Contents

Abstract
The field of reviews in the commercial and marketing application has achieved growth in recent years. The websites that are most used in these areas require reviews on the products and activities or services that they make available to the user. Our goal is to be able to recognize the veracity of reviews with different methods of machine learning and discriminate against fake reviews.

1 Introduction

The reviews today are increasingly held in consideration by users when they have to make a decision. The companies that make the content give much importance, for instance, to their online reputation and to digital marketing. Basically before making a purchase the reviews of other users are examined to see if the seller is reliable. Together with the fake reviews, the problem of the truth of online information expands if we consider identity verification, market protection, brand management, malicious behavior and data journalism and some organizations offer they help service given the importance of the web [1]. A huge number of sites provide the service of produce of fake reviews [2], [3], [4], [5].
"Now everyone has a license to speak, it's a question of who gets heard"[6]. This quote was by Aaron Swartz who talked about the transformation of the communication and the freedom of diffusion.
Web platforms today can diffuse a lot of information in a restricted time and it is difficult filtered the advice based on truth or fake. The safety, usability and reliability of some platforms are compromised by the prevalence of online antisocial behavior that can shape the others opinion [7]. Companies participate in customer with customer interactions in online public environments by do not shown their identities to increase interest, and decrease resistance and skepticism in the customers [8] [9]. The online interactions between client and business become increasingly pervasive and have an increase in fraud and deception [10]. A phenomena is the trolls, some previous work have demonstrated that trolls are born and not made with unique personality traits like machiavellianism, narcissism, psychopathy, direct sadism, vicarious sadism [11] and their motivation divided in two macro group: boredom, attention seeking, revenge and the other one fun, entertainment [12]. A dangerous method to produce fake news or to influence people, not only a review about a product or a service, is the sock puppetry phenomena [13]. These online modes are classified as malicious behavior [7].
A huge volume of reviews on an item then serves as a valuable source of information for a potential customer and it have a large effect on the popularity [14]. The form of electronic word of mouth among consumers is the source of information with the greatest impact in decision-making processes [15]. Consumers tend to refer to the reviews of others to minimize uncertainty and perceived risk [16]. A division of involvement of consumers shown that a low-involvement are affected by review quantity rather than quality while high-involvement consumers

are affected both by review quantity and by review quality [17]. Journalism has verified and written several facts about fake reviews [18], [19], [20], [21], [22]. An approach for the detection of fake reviews is proposed with the utilization of two different macro-features: the features of the text and the IP-address of the user for the detection of the people who fake and their modality [23]. When fake reviews are created by a group of bots or appropriate people, the impact is more substantial in terms of diffusion and belifs of the users, but the evidence has shown that the classification of fake reviews and reviewers is very difficult while classification fake reviewer groups is much easier [24]. Expedia.com asks the customer posting a review to have had an actual reservation at the hotel, TripAdvisor.com does not, thus creating an environment that can allow fake reviews. The reviews on TripAdvisor.com tend to be potentially fake respect to Expedia.com [25]. The TripAdvisor detection of fake reviews, it goes through a tracking system that examines a lot of of different attributes, from basic data points, the IP address of the reviewer, the screen resolution of the device that was used to submit the review [26]. An hotel near a competitor tends to have more false and negative reviews than an hotel without close competitors [27]. The dataset used in this project is the op_spam_v1.4 [28], [29]. The previous version of the dataset is used in [30], where the researchers is focused on the positive review spam with N-Gram and SYN features with SVM classifier that perform 90.1% The difference in datasets to train the model for detecting false reviews is a problem for creating an abstract model that can recognize the fake. It is a consequence of the different modality of the creation of the dataset and the features that could be extracted uniquely. It is shown with a comparison between the classification of the real-life data, that is considerably harder, than the AMT pseudo fake reviews data generated. The Turkers have not the same psychology state while writing such reviews as that of the authors of fake reviews who have real businesses [31].

To resume fake reviews detection:

- Lexical features
- Content and style similarity of reviews from different reviewers
- Semantic inconsistency
- Public data available from Web sites
- Web site private/internal data
- Product related features
- Complex relationships among reviewers, reviews, and entities

In this project the aim is to find a method to detect fake reviews using Machine Learning library like scikit-learn [32] and the Natural Language Toolkit (NLTK) platform to work in the human language and translate it into processable data [33].

2 Methods

2.1 Libraries

The following are the main libraries used for the implementation of the machine learning script for the spotting of fake reviews.

- NumPy is an open source extension of the Python language that add support for the vector, for the multidimensional matrix (also for big dimension) and with mathematics function, in high level, with do operations [34].
- Pandas is an open source easy-to-use data structures and data analysis tools for the Python programming language [35].
- Scikit-learn is an open source library of automatic learning for Python language. It contains algorithms for classification, regression, clustering, SVM, logistic regression, bayesian classifier, k-mean, DBSCAN. It's projected for operating with Numpy and Scipy libraries [32].
- Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms [36].
- NLTK is a leading platform for building Python programs to work with human language data. It provides along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, semantic reasoning and others [33], [37].

2.2 Description of the dataset

The text files is formed with the reviews of different hotels in Chicago. Truthful positive reviews come from TripAdvisor from the 20 most popular Chicago hotels. The reviews were 6.977 and from these are eliminated [28]:

- 3130 non-5-star reviews
- 41 non-English reviews
- 75 reviews with fewer than 150 characters since, by construction, deceptive opinions are at least 150 characters long
- 1,607 reviews written by new users who have not previously posted an opinion on TripAdvisor

The deceptive positive ones are provided by Amazon Mechanical Turk https://www.mturk.com/. The instructions were that the Turkers work for the hotel's marketing department, and to pretend that their boss wants them to write a fake review to be posted on a travel review website. On this dataset, it is unclear whether Turkers are representative of the general population that generate fake reviews

[38]. Another similar problem is the fact that the model that will be shown is trained only for Chicago hotel [38]. The number of truthful and deceptive opinions is balanced by selecting 400 of the remaining 2,124 truthful reviews. In this dataset are collected also 400 truthful negative reviews from Expedia, Hotels.com, Orbitz, Priceline, TripAdvisor and Yelp and 400 deceptive negative reviews from Mechanical Turk [29]. Its combination yields a dataset with a total of 1600 reviews.

