Class Imbalance

16 October 2023 13:18

Class imbalance refers to a situation in classification tasks where the classes are not represented equally.

It occurs when the observations in one class are significantly higher or lower in number than the observations in other classes.
This can pose challenges in building robust machine learning models, as the model might become biased toward the majority class, leading to poor performance in predicting the minority class.

Example of imbalanced data

Let's understand this with the help of an example.

Ex: In an utilities fraud detection data set you have the following data:

Total Observations = 1000

Fraudulent Observations = 20

Non Fraudulent Observations = 980

Event Rate= 2 %

The main question faced during data analysis is – How to get a balanced dataset by getting a decent number of samples for these anomalies given the rare occurrence for some them?
Challenges with standard Machine learning techniques

The conventional model evaluation methods do not accurately measure model performance when faced with imbalanced datasets.

Standard classifier algorithms like Decision Tree and Logistic Regression have a bias towards classes which have number of instances. They tend to only predict the majority class data. The features of the minority class are treated as noise and are often ignored. Thus, there is a high probability of misclassification of the minority class as compared to the majority class.

Evaluation of a classification algorithm performance is measured by the Confusion Matrix which contains information about the actual and the predicted class.

Actual	Predicted	
	Positive Class	Negative Class
Positive Class	True Positive(TP)	False Negative (FN)
Negative Class	False Positive (FP)	True Negative (TN)

Accuracy of a model = (TP+TN) / (TP+FN+FP+TN)

However, while working in an imbalanced domain accuracy is not an appropriate measure to evaluate model performance**.** For eg: A classifier which achieves an accuracy of 98 % with an event rate of 2 % is not accurate, if it classifies all instances as the majority class. And eliminates the 2 % minority class observations as noise.

Examples of imbalanced data

Thus, to sum it up, while trying to resolve specific business challenges with imbalanced data sets, the classifiers produced by standard machine learning algorithms might not give accurate results. Apart from fraudulent transactions, other examples of a common business problem with imbalanced dataset are:

- Datasets to identify customer churn where a vast majority of customers will continue using the service. Specifically, Telecommunication companies where Churn Rate is lower than 2 %.
- Data sets to identify rare diseases in medical diagnostics etc.
- Natural Disaster like Earthquakes

Dataset used

In this article, we will illustrate the various techniques to train a model to perform well against highly imbalanced datasets. And accurately predict rare events using the following fraud detection dataset:

Total Observations = 1000

Fraudulent Observations =20

Non-Fraudulent Observations = 980

Event Rate= 2 %

Fraud Indicator = 0 for Non-Fraud Instances

Fraud Indicator = 1 for Fraud

Approach to handling Imbalanced Data

2.1 Data Level approach: Resampling Techniques

Dealing with imbalanced datasets entails strategies such as improving classification algorithms or balancing classes in the training data (data preprocessing) before providing the data as input to the machine learning algorithm. The later technique is preferred as it has wider application.

The main objective of balancing classes is to either increasing the frequency of the minority class or decreasing the frequency of the majority class. This is done in order to obtain approximately the same number of instances for both the classes. Let us look at a few resampling techniques:

2.1.1 Random Under-Sampling

Random Undersampling aims to balance class distribution by randomly eliminating majority class

examples. This is done until the majority and minority class instances are balanced out.

Total Observations = 1000

Fraudulent Observations =20

Non Fraudulent Observations = 980

Event Rate= 2 %

In this case we are taking 10 % samples without replacement from Non Fraud instances. And combining them with Fraud instances.

Non Fraudulent Observations after random under sampling = 10 % of 980 =98

Total Observations after combining them with Fraudulent observations = 20+98=118

Event Rate for the new dataset after under sampling = 20/118 = 17%

- **Advantages**
- It can help improve run time and storage problems by reducing the number of training data samples when the training data set is huge.
- **Disadvantages**
- It can discard potentially useful information which could be important for building rule classifiers.
- The sample chosen by random under sampling may be a biased sample. And it will not be an accurate representative of the population. Thereby, resulting in inaccurate results with the actual test data set.

2.1.2 Random Over-Sampling

Over-Sampling increases the number of instances in the minority class by randomly replicating them in order to present a higher representation of the minority class in the sample.

Total Observations = 1000

Fraudulent Observations =20

Non Fraudulent Observations = 980

Event Rate= 2 %

In this case we are replicating 20 fraud observations 20 times.

Non Fraudulent Observations =980

Fraudulent Observations after replicating the minority class observations= 400

Total Observations in the new data set after oversampling=1380

Event Rate for the new data set after under sampling= 400/1380 = 29 %

- **Advantages**

- Unlike under sampling this method leads to no information loss.
- Outperforms under sampling
- **Disadvantages**
- It increases the likelihood of overfitting since it replicates the minority class events.

2.1.3 Cluster-Based Over Sampling

In this case, the K-means clustering algorithm is independently applied to minority and majority class instances. This is to identify clusters in the dataset. Subsequently, each cluster is oversampled such that all clusters of the same class have an equal number of instances and all classes have the same size.

Total Observations = 1000

Fraudulent Observations =20

Non Fraudulent Observations = 980

Event Rate= 2 %

- **Majority Class Clusters**
    1. Cluster 1: 150 Observations
    2. Cluster 2: 120 Observations
    3. Cluster 3: 230 observations
    4. Cluster 4: 200 observations
    5. Cluster 5: 150 observations
    6. Cluster 6: 130 observations
- **Minority Class Clusters**
    1. Cluster 1: 8 Observations
    2. Cluster 2: 12 Observations

After oversampling of each cluster, all clusters of the same class contain the same number of observations.

- **Majority Class Clusters**
    1. Cluster 1: 170 Observations
    2. Cluster 2: 170 Observations
    3. Cluster 3: 170 observations
    4. Cluster 4: 170 observations
    5. Cluster 5: 170 observations

6. Cluster 6: 170 observations

- **Minority Class Clusters**

1. Cluster 1: 250 Observations
2. Cluster 2: 250 Observations

Event Rate post cluster based oversampling sampling = 500/ (1020+500) = 33 %

- **Advantages**
- This clustering technique helps overcome the challenge between class imbalance. Where the number of examples representing positive class differs from the number of examples representing a negative class.
- Also, overcome challenges within class imbalance, where a class is composed of different sub clusters. And each sub cluster does not contain the same number of examples.
- **Disadvantages**
- The main drawback of this algorithm, like most oversampling techniques is the possibility of over-fitting the training data.

Other Common strategies to handle class imbalance:

Synthetic Data Generation: Synthetic data can be generated for the minority class to balance the dataset, using techniques like SMOTE (Synthetic Minority Over-sampling Technique).

Ensemble Methods: Using ensemble techniques such as bagging and boosting that combine multiple models can help improve the prediction performance on imbalanced datasets.

Algorithm Selection: Choosing appropriate algorithms that are less sensitive to class imbalance, such as decision trees, random forests, or support vector machines, can lead to better results compared to algorithms that are sensitive to class distribution, such as simple logistic regression.