Predicting Shipping Prices Using Machine Learning

Zachary J. Fuller

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

Abstract. Keywords: data analytics · data science · machine learning · logistics · shipping

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. Additionally, I am hopeful that what I learn from this study can be applied to the work I do in my professional life. My data will likely from Kaggle, as I have found a couple of decent data sources on there already. Google's Dataset search engine will also be helpful. If any data I find has tracking numbers associated with it, then UPS, FedEx, or the associated carrier's website should also provide some valuable data about the package. I would like to do some predictive analytics using previously shipped packages as a training set. Specifically, predicting the cost to ship a package based on multiple factors like weight, destination, origin, and distance traveled could likely be accomplished using a regression model. I feel that this could be useful 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 will follow to complete this project are:
1. I will first locate my dataset.
2. Then, I will clean the dataset using the python packages Pandas and NumPy, removing any drastic outliers if needed and deciding how to handle missing values.
3. Then, I will use SciKitLearn to separate the data into a train and test set. 4. I will evaluate the results. If needed, I will retrain and test the data, repeating steps 2 and 3.
5. After training and testing is complete, I will use a python plotting package, likely Plotly (my preferred choice) or Matplotlib to display the results.
6. Finally, I will write the report, making sure to examine any possible errors.

The key components of my approach will be finding the right data sources, making sure the data is cleaned in a way so as not skew results, and most importantly, the training and testing process. This will likely be done using multiple linear regression, so making sure that the data is trained correctly is key finding the best fit to account for each independent variable.

2 Methodology

3 Data Collection

My main data source was found on Kaggle [3]. I have also found some supplementary data on Data World [5] [4]. that can help to provide some general context for my data domain of shipping and logistics in general. The data is found in a few CSV files. I did not have to use any data scraping techniques, as my main data source and the supplementary data sources are made available for easy download through each of their host sites. From the "Shipping Optimization Challenge" dataset on Kaggle, I will use the sendtimestamp, sourcecountry, destinationcountry, freightcost, grossweight, and shipmentcharges fields. The dataset has thousands of columns, so I will likely need to fill in some null values. Additionally, I may slice off the timestamps and only use dates, as it appears that a lot of this data involves international shipping, and the timestamp may be less important and take up space/time. Finally, I will need to convert the country fields, as they are simply abbreviations at the moment. I will likely read the CSVs into a pandas dataframe in a Jupyter notebook, merge the necessary tables and fields together, and then perform this cleaning.

4 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 dataset for further analysis. In the case of my dataset, 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 missing columns, 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. I also used the python package plotly, specifically the module plotly express, as this is my prefered plotting package. The plotly.express function "histogram" was used to view the distribution of my data.

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 these 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 dataset:

• 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. [3]

• freight cost: The cost/kg of the shipment. [3]

• gross weight: The weight in kg of the shipment. [3]

• shipment charges: This one was confusing to me at first, but looking back through the documentation on Kaggle, I learned that this is the minimum amount that a company would charge for a specific shipment. I may not end up using this in my final analysis, but I figured I would keep it in here just in case I do. [3]

• 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. [3]

• shipping company: One of 3 anonymous shipping companies used. The shipment charges field values are based on this field's value. [3]

• shipping time: Time in days that it took the shipment to reach it's destination. [3]

• 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 map function to make it easier to understand. However, all of the values here are "Great Britain," so I may end 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 map function to make it easier to understand.

My independent variables will be gross weight, shipment mode, shipping time, and receiver. Other possible independent variables may be shipment charges, shipping company, and freight cost, although these will likely be unnecessary. My dependent variables that I am trying to predict is the total cost field.

5 Exploratory Data Analysis

Exploratory data analysis (EDA) is the process of investigating data to pull out some key insights, include 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 practice, 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 and 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. In my opinion, 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. 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. 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 annontated heatmap function, which allowed me to see the relationship between variables. The general result was that there were two very strongly related variables, 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 dataset 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 in this data set, 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 week correlation.

