ABSTRACT

In this information age, there are huge amount of publicly available information on the Internet. Hence, more people tend to consume news digitally as it is not only easy to access but also low cost. The underlying crisis is that the news is not certainly accurate and might cause people to make inappropriate decisions. There are difficulties for human to filter and read all relevant information as it is too time consuming and requires decision making skills. Hence, natural language processing (NLP) techniques and machine learning algorithms emerged as a solution to automate the process of detecting fake news.

This study illustrates how to build classification models which able to classify fake news and non-fake news. The machine learning algorithms used includes the traditional non-neural network model such as Logistic Regression, Support Vector Machine, Multinomial Naive Bayes, Decision Tree Classifier and Random Forest Classifier and neural network model such as Long Short-Term Memory (LSTM). In addition, different types of text representation are used with the machine leaning algorithms.

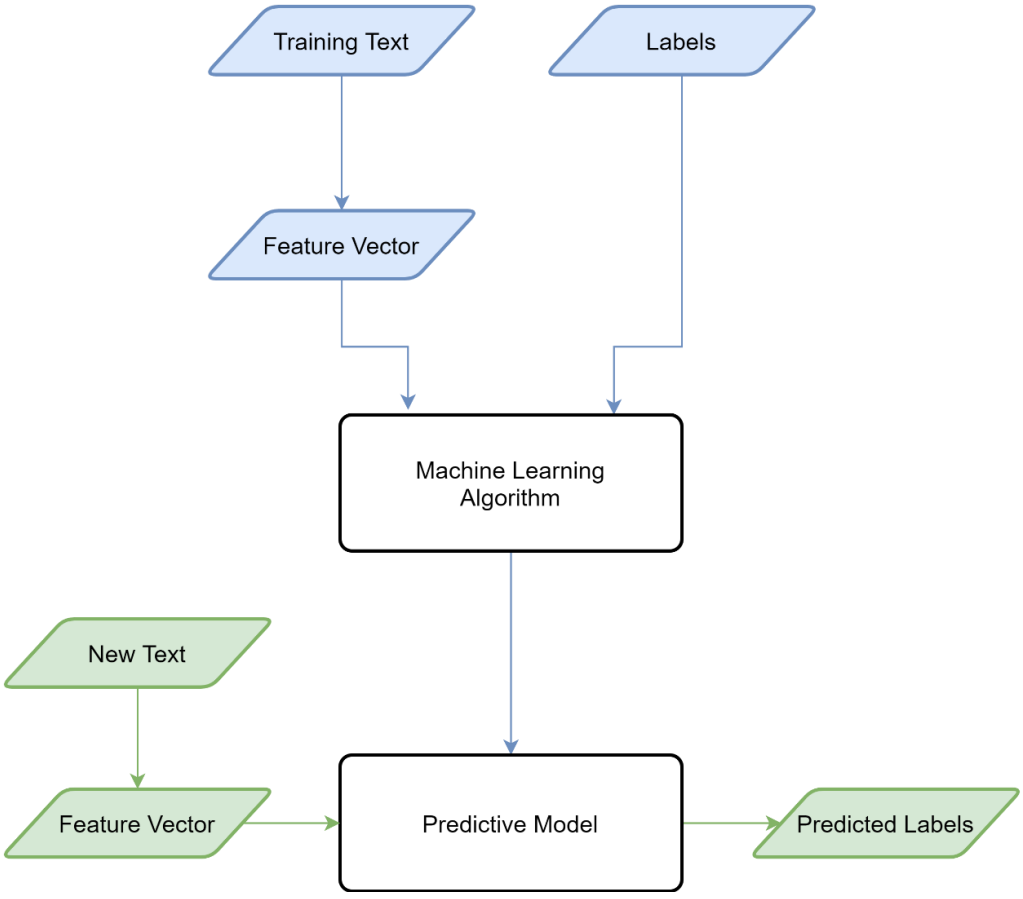
Furthermore, this study also provides the idea of using different types of mapping when converting multiple classes to binary class. In conclusion, three feature extraction techniques and six machine learning algorithms is investigated in the six-way classification and binary classification problems. Experimental evaluation yields the best performance using Term Frequency-Inverted Document Frequency (TF-IDF) as feature extraction technique, and Random Forest Classifier as the machine learning algorithm, with an accuracy of 65 %.

