ECE219 Large-Scale Data Mining: Models and Algorithms **(Due: Sun, 03/17/24)**

Project 4: Regression Analysis

Instructor: Vwani Roychowdhury

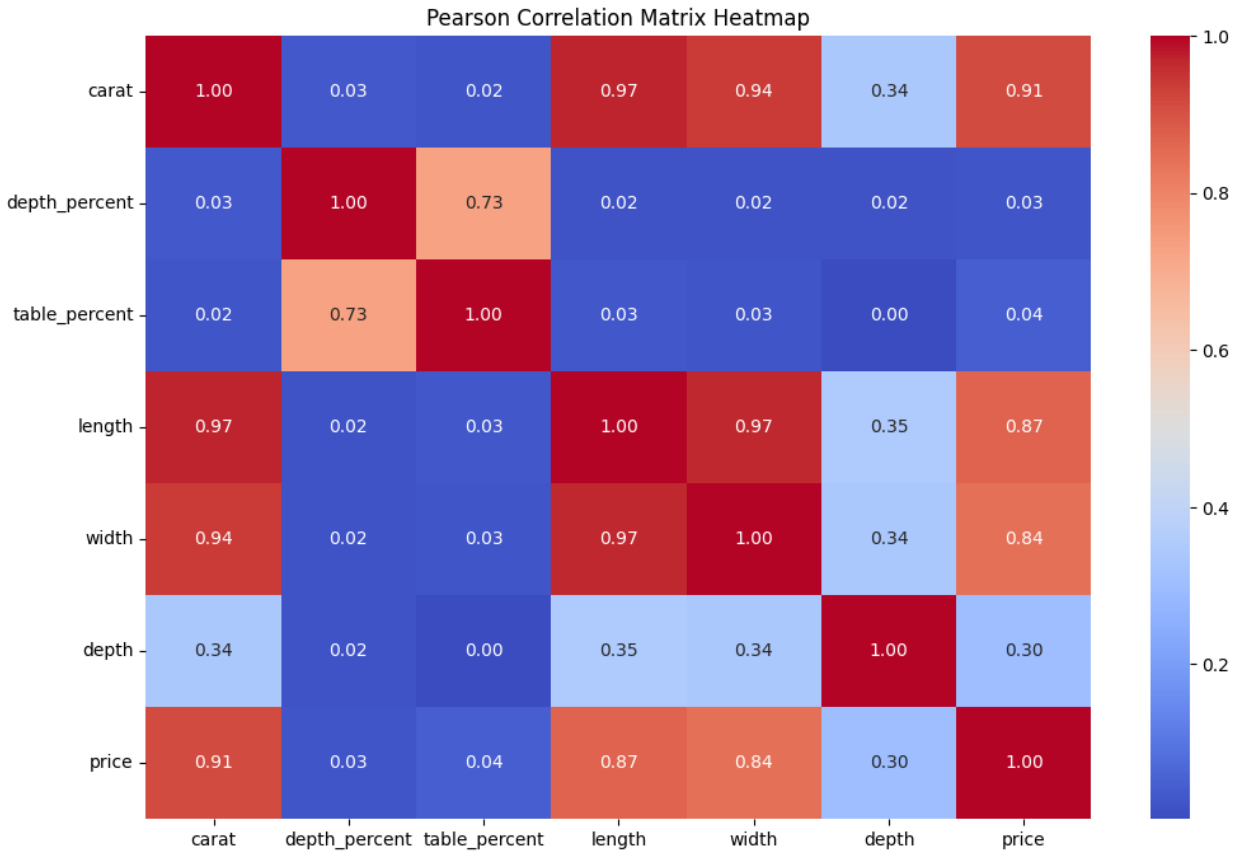
Aryaman Rajesh Gokarn 506303588

1. Question 1

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.

Answer:-

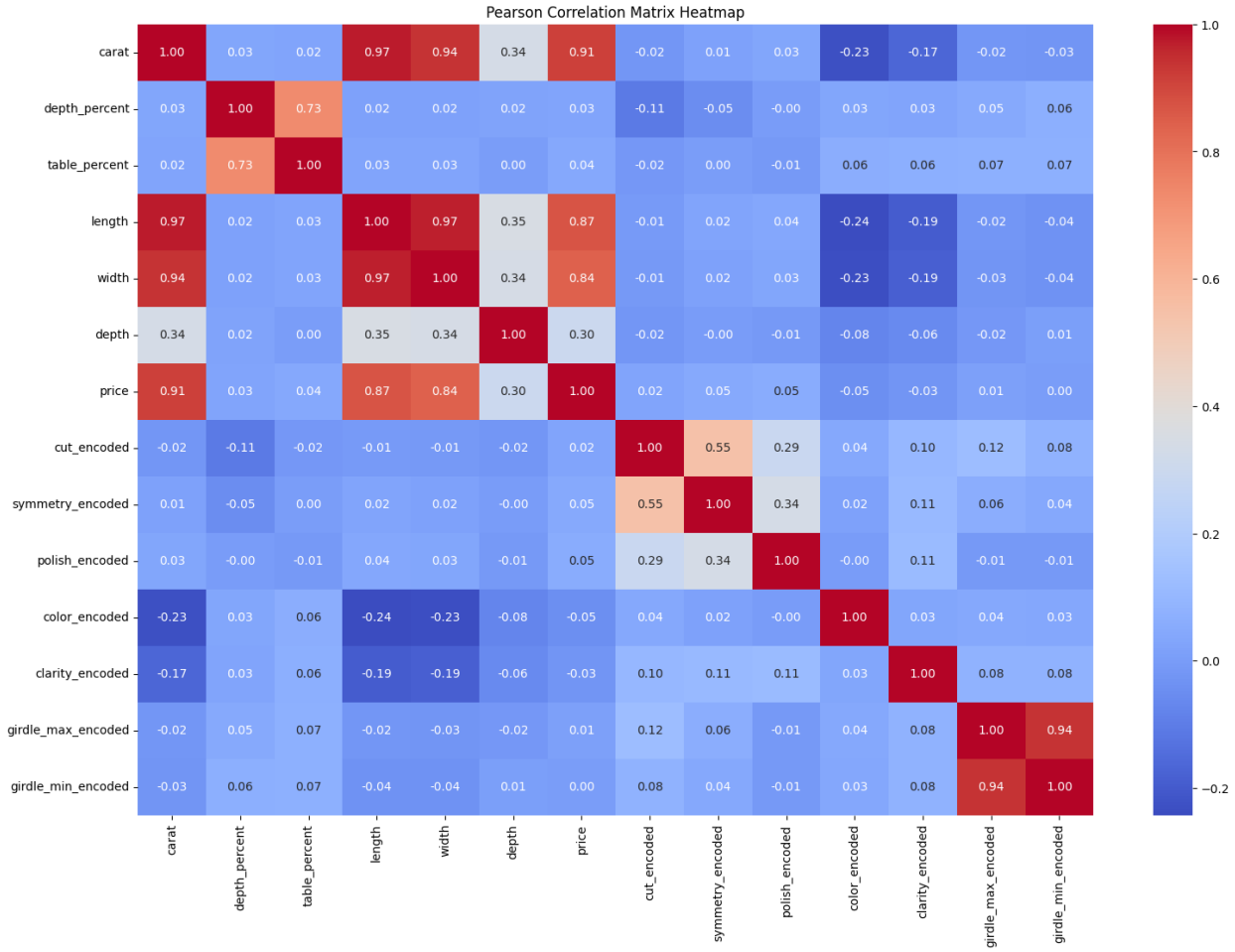
Before Transforming Categorical data into Numerical Data:-



Features with the highest absolute correlation with the target variable:-

carat	0.913479
length	0.869521
width	0.841887
depth	0.299696
table_percent	0.042453
depth_percent	0.025469

