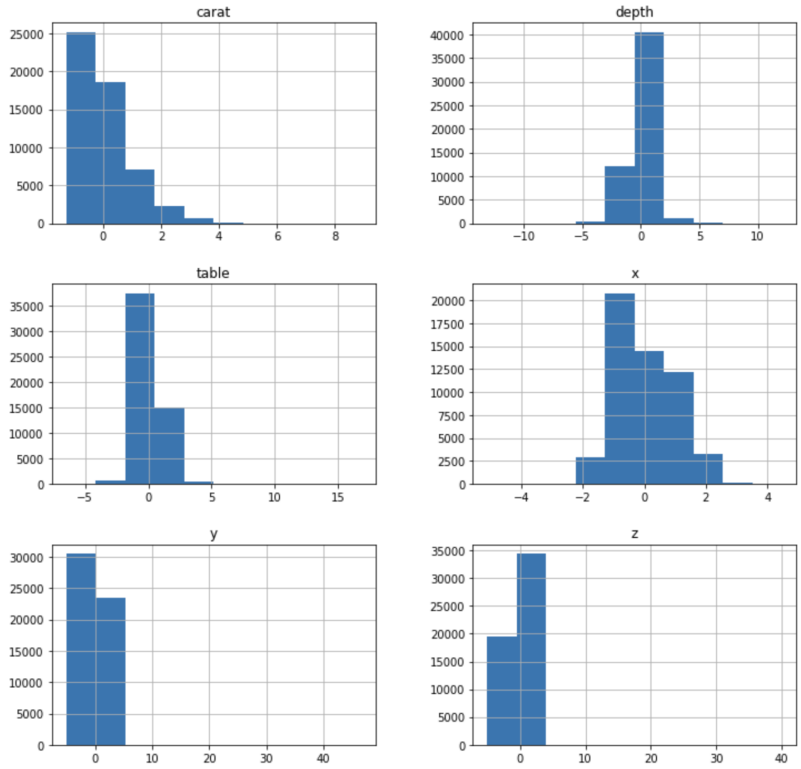
Project 4_ Report (Diamond dataset)
Haoting Ni (905545789), Yikai Wang (905522085), Yuanxuan Fang (005949389)

Question 1.1:

The heatmap is shown as above. Since our target feature is price for the diamond dataset, from the heatmap, we can tell that features of carat has the highest correlation value with 0.92. This makes sense because the bigger carat, the more expansive diamonds are.

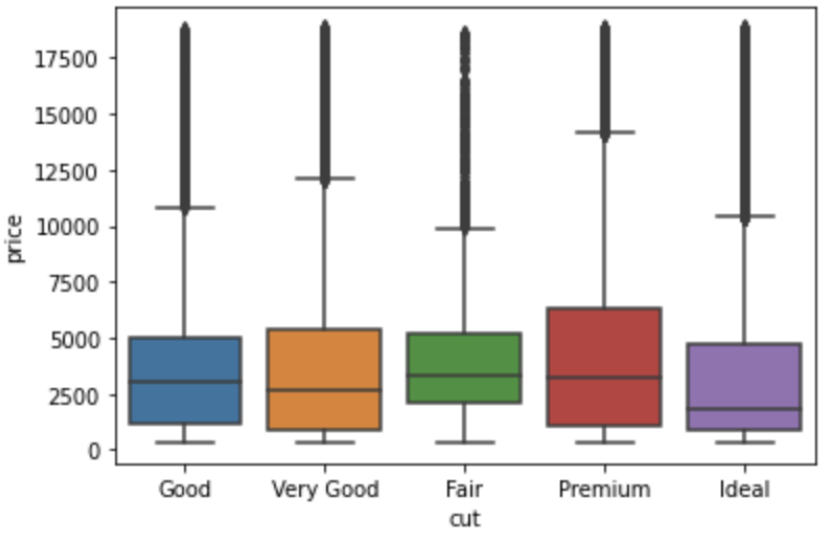
Question 1.2:

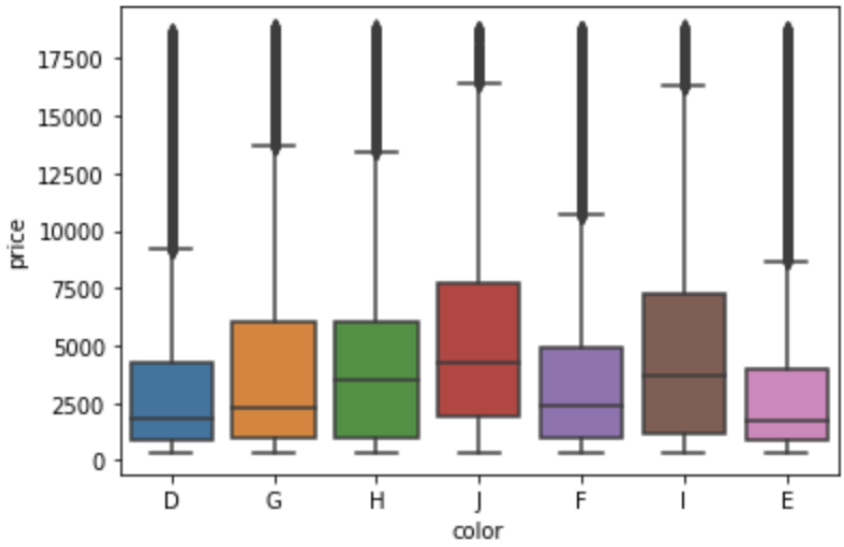
If a feature has high skewness, we can conduct nonlinear transformation and standardization for the feature to remove skewness.

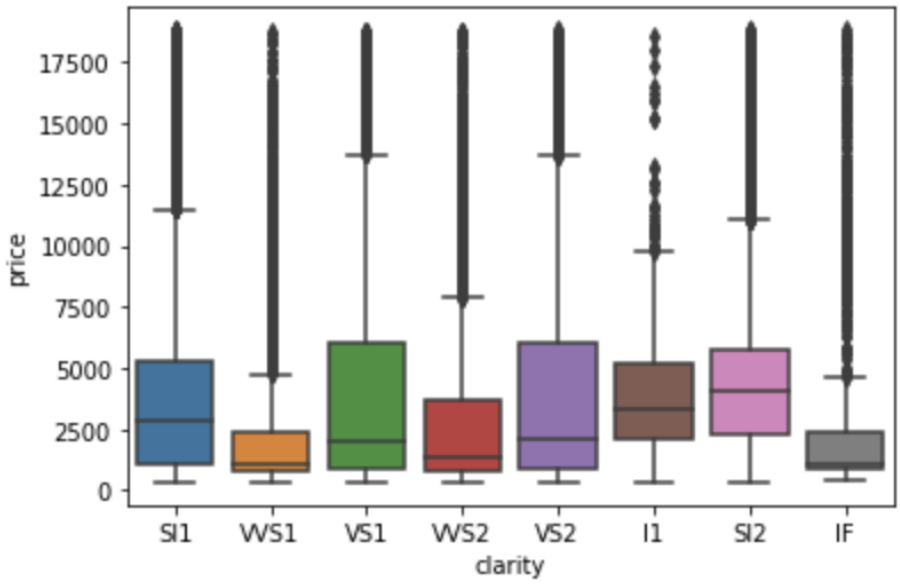


The histogram of the numerical features are shown above.

Question 1.3:







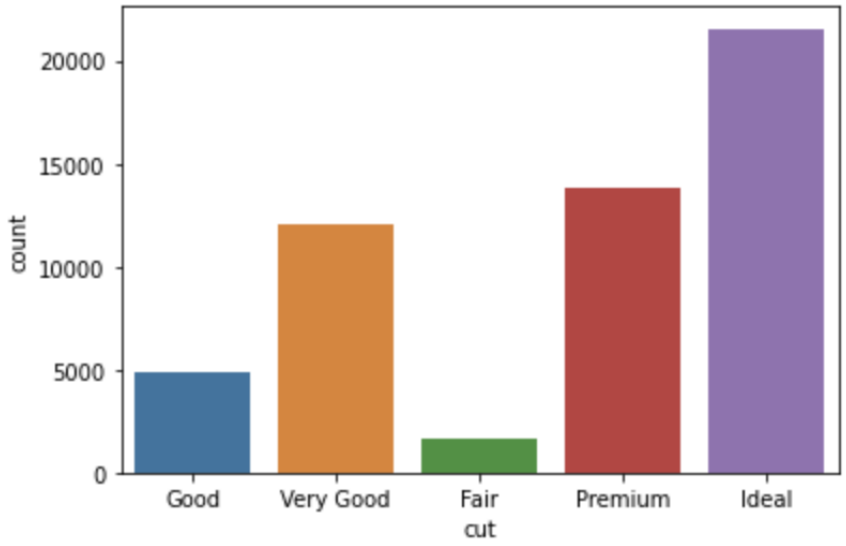
From the boxplot with categorical features we found that for each feature there are statistically separate. Prices are various depends on the features.

