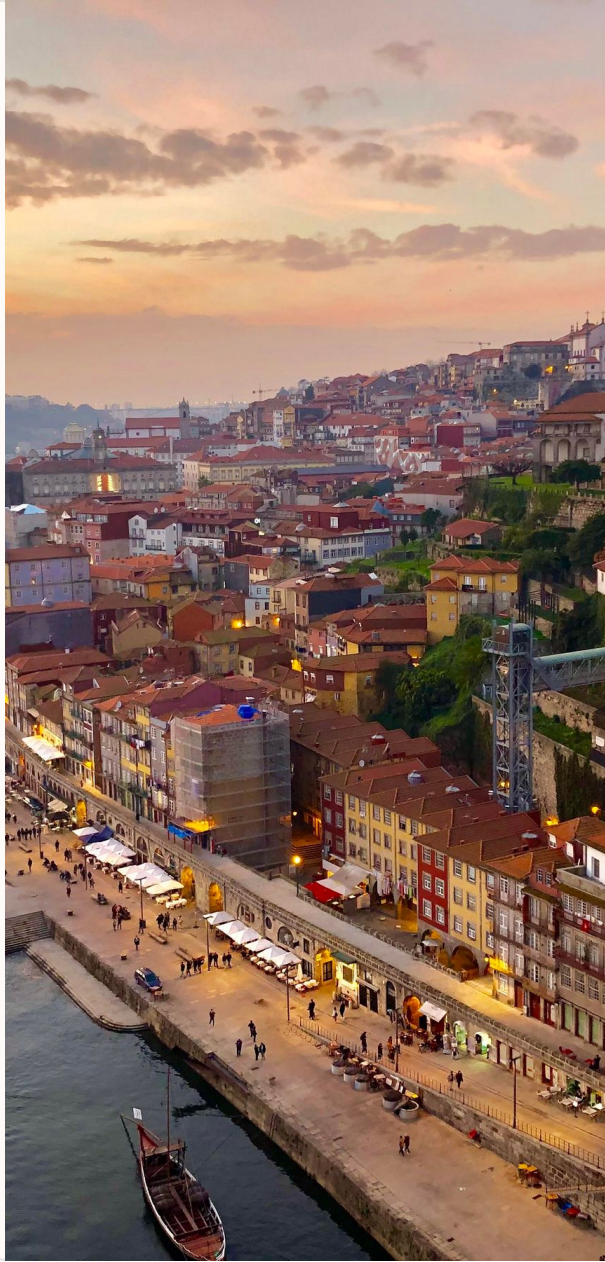
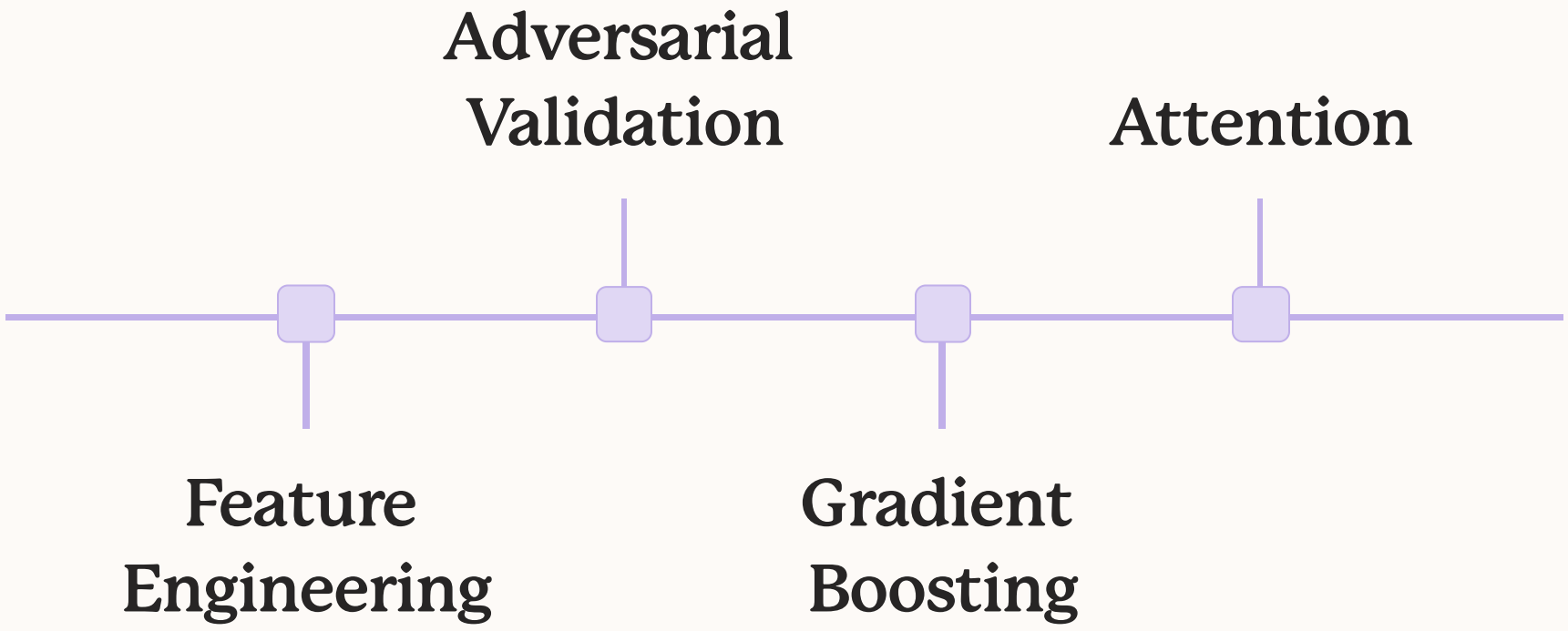
Predicting Taxi Ride Time in Porto

