UNIVERSITY OF TRENTO

Department of Psychology and Cognitive Science

Department of Information Engineering and Computer Science

Master's Degree in
Human-Computer Interaction

AUTOMATIC CLASSIFICATION OF THE PERCEPTION OF ARABS ON MIGRATION TO EUROPE USING SOCIAL MEDIA MONITORING AND NATURAL LANGUAGE PROCESSING

Supervisors
Prof. Sara Tonelli
Prof. Massimo Zancanaro

Student
Mohamad Baalbaki

2019-2020

Abstract

Arab influx to Europe continues to rise day by day, but Arabs' perceptions of it as a migration destination remains unclear to the world. Despite the fact that the exact number of Arabs in Europe is unknown, the scarce research that exists shows that it currently hovers around 6 million, slightly more than the entire population of Denmark. This study aimed to utilize Social Media Monitoring and Natural Language Processing to explore the possibility of creating a computer program that will be able to classify if a tweet in Arabic is: unrelated to, in favor of, or against migration to Europe. In simpler words, it answered the following question: Can a computer program identify if Arabs have a positive or negative perception on migration to Europe? In this context, a perception is defined as the individual's stance on this topic. A positive perception signifies that they find Europe as a favorable place to migrate to and a negative perception signifies that they express an aversion to migration to Europe. Knowing that data is a driving force for computer programs with machine learning applications, the collection and analysis of a substantial amount of relevant Arabic tweets provided us with an accurate insight on the former hypothesis. Consequently, based on a literature review on social media monitoring and stance detection, a Twitter crawler was used to collect and store Arabic tweets based on keywords related to migration to Europe. These tweets were then filtered by eliminating retweets and partially copied tweets, and used to create a dataset with *3527 tweets*. Each tweet was manually annotated with *1* if it conveyed a positive perception on migration to Europe, *0* if it conveyed a negative perception and *-1* if the tweet was either news or unrelated. After training our machine learning algorithms on the aforementioned tweets, the most optimal approach was the Soft Voting Classifier with an overall classification accuracy of ***62.32%***. Our algorithm determined that Arabs carry a **negative** perception on migration to Europe. Despite the fact that we believe our results were fair and above average, we assert that further research is needed with much more data paired up with Deep Learning algorithms. Such an approach might drastically improve classification quality and consequently yield a more accurate perception.

Acknowledgements

I would like to deeply thank my supervisors: Professor Sara Tonelli and Professor Massimo Zancanaro for their support and transfer of knowledge during the period of my thesis. They have been of great help to me throughout these couple of months. In addition to that, I would like to thank "Fondazione Bruno Kessler" for the opportunity they have given me both for my internship and for my thesis. My skills flourished under the guidance of their professionals.

I thank my parents, siblings and my aunt for their hard work these years to try to lift me up and help me become the best version of myself. I will never be able to pay them back for what they have provided me. They have my utter respect and a million thank yous to them would never be enough.

My deepest gratitude goes to my girlfriend Martina for her constant support, both academically and emotionally. We had some tough times together, some ups and downs, it was a roller coaster but even in the worst of times, she didn't give up on me and she kept pushing me to become better.

One of the key people I would like to thank is my best friend Adeen. He's always had my back, guiding and supporting me in my quest to finish my master's, as well as in other personal matters. Since we've had parallel lives, his words and advice will always be of great value to me.

A particular thank you goes to the Italian Government as if it wasn't for them, I wouldn't be here. The scholarship they gave me helped me spread my wings and fly to the largest professional services network in the world: Deloitte. If it wasn't for Italy, I don't think I would have ever dreamt of getting employed in such a company. Thus, I would like to emphatically say: **GRAZIE ITALIA!**

The biggest thank You goes to God. My faith in You has never and will never tremble. I thank You for every blessing You have given me. Therefore, I would like to start my thesis by saying:

بِسْمِ اللَّـهِ الرَّحْمَـن الرَّحِيمِ

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Arab presence in Europe began in the Iberian Peninsula with the rise of the Umayyad Caliphate in 711 AD [Afsaruddin, 2019]. They remained in Europe till this day, reaching an estimated population of around 6 million individuals [for Migration, 2010] in 2020. Their overall population has not been saturated yet, as the number of Arabs immigrating to Europe is on the rise everyday, with this decade being flagged as "the decade of Arab refugees" [Salameh, 2019]. Despite this continuous increase, to our knowledge there have not been any studies to try to identify the perception of Arabs on migration to Europe online in an automated manner. This is imperative as it would help us to better understand the narratives that influence their aspirations to migrate, as well as some false expectations that they might carry. Therefore, we believed that it is very interesting to try to see if it is technically possible to identify this perception using Social Media Monitoring, Machine Learning and Natural Language Processing. Despite the fact that there is constant a influx of Arabs to Europe, our research has surprisingly shown that a combination of the aforementioned techniques successfully identified an overall **negative** perception of Arabs on migration to Europe mainly caused by: religious differences, political grudges and economic inequalities between Arabs and Europeans. There have been few works on stance detection [Krejzl et al., 2017] in Arabic that we got inspired by. They are briefly explained below and with more detail in the Literature Review section.

[Baly et al., 2018] worked on stance detection for Arabic fact checking. They claimed that if a set of tweets has already been manually annotated, better results can be achieved when supervised learning techniques [Russell and Norvig, 2009] are used for stance prediction. Thus, we got inspired by their work and we opted for supervised learning in our study.

Another inspiring work was the one of [Soliman et al., 2014] on Sentiment Analysis of Arabic Slang Comments on Facebook. Their use of Support-Vector Machines (SVM) [Cortes and Vapnik, 1995] yielded an accuracy of 86.86%, therefore we started off by trying SVM and then moved on to try other supervised learning algorithms.

The most interesting work for us was the one of [Alayba et al., 2017]. Their creation of an Arabic dataset on health services after crawling tweets and their choice to test several machine learning algorithms for sentiment analysis totally fit our vision. Instead, we opted to try the same procedure but for the purpose of stance detection.

In our study, we carefully selected a total of 44 keywords related to Arab migration to Europe. Using these keywords, we crawled tweets on a span of 3 months and we manually annotated them as: *-1* which means unrelated, *0* which means that the tweet conveyed a negative perception on migration to Europe and *1* which means that it conveyed a positive perception on migration to Europe. At first, we wanted to focus on tweets only from Algeria, Egypt and Tunisia but we reverted from doing so after noticing that there were scarce tweets geotagged with these countries. Therefore, we resorted to focus on all

tweets in Arabic regardless of their geolocation. We programmatically removed retweets and partially copied tweets, and we kept only one instance of each set of similar tweets. We incorporated the use of a second annotator and we calculated the inter-annotator agreement and performed adjudication in order to enhance the quality of our annotations. In addition to that, we created the first dataset for Arabs' perceptions on migration to Europe that we named "APME" Dataset. At that point we started pre-processing our data and we tried out 6 different machine learning algorithms: Support-Vector Machines, Naïve Bayes, K-Nearest Neighbors, Random Forests, Logistic Regression and the Voting Classifier with 17 unique approaches for all of them that we mentioned in our "Machine Learning Approaches for Tweet Classification" section. At the end of our crawls, we had a total of *3527 tweets* and we garnered a maximum overall classification accuracy hovering around ***62%***, which was achieved by: Logistic Regression and the Voting Classifier (both Soft and Hard approaches). After analyzing the misclassified tweets, we determined that some of the reasons behind these errors could be justified by the use of dialectal Arabic and by the use of metaphors. At the end of our research and based on our sample of tweets, our algorithms finally detected that Arabs have a **negative** perception on migration to Europe.

This thesis is divided into 6 sections: Section 1 is the Introduction. Section 2 describes the literature of relevant works on stance detection in English and Arabic. Section 3 explains how we collected our corpus and created our dataset. Section 4 describes the different pre-processing approaches and machine learning algorithms we used. Section 5 shows our results in details with error analysis. Section 6 concludes and outlines further research.

Chapter 2

Literature Review

Stance detection is the automatic detection whether the author of a piece of text is in favor of a given target or against it. Despite the fact that there are few works in Arabic regarding that matter, the ones that were already accomplished have really paved the way for further research. In our paper, we mentioned some interesting studies on stance detection in English before shifting to stance detection in Arabic tweets.

[Mohtarami et al., 2018] worked on an English dataset for stance detection. They used an end-to-end memory network [Sukhbaatar et al., 2015] for its ability to remember information from previous paragraphs and to infer more accurately. Their network predicted whether a corpus of text agrees with, disagrees with, discusses or is unrelated to a given claim. In addition to that, it stored evidence for the inferences it has already made. They integrated convolutional and recurrent neural networks [Goodfellow et al., 2016], as well as a similarity matrix in order to handle noise. They achieved a weighted accuracy of 81.23% in the Fake News Challenge [Masood and Aker, 2018].

Another important work on stance detection in English was the work done by [Augenstein et al., 2016]. They used bidirectional LSTMs [Huang et al., 2015] with a conditional encoding mechanism for stance detection in political tweets. Their results have shown that conditional encoding is satisfactory when it comes to stance detection for unseen targets. In fact, they achieved the second best results on the SemEval 2016 Twitter Stance Detection corpus [Mohammad et al., 2016].

[Riedel et al., 2017] used a multilayer perceptron [Murtagh, 1991] with one hidden layer, in addition to lexical and similarity features. Despite the fact that they garnered third place in the Fake News Challenge with an overall accuracy of 81.7%, their results were quite misleading since their model was not very accurate at predicting the 2 most important classes: "agree" and "disagree".

[Baly et al., 2018]'s work on stance detection for Arabic fact checking was the first of its kind. They claimed that existing datasets in Arabic usually do not offer manual annotation of supporting evidence. In addition to that, they mentioned that although evidence can be automatically extracted through unsupervised techniques [Parikh et al., 2016], better results can be achieved when supervised or semi-supervised techniques are used. They mentioned that this can happen given a set of manually annotated tweets and that it yields better stance prediction. Therefore, since we had the expertise needed to annotate our data, we got inspired by their work and we opted for supervised learning.

[Darwish et al., 2017] improved on stance prediction for both Arabic and English. They built their study on the previous works of [Magdy et al., 2016, Pennacchiotti and Popescu, 2011, Wong et al., 2013] that claimed that using users' irrelevant previous tweets and retweets in regards to a certain topic can be used to identify their stance on it. Therefore, they opted to compute similarities between users based on their interaction patterns, hashtags and their retweets. Their results have shown significant

improvements in stance prediction for both languages.

Another significant related work was the one of [Soliman et al., 2014] on Arabic political comments written in dialectal Arabic from Arabic news sites like: Al-Jazeera and BBC Arabic. They used an SVM classifier for Arabic slang language to classify Arabic news comments on Facebook as: satisfied or dissatisfied. They also created an Arabic dialectal lexicon that they named: "Slang Sentimental Words and Idioms Lexicon" (SSWIL). They achieved good results, with their SVM classifier yielding an overall accuracy of 86.86%.

The work of [Touati et al., 2017] on Arabic Opinion Summarization using machine learning and more specifically Conditional Random Fields (CRF) [Lafferty et al., 2001] was one of the first of its kind. They annotated opinions in news articles by focusing on four categories: reporting, judgement, advice and sentiment. They used an arsenal of different features such as: n-grams, linguistic and semantic information, in addition to features that were specific to the Arabic language. They have shown the effectiveness of using Conditional Random Fields for this problem.

One of the most inspiring studies for us was the study done by [Alayba et al., 2017]. They created an Arabic dataset on tweets related to health services and they named it: "Arabic Health Services (AHS)" Dataset. Their aim was to run a machine learning algorithm that will be able to accurately classify unseen tweets related to health services as positive or negative. They tested several machine learning algorithms such as Naïve Bayes [Webb, 2010], Support-Vector Machines [Cortes and Vapnik, 1995] and Logistic Regression [Kleinbaum et al., 2002]. In addition to that, they also tried Deep Learning algorithms such as Deep Neural Networks and Convolutional Neural Networks [Goodfellow et al., 2016] with Word2Vec [Mikolov et al., 2013]. They collected tweets on a span of 6 consecutive months to have their data eventually labeled by three annotators. After removing irrelevant tweets, they were left with an overall number of 2026 tweets in their dataset where 31% of the tweets were positive and the rest 69% were negative. Their highest scoring classifier was SVM which yielded an overall accuracy of 85%.

Last but not least for works in Arabic, [Al-Smadi et al., 2018] used supervised Machine Learning and Deep Learning approaches for aspect-based sentiment analysis [Pontiki et al., 2014] of Arabic Hotel reviews. Their Machine Learning approach consisted of using Support-Vector Machines (SVM), while their Deep Learning one consisted of using Recurrent Neural Networks (RNN). Both algorithms were trained by using different features and were evaluated using a test set that contained 2291 reviews. Their results have shown that SVM outperformed RNN in classification by yielding a whooping 95.4% overall accuracy, while RNN scored 87%. It is important to state that RNN outperformed SVM only in the execution time required for training and testing the models.

The last two works have shown that Machine Learning approaches yielded more accurate results for classification tasks similar to ours. This can be explained by the relatively small sizes of the datasets in these studies. Whenever there are less tweets, it is usually better to resort to Machine Learning approaches such as Naïve Bayes and SVM rather than Deep Learning approaches such as Recurrent Neural Networks. Consequently, we decided to opt for Machine Learning algorithms instead of Deep Learning algorithms in our study.

Chapter 3

Corpus Collection and Dataset Creation

3.1 Social Networking Platform Selection

The social networking platform we chose in order to get the data from is Twitter [Murthy, 2018]. It is one of the most used micro-blogging platforms in the world and it is often used by people to express their opinions publicly [Stieglitz and Dang-Xuan, 2013]. It has long been an attraction for researchers because of its enormous number of textual information that grows everyday [Pak and Paroubek, 2010]. In addition to that, it is extensively used for its research on user intentions [Java et al., 2007]. We refrained from using Facebook because of its privacy restrictions and limited API that prevent us from collecting large amounts of data [Facebook, 2020]. On the other hand, Twitter's API allows developers to access much more data with less limitations [Twitter, 2020].

3.2 Keyword Selection

The importance of keyword selection for text mining has been highlighted in many studies [Cheong et al., 2011, Clifton et al., 2004, Li et al., 2009]. Therefore, it was of crucial importance to choose relevant keywords that will render quality tweets for our Machine Learning models. The language we chose to search for keywords in is Arabic since it is the native tongue of Arabs. It is important to specify that we are native Arabic speakers, thus no professional linguistic assistance in this language was needed. Furthermore, our choice of Arabic keywords guaranteed picking up tweets written in dialectal Arabic which enriched our collected data. From now on, every time we introduced an Arabic word or sentence, we used the implication symbol "=>" paired up with the sentence's English translation so that non-Arabic speaking readers can understand.
Since this entire topic revolved around the two concepts of "Migration" and "Europe", we started by selecting the lemmas "هجرة" => "Migration" and "أوروبا" => "Europe" and we combined them to create the base keyword "هجرة أوروبا" => "Migration Europe". We derived other keywords from our base keyword by combining "Europe" with the synonyms, derivations, singular and plural forms of "Migration" in Arabic to create unbiased and independent keywords. We tested all the keywords manually on Twitter to verify if they would yield a substantial amount of quality tweets. At the end of our testing, only 44 keywords survived and were chosen. These keywords along with their corresponding English translations are shown in Table 3.1.

Keywords	*English Translation*
هجرة أوروبا، إغتراب أوروبا، نزوح أوروبا	Immigration Europe
الهجرة أوروبا، الإغتراب أوروبا، النزوح أوروبا	The Immigration Europe
لجوء أوروبا	Asylum Europe
اللجوء أوروبا	The Asylum Europe
مهاجر أوروبا، مغترب أوروبا، نازح أوروبا	Immigrant Europe
المهاجر أوروبا، المغترب أوروبا، النازح أوروبا	The Immigrant Europe
لاجئ أوروبا	Asylum Seeker Europe
اللاجئ أوروبا	The Asylum Seeker Europe
مهاجرون أوروبا، مهاجرو أوروبا، مهاجرين أوروبا، مهاجري أوروبا، مغتربون أوروبا، مغتربو أوروبا، مغتربين أوروبا، مغتربي أوروبا، نازحون أوروبا، نازحو أوروبا، نازحين أوروبا، نازحي أوروبا	Immigrants Europe
المهاجرون أوروبا، المهاجرين أوروبا، المغتربون أوروبا، المغتربين أوروبا، النازحون أوروبا، النازحين أوروبا	The Immigrants Europe
لاجئون أوروبا، لاجئو أوروبا، لاجئين أوروبا، لاجئي أوروبا	Asylum Seekers Europe
اللاجئون أوروبا، اللاجئين أوروبا	The Asylum Seekers Europe
هجرة لجوء أوروبا	Immigration Asylum Europe
الهجرة اللجوء أوروبا	The Immigration The Asylum Europe
هجرة غير شرعية أوروبا	Illegal Immigration Europe
الهجرة غير الشرعية أوروبا	The Illegal Immigration Europe

Table 3.1: Arabic Keywords with their Corresponding English Translations

Despite the fact that Twitter is a nurturing environment for hashtags [Bruns and Burgess, 2011], we did not use any because the keywords we chose already cover the hashtags that contain the same words. For explanatory purposes, we used "*" to denote a sequence of 0 or more non-connecting Arabic letters or random characters including spaces and numbers.
The keyword "لجوء أوروبا" => "Asylum Europe" already covers itself plus 0 or more additions as demonstrated: "*لجوء*أوروبا*" => "*Asylum*Europe*". In addition to that, it covers the following collections of hashtags where the order of the words can be permuted:

- "*لجوء*#*#*أوروبا*" => "#*Asylum*#*Europe*"
- "*لجوء*#*أوروبا*" => "#*Asylum*Europe*"
- "*لجوء*#*أوروبا*" => "*Asylum*#*Europe*"

We refrained from writing the permutations of these keywords since they render the same

results from Twitter's API. Hence, "#*لجوء*#*أوروبا*" => "#*Asylum*#*Europe*" is the same as "#*أوروبا*#*لجوء*" => "#*Europe*#*Asylum*".
Our keyword selection technique helped us to reduce the number of chosen keywords, and consequently helped in diminishing the probability of having repetitive tweets.

