UNIVERSITY OF CALIFORNIA - SAN DIEGO
DEPARTMENT OF DATA SCIENCE

DSC180A - Quarter 1 Project
Flock Freight - Section A07
Offer Acceptance in the Freight Industry

Keagan Benson, Nima Yazdani, Benson Duong, Radu Manea

Fall Quarter
2022

Abstract

Flock Freight is a company that deals with different carriers and shipments, acting as a marketplace for freight carriers to place bids on shipping orders needed to be fulfilled, which is otherwise known as a "freight broker". Flock Freight is paid by other companies to carry their shipments and then outsources these orders to freight carriers. The company commonly faces issues concerning the bid acceptance process. Some shipment orders receive multiple offers, and minimizing the cost of orders is essential to the business in that it maximizes profits. In this project, we will cover a machine learning method to develop the optimal stopping point for the acceptance and reneging of delivery offers by carriers for Flock Freight's orders. Time is money, and any time spent being indecisive means loss of carrier clients. It is thus essential to provide a model and process that is reliable and scalable to Flock Freight's needs.

1 Introduction

1.1 Background

Flock Freight acts as an intermediary in the shipping industry. Shipping carriers and shippers trust Flock Freight as a "freight broker", providing a way to easily ship freight across the country, specializing in connecting multiple shippers' loads with a carrier as a "pool", in what is known as a "Shared-Truckload" (STL). Flock Freight receives orders from shippers, which carriers will bid on to take the load. Flock Freight then decides which offer it will take on a certain order and pay the carrier to deliver the shipment. Oftentimes, the decision to accept or reject a carrier's offer is very difficult. When trying to maximize margin, we must be wary of which offer we accept because there may be a future offer that is better than the one we choose. Further complicating the problem at hand, STLs are also important to consider, as other orders may come in that can be pooled after an offer has already been accepted for the original, individual order.

These opportunities to optimize the offer acceptance method are vital for increasing margin and reducing cost. By using information about orders, as well as historical (prior) data on orders and the offers they received, we can build a model that follows previous work in the optimal stopping problem. Our model takes into account the context of how offers are received, as well as using order information to make a prediction on how much an offer should cost.

1.2 Review of Prior Work

The issue of optimal stopping frequently appears regardless of industry or context, giving our problem an expansive set of previous knowledge on the topic. For example, a common scenario in the optimal stopping question is known as the secretary problem. This scenario involves an administrator looking to hire the best secretary out of n applicants for the position. The decision on an applicant must be made immediately after an interview. The administrator can gather information through the candidates that have been interviewed already, however they know nothing about the future, not-yet "seen" applicants. The problem is solved through the 1/e stopping rule, where e is the base of natural logarithms. Applying this rule to the scenario above, if we have 10 randomly selected applicants, we'd interview the first 10/e applicants and record the best applicant. We'd then interview the remaining applicants stopping at the first applicant who beats our recorded "best applicant". If we never find a better applicant and are interviewing the last applicant, we must select the final one. In applying this problem to freight brokering, our "applicants" are the offers we receive from carriers to ship an order. The caveat is that we do not know how many offers we will receive on one order, making it difficult to apply the secretary approach without first building a model to predict the amount of offers we may receive. We want to further improve this method by introducing another prior through using logistic regression in predicting a "good cost" basis for an offer, so that we do not accidentally miss a "good" offer when iterating through the secretary method. We also do not have a decisive way that offer will be accepted, as an offer may be pooled with a future order in the way of STLs. This further complicates our problem and the question remains, what is the optimal way to accept offers so that we can increase our margins?

1.3 Data Description

The full dataset includes 2 tables which contain: A) the orders, and descriptive information about it (Order Time, Pickup Deadline, accommodating conditions for its mode of delivery such as transport mode, refrigeration, and hazardness, etc); B) The offers by carriers to deliver said orders - this would be a many-to-one relationship, with the reference number column (assuming it's a singleton list) being the foreign key. The offers table includes mostly information such as the rate of the offer, whether it is pooled or not, whether it was selected, and whether it was uncovered. We also created a supplemental dataset that maps 3 digit zip code identifiers to latitude and longitude coordinates.

Table 1: Orders Data Dictionary

Column Name	Description
REFERENCE_NUMBER	Unique ID for the order
ORDER_DATETIME_PST	Date and time of order in Pacific standard time
PICKUP_DEADLINE_PST	Date and time order must be picked up from origination in Pacific standard time
DELIVERY_TIME_CONSTRAINT	Type of delivery time scheduling constraint
ORIGIN_3DIGIT_ZIP	The first three digits of the origination location ZIP Code
DESTINATION_3DIGIT_ZIP	The first three digits of the destination location ZIP Code
APPROXIMATE_DRIVING_ROUTE_MILEAGE	Approximate number of driving miles from origination to destination
PALLETIZED_LINEAR_FEET	The length and weight of the shipment converted to the percent amount of the truck filled
FD_ENABLED	Customer paid for upgraded service with delivery deadline and no transfer of truck (hub and spoke system not allowed)
EXCLUSIVE_USE_REQUESTED	Customer paid for shipment to be delivered on its own truck (cannot be pooled)
HAZARDOUS	The shipment is hazardous material (cannot be pooled)
REEFER_ALLOWED	The shipment can go on a refrigeration truck
STRAIGHT_TRUCK_ALLOWED	The shipment can go on a straight truck
LOAD_BAR_COUNT	The number of load bars required by the load
LOAD_TO_RIDE_REQUESTED	Delivery service without hub stops
ESTIMATED_COST_AT_ORDER	Flock Freight's estimated cost to fulfill the order (estimated at the time of order)
TRANSPORT_MODE	The type of shipment (FTL, LTL, PTL)

Table 2: Offers Data Dictionary

Column Name	Description
CARRIER_ID	Unique ID for the carrier providing the offer
REFERENCE_NUMBER	Set of order reference numbers the offer would deliver (more than one for a pool)
CREATED_ON_HQ	Date and time offer was submitted in Pacific standard time
RATE_USD	Amount carrier will be paid if offer is accepted
OFFER_TYPE	"pool" for two orders pooled together, or "quote" for one order
SELF_SERVE	Boolean field designating carrier made offer through the app without representative intervention
IS_OFFER_APPROVED	Boolean field designating if Flock Freight approved carrier's offer (carrier must still confirm contract)
AUTOMATICALLY_APPROVED	Boolean field designating if Flock Freight approval was done without representative intervention
MANUALLY_APPROVED	Boolean field designating if Flock Freight approval was done with representative intervention
WAS_EVER_UNCOVERED	Boolean field designating if agreed contract to deliver load was ever broken (e.g. carrier truck broke down)
COVERING_OFFER	Boolean field designating Flock Freight and carrier agreed contract together to deliver load
LOAD_DELIVERED_FROM_OFFER	Boolean field designating this offer was the offer to deliver load
RECOMMENDED_LOAD	Boolean field designating the load (set of order references numbers) was sent to the carrier as a recommended load

