Large-Scale Data Mining: Models and Algorithms
ECE 219 Winter 2023
Prof. Vwani Roychowdhury
UCLA, Department of ECE

Project 4: Regression Analysis and Define Your Own Task!

Due Mar. 19, 2023 by 11:59 PM

Yaman Yucel - 605704529
Ozgur Bora Gevrek - 505846360
Eray Eren - 006075032

1 Introduction:

Regression analysis is a statistical tool used to examine the relationship between a dependent variable and one or more independent variables. The main goal of regression analysis is to estimate the parameters of a regression model that best fit the observed data and to predict the value of the dependent variable for new values of the independent variables. The analysis provides insights into the strength and direction of the relationship between variables and helps to identify important factors that influence the dependent variable.

2 Datasets

There are two datasets that can be chosen for regression analysis. We have chosen the **diamonds** dataset.

2.1 Dataset 1: Diamond Characteristics

We have chosen the synthetic dataset which consists of 53940 round-cut diamonds. There are 10 features, we have discarded price as the target variable. Then we splitted dataset into 80% for training and used the remaining as a test dataset. In result, training dataset has size 43152x9 and test dataset has size 10788x9, whereas training labels have size 43152x1 and test labels have size 10788x1.

2.2 Dataset 2: Gas Turbine CO and Nox Emmision Data Set

One could also have chosen the emission dataset, however we did not perform regression analysis on the emission dataset.

3 Required Steps

In the following section, we show our results and discussion of diamonds dataset.

3.1 Before Training

It is always crucial to investigate the dataset before processing the dataset. We have found out that the dataset does not contain any na or null value, therefore data imputation is not required. “Cut”,”color” and “clarity” features are categorical variables, therefore they need to be handled differently than other numerical variables.

3.1.1 Handling Categorical Features

Ordinal transformation is performed on “cut”, “color” and “clarity” features since they are ordered in numerical sense. Worst value is encoded as 0 and the Best value is encoded as n - 1. n is the total number of different values that a feature can have.

Following ordering is used:

- Cut: Fair, Good, Very Good, Premium, Ideal (Ideal best and Fair worst)
- Color: J, I, H, G, F, E, D (D best and J worst)
- Clarity: I1,SI2, SI1, VS2, VS1, VVS2, VVS1, IF (IF best and I1 worst)

Although we have explained the preprocessing of numerical features in 3.1.3. We would like to note that at this point, we have **standardized the numerical features including the ordinal transformed categorical features.**

3.1.2 Data Inspection

Q1.1 Describe Correlation Patterns

After standardizing the columns, the Pearson correlation matrix is obtained.

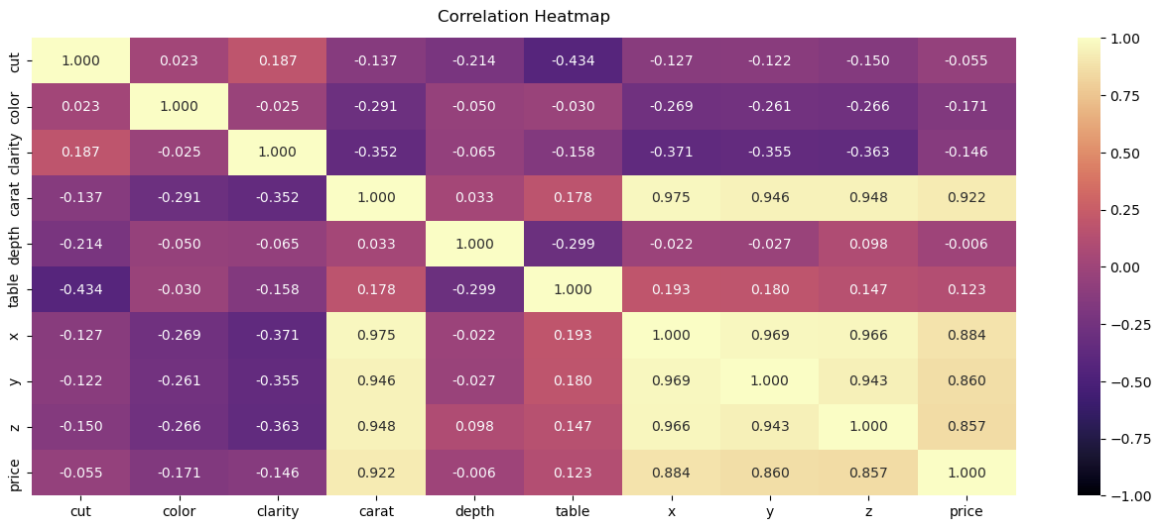


Figure 1: Pearson correlation heatmap

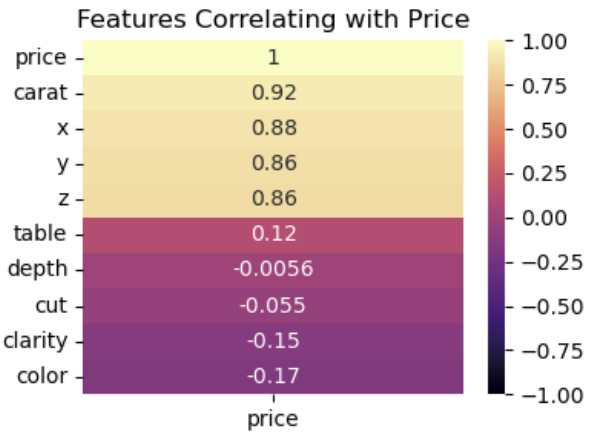


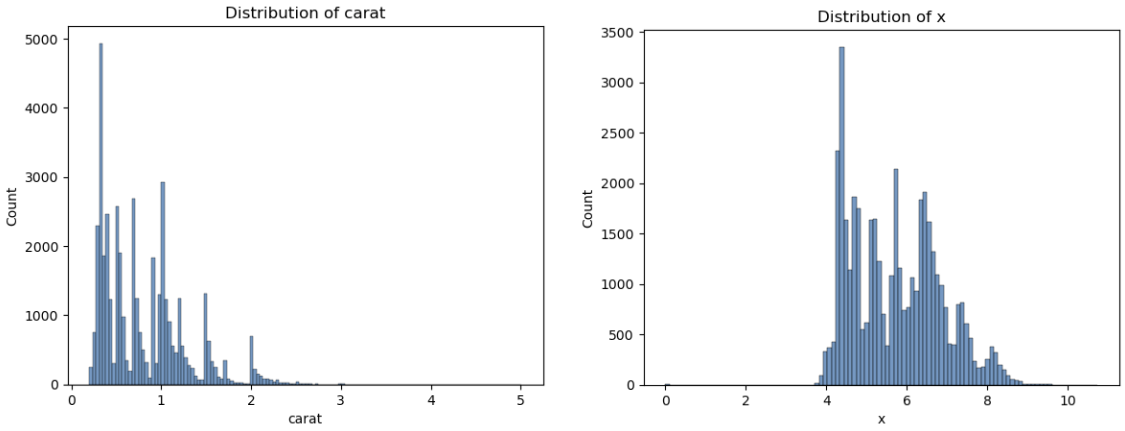
Figure 2: Features Correlating with Price, ordered.

Features that have the highest absolute correlation with the target variable are carat, x, y, z which makes sense since when the size of the diamond gets bigger the price of the diamond is higher.

From Figure 1, we can also easily observe that x,y,z, and carat are highly positively correlated. Diamonds have a specific shape, therefore in order to increase the size of the diamond, one should increase the size, so dimensions match the shape. Carat is related to the volume of the diamond which highly depends on the x, y, z values.

Q1.2 Histogram of numerical features, what preprocessing can be done if the distribution of a feature has high skewness.

Histograms of numerical features:



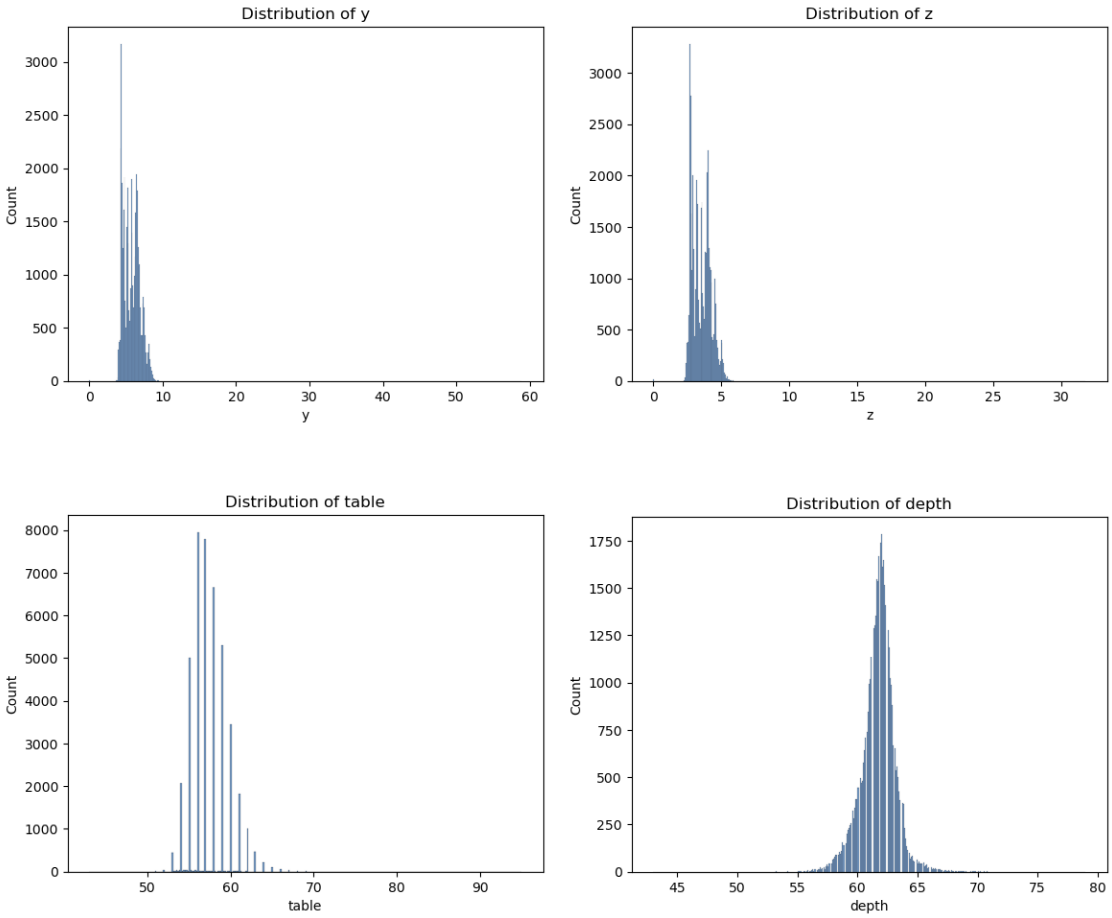


Figure 3: Histograms of numerical features

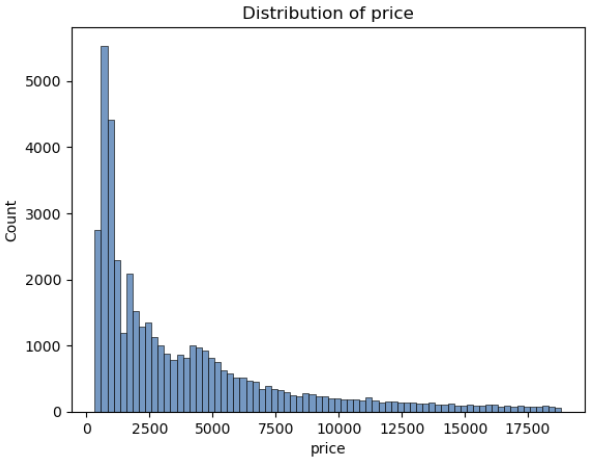


Figure 4: Histogram of target variable

For features that are highly skewed, power transformation can be used to map skewed distributions to gaussian zero mean and unit variance distribution. There are two types of power transformation which are box-cox and yeo-johnson. There are also easier transformations such square root, reciprocal or log transformation of the feature. One can also only use

standardization to achieve great model performance. We have chosen to use only standardization.

We have also plotted power transformed features using the yeo-johnson method.

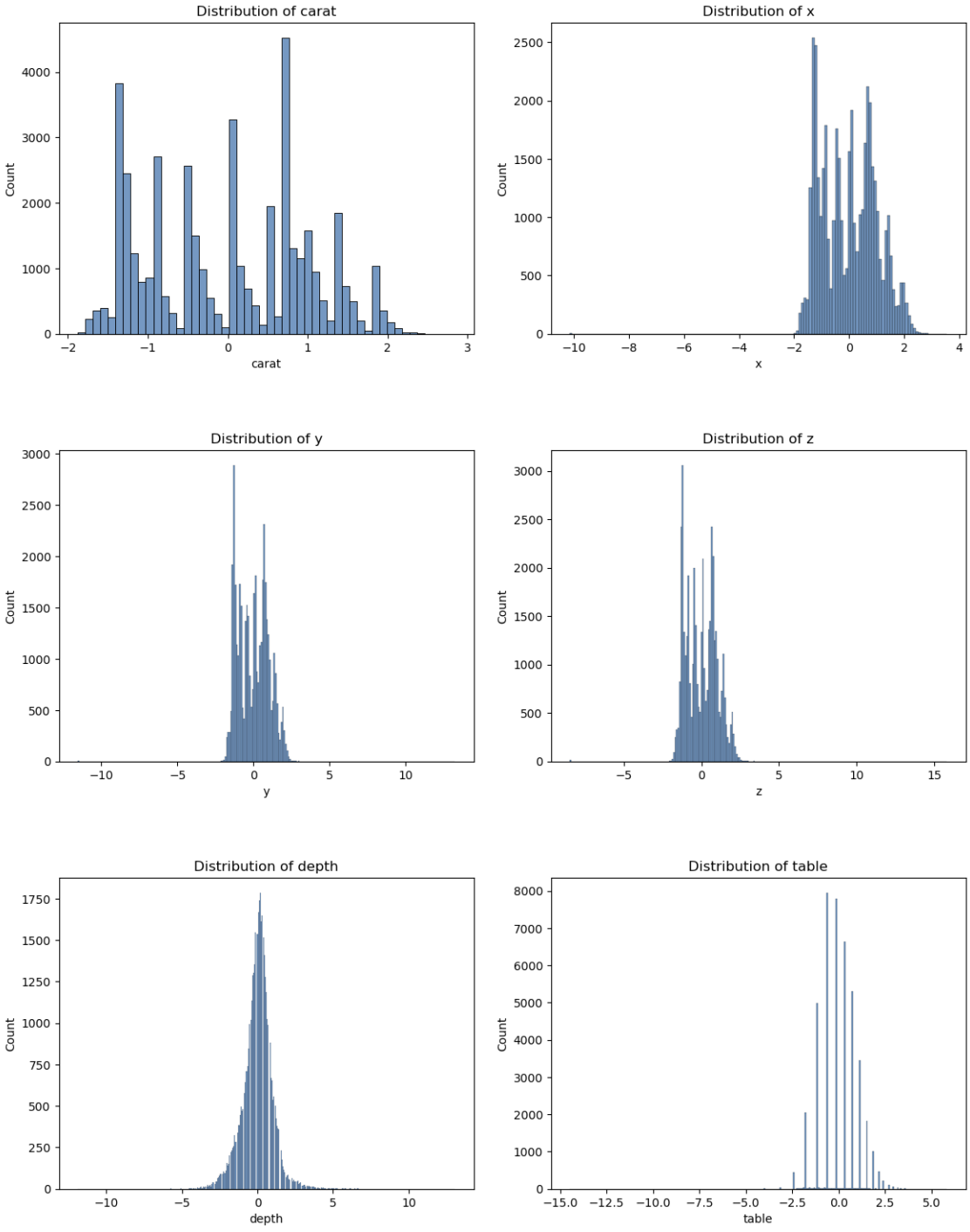


Figure 5: Histograms of power transformed numerical features

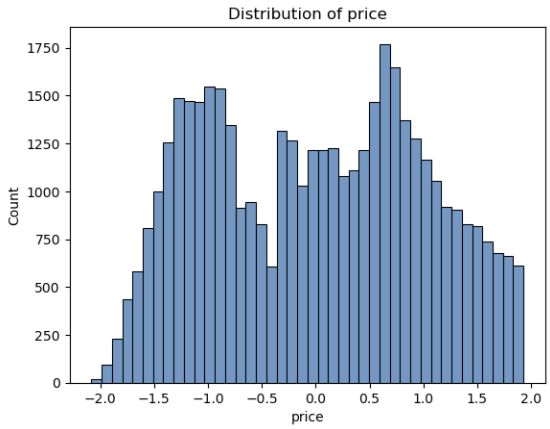


Figure 6: Power transformed target variable histogram

Q1.3 Construct and inspect the box plot of categorical features vs target variable. What do you find?

