ASSIGNMENT 2: FINAL REPORT

1. DATASET

For our assignment, we chose to focus on **predicting rideshare fares** using features from rideshare datasets because this task offers quantifiable insights and practical applications, unlike alternatives such as social media engagement, which tend to be more subjective. Initially, we explored various datasets such as healthcare data for predicting patient outcomes and diseases using ECG or EKG signals, housing data for property price prediction, energy consumption data for optimization trend analysis, and e-commerce data for purchase likelihood. However, we chose rideshare datasets because they provided a clear and measurable outcome that aligned well with the scope of our project. We decided to select the NYC Taxi and Limousine Commission (TLC) trip record data due to its detailed features, extensive size, and relevance to our task. Specifically, we used the High Volume For-Hire Vehicle (HVFHV) trip data from January and February 2022. Each row in this dataset represents an individual trip dispatched by NYC-licensed High Volume FHV bases such as Uber and Lyft, with approximately 30 million data points available combining both the datasets. Although we initially considered including more months, computational constraints led us to focus on these two months, which still provided a robust subset for our analysis.

The dataset is publicly available on the NYC TLC website[10], it has a variety of datasets for different vehicle types and time periods. To familiarize ourselves with the format of the data, which was stored in Parquet files, we referenced a helpful Kaggle project, NY_Trip_Datexland_Price_Model[11]. This resource helped us with processing Parquet files and provided insights into effective feature selection for fare prediction.

The dataset contains 17 features, which gives a wide range of information about each trip. The features included:

- *hvfhs_license_num*: The license number of the high-volume FHV service, such as "HV0003" for Uber or "HV0005" for Lyft.
- *dispatching_base_num*: The TLC Base License Number of the base dispatching the trip.
- *pickup_datetime* and *dropOff_datetime*: Timestamps for when the trip started and ended.
- *PULocationID* and *DOLocationID*: Taxi Zone IDs for the pickup and drop-off locations.
- *originating_base_num*: The base number of the entity that received the original trip request.
- *request_datetime* and *on_scene_datetime*: Timestamps indicating when the trip was requested and when the driver arrived at the pickup location.
- *trip_miles*: The total distance of the passenger trip.
- *trip_time*: The total duration of the trip in seconds.
- *base_passenger_fare*: The fare before tolls, tips, and additional fees.

- *tolls*, *bcf*, *sales_tax*, *congestion_surcharge*, and *airport_fee*: Various charges associated with the trip.
- *tips*: Gratuity paid by the passenger.
- *driver_pay*: The amount earned by the driver, excluding tips and after deductions.
- Binary flags such as *shared_request_flag*, *shared_match_flag*, *access_a_ride_flag*, *wav_request_flag*, and *wav_match_flag*: Indicators of whether the ride was shared, wheelchair-accessible, or part of NYC's Access-A-Ride program.

Our exploratory data analysis revealed interesting patterns and guided our model design. We found that the *base_passenger_fare* ranged from -$520.11 to $4,995.96, with a mean of $20.41 and a median of $16.09. To ensure meaningful predictions, we filtered out fares below $0 and above $100, as these were likely errors or outliers. The dataset had various attributes that were analyzed during preprocessing. For example, *trip_miles* had a mean of 4.53 miles, while *trip_time* averaged 1,049.89 seconds. Other important features included *tips*, which averaged $0.82, and *driver_pay*, which averaged $15.66. Extreme values in features like *trip_miles*, which ranged from 0 to 361.11, and *trip_time*, which varied from 0 to 99,152 seconds, were considered and filtered out during preprocessing to avoid skewing model results. Below is a summary of the key numeric features used in our analysis after filtering outliers and unreasonable entries. The table displays the mean, standard deviation, and distribution of values for features, which were crucial for identifying potential outliers and setting bounds during our data preprocessing steps.

Summary Statistics for Numeric Features										
	request_datetime	**trip_ miles**	**base_passenger _fare**	**tips**	**bcf**	**driver _pay**	**tolls**	**airport _fee**	**sales _tax**	**congestion _surcharge**
mean	2022-02-01 3:46:25	4.29	999.15	19.15	0.79	0.6	14.95	0.8	0.14	1.69
min	2021-12-31 22:55:05	0	0	0	0	0	-28.65	0	0	0
25%	2022-01-17 16:03:32	1.52	557	10.17	0	0.3	7.67	0	0	0.87
50%	2022-02-02 7:16:47	2.76	864	15.84	0	0.48	12.08	0	0	1.37
75%	2022-02-15 16:33:50	5.46	1,313	24.15	0	0.75	19.24	0	0	2.13
max	2022-03-01 0:10:00	50.13	2,999	99.99	200	7.74	303.21	80	5	22.92
std	NaN	4.22	580.14	12.04	2.21	0.42	9.68	3.12	0.57	1.13

2. PREDICTIVE TASK

The predictive task we chose was to predict rideshare fares in New York City. To predict ride fares using this dataset, the task involves building a regression model, where the target variable is the base passenger fare amount. Based on feature analysis, the key features include trip time, trip distance, time of day, day of the week, and other factors such as additional charges. The goal is

to uncover patterns that influence fares and use these insights to create an accurate predictive model.

Model performance will be evaluated using evaluation metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), which measure the model's prediction accuracy in real-world fare units. A baseline evaluation will also assess accuracy by hour and compare predictions across different rideshare services like Uber and Lyft. A simple linear model based on a single feature will give a clear indication of how well one variable explains the fare, making it a useful threshold. The baseline for Uber and Lyft will simply be the average fare for each service, providing a straightforward point of comparison. For example, if the rideshare is for Uber, the predicted fare would be the average fare of all Uber rideshares. These baselines help us assess the underlying patterns in the data and establish a threshold for us.

For the simple predictor based on the hour, we used the request hour as a feature and tried to predict the base fares. The average fares for each hour are shown below:

```
Average Fares by Hour:
 request_hour
0     19.827822
1     19.899676
2     20.000037
3     20.370871
4     22.584275
5     22.841371
6     21.962737
7     20.411652
8     19.144019
9     18.311364
10    18.279954
11    18.523328
12    18.727605
13    18.721559
14    18.721960
15    18.826096
16    18.583031
17    19.007457
18    19.106058
19    18.317436
20    18.023411
21    18.232088
22    19.340709
23    20.032108
```

The RMSE we got for this model was 11.99, and the MAE was 8.91. These are relatively high, as expected, because we are attempting to predict a complex and multifaceted problem based on just a single feature: the hour of the request. This serves as a simple baseline for comparison with more complex models.