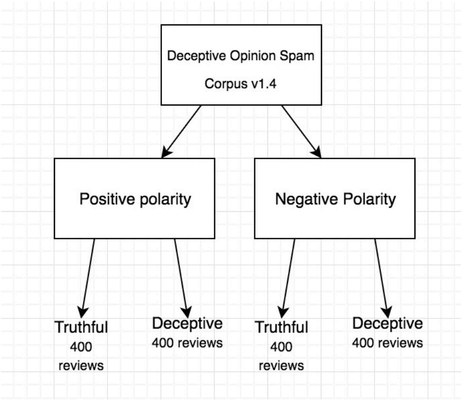


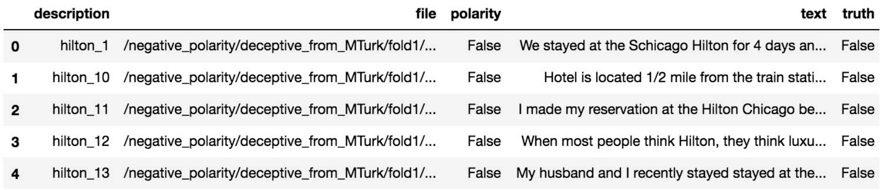
Figure 1: dataset

The original Dataset consists in txt files spread over several folders. Each file have attributes such as:

- hotel name
- polarity
- text of the review
- truthfulness

For the construction of a dataset useful for our objective, the .txt files are merged into a single .json file to facilitate processing and subsequent implementation. This .json file is then converted into a dataframe thanks to the Pandas library [35] with the following columns:

- text (str): the reviews
- description (str): hotel name and numeration of the review
- truth (binary): true or false
- polarity (binary): based on sentimental analysis, true or false



	description	file	polarity	text	truth
0	hilton_1	/negative_polarity/deceptive_from_MTurk/fold1/...	False	We stayed at the Schicago Hilton for 4 days an...	False
1	hilton_10	/negative_polarity/deceptive_from_MTurk/fold1/...	False	Hotel is located 1/2 mile from the train stati...	False
2	hilton_11	/negative_polarity/deceptive_from_MTurk/fold1/...	False	I made my reservation at the Hilton Chicago be...	False
3	hilton_12	/negative_polarity/deceptive_from_MTurk/fold1/...	False	When most people think Hilton, they think luxu...	False
4	hilton_13	/negative_polarity/deceptive_from_MTurk/fold1/...	False	My husband and I recently stayed stayed at the...	False

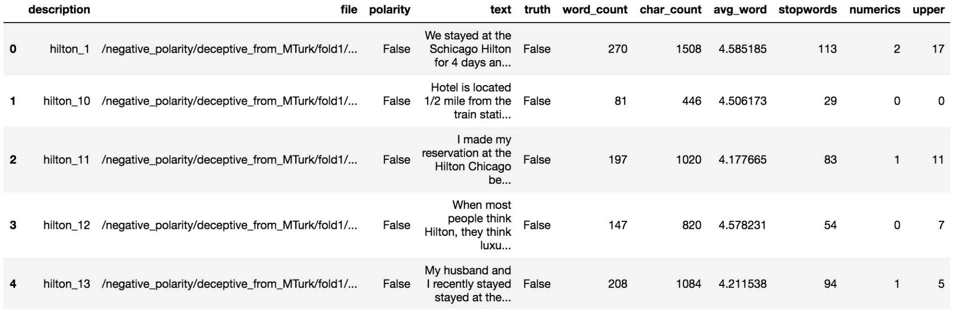
Figure 2: File .json

2.3 Basic Feature Extraction

2.3.1 Part I

The first elaboration on this dataset is been possible with the extraction of the easily features:

- Number of words
- Number of characters
- Average word length
- Number of stopwords
- Number of numerics
- Number of uppercase



	description	file	polarity	text	truth	word_count	char_count	avg_word	stopwords	numerics	upper
0	hilton_1	/negative_polarity/deceptive_from_MTurk/fold1/...	False	We stayed at the Schicago Hilton for 4 days an...	False	270	1508	4.585185	113	2	17
1	hilton_10	/negative_polarity/deceptive_from_MTurk/fold1/...	False	Hotel is located 1/2 mile from the train stati...	False	81	446	4.506173	29	0	0
2	hilton_11	/negative_polarity/deceptive_from_MTurk/fold1/...	False	I made my reservation at the Hilton Chicago be...	False	197	1020	4.177665	83	1	11
3	hilton_12	/negative_polarity/deceptive_from_MTurk/fold1/...	False	When most people think Hilton, they think luxu...	False	147	820	4.578231	54	0	7
4	hilton_13	/negative_polarity/deceptive_from_MTurk/fold1/...	False	My husband and I recently stayed stayed at the...	False	208	1084	4.211538	94	1	5

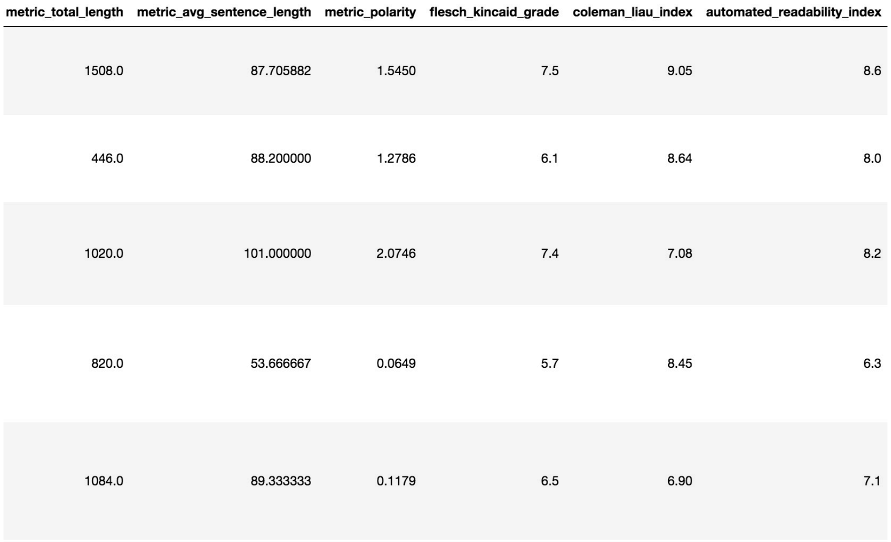
Figure 3: First elaboration

2.3.2 Part II

The second elaboration on this dataset is been possible in local for the verification of some peculiarities between them:

- Metric total length (int): total length of the review.

- Metric avg_sentence length (float): average length of the sentences in each review.
- Metric polarity (float): index of how positive or negative the polarity is.
- Flesch kincaid grade [39] (float): the Flesch Kincaid readability tests are readability tests designed to indicate how difficult a passage in English is to understand.
- Coleman liau index [40] (float): this is another readability index, it's based on the following formula: $0.0588*L-0.296*S-15.8$, where L is the average number of letters per 100 words and S is the average number of sentences per 100 words
- Automated readability index [41] (float): this readability index tells how difficult is a text for grade level, and it's calculated like 4.71*(characters/words) + 0.5*(words/sentences) - 21.43

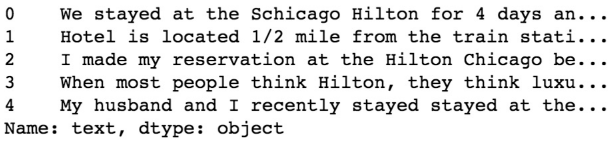


metric_total_length	metric_avg_sentence_length	metric_polarity	flesch_kincaid_grade	coleman_liau_index	automated_readability_index
1508.0	87.705882	1.5450	7.5	9.05	8.6
446.0	88.200000	1.2786	6.1	8.64	8.0
1020.0	101.000000	2.0746	7.4	7.08	8.2
820.0	53.666667	0.0649	5.7	8.45	6.3
1084.0	89.333333	0.1179	6.5	6.90	7.1

Figure 4: Second elaboration with the addition of indexes

2.4 Data processing

After processing the dataset, the processing that is carried out on the text that will subsequently be analyzed by the machine learning model is explained below. This processing includes different methods for structuring the text using NLTK [33]:

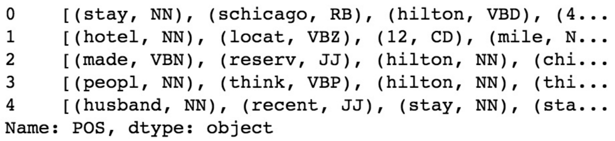


```
0    We stayed at the Schicago Hilton for 4 days an...
1    Hotel is located 1/2 mile from the train stati...
2    I made my reservation at the Hilton Chicago be...
3    When most people think Hilton, they think luxu...
4    My husband and I recently stayed stayed at the...
Name: text, dtype: object
```

Figure 5: Normal text

- Lowercase: typeface of small characters. For example, a, b, and c is lowercase and A, B, and C is uppercase.
- Removing punctuation: it does not add any extra information. However it removing all parts of it will help for the reduction of the size for the training data.
- N-grams: is a sequence of n items from a given sample of text or speech. Extract all 1-grams, 2-grams, 3-grams... and track word boundaries.
- Removal stopwords: refers to the most common words in a language, there is no single universal list of stop words used by all natural language processing tools Any group of words can be chosen as the stop words for a given purpose. For some search engines, these are some of the most common, short function words, such as the, is, at, which, and on. These are words which do not contain enough significance to be used without the algorithm.
- Stemming: usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes [42].
- Lemmanization: usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma [42].
- Tokenization: is the process of demarcating and possibly classifying sections of a string of input characters. The resulting tokens are then passed on to some other form of processing.
- Pos Tagging: reads text in some language and assigns parts of speech to each word (and other token), such as noun, verb, adjective...

This explanation has the sole purpose of specifying the sequential processes that will be, in part, performed subsequently in the transformation function (CountVectorizer and TF-IDF) of Sklearn for the features extraction and vectorization transform.



```
0    [(stay, NN), (schicago, RB), (hilton, VBD), (4...
1    [(hotel, NN), (locat, VBZ), (12, CD), (mile, N...
2    [(made, VBN), (reserv, JJ), (hilton, NN), (chi...
3    [(peopl, NN), (think, VBP), (hilton, NN), (thi...
4    [(husband, NN), (recent, JJ), (stay, NN), (sta...
Name: POS, dtype: object
```

Figure 6: Elaborated text

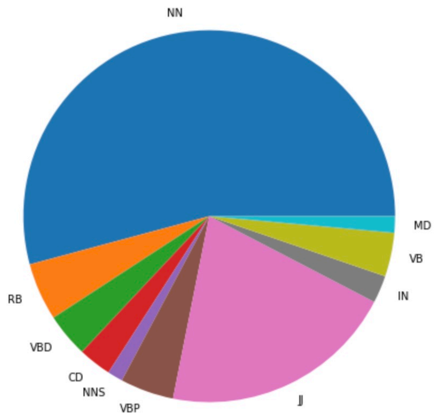


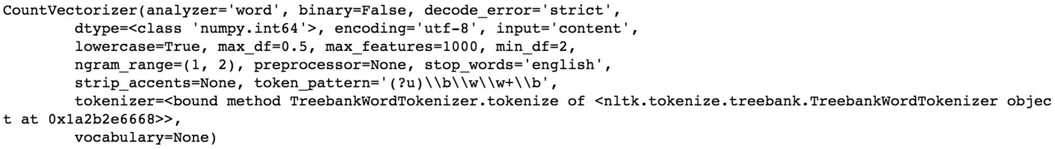
Figure 7: Pos tagging with $n > 1000$. Legend: NN: Noun, singular or mass; MD Modal; VB: Verb, base form; IN: Preposition or subordinating conjunction; JJ: Adjective; VBP: Verb, non-3rd person singular present; NNS: Noun, plural; CD: Cardinal number; VBD: Verb, past tense; RB: Adverb.

2.5 Word Embedding

Word Embedding are a group of methods for the features extraction from text files. With the utilization of these features it's possible the implementation in machine learning algorithms to study the Natural Language Processing (NLP). This method transforms the text into a vocabulary containing a vector, ready for insertion into the machine learning model. The process of converting NLP text into numbers is called vectorization in ML.

2.5.1 Count Vectorizer

The CountVectorizer transformer from the sklearn.feature_extraction model has its own internal tokenization and normalization methods. The fit method of the vectorizer expects an iterable or list of strings or file objects, and creates a dictionary of the vocabulary on the corpus. When transform is called, each individual document is transformed into a sparse array whose index tuple is the row (the document ID) and the token ID from the dictionary, and whose value is the count [43]. To sum up it counts the number of times a token shows up in the document and uses this value as its weight.

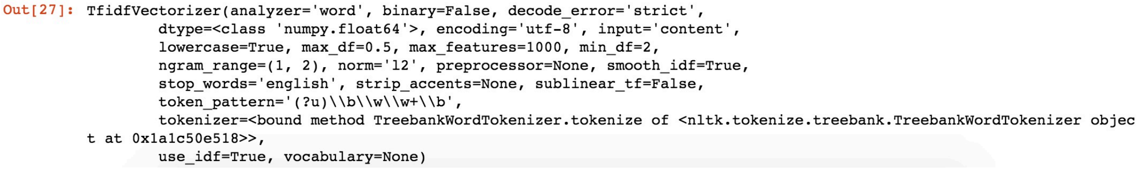


```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
        lowercase=True, max_df=0.5, max_features=1000, min_df=2,
        ngram_range=(1, 2), preprocessor=None, stop_words='english',
        strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=<bound method TreebankWordTokenizer.tokenize of <nltk.tokenize.treebank.TreebankWordTokenizer objec
t at 0x1a2b2e6668>>,
        vocabulary=None)
```

Figure 8: Parameters of Count Vectorized function

2.5.2 TF-IDF Vectorizer

TF–IDF, term frequency–inverse document frequency, encoding normalizes the frequency of tokens in a document with respect to the rest of the corpus. This encoding approach accentuates terms that are very relevant to a specific instance. TF–IDF is computed on a per-term basis, such that the relevance of a token to a document is measured by the scaled frequency of the appearance of the term in the document, normalized by the inverse of the scaled frequency of the term in the entire corpus [43].



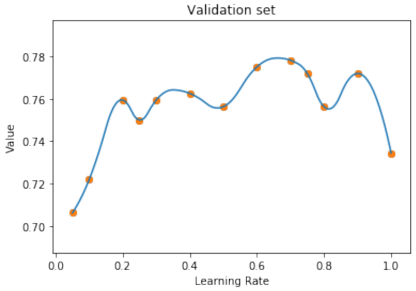
```
Out[27]: TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
                 dtype=<class 'numpy.float64'>, encoding='utf-8', input='content',
                 lowercase=True, max_df=0.5, max_features=1000, min_df=2,
                 ngram_range=(1, 2), norm='l2', preprocessor=None, smooth_idf=True,
                 stop_words='english', strip_accents=None, sublinear_tf=False,
                 token_pattern='(?u)\\b\\w\\w+\\b',
                 tokenizer=<bound method TreebankWordTokenizer.tokenize of <nltk.tokenize.treebank.TreebankWordTokenizer objec
         t at 0x1a1c50e518>>,
                 use_idf=True, vocabulary=None)
```

