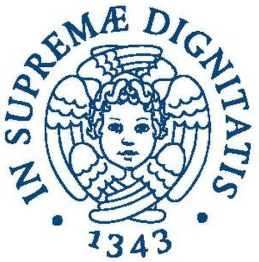


UNIVERSITÀ DI PISA

INFORMATICS DEPARTMENT

DISTRIBUTED DATA ANALYSIS AND MINING PROJECT

FAKE NEWS RECOGNITION

Authors:

Bruno Limón, Charlotte Brimont, Giovanni Battista D'Orsi, Rebecca Saitta

Academic year 2022/2023

Fake News Recognition

16 December 2022

Contents

List of Figures

List of Tables

1 Introduction

Misinformation, propaganda and fake news; these deceitful ways in which some seek to manipulate and twist information for their own benefit have always been a problem in society, recently however, with the advent of technology and a interconnected world, its spread has been exponential and out of control. Fake news coming from the web and social media are dangerous and an enormous risk that can create a distorted vision of reality that could alter and worsen the beliefs and actions of individuals.
This project aims to analyze and recognize the underlying characteristics of such dishonest information in the form of news articles, using Natural Language Processing techniques together with machine learning and the computing power of the spark platform through the use of *PySpark*. This with the intent to build a model that is able to discern between fake and real news.

2 Data Understanding & Preparation

The dataset to work with, *Fake and real news*, consists of 2 .csv files corresponding each to fake and real news, both contain 4 variables, namely *title, text, subject* and *date*, with 21.417 records pertaining to the real class and 23.381 to the fake one, which also has 3% of empty values.
An initial look at the files comprising the dataset and their variables show that all 4 variables are of string type, this is especially relevant for *date*, which for example can be "December 31, 2017"; this might need later formatting if there are intentions to take the date variable into account for building the classifying model. Another issue that arises with this initial analysis is that the subjects of the *True* and *Fake* datasets don't match, this needs to be addressed, otherwise the classifiers would just have to look at the subject to immediately predict the target variable.

Before starting to preprocess the data, it is mandatory to deepen the knowledge about them. The easiest way to understand the datasets better is to look for duplicated data especially when they should be unique. Since the dataset is an aggregation of news articles the attributes *title* and *text* are the only ones supposed to be unique. When counting the distinct values of each attributes on *Fake* and *True* dataset and comparing them to the total number of values, it shows those results:

Table 1: Number of unique values

	Fake	**True**
Unique titles	17,903	20,826
Unique texts	17,455	21,192
Total rows	23,481	21,417

This table shows that 25% of fake news are just copies of other fake news whereas it represents only 1% of the true news. It also highlights that for the fake news it appears that there are different titles for the same content although it seems to be the opposite for the true news. However, the duplicated data won't be dropped even though it can create bias by overfeeding some models or affect the general capacities. This choice can be explained by an ongoing debate in NLP (Natural Language Processing). Some specialists argue that duplicated data is part of the dataset because it represents the distribution. In fact, some classifying models can consider it as a pattern and use this to class duplicated articles in the right class. To begin the preprocessing phase, the first step is to merge both datasets into a single one and create a new variable to denote the target variable *fake*, which has a value of 1 for the news articles coming from the fake file and 0 otherwise. This new dataset contains 44,888 observations, with no null values except for that 3% of empty values on the fake texts.

To address the subject issue, the *regex_replace* function is used to turn every news belonging to *politicsNews* into *politics* and *worldnews* into *News*, this way no longer having subjects that appear only on real news, thus avoiding possible bias in classification later on. Regarding date, *regex_replace* is again used to prepare the date string to be split into year, month and day and putting each value into a new different column. Then, since the month is present either in full and abbreviated form, it is transformed into its full form with the help of function *from_unixtime* and then turned into its numeric counterpart. Finally , these values are concatenated to form a single and coherent date format as the following example, "2017-12-31".

To better understand the previous changes, information can be extracted with the use of *sqlContext.sql*, which allows to perform SQL queries on a temporal view created based on the data at hand, its results can be seen in table 2 and 3, before and after renaming the subject.

Table 2: Subject before renaming

Subject	Fake	True
politicsNews	0	11,272
worldnews	0	10,145
News	9,050	0
politics	6,836	0
left-news	4,456	0
Government News	1,568	0
US_News	783	0
Middle-east	778	0

Table 3: Subject after renaming

Subject	Total
News	19,195
politics	18,108
left-news	4,456
Government News	1,568
US_News	783
Middle-east	778

