README for Data Sampling

Group 2

02 12 2019

Corresponding R Scripts: Data_Sampling_ROSE; Data_Sampling_Smote.

1. Introduction

In classification problems, an issue to come across can be imbalanced class sizes in the training dataset. Such imbalances can lead to a classifier performing seemingly well when looking solely at model accuracy but on closer inspection being ineffective in classifying instances of the minority class. Because the underlying dataset in this project is heavily skewed towards non-drafted players (approximately 20:1) and therefore contains a majority of zero values for the labels, classification models tend to perform sub-optimally (i.e. predict the majority class much better than the minority class). This is due to the classifier being able to be "lazy" and only classify a player as drafted when absolutely certain, as is visible by the error rates being very close to the proportion of not drafted players in the dataset.

2. General Approach

To deal with the problem of imbalanced class sizes, a number of methods to rework the underlying dataset are commonly applyied, some of which shall be outlined below. In some cases, it might not be viable or necessary to aim for equally sized classes (e.g. when instances of the minority class are inherently easy to classify because they exert highly specific feature values), especially in high dimensional classification problems it is however appropriate to deal with imbalance.

In this project, the imbalance issue can be dealt with by sampling an approximately equal amount of drafted and non drafted players into the training set, either using one of the methods described in the following paragraphs or with a certain pre-made assumption about one of the variables. The latter was applied to the unsampled dataset by disregarding any players that played fewer than ten games (as is done in the basic unsampled dataset), for the main reason that these players do not have sufficient game data to be useful to the prediction and are intuitively less likely to be drafted due to severe lack of experience. To gain an oversight over the different sampling methods available, the approaches outlined below have been taken on top of the qualitative method and the resulting sampled datasets have been used as training data.

3. Random Oversampling for the Minority Class

In order to deal with imbalanced data, random oversampling increases the weight of the minority class, here drafted CFPs (class 1), by randomly replicating the minority class instances. This technique is known to increase the likelihood of occurring overfitting, although random oversampling does not increase information. Since the prevalent class, here class 0, amounts to 2011 observations, to obtain a balanced sample by random oversampling, the new sample size needs to be set to 4022.

3.1 Implementation with ROSE-Package

The Package Random Over-Sampling Examples (ROSE) provides functions to deal with binary classification problems in the presence of imbalanced classes. ROSE contains the function `ovun.sample` that implements traditional remedies to the class imbalance, such as oversampling the minority class (see present chapter), undersampling the majority class (see chapter 4), or a combination of over- and undersampling (see chapter 5). In order to perform the resampling technique `ovun.sample` is endowed with the argument method, which takes one value among `over`, `under` and `both`.

For randomly oversampling the minority class we make use of the argument `over`. This option determines oversampling with replacement from the minority class, here class 1, until a specified sample size `N` is reached. Since the prevalent class, here class 0, amounts to 2011 observations, to obtain a balanced sample by oversampling, we need to set the new sample size to 4022. The function `ovun.sample` generates a list from which we extract the new augmented data; the result is a new balanced data set:

```
CleanClass2007to2013_3_oversampling <- ovun.sample(Drafted~., data=CleanClass2007to2013_3, method="over

CleanClass2007to2013_3_oversampling <- as.data.frame(CleanClass2007to2013_3_oversampling$data)

table(CleanClass2007to2013_3_oversampling$Drafted)

   0    1
2011 2011
```

Alternatively, we could have designed the oversampling by setting argument `p` of the function `ovun.sample`, which represents the probability of the positive class in the new augmented sample. In this case, the proportion of positive examples would be only approximatively equal to the specified `p`.

4. Random Undersampling for the Majority Class

Through random undersampling for the majority class, here class 0, a randomly choosen subset from the class with more instances is selected to match the number of samples coming from each class. This method can potentially lead to loss of information from the left-out samples. On the other hand, if instances of the majority class are near to others, this technique could yield robust results. Since the minority class, here class 1, amounts to 327 observations, to obtain a balanced sample by random undersampling, we randomly pick 327 observations out of the 2011 not drafted cases.

