Regression Analysis

Kasen Teoh, Jiayu Zhang, Jiaxi Wang

Winter 2023

Question 1

Question 1.1: Correlation Patterns

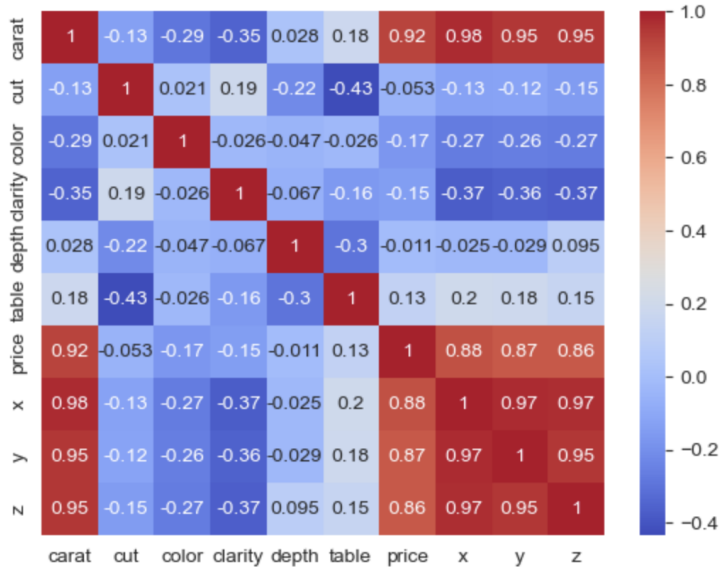

Figure 1: Correlation Heat Map

```
Feature with highest absolute correlation with the target variable: carat
```

Figure 2: Highest Correlation

The correlation pattern suggests a strong relationship between the weight and size of a diamond and its price.

Question 1.2: Histogram of Numerical Features

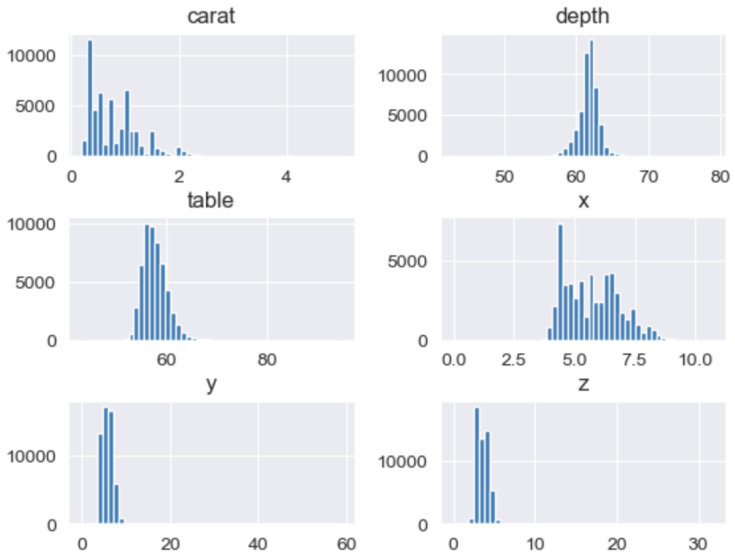

Figure 3: Histogram of Numerical Features

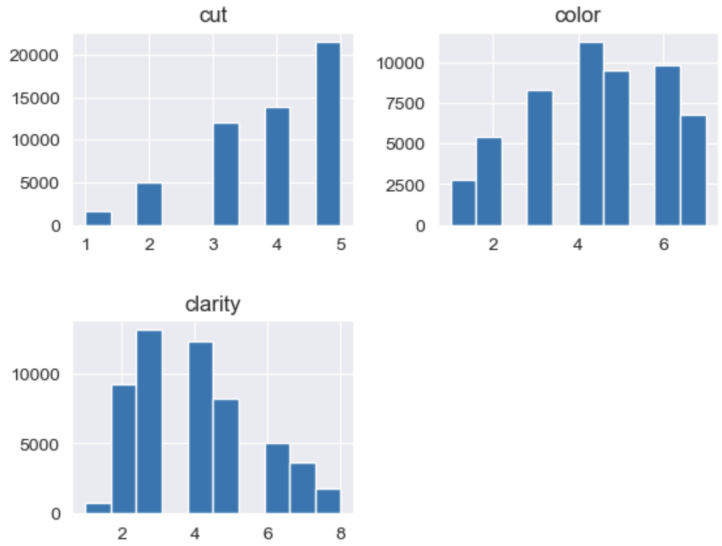

Figure 4: Histogram of Categorical Features Converted to Numerical

If the distribution of a feature has high skewness, we can perform a square root transformation, which takes the square root for each value in the feature. This only applies to non-negative values which fit our dataset greatly if there is high skewness.