For the model based on the rideshare service, we calculated the average base fares for Uber (0) and Lyft (1) separately:

```
Average Fares by Service:
 hvfhs_license_num_encoded
0    19.271395
1    18.820281
```

Using this basic approach, we predicted the average fares based on the service (Uber or Lyft). The RMSE for this model was 12.03, and the MAE was 8.95. While this is a simple model, it provides a basic benchmark for comparison. It also highlights the need for incorporating more detailed and diverse features to accurately capture the complexity of fare prediction.

Data preprocessing involved cleaning the dataset to remove outliers or invalid entries, such as trips with zero fares or unrealistic distances. Features were engineered, including calculating the day of the week and hour of the day, as well as encoding rideshare service details. The model's validity was assessed by splitting the data into training and test sets, and visualizing performance using graphs to ensure it generalized well across diverse scenarios.

To prepare the data for modeling, we performed feature engineering and preprocessing. We label encoded license numbers to analyze differences between service providers like Uber and Lyft, with Uber encoded as 0 and Lyft encoded as 1. We clipped the *base_passenger_fare* to range between 0 and 100, as fares exceeding $100 likely indicate trips outside New York, while negative fares suggest refunded rides. Similarly, we limited *trip_miles* to a range of 0 to 100 and *trip_time* to a range of 0 to 3000 to eliminate outliers. Several columns, including *PULocationID*, *DOLocationID*, *on_scene_datetime*, *pickup_datetime*, *dropoff_datetime*, and various flags, were removed as they did not contribute significantly to the model's performance. We have a more detailed explanation of the final features selected in the model section, as tuning and testing with different models revealed which features were most effective in minimizing the RMSE.

We included *request_datetime* in our analysis to examine whether predictions varied by date and time. Since the data was already in a date format, we could easily extract the hour and day of the week using *df['request_datetime'].dt.hour*. This allowed us to convert *request_datetime* into numerical values for the hour and day of the week. We decided to remove the *request_datetime* column because it was a complex string, and we had already extracted the key information from it. Numerical features, including trip miles, trip time, and fare components, were standardized to improve model performance. These features had units and differences in scale could lead to uneven importance on features with larger values, resulting in a biased model. Scaling was performed using the StandardScaler, with the scaled features applied consistently across all subsets of the data. We split the dataset into training, validation, and test sets, ensuring a rigorous evaluation process. The data was split based on the dates within each month: dates 1–20 were

used for training, dates 21–25 for validation, and dates 26-30 for testing. This was done to make sure that the model is evaluated on data that simulates a real-world scenario where future data points are not used in training.