3.3 Tweet Crawling

3.3.1 Pre-filtering

After choosing our 44 keywords, we started running our crawler every 8 days to collect tweets not on coinciding days since Twitter's API picks up tweets for a maximum of 7 days back.

- First Crawl: 25$^{\text{th}}$ of September 2019
  Gathered a total of *1890 tweets*.
- Second Crawl: 3$^{\text{rd}}$ of October 2019
  Gathered a total of *1919 tweets*.

At the end of our second crawl, we gathered an overall number of *3809 tweets*. This is slightly more than 3000 tweets, which is the recommended number proposed by [Sidorov et al., 2012] in their study on Machine Learning approaches for opinion mining in tweets.

3.3.2 Peri-filtering

Geo-Location Filtering Tentative with Retweet Removal

Once we gathered a good amount of tweets, we wanted to focus on the 3 predominant nations that migrate to Europe: Algeria, Egypt, Tunisia [Fargues and Fandrich, 2012, Forin and Healy, 2018]. We initially decided to filter tweets via geolocation and only keep the ones that were geotagged with these countries. In addition to that, we noticed that a substantial amount of our data was from retweets. This was not very favorable for us since they could have tampered with our statistics because of their duplicate annotations. Therefore, we decided to filter via geolocation and to remove retweets all while maintaining an overall number of tweets higher than 3000.
After programmatically removing retweets, only *45 tweets* out of a total of 3809 tweets remained, thus there was a whooping 98.81% loss of tweets. After further investigation, it turned out that there was an extremely scarce number of geotagged tweets with the aforementioned countries. A viable reason to explain this scarcity could be that tweets are not often geotagged in general [Li et al., 2018]. Therefore, we decided to refrain from filtering via geolocation and to focus on all tweets in Arabic while removing retweets. After applying this procedure, we were left with an overall number of *722 tweets* instead of *45 tweets*.

Retweets and Repetitive Tweets Removal via Levenshtein Distance

After we overcame the aforementioned mishap, we ran our third crawl.

- Third Crawl: 10$^{\text{th}}$ of October 2019
  Gathered a total of *2117 tweets*.

Upon the removal of retweets from these 2117 tweets, we were left with 300 tweets in this crawl and an overall number of *1022 tweets*. At that point, we noticed that there were

similar tweets that were partially copied and not retweeted. For example:

- *Tweet 1:*

  "الرئيس الفرنسي يدعو دول أوروبا إلى مواجهة الهجرة غير الشرعية"

  => "The French president calls European nations to face illegal immigration".

- *Tweet 2:*

  "الرئيس الفرنسي يدعو دول أوروبا إلى مواجهة الهجرة غير الشرعية عقب اجتماع بينه و بين المستشارة الألمانية أنجيلا ميركل"

  => "The French president calls European nations to face illegal immigration after a meeting between him and the German chancellor Angela Merkel".

These types of repetitions could have been problematic in the annotation phase of our dataset since we would have duplicate annotations for semantically similar tweets. Therefore, we decided to remove these repetitions all while keeping one tweet for each set of similar tweets. In order to do so, we used the Levenshtein distance algorithm for string similarity [Levenshtein, 1966] to identify similar tweets. After manually testing different thresholds, we finally chose a similarity threshold of *0.5* because we found that it yielded satisfying results for similarity in Arabic tweets. All but 1 tweet with a Levenshtein distance greater than *0.5* were considered as similar tweets and were removed.
While running this algorithm on our *722 tweets*, 157 tweets were identified as being similar tweets and 156 of them were consequently deleted. Therefore, we were left with an overall number of *865 unique tweets without any retweets or repetitions.*

3.3.3 Post-selection of our Filtering Strategy

Since we were left with unique tweets, we considered our filtering strategy of retweets and similar tweets removal using Levenshtein's algorithm as the final filtering strategy that we used till our last crawl. The following table 3.2 shows all of the crawls we performed after selecting our filtering strategy, with the relevant information for each one:

Crawl	*Date of Crawl*	*Crawled Tweets*	*Removed Retweets*	*Removed Similar Tweets*	*Total Number of Removed Tweets*	*Remaining Tweets in this Crawl*	*Previous Total Number of Tweets*	*TOTAL*
4th	21/10/2019	1820	1270	80	1350	470	865	1335
5th	30/10/2019	1618	1222	89	1311	307	1335	1642
6th	07/11/2019	1082	803	58	861	221	1642	1863
7th	15/11/2019	2406	1929	190	2119	287	1863	2150
8th	27/11/2019	1643	1315	49	1364	279	2150	2429
9th	05/12/2019	1632	1296	50	1346	286	2429	2715
10th	13/12/2019	1254	952	42	994	260	2715	2975
11th	21/12/2019	1024	698	49	747	277	2975	3252
12th (Final)	30/12/2019	1411	1055	81	1136	275	3252	3527

Table 3.2: Detailed Information about the Crawls

3.4 Dataset Creation

Shortly after our third crawl, thus after we collected *865 tweets*, we created the first dataset for Arab's perceptions on migration to Europe that we named: "Arabs' Perceptions on Migration to Europe (APME)" Dataset.

3.4.1 Variable Selection Rationale

Inspired by [Nabil et al., 2015]'s "Arabic Sentiment Tweets Dataset", we chose their same structure but we disregarded the tweets' English translations. Instead, we opted to add the tweet ids and locations. The variables we chose are:

1. Tweet ID: We chose to include the tweet ids in the dataset because according to Twitter's terms and conditions, if a dataset is to be published, one cannot include the tweets in the dataset. They can only include the tweets' respective ids for confidentiality reasons, as users have the right to delete their tweets without having them stored in a dataset.

2. Tweet: We included the tweets in the dataset to ease the annotation process, but it is imperative to mention that they are to be removed in case of any publication.

3. Location: We decided to include the location in the dataset in case we opt to identify the prevailing countries in which there are positive or negative perceptions on migration to Europe in the future.

4. Annotation: The last and most important variable we chose to include was the annotation which corresponds to the stance of a person on a certain tweet. This variable governed the entire quality of our final classifications. The following section clearly explains how we annotated the tweets by providing the guidelines that we followed in order to do so. In addition to that, table 3.3 shows 3 tweets that were annotated as 3 different classes. The translation column was not actually included in our final dataset, but it was shown in the table just to explain the Arabic tweets to non-Arabic speaking readers.

Tweet ID	*Tweet*	*Translation*	*Location*	*Annotation*
XYZ	والله العظيم لو نقدر كنت نركب اول قارب الى اوروبا	I swear if I was able, I would hop on the first ferry to Europe	Libya	1
X'Y'Z'	اذا كانت أوروبا هيه صانعة الازمه ازاي هتشارك في دعم اللاجئين	If Europe was the creator of the crisis, then how will it contribute in supporting refugees?	Egypt	0
X"Y"Z"	الأمم المتحدة: عدد المهاجرين الغرقى في البحر المتوسط تجاوز الألف منذ بداية العام	United Nations: The number of migrants that drowned in the Mediterranean sea exceeded a thousand since the beginning of the year	Europe	-1

Table 3.3: Snippet of the APME Dataset

3.5 Dataset Annotation

3.5.1 Annotation Guidelines

Drawing inspiration from [Mohammad et al., 2017]'s work, we created annotation guidelines of our own that were given to another annotator at a later stage. It is important to mention that the following guidelines served as the benchmark for our annotation strategy:

- Annotate with 1 if the subject of the corresponding tweet either explicitly or implicitly conveys a PERSONAL OPINION that yields a **POSITIVE** perception on migration to Europe or any specific European country (applies anywhere we mention Europe below). This positive perception can be interpreted as the following:

1. *The subject of the tweet either implicitly or explicitly portrays Europe or migration to Europe in a favorable/desirable way, as in they give a positive image about Europe.*
   Example:

   "عمرك ما تسمع عن القوانين دي في أوروبا وامريكا بل بالعكس بيشجعوا على الأنجاب ولما بيقل عندهم اعداد الشباب بيسحبوا كل الأيادي العامله المتميزه من دول العالم التالت بقوانين هجره مميزه عشان يفضلوا على القمه واحنا نفضل متخلفين"

   => "You never hear about these laws in Europe, nor in the United States. On the contrary, they encourage people to have children and when their number of youth decreases, they start pulling all of the special workforces from all over the third world with nifty migration laws in order to stay on top while we stay retarded".

2. *The subject of the tweet either implicitly or explicitly expresses a desire on migrating to Europe, or helps someone by giving advice on migration to Europe, or mentions that Arabs want to migrate to Europe because of its advantages (portrays Europe as a sought after/advantageous destination).*
   Example:

   "يعني شو بدنا بالقرف اولاد وحفازات اخر همي والله، هلأ هدفي الهجرة الى اوروبا"

   => "Why should we care about this disgust? I swear that I do not care at all about kids and diapers. My only concern is to migrate to Europe".

- Annotate with 0 if the subject of the corresponding tweet either explicitly or implicitly conveys a PERSONAL OPINION that yields a **NEGATIVE** perception on migration to Europe. This negative perception can be interpreted as the following:

1. *The subject of the tweet either implicitly or explicitly justifies Europe's problems by mentioning that it is because of migration and that migrants are "invading Europe" and that they are the reason behind these problems (portrays a negative image about Europe).*
   Example:

   "بإختصار شديد أوروبا عاوزه تخلص من موضوع اللاجئين تقوم تعمل لهم دوله على حدود مصر ويكونوا نواة للإرهاب وتطلقهم على مصر لتهديد أمنها وخلق توترات مع السودان الشقيقه بعد ما أوضاعها بدأت تستقر برعاية مصريه. إلعبوا غيرها"

=> "In a nutshell, Europe wants to end its migration crisis by creating a state for migrants on the borders of Egypt. Thus these refugees will become nuclei of terrorism and Europe will release them into Egypt to threaten its security and to create tensions with its neighbouring brotherly country Sudan, especially after the latter's situation started to become stable because of Egypt. Go play another game!".

2. *The subject of the tweet either implicitly or explicitly expresses an aversion to migration to Europe, or portrays Europe as a non favorable destination because of factors such as safety and income or because the subject mentions that Europe doesn't want any more migrants.*
Example:

"يريدونها كسوريا والعراق واليمن ـ ياشعب مصر حافظوا على الأمن وهي اكبر نعمة والا سوف تكتوون بنار الهجرة والغربة في بلدان اوروبا ... حافظوا على أمنكم ـ والجوع والفقر خير لكم من التهجير والضرب والسجن والقتل والدمار والخراب"

=> "They want it like Syria, Iraq and Yemen... Oh people of Egypt, preserve security which is the biggest blessing or else you will burn in the fire of migration to European countries... Preserve your security... Hunger and poverty are much better for you than displacement, brutality, imprisonment, murder, destruction and desolation".

3. *The subject of the tweet holds a grudge against Europe or expresses hate or disgust towards it or portrays migrants to Europe as lowly.*
Example:

"على الأقل محد احتل ارضي وهتك عرضي وشردني لاجئ في أوروبا وفوق هذا أنا كل صيف زي وفوق هذا امثالك وأمثال ابناء يخدم علي ويتلقى الأوامر مني مثل الخادم وينتظر الـ tip ينظف الطاولة من بعدي او الحمام عرفت الفرق يا مرتزق"

=> "At least no one occupied my land and stomped on my dignity and turned me into a homeless refugee in Europe. In addition to that, every summer people like you serve me and take orders from me like butlers and they wait for me to give them a tip. Then they clean the table and the bathroom once I leave. Now do you understand the difference you mercenary?".

- Annotate with -1 if the tweet is **UNRELATED**. Unrelated tweets can be interpreted as the following:

1. *The subject of the corresponding tweet is a news agency or someone reporting news, thus the tweet contains objective events.*
Example:

"المنظمة الدولية للهجرة: ارتفاع حصيلة القتلى الذين توفوا غرقا أثناء رحلات لطلب حق اللجوء إلى أوروبا عبر البحر المتوسط منذ مطلع العام وحتى السادس من أكتوبر الجاري إلى ١٠٧١ شخصا"

=> "International Organization for Migration (IMO): Death toll of drowning asylum seekers to Europe across the Mediterranean rises to 1071".

2. *The subject either implicitly or explicitly talks about someone else's opinion or reports an incident with objectivity.*
   Example:

   "#أوروبا خائفة من موجة لاجئين جديدة بسبب تهديد #تركيا لشمال شرق #سوريا"

   => "#Europe is scared of a new refugee wave because of #Turkey's threats to northeast #Syria".

3. *The subject is talking about an unrelated topic.*
   Example:

   "لنتفكر في خلق الله العظيم: حمامي أعربي هكذا يسمى في الكويت هو الصرد الرمادي الكبير Great grey shrike.
   له خط هجرة يمر بالكويت حيث تغطي خطوط هجرتها قارة اوروبا ومناطق شرق البحر المتوسط وشمال الخليج العربي"

   => "May we contemplate the creation of Almighty God: Arab Pigeon is its name in Kuwait, which is the Great Grey Shrike in English. Its migration line passes through Kuwait where it spans the continent of Europe and other regions East of the Mediterranean sea and North of the Arab Gulf".

4. *The tweet is neutral and doesn't convey any positive or negative perception on migration to Europe, or it is a joke that doesn't convey a specific stance.*
   Example:

   "ما عندي رأي محدد عن الهجرة إلى أوروبا"

   => "I don't have a specific opinion on migration to Europe".

Side Notes for Annotation:

- There is a huge number of tweets related to Turkey and the Turkish president, but since it is not part of the European Union, it is not to be considered in Europe in our study.
- If you have any doubt about the context of a tweet copy and paste it in Twitter to search for it and read the thread. Tweets with emoticons are often encoded with multiple "?" in spreadsheets, so remove the question marks before pasting the tweet to search for it on Twitter.

3.5.2 Cohen's Kappa Coefficient Calculation

Before Adjudication

The reliability and quality of annotations are very important factors when it comes to training a Machine Learning model to make accurate predictions. Knowing this, we followed the footsteps of [Refaee and Rieser, 2014] where we recruited two Arabic native speakers to annotate 300 identical tweets from our dataset.
Table 3.4 shows the annotations' matches (in white) and disagreements (in grey) between both annotators for all possible combinations of classes:

First Annotator	*Second Annotator*	*Total*
1	1	30
1	0	1
1	-1	4
0	1	3
0	0	41
0	-1	21
-1	1	11
-1	0	14
-1	-1	175
Total Disagreements		54

Table 3.4: Annotations with the Total Matches and Disagreements before Adjudication

After annotation, there were *54 total disagreements* out of a total of 300 tweets, amounting to a fraction of 18%. The calculation of the inter-annotator agreement using Cohen's Kappa Coefficient [Cohen, 1960] to assess the quality of the annotations immediately followed.
Our results yielded a Cohen's Kappa Coefficient $\kappa = 0.639$. According to [Landis and Koch, 1977], it is possible to assess κ according to the following rules:

- $\kappa < 0.00$: *Poor agreement*
- κ between 0.00 and 0.20: *Slight agreement*
- κ between 0.21 and 0.40: *Fair agreement*
- κ between 0.41 and 0.60: *Moderate agreement*
- κ between 0.61 and 0.80: *Substantial agreement*
- κ between 0.81 and 1.00: *Almost perfect agreement*

Therefore, since our $\kappa = 0.639$ was between 0.61 and 0.80, we had a substantial agreement. This was a good start but our results were subject to improvement using adjudication.

After Adjudication

Adjudication, as [Bamman, 2017] defined it: "is the process of deciding on a single annotation for a piece of text using information about the independent annotations". Therefore, we proceeded by re-reading the tweets with different annotations in order to decide which annotator's annotation was more accurate to choose. After we accomplished the re-reading process, we discovered some reasons behind the discrepancies such as:

- Different perceptions of the content of a tweet due to the complexity of the Arabic language that was used or due to the use of unfamiliar dialectal Arabic slang.
- Different perceptions of the content of a tweet due to a lack of knowledge of the context the tweet fell under.
- Different perceptions of the content of a tweet due to some ambiguity in the user's stance that didn't render the tweet fully comprehensible.

