Introduction

Social media have through recent years grown and have become an important part of our lives. This notebook wants to explore whether it is possible to use data from reddit to detect misogyny in social media platforms and create a machine learning model to detect misogyny on these platforms. The reason we have chosen reddit to detect misogynistic content is according to (Farrell et al., 2019) which results indicate that misogynistic content on reddit have been growing and the content have increased in a more violent manner. The dataset is from the article "An Expert Annotated Dataset for the Detection of Online Misogyny" (Ella Guest et. al., 19-Apr-2021) containing 6567 entries from reddit based on posts and comments. This article has been used as the main source of validity on defining misogyny, of which the article states that the dataset they present has the purpose of: "(The dataset) can be used to develop more accurate and nuanced classification models.".

The definition of misogynistic in the article derives from "2k Urban Dictionary definitions of which half are labelled as misogynistic." which they have summerised as the following definition: "Misogynistic content directs abuse at women or a closely related gendered group (e.g. feminists)"

The notebook will examine the problem by:

- Focusing on how to solve a categorical problem and build a machine learning model which can predict whether a comment is misogynistic or not.

Link to functioning notebook:
https://colab.research.google.com/drive/1E7e6sYaBHpH9p0RqtVvkSpVTx8qgPfFu?usp=sharing
(https://colab.research.google.com/drive/1E7e6sYaBHpH9p0RqtVvkSpVTx8qgPfFu?usp=sharing)

Order of operations

- Package loading
- Preprocessing / EDA
- Unsupervised machine learning
  - Topic modeling
  - Clustering
- Supervised machine learning
  - Logistic regression
  - XGB model
  - Hyperparameter tuning of XGB
- Applying SML model to new data
  - Preprocessing twitter data
  - Untuned XGB model
  - Tuned XGB model
- Conclusion
  - Caveats

Package loading

In [ ]:

```
import pandas as pd
import numpy as np
import spacy
nlp = spacy.load('en')
from sklearn.feature_extraction.text import TfidfVectorizer
import itertools
from collections import Counter
import matplotlib.pyplot as plt
from nltk.probability import FreqDist as FD
from wordcloud import WordCloud
from gensim.corpora.dictionary import Dictionary
from gensim.models.tfidfmodel import TfidfModel
from gensim.models.lsimodel import LsiModel
from gensim.similarities import MatrixSimilarity
!pip install "umap.learn==0.3.10" -qq
import umap.umap_ as umap
from sklearn.cluster import KMeans
import seaborn as sns
sns.set(rc = {'figure.figsize':(15,8)})
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report
from xgboost import XGBRegressor
from xgboost import XGBClassifier
import xgboost as xgb
!pip install tweet-preprocessor -qq
import preprocessor as prepro
```

Preprocessing & EDA

In [ ]:

```
# Loading the dataset and taking a look
miso_data = pd.read_csv('https://raw.githubusercontent.com/ellamguest/online-misogyny-eacl2021/main/data/final_labels.csv')
miso_data.head()
```

Out[ ]:

	entry_id	link_id	parent_id	entry_utc	subreddit	author	
0	exoxn7	t3_exoxn7	NaN	1580652620	badwomensanatomy	doggodone	Do yc s
1	fgb3bdv	t3_exoxn7	t3_exoxn7	1580658139	badwomensanatomy	Machaeon	Worı just c This gı extra insan hy he look
2	fgc6tlu	t3_exoxn7	t3_exoxn7	1580669695	badwomensanatomy	CuniculusVincitOmnia	H fa about th r profe ord
3	fge6msg	t3_exoxn7	t1_fgc6tlu	1580692566	badwomensanatomy	its331am	Sourc sou
4	fgawus5	t3_exoxn7	t3_exoxn7	1580656280	badwomensanatomy	JaxDefore	Dar mov the ba bloc to d did

In [ ]:

```
# Removing unnessecary columns, we dont need these for our purpose.
miso_data.drop(['entry_id' ,'link_id' ,'parent_id', 'entry_utc', 'image', 'week', 'group', 'sheet_order', 'highlight', 'split', 'level_2', 'level_3', 'strength'], axis=1, inplace=True)
```

In [ ]:

```
# We lack a couple observations of the body text
miso_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6567 entries, 0 to 6566
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   subreddit   6567 non-null   object
 1   author      6567 non-null   object
 2   body        6555 non-null   object
 3   label_date  6567 non-null   object
 4   level_1     6567 non-null   object
dtypes: object(5)
memory usage: 256.6+ KB
```

In [ ]:

```
# We could see that there were missing text in the "body" column which was replaced wit
h [removed] or [deleted]
miso_data.describe()
```

Out[ ]:

	subreddit	author	body	label_date	level_1
count	6567	6567	6555	6567	6567
unique	105	3921	6153	6	2
top	MGTOW	[deleted]	[removed]	16-03-2020	Nonmisogynistic
freq	600	165	75	1518	5868

In [ ]:

```
# We wanted to exclude these missing datapoints in the body column for later tokenizati
on, and replaced these with a NA value.
miso_data['body'] = miso_data['body'].astype(str).apply(lambda x: x.strip()).replace([
'[removed]','[deleted]', 'nan'], np.nan)
```

In [ ]:

```
# Removing NA values from the body column
miso_data.dropna(subset=['body'], inplace=True)
```

In [ ]:

```
# Resetting index, so the filtered observations starts from 0
miso_data.reset_index(inplace=True)
```

In [ ]:

```
# Removing the 'r/' from subreddit mentions, causing further processing of text data to
be cleaner.
miso_data['body'] = miso_data['body'].replace('r/', '')
```

In [ ]:

```
# Counting the amount of misogynistic and nonmisogynistic comments.
counted_level = Counter(miso_data['level_1'])
counted_level
```

Out[ ]:

```
Counter({'Misogynistic': 699, 'Nonmisogynistic': 5724})
```

In [ ]:

```
# Making lists for the tokens
clean_comments = []
dict_comments = []

# Taking the miso comment data 'body' and passing through the spacy nlp pipeline tokeni
zer
for comment in nlp.pipe(miso_data['body']):
# We process the tokens by lemmatizing, lowering capital letters, removing stopwords an
d punctuations.
  cmt = [token.lemma_.lower() for token in comment
         if token.is_alpha
         and not token.is_stop
         and not token.is_punct]
# Appending and extending the tokens into clean_comments and dict_comments respectively
as lists.
# We do both, because we are unable to use the appended list to create a proper diction
ary,
# Though we use the appended list to place into the dataset as a token column.
  clean_comments.append(cmt)
  dict_comments.extend(cmt)
```

In [ ]:

```
# Adding the tokens to a token column in the dataset.
miso_data['tokens'] = clean_comments
```

