Credit Fraud Prediction

EE 660 Course Project

Project Type: (1) Design a system based on real-world data

Number of student authors: 1

Keyan Chen keyanche@usc.edu

November 30th, 2020

1. Abstract

In my project, I need to solve the credit card fraud problem. The problem data consist of European September 2013 credit transactions which are imbalanced. It contains 28 PCA transformed features, time, and amount. I plan to use K-neighbors classifier and decision tree classifier as baseline and use logistic regression classifier to make comparisons. The under and over sampling technique will be applied when compare these results. For baselines, I will compare all three methods in under sampling. For logistic regression, I will compare it performance with under sampling and over sampling technique. Eventually, the logistic regression classifier with under sampling method have best performance after comparisons.

2. Introduction

2.1. Problem Type, Statement and Goals

In my project, I used the dataset about the credit card fraud data. I plan to find a good method of prediction from several models. When finding the best prediction method, I will use under sampling to deal with our imbalanced dataset and compare these systems. Moreover, for my planed prediction system, I will also use over sampling technique to see the performance and compare the estimator with under sampling estimator. In this classification problem. My systems are trying to predict if the credit card transactions are fraud or non-fraud. I think the problem is classic and practical in machine learning study. I also have following challenges need to solve.

Example sources of difficulty (nontriviality) include:

(1) A physical model that's inherently complicated and hard to abstract.
Since the features and datapoints of my dataset are huge. The dataset is hard to abstract. I will use different models to see how the model fit better and check the error estimation.
(2) High dimensionality of feature space.
When we use the data from real world, the number of factors which can influence the results will be large. Therefore, we need to see the data correlation and remove the outlier features.
(3) Significant amounts of preprocessing required.
Since the imbalanced dataset that I use has lots of features, I need to figure out how to implement under sampling and over sampling techniques.

2.2. Our Prior and Related Work - None

2.3. Overview of Our Approach

In my project, I will use the K Neighbors Classifier, decision tree algorithms and logistic regression to perform our data. For baseline systems, I will use K neighbor's algorithms as simple baseline and use decision tree model as trivial baseline. I will compare the confusion matrix, ROC-AOC score, f1 score, precision, and recall score. The significant part is under sampling and over sampling technique. I learned some knowledge from research and will try to implement them in a right way.

3. Implementation

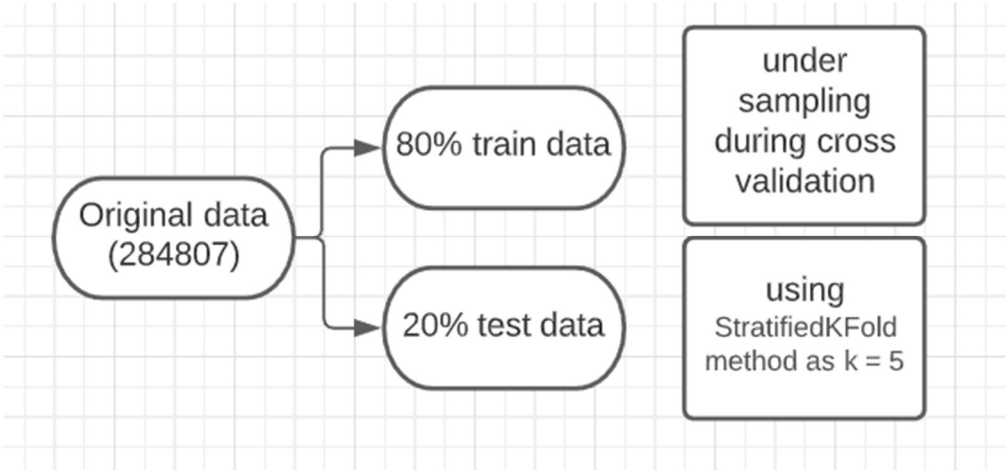
3.1. Data Set

In our dataset, we have only numerical input variables which are PCA transformation. There are 28 features obtained with PCA and 2 features which are time and amount. The time and amount features are not been transformed with PCA. Furthermore, feature class is the response variable which are represent by value 1 in case of fraud and value 0 in case of non-fraud. We have total 284807 datapoints and only 492 transactions are frauds. Apparently, our dataset is very imbalanced.

