****

**IBM NAAN MUDHALVAN PROJECT**

**PROJECT NAME :-FAKE NEWS DETECTION USING NLP**

**TEAM MEMBERS:-**

|  |  |  |
| --- | --- | --- |
| **Serial no:** | **Team members name** | **Team members email id** |
| 1. | P.Udhaiyalakshmi | udayalaxmi2002@gmail.com |
| 2. | V.Swarnalakshmi | swarnalakshmi065@gmail.com |
| 3. | M.Sneha | keccse21097@kingsedu.ac.in |
| 4. | K.Srivarthini | srivaruthik007@gmail.com |
| 5. | H.Stephy graph | graphstephy171@gmail.com |

Abstract:-

Here we will talk about how one can create an NLP to detect whether the news is real or fake. Nowadays, fake news has become a common trend. Even trusted media houses are known to spread fake news and are losing their credibility. By using the dataset collected were cleaned and processed , however the punctuations and mistakes that existed in the fake news were kept in the text.

**Data preprocessing:-**

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.

Why do we need Data Preprocessing?

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

It involves below steps:

* Getting the dataset
* Importing libraries
* Importing datasets
* Finding Missing Data
* Encoding Categorical Data
* Splitting dataset into training and test set
* Feature scaling

**Loading dataset:-**

With Python Standard Library, you will be using the module CSV and the function reader() to load your CSV files. Upon loading, the CSV data will be automatically converted to NumPy array which can be used for machine learning.

**Program for loading and preprocessing:**

**Import libraries**

# Modelling Algorithms

from sklearn.naive\_bayes import MultinomialNB

from sklearn.linear\_model import LogisticRegression

from sklearn.linear\_model import PassiveAggressiveClassifier

# Modelling Helpers

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

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn import metrics

# Computations

import itertools

# Visualization

import matplotlib.pyplot as plt

**Loading dataset**

train = pd.read\_csv("../input/fake-news/train.csv")

test = pd.read\_csv ("../input/fake-news/test.csv")

submit = pd.read\_csv ("../input/fake-news/submit.csv")

train.head()

print(f"Train Shape : **{**train.shape**}**")

print(f"Test Shape : **{**test.shape**}**")

print(f"Submit Shape : **{**submit.shape**}**")

output:

Train Shape : (20800, 5)

Test Shape : (5200, 4)

Submit Shape : (5200, 2)

**Data preprocessing**

train.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 20800 entries, 0 to 20799

Data columns (total 5 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 id 20800 non-null int64

1 title 20242 non-null object

2 author 18843 non-null object

3 text 20761 non-null object

4 label 20800 non-null int64

dtypes: int64(2), object(3)

memory usage: 812.6+ KB

train.isnull().sum()

Output:

id 0

title 558

author 1957

text 39

label 0

dtype: int64

train.dtypes.value\_counts()

Output:

object 3

int64 2

dtype: int64

**Handle missing values**

test=test.fillna(' ')

train=train.fillna(' ')

In:

*# Create a column with all the data available*

test['total']=test['title']+' '+test['author']+' '+test['text']

train['total']=train['title']+' '+train['author']+' '+train['text']

In:

*# Have a glance at our training set*

train.info()

train.head()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 20800 entries, 0 to 20799

Data columns (total 6 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 id 20800 non-null int64

1 title 20800 non-null object

2 author 20800 non-null object

3 text 20800 non-null object

4 label 20800 non-null int64

5 total 20800 non-null object

dtypes: int64(2), object(4)

memory usage: 975.1+ KB

output

| id | title | author | text | label | total |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | House Dem Aide: We Didn’t Even See Comey’s Let... | Darrell Lucus | House Dem Aide: We Didn’t Even See Comey’s Let... | 1 | House Dem Aide: We Didn’t Even See Comey’s Let... |
| 1 | 1 | FLYNN: Hillary Clinton, Big Woman on Campus - ... | Daniel J. Flynn | Ever get the feeling your life circles the rou... | 0 | FLYNN: Hillary Clinton, Big Woman on Campus - ... |
| 2 | 2 | Why the Truth Might Get You Fired | Consortiumnews.com | Why the Truth Might Get You Fired October 29, ... | 1 | Why the Truth Might Get You Fired Consortiumne... |
| 3 | 3 | 15 Civilians Killed In Single US Airstrike Hav... | Jessica Purkiss | Videos 15 Civilians Killed In Single US Airstr... | 1 | 15 Civilians Killed In Single US Airstrike Hav... |
| 4 | 4 | Iranian woman jailed for fictional unpublished... | Howard Portnoy | Print \nAn Iranian woman has been sentenced to... | 1 | Iranian woman jailed for fictional unpublished... |

*# Dividing the training set by using train\_test\_split*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(train['total'], train.label, test\_size=0.20, random\_state=0)

**Vectorizing our Data**

**Count Vectorizer**

*# Initialize the `count\_vectorizer`*

count\_vectorizer = CountVectorizer(ngram\_range=(1, 2), stop\_words='english')

*# Fit and transform the training data.*

count\_train = count\_vectorizer.fit\_transform(X\_train)

*# Transform the test set*

count\_test = count\_vectorizer.transform(X\_test)

