UNIVERSITÀ DI PISA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Laurea Magistrale in Artificial intelligence and Data engineer

Emoji prediction through sentiment analysis

Marcuccetti Gabriele

ANNO ACCADEMICO 2022/2023

Indice

1 Introduction

Emojis are among the most important communication tool.

Your mundane email should include a smiley face or a party hat to add a dash of color or a blip of personality. The emoji options are endless, which ensures that you can keep your readers on their toes at all times.

Not only do emojis liven up your conversations, but they also have the ability to pick up where correct grammar or physical cues are lacking.

For example, when you respond with "Ok..." it appears that you are frustrated or feeling impatient. When you utilize emojis and instead respond with "Ok... :)" it causes less worry and more displays a true understanding from the sender.

1.1 Purposes

The main goal is to create an application that can analyze tweets and can associate an emoji to the input.

In the IOS and android operating systems there is a function that associates, in a 1 to 1 correspondence, the respective emoji to a single word, but there is no function to associate an emoji with entire sentences.

To develop the final classifier used in application, it was necessary to pass through a supervised learning stage, which allowed us to test different classifier and choose the one that gave us better results.

2 Steps

2.1 Dataset

Two datasets were used, both from kaggle. The first includes about 100,000 rows, each containing a tweet and its associated sentiment: 1 for a positive sentiment, 0 for a neutral one and -1 for a negative one. In the second dataset there is a list including about 700 emojis, and for each the relative polarities regarding positive, neutral or negative feelings. An analysis was carried out on the latter dataset to obtain the 7 most representative emojis for each category: subsequently, the relative emojis were associated with each category.

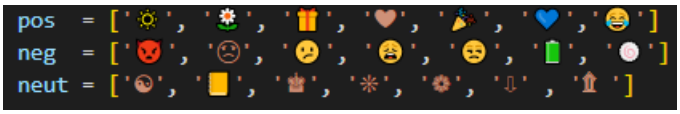


Figura 1: Arrays of emoji

Let's see a word cloud: a collection of words depicted in different sizes. The bigger and bolder the word appears, the more often it's mentioned within a given tweet and the more important it is.

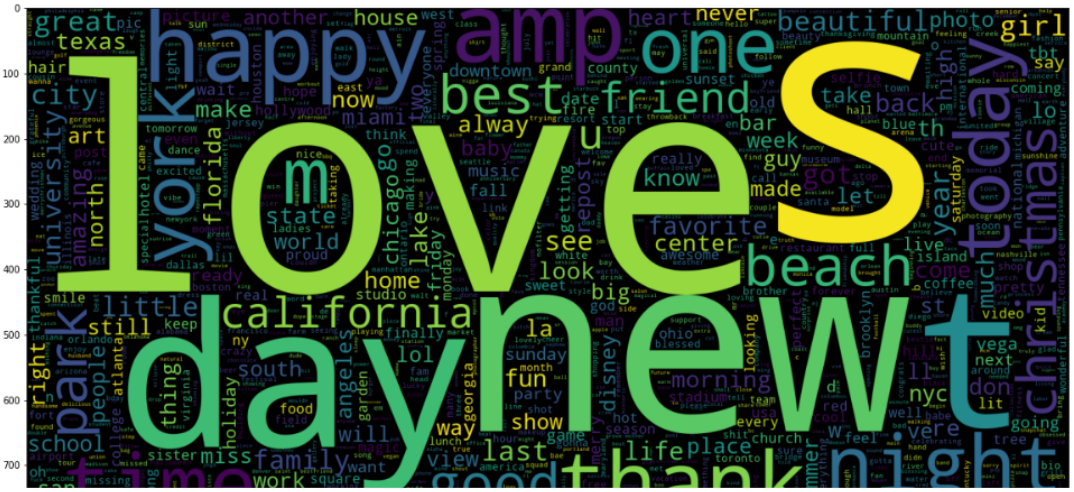


Figura 2: Word cloud

2.2 Preprocessing

When it comes to creating a Machine Learning model, data preprocessing is the first step marking the initiation of the process. Typically, real-world data is incomplete, inconsistent, inaccurate (contains errors or outliers), and often lacks specific attribute values/trends. This is where data preprocessing enters the scenario – it helps to clean, format, and organize the raw data, thereby making it ready-to-go for Machine Learning