Box plot of categorical features vs target variable are drawn:

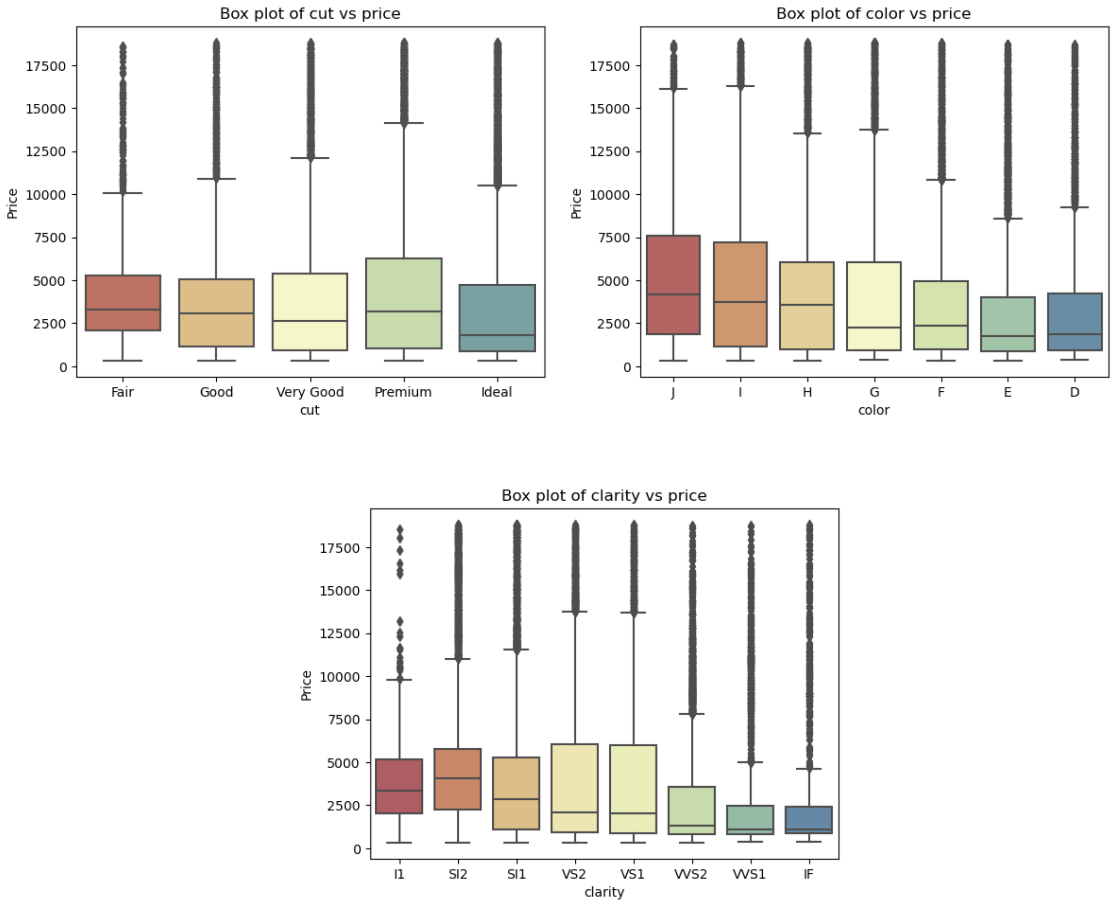


Figure 7: Box plot of categorical features vs target variable

We can observe that as the quality of the cut increases, the price increases. For color, we can see that the price is negatively correlated with the color. For clarity, we can see the price increases between I1 and VS1, however between VS1 and IF price decreases. We have also observed in later parts that categorical features are not effective as numerical features in the regression performance.

Q1.4 Plot the counts by color, cut and clarity

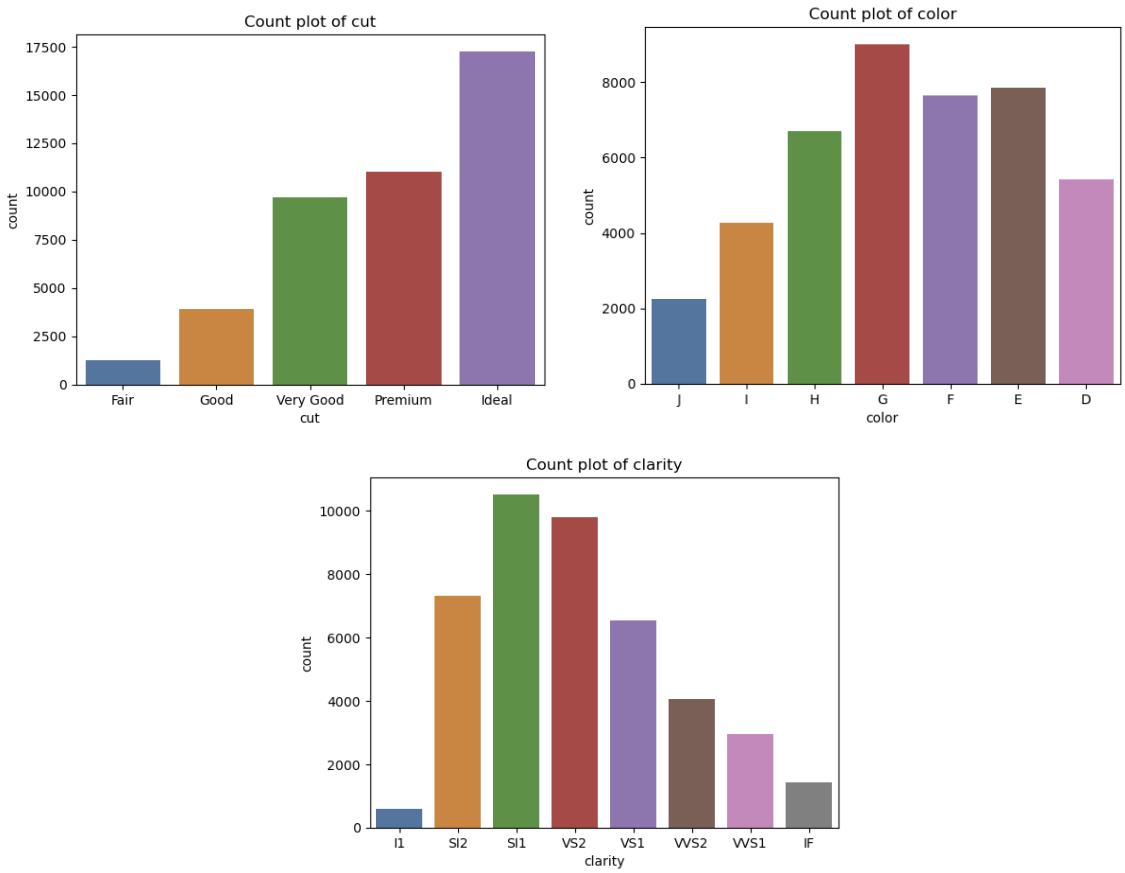


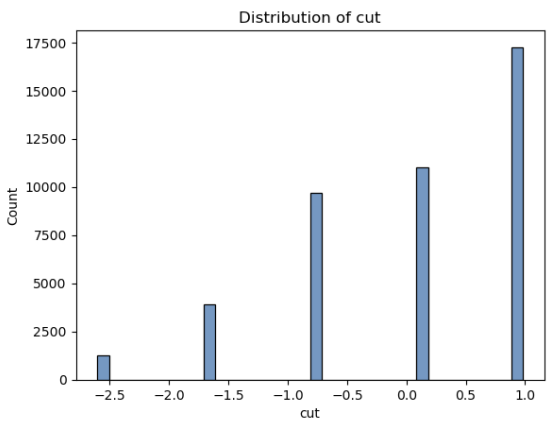
Figure 8 :Counts by color, cut and clarity

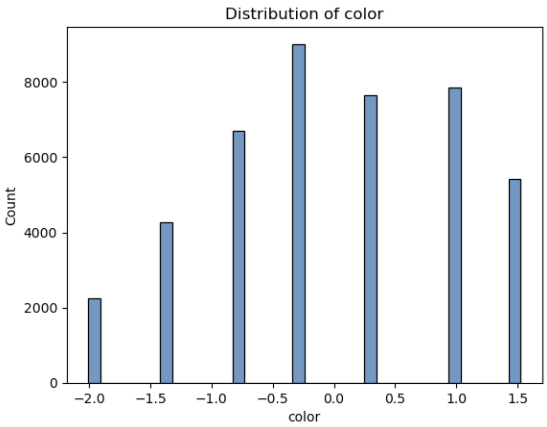
We can observe that samples that have fair cut or I1 clarity or J color are few in number. However, samples that have ideal cut or G color or SI1 clarity are high in number.

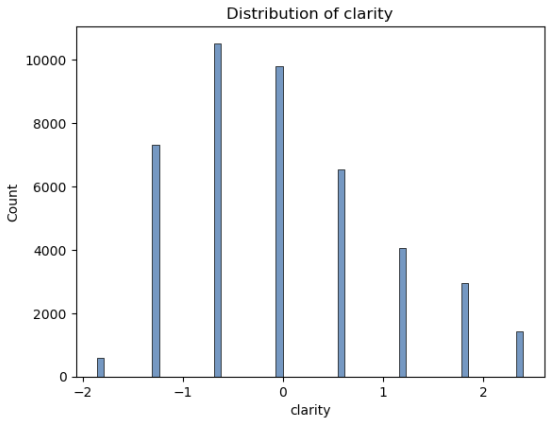
3.1.3 Standardization

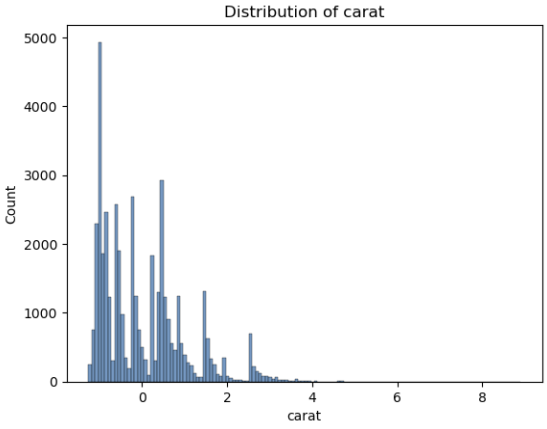
Q2.1 Standardize feature columns and prepare them for training

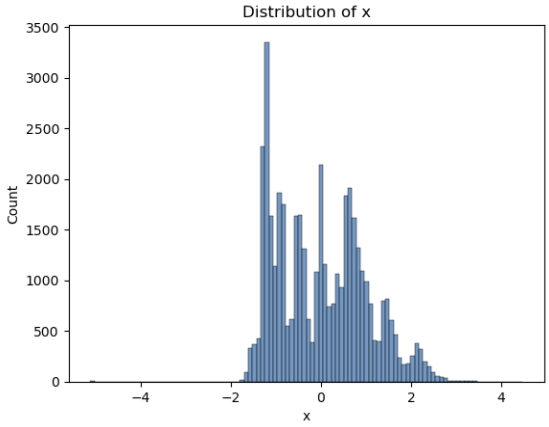
Preprocessing for numerical features should be performed to achieve model robustness and discard feature dominance due to scale. All features are scaled to become Gaussian with zero mean and unit variance. After standardization following histograms are obtained.

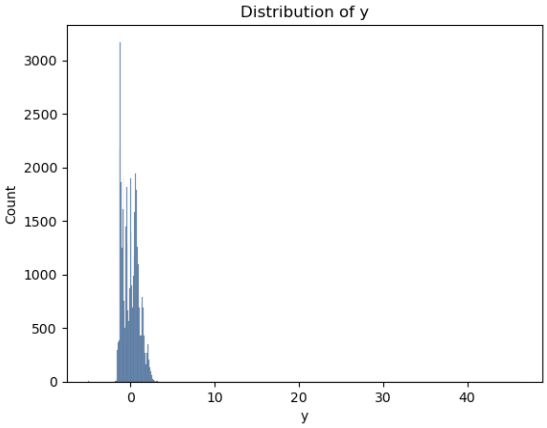

Distribution of cut
Count
cut


Distribution of color
Count
color


Distribution of clarity
Count
clarity


Distribution of carat
Count
carat


Distribution of x
Count
x


Distribution of y
Count
y

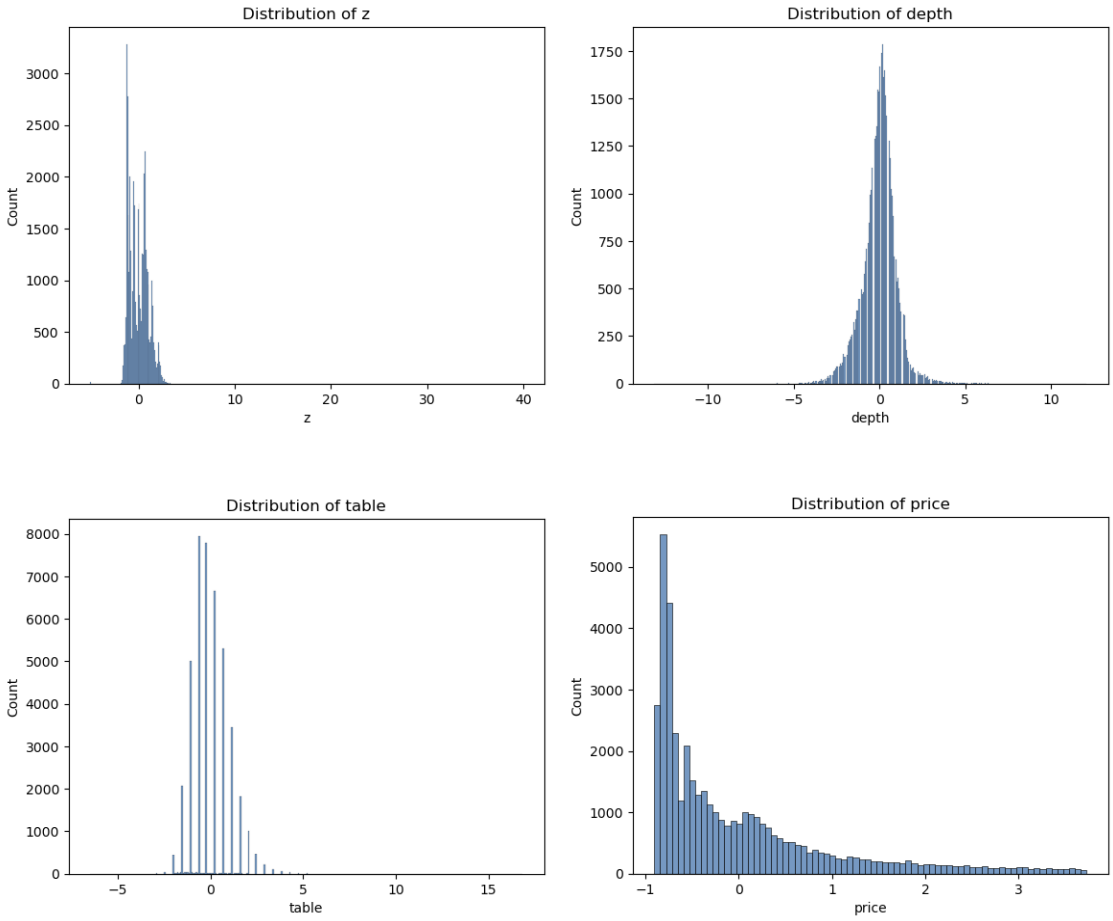


Figure 9: Histograms of standardized features and target (price)

3.1.4 Feature Selection

In this part, we have investigated the feature importance on target variable using two feature selection scoring methods.

1) Mutual Info Regression: MI between two random variables is a non-negative value which measures the dependency between the variables. If two random variables are independent, MI is 0.

2) F Regression: F Regression calculates F scores to show the significance of the improvement of a model when a new variable is added.

Q2.2 Describe how this step qualitatively affects the performance of your models in terms of test RMSE. Is it true for all model types? Also list two features for either dataset that has the lowest MI w.r.t to the target.

Feature(sorted according to MI score)	MI score
carat	1.63383819
y	1.41089569
x	1.39600013
z	1.35315558
clarity	0.21830227
color	0.13829415
cut	0.05790352
depth	**0.03043483**
table	**0.03033969**

Table 1: MI scores for each feature

Feature(sorted according to F score)	F score
carat	2.43413123e+05
x	1.54683575e+05
y	1.22678015e+05
z	1.19280878e+05
color	1.30300353e+03
clarity	9.44679968e+02
table	6.63671853e+02
cut	1.31765000e+02
depth	1.34510791e+00

Table 2: F scores for each feature

Feature	Ranking By F score	Ranking By MI score
carat	1	1
x	2	3
y	3	2
z	4	4
color	5	6
clarity	6	5
table	7	9
cut	8	7
depth	9	8

Table 3: Rankings for each score

We can observe that carat is the most important feature for both methods and depth is the least important feature for both methods.There is not much difference between depth and table in MI score and depth has the lowest score in F regression. For both methods, carat, x, y and z have much higher scores than the rest of the features.

Two features that have the lowest MI w.r.t. to the target are depth and table.

By adding one feature at a time according to ordering, Test RMSE with many models are found for both methods.

