Detecting Fake News

Alma del Carmen Ayaquica Ontiveros

Tecnologico de Monterrey. Vía Atlixcáyotl No. 2301, Reserva Territorial Atlixcáyotl, Puebla 72453, Mexico.
a01324264@itesm.mx

Abstract. During the last few years, social media has acquire an immense popularity for different consumption due to its fast dissemination, easy access and low cost. Its consumption goes from social context, entertainment up to information acquired by news. However, social media has a double-edge sword for news consumption, their quality is much lower than traditional news and it's easy access has enable the spread of fake news while being produced for a variety of purposes. "Fake news" are information that are inaccurate, may not include verifiable facts or sources and is mistakenly or intentionally written to mislead users. Detecting fake news ensures users to receive authentic information and maintain a trustworthy news ecosystem. Therefore, in this work I will be presenting the different methods that I used to detecting fake news with the help of WEKA and the score I got in KAGGLE with them.

Keywords: Fake news detection · Social Media · WEKA · KAGGLE.

1 Introduction

Usually, the 20% of all the published news are fake and most of the time users believe in them. Fake news are written and published usually with the intent to mislead in order to damage an agency, entity, or person, and/or gain financially or politically, often using sensationalist, dishonest, or outright fabricated headlines to increase readership.

Social media and news outlets publish fake news to increase readership. In general, the goal is profiting through click-baits which lure users and entice curiosity with flashy headlines or designs to click links to increase advertisements revenues. The growth of social media and online forums has spurred the spread of fake news causing it to easily blend with truthful information.

The problem is real and hard to solve because the fake news are getting better and created every day. Is not simple to detect when the information is true or not all the time, so we need to find better systems to help us understand the patterns of fake news besides helping to improve our social media, communication and prevent confusion in the world.

Fake news have significant differences compared with traditional information like: *(1)society' impact:* the impact of fake news in social media can be tremendous due to the massive user numbers globally, which is further boosted by the

extensive information sharing and propagation among the users; *(2)users initiative:* users in social media seek to receive and share news information with no sense about its correctness; *(3)identification difficulty:* identifying fake news with erroneous information is really challenging since it requires evidence-collector and fact-checking due to the lack of other comparative news articles available.

In this short article, I'll explain several ways to detect fake news using diverse data-science and machine-learning environments like WEKA and KAGGLE. As well as, describe the different algorithms used to recollect the 3 significant differences and how I used them in detecting fake news.

2 Related Works

As aforementioned, the detection of fake news ensures users to receive authentic information and maintain a trustworthy news ecosystem, but being able to detect them in the early stage is a really challenging work. In consequence, many researchers have been interested in helping and evaluating forms to solve this problem and find different ways to achieve it.

One article viewed during this semester was how the Role of Social Context help for Fake News Detection. During this paper by Kai Shu, Suhang Wang and Huan Liu, they propose a TriFN framework, which exploits both user-news interactions and publishers-news relations for learning news feature representations to predict fake news. Which is a principled way to model tri-relationship among publishers, news pieces, and users simultaneously. This proposal uses the publisher-news relationships and user-news interactions to provide new and different perspectives of social-context and give complementary information to improve the results in detection.

The TriFN framework has 5 major components, *(1)News contents embedding; (2)User embedding;(3)User-news interactions embedding; (4)Publisher-news relation embedding and (5)Semi-supervised classification*. I will explain each components during this section.

2.1 News contents embedding

During this component, the first step is doing a mapping of news from a bag-of-word features and illustrate the extraction of user latent features from user social relations. With the information obtained and using a non-negative matrix factorization (NMF) algorithm, project the news-word matrix to a latent semantic factor space with low dimension and model as an inner product. This method will find two non-negative matrices indicating low dimension representations of news pieces and words avoiding over-fitting.

