



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

1	INTRODUCTION	1
1.1	Literature Review	1
1.2	Problem Statement	2
1.3	Objectives	2
2	Theoretical background	3
2.1	What is Classification?	3
2.2	What is Evaluation for Classification?	3
2.2.1	Calculated Metrics using Confusion matrix ?	4
2.3	What Machine learning Algorithms in Classification used ?	6
3	Work Flow	11
3.1	Flow Chart	11
3.1.1	Algorithm for Text Data	11
3.1.2	Algorithm for Image Data	12
4	Program's	13
4.1	RANDOM FOREST	13
4.2	SVM	19
5	Conclusion and Future Scope	27
5.1	Result Comparision	27
5.2	Future Scope	27
	References	28

List of Figures

2.1	Classification	3
2.2	Classification	4
2.3	Accuracy	4
2.4	Precision	5
2.5	Recall	5
2.6	F1-Score	6
2.7	SVM	6
2.8	LINEAR KERNEL	7
2.9	POLYNOMIAL KERNEL	7
2.10	Radial Basis Function Kernel	8
2.11	Random Forest	8
2.12	BAGGING	9
2.13	BOOSTING	10
2.14	MATH	10
3.1	FlowChart	11
5.1	Comparision Tabel	27

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 features helped to identify misogynistic memes. The survey includes not 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 able 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

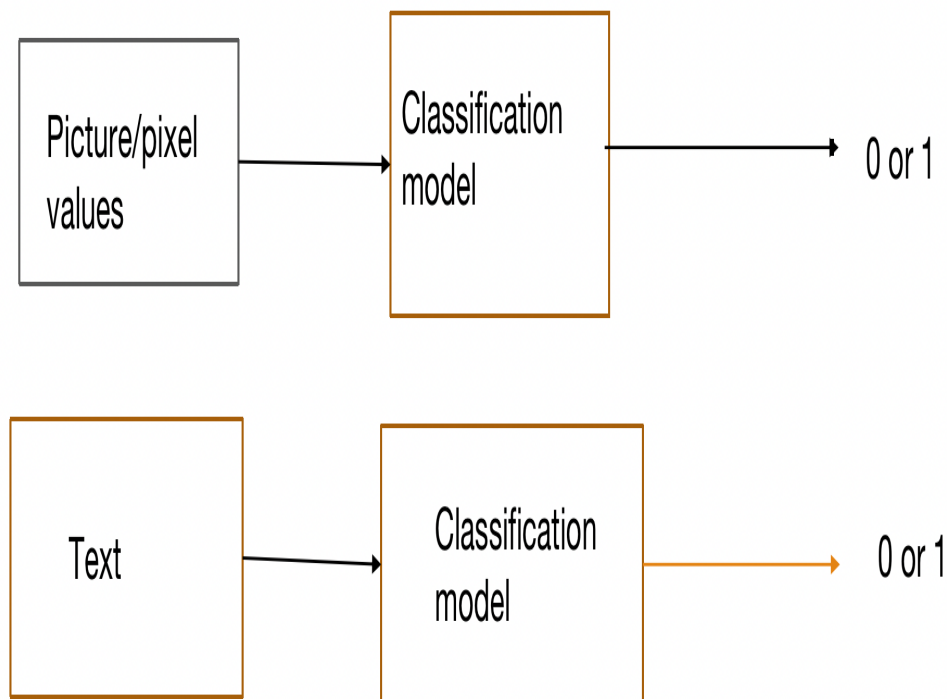


Figure 2.1: Classification

2.2 What is Evaluation for Classification?

Confusion matrix A Confusion matrix is an $N \times N$ 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

