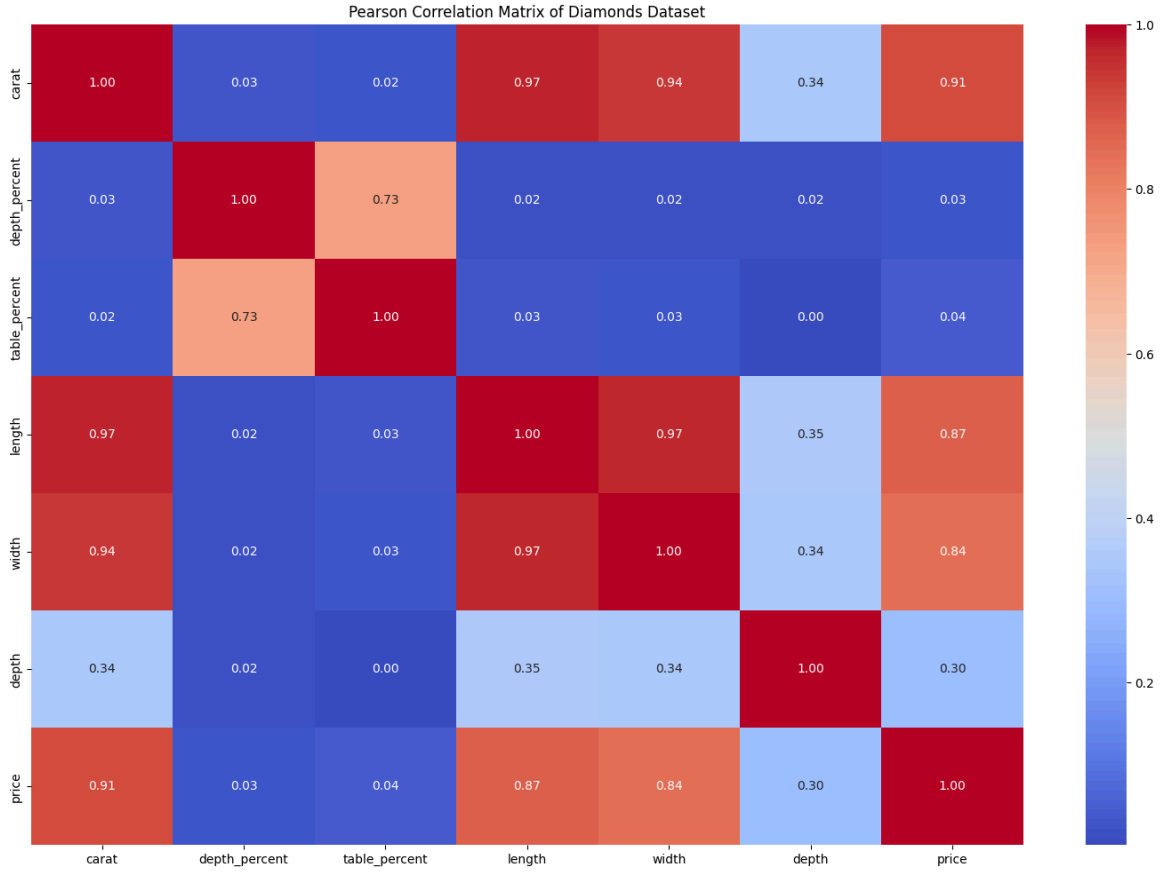
Project 4: Regression Analysis
Tania Rajabally
UID: 806153219

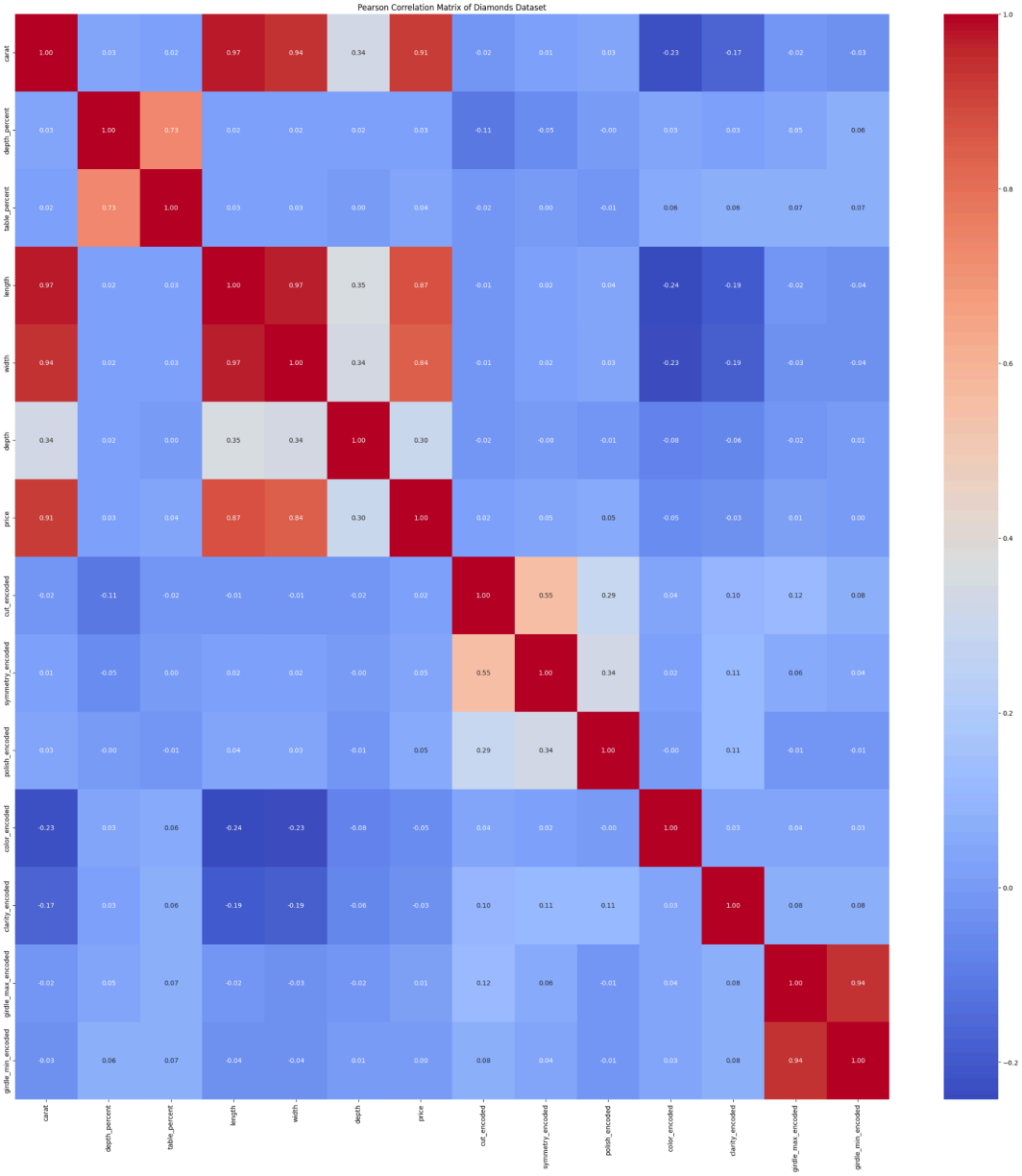
QUESTION 1:

- **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.**

  Below is the heatmap of the Pearson correlation matrix before encoding the columns.



We then encode all the categorical data and plot the heatmap after encoding and assigning some categorical features to a scalar. We get the below heatmap.

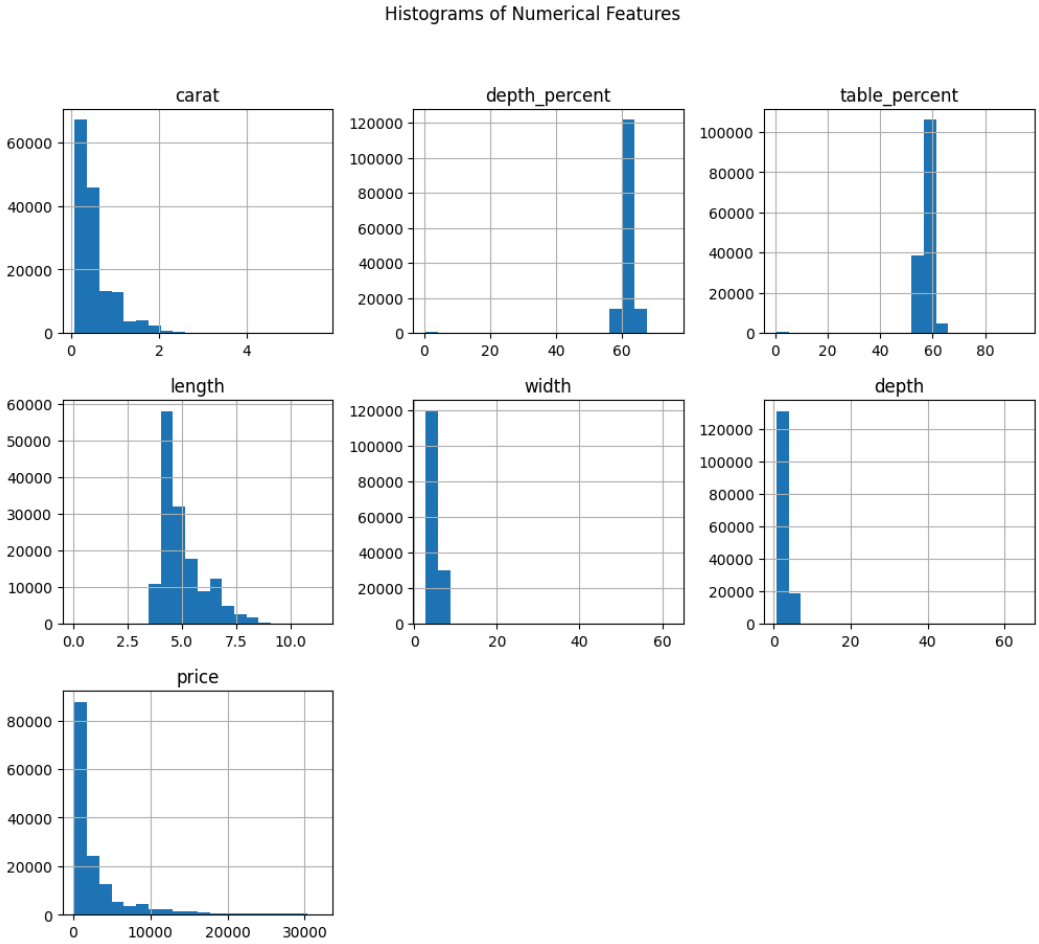


The value of Pearson coefficient correlation ranges from -1 to 1. -1 indicates a negative correlation. This means that an increase in one variable will lead to a decrease in the other variable. Whereas a positive value implies that an increase or decrease in one variable will lead to an increase or decrease in the other variable respectively. For the diamond characteristic dataset, we can see the correlations below:
Features with the highest absolute correlation with the target variable (price):

carat	0.913479
length	0.869521
width	0.841887
depth	0.299696
polish_encoded	0.054928
color_encoded	0.047189
symmetry_encoded	0.047149
table_percent	0.042453
clarity_encoded	0.026788
depth_percent	0.025469

cut_encoded 0.024356
girdle_max_encoded 0.013336
girdle_min_encoded 0.000814
We can see that carat has the highest correlation with the target variable price. Length and width also matter since they too have a high correlation. This means that carat along with length and width are the most informative while figuring out the price of a diamond. This aligns with the market pattern that the size and the carats of the diamond are very critical factors affecting the price of the diamond.

- **Question 1.2 - Plot the histogram of numerical features. What preprocessing can be done if the distribution of a feature has high skewness?**
  Below is the histogram of all the numerical features.



The skewness of the features is as below:
carat 2.331750
depth_percent -13.559472
table_percent -11.046453
length 1.283591
width 4.115307
depth 27.493024
price 3.071707

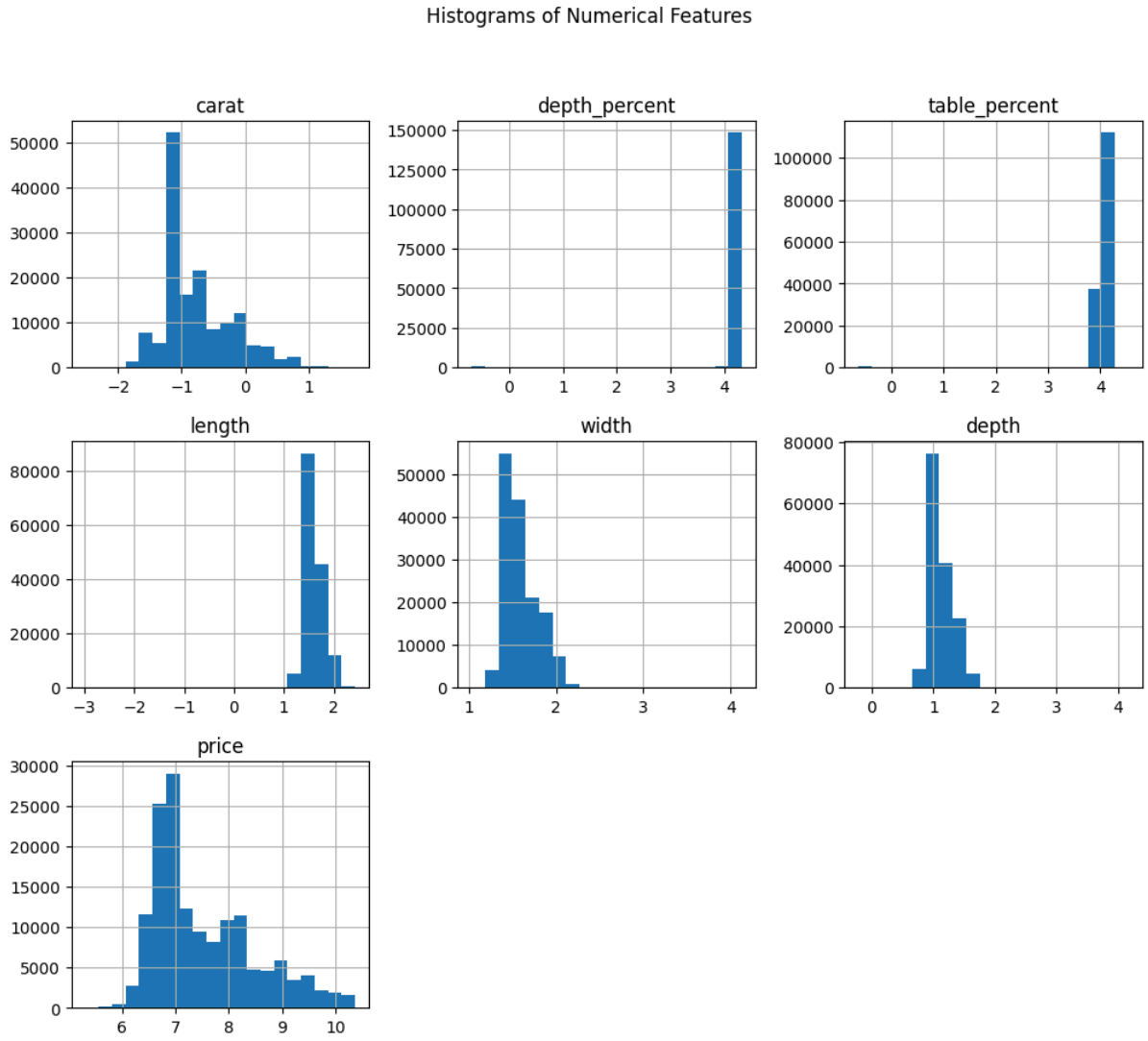
We can see that the features above have a high skewness. This means that there are expensive outliers. When a feature exhibits high skewness in its distribution, meaning it is not symmetrically distributed around

the mean, several preprocessing techniques can be applied to mitigate this skewness. Preprocessing steps like standardization can be done so that extreme values do not cause harm to the robustness of the model. Another common technique used is logarithmic transformation. This can help reduce the magnitude of extreme values and bring the distribution closer to a normal distribution. It mitigates the impact of highly skewed values while preserving the relative difference between smaller values.