Creating machine learning models to estimate taxi trip durations in Porto, Portugal using techniques in data preprocessing and deep learning. I utilized transformer architectures and gradient boosting models such as XGBoost to make predictions. Through this project, I learned the importance of data understanding and processing, adversarial validation, and model selection.

K **by Kai Breese**



Key Words


Adversarial Validation
Attention
Feature Engineering
Gradient Boosting

Data Exploration - Overview

1

1.7 Million Instances

The dataset contains approximately 1.7 million instances in the training set and a smaller test set of 320 samples.

2

Polyline Attribute

The target variable, trip duration, is inferred from the 'POLYLINE' attribute, which is an array of GPS coordinates.

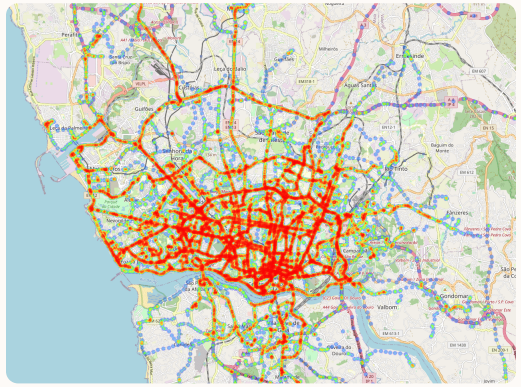
3

Categorical Features

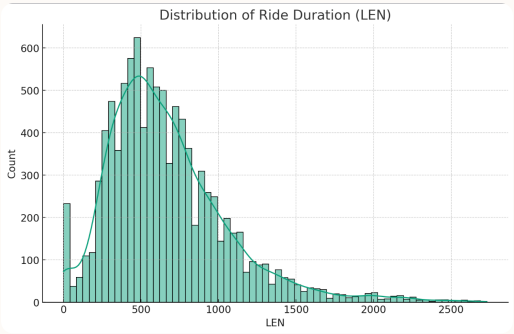
The vast majority of the features are categorical in nature, with the exception of the UNIX timestamp.

