Center for Computational Engineering and Networking

Amrita Vishwa Vidyapeetham

2021-2023

A

Term Project Report On

"MULTIMEDIA MISOGYNY IDENTIFICATION USING ML ALGORITHMS"

(SVM and Random Forest)

Ist year M.Tech-Data Science(CEN)

Batch(2021-23),Course:21DS602(21-22(Odd))

Date:31st January 2022

Submitted

By

KATIPALLY VIGHNESHWAR REDDY CB.EN.P2DSC21012

ABSTRACT

We present a benchmark dataset generated as part of a project for automatic identification of misogyny within online content, Which focuses in particular on memes. The Benchmark here described is composed of 10,100 memes collected from the most popular social media platforms, such as Facebook, Twitter, Instagram and Reddit, and consulting websites dedicated to the collection and creation of memes. To gather misogynistic memes, specific keywords that refer to misogynistic content have been considered as a search criterion, considering different manifestations of hatred against women, such as body shaming, stereotyping, objectification and violence. In parallel, memes with no misogynist content have been manually downloaded from the same web sources. Among all the collected memes, three-domain experts have selected a dataset of 10,100 memes equally balanced between misogynistic and non-misogynistic ones. Finally, for each meme, the text has been manually transcribed. The dataset provided is thus composed of the 10,100 memes, the labels given by the experts and those obtained by the crowdsourcing validation, and the transcribed texts. This data can be used to approach the problem of automatic detection of misogynistic content on the Web relying on both textual and visual cues, facing phenomenons that are growing every day such as cyber sexism and technology-facilitated violence. By using machine learning techniques, we finally segregate the misogyny memes with non-misogyny memes.

Contents

List of Figures

Chapter 1

INTRODUCTION

Women have a strong presence online, particularly in image-based social media such as Twitter and Instagram: 78 of women use social media multiple times per day compared to 65 of men. However, while new opportunities for women have been opened on the Web, systematic inequality and discrimination offline is replicated in online spaces in the form of offensive contents against them. Popular communication tools in social media platforms are Memes. A meme is essentially an image characterized by a pictorial content with an overlaying text a posterior introduced by human, with the main goal of being funny and/or ironic. Although most of them are created with the intent of making funny jokes, in a short time people started to use them as a form of hate against women, landing to sexist and aggressive messages in online environments that subsequently amplify the sexual stereotyping and gender inequality of the offline world. The proposed task, i.e. Multimedia Automatic Misogyny Identification (MAMI) consists in the identification of misogynous memes, taking advantage of both text and images available as source of information.
Task:A basic task about misogynous meme identification, where a meme should be categorized either as misogynous or not misogynous

1.1 Literature Review

In this article we survey approaches proposed in the literature to solve the problem of misogynistic memes recognition. [2]They tried to solve the problem of misogynistic text recognition so we thought of extending their ideas by including images also.These include classical machine learning models like Support Vector Machine, Naive Bayes, Logistic Regression and ensembles of different classical machine learning models they consider results of experiments with these models in different languages: English, Spanish and Italian tweets. In this survey they described How some featureshelped to identify misogynistic memes.The survey includesnot only-models-which help-to-identify misogyny but also systems which help-to recognize a target of an offense (an individual or a group of persons).

1.2 Problem Statement

We Should be abel to Distinguish misogyny post which is uploaded in mini Blogging platform, and able to take required actions against the posted post.

1.3 Objectives

- To perform automatic misogyny identification in both Text and Images (multi-modal) Existing solution uses only Text data.
- To Train the model using NLP and machine learning algorithms(SVM and Random Forest)in a supervised setting to detect online misogyny identification.

Chapter 2

Theoretical background

2.1 What is Classification?

Classification is a supervised learning approach that classifies some unknown items into a specific set of classes