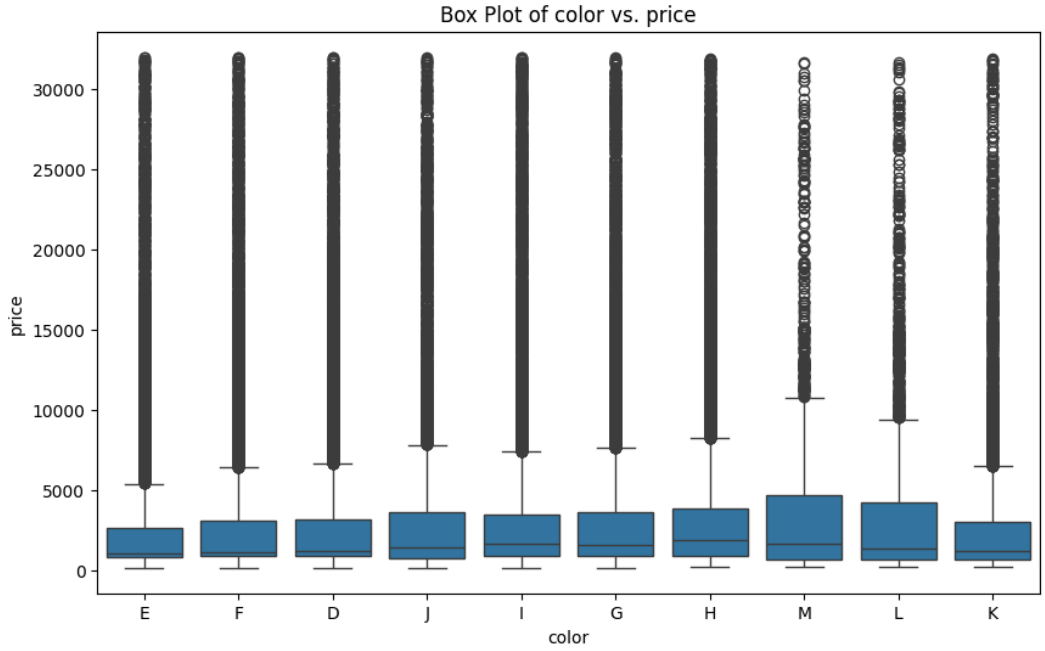
The skewness after logarithmic transformation is:

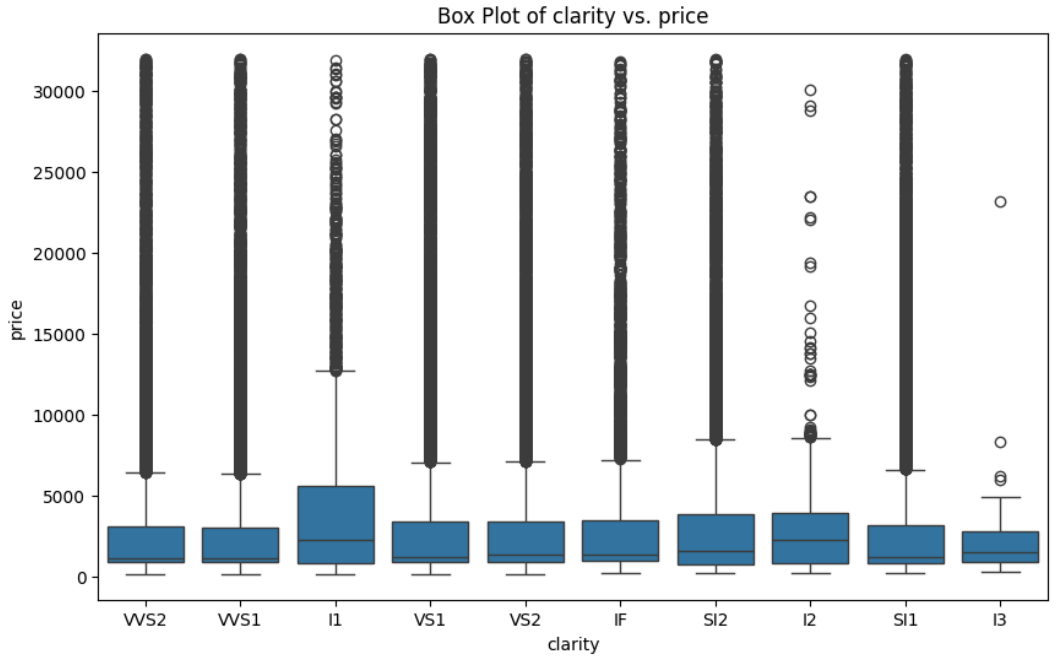
carat	0.844160
depth_percent	-16.200465
table_percent	-16.015117
length	0.734470
width	0.930143
depth	3.361822
price	0.897955

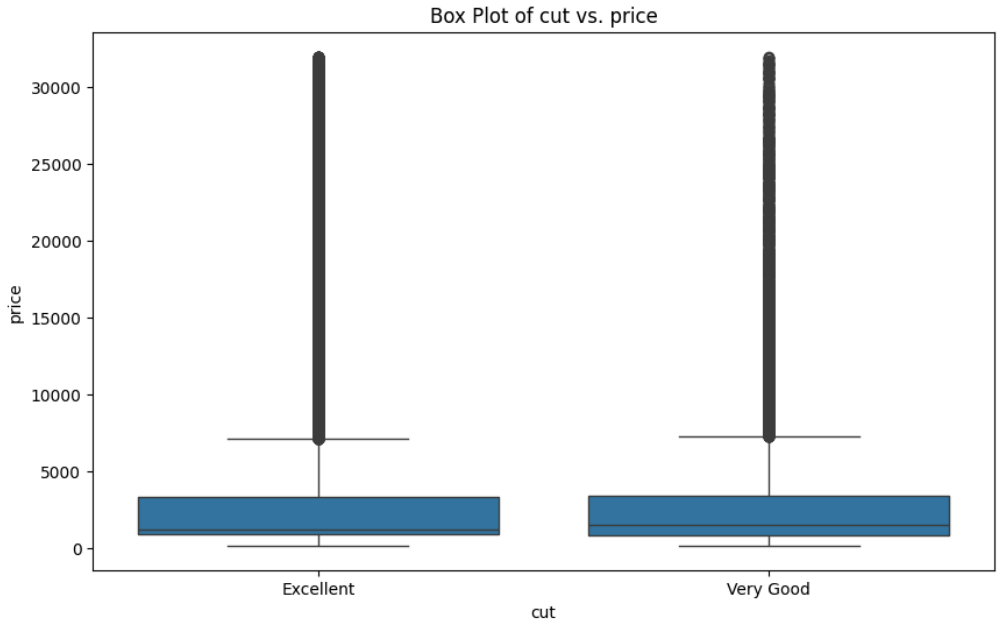
Below is the histogram of the features after logarithmic transformation

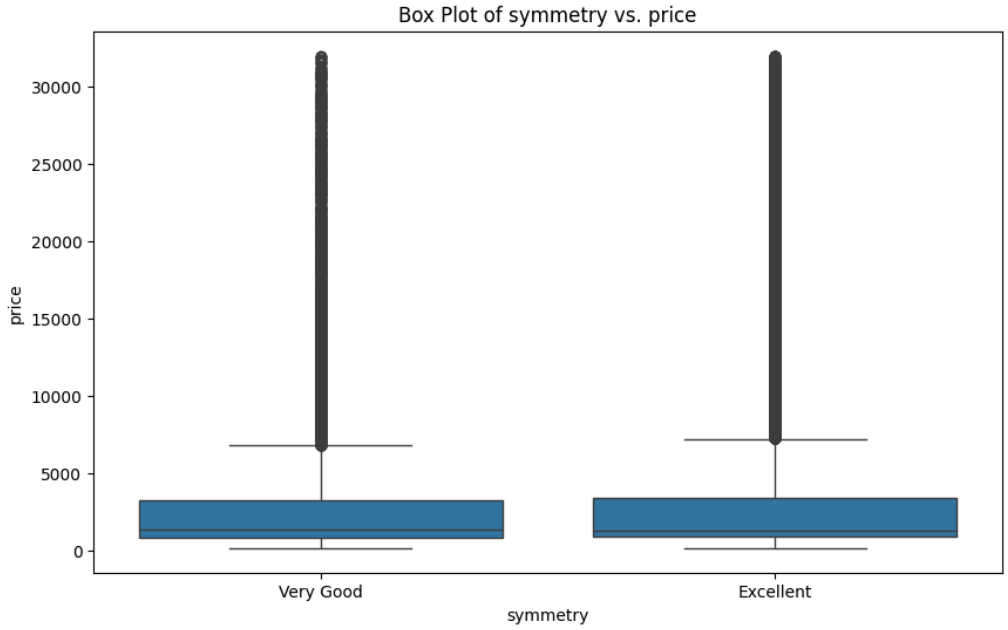


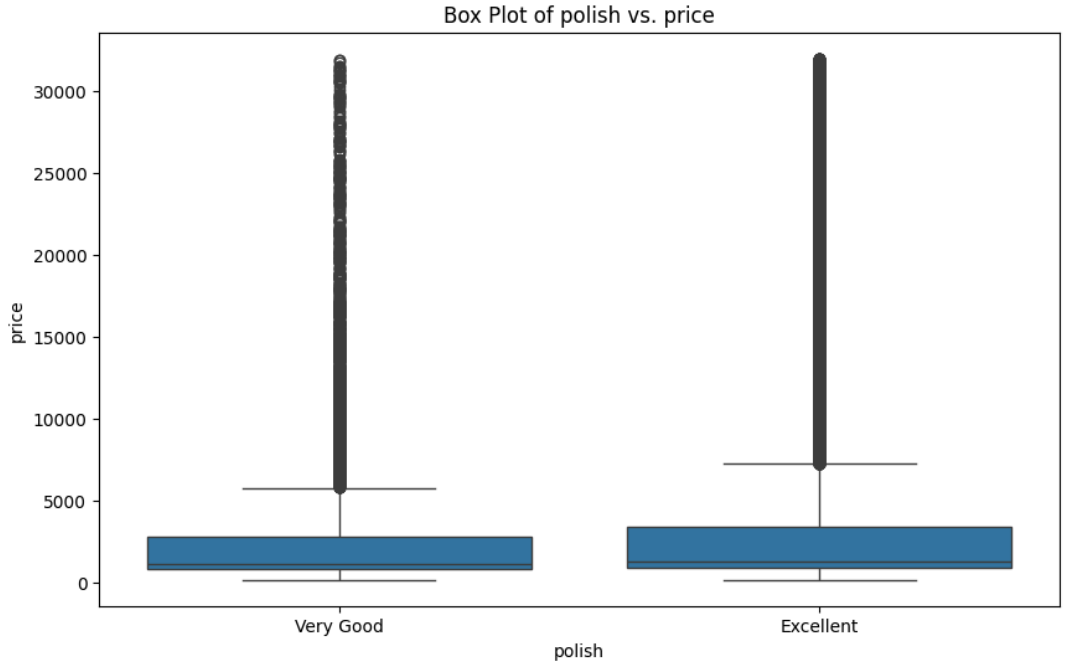
- **Question 1.3 - Construct and inspect the box plot of categorical features vs target variable. What do you find?**

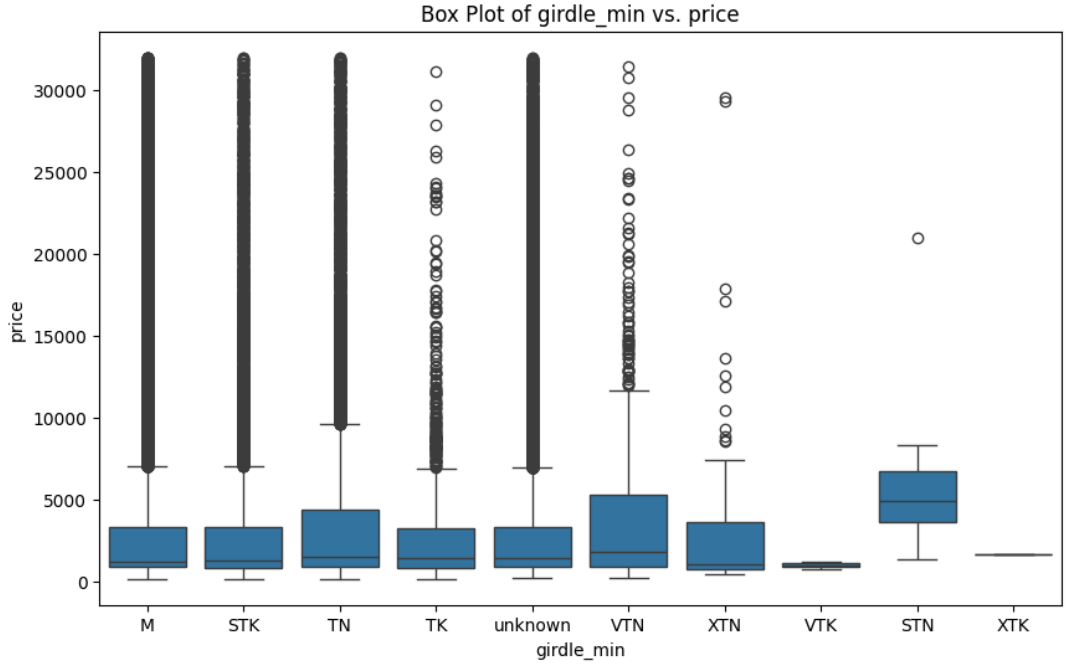


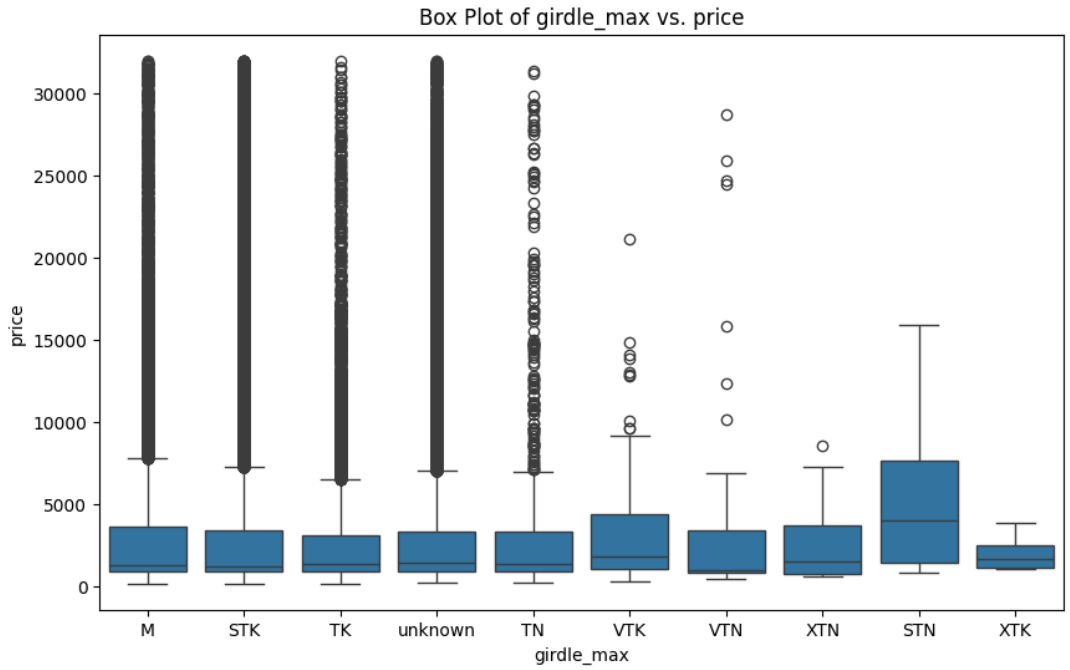



Box Plot of cut vs. price
price
30000
25000
20000
15000
10000
5000
0
Excellent
Very Good
cut