2.2 User embedding

The priority is to obtain a standard representation using non-negative matrix factorization to learn users' latent representation. This is acquired using the

following equation.

$$min \left\| Y \bigodot \left(A - UTU^T \right) \right\| \frac{2}{F} + \lambda \left(\|U\| \frac{2}{F} + \|T\| \frac{2}{F} \right) \tag{1}$$

Where U is the user latent matrix, T is the user-user correlation matrix, Y controls the contribution and $\bigodot$ denotes the Hadamard product operation, while avoiding over-fitting.

2.3 User-news interactions embedding

In this component, there is a need to model the user-news interactions by considering the relationships between user features and the labels of news items. Besides, it measures the user credibility score by the following steps: *(1)Detect and cluster coordinate users based on user similarities; (2)Weight each cluster based on the cluster size.*

By obtaining the user credibility score, there are two types of users, the less credible users and the high credibility users. Less credible users are more likely to coordinate with each other and create big clusters, while more credible will form small clusters. High credibility users are more likely to share true news, so the distance between latent features are minimized, this is obtained by the following:

$$\min_{U, D_L \geq 0} \sum_{i=1}^{m} \sum_{j=1}^{r} W_{ijc_i} (1 - \frac{1 + Y_{LJ}}{2}) \|U_i - D_{L_J}\| \frac{2}{2} \tag{2}$$

The low-credibility users are more likely to share fake, so the distance between latent features are minimize by the following:

$$\min_{U, D_L \geq 0} \sum_{i=1}^{m} \sum_{j=1}^{r} W_{ij} (1 - c_i)(1 - \frac{1 + Y_{LJ}}{2}) \|U_i - D_{L_J}\| \frac{2}{2} \tag{3}$$

This is to ensure to include fake news pieces and adjust the loss function.

2.4 Publisher-news relation embedding

Fake news is often written to convey opinions or claims that support the partisan bias of news publishers. The partisan bias labels are checked with a principled methodology that ensures the reliability an objectivity of the partisan annotations.To further ensure the accuracy of the labels, it's only consider those news publishers with the annotations and rewrite the labels as binary.

The idea is to utilize publisher partisan labels vectors and publisher-news matrix to optimize the news feature representation learning searching for optimization with the following objective function:

$$\min_{D \geq 0, q} \left\| e \bigodot \left(\bar{B} Dq - o \right) \right\| \frac{2}{2} + \lambda \|q\| \frac{2}{2} \tag{4}$$

2.5 Semi-supervised classification

With all the previous components, TriFN solves the following optimization problem

$$\begin{aligned}\min_{D,U,V,T\geq 0,p,q} & \left\| X - DV^T \right\|\frac{2}{F} + \alpha \left\| Y \bigodot (A - UTU^T) \right\| \frac{2}{F} \\ & + \beta tr(H^T LH) + \gamma \left\| e \bigodot \left(\bar{B}Dq - o\right) \right\| \frac{2}{2} \\ & + \eta \left\| D_{LP} - Y_L \right\| \frac{2}{2} + \lambda R\end{aligned} \tag{5}$$

The first term models the news latent features from news contents; the second extracts user latent features from their social relationships; the third incorporates the user-news interactions; the forth models publisher-new relationship, while the fifth adds a semi-supervised fake news classifier. This way, the framework provides a principled way to model tri-relationship for fake news prediction.

Early fake news detection prevents the future propagation on social media and aims to give early alert considering limited social context within a specific range of the original news posted. TriFN has demonstrated the importance of embedding user-news interactions to capture effective feature representations even on an early stage achieve good performance.

3 Our Proposal

To understand how many of this news are around us, WEKA will help us in testing our training set with different algorithms. WEKA in an open source tried-and-tested machine learning software used for teaching, research and industrial applications that contains a built-in-tools for standard machine learning tasks. While KAGGLE, in its web-based data-science environment, will help in publishing the resulting data set, along with exploring and building models.

Using the datasets obtained in KAGGLE, I added, for all the fake news, 16 new features - five regarding to sentiment analysis, six regarding to emotion extraction and five regarding intent analysis - by using the APIs of public frameworks *Meaning Clound and Paralleldots*.

