Processing Imbalanced Classes

Different Approaches for: Imbalanced Classes

SL #	Approach	Accuracy	Notes	My Observation
1	Do nothing	60-72 %	Load dataset as is and take % of data at random and split them in test, train and validate. Since the classes are imbalanced, there are good chances that we may lose some key data from minority class.	Not very effective way to classify and predict. Almost all models were limited to low accuracy.
2	Balance the training set in some way: ●Synthesize new minority classes. (SMOTE)	81-86%	Use SMOTE to create synthetic data for minority class which will balance the dataset and then we can take random 20% from the sample and perform the modeling.	Models accuracy increased drastically.
3	Balance the training set in some way: ●Oversample the minority class. ●Undersample the majority class.	86-87%	Undersampling of majority class and oversampling of minority class to make them equal distribution, did increased the overall accuracy, but there are good changes that some of the test data may not represent the same ratio.	
4	At the algorithm level: ● Adjust the class weight (misclassification costs). ● Adjust the decision threshold. ● Modify an existing algorithm to be more sensitive to rare classes.	80-86%	I am using different SVM kernels and XGBoost for class weights as these algorithms can inherently handle imbalanced data.	
5	XGBoost which inherently handles Imbalanced classes	82-85%	XGBoost are inherently able to handle imbalanced datasets, but still we see different kernels were resulting in different accuracy. And XGBoost resulted in accuracy of 85%.	
6	SVM # Linear kernel # Guassian kernel # sigmoid kernel #ploynomial kernel	#67% #71% #72% #76%	SVM are inherently able to handle imbalanced datasets like XGBoost, but still we see different kernels were resulting in different accuracy. And none of SVN resulted in accuracy of 86% or above.	

Some Expected Challenges / Observation

Following algorithms, in conjunction with one hot encoding and 5% of total data complains of all labels of same class.

1. RandomForestClassifier
2. ExtraTreesClassifier
3. GradientBoostingClassifier
4. LogisticRegression

Following algorithms, came up with very similar accuracy and precisions for train, test and validate.

1. DecisionTreeClassifier RandomForestClassifier MLPClassifier
2. AdaBoostClassifier
3. KNeighborsClassifier

There are few approach I would like to try in future

1. Pick up a different dataset with more features
2. Dimensionality reduction so that my test and train dataset have enough class representation
3. Algorithmic Ensemble Techniques
    a. Bagging
    b. Boosting
    c. Adaptive ADA
    d. Gradient Tree