After Transforming Categorical data into Numerical Data:-



Features with the highest absolute correlation with the target variable:-

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

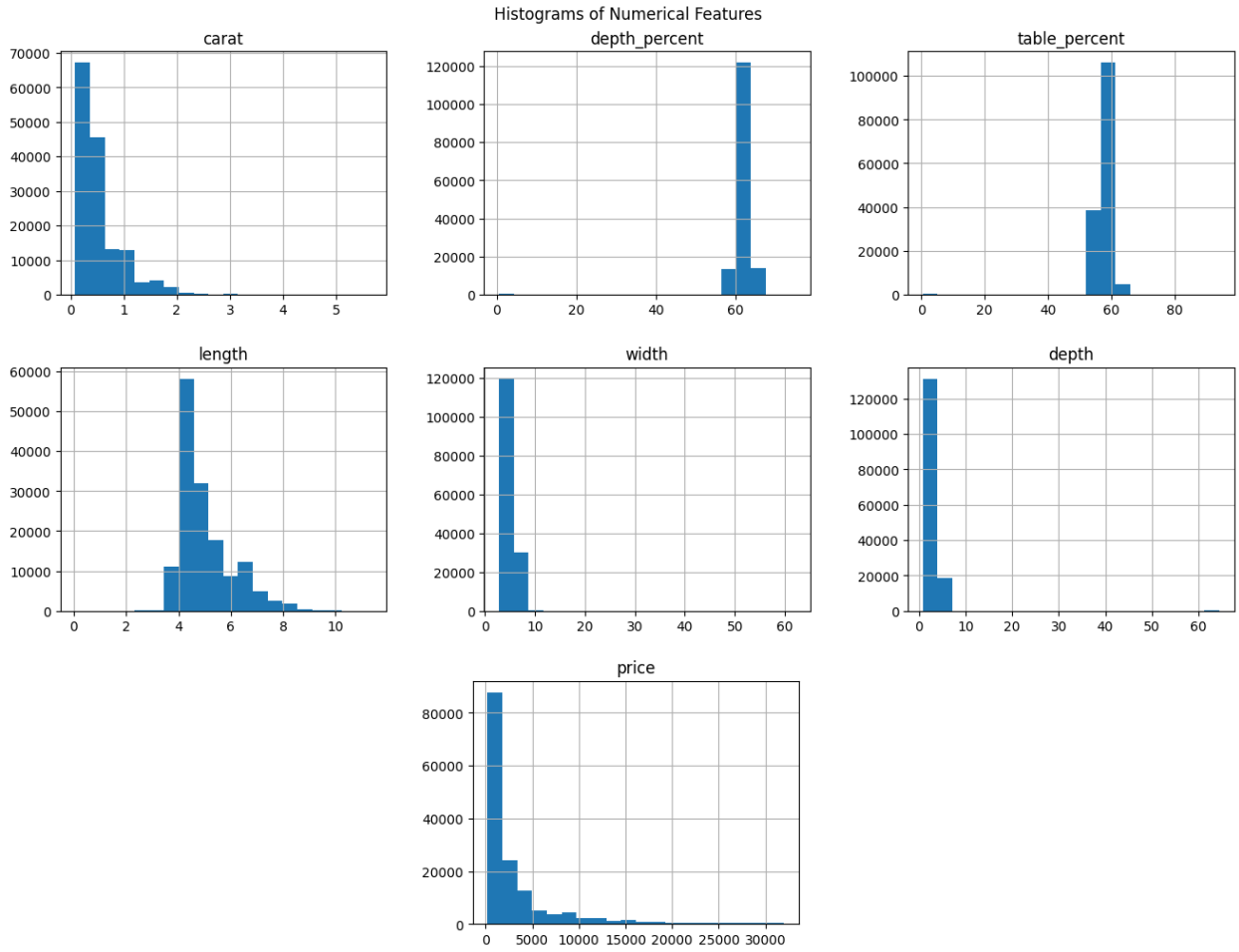
```
depth_percent          0.025469
cut_encoded            0.024356
girdle_max_encoded     0.013336
girdle_min_encoded     0.000814
```

The highest correlation with the target variable, price, is observed in carat. Length and width also show significant correlations, emphasizing their importance in determining diamond price. This reinforces the understanding that carat, along with length and width, holds key information for evaluating diamond prices, consistent with market trends where size and carat weight play critical roles in pricing diamonds.

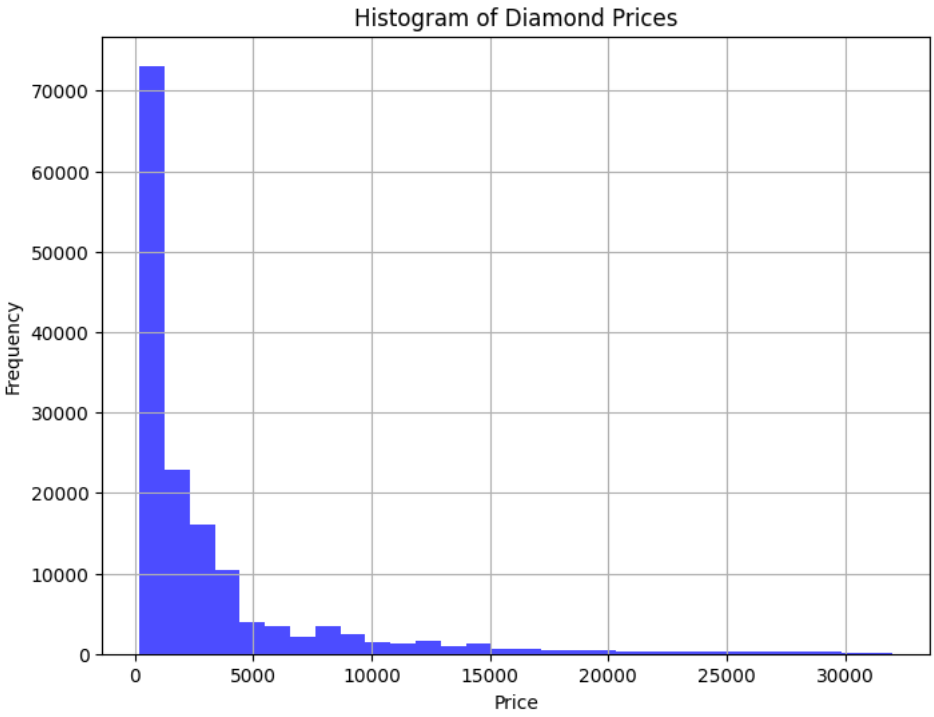
1.2. Plot the histogram of numerical features. What preprocessing can be done if the distribution of a feature has high skewness?

Answer:-

The histogram plot of the numerical features is as follows:-



For Skewness:-



We can see the price distribution is right skewed. Also there is a lot of high and medium prices which belongs to the rearest and nearly rear diamonds.

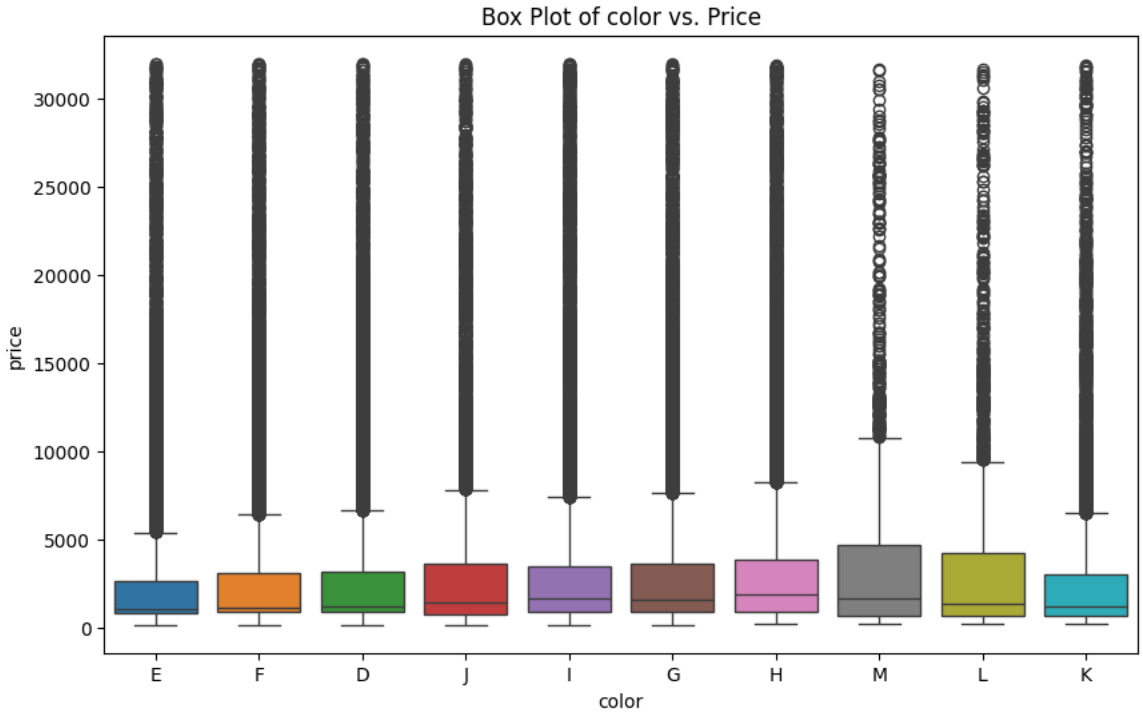
High skewness is observed in certain features, indicating the presence of costly outliers. When a feature's distribution is significantly skewed, meaning it lacks symmetry around the mean, several preprocessing techniques can be employed to address this issue. For instance, standardization can be implemented to mitigate the impact of extreme values on model robustness. Another common strategy involves logarithmic transformation, which helps reduce the magnitude of extreme values and brings the distribution closer to a normal shape. This transformation lessens the effect of heavily skewed values while preserving the relative differences among smaller values.

