Abhiram Tadepalli & Anveetha Suresh
& AXS220399
CS 4375
Prof. Sriraam Natarajan
5/13/2025

UTD Parking Prediction

Introduction

At UTD, we've noticed the challenge of finding parking in on-campus garages. To address this, we've developed models to predict garage occupancy based on factors like permit color, weather, time, day of the week, and traffic. We tested linear regression, neural networks, decision trees, naïve Bayes, and an ensemble method.

Data Preprocessing and Features

We collected data from UTD's "Available Spaces" site, which shows open parking spots by garage and permit color. We combined this with weather data from openweathermap.org and traffic data from Google Maps API, updating every five minutes. Each record includes features like date, time, day of week, garage, permit color, open spots, travel times, and weather conditions. We encode categorical features: ordinal encoding for color and conditions, and one-hot encoding for garage. We added a label feature, set to true if fewer than 15% of spots were available, false otherwise. Time was converted to minutes after midnight, and date to month only.

Model Choices and Methods

We tested several models to predict parking garage fullness, each chosen for its unique strengths.

Linear Regression was used as a baseline to model the relationship between features and available spots. Due to the complexity of parking behavior, we expected limited performance.

Decision Tree models split data based on feature importance, making them well-suited for classification tasks like ours. We expected good results from this method.

Neural Networks were chosen for their ability to capture complex patterns across many features. With proper tuning, we anticipated high accuracy in predicting availability.

Naïve Bayes predicts based on feature likelihoods, assuming independence. Since our features are often correlated, we expected this model to perform poorly.

Linear Regression

Results

For the linear regression, we had to convert all the variables to numerical values, including dates and times. With this adjustment, these were our results for training and test error:

```
Iterations=1000, LR=0.003300, Train Error=16045.51, Test Error=16042.86
```

It is quite obvious that this is not an ideal output. The MSE for both train and test are incredibly high and the linear regression is predicting nearly 100 spots difference for each example. Not only is it predicting 100 spots extra, it is also predicting approximately 200 spots below the expected number for some examples, and altering the iterations does not seem to make much of a difference.

```
Iterations=1000, LR=0.003300, Train Error=16492.49, Test Error=16488.18
Iterations=10000, LR=0.330000, Train Error=16043.18, Test Error=16040.83
```

We also tried to remove variables that may have potential for interaction, such as humidity, temperature, and visibility, as these variables will have significant correlation to the ‘conditions’ feature. The model still performed similarly.

```
Iterations=10000, LR=0.500000, Train Error=16043.18, Test Error=16040.83
```

Below are some specific examples of the model’s predictions:

```
Example | True spots_open | Predicted spots_open
      1 |           34.00 |              229.44
      2 |           48.00 |               89.95
      3 |           38.00 |              122.45
      4 |          424.00 |              259.89
      5 |           46.00 |              104.91
```

Because learning rate, iterations, and variable removal did not make a significant difference in the model’s performance, the poor performance could be partially due to the different, non-linear patterns in the features. Features like month, day, and time can have cyclic effects on the data. This could be adjusted by either adjusting the features themselves, or changing the model to a new one entirely. The model’s incredibly poor performance can be attributed most likely to the different inter and intra-feature relationships and patterns.

Neural Network

Results

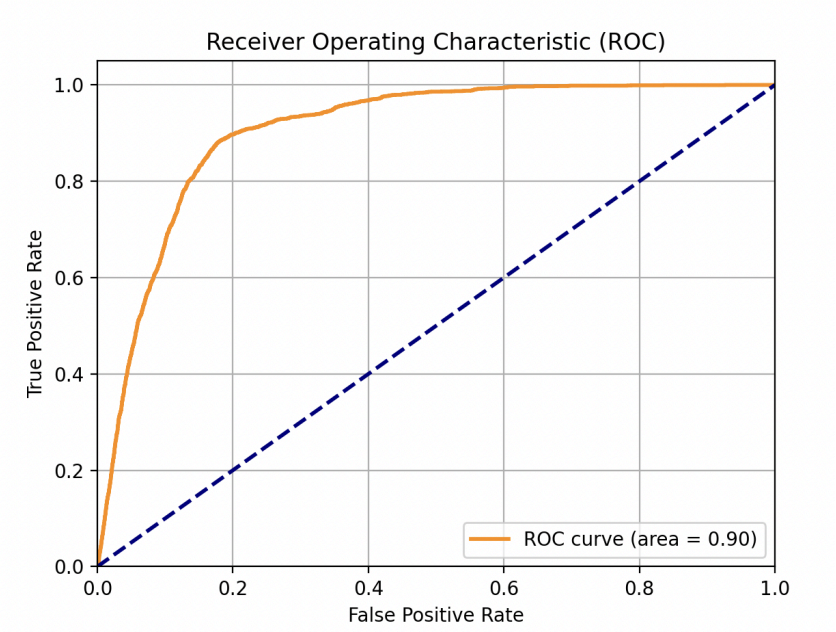
```
Neural Network Accuracy: 0.8986

Confusion Matrix:
Predicted   0     1
Actual  0  35208  3034
        1   1123  1611

Classification Report:
```

```
  Class   Precision      Recall       F1     Support
      0      0.97         0.92         0.94        38242
      1      0.35         0.59         0.44        2734

TPR (Sensitivity) at threshold 0.5: 0.5892
FPR at threshold 0.5: 0.0793
ROC AUC: 0.9048
```



