# 

**METHODOLOGY:**

**Step 1: Data Collection**

We gathered the datasets from Kaggle website. The data set contains a large of data comprising of both real and fake news.

The essential features of the dataset must have the following attributes assigned to them:

·       A unique id for news article

·       Title of news article

·       Author’s name

·       The main body of the text : The text article could be complete or incomplete.

·       Label assigned to them (0/1 : fake/real)

**Step 2: Data Preprocessing**

**Text Cleaning:** For the preparation of the appropriate to the python Machine Learning models a number of files/libraries are imported. These libraries along with their use in the program are listed below.

·       numpy is used here for making numpy arrays for computations in methods and functions used by various libraries.

·       pandas is used here for making data frames, in order to store data in the data frames and to filter out the data. The missing element rows are discarded. It also differentiate the source and target data for the learning process.

·       re is used for searching and dropping particular symbols and particular words from text/ paragraph.

·       pyplot from matplotlib library is used for plotting confusion matrix and the ROC graph.

·       stopwords are removed (common words like "and", "the", "is", “what”, “when” etc.) using the ntlk library.

·       lemmatize or stemming of words to their root forms are also incorporated in the text using the re/ntlk library.

**Feature Extraction:**

·       TfidfVectorizer is used to convert textual data into numeric data for model computations.

·       Missing values from the dataset is replaced by null strings

·       Data and label columns are separated.

**Step 3: Train-Test Split**

·       train\_test\_split is used to divide data into training data and testing data.

·       We took 25% of the total dataset as test data and remaining 75%  as train data. Vectorization of both the data sets are essential running the programs.

**Step 4: Model Building**

Different models are trained using sklearn library .

Some of them are listed below:

o   Logistic Regression

o   KMeans

o   Support Vector  Machine

o   Decision Tree Classifier

o   Random Forest Classifier

**Step 4: Model Evaluation:**

The evaluation of our proposed model is carried out using some crucial parameters like confusion matrix, accuracy score, precision score , recall score , f1 score and ROC curve .

The proposed model with logistic regression approach was compared with other existing classification models for justification of the scheme with logistic regression. Finally the project / scheme is tested both in python idle compiler and in a designed website.

**CHAPTER - 4**

**RESULTS AND DISCUSSION**

The results of the analysis of the datasets using the Logistic Regression has been depicted using the confusion matrix. The confusion matrix is automatically obtained by Python code using the cognitive learning library when running the algorithm code in Google Colab Platform. The confusion matrix so obtained from the analysis shows the False Positives and False Negatives as shown in fig. (1).

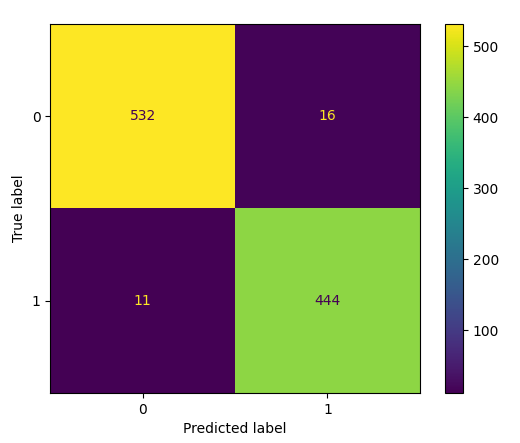


Fig. (1) Confusion Matrix for Logistic Regression

The figure shows 11 False Positives and 16 False Negatives using the Logistic Regression algorithm in a balanced dataset with a TF-IDF vectorizer. The train data contains about 3000 number of data-sets which is nearly 75% of data from the data-set. The data is subsequently filtered and stemmed out using functions of the natural language processing library and other functions used for processing text data in an organized form.

The evaluation parameters obtained from the different metrices library of Python are tabulated in Table- I. These shows that the results obtained from the test data are in good agreement with the actual value. The ROC curve shown in fig (2) clearly depicts the fact that the model is well working within permissible limit.

