ECE-219

Data Representation and Clustering

Team Member Names:

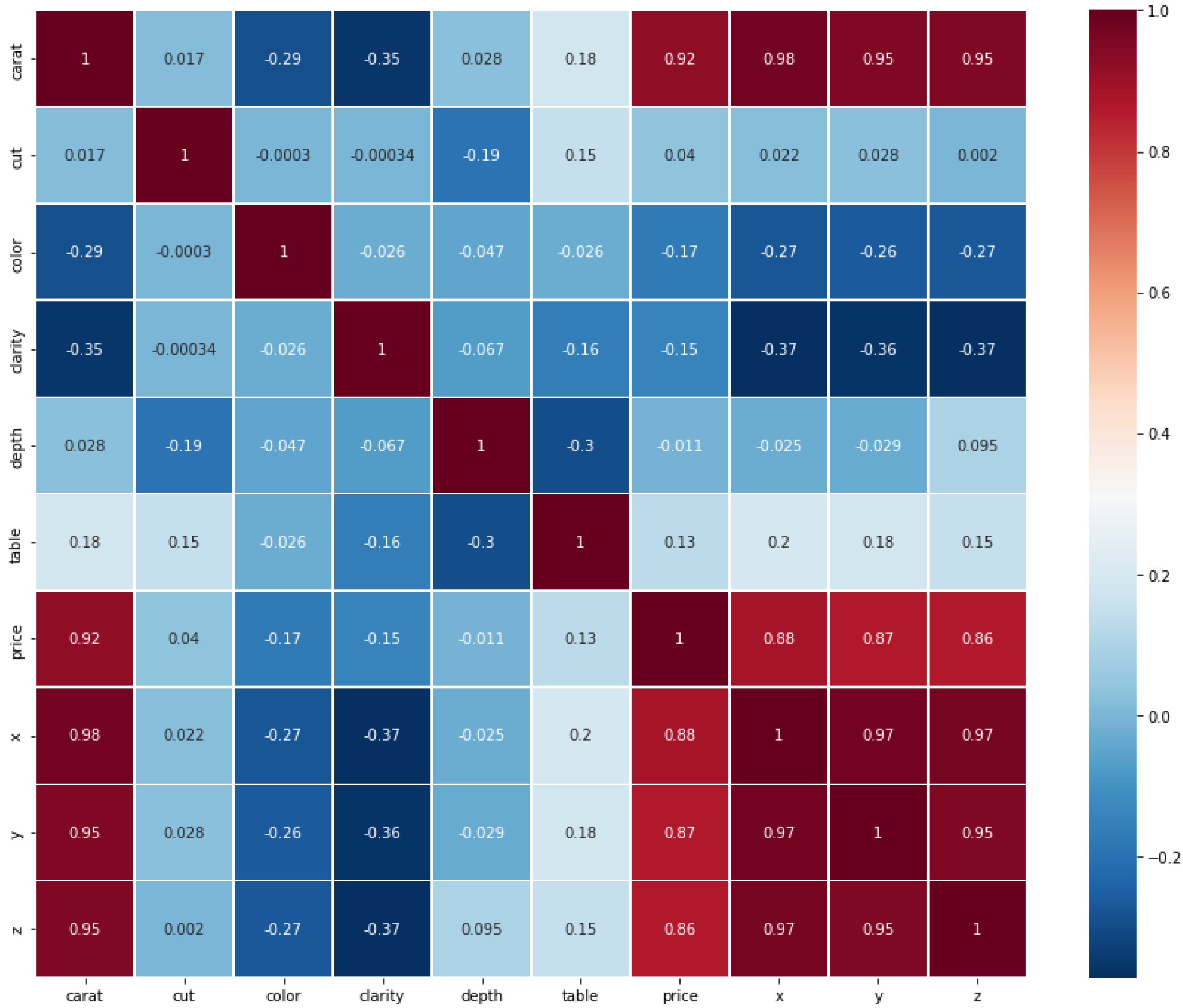
Tianpei Gu, 405863048
Yuxin Huang, 105711853
Yilin Xie, 405729012

Question 1

- We use sklearn.preprocessing.scale() function to standardize feature columns and prepare them for training.

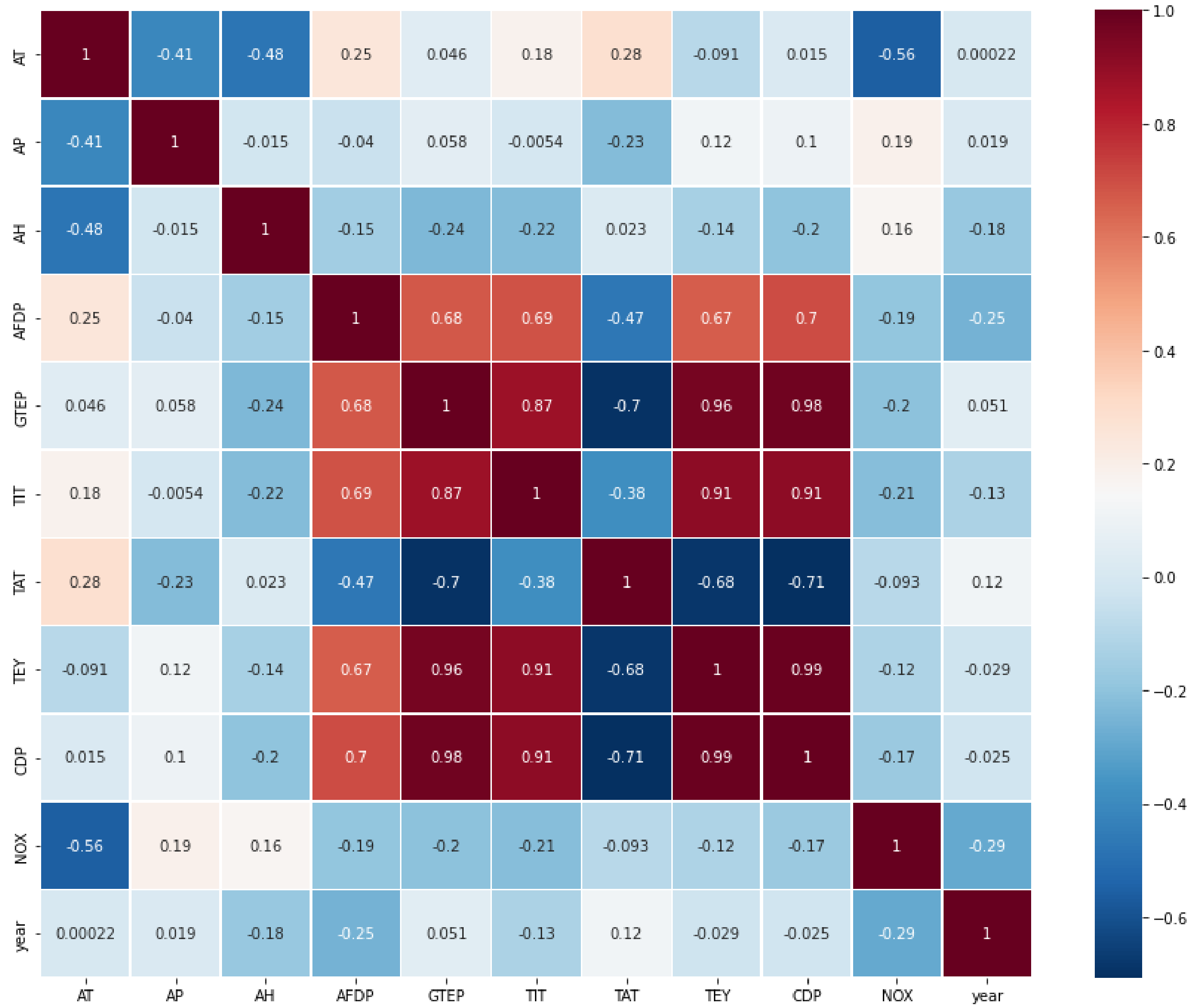
Question 2

- Diamond Dataset Heatmap



- Feature **carat** has the highest absolute correlation value with the target variable price. This suggests that carat is the most informative feature to predict the price of the diamond.

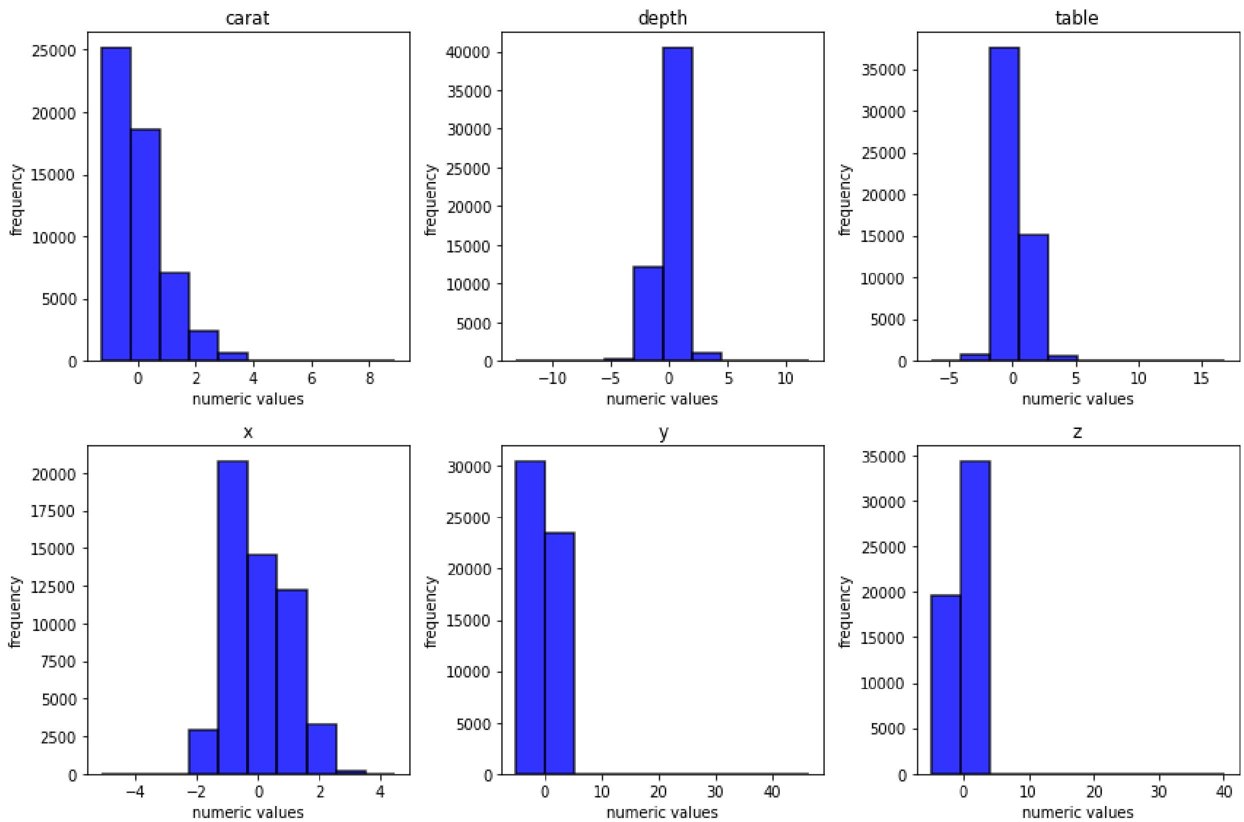
- Gas Turbine Dataset Heatmap (dropped “CO”)