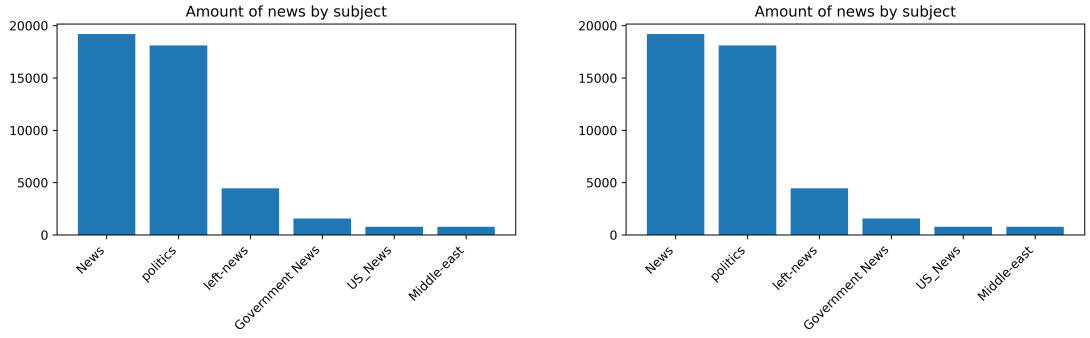
In addition to the tables, the information from the queries can then be plotted to provide a more clear visualization of the effect of transforming the subject (See figure 1). Also, now having a uniform format for date, it is possible to visualize its trends and look for a pattern regarding the amount of fake news depending on the period of time. This can be seen in figure 1.

2.1 Natural Language Processing

In order to prepare our raw data to be interpreted by the subsequent machine learning models to be used, it has to be fed into them in a machine-readable format. The first step to achieve this is to clean the documents inside our corpus. regexp_replace is used again but this time with a regular expression aimed at leaving only words in the observations from *title* and *text*, removing any symbol or number. In addition to this, *stopwords* are also to be removed, this are very frequent words in a particular language, for example in english "the", "on", "it", these words are very frequent but have little to no statistical significance inside the language model of a particular context, so they take a part of the probability mass of a certain word to appear in a text, by removing them, we can focus entirely on the words related more closely to task at hand, in this case, those related to fake news. This list of cleaned words is then tokenized, which basically splits the text into its individual components, to be later used to build a probabilistic language model, count their frequency inside the document or be grouped together in chunks of 2 to n-gram models, allowing to better capture word order, and thus, better context and meaning of how the words are used.

Having the words turned into tokens, it is now possible to feed them into a vectorizer algorithm, this is the part where we obtain the input that actually goes into the classification or clustering models. For this, *countVectorizer* from PySpark is used, paying attention to the attributes *minDF* and *maxDF*, which control the threshold at which

a certain item will be considered or not based on its relative frequency inside the whole collection of documents, in our case, minDf was set to .01 and maxDF to .9, this means that words appearing in less than 1% of the article news are discarded, as well as those appearing in over 90%, this is done to capture only the most important words, in the case of the rare ones, having such a low probability in the model, they are mostly irrelevant and those that appear most frequently can easily lead to bias and overfitting of the classification models.



(a) Before renaming

(b) After renaming

Figure 1: Amount of news by subject

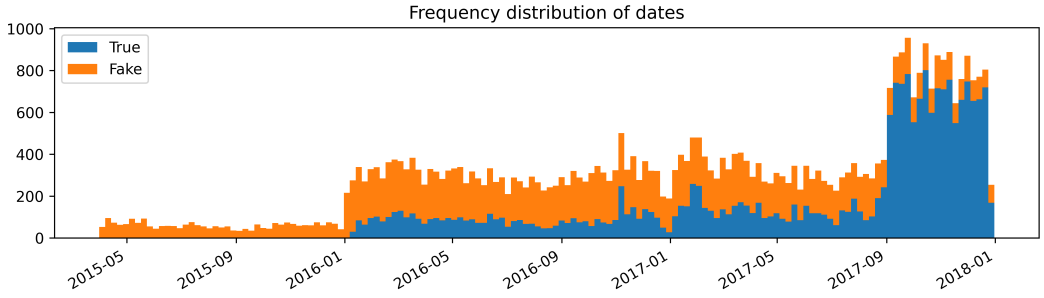


Figure 2: Frequency distribution of dates

Another useful thing about the vectorized data is that a number of statistical analysis can be performed with them, even if they come from natural text, they can now be treated just like any numerical feature, such is the case of a correlation analysis, with which it is possible how much influence a particular variable has with another one. For this case, those in *subject*, *date* and the target variable were considered, the other textual ones were not considered due to the large amount of features they contain, coming from a big vocabulary, so their visualization on a correlation matrix would be impossible. From figure 3 we can see how the subjects ”politics” and ”news” are heavily correlated between them, this likely because they contain a large portion stemming from the same dataset. We can also see some correlation between the target variable and the year, confirming the suspicions about how the period of time in which only fake news were reported could bias the models.

A further analysis was to look at word frequency for the different datasets as well as at the full one and visualize these words with a wordcloud that highlights the most frequent one. It is worth noting that most language corpus follow a zipf law and as such, their overall distribution is concentrated in a few elements while the 99% left of the distribution has a very low frequency, as such, it is expected to find similar words across fake and real datasets, nevertheless hoping to see some distinct words that appear only in one of them.

