# Johnson & Johnson North America’s Deliver Data Science Project

**Jonathan Sohn**

## Overview

The goal of the project was to produce a data science solution to create a cost estimator for total freight costs on shipments from the USAID *“Supply Chain Shipment Pricing”* Dataset. To achieve that end, I primarily relied upon using the Python programming language with various libraries and packages documented below and within the code. After using Python within a Jupyter Notebook for Exploratory Data Analysis/Data Cleaning, Model Building, and assessment of Model Performance, I have compiled my findings here.

These findings were developed under the assumption of working in a limited timeframe in a real-life working environment. As such, there is also a section at the end recommending future directions for experimentation and exploration I did not have time to get into, but I believe would be ways to explore model improvement.

## Summary of Findings

The USAID “*Supply Chain Shipment Pricing”* Dataset was found to have a wide variety of features ranging from datetime, numeric, to categorical. Sweetviz was used for Feature Selection and narrowed down a list of features considered to be significant. It was also discovered that many lines within the data were a part of multi-line shipments, and that grouping of data was required for accurate model construction.

Given the strong skew to the dataset, I tested using a Random Forest Regressor and Gradient Boosting Regressor. Further exploration normalizing the dataset by scaling or log transforms and reading into more linear regressors could be an interesting direction to take, but I chose the models I did for speed in the moment.

With all the above, a model was found that could predict cost with a MAE of ~$2000 and a MAPE of ~100%. The median errors were much smaller (~$1000 and ~30%). Looking at the mean errors suggest they are inflated by a small number of errors that were very large compared to the rest. I also found a MAE of ~$1000 and MAPE of ~20% with a more filtered dataset with results in the bottom of the notebook.

The models appeared to generally overstate lower values and understate higher values. Combined with some of the very large errors, some next steps I would recommend are:

* Improving outlier identification and removal
* Scaling input features (to test other models)
* Better understanding the business details and context of the dataset
  + This can help understand outliers
  + This also can help feature engineering of the many categorical columns

More details on my process to reach these findings are found on following pages. The Jupyter notebook can be found in the attached folder under the name “Data Exploration.ipnyb”.

## Languages, Libraries, and Packages Used

**Python – Written using Jupyter Notebooks**

* Data Manipulation, Plotting, Exploration, Cleaning
  + Pandas
  + Numpy
  + Sweetviz
  + Matplotlib
  + Seaborn
* Data Preparation, Model Building, Analysis
  + Scikit-learn
    - test\_train\_split
    - cross\_val\_score
    - RepeatedKFold
    - RandomizedSearchCV
    - RandomForestRegressor
    - GradientBoostingRegressor

## Step 1 - Data Exploration and Cleaning

Data was initially loaded in a .csv format through Pandas. Sweetviz was used to explore the shape of the data, and many columns were found to be typed as the wrong type of data, including the target column “Freight Cost (USD)”. It was also discovered that there were multiline shipments in the data, and that some of the shipment lines also didn’t have a Freight Cost.

First, data types were forced to numeric, categorical, or datetime types. Only one datetime feature was chosen to be extracted into year and month features based on similarities between the date columns. The properly typed data was then explored through Sweetviz again for feature selection by examining the association graphs and correlation details provided by Sweetviz.

A list of features selected with their supporting logic is provided below:

* Numerical Features – These were all chosen for their high-ish correlation with the target
  + Weight (Kilograms)
  + Line Item Value
  + Line Item Insurance (USD)
  + Line Item Quantity
* Categorical Features –
  + ASN/DN # - This was chosen purely to be used to group the multiline shipments
  + Country, Shipment Mode, Dosage Form
    - Many other categorical columns had high correlations, but were all suspected to overfit the data due too high correlation with the target, and from having too many unique values
    - I also narrowed features due to many categorical columns having strong associations with another

After all the features were selected, they were grouped by “ASN/DN #” to properly capture multiline shipments. Numerical features were summed under assumption that the whole shipment was worth their aggregated values. Categorical features were chosen by the “first” metric as it was assumed they would be the same across the shipment, a safe assumption except for the “Dosage Form” column. Further exploration can confirm the safety of that assumption.

Upon grouping, I then cleaned the data by removing null values, and values of 0 within the dataset under assumption that the data either shouldn’t have values of zero for most of these values, or that the 0 values were infrequent enough that they could be treated as outliers that would skew the dataset.

