Delivery Delay Prediction: Amazon Master Modeler Competition

Arora, Mansi
marora33@gatech.edu

Lokhande, Jeh
jeh@gatech.edu

Usmani, Nabila
nabila@gatech.edu

I. INTRODUCTION

Amazon shipments are adversely affected by unpredictable and seasonal weather patterns throughout the country. Weather events such as hurricanes, blizzards, rain cause power outages, dangerous roads and hazardous conditions that affect the delivery of packages. Trucks, trains, and airplanes cant navigate the massive influx of winter weather fast enough to support demand. Snow and ice make it impossible for planes to safely take off and land; trucks are slowed by congested, unplowed highways; and railroads cant clear the tracks fast enough to get through. Like a domino effect, these delays lead to other, often costly, problems for supply chains. This is a complex problem, as weather conditions depend on the infrastructure in the impacted region and can have different effects depending on the region. Predicting delivery delays can be very helpful for Amazon and its customers as it will improve scheduling, customer experience and competitive advantage under uncertain weather conditions.

II. DATA

A total of 24 files were provided:

- 1 file containing actual weather data
- 1 zip code-date level mapping file
- 22 files containing data for weather attributes

The actual weather data has not been utilized as this information will not be available when forecasting for 42 hours into the future. Instead we have used the forecast weather files. Each weather attribute file (e.g. wind, gust, snow, etc.) consists of forecasts of that weather attribute for all zip codes at an hourly level. The common attributes that binds these files are zip code and date time. The response variable predicted can be either a binary variable (delay or not) or a quantitative delay score. The binary variable is modeled by tuning a custom made cost function using the continuous score. The binary prediction of delay is what the business requires and hence is out choice of output.

III. APPROACH

The figure illustrates the framework of the approach taken and the steps followed, outlined as below:

- Data Preparation: Many weather attributes have data that is extremely sparse and handling missing values was a major part of the data preparation process. The data size of 17.5 GB posed an additional challenge in consolidating the weather attribute files. Forecasts for different weather attributes were at different resolutions: some variables such as temp (temperature) had hourly forecasts available from 60 hours up to 10 hours before while others such as asnow (average snow) had forecasts available only 48 hours into the future.

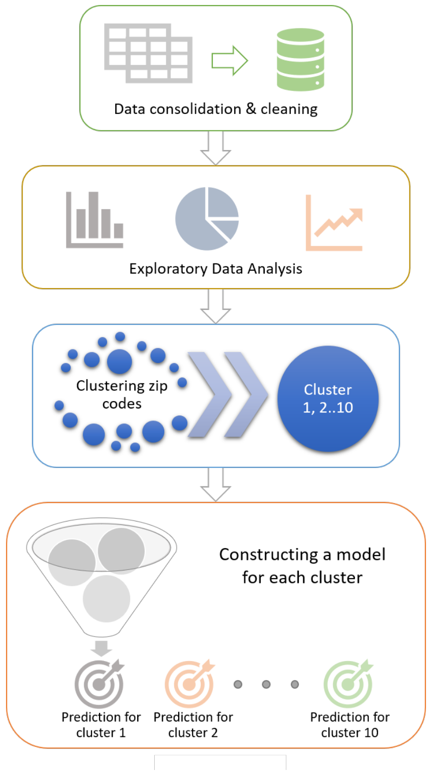

Data consolidation & cleaning
Exploratory Data Analysis
Clustering zip codes
Cluster 1, 2..10
Constructing a model for each cluster
Prediction for cluster 1
Prediction for cluster 2
Prediction for cluster 10

Fig. 1: Data Process Flowchart

- Clustering: Ideally, the models developed should be at a zip code level to capture the unique features of the zipcode such as transportation connectivity, sensitivity to weather events, etc. However, the data for delays (postive binary scores) is very sparse at a zipcode level. The number of delays also has potential seasonal patterns that are hard to capture with sparse data. An additional consideration is also that there are about 43,000 zip codes in the US, and 1423 zip codes in our data, so having a unique model for each zip code does not lend well to scalability of the models. As a means to combat the above issues, clustering of similar zip codes is performed. Here similarity is defined as being geographically close, having similar weather attributes as well as having similar correlation between weather and impact scores. In short, we are clustering zip codes that react similarly to weather conditions and that have similar climatic conditions.