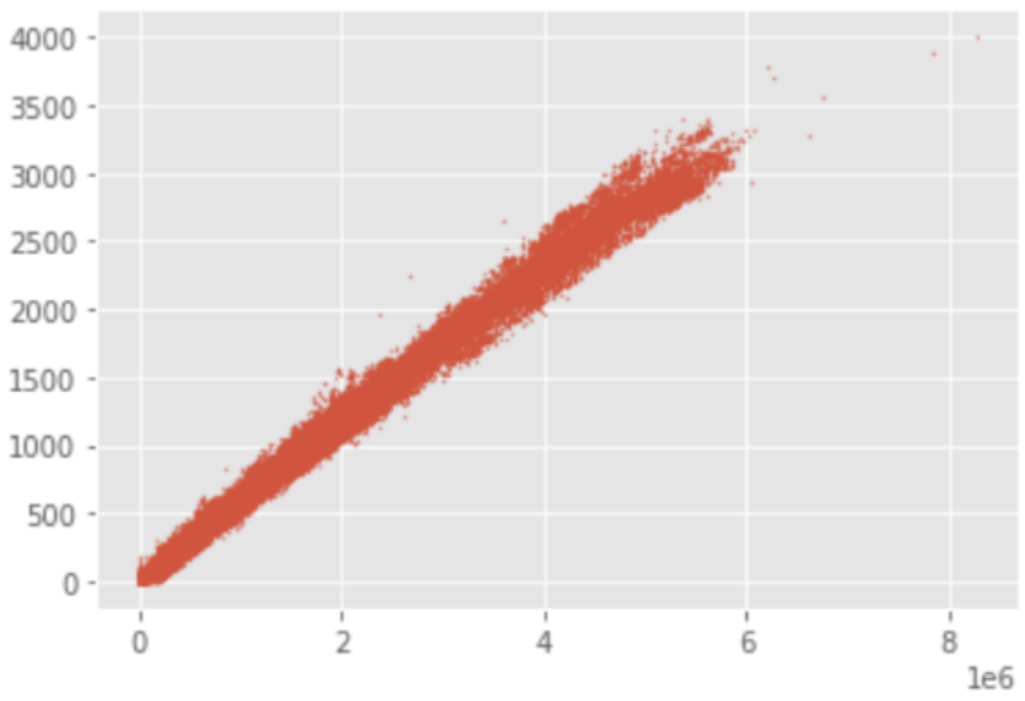
2 Methods

2.1 Exploratory Data Analysis

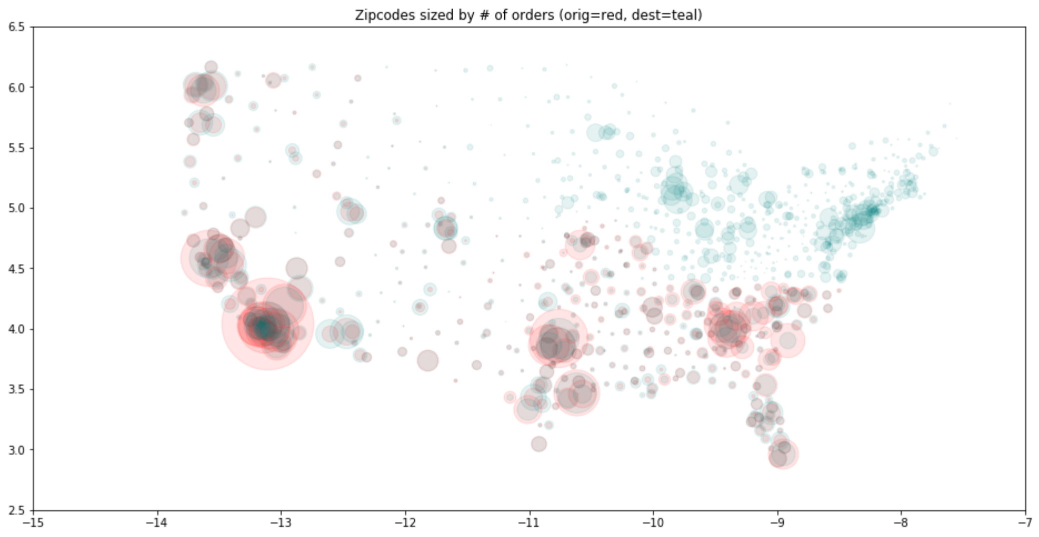
2.1.1 Geospatial Analysis

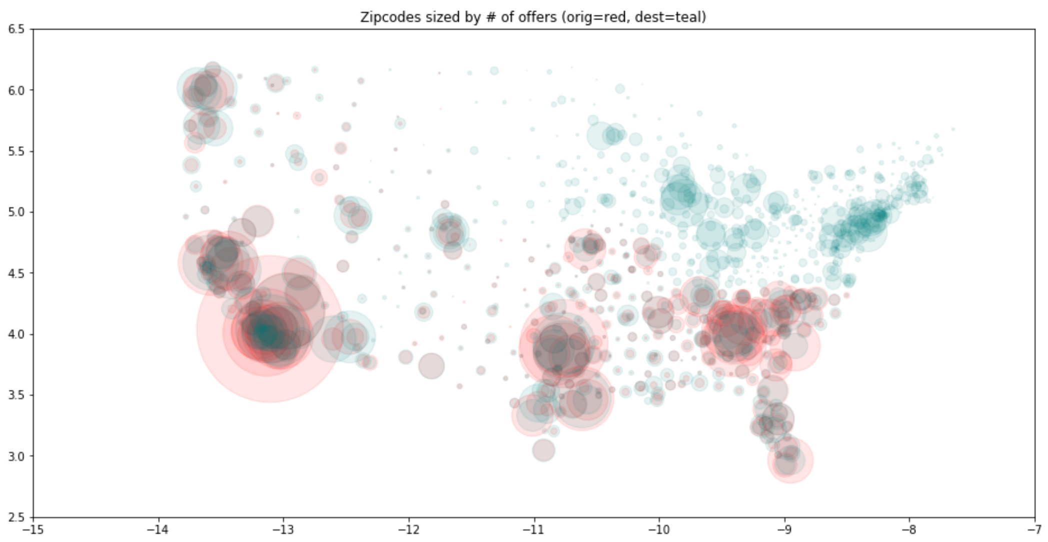
An automated python script first uses BeautifulSoup to web scrape from a government census website to download the latest 2020 census shapefile for zip code tabulation areas, (which is a geospatial data frame that maps each zip code to geographic polygonal areas) for data preparation with geopandas; Geopandas will be used to reproject the shapefile to pseudo-mercator. The original data only provides the first 3 (zero padded) digits identifier of the zip codes, whereas the shapefile zip codes are 5 digits; for this reason, the 5-digit zip code column had its 2 last trailing digits dropped (making it usable as a one-to-one foreign key), and Dissolve (a spatial group-by operation in Geopandas) was used to group up the zip code areas into unified polygons. The centroids of each zip code tabulation area will represent the x, y coordinates.

