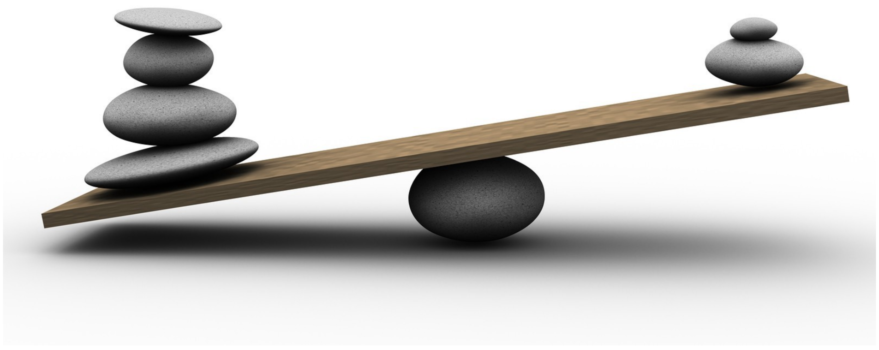
Having an Imbalanced Dataset? Here Is How You Can Fix It.

Different Ways to Handle Imbalanced Datasets.

Will Badr Follow
Feb 22 · 4 min read ★

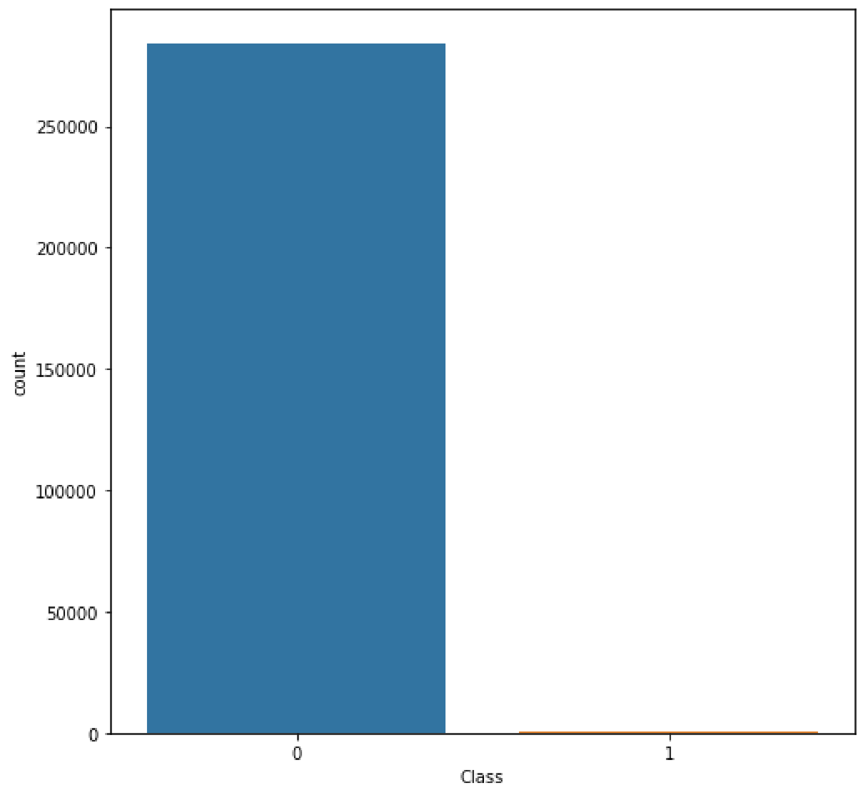


Classification is one of the most common machine learning problems. The best way to approach any classification problem is to start by analyzing and exploring the dataset in what we call **E**xploratory **D**ata **A**nalysis **(EDA)**. The sole purpose of this exercise is to generate as many insights and information out of the data as possible. It is also used to find any problems that might exist in the dataset. One of the common issues found in a dataset that is used for classification is **imbalanced classes** issue.

What Is Data Imbalance?

Data imbalance usually reflects an unequal distribution of classes within a dataset. For example, in a credit card fraud detection dataset, most of the credit card transactions are not fraud and a very few classes are fraud. This leaves us with something like 50:1 ratio between the fraud and non-fraud classes. In this article, I will use the credit card fraud transactions dataset from Kaggle. You can download this dataset from here.

