Capstone Project Norah Alhomaimidi
Machine Learning Engineer Nanodegree February 8st, 2019

Santander Customer Satisfaction

Definition

Project Overview

Since customer satisfaction is an essential factor to success in any service industry, identify dissatisfied customers early and maintain them before leaving is the most challenging task. Especially that Unhappy customers rarely voice their dissatisfaction. Therefore, the longer a client stays with an organization, the more value he creates.
By focusing on the operating environment of the banking market, it observed that it is very challenging and competitive because of its nature for the profit growing needs, against the clients' variable demands. Thus, banks' Customer Relationship Management increasingly focused on identifying customer segments, needs, and satisfaction to avoid customer churn.
So, in this project, I trained and tested a binary classifier on Santander Bank data, which were one of the Kaggle competitions in 2015, to predict whether a given customer is a satisfy or not.

Problem Statement

Santander Bank, which is a large corporation focusing principally on the market in the northeast United States, have asking Kagglers to help them identify dissatisfied customers early in their relationship through a Kaggle competition (Santander, 2015). This competition had aimed to predict whether a client will be dissatisfied in the future or not based on specific characteristics, to help Santander Bank to take proactive steps regarding improving a customer's happiness before it's too late and customers already left the bank. This competition has been provided by anonymized dataset containing a large number of numeric variables. This Data has been split into training and target sets.
To solve this problem, first of all, I decided to work on only the training set because it contains the predicted variable TARGET. Then, I discovered the training data to gain some basic understanding for the data set characteristics; I also conducted a PCA technique on the data to reduce dimensionality space of features into a smaller number of predictor variables that can represent the data very well. Second, I split the data in different ways into train and test datasets, then train and test the data on different classification algorithms as well; to pick the model that has the highest accuracy and the best performance to predicting the customer satisfaction information.

Metrics

- **Accuracy:**
  is one of common metric for evaluating binary classification models by taking in account both true positives and true negatives with equal weight.
  Formally, accuracy has the following definition:

$$Accuracy = \frac{True\ positives + True\ negatives}{Total\ number\ of\ predictions}$$

- **Precision:**
  is the number of True Positives divided by the total number of positive class values (True Positives and False Positives). A low precision can also indicate a large number of False Positives.
  Formally, precision has the following definition:

$$Precision = \frac{True\ positives}{True\ positives + False\ positives}$$

- **Recall (Sensitivity):**
  is the number of True Positives divided by the number of positive class values (True Positives and False Negatives). A low recall indicates many False Negatives.
  Formally, recall has the following definition:

$$Recall = \frac{True\ positives}{True\ positives + False\ negatives}$$

- **Fβ-Score:**
  It considers both precision and recall, where the Fβ score is the harmonic average of the precision and recall, where an Fβ score reaches its best value at 1 (perfect precision and recall) and worst at 0.
  Formally, Fβ-Score has the following definition:

$$F\beta = (1+\beta^2)\frac{Precision\ .\ Recall}{(\beta^2 .\ \mathrm{Precision}) + \mathrm{Recall}}$$

- **ROC Curve (receiver operating characteristic curve):**
  is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters (True Positive Rate and False Positive Rate). Formally, True Positive Rate and False Positive Rate have the following definitions:

$$\text{True positives rate} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}}$$

$$\text{False positives rate} = \frac{\text{False positives}}{\text{False positives} + \text{True negatives}}$$

- **AUC (Area under the ROC Curve):**
  is a measure the entire two-dimensional area underneath the entire ROC curve (value of integral calculus) from (0,0) to (1,1).

