Twitter Sentiment Classification

Giuseppe Esposito, Syed Ali Abbas
Politecnico di Torino
Student id: s302179, s292423
s302179@studenti.polito.it, s292423@studenti.polito.it

***Abstract*—The goal of this report is to provide a possible approach to solve the classification problem of tweets sentiment. Our work takes in account the case normalization carefully to get a good data preprocessing. The results obtained are quite accurate also before the hyperparameter tuning.**

I. Problem overview

The aim of this work is to solve a problem of supervised machine learning so we have 2 collection of tweets:

- Development set: which contains 225000 comments classified as positive sentiment or negative sentiment.
- Evaluation set: which contains 75000 tweets without label assignment.

Most of our observations will be based on the following table:

Variable	Cardinality
sentiments	2
ids	224716
Time zone	1
Flag	1
User	10647
Text	223106

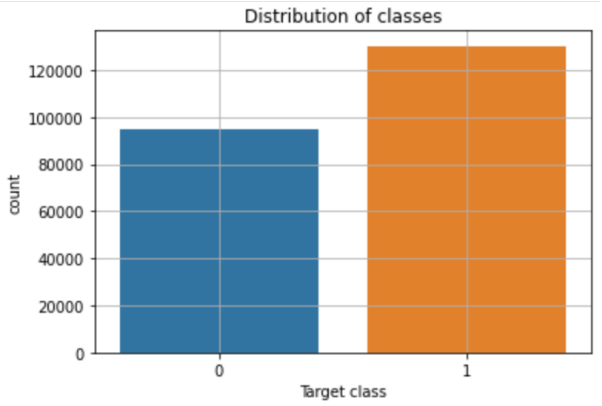
Table:1

Upon a first analysis it is noticeable saying that the analysis that we are carrying out is related to tweets which has no specific topic since the flag field has just one unique string value: ”NOQUERY”. Because of the aim of the analysis we chose **as target the sentiment field** where there are 2 unique binary values:

- 1: positive sentiment
- 0: negative sentiment

Moreover we can notice that it is a West-American community based analysis because in the Date field all the values has the same time zone that is PDT[1] so it is useful to analyze the general distribution of sentiment.

[1]Pacific Daylight Time



Looking at the sentiment distribution we can say that it is quite unbalanced because there are way more positive sentiments than negative one.

Furthermore, it is interesting looking at the correlation matrix. The most noteworthy observation that we can infer is that there exist a dependence between the user names and the sentiment and this means that some people have a specific behaviour which leads them to write tweets with a specific sentiment.

II. Proposed approach

You can use citations as follows: [1] (you can add BibTeX citations in the *bibliography.bib* file).

A. Preprocessing

Now let us have a look to the following table:

Variable	Cardinality
ids	224716
User	10647
Text	223106

Table 2

A noteworthy observation is the comparison between these 3 values, in fact there are more tweets than rows and we expected that the difference between these two cardinalities would have given us the number of duplicate data but then caught up with the idea that different people could have written the same comment so the way to obtain the cardinality of duplicate data was meaningless. Then we thought that, since the ids are uniquely associated to the tweets by definition, the actual number of duplicate data is

given by

$$NumberOfDuplicateData = RowsNumber - IdsNumber \quad (1)$$

As the duplicate data are about 1600 that is, in proportion, a small number with respect to the total number of tweets, we have decided to handle it checking in the column 'ids' the index of the row where the duplicate data appears and then deleting the corresponding row from the dataset. Another column that we had to consider to get useful information is the Date column where the values are a single string contains:

- Day of the week
- Month
- Day of the month
- Time
- Time Zone
- Year

Hence we split the string and we built the dataframe where we have a row for each "id" and where columns are the split-type discarding the value of "Time zone", "Year" (since they are the same for all the rows), "Day of the week" and "Time" , then we added the "user" column to this table. Then we performed a One-Hot Encoder on this table so we got a table of encoded categorical data that we will have been part of the dataset used to train the model. After that we tried to discard the whole "date" field and to keep just the "user" field but eventually we caught up with the idea that the analysis would have been more effective and performing by adding to the "user" column the "day of the month" and the "month" and then we applied a Label Encoder to the "month" values since we can establish an order among them. Moreover this reduced the number of dimension of 2 because we observed that its unique values were only 3: {April, May and June } (In the end this suggested us that the analysis is conducted in a time bracket of 3 months). We did not used the Label Encoder that has been already implemented in Sklearn library because we encountered that the order established by its method was an alphabetical one that did not cope with our problem. Hence, since the field to encode was just one and since the categorical valued to encode was just 3, we decided to implement an hand-made ad-hoc Label Enoder that assignes the values in the column under analysis to the index of the corresponding value in the ordered list of strings. After this we focused our attention on the text column where we have very dirty data and we had to perform some operation of case normalization that consist in:

- Considering just alphabet words only
- Making all the letters in lower case
- Bringing the words back to their stem
- Removing stop words

To clean data we used a powerful library that is "nltk"[2] that provides a pretty wide variety of methods and classes that helped us to deal with natural language processing. We can look for how to extract relevant information from the textual description by means of text mining techniques. The proper solution adopts tf-idf[3] representation limited to the first 110000 words to reduce the dimensionality of the problem.

Figure II-A

Figure II-A shows the words with the highest index of frequency given by:

$$tf - idf(t) = freq(t,d) \times \left(\log \frac{m}{(freq(t,D))}\right) \quad (2)$$

Where t is the term, freq(t,d) is the frequency that the term t has in the current tweet, freq(t,D) is the frequency that the term t has in the whole collection of tweets and m is the number of tweets in the collection.
As soon as we computed the tf-idf we had noticed that the some important words like negations were removed by the default list of stop words of TfidfVectorizer() then we decided that keeping the stop words was more worth than removing all of them. Furthermore we also realised that: setting a number of max feature equal to 110000, most of the irrelevant stop words were removed. Now that everything is set we can concatenate the two DataFrames: the one containing the encoded categorical data and the one containing the values of tf-idf.

B. Model selection

Now we are going to discuss more in dept the models considered for the classification tasks:

- **Random Forest**: an ensemble of decision trees trained with different subsets of variables. Generally it suits with multi-class problems and it is a very powerful model but very expensive in terms of training time.
- **Support Vector Machine**: a classifier that is both used for classification and regression tasks. Basically it looks for an hyper-plane that maximises the distance between itself and objects which belong to both classes [4]. In case we cannot identify an hyper-plane that suits with the two point clouds defined by the 2 classes, the plane becomes a surface and to deal with this it is better to transform the data from its original coordinate space in **y** into a new space

[2]Natural Language ToolKit

[3]Term Frequency - Inverse Document Frequency

[4]so-called margin

C. Hyperparameters tuning

At this point we had to split the DataSet in a training set and in a test set in order to tune our model. We decided to keep the 80% for the training set and the 20% for the test set. In order to find the best combination of training set and test set, we split the dataset by means of KFold method of Sklearn.model_selection (with the "shuffle" argument equal to True) and then iterated over the resulting indices of the dataset to check which one gave us the best score. However we also tried to deal with this in another way: we firstly used the train_test_split making the data order random and then, when we computed the Grid Search, we set the parameter cv = 5 (in order to have the same split in training set, 80%, and test set, 20%, as before) and we realised that this approach gave us the best score. To compute model tuning we took in account different combinations of parameters (which would have made sense to use in order to get a satisfactory baseline score) computing a grid search.

Model	Parameter configuration
Random Forest	"nEstimators": [100,200,300], "maxFeatures":['auto', 'sqrt', 'log2']
LinearSVC	"C":[0.1,1,10] "dual":[True,False] "fitIntercept" :[True,False] "maxIter" : 1000
Decision Tree	"splitter":['best', 'random'], "maxFeatures":['auto', 'sqrt', 'log2']

Table II-C

We repeated this process for each model and we tried the parameters shown in TableII-C.