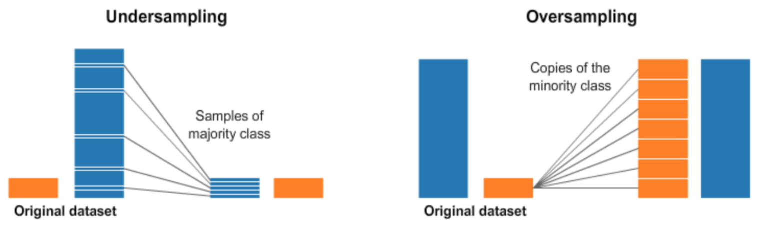
First, let's plot the class distribution to see the imbalance.



As you can see, the non-fraud transactions far outweigh the fraud transactions. If we train a binary classification model without fixing this problem, the model will be completely biased. It also impacts the correlations between features and I will show you how and why later on.

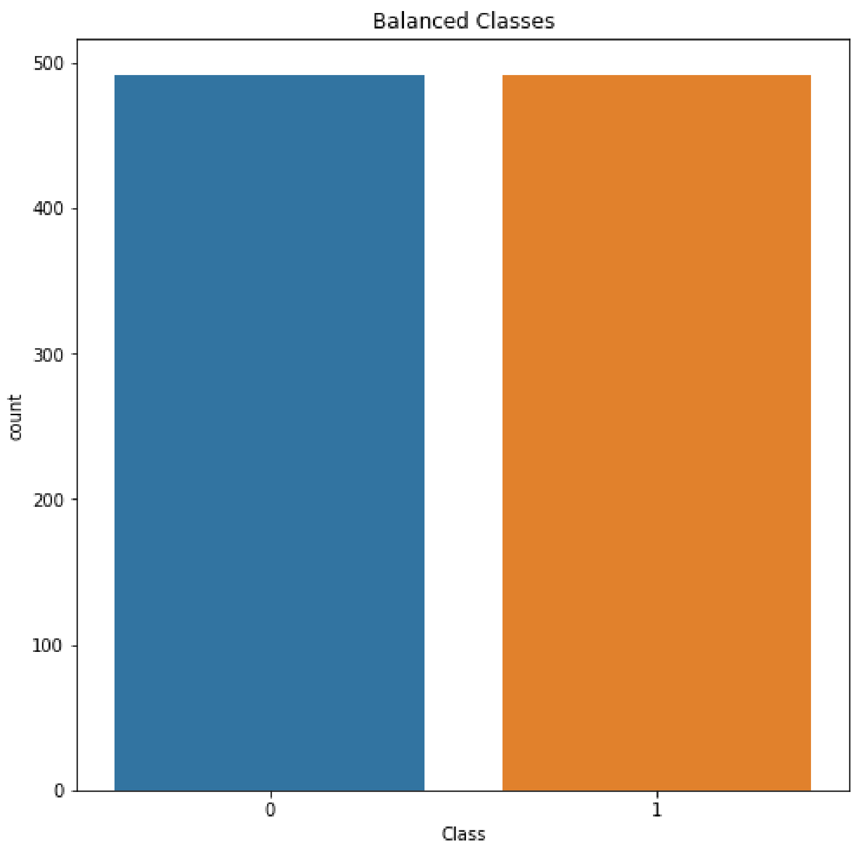
Now, there are a few techniques to solve this problem and I will go through each one of them.

1- Resampling (Oversampling and Undersampling):



