# CS 188 Final Project Report: NEXT Trucking

Drew Letvin and Jack Brewer

## Executive Summary

Understanding and mastering an industry is a key for company success. When it comes to freight and shipping, NEXT trucking must be able to efficiently match their driver base to customers based on flexible driver schedules. Knowing which drivers are high-performing helps NEXT understand driver capability and recruit the best new class of drivers. To help NEXT better understand their drivers, we investigated their driver dataset, found correlations between driver features, and built many predictive models to help classify drivers as high-performing.

To start, we investigated the dataset to learn more about the features. We classified high-performing drivers as those who were in the 75th percentile of total loads and most recent load date. We then went through each feature in the dataset to purge unneeded features and determine effective imputation strategies based on the data in said feature. After this, all features have been converted into numerical features and there were no null values left. We could then learn more about the correlations between the features, run our pipeline on the data, and start to build models. In addition to simple linear regression, we ran PCA on our remaining features to reduce some of the dimensionality. We then built predictive models following techniques such as ensemble methods, neural networks, and others.

After k-fold cross validation on each of our models, we found that the MLP classifier performed the best with an accuracy score of about 97.5%. Using this model, we can run new test data on our model to predict whether or not a driver is high-performing given their session. This will allow NEXT to understand which drivers are currently completing more loads so that dispatchers can better schedule drivers and recruiters can hire the best.

## Background/Introduction

In order to run the most effective business, you must recruit the most high-performing employees and understand how their contributions fit into the larger business goals. To do this, companies must be able to determine what high-performing employees look like and the amount of work they are capable of doing. NEXT Trucking is no different, as they must be able to determine the amount of shipments they can handle with their current driver base and expand their driver network to be able to accommodate for future growth.

NEXT is revolutionizing FreightTech by creating smart first-to-last mile solutions. They allow for drivers to work whenever they want to, so distinguishing drivers who are high performing will help NEXT dispatchers to understand those drivers who can take on the most load. In addition, the most high-performing drivers can be put on longer/more important missions where time may be critical. Predicting high-performing drivers can also help NEXT understand what makes a high-performing driver recruit similar driver types, allowing for faster growth.

NEXT's predictive model must be able to take a driver profile and make a prediction on whether the driver is considered high-performing. Drivers who are high-performing must be in both the 75th percentile of total loads completed and most recent load date. This allows for NEXT to determine the drivers who have recently completed a load and who have in general completed many loads in the past. Given a driver trip with an unknown number of total loads or how recently a driver has completed a load, the model must predict whether or not the driver is high-performing. This allows for NEXT to learn what makes drivers high-performing and help them recruit better drivers.

## Methodology

**Label Generation:**

The first step in our process was to generate labels for our training data so that we could properly validate our model accuracy after training. As specified in the project specifications, drivers in the 75th percentile of ‘total_loads’ and the 75th percentile of ‘most_recent_load_date’ were to be assigned a 1 while all others were to be assigned a 0. To achieve this result we first needed to collapse the rows of data such that there was only one per driver. In the form the data was given to us each row represented a unique encounter meaning drivers often had multiple rows assigned to them. Thus, we sorted the data by ‘load_day’ so that the most recent load for each driver would be encountered first. Then we dropped any duplicate driver rows after the first time a driver id was encountered.

The resulting dataframe contained a single row for each driver with the ‘load_date’ being their most recent recorded load date and their ‘total_loads’ being the correct cumulative loads for that driver. Using the quantile method we found the 75th percentile of ‘total_loads’ to be 17 loads. We then took the first 25 percent of rows and checked each driver’s ‘total_loads’ against our value for the 75th percentile. If they were equal to or above this number then they were assigned a label of 1, otherwise they were assigned a 0. This of course created an unbalanced data set with significantly more 0 labels than 1. To remedy this we then randomly dropped a number of the rows with a 0 label such that there was a 4 to 1 ratio of 1 labels to 0 labels.

**Data Strategy:**

Our data was first altered before step 3b, outputting basic statistics, of the project so that we could create a correlation matrix and better determine the significance of each variable as a predictor of the row’s label. The first issue that needed to be handled was that of non-numerical data through encoding. We used a variety of approaches based on the nature of the given feature. For the columns of ‘dim_carrier_company_name’, ‘home_base_city’, ‘id_carrier_number’, and ‘carrier_trucks’ we simply dropped the entire column as they all had many unique objects and would have required a high level of dimensionality to properly encode with one-hot-encoding. The columns ‘port_qualified’, ‘driver_with_twic’, ‘signup_source’, ‘interested_in_drayage’, and ‘dim_carrier_type’ were all binary and as such were able to be easily encoded with label encoding. ‘Weekday’ was not a binary value but as the days of the week are ordered we felt it was okay to encode with label encoding here as well. We created a custom encoding scheme for ‘home_base_state’ that created a binary encoding where 1 was CA and 0 was any other state. Lastly we encoded any dates as a number representing the number of seconds since the POSIX epoch, avoiding a high dimensional encoding scheme.