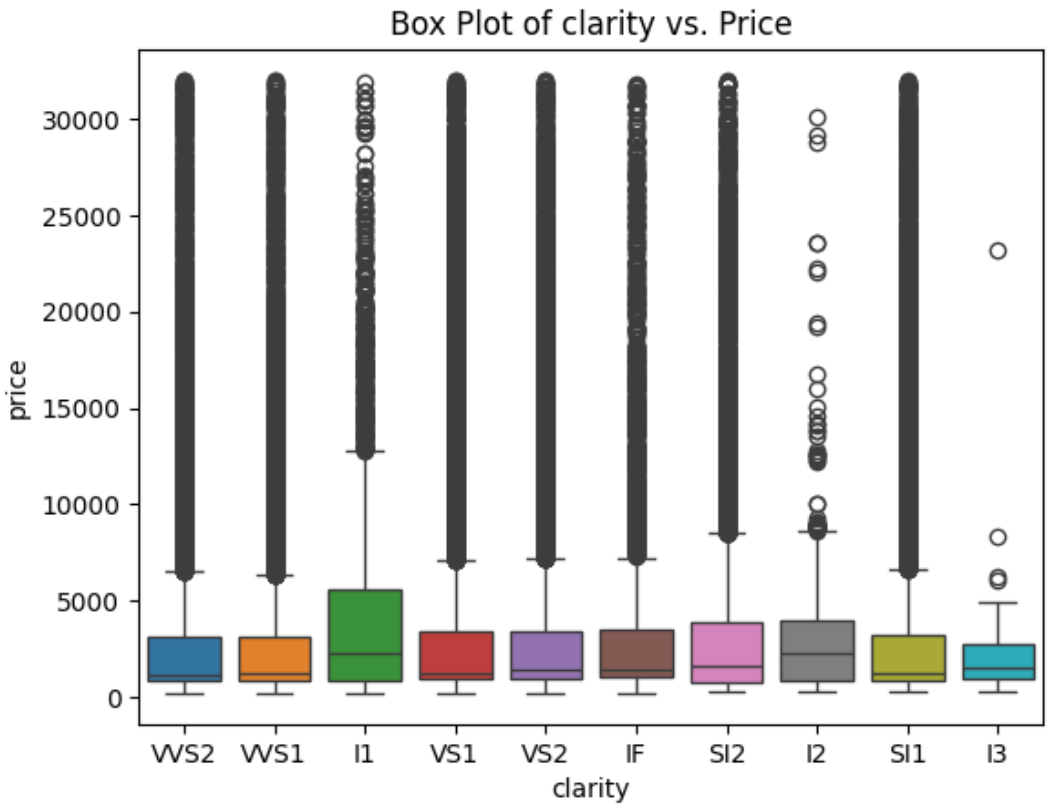
```
Skewnesss values before log transformation:-

      carat                 2.331750
      depth_percent       -13.559472
      table_percent       -11.046453
      length                1.283591
      width                 4.115307
      depth                27.493024
      price                 3.071707
      cut_encoded          -1.410662
      symmetry_encoded     -1.174542
      polish_encoded       -2.125702
      color_encoded        -0.466049
      clarity_encoded       0.053738
```

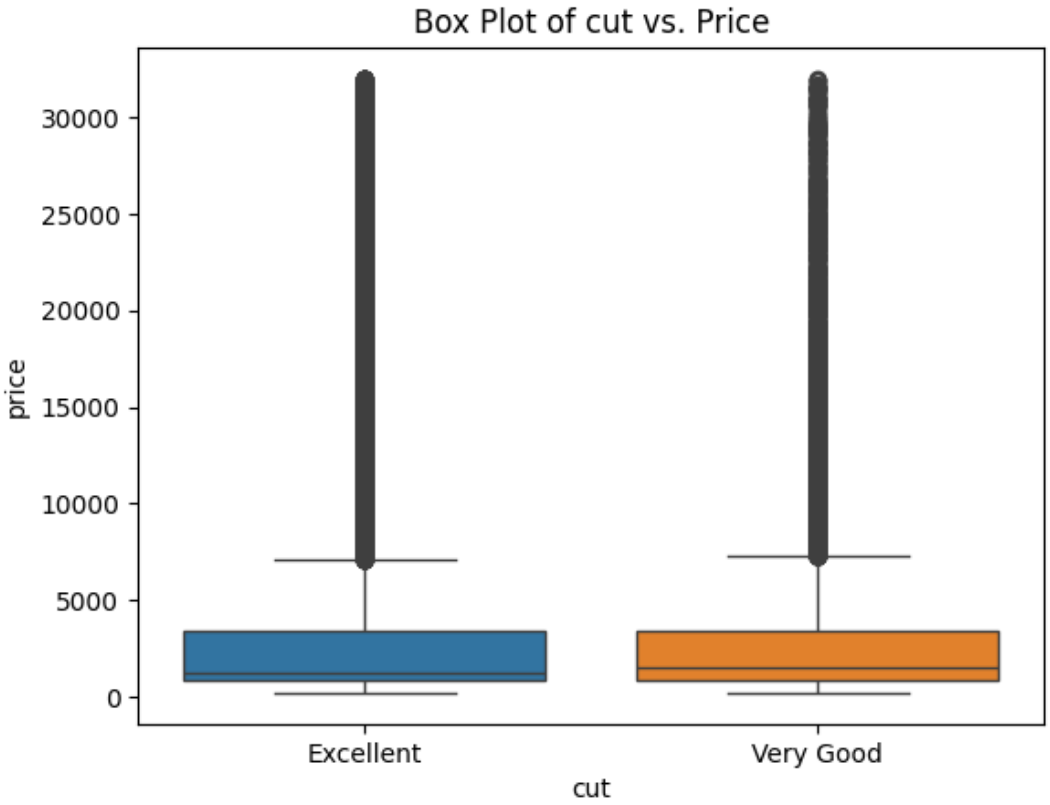
```
girdle_max_encoded    -0.521589
girdle_min_encoded    -0.45523

Skewnesss values before log transformation:-

carat                  0.844160
depth_percent        -16.200465
table_percent        -16.015117
length                 0.734470
width                  0.930143
depth                  3.361822
price                  0.897955
cut_encoded           -1.410662
symmetry_encoded      -1.174542
polish_encoded        -2.125702
color_encoded         -1.605959
clarity_encoded       -0.577243
girdle_max_encoded    -0.583845
girdle_min_encoded    -0.563389
```

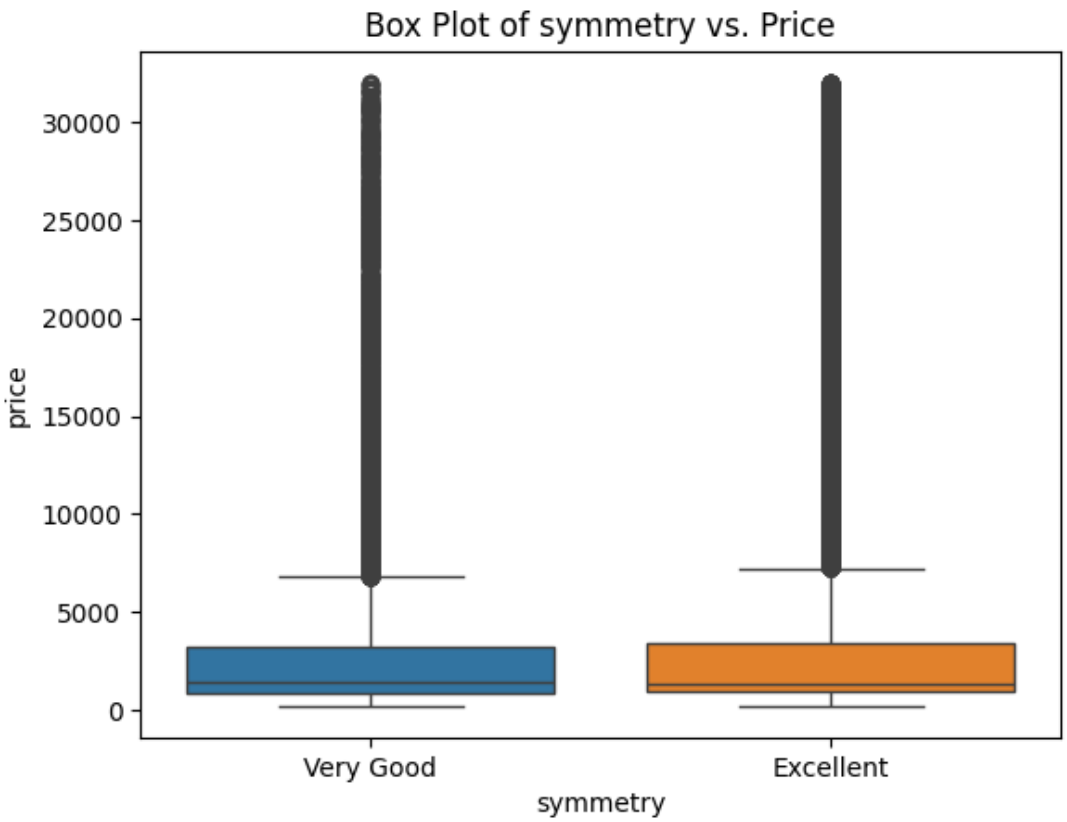
1.3.. Construct and inspect the box plot of categorical features vs target variable. What do you find?