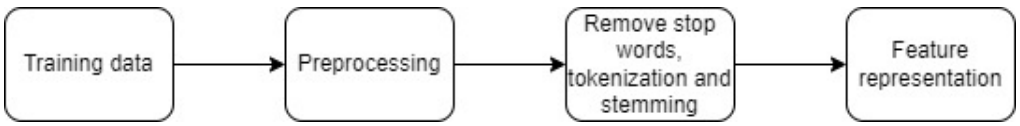
models. Let's explore various steps of data preprocessing in machine learning. There are various techniques, such as normalization, feature extraction, and dimension reduction, it is necessary to better accomplish the classification of data. The aim of preprocessing is to find the most informative set of features to improve the performance of the classifier. We also check for Missing values and split the data into independent and dependent variables along with splitting the dataset to Test and Train in case you don't have one. In particular, we do the following steps:

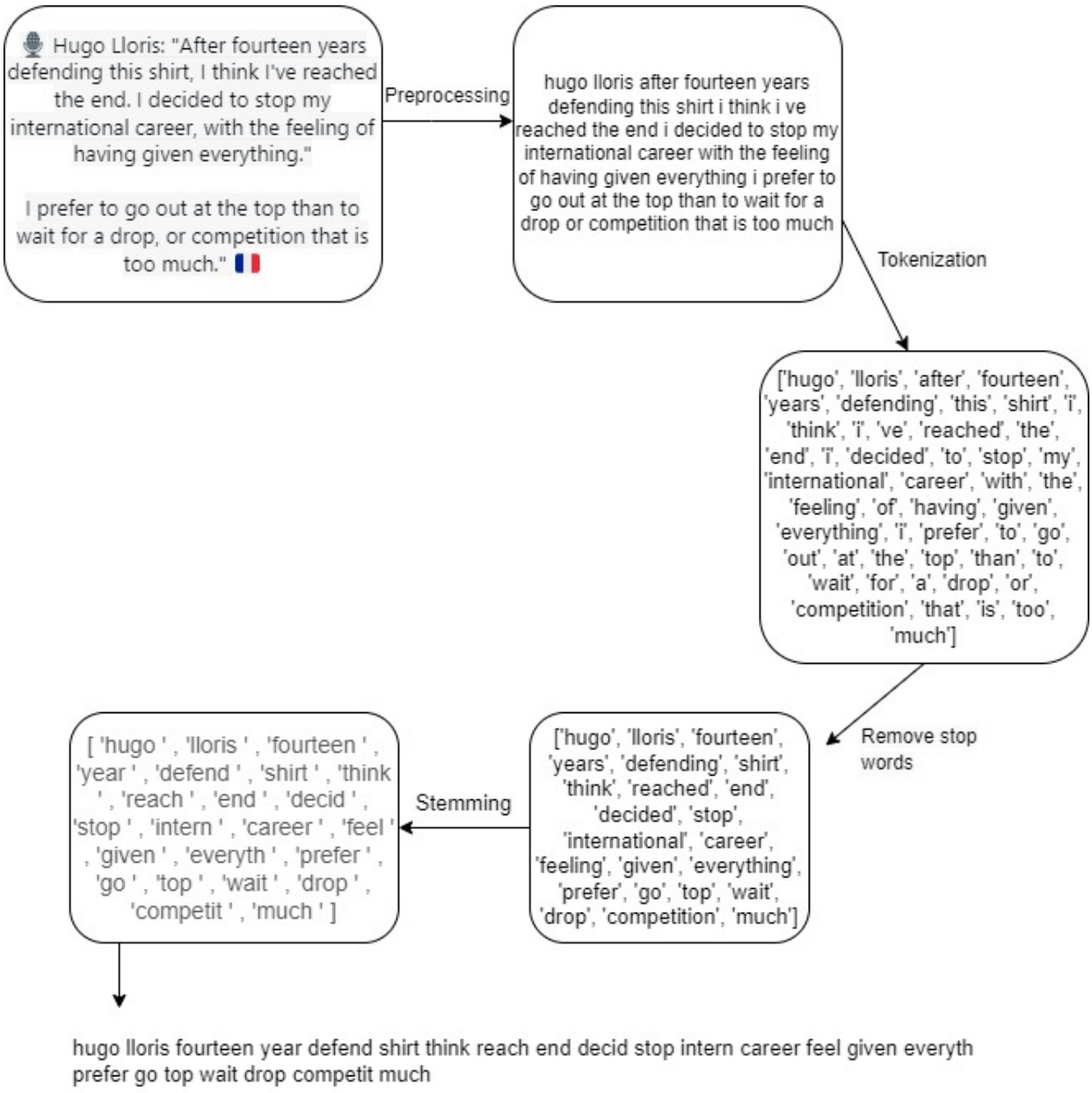
- remove all the @mentions, hashtag and re-tweet
- remove all the hyperlinks that starts with "http", "https", "www", and so on
- remove numbers and special characters
- Replace 3 or more consecutive letters by 2 letter : 'heyyyyyyyyyy' becomes 'heyy'
- remove punctuations and multiple spaces