Meaning Cloud Helps users to embed text analytics and semantic processing in any system. Is the easiest, most powerful, and most affordable way to extract the meaning of all kinds of unstructured content.

Excel Add-In This add-in can automatically extract the meaning from different texts and in a multilingual content as a csv file.

Parrallel Dots ParallelDots AI APIs are a comprehensive set of document classification and NLP APIs. Their NLP models are trained and provide accuracy on most common NLP use-cases such as sentiment analysis and emotion detection.

Implementation I came to an understanding that the API works for different types of programming languages, some being shell, ruby, python, php, c#, etc. Following the examples from the documentation, I used the following code to obtain emotion data.

```
# Import
import paralleldots
import csv
# Setting your API key
paralleldots.set_api_key("API-KEY")
# Viewing your API key
paralleldots.get_api_key()
with open('TrainingEmotion.csv', mode='r', encoding="utf8") as csv_file:
    csv_reader = csv.DictReader(csv_file)
    for row in csv_reader:
        print('The emotion analysis results for '+'ID: '+f'{row["ID"]}')
        response=paralleldots.emotion(lines)
        print(response)
```

For the intent analysis I used the following code

```
# Import
import paralleldots
import csv
# Setting your API key
paralleldots.set_api_key("API-KEY")
# Viewing your API key
paralleldots.get_api_key()
with open('TrainingEmotion.csv', mode='r', encoding="utf8") as csv_file:
    csv_reader = csv.DictReader(csv_file)
    for row in csv_reader:
        print('The intent analysis results for '+'ID: '+f'{row["ID"]}')
        response=paralleldots.intent(lines)
        print(response)
```

Using the former datasets, with the new features obtained by Parrallel Dots and Meaning Cloud, I've got a 16-features data-set with the following structure:

- *Polarity:* determine if it expresses a positive/negative/neutral sentiment
- *Agreement:* between the sentiments detected in the text it refers to.
- *Subjectivity:* Marks the subjectivity of the text.
- *Confidence:* Represents the confidence associated with the sentiment analysis performed on the text.
- *Irony:* indicates the irony of the text
- *Happy:* indicates the probability detected in the text for the happy sentiment
- *Angry:* indicates the probability detected in the text for the Angry sentiment
- *Bored:* indicates the probability detected in the text for the Bored sentiment
- *Fear:* indicates the probability detected in the text for the Fear sentiment

- *Sad:* indicates the probability detected in the text for the Sad sentiment
- *Excited:* indicates the probability detected in the text for the Excited sentiment
- *News:* The confidence score of the news intent in the text
- *Query:* The confidence score of the query intent in the text
- *Spam:* The confidence score of the spam intent in the text
- *Marketing:* The confidence score of the marketing intent in the text
- *Feedback:* The confidence score of the feedback intent in the text

Using the features obtained and with the help of WEKA I got the best result using **'logistic regression'**, known in WEKA as Logistic. Logistic regression is a powerful classification that predict probabilities directly through something called the "logit transform" - the inverse of the sigmoidal.It's a popular and powerful machine learning method that uses the logit transform to predict probabilities directly as it works internally with them, like Naïve Bayes does. Is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.The score I got in KAGGLE using this algorithm was ***0.69444*** of AUC.

4 Experimental Setup

As aforementioned, WEKA in an open source tried-and-tested machine learning software used for teaching, research and industrial applications that contains a built-in-tools for standard machine learning tasks. Many classification methods produce probabilities rather than black-or-white classifications.

But there are other methods too, some of the used ones for this project where: K-nn variants; Pattern-based classifiers, for example Random Forest and J48; Resampling Methods for imbalance problems as SpreadSubSample or Resample; Bagging and Adaboost methods; Bayes Network methods as BayesNet with Simple-Estimator or with BMAEstimator; and Feature Selection Methods as WrapperSubsetEva with GreedyStepWise, CfsSubsetEval with BestFirst; Function Classifiers as Logistic or Multilayer Perceptron .