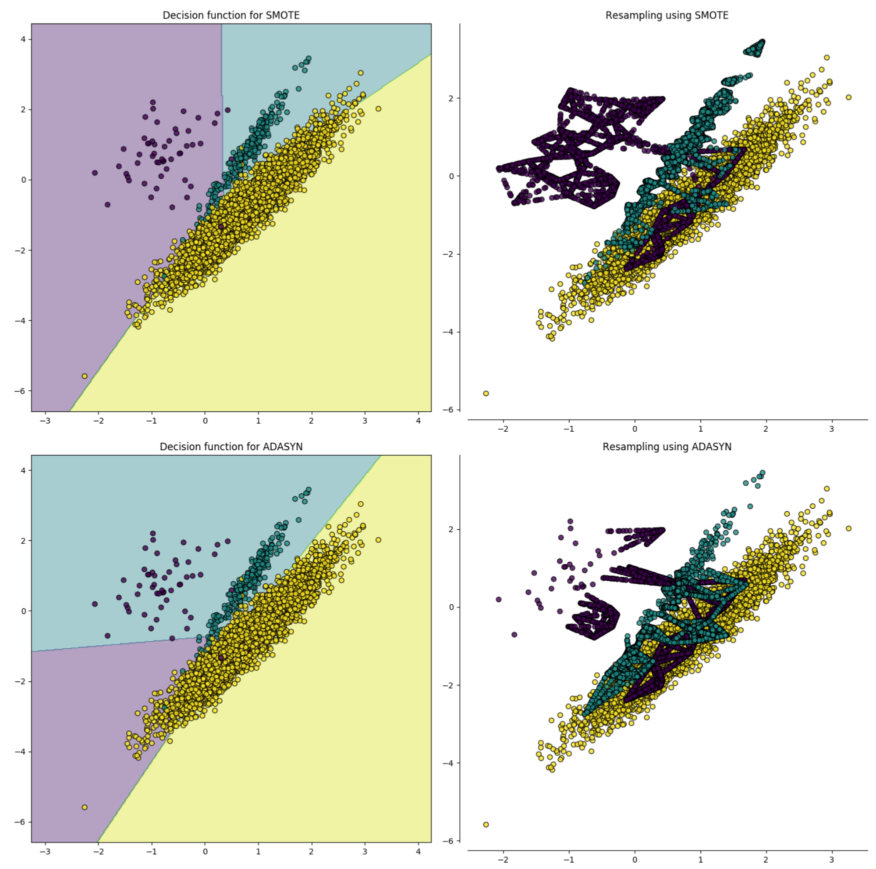
This is as intuitive as it sounds. **Undersampling** is the process where you randomly delete some of the observations from the majority class. An easy way to do that is shown below:

After undersampling the dataset, I plot it again and it shows an equal number of classes:



Balanced Dataset (Undersampling)

The second resampling technique is called, **Oversampling**. This process is a little more complicated than undersampling. It is the process of generating synthetic data that tries to randomly sample the attributes from observations in the minority class. There are a number of methods used to oversample a dataset for a typical classification problem. The most common technique is called **SMOTE (**Synthetic Minority Over-sampling Technique). In simple terms, it looks at the feature space for the minority class data points and considers its *k* nearest neighbours.



Source:https://imbalanced-learn.readthedocs.io/en/stable/over_sampling.html

To code this in python, I use a library called **imbalanced-learn or imblearn.** The code below shows how to implement SMOTE.

SMOTE Oversampling code.

Remember when I said how imbalanced data would affect the feature correlation? Let me show you the correlation before and after treating the imbalanced class.

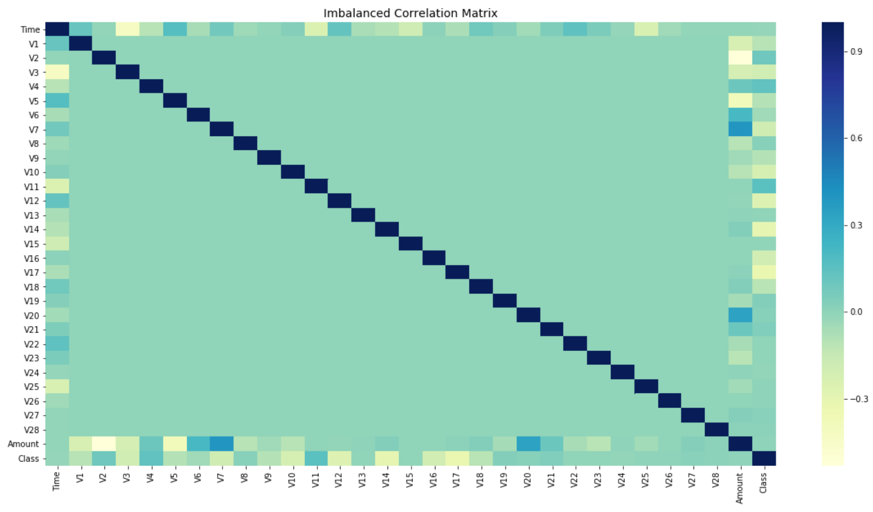
Before Resampling:

The code below plots the correlation matrix between all the features.