Box Plot of symmetry vs. price
price
30000
25000
20000
15000
10000
5000
0
Very Good
Excellent
symmetry


Box Plot of polish vs. price
price
30000
25000
20000
15000
10000
5000
0
Very Good
Excellent
polish

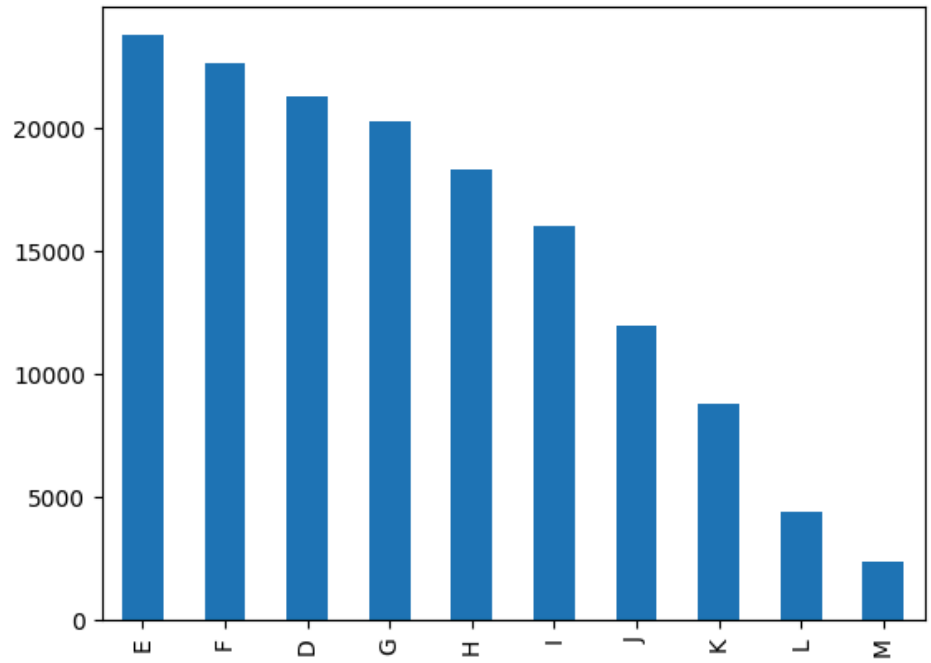

Box Plot of girdle_min vs. price
price
30000
25000
20000
15000
10000
5000
0
M
STK
TN
TK
unknown
VTN
XTN
VTK
STN
XTK
girdle_min



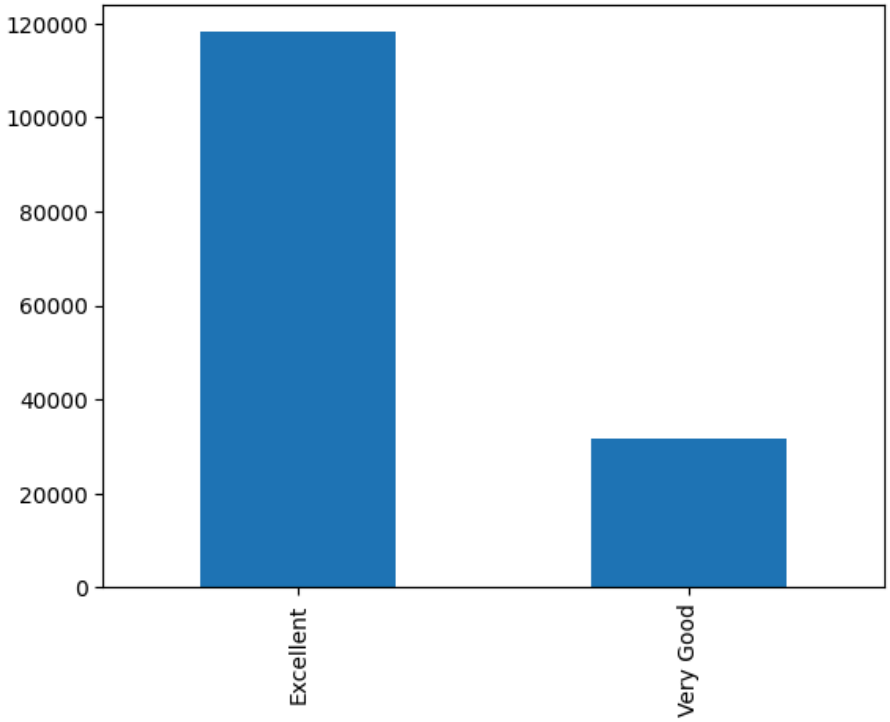
The box plot is an efficient way to capture information about the distribution of the features. The middle box displays the interquartile range which is the data points that lie within the 25th percentile to 75th percentile in the dataset. The extremes show the minimum and maximum quantities. From the box plot of cut, we can see that the median for very good is higher. For clarity, we can see that I1 had a higher price than IF. In reality, we expect a converse relation. Same is the case for color. From the clarity plot we can see that some categories are easily predictable by the model. For color, we see that E,F and D are highly skewed.

- **Question 1.4 - For the Diamonds dataset, plot the counts by color, cut and clarity.**

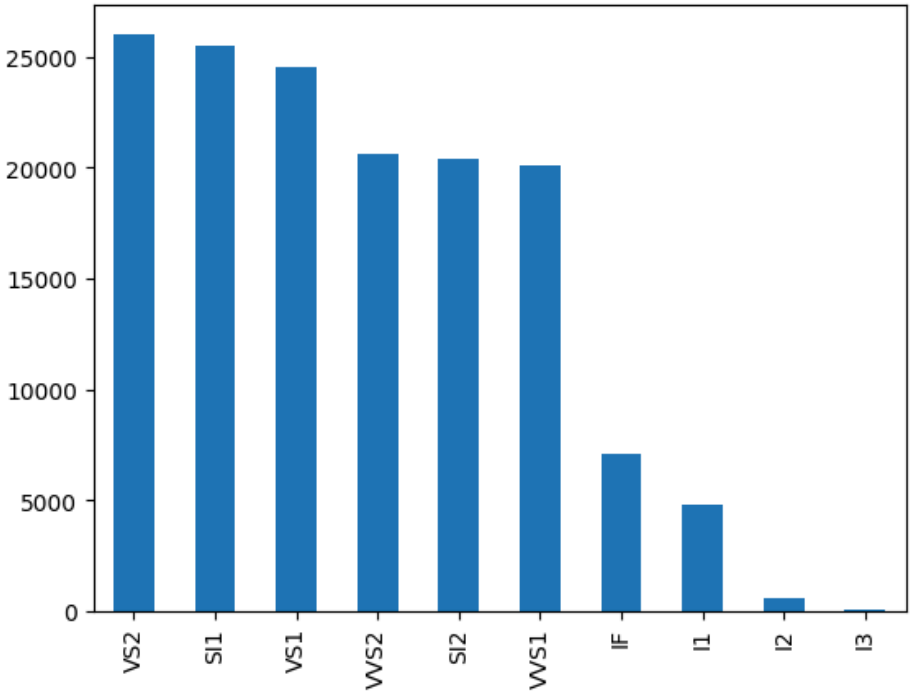
  Plot of counts by color:



  Plot of counts by cut:



Plot of counts by clarity:



QUESTION 2:

- **Question 2.1 - Standardize feature columns and prepare them for training.**
  Standard Scaler was used to standardize the feature columns.

- **Question 2.2 - You may use these functions to select features that yield better regression results (especially in the classical models). 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 w.r.t to the target.

Feature selection is very important for regression models as it selects the important features and only use those. Both mutual information (MI) regression and F regression are commonly used techniques for feature selection. MI Regression checks how dependent two random variables are i.e how dependent the feature and target variable are. In our case the target variable is price. The higher the value means there is more dependency. This means that it contributes highly to the target variable. So if we have a high value, it means that this feature affects the price significantly. By selecting the values with higher MI regression, we are reducing our dataset to only the important features which in turn improves the performance of the model. F scores measure the significance of the improvement in the model's performance when adding new variables (features). The higher the value, the more significant the variable. This means that adding this particular feature with a high value will improve the performance of the model. By selecting features with higher F scores, we can prioritize features that contribute the most to the model's performance.
Using these methods help improve model performance specially regression models by reducing overfitting and using only the important features. The model can better capture the underlying patterns leading to improved generalization performance and lower test root mean squared error (RMSE). The effectiveness of these depends on the dataset and the regression model used. Linear regression models benefit drastically since they are sensitive to multicollinearity and overfitting. But, more complex models like tree based algorithms inherently do feature selection and hence using MI regression or F score will not make a lot of difference.
The two features with the lowest MI w.r.t the target are:
['polish_encoded', table_percent'] an MI of 0.009602 and 0.023415 respectively.
The two features with the lowest F score w.r.t the target are:
['girdle_min_encoded', 'girdle_max_encoded'] with values 0.099306 and 26.659077 respectively.

QUESTION 4:

What is the objective function? Train three models: (a) ordinary least squares (linear regression without regularization), (b) Lasso and (c) Ridge regression, and answer the following questions.

In linear regression, the objective function is to minimize the difference between the observed target values and the values predicted. This means that we are reducing the mean squared error between the target value and the predicted value.
The objective functions are as below:

$$\text{OLS} : \min_{\beta} \|Y - X\beta\|_2^2$$

$$\text{Lasso} : \min_{\beta} \|Y - X\beta\|_2^2 + \lambda\|\beta\|_1$$

$$\text{Ridge} : \min_{\beta} \|Y - X\beta\|_2^2 + \lambda\|\beta\|_2$$

We can see the RMSEs obtained as below:

Unscaled Data:
Average RMSE for Ordinary Least Squares (OLS) for Unscaled Data: 1571.035361861907
Average RMSE for Lasso Regression for Unscaled Data: 1570.88201356099 (Best alpha for Lasso for Unscaled Data: 0.01)

Average RMSE for Ridge Regression for Unscaled Data: 1571.0354586551564 (Best alpha for Ridge for Unscaled Data:: 0.0001)

Scaled Data:
Average RMSE for Ordinary Least Squares (OLS) for Scaled Data: 1571.035361861908
Average RMSE for Lasso Regression for Scaled Data: 1570.8951241421228 (Best alpha for Lasso for Scaled Data: 0.1)
Average RMSE for Ridge Regression for Scaled Data: 1571.0353771863192 (Best alpha for Ridge for Scaled Data:: 0.0001)

Scaled Reduced Data:
Average RMSE for Ordinary Least Squares (OLS) for Reduced Scaled Data: 1581.83183864438
Average RMSE for Lasso Regression for Reduced Scaled Data: 1581.6691554265262 (Best alpha for Lasso for Reduced Scaled Data: 0.1)
Average RMSE for Ridge Regression for Reduced Scaled Data: 1581.8318524986157 (Best alpha for Ridge for Reduced Scaled Data:: 0.0001)

Scaled Reduced Skewed Data:
Average RMSE for Ordinary Least Squares (OLS) for Reduced Skewed Scaled Data: 0.15584804291480306
Average RMSE for Lasso Regression for Reduced Skewed Scaled Data: 0.1558603267153913 (Best alpha for Lasso for Reduced Scaled Data: 0.0001)
Average RMSE for Ridge Regression for Reduced Skewed Scaled Data: 0.15584804319490161 (Best alpha for Ridge for Reduced Scaled Data:: 0.0001)

Scaled Train-Test Split Data:
Lasso CV RMSE: 1619.500087777625 (Best alpha for Lasso: 0.1)
Ridge CV RMSE: 1619.512659888272 (Best alpha for Ridge: 0.0001)
Lasso Test RMSE: 1613.7549246183992 (Best alpha for Lasso: 0.1)
Ridge Test RMSE: 1613.7530377250657 (Best alpha for Ridge: 0.0001)

We can see that scaling and skewing features improves the performance of the model.

- **Question 4.1 - Explain how each regularization scheme affects the learned parameter set.**

Lasso Regression is linear regression with L1 regularization. Whereas Ridge Regression is linear regression with L2 regularization. These prevent overfitting by adding a penalty term to the loss function.
For Lasso Regression, the penalty term is proportional to the absolute values of the coefficients of the features. Lasso Regularization tends to produce sparse solutions, meaning it forces some of the coefficients of the features to be exactly zero. This makes it useful for models that need feature selection or while dealing with high dimensional data that has a lot of irrelevant features. By setting some values to zero, Lasso selects those features that have a high correlation to the target variable. This helps improve the model performance by focusing on the important features. The shrinkage performed prevents overfitting of the model by reducing the variance. The amount of regularization in Lasso is controlled by a hyperparameter called the regularization parameter (lambda or alpha). Increasing the regularization parameter increases the amount of regularization, which in turn increases the level of sparsity in the learned parameter set. However, higher regularization also increases bias while reducing variance. Therefore, there is a trade-off between bias and variance that needs to be carefully balanced when tuning the regularization parameter.