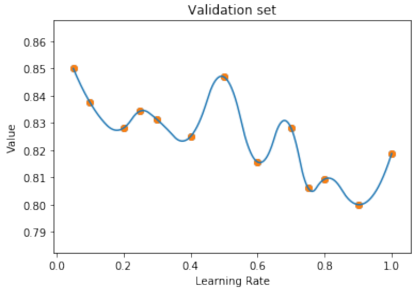
Figure 9: Parameters of TF-IDF Vectorized function

In both, the features used, specified the section Data Processing (2.4), are: lowercase, stopwords, tokenization, n-grams

2.6 Gradient Boosting Classifier

The Gradient Boosting Classifier (GBC) [32], [44] is a ensemble method that combines multiple decision trees to create a more powerful model. Gradient boosting works by building trees in a serial manner, where each tree tries to correct the mistakes of the previous one. The main idea behind gradient boosting is to combine many simple models, like shallow trees. Each tree can only provide good predictions on part of the data, and so more and more trees are added to iteratively improve performance. An important parameter of gradient boosting is the learning_rate, which controls how strongly each tree tries to correct the mistakes of the previous trees. A higher learning rate means each tree can make stronger corrections, allowing for more complex models. Adding more trees to the ensemble, which can be accomplished by increasing n_estimators, also increases the model complexity, as the model has more chances to correct mistakes on the training set. The main parameters of gradient boosted tree models are the number of trees, n_estimators, and the learning_rate, which controls the degree to which each tree is allowed to correct the mistakes of the previous trees. These two parameters are highly interconnected, as a lower learning_rate means that more trees are needed to build a model of similar complexity. In contrast to random forests, where a higher n_estimators value is always better, increasing n_estimators in gradient boosting leads to a more complex model, which may lead to overfitting. A common practice is to fit n_estimators depending on the time and memory budget, and then search over different learning_rates. Another important parameter is max_depth to reduce the complexity of each tree. Usually max_depth is set very low for gradient boosted models, often not deeper than five splits.
For instance, follow the validation test accuracy with 100 estimators on the left and 1500 estimators on the right with the Count Vectorized feature extraction in the Gradient Boosting Classifier:

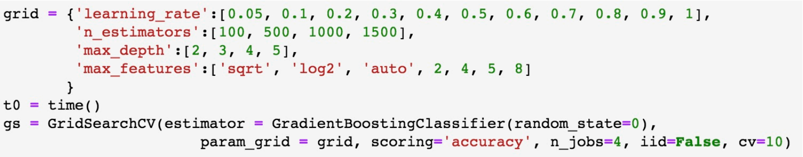




In the left figure: with n_estimators = 100, the accuracy increases with the increase of the learning rate.
In the right figure: with n_estimators = 1500, the accuracy decreases with the

increase of the learning rate.

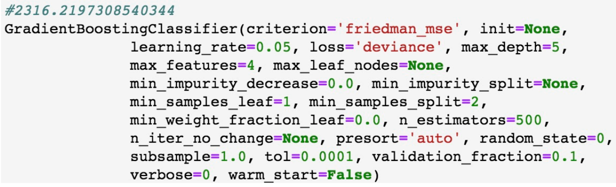
The function Gradient Boosting Classifier on the Sklearn library contains different parameters and for its implementation is used the GridSearchCV.
The machine for GridSearchCV is been a MacBook-Pro 2018, quad-core, with a processor 2,3 GHz Intel Core i5 and a memory 8 GB 2133 MHz LPDDR3.



```
grid = {'learning_rate':[0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1],
        'n_estimators':[100, 500, 1000, 1500],
        'max_depth':[2, 3, 4, 5],
        'max_features':['sqrt', 'log2', 'auto', 2, 4, 5, 8]
       }
t0 = time()
gs = GridSearchCV(estimator = GradientBoostingClassifier(random_state=0),
                  param_grid = grid, scoring='accuracy', n_jobs=4, iid=False, cv=10)
```

Figure 10: Parameters research

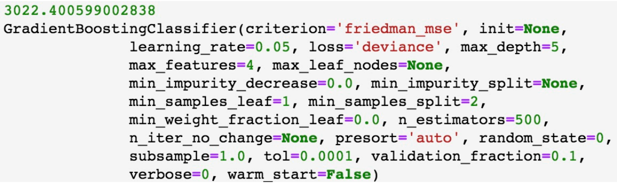
These are the parameters that the GridSearchCV returns with the count vectorization in the Gradient Boosting Classifier. The search tooks 2316 seconds.



```
#2316.2197308540344
GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.05, loss='deviance', max_depth=5,
              max_features=4, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, n_estimators=500,
              n_iter_no_change=None, presort='auto', random_state=0,
              subsample=1.0, tol=0.0001, validation_fraction=0.1,
              verbose=0, warm_start=False)
```

Figure 11: Parameters GBC with Count Vectorized

These are the parameters that the GridSearchCV returns with the TF-IDF vectorization in the Gradient Boosting Classifier. The search tooks 3022 seconds.



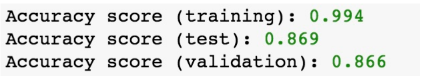
```
3022.400599002838
GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.05, loss='deviance', max_depth=5,
              max_features=4, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, n_estimators=500,
              n_iter_no_change=None, presort='auto', random_state=0,
              subsample=1.0, tol=0.0001, validation_fraction=0.1,
              verbose=0, warm_start=False)
```

Figure 12: Parameters GBC with TD-IDF

3 Results

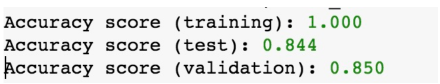
3.1 Accuracy

The three accuracy scores of the GBC Count Vectorized transformation

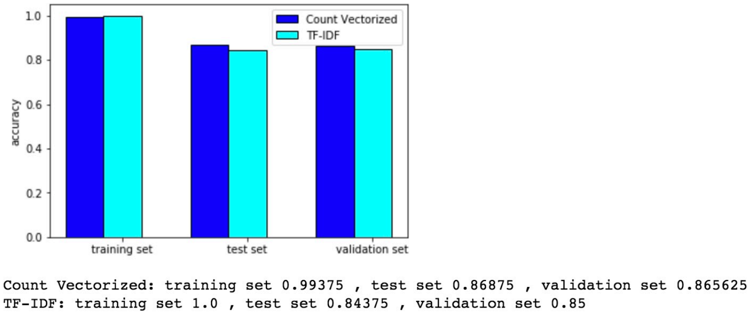


```
Accuracy score (training): 0.994
Accuracy score (test): 0.869
Accuracy score (validation): 0.866
```

The three accuracy scores of the GBC TF-IDF Vectorized transformation



```
Accuracy score (training): 1.000
Accuracy score (test): 0.844
Accuracy score (validation): 0.850
```



```
Count Vectorized: training set 0.99375 , test set 0.86875 , validation set 0.865625
TF-IDF: training set 1.0 , test set 0.84375 , validation set 0.85
```

Figure 13: Comparison

3.2 Error

Mean Square Error: of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors—that is, the average squared difference between the estimated values and what is estimated. MSE is a risk function, corresponding to the expected value of the squared error loss. The fact that MSE is almost always strictly positive (and not zero) is because of randomness or because the estimator does not account for information that could produce a more accurate estimate.