2.3 Text representation

In order to apply classification algorithms to the tweets we have to represent them as numerical vectors. In particular we have to follow this steps:

- Extract tokens from tweets (tokenization phase). We use the bag-of-words representation.
- Remove stop-words (stop-word filtering) from the set of extracted tokens
- Perform a stemming phase, in order to reduce each token to its "root form"
- Perform a tf-idf transformation, where the importance of each stem is computed using the Inverse Document Frequency





And finally the tf-idf transformation is performed.

2.4 Classification

In this phase, the classifier is learned from a manually labelled training set. We trained different models and we tested them using the cross-validation procedure. Finally, the classifier that gave us better results (in terms of accuracy, precision and recall per class) was chosen as the final classifier to be adopted for the classification of tweets. To train the model different experiments were performed, considering different classifiers and different text representations techniques. We decided to test 7 different classifiers:

- Multinomial classifier
- Bernoulli classifier
- Linear support vector classifier
- Random forest classifier
- K-nn classifier
- Adaboost classifier
- Decision tree classifier

2.5 Application: Classification on unlabelled data

In this phase the unlabelled tweets are classified using the classifier trained in the supervised learning stage (the one that gives better results), after passing through the pre-processing phase and the text representation phase. The application looks like a window with two inputs and two buttons: a sentence is entered in the first input, the set of emoji associated with the sentence appears in the second; then there is a button to empty all the fields, and one to submit the sentence:

Now, user can press "clear" to insert another sentence, and so on. The application was developed thanks to the Tkinter library present on python.