To verify this geospatial data preparation actually produced accurate results, we calculated the euclidean distances between the new x/y coordinates of the destinations and origins, and plotted them against the pre-existing column “Approximate Driving Route Mileage”; the resulting scatterplot is a nearly perfect straight diagonal line, attesting that it is reliable.

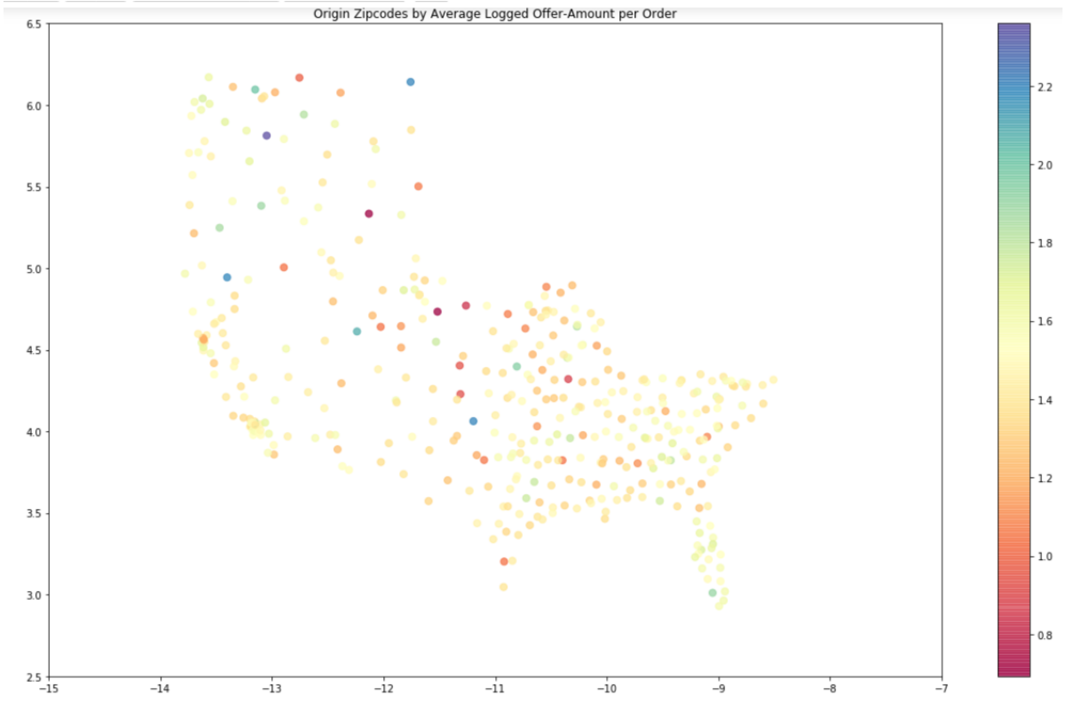


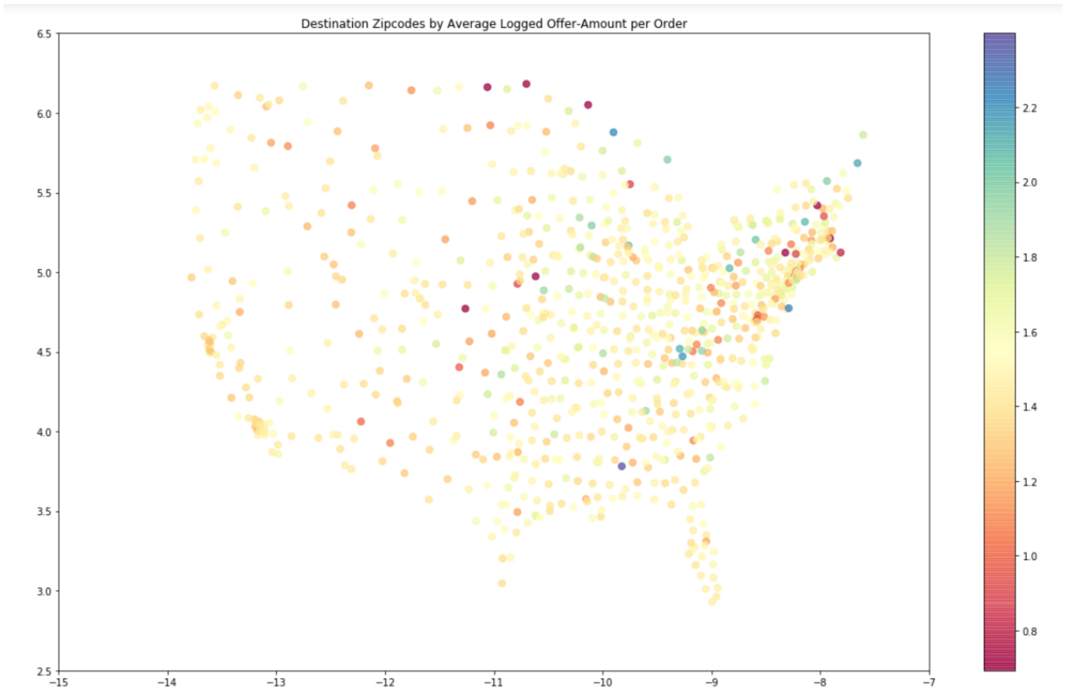
By plotting the orders in terms of their origin and destination zip codes, most scatter plots only show a population bias. The eastern half of the US tends to be more heavily populated than the western half. This makes sense since population centers tend to be logistic hubs, but is otherwise not useful.





The 2 following plots show a scatterplot of zipcode nodes colored according to the logged average amount of offers they see per order, (done for origin zipcodes and destination zipcodes).



