Predicting Shipping Prices from Great Britain to India and Bangladesh Using Machine Learning

Zachary J. Fuller

Northwest Missouri State University, Maryville MO 64468, USA
s553830@nwmissouri.edu and zachfuller1@gmail.com

Abstract. In this report, the total cost of shipments from Great Britain to India and Bangladesh are predicted based on a number of input features. Data is cleaned and explored using Pandas, Numpy, and Plotly Express and Plotly Figure Factory. Linear regression and random forest regressor models are created using Sklearn to predict the total cost of shipments and evaluate the relationships between them. The results show that gross weight was by far the most important feature in predicting the total cost shipments along these routes, and despite the low importance of the other input features, these models still performed very well.

Keywords: data analytics · data science · machine learning · logistics · shipping · pandas

1 Introduction

This project will cover the domain of logistics. I currently work for a freight auditing company, so my knowledge of the industry will be helpful in this analysis. My data comes from Kaggle, as there are a couple of decent data sources on there that are relevant to this topic. My goal was to do some predictive analytics using previously shipped packages as a training set in order to understand the most important input variables to predict the target variable, total cost of a shipment. Predicting the cost to ship something based on multiple factors like weight, destination, origin, and time spent in transit can be accomplished using a regression model. I feel that this could be useful to shippers, as carriers' pricing policies are not very transparent unless you are well versed in logistics, so having a pricing estimator could empower small business owners and online sellers to choose the best shipping option for their needs.

The steps I followed to complete this project were:
1. First, I located my data set.
2. Then, I cleaned the data set using the python packages Pandas and NumPy, removing any fields as necessary and deciding how to handle missing values.
3. Then, I used Sklearn to separate the data into a train and test set. 4. I then evaluated the results. As needed, I retrained and re-tested the data, repeating steps 2 and 3. This included some more data cleaning to remove outliers and testing different regression models.
5. After training and testing was complete, I used a python plotting package,

Plotly, to display the results.
6. Finally, I compiled this report, making sure to examine any possible errors.

The key components of my approach were finding the right data sources, making sure the data was cleaned in a way so as not skew results, and most importantly, the training and testing process. This was completed using multiple linear regression models and a random forest regressor model, so making sure that the data was trained correctly was key finding the best fit to account for each independent variable.

2 Data Collection

My data set was found on Kaggle [9]. I have also found some supplementary information regarding regression models and their results from a blog called Algosome [2]. The data was found in a few CSV files. I did not have to use any data scraping techniques, as my data set was made available for easy download through the host site. From the "Shipping Optimization Challenge" data set on Kaggle, I used the send time stamp, source country, destination country, freight cost, gross weight, and shipment charges fields. The data set had thousands of columns, so I planned to fill in some null values. Additionally, I considered slicing off the timestamps and only using dates, as it appeared that a lot of this data involves international shipping, and the timestamp could be less important and take up space/time. Ultimately, the approach I settled with was using a different field called shipping time. Finally, I needed to convert the country fields, as they were simply abbreviations in the original data set. I planned to read the CSV files into a Pandas data frame within a Jupyter notebook, merge the necessary tables and fields together, and then perform this cleaning.

3 Data Processing/Cleaning

Curating the data means that we clean and manipulate the data as needed, with the desired result being a more concise and complete data set for further analysis. In the case of my data set, I viewed the data in a Jupyter Notebook using a Pandas data frame. My data was in 2 files with the same structure, so I had to concatenate the 2 files. After examining the number of unique values and any missing values in the data frame, I determined which columns provided useful insight and which ones were bulk that could be dropped. I also had to recreate some of the columns that were stored as hard to understand abbreviations so that they could be more easily understood by myself and any observers of my final project.

I used the python packages Pandas to create and manipulate my data in the form of a data frame. I used different Pandas methods like dropna to drop rows with missing values, drop to drop unnecessary columns, unique to view the unique values in each column, concat to combine my 2 data sets, isnull and any to find columns with missing values, and map to create a new column based on given values in already existing columns [6]. I also used the python package Plotly, specifically the modules Plotly Express and Figure Factory, as Plotly is my preferred plotting package. The Plotly.express function "histogram" was used to view the distribution of my data [4].

The only column that was missing data was called "shipping time". About 19.7 percent of the values in this column were null. I figured that this would be a very important independent variable in my later model, and I didn't want to feed any poorly imputed data into my model, so I dropped all of the rows that had missing values in this column. The data science phrase "garbage in, garbage out" was a guiding principle in this decision.

After cleaning the data, I was left with 5114 records/rows and 10 columns/attributes.

Below are some important descriptions of the attributes that I kept in my data set:

• shipment id: This was a unique ID given to each shipment in the original data set. I debated whether or not to keep this in or simply refer to individual records with the data frame's index, but I ultimately decided that it may be useful to have this ID for further reference in the future. [9]

• freight cost: The cost/kg of the shipment. [9]

• gross weight: The weight in kg of the shipment. [9]

• shipment charges: This one was confusing to me at first, but looking back through the documentation on Kaggle [9], I learned that this is the minimum amount that a company would charge for a specific shipment. I did not end up using this in my final analysis, but at time of cleaning, I figured I would keep it in there just in case it was relevant. [9]

• total cost: This was a field I calculated by multiplying the freight cost by the gross weight, as the freight cost is the charge per kg. This will be the dependent variable that I want to predict.

• shipment mode: The way the product is shipped, either by air or ocean. [9]

• shipping company: One of three anonymous shipping companies used. The shipment charges field values are based on this field's value. [9]

• shipping time: Time in days that it took the shipment to reach it's destination. [9]

• sender: The origin country. This was originally stored in a field called source country with a 2-letter abbreviation as a value, so I calculated this field using the Pandas map function to make it easier to understand. However, all of the values here are "Great Britain," so I later ended up dropping this field.

• receiver: The destination country. This was originally stored in a field called destination country with a 2-letter abbreviation as a value, so I calculated this field using the Pandas map function to make it easier to understand.

In the end, the independent variables I decided to use were gross weight, shipping time, shipping company, shipment mode, and receiver. The dependent variable that I was predicting was the total cost field.

4 Exploratory Data Analysis

Exploratory data analysis (EDA) is the process of investigating data to pull out some key insights, including which are the most important variables and whether or not any outliers exist within the data. EDA allows us to get a better understanding of relationships that may exist in our data before creating our models. It is essential in a data science project because it allows us to check our assumptions before getting to the more complicated model building process, hopefully saving us some time and headaches at that point. If we begin creating and testing our model but don't understand our data, we may consistently get poor results or have to repeat our steps, whereas completing EDA should minimize any redundancies later in the process thanks to a better understanding of the data. Additionally, using EDA, we can spot any errors or outliers in the data and remove those before we get to the next steps. EDA allows us to perfect our data cleaning performed before, as we better understand our data and any flaws that may exist.

Univariate EDA, either graphical or non-graphical (meaning with or without visuals, respectively) is where we examine one variable by itself [8]. I performed univariate graphical analysis by creating histograms of each of my variables to better understand the distribution of each variable.

Multivariate EDA, either graphical or non-graphical, means that we examine multiple variables [8]. I created a few scatter plots to better understand the relationship between some key variables. I also created a correlation matrix to visualize which variables had strong relationships to one another.

For my univariate graphical analysis, I examined each variable that had more than one value on it's own histogram, created using Plotly. This allowed me to see whether or not data was skewed in any particular way. I found that most of the data was skewed to the left but with very long tails. I also created box plots with Plotly and found that a lot of my variables had several outliers.

