
Handling Of Imbalanced Data

A Project Report submitted by

Pankaj Yadav

in partial fulfillment of the requirements for the award of the degree of

Master of Science

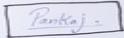


**Indian Institute of Technology Jodhpur
Department of Mathematics
19 June 2021**

Declaration

I hereby declare that the work presented in this Project Report titled Handling Of Imbalanced Data– M.Sc. submitted to the Indian Institute of Technology Jodhpur in partial fulfillment of the requirements for the award of the degree of M.Sc., is a bonafide record of the research work carried out under the supervision of Dr. Vivek Vijay. The contents of this Project Report in full or in parts, have not been submitted to, and will not be submitted by me too, any other Institute or University in India or abroad for the award of any degree or diploma.

Pankaj Yadav
M19MA011



Certificate

This is to certify that the Project Report titled Handling Of Imbalanced Data, submitted by Pankaj Yadav (M19MA011) to the Indian Institute of Technology Jodhpur for the award of the degree of M.Sc., is a bonafide record of the research work done by him under my supervision. To the best of my knowledge, the contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Vivek Vijay

Contents

1. Introduction

1.1 Literature Survey

2. Imbalanced Data Sets

2.1 Imbalanced Classe Distribution

2.2 Imbalanced Degree

2.3 Application Domains

3. Motivation

4. The Nature Of Imbalanced Classes

4.1 Small Sample Size

4.2 Class Separability

4.3 Within class Sub-Clusters

5. Handling Imbalanced Data

5.1 Under-Sampling Methods

5.1.1 Random Under-Sampling

5.1.2 Tomek Links

5.1.3 NearMiss

5.2 Over-Sampling Methods

5.2.1 Random Over Sampling

5.2.2 SMOTE

5.2.3 ADASYN

5.3 Combining Over and Under Sampling

6. Ensembled Methods

7. Demonstration I

8. Measures Of Performance

8.1 Accuracy

8.2 Precision and Recall

8.3 F-Measure

8.4 Support

8.5 Confusion Matrix

8.6 Geometric Mean

8.7 Dominance

8.8 ROC

8.8.1 AUC

9. Demonstration II

9.1 Class Weights

10.Bibliography

1 Introduction

Here we will discuss if we have a data set, then whether it is an imbalanced data set or not. If the given data set is imbalanced then some sampling methods how to balance that data set. Then in future,we will see some performance measures to check which machine learning algorithm is actually effective and give us more optimized results.

1.1 Literature Survey

Classification of data using imbalanced class distribution has experienced a significant disadvantage of the performance attainable by most standard classifier learning algorithms which are relatively based on balanced class distribution and equal misclassification costs. This report supplies a review of the classification of imbalanced data concerning: the application domains and the nature of the problem^[1].

By the continuous research on expansion of data availability in many large-scale and complex, networked systems, such as finance,surveillance, security, it becomes very important to advance the basic understanding of knowledge discovery and analysis by raw data to support decision-making methods. Using existing discovery, knowledge and data techniques have shown utmost success in most of the real-world applications, the problem caused by imbalanced data is a respectively new challenge that has increase attraction in growing attention from both industry and academia. The study of imbalanced problem is related with the performance of learning algorithms in the availability of underrepresented data and most class distribution skews. By existing complex characteristics of imbalanced data sets, learning from such data needs new principles and new understandings, algorithms, and some of tools to transform wide amounts of raw data efficiently into knowledge and information^[2].

The T-Link algorithm used in the pre processing phase as a method of data cleaning so as to remove noise. Combine T-Link with other sampling method such as Random Under-Sampling, Random Over-Sampling and Synthetic Minority Technique (SMOTE) so as to have a balanced class distribution. Classification then used using several ML algorithms such as Random Forest and Logistic Regression^[3].

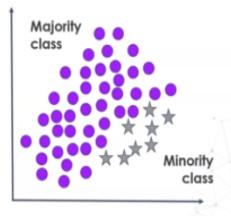
Here we will discuss if we have a data set, then whether it is an imbalanced data set or not. If the given data set is imbalanced then some sampling methods how to balance that data set. Then in future,we will see some performance measures to check which machine learning algorithm is actually effective and give us more optimized results.

2 Imbalanced Data Sets

Imbalanced data sets.

What are they? How often we find it? What is the nature of them ? why they may cause problem? All these questions we'll discuss in this section.

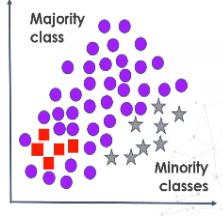
Imbalanced data sets have many more instances of certain classes than others .



Here are some nomenclature used in this report:

2.1 Imbalanced Class Distribution^[1]

In class distribution, the proportion of instances belonging to each class.
Imbalanced data sets can have one or more minority classes.



