FAKE NEWS DETECTION

Using Machine Learning Algorithms

INTRODUCTION

- Machine learning (ML) is an artificial intelligence (AI) application that allows systems to automatically learn and improve based on their experiences without the need for explicit programming in advance. It is the study of computer algorithms that can self-improve as a result of experience and data collection and analysis.
- Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or make judgments.
- Computer vision, medicine, email filtering, and speech recognition are just a few of the fields where machine learning algorithms are being used to solve problems where traditional algorithms would be difficult or impossible to create.We start the learning process.
- Computational learning theory is a branch of theoretical computer science that investigates the computational analysis and performance of machine learning algorithms.
- Given the scarcity of training sets and the uncertainty surrounding the future, learning theory does not always provide guarantees about algorithm performance. Instead, in computer science applications, probabilistic bounds on performance are fairly common. One method for estimating generalisation error is the bias–variance decomposition.
- In order to achieve the best generalisation performance, the hypothesis's complexity should be proportional to the complexity of the function underlying the data. Likewise, if the hypothesis is simpler than the function, the model will have under-fitted the observed data. If the complexity of the problem is increased, the training error decreases.

EXPERIMENT DESCRIPTION (Without Feature Selection)

Step 1: Import all the sklearn libraries like (sklearn.model_selection import train_test_split, sklearn.metrics import accuracy_score) and also Numpy and pandas.

Step 2: Importing Fake news and True news csv files using pd.read_csv function.

Step 3: We are implementing a class column in both Fake news and True news datasets where we are representation Fake news as **0** and True news as **1**.

Step 4: We created a variable **MARGE** and we are using a function concat where we are merging Fake news and True news datasets.

Step 5: By using the function column we are getting an output showing what are the columns present in the dataset and their respective data types.

Step 6: By using **drop** function we dropping all the unnecessary columns from the dataset.

Step 7: Using **isnull** we are finding out how many null values are there in the remaining dataset.

Step 8 By using **sample** function we are shuffling dataset.

Step 9: We created a function wordopt which convert the text in lowercase, remove the extra space, special chr., ulr and links and applying it in text column.

Step 10: We are taking 2 variables X & Y and assigning them with Text Column & Class Column respectively.

Step 11: We are implementing vectorization and splitting the datasets into testing set and training set.

Step 12: Now we will be implementing classifiers on the testing dataset like Decision Trees, SVM, ANN, KNN, Naive Bayes.

Decision Trees:

1. Importing libraries (sklearn.tree import DecisionTreeClassifier, sklearn.metrics import classification_report, confusion_matrix)
2. Assigning DT as an object of decision tree classifier.
3. Using fit function we are fitting testing set and training set.
4. Using predict function we are performing a prediction for each testing set.
5. Using score function we are finding out the accuracy of the model.
6. The classification_report function is helping us in finding the precision, recall, F1 score & accuracy.
7. By using confusion matrix function we are finding out the confusion matrix.
8. Lastly, by using metrics library we are finding out the error.

Naive Bayes:

1. Importing libraries (sklearn.naive_bayes import BernoulliNB)
2. Assigning BNB as an object of Bernoulli naive bayes.
3. Using fit function we are fitting testing set and training set.
4. Using predict function we are performing a prediction for each testing set.
5. Using score function we are finding out the accuracy of the model.
6. The classification_report function is helping us in finding the precision, recall, F1 score & accuracy.
7. By using confusion matrix function we are finding out the confusion matrix.
8. Lastly, by using metrics library we are finding out the error.

KNN:

1. Importing libraries (sklearn.neighbors import KNeighborsClassifier)
2. Assigning ng as an object of KNeighborsClassifier.
3. Using fit function we are fitting testing set and training set.
4. Using predict function we are performing a prediction for each testing set.
5. Using score function we are finding out the accuracy of the model.
6. The classification_report function is helping us in finding the precision, recall, F1 score & accuracy.
7. By using confusion matrix function we are finding out the confusion matrix.
8. Lastly, by using metrics library we are finding out the error.