For neural networks, we decided to alter the prediction threshold, which greatly impacted the NN's performance. The ROC curve and the AUC of the Neural Network is significantly better than all of the other models we have tested. Altering the prediction threshold and changing the class weights (5 times for class one) has changed the performance of the Neural Network to compensate for the imbalanced/skewed data. Neural Networks performed significantly better than our other models, primarily due to its flexibility and ability to work for almost any model. This has worked greatly in our favor, as the hidden units and layers are creating a significantly better performing model than the others.

Decision Tree

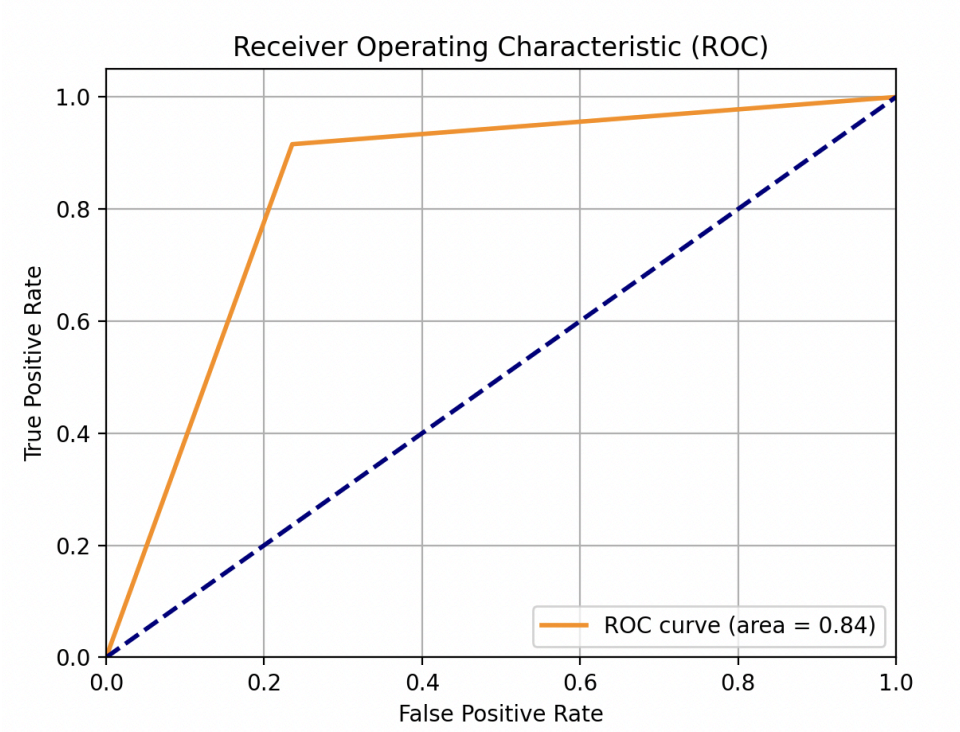
Results

```
For Max-depth 8:
Decision Tree Accuracy: 0.7762

Confusion Matrix:
Predicted   0     1
Actual  0  29239  9006
        1   164  2567
```

```
Classification Report:
  Class   Precision      Recall      F1    Support
      0      0.99        0.76        0.86        38245
      1      0.22        0.94        0.36        2731

TPR (Sensitivity) at threshold 0.5: 0.9399
FPR at threshold 0.5: 0.2355
ROC AUC: 0.8522
```



Naive Bayes

Preprocessing

Because Naive Bayes requires discretization of variables, we converted time and traffic variables to be more descriptive and categorical. We also noticed that breaking the bins down into more fine ranges creates for a slightly better performance of the model. We also noticed that as we trained the model, Naive Bayes had a tendency to predict false negatives. We found that this could also be due to the fact that there are less positive (full) examples in our data set. To combat this, we also utilized SMOTE via imblearn. This allowed us to train more positive samples to avoid training imbalance in the data.

