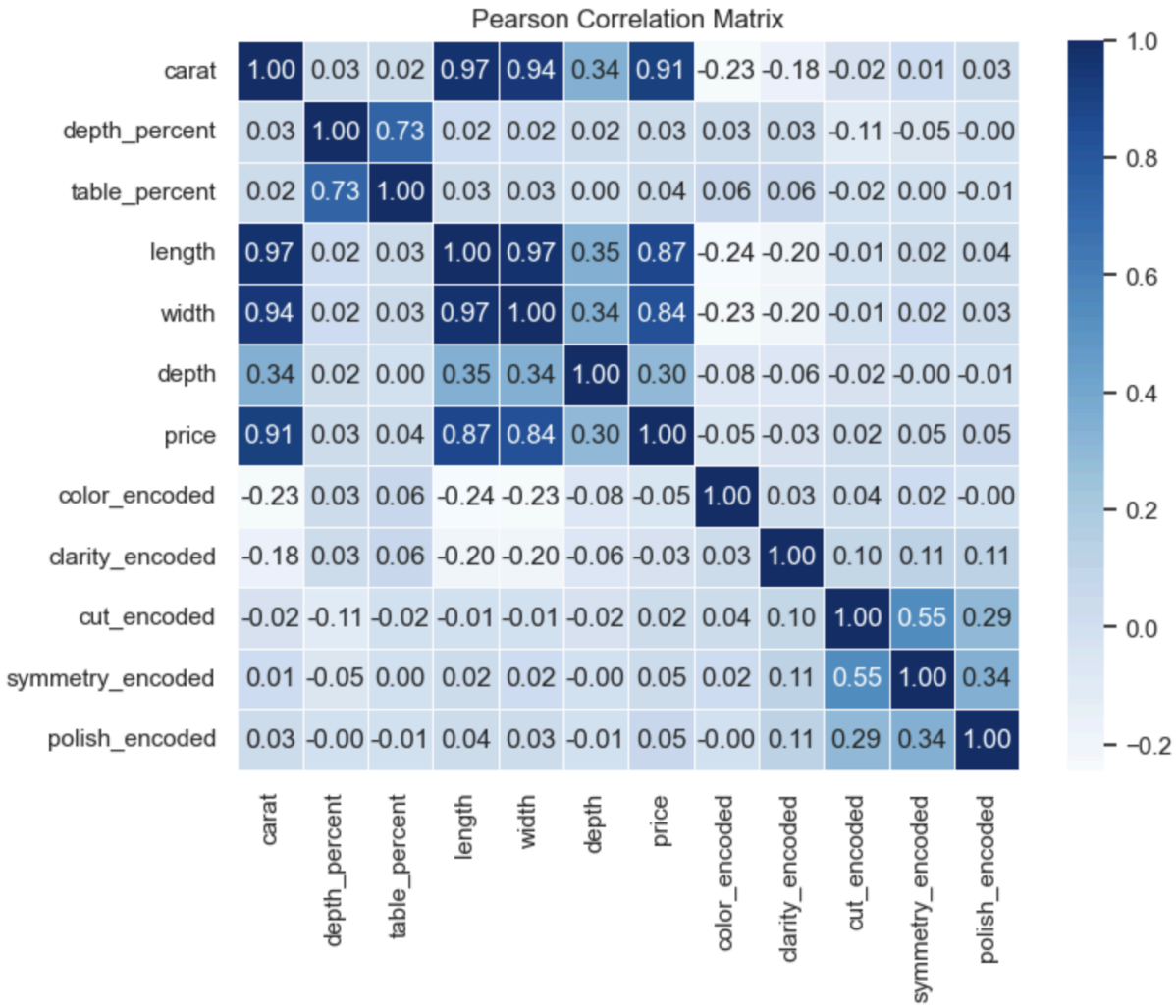
Project 4: Regression Analysis and Define Your Own Task

Lauren Mizner (UID# 005225768)

Part 1: Regression Analysis

Question 1.1:

Plot a heatmap of the Pearson correlation matrix of the dataset columns. Report which features have the highest absolute correlation with the target variable. In the context of either dataset, describe what the correlation patterns suggest.



Features with highest absolute correlation with the target variable (price):

Carat = 0.913479

Length = 0.869521

Width = 0.841887

Depth = 0.299696

Polished_encoded = 0.054928

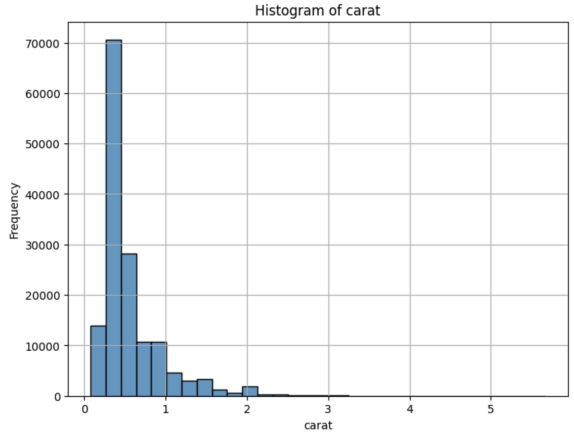
Color_encoded = 0.047189

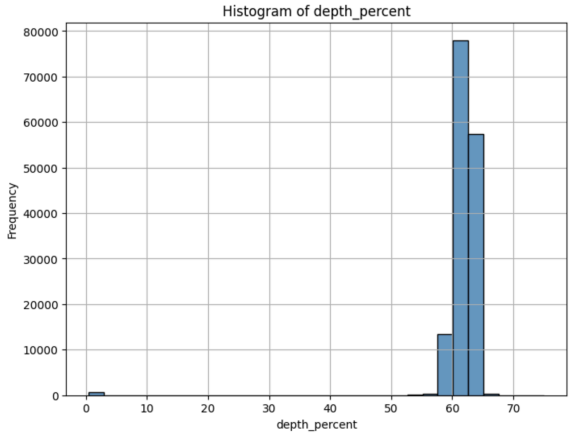
Symmetry_encoded = 0.047149
Table_percent = 0.042453
Clarity_encoded = 0.029951
Depth_percent = 0.025469
Cut_encoded = 0.024356

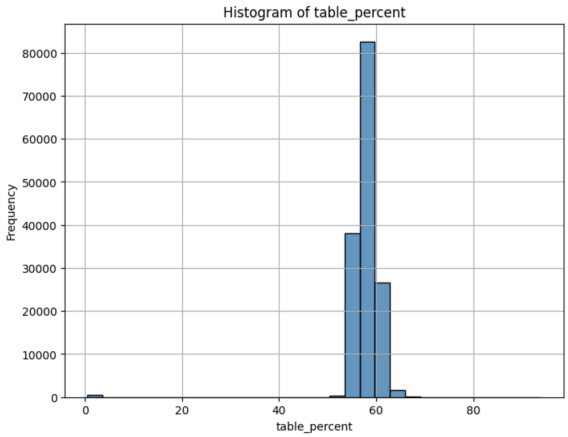
Question 1.2:

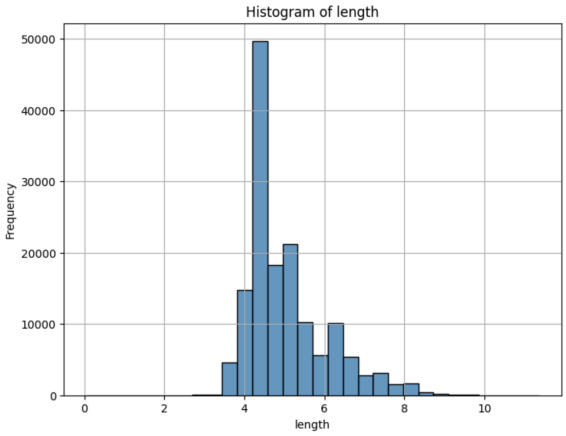
Plot the histogram of numerical features. What preprocessing can be done if the distribution of a feature has high skewness?