IV. DATA PREPARATION

The following steps were carried out for preparing the training dataset:

- A cleaning script was developed to standardize and clean each individual weather attribute file. Post cleaning, each weather file was transformed to contain 4 columns: zip code, date, hour, and value.
- The cleaned weather attribute files were merged with the parent zip code date file using zip code and date as a combined key. This merged file consisted of 25 columns: zip code, date, hour and 22 weather features.
- Since the data was at an hourly level, it was aggregated at zip code and date level, ignoring null values and 6 features were created for each weather attribute. These were mean, median, maximum, minimum, standard deviation and range. These features were created to better capture not only the change in forecasts for the same day over time, but also the change in weather attributes over the past 2 days.
- Post aggregation, some feature values still consisted of null values. These were handled differently depending on the feature. Variables such as ptornado (probability of a tornado) and phail (probability of hail) that had a high number of missing values were dropped from the dataset as it is hypothesized that their impact has been captured in other variables such as wind and ice respectively.
- The final training dataset consists of 1 million rows and 45 columns.

For preparing the testing dataset, the same steps are carried out except the aggregation step for the weather attributes. Here, instead of aggregating all values, only those 48-hour forecasted values are considered, since the objective is to be able to predict delays 48 hours in advance.

V. CLUSTERING

The k-means clustering algorithm was used to aggregate zip codes into clusters. To estimate the optimal number of clusters, the variation of the intra-cluster and inter-cluster distance with the number of clusters is used. Using this criterion, k was chosen to be 10, also keeping in mind that the final number of models. The following is a visualization of the 10 clusters obtained:

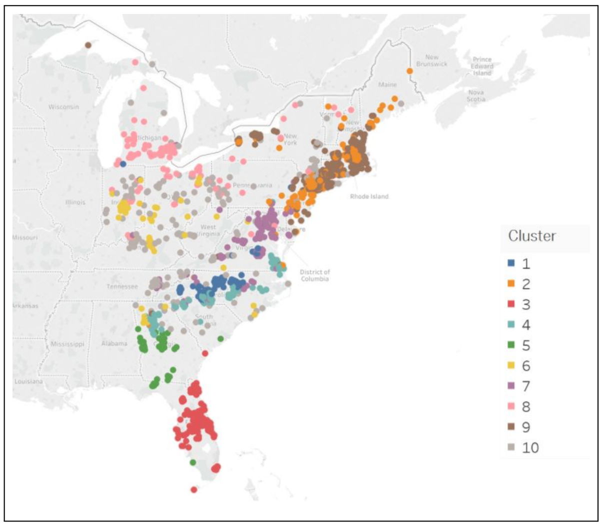


Fig. 2: Zip Code Clusters

As illustrated in the map, while some clusters have zip codes that are geographically very close such as cluster 3 (red) that has all but one zip codes in Florida, there are others that are more geographically dispersed such as cluster 10 (gray).

VI. EXPLORATORY DATA ANALYSIS

A comprehensive analysis of the trends and seasonality of the weather variables in conjunction with the impact score was conducted for each cluster.

The following chart illustrates the percentage of delayed package deliveries by month averaged for years 2015 to 2017. The winter months of January, November and December seem to face the highest proportion of delays for almost all clusters. Cluster 5, that mostly consists of zip codes in Georgia, has faced the highest percentage of delays in September.

This can be attribute to the unusual event of Hurricane Hermine that affected Georgia in September 2016.

