Untitled Paper

Capstone Team 1, Washington State University

Jessica Hosler, Emma Johnson, Minju Lee, Blake LeValley, Routhana Long

# 

# Executive Summary

This paper presents the study of a Schweitzer Engineering Laboratories’ dataset. Analysis is done providing insights in the organization’s transactions. A data oversampling technique, lag features, and three machine learning algorithms are engineered resulting in a 20% improvement over base models. Additionally, basic math, statistical analysis, and visual analysis are done to find patterns. Some of the main patterns found are that the data indicates most of the transactions are Early. Another pattern is that the Customer Size 0 has, on average, more days open in the open to close data.

Recommendations are made in future data analysis applications which include the usage or gathering of prevalent data to build more robust models and add to the data analysis department at the organization. We find that the basic math and statistical analysis give us great information about the data, but visualization helps us see patterns more quickly. We find that Machine Learning was a difficult task with this data set, but we were able to compile a somewhat efficient machine learning model. Overall, all models proved to provide some beneficial information, but that the Basic Math and Statistics provided the clearest path to patterns and information.

# Background

Schweitzer Engineering Laboratories (SEL) is an employee-owned corporation started and headquartered in Pullman, WA. The organization manufactures and sells products to help maintain and protect power grids. To excel in providing the best customer service and experience, SEL utilizes data analysis to catch patterns and streamline their processes.

In this project, the project mentor and SEL analyst, Wende, had the ability to reconstruct data with *simulated information* about certain aspects of the company. This data contained data on shipping, delivery, customers and more. As data analysts, our group was asked to discover patterns and provide valuable information from these data sets, just as data analysts in the company have been asked to. Specifically, our task was to focus on time-series data, data about customers, and different analytical strategies. This would allow us to form conclusions about the methods we use and patterns we find.

## Questions

Several questions were formed to help guide us all throughout the project. These questions focus both on the methods of research and the information to discover from the data. The questions were:

* Which method is most beneficial for analyzing the data: basic math, statistical analysis, or machine learning?
* What patterns can be found in the data?
  + What patterns about the overall customer ID (the specific customers)?
  + What patterns can be found using the time-series information?
* Visually, how can we best represent the outcome of this analysis so it is explanatory, efficient and beneficial?

With these questions in mind, we proceeded to analyze the simulated data provided to us using a variety of methods while looking for different patterns and other useful information. We performed this analysis as if we are performing it on real data, and with the intention of finding information to help streamline any processes or procedures in the company in question.

## Hypothesis

Looking at how similar data should be handled in the future and what would be most beneficial for the data analytics team handling a similar but factual data set, the hypothesis is that basic math looked at from a statistical standpoint would be most beneficial, as displayed in our application. This is because it gives a basic level of understanding how certain customers may compare or perform and can still help see patterns and understand what next moves should be for these customers. For instance, in our application we have a few tabs which help us look at individual customers, which are very beneficial when you’re attempting to see which customers are consistently getting quality service, which ones may have large orders that take a longer time, etc. This is most beneficial when taking the angle of trying to understand how particular customers are receiving the services. In one tab called “Delivery Metrics” there’s a graph which shows how certain customers are placed on a graph of delivery metrics, and the user can scroll over the data points and see the customer ID. Then in the “Customer Metrics” tab, you can enter in that customer ID and see the overall metrics from the data for this particular customer, as well as the top performing customers in certain criteria. This means in the future, when the SEL team would like to see how certain customers compare, they are able to enter in their own data using the load data functionality of the application and evaluate how certain customers compare. Additionally, machine learning may be very beneficial for making predictions based on certain feature sets. Our application allows for different variations of machine learning, with different feature sets in order to see which types of ML produce information at the desired level of quality.

## Problems

Since the project consists of simulated data, there were a few problems that were encountered. The first major problem was with regards to data cleaning and this encompassed several issues that needed to be addressed. These issues included:

· Null or blank spaces

· Inaccurate information or data types

· Impossible observations (e.g.: closed dates before open dates)

· Redundancy of information

These issues motivated us to clean and modify the data in order to obtain a data set which was more easily analyzed through the R programming language and additional libraries.

The second issue was that there might be some problems with actually formulating patterns in the data. Since patterns might have not been developed in the data, there might have been some difficulty coming to any real conclusions about the data. We had to keep this in mind when we were using certain analysis tools such as the building of machine learning regressors and classifiers. Machine learning depends heavily on known patterns to make predictions. This was kept in mind and noted as needed for the machine learning process.

