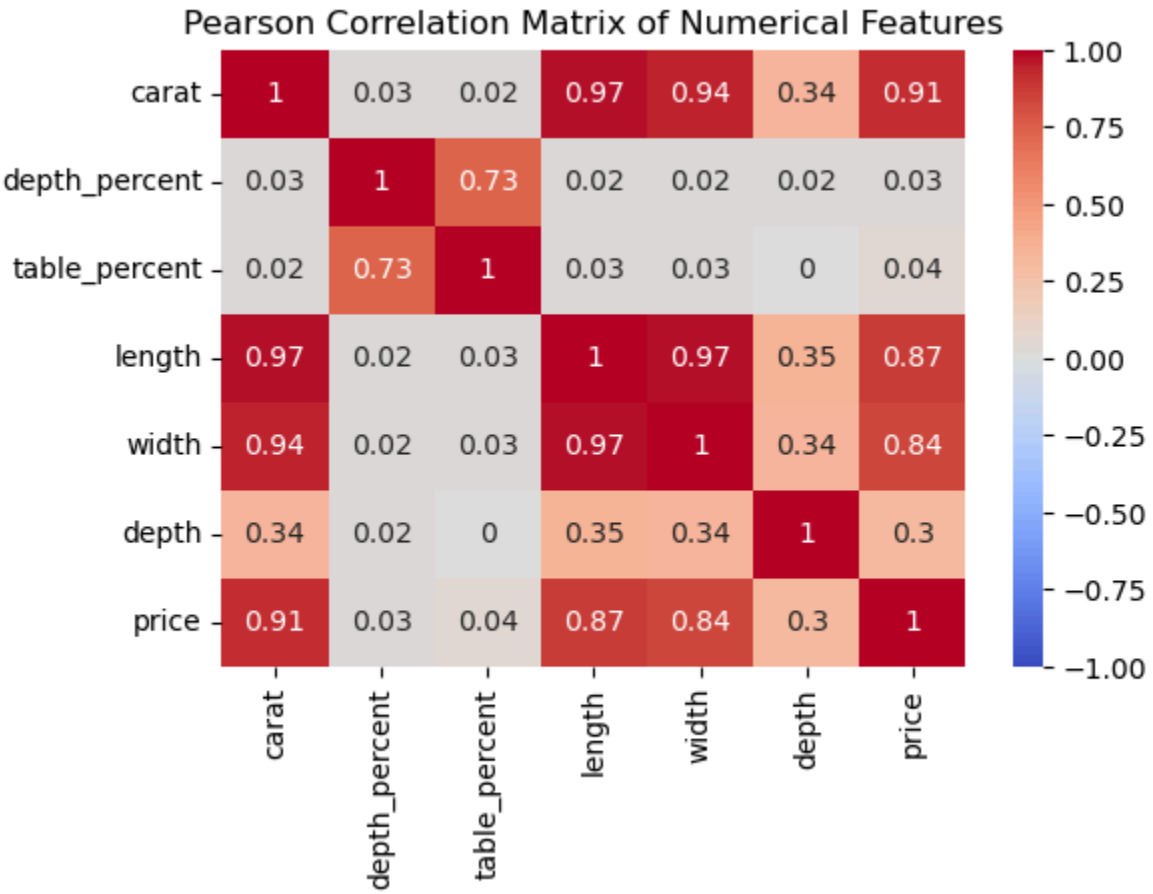
ECE 219 Project 4 Report

Shangyan Shen 106410632 uclassy2024@g.ucla.edu
Tianhao Zang: 706041741 tianhaoz77@g.ucla.edu
Yan Sun: 006406999 yas063@g.ucla.edu

Part 1 Regression Analysis:

Question 1.1



1. Variables that have strong correlations with price
- Carat: 0.91 - Extremely strong positive correlation with price, indicating that as the carat weight increases, the price increases substantially. This is the strongest predictor of price.

- Length: 0.87 - Very strong positive correlation with price.
- Width: 0.84 - Very strong positive correlation with price.

2. Correlation patterns suggest:

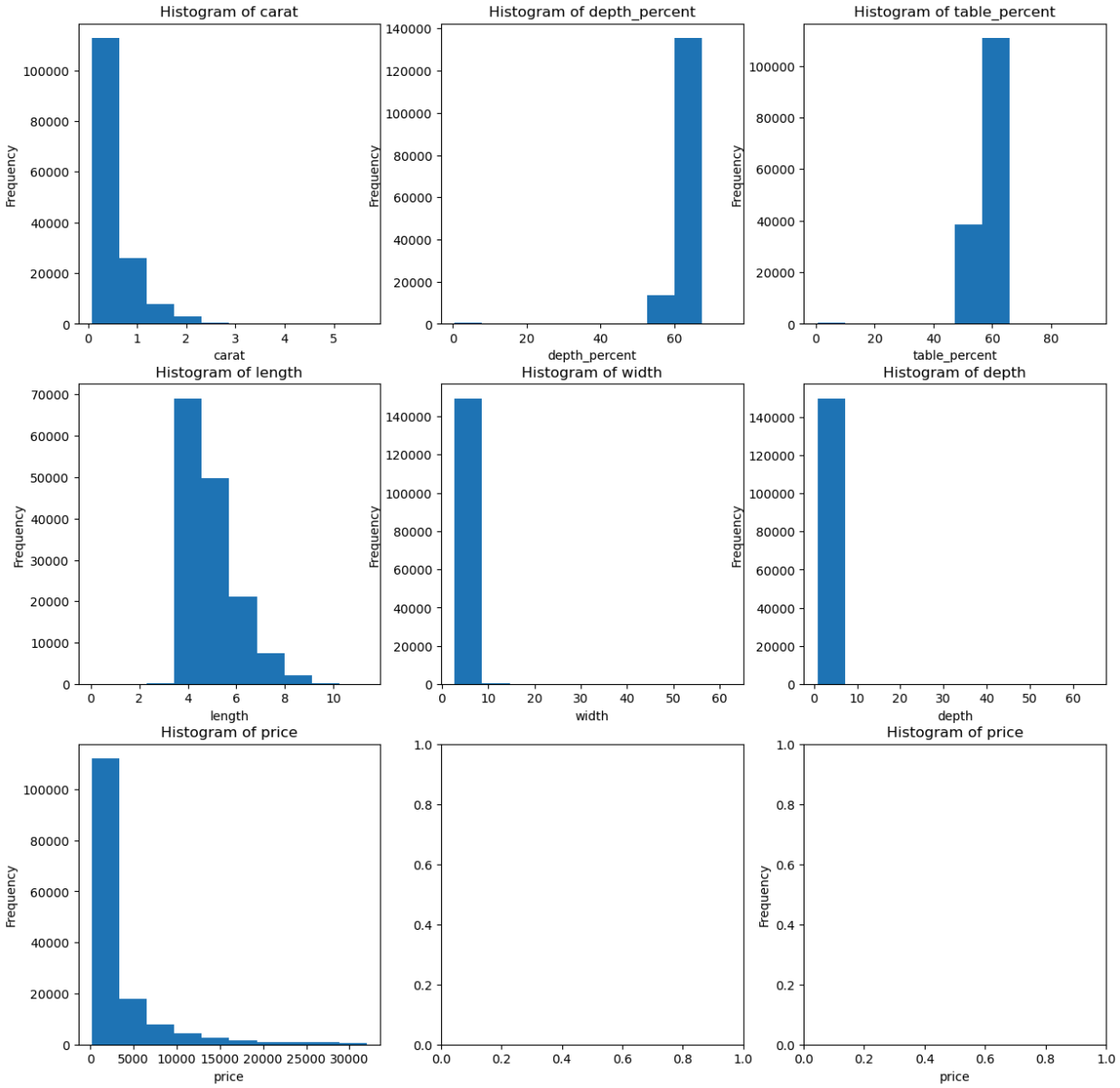
(1) **Multicollinearity Considerations** - Carat, Length, and Width: These three features are highly intercorrelated (0.94-0.97), which makes sense as larger diamonds (by carat weight) will naturally have greater physical dimensions. - Depth_percent and Table_percent: These show a strong correlation (0.73) with each other but minimal correlation with price.

(2) **Feature Importance:** Depth_percent and table_percent likely contribute little to price prediction and could potentially be excluded from a predictive model without significant loss of accuracy.

(3) **Dimension Reduction:** Given the high multicollinearity, dimension reduction techniques might be beneficial before building predictive models.