Table 3.5 shows some examples of tweets with both annotators' differing annotations, in addition to some viable reasons behind the differences:

Tweet	*Translation*	*First Annotator*	*Second Annotator*	*Adjudication Decision*
نعال المهاجرين السريين الذين ذابت أعمارهم في البحر الأبيض، والذين منعوا من دخول جنة أوروبا المشيدة من كنوز قارتهم والذين امتهن الأوربيون بيع أسلافهم قسرا ذات قرون.	The soles of the unidentified immigrants whose ages melted in the White Sea, and who were not allowed to enter the heaven of "Europe" that was built from the jewels of their continent and that Europeans forcibly sold their ancestors centuries ago	0	1	1
Reason	The second annotator's annotation prevailed because the word "heaven" was used to describe Europe, which means they are presenting it with a positive image			
الوضع بافريقيا زفت ومعظم الشباب عم يقبضوا معاشات متساوية مع معاش لبنان او اكتر شوي، الامر بينطبق على دول الخليج واكيد المهاجر على اوروبا مش عم يكب ملايين بلبنان	The situation in Africa sucks and most of the youth are getting identical salaries to Lebanon or a bit more. The same thing applies for Gulf countries, and of course immigrants in Europe and not pumping millions into Lebanon	0	-1	0
Reason	The first annotator's annotation prevailed because the user stated that the financial situation in Europe is horrible like it is in Africa, which yielded a negative image of Europe			

Table 3.5: Examples of Differing Annotations with the Reasons Behind the Adjudications

After adjudication, there were *27 total disagreements*, thus half the amount that existed before this phase. At that stage, we re-calculated Cohen's Kappa Coefficient to evaluate the robustness of our annotations and we achieved a new $\kappa = 0.825$. The value that we obtained signifies that we achieved an almost perfect agreement on these 300 tweets, which was a drastic improvement compared to the results we previously had. Table 3.6 shows the annotations' matches (in white) and disagreements (in grey) between both annotators for all possible combinations of classes after adjudication:

First Annotator	*Second Annotator*	*Total*
1	1	40
1	0	1
1	-1	1
0	1	1
0	0	50
0	-1	16
-1	1	3
-1	0	5
-1	-1	183
Total Disagreements		27

Table 3.6: Annotations with the Total Matches and Disagreements after Adjudication

Our results confirmed that the guidelines were clear and that annotators were consistently following them. The remainder of the dataset was annotated by the First Annotator and it consisted of *2715 tweets* by the 9^{th} crawl with the following distribution:

- *1582 tweets* were annotated with *-1* which consisted of ***58.27%*** of the entire dataset.
- *730 tweets* were annotated with *0* which consisted of ***26.89%*** of the entire dataset.
- *403 tweets* were annotated with *1* which consisted of ***14.84%*** of the entire dataset.

At that time, we were ready to start experimenting with different machine learning approaches.

Chapter 4

Machine Learning Approaches for Tweet Classification

4.1 Support-Vector Machines

Support-Vector Machines (SVM) [Cortes and Vapnik, 1995] are supervised linear classifiers that select the hyperplane that maximizes the separation margin between classes. They are referred to as large margin classifiers and their solution only depends on a small subset of training examples called support vectors. SVM are robust because they can easily be extended to non-linear separation using kernel machines [Passerini, 2019]. For running this Machine Learning algorithm as well as for the succeeding ones, we used the Scikit Learn Machine Learning library for Python [Pedregosa et al., 2011]. For word embeddings, we used Facebook's fastText library [Joulin et al., 2016a, Joulin et al., 2016b, Bojanowski et al., 2017] for its efficiency and its support of Arabic.

4.1.1 Preliminary Phase with Tweet Pre-processing

Shortly after our 9^{th} crawl, we had 2715 annotated tweets which was a sufficient amount to start experimenting with Machine Learning algorithms. In this approach as well as in the others to follow, we loaded the Arabic fastText embeddings model, then we split our dataset into: 80% for training (2172 tweets) and 20% for testing (543 tweets) as the code below shows:

```
print("Loading FT model")
model_ft = fasttext.load_model('/home/baalbaki/Desktop/FastText/cc.ar.300.bin')

try:
    from sklearn.model_selection import train_test_split
except ImportError:
    from sklearn.cross_validation import train_test_split

dataset = pd.read_csv(r'dataset.csv')
X, y = dataset.tweet, dataset.perception #X now holds the tweets and y holds the
    labels
training_tweets, testing_tweets, training_labels, testing_labels =
    train_test_split(X, y, test_size=0.2, random_state=42)
```

After splitting our dataset, we cleaned our tweets from the following 248 stop words which are not very useful for the learning process like "if", "so", "what" and others:

"إذ، إذا، إذما، إذن، أف، أقل، أكثر، ألا، إلا، التي، الذي، الذين، اللاتي، اللائي، اللتان، الذين، اللاتي، اللائي، اللتان، اللتيا، اللتين، اللذان، اللذين، اللواتي، إلى، إليك، إليكم، إليكما، إليكن، أم، أما، أما، إما، أن، إن، إنا، أنا، أنت، أنتم، أنتما، أنتن، إنما، إنه، أنى، أنى، آه، آها، أو، أولاء، أولئك، أوه، آي، أي، بماذا، أيها، إي، أين، أين، أينما، إيه، بخ، بس، بعد، بعض، بك، بكم، بكم، بكما، بكن، بل، بلى، بما، بمن، بنا، به، بها، بهم، بهما، بهن، بي، بين، بيد، تلك، تلكم، تلكما، ته، تي، تين، تينك، ثم، ثمة، حاشا، حبذا، حتى، حيث، حيثما، حين، خلا، دون، ذا، ذات، ذاك، ذان، ذانك، ذلك، ذلكم، ذلكما، ذلكن، ذه، ذو، ذوا، ذواتا، ذواتي، ذي، ذين، ذينك، ريث، سوف، سوى، شتان، عدا، عسى، عل، على، عليك، عليه، عما، عن، عند، غير، فإذا، فإن، فلا، فمن، في، فيم، فيما، فيه، فيها، قد، كأن، كأنما، كأي، كأين، كذا، كذلك، كل، كلا، كلاهما، كلتا، كلما، كليكما، كليهما، كم، كم، كما، كي، كيت، كيف، كيفما، لا، لاسيما، لدى، لست، لستم، لستما، لستن، لسن، لسنا، لعل، لك، لكم، لكما، لكن، لكنما، لكي، لكيلا، لم، لما، لن، لنا، له، لها، لهم، لهما، لهن، لو، لولا، لوما، لي، لئن، ليت، ليس، ليسا، ليست، ليستا، ليسوا، ما، ماذا، متى، مذ، مع، مما، ممن، من، منه، منها، منذ، مه، مهما، نحن، نحو، نعم، ها، هاتان، هاته، هاتي، هاتين، هاك، هاهنا، هذا، هذان، هذه، هذي، هذين، هكذا، هل، هلا، هم، هما، هن، هنا، هناك، هنالك، هو، هؤلاء، هي، هيا، هيت، هيهات، والذي، والذين، وإذ، وإذا، وإن، ولا، ولكن، ولو، وما، ومن، وهو، يا"

Once the tweets were cleaned from stop words, we used a fastText in-built function in order to extract the 300 dimensional vectors from the tweets. The following shows a tweet before and after being cleaned from stop words, along with its corresponding vector:

- **Tweet before being cleaned from stop words:**

  "ملف المهاجرين ورقة #رجب_طيب_أردوغان لابتزاز أوروبا كي تضخ المزيد من المساعدات
  "@DUMMYUSER 😡😡😡!!! https://t.co/a5c4tlPmRXSt

  => "The case of migrants is #Recep_Tayyip_Erdogan's card to blackmail Europe so that it provides more help https://t.co/a5c4tlPmRXSt !!!😡😡😡 @DUMMYUSER".

- **Cleaned stop words:**
  "كي، من"

- **Tweet after being cleaned from the aforementioned stop words:**

  "ملف المهاجرين ورقة #رجب_طيب_أردوغان لابتزاز أوروبا تضخ المزيد المساعدات
  "@DUMMYUSER 😡😡😡!!! https://t.co/a5c4tlPmRXSt

  => "The case of migrants is #Recep_Tayyip_Erdogan's card to blackmail Europe provides more help https://t.co/a5c4tlPmRXSt !!!😡😡😡 @DUMMYUSER".

- **Corresponding 300 dimensional vector after cleaning:**
  [0.0072241658344864845, ..., 0.0166148841381073]

The following code was responsible of cleaning the stop words and extracting the vectors from tweets:

```
def CleanStopWords(tweet):
      stop_words = stopwords.words('arabic')
      tweetSplitted = tweet.split(" ")
      tweet = [w for w in tweetSplitted if w not in stop_words]
      return(tweet)

def ExtractVectors(tweet, removeStopwords): #removeStopwords is a boolean
    variable
      tweet = tweet.rstrip()
      if (removeStopwords):
         tweet = CleanStopWords(tweet)
         tweet = ' '.join(tweet)
      vector = model_ft.get_sentence_vector(tweet)
      return(vector)
```

Once we were done with the aforementioned task, we used Scikit Learn's SVC to train our Machine Learning model using our training tweets and training labels. Our classifier "clf" took 3 parameters:

- **C:** which is the regularization parameter. It controls the trade-off between a smooth decision boundary and classifying training points correctly. It is always a good practice to have a lower value for C in order to avoid overfitting [Hawkins, 2004].
- **Kernel:** which is a function that computes inner products in feature space. It is usual for feature spaces to be too simple and consequently, linear division of data is not possible. Thus, kernels solve this problem by mapping features to a higher-dimensional feature space where linear data separation becomes possible.
- **Gamma:** which is the kernel coefficient that defines how far the influence of a single training example reaches. If gamma has a high value this means that each training example has a close reach. This way, our model would fail to generalize since it would be prone to excessively adjust to training examples. This is a behavior one would like to avoid and thus the lower the gamma, the better.

The snippet of code below denotes our choices for all of these 3 parameters:

```
clf = SVC(C=10, kernel='rbf', gamma=0.02)
clf.fit(training_tweets_vectors, training_labels_array)
predicted_labels = clf.predict(testing_tweets_vectors)
```

After the training process was done, we ran our classifier and got the results shown in table 4.1:

Class	Precision	Recall	F1-score	Support
-1	0.58	1.00	0.74	316
0	0.00	0.00	0.00	146
1	0.00	0.00	0.00	81
Accuracy			0.58	543
Macro Average	0.19	0.33	0.25	543
Weighted Average	0.34	0.58	0.43	543
Model Accuracy	***0.58195211786372***			

Table 4.1: Report of the Results of the Preliminary SVM Approach

In order to assess the quality of our results, it is imperative to understand the meaning of: precision, recall, f1-score and support. We explained them by referring to Scikit Learn's documentation [Pedregosa et al., 2011]:

- **Precision:** the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives. The precision is the ability of a classifier not to falsely label a sample.
- **Recall:** the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is the ability of the classifier to find all positive samples.
- **F1-score:** the weighted harmonic mean of precision and recall, where an F-beta score reaches its best value at 1 and worst score at 0.
- **Support:** the number of occurrences of each class in the correct labels.

As shown in table 4.1, classification for class *-1* achieved a fair precision of *0.58* and a perfect recall of *1.00*. But it did not recognize both classes *0* and *1* since the values of precision and recall were both *0.00*. The final model accuracy of *0.58195211786372* was actually the majority baseline, meaning that our model only classified the majority class *-1* which consisted of around *58%* of our entire dataset. This meant that our model did not learn anything for classes *0* and *1*. This was probably due to the random choice of splits for our training and test sets as well as the random choice for the parameters of our classifier. Our results could have been much better had we performed model selection. Therefore, we opted to search for the best hyperparameters (which are parameters with set values before the learning process) and the best combination of training and test sets using K-Fold Cross-Validation [Mosteller and Tukey, 1968] and Grid Search Cross-Validation [Pedregosa et al., 2011].

4.1.2 K-Fold Cross-Validation

As mentioned above, a viable explanation for achieving such a low accuracy could be that the splitting choice for our training and test sets was not optimal. Our selection was not very reliable as there might have been other splits that would have given more reliable results. As the accuracy that can be obtained with one combination of training and test sets can be better than another combination, we tested our model on one random split and we got better results than our baseline. Therefore, we opted to use K-Fold Cross Validation [Mosteller and Tukey, 1968] to automatically find out the best split possible and use it. K-Fold Cross Validation divides the data into folds and evaluates all the possible combinations of training and test sets. This ensures that each fold is used as a testing set at some point. Figure 4.1 below provides a visual representation of the K-Fold Cross-Validation approach.

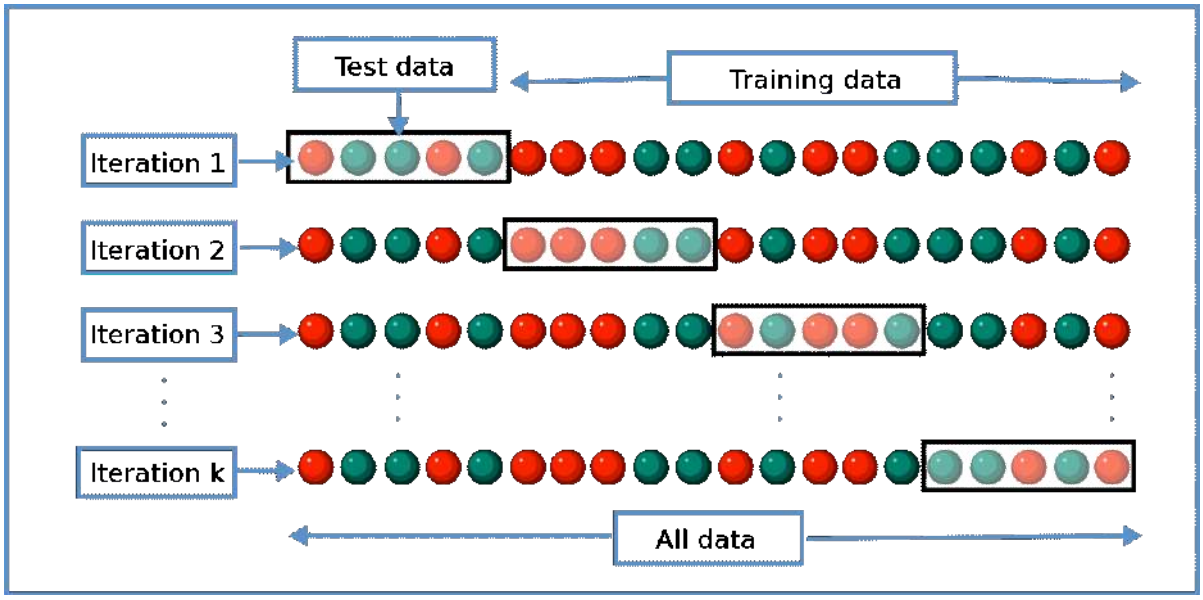


Figure 4.1: Visual Representation of the K-Fold Cross-Validation Approach [Wikipedia, 2019]

In order to perform K-Fold Cross-Validation, we used Scikit Learn's built-in KFold module as shown in the snippet of code below:

```
try:
    from sklearn.model_selection import KFold, cross_val_score
    legacy = False
except ImportError:
    from sklearn.cross_validation import KFold, cross_val_score
    legacy = True

if legacy:
    kf = KFold(len(training_labels),n_folds=10, shuffle=True, random_state=42)
else:
    kf = KFold(n_splits=10, shuffle=True, random_state=42)
```

Since we wanted to perform 10 splits, we set the number of splits to 10 in order to perform what is called: 10-Fold Cross-Validation. We iterated on each gamma, trained our classifier with it, computed cross-validated accuracy scores and stored them for each fold. At the end of our iterations, we performed hyperparameter optimization by getting the gamma with the highest mean accuracy and we trained over the full training set with the best gamma. Last but not least, we predicted on the test set in order to compute the accuracy of our model after Cross-Validation. The code below was responsible for this entire process:

```
gamma_values = [0.1, 0.05, 0.02, 0.01]
accuracy_scores = []

for gamma in gamma_values:

    clf = SVC(C=10, kernel='rbf', gamma=gamma)

    if legacy:
        scores = cross_val_score(clf, training_tweets_vectors,
            training_labels_array, cv=kf, scoring='accuracy')
```

```
    else:
        scores = cross_val_score(clf, training_tweets_vectors,
            training_labels_array, cv=kf.split(training_tweets_vectors),
            scoring='accuracy')

    accuracy_score = scores.mean()
    accuracy_scores.append(accuracy_score)

best_index = np.array(accuracy_scores).argmax()
best_gamma = gamma_values[best_index]

clf = SVC(C=10, kernel='rbf', gamma=best_gamma)
clf.fit(training_tweets_vectors, training_labels_array)

predicted_labels = clf.predict(testing_tweets_vectors)
```

After getting that the best gamma was equal to 0.1, we ran our classifier to predict on the test set with it. The results of our classifier are shown in table 4.2:

Class	**Precision**	**Recall**	**F1-score**	**Support**
-1	0.58	1.00	0.74	316
0	0.67	0.01	0.03	146
1	0.00	0.00	0.00	81
Accuracy			0.58	543
Macro Average	0.42	0.34	0.25	543
Weighted Average	0.52	0.58	0.44	543
Model Accuracy	***0.583793738489871***			

Table 4.2: Report of the Results of the K-Fold Cross-Validation Approach

Although the model's overall accuracy remained quite the same and did not really improve after Cross-Validation, the precision for class *0* significantly improved from *0.00* to *0.67*. On the other hand recall scored *0.01* which means that it required much more improvement. Nevertheless, these results meant that the model started to slowly learn for the aforementioned class, which was an important accomplishment but still not enough. Our poor results might have been due to our random choice of hyperparameters which is not a favorable practice. Therefore, we chose to apply Hyperparameter Tuning using Grid Search Cross-Validation [Pedregosa et al., 2011] in order to find the best parameters to train our model with.