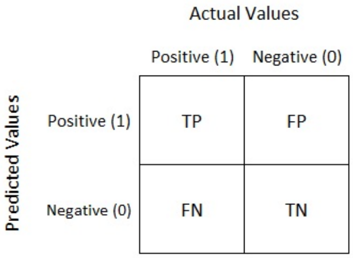
- The value of the GBC with count vectorized transformation: 0.1313
- The value of the GBC with TF-IDF transformation: 0.1562

3.3 Confusion Matrix and Classification Report

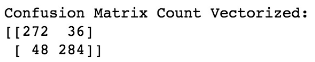
In the confusion matrix:

- True positive: declare a result as true when the reality demonstrate that it is true
- False positive (I type error): declare a result like true despite the reality demonstrate that it is false

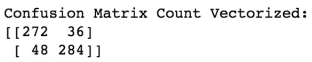
- False negative (II type error): declare a result like false despite the reality demonstrate that it is true
- True negative: declare a result as false when the reality demonstrate that it is false



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN



```
Confusion Matrix Count Vectorized:
[[272  36]
 [ 48 284]]
```



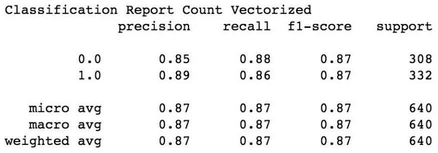
```
Confusion Matrix Count Vectorized:
[[272  36]
 [ 48 284]]
```

Confusion matrix of the GBC Count Vectorized transformation

Confusion matrix of the GBC TF-IDF Vectorized transformation

In the classification report:

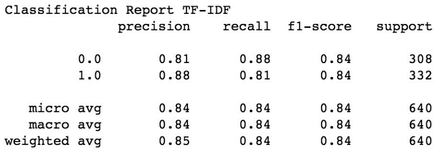
F1-Score $= 2 * \frac{precision*recall}{precision+recall}$
Precision $= \frac{TP}{TP+FP}$
Recall $= \frac{TP}{TP+FN}$



```
Classification Report Count Vectorized
              precision    recall  f1-score   support

         0.0       0.85      0.88      0.87       308
         1.0       0.89      0.86      0.87       332

   micro avg       0.87      0.87      0.87       640
   macro avg       0.87      0.87      0.87       640
weighted avg       0.87      0.87      0.87       640
```



```
Classification Report TF-IDF
              precision    recall  f1-score   support

         0.0       0.81      0.88      0.84       308
         1.0       0.88      0.81      0.84       332

   micro avg       0.84      0.84      0.84       640
   macro avg       0.84      0.84      0.84       640
weighted avg       0.85      0.84      0.84       640
```

Classification report of the GBC Vectorized transformation

Classification report of the GBC Count TF-IDF Vectorized transformation

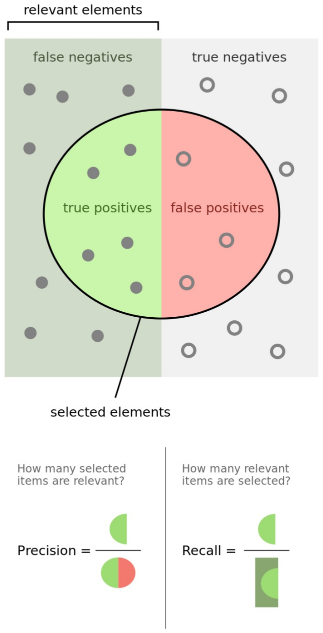


Figure 14: Precision-Recall

3.4 Roc Curve

ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The ROC curve is created by plotting the true positive rate and the false positive rate with various threshold. The true-positive rate is known like recall. The false-positive rate is also known as the probability of false alarm and can be calculated as (1 specificity). The ROC curve is thus the sensitivity as a function of fall-out.

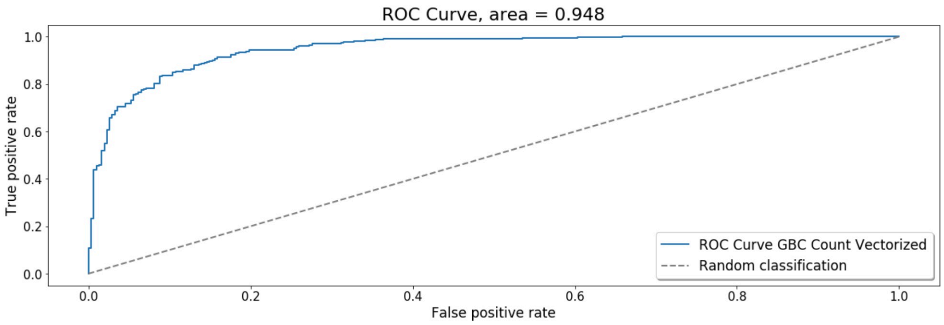


Figure 15: ROC Curve GBC Count Vectorized

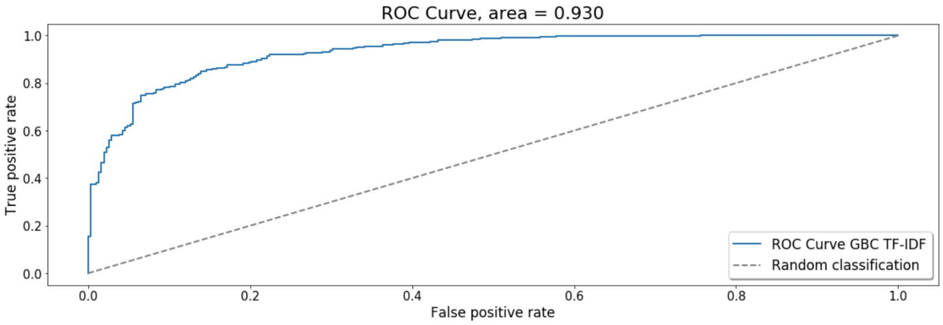


Figure 16: ROC Curve GBC TF-IDF Vectorized

4 Conclusion

In this project an automatic learning method is tested with the aim of detecting false reviews. The dataset where is extracted the data contains 800 true reviews and 800 fake reviews. In the first part of the elaboration of the dataset basic features is explained for a better comprehension. In the second part other features is explained, these are some of the most famous readability indexes, like Flesch Kincaid, Coleman Liau and Automaded readability index. The data processing is implemented with the utilization of Natural Languages Processing (NLP) used like the first step of the machine learning script. The hot part of the feature extraction is with the Word Embedding where two different ways for the transformation of the text of each review in a vector: count vectorizer and TF-IDF vectorizer. Next the transformation allows to the building of the machine learning algorithm and the model choose is been the Gradient Boosting Classifier (GBC). The high possibility in the choosing of the parameters that form the GBC made it necessary to use the GridSearchCV, and the return best estimator is been the model and the parameters choose. In the last parts are shown error, confusion matrix, classification report and the roc curve. Every data and processes are implemented in the Jupyter Notebook file.