For my multivariate graphical analysis, I created a correlation matrix using Pandas' corr function and Plotly Figure Factory's annotated heat map function, which allowed me to see the relationship between variables. The general result was that there were two very strongly related variables, total cost and gross weight, and the rest were not strongly correlated. I also created Plotly scatter plots, which allowed me to get a better understanding of how three variables related to one another thanks to an x-axis, a y-axis, and the ability to color points on the scatter plot.

Some interesting takeaways were that the gross weight of shipments in my data set were somewhat normally distributed except for having a very long tail, most shipments cost 100,000 dollars or less, air shipments were more common than ocean shipments, and most shipments took less than 6 days to reach their destination. Using multivariate graphical analysis, I found that the total cost of shipments was almost completely correlated to gross weight. Shipping cost had the next strongest correlation to total cost, but it was a weak correlation.

```
px.histogram(df, x='freight_cost')
```

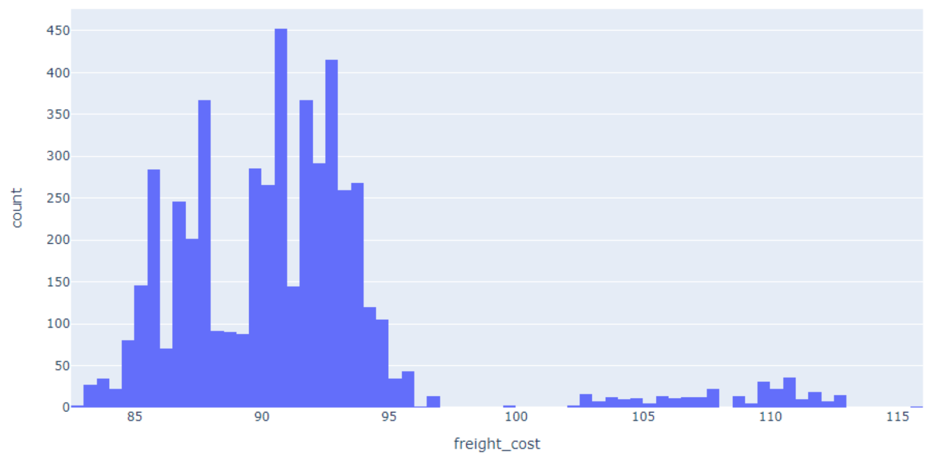


Fig. 1. Freight cost distribution

```
px.histogram(df, x='gross_weight')
```

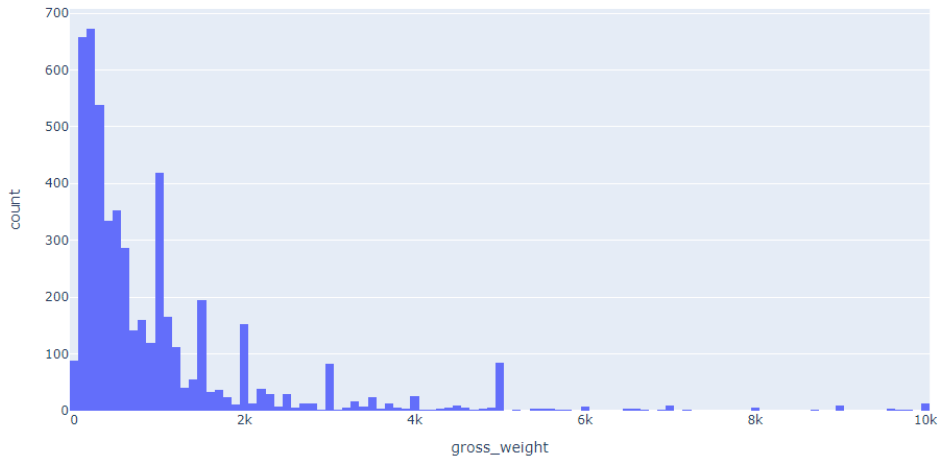


Fig. 2. Gross weight distribution

```
px.histogram(df, x='shipment_charges')
```