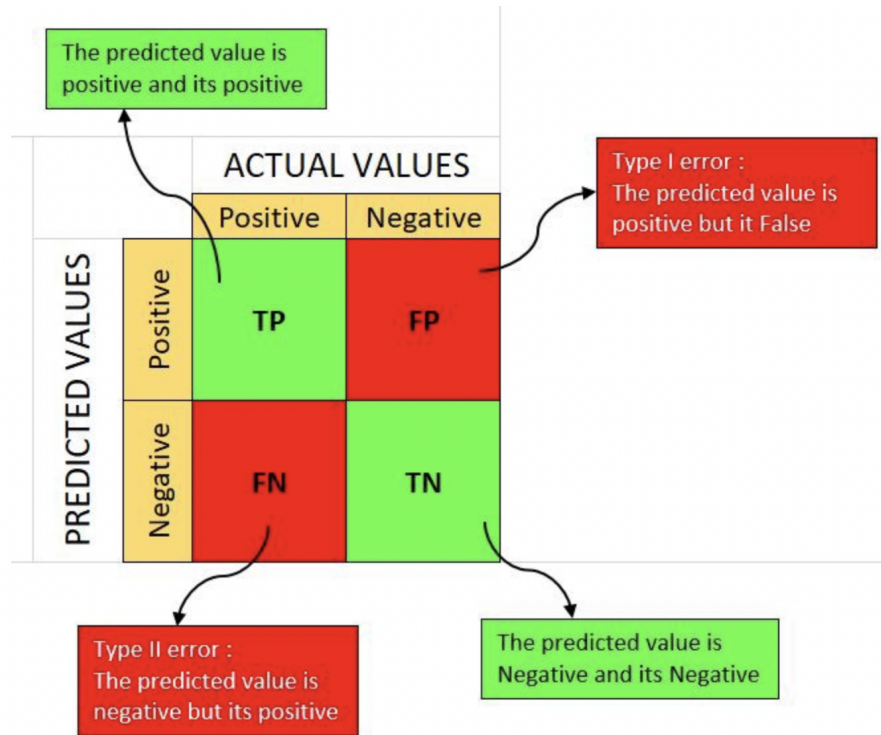


Figure 2.2: Classification

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 positive 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

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

Figure 2.4: Precision

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

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

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

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

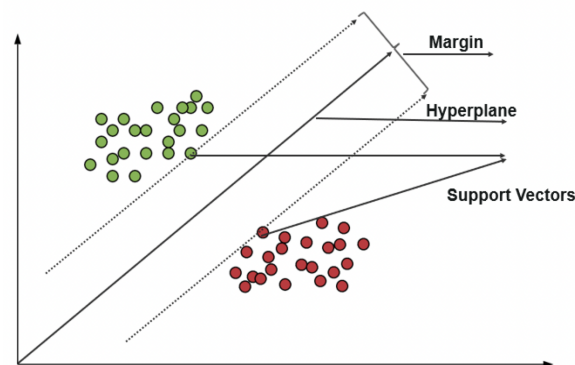


Figure 2.7: SVM

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.

$$f(x) = B(0) + \sum(a_i * (x, x_i))$$

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

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

$\|X_1 - X_2\|$ = Euclidean distance between X_1 & X_2

Figure 2.10: Radial Basis Function Kernel

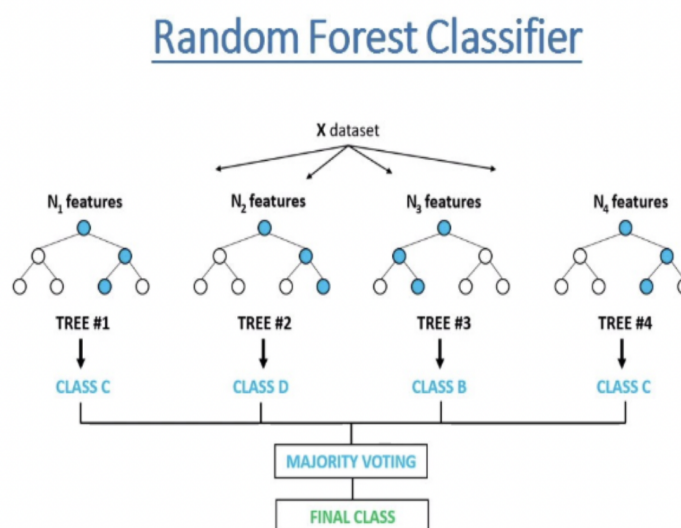


Figure 2.11: Random Forest

Random Forest is the most used supervised machine learning algorithm for classification. Generally it constructs a N number of decision trees 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:

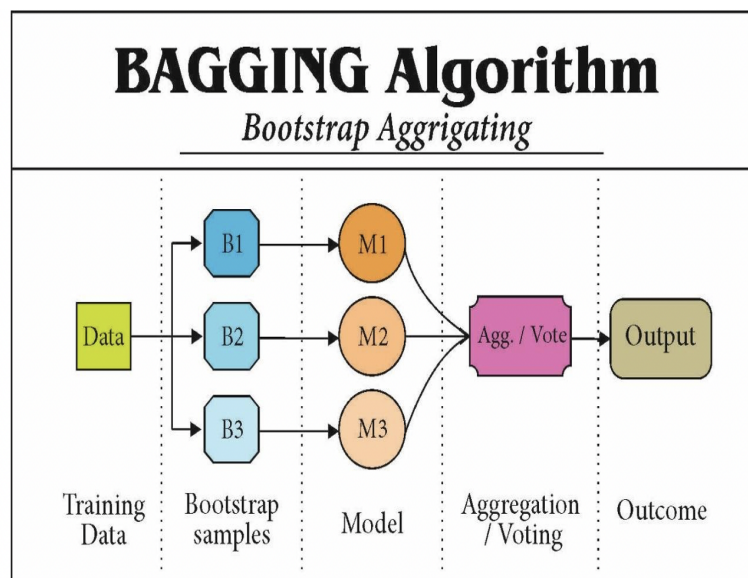


Figure 2.12: BAGGING

As there is limitation of having a large training set, so we take repeated Bootstrap samples (SB) with replacement from the same training set ($S(y_n; x_n), n = 1, \dots, N$, 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 $\phi(x, SB)$.

