Final Project

Group HOMEWORK. This final project can be collaborative. The maximum members of a group is 2. You can also work by yourself. Please respect the academic integrity. **Remember: if you get caught on cheating, you get F.**

A Introduction to the competition

Sexism is a growing problem online. It can inflict harm on women who are targeted, make online spaces inaccessible and unwelcoming, and perpetuate social asymmetries and injustices. Automated tools are now widely deployed to find, and assess sexist content at scale but most only give classifications for generic, high-level categories, with no further explanation. Flagging what is sexist content and also explaining why it is sexist improves interpretability, trust and understanding of the decisions that automated tools use, empowering both users and moderators.

This project is based on SemEval 2023 - Task 10 - Explainable Detection of Online Sexism (EDOS). Here you can find a detailed introduction to this task.

You only need to complete **TASK A - Binary Sexism Detection: a two-class (or binary) classification where systems have to predict whether a post is sexist or not sexist**. To cut down training time, we only use a subset of the original dataset (5k out of 20k). The dataset can be found in the same folder.

Different from our previous homework, this competition gives you great flexibility (and very few hints), you can determine:

- how to preprocess the input text (e.g., remove emoji, remove stopwords, text lemmatization and stemming, etc.);
- which method to use to encode text features (e.g., TF-IDF, N-grams, Word2vec, GloVe, Part-of-Speech (POS), etc.);

- which model to use.

Requirements

- **Input**: the text for each instance.
- **Output**: the binary label for each instance.
- **Feature engineering**: use at least 2 different methods to extract features and encode text into numerical values.
- **Model selection**: implement with at least 3 different models and compare their performance.
- **Evaluation**: create a dataframe with rows indicating feature+model and columns indicating Precision, Accuracy and F1-score (using weighted average). Your results should have at least 6 rows (2 feature engineering methods x 3 models). Report best performance with (1) your feature engineering method, and (2) the model you choose.
- **Format**: add explainations for each step (you can add markdown cells). At the end of the report, write a summary and answer the following questions:
  - What preprocessing steps do you follow?
  - How do you select the features from the inputs?
  - Which model you use and what is the structure of your model?
  - How do you train your model?
  - What is the performance of your best model?
  - What other models or feature engineering methods would you like to implement in the future?
- **Two Rules**, violations will result in 0 points in the grade:
  - Not allowed to use test set in the training: You CANNOT use any of the instances from test set in the training process.
  - Not allowed to use any generative AI (e.g., ChatGPT).

Evaluation

The performance should be only evaluated on the test set (a total of 1086 instances). Please split original dataset into train set and test set. The test set should NEVER be used in the training process. The evaluation metric is a combination of precision, recall, and f1-score (use `classification_report` in sklearn).

The total points are 10.0. Each team will compete with other teams in the class on their best performance. Points will be deducted if not following the requirements above.

If ALL the requirements are met:

- Top 25% teams: 10.0 points.
- Top 25% - 50% teams: 8.5 points.
- Top 50% - 75% teams: 7.0 points.

- Top 75% - 100% teams: 6.0 points.

If your best performance is above 0.80 (weighted F1-score) and meets all the requirements, you will also get full points (10.0 points).

⭐ Bonus points will be awarded to top 5 teams (ranked by weighted F1-score):

- Top 1 team: 3pt adding to final grade
- Top 2 team: 2pt adding to final grade
- Top 3-5 teams: 1pt adding to final grade

Submission

Similar as homework, submit both a PDF and .ipynb version of the report.

The report should include:

- (a)code AND outputs
- (b)explainations for each step
- (c)individual experimental results AND combine them in a table
- (d)summary

The due date is **May 2, Thursday by 11:59pm**.