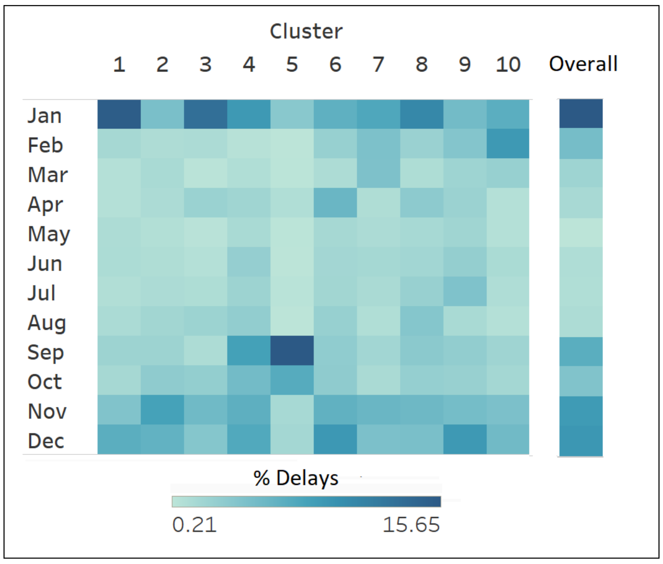
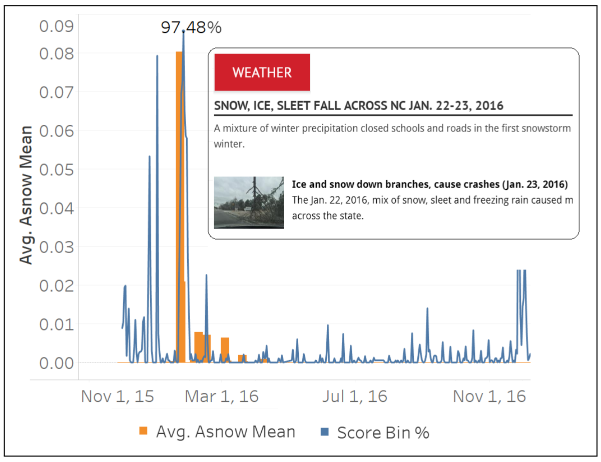


Fig. 3: % Delay by Clusters-Month

For cluster 1, that has zip codes mostly in North Carolina, the month of January has the highest proportion of delays. A closer look shows that there was a huge spike in snow in the region on Jan 22-23 2016 that caused about 97% deliveries to be delayed in that cluster.



SNOW, ICE, SLEET FALL ACROSS NC JAN. 22-23, 2016

A mixture of winter precipitation closed schools and roads in the first snowstorm winter.

Ice and snow down branches, cause crashes (Jan. 23, 2016)

The Jan. 22, 2016, mix of snow, sleet and freezing rain caused m across the state.

Fig. 4: % Delay - explained through live weather updates

Analyzing clusters with respect to day of the week, as illustrated in the chart below, indicates the presence of weekly seasonality in the data. This has been captured in the form of day-of-week features created using one-hot-encoding.

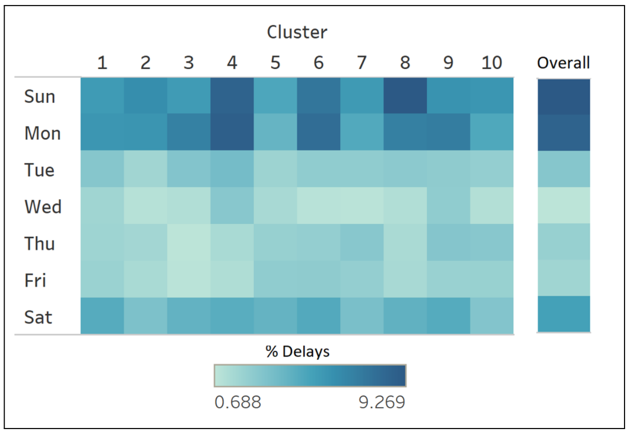


Fig. 5: % Delay by Clusters-Weekday

VII. FEATURE ENGINEERING

Many transformations and combinations of weather attributes were experimented with and some of there were found to be insignificant whereas some were more predictive than the raw weather attribute. The following are some of the features created to improve model fit:

- To capture monthly and weekly patterns, binary variables were created for each month and weekday and incorporated in the model. Both sets of variables were significant predictors and while weekday variables were incorporated in the final model as is, variables representing month were not directly used, as the final models created were by season. That is, for every two months a separate model was created.
- As holidays are a period of high demand, this may strain the system and may lead to delivery delays, binary variables were created to represent federal holidays. However, these were not found to significantly impact delivery delays and hence were not used in the final model.
- To capture the impact a weather event can have on the next few days, features using 7 day moving averages were created for all weather attributes.
- Variables that measure the difference between the weather attribute's last observed value and its forecasted value were also created as sudden changes in weather may impact delivery delays.

