MATH5470 Statistical Machine Learning
Final Project Report

Are your customers happy?

Yizhe Deng, Yifei Huang, Jiaze Sun, Jingming Wang

May 18, 2018

Abstract

Class imbalance is a frequently encountered problem in classification tasks, where the training data is not evenly distributed across all the classes. In most cases, one or more classes contain significantly more samples (i.e. the majority classes) than the other classes (i.e. the minority classes), thus often causing classifiers to make considerably worse predictions for the minority classes. To tackle this issue, numerous methods have been proposed over the years, including over-sampling, under-sampling, feature selection, cost sensitive learning, ensemble learning, and so forth. In this report, we shall apply some of the proposed methods above to a real world data set, *Santander Customer Satisfaction*, and subsequently draw a comparison of their performances. We will eventually show that using random forest without parameter tuning, under-sampling with ensemble learning performs significantly better than over-sampling; moreover, gradient boosting with parameter tuning produced the best overall result even without applying data level manipulations.

1 Introduction

1.1 Customer Satisfaction

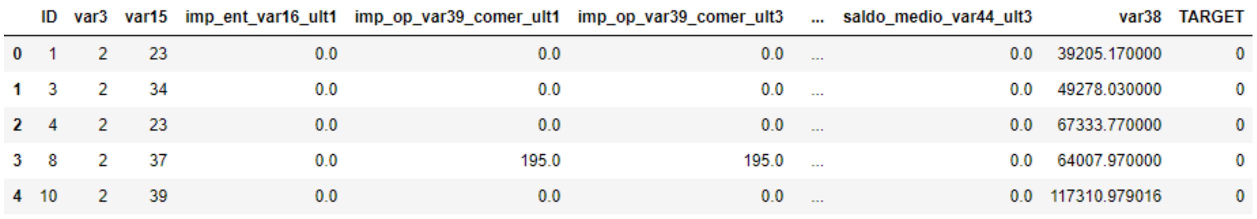
A key characteristic of a successful business is its ability to retain customers. Those who are happy with the services are undoubtedly not difficult to persuade. The challenge, however, lies with those who are not extremely impressed or perhaps even dissatisfied to a certain degree. The first step to retaining these customers is to *know* that they are actually unhappy, but this is already an exceptionally difficult task because unhappy customers usually do not voice their dissatisfaction, and those who do might already be too displeased to even consider staying. Hence, a better strategy is to know which customers are dissatisfied in advance so that those relationships can still be saved before it is too late.

The situation essentially boils down to a class-imbalance binary classification problem. The imbalance comes from the observation that most customers of a multi-billion dollar company should be satisfied, and only a few should be dissatisfied. Although having this imbalance reflects the company's performance positively, it nevertheless poses a substantial issue for statistical learning, as classifiers that are trained on imbalanced data usually perform quite poorly [2].

1.2 The Data

Our data is obtained from the Kaggle Santander Customer Satisfaction competition [6], in which competitors are asked to identify as accurately as possible the displeased customers. The data, provided by the Santander Bank, contains 76,020 samples (customers) and 371 columns. As shown in Figure 1, all the columns are anonymized to protect customer confidentiality except for two, `ID` and `TARGET`. `ID` simply assigns each individual a unique ID. As for `TARGET`, a `0` indicates a customer is satisfied, and a `1` means they are dissatisfied. `TARGET` will serve as the label Y in our subsequent supervised learning.

The fact that the features are anonymized presents a hindrance for us to know exactly what makes a customer satisfied or dissatisfied. In fact, such information may very well point us towards business strategies that the company might not be willing to share. Nonetheless, our emphasis will be on imbalanced classification, so the content of the features are irrelevant.



	ID	var3	var15	imp_ent_var16_ult1	imp_op_var39_comer_ult1	imp_op_var39_comer_ult3	...	saldo_medio_var44_ult3	var38	TARGET
0	1	2	23	0.0	0.0	0.0	...	0.0	39205.170000	0
1	3	2	34	0.0	0.0	0.0	...	0.0	49278.030000	0
2	4	2	23	0.0	0.0	0.0	...	0.0	67333.770000	0
3	8	2	37	0.0	195.0	195.0	...	0.0	64007.970000	0
4	10	2	39	0.0	0.0	0.0	...	0.0	117310.979016	0

Figure 1: The first 5 rows of the data (not all columns are shown).

1.3 Objective

The objective is straightforward: train a binary classification model $\hat{Y} = f(X)$ so that the prediction $\hat{Y}$ is as close to the truth Y as possible. X is the input matrix containing 369 features (i.e. excluding `ID` and `TARGET`). The main obstacle in this task is the class imbalance problem. Amongst all 76,020 customers, 73,012 are satisfied and only 3,008 are dissatisfied, making the former group 24 times the size of the latter (see Figure 2). In this project, we shall attempt to improve the classification performance for the Santander Customer Satisfaction data set by applying a range of different methods, including over-sampling, under-sampling, and ensemble learning. In the end, we shall draw a comparison between all these methods.

