IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

FAKE NEWS CLASSIFICATION

[1]Pratibha Patil, [2]Sitaram Longani, [3] Anil Kumar Jakkani

[1,2,3]Asst. Professor, Department of E&TC , ISBM COE, Nande, Pune,India

Abstract: Fake news can be interpreted as falsified information or wrong facts presented in the newspapers, television news, online media with a spiteful intent to damage the reputation of a person, organization or any other society for personal or professional benefits. Few media outlets present incorrect narrative about prominent personalities to gain Television Rating Points (TRP's), to increase viewership to yield higher revenue, to smear a person to settle previous conflicts etc. Given the serious consequences that result a fake news article and its impact on formulations of key decisions in everyday life, it is necessary to come up with a resolution plan to stop the spread of fake news and handle it in a suitable manner. In this project, we try to resolve the problem of fake news by utilizing Natural Language Processing (NLP) which is a Data Science sub-branch to come up with a solution that can be used to classify the fake news.

Index Terms - Logistic Regression, Multinomial Naive Bayes, Random Forest Support Vector Machine& XGBoost

I. INTRODUCTION

In variety of fields, including computer science, fake news has become the key research topic. Today's troublesome issue is that press, particularly social media, has become the place for false information that impacts the integrity of entire news environment. This problem is rising because any one can enroll on social media as a news source without any cost (e.g., anyone can create a Facebook page pretending to be a news media organization). There are increasing concerns about fake news outlets publishing "true" news stories, and often adding "fake" followers widely to those news stories. Since the widespread dissemination of fake news can have a significant adverse impact on individuals and society, the lack of robust fact-checking techniques is particularly worrying. We provide a detailed account of fake news detection as a text classification problem, to be solved using natural language processing (NLP) tools, and our tests show that fake news articles are detectable, particularly given enough training information.

The project is sub-divided following section. These are:

1. Loading necessary libraries
2. Loading Dataset from a CSV file or from a Table.
3. Summarization of Data to understand Dataset (Descriptive Statistics)
4. Visualization of Data to understand Dataset (Plots, Graphs etc.)
5. Data pre-processing and Data transformation
6. Hyper parameter tuning to find the optimal parameters for the classification models.
7. Applying different learning algorithms on the training dataset.
8. Evaluating the performance of the fitted model using evaluation metrics like confusion matrix, precision recall curves.

The source of this dataset is Kaggle. The dataset consists of text and metadata which is scrapped from 12,999 post on 244 different websites. The dataset has 20 columns and 12999 rows in the dataset where each row corresponds to a new article. For the classification problem under consideration we have used 'title' and 'text' as input columns and column 'type' as output variable. The title, text and type are of string data type. The data is classified into following types - bias, conspiracy, fake, satire, hate, junksci and state. However, in some of the rows some values were missing for some of the features and value for some of the features are 'NaN'. The data was cleaned in the Data pre-processing steps mentioned below, before using it for training different models.

Dataset URL: https://www.kaggle.com/mrisdal/fake-news
Columns: uuid, ord_in_thread, author, published, title, text, language, crawled, site_url, country, domain_rank, thread_title, spam_score, main_img_url, replies_count, likes, participants_count, comments, shares, type.

II. METHODS

Data Pre-processing Steps

Step 1: Feature Engineering
Features considered- Title and Test (Title feature used for handling missing text feature values).

Step 2: Data distribution based on output class.
The dataset is imbalanced.

Step 3: Handling missing text values.
Missing text values can be handled in two ways:
1. Simple Imputing missing values with most frequent strategy
2. Replacing the NaN values of text column with the corresponding title value

Step 4: Train Test Split. Keeping aside thirty percent of the data for test set.

Step 5:
a. Stemming: Stemming is a method of text standardization(or word standardization) in the Natural Language Processing area that is used to prepare text, words and documents for further processing Stemming is a method of reducing word inflexion to its root forms such as mapping a group of words to the same stem even though the stem itself is not a valid word in the language.
We use Snowball stemmer for this purpose. This algorithm is also known as the Porter2 stemming algorithm.

b. Stop words:
Stop Words are words used to be used in search queries that do not contain important meaning. These words are typically filtered out of search queries because they return a vast quantity of unnecessary information.

c. Removing numerical and special characters and converting the words to lower case.