Analysis

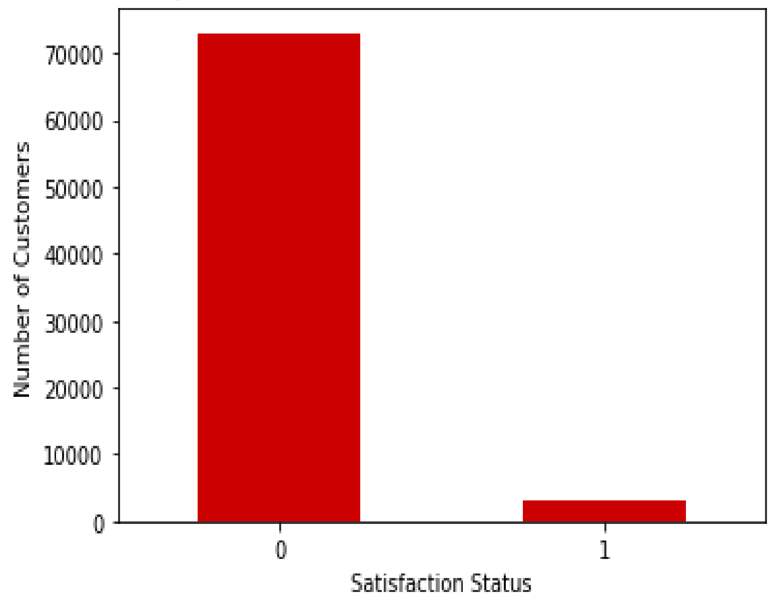
Data Exploration

The training dataset provided on Kaggle competition homepage contains 76,020 rows of data and 371 features included predicted-variable (TARGET); each row represents a different customer. TARGET-column equals 1 for unsatisfied customers and 0 for satisfied customers. However, this dataset was heavily imbalanced in term of predicted variable values, where approximately 96.04% of the customers (73012) were satisfied, and approximately 3.96% of customers (3008) were unsatisfied.
As well as, there were a 6.32% percentage of customers (4807) has the same values for each feature, which can be considered as duplicated records. Out of % 6 of satisfied customers (4614) were duplicated, and Out of % 6 of unsatisfied customers (193). Anyway, since they have different ID I have assumed that they are different customers.
Anyway, there were 34 columns have the same value per record constantly (zero). As, well as, approximately 56 % of features has a high correlation with each other.

Exploratory Visualization

- The plot below shows the heavily imbalanced in customers satisfaction values (predicted variable values):

Figure-1: Customers Satisfaction Status



- The two plots below show the duplicating in Satisfies/Non-Satisfies Customers:

Figure-2: Non Satisfies Customers - Duplicate Records

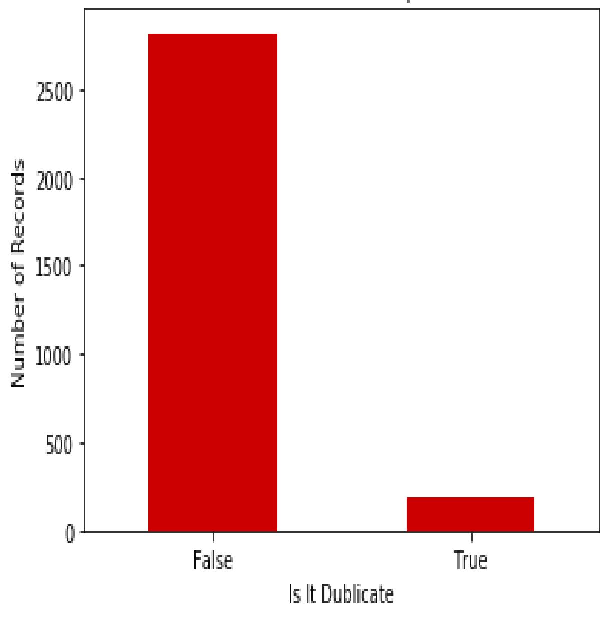
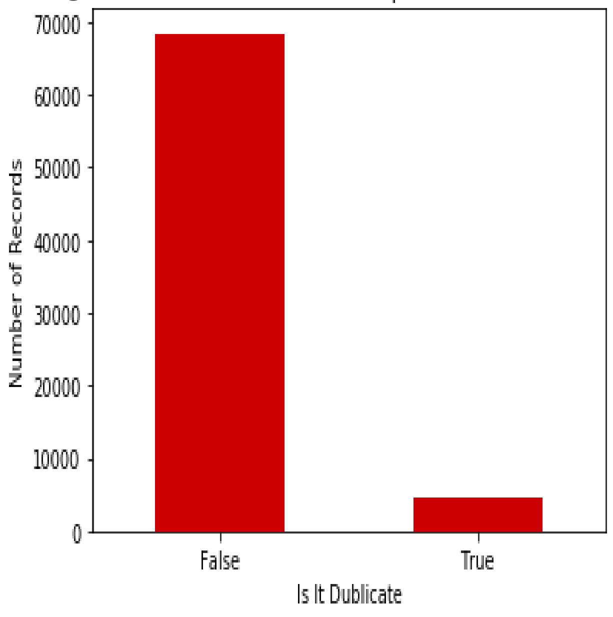
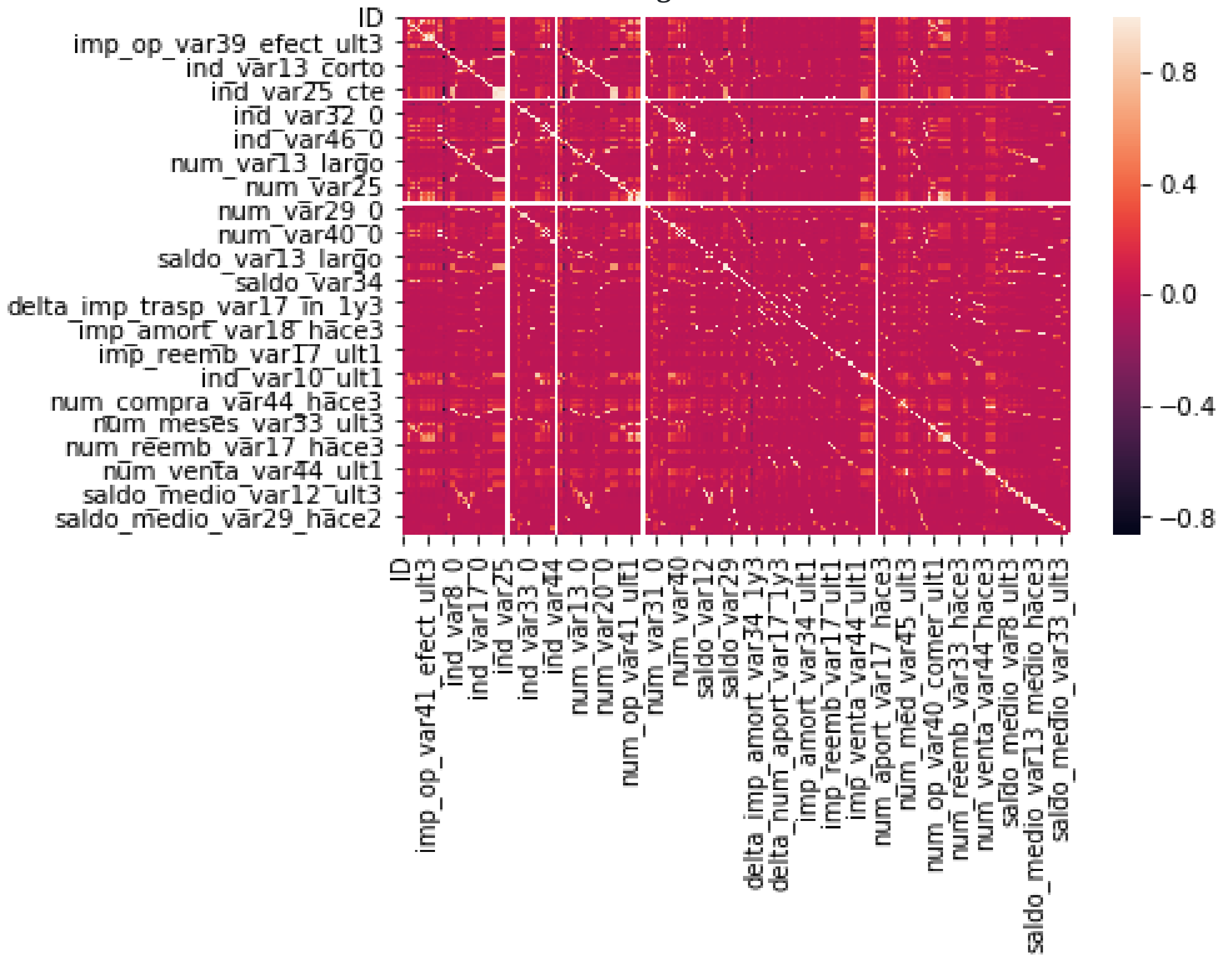


Figure-3: Satisfies Customers - Duplicate Records



- The plot below shows the high correlation between half of the features:

Figure-4:



Algorithms and Techniques

- **PCA Technique:**
  The principal component analysis is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called principal components. Thus, it usually used to minimize the dataset dimensions space (number of features) to fewer compound combinations of features in a way that has the best description of the data.