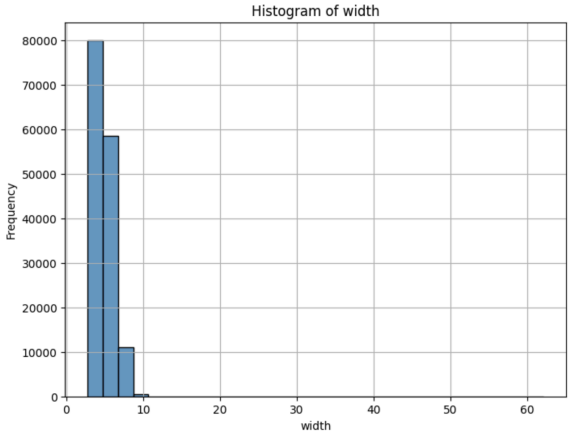
If the distribution of a numerical feature has a high-skew, which we can see in the histogram of the diamond carat feature with a left-skew, it means that the data is not symmetrically distributed around the average and can affect the overall performance of machine learning models. Preprocessing that can be done to mitigate the high-skew data includes normalization or standardization. Standardization scales the data to have a mean of zero and a standard deviation of one, and will later be performed on the data in step 3.1.3. Other options for preprocessing techniques to mitigate a high-skew include, but are not limited to, log transformation, square root transformation, and box-cox transformation.

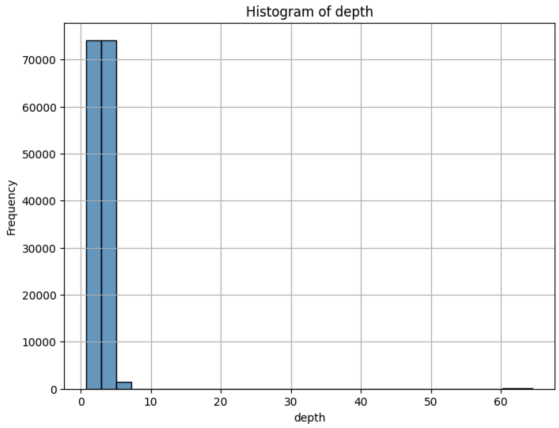


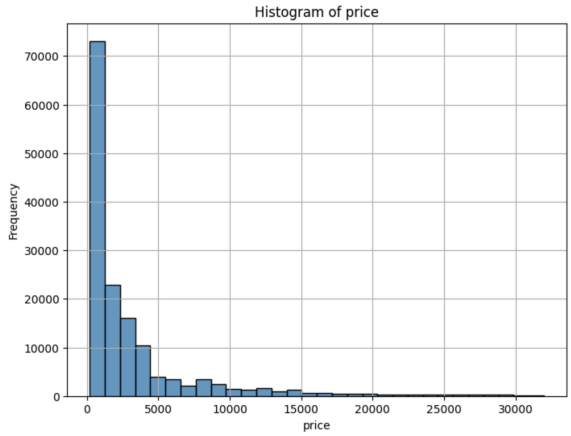


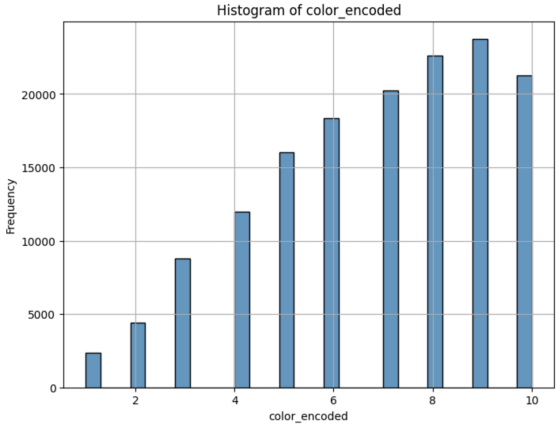


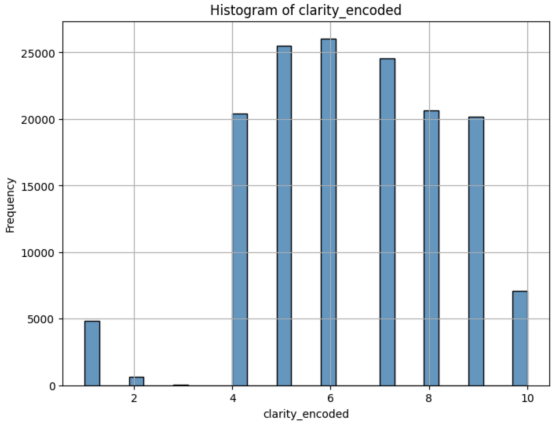


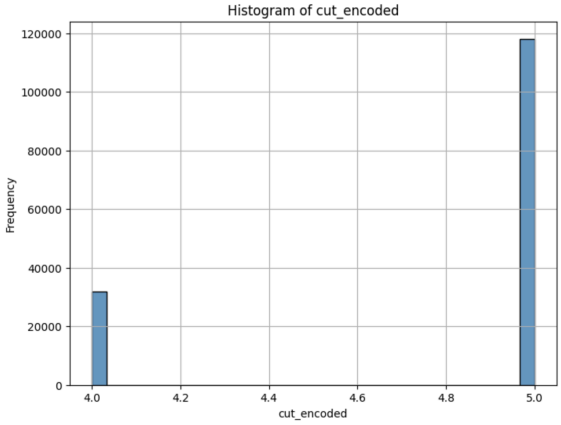

Histogram of width
Frequency
width
0
10000
20000
30000
40000
50000
60000
70000
80000
0
10
20
30
40
50
60

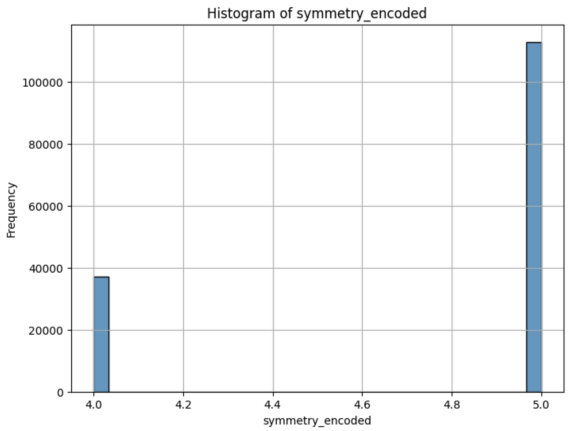

Histogram of depth
Frequency
depth
0
10000
20000
30000
40000
50000
60000
70000
0
10
20
30
40
50
60

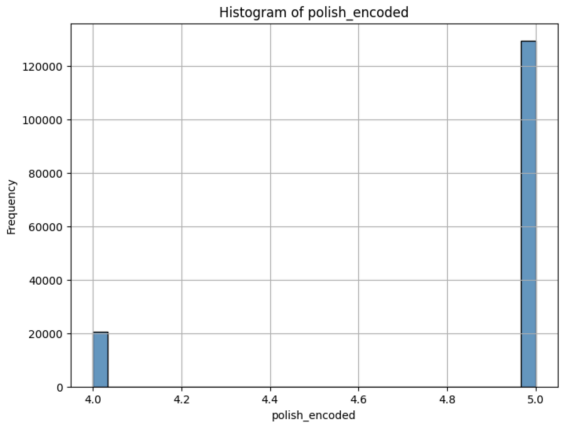

Histogram of price
Frequency
price
0
10000
20000
30000
40000
50000
60000
70000
0
5000
10000
15000
20000
25000
30000


Histogram of color_encoded
Frequency
color_encoded
0
5000
10000
15000
20000
2
4
6
8
10


Histogram of clarity_encoded
Frequency
clarity_encoded
0
5000
10000
15000
20000
25000
2
4
6
8
10