Data Exploration - Visualization

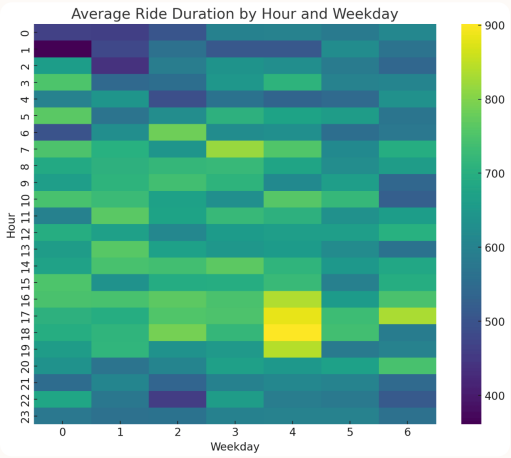
Where are all the taxis?



How long are rides?



What influences it?



Data Preprocessing

1 Features Removed

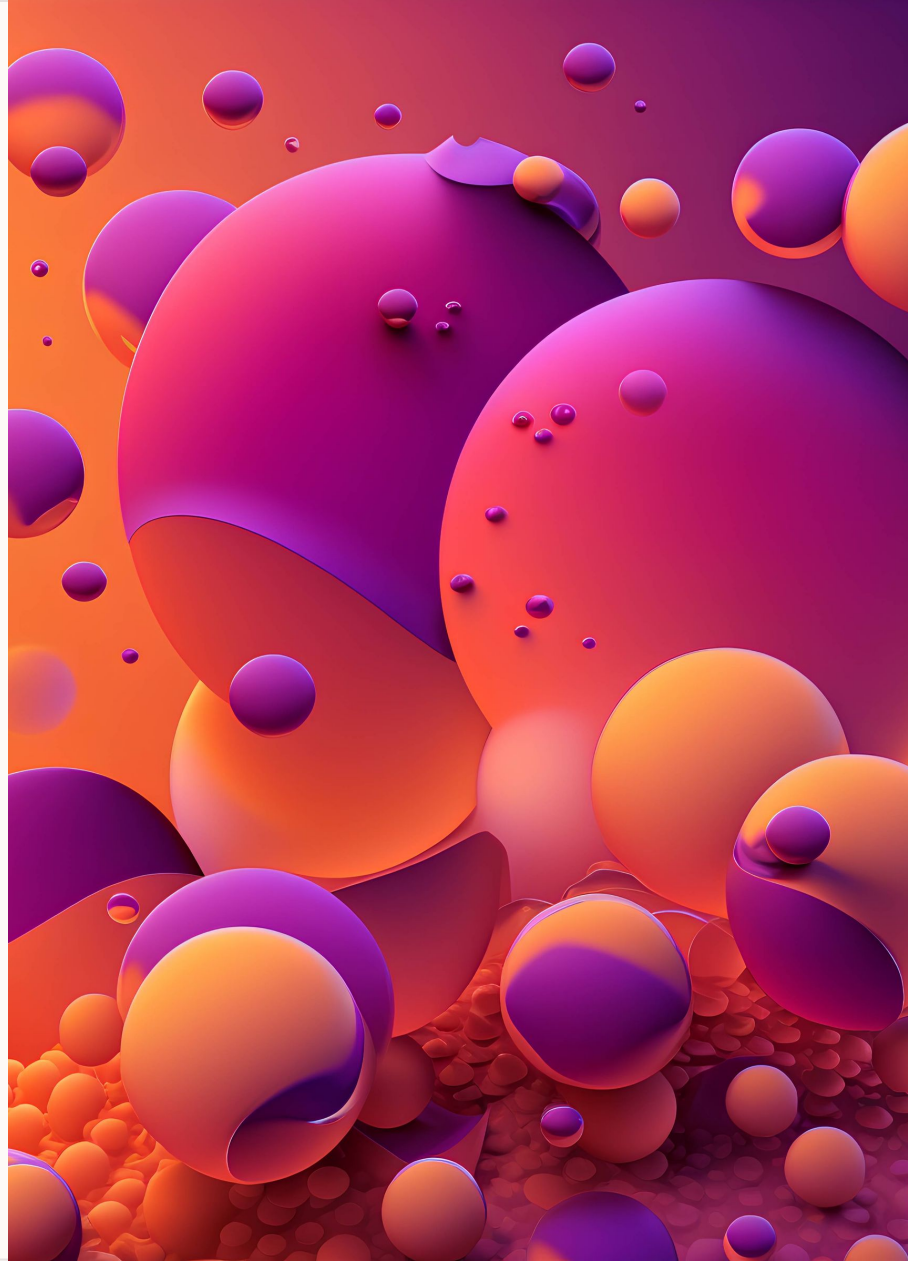
All non-contributing features were removed from the dataset, including the end coordinates of the trip.