Question 1.3: Boxplot of Numerical Features

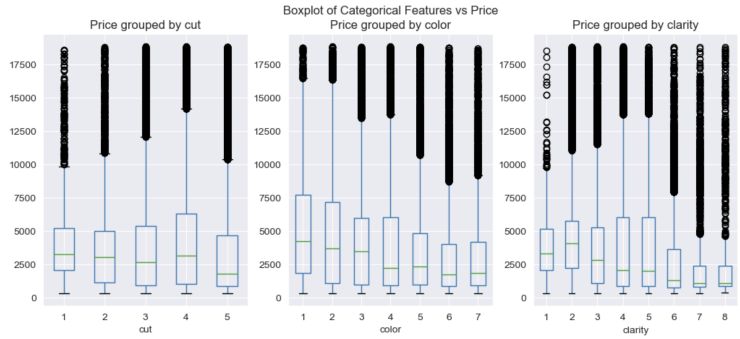

Figure 5: Boxplots of Categorical Variables

As we can see from the boxplots above, it's counterintuitive that as cut/color/clarity increases in the better direction, the price actually decreases, however, one important fact is the target feature is the price, not unit price. We will show later that as the three features grow in a better direction, the unit price actually increases as we'd expected.

Question 1.4: Yearly Trends

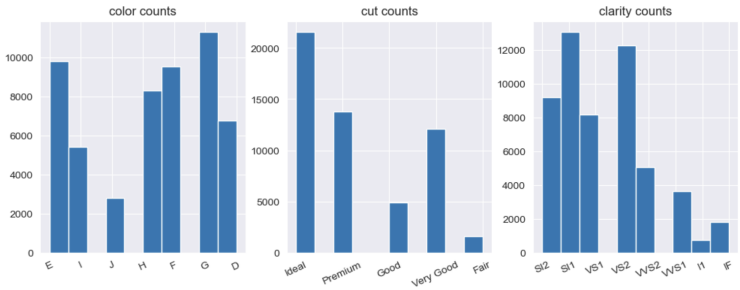

Figure 6: Categorical Variables Frequency

Question 2

Question 2.1: Standardize Features

```
q2_1_scaler = StandardScaler()
df[q1_2_original_num_col] = q2_1_scaler.fit_transform(df[q1_2_original_num_col])
df[q1_2_categorical_num_col] = q2_1_scaler.fit_transform(df[q1_2_categorical_num_col])
```

Figure 7: Standardizing Columns

We utilized StandardScaler from Scikit Learn's Pre-processing package. It aims to remove the mean and scaling to the unit variance.

Question 2.2: Feature Selection

Feature	Estimated Mutual Information
carat	1.655434
depth	0.030518
table	0.032906
x	1.412225
y	1.421402
z	1.363155
cut	0.057657
color	0.136499
clarity	0.216049

Figure 8: Mutual Information

Feature	F Score
carat	304051.486618
depth	6.115863
table	886.119363
x	193741.523066
y	160915.662263
z	154923.266553
cut	154.784468
color	1654.401244
clarity	1188.007065

Figure 9: F-Score

Depth and table are the two features that have the lowest MI w.r.t. the target.

This step is crucial because it affects the performance of all our models. If we select features that are not significant in predicting the price of a diamond, we may not get an as accurate prediction, while using a lot of computation power. If we select the right number of features andd take into consideration which ones are most important, we should be able to train a model with a low rmse. This is only true for regression models. The two lowest mutual information scores are the two variables depth and table.

Question 3: OOB and R^2 Score

In a random forest, there are bootstrapped samples of the training set, and with this, there are some rows of the training set that are left out of particular decision trees. The OOB score is the number of correctly predicted rows that were left out of the bag. There are many advantages of the OOB score, for instance, it has less variance and requires less computing power.

The R^2 metric is a value between 0 and 1 that determines where the model is a good fit for the data, i.e how much of the data's variance can it explain.

When evaluating a random forest model, the OOB score is often more accurate and useful.

Question 4

Question 4.1: Regularization Impact

Feature	OLS	Lasso	Ridge
carat	5085.348104	5039.562081	5082.899769
depth	-92.337499	-89.174239	-92.209558
x	-954.649062	-909.594286	-952.252907
cut	164.609641	163.834461	164.602501
color	549.355914	547.353154	549.286561
clarity	830.255965	830.169575	830.276261

Figure 10: Best Polynomial Regression Hyper-Parameters

Based on the above data, the absolute value of all coefficients are low for Lasso (L1) regression. It remains similar for Ridge (L2) regression compared with OLS.