5 Experimental Results and Discussion

5.1 K-NN

Euclidean

$k = 3$ I decided to implement Euclidean distance function with a k=3. This distance can be calculated from the Cartesian coordinates of the points using the Pythagorean theorem, and is occasionally called the Pythagorean distance.

With this information obtained I did the conversion into a csv file so that I could submit it to Kaggle.. With it, I upload the file to KAGGLE and obtained a score of *0.53888*.

Manhattan

$k = 9$ I decided to implement Manhattan distance function with a k=9. It's the distance between two points measured along axes at right angles.

With this information obtained I did the conversion into a csv file so that I could submit it to Kaggle.. With it, I upload the file to KAGGLE and obtained a score of *0.56944*.

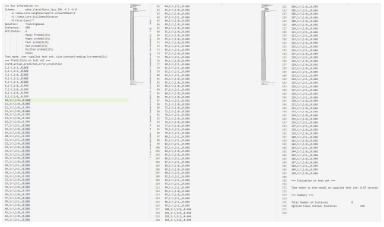


Fig. 1: WEKA Results for Euclidean distance and K=3

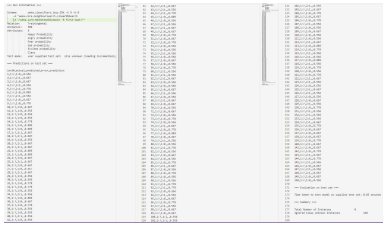
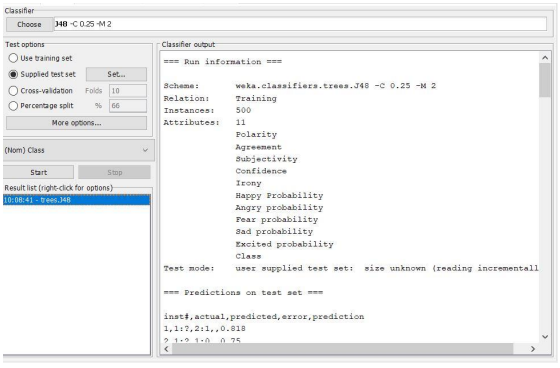


Fig. 2: WEKA Results for Manhattan distance and K=9

5.2 Pattern-based Classifiers

J48 I decided to try the J48 classifier in WEKA and try to find a better score than with K-NN. J48, also known as C4.5, is an algorithm used to generate a decision tree developed. The decision trees generated by J48 can be used for classification, and for this reason, is often referred to as a statistical classifier.

Fig. 3: Selecting J48 Classifier in Weka

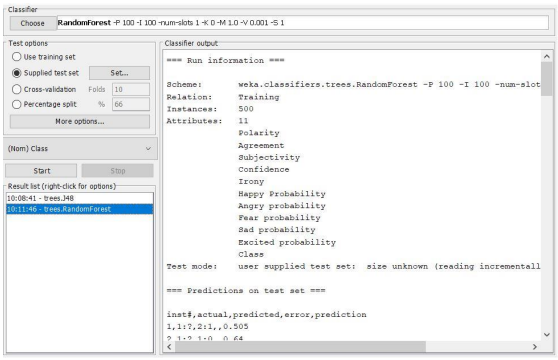


With this information obtained I did the conversion into a csv file so that I could submit it to Kaggle. With it, I upload the file to KAGGLE and obtained

a score of *0.5611*. Doing a comparison with the result in k=9 of Manhattan I notice that there was not an improvement in the score, but the values were close.

Random Forest It is an ensemble tree-based learning algorithm, the classifier is a set of decision trees from randomly selected subset of training set. It aggregates the votes from different decision trees to decide the final class of the test object.

Fig. 4: Selecting J48 Classifier in Weka



With this information obtained in Weka, I did the conversion into a csv file so that I could submit it to Kaggle. With it, I upload the file to KAGGLE and obtained a score of *0.58611*. Doing a comparison with the result in k=9 I notice that there was an improvement in the score.