Question 1.2

Histogram for numerical features:

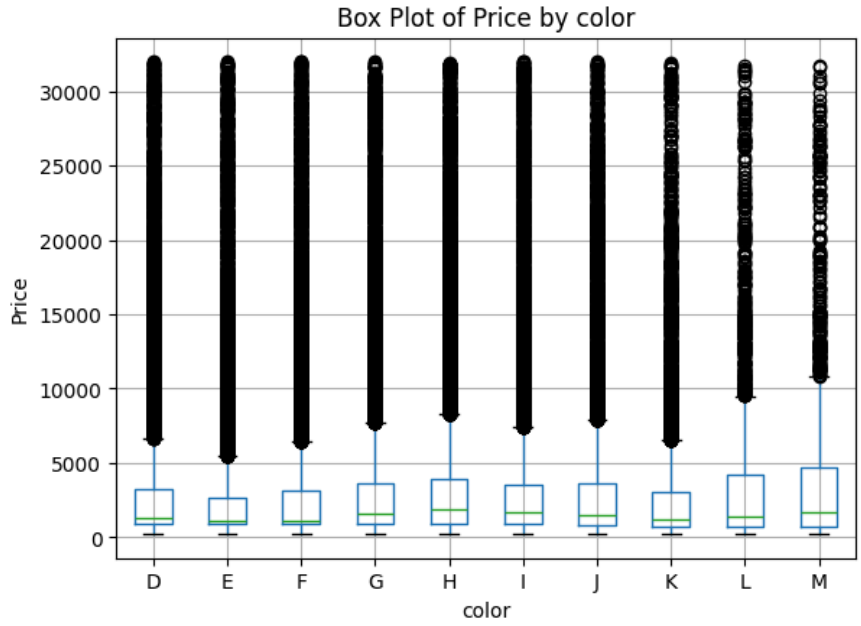


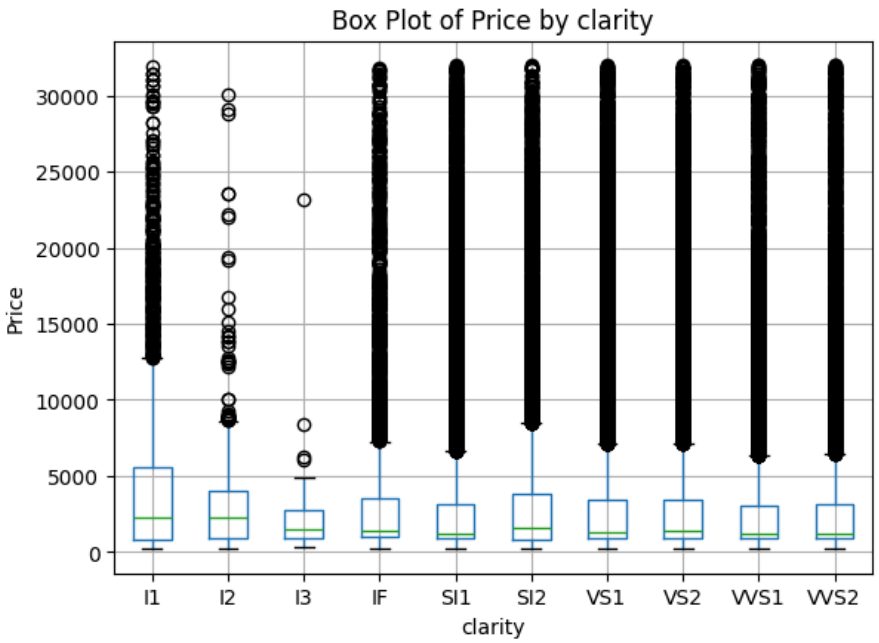
For high skewness, we could apply natural log transform to reduce the right skew. When a variable has a right-skewed distribution, it typically has a heavy tail with a small number of very large values. A log transformation compresses those larger values more than the smaller ones, pulling in the tail of the distribution. Because of that compression effect, the distribution's tail becomes shorter or "closer" to the rest of the data, reducing the overall

skew. If the data roughly span several orders of magnitude, log-transforming often shifts the distribution closer to a normal-like shape.

Question 1.3

Box plot of categorical features:

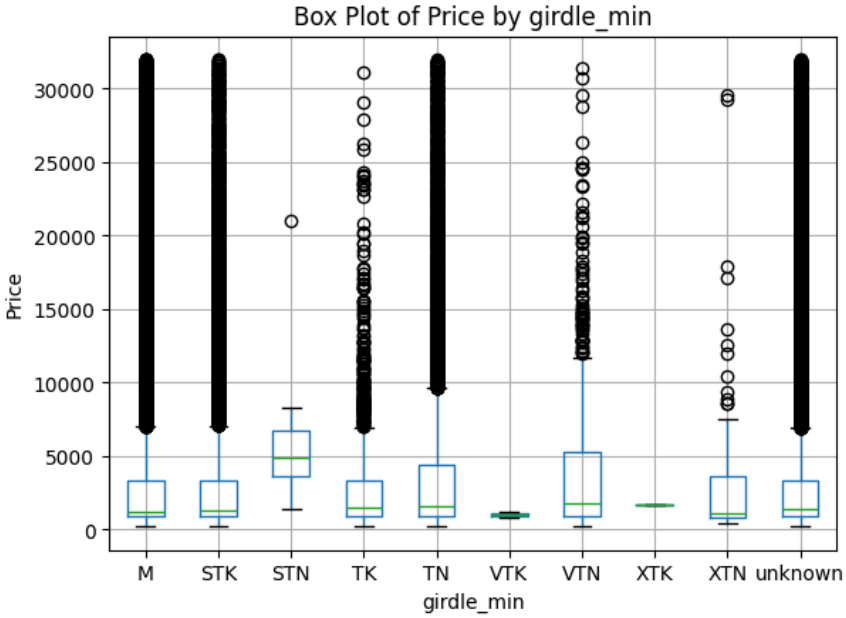


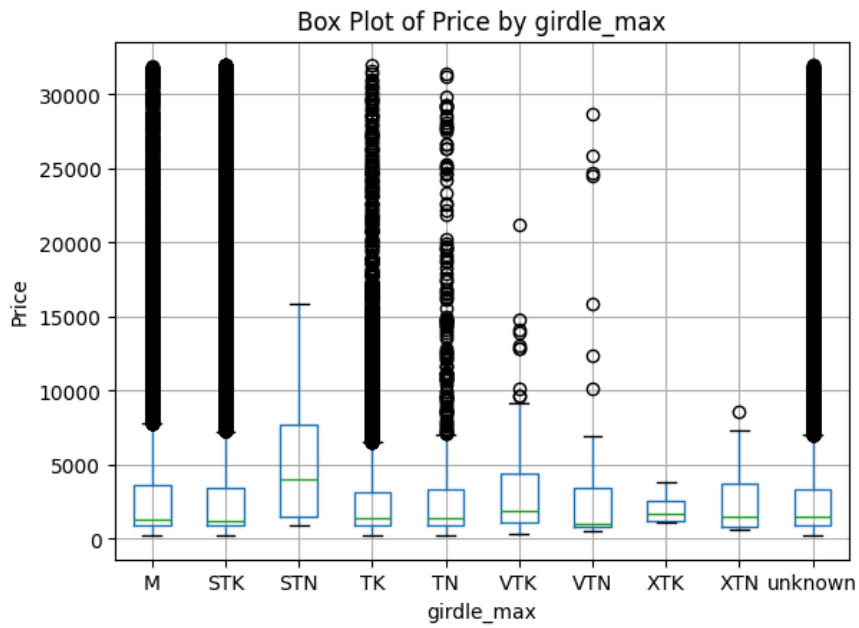

Box Plot of Price by clarity
Price
30000
25000
20000
15000
10000
5000
0
I1
I2
I3
IF
SI1
SI2
VS1
VS2
VVS1
VVS2
clarity


Box Plot of Price by cut
Price
30000
25000
20000
15000
10000
5000
0
Excellent
Very Good
cut


Box Plot of Price by symmetry
Price
30000
25000
20000
15000
10000
5000
0
Excellent
Very Good
symmetry