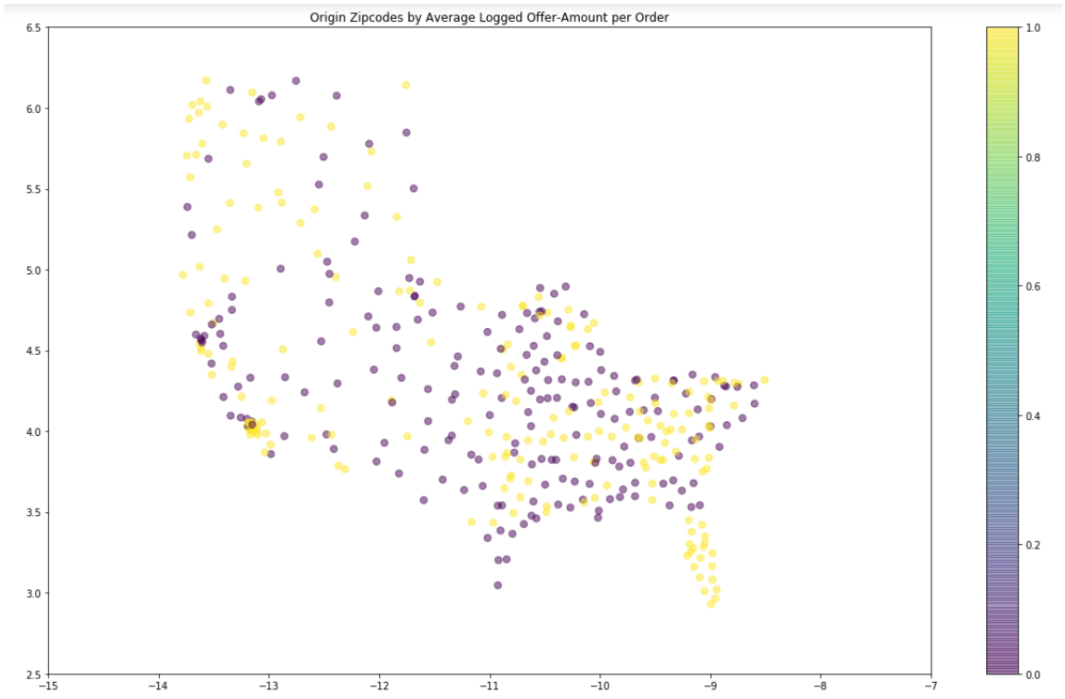


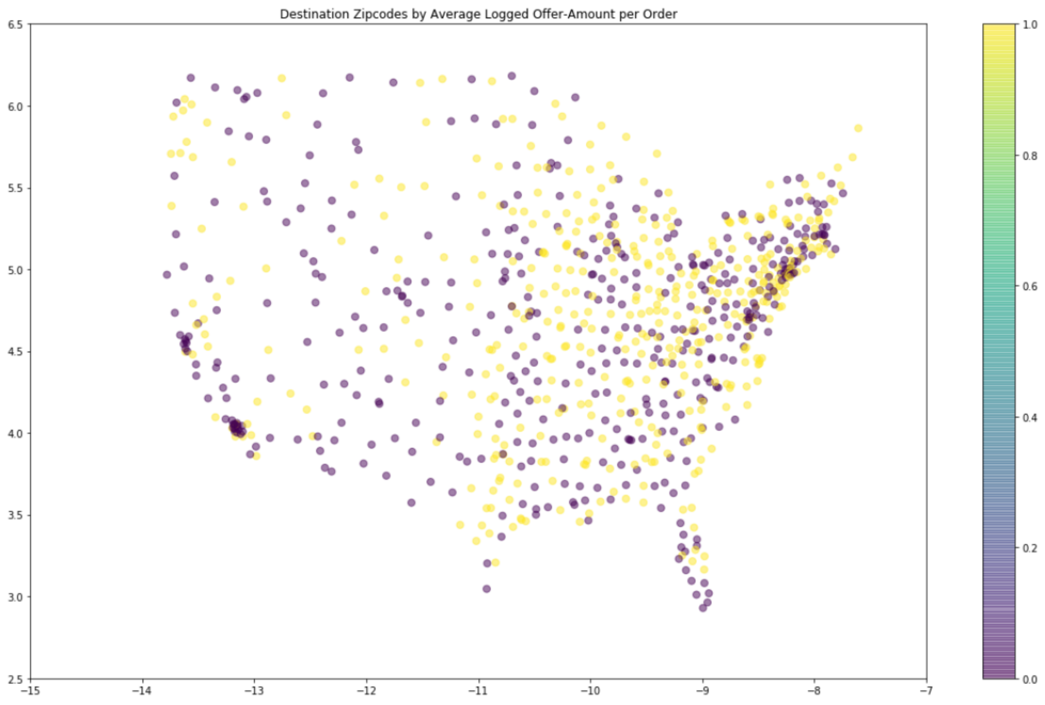
The next 2 plots are binarized versions (above or below the median) of the earlier 2, to make contrast easier to see.

* Clearly, the eastern half of the US and the urban west coast has higher population density, and there is an equal balance of both high and low offer amounts. So utilizing geographic features for these regions for offer amount prediction might not be helpful.

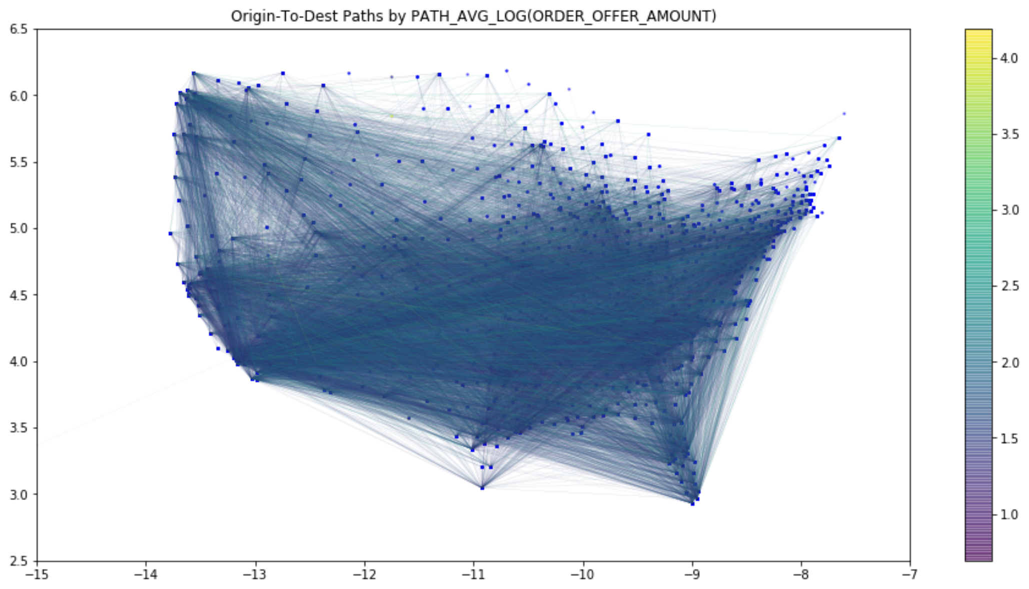
* Orders with an origin zipcode in the Southwest seem to have seem to have a mostly low (below-average) offer amount.