SVM:

1. Importing libraries (sklearn.svm import LinearSVC)
2. Assigning LS as an object of LinearSVC.
3. Using fit function we are fitting testing set and training set.
4. Using predict function we are performing a prediction for each testing set.
5. Using score function we are finding out the accuracy of the model.
6. The classification_report function is helping us in finding the precision, recall, F1 score & accuracy.
7. By using confusion matrix function we are finding out the confusion matrix.
8. Lastly, by using metrics library we are finding out the error.

ANN:

1. Importing libraries (sklearn.neural_network import MLPClassifier)
2. Assigning ANN as an object of MLPClassifier.
3. Using fit function we are fitting testing set and training set.
4. Using predict function we are performing a prediction for each testing set.
5. Using score function we are finding out the accuracy of the model.
6. The classification_report function is helping us in finding the precision, recall, F1 score & accuracy.
7. By using confusion matrix function we are finding out the confusion matrix.
8. Lastly, by using metrics library we are finding out the error.

EXPERIMENT DESCRIPTION (With Feature Selection)

Step 1: Import all the sklearn libraries like (sklearn.model_selection import train_test_split, sklearn.metrics import accuracy_score) and also Numpy and pandas.

Step 2: Importing Fake news and True news csv files using pd.read_csv function.

Step 3: We are implementing a label column in both Fake news and True news datasets where we are representation Fake news as **0** and True news as **1**.

Step 4: Importing nltk library we are downloading a keyword named stopwords.

Step 5: By using concat function we are merging both Fake and True news dataset and saving it in a variable new_dataset.

Step 6: By using sample function we are shuffling the whole dataset.

Step 7: Using **isnull** we are finding out how many null values are there in the remaining dataset.

Step 8: We are creating a column content and merging two columns which can reduce the computational time and power of the model. (This process is known as Feature Engineering)

Step 9: This process is known as steaming. Steaming is a process of reducing a word to its root word.

Step 10: We are applying this steaming function in the column content.

Step 11: We are taking 2 variables X & Y and assigning them with content Column & label Column respectively.

Step 12: We are implementing vectorization and splitting the datasets into testing set and training set.

Step 13: Now we will be implementing classifiers on the testing dataset like Decision Trees, SVM, ANN, KNN, Naive Bayes.

SNAPS OF CODE (WITHOUT FEATURE SELECTION)

In [74]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn import metrics
import re
import string
from sklearn.linear_model import LogisticRegression
```

In [2]:
```
fake_news=pd.read_csv("Fake.csv")
true_news=pd.read_csv("True.csv")
```

In [5]:
```
fake_news['class'] = 0
true_news['class'] = 1
```

In [6]:
```
fake_news.shape,true_news.sha
```

Out[6]: ((23481, 5), (21417, 5))

[13]:
```
marge = pd.concat([fake_news, true_news], axis =0 )
marge.head(10)
```

Out[13]:

	title	text	subject	date	class
0	Donald Trump Sends Out Embarrassing New Year'...	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017	0
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017	0
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	December 30, 2017	0
3	Trump Is So Obsessed He Even Has Obama's Name...	On Christmas day, Donald Trump announced that ...	News	December 29, 2017	0
4	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	News	December 25, 2017	0
5	Racist Alabama Cops Brutalize Black Boy While...	The number of cases of cops brutalizing and ki...	News	December 25, 2017	0
6	Fresh Off The Golf Course, Trump Lashes Out A...	Donald Trump spent a good portion of his day a...	News	December 23, 2017	0
7	Trump Said Some INSANELY Racist Stuff Inside ...	In the wake of yet another court decision that...	News	December 23, 2017	0
8	Former CIA Director Slams Trump Over UN Bully...	Many people have raised the alarm regarding th...	News	December 22, 2017	0
9	WATCH: Brand-New Pro-Trump Ad Features So Muc...	Just when you might have thought we d get a br...	News	December 21, 2017	0

In [14]:
```
marge.columns
```
Out[14]:
```
Index(['title', 'text', 'subject', 'date', 'class'], dtype='object')
```

In [15]:
```
df = marge.drop(["title","subject","date"],axis=1)
```

In [16]:
```
df.isnull().sum()
```
Out[16]:
```
text     0
class    0
dtype: int64
```

In [17]:
```
df = df.sample(frac = 1)          # Never Change this
```

In [18]:
```
df.head()
```

In [24]:
```
x = df["text"]
y = df["class"]
```

In [25]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30)
```

