**Analysis Of Algorithms**

**Report 2**

**Detecting Fake News on Social media**

**Group Member and Contribution:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Team member Name** | **Team member Email IDs** | **R# IDs** | **Contribution** |
| Vamsi Krishna Muppala | [vamuppal@ttu.edu](mailto:vamuppal@ttu.edu) | R11770853 | * Worked on the Report. * Done research on SVM and executed the method in the model. * Computed the F1 score and accuracy |
| Edwin Sahil Samuel Kathi | [ekathi@ttu.edu](mailto:ekathi@ttu.edu) | R11800320 |
| Yaswanth Kumar Chaganti | [ychagant@ttu.edu](mailto:ychagant@ttu.edu) | R11801037 | * Worked on the Report. * Done research on XGBoost and executed the method in the model. * Computed the confusion matrix |
| Jayakumar Kukkathotti | [jkukkath@ttu.edu](mailto:jkukkath@ttu.edu) | R11801622 |
| Spoorthy Gope Gowda | [sgopewgow@ttu.edu](mailto:sgopewgow@ttu.edu) | R11795421 | * Worked on the presentation. * Done research on future work. * Research work on the feature selections on Counts |
| Manoj Kumar Reddy | [manoredd@ttu.edu](mailto:manoredd@ttu.edu) | R11804056 |
| Rahul Bantu | [Rbantu@ttu.edu](mailto:Rbantu@ttu.edu) | R11798527 | * Worked on the presentation. * Read papers for future work. * Research work on the feature selections on TD-IDF |
| Thivatharan Jeganathan | [tjeganat@ttu.edu](mailto:tjeganat@ttu.edu) | R11845830 |

**Problem Statement:**

There is a lot of false information on social media. To detect it, designing a news monitor system that concentrates to alert the public about fake new is important.

Goals:

Identifying the fake news and guiding people to think about the false information.

Build a model which helps in predicting if the news is fake or real.

The text discusses the challenge of detecting fake news and the limitations of statistical approaches due to the lack of labeled benchmark datasets.

Dataset:

The dataset used is LIAR dataset, that can be used for fact-checking research and fake news detection. We clean the dataset to get the processed data which helps in detecting the fake ne

**Proposed work:**

**Cleaning Dataset:**

* We clean the raw dataset by eliminating punctuations, whitespaces, lowercases, stop words,

and lemmatized the tokens.

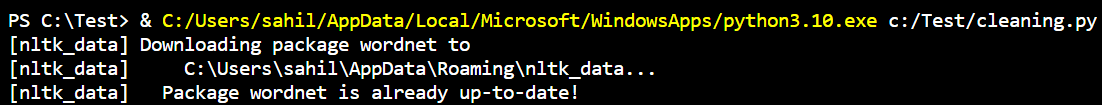
* After cleaning the data, we use the processed data to train the model.

**Training Model:**

* After preprocessing the data, we extract the features using Binary, Count Vectorizer, TFIDF
* For training the model we use different methodologies like Logistic Regression, SVM
* After selecting a methodology, we train the model using Scikit-Learn
* Now, we have calculated F1 score and confusion matrix for every methodology corresponding to every feature selection mentioned above.
* Based on the above results on F1 score and confusion matrix we select the best model and corresponding feature selection to train the model.

**Experimental Results:**

Cleaning:

****

Training

(i) Feature selection: TF-IDF (Models F1 score and confusion matrix)

(a) Logistic Regression

**Graphical user interface

Description automatically generated**

(b)XGBoost:

**Graphical user interface

Description automatically generated**

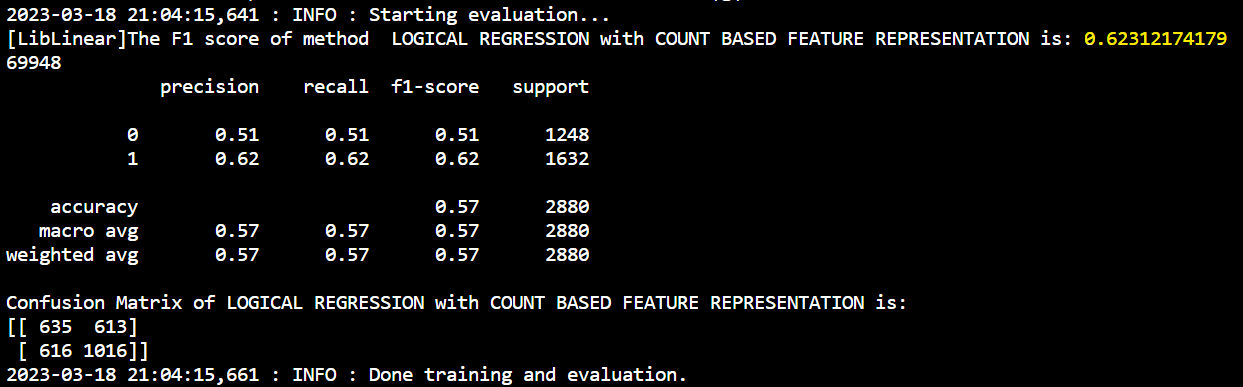
(c)SVM

**Graphical user interface

Description automatically generated**

(ii)Feature selection: Counts (Models F1 score and confusion matrix)

(a) Logistic Regression:

****

(b)XGBoost:

**Graphical user interface

Description automatically generated**

(c)SVM:

**Graphical user interface

Description automatically generated**

(iii)Feature selection: Binary (Models F1 score and confusion matrix)

(a) Logistic Regression:

**A screen shot of a computer

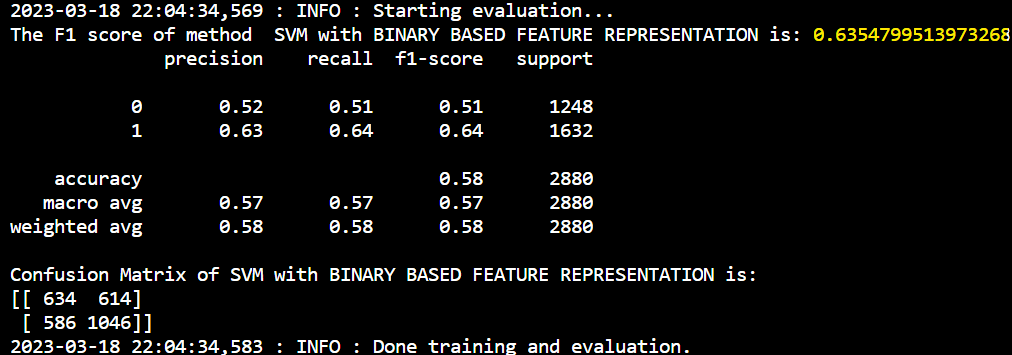
Description automatically generated with low confidence**

(b)XGBoost:

**Graphical user interface, text, application

Description automatically generated**

(c)SVM:

****

**Observation:**

From the above results we can see that the best F1 score and accuracy has been observed from SVM - TF-IDF model and the values are 0.67 and 0.61 respectively.

**Future Work:**

After researching a few papers we can get better results from Naïve Bayes and few other machine learning models.

**Pseudo Code:**

* **Cleaning the data**

import seaborn as sn # Import the seaborn library for data visualization

import re # Import the re library for regular expressions

import string # Import the string library for string operations

import nltk # Import the nltk library for natural language processing

from nltk.corpus import stopwords # Import the stopwords module from nltk corpus

from nltk.stem import WordNetLemmatizer # Import the WordNetLemmatizer module from nltk stemmer

nltk.download('wordnet') # Download the WordNetLemmatizer data

# Define a function to create a countplot of the label column in a given DataFrame

def create\_distribution(dataFile):

return sn.countplot(x='label', data=dataFile, palette='hls')

# Define a function to process a given text by removing non-alphabetic characters, punctuation, stopwords and lemmatizing the words

def process\_text(text):

text = re.sub(r'[^a-zA-Z\s]', '', text, re.I|re.A) # Remove non-alphabetic characters

nopunc = [char for char in text if char not in string.punctuation] # Remove punctuation

nopunc = ''.join(nopunc) # Convert list of characters back to string

clean\_words = [word for word in nopunc.split() if word.lower() not in stopwords.words('english')] # Remove stopwords

lemmatizer = WordNetLemmatizer() # Initialize the WordNetLemmatizer object

lemmatized\_words = [lemmatizer.lemmatize(word) for word in clean\_words if (len(word) > 2 and len(word) < 14)] # Lemmatize the words

return lemmatized\_words

# Define a function to join a list of words into a single string

def process\_text2(text):

text1 = " ".join(text)

return text1

# Define a function to map the label column in the given train, test and validation datasets to true/false values based on a dictionary

def map\_lables(train, test, val):

labels\_dict = {'true': 'true','mostly-true': 'true',

'half-true':'true', 'false':'false',

'barely-true':'false','pants-fire':'false'} # Define a dictionary to map labels

train= train.replace({"label": labels\_dict})[['label','statement']] # Replace the label column values in train dataset using the dictionary and keep only 'label' and 'statement' columns