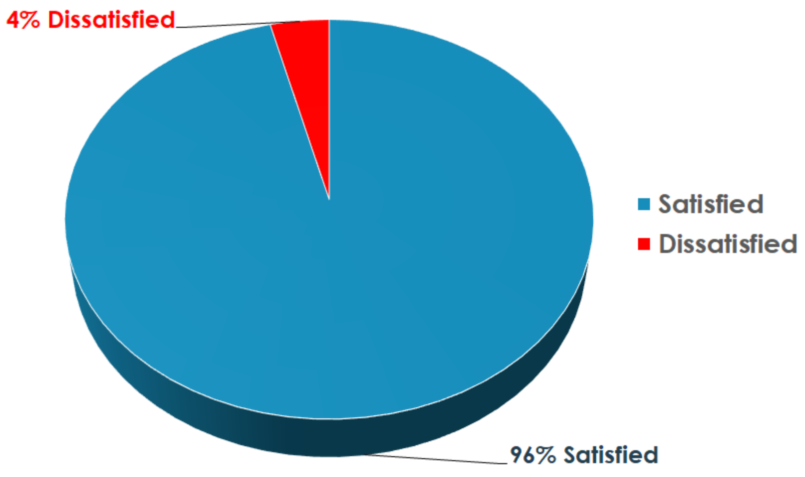


Figure 2: The class distribution of the data set.

2 Exploratory Data Analysis

2.1 Data Pre-processing

Cleaning the data usually improves the efficiency of machine learning. In our project, we first trimmed the number of features from 369 to 319 by removing columns that are repeated or constant, as they contribute nothing to the classification. Afterwards, we normalized the data to be zero-centred with unit variance.

2.2 Visualization

Since this is a high-dimensional problem, we used PCA to reduce the dimension from 319 to 2 so that all data points can be easily visualized. From Figure 3, one can see that the two classes are not very well separated. In fact, performing K-means and DBSCAN (Figure 4 and 5) on the 2-dimensional PCA data illustrates this point. K-means, as expected, was incapable of separating the two classes. DBSCAN clustered the points into more than 20 groups, but in Figure 5, we only show the top 2 clusters (green and orange), whereas all the others are put into one group (black). This way, DBSCAN could somewhat distinguish the 2 classes, but was still unsatisfactory compared to the truth.

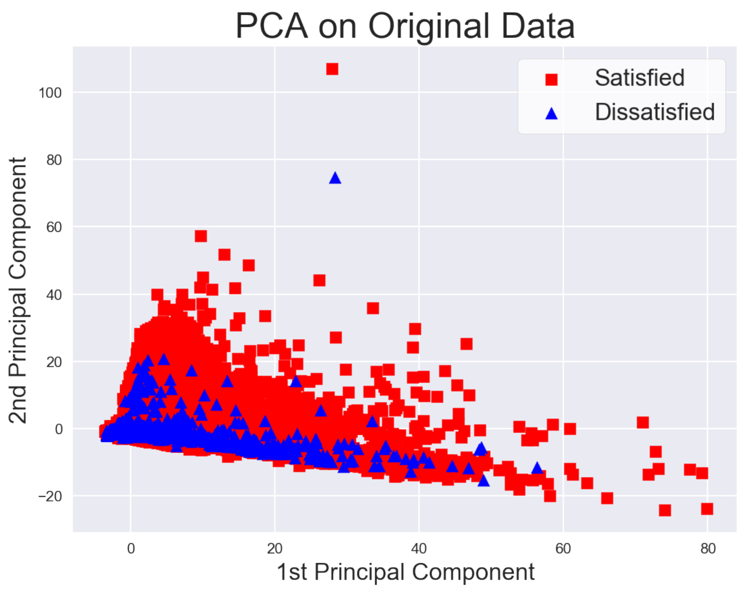


Figure 3: Performing PCA on the data.

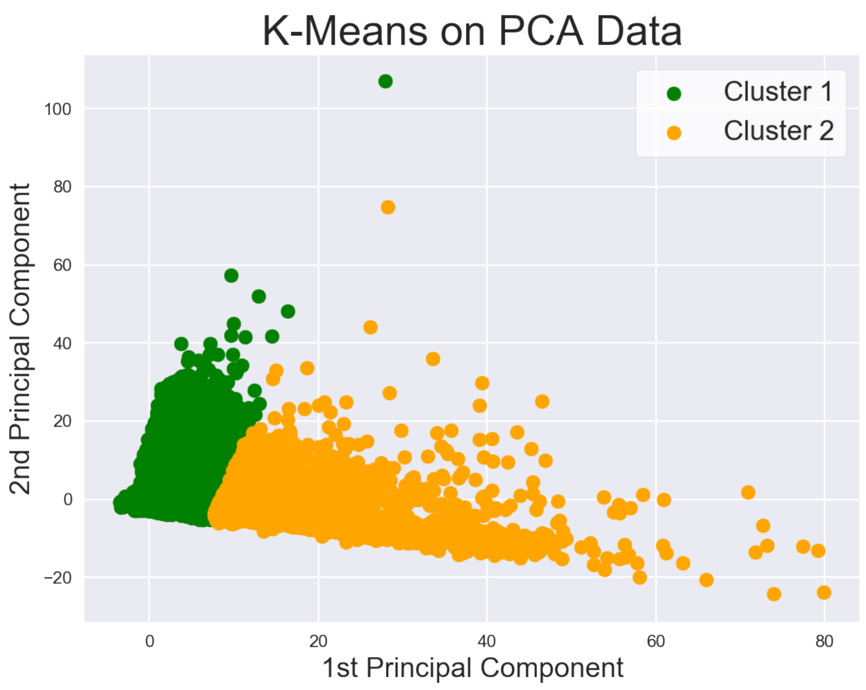


Figure 4: Performing K-means on the PCA data.