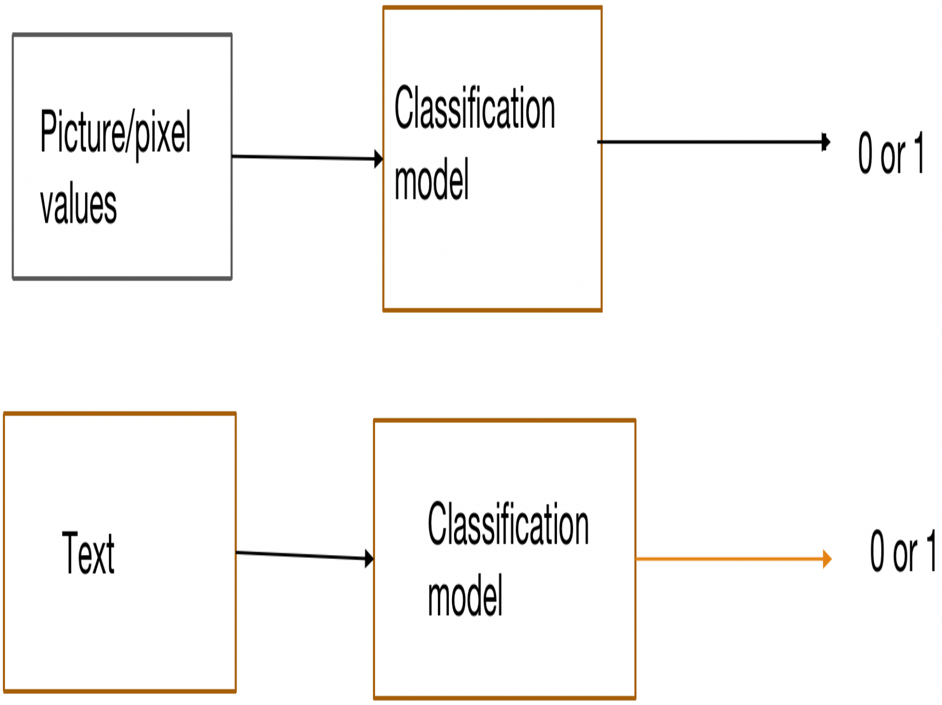


Figure 2.1: Classification

2.2 What is Evaluation for Classification?

Confusion matrix A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model

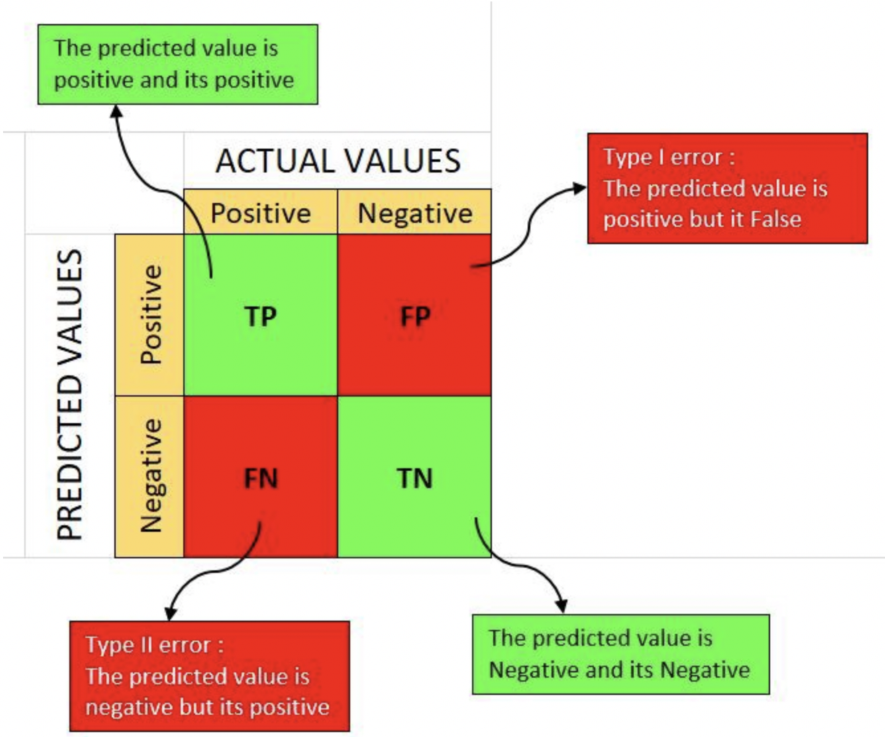


Figure 2.2: Classification

2.2.1 Calculated Metrics using Confusion matrix ?

1. **Accuracy**

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

Figure 2.3: Accuracy

Accuracy simply measures how often the classifier makes the correct prediction. It's the ratio between the number of correct predictions and the total number of predictions.

2. **Precision**

   It is a measure of correctness that is achieved in true prediction. In simple words, it tells us how many predictions are actually positive out of all the total postive predicted.

3. **Recall**

   It is a measure of actual observations which are predicted correctly, i.e. how many observations of positive class are actually predicted as positive. It is also known as Sensitivity. Recall is a

$$\text{Precision} = \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False Positive}}$$

Figure 2.4: Precision

$$\text{Recall} = \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False Negative}}$$

Figure 2.5: Recall

valid choice of evaluation metric when we want to capture as many positives as possible.

4. **F1-score** The F1 score is a number between 0 and 1 and is the harmonic mean of precision

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

Figure 2.6: F1-Score

and recall. We use harmonic mean because it is not sensitive to extremely large values, unlike simple averages.

2.3 What Machine learning Algorithms in Classification used ?

SVM

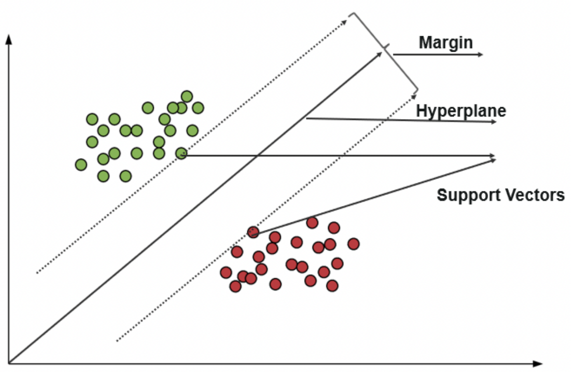


Figure 2.7: SVM

Support Vector Machine" (SVM) is a supervised machine learning algorithm that can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.

SVM Kernels

1. **Linear Kernel** – A linear kernel can be used as a normal dot product between any two given observations. The product between the two vectors is the sum of the multiplication of each pair of input values. Following is the linear kernel equation.

$$f(x) = B(0) + sum(ai * (x, xi))$$

Figure 2.8: LINEAR KERNEL

2. **Polynomial Kernel** – It is a rather generalized form of the linear kernel. It can distinguish curved or nonlinear input space. Following is the polynomial kernel equation.

$$K(X_1, X_2) = (a + X_1^T X_2)^b$$

b = degree of kernel & a = constant term.

Figure 2.9: POLYNOMIAL KERNEL

3. **Radial Basis Function Kernel**– The radial basis function kernel is commonly used in SVM classification, it can map the space in infinite dimensions. Following is the RBF kernel equation.