When y is numerical, we take average of $\phi(x, SB)$,

$$\phi_B(x) = \text{avg } \phi(x, SB).$$

When y is a class label, we take plurality vote of $\phi(x, SB)$ to form $\phi_B(x)$.

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:

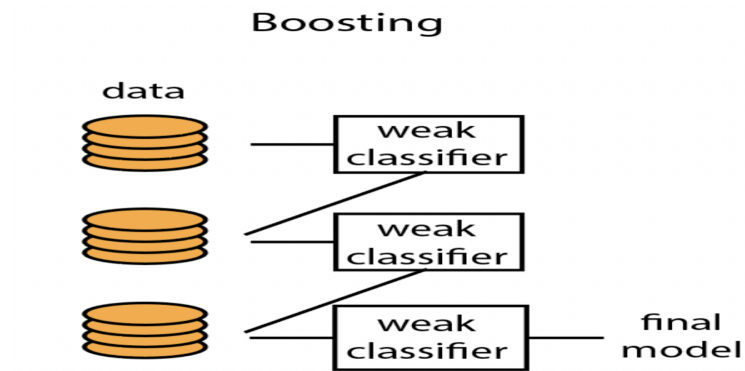


Figure 2.13: BOOSTING

MATH of Boosting algorithm:

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

Initialize $D_1(i) = 1/m$

For $t = 1, \dots, T$

- Train base learner 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

$$\epsilon_t(i) = \Pr(i) D_t(i) [h_t(x_i) \neq y_i]$$

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

where Z_t is a normalization factor

Output of the final classifier:

$$H(x) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(x) \right)$$

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

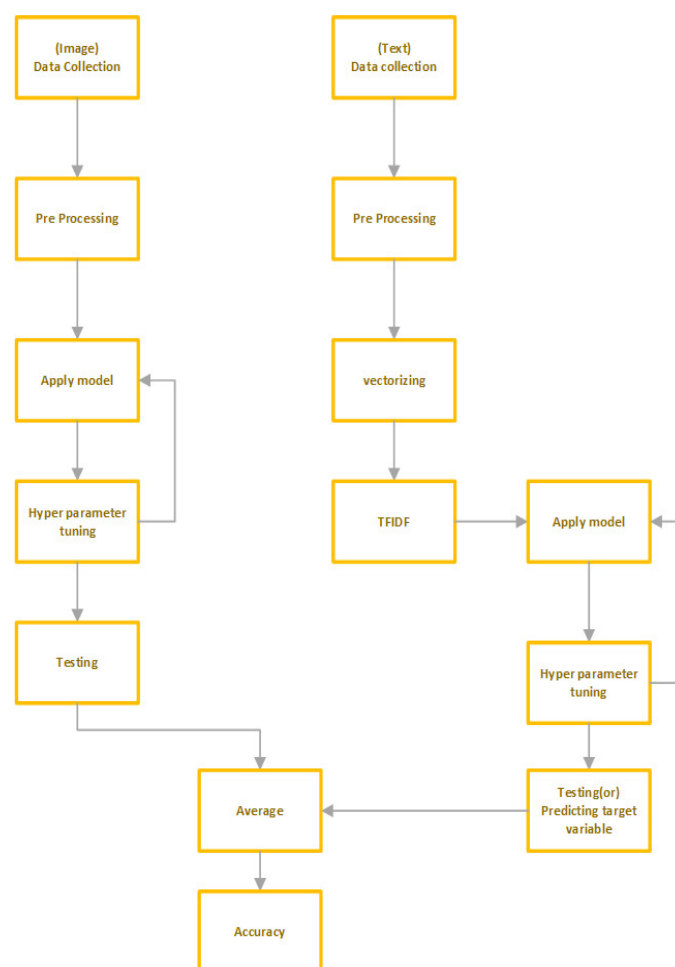


Figure 3.1: FlowChart

3.1.1 Algorithm for Text Data

1. From the data set consisting of 10,000 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 applied the ml algorithms
7. Finally merging the test and image accuracy.

