Data Science Task
Oracle Labs

Subject:

Tweet Classification

Done by:

Mohamed BAMOUH

Team Name :

PGX-DS-T42

21 August 2018

Table of Contents

Description:

This is a tweet classification task where the tweets could be either Politics or Sports (i.e., binary classification).

I- Feature Extraction:

1) Pretreatment and data cleaning

The training set offered by the Kaggle competition page is indeed rich and varied, but some modifications and alterations have to be done to cleanse the dataset from useless information, and to prepare it to extract better quality features in optimal conditions.

The steps I have taken to clean the training set data consist of, in this order:

- Removing links from tweets: links are usually unique and will result in poor quality features
- Eliminating punctuation and numbers
- Convert the tweets text to lowercase
- Removing the most common words
- Removing the rare words

I tried Stemming algorithms to merge words from the same root, but the implementation of those algorithms in the dataset ends up making semantical errors and undermine the tweet's features quality.

I also wanted to delete common stop words that don't give much information about the nature of the tweet subject, but it seems that by doing so the quality of the prediction decreases, so I won't rely on that method.

2) Features Extraction

I have decided to use the "Bag of Words" representation, as it is well suited for tweets in general. Indeed, the character limit imposed by Twitter (280 characters) forces its users to write as much significant and condensed info as possible in a few sentences, which means that we can tell the subject of the tweet at first glance, by reading its key words, which don't vary much from tweet to tweet.

This makes, in my opinion, the Bag of Words feature extraction algorithm one of the best for tweet classification.

3) Optimization

In order to maximize the quality of the features extraction process, we can further clean the training set by passing the features to a TF-IDF (Term Frequency times inverse document frequency) algorithm, which gives great results with short but information-heavy texts such as tweets, since getting rid of recurrent words and expressions from the training set will only result in increased performance.

II- Model Training:

1) Machine Learning algorithm

In my search for a suitable Machine Learning algorithm for text classification, I finally opted for the Naive Bayes approach. While simple in its mathematical aspect, it proved itself more performant in my case than other well-known ML algorithms (like SVM – Support Vector Machines). I used the Multinomial Naïve Bayes subclass of this algorithm as it's more suited to text classification.

2) Training set and testing set

This project uses tweets from the DeepTweets competition in Kaggle to fuel, as a training set, a Bag of Words Naive Bayes classifier. The test set is taken from the same source as the testing set, and serves as a performance indicator to measure the accuracy of the predictions resulted from the python script.

III- Performance:

1) Tools used for project implementation

Language: Python, since it's the most commonly used language choice for data science projects.

IDE: Jupyter notebook for easier data manipulation and visualization.

Python Libraries and modules:

- Scikit-learn library, which features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy,
- Re, in order to use regex expressions for string fetching,
- Pandas, for its large choice of functions/procedures specially conceived for data science projects, like .csv files reading and writing, as well as dataset manipulation,

- Csv, for creating and writing inside a csv file the output of the script.

2) Hyper-parameter tuning

As strange as the matter may seem, from my manual experimentations with changing the hyper parameters of the three functions the dataset goes through to extract it’s features and/or train the classifier, it seems that the default values of those function’s parameters are the most optimal for maximum accuracy. Any attempt of slightly changing the parameter’s values from their initial state results in a much less precise prediction.