2.2 Imbalanced Degree^[1]

An imbalanced Degree is the ratio of the sample size of the minority class to that of the majority class.

There is not a said rule that describes from this ratio onward we are going to call this data set as imbalanced data sets.

Typically, imbalanced ratios are 1:10 and smaller.

2.3 Application Domains^[1]

The availability of imbalanced data sets in many real-world applications has ignited a huge amount of research.

In certain applications, the correct classification of samples in minority class has a larger value than the majority.

Here are some applications^[1] from real-world :

- Medical Diagnosis
- Fraud Detection
- Network Intrusion Detection
- Equipment Manufacturing and Testing
- Detection of Oil Spills from Radar Images

3 Motivation

The Problem With an imbalanced Data Sets is that mostly Machine Learning (ML) Algorithm assumes balanced distributions.

As the minority examples occur rarely, rules to predict the small classes are difficult to find.

Samples from the minority class are most often misclassified.

When we work with imbalanced Data Set,we particularly interested in predicting the minority class.

4 The Nature of Imbalanced Classes^[1]

$$\text{Imbalanced Ratio} = \frac{X(\text{minority})}{X(\text{majority})}$$

In some cases, a ratio as low as 1:40 can be hard for building the best model, while in other cases, 1:5 is tough to deal with.

Here are some factors that influence the ability of a classifier so that it identifies the rare events.

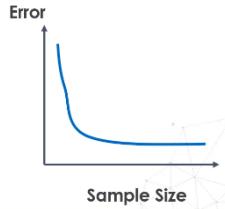
4.1 Small Sample Size^[1]

Small size plays a crucial role in determining the "goodness" of a model.^[1]

If the sample size is limited, finding patterns inherent to the small class is hard.

As the size increases, the error in the prediction decreases.

Having imbalanced classes may not be a problem if the data is big enough.

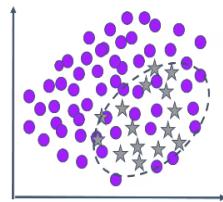


4.2 Class Separability^[1]

If patterns among classes overlap, it is harder to find rules.

The class imbalance itself maybe not a problem, instead the separability makes it harder to find rules to classify correctly the minority class.

Also, linearly separable domains are not sensitive to any amount of imbalance. i.e if 2 classes are really well separable then imbalance is not a problem.



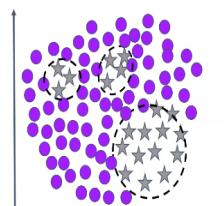
4.3 Within Class Sub-Clusters^[1]

As in many classification problems, a single class is composed of various sub-clusters.

These sub-clusters don't always contain the same number of examples.

This phenomenon is referred to as within class imbalance, corresponding to the imbalanced class distribution among sub-classes.

Within Class, sub-clusters increases the complexity and hence makes it harder to find boundaries to separate the class.



5 Handling Imbalanced Data

5.1 Under Sampling Methods

It is a process of reducing the number of samples from the Majority class.
How much should we reduce the data set and when to stop?

Balancing Ratio

$$R = \frac{X(\text{Minority})}{X(\text{Majority})}$$

if $x(\text{minority}) = x(\text{majority})$, $R(x) = 1$

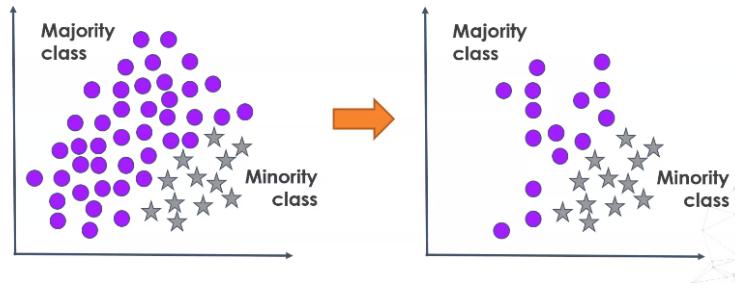
if $x(\text{majority}) = 2*x(\text{minority})$, $R(x) = 0.5$

5.1.1 Random Under Sampling

Random Under Sampling extract observation at random from the given majority class, until a certain balancing ratio is reached.

This is a naive technique.

It just extracts observations at random without making any major assumptions on the characteristic of the data.



Why this method doesn't work :

In process of random undersampling, it may end up removing the important information which is needed for the calculation of the performance of the model. Also, it is harder for algorithms to learn patterns to differentiate the classes.

5.1.2 Tomek Links

Tomek link (T- Link) : Tomek Link is a under-sampling method. This method is considered as related to the Nearest Neighbour Rule (NNR).

