Assessment 6 : CCCS660 Computational Intelligence

Quentin Lao — Matricule : 261233739

HANDS-ON OPTION

I implemented my own Python code.
The code is provided at https://github.com/soniquentin/CCCS660_McGill_Assignment6.

The notebook is divided in 4 parts :
— Part 1 : Import of libraries and definition of some global variables
— Part 2 : Feature extraction
— Part 3 : Processes the training and testing set, defines of a class `Perceptron`, trains an instance of this class and computes the evaluation metrics on the test set.
— Part 4 : Visualize the decision boundary (XAI [1])

1 Data understanding

The initial phase involved gaining an understanding of the available data. Here are some interesting information we could extract :
— It contains a total of 100,963 samples/posts.
— There are 70 distinct authors
— A large portion of the data is unlabels : 90,132 out of 100,963 samples which represents 89% of the dataset.
— The dataset is very unbalanced : only 23 samples are labeled as "good" and 10,808 as "spam"

2 Data preparation

2.1 Feature extraction

Our intuition, as proposed in the exercise, is that users who post abnormally often and get few up-votes might be undesirable users. Therefore, we added a feature representing the number of posts by each author. We calculate this value by simply counting the number of posts of each unique author in the dataset.

Therefore, each post/sample has two features :
— the number of up-votes
— the author's total number of posts

2.2 Normalization

Perceptrons and more generally a neural network struggle high-value features of varying ranges. A good practice is to normalize the data. We use the standard scaler from the library `scikit-learn` for this purpose.

1. Explainable AI

2.3 Split Training vs Testing set

Due to the severe imbalance in the data, using all labeled samples for training could bias the model. For example, a naive model classifying all posts as spam would have an accuracy of $10808/(10808+23) = 0.998$; but in practice, it did not learn anything. Many solutions exist to address this problem but detailing these is not the focus of this assessment. We simply applied undersampling : we randomly selected a subset of the majority class ; here we chose 23 spams from different authors. The resulting dataset is small but balanced and is composed of 46 samples (23 "spam" and 23 "good"). We divided this dataset into 80% for training and 20% for testing. Moreorver, to evaluate our model on more samples, we included the remaining $10,808 - 30 = 10,778$ spam posts in the testing set.

Following the preprocessing steps, we are able to visualize our training data in a 2D map (one axis per feature) as shown in figure 1 :

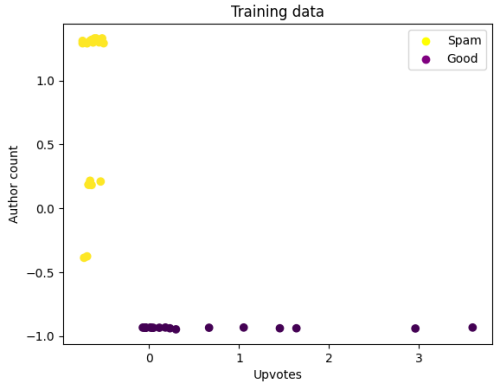

FIGURE 1 – Visualization of our training data

3 Results

3.1 XAI : visualize the decision boundary

The perceptron was trained for one epoch. The visualisation of the decision boundary is illustrated in figure 2 :

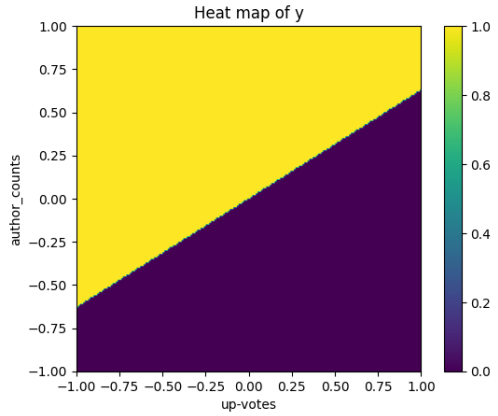

FIGURE 2 – Decision boundary of our model

The decision boundary appears as a simple line. The yellow area corresponds to spams and the purple one

to good posts. This confirms our initial hypothesis : posts receiving few upvotes and coming from authors who post frequently tend to be classified as spam (top left hand corner of figure 2).

3.2 Confusion Matrix

Here is the resulting confusion matrix for the testing set in figure 3 :

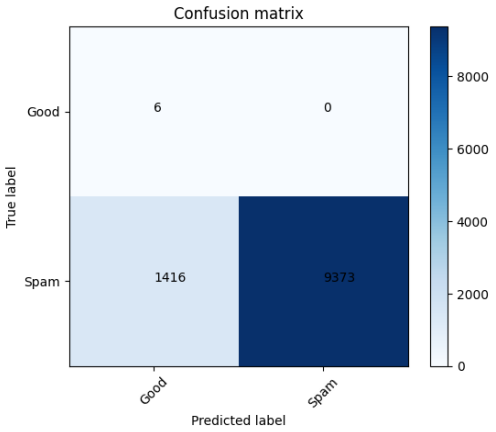

FIGURE 3 – Confusion matrix on the testing set

As observed, our model perfectly identified all the truely "good" posts with no misclassification as spam. This is an excellent point in the context of online discussion boards because it prevents from wrongful identifying legitimate posts and so auto-deletion or auto-banning which could make users upset.

For the truely "spam" posts, a significant majority were correctly identified ($9373/(9373+1416) = 0.867$). However, a substantial portion went undetected. This is probably due to the fact that our model was only trained on a tiny subset of spam posts. The 23 spam posts used for training might not be representative enough and then might not fully capture the diverse characteristics of a spam post.

3.3 Additional performance metric (In-depth stage for the hands-on option)

We compute some additional metrics :
— Accuracy = 86.9%
— Precision = 100%
— Recall = 86.9%
— F1 score = 92.9%

These figures reinforce our previous observations : the model is highly reliable when it identifies a spam (100% of Precision). Yet it does not detect all spam post (Recall of less than 90%). Thus, our model presents pretty good performance. Another advantage is its training duration. It takes only 0.015 seconds to train : this is due to the tiny training set.

4 Further steps

The model is doing quiet well but there are opportunities to enhance it. The first idea is to use data augmentation to rebalance the dataset using oversampling techniques like Synthetic Minority Oversampling TEchnique (SMOTE).

Future improvements could include using semi-supervised learning to increase the labeled data. We would train a preliminary model and use its predictions to label new data and retrain a model on the expanded

dataset.

Additionally, adding more features to better characterize posts could facilitate a more distinct differentiation between good and spam posts.