For Ridge Regularization, the penalty term is proportional to the squared magnitudes of the coefficients of the features. One of the key effects of Ridge regularization is that it shrinks the coefficients of the features towards zero. Unlike Lasso regularization, which can set coefficients exactly to zero, Ridge regularization only reduces the magnitude of the coefficients. This also prevents overfitting. Ridge regularization does not include sparsity. Ridge regression tends to retain all features in the model, even those that are less important for predicting the target variable. This can be advantageous when all features are potentially relevant and excluding any feature could lead to loss of information. Ridge regularization also has alpha which controls the regularization. Ridge regularization is particularly effective when dealing with multicollinear features, meaning features that are highly correlated with each other.
Hence L2 regularization i.e Ridge regularization is useful for shrinkage purposes whereas L1 regularization i.e Lasso Regularization is useful for feature selection.

- **Question 4.2 Report your choice of the best regularization scheme along with the optimal penalty parameter and explain how you computed it.**

We tried regression on unscaled data, scaled data, scaled reduced data and scaled reduced skewed data. Multiple values of alpha were tried to check which gives the lowest RMSE. The values for all the experiments can be found above.
We can see for the scaled reduced data with the top 6 features, we received the following RMSEs:
Average RMSE for Ordinary Least Squares (OLS) for Reduced Scaled Data: 1581.83183864438
Average RMSE for Lasso Regression for Reduced Scaled Data: 1581.6691554265262 (Best alpha for Lasso for Reduced Scaled Data: 0.1)
Average RMSE for Ridge Regression for Reduced Scaled Data: 1581.8318524986157 (Best alpha for Ridge for Reduced Scaled Data:: 0.0001)
The various values of alpha tried were from 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000.
We found Lasso Regression to perform the best with a value of alpha as 0.1 and the Average RMSE as 1581.6691554265262.

- **Question 4.3 - Does feature standardization play a role in improving the model performance (in the cases with ridge regularization)? Justify your answer.**

Yes, feature standardization plays an important role in improving the model performance. Ridge regularization penalizes large coefficients in the model by adding a term to the objective function that is proportional to the square of the magnitude of the coefficients. This penalty term aims to prevent overfitting by reducing the influence of individual features on the model's predictions. However this can be affected by the scaling of features. Features with different scales have different impacts. One with large scales has large coefficients after regularization and vice versa. This can lead to biased model estimates. Standardization ensures that all the features contribute equally to the regularization. Standardization also improves the stability and interpretability of the learned coefficients. In summary, by ensuring that all features are on the same scale, standardization helps to improve the effectiveness of ridge regularization by promoting more stable and interpretable model estimates. This, in turn, can lead to improved model performance, better generalization to unseen data, and enhanced interpretability of the model's predictions. The improvement in model performance with standardized features is reflected in a larger difference in the RMSE values obtained when using scaled features compared to unscaled features. This is why feature standardization is very important.
We can see in our case that standardization did not make much of a difference in the values of the RMSE. Feature scaling might be important depending on the type of regression and how the dataset is. If it is not a standard normally distributed data, it will not make much of a difference on the RMSE or might even increase the RMSE.

- **Question 4.4 - Some linear regression packages return p-values for different features2. What is the meaning of these p-values and how can you infer the most significant features? A qualitative reasoning is sufficient.**

In linear regression, p-values associated with each feature indicate the statistical significance of that feature's contribution to explaining the variation in the target variable. A low p-value (typically below a significance level, such as 0.05) suggests that the null hypothesis can be rejected, indicating that the corresponding feature is statistically significant in predicting the target variable. In other words, a low p-value suggests that there is strong evidence that the feature has a non-zero coefficient and contributes significantly to the model. Conversely, a high p-value suggests that there is insufficient evidence to reject the null hypothesis, indicating that the corresponding feature may not be statistically significant in predicting the target variable. Therefore, when interpreting p-values in linear regression, features with low p-values are considered more significant and are likely to have a stronger influence on the target variable. These features are typically retained in the model, while features with high p-values may be considered for removal if they do not contribute significantly to the model's predictive power.

The p values obtained for the reduced scaled dataset are:

const 0.000000e+00
carat 0.000000e+00
length 0.000000e+00
width 4.511933e-03
depth 7.563643e-12
color_encoded 0.000000e+00
clarity_encoded 0.000000e+00

<u>QUESTION 5:</u>

Perform polynomial regression by crafting products of features you selected in part 3.1.4 up to a certain degree (max degree 6) and applying ridge regression on the compound features. You can use scikit-learn library to build such features. Avoid overfitting by proper regularization. Answer the following:

- **Question 5.1 - What are the most salient features? Why?**

Top 10 most salient features with their corresponding coefficients corresponding the the best degree are:

carat depth 4113.474472
width depth 2509.078551
carat 2471.092102
length width 2101.523405
carat length 1701.180714
length depth 1508.769071
carat width depth 1250.523931
carat width color_encoded 1221.805363
carat^2 1195.777872
length^2 1165.312703

We can see that the top features are similar to what we got in the heatmap. These features show the highest correlation with the target variable i.e price. The salient features are hence the features that matter in predicting and highly impact the target variable.

- **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?**

The optimal degree of polynomial was found to be 3 with the below values:
Optimal degree: 3
Average RMSE for optimal degree: 765.0979014891634
We tried multiple polynomial degrees and found the above one to be the best. We also tried multiple alpha values to see which returns the lowest RMSE.
Polynomial regularizations having the highest order of terms/degree greater than 3 inflict serious/undesired penalties on the model during the training and backpropagation, which heavily decreases the model's capacity to strike back and learn useful content. Multiplicative combination of more than 3 input arbitrary features from the dataset is not a fair way of regularizing the learning behavior of the model during overshoot or undershoot. A very high-order polynomial implies that the model is highly flexible and can capture intricate patterns and fluctuations in the training data. Specifically, as the degree of the polynomial increases, the model becomes increasingly complex, allowing it to closely follow the training data points and potentially achieve a very low training error. However, such high flexibility comes with the risk of overfitting. Overfitting occurs when the model captures noise or random fluctuations in the training data, leading to poor generalization performance on unseen data. In the case of high-order polynomial regression, the model may fit the training data too closely, capturing not only the underlying patterns but also the noise present in the data. As a result, the model's performance on testing data, or unseen data, may degrade significantly.

QUESTION 6:

You will train a multi-layer perceptron (fully connected neural network). You can simply use the sklearn implementation:

- **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).**

I tried various combinations of hidden layers and alpha values to see which gives the least RMSE. The different experiments conducted can be observed in the code.
From all the experiments conducted, the best values were obtained for scaled reduced data with the parameters and values as below:
Best params {'activation': 'relu', 'alpha': 0.1, 'hidden_layer_sizes': (50, 50)}
Test RMSE: 666.029563090723
I also tried running on the unscaled data, scaled data, scaled reduced data and scaled reduced skewed data. It was observed that it performed better on the scaled reduced data.

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

From all the experiments conducted, the best values were obtained for scaled reduced data with the parameters and values as below:
Best params {'activation': 'relu', 'alpha': 0.1, 'hidden_layer_sizes': (50, 50)}
Test RMSE: 666.029563090723
MLPs can capture non linear relationships between input features and target variables. They can learn the patterns in the data which linear regression cannot capture very well. As a result, MLPs may outperform linear regression when the relationship between features and target variables is highly nonlinear. Linear regression relies on handcrafted features whereas MLPs can learn feature representations from raw data. MLPs require

careful regularization. Linear regression is less prone to overfitting due to its simplicity. Linear regression provides easily interpretable coefficients but MLPs are difficult to interpret the learned relationship between the features and the target variables. MLPs are computationally intensive as compared to linear regression.
The values obtained for regression on scaled reduced data is as below:
Average RMSE for Ordinary Least Squares (OLS) for Reduced Scaled Data: 1581.83183864438
Average RMSE for Lasso Regression for Reduced Scaled Data: 1581.6691554265262 (Best alpha for Lasso for Reduced Scaled Data: 0.1)
Average RMSE for Ridge Regression for Reduced Scaled Data: 1581.8318524986157 (Best alpha for Ridge for Reduced Scaled Data:: 0.0001)
Linear regression got RMSE in the range of 1500s. As expected NN has performed better than linear regression.

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

The activation function chosen is ReLU (Rectified Linear Unit). This was chosen because it is simple and efficient. The function is defined as f(x)=max(0,x), which involves only simple arithmetic operations. Therefore it is fast to compute during forward and backward propagation. ReLU provides sparse activation, meaning that only a subset of neurons in a layer are activated. Unlike sigmoid and tanh activation functions, ReLU does not saturate for positive input values. Therefore it avoids the vanishing gradient problem. Due to its non-saturating nature and simplicity, ReLU activation can lead to faster convergence during training. ReLU activation is a popular choice in neural networks due to its simplicity, efficiency, avoidance of vanishing gradients, and ability to promote faster convergence during training.

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

Increasing the depth beyond a certain point can lead to more risks. Overfitting is the most common of them. As the network becomes deeper, it gains more capacity to memorize the data in turn fitting noise as well. Deeper networks are more susceptible to vanishing or exploding gradients during training. Deeper networks require more computational resources to train and evaluate. As the depth of the network increases, the number of parameters and computations required also increases exponentially. The deeper the network, the more difficult it is to train as well. Deeper networks may encounter difficulties with gradient descent convergence. As gradients propagate through many layers, they may become too small or too large, leading to slow convergence or divergence during training. Increasing the depth of the network may not always lead to better performance, especially if the dataset is small or if the problem is relatively simple.

<u>QUESTION 7:</u>

We will train a random forest regression model on datasets, and answer the following: • Random forests have the following hyper-parameters:
- Maximum number of features;
- Number of trees;
- Depth of each tree;

The below values gave the best performance:
Best hyperparameters: {'max_depth': 10, 'max_features': 8, 'n_estimators': 100}
Test RMSE: 650.9824075604025

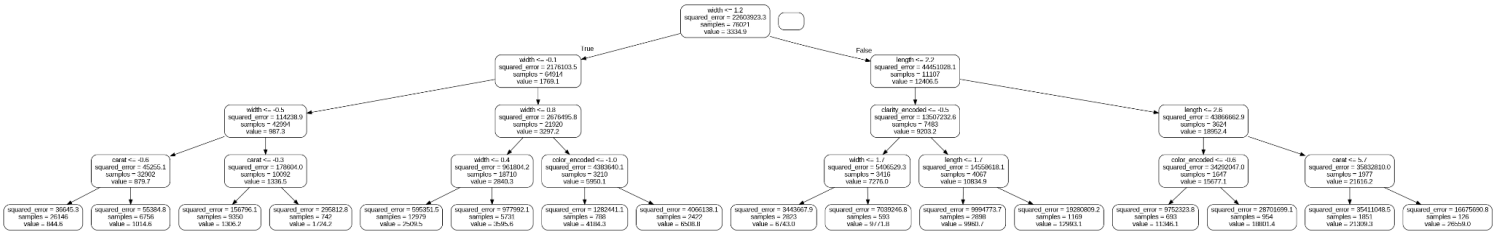
- **Question 7.1 - Explain how these hyper-parameters affect the overall performance. Describe if and how each hyper-parameter results in a regularization effect during training.**

- Maximum number of features - This hyperparameter is used to decide the maximum number of features that will be considered while splitting at each node in each decision tree. The higher this number, the greater the number of splits and hence more diverse trees. This will help it capture complex patterns and relations in the data. But, for high dimensional data, setting this value too high can lead to overfitting. Reducing the maximum number of features can help prevent overfitting by limiting the diversity of trees and promoting simpler models.
- Number of trees - This hyperparameter is used to define the number of decision trees in the random forest. As we increase the number of trees, the robustness and scalability increases as the variance in the predictions reduces. But, adding more trees also adds computational complexity and training time. Using too few trees can lead to underfitting. This is why finding the optimum number of trees is really important.
- Depth of each tree - This hyperparameter is used to decide the maximum depth of each decision tree in the random forest. Increasing the depth leads to increasing the complexity that can be captured. But, deeper trees can lead to overfitting especially when the data is noisy or has a lot of outliers. Limiting the depth of each tree can regularize the model by promoting simpler tree structures and reducing the risk of overfitting.
- Each of these hyperparameters can have a regularization effect during training by controlling the complexity of the random forest model and preventing overfitting. Regularization wants to reduce the complexity of the models and promote simple models that can generalize over unseen data. By finding the optimal values for these hyperparameters, we can create a balance between model complexity and generalization performance. We hence get a more robust and effective random forest model.

- **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 forests do so by combining multiple decision trees. Each of them is trained on a different subset of the data and features. Random Forests use an ensemble learning technique, where multiple decision trees are trained independently and then combined to make predictions. Each decision tree in the ensemble learns to capture different aspects of the data, resulting in a diverse set of base models. Random Forest also uses feature randomization. At each node of each decision tree, only a random subset of features is considered for splitting. With this, we can have diversity. Each tree in the Random Forest learns a different decision boundary. Some trees may focus on certain features or patterns in the data, while others focus on different features or patterns. Hence , it can capture a wide range of features and patterns in the data. To make predictions, Random Forests aggregate the predictions of individual decision trees. Random Forests are robust to noise and outliers in the data. They are also less prone to overfitting compared to individual decision trees. The ensemble nature of Random Forests allows them to generalize well to unseen data and capture complex non-linear relationships between input features and the target variable. A random forest has a large number of uncorrelated models with the final target variable being a majority decision fusion from all the trees.

- **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 that is selected at the root node for branching is width. This feature is selected for branching at the root node. This means that it is the most important feature used to start the splitting process.In a decision tree, the features closer to the root node are more salient and significant than the features near the leaf nodes. Based on p values, we observed that the most significant features were carat, color and clarity. We do observe carat, color and clarity as important features here as well. Thus, it is safe to say that the most important features found from p-values of linear regression have significant overlap with the most important features found in a random decision tree within the random forest for the Diamond dataset.

- **Question 7.4 - Measure "Out-of-Bag Error" (OOB). Explain what OOB error and R2 score means.**

Best Out-of-bag error: 0.019290294968740906.
Out of the Bag Error is used for measuring the performance of the model. In random forests, each decision tree is trained on a sample of the dataset. This sample is created by random sampling the original dataset with replacement. Therefore, some values may be included in this sample and some may not. The out-of-bag (OOB) error is calculated as the average error of the model on those observations that were not included in the bootstrap sample used to train a particular decision tree. Observations that were not included in the sample are considered "out-of-bag" for that particular tree. OOB error is the average error of all out of the bag observations. It is an unbiased estimate of the model's performance on data it has not seen earlier. It is useful in random forests because it allows us to evaluate the model's performance while utilizing all the data available for training. The R-squared (R2) score, also known as the coefficient of determination, is a metric used to evaluate the performance of a regression model. It measures the proportion of the variance in the target variable that is explained by the features of the model. It ranges from 0 to 1. 0 means that the model does not explain the variance in the target model and 1 means that the model perfectly explains the variance. The total sum of squares represents the total variance in the target variable, while the sum of squared residuals represents the unexplained variance by the model. The R2 score is then calculated as 1 minus the ratio of the sum of squared residuals to the total sum of squares. R2 score tells us how well the regression model fits the data. If the score is higher, it means that it fits well. However, R2 score alone does not provide information about how good the fit is relative to the complexity of the model.

