University of Windsor

Project Report - 02

Data Science & Cybersecurity

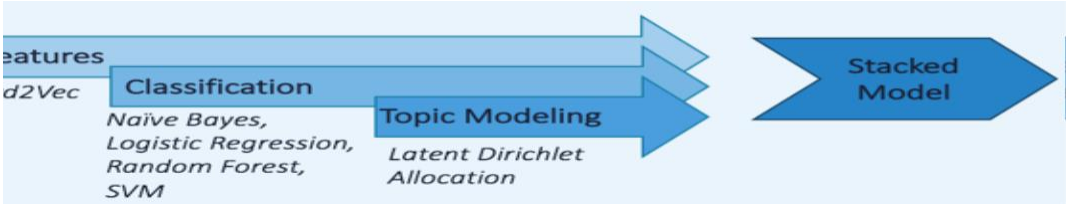
COMP 8920-01 Summer 2019

Project Supervisor: Dr. Sherif Saad

Author: Hemni Sri Rajeswari Karlapalepu

1. Workflow:

Our goal for this project was to find a way to utilize Natural Language Processing (NLP) to identify and classify fake articles. We gathered our data, preprocessed the text, and converted our articles into features for use in supervised models.



2. FEATURE ENGINEERING (Text to Features):

Tokenization	• Segment text into unigrams, using spaCy.load() to return a language object containing components needed to process text usually called NLP. • Segment text into bigrams, breaking into set of two words.
Remove Stop words and cleaning of Data	• List of most common words which are often a noise rather than features such as *('and', 'i', 'are' etc.*). • We removed symbols such as *('$','!')* from the text, then used ENGLISH_STOP_WORDS as a stop list and a custom list of ignore words *('our', 'you'*). • We have also removed punctuations (*single and double quotes, commas, whitespaces etc*.) and dropped duplicate rows from the dataset.
Lemmatization	• Lemmatization converts words in the second or third forms to their first form variants unlike stemming which only removes deriving affixes *('ed', 'ly'*). • spaCy determines the part-of-speech tag by default and assigns the corresponding lemma.
Count Vectorizing	• Override the string tokenization step will preserving the preprocessing and n-gram generation and we have used ENGLISH_STOP_WORDS inside Count Vectorizer function and extracted lemmas from it. • We converted a collection of text documents to a matrix of token counts which produces *sparse matrix*. • The idea is super simple. Create a vector that has as many dimensions as our dataset has unique words. Each unique word has a unique dimension and will be represented by a 1 in that dimension with 0s everywhere else.
TF-IDF Transform	• TF-IDF vectors are related to one-hot encoding. However, instead of just featuring a count, they feature numerical representations where words aren't just there or not there. Instead, words are represented by their term frequency multiplied by their inverse document frequency. • Words that occur a lot but everywhere should be given very little weighting or significance. We can think of this as words like "the" or "and" in the English language. They don't provide a large amount of value.

Table 1. Feature Engineering

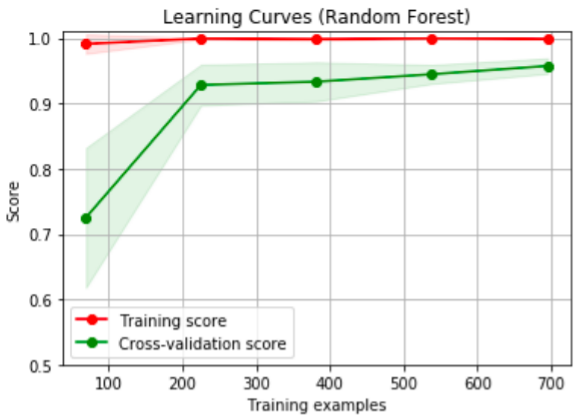
3. Model Designing, Evaluation and Debugging: Steps to design and implement fake news detector:

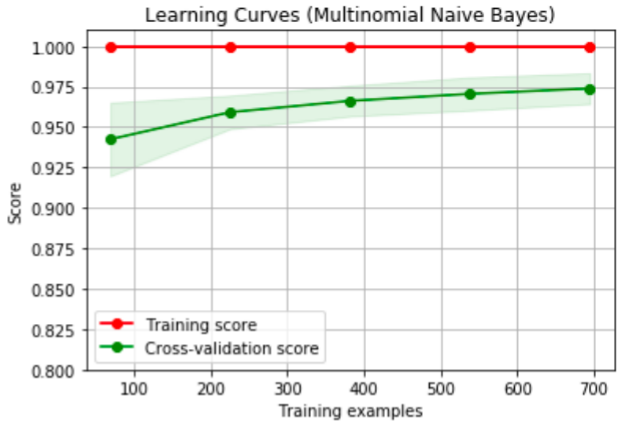
    1. We performed more EDA (Exploratory Data Analysis on datasets and analyzed the most frequently used words by applying CountVectorizer (ngram_range = (1,2)) on the data.
    2. We made note of common frequent noise words and added them to a custom ignore words list which we later used in modeling of the data.
    3. We have begun the modeling process early on but decided to conduct EDA first in order to get to know our data well.
    4. After the data analysis, we began to create and refine our predictive models. We set our predictor (News) and target (Type) variables, conducted a train/test split, and found the best parameters for our models through *tf-idf and count vectorizer*.
    5. We used a combination of vectorizers and classification models to find the best parameters that would give the highest accuracy score. We validate our accuracy score using cross validation score for training data to be compared with testing data.
    6. R2-Score is the percentage of the response variable variation about how close the data is to fittest regression line. *R^2= mean (variation of response data/ total variation) *100*
    7. Tested Results for 3000 Samples (1500 each from true and fake articles dataset) (results will v ary when run again):