In [ ]:

```
# Displaying dataset with token column.
miso_data
```

Out[ ]:

	index	subreddit	author	body	label_date	l
0	0	badwomensanatomy	doggodone	Do you have the skin of a 80 year old grandma? Worry no more, just drink water!	17-02-2020	Nonmisog
1	1	badwomensanatomy	Machaeon	This is taking a grain of truth and extrapolating to insanity.\n\nStay hydrated, it's healthy, you'll look and feel ...	17-02-2020	Nonmisog
2	2	badwomensanatomy	CuniculusVincitOmnia	Honestly my favorite thing about this is that they feel the need to cite beauty professionals in order to prove that...	17-02-2020	Nonmisog
3	3	badwomensanatomy	its331am	Source? Doesnt sound right to me idk	17-02-2020	Nonmisog
4	4	badwomensanatomy	JaxDefore	Damn, I saw a movie in which the old woman bathed in the blood if virgins to do this. How did no one tell her she ...	17-02-2020	Misog
...	...	...	...	...	...	
6418	6562	RedPillWomen	balletaurelie	Should I send my ex-boyfriends mom a thank you card? If so, how? Would love some help from the RP ladies!As I posted...	16-03-2020	Misog
6419	6563	TheRedPill	Su_shii	1. Whatever vibration you put out, you will attract \n\nA lot of you sound like garbage (or just have very flawed, i...	16-03-2020	Misog

	index	subreddit	author	body	label_date	l
6420	6564	TheRedPill	Su_shii	1. Whatever vibration you put out, you will attract \n\nA lot of you sound like garbage (or just have very flawed, i...	16-03-2020	Misog
6421	6565	Trufemcels	CrybabyOgress	>Please keep in mind that just because YOU view someone as a normie or a becky or whatever does not mean that other ...	16-03-2020	Misog
6422	6566	AskFeminists	mymiddlenameisrae	Yes. I mean, given an example:\n\nRacist white women will always called out before racist white men\n\nThis isn't to...	23-03-2020	Misog

6423 rows × 7 columns

In [ ]:

```
# Creating dictionary from tokens list with dict_comments with set function, to remove
 dublicated words for vectorization.
token_dictionary = list(set(dict_comments))
```

In [ ]:

```
# Vectorizing the dictionary, creating a dimensionality reduced variable (X) for later
 use in SML
vectorizer = TfidfVectorizer(vocabulary=token_dictionary)
X = vectorizer.fit_transform(miso_data['body'])
```

In [ ]:

```
# Creating bag of words
tok_bow = pd.DataFrame(X.A, columns=vectorizer.get_feature_names())
```

In [ ]:

```
# Start by merging the data with the bag of words
miso_data_tok = pd.merge(miso_data, tok_bow, left_index=True, right_index=True)
```

In [ ]:

```
# Creating a new dataframe, containing our bag of words, for the categories misogynisti
c and nonmisogynistic.
miso_tokens = miso_data_tok.loc[(miso_data_tok['level_1'] == 'Misogynistic')]
```

In [ ]:

```
# Counting and sorting by most popular tokens/words apperances in misogynistic tagged c
omments.
miso_tokens_2 = itertools.chain(* miso_tokens['tokens'])
miso_tokens_count = Counter(miso_tokens_2)
top_common_miso = miso_tokens_count.most_common()[:10]
```

In [ ]:

```
# Displaying the top 10 common words for misogynistic tagged comments.
top_common_miso
```

Out[ ]:

```
[('woman', 1294),
 ('not', 933),
 ('like', 824),
 ('man', 771),
 ('want', 588),
 ('girl', 542),
 ('time', 462),
 ('fuck', 416),
 ('guy', 391),
 ('get', 388)]
```

In [ ]:

```
# Visualizing common words for misogynistic comments in a word cloud.
# The larger the words in the cloud, the more the word occures in observations.
token_freq = FD(dict_comments)
wc = WordCloud(width=800, height=400, max_words=500).generate_from_frequencies(token_fr
eq)
plt.figure(figsize=(12,10))
plt.imshow(wc, interpolation='bilinear')
plt.axis('off')
plt.show()
```

In this word cloud we can observe that the majority of words has a focus on women and verbs, which does make sense considering the data revolves around misogyny.

Since misogyny is specifically hate towards women we think these tokens look good for further use in machine learning as we want the algorithms to learn to distinguish when a text is specifically making unfair assumptions about women in a derogatory way.

Unsupervised machine learning

We have chosen to utilize UML initially because it enables us to attempt to find new links in the data. As an example, the data we have has factors such as which subreddit the comments were made in, which might lead the machine learning algorithm to identify subreddits where misogynistic comments are more widespread and as such create a connection around that subreddit.

Because our sample size is pretty small and the different subreddits are differently represented we have instead chosen for our UML to focus on: topic modeling and clustering.

Topic modelling

Topic modelling is a statistical model for observing and discovering abstract “topics” found across different texts. It will look at the semantic structure of texts and find which words often occur together, to try to identify which comments are similar.

The algorithm attempts to put different data points together, so we can observe the most common ways to structure a sentence belonging to a specific "topic". Though we do know that since the model is unsupervised and misogynistic comments aren't the majority, that the model might find different topics to focus on, but this might lead us to identifying other factors that might play a role.

In [ ]:

```
# Creating a dictionary with all tokens
token_dic = Dictionary(miso_data['tokens'])
```

In [ ]:

```
# Creating a corpus, containing the tokens from the dictionary of all comments.
corpus = [token_dic.doc2bow(doc) for doc in miso_data['tokens']]
```

In [ ]:

```
# Showing that the randomly chosen comment with index number 945 contains dictionary wo
rd number 33, 44 and 114.
# The comma seperated numerical value represents the amount of times this word occours
 in the sentence.
corpus[945]
```

Out[ ]:

```
[(33, 1), (44, 1), (114, 2)]
```

In [ ]:

```
# Showing words from comment 945 from dictionary.
print(token_dic[33],',', token_dic[44],',', token_dic[114])
```

```
not , woman , hate
```

In [ ]:

```
# Showing the full comment from which the dictionary and corpus pulls info from.
# Notice how the word 'hate' appears twice.
miso_data['body'].iloc[945]
```

Out[ ]:

```
'I dont hate myself. I hate everyone else(men and women).'
```

In [ ]:

```
# Converting the corpus model to a TF-IDF model
tfidf = TfidfModel(corpus)
```

In [ ]:

```
# Calling the corpus list part of the tfidf model
tfidf_corpus = tfidf[corpus]
```

In [ ]:

```
# Generating topics from corpus list
lsi = LsiModel(tfidf_corpus, id2word=token_dic, num_topics=10)
```

In [ ]:

```
# Displaying top 5 topics
# The following topics shows the unsupervised ML representation of topics, for us thoug
h, the individual correlations and words does seem a bit ambigous
# Therefore, it would be preferable to cluster the data to see a more visual representa
tion of the UML.
lsi.show_topics(num_topics=5)
```

Out[ ]:

```
[(0,
  '0.281*"woman" + 0.233*"not" + 0.227*"man" + 0.197*"like" + 0.160*"want"
+ 0.150*"think" + 0.146*"girl" + 0.135*"good" + 0.134*"time" + 0.123*"gu
y"'),
 (1,
  '0.368*"theredpill" + 0.323*"bot" + 0.259*"moderator" + 0.240*"contact"
+ 0.231*"r" + 0.224*"question" + 0.204*"perform" + 0.204*"automatically" +
0.200*"concern" + 0.191*"admin"'),
 (2,
  '-0.671*"woman" + -0.455*"man" + 0.231*"not" + 0.129*"be" + 0.117*"good"
+ 0.103*"fuck" + 0.090*"get" + 0.090*"day" + 0.084*"time" + 0.080*"go"'),
 (3,
  '0.535*"not" + -0.344*"good" + 0.343*"fuck" + -0.252*"thank" + -0.175*"p
ost" + 0.151*"woman" + -0.143*"lol" + 0.138*"be" + -0.131*"girl" + -0.122
*"look"'),
 (4,
  '-0.714*"thank" + 0.258*"girl" + -0.244*"good" + 0.231*"lol" + -0.174*"n
ot" + -0.149*"post" + -0.147*"fuck" + 0.130*"like" + 0.129*"guy" + 0.108
*"nice"')]
```

