**Course:** EE590 Project Report

**Name:** Joseph Ruiz

**Student ID**: 7172842690

**Email**: [jdruiz@usc.edu](mailto:jdruiz@usc.edu)

**Term**: Spring 2023

**Project Title**: Equal Item Substitution Recovery

**Date**: 2023-04-11

**Executive Summary**

With the advent of the internet and increased online and direct to consumer shopping, more and more items have become available for purchase that are then shipped directly to the consumer regardless of the type of product. This has led to a rise in services and companies to manage this last-mile delivery; whether it’s the last-mile delivery for an order of jeans from an online boutique, a TV from a larger appliance store, or a bundle of goods from a local store. This has also given individuals more methods to search for items, select a method for how they’d like to receive those items, and select a service that delivers to them and/or compiles the items so they can simply pick them up. All of these methods aim to reduce the friction between the buyer and seller of goods no matter where they are located.

However, because of these ever increasing nodes of buying and selling, there is increased risk of inefficiencies developing along or within a delivery path. The inefficiencies can be the total miles an item has traveled, total miles the last-mile driver has traveled, the cost of the product to the company, the price of the product to the buyer, and additional gas emissions due to an increased number of miles driven. As consumers turn to mobile and web apps to purchase goods from a local store, there is an opportunity to consolidate their orders more efficiently to reduce the number of stores visited, number of round-trips to a store, and the number of miles driven.

My current company (hereinafter referred to as “Startup”) has been purchased by a larger retailer (hereinafter referred to as “Parent Company”). Startup currently completes last-mile deliveries for a large number of other retailers including the Parent Company. The Parent Company and the other retailers more often than not have the same exact items available for purchase at various prices and costs. However, the end prices for the consumer are often higher if purchasing from other retailers due to markups required to properly manage the additional catalogs. With efficiency in mind and lower prices for consumers already possible, there is a real opportunity for substituting an item from another retailer for the same exact item from Parent Company instead. If the consumer can enjoy lower prices and Startup can achieve greater efficiency for the delivery service as a whole, then making the switch automatically on behalf of the consumer, with no degradation in item quality, would be a large gain for Parent Company, Startup, and the consumers.

The focal points to measure in order to achieve the above goal are reduction on total miles from the current state, a reduction in cost and a reduction in price, and an offset for cases when the Parent Company does not readily appear to be able to fulfill an entire order’s item list. I compared the items between Parent Company that are available and were available at the time of purchase to all orders at Startup from January 2022 to March 2023 to find when Parent Company had lower costs and/or lower prices for the same exact item that the order’s original retailer also had in stock. In this general comparison over that time period of simple item matching, there was a $36m cost savings and $84.5m price savings opportunity. However, other factors in the order must be taken into account such as understanding if a single order’s items can all be substituted, only a certain percentage of items are covered and can be reasonably substituted, or the order is not eligible if/when a certain threshold is not met. When looking only at orders that have 100% coverage by Parent Company, the cost savings and price savings opportunities become $425k and $736k respectively; still a large amount that can be captured.

This does not mean that orders with lower “substitution eligibility” cannot be considered. There could be other ways in which Startup or Parent Company can make the individual whole due to the savings for the consumer and/or shorter delivery times. There can also be additional efficiencies still for the driver if they can drive shorter distances, reduce emissions, and get larger tips for the better deliveries.

If we reduce the eligibility requirements for an order from 100% coverage to at least 80% coverage, the cost and savings opportunities become $1.3m and $2m respectively, a nearly 300% increase from the 100% coverage requirement.

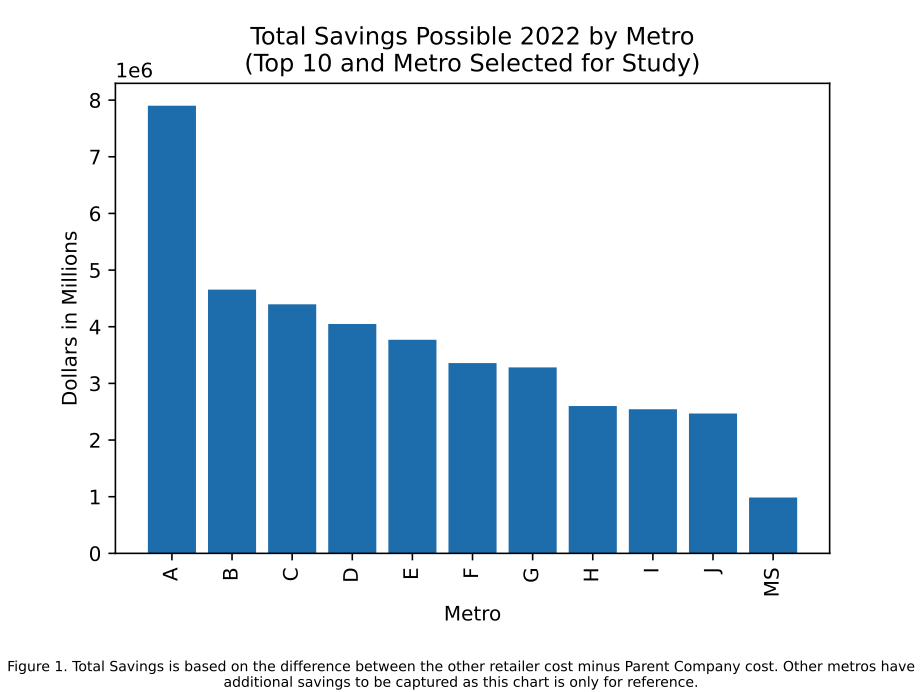
The problem then becomes a classic Traveling Salesman problem [1]. Each retailer location and customer location would become individual nodes and the paths between each would have weights that are aggregations of miles, eligibility, cost, and price. Then the overall measure of success would be comparing the minimized paths of an optimized current state to the minimized paths of the optimized hypothetical Parent Company substitutions state. The current model I created does not rank these weights in terms of importance as each customer may have different preferences. One may prioritize quicker delivery times while another may desire lower prices. However, the model can easily be altered to include such preferences if further consumer studies are done.

In order to complete this comparison with the computing resources at my disposal, I conducted the research on a single metro area, hereinafter referred to as Metro A. Overall, when comparing the optimized current pathing with the optimized hypothetical Parent Company pathing the average miles traveled is reduced by 3%, the cost is reduced by 1%, and the price is reduced by 4.5%. This amounts to $1k in cost savings and $1.5k in price savings for a total of $2.5k for the single day. This is a general analysis for a single metro for a single day with a few variances throughout the year. Other days and other metros achieve even greater results which are explored a bit further later while the bulk of the analysis is on Metro A.

Overall, there is ample evidence that doing an “item substitution”, which is truly a “retailer substitution, for an order can achieve greater efficiencies throughout the organization and have the benefits spread to drivers, consumers, and the Parent Company. If these simple small efficiencies can be had throughout the year, that amounts to about $547.5k in total savings for the year in just a single metro. This topic should be explored further outside of this analysis.