OLS results:

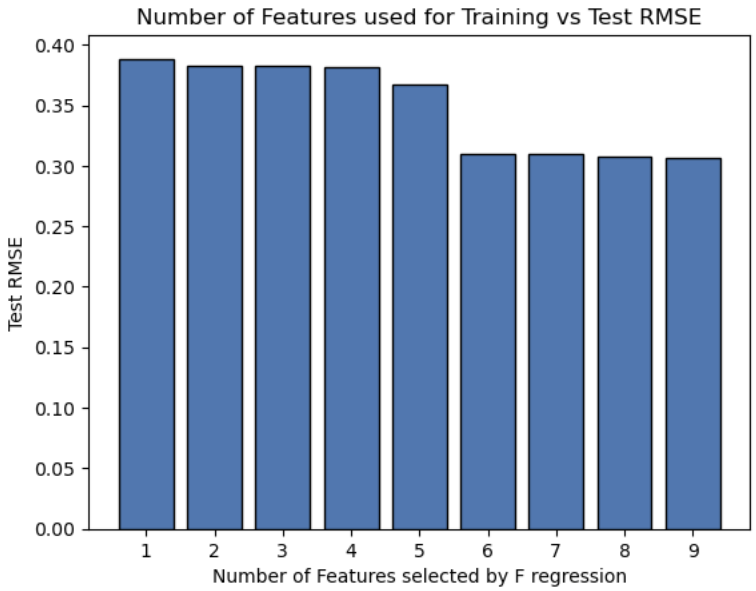


Figure 10: Number of Features selected by F regression and used for Training vs Test RMSE

Figure 11: Number of Features selected by MI and used for Training vs Test RMSE

For the case of linear regression we can see that, when we add features, our test RMSE is getting lower. In general, the general pattern is true for all model types, however the change between steps are different in each model since each model uses features differently to achieve better results. I have repeated the above experiment for lasso regression with bad alpha and optimized alpha, optimized neural network and optimized random forest regressor.

Lasso with alpha = 1e1 (bad alpha):

Figure 12: Lasso with alpha = 1e1 Results

Note that, when alpha is bad, **model performance can not get better or worsen with addition of new features.**

Lasso with alpha = 1e-2 (optimized alpha):

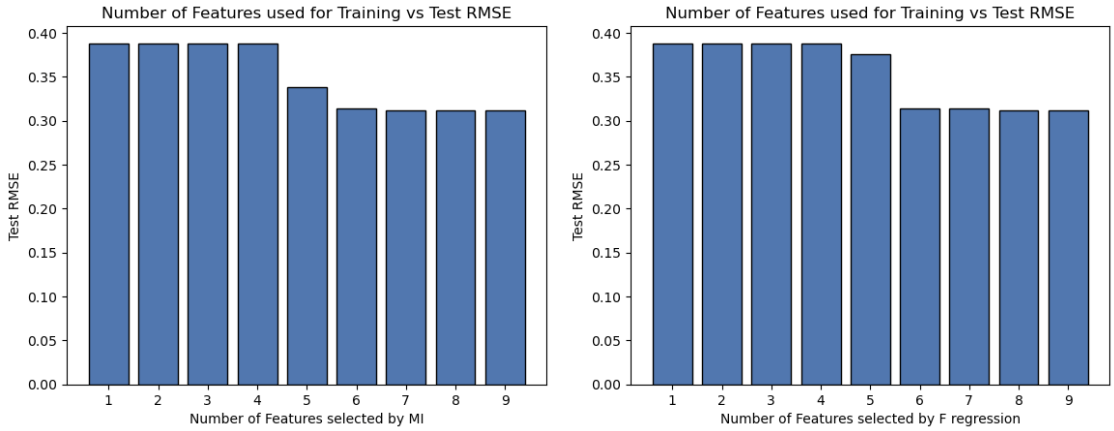


Figure 13: Lasso with alpha = 1-2 Results

Note that, when alpha is optimized, using all features is better than using a single feature. However, using 4 features results in a worse RMSE than using 1 feature. **Therefore, we can not conclude that addition of features results in monotonic decrease in test RMSE.**

Optimized MLPRegressor:

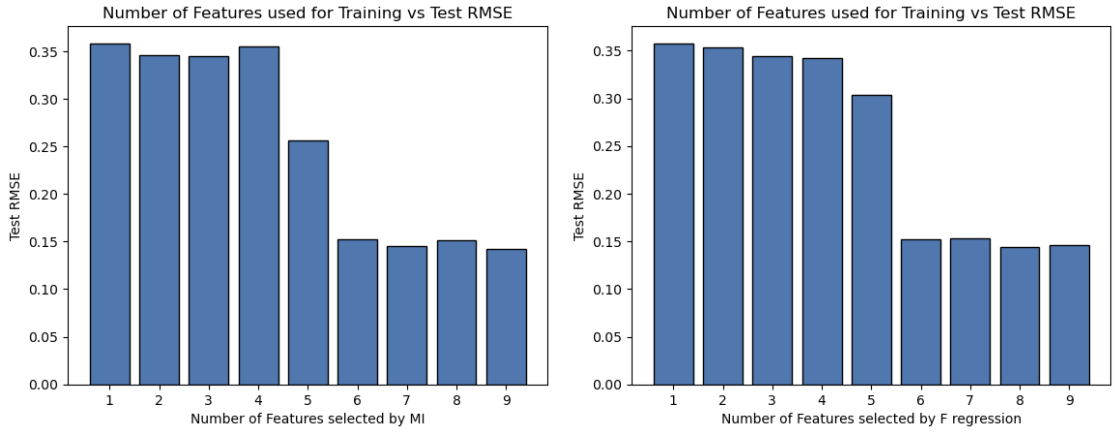


Figure 14: Optimized MLPRegressor Results

Note that, test RMSE does not decrease monotonically especially using 8 features is better than 9 features when features are selected with F regression. **Therefore, using all features does not always give the best test RMSE.** However, using 9 features is better than using a single feature which is true for all models.

RandomForest Regressor:

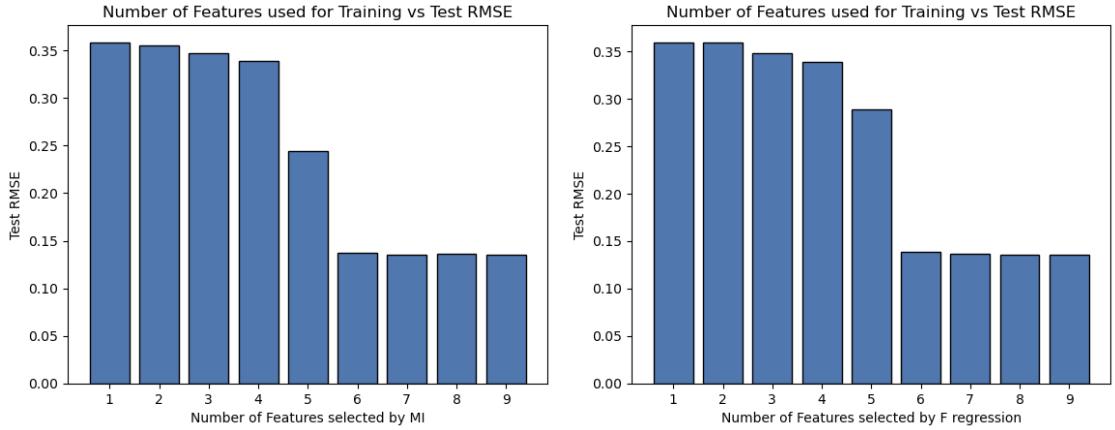


Figure 15: Optimized RandomForest Regressor Results

Same behavior with the OLS model is seen at the RandomForest Regressor, however the test RMSE is much lower than the OLS model. **Therefore, the effect of features for each model is different from each other. It was expected that the pattern will change according to model since some models utilize categorical variables better than numerical variables and some models utilize numerical variables better than categorical variables. Also, the effect of a feature is dependent on the model, meaning that each feature's performance is different for each model.**

In general, we can conclude that using a single feature yields a worse test RMSE, however we can not be sure whether using all features yields the best test RMSE. For most of the models, this holds true, however there are also some models that can achieve better test RMSE with lower feature number. For example, MLPRegressor. **We have decided to use all features.**

3.2 Training

In this part, several models are trained and optimized using 10 fold cross validation. After optimizing the hyperparameters, model's performance is calculated with a test dataset. However, since grid search is trying to find the best parameter set using a greater scoring technique, one can not use RMSE directly since greater scorer is required by grid search function. One needs to use negative RMSE. Also, test results are reported both in R^2 and RMSE. R^2 is explained in the next part and in the random forest part.

3.3 Evaluation

Q3. Explain what OOB error and R^2 score means

OOB:

OOB score is reported in the random forest model part, please check 3.3.4 for the reported result.

In Random Forest Regression, the Out-of-Bag (OOB) score is an estimate of the model's prediction accuracy, which is computed based on the samples that were not included in the model training process.

Random Forest Regression creates multiple decision trees by randomly selecting subsets of the features and observations, and then aggregates their predictions to form a final output. For each decision tree, a random subset of the observations is selected for training, and the remaining samples are used as OOB samples for evaluating the model's performance.

During training, the OOB samples are never used to fit the decision tree. Therefore, for each OOB sample, the model can use the corresponding decision trees to make a prediction and compare it to the true value. By averaging the prediction errors over all the OOB samples, we can obtain an estimate of the model's generalization performance, which is referred to as the OOB score.

The OOB score can be a useful metric for assessing the performance of the model and tuning its parameters. Because it is based on the samples that were not used during training, it provides a more reliable estimate of the model's prediction accuracy on new, unseen data compared to traditional cross-validation methods. OOB score is calculated using the R^2 which is explained next.

R^2

R^2 is a statistical metric commonly used in regression analysis that measures the goodness of fit of a regression model to the data. R^2, also known as the coefficient of determination, ranges from 0 to 1 and represents the proportion of the variance in the dependent variable that is explained by the independent variables included in the model.

An R^2 value of 0 indicates that the model does not explain any of the variability in the dependent variable, while an R^2 value of 1 indicates that the model explains all of the variability in the dependent variable. In general, the higher the R^2 value, the better the model fits the data.

$R^2 = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - \bar{y}_i)^2}$, where $\sum_i (y_i - f_i)^2$ is squared error and the denominator is constant. Therefore R^2 not only brings the model performance metric in a standard scale, but also negatively and perfectly correlates with RMSE. When RMSE is 0, R^2 is 1. When RMSE is huge, if the model performs better than average R^2 is between 0 and 1. When a model performs worse than average, it can get negative values.

3.3.1 Linear Regression

What is the objective function? Train three models (a) OLS, (b) Lasso, (c) Ridge regression.

Objective Function for OLS: $\hat{y}_i$ is the prediction of model for the i th sample

$$L = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \hat{y}_i\right)^2$$

Objective Function for Lasso: $\hat{y}_i$ is the prediction of the model for the i th sample and theta is coefficients for features.

$$L = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \hat{y}_i\right)^2 + \frac{\lambda}{2}||\theta||$$

Objective Function for Ridge: $\hat{y}_i$ is the prediction of the model for the i th sample and theta is coefficients for features.

$$L = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \hat{y}_i\right)^2 + \frac{\lambda}{2}||\theta||^2$$

Q4.1 Explain how each regularization scheme affects the learned parameter set

Lasso makes some coefficients exactly equal to zero, therefore it also performs feature selection. Ridge makes coefficients shrink to zero, but not zero. Lambda hyperparameter determines the number of features to be zero in lasso, and shrinking amount in ridge regression.

We have prepared some results to show the effect of regularization on coefficients.

OLS:

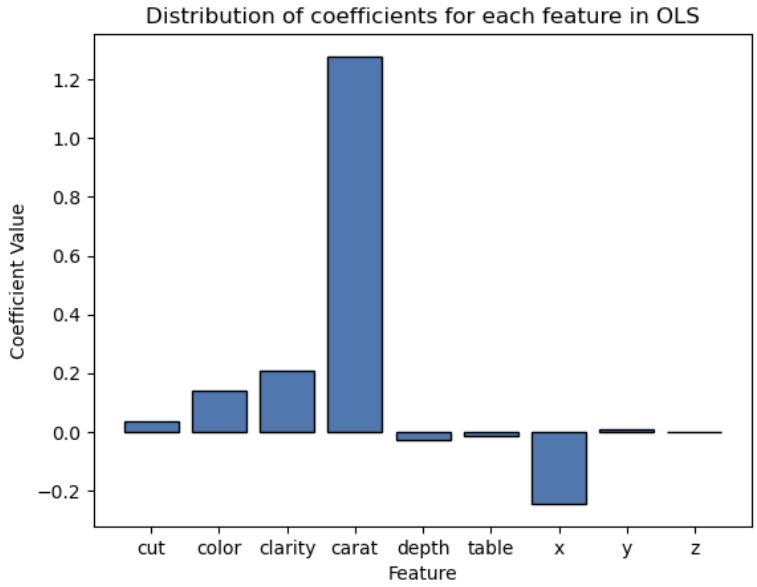


Figure 16: Coefficient values for each feature in OLS

Note that carat has the biggest coefficient indicating that carat is an important feature. We can reach that conclusion since each feature is on the same scale. We also see that y and z are not used in the regression since they are statistically insignificant(which is shown in the p-value part).

Lasso with alpha = 0.001:

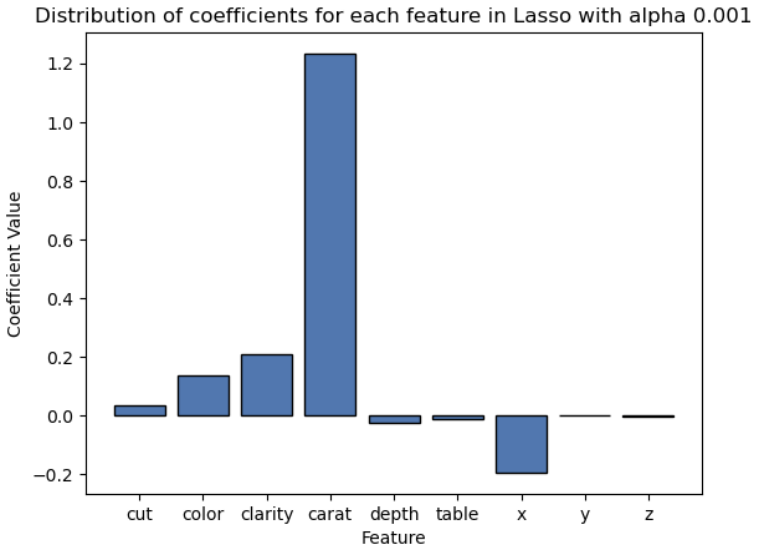


Figure 16: Coefficient values for each feature in Lasso with alpha 0.001

With small alpha, results are very similar to OLS results except coefficient of y is zero.

Lasso with alpha = 0.01:

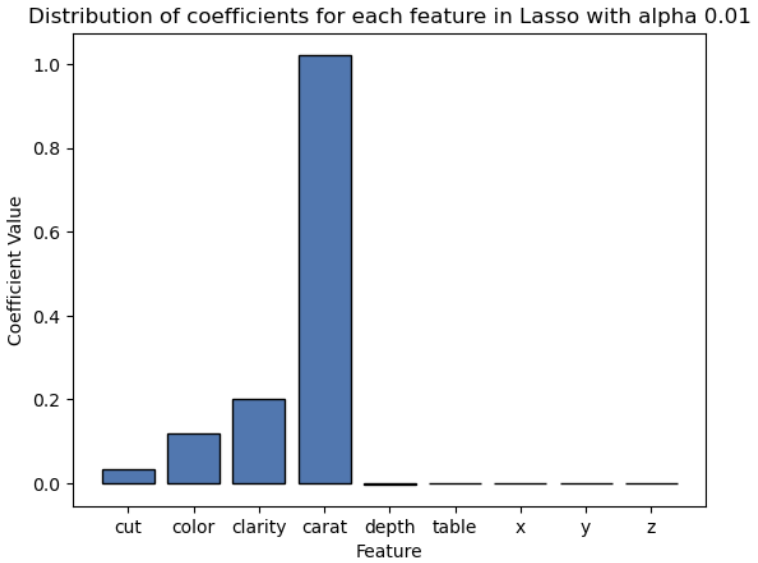


Figure 16: Coefficient values for each feature in Lasso with alpha 0.01

Note that, coefficients of table, x, y and z are zero.

Lasso with alpha = 1:

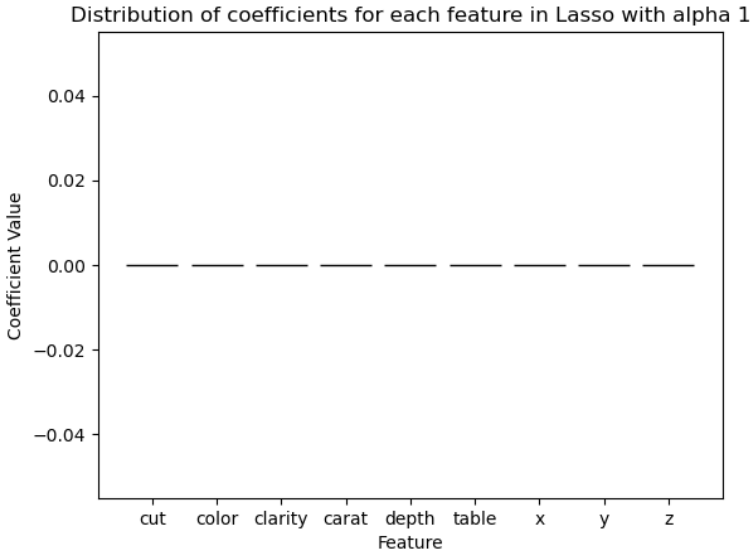


Figure 17: Coefficient values for each feature in Lasso with alpha 1

Note that, all coefficients are zero. Therefore as alpha gets bigger coefficients become zero.

