FLIP ROBO

Fake News Classifier Project

Submitted by:

Moncy Kurien

ACKNOWLEDGMENT

INTRODUCTION

- Business Problem Framing

  Recently, the number of fake news out in public has increased. Several public concerns about this problem and some approaches to mitigate the problem were expressed. In this problem we are going to try and solve this problem using data science and machine learning.

- Conceptual Background of the Domain Problem

  The authenticity of Information has become a longstanding issue affecting businesses and society, both for printed and digital media. On social networks, the reach and effects of information spread occur at such a fast pace and so amplified that distorted, inaccurate, or false information acquires a tremendous potential to cause real-world impacts, within minutes, for millions of users.

- Motivation for the Problem Undertaken

  It is very important to know if a news that reaches the public is real or fake. Because fake news can create adverse effects in the lives of the public and society. A model to detect a fake news will be very beneficial for the society.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

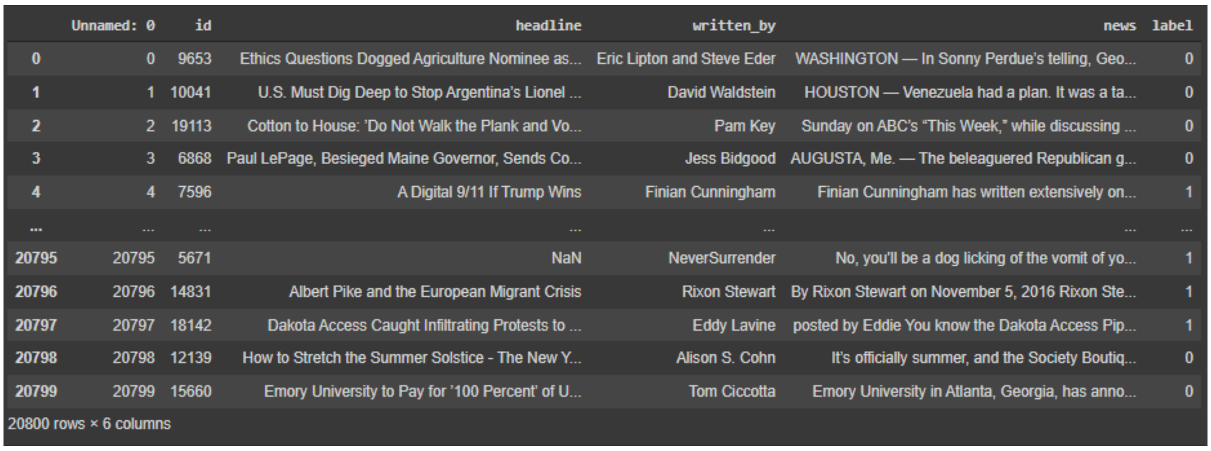
  The news texts, headlines and the author names are merged and are cleaned, lemmatized and vectorized. These vectors are then used as input for the machine learning algorithm.

- ## Data Sources and their formats

  There are 6 columns in the dataset provided to you. The description of each of the column is given below:

  1. “id”: Unique id of each news article
  2. “headline”: It is the title of the news.
  3. “news”: It contains the full text of the news article
  4. “Unnamed:0”: It is a serial number
  5. “written_by”: It represents the author of the news article
  6. “label”: It tells whether the news is fake (1) or not fake (0).

  Screen-Shot:



	Unnamed: 0	id	headline	written_by	news	label
0	0	9653	Ethics Questions Dogged Agriculture Nominee as...	Eric Lipton and Steve Eder	WASHINGTON — In Sonny Perdue's telling, Geo...	0
1	1	10041	U.S. Must Dig Deep to Stop Argentina's Lionel ...	David Waldstein	HOUSTON — Venezuela had a plan. It was a ta...	0
2	2	19113	Cotton to House: 'Do Not Walk the Plank and Vo...	Pam Key	Sunday on ABC's “This Week,” while discussing ...	0
3	3	6868	Paul LePage, Besieged Maine Governor, Sends Co...	Jess Bidgood	AUGUSTA, Me. — The beleaguered Republican g...	0
4	4	7596	A Digital 9/11 If Trump Wins	Finian Cunningham	Finian Cunningham has written extensively on...	1
...	...	...	...	...	...	...
20795	20795	5671	NaN	NeverSurrender	No, you'll be a dog licking of the vomit of yo...	1
20796	20796	14831	Albert Pike and the European Migrant Crisis	Rixon Stewart	By Rixon Stewart on November 5, 2016 Rixon Ste...	1
20797	20797	18142	Dakota Access Caught Infiltrating Protests to ...	Eddy Lavine	posted by Eddie You know the Dakota Access Pip...	1
20798	20798	12139	How to Stretch the Summer Solstice - The New Y...	Alison S. Cohn	It's officially summer, and the Society Boutiq...	0
20799	20799	15660	Emory University to Pay for '100 Percent' of U...	Tom Ciccotta	Emory University in Atlanta, Georgia, has anno...	0

20800 rows × 6 columns

- ## Data Preprocessing Done

  1. Removed records with missing values.
  2. Merged the columns headline, written_by and news to form a new column ‘combined_text’.
  3. Converted all the characters in the ‘combined_text’ column to lowercase.
  4. Replaced all email addresses, website links, currencies, phone numbers and any numbers with constant texts link emailaddress, webaddress, currencyamount, phonenumber and numbr respectively.
  5. Removed all non-alphabetic characters.
  6. Replaced all multiple blank spaces with single blank space.
  7. Changed the datatype of ‘combined_text’ as ‘str’.
  8. Removed the stopwords.