Chapter 4

Program's

4.1 RANDOM FOREST

```
1
2 from google.colab import drive
3 drive.mount('/content/gdrive')
4
5 !unzip gdrive/My\Drive/TRAINING.zip
6
7 !unzip -P *MaMiSemEval2022! gdrive/My\Drive/trial.zip
8
9 !unzip gdrive/My\Drive/train_pixell.csv.zip
10
11 #!ls '/content/drive/MyDrive/mami_data'
12
13 # import zipfile
14
15 # zip_file = "/content/drive/MyDrive/mami_data/training.zip"
16
17 # try:
18 #     with zipfile.ZipFile(zip_file) as z:
19 #         z.extractall("/content/drive/MyDrive/mami_data/training")
20 #         print("Extracted all")
21 # except:
22 #     print("Invalid file")
23
24 #!ls '/content/drive/MyDrive/mami_data/training/TRAINING/training.csv'
25
26 # import zipfile
27
28 # zip_file = "/content/drive/MyDrive/mami_data/trial.zip"
29
30 # try:
31 #     with zipfile.ZipFile(zip_file) as z:
32 #         z.setpassword(pwd = bytes('*MaMiSemEval2022!', 'utf-8'))
33 #         z.extractall("/content/drive/MyDrive/mami_data/trial")
```

```

34 #         print("Extracted all")
35 # except:
36 #         print("Invalid file")
37
38 !ls '/content/drive/MyDrive/mami_data/trial/Users/fersiniel/Desktop/MAMI - TO
    LABEL/TRIAL DATASET'
39
40 #import zipfile
41
42 #zip_file = "/content/drive/MyDrive/mami_data/test.zip"
43
44 #try:
45 #     with zipfile.ZipFile(zip_file) as z:
46 #         z.setpassword(pwd = bytes('*MaMiSemEval2022!', 'utf-8'))
47 #         z.extractall("/content/drive/MyDrive/mami_data/test")
48 #         print("Extracted all")
49 #except:
50 #     print("Invalid file")
51
52 !ls '/content/drive/MyDrive/mami_data/test/test/Test.csv'
53
54 #!cp '/content/drive/MyDrive/mami_data/trial/Users/fersiniel/Desktop/MAMI - TO
    LABEL/TEST DATASET/test.csv' '/content/drive/MyDrive/mami_data/test'
55
56 !cp '/content/drive/MyDrive/mami_data/trial/Users/fersiniel/Desktop/MAMI - TO
    LABEL/TRIAL DATASET/trial.csv' '/content/drive/MyDrive/mami_data/trial'
57
58 //Applying preprocessing with cleaningg tokenizing and lematization
59
60 from sklearn.metrics import classification_report
61 from sklearn.model_selection import train_test_split
62 from sklearn.metrics import roc_auc_score
63 from sklearn.model_selection import GridSearchCV
64 import pandas as pd
65 import seaborn as sns
66 data1=pd.read_csv('TRAINING/training.csv', sep='\t')
67 data2=pd.read_csv('trial.csv', sep='\t')
68 #training_label=data['misogynous']
69 # print(training_data.head())
70 import re
71 import nltk
72 nltk.download('punkt')
73 nltk.download('wordnet')
74 nltk.download('stopwords')
75 from nltk.corpus import stopwords
76 from nltk.stem import WordNetLemmatizer
77 stop_words=stopwords.words('english')
78 stop_words.append('imgflipcom')
79 stop_words.append('zip')
80 print(stop_words)

```

```

81 lemmatizer=WordNetLemmatizer()
82 #training data
83 for index,row in data1.iterrows():
84     #print(row)
85     filter_sentence=[]
86     sentence=row['Text Transcription']
87     sentence = sentence.lower()
88     #print(sentence)
89     sentence=re.sub(r'^\w\s',' ',sentence)#cleaning
90     words=nltk.word_tokenize(sentence)
91     words=[w for w in words if not w in stop_words]
92     for word in words:
93         filter_sentence.append(lemmatizer.lemmatize(word))
94     #print(filter_sentence)
95     listToStr = ' '.join([str(elem) for elem in filter_sentence])
96     data1.loc[index,"Text Transcription"]=listToStr
97 #trail data
98 for index,row in data2.iterrows():
99     #print(row)
100     filter_sentence=[]
101     sentence=row['Text Transcription']
102     sentence = sentence.lower()
103     #print(sentence)
104     sentence=re.sub(r'^\w\s',' ',sentence)#cleaning
105     words=nltk.word_tokenize(sentence)
106     words=[w for w in words if not w in stop_words]
107     for word in words:
108         filter_sentence.append(lemmatizer.lemmatize(word))
109     #print(filter_sentence)
110     listToStr = ' '.join([str(elem) for elem in filter_sentence])
111     data2.loc[index,"Text Transcription"]=listToStr
112 print(data1.head())
113 print(data1.shape)
114 print(data2.head())
115 print(data2.shape)
116 #data=pd.concat([data1 , data2])
117 #print(data.head())
118 #print(data.shape)
119
120 //applying countvectorizer
121
122 from sklearn.feature_extraction.text import CountVectorizer
123 count_vect = CountVectorizer()
124 training_data=data1['Text Transcription']
125 training_label=data1['misogynous']
126 X_train_counts = count_vect.fit_transform(training_data)
127 X_train_counts.shape
128
129
130 //TF-IDF transormer