2 Outliers Removed

Trips lasting for less than one minute or beyond three standard deviations above the mean were eliminated from the dataset.

3 Feature Engineering

New features, including month, day-of-the-month, and day-of-the-week were created from TIMESTAMP column data.



Feature Engineering

Starting Locations

The POLYLINE feature was used to construct a lookup dictionary indexed by CALL_TYPE, MON, WK, and HR to impute starting coordinates.

Time is Cyclical

Time features were encoded using sinusoidal functions to capture the cyclic nature of the features and allow models to learn patterns in them.

Adversarial Validation

The training data should resemble the test data to ensure the test set is representative and prevent overfitting. A discriminator was trained for this.

Adversarial Validation

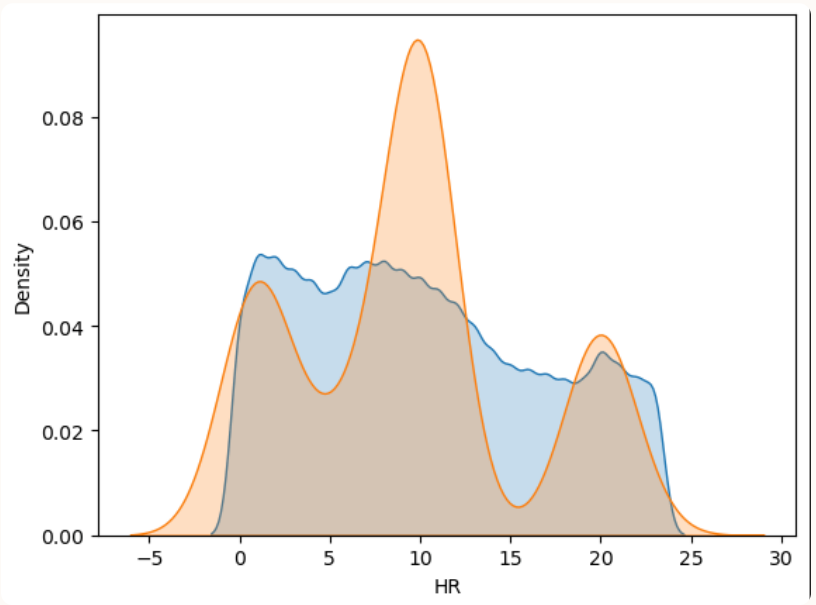
- Random forest binary classifier trained
- Predict whether data points belong to train or test sets
- Achieve even distribution between train and test sets
- Address high AUC score (>0.9) observed in random forest model
- Filter train set to match distribution of hours between train and test sets