9. Used the ‘cleaned_combined_text’ field to create features using TFIDF with monograms, bigrams and trigrams.
10. Finally the cleaned_combined_text is lemmatized.

- ## Data Inputs- Logic- Output Relationships

The input data consists of float values which are derived using TFIDF method from the ‘lemma_cleaned_combined_text’ column in the dataset.

The TFIDF method uses monograms, bigrams and trigrams to create the independent features for the model and the output contains numerical classes.

The model approximates the function between the input and the output.

- ## Hardware and Software Requirements and Tools Used

1. Google Colab
2. SKLEARN
3. MATPLOTLIB
4. PANDAS
5. NUMPY
6. Google Translator
7. Textblob
8. TensorFlow

Model/s Development and Evaluation

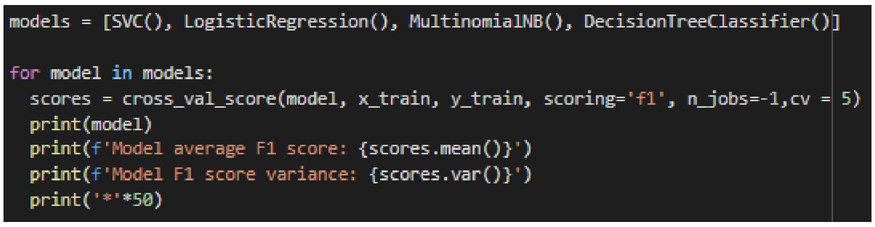
- Identification of possible problem-solving approaches (methods)
  1. Clean the dataset using NLP approaches.
  2. Lemmatize the cleaned text.
  3. Vectorize the text inputs.
  4. Compare different models and identify the suitable model.
  5. F1 score is used as the primary evaluation metric.
  6. Accuracy, Confusion matrix and AUC ROC score is used as secondary metrics.

- Testing of Identified Approaches (Algorithms)
  1. DecisionTreeClassifier
  2. MultinomialNB
  3. LogisticRegression
  4. SVC
  5. RandomForestClassifier
  6. AdaBoostClassifier
  7. GradientBoostingClassifier
  8. XGBClassifier

- Run and Evaluate selected models

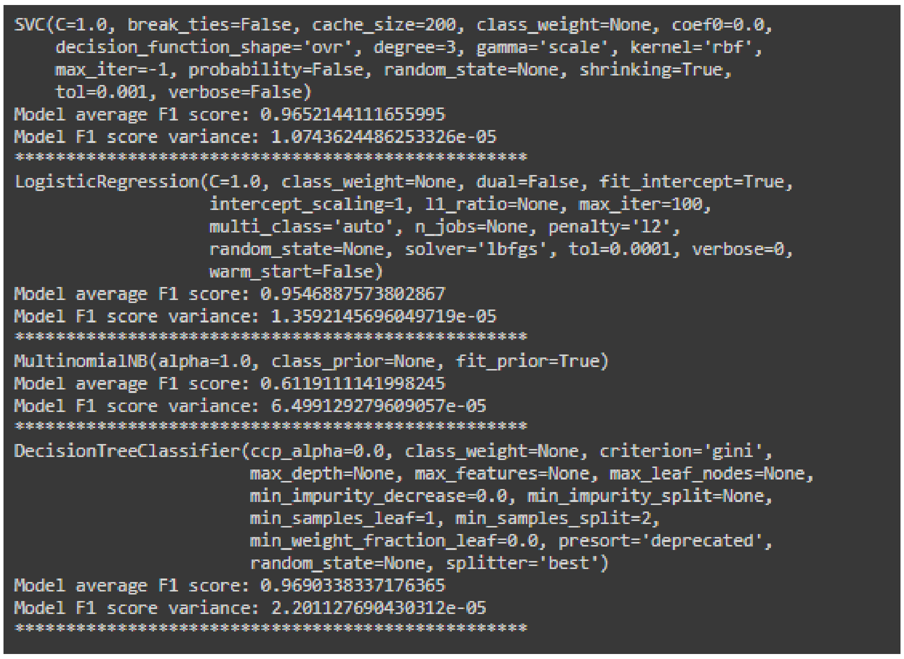
<u>Comparing basic models using F1 Score metric:</u>

Code:



```
models = [SVC(), LogisticRegression(), MultinomialNB(), DecisionTreeClassifier()]

for model in models:
  scores = cross_val_score(model, x_train, y_train, scoring='f1', n_jobs=-1,cv = 5)
  print(model)
  print(f'Model average F1 score: {scores.mean()}')
  print(f'Model F1 score variance: {scores.var()}')
  print('*'*50)
```

