Water Programming: A Collaborative Research Blog

Tips and tricks on programming, evolutionary algorithms, and doing research

FEBRUARY 8, 2021 BY KEYVAN MALEK

How do we deal with extreme events and imbalanced datasets in machine learning?

One of the most popular goals of machine learning is binary classification. Many of the problems with binary classification arise when we need to identify and classify rare events in our datasets. The binary classification of rare events happens frequently in the detection of rare diseases, fraudulent financial activities, and manufactured products. In water system research, we can take the detection of flood risk as an example. We might want to identify flood days from a combination of precipitation, snow coverage, temperature, time of the year, soil moisture, and many other factors.

Training a machine learning model such as this can be very challenging because, in our historical record, we might only have 1% extreme flood days (or even much less), and the rest of the days are non-flood days (normal days). In this blog post, I go over some of the considerations that need to be taken into account when dealing with rare events. I will also discuss some of the techniques that can help us address the issues of imbalanced datasets.

Confusion Matrix

In machine learning, we often use confusion matrices (also known as error matrices) to investigate the performance of classification models. To make this more understandable, I am going to use the example of floods. For example, let's imagine that our goal is to predict whether each day is a flood day or a non-flood day. We will train a machine learning model to identify flood days. In this case, the predicted label of each day can fall into one of the following categories:

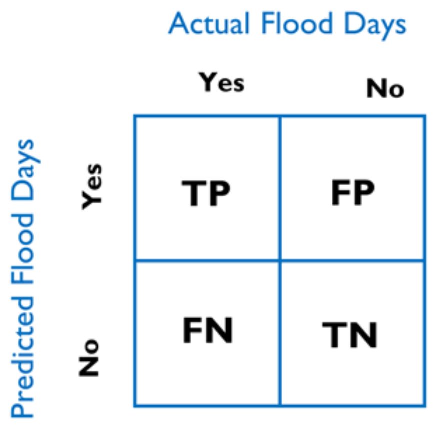
True positive (TP): We correctly classify a flood day as a flood day.

True negative (TN): We correctly classify a non-flood day as a normal day.

False positive (FP): We misclassify a normal day as a flood day.

False negative (FN): we misclassify a flood day as a normal day.

We then count the number of TPs, TNs, FPs, and FNs. After that, we can draw the following table, called a confusion matrix, in order to visualize the outcomes of our binary classification model and use this to better understand the performance of our model.



Accuracy of Prediction

One of the most intuitive ways to perform error calculation is to count how many times our model classification is right and divide it by the total number of events. This is called accuracy and can be calculated from the following equation:

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

Accuracy Paradox

By definition, extreme and rare events are a small portion of our dataset, and using a single accuracy measure as the performance metric can cause significant bias in our model evaluation. I'll provide an example to make this clearer. Let's say that we need to identify a specific fraudulent activity in an online purchase dataset that only happens 0.1% of the time on average; as such, our goal is to create a high-accuracy model to identify these events. Let's assume that we have a model that classifies all activities as normal activities. In this case, the accuracy of our prediction is 99.9%, which looks quite decent. However, this model misclassifies all of the rare activities that we are looking for. Therefore, we have a very bad model with a very high accuracy. This is called the accuracy paradox. To respond to this paradox, different methods and error metrics have been introduced.

Other Performance Indicators

1. True Positive Rate (TPR), which is also known as *sensitivity* or *recall*:

$$TPR = \frac{TP}{TP + FN}$$

2- True Negative Rate (TNR), which is also known as *specificity*:

$$TNR = \frac{TN}{TN + FP}$$

3- False Positive Rate (FPR):

$$TNR = \frac{TN}{TN + FP}$$

4- Positive Predictive Value (PPV):

$$FPR = \frac{TP}{TP + FP}$$

There are many other performance measures that focus on different aspects of model performance, and more information on these binary classification metrics can be found in here (https://en.wikipedia.org/wiki/Evaluation_of_binary_classifiers).

The selection of these metrics depends on the actual problem at hand and the concerns of stakeholders. In some cases, we have to give the priority to reducing false negatives. In the flood example, reducing false negatives might improve the reliability of the system in flood protection. However, in some other situations, reducing false positives can be more important because having incorrect flood alarms can trigger flood safety measures such as reducing the water volume in dams, which can put more pressure on irrigation supply systems.

Therefore, one of the main issues that we need to take into account in classifying rare events is carefully selecting the performance metrics that we use to train our model.