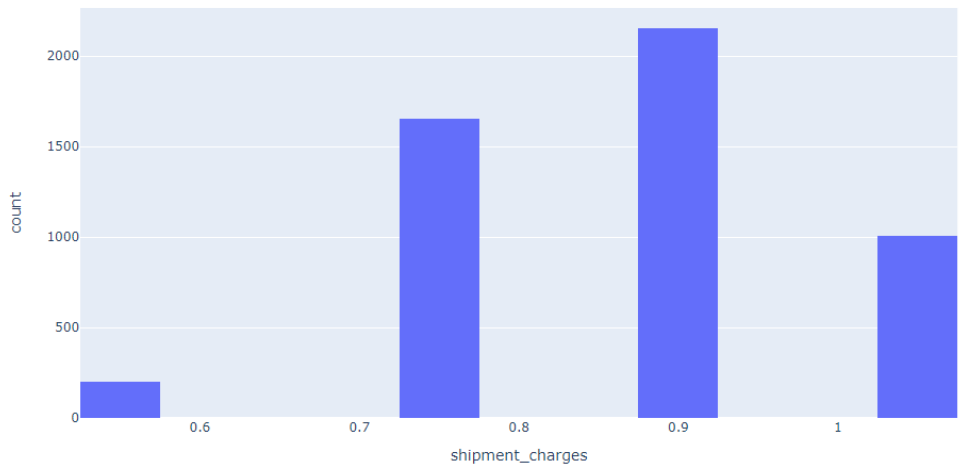


Fig. 3. Shipment charges distribution

```
px.histogram(df, x='total_cost')
```

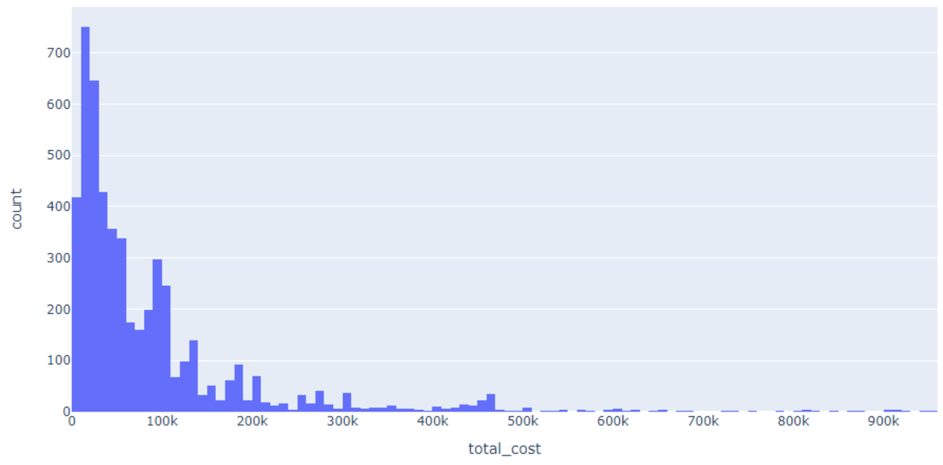


Fig. 4. Total cost distribution

```
px.histogram(df, x='shipment_mode')
```

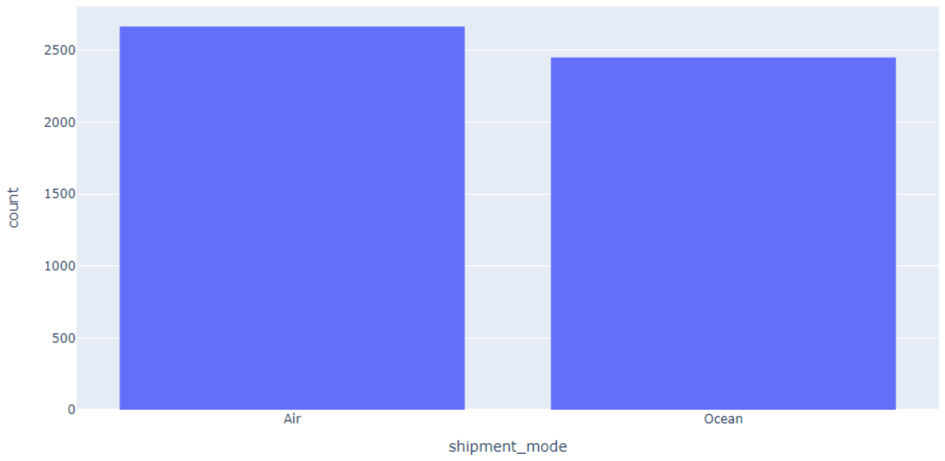


Fig. 5. Shipment mode distribution

```
px.histogram(df, x='shipping_company')
```

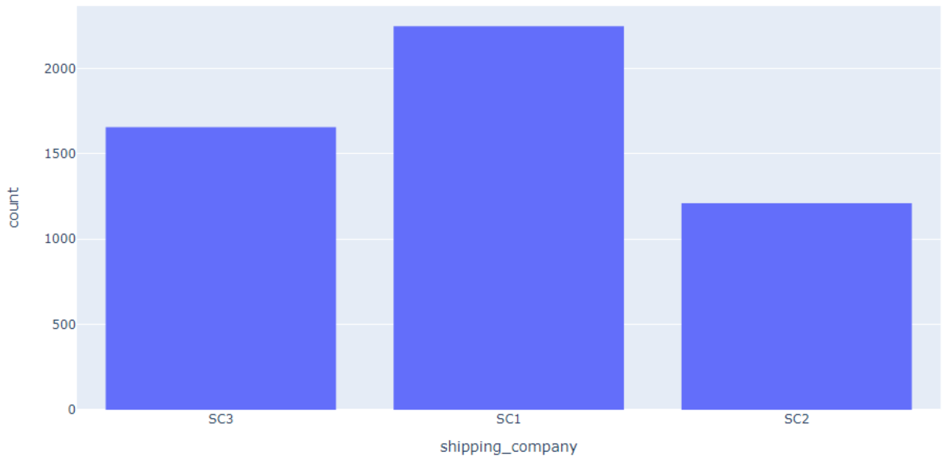


Fig. 6. Shipping company distribution

```
px.histogram(df, x='shipping_time')
```