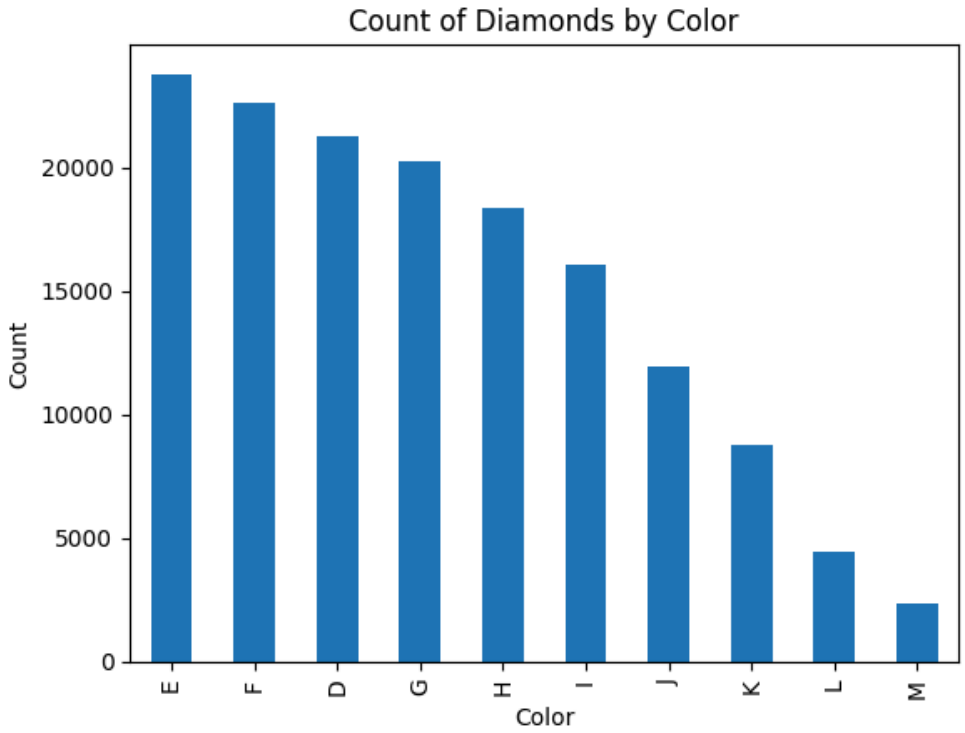

Box Plot of Price by polish
Price
30000
25000
20000
15000
10000
5000
0
Excellent
Very Good
polish

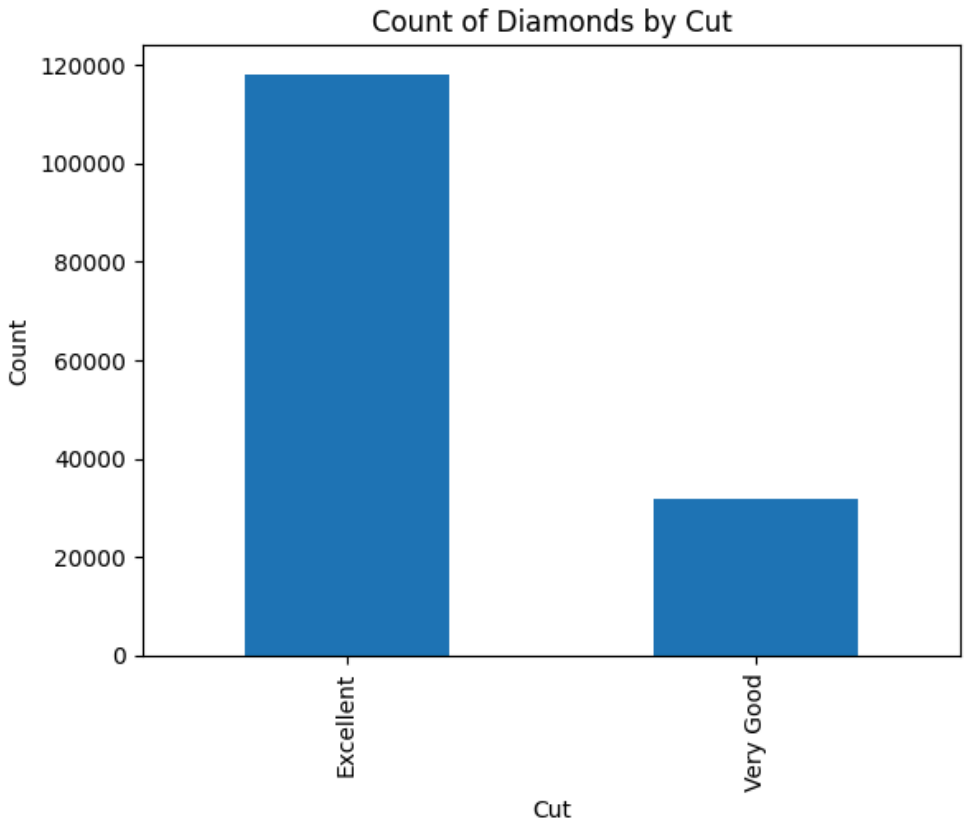

Box Plot of Price by girdle_min
Price
30000
25000
20000
15000
10000
5000
0
M
STK
STN
TK
TN
VTK
VTN
XTK
XTN
unknown
girdle_min

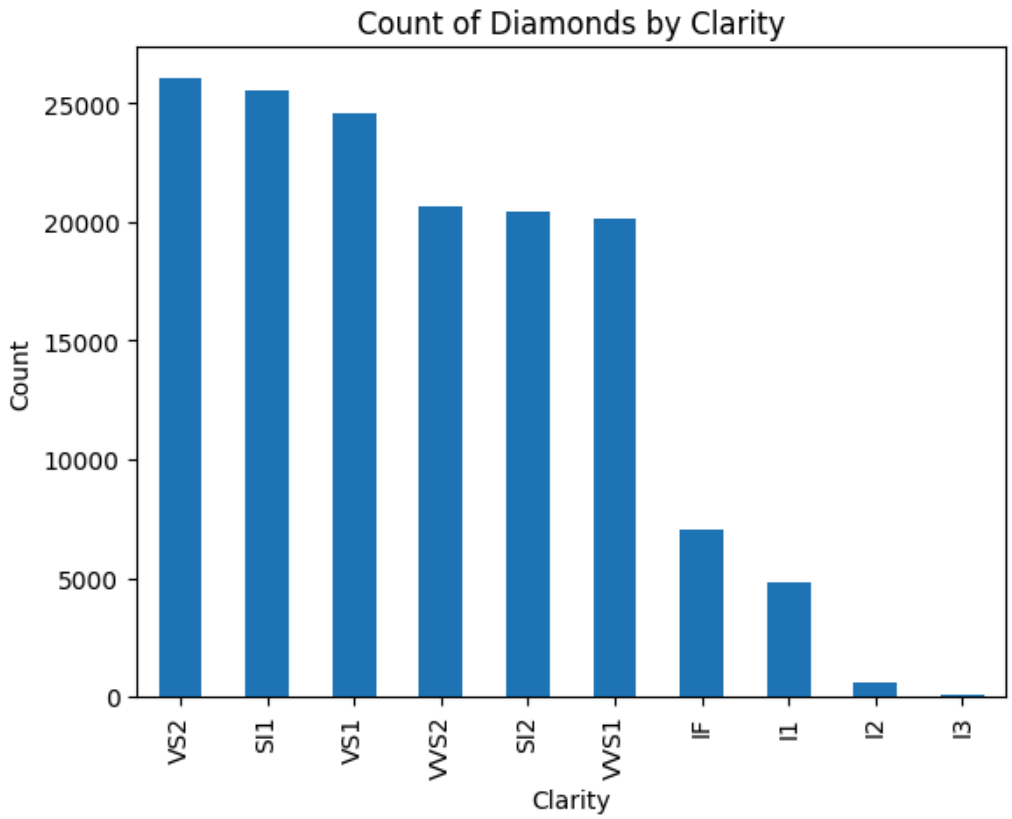


A notable observation is that for each category, regardless of its specific values, there are a significant number of outliers. For instance, **girdle_min** consistently exhibits many outliers across all its values, indicating that these categorical features alone are not strong predictors of **price**. For example, even if a diamond has an **"Excellent"** polish rating, it does not necessarily lead to a narrower price range. However, some specific features and values show a stronger connection to **price fluctuations**. In particular, for **girdle_max**, the categories **"STN"** and **"XTK"**, and for **girdle_min**, the categories **"VTK"** and **"XTK"**, appear to have a more distinct impact. Diamonds falling under these specific girdle classifications—such as those with an **extra thick max girdle**—tend to be valued lower and exhibit fewer outliers in terms of price distribution.