Next, we have investigated the effect of Ridge Regression on coefficients.

Ridge with alpha = 1:

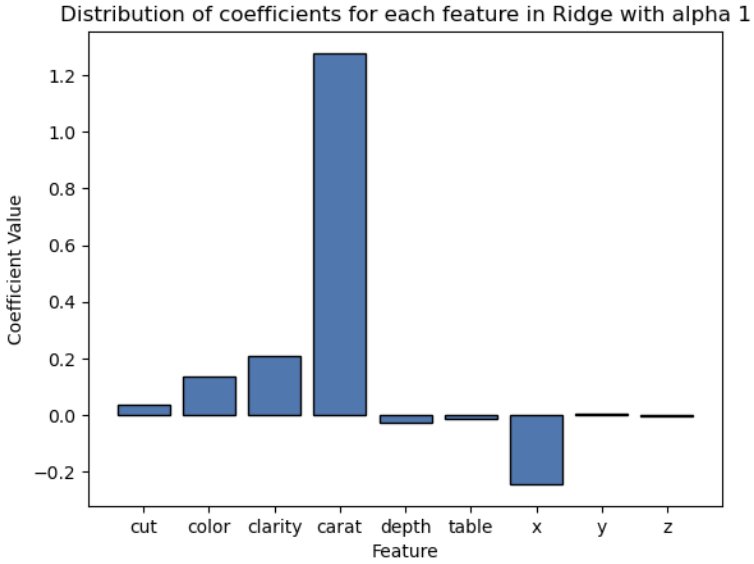


Figure 18: Coefficient values for each feature in Ridge with alpha 1

Note that, coefficients are similar to OLS, and every coefficient is different than zero.

Ridge with alpha = 1e3:

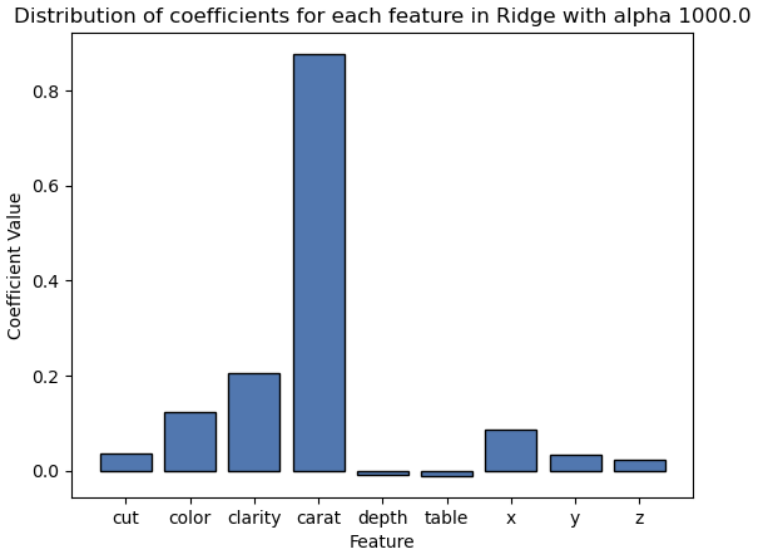


Figure 19: Coefficient values for each feature in Ridge with alpha 1e3

Coefficients got smaller, look at the carat.

Ridge with alpha = 1e12:

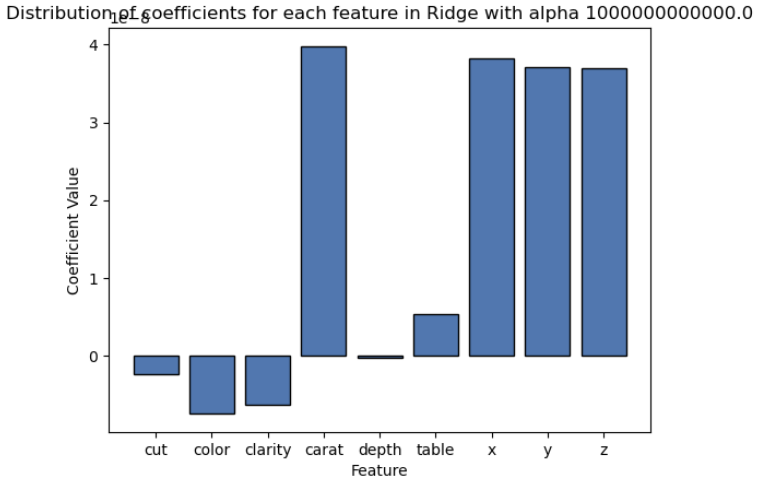


Figure 20: Coefficient values for each feature in Ridge with alpha 1e12

Some coefficients got smaller, look at the carat.

In brief, lasso makes coefficients zero and ridge makes coefficients shrink to zero.

Q4.2 Report your choice of the best regularization scheme along with the optimal penalty parameter and explain how you computed it.

- Best regularization model is Lasso with alpha = 0.0001 with average validation RMSE score = 0.3045837
- Best Ridge model has alpha = 10 and ranked as second with average validation RMSE score = 0.3046295
- The Ordinary Least Square model is ranked as 13th with average validation RMSE score = 0.3046462

Test Results:

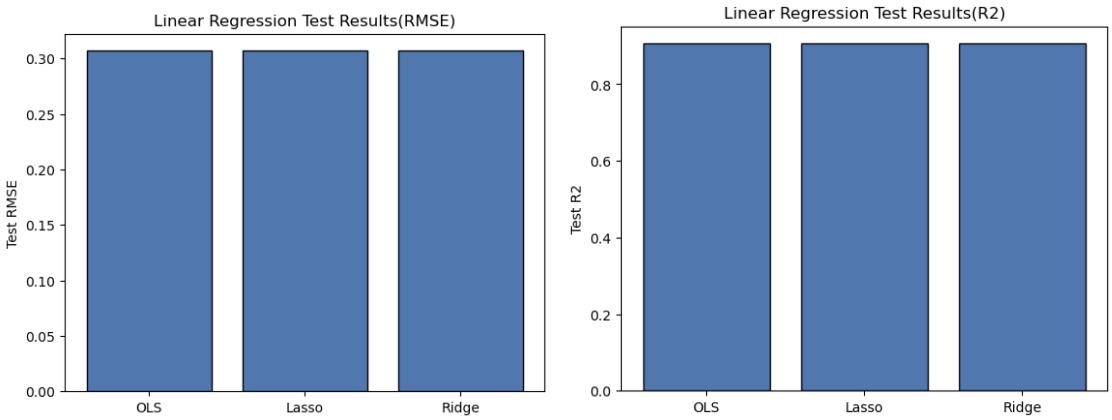


Figure 21: Test Results for Linear Regression best Lasso, best Ridge and OLS

Model	Test RMSE	Test R^2
Lasso(alpha = 0.0001) Ranked 1st	0.3069262427883038	0.9056603356563919
Ridge(alpha = 10) Ranked 2nd	**0.3069078668918593**	**0.905671631686289**
OLS Ranked 13th	0.30692312581679887	0.9056622517685481

Table 4: Linear Regression Results

In order to find the best model with the best hyperparameter we have done grid search for three models. 27 models are cross validated with 10 folds. For scoring, we have used negative RMSE since grid search requires a greater scorer. Test results are obtained with refitting a model with the found parameters and testing with the testing dataset. Note that the Ridge model is slightly better in testing and slightly worse in validation.

Alpha values that are searched for are:
[1e-6,1e-5,1e-4,1e-3,1e-2,1e-1,1,1e1,1e2,1e3,1e4,1e5,1e6

Q4.3 Does feature standardization play a role in improving the model performance(in the cases with ridge regularization)?

In order to test the role of standardization in model performance, we have prepared a dataset that is not standardized and for different values of alpha, cross validation with RMSE and R-squared are performed for each dataset.

OLS results:

Cross validation results for OLS with non-scaled(rmse): 1215.3611477778304
Cross validation results for OLS with non-scaled(r^2): 0.9071118901262294
Cross validation results for OLS with scaled(rmse): 0.30460500670267604
Cross validation results for OLS with scaled(r^2): 0.9071118901262294

Note that RMSE is very different between two datasets because rmse is not standard scaled and has an unit which is the unit of price. Therefore, it is wise to use R-squared to compare model performances. R-squared values are the same in both cases since we do not have a regularizing term and loss function is not affected by the values of coefficients. For example, if we multiply a feature by 2, the coefficient will be halved. Values of coefficients are important since they will have scale in the non-scaled case.

Ridge Results:

In order to see the difference of performance between standardized dataset and non-standardized dataset, cross validation with several alpha values are done and average validation R-squared value is plotted w.r.t alphas.

Alpha values that are used in this experiment are:

[1e-6,1e-5,1e-4,1e-3,1e-2,1e-1,1,1e1,1e2,1e3,1e4,1e5,1e6,1e7,1e8]

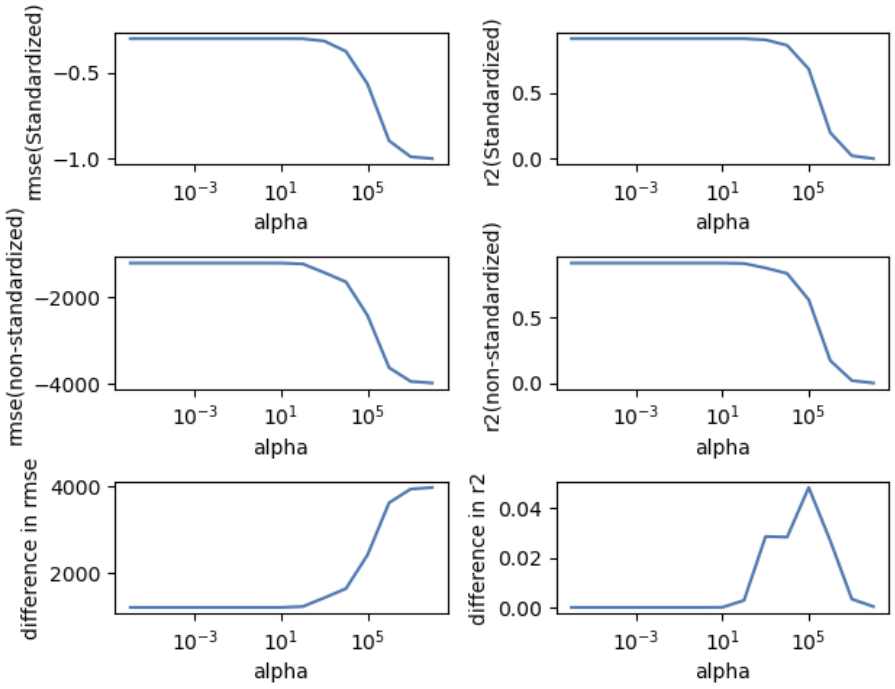


Figure 22: Results of the experiment

When we focus on the subplot at the bottom right Figure 22, we can see that there is a difference in the performance between standardized and non standardized dataset. For small alphas, there is no difference as in the case of OLS results, for very high alphas, both models perform average, therefore there is no difference. Between alpha = 10 and alpha = 1e7, we can see that there is a performance difference. Standardizing the dataset increases the model robustness, especially when the model's loss function is affected by the values of the coefficients. Coefficients can have different units which yield to loss in performance.

Best alpha for Ridge(rmse)(Standardized): 10.0
Best alpha for Ridge(r2)(Standardized): 10.0
Best alpha for Ridge(rmse)(non-Standardized): 1
Best alpha for Ridge(r2)(non-Standardized): 1

Best rmse for Ridge(rmse)(Standardized): 0.30458073386693996
Best r2 for Ridge(r2)(Standardized): 0.9071283394549947
Best rmse for Ridge(rmse)(non-Standardized): 1215.3568312664615
Best r2 for Ridge(r2)(non-Standardized): 0.907113075634086

Lasso Results: (although only ridge is required, we have also provided explanation for lasso).

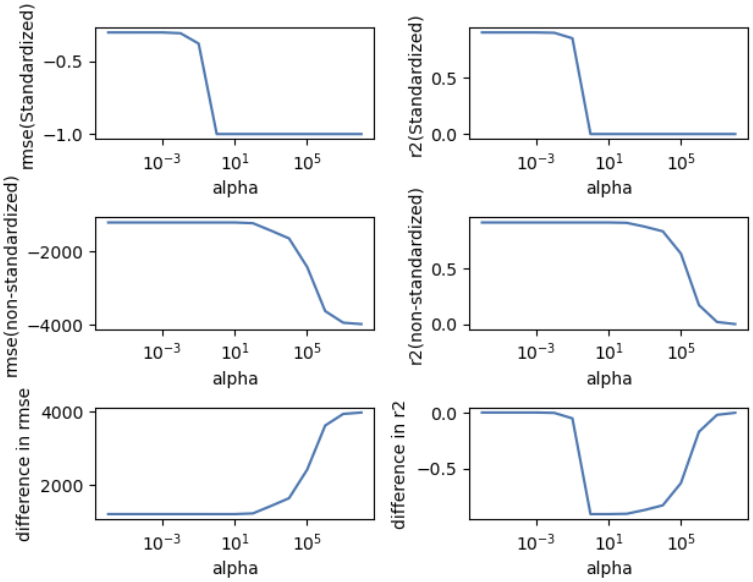


Figure 23: Results of the experiment

When we focus on the subplot at the bottom right Figure 23, we can see that there is a difference in the performance between standardized and non standardized dataset. For small alphas, there is no difference as in the case of OLS results, for very high alphas, both models perform average, therefore there is no difference. Between alpha = 1e-2 and alpha = 1e4, we can see that there is a difference of performance. Feature selection started earlier in the standardized dataset, therefore after alpha = 1e-2, the model selects too many features, resulting in a decrease in the r-squared. However, this phenomenon starts at alpha = 1e4 for non-standardized dataset. This explains the r-square difference between those alpha.

The difference is nearly 1 meaning the model started early to underfit to data in lasso, whereas in the ridge regression the difference was 0.05 indicating that there is a performance loss in non-standardized dataset.

Best alpha for Lasso(rmse)(Standardized): 0.0001
Best alpha for Lasso(r2)(Standardized): 0.0001
Best alpha for Lasso(rmse)(non-Standardized): 1
Best alpha for Lasso(r2)(non-Standardized): 1

Best rmse for Lasso(rmse)(Standardized): 0.3045371804770201
Best r2 for Lasso(r2)(Standardized): 0.9071547552626811
Best rmse for Lasso(rmse)(non-Standardized): 1215.3568312664615
Best r2 for Lasso(r2)(non-Standardized): 0.907113075634086

Q4.4 Some linear regression packages return p-values for different features. What is the meaning of these p-values and how can you infer the most significant features?

OLS Regression Results

Dep. Variable:	y	R-squared:	0.907
Model:	OLS	Adj. R-squared:	0.907
Method:	Least Squares	F-statistic:	4.694e+04
Date:	Sat, 11 Mar 2023	Prob (F-statistic):	0.00
Time:	21:32:33	Log-Likelihood:	-9905.6
No. Observations:	43152	AIC:	1.983e+04
Df Residuals:	43142	BIC:	1.992e+04
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
constant	2.96e-17	0.001	2.02e-14	1.000	-0.003	0.003
cut	0.0346	0.002	19.402	0.000	0.031	0.038
color	0.1379	0.002	88.922	0.000	0.135	0.141
clarity	0.2074	0.002	127.950	0.000	0.204	0.211
carat	1.2786	0.007	186.740	0.000	1.265	1.292
depth	-0.0290	0.002	-15.373	0.000	-0.033	-0.025
table	-0.0151	0.002	-8.207	0.000	-0.019	-0.012
x	-0.2452	0.010	-23.638	0.000	-0.266	-0.225
y	0.0066	0.006	1.101	0.271	-0.005	0.018
z	-0.0028	0.006	-0.433	0.665	-0.015	0.010

Omnibus:	9870.380	Durbin-Watson:	1.991
Prob(Omnibus):	0.000	Jarque-Bera (JB):	506218.328
Skew:	-0.143	Prob(JB):	0.00
Kurtosis:	19.777	Cond. No.	16.9

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

Figure 24: Hypothesis testing for coefficients

Statsmodel library also includes statistical analysis of regression. We have used ordinary least squares for the regression analysis.

In regression analysis, p-values are used to test the statistical significance of the estimated coefficients of the independent variables in the regression model.

The null hypothesis for each coefficient is that it is equal to zero. The corresponding independent variable has no effect on the dependent variable. The alternative hypothesis is that the coefficient is not equal to zero, indicating that the corresponding independent variable has a significant effect on the dependent variable.

The p-value associated with each coefficient represents the probability of observing a coefficient as large or larger than the estimated value, assuming that the null hypothesis is true. A small p-value indicates that the coefficient is statistically significant and that the corresponding independent variable has a significant effect on the dependent variable.