Step 6: Bag of Words
CountVectorizer: Convert a collection of text documents to a matrix of token counts
TfidfTransform: Transform a count matrix to a normalized tf or tf-idf representation
TF-IDF: Vector representation of Text. TF-IDF is an abbreviation for Term Frequency-Inverse Document Frequency and is a very common algorithm to transform text into a meaningful representation of numbers.
Convert a collection of raw documents to a matrix of TF-IDF features.
TfidfVectorizer: Equivalent to CountVectorizer followed by TfidfTransformer.
We conduct our experiment by implementing the following classification models:
1. Logistic Regression
2. Multinomial Naïve Bayes
3. Random Forest
4. Support Vector Machine
5. XGBoost

Approach:

Creating a model pipeline from the imblearn package. It takes care to upsample using the upsampling method specified in the pipeline. It performs upsampling when fit() is called on the pipeline, and does not upsample test data (when called predict()).
We use Randomized Search CV for tuning the hyper parameters. In comparison to Grid Search CV, it uses random combinations of hyper parameters in every iteration and less number of parameter settings are tried, but achieves a better coverage due to randomness and consumes less computational time. Perform oversampling using SMOTE (Synthetic Majority Oversampling Technique) technique. Here specifying SMOTE upsampling method in the pipeline and performing oversampling as part of parameter tuning using Randomized Search Cross Validation.

For evaluation metrics, here use macro-averages of precision, recall, f1-score, accuracy score and hamming loss for each class. A macro average metric measures the score for each class separately and then takes the average by treating all the classes equally. For XGBoost, first encode the output labels using Label Encoder and then convert the dataframe into XGBoost's Dmatrix object prior to fitting the model. We use XGBClassifier with tuned parameters. For tuning the hyper parameters, we have used Bayesian Optimization technique as it takes less number of steps in finding the optimal parameters for XGBoost model when compared to randomized search. The computational time is expensive for random search in XGBoost model when compared to other classification models. Bayesian methods use the outcomes of previous assessment results to decide the next values for evaluation.

IV. RESULTS AND DISCUSSION

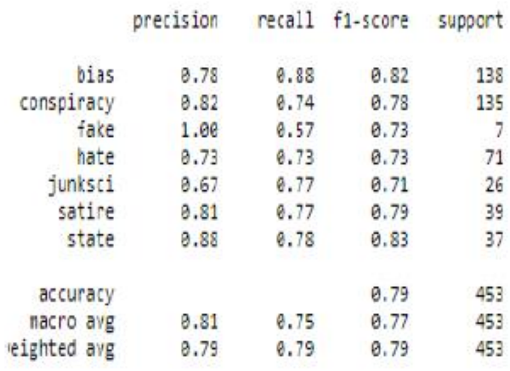
We use classification reports, confusion matrix, confusion matrix for each class and precisionrecall- f1 curves as the evaluation metrics to analyze the results obtained for our models. Overall, XGBoost generalizes well when compared to Logistic Regression, Multinomial Naïve Bayes, Random Forest and Support Vector Machine.

Table 1: Results Summary

MODEL	LOGISTIC REGRESSION	MULTINOMIAL NAÏVE BAYES	RANDOM FOREST	SVM	XGBOOST
Precision	0.81	0.56	0.80	0.76	0.88
Recall	0.75	0.63	0.74	0.46	0.84
Accuracy	0.79	0.52	0.72	0.59	0.82
F1-Score	0.77	0.50	0.75	0.52	0.85
Hamming Loss	0.21	0.48	0.28	0.41	0.15

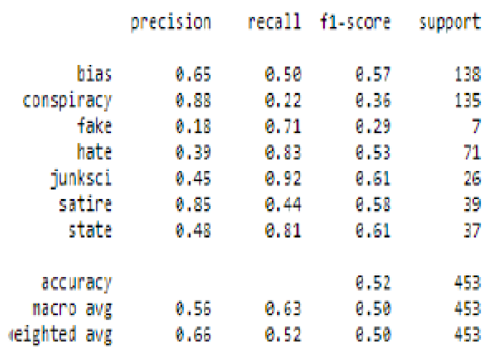
DISCUSSION: We can improve the model in the following ways:
1. Adding more data Using huge volumes of data to train the models will help improve performance, leading to more accurate models.
2. Enhancing the quality of data by collecting news articles published in various domains as the vocabulary varies with domain. The news articles collected from social media platform will contain improper words like "awsm, fyn, baaad" etc are not used in news articles by news agencies.
3. Using an Exhaustive Stopword List: Apart from language stopwords, there are some other supporting words as well which are of lesser importance than any other terms. These includes: Location stopwords – Country names, Cities names etc. Time stopwords and Numerical stopwords
4. Eliminating features with extremely low frequency: There are words that rarely occur in many news articles and these words usually do not play much role in the text classification. Removing features for the words that rarely occur in news dataset can result in improving in the performance for different models.
5. Use Complex Features: N-grams and part of speech tags. Joining multiple words together to form a single feature and using these features along with single words for features can help in improving the model accuracy. Combination of N words into a single feature are known N-grams.
6. Using domain specific knowledge and human insight can help in choosing better features for the prediction task under consideration.
7. Multiple Approaches: Using different classification algorithms for the news classification task can help in choosing the best solution. Hence, using multiple approaches and comparing their performance on validation dataset can help in choosing the correct approach to solve the classification problem under consideration.