**Introduction**

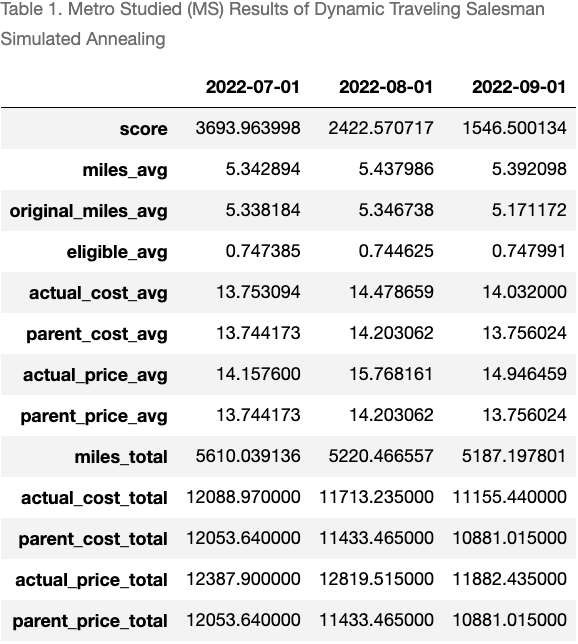
With ever increasing needs for more efficient last-mile delivery, more and more services and companies have been established to handle these increasing needs. It has led to a variety of companies that handle traditional long-haul deliveries, Less-Than Full Load deliveries, and shorter last-mile deliveries directly to the consumer. Consumers desire more choice in the market around how they shop and receive their items in addition to greater item choice and availability. With increased demand for goods as quickly as possible, this can lead to greater inefficiencies. With each new last-mile delivery company, new store, and new customer, the number of participants (nodes) increases greatly and can lead to inefficient trips and paths occurring between end consumers and the original store. These inefficiencies include greater miles traveled, higher costs for products, higher prices for products, greater emissions by various vehicles, more packaging, and less job satisfaction for drivers. The issue becomes managing and forecasting the best paths the drivers can take in an area to get the same items to the consumer without any loss at all in quality. The question becomes, “how can we get the item the consumer wants at the lowest possible cost, price, miles, and emissions?”

My current company, hereinafter referred to as Startup, was purchased by a larger retailer, hereinafter referred to as Parent Company. Parent Company more often than not has the same exact items available for purchase as the other retailers. Due to the Startup being a subsidiary of Parent Company, the costs and prices for items purchased from Parent Company are often lower than that of the other retailers. The markups placed on other retailers’ items reflect the additional costs incurred to maintain other retailers varying catalogs and monitoring their stores for availability of items and store hours just as much as if Startup worked directly for the other retailers or were part of other retailers. Therefore, there is greater need and opportunity for gains in efficiency by substituting an order or items in an order for the same exact item from Parent Company instead of the order’s original other retailer. The opportunity for cost savings and price savings, which add up to total savings, can be seen in Figure 1 below.

In addition to cost and price reduction, the effort to have less miles driven and therefore lower emissions must also be considered. Based on a report from the World Economic Forum in January 2019, by 2030 delivery traffic will increase by 32% and emissions by 32% from those vehicles [2]. This shows a real need for more creative methods to determine optimal paths for deliveries to get to consumers. Back and forth trips by drivers across their metros cannot be maintained if the delivery service industry wishes to help tackle climate change.

There are a variety of measures that can be taken into account when trying to optimize the travel routes for these last mile deliveries. These can include miles driven, product costs, product prices, time spent delivering, time window of actual delivery drop off, effective take-home pay of driver, driver ratings, route safety, etc. The main focus of this paper is miles driven, product costs, and product prices. The additional metric taken into account is also “order eligibility” which intends to be a percentage of items in the order that can be fulfilled by Parent Company. These measures become the “weights” of each bath in a Dynamic Traveling Salesman Problem [3]. We can attempt to minimize miles driven, product cost, product price, and order **in**eligibility (1 - order eligibility) when connecting each store location node and customer location node in the classic Traveling Salesman Problem style. Each path then takes on the weights for the miles between each store and customer and the potential costs and prices their orders would be at each location. Once the model has found the optimized route, it can be compared to what occurred in reality and/or to an additional hypothetical optimized path for reality where the original store locations are kept in place.

I conducted the algorithm on the specific Metro Studied and output results for 3 different single days in 2022. Below are the basic results of that analysis in Table 1.



Here we can see how the model scored each day. This score is relatively meaningless as it is only the calculation used to minimize the pathing taken by the algorithm. The miles\_avg value is the average miles that the algorithm found when traveling to a Parent Company location and the original\_miles\_avg is the optimized hypothetical path of the original store locations. Eligible\_avg is the average percentage of items per order that can be fulfilled by Parent Company, actual\_cost\_avg represents the cost that actually occurred at the original store locations, and parent\_cost\_avg is the cost at Parent Company locations. The other variables follow. We can see from this small sample of analysis that dates 2022-07-01 and 2022-08-01, the miles\_avg of the Parent Company pathing is relatively equal to that of the original pathing. The miles\_avg being even 4% higher than the original pathing, such as represented by 2022-09-01, is rare and largely shown for discussion purposes. The true benefit occurs, again for this particular metro, when comparing costs and prices. Avg costs are $0.01, $0.27, and $0.28 lower for 2022-07-01, 022-08-01, and 2022-09-01 respectively. Avg prices are $0.41, $1.56, and $1.19 lower respectively. That dollar in additional price savings alone for each order at relatively little change in miles alone is a strong case to make the item substitution toward Parent Company and reduce costs and prices for consumers and Parent Company.

There are additional metros that performed better in regards to these metrics and not all metros have been studied. It is entirely possible that other metros would not necessarily benefit from this item substitution. The specific metro studied was selected in an effort to show the relative opportunity at hand while metering expectations. It has also been selected to control for computation time and effort. Nevertheless, the item substitution algorithm appears to hold and have real application in the market.

**Problem Statement**

The bulk of the problem is mitigating the rise of last-mile deliveries in all its forms. The study of last-mile delivery has occurred for centuries with the term “supply chain” being attributed to the newspaper The Independent printing the term in 1905 [3]. The rise of the internet and more consumer choice in regards to item selection and procurement options led to increased risk of “sprawl”. Each consumer, each store location, each warehouse, and each factory have become a sprawling network of nodes that are all now interconnected. While additional choice is desired in a capitalistic economy, it also increases the risk of inefficiencies developing undetected. These inefficiencies again include more miles driven and increased costs and prices.

This also increases the risk of motor accidents that may occur while trying to deliver the goods to its end goal. According to a study by the Bureau of Labor Statistics released in July 2021, fatal injuries hit a 5-year high in 2019; specifically those who use their own vehicles to make these last-mile deliveries. Creating safe and efficient routes and paths for these drivers must be at the forefront of any company’s decision making efforts, especially to those whose main purpose is to connect and ensure the delivery of goods to end consumers.