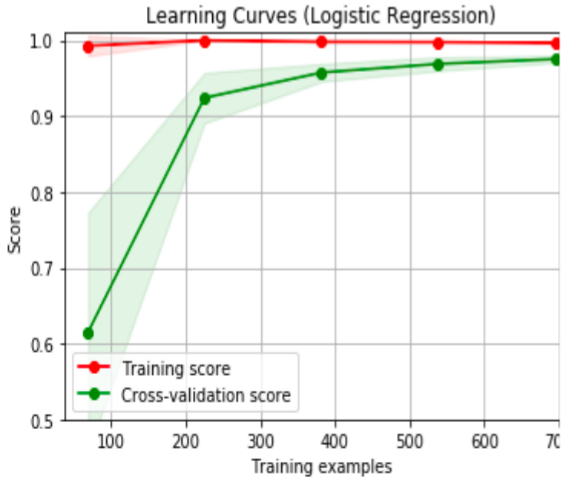
Classifier	*R2-Score*	*Accuracy*	*Execution Time*	*Validation Score*	*Misclassification*
Naïve Bayes	90.63%	97.66%	1.20 sec	Mean:98, Std: 0.05(+/-)	7
Logistic Regression	89.27%	97.32%	1.77sec	Mean:98, Std: 0.03(+/-)	8
Random-Forest	91.95%	97.99%	2.63sec	Mean:96, Std: 0.06(+/-)	4
SVM	79.93%	94.31%	13.133sec	Mean:93, Std: 0.08(+/-)	17

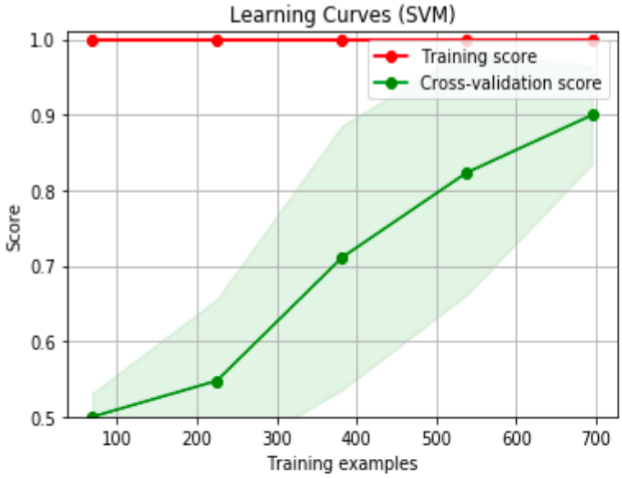
4. Dataset Criticization:
    1) How did they categorize the fake and non-fake which means what criteria is used to detect the fake news?
    2) Additional possible set of Features that should be added to dataset such as source authentication, linkage between news (any 2-news coming from the same source which are based on same topic).
    3) Satirical news means the context of news is unknown.
    4) Source of the news and targeted audience should be used as a feature for classification.
    5) Articles must not be rebutted i.e. they must not be off topic or out of the time frame.
    6) In the process of defining fake news it is important not get to confused with news bias.
    7) Since the majority of fake news articles in the dataset fall under the category of business as well as domestic and world politics it's important that the real news also fall these same categories.
    8) Most articles are similar in contextual terms just the choice of words is different.

5. Learning Curves for Classifiers:









- We implemented four models with the use of CountVectorizer and TfidVectorizer paired with Logi sticRegression, Random Forest, SVM (support vector machine) and MultinomialNB.
- In our opinion the best model for achieving the highest test accuracy score within least execution time is MultinomialNB with accuracy score of 97.66% with execution time 1.20 seconds for 1500s ec samples.
- For large data samples as we have for this project MultinomialNB will predict quickly as compare t o others but we also kept in mind that misclassifications are higher in MultinomialNB as compared to Random Forest. In this regard Random Forest is better than other classifiers.