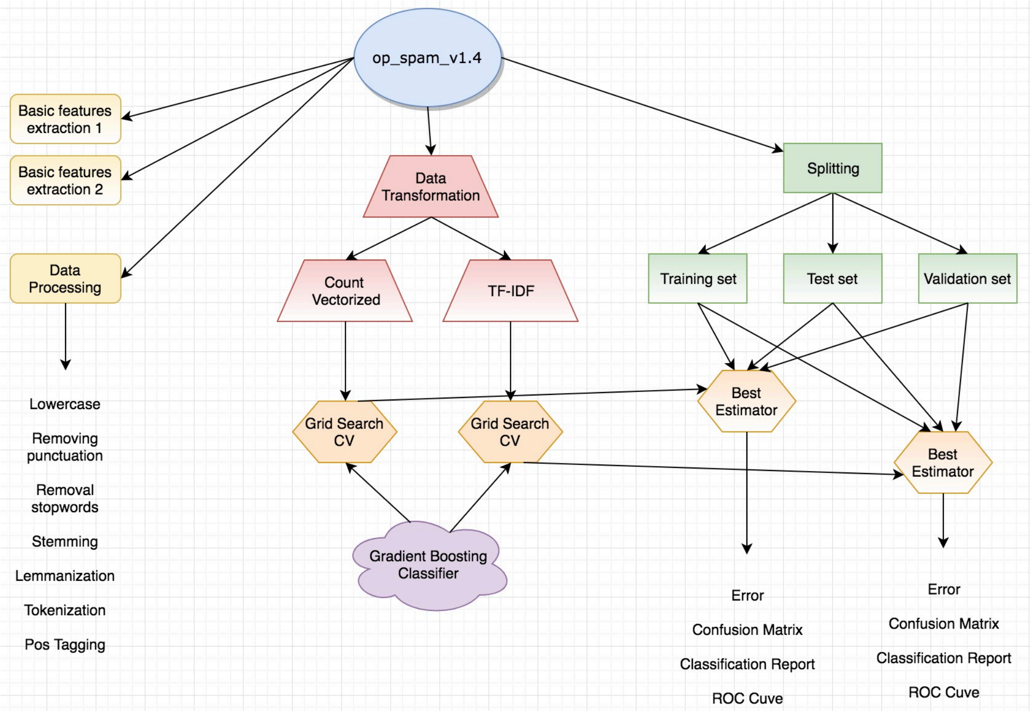
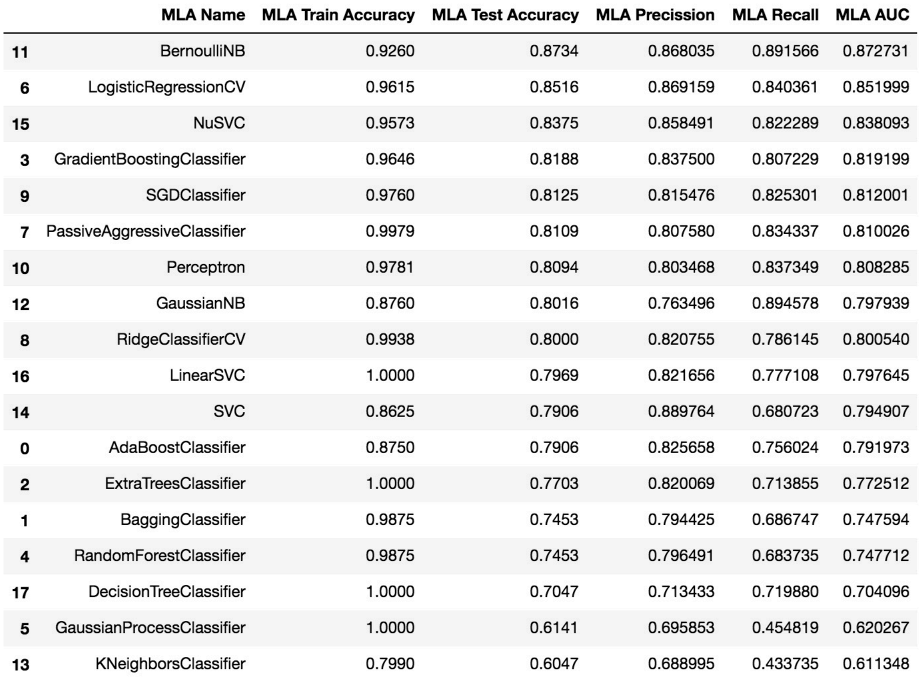


Figure 17: Summary

5 Extra

Other models of machine learning have been tested with the same data processing and the transformation TF-IDF, and these are the results:



	MLA Name	MLA Train Accuracy	MLA Test Accuracy	MLA Precission	MLA Recall	MLA AUC
11	BernoulliNB	0.9260	0.8734	0.868035	0.891566	0.872731
6	LogisticRegressionCV	0.9615	0.8516	0.869159	0.840361	0.851999
15	NuSVC	0.9573	0.8375	0.858491	0.822289	0.838093
3	GradientBoostingClassifier	0.9646	0.8188	0.837500	0.807229	0.819199
9	SGDClassifier	0.9760	0.8125	0.815476	0.825301	0.812001
7	PassiveAggressiveClassifier	0.9979	0.8109	0.807580	0.834337	0.810026
10	Perceptron	0.9781	0.8094	0.803468	0.837349	0.808285
12	GaussianNB	0.8760	0.8016	0.763496	0.894578	0.797939
8	RidgeClassifierCV	0.9938	0.8000	0.820755	0.786145	0.800540
16	LinearSVC	1.0000	0.7969	0.821656	0.777108	0.797645
14	SVC	0.8625	0.7906	0.889764	0.680723	0.794907
0	AdaBoostClassifier	0.8750	0.7906	0.825658	0.756024	0.791973
2	ExtraTreesClassifier	1.0000	0.7703	0.820069	0.713855	0.772512
1	BaggingClassifier	0.9875	0.7453	0.794425	0.686747	0.747594
4	RandomForestClassifier	0.9875	0.7453	0.796491	0.683735	0.747712
17	DecisionTreeClassifier	1.0000	0.7047	0.713433	0.719880	0.704096
5	GaussianProcessClassifier	1.0000	0.6141	0.695853	0.454819	0.620267
13	KNeighborsClassifier	0.7990	0.6047	0.688995	0.433735	0.611348

Figure 18: Summary table of tested models

The elaboration of the data includes the same parameters specified in the section TF-IDF Vectorizer (2.5.2). The tested models are not parameterized with a GridSearchCV, but they are setting by default implementation by Sklearn. The tested models fall into the categories:

- Ensemble Methods
- Gaussian Processes
- Linear Model
- Navies Bayes
- Nearest Neighbor
- Support Vector Machine
- Trees