AUC-ROC: 0.95552538290

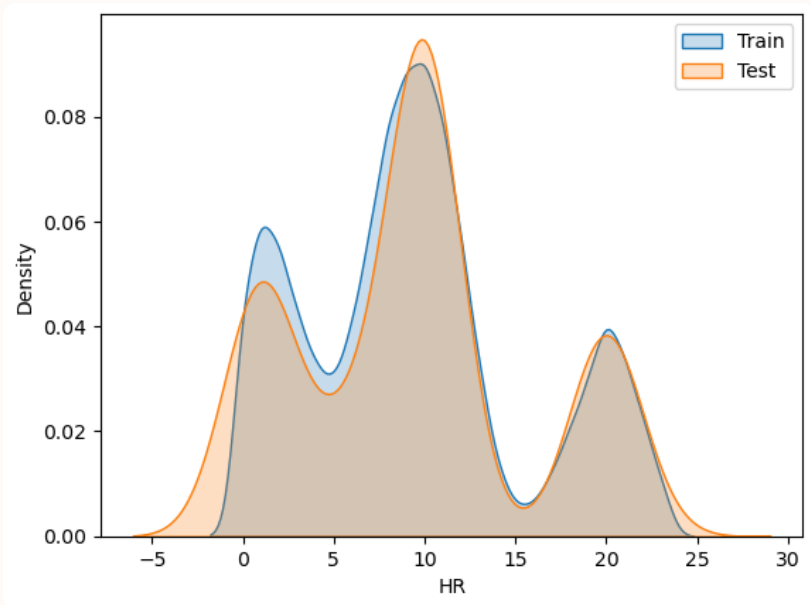
Feature	RF_Score
DAY	0.333810
startLON	0.184048
HR	0.163876
startLAT	0.114321
WK	0.098973
MON	0.055640
ORIGIN_STAND	0.024061
B	0.014588
C	0.007364
TAXI_ID	0.001738
A	0.001580

Engineering Tricks

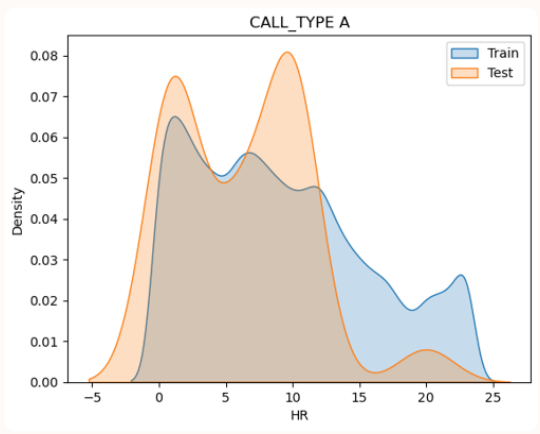
Probabilistically sampling from the training data to better match the test set was used by creating KDE to generate probabilities for each value in features with greatly different distributions. I chose to sample based on HR, subdivided by CALL_TYPE.