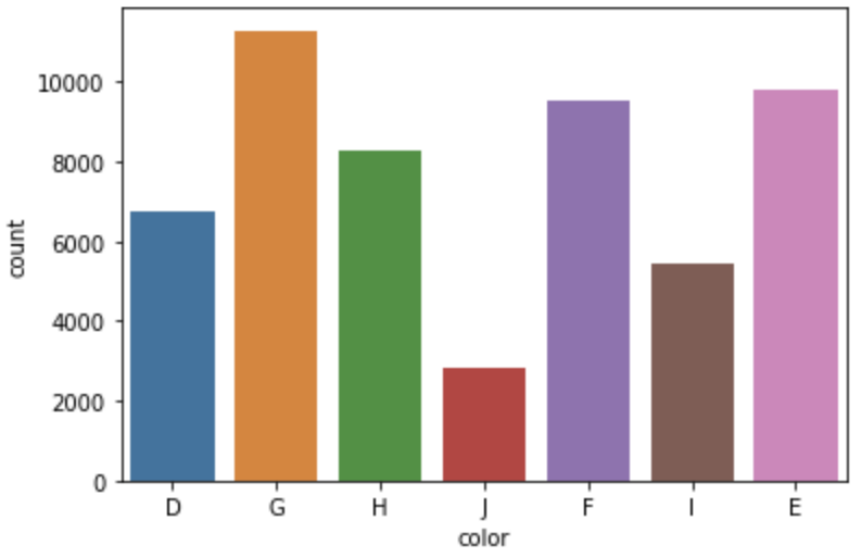
For the cut feature, premium cut has the wide range of the price.

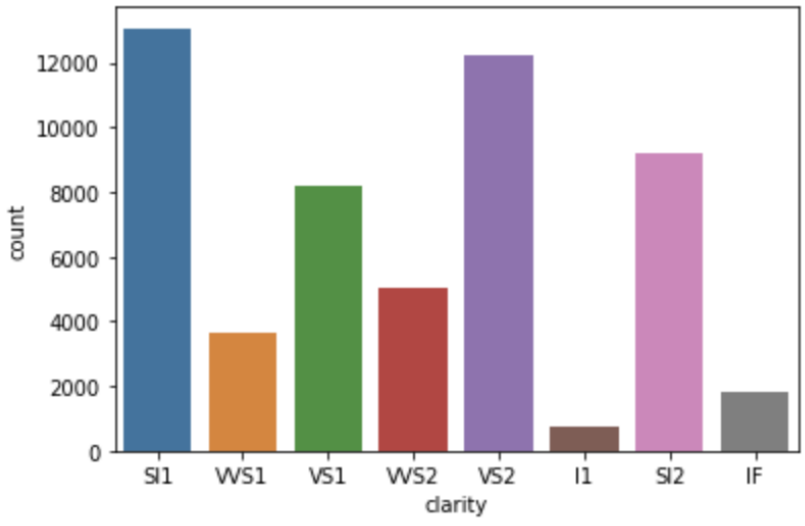
For the color feature, I and J color have the wide range of price. D has the narrow range.

For the clarity feature, VS1 and VS2 have the wide range of price.

Question 1.4: Counts by cut, color and clarity.







Question 2.1:

	carat	cut	color	clarity	depth	table	price	x	y	z
0	-1.198168	0.981473	0.063758	-1.245215	-0.174092	-1.099672	-0.903594	-1.587837	-1.536196	-1.571129
1	-1.240361	0.085889	0.063758	-0.638095	-1.360738	1.585529	-0.904346	-1.641325	-1.658774	-1.741175
2	-1.198168	-1.705279	0.063758	0.576145	-3.385019	3.375663	-0.904095	-1.498691	-1.457395	-1.741175
3	-1.071587	0.085889	-1.471547	-0.030975	0.454133	0.242928	-0.901839	-1.364971	-1.317305	-1.287720
4	-1.029394	-1.705279	-2.239199	-1.245215	1.082358	0.242928	-0.901588	-1.240167	-1.212238	-1.117674
...	...	...	...	...	...	...	...	...	...	...
53935	-0.164427	0.981473	1.599062	-0.638095	-0.662711	-0.204605	-0.294982	0.016798	0.022304	-0.054888
53936	-0.164427	-1.705279	1.599062	-0.638095	0.942753	-1.099672	-0.294731	-0.036690	0.013548	0.100988
53937	-0.206621	-0.809695	1.599062	-0.638095	0.733344	1.137995	-0.294480	-0.063434	-0.047741	0.030135
53938	0.130927	0.085889	-0.703895	-1.245215	-0.523105	0.242928	-0.295232	0.373383	0.337506	0.285204
53939	-0.101137	0.981473	1.599062	-1.245215	0.314528	-1.099672	-0.294230	0.088115	0.118616	0.143499

53940 rows × 10 columns

After the standardization, the dataset looks like the graph above.

Question 2.2:

```
from sklearn.feature_selection import mutual_info_regression, f_regression
import numpy as np
mi = mutual_info_regression(diamonds_std_df, diamonds_std_df['price'])
mi /= np.max(mi)
mi_features = diamonds_std_df.columns[np.argsort(mi)[::-1]]
print("Features in Mutual information: {}".format(mi_features.tolist()))
print("Mutual information: {}".format(np.sort(mi)[::-1]))
```

```
Features in Mutual information: ['price', 'carat', 'y', 'x', 'z', 'clarity', 'color', 'cut', 'table', 'depth']
Mutual information: [1.         0.19103625 0.16444164 0.16339115 0.15751541 0.02539437
 0.01330984 0.00686466 0.00426242 0.00388006]
```

+ Code + Text

```
f_score, _ = f_regression(diamonds_std_df, diamonds_std_df['price'])
f_score /= np.max(f_score)
f_features = diamonds_std_df.columns[np.argsort(f_score)[::-1]]
print("Features in decreasing F score with target: {}".format(f_features.tolist()))
print("Decreasing F score: {}".format(np.sort(f_score)[::-1]))
```

```
Features in decreasing F score with target: ['price', 'carat', 'x', 'y', 'z', 'color', 'clarity', 'table', 'cut', 'depth']
Decreasing F score: [1.00000000e+00 8.26107287e-14 5.26395335e-14 4.37207536e-14
 4.20926209e-14 3.23665553e-16 3.22781284e-16 2.40758456e-16
 4.20549092e-17 1.66167889e-18]
```

From the mutual information and F score we got from the above, we choose to select feature carat, x, y, z as our main features that works on.

This step will improve our regression performances because we will select features which are the most relative to the price. We ignore the features that are less relative to the target feature, price.(这里还得list lowest mi features)

Question 3:

From the link website, for OOB, it bootstrap the train dataset on the random forest tree, some data might not work for the tree, and the RMSE for that data is out of bag error.

For R square score, it is defined by the ratio of target's variance by the characteristic in the dataset. It access all the data points in the training dataset. In other words, R square score shows how well the data fit the regression model.

Linear regression:

Question 4.1:

For ordinary least square, there is no regulation on it.

For lasso regression, it uses L1 regularization. It adds the "absolute value of magnitude" of the coefficient to the loss function.

For ridge regression, it uses L2 regularization. It adds the "squared magnitude" of the coefficient to the loss function.

Both regulatization methods encourage weight to be small, and produce sparse solution. It might prevent from overfitting.

Question 4.2:

```
# ordinary least squares (linear regression without regularization)
train_rmse, test_rmse = train(LinearRegression(), X_feature, y)
print(train_rmse, test_rmse)
```

```
0.3791444488323997 0.3521869849587446
```

```
# Lasso (choice of the best regularization scheme along with the optimal penalty parameter)
alpha_list = [0.00001,0.0001,0.001,0.01,0.1,1, 10,100,1000]
rmse_training = []
rmse_validation = []
for i in alpha_list:
    train_rmse, test_rmse = train(Lasso(alpha=i), X_feature, y)
    rmse_training.append(train_rmse)
    rmse_validation.append(test_rmse)
best_rmse = np.min(rmse_validation)
index = rmse_validation.index(best_rmse)
best_alpha = alpha_list[index]
print("The best regularization scheme with smallest validation RMSE: ", best_rmse, " with ", best_alpha)
```

```
The best regularization scheme with smallest validation RMSE:  0.3522049662159326  with  1e-05
```

```
# Ridge (choice of the best regularization scheme along with the optimal penalty parameter)
alpha_list = [0.00001,0.0001,0.001,0.01,0.1,1, 10,100,1000]
rmse_training = []
rmse_validation = []
for i in alpha_list:
    train_rmse, test_rmse = train(Ridge(alpha=i), X_feature, y)
    rmse_training.append(train_rmse)
    rmse_validation.append(test_rmse)
best_rmse = np.min(rmse_validation)
index = rmse_validation.index(best_rmse)
best_alpha = alpha_list[index]
print("The best regularization scheme with smallest validation RMSE: ", best_rmse, " with ", best_alpha)
```

```
The best regularization scheme with smallest validation RMSE:  0.35218698522154535  with  1e-05
```

As above result shows, for Lasso regression, the smallest rmse is 0.3522049662159326 with alpha 0.00001. For Ridge Regression, the smallest rmse is 0.0.35218698522154535 wit号

alpha 0.00001. The alpha list value we choose is : alpha_list = [0.00001,0.0001,0.001,0.01,0.1,1, 10,100,1000].

From the validation rmse result, we can see that ridge regression has really close to the ordinary least squares' but it is slightly bigger than ridge's. Thus, the optimal choice is linear regression without regularization.

Question 4.3:

For linear regression, it does not play role because scaling does not affect mean test score. Standanrization does not change the coefficient.

For lasso and ridge, it plays a role because scaling affects mean test score. Standanrization changed the coefficient. It will improve the rmse mean by reducing its mean.

Question 4.4:

P value is the probability of obtaining test results at least as extreme as the result actually observed. When lower p value is respect to the feature it means they are relevant. Otherwise, they are irrelevant.