In [4]:
```
import pandas as pd
import re
import nltk
from nltk.tokenize import TweetTokenizer
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

nltk.download('wordnet')
nltk.download('stopwords')

def remove_noise(text):
    text = re.sub(r'https?:\/\/.*[\r\n]*', '', text)  # remove links
    text = re.sub('<.*?>', '', text)  # remove HTML tags
    text = re.sub(r'[^\w\s]', '', text)  # remove punctuation
    text = re.sub(r'\d+', '', text)  # remove numbers
    return text

def normalize_case(text):
    return text.lower()

tokenizer = TweetTokenizer(preserve_case=False, strip_handles=True, reduce_len=True

def tokenize(text):
    return tokenizer.tokenize(text)

stop_words = set(stopwords.words('english'))
```

```
def remove_stopwords(tokens):
    return [word for word in tokens if not word in stop_words]

def process_hashtags_and_mentions(text):
    text = re.sub(r'@\w+', '', text)  # remove mentions
    text = re.sub(r'#', '', text)  # remove hashtag symbol but keep the text
    return text

lemmatizer = WordNetLemmatizer()

def lemmatize(tokens):
    return [lemmatizer.lemmatize(token) for token in tokens]

def preprocess_tweet(tweet):
    tweet = remove_noise(tweet)
    tweet = normalize_case(tweet)
    tweet = process_hashtags_and_mentions(tweet)
    tokens = tokenize(tweet)
    tokens = remove_stopwords(tokens)
    tokens = lemmatize(tokens)
    return ' '.join(tokens)  # Convert tokens back to string

# Read data from CSV file
df = pd.read_csv('edos_labelled_data.csv')

# Split the DataFrame into training and test sets based on the 'split' column
train_df = df[df['split'] == 'train'].copy()
test_df = df[df['split'] == 'test'].copy()

# Apply preprocessing to the 'text' column of both the training and test DataFrames
train_df['processed_tweet'] = train_df['text'].apply(preprocess_tweet)
test_df['processed_tweet'] = test_df['text'].apply(preprocess_tweet)

# Print the processed training DataFrame to see the results
print(train_df[['text', 'processed_tweet']])

# Print the processed test DataFrame to see the results
print(test_df[['text', 'processed_tweet']])

# Save the processed training data back into a new CSV file
train_df.to_csv('ProcessedTrainData.csv', index=False)

# Save the processed test data back into a new CSV file
test_df.to_csv('ProcessedTestData.csv', index=False)
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\hamad\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\hamad\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
                                                   text  \
1086  In Nigeria, if you rape a woman, the men rape ...
1087                         Then, she's a keeper. 😉
1088  This is like the Metallica video where the poo...
1089                                             woman?
1090                        I bet she wished she had a gun
...                                                 ...
5274  Supporting toxic men and glorifying toxic male...
5275  Find a girl with common beliefs. I have. They ...
5276  not to mention that she's an outright commie w...
5277  Only if you make it clear you're not looking f...
5278                       Yea. Most trans women hate men.

                                        processed_tweet
1086  nigeria rape woman men rape back nsfw nigeria ...
1087                                        shes keeper
1088  like metallica video poor mutilated bastard sa...
1089                                              woman
1090                                    bet wished gun
...                                                 ...
5274  supporting toxic men glorifying toxic male beh...
5275            find girl common belief consider friend
5276  mention shes outright commie made nomination i...
5277  make clear youre looking ltr youre misleading ...
5278                             yea trans woman hate men

[4193 rows x 2 columns]
                                                   text  \
0     Fuck the niggers and the jews. Both have a his...
1     Well then good because someone has to knock he...
2     #USA #Texas #Islam #Muslims #Islamization #Sha...
3     Yes, normal women want to be dominated. Social...
4     She didn't have to be a bitch about it. She li...
...                                                 ...
1081             So did you break it off with her then.
1082  In early middle school I was physically bullie...
1083  It was like a big sisterhood all stemming from...
1084  It goes like this: I'm on the dance floor and ...
1085  It could be like for the ladies' corner of you...

                                        processed_tweet
0     fuck nigger jew history eating white people fu...
1     well good someone knock as back line act like ...
2     usa texas islam muslim islamization sharialaw ...
3     yes normal woman want dominated social scienti...
4     didnt bitch literally went way rub rejection f...
...                                                 ...
1081                                              break
1082  early middle school physically bullied girl to...
1083  like big sisterhood stemming feminist bitch ol...
1084  go like im dance floor there girl seems checki...
1085  could like lady corner man cave could pause ch...

[1086 rows x 2 columns]
```

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, accuracy_score
from xgboost import XGBClassifier
import warnings

# Suppress all warnings
warnings.filterwarnings("ignore")

# Load data
train_data = pd.read_csv('ProcessedTrainData.csv')
test_data = pd.read_csv('ProcessedTestData.csv')

# Extract features and labels
x_train = train_data['processed_tweet']
y_train = train_data['label']
x_test = test_data['processed_tweet']
y_test = test_data['label']

