CS 188 NEXT Project Report

Marcus Pearce, Ryan Lam, Michelle Zhuang, Ashwin Ranade

Executive Summary:

The goal of this final project is to develop a predictive model to identify high-performing drivers for NEXT Trucking, one of California's fastest FreightTech companies. In the process, we apply a variety of the data science tools learned in this course, including concepts such as Neural Networks, PCA, linear regression, as well as practice with the practical application of these ideas through data visualization, model training, and result evaluation.

We first identified a number of issues with the dataset we needed to address. Most importantly, we wanted to identify the most relevant predictors for our classification model, and decide what augmented features we could create to further capture relevant information. We also had to deal with a highly imbalanced dataset in terms of labels, as well as a high level of collinearity between some of the predictors. Our approach to addressing the above issues is explained in more detail in this report.

At a high level, we found that balancing the dataset by the labels drastically improved all of our models. After experimentally preprocessing the data in our pipeline, we found the set of augmented features poweronly, dryvan, reefer, boxtruck, flatbed, truck_unknown, weekday, good_port_person, and specified_preferred_lanes helped improve the performance of our models, leading us to conclude that artificial predictors helped capture relevant information beyond that directly from the given predictors.

In terms of finding the best model, we found a neural network gave us the best results. With a high F1 score peaking around 67.5%, we are confident our model will be a valuable tool in evaluating drivers for NEXT Trucking.

Background/Introduction:

Our goal is to predict high performing drivers using the given dataset and predictors. We define high performing drivers as those in the 75th percentile of ‘loads’ and the 75th percentile of ‘most_recent_load_date’, indicating that they are among the top of their peers in the number of loads they have serviced and how recently they have completed a job.

This is a challenging task, since it’s not immediately obvious what features correlate to high performing drivers. Much of the data is also incomplete, or includes irrelevant features (such as dim_carrier_company_name). Additionally, many features are related to one another, causing collinearity. For example, ts_first_approved, ts_signup, and days_signup_to_approval are all related to one another, and including all of these in our final model would lead to negative side effects.

Further, the data requires a lot of preprocessing to prepare it for modeling. Many of the predictors are categorical, which requires strategies like One Hot Encoding so it can be analyzed by models. There is also the issue of missing data which requires imputation strategies such as median imputation. Unit or scaling issues can be addressed using StandardScaler, to give the features unit variance so no predictor “overwhelms” the others in a given model. Another large issue is the class imbalance in the data, with the majority of drivers not falling into the ‘high-performing driver’ category.

The largest problem is identifying which features are relevant to this study, as well as figuring out how to find relevant relationships in the data. We address this by augmenting new features to try to capture information not directly expressed by the original predictors. We also want to drop irrelevant features to reduce the number of predictors in the model and combat overfitting.

From our industry research, we found that there is little existing research in identifying relevant predictors to driver performance in the context of this study. While there are other existing studies trying to measure driver safety or efficiency, our metrics of ‘high-performing’ mean maximizing the number of loads completed as well as how recently a driver completed a job. Thus, in this context, a driver who does lots of very short trips every day would be perceived as much higher-performing than one who does cross-country trips weekly (even though the latter may take more ‘skill’).

Methodology:

Generation of Labels

We first collapsed the duplicate driver rows, grouping the tuples by id_driver and then keeping only the most recent entry for each driver. We then loop through the resulting data entries and generate a label of 1 if the driver is in the 75th percentile of the 'total_loads' and 'most_recent_load_date' features. Otherwise, we generate a label of 0. We append these labels to the dataset.

Data Processing Pipeline

To address the issue of imbalanced labels, we decided to oversample the minority class (high-performing drivers, or label=1) by using SMOTE, or Synthetic Minority Oversampling Technique. With this strategy, the synthetic examples are generated using a strategy similar to k-NN with the existing data. In this way, we get an equal number of datapoints with high-performing drivers and non-high-performing drivers. We also experimented with using a decision tree based classifier (Random Forests), which is known to deal with imbalanced datasets well.

Our data strategy seeks to prepare the data to best capture the relevant information for predicting high-performing drivers, as well as addressing issues like categorical variables, missing values, and scaling issues.

We first dropped the predictors dim_carrier_company_name, as this is duplicate information to the feature id_carrier_number, and we identified that the name itself holds no relevance in our predictions. We drop the features ts_signup and ts_first_approved as the period between signup and approval is already stored in the days_signup_to_approval feature, and the dates themselves hold little relevance.

For numeric variables, we decided to use median imputation for num_trucks and days_signup_to_approval to avoid outliers. We also noted that total_loads is split into marketplace_loads and brokerage_loads. We decided that the relative proportions of marketplace_loads and brokerage_loads could have a say on a driver's performance pattern, so we added a marketplace_vs_brokerage feature to capture this information.