K Nearest Neighbour Rule:

The k-nearest-neighbor classifier depends on the Euclidean distance between the identified training samples and a test sample. Let y_i be an input sample with p features $(y_{i1}, y_{i2}, \dots, y_{ip})$, n is the total number of input samples ($i = 1, 2, \dots, n$) and p be the total number of features where, $(j = 1, 2, \dots, p)$. The Euclidean distance between y_i & y_l samples, where ($l = 1, 2, \dots, n$) is defined as:

$$d(y_i, y_l) = \sqrt{(x_{i1} - x_{l1})^2 + (y_{i2} - y_{l2})^2 + \dots + (y_{ip} - y_{lp})^2}$$

Working of this algorithm is as follows :

Let a be an observation of class A(suppose Majority) and b be an observation of class B(suppose Minority)^[3].

Let $d(a,b)$ be the distance between a and b .

(a,b) is a T - Link ,if for any observation c ,

$$d(a, b) < d(a, c)$$

and $d(a, b) < d(b, c)$.

If any two examples are T-Link then one of these examples is noise or otherwise, both examples are located on the boundary of the classes^[3].

T-Link method is a method under-sampling where the observations from the majority class are removed.^[3]

If two observations in a data set are the nearest neighbor to each other and from a different class, then they are TOMEK LINKS.

This procedure removes the tomek link from the majority class.

This under-sampling procedure belongs to the cleaning type of procedure as it eliminates observations of the boundary.

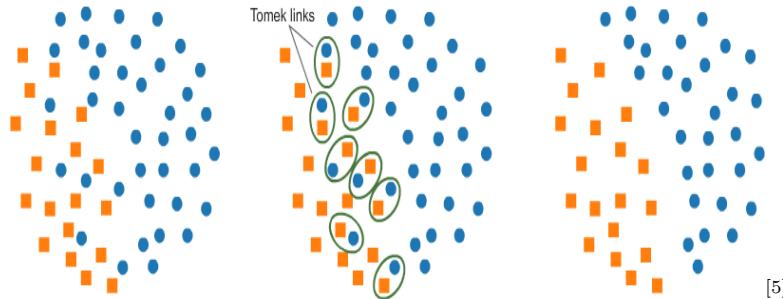
The final shape of the data set varies,

The assumption of this technique is that boundary is noise.

If two observations are very similar and yet they are from different classes then these are not the most important one on which algorithm should focus to learn the parameters to discriminate the classes.

The thing that makes this method useful is that with tomek links removing the noise we are preventing our machine learning algorithms from the cases that are really hard to classify. Therefore by removing noise we will improve the performance of the algorithm.

By not having these cases at the boundary we will tend to miss classify that hard cases.



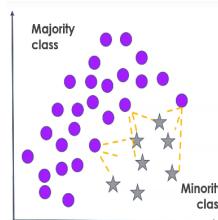
5.1.3 NEARMISS^[2]

NearMiss is a fixed method of Under Sampling so that it returns as many as observations from majority classes as those from the minority classes.

NearMiss approach has 3 versions.

NearMiss, Version 1^[2]

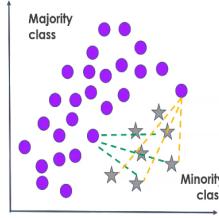
Determine the mean distance between each observation to each k- closest observation from the minority class.



The NearMiss-1 method selects those observations of majority whose average distance to the three closest minority class observations is the smallest.

Near Miss, Version 2^[2]

Determine the mean distance to each k- furthest neighbours from x(min).

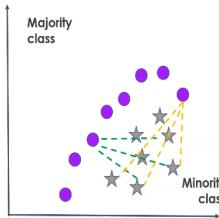


The NearMiss-2 method selects the observations of majority class whose average distance to the three farthest minority class observations is the smallest.

Near Miss, Version 3^[2]

It is a two-step process.

NearMiss-3 chooses a given number of the closest majority observations for each minority observation to guarantee that every minority observation is surrounded by someof majority observations.



All in all, the “most distance” method selects observations from the majority class whose average distance to the three closest minority class oservations is the largest. Experimental results suggested that the NearMiss-2 method can provide competitive results for imbalanced learning ^[2].

5.2 OverSampling Methods

Over-Sampling is the process of increasing the minority class.

Typically, samples are increased until a desired balancing ratio is reached. which in most cases is 1.

Over Sampling methods can be grouped into those methods that extracts sample at random from the minority class in which case all the observations and only method falls in this category is Random Over Sampling,

Alternatively, We can increase the number of samples in the minority class by generating new samples and this method takes the existing samples in the minority class as templates to create new samples. This is called a sample generation. In this case, when we generate new samples, the new observations will be slightly different from the existing one, so we won't be just duplicating the data, but they will be somewhat similar.