Lastly, a problem area which had to be addressed came during the joining of tables by the overall customer ID. Currently, the data is in terms of a location customer ID, which is more granular than the overall customer ID. We were asked to perform our analysis by the overall (less granular) customer ID to maximize the understanding of analysis for certain customers.

## Objectives

The overall objective of this project was to examine different approaches to effectively analyze the data provided by SEL. By understanding the capabilities of these methods, this could provide the organization with a basis to distinguish which method was most useful for data analysis on future, similar data sets. Additionally, we can show Wende a broad view of the data analysis methods we have been learning in our education.

Specifically, machine learning as a newer method of analysis. We are performing an analysis and developing a justified conclusion about patterns found, how we found them, and what methods helped us get to those conclusions. We are also making an application which could be useful for Wende to perform her own analysis on future data with the techniques we will be performing.

# Literature Review

In the beginning of our process, we were given 5 different tables of data each connected by some primary or foreign key of customer ID. To understand our data and how to use it, we needed to do some data exploration. A 2015 article by Srivastava on Analytics Vidhya provided some foundation to our research. Titled “Comprehensive guide for Data Exploration in R”, techniques from this article were used to find ways to discover data patterns and understand the data more. In part 1 of the document, we learned how to load data from a tab separated file or a csv file. This was useful for us because our original data came in an xlsx file format with different sheets which would eventually be converted to csv files.

Part 5 of the article assisted us in making histograms, allowing us to understand distributions and proportions of data we have (Srivastava, 2015) which included, for instance, understanding how the Customer Size bins were proportioned over all customers. This was important to know and keep in mind since not all the data was evenly spread among each attribute. Parts 10 and 11 were vital for the transformation of information by joining certain tables and eliminating some null values (Srivastava, 2015). We used this to understand how certain attributes were related such as type and dates. These did not exist in the same table, so we had to join them together under the guidance of this article.

Additionally, we used some different libraries and models in our analysis we had little to no experience in. Because of this, we utilized online documentation often to guide us. This includes the Shiny documentation because none of us on the team had previously used a Shiny app with this purpose.

Some of the more helpful pieces of this documentation were the examples of other applications, and format. They were helpful because we decided to showcase what we learned through the creation of tabs on the side of our application which each had a different function. One of the main functions was having the ability to upload your own data. This was developed through the document titled “File Upload Control” which outlines the bare bones of having a function to upload data (“Shiny - fileInput,” n.d.). We were then able to take this outline, adjust it for our needs, and now we can upload our own data to the application.

### Data Augmentation with SMOTE

Class imbalance is a problem that is not only prevalent in this application, but also very prevalent in many classification applications. Even properly sampled, split, and cross-validated models struggle to classify minority classes since objective functions traditionally attempt to maximize metrics that with highly imbalanced classes may be easier to predict by maximizing the amount of majority classes it can correctly predict. Synthetic minority oversampling technique (SMOTE) is a method designed to oversample the minority class by data augmentation to bring more balance to imbalanced classifications.

Inspired by data augmentation techniques from image classification researchers in the 1990s, SMOTE uses KNN on the feature space (Chawla, 2002). It randomly selects closest neighbors and draws a line between these neighbors. Afterwards, a random value within this line is randomly selected as a value for the feature and this technique is repeated over and over to create data. SMOTE is effective in that it counters the issues of random oversampling. Random oversampling random selects and duplicates data which can lead to model overfitting. SMOTE addresses this by creating data that is plausible yet almost original. Along with the previously mentioned techniques, this project employed SMOTE to address the imbalanced classes in this application to potentially improve model performance.

# Methods

Data cleaning and manipulation was performed mostly using the R statistical programming language. The data was parsed into a local SQLite data store using Python. Exploratory data analysis was performed using R with libraries typical in data analysis such as dplyr, ggplot2, and others. Machine learning was performed using R, Python, and Weka.