In [26]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [27]:
```
vectorization = TfidfVectorizer()
xv_train = vectorization.fit_transform(x_train)
xv_test = vectorization.transform(x_test)
```

Creating a function to convert the text in lowercase, remove the extra space, special chr., ulr and links.

n [83]:
```
def wordopt(text):
    text = text.lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub("\\W"," ",text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    text = np.size(data[0::,1].astype(np.float))
    return text
```

n [23]:
```
df["text"] = df["text"].apply(wordopt)
```

In [114]:
```
from sklearn.naive_bayes import BernoulliNB
```

In [115]:
```
BNB = BernoulliNB()
```

In [116]:
```
BNB.fit(xv_train, y_train)
```

Out[116]:
```
BernoulliNB()
```

In [117]:
```
pred_bnb = BNB.predict(xv_test)
```

In [118]:
```
BNB.score(xv_test,y_test)
```

Out[118]:
```
0.942364824717766
```

In [119]:
```
print(classification_report(y_test, pred_bnb))     #Precision, Recall, F-Measure, Accuracy
```

```
              precision    recall  f1-score   support

           0       0.96      0.93      0.94      7083
           1       0.92      0.96      0.94      6381

    accuracy                           0.94     13464
   macro avg       0.94      0.94      0.94     13464
weighted avg       0.94      0.94      0.94     13464
```

In [106]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [107]:
```
DT = DecisionTreeClassifier()
DT.fit(xv_train, y_train)
```

Out[107]:
```
DecisionTreeClassifier()
```

In [108]:
```
pred_dt = DT.predict(xv_test)
```

In [109]:
```
DT.score(xv_test, y_test)
```

Out[109]:
```
0.9956922162804516
```

In [110]:
```
print(classification_report(y_test, pred_dt))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      7083
           1       1.00      0.99      1.00      6381

    accuracy                           1.00     13464
   macro avg       1.00      1.00      1.00     13464
weighted avg       1.00      1.00      1.00     13464
```

In [129]:
```
from sklearn.svm import LinearSVC
```

In [130]:
```
ls = LinearSVC()
```

In [131]:
```
ls.fit(xv_test, y_test)
```

Out[131]:
```
LinearSVC()
```

In [132]:
```
pred_ls = ls.predict(xv_test)
```

In [133]:
```
ls.score(xv_test,y_test)
```

Out[133]:
```
0.9994800950683304
```

In [134]:
```
print(classification_report(y_test, pred_ls))     # Precision , Recall , F-measure , Accuracy
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      7083
           1       1.00      1.00      1.00      6381

    accuracy                           1.00     13464
   macro avg       1.00      1.00      1.00     13464
weighted avg       1.00      1.00      1.00     13464
```

In [122]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [123]:
```
ng = KNeighborsClassifier()
ng.fit(xv_test, y_test)
```

Out[123]:
```
KNeighborsClassifier()
```

In [124]:
```
pred_knn = ng.predict(xv_test)
```

In [125]:
```
ng.score(xv_test,y_test)
```

Out[125]:
```
0.672979797979798
```

In [126]:
```
print(classification_report(y_test, pred_knn))     # Precision , Recall , F-measure , Accuracy
```

```
              precision    recall  f1-score   support

           0       0.62      0.99      0.76      7083
           1       0.96      0.32      0.48      6381

    accuracy                           0.67     13464
   macro avg       0.79      0.66      0.62     13464
weighted avg       0.78      0.67      0.63     13464
```

