Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

ECE 219 Project 4 Report

Regression Analysis

In this part of the project, our goal was to perform regression analysis on two different datasets. The first one was related to determining the price of a diamond based on its characteristics. The second one was related to determining the CO emission of gas turbines.

Dataset 1: Diamond Characteristics

Handling Categorical Features and Standardization

Q1

Before performing training, we needed to handle the categorical features of this dataset. In this set, the categorical features are **cut, color** and **clarity.** Here, it is important to note that all of the values of these features are ordered in some way. Cut goes from fair to ideal, color goes from J(worst) to D(best) and clarity goes from I1(worst) to IF(best). Therefore, we utilized OrdinalEncoder to transform the features to numerical values. This encoding gave the value 0 to the worst values and gave the appropriate integer to the best values.

After this step, we standardized the features using sklearn's StandardScaler. Here, an important aspect to point out is that the label Price was also normalized along with the features. This allowed better training of some models.

After normalization, our features data had 9 columns that each had 0 mean and 1 variance. Our label consisted of a single column that also had 0 mean and 1 variance.

Data Inspection

Q2

After normalizing the columns, we can determine the relation of each feature to the target using the Pearson correlation matrix. For this dataset, the result of this operation is as follows:

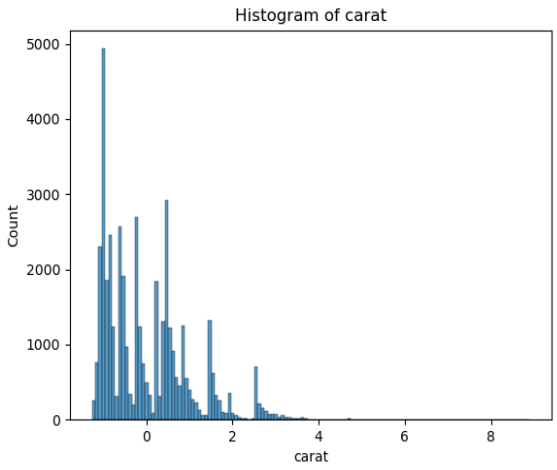
Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

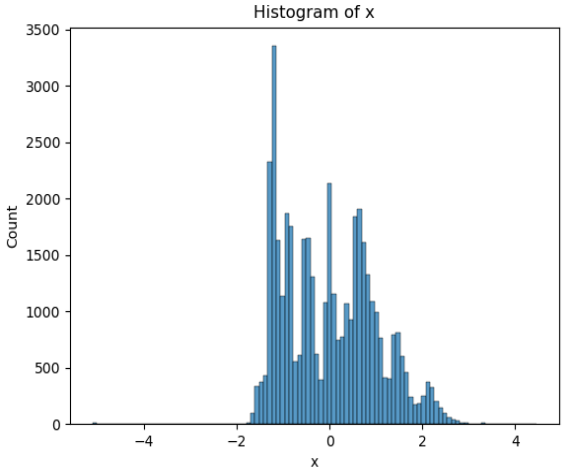
From the result, we can see that the features that are most related to the target price are **x, y, z** and **carat**. This high correlation suggests that as these features increase, price also increases. This is logical since as we know, one of the defining characteristics that define a diamond's worth is carat. Since carat is related to the volume of the diamond which depends on x, y and z values it is logical that price also depends on these features.

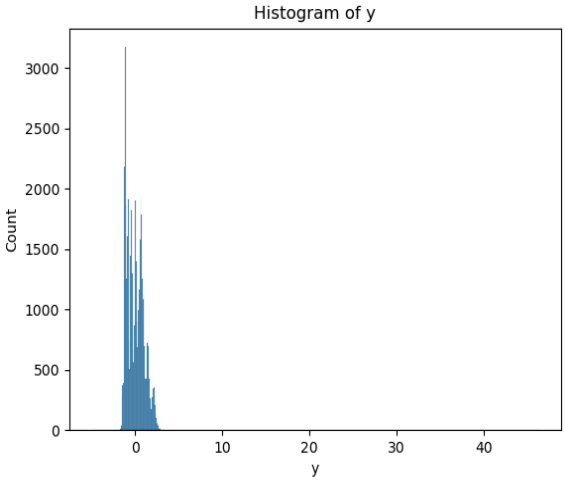
Q3

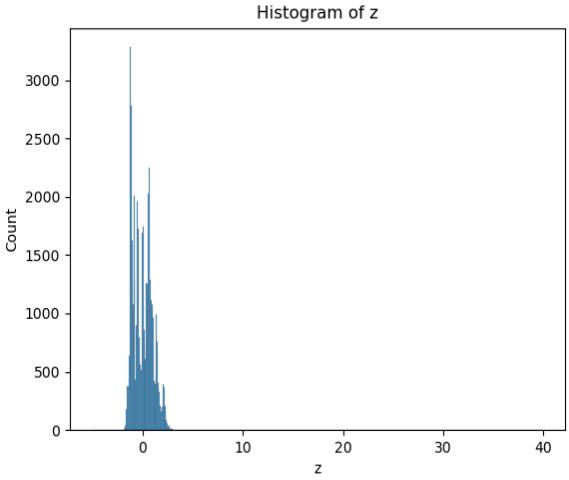
Let us plot the histogram of the numerical features after normalization.

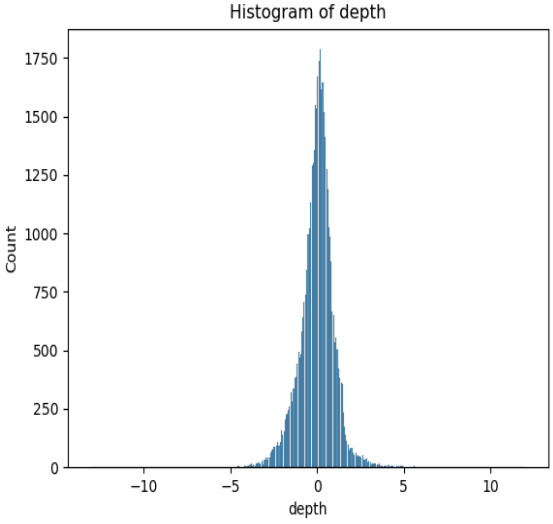
Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

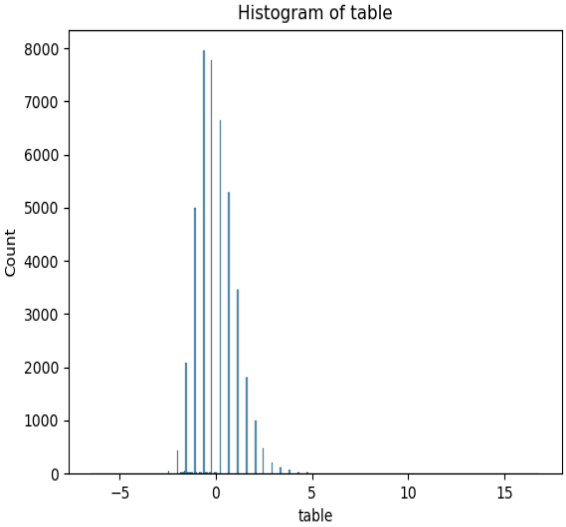










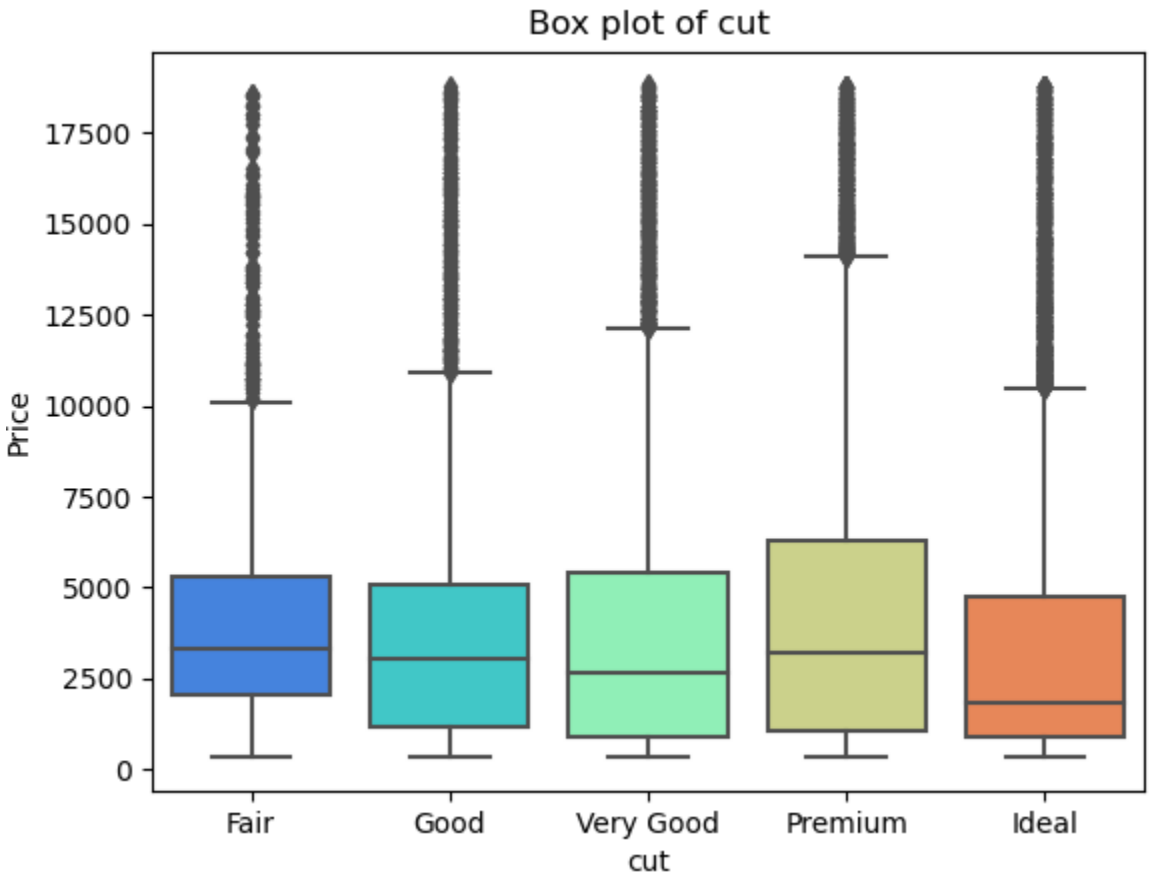


Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

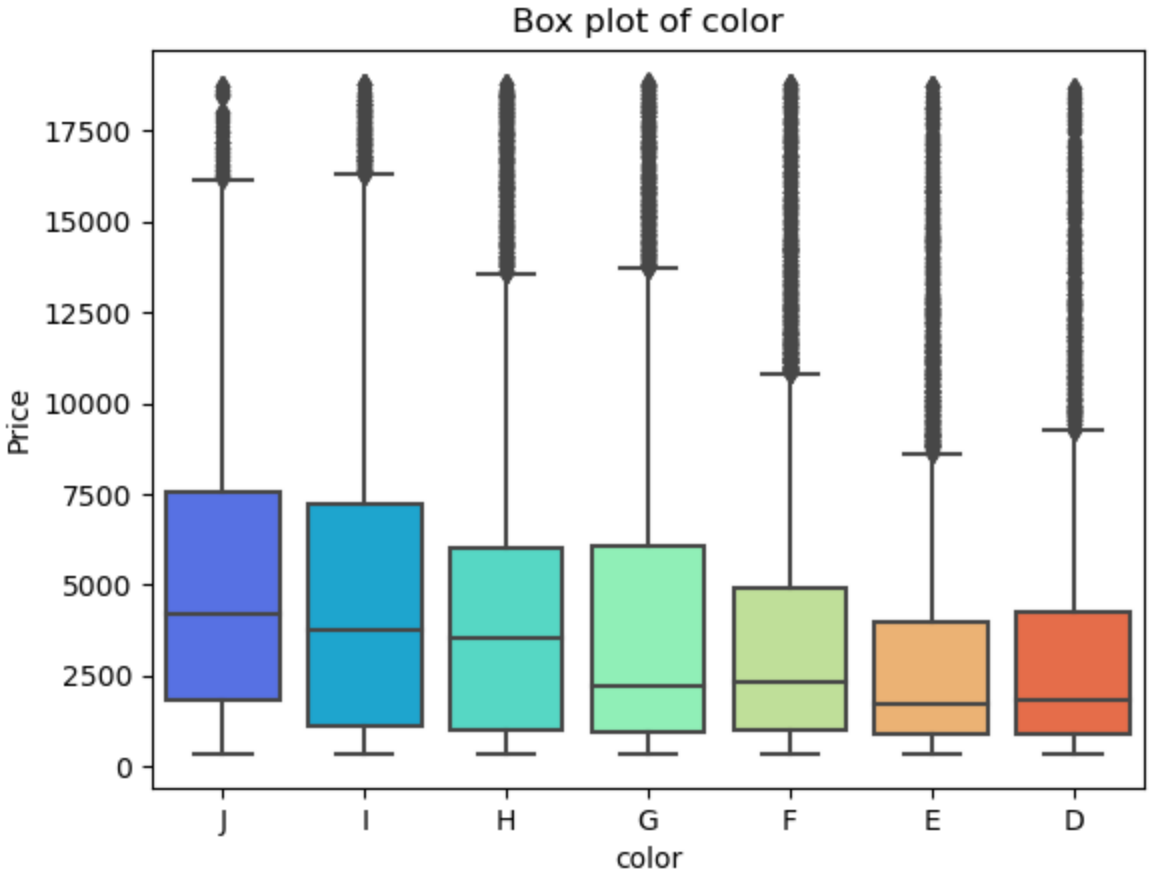
For features that are highly skewed, we can transform the feature in a way that reduces the skewness. Some of the transforms we can try are square root transformation where we take the square root of the feature, reciprocal transformation where we take the multiplicative inverse of the feature or log transform where we take the log of the feature. As a more advanced preprocessing approach, we can also implement the Box Cox transform.

Q4

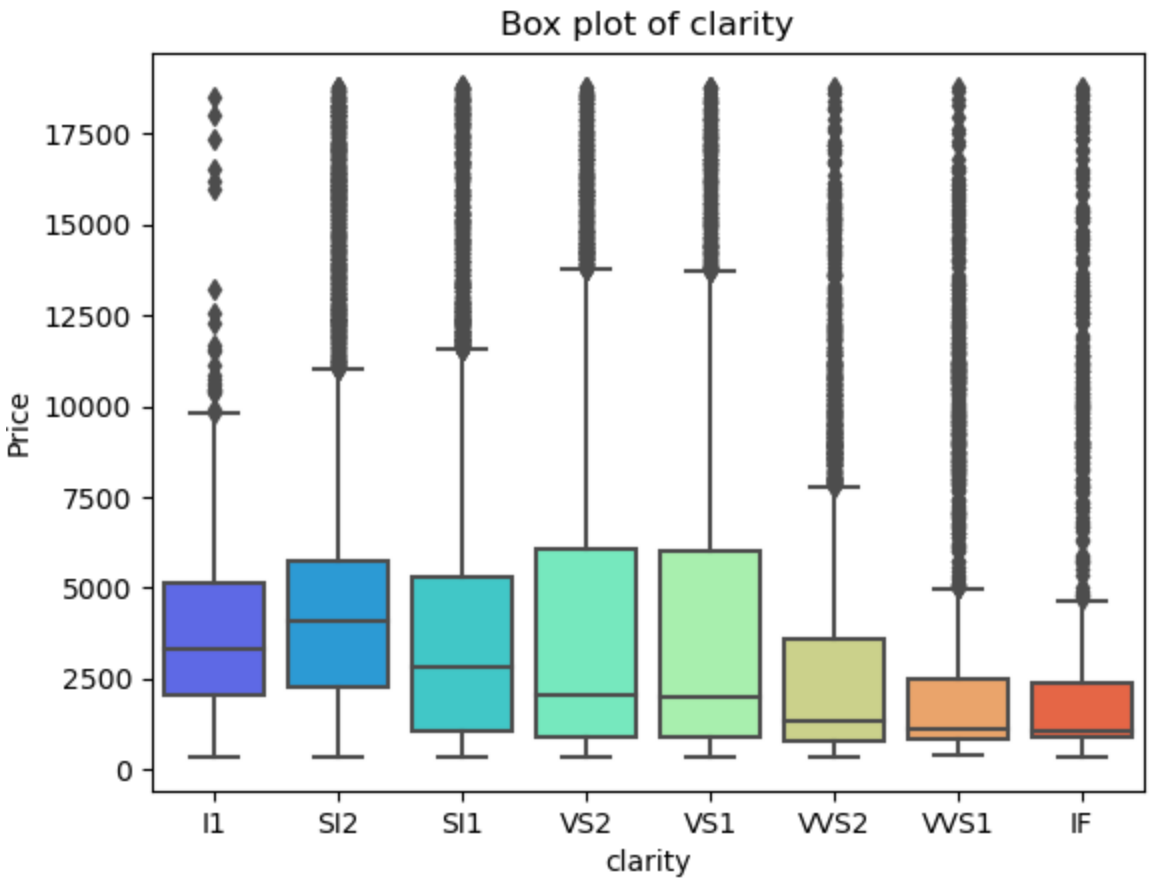
Let us now plot the box plot of the categorical features vs the target variable.



Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839



Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

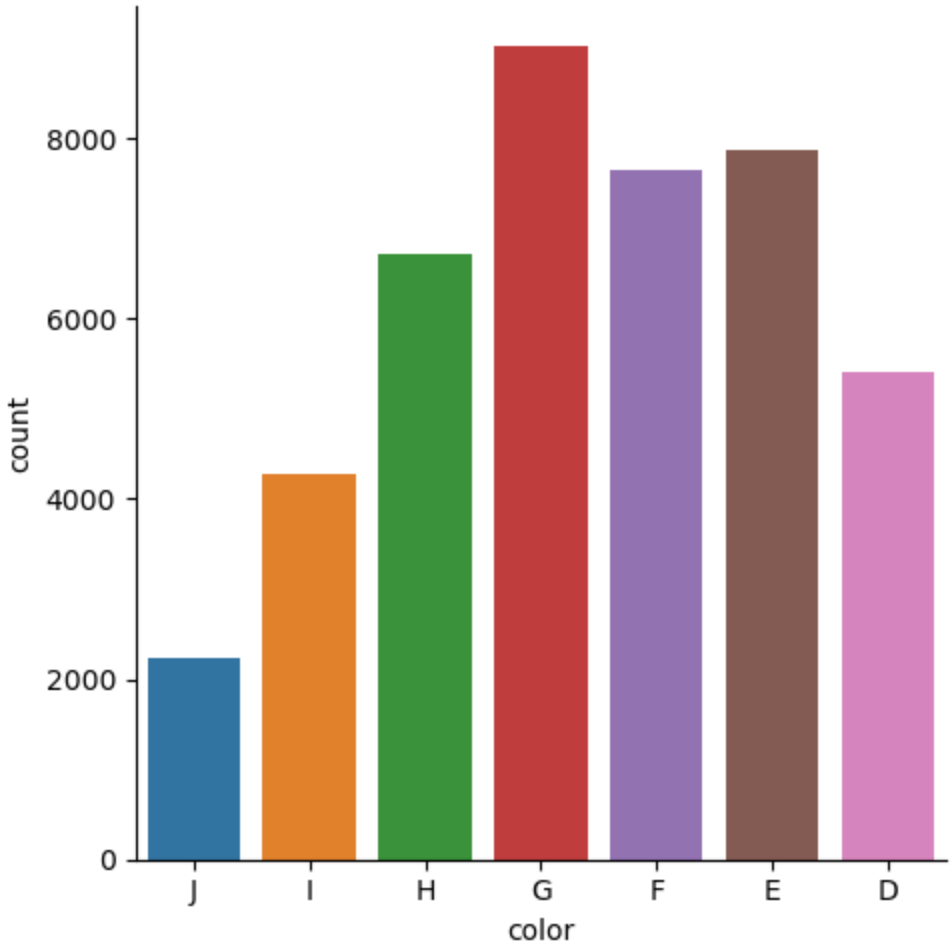


By examining the plots, we can determine that as the quality of the cut increases, the price of the diamond also tends to increase. This is an expected result. However, we also see that the color rating of J which is the worst rating has the highest mean price. Similarly, the highest rating for clarity has the least mean price.These are unexpected results, as we expected the price to increase as the rating of the feature improves. The observed results can imply that the color and clarity features are not heavily related with the price of a diamond or we may not have enough data points that accurately represent all the possible values of the categorical features.

Q5

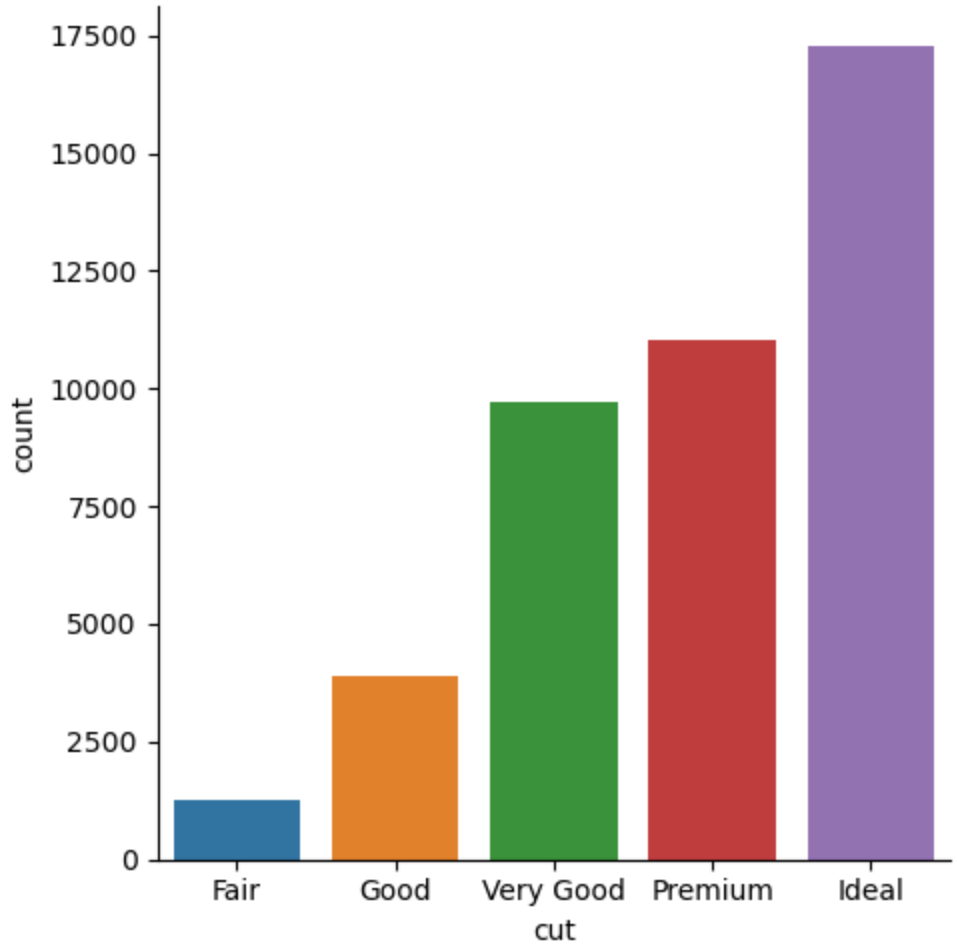
Let us now plot the counts of the categorical features of our data.

Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839



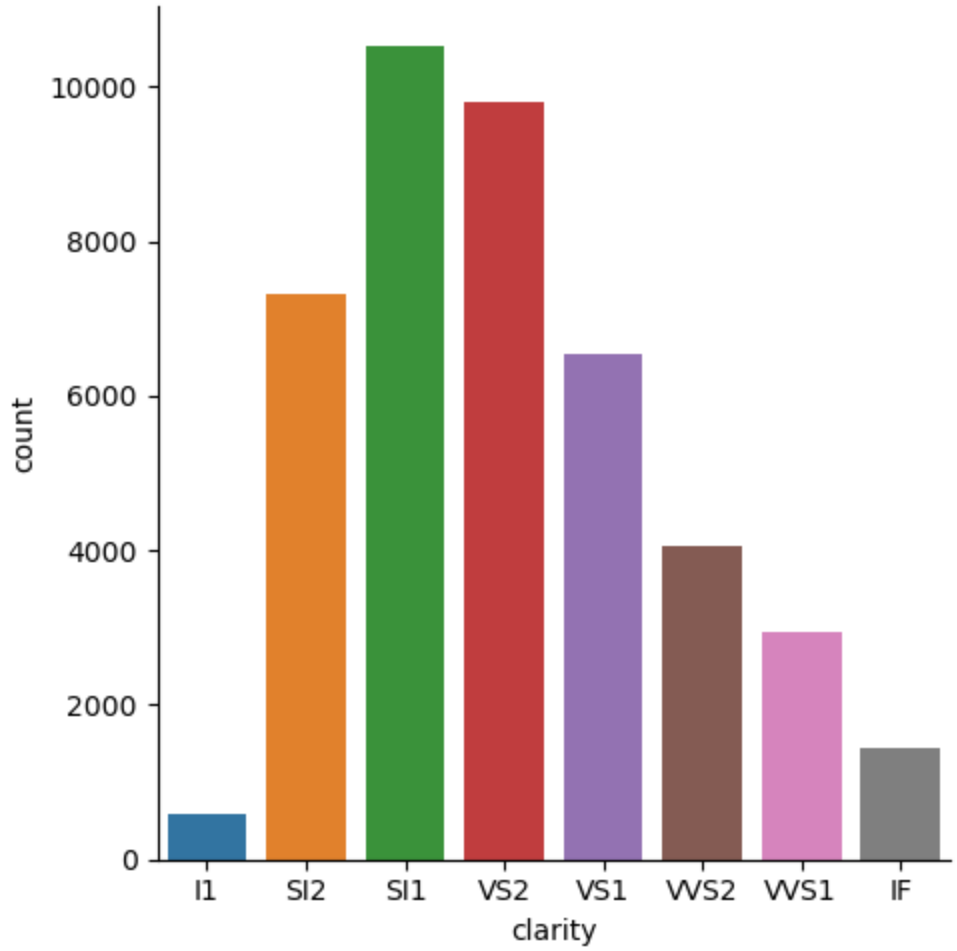
As can be seen, the color rating with the highest element is G and color rating with the least elements is J.

Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839



As can be seen, the amount for each cut quality increases as the quality of the cut increases.

Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839



As can be seen, the clarity rating that has the biggest number of elements is SI1 and the one with the lowest number of elements is I1.

Q7

In this part of the project, we are given two methods to determine the importance of the features in our dataset. These are Mutual Information and F-regression. Let us use these methods to determine the importance of the features in the dataset.

The features of our data in order are: ['cut', 'color', 'clarity', 'carat', 'depth', 'table', 'x', 'y', 'z']

When using mutual information, the importance scores of each feature is determined to be:
[0.06058756 0.13301784 0.21727135 1.63203697 0.03104078 0.03291637 1.39617896
1.41249869 1.3517641]

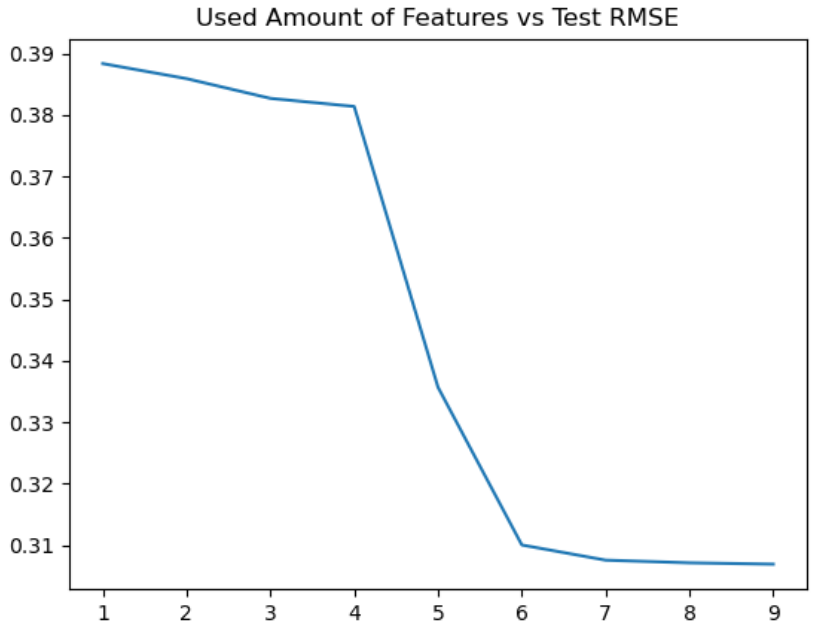
When using F-regression scores, the importance scores are calculated as follows:
[1.31765000e+02 1.30300353e+03 9.44679968e+02 2.43413123e+05
 1.34510791e+00 6.63671853e+02 1.54683575e+05 1.22678015e+05 1.19280878e+05]

Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

According to both methods, the most important feature that affects the price is carat and the feature that affects the price the least is depth.

We expected the test RMSE to improve as the features that were not related to the label were removed from the dataset. Removing one or two features that have importance scores close to depth should get rid of the features that are not closely related to price, which will allow the model to put more importance on the features that matter. So, removing 1 or 2 features was expected to improve the test RMSE results.

To test this hypothesis, we fit a simple linear regression model after performing feature selection to observe the trend in test rmse. The results are as follows:



From the results, we saw that removing features actually caused the test rmse to worsen. This could be related to the fact that there are not that many features in our data to begin with. So, dropping features causes a loss of information that makes the performance worse. Based on this observation, we decided not to include feature selection and use all of the available features for the remainder of the methods.

Linear Regression

Q8

For regular linear regression, the hypothesis is that there is a linear relationship between the predictor features and the target variable. When we utilize ridge regression, we enforce the weights of each feature to be small. This functions as a regularization technique in order to overcome overfitting. On top of forcing the coefficient of each feature to be small, lasso

regression also forces the weights of certain coefficients to be 0. Therefore, in lasso regression, the hypothesis is that some of the features have no effect on the target variable.

Q9

In order to find the best regularization scheme with the optimal penalization coefficient, we utilized the grid search algorithm. Here our parameter space consisted of the following:

Regularization Scheme	**Penalty**
None	None
Ridge	0.01,0.1,1,10
Lasso	0.01,0.1,1,10

Here, we evaluated each combination using 10 fold cross validation on the train set. We took the combination that gave the best average validation RMSE as our chosen approach.

After performing grid search, the best combination was found to be **ridge regression** with a penalty of **10**. This combination gave an average validation RMSE of **0.0928** and an average train RMSE of **0.0926.** As for each fold the trained model does not have access to do the validation data during training, the validation error was observed to be slightly higher than training error.

The best combination was trained on the entire train set and tested on the test data. The RMSE for the test set was found to be **0.306.**

Q10

In order to see the effect of feature scaling, we looked at the 10-fold cross validation performance of a linear regression model with different regularizations on both scaled and unscaled data. We then calculated the RMSE error on the test data for both cases.

For regular linear regression, the average training RMSE across folds was found to be **0.0926**, the average validation RMSE was **0.0928,** and the RMSE on the test data was found to be **0.3069** when no feature scaling was used. When feature scaling was used, the average training RMSE across folds was found to be **0.0926**, the average validation RMSE was **0.0928,** and the RMSE on the test data was found to be **0.3069.** From the results, we can infer that feature scaling does not change the results for regular linear regression. This is expected, as changing the scale of the features just changes the coefficients of the final result. For example, multiplying one feature by 2 divides the coefficient of that feature by 2 in the final model. So, there is no effect.

Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

To see the effect of feature scaling on ridge regression, we utilized a ridge regression model with alpha=100. For this model, when no standardization was applied the average training RMSE across folds was found to be **0.0956**, the average validation RMSE was **0.0958,** and the RMSE on the test data was found to be **0.3102.** When standardization was applied the average training RMSE across folds was found to be **0.0929**, the average validation RMSE was **0.0930,** and the RMSE on the test data was found to be **0.30708.** The difference in error is related to the fact that ridge regression forces the weights to be close to each other and treats each weight equally. However, when the scale of the features are different, it is not fair to penalize each weight in the same manner.

To see the effect of feature scaling on lasso regression, we utilized a lasso regression model with alpha=0.1. For this model, when no standardization was applied the average training RMSE across folds was found to be **0.197**, the average validation RMSE was **0.198,** and the RMSE on the test data was found to be **0.4407.** When standardization was applied the average training RMSE across folds was found to be **0.1451**, the average validation RMSE was **0.1451,** and the RMSE on the test data was found to be **0.3795.** As with ridge regression, lasso regression penalizes the weights in the same fashion, without attention to the scale of the features the weights correspond to. When the features are of different scales, this causes an unfair penalization across the weights. This explains the difference in performance.

Q11

The p-value is the result of testing the null hypothesis for each feature. In other words, it tests if the feature affects the target values. A low p value indicates that the hypothesis is wrong and the feature is related to the target variable. A high p value indicated that the feature is not significant for the target variable.

As such, we can infer the most important features by considering the features that have the lowest p values.

If we look at the p values after fitting a linear regression model to our data, we get the following: [0.999999999999969,1.698205454446913e-83,0.0,0.0,0.0,3.4568687513444134e-53,2.3303965701145696e-16,9.403174757440107e-123,0.27078510702616815]

For the features: ['cut', 'color', 'clarity', 'carat', 'depth', 'table', 'x', 'y', 'z']

From the scores, most important features are found to be clarity, carat and depth while the least important feature is cut.

Polynomial Regression

Q12

In order to determine the most salient features, we first needed to determine the degree of the polynomial that will be used. To do that, we run the grid search algorithm over possible polynomial degrees. The details of this process are explained in Q13 of the report. After this process, the best degree of polynomial was found to be 2.

Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

After transforming the dataset using a polynomial feature object of degree 2, the resulting set had a total of 55 features. To find the most salient among these, we can again refer to the F-regression scores. The scores for the generated set of features was as follows:

[nan 1.31765000e+02 1.30300353e+03 9.44679968e+02 2.43413123e+05 1.34510791e+00
6.63671853e+02 1.54683575e+05 1.22678015e+05 1.19280878e+05 8.28791217e+00
7.01912941e+00 3.38573749e+01 3.79653882e+00 1.36362503e+00 9.63953620e+00
2.00301123e+01 1.83975760e+01 1.25984235e+01 1.01837096e+02 4.66652849e+02
3.32482847e+03 1.17064696e+01 4.91417951e+01 1.91878532e+03 1.85175520e+03
1.77952860e+03 1.42130047e+02 7.79517841e+02 1.00856822e+01 5.03459876e+00
7.38740306e+01 7.59047564e+01 6.30709921e+01 2.45993634e+04 1.19557924e+01
3.39979015e+02 2.46474625e+04 1.97858855e+04 2.20785333e+04 3.72655963e+01
1.34017902e+00 6.27546549e+01 6.70846776e+01 5.37780603e+00 2.20322741e-01
1.42416775e+02 1.37140140e+02 1.06263933e+02 1.81925268e+04 1.51698244e+04
1.67455559e+04 1.85228716e+02 5.38378468e+03 2.54194208e+02]

From the scores, the most important feature is again found to be carat. The remaining best features are x, y and z. This is expected as the multiplication of x, y and z gives the volume of the diamond, which is related to carat. In terms of combinations, the most relevant one is the combination of carat and x and carat^2. These are all expected since carat is the most relevant feature and combining this with other relevant features results in alternative features that are also relevant.

Q13

In order to determine the best degree of the polynomial, we performed grid search. The space we searched was as follows:

Polynomial Degree	2, 3, 4, 5

After changing the dataset based on these polynomial degrees, we fit a linear regression model to perform predictions.