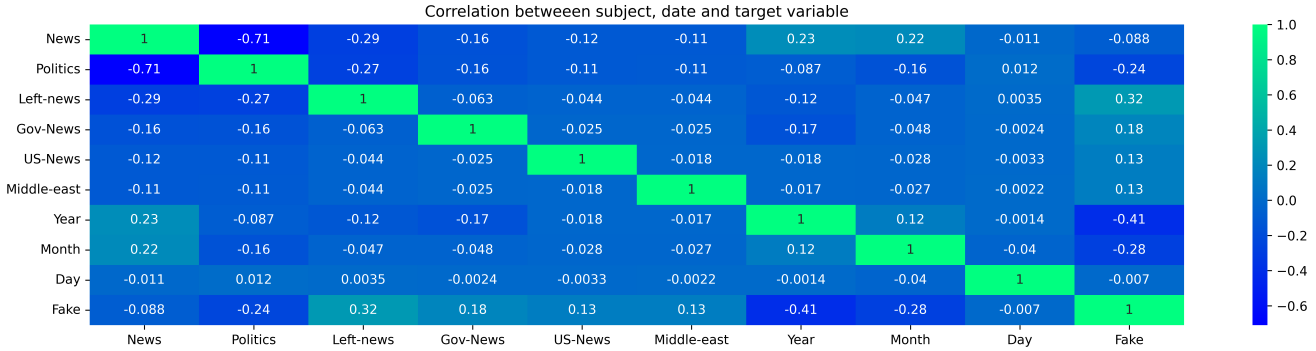
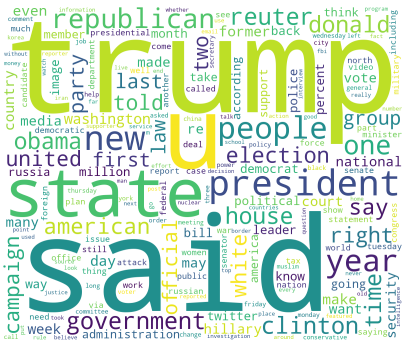
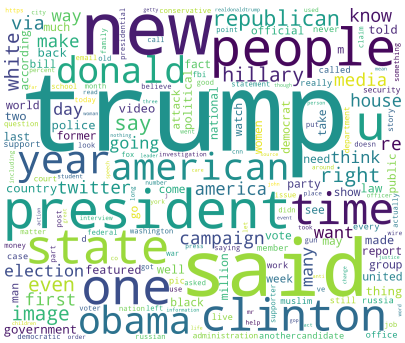


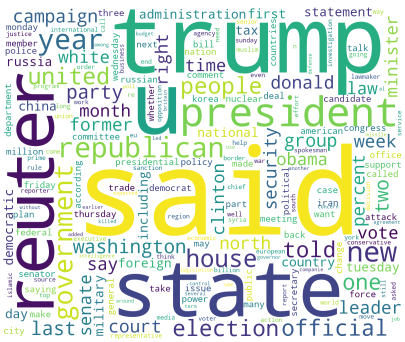
Figure 3: Correlation matrix between different *subjects*, *date* and *fake* variable



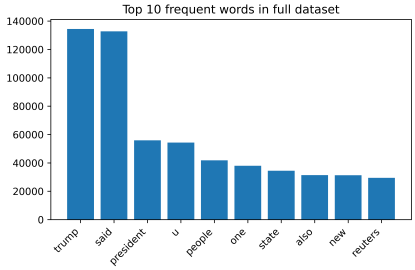
(a) Full dataset



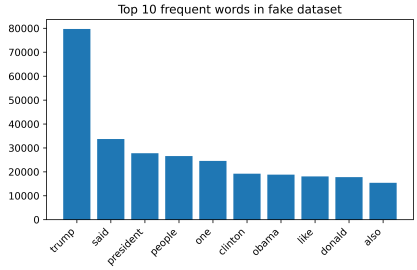
(b) Fake dataset



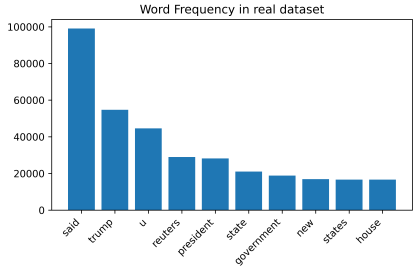
(c) True dataset



(d) Full dataset



(e) Fake dataset



(f) True dataset

Figure 4: Wordclouds for different datasets and their most frequent words

In figure 4 it is noticeable that some words that are common, such as “Trump” and “president”. In the *True* dataset, we can quote “Reuters”. Reuters is one of the largest and most famous news agencies in the world. Journalists use it because it is considered as a reliable source or because they work for this media. These two reasons explain why “Reuters” appear so much in the real news and not that much in the fake ones.