In [137]:
```python
from sklearn.neural_network import MLPClassifier
```

In [138]:
```python
ann = MLPClassifier()
```

In [139]:
```python
ann.fit(xv_test, y_test)
```

Out[139]:
```
MLPClassifier()
```

In [140]:
```python
pred_ann = ann.predict(xv_test)
```

In [141]:
```python
ann.score(xv_test,y_test)
```

Out[141]:
```
1.0
```

In [142]:
```python
print(classification_report(y_test, pred_ann))      # Precision , Recall , F-measure , Accuracy
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      7083
           1       1.00      1.00      1.00      6381

    accuracy                           1.00     13464
   macro avg       1.00      1.00      1.00     13464
weighted avg       1.00      1.00      1.00     13464
```

CODE USING FEATURE SELECTION

In [1]:
```
import numpy as np
import pandas as pd
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
```

In [2]:
```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\KIIT\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[2]: True

In [3]:
```
# printing the stopwords in English
print(stopwords.words('english'))
```

In [4]:
```
fake_news=pd.read_csv("Fake.csv")
true_news=pd.read_csv("True.csv")
```

In [5]:
```
fake_news['label'] = 0
true_news['label'] = 1
```

In [6]:
```
print(fake_news.shape,true_news.shape)
```

```
(23481, 5) (21417, 5)
```

In [7]:
```
news_dataset = pd.concat([fake_news, true_news], axis =0 )
news_dataset.head()
```

In [8]:
```
news_dataset = news_dataset.sample(len(news_dataset))
news_dataset.head()
```

Out[8]:

	title	
15580	Somalia's Islamist insurgency executes four me...	(Reuters) - Somalia s
16218	FATHER OF SON KILLED BY ILLEGAL Speaks Up On D...	
21164	Rome's 5-Star mayor launches bid to save ailin...	ROME (Reuters) - Rome s
13235	MEDIA IMMEDIATELY REPORTS Alleged Killer Of Im...	There is no questi
15255	Syria's Eastern Ghouta faces 'complete catastr...	GENEVA (Reuters) - Th

In [9]:
```
news_dataset.shape
```

Out[9]:
```
(44898, 5)
```

In [10]:
```
# counting the number of missing values in the dataset
news_dataset.isnull().sum()
```

Out[10]:
```
title      0
text       0
subject    0
date       0
label      0
dtype: int64
```

In [11]:
```
# merging the subject name and news title
news_dataset['content'] = news_dataset['subject']+' '+news_dataset['title']
```

In [12]:
```
print(news_dataset['content'])
```

```
15580    worldnews Somalia's Islamist insurgency execut...
16218    Government News FATHER OF SON KILLED BY ILLEGA...
21164    worldnews Rome's 5-Star mayor launches bid to ...
13235    politics MEDIA IMMEDIATELY REPORTS Alleged Kil...
15255    worldnews Syria's Eastern Ghouta faces 'comple...
                                ...
5309     News  Loser Donald Trump Ordered To Pay $300,0...
23268    Middle-east Episode #152 – SUNDAY WIRE: 'From ...
3791     News  WATCH: NSA Chief Drops HUGE Bombshell Ab...
15855    politics BREAKING: IRAN THROWS DOWN ULTIMATUM:...
73       politicsNews White House aide sees temporary f...
Name: content, Length: 44898, dtype: object
```

Stemming:

actor, actress, acting --> act

In [13]:
```
port_stem = PorterStemmer()
```

In [14]:
```
def stemming(content):
    stemmed_content = re.sub('[^a-zA-Z]',' ',content)
    stemmed_content = stemmed_content.lower()
    stemmed_content = stemmed_content.split()
    stemmed_content = [port_stem.stem(word) for word in stemmed_content if not word in stopwords.words('english')]
    stemmed_content = ' '.join(stemmed_content)
    return stemmed_content
```

In [15]:
```
news_dataset['content'] = news_dataset['content'].apply(stemming)
print(news_dataset['content'])
```

