CSE 151B Project Milestone Report

Kai Breese
kbreese@ucsd.edu
https://github.com/ninjakaib/taxiregression

1 Task Description and Background

1.1 Problem A

The objective of this research endeavor is to develop a robust predictive model capable of accurately estimating the duration of taxicab journeys in Porto, Portugal. This task is initiated with minimal information concerning the taxi's route and destination, requiring us to implement advanced predictive modeling techniques to infer ride durations.

The significance of this task extends well beyond academic interest. For major transportation organizations, such as Uber or other taxi service providers, the ability to forecast trip durations with precision can dramatically enhance operational efficiency. This could lead to substantial cost savings, optimally scheduling rides to reduce idle time and wasteful overlap. Moreover, it has the potential to alleviate road congestion, consequently diminishing the environmental footprint of urban transportation - an issue of pressing global concern.

From a user perspective, having accurate estimates of trip durations can vastly improve customer satisfaction. It facilitates better planning and reduces uncertainty, leading to an enhanced user experience. Furthermore, the applicability of deep learning for prediction and forecasting tasks is not confined to estimating taxi ride times. The methodologies and techniques employed in this project have wide-ranging implications. They could be utilized for sales and weather forecasting, time-series analysis in financial markets, and any predictive modeling involving data in tabular form.

In essence, this research has the potential to contribute significant advancements in the field of predictive modeling, with substantial real-world impact across various sectors.

1.2 Problem B

Let's denote the input features as $X \in \mathbb{R}^{n \times d}$, where n is the number of samples and d is the number of features. Each row in X represents a single trip with features including cyclical time, geolocation, and other categorical data, transformed into numerical form.

The corresponding output (trip duration) is denoted as $y \in \mathbb{R}^n$, where each element y_i represents the duration of the i-th trip.

The prediction task can be formulated as an optimization problem, where the goal is to find a function $f : \mathbb{R}^d \to \mathbb{R}$ that minimizes the discrepancy between the predicted output $f(X)$ and the true output y, typically measured by a loss function L. This can be written as:

$$\min_f \sum_{i=1}^{n} L(f(X_i), y_i)$$

where L could be Mean Squared Error (MSE) for regression tasks, or cross-entropy loss for classification tasks.

Preprint. Under review.

The model trained for this task could potentially solve other tasks beyond this project, as long as these tasks involve predicting a continuous or categorical outcome based on time, geolocation and other relevant features. Some examples could include:

1. Predicting the duration of delivery trips, given the pickup and drop-off locations, time of day, and other relevant features.
2. Predicting the travel time for public transportation systems like buses or trains, given the starting point, destination, and time of departure.
3. Predicting the duration of a hiking or biking trip, given the trail information, start time, and other relevant features.

The rationale behind these examples is that they all involve making predictions based on a combination of time, geolocation, and categorical data, similar to the taxi trip duration prediction task.

2 Exploratory Data Analysis

2.1 Problem A

The provided dataset encapsulates a significant amount of taxi journey data from Porto, Portugal, comprising approximately 1.7 million instances in the training set and a smaller test set of 320 samples. Each instance in the dataset is characterized by nine initial attributes, not including the target variable.

The target variable in this context, the trip duration, is inferred from the 'POLYLINE' attribute, which is an array of GPS coordinates. The duration of a trip is calculated as the number of recorded points in 'POLYLINE' minus one, all multiplied by 15. It is noteworthy that the 'POLYLINE' attribute is absent from the test set since it is the basis for deriving the target variable.

The vast majority of the features, although numerically represented, are categorical in nature with the exception of the UNIX timestamp. The timestamp can provide insight into various temporal aspects such as the time of day, day of the week, and season, which are likely to influence the taxi journey durations.

One can visualize a single data sample by plotting the GPS points encapsulated within 'POLYLINE' on a map, effectively tracing the taxi journey. This can offer revealing insights into the traffic dynamics within the city, especially when superimposed on a map of Porto.