```

```

131
132 from sklearn.feature_extraction.text import TfidfTransformer
133 tfidf_transformer = TfidfTransformer()
134 X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
135 X_train_tfidf.shape
136 print(X_train_tfidf)
137
138
139 from sklearn.pipeline import Pipeline
140 import numpy as np
141 from sklearn import metrics
142 from sklearn.model_selection import cross_val_score
143 sns.countplot(training_label)
144 text_clf = Pipeline([
145     ('vect', CountVectorizer()),
146     ('tfidf', TfidfTransformer()),
147     ('clf', RandomForestClassifier(max_depth=2, random_state=0)),
148 ])
149 scores=cross_val_score(text_clf, training_data, training_label, cv=5, scoring='
    accuracy')
150 print(scores)
151 print(scores.mean())
152
153
154 //To find best parameter we need tuning hyper parameter using grid search cv
155
156
157
158 from sklearn.model_selection import GridSearchCV
159 param_grid = { 'bootstrap': [True], 'max_depth': [5, 10, None], 'max_features':
    ['auto', 'log2'], 'n_estimators': [5, 6, 7, 8]}
160 print(param_grid)
161 clf=RandomForestClassifier()
162 grid=GridSearchCV(clf, param_grid, cv=3, scoring='accuracy', n_jobs=-1)
163 grid.fit(X_train_tfidf, training_label)
164 #grid.grid_scores_
165 grid.cv_results_
166 means = grid.cv_results_[ 'mean_test_score' ]
167 params = grid.cv_results_[ 'params' ]
168 for mean, param in zip(means, params):
169     print("%f with: %r" % (mean, param))
170 grid.mean_scores=means
171 print(grid.best_score_)
172 print(grid.best_params_)
173 print(grid.best_estimator_)
174
175 // validation accuracy for training data
176 clf=RandomForestClassifier(bootstrap= True, max_depth=None, max_features='auto',
    n_estimators= 8)
177 scores=cross_val_score(clf, X_train_tfidf, training_label, cv=3, scoring='accuracy')

```

```

178 print(scores)
179 print(scores.mean())
180
181 //testing accuracy
182 X_train=X_train_tfidf
183 X_test=training_label
184 y_train=data2['Text Transcription']
185 y_test=data2['misogynous']
186 text_clf = Pipeline([
187     ('vect', CountVectorizer()),
188     ('tfidf', TfidfTransformer()),
189     ('clf', RandomForestClassifier(bootstrap= True, max_depth=None, max_features
        ='auto', n_estimators= 8)),
190 ])
191 text_clf.fit(training_data, training_label)
192 y_pred = text_clf.predict(y_train)
193 metrics.accuracy_score(y_test, y_pred)
194 metrics.confusion_matrix(y_test, y_pred)
195 print(metrics.classification_report(y_test, y_pred
196     ))
197
198 from google.colab import drive
199 drive.mount('/content/drive')
200
201 #!unzip /content/drive/MyDrive/mami_data/train_pixell.zip
202
203
204 #!ls /content/drive/MyDrive/mami_data/training/TRAINING/
205
206 #!unzip -P *MaMiSemEval2022! /content/drive/MyDrive/mami_data/training/TRAINING/
    trial.zip
207
208
209 # !unzip -P *MaMiSemEval2022! /content/drive/MyDrive/mami_data/training/TRAINING
    /test.zip
210
211 first creating csv file with feature extraction from images and reading csv file
212
213 #import glob
214 import pandas as pd
215 import numpy as np
216 from keras.preprocessing.image import img_to_array, array_to_img, load_img
217 train_image = pd.read_csv('train_pixell.csv', index_col=0)
218 print(train_image.shape)
219 test_image = pd.read_csv('test_pixel.csv', index_col=0)
220 print(test_image.shape)
221
222
223 //reading training and testing datasets
224 import pandas as pd

```

```

225 total_data_train=pd.read_csv('training.csv',sep='\t')
226 total_data_test=pd.read_csv('trial.csv',sep='\t')
227 print(total_data_train['file_name'])
228
229 // y_train is extracting from training dataset by checking filename and file_list
230 import os
231 file_list=os.listdir(r"./TRAINING/")
232 #print(file_list)
233 y_train=[]
234 for j in range(0,2001):
235     for i in range(len(total_data_train)):
236         if(file_list[j]==(total_data_train['file_name'][i])):
237             y_train.append(total_data_train['misogynous'][i])
238         else:
239             continue
240 print(len(y_train))
241
242 // y_test is extracting from training dataset by checking filename and file_list
243 file_list2=os.listdir(r"/Users/fersiniel/Desktop/MAMI - TO LABEL/TRIAL DATASET/")
244 )
245 y_test=[]
246 for j in file_list2:
247     for i in range(len(total_data_test)):
248         if(j==(total_data_test['file_name'][i])):
249             y_test.append(total_data_test['misogynous'][i])
250         else:
251             continue
252
253 X_train=train_image#train_image is a feature extraction from image dataset
254 X_test=test_image#train_image is a feature extraction from image dataset
255
256 from imblearn.over_sampling import SMOTE
257 sm = SMOTE(random_state=42)
258 X_train , y_train = sm.fit_resample(X_train , y_train)
259
260 //Random Forest algorithm is applied to model
261
262 from sklearn.ensemble import RandomForestClassifier
263 import numpy as np
264 import seaborn as sns
265 from sklearn import metrics
266 from sklearn.model_selection import cross_val_score
267 sns.countplot(y_train)
268 clf=RandomForestClassifier(max_depth=2, random_state=0)
269 scores=cross_val_score(clf, X_train , y_train , cv=3, scoring='accuracy')
270 print(scores)
271 print(len(X_train))
272 print(scores.mean())
273