Last-mile delivery services will continue to exist and grow, as already evidenced by the World Economic forum. Additionally, consumers will continue to use these services in order to have additional time to complete other tasks or enjoy other activities. According to a study by the Bureau of Labor Statistics released in June 2022, in regards to household activities which includes food preparation overall, women spent on average 2.7 hours on such activities and men spent 2.2 hours [4]. Using these last-mile delivery services can free up these hours in addition to other hours in the week for individuals to spend on leisure or work activities. This leads to the need to monitor the health of these services and ensure they provide the benefits expected.

These stated issues have led to the main point of this paper which is to analyze and optimize these deliver paths and products such that drivers spend minimal time on the road, consumers still receive the products they need and want in the most convenient method, and the costs and prices of the products continues to be as low as possible. While optimizing routes and drivers for deliveries, miles driven are the main variable taken into consideration. While this should remain a high priority, other factors such as safety, costs, prices, and product quality also need to be included in the equation. Last-mile delivery will continue to face challenges, much of which we’ve seen during the Covid-19 pandemic. Even getting shipping containers from ports to warehouses to local stores became a challenge. One measure of such difficulties is the Freights Baltic Index which measures container freight rates over time [5]. Since its inception in October 2016, it reached its peak of $11,109 during the week of September 10th, 2021, and only now in March and April 2023 has it gotten back down to pre-pandemic costs.

These issues are all part of that bigger problem of getting items to people where they are. My Startup is part of that ecosystem and must examine each area of the shipping process it touches in order to find optimization opportunities. That is what has defined this research paper with the overall objective of reduced miles, costs, prices, and time waste. If an item can already be fulfilled by Parent Company for a lower cost, lower price, and less than or equal to the current miles driven and/or time spent, that should net a better outcome for the shipping ecosystem overall. This lever is selected as it can be directly influenced and adjusted by the Startup. With proper maintenance of the model to find best candidate orders, this can truly improve order and delivery outcomes for consumers.

**Approach**

The first issue encountered when collecting data was the item cost and item price history for all items and all catalogs for as much history that exists at Startup. The order history is fairly decently maintained and Startup can accurately look back at older orders to see where they were delivered and final costs and prices on the items within the order, but that does not necessarily reflect which items were available for each store location and each stores costs and prices at the time the order was being placed. The data daily, or even more frequently, is ingested through each retailer and/or each store’s specific catalog API. This is only an ingestion of the entire store’s catalog for a specific day, whether the item costs and prices changed or not. While completing an entire copy of a catalog into Startup’s ecosystem is good for historical purposes, it is not efficient for computing purposes. Any SQL statement will have to traverse the entire table and output multiple rows for each item. Then it’d have to compare the order placement date with the dates on the catalog table to attempt to show prices for that specific item at that time. Not only would this have to be done for the store location’s specific catalog, it would also have to be done for Parent Company’s nearest store location so item costs and prices can be compared at that time.

This required a full rebuild of the catalog table down to the JSON payload level. I created a python UDF to be used within our SQL database to traverse the JSON object for each catalog, pull item costs and prices for each store location, compare those to the other day’s costs and prices on a rolling basis, and only create a new row of data in a new table if changes actually occurred. This would at leasta table that is either the same size or smaller than the current table state. Analysis was not completed on how frequently an item’s cost or price is changed, or even how often it goes on sale, but this was a first step to create a potentially more efficient table for analysis. Additionally, since this had not been done yet, the current data sample only contains data from 2022-01-01 to 2023-03-24. This pricing history will be expanded soon as there has been increased interest at Startup for this data. This is a process I’m working with others to address which will not be completed in time for this analysis.

After getting that pricing history for each item at each store location in Startup’s ecosystem, the next step was to calculate the closest Parent Company location to each other retailer store location. This required pulling each store location's latitude and longitude data into a single table, pulling out Parent Company’s store locations into its own table, and then comparing store locations that existed within the same metro as defined by Startup. Then I calculated the distance in miles based on the latitude and longitude using a python package called “geopy”. One issue that may arise here is that I am calculating the distance in a straight line as opposed to navigable street distance. The street distance data is available in another poorly optimized table that would require more data cleansing as the item cost and price data. Since the latitude and longitude data exists readily for store locations and for customers, that would be a reasonable proxy to save on computing costs and data cleansing time. Due to the nature of these types of last-mile deliveries, they mostly occur in densely populated areas and therefore the latitude and longitude distances would be fairly accurate.

After collecting the item cost and price histories, and then finding the nearest Parent Company store location, it was simple to also pull in customer latitude and longitude information to calculate the distances each node was to the customer; each node being a store location, a Parent Company store location, and the customer location (the miles will remain in this paper, latitude and longitude data is completely omitted.) These nodes are then combined as shown in the small subset node graph below in Figure 2. These drawn paths make some assumptions about what can occur in the market. Each store and customer is connected which creates a new variable for each path. Additionally, each store is connected to each other to indicate a trip from store to customer to next store to next customer. Possible “bundling” of orders does not occur in which a driver may take 2 orders from the store at the same time and deliver to 2 different customers. This model behavior can be included in future models. 

Once connected, the Traveling Salesman Problem [1] can be used to calculate the shortest path, each path following this simple calculation:

In this calculation, TotalWeight\_p is the total weight for each path, miles is the distance between each location, eligibility is the percentage of items within an order that have an exact equal at a Parent Company store location, and the item\_cost and item\_price are the cost and price of those exact items at the Parent Company store location. This becomes the general minimization cost function as follows:

where Z is the total cost. As seen here, the current state does not weigh one cost over another. I’ve also excluded constraints here and allowed only the costs to find the lowest cost total pathing. Additionally, each variable was normalized in order to avoid any one variable overshadowing another due to its greater magnitude. This formula is as follows:

where “value” is an individual data point and “variable” is the selected variable in question.

Due to the larger number of variables in what also amounts to a linear programming problem, libraries such as gurobipy [6] were unavailable as the computing power and software became a proprietary cost. Therefore, I opted for a simulated annealing method as first described by Kirkpatrick, et al. in 1983 [7]. In their study, the simulated annealing algorithm begins with a certain temperature and cooling rate. It uses these values and differencing between current solutions and the next candidate solution to probabilistically choose either the candidate solution, even if the candidate solution is not better than the current solution. This method attempts to avoid getting stuck in a locally optimized point by randomly broadening the search again for a better solution. My application of the algorithm overall can be described in these formulas as follows.

1. Initialize a starting Temperature and cooling rate. These variables determine the probability of selecting the new candidate solution as follows:

Here, initialTemperature is the starting temperature selected, delta\_cost is the difference between the current solution’s cost and the new candidate solution’s cost, and , which is constantly reduced by the 1 - the cooling rate with each iteration.