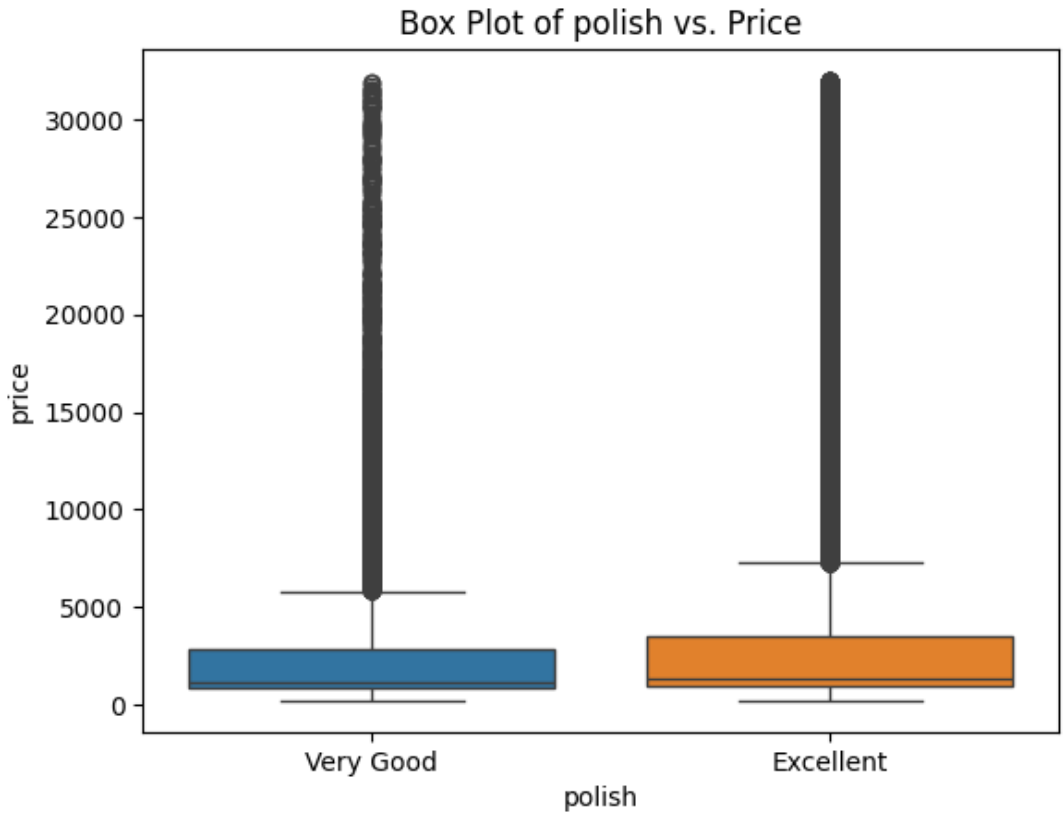
Answer:-

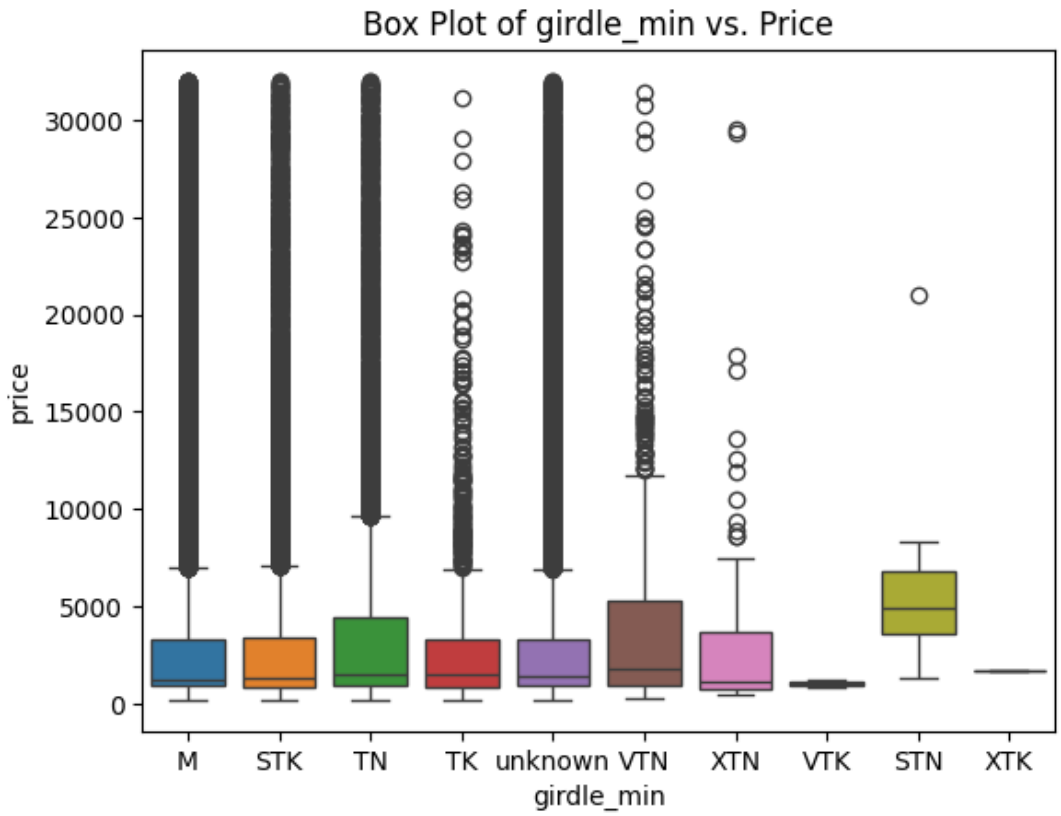


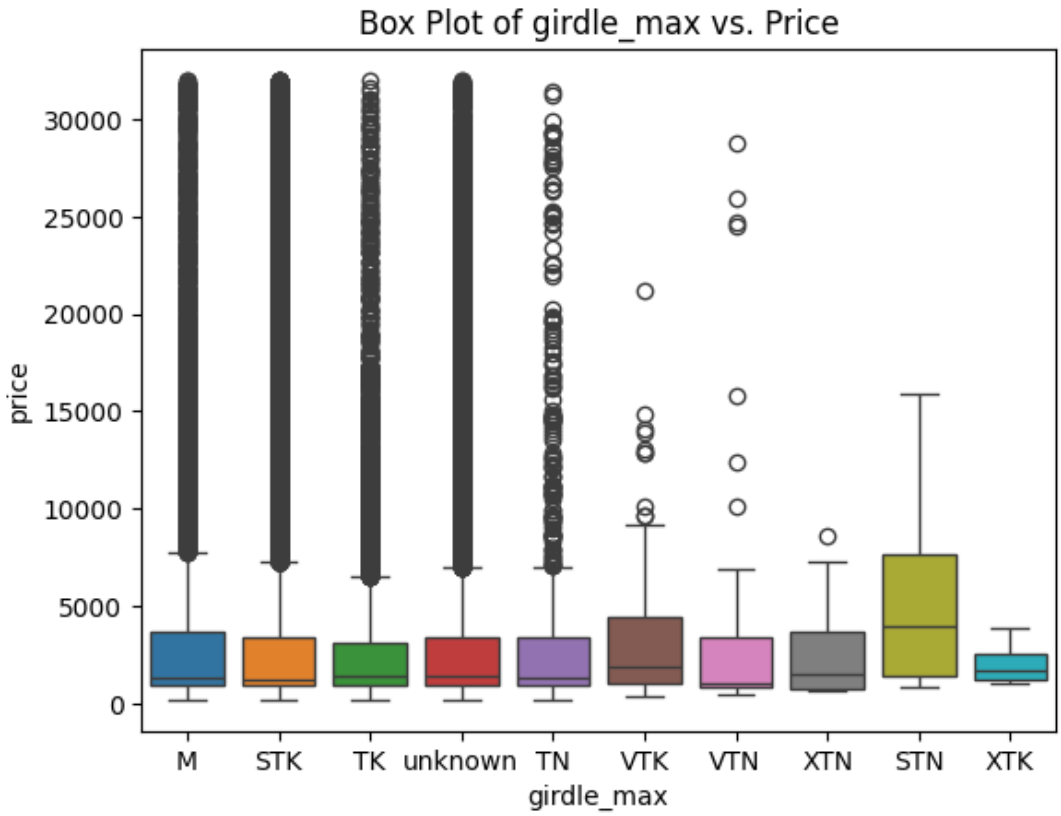

Box Plot of clarity vs. Price
price
30000
25000
20000
15000
10000
5000
0
VVS2
VVS1
I1
VS1
VS2
IF
SI2
I2
SI1
I3
clarity


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

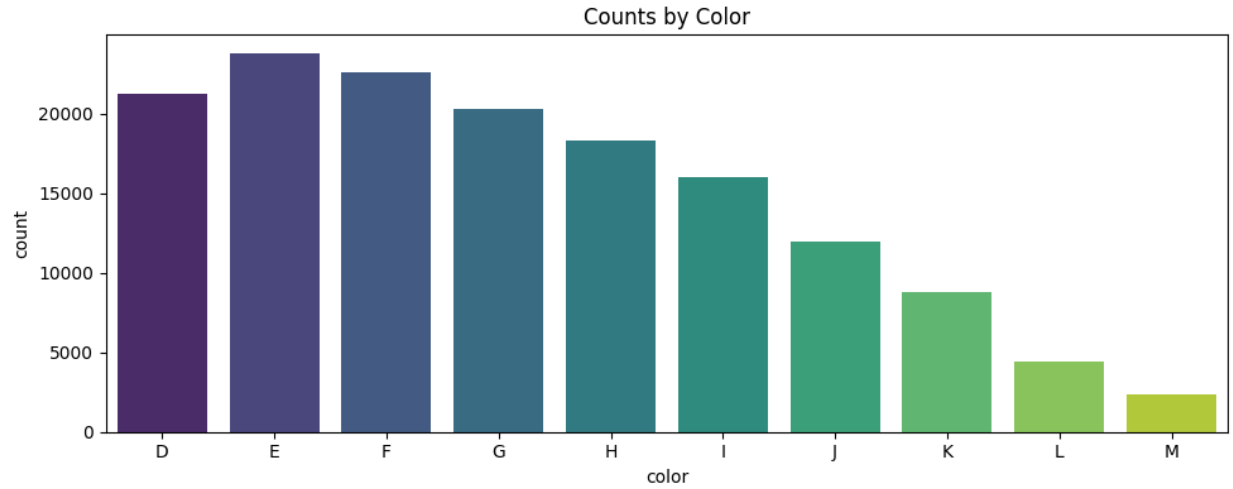

Box Plot of polish vs. Price
price
30000
25000
20000
15000
10000
5000
0
Very Good
Excellent
polish