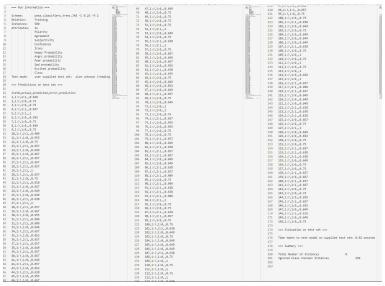


Fig. 5: WEKA Results for J48 Classifier

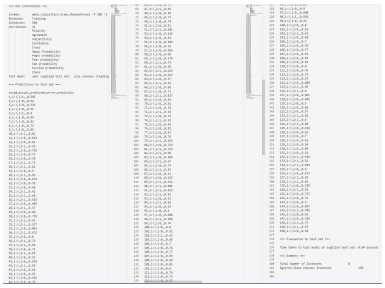


Fig. 6: WEKA Results for Random Forest Classifier

5.3 Resampling Methods for imbalance problems

Resample With CostSensitiveClassifier And RandomForest Resample produces a random subsample of a dataset using either sampling with replacement or without replacement. CostSensitiveClassifier re-weight training instances according to the total cost assigned to each class or predicts the classes with minimum expected misclassification cost.

Fig. 7: Selecting the Resample preprocess

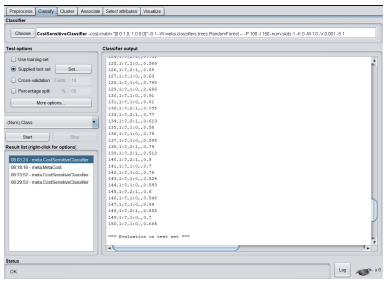


Fig. 8: Selecting the CostSensitiveClassifier and RandomForest

With this information obtained in Weka, I did the conversion into a csv file so that I could submit it to Kaggle. With it, I upload the file to KAGGLE and obtained a score of *0.59722*. Doing a comparison with the result in Random Forest I notice that there was an improvement in the score.

SpreadSubSample With MetaCost And RandomForest SpreadSubSample produces a random subsample of a dataset. MetaCost produces a single cost-sensitive classifier of the base learner, giving the benefits of fast classification and interpretative output

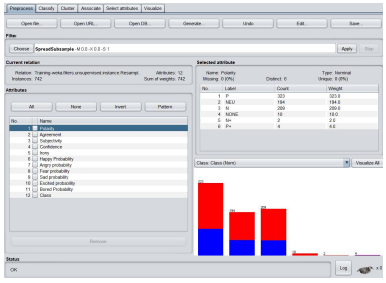


Fig. 9: Selecting the SpreadSubSample preprocess

Fig. 10: Selecting the MetaCost and RandomForest

With this information obtained in Weka, I did the conversion into a csv file so that I could submit it to Kaggle. With it, I upload the file to KAGGLE and obtained a score of *0.60000*. Doing a comparison with the result in Resample With CostSensitiveClassifier And RandomFores I notice that there was an improvement in the score.

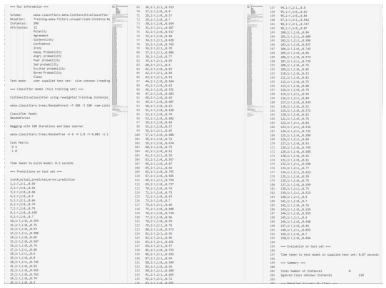


Fig. 11: WEKA results for Resample With CostSensitive Classifier And RandomForest

