1

Let's start by defining the Minimum Square Error (MSE) loss function:

$$MSE_{(w)} = \frac{1}{m}\sum_{i-1}^{m}(h_w(x^{(i)}) - y^{(i)})^2 \tag{1}$$

In general, the gradient of an f function is given by the following equation:

$$\nabla f_{(w)} = \begin{pmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \\ ... \\ \frac{\partial f}{\partial w_n} \end{pmatrix}$$

Thus, in our case $\nabla MSE_{(w)}$ can be written as:

$$\nabla MSE_{(w)} = \begin{pmatrix} \frac{\partial}{\partial w_j} MSE_{(w_1)} \\ \frac{\partial}{\partial w_j} MSE_{(w_2)} \\ ... \\ \frac{\partial}{\partial w_j} MSE_{(w_n)} \end{pmatrix} \tag{2}$$

So, we need to compute each partial derivatives of the MSE loss function.
From equation (1) we get:

$$\begin{aligned} \nabla MSE_{(w)} &= \frac{\partial}{\partial w_j}(\frac{1}{m}\sum_{i-1}^{m}(h_w(x^{(i)}) - y^{(i)})^2) \\ &= \frac{1}{m}\frac{\partial}{\partial w_j}(\sum_{i-1}^{m}(h_w(x^{(i)}) - y^{(i)})^2) \end{aligned} \tag{3}$$

For $h_w(x)$ we can denote:

$$\begin{aligned} \frac{\partial}{\partial w_j}(h_w(x)) &= \frac{\partial}{\partial w_j}(w_0 + w_1x_1 + .. + w_nx_n) \\ &= \frac{\partial}{\partial w_j}(w_jx_j) \\ &= x_j \end{aligned} \tag{4}$$

Furthermore for $h_w(x^{(i)})$ we get:

$$h_w(x^{(i)}) = w_0x_0^{(i)} + w_1x_1^{(i)} + ... + w_nx_n^{(i)} = w \cdot x^{(i)} \tag{5}$$

By applying the chain rule on the equations (3),(4) & (5) we get:

$$\frac{\partial}{\partial w_j}MSE_{(w)} = \frac{2}{m}\sum(w \cdot x^{(i)} - y^{(i)})x_j^{(i)} \tag{6}$$

So equation (2) can be written as follows:

$$\nabla MSE_{(w)} = \begin{pmatrix} \frac{2}{m}\sum(w \cdot x^{(i)} - y^{(i)})x_1^{(i)} \\ \frac{2}{m}\sum(w \cdot x^{(i)} - y^{(i)})x_2^{(i)} \\ \dots \\ \frac{2}{m}\sum(w \cdot x^{(i)} - y^{(i)})x_n^{(i)} \end{pmatrix}$$

$$= \frac{2}{m}\sum_{i-1}^{m}(w \cdot x^{(i)} - y^{(i)}) \begin{pmatrix} \sum x_1^{(i)} \\ \sum x_2^{(i)} \\ \dots \\ \sum x_n^{(i)} \end{pmatrix}$$

We observe that the last array is the transpose X matrix, X^T:

$$\nabla MSE_{(w)} = \frac{2}{m}\sum_{i-1}^{m}(w \cdot x^{(i)} - y^{(i)})X^T \tag{7}$$

Lastly, trough matrix notation the above equation can be transformed into:

$$\begin{aligned} \nabla MSE_{(w)} &= \frac{2}{m}(Xw - y)X^T \\ &= \frac{2}{m}X^T(Xw - y) \end{aligned} \tag{8}$$

2

2.1 General

In this task we received a corpus of tweets. The task was to classify the tweets in one of the following three categories; *Neutral, Pro Vax and Anti Vax*, using the **LogisticRegression** classifier.

2.2 Dataset

The training dataset that we were given contained *15976* rows and *3* columns. The first column contained the index of the row and therefore was discarded. The second and third columns contained the tweets and their labels respectively. The distribution of the labels is shown on Fig.1.
In the same notion, the validation dataset contained *2282* rows. The columns were treated the same way as the ones of the train dataset. The distribution of the labels is shown on Fig.2.

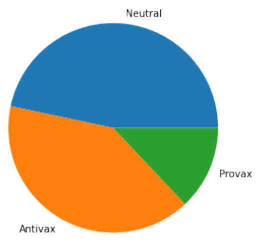

Figure 1: Train Dataset Cardinality

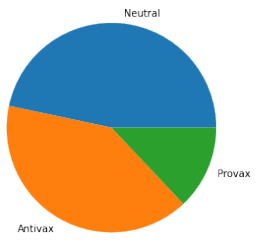

Figure 2: Validation Dataset Cardinality

We observe that the distribution of labels is analogous in both sets, meaning that the cardinality these values will not be an obstacle to the classifier's performance.

2.3 Data Cleaning

Prior to every classification task, text preparation methods need to be performed. In that notion, the following preparation steps were followed:

- Html tags removal
- User mentions removal
- hastags removal
- Text to lower case transformation
- Unicode removal
- Punctuation removal
- Stopwords removal
- Single char removal
- Digits removal
- Multiple spaces removal
- Commoner morphological and inflexional endings removal via *StemPorter*

After 'data cleaning' a sentance that was originally in this form:

"Nothing like killing ourselves w/ our own fear MT@Alyssa_Milano: Vaccine fears have fueled the resurgence of preventable diseases."

is now transformed into:

"nothing like killing fear vaccine fears fueled resurgence preventable diseases"

3 Model Selection

As previously mentioned, in this particular task we had to use the **LogisticRegression** classifier. Since this is a multiclass classification, the parameter *multiclass* is set to *'multinomial'*. In addition, the *max_iter* of the classifier are set to *10000*.