On the other hand, we can notice that “Clinton” and “Obama” are very frequent in the *Fake* dataset and way less in the *True* one. It’s difficult to really explain why and we can only suppose reasons and purposes of such a high use of these. Nowadays, fake news are a way to manipulate public opinion and even more during election periods. The fastest and easiest way to discredit or praise a candidate is to attack them personally. Indeed, it will be easier for electors to remember personal facts on a candidate whether it is positive or negative than the election manifesto which can be complex. For instance during the French presidential election of 2017, Emmanuel Macron was

targeted by a fake news campaign concerning his personal love life in order to discredit him and some people still take those fake news into account. Since candidates are related to their political parties, influencing electors on the candidates themselves or the outgoing president might affect their votes.

3 Clustering - Topic modeling

The PySpark clustering library includes five different algorithms: *K-Means*, *Bisecting K-Means*, *Latent Dirichlet Allocation*, *Gaussian Mixture Model* and *Power Iteration Clustering*. We used the first 2 to perform clustering on the text variable while the third one was applied for topic modeling, the main difference being that LDA does not use distance measures, instead relying on producing a distribution of groupings per item, instead of a distinct cluster as in the other methods. At the end though, for natural language they serve similar purposes, to group observations based on their underlying topics, to get a sense of the main points of discussion inside our data.

3.1 K-means

Before running K-Means, we execute calculate the silhouette score in order to find the optimal value of k clusters: in a range going from two to ten, we found that the optimal value of k was 4, with a silhouette score equal to 0.23. Then the text data is partitioned into 4 k-means clusters.

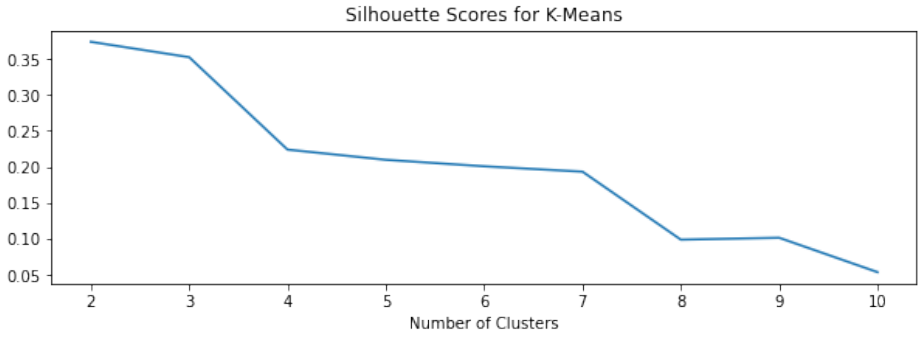


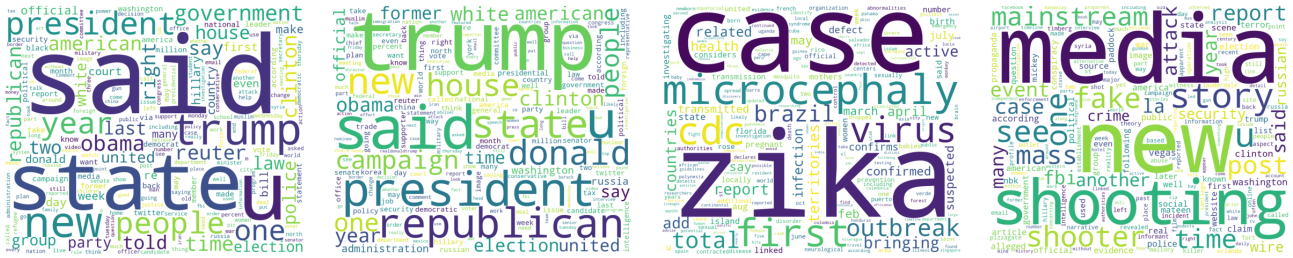
Figure 5: Elbow method with silhouette score for K-Means.

The first cluster was composed by 29,764 rows, with words like ”Trump”, ”Obama”, ”state” and ”government” being the most frequent as we can see from the wordcloud in figure 6(a). This means that this cluster contains general news about politics in the United States (the major part of the dataset). The second cluster was composed by 7,416 rows and the most frequent words were ”Trump”, ”Clinton”, ”Campaign” and ”House”; meaning that those news had been published during the election campaign in the United States.

The third cluster was composed by 7,437 rows, and this is were we start to see a different theme. The most frequent words are ”Zika”, ”Virus”, ”Brazil” and ”Bringing”; this means that this cluster is composed by those news related to the spread of the Zika virus and the relative fever. The fourth cluster was composed by only 326 rows: as the previous cluster, it also depicts a different them. The most frequent words were ”Media”, ”Shooting”, ”Mass” and ”Attack”: this means that this cluster is composed by those news related to mass shootings in the United States.

3.2 Bisecting K-Means

The procedure to execute the Bisecting K-Means has been more or less the same done for the K-Means, with the only difference being that this time the elbow method with the silhouette score returned 5 as optimal k.



