Fake News Detection using different Machine Learning models

A Project report submitted
in partial fulfillment of requirements
for the award of degree of

Bachelor of Technology
In
Information Technology

By

KANDULA ANUSHA **(Reg No: 17131A1251)**
BORRA KAMAL **(Reg No: 17131A1217)**
ARISETTY AMRUTH **(Reg No: 17131A1210)**
KARRI ADITYA PAVAN **(Reg No: 17131A1255)**

Under the esteemed guidance of

Mr.S. KANTHI KIRAN

M.Tech , Associate Professor.
Department of Information Technology

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING (AUTONOMOUS)
(Affiliated to JNTU-K, Kakinada)
VISAKHAPATNAM
2020 - 2021

Gayatri Vidya Parishad College of Engineering (Autonomous) Visakhapatnam

CERTIFICATE

This report on ***"FAKE NEWS DETECTION USING DIFFERENT MACHINE LEARNING MODELS"*** is a bonafide record of the main project work submitted

By

KANDULA ANUSHA **(Reg No: 17131A1251)**
BORRA KAMAL **(Reg No: 17131A1217)**
ARISETTY AMRUTH **(Reg No: 17131A1210)**
KARRI ADITYA PAVAN **(Reg No: 17131A1255)**

in their VIII semester in partial fulfillment of the requirements for the Award of Degree of

Bachelor of Technology

In

Information Technology

During the academic year 2020-2021

Mr.S. Kanthi Kiran — Project guide

Dr.K.B.Madhuri — Head of the Department, Department of information technology

DECLARATION

I/we here by declare that this main project entitled **"*Fake News Detection using different Machine Learning models*"** is a bonafide work done by me and submitted to **Department of Information Technology G.V.P College of Engineering (Autonomous) Visakhapatnam,** in partial fulfilment for the award of the degree of B.Tech is of my own and it is not submitted to any other university or has been published any time before.

PLACE: VISAKHAPATNAM

DATE: 19-07-2021

K. ANUSHA (17131A1251)

B. KAMAL (17131A1217)

A. AMRUTH (17131A1210)

K. ADITYA (17131A1255)

ACKNOWLEDEGEMENT

We thank our faculty at the department particularly **Mr. S.Kanthi Kiran, IT** for the kind suggestions and guidance through our in house main project work .we also thank him for his guidance that enabled the successful completion of our project work.

we wish to express our deep graduate to **Sri B.MADHURI, Head of the Department of IT ,GVPCOE** for giving us an opportunity to do the project in college .

we wish to express our deep gratitude to **Dr. A. B. KOTESWARARAO, Principal ,GVPCOE** for giving us opportunity to do the project work, giving us a chance to explore and learn new technologies in the form of main project

finally we would like to thank all those people who helped us in many ways in completing this project.

KANDULA ANUSHA (Reg No: 17131A1251)
BORRA KAMAL (Reg No: 17131A1217)
ARISETTY AMRUTH (Reg No: 17131A1210)
KARRI ADITYA PAVAN (Reg No: 17131A1255)

ABSTRACT

This Project comes up with the applications of NLP (Natural Language Processing) techniques for detecting the 'fake news', that is, misleading news stories that comes from the non-reputable sources. Only by building a model based on a count vectorizer (using word tallies) or a (Term Frequency Inverse Document Frequency) tfidf matrix, (word tallies relative to how often they're used in other articles in your dataset) can only get you so far. But these models do not consider the important qualities like word ordering and context. It is very possible that two articles that are similar in their word count will be completely different in their meaning. The data science community has responded by taking actions against the problem. So a proposed work on assembling a dataset of both fake and real news and employ a Naive Bayes ,logistic regression,randomforest classifier in order to create a model to classify an article into fake or real based on its words and phrases.

INDEX

INTRODUCTION

1.1 OBJECTIVE

The goal of this project is to find the effectiveness and limitations of language-based techniques for detection of fake news through the use of machine learning algorithms including but not limited to convolutional neural networks and recurrent neural networks .

1.2 ABOUT THE ALGORITHM

The algorithm is developed on Jupyter notebook. It is developed using python neural network programming which is an extension of the python language which is specially designed for deep learning algorithm development. It is developed for classifying any type of text data ,In this context it is applied to rumour based tweets.

1.3 PURPOSE

The purpose of developing the algorithm is to create a state of art algorithm which can be used to restrict the damage due to spreading of fake rumours.

1.4 SCOPE

The scope of the algorithm is not restricted to the system on which it is developed. It can be served as an API to simply pass the text and get the classified output.It can be used for classifying any type of text and can be extended to other social media platforms as well.

SOFTWARE REQUIRMENTS SPECIFICATION

2.1 FUNCTIONAL REQUIRMENTS

A functional requirements defines a function of a system or it component .A function described as a set of inputs, the behavior, and outputs**.**

2.1.1 SOFTWARE REQUIREMENTS

- Operating system: Windows,Mac,Linux
- Programming language: Python
- Anaconda IDE(Integrated Development Environment)
- Python 3.6
- Packages : numpy 1.16.2

  pandas 0.21.0

  tensorflow 1.13.1

  keras 2.1.2

  sklearn 0.0

2.1.1 HARDWARE REQUiREMENTS

- Processor: Intel/ARM processor
- RAM:8GB
- Disk space: 1 TB

2.2 NON FUNCTIONAL REQUIREMENTS

A non- functional requirement is a requirement that specify criteria that can be used to judge the operation of a system, rather than specific behaviors .Nonfunctional requirements are called qualities of a system, there are as follows:

1. Performance

- Response time

  How long the algorithm takes to classify ?