<u>QUESTION 8:</u>
In this part, pick either one of the datasets and apply LightGBM OR CatBoost. If you do both, we will only look at the first one.

- **Question 8.1 - Read the documentation of LightGBM OR CatBoost and determine the important hyperparameters along with a search space for the tuning of these parameters (keep the search space small).**

I have implemented LightGBM. Important hyper parameters are:
- num_leaves - It controls the maximum number of leaves in each tree. The higher the value the more complex the model and can lead to overfitting.
- learning_rate - This controls the step size during gradient boosting. Lower values result in slower learning but can lead to better generalization.
- max_depth - Maximum depth of each tree. Higher values make the model more complex and increase the likelihood of overfitting.
- min_data_in_leaf: Minimum number of data points in each leaf. Larger values can prevent overfitting but may lead to underfitting.
- feature_fraction: Fraction of features to consider for each tree. Lower values can help reduce overfitting by introducing randomness.

- boosting_type - It determines the type of boosting technique used during training. Boosting is an ensemble learning technique that combines multiple weak learners (usually decision trees) sequentially to create a strong learner. Different boosting techniques introduce variations in how the weak learners are combined. Different values are gbdt (Gradient Boosting Decision Tree), rf (Random Forest), dart (Dropouts meet Multiple Additive Regression Trees), goss (Gradient-based One-Side Sampling).

- **Question 8.2 - Apply Bayesian optimization using skopt.BayesSearchCV from scikit-optmize 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.**

In Bayesian optimization, a Gaussian process is utilized to approximate the objective function, which allows priors on the distribution of moments such as mean and uncertainty to propagate in the forward direction as the search process progresses.
I tried on various values with different hyperparameters as mentioned below:

```
param_space = {
    'num_leaves': (20, 100),
    'learning_rate': (0.01, 0.1),
    'max_depth': (5, 15),
    'min_child_samples': (10, 30),
    'subsample': (0.7, 0.9)
}
```

The ideal hyperparameters found were as below:
Best Hyperparameters: OrderedDict([('learning_rate', 0.08332138524886121), ('max_depth', 10), ('min_child_samples', 24), ('num_leaves', 95), ('subsample', 0.7143758614840212)])
Best RMSE: 627.647251922508
Test RMSE: 620.691045915077

- **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?**

Different hyperparameters impact the model's performance, regularization, and fitting efficiency in the following ways:

- Performance Improvement - Hyperparameters that directly influence the model's ability to capture patterns in the data or reduce error typically lead to performance improvement. If we increase the number of trees, the model's predictive power by incorporating more diverse and robust decision boundaries.
- Regularization - It controls the complexity of the model and helps prevent overfitting, thereby reducing the generalization gap. For instance, parameters like the regularization strength (lambda) in ridge regression or the maximum depth of trees in decision trees can be tuned to balance between fitting the training data well and avoiding overfitting on unseen data.
- Fitting efficiency - Certain hyperparameters may affect the model's fitting efficiency, influencing how quickly the model converges to an optimal solution during training. For example, the learning rate in gradient boosting algorithms or the kernel bandwidth in kernel methods can impact the convergence speed and overall efficiency of the training process.

In Bayesian optimization, the goal is to find the optimal combination of hyperparameters that maximizes the model's performance while maintaining good generalization and efficient training. By analyzing the impact of different hyperparameters on these aspects, one can determine which ones contribute most significantly to each criterion and adjust them accordingly to achieve better overall results. We have tried various

hyperparameters as shown above. Increasing the number of trees generally leads to better model performance by capturing more complex patterns in the data. However, adding too many trees can risk overfitting. A lower learning rate typically results in slower but more robust learning, allowing the model to generalize better. We see above that we have a lower learning rate. Deeper trees can capture more complex relationships in the data, potentially improving performance. However, excessively deep trees can also lead to overfitting, especially with smaller datasets. For our dataset, a depth of 10 worked the best. Randomly subsampling the training data for each tree can introduce regularization, preventing the model from overfitting to the training set. Our best subsample value was 0.7. Increasing the number of leaves per tree can improve the model's capacity to capture complex interactions but may also increase computational complexity and training time. By tuning these hyperparameters using Bayesian optimization or other techniques, one can strike a balance between model performance, regularization, and fitting efficiency to achieve the best overall results for the specific task and dataset at hand.

QUESTION 9:

- **Question 9.1 - Report the following statistics for each hashtag, i.e. each file has:**

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

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

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

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

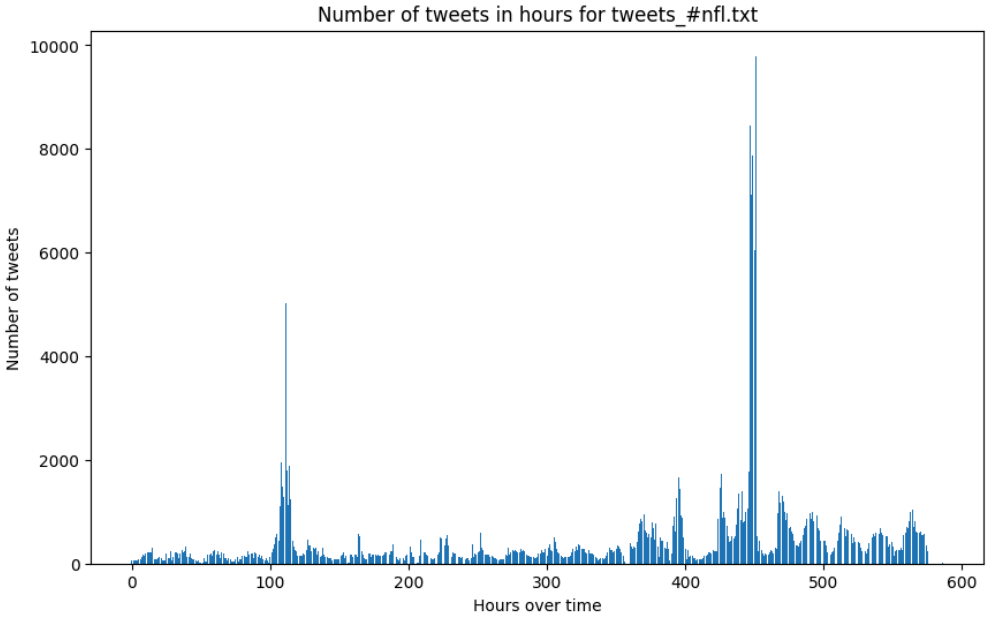
For ECE219_tweet_data/tweets_#patriots.txt\
Average tweets per hour are: 750.8942646068899\
Average number of followers of users posting the tweets per tweet are: 3280.4635616550277\
Average number of retweets per tweet: 1.7852871288476946

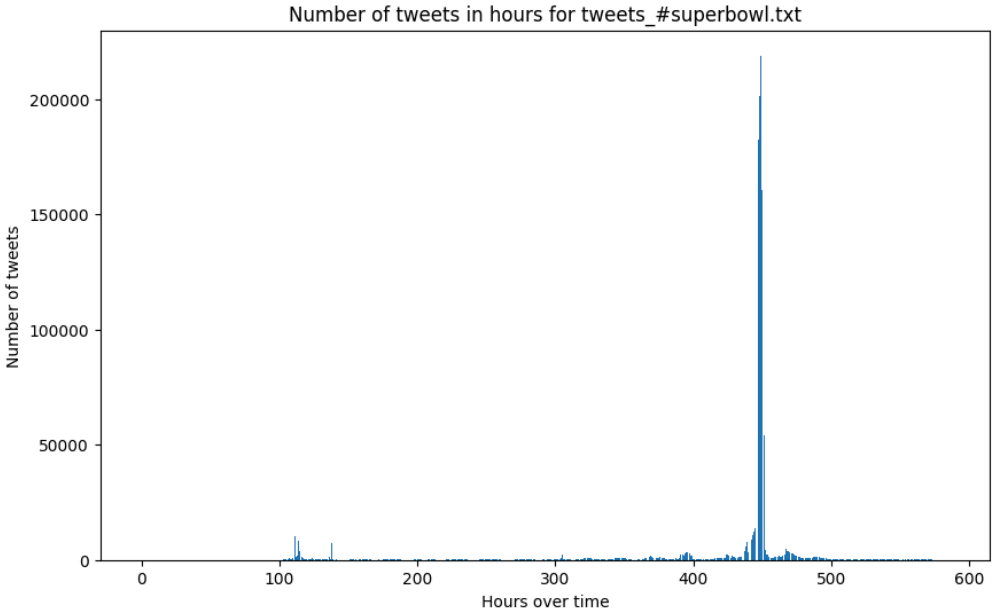
For ECE219_tweet_data/tweets_#sb49.txt\
Average tweets per hour are: 1276.8570598680474\
Average number of followers of users posting the tweets per tweet are: 10374.160292019487\
Average number of retweets per tweet: 2.52713444111402

For ECE219_tweet_data/tweets_#superbowl.txt\

Average tweets per hour are: 2072.11840170408\
Average number of followers of users posting the tweets per tweet are: 8814.96799424623\
Average number of retweets per tweet: 2.3911895819207736

- **Question 9.2 - Plot “number of tweets in hour” over time for #SuperBowl and #NFL (a bar plot with 1-hour bins). The tweets are stored in separate files for different hashtags and files are named as tweet [#hashtag].txt.**





Both the histograms have different life cycle patterns but we can see that they both peak at the 450th hour. We know that the data is collected 2 weeks before the game to a week after the game. Therefore, the peak is when the game is going on. The discussion is concentrated around the time when the superbowl and nfl were actually going on.

QUESTION 10:

- **Describe your task.**
- **Explore the data and any metadata (you can even incorporate additional datasets if you choose).**
- **Describe the feature engineering process. Implement it with reason: Why are you extracting features this way - why not in any other way?**
- **Generate baselines for your final ML model.**
- **A thorough evaluation is necessary.**
- **Be creative in your task design - use things you have learned in other classes too if you are excited about them!**

TASK - Library of Prediction Tasks given 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.**

We first carried out some feature analysis on the data. We extract only certain features from the entire dataset. The features are as listed below:
['Number of tweets', 'Total number of retweets', 'Sum of the number of followers', 'Maximum number of followers', 'Time of the day', 'Sum of ranking score', 'Sum of passivity', 'Total number of unique users','Total number of unique authors', 'Total number of user mentions']
For all the 6 files, we extracted these features and then fit them to an OLS Regression model. We found the mean square error and plotted the scatter plot for the top 5 features in each file. This gives us an idea of what the files contain and how the features are scattered in each file. The features in each file have important importance levels.
We can see the OLS Regression results below:
Hashtag: ECE219_tweet_data/tweets_#gohawks.txt
MSE: 296042.31731156004

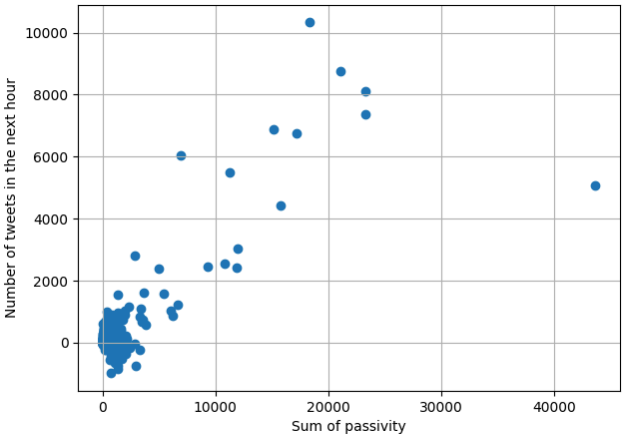
OLS Regression Results

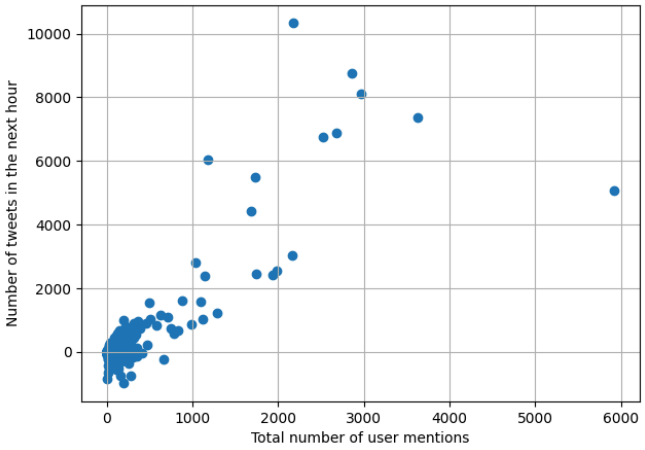
Dep. Variable:	y	R-squared (uncentered):	0.783
Model:	OLS	Adj. R-squared (uncentered):	0.779
Method:	Least Squares	F-statistic:	201.8
Date:	Sat, 16 Mar 2024	Prob (F-statistic):	1.43e-178
Time:	03:41:27	Log-Likelihood:	-4407.0
No. Observations:	571	AIC:	8834.
Df Residuals:	561	BIC:	8878.
Df Model:	10		
Covariance Type:	nonrobust		

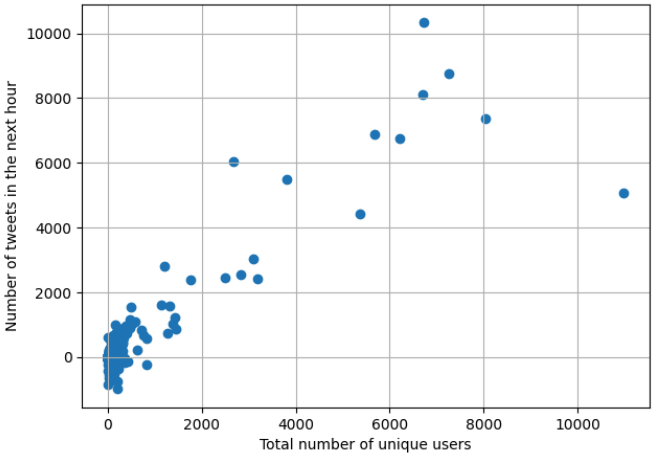
	coef	std err	t	P>|t|	[0.025	0.975]
x1	-0.0227	4.277	-0.005	0.996	-8.424	8.379
x2	-0.0854	0.030	-2.851	0.005	-0.144	-0.027
x3	-4.578e-05	7.74e-05	-0.591	0.555	-0.000	0.000
x4	0.0002	0.000	1.469	0.143	-6.04e-05	0.000
x5	-2.5729	2.078	-1.238	0.216	-6.655	1.509