(a) Cluster 0 (b) Cluster 1 (c) Cluster 2 (d) Cluster 3

Figure 6: Wordclouds for the different clusters found by K-means

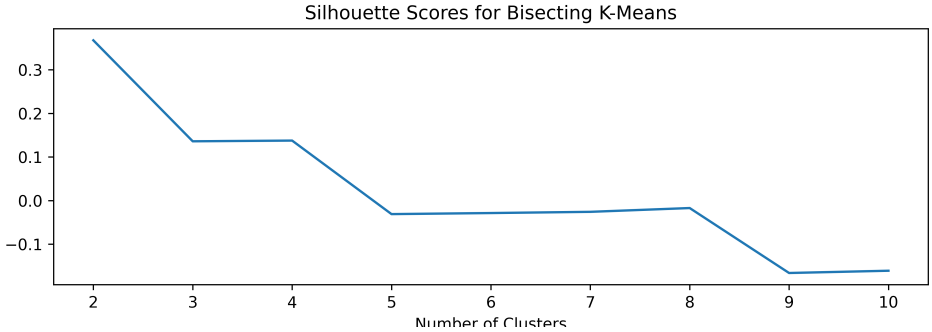
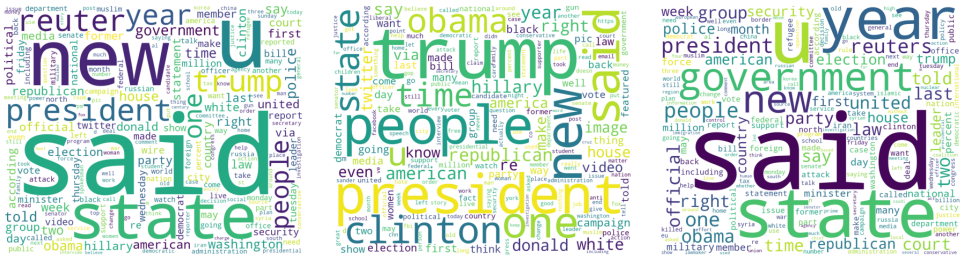


Figure 7: Elbow method with silhouette score for Bisecting K-Means.

The resulting clusters, differently from the K-Means ones, were more balanced, in terms of rows composing clusters. In fact, the first was composed by 16,719 rows, the second by 9,431, the third by 9,589 rows, the fourth by 5,896, while the last one contained 3,218 rows. However, their contents were not as insightful as those before, since this time the words related to each cluster are fairly homogeneous, with words like ”Trump”, ”President” or ”said” being the most frequent and appearing in practically every cluster, as seen in Figure 8. This shows that the performance of Bisecting K-Means for our particular task at hand is worse than that of K-means, at least on a subjective and qualitative level.



(a) Cluster 0 (b) Cluster 1 (c) Cluster 2

Figure 8: Wordclouds for the different clusters found by Bisecting K-means

3.3 Latent Dirichlet Allocation

The LDA algorithm works by stating that each document in a corpus is a combination of a fixed number of topics. A certain topic then has a probability of generating various words, where the words are all the observed words in the corpus. So each document can be ”classified” as belonging to a certain topic by looking at the highest co-occurrence of words and their distribution across the corpus.
Then we run the algorithm, printing the lower bound of the log likelihood, equal to -91114230.74077255, and the upper bound of the perplexity measure, used by convention in language modeling to represent how well a model predicts a certain sample, this was equal to 8.677600329751838, we also plotted the resulting topics described by their top-weighted terms as seen in figure 9.

```
topic 1: ['eu', 'britain', 'said', 'may', 'british']
topic 2: ['muslim', 'refugees', 'muslims', 'united', 'states']
topic 3: ['trump', 'said', 'u', 'president', 'russia']
topic 4: ['french', 'france', 'paris', 'macron', 'emmanuel']
topic 5: ['obama', 'president', 'cuba', 'first', 'trip']
topic 6: ['energy', 'oil', 'coal', 'said', 'gas']
topic 7: ['trump', 'said', 'u', 'administration', 'order']
topic 8: ['said', 'military', 'u', 'state', 'reuters']
topic 9: ['workers', 'food', 'state', 'work', 'people']
topic 10: ['police', 'video', 'protesters', 'twitter', 'people']
topic 11: ['fbi', 'comey', 'director', 'investigation', 'james']
```

Figure 9: LDA Topics.

As seen before, the terms with the highest probability weights assigned to each topic are somewhat repetitive and again represent the trending topics of the whole dataset, however, this time very promising results can be seen regarding the interpretability of the model results, this is in line with expectations, since LDA has shown state of the art results for several natural language tasks. As we can in figure 9. The model is finally able to capture the details inside of the text, being able to even detect very subtle pieces of context that tie the probabilities of certain words to others, and thus is able to group together words like ”mexico”, ”border”, and ”illegal” into the same topic for example.