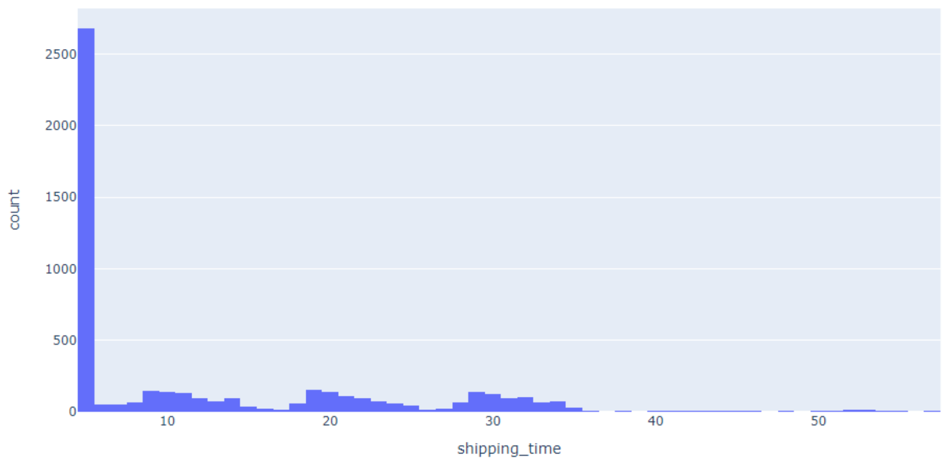


Fig. 7. Shipping time distribution

```
px.histogram(df, x='receiver')
```

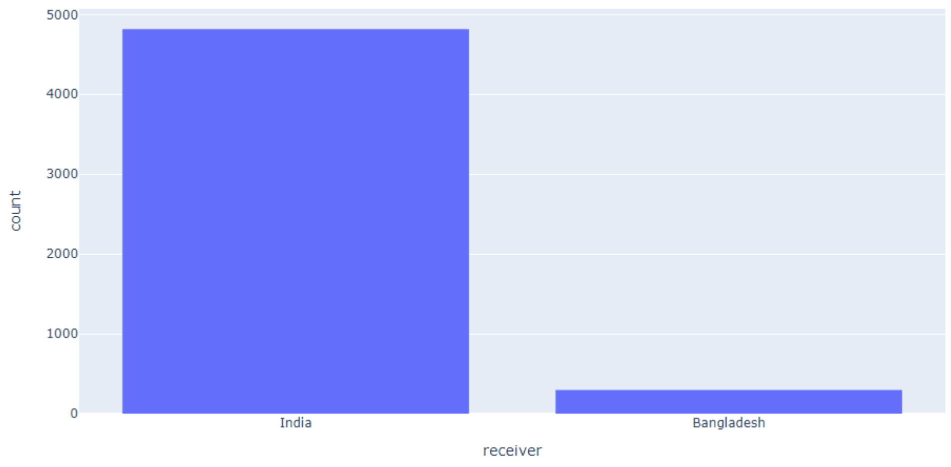


Fig. 8. Receiver distribution

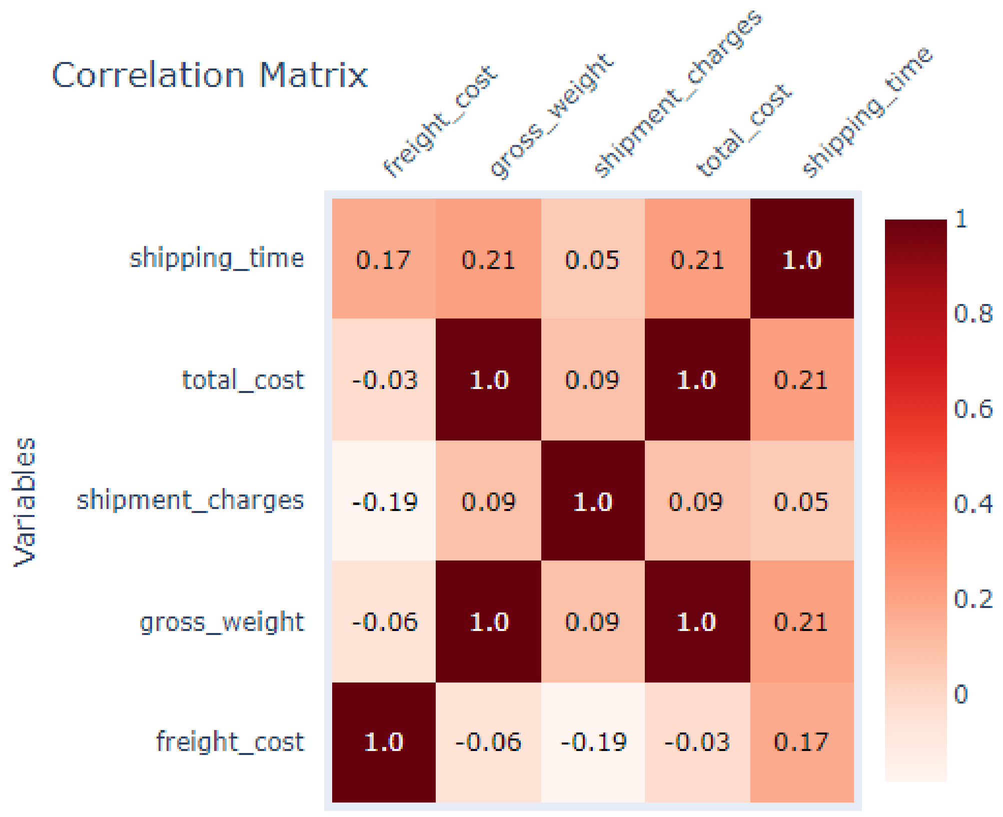


Fig. 9. Correlation matrix

Using box plots (a form of univariate graphical analysis), I found that gross weight, total cost, and shipping time all had a lot of outliers, which was also apparent in the histograms. However, the histograms still looked somewhat normally distributed to me, so I figured that for the time being, it was okay to keep the outliers, as they may be relevant data points. However, when testing my regression model later, the results were suspicious, so I had to come back and clean the data a bit more by removing these outliers.

```
px.box(df, y='total_cost', title='total_cost')
```

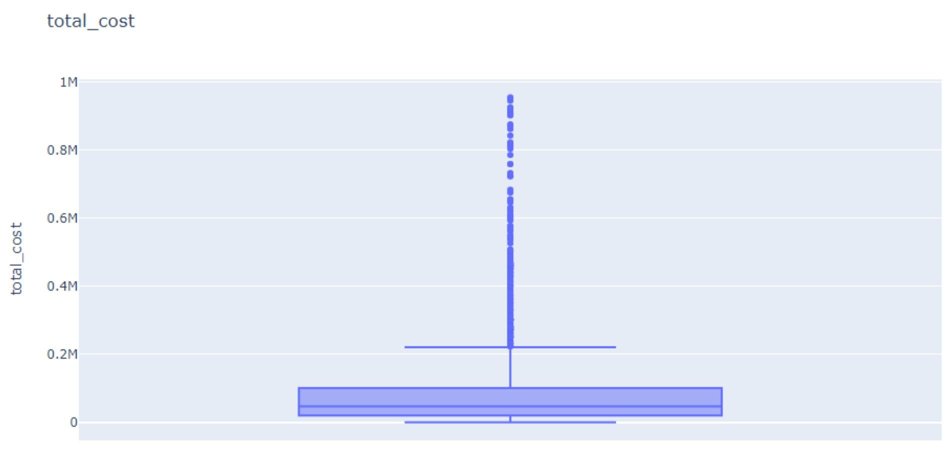


Fig. 10. Total cost box plot

```
px.box(df, y='gross_weight', title='gross_weight')
```

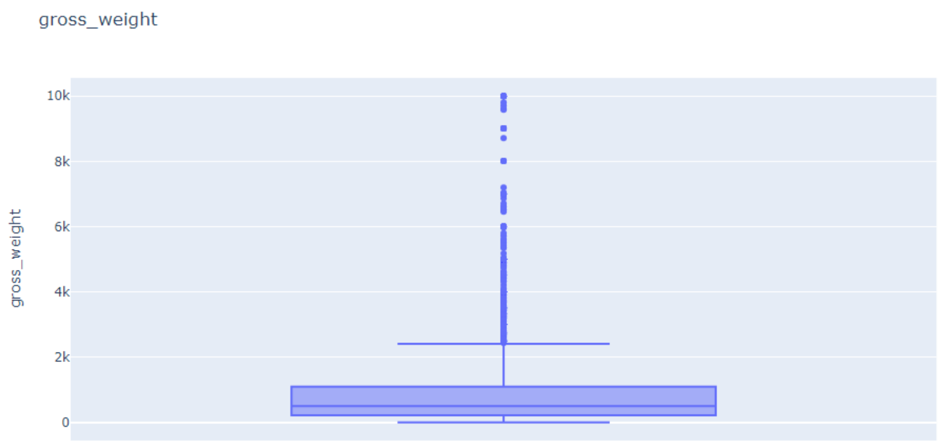


Fig. 11. Gross weight box plot

```
px.box(df, y='shipping_time', title='shipping_time')
```