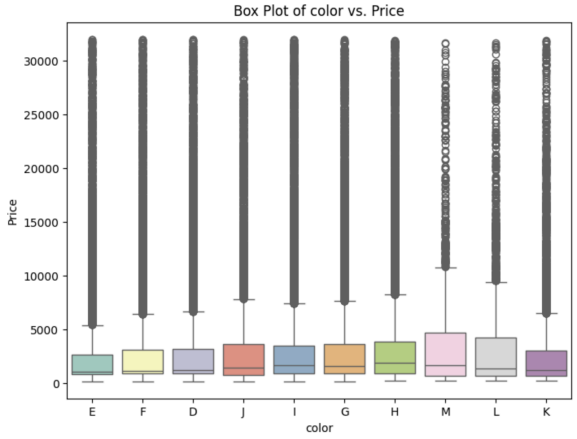

Histogram of cut_encoded
Frequency
cut_encoded
0
20000
40000
60000
80000
100000
120000
4.0
4.2
4.4
4.6
4.8
5.0

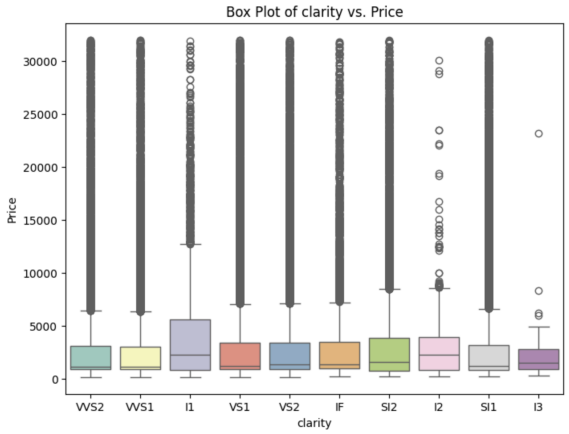


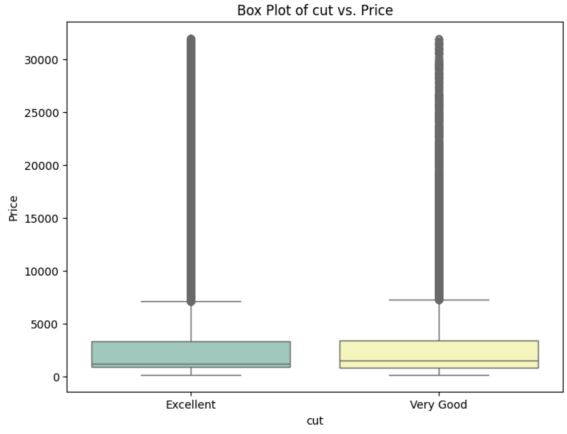


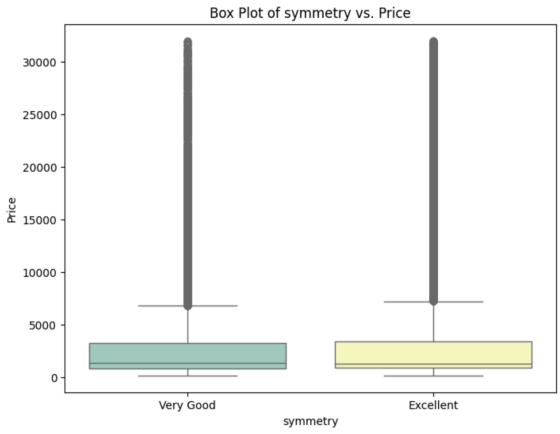
Question 1.3:

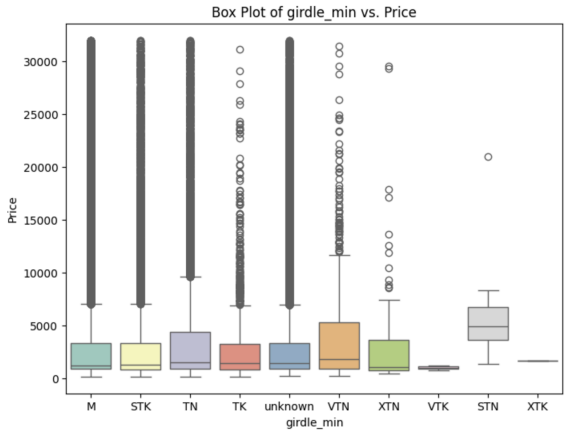
Construct and inspect the box plot of categorical features vs target variable. What do you find?