Despite that, various tests were performed in order to determine which type of vectorizer is suitable as well as whether dimensional reduction works on this task.

For this task, two vectorizers were tested; **CountVectorizer** and **TF_IDF**. The parameters selected for both of them are presented bellow:

- Tf_Idf with default parameters
- Tf_Idf with *n_grams* parameter set to (1,2)
- Tf_Idf with *n_grams* parameter set to (1,3)
- Tf_Idf with *n_grams* parameter set to (1,4)
- CountVectorizer with default parameters
- CountVectorizer with *n_grams* parameter set to (1,2)
- CountVectorizer with *n_grams* parameter set to (1,3)
- CountVectorizer with *n_grams* parameter set to (1,4)

For these experiments, a *cross-validation* technique with *5 folds* was performed on the training set. To evaluate the results, *precision, recall and f1 scores* were calculated. Due to the use of 'micro' average as parameter, all of these metrics resulted in identical scores. Thus, only one diagram per experiment is presented to represent the metrics' results.

First, the **TF_IDF** vectorizer was examined. The results of the *5-fold CV* are presented in Fig.3. From the plot, is is clear that **TF_IDF + ngrams_range(1,2)** outperforms the other cases. To check whether the dimensions of the vector need to be reduced, **SVD** -with different number of components- was tested in combination with *TF_IDF + ngrams_range(1,2)*. As we can denote from Fig.4, SVD does not assists this task, consequently should not be used.

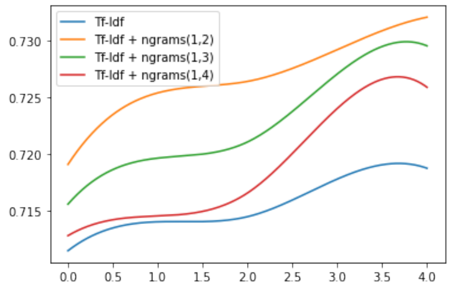

Figure 3: TF-IDF

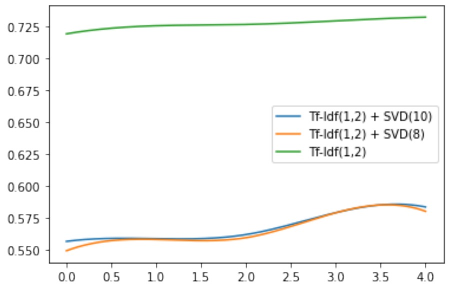

Figure 4: TF-IDF + SVD

The same experiments were performed for the case of **CountVectorizer**. In this case, by looking in Fig.7, we notice that CountVectorizer performs better when *'n_gramms'* are set to a range greater that 1. However, when combined with **SVD** the performance of the model reduces rapidly, as shown in Fig.4.

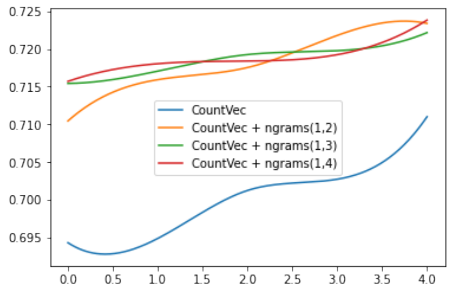

Figure 5: CountVectorizer

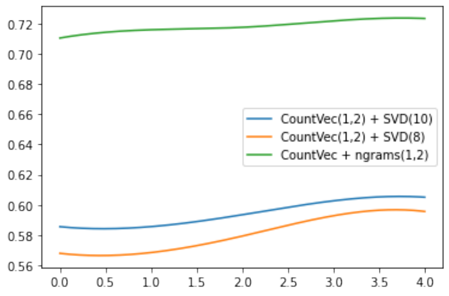

Figure 6: CountVectorizer + SVD

Last but not least, we can conlude that **TF-IDF + n_grams(1,2)**, achieves a better performance than CountVectorizer.

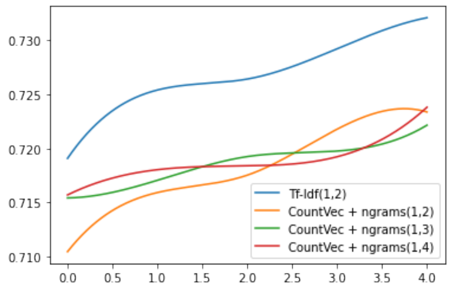

Figure 7: CountVectorizer vs TF-IDF + n_grams(1,2)

4 Results

After applying **TF-IDF** and **LogisticRegression** to our data, the model resulted in **0.729** precision, recall f1 scores. As mentioned earlier, these scores are computed with micro 'average', resulting in to be identical. In order to prove that the model does not overfit or underfit, the learning curve of f1 score is plotted in Fig.8

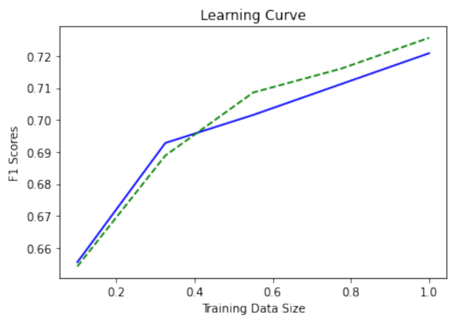

Figure 8: Learning Curve

The blue line denotes the *training score*, while the green one the *validation score*.

To compute the learning curve, a random sample was subtracted from the train set. This sample has equal number of rows with the validation set. The model is then trained on the rest of the instanced that remained on the training set.

5 References

The code section of 'data cleaning' has been previously implemented at the assignment of the course 'Big Data Mining Techniques'.