Fundamentals of Data Science
Final Project
SMS Spam Classification

Simone Boesso, Davide Cacciatore,
Giulio Corsetti, Francesca Possenti, Letizia Russo

Fall semester - 2021

Abstract

The purpose of this project is to build a model able to detect if a SMS is a spam message or not. Many of our phones and sms apps have a spam detector that helps us to avoid frauds, virus and malicious contents. We think this is an important topic and we wanted to understand better how it works.
In our project we analyzed the **SMS Spam Collection Dataset**[1] from Kaggle. This dataset contains 5572 sms divided in two classes: ham and spam.
We pre-processed the messages, we divided the dataset in train and test set and then we implemented the *Naïve Bayes Model*. In addition, we performed a *cross validation* analysis to avoid considering only ham messages in our test set, because the dataset is unbalanced.
Then, we compared our best results with the metrics obtained by the most-voted notebook in the Kaggle's code section.
We also tried to change the default decision threshold and tried to use a *Logistic Regression* model and compare our initial results with all this different setups. Finally, we saw what are some of the mis-predicted sms in order to understand what could be the reasons of the model's mistakes.
For this project, we worked with Python and some of its libraries.

Introduction

This report investigates how to find out the nature of a SMS message. In particular we analyzed a Kaggle dataset, named "SMS Spam Collection Dataset", which contains one set of SMS messages in English with 5574 messages, tagged according being *ham* (legitimate) or *spam*.
The goal of this analysis was to classifying SMS as spam or ham using the *Naïve Bayes Model*, in order to avoid two main problems: have our inbox full with undesired messages and receive malicious content.

Related Work

For this job, we considered two related works:

- Spam or Ham: SMS Classifier – Karnika Kapoor[2];
- How to create a spam filter using Bayes' theorem? – Roman Studer[3].

Exploratory Data Analysis

First of all, we can see one example of a spam message and one of a ham sms in our dataset:

- **Ham SMS**: *I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.*
- **Spam SMS**: *WINNER!! As a valued network customer you have been selected to receivea å£900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.*

We can immediately see the differences and recognize that the second one is obviously a spam message, we want that our model recognizes it and targets it as spam.
We can observe how the total amount of messages is divided in the dataset with a barplot.

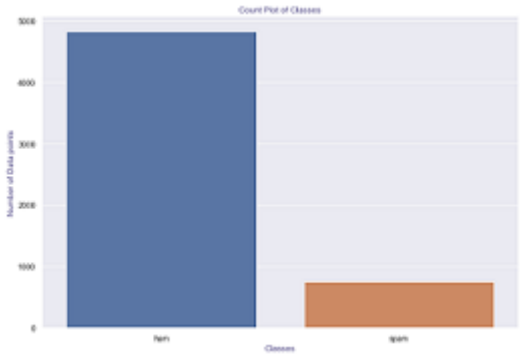

Figura 1: Spam or Ham SMS

As we can see, we have 4825 *ham* SMS and 747 *spam* ones, so the percentage of undesired messages is only the 13.4%, a little part of the total dataset that we're considering, for this reason we can definitely consider it as unbalanced.
Also, we compared the number of characters, words and sentences into the two classes of text messages and, as we can observe into the following figure, in general, the *spam* SMS are longer than the *ham* ones. This section has been mainly taken care by Giulio Corsetti, Francesca Possenti and Letizia Russo.

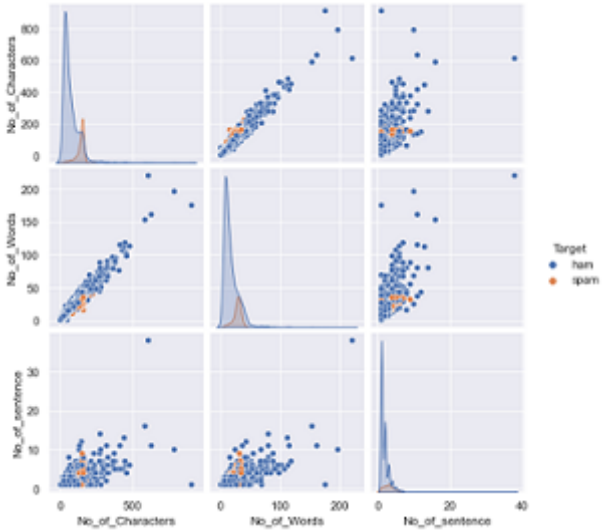

Figura 2: Spam vs Ham

Pre-processing

The next step was the pre-processing of all the messages in the dataset, leveraging the *nltk* library.
We removed the punctuation, we transformed all our texts in lowercase, and we removed all the words that are considered stopwords by the library. Always with the help of *nltk*, we lemmatized all the texts. Lemmatization is an algorithimc process that determins the lemma of a word based on its intended meaning.
Finally, we splitted the words of each message and, at the end, we vectorized each SMS. It

means that we created a vector for each sms, that has a 1 every time a specific word appears in the text and 0 otherwise.
In this way, we created a new dataframe that has all the messages on the rows and each word of our corpus of documents on the columns. This section has been mainly taken care by Giulio Corsetti, Francesca Possenti and Letizia Russo.

