Ronan Neary - 993213

6219COMP – Implementation Summary

1. Explanation and Justification of Dataset and AI Technique Selection

1.1 Dataset Properties and Selection Justification

For this implementation, the selected dataset was the Yellow Taxi Trip Records - February 2024, provided by the New York City Taxi and Limousine Commission (TLC) and available publicly through the NYC Open Data portal [1]. The dataset contains detailed records of individual taxi trips including pickup and drop-off times, trip distance, passenger counts, fare amounts, and location information.

- **Total Records**: ~3 million entries
- **Number of Columns**: 19
- **Key Attributes:** pickup_datetime, dropoff_datetime, trip_distance, passanger_count, fare_amount, PULocationID, DOLocationID

This dataset was chosen because it was freely available and licensed for public academic use without restrictions [1]. It provides a rich and realistic transportation dataset that reflects the urban travel patterns in a large metropolitan area (New York), by offering high-quality geographic and trip related features that are suitable for applying AI-based predictive models.

It will be used to predict trip duration based on trip details; this problem has a direct impact on real-world benefits for route optimisation, passenger service improvements and estimating fares.

1.2 AI Technique Selection and Justification

The AI technique selected for this task was the random forest regressor, an ensemble-based machine learning model [2].

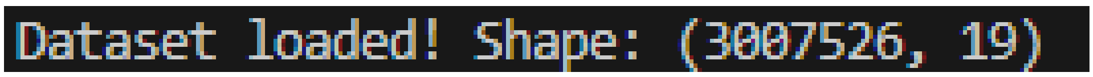
Random forest was selected because it handles non-linear relationships effectively, which is crucial since trip duration is influenced by several different factors (distance, time of day, etc) so the relationship will not be linear. This method is also good at handling noise/ outliers, which will be common in such a large dataset concerning urban transport. Also, it provides feature importance scores which allow the developer to interpret the model's performance.

Overall, given the size and complexity of the dataset, the random forest-based technique provides a strong balance between accuracy and speed.

2. Data Preparation

The following steps were conducted to prepare the data for the models training:

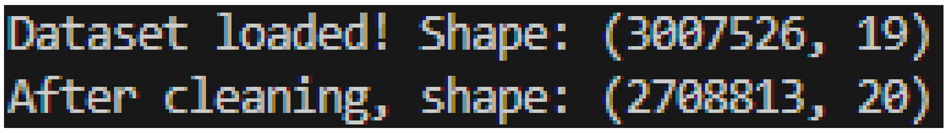
1. Loading the dataset:
   The Yellow Taxi February 2024 dataset was loaded form a '.paraquet' file using the pandas library



```
Dataset loaded! Shape: (3007526, 19)
```

Figure 1: Loading Dataset Terminal Output

   This above being the output into the terminal when the code was ran, displaying the dataset has been loaded and how many entries and columns it contains.
2. Initial Cleansing:
   - Rows were dropped with missing (null) values in critical columns (pickup_datetime, dropoff_datetime, trip_distance, passanger_count, fare_amount)
   - Calculated a new trip trip_duration column as the difference between dropoff and pickup times.
   - Removed trips where:
     - trip_distance <= 0 miles, or > 100 miles.
     - trip_duration < 60 seconds or > 7200 seconds (2 hours).
     - passenger_count <= 0
     - fare_amount <= 0

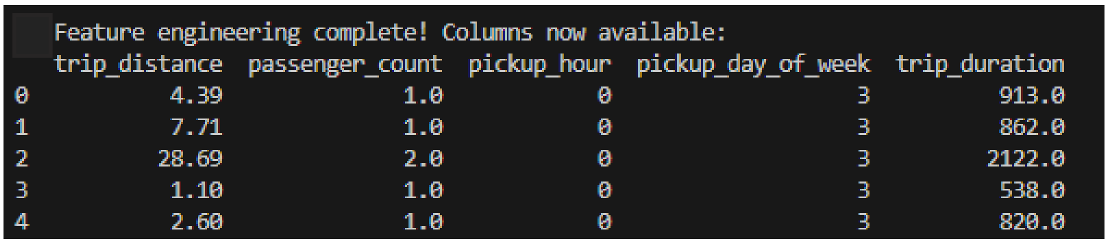


```
Dataset loaded! Shape: (3007526, 19)
After cleaning, shape: (2708813, 20)
```