Question 1.4


Count of Diamonds by Color
Count
0
5000
10000
15000
20000
E
F
D
G
H
I
J
K
L
M
Color


Count of Diamonds by Cut
Count
0
20000
40000
60000
80000
100000
120000
Excellent
Very Good
Cut



Question 2.1

Standardization code is shown in the jupyter notebook.
After standardlization, each numerical feature has 0 mean and 1 variance, as shown below:

```
...   (149871, 7)
      carat           -2.002611e-16
      depth_percent   -6.527401e-15
      table_percent   -5.279610e-16
      length           1.104470e-15
      width           -3.155629e-16
      depth           -1.213703e-16
      price            7.529973e+00
      dtype: float64
      carat            1.000003
      depth_percent    1.000003
      table_percent    1.000003
      length           1.000003
      width            1.000003
      depth            1.000003
      price            0.966629
      dtype: float64
```

Question 2.2

Feature selection can significantly influence model performance by removing irrelevant or weakly correlated features, which can reduce noise during training and help prevent overfitting—thereby often improving generalization and lowering test RMSE. With fewer features, classical models such as Linear Regression, Ridge, Lasso, and tree-based methods tend to train faster, produce simpler solutions, and become more interpretable, especially for linear models where each feature's contribution can be readily understood. However, feature selection may not provide the same benefit for all model types. Deep neural networks, for instance, can inherently "ignore" less informative features by learning appropriate internal weights, particularly when large datasets are available. Similarly, models with built-in regularization (e.g., Lasso or Ridge) already penalize unimportant features, reducing the need for extensive manual feature selection. Nonetheless, most classical models stand to gain the most in terms of reduced overfitting and improved interpretability when strong feature selection techniques are applied.

```
Lowest MI Features:
girdle_min_VTK    0.0
girdle_min_STN    0.0
dtype: float64
```

```
Lowest F-Regression Features:
girdle_min_XTN    0.003495
girdle_max_XTK    0.001731
dtype: float64
```

Question 4.1

The objective function are

Ordinary Least Square : min||Y − Xβ||^2
Lasso : min||Y − Xβ||^2+ λ||β||1

Ridge : min||Y − Xβ||^2+ λ||β||2

We choose 4 features: carat, width ,length, depth based on mutual information between features and label to train the future models. Plus we add a constant variable for the 3 linear regression models.

(1) OLS model:

10-Fold RMSE Scores using OLS: [0.31761172944990873, 0.3174413147041658, 0.3144039204268551, 0.3133308962704636, 0.3147540691239164, 0.3128243830941855, 0.3144247899333311, 0.3116293068520818, 0.31427932758560073, 0.3309650976331355]

Average RMSE using OLS: 0.3162

Average learned params using OLS: {'const': 7.529971288239308, 'carat': 0.6465776605004354, 'width': 0.07196186003889238, 'length': 0.19745116714305166, 'depth': 0.0046951258857344065}

For OLS (no regularization), all four features remain with non-zero learned parameters.

(2) Lasso model:

10-Fold RMSE Scores using Lasso: [0.31779917238822264, 0.3176478237662559, 0.3146392680011652, 0.3136325662440399, 0.31490602219021285, 0.3130864217245631, 0.3140836244138715, 0.31174222878370156, 0.3146656089030573, 0.33022332578790475]
Average RMSE using Lasso: 0.3162
Average learned params using Lasso: {'const': 0.0, 'carat': 0.6509221636438502, 'width': 0.0674376088274194, 'length': 0.18910875297936575, 'depth': 0.0}

For Lasso (L1 regularization) model, only depth feature and constant are zero and other parameters are non-zero. The parameter λ determines the

regularization strength in the model. A higher λ increases the penalty on coefficients, causing more of them to shrink to zero, which enhances sparsity but may also lead to underfitting. Lasso regression naturally promotes sparsity by selecting a solution with fewer nonzero coefficients, making it especially beneficial when many features are expected to be irrelevant or when a more interpretable model is preferred.

(3) Ridge (L2 regularization) model:

10-Fold RMSE Scores using Ridge: [0.317691265090179, 0.3174881344395336, 0.31423020200225416, 0.31316760254348075, 0.3148184847156448, 0.31266418473740865, 0.3146661204669945, 0.3114516361807728, 0.3141292346818035, 0.3312642711266639]

Average RMSE using Ridge: 0.3162

Average learned params using Ridge: {'const': 0.0, 'carat': 0.621904160323146, 'width': 0.07561210803828142, 'length': 0.21823367356219733, 'depth': 0.004874777209553859}
Average r-square score using Ridge: 0.8929908604232134

For Ridge model, all four features still remain non-zero, but the constant feature becomes zero. The regularization strength λ controls the trade-off between accurately fitting the training data and constraining the size of the coefficients. A larger λ applies stronger shrinkage, reducing variance but potentially increasing bias. Ridge regression is especially useful when the dataset has more parameters than observations, as it helps stabilize the model by preventing overfitting.

Question 4.2

For Lasso regression, I chose to use LassoCV by 10-fold cross validation on using np.logspace(-2, 2, 100) where choose 100 candidates from 0.01 to 100 so that we could find the optimal alpha. And last we used R squared as the score to decide to use 0.01 since it has the highest R squared score.

For Ridge regression, I chose to use RidgeCV by 10-fold cross validation on using np.logspace(-2, 2, 100) where choose 100 candidates from 0.01 to 100 so that we could find the optimal alpha and also use the R squared score as the metrics to find the optimal alpha. Since applying alpha = 100 has the best score.

Since the RMSE of Lasso and Ridge are the same, we chose to compare their R squared score.

Average r-square score using Lasso: 0.8929363100560297
Average r-square score using Ridge: 0.8929908604232134

R-square score of Ridge has a larger number which means it has a better performance. Therefore, we chose to use Ridge as our best model.

Question 4.3

In Ridge regression, standardizing features is important for improving model performance. This is because features with larger numerical scales can lead to disproportionately high coefficient values, which may overly influence the regularization term. But in our experiment, the standardization does not provide an obvious improvement to the model performance. One of the possible reasons is that our data was transformed by log to deal with the high skewness. The RMSE and R-squared score does not change too much.

Average RMSE using Ridge: 0.3162
Average r-square score using Ridge: 0.8929908604232134
Average RMSE using Ridge without standardization: 0.3153
Average r-squre score using Ridge without standardization: 0.8756571246011909

Question 4.4

OLS Regression Results

Dep. Variable:	price	R-squared:	0.895
Model:	OLS	Adj. R-squared:	0.895
Method:	Least Squares	F-statistic:	2.861e+05
Date:	Tue, 11 Mar 2025	Prob (F-statistic):	0.00
Time:	02:05:33	Log-Likelihood:	-35019.
No. Observations:	134884	AIC:	7.005e+04
Df Residuals:	134879	BIC:	7.010e+04
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	7.5297	0.001	8815.258	0.000	7.528	7.531
carat	0.4662	0.011	43.449	0.000	0.445	0.487
width	0.0531	0.003	17.922	0.000	0.047	0.059
length	0.3967	0.011	36.171	0.000	0.375	0.418
depth	0.0052	0.001	5.648	0.000	0.003	0.007

Omnibus:	6321.217	Durbin-Watson:	1.146
Prob(Omnibus):	0.000	Jarque-Bera (JB):	11232.968
Skew:	-0.378	Prob(JB):	0.00
Kurtosis:	4.195	Cond. No.	31.4

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

P-values for each feature:
const 0.000000e+00
carat 0.000000e+00
width 9.644068e-72
length 4.159034e-285
depth 1.622369e-08
dtype: float64

In linear regression, p-values indicate whether a feature significantly contributes to predicting the target variable (price). A small p-value (≤ 0.05) suggests that the feature's effect is statistically significant and unlikely to be due to chance, whereas a large p-value (which is larger than 0.05) suggests that the feature may not meaningfully impact the target. Based on the given OLS regression results, all four features including carat, width, length, depth are almost zero which mean they are strong relationship with the label price. Given these insights, the model would benefit from these four features.