Naïve Bayes Model

Before starting the analysis, we splitted the dataset in *train* and *test* set considering the 20% of the original dataset as test-set.
We implemented the *Naïve Bayes Model* from the *sklearn* library. After having trained our model on the train set, we runned it on the test set to see its results. We obtained an high accuracy (98%) and good values also in precision and recall as shown in the table and in the confusion matrix below.

Accuracy	Recall	Precision
0.98	0.92	0.93

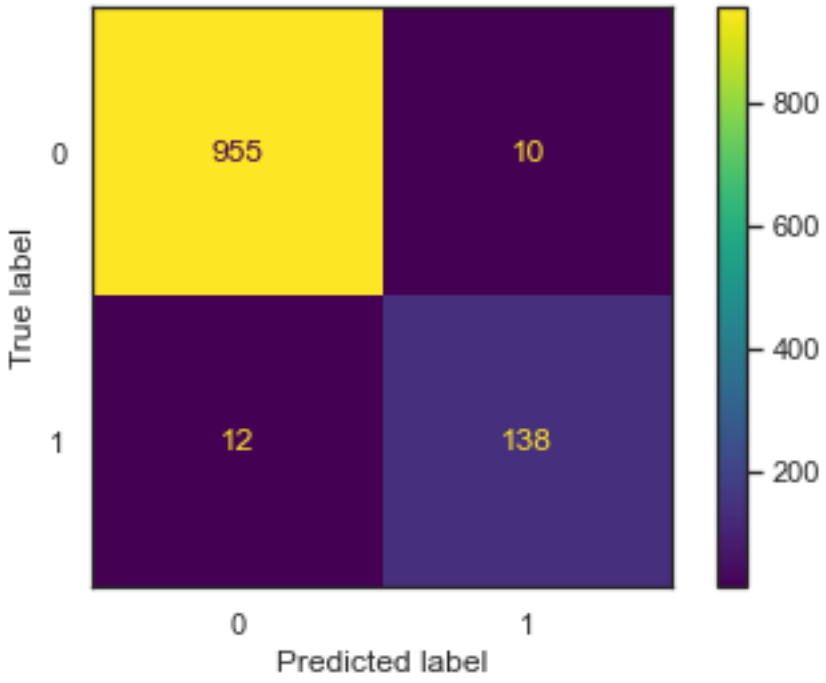

Figura 3: Confusion Matrix

We can observe that only 10 sms have been detected as *spam*, even if they are actually *ham*, and 12 sms have been detected as *ham*, even if they are actually *spam*. All the other ones has been detected correctly.
Anyway, since we have only few SMS labeled as *spam*, our dataset is unbalanced, so we wanted to avoid to have only *ham* in the test-set and we decided to perform a cross-validation analysis, in particular a *5-Fold cross validation*. The dataframe has been split into five chunks and every time a different chunk is considered as the test-set. Also this time, the results are pretty good.

	Accuracy	Recall	Precision
1	0.985	0.953	0.935
2	0.984	0.953	0.929
3	0.983	0.926	0.945
4	0.979	0.926	0.920
5	0.988	0.966	0.947
Average	0.984	0.945	0.935

We noticed that the best model we had from the cross validation is the fifth one. It has an accuracy of $\sim$ 99%, a recall of $\sim$ 97% and a precision of $\sim$ 95%. This section has been mainly taken care by Simone Boesso, Davide Cacciatore and Letizia Russo.

Benchmark and comparisons

We decided to compare the metrics of our best model between the five of the cross-validation with the metrics obtained by the most-voted notebook[2] in the Kaggle's code section:

	Accuracy	Recall	Precision
Our Best Model	0.988	0.966	0.947
Naïve Bayes	0.975	0.705	1.00
Random Forest	0.975	0.816	1.00
KNeighbours	0.975	0.323	0.978
SVC	0.975	0.801	0.991

We realized that, in general, our model trades a lower precision with an higher recall. This was exactly what we wanted because it's crucial to get an high number of true positive, avoiding the situation in which a *spam* SMS is wrongly detected as a legitimate one and goes into our inbox. This section has been mainly taken care by Davide Cacciatore.

Threshold

At first, we decided to estimate the model using the default decision threshold by *sklearn*. It labels a message as spam if its predicted probability to be a spam message is > 0.5. We get the following metrics:

True Positive Rate	False Positive Rate
0.987	0.067

Often we can optimize the threshold to get the desired performances.
We decided to find the threshold that generate the largest difference between *TPR* and *FPR* because it's important to get the lowest value of false positives, but we also need a good value of true positives.
We obtained a new threshold of 0.758, it means that now we label a message as spam, only when its predicted probability is larger than that value.
Using the found threshold we got:

True Positive Rate	False Positive Rate
0.92	0.005

We got a lower value of *TPR*, but a significantly lower value of *FPR*. Obviously this is a more conservative model that tends to label a message as spam only when it is almost sure of that. Infact, it predicts very rarely a message to be spam when it's not.
It's important to keep in mind that this value depends on the dataset that we used to get it. So, in a more realistic environment, the *TPR* and *FPR* values can vary. It'll be important to test this threshold value on other datasets to see the performances. This section has been mainly taken care by Simone Boesso.