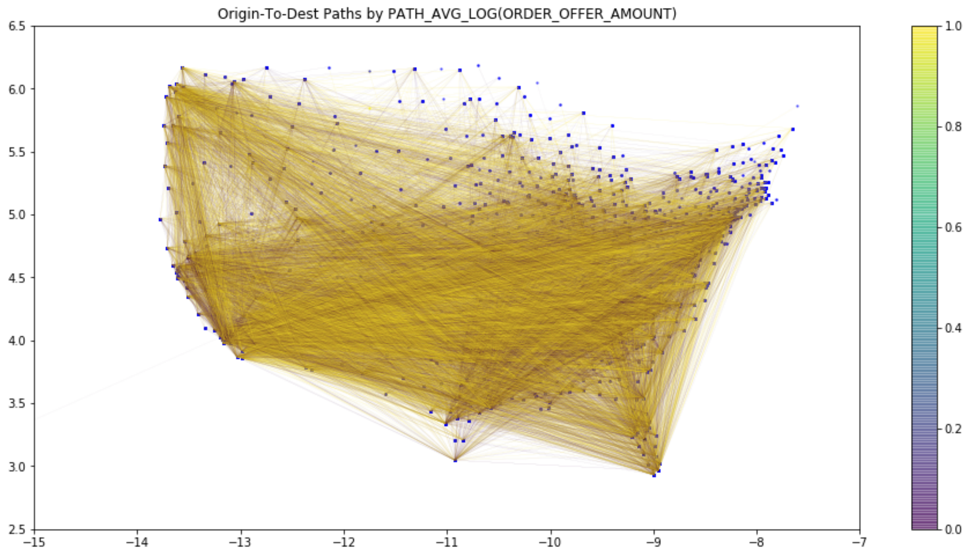
* Orders with an origin zipcode in Florida seem to have a confidently high (above-average) offer amount.



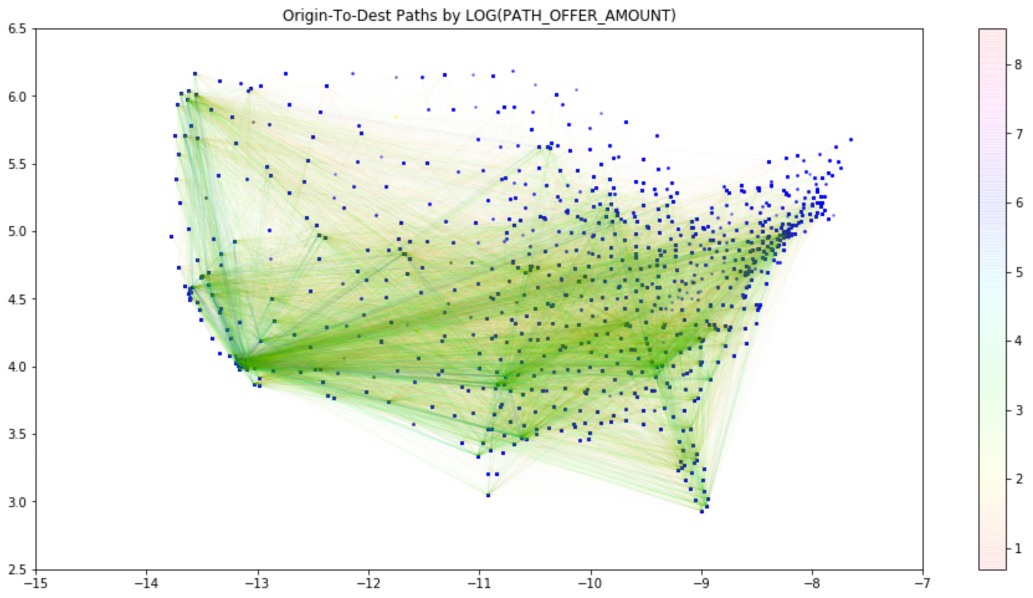


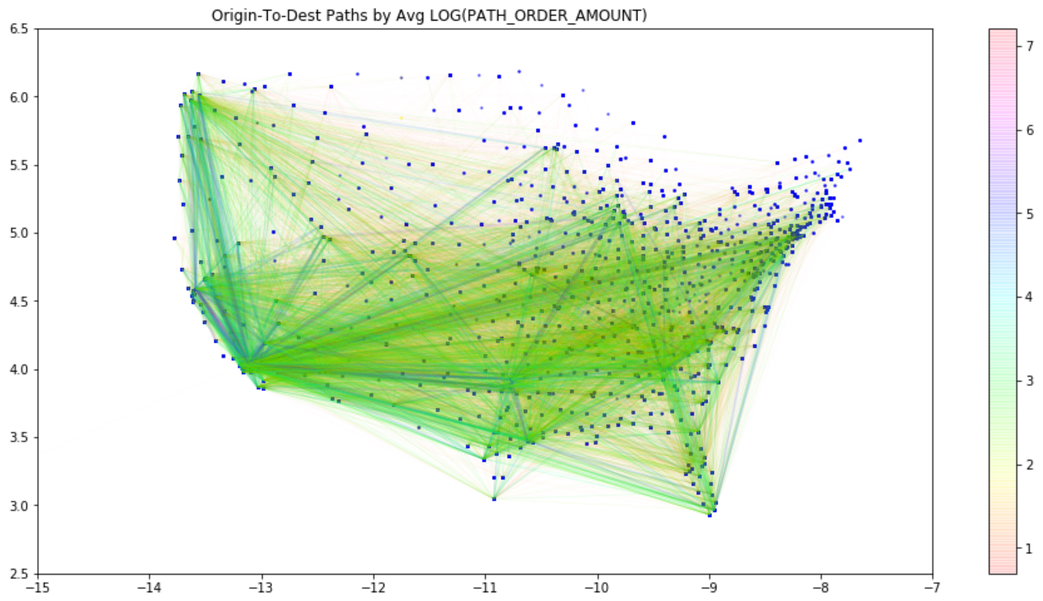
The 2 following plots also try to visualize logged average amount of offers per order, but now it's for shipping routes instead of Nodes. The plot is too difficult to see any differences, so the 2nd plot binarizes the values (below or above the median).