Question 4.2: Best Regularization Scheme

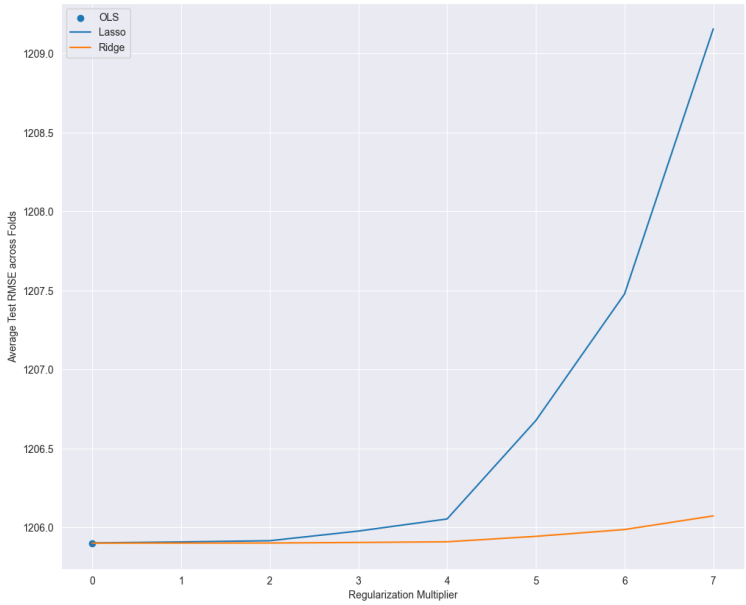

Figure 11: Best Polynomial Regression Hyper-Parameters

As we can see in the graph, regularization only increase the RMSE, so the best regularization scheme is no regularization.

Question 4.3: Feature Standardization in Model Performance

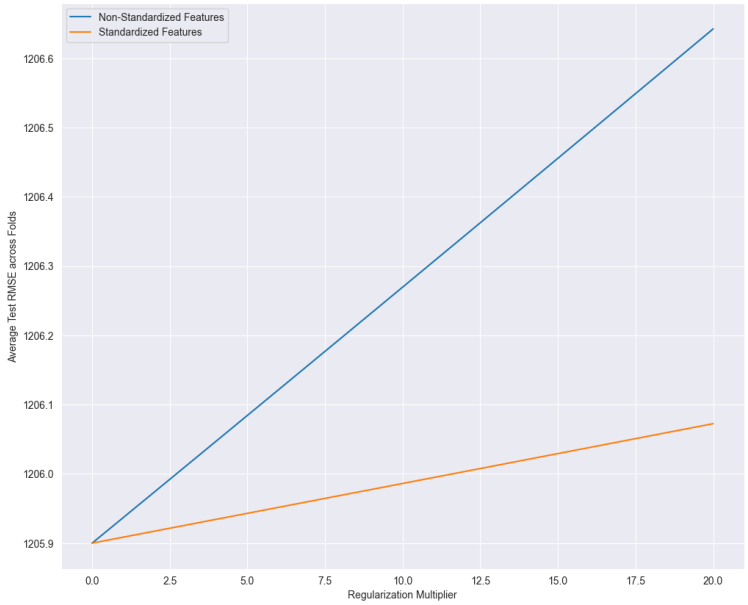

Figure 12: Best Polynomial Regression Hyper-Parameters

Feature standardization helps performance, but really minor. As we can see in the above plot.

Question 4.4: Linear Regression P-Values

The p-value is the probability of observing a value that is equal to 0, in other words, whether a specific feature will add information into our model, significant or not. To determine the most significant features, we can use a threshold for the p-vale and whether it is below a threshold, then the probability of it being 0, is very low.

Question 5

Question 5.1: Salient Features

	Best Polynomial Regression
Best Estimator	(PolynomialFeatures(degree=3), Ridge())
Best Parameters	{'poly__degree': 3}
RMSE	25.719061

Figure 13: Best Polynomial Regression Hyper-Parameters

The top 5 most salient features from the features that we had picked out turned out to be clarity, x, carat, cut, and color.