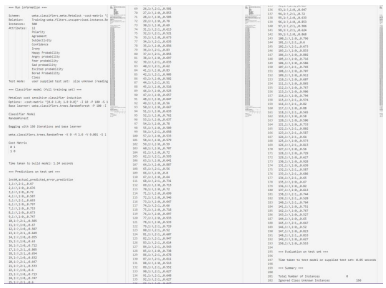
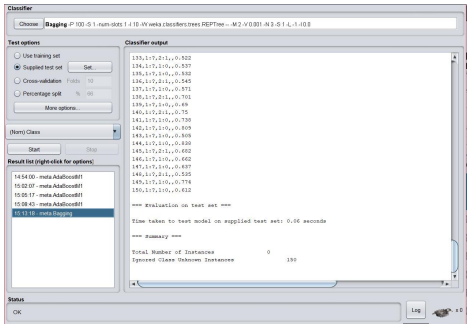


Fig. 12: WEKA results for SpreadSubSample With MetaCost And Random Forest

5.4 Bagging and Adaboost Methods

Bagging Bagging is a simple and very powerful ensemble method, where the application of the Bootstrap procedure is to a high-variance machine learning algorithm, typically decision trees. The only parameters when bagging decision trees is the number of samples and hence the number of trees to include. This can be chosen by increasing the number of trees on run after run until the accuracy begins to stop showing improvement.

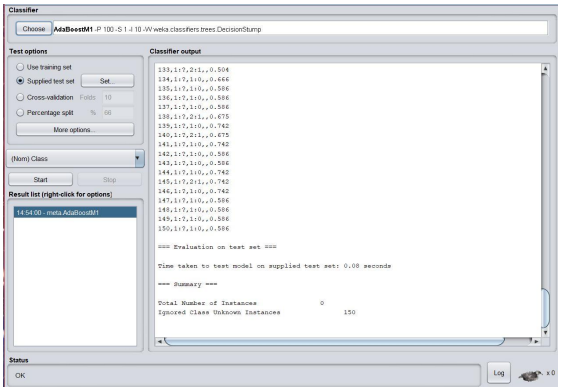
Fig. 13: Selecting Bagging Classifier



With this information obtained in Weka, I did the conversion into a csv file so that I could submit it to Kaggle. With it, I upload the file to KAGGLE and obtained a score of *0.56944*. Doing a comparison with the result in SpreadSubSample With MetaCost And RandomForest I did not notice an improvement in the score.

Boosting Boosting refers to a group of algorithms that utilize weighted averages to make weak learners into stronger learners. Unlike bagging that had each model run independently and then aggregate the outputs at the end without preference to any model. Boosting is all about "teamwork". Each model that runs, dictates what features the next model will focus on.

Fig. 14: Selecting Boosting Classifier



With this information obtained in Weka, I did the conversion into a csv file so that I could submit it to Kaggle. With it, I upload the file to KAGGLE and obtained a score of *0.58888*. Doing a comparison with the result in SpreadSubSample With MetaCost And RandomForest I did not notice an improvement in the score.

5.5 Bayes Network Methods

BayesNet with Simple-Estimator BayesNet, also known as Bayes Network, learn about the features using various search algorithms and quality measures, uses Bayesian inference for probability computations that aims to model conditional dependence. Through these relationships, one can efficiently conduct inference on the random variables in the graph through the use of factors. While SimpleEstimator is used for estimating the conditional probability tables of a Bayes network once the structure has been learned. Estimates probabilities directly from data.

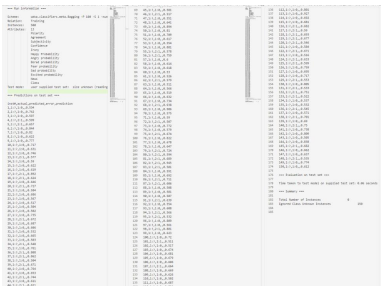


Fig. 15: WEKA results for Bagging Classifier

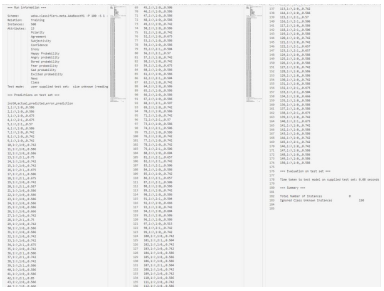


Fig. 16: WEKA results for Boosting Classifier

Fig. 17: Selecting BayesNet Classifier

Fig. 18: Selecting Simple Estimator

With the information obtained in Weka, I did the conversion into a csv file so that I could submit it to Kaggle. With it, I upload the file to KAGGLE and obtained a score of *0.58888*. Doing a comparison with the result in Boosting I did not notice an improvement nor a decrease in the score.