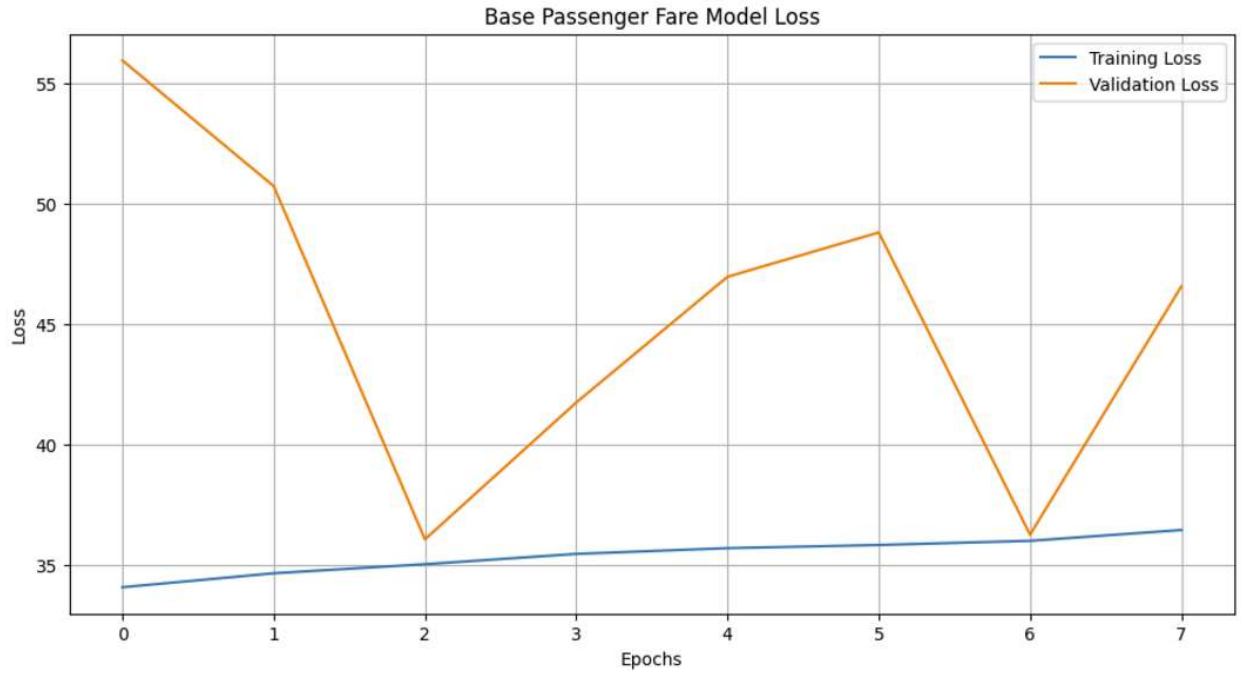
3. MODEL

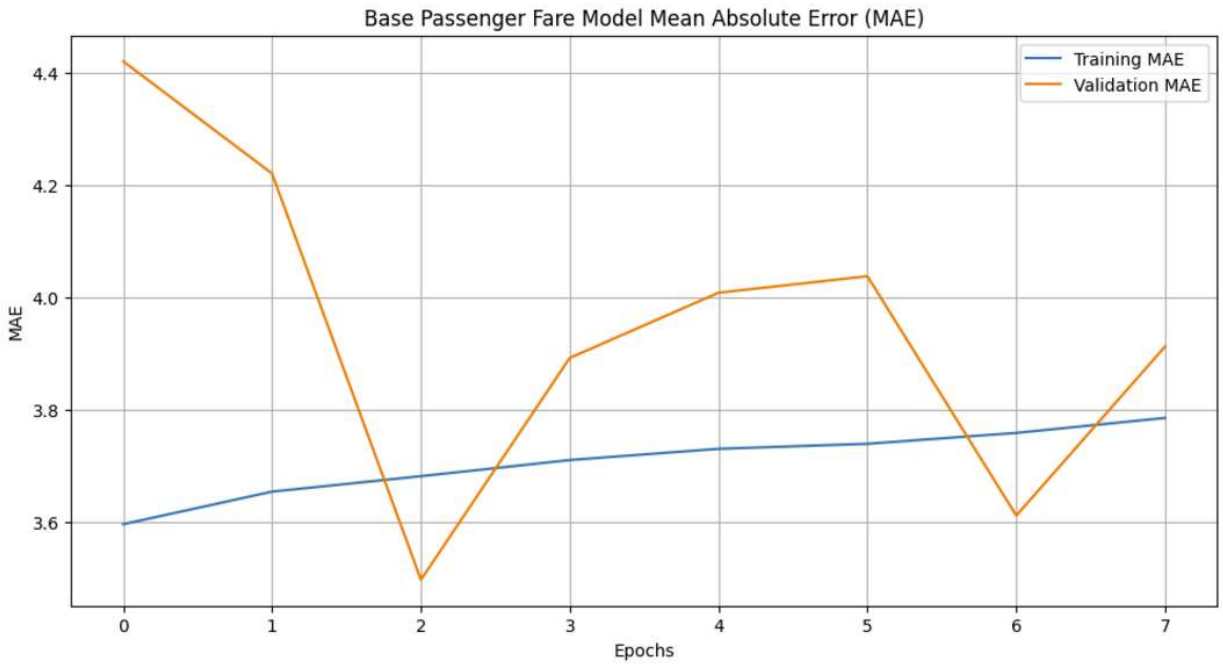
We decided that using a neural network model would be the best choice for predicting fare prices because of its flexibility in capturing complex and non-linear relationships between the features. Unlike models like decision trees, random forests, or gradient boosting, which may struggle with dynamic interactions between variables, a neural network can learn intricate patterns in the data. This is especially important for predicting fares, where multiple factors like trip distance, time of day, location and trip time interact in non-obvious ways. One of the strengths of the neural network model is its ability to adapt and scale with the complexity of the data. By adjusting the architecture, such as adding more layers or neurons, we can allow the model to capture even more subtle patterns that might otherwise be missed. We also encountered some of the challenges that might arise with using a neural network, particularly its long training time and the risk of overfitting. To explore different model options and understand the best approach for our problem, we referenced a textbook chapter on Feature-Rich Recommender Systems[1], which provided insights into using neural networks in similar contexts.

Initial Model

In our initial model, we modularized our code by separating the model-building process into a function and utilized hyperparameter tuning with grid search to optimize a neural network model using *tensorflow.keras*. However, after implementing grid search, we encountered a lot of major issues. The loss value was about 9, which was too high given that the base passenger fares ranged between 0 and 100. This was also not significantly lower compared to the baseline models. We suspected that the high loss value was caused by overfitting. For our grid search, we experimented with different configurations of neurons, such as (128, 64, 32) and (64, 32), as well as different optimizers like Adam and RMSprop. We also tested different dropout values to identify the best combination for our model. Due to the large dataset and extended training times, we limited the number of epochs to 5 for the grid search, which allowed us to evaluate different hyperparameter configurations more efficiently. These were our results upon running the model on the best parameters from the grid search:

```
Epoch 1/20
604250/604250 ━━━━━━━━━━━━━━━━━━━━ 986s 2ms/step - loss: 37.8866 - mae: 3.6881 - val_loss: 55.9547 - val_mae: 4.4199
Epoch 2/20
604250/604250 ━━━━━━━━━━━━━━━━━━━━ 983s 2ms/step - loss: 34.3787 - mae: 3.6406 - val_loss: 50.7300 - val_mae: 4.2201
Epoch 3/20
604250/604250 ━━━━━━━━━━━━━━━━━━━━ 1048s 2ms/step - loss: 34.8581 - mae: 3.6751 - val_loss: 36.0494 - val_mae: 3.4964
Epoch 4/20
604250/604250 ━━━━━━━━━━━━━━━━━━━━ 985s 2ms/step - loss: 35.2704 - mae: 3.6974 - val_loss: 41.7358 - val_mae: 3.8917
Epoch 5/20
604250/604250 ━━━━━━━━━━━━━━━━━━━━ 979s 2ms/step - loss: 35.6765 - mae: 3.7234 - val_loss: 46.9632 - val_mae: 4.0078
Epoch 6/20
604250/604250 ━━━━━━━━━━━━━━━━━━━━ 1092s 2ms/step - loss: 35.8123 - mae: 3.7386 - val_loss: 48.8038 - val_mae: 4.0371
Epoch 7/20
604250/604250 ━━━━━━━━━━━━━━━━━━━━ 1165s 2ms/step - loss: 35.9023 - mae: 3.7511 - val_loss: 36.2439 - val_mae: 3.6109
Epoch 8/20
604250/604250 ━━━━━━━━━━━━━━━━━━━━ 1148s 2ms/step - loss: 36.3856 - mae: 3.7812 - val_loss: 46.5604 - val_mae: 3.9120
139878/139878 ━━━━━━━━━━━━━━━━━━━━ 126s 903us/step
Target: base_passenger_fare | MSE: 8.86 | MAE: 5.16
```





The issue can be seen when examining the training loss, training MAE, validation loss, and validation MAE during training. As shown in the log output, the training loss and MAE steadily increase after a few epochs, while the validation loss fluctuates significantly which indicates instability in the model generalization. For example, at epoch 8, the training loss was 36.39, but the validation loss was highly variable, ranging from 36.05 to 48.8. This suggests that the model was starting to overfit the training data, leading to poor performance on the unseen validation data. The graphs further illustrate this, where the training loss and MAE continually rise, while the validation loss behaves erratically, showing that the model was struggling to generalize.