# Encode labels
label_encoder = LabelEncoder()
y_train = label_encoder.fit_transform(y_train)
y_test = label_encoder.transform(y_test)

# Initialize vectorizers
tfidf_vectorizer = TfidfVectorizer()
count_vectorizer = CountVectorizer()

# Vectorize data
X_train_tfidf = tfidf_vectorizer.fit_transform(x_train)
X_test_tfidf = tfidf_vectorizer.transform(x_test)
X_train_count = count_vectorizer.fit_transform(x_train)
X_test_count = count_vectorizer.transform(x_test)

# Define models with reproducibility settings where applicable
models = {
    'Logistic Regression': LogisticRegression(random_state=0, class_weight='balance
    'SVC': SVC(random_state=0, class_weight='balanced', probability=True),
    'XGBoost': XGBClassifier(use_label_encoder=False, eval_metric='logloss', random
}

# Function to evaluate models
def evaluate_models(models, X_train, X_test, y_train, y_test):
    results = {}
    for name, model in models.items():
        model.fit(X_train, y_train)
        predictions = model.predict(X_test)
        report = classification_report(y_test, predictions, output_dict=True)
        results[name] = {
            'Non-Sexist Precision': report['0']['precision'],
```

```
            'Non-Sexist Recall': report['0']['recall'],
            'Non-Sexist F1-Score': report['0']['f1-score'],
            'Sexist Precision': report['1']['precision'],
            'Sexist Recall': report['1']['recall'],
            'Sexist F1-Score': report['1']['f1-score'],
            'Weighted Precision': report['weighted avg']['precision'],
            'Weighted Recall': report['weighted avg']['recall'],
            'Weighted F1-Score': report['weighted avg']['f1-score']
        }
    return results

# Evaluate models on both TF-IDF and Count Vector data
results_tfidf = evaluate_models(models, X_train_tfidf, X_test_tfidf, y_train, y_tes
results_count = evaluate_models(models, X_train_count, X_test_count, y_train, y_tes

# Print results
print("Results with TF-IDF Vectorization:")
for model, metrics in results_tfidf.items():
    print(f"{model}: {metrics}")

print("\nResults with Count Vectorization:")
for model, metrics in results_count.items():
    print(f"{model}: {metrics}")
```

```
Results with TF-IDF Vectorization:
Logistic Regression: {'Non-Sexist Precision': 0.8697421981004071, 'Non-Sexist Recal
l': 0.8124207858048162, 'Non-Sexist F1-Score': 0.8401048492791612, 'Sexist Precisio
n': 0.5759312320916905, 'Sexist Recall': 0.6767676767676768, 'Sexist F1-Score': 0.62
22910216718266, 'Weighted Precision': 0.7893905803245426, 'Weighted Recall': 0.77532
22836095764, 'Weighted F1-Score': 0.7805369792981499}
SVC: {'Non-Sexist Precision': 0.8355342136854742, 'Non-Sexist Recall': 0.88212927756
65399, 'Non-Sexist F1-Score': 0.8581997533908755, 'Sexist Precision': 0.632411067193
6759, 'Sexist Recall': 0.5387205387205387, 'Sexist F1-Score': 0.5818181818181818, 'W
eighted Precision': 0.7799839609156178, 'Weighted Recall': 0.7882136279926335, 'Weig
hted F1-Score': 0.7826147379607742}
XGBoost: {'Non-Sexist Precision': 0.8342857142857143, 'Non-Sexist Recall': 0.9252217
997465145, 'Non-Sexist F1-Score': 0.877403846153846, 'Sexist Precision': 0.720379146
9194313, 'Sexist Recall': 0.5117845117845118, 'Sexist F1-Score': 0.5984251968503936,
'Weighted Precision': 0.8031344707242171, 'Weighted Recall': 0.8121546961325967, 'We
ighted F1-Score': 0.8011085801841173}