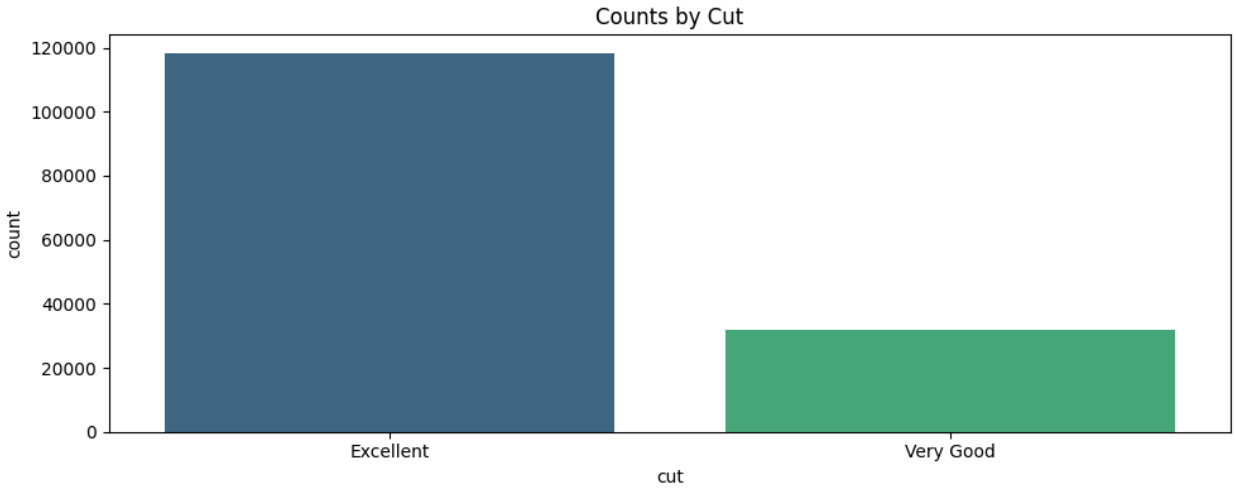

Box Plot of girdle_min vs. Price
price
0
5000
10000
15000
20000
25000
30000
M
STK
TN
TK
unknown
VTN
XTN
VTK
STN
XTK
girdle_min

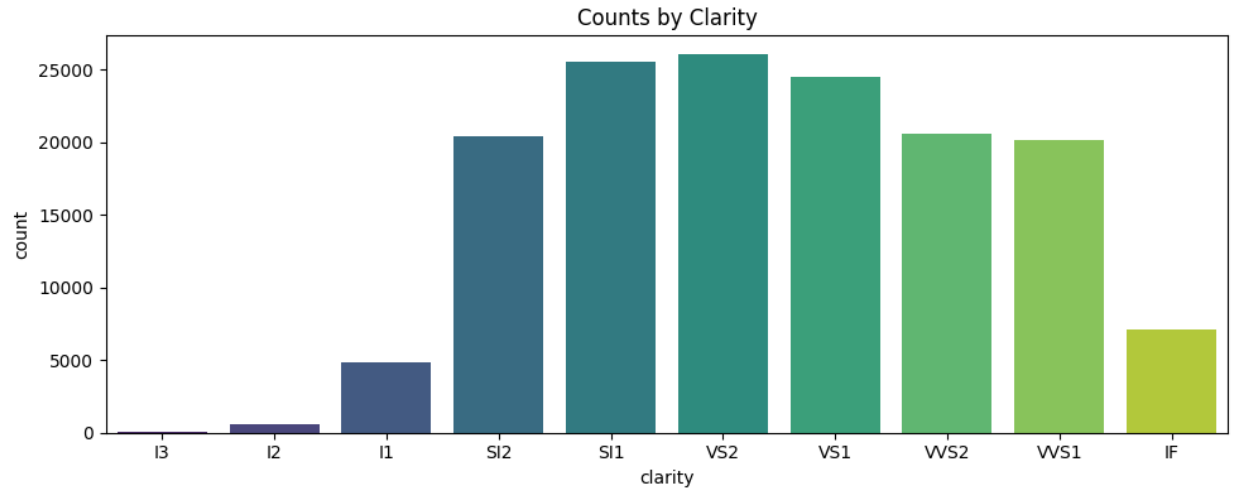

Box Plot of girdle_max vs. Price
price
0
5000
10000
15000
20000
25000
30000
M
STK
TK
unknown
TN
VTK
VTN
XTN
STN
XTK
girdle_max

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

Answer:-







2. Question 2

2.1. Standardize feature columns and prepare them for training.

Answer:-

The feature columns were standardized using the **Standard Scaler**. The standard score of a sample x is calculated as: $z = (x - u) / s$; where u is the mean of the training samples, and s is the standard deviation of the training samples. Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set.

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.

Answer:-

Feature selection holds significant importance in regression models as it identifies and utilizes only the essential features. Two commonly employed techniques for feature selection are mutual information (MI) regression and F regression. MI regression evaluates the dependency between two random variables, namely the feature and the target variable—in our case, the target variable being price. A higher MI value signifies a stronger dependency, indicating a noteworthy contribution to the target variable, price. Consequently, selecting features with higher MI regression values reduces the dataset to crucial features, thereby enhancing model performance. On the other hand, F scores gauge the significance of augmenting the model's performance by incorporating new variables (features). Higher F scores denote greater significance of the variable, implying that including features with high F scores can enhance the model's performance. This selection process prioritizes features that contribute most effectively to the model's performance.

Implementing these methods aids in enhancing model performance, particularly in regression models, by mitigating overfitting and utilizing only pertinent features. This approach enables the model to capture underlying patterns more effectively, resulting in improved generalization performance and reduced test root mean squared error (RMSE). However, the efficacy of these techniques is contingent upon the dataset and the regression model employed. Linear regression models benefit substantially from feature selection as they are susceptible to issues like multicollinearity and overfitting. Conversely, more complex models such as tree-based algorithms inherently perform feature selection, thereby minimizing the impact of MI regression or F score techniques.

```
Features with the lowest mutual information scores:

          Feature      MI Score     F Score        P Value
    polish_encoded    0.012233    453.535242    1.730325e-100
       cut_encoded    0.023815     88.958031    4.087555e-21

  Features with the lowest F scores:
         Feature      MI Score     F Score        P Value
  girdle_min_encoded  0.025192    0.099306      7.526642e-01
  girdle_max_encoded  0.037749    26.659077     2.430170e-07
```

3. Question 3

N/A

4. Question 4

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

Answer:-

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.

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

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

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

- Ordinary Least Squares (OLS) Regression: OLS aims to minimize the sum of squared residuals without any regularization. It estimates the parameters (coefficients) of the linear regression model by minimizing the sum of squared differences between the observed and predicted values. Without regularization, OLS may lead to overfitting, especially when dealing with high-dimensional datasets or multicollinearity among features.

- Lasso Regression: Lasso adds an L1 regularization term to the objective function, which penalizes the absolute values of the coefficients. This penalty encourages sparsity in the parameter set by shrinking some coefficients to exactly zero. As a result, Lasso can perform feature selection by automatically removing irrelevant features from the model.

- Ridge Regression: Ridge adds an L2 regularization term to the objective function, which penalizes the squared values of the coefficients. This penalty encourages smaller coefficients and shrinks them towards zero. However, Ridge does not force coefficients to be exactly zero, allowing all features to be included in the model. Ridge can improve the conditioning of the problem and reduce the variance of the estimates, especially when multicollinearity is present among the features.

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

Answer:-

We tried regression on unscaled data and scaled reduced data. Multiple values of alpha were tried to check which gives the lowest RMSE. 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 1570.88201356099

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

Answer:-

Feature standardization (scaling) is essential in Ridge regularization because Ridge penalizes the squared values of the coefficients, and the magnitude of the coefficients directly affects the regularization term. Features with larger scales may dominate the penalty term, leading to biased parameter estimates. By standardizing the features to have zero mean and unit variance, all features are put on the same scale, preventing any single feature from dominating the regularization process. Therefore, feature standardization can play a crucial role in improving the model performance when using Ridge regularization.

In our specific case, we observe that standardization did not substantially alter RMSE values. However, the necessity of feature scaling depends on the regression type and dataset characteristics. For non-normally distributed data or specific regression types, feature scaling may not significantly impact RMSE or could even increase it.

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.