Poly regression

Question 5.1:

As applying the grid search on different degrees from 2 to 6, arranging them by negative root mean square error, we got the top 5 most salient features are:

```
Top 5 Salient features: ['y', 'carat', 'x', 'carat^2', 'z']
```

Thus, predicting the diamond price might be a polynomial combination of these features.

Question 5.2:

The best degree of polynomial is 2. We found it by tried different degrees from 2 to 6, train with certrain polynomial degree with ridge model. Then, we compared the RMSE result, the best degree is the one with the smallest validation error.

High-order polynomial implies a complex model, it might better fit the training data, but also might lead to overfitting. So we need to focus on the test set error to decide the best degree of polynomial.

Question 6 (Neural Network):

Q 6.1

First element the para is hidden_layer_sizes, second is weight decay regularization, third is learning rate.

Following is all the combination we tried:

```
10 Fold average training RMSE with para [(25, 50, 75), 0.0001, 0.0001]  is  0.33616200034040156,
10 Fold average testing RMSE with para [(25, 50, 75), 0.0001, 0.0001]  is  0.35888944418709673,
10 Fold average training RMSE with para [(25, 50, 75), 0.0001, 0.001]  is  0.3342262595922583,
10 Fold average testing RMSE with para [(25, 50, 75), 0.0001, 0.001]  is  0.3555272206561454,
10 Fold average training RMSE with para [(25, 50, 75), 0.1, 0.0001]  is  0.33909624836729135,
10 Fold average testing RMSE with para [(25, 50, 75), 0.1, 0.0001]  is  0.35314426432147594,
10 Fold average training RMSE with para [(25, 50, 75), 0.1, 0.001]  is  0.3370366621608094,
10 Fold average testing RMSE with para [(25, 50, 75), 0.1, 0.001]  is  0.35471041902971756,
10 Fold average training RMSE with para [(50, 75, 100), 0.0001, 0.0001]  is  0.33463566320555027,
10 Fold average testing RMSE with para [(50, 75, 100), 0.0001, 0.0001]  is  0.3545937386550367,
10 Fold average training RMSE with para [(50, 75, 100), 0.0001, 0.001]  is  0.33389909898947234,
10 Fold average testing RMSE with para [(50, 75, 100), 0.0001, 0.001]  is  0.3805581751667448,
10 Fold average training RMSE with para [(50, 75, 100), 0.1, 0.0001]  is  0.3368864400913931,
10 Fold average testing RMSE with para [(50, 75, 100), 0.1, 0.0001]  is  0.36252716820542247,
10 Fold average training RMSE with para [(50, 75, 100), 0.1, 0.001]  is  0.33693892447311424,
10 Fold average testing RMSE with para [(50, 75, 100), 0.1, 0.001]  is  0.3571657211123037,
10 Fold average training RMSE with para [(75, 50, 25), 0.0001, 0.0001]  is  0.33520541802218606,
10 Fold average testing RMSE with para [(75, 50, 25), 0.0001, 0.0001]  is  0.3558560319408496,
10 Fold average training RMSE with para [(75, 50, 25), 0.0001, 0.001]  is  0.3343422888582804,
10 Fold average testing RMSE with para [(75, 50, 25), 0.0001, 0.001]  is  0.36487559945219145,
10 Fold average training RMSE with para [(75, 50, 25), 0.1, 0.0001]  is  0.3368947287526566,
10 Fold average testing RMSE with para [(75, 50, 25), 0.1, 0.0001]  is  0.3758483899700766,
10 Fold average training RMSE with para [(75, 50, 25), 0.1, 0.001]  is  0.3386904663014517,
10 Fold average testing RMSE with para [(75, 50, 25), 0.1, 0.001]  is  0.34247487255644793,
10 Fold average training RMSE with para [(150, 150), 0.0001, 0.0001]  is  0.3348127875973722,
10 Fold average testing RMSE with para [(150, 150), 0.0001, 0.0001]  is  0.35467926384351567,
10 Fold average training RMSE with para [(150, 150), 0.0001, 0.001]  is  0.3352684021660421,
10 Fold average testing RMSE with para [(150, 150), 0.0001, 0.001]  is  0.3840747507395522,
10 Fold average training RMSE with para [(150, 150), 0.1, 0.0001]  is  0.3384506373743693,
10 Fold average testing RMSE with para [(150, 150), 0.1, 0.0001]  is  0.35694360736307074,
10 Fold average training RMSE with para [(150, 150), 0.1, 0.001]  is  0.33910551242846937,
10 Fold average testing RMSE with para [(150, 150), 0.1, 0.001]  is  0.3542853668023705,
Neural network, the best rmse with para [(75, 50, 25), 0.1, 0.001] is 0.34247487255644793
```

```
: print(best_hidden_layer_sizes)
  print(best_alpha_list)
  print(best_learning_rate_init)
  print(best_rmse)
```

```
(75, 50, 25)
0.1
0.001
0.34247487255644793
```

From above, we can see that the best parameter we obtained under 20 combinations is having hidden layer size (75,50,25), weight decay regularization 0.1 and learning rate 0.001.

Q 6.2

The best linear regression rmse is 0.3521870112389317 with Ridge regression , compared with neural network rmse which is 0.34247487255644793. The neural network has a 3% improvement. The performance is slightly better.

Q 6.3

Relu.

Q 6.4

If depth of the neural network is too far, the gradient will be smaller and smaller. It may lead to no learning at the end. Vanishing gradients may cause a bad local minima.

For a larger dataset, increasing the depth of network too far will result in a massive training time and large memory requirement.
It will also lead to overfitting, if the depth of network is too far. Because it will increase the complexity of the model and start to tune the model on training data set.

Q 7 (Random Forest)

First element the para is max_features, second is n_estimators, third is max_depth.

Following is all the combination we tried:
Average training RMSE with para [1, 50, 3] is 0.35096812604982974,
Average validation RMSE with para [1, 50, 3] is 0.3842190809806776,
Average training RMSE with para [1, 50, 5] is 0.33946809559059216,
Average validation RMSE with para [1, 50, 5] is 0.3673787970606872,
Average training RMSE with para [1, 50, 7] is 0.3315624383486824,
Average validation RMSE with para [1, 50, 7] is 0.3630324416942055,
Average training RMSE with para [1, 100, 3] is 0.3507835163550938,
Average validation RMSE with para [1, 100, 3] is 0.38490273396714775,
Average training RMSE with para [1, 100, 5] is 0.33926981604934714,
Average validation RMSE with para [1, 100, 5] is 0.36731310637114095,
Average training RMSE with para [1, 100, 7] is 0.33148901414945603,
Average validation RMSE with para [1, 100, 7] is 0.36272354595668654,
Average training RMSE with para [1, 150, 3] is 0.35075781154719576,
Average validation RMSE with para [1, 150, 3] is 0.3837397809144633,
Average training RMSE with para [1, 150, 5] is 0.3391677013423679,
Average validation RMSE with para [1, 150, 5] is 0.36746249898823125,
Average training RMSE with para [1, 150, 7] is 0.3313732581287591,
Average validation RMSE with para [1, 150, 7] is 0.3627899866034745,
Average training RMSE with para [1, 200, 3] is 0.350629655935507,
Average validation RMSE with para [1, 200, 3] is 0.3838824943425093,
Average training RMSE with para [1, 200, 5] is 0.3392682954200117,
Average validation RMSE with para [1, 200, 5] is 0.36735068270609095,
Average training RMSE with para [1, 200, 7] is 0.3314237338687723,
Average validation RMSE with para [1, 200, 7] is 0.3625958256309643,
Average training RMSE with para [2, 50, 3] is 0.34979545961689573,
Average validation RMSE with para [2, 50, 3] is 0.38578163429552903,
Average training RMSE with para [2, 50, 5] is 0.3376034889414761,
Average validation RMSE with para [2, 50, 5] is 0.36647434173503324,
Average training RMSE with para [2, 50, 7] is 0.3291610128386481,
Average validation RMSE with para [2, 50, 7] is 0.3620474226137125,
Average training RMSE with para [2, 100, 3] is 0.34971006699640406,
Average validation RMSE with para [2, 100, 3] is 0.3851995788940037,