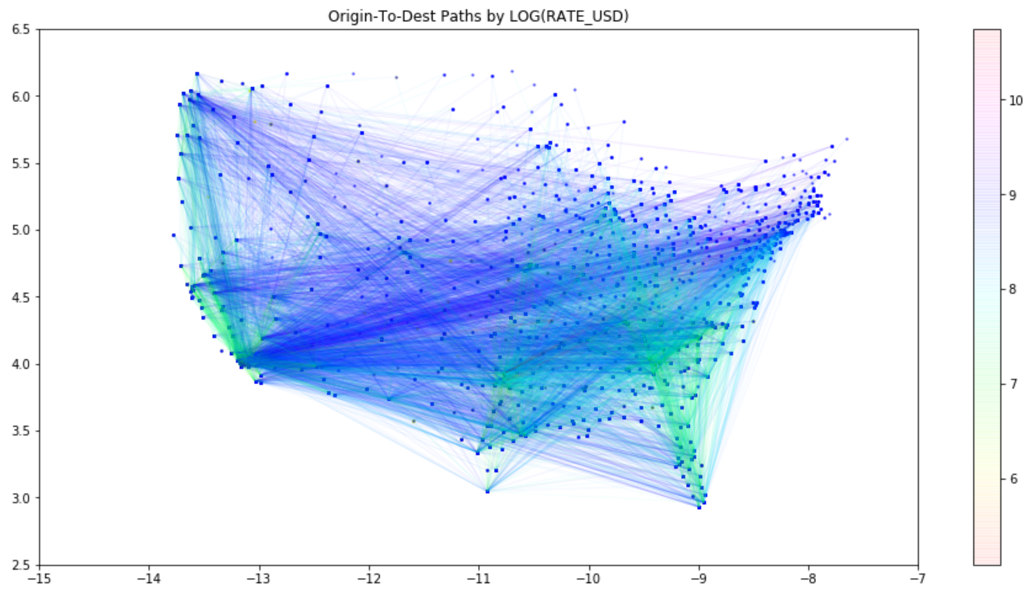


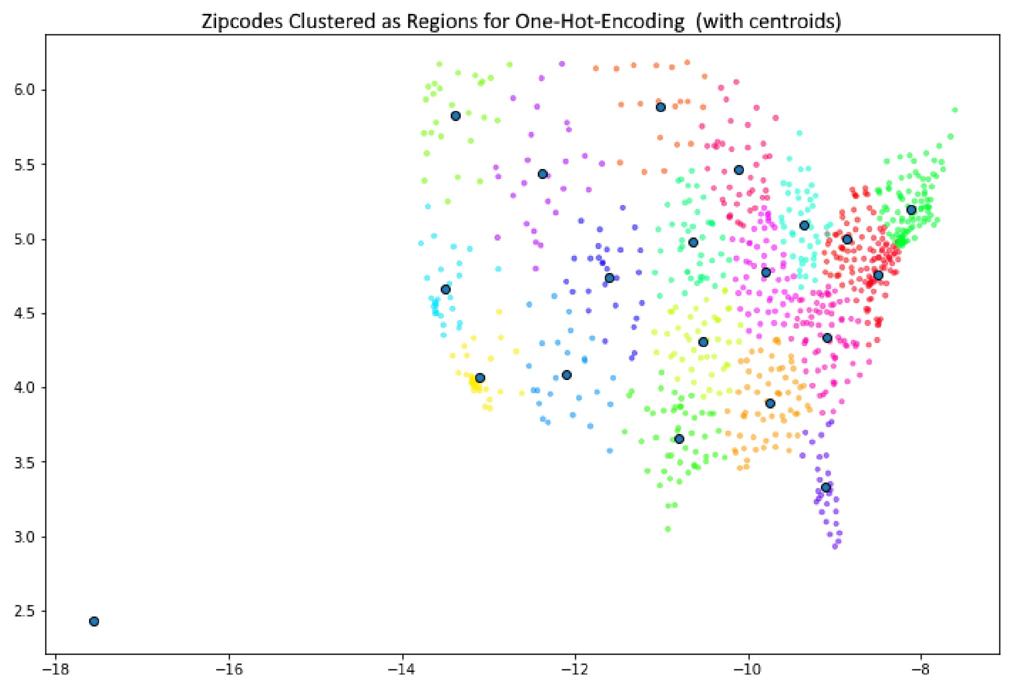
The network of connections between the origins and destinations for orders is also plotted; Side note: some zip code nodes seem fully isolated, but they do in fact have a link ; the links are just at a very low opacity (0.1). What this can mean: Paths with origin / destination zip code nodes that are connected to the cities, will have more orders. In addition, the area in what seems to be between the Upper South and Mid Atlantic have noticeable activity with orders. Shipping between Los Angeles and New York is also very heavy. This could be useful in our prediction model for number of offers.





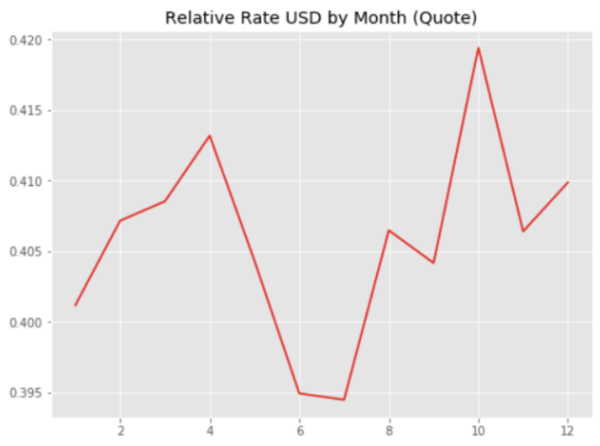
This last figure (Shipping Paths by Log(Rate USD)) shows that several shipping corridors are cheaper than others. These include most deliveries entering and exiting Florida (From Chicago to Florida? And Dallas to Chicago?), and deliveries along the west coast between Southern California to Seattle.

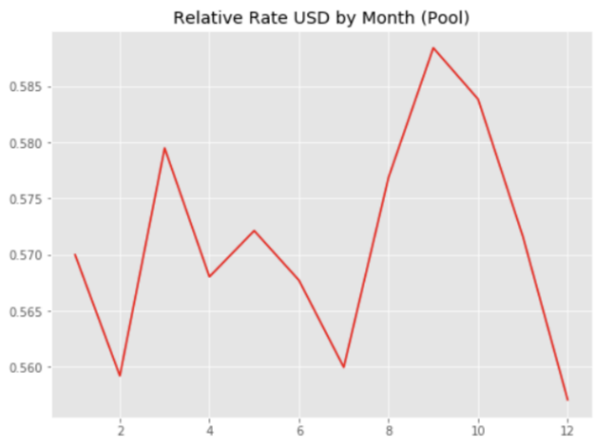




We also tried to use unsupervised ML, specifically grouping the zipcodes into clusters. This was done by applying K-means clustering (N=20) onto 3 columns: the zipcode's X and Y coordinates, and a 3rd dimension to symbolize density (We went with the number of offers seen per zipcode), allowing the clustering procedure to extract metropolitan areas. This is going to be used as a way to one-hot-encode the zipcodes as generalized metropolitan clusters, 20 columns rather than hundreds of zipcodes.