- Processing time

  How long the algorithm takes to train on data ?

- Query and reporting times

  (Considerable when providing an API)

2. Capacity and scalability

   - Throughput

     How many tasks our system needs to handle ? (Considerable when providing an API).

   - Storage

     How much data are we going to need to achieve what we need?

   - Coding standard

     Did we decide on the coding standards for the algorithm and stick to them?

ALGORITHM ANALYSIS

3.1 EXISTING ALGORITHMS

For classifying there are several existing algorithms like RNN(Recurrent Neural Network) ,Decision Trees,SVM(Support Vector Machines) etc. But they lack the accuracy and performance to be completely reliable.

DRAWBACKS OF EXISTING ALGORITHMS:

- Low Accuracy.
- Low Reliablity.
- Less Robust.
- Less Adaptability.

3.2 PROPOSED ALGORITHM

In the proposed system, each news goes through tokenization process first. Then, unnecessary words are removed and candidate feature words are generated. Each candidate feature words are checked against the dictionary and if its entry is available in the dictionary then its frequency is counted and added to the column in the feature vector that corresponds the numeric map of the word. Alongside with counting frequency, the length of the review is measured and added to the feature vector. Finally, sentiment score which is available in the data set is added in the feature vector. We have assigned negative sentiment as zero valued and positive sentiment as some positive valued in the feature vector. The system is very fast and effective due to semi-supervised and supervised learning. Focused on the content of the review based approaches. As feature we have used word frequency count, sentiment polarity and length of review

ADVANTAGES OF PROPOSED SYSTEM:

- High accuracy compared to existing algorithms.
- It is highly reliable.

- Works on huge amount of data.
- Not restricted to a single scan knowledge extraction

SOFTWARE DESCRIPTION

4.1 ANACONDA IDE

Directly from the platform and without involving DevOps, data scientists can develop and deploy AI and machine learning models rapidly into production. Anaconda provides the tools needed to easily:

- Collect data from files, databases, and data lakes.
- Manage environments with Conda (all package dependencies are taken care of at the time of download).
- Share, collaborate on, and reproduce projects.
- Deploy projects into production with the single click of a button.

Data scientists that work in silos struggle to add value for their organization. That's why Anaconda created an integrated, end-to-end data experience.

4.2 TENSORFLOW:

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the **Google Brain** team for internal Google use. It was released under the Apache License 2.0 on November 9, 2015.

4.3 KERAS:

Keras is an open-source neural-network library written in Python. It is capable of running on top

of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. Chollet also is the author of the XCeption deep neural network model. In 2017, Google's TensorFlow team decided to support Keras in TensorFlow's core library. Chollet explained that Keras was conceived to be an interface rather than a standalone machine learning framework. It offers a higher-level, more intuitive set of abstractions that make it easy to develop deep learning models regardless of the computational backend used.

4.4 PANDAS :

pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals

4.5 SKLEARN :

Scikit-learn (formerly **scikits.learn**) is a free software machine learning library for the Python programming language.

It featuresvarious classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

PROJECT DESCRIPTION

5.1 PROBLEM DEFINITION

This algorithm developed will help in classifying text as real and fake and it greatly helps to curb the spreading of fake news that has devastating impacts on the lives of people.

The objective is to pass a news to the algorithm and identify whether it is a fake or real thereby serving our purpose.

5.2 PROJECT OVERVIEW

This algorithm is built over Logistic regression using tf-idf.

- Deep learning is a class of machine learning algorithms that use multiple layers to progressively extract higher level features from raw input.
- Machine Learning may find a decent outcome on such a critical issue as the spread of fake news issues worldwide. Accordingly, the aftereffects of this examination propose much more, that systems like this might come very much handy and be effectively used to handle this critical issue.
- The dataset in this analysis is relied upon machine learning based statistical calculations like Support Vector Machines (SVM), Naive Bayes (NB), Logistic Regression (LR), Random Forest (RF). In this analysis , Logistic Regression performs best for characterization technique.

5.3 MECHANISM DESCRIPTION

5.3.1 Naïve Bayes Classification:

In machine learning, Naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with powerful (naive) independent assumptions between the features. Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers. The formula for naïve bayes classifier is:

$$P(A|B) = \frac{P(A \bigcap B)}{P(B)} = \frac{P(A) \cdot P(B|A)}{P(B)}$$

A pseudo-count will be implemented in every probability estimate. This ensures that no probability will be zero. It is a way of regularizing Naïve Bayes. where the pseudo-count $\alpha > 0$ is the smoothing parameter ($\alpha = 0$ corresponds to no smoothing). Additive smoothing is a type of shrinkage estimator, as the resulting estimate will be between the empirical estimate xi / N, and the uniform probability 1/d. Most of the times, α is taken as 1 but a smaller value can also be chosen depending on the requirements. The frequency-based probability might introduce zeros when multiplying the probabilities, leading to a failure in preserving the information contributed by the non-zero probabilities. Therefore, a smoothing approach, for example, the Lidstone smoothing, must be adopted to counter this problem. After deciding on these problems, the Naïve Bayes classifier will be mostly used to obtain reasonable results. A smoothing approach will increase the accuracy of the problem which is being attempted. It is also being seen that Naïve Bayes classifier is both simple and powerful for Natural Language Processing (NLP) tasks such as text classification problems

5.3.2 Random Forest Algorithm:

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based

on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

Random Forest is capable of performing both Classification and Regression tasks. It is capable of handling large datasets with high dimensionality. It enhances the accuracy of the model and prevents the overfitting issue.