Additionally, the extended training time required by the grid search (8-10 hours) due to the large dataset and many hyperparameter combinations led to further problems. The increased complexity by exploring multiple hyperparameters—such as neurons, batch sizes, dropout rates, and optimizers—may have caused the model to become too sensitive to the training data. This led to higher variance in performance across epochs, as shown in the erratic validation results. The excessive training time also made it difficult to quickly detect and address overfitting, contributing to the instability.

The pattern of increasing training loss and erratic validation loss led us to use regularization techniques such as adjusting the dropout rate, using a more streamlined model architecture, and carefully monitoring overfitting in order to improve the model's performance and stability.

Intermediate Model

The neural network model that ultimately had the best results consists of three hidden layers with decreasing numbers of neurons (128, 64, and 32), utilizing ReLU activation functions. We added

BatchNormalization and Dropout layers after each hidden layer to improve generalization and minimize overfitting. The output layer uses a linear activation function since this is a regression problem. The model is compiled using the Adam optimizer, with mean squared error (MSE) as the loss function and mean absolute error (MAE) as a metric. In our model, we used early stopping to prevent overfitting. Early stopping monitors the validation loss during training and halts training if the validation loss does not improve for a set number of epochs (patience). This helps avoid overfitting by stopping the training process before the model starts to memorize the training data instead of generalizing well to unseen data. We also used dropout as a regularization technique to prevent overfitting by randomly dropping neurons during training, forcing the model to rely on multiple pathways and not become too dependent on any one feature. BatchNormalization was included to stabilize and accelerate training by normalizing the activations of each layer.

The final model achieved improved performance with reduced overfitting, as shown in the training logs:

```
Epoch 1/20
604250/604250 ━━━━━━━━━━━━━━━━━━━━ 482s 795us/step - loss: 0.2409 - mae: 0.3185 - val_loss: 1.0539 - val_mae: 0.2704
Epoch 2/20
604250/604250 ━━━━━━━━━━━━━━━━━━━━ 483s 798us/step - loss: 0.2269 - mae: 0.3103 - val_loss: 0.2400 - val_mae: 0.2755
Epoch 3/20
604250/604250 ━━━━━━━━━━━━━━━━━━━━ 496s 820us/step - loss: 0.2259 - mae: 0.3098 - val_loss: 0.4286 - val_mae: 0.2673
Epoch 4/20
604250/604250 ━━━━━━━━━━━━━━━━━━━━ 494s 817us/step - loss: 0.2251 - mae: 0.3094 - val_loss: 0.8721 - val_mae: 0.2829
Epoch 5/20
604250/604250 ━━━━━━━━━━━━━━━━━━━━ 500s 827us/step - loss: 0.2243 - mae: 0.3091 - val_loss: 0.5640 - val_mae: 0.2908
Epoch 6/20
604250/604250 ━━━━━━━━━━━━━━━━━━━━ 516s 854us/step - loss: 0.2243 - mae: 0.3092 - val_loss: 0.7594 - val_mae: 0.2691
Epoch 7/20
604250/604250 ━━━━━━━━━━━━━━━━━━━━ 516s 852us/step - loss: 0.2238 - mae: 0.3087 - val_loss: 0.5923 - val_mae: 0.2870
```

The post-training evaluation metrics on the test data were:

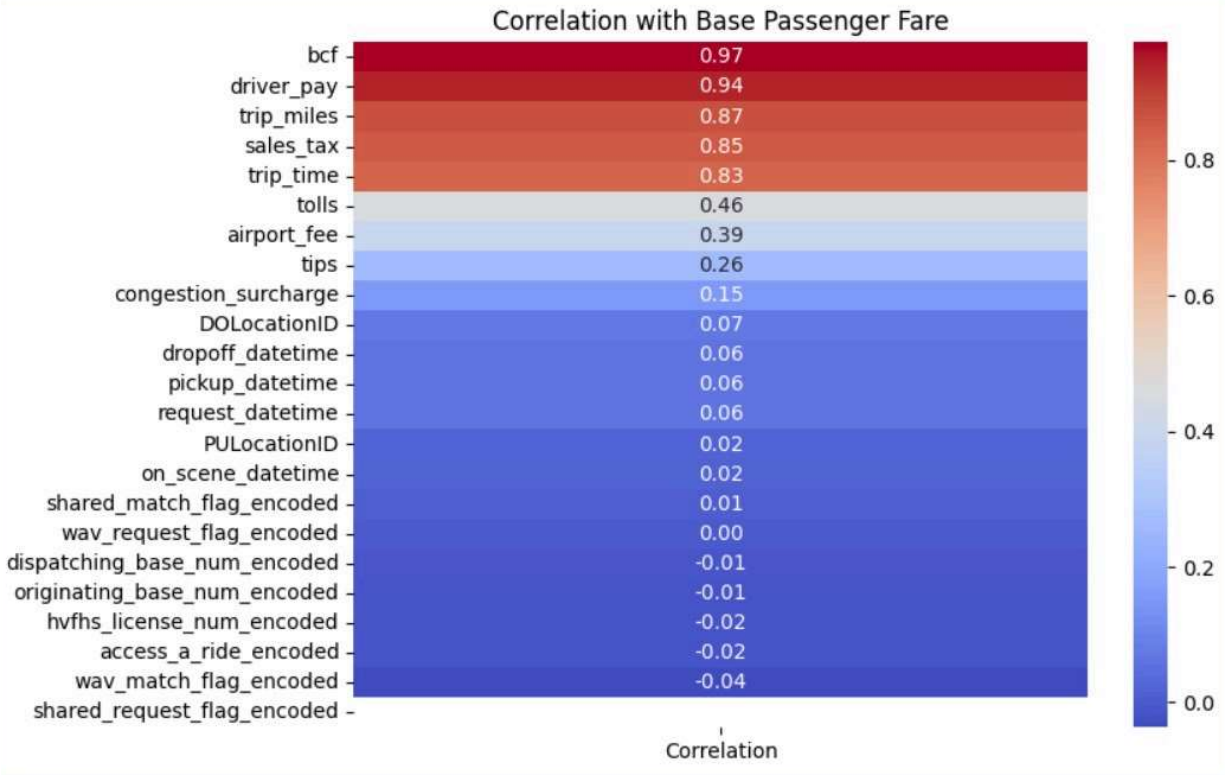
- Root Mean Squared Error (RMSE): 0.6840
- Mean Absolute Error (MAE): 0.3855

Compared to the initial model, the optimized model had better alignment between training and validation losses, with significantly lower variability. These RMSE and MAE values are also significantly lower than both the baseline models, which means we are capturing the nuances in data (RMSE is over 15 times less). The validation MAE remained stable, indicating improved generalization. While the RMSE and MAE highlight room for further refinement, the results confirm that regularization techniques and architectural adjustments effectively mitigated overfitting, leading to a more robust model.