Random Forest

$$K(X_1, X_2) = exponent(-\gamma\|X_1 - X_2\|^2)$$

$$\|X1 - X2\| = \text{Euclidean distance between } X1 \,\&\, X2$$

Figure 2.10: Radial Basis Function Kernel

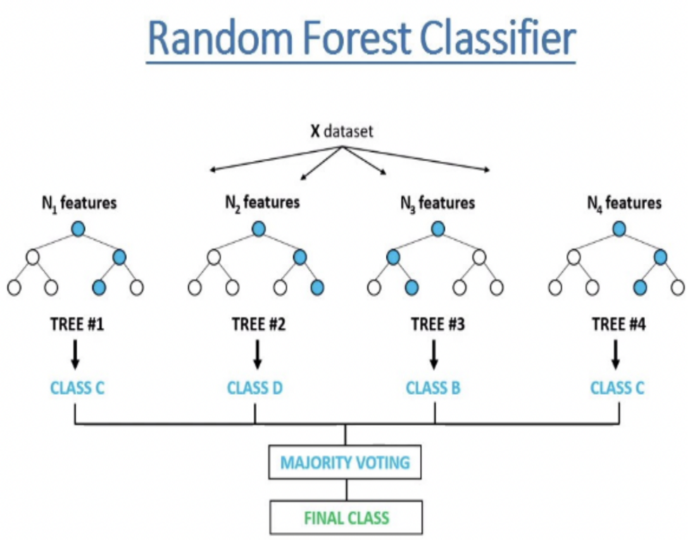


Figure 2.11: Random Forest

Random Forest is the most used supervised machine learning algorithm for classification. Generally it constructs a N number of decision tress at training time.
Random Forest is trained with Bagging method Uses Ensemble learning method.

Ensemble Learning:

Method in which the predictions are based on the combined results of various individual models.
Methods:

1. **Bagging** The process where training bunch of individual models parallelly , and each model is trained by a random subset of the data.
 MATH of Bagging algorithm:

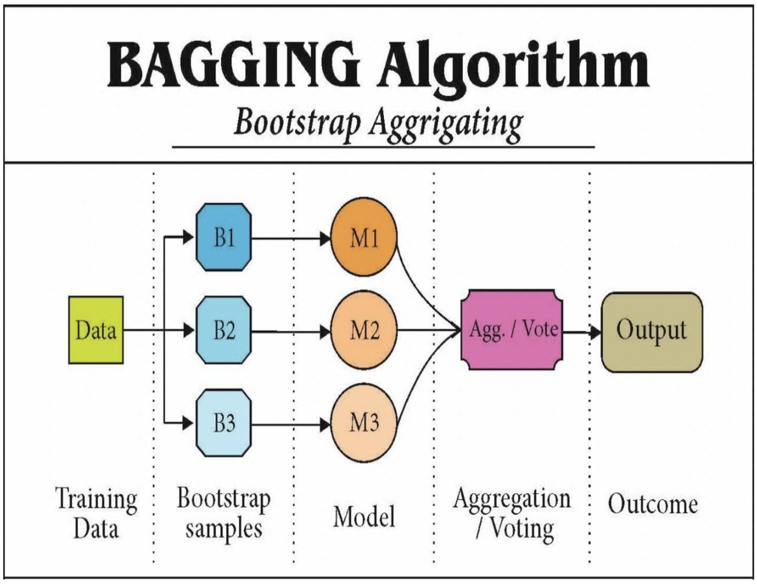


Figure 2.12: BAGGING

As there is limitation of having a large training set, so we take repeated Bootstrap samples (SB) with replacement from the same training set (S (yn ; xn), n = 1,. . . ., N, where y is either a class or numerical target response.

The main objective is to use (SB) to get a better predictor than the single training set predictor ø (x, SB).

When y is numerical, we take average of ø (x, SB),

øB (x) = avg B ø (x, SB).

When y is a class label, we take plurality vote of ø (x, SB) to form øB (x).

When we want to reduce the Variance in the model, we can use Bagging.

2. **Boosting** The process where training bunch of individual models in a sequential way.Each individual model learn from mistakes made by the previous model.
MATH of Boosting algorithm:

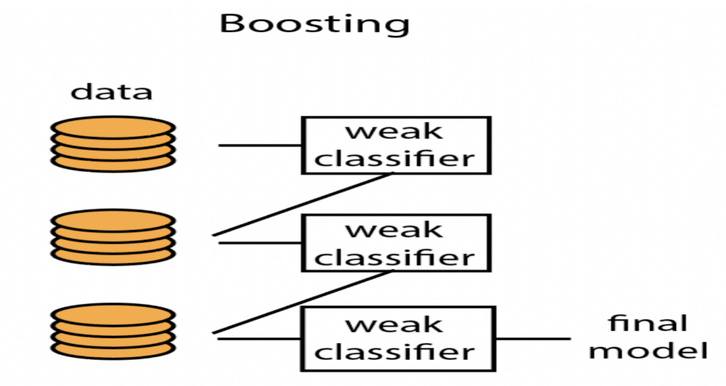
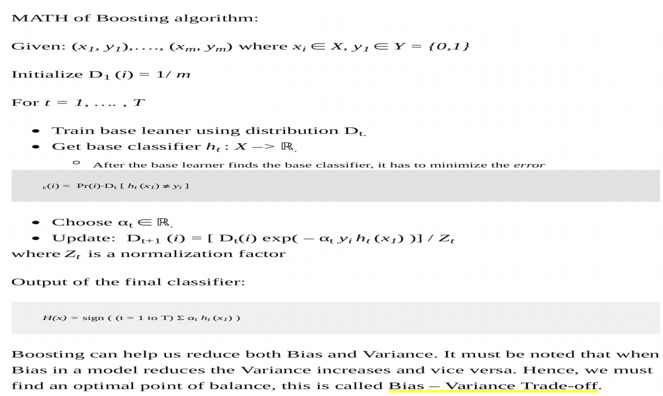