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

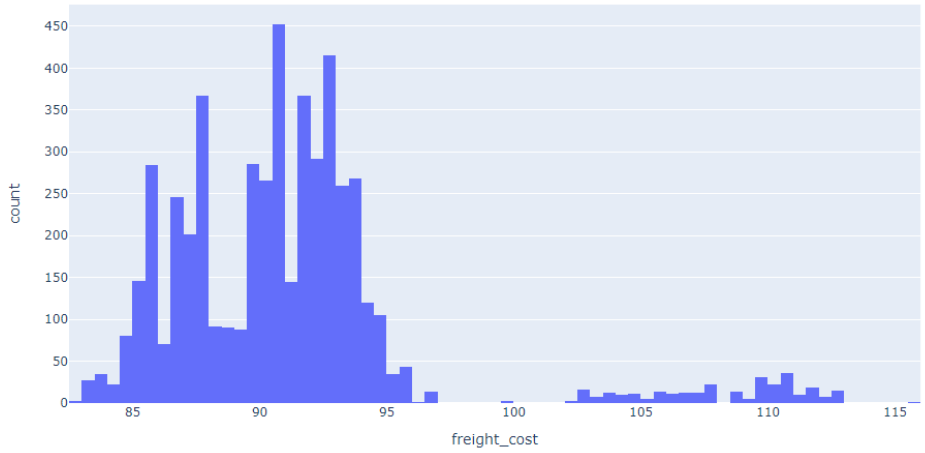

Fig. 1. Freight cost distribution

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

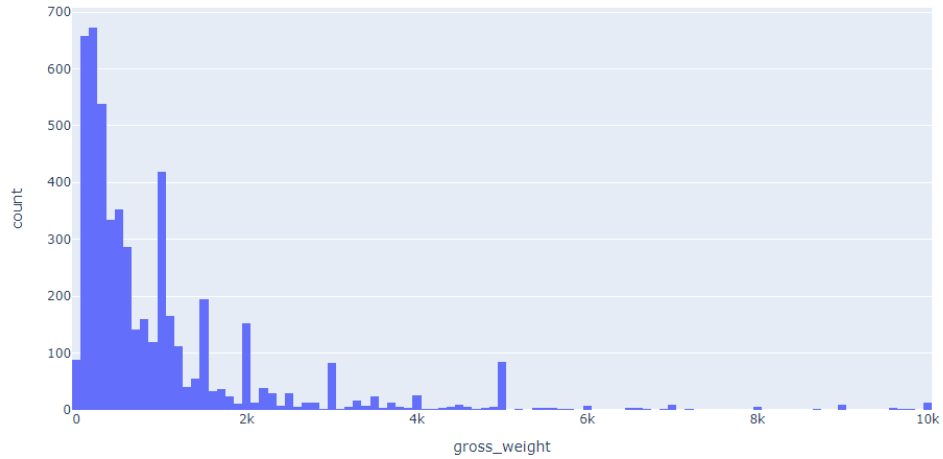

Fig. 2. Gross weight distribution

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

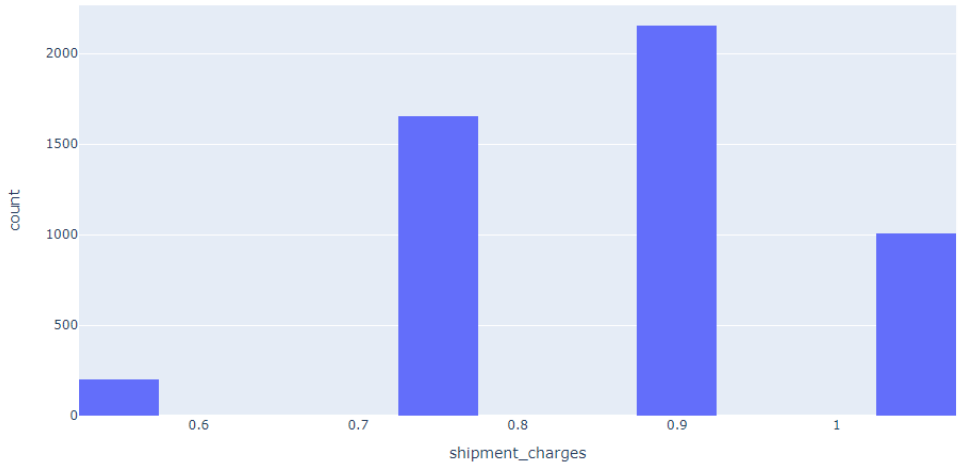