Average training RMSE with para [2, 100, 5] is 0.3374804131733849,
Average validation RMSE with para [2, 100, 5] is 0.36609899513028676,
Average training RMSE with para [2, 100, 7] is 0.3291141361329872,
Average validation RMSE with para [2, 100, 7] is 0.3621903485237676,
Average training RMSE with para [2, 150, 3] is 0.34967960595975106,
Average validation RMSE with para [2, 150, 3] is 0.3852842920304221 4,
Average training RMSE with para [2, 150, 5] is 0.33753131329723407,
Average validation RMSE with para [2, 150, 5] is 0.36635574049633496,
Average training RMSE with para [2, 150, 7] is 0.32906274308132055,
Average validation RMSE with para [2, 150, 7] is 0.3618769392742204,
Average training RMSE with para [2, 200, 3] is 0.34972698166399707,
Average validation RMSE with para [2, 200, 3] is 0.38493886646776754,
Average training RMSE with para [2, 200, 5] is 0.33747376347684765,
Average validation RMSE with para [2, 200, 5] is 0.36624132820012123,
Average training RMSE with para [2, 200, 7] is 0.3290457917274534,
Average validation RMSE with para [2, 200, 7] is 0.3620079409159619,
Average training RMSE with para [3, 50, 3] is 0.3500407525398834,
Average validation RMSE with para [3, 50, 3] is 0.3859504581446822,
Average training RMSE with para [3, 50, 5] is 0.33686757615002794,
Average validation RMSE with para [3, 50, 5] is 0.3658944515724773,
Average training RMSE with para [3, 50, 7] is 0.32832635694928036,
Average validation RMSE with para [3, 50, 7] is 0.3619700421390539,
Average training RMSE with para [3, 100, 3] is 0.3499770052958662,
Average validation RMSE with para [3, 100, 3] is 0.3868561597963756,
Average training RMSE with para [3, 100, 5] is 0.3368570379082284,
Average validation RMSE with para [3, 100, 5] is 0.3663646005045185,
Average training RMSE with para [3, 100, 7] is 0.3281909944331029,
Average validation RMSE with para [3, 100, 7] is 0.3620401787028841,
Average training RMSE with para [3, 150, 3] is 0.35009867865243643,
Average validation RMSE with para [3, 150, 3] is 0.3871915095923903,
Average training RMSE with para [3, 150, 5] is 0.3368391652499005,
Average validation RMSE with para [3, 150, 5] is 0.366200443588783,
Average training RMSE with para [3, 150, 7] is 0.3281119051067607,
Average validation RMSE with para [3, 150, 7] is 0.36194256125148494,
Average training RMSE with para [3, 200, 3] is 0.3500304934580761,
Average validation RMSE with para [3, 200, 3] is 0.3863559711299699,
Average training RMSE with para [3, 200, 5] is 0.3367736743402251,
Average validation RMSE with para [3, 200, 5] is 0.366136718000058,
Average training RMSE with para [3, 200, 7] is 0.32814796986922923,
Average validation RMSE with para [3, 200, 7] is 0.36182404953592273,
Average training RMSE with para [4, 50, 3] is 0.35212721730128016,
Average validation RMSE with para [4, 50, 3] is 0.3910025092011896,
Average training RMSE with para [4, 50, 5] is 0.3369180047133965,
Average validation RMSE with para [4, 50, 5] is 0.3666162972092498,
Average training RMSE with para [4, 50, 7] is 0.32797892783924193,
Average validation RMSE with para [4, 50, 7] is 0.36247296177938504,

Average training RMSE with para [4, 100, 3] is 0.3521728919120621,
Average validation RMSE with para [4, 100, 3] is 0.3900944350606682,
Average training RMSE with para [4, 100, 5] is 0.3368419828845303,
Average validation RMSE with para [4, 100, 5] is 0.3665170243297219,
Average training RMSE with para [4, 100, 7] is 0.32789041251358864,
Average validation RMSE with para [4, 100, 7] is 0.3620717085887844,
Average training RMSE with para [4, 150, 3] is 0.3519896138712947,
Average validation RMSE with para [4, 150, 3] is 0.3898617012409924,
Average training RMSE with para [4, 150, 5] is 0.3368208232671729,
Average validation RMSE with para [4, 150, 5] is 0.36668040329532736,
Average training RMSE with para [4, 150, 7] is 0.3278595695591651,
Average validation RMSE with para [4, 150, 7] is 0.36213778289373527,
Average training RMSE with para [4, 200, 3] is 0.3521602563148646,
Average validation RMSE with para [4, 200, 3] is 0.3901347948073941,
Average training RMSE with para [4, 200, 5] is 0.33682094040007476,
Average validation RMSE with para [4, 200, 5] is 0.36675062142193643,
Average training RMSE with para [4, 200, 7] is 0.327896426540982,
Average validation RMSE with para [4, 200, 7] is 0.36204637081268587,
Neural network, the smallest rmse with para [3, 200, 7] is 0.36182404953592273

```
print(best_max_features)
print(              best_n_estimators_)
print(               best_max_depth)
print(best_rmse)
```

```
3
200
7
0.36182404953592273
```

Q7.1
Maximum number of features: It determines how many features will be used in a single tree in the model. If we can decrease the number of features that can be seen for each tree to a certain point, the complexity of each tree will decrease, which can result in a regularization effect.

Number of trees:We can increase the number of trees to result in a regularization effect. When we average the result of increasing number of trees, the prediction results will be more generalizaed and reduce the variance.

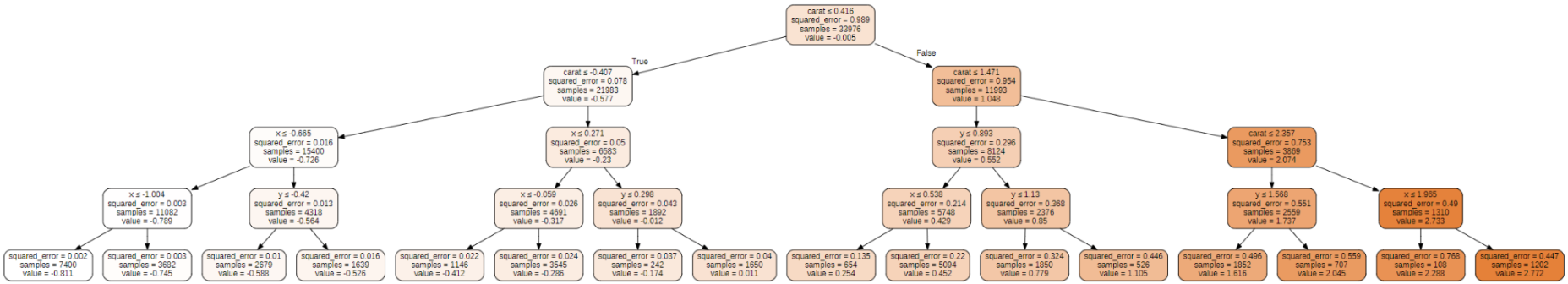
Depth of each tree: We can limit the depth of a tree to result in a regularization effect. Because if the tree is too deep, the complexity of single tree will increase. It will fit the training data better, which will be less generalized.

Q 7.2

Because random forest combines all the prediction results from different trees, and each tress determine different parts of the decision boundary.

Q7.3
We pick randomforest model with max feature 2, n_estimator 100 and max depth 4:

.


carat ≤ 0.416
squared_error = 0.989
samples = 33976
value = -0.005
True
False
carat ≤ -0.407
squared_error = 0.078
samples = 21983
value = -0.577
carat ≤ 1.471
squared_error = 0.954
samples = 11993
value = 1.048
x ≤ -0.665
squared_error = 0.016
samples = 15400
value = -0.726
x ≤ 0.271
squared_error = 0.05
samples = 6583
value = -0.23
y ≤ 0.893
squared_error = 0.296
samples = 8124
value = 0.552
carat ≤ 2.357
squared_error = 0.753
samples = 3869
value = 2.074
x ≤ -1.004
squared_error = 0.003
samples = 11082
value = -0.789
y ≤ -0.42
squared_error = 0.013
samples = 4318
value = -0.564
x ≤ -0.059
squared_error = 0.026
samples = 4691
value = -0.317
y ≤ 0.298
squared_error = 0.043
samples = 1892
value = -0.012
x ≤ 0.538
squared_error = 0.214
samples = 5748
value = 0.429
y ≤ 1.13
squared_error = 0.368
samples = 2376
value = 0.85
y ≤ 1.568
squared_error = 0.551
samples = 2559
value = 1.737
x ≤ 1.965
squared_error = 0.49
samples = 1310
value = 2.733
squared_error = 0.002
samples = 7400
value = -0.811
squared_error = 0.003
samples = 3682
value = -0.745
squared_error = 0.01
samples = 2679
value = -0.588
squared_error = 0.016
samples = 1639
value = -0.526
squared_error = 0.022
samples = 1146
value = -0.412
squared_error = 0.024
samples = 3545
value = -0.286
squared_error = 0.037
samples = 242
value = -0.174
squared_error = 0.04
samples = 1650
value = 0.011
squared_error = 0.135
samples = 654
value = 0.254
squared_error = 0.22
samples = 5094
value = 0.452
squared_error = 0.324
samples = 1850
value = 0.779
squared_error = 0.446
samples = 526
value = 1.105
squared_error = 0.496
samples = 1852
value = 1.616
squared_error = 0.559
samples = 707
value = 2.045
squared_error = 0.768
samples = 108
value = 2.288
squared_error = 0.447
samples = 1202
value = 2.772

You can zoom to see the detail.
The feature at root node is carat. Meaning this feature will first filter all the data that not satisfy this requirement. It's very important.

Q 7.4:

```
print('OOB of best:',max(oob_score))
```

```
OOB of best: 0.8830539693816566
```

The best oob score among all the combination we shown above is0.8830539693816566, which has an error of 0.11694603061834341.

From the link website, for OOB, it bootstrap the train dataset on the random forest tree, some data might not work for the tree, and the RMSE for that data is out of bag error.

For R square score, it is defined by the ratio of target's variance by the characteristic in the dataset. It access all the data points in the training dataset. In other words, R square score shows how well the data fit the regression model.