5.3.3 Support Vector Machine (SVM):

A Support Vector Machine (SVM) is a supervised machine learning algorithm that can be used for both classification and regression purposes. SVMs are mostly used in classification problems. SVMs are founded on the idea of finding a hyperplane that best divides a dataset into two classes. Support vectors are the data points nearest to the hyperplane, the points of a data set that, if deleted, would alter the position of the dividing hyperplane. Because of this, they can be considered the critical elements of a data set. The distance between the hyperplane and the nearest data point from either set is known as the margin. The aim is to choose a hyperplane with the greatest possible margin between the hyperplane and any point within the training set, giving a higher chance of new data being classified correctly. The expression for this kernel is given by the following expression:

$$G(x;\sigma) = \frac{1}{\sqrt[2]{2\pi}\sigma} exp(\frac{x^2}{2\sigma^2})$$

5.3.4 Logistic Regression:

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Unlike linear regression which outputs continuous number values, logistic regression changes its yield utilizing the calculated sigmoid capacity to restore a likelihood esteem which would then be able to be mapped to at least two discrete classes. The LR model uses gradient descent to converge onto the optimal set of weights (θ) for the training

set. For our model, the hypothesis used is the sigmoid function.

5.3.5 Count Vector:

Count Vector represents a notation in the form of a matrix data set matrix notation in which corpus document is represented by each row, each column represents a corpus term, and each cell represents the frequency count of a particular term in a particular document.

5.3.6 TF-IDF:

TF-IDF represents how frequent a term is in an entire document. It tries to assign a metric value to represent the presence of that term. This is widely and frequently utilized in text mining. This weight is a factual measure used to assess how essential a word is to a report in a gathering or corpus.

5.3.7 N- Gram:

It is a sequence of tokens. In the context of computational linguistics, these tokens are usually words, though they can be characters or subsets of characters. The n simply refers to the number of tokens.

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + exp(-\theta^T x)}$$

5.3.8 Evaluation Metrics:

To evaluate the performance of algorithms for fake news de-tection problem, various evaluation metrics have been used. In this subsection, we review the most widely used metrics for fake news detection. Most existing approaches consider the fake news problem as a classification problem that pre-dicts whether a news article is fake or not:

- True Positive (TP): when predicted fake news pieces are actually annotated as fake news;

- True Negative (TN): when predicted true news pieces are actually annotated as true news;
- False Negative (FN): when predicted true news pieces are actually annotated as fake news;
- False Positive (FP): when predicted fake news pieces are actually annotated as true news.

By formulating this as a classification problem, we can define following metrics,

Precision =|T P | / |T P |+|F P | (2) Recall =|T P | / |T P |+|F N | (3)

F1=2· Precision ·Recall / Precision +Recall (4)

Accuracy =|T P |+|T N | / |T P |+|T N |+|F P |+|F N| (5)

These metrics are commonly used in the machine learning community and enable us to evaluate the performance of a classifier from different perspectives. Specifically, accuracy measures the similarity between predicted fake news and real fake news. Precision measures the fraction of all detected fake news that are annotated as fake news, addressing the important problem of identifying which news is fake. How- ever, because fake news datasets are often skewed, a high precision can be easily achieved by making fewer positive predictions. Thus, recall is used to measure the sensitivity, or the fraction of annotated fake news articles that are predicted to be fake news. F1 is used to combine precision and recall, which can provide an overall prediction performance for fake news detection. Note that for Precision, Recall, F1, and Accuracy, the higher the value, the better the performance.