Among this method, we find SMOTE and its variant and adasyn.

5.2.1 Random Over-Sampling

Random Over Sampling extracts observations at random from the minority class, until a certain balancing ratio is reached.

It is a naive technique because it just takes a sample at random without making any assumptions on the distributions.

By extracting observations at random from the minority class, with replacement. we are in essence duplicating samples from majority class. So in our final data set we will have plenty of observations that are identical to each other.

This may cause the machine learning model to over fit as it may think all samples are same or alternatively they belong to other class, this is the main criticism of random over-sampling.

To create less duplicated samples, we can decrease the balancing ratio, so that we extract fewer samples from minority class.

5.2.2 SMOTE

SMOTE means Synthetic Minority Over Sampling Technique.

It creates new observations from the minority class by interpolation.

Interpolation is a type of estimation, where we create new data points. With SMOTE, the minority class is "over-sampled" by creating "synthetic examples" instead of extracting data at random.

Hence SMOTE prevents duplication of the data. The new observations from the minority class will not be identical to the original ones.

HOW IT WORKS

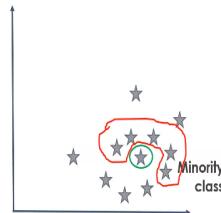
SMOTE will only work with the observations from the minority class. So the first step is to isolate those observations.

K Nearest Neighbour Rule:

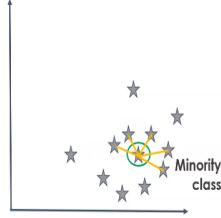
The k-nearest-neighbor classifier depends on the Euclidean distance between the identified training samples and a test sample. Let y_i be an input sample with p features $(y_{i1}, y_{i2}, \dots, y_{ip})$, n is the total number of input samples ($i = 1, 2, \dots, n$) and p be the total number of features where, ($j = 1, 2, \dots, p$). The Euclidean distance between y_i & y_l samples, where ($l = 1, 2, \dots, n$) is defined as:

$$d(y_i, y_l) = \sqrt{(x_{i1} - x_{l1})^2 + (y_{i2} - y_{l2})^2 + \dots + (y_{ip} - y_{lp})^2}$$

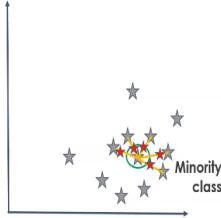
Then it trains a KNN algorithm utilizing all the observations from the minority class, typically k is 5.



SMOTE determines the distance between the original sample (in green circle) and its 5 nearest neighbours as shown in figure.



Then it will interpret new samples along those orange lines that determine between the original sample and it's neighbor to create these artificial samples.



suppose a given training data set T with m examples , we define: $T_{min} = (x_i, y_i), (i = 1; \dots; m)$, where $x_i \in X$ is an observation in the n-dimensional space, $X = (f_1; f_2; \dots; f_n)$, and $y_i \in Y = 1; \dots; S$ is a class identity label related with instance x_i . Typically, $S = 2$ shows the two-class classification problem. ,Hence we can define subsets $T_{min} \subset T$ and $T_{maj} \subset T$, where T_{min} is the set of minority class examples in T, and $T_{min} \cap T_{maj} = \emptyset$ and $T_{min} \cup T_{maj} = S^{[2]}$.

The SMOTE algorithm creates synthetic data by using some resemblance between available minority examples. For subset $T_{min} \in T$, consider the K-nearest neighbors for each example $x_i \in T_{min}$, for some specified integer K; the K-nearest neighbors are described as the K elements of T_{min} whose euclidian distance between itself and $x_i[2]$ under consideration shows the smallest magnitude along the n-dimensions of feature space X.^[2] For creating a synthetic sample, randomly select one of the K-nearest neighbors, then multiply the respective feature vector difference by a random number between [0, 1], and then add this vector to $x_i^{[2]}$

$$x_{new} = x_i + \delta \times (\hat{x}_i - x_i) \quad (1)^{[2]}$$

where $x_i \in T_{min}$ is the minority observation under consideration, \hat{x}_i is one of the K-nearest neighbors for x_i : $\hat{x}_i \in T \in min$, and [0, 1] is a random number. Hence, the final synthetic observation according is a point along the line segment joining x_i under consideration and the randomly selected K-nearest neighbor $\hat{x}_i^{[3]}$.

5.2.3 ADASYN

ADASYN is another method to create synthetic data from the minority class.

It uses a weighted distribution of the minority class according to how difficult the observations are to be learned or classified.

More synthetic data is generated from samples that are harder to classify.

Here are few differences between SMOTE and ADASYN:

SMOTE : It uses all samples from minority class to create the synthetic data.