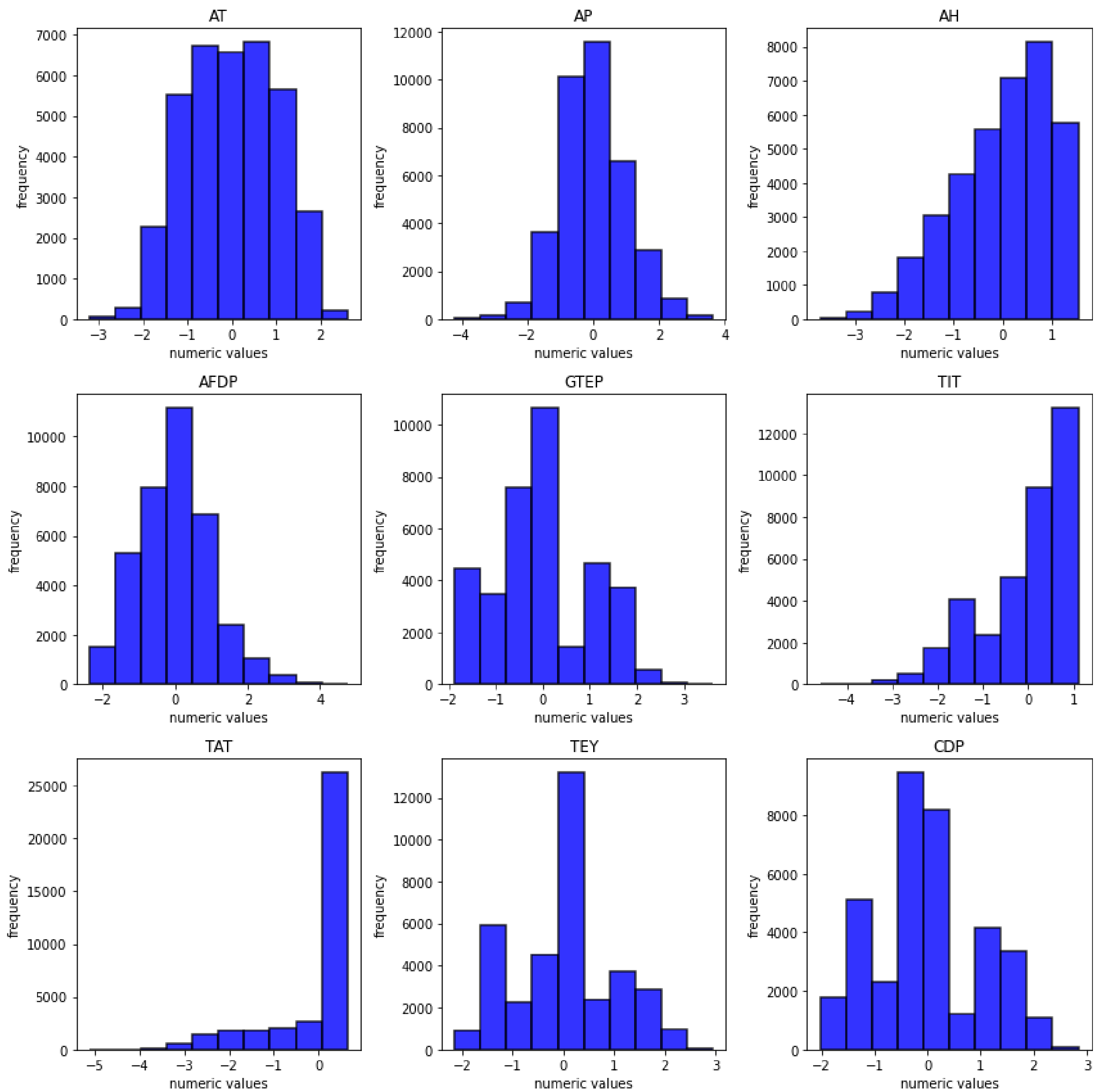
- Feature **AT** has the highest absolute correlation value with the target variable NOx. This suggests that AT is the most informative feature to predict NOx emission.

Question 3

- If a feature has high skewness, we can preprocess the data by **standardization** to the high-skewed features.
- Diamond

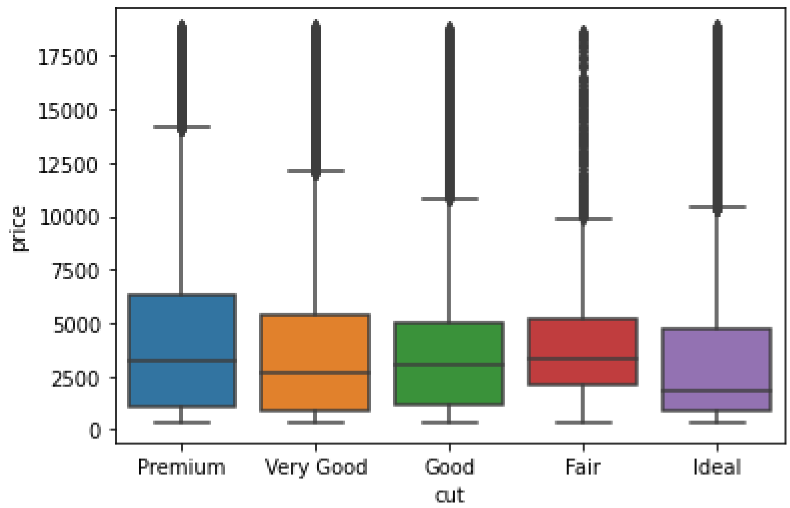


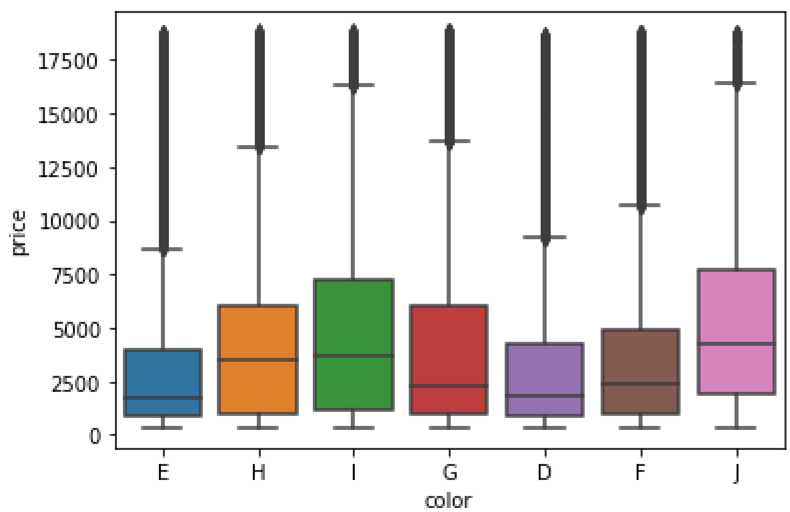
- Gas

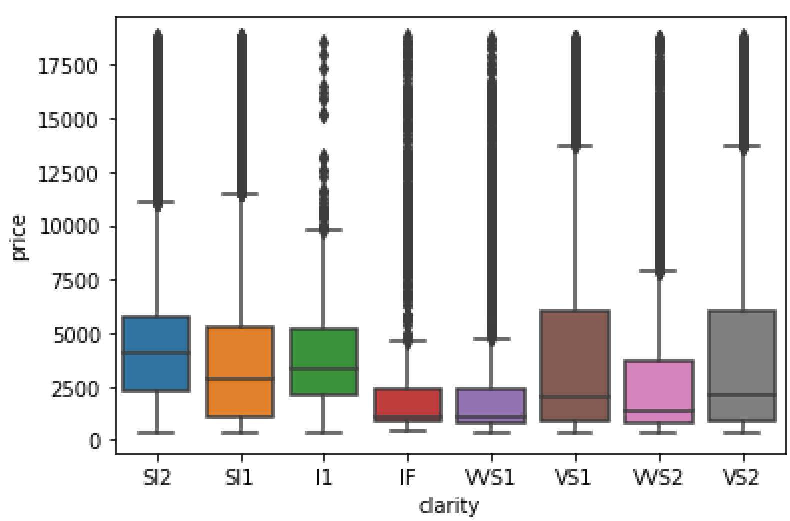


Question 4

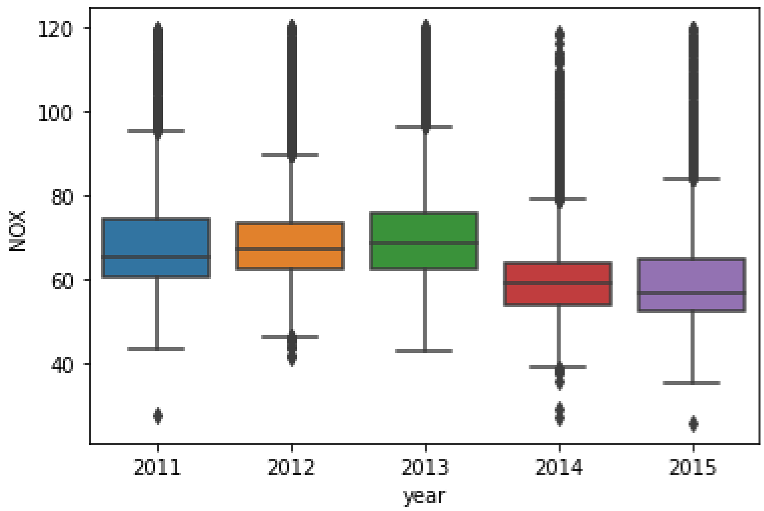
- Diamond
    - Categorical features: cut, color, clarity
    - Intuitions:
        - Cut: premium quality diamonds generally worth higher price
        - Color: diamond color I and J generally worth higher price
        - Clarity: VS1 and VS2 clarity generally worth higher price