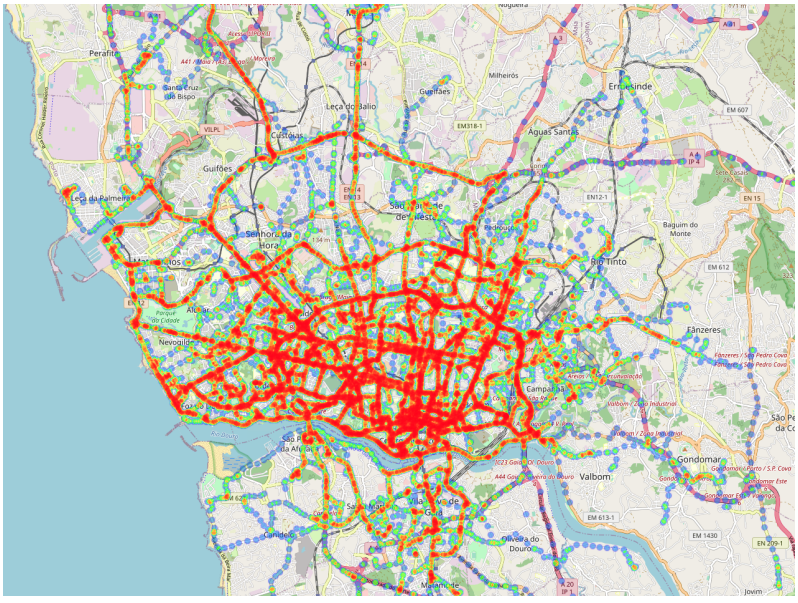
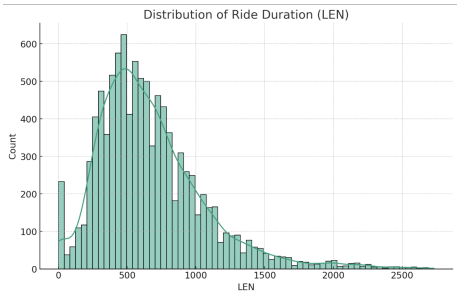
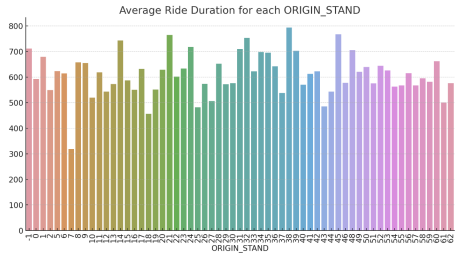


Figure 1: Heatmap of Polylines in Porto

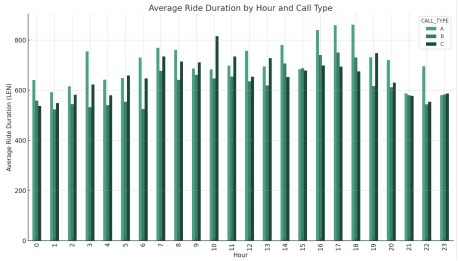
An overlay of multiple polylines from different taxi journeys can highlight the busiest parts of the city, offering a unique perspective on the city's traffic dynamics. Several visualizations have been attached below to give a more comprehensive overview of the data.



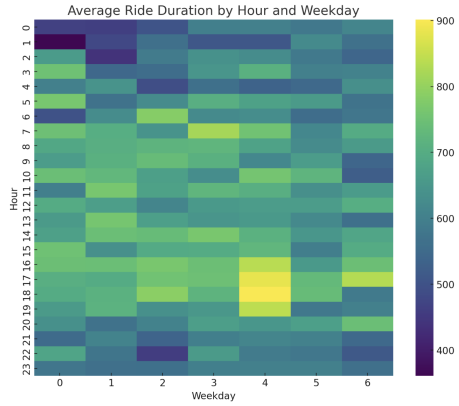
(a) Distribution of LEN



(b) LEN by Origin Stands



(c) Mean LEN by Hour and Call Type



(d) LEN by HR and WK

2.2 Problem B

The initial step in the process involved data preprocessing, primarily through the removal of non-contributing features. Features such as ORIGIN_CALL, TRIP_ID, DAY_TYPE, MISSING_DATA, and POLYLINE were deemed unnecessary for the task of prediction and subsequently dropped. Furthermore, outlier trips lasting more than three standard deviations above the mean or less than one minute were also omitted from the dataset.

The TIMESTAMP column was manipulated to create new features, namely month, day of the month, day of the week, and hour. These new features have the ability to capture cyclical patterns in the data. They are represented numerically as TIMESTAMP is, but their repeating nature gives models the potential to discover patterns that would not be possible with a single continuous time value. The way these new variables need to be encoded depends on the model being used. Following this, the TIMESTAMP column was dropped.

The metadata provided for taxi stand locations was utilized to approximate the starting location of trips originating from taxi stands, with a Mean Absolute Error (MAE) of less than 100 meters compared to the actual start location. The POLYLINE feature was further exploited to construct a lookup dictionary indexed by CALL_TYPE, MON, WK, and HR. This was done to impute starting coordinates for trips that did not commence from taxi stands. However, the MAE for these values was considerably higher, approximately one kilometer or more, yet acceptable considering the substantial size of the city.