4.1.3 Hyperparameter Tuning using Grid Search Cross-Validation

Grid Search Cross-Validation is a method to perform hyperparameter tuning, as in to find the best combination of hyperparameters. These are parameters that are passed to the Machine Learning model and that it doesn't learn by itself, such as: C, Gamma, Kernel. Each combination of hyperparameters represents a Machine Learning model, thus different combinations yield different models. The goal of Grid Search Cross-Validation is to train each of these models on the full training set, find the best hyperparameters and evaluate them using Cross-Validation on the full testing set that was not used during model selection. At the end, the best-performing model gets selected. In our case, we decided to choose 4 different values of C, 6 different values of Gamma and 3 different values for the Kernel as possible hyperparameters. The following code includes our use of Scikit Learn's default GridSearchCV library with the aforementioned possible hyperparameters:

```
try:
    from sklearn.model_selection import GridSearchCV
except ImportError:
    from sklearn.grid_search import GridSearchCV

possible_parameters = {
    'C': [1e0, 1e1, 1e2, 1e3],
    'gamma': [0.0001,0.001,0.05,0.02,0.01,0.1],
    'kernel': ['linear', 'rbf','poly']
}

clf = GridSearchCV(SVC(), possible_parameters, n_jobs=4, cv=3)
clf.fit(training_tweets_vectors, training_labels_array)
print("Best parameters: ",clf.best_params_)
predicted_labels = clf.predict(testing_tweets_vectors)
```

The best hyperparameters that we got were:
Gamma = 0.01, C = 1000.0, Kernel = rbf
Since GridSearchCV automatically sets the best hyperparameters to the classifier, we simply ran it to predict on the test set. The results we got are denoted in table 4.3:

Class	Precision	Recall	F1-score	Support
-1	0.71	0.82	0.76	316
0	0.48	0.39	0.43	146
1	0.50	0.37	0.43	81
Accuracy			0.64	543
Macro Average	0.56	0.53	0.54	543
Weighted Average	0.62	0.64	0.62	543
Model Accuracy	***0.6372007366482505***			

Table 4.3: Report of the Results of the Grid Search Cross-Validation Approach

Our overall accuracy improved by an impressive factor of *0.05* compared to the previous sole approach of Cross-Validation. Diving more into specifics, despite the fact that for class *0* precision dropped from *0.67* to *0.48*, recall improved from *0.01* to *0.39*. Last but not least, there was a drastic improvement for both precision and recall for class *1*, as they rose from *0.00* each to *0.50* for precision and *0.37* for recall respectively. Even though our results carried substantial improvements, we wanted to see if cleaning our tweets more would render higher quality results.

4.1.4 Further Tweet Pre-processing

After thorough reflection, the components that we decided to remove were: URLs, user mentions, punctuations, English numbers, Arabic numbers and emojis. As they are not very useful for our learning process, having vectors for them would have evoked noise in our data which is something that should be avoided. The only component that could have been useful for learning was emojis, but since most of the times they were encoded as "?" in spreadsheets, we decided to completely remove them for consistency. The following list contains all of the regular expressions or code that we used in order to clean the tweets from the aforementioned components:

- **URLs:**
  `http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+`

- **User Mentions:**
  `(?<=^|(?<=[^a-zA-Z0-9-\.]))@([A-Za-z0-9_]+)`
- **Punctuations:**

```
punctuations = ['!','$','%','^','&','*'] #there are much more but we
    only showed these for brevity
for i in punctuations:
    tweet = tweet.replace(i, '')
```

- **English Numbers:**
  `[0-9]+`
- **Arabic Numbers:**

```
arabic_numbers=[٩،٨،٧،٦،٥،٤،٣،٢،١،٠]
for i in arabic_numbers:
    tweet = tweet.replace(i, '')
```

- **Emojis:**

```
from emoji import UNICODE_EMOJI #default emoji library that contains a
    dictionary of a wide range of emojis

def CleanEmojis(sentence):
  emojiList=list(UNICODE_EMOJI.keys())
  for emojiIndex in range(len(emojiList)):
     for sentenceIndex in range(len(sentence)):
        if emojiList[emojiIndex] in sentence[sentenceIndex]:
           sentence[sentenceIndex] =
               sentence[sentenceIndex].replace(emojiList[emojiIndex],'')
```

The example below clearly shows the tweet that we mentioned earlier in the study before and after being cleaned from the new components, along with its corresponding vector:

- **Tweet before being cleaned from the new components:**

"ملف المهاجرين ورقة #رجب ـطيب ـأردوغان لابتزاز أوروبا تضخ المزيد المساعدات
"@DUMMYUSER 😟😟😟!!! https://t.co/a5c4tlPmRXSt

=> "The case of migrants is #Recep_Tayyip_Erdogan's card to blackmail Europe provides more help https://t.co/a5c4tlPmRXSt !!!😟😟😟 @DUMMYUSER".

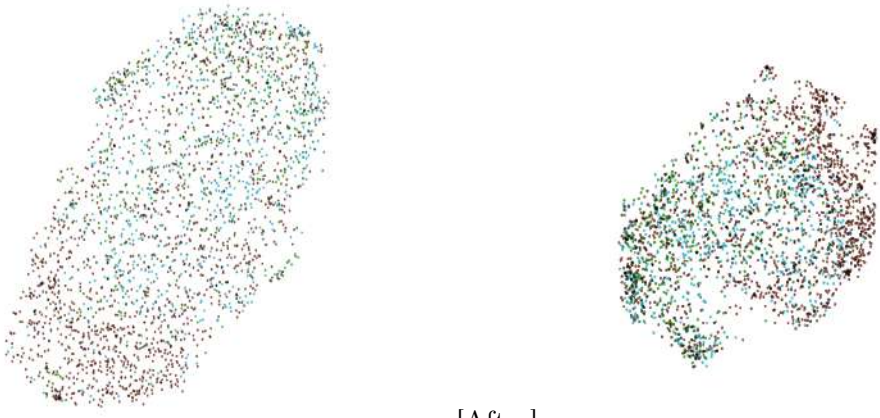
- **Cleaned components:**
  "https://t.co/a5c4tlPmRXSt, @DUMMYUSER, !!!, 😟😟😟 "
- **Tweet after being cleaned from the new components:**

"ملف المهاجرين ورقة #رجب ـطيب ـأردوغان لابتزاز أوروبا تضخ المزيد المساعدات"

=> "The case of migrants is #Recep_Tayyip_Erdogan's card to blackmail Europe provides more help"

- **Corresponding 300 dimensional vector after cleaning:**
  $[0.0063792357907902112, ..., 0.0233562108974325]$

The significance of the cleaning procedure is visually represented in figure 4.2, where the vectors of the 2715 tweets are shown before and after cleaning them from the new components:



[Before] [After]

Figure 4.2: Vectorial Representations of the Tweets Before and After Cleaning: Red is *-1*, Blue is *0* and Green is *1*

It is evident that before cleaning, the different classes were more mixed and there was higher dispersion among classes themselves. On the other hand after cleaning, each class was more compact and the overall vectors were less dispersed. This is visual proof that cleaning the data was very useful to reduce noise. At that stage we were ready to retry K-Fold Cross-Validation all while finding the best hyperparameters, in addition to Hyperparameter Tuning using Grid Search Cross-Validation.

4.1.5 K-Fold Cross-Validation

In our previous K-Fold Cross-Validation tentative, we had been only changing the gamma hyperparameter. In order to find a more optimal solution, we decided to try 72 different combinations of hyperparameters, thus we also chose values for both C and the Kernel. The following code explains this process:

```
try:
    from sklearn.model_selection import KFold, cross_val_score
    legacy = False
except ImportError:
    from sklearn.cross_validation import KFold, cross_val_score
    legacy = True

if legacy:
    kf = KFold(len(training_labels),n_folds=10, shuffle=True, random_state=42)
else:
    kf = KFold(n_splits=10, shuffle=True, random_state=42)

gamma_values = [0.0001,0.001,0.05,0.02,0.01,0.1]
C_values = [1e0, 1e1, 1e2, 1e3]
```

```
kernel_values = ['linear', 'rbf','poly']
accuracy_scores = []
the_gammas=[]
the_kernels=[]
the_Cs=[]

for gamma in gamma_values:
   for C in C_values:
      for kernel in kernel_values:
         clf = SVC(C=C, kernel=kernel, gamma=gamma)
         if legacy:
               scores = cross_val_score(clf, training_tweets_vectors,
                   training_labels_array, cv=kf, scoring='accuracy')
         else:
                 scores = cross_val_score(clf, training_tweets_vectors,
                     training_labels_array, cv=kf.split(training_tweets_vectors),
                     scoring='accuracy')
         the_gammas.append(gamma)
         the_kernels.append(kernel)
         the_Cs.append(C)
         accuracy_score = scores.mean()
         accuracy_scores.append(accuracy_score)

best_index = np.array(accuracy_scores).argmax()
best_gamma = the_gammas[best_index]
best_C = the_Cs[best_index]
best_kernel = the_kernels[best_index]
print("The best combination of hyperparameters is:
    Gamma=",str(best_gamma),"\t","C=",str(best_C),"\t","Kernel=",str(best_kernel))
```

After looping on all of the possible combinations, we got that the best hyperparameters were:
Gamma = 0.05, C = 100.0, Kernel = rbf
Thus we set our classifier with these hyperparameters and ran it on the test set as shown below:

```
clf = SVC(C=best_C, kernel=best_kernel, gamma=best_gamma)
clf.fit(training_tweets_vectors, training_labels_array)

predicted_labels = clf.predict(testing_tweets_vectors)
```

The results we got are denoted in the following table 4.4:

Class	**Precision**	**Recall**	**F1-score**	**Support**
-1	0.68	0.84	0.75	316
0	0.50	0.35	0.41	146
1	0.51	0.32	0.39	81
Accuracy			0.63	543
Macro Average	0.56	0.50	0.52	543
Weighted Average	0.61	0.63	0.61	543
Model Accuracy	***0.6298342541436464***			

Table 4.4: Report of the Results of the Secondary K-Fold Cross-Validation Approach with the Best Hyperparameters

This model performed better than our preliminary Cross-Validation approach. The F1-score of class *0* rose from *0.03* to *0.41* and in class *1* from *0.00* to *0.39*. These were great indicators that our model actually started learning. Nevertheless, we decided to verify how well would Hyperparameter Tuning using Grid Search Cross-Validation perform this time.

4.1.6 Hyperparameter Tuning using Grid Search Cross-Validation

As there were no major changes in the Grid Search Cross-Validation part except for tweet cleaning, we directly proceeded by showing our results for this sub-section. The best hyperparameters we got were:
Gamma = 0.1, C = 100.0, Kernel = rbf
After repeating the routine procedure of running our classifier on the test set, we got the following results shown in table 4.5

Class	Precision	Recall	F1-score	Support
-1	0.70	0.81	0.75	316
0	0.50	0.39	0.44	146
1	0.52	0.40	0.45	81
Accuracy			0.64	543
Macro Average	0.57	0.53	0.55	543
Weighted Average	0.62	0.64	0.62	543
Model Accuracy	***0.6372007366482505***			

Table 4.5: Report of the Results of the Secondary Grid Search Cross-Validation Approach

We got the exact same overall model accuracy score of *0.6372007366482505* like the first Grid Search Cross-Validation approach. This signified that tweet cleaning did not come in handy for this method. Therefore, we decided to try to see if splitting hashtags, in addition to having more annotated tweets would increase this accuracy as well as the K-Fold Cross-Validation one.

4.1.7 Splitting Hashtags

Shortly after our 10^{th} crawl, we gained 260 tweets that we annotated, thus remaining with a total of 2975 tweets. 2380 tweets of them were used for training and the rest 595 tweets were used for testing. In addition to that, we split hashtags by removing the "#" sign and by replacing "_" with spaces in order to separate each word in the hashtags. This procedure is shown in the snippet of code below:

```
tweet = tweet.replace('#', '')
tweet = tweet.replace('_', ' ')
```

The following example clearly shows the tweet we had previously mentioned before and after having the hashtag split, as well as its corresponding vector:

- **Tweet before splitting the hashtag:**

  "ملف المهاجرين ورقة #رجب_طيب_أردوغان لابتزاز أوروبا تضخ المزيد المساعدات"

  => "The case of migrants is #Recep_Tayyip_Erdogan's card to blackmail Europe provides more help"

- **Split hashtag:**
  "#رجب_طيب_أردوغان"
  => "#Recep_Tayyip_Erdogan"

- **Tweet after splitting the hashtag:**

  "ملف المهاجرين ورقة رجب طيب أردوغان لابتزاز أوروبا تضخ المزيد المساعدات"

  => "The case of migrants is Recep Tayyip Erdogan's card to blackmail Europe provides more help"

- **Corresponding 300 dimensional vector after splitting the hashtag:**
  $[0.008732566783111098 6, ..., 0.0321165562703568]$

4.1.8 K-Fold Cross-Validation

After the cleaning process was done, we ran our classifier in addition to K-Fold Cross-Validation and we got the following best hyperparameters:
Gamma = 0.05, C = 100.0, Kernel = rbf
The detailed report of our results is shown in table 4.6:

Class	Precision	Recall	F1-score	Support
-1	0.66	0.87	0.75	327
0	0.57	0.40	0.47	174
1	0.50	0.24	0.33	94
Accuracy			0.63	595
Macro Average	0.58	0.50	0.52	595
Weighted Average	0.61	0.63	0.60	595
Model Accuracy	***0.6302521008403361***			

Table 4.6: Report of the Results of the Tertiary K-Fold Cross-Validation Approach with the Best Hyperparameters and with Split Hashtags

There were some negligible fluctuations in precision and recall for all classes in comparison to the secondary K-Fold Cross-Validation approach and the overall model accuracy remained quite the same, hovering around *0.63*. Thus the addition of 260 tweets and splitting the hashtags did not really improve our model accuracy for the K-Fold Cross-Validation approach. Therefore, we resorted to see if it was going to be the same issue with Hyperparameter Tuning using Grid Search Cross-Validation.

4.1.9 Hyperparameter Tuning using Grid Search Cross-Validation

Using the new and more populated dataset with split hashtags, we ran our classifier with the Grid Search Cross-Validation approach. The best hyperparameters we got were:
Gamma = 0.1, C = 100, Kernel = rbf
Our full results are shown in table 4.7:

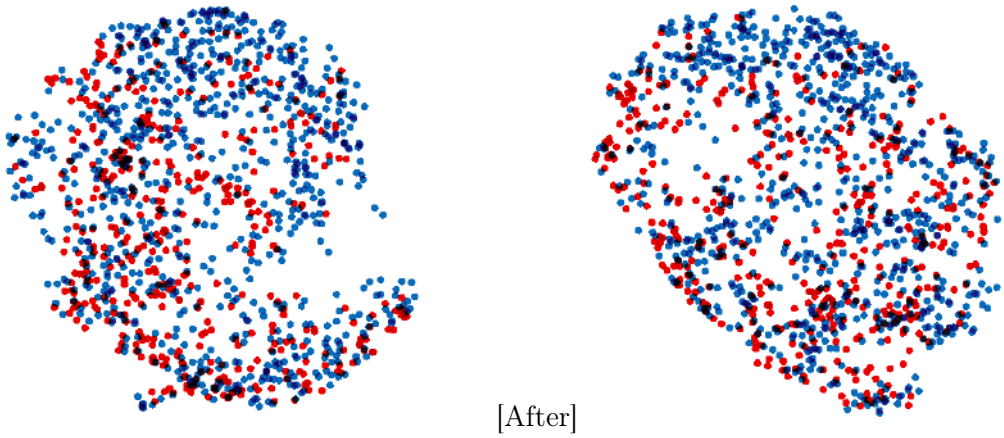
Class	Precision	Recall	F1-score	Support
-1	0.67	0.84	0.75	327
0	0.58	0.44	0.50	174
1	0.49	0.29	0.36	94
Accuracy			0.63	595
Macro Average	0.58	0.52	0.53	595
Weighted Average	0.62	0.63	0.61	595
Model Accuracy	***0.6336134453781512***			

Table 4.7: Report of the Results of the Tertiary Grid Search Cross-Validation Approach with Split Hashtags

Like its previous counterpart, this approach's accuracy remained quite the same, also hovering around *0.63*. The impact that splitting hashtags had on our models was minor. Therefore, we decided to model the classification task in a different way by running our classifier on the 2 classes: *0* and *1* in order to determine if we will get a higher accuracy in binary classification.

4.1.10 K-Fold Cross-Validation with 2 Classes: *0* and *1*

We removed all tweets annotated with *-1* and we were left with a total of 1240 tweets where 808 tweets were annotated with *0* and 432 tweets were annotated with *1*. In order to get a clearer idea on how classes *0* and *1* were dispersed, we decided to show their visual representations before and after cleaning in figure 4.3 below:



[Before] [After]

Figure 4.3: Vectorial Representations of the Tweets Before and After Cleaning: Blue is *0* and Red is *1*

The figure above shows the positive effect of cleaning as the noise in data was reduced. After accomplishing this step, we ran the same K-Fold Cross-Validation technique we had run in the previous subsections. We found that the best hyperparameters were:
Gamma = 0.02, C = 1000.0, Kernel = rbf
The detailed results are reported in table 4.8:

Class	Precision	Recall	F1-score	Support
0	0.78	0.90	0.84	168
1	0.69	0.47	0.56	80
Accuracy			0.76	248
Macro Average	0.74	0.69	0.70	248
Weighted Average	0.75	0.76	0.75	248
Model Accuracy	***0.762096774193548***			

Table 4.8: Report of the Results of the Quaternary K-Fold Cross-Validation Approach with the Best Hyperparameters for Classes: *0* and *1*

We achieved satisfactory results with an overall classification accuracy of *0.762096774193548*. This meant that approximately three quarters of the tweets have been classified correctly. At that point we wanted to see if Grid Search Cross-Validation would perform better.