Q8 (Catboost and Bayesian Optimization)

```
parameters = {'n_estimators': [50,100,150,200],
              'num_leaves': [100,250,500,750],
              'max_depth': [4,6,8,10],
              'learning_rate': [0.003,0.03]
             }

grid = BayesSearchCV(CatBoostRegressor(grow_policy='Lossguide', verbose=False), parameters, cv=10, scoring='neg_root_mean_squared_error')
grid.fit(X_feature, y)

print('best model {}'.format(grid.best_estimator_.get_params()))
print('best rmse: {}'.format(-grid.best_score_))
```

```
best model {'learning_rate': 0.025277741835475726, 'loss_function': 'RMSE', 'verbose': False, 'max_depth': 10, 'n_estimators': 150, 'grow_policy': 'Lossguide', 'num_leaves': 750}
best rmse: 0.3588138433124769
```

Q 8.1:
We choose n_estimators, num_leaves,max_depth and learning_rate as out parameters for tuning. And the search space is provided in the above image.

Q 8.2:
From given image, we can tell the best parameters is learning_rate': 0.025277741835475726, 'max_depth': 10, 'n_estimators': 150, 'num_leaves': 750. The best RMSE is 0.3588138433124769.

Q 8.3:

Learning rate : By setting a lower learning rate, it helps with the performance and regularization. Because the latest batch will not be important, which generalize the model. However, it requires more time to converge, which will decrease the fitting efficiency.

Max_depth:By increasing the depth, it helps with the performance on training data as it allows more complexity in single tree. However, deeper trees will lead to a long training time which decrease the fitting efficiency. A smaller depth of tree will not fit the training so well that prevent overfitting and help with regularization.

N_estimators: Increasing the number of trees in the ensemble aprt will lead to a generalized result which will improve the preformance and also help with the generalization. More trees will also increase the training time, wichi decrease the fitting efficiency.

Num_leaves : A larger number of leaves can obtain more features in the model which improve the performance. Limiting number of leaves will help with regularization as it introduce noise by not capturing all the features.

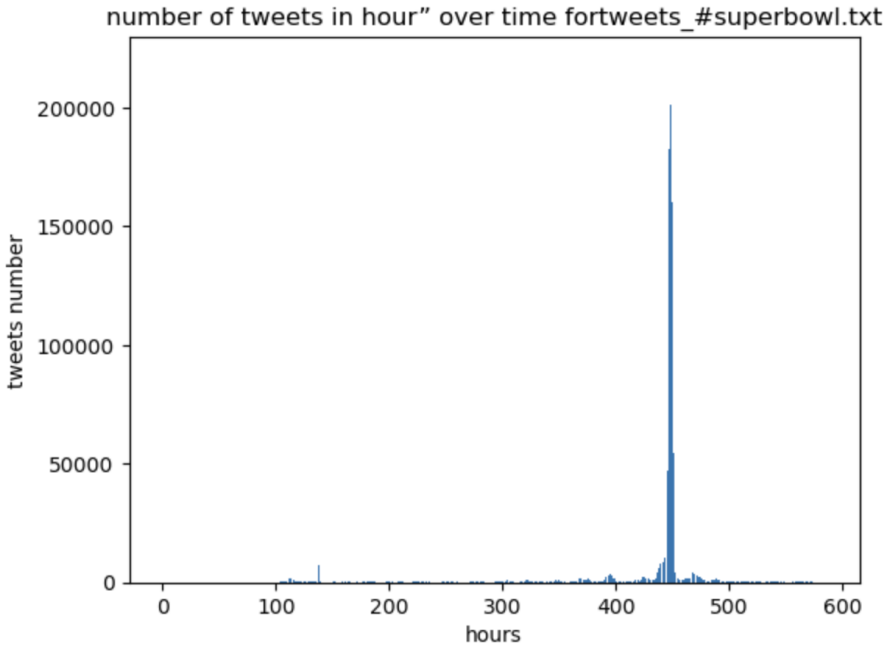
Twitter Data

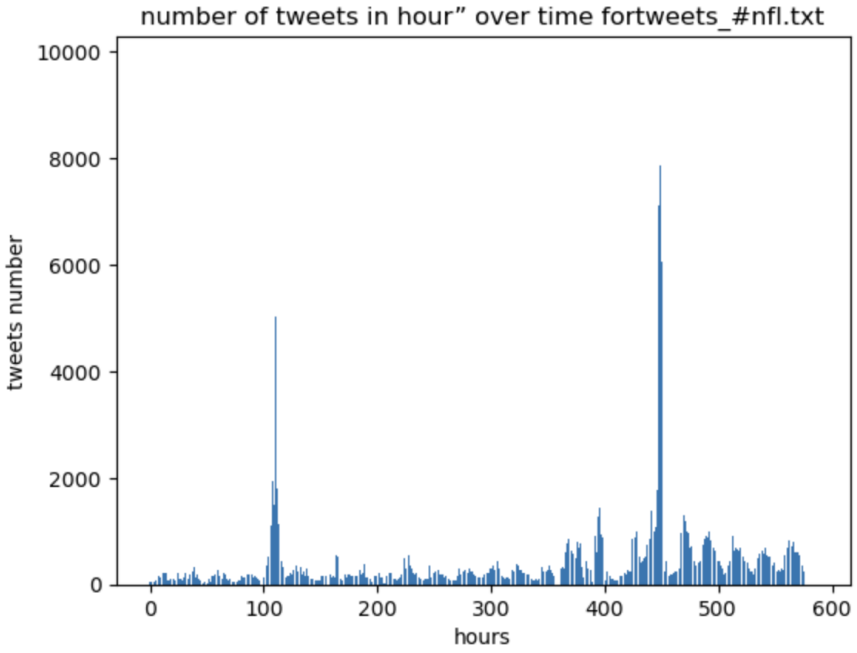
Question 9.1:

```
tweets_#superbowl.txt
Average tweets per hour: 2072.11840170408
Average retweets per tweet: 2.3911895819207736
Average followers of users posting the tweets per tweet: 8814.96799424623
tweets_#nfl.txt
Average tweets per hour: 397.0213901819841
Average retweets per tweet: 1.5344602655543254
Average followers of users posting the tweets per tweet: 4662.37544523693
tweets_#gopatriots.txt
Average tweets per hour: 40.95469800606194
Average retweets per tweet: 1.4081919101697078
Average followers of users posting the tweets per tweet: 1427.2526051635405
tweets_#patriots.txt
Average tweets per hour: 750.89426460689
Average retweets per tweet: 1.7852871288476946
Average followers of users posting the tweets per tweet: 3280.4635616550277
tweets_#gohawks.txt
Average tweets per hour: 292.48785062173687
Average retweets per tweet: 2.0132093991319877
Average followers of users posting the tweets per tweet: 2217.9237355281984
tweets_#sb49.txt
Average tweets per hour: 1276.8570598680474
Average retweets per tweet: 2.52713444111402
Average followers of users posting the tweets per tweet: 10374.160292019487
```

All the statistics for each hashtags such as average tweets per hour, average reweets per tweet and average followers of users posting the tweets per tweet are shown above.

Question 9.2:





The above two plots are the number of tweets in hour over time for tweets of two different hashtags “superbowl”, and “nfl”

Question 10:

Prediction 1: The number of retweets

For this part, we wll process and analyze all the 6 files, .json type file.. To predict the retweets number in the next hour, we choose the features: retweets, time, tweets, followers, ranking score, days per tweet, and mentions. The model we choose is linear regression model with ordinary least square. We choose this model because the model fits the relationship between explanatory variables that minimizes the sum of square errors, and we will report MSE for each file.

Below are the model summary for the files:

```
tweets_#superbowl.txt
MSE: 774178344.9187443
                                 OLS Regression Results
================================================================================
Dep. Variable:                      y   R-squared (uncentered):                   0.920
Model:                            OLS   Adj. R-squared (uncentered):              0.919
Method:                 Least Squares   F-statistic:                              923.2
Date:                Thu, 16 Mar 2023   Prob (F-statistic):                   4.03e-304
Time:                        22:29:43   Log-Likelihood:                         -5756.8
No. Observations:                 571   AIC:                                  1.153e+04
Df Residuals:                     564   BIC:                                  1.156e+04
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1            -0.4596      0.134     -3.424      0.001      -0.723      -0.196
x2           -26.2030     18.834     -1.391      0.165     -63.196      10.790
x3           -40.2836      5.897     -6.832      0.000     -51.866     -28.701
x4             0.0003   2.66e-05     12.607      0.000       0.000       0.000
x5             8.2292      1.262      6.521      0.000       5.750      10.708
x6            -0.0570      0.099     -0.574      0.566      -0.252       0.138
x7             4.0589      0.386     10.513      0.000       3.301       4.817
==============================================================================
Omnibus:                      268.387   Durbin-Watson:                   1.567
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           177537.372
Skew:                           0.522   Prob(JB):                         0.00
Kurtosis:                      89.378   Cond. No.                     6.08e+06
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 6.08e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
**************************************
```

tweets_#nfl.txt
MSE: 340253077.3547606

OLS Regression Results

Dep. Variable:	y	R-squared (uncentered):	0.920
Model:	OLS	Adj. R-squared (uncentered):	0.919
Method:	Least Squares	F-statistic:	923.2
Date:	Thu, 16 Mar 2023	Prob (F-statistic):	4.03e-304
Time:	22:29:51	Log-Likelihood:	-5756.8
No. Observations:	571	AIC:	1.153e+04
Df Residuals:	564	BIC:	1.156e+04
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
x1	-0.4596	0.134	-3.424	0.001	-0.723	-0.196
x2	-26.2030	18.834	-1.391	0.165	-63.196	10.790
x3	-40.2836	5.897	-6.832	0.000	-51.866	-28.701
x4	0.0003	2.66e-05	12.607	0.000	0.000	0.000
x5	8.2292	1.262	6.521	0.000	5.750	10.708
x6	-0.0570	0.099	-0.574	0.566	-0.252	0.138
x7	4.0589	0.386	10.513	0.000	3.301	4.817