DEVELOPMENT

6.1 DATASET:

Taken from Kaggle: https://www.kaggle.com/nopdev/real-and-fake-news-dataset

6.2 CODE :

```
#-------------------------------------------------------------
# Include Libraries
#-------------------------------------------------------------

import pandas as pd
import pandas

print(pandas.__version__)
from sklearn.model_selection import train_test_split
import sklearn
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import plot_confusion_matrix
from sklearn.svm import SVC
from sklearn import metrics
#from pandas_ml import ConfusionMatrix
from matplotlib import pyplot as plt
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.feature_extraction.text import HashingVectorizer
import itertools
import numpy as np
import re
import csv
import pickle
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.stem import SnowballStemmer

#-------------------------------------------------------------
# Importing dataset using pandas dataframe
#-------------------------------------------------------------
```

```
df = pd.read_csv("fake_or_real_news.csv")

# Inspect shape of `df`
df.shape

# Print first lines of `df`
df.head()

# Set index
df = df.set_index("Unnamed: 0")

# Print first lines of `df`
df.head()
print(df.head())



#------------------------------------------------------------
# Separate the labels and set up training and test datasets
#------------------------------------------------------------
y = df.label
df.drop("label", axis=1)      #where numbering of news article is done that column is dropped in dataset
X_train, X_test, y_train, y_test = train_test_split(df['text'], y, test_size=0.33, random_state=53)


with open('Train-SetX.csv','w',encoding='utf-8',newline='') as file:
        writer = csv.writer(file, delimiter=',')
        for line in X_train:
                print(line)
                writer.writerow([line])


with open('Test-SetX.csv','w',encoding='utf-8',newline='') as file:
        writer = csv.writer(file, delimiter=',')
        for line in X_test:
                writer.writerow([line])

with open('Train-SetY.csv','w',encoding='utf-8',newline='') as file:
        writer = csv.writer(file, delimiter=',')
        for line in y_train:
                writer.writerow([line])


with open('Test-SetY.csv','w',encoding='utf-8',newline='') as file:
        writer = csv.writer(file, delimiter=',')
        for line in y_test:
                writer.writerow([line])


count_vectorizer = CountVectorizer(stop_words='english')
count_train = count_vectorizer.fit_transform(X_train)              # Learn the vocabulary dictionary and return
term-document matrix.
count_test = count_vectorizer.transform(X_test)
```

```
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)    # This removes words which appear in more than 70% of the articles
tfidf_train = tfidf_vectorizer.fit_transform(X_train)
tfidf_test = tfidf_vectorizer.transform(X_test)


n_vect = CountVectorizer(min_df = 5, ngram_range = (1,2)).fit(X_train)
n_train = n_vect.fit_transform(X_train)
n_test = n_vect.transform(X_test)


#---------------------------------------------------------------
# Function to plot the confusion matrix
#---------------------------------------------------------------

def plot_confusion_matrix(cm, classes,normalize=False,title='',cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.show()




#---------------------------------------------------------------
# Naive Bayes classifier for Multinomial model
#---------------------------------------------------------------
def NaiveBayes(xtrain,ytrain,xtest,ytest,ac):
            clf = MultinomialNB(alpha=.01, fit_prior=True)
            clf.fit(xtrain, ytrain)
            pred = clf.predict(xtest)
            score = metrics.accuracy_score(ytest, pred)
            print("accuracy:   %0.3f" % score)
            cm = metrics.confusion_matrix(ytest, pred, labels=['FAKE', 'REAL'])
```

```
        plot_confusion_matrix(cm, classes=['FAKE', 'REAL'],title='Confusion matrix Naive Bayes')
        print(cm)
        ac.append(score)


def Logreg(xtrain,ytrain,xtest,ytest,ac):
    i=1
    logreg = LogisticRegression(C=9)