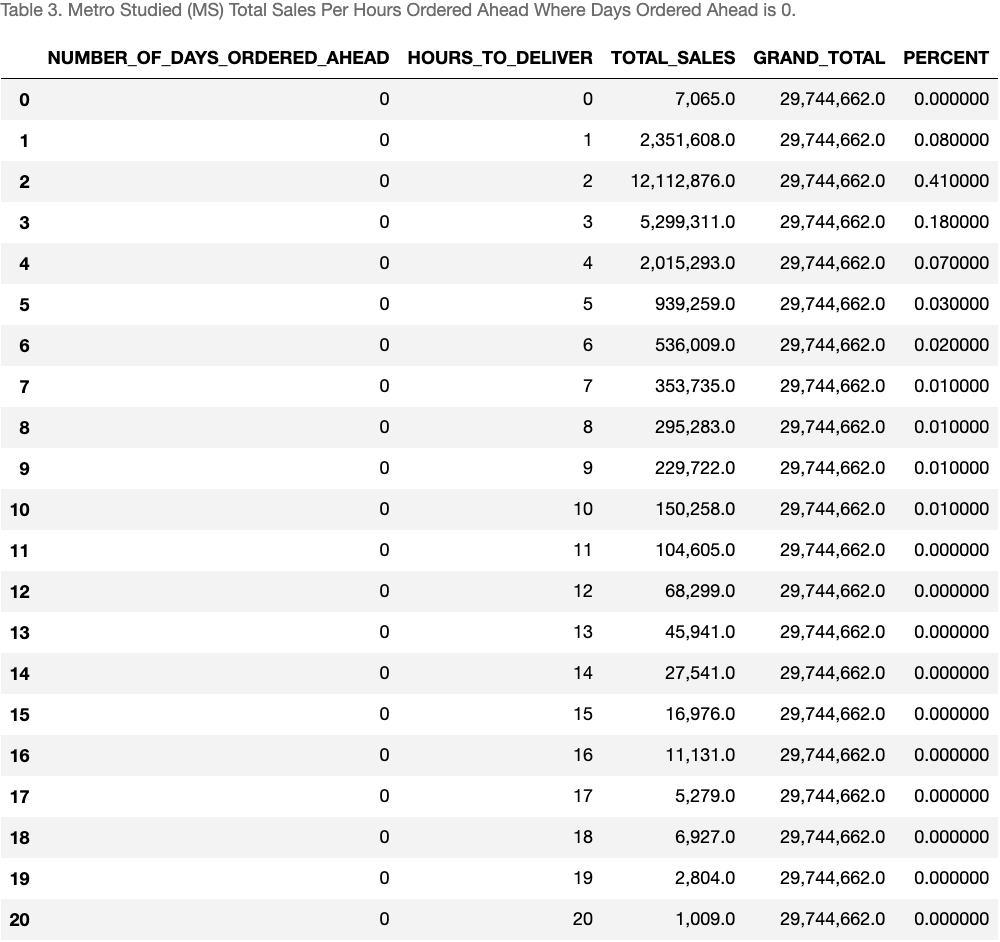
1. Randomly generate a possible pathing that starts at a store and alternates between connecting a customer and then a store without any customer node repeats.
2. Sum the cost of that pathing and initialize as the current solution
3. Randomly switch 2 nodes to create a new candidate solution
4. Calculate the new solution’s cost.
   1. If it is less than the current solution cost then accept it as the newer current solution.
   2. If it is greater than the current solution, then apply the function in step 1. Then compare step 1 as follows:

:

This utilized the python “random” package. The random.random() method randomly generates a float number between 0 and 1. Again this attempts to pull the algorithm out of locally optimized values and re-broaden its search at random increments.

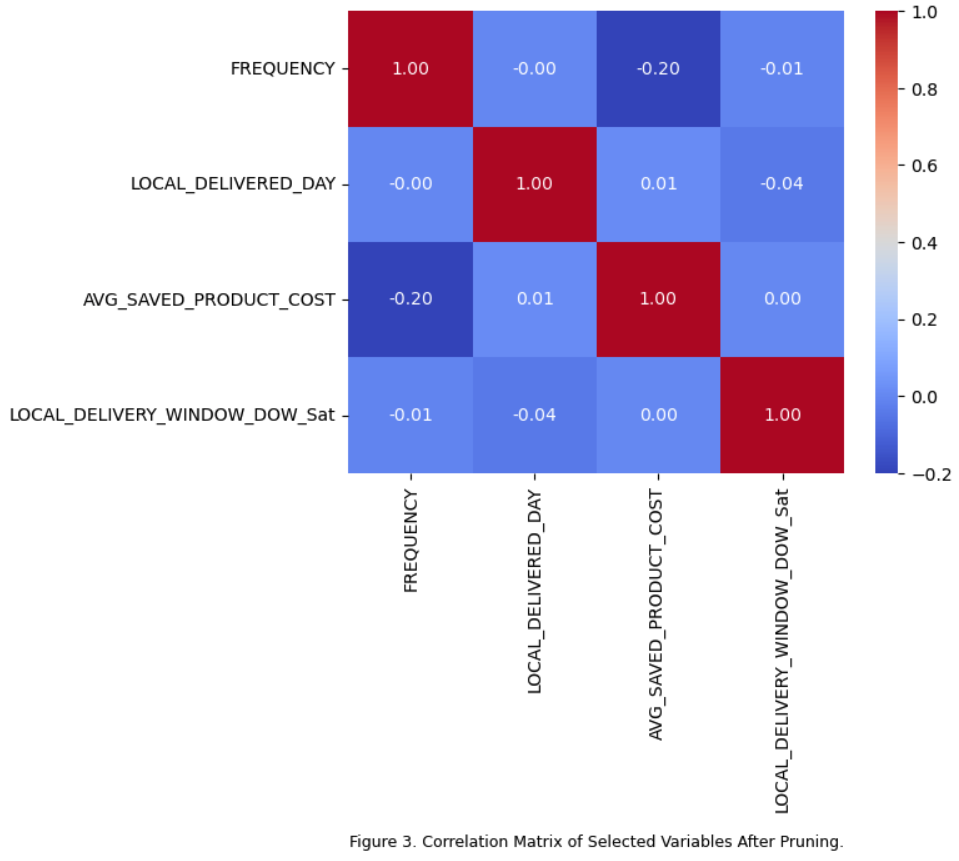
1. This continues until the value gets to a lower threshold of 1e-8 (0.00000001)

The final output of the above steps is the best solution the algorithm found. The results of which can be seen again in Table 1. This algorithm is applied to a scenario where the original store location pathing occurred and hypothetically was also optimized, and then also where the optimized Parent Company location pathing occurred. This comparison was done in order to have a more 1-to-1 comparison of both situations as it would also be completely possible to use this optimized pathing algorithm with the current stores. However, the main case is showing the possibility of item fulfillments with items from the Parent Company.