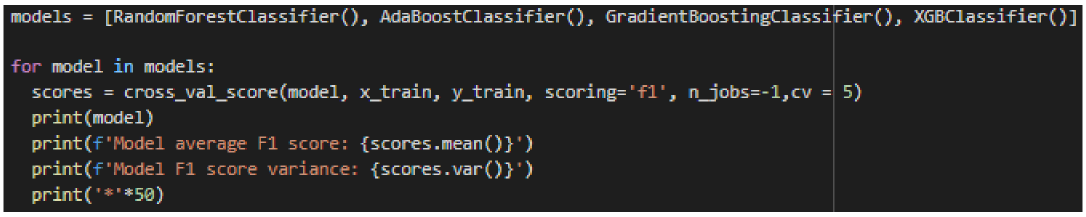
Results:



```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
Model average F1 score: 0.9652144111655995
Model F1 score variance: 1.0743624486253326e-05
**************************************************
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
Model average F1 score: 0.9546887573802867
Model F1 score variance: 1.3592145696049719e-05
**************************************************
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
Model average F1 score: 0.6119111141998245
Model F1 score variance: 6.499129279609057e-05
**************************************************
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, 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, presort='deprecated',
                       random_state=None, splitter='best')
Model average F1 score: 0.9690338337176365
Model F1 score variance: 2.201127690430312e-05
**************************************************
```

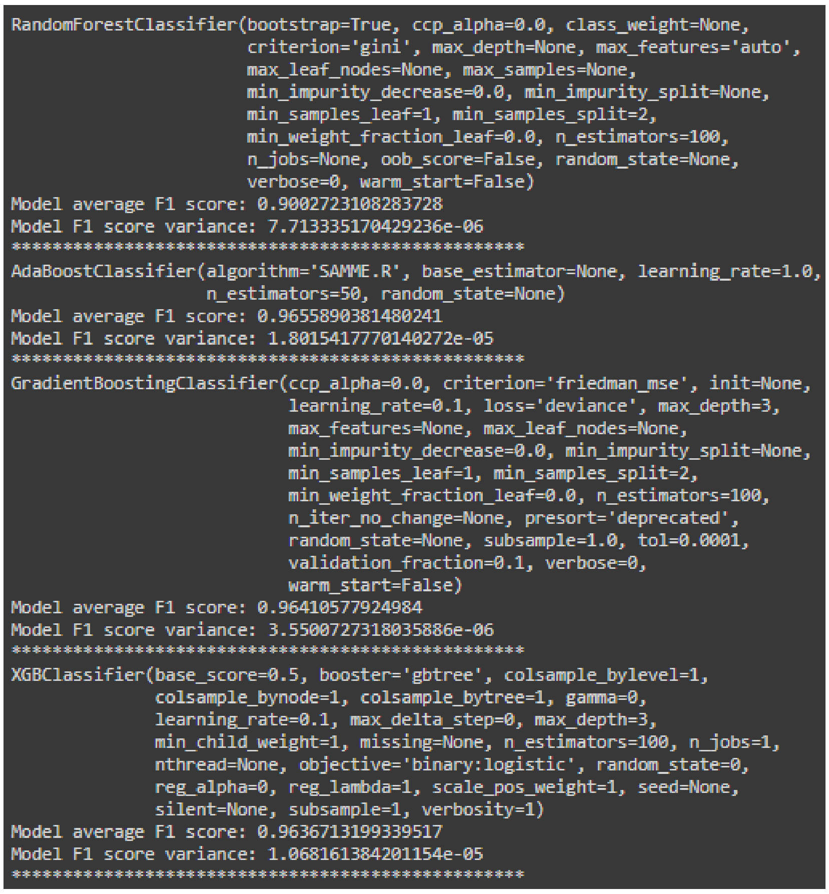
Comparing Ensemble techniques using F1 score metric:

Code:



```
models = [RandomForestClassifier(), AdaBoostClassifier(), GradientBoostingClassifier(), XGBClassifier()]

for model in models:
  scores = cross_val_score(model, x_train, y_train, scoring='f1', n_jobs=-1,cv = 5)
  print(model)
  print(f'Model average F1 score: {scores.mean()}')
  print(f'Model F1 score variance: {scores.var()}')
  print('*'*50)
```

Results:



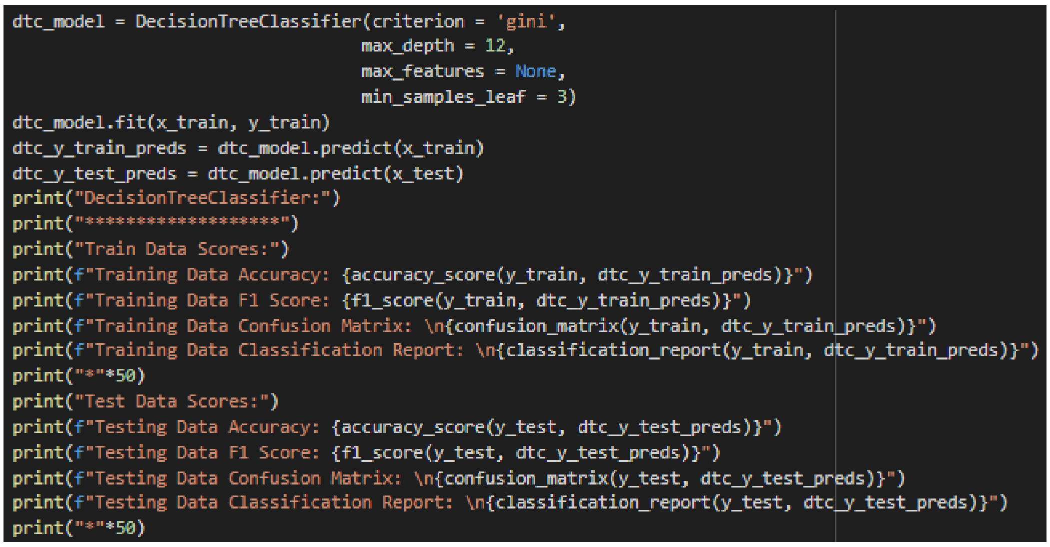
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
Model average F1 score: 0.9002723108283728
Model F1 score variance: 7.713335170429236e-06
**************************************************
AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=1.0,
                   n_estimators=50, random_state=None)
Model average F1 score: 0.9655890381480241
Model F1 score variance: 1.8015417770140272e-05
**************************************************
GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='deviance', max_depth=3,
                           max_features=None, 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=100,
                           n_iter_no_change=None, presort='deprecated',
                           random_state=None, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)
Model average F1 score: 0.96410577924984
Model F1 score variance: 3.5500727318035886e-06
**************************************************
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
Model average F1 score: 0.9636713199339517
Model F1 score variance: 1.068161384201154e-05
**************************************************
```

Observations:

1. DecisionTreeClassifier is giving a good average F1 score and the variance is also low.
2. SVC is also giving a good F1 average score close to DecisionTreeClassifier.
3. AdaBoostClassifier and GradientBoostingClassifier are giving pretty close scores.

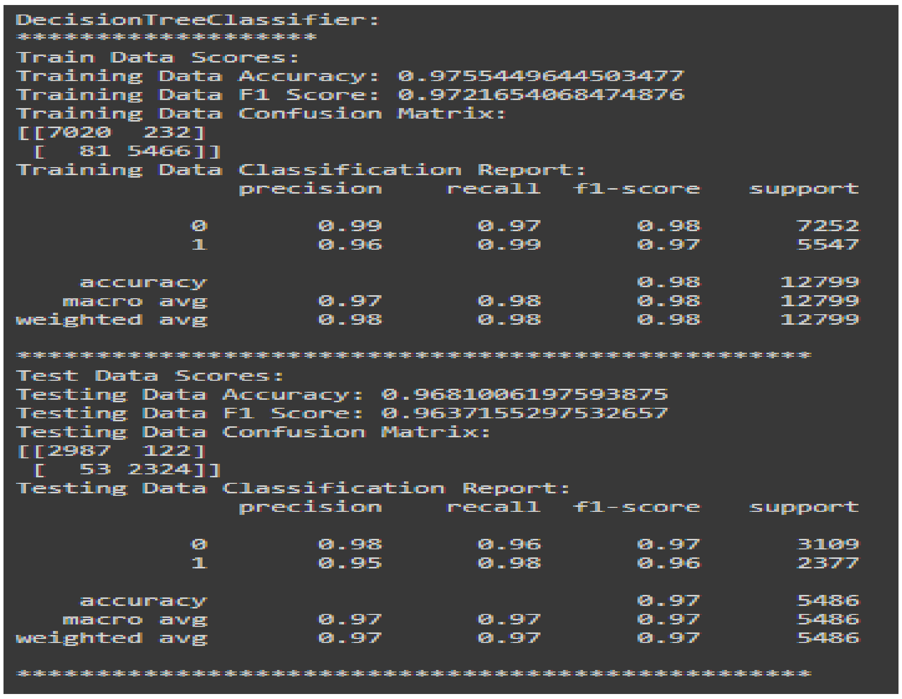
Comparing the top 2 models:

DecisionTreeClassifier:



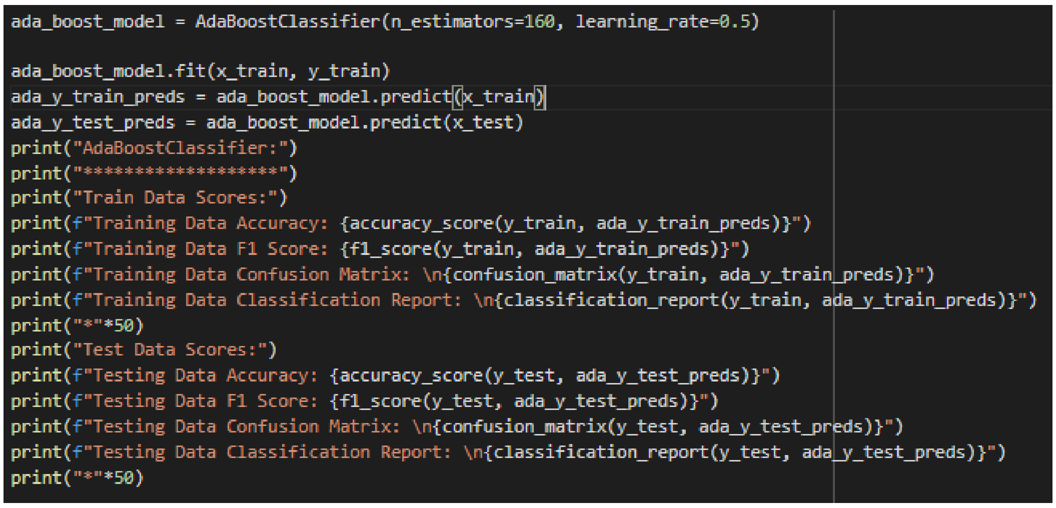