4 Classification

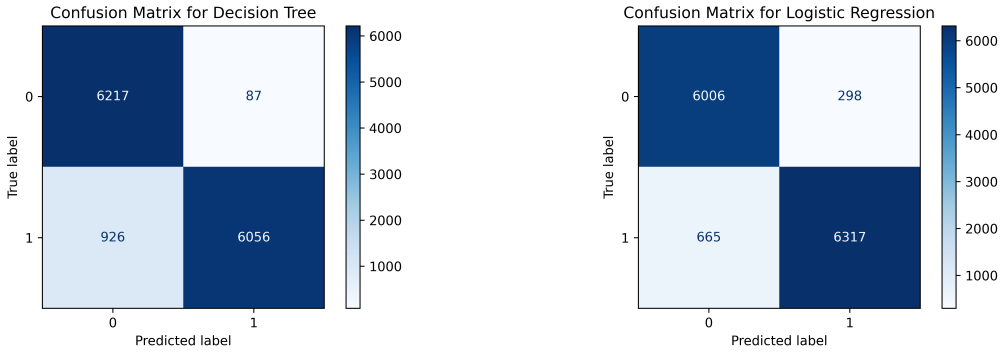
Regarding the main objective of this project, that is, identifying whether a news article is *fake* or *real*, the classification libraries of PySpark come in handy. In order to use them though, it is first necessary to perform one final decision and aggregation of the previously vectorized data to feed the models. This step is crucial, since it will determine the level of over/under fitting that the model might have and in turn, its generalization capabilities.

Based on the previous sections, it is decided to put together a vector containing the title or the article, its subject and date. The logic behind this decision being that the aim is to be able to detect a news based hopefully only on a quick snapshot of it, without going into too much detail on the actual contents of the article, this allows for a quicker and lightweight performance. Regarding the *subject* and *date* variables, even though there was some degree of apprehension at first concerning their use and a possible inclusion of bias to the model due to the the subjects that are only present in one of the two datasets or the period of time going from March 2015 to January 2016. However, as seen in figure 3, the correlation matrix shows that they are actually not that heavily correlated to the target variable, and are thus safe to include as input data.

Having the data ready, the last step is to split it into training and testing set, then it is now possible to feed into a model, the classifying models to use will come from PySpark’s *MLib* library, in particular they are Decision

Tree, Logistic Regression, Support Vector Machines and Random Forests. At first, these models are created with the standard parameters and fitted to the training data, then a set of predictions is obtained from the testing set. Depending on the model used, they can contain the final predictions, From this predictions, the probabilities and final predictions for each observation can be observed, and extrapolating them most of the classifying results can be produced, such as accuracy, precision, recall, f1-score, area under the ROC curve and area under the Precision-Recall curve.

Another use that can be extracted from the predictions made on the testing set is to visualize them with the help of a confusion matrix, which have been plotted with the help of Sklearn's libraries. An example of this can be seen in figure 10.



(a) DT (b) LR

Figure 10: Confusion matrix for Decision Tree and Logistic Regression

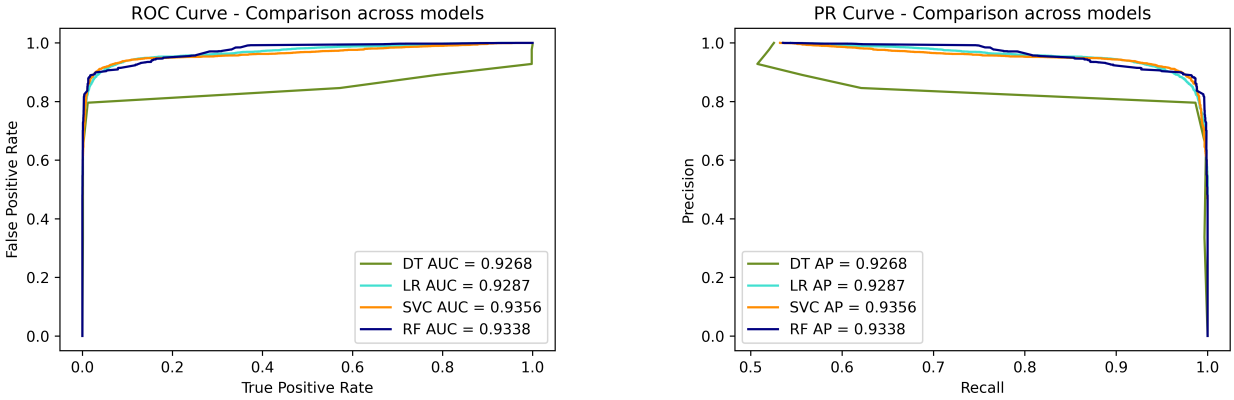
Continuing with Sklearn's libraries, it is also possible to pass the probabilities and predictions to compute the True Positive Rate as well as the False Positive Rate to compute the ROC curve, as well as the Precision-Recall curve.