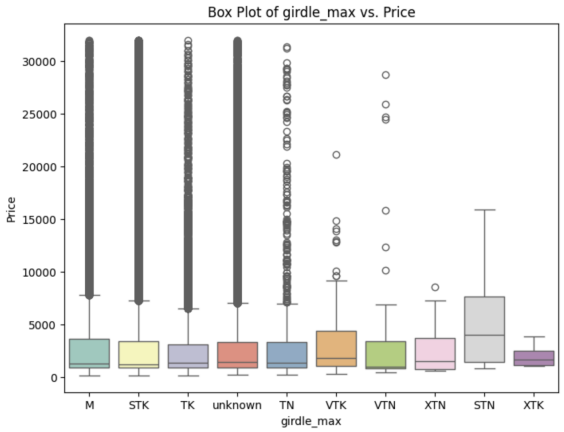








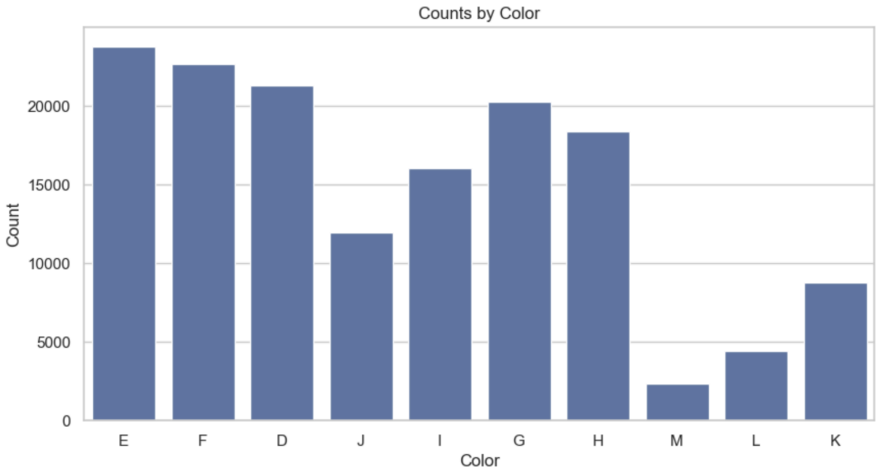


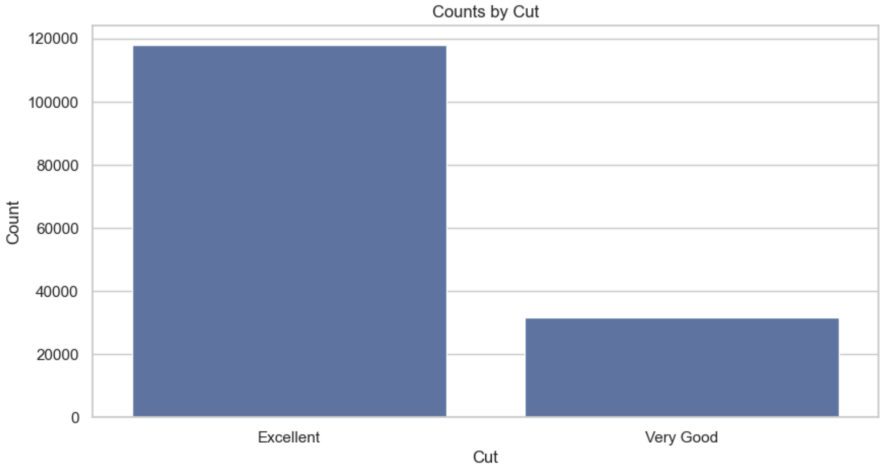


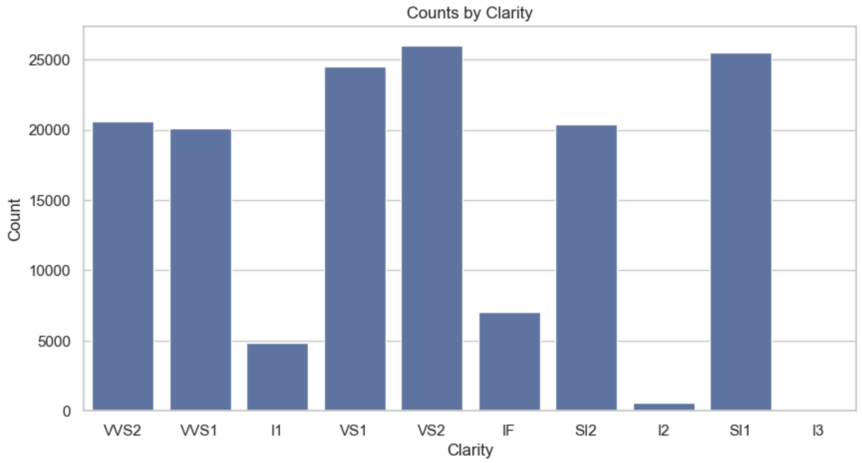
From the boxplots shown below, we can see that there are many outliers amongst the data for most, if not all, of the features when compared to price. Typically, across all features, we're seeing a maximum price between 5000 and 10000 dollars, with the outliers continuing on from there. We're also seeing extremely similar plots when looking at the features of cut, symmetry, and polish when compared to price. This shows a very high correlation and consistency across price when looking at these features.

Question 1.4:

For the diamonds dataset, plot the counts by color, cut, and clarity.







Question 2.1:
Standardize feature columns and prepare them for training.

carat	depth_percent	table_percent	length	width	depth	price
-1.157106	0.215866	0.345119	-2.146391	-2.078247	-0.730430	-0.659094
-1.157106	0.014689	0.345119	-2.156289	-2.059209	-0.735681	-0.659094
-1.157106	-0.186488	0.345119	-2.116697	-2.049690	-0.740932	-0.659094
-1.157106	0.039836	0.345119	-2.136493	-2.068728	-0.735681	-0.659094
-1.157106	0.769101	0.218693	-2.205778	-2.116324	-0.714676	-0.659094

Question 2.2:
Describe how this step qualitatively affects the performance of your models in terms of test RMSE. Is it true for all model types? Also list two features for either dataset that has the lowest MI with respect to the target.

When calculating the test RMSE for the full dataset/all features we get a test RMSE value of 1597.66. When utilizing feature selection, the features with the largest MI (Mutual Information) score were compared to the top 5 features generated from mutual_info_regression and f_regression. The top features from these analyses were determined to be carat, width, length, depth, color_encoded, and clarity_encoded. The test RMSE value obtained from this feature selection was calculated as 1605.90. Feature selection generally aims to identify the most informative features and cut out the more irrelevant features while maintaining model performance. From the results, we did a pretty good job at reducing the data load through feature selection while maintaining a very similar test RMSE value. Most models using feature selection are expected to result in a comparable or slightly worse performance compared to using all the features.

Test RMSE Values:
Test RMSE on All Features = 1597.66
Test RMSE on Feature Selection = 1605.90

Two features with the Lowest MI Score:
Polish_encoded = 0.012901
Table_percent = 0.024630

Linear Regression

Question 4.1:

What is the objective function? Explain how each regularization scheme affects the learned parameter set.

The objective function, also known as the loss function or cost function, quantifies the model's performance by measuring the difference between predicted and actual values.

Objective Functions:

$$\text{OLS} \rightarrow min \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$\text{Lasso (L1 Regularization)} \rightarrow min \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \alpha \sum_{j=1}^{p} |w_j|$$

$$\text{Ridge (L2 Regularization)} \rightarrow min \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \alpha \sum_{j=1}^{p} w_j^2$$

OLS:

The objective function for OLS regression is the sum of squared differences between the observed and predicted values. OLS is used to minimize the sum of squared residuals directly, and the learned parameter set, or coefficients, are determined solely based on the minimizing of the MSE between the predicted value and actual values in the data.

Lasso (L1 Regularization):

The objective function for Lasso regression is the sum of squared differences between the observed and predicted values plus the sum of the absolute values of the coefficients multiplied by a regularization parameter, alpha. Lasso regression uses an L1 penalty term to encourage sparsity in the learned parameter set by driving some coefficients to zero. This is so that Lasso regression can perform feature selection, effectively removing the less important features from the model.

Ridge (L2 Regularization):

The objective function for Ridge regression is the sum of squared difference between the observed and predicted values plus the sum of the squared values of the coefficients multiplied by a regularization parameter, alpha. Ridge regression uses an L2 penalty term to penalize large coefficients and shrink them toward zero without actually enforcing sparsity, in an attempt to produce a more stable, conditioned model and avoid overfitting.

Question 4.2:

Report your choice of the best regularization scheme along with the optimal penalty parameter and explain how you computed it.

Best Regularization Scheme = Lasso
Optimal Penalty Parameter (alpha) = 0.1

The results were very close between Lasso with an alpha of 0.1 and Ridge with an alpha of 1.0, with a very slight difference between the MSE values to determine which regularization scheme was better. Each regularization scheme, lasso and ridge, was evaluated over multiple penalty parameters, alpha. A cross-validated grid search was utilized to search over the hyperparameters, while optimizing for the mean square error (MSE) value as our performance metric. From there we were able to compare MSE score to determine which combination of regularization scheme and penalty parameter produced the best or smallest MSE value.

Question 4.3:

Does feature standardization play a role in improving the model performance (in the cases with ridge regularization)? Justify your answer.

RMSE Ridge Regression w/Standardized Features = 1597.664
RMSE Ridge Regression w/Unstandardized Features = 1597.667

Based on the results seen in our analysis, standardizing the features, specifically with ridge regression, does not result in a significant improvement in model performance. The resulting RMSE values for standardized and unstandardized ridge regression models were almost identical, as shown above. While it is good practice to use feature standardization, it does not necessarily result in better performance of the model.

Question 4.4:

Some linear regression packages return p-values for different features. What is the meaning of these p-values and how can you infer the most significant features? Qualitative reasoning is sufficient.

The p-values associated with features in linear regression models provide a measure of statistical significance of the feature's contribution to explaining the target variable. A low p-value (typically below 0.05) tells us that the corresponding feature is statistically significant, meaning there is strong enough evidence to reject the null hypothesis, which assumes the feature has no effect on the target variable. On the other hand, a high p-value (typically above 0.05) tells us that the corresponding feature is not statistically significant and that we do not have enough evidence to reject the null hypothesis.

```
OLS Regression Summary:
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.884
Model:                            OLS   Adj. R-squared:                  0.884
Method:                 Least Squares   F-statistic:                 8.313e+04
Date:                Tue, 21 May 2024   Prob (F-statistic):               0.00
Time:                        15:15:00   Log-Likelihood:            -1.0550e+06
No. Observations:              119896   AIC:                         2.110e+06
Df Residuals:                  119884   BIC:                         2.110e+06
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          -4378.8867    133.158    -32.885      0.000   -4639.875   -4117.898
carat           1.392e+04     49.580    280.843      0.000    1.38e+04     1.4e+04
depth_percent     -34.9134      1.739    -20.073      0.000     -38.323     -31.504
table_percent      26.4082      1.738     15.195      0.000      23.002      29.815
length           -961.9445     27.061    -35.548      0.000   -1014.983    -908.906
width             -41.3086     18.562     -2.225      0.026     -77.690      -4.927
depth             -13.1069      2.639     -4.967      0.000     -18.279      -7.935
color_encoded     335.9743      2.038    164.870      0.000     331.980     339.968
clarity_encoded   293.8248      2.378    123.579      0.000     289.165     298.485
cut_encoded       226.4799     13.807     16.404      0.000     199.419     253.541
symmetry_encoded   67.9619     13.197      5.150      0.000      42.096      93.828
polish_encoded     19.3268     14.577      1.326      0.185      -9.245      47.898
==============================================================================
Omnibus:                    52809.570   Durbin-Watson:                   1.997
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          4183888.042
Skew:                           1.242   Prob(JB):                         0.00
Kurtosis:                      31.833   Cond. No.                     2.53e+03
==============================================================================
```

Above are results obtained from running an OLS model summary. We can see that most of the features return a p-value less than 0.05, meaning that those features are statistically significant. One feature, polish_encoded, returns a p-value of 0.185, indicating that it is not statistically significant in this model.