Table - I

|  |  |
| --- | --- |
| Evaluation Parameters | Obtained Scores |
| Correct Results | 976 out of 1003 |
| Accuracy Score | 0.973 |
| Precision Score | 0.965 |
| Recall Score | 0.976 |
| F1 Score | 0.970 |

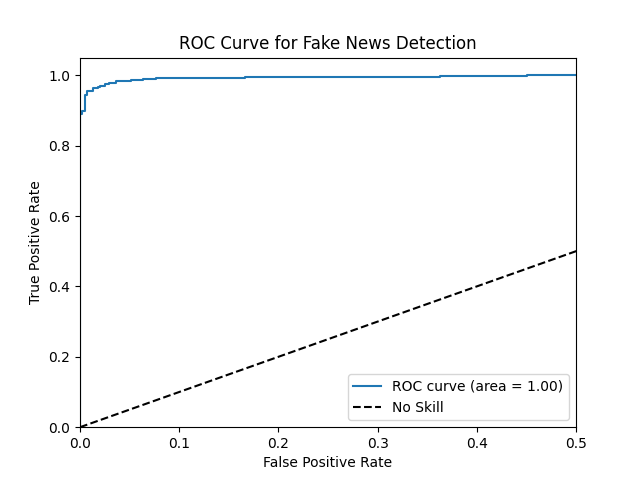


Fig. 2. ROC curve for Logistic Regression

The scheme is tested with the different commonly used Classification algorithms. The accuracy scores for all those cases are tabulated in Table – 2. Comparison among all these schemes shows that the Logistic Regression and the SVM gives better and nearly equal results. However, among these two, the Logistic Regression should be the preferred choice as it work relatively faster and is also easy to implement which clearly justifies the choice of model in this work.

Table - II

|  |  |
| --- | --- |
| **Classification Algorithm** | **Accuracy Score** |
| Decision Tree | 0.962 |
| Random Forest | 0.966 |
| K Nearest Neighbour | 0.894 |
| Support Vector Machine | 0.989 |
| Logistic Regression | 0.974 |
| Naïve Bayes | 0.841 |

The screenshot of the Accuracy Result of the Python program for various commonly used Classification Models obtained under sklearn library, with the given data-sets is shown in fig. 3.

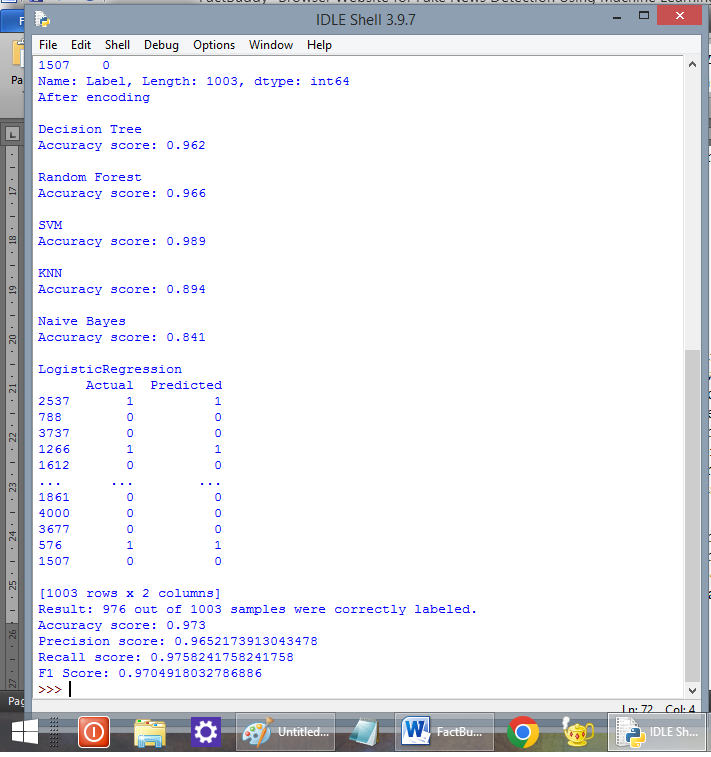


Fig. 3. Screenshot of the Accuracy scores with different Models

After saving the model and the TfidfVectorizer , the model is tested without training. The input for this testing verification is done using a different data-set. The result obtained from actual implementation is shown in the screenshot in fig. 4.