4.1.11 Hyperparameter Tuning using Grid Search Cross-Validation with 2 Classes: *0* and *1*

We ran the previously explained Grid Search Cross-Validation technique on the dataset with the 2 classes: *0* and *1*, and we were left with the following best hyperparameters:
Gamma = 0.1, C = 100.0, Kernel = rbf
Our detailed results are shown in the following table 4.9:

Class	Precision	Recall	F1-score	Support
0	0.78	0.92	0.84	168
1	0.72	0.45	0.55	80
Accuracy			0.77	248
Macro Average	0.75	0.68	0.70	248
Weighted Average	0.76	0.77	0.75	248
Model Accuracy	***0.7661290322580645***			

Table 4.9: Report of the Results of the Quaternary Grid Search Cross-Validation Approach for Classes: *0* and *1*

Hyperparameter Tuning using Grid Search Cross-Validation performed quite well as its K-Fold Cross-Validation counterpart, with an overall accuracy of *0.7661290322580645*. This approach with 2 classes led us to the idea explained in the following sub-section.

4.1.12 The 2 Classifiers Approach

As proven in the previous section, it is evident that the 2 classes approach yielded a much higher accuracy for our classification task. Therefore, we thought to extend this idea into including 2 binary classifiers that operated on two different datasets:

- **Classifier 1:** Operated on dataset 1 (2975 tweets) that contained 2 classes:
  -1: Which was the class that represented the unrelated tweets (1735 tweets).
  2: Which was a merger of class *0* and class *1*. It represented related tweets (1240 tweets).

- **Classifier 2:** Operated on dataset 2 (1240 tweets) that contained 2 classes:
  0: Which was the class that represented the tweets with a negative perception on

migration to Europe (808 tweets).
1: Which was the class that represented the tweets with a positive perception on migration to Europe (432 tweets).

The idea was to run the first classifier on "dataset 1" and then store the tweets whose predicted class is *2* in order to feed them for training the second classifier. After the latter has been trained with the majority of them, a small proportion was to be used as testing with the correct labels being the ones present in "dataset 2". This way, we would have benefited from binary classification, all while classifying the 3 classes. This approach is explained with more clarity in figure 4.4:

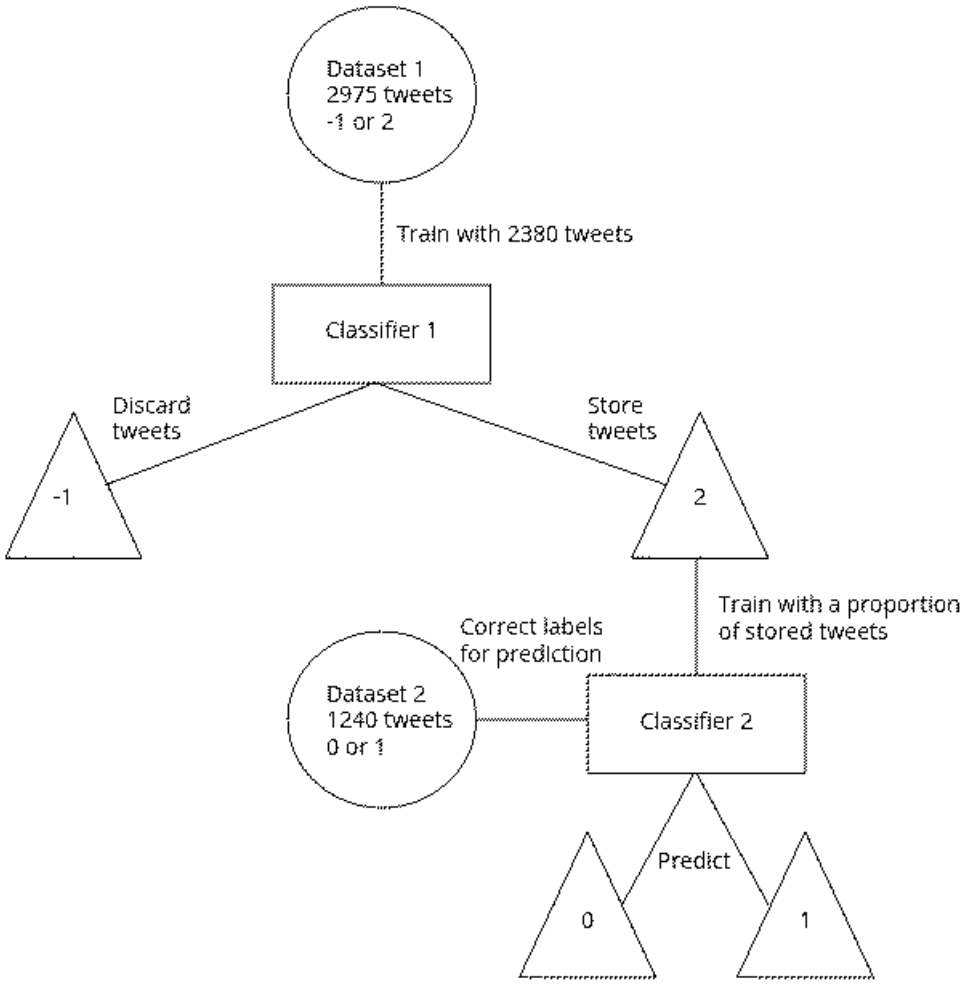
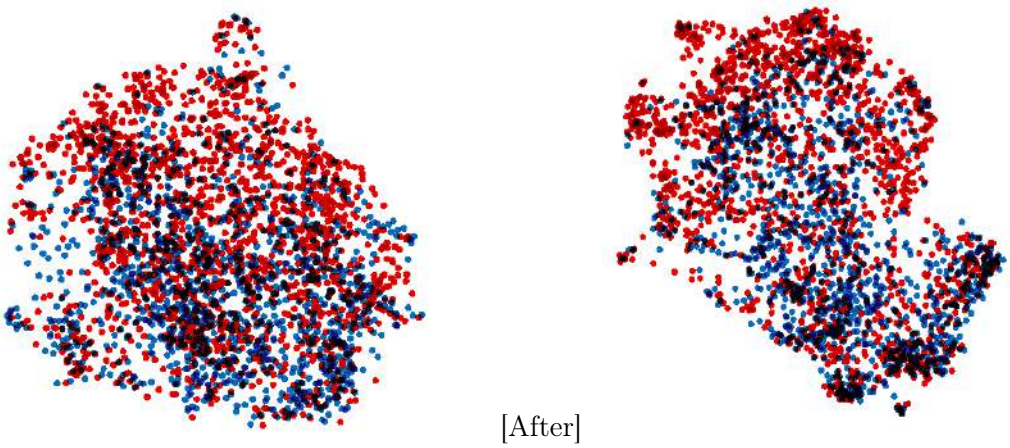


Figure 4.4: Visual Representation of the 2 Classifiers Approach

In order to understand the distribution of the vectors of the unrelated class *-1* and the related class *2* in vector space, we resorted to see the visual representations of their vectors before and after cleaning. They are shown in figure 4.5:



[Before] [After]

Figure 4.5: Vectorial Representations of the Tweets Before and After Cleaning: Red is *-1* and Blue is *2*

The effect of tweet cleaning was extremely evident. Before cleaning we can see that the tweets of the *-1* class in red were spread out across all of the vector space. While after cleaning they became more concentrated in the top and the tweets of class *2* in blue remained concentrated in the center and in the bottom. Since this increased our chances of getting a higher accuracy score, we decided to run both K-Fold Cross-Validation and Grid Search Cross-Validation for this binary classification task to see which one would outperform the other.

K-Fold Cross-Validation with 2 Classes: *-1* and *2*

We found that the best hyperparameters for this approach were:
Gamma = 0.0001, C = 10.0, Kernel = linear
The detailed results are reported in table 4.10:

Class	Precision	Recall	F1-score	Support
-1	0.72	0.75	0.74	327
2	0.68	0.64	0.66	268
Accuracy			0.70	595
Macro Average	0.70	0.70	0.70	595
Weighted Average	0.70	0.70	0.70	595
Model Accuracy	***0.7025210084033613***			

Table 4.10: Report of the Results of the Quintenary K-Fold Cross-Validation Approach for Classes: *-1* and *2*

The results we got were fair but not good enough, as *0.7* would be the upper-bound that "classifier 2" would reach which was not favorable for us. Thus we tried Hyperparameter Tuning using Grid Search Cross-Validation to see if we would get better results.

Hyperparameter Tuning using Grid Search Cross-Validation with 2 Classes: *-1* and *2*

We found that the best hyperparameters for this approach were:
Gamma = 0.05, C = 100.0, Kernel = rbf
The detailed results are reported in table 4.11:

Class	Precision	Recall	F1-score	Support
-1	0.71	0.75	0.73	327
2	0.67	0.63	0.65	268
Accuracy			0.70	595
Macro Average	0.69	0.69	0.69	595
Weighted Average	0.70	0.70	0.70	595
Model Accuracy	***0.6974789915966386***			

Table 4.11: Report of the Results of the Quintenary Grid Search Cross-Validation Approach for Classes: *-1* and *2*

Nearly identical results of the previous K-Fold Cross-Validation approach occurred, thus we realized that with a small amount of data to train "classifier 2" with, we will not be getting quality results. Therefore, we dropped this approach and we suggest experimenting with it only if there are around *3000 tweets* to train "classifier 2" with. We opted to crawl for 552 more tweets and we retried K-Fold Cross-Validation and Hyperparameter Tuning using Grid Search Cross-Validation for the multiclass classification task on our final dataset of *3527 tweets*.

4.1.13 K-Fold Cross-Validation

We used the same values for the previously mentioned hyperparameters and the best combination we got was:
Gamma = 0.1, C = 100.0, Kernel = rbf
Thus, we set our SVM model with these hyperparameters and then we ran it on the test set to get the following results shown in table 4.12:

Class	Precision	Recall	F1-score	Support
-1	0.64	0.85	0.73	381
0	0.55	0.40	0.47	210
1	0.59	0.23	0.33	115
Accuracy			0.61	706
Macro Average	0.59	0.49	0.51	706
Weighted Average	0.60	0.61	0.58	706
Model Accuracy	***0.6147308781869688***			

Table 4.12: Report of the Results of the K-Fold Cross-Validation Approach with the Best Hyperparameters

The SVM model achieved an accuracy of *0.6147308781869688* for the multiclass classification task, which is higher than the baseline by 3%. Precision for classes *-1, 0 and 1* were *0.64, 0.55 and 0.59* respectively, all above average value. While for recall there were more fluctuations, with the values being *0.85, 0.40 and 0.23* for all classes respectively. The following figure 4.6 is a visual representation of the learning process and it shows how much learning improved after K-Fold Cross-Validation on the *3527 tweets*:

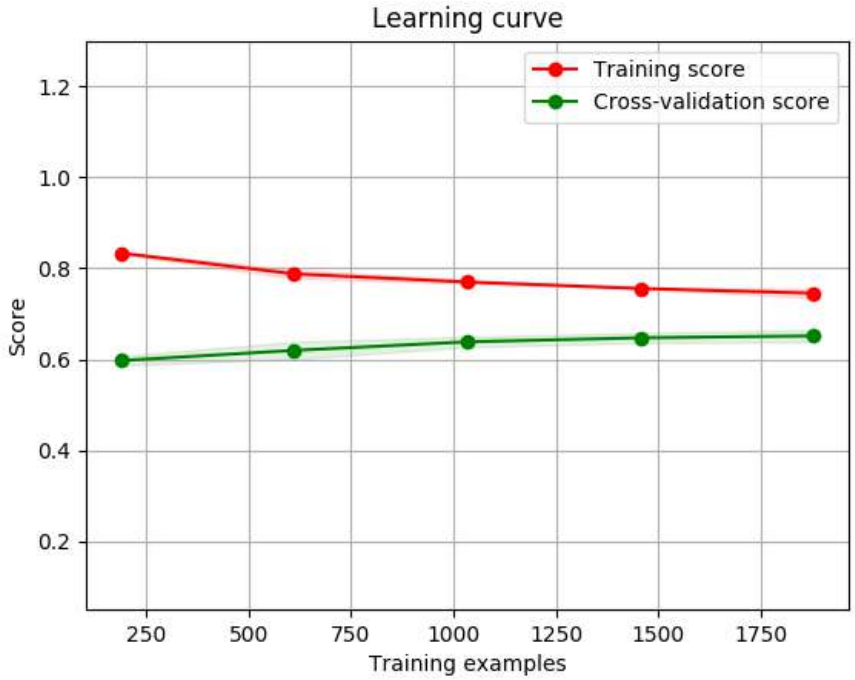


Figure 4.6: Visual Representation of the Learning Curve after K-Fold Cross-Validation

After that approach, we opted to see if Hyperparameter Tuning using Grid Search Cross-Validation would perform better.

4.1.14 Hyperparameter Tuning using Grid Search Cross-Validation

The best hyperparameters for this approach were the same as the K-Fold Cross-Validation one:
Gamma = 0.1, C = 100.0, Kernel = rbf
The full results we got are shown in the following table 4.13

Class	Precision	Recall	F1-score	Support
-1	0.64	0.85	0.73	381
0	0.55	0.40	0.47	210
1	0.59	0.23	0.33	115
Accuracy			0.61	706
Macro Average	0.59	0.49	0.51	706
Weighted Average	0.60	0.61	0.58	706
Model Accuracy	***0.6147308781869688***			

Table 4.13: Report of the Results of the Grid Search Cross-Validation Approach

Overall, we were satisfied with our results since the overall accuracy was greater than the majority baseline with above average values for precision. After exhausting our attempts for the SVM classifier, it was a suitable time to start experimenting with other Machine Learning algorithms to see if they would outperform it.

4.2 Naïve Bayes

Naïve Bayes [Webb, 2010] is a generative Machine Learning model [Ng and Jordan, 2002] which utilizes Bayes' Theorem [Joyce, 2003] in order to compute the conditional probabilities of features, all while assuming that they are strongly conditionally independent. Bayes' Theorem is shown in the following equation 4.1:

$$P(y|\mathbf{X}) = \frac{P(\mathbf{X}|y)}{P(\mathbf{X})}P(y) \tag{4.1}$$

- y is the class and $\mathbf{X}$ is the vector of features.
- P(y|$\mathbf{X}$) is the posterior probability.
- P($\mathbf{X}$|y) is the maximum likelihood.
- P(y) is the prior probability.
- P($\mathbf{X}$) is the evidence.

For this approach, as well as all of the approaches that followed, we ran the exact same pre-processing we used in the previous SVM approach in order to clean the tweets. We used the Arabic fastText model but this time we paired it up with Scikit Learn's Gaussian Naïve Bayes model instead of the SVM one. Since the Gaussian Naïve Bayes model doesn't have hyperparameters, it is not possible to perform Hyperparameter Tuning and we simply fit our data to the model and ran it to predict the labels as shown in the snippet of code below:

```
from sklearn.naive_bayes import GaussianNB
#Create a Gaussian NB Classifier
gnb = GaussianNB()
#Train the model using the training set
gnb.fit(training_tweets_vectors, training_labels_array)
#Predict the results on the testing set
predicted_labels = gnb.predict(testing_tweets_vectors)
```

The results we got are shown in the following table 4.14

Class	Precision	Recall	F1-score	Support
-1	0.69	0.49	0.57	381
0	0.39	0.53	0.45	210
1	0.35	0.46	0.40	115
Accuracy			0.50	706
Macro Average	0.48	0.49	0.47	706
Weighted Average	0.55	0.50	0.51	706
Model Accuracy	***0.49575070821529743***			

Table 4.14: Report of the Results of the Gaussian Naïve Bayes Approach

We got an overall accuracy of *0.49575070821529743* which was quite less than our previously set baseline accuracy of *0.58195211786372*. Therefore, the Gaussian Naïve Bayes approach performed quite poorly and we opted to try other algorithms.

4.3 K-Nearest Neighbors

K-Nearest Neighbors (KNN) [Altman, 1992] is a supervised Machine Learning model used for both classification and regression. It is non-parametric, meaning that it does not make any assumptions about the distribution of the data. In KNN classification, a new example's label is assigned to the majority class of it's K-Nearest Neighbors. For example in figure 4.7, if K=2 (meaning that the number of neighbors is equal to 2), the new example represented by the green circle will get assigned to the red triangle class since its 2 nearest neighbors are red triangles.

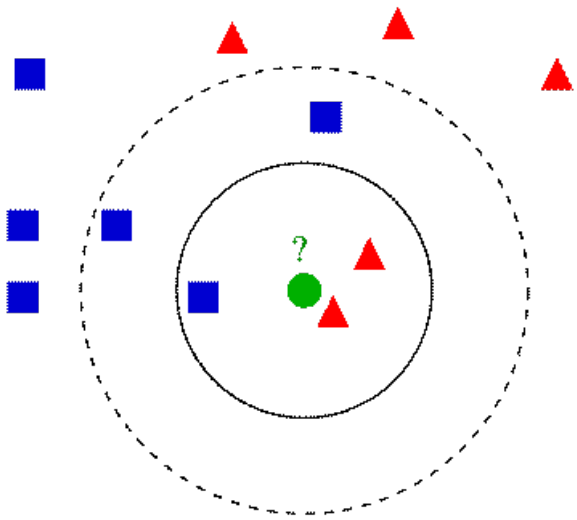


Figure 4.7: Visual Representation of the K-Nearest Neighbors Classifier [Wikipedia, 2007]

The pseudocode that governs KNN is shown below:

```
for all test examples x do
    for all training examples (x_i,y_i) do
        compute distance d(x,x_i)
    end for
    select the K-Nearest Neighbors of x
    return class of x as majority class among neighbors:
```

$$argmax_y \sum_{i=1}^{\kappa} \sigma(y, y_i)$$

```
end for
```

We performed the same pre-processing done for the previous algorithms, but we used Scikit Learn's KNeighborsClassifier. After that, we ran K-Fold Cross-Validation and Hyperparameter Tuning using Grid Search Cross-Validation.