Polynomial Regression

Question 5.1:

What are the most salient features? Why?

The most salient features are those that have the strongest impact on predicting the target variable. We can identify the most salient features as those with high coefficient values, as shown below, and low p-values, as shown above. Below are the top 5 most salient features based on their coefficient values. The higher the coefficient value the larger the effect that feature has on the predicted values of the target variable.

Top 5 Most Salient Features:
Length → Coefficient = 302.18946
Width → Coefficient = 288.0446
Color_encoded → Coefficient = 161.9015
Clarity_encoded → Coefficient = 107.6702
Depth → Coefficient = 4.4897

Question 5.2:

What degree of polynomial is best? How did you find the optimal degree? What does a very high-order polynomial imply about the fit on the training data? What about its performance on testing data?

Best Polynomial Ridge Regression Model:
Polynomial Degree = 5
Penalty Parameter, alpha = 100
RMSE = 774.571

The best degree of polynomial was found to be a degree of 5. This was found by setting a hyperparameter grid with different degrees of polynomial features and different values of alpha for ridge regularization. A cross-validated grid search was utilized to find the best combination of hyperparameters with RMSE being utilized as the performance metric to judge the combinations on. The best model is evaluated on the testing data by making predictions and calculating the RMSE. A very high-order polynomial implies that the model becomes very flexible and is able to fit the training data very closely. This helps to capture all the patterns and intricacy of the data, resulting in very low training error. On the other hand, a very high-order polynomial can perform poorly on testing data due to overfitting. This is because the model learns the noise and fluctuations in data more so than the underlying relationship of the data, and, as a result, is unable to generalize well on unseen data.

Neural Network

Question 6.1:

Adjust your network size (number of hidden neurons and depth), and weight decay as regularization. Find a good hyper-parameter set systematically (no more than 20 experiments in total).

Best Hyperparameters:
Hidden Layer Size = (93, 0) or 1 layer of 93 neurons
Alpha = 0.01
Activation Function = tanh
RMSE = 703.238

These hyperparameters were obtained by defining parameter distributions for cross-validated randomized search, where the various combinations of these parameters were tested and performance was determined by the resulting RMSE value.

Question 6.2:
How does the performance generally compare with linear regression? Why?

In general, neural networks are better at capturing complex, non-linear relationships in data. Since neural networks are more complex themselves, they are able to learn more complex, intricate patterns in the data. On the other hand, linear regression models are more straightforward, utilizing coefficients to represent the direction and magnitude of the relationships between individual features and the target variable. So whether to use a linear regression model or a neural network depends on many factors such as data size, complexity of the data, etc. If the data is relatively simple it would make sense for a linear regression model to outperform a neural network, but if there is a large amount of data with complex patterns, then it would make sense for a neural network to outperform a linear regression model.

Question 6.3:
What activation function did you use for the output and why? You may use none.

When defining the parameter distributions for cross-validated randomized search, three different activation functions were considered: logistic, tanh, and relu. These activation functions, along with the other parameter distributions, were utilized to create different parameter combinations and test performance based on the resulting RMSE value. Based on the results, the activation function tanh, proved to deliver the best performance. The hyperbolic tangent activation function does have some pros such as a zero-centered output, which helps with the optimization process, smooth gradient property, which assists in optimization methods that rely on gradient descent such as backpropagation, and, lastly, the output ranges from -1 to 1 allowing for stronger gradients and better convergence.

Question 6.4:
What is the risk of increasing the depth of the network too far?

As a neural network becomes deeper, the gradients in backpropagation become very small, leading to slow or stalled learning. On the other hand, there's also the risk of exploding gradients, where the gradients become extremely large during training, making optimization difficult. Also, while deeper neural networks have a higher capacity to learn more complex patterns in training, this can lead to overfitting if the model memorizes the training data rather than generalizing well to unseen data.

Random Forest

Question 7.1:

Random forests have the following hyperparameters: maximum number of features, number of trees, and depth of each tree. Explain how these hyper-parameters affect the overall performance. Describe if and how each hyper-parameter results in a regularization effect during training.

Maximum Number of Features:

This parameter controls the maximum number of features that are considered for splitting at each node in a decision tree within a random forest. Increasing this value allows each tree to consider more features and can lead to more complex trees and potentially overfitting. Decreasing this value can limit the number of features each tree considered, leading to more simple trees and reduced risk of overfitting. This parameter can have a regularization effect by limiting the number of features to act as a form of feature selection. This can encourage the model to generalize better to unseen data.

Number of Trees:

This parameter controls the number of decision trees within the random forest. By increasing this value, we can improve the model's performance because it allows for more robust and stable predictions. If this value is too large it can lead to diminishing returns and increase computational cost without significant gains in performance. Increasing the number of trees can act as a form of regularization because it averages out the predictions of multiple trees, reducing the impact of noise or overfitting trees to the overall model.

Depth of Each Tree:

This parameter controls the maximum depth of each decision tree within the random forest. Increasing the depth allows trees to capture more complex patterns and relationships within the data, but could potentially lead to overfitting. On the other hand, decreasing the depth of trees can promote more simple models which are less likely to overfit, but not as powerful. By limiting the depth of each tree, it can have a regularization effect as it prevents overfitting and encourages better generalization on unseen data.

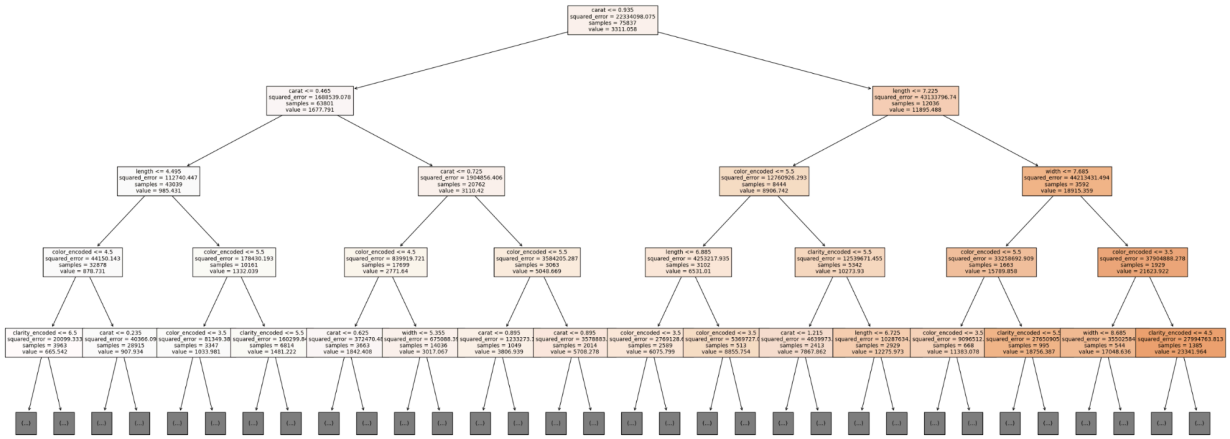
Question 7.2:

How do random forests create a highly non-linear decision boundary despite the fact that all we do at each layer is apply a threshold on a feature?

Random forest can create a highly non-linear decision boundary through the combination of ensemble learning, random subsampling, and random feature selection. The ensemble nature of random forests enables the decision trees to capture complex non-linear relationships in the data.

Question 7.3:

Randomly pick a tree in your random forest model (with maximum depth of 4) and plot its structure. Which feature is selected for branching at the root node? What can you infer about the importance of this feature as opposed to others? Do the important features correspond to what you got in part 3.3.1?



The feature selected for the root node is carat, which corresponds to the results below listing the features by order of importance with carat as the most important feature. Part 3.3.1 corresponded to the application of linear regression. The only question regarding important features in that section corresponds to question 4.4 which relates the p-values to the significance of the feature. Based on this question, all features were shown to be statistically significant except for polish_encoded. Those results do correspond to the important features obtained in this section. A better comparison may be with the results obtained in section 3.1.4 where feature selection was used to determine the features with the greatest impact on the model. Those results returned carat, width, length, depth, color_encoded, and clarity encoded as the top features which directly correspond with the results obtained in this section, as seen below.

