**Using Credit Card Customers Dataset to Predict Churning Customers**

The dataset comes from Kaggle.

<https://www.kaggle.com/sakshigoyal7/credit-card-customers?select=BankChurners.csv>

Managers at the bank are disturbed with more and more customers leaving their credit card services. They would really appreciate if one could predict for them who is gonna get churned so they can proactively go to the customer to provide them better services and turn customers’ decisions in the opposite direction. This dataset consists of 10,000 customers mentioning their age, salary, marital\_status, credit card limit, credit card category, etc. There are nearly 22 features. I try to train a model to predict who will leave their credit card services. In addition, another objective is to show the workflow in any predictive modeling problem. What’s more, due to only 16.07% of customers who have churned, I have to deal with the imbalanced data before using it.

In order to enhance the predicting accuracy, some methods I used including oversampling, standardization, models comparison and selection, hyper-parameters tuning, and ensemble learning. By these methods being adopted, the accuracy improves from 86% to more than 94%.

**Data preprocessing**

This dataset consists of more than 10,000 customers mentioning their age, salary, marital\_status, credit card limit, credit card category, etc. There are nearly 22 features. Among them, there are 6 columns data of object type, and I need to convert into number in order to involve into the calculation. The details of data preprocessing as following.

**EDA（Exploratory Data Analysis）**

Before label encoding (factorizing), converting a (string, category, object) column to integers, firstly, I do EDA（Exploratory Data Analysis）. Exploratory Data Analysis or EDA is very crucial for the success of all data science projects. It is an approach to analyze and understand the various aspects of the data. Through EDA, I must understand the relationship among the features and I must be able to make out conclusions or gather insights about the data. The purpose of performing EDA on any dataset is to make sure that the data is clean and there are no redundancies, missing values, or null values in the dataset. I need to identify the significant features in the dataset and remove the unnecessary noise in the dataset that could hamper the accuracy of our conclusions when I work on building the model. EDA is very important for all machine learning problems. Most of the real-world datasets have dirty or unclean data, data may also be unsuitable for machine learning techniques that you want to apply. Several features are shown by figures which help us understand the data. There are no missing data and I can skip the process of filling null value.

**Encoding**

Scikit-learn does not accept non-numerical features. And in our dataset, there are 6 columns features with non-numerical types which contain very useful information. Therefore, it would be preferable to retain the features and convert them into numbers. To do this, I conduct dummy variables and also encode categorical values with a technique called "label encoding", which allows us to convert each value in a column to a number. Numerical labels are always between 0 and n\_categories-1.

**Heatmap**

After that, I adopt a heatmap to analyze the important features. From the heatmap, I find the correlation feature is “Total\_Revolving\_Bal” and the factor is up to 0.64, the minimal feature is “Total\_Trans\_Ct” and the value is 0.003. All the features show a positive correlation to the target and no negative correlation exists. It means an increase in some features will lead to an increase in the target. Eliminating redundant features can reduce training time and many such advantages; therefore, I check the dataset and find that there is no redundant feature in the dataset, and finally, I keep all of them. Since there are only 22 features in the dataset, which is not a relatively big dataset for my PC’s computing power, I keep all the features in order to acquire higher accuracy.

**Data Scaling and Standardization**

A dataset is made up of different values which can be drawn from different distributions, having different scales. A machine learning algorithm isn't naturally able to distinguish among these various situations. Some machine learning algorithms are sensitive to feature scaling, while others are virtually invariant to it. Standardization is a scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation. Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling. I choose standardization in this project because the data follows a Gaussian distribution and does not have a bounding range. Even if some values have outliers in data, they will not be affected by standardization. Additionally, comparing performance for best results by fitting my models, standardized data show a better result. Therefore, I choose the Standardization method to pre-processing my dataset.

**Splitting**

In order to verify the model performance after training and avoid overfitting, I separate train dataset and test dataset independently. In this project, I randomly separate the dataset into train and test dataset with a ratio of 0.75: 0.25 respectively.

**Oversampling and Data Augmentation**

In this dataset, only 16.07% of customers have churned, which will lead to significant differences in the rarity of different classes of data being predicted. The overall performance of any model trained on such data will be constrained by its ability to predict rare points. In problems where these rare points are only equally important or perhaps less important than non-rare points, this constraint may only become significant in the later "tuning" stages of building the model. But in problems where the rare points are important, or even the point of the classifier, dealing with their scarcity is a first-order concern for the model builder. Several different techniques exist in the practice for dealing with an imbalanced dataset. In this assignment, I augment data by methods of random oversampling, SMOTE, and Boderline-SMOTE. The test results show using SMOTE or Boderline-SMOTE method to augment data can improve the prediction performance than using original data or random oversampling data.

**Comparing Models**

The number of shiny models out there can be overwhelming, but I need to how to quickly and efficiently narrow down the space of available models to find those that are most likely to perform best on our problem type. Additionally, different models need various computing power, training time, and so on. It is hard to pick up the best model immediately. I design a function to train data and test data by different models, including knn, Decision Tree, Random Forest, and Neural Network. By contrast, I find the Random Forest model can acquire a relatively higher accuracy.

**Hyperparameter tuning for best models**

Based on iteration computation, I find the best value of 'n\_estimators' is 114 and max\_depth=9, max\_features=16. Using these parameters, the accuracy can be up to 94% and the Recall ratio goes up to 0.90, which obviously improves the accuracy comparing using default parameters.

**Ensemble modeling**

Ensemble is a good way to increase the accuracy or performance of a model. In simple words, it is the combination of various simple models to create a single powerful model. Ensemble modeling is akin to making a decision using the opinions from multiple experts, which can decrease the bias or limitation of a single model. I prepare multiple different models, including Random Forest, Decision Tree, and DNN, then combine their predictions. Then, I choose a voting classifier to combine the predictions coming from the 3 classifiers. The final accuracy has been improved.