

# Data mining and machine learning project

DEVELOPMENT OF AN APPLICATION TO ANALYZE REAL-TIME TWEETS  
ABOUT EARTHQUAKES

---

Giacomo Mantovani

Stefano Poleggi

Artificial Intelligence and Machine Learning

2019/2020



# Motivation

---

Social network have entered our lives massively. The idea behind our work is to analyze tweets and to use them as a sort of sensor to detect study and detect their usage in case of natural disaster like earthquakes.

We developed an application for real-time monitoring of earthquake event detection. To do that, we analyzed tweets related to that topic using text analysis techniques; at the end we created a model able to classify tweets.

# Dataset (I)

---

In order to build our classifier raw tweets have been scraped using a twitter scraping tool called Twint written in Python. Given location, tags to search for and a timestamp (or a date interval), returns the informations about the tweets matching those parameters.

To build the training set, two different queries have been used:

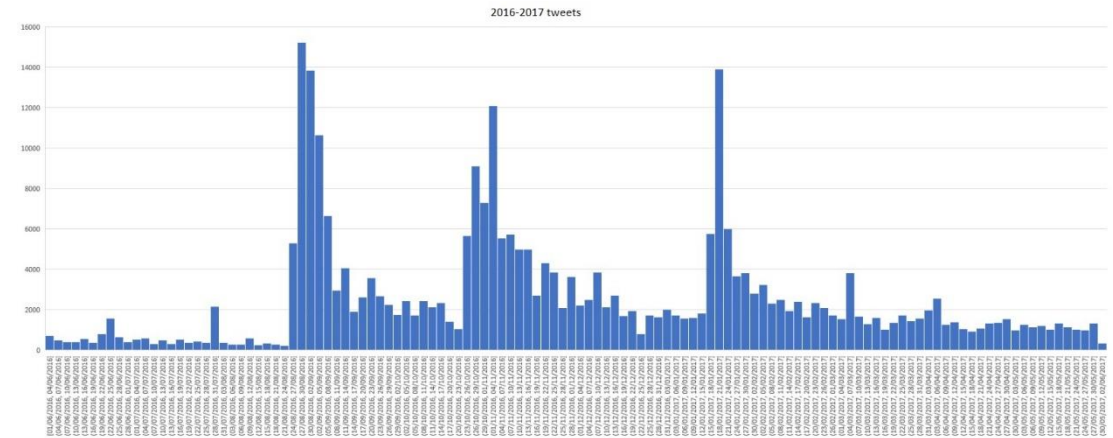
- The first query selects tweets in Italy matching the tag #terremoto
- The second query selects the tweets in Italy without tags

Another database was created by collecting (in the same way) tweets between the dates 2017-06-30 and 2016-06-01.

# Dataset (II)

Once the database about tweets of 2016-2017 were created, we realized an histogram to check if there are spikes of tweets corresponding to catastrophic earthquakes in order to perform an analysis.

As we can see, there 3 major spikes relative to the 24-08-2016, 30-10-2016 and 28-01-2017 huge earthquakes.



# Training set

---

All labels (earthquakes, non-earthquakes) on tweets have been added manually. We ended up having a balanced data set of circa 300 tweets per class, 600 tweets in total.



# Pre-processing

---

All the preprocessing steps were performed using Weka API and java.

After the tweets have been fetched, all of them have been pre-processed to extract only the raw text and remove all meta-information associated. All useless information have been discarded, like hashtags, mentions, links etc.

Finally, all character have been converted to lower case.

A solid orange horizontal bar spanning the width of the slide, located at the bottom.

# Classification model

---

Once the raw tweets have been elaborated during the pre-processing steps, we managed to try different classification models to find the one that provides the best results.

Different classifiers have been applied to our training set, in particular we focused on SMO (Sequential Minimal Optimization), J48 (weka's implementation of the C4.5 algorithm), kNN (k-Nearest Neighbors, we focused on two kNN models, with k equal to 1 and 3) and the NB (the naïve Bayesian classifier, based on the Bayes' s theorem).

# Classification model (II)

---

- SMO

Correctly Classified Instances	575	94.5724 %
Incorrectly Classified Instances	33	5.4276 %
Kappa statistic	0.8914	
Mean absolute error	0.0543	
Root mean squared error	0.233	
Relative absolute error	10.855 %	
Root relative squared error	46.5934 %	
Total Number of Instances	608	

- J48

Correctly Classified Instances	541	88.9803 %
Incorrectly Classified Instances	67	11.0197 %
Kappa statistic	0.7796	
Mean absolute error	0.1389	
Root mean squared error	0.2836	
Relative absolute error	27.7822 %	
Root relative squared error	56.7168 %	
Total Number of Instances	608	



# Classification model (III)

---

- 1NN

Correctly Classified Instances	536	88.1579 %
Incorrectly Classified Instances	72	11.8421 %
Kappa statistic	0.7632	
Mean absolute error	0.1315	
Root mean squared error	0.2965	
Relative absolute error	26.2923 %	
Root relative squared error	59.3034 %	
Total Number of Instances	608	