Other outliers were identified and removed through removing all values above the cut-off “IQR \* 1.5” for a given numeric column. This method was chosen both for simplicity and because there were some especially infrequent but extreme values in all the columns that extremely skewed the dataset. In future exploration, we could explore the business reasons behind these anomalous values to better identify them to either remove them, or to build them into the model.

Finally, after cleaning, I one-hot encoded the categorical values so they could finally be read into a model, and the model building phase and analysis phase began.

## Step 2 - Model Building and Optimization

Values were split 70/30 into a training and testing set for initial model construction and review. I used Random Forest and Gradient Boosting Regression models for their ease with working and due to the strong skews in the data. As stated before, a future point of exploration could be the performance of other models using normalized/scaled input values, but for now I ran with these two models for speed and efficiency’s sake.

I built plots to view model performance and errors from the train/test split data, along with sections of code to cross-validate the data to check for overfitting of data as I believe this dataset could be quite prone to it due to the specificity of some of the lines and the relatively sparse data (even more sparse after the grouping and data cleaning).

Not currently captured in the notebook was a brief amount of exploration of trying out different categorical values for their impact on model performance. I found that using no categorical columns degraded model performance, while adding too many caused overfitting of data that degraded cross-validated scores. The categorical columns selected above were what I found manually (and quickly to be more effective. Further exploration can be conducted to verify that process through a less manual process however.

To optimize the models, I also ran a Random Search Cross Validation to find optimal hyperparameters, then used those in the “final” models. I deviated a small amount from the suggested value for parameter “n\_estimators” since typically it has large diminishing returns after a certain amount. Making it too large causes models to be too slow to run frequently while testing, so I made it smaller.

## Step 3 - Analysis of Results

The MAE and MAPE of the models were quite similar with only marginal differences between the two, which was somewhat expected. They were also found to be quite inflated vs the median values due to some few very large values that greatly skewed the dataset.

Graphing the errors through Sweetviz and creating my own histogram showed that the errors were generally a lot smaller than the mean. The few but very large differences suggest that there are still some major outliers greatly skewing the dataset and affecting the performance of the model. I had only removed the high value outliers, and I suspect that the large errors in MAPE were primarily caused from the small number of very low-cost shipments (<$1000).

Graphing the predicted values against actuals shows a tendency to overestimate small values and underestimate large values. Future points of exploration to improve the models here besides outlier removal are scaling the data and oversampling from large and small values to have the model weigh those more heavily.

Altogether, the model as it currently stands looks to be okay at estimating values within the most common band of possible inputs. However, it begins to be drastically off on the possible extremes of values, which inflates the MAE and MAPE.

I also compared the results against baseline models using the median and mean of the training set. By comparing the results of the Machine Learning models against these naïve baselines, I found that the Machine Learning models did see a significant improvement in MAE and MAPE against the baseline. The mean and median performances were better, and the standard deviation and variance of errors were lower as well, suggesting a better distribution of errors and better model performance in general.

However, the range of errors was about the same, suggesting that the Machine Learning models might be mispredicting some of the errors in the same manner as the baseline models. This would be most likely for very low and very high values in the dataset, as suggested by my suspicion above with low Freight Cost shipments that make less than 5% of the grouped dataset.

At the very bottom of the notebook, I ran a quick test where I also filtered out all rows with Freight Costs <$1000, and found a cursory examination of results returned a MAE of ~$1000 and a MAPE of ~20%, which was a massive improvement. This increases evidence for the recommendation to explore better methods of outlier identification and handling, as those appear to be the largest impactors upon the performance of the model.

## Recommendations for Future Exploration

* Better understand the business context of the dataset
  + This can help with outlier identification
  + This can also help with better feature selection
* Improve outlier identification/removal, and scale input features
  + This can help with the extreme values within the data
    - There are some very large values that skew the dataset far to the right
    - There are also some very small values that might inflate MAPE
    - Both large and small values can inflate error due to the large of orders of magnitude in the data (the smallest freight cost is $0.75 while the largest is ~$300,000, 6 orders of magnitude apart)
  + We can also explore the use of other models that assume Gaussian distributions this way
    - With several models, we can also build ensembles/stacked/bagged models that can also possibly improve performance
* Use more data
  + Either gain more datapoints if possible, or not remove as many values
  + If we better understand the dataset, we might be able to keep more values (currently, we lose over 50% of initial rows through grouping and dropping)

## Other Next Steps

Once final model parameters are found and sufficient outlier protocols defined, a production model could be constructed. A new process for taking in new data and properly cleaning it and inputting it would need to be made, along with a process for updating the model with new data over time.