Logistic Regression



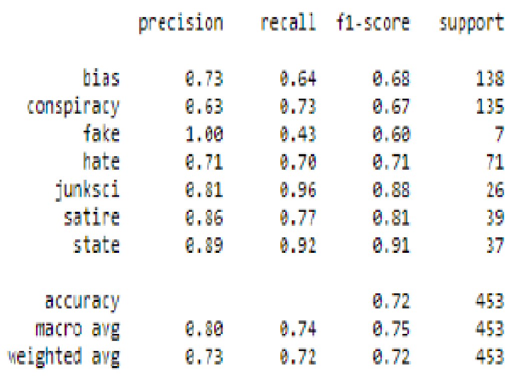
```
              precision    recall  f1-score   support

        bias       0.78      0.88      0.82       138
  conspiracy       0.82      0.74      0.78       135
        fake       1.00      0.57      0.73         7
        hate       0.73      0.73      0.73        71
     junksci       0.67      0.77      0.71        26
      satire       0.81      0.77      0.79        39
       state       0.88      0.78      0.83        37

    accuracy                           0.79       453
   macro avg       0.81      0.75      0.77       453
weighted avg       0.79      0.79      0.79       453
```

Multinomial Naive Bayes Classifier



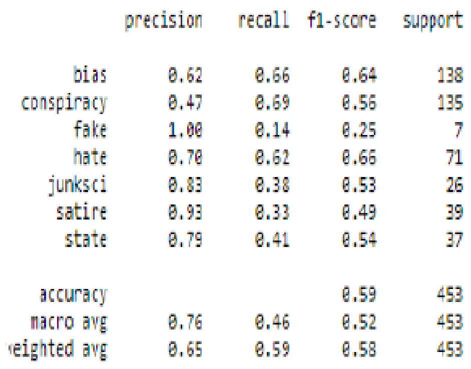
```
              precision    recall  f1-score   support

        bias       0.65      0.50      0.57       138
  conspiracy       0.88      0.22      0.36       135
        fake       0.18      0.71      0.29         7
        hate       0.39      0.83      0.53        71
     junksci       0.45      0.92      0.61        26
      satire       0.85      0.44      0.58        39
       state       0.48      0.81      0.61        37

    accuracy                           0.52       453
   macro avg       0.56      0.63      0.50       453
weighted avg       0.66      0.52      0.50       453
```

Random Forest Classifier



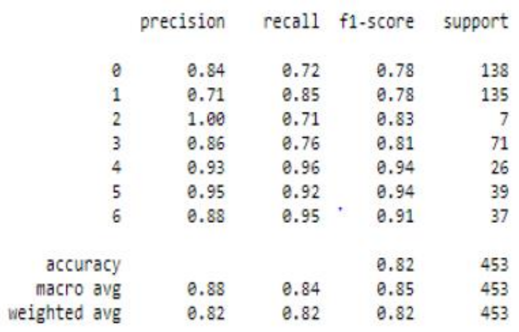
```
              precision    recall  f1-score   support

        bias       0.73      0.64      0.68       138
  conspiracy       0.63      0.73      0.67       135
        fake       1.00      0.43      0.60         7
        hate       0.71      0.70      0.71        71
     junksci       0.81      0.96      0.88        26
      satire       0.86      0.77      0.81        39
       state       0.89      0.92      0.91        37

    accuracy                           0.72       453
   macro avg       0.80      0.74      0.75       453
weighted avg       0.73      0.72      0.72       453
```

Support Vector Machine



```
              precision    recall  f1-score   support

        bias       0.62      0.66      0.64       138
  conspiracy       0.47      0.69      0.56       135
        fake       1.00      0.14      0.25         7
        hate       0.70      0.62      0.66        71
     junksci       0.83      0.38      0.53        26
      satire       0.93      0.33      0.49        39
       state       0.79      0.41      0.54        37

    accuracy                           0.59       453
   macro avg       0.76      0.46      0.52       453
weighted avg       0.65      0.59      0.58       453
```