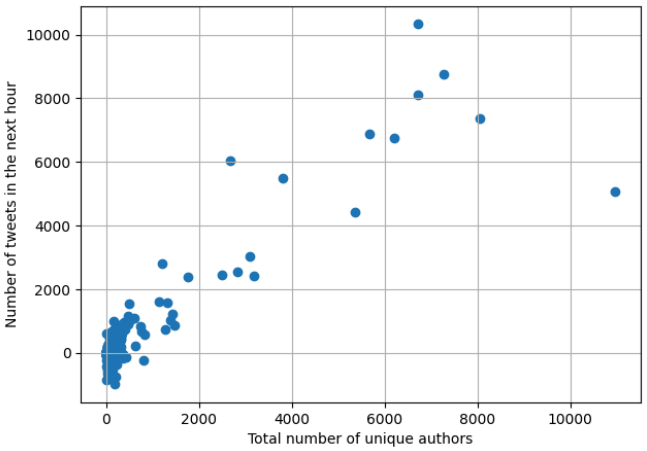
x6	0.7652	0.832	0.920	0.358	-0.869	2.399
x7	-0.6288	0.055	-11.412	0.000	-0.737	-0.521
x8	-99.6988	17.115	-5.825	0.000	-133.315	-66.082
x9	97.3100	17.047	5.708	0.000	63.826	130.794
x10	3.0031	0.357	8.421	0.000	2.303	3.704

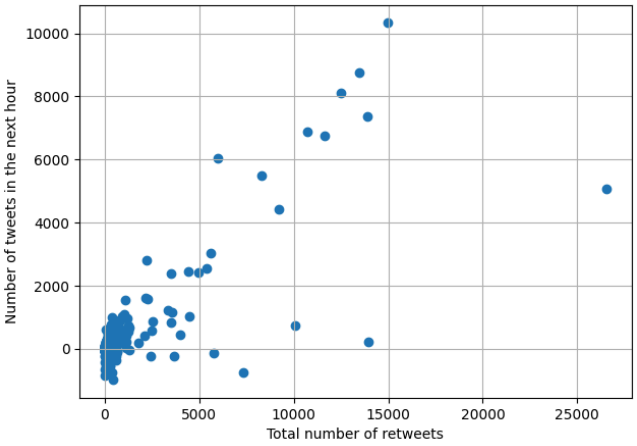
Omnibus:	631.309	Durbin-Watson:	2.047
Prob(Omnibus):	0.000	Jarque-Bera (JB):	98754.441
Skew:	4.748	Prob(JB):	0.00
Kurtosis:	66.723	Cond. No.	2.59e+06











Hashtag: ECE219_tweet_data/tweets_#gopatriots.txt

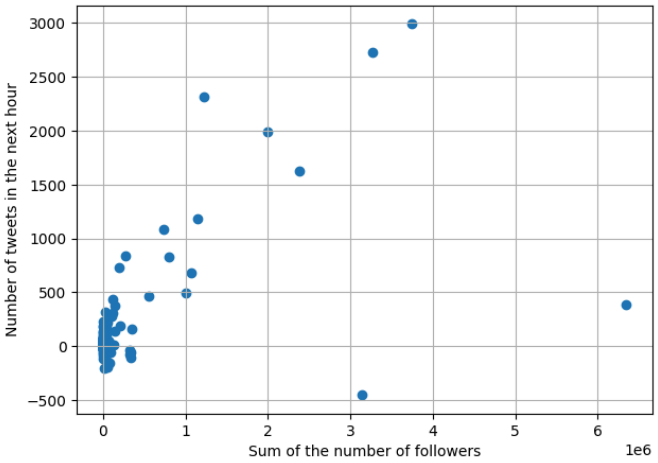
MSE: 21727.945699825843

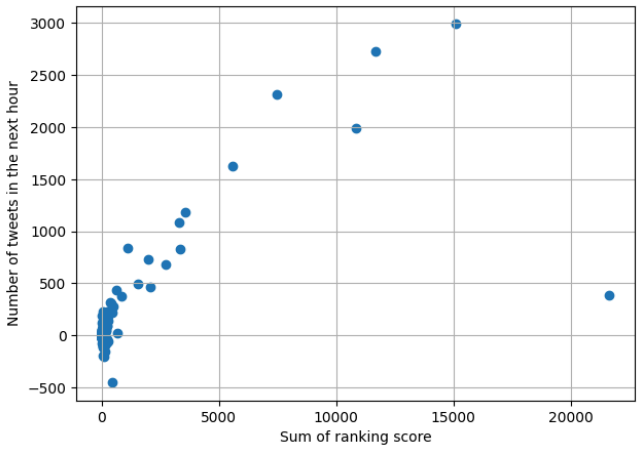
OLS Regression Results

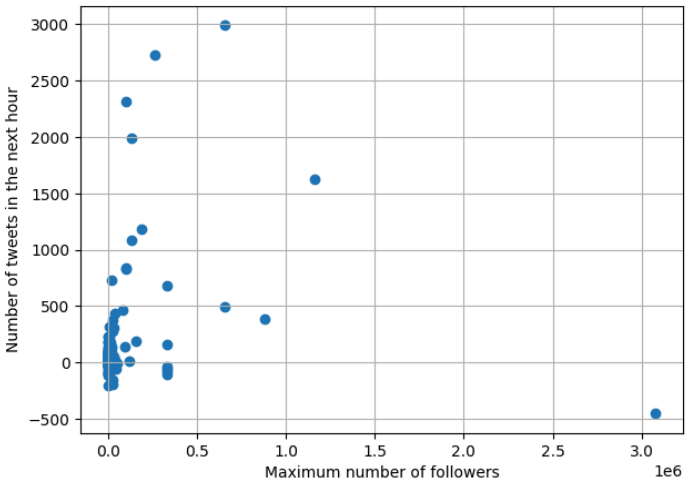
Dep. Variable:	y	R-squared (uncentered):	0.742
Model:	OLS	Adj. R-squared (uncentered):	0.738
Method:	Least Squares	F-statistic:	161.6
Date:	Sat, 16 Mar 2024	Prob (F-statistic):	5.56e-158
Time:	03:41:30	Log-Likelihood:	-3661.3
No. Observations:	571	AIC:	7343.
Df Residuals:	561	BIC:	7386.
Df Model:	10		
Covariance Type:	nonrobust		

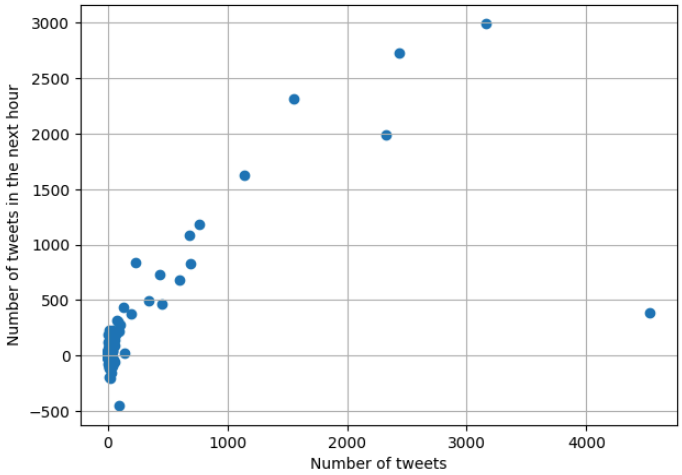
	coef	std err	t	P>|t|	[0.025	0.975]
x1	-40.5461	3.691	-10.986	0.000	-47.795	-33.297
x2	-0.5275	0.229	-2.304	0.022	-0.977	-0.078
x3	-0.0019	0.000	-13.422	0.000	-0.002	-0.002
x4	0.0017	0.000	11.098	0.000	0.001	0.002
x5	-0.4769	0.492	-0.970	0.333	-1.443	0.489
x6	7.5205	0.619	12.146	0.000	6.304	8.737
x7	0.3157	0.069	4.550	0.000	0.179	0.452
x8	61.6785	23.340	2.643	0.008	15.835	107.522
x9	-54.3140	23.768	-2.285	0.023	-101.000	-7.628
x10	4.4767	0.598	7.492	0.000	3.303	5.650

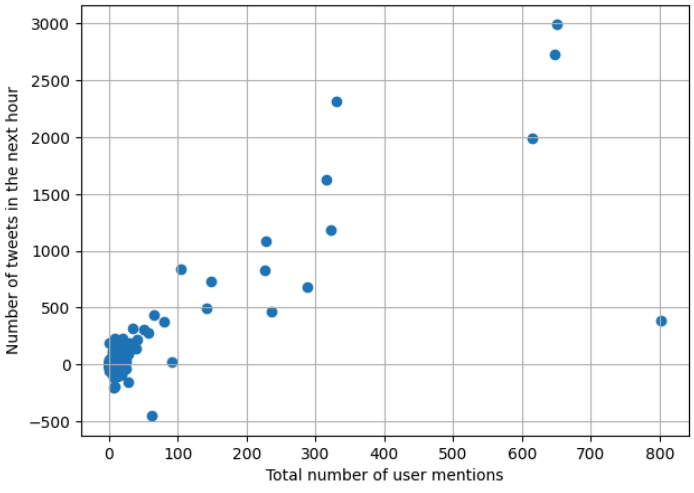
Omnibus:	678.303	Durbin-Watson:	2.128
Prob(Omnibus):	0.000	Jarque-Bera (JB):	317973.744
Skew:	4.976	Prob(JB):	0.00
Kurtosis:	118.178	Cond. No.	2.22e+06











Hashtag: ECE219_tweet_data/tweets_#nfl.txt

MSE: 184288.3990593135

OLS Regression Results

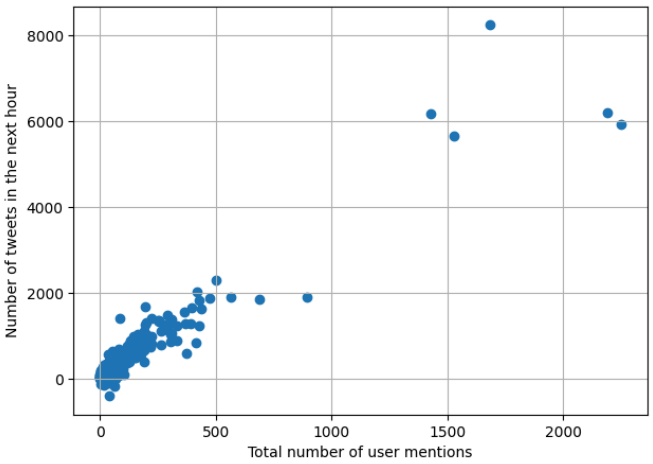
==========

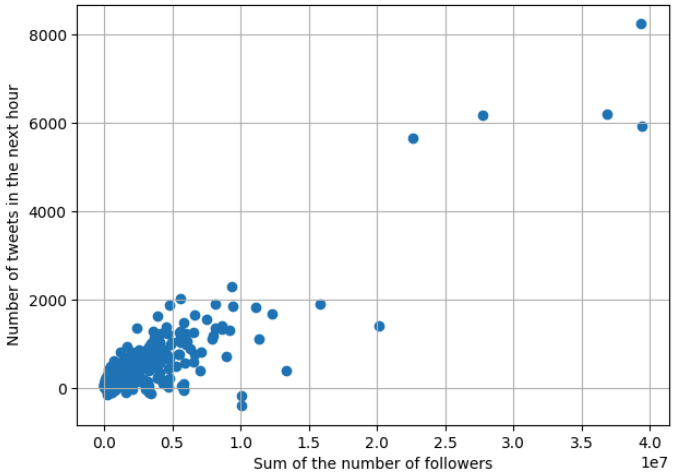
===

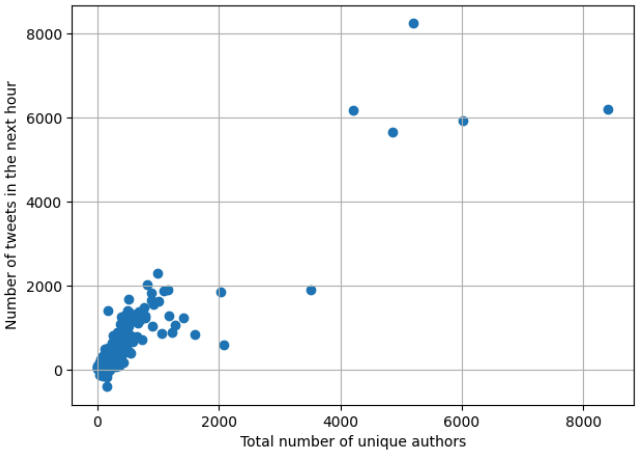
Dep. Variable:	y	R-squared (uncentered):	0.768
Model:	OLS	Adj. R-squared (uncentered):	0.764
Method:	Least Squares	F-statistic:	186.0
Date:	Sat, 16 Mar 2024	Prob (F-statistic):	7.02e-171
Time:	03:41:56	Log-Likelihood:	-4271.7
No. Observations:	571	AIC:	8563.
Df Residuals:	561	BIC:	8607.
Df Model:	10		
Covariance Type:	nonrobust		

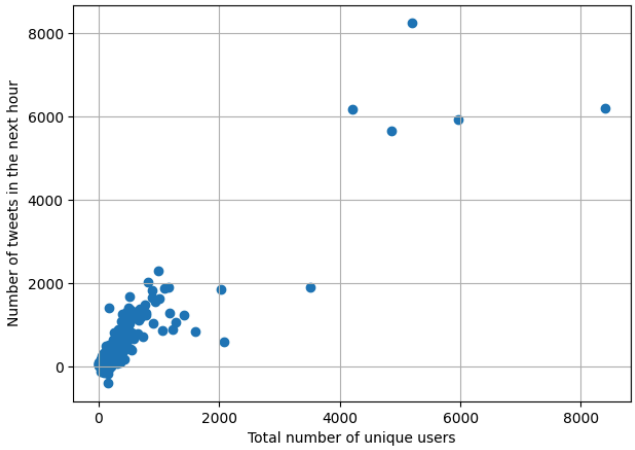
	coef	std err	t	P>|t|	[0.025	0.975]
x1	5.8995	1.413	4.176	0.000	3.124	8.675
x2	-0.0639	0.069	-0.923	0.356	-0.200	0.072
x3	0.0002	2.34e-05	6.511	0.000	0.000	0.000
x4	-0.0002	3.38e-05	-5.920	0.000	-0.000	-0.000
x5	3.5264	1.878	1.878	0.061	-0.162	7.215
x6	-1.2316	0.309	-3.988	0.000	-1.838	-0.625
x7	-0.0892	0.063	-1.421	0.156	-0.212	0.034
x8	79.5622	12.460	6.386	0.000	55.089	104.035
x9	-80.4155	12.584	-6.390	0.000	-105.133	-55.698
x10	3.9251	0.573	6.851	0.000	2.800	5.051