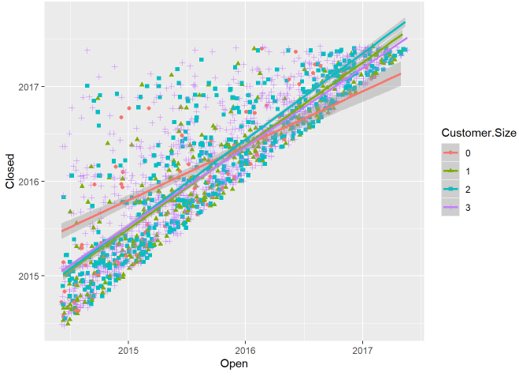
### Data Cleaning and Management

Data was initially delivered in Excel format. The data was parsed and transformed in R, removing null values, converting dates to proper date-time format, and removing observations that were incorrect such as negative parts delivered and closed dates occurring prior to opened dates. These operations were completed prior to saving to CSV format. This was a necessity due to the errors and slow run-time while processing the original Excel file due to a bug in Table B. The data was then migrated to a local SQLite store.

Data migration from CSV files to the SQLite relational database (RDB) had one requirement--to design initial relations and force its constraints. It was a suggestion by the mentor and could help streamline queries in a team using multiple languages and technologies. For instance, instead transforming data twice in Python and R, the query could be reused in both languages.

### Exploratory Data Analysis

Exploratory analysis was integral to the project. It began by examining each individual table, looking at statistics and graphs for potential patterns and clues so insights could be presented to the organization and optimal routes could be identified in applying machine learning algorithms. This included analysis of data distribution which is a necessity when understanding the applicability of parametric and non-parametric methods. We also performed basic statistics on each of these tables to better understand any general patterns.

We then moved on to merging some tables together to see how attributes connected and worked together. All tables had some form of attribute “Customer ID” which was labeled differently in each table but corresponded to the same customer number. We merged each and analyzed the patterns found, or the lack of patterns found. 

We then performed some visualizations of the statistics found to view the patterns. These visualizations included histograms, time-series plots, and plots against certain attributes. Some of the most important visualizations from this were found to be the plots against certain attributes. We used one plot, seen above in Figure 1, to assist in cleaning our data.

This plot shows the Closed Dates against the Open Dates, which helps us see if there are any chronological inaccuracies in the data. For instance, before cleaning the data we had some points which were below the upper triangle (in the lower triangle) which indicated that some points had “closed” before they had “opened” therefore not making logical sense. We used this graph to find these points and remove the inaccurate data.

\*\*ADD

### Data Selection and Machine Learning

Machine learning used two datasets, the original dataset and a dataset including transformed lag variables across three algorithms: logistic regression, decision tree, and the random forests classifier. The two datasets were chosen due to a variety of factors including the number of null values and usefulness to the organization.

Table B contained the features Requested, Early, On Time, and Late as continuous data. These variables were defined as the number of parts that was delivered given a request. The decision was to predict whether requested parts would be on time or late, encoded 1 or 2. This resulted in a 2-class classification problem where this new categorical variable Delivery Status was created to be used as the predicted output for both datasets.

Since the dataset contained few features, several new features were created for the dataset including lags. The first was the Delivery Status. The rest included 5 lag variables each for Delivery Status, Requested, Early, On Time, and Late looking back at the latest 5 observations for delivery status and requested parts for each customer ID. This data was split through stratified random sampling to generate a 90% training set and 10% test set that represented the original distributions of the dataset.

The three algorithms were selected to help distinguish the tradeoff between interpretability and performance. Logistic regression and shallow decision trees are often seen as being the most human interpretable. In contrast, random forests can operate more as black boxes especially when the number and depth of trees of the forests makes it more difficult in terms of human interpretability.

All models were hyperparameter tuned resulting in 9 models. Models and hyperparameters were tuned according to the results of stratified, shuffled 5-fold cross validation to account for variations and the data imbalance that could lead to poor generalization. During the cross-validation sampling process, Synthetic Minority Oversampling Technique (SMOTE) and random under sampling were used to subsample data that would counteract class imbalances but still generalize data well. Though train and test performance were examined, the average cross-validated score was employed for model selection from hyperparameter tuning and for model comparisons built on the base dataset and the new dataset with lag features. These models were averaged upon weighted F1 scores. The logistic regression models were tuned with L2 normalized data for quicker convergence and training.