ADASYN : It uses more samples that are harder to classify and less, that are easy to classify to create the synthetic data.

SMOTE : Uses only minority class to train the KNN.

ADASYN : uses all samples to train the KNN.

SMOTE : Interpolates between samples of minority class.

ADASYN : Interpolates between sample of minority class and neighbour from minority or majority class.

How it works ?

Step 1 : Determine the Balancing Ratio.

$$R(X) = \frac{X(\text{Minority})}{X(\text{Majority})}$$

Step 2 :

Determine the number of samples to generate :

$$G = (X(\text{Majority}) - X(\text{Minority})) * \text{factor.}$$

If $X(\text{maj}) = 900$ and $X(\text{min})= 100$; $G=800$

Factor is 1 for full balancing or Balancing Ratio = 1.

Step 3 :

Train KNN usimg entire dataSet(not only minority class as SMOTE).

Find K closet neighbours for each sample of minority class.

Determine the weighting r: $r = \frac{D}{k}$

D = Neighbours from the majority class.

K = neighbours

Step 4:

Normalise r: $r(\text{norm}) = \frac{r}{\text{sum}(rs)}$

Remember that there are 1r value per observations of the minority class.

Step 5: Calculate the number of synthetic samples that needs to be generate for each observations of the minority class.

$g(i) = r(i)*G$

$r(i)$ = Weight of observation i

G = Number of samples to generate

Step 6: For each minority class example $X(i)$, generate $g(i)$ synthetic samples.

$g(i)$ = Number of samples to generate from observation i

New Sample = Minority Sample - factor*(Minority Sample - Neighbour)

The neighbour can be from the majority or minority class.

The KNN is trained on the entire dataset.

Different from SMOTE.

5.3 Combining Over and Under Sampling

Over Sampling and Under Sampling have some pro's and con's:

Among the positive things of oversampling is that we increase the number of observations from minority class which is indeed the problem we have in dataset.

However, when we amplify these minority class observations be it at random or be at generating new samples in many example we could amplify noise.

Under-sampling :

we may loose important information when we undersample the majority class.

On the other hand side some under sampling techniques remove noisy observations.

can we combine the things of over sampling and under sampling methods to create even better data sets ?

yes!, we could take the best of both the worlds if we combine the right oversampling and undersampling techniques.

We could use an oversampling method to produce more observations from the minority class, which is the class of an under represented data set and we can then use undersampling method to remove those noisy observations.

Therefore ,limiting the impact of this oversampling technique on the noise and as well limiting the impact of undersampling technique on the loss of information from the majority class.

So by combining oversample and undersample method we can retain all of the majority class, increasing the observation from minority and remove the noisy observations that we could have created in the process.

Which to Combine ?

we could combine an oversample method that creates new samples from all of the observations from the minority class and then we can use undersampling technique to remove noisy observations therefore removing

the noisy samples that were created in oversampling.

SMOTE + Tomek Links = Over+Under sampled datasets

SMOTE + ENN = Over+Under sampled datasets

6 Ensembled Methods For Imbalanced Data

Objective: Improve Model Performance.

Idea: Combining several classifiers whose combination outperforms every individual counterpart.

How: Several classifiers are built from the data, and their decision combined or aggregated.

To classify a new example, the observation is submitted to all the classifiers; and the prediction of each of the classifiers are considered to make the final decision.

Briefly : In ensembled method we construct multiple classifiers from the original data and then we aggregate their prediction.

The problem of using ensemble with Machine Learning algorithms is that these classifiers tends to optimize their accuracy. So then by combining classifiers that are optimizing the accuracy , we are not predicting actually the data imbalance issue.

So in order to make ensemble work well with imbalanced data set, what has been done is to combine the use of classifier through **Bagging** and **Boosting** with either data level approaches or cost sensitive approaches.

The **limitation** for ensemble approaches with imbalanced data sets is that many of the methods have been described and not yet implemented in open source libraries. If you wanted to do any of the methods that have been described by other people we would have to actually code the implementation.

7 Demonstration I

Throughout the report, we will use the data set, which we can upload directly from the Python package imbalanced-learn:

Here I take the data from kdd-cup 2004 which is a Protein Homology Dataset^[4].

Homology modeling haveplays a important central role in determining protein structure in the structural genomics project.^[5]

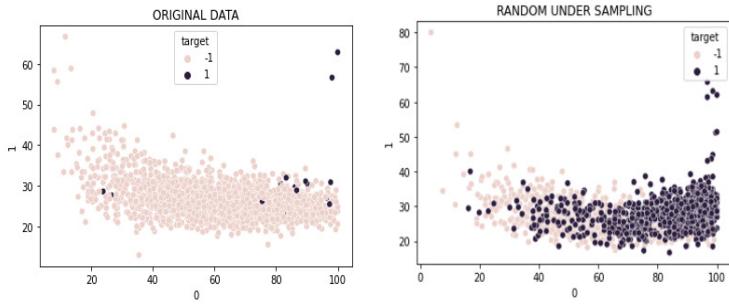
The Data set is described as:

target: whether a protein is homologous to a target protein variables: 74. The features describe the match (e.g. the score of a sequence alignment) between the native protein sequence and the sequence that is tested for homology.