- 3NN

Correctly Classified Instances	505	83.0592 %
Incorrectly Classified Instances	103	16.9408 %
Kappa statistic	0.6612	
Mean absolute error	0.2008	
Root mean squared error	0.3384	
Relative absolute error	40.1508 %	
Root relative squared error	67.6714 %	
Total Number of Instances	608	

# Classification model (IV)

---

- NB

Correctly Classified Instances	531	87.3355 %
Incorrectly Classified Instances	77	12.6645 %
Kappa statistic	0.7467	
Mean absolute error	0.1326	
Root mean squared error	0.3155	
Relative absolute error	26.5251 %	
Root relative squared error	63.1065 %	
Total Number of Instances	608	

- NB Multinomial

Correctly Classified Instances	548	90.1316 %
Incorrectly Classified Instances	60	9.8684 %
Kappa statistic	0.8026	
Mean absolute error	0.1106	
Root mean squared error	0.2581	
Relative absolute error	22.1205 %	
Root relative squared error	51.6258 %	
Total Number of Instances	608	

# Classification model (V)

---

To evaluate each classification model, we used an n-fold cross-validation (with  $n = 10$ ). The results obtained about accuracy for each classifier are summarized here:

Classifier	Accuracy (%)
SMO	96.22%
J48	88.98%
1NN	88.16%
3NN	83.06%
NB	87.33%
NB Multinomial	90.13%

# SMO Classifier

---

To have a better view about the classifier we studied their confusion matrix and other measures. Here, we showed those results for the SMO classifier.

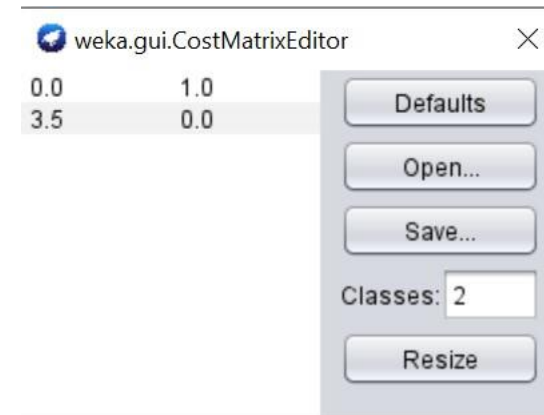
```
=== Confusion Matrix ===
```

				TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	a	b	<-- classified as	0.938	0.046	0.953	0.938	0.945	0.892	0.946	0.925	earthquake
	285	19	a = earthquake	0.954	0.063	0.939	0.954	0.946	0.892	0.946	0.918	non-earthquake
	14	290	b = non-earthquake	Weighted Avg. 0.946	0.054	0.946	0.946	0.946	0.892	0.946	0.922	

# Note about classifier

---

We performed a cost sensitive classification schema, because we wanted to give an higher weight to the “earthquake” class than to “non-earthquake” class.

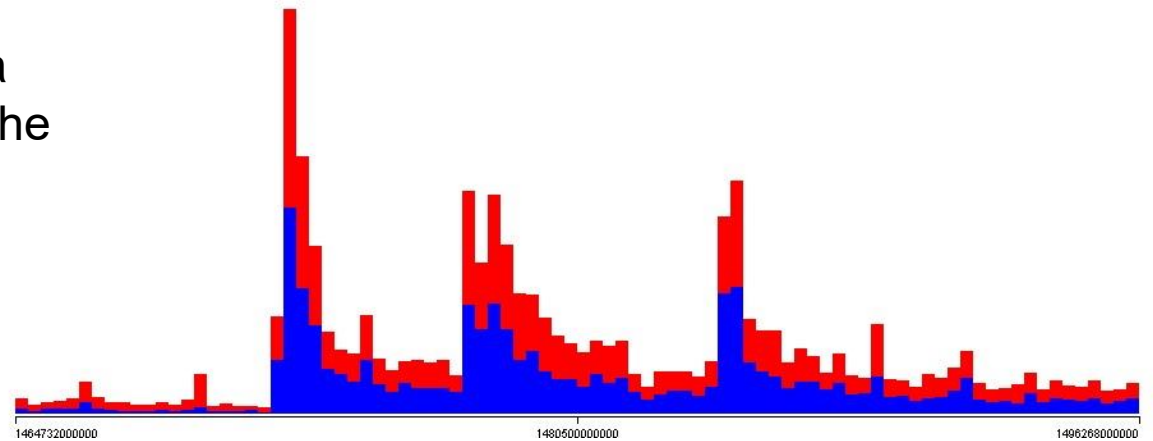


# Using the classifier on the database

---

Since the SMO classifier provided us the best results, we used it in the application. To exploit a further test, we applied the model build on the database containing the 2016-2017 tweets, containing the “terremoto” hashtag.

We can see that the classifier has labeled a good amount of tweets as “earthquake” at the peaks seen previously.



# Toward the application

---

Once we found the right classifier, we started implementing the application using java language. The application fetches real-time tweets containing the keyword “terremoto”, performs a real\_time classification of each fetched tweet; if a positive tweet is found, the information about the tweet are stored in a mysql database. At the end of the fetch step, the user can visualize the results in a dedicated are of the application.