Answer:-

The p-values associated with each feature in linear regression indicate the statistical significance of that feature in predicting the target variable. A small p-value (typically less than a chosen significance level, e.g., 0.05) indicates that the null hypothesis, which states that the coefficient of the feature is zero (no effect), can be rejected. Therefore, features with small p-values are considered statistically significant predictors of the target variable. On the other hand, features with large p-values are not statistically significant predictors. Therefore, to infer the most significant features, you can look for features with the smallest p-values, indicating high statistical significance. These features are likely to have a meaningful impact on the target variable.

The p-values obtained are as follows:-

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

5. Question 5

5.1. 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: What are the most salient features? Why?

Answer:-

```
Top 10 most salient features:

      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
```

The predominant features align closely with those identified in the heatmap analysis. These features exhibit the strongest correlation with the target variable, price. Therefore, these notable features are instrumental in prediction and exert a significant influence on the target variable.

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?

Answer:-

After testing various polynomial degrees, it was found that the degree mentioned below performed the best. Additionally, multiple alpha values were experimented with to identify the one yielding the lowest RMSE.

```
Optimal degree: 3
Average RMSE for optimal degree: 765.0979014891634
```

When employing polynomial regularizations with degrees greater than 3, the model faces significant penalties during training and backpropagation. This penalization substantially diminishes the model's ability to effectively learn and adapt. Utilizing multiplicative combinations of more than 3 input features from the dataset for regularization purposes is deemed unfair in regulating the model's learning behavior during overshoot or undershoot.

A polynomial of very high order indicates a highly flexible model capable of capturing intricate patterns and fluctuations in the training data. However, as the polynomial degree increases, the model's complexity also rises, enabling it to closely fit the training data points and potentially achieve a low training error. Nevertheless, this heightened flexibility poses a risk of overfitting, where the model captures not only the underlying patterns but also the noise present in the data.

Overfitting can significantly degrade the model's performance on unseen data since it may excessively adhere to the training data, resulting in poor generalization. In the context of high-order polynomial regression, this overfitting risk is exacerbated, potentially leading to a considerable decline in the model's performance when applied to testing data.

6. Question 6

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

Answer:-

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

Answer:-

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.

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: 662.585629480836

The values obtained for regression on scaled reduced data is as below:

- Average RMSE for Ordinary Least Squares (OLS): 1571.035361861907
- Average RMSE for Lasso Regression: 1570.88201356099
- Average RMSE for Ridge Regression: 1571.0354586551564

Linear regression got RMSE in the range of 1500s. As expected NN has performed better than linear regression.

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

Answer:-

The chosen activation function is Rectified Linear Unit (ReLU), primarily for its simplicity and efficiency. Defined as f(x) = max(0, x), ReLU involves straightforward arithmetic operations, facilitating swift computation during both forward and backward propagation. Moreover, ReLU induces sparse activation, activating only a subset of neurons within a layer. Unlike sigmoid and tanh activation functions, ReLU does not saturate for positive input values, thus circumventing the vanishing gradient problem. Its non-saturating nature and simplicity often result in faster convergence during training. Consequently, ReLU activation remains a popular choice in neural networks due to its simplicity, efficiency, ability to prevent vanishing gradients, and propensity to expedite convergence during training.

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

Answer:-

Expanding the depth of a neural network beyond a certain threshold introduces additional risks, with overfitting being the most prevalent. As the network deepens, it gains greater capacity to memorize data, potentially fitting noise alongside relevant patterns. Deeper networks are particularly susceptible to issues like vanishing or exploding gradients during training. Additionally, deeper architectures necessitate increased computational resources for training and evaluation, with the number of parameters and computations growing exponentially with depth. Consequently, training becomes progressively more challenging with deeper networks, often encountering issues with gradient descent convergence. Gradients may diminish or amplify excessively as they propagate through numerous layers, resulting in slow convergence or divergence during training. Notably, augmenting network depth doesn't consistently translate to improved performance, especially when dealing with small datasets or relatively straightforward problems.

7. Question 7

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

Answer:-

The hyperparameter "Maximum number of features" determines the maximum number of features considered during splitting at each node in every decision tree. A higher value increases the number of splits, yielding more diverse trees capable of capturing complex data patterns and relationships. However, setting this value too high for high-dimensional data can lead to overfitting. To mitigate overfitting, reducing the maximum number of features limits tree diversity, promoting simpler models.

The hyperparameter "Number of trees" specifies the quantity of decision trees within the random forest. As the number of trees increases, the model's robustness and scalability improve, reducing variance in predictions. Nonetheless, increasing the number of trees escalates computational complexity and training duration. Conversely, employing too few trees can result in underfitting. Consequently, determining the optimal number of trees is crucial.

The hyperparameter "Depth of each tree" dictates the maximum depth of individual decision trees in the random forest. Greater depth allows for capturing increased complexity, yet deeper trees may lead to overfitting, particularly with noisy or outlier-laden data. Constraining tree depth can regularize the model by favoring simpler tree structures and diminishing the risk of overfitting.

Each of these hyperparameters can act as regularization mechanisms during training, controlling the complexity of the random forest model and preventing overfitting. Regularization aims to simplify models and facilitate generalization to unseen data. By identifying optimal values for these hyperparameters, a balance between model complexity and generalization performance can be achieved, resulting in a more robust and effective random forest model.

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?

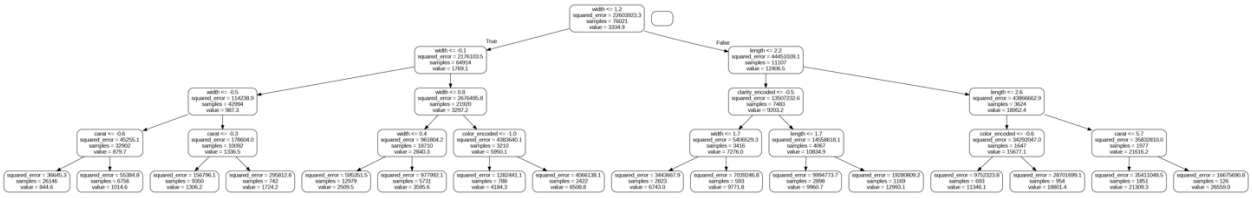
Answer:-

Random forests operate by combining multiple decision trees, each trained on different subsets of both data and features. Employing ensemble learning, random forests train multiple decision trees independently and then aggregate their predictions. Within this ensemble, each decision tree aims to capture distinct aspects of the data, resulting in a diverse array of base models. Additionally, random forests incorporate feature randomization, wherein only a random subset of features is considered for splitting at each node of every decision tree. This diversity ensures that each tree learns a different decision boundary, allowing the random forest to effectively capture a wide range of data features and patterns.

During prediction, random forests aggregate the predictions of individual decision trees. Notably, they exhibit robustness to data noise and outliers, making them less prone to overfitting compared to individual decision trees. The ensemble nature of random forests enables them to generalize well to unseen data and capture complex, nonlinear relationships between input features and the target variable. Essentially, a random forest comprises numerous uncorrelated models, with the final prediction derived from a majority decision fusion across all trees.

7.3. 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?

Answer:-



The attribute chosen at the initial node for branching is width. This attribute serves as the pivotal point for branching at the starting node, indicating its paramount importance in initiating the splitting process. In a decision tree structure, attributes closer to the root node hold greater significance compared to those nearer to the leaf nodes. Upon evaluating p-values, it was noted that the most influential attributes were carat, color, and clarity. This observation underscores the importance of carat, color, and clarity in this context as well. Consequently, it can be inferred that the key features identified through linear regression's p-values substantially coincide with the critical features identified in a random decision tree within the random forest framework for the Diamond dataset.

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

Answer:-

“Out-of-Bag Error” (OOB) Values: 0.019290294968740906

The Out-of-Bag (OOB) error and R2 score are both important metrics used to assess the performance of machine learning models, particularly in the context of regression and ensemble methods like Random Forests. The OOB error serves as a measure of a model's predictive accuracy on unseen data points. In Random Forest, each decision tree is trained on a bootstrap sample of the original dataset, with some data points left out (out-of-bag). The OOB error is then calculated as the average error rate of each individual tree on these out-of-bag data points. This provides an estimate of how well the model generalizes to new data without the need for a separate validation set. On the other hand, the R2 score, also known as the coefficient of determination, quantifies the proportion of variance in the dependent variable (target) that is explained by the independent variables (features). It ranges from 0 to 1, with higher values indicating a better fit of the model to the observed data. However, it's important to note that while a high R2 score suggests a good fit, it doesn't necessarily guarantee the model's accuracy or its ability to generalize to unseen data. Therefore, both the OOB error and R2 score should be interpreted in conjunction with other evaluation metrics and validated on independent datasets to ensure robust model performance.

8. Question 8

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).

Answer:-

I've employed LightGBM, with several key hyperparameters to consider:

1. num_leaves: This parameter governs the maximum number of leaves in each tree. Higher values increase model complexity, potentially leading to overfitting.

2. learning_rate: It regulates the step size during gradient boosting. Lower values yield slower learning rates but may enhance generalization capabilities.

3. max_depth: This parameter sets the maximum depth of each tree. Elevated values amplify model complexity and the risk of overfitting.

4. min_data_in_leaf: This determines the minimum number of data points required in each leaf. Larger values can prevent overfitting but may increase the risk of underfitting.

5. feature_fraction: It specifies the fraction of features considered for each tree. Decreasing this value introduces randomness, aiding in reducing overfitting.