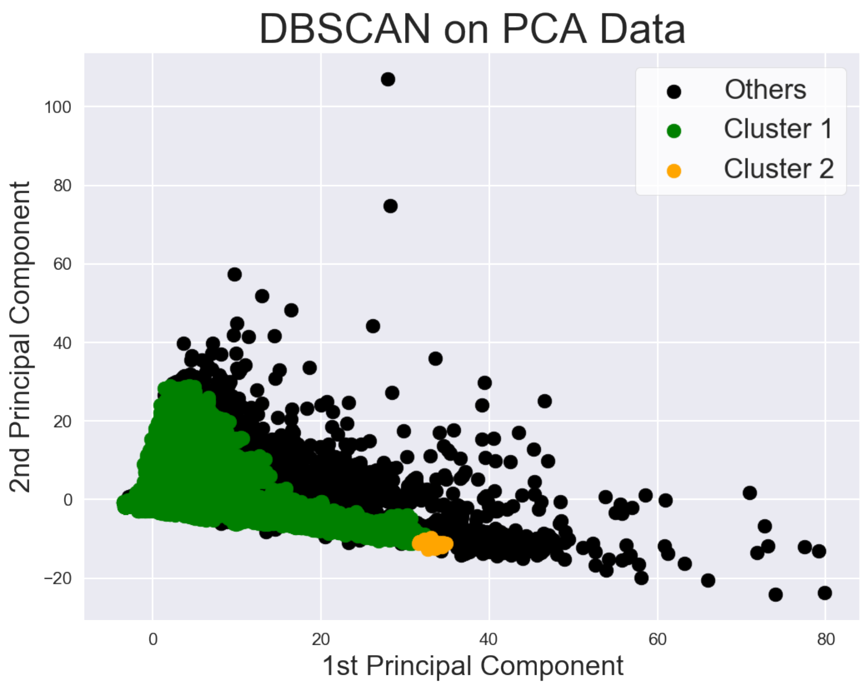


Figure 5: Performing DBSCAN on the PCA data.

3 Methodology

In this section, we give an overview of all the methods that we have used in our project, including the techniques of dealing with imbalanced data sets, and the classifiers.

3.1 Techniques for Imbalanced Classification

3.1.1 Over-sampling

As its name suggests, over-sampling is a data-augmentation technique whose purpose is to *enlarge* the minority class so that it matches the majority class. In this project, we studied 2 types of over-sampling methods:

1. **Naïve over-sampling**: For this method, we randomly duplicate samples from the minority class until the two classes are balanced (see Figure 6) [4].

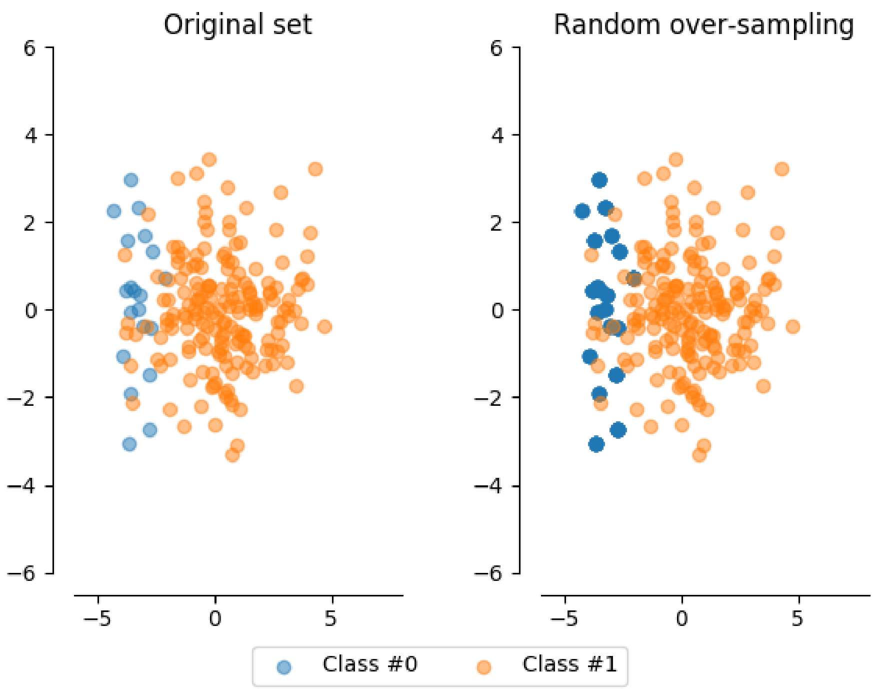


Figure 6: An illustration of naïve oversampling. Note that the blue dots on the right are thicker because of repeated sampling.

2. **Synthetic minority over-sampling technique (SMOTE)**: The effectiveness of naïve over-sampling is usually quite limited, so to improve upon that, SMOTE was introduced. Now, instead of using the same data points from the minority class, SMOTE constructs new points (synthetic points) around the original minority data points [7]. To be precise, for each point, one choose another point from its k-nearest neighbours (amongst the minority group) and draw a straight line between the two, then a random point on the straight line is chosen and added to the minority class (see Figure 7) [1].