VIII. TRAINING

Various classification algorithms such as Support Vector Machine (SVM), Random Forests were experimented with. These algorithms albeit are great clustering algorithms suffer from performance issues when the data volume increases. Since gradient boosting penalizes misclassified data points and blends well with a customized cost function, we decided to implement the same using XGBoost.

After iterating through multiple combinations, the final models created were at a cluster and season level. Season here is defined as a block of two months, paired by analyzing their weather patterns. December-January is defined as one seasonal block, as they have similar weather attributes as well as as delay patterns.

A. *XGBoost*

Extreme Gradient boosting or XGBoost is an ensemble technique that implements many improvisations in the traditional GBM (Gradient Boosting Machine) algorithm. XGBoost uses a more regularized model formalization to control for over-fitting, which gives a better performance. It is also implemented to be much more computationally efficient than the traditional GBM algorithm.

Why XGBoost?
- Highly efficient gradient boosting algorithm
- Can be Parallelized on multiple CPU/GPU cores
- Can handle highly imbalanced classes
- Can be regularized using L1-L2 penalty

B. *Implementation*

- For each cluster, the dataset was split into 6 seasons, where each season is a combination of two months. A separate model was built for each cluster-season combination, with the categorical response variable score_bin.
- Feature selection for each model has been done by using the correlation between the weather attributes and the score variable in order to select a minimum number of features that explain the maximum variability in the score.
- The dataset is split into training (80%) and validation (20%) sets, wherein the validation set is used to tune the hyper-parameters of the model.

C. *Parameter Tuning*

- In order to optimize the maximum depth of a tree and weight associated with high class imbalance, a grid search was performed to select the parameters with lowest associated cost on the validation dataset, as per the cost of false positives and false negatives defined by Amazon. This optimization was performed for each cluster separately.
- The learning rate was set to 0.1, based on prior experience with the XGBoost algorithm, and other hyper-parameters were tuned around this value.
- The testing data is also split by season and cluster and the corresponding model is used to predict delays in the test set.

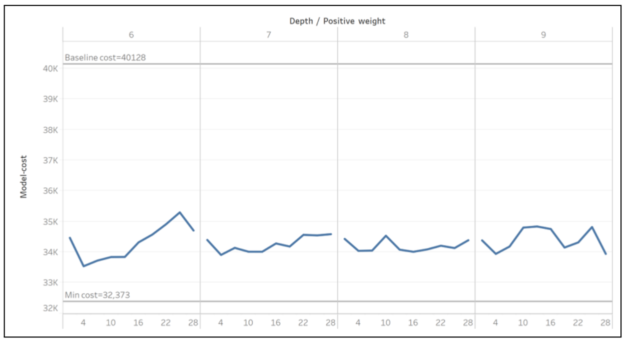


Fig. 6: Parameter tuning for a cluster

- The above chart shows how the cost changes as we tune the max_depth and scale_pos_weight parameters of the XGBoost model. The curve gets flatter as the max_depth is increased. Furthermore, the cost decreases as the scale_pos_weight parameter is increased, till a point but then increases afterwards. This can be attributed to high class imbalance in the data.

IX. MODEL RESULTS

After variable selection and model level (cluster-season), the models were able to capture the variability in the weather data. Since variable selection was done based on correlation between the score and weather data, we limited the weather attributes to between 3-5. The below figure is the variable importance chart for cluster 10, season December-January, where temperature, snow and APCP (average precipitation) have a very high importance in prediction. Other attributes such as wind, wdir and gust do not play a role in this model, thereby reducing redundancy and overfitting in the model.

