1. **(2 points) What mechanism or a pipeline is built to accomplish a desired predictive application? You may use a diagram to explain your mechanism.**
   * The pipeline to build my predictive model was as follows (some of this will be a recap of previous modules):
     + 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 looking 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 module.
     + 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 to best understand the performance of the model.
2. **(2 points) What are machine learning algorithms used to analyze your data?**
   * 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.
3. **(2 points) Explain the training and testing process related to your data?**
   * I used sklearn.model\_selection's train\_test\_split function to split my data into train and test sets, with a test size of 0.2, meaning that 20% of my data was set aside for testing, leaving 80% 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 interation\_only set to True. Then I fit used fit\_transform to to fit the polynomials to my input features, and the results of that were used in one of my model's.
4. **(2 points) Explain the implementation and evaluation process of your analysis. Ensure you include GitHub repository URLs of your source code in the references.**
   * 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."](https://www.algosome.com/articles/dummy-variable-trap-regression.html)
   * 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 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 expecting some overfitting 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.
   * Learning about the importance, I was able to determine that one input, gross weight, was vastly more important than the rest, and it predicted the target feature so well that the low importance of the other features did not matter.
5. **(2 points) What are the results of your analysis, and display them in an appropriate format?**
6. 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.
7. ![A screenshot of a number of numbers

   Description automatically generated]()
8. ![A graph showing a line of blue dots

   Description automatically generated]()