Question 5.1

The most salient features are:

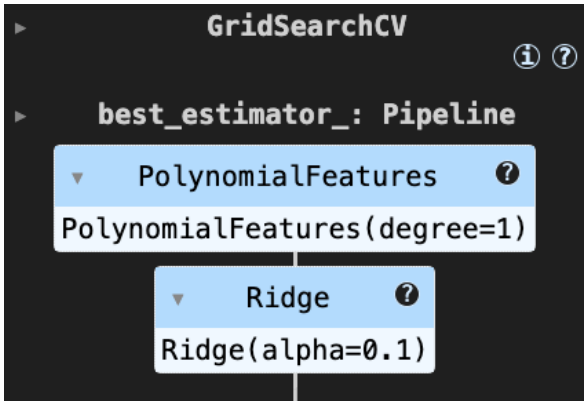
1. carat: 0.6579
2. width: 0.1782
3. length: 0.0796
4. depth: 0.0046

The reason is they are the features that have high correlation (mutual information) with the target variable.

Question 5.2

Using GridSearchCV, I tested for:
polynomial_degree: [1, 2, 3, 4, 5, 6] and ridge_alpha: [0.1, 1, 10]

The best combination of parameters is poly_degree=1 and ridge_alpha=0.1. And its RMSE on Test Set: 0.3172



A higher order polynomial would overfit the training data and thus has poor performance on test data.

Question 6.1

I perform grid search on:

```
param_grid = {
    'hidden_layer_sizes': [(100,), (100, 100), (100, 100, 100)],
    'alpha': [0.001, 0.01]
}
```

The combination of parameters that has the best performance is:

```
Best Parameters: {'alpha': 0.01, 'hidden_layer_sizes': (100, 100, 100)}
RMSE on Test Set: 0.1568
```

Question 6.2

The performance of neural network(RMSE: 0.156) is generally better than that of linear regression(0.317).
The reason is neural network can capture complex non-linear patterns in data.

Question 6.3

We use ReLU activation for the output layer since we are predicting price, which is an non-negative number in [0, inf].

Question 6.4

Increase the network depth too much would make the network too complex and result in overfitting, which will lead to poor test performance.

Question 7.1

Maximum Number of Features

This hyperparameter controls how many features each tree considers when making a split decision.

Performance effects:

Too few features may prevent trees from finding meaningful patterns. Too many features may cause trees to become too similar, reducing ensemble diversity

Regularization effect:

Limiting features is a form of regularization that helps prevent overfitting. By considering only a subset of features at each split, random forests reduce

the chance of individual trees capturing noise in the data. This creates decorrelated trees, which is crucial for the ensemble's ability to generalize

Number of Trees

This determines how many decision trees comprise the random forest.

Performance effects:

Too few trees may lead to unstable predictions. Too many trees increase computational cost without meaningful accuracy gains

Regularization effect:

Adding more trees doesn't cause overfitting, unlike many other model complexity increases

Depth of Each Tree

This controls how many levels deep each decision tree can grow.

Performance effects:

Deeper trees can capture more complex patterns and interactions. Shallow trees may underfit by missing important relationships

Regularization effect:

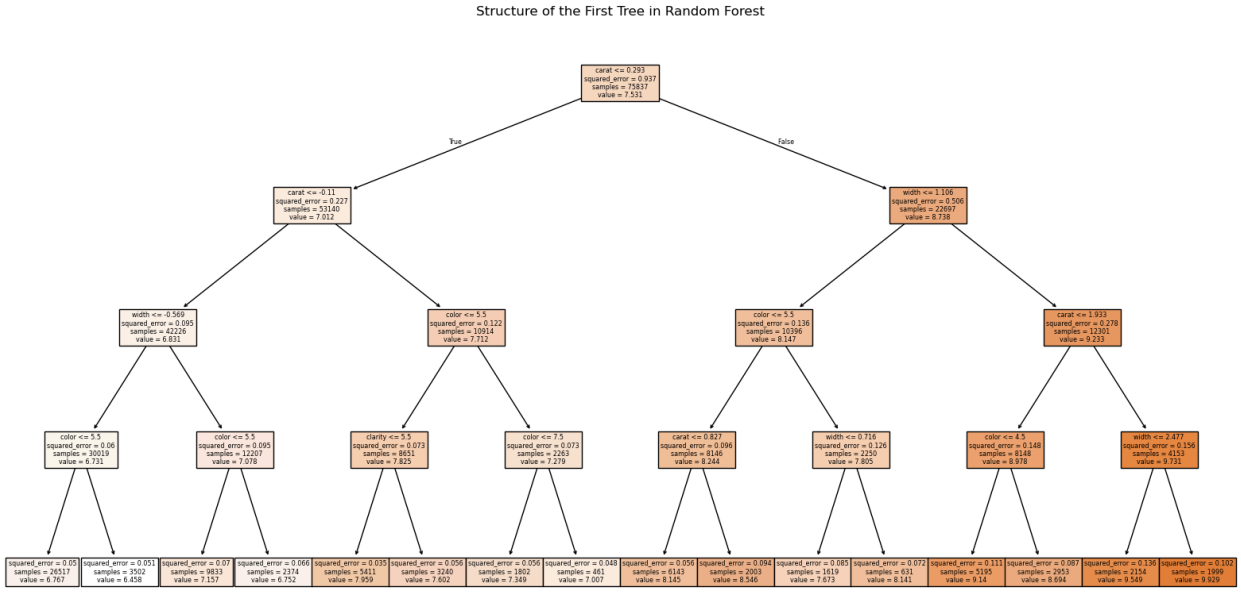
Depth constraints prevent trees from creating leaf nodes with very few samples. This prevents each individual tree from overfitting while preserving the ensemble's overall performance

Question 7.2

Random forests create non-linear decision boundaries by combining many simple threshold splits in a hierarchical manner. While each individual split is just a straight line/plane in the feature space, the recursive application of these splits within each tree creates complex, stair-step shaped regions. When multiple trees with different split patterns (due to feature randomization and bootstrap sampling) are averaged together, their collective "votes" form a probability gradient that produces a smooth, highly non-linear decision boundary. This ensemble effect transforms simple axis-parallel splits into sophisticated boundaries capable of capturing complex relationships in the data.

Question 7.3

Random forest structure:



Feature ‘carat’ is selected for branching at root node.
The importance of ‘carat’ is greater than other features, and it agrees with linear regression.

```
Feature Importances:
               Feature  Importance
25               carat    0.782216
29               width    0.147376
28              length    0.036981
0                color    0.029454
1              clarity    0.003932
30               depth    0.000042
6       girdle_min_STK    0.000000
19       girdle_max_TN    0.000000
2                  cut    0.000000
3               polish    0.000000
```

Question 7.4

OOB Error:
Out-of-Bag (OOB) error is a method used to evaluate random forest models without needing a separate validation dataset. It is calculated by using each tree to predict the samples that were not used in its training (the OOB samples). For each observation in the original dataset, predictions are made only by trees that did not have this observation in their bootstrap sample. These predictions are then aggregated to get the final OOB prediction for each observation, and the error is calculated by comparing with the actual values.

R^2 score, also known as the coefficient of determination, is a statistical measure that represents the proportion of variance in the dependent variable that is predictable from the independent variables.

R^2 ranges from 0 to 1 (or can be negative for badly fitting models):

- $R^2 = 1$: The model explains all the variability in the target variable
- $R^2 = 0$: The model explains none of the variability
- $R^2 < 0$: The model performs worse than a horizontal line

For our experiment, we have:

Out-of-Bag Error: 0.0642
R2 Score: 0.9358

Question 8.1

For lightGBM, 4 most important hyperparameters are:
'num_leaves': (20, 100),
'max_depth': (3, 10),
'learning_rate': (0.01, 0.1, 'log-uniform'),
'n_estimators': (50, 200)

Question 8.2

Best Parameters:
OrderedDict({'learning_rate': 0.037930317378906 57, 'max_depth': 8, 'n_estimators': 156, 'num_leaves': 83})

Best RMSE: 0.1025

Question 8.3

Performance: n_estimators

Number of trees in the ensemble. Higher values improve performance but increase training time and may cause overfitting.

Regularization: lambda_l2

L2 regularization parameter that penalizes large feature weights. Helps prevent overfitting by encouraging simpler models. Higher values (0.1-10) create smoother decision boundaries.

Efficiency: max_bin

Controls how continuous features are discretized into bins. Lower values (63-127) drastically improve speed and reduce memory usage. Higher values (255+) may capture more detail but consume more resources. This parameter is key to LightGBM's computational efficiency.