4.1 Comparing classifiers

To get a better understanding of how the different classifiers with their default parameters behave, their metrics performance is compared in table 4. as well as their resulting curves in figure 11.
From fig x, it is possible to see that most classifiers perform fairly well, and with very similar metrics, this either indicates that the models have correctly learned the patterns needed to accurately classify a news article, or that some form of bias leaked through the data, be it through the subject or date variable, or by the contents of the articles themselves, proving to be too "easy" to classify, providing clear clues as to the target class, as we have seen with very frequent words like "reuters" that can be found predominantly on the real dataset. Either way, a further analysis is conducted to rule out this unfavorable possibilities.

4.2 Comparing input data

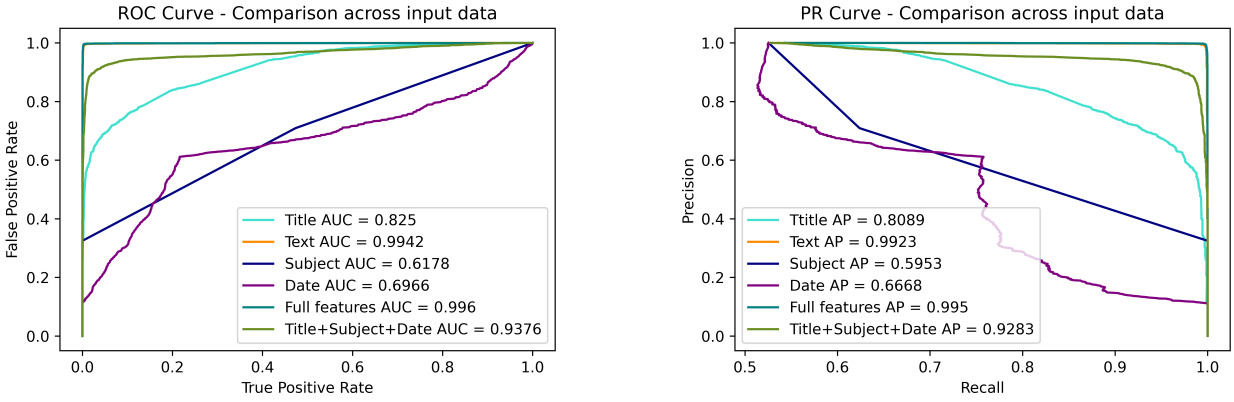
In order to assess how informative the previous classifying results actually are, and to see if the models are not biased, a subsequent analysis is performed, this time to see how different training vectors might influence the fit of the model and its generalization capabilities.To do this, the best performing model from before, SVC has been chosen and fitted with the different data vectors available from the preprocessing phase. Its results can be appreciated in figure 12.



(a) ROC-AUC (b) PR-AP

Figure 11: Performance comparison between different models

These results indicate that there was indeed some bias present on the previous models, especially with the date variable, which was able to get an area under the ROC curve of almost .70, a result much better than expected, considering that it should be practically impossible to classify a news article based solely on the date. Also, as expected, classifying based on the text content leads to near-perfect results, again indicating an overly optimistic result that might perform poorly on slightly different datasets, since, as we have seen in previous sections, most of the news and topics contained within this dataset are very monotonic and don't deal with a wide array of topics. As such, it is perhaps with the title variable that the algorithms could build a more generalizing model, which could be able to detect fake news in different scenarios.



(a) ROC-AUC (b) PR-AP

Figure 12: Performance comparison across different data inputs

4.3 Hyperparameter tuning

The last thing to test for the classification task is hypertuning, in an attempt to improve upon the previously implemented models. To do this, PySpark provides the paramGridBuilder and crossValidator methods, both from the tuning section of the machine learning packages, with them it is possible to iterate over a defined set of parameters to try while fitting the models with them and their different combinations, this is then tested with a 5-fold cross-validation technique and the best scoring (based on AUC) model is returned. Overall, decision trees and Random Forests were the methods that benefited with this approach, although just slightly, their performance increased, while logistic regression and support vector machines were unable to find a better model than that of the default parameters. The complete metrics for baseline and tuned models can be found in table 4.

Model performance

		Accuracy	Precision	Recall	F1-Score	AUC	AP
DT	Baseline	0.9238	0.9310	0.9238	0.9237	0.9268	0.9553
	Tuned	0.9483	0.9500	0.9483	0.9483	0.9497	0.9603
	Improvement	2.65%	2.04%	2.65%	2.65%	2.47%	0.52%
LR	Baseline	0.9275	0.9290	0.9275	0.9276	0.9345	0.9287
	Tuned	0.9275	0.9290	0.9275	0.9276	0.9345	0.9287
	Improvement	0%	0%	0%	0%	0%	0%
SVC	Baseline	0.9341	0.9363	0.9341	0.9341	0.9356	0.9466
	Tuned	0.9341	0.9363	0.9341	0.9341	0.9356	0.9466
	Improvement	0%	0%	0%	0%	0%	0%
RF	Baseline	0.9312	0.9368	0.9312	0.9312	0.9338	0.9580
	Tuned	0.9552	0.9572	0.9552	0.9552	0.9567	0.9702
	Improvement	2.57%	2.17%	2.57%	2.57%	2.45%	1.27%

