Handling Imbalanced Data Set: A Case Study for Binary Class Problems

Richmond Addo Danquah
Advisor: Dr. Beidi Qiang
Dr. Andrew Neath
Dr. Song Foh Chew

Southern Illinois University Edwardsville
Department of Mathematics and Statistics

April 30, 2020

Outline

Introduction

Definition 1: A data set is said to be imbalanced, if sample from one class is in higher number than other.

Definition 2: A binary classification problem has all examples belong to one of two classes.

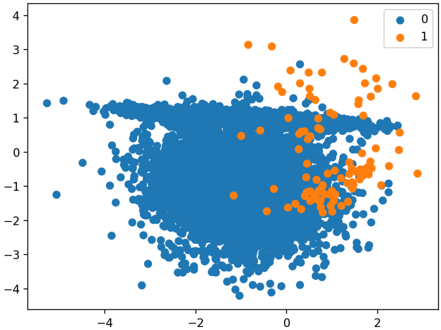


Figure 1: A schematic diagram of a binary imbalanced class distribution

Introduction

- Machine learning classifiers are built on the assumption that there are even class distributions within the data set.
- Classifiers are much more likely to classify new observations to the majority class.
  - Leading to a bias and misleading results.
  - Balancing the data set is necessary to balance the bias in the learning process of the classifiers.
- In this study, we will focus on using synthetic oversampling techniques to handle imbalance data set.

Performance Measures

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figure 2: Confusion Matrix

- Accuracy:

$$\frac{TP + TN}{TP + TN + FP + FN}$$

Performance Measures

- Precision:
$$\frac{TP}{TP + FP}$$
- Recall:
$$\frac{TP}{TP + FN}$$
- F1-Score:
$$\frac{2 * (Recall * Precision)}{Recall + Precision}$$
- AUC:
$$\frac{1 + TPR - FPR}{2}$$

Oversampling Technique

- This technique creates a balanced data set by generating new samples to be added to the minority class.
- Oversampling can be done in two ways:
  - Random oversampling.
  - Synthetic oversampling.
- Our focus will be on handling imbalance data set using synthetic oversampling;SMOTE ADASYN
  - Compared to random oversampling, this method avoids over fitting and improves the generalization ability of classifiers.

SMOTE

- Oversamples the minority class by generating new synthetic examples
- Synthetic examples are generated along the lines joining any/all of the K minority class neighbours
- Forces the decision boundary of the minority class to be more general
- Unlike random oversampling, SMOTE avoids overfitting

How SMOTE works

- Input:
  - Let x_1, x_2,...,x_n be the minority class feature vectors in the n dimensional space of X
  - Let N be the number of synthetic instances to generate
  - Let K be the number of nearest neighbour

SMOTE

- Output:
- For i in range (N) do,
  - Select randomly a minority class feature vector x_i
  - From x_i's K-nearest minority class neighbors, randomly select a neighbor $\hat{x}_i$
  - diff = $\hat{x}_i$ - x_i
  - Let δ =random number between 0 and 1
  - $newSample = \mathrm{x}_i + \mathrm{diff} * \delta$
  - Synthetic $\Longleftarrow$new Sample

end for

SMOTE

Application of SMOTE on a fictitious data

Table 1: Example of class imbalance data set

No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
5	4	5	2	1	3	4	4	5	5
3	3	2	6	4	2.5	3	4.5	5	6

- Yes $\Longrightarrow$Positive class
- No $\Longrightarrow$ Negative class
- Lets generate 2 (N=2) synthetic data points using 2 K-nearest neighbors

SMOTE

- Output:
- For i in range (N=1),
  - Select (4 3)
  - Randomly select a neighbor (5 3)
  - diff = (5 3) - (4 3) = (1 0)
  - $\delta = 0.5$
  - New sample = (4 3) + [(1 0) * 0.5]
  - Synthetic 1 $\Longleftarrow$(4.5 3)

SMOTE

- Output:
- For i in range (N=2) do,
  - Select (5 2)
  - Randomly select a neighbor (5 3)
  - diff = (5 3) - (5 2) = (0 1)
  - New sample = (5 2) + [(0 1) * 0.5]
  - Synthetic 2 $\Longleftarrow$(5 2.5)

SMOTE

Figure 3: A schematic diagram of the class data before and after the application of SMOTE algorithm.