Figure 2.13: BOOSTING



MATH of Boosting algorithm:

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_1 \in Y = \{0,1\}$

Initialize $D_1(i) = 1/m$

For $t = 1, \ldots, T$

- Train base leaner using distribution D_t.
- Get base classifier $h_t : X \rightarrow \mathbb{R}$.
  - After the base learner finds the base classifier, it has to minimize the *error*

$\varepsilon(i) = \Pr(i) \sim D_t [\, h_t(x_1) \neq y_i \,]$

- Choose $\alpha_t \in \mathbb{R}$.
- Update: $D_{t+1}(i) = [\, D_t(i) \exp(-\alpha_t y_i h_t(x_1)) \,] / Z_t$

where Z_t is a normalization factor

Output of the final classifier:

$H(x) = \text{sign}\,(\,(t = 1 \text{ to } T)\ \Sigma\ \alpha_t h_t(x_1)\,)$

Boosting can help us reduce both Bias and Variance. It must be noted that when Bias in a model reduces the Variance increases and vice versa. Hence, we must find an optimal point of balance, this is called Bias – Variance Trade-off.

Figure 2.14: MATH

Chapter 3

Work Flow

3.1 Flow Chart

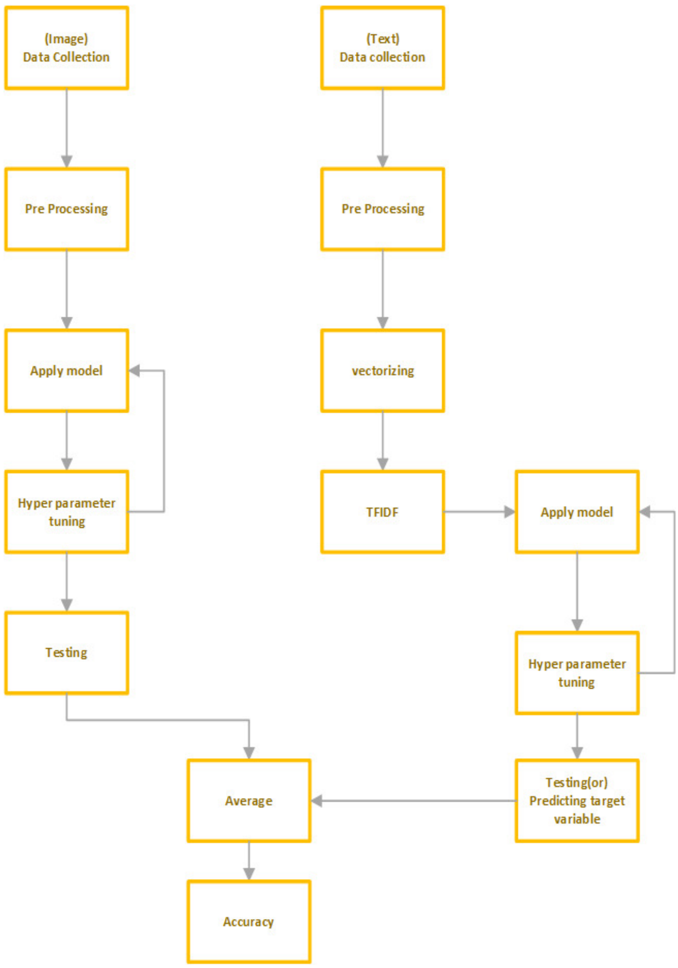


Figure 3.1: FlowChart

3.1.1 Algorithm for Text Data

1. From the data set consisting of10,000 different memes taking it as training set and 7 features.
2. But we have taken misogyny column as it is the target variable.
3. Considering the test transcription.

4. Then feature generation takes place
5. From this we do pre processing.
6. We use grid search cv to get best performance out of it .
7. Apply the best hperparameters to train the model.
8. Then we can apply Test data to predict the Outcome.

3.1.2 Algorithm for Image Data

1. Extracted files are kept in csv file.
2. We append the csv file (image) and test transcription data
3. mages are random so we compare the file list with file name and try to equate and find whether it is misogyny or not for both ytrain and ytest.
4. Then used first 2000 image sampels as Train Data and we converted image to grayscale and we took pixels of image into features .
5. Later found some imbalance in data so, we use SMOTE to balance the data. //SOMTE is upper sampaling method
6. Then applyed the ml algorithms
7. Finally merging the test and image accuracy.