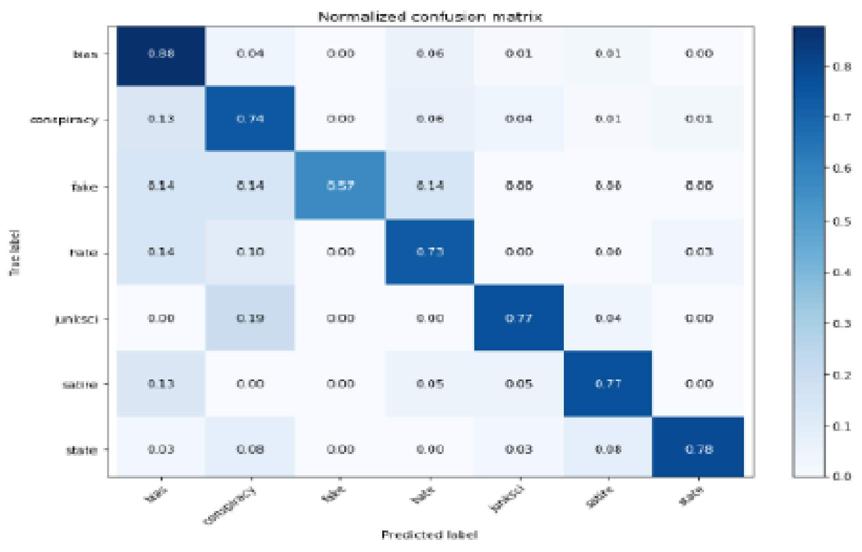
XGBoost



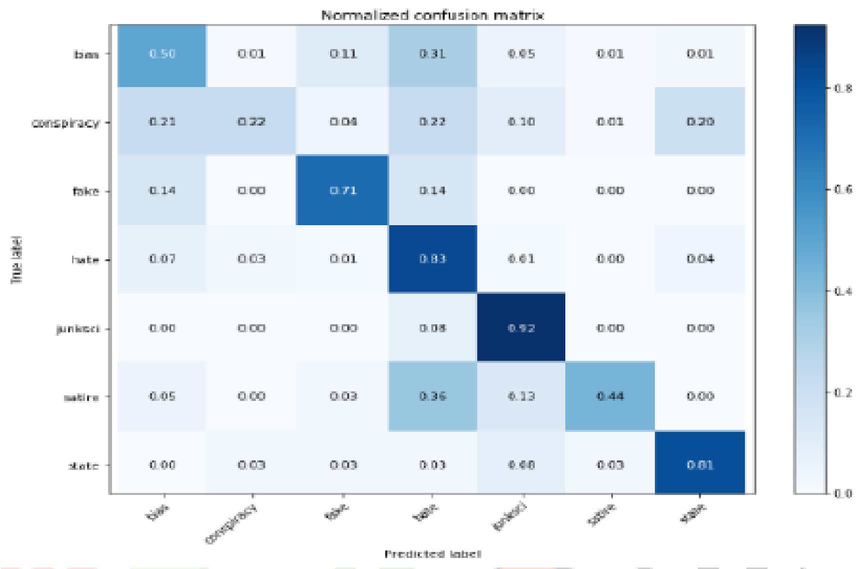
```
              precision    recall  f1-score   support

           0       0.84      0.72      0.78       138
           1       0.71      0.85      0.78       135
           2       1.00      0.71      0.83         7
           3       0.86      0.76      0.81        71
           4       0.93      0.96      0.94        26
           5       0.95      0.92      0.94        39
           6       0.88      0.95      0.91        37

    accuracy                           0.82       453
   macro avg       0.88      0.84      0.85       453
weighted avg       0.82      0.82      0.82       453
```