The standard practice is to use a significance level of 0.05, meaning that a p-value less than 0.05 indicates statistical significance. If the p-value is less than the significance level, the null hypothesis is rejected in favor of the alternative hypothesis, and it is concluded that the corresponding independent variable has a significant effect on the dependent variable.

Therefore, since p-values of y and z are 0.271 and 0.665 which are bigger than 0.05, we can conclude that y and z are statistically insignificant. Also, note that we do not need to use constant term for OLS since constant term is very small and p-value is 1. Other features are statistically significant since they have very small p-value(nearly 0). We have done a quick experiment to prove that y and z are useless.

We have trained two OLS regressors, one with all features and one without y,z features.

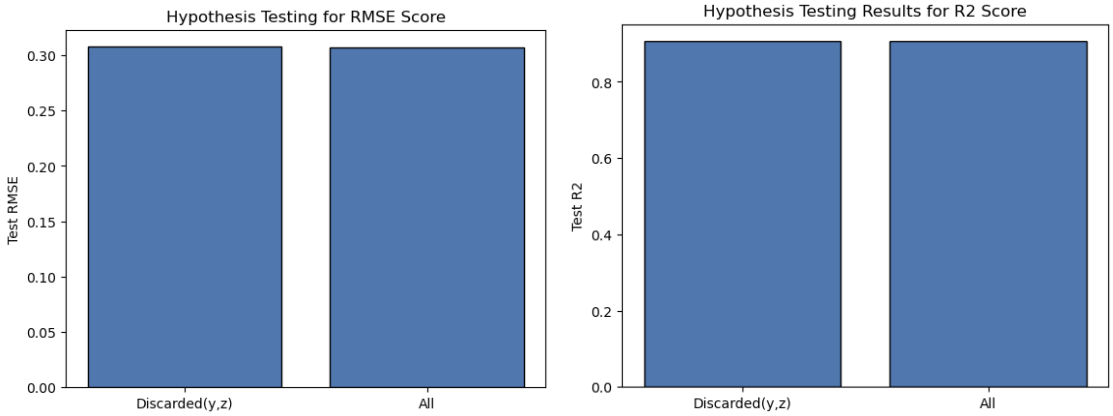


Figure 24: Hypothesis testing for coefficients

RMSE with y,z: 0.30757513955397314
R2 with y,z: 0.9052610122618255
RMSE without y,z: 0.30692312581679887
R2 without y,z: 0.9056622517685481

Note that, RMSE and R-squared values are nearly same for both cases, meaning that y,z does not contribute to regression when other features are available.

3.3.2 Polynomial Regression

Q5.1 What are the most salient features? Why?

We have found out from Q5.2 that using degree = 2, gives the optimal hyperparameter. Therefore, we have constructed new features with degree = 2. After constructing the new data, we have used MI and F-regression methods to score new features. When we use degree = 2, the new feature space size is 54.

F-Score Results:

	cut	color	clarity	carat	depth	table	x	y	z	f_score
36	0	0	0	1	0	0	1	0	0	248978.052771
3	0	0	0	1	0	0	0	0	0	243413.123241
38	0	0	0	1	0	0	0	0	1	236356.451542
34	0	0	0	1	1	0	0	0	0	230977.438230
37	0	0	0	1	0	0	0	1	0	228559.603766
35	0	0	0	1	0	1	0	0	0	225005.529883
48	0	0	0	0	0	0	2	0	0	203784.071083
50	0	0	0	0	0	0	1	0	1	189191.574219
49	0	0	0	0	0	0	1	1	0	181333.434011
33	0	0	0	2	0	0	0	0	0	166862.826563
6	0	0	0	0	0	0	1	0	0	154683.574568
41	0	0	0	0	1	0	1	0	0	146296.718780
52	0	0	0	0	0	0	0	1	1	137998.454162
7	0	0	0	0	0	0	0	1	0	122678.014900
45	0	0	0	0	0	1	1	0	0	122167.520050
42	0	0	0	0	1	0	0	1	0	119621.205793
8	0	0	0	0	0	0	0	0	1	119280.877551
47	0	0	0	0	0	1	0	0	1	106312.339643
46	0	0	0	0	0	1	0	1	0	103777.907546
43	0	0	0	0	1	0	0	0	1	102737.506140
12	1	0	0	1	0	0	0	0	0	56671.982482
27	0	0	1	1	0	0	0	0	0	36854.088997
53	0	0	0	0	0	0	0	0	2	32307.121315
20	0	1	0	1	0	0	0	0	0	20549.135756
51	0	0	0	0	0	0	0	2	0	20084.805754
15	1	0	0	0	0	0	1	0	0	6707.240539
17	1	0	0	0	0	0	0	0	1	6561.632132
16	1	0	0	0	0	0	0	1	0	6522.145942
19	0	1	1	0	0	0	0	0	0	1567.924039
30	0	0	1	0	0	0	1	0	0	1465.703583
31	0	0	1	0	0	0	0	1	0	1420.863779
32	0	0	1	0	0	0	0	0	1	1415.204144
1	0	1	0	0	0	0	0	0	0	1303.003535
21	0	1	0	0	1	0	0	0	0	1302.320823
10	1	1	0	0	0	0	0	0	0	1215.763018
22	0	1	0	0	0	1	0	0	0	1169.803463
18	0	2	0	0	0	0	0	0	0	1137.706508
28	0	0	1	0	1	0	0	0	0	947.741900
2	0	0	1	0	0	0	0	0	0	944.679968
11	1	0	1	0	0	0	0	0	0	883.787781
26	0	0	2	0	0	0	0	0	0	856.348567
29	0	0	1	0	0	1	0	0	0	846.357387
23	0	1	0	0	0	0	1	0	0	790.357626
24	0	1	0	0	0	0	0	1	0	769.124872
25	0	1	0	0	0	0	0	0	1	757.756323
5	0	0	0	0	0	1	0	0	0	663.671853
44	0	0	0	0	0	2	0	0	0	643.077741
40	0	0	0	0	1	1	0	0	0	624.626145
9	2	0	0	0	0	0	0	0	0	189.431223
13	1	0	0	0	1	0	0	0	0	134.950142
0	1	0	0	0	0	0	0	0	0	131.765000
14	1	0	0	0	0	1	0	0	0	82.433580
4	0	0	0	0	1	0	0	0	0	1.345108
39	0	0	0	0	2	0	0	0	0	0.940653

Figure 25: F-score results for new features. Rows represent the new feature, values at rows represent the power of the original feature. For example, feature 36 is carat * x with f score 248979. Also, feature 37 is carat * y with f score 228559.

When we look at the f-scores, we can see that products that contain the carat feature ranks best, after those features, products that contain x ranks best. Also, products that contain the depth or table ranks worst. Therefore we can conclude that the most salient features are the ones that contain carat features in their product.

MI Results:

	cut	color	clarity	carat	depth	table	x	y	z	mi_score
3	0	0	0	1	0	0	0	0	0	1.634221
33	0	0	0	2	0	0	0	0	0	1.630034
37	0	0	0	1	0	0	0	1	0	1.557618
36	0	0	0	1	0	0	1	0	0	1.536699
35	0	0	0	1	0	1	0	0	0	1.489833
38	0	0	0	1	0	0	0	0	1	1.489120
27	0	0	1	1	0	0	0	0	0	1.470276
51	0	0	0	0	0	0	0	2	0	1.412635
52	0	0	0	0	0	0	0	1	1	1.411098
7	0	0	0	0	0	0	0	1	0	1.411085
50	0	0	0	0	0	0	1	0	1	1.402359
34	0	0	0	1	1	0	0	0	0	1.398110
48	0	0	0	0	0	0	2	0	0	1.394405
6	0	0	0	0	0	0	1	0	0	1.394286
49	0	0	0	0	0	0	1	1	0	1.385358
53	0	0	0	0	0	0	0	0	2	1.351801
8	0	0	0	0	0	0	0	0	1	1.351600
20	0	1	0	1	0	0	0	0	0	1.318447
42	0	0	0	0	1	0	0	1	0	1.313917
41	0	0	0	0	1	0	1	0	0	1.307521
12	1	0	0	1	0	0	0	0	0	1.258867
30	0	0	1	0	0	0	1	0	0	1.245584
31	0	0	1	0	0	0	0	1	0	1.235795
32	0	0	1	0	0	0	0	0	1	1.195063
47	0	0	0	0	0	1	0	0	1	1.161407
43	0	0	0	0	1	0	0	0	1	1.125540
45	0	0	0	0	0	1	1	0	0	1.110905
46	0	0	0	0	0	1	0	1	0	1.105612
17	1	0	0	0	0	0	0	0	1	1.034661
15	1	0	0	0	0	0	1	0	0	1.031437
16	1	0	0	0	0	0	0	1	0	1.029631
24	0	1	0	0	0	0	0	1	0	0.907857
23	0	1	0	0	0	0	1	0	0	0.899342
25	0	1	0	0	0	0	0	0	1	0.872678
19	0	1	1	0	0	0	0	0	0	0.426213
29	0	0	1	0	0	1	0	0	0	0.232704
11	1	0	1	0	0	0	0	0	0	0.227274
2	0	0	1	0	0	0	0	0	0	0.214539
26	0	0	2	0	0	0	0	0	0	0.214512
28	0	0	1	0	1	0	0	0	0	0.186846
10	1	1	0	0	0	0	0	0	0	0.158859
22	0	1	0	0	0	1	0	0	0	0.146463
1	0	1	0	0	0	0	0	0	0	0.137580
18	0	2	0	0	0	0	0	0	0	0.133589
21	0	1	0	0	1	0	0	0	0	0.101325
14	1	0	0	0	0	1	0	0	0	0.077960
13	1	0	0	0	1	0	0	0	0	0.065589
9	2	0	0	0	0	0	0	0	0	0.057375
0	1	0	0	0	0	0	0	0	0	0.053732
40	0	0	0	0	1	1	0	0	0	0.041056
4	0	0	0	0	1	0	0	0	0	0.032360
39	0	0	0	0	2	0	0	0	0	0.032223
5	0	0	0	0	0	1	0	0	0	0.031549
44	0	0	0	0	0	2	0	0	0	0.029441

Figure 26: MI scoring results for new features. Rows represent the new feature, values at rows represent the power of the original feature. For example, feature 3 is carat with MI 1.634221. Also, feature 33 is carat^2 with MI 1.630034.

When we look at the MI scores, we can see that products that contain the carat feature ranks best, after those features, products that contain x ranks best. Also, products that contain the depth or table ranks worst. Therefore we can conclude that the most salient features are the ones that contain carat features in their product.

Q5.2 What degree of polynomial is best? How did you find the optimal degree? What does a very high-order polynomial imply about the fit on the training data? What about its performance on testing data?

- Best degree of polynomial is 2 with alpha = 1000 for Ridge regression with average RMSE validation error 0.218255

Grid search on following parameters are carried out to find the best hyperparameter set that yields the greatest negative RMSE.

- Degree = 2, 3, 4 (We were not able to do grid search with degree values 5 and 6 since they required huge memory space. Note that, we are using all features for obtaining best model performance)

- Alpha for Ridge: 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3, 1e4, 1e5

Although we have used degrees higher than 2, we have found out that the best degree is 2. Polynomial degrees can increase the complexity of a model by introducing interaction terms and high powers which leads the model to have non-linearity. We could have decreased training error to 0 by having large degrees. We were able to fit training data perfectly, however our model will not generalize well to the testing data since using high degrees lead to overfitting and fitting to noise. Therefore, it is possible to see very low training error, but high testing error for the cases with high polynomial degree. This is the main reason why grid search chose degree 2 rather than 3 or 4. On one hand, as we have proved with test results, polynomial regression with degree 2 performs better than all models in the linear regression. Because by introducing non-linearity to fitting, we were able to grasp non-linear relationships which improves the model performance. From Table 5 we can see that the best model is the polynomial regression model.

Test Results for Ridge regression with alpha = 1000 and polynomial degree = 2.

Model	Test RMSE	Test R^2
Lasso(alpha = 0.0001)	0.3069262427883038	0.9056603356563919
Ridge(alpha = 10)	0.3069078668918593	0.905671631686289
OLS	0.30692312581679887	0.9056622517685481
Polynomial degree 2 + Ridge(1000)	**0.21896731503022598**	**0.9519841228606234**

Table 5: Test Results for linear regression and polynomial regression

3.3.3 Neural Network

Q6.1 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).

A grid search on the following hyperparameters are performed to find the best hyper-parameter set.

Weight decay(alpha): 1e-6, 1e-3, 1e1
Hidden Layer Sizes: (10), (10,10), (100), (100,100), (200), (200,200)
Activation : identity(none), ReLU

Although we know that there are 36 models to search, we wanted to see the difference between identity and ReLU activation. For this question, you may disregard identity activation which will bring 18 models to search (Qualifies < 20). Every ReLU model is better than the identity model which is explained in the 6.3.

Best hyperparameter set is: alpha = 0.000001, hidden layer size = (100,100), activation = relu

Test Results of Best hyperparameter set:

Model Best MLPRegressor	Test RMSE	Validation Average RMSE	Test R2
alpha = 1e-3, hidden layer size = (200,200), activation = relu	**0.14004087951847932**	**0.140744**	**0.9803602506904753**

Table 5: Results of Fully Connected Neural Network Regressor

Q6.2 How does the performance generally compare with linear regression? Why?

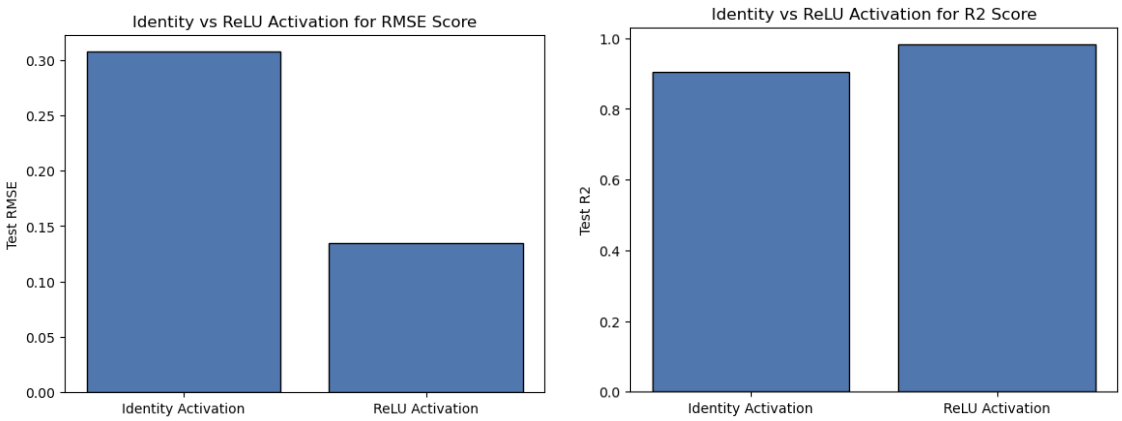


Figure 25: Best Identity activation model vs Best ReLU activation model

Model Best	Test RMSE	Test R2
Identity	0.30821918314476354	0.9048638415420397
ReLU	**0.14004087951847932**	**0.9803602506904753**
Lasso(alpha = 0.0001)	0.3069262427883038	0.9056603356563919

Table 6: Comparison between identity activation, ReLU activation and best linear regression model

Note that, when we have used Identity activation, the results are nearly the same with the linear regression. However, when we have introduced non-linearity we can see the increase in performance with ReLU.

Using a linear activation function in a neural network is similar to performing linear regression because both models assume a linear relationship between the input and output variables. In

both cases, the model tries to fit a straight line to the data by adjusting the weights or coefficients of the input variables.

In a neural network, the output of each neuron is computed as a linear combination of the input values, followed by the application of the activation function. If the activation function is linear, it simply returns the same value as the linear combination of the input values, so the output of the neuron is proportional to the input values. Therefore, the output of the entire network can be written as a linear function of the input values, with the weights representing the coefficients of the linear regression model.

Similarly, in linear regression, the output variable is modeled as a linear function of the input variables, with the coefficients representing the weights of the model. The objective of linear regression is to find the values of the coefficients that minimize the difference between the predicted and actual values of the output variable.

Therefore, using a linear activation function in a neural network yields the same result as linear regression because both models assume a linear relationship between the input and output variables and try to fit a straight line to the data by adjusting the weights or coefficients of the input variables. However, neural networks have the advantage of being able to model non-linear relationships between the input and output variables by using non-linear activation functions, while linear regression is limited to linear relationships.

Q6.3 What activation function did you use for the output and why?

We have used ReLU (Rectified Linear Unit) because it offers several advantages over the identity activation function and other activation functions, including:

1. Non-linearity: ReLU introduces non-linearity into the network, which allows the network to learn more complex and non-linear relationships between the input and output. This can help the network to better capture the underlying patterns in the data and improve its predictive accuracy.
2. Sparsity: ReLU can produce sparse representations, meaning that some of the output values can be zero, while others can be non-zero. This can help to reduce the number of parameters in the network and make it more computationally efficient.
3. Gradient propagation: ReLU can help to mitigate the vanishing gradient problem by allowing gradients to propagate more easily through the network. This can make it easier to train deep networks and improve their performance.
4. Interpretability: ReLU can be more interpretable than other activation functions, such as sigmoid or tanh, because it produces output values that are closer to the input range. This can make it easier to understand the behavior of the network and interpret its predictions.