Logistic Regression

We decided to build another model, just to compare our results. So, we performed a *Logistic Regression* (always from the *sklearn* library) on the same dataset.
The obtained results are shown in the confusion matrix below:

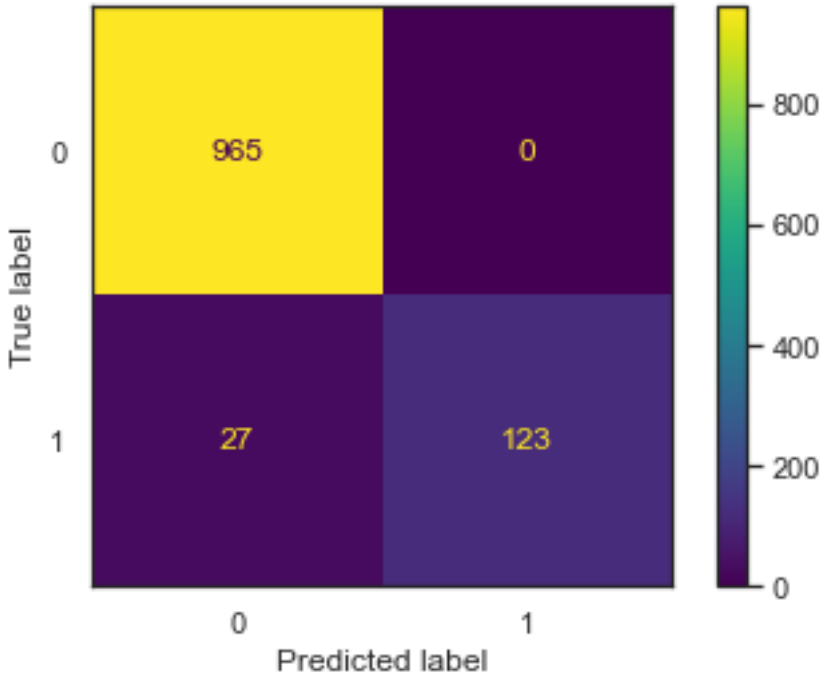

Figura 4: Confusion Matrix

With the following metrics:

Accuracy	Recall	Precision
0.976	0.82	1.00

This model is really precise, it has 0 false positives. Instead, the overall accuracy and the recall are lower than the ones obtained with the Naive Bayes. Infact, it considers 27 spam messages as legitimates.
We can say that, between the two tested models and the others compared, the Naive Bayes model seems to be the one that suits better this kind of task.

Mis-predicted SMS

We decided to find some of the mis-predicted SMS and read them trying to understand the reasons of the Naive Bayes' errors.
We report here two of these messages that, in our opinion, represent two relevant cases.

- **SMS 1066**: *Do you ever notice that when you're driving, anyone going slower than you is an idiot and everyone driving faster than you is a maniac?*

This sms is labeled as *spam* in our dataset, but the model predicted it to be *ham*. This is a **Type II error**. Actually, this sms is well-written and it is really difficult to recognize it as a spam even for a human.

- **SMS 319**: *Total video converter free download type this in google search:)*

This sms is labeled as *ham* in our dataset, but the model predicted it to be *spam*. This is a **Type I error**. In this case, the presence of words as *free* and *download* maybe brought the model to fail. Probably, this message can be written also by some friend that suggests us how to solve a issue with our PC. Anyway, this kind of sentences are really frequent also in spam messages, that ask us to download something for free with bad purposes. This section has been mainly taken care by Davide Cacciatore.

Conclusions

We can say that the *Naive Bayes* model works really fine to solve this kind of task. We had very good results in terms of accuracy, recall and precision and our text pre-processing brought slightly better results than the ones on Kaggle.
We experimented that we can obtain more or less conservative models simply changing the decision threshold. An high one will make our model really conservative, it will predict an sms to be spam only when there is an high predicted probability that is true.
Taking a look at the mis-predicted sms, we realized that many errors maybe can be avoided with a larger initial dataset. A larger corpus of documents will give us more information about many words and it will be helpful to detect more spam messages.
Anyway, some spam messages are really difficult to detect. They seems to be legitimate and they are able to excape many spam detectors. This means that we must always be careful and not trusting 100% our phone's spam detector, because some messages can hide bad contents or frauds even if they seems to be normals.

References

1. SMS Spam Collection Dataset,
https://www.kaggle.com/uciml/sms-spam-collection-dataset

2. Spam or Ham: SMS Classifier,
https://www.kaggle.com/karnikakapoor/spam-or-ham-sms-classifier

3. How to create a spam filter using Bayes' theorem?,
https://medium.com/swlh/how-to-create-a-spam-filter-using-bayes-theorem-f3811f213046