BayesNet with BMAEstimator As already explained, BayesNet does probability computations that aims to model conditional dependence, and through these relationships, one can efficiently conduct inference on the random variables. BMAEstimator estimates conditional probability tables of a Bayes network using Bayes Model Averaging (BMA), the standard approach to model uncertainty within the Bayesian paradigm, where it is natural to reflect uncertainty through probability.

Fig. 19: Selecting BayesNet Classifier

Fig. 20: Selecting BMAEstimator

With the information obtained in Weka, I did the conversion into a csv file so that I could submit it to Kaggle. With it, I upload the file to KAGGLE and obtained a score of *0.58888*. Doing a comparison with the result in Boosting I did not notice an improvement nor a decrease in the score.

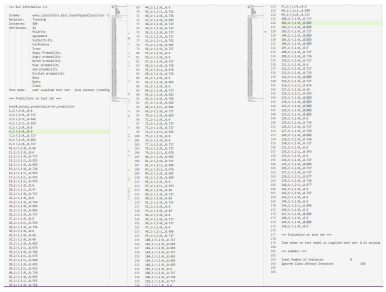


Fig. 21: WEKA results for BayesNet with Simple-Estimator

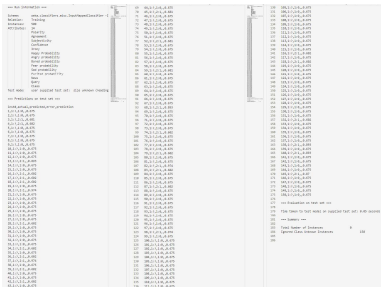


Fig. 22: WEKA results for BayesNet with BMAEstimator

5.6 Feature Selection Methods

WrapperSubsetEva with GreedyStepWise WrapperSubsetEval evaluates attribute sets by using a learning scheme and uses Cross validation to estimate the accuracy of the learning scheme for a set of attributes.GreedyStepwise performs forward or backward search through the space of attribute subsets, this may start with no/all attributes or from an arbitrary point in the space and Stops when the addition/deletion of any remaining attributes results in a decrease in evaluation.

Fig. 23: Selecting the WrapperSubsetEva attribute Evaluation with RandomForest

Fig. 24: Selecting the GreedyStepWise search method

Fig. 25: Features Selected

```
=== Attribute Selection on all input data ===

Search Method:
        Greedy Stepwise (forwards).
        Start set: no attributes
        Merit of best subset found:    0.612

Attribute Subset Evaluator (supervised, Class (nominal): 15 Class):
        Wrapper Subset Evaluator
        Learning scheme: weka.classifiers.trees.RandomForest
        Scheme options: -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1
        Subset evaluation: classification accuracy
        Number of folds for accuracy estimation: 5

Selected attributes: 4,5 : 2
                     Confidence
                     Irony
```

With the information obtained in Weka, I did the conversion into a csv file so that I could submit it to Kaggle. With it, I upload the file to KAGGLE and obtained a score of *0.566666*. Doing a comparison with the result in BayesNet with BMAEstimator I did not notice an improvement in the score.

CfsSubsetEval with BestFirst CfsSubsetEval evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. BestFirst searches the space of attribute subsets by greedy hillclimbing augmented with a backtracking facility. It may start with the empty set of attributes and search forward, or start with the full set of attributes and search backward, or start at any point and search in both directions.