- **Random Under Sampling Technique:**
  It is a technique used to adjust the class distribution of a dataset, in order to dealing with imbalanced datasets. Where it randomly eliminates instances from the majority class of a dataset and assign it to the minority class.

- **Random Over Sampling Technique:**
  It is a technique used to adjust the class distribution of a dataset, in order to dealing with imbalanced datasets. Where it randomly increases the instances corresponding to the minority class by replicating them up to a constant degree.

- **Synthetic Over Sampling Technique (SMOTE):**
  It is a technique used to adjust the class distribution of a dataset, in order to dealing with imbalanced datasets. Where SMOTE is an oversampling method which creates "synthetic" example rather than oversampling by replacements. The minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. Depending upon the amount of over-sampling required, neighbors from the k nearest neighbors are randomly chosen.

- **Logistic Regression Algorithm:**
  Logistic Regression is a statistical method for binary classification problems. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). It is fast in training and prediction time, and it can avoid overfitting and gives good results in case of fewer features. Also, can be updated easily with new data using stochastic gradient descent.

- **Decision Tree Algorithm:**
  Decision Tree classifier takes a discrete set of values as a target variable called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. In other words, the Decision Trees are diagrams that attempt to display the range of possible outcomes and subsequent decisions made after an initial decision, where the top few nodes on which the tree is split are the most important variables within the dataset. One of the most useful aspects of decision trees that there is no extra effort for data preparation (no need to normalize, scale features), and it is not sensitive to outliers (since the splitting happens based on the proportion of samples within the split ranges and not based on absolute values).

- **Random Forest Algorithm:**
  Random Forest classifier is a meta estimator that fits many decision tree classifiers on various sub-samples of the dataset and uses averaging. I used to improve the predictive accuracy, avoid the overfitting problem, handle missing values very well, and it can be modeled for categorical values. Furthermore, it can be used for both classification and regression tasks.

- **Naive Bayes Algorithm:**
  Gaussian Naive Bayes is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions between the features. It is very simple, which mean easy to implement and fast. As well as, it is highly scalable, which indicate It scales linearly with the number of predictors and data points. Furthermore, it needs less training data, and it can be used for both binary and multiclass classification problems. Finally, it can handle both continuous and discrete data and it not sensitive to irrelevant features.

Benchmark

The benchmark for this classification problem is a Logistic Regression model that predicts the satisfaction status of a certain customer whether he satisfy or not. Since that the prediction result is either 0 or 1, 0 for satisfy and 1 for dissatisfy. Logistic Regression is usually the first model used to predict a binary classification problem. With default parameters, the model scored on the testing dataset 0.4995 of the ROC-AUC metrics. As well as, the execution time for training was 2.873 sec, and the execution time for testing was 0.00328 sec.

Methodology

Data Preprocessing

Overall the dataset was complete, clean, prepared for modeling, and all the features had been set to valid values. However, some kind of cleansing has been done on the data to remove unnecessary features such as remove ID column and removing the constant columns that have been mentioned above.

Since that approximately half of features are highly correlated, which means can be represented by each other, and the other half are not, and to improve model's performance with a smaller number of independent variables, Principal component analysis (PCA) was an essential step in the Data Preprocessing process. PCA is an approach that considers the total variance in the data and transforms the original variables into a smaller set of linear combinations in a way that has the best description of features.

Furthermore, another critical process has been done is Scaling the Features. Because of the highly varying in magnitudes between some feature's values, and the PCA use the distance between data points in its computations (PCA tries to get the features with maximum variance, and the variation is high for high magnitude features, so having highly varying would skew the PCA towards high magnitude features).