Omnibus:	268.387	Durbin-Watson:	1.567
Prob(Omnibus):	0.000	Jarque-Bera (JB):	177537.372
Skew:	0.522	Prob(JB):	0.00
Kurtosis:	89.378	Cond. No.	6.08e+06

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 6.08e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
****

tweets_#gopatriots.txt
MSE: 371562792.5029074

OLS Regression Results

Dep. Variable:	y	R-squared (uncentered):	0.920
Model:	OLS	Adj. R-squared (uncentered):	0.919
Method:	Least Squares	F-statistic:	923.2
Date:	Thu, 16 Mar 2023	Prob (F-statistic):	4.03e-304
Time:	22:29:52	Log-Likelihood:	-5756.8
No. Observations:	571	AIC:	1.153e+04
Df Residuals:	564	BIC:	1.156e+04
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
x1	-0.4596	0.134	-3.424	0.001	-0.723	-0.196
x2	-26.2030	18.834	-1.391	0.165	-63.196	10.790
x3	-40.2836	5.897	-6.832	0.000	-51.866	-28.701
x4	0.0003	2.66e-05	12.607	0.000	0.000	0.000
x5	8.2292	1.262	6.521	0.000	5.750	10.708
x6	-0.0570	0.099	-0.574	0.566	-0.252	0.138
x7	4.0589	0.386	10.513	0.000	3.301	4.817

Omnibus:	268.387	Durbin-Watson:	1.567
Prob(Omnibus):	0.000	Jarque-Bera (JB):	177537.372
Skew:	0.522	Prob(JB):	0.00
Kurtosis:	89.378	Cond. No.	6.08e+06

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 6.08e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
****

tweets_#patriots.txt
MSE: 238946312.50980747

OLS Regression Results

Dep. Variable:	y	R-squared (uncentered):	0.920
Model:	OLS	Adj. R-squared (uncentered):	0.919
Method:	Least Squares	F-statistic:	923.2
Date:	Thu, 16 Mar 2023	Prob (F-statistic):	4.03e-304
Time:	22:30:05	Log-Likelihood:	-5756.8
No. Observations:	571	AIC:	1.153e+04
Df Residuals:	564	BIC:	1.156e+04
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
x1	-0.4596	0.134	-3.424	0.001	-0.723	-0.196
x2	-26.2030	18.834	-1.391	0.165	-63.196	10.790
x3	-40.2836	5.897	-6.832	0.000	-51.866	-28.701
x4	0.0003	2.66e-05	12.607	0.000	0.000	0.000
x5	8.2292	1.262	6.521	0.000	5.750	10.708
x6	-0.0570	0.099	-0.574	0.566	-0.252	0.138
x7	4.0589	0.386	10.513	0.000	3.301	4.817

Omnibus:	268.387	Durbin-Watson:	1.567
Prob(Omnibus):	0.000	Jarque-Bera (JB):	177537.372
Skew:	0.522	Prob(JB):	0.00
Kurtosis:	89.378	Cond. No.	6.08e+06

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 6.08e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
****

tweets_#gohawks.txt
MSE: 340149063.52530336

OLS Regression Results

Dep. Variable:	y	R-squared (uncentered):	0.920
Model:	OLS	Adj. R-squared (uncentered):	0.919
Method:	Least Squares	F-statistic:	923.2
Date:	Thu, 16 Mar 2023	Prob (F-statistic):	4.03e-304
Time:	22:30:10	Log-Likelihood:	-5756.8
No. Observations:	571	AIC:	1.153e+04
Df Residuals:	564	BIC:	1.156e+04
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
x1	-0.4596	0.134	-3.424	0.001	-0.723	-0.196
x2	-26.2030	18.834	-1.391	0.165	-63.196	10.790
x3	-40.2836	5.897	-6.832	0.000	-51.866	-28.701
x4	0.0003	2.66e-05	12.607	0.000	0.000	0.000
x5	8.2292	1.262	6.521	0.000	5.750	10.708
x6	-0.0570	0.099	-0.574	0.566	-0.252	0.138
x7	4.0589	0.386	10.513	0.000	3.301	4.817

Omnibus:	268.387	Durbin-Watson:	1.567
Prob(Omnibus):	0.000	Jarque-Bera (JB):	177537.372
Skew:	0.522	Prob(JB):	0.00
Kurtosis:	89.378	Cond. No.	6.08e+06

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 6.08e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

```
tweets_#sb49.txt
MSE: 33471784.49172828
                                 OLS Regression Results
================================================================================
Dep. Variable:                          y   R-squared (uncentered):                   0.920
Model:                                OLS   Adj. R-squared (uncentered):              0.919
Method:                     Least Squares   F-statistic:                              923.2
Date:                    Thu, 16 Mar 2023   Prob (F-statistic):                   4.03e-304
Time:                            22:30:33   Log-Likelihood:                         -5756.8
No. Observations:                     571   AIC:                                  1.153e+04
Df Residuals:                         564   BIC:                                  1.156e+04
Df Model:                               7
Covariance Type:                nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1            -0.4596      0.134     -3.424      0.001      -0.723      -0.196
x2           -26.2030     18.834     -1.391      0.165     -63.196      10.790
x3           -40.2836      5.897     -6.832      0.000     -51.866     -28.701
x4             0.0003   2.66e-05     12.607      0.000       0.000       0.000
x5             8.2292      1.262      6.521      0.000       5.750      10.708
x6            -0.0570      0.099     -0.574      0.566      -0.252       0.138
x7             4.0589      0.386     10.513      0.000       3.301       4.817
==============================================================================
Omnibus:                      268.387   Durbin-Watson:                   1.567
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           177537.372
Skew:                           0.522   Prob(JB):                         0.00
Kurtosis:                      89.378   Cond. No.                     6.08e+06
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 6.08e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
**************************************
```

x1,x2…x7 are the features corresponding to retweets, time, tweets, followers, ranking score, days per tweet, and mentions.
From the summary chart above, we can see the P value of x6 and x2 which are features days per tweet and time are the key components for the prediction of reweets numbers which are make sense because the more you post the same hashtag which make this topic hot during a specific time period, and it will get more retweets.

Prediction 2: Fan location

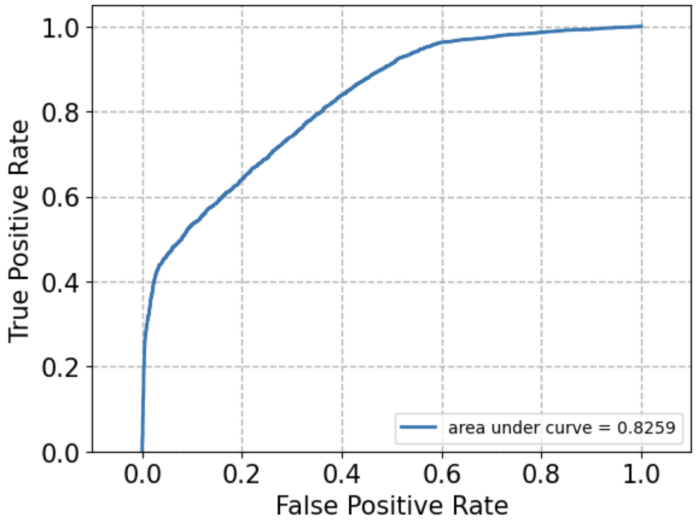
For the fan location prediction, we will only look at the #superbowl file, since the other files such as #gohawks is probably a fan based hashtag which is biased for the location prediction because people in the same region tend have a same team to support. We define the binary location base: tag WA, contains ip address Wshington, seattle, WA; tag MA, contains ip address Massachusetts, Boston, MA.

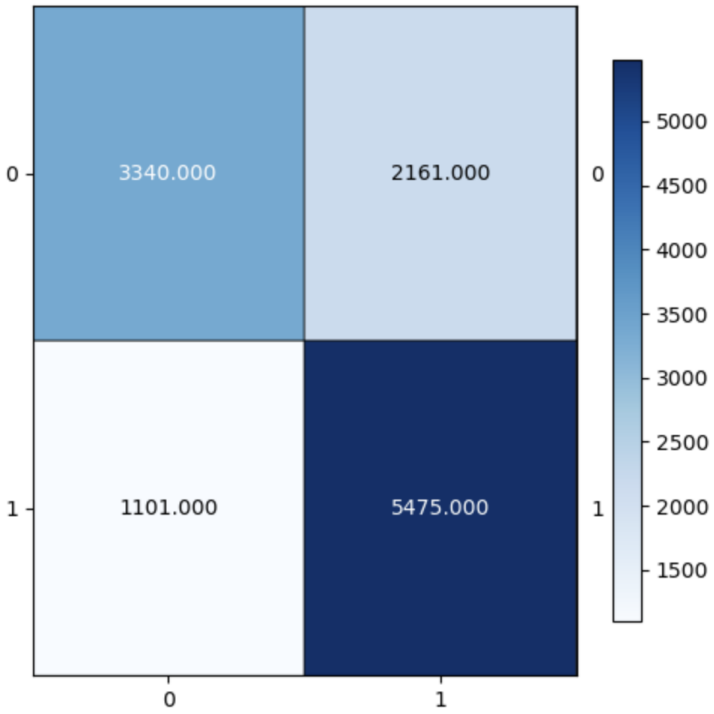
Since we want to find the key word in the text, we first do the text clean, removing the punctuation, space, etc. We also lemmatize the text after the text clean. Now we get the train dataset, the model we choose for this binary problem is truncated SVD. We get the feature extraction by construct the TF-IDF matrix. We train the model by 3 ways, no regularization, L1

regularization and L2 regularization. We also report the ROC curve to show the performance of classification over the threshold.