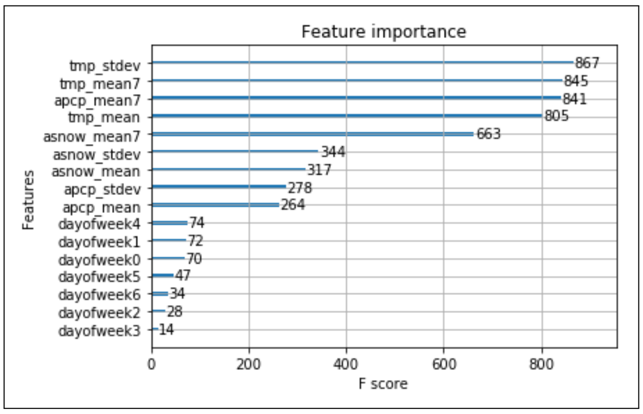


Fig. 7: Feature importance in XGBoost

The cost of misclassification of the different clusters is posted below. The bars in gray depict the cost incurred by the model. The red lines imply the cost incurred when we predict all zeros as the output. The blue lines are the minimum cost that can be achieved. Since the model is designed to classify delays more aggressively than non delays (since we can affort to inform a customer of a delay and then deliver on time, than not inform him/her and deliver late), the models performs better than a naive zero prediction when the season/month actually has a lot of delays. However in case of seasons/months with less delays, the model does nearly as well if not better than a naive zero prediction.

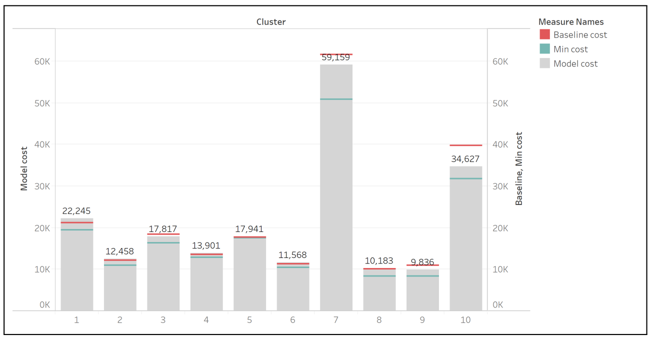


Fig. 8: Cluster wise Model Performance

X. CONCLUSIONS

A. *Solution Creativity*

- The solution implemented was developed taking into account not only the weather patterns but also other attributes such as holidays (to account for shopping patterns) and date attributes (to account for general delivery patterns followed by Amazon, such as no deliveries on Sundays might put pressure on deliveries on Monday). These attributes though were not significant to be included in the final model.
- The solution also takes into account how forecasts might change, and their impact on preparedness to handle such situations, for example; a quick change in forecasts for a day might affect the city council to deal with the situation thereby resulting in delays
- Clustering zip codes based on how zips react to weather rather than just geographic factors helps in taking into account the fact that certain zip codes in the same geographic region might be differently equipped to handle weather conditions (eg: rural and urban regions might react differently to handle snow)

B. *Ease of Implementation and Scalability*

- The XGBoost algorithm is simple but effective algorithm to implement where you can experiment with a variety of model parameters
- It is a highly scalable algorithm with the ability to parallelize operations on CPU or GPU thereby providing us the ability to scale the algorithm to handle large data volume. The current model takes less than 5 minutes to train for the whole data, while the predictions take less than 10 seconds to generate. Since additions of zip codes will increase the number of models and not the data in each model, the time taken to predict results will **not increase**. Please note that this model is being run on a machine with **16GB RAM**, and a **2.2 GHZ** processor with **4 cores**. All the computations are being done on the CPU. Computation time can be further reduced if we use a machine with higher processing power and by running the XGBoost model on GPU instead of CPU
- The data generation initially took a long time since it required joining on the large data files. The weather data can be stored in a data warehouse (a SQL warehouse would suffice given the scale of the data) and an ETL process can be built to continually load new weather data
- The Train_Model script can be scheduled to run every night after a data refresh to train the model on the latest data
- All in all, the predictive model is highly scalable and will cause any issues, when scaled to all the zip codes in the US. The extracted data that is cleaned is also far lesser than the maximum size that can be read into RAM, and hence will not pose an issue

C. *Further Improvements to the model*

- Weather data often has a lot of noise. Models that fit well to a certain pattern in the data might not capture some other patterns. A lot depends on the training split supplied to the model. Improvements in prediction can be made using an ensemble of models by averaging the predicted probabilities or by selecting the majority of binary predicted values from all the models to predict delays

- Predictions will improve if there isn't any missing data. We can hence include a script to parse weather.com or NOAA to extract data for missing days
- Data for tornadoes or weather warnings are sparse and not needed in model building. We can instead add a pre-check on these variables before model to pro-actively flag zip codes that have a high probability of a tornado or a severe weather alert.