There are other ways to investigate the performance of binary classification models that I am not covering in this blog post, such as developing a Receiver Operating Characteristic (ROC) curve or calculating the area under the ROC curve (AUC). However, more information about these methods can be found at here (https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc#:~:text=An%20ROC%20curve%20(receiver%20operating,False%20Positive%20Rate))and at here (https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5).

Manipulating the Training Dataset

As expected, rare events do not appear in our dataset frequently; however, they can cause important ramifications. When we train our model for an imbalanced dataset that only has 5% TRUE labels, for example, the model tends to learn more about FALSE labels (the majority) and do poorly in identifying TRUE labels. To avoid this, we can modify or resample our training dataset to force the model to focus on the TRUE labels (the minority class). There are various approaches to do that, and I am going to introduce four of these methods here: 1) undersampling; 2) oversampling; 3) SMOTE (Synthetic Minority Over-sampling TEchnique); and 4) increasing the cost of misclassifying the minority class.

Undersampling

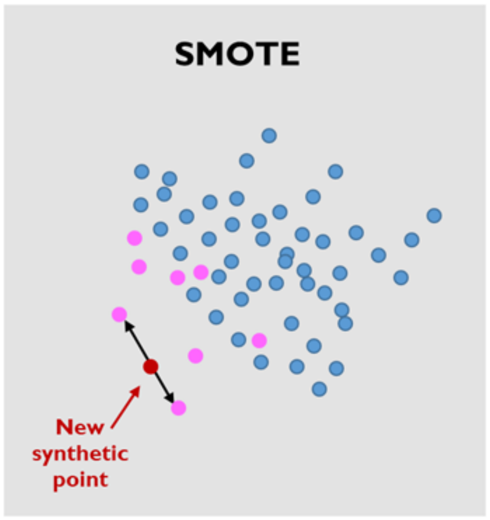
Undersampling reduces the size of the majority class in our training dataset in order to balance the minority and majority classes. There are different resampling methods; for example, we can randomly select from the majority dataset and remove them. However, this method can lead to a loss of useful information and reduce our model's performance. There are other, more intelligent ways that tend to preserve the useful information and focus on removing redundant samples. Condensed Nearest Neighbor Rule (CNN), Tomek Links, and One-Sided Selection are some examples of these methods. You can refer to this blog post (https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification/)for more information about resampling techniques.

Oversampling

Another way of modifying the training dataset is to increase the size of the minority class. The most basic way of doing that is to duplicate the minority class members until the data is not imbalanced anymore.

SMOTE

SMOTE (Nitesh Chawla et al., 2002 (https://arxiv.org/abs/1106.1813)) is an oversampling technique that synthetically generates new samples from the original minority group. Here is how the SMOTE algorithm works. 1) The algorithm first selects a random sample. 2) It finds k nearest neighbors of the selected point and randomly chooses one of them. 3) The algorithm then finds the distance between the two points. 4) It generates a random number between 0 and 1. 5) The algorithm then multiplies the feature vector by that random number to generate a new point that is a reasonable distance from our original minority class. 7) We start over and continue this process until the minority class reaches the desired size. There are some other variants of the basic SMOTE technique, which this blog post (https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/) discusses. We can also use a combination of oversampling and undersampling to achieve a better result.



Penalized Models

Another method to reduce the impact of imbalanced datasets is to increase the cost of misclassifying the minority class. To do that, we can add a new column to our feature list that has a low number for the majority class (e.g., 1) and has much greater values for the minority class (e.g., 100). This forces the model to pay more attention to the minority class in order to reduce the errors during training.

Software Packages for Manipulation of Imbalanced Datasets

Here, I introduce some of the Python and R libraries that have been developed to address the problems that arise when dealing with imbalanced datasets.

Python:

1. **scikit-learn** offers *confusion_matrix*, *roc_curve* and *auc* modules that can be used to generate the confusion matrix.
2. **imbalanced-learn** offers several over- and undersampling techniques as well as combined methods.

R:

1. **pROC** can be used for calculating ROC curve and AUC.
2. **ROSE** performs simple over-, under-, and combined sampling.
3. **DMwR** performs SMOTE.

This entry was posted in Uncategorized. Bookmark the permalink.

2 thoughts on "*How do we deal with extreme events and imbalanced datasets in machine learning?*"

Pingback: Potential Biases and Pitfalls of Machine Learning – Water Programming: A Collaborative Research Blog

Pingback: Potential Biases and Pitfalls of Machine Learning – Hydrogen Water