTM model ranks second in performance, with an MAE of around 26. While it maintains reliable performance in capturing long-term patterns, it shows notably higher error rates compared

to BiLSTM, though still performs better than the GRU model.
The GRU model, despite its reputation for faster convergence and lower computational cost, demonstrates the highest MAE at approximately 30. While it remains an option for environments with limited resources or where rapid prototyping is required, it shows the least favourable performance metrics among the three models in this specific implementation.
In summary, BiLSTM emerges as the most accurate and robust choice for traffic prediction, followed by LSTM, while GRU, despite its computational efficiency, shows the highest prediction error rates in this comparative analysis.

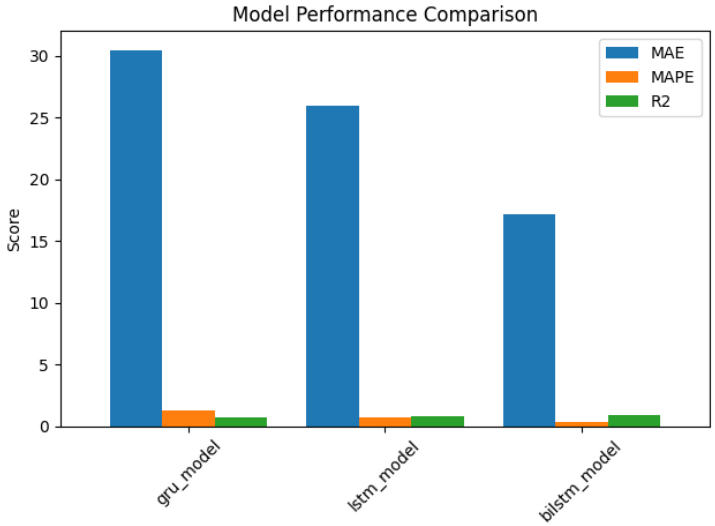

Figure 12: Bar chart visualising model performance comparison

8 Research Initiatives

This project's research initiative is the implementation of an interactive route visualisation system using Folium, an open-source Python library that leverages OpenStreetMap. The system addresses a critical challenge where the original SCATS coordinates did not align correctly with actual intersections on modern maps. Through empirical testing, a calibration offset of +0.0012 was applied to both latitude and longitude coordinates:
The visualisation system features:

- Dynamic base map generation centered on the SCATS network
- Color-coded representation of traffic sites (blue markers)
- Network connectivity visualization (blue connecting lines)

- Route highlighting (red polylines for selected paths)
- Interactive popups showing SCATS information
- Real-time route updates based on algorithm selection

This approach not only solves the coordinate mapping issue but also provides an intuitive visualization platform similar to Google Maps Traffic, allowing users to visually verify route recommendations, understand network connectivity, and interact with SCATS locations.

9 Acknowledgements and Resources

Tools and Libraries

- TensorFlow, Keras, Scikit-learn: For ML model development
- Pandas, NumPy: Data preprocessing
- Folium, Tkinter: GUI, map visualisation

10 References

Yu, B., Yin, H., Zhu, Z. (2018). "Spatio-temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting." IJCAI.

Vaswani, A., et al. (2017). "Attention Is All You Need." NeurIPS.

Christ, M., et al. (2018). "Time Series Feature Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package)." Neurocomputing.

Chollet, F. (2017). "Deep Learning with Python." Manning Publications.

11 Contribution

Student	*Contribution (out of 100%)*
Hoang Trang Anh Pham - 105006870	100%
Syed Haider Bukhari - 104083199	60%
Farzana Moietry - 103804690	60%
Parav Sharma - 105321434	60%