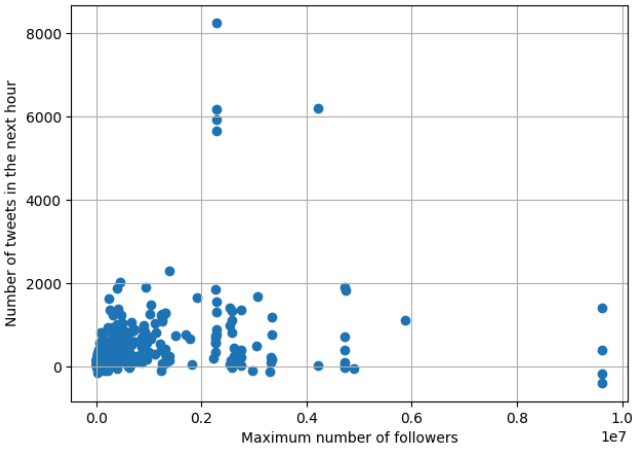
Omnibus:	542.024	Durbin-Watson:	1.969
Prob(Omnibus):	0.000	Jarque-Bera (JB):	86087.002
Skew:	3.588	Prob(JB):	0.00
Kurtosis:	62.723	Cond. No.	4.24e+06











Hashtag: ECE219_tweet_data/tweets_#patriots.txt

MSE: 3619767.990944837

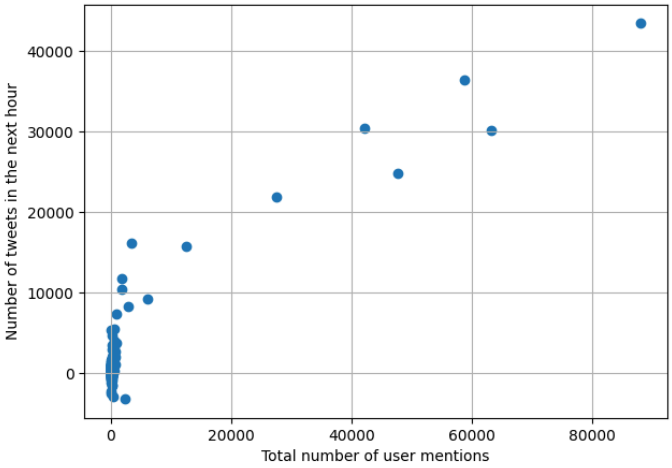
OLS Regression Results

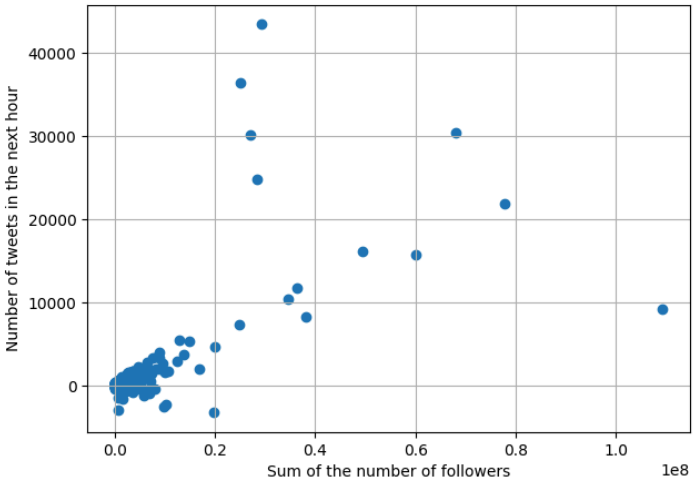
Dep. Variable:	y	R-squared (uncentered):	0.783
Model:	OLS	Adj. R-squared (uncentered):	0.779
Method:	Least Squares	F-statistic:	202.0
Date:	Sat, 16 Mar 2024	Prob (F-statistic):	1.18e-178
Time:	03:42:35	Log-Likelihood:	-5121.8
No. Observations:	571	AIC:	1.026e+04
Df Residuals:	561	BIC:	1.031e+04
Df Model:	10		
Covariance Type:	nonrobust		

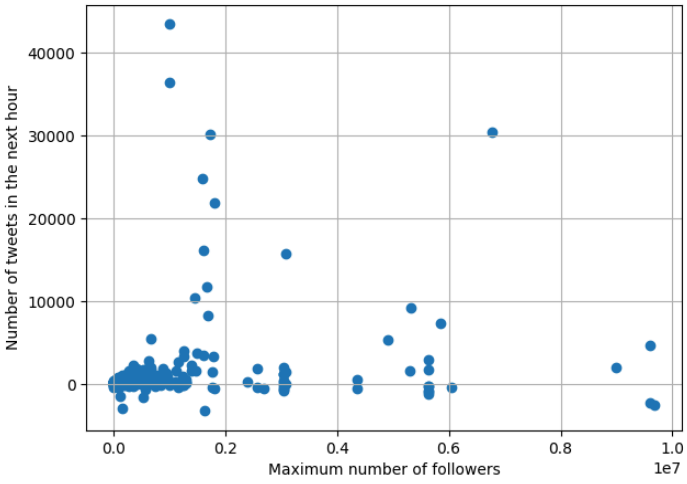
	coef	std err	t	P>|t|	[0.025	0.975]
x1	-26.0999	3.860	-6.762	0.000	-33.681	-18.519
x2	-0.0272	0.084	-0.325	0.745	-0.192	0.137

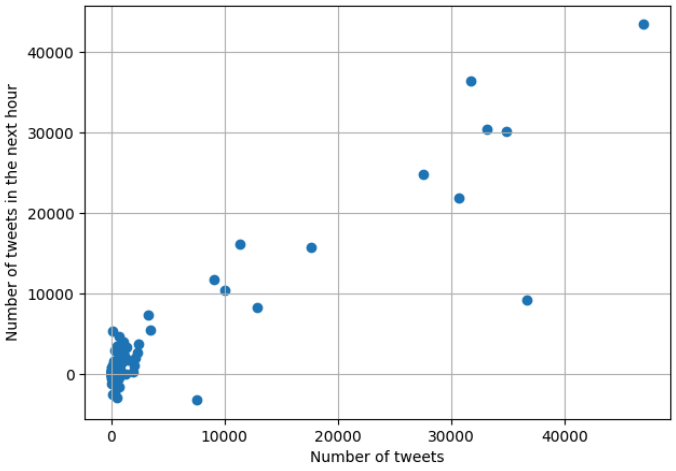
x3	0.0006	6.41e-05	9.538	0.000	0.000	0.001
x4	-0.0009	0.000	-7.476	0.000	-0.001	-0.001
x5	9.5584	6.696	1.427	0.154	-3.595	22.711
x6	4.7119	0.725	6.500	0.000	3.288	6.136
x7	-0.0893	0.045	-1.987	0.047	-0.178	-0.001
x8	41.7522	37.414	1.116	0.265	-31.736	115.240
x9	-38.1127	37.304	-1.022	0.307	-111.385	35.159
x10	2.1742	0.179	12.129	0.000	1.822	2.526

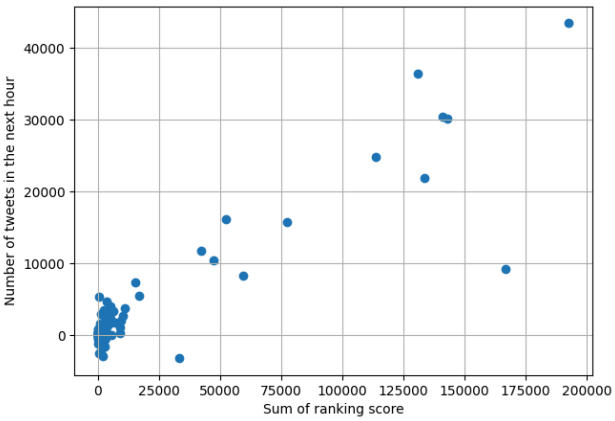
Omnibus:	938.060	Durbin-Watson:	1.898
Prob(Omnibus):	0.000	Jarque-Bera (JB):	500473.514
Skew:	9.614	Prob(JB):	0.00
Kurtosis:	146.757	Cond. No.	5.59e+06











Hashtag: ECE219_tweet_data/tweets_#sb49.txt

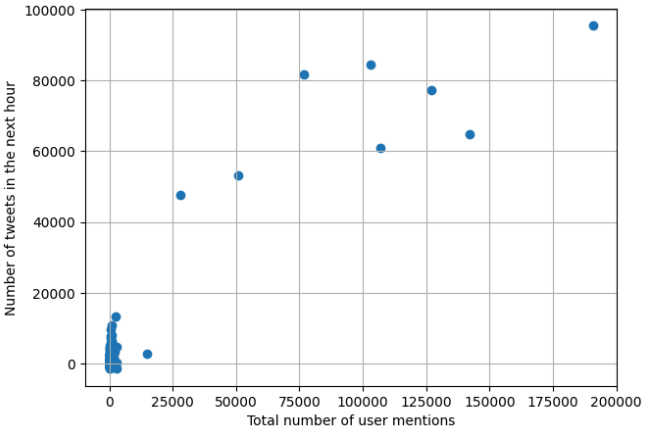
MSE: 8860126.964503309

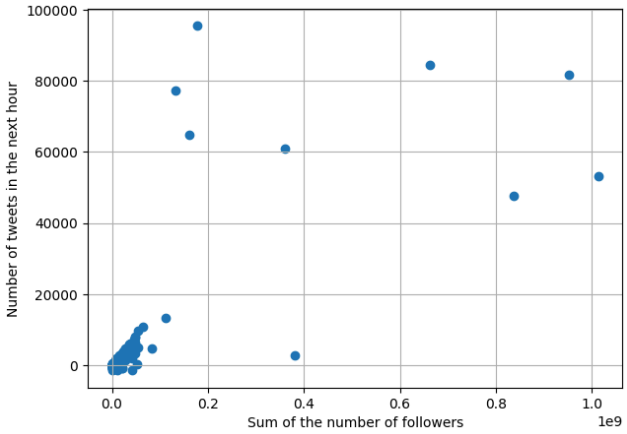
OLS Regression Results

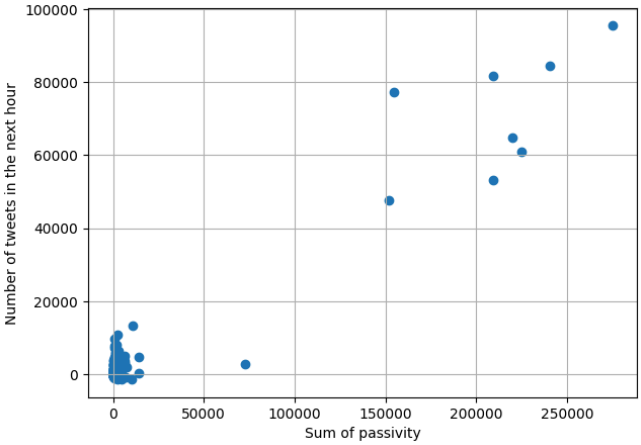
Dep. Variable:	y	R-squared (uncentered):	0.895
Model:	OLS	Adj. R-squared (uncentered):	0.894
Method:	Least Squares	F-statistic:	480.7
Date:	Sat, 16 Mar 2024	Prob (F-statistic):	1.30e-267
Time:	03:43:42	Log-Likelihood:	-5377.4
No. Observations:	571	AIC:	1.077e+04
Df Residuals:	561	BIC:	1.082e+04
Df Model:	10		
Covariance Type:	nonrobust		

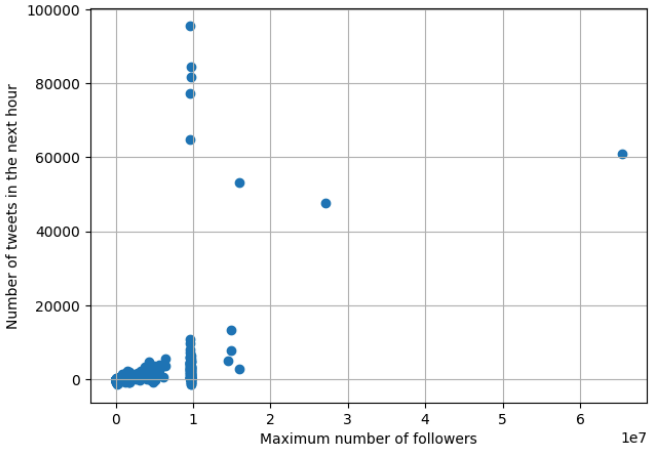
	coef	std err	t	P>|t|	[0.025	0.975]
x1	-16.0877	8.446	-1.905	0.057	-32.677	0.502
x2	-0.0598	0.072	-0.830	0.407	-0.201	0.082
x3	0.0002	3e-05	6.827	0.000	0.000	0.000
x4	-0.0002	5.2e-05	-3.073	0.002	-0.000	-5.76e-05
x5	-15.5423	10.336	-1.504	0.133	-35.844	4.759
x6	2.7952	1.793	1.559	0.120	-0.727	6.318
x7	-0.4180	0.065	-6.397	0.000	-0.546	-0.290
x8	-111.1858	70.212	-1.584	0.114	-249.097	26.726
x9	113.0474	69.914	1.617	0.106	-24.278	250.373
x10	2.4027	0.249	9.637	0.000	1.913	2.892

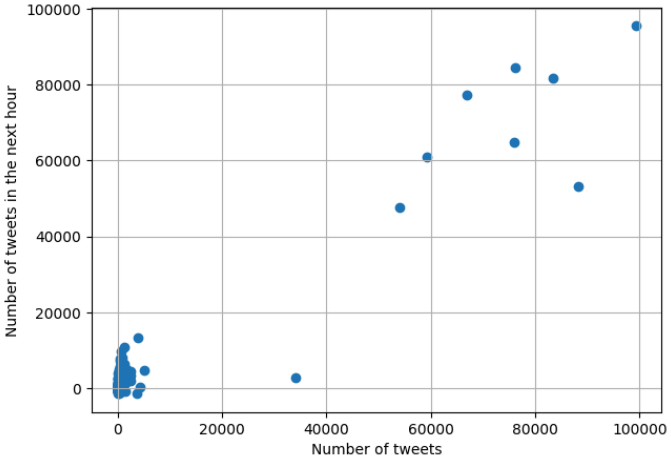
Omnibus:	979.614	Durbin-Watson:	1.676
Prob(Omnibus):	0.000	Jarque-Bera (JB):	886991.851
Skew:	10.269	Prob(JB):	0.00
Kurtosis:	194.989	Cond. No.	6.21e+07