**Tf-IDF Vectorizer**

*#Initialize the `tfidf\_vectorizer`*

tfidf\_vectorizer = TfidfVectorizer(stop\_words='english', ngram\_range=(1, 2))

*#Fit and transform the training data*

tfidf\_train = tfidf\_vectorizer.fit\_transform(X\_train)

*#Transform the test set*

tfidf\_test = tfidf\_vectorizer.transform(X\_test)

**Using Suitable ML models with Count Vectorizer and TF-IDF Vectorizer**

**Multinomial Naive Bayes with Count Vectorizer (BagofWords)**

nb\_classifier = MultinomialNB(alpha = 0.1)

nb\_classifier.fit(count\_train, y\_train)

pred\_nb\_count = nb\_classifier.predict(count\_test)

acc\_nb\_count = metrics.accuracy\_score(y\_test, pred\_nb\_count)

print(acc\_nb\_count)

0.942548076923077

Input:

*# tune* for alpha **in** np.arange(0,1,.05):

nb\_classifier\_tune = MultinomialNB(alpha=alpha)

nb\_classifier\_tune.fit(count\_train, y\_train)

pred\_tune = nb\_classifier\_tune.predict(count\_test)

*the hyperparameter alpha for the naive bayes classifier*

score = metrics.accuracy\_score(y\_test, pred\_tune)

print("Alpha: **{:.2f}** Score: **{:.5f}**".format(alpha, score))

Alpha: 0.00 Score: 0.91106

Alpha: 0.05 Score: 0.94183

Alpha: 0.10 Score: 0.94255

Alpha: 0.15 Score: 0.94279

Alpha: 0.20 Score: 0.94231

Alpha: 0.25 Score: 0.94111

Alpha: 0.30 Score: 0.93990

Alpha: 0.35 Score: 0.93822

Alpha: 0.40 Score: 0.93750

Alpha: 0.45 Score: 0.93630

Alpha: 0.50 Score: 0.93510

Alpha: 0.55 Score: 0.93486

Alpha: 0.60 Score: 0.93341

Alpha: 0.65 Score: 0.93149

Alpha: 0.70 Score: 0.93029

Alpha: 0.75 Score: 0.92909

Alpha: 0.80 Score: 0.92788

Alpha: 0.85 Score: 0.92644

Alpha: 0.90 Score: 0.92428

Alpha: 0.95 Score: 0.92308

The best score is obtained for alpha = 0.15, and is equal to 0.94279.

In [17]:

*# Let's re-run our fine-tuned model and plot the confusion matrix*

nb\_classifier = MultinomialNB(alpha = 0.15)

nb\_classifier.fit(count\_train, y\_train)

pred\_nb\_count = nb\_classifier.predict(count\_test)

cm = metrics.confusion\_matrix(y\_test, pred\_nb\_count, labels=[0,1])

*# Creating a function that outputs a confusion matrix*

def plot\_confusion\_matrix(cm, classes,

normalize=False,

title='Confusion matrix',

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)

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')

plot\_confusion\_matrix(cm, classes=['TRUE','FAKE'], title ='Confusion matrix for a MultinomialNB with Count Vectorizer')

**Multinomial Naive Bayes with TF-IDF Vectorizer**

nb\_classifier = MultinomialNB(alpha = 0.1)

nb\_classifier.fit(tfidf\_train, y\_train)

pred\_nb\_tfidf = nb\_classifier.predict(tfidf\_test)

acc\_nb\_tfidf = metrics.accuracy\_score(y\_test, pred\_nb\_tfidf)

print(acc\_nb\_tfidf)

0.926923076923077

In [19]:

*# tune the hyperparameter alpha for the naive bayes classifier*

for alpha **in** np.arange(0,0.1,.01):

nb\_classifier\_tune = MultinomialNB(alpha=alpha)

nb\_classifier\_tune.fit(tfidf\_train, y\_train)

pred\_tune = nb\_classifier\_tune.predict(tfidf\_test)

score = metrics.accuracy\_score(y\_test, pred\_tune)

print("Alpha: **{:.2f}** Score: **{:.5f}**".format(alpha, score))

Alpha: 0.00 Score: 0.90649

Alpha: 0.01 Score: 0.94375

Alpha: 0.02 Score: 0.94375

Alpha: 0.03 Score: 0.94159

Alpha: 0.04 Score: 0.93966

Alpha: 0.05 Score: 0.93822

Alpha: 0.06 Score: 0.93462

Alpha: 0.07 Score: 0.93317

Alpha: 0.08 Score: 0.93005

Alpha: 0.09 Score: 0.92861

In [20]:

*# Let's run the optimized model with best value of hyperparameter and check the confusion matrix*

nb\_classifier = MultinomialNB(alpha = 0.01)

nb\_classifier.fit(tfidf\_train, y\_train)

pred\_nb\_tfidf = nb\_classifier.predict(tfidf\_test)

cm2 = metrics.confusion\_matrix(y\_test, pred\_nb\_tfidf, labels=[0,1])

plot\_confusion\_matrix(cm2, classes=['TRUE','FAKE'], title ='Confusion matrix for a MultinomialNB with Tf-IDF')

conclusion:

Thus the loading and preprocessing of data was performed successfully.