The general algorithm works in finding an optimized path for orders that have already occurred. It was able to take the orders in a day and recreate a better path that would have not only decreased the miles traveled, but also reduced the total costs and prices if switching to Parent Company item fulfillment. This was the main bulk of the analysis and appears promising. However, a bit of analysis had to take place in forecasting the next day’s orders in order to pre-optimize the pathing prior to dispatching a driver on their way. To determine the size of the preordered orders that occur in this metro’s ecosystem, I aggregated total sales based on the number of days ahead an order is placed, and then the number of hours ahead an order is placed. Those 2 tables are below. During the time period of 2022-01-01 to 2023-03-24, customers generally order within the same day they expect the order with 83% of sales ordered and delivered within the same day and 41% of sales are ordered and delivered within 2 hours. About 16% of sales are ordered a day ahead which amounts to a total of about $4.8m. When comparing these items to Parent Company substitution costs and pricing, a total of $153,534 would be saved. This in itself does not hinder or disqualify this research. If there is substantial miles reduction, cost savings, and price savings that can still be had, there can be additional incentives placed on either prompting customers to delay their orders to the next day or other incentives that can allow optimization to occur. These additional incentives are still outside the scope of this paper.

However, additional analysis can be done in forecasting the next day’s orders and running the potential optimal path with the item substitutions. The forecast can be completed at a metro area, an area level representing a cluster of customers, or at a customer level, although that granular of a level is computationally intensive and also beyond the scope of this paper.

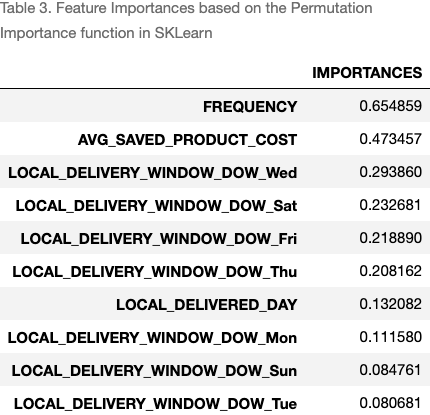
The 2 methods I focussed on are Keras Neural Network [8] and GARCH [9] modeling. The purpose of the neural network is to begin to find patterns within the order specific data itself and to establish a model for analysis. The model can then be slowly updated with additional variables such as shopper ratings, customer feedback, and general app/website usage information to better predict the next day’s sales. The GARCH model is developed to understand the general seasonality of the underlying sales data and the trends of the single variable, independent of the other variables used in the Keras Neural Network. Through this basic framework we can slowly build a better understanding of our customers and our delivery needs while also maintaining adaptability in the models.

The first model explored was the Keras Neural Network, performed on a training set and a testing set of data split by sklearn’s train\_test\_split method which randomly splits the data based on the percentage of testing data desired. The general understanding of the structure of a neural network can be seen in Appendix A. The mathematical model within each layer essentially works to assign the correct weights and biases as follows which is the dot product of the **w** and **x** vectors and includes a bias term. The **w** vector is the coefficient, or weight, on the variable **x**, while the bias **b** is essentially the constant value present in each **z** calculation. These act similarly to any regression function with coefficients and a single constant. The layers then work together to minimize the sum of the z value based on whichever loss function is passed to the model. The “loss” will be the deviation from the actual target variable one is trying to predict. The models then allow for different loss functions to be applied to each layer, and a neural network inherently allows for different numbers of layers, iterations, loss tolerances, and optimization methods to be applied. Therefore, in order to select the best parameters to find the best model based on the data used for training, I used a python package called GridSearchCV [10]. This package allows one to input nearly any model and a list of potential candidates for each parameter. It creates every permutation of all parameters listed. Then for every permutation, it will create a number of subsets of data based on the desired cross-validation, shown in Appendix B, number input by the user, which can also be hyper-parameterized. It will then generate scores based on the subsetted data scores and calculate the desired loss function value; the loss function also being one of many available. 

I used a limited list of variables that were available on the order and performed this analysis for a broader range of variables. However, after analyzing the losses and train and test errors, I focussed on simplicity with very minimal increase in errors. The results are excluded from here for brevity. A small correlation matrix to the right in Figure 3 represents the main variables used in the network to calculate Next Day Sales. The variables Local\_Delivery\_Window\_DOW\_Sat represents one of the 7 days part of the variable Local\_Delivery\_Window which was broken out into dummy variables.

The search will return the model with the smallest loss total, which again depends on the loss function input. In the case of this data, the Huber Loss function performed best. This loss function is as follows:

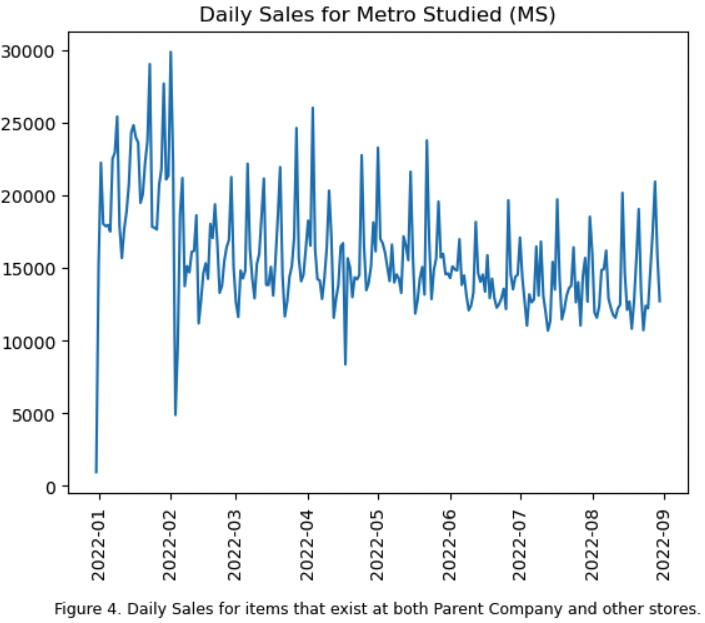
1. Huber Loss , where delta is the threshold to switch between the 2 scales, x\_n is the current datapoint and y\_n is the variable’s mean. The Huber loss can also be calculated as a mean instead of a sum, but this paper focuses on the summation. The Huber loss is often used to automatically account for any larger outliers in the dataset. Since this was the loss function chosen by GridSearchCV, there could be more analysis around the larger orders to determine causes and opportunities around other product offerings.