Figure 2:Cleansing Data Terminal Output

Here in **Figure 2** we can see the effect of the cleansing techniques on the shape of the dataset. As mentioned, another column was added ('trip_duration') and the number of entries decreased by 9.93% to 2,708,813 from 3,007,526 due to cleansing methods previously stated.

3. Feature Engineering:
   'pickup_hour' was created to store the hour a passenger was picked up (0-23) from the pickup datetime column. Also, 'pickup_day_of_week' (0 = Monday to 6 = Sunday) from the pickup datetime column.

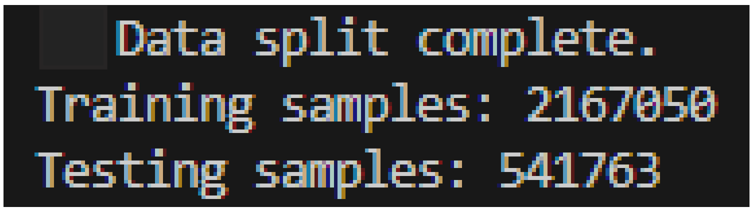


```
Feature engineering complete! Columns now available:
   trip_distance  passenger_count  pickup_hour  pickup_day_of_week  trip_duration
0           4.39              1.0            0                   3          913.0
1           7.71              1.0            0                   3          862.0
2          28.69              2.0            0                   3         2122.0
3           1.10              1.0            0                   3          538.0
4           2.60              1.0            0                   3          820.0
```

Figure 3: Feature Engineering Terminal Output

Figure 3 shows the output of the feature engineering step with the new columns made ready for the splitting.

4. Train-Test Split:
   80% of the data was used for training the last 20% was used for testing. The features trip_distance, passenger_count, pickup-hour and pickup_day_of_week are used in this process and the target variable is trip_duration.



```
Data split complete.
Training samples: 2167050
Testing samples: 541763
```

Figure 4: Data Split Terminal Output

Figure 4 is the terminal output after the data split step is completed, here we see the samples are divided with 80% going to training samples and 20% going to testing, as expected.

3. Implementation

3.1 Model Implementation and Parameter Setting

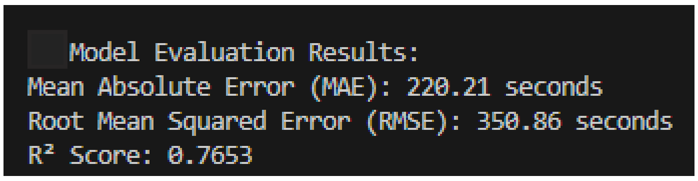
A Random Forest Regressor was implemented using the following settings:

- n_estimators = 100 – enough trees to maintain a reasonable training time and produce consistent results.
- max_depth = 20 – Limit the tree depths to avoid overfitting, making them too complex.
- random_state = 42 – allows the results to be reproduced
- n_jobs = -1 – Uses all the CPUs cores for faster training.

Model fitting was performed on the training set (x_train, y_train) which was followed by predictions on the test set (x_test).

3.2 Explanation and Interpretation of Evaluation Results

The model's performance on the test data was evaluated using three metrics as shown in **Figure 5:**



```
   Model Evaluation Results:
Mean Absolute Error (MAE): 220.21 seconds
Root Mean Squared Error (RMSE): 350.86 seconds
R² Score: 0.7653
```