Fig. 3. Shipment charges distribution

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

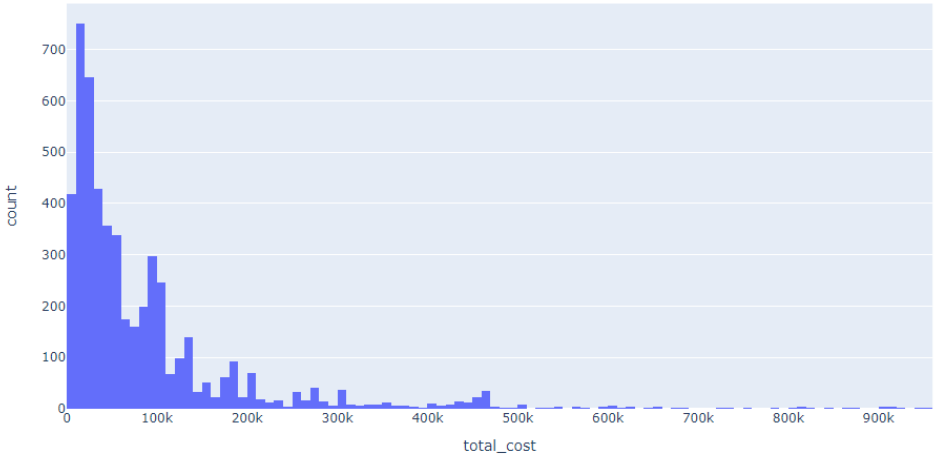

Fig. 4. Total cost distribution

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

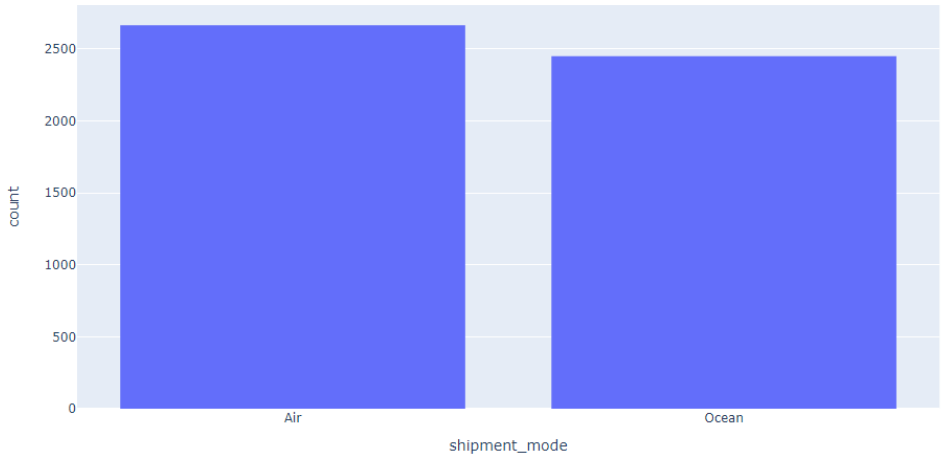

Fig. 5. Shipment mode distribution

```
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')
```