Input	
time	The time of the transaction
amount	The amount of money of the transaction
V1 to V28 (28 features)	The features that obtained by PCA
output	
Class	Non-fraud is 0 and Fraud is 1

3.2. Dataset Methodology

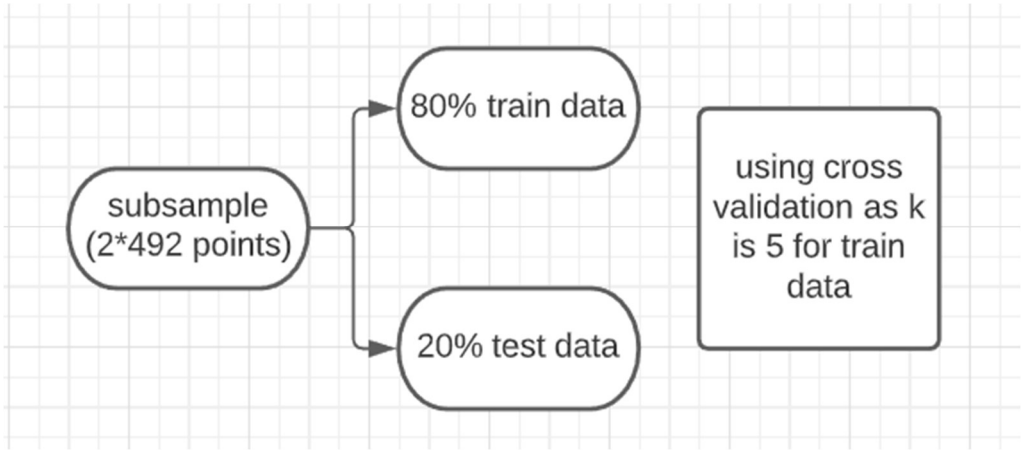
We have total 284807 datapoints in our dataset, and since our data set is imbalanced, I need to use under sampling and over sampling techniques to deal with the data. Before we use these techniques, I need to split the original data to train set and test set:



Here, after splitting data, we also need to check the test set and train set has same ratio of output class. For cross validation model, I used nested cross validation to see scores.

For under sampling:

We have 492 Fraud transactions here, if we use under sampling technique, we need to make a sub sample first, to make the sub sample's fraud and non- fraud numbers are same. The reason that create sub sample is that if we use original data, it will cause overfitting and get wrong corelations.

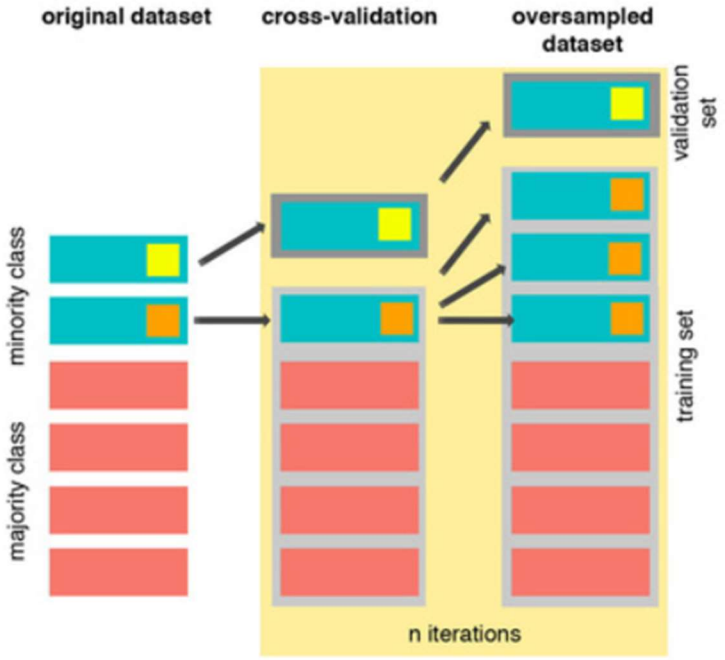


Moreover, the significant thing is that we need to apply the under-sampling technique during the cross validation.

Here, I will find the cross-validating score and compare the methods, after I trained from subsample, I will apply under sample technique to all best classifiers from each algorithm by using original data by making imbalanced-pipeline. For under sampling technique, I will use NearMiss function to train our classifier again. Then the classifier will be the one to predict the original train set.

For over sampling:

We still need to apply the technique during cross validation. As the following diagram shown.



(citation from [1])

For test data part:

The sub sample test data will used by under sampling comparison for all three models and the original test data will used by oversampling comparison and under sampling comparison for logistic regression since the decision tree model and K-Nearest-Neighbor model is baseline for us. I will deeply compare the logistic regression. The test set will used twice, one for under sample, and on for oversample.

3.3. Preprocessing, Feature Extraction, Dimensionality Adjustment

In our dataset, we do not have any missing value, but we have two features which are amount and time need to scale first. Here, I used the Robust-

Scaler from sklearn library. I used this scaler because this is less prone to outliers. Also, when reading the file, I will use pandas and numpy to deal with our dataset.