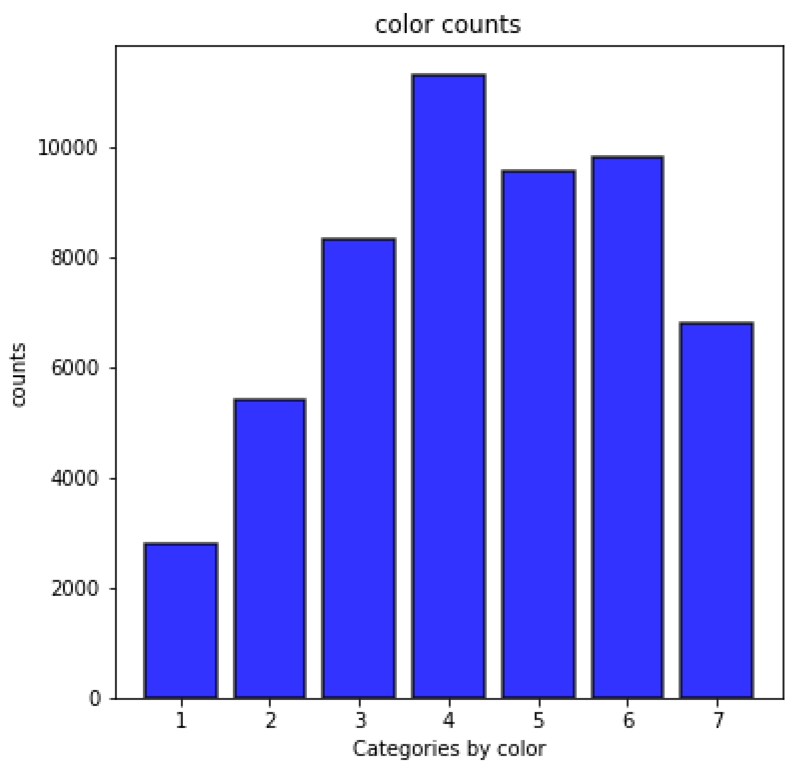


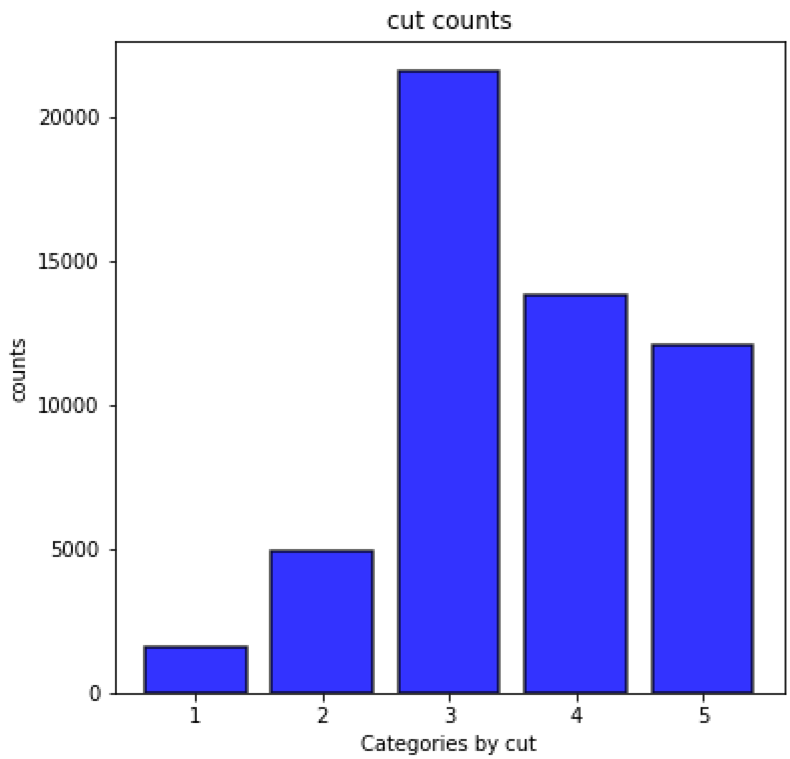


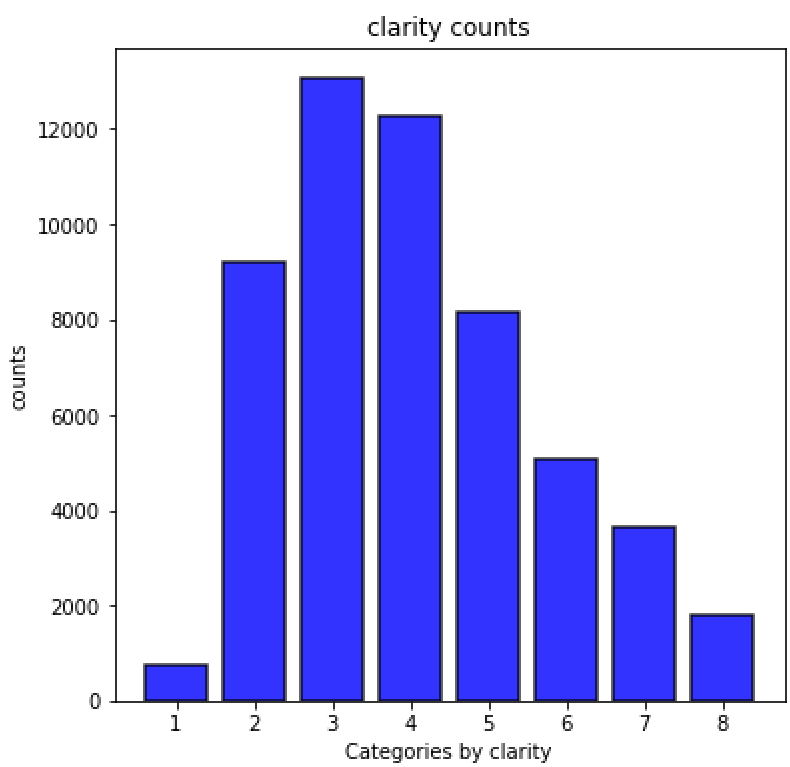
- Gas
    - Categorical features: year
    - Intuition:
        - NOx emission rate drops in the year of 2014 and 2015


120
100
80
60
40
NOX
2011
2012
2013
2014
2015
year

Question 5

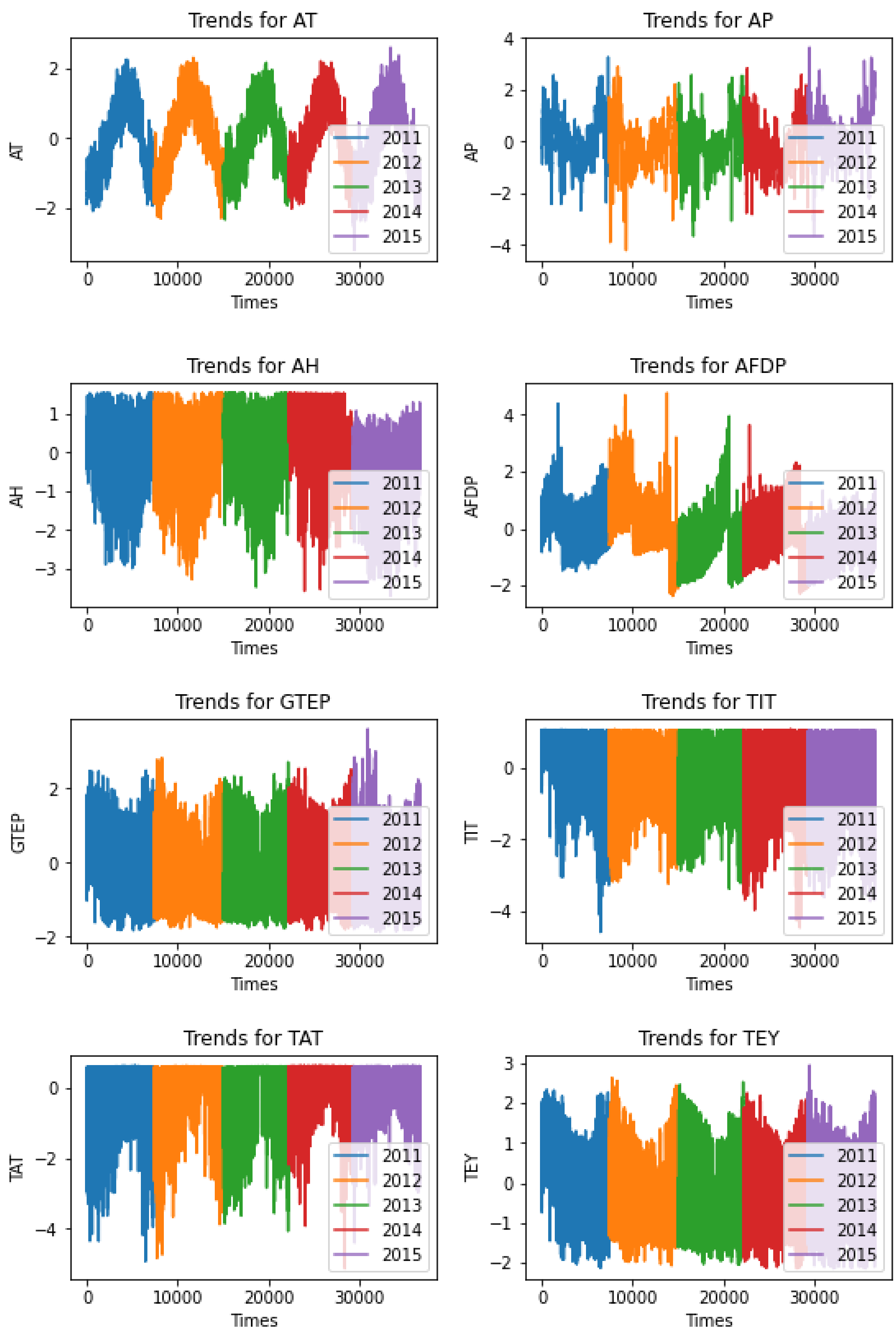

color counts
10000
8000
6000
4000
2000
0
counts
1
2
3
4
5
6
7
Categories by color

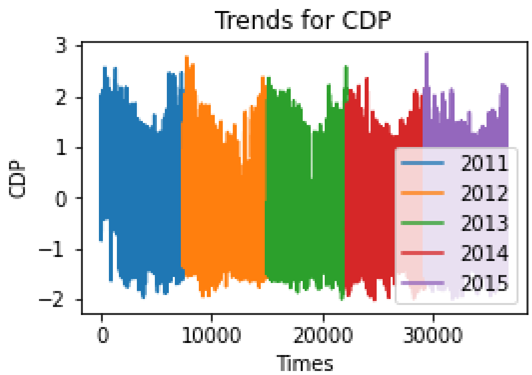




Question 6

- We can see that the trend for each feature for each year is nearly identical.


Trends for AT
AT
Times
Trends for AP
AP
Times
Trends for AH
AH
Times
Trends for AFDP
AFDP
Times
Trends for GTEP
GTEP
Times
Trends for TIT
TIT
Times
Trends for TAT
TAT
Times
Trends for TEY
TEY
Times
2011
2012
2013
2014
2015



Question 7

- The testing result by applying feature selection would be improved for linear models. Less relevant features sometimes include the bias into the model and increases the risk of overfitting, getting rid of these features can help the model find more robust regression.

Question 8

- Ordinary Least Square : No regularization
- Lasso Regression: It uses **L1 regularization** technique. Only a fraction of features are active and it allows you to shrink or regularize these coefficients to avoid overfitting and make them work better on different datasets.
- Ridge Regression: It uses **L2 regularization** technique. All features are active. Ridge regression is a method of estimating the coefficients of multiple-regression models in scenarios where linearly independent variables are highly correlated.

Question 9

- We choose five top features based on both Mutual Information and F scores. For each model, we calculate on both standardized and non-standardized data. We perform grid search over alpha parameters from [0.001,0.01,0.1,1,10,100] for Ridge and Lasso Regression. Eventually we choose the model with least testing RMSE.

- Diamond dataset
    - From the top 10 regularization schemes, we can see that the ordinary least squares model works as the best model. Feature selection method and standardization do not affect the mean test score.

	mean_test_score	mean_train_score	param_model	param_model__alpha	Standardize	Feature Selection
0	-1408.599008	-1504.205849	LinearRegression()	N/A	True	F Scores
1	-1408.599008	-1504.205849	LinearRegression()	N/A	False	F Scores
2	-1408.599008	-1504.205849	LinearRegression()	N/A	True	Mutual Information
3	-1408.599008	-1504.205849	LinearRegression()	N/A	False	Mutual Information
4	-1408.599082	-1504.205849	Ridge(alpha=0.001, max_iter=10000, random_stat...	0.001	True	F Scores
5	-1408.599082	-1504.205849	Ridge(alpha=0.001, max_iter=10000, random_stat...	0.001	True	Mutual Information
6	-1408.599410	-1504.205849	Ridge(alpha=0.001, max_iter=10000, random_stat...	0.001	False	F Scores
7	-1408.599410	-1504.205849	Ridge(alpha=0.001, max_iter=10000, random_stat...	0.001	False	Mutual Information
8	-1408.599754	-1504.205849	Ridge(alpha=0.001, max_iter=10000, random_stat...	0.01	True	Mutual Information
9	-1408.599754	-1504.205849	Ridge(alpha=0.001, max_iter=10000, random_stat...	0.01	True	F Scores
10	-1408.600482	-1504.205849	Lasso(max_iter=10000, random_state=42)	0.001	True	Mutual Information

- Gas Turbine dataset
    - From the top 10 regularization schemes, we can see the Mutual Information outcompetes F scores. With the Mutual Information feature selection method, Lasso linear regression model with standardized data performs the best.

	mean_test_score	mean_train_score	param_model	param_model__alpha	Standardize	Feature Selection
0	-9.450098	-9.149821	Lasso(max_iter=10000, random_state=42)	0.01	True	Mutual Information
1	-9.453797	-9.144270	Lasso(max_iter=10000, random_state=42)	0.001	True	Mutual Information
2	-9.454315	-9.146621	Ridge(alpha=0.001, max_iter=10000, random_stat...	10.0	True	Mutual Information
3	-9.454963	-9.144242	Ridge(alpha=0.001, max_iter=10000, random_stat...	1.0	True	Mutual Information
4	-9.455322	-9.144213	Ridge(alpha=0.001, max_iter=10000, random_stat...	0.1	True	Mutual Information
5	-9.455362	-9.144213	Ridge(alpha=0.001, max_iter=10000, random_stat...	0.01	True	Mutual Information
6	-9.455366	-9.144213	Ridge(alpha=0.001, max_iter=10000, random_stat...	0.001	True	Mutual Information
7	-9.455366	-9.144213	LinearRegression()	N/A	False	Mutual Information
8	-9.455366	-9.144213	LinearRegression()	N/A	True	Mutual Information
9	-9.455366	-9.144213	Ridge(alpha=0.001, max_iter=10000, random_stat...	0.001	False	Mutual Information

Question 10

- **For models without regularization** (i.e. ordinary least squares model), feature scaling does not affect the mean test score. This is because standardization will not change the coefficients.
    - We can observe from the grid search result on diamond dataset:

	mean_test_score	mean_train_score	param_model	param_model__alpha	Standardize	Feature Selection
0	-1408.599008	-1504.205849	LinearRegression()	N/A	True	F Scores
1	-1408.599008	-1504.205849	LinearRegression()	N/A	False	F Scores

    - As well as on gas turbine dataset:

7	-9.455366	-9.144213	LinearRegression()	N/A	False	Mutual Information
8	-9.455366	-9.144213	LinearRegression()	N/A	True	Mutual Information

- **For models with regularization** (i.e. Ridge and Lasso regression model), feature scaling affects the mean test score. This is because normalization will cause changes to estimated coefficients. We can see from both diamond and gas turbine dataset that model with scaling, while other parts in the pipeline are identical, performs better than model without standardization.

