# FAKE NEWS DETECTION USING MACHINE LEARNING


## TABLE OF CONTENTS



## ABOUT

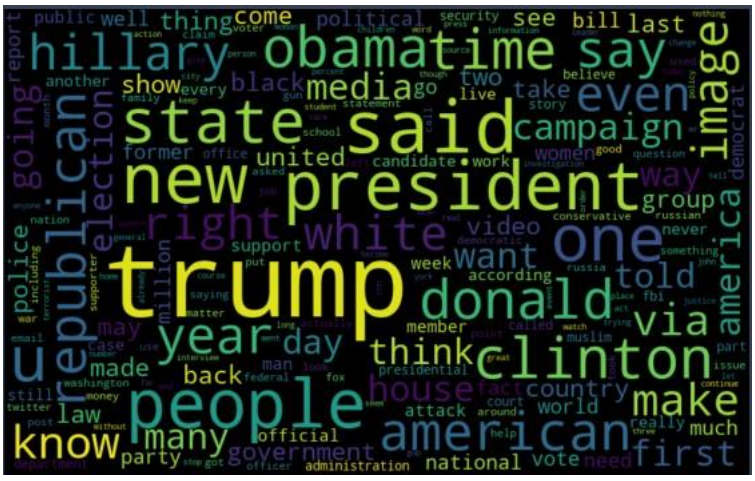
Politics is an extremely common topics in most household and majority depend on news to be updated. Fake news might cause problems or mislead people into believing things that might not have happened. The proposed approach is to use machine learning to detect fake news. Using vectorisation of the news title and then analysing the tokens of words with our dataset. The dataset we are using is a predefined curated list of news with their property of being a fake news or not. Our goal is to develop a model that classifies a given article as either true or fake.

## INTRODUCTION

### a. What are fake news?

In this project detection of fake news is based only on textual information by applying traditional machine learning techniques.

In order to work on fake news detection, it is important to understand what is fake news and how they are characterized. The first is characterization or what is fake news and the second is detection. In order to build detection models, it is need to start by characterization, indeed, it is need to understand what is fake news before trying to detect them.



# METHODOLOGY

| Verifying data collection | Data Mining | Modeling | Effectiveness |
|---|---|---|---|
| •Read data from csv files<br>•Two files: Fake & True News | •Tokenization<br>•Removing stop words<br>•Stemming<br>•Splitting of data set | •Selecting supervised algorithms<br>•Testing the algorithms and finding accuracy | •Verifying the effectiveness of the algorithms using accuracy score<br>•Also comparing all algorithms using data visualizations |

# ALGORITHMS

## a. Naïve Bayes Classifier

NAIVE BAYES CLASSIFIER

```
In [35]: dct = dict()
         from sklearn.naive_bayes import MultinomialNB
         NB_classifier = MultinomialNB()
         pipe = Pipeline([('vect', CountVectorizer()),('tfidf', TfidfTransformer()),('model', NB_classifier)])
         model = pipe.fit(X_train, y_train)
         prediction = model.predict(X_test)
         print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
         dct['Naive Bayes'] = round(accuracy_score(y_test, prediction)*100,2)

         accuracy: 95.26%
```

## b. Logistic Regression

LOGISTIC REGRESSION

```
In [38]: from sklearn.linear_model import LogisticRegression
         pipe = Pipeline([('vect', CountVectorizer()),('tfidf', TfidfTransformer()),('model', LogisticRegression())])
         model = pipe.fit(x_tr, y_tr)
         prediction = model.predict(x_te)
         print("accuracy: {}%".format(round(accuracy_score(y_te, prediction)*100,2)))
         dct['Logistic Regression'] = round(accuracy_score(y_te, prediction)*100,2)

         accuracy: 98.82%
```

## c. Decision Tree Classifier

```
In [40]: from sklearn.tree import DecisionTreeClassifier
         pipe = Pipeline([('vect', CountVectorizer()),('tfidf', TfidfTransformer()),('model', DecisionTreeClassifier
         model = pipe.fit(x_tr, y_tr)
         prediction = model.predict(x_te)
         print("accuracy: {}%".format(round(accuracy_score(y_te, prediction)*100,2)))
         dct['Decision Tree'] = round(accuracy_score(y_te, prediction)*100,2)

         accuracy: 99.68%
```

d. **Random Forest Classifier**

RANDOM FOREST CLASSIFIER

```
In [42]: from sklearn.ensemble import RandomForestClassifier
         pipe = Pipeline([('vect', CountVectorizer()),('tfidf', TfidfTransformer()),('model', RandomForestClassifier
         model = pipe.fit(x_tr, y_tr)
         prediction = model.predict(x_te)
         print("accuracy: {}%".format(round(accuracy_score(y_te, prediction)*100,2)))
         dct['Random Forest'] = round(accuracy_score(y_te, prediction)*100,2)

         accuracy: 99.08%
```