After the grid search, the best results were obtained with a polynomial degree of 2. Here, the mean cross validation train RMSE was found to be **0.03.** The mean validation RMSE was **0.6697.** Again, as the model is trained on the train portions during each fold, there is a difference between the train and validation RMSEs. The large validation RMSE was attributed to the fact that there was a high error observed for one of the folds. The test RMSE for this setup was found to be **0.195.**

As seen in the cross validation results, higher degree polynomials lead to worse validation performance. This is because higher degree polynomials tend to overfit to the train dataset, which results in very low train RMSEs but high validation RMSEs. So, we can say that a higher order polynomial implies overfitting to the training data.

Q14

Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

For this part of the experiment, we were asked to generate combined features. In our implementation, we multiplied x,y and z features to form another feature we called volume.

This approach made sense because a high volume diamond also implies a heavier diamond, which implies a higher price. Instead of expecting the model to learn the relationship between x, y, z and the price separately, combining these features allows the model to more easily learn the relationship. This is expected to improve the performance of the model.

When the volume feature was added to the data, the mean 10-fold training RMSE was **0.0344,** the mean validation RMSE was **16.60.** This was unexpected as the mean validation RMSE is higher than when volume was not added. Further investigation of the results showed that for one of the folds, validation RMSE was 158, which explained the large average error. This implies that the model overfit during that fold. However, the mean test RMSE dropped to **0.188** with this approach.

Neural Network

Q15

The reason why neural networks perform better than linear regression is because they are capable of learning non-linear relationships between the input features and target variables. While linear regression can only linearly relate the features to the labels, neural networks can learn more complicated relationships.

Q16

In order to find good values for the amount of layers, the amount of neurons in each layer and the weight decay parameter, we again performed grid search. The hyperparameter space that we searched was as follows:

Hidden layer sizes	(50),(100),(150),(50,50),(50,100),(100,150),(50,150)
Weight decay	0.00001, 0.0001, 0.001, 0.01, 0.1

The best parameter combination had the least mean validation RMSE across the 10 folds. The best combination was found to be a neural network with hidden layer sizes (50,100) and a weight decay parameter of 0.001. For this combination, the mean training RMSE was **0.018** and the mean validation RMSE was **0.024.** As explained before, since the model performs updates according to the training data, the RMSE for training is lower. The lack of significant difference between mean train and validation RMSEs shows that there was no overfitting. The test RMSE for this setup was found to be **0.1374**.

Q17

The activation function of the last layer should be None. As we are doing regression, we want the output of our model to be a continuous value. As we know, activation functions generally tend to saturate to some value for high and low inputs, so using an activation value would not be

appropriate. Instead, we should treat the output of the last neuron as it is and try to make this value as close as possible to the labels.

Q18

There are two main risks if we increase the depth too far. The first and the most obvious one is overfitting. As we add layers, the amount of parameters of the model increases. To train a more complicated model, we need more training data. If we do not have enough train data, the model will overfit.

The second risk is the vanishing or exploding gradients problem. Neural networks are updated through a process called backpropagation. Here, the error gradient is passed to the first layers from later layers. During this passage, the gradient is multiplied by the weights of the later layers. When we have too many layers, the information propagated to the first layers can get smaller and smaller as it is multiplied by the weights. Conversely, it can also get too large. This causes suboptimal weight updates for the first layers and leads to poor training.

Random Forest

Q19

In order to find the optimal number of hyperparameters, we again resorted to using the grid search algorithm. Here, the parameter space that we tried was as follows:

Number of Estimators	50,100,150,200
Max Features	3,5,7,9
Max Depth	10,100,500,1000,2500, None

The combination that had the lowest mean validation RMSE was deemed to be the best model combination. After the search process, this combination was found to have 150 estimators, with a maximum of 5 features considered at each split and a maximum depth of 500. For this combination, the average training error was found to be **0.0025** and the average validation error was found to be **0.018**. As explained earlier, as the model sees the training data, the average error on the train set is less than the average error on the validation set. For this setup, the test RMSE was found to be **0.1338**.

For the random forest model, we are also asked to measure the Out-of-Bag(OOB) error. As can also be understood from the code in the given link, the OOB error is simply the average error on the unsampled data. Each tree in a random forest model is built using a random subsample of the entire training data. To calculate the OOB error, we ask each tree to predict a value for the samples it has not seen. We then average the predictions for each data entry to get the OOB results. To calculate the error, we look at the R2 score between the correct labels for these samples and the OOB scores.

Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

The R^2 score is the proportion of the variation in the dependent variable that is predictable from the independent variable. In our case, it simply means how much of the variation of the true labels is related to the OOB scores. A high R^2 score implies that the OOB predictions are related to the labels while a low score indicates that they are not related.

To measure the OOB error, we trained a random forest model on our set and set the oob_score parameter to True. As the maximum OOB score is 1, we subtracted the returned value from 1 to get the error. This revealed the error to be **0.01829**.

Overall, it was found that limiting the maximum amount of features to 5 or 7 improved performance. Furthermore, it appears that using trees in the range of 150 to 200 proved to be effective. Keeping the maximum depth around 100 or 500 also proved to lead to better results.

The features number of estimators and max depth have a regularization effect on the model. Having small values for these features forces the model to have low complexity, which can be utilized to overcome overfitting. Alternatively, setting large values to these parameters results in a complex model, which can be thought as a model without regularization. A similar aspect can also be said about the maximum number of features. When this value is high, the model is free to always rely on the most defining feature in the data. However, when it is low, we force the model to use other features as well, which decreases the likelihood of overfitting by increasing the randomness. Examining the grid search results also shows this to be true, as the difference between mean train and validation RMSEs slightly increase when max_feature is set high.

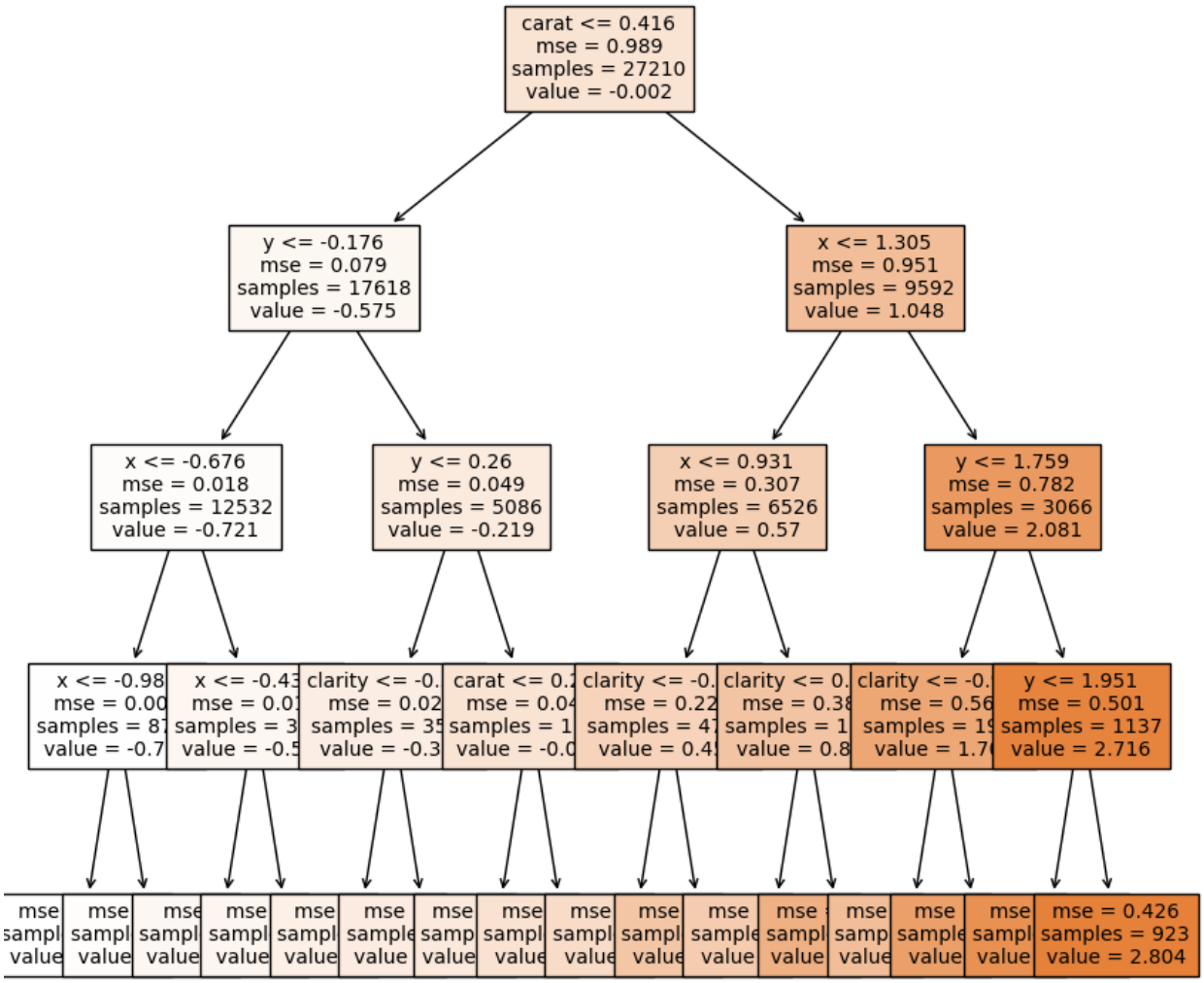
Q20

Random forests work well because they create a large number of uncorrelated models. Each tree in a random forest is built using a random subsection of the available data. Furthermore, each split is made based on a random subset of the available feature space. These add randomness to the trees. Combining these trees results in a large ensemble where each model is uncorrelated to each other. This greatly reduces the risk of overfitting and makes the model more robust compared to other approaches that rely on a single predictor. This is why random forests tend to perform better when compared to most other algorithms.

Q21

In the question, we are asked to randomly select a tree from our random forest model and visualize it. As the minimum tree depth in our optimized random forest model was 31, we decided to create another random forest model with a max depth of 4. To do so, we trained a random forest model with a max depth of 4, 150 estimators and max of 5 features at each split. Then, we randomly picked a tree to visualize. The results are follows:

Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839



As can be seen, the feature selected for branching at the root node is carat. Then, this tree utilizes features y, x and clarity. Among these features, we can infer that the feature closer to the root is more important.

In part 3.2.1, by examining the p values we found the most important features to be carat, x, y and z. Therefore, as this tree also makes use of these features, we can state that the important features match our findings.

LightGBM

Q22

After reading the documentation of the LightGBM algorithm and performing experiments on the dataset, the following search space was created to perform bayesian optimization.

Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

Parameter Name	**Search Range**
reg_sqrt	[True, False]
Learning rate	Real(0.01, 1.0, 'log-uniform')
Number of estimators	Integer(30, 800)
Number of leaves	Integer(2, 512)
Max Depth	Integer(-1, 256)
Subsample	Real(0.01, 1.0, 'uniform')
Subsample Frequency	Integer(1, 10)
colsample_bytree	Real(0.01, 1.0, 'uniform')
reg_lambda	Real(1e-9, 100.0, 'log-uniform')
reg_alpha	Real(1e-9, 100.0, 'log-uniform')

Q23

After performing bayesian optimization for 300 iterations using the parameter space defined on Q22, the best parameter combination was found to be as follows:

Parameter Name	**Best Value**
reg_sqrt	False
Learning rate	0.025
Number of estimators	800
Number of leaves	123
Max Depth	63
Subsample	1
Subsample Frequency	9
colsample_bytree	0.9145

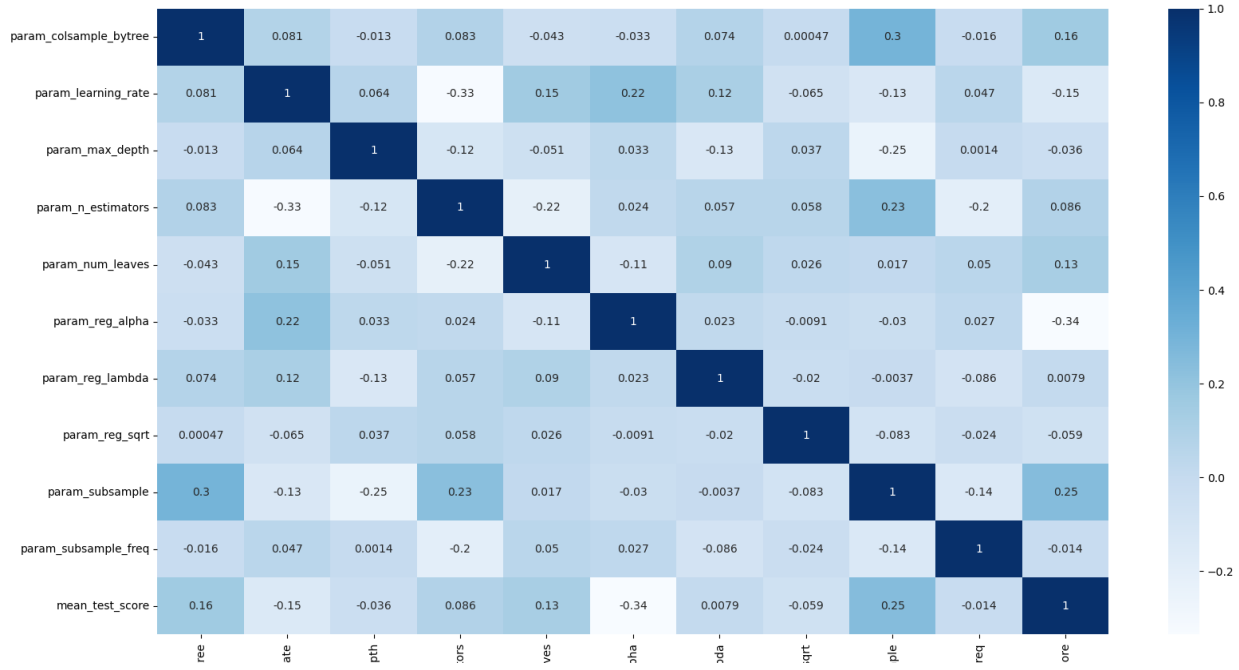
Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

reg_lambda	5.97e-06
reg_alpha	1.8665

This parameter combination achieved an average train RMSE of **0.095** and an average validation RMSE of **0.133**. On the test set, it had an RMSE of **0.1314**.

Q24

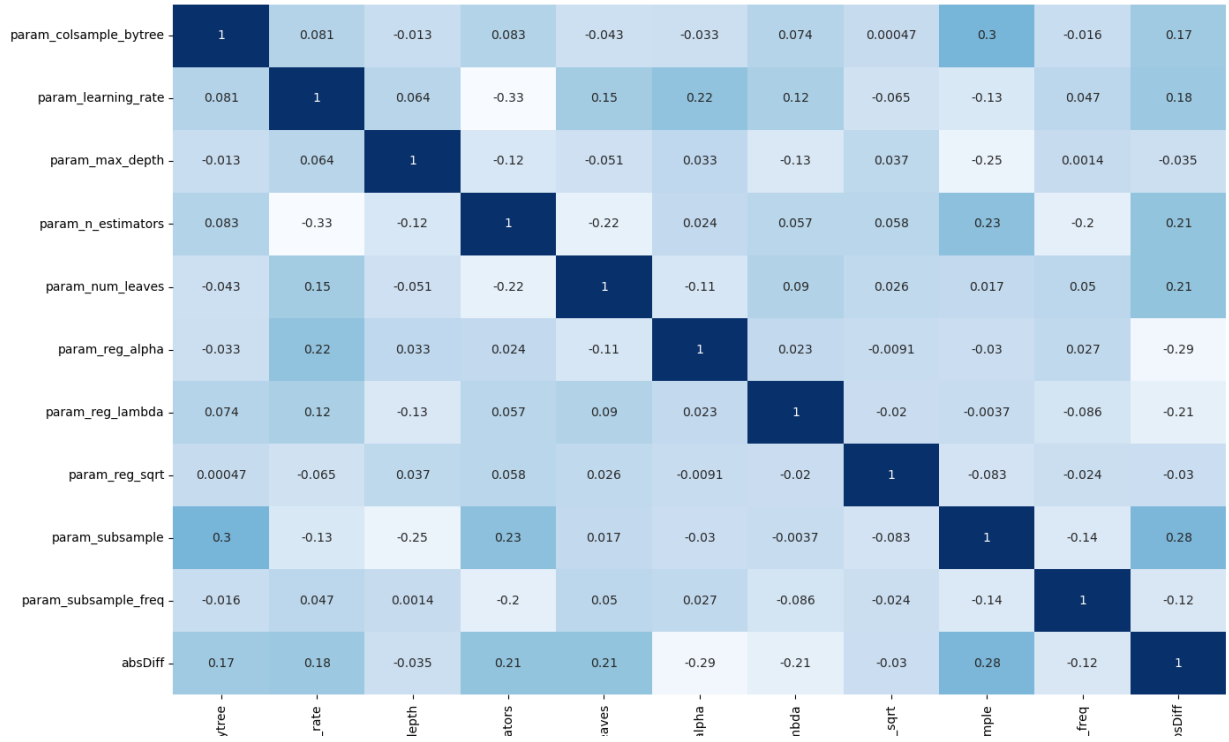
To see the effect of different parameters on performance, we can use the correlation matrix between the mean test error and the changed parameters.



From the graph, it can be seen that the parameters that affect performance the most are subsample, number of leaves and colsample_bytree. These seem to have a direct relationship with performance. Furthermore, we can see that there is also a strong inverse relationship with regularization alpha and performance. This is related to the fact that increasing the regularization parameter too much causes the model to be simple, which hinders performance.

To see the effect of the parameters on regularization, let us compare the absolute difference between mean train and validation RMSEs based on changed parameters.