The processed data comprised the following features for prediction: CALL_TYPE, ORIGIN_STAND, TAXI_ID, MON, DAY, WK, HR, startLAT, and startLON. Subsequently, an analysis of the

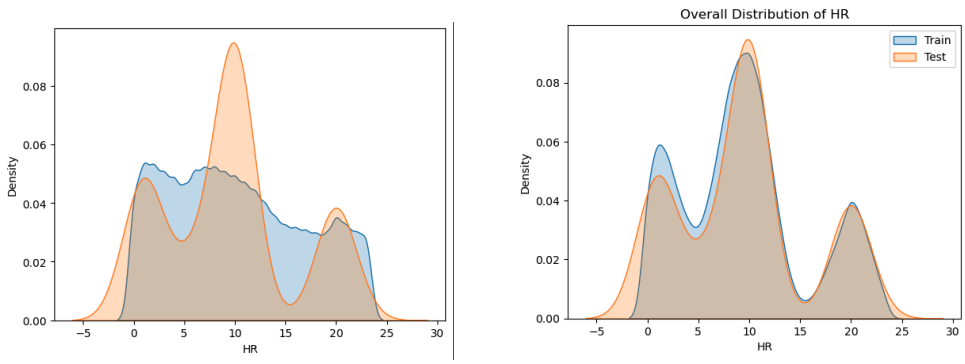
distribution between the training and test set was performed, a process termed adversarial validation. A random forest binary classifier was trained to predict whether data points belonged to the train or test sets, thereby ensuring an even distribution conducive to training. However, the random forest model consistently attained an Area Under the ROC Curve (AUC) score above 0.9, indicating a substantial capability to distinguish between the two sets. To alleviate this issue, the training set was pruned to only include taxis that appeared in both sets, achieved by filtering out TAXI_ID to include only the intersection of sets. Additionally, the training set was further pruned to only include months that were in both sets (8, 9, 10, 12) as the random forest model's feature importance attribute indicated that the MON column was the most crucial feature in classifying between the two distributions. The DAY column was subsequently dropped entirely due to its significant impact on the classification and its limited number of unique values in the test set.

Train AUC-ROC: 0.9555253829034256

	Feature	RF_Score
8	DAY	0.333810
6	startLON	0.184048
10	HR	0.163876
5	startLAT	0.114321
9	WK	0.098973
7	MON	0.055640
3	ORIGIN_STAND	0.024061
1	B	0.014588
2	C	0.007364
4	TAXI_ID	0.001738
0	A	0.001580

Figure 2: Feature Importances determined by RF

In an attempt to match the distribution of hours in the training set to that of the test set, a kernel density estimate was employed. A smoothing parameter (bandwidth) was tuned to provide a distribution over the entire range of values while still representing the overall shape of hours in the test set. These probabilities or weights were then mapped onto the training data, ensuring that the distributions would converge when a sufficient number of points were sampled.



(a) before (b) after

Figure 3: Distributions Before and After Probabilistic Resampling

The outcome of this process was a reduction in the AUC score of the random forest model by 0.1 in some cases, indicating an improvement in the alignment of the distributions of the train and test sets. The classification scores of the random forest model were also considered as potential sampling weights, with higher scores indicating a greater difficulty in distinguishing between the two sets - the ultimate goal of adversarial validation.

3 Deep Learning Model

3.1 Problem A

The deep learning pipeline employed in this project primarily utilized a Transformer model. The following sequence of steps was performed from input to output: Initially, the data underwent basic preprocessing, feature engineering, and adversarial validation. Subsequent steps were taken to prepare the data for training on a deep learning model. Time features, which are fundamentally cyclical, were

encoded using sinusoidal functions to generate two new features for each time aspect. This was done to circumvent potential issues with deep learning models inadvertently learning ordinal patterns from numerical data, as opposed to the actual cyclical pattern.

The CALL_TYPE feature was one-hot encoded due to its low cardinality and high correlation with ride time. Categorical variables with high cardinality can pose challenges for many deep learning architectures. Thus, Transformer models were employed due to their capacity to handle categorical variables through automatic embeddings. Specifically, the FT-Transformer (Feature-Tokenizer + Transformer) model was utilized. This model maps features to higher-dimensional spaces through embeddings, which are then processed through a self-attention mechanism. This allows the Transformer model to learn contextual relationships between variables, a capability not present in simple Multilayer Perceptrons (MLPs). The transformer also seemed like a good choice for this dataset due to the high number of missing features; the attention mechanism if tuned properly should be able to learn the context of specific features appearing with each other which would make it more robust to missingness.