Question 5.2: Optimal Degree

We used a grid search with a 10-fold cross-validation across varying degrees from 1 to 6. We found that the best degree was 3. If the degree of the training set is very high, it is very susceptible to overfitting because it is adding more features and making the model more complex. We see that the mean test RMSE score is 25.498

Question 6

Question 6.1: Good Hyper-parameter Set

- Hyper parameter hidden layer size: 50, 100, 150
- alpha: 0.0001, 0.001, 0.01
- output activation function: identity, relu

	Best MLP
Best Estimator	MLPRegressor(hidden_layer_sizes=150, max_iter=...
Best Parameters	{'alpha': 0.0001, 'hidden_layer_sizes': 150}
RMSE	1219.600341

Figure 14: Identity: Best Polynomial Regression Hyper-Parameters

	Best MLP
Best Estimator	MLPRegressor(hidden_layer_sizes=100, max_iter=...
Best Parameters	{'alpha': 0.0001, 'hidden_layer_sizes': 100}
RMSE	1233.537242

Figure 15: ReLu: Best Polynomial Regression Hyper-Parameters

Question 6.2: MLP vs Linear Regression

The performance of a neural net generally will do better than a simple linear regression because a linear regression only looks for the linear relationship between the data. While a neural net, being a deep learning model, will look for all kinds of relationships within the data. Additionally, linear regression has multiple restrictions, such as with multicollinearity, a neural net is able to bypass these restrictions and continue detecting linear and nonlinear relationships.

Question 6.3: Activation function

We tried ReLu and Identity as activation function of output layer and finally decided to use the ReLu activation function because other activation functions typically run into the Vanishing Gradient problem where neuron outputs are fairly small and hence as we advance layers, the values will become smaller and smaller, leading to a very slow convergence to their optimum value. ReLu on the other hand, the maximum threshold is infinity and hence will not have consecutive small neuron outputs, avoiding the Vanishing Gradient Problem.

Question 6.4: Risk of Increasing Network Depth

Increasing the depth of a neural network runs the risk of overtraining the model. If the problem at hand is not very complex, the addition of hidden layers will begin to identify the individual differences between the data, being unable to generalize the results, leading to a possibly higher training accuracy with a significant lower testing accuracy. Besides overfitting, the it will take longer for the model to train, even on the training data, because as the neural network becomes deeper, during back-propagation, the gradients need to be passed through too many layers and becomes super close to 0, making it hard for earlier layers to learn.

Question 7

Question 7.1: Hyper-Parameters in Overall performance

Maximum Number of Features:

- May increase the accuracy because it gives more options to the nodes in the forest; however, it decreases the diversity in each tree, and hence may, in turn, decrease the accuracy

Number of Trees:

- Higher number of trees allows the regression to be more accurate but will also increase computation power and decrease the speed

Depth of each tree:

- The larger the depth of a tree in a forest means the more the tree is able to split and capture the differences in the entire dataset, however, too much depth may lead to overfitting

Question 7.2: Non-linear Decision Boundary in Random Forests

Random forests use nested if-else statements to draw the boundaries between the data. By using these if-else statements, it draws partial rectangles between the data and hence these rectangles are of varying sizes, being able to capture non-linearity within the dataset.

Question 7.3: Root Branch and Significance

	Best Random Forest
Best Estimator	(DecisionTreeRegressor(max_depth=4, max_featur...
Best Parameters	{'max_depth': 4, 'max_features': 3}
RMSE	1595.960927
OOB Score	0.925787