Figure 5:Model Prediction Performance Results Terminal Output

The three metrics shown above can be interpreted in their own ways; the Mean Absolute Error (MAE) being 220.21 seconds means that on average predictions are about 220 seconds (3 minutes 40 seconds) off the true trip duration. Which in the context of urban taxi trips, where trips can last anywhere between 5 minutes and an hour, a 3–4-minute average error is reasonable. The Root Mean Squared Error (RMSE) is shown to be 350.86 seconds. The RMSE reflects larger errors more strongly as it squares the difference. So, the RMSE being around 350 seconds (5 minutes 50 seconds). In real world taxi data, some trips will naturally have bigger errors due to unexpected traffic jams or detours. So, the RMSE being a little higher than the MAE shows there are not massive, frequent outlier mistakes, only occasional large errors. Finally, the R^2 score is shown to be 0.7653 (76.5%), meaning the model explains approximately 76.5% of the variation in trip durations, a strong result for real-world traffic data.

Additionally, a scatter graph was plotted to help visualise model performance for assessment.

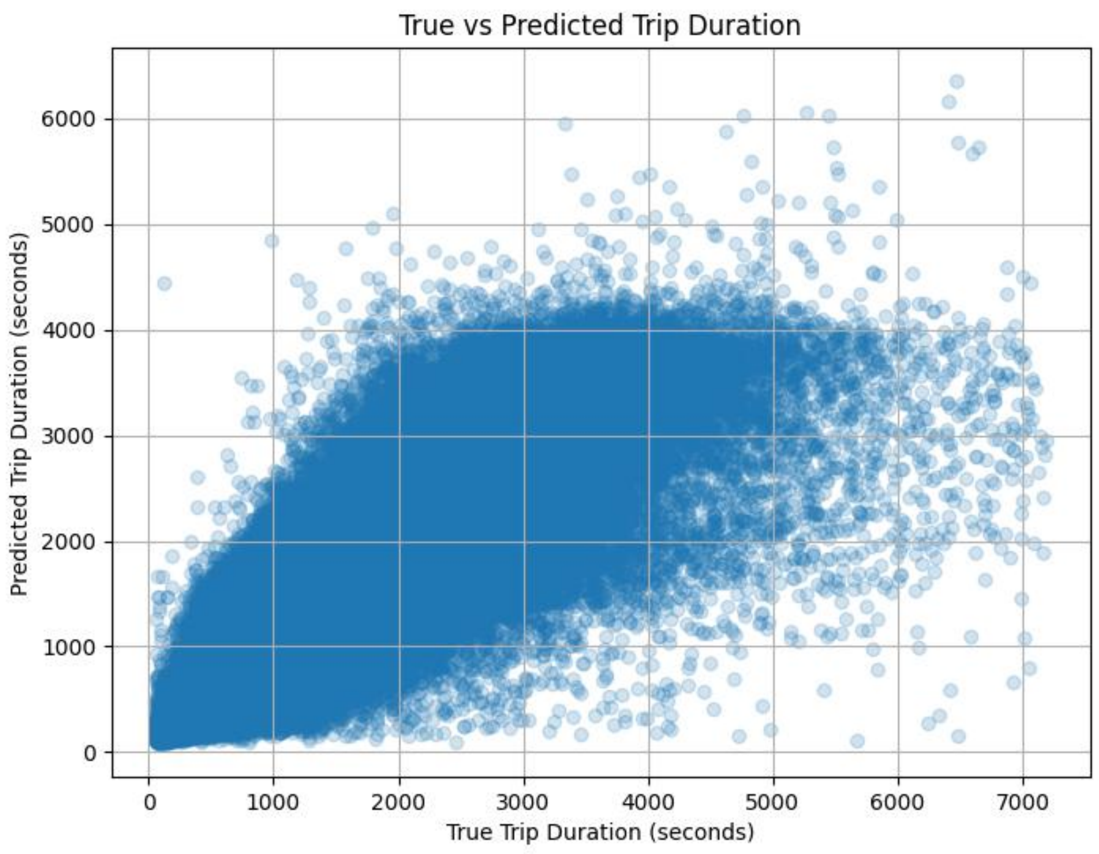


Figure 6: Scatter Graph Showing Relationship of True and Predicted Durations

The plot shown in **Figure 6** shows that while most predictions align closely with the true values, there is a greater spread for longer trip locations, consistent with the unpredictable nature of urban traffic in a city like New York.