Final Model

Upon our initial evaluation of the dataset, we selected the following features: *hvfhs_license_num*, *request_datetime*, *trip_miles*, and *trip_time*, focusing on *trip_miles* and *trip_time*. These were because of their relevance and frequency in similar studies using temporal features, while other

features were initially not chosen due to unfamiliarity and fear of over-complicating the model. To further refine our feature selection, we visualized feature correlations with the *base_passenger_fare* using a heatmap, as shown below:



This analysis highlighted strong relationships between the base passenger fare and several features, including *bcf*, *trip_miles*, *trip_time*, *tolls*, and other fees. Based on these insights, we included these features in the final dataset. However, features such as binary flags, datetimes, and locations exhibited weak correlations with the target variable and were excluded to simplify the model and reduce noise in the data.

Our refined neural network model performed significantly better, as shown in the training logs below:

```
Epoch 1/20
629409/629409 ━━━━━━━━━━━━━━━━━━━━ 3417s 5ms/step - loss: 0.0845 - mae: 0.2035 - val_loss: 0.9510 - val_mae: 0.0925
Epoch 2/20
629409/629409 ━━━━━━━━━━━━━━━━━━━━ 2300s 4ms/step - loss: 0.0699 - mae: 0.1917 - val_loss: 0.0218 - val_mae: 0.0854
Epoch 3/20
629409/629409 ━━━━━━━━━━━━━━━━━━━━ 2070s 3ms/step - loss: 0.0691 - mae: 0.1909 - val_loss: 0.0659 - val_mae: 0.0898
Epoch 4/20
629409/629409 ━━━━━━━━━━━━━━━━━━━━ 1567s 2ms/step - loss: 0.0692 - mae: 0.1906 - val_loss: 0.3866 - val_mae: 0.0945
Epoch 5/20
629409/629409 ━━━━━━━━━━━━━━━━━━━━ 1577s 3ms/step - loss: 0.0686 - mae: 0.1903 - val_loss: 0.1327 - val_mae: 0.0975
Epoch 6/20
629409/629409 ━━━━━━━━━━━━━━━━━━━━ 2450s 4ms/step - loss: 0.0686 - mae: 0.1903 - val_loss: 0.1126 - val_mae: 0.0995
Epoch 7/20
629409/629409 ━━━━━━━━━━━━━━━━━━━━ 2812s 4ms/step - loss: 0.0685 - mae: 0.1901 - val_loss: 0.4473 - val_mae: 0.0994
```

***We talk more about the final results on the test set in the Results section.**

The updated feature selection improved the model's predictive performance significantly, as demonstrated by the reduced RMSE and MAE. The inclusion of strongly correlated features like *trip_miles* and service charges helped the model better capture the underlying relationships in the data. The final model displayed stable training and validation losses, with validation MAE values consistently below 0.1, indicating good generalization. The decision to exclude weakly correlated features also contributed to a leaner, simple, and more efficient model.

These results validate the importance of correlated feature selection and regularization techniques in developing a robust neural network for regression tasks.

4. LITERATURE

Price prediction using machine learning models is a widely explored concept and has been successfully applied in various areas, including stock price prediction[2], housing price prediction[3], car price predictions[4], and agricultural products price predictions[5]. These papers have used different models ranging from linear to neural nets. Specifically, some models we came across were Structural Support Vector Machines (SSVMs), Regression, Clustering, Bayesian, Decision Tree, Random Forest, artificial neural networks, and ensembles of these models. According to Bayona-Oré et al.[5], neural networks work best for predictive tasks.

Many studies have analyzed rideshare datasets and some have specifically focused on fare predictions. Chen et al.[6], for instance, conducted a deep learning study on short-term demand prediction for ridesharing services, emphasizing the challenges of unevenly distributed pickup patterns using temporal and spatial features. They used the same dataset as our project but primarily focused on data analysis and trend identification before shifting to pick up prediction based on spatial features. In their model comparison, which included Support Vector Regression (SVR), Random Forest, and PROPHET (a statistical model), they achieved the lowest RMSE with a hybrid model that incorporated a neural network. While their UberNet approach leverages a rich set of features to analyze the entire dataset and predict multiple aspects, our work emphasizes temporal and spatial simplicity while still delivering accurate fare predictions. Similarly, Konishi et al.[7] also used the same dataset, conducting a comprehensive analysis of the data. While they focused on comparing driver pay and base passenger fares, examining factors like pay rules, company commissions, and temporal changes, our project uses the dataset to predict passenger fares directly, addressing a different facet. Moreover, their findings on the relationship between passenger fares and temporal trends highlight the importance of incorporating features like trip time and hour into our model.

Chao[8] examines the factors affecting Uber pricing across different locations and times in New York, using a regression predictor based on specific features from their dataset. Their analysis highlights that surge pricing is heavily influenced by rush hours, with prices peaking around 7–8

AM and 6–7 PM. Location also played a significant role in determining prices. However, they did not evaluate the overall performance of their model, reporting only the R-squared value. While R-squared indicates the proportion of variance explained by the model, it does not directly measure accuracy. However, they identified a strong correlation between the time of day (hour) and price coefficients, reinforcing the importance of including the hour as a key feature in our model, which we extracted manually from *request_datetime*. In a Medium post, Pundhir[9] examines different approaches for price prediction and evaluates them based on RMSE. They compared the baseline linear regression model with K-Nearest Neighbor, XGBoost, and neural networks. A key observation they made was that because the RMSE of neural nets was a lot less than that of linear regression, they can say that the data had some non-linearity and nuances that the linear model could not capture. This is something we also experienced while selecting a model. Even though the data we used was different, our RMSE for price prediction was 10 times less than their RMSE, which could be due to the fact that we trained on a larger dataset or that our feature extraction was better.