Implementation

The implementation was done by Python 3.6, and libraries have used were (numpy, pandas, scikit-learn).

First, the four different models (Logistic Regression, Decision Tree, Random Forest, Naive Bayes) trained and tested in the processed data, but the accuracy results weren't acceptable for all models because of the bias in the target variable. Then, the same four models trained and tested after re-sampling the dataset. Three techniques of re-sampling have been conducted (random under-sampling, random over-sampling, synthetic over-sampling/SMOTE) to comparing the results of accuracy each time.

The evaluation was in two parts (execute performance and accuracy metrics) as followed:
-Execute Performance:
Time of Training in seconds and Time of Testing in seconds.

-Accuracy Metrics:
Fβ-Score, Precision and Recall for dissatisfied customers, and ROC-AUC score on the testing data.

Refinement

After doing some researches, I found out that re-sampling helps to deal with imbalanced datasets. Thus, various strategies of re-sampling have been conducted before training and testing the dataset, and then comparing the results. The re-sampling procedures that have been performed contain random under-sampling, random over-sampling, and synthetic over-sampling/SMOTE. After each re-sampling method, I trained and tested the data, and evaluate models' accuracy. Anyway, the evaluation metrics were much better and had an obvious result after this process, which have got me a vast view. So, based on that I decided to the best model would be the best model in different sampling.

Results

Model Evaluation and Validation

As mentioned above, after the first modeling trail, three different resampling procedures have been implements, and the evaluation results for the whole experiments were as follows:

- **Performance:**

Trail/Algorithm	Logistic Regression	Decision Tree	Random Forest	Naive Bayes
Train Data	2.8731	0.2791	0.4871	**0.0238**
Test Data	**0.0032**	0.0259	0.1449	0.0298

- **Evaluation Metrics:**

Accuracy

Sampling	**Trail/Algorithm**	Logistic Regression	Decision Tree	Random Forest	Naive Bayes
Non	**Test Data**	**0.9579**	0.9441	0.9496	0.0464
Under	**Test Data**	0.4764	0.5479	0.5673	0.5191
Over	**Test Data**	0.5004	0.8965	**0.8987**	0.5002
SMOTE	**Test Data**	0.0442	0.9425	**0.9497**	0.0445

Precision

Sampling	**Trail/Algorithm**	Logistic Regression	Decision Tree	Random Forest	Naive Bayes
Non	**Test Data**	0.02	0.05	0.06	0.04
Under	**Test Data**	0.48	0.52	0.54	0.11
Over	**Test Data**	0.50	0.83	**0.84**	0.50
SMOTE	**Test Data**	0.04	0.03	0.05	0.04

Recall

Sampling	**Trail/Algorithm**	Logistic Regression	Decision Tree	Random Forest	Naive Bayes
Non	**Test Data**	0.00	0.02	0.02	1.00
Under	**Test Data**	1.00	**0.59**	0.58	0.00
Over	**Test Data**	1.00	0.99	0.99	1.00
SMOTE	**Test Data**	1.00	0.02	0.02	1.00

F1-Score

Sampling	**Trail/Algorithm**	Logistic Regression	Decision Tree	Random Forest	Naive Bayes
Non	**Test Data**	0.00	0.03	0.03	0.08
Under	**Test Data**	**0.65**	0.55	0.56	0.00
Over	**Test Data**	0.67	**0.91**	**0.91**	0.67
SMOTE	**Test Data**	0.07	0.02	0.02	0.07

AUC

Sampling	Trail/Algorithm	Logistic Regression	Decision Tree	Random Forest	Naive Bayes
Non	**Test Data**	0.4995	0.5017	0.5035	0.5012
Under	**Test Data**	0.4994	0.5496	0.5676	0.4963
Over	**Test Data**	0.5022	**0.8969**	**0.8990**	0.5019
SMOTE	**Test Data**	0.5021	0.4976	0.5013	0.5022

Justification

The final result of the Decision Tree & Random Forest models was excellent and very close when train/test the models within balanced data that have been picked using Random Over Sample. The ROC-AUC score of Decision Tree on the testing dataset is 0.8969, and the ROC-AUC score of Random Forest on the testing dataset is 0.8990. Those results surpasses the first initial benchmark model in terms of learning the target concept because it's close to 1.
On the other hand, the Decision Tree execution time for training is 0.2791, while the Random Forest execution time for training id 0.487. As well as, the Decision Tree execution time for prediction is 0.02591, while the Random Forest execution time for prediction is 0.1449. In focusing on performance, the first initial benchmark model has better performance, in training or prediction.
Anyway, these results show that the approach is the right direction, although the data sampling for training need so much to improve.