4. Comparison with Other Possible Techniques

When solving the problem of predicting taxi trip durations, several alternative AI techniques could have been considered apart from random forest. Each technique offers its own advantages and challenges depending on the dataset's properties, computational resources and project goals.

4.1 Linear Regression

Linear Regression models the relationship between the dependent variable and one or more independent variables by fitting a linear equation to observed data [4]. Its potential pros are simplicity as it is easy to implement and interpret, speed as it is fast to train even on large datasets, and it provides a good baseline to simplify more complex models.

However, in this context some issues arise. This model assumes the data is linear, whereas the trip duration depends on distance, time, and other factors in a non-linear fashion. Also, Linear Regression models poorly handle outliers and can lead to the models being highly sensitive to extreme values which can occur in a real-world dataset [4]. Finally, this model will not be as accurate compared to the Random Forest in terms of capturing complex interaction in the data. Another reason to why It wasn't chosen for this solution.

Given that the relationships between trip features (distance, hour of day, day of the week) and trip duration are non-linear and multi-dimensional, Linear Regression would not be a suitable choice of model for this implementation.

4.2 Gradient Boosting Machines (GBMs)

Gradient Boosting [5] is an ensemble method where models are built sequentially, each new model correcting the errors of the previous ones.

It can lead to higher accuracy and could often outperform Random Forest in a real-world task like this. It is flexible as it can handle missing data and mixed data types natively and like random forest GBMs can also provide insights into which features matter the most.

But this comes at a high computational cost as training is significantly slower than Random Forest, especially without GPU acceleration. Also, with a higher complexity in terms of implementation due to the fine tuning required for this model.

Gradient Boosting Machines like XGBoost [6] could potentially achieve a better predictive performance than Random Forest. However, given the simplicity, interpretability and results, Random Forest was considered the more appropriate choice for a local implementation.

4.3 Neural Networks (Multi-Layer Perceptron)

Neural networks model complex non-linear relationships through layers of interconnected nodes (neurons) [7].

This method can be highly flexible as it can theoretically model any function making it effective for complex patterns, especially ones which involve high-level abstractions.

However, similar to GBMs it has a high computational cost to train and extensive tuning to produce sufficient results [8]. With small datasets there would be a risk of overfitting, and it can be a struggle to interpret the performance of neural networks as they don't inherently feature clear methods of evaluating model results/ performance. Making it more fitting to employ a method like Random Forest which provided a better balance between performance and interpretability.

Ultimately Random Forest was preferred for this implementation because it provides high predictive accuracy without heavy parameter tuning, required minimal data preprocessing and maintains good interpretability throughout this process. While on top of this being able to handle the non-linear nature of the New York taxi trip records. While

GBMs like XGBoost could offer slight improvements in prediction accuracy, the Random Forest model already achieved such a strong R^2 score of 0.7653, justifying its selection for the scope and goals of this problem.

References

[1] *TLC* (2024). *TLC Trip Record Data*

https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page.

[2] LJMU (2025), Session 06 SVM, Ensemble learning and Evaluation

[3] Breiman, L. (2001) 'Random Forests,' *Machine Learning*, 45(1), pp. 5–32. https://doi.org/10.1023/a:1010933404324.

[4] LJMU (2025), Session 05 Regression and Decision Tree

[5] Friedman, Jerome. (2000). Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics. https://www.researchgate.net/publication/2424824_Greedy_Function_Approximation_A_Gradient_Boosting_Machine

[6] Chen, T. and Guestrin, C. (2016) 'XGBoost,' *XGBoost: A Scalable Tree Boosting System*, pp. 785–794. https://doi.org/10.1145/2939672.2939785.

[7] LJMU (2025), session 09 Biologically inspired models of AI

[8] Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep learning*, *MIT Press eBooks*. https://dl.acm.org/citation.cfm?id=3086952.