Figure 2: Confusion Matrix

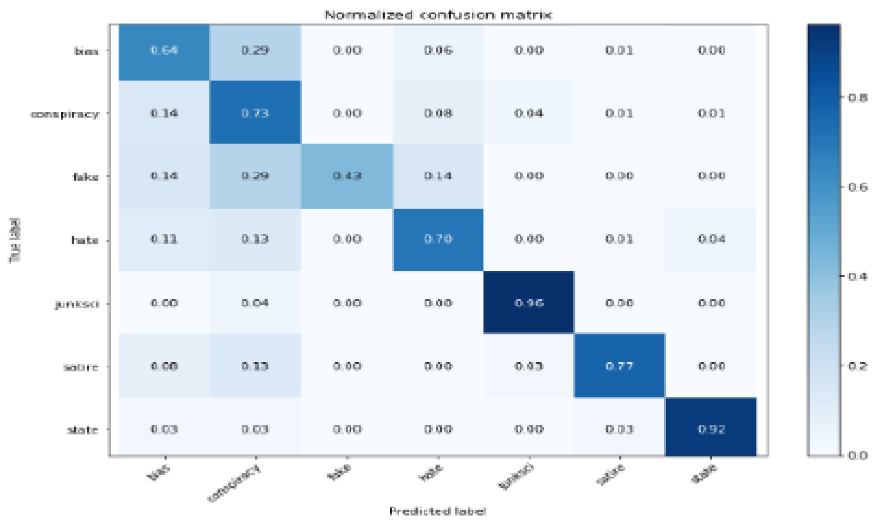
Logistic Regression



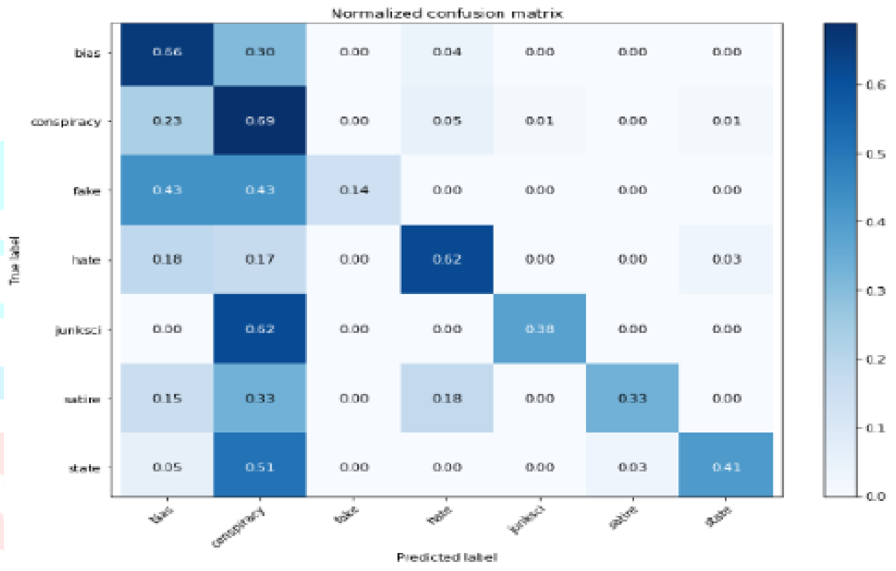
Multinomial Naive Bayes Classifier



Random Forest Classifier



Support Vector Machine



XGBoost

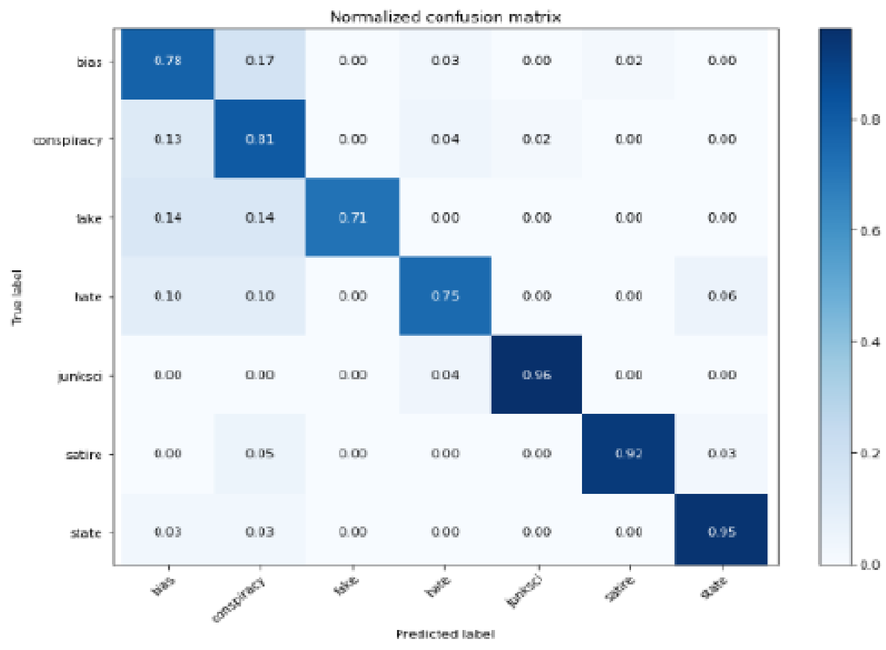
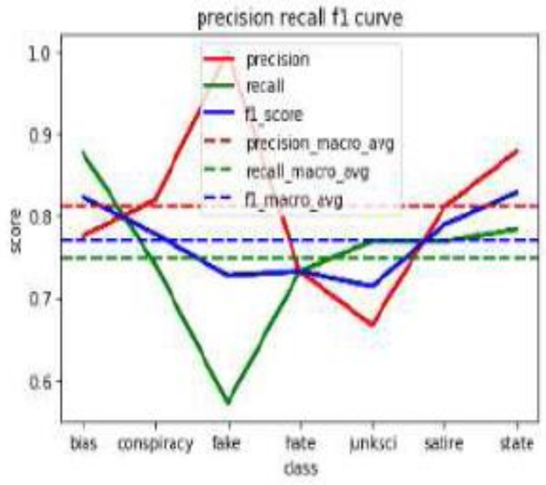
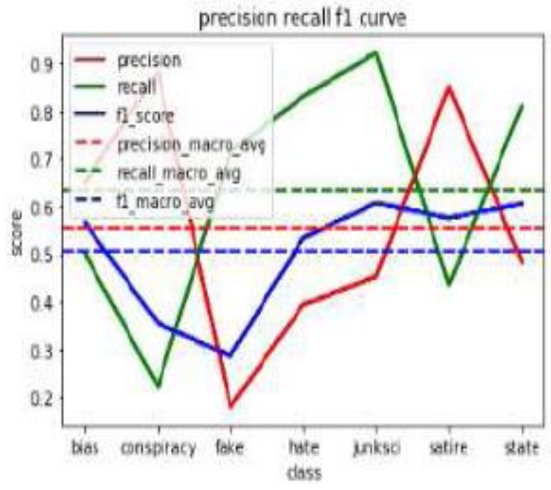