    logreg.fit(xtrain,ytrain)
    pred = logreg.predict(xtest)
    score = metrics.accuracy_score(ytest, pred)
    print("accuracy:   %0.3f" % score)
    cm = metrics.confusion_matrix(ytest, pred, labels=['FAKE', 'REAL'])
    plot_confusion_matrix(cm, classes=['FAKE', 'REAL'],title='Confusion matrix Logistic')
    print(cm)
    ac.append(score)

def RForest(xtrain,ytrain,xtest,ytest,ac):
    clf1 = RandomForestClassifier(max_depth=50, random_state=0,n_estimators=25)
    clf1.fit(xtrain,ytrain)
    pred = clf1.predict(xtest)
    score = metrics.accuracy_score(ytest, pred)
    print("accuracy:   %0.3f" % score)
    cm = metrics.confusion_matrix(ytest, pred, labels=['FAKE', 'REAL'])
    plot_confusion_matrix(cm, classes=['FAKE', 'REAL'],title='Confusion matrix RForest')
    print(cm)
    ac.append(score)


def SVM(xtrain,ytrain,xtest,ytest,ac):
    clf3 = SVC(C=100, gamma=0.1)
    clf3.fit(xtrain, ytrain)
    pred = clf3.predict(xtest)
    score = metrics.accuracy_score(ytest, pred)
    print("accuracy:   %0.3f" % score)
    cm = metrics.confusion_matrix(ytest, pred, labels=['FAKE', 'REAL'])
    plot_confusion_matrix(cm, classes=['FAKE', 'REAL'],title='Confusion matrix SVM')
    print(cm)
    ac.append(score)



def process(xtrain,ytrain,xtest,ytest,ac):
        print("For Multinomial Naive BayesModel")
        NaiveBayes(xtrain,ytrain,xtest,ytest,ac)

        print("For Random Forest Classifiers")
        RForest(xtrain,ytrain,xtest,ytest,ac)

        print("For Support Vector Machine_Radial Basis Function Classifier")
        SVM(xtrain,ytrain,xtest,ytest,ac)

        print("For Logarithamic Classifier")
        Logreg(xtrain,ytrain,xtest,ytest,ac)
```

```
colors = ["#1f77b4", "#ff7f0e", "#2ca02c", "#d62728", "#8c564b"]
explode = (0.1, 0, 0, 0, 0)

al=["NaiveBayes","Random Forest","LogisticRegressio","SVM"]
cac=[]
tac=[]
nac=[]

process(count_train,y_train,count_test,y_test,cac)
print(cac)

result2=open('CountAccuracy.csv', 'w')
result2.write("Algorithm,Accuracy" + "\n")
for i in range(0,len(cac)):
    print(al[i]+","+str(cac[i]))
    result2.write(al[i] + "," +str(cac[i]) + "\n")
result2.close()


fig = plt.figure(0)
df =  pd.read_csv('CountAccuracy.csv')
acc = df["Accuracy"]
alc = df["Algorithm"]
plt.bar(alc, acc, align='center', alpha=0.5,color=colors)
plt.xlabel('Algorithm')
plt.ylabel('Accuracy')
plt.title('Count Accuracy Value')
fig.savefig('CountAccuracy.png')
plt.show()

process(tfidf_train,y_train,tfidf_test,y_test,tac)
print(tac)

result2=open('TfidfAccuracy.csv', 'w')
result2.write("Algorithm,Accuracy" + "\n")
for i in range(0,len(cac)):
    print(al[i]+","+str(cac[i]))
    result2.write(al[i] + "," +str(tac[i]) + "\n")
result2.close()


fig = plt.figure(0)
df =  pd.read_csv('TfidfAccuracy.csv')
acc = df["Accuracy"]
alc = df["Algorithm"]
plt.bar(alc, acc, align='center', alpha=0.5,color=colors)
plt.xlabel('Algorithm')
plt.ylabel('Accuracy')
plt.title('Tfidf Accuracy Value')
fig.savefig('TfidfAccuracy.png')
plt.show()

process(n_train,y_train,n_test,y_test,nac)
```

```
print(nac)

result2=open('NgramAccuracy.csv', 'w')
result2.write("Algorithm,Accuracy" + "\n")
for i in range(0,len(cac)):
    print(al[i]+","+str(cac[i]))
    result2.write(al[i] + "," +str(nac[i]) + "\n")
result2.close()


fig = plt.figure(0)
df =  pd.read_csv('NgramAccuracy.csv')
acc = df["Accuracy"]
alc = df["Algorithm"]
plt.bar(alc, acc, align='center', alpha=0.5,color=colors)
plt.xlabel('Algorithm')
plt.ylabel('Accuracy')
plt.title('Ngram Accuracy Value')
fig.savefig('NgramAccuracy.png')
plt.show()

#------------------------------------------------------------
# Creating pickle files
#------------------------------------------------------------

tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)
tfidf_train = tfidf_vectorizer.fit_transform(X_train)
pickle.dump(tfidf_vectorizer, open('C:\\Users\\kamal\\Downloads\\fake news detection
project\\pickles\\pre_training.pkl', 'wb'))

logreg = LogisticRegression(C=9)
logreg.fit(tfidf_train,y_train)
pickle.dump(logreg, open('C:\\Users\\kamal\\Downloads\\fake news detection project\\pickles\\model.pkl',
'wb'))
```