Feature	Importance Score
carat	0.499863
length	0.219717
width	0.171990
color_encoded	0.062456
clarity_encoded	0.044062
depth	0.001911

Question 7.4:
Measure “Out-of-Bag Error” (OOB). Explain what OOB error and R2 score means.

OOB Error = 0.02048
R2 Score = 0.98013

Out-of-Bag Error or OOB is a metric used in learning methods such as Random Forests. It is an estimate of how well the Random Forest model will generalize to unseen data, with a lower value indicating better performance.

The R2 score, also known as the coefficient of determination, is a measure of how well features explain the variance in the target variable. R2 scores range from 0 to 1, where 1 indicates a perfect fit and 0 indicates that the model does not explain any of the variance in the target variable.

Based on the values obtained for OOB Error and R2 Score, we are seeing strong performance for this model as the OOB error is very low and the R2 score is very close to 1.

LightGBM, CatBoost, and Bayesian Optimization

Question 8.1:
Read the documentation of LightGMB or CatBoost and determine the important hyperparameters along with a search space for the tuning of these parameters (keep the search space small).

LightGMB Parameters:

```
param_space = {
    'num_leaves': (10, 50),              # Number of leaves in each tree
    'learning_rate': (0.01, 0.1),        # Learning rate
    'max_depth': (3, 10),                # Maximum depth of each tree
    'min_child_samples': (20, 100),      # Minimum number of samples required to form a leaf node
    'reg_alpha': (0.1, 1.0),             # L1 regularization term
    'reg_lambda': (0.1, 1.0)             # L2 regularization term
}
```

Number of leaves:
This parameter defines the maximum number of leaves in one tree and is important for controlling the complexity of the trees.

Learning Rate:
The learning rate shrinks the contribution of each tree. A lower learning rate generally requires more trees to achieve a similar level of performance.

Max Depth:
This parameter controls the maximum depth of a tree. By limiting the maximum depth of the tree nodes, it controls the complexity of the tree.

Min Child Samples:
This parameter controls the minimum number of data points required in a child or leaf node.

Reg Alpha:
This parameter, also known as L1 regularization term, adds a penalty to the absolute value of the weights in order to prevent overfitting.

Reg Lambda:
This parameter, also known as L2 regularization term, adds a penalty to the square of the weights in order to prevent overfitting.

Question 8.2:
Apply Bayesian optimization using skopt.BayesSerachCV from scikit-optimize to find the ideal hyperparameter combination in your search space. Keep your search space small enough to finish running on a single Google Colab instance within 60 minutes. Report the best hyperparameter set found and the corresponding RMSE.

Best Hyperparameters: OrderedDict([('learning_rate', 0.0853649519997956), ('max_depth', 9), ('min_child_samples', 44), ('num_leaves', 48), ('reg_alpha', 0.8777151239184556), ('reg_lambda', 0.15608164675966435)])

RMSE on Test Set: 627.2056089845773

Best Hyperparameters:
Learning Rate = 0.085
Max Depth = 9
Min Child Samples = 44
Number of Leaves = 48
Reg Alpha = 0.877
Reg Lambda = 0.156
RMSE = 627.205

Question 8.3:

Qualitatively interpret the effect of the hyperparameters using the Bayesian optimization results: Which of them helps with performance? Which helps with regularization (shrinks the generalization gap)? Which affects the fitting efficiency?

Number of leaves:

The number of leaves directly impacts the complexity of the tree structures. A moderate number of leaves helps us balance capturing data patterns while preventing overfitting. It has a direct impact on regularization as well, with more leaves leading to overfitting and less leaves resulting in a more simple, generalized model.

Learning Rate:

The learning rate controls the step size during the gradient descent optimization. A learning rate of 0.085 allows us to balance between convergence speed and accuracy of the model when finding the optimal solution. The learning rate also indirectly affects regularization by influencing the convergence behavior. A moderate learning rate, like 0.085, helps prevent overfitting during training.

Max Depth:

The max depth controls the complexity of the individual trees in the random forest and a max depth of 9 allows us to balance between capturing more complex patterns in the data, while still preventing overfitting. Also limiting the max depth, limits the complexity and acts as a form of regularization. With this choice we're able to prevent the trees from becoming overly complex and fitting to the noise in the data.

Min Child Samples:

This parameter controls the minimum number of samples required to create a leaf node in a decision tree. A minimum child sample value of 44 indicates that each leaf node in the tree must contain at least 44 samples, which helps to promote generalization in the model. A higher minimum child samples parameter acts as a form of regularization by preventing the trees from splitting too deeply and fitting the noise in the data.

Reg Alpha:

This regularization parameter works by adding a penalty to the model's weights in order to prevent overfitting. The value directly controls the model's regularization strength and a higher value indicates a stronger regularization effect. This helps to increase generalization and discourages overly complex models.

Reg Lambda:

This regularization parameter works by adding a penalty to the model's weights in order to prevent overfitting. The value directly controls the model's regularization strength and a higher value indicates a stronger regularization effect. This helps to increase generalization and discourages overly complex models.

Part 2: Define Your Own Task - Twitter Data

Question 9.1:

Report the following statistics for each hashtag:

- Average number of tweets per hour
- Average number of followers of user posting the tweets per tweet (to make it simple, we average over the number of tweets; if a user's posted twice, we count the user and the user's followers twice as well)
- Average number of retweets per tweet

ECE219_tweet_data/tweets_#gohawks.txt
Average number of tweets per hour: 292.48785062173687
Average number of followers of users posting the tweets per tweet: 2217.9237355281984
Average number of retweets per tweet: 2.0132093991319877

ECE219_tweet_data/tweets_#gopatriots.txt
Average number of tweets per hour: 40.954698006061946
Average number of followers of users posting the tweets per tweet: 1427.2526051635405
Average number of retweets per tweet: 1.4081919101697078

ECE219_tweet_data/tweets_#nfl.txt
Average number of tweets per hour: 397.0213901819841
Average number of followers of users posting the tweets per tweet: 4662.37544523693
Average number of retweets per tweet: 1.5344602655543254

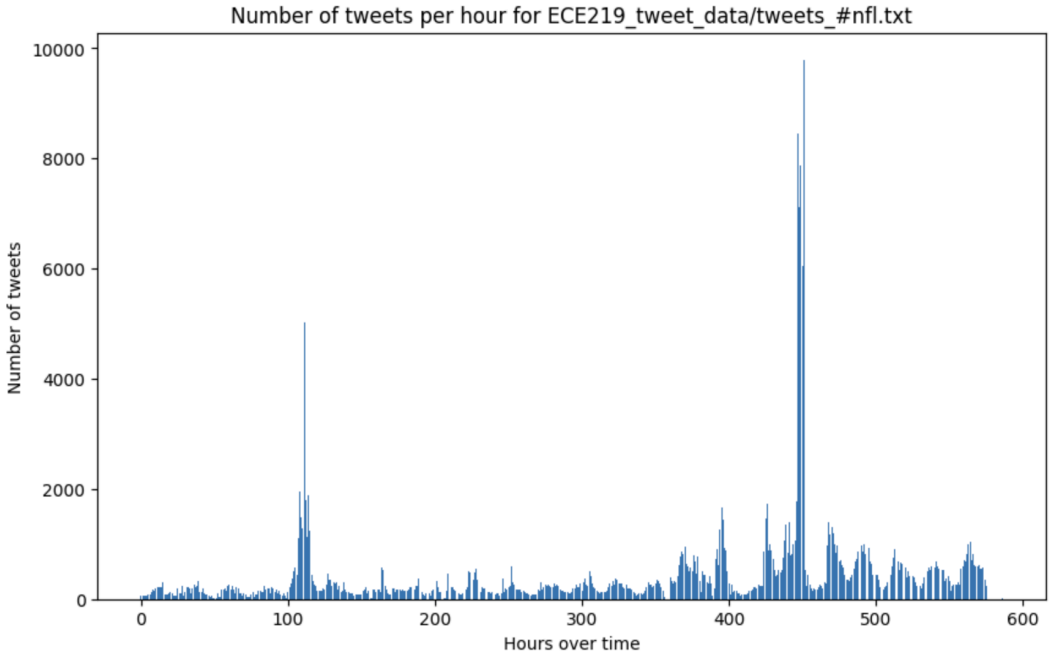
ECE219_tweet_data/tweets_#patriots.txt
Average number of tweets per hour: 750.8942646068899
Average number of followers of users posting the tweets per tweet: 3280.4635616550277
Average number of retweets per tweet: 1.7852871288476946