```python
dtc_model = DecisionTreeClassifier(criterion = 'gini',
                                   max_depth = 12,
                                   max_features = None,
                                   min_samples_leaf = 3)
dtc_model.fit(x_train, y_train)
dtc_y_train_preds = dtc_model.predict(x_train)
dtc_y_test_preds = dtc_model.predict(x_test)
print("DecisionTreeClassifier:")
print("*******************")
print("Train Data Scores:")
print(f"Training Data Accuracy: {accuracy_score(y_train, dtc_y_train_preds)}")
print(f"Training Data F1 Score: {f1_score(y_train, dtc_y_train_preds)}")
print(f"Training Data Confusion Matrix: \n{confusion_matrix(y_train, dtc_y_train_preds)}")
print(f"Training Data Classification Report: \n{classification_report(y_train, dtc_y_train_preds)}")
print("*"*50)
print("Test Data Scores:")
print(f"Testing Data Accuracy: {accuracy_score(y_test, dtc_y_test_preds)}")
print(f"Testing Data F1 Score: {f1_score(y_test, dtc_y_test_preds)}")
print(f"Testing Data Confusion Matrix: \n{confusion_matrix(y_test, dtc_y_test_preds)}")
print(f"Testing Data Classification Report: \n{classification_report(y_test, dtc_y_test_preds)}")
print("*"*50)
```

Results:



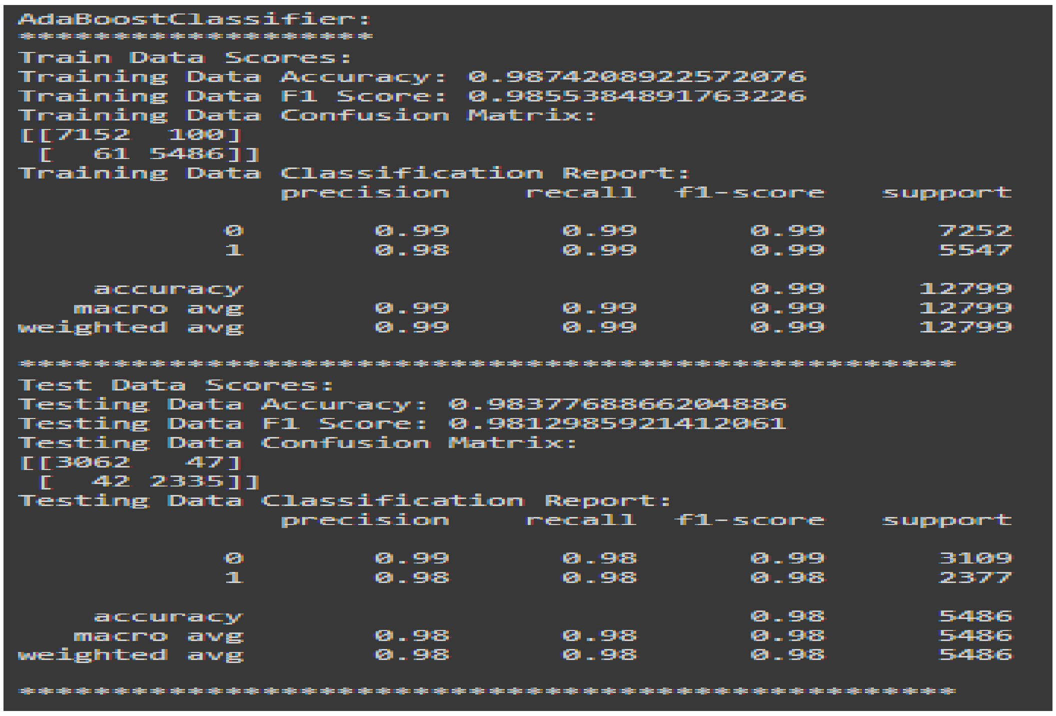
```
DecisionTreeClassifier:
*******************
Train Data Scores:
Training Data Accuracy: 0.9755449644503477
Training Data F1 Score: 0.9721654068474876
Training Data Confusion Matrix:
[[7020  232]
 [  81 5466]]
Training Data Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      7252
           1       0.96      0.99      0.97      5547