Most of the research papers we read and models we saw talked about using neural networks for fare prediction because the price of rideshare does non-linearly depending on the spatial and temporal features. Although not a lot of papers reported the evaluation of their whole model, our model gives a competitive performance, if not better, compared to the models that exist for fare prediction. Overall, our findings highlight that carefully extracted features and extensive preprocessing can lead to a high-performing model even without creating overly complex models.

5. RESULTS

Final Model Results

The evaluation metrics of the final model on the test data were:

- Root Mean Squared Error (RMSE): **0.3765**
- Mean Absolute Error (MAE): **0.1154**

Including the strongly correlated features, such as *trip_miles*, *trip_time*, and service charges allowed the model to effectively capture complex relationships in the data, leading to more reductions in both RMSE and MAE compared to the intermediate model. The final RMSE of 0.3765 is impressive, considering that fares range from 0 to 100. **The RMSE is 2 times less than the intermediate model and 30 times less than both the baseline models.** The low MAE values show the model's ability to make precise predictions, with only small deviations from the true fares. It highlights the effectiveness of our feature selection and regularization techniques throughout the model selection.

Model Comparison: Uber vs. Lyft

After analyzing fare predictions for specific license plates representing Uber (HV0003) and Lyft (HV0005), the following metrics were observed:
Uber (HV0003):

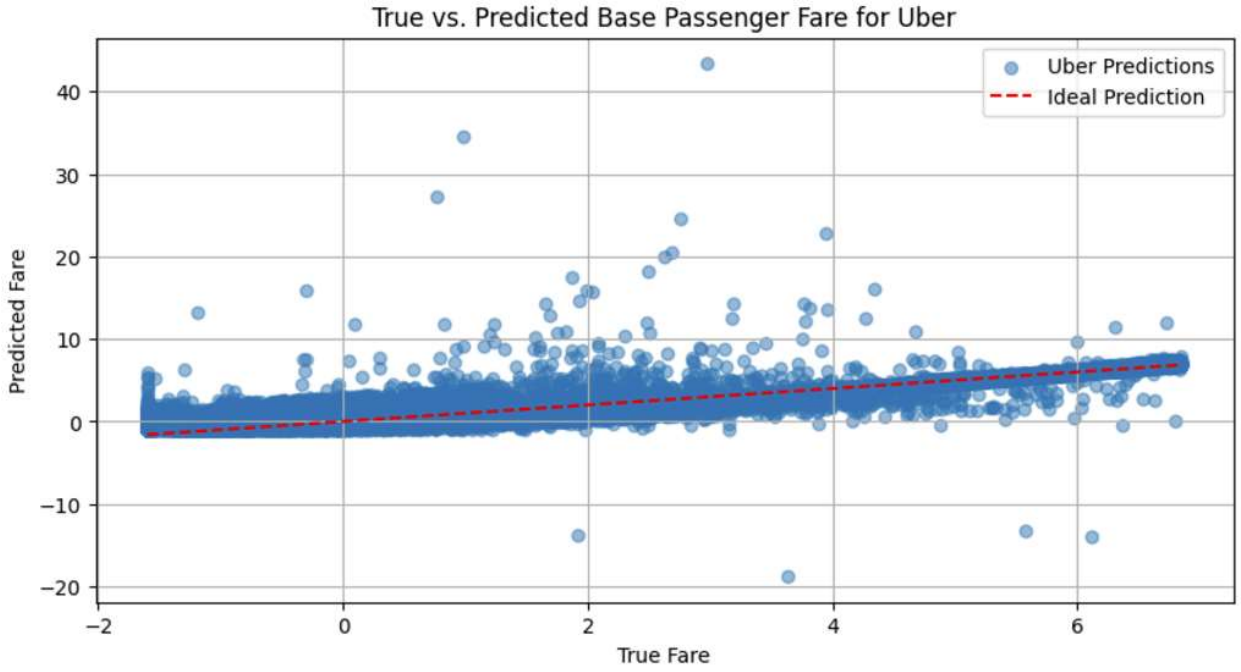
- RMSE: 0.16
- MAE: 0.09

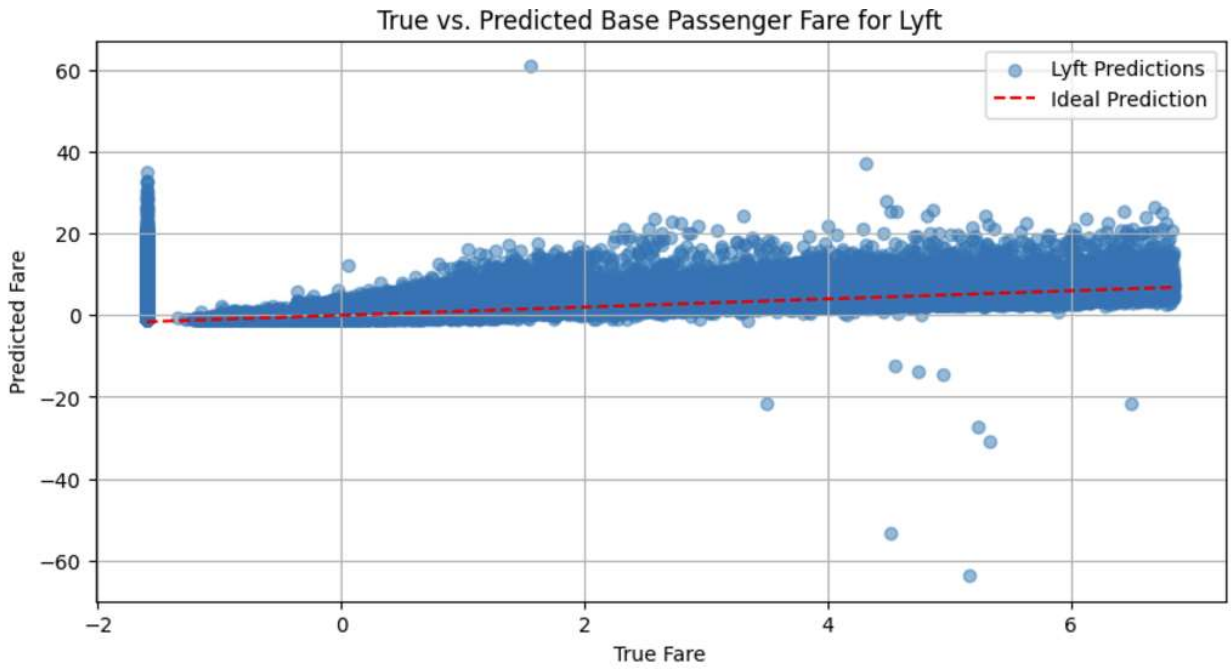
Lyft (HV0005):