# Results and Discussion

## Exploratory Analysis and Statistics

### Basic Math and Basic Statistics

In R, the summarize function was used to calculate basic statistics about the dataset. Table B and Customer Info were merged by Customer ID to get the average number of units that were early, late, and on time for each Overall Customer ID through aggregation. The same was done for Customer Sizes 0-3. This can then be used to predict numbers for future orders based on the size of the customer or the Overall Customer ID. The same process was used to calculate the average number of days elapsed for projects to be completed for Overall Customer ID and Customer Sizes 0-3.

This method is useful when trying to understand some trends as to which companies are consistently having their orders take longer than the average or which customer sizes are getting their parts delivered consistently early while others late. Using the basic math is helpful in predicting future orders but may not be the most accurate since there is a high amount of variance in the data.

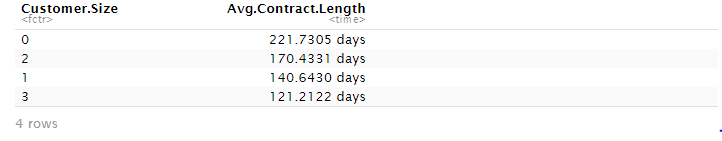
Some of the more tangible information and patterns we were able to discover came from Table B. This is because Table B contained information about an order or sale and how many units were requested, late, on time, or early. With this we were able to find statistics which were valuable to seeing how these orders connect to other attributes or tables. From our exploratory analysis we found the table below in Figure 2 to be the means of each.



*Figure 2: Table B Means*

What this tells us is that on average, more of the units requested are early than they are late or on time, but more of the units are late than they are on time. This makes sense since a due date is usually a difficult target to meet exactly. This is also positive news based on this data set and this data would suggest deliveries are in the right direction, where most of the deliveries are before the due date. However, it’s important to note that this is a very broad look at products, and it may be useful to get more data on what products are being sold in each tuple, which could tell us more about which products are causing late deliveries, maybe there’s an underlying issue with the supplier’s ability to get us the products needed to ship out a finished product. Additionally, grouping by product may help us get a better understanding of the requested units. The range of requested units is very wide from 1 to 750462. This means more details may be necessary to make proper understanding of the information.

In Table E one of the more tangible sections was the open to close data. We used this to create a column which told us how long this data was “open.” From there we connected this to other tables such as customer information to learn more about customer size. In Figure 3 we look at the average days open for each corresponding customer size.



*Figure 3: Average Days Open for Each Customer Size*

From this table we learn that customer size 0 on average has longer days open than 1, 2, and 3. This is statistically significant in the way which it begs the questions:

* Does the length of days open correspond to the customer size?
* Why does the largest customer have the smallest average length of days open?

With these questions in mind, we can look at the data as a whole and see that customer size 2 has longer average days open than customer size 1, which may indicate that the two are not related directly.

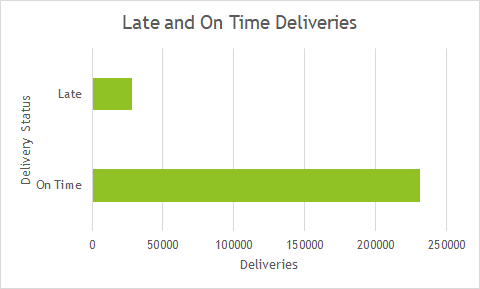
## Machine Learning

The feature selection process began by questioning the usefulness of features as outputs in applications for the organization. The features in Customer Info had unique identifiers and Customer Size. Customer Size was not an output candidate since the organization should be able to look this up or ask the customer for an immediate and accurate response. The same could be said of Type in tables A and D which are noted upon data entry. The features that could be of use would be the time between transaction Closed Date and Open Date or Requested, Early, On Time, or Late parts.

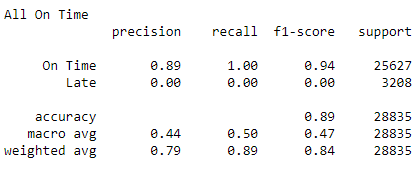
The length of a transaction was not a viable output candidate since these amounts of days can vary requiring constant updating for an accurate regressor. Requested parts had too little information to predict while using any of Early, On time, or Late parts features would result in leakage of future data compromising the model. Thus, the decision was to convert Early, On Time, and Late values into a single column indicating if parts were on time or late.