After using the scaler to deal with amount and time, we move the two columns to the first and second column in out data like the shown below:

	rs_amount	rs_time	V1	V2	V3	V4	V5	V6	V7	V8	...	V20	V21	V22	
0	1.783274	-0.994983	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	...	0.251412	-0.018307	0.277838	-0.110
1	-0.269825	-0.994983	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	...	-0.069083	-0.225775	-0.638672	0.101
2	4.983721	-0.994972	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	...	0.524980	0.247998	0.771679	0.909
3	1.418291	-0.994972	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	...	-0.208038	-0.108300	0.005274	-0.190
4	0.670579	-0.994960	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	...	0.408542	-0.009431	0.798278	-0.137
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
284802	-0.296653	1.034951	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	...	1.475829	0.213454	0.111864	1.014
284803	0.038986	1.034963	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	...	0.059616	0.214205	0.924384	0.012
284804	0.641096	1.034975	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	...	0.001396	0.232045	0.578229	-0.037
284805	-0.167680	1.034975	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	...	0.127434	0.265245	0.800049	-0.163
284806	2.724796	1.035022	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	...	0.382948	0.261057	0.643078	0.376

284807 rows × 31 columns

And then this will become our final original dataset after we scaled the two features. However, since we need to apply the under-sampling technique, I need to make a subsample to let the non-fraud transaction number equal to fraud transaction randomly. We need to check it:

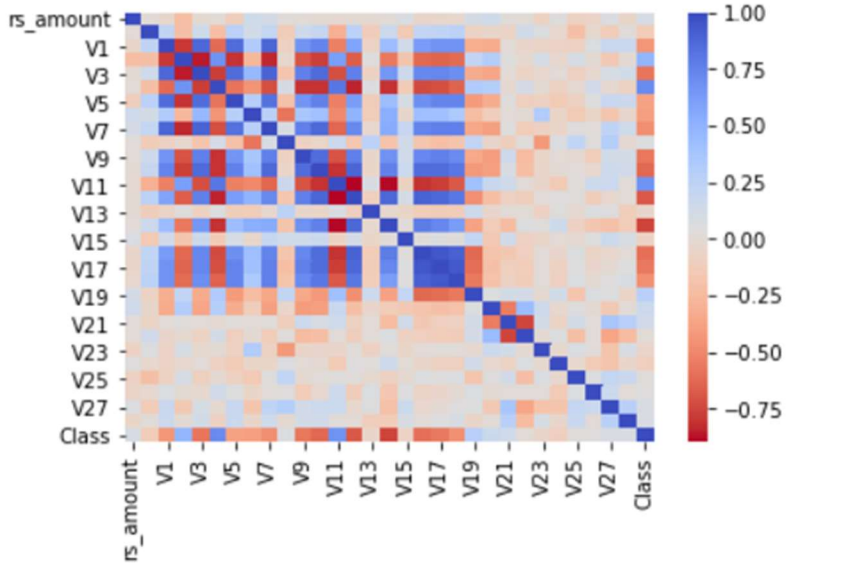
```
subsample distribution
1    0.5
0    0.5
Name: Class, dtype: float64
```

After getting the randomly 492 Non-fraud datapoints, we need to shuffle it by using normal distribution and getting like this:

	rs_amount	rs_time	V1	V2	V3	V4	V5	V6	V7	V8	...	V20	V21	V22	
9262	-0.098232	-0.838297	1.226878	0.063952	0.795418	0.109193	-0.491986	-0.345963	-0.409757	-0.090610	...	-0.003457	-0.297851	-0.635229	0.09
6427	-0.293440	-0.905579	0.725646	2.300894	-5.329976	4.007683	-1.730411	-1.732193	-3.968593	1.063728	...	0.504646	0.589669	0.109541	0.60
133014	0.139733	-0.052656	1.135322	0.142540	0.122711	0.474670	-0.003014	-0.078456	-0.068025	0.101293	...	-0.035654	-0.213125	-0.688959	0.07
249963	-0.296653	0.821967	-0.679521	4.672553	-6.814798	7.143500	0.928654	-1.873013	-2.306689	0.993702	...	0.872006	0.566849	-0.321691	-0.28
204079	1.208831	0.592230	1.862102	-0.124052	-1.989752	0.382609	0.473032	-0.674517	0.298621	-0.282416	...	0.150727	-0.204158	-0.511441	0.07
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
144754	4.216726	0.019784	-0.670238	0.945206	0.610051	2.640065	-2.707775	1.952611	-1.624608	-5.229908	...	1.474929	-2.504450	1.436472	0.35
247673	3.156012	0.810172	-5.192496	3.164721	-5.047679	2.246597	-4.011781	-0.638908	-2.873463	1.576318	...	-1.850470	1.167244	-1.006617	0.77
61367	-0.279746	-0.409908	1.252850	0.293765	-0.170780	0.243902	0.528910	0.413733	-0.052919	0.101599	...	-0.001047	-0.273833	-0.738917	-0.00
151730	-0.042479	0.134435	-1.952933	3.541385	-1.310561	5.955664	-1.003993	0.983049	-4.587235	-4.892184	...	1.965030	-1.998091	1.133706	-0.04
74794	4.051003	-0.339901	-6.003422	-3.930731	-0.007045	1.714669	3.414667	-2.329583	-1.901512	-2.746111	...	-4.128186	1.101671	-0.992494	-0.69