6. boosting_type: This determines the type of boosting technique utilized during training. Options include gbdt (Gradient Boosting Decision Tree), rf (Random Forest), dart (Dropouts meet Multiple Additive Regression Trees), and goss (Gradient-based One-Side Sampling). Each technique introduces variations in how weak learners are combined within the ensemble, impacting model performance and robustness.

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

Answer:-

```
Best Hyperparameters are as follows:

      '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
```

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?

Answer:-

Various hyperparameters exert different effects on the model's performance, regularization, and fitting efficiency:

1. Performance Improvement: Hyperparameters that directly impact the model's ability to discern patterns in the data or diminish error typically enhance performance. Increasing the number of trees, for instance, augments the model's predictive prowess by incorporating a more diverse array of robust decision boundaries.

2. Regularization: This governs the model's complexity and aids in averting overfitting, thus diminishing the generalization gap. Parameters like the regularization strength (lambda) in ridge regression or the maximum tree depth in decision trees can be adjusted to strike a balance between effectively fitting the training data and evading overfitting on unseen data.

3. Fitting Efficiency: Certain hyperparameters influence the model's fitting efficiency, impacting the speed at which 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 influence the convergence speed and overall efficiency of the training process.

In Bayesian optimization, the objective is to pinpoint the optimal combination of hyperparameters that maximizes model performance while ensuring good generalization and efficient training. By evaluating the impact of different hyperparameters on these aspects, one can identify those that significantly contribute to each criterion and adjust them accordingly to attain superior overall results. Experimentation with various hyperparameters reveals insights. For instance, increasing the number of trees typically enhances model performance by capturing more intricate data patterns, yet excessive trees may risk overfitting. Lowering the learning rate often results in steadier but more robust learning, facilitating improved generalization. Deeper trees may capture complex data relationships but can also lead to overfitting, especially with smaller datasets. Random subsampling of training data for each tree introduces regularization, safeguarding against overfitting. Tuning these hyperparameters using Bayesian optimization or other methods enables finding a balance between model performance, regularization, and fitting efficiency, ultimately optimizing results for the specific task and dataset at hand.

9. Question 9

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.

Answer:-

```
--------------------------------------------------------------------------

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

--------------------------------------------------------------------------
```

9.2. Plot “number of tweets in hour” over time for #SuperBowl and #NFL (a bar plot with 1-hour bins)

Answer:-

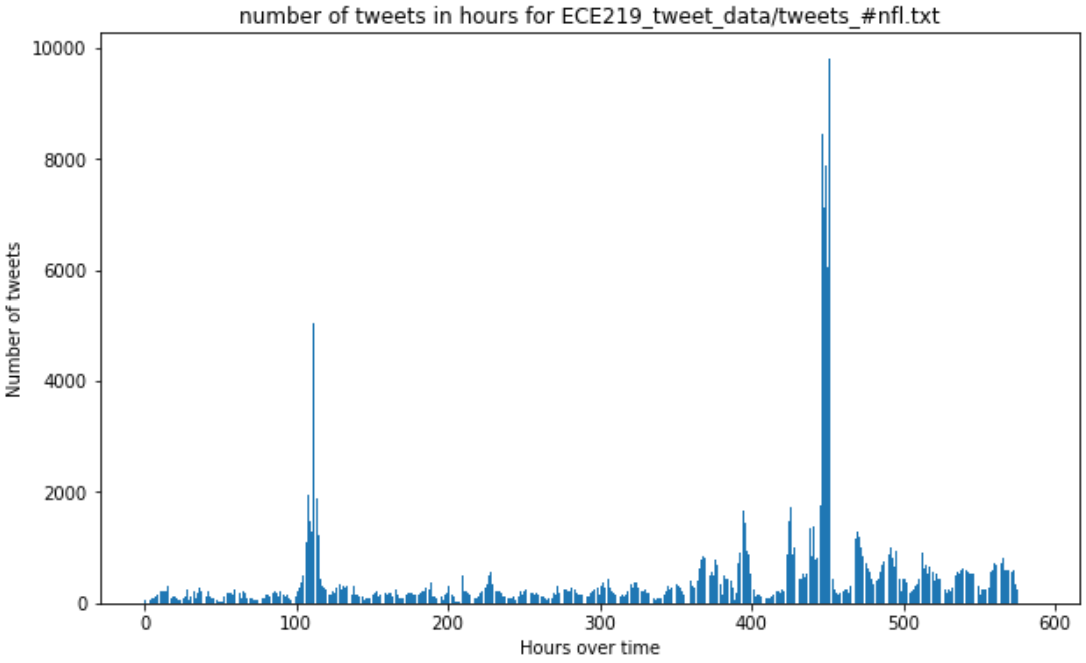


Fig: Number of Tweets in Hour for SupoerBowl

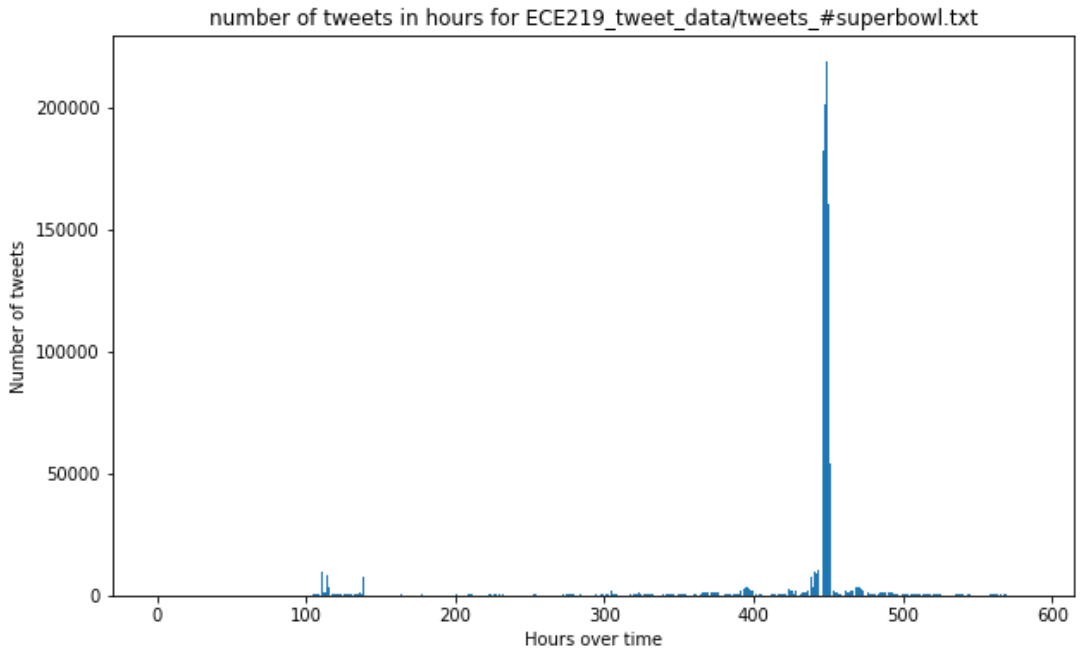


Fig: Number of Tweets in Hour for NFL

10. Question 10

The textual content of a tweet can reveal some information about the author. For instance, users tweeting on a topic may have opposing views about it. In particular, tweets posted by fans of different teams during a sport game describe similar events in different terms and sentiments.

10.1. Recognizing that supporting a sport team has a lot to do with the user location, I tried to use the textual content of the tweet posted by a user to predict their location. In order to make the problem more specific, I considered all the tweets including #superbowl, posted by the users whose specified location is either in the state of Washington (not D.C.!) or Massachusetts.

1. To find out the tweets whose authors belong to Washington, we go to the location field of each tweet (json object['tweet']['user']['location']) and decide whether the location contains any of the following substrings: {'Washington', 'Seattle', 'WA'}. If it is true, then we extract the tweet content together with the location for later textual data classification analysis. Similarly, to find out the tweets whose authors belong to Massachusetts, the corresponding substrings list is {'Massachusetts', 'Boston', 'MA'}.

2. Before training, we first clean the textual data by lemmatizing each word token in the tweet content. After that, we calculate the tf-idf metrics of the textual content using TfidfVectorizer from sklearn.feature extraction.text package. Last but not least, we perform dimension reduction through singular value decomposition on the tf-idf metrics as our final input features.

3. Since the task here is to predict the location of the author (Washington or Massachusetts) purely based on the textual content of the tweet, we are dealing with a binary classification problem. Three classification learning algorithms considered here are logistic regression, random forest classifier, and gradient boosting classifier, and we will report their respective model performance on the left-out testing data.