Results with Count Vectorization:
Logistic Regression: {'Non-Sexist Precision': 0.8611464968152867, 'Non-Sexist Recal
l': 0.8567807351077313, 'Non-Sexist F1-Score': 0.8589580686149936, 'Sexist Precisio
n': 0.6245847176079734, 'Sexist Recall': 0.632996632996633, 'Sexist F1-Score': 0.628
7625418060201, 'Weighted Precision': 0.7964514246011318, 'Weighted Recall': 0.795580
1104972375, 'Weighted F1-Score': 0.7960040433274567}
SVC: {'Non-Sexist Precision': 0.8362282878411911, 'Non-Sexist Recall': 0.85424588086
18505, 'Non-Sexist F1-Score': 0.845141065830721, 'Sexist Precision': 0.5892857142857
143, 'Sexist Recall': 0.5555555555555556, 'Sexist F1-Score': 0.5719237435008666, 'We
ighted Precision': 0.768694269106406, 'Weighted Recall': 0.7725598526703499, 'Weight
ed F1-Score': 0.7704214113813962}
XGBoost: {'Non-Sexist Precision': 0.8254504504504504, 'Non-Sexist Recall': 0.9290240
811153359, 'Non-Sexist F1-Score': 0.874180083482409, 'Sexist Precision': 0.717171717
1717171, 'Sexist Recall': 0.4781144781144781, 'Sexist F1-Score': 0.5737373737373737,
'Weighted Precision': 0.795838310686377, 'Weighted Recall': 0.8057090239410681, 'Wei
ghted F1-Score': 0.7920148120327999}
```

	Sexist			Non-sexist			Weighted-Average		
	Precision	Recall	F1-score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
[TFIDF] + [Logistic Regression]	0.575931232	0.676767677	0.622291022	0.869742198	0.812420786	0.840104849	0.78939058	0.775322284	0.780536979
[TFIDF] + [svc]	0.632411067	0.538720539	0.581818182	0.835534214	0.882129278	0.858199753	0.779983961	0.788213628	0.782614738
[TFIDF] + [XGBoost]	0.720379147	0.511784512	0.598425197	0.834285714	0.9252218	0.877403846	0.803134471	0.812154696	0.80110858
[Count] + [Logistic Regression]	0.624584718	0.632996633	0.628762542	0.861146497	0.856780735	0.858958069	0.796451425	0.79558011	0.796004043
[Count] + [SVC]	0.589285714	0.555555556	0.571923744	0.836228288	0.854245881	0.845141066	0.768694269	0.772559853	0.770421411
[Count] + [XBoost]	0.717171717	0.478114478	0.573737374	0.82545045	0.929024081	0.874180083	0.795838311	0.805709024	0.792014812

Summary

1. **What preprocessing steps do you follow?**

    - Remove Noise: Links, HTML tags, punctuation, and numbers are removed to clean the text.
    - Case Normalization: All text is converted to lowercase to ensure consistency.
    - Remove Mentions and Hashtags: Social media handles and the hashtag symbol are removed, though the text of the hashtag is retained.
    - Tokenization: Text is split into tokens (words) using a tokenizer optimized for social media text.

- Remove Stopwords: Common words that add little value in text analysis (like "and", "the", etc.) are removed.
- Lemmatization: Words are reduced to their base or dictionary form (lemma).

2. **How do you select the features from the inputs?**

   - TF-IDF Vectorization: This method weighs the text tokens based on their frequency across documents, but adjusted by their rarity across all documents, helping highlight more meaningful terms.
   - Count Vectorization: This simpler method counts the frequency of each word in the documents, representing texts as vectors of term counts.

3. **Which model do you use and what is the structure of your model?**

   - Logistic Regression: A linear model for binary classification tasks.
   - SVC (Support Vector Classifier): A classifier that uses kernel methods to project data into higher dimensions where a hyperplane can be used to separate classes.
   - XGBoost: An implementation of gradient boosted decision trees designed for speed and performance.

4. **How do you train your model?**

   - Each model is trained on the feature sets created by TF-IDF and count vectorization methods. The training process involves fitting the model on the training data and then validating it on a separate test set.

5. **What is the performance of your best model?**

   - The best performance was achieved using the XGBoost model with TF-IDF vectorization:
     - Weighted F1-Score: 0.8011
     - Weighted Precision: 0.8031
     - Weighted Recall: 0.8122
     - These metrics indicate a good balance between precision and recall, particularly in identifying both classes effectively.

6. **What other models or feature engineering methods would you like to implement in the future?**

   - Using a deep learning model like BERT would be very interesting to implement, as it would likely have a better contextual understanding.
   - For feature engineering, we could implement more advanced NLP techniques such as POS tagging or sentiment analysis to get improved feature sets.