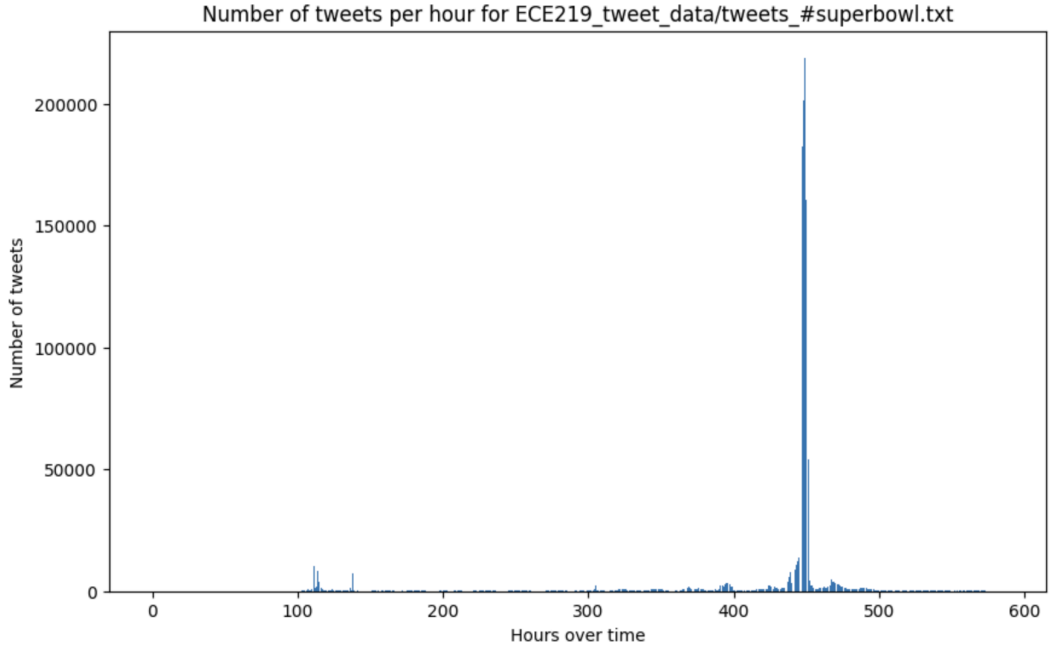
ECE219_tweet_data/tweets_#sb49.txt
Average number of tweets per hour: 1276.8570598680474
Average number of followers of users posting the tweets per tweet: 10374.160292019487
Average number of retweets per tweet: 2.52713444111402

ECE219_tweet_data/tweets_#superbowl.txt
Average number of tweets per hour: 2072.11840170408
Average number of followers of users posting the tweets per tweet: 8814.96799424623
Average number of retweets per tweet: 2.3911895819207736

Question 9.2:

Plot “number of tweets in an hour” over time for #SuperBowl and #NFL (a bar plot with 1-hour bins). The tweets stored in separate files for different hashtags and files are named as tweet_[#hashtag].txt.





Question 10:

Describe your Task:

For the purpose of this project, I opted to further explore the third option given within the project guidelines. The task is defined below.

Library of Prediction Tasks give a tweet: Predict the hashtags or how likely it is that a tweet belongs to a specific team fan. Predict the number of retweets/likes/quotes. Predict the relative time at which a tweet was posted.

For each of these tasks the data was explored, feature engineering options were optimized while balancing computational complexity, machine learning models were built, and models were evaluated based on the appropriate performance metrics. These performance metrics were compared to baseline models to provide a reference point for understanding the quality of the machine learning model performance.

Explore the Data:

Preprocessing of the data was completed prior to addressing the task described in the previous section. First, given the large number of tweets found within all six text files, a subsample of the data was taken in order to reduce the overall computational load of the task. For each of the six files, 5% of the data was pulled to represent that data file within the subsample.

There was a large amount of data provided within each column of the original twitter text files. In order to clean up the data and make the necessary features easier to access, the following features were pulled from the original twitter text files and used as the columns in the new data frame: Time of Day, Number of Tweets, Number of Retweets, Number of Followers, Tweet Text, Hashtag, and Hashtag Label, which indicates the hashtag text file the data originates from.

Further data cleaning consists of utilizing a label encoder on all categorical data in preparation for use in the machine learning models that are discussed later on. Specifically, this is necessary for the task of predicting the hashtags associated with each data point, as there are 6 different categories associated with the hashtag label feature.

Lastly, after data has been processed, cleaned, and converted to a dataframe, we must account for the potential of missing data in the dataset. Options for addressing missing data include taking the mean or median of data for imputation or dropping data rows associated with no data or NaN values. For this project we opted for dropping data rows associated with NaN or no data values for any of the features.

Before the data is utilized for the machine learning models all datasets for all tasks addressed in this project are split into a training set of 80%, used for training the machine learning model, and a testing set of 20% used for evaluating the performance of said models.

Feature Engineering:

For the purposes of feature engineering, the approaches will be described for each of the three features we are looking to train and test our model on for prediction purposes.

The first task at hand is predicting the hashtags associated with each datapoint in the set. For predicting the hashtag, the technique of TF-IDF is used to represent and encode the tweet text for analysis. In addition to this, we can utilize the following numerical features for the purposes of training in case there are any underlying patterns within the data: Number of Tweets, Number of Retweets, Number of Followers, and Time of Day.

The next task is to predict the number of retweets associated with each datapoint in the set. For predicting the number of retweets, we again use the technique of TF-IDF for representing the tweet text for analysis. TF-IDF, or term frequency-inverse document frequency, is a statistical measure used to evaluate how important a word is to a document, or in this case tweet, in a collection. The TfidfVectorizer is used to convert the text data into numerical features. In this case we limit the number of features to 500 terms to reduce computational cost. In addition to this, we can utilize the following numerical features from the dataset: number of followers, time of day. Numerical features are then standardized in order to scale the numerical features to have zero mean and unit variance, in order to improve the performance of the machine learning algorithm. Lastly, to assist with the computational cost of this model, dimensionality reduction is conducted with PCA, or principal component analysis. This projects the data onto a lower-dimensional subspace while preserving the maximum variance. By reducing the dimensionality of the combined feature matrix we can reduce the computational complexity of the model and potentially improve the model's ability to generalize well to unseen data.

The final task is to predict the relative time in which a tweet was posted. For this model, we utilize one-hot encoding to convert the categorical variable of hashtag label into a numerical format. This way each category in the hashtag label variable is represented as a binary feature column, allowing the linear regression model to interpret and use the categorical variables effectively. The tweet text data is then transformed into numerical features as well, using TF-IDF vectorization. This assigns weights to the words based on their frequency in a tweet and inverse frequency across all tweets. All numerical features are then combined into a feature matrix for the use of training the linear regression model, which is further discussed in the following section.

Machine Learning Model:

For the purposes of the machine learning model, the approaches will be described for each of the three features we are looking to train and test our model on for prediction purposes.

Before building our machine learning models, we first need to establish baselines. For each prediction task, we can start with simple baselines such as mean/mode prediction or linear regression using basic features. These baselines provide a benchmark for evaluating the performance of more complex machine learning models. The results of the benchmark models are included in the evaluation section alongside the machine learning model performance metrics for comparison purposes.