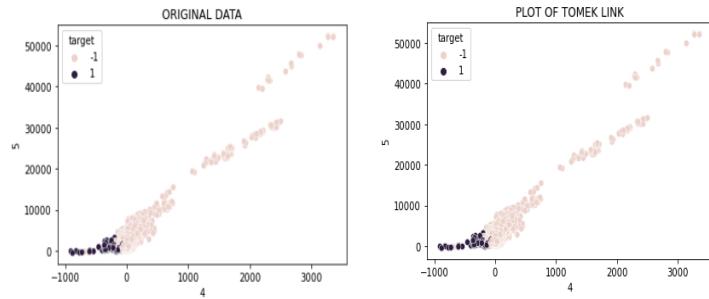
I used python 3 and get these plots using this data
my data set when I loaded look like this^[4]

	0	1	2	3	4	5	6	7	8	9	...	65	66	67	68	69	70	71	72	73	target
0	52.0	32.69	0.30	2.5	20.0	1256.8	-0.89	0.33	11.0	-55.0	...	1595.1	-1.64	2.83	-2.0	-50.0	445.2	-0.35	0.26	0.76	-1
1	58.0	33.33	0.00	16.5	9.5	608.1	0.50	0.07	20.5	-52.5	...	762.9	0.29	0.82	-3.0	-35.0	140.3	1.16	0.39	0.73	-1
2	77.0	27.27	-0.91	6.0	58.5	1623.6	-1.40	0.02	-6.5	-48.0	...	1491.8	0.32	-1.29	0.0	-34.0	658.2	-0.76	0.26	0.24	-1
3	41.0	27.91	-0.35	3.0	46.0	1921.6	-1.36	-0.47	-32.0	-51.5	...	2047.7	-0.98	1.53	0.0	-49.0	554.2	-0.83	0.39	0.73	-1
4	50.0	28.00	-1.32	-9.0	12.0	464.8	0.88	0.19	8.0	-51.5	...	479.5	0.68	-0.59	2.0	-36.0	-6.9	2.02	0.14	-0.23	-1

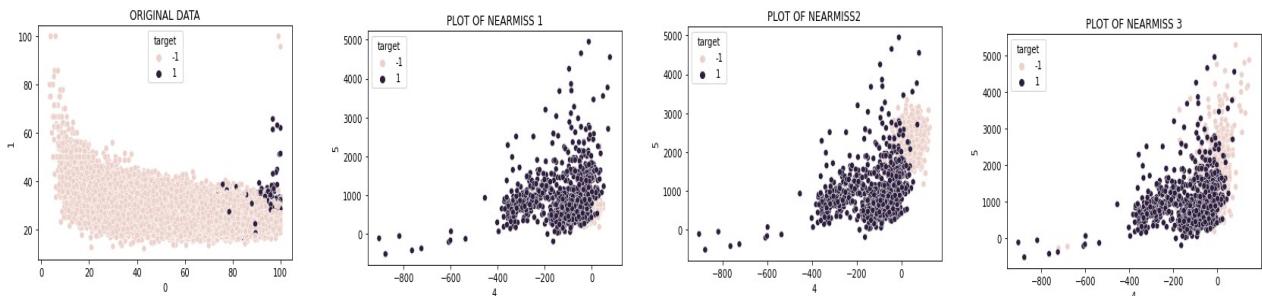
Plots generated using under sampling methods