984 rows × 31 columns

Moreover, I want to see the heatmap of our data correlation, and I need to learn the features are positive correlation or negative correlation. We cannot use the original data here because it is extremely imbalanced, so that we will learn nothing. By using the seaborn library, we analyze the subsample and got this picture:



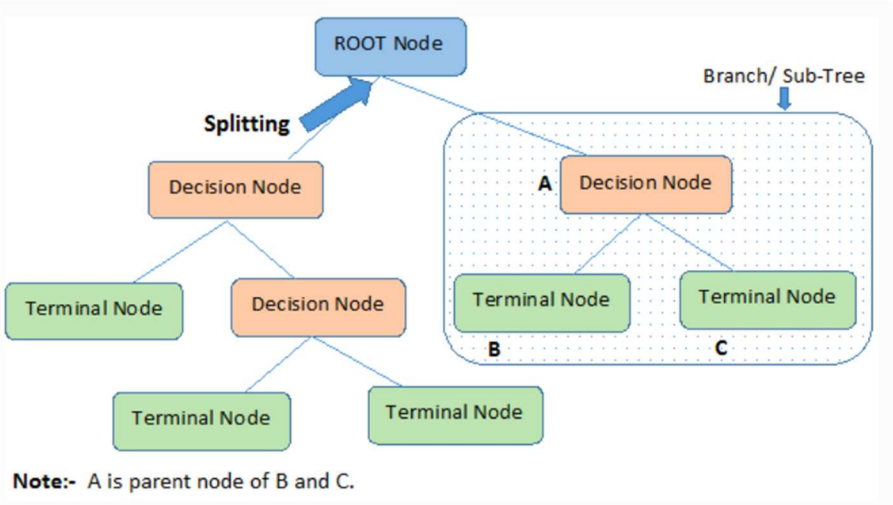
From the diagram, we can see that V17, V14, V12, V10 are negatively corelated. We can also see that V2, V4 V11 and V19 are positively correlated.

Furthermore, I will split our subsample to train data and test data as 4:1 ratio.

3.4. Training Process

Decision Tree Model:

This method is a simple method for baseline, so I choose this to become the nontrivial baseline. Moreover, this is the model I learned from EE660. The algorithm is like the shown below:



(citation from [2])

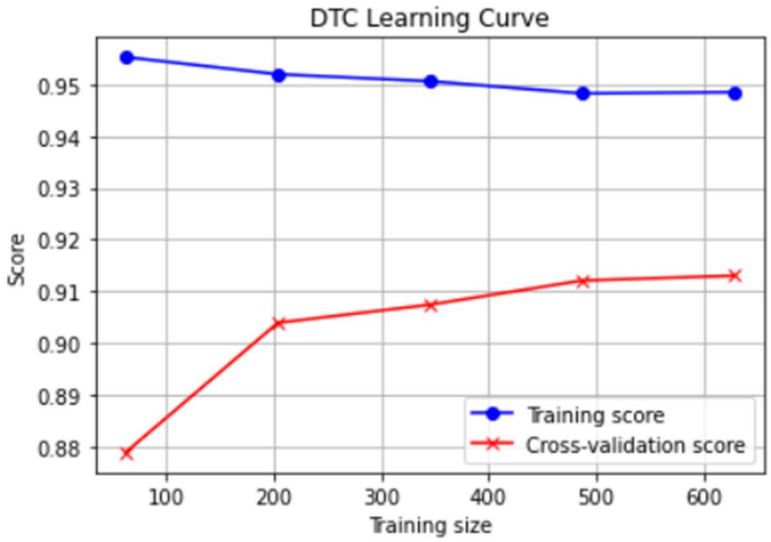
First for under sampling, I use the train set form subsample to train the data by using decision tree function. I will display the training score by using cross validation score function by choosing the k number is 5 and get the average number of the training score:

```
decision tree Classifier training score  89.0
```