4.1 Implementation with ROSE-Package

For randomly undersampling the majority class with the function `ovun.sample` from ROSE-package we make use of the argument `under`. This option determines simple undersampling without replacement of the majority class, here class 0, until the specified sample size `N` is reached. Since the minority class, here class 1, amounts to 327 observations, to obtain a balanced sample by undersampling, we need to set the new sample size to 654. Again, the function `ovun.sample` generates a list from which we extract the new augmented data; the result is a new balanced data set:

```
CleanClass2007to2013_3_undersampling <- ovun.sample(Drafted~.,data=CleanClass2007to2013_3,method="under

CleanClass2007to2013_3_undersampling <- as.data.frame(CleanClass2007to2013_3_undersampling$data)

table(CleanClass2007to2013_3_undersampling$Drafted)

  0   1
327 327
```

Alternatively, we could have designed the undersampling by setting argument `p`, see explenation above in chapter 3.1.

5. Combination of Over- and Undersampling

Applying a combination of over- and undersampling both the minority class, here class 1, is oversampled with replacement and the majority class, here class 0, is undersampled without replacement. In essence, the minority class is oversampled to reach a size definite as a realization of a binomial random variable

with probability `p`, the probability of the positive class in the new augmented sample, and size `N`. The undersampling for the majority class is then performed correspondingly, to abide by the specified `N`.

5.1 Implementation with ROSE-Package

For the application of a combination of over- and undersampling with the function `ovun.sample` from ROSE-package we make use of the argument `both`. In this case, both the arguments `N` and `p` have to be set to establish the amount of oversampling and undersampling. We set the new sample size `N` to 2338 according to the initial total size of our unsampled data. Argument `p` is set to 0.5, which is the default value. Again, the function `ovun.sample` generates a list from which we extract the new augmented data; the result is a new balanced data set:

```
CleanClass2007to2013_3_both <- ovun.sample(Drafted~., data=CleanClass2007to2013_3,
                    method="both",
                    p=0.5,      # probability of class 1 in new sample; default 0.5.
                    seed=6969, # specify random seed
                    N=2338)    # specified sample according to initial sample size

CleanClass2007to2013_3_both <- as.data.frame(CleanClass2007to2013_3_both$data)

table(CleanClass2007to2013_3_both$Drafted)

   0    1
1150 1188
```

6. Synthetic Minority Oversampling Technique (SMOTE)

SMOTE is a variant of oversampling, following the same basic logic as described in paragraph 3. Differing from "regular" oversampling by effectively replicating minority class instances, SMOTE generates additional synthetic instances similar to the existing ones. This is achieved through placing artificial datapoints between the original ones, while considering a certain number of nearest neighbors as an indication for what such synthetic data could look like. Because class imbalance is not combatted by using more instances of the same minority data, this can have the advantage of reducing overfit on possibly unique feature manifestations of datapoints in the existing set, however, strong bias can be induced especially in models based on the euclidian distance and especially if a high number of nearest neighbors is used.

6.1 Implementation with SMOTE-Package

To perform synthetic minority oversampling and therefore provide a method to combat class imbalance, the `smotefamiliy` package was used. This package contains the `SMOTE()` function which takes the arguments `K` and `dupsize`. The value passed to the `K` argument determines the number of nearest neighbors considered in the synthesis. In the script, the default value of 5 nearest neighbors was used, which leaves room for future exploration of a more optimal value through comparison of bias and variance of especially KNN when using the sampled data. The value passed to the `dupsize` argument determines the factor with which the original minority class is to be scaled. In the script, the fraction of majority to minority instances was used for this argument to achieve roughly equally sized classes. The SMOTE sampling process was performed for every position and every year to account for possible differences within the original data.

The function `SMOTE()` generates a list from which we extract the synthesized data; the result is a new balanced data set:

```
D_smote <- SMOTE(X = select(cleanData_var, -y), target = cleanData_var$y,
                 K = 5,
                 dup_size = length(which(cleanData_var$y == 0))/length(which(cleanData_var$y == 1)))
```