Part 2

Question 9.1

Processing file: tweets_#nfl.txt
Average number of tweets per hour: 1978.763670383054
Average number of followers of users posting the tweets per tweet: 4662.37544523693
Average number of retweets per tweet: 1.5344602655543254

Processing file: tweets_#superbowl.txt
Average number of tweets per hour: 127827.87722079101
Average number of followers of users posting the tweets per tweet: 8814.96799424623
Average number of retweets per tweet: 2.3911895819207736

Processing file: tweets_#sb49.txt
Average number of tweets per hour: 66125.1947962009
Average number of followers of users posting the tweets per tweet: 10374.160292019487
Average number of retweets per tweet: 2.52713444111402

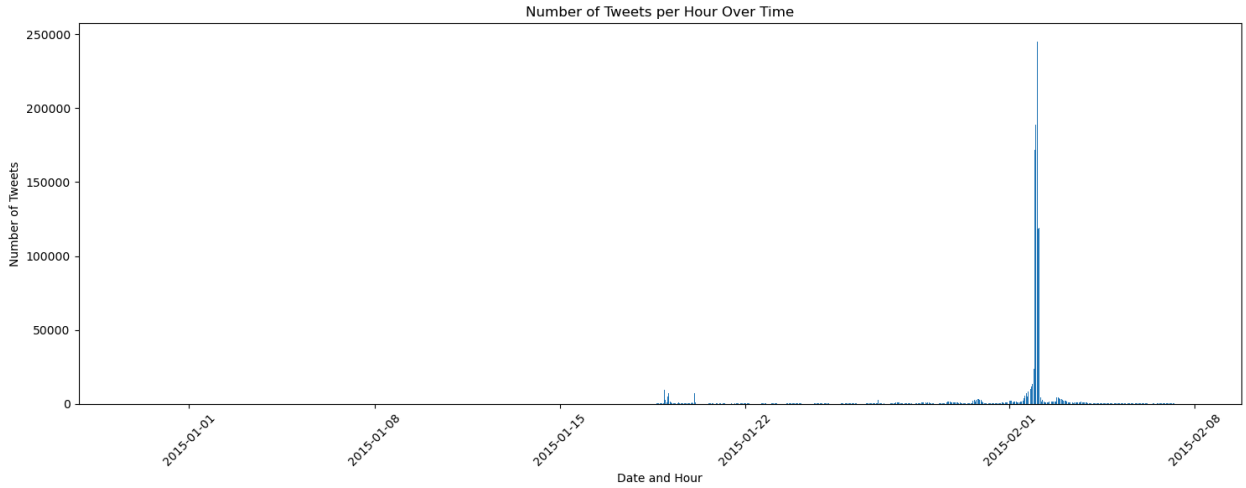
Processing file: tweets_#patriots.txt
Average number of tweets per hour: 21695.875963696693
Average number of followers of users posting the tweets per tweet: 3280.4635616550277
Average number of retweets per tweet: 1.7852871288476946

Processing file: tweets_#gohawks.txt
Average number of tweets per hour: 5249.860313856269
Average number of followers of users posting the tweets per tweet: 2217.9237355281984
Average number of retweets per tweet: 2.0132093991319877

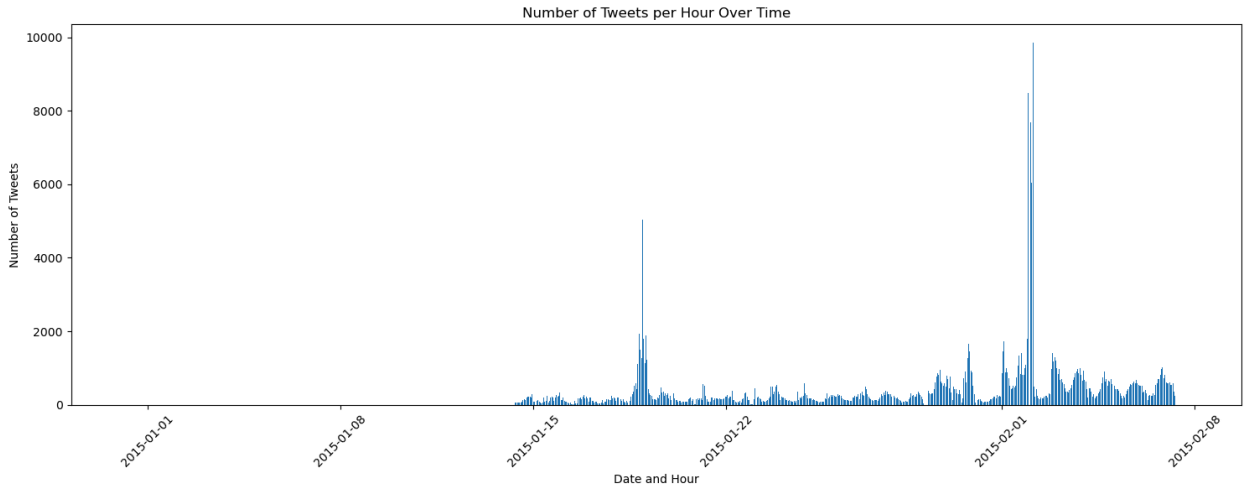
Processing file: tweets_#gopatriots.txt
Average number of tweets per hour: 1869.8754200161627
Average number of followers of users posting the tweets per tweet: 1427.2526051635405
Average number of retweets per tweet: 1.4081919101697078

Question 9.2

“number of tweets in hour” over time for #SuperBowl



“number of tweets in hour” over time for #NFL



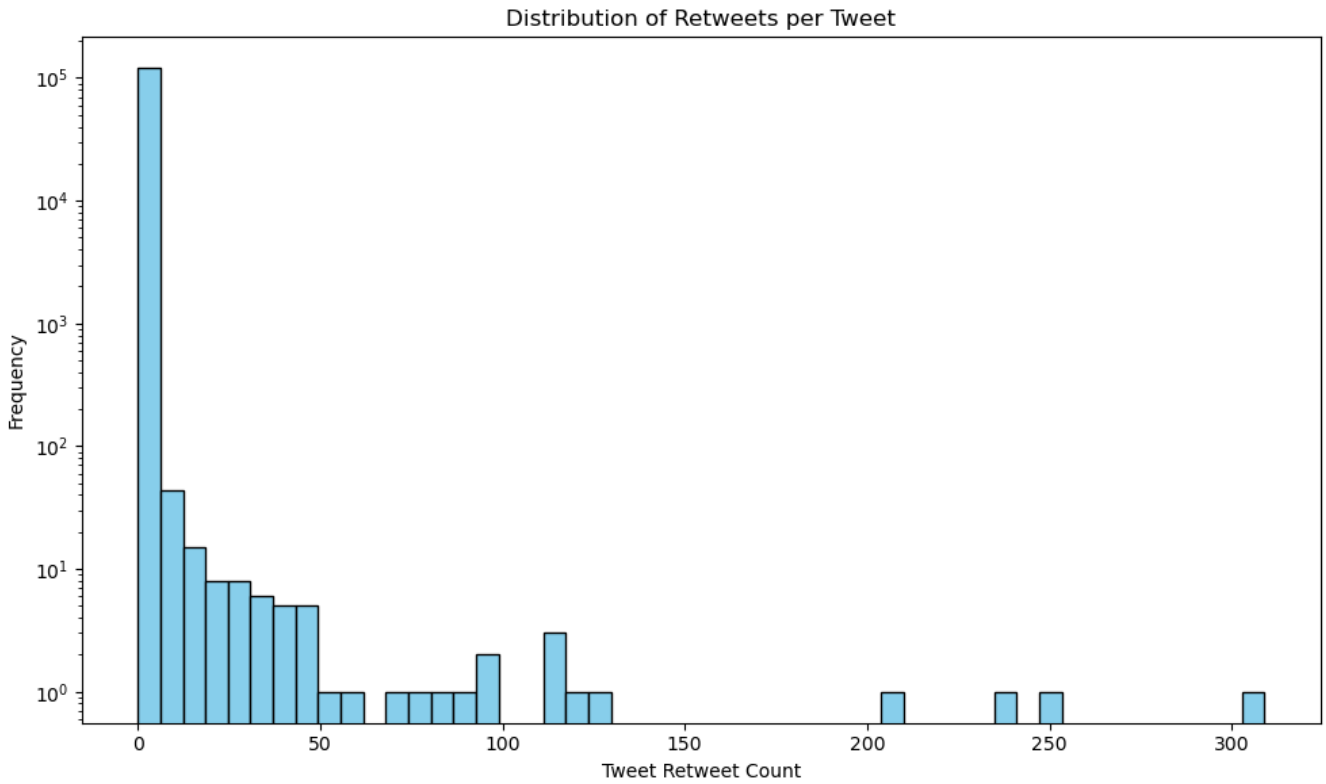
Twitter Engagement Regression Analysis Report

Project Overview

This report summarizes the results of a regression analysis project aimed at predicting tweet engagement metrics (retweet count) from the SuperBowl tweets dataset. The analysis compares several regression models and evaluates their performance in predicting how many retweets a tweet will receive.