- RMSE: 0.67
- MAE: 0.19

The results indicate that the model predicts fares for Uber trips more accurately than for Lyft trips. This discrepancy could stem from differences in data distributions or the features associated with the two services. For instance, Lyft's pricing structure or ride patterns might have greater variability or complexity, which the model finds harder to capture. Another factor could be because our dataset had fewer data points for Lyft rideshares as compared to Uber rideshares.

The scatter plots of true vs. predicted fares for both Uber and Lyft are shown below:

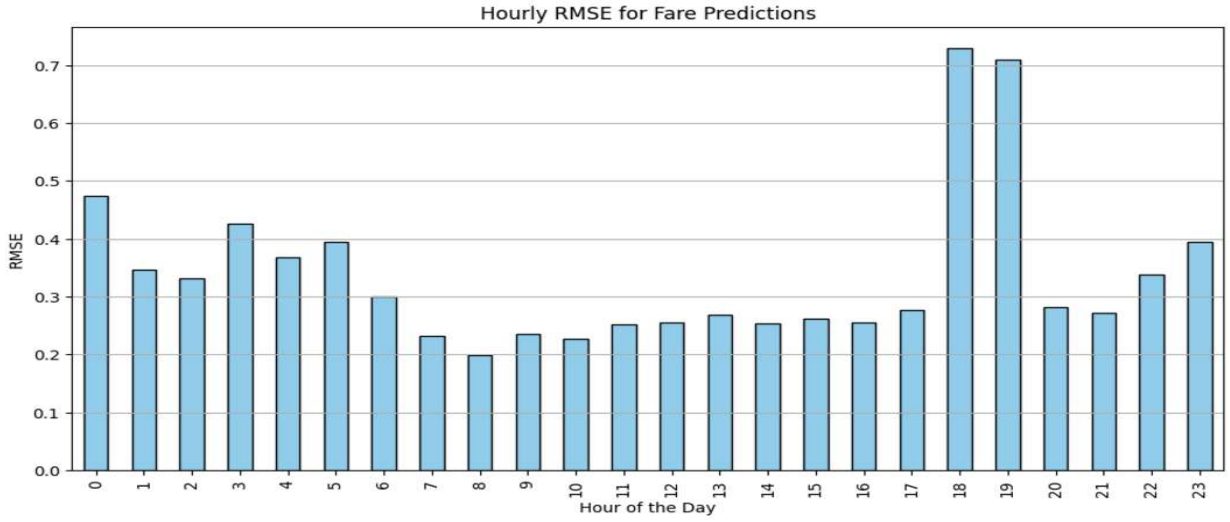




The scatter plots reveal that for Uber the points are closer to the ideal prediction line (y = x), suggesting strong predictive performance. For Lyft, there is greater dispersion around the ideal line, indicating less accurate predictions and potential model bias or underfitting.

The main motivation for analyzing the final model's performance separately for Uber and Lyft was to gain deeper insights into how effectively the model generalizes across different rideshare platforms. One of our baseline models was averaging the fares across Uber and Lyft and predicting the fare based on what service the rideshare belonged to. This made us interested in looking at how well our model is doing on separate services. Compared to the baseline model, our final model for Uber and Lyft specifically is doing significantly better.

Evaluate Accuracy by Hour

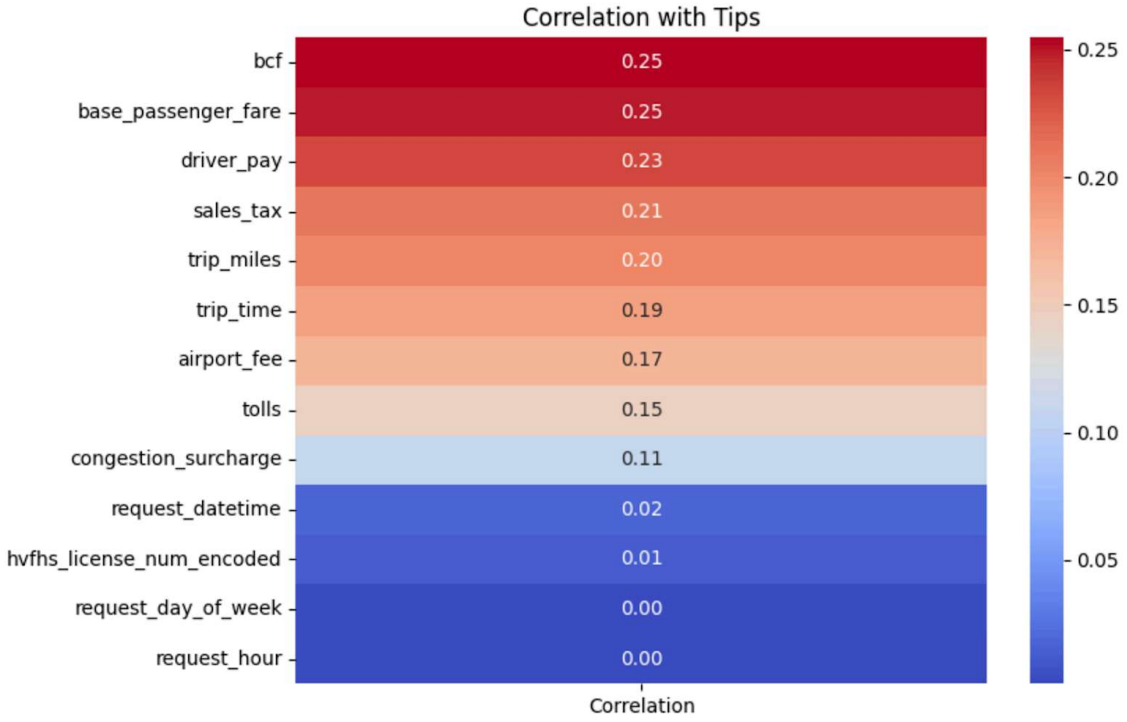


The peak RMSE of 0.7 is from 6-8pm. This indicates that the model struggles the most to predict fares accurately during this period. As observed in one of the research papers, this is peak rush hour. As a result, traffic patterns and variability in travel conditions could have introduced more complex relationships. It could also be that the companies use surge pricing to earn more. However, this is harder to capture in a simple neural network model. RMSE values are also comparatively higher from 12-5am, which could be a result of fewer data points. Moreover, it could also be due to price surging and dynamic pricing. Because there are fewer drivers during early hours, service providers tend to increase prices. All the variability could prevent the accuracy from going any lower. The RSME is the lowest from 8am-5pm. This suggests that fare prediction during regular commuting hours is more predictable because of consistent patterns in trips and fares.