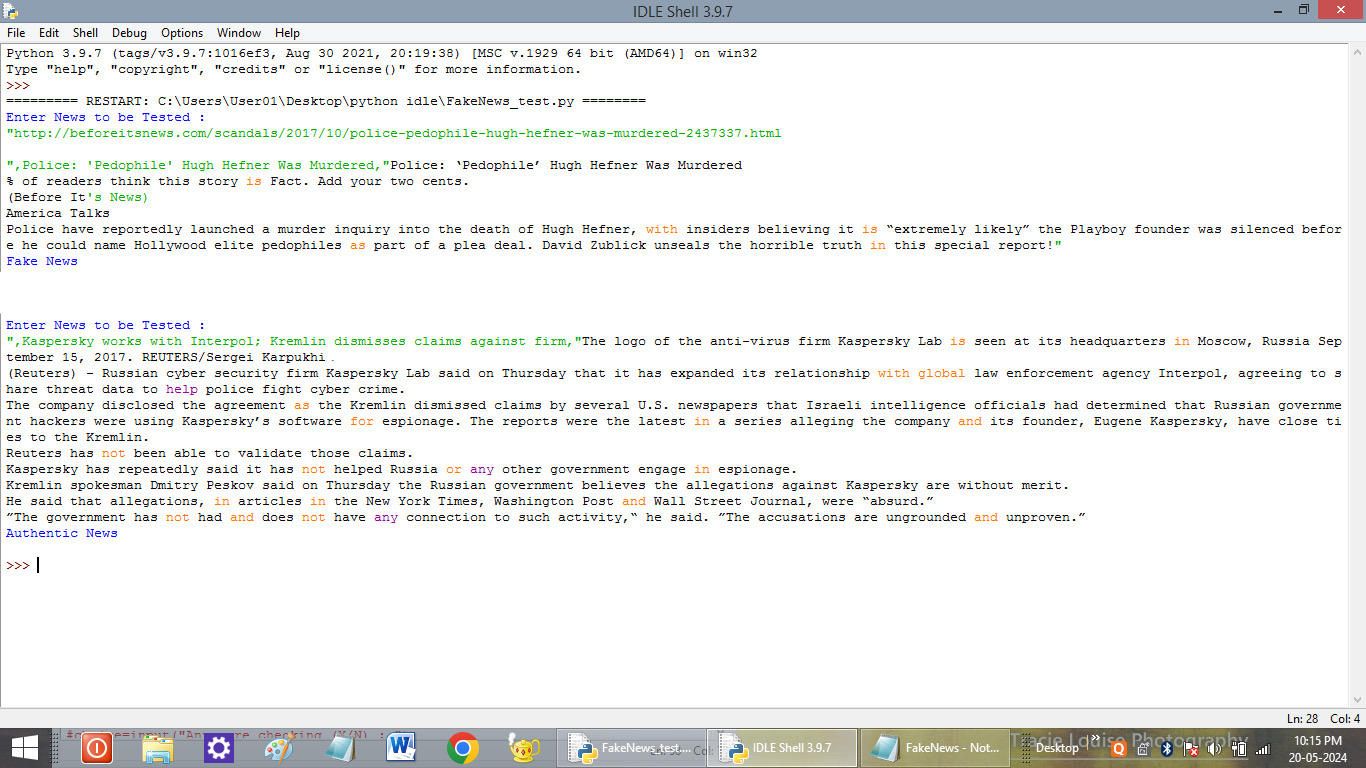


Fig. 4. Results showing Fake News and Authenticated News Detection

APPENDIX

**Python Code**

import pandas as pd

import numpy as np

import re

import matplotlib as plt

import matplotlib.pyplot as pyplot

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.linear\_model import LogisticRegression

from sklearn.cluster import KMeans

from sklearn import svm

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay,precision\_score,recall\_score,f1\_score,roc\_curve,auc

import nltk

nltk.download('stopwords')

from sklearn import preprocessing

import os

input\_dir='C:/Users/User01/Desktop/python idle/Data/'

df\_train=pd.read\_csv(input\_dir+'FakeNews.csv')

col\_name=list(df\_train)

df\_train=df\_train.fillna(' ')

df\_train['content']=df\_train['Headline']+' '+df\_train['Body']

X=df\_train.drop(['Label'],axis=1)

y=df\_train['Label']

# stemming

port\_stem=PorterStemmer()

def stemming(content):

stemmed\_content=re.sub('[^a-zA-Z]',' ',content)

stemmed\_content=stemmed\_content.lower()

stemmed\_content=stemmed\_content.split()

for word in (stemmed\_content):

if word not in stopwords.words('english'):

stemmed\_content=port\_stem.stem(word)

stemmed\_content=' '.join(stemmed\_content)

return stemmed\_content

df\_train['content']=df\_train['content'].apply(stemming)