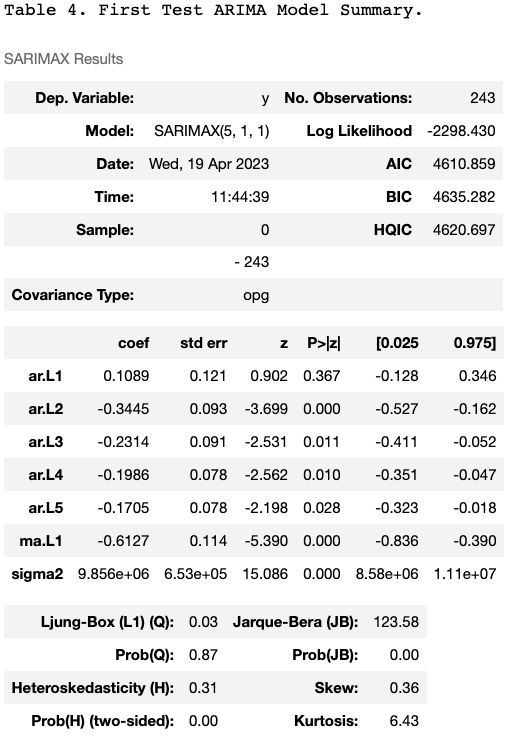
The rest of the best parameters can be reviewed in the accompanied code. After finding these best parameters I fit the data and calculated the train and test errors again using the Huber Loss function. The train and test error were 0.0158 and 0.0157 respectively. This was against the normalized data just as it was normalized in the Traveling Salesman Simulated Annealing model prior, except this is on a larger dataset. This error is in respect to the target variable of Next Day Sales since again the idea is to optimize the next day’s pathing ahead of time. When the train and test errors are denormalized, the train and test error are 0.323 and 0.321 respectively. This amounts to about a $0.32 error. This data is at the specific item level sales forecast. Therefore that $0.32 is in regards to the average next day sales per item, including cases of zero dollar sales the next day. This average is $6.73, which means the $0.32 represents a plus or minus 4.75% error which is excellent performance. This establishes the model and possibility of being able to forecast which products would be sold the next day and where, which can assist with planning and creating the optimal pathing.

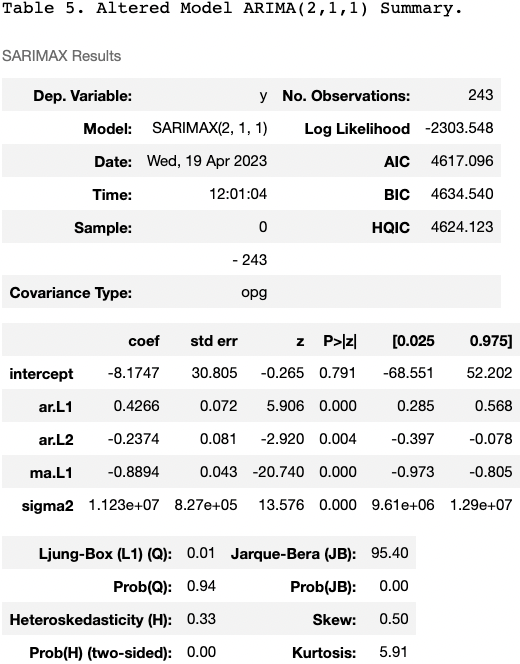
The final portion of the neural network to mention is the feature importances. Although neural networks are often difficult to explain, I would not use it as a model without having some explainability. Therefore I relied on the permutation\_importance function from the Sklearn library [12], represented in Table 3. The function randomly shuffles the target feature, recalculates the scores of the original model, and then compares it to the original best fit score. It calculates an average score based on how many times it reshuffles the target variable data; which can be controlled by the user. Higher score naturally means higher importance. The first thing to observe is that Frequency has the highest score. This value is a proxy for the product\_id’s themselves. Due to the large number of different products, I simply calculated the frequency of the items in the dataset to attempt to save on compute cost and time. This being the most significant feature can make intuitive sense. An item purchased the most often will most likely be purchased again the next day. We can start to forecast and plan for items that are purchased consistently and plan paths ahead of time based on this information. With more resources, each individual product can be included at perhaps a more granular level. (A QQPlot of the residuals of this model is included in Appendix D.)

As previously mentioned, I also applied a quicker GARCH (Generalized AutoRegressive Conditional Heteroskedasticity) model to the analysis. Since the product frequency and cost were good predictors of the next day's sales, which can make intuitive sense, I decided a simpler GARCH model can help serve as a less computation-heavy model to make predictions at any level; metro, product, or customer. The model works in by creating 2 distinct models as follows:

Main\_model = , Residuals\_model =

where , meaning that each daily period's error term is normally distributed and independent of prior day errors. Then is the error term of the residuals model. GARCH works to find the trends in the daily sales change and the trends in the error term/variance itself to capture any volatility. These models use the past values of the target variable to predict the future values. It assumes in general that the future is a function of prior periods values and the prior period’s error terms (or variance). 

In my analysis, I looked at how the models together would forecast the next day's sales for the period leading up to 2022-09-01, the final day reviewed in the rest of the paper. I used a Python package called pmdarima[13] to automatically fit the data and then I reviewed the coefficients it selected. Here is Figure 4 which plots the daily data. The results of the autofitting of only the model, without modeling the residuals, is in Table 4. From the results, we can see that the model has a low Ljung-Box score, indicating that the residuals of the model’s forecasting are independently distributed, however the first coefficient does not appear significant. Additionally, this model is ARIMA(5,1,1) (Autoregressive Integrated Moving Average,) indicating a possibly more complicated model than desired or necessary. Due to the parsimony principle, we’d prefer the simplest model possible with only a small increase in errors of prediction. I therefore manually reduced the lags needed in the model to ARIMA(2,1,1). The results of that model are in Table 4. (Diagnostic plots are also available in appendix E Figures 5 and 6). Since the Ljung-Box Score improves and all coefficients (besides the intercept) of the ARIMA(2,1,1) model are significant, having this simpler model has strong statistics to indicate its usefulness in predicting the next day sales overall.

Using the ARIMA(2,1,1) model, I then worked to create a model for the residuals. This model’s results are included in Table 6 in Appendix F. Both the “Parsimony fix” model and residuals model indicate that the error in prediction terms are not autocorrelated, yet they also do not appear to be normally distributed, contrary to the QQplot in the diagnostics figures. 

Overall, the models indicate strong promise, yet with lack of additional compute resources, I was unable to gather more years worth of data which hindered the seasonal analysis the model would benefit from. Even with this issue, there still is a bit of promise in using this historically financial based modeling to measure the trends and seasonality of the sales data of Startup’s delivery services and therefore predict the next day sales of a product and optimize pathing.

**Conclusion**

Overall, substituting items from other stores for Parent Company items that are exactly the same seems like a plausible option. It reduces miles, reduces costs, and reduces prices. Additional benefits to the drivers is likely, besides reduction in miles driven, but more analysis is needed around driver ratings and tips in that regard. The neural network outlined in this paper also appears promising on determining the plausible orders for the next day that can then allow an optimal path to be created. It can also open the possibility for Startup to review incentives for consumers to either delay their order in a day (which we saw they usually order and then receive their order within 2 hours) to allow for optimal pathing to be created.

The ARIMA model requires more study. Although it is a useful model in its simplicity, the lack of more historical data hindered the analysis greatly. The work on getting proper historical data for item prices is continuing and I’ll be able to maintain these models and run them on a larger dataset over the summer. The framework remains and can be applied once better data is achieved.