```

```

274
275 from sklearn.model_selection import GridSearchCV
276 param_grid = { 'bootstrap': [True], 'max_depth': [5, 10, None], 'max_features':
    ['auto', 'log2'], 'n_estimators': [5, 6, 7, 8]}
277 print(param_grid)
278 clf=RandomForestClassifier( )
279 grid=GridSearchCV(clf, param_grid, cv=2, scoring='accuracy', n_jobs=-1)
280 grid.fit(X_train, y_train)
281 #grid.grid_scores_
282 grid.cv_results_
283 means = grid.cv_results_[ 'mean_test_score' ]
284 params = grid.cv_results_[ 'params' ]
285 #for mean, param in zip(means, params):
286 #print("%f with: %r" % (mean, param))
287 grid.mean_scores=means
288
289 print(grid.best_score_)
290 print(grid.best_params_)
291 print(grid.best_estimator_)
292
293 clf=RandomForestClassifier(bootstrap= True, max_depth= None, max_features= 'log2
    ', n_estimators= 8)
294 scores=cross_val_score(clf, X_train, y_train, cv=2, scoring='accuracy')
295 print(scores)
296 print(scores.mean())
297
298
299 clf=RandomForestClassifier(bootstrap= True, max_depth= None, max_features= 'log2
    ', n_estimators= 8)
300 clf.fit(X_train, y_train)
301 y_pred = clf.predict(X_test)
302 print(metrics.classification_report(y_test, y_pred))

```

4.2 SVM

```

1 from google.colab import drive
2 drive.mount('/content/gdrive')
3
4 !unzip gdrive/My\Drive/TRAINING.zip
5
6 !unzip -P *MaMiSemEval2022! gdrive/My\Drive/trial.zip
7
8 !unzip gdrive/My\Drive/train_pixell.csv.zip
9
10 #!ls '/content/drive/MyDrive/mami_data'
11

```

```

12 # import zipfile
13
14 # zip_file = "/content/drive/MyDrive/mami_data/training.zip"
15
16 # try:
17 #     with zipfile.ZipFile(zip_file) as z:
18 #         z.extractall("/content/drive/MyDrive/mami_data/training")
19 #         print("Extracted all")
20 # except:
21 #     print("Invalid file")
22
23 !ls '/content/drive/MyDrive/mami_data/training/TRAINING/training.csv'
24
25 # import zipfile
26
27 # zip_file = "/content/drive/MyDrive/mami_data/trial.zip"
28
29 # try:
30 #     with zipfile.ZipFile(zip_file) as z:
31 #         z.setpassword(pwd = bytes('*MaMiSemEval2022!', 'utf-8'))
32 #         z.extractall("/content/drive/MyDrive/mami_data/trial")
33 #         print("Extracted all")
34 # except:
35 #     print("Invalid file")
36
37 !ls '/content/drive/MyDrive/mami_data/trial/Users/fersiniel/Desktop/MAMI - TO
    LABEL/TRIAL DATASET'
38
39 #import zipfile
40
41 #zip_file = "/content/drive/MyDrive/mami_data/test.zip"
42
43 #try:
44 #    with zipfile.ZipFile(zip_file) as z:
45 #        z.setpassword(pwd = bytes('*MaMiSemEval2022!', 'utf-8'))
46 #        z.extractall("/content/drive/MyDrive/mami_data/test")
47 #        print("Extracted all")
48 #except:
49 #    print("Invalid file")
50
51 !ls '/content/drive/MyDrive/mami_data/test/test/Test.csv'
52
53 #!cp '/content/drive/MyDrive/mami_data/trial/Users/fersiniel/Desktop/MAMI - TO
    LABEL/TEST DATASET/test.csv' '/content/drive/MyDrive/mami_data/test'
54
55 !cp '/content/drive/MyDrive/mami_data/trial/Users/fersiniel/Desktop/MAMI - TO
    LABEL/TRIAL DATASET/trial.csv' '/content/drive/MyDrive/mami_data/trial'
56
57 //Applying preprocessing with cleaningg tokenizing and lematization
58