Adaptive Synthetic (ADASYN) Sampling Approach

- An extension of Synthetic Minority Oversampling Technique (SMOTE).
- Reduce bias and adaptively learn
- Synthetic data points are generated based on density distribution.
- More synthetic data are generated for minority class samples that are harder to learn.

ADASYN

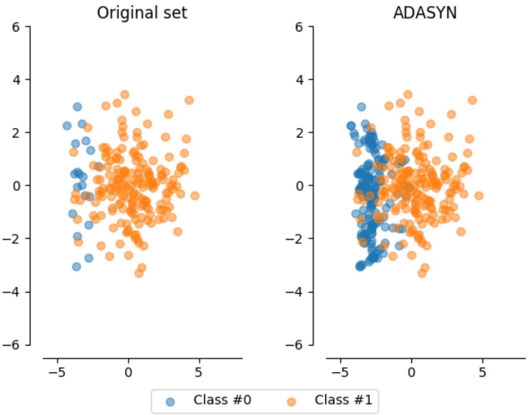


Figure 4: A schematic diagram of a hard to learn class data before and after the application of ADASYN algorithm.

How ADASYN works

- Input:
  - Let n be the number of minority samples
  - Let m be the number of majority samples
  - Let β be the ratio of the balance level of the synthetic samples.
  - Let x_i for i=1,2,3...m be the minority class feature vectors in the n dimensional space of X
  - Let G be the number of synthetic instances to generate
  - Let g_i for i=1,2,3...m be the number of synthetic data generated for each x_i
  - Let K be the number of nearest neighbour
  - Let δ =random number between 0 and 1

How ADASYN works

- Output:
  - $G = \beta \times (n - m)$
  - For i in range (m),
  - Find K for every x_i
  - Calculate $r_i = f_k/\mathrm{K}$, where f_k is the number of feature vectors in the K nearest neighbors belonging to the majority class
  - Calculate $\hat{r}_i = r_i \ / \sum_{i=1}^{m} r_i$, so that $(\sum_{i=1} \hat{r}_i = 1)$
  - Calculate $\mathrm{g}_i = \hat{r}_i \ \times G$

How ADASYN works

- Output:
  - For i in range (g_i) and for j in range (m), do
  - From x_i's K-nearest minority class neighbors, randomly select a neighbor $\hat{x}_{ij}$
  - diff = $\hat{x}_{ij}$ - x_i
  - New Sample$_{ij}$ =$\mathrm{x}_i + diff * \delta$
  - Synthetic $\Longleftarrow$New Sample$_{ij}$

    end for

Input

Table 2: Example of class imbalance data set

No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
5	4	5	2	1	3	4	4	5	5
3	3	2	6	4	2.5	3	4.5	5	6

- Let m = 3, n = 7.
- K = 2.
- $\beta = 0.75$.
- $\delta = 0.5$

ADASYN

- Output:
  - G = (7-3) * 0.75 = 3
  - $r_i = 1/2$ for $i = 1$ to 3
  - $\sum_{i=1}^{3} r_i = 1/2 + 1/2 + 1/2 = 3/2$
  - $\hat{r}_i$= 1/2 * 2/3 = 1/3 for i=1 to 3
  - g_i = $1/3 * 3 = 1$ for $i = 1$ to 3
- From (4 3) 2-nearest minority class neighbors, randomly select a neighbor (5 3)
  - diff = (5 3) - (4 3)
  - New sample$_{ij}$= (4 3) + [(1 0) * 0.5]
  - Synthetic $\Longleftarrow$(4.5 3)

ADASYN

- Output:
- From (5 3) 2-nearest minority class neighbors, randomly select a neighbor (5 2)
  - diff = (5 2) - (5 3)
  - New Sample$_{ij}$ = (5 3) + [(0 -1) * 0.5]
  - Synthetic $\Longleftarrow$(5 2.5)
- From (5 2) 2-nearest minority class neighbors, randomly select a neighbor (4 3)
  - diff = (4 3) - (5 2)
  - New Sample$_{ij}$ = (5 2)+ [(-1 1) * 0.5]
  - Synthetic $\Longleftarrow$(4.5 2.5)