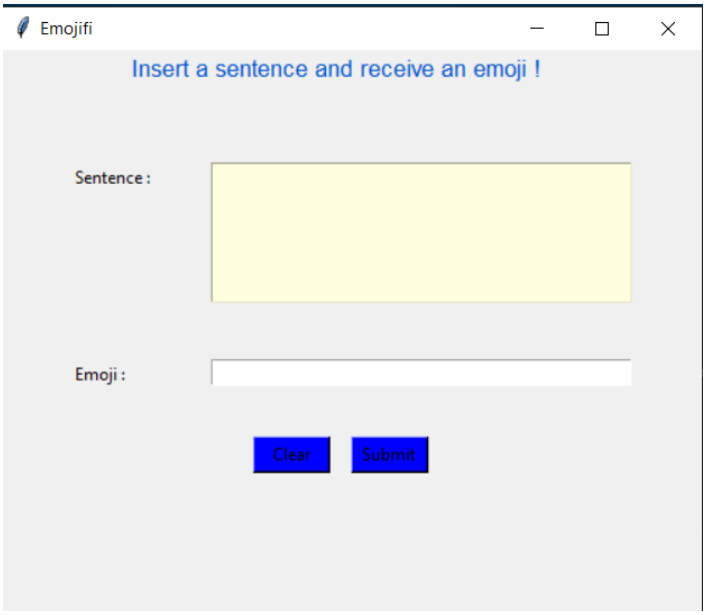


Figura 3: Application window

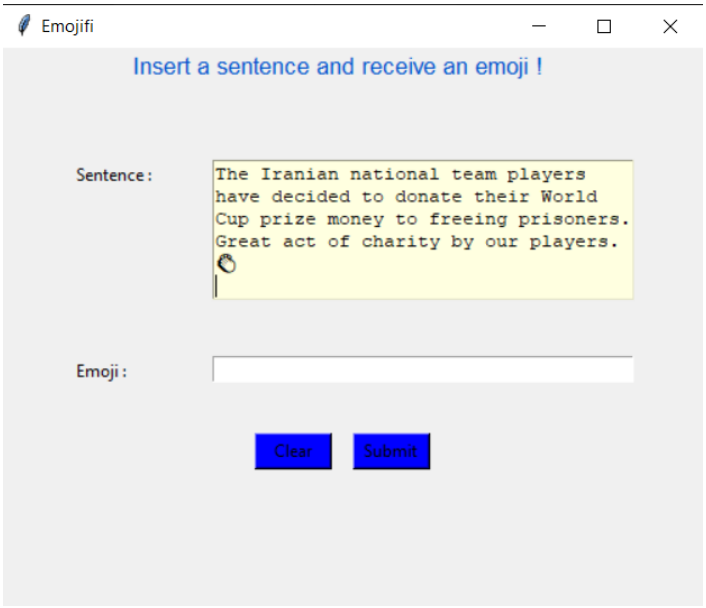


Figura 4: Insert a sentence: Subsequently the user can enter a sentence, let's see this tweet as an example

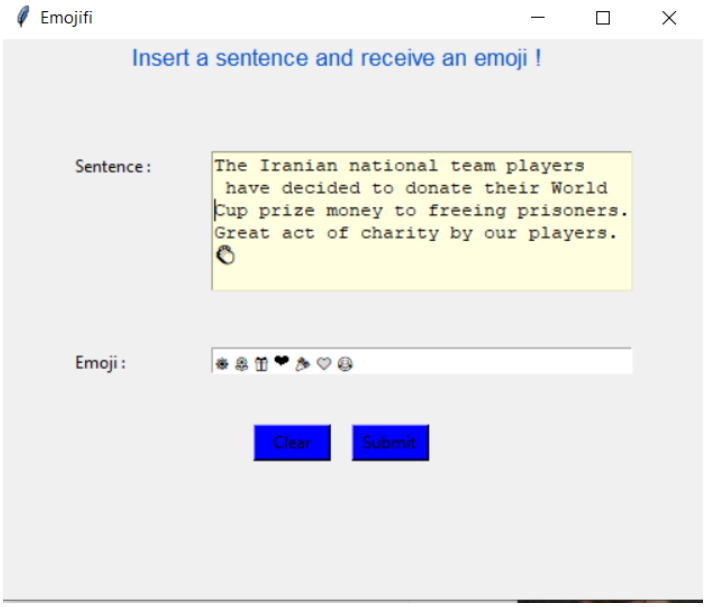


Figura 5: Receive the response: After pressing the button and submitting the sentence, the set of emojis is output

3 Classification results

The performances were evaluated with a 10-fold cross validation with a train test split of 80% and 20%.

3.1 Support Vector Classifier

Accuracy: 0.89 Execution time: 6 seconds

	precision	recall	f1-score	support
-1.0	0.87	0.80	0.83	7170
0.0	0.88	0.95	0.91	10979
1.0	0.92	0.89	0.91	14430
macro avg	0.89	0.88	0.88	32579
weighted avg	0.89	0.89	0.89	32579

3.2 Multinomial Naive Bayes classifier

Accuracy : 0.58 Execution time: 0.43 seconds

	precision	recall	f1-score	support
-1.0	0.94	0.12	0.22	7170
0.0	0.87	0.35	0.49	10979
1.0	0.52	0.98	0.68	14430
macro avg	0.78	0.48	0.46	32579
weighted avg	0.73	0.58	0.52	32579