    accuracy                           0.98     12799
   macro avg       0.97      0.98      0.98     12799
weighted avg       0.98      0.98      0.98     12799

**************************************************
Test Data Scores:
Testing Data Accuracy: 0.9681006197593875
Testing Data F1 Score: 0.9637155297532657
Testing Data Confusion Matrix:
[[2987  122]
 [  53 2324]]
Testing Data Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.96      0.97      3109
           1       0.95      0.98      0.96      2377

    accuracy                           0.97      5486
   macro avg       0.97      0.97      0.97      5486
weighted avg       0.97      0.97      0.97      5486

**************************************************
```

AdaBoostClassifier:



```
ada_boost_model = AdaBoostClassifier(n_estimators=160, learning_rate=0.5)

ada_boost_model.fit(x_train, y_train)
ada_y_train_preds = ada_boost_model.predict(x_train)
ada_y_test_preds = ada_boost_model.predict(x_test)
print("AdaBoostClassifier:")
print("*******************")
print("Train Data Scores:")
print(f"Training Data Accuracy: {accuracy_score(y_train, ada_y_train_preds)}")
print(f"Training Data F1 Score: {f1_score(y_train, ada_y_train_preds)}")
print(f"Training Data Confusion Matrix: \n{confusion_matrix(y_train, ada_y_train_preds)}")
print(f"Training Data Classification Report: \n{classification_report(y_train, ada_y_train_preds)}")
print("*"*50)
print("Test Data Scores:")
print(f"Testing Data Accuracy: {accuracy_score(y_test, ada_y_test_preds)}")
print(f"Testing Data F1 Score: {f1_score(y_test, ada_y_test_preds)}")
print(f"Testing Data Confusion Matrix: \n{confusion_matrix(y_test, ada_y_test_preds)}")
print(f"Testing Data Classification Report: \n{classification_report(y_test, ada_y_test_preds)}")
print("*"*50)
```

Results:



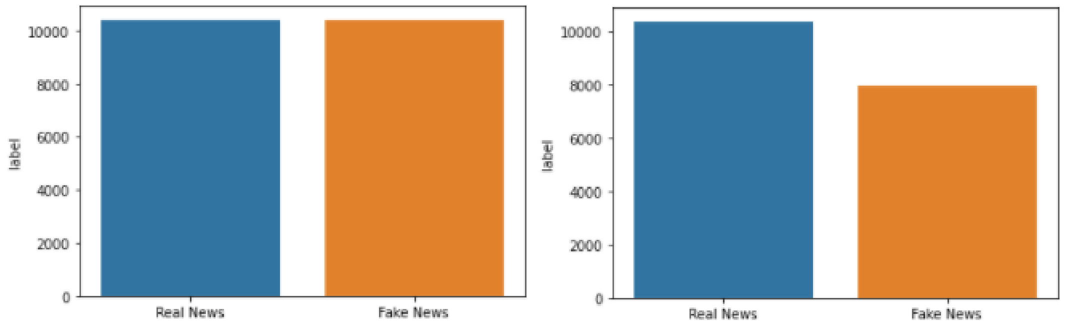
```
AdaBoostClassifier:
*******************
Train Data Scores:
Training Data Accuracy: 0.9874208922572076
Training Data F1 Score: 0.9855384891763226
Training Data Confusion Matrix:
[[7152  100]
 [  61 5486]]
Training Data Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      7252
           1       0.98      0.99      0.99      5547

    accuracy                           0.99     12799
   macro avg       0.99      0.99      0.99     12799
weighted avg       0.99      0.99      0.99     12799

**************************************************
Test Data Scores:
Testing Data Accuracy: 0.9837768866204886
Testing Data F1 Score: 0.9812985921412061
Testing Data Confusion Matrix:
[[3062   47]
 [  42 2335]]
Testing Data Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      3109
           1       0.98      0.98      0.98      2377

    accuracy                           0.98      5486
   macro avg       0.98      0.98      0.98      5486
weighted avg       0.98      0.98      0.98      5486