Chapter 4

Program's

4.1 RANDOM FOREST

```

from google.colab import drive
drive.mount('/content/gdrive')

!unzip gdrive/My\ Drive/TRAINING.zip

!unzip -P *MaMiSemEval2022! gdrive/My\ Drive/trial.zip

!unzip gdrive/My\ Drive/train_pixel1.csv.zip

#!ls '/content/drive/MyDrive/mami_data'

# import zipfile

# zip_file = "/content/drive/MyDrive/mami_data/training.zip"

# try:
#     with zipfile.ZipFile(zip_file) as z:
#         z.extractall("/content/drive/MyDrive/mami_data/training")
#         print("Extracted all")
# except:
#     print("Invalid file")

#!ls '/content/drive/MyDrive/mami_data/training/TRAINING/training.csv'

# import zipfile

# zip_file = "/content/drive/MyDrive/mami_data/trial.zip"

# try:
#     with zipfile.ZipFile(zip_file) as z:
#         z.setpassword(pwd = bytes('*MaMiSemEval2022!', 'utf-8'))
#         z.extractall("/content/drive/MyDrive/mami_data/trial")
```

```
#           print("Extracted all")
# except:
#       print("Invalid file")

!ls '/content/drive/MyDrive/mami_data/trial/Users/fersiniel/Desktop/MAMI - TO LABEL/TRIAL DATASET'

#import zipfile

#zip_file = "/content/drive/MyDrive/mami_data/test.zip"

#try:
#    with zipfile.ZipFile(zip_file) as z:
#         z.setpassword(pwd = bytes('*MaMiSemEval2022!', 'utf-8'))
#          z.extractall("/content/drive/MyDrive/mami_data/test")
#          print("Extracted all")
#except:
#      print("Invalid file")

!ls '/content/drive/MyDrive/mami_data/test/test/Test.csv'

#!cp '/content/drive/MyDrive/mami_data/trial/Users/fersiniel/Desktop/MAMI - TO LABEL/TEST DATASET/test.csv' '/content/drive/MyDrive/mami_data/test'

!cp '/content/drive/MyDrive/mami_data/trial/Users/fersiniel/Desktop/MAMI - TO LABEL/TRIAL DATASET/trial.csv' '/content/drive/MyDrive/mami_data/trial'

//Applying preprocessing with cleaningg tokenizing and lematization

from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import GridSearchCV
import pandas as pd
import seaborn as sns
data1=pd.read_csv('TRAINING/training.csv', sep='\t')
data2=pd.read_csv("trial.csv",sep='\t')
#training_label=data['misogynous']
# print(training_data.head())
import re
import nltk
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
stop_words=stopwords.words('english')
stop_words.append('imgflipcom')
stop_words.append('zip')
print(stop_words)
```

```
lemmatizer=WordNetLemmatizer()
#training data
for index,row in data1.iterrows():
  #print(row)
  filter_sentence=[]
  sentence=row['Text Transcription']
  sentence = sentence.lower()
  #print(sentence)
  sentence=re.sub(r'[^\w\s]','',sentence)#cleaning
  words=nltk.word_tokenize(sentence)
  words=[w for w in words if not w in stop_words]
  for word in words:
    filter_sentence.append(lemmatizer.lemmatize(word))
  #print(filter_sentence)
  listToStr = ' '.join([str(elem) for elem in filter_sentence])
  data1.loc[index,"Text Transcription"]=listToStr
#trail data
for index,row in data2.iterrows():
  #print(row)
    filter_sentence=[]
    sentence=row['Text Transcription']
    sentence = sentence.lower()
    #print(sentence)
    sentence=re.sub(r'[^\w\s]','',sentence)#cleaning
    words=nltk.word_tokenize(sentence)
    words=[w for w in words if not w in stop_words]
    for word in words:
        filter_sentence.append(lemmatizer.lemmatize(word))
    #print(filter_sentence)
    listToStr = ' '.join([str(elem) for elem in filter_sentence])
    data2.loc[index,"Text Transcription"]=listToStr
print(data1.head())
print(data1.shape)
print(data2.head())
print(data2.shape)
#data=pd.concat([data1,data2])
#print(data.head())
#print(data.shape)

//applying countvectorizer

from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()
training_data=data1['Text Transcription']
training_label=data1['misogynous']
X_train_counts = count_vect.fit_transform(training_data)
X_train_counts.shape


//TF-IDF transormer
```

```

from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
X_train_tfidf.shape
print(X_train_tfidf)


from sklearn.pipeline import Pipeline
import numpy as np
from sklearn import metrics
from sklearn.model_selection import cross_val_score
sns.countplot(training_label)
text_clf = Pipeline([
      ('vect', CountVectorizer()),
      ('tfidf', TfidfTransformer()),
      ('clf',RandomForestClassifier(max_depth=2, random_state=0)),
 ])
scores=cross_val_score(text_clf,training_data,training_label,cv=5,scoring='accuracy')
print(scores)
print(scores.mean())


//To find best parameter we need tuning hyper parameter using grid search cv



from sklearn.model_selection import GridSearchCV
param_grid = { 'bootstrap': [True], 'max_depth': [5, 10, None], 'max_features': ['auto', 'log2'],'n_estimators': [5, 6, 7, 8]}
print(param_grid)
clf=RandomForestClassifier()