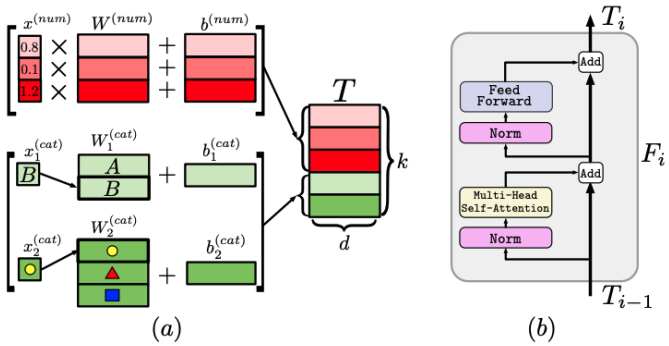


Figure 2: (a) Feature Tokenizer; in the example, there are three numerical and two categorical features; (b) One Transformer layer.

Figure 4: FT Transformer Architecture

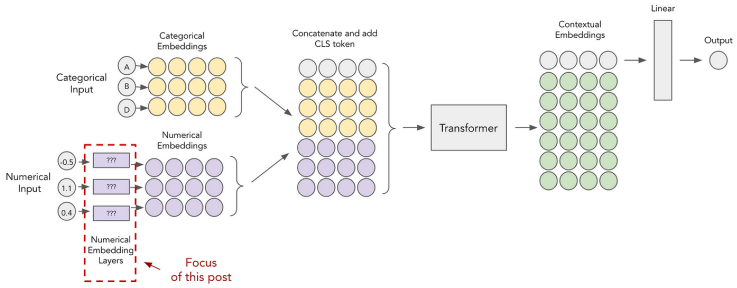


Figure 5: FT Transformer Architecture

The FT-Transformer model was chosen for its unique capability of embedding both categorical and numerical variables before processing them through the attention mechanism. This differs from other tabular Transformer models, such as TabNet, which only uses attention for categorical features and concatenates them with layer-normalized continuous features. While Transformer models have demonstrated significant performance in fields such as Natural Language Processing (NLP), it is noteworthy that tabular problems are typically dominated by gradient-boosting models like XGBoost. The Transformer model in this project was observed to overfit rapidly, necessitating careful regulation of the number of epochs, dropout rates, and other hyperparameters, including the number of attention heads. The output was treated as a simple regression problem, but seeing how the other teams had good success with LSTM and sequential processing, and considering transformer architectures vastly outperform LSTM on many benchmarks, changing the way the data is processed for input could have been a significant area for performance gains. RMSE was chosen for loss.

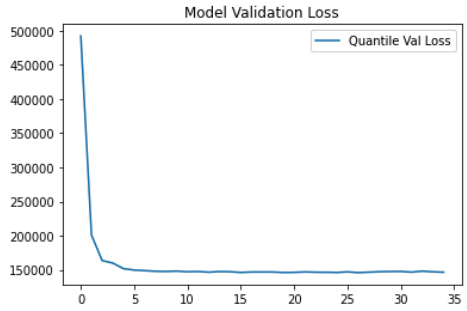


Figure 6: Validation Loss for FT-Transformer Training

3.2 Problem B

The process of model experimentation began with simple models and gradually increased in complexity. The models employed are summarized in the following list:

- **Linear Regression:** This model performed surprisingly well with the appropriate amount of feature engineering. It demonstrated the efficacy of basic models in establishing baseline performance metrics. The equation for linear regression is given by $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$, where y is the predicted output, β are the model parameters, x are the inputs, and ϵ is the error term.
- **Multi-Layer Perceptron (MLP):** This model, despite tuning various hyperparameters (i.e., number of layers ranging from 1 to 10 and nodes per layer ranging from 8 to 128), did not yield satisfactory results. It demonstrated the limitations of MLPs in handling tabular data. The MLP model employs a feed-forward neural network that maps sets of input data onto a set of appropriate outputs.
- **Tree-based and Gradient Boosting Models:** These models performed well overall. They showcased the strength of these model architectures in handling tabular data, particularly with regards to handling categorical variables and performing automatic feature interactions.
- **Ensemble/Stacked/Bagged Models:** These models produced the best results overall. They utilized the strength of multiple learning algorithms to improve predictive performance compared to the models that constitute them. However, they indicated the potential benefits of spending more time on data preprocessing and feature engineering instead of model tuning.