6.3.1 TRAIN OUTPUT:

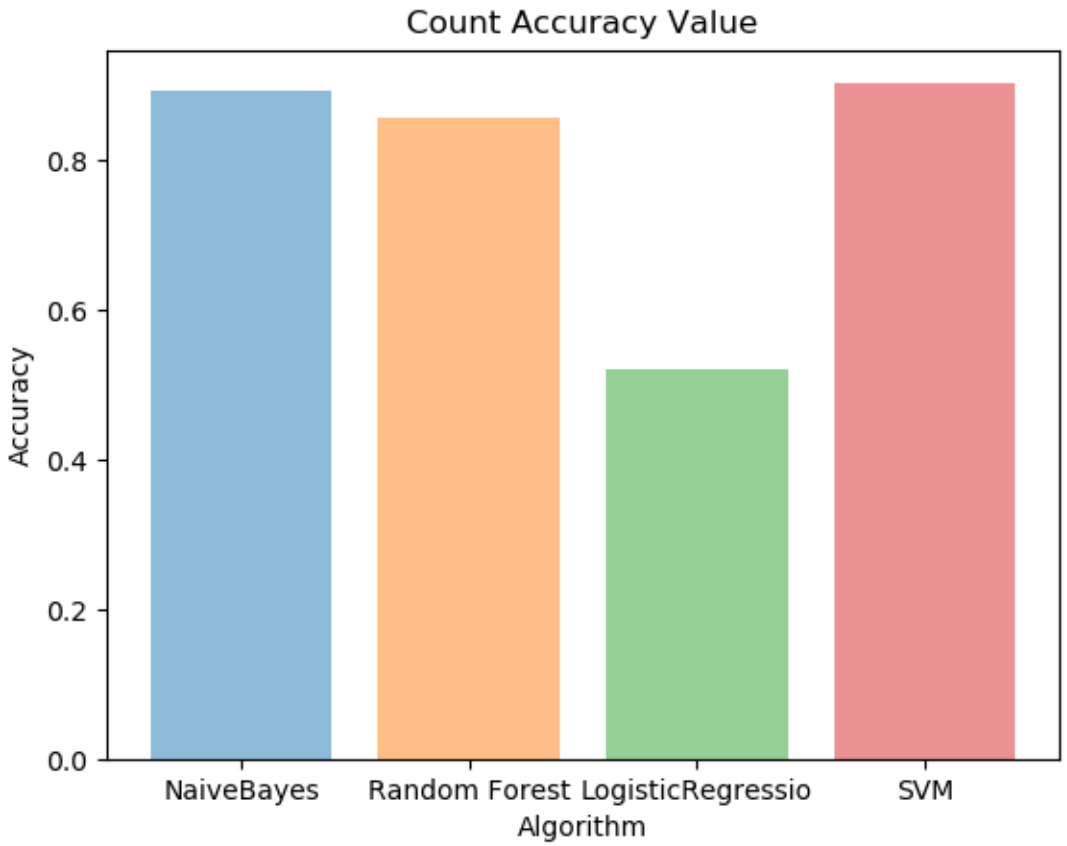
```
Epoch 1/15
2882/2882 [==============================] - 9s 3ms/step - loss: 0.6481 - acc: 0.6426 - va
l_loss: 0.4734 - val_acc: 0.7835
Epoch 2/15
2882/2882 [==============================] - 7s 2ms/step - loss: 0.4890 - acc: 0.7882 - va
l_loss: 0.8193 - val_acc: 0.8100
Epoch 3/15
2882/2882 [==============================] - 6s 2ms/step - loss: 0.4728 - acc: 0.8151 - va
l_loss: 0.3740 - val_acc: 0.8598
```

```
Epoch 4/15
2882/2882 [==============================] - 6s 2ms/step - loss: 0.3867 - acc: 0.8359 - va
l_loss: 0.3169 - val_acc: 0.8723
Epoch 5/15
2882/2882 [==============================] - 6s 2ms/step - loss: 0.3441 - acc: 0.8492 - va
l_loss: 0.3023 - val_acc: 0.8785
Epoch 6/15
2882/2882 [==============================] - 6s 2ms/step - loss: 0.3219 - acc: 0.8636 - va
l_loss: 0.3059 - val_acc: 0.8660
Epoch 7/15
2882/2882 [==============================] - 6s 2ms/step - loss: 0.3074 - acc: 0.8714 - va
l_loss: 0.2841 - val_acc: 0.8988
Epoch 8/15
2882/2882 [==============================] - 6s 2ms/step - loss: 0.2879 - acc: 0.8773 - va
l_loss: 0.2817 - val_acc: 0.8863
Epoch 9/15
2882/2882 [==============================] - 6s 2ms/step - loss: 0.2742 - acc: 0.8858 - va
l_loss: 0.2745 - val_acc: 0.9034
Epoch 10/15
2882/2882 [==============================] - 6s 2ms/step - loss: 0.2644 - acc: 0.8890 - va
l_loss: 0.2698 - val_acc: 0.8941
Epoch 11/15
2882/2882 [==============================] - 6s 2ms/step - loss: 0.2534 - acc: 0.8930 - va
l_loss: 0.2763 - val_acc: 0.8941
Epoch 12/15
2882/2882 [==============================] - 6s 2ms/step - loss: 0.2556 - acc: 0.8909 - va
l_loss: 0.2544 - val_acc: 0.9143
Epoch 13/15
2882/2882 [==============================] - 6s 2ms/step - loss: 0.2271 - acc: 0.9093 - va
l_loss: 0.2605 - val_acc: 0.8972
Epoch 14/15
2882/2882 [==============================] - 6s 2ms/step - loss: 0.2148 - acc: 0.9098 - va
l_loss: 0.2661 - val_acc: 0.9003
Epoch 15/15
2882/2882 [==============================] - 6s 2ms/step - loss: 0.2078 - acc: 0.9146 - va
l_loss: 0.2763 - val_acc: 0.8910
```