Exploratory Data Analysis

1. Target Distribution



Target Variable Distribution

- **Mean**: 0.07 retweets per tweet
- **Median**: 0.00 retweets per tweet (most tweets receive no retweets)
- **Max**: 309.00 retweets (most retweeted tweet)
- **Tweets with at least one retweet**: 3.35% (4,020 tweets)

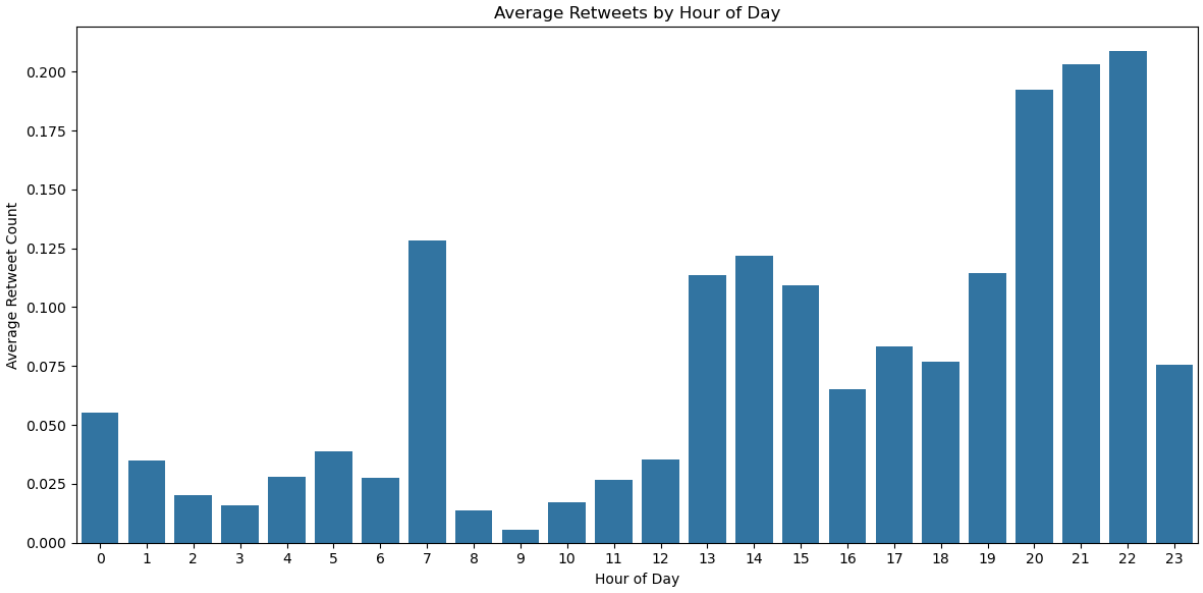
This shows that retweet behavior is highly skewed, with the vast majority of tweets receiving no retweets at all. This imbalance presents a challenge for regression models.

2. Check For Missing Values:

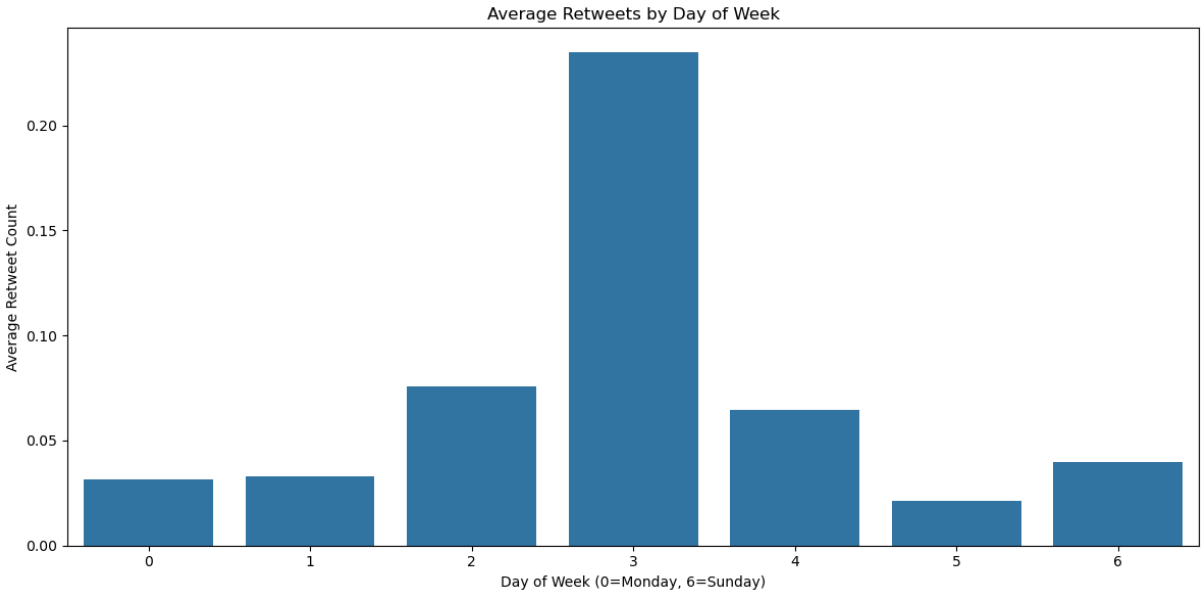
```
Missing values per column:
tweet_contributors                120000
tweet_in_reply_to_status_id       113520
tweet_coordinates                 115427
tweet_timestamp_ms                  4020
tweet_in_reply_to_screen_name     106368
tweet_in_reply_to_user_id         106368
tweet_geo                         115427
tweet_in_reply_to_user_id_str     106368
tweet_in_reply_to_status_id_str   113520
tweet_place                       113842
tweet_extended_entities            98958
author_description                  8912
original_author_description         8844
dtype: int64
```