```



```

59 from sklearn.metrics import classification_report
60 from sklearn.model_selection import train_test_split
61 from sklearn.metrics import roc_auc_score
62 from sklearn.model_selection import GridSearchCV
63 import pandas as pd
64 import seaborn as sns
65 data1=pd.read_csv('training.csv', sep='\t')
66 data2=pd.read_csv("trial.csv", sep='\t')
67 #training_label=data['misogynous']
68 # print(training_data.head())
69 import re
70 import nltk
71 nltk.download('punkt')#sentence tokenizer
72 nltk.download('wordnet')#is another nltk corpus reader
73 nltk.download('stopwords')
74 from nltk.corpus import stopwords
75 from nltk.stem import WordNetLemmatizer
76 stop_words=stopwords.words('english')
77 stop_words.append('imgflipcom')
78 stop_words.append('zip')
79 print(stop_words)
80 lemmatizer=WordNetLemmatizer()
81 #training data
82 for index,row in data1.iterrows():
83     #print(row)
84     filter_sentence=[]
85     sentence=row['Text Transcription']
86     sentence = sentence.lower()
87     #print(sentence)
88     sentence=re.sub(r'^\w\s','',sentence)#cleaning
89     words=nltk.word_tokenize(sentence)
90     words=[w for w in words if not w in stop_words]
91     for word in words:
92         filter_sentence.append(lemmatizer.lemmatize(word))
93     #print(filter_sentence)
94     listToStr = ' '.join([str(elem) for elem in filter_sentence])
95     data1.loc[index,"Text Transcription"]=listToStr
96 #trail data
97 for index,row in data2.iterrows():
98     #print(row)
99     filter_sentence=[]
100     sentence=row['Text Transcription']
101     sentence = sentence.lower()
102     #print(sentence)
103     sentence=re.sub(r'^\w\s','',sentence)#cleaning
104     words=nltk.word_tokenize(sentence)
105     words=[w for w in words if not w in stop_words]
106     for word in words:
107         filter_sentence.append(lemmatizer.lemmatize(word))
108     #print(filter_sentence)

```

```

109     listToStr = ' '.join([str(elem) for elem in filter_sentence])
110     data2.loc[index, "Text Transcription"] = listToStr
111 print(data1.head())
112 print(data1.shape)
113 print(data2.head())
114 print(data2.shape)
115 #data=pd.concat([data1, data2])
116 #print(data.head())
117 #print(data.shape)
118
119 //applying countvectorizer
120
121 from sklearn.feature_extraction.text import CountVectorizer
122 count_vect = CountVectorizer()
123 training_data=data1['Text Transcription']
124 training_label=data1['misogynous']
125 X_train_counts = count_vect.fit_transform(training_data)
126 X_train_counts.shape
127
128
129 //TF-IDF transormer
130
131 from sklearn.feature_extraction.text import TfidfTransformer
132 tfidf_transformer = TfidfTransformer()
133 X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
134 X_train_tfidf.shape
135 print(X_train_tfidf)
136
137
138 from sklearn.pipeline import Pipeline
139 import numpy as np
140 from sklearn import metrics
141 from sklearn.model_selection import cross_val_score
142 sns.countplot(training_label)
143 text_clf = Pipeline([
144     ('vect', CountVectorizer()),
145     ('tfidf', TfidfTransformer()),
146     ('clf', SVC(gamma='auto')),
147 ])
148 scores=cross_val_score(text_clf, training_data, training_label, cv=10, scoring='
    accuracy')
149 print(scores)
150 print(scores.mean())
151
152
153 //To find best parameter we need tuning hyper parameter using Gridsearchcv
154
155
156
157 from sklearn.model_selection import GridSearchCV

```

```

158 param_grid = {'C': [0.1, 1, 10],# control parameter of error
159               'gamma': [1, 0.1, 0.01],#curvature shape
160               'kernel': ['rbf']}#only stores support vector and radial basis
               function
161 print(param_grid)
162 clf=SVC()
163 grid=GridSearchCV(clf , param_grid ,cv=3,scoring='accuracy' ,n_jobs=-1)
164 grid.fit(X_train_tfidf , training_label)
165 #grid.grid_scores_
166 grid.cv_results_
167 means = grid.cv_results_['mean_test_score']
168 params = grid.cv_results_['params']
169 for mean,param in zip(means,params):
170     print("%f with: %r" % (mean,param))
171 grid.mean_scores=means
172 print(grid.best_score_)
173 print(grid.best_params_)
174 print(grid.best_estimator_)
175
176 #validation accuracy for training data
177 clf=SVC(C=1, gamma=1)
178 scores=cross_val_score(clf , X_train_tfidf , training_label ,cv=3,scoring='accuracy')
179 print(scores)
180 print(scores.mean())
181
182 #testing accuracy
183 X_train=X_train_tfidf
184 X_test=training_label
185 y_train=data2['Text Transcription']
186 y_test=data2['misogynous']
187 text_clf = Pipeline([
188     ('vect', CountVectorizer()),
189     ('tfidf', TfidfTransformer()),
190     ('clf',SVC(C=1, gamma=1)),
191 ])
192 text_clf.fit(training_data , training_label)
193 y_pred = text_clf.predict(y_train)
194 metrics.accuracy_score(y_test , y_pred)
195 metrics.confusion_matrix(y_test , y_pred)
196 print(metrics.classification_report(y_test , y_pred
197     ))
198
199 #rom google.colab import drive
200 #drive.mount('/content/drive')
201
202 #!unzip /content/drive/MyDrive/mami_data/train_pixell.zip
203
204
205 #!ls /content/drive/MyDrive/mami_data/training/TRAINING/
206