The identity activation function, on the other hand, is a linear function that simply returns the input value. When identity activation function is used, it is useless to introduce additional layers

to the neural network, since all layers can be summarized with a single weight vector which is multiplication of weight vectors. Sigmoid and tanh can be easily saturated at both extremes which will yield no learning.

Q6.4 What is the risk of increasing the depth of the network too far?

Increasing the depth of a neural network beyond a certain point can lead to several risks, including:

1. Overfitting: When a neural network becomes too complex, it can start to memorize the training data rather than learning general patterns that can be applied to new data. This leads to overfitting, where the network performs well on the training data but poorly on the test data. This can be especially problematic if the training data is noisy or unrepresentative of the population from which the data was sampled.
2. Vanishing gradients: When the network becomes too deep, the gradients that are used to update the weights during training can become very small or even zero, which can cause the network to stop learning. This is known as the vanishing gradient problem, and it can make it difficult to train deep networks.
3. Exploding gradients: On the other hand, the gradients can also become very large, which can cause the weights to update too much during training, leading to numerical instability and making the network difficult to train.
4. Computational complexity: As the depth of the network increases, the number of parameters to be learned also increases, which can make the training process computationally expensive and slow. This can make it difficult to scale the network to larger datasets or to deploy it on resource-limited devices.
5. Difficulty of optimization: As the depth of the network increases, the optimization problem becomes more complex, and it can become difficult to find the optimal set of weights that minimize the loss function.

In summary, increasing the depth of a neural network beyond a certain point can lead to diminishing returns or even result in poorer performance. It is important to carefully balance the complexity of the network with the amount of available data and the complexity of the task at hand. Techniques such as regularization, dropout, and batch normalization can help mitigate some of these risks and make it easier to train deep neural networks.

3.3.4 Random Forest

Random Forest hyperparameters:

- Maximum number of features
- Number of trees
- Depth of each tree

Q7.1 Explain how these hyper-parameters affect the overall performance. Describe if and how each hyper-parameter results in a regularization effect during training.

Effect of number of trees:

The number of trees in a random forest regressor can have a significant effect on its overall performance. In general, increasing the number of trees in a random forest can improve the accuracy and stability of the model's predictions, up to a certain point.

When the number of trees is too low, the model may suffer from high variance and overfitting, meaning that it has learned the training data too well and may not generalize well to new, unseen data. As a rule of thumb, increasing the number of trees can improve the performance of a random forest regressor up to a certain point of diminishing returns. Beyond that point, adding more trees may not yield significant improvements in performance but may increase the computational cost and training time of the model.

Effect of maximum number of features:

The maximum number of features used in each split of a Random Forest Regressor can also have a significant effect on the overall performance of the model.

On one hand, using a larger number of features can improve the model's ability to capture complex patterns and interactions in the data. This is because a larger number of features provides more information to the model, allowing it to better discriminate between the different classes or predict the target variable. Using more features can also increase the diversity among the trees in the forest, which can improve the model's performance and reduce overfitting.

On the other hand, using too many features can also lead to overfitting, where the model learns noise in the training data rather than the underlying patterns. This is because some of the features may be irrelevant or redundant, and including them in the model can increase the complexity of the model without improving its performance. Therefore, using small amounts of features can have a regularization effect.

Effect of depth of each tree:

The depth of each tree in a Random Forest Regressor can have a significant effect on the overall performance of the model.

Note that deeper trees can capture more complex relationships in the data, leading to higher accuracy and a better fit to the training data. This is because deeper trees can split the data into smaller and more homogeneous subsets, allowing the model to learn more detailed patterns and interactions between the input features.

However, deeper trees can also lead to overfitting, where the model learns the noise in the training data rather than the underlying patterns, resulting in poor generalization performance on new data. Overfitting can occur when the model becomes too complex and is able to fit the noise in the training data, leading to high variance and poor performance on new data.

Validation and Test Results:

A grid search on the following hyperparameters are performed to find the best hyper-parameter set.

Max depth = [2,3,4,5,6,7,8,9,10,15,25,50,100,250]
N estimators = [1,5,10,100]
Max features =[1,2,3,4,5,6,7,8,9]

In total there are 504 candidates, totalling 5040 fits, however since the depth of the tree and number of features are small, grid search does not take too long.

- Best parameter set is: depth = 15, max features = 6, number of trees = 100
- Average validation RMSE for best parameter set: **0.135473**
- OOB score: 0.981747, which means that number of estimators is adequate.
- Test R-square: **0.982136**
- Test RMSE: **0.133557**

Note that Test R-square is very similar to OOB score which means that OOB score can be used to estimate model performance without using extra data such as test dataset.

Feature(sorted according to RF score)	Random Forest Feature Importance
carat	0.51705338
y	0.28003791
x	0.09411942
clarity	0.06313826
color	0.03207181
z	0.00674612
depth	0.00309083
table	0.00188094
cut	0.00186134

Table 7: RF scores for each feature

Note that, there is a slight deviation between MI score ranking and RF score ranking, however carat and y gets the most score in every scorer. Random Forest gives higher scores for categorical variables clarity and color.

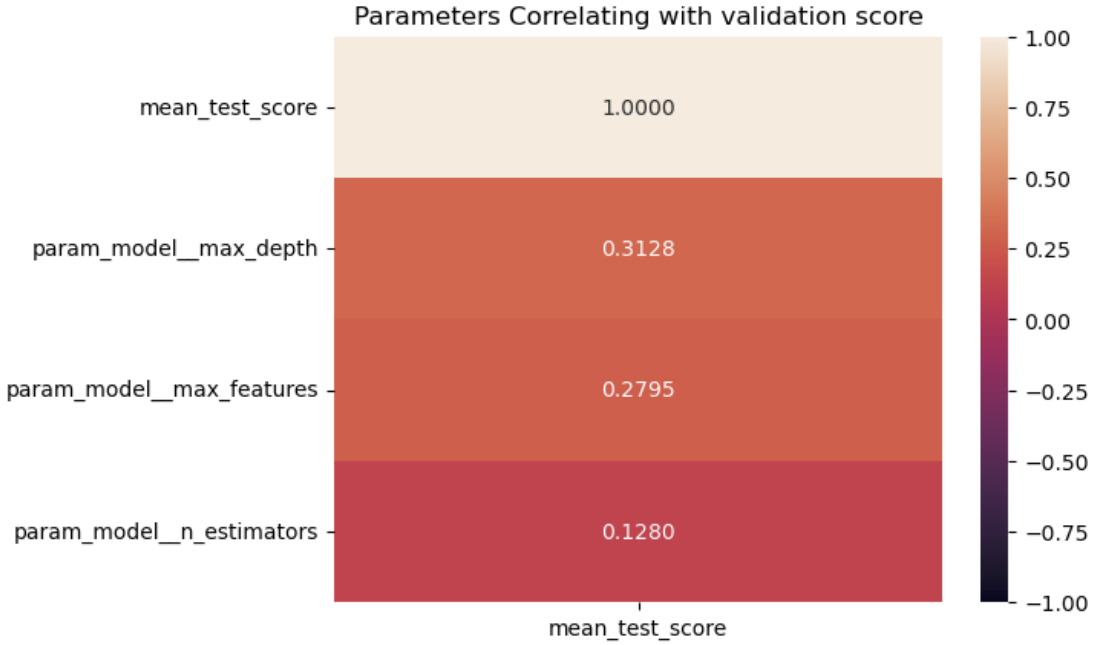


Figure 26: Correlation between hyperparameters and average validation R-square

It can be easily seen from the Figure 26 that number of estimators does not yield to much increase in performance. However, we can see that max_depth and max_features hyperparameters are important for model complexity and generalization.

Q7.2 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 create a highly non-linear decision boundary by combining multiple decision trees, each of which uses a different subset of features to make decisions. Although each decision tree applies a threshold on a single feature at each node, the combination of multiple trees with different feature subsets can capture complex non-linear relationships between features.

In a random forest, each decision tree is trained on a random subset of the training data, as well as a random subset of the available features. This randomness helps to reduce overfitting and increase the diversity of the trees in the forest.

During training, each decision tree recursively splits the data into smaller subsets based on the values of the selected features at each node. By using different subsets of features, the trees in the forest can capture different aspects of the underlying relationship between the input features and the target variable. The final prediction of the random forest is then made by aggregating the predictions of all the individual trees.

The combination of multiple decision trees with different feature subsets can create a highly non-linear decision boundary, even though each individual tree only applies a threshold on a single feature at each node. This is because the combination of different thresholds on different features can capture complex interactions between the features, allowing the model to represent non-linear relationships between the input features and the target variable.

Q7.3 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?

We have picked the 12th tree from the RandomForestRegressor(max_depth = 4, max_features = 6, n_estimators = 100, oob_score = True). This parameter set yields to best validation score when depth = 4.

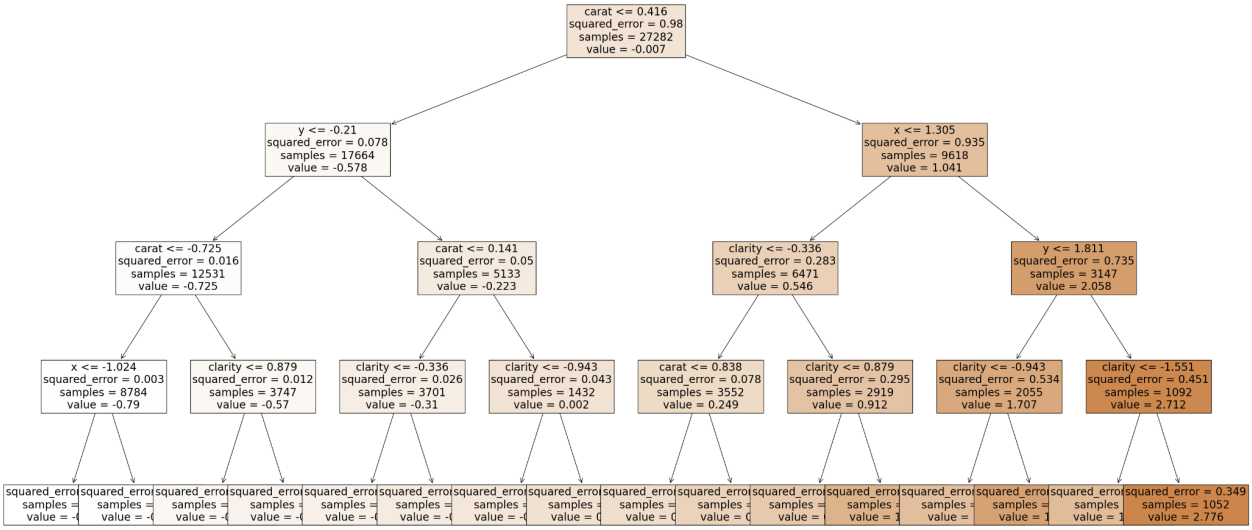


Figure 27: 12th tree from the best RF Regressor with depth = 4

At the root node, the carat feature is selected as expected. Carat feature has the most descriptive statistics for the price as it was seen from MI and F score. From the tree we can see that the mostly used features for division are carat,y, x and clarity. We have obtained a 0.9283 OOB score with only a depth of 4 which is better than the linear regression.

Feature(sorted according to RF score)	Random Forest Feature Importance
carat	4.33500727e-01
y	4.06234279e-01
x	1.08239895e-01
clarity	3.89619859e-02
color	6.87621481e-03
z	5.72789320e-03
depth	3.04407586e-04
cut	1.54597876e-04
table	0.00000000e+00

Table 8: RF scores for each feature, scores for the RF Regressor with depth = 4

Feature	Ranking By F score	Ranking By MI score	Ranking By RF score
carat	1	1	1
x	2	3	3
y	3	2	2
z	4	4	6
color	5	6	5
clarity	6	5	4
table	7	9	9
cut	8	7	8
depth	9	8	7

Table 9: Rankings of features according to three different scores.

Note that, carat is the ranked 1st feature for all rankings. Other features can be grouped into three categories. Features x and y are ranked 2nd or 3rd for every ranking type. Features z, color and clarity are ranked 4th or 5th or 6th for every ranking type. Features table, cut and depth are ranked 7th, 8th and 9th for every ranking type.

Q7.4 Measure “Out-of-Bag Error” (OOB). Explain what OOB error and R2 score means.

OOB score for the best model is 0.9817472011947562 which is very close to the average validation R-Square which is 0.981652. We were able to compute validation errors by simply using out of bag samples.

OOB score for the best model with depth = 4 is 0.9283783394222521 which is very close to the test R-Square which is 0.929408336752099. Test RMSE is 0.26549.

OOB:

In Random Forest Regression, the Out-of-Bag (OOB) score is an estimate of the model's prediction accuracy, which is computed based on the samples that were not included in the model training process.

Random Forest Regression creates multiple decision trees by randomly selecting subsets of the features and observations, and then aggregates their predictions to form a final output. For each decision tree, a random subset of the observations is selected for training, and the remaining samples are used as OOB samples for evaluating the model's performance.

During training, the OOB samples are never used to fit the decision tree. Therefore, for each OOB sample, the model can use the corresponding decision trees to make a prediction and compare it to the true value. By averaging the prediction errors over all the OOB samples, we can obtain an estimate of the model's generalization performance, which is referred to as the OOB score.

The OOB score can be a useful metric for assessing the performance of the model and tuning its parameters. Because it is based on the samples that were not used during training, it provides a more reliable estimate of the model's prediction accuracy on new, unseen data compared to traditional cross-validation methods

R^2

R^2 is a statistical metric commonly used in regression analysis that measures the goodness of fit of a regression model to the data. R^2, also known as the coefficient of determination, ranges from 0 to 1 and represents the proportion of the variance in the dependent variable that is explained by the independent variables included in the model.

An R^2 value of 0 indicates that the model does not explain any of the variability in the dependent variable, while an R^2 value of 1 indicates that the model explains all of the variability in the dependent variable. In general, the higher the R^2 value, the better the model fits the data.

$R^2 = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - \bar{y}_i)^2}$, where $\sum_i (y_i - f_i)^2$ is squared error and the denominator is constant. Therefore R^2 not only brings the model performance metric in a standard scale, but also negatively and perfectly correlates with RMSE. When RMSE is 0, R^2 is 1. When RMSE is huge, if the model performs better than average R^2 is between 0 and 1. When a model performs worse than average, it can get negative values.

For random forest, OOB scoring uses the R^2 technique to show the performance of model without using extra data.

3.3.5 LightGBM and Bayesian Optimization

For this section, we selected the LightGBM boosting algorithm. We used a train/test split of 0.8/0.2. We used 10-fold cross-validation for the Bayesian optimization. RMSE metric (negative RMSE) was used as the scoring function.

Q8.1 Determine important hyperparameters of LightGBM and their search space.

We examined the algorithm and its hyperparameters from the documentation. The selected hyperparameters are shown in Table 10.

Learning Rate (learning_rate): It is the learning rate of the LightGBM training.

Max. Depth (max_depth): Max. tree depth for the base learners. 0 or -1 indicate that there is no limit for depth.

No. of Iterations (num_iterations): The number of training (boosting) iterations.

No. of Leaves (num_leaves): Max. tree leaves for the base learners.

Subsample Ratio of Features (colsample_bytree): Subsample ratio of features (columns) for each tree.

Subsample Ratio of Training Instance (subsample): Subsample ratio of the training instance.

L1 Regularization Term (reg_lambda): The L1 regularization term for weights in the objective.

L2 Regularization Term (reg_alpha): The L2 regularization term for weights in the objective.

No. of Estimators (n_estimators): The number of boosting trees to be fitted.

Min. Data in Leaf (min_data_in_leaf): Min. no. of observations required for a node to be added to the tree.

Hyperparameter	Search Space
Learning Rate	Real(0.00001, 0.9, prior='log-uniform')
Max. Depth	Integer(-1,500)
No. of Iterations	Integer(2, 500)
No. of Leaves	Integer(5,500)
Subsample Ratio of Features (Columns)	Real(0.05, 1.0, 'uniform')
Subsample Ratio of Training Instance	Real(0.05, 1.0, 'uniform')

L1 Regularization Term (alpha)	Real(1e-8, 1e+3, 'log-uniform')
L2 Regularization Term (lambda)	Real(1e-8, 1e+3, 'log-uniform')
No. of Estimators	Integer(5,500)
Min. Data in Leaf	Integer(2,200)

Table 10: Hyperparameters of LightGBM and their corresponding search space for Bayesian optimization.

Q8.2 Apply Bayesian optimization and find the best combination of hyperparameters. Report the best parameters and the corresponding RMSE.

As shown in Table 11, the average validation RMSE is **0.13511**, the test RMSE is **0.1339**, and the test R2 is **0.98204**.

Metrics for Optimal Hyperparameters	**Value**
Avg. Train RMSE	0.08267
Avg. Validation RMSE	0.13511
Test RMSE	0.13390
Test R2	0.98204

Table 11: Training, validation and test metrics for the best model after Bayesian optimization.

Table 12 shows the optimal values of hyperparameters from the best model.