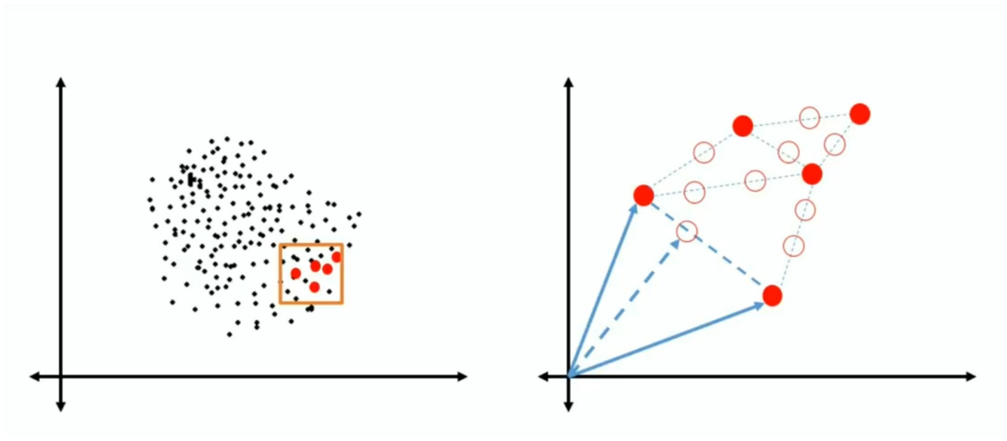


Figure 7: An illustration of SMOTE.

3.1.2 Under-sampling

As opposed to over-sampling, under-sampling randomly removes points from the majority class so that the resulting two classes are balanced (see Figure 8) [5].

However, information regarding the majority class is lost during under-sampling, and the resulting classifier might not be able to capture fully the characteristics of the majority class. To compensate for this, we shall use an ensemble learning method, in which we obtain 50 subsamples via under-sampling, train an classifier for each subsample, and make a final prediction by averaging the individual predictions of the 50 classifiers. This way, we can have as much coverage of the whole data set as possible.

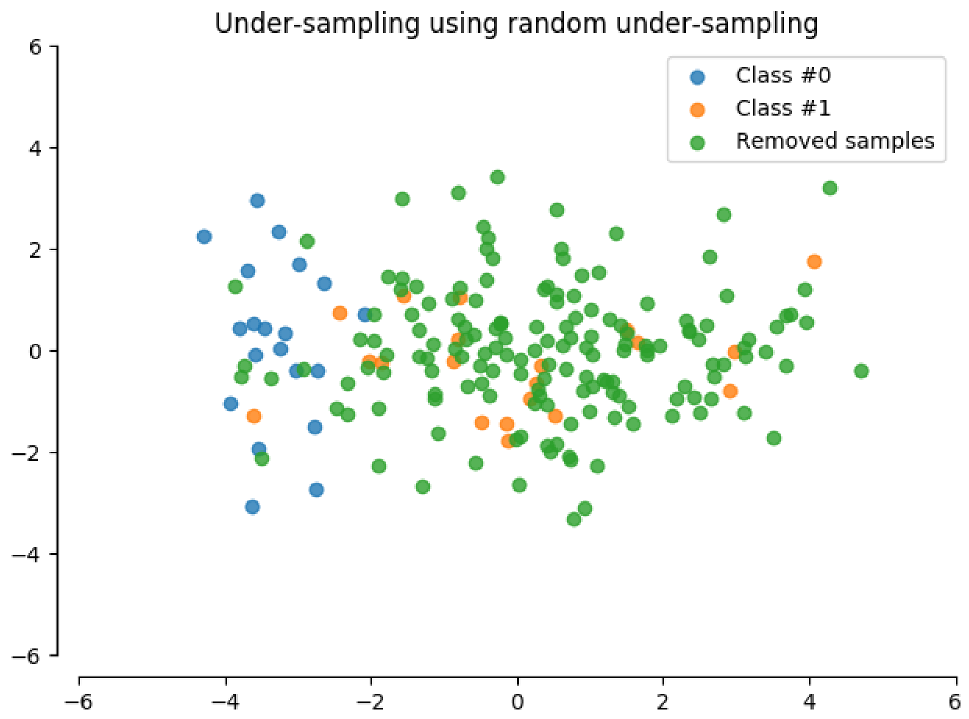


Figure 8: An illustration of under-sampling.

3.2 Classifiers

3.2.1 Random Forest

Random forest is based upon the idea of decision tree and bagging. It performs bagging in 2 ways: each time, the algorithm selects a random sample (with replacement) out of not only all the training examples, but also all the features, and builds a decision tree using this bootstrapped data set. The algorithm repeats this many times and ends up with many trees, thus gaining the name random *forest* (see Figure 9). Finally, a prediction is made by obtaining a majority vote from all the trees.

Random forest retains the nice properties of decision trees, such as having a low bias. It also improves upon decision trees by having a lower variance, which is a result of bagging. In addition, unlike ordinary tree bagging, random forest can effectively handle data having correlated columns, as each tree is built upon a different random set of features.