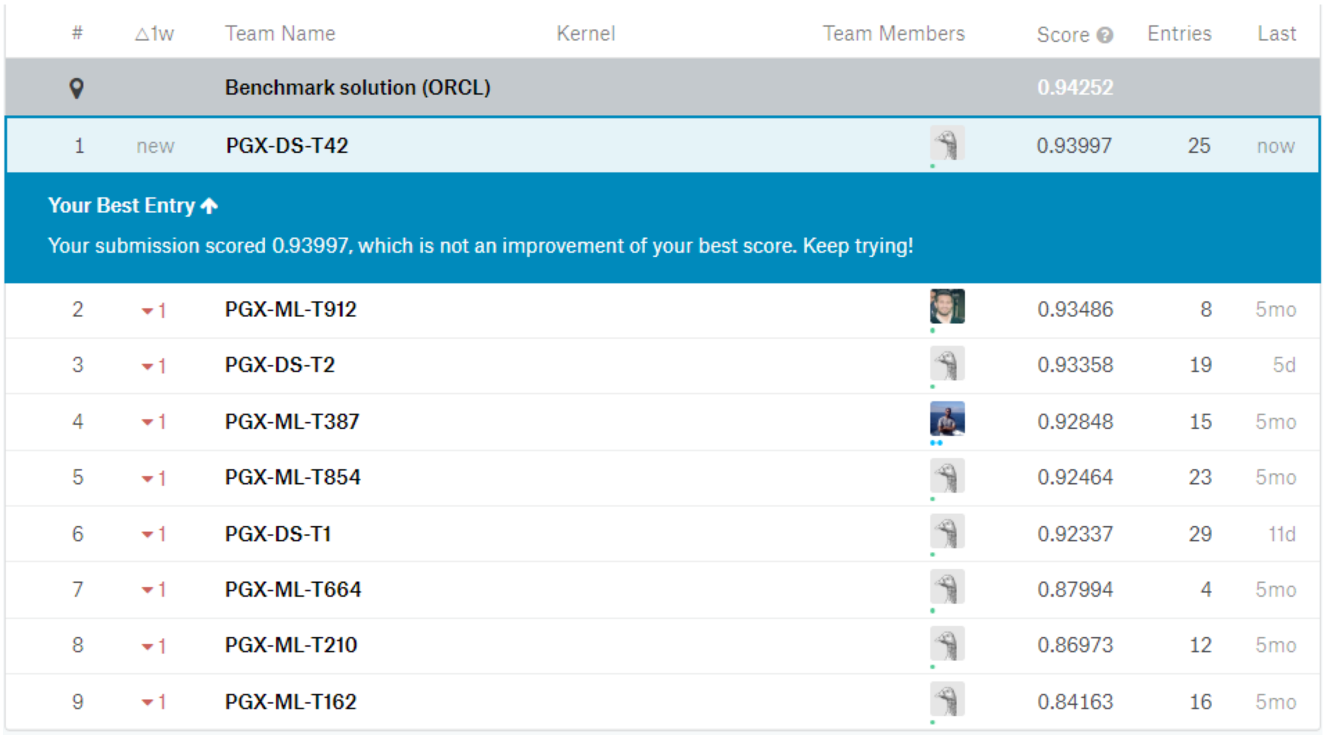
3) Implementation steps

The script follows the following steps (in this order, see code for comments):

1. Library importation,
2. Definition of the functions used in the script,
3. Load training set and testing set,
4. Cleaning both training and testing sets,
5. Apply features extraction algorithm to train set,
6. Apply TF-IDF to training set,
7. Feed the cleaned tweets and their classification to the classifier,
8. Apply features extraction algorithm and TF-IDF to testing set,
9. Predict the classification of the testing set,
10. Display both tweets from the test set and their predicted classification,
11. Create a csv file to store the submission.

4) Output and accuracy

The most recent version of the script scored 0.93997 on the competition's leaderboard on Kaggle.



#	△1w	Team Name	Kernel	Team Members	Score	Entries	Last
		Benchmark solution (ORCL)			0.94252		
1	new	PGX-DS-T42			0.93997	25	now
2	▾1	PGX-ML-T912			0.93486	8	5mo
3	▾1	PGX-DS-T2			0.93358	19	5d
4	▾1	PGX-ML-T387			0.92848	15	5mo
5	▾1	PGX-ML-T854			0.92464	23	5mo
6	▾1	PGX-DS-T1			0.92337	29	11d
7	▾1	PGX-ML-T664			0.87994	4	5mo
8	▾1	PGX-ML-T210			0.86973	12	5mo
9	▾1	PGX-ML-T162			0.84163	16	5mo

Your Best Entry ↑

Your submission scored 0.93997, which is not an improvement of your best score. Keep trying!

V- Conclusion and Perspectives:

Text classification require analytical thinking, as there are many approaches possible to optimize the predictions' accuracy. In my script, I focused mainly on the data cleaning, so if more time was given to me, I would most likely maximize the classification accuracy by involving myself more in the tuning of the feature extractor and the Multinomial NB algorithm's hyper-parameters. Also, I would probably have tested more algorithms for feature extraction and classification, and compare the results of each combination to find the most optimal one.