Conclusion

Free-Form Visualization

Figure 5: The change in the number of records after Under-Sampling the data

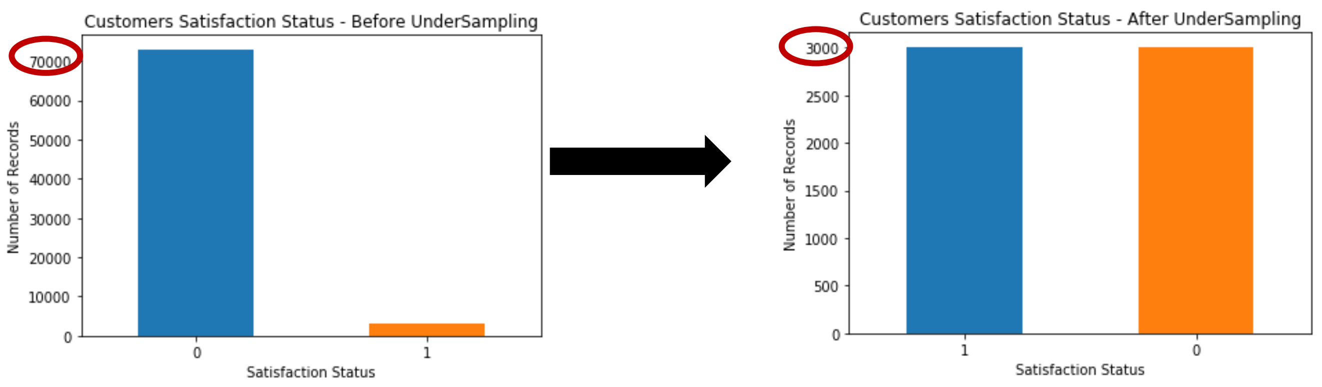


Figure 6: The change in the number of records after Over-Sampling the data

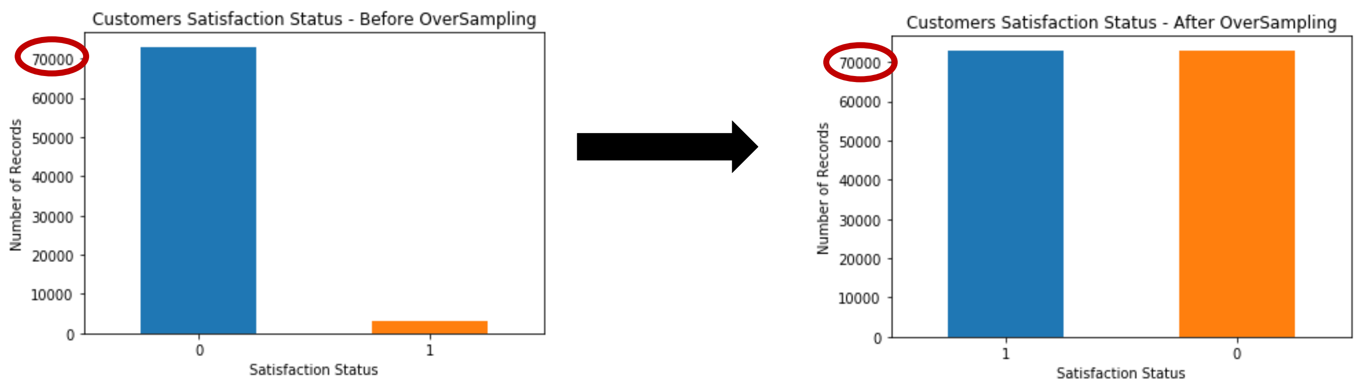
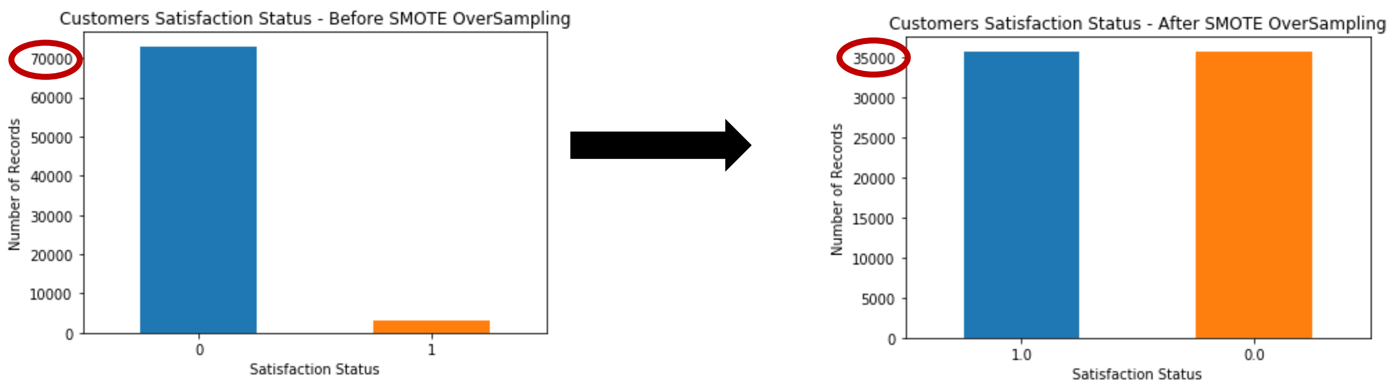