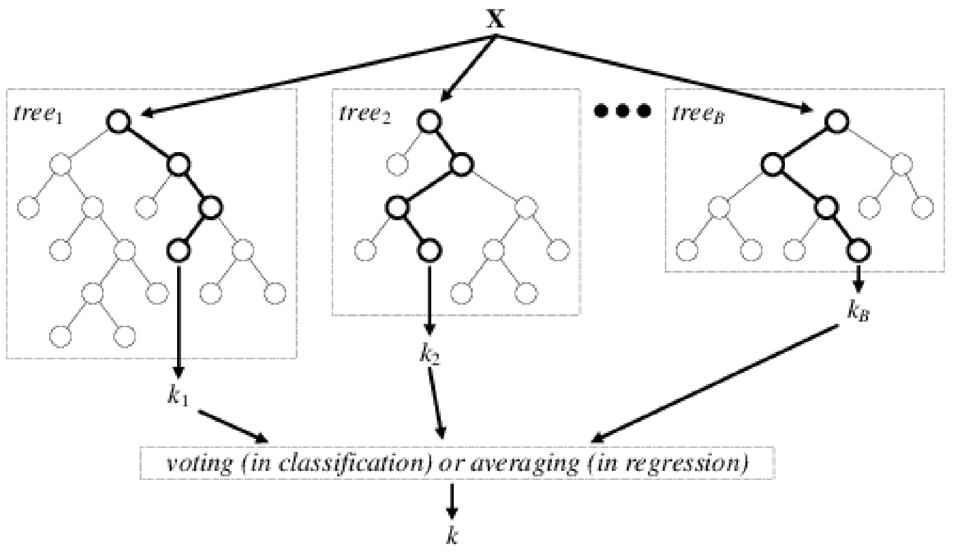


Figure 9: An illustration of random forest.

3.2.2 Gradient Boosting

Boosting is another commonly used method to deal with class imbalance. In fact, it is so effective that it has been described as 'an advanced data sampling technique' [3]. Amongst a plethora of boosting methods, for this project we chose the gradient boosting method, which consists of 3 parts:

- **Loss function**: The exact form of the loss depends on the type of problem involved, but it more or less describes the difference between the truth and the prediction. A benefit of the gradient boosting framework is that a new boosting algorithm does not have to be derived for each loss function that we may want to use, instead, it is a generic enough framework that any differentiable loss function can be used.
- **Weak learners**: Gradient boosting makes predictions through weak learners, which can be any suitable models. In our project, we used decision trees are as our weak learners. Trees are constructed in a greedy manner, choosing the best split points based on purity scores like Gini or to minimize the loss. This is to ensure that the learners remain weak, but can still be constructed in a greedy manner.
- **Adding weak learners**: Gradient boosting is an ensemble learning method, and it builds an ensemble by adding a group of weak learners together. Every time the

algorithm decides to add a tree, it must ensure that adding the predictions of this new tree can reduce the loss in the maximum possible way, and that is exactly achieved by following the opposite direction of the gradient of the loss function with respect to the previous learner, which gives it the name *gradient* boosting. Essentially, each learner added is an attempt to correct the mistakes that the previous ones make. The iteration stops when the loss is sufficiently small or the model no longer improves.

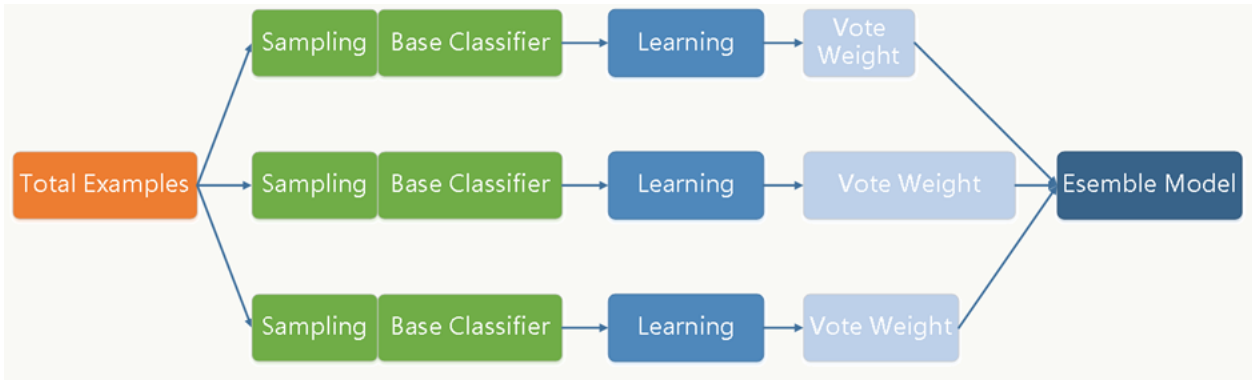


Figure 10: The idea of random forest.

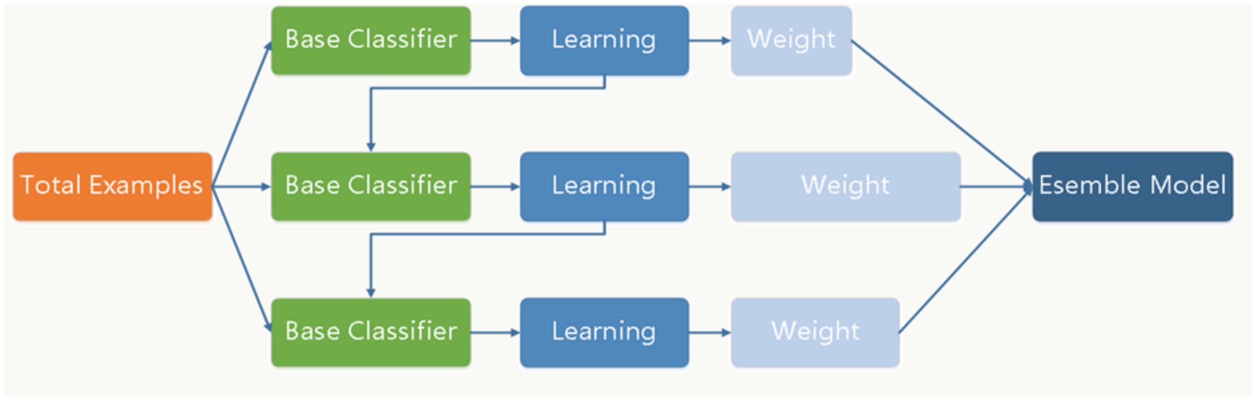


Figure 11: The idea of gradient boosting.