Figure 16: Best Random Forest Hyper-Parameters

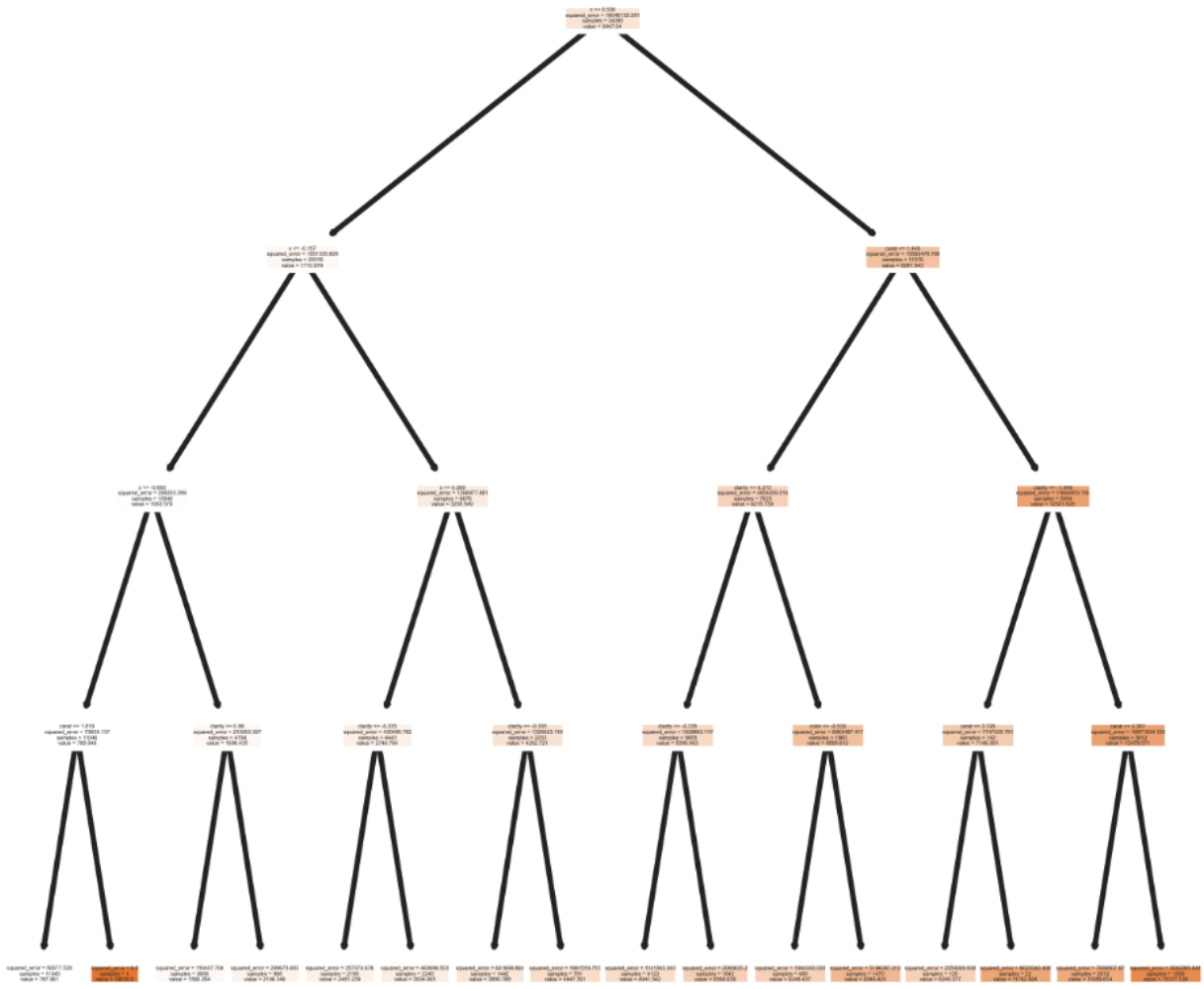

Figure 17: Single Tree Plot

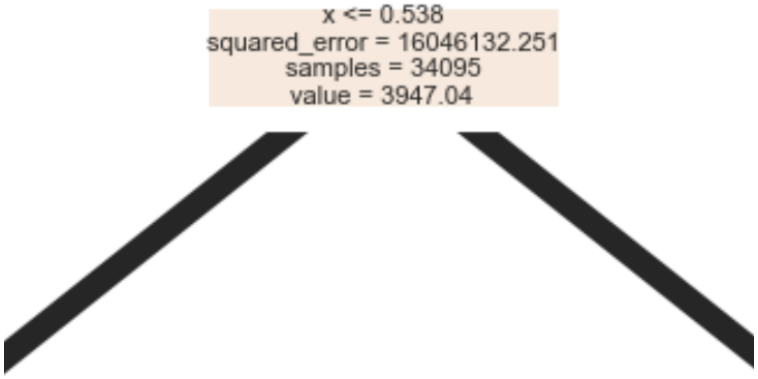

Figure 18: Root Node Split

The feature selected for branching at the root node is the x value. The importance of this feature is that it is the feature that offers the highest information gain, whether the x dimension is less than or equal to a specific value. This is inline with what we found with the mutual information and f-score.

Moreover, the results does match what we got in 3.3.1. As we can see the results from 3.3.1, x has the most negative coefficient.

Question 7.4: OOB and R^2

	Best Random Forest
OOB Score	0.923969
R^2 Score	0.925178

Figure 19: Test OOB Score and R^2

We see that the OOB score is 0.926. This means that we are able to correctly predict 92.6% of the data. The R^2 metric is a value between 0 and 1 that determines where the model is a good fit for the data, i.e how much of the data's variance can it explain.

R^2 is a measure of how well the model does on the regression job. Best is 1.0 and scores can be negative. A model predicting the average y for all input x will get a score of 0.0.

Question 8

Question 8.1: Important Parameters

Some important hyper-parameters are the learning rate, depth, l2 leaf ref, and the number of estimators.