The second problem we faced was that of null values. To handle this we again utilized a variety of strategies. We dropped the columns of ‘ts_first_approved’, ‘days_signup_to_approval’, and ‘dim_preferred_lanes’ as they had a high percentage of null values. Some columns with null values were already dropped above due to encoding issues so this took care of them. We handled the null values in num_trucks using KNN imputation with k equal to 2 such that null values were replaced with an average of the k nearest neighbor rows. Lastly, the encoding scheme used on ‘home_base_city’ dealt with the null values by encoding them to 0, or not CA.

Once these preliminary steps had been taken we were able to generate a correlation matrix and see how correlated each feature was with the row’s labels. Upon analysis of this we found that the features of ‘signup_source’ and ‘weekday’ were indeed not highly correlated to the label. We had suspected this could be a possibility and after confirming this here we dropped the two columns in step 4a. Despite low correlations in ‘dim_carrier_type’ and ‘home_base_state’ we suspected that these could actually be useful despite this just based on intuition and thus did not immediately drop them so that we could look into it more later on. After running a basic linear regression in step 5, we were able to confirm that these were not significant predictors based on the fact that their p-values were above 0.05. We then subsequently dropped the two columns and repiplined our data.

We chose to keep the remaining 16 variables, one of those being our augmented feature of driver longevity created by crossing dt and first_load_date, because they were deemed to be significant predictors of the label through both the correlation table and linear regression for feature importance. It is important to note that we did not include the variables of ‘id_driver’, ‘year’, ‘first_load_date’, or ‘brokerage_loads_otr’ in the linear regression to avoid as much collinearity as possible. Even with these precautions there was a warning of some colinearity due to a low eigenvalue. When I say that we kept all these predictors it is not completely accurate as we did perform PCA to reduce the number of dimensions from 16 to 7. This was around when the variance captured by each new vector descended below 5%. With this strategy we were able to keep about 87.26% of the total variance.

**Modeling and Techniques:**

We included multiple ensemble methods in our perspective set. These included an AdaBoost model, a RandomForest model, and a GradientBoostingClassifier. We went about determining the parameters of each of these models in the same way. First we created a grid of the possible parameters with a range of plausible values filling out each row. Plausible values were determined using graphs showing the relationship of accuracy with that given variable as can be seen in section 7 of our project. This grid was then exhaustively tested using RandomizedSearchCV with a kfold cross validation of 5 folds being used to determine the parameters with the highest accuracy.

We also implemented two neural networks, one in section 8 of our project and the other (MLP) in section 10. Before these we first attempted to create a pytorch model. However, after troubles cross validating this model we decided to switch to keras, the model in section 8. Our neural network takes 7 inputs through an input layer. This layer then passes these inputs through 2 hidden layers, each 100 units wide, using relu activation functions. Finally the output layer is only a one unit output that uses a sigmoid activation to classify a given row as high performing. The second neural net is a sklearn MLP classifier that is of the same structure as the above model except all activation functions are ReLU, and the learning rate is variable as opposed to fixed. Additionally we implemented a simple KNN classifier. To select the correct k value we tried k values from 1 to 40 and selected the k value that produced the best score.

The final model we chose to implement was a special kind of ensemble model known as stacking. This model is a combination of multiple other classifiers, in our case these were the AdaBoost, GradientBoost, and RandomForest classifiers we had made previously. Our stacking classifier helps to better generalize the results by leveraging the strengths of each individual estimator to make a final prediction. All the models, including this one, were then submitted to Kfold cross validation as well as stratified shuffle split to check which would likely have the highest accuracy on a more generalized test-set. Each of these utilized 10 splits with shuffling set to True. This made it easy for us to pick our best performing model and controlled for overfitting to the training data to a degree.

## Results

### Statistics:

Through a summary of statistics output by the describe() method we are able to observe some interesting trends. First, almost every type of load has high variation between drivers with some having 0 of that given load type and others having thousands of that same type of load. This holds true for the generic loads feature as well as while most drivers only carry 1 load per interaction, some have carried up to 26. In many of the binary categorical features there is a strong majority with most being the same as evidenced by the 25-75th percentiles all being the same value. We can also get a better idea of our time range by looking at the min and max years. It appears that tracking started in 2015 and goes up to the present. That being said, almost all of the data is from 2018 and beyond.

The next major statistical output was the correlation matrix displaying the correlations between each of the 20 variables with all others. Most important of these grid columns was the label column displaying each variable's correlation with a given row's label. Many of the variables ranged in a middle ground of around 0.19 to 0.39. The strongest predictors were 'port_qualified' with 0.56, 'driver_with_twic' with 0.53, 'year' with 0.61, 'dt' with 0.58, and 'id_driver' as well as 'first_load_date' each with 0.4 pearson correlation coefficients. These values are thus likely to be strong predictors of labels. On the other side of the spectrum, features such as 'dim_carrier_type', 'home_base_state', 'signup_source', 'brokerage_loads_otr', and 'brokerage_loads' were the lowest valued coefficients and have a lower likelihood of being significant predictors. The other insight this table grants us is that of which variables contain significant collinearity. Some of these were expected such as year and date or brokerage_loads and brokerage_loads_otr while others such as year and driver_with_twic were less expected.