**************************************************
```

Observations:

- The AdaBoostClassifier is performing better with the tuned Hyperparameters based on all the key evaluation metrics.
- For AdaBoostClassifier, the F1 score for the train data is 0.985 and for the F1 score for the test data is 0.981.
- The AdaBoostClassifier shows a better True Positive and True Negative values than DecisionTreeClassifier.

- Key Metrics for success in solving problem under consideration
    1. F1 Score is used as the primary key metric for evaluation.
    2. Accuracy, confusion matrix, ROC AUC scores are used as secondary metrics.
- Visualizations

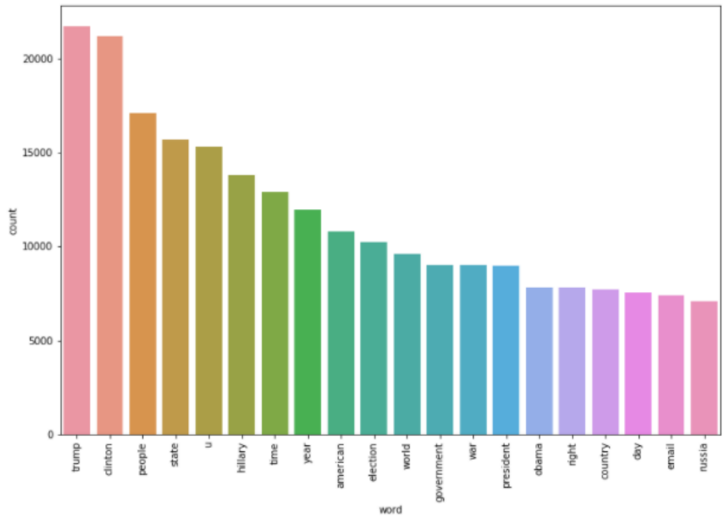
1. Target variable frequency with missing data(left) vs Target variable frequency without missing data(right):



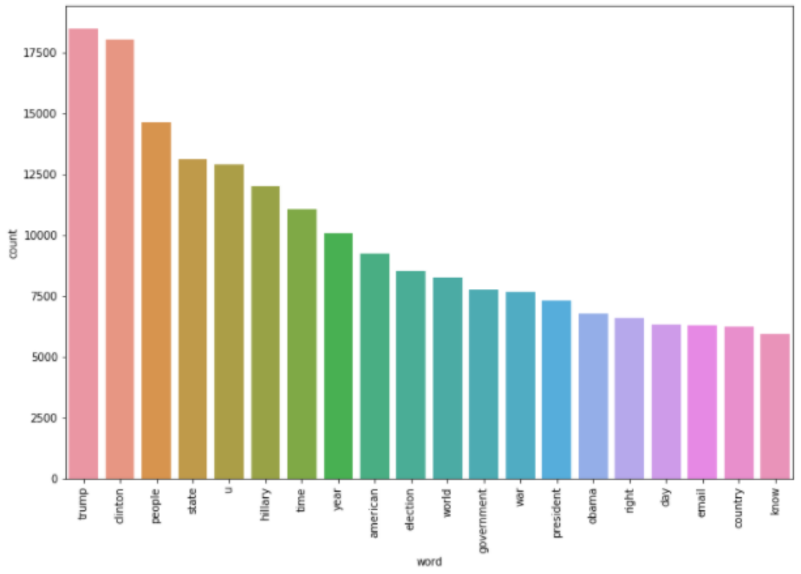
Observations:

- Most of the missing data is in the 'Fake News' samples.

2. Top words in Fake news samples :
    a. contains missing values:

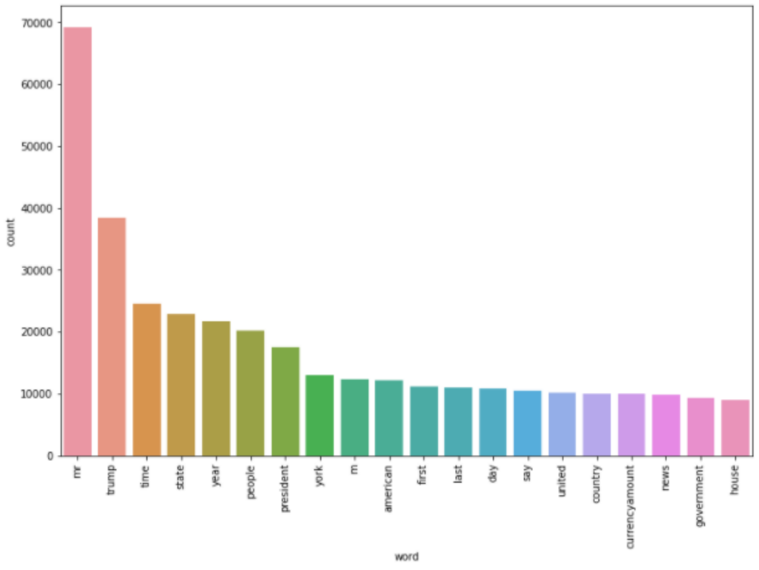


b. Does not contain missing values:

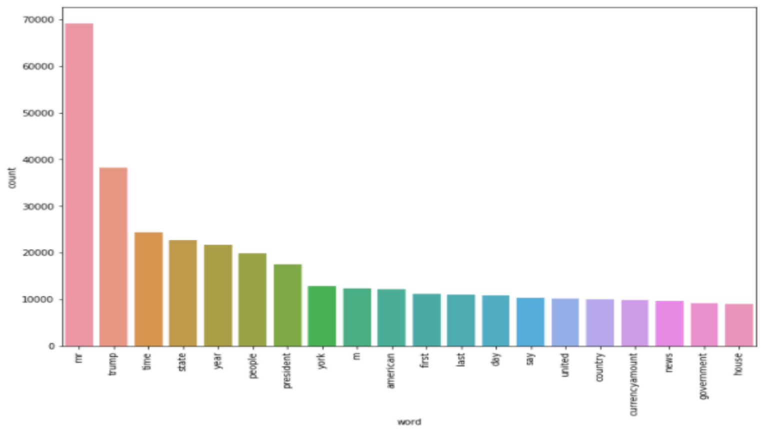


3. Top words in Real news samples:

a. Contains missing data:



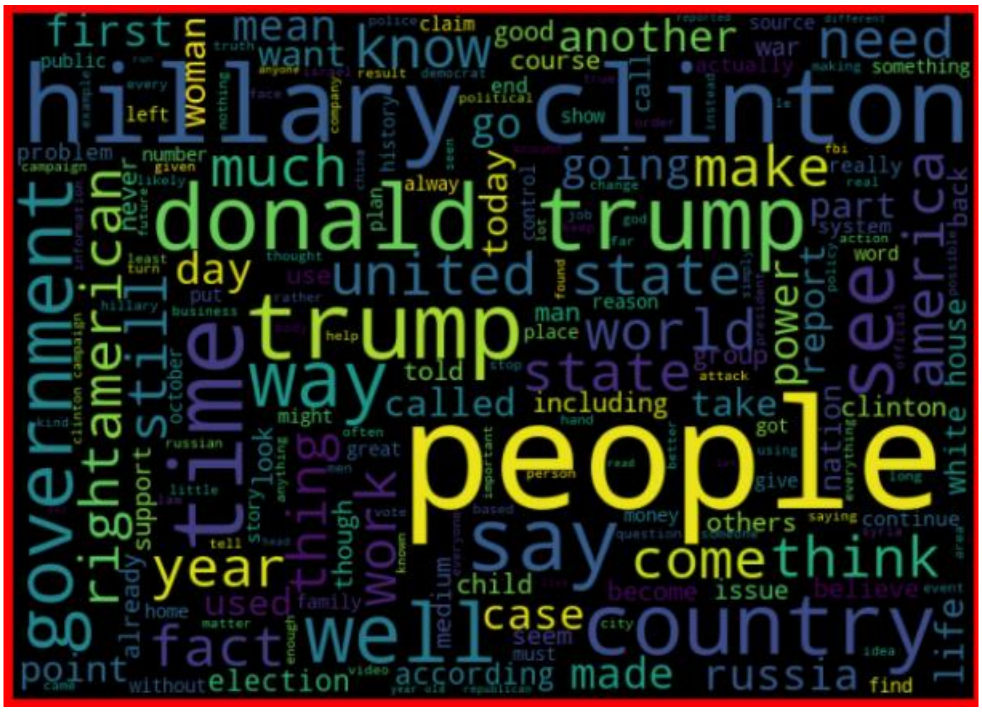
b. Does not contain missing data:



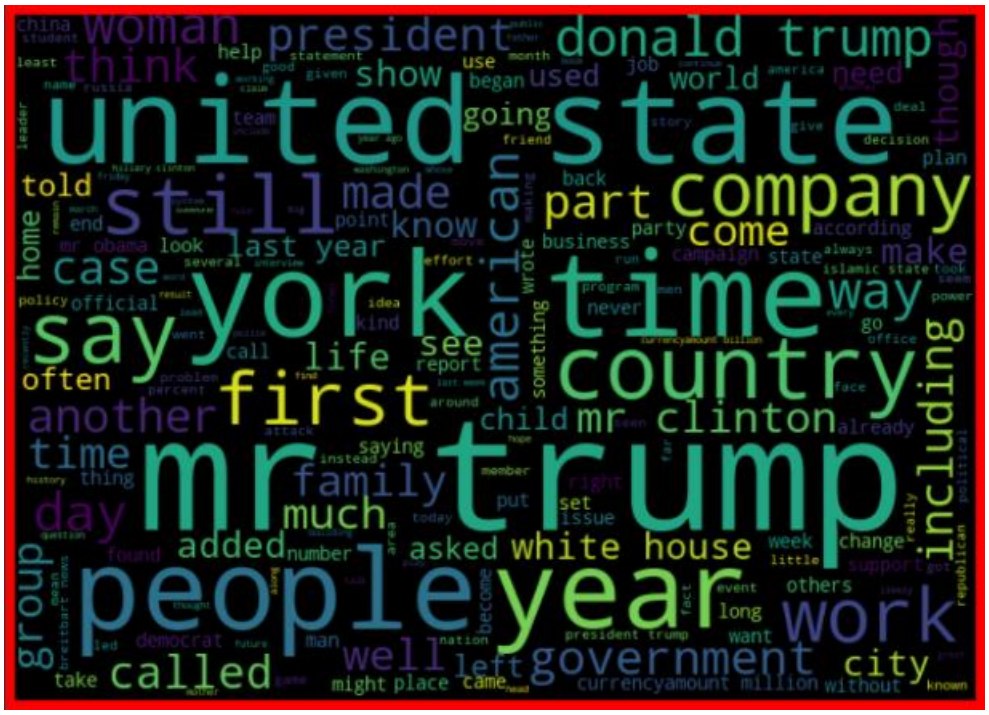
Observations:

- The above plots show that the top words in Real news with and without missing data are all the same.
- The tops words in Fake news are slightly different between data with and without missing data.

4. Word Cloud:



Fake News Word Cloud



Real News Word Cloud

Observations:

1. There are some distinctions between the words in fake news vs real news.
2. Some of the words are common between fake and real news.
3. It would make sense to use monogram, bigram and trigram while using TFIDF.

- Interpretation of the Results
  1. Although there are slightly different words between Real news and Fake news, most of the monograms are common between them. The model could get confused between classifying the records as Fake and Real. Hence using bi-grams and tri-grams along with mono-grams in the TFIDF Vectorizer is a good idea.
  2. Ensemble model works better in this problem.

CONCLUSION

- Key Findings and Conclusions of the Study
    1. Most of the missing values were in the fake news samples.
    2. There are lot of similar words(monogram) in Fake news and Real news samples. However, bi-grams and tri-grams show some difference.
    3. AdaBoostClassifier is able to perform well for the data used.
- Learning Outcomes of the Study in respect of Data Science

  In problems like these the monograms may not be very helpful. Using n-grams are very useful since Fake news are written using the commonly used words from Real news. It is important to understand the context of the actual news to classify if it is fake or real.

- Limitations of this work and Scope for Future Work

  The final model is working well with the data at hand. However, using more sample data the model can we trained and improved for future predictions and reduce the risk of model drift.