In [ ]:

```
# Converting the tfidf corpus to LSI (Latent semantic analysis) for topic cluster model
ing.
# LSI is used to analyse data relationships based upon topics.
lsi_corpus = lsi[tfidf_corpus]
```

In [ ]:

```
# Create the document-topic-matrix
document_topic_matrix = MatrixSimilarity(lsi_corpus)
document_topic_matrix_ix = document_topic_matrix.index
```

In [ ]:

```
# Showing similarity of comments based on topics, examplified again for comment number
 945.
sims = document_topic_matrix[lsi_corpus[945]]
# Sorting the percentage similarity with the key of a lambda function for item 1 going
 through all observations.
sims = sorted(enumerate(sims), key=lambda item: item[1], reverse=True)
# Showing that comment 945 is the most similar to comment number 3110
print(sims[:10])
```

```
[(945, 0.99999994), (120, 0.9916045), (3110, 0.9897206), (5663, 0.9890498
5), (2572, 0.9876551), (990, 0.98746574), (4735, 0.9828154), (1821, 0.9776
441), (1917, 0.9770619), (239, 0.9765852)]
```

In [ ]:

```
print('\033[1m'+'Comment 945 reads:\n'+'\033[0m', '"'+ miso_data['body'][945] +'"', '\0
33[1m'+'\nAnd the most similar comment 3110 reads:\n'+'\033[0m','"'+miso_data['body'][3
110]+'"')
```

```
Comment 945 reads:
 "I dont hate myself. I hate everyone else(men and women)."
And the most similar comment 3110 reads:
 "Dont blame men for your lack of empathy when there wasnt much to begin w
ith. And in terms of whining and entitlement, white women have the olympic
gold on that despite being the most privileged demo in the history of ma
n."
```

Showing the similarity between these comments and displaying the actual text, these two comments may not be far off from misogyny, but its still hard to tell if the uml actually identified misogyny on its own.

Clustering

Since we now have an idea of how the misogynistic comments are structured through topic modelling, we want test an UML algorithm to cluster the comments together to see if it can identify the misogynistic comments in other ways than by use of the predefined categorical values.

In [ ]:

```
# Dimensionality reduction of data to plot in clusters, scaling the topic matrix index.
embeddings = umap.UMAP(n_neighbors=15, metric='cosine').fit_transform(document_topic_ma
trix_ix)
```

In [ ]:

```
# Choosing amount of clusters based upon the amount of catagorical outcomes; misogynism
and not misogynism.
clusterer = KMeans(n_clusters = 2)
# Fitting the data to the clustering
clusterer.fit(document_topic_matrix_ix)
```

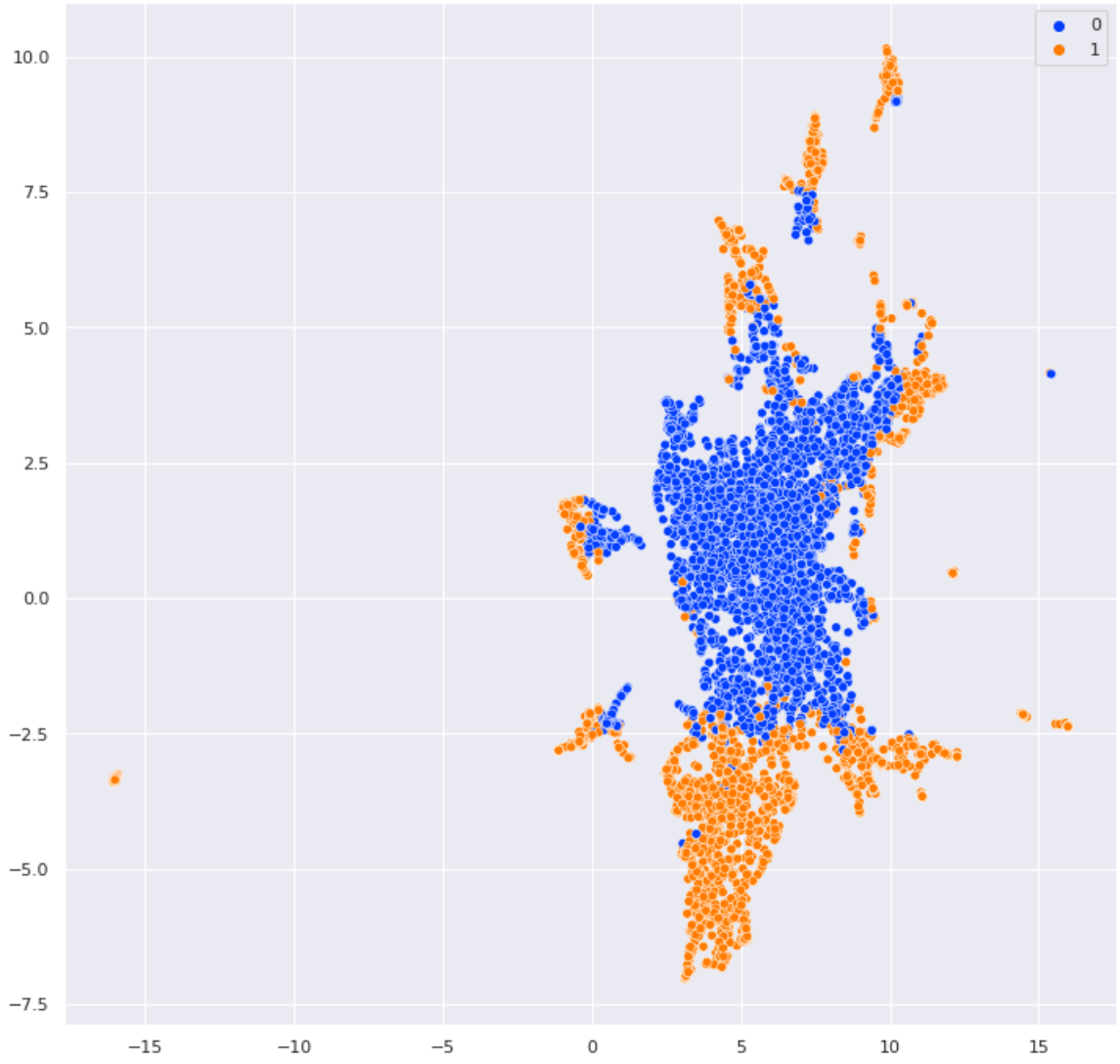
Out[ ]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

In [ ]:

```
# Plotting clusters, color based on UML clustering.
plt.rcParams.update({'font.size': 12})
plt.figure(figsize=(12,12))
g = sns.scatterplot(*embeddings.T,
                    hue=clusterer.labels_,
                     palette="bright",
                    legend='full')
```