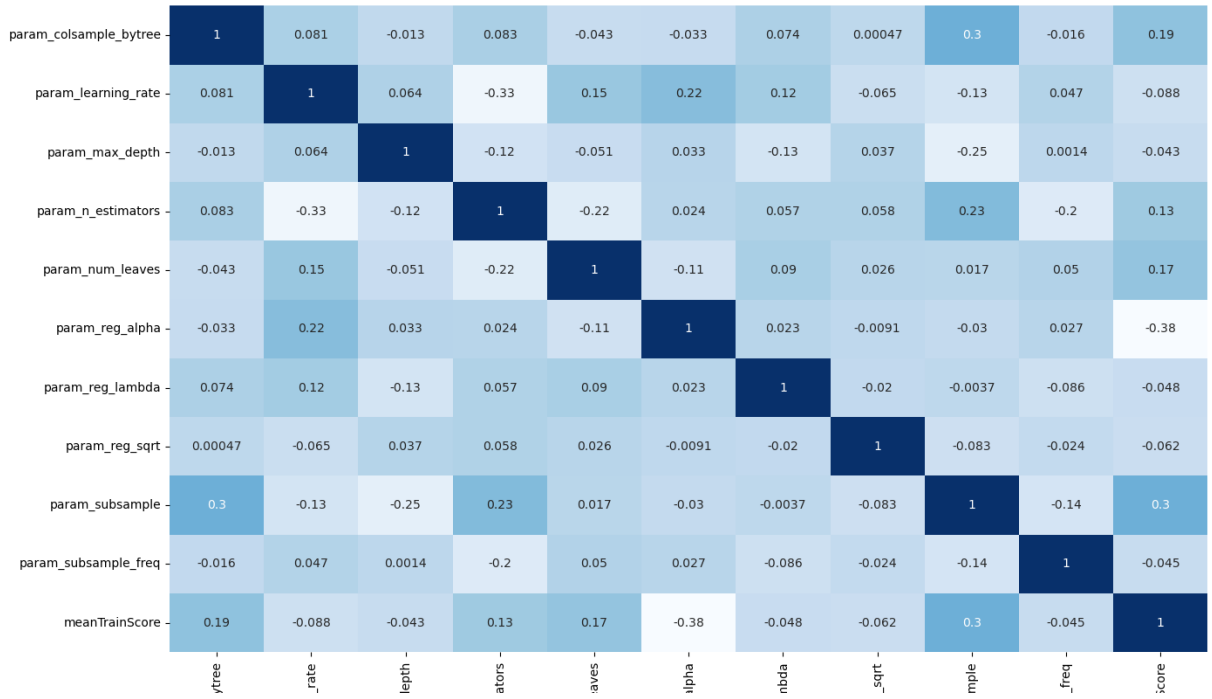
Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839



Certain trends can be seen here. For example, as the number of estimators, number of leaves and subsample parameters increase, the absolute difference between the validation and train scorers increase. This is expected as an increase in these parameters lead to overfitting, so decreasing these has a regularization effect. Moreover, as can be seen from the correlation map, the regularization parameters alpha and lambda have an inverse relation with mean absolute difference. As increasing these parameters means improving the strength of regularization,

Finally , let us see the effect on fitting efficiency by examining the relationship between the mean train score and the parameters of the model.

Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839



From the heatmap, we can tell that as the subsample ratio increases, the mean train score increases. This is expected since when subsample is 1, each tree is fitted to the entire training data, resulting in a better fit to the train data. Furthermore, we can see that the regularization parameter inversely affects the fitting efficiency as it has a negative correlation with training data performance. This is also expected as setting the regularization too high decreases the complexity of the model, resulting in a poorer fit.

CatBoost

After reading the documents and performing experiments, the parameter space to search for catboost was determined to be as follows:

Parameter Name	**Search Range**
iterations	Integer(10, 1000)
Learning rate	Real(0.01, 1.0, 'log-uniform')
depth	Integer(1, 12)
L2 leaf regularization	Integer(2, 100)
Random strength	Real(1e-9, 10, 'log-uniform')
Bagging temperature	Real(0.0, 1.0)

Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

Q22

After performing Bayesian optimization for 600 iterations, the best parameters were found to be as follows:

Parameter Name	**Optimal Value**
iterations	775
Learning rate	0.05
depth	9
L2 leaf regularization	2
Random strength	1e-9
Bagging temperature	1

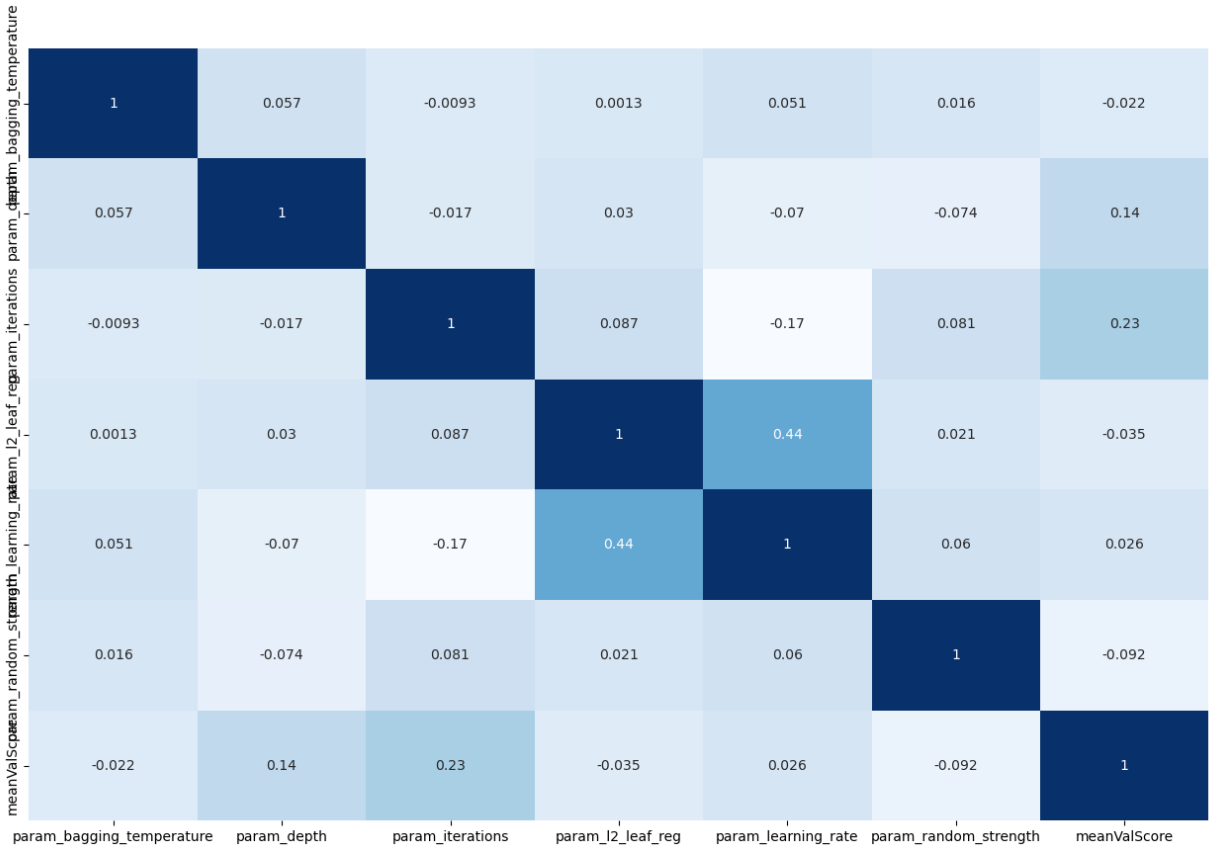
For this hyperparameter combination, the average train RMSE was found to be **0.097** and average validation RMSE was found to be **0.1317**. The test RMSE for this case was found to be **0.1298**.

Q23

To determine the effect of hyperparameters on the optimization results, we can again refer to the correlation matrix.

To determine the effect on performance, we can examine the correlation matrix between mean validation performance and the parameters.

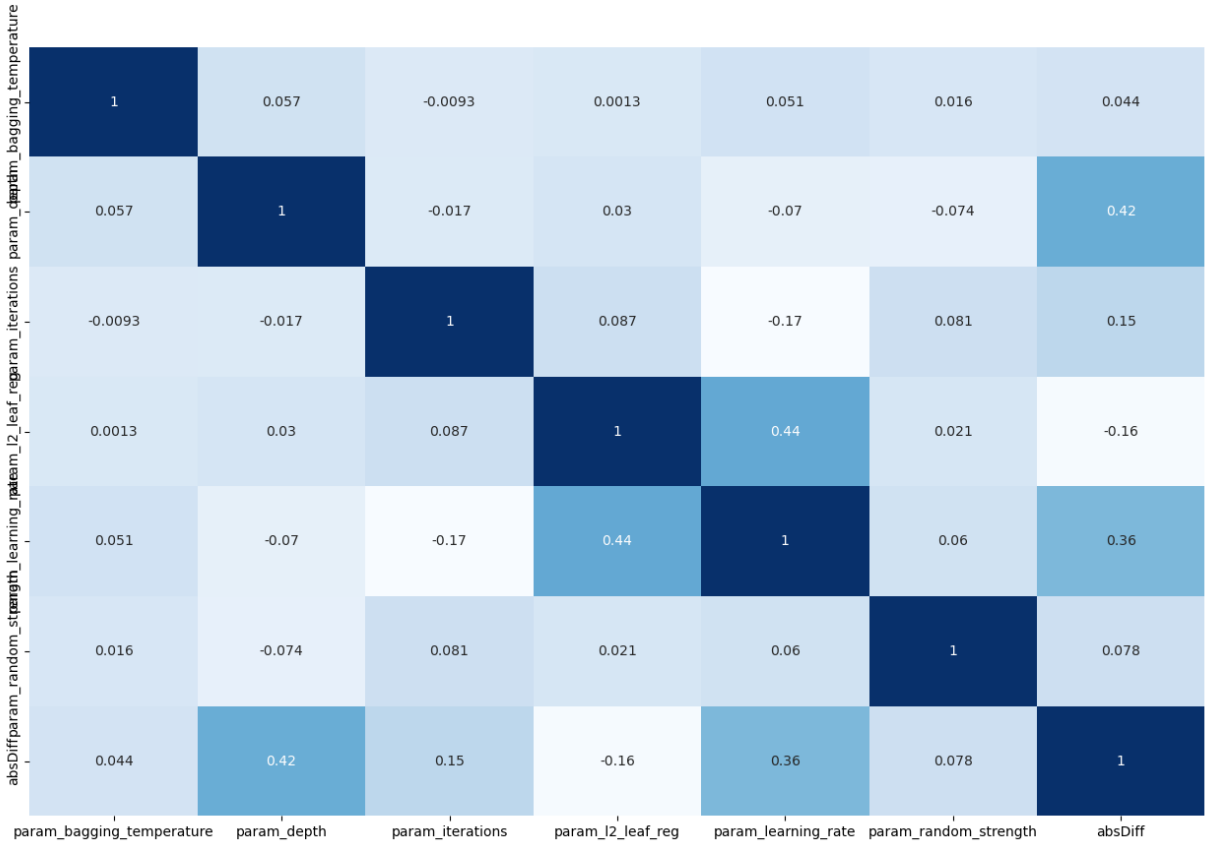
Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839



By examining the matrix, we can determine the parameter that most affects the performance is iterations. This is expected, since as we increase the number of trees, we get a more complex model that can provide a better fit. A similar remark can also be made about depth.

To see the effect of parameters on regularization, let us again examine how each feature affects the absolute difference between mean validation and train RMSEs.

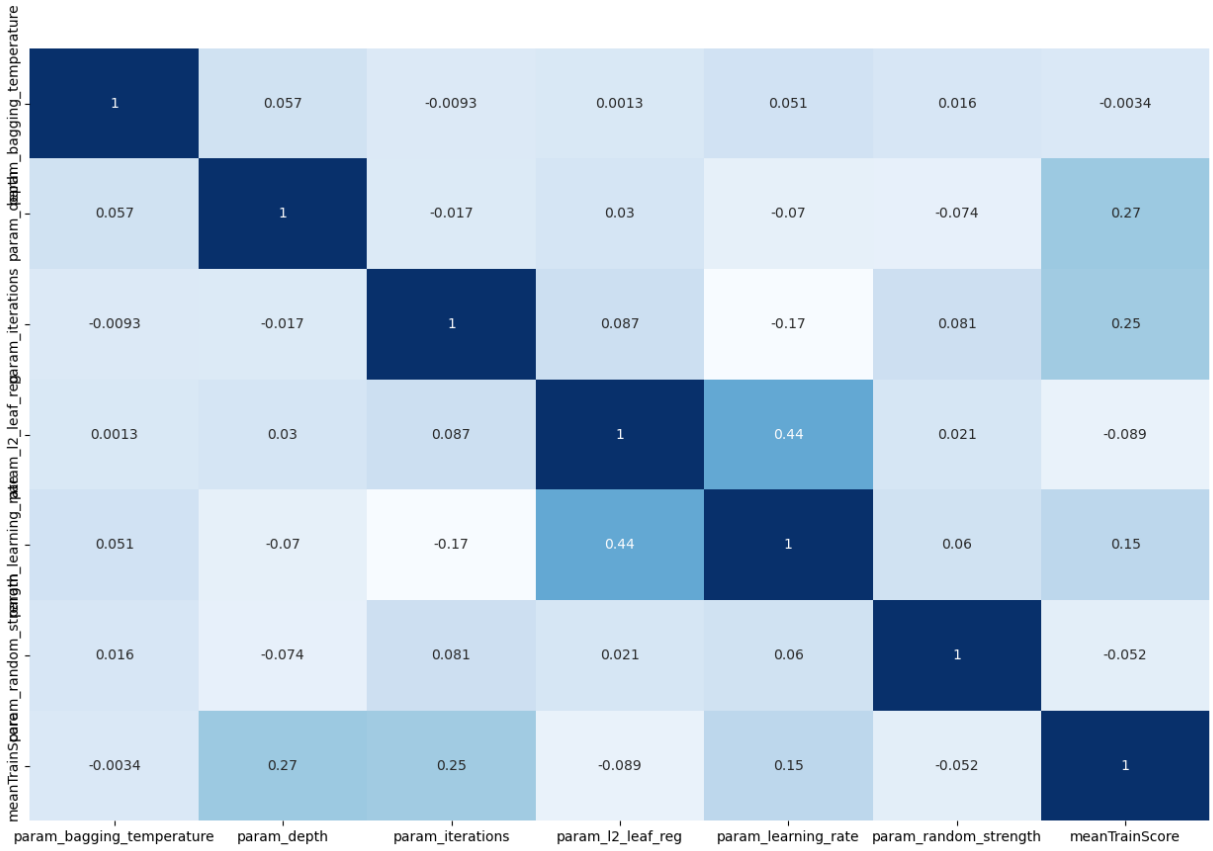
Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839



As can be seen the parameter that most affects the difference is depth. This is expected, since as depth increases, the complexity of the model also increases, which can easily lead to overfitting. Therefore, putting a limit on depth can have a regularizing effect. Furthermore, as expected, we can see that the L2 leaf regularization has a regularizing effect as an increase in this value leads to a decrease in the absolute difference.

Finally, to see the effect on the efficiency of the fit, we can examine the effect of the parameters on the mean training performance.

Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839



From the matrix, we can determine that the two parameters that affect the fighting efficiency the most are depth and iterations. An increase in these parameters leads to a more complex model, it makes it easier to find a model that performs well on the training set. Therefore, increasing these parameters lead to a more efficient fit for the model.

Dataset 2: Gas Turbine CO and NOx Emission Data Set

Now, let us perform regression analysis on Dataset 2 in order to predict CO emissions.

Handling Categorical Features and Standardization

Q1

Before performing training, we need a way to deal with categorical features. In this dataset, the only categorical feature is year, which goes from 2011 to 2015. As this is also a feature that has a specific order, we used OrdinalEncoder to deal with it. Here,the value 0 was given to the year 2011 and 4 was given to the year 2015. The remaining yars were given values based on their order.

Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

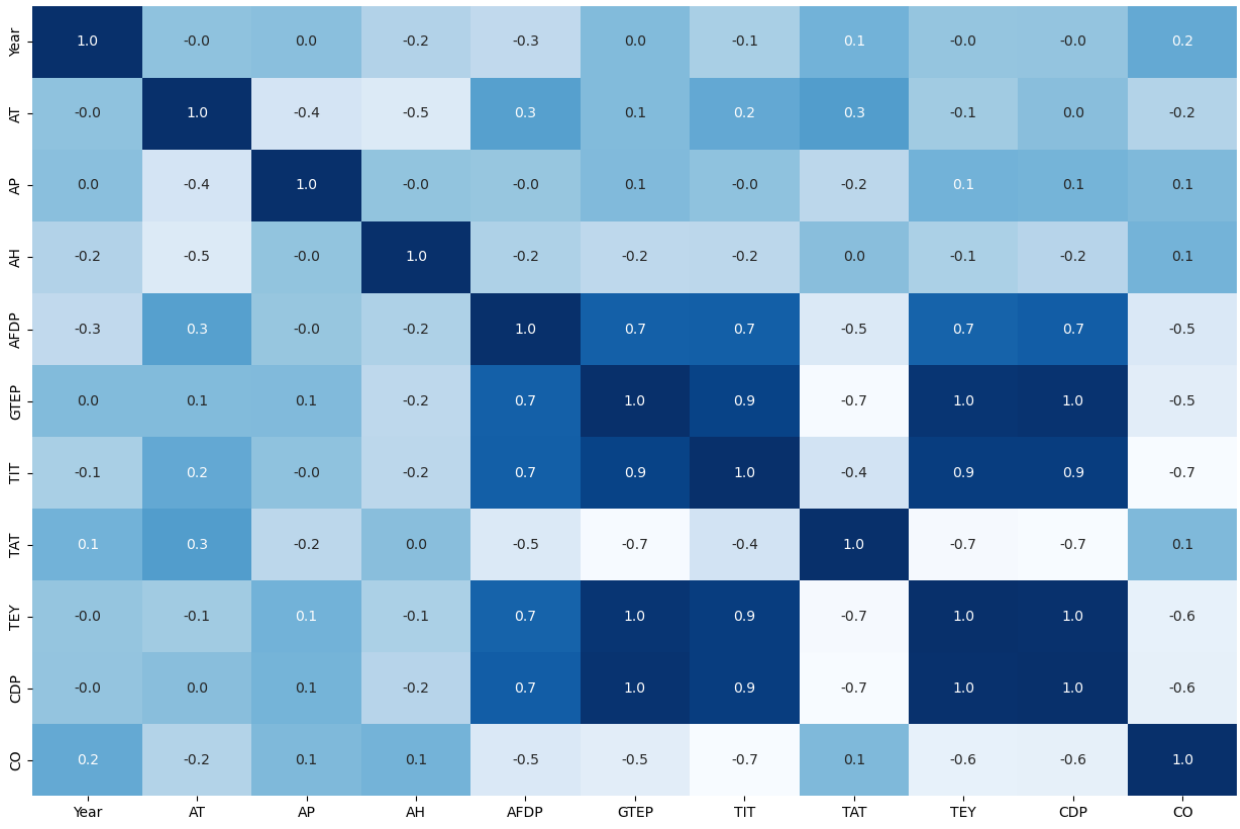
After this step, we standardized all the features to have 0 mean and 1 variance using StandardScaler. As before, we also normalized the label that we are trying to predict, which was chosen to be CO.