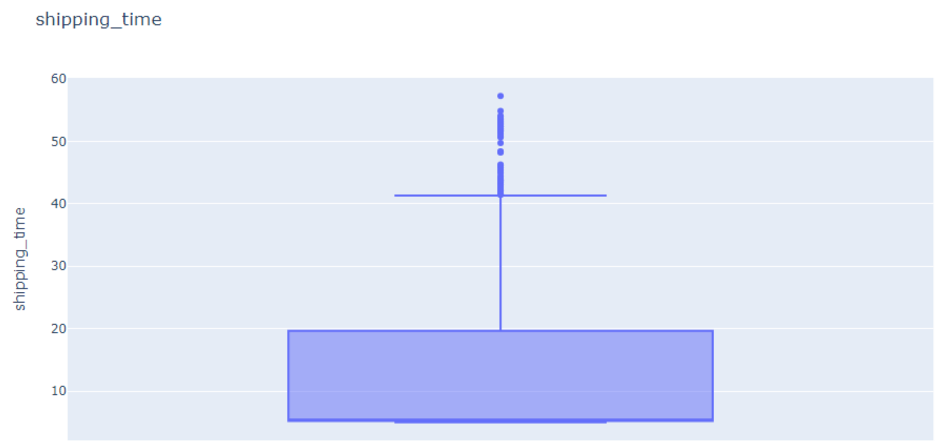


Fig. 12. Shipping time box plot

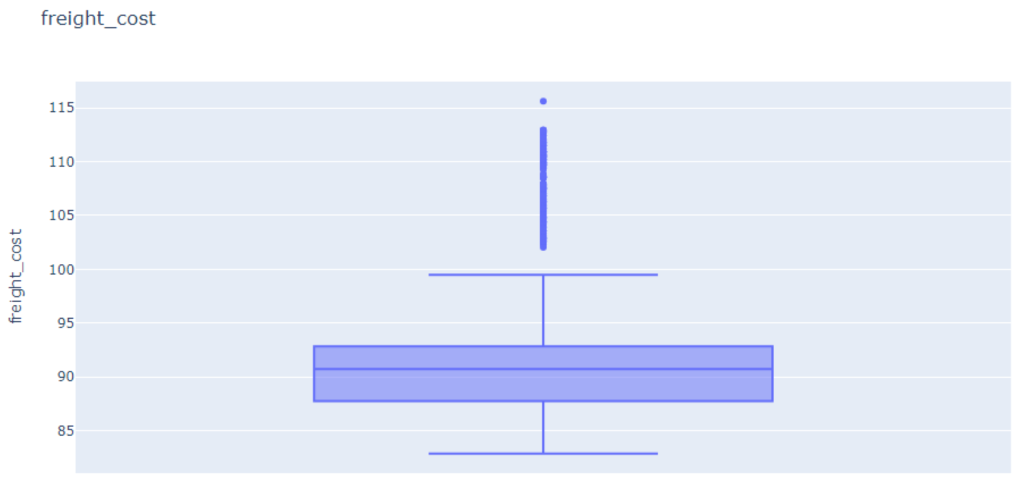
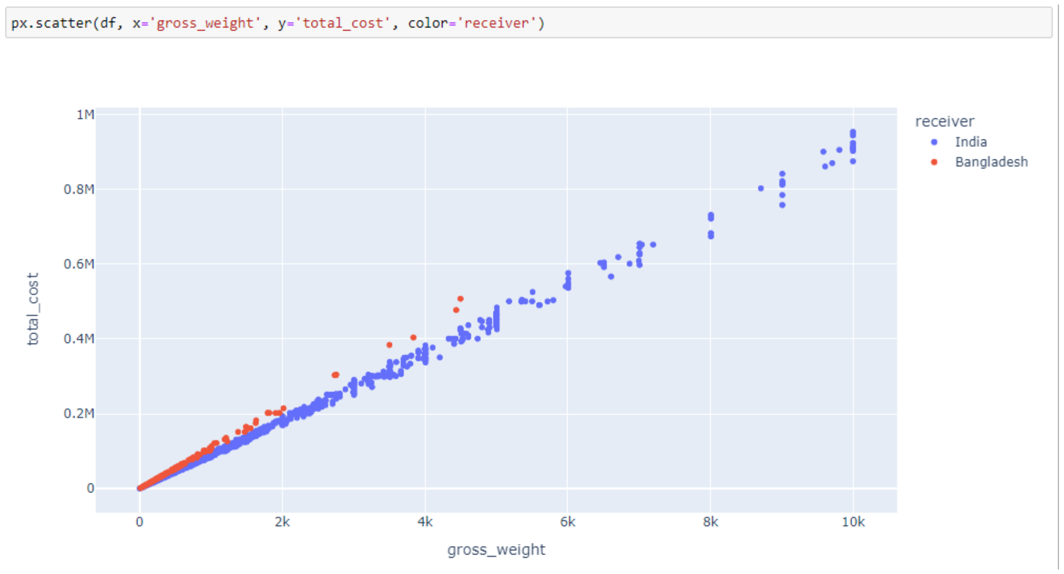


Fig. 13. Freight cost box plot

I created scatter plots (a form of multivariate graphical analysis) and saw that of the two receiver countries, India received the highest cost and weight of shipments, the most expensive and heaviest shipments were shipped through ocean shipping as opposed to air, and anonymous shipping company SC1 handled the heaviest and most expensive shipments.



```
px.scatter(df, x='gross_weight', y='total_cost', color='receiver')
```

Fig. 14. Comparison of gross weight, total cost, and receiver

```
px.scatter(df, x='gross_weight', y='total_cost', color='shipment_mode')
```

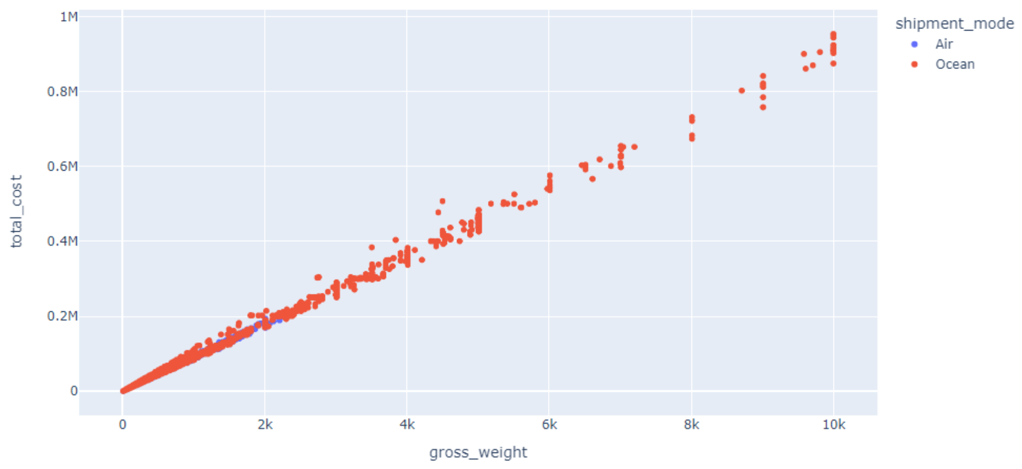


Fig. 15. Comparison of gross weight, total cost, and shipment mode

```
px.scatter(df, x='gross_weight', y='total_cost', color='shipping_company')
```

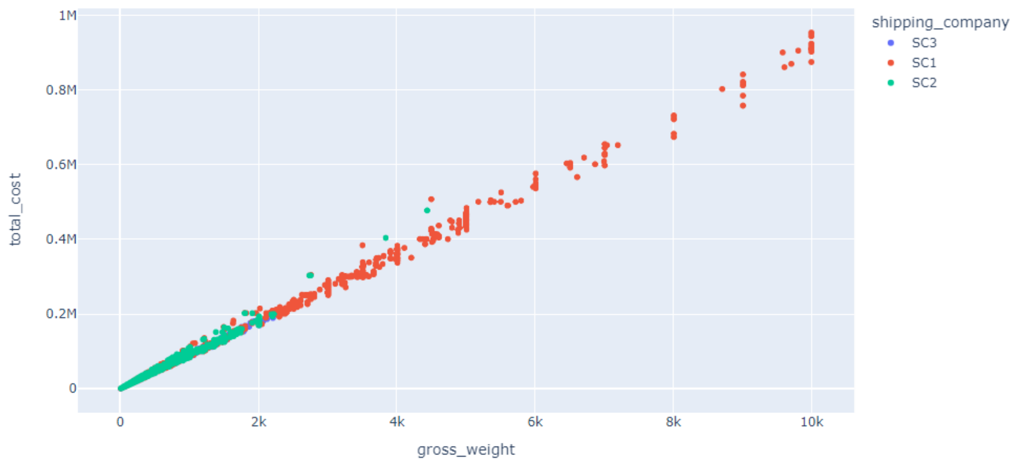


Fig. 16. Comparison of gross weight, total cost, and shipping company

Overall, I learned that the most important independent variables would likely be gross weight, shipping time, shipping company, receiver and shipment mode. Total cost would remain the dependent variable. I would have to clean my data a bit more to drop some unnecessary columns, and I planned to possibly later remove outliers if they became a problem in my model testing.

5 Predictive Analysis

The pipeline to build my predictive model was as follows (some of this is a recap of previous sections):

• Data Collection, where I had to find a suitable data set from the web.

• Data Processing and Cleaning, where some feature engineering occurred and features were selected as relevant, possibly manipulated, turned into new features, or dropped.

• Exploratory Data Analysis occurred and meant that I took a more in depth look at the data using univariate and multivariate graphical analysis, and from here, a bit more data cleaning happened. Relationships between variables were also explored.

• Then, my data was split into training and test data using Sklearn's model selection train test split function [7].

• Next, I fit my model and evaluated the results.

• After this, I cleaned the data a bit more and re-fit the model and re-evaluated the results. This process was repeated a few times until I had a result that I was happy with.

• Finally, I created some visualizations with Plotly to best understand the performance of the models.