With no regularizaition:

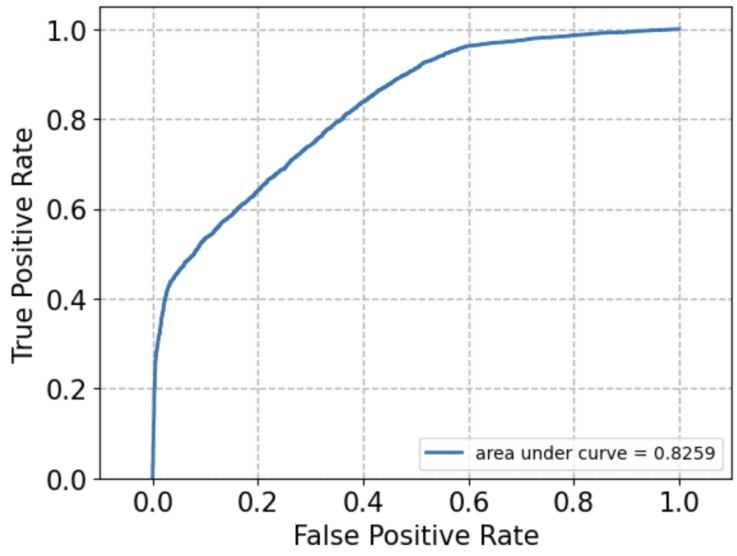
```
Precision of model is 0.7345406463581827
Recall of model  is 0.7198676634108994
F1-Score of model is 0.7211863284958755
Accuracy of model is 0.7298998095553532
```

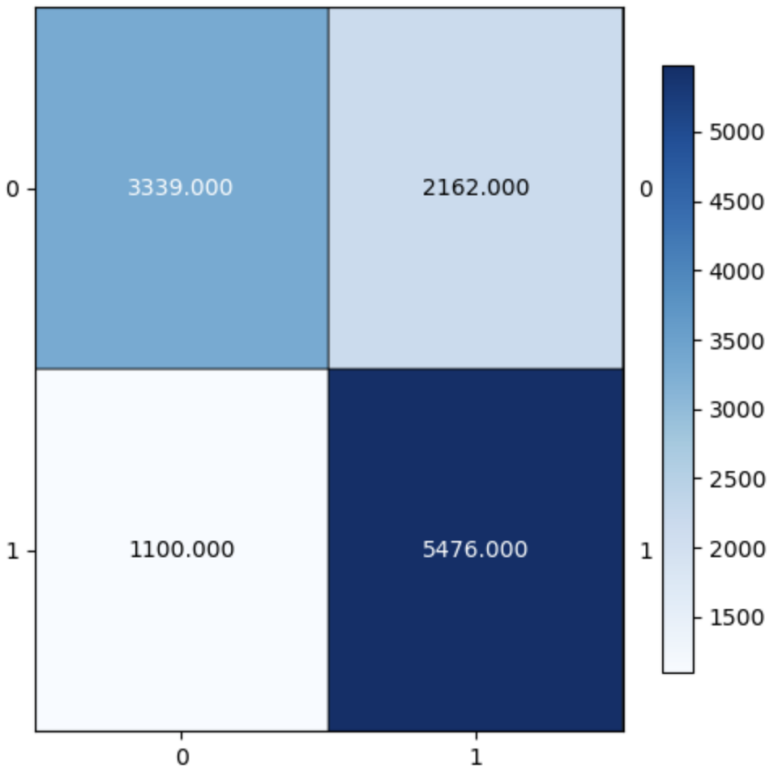




With L1 regularization:

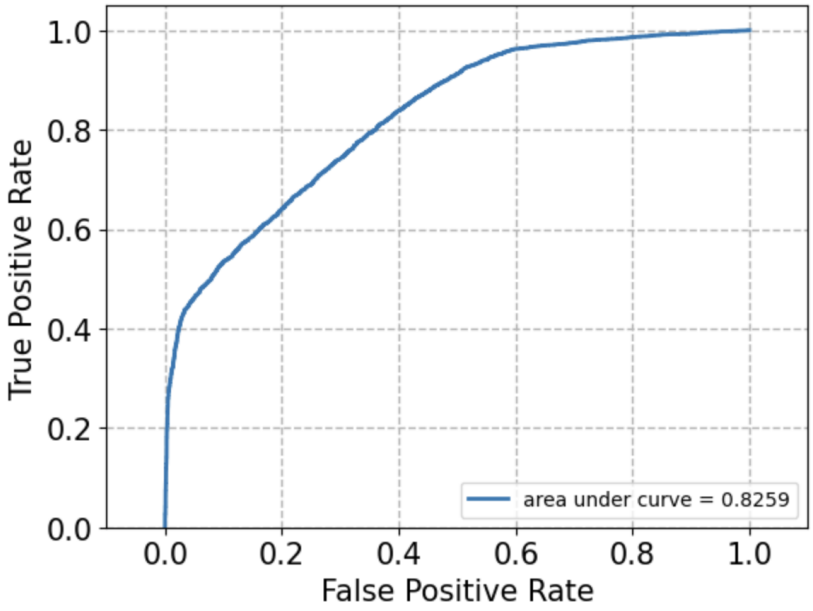
```
Precision of model is 0.7345690241952519
Recall of model  is 0.7198528049091716
F1-Score of model is 0.7211694679120169
Accuracy of model is 0.7298998095553532
```

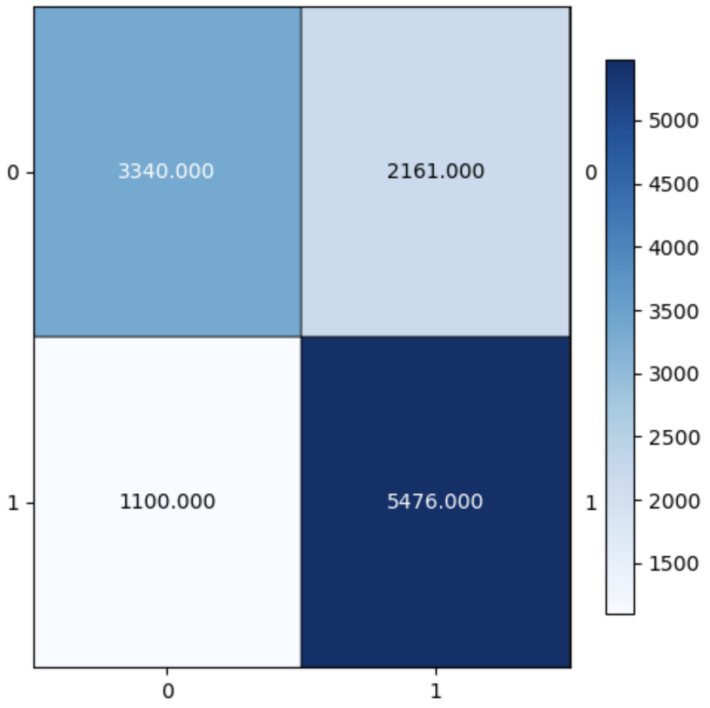




With L2 regularization:

```
Precision of model is 0.7346438686951977
Recall of model  is 0.7199436974741598
F1-Score of model is 0.7212633762444749
Accuracy of model is 0.7299826115757224
```





Prediction 3: Hashtag prediction

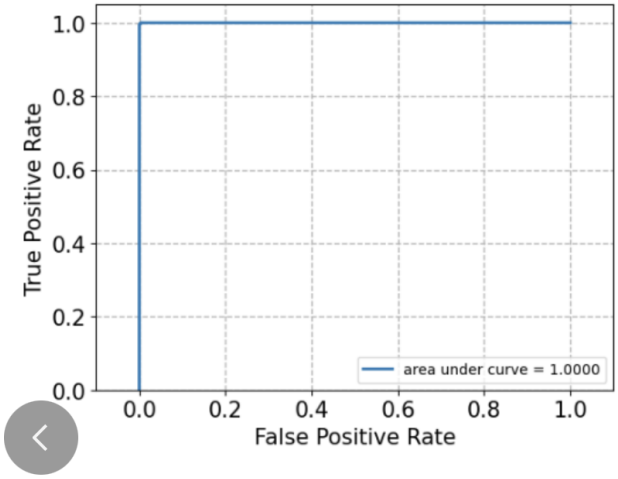
For hashtag prediction, we use the similar process as what we did for fan location prediction. We focus on two files: gopatriots and gohawks.

We choose the 20% of the dataset as testing dataset. We also clean the data first as what we did for fan location prediction and transfer it to the count vector, then we get TF-IDF for train and test dataset. We use truncated SVD for the problem. We get the feature extraction by construct the TF-IDF matrix. We train the model by 3 ways, no regularization, L1 regularization and L2 regularization. We also report the ROC curve to show the performance of classification over the threshold.