III. Results

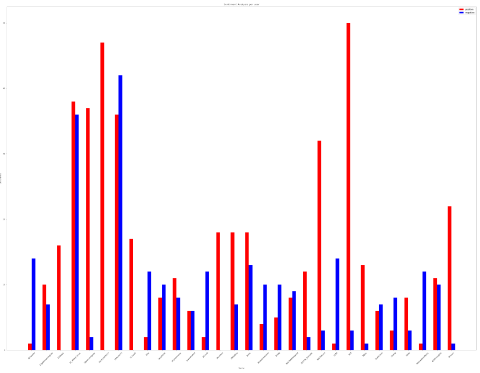
Now we are going to discuss the main results of the analysis.

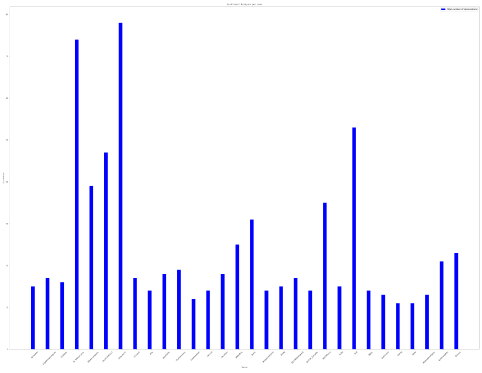
Model	Optimal Param. config.	F1-score
Random Forest	"nEstimators": 300 "maxFeatures": 'sqrt'	F1-score
LinearSVC	"nEstimators": "C":0.1, "dual":True "fitIntercept" :True "maxIter" : 1000	F1-score
Decision Tree	"splitter":['best', 'random'], "maxFeatures": "sqrt"	F1-score

Table II-C

It is noteworthy noticing in Table III that the worst score is preformed by Decision Tree and a reason is embedded in the definition of decision tree algorithm because this model was optimized for classification tasks which provides more than 2 classes.
Another reason should be sought in the number of features because when it is very high that is our case (120649) decision trees are not the best option because the Random Forest outperforms it, since it randomly selects observations and specific features to build multiple decision trees and then each decision tree 'votes' for the best class assignment.
Nevertheless, looking at the Random Forest score we can observe that it is not the best one and the reason of this is the same of the Decision Tree algorithm: it is optimized for classification tasks which provides more than 2 classes.
We can finally encounter that Linear Support Vector Machine reaches the best score and the reason lies in the way it manages sparse matrices, the high dimensionality of the problem and the scalability of the model.

IV. Discussion





Overall it is interesting saying that the differences among the three scores are quite small and this means that the three models works pretty well with the approach proposed.
Another measure that is interesting to take in account is the time expenditure that is hugely higher in Random Forest model building because it is computing a number of trees that is equal to the number of estimators and it takes much time.
We have also thought to a possible future implementation that could make the analysis more accurate so that the score would be improved. The changing on our approach will be the following:

- Analyse the correlation between the sentiment and the user. Since we are comparing a categorical field and a binary one we have shown this correlation in FigureIV
- We can notice that for some users there exist this correlation because some of them have definitely a negative or a positive behaviour. In order to make this assumption more meaningful we could set a threshold on the maximum number of observations as shown in figure IV.
- This means that the accuracy of our model is badly affected by the sentiments, related to the user, that are different from his usual behaviour.
- Then the idea could be: for each user extract the corresponding tweet text which has a different behaviour from the usual one.
- then compute a tf-idf only for these ”strange” tweets (Output = Vector-2)
- check the words with the highest tf-idf value that will be the words that will most negatively affect the accuracy of the model
- In the end compute the tf-idf vectorizer on the whole column of text (Output = Vector-1)
- then remove the words that in Vector-1 have the highest value of tf-idf from the Vector-2 if they are still in there (maybe a possible strategy could be: substituting them to the list of stop words since we have not removed them).
- so that we compute a sort of biased tf-idf vectorizer in order to reduce the dimensionality of the problem and the analysis will be also more meaningful.

REFERENCES

[1] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

can we put also refernces to the slides of the course?