First, for the purpose of predicting hashtags associated with each data point in the dataset, our machine learning model is a random forest model. The random forest classifier is an ensemble learning method for classification tasks. The model constructs multiple decision trees during training and outputs the mode of the classes for classification. For this model we utilized 100 estimators, which specifies the number of trees in the forest. The model is then evaluated for its performance on the 20% split of unseen testing data. The results of this model, and its corresponding baseline model, are discussed in the following section.

Next, for predicting the number of retweets associated with a data point, a random forest regression model was utilized. In order to improve model performance, random search cross validation was used for tuning the hyperparameters. The number of estimators, maximum depth, and minimum number of samples split are specified within the parameters distribution dictionary. The randomized search cross validation then performs a randomized search over the defined hyperparameter space for the specified number of iterations, in this case 8, to find the best combination of hyperparameters. This way we are able to iterate over multiple different model configurations and obtain the best random forest regression model through the training data. From there we are able to make predictions on the test set, and evaluate the model performance on unseen test data. The results of this model, and its corresponding baseline model, are discussed in the following section.

Last, for predicting the relative time in which the tweet was posted, a simple linear regression model is utilized as the predictive model for this task. This model fits a linear relationship between the independent variables, or features, and the dependent variable, or target. In this case the target variable is the relative time in which the tweet was posted. A training and testing split of 80% to 20% is utilized. During training, the model learns the coefficients or weights for each feature and the intercept terms, which it then utilizes to predict the target variable for the unseen test data. The results of this model, and its corresponding baseline model, are discussed in the following section.

Evaluation:

For the purposes of the performance evaluation, the approaches will be described for each of the three features we are looking to train and test our model on for prediction purposes.

Evaluation is crucial to assess the performance of our models accurately. For hashtag prediction or team fan affiliation, we can use metrics like accuracy or F1 score. For engagement prediction, we can use metrics like Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE) for regression tasks, or metrics like precision, recall, and F1 score for classification tasks.

When evaluating categorical data like the 6 different hashtag labels that the twitter data is based on, the metrics used for measuring the performance consist of precision, recall, and f1-score. The precision measures the accuracy of positive predictions made by the model and is represented by the ratio of true positive predictions to the total number of positive predictions. The recall measures the ability of the model to capture all positive instances in the dataset and is represented by the ratio of true positive predictions to the total number of actual positive instances in the dataset. Lastly, the f1-score provides a single metric that balances precision and recall for comparing models and understanding their overall performance in binary classification tasks such as this. The results of these performance metrics can be seen below for the machine learning model used to predict the hashtag associated with each data point.

Table 1: Performance Metrics for Predicting Hashtags

Hashtag	Precision	Recall	F1-Score	Support
gohawks	0.90	0.93	0.91	1681
gopatriots	0.87	0.84	0.86	223
nfl	0.82	0.78	0.80	2331
patriots	0.89	0.88	0.88	4470
sb49	0.99	0.99	0.99	7437
superbowl	0.95	0.97	0.96	12097

For the purpose of evaluating the performance of the model, a baseline model was used to establish a benchmark. The baseline model results, which utilizes a basic strategy where labels are randomly assigned to instances based on the class distribution observed in the training data, can be seen below.

Table 2: Baseline Model Performance Metrics for Predicting Hashtags

Hashtag	Precision	Recall	F1-Score	Support

gohawks	0.06	0.06	0.06	1681
gopatriots	0.01	0.01	0.01	223
nfl	0.10	0.10	0.10	2331
patriots	0.15	0.15	0.15	4470
sb49	0.26	0.26	0.26	7437
superbowl	0.42	0.43	0.43	12097

If we look at the machine learning model performance metrics, shown in Table 1, and the baseline machine learning model performance metrics, shown in Table 2, we can see that the performance across all metrics is stronger in the machine learning model than in the baseline model. This shows that our feature engineering and machine learning model are effective for predicting hashtag labels for a given dataset, and results in better predictive performance than a standard baseline model.

For the next task, the model was evaluated for predicting the number of retweets associated with a data point. The following metrics were considered when evaluating the performance of the machine learning model: mean absolute error, mean squared error, root mean squared error, and r-squared.

The mean absolute error measures the average absolute difference between the predicted values and the actual values. A lower mean absolute error indicates better performance. The mean squared error measures the average squared difference between the predicted values and the actual values. It penalizes large errors more than mean absolute error and is useful for understanding the spread of errors. Again, a lower mean squared error indicates better performance. The root squared mean error is expressed in the same units as the target variable, making it a more interpretable measure of the average error. Lastly, the r-squared score is a measure of the proportion of the variance in the target variable that is explained by the model. The r-squared score ranges from 0 to 1, where 1 indicates a perfect fit. The results for both the machine learning model and the baseline model are shown below.

Original Machine Learning Model Evaluation Metrics:
Mean Absolute Error: 5.535046352890463
Mean Squared Error: 365.61456975189583
Root Mean Squared Error: 19.121050435368236
R-squared (R^2) Score: -0.6435524383198834

Machine Learning Model Optimized with Randomized Search CV Evaluation Metrics:
Mean Absolute Error: 2.3306063208218726
Mean Squared Error: 319.5926803656248
Root Mean Squared Error: 17.877155264907916
R-squared (R^2) Score: -0.43666957649021865

Baseline Model Evaluation Metrics:
Mean Absolute Error: 2.86311979720438
Mean Squared Error: 222.95390778487854
Root Mean Squared Error: 14.931641161803968
R-squared (R^2) Score: -0.002247911021285187

If we look at the machine learning model performance metrics we can see that the original model performance was not as strong as we would prefer. To gain better results, an optimized model utilizing randomized search cross validation was used. The results did improve, providing a better mean absolute error value obtained in the optimized model than seen in the baseline model. Other than this metric, we still see better performance within the baseline model across mean squared error, root mean squared error, and r-squared score. This indicates that our machine learning model is not producing better predictive performance than a standard baseline model.

In order to improve the model to achieve better performance than the baseline model, there are a few areas we could look to optimize. First, we could look into better aligning the feature engineering by adding more relevant features to help the model better capture patterns in the data. This would involve going back to the data exploration and data cleaning steps and pulling additional features from the original dataset. Another option would be to experiment with different types of regression algorithms, other than random forest, such as gradient boosting regressor, support vector regressor, or neural network. Different algorithms may capture different aspects of the data better, potentially resulting in better performance. Lastly, we could further tune the hyperparameters, defining more value within the parameters distributions dictionary. This does build additional computational complexity into the model, requiring infinitely more time to evaluate the models, and pushing outside the scope of this project.

For the final task, the model was evaluated for predicting the relative time that a tweet was posted. The following metrics were considered when evaluating the performance of the machine learning model: mean absolute error, mean squared error, root mean squared error, and r-squared.

The mean absolute error measures the average absolute difference between the predicted values and the actual values. A lower mean absolute error indicates better performance. The mean squared error measures the average squared difference between the predicted values and the actual values. It penalizes large errors more than mean absolute error and is useful for understanding the spread of errors. Again, a lower mean squared error indicates better performance. The root squared mean error is expressed in the same units as the target variable, making it a more interpretable measure of the average error. Lastly, the r-squared score is a measure of the proportion of the variance in the target variable that is explained by the model. The r-squared score ranges from 0 to 1, where 1 indicates a perfect fit. The results for both the machine learning model and the baseline model are shown below.

Machine Learning Model Performance Metrics:
Mean Absolute Error: 6.293879696934464e-13
Mean Squared Error: 7.7329435180892385e-25
Root Mean Squared Error: 8.793715664091737e-13
R-squared (R^2) Score: 1.0

Baseline Model (Mean Predictor):
Mean Absolute Error: 4.717447501682071
Mean Squared Error: 32.07096568575375
Root Mean Squared Error: 5.663123315428841
R-squared (R^2) Score: -0.016396757844456156

If we look at the machine learning model performance metrics and the baseline machine learning model performance metrics, we can see that the performance across all metrics is stronger in the machine learning model than in the baseline model. This shows that our feature engineering and machine learning model are effective for predicting the relative time that a tweet is posted for a given dataset, and results in better predictive performance than a standard baseline model.