X=df\_train['content'].values

Y=df\_train['Label'].values

#Vectorization

vectorizer=TfidfVectorizer()

vectorizer.fit(X)

X=vectorizer.fit\_transform(X)

# Training and testing set of data

X\_train,X\_test,y\_train,y\_test=train\_test\_split( X,y,test\_size=0.25,shuffle=True,random\_state=0)

print('Train Data')

print(X\_train,y\_train)

print('Test Data')

print(X\_test,y\_test)

print('After encoding')

print("\nDecision Tree")

clf=DecisionTreeClassifier(random\_state=10)

clf.fit(X\_train,y\_train)

res=clf.predict(X\_test)

correct\_results = sum(y\_test == res)

print('Accuracy score: %0.3f' % (correct\_results/float(y\_test.size)))

print("\nRandom Forest")

clf=RandomForestClassifier(random\_state=10 )

clf.fit(X\_train,y\_train)

res=clf.predict(X\_test)

correct\_results = sum(y\_test == res)

print('Accuracy score: %0.3f' % (correct\_results/float(y\_test.size)))

print("\nSVM")

clf=svm.LinearSVC()

clf.fit(X\_train,y\_train)

res=clf.predict(X\_test)

correct\_results = sum(y\_test == res)

print('Accuracy score: %0.3f' % (correct\_results/float(y\_test.size)))

print("\nKNN")

from sklearn.neighbors import KNeighborsClassifier

clf = KNeighborsClassifier(n\_neighbors=4 )

clf.fit(X\_train,y\_train)

res=clf.predict(X\_test)

correct\_results = sum(y\_test == res)

print('Accuracy score: %0.3f' % (correct\_results/float(y\_test.size)))

print("\nNaive Bayes")

from sklearn.naive\_bayes import BernoulliNB

clf = BernoulliNB()

clf.fit(X\_train,y\_train)

res=clf.predict(X\_test)

correct\_results = sum(y\_test == res)

print('Accuracy score: %0.3f' % (correct\_results/float(y\_test.size)))

print("\nLogisticRegression")

clf=LogisticRegression()

clf.fit(X\_train,y\_train)

#Model analysis

res=clf.predict(X\_test)

y\_pred\_proba = clf.predict\_proba(X\_test)[:, 1]

correct\_results = sum(y\_test == res)

print(pd.DataFrame({'Actual': y\_test, 'Predicted':res}))

precision = precision\_score(y\_test, res)

recall = recall\_score(y\_test, res)

f1 = f1\_score(y\_test, res)

print("Result: %d out of %d samples were correctly labeled." % (correct\_results, y\_test.size))

print('Accuracy score: %0.3f' % (correct\_results/float(y\_test.size)))

print("Precision score:", precision)

print("Recall score:", recall)

print("F1 Score:", f1)

import pickle

with open('model.bm1', 'wb') as f:

pickle.dump(clf, f)

with open('vectorizer.bm1', 'wb') as f:

pickle.dump(vectorizer, f)

# for external data

choice=input("Do you want to test news (Y for yes N for no) : ")

while choice=="Y" or choice=="y":

news=input("Enter News to be Tested :\n")

testing\_news={"text":[news]}

new\_def\_test=pd.DataFrame(testing\_news)

new\_def\_test["text"]=new\_def\_test["text"].apply(stemming)

new\_x\_test=new\_def\_test["text"]

#print (new\_x\_test)

new\_xv\_test=vectorizer.transform(new\_x\_test)

test\_res=clf.predict(new\_xv\_test)

if test\_res[0]==0:

print("Fake News")

else:

print("Authentic News")

choice=input("Any more checking (Y/N) : ")

# Plot

cm = confusion\_matrix(y\_test, res, labels=clf.classes\_)

disp = ConfusionMatrixDisplay(confusion\_matrix=cm,display\_labels=clf.classes\_)

disp.plot()

pyplot.show()

# Calculate ROC curve

fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred\_proba)

roc\_auc = auc(fpr, tpr)

# Plot the ROC curve

pyplot.figure()

pyplot.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % roc\_auc)

pyplot.plot([0, 1], [0, 1], 'k--', label='No Skill')

pyplot.xlim([0.0, 0.5])

pyplot.ylim([0.0, 1.05])

pyplot.xlabel('False Positive Rate')

pyplot.ylabel('True Positive Rate')

pyplot.title('ROC Curve for Fake News Detection')

pyplot.legend()

pyplot.show()