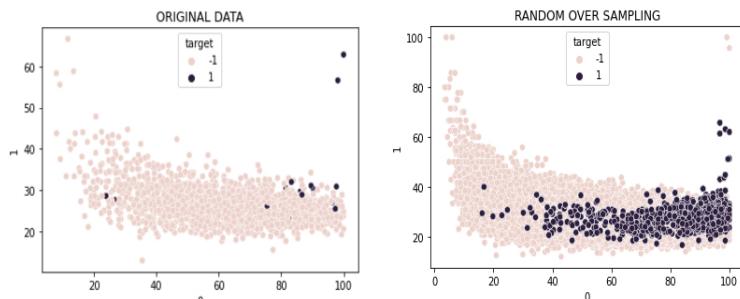
Plots generated using TOMEK links



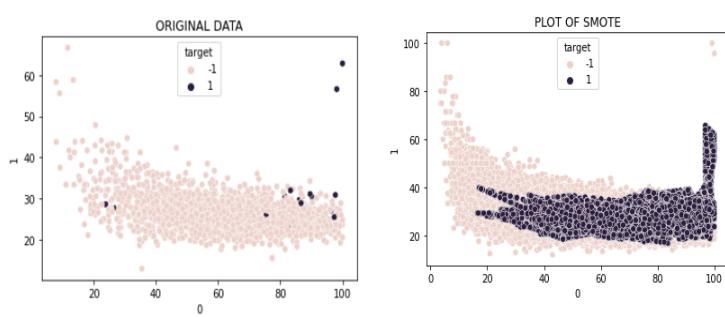
Plot generated using NearMiss



Plot generated using Random Over Sampling



Plot generated using SMOTE



8 Measures of Performance

8.1 ACCURACY

Accuracy represents the percentage or fraction of correct predictions or Fraction of the predictions that the model got right.

accuracy = number of correct predictions/Total number of predictions

For Binary classification

Accuracy = (True Positive + True Negative)/Total number of predictions

For imbalanced data set ,accuracy is not an appropriate metric since it does not distinguish between the number's of correctly classified examples of different classes.

The minority class has very little impact on the accuracy as compared to that of the majority class.

Hence Accuracy is not a clear view of the performance of an algorithm. So we can use another group of matrix to check the accuracy / performance of algorithm.

8.2 PRECISION AND RECALL

As the accuracy is not a good indicator of model performance for imbalanced data-sets because it doesn't gives a proper view of how the model is doing in each of classes of our data-sets.

TRUE POSITIVE RATE(TPR) or (RECALL or SENSITIVITY)

RECALL = TP/(TP+FN)

Positive Predictive Value / PRECISION

PP value = TP/(TP + FP)

We are usually interested in predictive minority class in most of our cases in imbalanced data sets.

As Recall indicates the total number of positive samples that were correctly identified as positive by the model .Therefore an increase in Recall decrease the probability of miss classifying a sample from minority class.

Hence we want to have a high Recall rate.

Precision on the other hand indicates the total samples identified as positive by the model and how many of them are really positive . Therefore an increase in Precision is a reduction in number of false positive

Both Precision and Recall vary between 0 and 1.

To select and tune machine learning models, our goal is to maximize both Precision and Recall.

Both Precision and Recall depends on a probability threshold.(i.e the probability value above which we consider the sample belongs to minority class)

8.3 F - Measure

The F-Measure is a weighted harmonic mean of Precision and Recall.

F-Measure = $2 * \text{Precision} * \text{Recall} / (\text{Precision} + \text{Recall})$

The value of F- Measure varies between 0 and 1.

The best score is 1.0 and the worst is 0.0 .

Optimizing the F - Measure produces the best balance between Precision and Recall.

DISCRIMINATION THRESHOLD

Probability above which we classify a sample as positive.

The optimal threshold is that at which F-Measure is highest.

8.4 Support

Support is another key element to measure the performance.

Support is the number of actual occurrences of the class in the specified data-set.

It indicates the structural weakness in the reported scores and highlights the imbalanced data sets.

Support does not change between models but instead diagnoses the evaluation process.

8.5 Confusion Matrix

In a two class problem, the results of the correctly and incorrectly classified samples are recorded in a confusion matrix.

	predictive negative	predictive positive
Actual Negative	TN	FP
Actual Positive	FN	TP

TN(True Negative) : Number of negative samples correctly classified.

FP(False Positive) : Number of negative samples incorrectly classified as positive.

FN(False Negative) : Number of positive samples incorrectly classified as negative.

TP(True Positive) : Number of positive samples correctly classified.

$$\text{False Positive Rate(FPR)} = \text{FP}/(\text{FP}+\text{TN})$$

$$\text{False Negative Rate(FNR)} = \text{FN}/(\text{TP}+\text{FN})$$

Both FPR and FNR vary between 0 and 1.

Our goal is to minimize both FPR and FNR.

Minimise FNR :

Example : We want to minimize the number of sick people that we do not diagnose correctly.

Minimize FPR :

Example : We want to minimize the number of drugs that we think could be beneficial, but they are not.

The confusion matrix, FPR and FNR depend on a probability threshold.

8.6 Geometric Mean

The G - Mean tries to maximize the accuracy on each of the classes while keeping TNR(Majority), Recall(Minority) accuracies balanced.

It's best value is 1 and the worst value is 0.

$$G\text{-MEAN} = \sqrt{\overline{(TP/TP + TN)} * \overline{(TN/TN + FP)}} = \sqrt{Sensitivity(Recall) * Specificity(TNR)}$$

8.7 Dominance

It is the difference between the accuracy's in Minority and Majority.