After normalization, our features data had 10 columns each with 0 mean and 1 variance. Our label consisted of a single column that also had 0 mean and 1 variance.

Data Inspection

Q2

To see the effect each feature has on our label, we can examine the Pearson correlation matrix between the features and the labels. The result for this dataset is as follows:

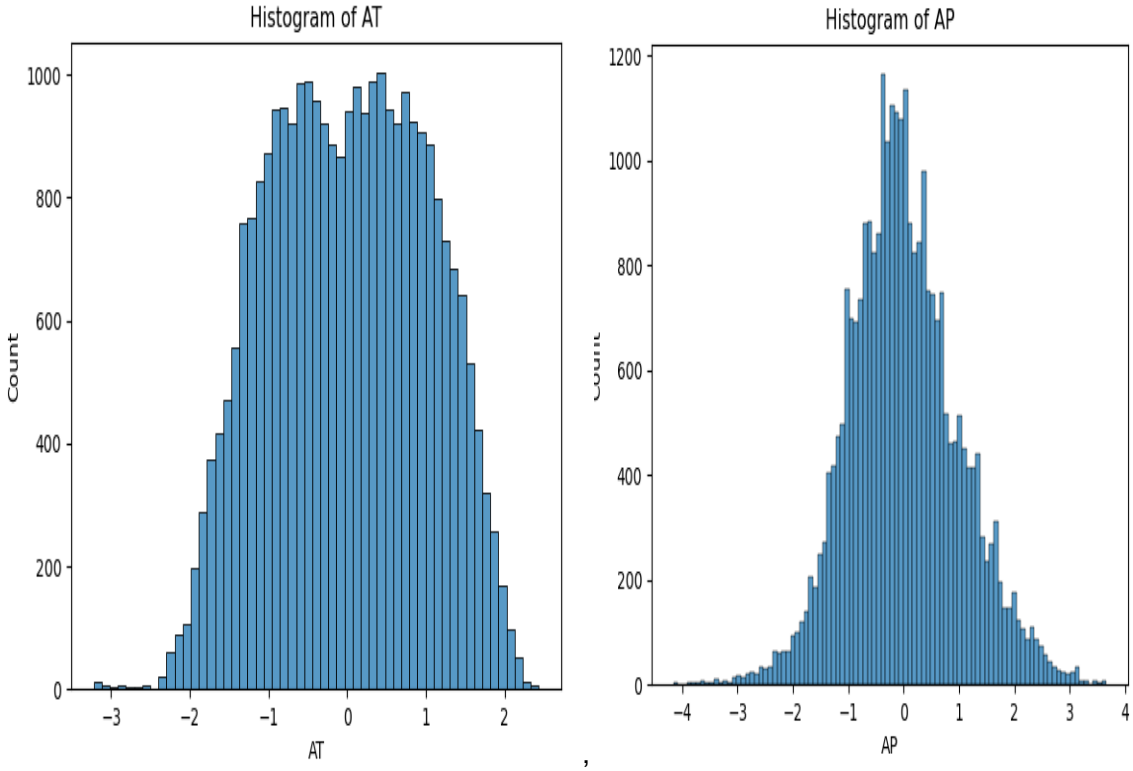


As can be seen the features TIT, TEY and CDP have the highest correlation with the label CO. These features and the label are inversely correlated. If we look at the paper associated with this dataset, we learn that TIT is Turbine Inlet Temperature, TEY is Turbine Energy Yield and CDP is Compressor Discharge Pressure. By examining the correlation matrix, we can determine that decreasing these values results in an increase in CO.

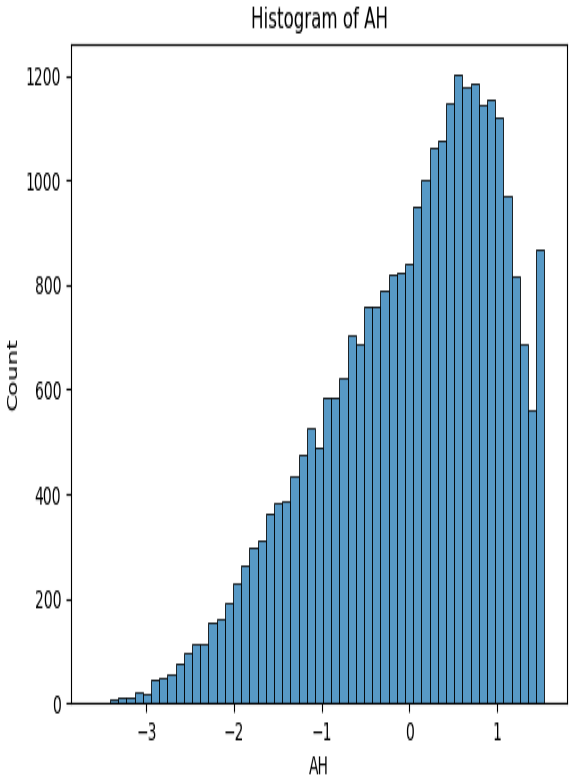
Q3

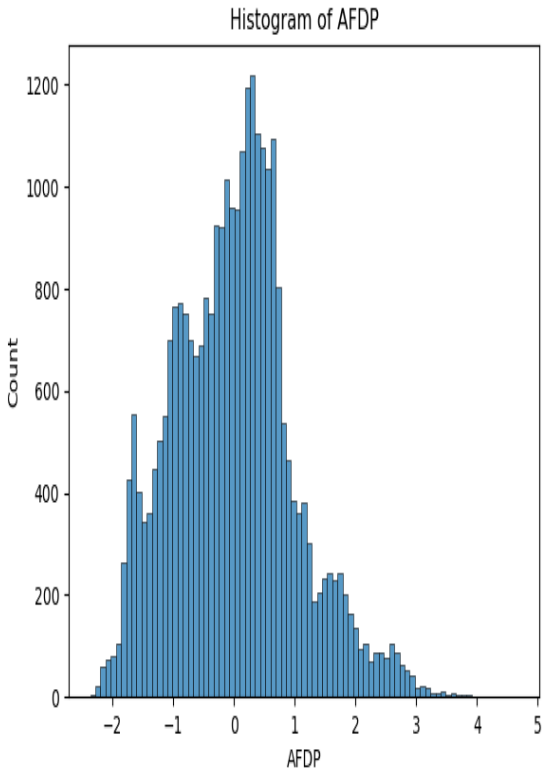
Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

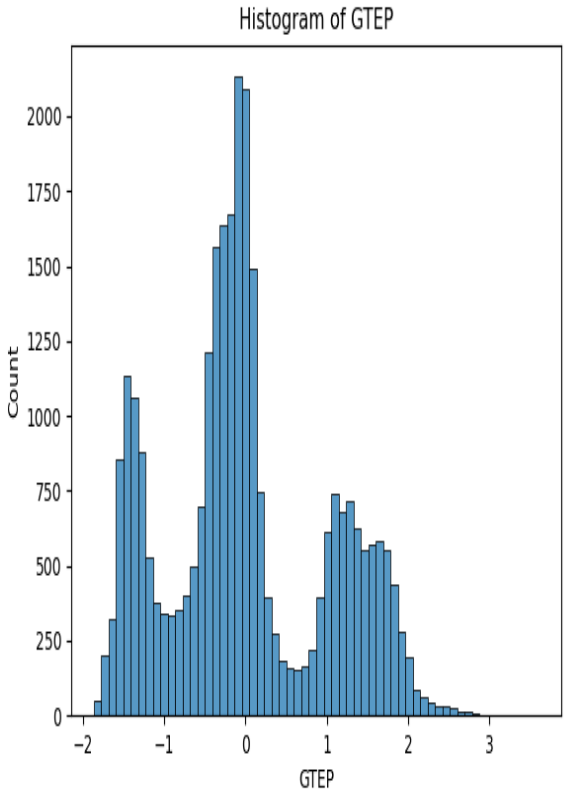
The histogram of the numerical features after standardizatipn are as follows:

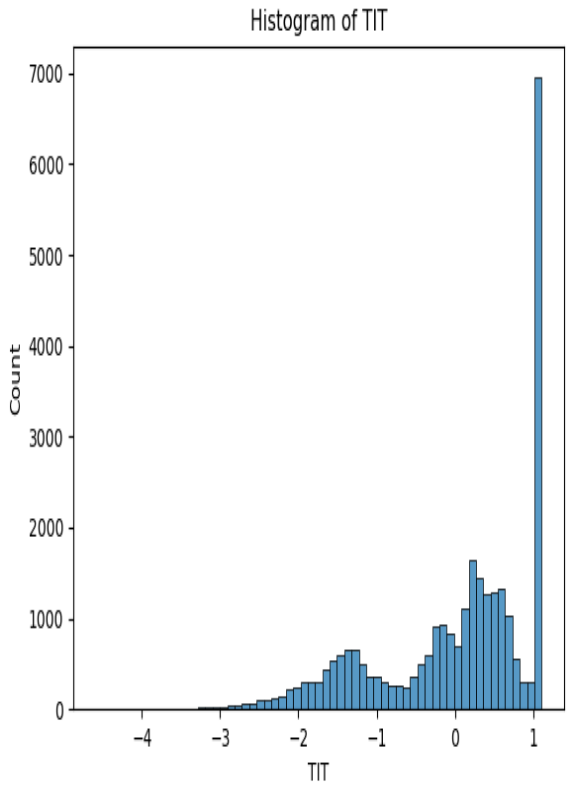


Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

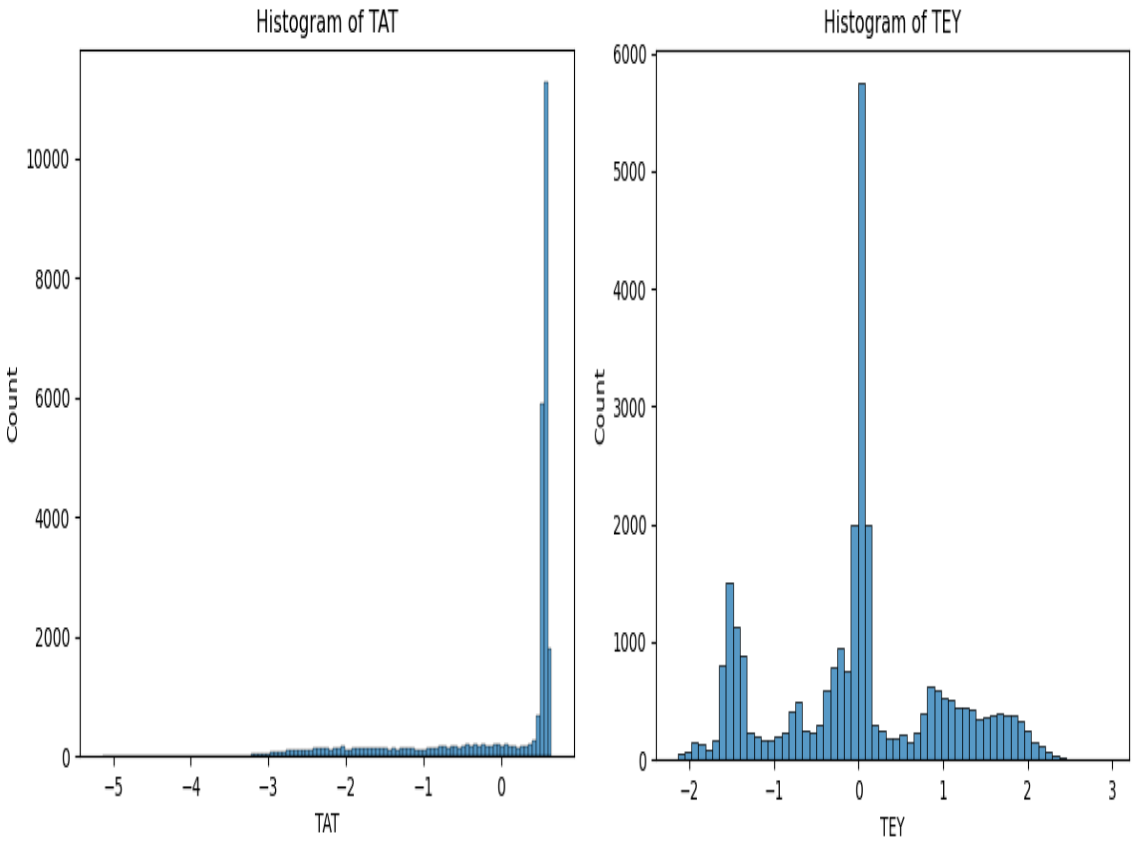


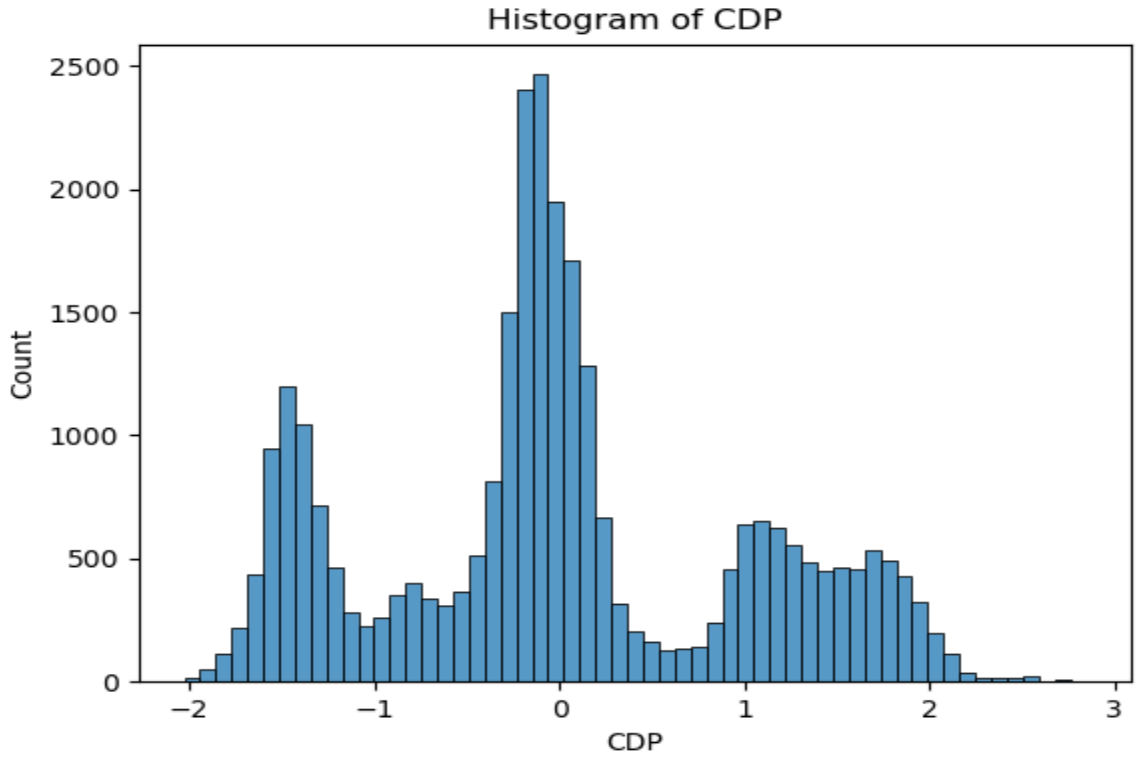






Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839



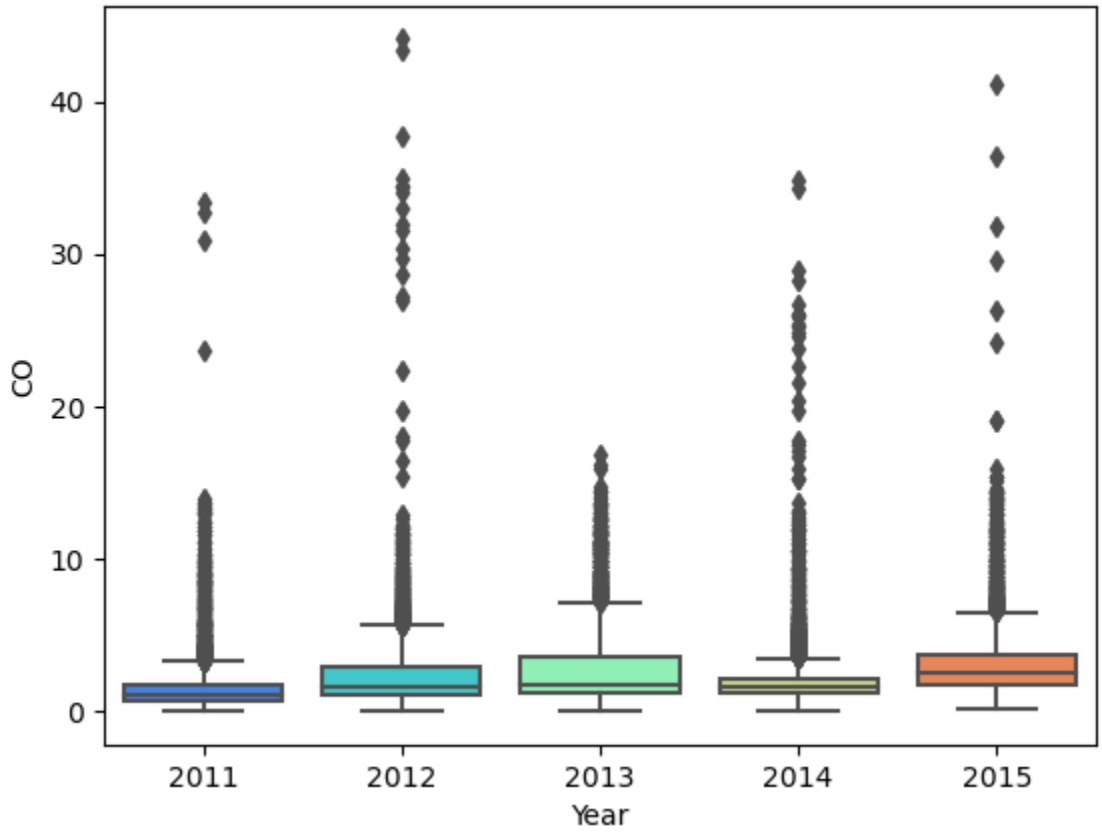


Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

As mentioned before, to reduce the skewness in some numerical features, we can try transformations such as square root transformation, log transform, reciprocal transformation or the Box Cox transform.