6.3.2 OUTPUT:

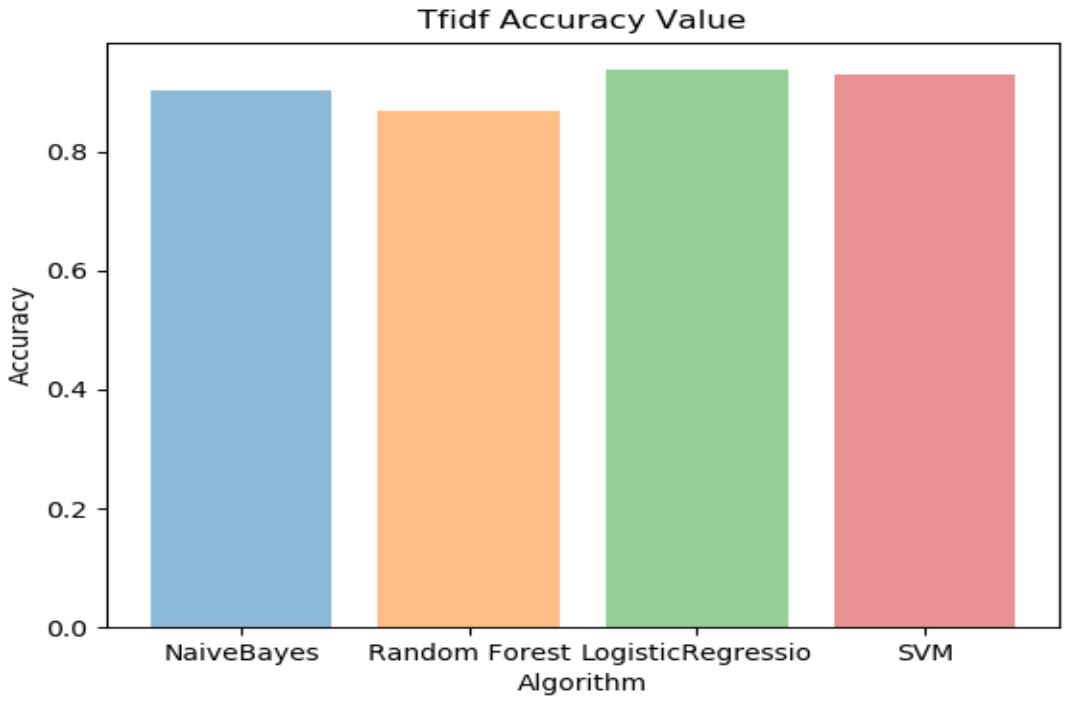
Count Vectorizer, Accuracy

NaiveBayes,0.8923959827833573
Random Forest,0.8565279770444764
LogisticRegressio,0.5212816834050693
SVM,0.9029172644667623



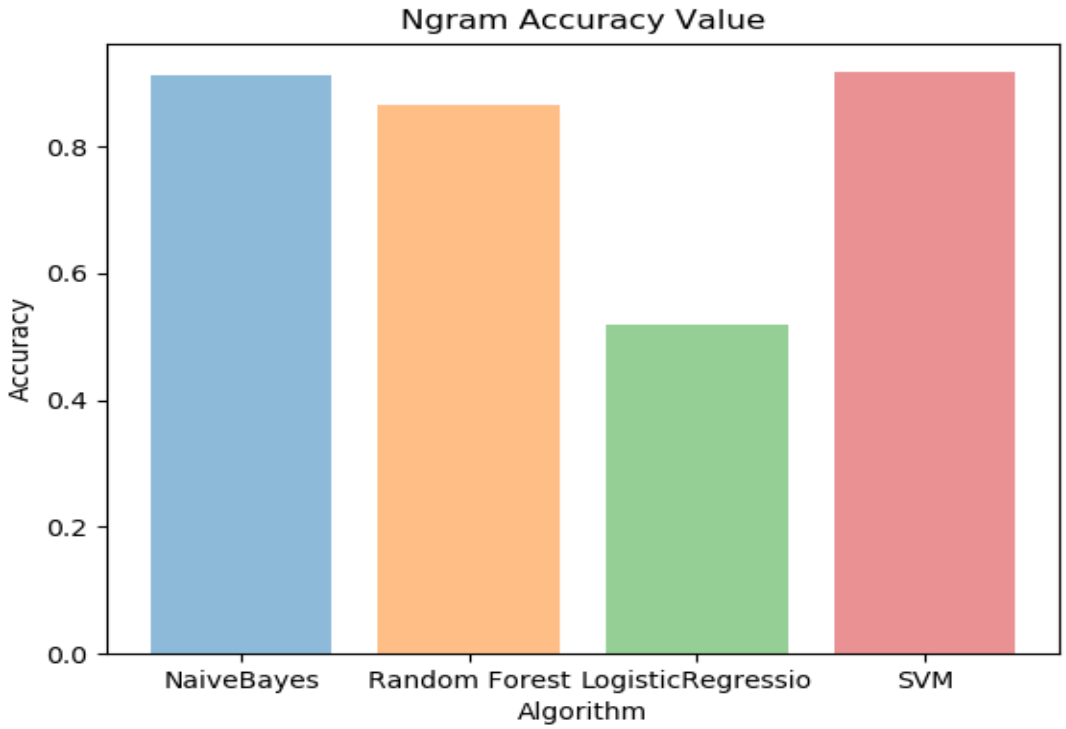
Tf-Idf Algorithm, Accuracy

NaiveBayes,0.9033955045432808
Random Forest,0.8675274988043998
LogisticRegressio,0.937350549976088
SVM,0.9311334289813487