4.3.1 K-Fold Cross Validation

Since the hyperparameters for the K-Nearest Neighbors algorithm are different than SVM's hyperparameters, we explained KNN's ones below:

- N_neighbors: which is the number of neighbors that a new example should take into consideration upon classification (previously denoted as K).

- Weights: which is the weight function used in the predictions. It determines if all examples are weighted equally or not.
- Metric: which is the type of distance to be measured between the new example and its corresponding neighbors.

The snippet of code below shows our choices for all of these parameters:

```
n_neighbors = [3,5,7]
weights = ['uniform','distance']
metric = ['euclidean','manhattan']
```

We performed K-Fold Cross-Validation by trying 12 different combinations of hyperparameters based on the values mentioned above. The code that explains this process is shown below:

```
try:
    from sklearn.model_selection import KFold, cross_val_score
    legacy = False
except ImportError:
    from sklearn.cross_validation import KFold, cross_val_score
    legacy = True

if legacy:
    kf = KFold(len(training_labels),n_folds=10, shuffle=True, random_state=42)
else:
    kf = KFold(n_splits=10, shuffle=True, random_state=42)

n_neighbors = [3,5,7]
weights = ['uniform','distance']
metric = ['euclidean','manhattan']
accuracy_scores = []
the_neighbors=[]
the_weights=[]
the_metrics=[]

from sklearn.neighbors import KNeighborsClassifier

for neighbor in n_neighbors:
   for weight in weights:
      for m in metric:
         knn = KNeighborsClassifier(n_neighbors=neighbor, weights=weight,
             metric=m)
         if legacy:
               scores = cross_val_score(knn, training_tweets_vectors,
                   training_labels_array, cv=kf, scoring='accuracy')
         else:
                 scores = cross_val_score(knn, training_tweets_vectors,
                     training_labels_array, cv=kf.split(training_tweets_vectors),
                     scoring='accuracy')
         accuracy_score = scores.mean()
         accuracy_scores.append(accuracy_score)
         the_neighbors.append(neighbor)
         the_weights.append(weight)
         the_metrics.append(m)
```

```
best_index = np.array(accuracy_scores).argmax()
best_neighbor = the_neighbors[best_index]
best_weight = the_weights[best_index]
best_metric = the_metrics[best_index]
print("The best combination of hyperparameters is:
    Neighbors=",str(best_neighbor),"\t","Weights=",str(best_weight),"\t",
"Metric=",str(best_metric))
```

After trying out all of the possible hyperparameter combinations, we got that the best one was:
N_ neighbors = 7, Weights = distance, Metric = euclidean
Therefore, we set our K-Nearest Neighbor model with these hyperparameters and then we ran it to predict on the test set as shown below:

```
knn = KNeighborsClassifier(n_neighbors=best_neighbor, weights=best_weight,
    metric=best_metric)
knn.fit(training_tweets_vectors, training_labels_array)

predicted_labels = knn.predict(testing_tweets_vectors)
```

Our classification results are shown in the following table 4.15:

Class	Precision	Recall	F1-score	Support
-1	0.64	0.76	0.69	381
0	0.42	0.41	0.41	210
1	0.52	0.19	0.28	115
Accuracy			0.57	706
Macro Average	0.53	0.45	0.46	706
Weighted Average	0.55	0.57	0.54	706
Model Accuracy	***0.5651558073654391***			

Table 4.15: Report of the Results of the K-Fold Cross-Validation Approach with the Best Hyperparameters

The model accuracy we got was *0.5651558073654391*, which is a bit less than the baseline accuracy of *0.58195211786372*. This means that the K-Nearest Neighbor algorithm performed badly like its previous Naïve Bayes counterpart and we opted to see if Hyperparameter Tuning using Grid Search Cross-Validation would improve the results.

4.3.2 Hyperparameter Tuning using Grid Search Cross-Validation

The parameters that we chose were the same ones we had previously mentioned and we used Scikit Learn's GridSearchCV estimator as show in the code below:

```
try:
    from sklearn.model_selection import GridSearchCV
except ImportError:
    from sklearn.grid_search import GridSearchCV

possible_parameters = {
    'n_neighbors': [3,5,7],
    'weights': ['uniform','distance'],
    'metric': ['euclidean','manhattan']
```

```
}

from sklearn.neighbors import KNeighborsClassifier

knn = GridSearchCV(KNeighborsClassifier(), possible_parameters, n_jobs=4, cv=3)
knn.fit(training_tweets_vectors, training_labels_array)
print("Best parameters: ",knn.best_params_)

predicted_labels = knn.predict(testing_tweets_vectors)
```

The best hyperparameters for this approach were:
N_ neighbors = 7, Weights = uniform, Metric = manhattan
The full results we got are shown in the following table 4.16

Class	**Precision**	**Recall**	**F1-score**	**Support**
-1	0.61	0.80	0.69	381
0	0.42	0.38	0.40	210
1	0.50	0.10	0.16	115
Accuracy			0.56	706
Macro Average	0.51	0.42	0.42	706
Weighted Average	0.54	0.56	0.52	706
Model Accuracy	***0.5580736543909348***			

Table 4.16: Report of the Results of the Grid Search Cross-Validation Approach

This approach scored an accuracy of *0.5580736543909348*, which was even less than the K-Fold Cross-Validation approach. Therefore, we believed that K-Nearest Neighbors algorithm performed poorly for this classification task and we resorted to keep trying other algorithms to see if they would outperform it.

4.4 Random Forests

Random Forests [Ho, 1995] is an ensemble learning approach [Opitz and Maclin, 1999] that consists of a large number of decision trees [Breiman, 2017], each of which classifies a new example independently. The final classification result will be the majority label of all of the decision trees as shown in figure 4.8 below:

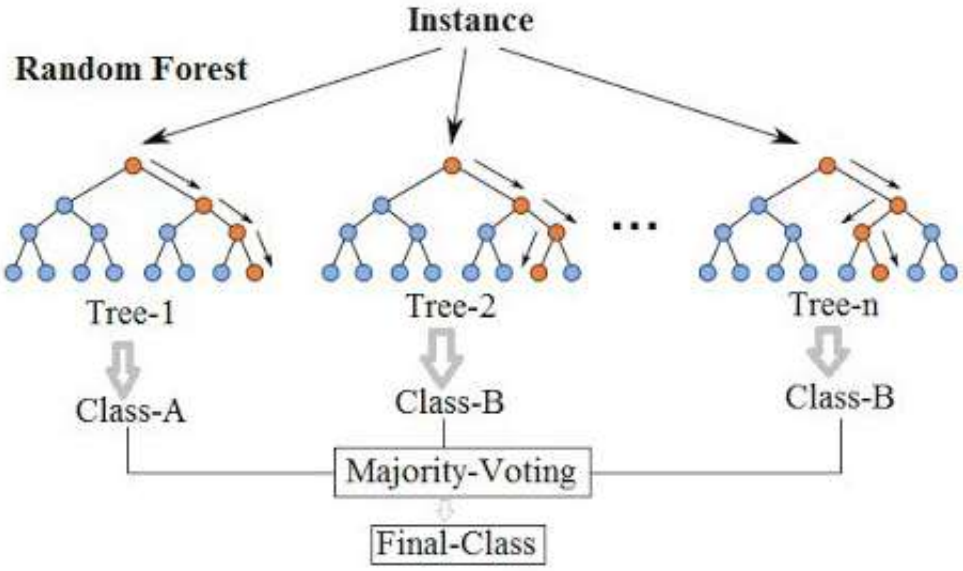


Figure 4.8: Visual Representation of the Random Forests Classifier [Medium, 2017]

For this approach, we performed the same pre-processing we had already done with the other algorithms and we used Scikit Learn's RandomForestClassifier. Since Random Forests have hyperparameters to tune, we performed model selection by using K-Fold Cross-Validation and Hyperparameter Tuning using Grid Search Cross-Validation.

4.4.1 K-Fold Cross-Validation

Random Forests have their own hyperparameters, thus we opted to explain them below:

- N_estimators: which is the number of trees in the forest.
- Max_depth: which is the maximum depth that the trees can reach.
- Min_samples_split: which is the minimum number of samples required to split an internal node.
- Min_samples_leaf: which is the minimum number of samples required to be at a leaf node.

The values we chose for the hyperparameters are shown in the snippet of code below:

```
n_estimators = [100, 300, 500, 800, 1200]
max_depth = [5, 8, 15, 25, 30]
min_samples_split = [2, 5, 10, 15, 100]
min_samples_leaf = [1, 2, 5, 10]
```

We performed K-Fold Cross-Validation with 500 different possible combinations of hyperparameters:

```
try:
    from sklearn.model_selection import KFold, cross_val_score
    legacy = False
except ImportError:
    from sklearn.cross_validation import KFold, cross_val_score
```

```
    legacy = True

if legacy:
    kf = KFold(len(training_labels),n_folds=10, shuffle=True, random_state=42)
else:
    kf = KFold(n_splits=10, shuffle=True, random_state=42)

n_estimators = [100, 300, 500, 800, 1200]
max_depth = [5, 8, 15, 25, 30]
min_samples_split = [2, 5, 10, 15, 100]
min_samples_leaf = [1, 2, 5, 10]
accuracy_scores = []
the_estimators=[]
the_depths=[]
the_min_splits=[]
the_min_leaves=[]

from sklearn.ensemble import RandomForestClassifier

for estimator in n_estimators:
   for depth in max_depth:
      for min_sample_split in min_samples_split:
         for min_sample_leaf in min_samples_leaf:
            rf = RandomForestClassifier(n_estimators=estimator, max_depth=depth,
                min_samples_split=min_sample_split,min_samples_leaf=min_sample_leaf)
            if legacy:
               scores = cross_val_score(rf, training_tweets_vectors,
                   training_labels_array, cv=kf, scoring='accuracy')
            else:
               scores = cross_val_score(rf, training_tweets_vectors,
                   training_labels_array, cv=kf.split(training_tweets_vectors),
                   scoring='accuracy')
            accuracy_score = scores.mean()
            accuracy_scores.append(accuracy_score)
            the_estimators.append(estimator)
            the_depths.append(depth)
            the_min_splits.append(min_sample_split)
            the_min_leaves.append(min_sample_leaf)

best_index = np.array(accuracy_scores).argmax()
best_estimator = the_estimators[best_index]
best_depth = the_depths[best_index]
best_split = the_min_splits[best_index]
best_leaf = the_min_leaves[best_index]
print("The best combination of hyperparameters is:
    Estimators=",str(best_estimator),"\t","Depth=",str(best_depth),"\t",
"min_samples_split=",str(best_split),"\t","min_samples_leaf=",str(best_leaf))
```

The best combination of hyperparameters turned out to be the following:
N_estimators = 500, Max_depth = 15, Min_samples_split = 10, Min_samples_leaf = 2
After we got the best combination, we set it on our Random Forests model and we ran it to predict on the test set:

```
rf = RandomForestClassifier(n_estimators=best_estimator, max_depth=best_depth,
    min_samples_split=best_split,min_samples_leaf=best_leaf)
rf.fit(training_tweets_vectors, training_labels_array)
```

```
predicted_labels = rf.predict(testing_tweets_vectors)
```

The results we got are shown in table 4.17:

Class	Precision	Recall	F1-score	Support
-1	0.57	0.97	0.72	381
0	0.57	0.14	0.23	210
1	1.00	0.03	0.05	115
Accuracy			0.57	706
Macro Average	0.71	0.38	0.33	706
Weighted Average	0.64	0.57	0.46	706
Model Accuracy	***0.5694050991501416***			

Table 4.17: Report of the Results of the K-Fold Cross-Validation Approach with the Best Hyperparameters

The Random Forests model achieved a near perfect recall of *0.97* for class *-1* with a precision of *0.57* which was above average. For class *0*, it achieved the same precision of class *-1* but with a much less recall of *0.14*. As far as class *1* is concerned, it achieved a perfect precision of *1.00* but a terrible recall of *0.03*. The overall accuracy that this model yielded was *0.5694050991501416*, which was still less than the baseline accuracy of *0.58195211786372*. This means that it performed poorly and we opted to see if Hyperparameter Tuning using Grid Search Cross-Validation would outperform it.

4.4.2 Hyperparameter Tuning using Grid Search Cross-Validation

For this part of our research, we used the same hyperparameters that we previously mentioned, in addition to Scikit Learn's GridSearchCV estimator:

```
try:
    from sklearn.model_selection import GridSearchCV
except ImportError:
    from sklearn.grid_search import GridSearchCV

possible_parameters = {
   'n_estimators': [100, 300, 500, 800, 1200],
   'max_depth': [5, 8, 15, 25, 30],
   'min_samples_split': [2, 5, 10, 15, 100],
   'min_samples_leaf': [1, 2, 5, 10]
}

from sklearn.ensemble import RandomForestClassifier

rf = GridSearchCV(RandomForestClassifier(), possible_parameters, n_jobs=4, cv=3)
rf.fit(training_tweets_vectors, training_labels_array)
print("Best parameters: ",rf.best_params_)

predicted_labels = rf.predict(testing_tweets_vectors)
```

The best combination of hyperparameters we got for this approach was:
N_estimators = 300, Max_depth = 30, Min_samples_split = 2, Min_samples_leaf = 1
Our results are detailed in table 4.18:

Class	Precision	Recall	F1-score	Support
-1	0.58	0.96	0.72	381
0	0.57	0.17	0.26	210
1	0.86	0.05	0.10	115
Accuracy			0.58	706
Macro Average	0.67	0.39	0.36	706
Weighted Average	0.62	0.58	0.48	706
Model Accuracy	***0.5779036827195467***			

Table 4.18: Report of the Results of the Grid Search Cross-Validation Approach

This approach scored an accuracy of *0.5779036827195467* with nearly identical results to the previous K-Fold Cross-Validation approach. Since it was still less than the majority baseline, we decided to keep trying other algorithms.

4.5 Logistic Regression

Logistic Regression [Kleinbaum et al., 2002] is a supervised Machine Learning algorithm used for classification. It uses a Sigmoid Activation Function [Han and Moraga, 1995] as its cost function instead of a linear function in order to map the predicted labels to their corresponding probabilities. The following figure 4.9 shows the Logistic Regression approach in action in a binary classification task:

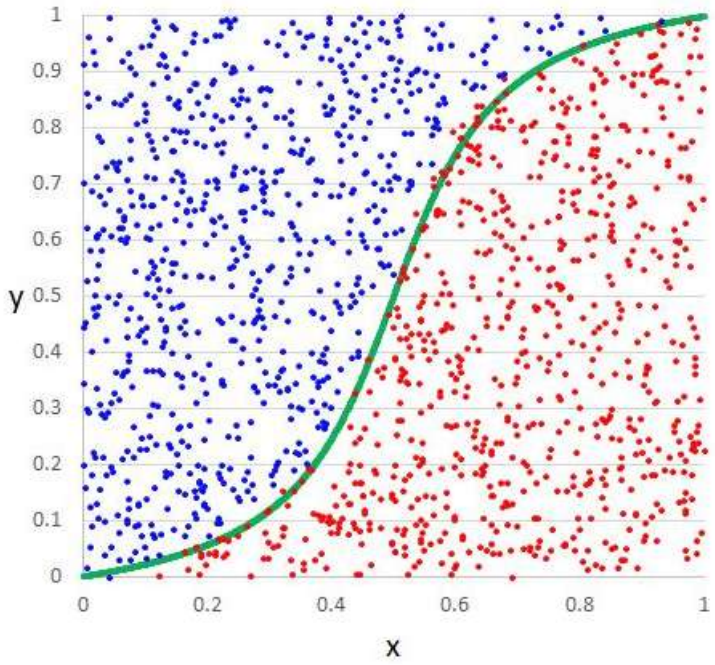


Figure 4.9: Visual Representation of the Logistic Regression Classifier in a Binary Classification Task [HelloAcm, 2016]

We used the same pre-processing technique of all the aforementioned algorithms, but this time we paired it up with Scikit Learn's LogisticRegression library and then we ran K-Fold Cross-Validation and Hyperparameter Tuning using Grid Search Cross-Validation.

4.5.1 K-Fold Cross-Validation

Since Logistic Regression has its own hyperparameters, we explained them below:

- C: which is the same regularization parameter previously explained in the Support-Vector Machines section.
- Penalty: which is the penalty added to the Sigmoid Activation Function in order to prevent overfitting.