grid=GridSearchCV(clf,param_grid,cv=3,scoring='accuracy',n_jobs=-1)
grid.fit(X_train_tfidf,training_label)
#grid.grid_scores_
grid.cv_results_
means = grid.cv_results_['mean_test_score']
params = grid.cv_results_['params']
for mean,param in zip(means,params):
     print("%f with: %r" % (mean,param))
grid.mean_scores=means
print(grid.best_score_)
print(grid.best_params_)
print(grid.best_estimator_)

//validation accuracy for training data
clf=RandomForestClassifier(bootstrap= True, max_depth=None, max_features='auto', n_estimators= 8)
scores=cross_val_score(clf,X_train_tfidf,training_label,cv=3,scoring='accuracy')
```

```
print(scores)
print(scores.mean())

//testing accuracy
X_train=X_train_tfidf
X_test=training_label
y_train=data2['Text Transcription']
y_test=data2['misogynous']
text_clf = Pipeline([
     ('vect', CountVectorizer()),
     ('tfidf', TfidfTransformer()),
     ('clf',RandomForestClassifier(bootstrap= True, max_depth=None, max_features
        ='auto', n_estimators= 8)),
 ])
text_clf.fit(training_data, training_label)
y_pred = text_clf.predict(y_train)
metrics.accuracy_score(y_test,y_pred)
metrics.confusion_matrix(y_test, y_pred)
print(metrics.classification_report(y_test,y_pred
     ))

from google.colab import drive
drive.mount('/content/drive')

#!unzip /content/drive/MyDrive/mami_data/train_pixel1.zip


#!ls /content/drive/MyDrive/mami_data/training/TRAINING/

#!unzip -P *MaMiSemEval2022! /content/drive/MyDrive/mami_data/training/TRAINING/
    trial.zip


# !unzip -P *MaMiSemEval2022! /content/drive/MyDrive/mami_data/training/TRAINING
    /test.zip

first creating csv file with feature extraction from images and reading csv file

#import glob
import pandas as pd
#import numpy as np
from keras.preprocessing.image import img_to_array,array_to_img,load_img
train_image = pd.read_csv('train_pixel1.csv',index_col=0)
print(train_image.shape)
test_image = pd.read_csv('test_pixel.csv',index_col=0)
print(test_image.shape)


//reading training and testing datasets
import pandas as pd
```

```
total_data_train=pd.read_csv('training.csv',sep='\t')
total_data_test=pd.read_csv('trial.csv',sep='\t')
print(total_data_train['file_name'])

//y_train is extracting from training dataset by checking filename and file_list
import os
file_list=os.listdir(r"./TRANING/")
#print(file_list)
y_train=[]
for j in range(0,2001):
  for i in range(len(total_data_train)):
      if(file_list[j]==(total_data_train['file_name'][i])):
        y_train.append(total_data_train['misogynous'][i])
      else:
        continue
print(len(y_train))

//y_test is extracting from training dataset by checking filename and file_list
file_list2=os.listdir(r"/Users/fersiniel/Desktop/MAMI - TO LABEL/TRIAL DATASET/"
    )
y_test=[]
for j in file_list2:
  for i in range(len(total_data_test)):
      if(j==(total_data_test['file_name'][i])):
        y_test.append(total_data_test['misogynous'][i])
      else:
        continue

X_train=train_image#train_image is a feature extraction from image dataset
X_test=test_image#train_image is a feature extraction from image dataset

from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state=42)
X_train, y_train = sm.fit_resample(X_train, y_train)

//Random Forest algorithm is applied to model

from sklearn.ensemble import RandomForestClassifier
import numpy as np
import seaborn as sns
from sklearn import metrics
from sklearn.model_selection import cross_val_score
sns.countplot(y_train)
clf=RandomForestClassifier(max_depth=2, random_state=0)
scores=cross_val_score(clf,X_train,y_train,cv=3,scoring='accuracy')
print(scores)
print(len(X_train))
print(scores.mean())
```

```

from sklearn.model_selection import GridSearchCV
param_grid = { 'bootstrap': [True], 'max_depth': [5, 10, None], 'max_features':
    ['auto', 'log2'],'n_estimators': [5, 6, 7, 8]}
print(param_grid)
clf=RandomForestClassifier( )
grid=GridSearchCV(clf,param_grid,cv=2,scoring='accuracy',n_jobs=-1)
grid.fit(X_train,y_train)
#grid.grid_scores_
grid.cv_results_
means = grid.cv_results_['mean_test_score']
params = grid.cv_results_['params']
#for mean,param in zip(means,params):
#print("%f  with:  %r" % (mean,param))
grid.mean_scores=means

print(grid.best_score_)
print(grid.best_params_)
print(grid.best_estimator_)

clf=RandomForestClassifier(bootstrap= True, max_depth= None, max_features= 'log2
    ', n_estimators= 8)
scores=cross_val_score(clf,X_train,y_train,cv=2,scoring='accuracy')
print(scores)
print(scores.mean())


clf=RandomForestClassifier(bootstrap= True, max_depth= None, max_features= 'log2
    ', n_estimators= 8)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(metrics.classification_report(y_test,y_pred))
```