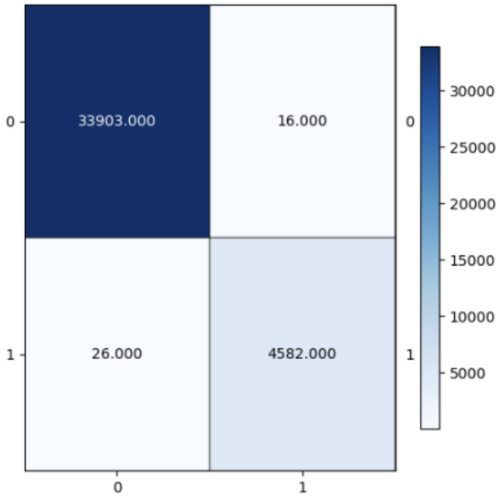
Besides that, we also train linear SVM to see how well the predictions are. We did both hard margin and soft margin, and we compare their performance.

Hard margin SVM:

```
Precision of model is 0.9978769600377402
Recall of model  is 0.9969429634345385
F1-Score of model is 0.9974093636046255
Accuracy of model is 0.9989098554260648
```

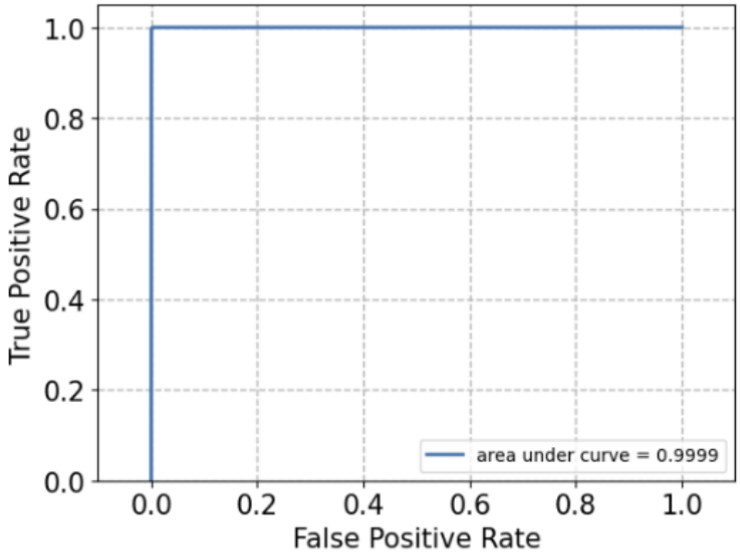


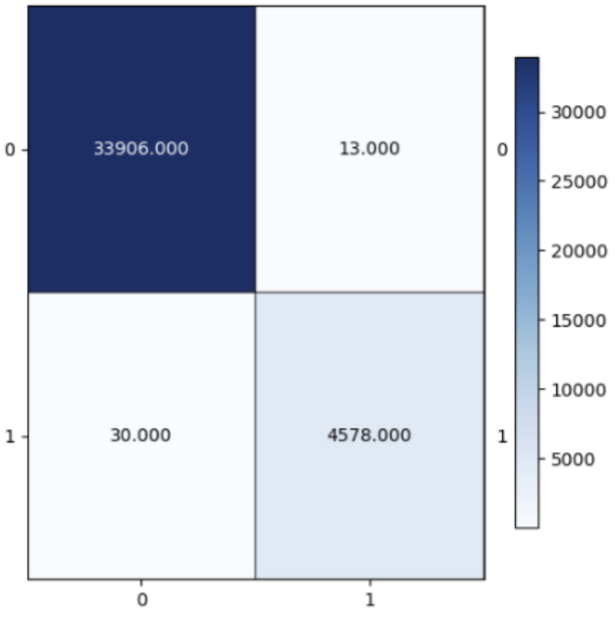

```
plot_mat(contingency_matrix(test_y,hard_Margin_prediction), size=(5,5))
```



Soft margin SVM:

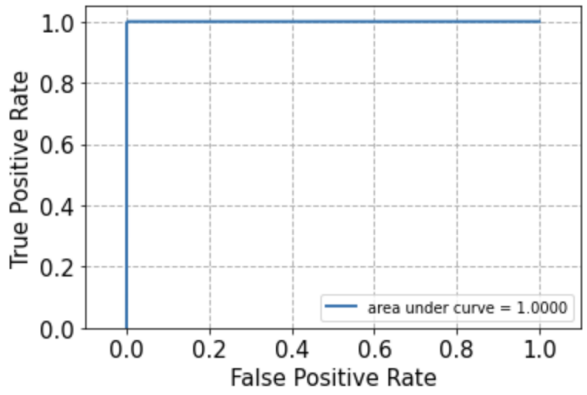
```
Precision of model is 0.9981421779651904
Recall of model  is 0.996553158658618
F1-Score of model is 0.9973459373225148
```

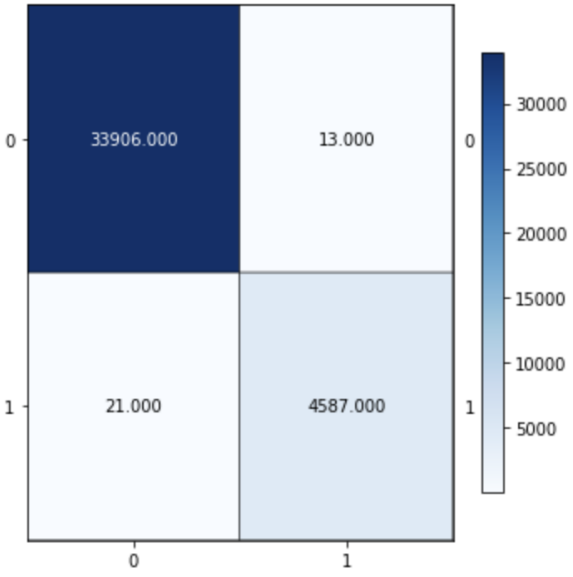




With no regularization:

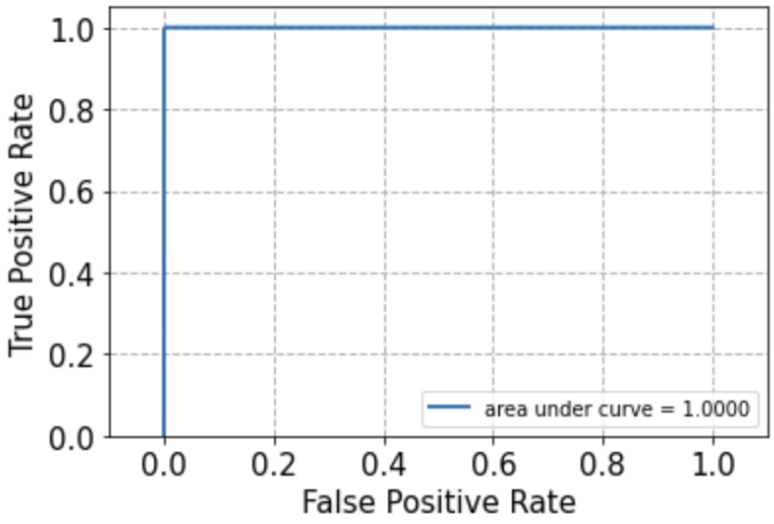
```
Precision of model is 0.9982774685033466
Recall of model  is 0.997529721158618
F1-Score of model is 0.9979032118607242
Accuracy of model is 0.9991175020115763
confusion_matrix is:
[[ 4587    21]
 [   13 33906]]
```

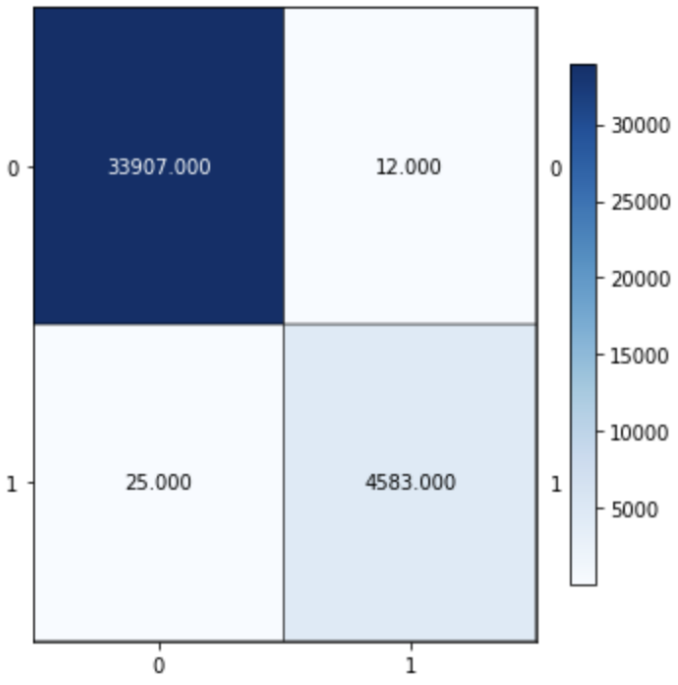




With L1 regularization:

```
Precision of model is 0.9983258490353298
Recall of model  is 0.9971104343814594
F1-Score of model is 0.9977171296778534
Accuracy of model is 0.9990396345420095
confusion_matrix is:
[[ 4583    25]
 [   12 33907]]
```





With L2 regularization:

```
Precision of model is 0.9983258490353298
Recall of model  is 0.9971104343814594
F1-Score of model is 0.9977171296778534
Accuracy of model is 0.9990396345420095
confusion_matrix is:
[[ 4583    25]
 [   12 33907]]
```