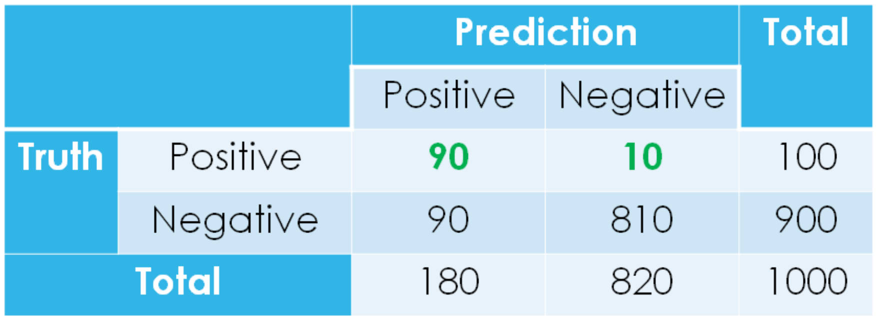
4 Classification Results

As no pre-made validation data set was provided, we separated 500 samples from each class to form a balanced validation data set of 1,000 samples. The rest were used as training examples. All final results are obtained by testing the models on the validation data set.

4.1 ROC v.s. Accuracy

To evaluate the performance of our models, we need a metric that is representative and suitable for imbalanced data sets. This automatically leaves accuracy out of the question, and an obvious alternative is the receiver operating characteristic (ROC) curve, as the latter monitors both the True Positive Rate (TPR) and the False Positive Rate (FPR). The advantage of ROC curves can be illustrated with an example.

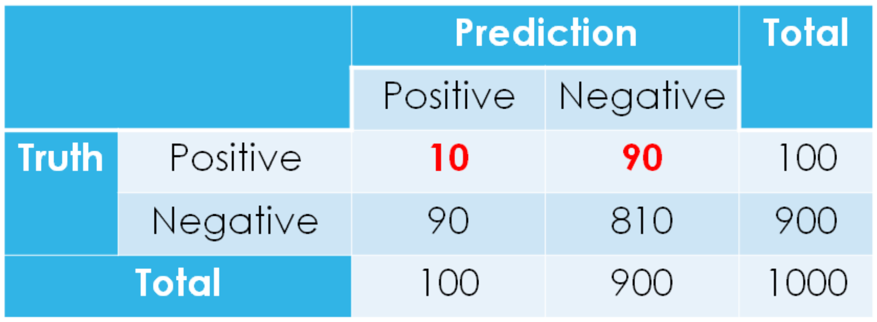
- Suppose a classifier gives the results as in Figure 12, then the accuracy is 90%. The **(TPR, FPR)** is $(90\%, 10\%)$, representing a point on the top left corner of the ROC plot. In this case, both metrics agree that the classifier is decent.



		Prediction		**Total**
		Positive	Negative	
Truth	Positive	**90**	**10**	100
	Negative	90	810	900
Total		180	820	1000

Figure 12: A good classifier.

- Now suppose a classifier gives the results as in Figure 13, then the accuracy is 82%. However, the **(TPR, FPR)** is $(10\%, 10\%)$, which is a point on the line $y = x$. In this case, accuracy still insists that the classifier is not bad, whereas ROC clearly shows that the classifier is about as good as random guessing.



		Prediction		Total
		Positive	Negative	
Truth	Positive	**10**	**90**	100
	Negative	90	810	900
Total		100	900	1000

Figure 13: A bad classifier (random guessing).

4.2 Results

Using the methods we have described in the last section, we can now draw a comparison between all the methods we have tested (see Figure 14).