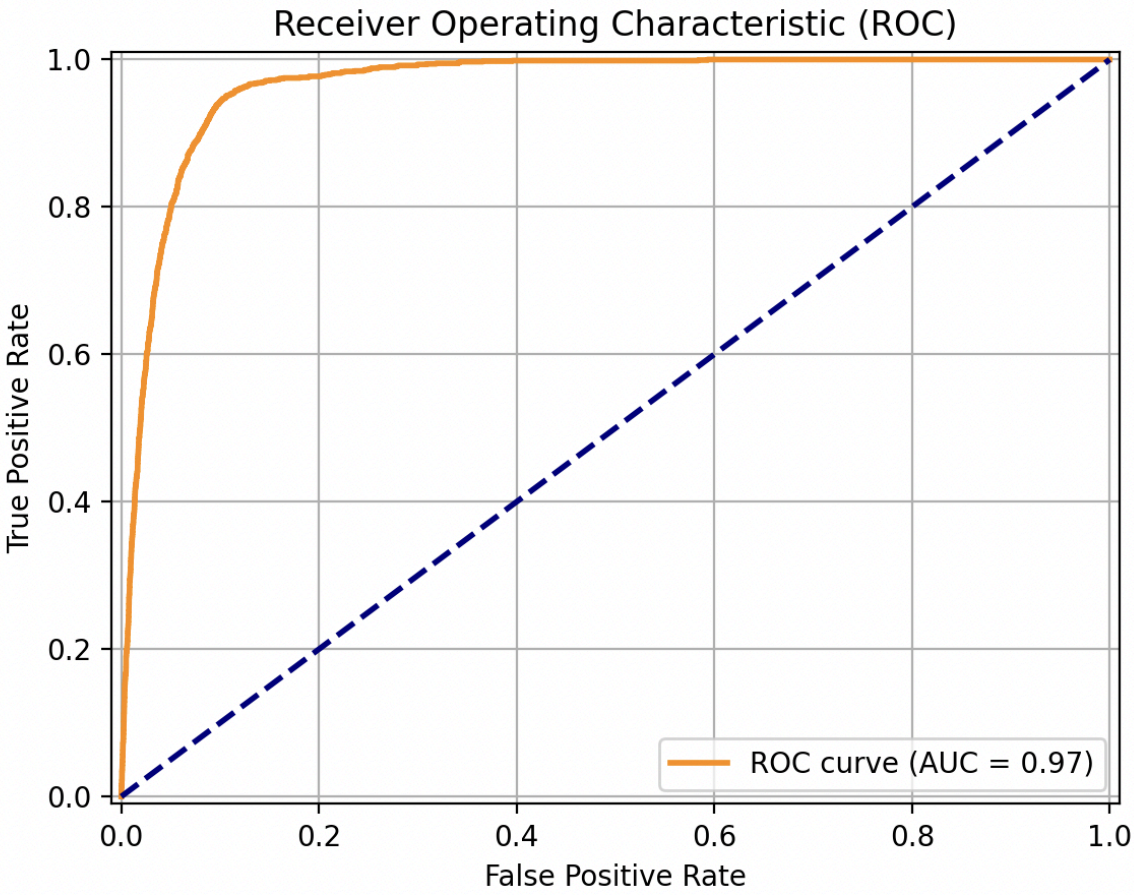
Results

```
Accuracy: 89.48%

Confusion Matrix:
```

```
Predicted      0     1
Actual
0          18167  2223
1             66  1294

Classification Report:
   Class  Precision     Recall         F1    Support
       0       1.00       0.89       0.94      20390
       1       0.37       0.95       0.53       1360
```



The ROC curve for Naive Bayes is also demonstrating good results for the model. While the accuracy could be higher, the reason it is that way is because of the difficulty in predicting the positive examples. The imbalance in our dataset causes suboptimal training. But given this information, we can say that Naive Bayes is one of the best models out of our chosen ones. This may be due to the fact that Naive Bayes compares test examples with other similar, historical examples and predicts a label based on that.