Figure 3: Precision Recall F1 Score

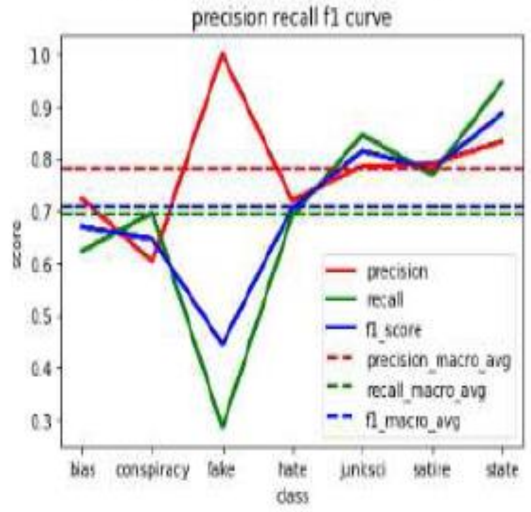
Logistic Regression



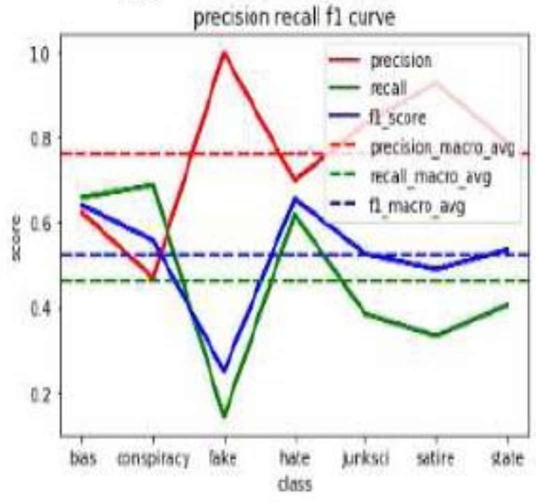
Multinomial Naive Bayes Classifier



Random Forest Classifier



Support Vector Machine



XGBoost

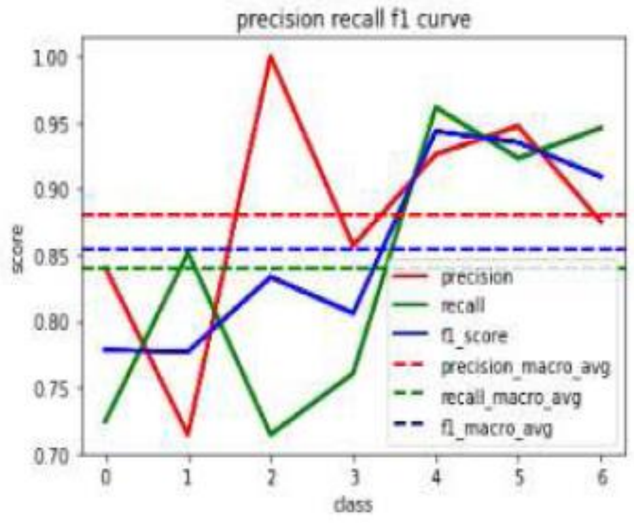
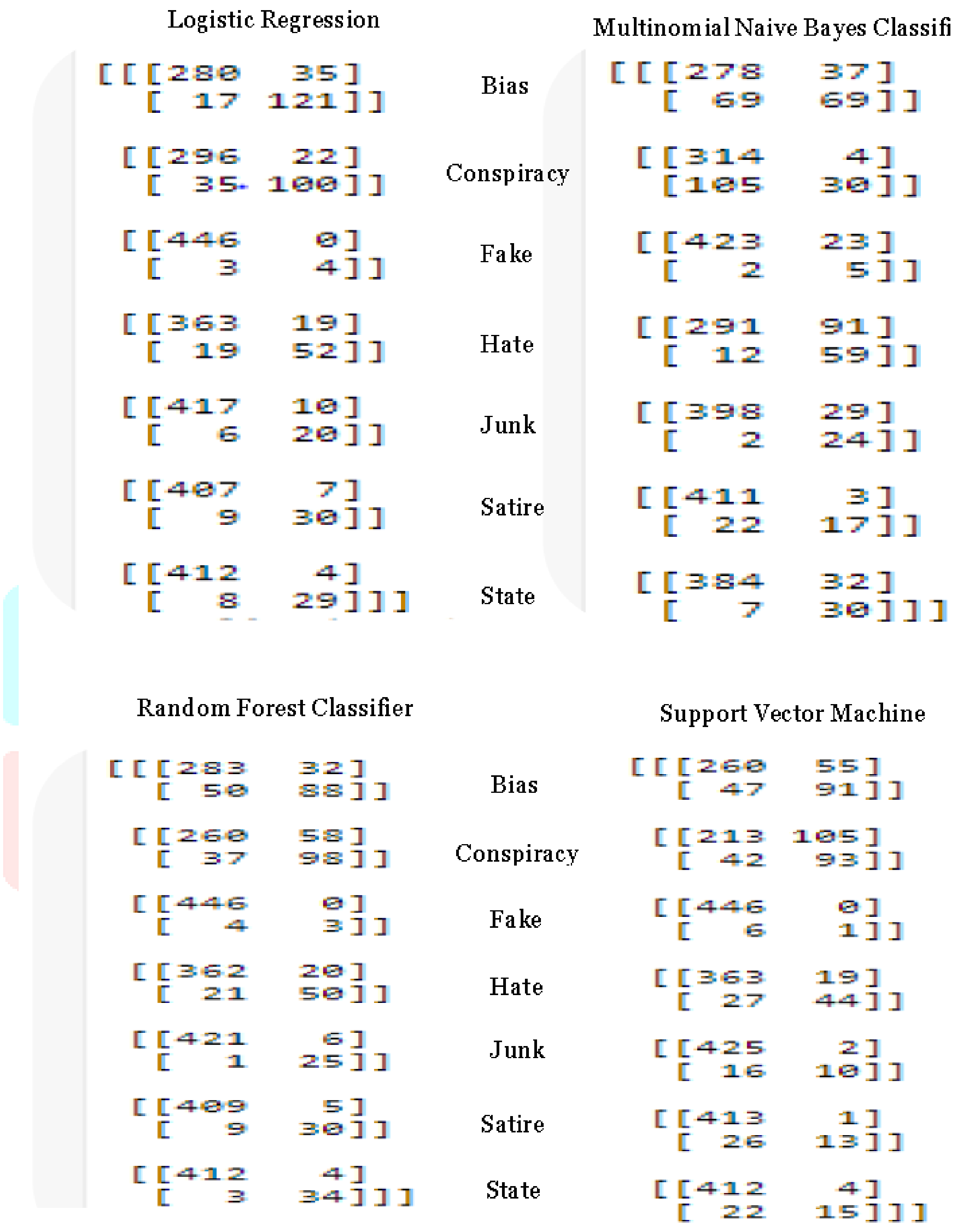


Figure 4: Confusion Matrix for Each Class



Logistic Regression

```
[[[280  35]                Bias
  [ 17 121]]

 [[296  22]                Conspiracy
  [ 35 100]]

 [[446   0]                Fake
  [  3   4]]

 [[363  19]                Hate
  [ 19  52]]

 [[417  10]                Junk
  [  6  20]]

 [[407   7]                Satire
  [  9  30]]

 [[412   4]                State
  [  8  29]]]
```