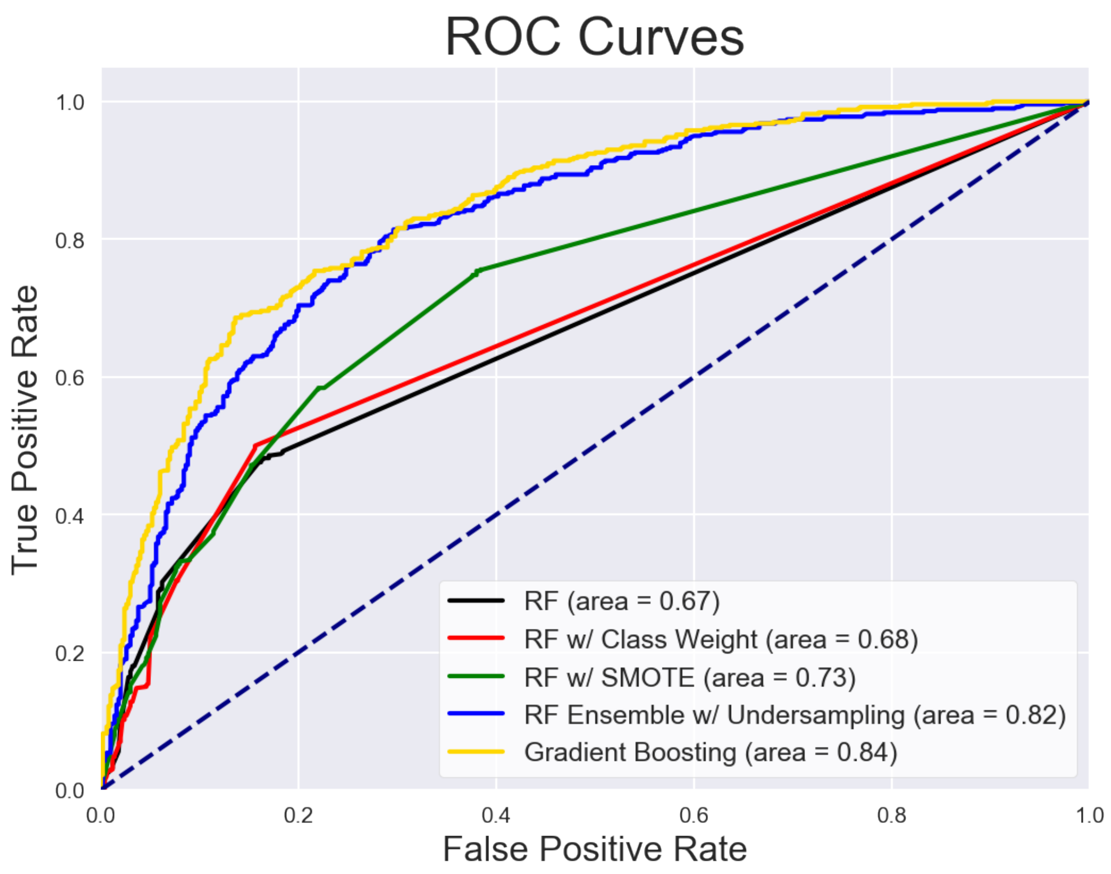


Figure 14: The ROC curves of various methods.

- **Random forest (RF) only**: Using RF with no other techniques yielded an ROC AUC (i.e. area under the ROC curve) of 0.67, which is only mildly better than random guessing.
- **RF with naïve over-sampling**: Labeled as "RF w/ Class Weight" in Figure 14), this method gave us an ROC AUC of 0.68, which is only marginally better than RF alone. In fact, this method did not seem to improve the classification results at all, as in some trials it actually performed worse than just using RF.
- **RF with SMOTE**: Using SMOTE on the data set produced an ROC AUC of 0.73, a slightly better result than the previous two methods.
- **RF ensemble with under-sampling**: As mentioned in the last section, the prediction was generated by averaging 50 RF models that were trained separately on 50 subsamples. As shown in Figure 14, this method produced significantly better results than all previous methods, obtaining an ROC AUC of 0.82.
- **Gradient Boosting**: For this method, we did not implement any data augmentation or pruning. Using the `XGBoost` package with further parameter tuning, we were able to obtain an ROC AUC of 0.84, which is our best overall result for this project.

5 Discussion

We have studied various techniques of dealing with class imbalance in classification tasks, including data level approaches such as naïve over-sampling, SMOTE, ensemble with under-sampling, as well as using alternative models such as gradient boosting. Our main results are discussed below.

- **RF alone is not sufficient.** A possible explanation for this is that RF first draws a random sub-sample from the original imbalanced data set, making the information for the minority class even more scarce. As Ali et. al. have shown [2], if the number of samples is not sufficient, class imbalance will become a too big an obstacle for the classifiers to learn anything useful about the minority class (see Figure 15).

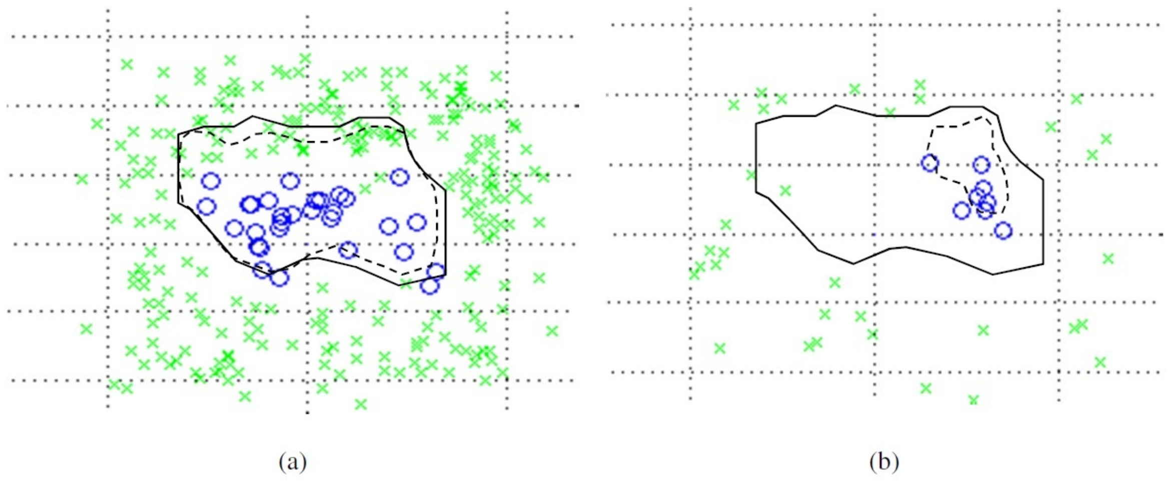
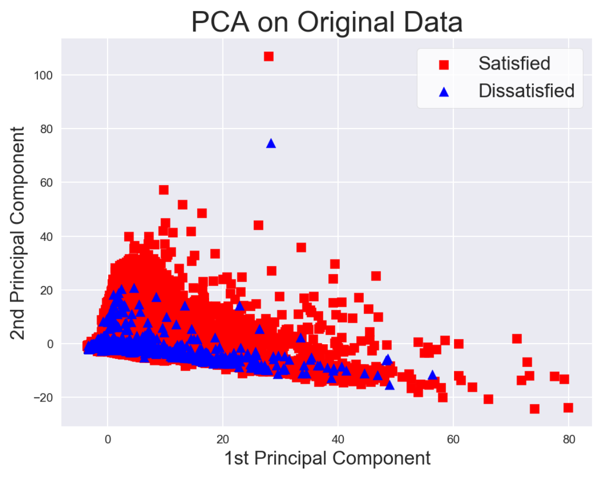