3.3 Decision tree classifier

Accuracy: 0.80 Execution time: 233 seconds

	precision	recall	f1-score	support
-1.0	0.71	0.66	0.68	7170
0.0	0.82	0.88	0.85	10979
1.0	0.83	0.81	0.82	14430
macro avg	0.79	0.79	0.79	32579
weighted avg	0.80	0.80	0.80	32579

3.4 Bernoulli Naive Bayes Classifier

Accuracy: 0.76 Execution time: 0.50 seconds

	precision	recall	f1-score	support
-1.0	0.71	0.46	0.56	7170
0.0	0.79	0.82	0.81	10979
1.0	0.75	0.85	0.80	14430
macro avg	0.75	0.71	0.72	32579
weighted avg	0.75	0.76	0.75	32579

3.5 K-Nearest Neighbor

Accuracy: 0.53 Execution time: 494 seconds

	precision	recall	f1-score	support
-1.0	0.61	0.30	0.40	7170
0.0	0.44	0.88	0.58	10979
1.0	0.77	0.37	0.50	14430
macro avg	0.61	0.52	0.50	32579
weighted avg	0.62	0.53	0.51	32579

3.6 Random forest classifier

Accuracy: 0.79 Execution time: 332 seconds

	precision	recall	f1-score	support
-1.0	0.77	0.58	0.66	7170
0.0	0.79	0.87	0.83	10979
1.0	0.80	0.83	0.81	14430
macro avg	0.79	0.76	0.77	32579
weighted avg	0.79	0.79	0.79	32579

3.7 Adaboost classifier

Accuracy: 0.76 Execution time: 72 seconds

	precision	recall	f1-score	support
-1.0	0.80	0.55	0.65	7170
0.0	0.65	0.96	0.78	10979
1.0	0.89	0.71	0.79	14430
macro avg	0.78	0.74	0.74	32579
weighted avg	0.79	0.76	0.76	32579

As can be seen in the summary tables, some models manage to obtain very positive results, while others fail to generalize well. In particular, the model that manages to obtain the best results both in terms of f-score and accuracy is the Support vector. For this reason, we choose the latter as the model to use for the user-side application.

4 Conclusion

In summary, we started from two datasets: from the first we extracted the emoji associated with a certain emotion, from the second we trained a model that associates a specific emoji with a tweet. The model that obtained the best results both from the point of view of the f-score and from that of the accuracy is the Support Vector Classifier, with the aforementioned values close to 90Subsequently, an interface was created that allows a user to enter a sentence: the sentence is processed, given as input to the model, which returns the possible emoji associated with the sentence. Possible improvements may concern the ability to output a single emoji and not a set of emojis. The first approach to this project concerned precisely this possibility, but without obtaining positive results. Furthermore, there is a paper (https://ieeexplore.ieee.org/abstract/document/9544680) in which this possibility is discussed. In the work in question we first try to create a classifier with the machine learning methods that we have seen in the course and we come to the conclusion that it is not really suitable for the purpose. For this, neural networks are used, with which it is possible to obtain positive results (accuracy around 90%). Therefore it is probable that to obtain good results there is the need to resort to these tools.

5 Confusion matrix

A confusion matrix is a technique for summarizing the performance of a classification algorithm. Classification accuracy alone can be misleading if you have an unequal number of observations in each class or if you have more than two classes in your dataset. Calculating a confusion matrix can give you a better idea of what your classification model is getting right and what types of errors it is making.