Q4

As there is only 1 categorical feature, let us plot the box plot of the year feature.

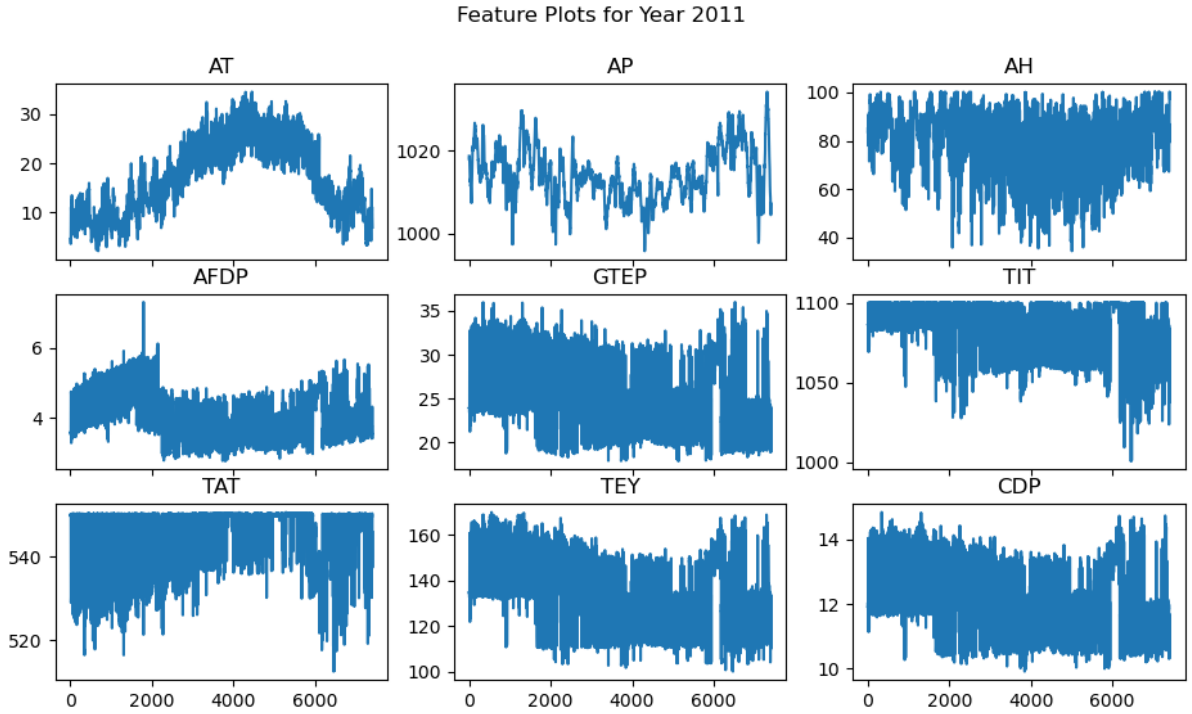


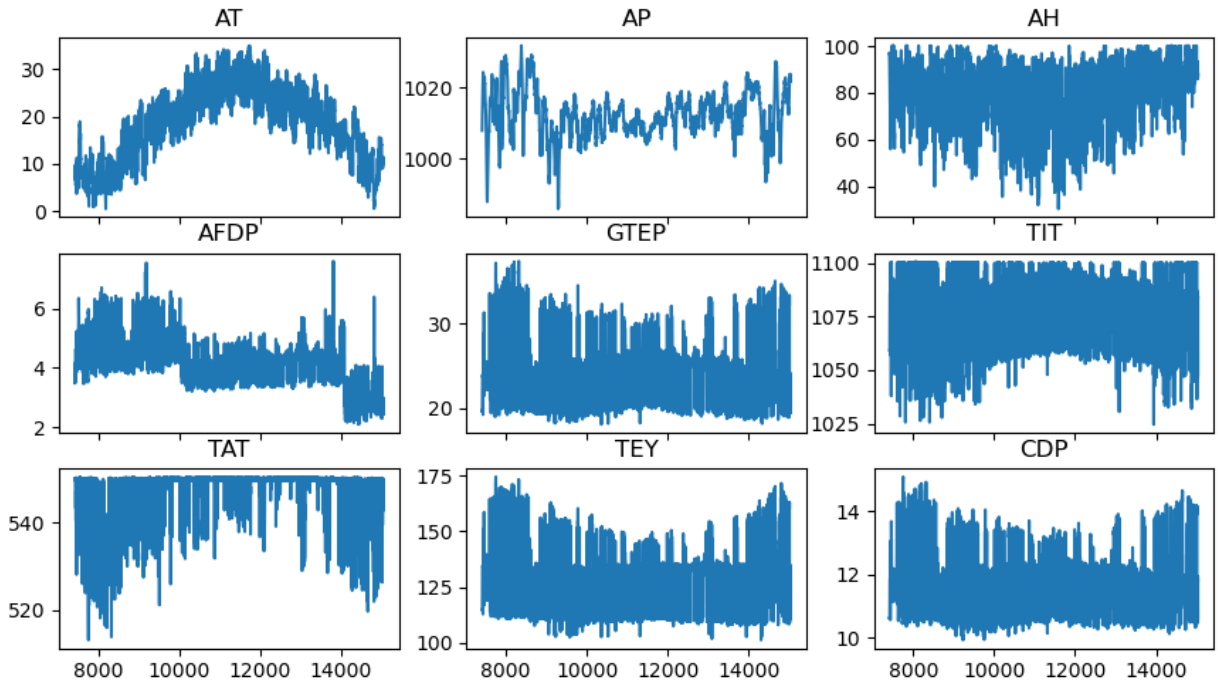
From the box plot, we can determine that there is an increasing trend in CO between 2011 and 2013. This trend is broken in 2014, where there is a reduction in overall CO emissions. In 2015, CO emissions rise again, with the highest mean CO emission observed in this year.

Q6

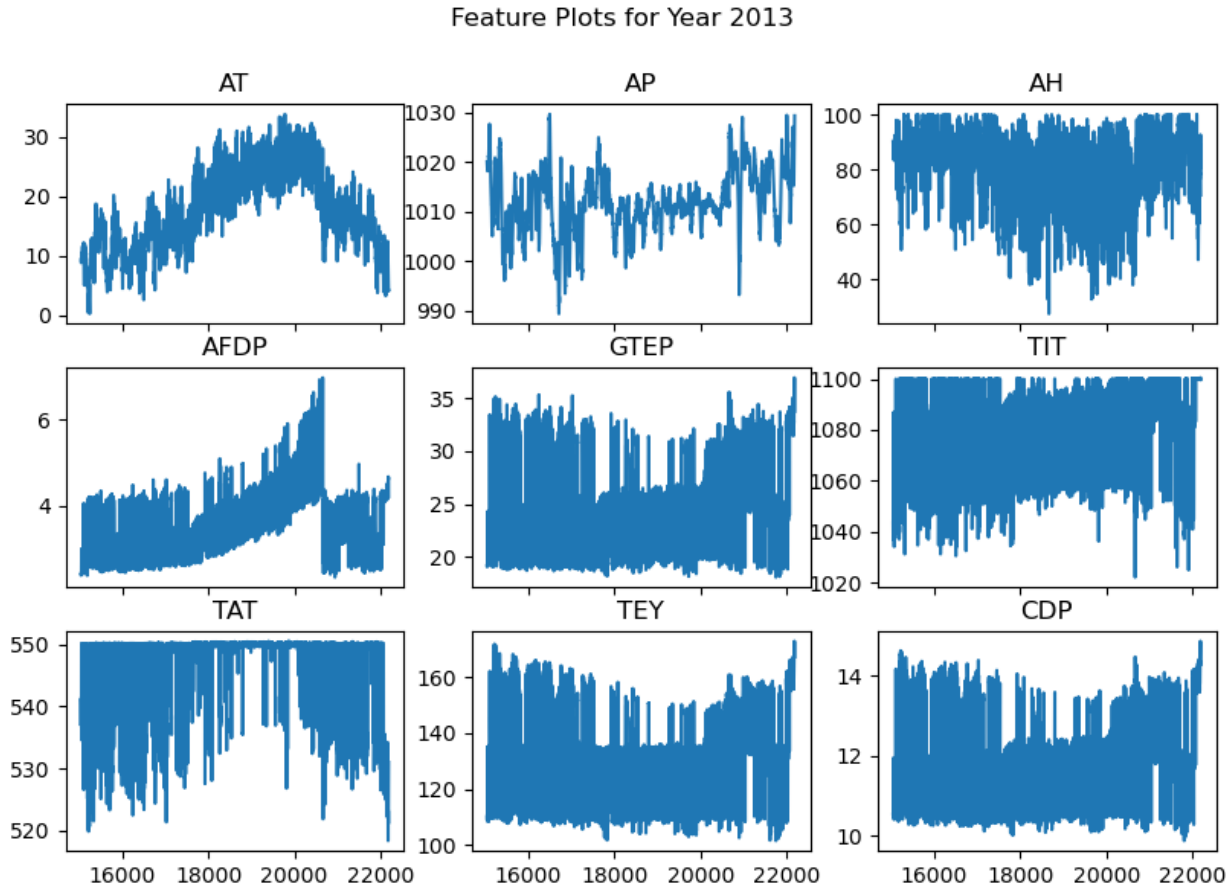
The change in all of the features for the available years can be seen below.

Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

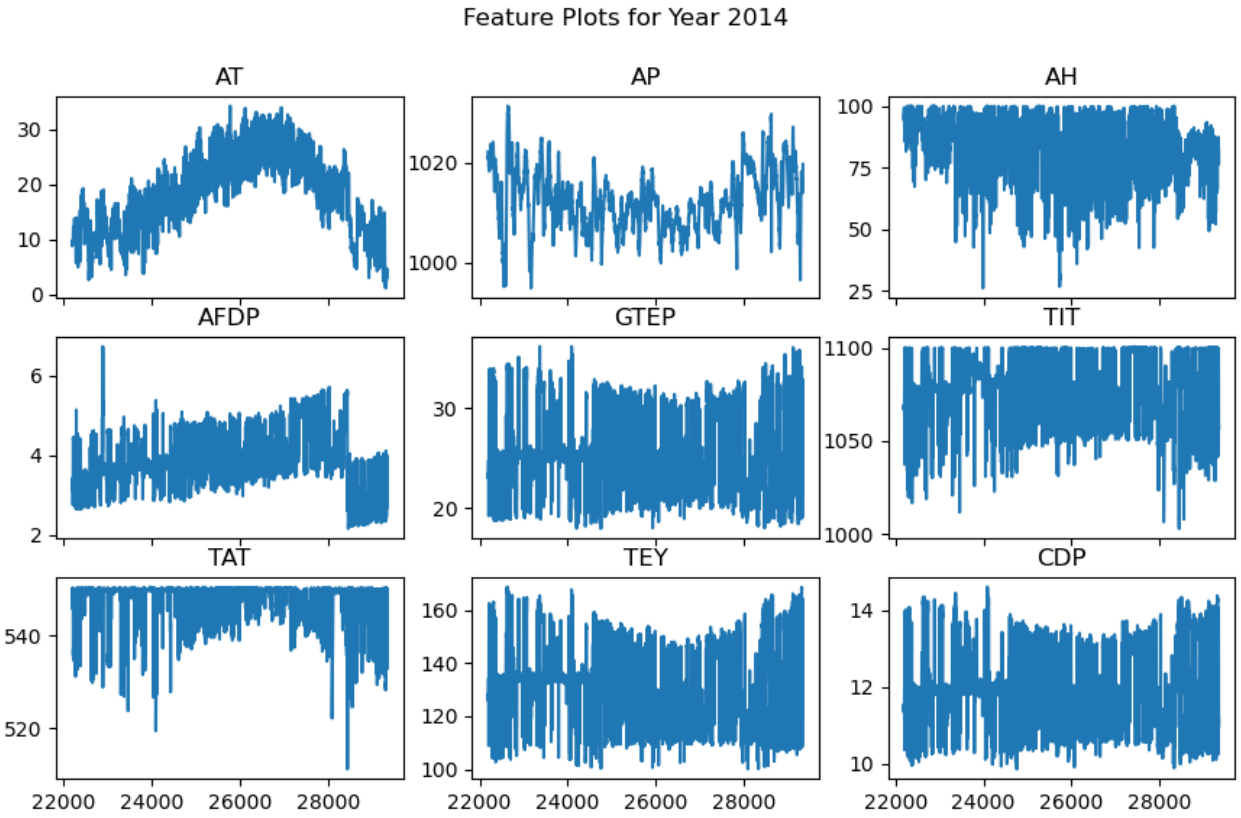




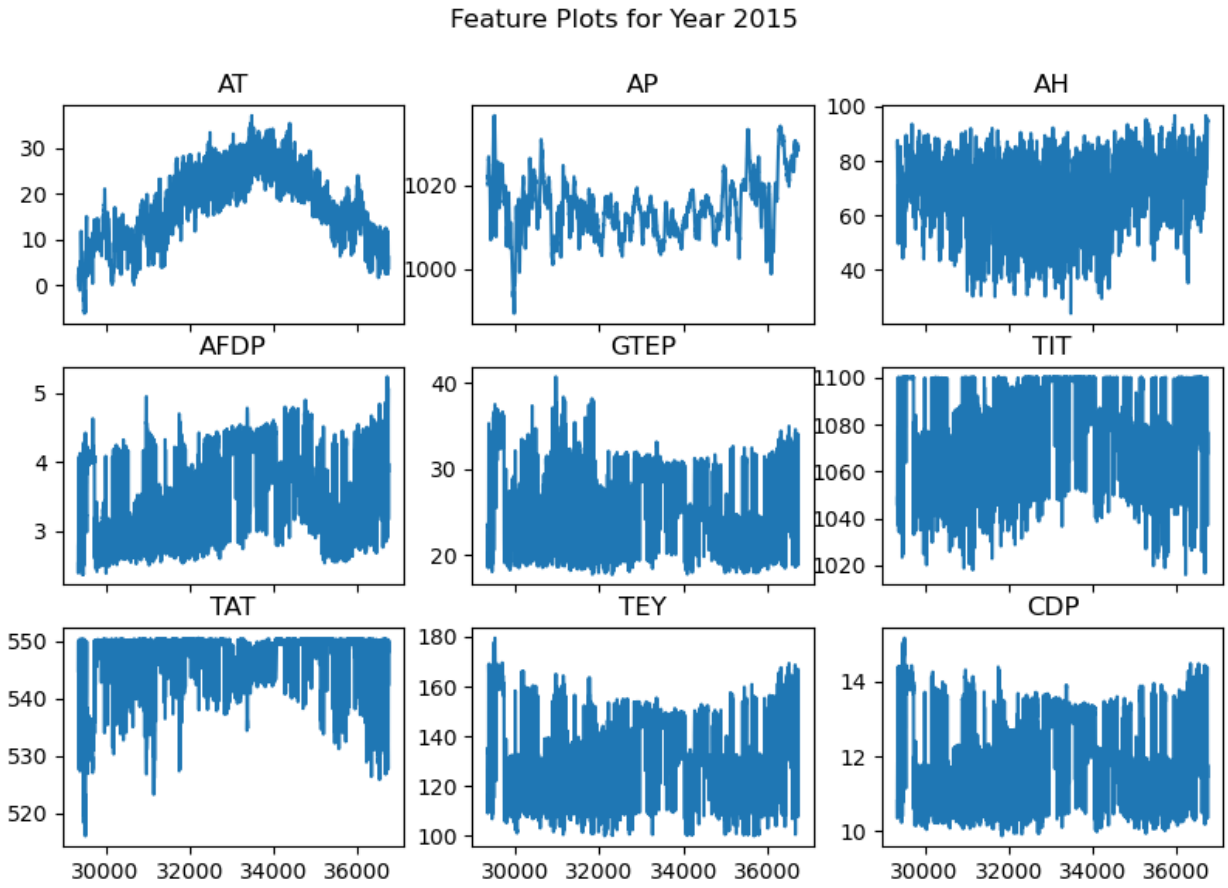
Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839



Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839



Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839



As can be seen, the general trends of most of the features are consistent throughout the years.

AT tends to increase until the third quarter for all of the years then decreases.

AP is oscillating around 1010. There is a slight decrease for the first half of the year which is followed by a slight increase.

AH shows high degrees of oscillation between 100 and 40. However, it does not reach 100 during 2015.

For years 2011 and 2012, AFDP increases for two quarters and oscillates. In 2013 and 2014, it shows a noticeable increasing trend for the first 3 quarters then has a major decrease. In 2015, it oscillates between 3 and 5 throughout the year.