I used a linear regression model, which is a simple supervised learning algorithm that tries to find the line of best fit (the regression line) between input variable(s) and a target variable. This can work well with one or two input variables, but I learned that as you add more input variables, the results can be harder to interpret. I also used a random forest regressor model, which is another supervised learning algorithm. This one is based on decision trees. I chose to use this model as well because I was sensing some over-fitting occurring with my regression model, and a random forest regressor model can help to alleviate that, or at least better understand the relationship between variables [7].

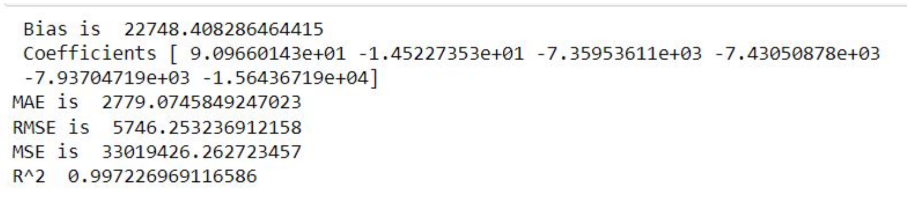
To train and test my models, I used Sklearn's model selection train test split function to split my data into train and test sets, with a test size of 0.2, meaning that 20 percent of my data was set aside for testing, leaving 80 percent of it for training. After fitting and evaluating my model a couple of times, I decided to try and add polynomial features instead to see if my results changed, as the r2 score was suspiciously high at over 0.99. I used Sklearn.preprocessing's PolynomialFeatures function to do this, with biases excluded and interaction only set to True. Then I used fit transform to fit the polynomials to my input features, and the results of that were used in one of my model's.

The overall implementation and evaluation process of my analysis was as follows:

• Before splitting for training or testing, I created dummy variables to account for categorical variables that I wanted to use as inputs in my models. I used Pandas get dummies function and set drop first to True so as to avoid the "dummy variable trap." According to the blog Algosome, the dummy variable trap occurs when some of the independent variables are co-linear, meaning that they are correlated and can explain one another. [2]

• Then, I split my model into train and test data as explained above, but without polynomials.

• Next, I fit the linear regression model with the training data. To evaluate the data, I examined the mean absolute error, the root mean squared error, the mean squared error, and the r2 score, as well as the coefficients. I noticed that my r2 score was very high at over 0.99, and my coefficients were all over the place, some in ways that didn't make sense.



```
 Bias is  22748.408286464415
 Coefficients [ 9.09660143e+01 -1.45227353e+01 -7.35953611e+03 -7.43050878e+03
 -7.93704719e+03 -1.56436719e+04]
MAE is  2779.0745849247023
RMSE is  5746.253236912158
MSE is  33019426.262723457
R^2  0.997226969116586
```

Fig. 17. Results of first linear regression model run

• I then created some line plots (for numeric variables) and box plots (for the previously string-type dummy variables) and saw some pretty odd and hard to interpret results. To understand the box plots for the dummy variables, it is important to understand that for all possible dummy variables, one of the original values is left out to evaluate against. So, in the case of shipping company, which had 3 original values, only 2 new variables were created, with a result of either 0 (false) or 1 (true). In the case of the box plot below, 0 are instances where the shipping company is not SC2, and 1 are cases where the shipping company is SC2. Box plots were chosen for the dummy variables due to the fact that only 0 or 1 would be displayed, leading to messy plots. The takeaway here was that there a lot of outliers.

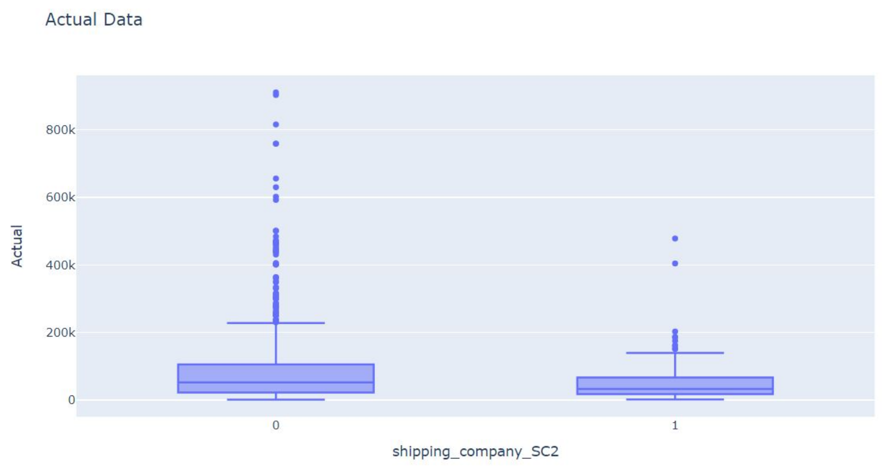


Fig. 18. Results of first linear regression model run - Shipping Company 2 Dummy Variable Box Plot

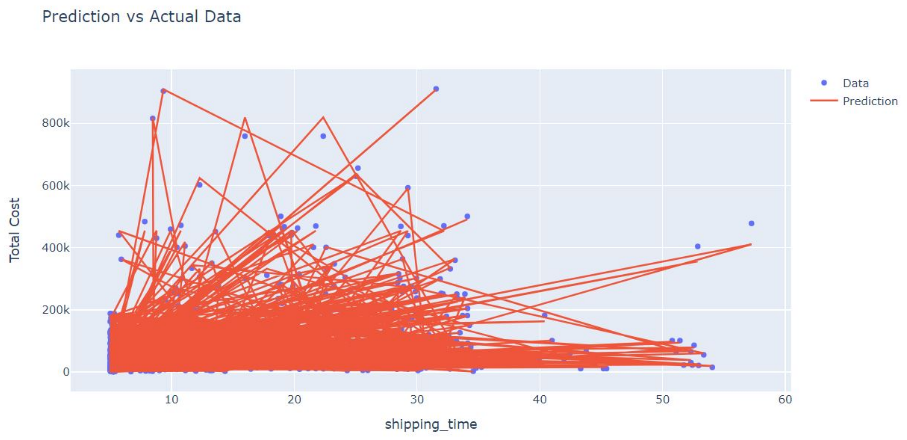
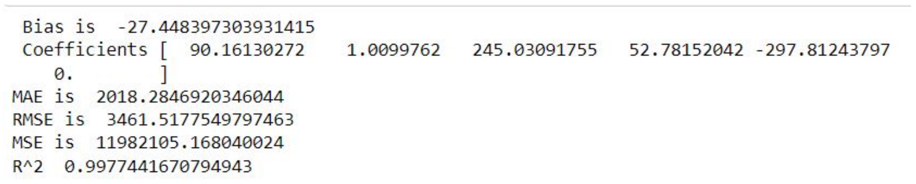


Fig. 19. Results of first linear regression model run - Shipping Time vs Total Cost

• Given the odd results, I performed some more data cleaning by removing outliers from my data using Numpy's abs function [3] and Scipy's stats' zscore function [5].

• Then, I split my data again and refit the linear regression model. I noticed that the values I evaluated previously had changed drastically for some variables, but not very much for other variables. Some of the coefficients simply didn't make sense given the nature of the data. For example, the coefficient of shipping time (one of my input features) on the target feature (total cost) was 1.009, meaning that for every 1 unit of movement for my target feature, the shipping time increased by 1.009. This did not make much sense though given that the relationship between these variables did not seem very strong before.