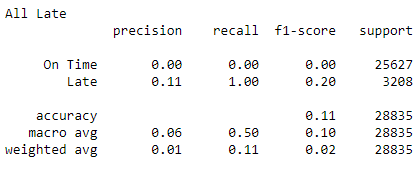
This data was then joined with the Customer Info table so that the Customer Size would be an input. Other tables were not joined since there would be a loss of information due to the differences between the IDs in Table B as opposed to tables A, D, and E. The Customer Info table, though, contained all IDs across all tables, resulting in no loss of information upon a join.

The joined data was split through stratified sampling with a 90% training set and 10% test set resulting in 259506 and 28835 observations, respectively. Class imbalances showed that most transactions of parts were delivered on time. Around 89% of transactions were On Time while 11% were delivered on time. The distributions were roughly preserved, with classes being within 0.3% of the original proportions (FIGURE NUMBER).

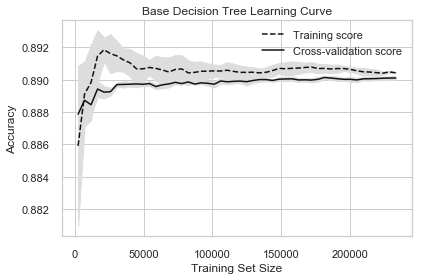


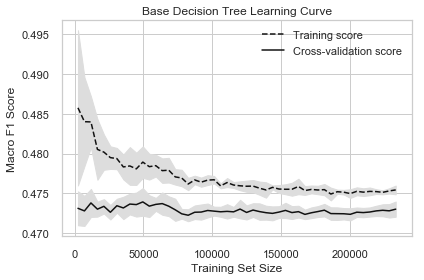
This class imbalance also leads to the following performance if an individual were to simply claim that all transactions are On Time (FIGURE NUMBER) or Late (FIGURE NUMBER). Naturally, recall would be perfect upon each specific instance, but precision is very low in the case of All Late predictions.





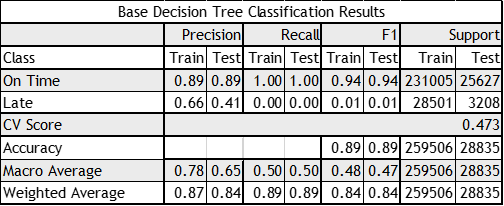
The decision tree models were the first to be built. The base decision tree took about 1.8 minutes to finish 400 models. The best model had an average macro F1 score of 0.473 for cross-validation. The CV training and test results can be seen in FIGURE NUMBER. Learning curves were all similar, showing convergence of data across the training set.

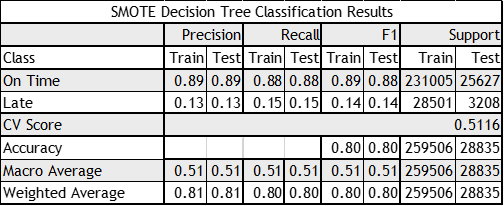


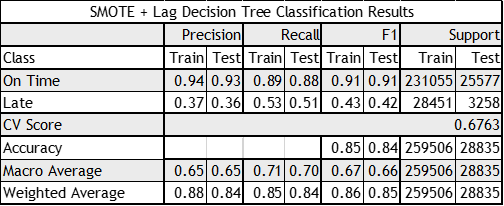


Addition of SMOTE oversampling to the training process resulted in some pros and cons. The macro score, which accounts for the class balance, increased from a cross-validation average of 0.4730 to 0.5116, which is almost a 4% improvement. Though the model sacrificed in On Time classification, it improved in Late classifications as overall accuracy dropped from .89 to .80 for both the training and test sets from the base model. Unfortunately, oversampling increased the training time for the 400 tasks from 1.8 minutes to 22.5 minutes which is understandable since it is an extra process for every hyperparameter combination used in every iteration of grid search cross validation. Since training time was significantly increased, fewer hyperparameters would be used in subsequent instances of model training.

The addition of lag variables to the SMOTE-based decision tree resulted in a jump in accuracy. The CV score increased by 0.1647 from 0.5116 to 0.6763. [FIGURE] shows how the decision tree splits data, which is a stark contrast from SMOTE alone and about a 20% increase from the base model.