The values we chose for both of these hyperparameters are denoted in the code below:

```
C = np.logspace(0, 4, 10) #the numbers on a log scale
penalty = ['l1', 'l2']
```

We performed K-Fold Cross-Validation on 20 different combinations of hyperparameters:

```
try:
    from sklearn.model_selection import KFold, cross_val_score
    legacy = False
except ImportError:
    from sklearn.cross_validation import KFold, cross_val_score
    legacy = True

if legacy:
    kf = KFold(len(training_labels),n_folds=10, shuffle=True, random_state=42)
else:
    kf = KFold(n_splits=10, shuffle=True, random_state=42)

C = np.logspace(0, 4, 10)
penalty = ['l1', 'l2']
accuracy_scores = []
the_Cs=[]
the_penalties=[]

from sklearn.linear_model import LogisticRegression

for v in C:
   for p in penalty:
      logreg = LogisticRegression(C=v,penalty=p)
      if legacy:
         scores = cross_val_score(logreg, training_tweets_vectors,
             training_labels_array, cv=kf, scoring='accuracy')
      else:
         scores = cross_val_score(logreg, training_tweets_vectors,
             training_labels_array, cv=kf.split(training_tweets_vectors),
             scoring='accuracy')
      accuracy_score = scores.mean()
      if np.isnan(accuracy_score):
         accuracy_score=0 #skipped nan
      accuracy_scores.append(accuracy_score)
      the_Cs.append(v)
      the_penalties.append(p)

best_index = np.array(accuracy_scores).argmax()
best_C = the_Cs[best_index]
best_penalty = the_penalties[best_index]
print("The best combination of hyperparameters is: C=",str(best_C),"\t","
    Penalty=",str(best_penalty))
```

The combination of hyperparameters that prevailed was the following:
C = 21.544346900318832, Penalty = l2
Once we got our best hyperparameters, we set them on the Logistic Regression model and we ran it to predict on the test set like the following code shows:

```
logreg = LogisticRegression(C=best_C, penalty=best_penalty)
logreg.fit(training_tweets_vectors, training_labels_array)

predicted_labels = logreg.predict(testing_tweets_vectors)
```

The results we got are shown in table 4.19:

Class	Precision	Recall	F1-score	Support
-1	0.65	0.84	0.73	381
0	0.57	0.42	0.48	210
1	0.57	0.28	0.37	115
Accuracy			0.62	706
Macro Average	0.60	0.51	0.53	706
Weighted Average	0.61	0.62	0.60	706
Model Accuracy	***0.623229461756374***			

Table 4.19: Report of the Results of the K-Fold Cross-Validation Approach with the Best Hyperparameters

This accuracy of *0.623229461756374* exceeded the baseline by around 4%, which was quite a good improvement for the multiclass classification task on our final dataset of *3527 tweets*. Precision for all 3 classes hovered around *0.6*, with recall for the *-1* class achieving an impressive score of *0.84*. Recall for the *0* class achieved a near average score of *0.42*, while for class *1* it scored poorly with *0.28*. Overall, we were satisfied with the results and we opted to try Hyperparameter Tuning using Grid Search Cross-Validation like we did with the other algorithms to see if this accuracy might increase.

4.5.2 Hyperparameter Tuning using Grid Search Cross-Validation

We used the GridSearchCV estimator as shown below:

```
try:
    from sklearn.model_selection import GridSearchCV
except ImportError:
    from sklearn.grid_search import GridSearchCV

penalty = ['l1', 'l2']
C = np.logspace(0, 4, 10)
possible_parameters = dict(C=C, penalty=penalty)

from sklearn.linear_model import LogisticRegression

logreg = GridSearchCV(LogisticRegression(), possible_parameters, n_jobs=4, cv=3)
logreg.fit(training_tweets_vectors, training_labels_array)
print("Best parameters: ",logreg.best_params_)
```

The best hyperparameters and results we got for this approach were the same ones we got in the previous K-Fold Cross-Validation approach:

C = 21.544346900318832, Penalty = l2
The results are detailed in table 4.20:

Class	Precision	Recall	F1-score	Support
-1	0.65	0.84	0.73	381
0	0.57	0.42	0.48	210
1	0.57	0.28	0.37	115
Accuracy			0.62	706
Macro Average	0.60	0.51	0.53	706
Weighted Average	0.61	0.62	0.60	706
Model Accuracy	***0.623229461756374***			

Table 4.20: Report of the Results of the Grid Search Cross-Validation Approach

Since the Grid Search Cross-Validation approach provided us with the same results of the K-Fold Cross-Validation approach, we decided to try to see if the final Voting Classifier explained in the next section would outperform all of the previous algorithms.

4.6 Voting Classifier

The Voting Classifier is an ensemble learning approach that gets fed with other Machine Learning classifiers, learns from their inaccuracies and votes for the class that they predicted in two ways:

- Hard Voting: which is voting based on the majority class that the models predicted.
- Soft Voting: which is voting based on the maximum probability for each predicted class.

Figure 4.10 demonstrates a visual representation of the Voting Classifier:

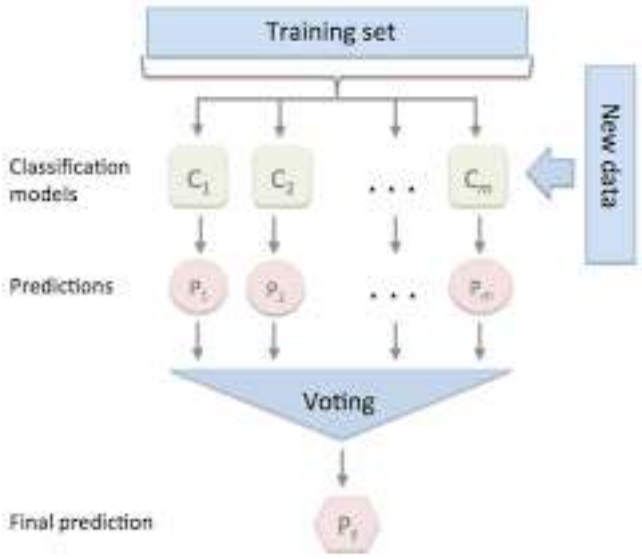


Figure 4.10: Visual Representation of the Voting Classifier [Medium, 2019]

In this approach, we used Scikit Learn's VotingClassifier and we fed it with all of the previously explained classifiers with their best hyperparameters. Then we ran both the Hard Voting and Soft Voting classifiers in order to see which one would score better.

4.6.1 Hard Voting

The code that shows how we fed the Machine Learning classifiers with their best hyperparameters to the Hard Voting Classifier is shown below:

```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier, VotingClassifier

clf1 = LogisticRegression(C= 21.544346900318832, penalty='l2')
clf2 = RandomForestClassifier(max_depth = 30, min_samples_leaf= 1,
    min_samples_split=2, n_estimators= 300)
clf3 = SVC(C=100, gamma=0.1, kernel='rbf', probability=True)
clf4 = GaussianNB()
clf5 = KNeighborsClassifier(metric = 'manhattan', n_neighbors = 7, weights =
    'uniform')

eclf1 = VotingClassifier(estimators=[('lr', clf1), ('rf', clf2), ('svm', clf3),
    ('gnb', clf4),('knn', clf5)], voting='hard')
eclf1.fit(training_tweets_vectors, training_labels_array)

predicted_labels = eclf1.predict(testing_tweets_vectors)
```

The results we got for this approach are shown in table 4.21:

Class	Precision	Recall	F1-score	Support
-1	0.63	0.90	0.74	381
0	0.61	0.37	0.46	210
1	0.66	0.18	0.29	115
Accuracy			0.63	706
Macro Average	0.63	0.48	0.50	706
Weighted Average	0.63	0.63	0.58	706
Model Accuracy	***0.6260623229461756***			

Table 4.21: Report of the Results of the Hard Voting Classifier Approach

This approach was the highest scoring one in our entire research with an overall classification accuracy of *0.6260623229461756*. Precision was above average for classes *-1, 0 and 1* with their values being *0.63, 0.61 and 0.66* respectively. As far as recall is concerned, its values were *0.90* for class *-1*, *0.37* for class *0* and *0.18* for class *1*. Despite the fact that recall for the last two classes was below average, the results were quite acceptable since the overall classification accuracy exceeded the baseline by around 4%. Nevertheless, we opted to see how well would the Soft Voting approach score in comparison to the Hard Voting one.

4.6.2 Soft Voting

The following code shows how we fed the Machine Learning classifiers with their best hyperparameters to the Soft Voting classifier:

```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
```

```
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier, VotingClassifier

clf1 = LogisticRegression(C= 21.544346900318832, penalty='l2')
clf2 = RandomForestClassifier(max_depth = 30, min_samples_leaf= 1,
    min_samples_split=2, n_estimators= 300)
clf3 = SVC(C=100, gamma=0.1, kernel='rbf', probability=True)
clf4 = GaussianNB()
clf5 = KNeighborsClassifier(metric = 'manhattan', n_neighbors = 7, weights =
    'uniform')

eclf2 = VotingClassifier(estimators=[('lr', clf1), ('rf', clf2), ('svm', clf3),
    ('gnb', clf4),('knn', clf5)], voting='soft')
eclf2.fit(training_tweets_vectors, training_labels_array)

predicted_labels = eclf2.predict(testing_tweets_vectors)
```

The results we got are shown in table 4.22:

Class	Precision	Recall	F1-score	Support
-1	0.69	0.78	0.73	381
0	0.52	0.50	0.51	210
1	0.50	0.33	0.40	115
Accuracy			0.62	706
Macro Average	0.57	0.54	0.55	706
Weighted Average	0.61	0.62	0.51	706
Model Accuracy	***0.623229461756374***			

Table 4.22: Report of the Results of the Soft Voting Classifier Approach

The Soft Voting approach garnered an overall accuracy of *0.623229461756374* which was nearly identical to the accuracy of its Hard Voting counterpart. Despite the fact that their accuracies were very similar, this approach performed better on recall with more balanced values. Two thirds of the recall values were greater than or equal to *0.50*: being *0.78* for class *-1* and *0.50* for class *0*. Class *1* had a recall of *0.33* which was still greater than its recall value for the Hard Voting approach that amounted to *0.18*. As far as precision is concerned, the values for all classes were greater than or equal to *0.50* as classes *-1, 0, 1* achieved precision values of *0.69, 0.52 and 0.50* respectively.

At the end of our research, we had tried 17 different unique approaches which were the following:

1. SVM without K-Fold Cross-Validation nor Hyperparameter Tuning using Grid Search Cross-Validation
2. SVM with K-Fold Cross-Validation
3. SVM with Hyperparameter Tuning using Grid Search Cross-Validation
4. SVM with K-Fold Cross-Validation and Further Tweet Pre-processing
5. SVM with Hyperparameter Tuning using Grid Search Cross-Validation and Further Tweet Pre-processing
6. SVM with K-Fold Cross-Validation and Splitting Hashtags

7. SVM with Hyperparameter Tuning using Grid Search Cross-Validation and Splitting Hashtags
8. SVM with the 2 Classes Approach
9. Naïve Bayes
10. K-Nearest Neighbors with K-Fold Cross-Validation
11. K-Nearest Neighbors with Hyperparameter Tuning using Grid Search Cross-Validation
12. Random Forests with K-Fold Cross-Validation
13. Random Forests with Hyperparameter Tuning using Grid Search Cross-Validation
14. Logistic Regression with K-Fold Cross-Validation
15. Logistic Regression with Hyperparameter Tuning using Grid Search Cross-Validation
16. Hard Voting Classifier
17. Soft Voting Classifier

All of our final results along with their corresponding error analysis were discussed in the following section.

Chapter 5

Results and Error Analysis

5.0.1 Results

The results for all the approaches we conducted, including their overall accuracies are detailed in table 5.1 in chronological order. It is important to mention that everything colored in grey means that this is the majority baseline. Everything in white signifies that these were the intermediate steps that led to the valid approaches that are in green. By valid approaches we mean that they scored higher than the majority baseline in the multiclass classification task of the *3527 tweets*. Everything colored in red means that they were either for the binary classification task that we discarded, or algorithms that were for the multiclass classification task but that scored less than the majority baseline and were consequently discarded.

Approach	**Number of Tweets**	**Macro Avg. F1-Score**	**Weighted Avg. F1-Score**	**Overall Accuracy**
SVM without CV nor Grid Search CV	2715	0.25	0.43	***0.5819 (majority baseline)***
SVM with CV	2715	0.25	0.44	***0.5837 (majority baseline)***
SVM with Grid Search CV	2715	0.54	0.62	***0.6372***
SVM with CV and Further Tweet Pre-processing	2715	0.52	0.61	***0.6298***
SVM with Grid Search CV and Further Tweet Pre-processing	2715	0.55	0.62	***0.6372***
SVM with CV and Splitting Hashtags	2975	0.52	0.60	***0.6302***
SVM with Grid Search CV and Splitting Hashtags	2975	0.53	0.61	***0.6336***
SVM with CV	3527	0.51	0.58	***0.6147***
SVM with Grid Search CV	3527	0.51	0.58	***0.6147***

SVM with CV on 2 Classes: 0 and 1	1240	0.70	0.75	***0.7620***
SVM with Grid Search CV on 2 Classes: 0 and 1	1240	0.70	0.75	***0.7661***
SVM with CV on 2 Classes: -1 and 2	2975	0.70	0.70	***0.7025***
SVM with Grid Search CV on 2 Classes: -1 and 2	2975	0.69	0.70	***0.6974***
Naïve Bayes	3527	0.47	0.51	***0.4957 (less than the majority baseline)***
K-Nearest Neighbors with CV	3527	0.46	0.54	***0.5651 (less than the majority baseline)***
K-Nearest Neighbors with Grid Search CV	3527	0.42	0.52	***0.5580 (less than the majority baseline)***
Random Forests with CV	3527	0.33	0.46	***0.5694 (less than the majority baseline)***
Random Forests with Grid Search CV	3527	0.36	0.48	***0.5779 (less than the majority baseline)***
Logistic Regression with CV	3527	0.53	0.60	***0.6232***
Logistic Regression with Grid Search CV	3527	0.53	0.60	***0.6232***
Hard Voting Classifier	3527	0.50	0.58	***0.6260***
Soft Voting Classifier	3527	0.55	0.51	***0.6232***

Table 5.1: Report of the Results of all of the Machine Learning Approaches

The results clearly showed that the best performing algorithms sorted by ascending order (overall accuracy) for the multiclass classification scenario of the *3527 tweets* were:

- **Support-Vector Machines:** with an overall accuracy of ***0.6147***, a macro average F1-score of *0.51* and a weighted average F1-score of *0.58*.
- **Soft Voting Classifier:** with an overall accuracy of ***0.6232***, a macro average F1-score of *0.55* and a weighted average F1-score of *0.51*.
- **Logistic Regression:** with an overall accuracy of ***0.6232***, a macro average F1-score of *0.53* and a weighted average F1-score of *0.60*.
- **Hard Voting Classifier:** with an overall accuracy of ***0.6260***, a macro average F1-score of *0.50* and a weighted average F1-score of *0.58*.

The approach that we preferred over all others based on our results was the Soft Voting Classifier even if it was not the highest scoring one. This is due to the fact that its values for both precision and recall 4.22 were the most balanced between all classes in comparison to all of the other approaches. In addition to that, its overall classification accuracy of *0.6232* was quite acceptable, exceeding the majority baseline by around 4%, which meant that this model actually learned from our data.

That was a suitable time to see what perception of Arabs on migration to Europe our Soft Voting Classifier identified. To properly explain this perception, we first opted to mention how our dataset was composed at the end. It consisted of a total of *3527 tweets*:

- *2049 tweets* were annotated as *-1* and consequently did not count in our perception identification.
- *970 tweets* were annotated as *0*, thus they exhibited a negative perception on migration to Europe.
- *508 tweets* were annotated as *1*, thus they exhibited a positive perception on migration to Europe.

Therefore, there was a total of *1478 tweets* in our entire dataset that contributed into perception identification with ***65.6%*** of them having a negative perception on migration to Europe and the rest ***34.4%*** having a positive perception on migration to Europe. Thus in "real life", we concluded that the perception of Arabs on migration to Europe is **negative**. We wanted to see if our classifier's results would align with this "real life" perception after predicting on the testing set.

706 random tweets of the aforementioned dataset were used for testing:

- *381 tweets* were annotated as *-1* and consequently did not count in our perception identification.
- *210 tweets* were annotated as *0*, thus they exhibited a negative perception on migration to Europe.
- *115 tweets* were annotated as *1*, thus they exhibited a positive perception on migration to Europe.

Therefore, there was a total of *325 tweets* in the testing set that contributed into perception identification. Our results showed that the Soft Voting Classifier correctly classified *105 tweets* out of the *210 tweets* as having a negative perception on migration to Europe. In addition to that, it classified *38 tweets* out of the *115 tweets* as having a positive perception on migration to Europe. Therefore, it detected that ***80.4%*** of tweets carried a **negative** perception on migration to Europe with an overall classification accuracy of ***62.32%***. Based on our findings, we can confidently say that our model did an above average job in identifying if Arabs have a positive or negative perception on migration to Europe. This perception turned out to be mostly **negative**, which aligns perfectly with the "real life" perception we had previously inferred. We are aware that such a small sample of tweets might not reflect the perception of all Arabs on migration to Europe, therefore we stress on the fact that further data is needed to identify a more accurate perception.

5.0.2 Error Analysis

Our Soft Voting Classifier misclassified a total of *267 tweets* out of *706 tweets*. Consequently, we decided to try to analyze some tweets that were misclassified in order to determine the reasons behind these errors. Some cleaned tweets in addition to our proposed misclassification reasons are mentioned below:

- **Bias because of one word:**
  First Tweet Example:

  "بالنسبة للناشطين المدنيين لاتخافون الخطف لأن كلهن عشرة أيام ويهدوك هاي العشرة أيام راح تفيدك تطلب لجوء لدول أوروبا امريكا"

  => "For what concerns civil rights activists, do not be scared of kidnapping because it'll just be ten days and you will be released. These ten days will benefit you to ask for asylum in European countries or America".