Question 11

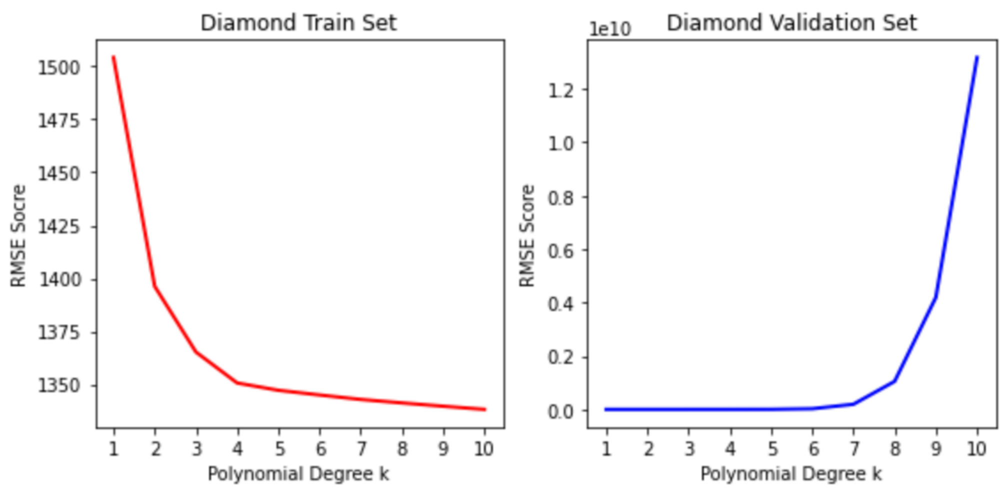
- P-values and coefficients in regression analysis work together to tell you which relationships in your model are statistically significant and the nature of those relationships. The coefficients describe the mathematical relationship between each independent variable and the dependent variable. The p-values for the coefficients indicate whether these relationships are statistically significant.
- If the p-value for a feature is close to 0, then that particular feature is significant in the linear model.
- For diamond dataset, the 5 most significant features with very small values include:
    - carat 0.000000e+00
    - depth 1.493370e-294
    - table 1.648931e-239
    - x 3.392377e-203
    - y 9.353084e-03
- For gas turbine dataset, the 5 most significant features with very small values include:
    - AT 0.000000e+00
    - AH 0.000000e+00
    - TIT 0.000000e+00
    - TAT 0.000000e+00
    - TEY 0.000000e+00

Question 12

- The most salient features are features with greatest absolute coefficients with the target variables.
- For Diamond dataset, the most salient features:
    - Carat
    - x
- For Gas Turbine dataset, the most salient features:
    - AT
    - year

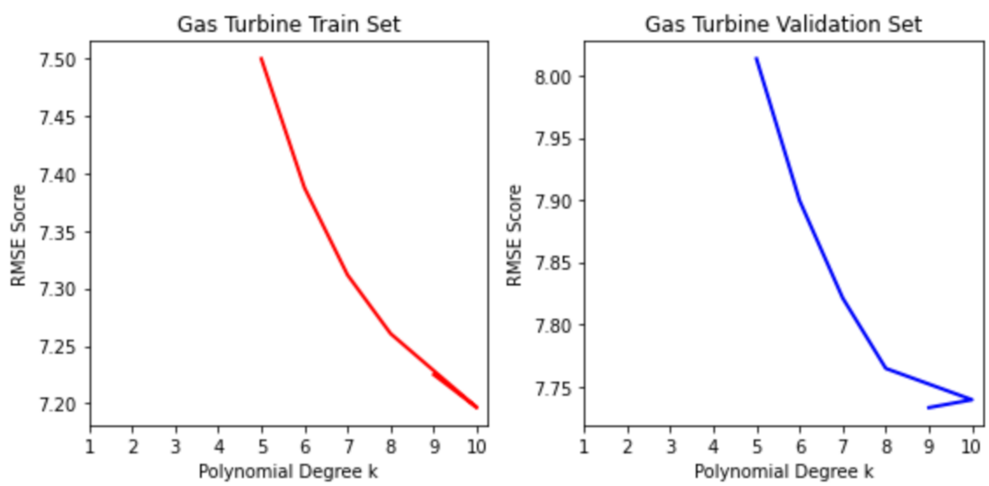
Question 13

- For Diamond Dataset:



- K = 6. The RMSE after 6 is increasing in validation set while decreasing in training set and it indicates a possible overfitting due to higher degree.

- For Gas Dataset:



- K = 9. The RMSE in train set and val set decrease together, and K=9 reaches a local minima.

- In both dataset, very high-order polynomial imply better fitting result in training set.

Question 14

- We choose **x1=carat, x2=x.**
- This is because they have the top two largest absolute correlation coefficients with the “price” variable.
- By applying x1*x2, RMSE drops from **1408** to **1394** and this technique indeed boosts the performance.

Question 15

- Neural network provides non-linear activation function between each layers and thus able to handle non-linear dependencies between features.

Question 16

- Diamond
    - Hyperparameter alpha: 0.1, 0.01, 0.005.
    - Hyperparameter activation: 'relu', 'logistic', 'tanh'.
    - Hyperparameter hidden_layer_sizes: 100,150,200.
    - The result of neural network with **one** hidden layer

	mean_test_score	mean_train_score	param_model__alpha	param_model__activation	param_model__hidden_layer_sizes
0	-1396.690211	-1364.131279	0.1	relu	100
1	-1405.230769	-1363.280250	0.1	relu	150
2	-1467.181005	-1360.854024	0.1	relu	200
3	-1395.304163	-1364.691679	0.01	relu	100
4	-1410.932057	-1364.175990	0.01	relu	150
5	-1458.576289	-1361.262772	0.01	relu	200
6	-1398.025995	-1364.211330	0.005	relu	100
7	-1417.213567	-1362.829874	0.005	relu	150
8	-1460.068024	-1361.313401	0.005	relu	200
9	-1426.272355	-1339.336097	0.1	logistic	100
10	-1426.186496	-1340.773830	0.1	logistic	150
11	-1428.853475	-1342.436359	0.1	logistic	200
12	-1426.542244	-1339.499377	0.01	logistic	100
13	-1425.865748	-1341.155097	0.01	logistic	150
14	-1429.095331	-1342.213350	0.01	logistic	200
15	-1427.411564	-1339.542302	0.005	logistic	100
16	-1425.844389	-1341.161313	0.005	logistic	150
17	-1429.105794	-1342.209670	0.005	logistic	200
18	-1414.061300	-1328.726233	0.1	tanh	100
19	-1420.154619	-1329.533641	0.1	tanh	150
20	-1420.678641	-1331.031741	0.1	tanh	200
21	-1413.664580	-1328.003463	0.01	tanh	100
22	-1418.075724	-1329.208262	0.01	tanh	150
23	-1421.692497	-1330.912207	0.01	tanh	200
24	-1416.211789	-1329.074609	0.005	tanh	100
25	-1417.583428	-1329.031757	0.005	tanh	150
26	-1422.117845	-1330.799048	0.005	tanh	200

- Gas Turbine
    - Hyperparameter alpha: 0.1, 0.01, 0.005.
    - Hyperparameter activation: 'relu', 'logistic', 'tanh'.
    - Hyperparameter hidden_layer_sizes: 100,150,200.
    - The result of neural network with **two** hidden layer

	mean_test_score	mean_train_score	param_model__alpha	param_model__activation	param_model__hidden_layer_sizes
0	-7.509562	-6.047456	0.1	relu	(100, 100)
1	-7.469834	-6.030676	0.1	relu	(150, 150)
2	-7.737920	-6.043765	0.1	relu	(200, 200)
3	-7.839761	-6.062116	0.01	relu	(100, 100)
4	-7.474214	-5.974080	0.01	relu	(150, 150)
5	-7.627646	-6.018200	0.01	relu	(200, 200)
6	-7.607163	-6.018576	0.005	relu	(100, 100)
7	-7.460844	-5.989425	0.005	relu	(150, 150)
8	-7.790313	-6.079037	0.005	relu	(200, 200)
9	-7.400975	-6.151433	0.1	logistic	(100, 100)
10	-7.517762	-6.142766	0.1	logistic	(150, 150)
11	-7.356913	-6.048584	0.1	logistic	(200, 200)
12	-7.428871	-5.885184	0.01	logistic	(100, 100)
13	-7.536814	-5.785658	0.01	logistic	(150, 150)
14	-7.414612	-5.679658	0.01	logistic	(200, 200)
15	-7.470452	-5.863358	0.005	logistic	(100, 100)
16	-7.404041	-5.730328	0.005	logistic	(150, 150)
17	-7.411395	-5.565917	0.005	logistic	(200, 200)
18	-7.574076	-5.786126	0.1	tanh	(100, 100)
19	-7.588302	-5.694449	0.1	tanh	(150, 150)
20	-7.502897	-5.626410	0.1	tanh	(200, 200)
21	-7.467286	-5.736388	0.01	tanh	(100, 100)
22	-7.544029	-5.654967	0.01	tanh	(150, 150)
23	-7.423638	-5.569926	0.01	tanh	(200, 200)
24	-7.421072	-5.779633	0.005	tanh	(100, 100)
25	-7.576225	-5.674139	0.005	tanh	(150, 150)
26	-7.416821	-5.617735	0.005	tanh	(200, 200)

Question 17

- Diamond
    - According to the result, we choose **relu**.

- Gas Turbine
    - According to the result, we choose **logistic**.

Question 18

- The main problem of increasing the depth of neural network is the risk of overfitting. Moreover, a deeper network leads to more training and inference time which is miserable for hyperparameter tuning.