Figure 7: The change in the number of records after SMOTE the data



To solve the imbalance problem in the dataset, I tried different methodologies for sampling the data. The first method, Under-Sampling (figure-5), which removing samples from the majority class (satisfied customers on this dataset). The second method, Over-Sampling (figure-6), which adding more examples from the minority class (unsatisfied customers in this dataset). The second method, SMOTE (figure-7), which consists of synthesizing elements for the minority class by adding synthetic points between the chosen points and their neighbors. Anyhow, It observed that the number of minority class became balanced with the number of majority class, which achieves the balance in training data.

Reflection

First, I build a benchmark model for binary classification problem then try to apply different models to gain butter accuracy results using different algorithms. Initially, I thought working with such as problem will be really easy, but having a high dimensionality space dataset was the first problem I faced in this project. Secondly, I realize that the dataset is imbalanced and require a specific way of treating, which had getting me know that I was wrong and It is not to easy. Having a clear understanding of the data and their nature gives another perspective from different angles in terms of which approaches have in how choose the technical solutions.

Therefore, I tried to use one of reducing techniques (PCA) to feed models with a smaller number of features. As well as, I read a lot about imbalanced dataset to learn how to deal with such problems. I tried to set up different improvements solutions in terms of sampling and training imbalanced data to come up with better accuracy result.

The final models perform were better than my benchmark model, and now I have a better understanding of some concepts about the imbalanced dataset and sampling techniques, that makes me more interested in such as projects and datasets to learn more about them.

Improvement

This was my first experience working with highly dimensionality space dataset, and I found that they require a lot more cleaning computing and reducing. As well as, this was my first experience working with imbalanced data, and I found that very hard to sampling data to training models.

Of course, training models on the data can be better, if I tried different parameters. So, I believe tuning the data by Grid Search to set the optimal hyperparameters would give a better result for different algorithms. As well as, different transformation method could also be used to scale the data to provide a better accuracy score.

Certainly, a different algorithm such as xgboost and LightGBM can produce better performances and high accuracy scores.

References

https://en.wikipedia.org/wiki/Principal_component_analysis

https://www.datacamp.com/community/tutorials/diving-deep-imbalanced-data

https://en.wikipedia.org/wiki/Oversampling_and_undersampling_in_data_analysis

https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall

https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

https://en.wikipedia.org/wiki/F1_score

https://developers.google.com/machine-learning/crash-course/classification/accuracy