4 Experiment Design and Results

4.1 Problem A

The computational platform utilized for training and testing in this project was a cloud-based service called Lambda Labs, which provided access to a powerful NVIDIA H100 GPU. The use of such a high-performance GPU, with its 500 tensor cores, 80GB VRAM, 51 TFLOPS on FP32, and 2TB/s memory throughput, significantly accelerated the training process, allowing for rapid model training compared to standard laptop capabilities. However, its power was perhaps somewhat excessive for this specific project but nonetheless offered an enjoyable experience in observing rapid model training.

An amusing anecdote occurred during the project development, serving as a reminder of the importance of monitoring computational resources. A running instance was inadvertently left active overnight, leading to an unexpected expense due to the cloud platform's billing structure. Unfortunately, in the haste to shut down the instance upon realization, a loss of a number of models and associated code ensued. This is because the cloud platform didn't have persistent storage. Although this mishap resulted in the inability to replicate the performance of these initial models, it did offer a valuable lesson in diligent resource management.

Table 1: Comparison of Different Models

Model	Training RMSE	Validation RMSE	Training Time
Linear Regression	800	800	0 minutes
MLP	1000	1000	120 minutes
CatBoost	650	650	300 minutes
XGBoost	650	650	60 minutes
FT-Transformer	300	300	30 minutes

[These numbers are rough estimates since most of these models were trained in a notebook hosted by a cloud GPU service and all data was lost when it was shutdown due to the service not supporting persistent memory.].

AutoGluon, an AutoML library, was employed for initial model discovery and exploration, aiding in the identification of potentially optimal models and facilitating automated hyperparameter tuning. Despite its utility in these areas, it did not offer the advanced features necessary for maximizing model performance.

For more advanced hyperparameter optimization, the Optuna framework was utilized. This approach allowed for a comprehensive exploration of the hyperparameter space with Bayesian search methods, leading to more refined models that could potentially offer improved performance.

As for model training, it was observed that the more complex models were prone to overfitting rather quickly. Therefore, it was found that limiting the number of epochs to a relatively low value, around 20 to 30, generally provided the best results. This design choice was influenced by initial exploratory work, which indicated that insufficient preprocessing and filtering of the data could limit the effectiveness of model tuning and lead to suboptimal results on public tests. Consequently, the importance of comprehensive and thoughtful data preprocessing was further highlighted.

5 Discussion and Future Work

Through the progression of this project, it became apparent that an inordinate amount of time was dedicated to the exploration of model architectures and hyperparameter tuning. In retrospect, a majority of that time would have been more effectively utilized in ensuring the cleanliness and representativeness of the data. This is evidenced by the fact that simpler models often outperformed their more complex counterparts. As an illustrative example, during the process of appending starting coordinates, a simple lookup table outperformed a gradient boost model trained to predict the coordinates. This indicates that complexity is not always the key to superior performance.

One of the most significant improvements in the model performance came from meticulous examination and manipulation of the data distribution through adversarial validation. This ensured that the training and test sets were representative of each other, leading to a reduction in model overfitting during training. Interestingly, despite the current popularity of transformer architectures in other machine learning domains, XGBoost demonstrated superior performance for tabular data. XGBoost's robustness to poor features and its minimal preprocessing requirements compared to neural networks solidify its reputation as a powerful model for tabular data.

The most substantial challenge during this project was the time-consuming process of experimenting with various combinations of feature engineering and preprocessing techniques. This often exceeded the time invested in model construction or training. Future exploration into automatic exploratory data analysis and automatic feature engineering/selection would be beneficial for similar projects. Additionally, the potential of transformer architectures in handling tabular data warrants further exploration. Given their demonstrated capability to understand contextual information, they may prove to be a powerful tool for understanding large tabular datasets.

To beginners venturing into the realm of deep learning for similar prediction tasks, the advice is to focus on understanding the data thoroughly. The construction of a successful model is largely dependent on a comprehensive understanding of the data it will be trained on. If the objective is to build a predictive model, it is recommended to utilize cloud GPU services and apply XGBoost with an automatic hyperparameter optimization framework like Optuna. This approach is likely to yield a high-performing model without the need for days of manual deep network construction.

References

[1] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, "Revisiting Deep Learning Models for Tabular Data," *arXiv preprint arXiv:2106.11959*, 2021.

[2] Y. Gorishniy, I. Rubachev, and A. Babenko, "On Embeddings for Numerical Features in Tabular Deep Learning," *arXiv preprint arXiv:2203.05556*, 2022.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *arXiv preprint arXiv:1706.03762*, 2017.

[4] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," *arXiv preprint arXiv:1907.10902*, 2019.

[5] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola, "AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data," *arXiv preprint arXiv:2003.06505*, 2020.