Figure 15: When number of samples is insufficient, classifiers tend to come up with bad decision boundaries (right). Here, solid black lines are the true decision boundary, while the dashed lines are estimated.

- **SMOTE works better than naïve over-sampling.** The latter achieves class balance by duplicating the minority class, which, although makes it more likely for RF to draw balanced bootstrap samples, provides no new information for learning, thus quickly leading to over-fitting. SMOTE avoids this problem by generating new 'fake' data (see Figure 16), thereby producing better results. However, SMOTE still could not give the best result, possibly due to the bias introduced by the 'fake' data.
- **Under-sampling is better than over-sampling.** The reason why under-sampling performs significantly better is because it avoids all the cons that the previous methods



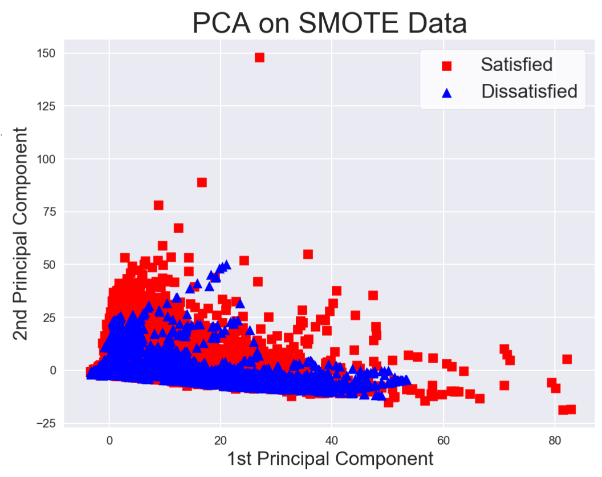


Figure 16: Instead of duplicating, SMOTE generates 'fake' data around original ones.

have. First, it solves the class imbalance problem, every subsample drawn by RF is now evenly distributed. Second, unlike SMOTE, it does not introduce additional bias to the data set, thus giving a better generalization result. As a result, even without parameter tuning, it was able to match gradient boosting in terms of performance.

The only drawback of this method is the loss of information during under-sampling. Despite that we used 50 subsamples, it is not guaranteed that every example is chosen; even then, some examples are chosen more often than others. This might be why it still performed worse than gradient boosting.

- **Gradient boosting works well without data level manipulations.** It is not surprising that gradient boosting is able to beat all the previous methods, as it avoids all the disadvantages mentioned before: loss of information and additional bias. Without manipulating the actual data, it also solves the class imbalance problem. This is due to the fact that the minority examples are more likely to be misclassified in the early run, and thus are more likely to be corrected in the subsequent iterations.

6 Future Work

Class imbalance is a very well known problem in statistical learning, and there are still many possible directions for future research. First, there are many other methods for over-sampling and under-sampling which we did not cover in this project, such as improved versions of SMOTE, ADASYN, Tomek links, and so on. Another data level approach is feature selection. In our project, we only performed a basic feature selection by removing those which are repeated or constant, while in fact we can refine our selection even further, for instance, by controlling the ROC AUC [2]. Apart from data level methods, there are also algorithm approaches that we did not explore, such as improved algorithms, one-class learning, and cost-sensitive learning. In addition, one can also explore alternative classifiers such as SVM, logistic regression, neural networks, and so forth.

References

[1] Smote - synthetic minority oversampling technique. `https://www.youtube.com/watch?v=FheTDyCwRdE`.

[2] Siti Mariyam Shamsuddin Aida Ali and Anca L. Ralescu. Classification with class imbalance problem: A review. *International Journal Of Advances In Soft Computing And Its Advances*, 7(3):176–204, 2015.

[3] Jason Van Hulse Chris Seiffert, Taghi M. Khoshgoftaar and Amri Napolitano. Rusboost: A hybrid approach to alleviating class imbalance. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS – PART A: SYSTEMS AND HUMANS*, 40(1):185–197, 2010.

[4] Imbalanced-Learn. Random over-sampling. `http://contrib.scikit-learn.org/imbalanced-learn/stable/auto_examples/over-sampling/plot_random_over_sampling.html`.

[5] Imbalanced-Learn. Random under-sampling. `http://contrib.scikit-learn.org/imbalanced-learn/stable/auto_examples/under-sampling/plot_random_under_sampler.html`.

[6] Kaggle. Santander customer satisfaction. `https://www.kaggle.com/c/santander-customer-satisfaction`.

[7] Lawrence O. Hall W. Philip Kegelmeyer Nitesh V. Chawla, Kevin W. Bowyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.