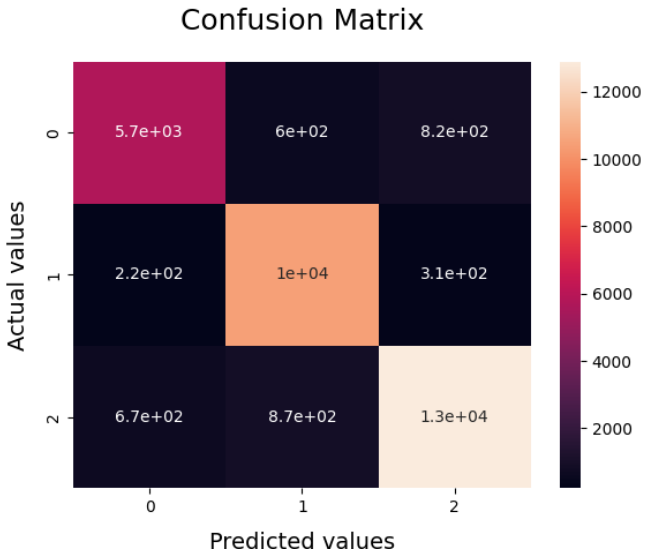


Figura 6: Confusion matrix of Support vector classifier

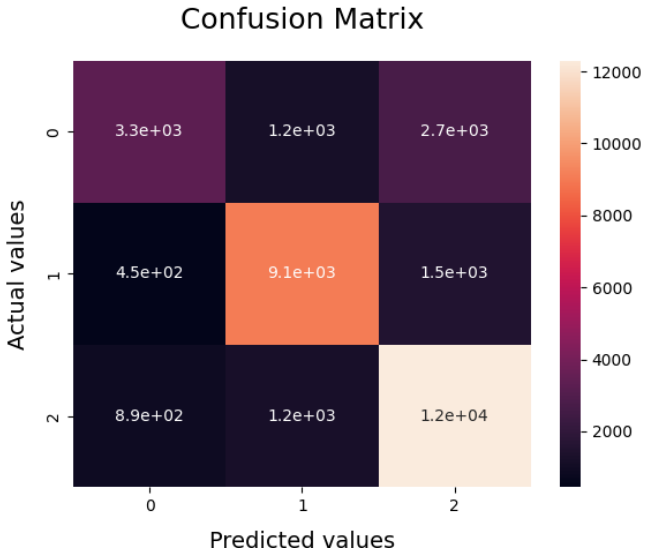


Figura 7: Confusion matrix of Bernoulli classifier

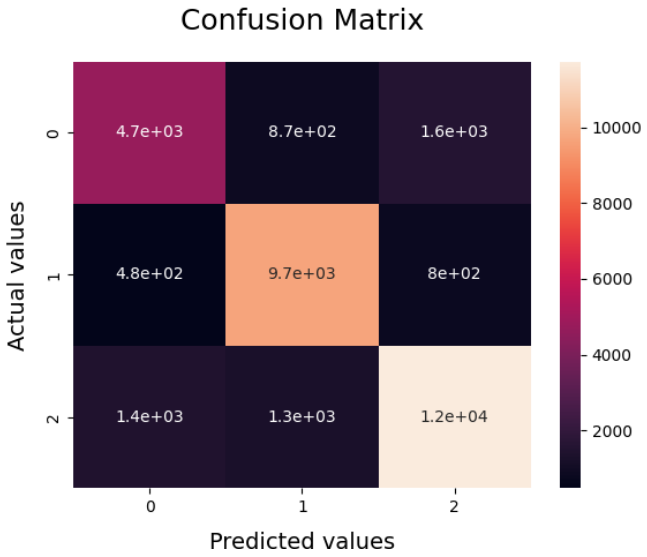


Figura 8: Confusion matrix of Decision tree classifier

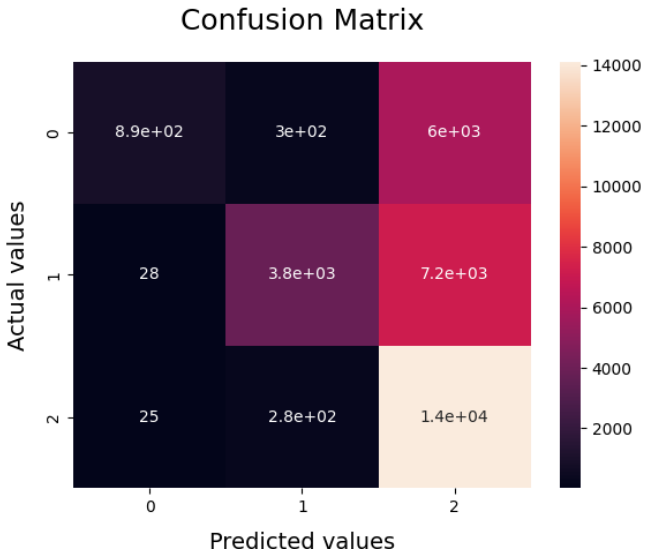