After setting the tree parameter in a reasonable range (learned from internet), I will use gridsearchCV and fit to find the best parameter and best estimator. After these steps, I will use cross validation function again to see the cross-validation score:

```
decision tree Classifier cross validation score 91.87
```

During the cross validation, I will apply the NearMiss technique for under sampling and make pipeline to fit the decision tree classifier. Moreover, I plot the learning curve of the decision tree classifier:



the number of data point for the train sample is 787 in subsample and number of total input features are 30. I also found the best parameters after fit the subsample train data X and train data Y:

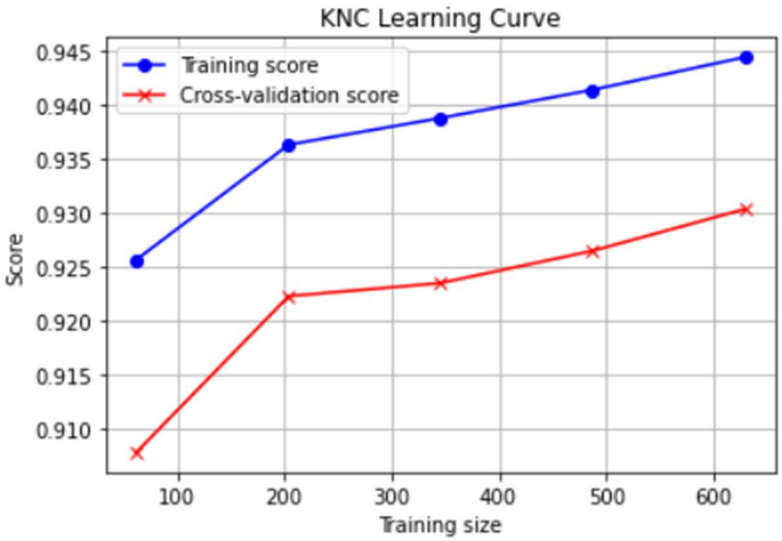
```
Best Parameters of DTC:
{'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 5}
```

K-Neighbor method:

This is also the method learned from EE660 class, and I choose this method to become another baseline to compare. In the algorithm, k is a constant, and a test point will be classified by assigning a label which is most frequent among k training points. By using this method, I got:

```
KNN Classifier training score  88.69
KNN Classifier cross validation score 92.88
```

I applied the same procedure with decision tree method to apply under sampling method. Therefore, I got the KNN learning curve:



Same as Decision tree method, the number of data point for the train sample is 787 in subsample and number of total input features are 30. From KNN classifier, I got the best parameter after fitting train data X and train data Y.

```
Best Parameters of KNN:
{'algorithm': 'auto', 'n_neighbors': 4}
```

Logistic Regression:

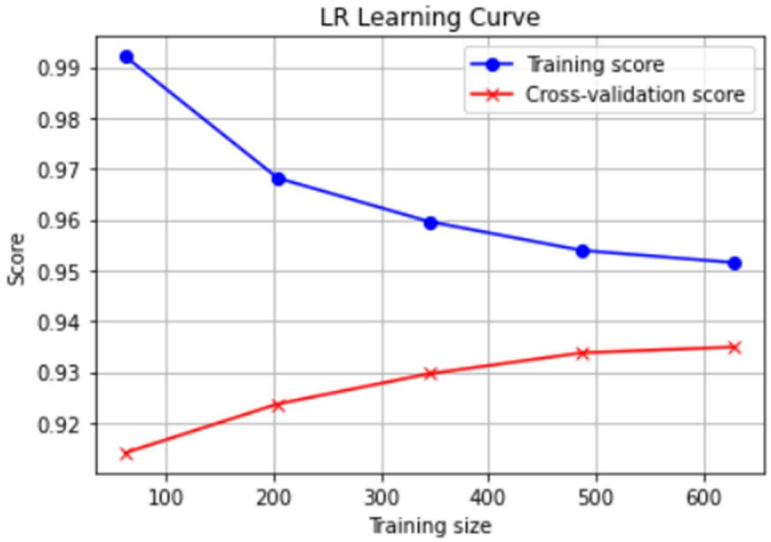
By learning from EE660, I believe logistic Regression is a good solution for our card fraud problem. In this method, I applied both under sampling method and over sampling method and see the results.