Hashtag: ECE219_tweet_data/tweets_#superbowl.txt

MSE: 15458967.54483511

OLS Regression Results

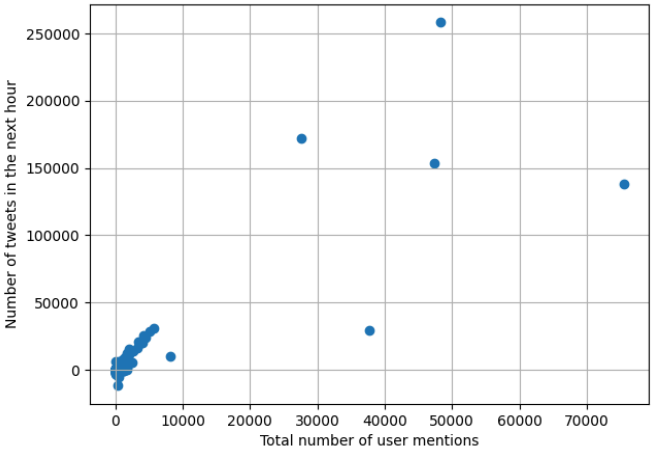
==========

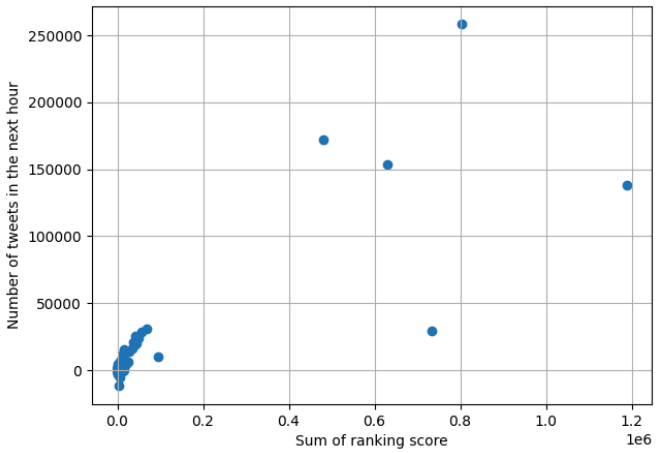
===

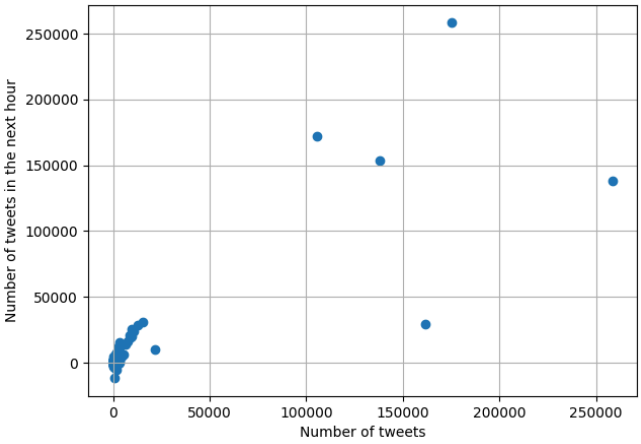
Dep. Variable:	y	R-squared (uncentered):	0.943
Model:	OLS	Adj. R-squared (uncentered):	0.942
Method:	Least Squares	F-statistic:	932.3
Date:	Sat, 16 Mar 2024	Prob (F-statistic):	0.00
Time:	03:45:31	Log-Likelihood:	-5536.3
No. Observations:	571	AIC:	1.109e+04
Df Residuals:	561	BIC:	1.114e+04
Df Model:	10		
Covariance Type:	nonrobust		

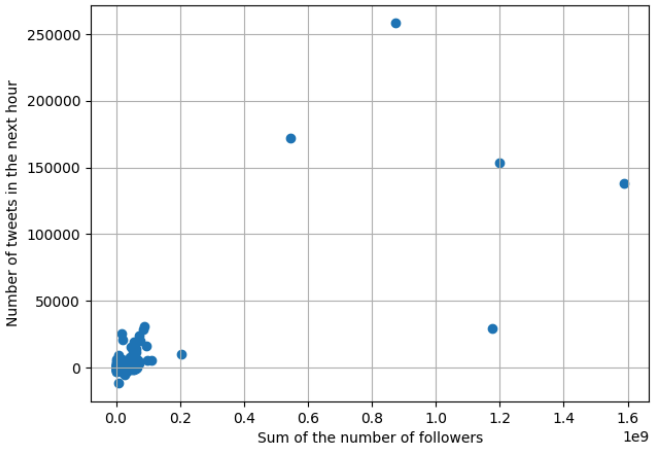
	coef	std err	t	P>|t|	[0.025	0.975]
x1	-41.2998	5.588	-7.391	0.000	-52.275	-30.325
x2	-0.7473	0.119	-6.294	0.000	-0.981	-0.514
x3	-0.0001	2.31e-05	-6.431	0.000	-0.000	-0.000
x4	9.802e-05	8.58e-05	1.143	0.254	-7.04e-05	0.000
x5	-33.3818	14.823	-2.252	0.025	-62.498	-4.266
x6	8.8292	1.163	7.589	0.000	6.544	11.114
x7	0.1727	0.047	3.640	0.000	0.079	0.266
x8	36.5074	25.912	1.409	0.159	-14.389	87.404
x9	-34.0838	25.813	-1.320	0.187	-84.785	16.618
x10	8.8973	0.408	21.820	0.000	8.096	9.698

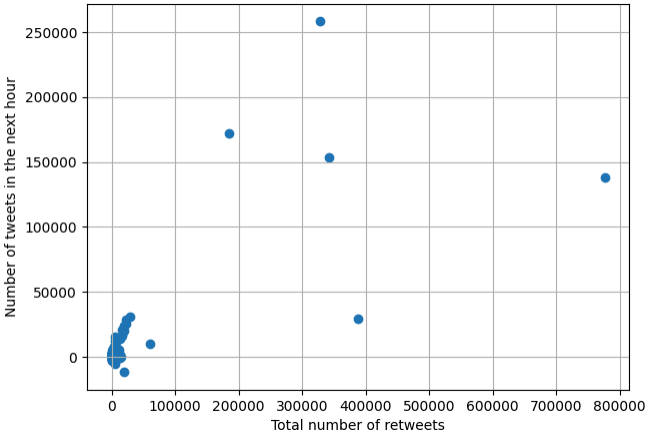
Omnibus:	1012.058	Durbin-Watson:	1.939
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1207472.208
Skew:	10.891	Prob(JB):	0.00
Kurtosis:	227.226	Cond. No.	2.39e+07











Predict the hashtags:

We now first extract the data again. This time we combine the data from all the 6 files and extract only certain columns which are - tweet id, citation date, author name and tweet content. We only keep these columns since we will be requiring them for our analysis. We store this in a dictionary with the hashtag as the key and the value as a dataframe of the above extracted features. For predicting the hashtags, we considered X as the tweet content and the label as the hashtag. We now split the data into train and test and do some feature engineering. We used a TF-IDF Vectorizer and then passed it to a Random Forest Classifier. Extracting features using TF-IDF vectorization is a common approach for natural language processing tasks like text classification or regression. TF-IDF stands for Term Frequency-Inverse Document Frequency, and it is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. TF-IDF helps in reducing the dimensionality of the text data by representing each document as a vector in a high-dimensional space, where each dimension corresponds to a unique word in the corpus. TF-IDF assigns higher weights to words that are more relevant to a specific document but are less common across all documents in the corpus. TF-IDF normalizes the word frequencies by considering both the term frequency (TF) and the inverse document frequency (IDF). TF-IDF representations are sparse, meaning most elements in the vector are zero. The classification report for each hashtag was observed. It was seen that an accuracy of almost 1 was obtained. This was because all the tweet content contained the hashtag in some form or the other and hence the model could easily predict which hashtag it belonged to for all the data.

Likeliness that the tweet belongs to a specific team fan:

The same data that was extracted earlier was used. The below teamwords were used for mapping and a new column named fan_affiliation was added.

```
team_keywords = {
    'Seahawks': ['#gohawks'],
    'Patriots': ['#patriots', '#gopats', '#gopatriots'],
    'NFL': ['#nfl'],
    'Sb49': ['#sb49'],
    'Superbowl': ['#superbowl']
}
```

We again split the data, applied TF-IDF Vectorizer and ran Random Forest. The below results were obtained for the different files.

Predicting fan affiliation for #gohawks:

Classification Report:

	precision	recall	f1-score	support
Seahawks	1.00	1.00	1.00	33825
accuracy			1.00	33825
macro avg	1.00	1.00	1.00	33825
weighted avg	1.00	1.00	1.00	33825

Predicting fan affiliation for #gopatriots:

Classification Report:

	precision	recall	f1-score	support
Patriots	1.00	1.00	1.00	4688
Seahawks	1.00	0.33	0.50	15
accuracy			1.00	4703
macro avg	1.00	0.67	0.75	4703
weighted avg	1.00	1.00	1.00	4703

Predicting fan affiliation for #nfl:

Classification Report:

	precision	recall	f1-score	support
NFL	0.97	0.99	0.98	38591
Patriots	0.93	0.81	0.86	7006
Seahawks	1.00	0.97	0.98	1008
accuracy			0.96	46605
macro avg	0.96	0.92	0.94	46605

```
weighted avg       0.96      0.96      0.96     46605

--------------------------------------------------
Predicting fan affiliation for #patriots:
Classification Report:
              precision    recall  f1-score   support

    Patriots       1.00      1.00      1.00     87949
    Seahawks       1.00      0.17      0.29       176

    accuracy                           1.00     88125
   macro avg       1.00      0.59      0.65     88125
weighted avg       1.00      1.00      1.00     88125

--------------------------------------------------
Predicting fan affiliation for #sb49:
Classification Report:
              precision    recall  f1-score   support

         NFL       0.98      0.22      0.36      1704
    Patriots       1.00      0.92      0.96     38741
        Sb49       0.95      1.00      0.98    105074
    Seahawks       1.00      0.80      0.89      3211

    accuracy                           0.97    148730
   macro avg       0.98      0.74      0.80    148730
weighted avg       0.97      0.97      0.96    148730
```

I also tried another method where I only considered the superbowl file to find fan affiliation. First, I found a count of the locations that are there. To get equally distributed data, we consider only two locations which are Washington and Massachusetts. We find 24367 entries for Washington and 19036 entries for Massachusetts. I first lemmatize the data, then apply a TF-IDF Vectorizer with min_df=3 and stopwords=english. An SVD with 50 components is then applied. To reduce the number of features we select the top 50 features using SVD decomposition, Truncated SVD() function. We now try multiple models as listed below:

For Logistic Regression:
Best parameters: {'C': 10, 'penalty': 'l1', 'solver': 'liblinear'}
Accuracy: 0.7256894049346879
Confusion Matrix:
 [[1020 1139]
 [184 2480]]
Accuracy: 0.7256894049346879
Recall: 0.9309309309309309
Precision: 0.6852721746338768
F1_score: 0.7894318000954959

For Random Forest:

Accuracy: 0.723729
Max_depth: 30
Confusion_matrix:
[[1336 823]
[484 2180]]
Accuracy: 0.7290068422143894
Recall: 0.8183183183183184
Precision: 0.7259407259407259
F1_score: 0.7693665078524792

For Gradient Boosting Classifier:
Confusion_matrix:
[[1332 827]
[542 2122]]
Accuracy: 0.7161517727555463
Recall: 0.7965465465465466
Precision: 0.7195659545608681
F1_score: 0.756101906288972

We can see that the accuracy of Random Forest with a max_depth of 30 is the highest.
Our inference from these results is that the model does pretty well for the task of classifying between the fan teams of Patriots and Hawks from the superbowl dataset. This inference is backed by the accuracy metrics and the confusion matrix where a clear binary classification between 2 classes is observed.

Predict the relative time at which a tweet was posted:
For this, we will be using the timestamp column as the target variable and the tweet content as the feature. We find the relative time. We do the same feature engineering. We split the data, perform TD-IDF Vectorization and then run a random forest regressor. For few of the files, we get the below results:
For superbowl:

accuracy			0.72	200
macro avg	0.71	0.71	0.71	200
weighted avg	0.71	0.72	0.71	200

The accuracy is not very good. Random Forest did not give very good results with this.

Predict the number of retweets/likes/quotes:
For this, we will be using the retweet count column as the target variable and the tweet content as the feature. We do the same feature engineering. We split the data, perform TD-IDF Vectorization and then run a random forest regressor. For few of the files, we get the below results:
Predicting retweets for gohawks:
Mean Squared Error: 19.89460458240946
Classification Report:

	precision	recall	f1-score	support
accuracy			0.97	33825
macro avg	0.05	0.04	0.04	33825
weighted avg	0.95	0.97	0.96	33825

Predicting retweets for gopatriots:
Mean Squared Error: 0.046991282160323194
Classification Report:

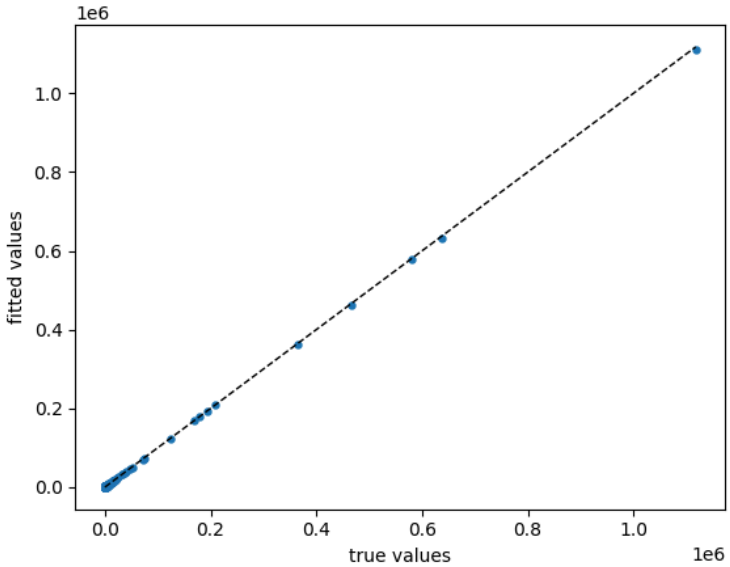
	precision	recall	f1-score	support
accuracy			0.98	4703
macro avg	0.14	0.14	0.14	4703
weighted avg	0.96	0.98	0.97	4703

Predicting retweets for nfl:
Mean Squared Error: 89.18528054929729
Classification Report:

	precision	recall	f1-score	support
accuracy			0.98	46605
macro avg	0.05	0.04	0.04	46605
weighted avg	0.97	0.98	0.97	46605

We can see that the accuracy is pretty good for all 3 files that I tried on. We could predict on the test set very well the number of retweets that we would see.

I also tried another way of testing this. The predictive value is changed from the number of tweets in the next hour to the number of retweets in the next hour. The application here is that the number of retweets may be used as a key indication of web advertising. Consider a corporation that can use an embedded advertising technique in a single tweet and then spread the advertisement to a larger audience through a high number of retweets. To forecast the amount of retweets in the following hour, we used five parameters including the number of tweets, total number of retweets, follower numbers, maximum follower number, and time of day. We tried gradient boosting method and performed grid search to find the best fitting model from cross validation and reported the MSE here.
max_depth= 10 max_features= auto min_samples_leaf= 1 min_samples_split= 2 n_estimators= 50
RMSE_train= 368.95557493797816 RMSE_test= 63840.45575879547



The charts above shows that there is an intermediate-strength linear relationship between

the true and forecasted values. As a result, extremely influential tweets have a huge and significant impact on the overall mood level. The quantity of impressions is a crucial aspect that influences the overall emotion level.