Figura 9: Confusion matrix of Multinomial classifier

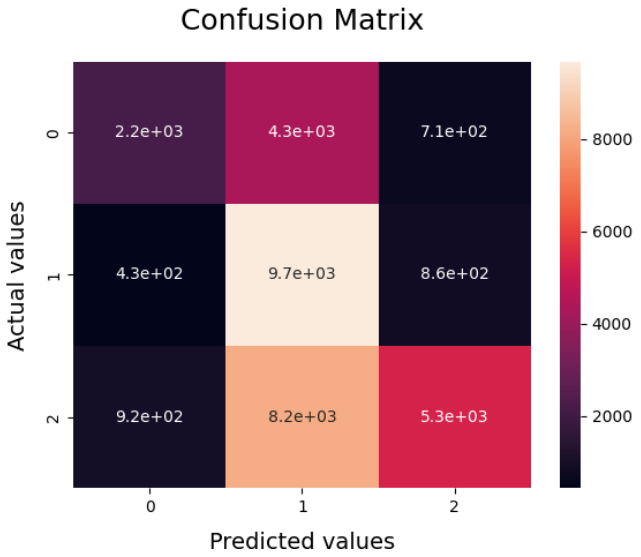


Figura 10: Confusion matrix of K-nearest neighbour

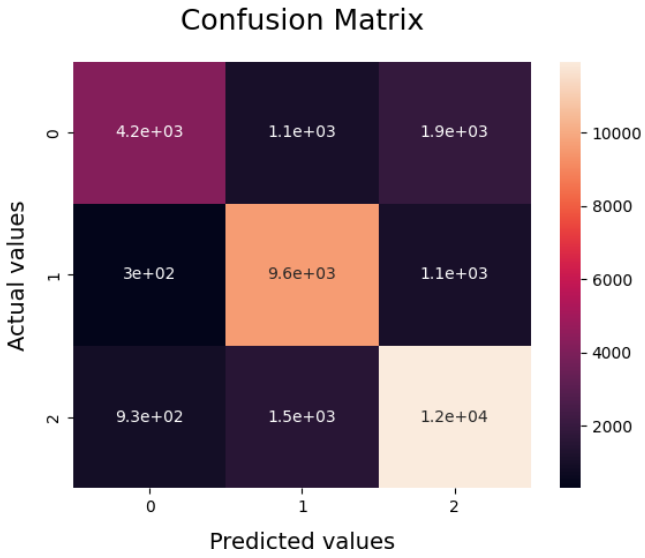


Figura 11: Confusion matrix of Random forest

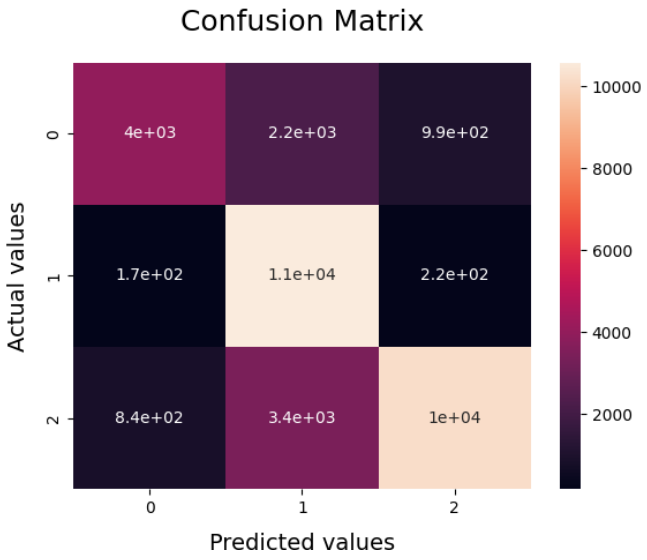


Figura 12: Confusion matrix of Adaboost classifier