Keywords: Fake News Detection, NLP, Machine Learning

RESEARCH METHODOLOGY

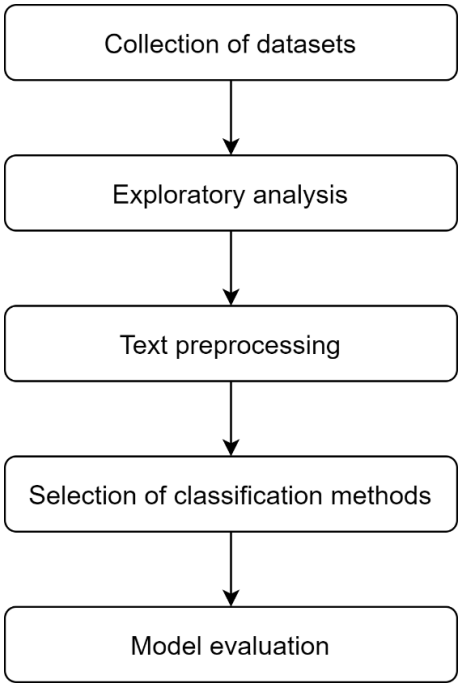
Supervised learning model

The research methodology of this research is to build a supervised learning model which started by providing the machine learning algorithm with the training text and labels. The feature from the training text is needed to be extracted and transformed to appropriate form of feature vector before feeding into the algorithm. The predictive model obtained can be used to predict labels for new text. Of course, the new text is required to go through the same preprocessing steps as the training text to be transformed into useful feature vector.



Overall workflow

There are several main tasks to be carried out to build the scenario shown in the previous figure:



1. **Collection of datasets**

After carefully considering various datasets based on their characteristics, LIAR-PLUS dataset is chosen for this study. LIAR-PLUS dataset is an extended version of LIAR dataset (Wang, 2017) released by Alhindi et al. (2018). Besides of having short statements, it has an additional column that has evidence extracted automatically from the full-text verdict report.

2. **Exploratory analysis**

a. **Description of dataset**

The LIAR-PLUS dataset consists of 15 columns and the description of each column is shown in the table below. The LIAR dataset consists of the first 14 columns while LIAR-PLUS has the additional column at the end, which is the extracted justification.

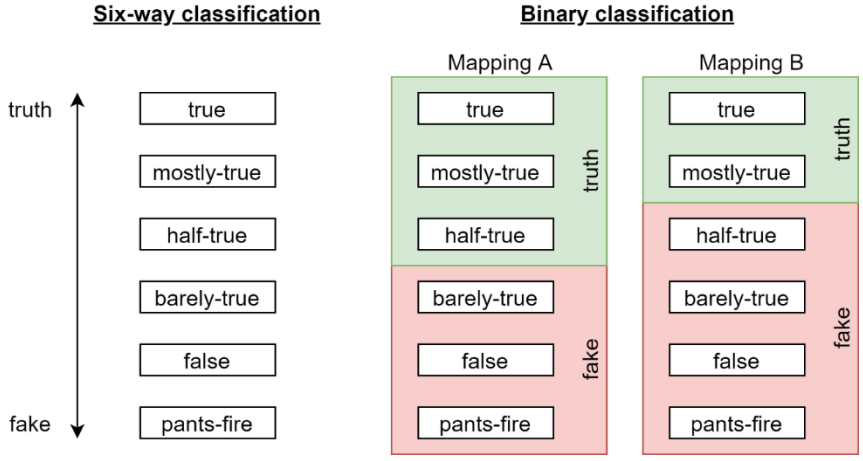
Column 1	The ID of the statement
Column 2	The label
Column 3	The statement

Column 4	The subject(s)	
Column 5	The speaker	
Column 6	The speaker's job title	
Column 7	The state info	
Column 8	The party affiliation	
Column 9	The total credit history count, including the current statements	Barely true counts
Column 10		False counts
Column 11		Half true counts
Column 12		Mostly true counts
Column 13		Pants on fire counts
Column 14	The context (venue/ location of speech or statement)	
Column 15	The extracted justification	

b. Mapping of six classes to two classes labels

The original number of labels is six, which are true, mostly-true, half-true, mostly-false, false and pants-fire. So, six-way classification will be carried out as first task of this study.