**e. Support Vector Machine (SVM)**

SVM CLASSIFIER

```
In [44]: from sklearn import svm
         clf = svm.SVC(kernel='linear')
         pipe = Pipeline([('vect', CountVectorizer()),('tfidf', TfidfTransformer()),('model', clf)])
         model = pipe.fit(x_tr, y_tr)
         prediction = model.predict(x_te)
         print("accuracy: {}%".format(round(accuracy_score(y_te, prediction)*100,2)))
         dct['SVM'] = round(accuracy_score(y_te, prediction)*100,2)

         accuracy: 99.6%
```

# DATASETS

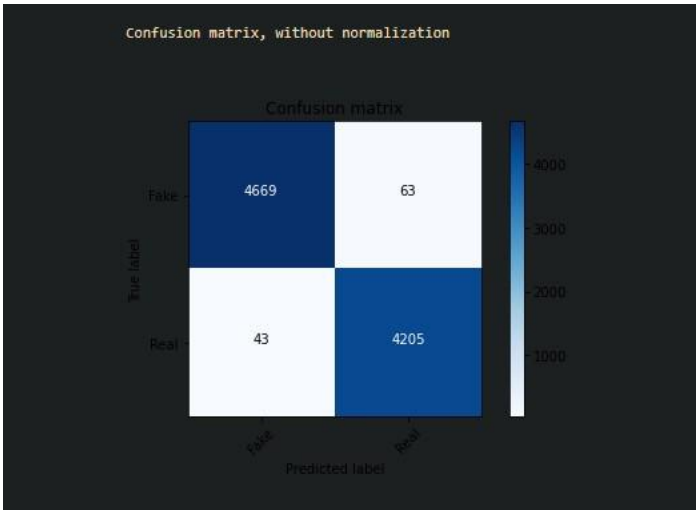
The data includes both fake and truthful news articles from multiple domains. The truthful news articles published contain true description of real-world events, while the fake news websites contain claims that are not aligned with facts. I have used two datasets in this study referred to as True and Fake. These 2 final datasets are combined into one large final dataset referred as data.

# PERFORMANCE METRICS

To evaluate the performance of the algorithms, I used confusion matrix.
Confusion matrix is a tabular representation of a classification model performance on the test set, which consists of four parameters: true positive, false positive, true negative, and false negative.

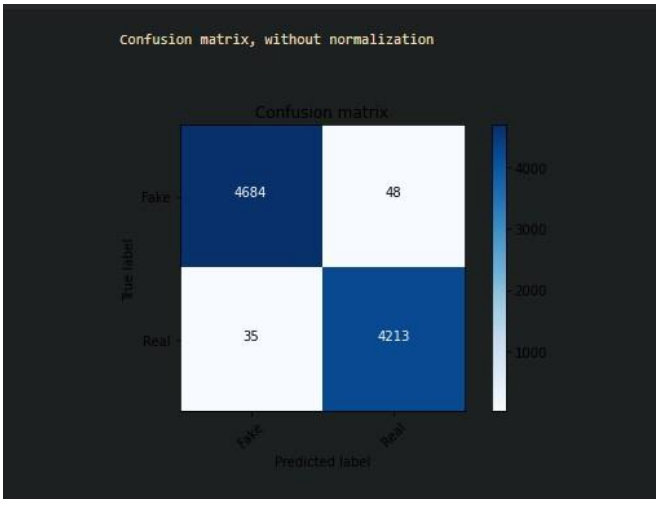
## A. NAÏVE BAYES CLASSIFIER

## B. LOGISTIC REGRESSION




## C. DECISION TREE CLASSIFIER


## D. RANDOM FOREST CLASSIFIER



### E. SVM


## CONCLUSION

Highest Accuracy Model: Decision Tree Classifier
Lowest Accuracy Model: Naïve Bayes Classifier
Therefore, we can opt for the Decision Tree Classifier and the results of this model can be used to detect fake news.

| CLASSIFIER | ACCURACY |
|---|---|
| **Naïve Bayes** | **94.91%** |
| **Support Vector Machine (SVM)** | **99.52%** |
| **Random Forest** | **99.22%** |
| **Logistic Regression** | **98.91%** |
| **Decision Tree** | **99.91%** |