For years 2011 to 2014, GTEP oscillates between 20 and 35. In 2015, it reaches values close to 40.

TIT tends to oscillate between 1100 and 1000 for all the years. However, the level of oscillation seems to be small for the first half of 2011.

Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

For all the years, TAT has a value between around 520 to 550. It maxes out at 550.

TEY oscillates between 100 and 180 through the years.

CDP also oscillates between 10 and 14 through all the years available in the data.

Based on the results, we can determine that most features tend to oscillate between certain values through the years.

Q7

As done previously, let us use F-regression and mutual information scores to determine the importances of the features.

The features in our data are: ['Year', 'AT', 'AP', 'AH', 'AFDP', 'GTEP', 'TIT', 'TAT', 'TEY', 'CDP']

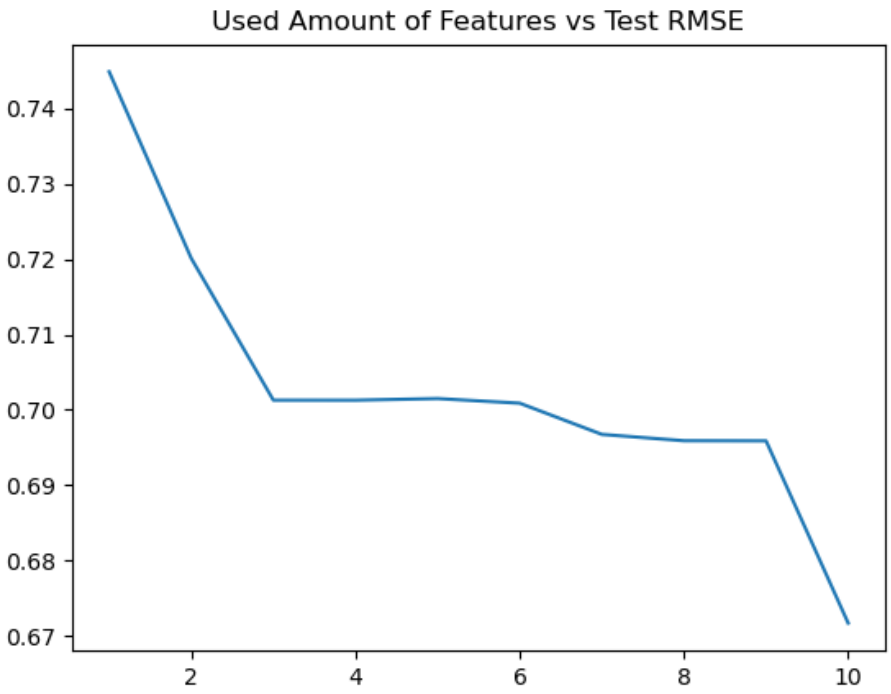
When using mutual information, the importance scores of each feature is determined to be:
[0.12312685 0.09903186 0.04295033 0.0263896 0.28183369 0.44563404
 0.5365391 0.15915222 0.49901823 0.47403852]

When using F-regression scores, the importance scores are calculated as follows:
[994.7381948   966.22469868   123.15814675   356.91391437
  7682.30333515 11088.10423195 29661.716599   119.74014885
 14401.87985866 13101.39699602]
According to both methods, the feature that has the most effect on CO is TIT. According to mutual information, the feature that affects CO the least is AH and according to F-regression scores, the feature that least affects CO is TAT. These are in parallel with our observations from the correlation matrix.

As before, we expected the test RMSE to improve as the features that were not related to the label were removed from the dataset. Removing one or two features that have low importance scores should get rid of the features that are not closely related to CO emissions was expected to improve the test RMSE results.

To test this hypothesis, we fit a simple linear regression model after performing feature selection to observe the trend in test rmse. The results are as follows:

Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839



As can be seen, using all the features led to the least test RMSE. As we did not want to include the amount of chosen features as a hyperparameter during model optimization and as the effect of getting rid of features seemed to be negative, we decided to keep all the features in the following steps.

Linear Regression

Q8

For regular linear regression, the hypothesis is that there is a linear relationship between the predictor features and the target variable. When we utilize ridge regression, we enforce the weights of each feature to be small. This functions as a regularization technique in order to overcome overfitting. Instead of only forcing the coefficient of each feature to be small, lasso regression forces the weights of certain coefficients to be 0. Therefore, in lasso regression, the hypothesis is that some of the features have no effect on the target variable.

Q9

To find the optimal parameter comb,nation, we utilized grid search with 10 fold cross validation. The parameter space that we searched through was as follows:

Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

Regularization Scheme	**Penalty**
None	None
Ridge	0.01,0.1,1,10
Lasso	0.01,0.1,1,10

As before, the combination that had the least average validation error was deemed to be the best. This combination was found to be **Ridge regression** with penalty **0.1**. This combination had an average train RMSE of **0.4063** and an average validation RMSE of **0.4068**. As explained before, since the goal of the model is to decrease the error on the train set, the average train RMSE is lower. For this parameter combination, the test RMSE was found to be **0.6716**.

Q10

In order to see the effect of feature scaling, we looked at the 10-fold cross validation performance of a linear regression model with different regularizations on both scaled and unscaled data. We then calculated the RMSE error on the test data for both cases.

For regular linear regression, the average training RMSE across folds was found to be **0.4063**, the average validation RMSE was **0.4068,** and the RMSE on the test data was found to be **0.6716** when no feature scaling was used. When feature scaling was used, the average training RMSE across folds was found to be **0.4063**, the average validation RMSE was **0.4068,** and the RMSE on the test data was found to be **0.6716.** From the results, we can again see that feature scaling does not change the results for regular linear regression. The reason behind this has been explained earlier in the report.

To see the effect of feature scaling on ridge regression, we utilized a ridge regression model with alpha=100. For this model, when no standardization was applied the average training RMSE across folds was found to be **0.4063**, the average validation RMSE was **0.4068,** and the RMSE on the test data was found to be **0.6716.** When standardization was applied the average training RMSE across folds was found to be **0.4094**, the average validation RMSE was **0.4098,** and the RMSE on the test data was found to be **0.6742.** The difference in error is related to the fact that ridge regression forces the weights to be close to each other without any consideration for what features the weights correspond to. However, when the scale of the features are different, it is not fair to treat each weight in the same manner.

To see the effect of feature scaling on lasso regression, we utilized a lasso regression model with alpha=0.1. For this model, when no standardization was applied the average training RMSE across folds was found to be **0.4352**, the average validation RMSE was **0.4356,** and the RMSE on the test data was found to be **0.6934.** When standardization was applied the average training RMSE across folds was found to be **0.4774**, the average validation RMSE was **0.4778,** and the RMSE on the test data was found to be **0.7309.** As with ridge regression, lasso

regression penalizes the weights in the same fashion, without attention to the scale of the features the weights correspond to. When the features are of different scales, this causes an unfair penalization across the weights. This explains the difference in performance.

Q11

As explained before, p-value is the result of testing the null hypothesis for each feature. A low p value indicates that the feature is related to the target variable. A high p value indicates that the feature is not significant for the target variable.

If we look at the p values after fitting a linear regression model to our data, we get the following: [0.9999999999991276 , 0.0, 2.887743116156451e-88, 2.6262532046084114e-05, 3.5589612737223327e-10, 0.40425620306986254, 0.007446547422453893, 1.5333045948299617e-63, 0.0, 4.243321878051149e-169]

For the features: ['Year', 'AT', 'AP', 'AH', 'AFDP', 'GTEP', 'TIT', 'TAT', 'TEY', 'CDP']

From the p scores, the most important features are found to be AT and TEY.

Polynomial Regression

Q12

As before, the first step to determine the optimal polynomial degree. The details of this step are explained in Q13. After this process, the best degree of polynomial was found to be 4.

After transforming the dataset with a polynomial of degree 4, the resulting set had 1001 features. To determine the most salient feature, we referred to the F-regression scores. Here, the highest related feature was determined to be the 511th feature with a score of 1567.7. Although it is difficult to tell which combination of features this particular feature corresponds to, we can infer that this is most likely the combination of the 4 features that are the most related to CO emissions.

Q13

In order to determine the best degree of the polynomial, we performed grid search with 10 fold cross validation. The space we searched was as follows:

Polynomial Degree	2, 3, 4, 5

After changing the dataset based on these polynomial degrees, we fit a linear regression model to perform predictions.

After the grid search, the best results were obtained with a polynomial degree of 4. Here, the mean cross validation train RMSE was found to be **0.1710** and the mean validation RMSE was **0.2228.** The reason behind the discrepancy between mean train and validation RMSEs have been explained before. The test RMSE for this setup was found to be **0.5013.**

Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

As explained before, a high degree fit on the training set usually implies overfitting. This is apparent since an infinite degree polynomial would achieve 0 train error. However, it would not perform well on unseen data. This interpretation is also supported by examining the grid search results.

Neural Networks

Q15

As explained before, neural networks are capable of learning non-linear relationships between features and target variables. This increases their performance when compared to linear regression.

Q16

In order to find good values for the amount of layers, the amount of neurons in each layer and the weight decay parameter, we again performed grid search with 10-fold cross validation. The hyperparameter space that we searched was as follows:

Hidden layer sizes	(50),(100),(150),(50,50),(50,100),(100,150),(50,150)
Weight decay	0.00001, 0.0001, 0.001, 0.01, 0.1

The best parameter combination had the least mean validation RMSE across the 10 folds. This combination was achieved with hidden layer sizes of (100,150) and a weight decay parameter of 0.001. For this combination, the mean training RMSE was **0.1238** and the mean validation RMSE was **0.2131.** As explained before, since the model performs updates according to the training data, the mean RMSE for training is lower. The test RMSE for this setup was found to be **0.5011**.

Q17 & Q18

The answers for Q17 and Q18 have been given in the earlier sections of the report. Therefore, we will not repeat the same answers here as well.

Random Forest

Q19

In order to find the optimal number of hyperparameters, we again resorted to using the grid search algorithm. Here, the parameter space that we tried was as follows:

Number of Estimators	50,100,150,200
Max Features	3,5,7,9,10

Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

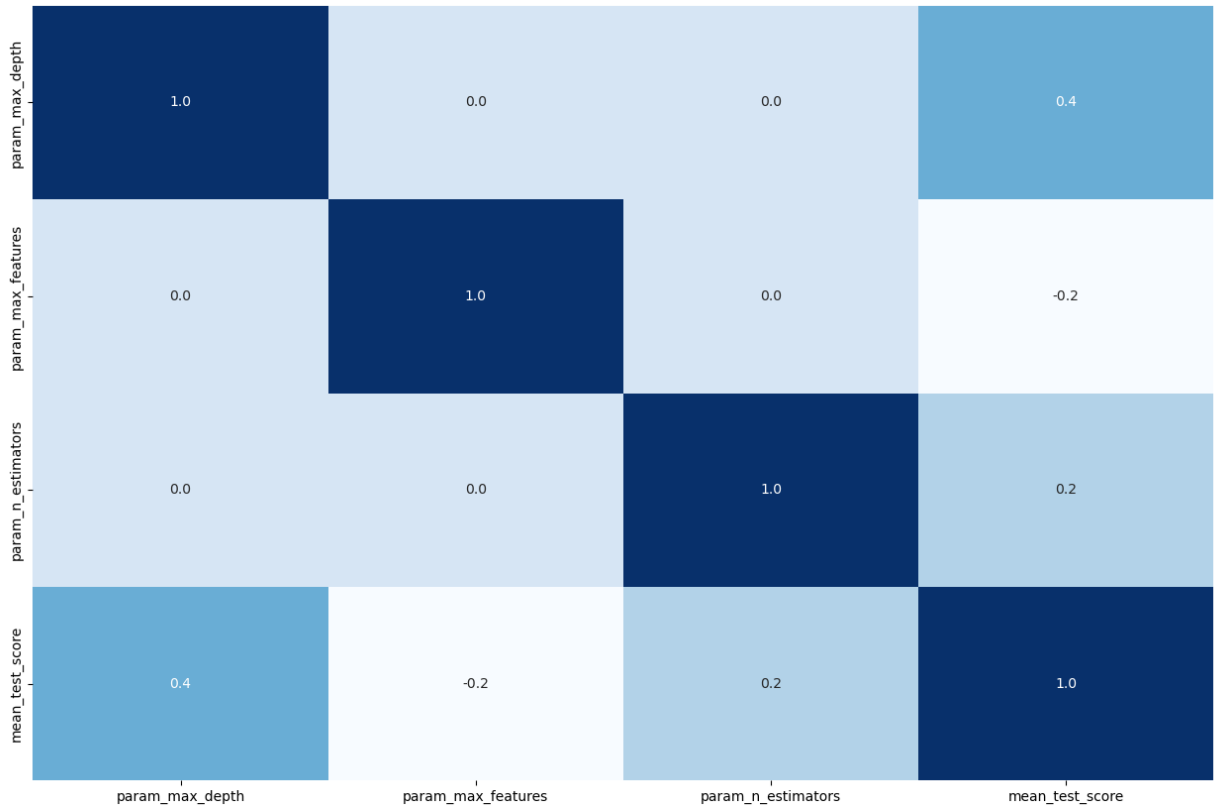
Max Depth	10,100,500,1000,2500, None

The combination that had the lowest mean validation RMSE was deemed to be the best parameter combination. After the search process, this combination was found to have 200 estimators, with a maximum of 7 features considered at each split and a maximum depth of 100. For this combination, the average training error was found to be **0.0311** and the average validation error was found to be **0.2234**. As explained earlier, as the model sees the training data, the average error on the train set is less than the average error on the validation set. For this setup, the test RMSE was found to be **0.4697**.

As we have already explained what OOB error and R^2 is when doing this part of the project for the Diamonds dataset, we will not go into the details again.

To measure the OOB error, we trained a random forest model using the found optimal parameters and set the oob_score parameter to True. As the maximum OOB score is 1, we subtracted the returned value from 1 to get the error. This revealed the error to be **0.2244**.

To see the effect of each parameter, we can look at the correlation matrix between the parameters and the mean validation error.



As can be seen, the feature that most affects the mean test score is max_depth. B also lookşng at grid search results, we can see that increasing max depth led to an increase in performance for this data. Examining the results also showed that keeping the max_features around 3 or 5

offered the best performance. For the number of estimators, keeping the value at 100 offered the best performance. When the number of estimators was kept at 50, the model performance suffered, possibly due to underfitting. However, setting the value to 150 also proved to be disadvantageous due to overfitting.

As explained earlier, features max depth and number of estimators can be thought to have a regularization effect as low values force the model to be simple, which decreases the effect of overfitting. Similarly, keeping the maximum features low forces the model to generate splits based on different feature subsets, which increases randomness and decreases the likelihood of overfitting. Therefore, it can be thought to have a regularizing effect.

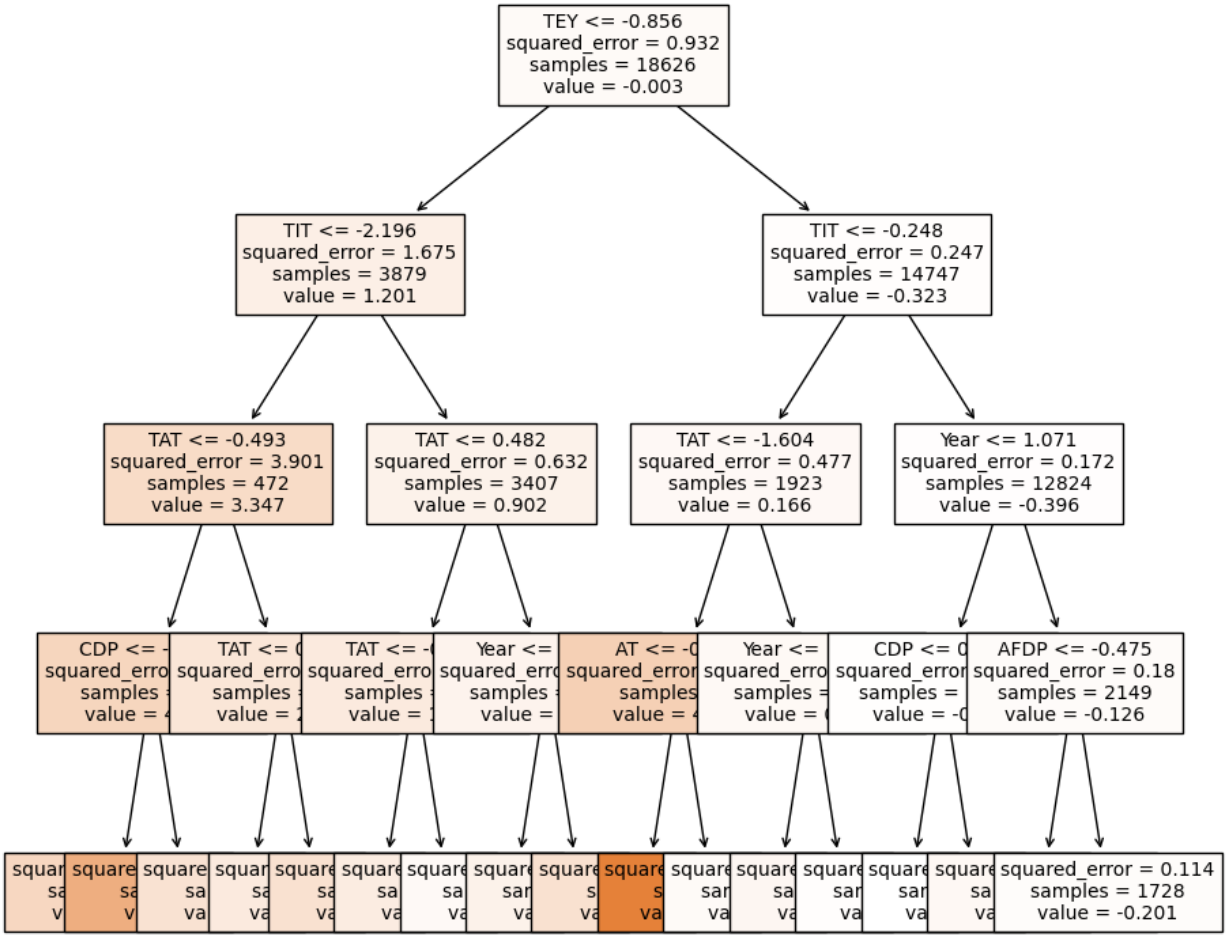
Q20

Q20 has been explained earlier, so we will not get into the details again here. In a nutshell, random forest has a large number of uncorrelated predictors, which causes an improvement in performance.