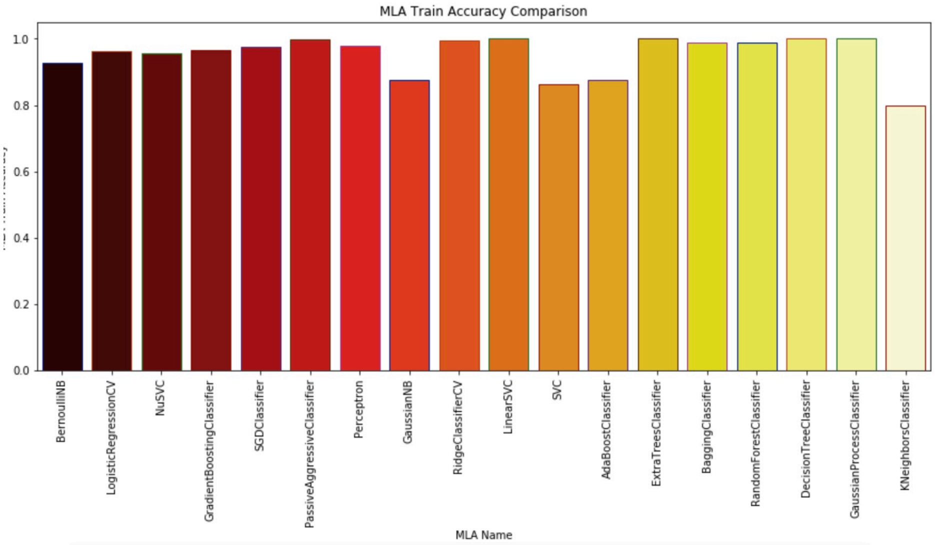


Figure 19: Train accuracy

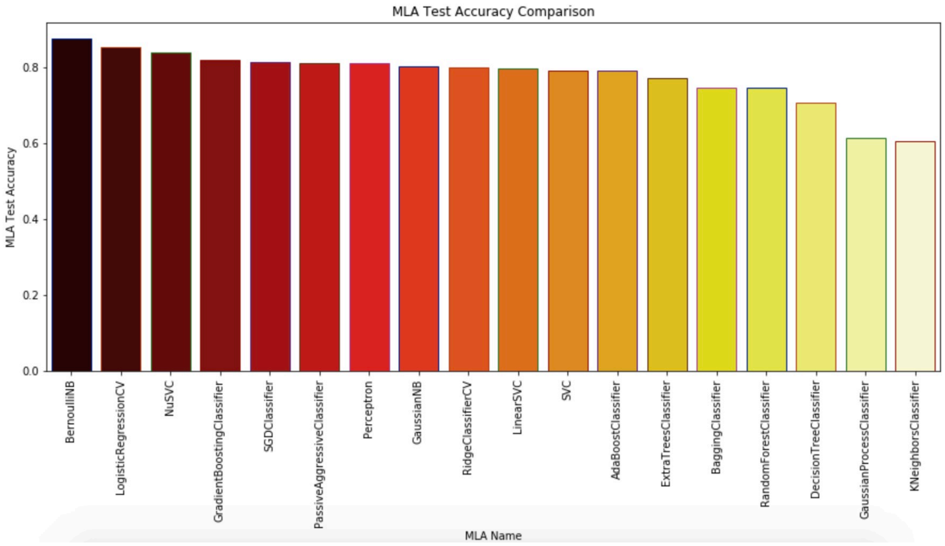


Figure 20: Test accuracy

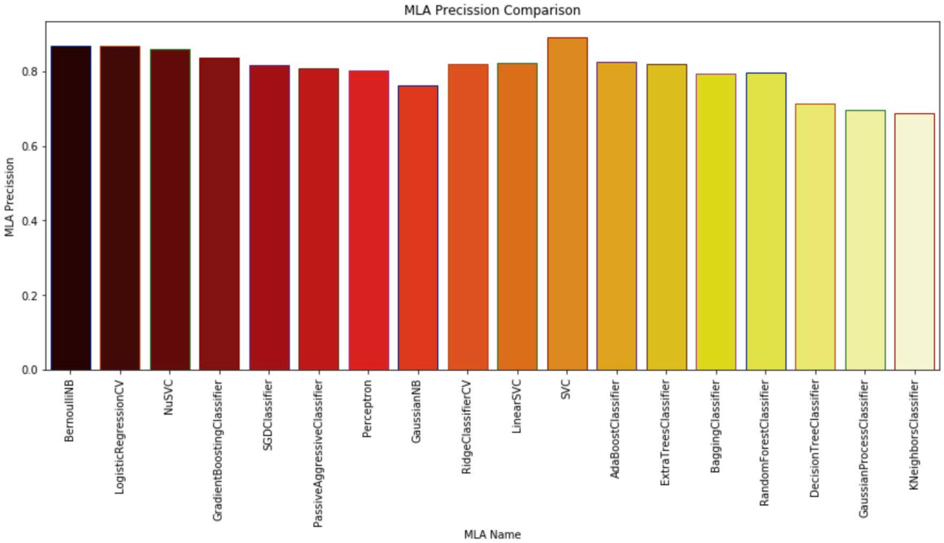


Figure 21: Precision

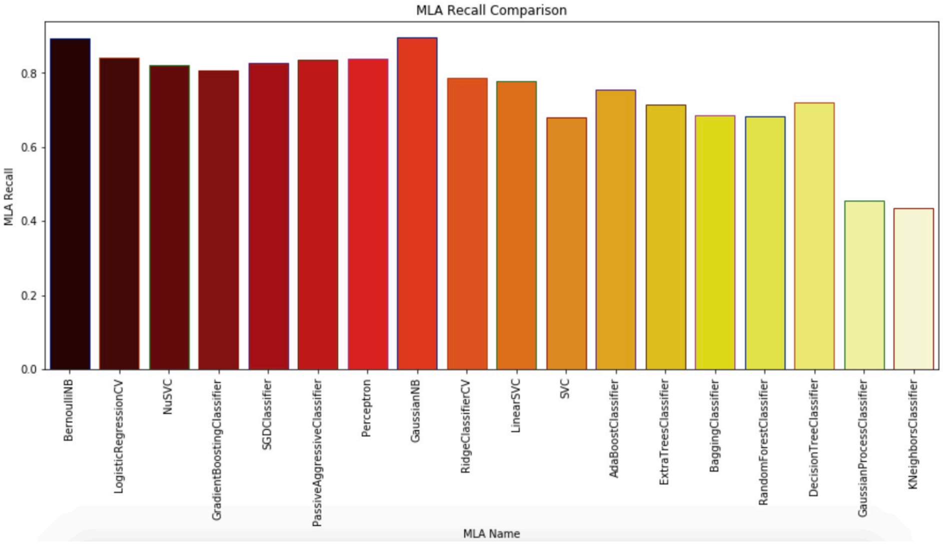


Figure 22: Recall

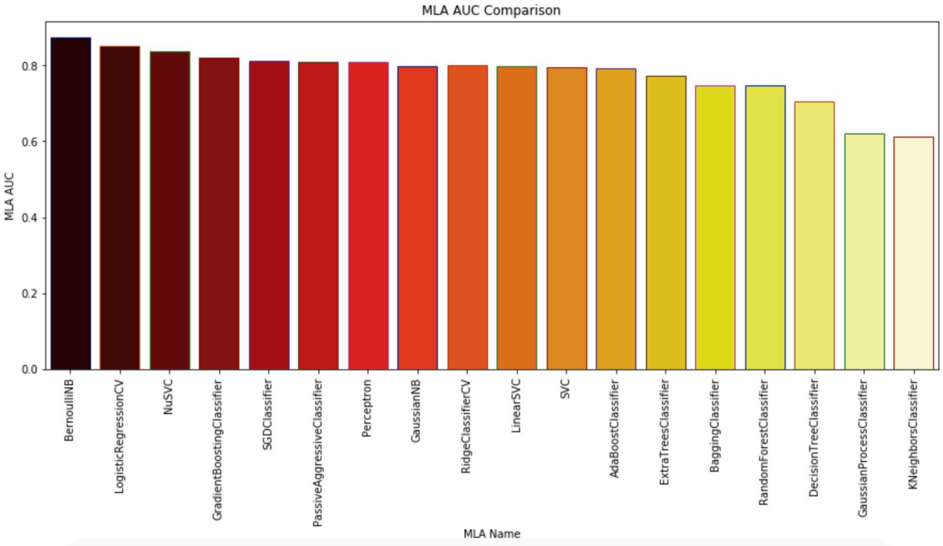


Figure 23: Area Under Curve

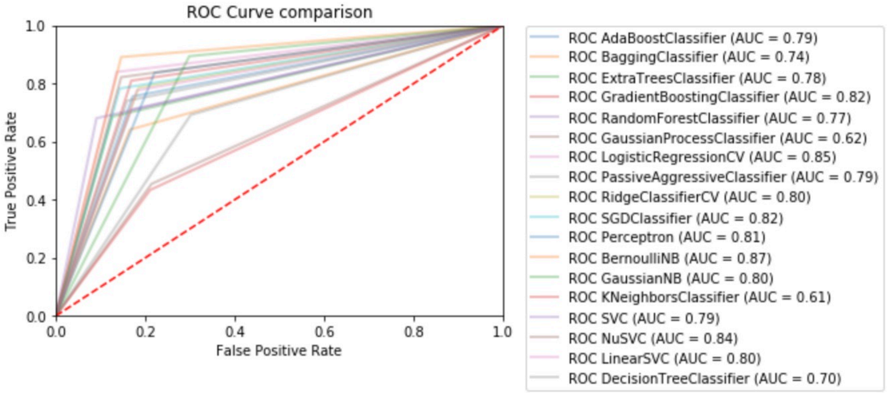


Figure 24: ROC Curve

Bibliography

[1] Chisito. Web reputation sentiment analysis, 2018.

[2] Posting positive reviews, 2011.

[3] SponsoredReviews.com. Sponsored reviews, 2011.

[4] Pay per post, 2019.

[5] Domain Market. Product review writer, 2019.

[6] Aaron Swartz. The network transformation, 2007.

[7] Jure Leskovec Srijan Kumar, Justin Cheng. Malicious behavior on the web: Characterization and detection, 2017.

[8] Robert Sprague and Mary Ellen Wells. Regulating online buzz marketing: Untangling a web of deceit. *American Business Law Journal*, 47(3):415–454, 2010.

[9] Friestad M. Wright P. Boush, D. M. Deception in the marketplace: The psychology of deceptive persuasion and consumer self-protection. *New York, NY, US: Routledge/Taylor Francis Group.*, 2009.

[10] Ahmed Abbasi, Zhu Zhang, David Zimbra, Hsinchun Chen, and Jay F. Nunamaker. Detecting fake websites: The contribution of statistical learning theory. *MIS Quarterly*, 34:435–461, 2010.

[11] Erin E. Buckels, Paul D. Trapnell, and Delroy L. Paulhus. Trolls just want to have fun. *Personality and Individual Differences*, 67:97 – 102, 2014. The Dark Triad of Personality.

[12] Pnina Fichman and Noriko Hara. Beyond vandalism: Wikipedia trolls. *J. Information Science*, 36:357–370, 05 2010.

[13] Jeff Jarvis. America's absurd stab at systematising sock puppetry, 2011.

[14] Theodoros Lappas. Fake reviews: The malicious perspective. In Gosse Bouma, Ashwin Ittoo, Elisabeth Métais, and Hans Wortmann, editors, *Natural Language Processing and Information Systems*, pages 23–34, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[15] Walsh G. Gremler D. D. Hennig-Thurau T., Gwinner K. P. Electronic word-of-mouth via consumer opinion platforms: What motivates consumers to articulate themselves on the internet? *Journal of Interactive Marketing*, 18:38–52, 2004.

[16] Murray K. B. A test of services marketing theory: Consumer information acquisition activities. *Journal of Marketing*, 55:10–25, 1991.

[17] Do-Hyung Park, Jumin Lee, and Ingoo Han. The effect of on-line consumer reviews on consumer purchasing intention: The moderating role of involvement. *International Journal of Electronic Commerce*, 11(4):125–148, 2007.

[18] David Streitfeld. The best book reviews money can buy, 2012.

[19] Claire Cain Miller. Company seattles case of review it faked, 2009.