4.2 SVM

```
from google.colab import drive
drive.mount('/content/gdrive')

!unzip gdrive/My\ Drive/TRAINING.zip

!unzip -P *MaMiSemEval2022! gdrive/My\ Drive/trial.zip

!unzip gdrive/My\ Drive/train_pixel1.csv.zip

#!ls '/content/drive/MyDrive/mami_data'

```

```
# import zipfile

# zip_file = "/content/drive/MyDrive/mami_data/training.zip"

# try:
#     with zipfile.ZipFile(zip_file) as z:
#         z.extractall("/content/drive/MyDrive/mami_data/training")
#         print("Extracted all")
# except:
#     print("Invalid file")

!ls '/content/drive/MyDrive/mami_data/training/TRAINING/training.csv'

# import zipfile

# zip_file = "/content/drive/MyDrive/mami_data/trial.zip"

# try:
#     with zipfile.ZipFile(zip_file) as z:
#         z.setpassword(pwd = bytes('*MaMiSemEval2022!', 'utf-8'))
#         z.extractall("/content/drive/MyDrive/mami_data/trial")
#         print("Extracted all")
# except:
#     print("Invalid file")

!ls '/content/drive/MyDrive/mami_data/trial/Users/fersiniel/Desktop/MAMI - TO LABEL/TRIAL DATASET'

#import zipfile

#zip_file = "/content/drive/MyDrive/mami_data/test.zip"

#try:
#   with zipfile.ZipFile(zip_file) as z:
#       z.setpassword(pwd = bytes('*MaMiSemEval2022!', 'utf-8'))
#         z.extractall("/content/drive/MyDrive/mami_data/test")
#         print("Extracted all")
#except:
#     print("Invalid file")

!ls '/content/drive/MyDrive/mami_data/test/test/Test.csv'

#!cp '/content/drive/MyDrive/mami_data/trial/Users/fersiniel/Desktop/MAMI - TO LABEL/TEST DATASET/test.csv' '/content/drive/MyDrive/mami_data/test'

!cp '/content/drive/MyDrive/mami_data/trial/Users/fersiniel/Desktop/MAMI - TO LABEL/TRIAL DATASET/trial.csv' '/content/drive/MyDrive/mami_data/trial'

//Applying preprocessing with cleaningg tokenizing and lematization
```

```
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import GridSearchCV
import pandas as pd
import seaborn as sns
data1=pd.read_csv('training.csv', sep='\t')
data2=pd.read_csv("trial.csv",sep='\t')
#training_label=data['misogynous']
# print(training_data.head())
import re
import nltk
nltk.download('punkt')#sentence tokenizer
nltk.download('wordnet')#is another nltk corpus reader
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
stop_words=stopwords.words('english')
stop_words.append('imgflipcom')
stop_words.append('zip')
print(stop_words)
lemmatizer=WordNetLemmatizer()
#training data
for index,row in data1.iterrows():
  #print(row)
  filter_sentence=[]
  sentence=row['Text Transcription']
  sentence = sentence.lower()
  #print(sentence)
  sentence=re.sub(r'[^\w\s]','',sentence)#cleaning
  words=nltk.word_tokenize(sentence)
  words=[w for w in words if not w in stop_words]
  for word in words:
    filter_sentence.append(lemmatizer.lemmatize(word))
  #print(filter_sentence)
  listToStr = ' '.join([str(elem) for elem in filter_sentence])
  data1.loc[index,"Text Transcription"]=listToStr
#trail data
for index,row in data2.iterrows():
  #print(row)
    filter_sentence=[]
    sentence=row['Text Transcription']
    sentence = sentence.lower()
    #print(sentence)
    sentence=re.sub(r'[^\w\s]','',sentence)#cleaning
    words=nltk.word_tokenize(sentence)
    words=[w for w in words if not w in stop_words]
    for word in words:
        filter_sentence.append(lemmatizer.lemmatize(word))
    #print(filter_sentence)
```

```
    listToStr = ' '.join([str(elem) for elem in filter_sentence])
    data2.loc[index,"Text Transcription"]=listToStr
print(data1.head())
print(data1.shape)
print(data2.head())
print(data2.shape)
#data=pd.concat([data1,data2])
#print(data.head())
#print(data.shape)

//applying countvectorizer

from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()
training_data=data1['Text Transcription']
training_label=data1['misogynous']
X_train_counts = count_vect.fit_transform(training_data)
X_train_counts.shape


//TF-IDF transormer

from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
X_train_tfidf.shape
print(X_train_tfidf)


from sklearn.pipeline import Pipeline
import numpy as np
from sklearn import metrics
from sklearn.model_selection import cross_val_score
sns.countplot(training_label)
text_clf = Pipeline([
      ('vect', CountVectorizer()),
      ('tfidf', TfidfTransformer()),
      ('clf',SVC(gamma='auto')),
 ])
scores=cross_val_score(text_clf,training_data,training_label,cv=10,scoring='
    accuracy')
print(scores)
print(scores.mean())


//To find best parameter we need tuning hyper parameter using Gridsearchcv



from sklearn.model_selection import GridSearchCV
```

```
param_grid = {'C': [0.1, 1, 10],# control parameter of error
              'gamma': [1, 0.1, 0.01],#curvature shape
              'kernel': ['rbf']}#only stores support vector and  radial basis
                 function
print(param_grid)
clf=SVC()
grid=GridSearchCV(clf,param_grid,cv=3,scoring='accuracy',n_jobs=-1)
grid.fit(X_train_tfidf,training_label)
#grid.grid_scores_
grid.cv_results_
means = grid.cv_results_['mean_test_score']
params = grid.cv_results_['params']
for mean,param in zip(means,params):
    print("%f  with:  %r" % (mean,param))
grid.mean_scores=means
print(grid.best_score_)
print(grid.best_params_)
print(grid.best_estimator_)

#validation accuracy for training data
clf=SVC(C=1, gamma=1)
scores=cross_val_score(clf,X_train_tfidf,training_label,cv=3,scoring='accuracy')
print(scores)
print(scores.mean())

#testing accuracy
X_train=X_train_tfidf
X_test=training_label
y_train=data2['Text Transcription']
y_test=data2['misogynous']
text_clf = Pipeline([
     ('vect', CountVectorizer()),
     ('tfidf', TfidfTransformer()),
     ('clf',SVC(C=1, gamma=1)),
 ])
text_clf.fit(training_data, training_label)
y_pred = text_clf.predict(y_train)
metrics.accuracy_score(y_test,y_pred)
metrics.confusion_matrix(y_test, y_pred)
print(metrics.classification_report(y_test,y_pred
     ))

#rom google.colab import drive
#drive.mount('/content/drive')

#!unzip /content/drive/MyDrive/mami_data/train_pixel1.zip


#!ls /content/drive/MyDrive/mami_data/training/TRAINING/