test = test.replace({"label": labels\_dict})[['label','statement']] # Replace the label column values in test dataset using the dictionary and keep only 'label' and 'statement' columns

val = val.replace({"label": labels\_dict})[['label','statement']] # Replace the label column values in validation dataset using the dictionary and keep only 'label' and 'statement' columns

return train, test, val # Return the updated train, test and validation datasets

* **Training the data**

// Import necessary dependencies and settings

// Define the function to extract features for all models

function extract\_features(field, training\_data, testing\_data, type) {

logging.info("Extracting features and creating vocabulary...")

if "binary" in type {

// BINARY FEATURE REPRESENTATION

cv= CountVectorizer(binary=True, max\_df=0.95, analyzer=process\_text)

cv.fit\_transform(training\_data.values)

}

else if "counts" in type {

// COUNT BASED FEATURE REPRESENTATION

cv= CountVectorizer(binary=False, max\_df=0.95, analyzer=process\_text)

cv.fit\_transform(training\_data.values)

}

else {

// TF-IDF BASED FEATURE REPRESENTATION

tfidf\_vectorizer=TfidfVectorizer(use\_idf=True, max\_df=0.95, analyzer=process\_text)

tfidf\_vectorizer.fit\_transform(training\_data.values)

}

}

// Define the function to train the model for all methods

function train\_model(classifier, train\_val, field, feature\_rep, name) {

logging.info("Starting model training...")

// GET A TRAIN TEST SPLIT (set seed for consistent results)

training\_data, testing\_data = train\_test\_split(train\_val, random\_state=2000)

// Get features

X\_train=training\_data['statement']

X\_test=testing\_data['statement']

// Get labels

Y\_train=training\_data['label'].values

Y\_test=testing\_data['label'].values

// GET FEATURES

train\_features,test\_features,feature\_transformer=extract\_features(field, X\_train, X\_test, type=feature\_rep)

logging.info("Training a Classification Model...")

model=classifier.fit(train\_features,Y\_train)

// GET PREDICTIONS

predictions = model.predict(test\_features)

// GET EVALUATION NUMBERS ON TEST SET

logging.info("Starting evaluation...")

score = f1\_score(Y\_test,predictions)

// print F1 score and confusion matrix

return model, feature\_transformer, score

}

// Check if the code is being run as the main program

if the current module is the main module:

// load the processed data

load the train data from "C:\\AOA\_proj\\Detecting-Fake-News-On-Social-Media-main\\data\\processed\\train.csv"

load the validation data from "C:\\AOA\_proj\\Detecting-Fake-News-On-Social-Media-main\\data\\processed\\val.csv"

// define the path for model and feature transformer

set the model path as "C:\\AOA\_proj\\Detecting-Fake-News-On-Social-Media-main\\models\\final\_model.pkl"

set the transformer path as "C:\\AOA\_proj\\Detecting-Fake-News-On-Social-Media-main\\models\\transformer.pkl"

#Merging the training and validation data together to train the model

encode the label values using LabelEncoder()

concatenate train and validation data

count the label values

transform the label values

train\_val = pd.concat(frames)

train\_val['label'] = labelEncoder.fit\_transform(train\_val['label'])

// Logistic Regression training model

set the field as "statement"

set the Logistic Regression model parameters

train the model using the train\_model() function and obtain the trained model, transformer, and score.

LogR\_clf = LogisticRegression(verbose=1, solver='liblinear',random\_state=0, C=5, penalty='l2',max\_iter=1000)

lr\_model,transform,score = train\_model(LogR\_clf,train\_val,field="statement",feature\_rep="",name='LOGICAL REGRESSION')

// XGBoost training model

set the field as "statement"

set the XGBoost model parameters

train the model using the train\_model() function and obtain the trained model, transformer, and score.

XGBoost\_clf = XGBClassifier(max\_depth=15, learning\_rate=0.3, n\_estimators=150)

XGBoost\_model,transform,score = train\_model(XGBoost\_clf,train\_val,field="statement",feature\_rep="",name='XGBoost')

// SVM training model

set the field as "statement"

set the SVM model parameters

train the model using the train\_model() function and obtain the trained model, transformer, and score.

svm\_clf = svm.SVC(kernel='linear')

svm\_model,transform,score = train\_model(svm\_clf,train\_val,field="statement",feature\_rep="",name='SVM')