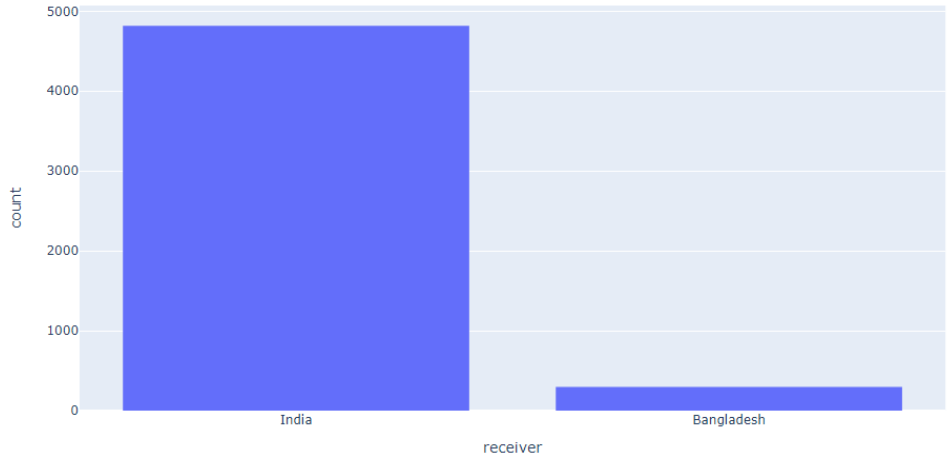

Fig. 8. Receiver distribution

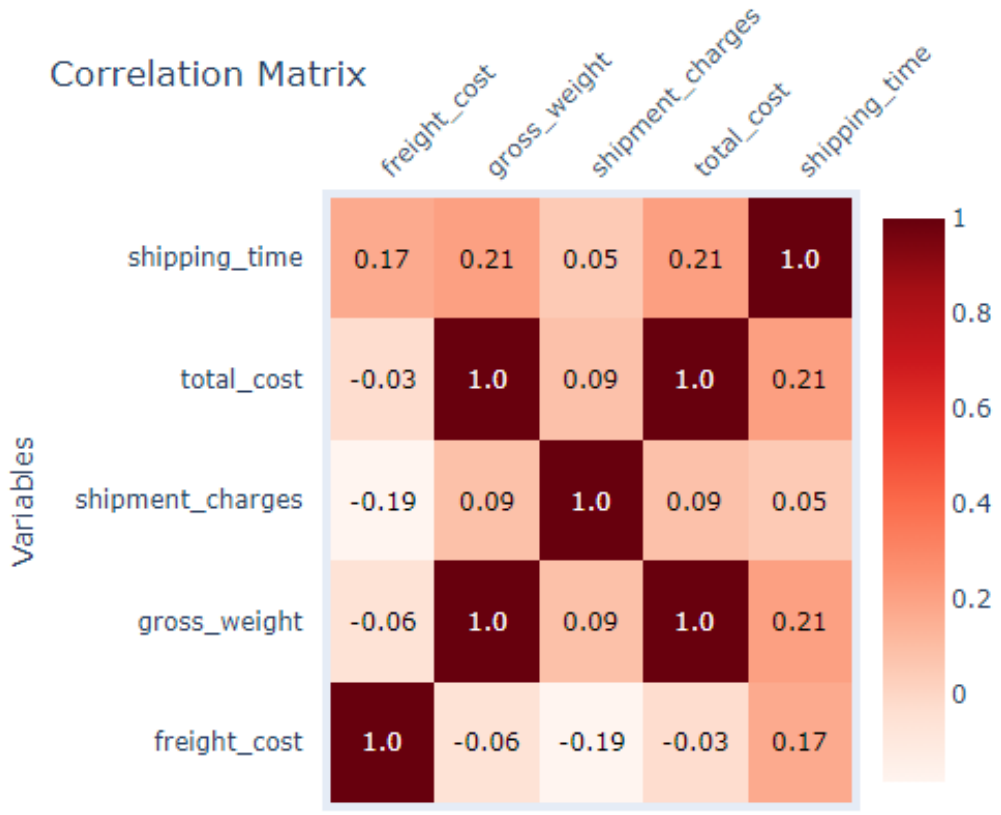

Fig. 9. Correlation matrix

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 is okay to keep the outliers, as they may be relevant data points. However, when testing my regression model later, if the results are poor, I may have to come back and clean the data some more by removing these outliers.