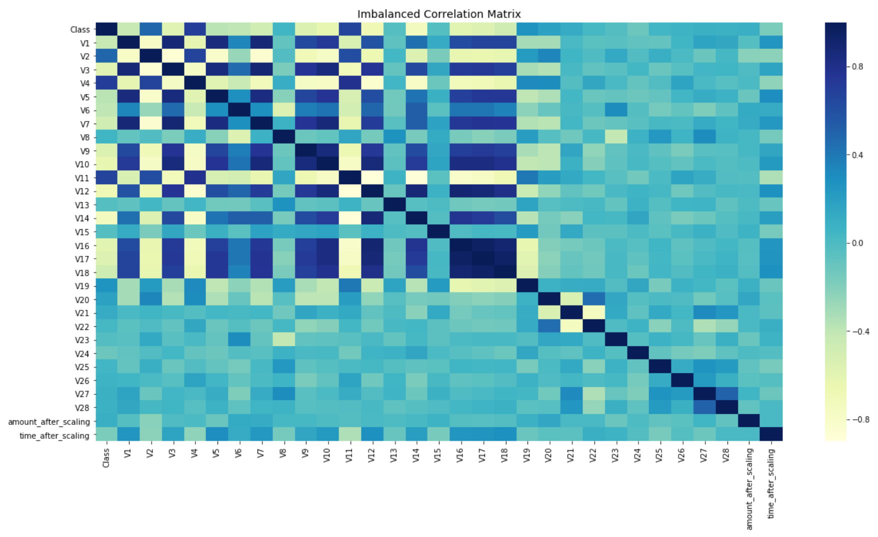
```
# Sample figsize in inches
fig, ax = plt.subplots(figsize=(20,10))

# Imbalanced DataFrame Correlation
corr = credit_df.corr()
sns.heatmap(corr, cmap='YlGnBu', annot_kws={'size':30},
ax=ax)
ax.set_title("Imbalanced Correlation Matrix", fontsize=14)

plt.show()
```



After Resampling:



Notice that the feature correlation is much more obvious now. Before fixing the imbalance problem, most of the features did not show any correlation which would definitely have impacted the performance of the model.

2- Ensembling Methods (Ensemble of Sampler):

When using ensemble classifiers, bagging methods work by building multiple estimators on a different randomly selected subset of data. In scikit-learn, the classifier is named `BaggingClassifier` . However, this classifier does not allow to balance each subset of data. Therefore, when training on imbalanced data set, this classifier will favour the majority classes and create a biased model.

In order to fix this, we can use `BalancedBaggingClassifier` from **imblearn** library. It allows the resampling of each subset of the dataset before training each estimator of the ensemble. Therefore, `BalancedBaggingClassifier` takes the same parameters as the scikit-learn `BaggingClassifier` in addition to two other parameters, `sampling_strategy` and `replacement` which control the behaviour of the random sampler. Here is some code that shows how to do this:

```
from imblearn.ensemble import BalancedBaggingClassifie
from sklearn.tree import DecisionTreeClassifier

#Create an object of the classifier.
bbc = BalancedBaggingClassifier(base_estimator=Decisic
                                sampling_strategy='aut
                                replacement=False,
                                random_state=0)

y_train = credit_df['Class']
X_train = credit_df.drop(['Class'], axis=1, inplace=Fa
```

Train Imbalanced Dataset using Ensembling Samplers

That way, you can train a classifier that will handle the imbalance without having to undersample or oversample manually before training.

In Conclusion, everyone should know that the overall performance of ML models that are trained on the imbalanced dataset, will be constrained by its ability to predict rare and minority points. Identifying and resolving the imbalance of those points is crucial to the quality and performance of the generated models.