[20] David Streitfeld. For 2 dollar a star, an online retailer gets 5-star product reviews, 2012.

[21] Amy Harmon. Amazon glitch unmasks war of reviewers, 2004.

[22] Mary Pilon. A fake amazon reviewer confesses, 2009.

[23] Huayi Li, Zhiyuan Chen, Bing Liu, Xiaokai Wei, and Jidong Shao. Spotting fake reviews via collective positive-unlabeled learning. In *Proceedings of the 2014 IEEE International Conference on Data Mining*, ICDM '14, pages 899–904, Washington, DC, USA, 2014. IEEE Computer Society.

[24] Arjun Mukherjee, Bing Liu, and Natalie Glance. Spotting fake reviewer groups in consumer reviews. *WWW'12 - Proceedings of the 21st Annual Conference on World Wide Web*, 05 2012.

[25] Sungwoo Choi, Anna S. Mattila, Hubert B. Van Hoof, and Donna Quadri-Felitti. The role of power and incentives in inducing fake reviews in the tourism industry. *Journal of Travel Research*, 56(8):975–987, 2017.

[26] Simon Parkin. The never-ending war on fake reviews, 2018.

[27] Chevalier J. Mayzlin D., Dover Y. Promotional reviews: An empirical investigation of online review manipulation. *American Economic Review*, 104 (8):2421–55, 2014.

[28] Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 309–319, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[29] Myle Ott, Claire Cardie, and Jeffrey T. Hancock. Negative deceptive opinion spam. In *HLT-NAACL*, 2013.

[30] Vanessa Wei Feng and Graeme Hirst. Detecting deceptive opinions with profile compatibility. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 338–346. Asian Federation of Natural Language Processing, 2013.

[31] Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie S. Glance. Fake review detection : Classification and analysis of real and pseudo reviews. 2013.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[33] NLTK. Natural language toolkit, 2018.

[34] NumPy developers. Numpy, 2018.

[35] Pandas Team. Python data analysis library, 2018.

[36] Eric Firing Michael Droettboom John Hunter, Darren Dale and the Matplotlib development team. Matplotlib, 2012.

[37] NLTK community. Nltk, 2018.

[38] Jiwei Li, Myle Ott, Claire Cardie, and Eduard Hovy. Towards a general rule for identifying deceptive opinion spam. volume 1, pages 1566–1576, 06 2014.

[39] John Talburt. The flesch index: An easily programmable readability analysis algorithm. pages 114–122, 1985.

[40] Liau T. L. Coleman, M. A computer readability formula designed for machine scoring. pages 283–284, 1975.

[41] E.A Senter, R.J.; Smith. Automated readability index. 1967.

[42] Cambridge University Press. Stemming and lemmatization, 2008.

[43] Benjamin Bengfort Tony Ojeda, Rebecca Bilbro. Ctext vectorization and transformation pipelines, 2019.

[44] Aarshay Jain. Complete guide to parameter tuning in gradient boosting (gbm) in python, 2016.