**Model Results:**

Starting with our primary ensemble method found in section 7, we will examine the results of our AdaBoost model. We first determined that the ideal parameter ranges to test were between 0 and 2 for learning rate and between 1 and 500 for number of estimators. Outputs of the ideal parameters changed depending on the random downsampling but they were typically around 0.3 for learning rate and 350 for number of estimators. By training our model with these best parameters we were able to achieve around a 93% accuracy rate on the training data. When we used cross validation and stratified shuffle split, however, we achieved a higher accuracy of roughly 96.5% in both cases.

Moving on to our other two ensemble methods of GradientBoost and RandomForest, found in section 10, we saw roughly the expected results. RandomForest performed slightly worse than AdaBoost with cross validation accuracies of around 95.5% despite often having higher accuracies on the raw training data. Conversely, GradientBoost was actually able to outperform AdaBoost slightly. When it came to kfold cross validation both models had very similar accuracies of about 96.5% but GradientBoost was typically able to edge out AdaBoost when it came to stratified shuffle split.

For our neural network, found in section 8, we once again found results that were close to as good as AdaBoost, but still slightly worse. Our keras model described above in the methodology section had a kfold cross validation score of about 95.4% and a stratified shuffle split score of about 95%. This was more on par with the RandomForest model than our top two models of GradientBoost and AdaBoost. On the contrary our MLP neural net from section 10 ended up outperforming all other models with a cross validated accuracy of about 97.5%.

Despite this it didn't perform so well on the kaggle test set and thus overfitting is suspected. Our last stand-alone model was the KNN classifier. The ideal k value oscillated between 6 and 1, with the 1 raising concerns for significant overfitting. Despite this the KNN model did decently well scoring accuracies of about 95% in both cross validation scorings.

The final model we tested was our stacked classifier, an ensemble classifier that utilized the AdaBoost, GradientBoost, and RandomForest classifiers discussed previously to make more generalized predictions. While this model did not have as high of an accuracy score as either AdaBoost or GradientBoost, we hoped that it would generalize better to the separate score set. This was a sentiment we gained after testing alterations to our models that increased their cross validation scores but subsequently led to worse performance on Kaggle. That being said the stacked classifier still performed admirably with an accuracy of about 94% on both the stratified shuffle split and kfold cross validation tests.

## Discussion

From these results, we can predict high-performing drivers from a driver profile with no labels, total loads, or most recent load date to high accuracy. As seen, the MLP classifier has the best accuracy with a score of  about 97.5%. This predictor ended up getting worse performance on Kaggle, where our AdaBoost classifier did the best. The MLP is possibly overfitting, leading to worse performance on the Kaggle set. On our training and test sets, these predictors almost always classify drivers correctly which means that NEXT can take new driver profiles and relatively confidently predict whether or not they are high-performing.

NEXT can use this predictor to help recruit new drivers. Not only could NEXT run prospective driver's past records through their algorithm, but they also implement a trial period for a new driver. Of course, NEXT allows drivers to work when they want, but they also want to recruit the best drivers possible. Through a trial program (on maybe less time sensitive or important loads), NEXT could collect data on the prospective drivers and will be able to determine if they will be high performing enough for the company by running the trial data through the model.

In addition, NEXT can use this predictor to help predict the future loads that drivers will be able to make. If NEXT wants to grow their company and build customer trust, they will need to be able to take on more loads and never cancel on a customer. To better determine how many drivers a dispatcher will have available to them, NEXT can run the model on available drivers to see how many more high-performing ones they have. This will give NEXT more insights on how many customer requests that they can accept at a given time.

The next steps for NEXT data analytics may be to predict if a prospective driver will be high-performing without a given trial period. Our current model takes in NEXT data records where each row is a given trip. This means it includes many features such as load day, loads, weekday, and others that are specific to a given trip. If the data had only driver info and a model was created on that, they may be better able to classify new drivers before they hit the road.

## Conclusion

NEXT trucking is revolutionizing the way freight and shipping is done through smart customer to driver matching. In order for NEXT to continue to grow, they must be able to characterize high-performing drivers to better recruit the next generation of drivers and know their capabilities. Through a predictive model learning from all driver trips, NEXT will be able to take a given trip and decide whether the driver is high-performing with great accuracy. After pipelining the code and running through PCA to get reduced dimensionality, we were able to create multiple predictive models. Through our model creations of linear regression, AdaBoost, random forests, GradientBoost, neural nets, stacking, MLP, and K-Nearest-Neighbors, we were able to obtain the greatest cross validation accuracy of about 97.5% using MLP. NEXT will now be able to input new driver trip information to predict if the driver is high-performing. This will allow for dispatchers to have a better understanding of the remaining available drivers at a given time and help NEXT recruit new drivers.