Ensemble Method

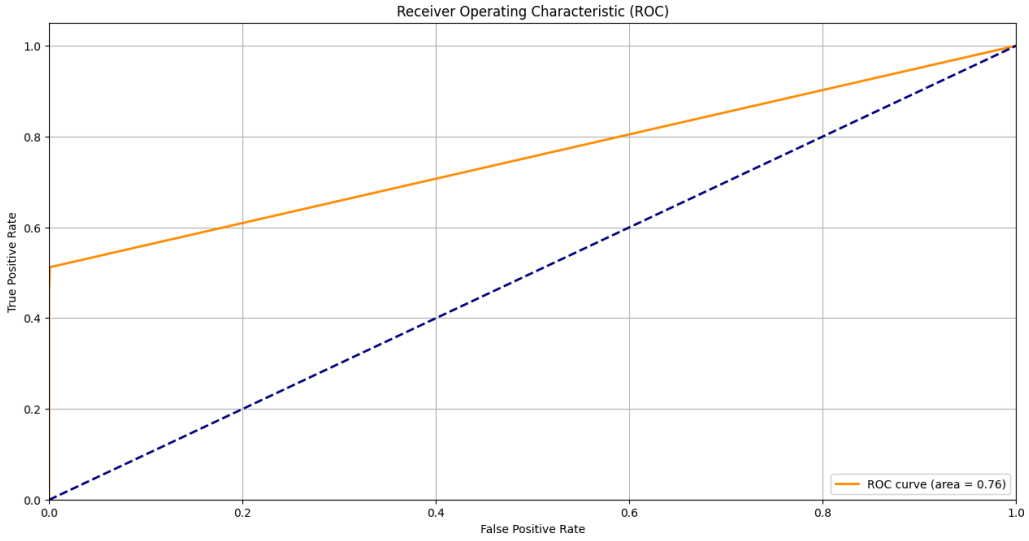
Results

```
Bagging Accuracy: 0.9622
Confusion Matrix:
```

```
Predicted    0     1
Actual  0   3118     0
        1    106    56

Classification Report:
  Class  Precision     Recall         F1    Support
      0       0.97       1.00       0.98       3118
      1       1.00       0.35       0.51        162

TPR (Sensitivity) at threshold 0.5: 0.3951
FPR at threshold 0.5: 0.0000
ROC AUC: 0.6970
```



While the area under the curve in this ROC chart is high, it is still a linear curve. This tells us that the model is predicting in a somewhat random manner, but it is still somewhat able to differentiate between positive and negative examples. The poor performance of the model, and its high number of false negatives could be due to either errors in our code, or overfitting to the data, although bagging is not highly prone to overfitting.

We also used bagging over boosting to minimize variance, and we applied it to decision trees, as they performed relatively the best. In theory, bagging would have improved the performance of the decision tree. In the future, we would like to combine different models into one ensemble method as well.

Conclusions & Overall Performance

Throughout our project, we noticed several issues during the data engineering and preprocessing stages that may have influenced our models' performance. One of the biggest

challenges was the significant class imbalance in our dataset, with a much higher number of positive examples than negative ones. This imbalance likely skewed the learning process and contributed to the less-than-ideal results we saw across all models.

Despite this, neural networks and Naive Bayes performed better than decision trees, likely because they handle imbalanced data well and can capture complex patterns in the data. Given more time for tuning, neural networks could have performed even better. We would have liked to explore different neural network architectures in more depth (experimenting with the number of hidden layers, layer sizes, and various activation functions) to understand how these changes affect performance and generalization. It also would have been valuable to analyze our data and models further to identify which features and modeling choices had the most impact on results.

We also found it surprising that our bagging ensemble method underperformed compared to expectations. Since bagging usually helps boost the performance of high-variance models like decision trees, this result was unexpected. We suspect this might be due to issues in our implementation or not enough tuning of the base estimators and ensemble parameters. With more time, we would want to investigate this further to gain more insight into both the model behavior and our ensemble setup.

Overall, a key takeaway from this project is how much different training strategies can affect model performance, even when using the same data. Approaches like cross-validation, bagging, and boosting really highlight how the way we train our models can make a significant difference. With additional time and resources, we would like to explore these methods more deeply to see how they impact our results.