Under sampling:

I did the same procedure with previous methods, the train score and cross validation score are:

```
Logistic regression training score  88.69
Logistic regression cross validation score 93.77
```

Also, we can see the learning curve which can represent that how the training score and cross validation score change as train size increase:



Same as previous methods, the number of data point for the train sample is 787 in subsample and number of total input features are 30. And I found the best parameters are:

```
Best Parameters of under sample LR:
{'C': 1, 'penalty': 'l2'}
```

For over sampling:

I use SMOTE to over sampling our data. In this case, the number of data point for the train sample is 227846 and number of test set data is 56961, the parameters for oversampling

```
Best Parameters of over sample:
{'penalty': 'l2', 'C': 0.001}
```

Here, I have gotten the final hg from each H set with best parameters.

3.5. Model Selection and Comparison of Results

In this part, I will show the other results that I got from each dataset and each method:

By looking MSE first:

method		MSE
K-Neighbor	Subsample test data	0.0761
Decision Tree	Subsample test data	0.2944
Logistic Regression	Subsample test data	0.2589
	Original test data (under sample)	0.0247
	Original test data (over sample)	0.0246

From MSE, we can see that the result will be largely influenced by imbalanced dataset. In our case this is not reliable to see performances of each method.

Therefore, I will check advanced scores for subsample:

Precision: True Positives/(True Positives + False Positives)

Recall: True Positives/(True Positives + False Negatives)

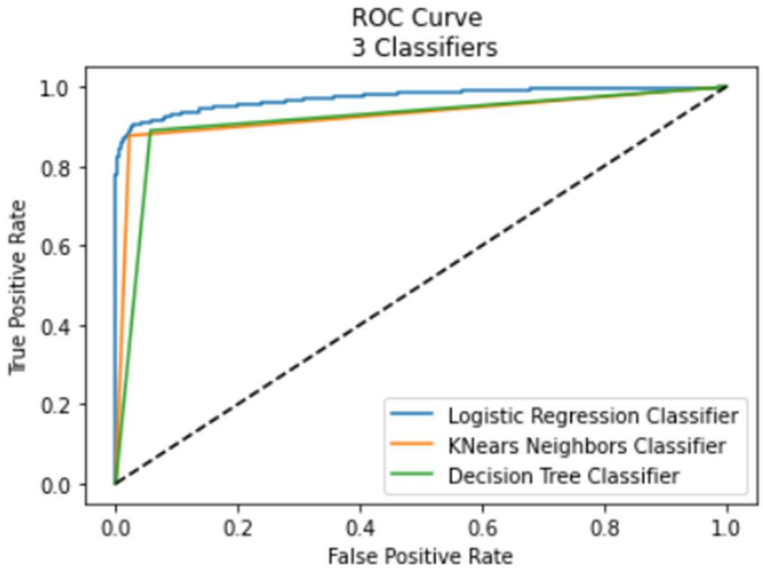
F1 = 2 * (precision * recall) / (precision + recall)

Method	Recall score	Precision score	F1 score	accuracy
K-Neighbor	0.95	0.91	0.93	0.92
Decision Tree	0.97	0.66	0.79	0.71
Logistic Regression (under sample)	0.98	0.69	0.81	0.74
Logistic Regression (over sample)	0.93	0.53	0.55	0.98

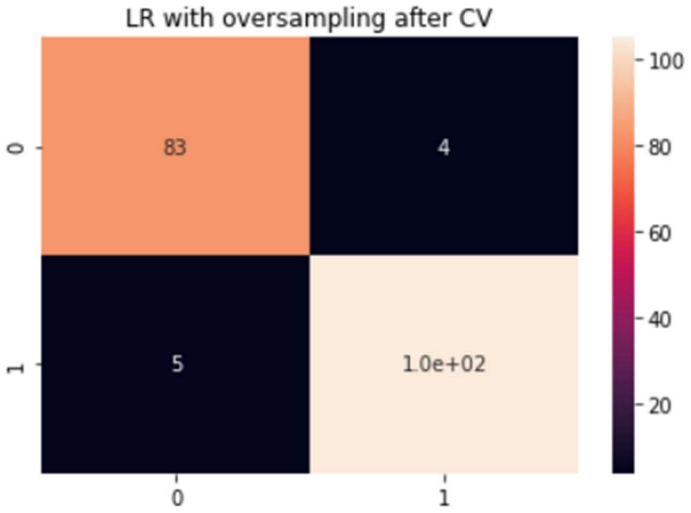
Therefore, from the table, the cross-validation score and train score form previous part. I can say that logistic regression classifier shows the best score. However, it is not enough, and I also want to see the ROC curve and score to see the classifier performance. ROC can illustrate the diagnostic ability for our binary classifier. It is created by true positive rate vs false positive rate:

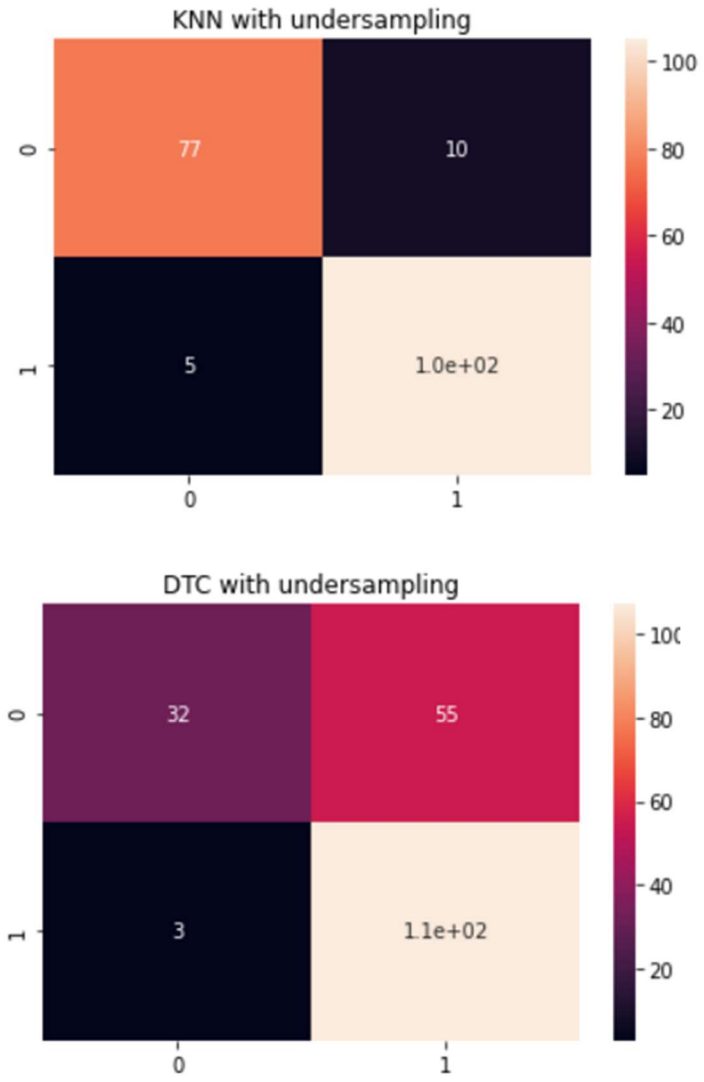
Method	ROC score
K-Neighbor	0.9274
Decision Tree	0.9166
Logistic Regression	0.9727

The curve is like this:



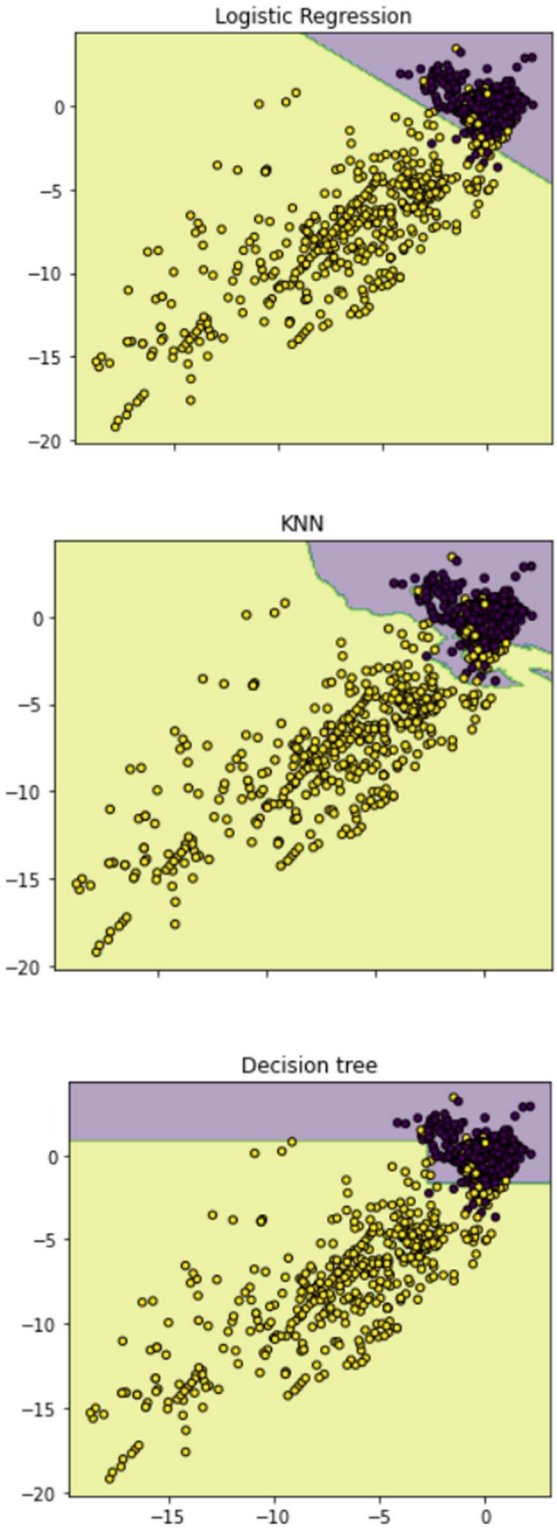
After these, I also state confusion matrix heat map for the three classifiers for subsample because confusion matrix is not good for imbalanced data such as our original dataset.