2.1.2 Time Data Analysis



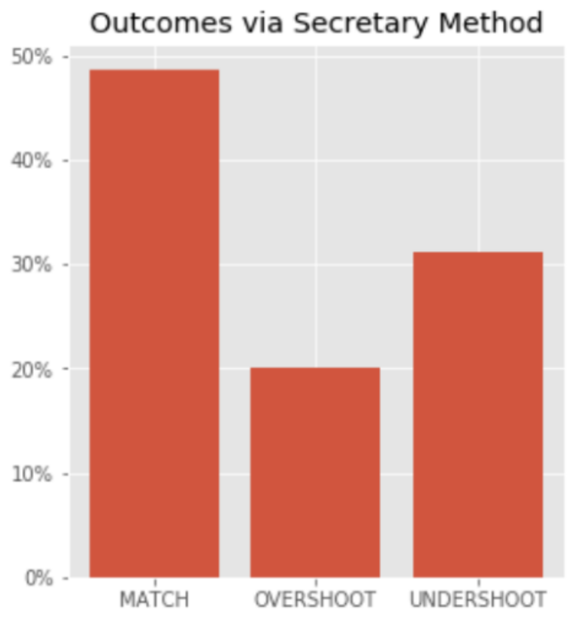


When comparing the metrics of different business days, it was found that Thursday had unusual statistical significance in having the cheapest of rates (relative to the other offers for an order). In other words, we logged the RATE USD column of the joined tables, then z-scored it (not in terms of the whole table, but to each group of offers per order), then binarized it so that being >= 0 means above-average (since z-scoring re-centers the mean at 0), and in doing so, found that Thursday has the lowest percentage of above-average offer rates compared to other business days; even with further hypothesis testing (Chi-Square, T-Testing), Thursday continues to pop up as statistically significant. If we put the 2 last columns of the table below through a chi-square hypothesis test, the resulting p-value is 0.0010375. This could be a promising area of potential use for future modeling.

Day	% of Lognorm(Rate)>0	# of Lognorm(Rate)>0	# of Lognorm(Rate)<=0
Friday	0.451306	65670	79841
Monday	0.454253	76642	92079
Thursday	0.448788	74901	91995
Tuesday	0.453000	77902	94067
Wednesday	0.455491	76220	91116

2.2 The Secretary Method

Secretary Method Outcomes - The following analysis shows that using the raw secretary approach by itself is still a very reliable method: There should be 3 types of outcomes of the secretary method: 1) Match - the method happens to land exactly on the best offer; 2) undershoot - the method stops on an offer better than anything before the 1/e mark, but is not in fact the best one yet; 3) Overshooting - the method had long passed the best offer already- meaning the best offer was actually before the 1/e mark, and the method has no choice but to stop on the last offer, regardless if that last offer is good or bad; Overshooting is the worst outcome; As seen by this barplot for the secretary method, there's a 0.8 chance that either a perfect match or undershooting occurs; While undershooting is not as good as a match, it's a far better position to be in than overshooting - so this is still a good deal.



3 Model

Our model comprises of 3 sub-models. They are prediction models that take in the data of an order, and outputs 3 columns representing the average rate of an order, the standard deviation of said average rate of an order, and the number of offers the order gets. These 3 columns are the features of another final model that classifies whether or not an incoming offer for an order is acceptable or not. For the time being, this unifying model will be a simple if-statement, which checks if an incoming offer's rate is lower than that order's predicted average rate minus the predicted standard deviation of said order's rates.

* If true, that means the current offer is extremely cheap, and probably cheaper than most of the future incoming offers, meaning it should be taken.

* If false, that means we should reject it and wait to see the next offer.

* This model setup also has the benefit of letting you adjust the range of tolerance for the threshold (without needing to retrain the model) by multiplying the standard deviation with some positive constant.

* The column for the predicted amount of offers allows more carefulness when rejecting offer: if the amount is high, we are free to gamble with rejecting incoming offers until we get one that is below the threshold.

The Offer Amount sub-model uses lead-time for sample-weight adjustment during the training.

The Rate Average Model uses linear regression and is very accurate at 85% to 90%. The Rate Standard Deviation model performed poorly when any sort of regression method was used, so the model task was re-simplified as ordinal classification for the time being: the standard deviation is now split in tiers, and the tier cut-off values serve as the "standard deviations".

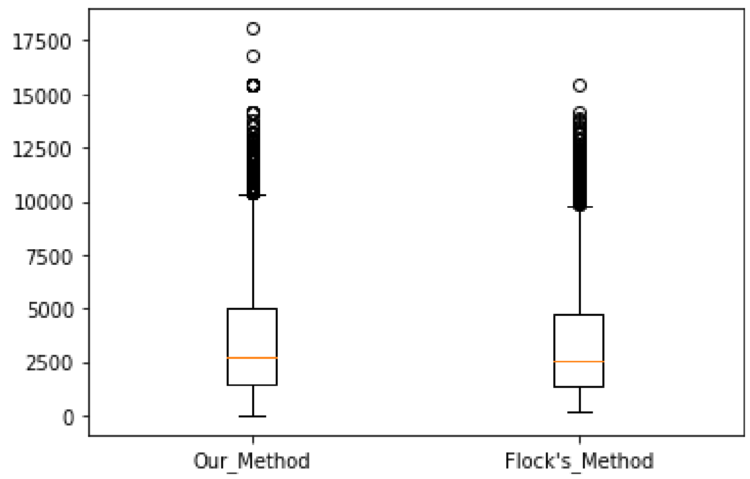
For example, if the value 0.3 splits the observed distribution standard deviation in the training set into 50%-50% (i.e. median), then stdev's in the range 0 <= x <= 0.3 are labeled Low and represented by 0. And those 0.3 <= x <= inf are labeled High and represented by 0.3.

Both the Rate Average Model and Rate Standard Deviation model mostly uses the same features, with closely identical input data frames. The features are only allowed to come from the orders dataset.