   First off, we perform a grid search to find out the best hyper-parameter settings for logistic regression, random forest and gradient boosting respectively. For logistic regression, we are grid-searching both penalty category and regularization strength. While for both random forest and gradient boosting, due to the large volume of this twitter dataset, we will limit our search scope and mainly focus on one argument max depth. The top 5 hyper-parameter settings through cross-validation are reported in the tables below.

mean_test_score	param_C	param_penalty
0.724697	100	l2
0.724651	10	l2
0.724651	1000	l2
0.723890	1	l2
0.719582	0.1	l2

Table: Grid Search Result for Logistic Regression on Location Prediction

mean_test_score	max_depth
0.724766	30
0.722623	70
0.722623	100
0.722623	200
0.722623	None

Table: Grid Search Result for Random Forest on Location Prediction

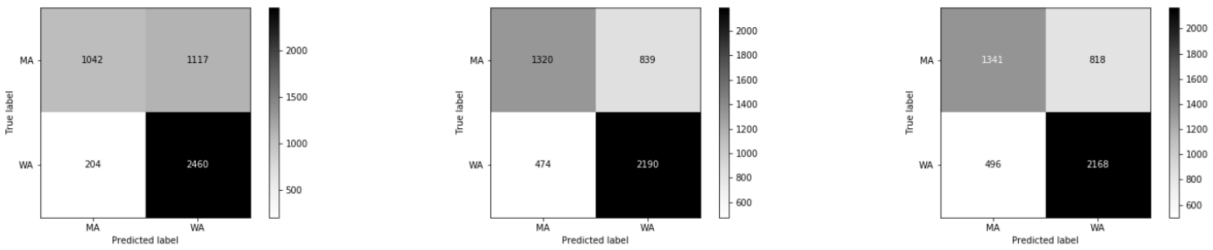
mean_test_score	max_depth
0.721978	10
0.698132	30
0.687141	70
0.686980	100
0.686980	200

Table: Grid Search Result for Gradient Boosting on Location Prediction

From there, we further predict the location of author for the left-out testing textual data, and calcu- late accuracy, recall, precision, F-1 score, as well as plotting the confusion matrices for three different classification algorithms as follows:-

	Logistic Regression	Random Forest	Gradient Boosting
Accuracy	0.7261	0.7278	0.7278
Recall	0.9234	0.8221	0.8221
Precision	0.6877	0.7230	0.7230
F-1 Score	0.7883	0.7694	0.7694

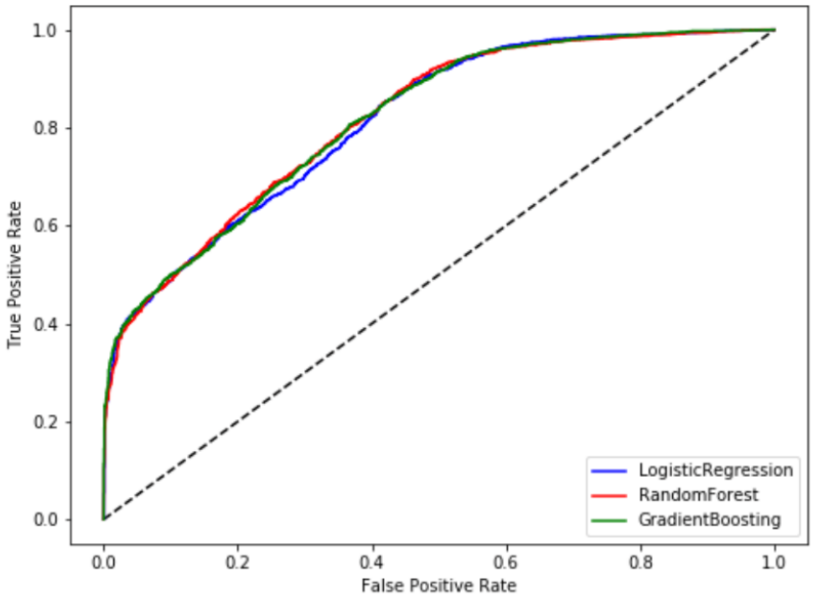
Table: Classification Measures for Three Different Classification Algorithms



Logistic Regression Random Forest Gradient Boosting

Figure: Confusion Matrices for Three Different Classification Algorithms

Last but not least, we also report the ROC curves for logistic regression, random forest classifier and gradient boosting classifier in one figure for better comparison.



In summary, logistic regression performs worst in terms of AUC score, while random forest and gradient boosting share similar prediction performance.

10.2. Now, we would like to perform sentiment analysis on fans of the opponent teams especially when the Super Bowl event was active, to have an overall understanding about how their emotion changes as the game going on.

Since we have no idea about which team the author of a tweet supports, we simply take any tweet with the hashtag #gohawks as tweets posted by Seattle Seahawks' fans, and take any tweet with hashtag #gopatriots as tweets posted by New England Patriots' fans. Then, we predict the sentiment polarity given the textual content of each tweet using a well-trained predictive model called TextBlob. We further aggregate the sentiment polarity using a 1-hour window and

calculate the average polarity within each window for tweets posted from Feb. 1, 8:00 a.m. to Feb. 1, 8:00 p.m.. The result is shown in Figure below.

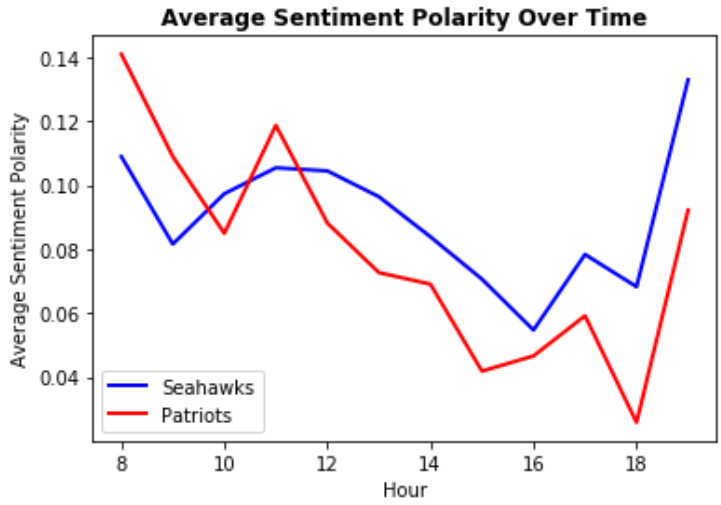
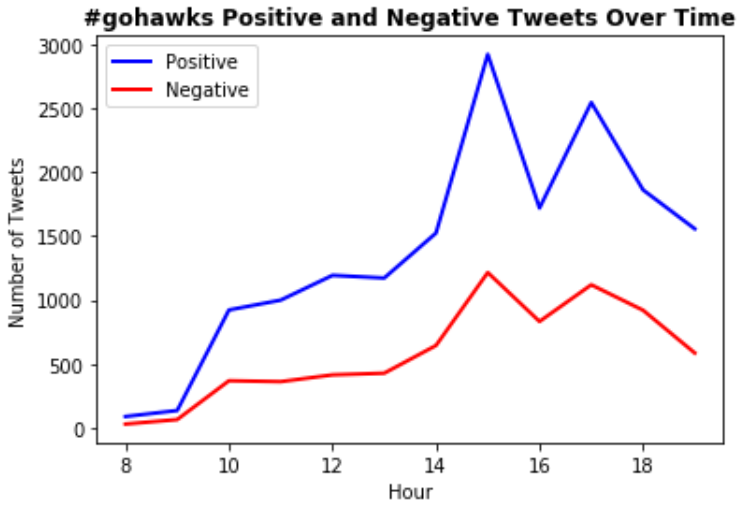
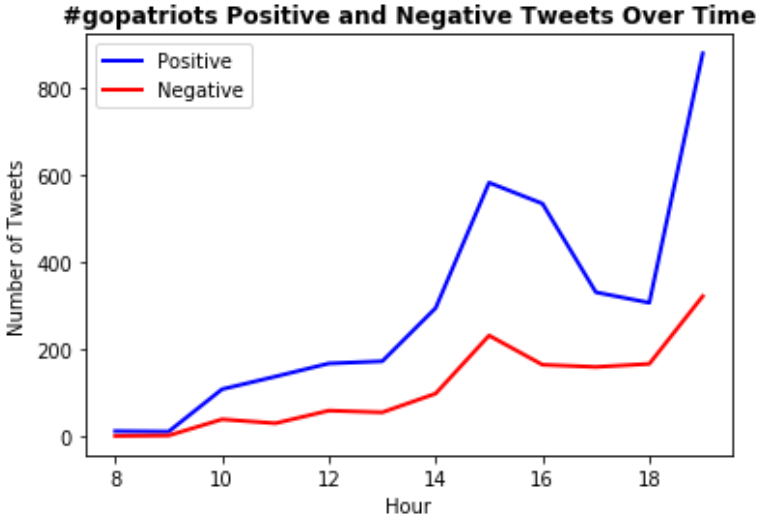


Figure: Average Sentiment Polarity Over Time For Fans of Opponent Teams

From this sentiment analysis, we figure out that sentiment polarity of fans of New England Patriots dropped significantly around 6 p.m. compared to fans of Seattle Seahawks. According to the game record in history, we would ascribe this phenomenon to the fact that team New England Patriots lost the third quarter to Seattle Seahawks with 0-10, and their fans must be frustrated at that point consequently. Moreover, polarity of fans from both teams increases dramatically at the end of the game, since scores were very close when the game was approaching to the end, and The Patriots rallied to take a 28-24 lead at the last moment.

10.3. We define a tweet to be positive if its corresponding sentiment polarity is positive, and vice versa. Based on this, we can plot the changes of number of positive and negative tweets for fans from both teams (Figure below) to perform another sentiment analysis




#gopatriots Positive and Negative Tweets Over Time
Positive
Negative
Number of Tweets
0
200
400
600
800
8
10
12
14
16
18
Hour