- learning rate: determines the step size at each iteration. A small search space is possibly between 0.01 and 0.2
- depth: the depth of the tree. A small search space is between 5 and 10
- l2 leaf regularization: coefficient at the L2 regularization term. A small search space is between 1 and 10
- number of estimators: the number of trees in the entire model. A small search space is possibly between 50 and 1000

Question 8.2: Best Hyper-Parameter and RMSE

Best Parameters: depth: 8; l2_leaf_reg: 7; learning_rate: 0.05724084038448387; n_estimators: 1000

Best RMSE: 533.4048519689828

Question 8.3: Interpret Effect of Hyper-Parameters

- iteration helps with performance, especially accuracy. Random_strength,depth,learning_rate also has the greatest effect on optimizing the CatBoost evaluation metrics
- l2_leaf_reg affects regularization.
- max_depth and learning rate affects fitting efficiency. Decreasing max_depth would increase training time and decreasing learning rate would increase training time

Question 9

Question 9.1: Twitter Statistics

	avg_tweets_hour	avg_follower_tweet	avg_retweets_tweet
subset_tweets_#gohawks.txt	34.882277	2570.136662	2.603156
subset_tweets_#gopatriots.txt	34.904948	1440.826929	1.412466
subset_tweets_#nfl.txt	34.049396	5771.680435	1.486482
subset_tweets_#patriots.txt	34.538433	3889.987776	1.850514
subset_tweets_#sb49.txt	34.732980	12840.943765	2.536324
subset_tweets_#superbowl.txt	35.743066	8479.468245	2.208087

Figure 20: Tweet Statistics

Question 9.2: Histograms

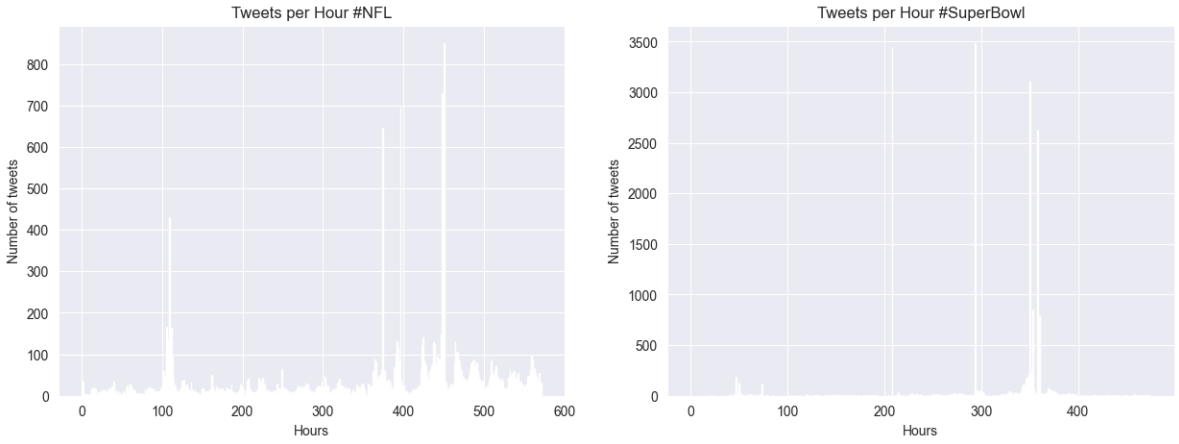

Figure 21: Tweets per Hour for #NFL and #SuperBowl

Question 10: Create your own task

Introduction

We designed 3 tasks which cover most of aspects of the dataset, including (1) number of retweets, (2) tweet hour, (3) tweet hashtag. Note that (1) and (2) are regression tasks and (3) is classification task. We intend to use GloVe embedding as feature engineering for all tasks. Moreover, we plan to use neural approaches for the regression tasks and Random Forest Classifier for (3). For baseline models, we used Linear Regression and Decision Tree. After

experiment, we reached lower RMSE for all 3 tasks. Due to limited computing resources, we down-sampled the dataset.

Tasks

(1) Predicting Number of Retweets

Using the feature engineered sentence embedding vector, fine-tune a multi-layer perceptron that predicts the number of retweets for the tweet.

(2) Predicting Tweet Hour

Using the feature engineered sentence embedding vector, fine-tune a multi-layer perceptron that predicts the hour that the tweet was posted.

(3) Predicting Hashtag of Tweets

Using the feature engineered sentence embedding vector, with hashtag pre-removed, fine-tune a random forest classifier that predicts the hashtag that was originally in the tweet.

Data Exploration

We first draw the histogram with all the target to get a sense of the target's distirbution.