ADASYN

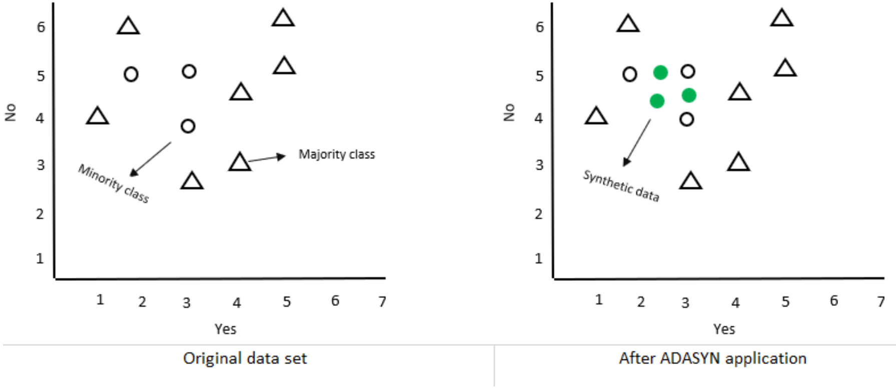


Figure 5: A schematic diagram of the class data before and after the application of ADASYN algorithm.

Experiment

Table 3: Description of data sets

Data Set	Attributes	Sample Size
Blood Transfusion Service Center	5	748
Pima Indians Diabetes	8	768
IBM HR Analytics Employee Attrition	35	1470

Data Set	Minority	Majority
Blood Transfusion Service Center	178	570
Pima Indians Diabetes	268	500
IBM HR Analytics Employee Attrition	237	1233

Blood Transfusion Service Center Data Set

Original Data Set					
Classifier	Accuracy	Precision	Recall	F1	AUC
Logistic	0.73	0.57	0.10	0.20	0.54
SVM	0.74	0.67	0.10	0.20	0.54
Random Forest	0.71	0.45	0.32	0.37	0.59
XGBoost	0.75	0.58	0.37	**0.45**	**0.63**
SMOTE					
Logistic	0.75	0.74	0.79	0.77	0.75
SVM	0.76	0.79	0.74	0.76	0.76
Random Forest	0.80	0.82	0.78	0.80	0.80
XGBoost	0.80	0.81	0.81	**0.81**	**0.80**
ADASYN					
Logistic	0.70	0.75	0.71	0.73	0.70
SVM	0.66	0.72	0.63	0.67	0.66
Random Forest	0.74	0.78	0.75	**0.77**	**0.74**
XGBoost	0.72	0.75	0.74	0.75	0.71

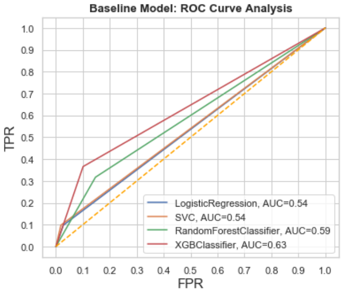
Pima Diabetes Data Set

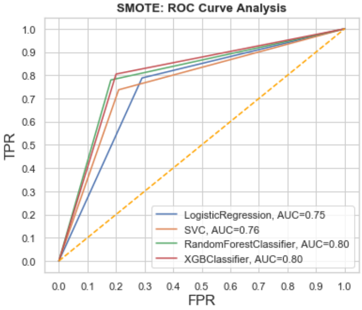
Original Data Set					
Classifier	Accuracy	Precision	Recall	F1	AUC
Logistic	0.82	0.76	0.62	0.68	0.77
SVM	0.79	0.70	0.55	0.62	0.73
Random Forest	0.81	0.71	0.64	0.67	0.76
XGBoost	0.82	0.70	0.70	**0.70**	**0.79**
SMOTE					
Logistic	0.81	0.78	0.82	0.80	0.81
SVM	0.85	0.83	0.87	**0.85**	**0.86**
Random Forest	0.85	0.83	0.87	**0.85**	**0.86**
XGBoost	0.85	0.82	0.88	0.85	0.85
ADASYN					
Logistic	0.72	0.73	0.71	0.72	0.72
SVM	0.82	0.79	0.87	0.83	0.82
Random Forest	0.85	0.82	0.90	**0.86**	**0.85**
XGBoost	0.82	0.78	0.89	0.83	0.82