```

```

207 #!unzip -P *MaMiSemEval2022! /content/drive/MyDrive/mami_data/training/TRAINING/
    trial.zip
208
209
210 # !unzip -P *MaMiSemEval2022! /content/drive/MyDrive/mami_data/training/TRAINING
    /test.zip
211
212 first creating csv file with feature extraction from images and reading csv file
213
214 #import glob
215 import pandas as pd
216 #import numpy as np
217 from keras.preprocessing.image import img_to_array, array_to_img, load_img
218 train_image = pd.read_csv('train_pixell.csv', index_col=0)
219 print(train_image.shape)
220 test_image = pd.read_csv('test_pixel.csv', index_col=0)
221 print(test_image.shape)
222
223
224 //reading training and testing datasets
225 import pandas as pd
226 total_data_train=pd.read_csv('training.csv', sep='\t')
227 total_data_test=pd.read_csv('trial.csv', sep='\t')
228 print(total_data_train['file_name'])
229
230 //y_train is extracting from training dataset by checking filename and file_list
231 import os
232 file_list=os.listdir(r"./TRAINING/")
233 #print(file_list)
234 y_train=[]
235 for j in range(0,2001):
236     for i in range(len(total_data_train)):
237         if(file_list[j]==(total_data_train['file_name'][i])):
238             y_train.append(total_data_train['misogynous'][i])
239         else:
240             continue
241 print(len(y_train))
242
243 #y_test is extracting from training dataset by checking filename and file_list
244 file_list2=os.listdir(r"./Users/fersiniel/Desktop/MAMI - TO LABEL/TRIAL DATASET/
    ")
245 y_test=[]
246 for j in file_list2:
247     for i in range(len(total_data_test)):
248         if(j==(total_data_test['file_name'][i])):
249             y_test.append(total_data_test['misogynous'][i])
250         else:
251             continue
252
253 X_train=train_image#train_image is a feature extraction from image dataset

```

```

254 X_test=test_image#train_image is a feature extraction from image dataset
255
256 from imblearn.over_sampling import SMOTE
257 sm = SMOTE(random_state=42)
258 X_train , y_train = sm.fit_resample(X_train , y_train)
259
260 //SVC algorithm is applied to model
261
262 import numpy as np
263 from sklearn.svm import SVC
264 import seaborn as sns
265 from sklearn import metrics
266 from sklearn.model_selection import cross_val_score
267 sns.countplot(y_train)
268 clf=SVC(gamma='auto')
269 scores=cross_val_score(clf,X_train,y_train,cv=3,scoring='accuracy')
270 print(scores)
271 print(len(X_train))
272 print(scores.mean())
273
274
275 To find best parameter we need tuning hyper parameter using grid search cv
276
277
278
279 from sklearn.model_selection import GridSearchCV
280 param_grid = {'C': [0.1, 1],#weight of the error
281               'gamma': [1, 0.1],#intercept
282               'kernel': ['rbf']}
283 print(param_grid)
284 clf=SVC( )
285 grid=GridSearchCV(clf,param_grid,cv=2,scoring='accuracy',n_jobs=-1)
286 grid.fit(X_train,y_train)
287 #grid.grid_scores_
288 grid.cv_results_
289 means = grid.cv_results_[ 'mean_test_score' ]
290 params = grid.cv_results_[ 'params' ]
291 #for mean,param in zip(means,params):
292 #print("%f with: %r" % (mean,param))
293 grid.mean_scores=means
294
295 print(grid.best_score_)
296 print(grid.best_params_)
297 print(grid.best_estimator_)
298
299
300 clf=SVC(C=0.1,gamma=1,kernel='rbf')
301 scores=cross_val_score(clf,X_train,y_train,cv=3,scoring='accuracy')
302 print(scores)
303 print(scores.mean())

```

```
304  
305  
306 clf= SVC()  
307 clf.fit(X_train , y_train)  
308 y_pred = clf.predict(X_test)  
309 print(metrics.classification_report(y_test , y_pred))  
310  
311 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 senti-ment analysis which can give more chances to predict the outcome 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 ´ ´in-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