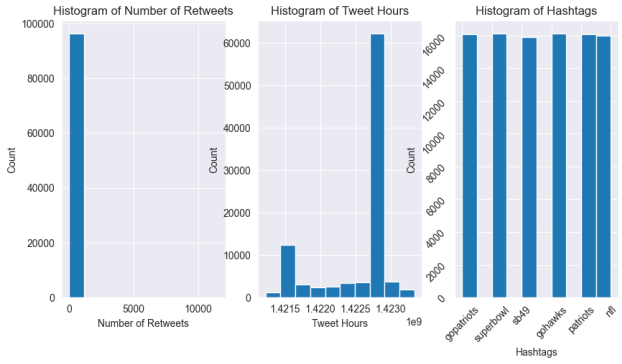

Figure 22: Histogram of Targets

However, we see that most of the number of retweets are 1 or 2. Thus, we plot the value counts.

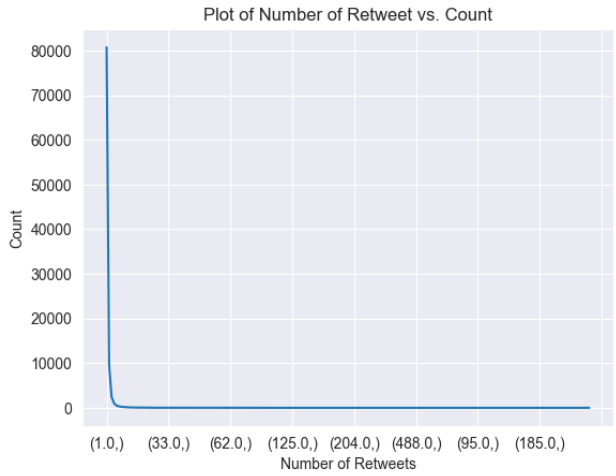

Figure 23: Plot of Number of Retweets vs. Counts

Feature Engineering

We used GloVe word embedding as the first step of feature engineering to convert each raw token to meaningful 300-dimensional vectors. Then we applied feature engineering experience from Project 1, where we take the mean of the embeddings of all the words in a sentence to represent as sentence embedding. The reason being that two of our tasks are strongly neural approach, so it would make sense to use neural approach for feature engineering also. Moreover, neural approaches in many cases performs better than classical feature engineering approaches, thus, we use the above feature engineering process.

Baseline

(1) Number of Retweet - Linear Regression

```
Baseline RMSE:  15.212235
```

Figure 24: RMSE of Linear Regression

(2) Tweet Hour - Linear Regression

```
Baseline RMSE:  467454.66
```

Figure 25: RMSE of Linear Regression

(3) Hashtag of Tweets - Decision Tree

```
Baseline accuracy:  0.336201521386706 56
```

Figure 26: Accuracy of Decision Tree

Evaluation

(1) Number of Retweet

For this task, we tried all combinations of the following parameter options.

Hidden Layer Sizes	Activation Function (after each layer)	L2 Regularization Coefficient
150, 50	ReLU	0.001
200, 100, 50	Identity	0.1
200, 500, 200, 100, 50	Sigmoid	10
		1000

It turns our that the following combination has the best performance:

- Hidden Layer Sizes:

$$200, 100, 50$$

- Activation Function: Sigmoid
- L2 Regularization Coefficient: 0.1

	Best MLP
Best Estimator	<class 'skorch.regressor.NeuralNetRegressor'>[...
Best Parameters	{'module__activation_func': 'sigmoid', 'module...
RMSE	43.04315

Figure 27: RMSE of Multi-Layer Perceptron on Cross Validation

The final test result of the model with the above parameter set trained on the entire training set is,

```
With the test set, the RMSE is 15.035945
```

Figure 28: RMSE of Multi-Layer Perceptron on Test Set

Therefore, we improved over the baseline.

(2) Tweet Hour

For this task, we tried all combinations of the following parameter options.

Hidden Layer Sizes	Activation Function (after each layer)	L2 Regularization Coefficient
150	ReLU	0.001
200, 100, 50	Identity	0.1
250, 200, 150, 100, 50	Sigmoid	10
200, 500, 500, 200, 100, 50		1000

It turns our that the following combination has the best performance:

- Hidden Layer Sizes: 200, 100, 50
- Activation Function: ReLU
- L2 Regularization Coefficient: 10

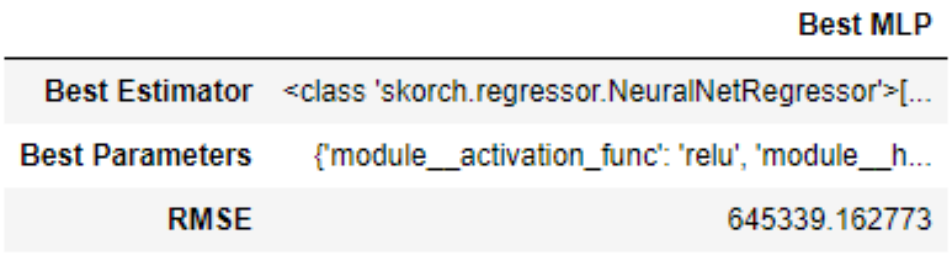

Figure 29: RMSE of Multi-Layer Perceptron on Cross Validation

Note that the RMSE on of the best model on validation set greatly exceed that of the baseline. It might be a good idea to normalize our target, the time at which the tweet was posted. Because timestamp is large number. Thus, we perform normalization on the data and re-evaluate the baseline and the multi-layer perceptron.