  Correct Label: 1
  Predicted Label: 0

  In this tweet, we analyzed that the algorithm might have been biased towards the word "الخطف" => "Kidnapping", and thus predicted the entire tweet as negative while actually it was positive since it revealed the individual's implicit desire to migrate to Europe.

  Second Tweet Example:

  "ذكاء الزعيم اردوغان انه بعيد ساعات بدئ عملية نبع السلام وبعد هجوم قواد الغرب هددهم بأنه ربما يسمح ل مليون لاجئ ان يذهبوا الى جنة اوروبا معظم قادة الغرب الرسميين صاروا يتفهمون دوافع تركيا وان الحق الدفاع حدودها هههه عالم متناقض"

  => "The intelligence of the leader 'Erdogan': he is hours away from starting Operation Peace Spring, and after the attack of the pimps of the West, he threatened them that he might allow 1 million refugees to go to the paradise of Europe. Most official leaders of the West are now aware of Turkey's intentions and that it has the right to defend its borders, hahaha such a contradictory world".

  Correct Label: 1
  Predicted Label: -1

  Like the previous tweet, we reasoned that this one might be biased by a word which we believed was "اردوغان" => "Erdogan". This word was present in around *500 tweets* in our dataset and its related tweets were mostly classified as unrelated due to news about Erdogan's threats to the European Union regarding migration. Therefore, the algorithm probably saw the word "Erdogan" and classified the tweet as unrelated. But it failed to value the most important sentence which was "جنة اوروبا" => "The paradise of Europe" which clearly showed that Europe was portrayed positively as a paradise. Thus this tweet should have been classified as carrying a positive perception about Europe.

- **Use of sophisticated Arabic:**

  "العوام يتمنون الهجرة أوروبا و المستقيمون يتمنون الهجرة بلد التوحيد فشتان الهجرتين"

=> "Common people wish to migrate to Europe and rational people wish to migrate to the country of monotheism, and what a difference between both migrations".

Correct Label: 0
Predicted Label: 1

This eloquent use of Arabic mislead the classifier, as it fell into the trap of classifying the tweet as positive just because of the sentence "Common people wish to migrate to Europe", while it didn't give the sentence "Rational people wish to migrate to the country of monotheism" a bigger weight. If it had given it a bigger weight, it would have classified the tweet correctly as carrying a negative perception on migration to Europe.

- **Use of dialectal Arabic:**

"خي مسيحي اوروبا دغري بتعطيهن فيز هجرة وكمان بتبعتلن طيارات وبواخر ومع السلامة"

=> "Brother, Europe gives Christians immigration visas immediately and it also sends them planes and ships. And goodbye".

Correct Label: 1
Predicted Label: -1

In this tweet, dialectal Arabic was used which is quite different than the majority of the tweets that were in traditional Arabic and that the algorithm was trained on. Therefore, the algorithm classified this tweet as unrelated while it should have been classified as positive since it claimed that Europe pampers Christian immigrants.

- **Connecting unconnectable words:**
For explanatory purposes, we opted to intentionally connect the words in the English translation.

"عائشة الجراد الهجرة ألف لاجيء سوري يدخلون أوربا رغماً أنف أوروبا سمحت أوروبا بزواج المثليّين وتلك حكمة الله وعدله لينقطع نسل الفجّارإلى الأبد ويأبى الله يتمّ نوره فيرسل سبحانه المهاجرين لتلك البلادفيزداد نسل المسلمين أوروبا"

=> "Aisha Al-Jarad's immigration: a thousand Syrian refugees enter Europe whetherEurope likes it or not. Europe allowed gay marriage and that's because of God's wisdom and his justice: may the offspring of immoralsBeCutOff forever. And God refuses that His light fades, so He sends immigrants to those countries-SoThatThereWillBeAnIncrease in the number of Muslim offspring in Europe".

Correct Label: 0
Predicted Label: -1

Due to the fact that many words were wrongly connected, they received one vector all together which created noise in the data and eventually yielded more errors. Therefore, this tweet was predicted as unrelated rather than negative.

- **Writing English words with Arabic letters:**

"اول واحد بيضيع انت بتروح لاجي هوملس اوروبا"

=> "Due to the first one getting lost, you will go to Europe as a homeless refugee".

Correct Label: 0
Predicted Label: 1

In this tweet, the word "Homeless" was written in Arabic letters => "هوملس", which is not an Arabic word. Therefore, the algorithm didn't pick it up as a negative word towards immigrants in Europe and it predicted the tweet as being positive rather than negative.

- **Use of metaphors:**

"وانا بالنسبة إلِي بتسوى قشرة بصلة وفرنك مقدوح تعتل اذا بتضل هيك النازحين رح يرجعوا ينزحوا ع اوروبا بركي بتروح بتتضامن معهم هونيك"

=> "To me, you are worth an onion's crust and your oven is lit. If you remain like this, migrants will re-immigrate to Europe and then maybe you will go and show solidarity with them there".

Correct Label: -1
Predicted Label: 0

In this tweet, derogatory comments were written to another individual using a metaphor. Due to this eloquent wording, the algorithm misclassified this tweet and labeled it as negative while it should have been classified as unrelated.

Chapter 6

Conclusion and Further Research

In our research, we crawled *3527* tweets in Arabic related to migration to Europe on a span of 3 months. Using these tweets, we created the first Arabic dataset on Arabs' perceptions on migration to Europe that we named "APME". We pre-processed the tweets and we ran 17 unique machine learning approaches with 6 different machine learning algorithms to try to identify if the aforementioned perception is positive or negative. We succeeded in doing so, as our most optimal algorithm was the Soft Voting Classifier which had an above average classification accuracy of ***62.32%***. It successfully identified this perception as **negative**, which is exactly the same as the one we inferred from our dataset. Thus it is safe to say that it is possible for a computer program to identify the perception of Arabs on migration to Europe. Despite the fact that we achieved a fair classification accuracy, it could have been quite higher had we crawled more tweets. In addition to that, the use of Deep Learning algorithms merged with advanced NLP techniques might have rendered higher quality results. Another approach that could have improved our results was to try to disconnect unconnectable words, but we refrained from doing so as it is quite a risky procedure. In a nutshell, we would say that for further research, having around 15000 annotated tweets that will be used with an algorithm such as Convolutional Neural Networks [Goodfellow et al., 2016] would probably improve precision, recall and consequently improve the overall classification accuracy. This way, the sample of people being studied would be bigger and this would eventually yield a more global and accurate perception.

Bibliography

[Afsaruddin, 2019] Afsaruddin, A. (2019). Umayyad dynasty. `https://www.britannica.com/topic/Umayyad-dynasty-Islamic-history`.

[Al-Smadi et al., 2018] Al-Smadi, M., Qawasmeh, O., Al-Ayyoub, M., Jararweh, Y., and Gupta, B. (2018). Deep recurrent neural network vs. support vector machine for aspect-based sentiment analysis of arabic hotels' reviews. *Journal of computational science*, 27:386–393.

[Alayba et al., 2017] Alayba, A. M., Palade, V., England, M., and Iqbal, R. (2017). Arabic language sentiment analysis on health services. In *2017 1st International Workshop on Arabic Script Analysis and Recognition (ASAR)*, pages 114–118. IEEE.

[Altman, 1992] Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185.

[Augenstein et al., 2016] Augenstein, I., Rocktäschel, T., Vlachos, A., and Bontcheva, K. (2016). Stance detection with bidirectional conditional encoding. *arXiv preprint arXiv:1606.05464*.

[Baly et al., 2018] Baly, R., Mohtarami, M., Glass, J., Màrquez, L., Moschitti, A., and Nakov, P. (2018). Integrating stance detection and fact checking in a unified corpus. *arXiv preprint arXiv:1804.08012*.

[Bamman, 2017] Bamman, D. (2017). Lecture 5: Truth and ethics. `http://people.ischool.berkeley.edu/~dbamman/nlpF17/slides/5_truth_ethics.pdf`.

[Bojanowski et al., 2017] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

[Breiman, 2017] Breiman, L. (2017). *Classification and regression trees*. Routledge.

[Bruns and Burgess, 2011] Bruns, A. and Burgess, J. E. (2011). The use of twitter hashtags in the formation of ad hoc publics. In *Proceedings of the 6th European Consortium for Political Research (ECPR) General Conference 2011*.

[Cheong et al., 2011] Cheong, H., Chiu, I., Shu, L., Stone, R. B., and McAdams, D. A. (2011). Biologically meaningful keywords for functional terms of the functional basis. *Journal of Mechanical Design*, 133(2):021007.

[Clifton et al., 2004] Clifton, C., Cooley, R., and Rennie, J. (2004). Topcat: Data mining for topic identification in a text corpus. *IEEE transactions on knowledge and data engineering*, 16(8):949–964.

[Cohen, 1960] Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

[Darwish et al., 2017] Darwish, K., Magdy, W., and Zanouda, T. (2017). Improved stance prediction in a user similarity feature space. In *Proceedings of the 2017 IEEE/ACM international conference on advances in social networks analysis and mining 2017*, pages 145–148. ACM.

[Facebook, 2020] Facebook (2020). Rate limits - graph api - documentation - facebook for developers. `https://developers.facebook.com/docs/graph-api/overview/rate-limiting/`.

[Fargues and Fandrich, 2012] Fargues, P. and Fandrich, C. (2012). Migration after the arab spring. `https://cadmus.eui.eu/handle/1814/23504`.

[for Migration, 2010] for Migration, I. O. (2010). *Intra-regional Labour Mobility in the Arab World*. IOM International Organization for Migration.

[Forin and Healy, 2018] Forin, R. and Healy, C. (2018). Trafficking along migration routes to europe: Bridging the gap between migration, asylum and anti-trafficking. `https://childhub.org/sites/default/files/webinars/bridging_the_gap_between_migration_asylum_and_anti-trafficking.pdf`.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[Han and Moraga, 1995] Han, J. and Moraga, C. (1995). The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International Workshop on Artificial Neural Networks*, pages 195–201. Springer.

[Hawkins, 2004] Hawkins, D. M. (2004). The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12.

[HelloAcm, 2016] HelloAcm (2016). Logistic regression. `https://helloacm.com/wp-content/uploads/2016/03/logistic-regression-example.jpg`.

[Ho, 1995] Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE.

[Huang et al., 2015] Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

[Java et al., 2007] Java, A., Song, X., Finin, T., and Tseng, B. (2007). Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 56–65. ACM.

[Joulin et al., 2016a] Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., and Mikolov, T. (2016a). Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

[Joulin et al., 2016b] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016b). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

[Joyce, 2003] Joyce, J. (2003). Bayes' theorem. `https://plato.stanford.edu/entries/bayes-theorem/`.

[Kleinbaum et al., 2002] Kleinbaum, D. G., Dietz, K., Gail, M., Klein, M., and Klein, M. (2002). *Logistic regression*. Springer.

[Krejzl et al., 2017] Krejzl, P., Hourová, B., and Steinberger, J. (2017). Stance detection in online discussions. *arXiv preprint arXiv:1701.00504*.

[Lafferty et al., 2001] Lafferty, J., McCallum, A., and Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. `https://repository.upenn.edu/cgi/viewcontent.cgi?article=1162&context=cis_papers`.

[Landis and Koch, 1977] Landis, J. R. and Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, 33 1:159–74.

[Levenshtein, 1966] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

[Li et al., 2018] Li, P., Lu, H., Kanhabua, N., Zhao, S., and Pan, G. (2018). Location inference for non-geotagged tweets in user timelines. *IEEE Transactions on Knowledge and Data Engineering*, 31(6):1150–1165.

[Li et al., 2009] Li, Y.-R., Wang, L.-H., and Hong, C.-F. (2009). Extracting the significant-rare keywords for patent analysis. *Expert Systems with Applications*, 36(3):5200–5204.

[Magdy et al., 2016] Magdy, W., Darwish, K., Abokhodair, N., Rahimi, A., and Baldwin, T. (2016). # isisisnotislam or# deportallmuslims?: Predicting unspoken views. In *Proceedings of the 8th ACM Conference on Web Science*, pages 95–106. ACM.

[Masood and Aker, 2018] Masood, R. and Aker, A. (2018). The fake news challenge: Stance detection using traditional machine learning approaches. In *KMIS*.

[Medium, 2017] Medium (2017). Random forests. `https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d`.

[Medium, 2019] Medium (2019). The voting classifier. `https://miro.medium.com/max/241/1*9JlC4KdtYg4_pJ9TI1QKBw.png`.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Mohammad et al., 2016] Mohammad, S., Kiritchenko, S., Sobhani, P., Zhu, X., and Cherry, C. (2016). Semeval-2016 task 6: Detecting stance in tweets. *Proceedings of International Workshop on Semantic Evaluation, Semeval-2016*, pages 31–41.

[Mohammad et al., 2017] Mohammad, S. M., Sobhani, P., and Kiritchenko, S. (2017). Stance and sentiment in tweets. *ACM Transactions on Internet Technology (TOIT)*, 17(3):26.

[Mohtarami et al., 2018] Mohtarami, M., Baly, R., Glass, J., Nakov, P., Màrquez, L., and Moschitti, A. (2018). Automatic stance detection using end-to-end memory networks. *arXiv preprint arXiv:1804.07581*.

[Mosteller and Tukey, 1968] Mosteller, F. and Tukey, J. W. (1968). Data analysis, including statistics. *Handbook of social psychology*, 2:80–203.

[Murtagh, 1991] Murtagh, F. (1991). Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(5-6):183–197.

[Murthy, 2018] Murthy, D. (2018). *Twitter*. Polity Press Cambridge, UK.

[Nabil et al., 2015] Nabil, M., Aly, M., and Atiya, A. (2015). Astd: Arabic sentiment tweets dataset. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2515–2519.

[Ng and Jordan, 2002] Ng, A. Y. and Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848.

[Opitz and Maclin, 1999] Opitz, D. and Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198.

[Pak and Paroubek, 2010] Pak, A. and Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining. In *Language Resources and Evaluation Conference (LREC)*, volume 10, pages 1320–1326.

[Parikh et al., 2016] Parikh, A. P., Täckström, O., Das, D., and Uszkoreit, J. (2016). A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

[Passerini, 2019] Passerini, A. (2019). Support-vector machines. `http://disi.unitn.it/~passerini/teaching/2017-2018/MachineLearning/slides/15_svm/talk.pdf`.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

[Pennacchiotti and Popescu, 2011] Pennacchiotti, M. and Popescu, A.-M. (2011). Democrats, republicans and starbucks afficionados: user classification in twitter. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 430–438. ACM.

[Pontiki et al., 2014] Pontiki, M., Galanis, D., Pavlopoulos, J., Papageorgiou, H., Androutsopoulos, I., and Manandhar, S. (2014). SemEval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, Dublin, Ireland. Association for Computational Linguistics.

[Refaee and Rieser, 2014] Refaee, E. and Rieser, V. (2014). An arabic twitter corpus for subjectivity and sentiment analysis. In *Language Resources and Evaluation Conference (LREC)*, pages 2268–2273.

[Riedel et al., 2017] Riedel, B., Augenstein, I., Spithourakis, G. P., and Riedel, S. (2017). A simple but tough-to-beat baseline for the fake news challenge stance detection task. *arXiv preprint arXiv:1707.03264*.

[Russell and Norvig, 2009] Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition.

[Salameh, 2019] Salameh, M. T. B. (2019). Migration from the arab spring countries to europe: Causes and consequences. In *Smart Technologies and Innovation for a Sustainable Future*, pages 243–254. Springer.

[Sidorov et al., 2012] Sidorov, G., Miranda-Jiménez, S., Viveros-Jiménez, F., Gelbukh, A., Castro-Sánchez, N., Velásquez, F., Díaz-Rangel, I., Suárez-Guerra, S., Trevino, A., and Gordon, J. (2012). Empirical study of machine learning based approach for opinion

mining in tweets. In *Mexican international conference on Artificial intelligence*, pages 1–14. Springer.

[Soliman et al., 2014] Soliman, T. H., Elmasry, M., Hedar, A., and Doss, M. (2014). Sentiment analysis of arabic slang comments on facebook. *International Journal of Computers & Technology*, 12(5):3470–3478.

[Stieglitz and Dang-Xuan, 2013] Stieglitz, S. and Dang-Xuan, L. (2013). Social media and political communication: a social media analytics framework. *Social network analysis and mining*, 3(4):1277–1291.

[Sukhbaatar et al., 2015] Sukhbaatar, S., Szlam, A., Weston, J., and Fergus, R. (2015). End-to-end memory networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 2440–2448, Cambridge, MA, USA. MIT Press.

[Touati et al., 2017] Touati, I., Graja, M., Ellouze, M., and Belguith, L. H. (2017). Crf-based arabic opinion summarization system. In *LPKM*.

[Twitter, 2020] Twitter (2020). Rate limiting - twitter. `https://developer.twitter.com/en/docs/basics/rate-limiting`.

[Webb, 2010] Webb, G. I. (2010). *Naïve Bayes*, pages 713–714. Springer US, Boston, MA.

[Wikipedia, 2007] Wikipedia, t. f. e. (2007). K-nearest neighbors. `https://it.wikipedia.org/wiki/K-nearest_neighbors#/media/File:KnnClassification.svg`.

[Wikipedia, 2019] Wikipedia, t. f. e. (2019). K-fold cross-validation. `https://en.wikipedia.org/wiki/Cross-validation_(statistics)#/media/File:K-fold_cross_validation_EN.svg`.

[Wong et al., 2013] Wong, F. M. F., Tan, C.-W., Sen, S., and Chiang, M. (2013). Quantifying political leaning from tweets and retweets in proceedings of the 7th international conference on weblogs and social media.