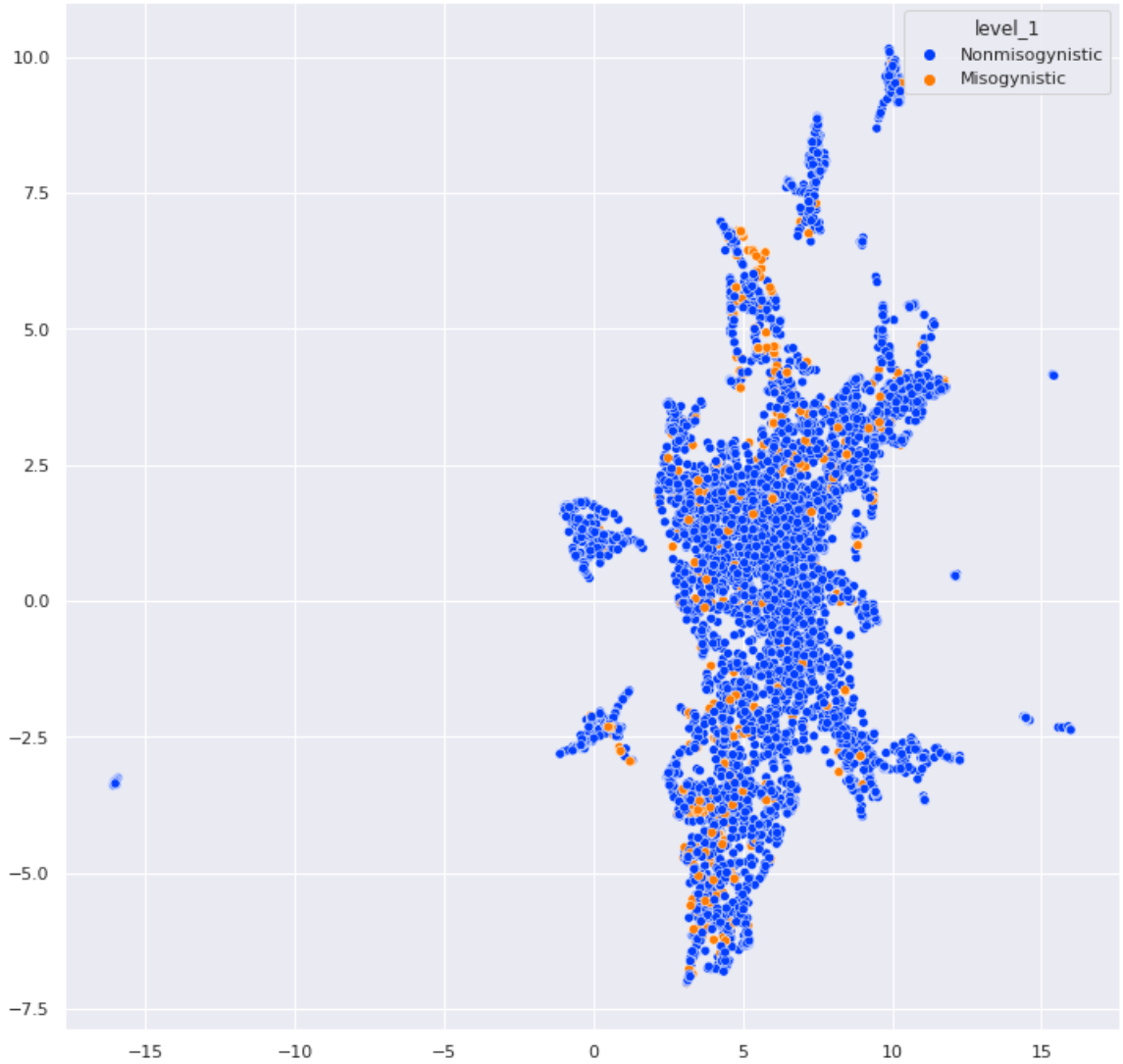
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWa
rning: Pass the following variables as keyword args: x, y. From version 0.
12, the only valid positional argument will be `data`, and passing other a
rguments without an explicit keyword will result in an error or misinterpr
etation.
  FutureWarning
```


0
1
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−15
−10
−5
0
5
10
15

In [ ]:

```
# Plotting clusters, color based on UML clustering. Showing clustering coloring based o
n misogynistic labeled comments.
plt.rcParams.update({'font.size': 12})
plt.figure(figsize=(12,12))
g = sns.scatterplot(*embeddings.T,
                    hue=miso_data['level_1'],
                     palette="bright",
                    legend='full')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWa
rning: Pass the following variables as keyword args: x, y. From version 0.
12, the only valid positional argument will be `data`, and passing other a
rguments without an explicit keyword will result in an error or misinterpr
etation.
  FutureWarning
```



As seen in the above clusters, the UML models used were not particularly able to cluster up the misogynistic comments. You can see in the top cluster that it has a big central cluster with a surrounding cluster, but when we relabel the data in the scatterplot for the categorical comment tags, it becomes clear that the non misogynistic comments are widespread in the cluster and there's no intuitive reasoning for the clustering.

There can be several reasons why this happened, such as our data set being particularly small and the algorithm not really having enough misogynistic comments to identify. It can also be argued that misogyny has a very specific structure where its hard to identify because the definition of misogyny is subjective. I.e. a personal attack, will not be tagged as misogyny. The comment has to specifically be targeting women as a gender and not just target a specific woman, which could make it hard to differentiate.

Supervised machine learning

By using supervised machine learning, we can create a model that predicts the probability of the comments of the dataset being labeled as misogynistic using the reddit comment data.

Logistic regression

The reason we have chosen a regression model, is that we want to predict whether or not a specific comment from the data set is misogynistic or nonmisogynistic. Therefore, we are talking about a classification problem where the response variable is either true/false etc.

For test size, we chose 5% of the dataset, which indicate a rather small test size. Reason being that we want to train the model as much as possible because we want to use it on another dataset. Another argument is that the amount of misogynistic compared to nonmisogynistic is rather small, so we ought to find a test seed with enough misogynistic comments.

In [ ]:

```
# Labeling the "level_1" column with labelencoder to use it in the crosstab as our cata
gorical variable
Y_cat = miso_data['level_1']
labelencoder_y = LabelEncoder()
Y_cat = labelencoder_y.fit_transform(Y_cat)
```

In [ ]:

```
# Splitting the data to test and train sets, using 95% for training and 5% for testing.
X_train, X_test, Y_train, Y_test = train_test_split(X, Y_cat, test_size = 0.05, random_
state = 2**32-1)
```

In [ ]:

```
# Running the logistical regression model as the default binary classification because
 we only have 2 catagorical variables.
model = LogisticRegression()
model = model.fit(X_train, Y_train)
```

In [ ]:

```
# Showing the R^2 score of the model, showing a prediction accuracy of ~91%
# Though, this is not very representative for the acutal accuracy because of the focus
 on misogynistic comments.
# Whereas this score is only representative of the overall accuracy.
model.score(X_test, Y_test)
```

Out[ ]:

```
0.9099378881987578
```

In [ ]:

```
# Displaying the probability predictions of the comments.
# The numerical probabilities is shown as probability for misogynism on the left and vi
ce versa
# Though this is better shown in a crosstab.
model.predict_proba(X_test)[:10]
```

Out[ ]:

```
array([[0.05567812, 0.94432188],
       [0.07604822, 0.92395178],
       [0.04986013, 0.95013987],
       [0.04885254, 0.95114746],
       [0.03768229, 0.96231771],
       [0.5562568 , 0.4437432 ],
       [0.02919138, 0.97080862],
       [0.14260556, 0.85739444],
       [0.06071131, 0.93928869],
       [0.03710111, 0.96289889]])
```

In [ ]:

```
# Transforming the categorical variables back to the original labels 'Misogynistic' and
'Nonmisogynistic' for display in the crosstab.
true_cat = labelencoder_y.inverse_transform(Y_test)
predicted_cat = labelencoder_y.inverse_transform(model.predict(X_test))

# Convert the data back into a dataframe.
data_pred_true = pd.DataFrame({'predicted_cat': predicted_cat, 'true_cat': true_cat})
```

In [ ]:

```
# Displaying crosstab for predicted and actual categorical placement of comments:
pd.crosstab(data_pred_true.true_cat, data_pred_true.predicted_cat)
```

Out[ ]:

predicted_cat true_cat	Misogynistic	Nonmisogynistic
Misogynistic	11	28
Nonmisogynistic	1	282

In [ ]:

```
# Showing a classification report for the predictions:
print(classification_report(true_cat,predicted_cat, labels=labelencoder_y.classes_))
```

```
                precision    recall  f1-score   support

   Misogynistic       0.92      0.28      0.43        39
Nonmisogynistic       0.91      1.00      0.95       283

       accuracy                           0.91       322
      macro avg       0.91      0.64      0.69       322
   weighted avg       0.91      0.91      0.89       322
```

As shown above the r^2 score relatively high, 90%, but looking at the crosstab and the classification report we recognize a problem; the recall from the classification report is 0.28 which is the measure of how great our model is at identifying a true positive. In this case 0.28 in recall is not a great result if we will use this model on another dataset. Therefore, we will try to use another algorithm to test the data on.

XGB model

XGB or Extreme Gradiant Boosting is a decision tree algorithm, which is efficient at handling sparse data. This should result in it being good in the case of our purpose of classifying on a binary categorical value as well as the data consisting of only around 10% of misogynistic comments. Whereas we will use the XGB classifier model for this purpose.

In [ ]:

```
# Same procedure as the logistical regression, but using XGB instead.
Y_xgb = miso_data['level_1']
labelencoder_y = LabelEncoder()
Y_xgb = labelencoder_y.fit_transform(Y_xgb)
```

In [ ]:

```
# Splitting the data using the assumptions as before
X_train_xgb, X_test_xgb, Y_train_xgb, Y_test_xgb = train_test_split(X, Y_xgb, test_size
= 0.05, random_state = 2**32-1)
```

In [ ]:

```
# Running the XGB model with default settings
model_xgb = XGBClassifier()
model_xgb.fit(X_train_xgb,Y_train_xgb)
Y_pred_xgb = model_xgb.predict(X_test_xgb)
model_xgb.score(X_test_xgb, Y_test_xgb)
```

Out[ ]:

```
0.9347826086956522
```

This appears slightly more precise compared with the logistical regression that was at ~91%, therefore the XGB model is approximatly 2.5 percentages points better. But we still want to inspect it visually by using a cross tab and get a classification report

In [ ]:

```
true_explained_xgb = labelencoder_y.inverse_transform(Y_test_xgb)
predicted_explained_xgb = labelencoder_y.inverse_transform(model_xgb.predict(X_test_xgb
))

# Convert the data back into a dataframe.
data_pred_true_xgb = pd.DataFrame({'predicted_explained_xgb': predicted_explained_xgb,
'true_explained_xgb': true_explained_xgb})

# Plot using a crosstab.
pd.crosstab(data_pred_true_xgb.true_explained_xgb, data_pred_true_xgb.predicted_explain
ed_xgb)
```

Out[ ]:

predicted_explained_xgb	Misogynistic	Nonmisogynistic
true_explained_xgb		
Misogynistic	22	17
Nonmisogynistic	4	279

In [ ]:

```
print(classification_report(true_explained_xgb,predicted_explained_xgb, labels=labelenc
oder_y.classes_))
```

```
                precision    recall  f1-score   support

   Misogynistic       0.85      0.56      0.68        39
Nonmisogynistic       0.94      0.99      0.96       283

       accuracy                           0.93       322
      macro avg       0.89      0.77      0.82       322
   weighted avg       0.93      0.93      0.93       322
```

Contrary to the regular logistic regression model, the XGB classifier on default settings shows a more promising result, as the recall of identifying misogynistic comments has increased by 0.28 or 11 observations on the same split.

A slight decrease in overall f1-score for the nonmisogynistic category has no alarming effect on the results, caused by the intention of the model; we seek to tag potentially misogynistic comments for review, whereas it is expected to contain a few false positives.

Given the above classification report and crosstab is representative as a sample size of a ‘real’ implication of the model, we are still only getting slightly above 50% of the misogynistic comments flagged, even with the XGB model.

Were these comments to be deemed inappropriate on SoMe platforms, we would prefer to have more true positives with respect to identifying misogynism rather than letting half of them pass through the algorithm.

Hyperparameter tuning the XGB model is deemed preferable to increase recall score on the misogynistic category

Hyperparameter tuning of XGB

Hyperparameter tuning is effectively tweaking the settings of the, in our case, XGB model as to make it better in our specific case of prediction. This may or may not increase the model's efficiency, precision and accuracy dependent on which settings are tuned. Tuning the model calls for testing by trial and error, and in the end, a model that scores better after tuning may be prone to overfitting.

In [ ]:

```
# Tuning for max depth, estimators, gamma and scale pos weight
# Max depth was selected based on a typical value of 3-10, though our chosen value of 1
0 may cause overfitting.
# n_estimators was selected based on www.analyticsvidhya.com, who states amount of tree
s (n_estimators) should be an amount the computing processor can handle for consistency
of the model.
# Gamma was selected as 1 to make the model more conservative, reducing the threshold f
or minimum loss reduction for making a split.
# Scale_pos_weight was tuned because of the imbalance in the class scales.
  # Stated on a XGBoost guidepage, we found that a preferable value would be the positi
ve instances divided by the negative instances.
xgb_tuned=xgb.XGBClassifier(max_depth=10, n_estimators=600, gamma=1, scale_pos_weight=2
80/22)
```

In [ ]:

```
# Displaying the new R^2 score for the tuned xgb model
xgb_tuned.fit(X_train_xgb,Y_train_xgb)
Y_pred_xgb = xgb_tuned.predict(X_test_xgb)
xgb_tuned.score(X_test_xgb, Y_test_xgb)
```

Out[ ]:

```
0.9503105590062112
```

In [ ]:

```
#Processing the data as before, preparing it for plotting in a crosstab
true_explained_xgb = labelencoder_y.inverse_transform(Y_test_xgb)
predicted_explained_xgb = labelencoder_y.inverse_transform(xgb_tuned.predict(X_test_xgb
))

data_pred_true_xgb = pd.DataFrame({'predicted_explained_xgb': predicted_explained_xgb,
'true_explained_xgb': true_explained_xgb})

pd.crosstab(data_pred_true_xgb.true_explained_xgb, data_pred_true_xgb.predicted_explain
ed_xgb)
```

Out[ ]:

predicted_explained_xgb true_explained_xgb	Misogynistic	Nonmisogynistic
Misogynistic	26	13
Nonmisogynistic	3	280

In [ ]:

```
print(classification_report(true_explained_xgb,predicted_explained_xgb, labels=labelenc
oder_y.classes_))
```