The main motivation for this analysis was because we wanted to see how well our model was doing compared to the hour baseline model. The simple predictor based on hours was much worse compared to our final model. The reduced RMSE across all hours of the day, specifically during periods of high variability rush hour, shows how well our model is doing!

Tips Model

We were curious to know how well our model structure would perform while predicting tips. This is because giving tips is more of a personal choice which means that there are a lot less correlations between tips and any of the features as shown below:



We used the same model structure and adjusted our target feature to be *tips.* The training logs of our tips model can be seen below:

```
Epoch 1/20
629409/629409 ━━━━━━━━━━━━━━━━━━━━ 3112s 5ms/step - loss: 0.9329 - mae: 0.5706 - val_loss: 0.9226 - val_mae: 0.578
9
Epoch 2/20
629409/629409 ━━━━━━━━━━━━━━━━━━━━ 1629s 3ms/step - loss: 0.9243 - mae: 0.5666 - val_loss: 0.9392 - val_mae: 0.570
9
Epoch 3/20
629409/629409 ━━━━━━━━━━━━━━━━━━━━ 1610s 3ms/step - loss: 0.9212 - mae: 0.5665 - val_loss: 0.9258 - val_mae: 0.562
1
Epoch 4/20
629409/629409 ━━━━━━━━━━━━━━━━━━━━ 1825s 3ms/step - loss: 0.9214 - mae: 0.5664 - val_loss: 1.2313 - val_mae: 0.581
8
Epoch 5/20
629409/629409 ━━━━━━━━━━━━━━━━━━━━ 1641s 3ms/step - loss: 0.9224 - mae: 0.5664 - val_loss: 1.2400 - val_mae: 0.562
5
Epoch 6/20
629409/629409 ━━━━━━━━━━━━━━━━━━━━ 1979s 3ms/step - loss: 0.9222 - mae: 0.5667 - val_loss: 1.3493 - val_mae: 0.567
6
```

The evaluation metrics of the final model on the test data while predicting tips were:

- RMSE: **1.0541**
- MAE: **0.6225**

Surprisingly, our model still performed well while predicting tips considering the tips ranged from 0-200 dollars. The model was still able to capture meaningful patterns in the tipping data, even with limited feature correlations. The performance shows that while tipping behavior may appear random, there are still underlying factors like trip duration, fare amount, or time of day that influence tipping patterns. Our neural network is picking up on these trends, even if the relationships between tips and the features are weaker. This confirms the robustness of our model structure and its ability to adapt to different regression price prediction tasks, even for targets with lower predictability.

6. REFERENCES

1. D2L.ai. "An Online Advertising Dataset." In *Chapter on Recommender Systems: Click-Through Rate Prediction*. Retrieved from https://d2l.ai/chapter_recommender-systems/ctr.html#an-online-advertising-dataset.
2. Carson Kai-Sang Leung, Richard Kyle MacKinnon, and Yang Wang. 2014. A machine learning approach for stock price prediction. In Proceedings of the 18th International Database Engineering & Applications Symposium (IDEAS '14). Association for Computing Machinery, New York, NY, USA, 274–277. https://doi.org/10.1145/2628194.2628211.
3. Park, B.; Bae, J. K. Using machine learning algorithms for housing price prediction: The case of Fairfax County, Virginia housing data. Expert Systems With Applications 2014, 42 (6), 2928–2934. https://doi.org/10.1016/j.eswa.2014.11.040.
4. Gegic, E.; Isakovic, B.; Keco, D.; Masetic, Z.; Kevric, J. Car Price Prediction using Machine Learning Techniques. *TEM Journal* **2019**, 113–118. https://doi.org/10.18421/tem81-16.

5. Bayona-Oré, S.; Cerna, R.; Hinojoza, E. T. Machine learning for price prediction for agricultural products. WSEAS TRANSACTIONS ON BUSINESS AND ECONOMICS 2021, 18, 969–977. https://doi.org/10.37394/23207.2021.18.92.
6. Chen, L.; Thakuriah, P.; Ampountolas, K. Short-Term Prediction of demand for Ride-Hailing Services: A deep learning approach. *Journal of Big Data Analytics in Transportation* **2021**, *3* (2), 175–195. https://doi.org/10.1007/s42421-021-00041-4.
7. Konishi, A., Ramakrishnan, V., Waheed, S., & Herrera, L. (2023). Analysis of High Volume For-Hire Vehicle Data for New York City: Selected Months, 2019–2022. UCLA: Institute for Research on Labor and Employment. https://escholarship.org/uc/item/5q09h352.
8. Chao, J. 2019. Modeling and Analysis of Uber's Rider Pricing. In Proceedings of the 2019 International Conference on Economic Management and Cultural Industry (ICEMCI 2019), January 2019. Retrieved from https://doi.org/10.2991/aebmr.k.191217.127
9. Pundhir, R.K. 2021. Uber fare and demand prediction, data analysis. Medium.Retrieved from https://medium.com/@rishabh21071/uber-fare-and-demand-prediction-data-analysis-fc26201b03f.
10. New York City Taxi and Limousine Commission (TLC). TLC Trip Record Data. Accessed December 3, 2024. Available at: https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page.
11. Bonella, A. NY Trip Data and Price Model. Kaggle. Accessed December 3, 2024. Available at: https://www.kaggle.com/code/alexbonella/ny-trip-datexland-price-model.