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

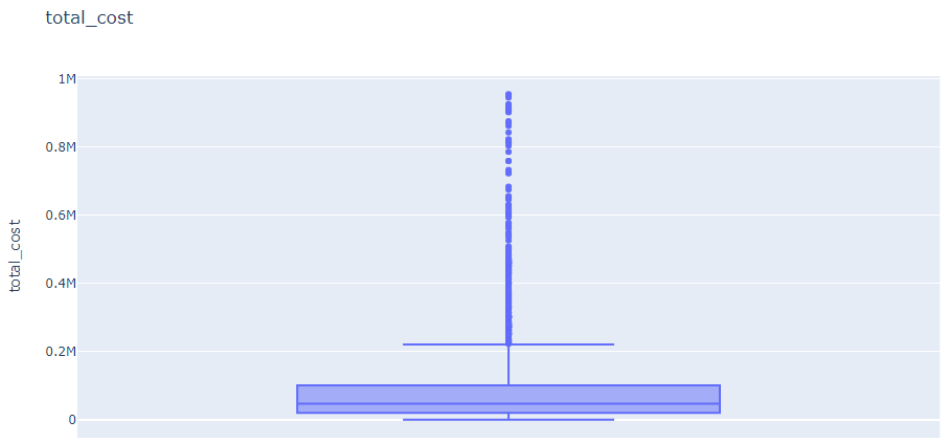

Fig. 10. Total cost box plot

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

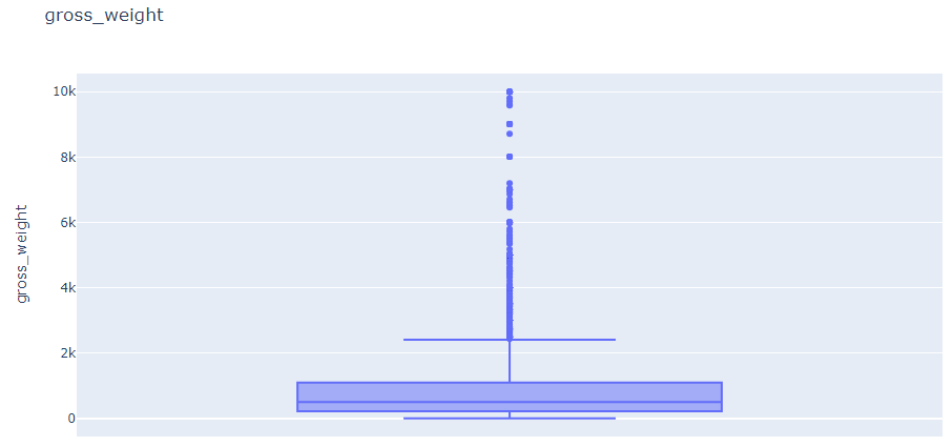

Fig. 11. Gross weight box plot

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

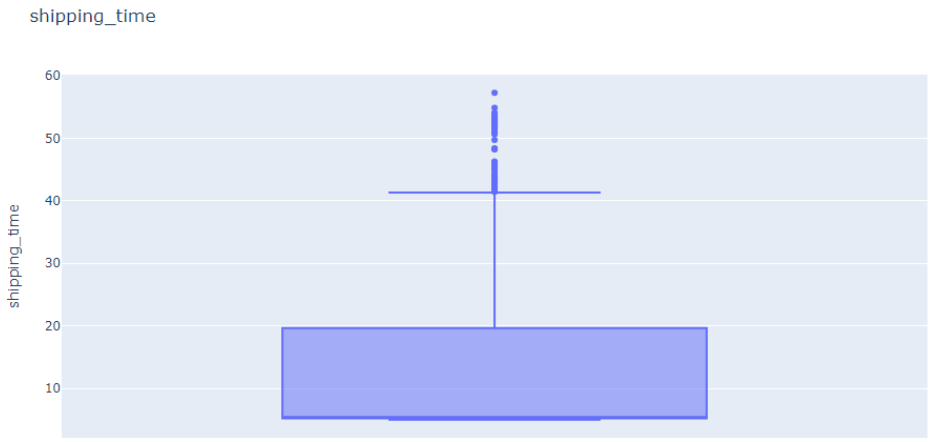

Fig. 12. Shipping time box plot

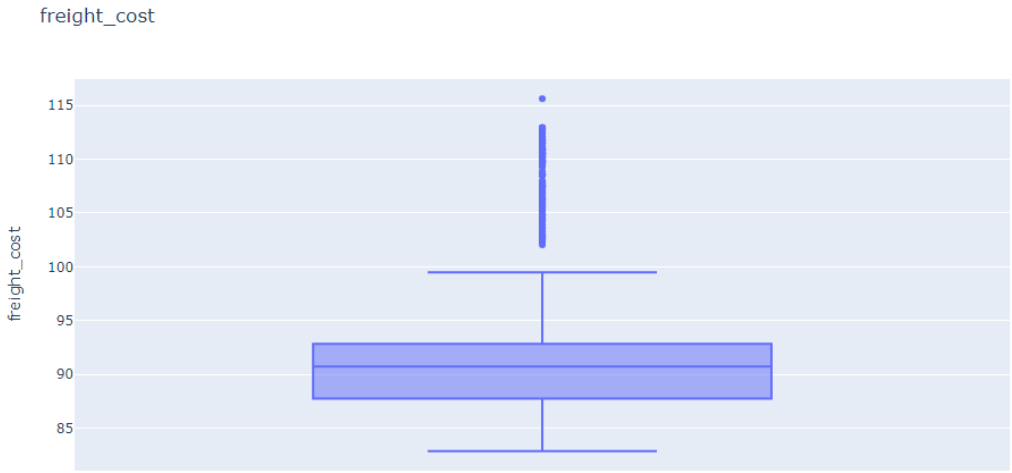

Fig. 13. Freight cost box plot

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 one handled the heaviest and most expensive shipments.