```

```
#!unzip -P *MaMiSemEval2022! /content/drive/MyDrive/mami_data/training/TRAINING/
    trial.zip


# !unzip -P *MaMiSemEval2022! /content/drive/MyDrive/mami_data/training/TRAINING
    /test.zip

first creating csv file with feature extraction from images and reading csv file

#import glob
import pandas as pd
#import numpy as np
from keras.preprocessing.image import img_to_array,array_to_img,load_img
train_image = pd.read_csv('train_pixel1.csv',index_col=0)
print(train_image.shape)
test_image = pd.read_csv('test_pixel.csv',index_col=0)
print(test_image.shape)


//reading training and testing datasets
import pandas as pd
total_data_train=pd.read_csv('training.csv',sep='\t')
total_data_test=pd.read_csv('trial.csv',sep='\t')
print(total_data_train['file_name'])

//y_train is extracting from training dataset by checking filename and file_list
import os
file_list=os.listdir(r"./TRAINING/")
#print(file_list)
y_train=[]
for j in range(0,2001):
  for i in range(len(total_data_train)):
      if(file_list[j]==(total_data_train['file_name'][i])):
        y_train.append(total_data_train['misogynous'][i])
      else:
        continue
print(len(y_train))

#y_test is extracting from training dataset by checking filename and file_list
file_list2=os.listdir(r"./Users/fersiniel/Desktop/MAMI - TO LABEL/TRIAL DATASET/
    ")
y_test=[]
for j in file_list2:
  for i in range(len(total_data_test)):
      if(j==(total_data_test['file_name'][i])):
        y_test.append(total_data_test['misogynous'][i])
      else:
        continue

X_train=train_image#train_image is a feature extraction from image dataset
```

```
X_test=test_image#train_image is a feature extraction from image dataset

from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state=42)
X_train, y_train = sm.fit_resample(X_train, y_train)

//SVC algorithm is applied to model

import numpy as np
from sklearn.svm import SVC
import seaborn as sns
from sklearn import metrics
from sklearn.model_selection import cross_val_score
sns.countplot(y_train)
clf=SVC(gamma='auto')
scores=cross_val_score(clf,X_train,y_train,cv=3,scoring='accuracy')
print(scores)
print(len(X_train))
print(scores.mean())


To find best parameter we need tuning hyper parameter using grid search cv



from sklearn.model_selection import GridSearchCV
param_grid = {'C': [0.1, 1],#weight of the error
              'gamma': [1, 0.1],#intercept
              'kernel': ['rbf']}
print(param_grid)
clf=SVC( )
grid=GridSearchCV(clf,param_grid,cv=2,scoring='accuracy',n_jobs=-1)
grid.fit(X_train,y_train)
#grid.grid_scores_
grid.cv_results_
means = grid.cv_results_['mean_test_score']
params = grid.cv_results_['params']
#for mean,param in zip(means,params):
#print("%f  with:  %r" % (mean,param))
grid.mean_scores=means

print(grid.best_score_)
print(grid.best_params_)
print(grid.best_estimator_)


clf=SVC(C=0.1,gamma=1,kernel='rbf')
scores=cross_val_score(clf,X_train,y_train,cv=3,scoring='accuracy')
print(scores)
print(scores.mean())
```

```
clf= SVC()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(metrics.classification_report(y_test,y_pred))

print((0.67+0.90)/2)
```

Chapter 5

Conclusion and Future Scope

5.1 Result Comparision

Algoritham	Text	Image	avg
SVM	90	67	0.78
Random forest	89	57	0.73

Figure 5.1: Comparision Tabel

- Testing accuracy for text data **SVM** given high test accuracy of ”90” and **RANDOM FOREST** given ”89” .
- Testing accuracy for image data **SVM** given the high test accuracy of ”67” and **RANDOM FOREST** given ”57”.
- As per average accuracy of text and image data **SVM** given the high test accuracy of ”78” and **RANDOM FOREST** given ”73”.
- By **Comparing** Algorithms in terms of **Accuracy** I can conclude **SVM Work’s better** than **RANDOM FOREST** on TEXT as well as on IMAGE.

5.2 Future Scope

- We can try applying Algorithms of Deep learning **i.e FNN,CNN,etc** we can try to image sentiment analysis which can give more chances to predict the ouctcome in terms of image.

- With the Combination of Deep Learning techniques we can also use Sequential learning like **RNN,GRU,LSTM,Bi-LSTM**.etc to get sequential information of text data which increases chances of geeting right Decisions .

References

[1] D. Kiela, H. Firooz, A. Mohan, V. Goswami, A. Singh, P. Ringshia, D. Testuggine, The hateful memes challenge: Detecting hate speech in multimodal memes, 2021. arXiv:2005.04790.

[2] M. Anzovino, E. Fersini, P. Rosso, Automatic identification and classification of misogynistic language on twitter, in: International Conference on Applications of Natural Language to Information Systems, Springer, 2018, pp. 57–64.

[3] F. Gasparini, I. Erba, E. Fersini, S. Corchs, Multimodal classification of sexist advertisements., in: ICETE (1), 2018, pp. 565–572.

[4] S. Frenda, B. Ghanem, M. Montes-y Gomez, P. Rosso, Online hate speech against women: Automatic identification ´ of misogyny and sexism on twitter, Journal of Intelligent , Fuzzy Systems 36 (2019) 4743–4752.

[5] F.-M. Plaza-Del-Arco, M. D. Molina-Gonzalez, L. A. Ure ´ na-L ˜ opez, M. T. Mart ´ ´ın-Valdivia, Detecting misogyny and xenophobia in spanish tweets using language technologies, ACM Transactions on Internet Technology (TOIT) 20 (2020) 1–19.

[6] L. Shifman, Memes in a Digital World: Reconciling with a Conceptual Troublemaker, Journal of ComputerMediated Communication 18 (2013) 362–377.

[7] T. Farrell, M. Fernandez, J. Novotny, H. Alani, Exploring misogyny across the manosphere in reddit, in: Proceedings of the 10th ACM Conference on Web Science, 2019, pp. 87–96.

[8] B. Battula, L. Parayitam, T. S. Prasad, P. Balakrishna and C. Patibandla, "Classifications of High Resolution Optical Images using Supervised Algorithms," 2018 IEEE 8th International Advance Computing Conference (IACC), 2018, pp. 126-130, doi: 10.1109/IADCC.2018.8692132.

[9] B. Battula, L. Parayitam, T. S. Prasad, P. Balakrishna and C. Patibandla, "Classifications of High Resolution Optical Images using Supervised Algorithms," 2018 IEEE 8th International Advance Computing Conference (IACC), 2018, pp. 126-130, doi: 10.1109/IADCC.2018.8692132