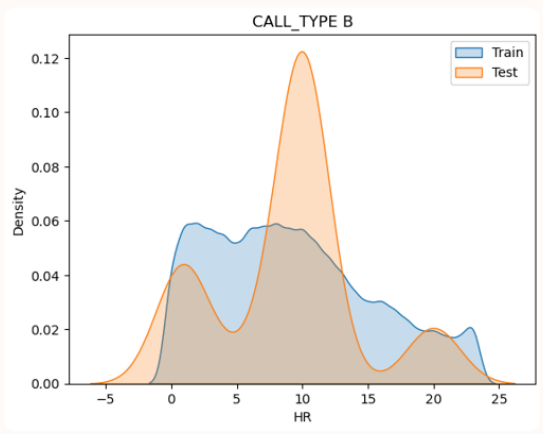


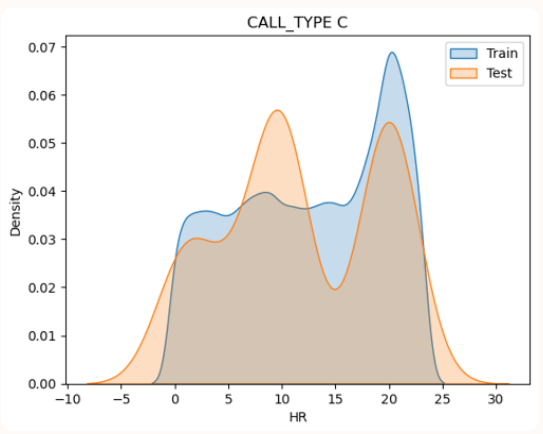
Original Distribution

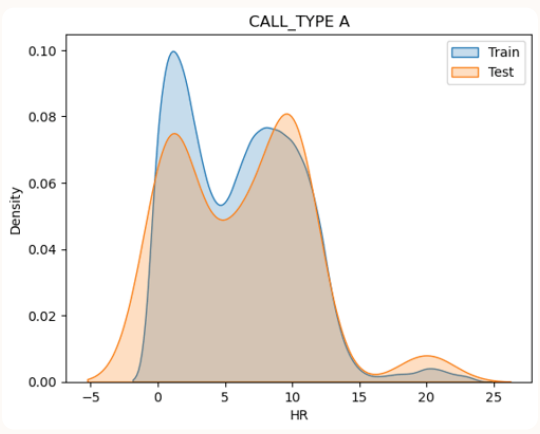


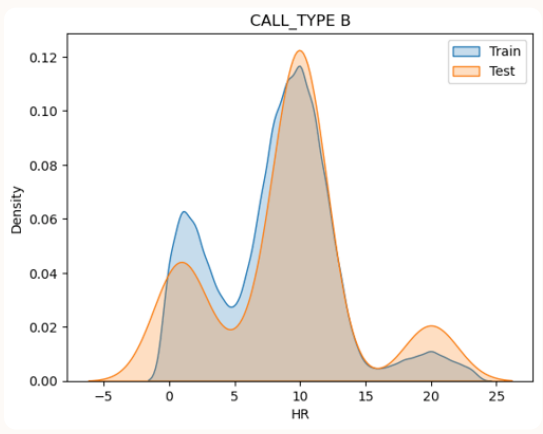
After resampling

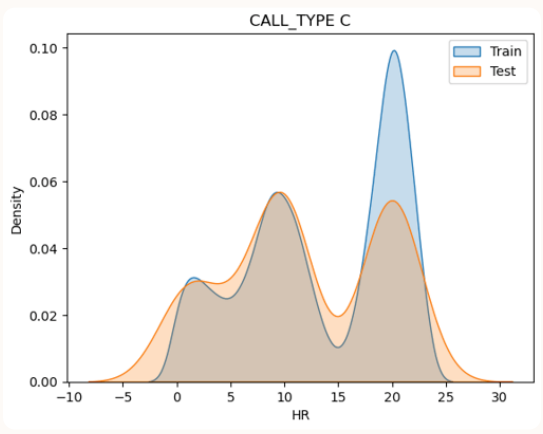












[1] top is before, bottom after

Deep Learning Model: Transformer

Embeddings

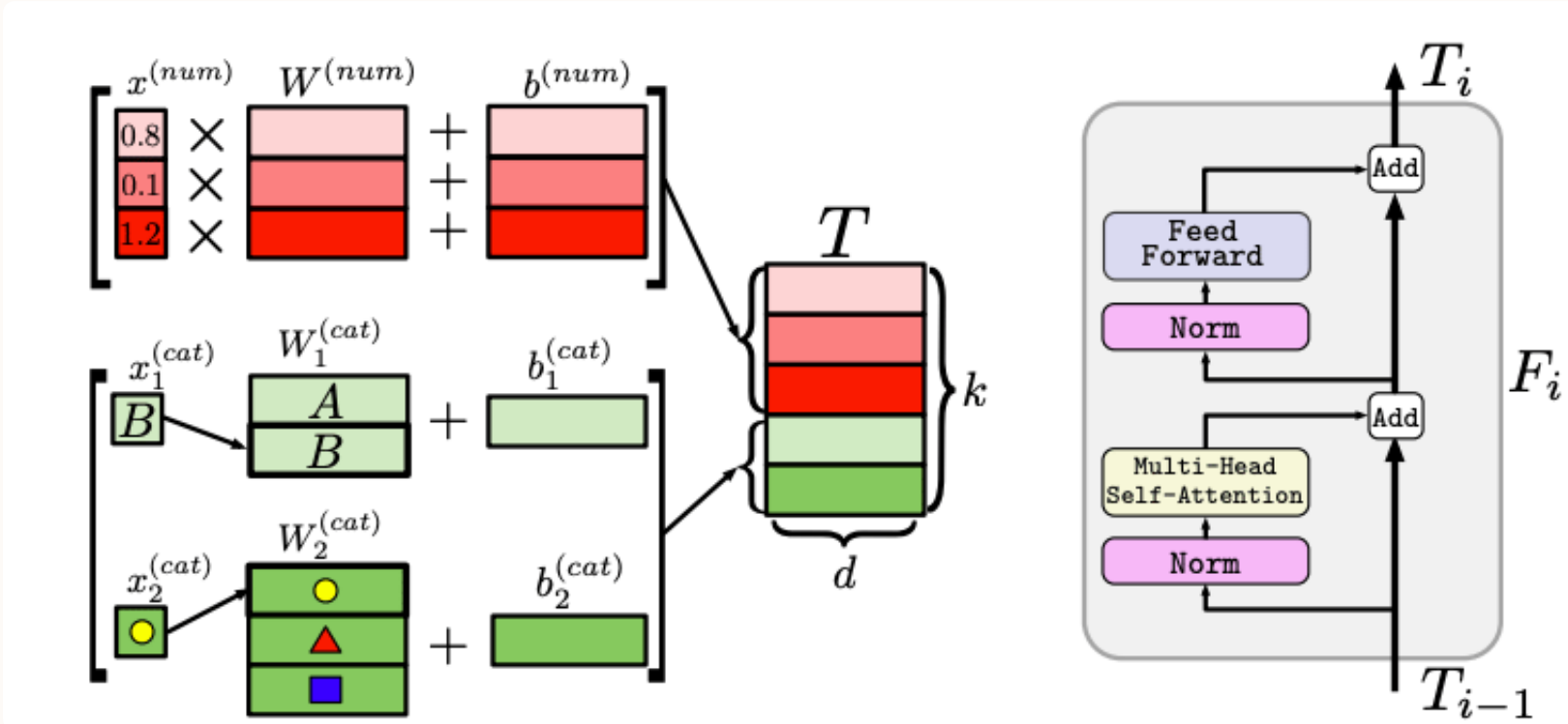
- Maps features to higher-dimensional spaces
- Handles both categorical and numerical variables
- Reduces cardinality for high 'vocabulary' features

Attention

- Learns contextual information about features, numerical and categorical
- Could improve performance on highly missing data
- Improves performance of simple MLPs

Pitfalls

- Easy to overfit without careful tuning of epochs and dropout
- Potentially sensitive to poorly chosen and engineered features
- Still not as good as gradient boosting for tabular data



[1]The transformer did not end up being the best performing model, even with its extremely low validation loss. Model was also lost after closing a cloud GPU instance.

Other Experiments

Linear Regression

Simpler models like Linear Regression performed surprisingly well, beating out the MLP in some cases (with little tuning) and is incredibly easy to implement.

Tree-Based and Gradient Boosting Models

Tree-Based and Gradient Boosting models were also used and demonstrated strong performance in tabular data. Industry standard.

Ensemble Models

Ensemble models produced the best results overall in this project. These were made by combining other types of models like the previously discussed.

Challenges and Future Work

1 Overfitting

A significant challenge was encountered during the project wherein more complex models would overfit the training data. They would have exceptionally low training loss, but the validation and public test scores were terrible.

2 Automatic Analysis

Exploration into automatic exploratory data analysis and automatic feature engineering/selection could be beneficial for speeding up this process as it took the majority of time in this project.

3 Transformers, again

Although the transformer didn't give the best results overall, it was still promising and showed a noticeable improvement over the simple MLP. Trying more hyperparameter optimization and different architectures could yield SOTA.