```
15580    worldnew somalia islamist insurg execut four m...
16218    govern news father son kill illeg speak democr...
21164    worldnew rome star mayor launch bid save ail c...
13235    polit media immedi report alleg killer imam as...
15255    worldnew syria eastern ghouta face complet cat...
                                   ...
5309     news loser donald trump order pay judg florida...
23268    middl east episod sunday wire ground zero syri...
3791     news watch nsa chief drop huge bombshel involv...
15855          polit break iran throw ultimatum move barri
73       politicsnew white hous aid see temporari fund ...
Name: content, Length: 44898, dtype: object
```

In [25]:
```
#separating the data and label
x = news_dataset['content'].values
y = news_dataset['label'].values
```

In [26]:
```
print(x)
```

```
['worldnew somalia islamist insurg execut four men accus spi'
 'govern news father son kill illeg speak democrat ignor like ignor video'
 'worldnew rome star mayor launch bid save ail citi transport firm' ...
 'news watch nsa chief drop huge bombshel involv wikileak russia elect'
 'polit break iran throw ultimatum move barri'
 'politicsnew white hous aid see temporari fund fix children health program']
```

In [29]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30)
```

In [30]:
```
x_train
```

Out[30]:
```
array(['worldnew chines envoy exchang view korean peninsula issu north korea',
       'politicsnew cuba want sign accord u obama exit offici',
       'left news pro trump chicagoan hit back use real fake sculptur front trump tower monument cnn',
       ..., 'politicsnew trump shift away complet muslim ban penc',
       'news trump biggest kkk fan back love anti muslim tweet morn',
       'politicsnew elect bid ohio senat keep safe distanc trump'],
      dtype=object)
```

In [31]:
```
vectorization = TfidfVectorizer()
xv_train = vectorization.fit_transform(x_train)
xv_test = vectorization.transform(x_test)
```

Decision Tree Classification

In [40]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [41]:
```
DT = DecisionTreeClassifier()
DT.fit(xv_train, y_train)
```

Out[41]:
```
DecisionTreeClassifier()
```

In [42]:
```
pred_dt = DT.predict(xv_test)
```

In [43]:
```
DT.score(xv_test, y_test)
```

Out[43]:
```
1.0
```

In [44]:
```
print(classification_report(y_test, pred_dt))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      7029
           1       1.00      1.00      1.00      6441

    accuracy                           1.00     13470
   macro avg       1.00      1.00      1.00     13470
weighted avg       1.00      1.00      1.00     13470
```

Naive Bayes using Bernoulli naive bayes

In [48]:
```
from sklearn.naive_bayes import BernoulliNB
```

In [49]:
```
BNB = BernoulliNB()
```

In [50]:
```
BNB.fit(xv_train, y_train)
```

Out[50]:
```
BernoulliNB()
```

In [51]:
```
pred_bnb = BNB.predict(xv_test)
```

In [52]:
```
BNB.score(xv_test,y_test)
```

Out[52]:
```
0.9962880475129918
```

In [54]:
```
print(classification_report(y_test, pred_bnb))    # Precision , Recall , F-measure , Accuracy
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      1.00      7029
           1       0.99      1.00      1.00      6441

    accuracy                           1.00     13470
   macro avg       1.00      1.00      1.00     13470
weighted avg       1.00      1.00      1.00     13470
```

In [55]:
```
print(confusion_matrix(y_test, pred_bnb))                #confusion Matrix
```

```
[[6983   46]
 [   4 6437]]
```

K nearest neighbor

In [57]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [58]:
```
ng = KNeighborsClassifier()
ng.fit(xv_test, y_test)
```

Out[58]:
```
KNeighborsClassifier()
```

In [60]:
```
pred_knn = ng.predict(xv_test)
```

In [61]:
```
ng.score(xv_test,y_test)
```

Out[61]:
```
0.9468448403860431
```

In [62]:
```
print(classification_report(y_test, pred_knn))      # Precision , Recall , F-measur
```

```
              precision    recall  f1-score   support

           0       0.97      0.93      0.95      7029
           1       0.93      0.97      0.95      6441

    accuracy                           0.95     13470
   macro avg       0.95      0.95      0.95     13470
weighted avg       0.95      0.95      0.95     13470
```

ANN (Artificial Neural Network)