By compare all three methods, I will choose logistic regression to test the original data set since the logistic regression has the best performance. The results will be shown in next part.

Moreover, I followed the instructions to plot the decision boundary with two significant features V12 and V14 (the plotting code is changed from internet):


Logistic Regression
0
−5
−10
−15
−20
KNN
0
−5
−10
−15
−20
Decision tree
0
−5
−10
−15
−20
−15
−10
−5
0

4. Final Results and Interpretation

After analyzing the score from last part with other baselines, I can say that logistic regression has best performance in under sampling. In this part, I will see logistic regression deeply. Then I will display average precision score. In our data set case, if the average precision score is lower, the performance is better.

From our logistic regression classifier, for under sampling technique, we got the average precision after we test it using original test data:

```
Average precision-recall: 0.08
```

Then here is the classifier using SMOTE over sampling technique, we got the average precision after we test it using original test data:

```
Average precision-recall of oversample by logestic regression: 0.73
```

Therefore, the under-sampling method for logistic regression should be better. The parameter that for this is:

```
Best Parameters of under sample LR:
{'C': 1, 'penalty': 'l2'}
```

From this project, I learned three most important parts that I would like to share. First, we should never test the over sample or under sample data sample. We should test on subsample or original dataset. Second, when we need to implement cross validation, we need to oversample or under sample my training data during cross-validation. Third, we should try not to use accuracy score as metric for our imbalanced data because the accuracy score will be usually high so that misleading. In our case, we should try to use f1 score, precision score as metric to see the classifier metric. Moreover, we should not use MSE to see the result because it will also easily be influenced by imbalance data. Besides these, I also learned under and over sampling techniques and data preprocessing scaling. It is very challenging for me to use knowledge from course into real world. I practiced a lot.

In the project, the logistic regression method with under sampling is the best system. I think the reason is that it works better for less datapoints in case of 30 features. In other words, it will over fit if we use over sampling technique. The overfitting when training 2 hundred thousand data point will be huge. We should have more ways to improve the system. I think one way should be continuing deal with data prepossessing. Because we have 30 features, we can learn about all features and see correlation, then remove the useless features or outliers. I should consider about it next time.

5. Contributions of each team member-Single

6. Summary and conclusions

In conclusion, I figured out that dealing with imbalanced data will be usual in our real world. This dataset indeed is a classic problem in machine learning. Like in our card fraud problem, because of the machine learning algorithm, we will find our transaction will be blocked in real world. When training the imbalanced data, we should use under sampling or over sampling techniques depend on dataset situations. Moreover, by solving the binary output card problem, logistic regression is the best solution method for our data set. By check various scores, we will also learn which scores are important in imbalanced data and which scores are not. For the useful metric, we have F1 score, precision score and ROC score. For the misleading metric, we have accuracy score. In the project, we also can learn the ROC form learning curve which is also show performance of our algorithm.

7. References

[1] "DEALING WITH IMBALANCED DATA: UNDERSAMPLING, OVERSAMPLING AND PROPER CROSS-VALIDATION," 17 August 2015. [Online]. Available: https://www.marcoaltini.com/blog/dealing-with-imbalanced-data-undersampling-oversampling-and-proper-cross-validation

[2] Nagesh Singh Chauhan, "Decision Tree Algorithm Explained," January 2020[Online]. Available: https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html

[3] "Receiver Operating Characteristic (ROC) Curve: Definition, Example," 27 August 2016. [Online]. Available: https://www.statisticshowto.com/receiver-operating-characteristic-roc-curve/

[4] "Machine Learning Basics with the K-Nearest Neighbors Algorithm," 10 September 2018. [Online]. Available: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

[5] "Accuracy, Precision, Recall or F1," 15 march 2018. [Online]. Available: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9