Q21

In the question, we are asked to randomly select a tree from our random forest model and visualize it. The minimum depth for the optimized random forest model was found to be 39. Therefore, in order to fit the depth requirement, we trained another random forest model with 200 estimators and a maximum of 7 features but this time we set the max depth to 4. Afterwards, we picked a random tree for visualization. The results are as follows:

Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839



From the figure, we can see that the feature selected for branching at the root node is TEY. Then, this tree utilizes features TIT, TAT and Year. Among these features, we can infer that the feature closer to the root is more important. So TEY is the most important feature. It is followed by TIT then TAT and Year.

In part 3.2.1, we have determined that TEY is one of the most important features in the set, so having it in the root node makes sense. Among the important features of the tree there is also AT, however, the tree does not make use of this feature, which contradicts with our findings. Furthermore, according to the results obtained in part 3.2.1, year has almost no relation to CO emissions. So, it is surprising to see that it has been used as the fourth most important feature on the tree.

Twitter Data

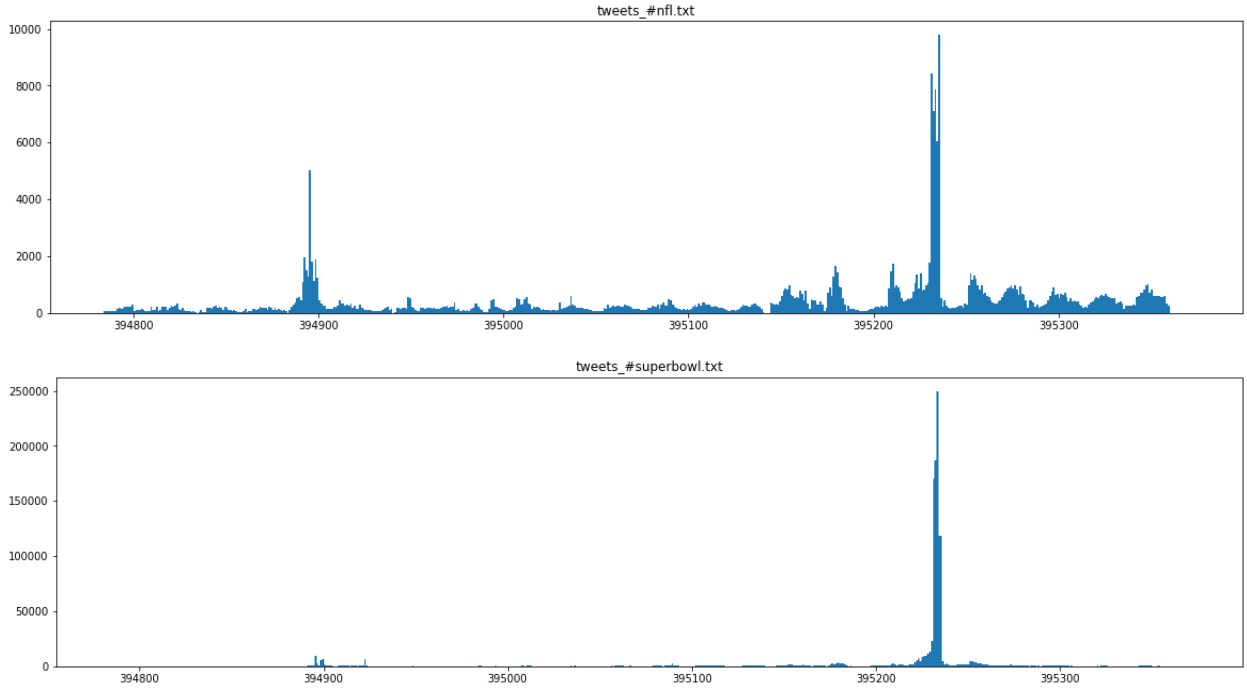
Q27

Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

	Average number of tweets/hr	Average number of followers	Average number of retweets
tweets_#superbowl.txt	2072.11840170408	8814.96799424623	2.391189581920
tweets_#patriots.txt	750.89426460689	3280.4635616550277	1.7852871288476
tweets_#sb49.txt	1276.85705986804	10374.160292019487	2.52713444111402
tweets_#gopatriots.txt	40.95469800606194	1427.2526051635405	1.40819191016970
tweets_#nfl.txt	397.0213901819841	4662.37544523693	1.53446026555432
tweets_#gohawks.txt	292.4878506217368	2217.9237355281984	2.01320939913198

Q28

Number of tweets in an hour for #nfl and #superbowl/



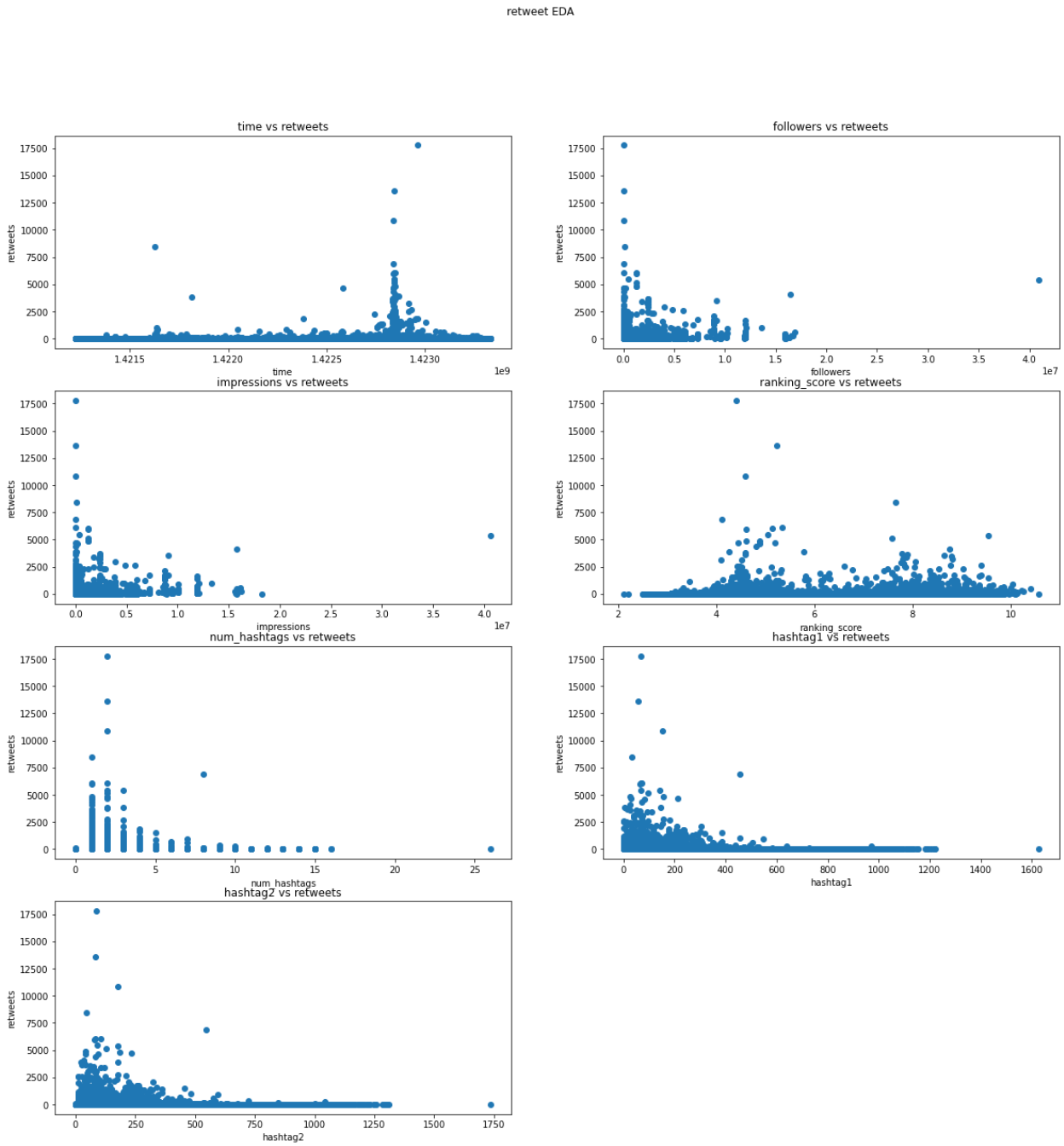
Q29

Task: Library of Prediction Tasks given a tweet

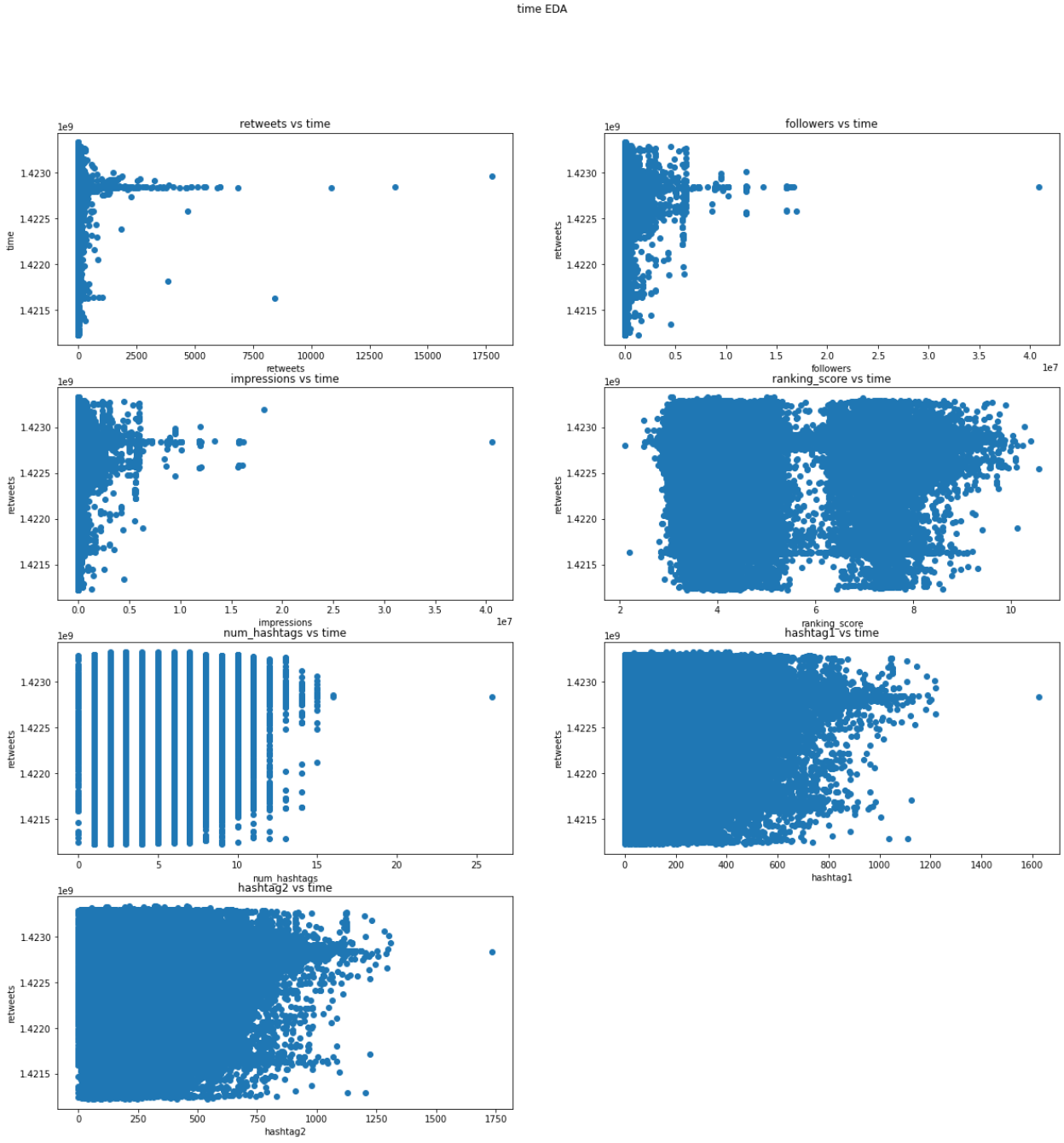
Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839

For this task our goal was to build models to regress the number of retweets and time that a tweet had been published. We performed this on the #superbowl.txt dataset.

Data Exploration: Here we've plotted the labels vs the features used for each of the respective tasks. From the data exploration we are able to see some correlations between the features and the label of interest, even if perhaps not a single feature is a very good predictor.



Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839



Feature Engineering:

We used the features: Time, Retweets, Ranking_Score, Impressions and Hashtags to train both models, omitting the respective features and using them as labels as appropriate for each task (Time, Retweets). For the hashtags, we preprocessed them into 3 different features, in order to keep a consistent size. We used the number of hashtags, and then summed the indices of the first and second hashtags respectively to create the features, hashtag1 and hashtag2. From the

data exploration of these features we were able to see some relationships between either time or retweets and the features so we felt they could help in the prediction task.

We also preprocessed the tweet text, cleaning it to remove hashtags. We then attempted to use GLoVE to find the sentence embeddings but due to resource issues (time/memory) we were unable to use the sentence embeddings in our models.

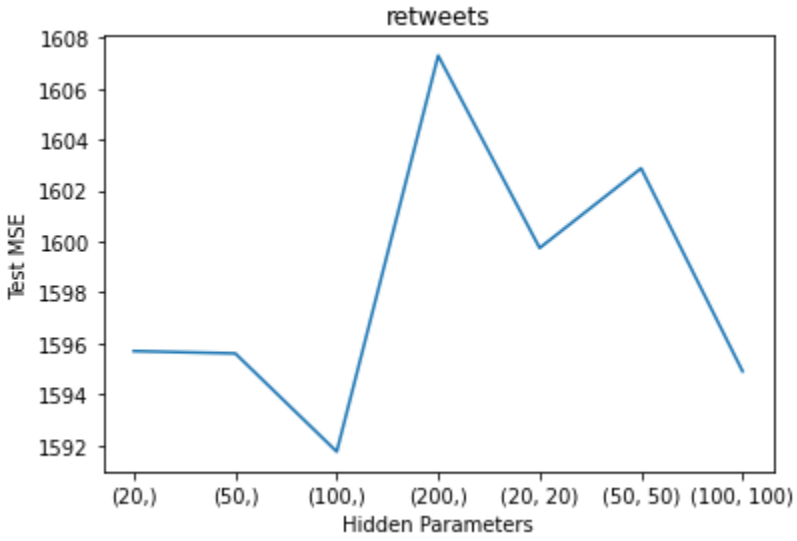
Baselines:

As baselines, we used three models: LinearRegression, RidgeRegression and simplified neural network with a single hidden layer of size 10.

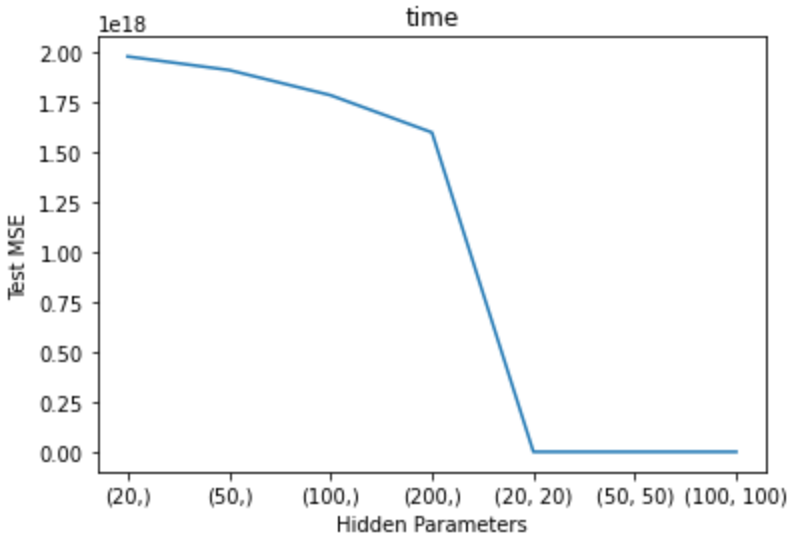
Evaluation/Performance:

We performed Cross Validation across various NN architectures for both tasks, and chose the best performing model to compare against our baselines. For the retweet prediction task, the 100 param NN outperformed all other models, followed by the 10 NN and then the Ridge Regression and last, the Linear Regression. For the Time prediction task however, the Ridge and Linear models were our most performant models, followed by the (20,20) NN and far behind the (10,) NN.

5-Fold Cross Validation for Different NN Architectures



Deniz Orkun Eren – UID: 905624625
Matthew Waliman – UID: 605848839



Performance on Retweet Regression

	Test MSE
MLP (100,)	**690.7681096765979**
MLP (10,)	696.153928009655
LinearRegression	705.6724949895731
RidgeRegression	705.6724507603061

Performance on Time Regression

	Test MSE
MLP (20,20)	75504136720.94133
MLP (10,)	1.9872063718381125e+18
LinearRegression	74066028120.02184
RidgeRegression	**74066026580.47653**

Setbacks:
Although our performance is less than ideal, we faced many setbacks trying to train these prediction tasks. We developed GloVE sentence embeddings for the tweet text but had to remove them due to memory issues. We originally tried to train the tasks on Google Colab but were unable to fit the data into the memory available. In addition we had to scale back the size of the dataset to only include the #superbowl tweets. Training these simple models required over a day of training on our local machines. Therefore, we were restricted to only use simple

approaches. However, we were still able to use these models to perform regression on the available data.