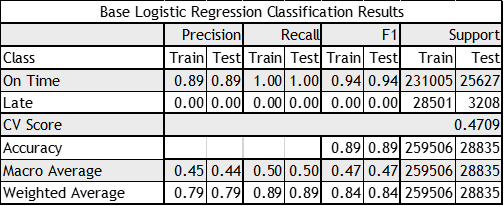


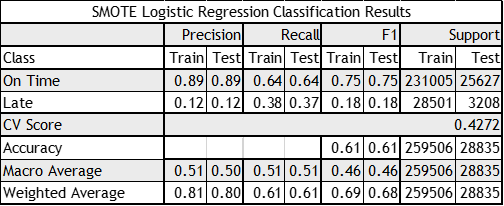


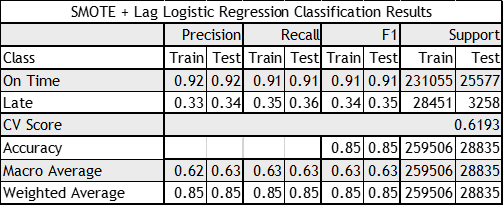
The base logistic regression model resulted in a CV score of 0.4709 which was comparable to the 0.4730 score in the base tree model. The addition of SMOTE sampling to logistic regression did not yield an improvement, though, with about a 0.045 decrease in macro F1 CV score to 0.4272 which was largely due to a massive decrease in On Time classification recall. Like the decision tree models, SMOTE improved both precision and recall for Late delivery classifications, increasing precision by 0.12 and recall by 0.37.

[Logistic Regression Decision Boundary]

The addition of lag variables increased the performance of logistic regression with oversampled data. The CV score increased to 0.6193 while improving in precision for On Time classification and both precision and recall for Late classification.

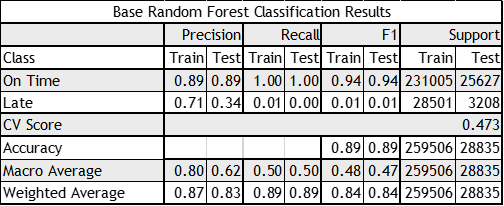


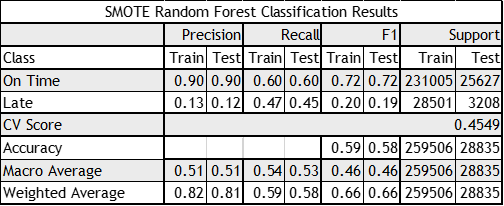


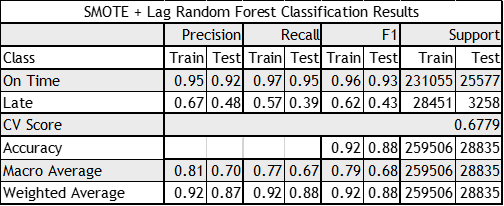


Random forest displayed similar results to the other algorithms for these data sets and sampling strategies. The base model had a CV score of 0.4730 with improved train precision and worse test precision for the late class and seemingly little ability to recall the class. SMOTE resulted in a decrease in performance for random forests. CV score decreased almost 2%. While recall increased from zero to about 45% for On Time deliveries, all other metrics decreased.

Lastly, the addition of lag variables to the random forests model with SMOTE either maintained the same metrics across the board or increased them. The CV score was similar to that of the lag and SMOTE based decision tree classifier at 0.6779.