Table 4: Performance across models before and after tuning

5 Frequent Pattern Mining

In the last section a final analysis has been done on part of the dataset (only the fake news set) in order to check for *frequent pattern* e *associative rules* between the items. To do so the ”text” section has been divided in token and, since it is required by the algorithm *FP-Growth*, every word has been made unique for each record.
The most important parameter to be set is the *minimum support*, this value represents the minimum lower bound for the *support* measure, used to prune candidate association rules.
With a minimum support greater than $0,4$ no associative rules were found; with a minimum support equal to $0,25$ the model was able to create 31 associative rules, mostly having as antecedent the world: trump, president, people, like, image (as shown in the figure 13 with the associated level of confidence, lift and support).
As expected from the previous analysis of the dataset the frequent pattern found are mostly covered by the words shown before in the *worldcloud*. In the top ten of the most frequent pattern the items are made up of only one word, except for the world ”donald” and ”trump” which are a frequent pattern both separated and joined.

antecedent	consequent	confidence	lift	support
[like]	[one]	0.631547736706859	1.203267873061668	0.27225086276681865
[like]	[trump]	0.6058509586874876	1.1355048990939889	0.26117336287333304
[like]	[people]	0.6127693220003954	1.2579645549436962	0.2641557666908099
[like]	[said]	0.613757659616525	1.1271913950594254	0.26458182437902095
[trump]	[president]	0.61151481274455	1.3627861915996329	0.3262749776319714
[trump]	[donald]	0.7249061726423381	1.8647821983875843	0.38677516935793105
[via]	[one]	0.6216432509336653	1.184397170441112	0.29785692982829876
[via]	[said]	0.6436066156855771	1.1820102407477449	0.30838055472711007
[via]	[featured]	0.6219989329539392	1.7203555214897368	0.2980273529035831
[via]	[image]	0.6546327583140672	1.7089184151250663	0.31366367006092627
[donald]	[trump]	0.9949583516001753	1.864782198387584	0.38677516935793105
[one]	[said]	0.6330059258056661	1.162541634161564	0.33223978526692516
[people]	[one]	0.6255575964313829	1.1918550487735196	0.3047164586084956
[people]	[said]	0.6326423510889531	1.1618739141164958	0.30816752588300456
[featured, via]	[image]	0.983702644746247 4	2.567955152356709	0.29317029525797794
[image]	[featured]	0.9014570125681237	2.4932945489024783	0.34531975629500233
[image]	[one]	0.6573239906573239	1.2523785522134954	0.2518000937326914
[image]	[via]	0.8188188188188188	1.7089184151250663	0.31366367006092627
[image]	[trump]	0.7075964853742631	1.3261995614644517	0.27105790123982787
[featured]	[via]	0.8242988451567287	1.7203555214897368	0.2980273529035831

Figure 13: Associative rules found with a Minimum Support equal to $2,5$

6 Conclusion

The aim of this project was to analyze a dataset made up of news both fake and true through the use of Natural Language Processing techniques together with machine learning.
After getting familiar with the dataset, the first important information has been collected by studying the distribution of news through the years and by deleting the element of the text useless for our scope (articles, numbers, punctuation). Then the data has been divided in single words (token) in order to be studied through specific algorithms. From the distribution of the tokens it has been evident how the news were mostly related to politics, and the word *Trump* was the most used.

A second analysis has been conducted by using three algorithms of clustering and topic modeling: *K-Means, Bisecting K-Means and Latent Dirichlet Allocation*. Before the run of the first two functions, the silhouette has been calculated and the optimal value for the number of clusters has been found equal to 4 for the first algorithm and 5 for the second one. The *K-means* built 4 different clusters in which the most conspicuous had half of the data related only to politics. In the others unique topics (mass shootings, politics, virus) were grouped, meaning that any of division was significant. *Bisecting K-Means* built more balanced group but the themes were not well-grouped. *Latent Dirichlet Allocation* finally showed optimal results, confirming its favored use for natural language tasks, being able to capture the context inside the words being used and producing very specific topics inside the corpus.
For the classification part, the processed data from before was used to be fed into several classification algorithms, namely decision trees, logistic regression, support vector machines and random forests. They all showed very good results above 90% for every metric, and thus, a subsequent analysis was performed to rule out bias, overfitting and overall to not be mislead by overly optimistic results. So the different variable available from before were used to classify again with the best performing algorithm, svc, this in fact showed that the previous models indeed suffered slightly from bias, coming especially from the *date* variable, which contained some periods of time with unbalanced proportions of fake and real news. In the end, a hypertuning of parameters phase was also tested to see if the models could improve, only decision trees and random forests saw moderate improvements, while the rest remained the same.
In the last section a brief study on the frequent pattern has been conducted and it confirmed the results found before. Some associative rules has been built and also in this case the prevalent theme was politics.