Multinomial Naive Bayes Classifi

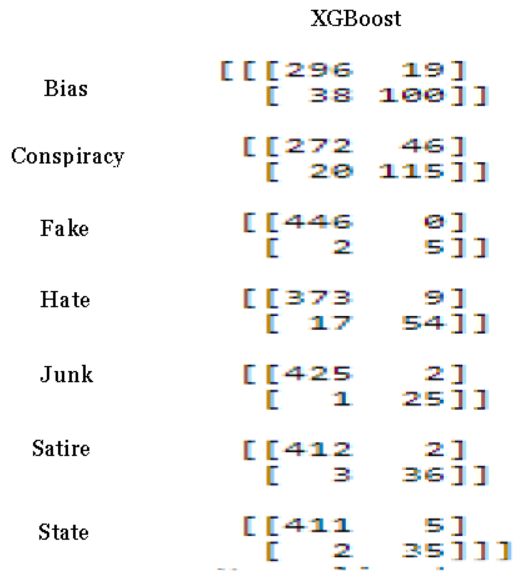
```
[[[278  37]                Bias
  [ 69  69]]

 [[314   4]                Conspiracy
  [105  30]]

 [[423  23]                Fake
  [  2   5]]

 [[291  91]                Hate
  [ 12  59]]

 [[398  29]                Junk
  [  2  24]]

 [[411   3]                Satire
  [ 22  17]]

 [[384  32]                State
  [  7  30]]]
```

Random Forest Classifier

```
[[[283  32]                Bias
  [ 50  88]]

 [[260  58]                Conspiracy
  [ 37  98]]

 [[446   0]                Fake
  [  4   3]]

 [[362  20]                Hate
  [ 21  50]]

 [[421   6]                Junk
  [  1  25]]

 [[409   5]                Satire
  [  9  30]]

 [[412   4]                State
  [  3  34]]]
```

Support Vector Machine

```
[[[260  55]                Bias
  [ 47  91]]

 [[213 105]                Conspiracy
  [ 42  93]]

 [[446   0]                Fake
  [  6   1]]

 [[363  19]                Hate
  [ 27  44]]

 [[425   2]                Junk
  [ 16  10]]

 [[413   1]                Satire
  [ 26  13]]

 [[412   4]                State
  [ 22  15]]]
```



XGBoost

Bias	`[[[296  19]`
	`  [ 38 100]]`

Conspiracy	`[[272  46]`
	`  [ 20 115]]`

Fake	`[[446   0]`
	`  [  2   5]]`

Hate	`[[373   9]`
	`  [ 17  54]]`

Junk	`[[425   2]`
	`  [  1  25]]`

Satire	`[[412   2]`
	`  [  3  36]]`

State	`[[411   5]`
	`  [  2  35]]]`

References

[1] Fake News Dataset, https://www.kaggle.com/mrisdal/fake-news

[2] Fatemeh Torabi Asr, Maite Taboada, 2019. Big Data and quality data for fake news and misinformation detection, https://journals.sagepub.com/doi/full/10.1177/2053951719843310

[3] Jillian Tompkins, 2018. Disinformation Detection: A review of linguistic feature selection and classification models in news veracity assessments, https://www.albany.edu/~jt972467/tompkins_669final.pdf

[4] Miriam Seoane Santos, Jastin Pompeu Soares, Pedro Henriques Abreu, Helder Araujo and Joao Santos, 2018. Cross-Validation for Imbalanced Datasets: Avoiding Overoptimistic and Overfitting Approaches,https://www.researchgate.net/publication/328315720_Cross Validation_for_Imbalanced_Datasets_Avoiding_Overoptimistic_and_Overfitting_App roaches

[5] Trevor Hastie, Robert Tibshirani and Jerome Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition, 2008.

[6] Randomized Search on hyper parameters, https://scikitlearn. org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

[7] Stemming and Lemmatization in Python – by Hafsa Jabeen, https://www.datacamp.com/community/tutorials/stemming-lemmatization-python

[8] Imbalanced learn pipeline, https://imbalancedlearn.readthedocs.io/en/stable/generated/imblearn.pipeline.Pipeline.html

[9] Metrics and scoring: quantifying the quality of predictions, https://scikitlearn. org/stable/modules/model_evaluation.html

[10] Oversampling of imbalanced data, https://www.researchgate.net/post/should_oversampling_be_done_before_or_within _cross-validation

[11] Boosting Algorithms Simplified – by Tavish Srivastava, https://www.analyticsvidhya.com/blog/2015/05/boosting-algorithms-simplified/

[12] XGBoost Python Package API Reference, https://xgboost.readthedocs.io/en/latest/python/python_api.html

[13] Bayesian Optimization Documentation, https://github.com/fmfn/BayesianOptimization#bayesian-optimization