Fig. 26: Selecting the CfsSubsetEval attribute Evaluation

Fig. 27: Selecting the BestFirst search method

Fig. 28: Features Selected

```
Attribute Subset Evaluator (supervised, Class (nominal): 15 Class):
        CFS Subset Evaluator
        Including locally predictive attributes

Selected attributes: 2,3,4,13 : 4
                     Agreement
                     Subjectivity
                     Confidence
                     Query
```

With the information obtained in Weka, I did the conversion into a csv file so that I could submit it to Kaggle. With it, I upload the file to KAGGLE and obtained a score of *0.60555*. Doing a comparison with the result in SpreadSub-Sample With MetaCost And RandomForest I did notice an improvement in the score

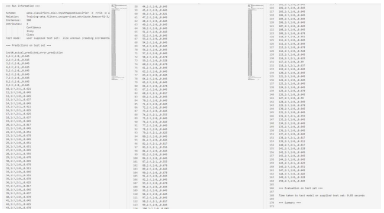


Fig. 29: WEKA results for WrapperSubsetEva with GreedyStepWise

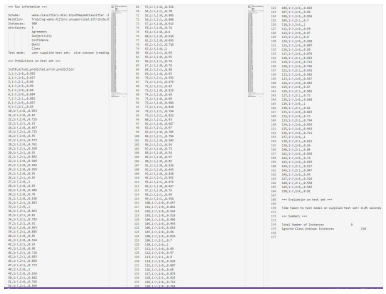


Fig. 30: WEKA results for CfsSubsetEval with BestFirst

5.7 Function Classifiers

Multilayer Perception It's a classifier that uses backpropagation to learn a multi-layer perceptron to classify instances. A quick test showed that a multilayer perceptron with one hidden layer gave better results than other methods on two out of six data sets, but it was 10–2000 times slower than other methods, which is a bit of a disadvantage.

With the information obtained in Weka, I did the conversion into a csv file so that I could submit it to Kaggle. With it, I upload the file to KAGGLE and obtained a score of *0.53333*. Doing a comparison with the result in CfsSubsetEval with BestFirst I did not notice an improvement in the score being the best algorithm that I used.

Logistic Logistic regression is a powerful classification that predict probabilities directly through something called the "logit transform" - the inverse of the sigmoidal. It uses the logit transform to predict probabilities directly as it works internally with them. Is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

With the information obtained in Weka, I did the conversion into a csv file so that I could submit it to Kaggle. With it, I upload the file to KAGGLE and obtained a score of ***0.69444***, being the best score and placing myself in the ***8th position*** on Kaggle.

6 Conclusion

In this paper, I explained all the algorithms and methods used in WEKA for detecting Fake News with the data sets available in KAGGLE, as well as explaining how I obtained the 16 features used for it. During this months, I have investigated different feature extraction and machine learning techniques to achieve a good accuracy and score in KAGGLE, as already explained.

After all this time, and with all the techniques used, I believe that one of the best algorithms for detecting fake news is Logistic regression. This Classifier is used to predict the probability of a categorical dependent variable. The dependent variable is a binary variable that contains data coded as 1 (yes, success, etc) or 0 (no, failure, etc.), in other words, it predicts $f(X) = P(Y = 1)$. This idea is what I was searching for, finding which are True and False, being this classification the ideal for detecting fake news in my research.

References

1. Loyola-González, Octavio. "Detecting Fake News." Kaggle, 3 Aug. 2020, www.kaggle.com/c/detecting-fake-news/overview. Last accessed 18 Nov 2020
2. Shu, Kai., Wang, Suhang., Liu, Huan: "Beyond News Contents:The Role of Social Context for Fake News Detection" (10 Dec 2018)
3. Weka Documentation. Packages Documentation, https://weka.sourceforge.io/doc.dev/ Last accessed 12 Nov 2020
4. MeaningCloud. "Sentiment Analysis." MeaningCloud, 2020, www.meaningcloud.com/developer/sentiment-analysis/doc/2.1/response Last accessed 12 Nov 2020
5. ParrallelDots. "ParallelDots Text API." ParrallelDots AI API's, 2020, www.apis.paralleldots.com/text_docs/index.html Last accessed 12 Nov 2020