Hyperparameter	**Optimal Value**
Learning Rate	0.378
Max. Depth	-1 (no limit)
No. of Iterations	500
No. of Leaves	500
Subsample Ratio of Features (Columns)	0.616
Subsample Ratio of Training Instance	0.05
L1 Regularization Term (alpha)	3.115e-07
L2 Regularization Term (lambda)	1000

No. of Estimators	5
Min. Data in Leaf	2

Table 12: Optimal values of the hyperparameters for the LightGBM after Bayesian optimization.

Q8.3 Interpret the effects of hyperparameters using the Bayesian optimization results for performance, regularization (generalization gap), and fitting efficiency.

To measure hyperparameters' effect on the performance, we calculated the Pearson correlation coefficient for **negative** average validation RMSE with respect to each hyperparameter. We sorted

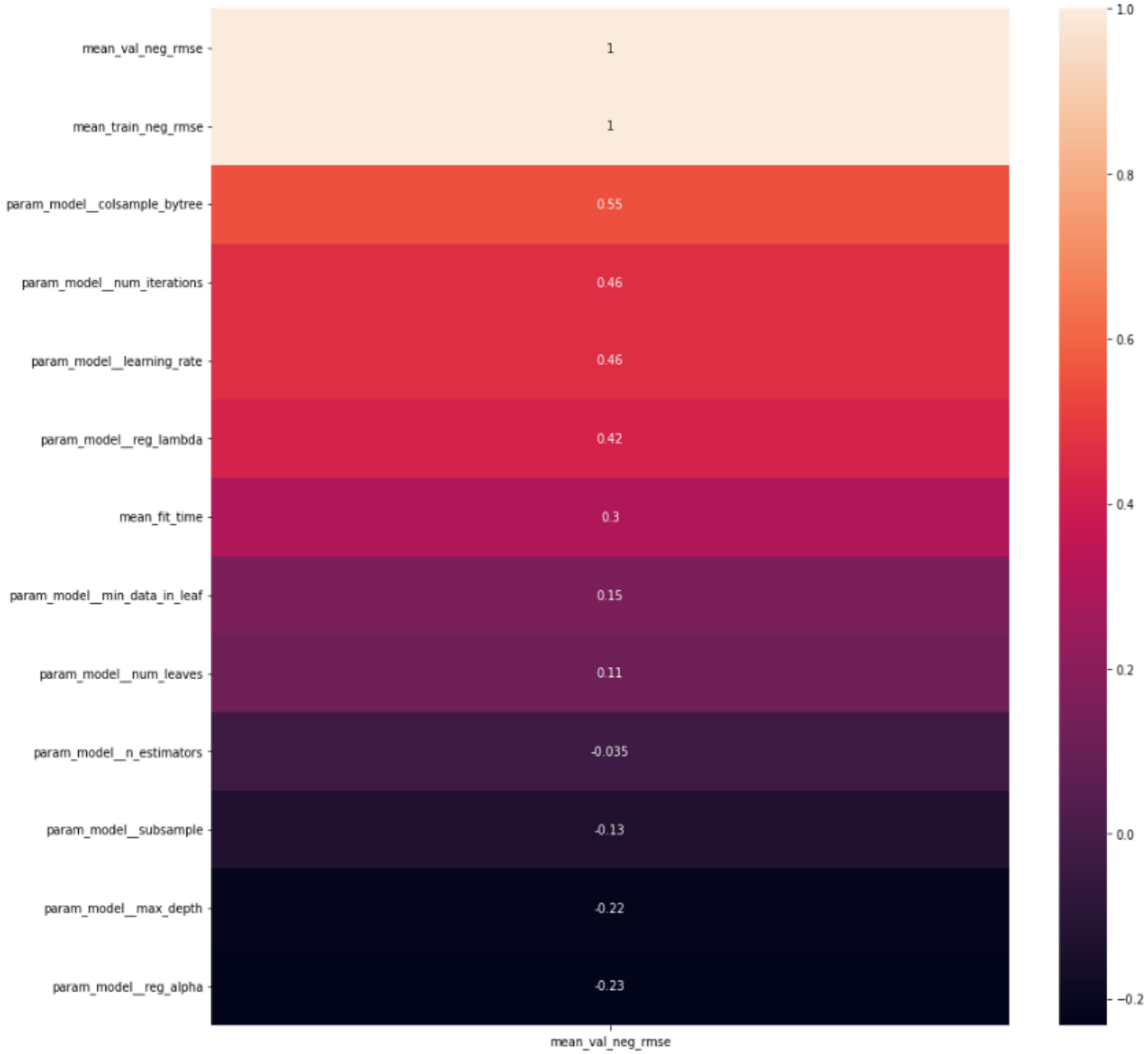


Figure 28: Sorted Pearson correlation coefficients for **negative** average validation RMSE with respect to hyperparameters.

Figure 28 demonstrates the sorted correlation coefficients for the performance using RMSE. We observe that subsample ratio of features (columns), the number of iterations, learning rate, L2 regularization term, L1 regularization term, and max. depth hyperparameters significantly affect the validation performance with the given order. Among these parameters, subsample ratio of features (columns), the number of iterations, and learning rate dramatically affect the validation performance.

We also observe that the subsample ratio of features has a strong positive relation with the validation performance. The number of training iterations, learning, and L2 regularization also have strong positive relation with the validation performance. One interesting observation is a very big L2 term value. To explain this behavior, we should look at the other hyperparameters such as (number of iterations)=500, (number of leaves)=500. The high number of iterations and leaves makes the model complicated, so as to compensate for this model uses high L2 regularization. On the other hand, L1 regularization term and max. depth have negative relationship with the validation performance.

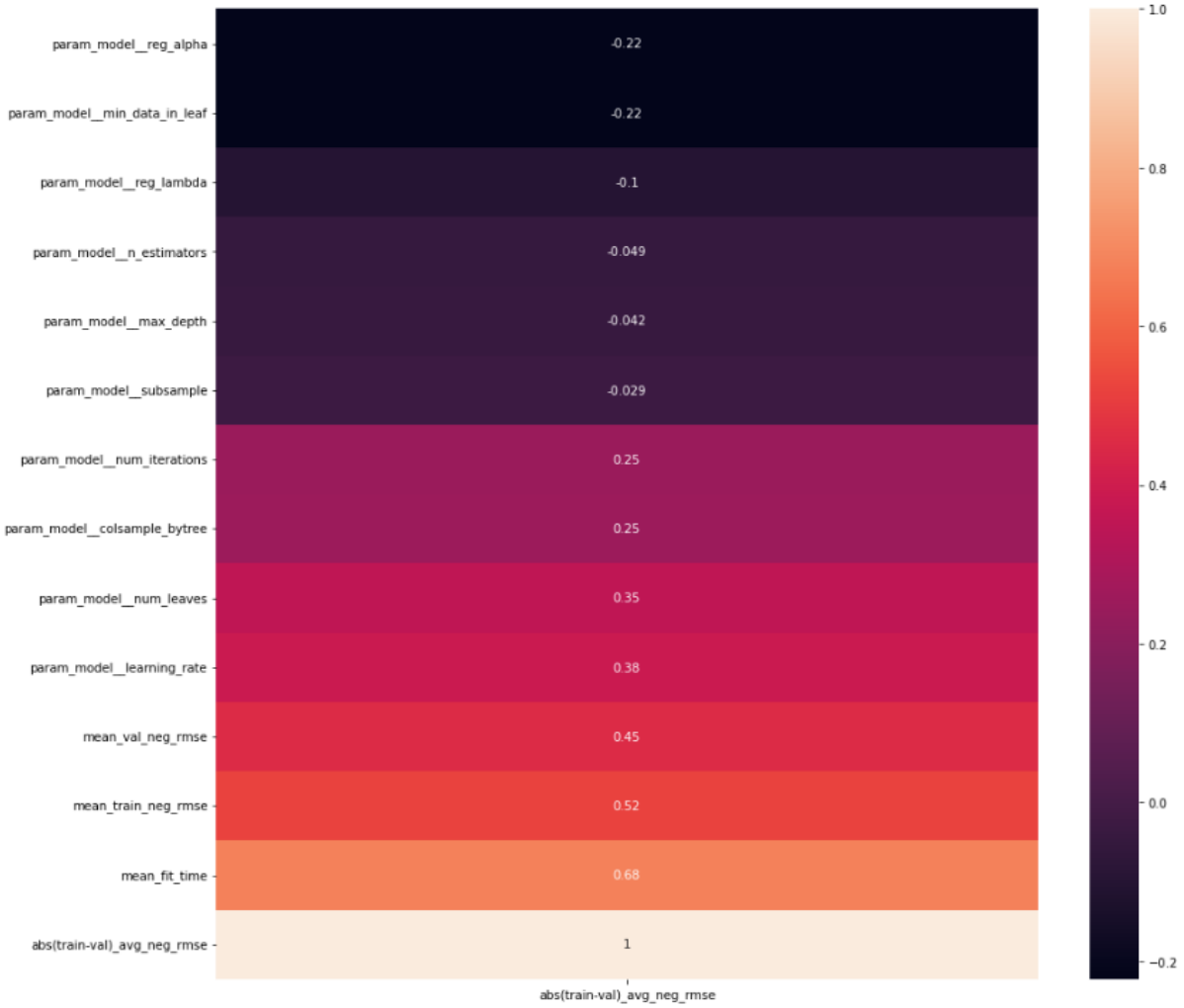


Figure 29: Sorted Pearson correlation coefficients for the absolute value of **difference** between the average training and validation RMSEs with respect to hyperparameters.

Figure 29 demonstrates the relationship between the absolute value of the **difference** between the average training and validation RMSEs to measure the regularization or generalization gap effects. We can clearly see that increasing L1 and L2 regularization terms improve the generalization since the correlation coefficients are negative. Additionally, increasing the minimum number of observations required for a leaf node improves the generalization or regularization.

Although lowering the values of the number of estimators and max. depth is usually linked with better generalization, the corresponding correlation coefficients are not significant. This may be partly due to relatively small subspace search in which these hyperparameters can be compensated with other hyperparameters that have the opposite effect.

Furthermore, increasing the number of training iterations, subsample ratio of features (columns), and the number of leaves decreases the generalization significantly as expected since these hyperparameters increase the complexity.

Moreover, increasing the learning rate also decreases the generalization as expected since too high learning rate may cause overfitting.

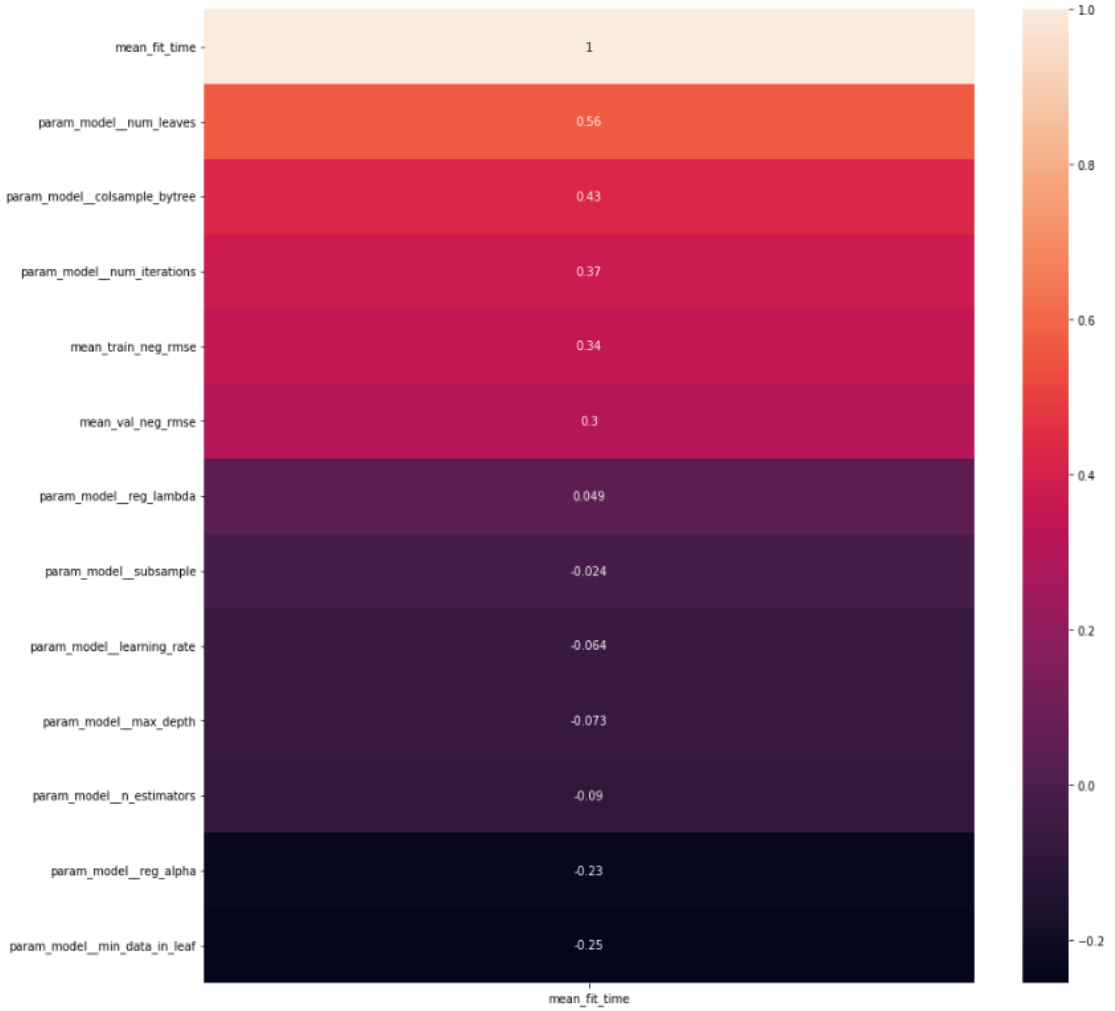


Figure 30: Sorted Pearson correlation coefficients for the average fit time with respect to hyperparameters.

As shown in Figure 30, to measure the effect of hyperparameters on model efficiency for **training**, we calculated the sorted Pearson correlation coefficients for the average fit time with respect to the hyperparameters.

Similarly, as shown in Figure 31, to measure the effect of hyperparameters on model efficiency for **evaluation**, we calculated the sorted Pearson correlation coefficients for the average score time with respect to the hyperparameters.

We observe that increasing the number of leaves, subsample ratio of features (columns), and the number of training iterations significantly increases the average fit time, which is linked with reduced efficiency.

Although increasing the values of the number of estimators and max. depth is usually linked with reduced efficiency, the corresponding correlation coefficients are not significant. Again, this may be partly due to relatively small subspace search in which these hyperparameters can be compensated with other hyperparameters that have the opposite effect.

Additionally increasing L1 regularization term and the minimum number of observations required for a leaf node decreases the average fit time.

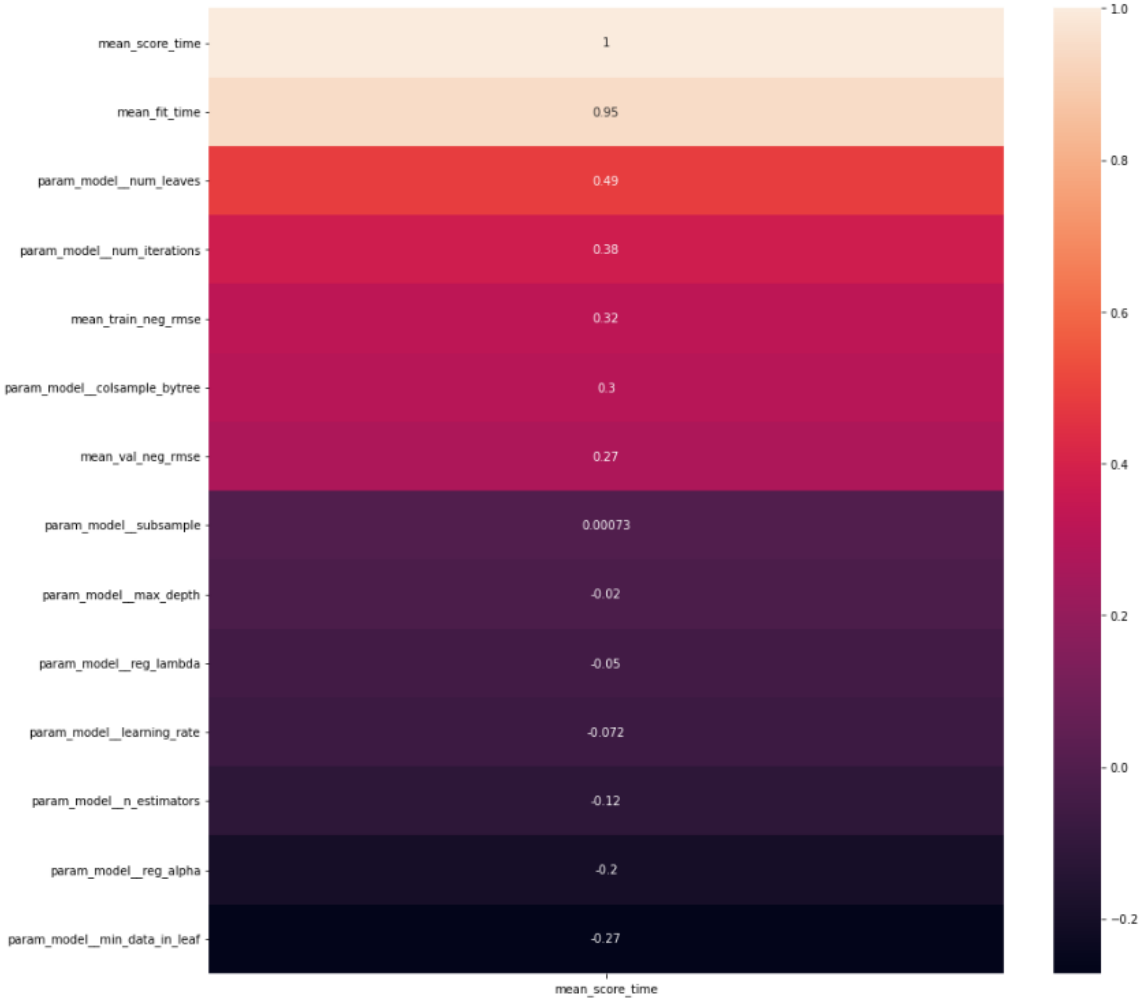


Figure 31: Sorted Pearson correlation coefficients for the average evaluation time with respect to hyperparameters.