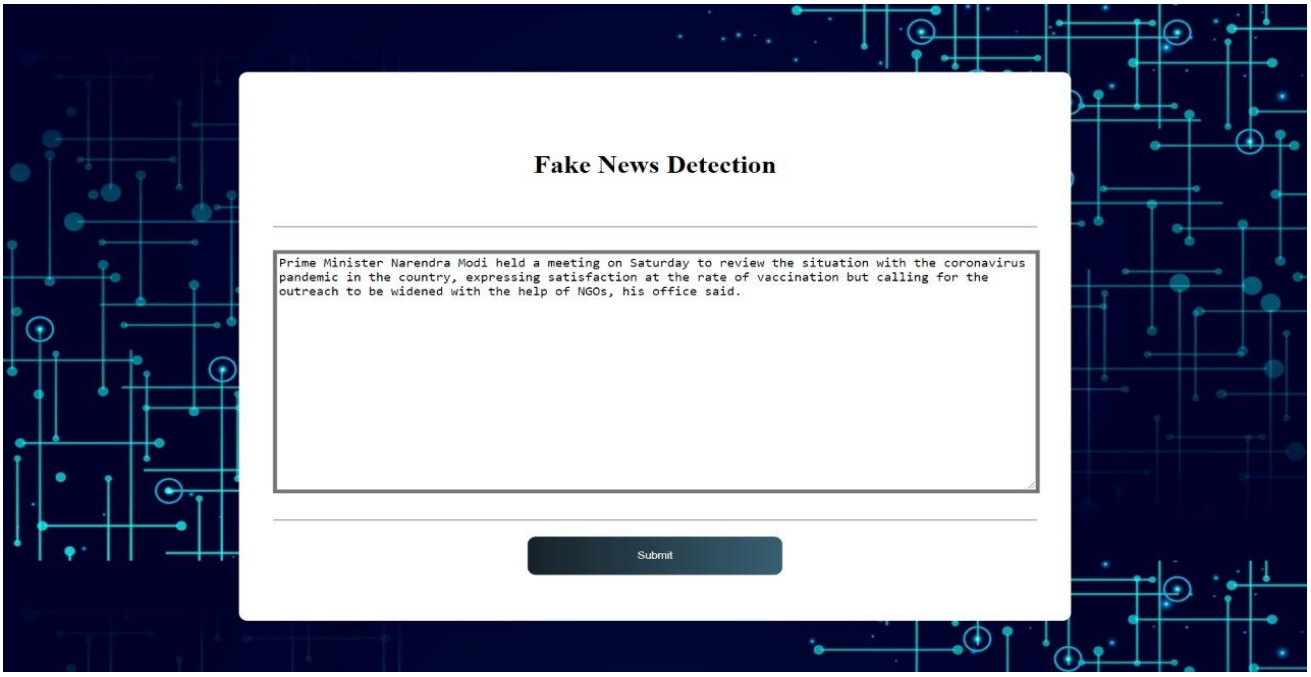


N-gram Algorithm, Accuracy

NaiveBayes,0.9120038259206121
Random Forest,0.8641798182687709
LogisticRegressio,0.5188904830224773
SVM,0.9167862266857962




Fake News Detection
Enter news to detect
Submit


Fake News Detection
Prime Minister Narendra Modi held a meeting on Saturday to review the situation with the coronavirus pandemic in the country, expressing satisfaction at the rate of vaccination but calling for the outreach to be widened with the help of NGOs, his office said.
Submit


Fake News Detection
Prime Minister Narendra Modi held a meeting on Saturday to review the situation with the coronavirus pandemic in the country, expressing satisfaction at the rate of vaccination but calling for the outreach to be widened with the help of NGOs, his office said.
Entered news is real


Fake News Detection
Narendra Modi was shot on 30 January 1948 by the Hindu fanatic Nathuram Godse.
Submit


Fake News Detection
Narendra Modi was shot on 30 January 1948 by the Hindu fanatic Nathuram Godse.
Entered news is fake

CONCLUSION

In the 21st century, the majority of the tasks are done online. Newspapers who were earlier preferred as hard-copies are now being substituted by applications like Facebook, Twitter, and news articles to be read online. The growing problem of fake news only makes things more complicated and tries to change or hamper the opinion and attitude of people towards use of digital technology. When a person is deceived by the real news two possible things happen.People start believing that their perceptions about a particular topic are true as assumed. Another problem is that even if there is any news article available which contradicts a supposedly fake one, people believe in the words which just support their thinking without taking in the measure the facts involved. Thus, in order to curb the phenomenon, Google and Facebook.

BIBLIOGRAPHY

1. Conroy, Niall & Rubin, Victoria & Chen, Yimin. (2015). Automatic Deception Detection: Methods for Finding Fake News. USA
2. Ball, L. & Elworthy, J. J Market Anal (2014) 2: 187. https://doi.org/10.1057/jma.2014.15
3. Lu TC. Yu T., Chen SH. (2018) Information Manipulation and Web Credibility. In: Bucciarelli E., Chen SH., Corchado J. (eds) Decision Economics: In the Tradition of Herbert A. Simon's Heritage. DCAI 2017. Advances in Intelligent Systems and Computing, vol 618. Springer, Cham
4. Rubin, Victoria & Conroy, Niall & Chen, Yimin & Cornwell, Sarah. (2016). Fake News or Truth? Using Satirical Cues to Detect Potentially Misleading News.10.18653/v1/W160802.
5. 0. Wang, W.Y.: Liar, Liar Pants on fire: a new Benchmark dataset for fake newsdetection. arXiv preprint (2017). arXiv:1705.00648.
6. Rubin., Victoria, L., et al.: Fake news or truth? Using satirical cues to detect potentially misleading news. In: Proceedings of NAACL-HLT (2016).
7. Shivam B. Parikh and Pradeep K. Atrey, “Media-Rich Fake News Detection: A Survey”, IEEE Conference on Multimedia Information Processing and Retrieval, 2018.
8. Hunt Allcott and Matthew Gentzkow. Social mediaand fake news in the 2016 election. Technical report,National Bureau of Economic Research, 2017.