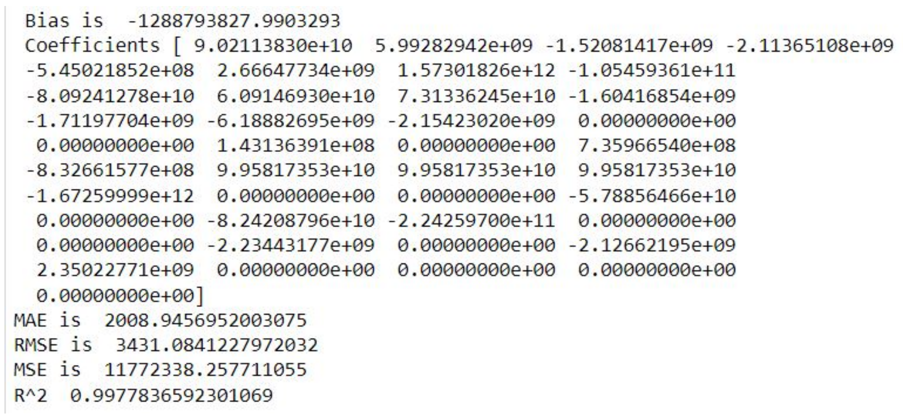


```
 Bias is  -27.448397303931415
 Coefficients [  90.16130272    1.0099762   245.03091755   52.78152042 -297.81243797
    0.        ]
MAE is  2018.2846920346044
RMSE is  3461.5177549797463
MSE is  11982105.168040024
R^2  0.9977441670794943
```

Fig. 20. Results of second linear regression model run

• This is when I created polynomial features and transformed my input variables using them [7].

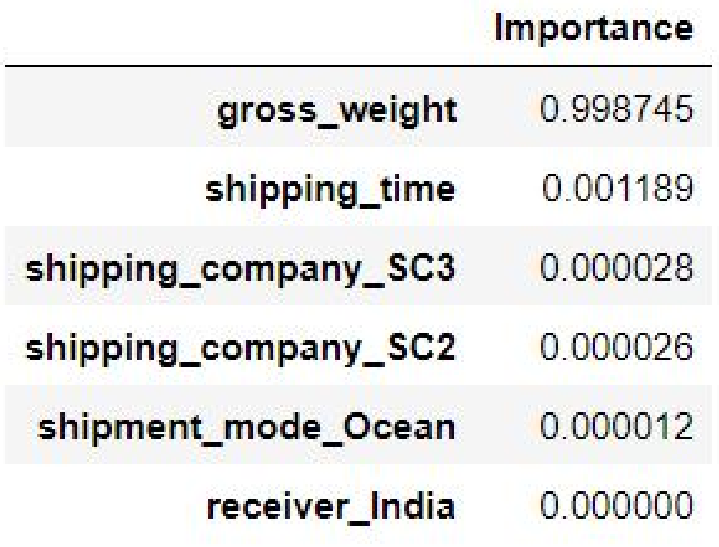
• Then, I refit my linear regression model again and evaluated the same metrics as before. I saw that the values I evaluated previously had not changed much from my first run of the model, so I knew I needed to take another approach.



```
 Bias is  -1288793827.9903293
 Coefficients [ 9.02113830e+10  5.99282942e+09 -1.52081417e+09 -2.11365108e+09
 -5.45021852e+08  2.66647734e+09  1.57301826e+12 -1.05459361e+11
 -8.09241278e+10  6.09146930e+10  7.31336245e+10 -1.60416854e+09
 -1.71197704e+09 -6.18882695e+09 -2.15423020e+09  0.00000000e+00
  0.00000000e+00  1.43136391e+08  0.00000000e+00  7.35966540e+08
 -8.32661577e+08  9.95817353e+10  9.95817353e+10  9.95817353e+10
 -1.67259999e+12  0.00000000e+00  0.00000000e+00 -5.78856466e+10
  0.00000000e+00 -8.24208796e+10 -2.24259700e+11  0.00000000e+00
  0.00000000e+00 -2.23443177e+09  0.00000000e+00 -2.12662195e+09
  2.35022771e+09  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00]
MAE is  2008.9456952003075
RMSE is  3431.0841227972032
MSE is  11772338.257711055
R^2  0.9977836592301069
```

Fig. 21. Results of third linear regression model run with polynomial features added

• Finally, I decided to try a random forest regressor, as I was suspecting some over-fitting was occurring from the linear regression models. I split the data into train and test sets again and fit the random forest regressor model. The results were similar, but I was able to visualize the importance of each feature, and I found that one feature, gross weight, had an importance score of 0.99, while the rest all had less than 0.01 importance. The r2 score still remained high at 0.99, but it now made sense why it was so high.



	Importance
gross_weight	0.998745
shipping_time	0.001189
shipping_company_SC3	0.000028
shipping_company_SC2	0.000026
shipment_mode_Ocean	0.000012
receiver_India	0.000000

Fig. 22. Importance of each input variable in the random forest regressor

• Learning about the importance, I was able to determine that one input, gross weight, was vastly more important than the rest or the inputs, and it predicted the target feature so well that the low importance of the other features did not matter.

From my analysis, I learned that only one of my input features was relevant in predicting the target feature. Essentially, gross weight was far more important in predicting total cost than any other input feature. This was interesting to me, as I thought the receiver county, transport mode, or shipping company would be important here, but they were not.

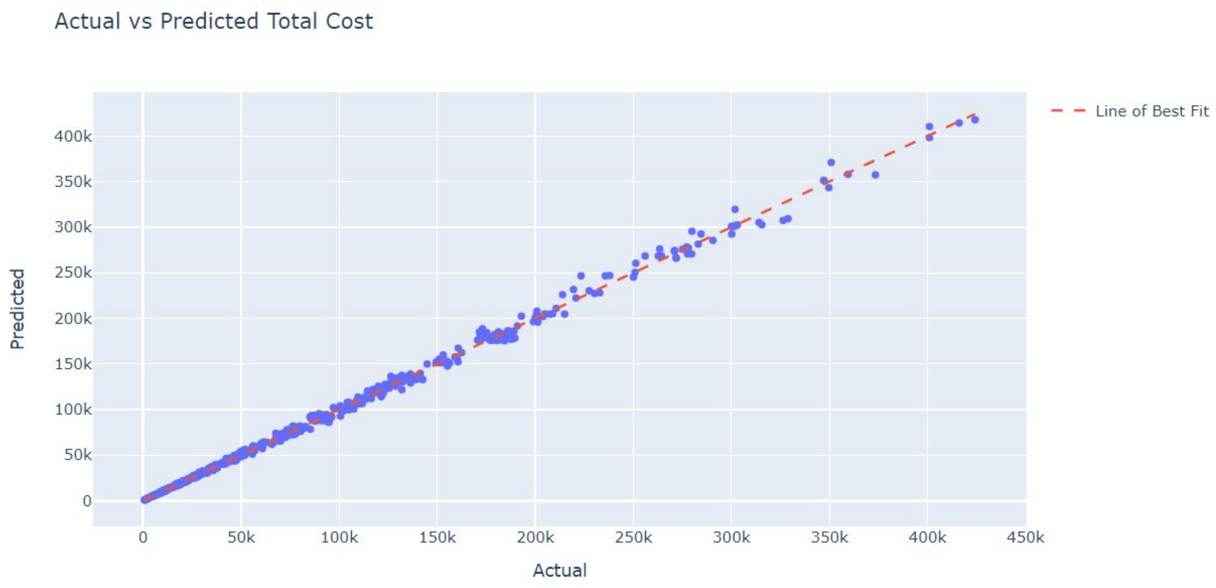


Fig. 23. Actual vs. Predicted Cost of the Total Cost of Shipments

6 Interpretation of Results

The resulting plots from this study included a scatter plot showing the actual values of my target feature (total cost) on the x-axis and the predicted cost of feature on the y-axis, which helped me to best communicate my findings in the section above. Additionally, a table and bar chart showing the importance of each feature helped to visualize the relative importance of each feature on the random forest regressor model. In previous modules, charts that were helpful to explore the relationship between data variables included scatter plots, box plots, and a heat map to explore correlation.