In order to have easier understanding, binary classification will be conducted as the second task of this study. Since it is ambiguous to just consider half-true as "truth", two types of mapping are used as shown in the figure below. Namely, mapping A which consider true, mostly-true and half-true as "truth", and mostly-false, false and pants-fire as "fake". Mapping B considers only true and mostly-true as "truth", and the rest of the labels are considered as "fake".



3. Text preprocessing

- Tokenization
- Stemming
- Stop word removal

4. Feature extraction

As machine learning algorithms works with numerical features, the features are required to be extracted from the data and represented in vectors. In other words, the list of words is converted to feature vectors in this step using:

- Bag-of-words variant: Term Frequency-Inverse Document Frequency (TF-IDF)
- Bag-of-words variant: Binary weighting
- Word2Vec

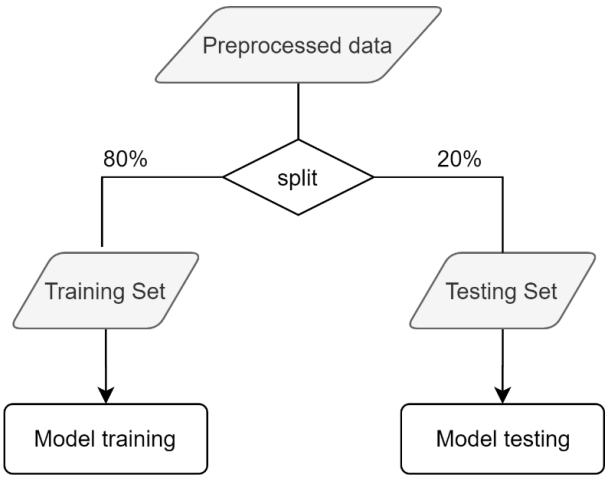
5. Machine learning algorithms

- Logitstic Regression
- Multinomial Naïve Bayes
- Support Vector Machine
- Decision Tree Classifier
- Random Forest Classifier
- Long Short-Term Memory (LSTM) (not shown in this repository)

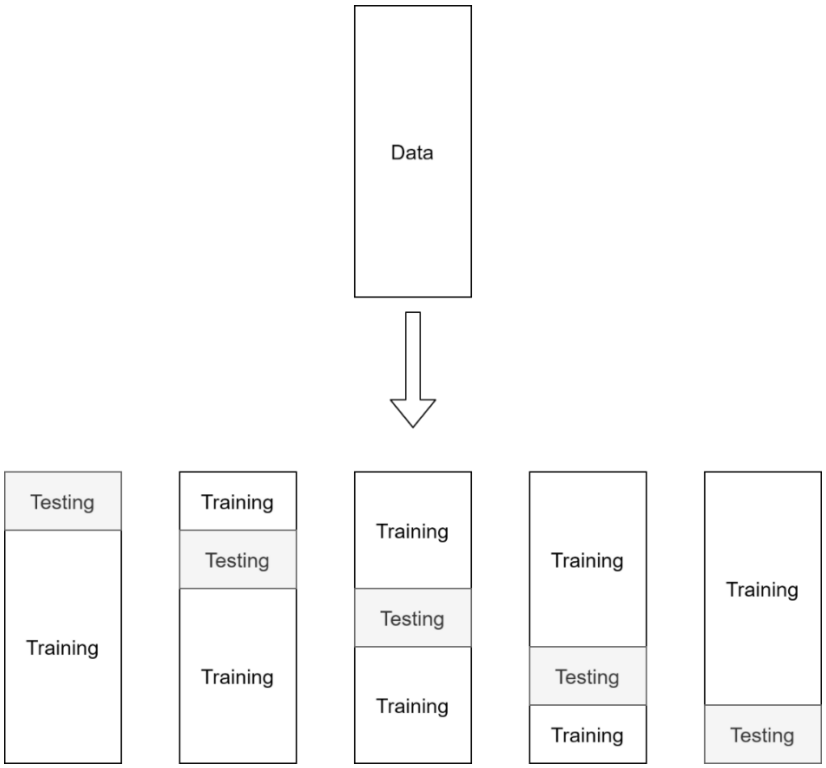
6. Model evaluation

Model evaluation is the process of choosing between models, tuning parameters or features. Model evaluation procedure is used to estimate how well a model will generalize to out-of-sample data. Then, model evaluation metric is used to quantify how well the model perform.