```
                precision    recall  f1-score   support

   Misogynistic       0.90      0.67      0.76        39
Nonmisogynistic       0.96      0.99      0.97       283

       accuracy                           0.95       322
      macro avg       0.93      0.83      0.87       322
   weighted avg       0.95      0.95      0.95       322
```

Through multiple run-throughs of different tuning, we found settings that made the model better at separating misogynistic comments from nonmisogynistic on the same test/train split as before. These new results concluded, as seen above in the classification report, at a recall of 0.67, an increase of 0.11. Causing the overall r^2 score to increase by ~1.5 percentage points as well.

This improvement may seem good, though we must keep in mind the potential for overfitting to the reddit comments dataset, which is why we wish to test both the tuned and untuned XGB models on a new dataset to test the reliability of the models.

Applying the SML models to new data

Now that we have a working machine learning model, we choose to move away from our training data so we could give it a chance to work with outside data.

In this case we have chosen to use the twitter hate speech dataset utilized in earlier assignments, as we think this dataset might fit well with what we are trying to achieve.

The dataset contains a lot of hateful tweets targeted at specific demographics, which gives us a chance to see if the model is able to specifically identify if there are tweets which are targeting females in a misogynistic sense.

It is also a somewhat realistic environment for an algorithm as the goal was to have a model which could go through social media posts in an attempt to identify misogynistic statement so that they can be removed.

Preprocessing twitter data

In [ ]:

```
#Loading hate speech tweets dataset
data_tweets = pd.read_csv('https://github.com/SDS-AAU/SDS-master/raw/master/M2/data/twi
tter_hate.zip')
```

In [ ]:

```
# Setting preprocess options to handle the tweets format
prepro.set_options(prepro.OPT.URL, prepro.OPT.NUMBER, prepro.OPT.RESERVED, prepro.OPT.M
ENTION, prepro.OPT.SMILEY)
```

In [ ]:

```
# Running the data through a cleaner, removing hashtags and normalizing the tweets.
def preprocessTweets(data_tweets):
  clean_text = data_tweets.map(lambda t: prepro.clean(t))
  clean_text = clean_text.str.replace('#','')
  return vectorizer.transform(clean_text)
```

In [ ]:

```
# Removing unnecessary columns from the dataset.
data_tweets.drop(['class','Unnamed: 0'], axis=1, inplace=True)
```

In [ ]:

```
# Single out the preprocessed tweets for predicting
X_new_tweets = preprocessTweets(data_tweets['tweet'])
```

Testing the un-tuned XGB model on the hatespeech twitter data.

By applying the untuned model to the new twitter hate speech dataset, we can compare this to the tuned model as a control to indicate if the new model is potentially overfitted to the reddit comments dataset.

In [ ]:

```
# Running tweets through untuned model
predictions_new_tweets = model_xgb.predict_proba(X_new_tweets)
```

In [ ]:

```
# Adding the predictions to new columns in the dataframe
data_tweets['misogynistic'] = predictions_new_tweets[:,0]
data_tweets['nonmisogynistic'] = predictions_new_tweets[:,1]
```

In [ ]:

```
# Creating a dataframe with only the tweets and the probability scores for displaying:
sorted_prob = pd.DataFrame({'tweet': data_tweets['tweet'], 'misogynistic': data_tweets[
'misogynistic'],
                            'nonmisogynistic': data_tweets['nonmisogynistic']})
```

In [ ]:

```
# Display settings for text visualization to allow more text on screen.
pd.set_option('display.max_colwidth', 120)
```

In [ ]:

```
# Sorting for misogynism probability, displaying the tweet with the highest probability
of being in the misogynistic category:
sorted_prob = sorted_prob.sort_values('misogynistic', ascending=False)
sorted_prob.head()
```

Out[ ]:

	tweet	misogynistic	nonmisogynistic
14844	RT @D0PESHIT0NLY: Bitches be having\nNo job\nNo future\nNo education\n& have the nerve to say "all these niggas ...	0.998700	0.001299
13311	Niggas already acting like bitches so why not get a real bitch right? Hell if you gone act like a pussy I rather jus...	0.998328	0.001672
5338	@_Perarl bitch all these thick bitches with they ass out on TV , do what you want	0.992617	0.007383
20156	RT @thecoreyholcomb: When gay girls suck pussy from a bitch who just took a plan B pill the other day that means the...	0.992174	0.007826
2968	@DarienDaywalt bitch shut the fuck up goddam your a slut bitch whore nigga	0.992109	0.007891

In [ ]:

```
#Displaying the full top misogynistic tweet.
data_tweets['tweet'][14844]
```

Out[ ]:

```
'RT @D0PESHIT0NLY: Bitches be having\nNo job\nNo future\nNo education\n&am
p; have the nerve to say "all these niggas want is pussy"\nBitch cause tha
t&#8230;'
```

In [ ]:

```
# Create new column for defining whether or not a comment is flagged as misogynistic on
a True/False boolean.
# Threshold for misogynism is based upon the default threshold for the XGB model of 5
0%.
data_tweets['misogonystic_comment?'] = (data_tweets[['misogynistic']]>= 0.5)
```

In [ ]:

```
# Counting the amount of misogynistic and nonmisogynistic tweets based on the threshol
d.
Counter(data_tweets['misogonystic_comment?'])
```

Out[ ]:

```
Counter({False: 12289, True: 12494})
```

Displaying the tweet with the highest probability for misogynistic content, we can observe that it generalizes women as, figuratively, having no future prospects but blaming men on only wanting intercourse.

Though this example may seem like an example of misogyny, making conclusions upon this may be incorrect as it may be context deprived and in the context of i.e. a specific demography of women, which in of itself can be concerning in of itself. It is therefore up to subjective matter to tag tweets phrased as such.

Testing the tuned XGB model on the hatespeech twitter data.

Applying the tuned XGB model, we would preferably have even more tweets categorized for hatespeech, assuming that these weren't already represented in the untuned XGB model on the tweet data.

This model should be better at differentiating between misogynistic and nonmisogynistic comments/tweets, given that we didn't overfit the model in the hyperparameter tuning process.

In [ ]:

```
# Predicting on the tweets, using the tuned XGB model
predictions_new_tweets = xgb_tuned.predict_proba(X_new_tweets)
```

In [ ]:

```
# Adding the predictions to new columns in the dataframe
data_tweets['misogynistic'] = predictions_new_tweets[:,0]
data_tweets['nonmisogynistic'] = predictions_new_tweets[:,1]
```

In [ ]:

```
# Creating a dataframe with only the tweets and the probability scores for displaying:
sorted_prob = pd.DataFrame({'tweet': data_tweets['tweet'], 'misogynistic': data_tweets[
'misogynistic'],
                            'nonmisogynistic': data_tweets['nonmisogynistic']})
```

In [ ]:

```
# Sorting for misogynism probability, displaying the tweet with the highest probability
of being in the misogynistic category:
sorted_prob = sorted_prob.sort_values('misogynistic', ascending=False)
sorted_prob.head()
```

Out[ ]:

	tweet	misogynistic	nonmisogynistic
13311	Niggas already acting like bitches so why not get a real bitch right? Hell if you gone act like a pussy I rather jus...	0.999949	0.000051
20890	Slut ass queen bee bitch.	0.999842	0.000158
8045	Bitch duck on whore ass slut cunt ha Biden it's Gucci. Tho whatever. Nigea @mdsaab1 @t_jawad89	0.999838	0.000162
3541	@IvanJ_Reyes, um no that only a fuck you bitch and done xavier will hopefully respect girls bc I am doing it alone a...	0.999801	0.000199
10248	I don't want no Philly disease lol Thot bitch	0.999785	0.000215

In [ ]:

```
# With the tuned model, we get a different outcome on most probable misogynistic tweet
 which is displayed below:
data_tweets['tweet'][13311]
```

Out[ ]:

```
'Niggas already acting like bitches so why not get a real bitch right? Hel
l if you gone act like a pussy I rather just have one. Shit.'
```

In [ ]:

```
# Create new column for defining whether or not a comment is flagged as misogynistic on
a True/False boolean.
# Threshold for misogynism is based upon the default threshold for the XGB model of 5
0%.
data_tweets['misogonystic_comment?'] = (data_tweets[['misogynistic']]>= 0.5)
```

In [ ]:

```
# Counting the amount of misogynistic and nonmisogynistic tweets based on the threshol
d.
Counter(data_tweets['misogonystic_comment?'])
```

Out[ ]:

```
Counter({False: 12340, True: 12443})
```

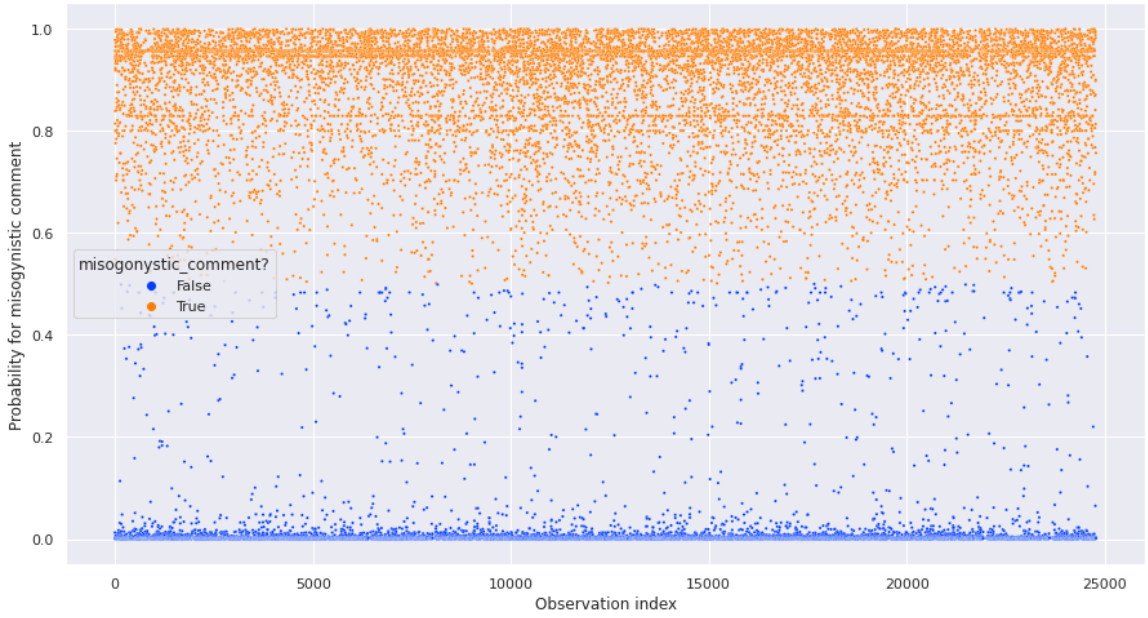
In [ ]:

```
# Making a scatterplot to show the distribution of the tweets.
xyz = sns.scatterplot(data_tweets.index, data_tweets['misogynistic'],
                      hue=data_tweets['misogonystic_comment?'],
                       palette="bright",
                      legend='full', s=5)
plt.ylabel('Probability for misogynistic comment')
plt.xlabel('Observation index')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWa
rning: Pass the following variables as keyword args: x, y. From version 0.
12, the only valid positional argument will be `data`, and passing other a
rguments without an explicit keyword will result in an error or misinterpr
etation.
  FutureWarning
```

Out[ ]:

```
Text(0.5, 0, 'Observation index')
```



As shown above in the scatterplot the points are tagged as misogynistic (orange) and nonmisogynistic (blue). The scatterplot indicates that the tweets which are tagged orange has a greater spread compared to the tweets which does not.

We noticed the straight line passing through the 0.8 percent probability mark and will investigate this later on.

In [ ]:

```
# Adding the boolean column for misogynistic and nonmisogynistic tags on tweets to the
 dataframe sorted by probability for misogynism.
sorted_prob['misogonystic_comment?'] = data_tweets['misogonystic_comment?']
```

In [ ]:

```
# resetting index, causing the first indexed observation to be the highest probability
 for misogynism
sorted_prob = sorted_prob.reset_index(drop=True)
```

In [ ]:

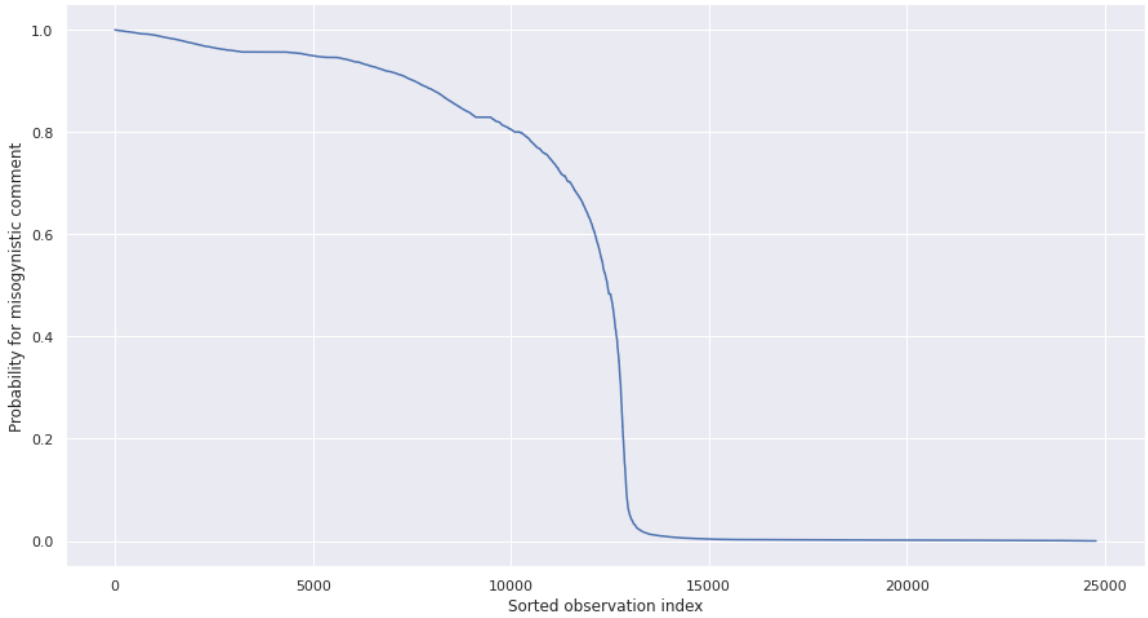
```
# Lineplotting based on sorted observations for highest misogynistic probability.
xyz = sns.lineplot(sorted_prob.index, sorted_prob['misogynistic'],

                   palette="bright",
                  legend='full')
plt.ylabel('Probability for misogynistic comment')
plt.xlabel('Sorted observation index')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWa
rning: Pass the following variables as keyword args: x, y. From version 0.
12, the only valid positional argument will be `data`, and passing other a
rguments without an explicit keyword will result in an error or misinterpr
etation.
  FutureWarning
```

Out[ ]:

```
Text(0.5, 0, 'Sorted observation index')
```



Ordering the index of the data from highest probability of being misogynistic we created the line graph above which is representative of the tweets categorical probability distribution.

It shows that the model is able to identify about 10000 misogynistic tweets with an above 80% certainty. In this graph though we can again identify that the model is less certain about what is misogyny compared to nonmisgynistic. Because we can observe the first half of the observations where the probability of being misogynistic steadily falls while nearing the middle of the data, where the probability then dives straight down to near 0%, where the model is extremely certain that these comments are not misogynistic.

Misogynism can be a varied topic because there's different perspectives on the definition. The basic definition always have focus on women, which might explain why there are so many tweets which the model is sure isn't misogyny, because the tweets simply does not include women.

Regarding the threshold for which tweets are tagged as misogynistic, we chose the default of XGB to be 50% probability. We do think it might be worth to increase this threshold a fair bit because the model has a high level of uncertainty for a lot of tweets. As such we should maybe consider using this model with a 70-80% threshold to be sure the tweets are misogynistic as we cant be confident that all the 50-70% probability tweets necessarily are misogynistic, as they are very spread out compared to the top probability tweets, as shown in the top scatterplot.

Furthermore we can identify in this graph as well, that the flat line at 80% consist of a bunch of comments which the model thinks are very equal chance of being misogynistic, which might be because of duplicate comments and as such, we have taken some steps to verify whether this is the case:

In [ ]:

```
# Trying to find why there is a flatline of probability on some observations,
# We figured that there may be dublicated tweets in the dataset:
sorted_prob['tweet'].nunique()
```

Out[ ]:

```
24783
```

In [ ]:

```
# All tweets seem to be unique
sorted_prob.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24783 entries, 0 to 24782
Data columns (total 4 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   tweet                 24783 non-null  object
 1   misogynistic          24783 non-null  float32
 2   nonmisogynistic       24783 non-null  float32
 3   misogonystic_comment?  24783 non-null  bool
dtypes: bool(1), float32(2), object(1)
memory usage: 411.6+ KB
```

In [ ]:

```
# We can see that the occurance of the specific probability of ~80.3% is 405 times as w
ell as a lot of dublicate probability values for the upper and lower bounds.
proba_count = Counter(sorted_prob['misogynistic'])
proba_count.most_common(5)
```

Out[ ]:

```
[(0.001642763614654541, 2230),
 (0.9566750526428223, 1108),
 (0.8291589021682739, 381),
 (0.0022307634353637695, 336),
 (0.9461961388587952, 262)]
```

Therefore, the tweets must share same style of language of which the XGB model treats these tweets equally. All seems to be in order.

Conclusion

To conclude, we have trained the XGB model to attempt to identify misogynistic comments based on an expert annotated reddit dataset. Out of the tested models, we chose the XGB hyperparameter-tuned model because of the lack of misogynistic comments and we wanted to use the model with the highest recall of our tested models. By tuning the parameters of the model we managed to get the following classification report for finding misogynistic and non misogynistic sentiment in short texts:

In [ ]:

```
print(classification_report(true_explained_xgb,predicted_explained_xgb, labels=labelenc
oder_y.classes_))
```

```
                precision    recall  f1-score   support

   Misogynistic       0.90      0.67      0.76        39
Nonmisogynistic       0.96      0.99      0.97       283

       accuracy                           0.95       322
      macro avg       0.93      0.83      0.87       322
   weighted avg       0.95      0.95      0.95       322
```

For the practical implications of the model we think it could be used on a variety of social media platforms looking to filter out misogynistic commentary. We realise that training data for the model does not contain all of the needed categories for a standalone moderation model, as it will probably be a requirement to find other forms of derogatory text such as racism. With that being said this model could easily be fitted into a more overarching moderation tool where finding misogynistic comments would be valuable.

Talking about the specific thresholds we've set up in the model these would also need to be modified on a platform level depending on the goal of the platform, because as discussed throughout the notebook, misogynism is subjective, and as such some platforms might want to put the threshold higher, to not punish users too harshly. An example we thought of could be: Set the overall threshold to 75% where comments over 90% would get automatically removed while comments in the 75-90% range would be flagged to get manually reviewed by a moderator. This could enable the platform to automatically have clear cut cases removed, while still leaving room for interpretation on the less clear cases.

This would of course depend on the goal of the platform. Reddit can in this case be a great example because it is divided into smaller sub-forums with their own individual rules. Here some subreddits will have different moderation goals, and as an example subreddits tailored towards women might have higher standards for what they consider misogynistic, and how to deal with it.

So to summarize, we have created a model which is intended to be used by social media platforms to moderate based on locating misogynistic sentiment in text. The output of the model can change depending on the goals of the platform to specifically tailor the task they are trying to achieve. Be that automatically removing misogynistic comments or simply tagging the comments for other systems / review processes.

With that being said, there are some caveats we have considered, which are the following:

Caveats

- Firstly, it is important to state that the amount of observations from the Reddit comments that are recognised as misogynism is quite sparse. It creates the possibility that the bag-of- words (dictionary) in our model doesn't know how to differentiate whether a specific comment/tweet might be categorised as a misogynistic comment. Therefore, it would be beneficial if we would have added more custom misogynistic words for the model to associate. This could be "women = bird", to increase the chance of the model to predict the sentiment in the sentence by having a more widespread dictionary, as many communities will use different language and slang.
- Secondly, it is important to state that a specific comment in the Reddit dataset used for training, is placed in their category based on subjectivity of the annotaters. Even though the article that created the dataset wanted to use some generic definitions of misogynism, and was judged by some experts, it still creates a possibility of a subjective evaluation of how the words are placed in a specific sentence. Therefore, we think it's important to consider the fallacy of this subjectivism, as the end user might have a different understanding of misogynism, compared to the annotaters.
- Thirdly, the idea of creating a model that could define whether or not a comment is misogynistic is good, but we need to make sure that we are creating a more reliable model which does not contain this amount of uncertainty. We could improve the reliability of the model by improving the bag-of-words. This could also have been done by increasing the amount and variety of training data, such as text with a derogatory sentiment towards other entities than women, as well as text data from other social media platforms.
- Lastly, we are using two different datasets where there might be some cultural differences because of what kind of communication is accepted on Twitter compared to Reddit. Therefore, we can't take the reported result for granted. It seems like Reddit in comparison to Twitter is more moderated overall, with fewer people expressing themselves negatively towards others. This goes in line with the fact that reddit "outsources" their subreddit moderation to users in the community, whereas twitter is one big social platform.

In [2]:

```
!jupyter nbconvert --to html "/content/M2_Exam_notebook (1).ipynb"
```

```
[NbConvertApp] Converting notebook /content/M2_Exam_notebook (1).ipynb to
html
[NbConvertApp] Writing 1574662 bytes to /content/M2_Exam_notebook (1).html
```