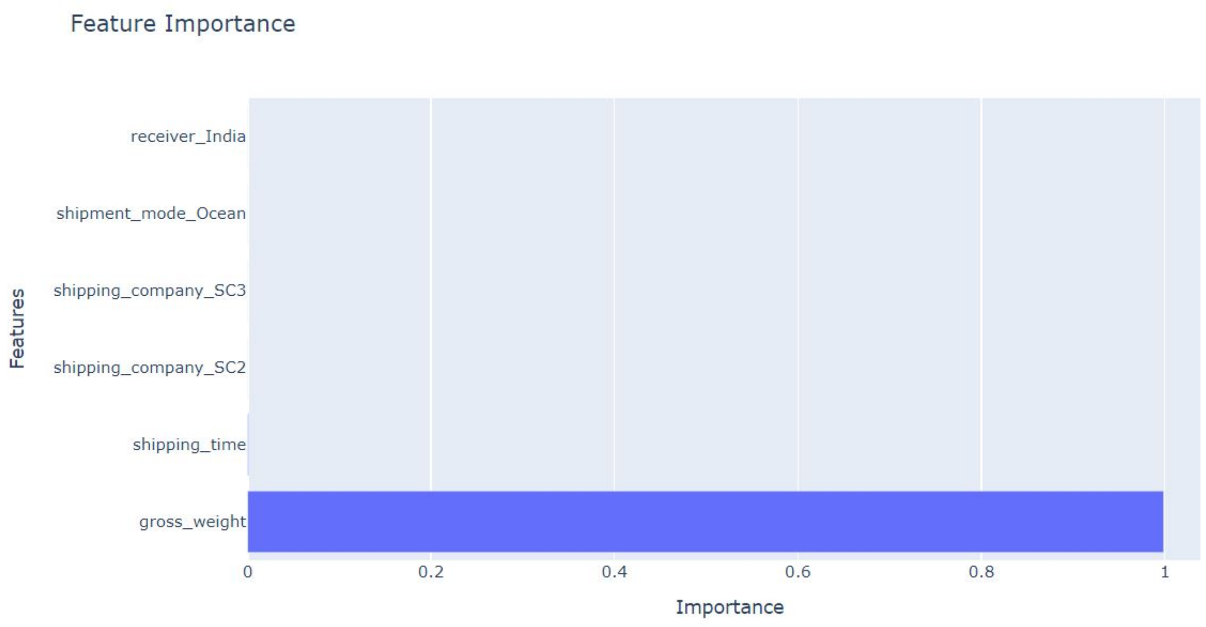


Fig. 24. Feature importance from the random forest regressor model

Using the bar chart of feature importance, I was able to see that only one independent variable, gross weight, was remotely important in predicting the total cost of a shipment, as it's bar dwarfed the bars of the other variables to the point that you couldn't even see them. Additionally, the scatter plot in figure 23, showing actual vs predicted values, displays the strength of the random forest regressor model at predicting accurate values.

The data lacked significant features, other than gross weight, to predict the total cost of a shipment. However, this does not necessarily mean that there are no other important features that could have influenced the total cost of a shipment. There could have been other data points that simply weren't tracked in this data set that had more importance in predicting the target. Gross weight was an important enough feature in the prediction of total cost that it carried several models by itself, meaning that the model performed well despite the noise of other input features.

As stated above, gross weight was the only important feature present in this data set to use when predicting the total cost of a shipment. That feature had an importance score 0.998745, while the others had scores of nearly or less than 0.001. The other input variables, such as shipping time, shipping company, and receiver, had little to no influence on the total cost of a shipment.

It was interesting to me that after gross weight, the next most important variable was shipping time, with a score of roughly 0.001. I thought shipping time would be much more important, but in the end, it held very little weight in this study using this data set. Additionally, one of my dummy variables, receiver India (referring to instances where the country receiving the shipment from Great Britain was India over Bangladesh), was 0. This surprised me, as I thought the country receiving the shipment would have a lot to do with the total cost of the shipment, but it turns out that the country receiving the package had no impact on the total cost of the shipment.

7 Limitations

Although running the linear regression and random forest regressor models displayed the strength of the relationship between gross weight and total shipment costs, the original data set, models, and results are not without their limitations.

For starters, the data set was taken from Kaggle, a site with user submitted data for various purposes. In this case, the data was uploaded for use in a competition [9]. This means that the user who uploaded the data could have picked and chosen the variables to keep, and other important variables could have been left out. Additionally, the scope of this data is rather small, containing information for only two shipment modes (air and ocean), three unnamed shipping companies, one origin country (Great Britain) and two receiver countries that are relatively close to each other when examined at a global scale (India and Bangladesh). Having more specific location information, more companies, more transport modes, information about what was being transported, and more information about the breakdown of charges could have helped these models to have much stronger predictive capabilities. I believe that with more of these data points included, I could have created models that better defined the relationships between some of these variables and total shipment costs.

These models do a good job of highlighting the influence of gross weight on total shipments costs, but they have not revealed much information that couldn't be inferred with some exploratory analysis. At most, the results can help to estimate the cost of a shipment based on weight alone, but not other factors.

8 Conclusions

The overall goal of this project was to predict the cost of shipments based on a number of input features. Using tools available in Pandas, Numpy, and Plotly Express, I was able to perform data cleaning and exploratory data analysis on a data set found on Kaggle to identify the most important features for my analysis. Then, Sklearn was used to create and evaluate linear regression models and a random forest regressor model. The results were a few models that very accurately predicated the total cost of shipments from Great Britain to India and Bangladesh, but the models were heavily carried by the most important input feature, gross weight. Even so, the results can help determine the cost to ship something along these routes with very high accuracy.

[]

References

1. Capstone github repo, https://github.com/HundredDucks/Capstone/tree/master
2. Dummy variable trap in regression models, https://www.algosome.com/articles/dummy-variable-trap-regression.html
3. Numpy user guide, https://numpy.org/doc/stable/user/index.htmluser
4. Plotly express in python, https://plotly.com/python/plotly-express/
5. Scipy user guide, https://docs.scipy.org/doc/scipy/tutorial/index.htmluser-guide
6. User guide, https://pandas.pydata.org/docs/user$_guide/index.html$
7. User guide, https://scikit-learn.org/stable/user$_guide.html$
8. What is exploratory data analysis?, https://www.ibm.com/topics/exploratory-data-analysis
9. GAUTAM, S.: [1] shipping optimization challenge, https://www.kaggle.com/datasets/salil007/1-shipping-optimization-challenge?select=train$_2pr.csv$
10. Pollack, N.: 3 ways to improve logistics management using shipping data, https://transimpact.com/nextsights/3-ways-to-improve-logistics-management-using-shipping-data/