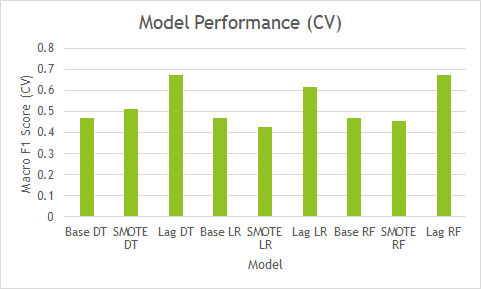


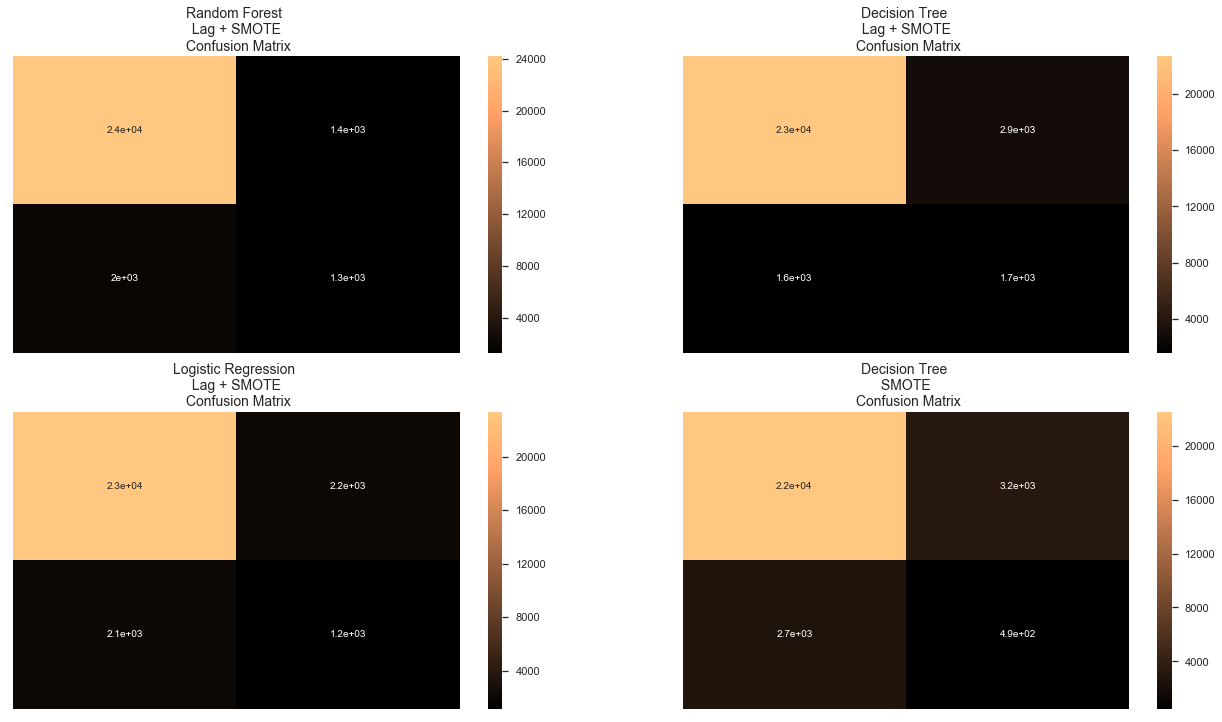




The best models can be seen below, with the four best models being random forests with lag and SMOTE, decision tree with lag and SMOTE, logistic regression with lag and SMOTE, and decision tree with SMOTE. The two best models were able to maintain greater than 90% precision and recall for On Time classifications which is slightly greater in precision and slightly worse in recall as opposed to blindly predicting everything as On Time. A blind assignment of On Time predictions would result in 0 precision and 0 recall for Late classes, though, and this is where the best model, the random forest model, made its greatest improvements, pushing to 0.48 precision and 0.39 recall.

Thus, it was shown that additional data could bring the organization a potentially useful and accurate indicator.





# Conclusions

The data revealed nuggets of insight acquired by cleaning and gleaning over the information and its transformations. Exploratory data analysis and statistical methods provided us with the most valuable insight, as key correlations were uncovered throughout the process, such as customer size and their relation to Requests.

Overall, experimentation with the machine learning models showed that lag variables and oversampling greatly improved models over the original data or simply data augmentation through SMOTE. Smote had varied results alone. It improved the decision tree classifier over its base but decreased the performance of the random forests and logistic regression models. This came not only at the cost of precision and recall for the majority On Time class but also at the cost of even Late classification precision.

Future work can explore building models for class probabilities instead of class prediction, which could give the organization slightly more insight for decision making by providing probabilities across the Early, On Time, and Late classes. Furthermore, the organization should consider analyzing and use features from other information or gathering them to potentially improve insights and predictive modeling.

The data showed that machine learning to this dataset is possible, but additional data, whether it be augmented or gathered from other sources, are a necessity.

## Reference List

Brownlee, J. (2020, January 14). 8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset. Retrieved May 1, 2020, from <https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/>

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321–357. doi: 10.1613/jair.953

Srivastava, T. S. (2015, April 26). Comprehensive guide for Data Exploration in R.

Retrieved from <https://www.analyticsvidhya.com/blog/2015/04/comprehensive-guide-data-exploration-r/>

Shiny - fileInput. (n.d.). Retrieved April 10, 2020, from <https://shiny.rstudio.com/reference/shi>

ny/1.0.0/fileInput.html