a. <u>Train/Test Split</u>



b. <u>K-fold cross validation</u>



c. <u>Model evaluation metric</u>

- Classification accuracy
- Confusion matrix

IMPLEMENTATION

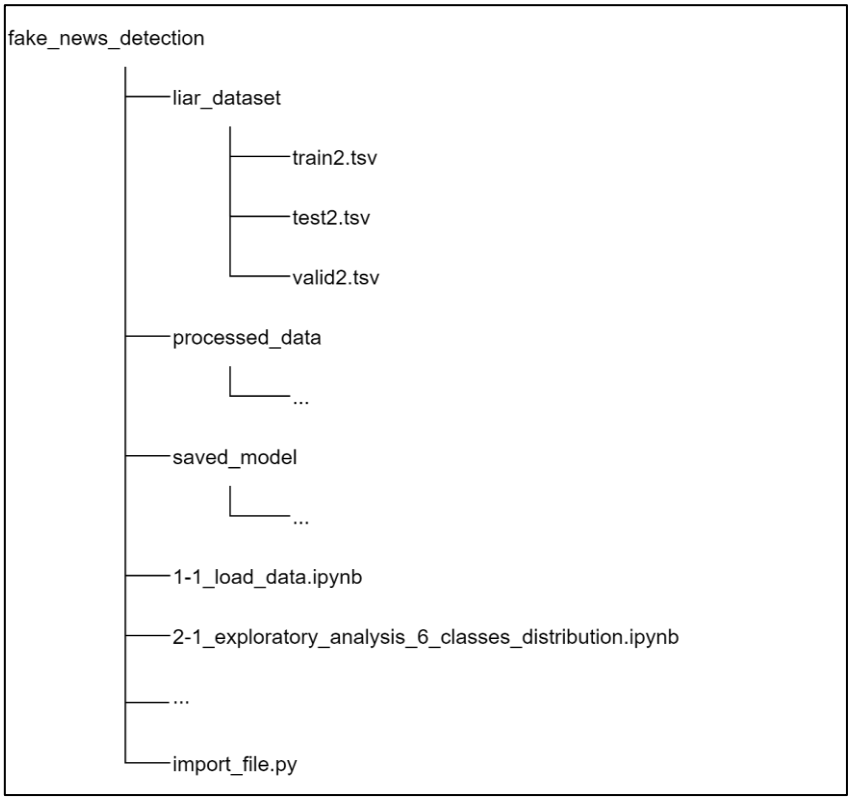
Specifications of implementation

Tools/ Library	Version
Python	3.9.4
Jupyter Notebook	6.1.4
NumPy	1.20.2
Pandas	1.2.5
Matplotlib	3.3.4
Seaborn	0.11.1
Scikit-learn	0.24.2
NLTK	3.6.2
wordcloud	1.8.1

Repeatable and presentable workflow

Although it is possible to implement all the tasks in a single notebook, but it is hard to locate error and lack of clarity. So, the tasks are separated to multiple notebooks with consistent naming scheme.

1. **Structure of directory**



2. **Structure of notebook**

For simplicity, the start of each notebook will include the same imports in “import_file.py”.

3. **Data flow between notebooks**

All of these practices will build a workflow which enables data flow between multiple notebooks without corrupting the output (processed data, model, etc.) of previous notebook. An analogy of data flow between the notebooks is shown in the figure below.

The original files which are train.tsv, test.tsv and valid.tsv will remain unchanged after “load_data” process. After “load_data” process, the processed data will be saved in pickles. Both “model_building_1” and “model_building_2” process can use processed_data.pkl

repeated times without interrupting each other. Each of the model building processes will save their model separately too. Then, each saved model can be used separately to do predictions.