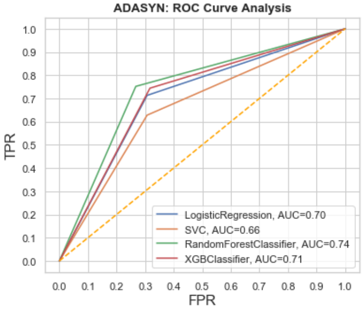
IBM HR Analytics Employee Attrition data set

Original Data Set					
Classifier	Accuracy	Precision	Recall	F1	AUC
Logistic	0.89	0.83	0.41	**0.55**	**0.70**
SVM	0.86	0.90	0.20	0.31	0.59
Random Forest	0.84	0.71	0.10	0.18	0.55
XGBoost	0.87	0.75	0.31	0.43	0.64
SMOTE					
Logistic	0.87	0.90	0.85	0.88	0.87
SVM	0.88	0.93	0.84	0.88	0.88
Random Forest	0.91	0.93	0.90	**0.91**	**0.91**
XGBoost	0.89	0.92	0.86	0.89	0.89
ADASYN					
Logistic	0.87	0.89	0.84	0.86	0.87
SVM	0.89	0.91	0.86	0.88	0.88
Random Forest	0.91	0.92	0.90	**0.91**	**0.91**
XGBoost	0.89	0.91	0.86	0.88	0.88

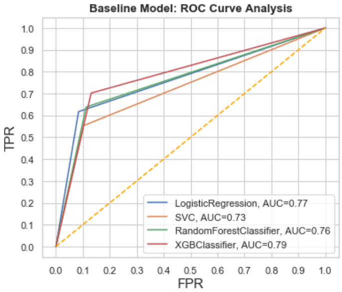
Blood Transfusion Service Center Data Set - AUC score

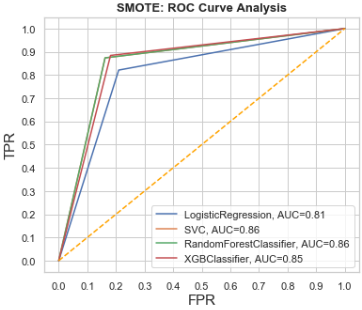


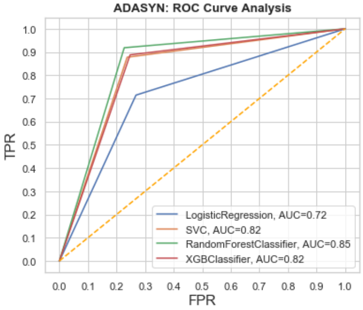




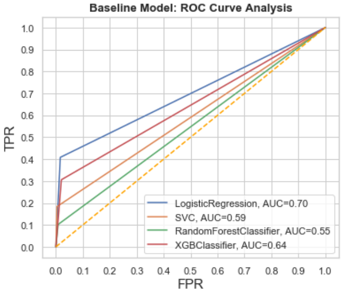
Pima Diabetes data set - AUC score

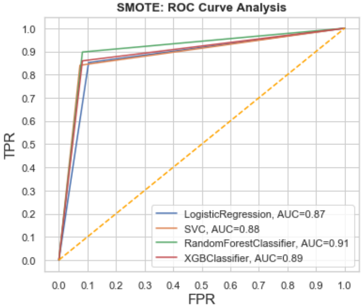


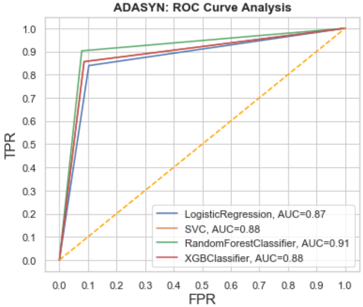




IBM HR Analytics Employee Attrition data set - AUC score







Conclusion

- Results from imbalance data set are often bias and misleading
- Accuracy (common and popular) is not a good performance measure when you have asymmetric data set
- F1-score and AUC score are better performance measures compared to Accuracy especially when you have imbalance data set
- The application of SMOTE and ADASYN improved significantly the classifiers performance measures
- There are not enough evidence to generalize based on this study that, SMOTE performs better than ADASYN in handling class imbalance
- Combination of the sampling technique and the classifier is essential to handling Imbalanced data set in the best possible way