3. Time-based Analysis

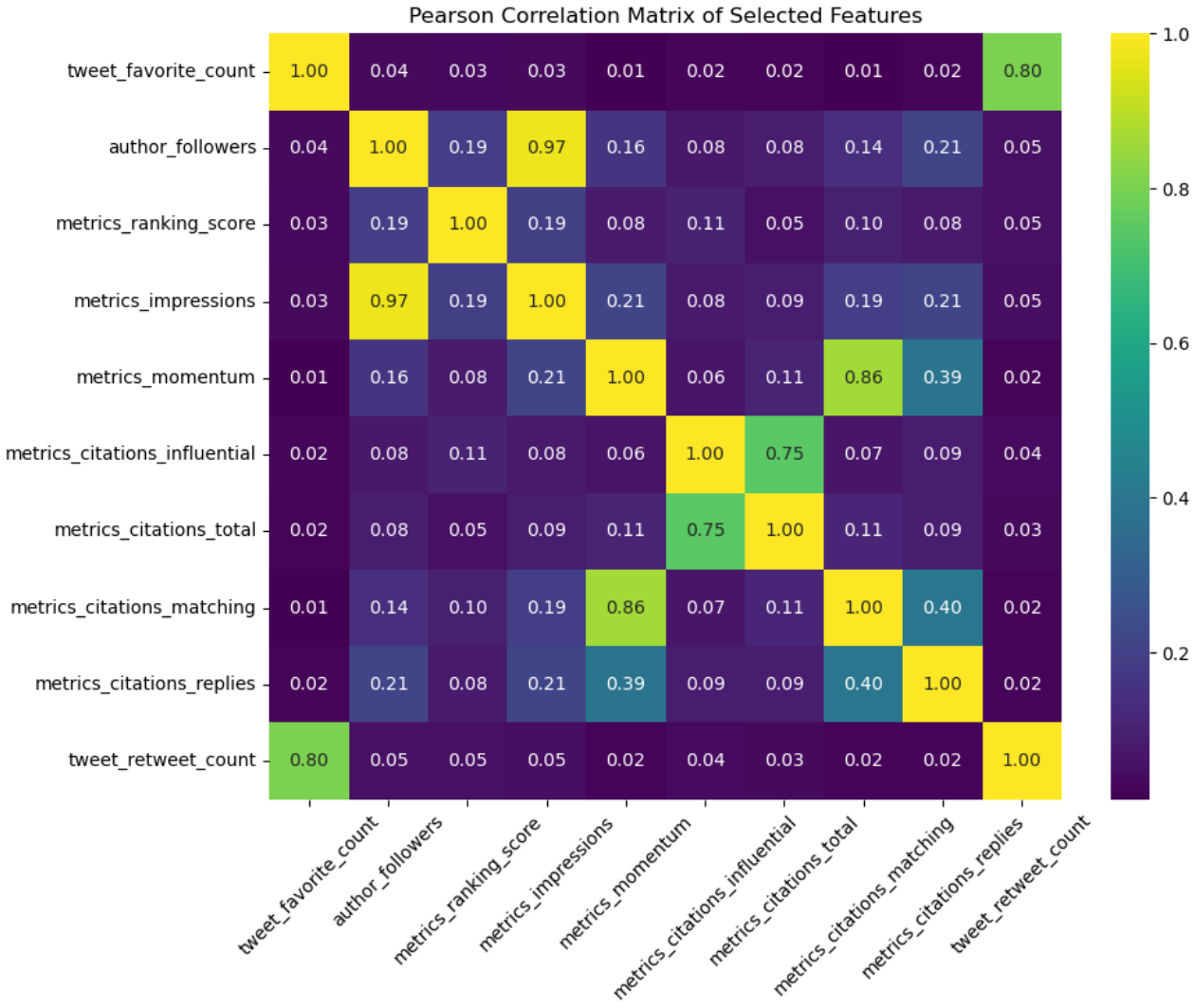
    1. Plot retweets by hour of day



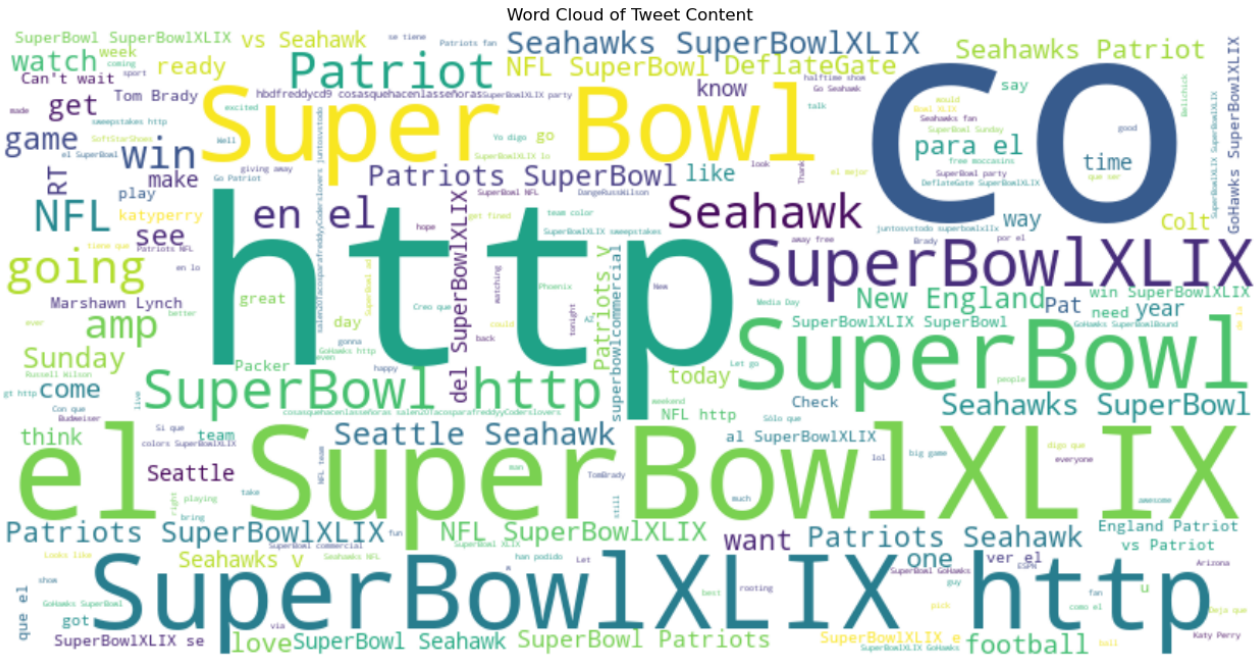
    2. Plot retweets by day of week



4. Pearson Correlation Matrix



6. Content Analysis: Word Cloud



Feature Engineering

The dataset consists of 120,000 tweets related to the SuperBowl, with 38 features extracted through our feature engineering process. These features fall into three main categories:

1. **Text-Based Features**: Sentiment scores, tweet length, readability, hashtag/mention counts, etc.
2. **User-Based Features**: Follower counts, verification status, user description length, etc.
3. **Temporal Features**: Hour of day, day of week, cyclical time encoding, etc.

Model Comparison

We implemented and compared six different regression models:

1. **Linear Regression (without regularization)**
2. **Lasso Regression (L1 regularization)**
3. **Ridge Regression (L2 regularization)**
4. **Logistic Regression** (for binary classification: retweet vs. no retweet)
5. **XGBoost**
6. **Neural Network** (3-layer architecture)

Regression Performance Metrics

Model	MSE	RMSE	MAE	R^2
Linear Regression	2.8885	1.6995	0.1540	-0.0004
Lasso Regression (α=0.001)	2.8877	1.6993	0.1513	-0.0001
Ridge Regression (α=100.0)	2.8884	1.6995	0.1539	-0.0003
XGBoost	2.8805	1.6972	0.1353	0.0024
Neural Network (3 layers)	2.8751	1.6956	0.2101	0.0043

Binary Classification Performance (Logistic Regression)

- **Accuracy**: 96.72%
- **Precision (Class 1)**: 50.00%
- **Recall (Class 1)**: 0.00%
- **F1-Score (Class 1)**: 0.01%

Key Findings

1. **Overall Model Performance**:
    - All regression models performed similarly with R^2 scores close to zero, indicating that predicting the exact number of retweets is extremely challenging

- The Neural Network and XGBoost slightly outperformed linear models
- The high accuracy but near-zero recall for the positive class in logistic regression shows the model is primarily predicting "no retweets" due to class imbalance

2. **Most Important Features**:

   For Linear Models:

   - Author follower count
   - Time-related features (day cosine, hour cosine, day of week)
   - Sentiment compound score
   - URL count

   For XGBoost:

   - Tweet character count
   - Sentiment compound score
   - Author description length
   - Readability score
   - Tweet word count

3. **Feature Importance Patterns**:
   - User influence metrics (followers) consistently appear as important
   - Content characteristics (length, sentiment) are strong predictors
   - Temporal features show moderate importance across models

Challenges and Limitations

1. **Extreme Class Imbalance**: With only 3.35% of tweets receiving any retweets, models struggle to identify patterns for successful tweets
2. **Non-linear Relationships**: Standard linear models fail to capture complex non-linear relationships between features and retweet counts
3. **Limited Predictive Power**: Even the best models achieved R^2 scores close to zero, suggesting either:
   - Missing important predictive features
   - Random/viral nature of retweet behavior that's inherently unpredictable
   - Need for more sophisticated modeling approaches
4. **Binary Classification Approach**: The logistic regression model achieved high accuracy but failed to identify any positive cases, making it ineffective despite its seemingly good performance

Conclusions

1. Predicting the exact number of retweets for tweets is extremely challenging, as evidenced by the low R^2 scores across all models.

2. The Neural Network and XGBoost models marginally outperformed linear models, suggesting there are non-linear relationships in the data that more complex models can partially capture.
3. The most important predictive features relate to:
    - User influence (follower counts)
    - Tweet content (length, sentiment, readability)
    - Timing (hour of day, day of week)
4. The extreme class imbalance makes it difficult to predict which tweets will be retweeted at all, with most models defaulting to predicting zero retweets.

Future Work

1. **Address Class Imbalance**: Use techniques like oversampling, SMOTE, or weighted loss functions to better handle the imbalanced dataset
2. **Feature Engineering**:
    - Extract more advanced text features using topic modeling or embeddings
    - Incorporate network effects and interaction history
    - Add external context features (trending topics, related events)
3. **Model Improvements**:
    - Try ensemble methods combining multiple model types
    - Implement more sophisticated neural network architectures
    - Consider two-stage models: first classify if a tweet will be retweeted, then predict how many retweets
4. **Alternative Approaches**:
    - Frame the problem as an ordinal classification task with retweet count buckets
    - Use zero-inflated models specifically designed for data with excessive zeros
    - Consider time-series approaches to model viral spread patterns