Question 19

- For **diamond** dataset
    - Maxinum number of features:

	mean_test_score	mean_train_score	param_model__max_features
0	-1452.755434	-515.503876	0.3
1	-1453.624578	-515.662162	0.1
2	-1454.459408	-515.879388	0.2
3	-1459.836219	-518.626148	0.5
4	-1460.855967	-518.428692	0.4
5	-1468.198860	-521.320605	0.7
6	-1470.175969	-520.454986	0.6
7	-1473.700590	-523.038468	0.9
8	-1473.866624	-523.155849	0.8
9	-1480.429990	-525.082720	1.0

    - Number of trees:

	mean_test_score	mean_train_score	param_model__n_estimators
0	-1475.893421	-520.295100	190
1	-1476.618646	-520.127955	200
2	-1476.817459	-522.221505	130
3	-1476.847772	-520.563772	170
4	-1476.935283	-524.755174	100
5	-1477.001514	-523.493186	120
6	-1477.004211	-520.614566	180
7	-1477.018578	-520.949339	160
8	-1477.321461	-522.117500	140
9	-1478.100238	-521.521195	150
10	-1479.241596	-525.765898	90

    - Depth of trees:

	mean_test_score	mean_train_score	param_model__max_depth
0	-1432.114225	-1247.060799	9
1	-1433.003364	-1215.059738	10
2	-1433.648161	-1176.977619	11
3	-1434.202847	-1274.266487	8
4	-1435.532886	-1134.415418	12
5	-1436.881460	-1085.722704	13
6	-1438.625244	-1297.428737	7
7	-1442.664914	-1032.769188	14
8	-1444.432479	-974.762671	15
9	-1446.886553	-1318.084857	6

- For **gas** dataset:
    - Maxinum number of features:

	mean_test_score	mean_train_score	param_model__max_features
0	-7.378146	-2.205668	0.2
1	-7.381496	-2.207027	0.1
2	-7.386176	-2.207446	0.3
3	-7.426436	-2.172261	0.4
4	-7.438703	-2.172723	0.5
5	-7.481258	-2.174597	0.7
6	-7.504523	-2.174859	0.6
7	-7.556498	-2.183932	0.9
8	-7.573901	-2.180890	0.8
9	-7.636391	-2.193490	1.0

    - Number of trees:

	mean_test_score	mean_train_score	param_model__n_estimators
0	-7.620168	-2.168295	200
1	-7.620639	-2.193384	100
2	-7.623862	-2.172810	170
3	-7.624399	-2.183039	130
4	-7.624579	-2.173634	160
5	-7.627191	-2.176529	140
6	-7.631562	-2.168281	190
7	-7.633433	-2.175126	150
8	-7.633921	-2.171593	180
9	-7.637480	-2.203572	80
10	-7.640177	-2.199120	90

    - Depth of each tree:

	mean_test_score	mean_train_score	param_model__max_depth
0	-7.496697	-5.072578	11
1	-7.506729	-4.723843	12
2	-7.516061	-4.379089	13
3	-7.520916	-5.419586	10
4	-7.528310	-5.758460	9
5	-7.529124	-4.047270	14
6	-7.563647	-3.725714	15
7	-7.564116	-3.453639	16
8	-7.566298	-6.091478	8
9	-7.578409	-3.201785	17
10	-7.586883	-2.989634	18

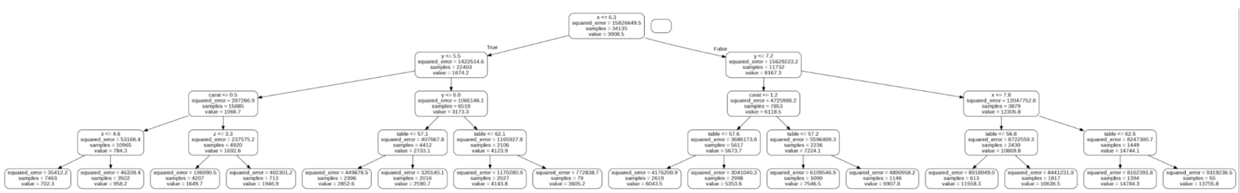
- How these hyper-parameters affect the overall performance:
    - Maximum number of features: fewer number of features (20%~30%) lead to better result in both dataset
    - Number of trees: Larger number of trees help them model to better
    - Depth of each tree: Deeper
    - **Maximum number of features** tend to have higher regularization effect

Question 20

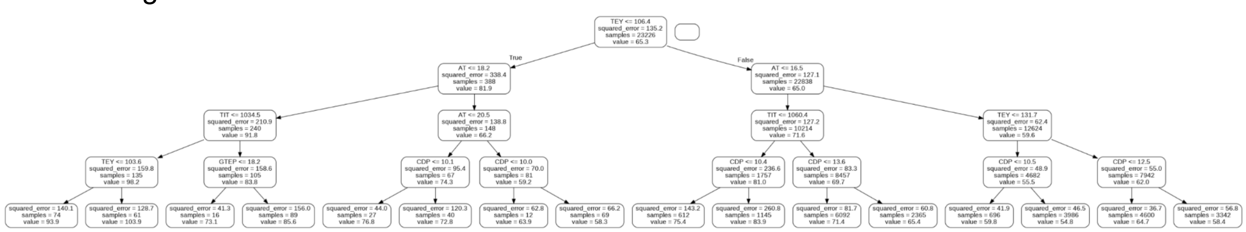
- Random forest is able to process features with different qualities. By bootstrapping, random forest selects the best (set of) features without additional preprocessing and avoids the model overfits on dominant features.

Question 21

- For diamond dataset:



    - x is selected for branching at the root node
    - We should conclude that x is the most important feature in this tree
    - Yes, it matches our result in 3.2.1
- For gas dataset:



    - TEY is selected for branching at the root node
    - We should conclude that TEY is the most important feature in this tree
    - Yes, it matches our result in 3.2.1

Question 22

- We decide to perform on **diamond dataset**
- LightGBM:
    - Hyperparameters:
        - max_depth: range from 1 to 30
        - n_estimators: range from 10 to 200 with an interval of 10
        - lambda_l2: range from 10 to 100 with an interval of 10

- CatBoost:
    - Hyperparaters

        - max_depth: range from 1 to 9
        - n_estimators: range from 10 to 200 with an interval of 10
        - l2_leaf_reg: range from 1 to 30

Question 23

- LightGBM:
    - Below are the top 5 hyperparameter combinations with minimum RMSE.
    - The best hyperparameters:
        - max_depth: 26
        - n_estimators: 40
        - lambda_l2: 90
    - Minimum training RMSE: **1306.2860**
    - Minimum testing RMSE: **1420.9413**

	mean_test_score	mean_train_score	param_model__max_depth	param_model__n_estimators	param_model__lambda_l2
13	-1420.941270	-1306.286014	26	40	90
14	-1420.941270	-1306.286014	17	40	90
15	-1422.277001	-1302.957209	17	40	60
16	-1423.256684	-1283.001509	27	80	90
17	-1423.473742	-1283.763837	27	80	100

- CatBoost:
    - Below are the top 5 hyperparameter combinations with minimum RMSE.
    - The best hyperparameters:
        - max_depth: 9
        - n_estimators: 200
        - l2_leaf_reg: 11
    - Minimum training RMSE: **1324.3227**
    - Minimum testing RMSE: **1429.5393**

	mean_test_score	mean_train_score	param_model__max_depth	param_model__n_estimators	param_model__l2_leaf_reg
8	-1429.539339	-1324.322736	9	200	11
9	-1429.920840	-1325.982611	9	190	11
11	-1430.126213	-1326.324099	9	200	15
12	-1430.653384	-1327.928723	9	190	15
13	-1431.039571	-1328.535176	9	200	20

Question 24

- LightGBM:
    - N_estimators helps the performance, determining the model's accuracy.
    - Max_depth affects fitting efficiency. This parameter is an integer that controls the maximum distance between the root node of each tree and a leaf node. Decrease max_depth to reduce training time.

        - Lambda_l2 affects regularization.

- CatBoost:
    - N_estimators helps the performance, determining the model's accuracy.
    - Max_depth affects fitting efficiency. This parameter is an integer that controls the maximum distance between the root node of each tree and a leaf node. Decrease max_depth to reduce training time.
    - l2_leaf_reg affects regularization.

Question 25

- We use the optimal RandomForestRegressor to perform the 10-fold CV.
- Diamond dataset:
    - RMSE for train data= 1275.6840014315767
    - RMSE for val data= 1433.9212681958873
- Gas dataset:
    - RMSE for train data= 5.683679914626244
    - RMSE for val data= 7.415761073155126
- The RMSE for the validation set is generally higher than that of the training set, and it might be due to the overfitting problem. Besides, the best feature in training set might not be the best feature in the validation set

Question 26

- By applying bootstrapping, some data may be randomly sorted out of the tree, then the RMSE for such data is called "Out-of-Bag Error" (OOB). R2 evaluates how well the model is trained on the training set.
- For diamond dataset:
    - Best Random Forest Model for Diamond Dataset:
    - OOB score: 0.8847
    - R^2 score: 0.8927
- For gas dataset:
    - Best Random Forest Model for Gas Turbine Dataset:
    - OOB score: 0.6984
    - R^2 score: 0.7548

Twitter Part

Question 27

Report for #gohawks
- Average number of tweets per hour: 292.48785062173687
- Average number of followers: 2217.9237355281984
- Average number of retweets: 2.0132093991319877

Report for #gopatriots

- Average number of tweets per hour: 40.95469800606194
- Average number of followers: 1427.2526051635405
- Average number of retweets: 1.4081919101697078

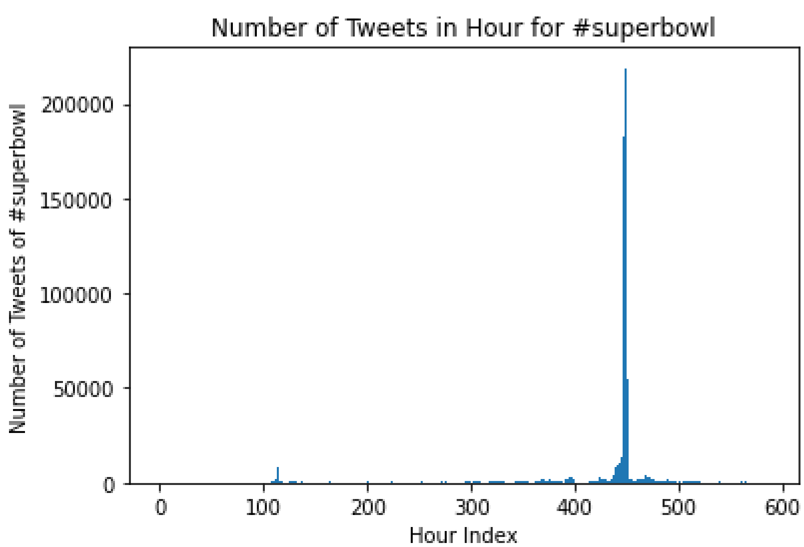
Report for #nfl
- Average number of tweets per hour: 397.0213901819841
- Average number of followers: 4662.37544523693
- Average number of retweets: 1.5344602655543254

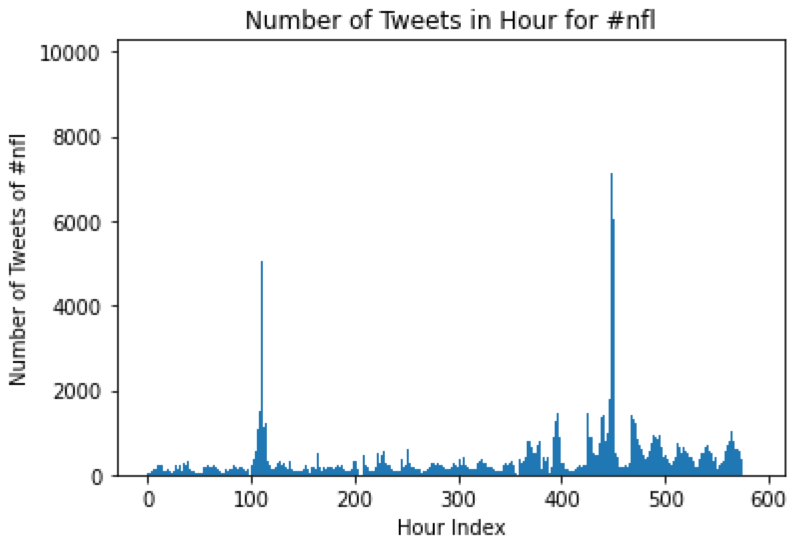
Report for #patriots
- Average number of tweets per hour: 750.89426460689
- Average number of followers: 3280.4635616550277
- Average number of retweets: 1.7852871288476946