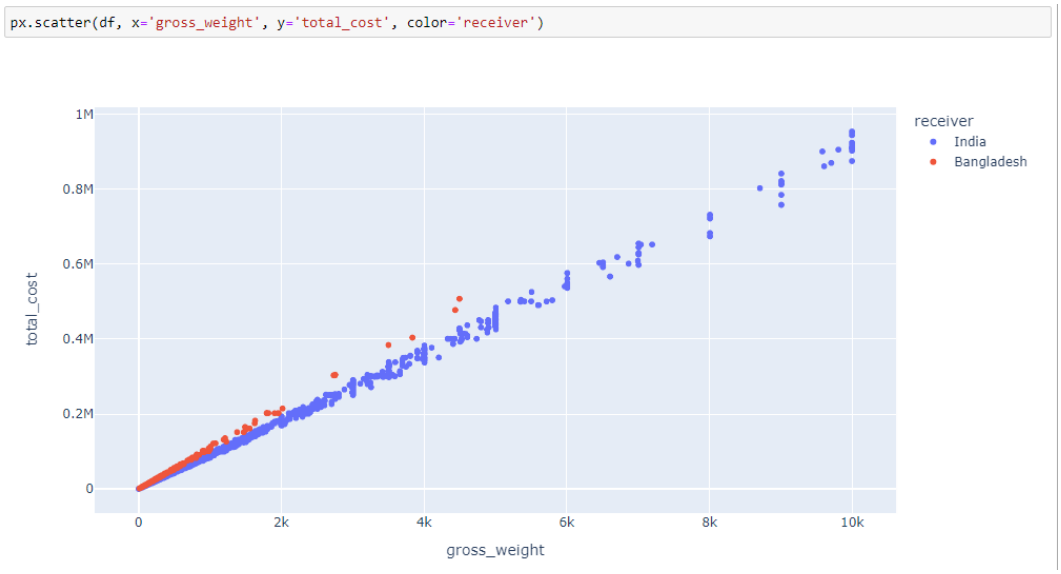

Fig. 14. Comparison of gross weight, total cost, and receiver

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

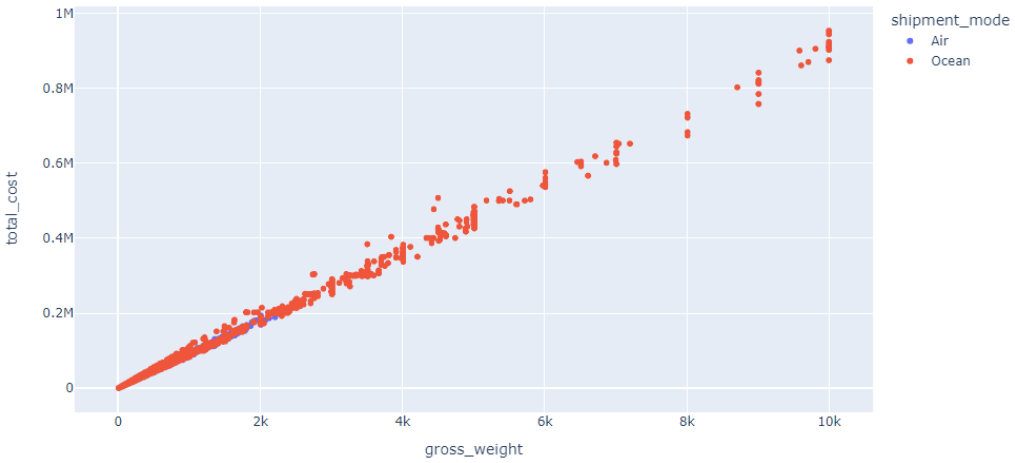

Fig. 15. Comparison of gross weight, total cost, and shipment mode

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

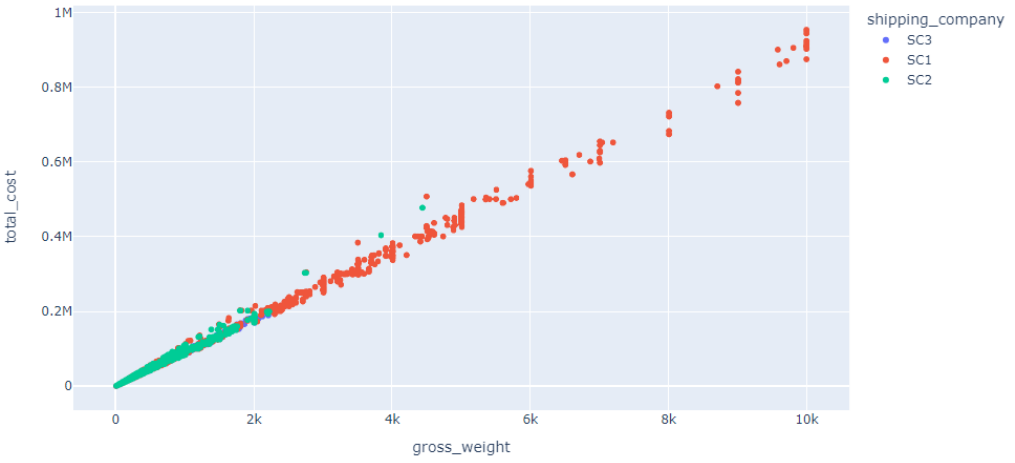

Fig. 16. Comparison of gross weight, total cost, and shipping company

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

6 Predictive Analysis

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

- Data Collection, where I had to find a suitable dataset 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.
- 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 model.

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 overfitting 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.

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. 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.

• I then created some line plots (for numeric variables) and box plots (for the previously string dummy variables) and saw some pretty odd and hard to interpret results.

• Given the odd results, I performed some more data cleaning by removing outliers from my data.

• 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.

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

• 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.

• Finally, I decided to try a random forest regressor, as I was suspecting some overfitting 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.

• 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.

	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. 17. Importance of each input variable in the random forest regressor

Fig. 18. Actual vs. Predicted Cost of the Total Cost of Shipments

7 Limitations

8 Conclusions

[]

References

1. Capstone github repor, 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. GAUTAM, S.: [1] shipping optimization challenge, https://www.kaggle.com/datasets/salil007/1-shipping-optimization-challenge?select=train$_{2}p$r.csv
4. Griffith, B.: Ups$_{d}aily_{r}ates.xlsx, https : //data.world/siyeh/crm - project/workspace/file?filename = UPS_{d}aily_{r}ates.xlsx$
5. Hoov, G.: How freight moves, https://data.world/garyhoov/how-freight-moves
6. Pollack, N.: 3 ways to improve logistics management using shipping data, https://transimpact.com/nextsights/3-ways-to-improve-logistics-management-using-shipping-data/