In [74]:
```
from sklearn.neural_network import MLPClassifier
```

In [75]:
```
ann = MLPClassifier()
```

In [76]:
```
ann.fit(xv_test, y_test)
```

Out[76]:
```
MLPClassifier()
```

In [77]:
```
pred_ann = ann.predict(xv_test)
```

In [78]:
```
ann.score(xv_test,y_test)
```

Out[78]:
```
1.0
```

In [79]:
```
print(classification_report(y_test, pred_ann))      # Precision , Recall , F-measure , A
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      7029
           1       1.00      1.00      1.00      6441

    accuracy                           1.00     13470
   macro avg       1.00      1.00      1.00     13470
weighted avg       1.00      1.00      1.00     13470
```

Support Vector Machine

In [66]:
```
from sklearn.svm import LinearSVC
```

In [67]:
```
ls = LinearSVC()
```

In [68]:
```
ls.fit(xv_test, y_test)
```

Out[68]:
```
LinearSVC()
```

In [69]:
```
pred_ls = ls.predict(xv_test)
```

In [70]:
```
ls.score(xv_test,y_test)
```

Out[70]:
```
1.0
```

In [71]:
```
print(classification_report(y_test, pred_ls))      # Precision , Recall , F-measure , Accuracy
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      7029
           1       1.00      1.00      1.00      6441

    accuracy                           1.00     13470
   macro avg       1.00      1.00      1.00     13470
weighted avg       1.00      1.00      1.00     13470
```

RESULT ANALYSIS (without feature selection)

Decision Tree						
	Precision	Recall	F1-Score	Accuracy	Confusion Matrix	Error (root mean)
Fake News Data set -1	1	1	1	1	[[7039 21] [24 6380]]	0.06333
Fake News Data set -2	0.98	0.98	0.98	0.99	[[1858 42] [40 6386]]	0.09924
Fake News Data set -3	0.68	0.58	0.63	0.94	[[363 264] [167 6260]]	0.25536
Fake News Data set -4	1	0.99	0.99	0.99	[[6942 41] [34 6453]]	0.07461

K Nearest Neighbor						
	Precision	Recall	F1-Score	Accuracy	Confusion Matrix	Error
Fake News Data set -1	0.62	0.99	0.76	0.67	[[6989 94] [4309 2072]]	0.57354
Fake News Data set -2	0.31	0.98	0.47	0.5	[[1862 38] [4103 2323]]	0.70523
Fake News Data set -3	0.59	1	0.74	0.94	[[624 3] [437 5990]]	0.25144
Fake News Data set -4	0.61	0.99	0.75	0.67	[[6905 78] [4418 2009]]	0.57773

Artificial Neural Network						
	Precision	Recall	F1-Score	Accuracy	Confusion Matrix	Error
Fake News Data set -1	1	1	1	1	[[7083 0] [0 6381]]	0.00861
Fake News Data set -2	1	1	1	1	[[1900 0] [1 6425]]	0.01095
Fake News Data set -3	1	1	1	1	[[627 0] [0 6427]]	0
Fake News Data set -4	1	1	1	1	[[6983 0] [0 6487]]	0

Support Vector Machine						
	Precision	Recall	F1-Score	Accuracy	Confusion Matrix	Error
Fake News Data set -1	1	1	1	1	[[7079 4] [3 6378]]	0.02111
Fake News Data set -2	1	1	1	1	[[1897 3] [3 6423]]	0.026844
Fake News Data set -3	1	0.97	0.99	1	[[611 16] [2 6425]]	0.05324
Fake News Data set -4	1	1	1	1	[[6980 3] [1 6486]]	0.01723

Naive Bayes						
	Precision	Recall	F1-Score	Accuracy	Confusion Matrix	Error
Fake News Data set -1	0.96	0.92	0.94	0.94	[[6486 574] [262 6142]]	0.23364
Fake News Data set -2	0.68	0.64	0.66	0.85	[[1208 692] [576 5850]]	0.39024
Fake News Data set -3	0.97	0.37	0.54	0.94	[[234 393] [7 6420]]	0.24342
Fake News Data set -4	0.96	0.93	0.94	0.94	[[6479 504] [260 6227]]	0.2381