Report for #sb49
- Average number of tweets per hour: 1276.8570598680474
- Average number of followers: 10374.160292019487
- Average number of retweets: 2.52713444111402

Report for #superbowl
- Average number of tweets per hour: 2072.11840170408
- Average number of followers: 8814.96799424623
- Average number of retweets: 2.3911895819207736

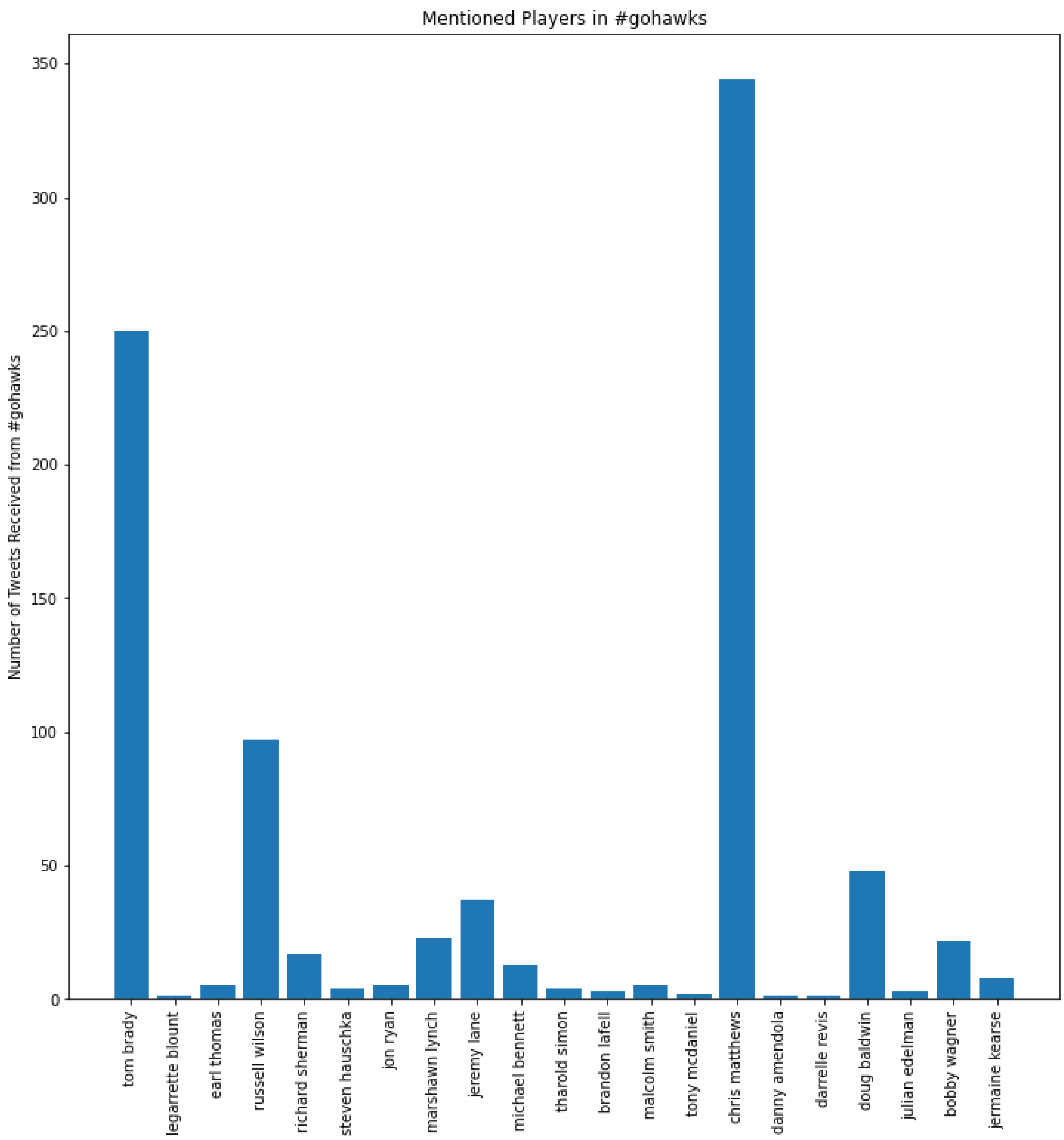
Question 28

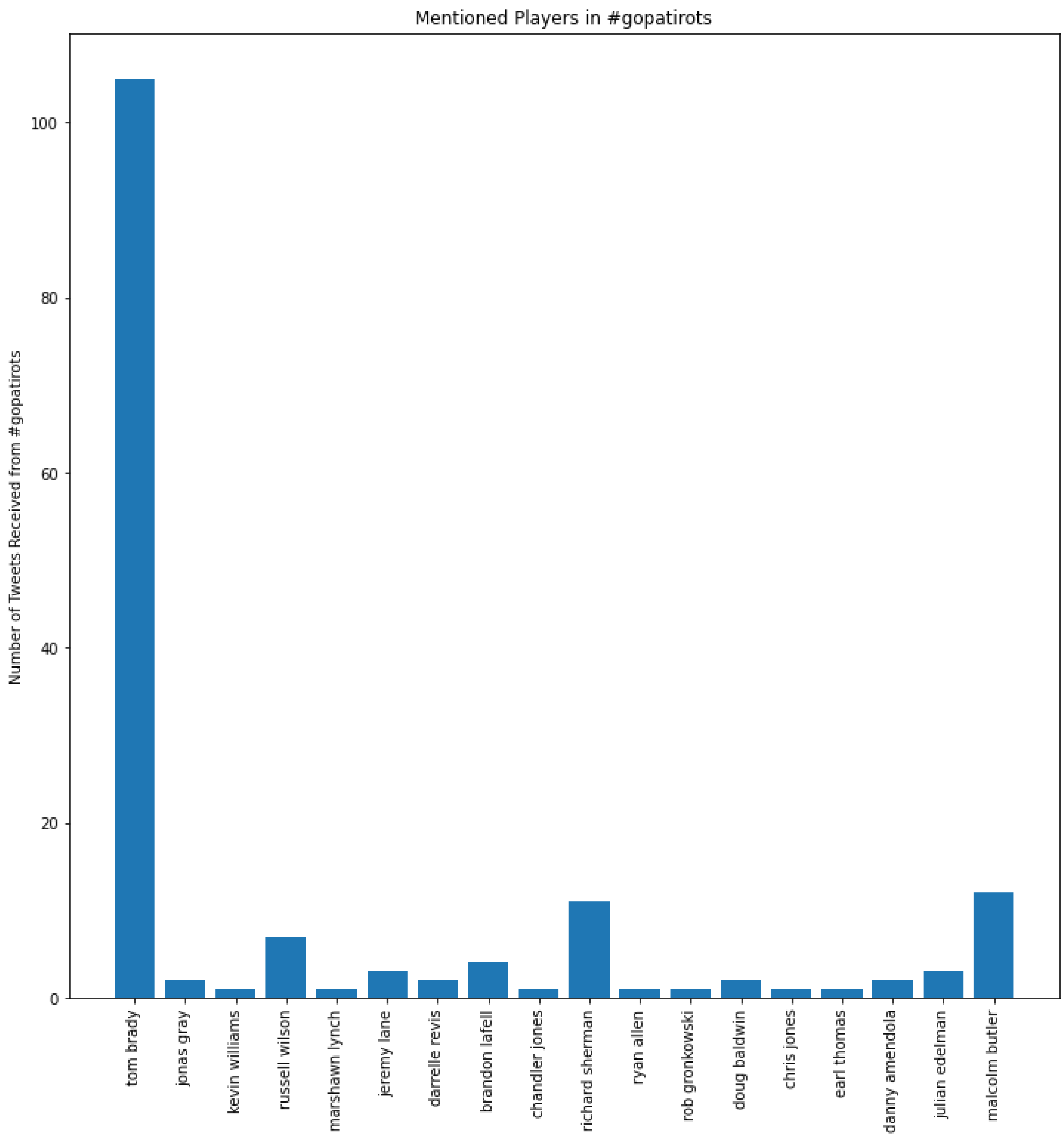




Question 29

- In this design problem, we mainly proposed two tasks, which will be discussed later in detail. Since the original data is too big, we decided to only focus on the tweets posted during the game. The time bounds we used are:
    - datetime.datetime(2015, 2, 1, 23, 15, 0, 0, pst_tz)
    - datetime.datetime(2015, 2, 2, 3, 15, 0, 0, pst_tz)
- Besides, we filtered out tweets that are non-English, and removed urls, hashtags, tags/retweets/replies, etc (noise). Here are the counts of tweets we found for two teams.
    - Number of tweets posted #gohawks: 25875
    - Number of tweets posted #gopatriots: 6933
- Among these tweets, we need to find the tweets that mention the players from both teams only. We used a NER model from **SpaCy** to get the entity type of each word. If a word's entity is **PERSON** and the name appears in the player lists from either team, we then include the corresponding tweet for further processing. Here are the number of tweets mentioning each player during the game.


Mentioned Players in #gohawks
Number of Tweets Received from #gohawks
0
50
100
150
200
250
300
350
tom brady
legarrette blount
earl thomas
russell wilson
richard sherman
steven hauschka
jon ryan
marshawn lynch
jeremy lane
michael bennett
tharold simon
brandon lafell
malcolm smith
tony mcdaniel
chris matthews
danny amendola
darrelle revis
doug baldwin
julian edelman
bobby wagner
jermaine kearse



- Since most of the players have low frequency of mentioning, we only consider the top 5 mentioned players.

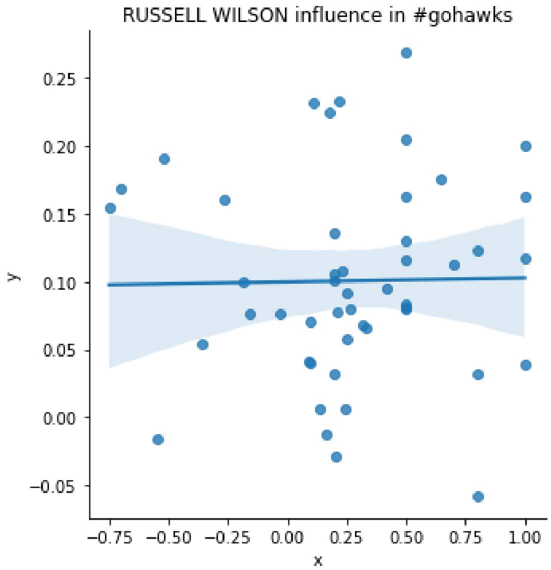
	player	mentioned times
26	tom brady	355
4	chris matthews	344
22	russell wilson	104
7	doug baldwin	50
9	jeremy lane	40

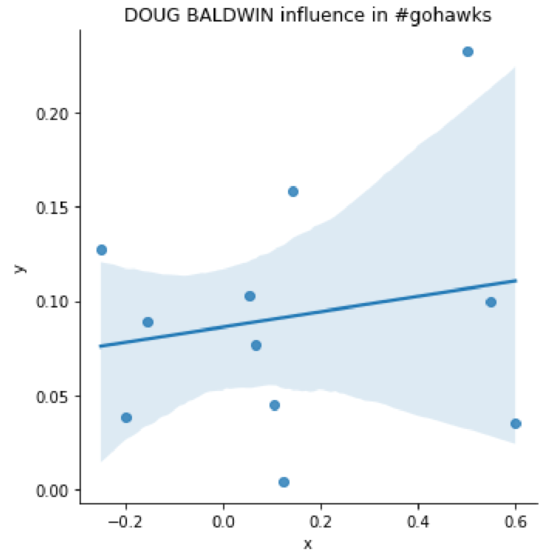
- The top 5 players we are considering are:
    - `'chris matthews',`
    - `'doug baldwin',`
    - `'jeremy lane',`
    - `'russell wilson',`
    - `'tom brady'`
- From the given link of the significant events, we pinpointed the timestamp of the following events, and stored them as a dictionary:

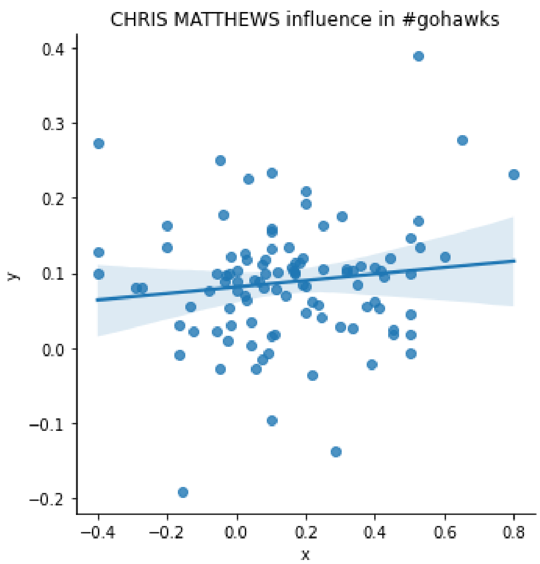
```
- big_events = {
-     1422836015 - 60: 7,         # touchdown P 7: 0
-     1422837198 - 60: 0,         # touchdown H 7: 7
-     1422838767 - 6 * 60: 7,     # touchdown P 14: 7
-     1422838767 - 60: 0,         # touchdown H 14: 14
-     1422841327 - 60: -3,        # field goal H 14: 17
-     1422841327 + 3 * 60: -3,    # interception H 14: 17
-     1422842399 - 60: -10,       # touchdown H 14: 24
-     1422844127 - 60: -3,        # touchdown P 21: 24
-     1422845305 - 60: 4,         # touchdown P
-     1422846605 - 5 * 60: 4,     # interception P
-     1422846605 : 4              # game: P won
- }
```

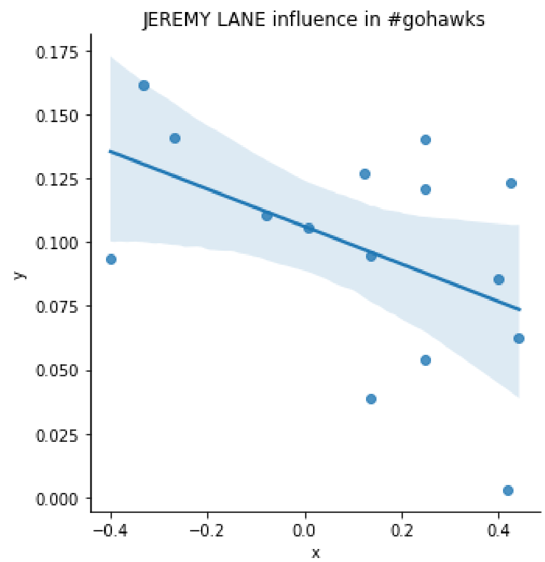
- The value of each event is the difference between the scores of two teams. A positive score means that the Patriots is leading.

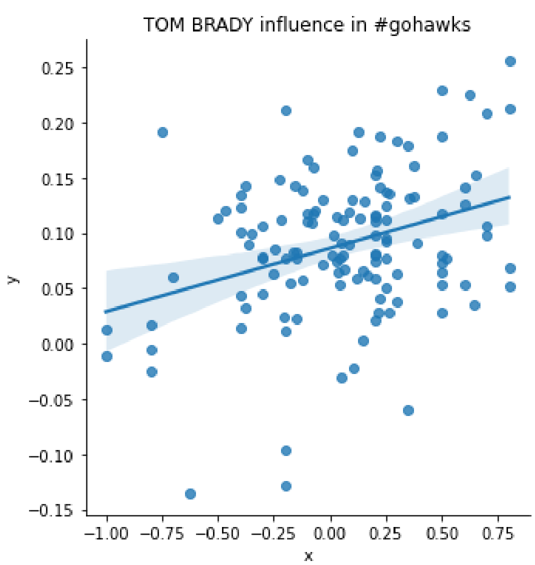
- The first task is that we want to see **how influential a player is**. Specifically, we measure the sentiment level in time of the top 5 players, and try to find out if the sentiment towards each player has an impact on the **overall** sentiment of all people. Since we filtered out some samples, we decided to use a comparatively small time window, which is 10 seconds.


RUSSELL WILSON influence in #gohawks
y
x


DOUG BALDWIN influence in #gohawks
y
x

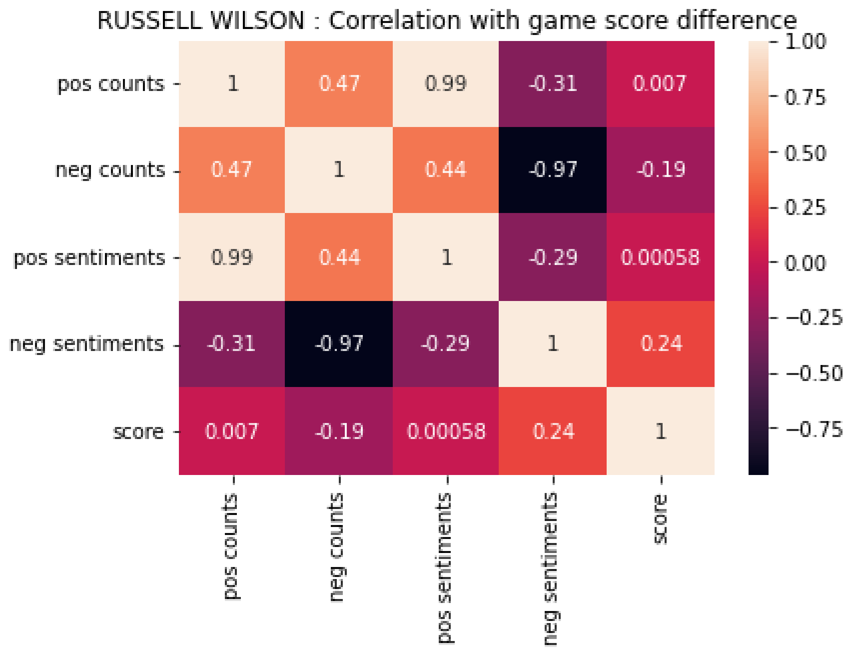

CHRIS MATTHEWS influence in #gohawks
y
x


JEREMY LANE influence in #gohawks
y
x

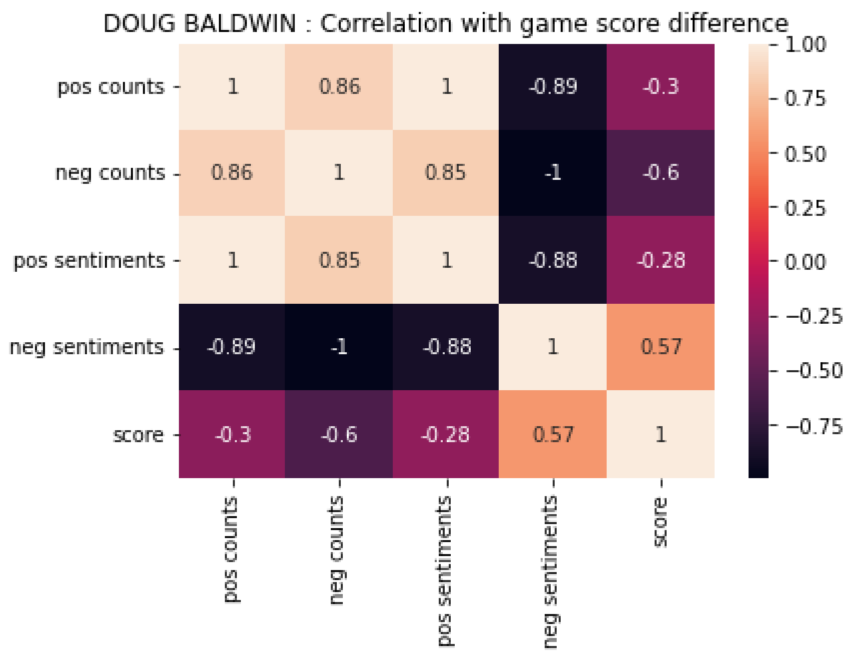

TOM BRADY influence in #gohawks
y
x

- From the scatter plots above, we can see that the sentiments towards “Tom Brady” and “Chris Matthews” have a comparatively higher correlation with the overall sentiment level of the public. (The test on #gopatriots has very few data points so we decided to omit it).

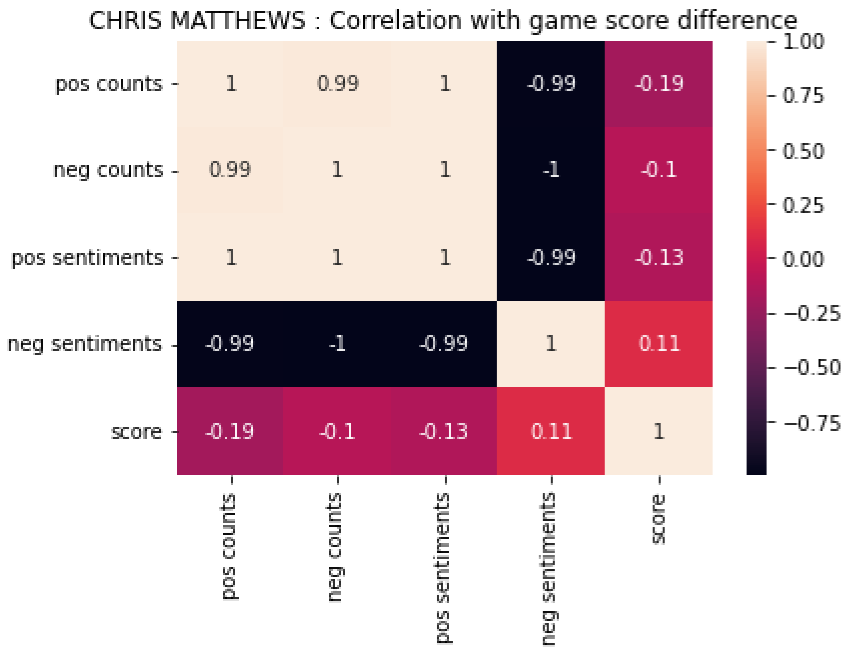
- **The second task is that, if we take the number of positive tweets, and the number of negative tweets, and the sentiment level towards each player in the periods of time when a significant event happened, can we predict the score difference between two teams.**
- As mentioned above, we already have the timestamp of each big event (touchdown, field goal, interception). We construct the ranges of the big events. The range of an event is from **the time when the event happened** to **the time when the next event happened**. For the range of the last event, we set it from the time when it happened to the max timestamp recorded. Here are the timestamp ranges of all 11 events.
    - `[(1422835955, 1422837138),`
    - `(1422837138, 1422838407),`
    - `(1422838407, 1422838707),`
    - `(1422838707, 1422841267),`
    - `(1422841267, 1422841507),`
    - `(1422841507, 1422842339),`
    - `(1422842339, 1422844067),`
    - `(1422844067, 1422845245),`
    - `(1422845245, 1422846305),`
    - `(1422846305, 1422846605),`
    - `(1422846605, 1422846900)]`
- The features we need to get are
    - **the number of positive tweets** in each range
    - **the number of negative tweets** in each range
    - **the sentiment level** in each range
- A positive tweet can reflect how a fan responded to the happening of a significant event. It usually happened when their supported team led the game. Same for the negative tweets. Additionally, the sentiment can be an indicator of the team that the mentioned player belonged to.
- The target variable in this case will be the score difference between two teams. As mentioned before, since the sample size of #gopatriots selected is not big enough, we mainly focus on #gohawks.
- Here are the heatmap of the top 5 mentioned players based on #gohawks:


RUSSELL WILSON : Correlation with game score difference
pos counts
neg counts
pos sentiments
neg sentiments
score
1
0.47
0.99
-0.31
0.007
0.47
1
0.44
-0.97
-0.19
0.99
0.44
1
-0.29
0.00058
-0.31
-0.97
-0.29
1
0.24
0.007
-0.19
0.00058
0.24
1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

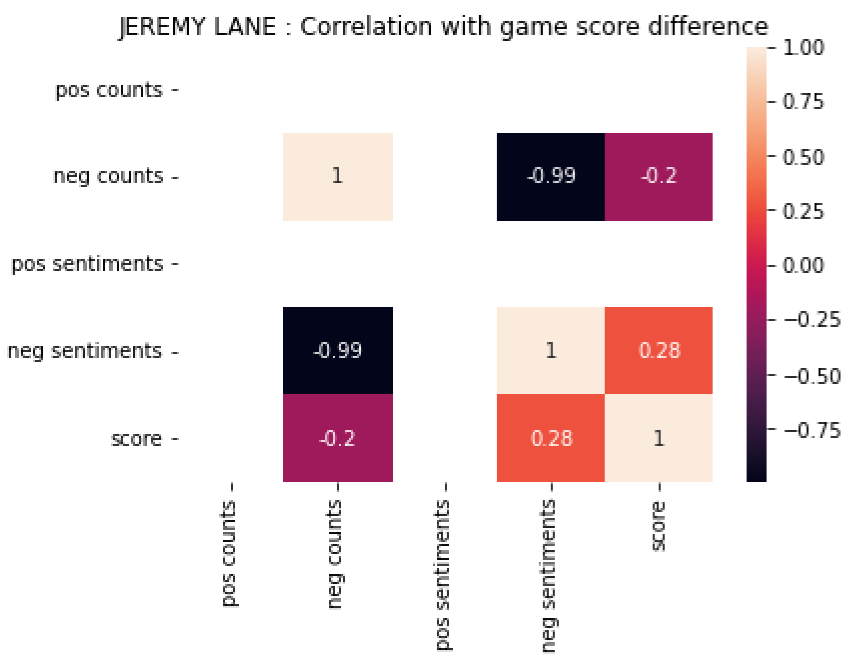
-


DOUG BALDWIN : Correlation with game score difference
pos counts
neg counts
pos sentiments
neg sentiments
score
1
0.86
1
-0.89
-0.3
0.86
1
0.85
-1
-0.6
1
0.85
1
-0.88
-0.28
-0.89
-1
-0.88
1
0.57
-0.3
-0.6
-0.28
0.57
1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

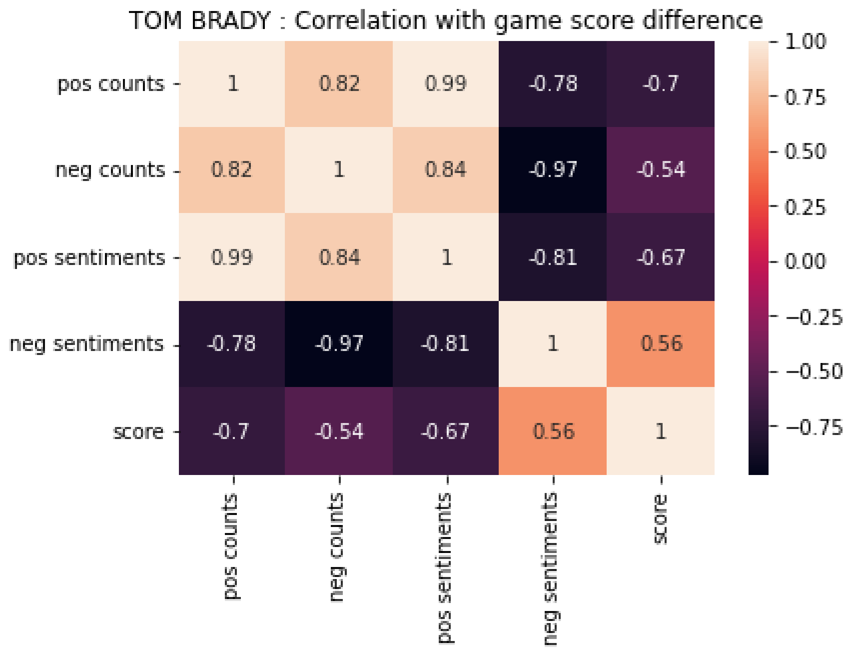
-


CHRIS MATTHEWS : Correlation with game score difference
pos counts
neg counts
pos sentiments
neg sentiments
score
1 0.99 1 -0.99 -0.19
0.99 1 1 -1 -0.1
1 1 1 -0.99 -0.13
-0.99 -1 -0.99 1 0.11
-0.19 -0.1 -0.13 0.11 1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

-


JEREMY LANE : Correlation with game score difference
pos counts
neg counts
pos sentiments
neg sentiments
score
1 -0.99 -0.2
-0.99 1 0.28
-0.2 0.28 1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

-



- Notice that Player “Jeremy Lane” has no positive tweets received so the corresponding area is blank. Besides, “Tom Brady” has the highest **absolute** correlation with the score difference (pos counts). It’s negative because we are testing it on #gohawks where we assume that people who tweeted with “#gohawks” are all Seahawks fans whereas Tom Brady is a Patriots player. It also indicates that Tom Brady was the potential MVP of the game.
- After getting the Xs and Ys, we can then try out different ML models. The models we tried include:
    - `LinearRegression,`
    - `Ridge,`
    - `Lasso`
- Here are the test scores of these models.
- Linear Regression
- `RUSSELL WILSON Linear Regression`
- `   fit_time  score_time  test_score  train_score`
- `0  0.004483    0.001956   -3.760876    -4.841501`
- `1  0.004268    0.001851   -4.104803    -4.832058`
- `2  0.004944    0.002641   -4.512260    -4.791824`
- `3  0.004312    0.001779   -6.720063    -4.701494`
- `4  0.004344    0.001765   -7.378392    -4.482794`
- `DOUG BALDWIN Linear Regression`
- `   fit_time  score_time  test_score  train_score`
- `0  0.003969    0.001738   -2.000000    -3.405877`
- `1  0.003964    0.001712   -2.000000    -3.405877`
- `2  0.004018    0.001725   -2.000000    -3.405877`
- `3  0.004050    0.001734   -2.500000    -3.376389`
- `4  0.005413    0.002019   -3.866391    -3.177300`

- CHRIS MATTHEWS Linear Regression
-    fit_time score_time test_score train_score
- 0 0.004062   0.001710   -0.336866   -2.663301
- 1 0.005712   0.002029   -0.637206   -2.659985
- 2 0.004138   0.001721   -2.486149   -2.637850
- 3 0.004949   0.001845   -2.967170   -2.513994
- 4 0.004494   0.001730   -3.116229   -2.504077
- JEREMY LANE Linear Regression
-    fit_time score_time test_score train_score
- 0 0.004094   0.001726   -1.250   -3.339162
- 1 0.004274   0.002021   -3.250   -3.217142
- 2 0.004838   0.001953   -3.250   -3.217142
- 3 0.004796   0.002267   -3.250   -3.217142
- 4 0.004081   0.001715   -4.625   -3.063903
- TOM BRADY Linear Regression
-    fit_time score_time test_score train_score
- 0 0.005746   0.002045   -1.139897   -3.003168
- 1 0.004815   0.002097   -2.714933   -2.923441
- 2 0.004857   0.002002   -3.258032   -3.009340
- 3 0.004829   0.002014   -3.985794   -2.806260
- 4 0.005395   0.002255   -4.038981   -2.873049
-
- Ridge
- RUSSELL WILSON Ridge
-   mean_test_score mean_train_score param_poly_transform__degree
- 0   -4.218988   -2.863532   1
- 1   -11.872101   -0.866559   2
- 2   -25.350811   -0.772904   5
- 3   -31.751527   -0.793406   3
- 4   -33.065344   -0.778602   4
- DOUG BALDWIN Ridge
-   mean_test_score mean_train_score param_poly_transform__degree
- 0   -4.218988   -2.863532   1
- 1   -11.872101   -0.866559   2
- 2   -25.350811   -0.772904   5
- 3   -31.751527   -0.793406   3
- 4   -33.065344   -0.778602   4
- CHRIS MATTHEWS Ridge
-   mean_test_score mean_train_score param_poly_transform__degree
- 0   -4.218988   -2.863532   1
- 1   -11.872101   -0.866559   2
- 2   -25.350811   -0.772904   5
- 3   -31.751527   -0.793406   3
- 4   -33.065344   -0.778602   4
- JEREMY LANE Ridge
-   mean_test_score mean_train_score param_poly_transform__degree

- 0   -4.218988   -2.863532   1
- 1   -11.872101   -0.866559   2
- 2   -25.350811   -0.772904   5
- 3   -31.751527   -0.793406   3
- 4   -33.065344   -0.778602   4
- TOM BRADY Ridge
-   mean_test_score mean_train_score param_poly_transform__degree
- 0   -4.218988   -2.863532   1
- 1   -11.872101   -0.866559   2
- 2   -25.350811   -0.772904   5
- 3   -31.751527   -0.793406   3
- 4   -33.065344   -0.778602   4
-
- Lasso
- RUSSELL WILSON Lasso
-    mean_test_score mean_train_score param_poly_transform__degree
- 0   -251.183960   -3.804564   5
- 1   -576.396709   -3.803392   6
- 2   -1242.863880   -3.803919   7
- 3   -2610.463045   -3.804755   8
- 4   -4905.971952   -3.805319   9
- DOUG BALDWIN Lasso
-    mean_test_score mean_train_score param_poly_transform__degree
- 0   -17.010939   -3.271431   5
- 1   -17.010939   -3.271431   6
- 2   -17.010939   -3.271431   7
- 3   -17.010939   -3.271431   8
- 4   -17.010939   -3.271431   9
- CHRIS MATTHEWS Lasso
-    mean_test_score mean_train_score param_poly_transform__degree
- 0   -1755.052006   -2.382248   5
- 1   -3568.388975   -2.382223   6
- 2   -6957.757831   -2.382190   7
- 3   -31799.674003   -2.382131   8
- 4   -55233.933949   -2.382123   9
- JEREMY LANE Lasso
-    mean_test_score mean_train_score param_poly_transform__degree
- 0   -6.050291   -3.178121   10
- 1   -6.050314   -3.178227   6
- 2   -6.050314   -3.178182   7
- 3   -6.050314   -3.178302   5
- 4   -6.050314   -3.178154   8
- TOM BRADY Lasso
-    mean_test_score mean_train_score param_poly_transform__degree

- 0 -23.076516 -1.394640 5
- 1 -23.100587 -1.394142 6
- 2 -23.110426 -1.393988 7
- 3 -23.115393 -1.393931 8
- 4 -23.118265 -1.393905 9

- From the test results above, we can see that Linear Regression performed the best whereas Ridge somehow failed, and Lasso had an overfitting problem.
- Our assumption is that the only one or two features we selected are valuable in terms of predicting the score difference between two teams. Plus the number of samples selected is way too less and consequently Ridge and Lasso performed poorly.

- A way to improve the performance of our model will be increasing the number of events such as successful passes, distance proceeded, possession time, etc. This will increase the number of datapoints that can be used to predict the score.

- If we predict the score difference using X where the positive tweets count for Tom Brady is very high, then we will get a large negative number meaning that the Seahawks is leading the game, which makes sense because as mentioned before Tom Brady is a Patriots player. In other words, the positive tweets are more like teasing Tom Brady for his turn-overs.