Lastly, we scaled the numerical features using Standard Scaler to center the mean and scale to unit variance. This prevents some predictors dominating the model due to scale.

For non-numeric variables, we further split them into binary variables and categorical variables.

For the binary variables, besides variables that only had two unique values (such as port_qualified, signup_source, etc.), we also included weekday and dim_preferred_lanes. Despite there being a natural numeric ordering for weekdays (Monday - 1, Tuesday - 2, etc.), it didn't make sense to encode relationships such as Tuesday > Monday, so we directly transformed the weekday feature to have binary workday vs. weekend values. The other one was dim_preferred_lanes. Few drivers specified the exact lanes that they preferred, and the majority of drivers simply didn't fill out the field. As we didn't want each specific lane to be a feature and blow up the feature space, we transformed this feature to be a binary variable between drivers that specified a lane and drivers that didn't. We also added an augmented feature of a good_port_person if a driver is True for all the following fields: interested_in_drayage, port_qualified, and driver_with_twic. Finally, we used an Ordinal Encoder to transform any strings into binary numbers 0 and 1.

Additionally, we note that the feature carrier_trucks had values in a list format, which were not sorted in any particular order. As a result, we treated each possible value in the list as a binary feature to better extract the feature information from carrier_trucks.

The categorical features are home_base_city, home_base_state, num_trucks, and days_signup_to_approval. To impute the NaN values in home_base_city and home_base_state, we decided that it may be misleading to take the mode (most frequent category value), so we decided to treat a missing value as its own category (as not including this information may be correlated with a driver's performance). Finally, we did One Hot Encoding on all the categories.

Experimentation on Various Models

While experimenting with different models to find a high-performing one, we tried a Random Forest Classifier. We tuned the parameters of the number of estimators, maximum tree depth, and maximum proportion of features to choose from at each split. We found that we achieved an accuracy and F1 score of about 95% on the training data. We also tried an AdaBoost classifier, similarly tuning the parameters of number of estimators and maximum tree depth. We achieved an accuracy score and F1 score of around 97% on the training data. Overall, we found that the neural network we used in Part 9 of the assignment was our best classifier, so we used that model to generate our predictions on the score dataset.

Results:

In this section, we'll discuss some basic statistics on our variables (step 3), the results of a neural network classifier on our data set (step 8), and the results of k-fold cross validation on our results (step 9).

In terms of basic statistics, we have 29 features we can use for prediction. Most of the features were generally complete, although some features had many missing values, including ts_first_approved, days_signup_to_approval, and dim_preferred_lanes. Additionally, most of our features are categorical in nature.

Most of the distributions for our numerical variables are extremely left skewed, such as days_signup_to_approval and brokerage_loads. This indicates that most truck drivers' features are concentrated in lower numerical values, while there are a few outliers. Additionally, many features are related to one another (collinear), such as brokerage_loads and brokerage_loads_atlas. Some categorical features are mostly one category as well, such as California in 'home_base_state' (expected since NEXT Trucking is headquartered in California).

Our neural network employed a ReLU activation function, along with Adam as an optimizer, and 4 hidden layers, with 100, 100, 100, and 50 nodes respectively. We ran the neural network for a max of 200 iterations, and achieved a resulting f1 score of ~97% and a resulting accuracy score of ~95%. We tuned hyperparameters by changing the number of nodes in each hidden layer, and by changing the max iterations; the hyperparameters above are those that gave us the best results.

In order to get a better feel for our results, we used K-Fold Cross Validation with 10 folds, on both our SVM ensemble model, and our neural network. Our results on the training data were very promising — the F1 score of our neural network was around 97%, while the SVM ensemble was around 94%. This shows that our models are robust to different train-test splits, and not getting lucky on a particular instance of the training data.

Discussion:

In the context of this study, our high performing models indicate that the set of predictors given in this dataset, and by extension the information they contain, are sufficient to accurately predict high-performing drivers. In particular, we find

In terms of leveraging our findings, NEXT can use our model to improve the quality of drivers they select to drive their fleet. Additionally, NEXT can perform further analytics to select optimal drivers in their fleet, such as those least likely to cause an accident, the most efficient drivers in terms of driving time, etc.

Conclusion:

In this project, we've created a model for predicting high-performing drivers, that maximizes the F1 score, the harmonic mean of precision and recall. We built our model by first preprocessing our data, then running various machine learning algorithms to find the best architecture for our data set; we found that a neural network provided the highest metrics. Finally, we tuned hyperparameters to further improve our F1 score, climaxing at 67.5% F1 score on our test data set. Data science is a powerful tool in any field, and we hope NEXT Trucking continues to utilize machine learning for prediction in the future.