Twitter Data

3.4 About the Data

Q9.1 Report the average number of tweets per hour, followers of users posting tweets per tweet, retweets per tweet for each hashtag.

The results are shown in Table 13.

Hashtag	Avg. #(tweets per h)	Avg. #(followers)	Avg. #(retweets)
#gohawks	292.49	2217.9	2.0132
#gopatriots	40.955	1427.3	1.4082
#nfl	397.02	4662.4	1.5345
#patriots	750.89	3280.5	1.7853
#sb49	1276.9	10374	2.5271
#superbowl	2072.1	8814.9	2.3912

Table 13: Average number of tweets per hour, followers, and retweets for each tag.

Q9.2 Plot number of tweets in hour over time for #SuperBowl and #NFL (a bar plot with 1-hour bins).

Figure 32 shows the number of tweets for #nfl hashtag, and Figure 33 shows the number of tweets for #superbowl hashtag.

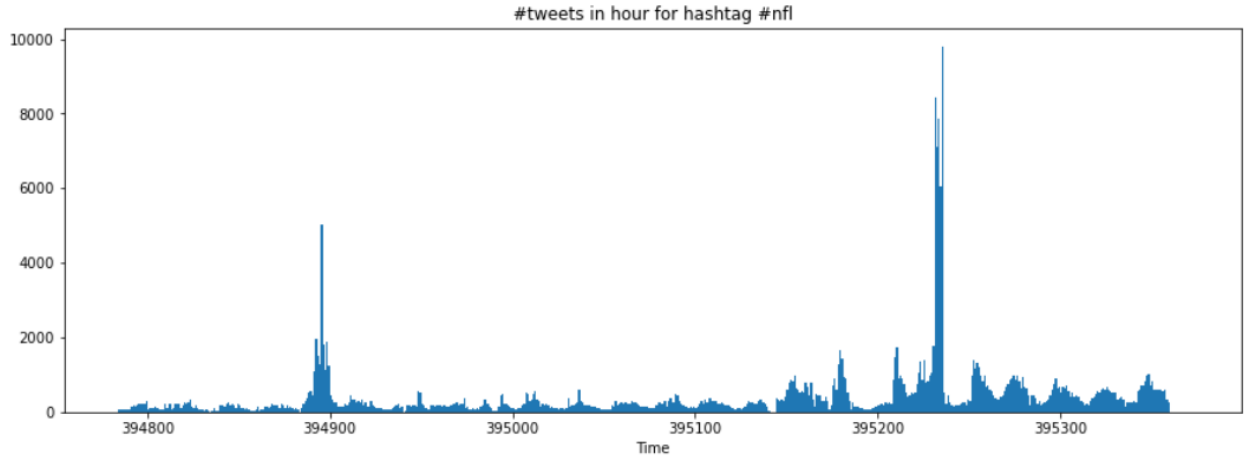


Figure 32: The number of tweets in hour over time for hashtag #nfl.

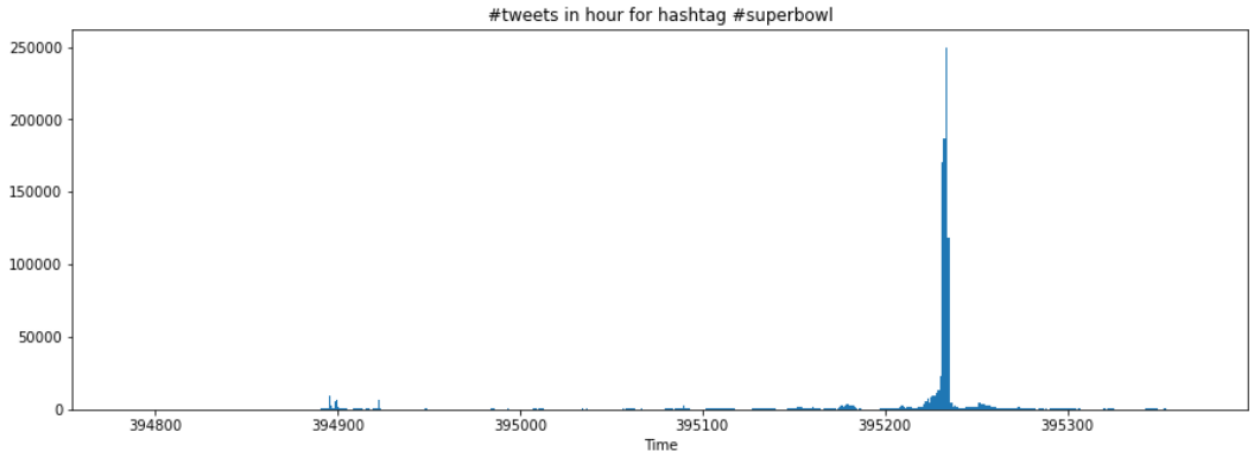


Figure 33: The number of tweets in hour over time for hashtag #superbowl.

Q10

a) Describe your task.

The task at hand involves using various Twitter features, including the number of followers, the time the tweet was posted, and additional features such as impressions and ranking scores, to predict the number of retweets that a tweet is likely to receive. To accomplish this, we will utilize a regression model known as a Multi-Layer Perceptron (MLP), which has different hidden sizes.
An MLP is a type of artificial neural network that is often used for regression and classification tasks. It consists of multiple layers of nodes, each connected to the nodes in the previous and next layers. The nodes in each layer receive inputs from the previous layer, and apply a mathematical function to those inputs to produce an output, which is then passed on to the next layer.
In the context of this task, we will train an MLP to learn the relationship between the various Twitter features and the number of retweets. The MLP will take as inputs the number of followers, the time the tweet was posted, and additional features such as impressions and ranking scores, and output a prediction of the number of retweets.

b) Explore the data and any metadata (you can even incorporate additional datasets if you choose).

We explore the correlations between different features and display them below. We observe that although the number of retweets are not strongly correlated with only one feature to some extent it correlates with other features. However, we should note that it would not be wise choice to select any feature as the single predictor as can be interpreted from the figures below:

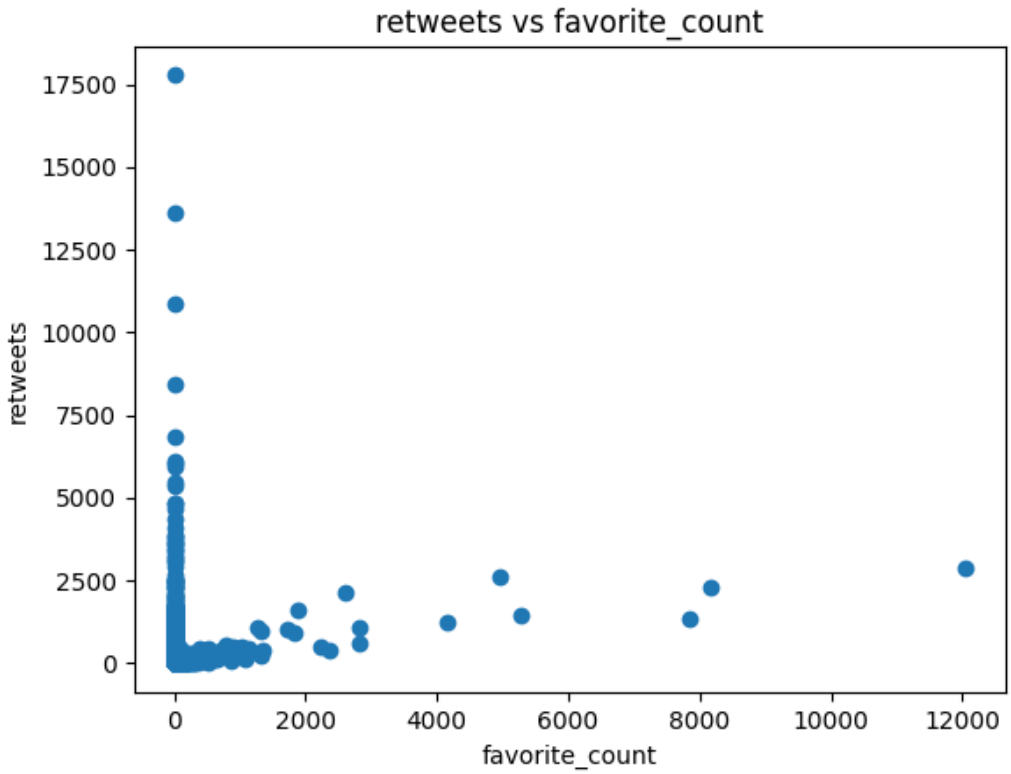


Figure 34: Retweets vs Favorite Count of a Tweet

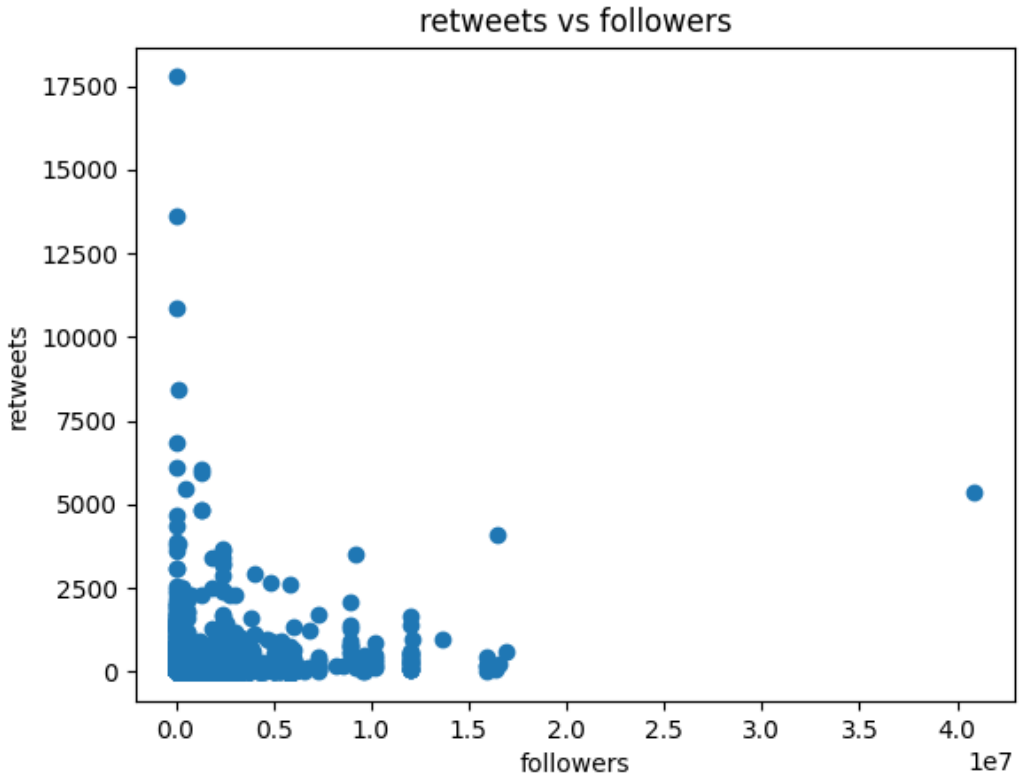


Figure 35: Retweets vs Followers

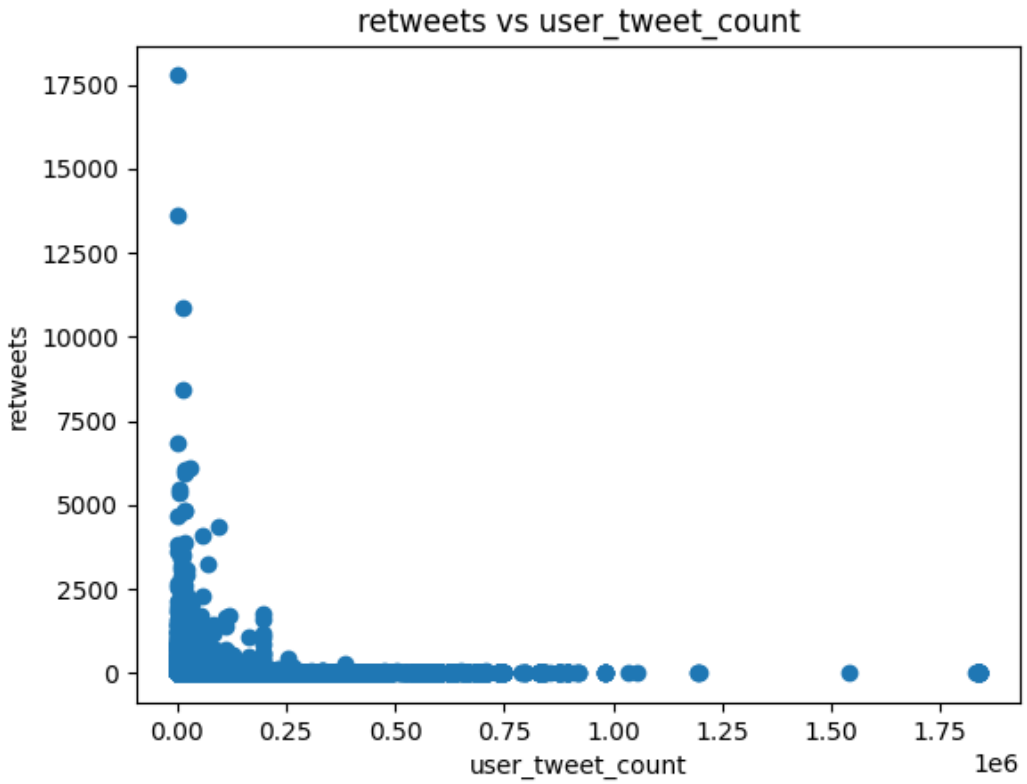


Figure 36: Retweets vs User Tweet Count

From the illustrative figures, we decide on constructing our regression model depending on the features of each tweets. Particularly, we utilize the following features `['followers', 'favorite_count', 'user_tweet_count', 'passivity', 'impression_count','ranking_score', 'mention_count', 'density', 'degree', 'url_count','hashtag_count']`. In addition to these features we also integrate the timestamp of the tweet into the regression model.

c) Describe the feature engineering process. Implement it with reason: Why are you extracting features this way - why not in any other way?

Since the data consists of different time frames including the time prior to the game, during the game and the time frame after the game we divide the time data into day and hour data and feed the network with two different information in addition to the aforementioned features. Since it might add further complication to the regression model we refrain from using the user_id as an extra feature and only included the features where we can observe a correlation with the retweet count. Then, to be able to avoid overfitting we employ a Standard Scaler. This is particularly important since some features have an order of magnitude difference with the rest of the data such as passivity. We note that we use all 6 tweet tags and carry out our experiments on each task separately.

d) Generate baselines for your final ML model.

We generate baseline models as Ridge Regression model and MLP models with different hidden layers. The reason why we include the Ridge Regression model to the baselines is that the special loss function shrinks the coefficients of the independent variables towards zero, which helps to reduce the impact of multicollinearity in the data.

e) A thorough evaluation

Root Mean Square Error	gohawks	gopatriots	nfl	patriots	sb49	superbowl
Ridge Regression	39.51	2.14	5.33	13.66	47.65	26.47
MLP(20,)	40.40	1.89	**5.17**	13.16	**46.40**	**25.74**
MLP(20,20)	**39.08**	**1.69**	5.18	**12.48**	47.21	25.79

Table 14: Evaluation of Different Architectures on the Same Prediction Analysis

We should note that the MLP test and train datasets are split as 20-80 percent of the data respectively. In addition to that, a shuffling operation is carried out before the training starts and Standard Scaling is applied to each data chunks. The models are trained for 100 iterations and the optimizer method is selected as “adam” and the activation function is ”ReLU”. The training is 5-fold cross validated and the best results are taken.

As can be seen from the evaluations of RMSE values for different architectures the nonlinearity added with the MLP model facilitates the prediction task. Although the smaller MLP model outperforms in some tags it can be observed that MLP(20,) model performance is not significantly better than MLP(20,20,) model in tags “nfl” and “superbowl”. This suggests that prediction with these tags are not complicated tasks and can be achieved with a small MLP model and increasing the depth of the MLP model doesn’t significantly change the model performance.