Normalized Target: Baseline - Linear Regression

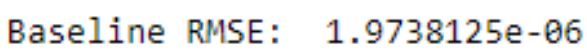

Figure 30: RMSE of Linear Regression

Normalized Target: Multi-Layer Perceptron

We tried all combinations of the following parameter options, we used experience we gathering from the unnormalized data traning to narrow down the search space.

Hidden Layer Sizes	Activation Function (after each layer)	L2 Regularization Coefficient
200, 100, 50	ReLU	0.1
200, 500, 500, 200, 100, 50	Sigmoid	10

It turns our that the following combination has the best performance:

- Hidden Layer Sizes: 200, 100, 50
- Activation Function: ReLU
- L2 Regularization Coefficient: 0.1

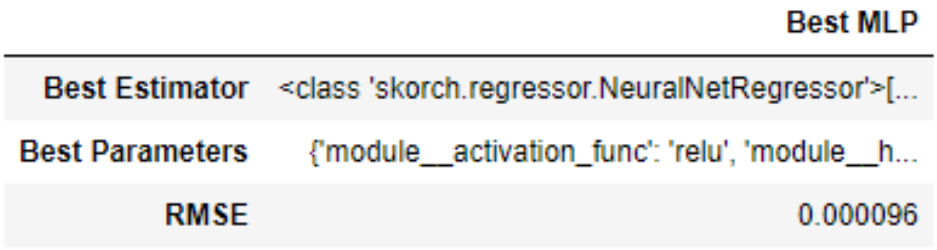

Figure 31: RMSE of Multi-Layer Perceptron on Cross Validation

As we can see, the validation RMSE of the best model during grid search improved a lot with respect to that of the baseline relatively. Therefore, we can use this parameter set to train the model on the entire training set and evaluate the RMSE on the test set.

The final test result of the model with the above parameter set trained on the entire training set is,

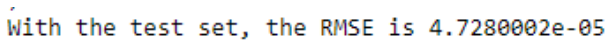

Figure 32: RMSE of Multi-Layer Perceptron on Test Set

Note that we did not perform better than baseline. There is probably room for improvement so that neural approach can outperform linear regression. However, due to limited computing resources, we will keep exploring after the quarter ends.

(3) Hashtag of Tweets

For this task, we tried all combinations of the following parameter options.

Max Depth	Number of Estimators
2	100
3	200
4	500
5	1000

It turns our that the following combination has the best performance:

- Max Depth: 5
- Number of Estimators: 1000

	Best Random Forest
Best Estimator	(DecisionTreeClassifier(max_depth=5, max_featu...
Best Parameters	{'max_depth': 5, 'n_estimators': 1000}
Accuracy	0.368092
OOB Score	0.368248

Figure 33: Accuracy and OOB Score for Random Forest on Cross Validation

The final test result of the model with the above parameter set trained on the entire training set is,

```
The OOB score is 0.36787354768981356
The test accuracyis 0.37061977802718543
```

Figure 34: RMSE of Multi-Layer Perceptron on Test Set

Therefore, we improved over the baseline by a lot. One observation about the grid search result is that there might be a better model if we can further increase the two parameters' value. However, due to limited computing resources, we will only make plan on this.

Possible Improvement

One observation we made was that the model always has lower RMSE when trained on the entire training set than those trained during cross validation. A possible hypothesis for this is that during cross validation, the training set's size is smaller. So we could potentially increase the training data subset's size to improve performance. We plan to do so, however, due to limited computation resources, we are unable to finish it before the project is due.

Remarks for Neural Approach

We are using the neural approach, however, one thing worth noting is that sklearn's implementation of neural models, such as MLPRegressor is incapable of using GPU acceleration. Our group fortunately do have GPU access, so it is a waste of time for us to grid search without GPU accelration when GPU is available. Thus, we used a package called skorch which connects PyTorch and sklearn. So we can define a neural model as we usually do in PyTorch and skorch will connect it with the API of sklearn models, therefore giving us the ability to use sklearn's GridSearchCV function. Thus greatly boosted our project progress, in terms of the number of combinations we can try in grid search.