RESULT ANALYSIS (with feature selection)

Decision Tree

	Precision	Recall	F1-Score	Accuracy	Confusion Matrix	Error (root mean)
Fake News Data set -1	1	1	1	1	[[7029 0] [0 6441]]	0
Fake News Data set -2	0.7	0.66	0.68	0.86	[[1255 637] [531 5903]]	0.3745
Fake News Data set -3	0.59	0.62	0.6	0.93	[[390 237] [273 6154]]	0.2688
Fake News Data set -4	1	1	1	1	[[7023 0] [0 6447]]	0

K Nearest Neighbor

	Precision	Recall	F1-Score	Accuracy	Confusion Matrix	Error
Fake News Data set -1	0.97	0.93	0.95	0.95	[[6533 496] [220 6221]]	**0.2305**
Fake News Data set -2	0.99	0.93	0.96	0.98	[[1756 136] [26 6408]]	**0.1394**
Fake News Data set -3	0.09	1	0.17	0.14	[[626 1] [6078 349]]	**0.92832**
Fake News Data set -4	0.96	0.93	0.95	0.95	[[6550 473] [252 6195]]	**0.2319**

Support Vector Machine

	Precision	Recall	F1-Score	Accuracy	Confusion Matrix	Error
Fake News Data set -1	1	1	1	1	[[7029 0] [0 6441]]	**0**
Fake News Data set -2	0.98	0.93	0.95	0.98	[[1814 145] [38 6329]]	**0.1482**
Fake News Data set -3	0.99	0.95	0.97	0.99	[[594 33] [4 6423]]	**0.07242**
Fake News Data set -4	1	1	1	1	[[7023 0] [0 6447]]	**0**

Artificial Neural Network

	Precision	Recall	F1-Score	Accuracy	Confusion Matrix	Error
Fake News Data set -1	1	1	1	1	[[7029 0] [0 6441]]	0
Fake News Data set -2	1	1	1	1	[[1891 1] [0 6434]]	0
Fake News Data set -3	1	1	1	1	[[627 0] [0 6427]]	0
Fake News Data set -4	1	1	1	1	[[7023 0] [0 6447]]	0

Naive Bayes

	Precision	Recall	F1-Score	Accuracy	Confusion Matrix	Error
Fake News Data set -1	1	0.99	1	1	[[6983 46] [4 6437]]	0.0609
Fake News Data set -2	0.77	0.85	0.81	0.91	[[1604 288] [470 5964]]	0.30172
Fake News Data set -3	0.9	0.33	0.49	0.94	[[210 417 [24 6403	0.25003
Fake News Data set -4	1	0.99	1	1	[[6986 37] [1 6446]]	0.05311

CONCLUSION

Finally, classification learning algorithms can be benefitted from the abundance of training data. This work has examined several human annotated datasets that may be utilised for training a machine that successfully detects false news. A generalised model that is able to capture the characteristics of each dataset and learn what is valid and what is not, may be trained on many datasets, providing that they are correctly prepared and supplied to the model.

Despite the relative amount of existent studies addressing fake news detection, there is still plenty of opportunity for experimentation, and the discovery of new insights on the nature of false news may lead to more efficient and accurate models. In addition, this research is, to the best of our knowledge, the first to suggest generalisation of models used for false news identification. As shown in this article, such models tend to operate well on a given dataset, but do not generalise well. New boundaries can be explored by examining generalisation of false news detection algorithms.

Looking at all the results derived after implementing the following classifiers : 1. Decision Trees 2. Naive Bayes 3. KNN 4. ANN 5. SVM on the 4 Fake News Datasets we have arrived at a conclusion that ANN is the most efficient algorithm as it gives the best precision and accuracy and least error.

THANK YOU!

BY:

Adrita Chatterjee(1929133)

Avirup Chattopadhyay(1929209)

Harsh Raj(1929216)

Prayash Mohapatra(1929232)