Dominance = TPR - TNR = Recall - TNR

Dominance ranges from -1 to 1 .

The value of +1 indicates perfect accuracy on the Minority class , but all cases of majority class are miss-classified.

A value of -1 corresponds to the opposite situation.

8.8 Reciving Operating Characteristic (ROC)

The ROC curve plots the benefits (TP rate) and costs (FP rate) at different classification thresholds.

Every point on the ROC Curve represents a Probability Threshold the model performance trade-off.

ROC evaluates how well a classifier can separate positive and negative examples and helps to identify the best threshold to separate them.

8.8.1 Area Under The ROC(AUC) :

AUC is the area under the Roc Curve.

AUC provides an aggregate measure of performance across all possible classification threshold.

Higher the AUC indicates the model is better at predicting both classes.

For Perfect model: AUC = 1

For Random Model: AUC = 0.5

9 Demonstration II

9.1 Class weights

In the process of doing undersampling and oversampling , there are some limitations or disadvantages of these methods. Like for Undersampling , we lost important information related to our data set.

For oversampling , the process is computationally expensive and algorithmically more complex.

Hence we used class weights hyper parameters to directly balance our data by giving suitable weights to our minority class data so that our both classes in proportion.

In continuation to protein homology data. Here are some results based on class weights and our model showed improvement using class weights.

This is how Data set looks like.

Out[2]:	0	1	2	3	4	5	6	7	8	9	...	65	66	67	68	69	70	71	72	73	target
0	52.0	32.69	0.30	2.5	20.0	1256.8	-0.89	0.33	11.0	-55.0	...	1595.1	-1.64	2.83	-2.0	-50.0	4452	-0.35	0.26	0.76	-1
1	58.0	33.33	0.00	16.5	9.5	608.1	0.50	0.07	20.5	-52.5	...	762.9	0.29	0.82	-3.0	-35.0	140.3	1.16	0.39	0.73	-1
2	77.0	27.27	-0.91	6.0	58.5	1623.6	-1.40	0.02	-6.5	-48.0	...	1491.8	0.32	-1.29	0.0	-34.0	658.2	-0.76	0.26	0.24	-1
3	41.0	27.91	-0.35	3.0	46.0	1921.6	-1.36	-0.47	-32.0	-51.5	...	2047.7	-0.98	1.53	0.0	-49.0	554.2	-0.83	0.39	0.73	-1
4	50.0	28.00	-1.32	-9.0	12.0	464.8	0.88	0.19	8.0	-51.5	...	479.5	0.68	-0.59	2.0	-36.0	-6.9	2.02	0.14	-0.23	-1

5 rows x 75 columns

We are using XG Boost classifier on Data set.

Demonstration of XG Boost without using class weights.

```
| from xgboost import XGBClassifier
```

```
| xgb = XGBClassifier(seed=42)
```

Performance measures without using class weights.

```
| print(f1_score(y_test,xgb.predict(X_test)))
| print(accuracy_score(y_test,xgb.predict(X_test)))
```

```
0.8688946015424165
0.9978793297018587
```

Not that great, right? We achieve a decent f1 score and a pretty high accuracy (Sounds good, doesn't work!). Let us now see how much of a difference does using scale_pos_weight to balance out the effect of skewed classes has on this.

How XG Boost calculates class weights.

```
| class_weight = int(y_train.value_counts()[-1]/y_train.value_counts()[1])
| class_weight
: 111
```

Using XG Boost with class weights.

```
| xgb = XGBClassifier(scale_pos_weight=class_weight,seed=42)
```

Performance Measures related to XG Boost.

```
print(f1_score(y_test,xgb.predict(X_test)))
print(accuracy_score(y_test,xgb.predict(X_test)))

0.8766788766788767
0.9979001205871346
```

Wow, that worked! We just made quite a leap in the f1 score! That really did do wonders. The accuracy also improved a tad, but we don't really want to read too much into the accuracy because it is a useless metric here.

10 Bibliography

References

- [1] Yanmin Sun, Andrew k.c.wong, Mohamed s. kamel, Classification of imbalanced data: a review, p. (689),(690),(691)
- [2] Haibo He and Edwardo A. Garcia, Learning from Imbalanced Data,p.(1263),(1266),(1267)
- [3] Elhassan AT1, Aljourf M, Al-Mohanna ,Shoukri M, Classification of Imbalance Data using Tomek Link (T-Link) Combined with Random Under-sampling (RUS) as a Data Reduction Method,p.(2)
- [4] DATA SET,
<https://www.kdd.org/kdd-cup/view/kdd-cup-2004/Data>
- [5] Zhexin Xiang, Advances in homology protein structure modeling,p.(1)