4 Results

Our resulting offer acceptance method consists of a baseline secretary method that utilizes a trained linear regression model that estimates the number of offers expected and a prior that uses a linear regression model that predicts the estimated cost. The expected number of offers is modeled using the following features: 'FD_ENABLED', 'EXCLUSIVE_USE_REQUESTED',
'HAZARDOUS', 'REEFER_ALLOWED', 'STRAIGHT_TRUCK_ALLOWED',
'LOAD_TO_RIDE_REQUESTED', 'APPROXIMATE_DRIVING_ROUTE_MILEAGE',
'PALLETIZED_LINEAR_FEET', 'SECONDS_BETWEEN_ORDER_AND_DEADLINE',
'LOAD_BAR_COUNT', and 'ESTIMATED_COST_AT_ORDER' while the estimated cost is trained using two features: SECONDS_BETWEEN_ORDER_AND_DEADLINE' and 'ESTIMATED_COST_AT_ORDER'. The seconds between order and deadline is a new column generated by taking the difference between the 'PICKUP_DEADLINE_PST' and the

'ORDER_DATETIME_PST' column. Our prior works by comparing the estimated cost and the rate given for the first n/e offers and if any of those offers have a rate less than the estimated cost to ship then we automatically accept that offer instead of moving on with the secretary method. We evaluated our offer acceptance method by comparing the average rate of the offers we chose in the test dataset, to the average rate of the historically accepted offers and found that the historically accepted rates were around \$150 cheaper. The distribution of our accepted offers and the historically accepted offers can be seen below.



5 Discussion

While we wanted to provide a model that accepts lower-cost offers than the historically accepted offers by Flock Freight, there is a lot to learn from this information. The constraints of the secretary method do not entirely match up with the real-world problem we attempted to solve. For example, the fact that in the environment we created we could not go back on a previously "rejected" offer is not equivalent to the choices that can be made in the real-world environment. We can look at multiple offers that are on the table and see which of them are better than others. The introduction of STLs also plays a key role in offer acceptance, which we do not account for in the secretary method. For example, a type of prorated offer can be introduced that increases when orders are attached to others that can be pooled together. In our future work modeling offer acceptance, we will attempt to incorporate the possibility of pooling. Our results can confirm that the problem at hand is not fully generalizable to previous methods of optimal stopping. We noticed that our model for estimating cost can be thoroughly improved by incorporating geographical features developed in EDA. This goes the same for our prediction on the number of offers we receive. When looking at the baseline secretary approach from our EDA, we can also pose further questions such as: If we undershoot on many orders, should we reevaluate how we predict the number of offers we get, in order to still observe some future offers we negated previously by choosing one too early? While we answered these types of questions when building our model, our answers proved only that there is more that can be done through changing the framing of our problem, to eliminate some of the constraints in the secretary method and better fit the context of freight brokering. The way we choose our offer is limited and can be improved through a better prediction of a “good offer” basis, as well as a better prediction of the number of offers we will receive on an order that is based off of similar orders in size and geographic location.

6 References

Thomas S. Ferguson. "Who Solved the Secretary Problem?." Statist. Sci. 4 (3) 282 - 289, August, 1989. https://doi.org/10.1214/ss/1177012493

Hill, Theodore. "Knowing When To Stop" American Scientist, 6 February, 2017 https:www.americanscientist.org/article/knowing-when-to-stop

https://www.census.gov/geographies/mapping-files/ time-//series/geo/cartographic-boundary.2020.html

7 Appendix

Proposal Background Flock Freight acts as an intermediary in the shipping industry. Shipping carriers and shippers trust Flock Freight as a "freight broker", providing a way to easily ship freight across the country, specializing in connecting multiple shippers' loads with a carrier in what is known as a "Shared-Truckload" (STL). Flock Freight receives orders from shippers, which carriers will bid on to take the load. Flock Freight then decides which offer it will take on a certain order and pay the carrier to deliver the shipment. Oftentimes, the decision to accept or reject a carrier's offer is very difficult. When trying to maximize margin, we must be wary of which offer we accept because there may be a future offer that is better than the one we choose. Broad Problem Statement In this project, the core issue is being able to optimize for the acceptance and reneging of delivery offers by carriers for Flock Freight's orders. We'd need to be able to determine that a given offer will be the best one seen compared to all future offers (basically gambling against unseen opportunities). On the other hand, if we know a current offer isn't good, we'd also need to gamble the risk on reneging it for a better future offer, should one even arrive. Further complicating the issue is that being indecisive for too long will make carriers uninterested and look for a different freight broker, thus losing clients. We propose a model (or group of models) that should ultimately assist (to some extent) in a general task where given an order and an offer for it, should be able to determine whether that offer is an optimal selection, doing so in a fast, scalable way within milliseconds. Narrowed Problem Statement Further complicating the problem at hand, STLs are also important to consider, as other orders may come in that can be pooled after an offer has already been accepted for the original, individual order. These opportunities to optimize the offer acceptance method are vital for increasing margin and reducing cost. By using information about orders, as well as historical (prior) data on orders and the offers they received, we can build a model that follows previous work in the optimal stopping problem. Our model takes into account the context of how offers are received, as well as using order information to make a prediction on how much an offer should cost. It is thus essential to provide a model and process that is reliable and scalable to Flock Freight's needs. This means that we must build off of the introduced secretary method for optimal stopping in order to include models for predicting cost, quality of offers, and number of offers. By implementing models for estimating all of these we can improve on the estimated cost already evaluated and the historically chosen offers. With our final stopping method concluded we will be able to write a paper describing how we optimize our baseline model to include more effective prediction methods far beyond what our original model/method accomplishes. Finally, we hope to develop an optimal stopping method that performs better than the historically chosen offers given to us in the data from quarter one. Our new model should contain the following set of aspects and considerations that were duly noted over the course of Quarter 1 by our teammates: The Secretary Method should still regard undershooting as an adequate solution, even if it is not as good as a perfect match; overshooting is a scenario that must be avoided. Better Use of the estimated cost to find good offers potentially utilizing a quantile regression model to evaluate a threshold to identify the quality of each offer Better training of both estimated cost and estimated number of offers model by utilizing new features and testing out models other than logistic regression to see what performs best Employment of the potentially useful priors found and mentioned in the report paper, for example: the over-representation and dense coverage of orders with paths linked to zip codes in the Upper South / Mid Atlantic Region. This is potentially useful for the offer amount prediction model Employment of Thursday as a model feature, due to its unusually statistically significant prominence in the time related exploratory data analysis, mainly for tasks related to rate comparison. Data The data for this project will be based on the project data already used in the first quarter. This data contains all the sufficient information necessary for our models to adequately deliver results, and is adequate in scope enough - if not unavoidably inextensible any further due to Flock Freight's legal capacity to share company data. Output The primary output of this project will be prioritized as a research paper detailing comparison of model metrics and other model explainability.

8 Contributions

Radu Manea - Abstract, Introduction, Results, and Discussion, Report Configuration, Model Building
Benson Duong - Abstract, Introduction, Methods, Report Configuration, Model Building and EDA, Github Repo Configuration
Keagan Benson - Model Building, EDA, and Github Repo Configuration
Nima Yazdani - Introduction, Results, Model Building and EDA