In conclusion, this paper’s framework of analysis makes a strong case that item substitution is possible. In the worst case scenario, this paper reveals that this program requires additional study in order to refute its significance.

**Check List for Proposal and Final Report  
Check that you addressed each item and initial it to the right**

1. **Structure of document**: fond, spaces, minimum number of required pages. Font should be 11. Space should be single or 1.15. Margins should be 1in all around \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_JR\_\_\_\_\_\_
2. All **abbreviations** are defined when first appeared\_\_\_\_\_\_\_\_\_JR\_\_\_\_\_\_\_\_\_\_\_
3. All **figures and tables** are labeled with numbers and captions. Figure

captions go under the plots and table captions above tables. Make sure you

indicate units. \_\_\_\_\_\_\_\_\_\_\_\_JR\_\_\_\_\_\_\_\_\_

1. Figures, tables, diagrams, pictures taken from the web or papers should

include the source otherwise it is plagiarism\_\_\_\_\_\_\_\_\_\_\_\_\_\_JR\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. No text should be copied and paste from anywhere as it is plagiarism. If you

quote someone you should put it under “....” and identify the source as a

reference but it should not be more than a couple of lines.\_\_\_\_\_\_\_JR\_\_\_\_\_\_\_\_\_\_\_\_

1. All listed references should be mentioned in text. Use format [1], [2]
2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_JR\_\_\_\_\_\_\_\_\_\_\_\_\_
3. **Grammar and English** should be carefully checked . No report with typos

and grammatical errors will be accepted. Check the tenses. Some students tend to use parts of the proposal that uses the words ‘I will’ , ‘I plan’ etc. The final report should state what you did not plan to do. \_\_\_\_\_\_\_\_\_\_\_\_JR\_\_\_\_\_\_\_\_\_\_\_

**PLIAGIARISM:** *Any identified form of plagiarism will be reported to the University and in addition to receiving no credit for EE590 the graduation of the student may be affected. All past reports are kept and a software program is used to identify overlaps with past reports and literature. If you use figures or tables or diagrams from a website or papers you need to indicate that and include the reference..*

I confirm that my proposal/report abides with the above guidelines and no material was copied from any source that is not indicated in the report and referenced accordingly.

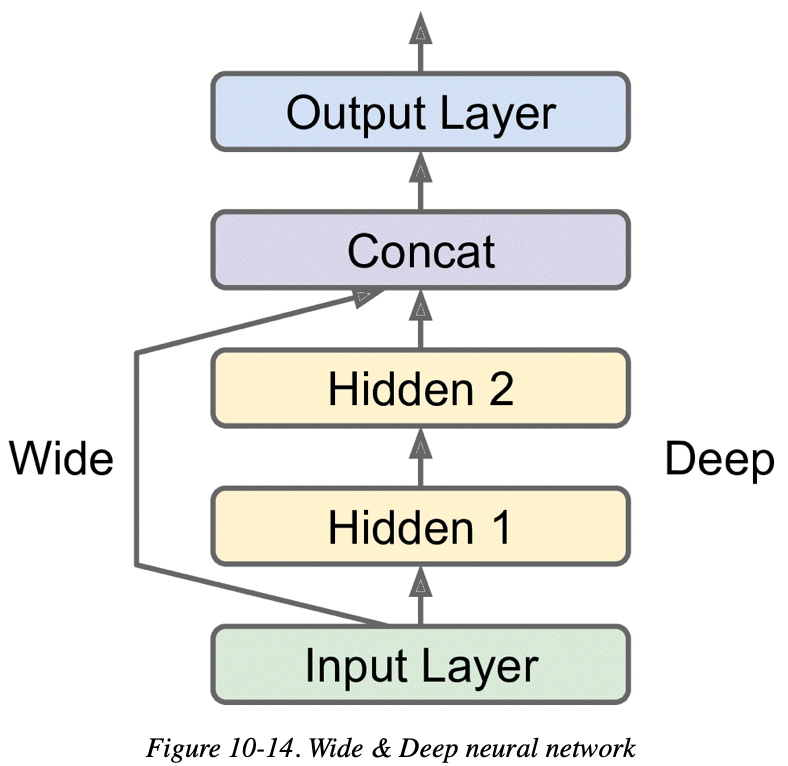
Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_Joseph David Ruiz\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SignatureText, letter

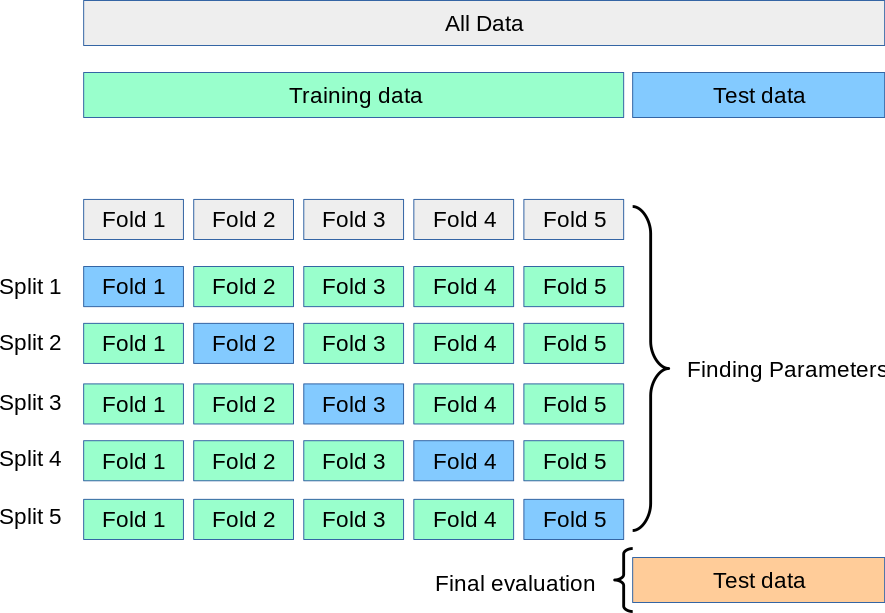
Description automatically generated

**Appendix**

**A.)**

**Taken from Reference [8]**

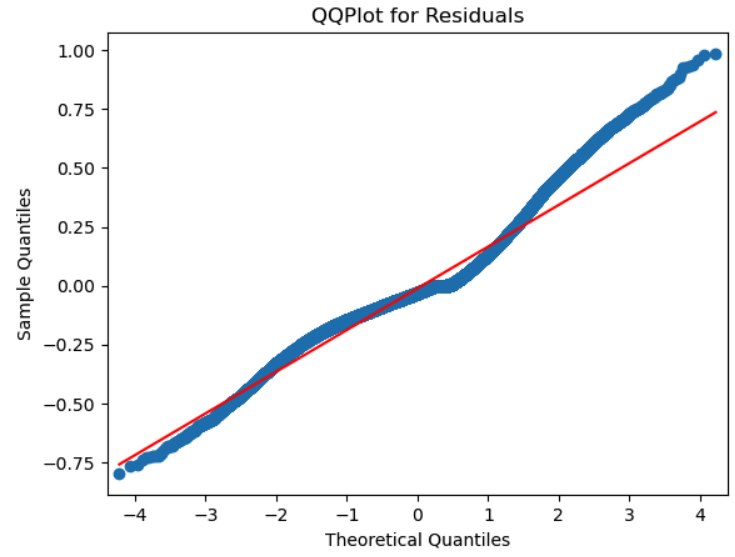
**B.)**

** Taken from reference 11**

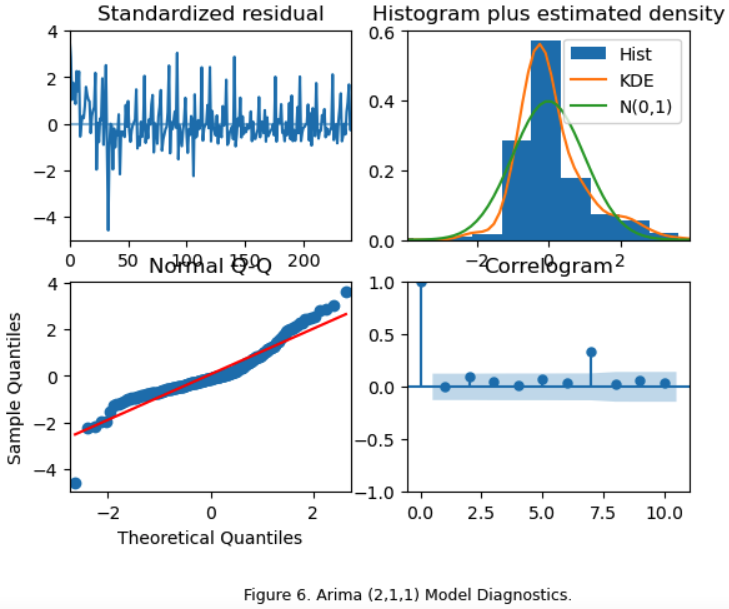
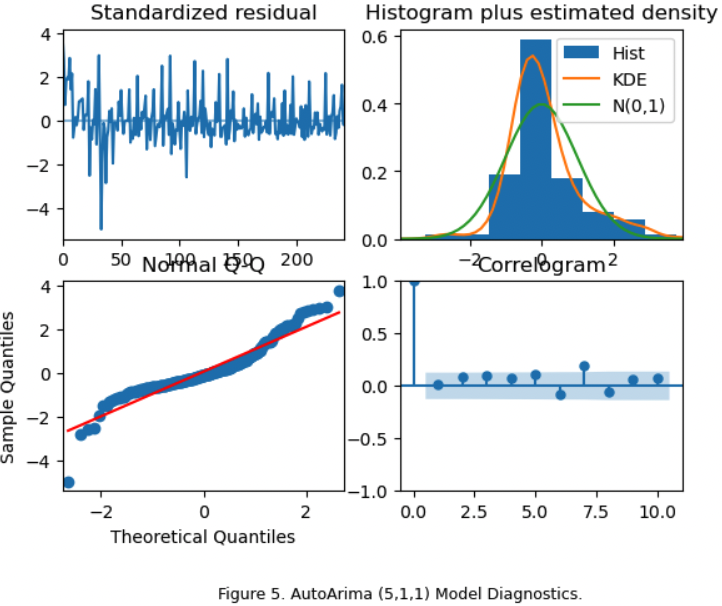
**C.) Software Used:**

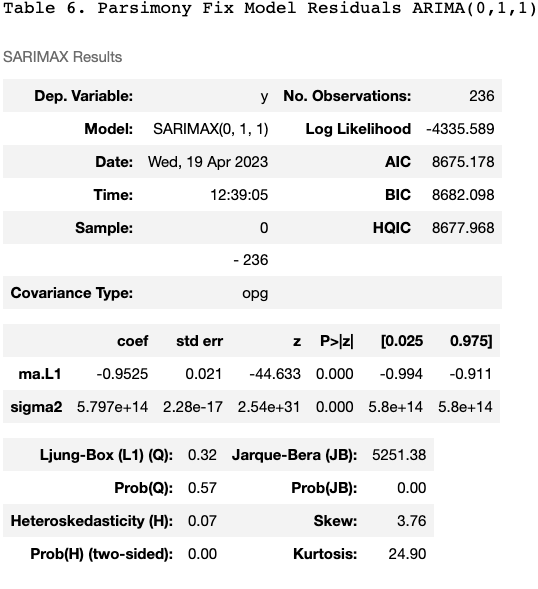
* **Anaconda**
* **Python**
* **Jupyter Notebook**
* **Snowflake (SQL)**
* **Google Sheets**
* **Google Colab (code sharing for final report)**

**D. KNN QQPlot of Residuals. Datapoints closer to the redline indicate a closer normal distribution of the residuals**

****

**E.)**



**F.)** 

**References**

1.) Lawler, E. L., Lenstra, J. K., Rinnooy, K. A. H. G., & Shmoys, D. B., The travelling salesman problem. A guided tour of combinatorial optimization. Chichester., 1987

2.) <https://www3.weforum.org/docs/WEF_Future_of_the_last_mile_ecosystem.pdf>

3.) <https://www.supplychainopz.com/2013/05/history-of-logistics.html>

4.) [Fatal injuries at a 5-year high for driver/sales workers in 2019 : The Economics Daily](https://www.bls.gov/opub/ted/2021/fatal-injuries-at-a-5-year-high-for-driver-sales-workers-in-2019.htm)

5.) [Freightos Baltic Index](https://fbx.freightos.com/)

6.) [Gurobi Optimization](https://www.gurobi.com/)

7.) KIRKPATRICK, S., GELATT, C. D., and VECCHI, M. P. 1983. Optimization by

simulated annealing. Science 220

8.) Géron, Aurélien. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media. Chapter 10. Introduction to Artificial Neural Networks with Keras and Chapter 11. Training Deep Neural Networks.

9.) Brooks, Chris. Introductory Econometrics for Finance (p. 396). Cambridge University Press. Chapter 9.8 Generalised ARCH (GARCH) Models.

10.) [sklearn.model\_selection.GridSearchCV — scikit-learn 1.2.2 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV)

11.) [3.1. Cross-validation: evaluating estimator performance — scikit-learn 1.2.2 documentation](https://scikit-learn.org/stable/modules/cross_validation.html)

12.) [4.2. Permutation feature importance — scikit-learn 1.2.2 documentation](https://scikit-learn.org/stable/modules/permutation_importance.html)

13.) [ARIMA estimators for Python — pmdarima 2.0.3 documentation](http://alkaline-ml.com/pmdarima/)