Development of gender classification using transformers

Saul Ramirez

Introduction

One of the main applications of Data Science is in marketing. The better you can understand your audience's demographic, the better you can sell to them. However, most users on the internet have become more skeptical about knowingly giving their information away, and typically avoid filling out their profile information. User gender information, or lack thereof, may produce heteroskedasticity further down our pipeline. As a result, this data should be imputed can be used in any recommendation system. For this project, I analyze a small dataset from Kaggle, consisting of 20,050 Twitter profiles to determine if gender classification is possible.

This project was adopted from the following tutorial:
https://www.analyticsvidhya.com/blog/2021/08/twitter-based-gender-classification-a-machine-learning-project/

Methods

As part of my implementation, I examine profile's handle, description, along with a sample tweet to predict the user's gender. I approach this problem is by using Naive Bayes classification, and XG Boost to get an initial estimate of the difficulty of the problem. Then I use a classification transformer and compare the accuracies at the end.

Although the topic of gender is very complicated and it is known that there are more than two genders, I will assume there are only two genders for simplicity. As there are only two classes, the statistical baseline of guessing correctly due to random chance is 50%, this is our accuracy baseline. The Kaggle dataset was a scraped dataset, out of the 20,500 profiles, only 11,194 instances belonged to Male and Female users; the rest of the profiles belonged to brands, or the gender was unknown. I only look at profiles from males and females and drop any instances with missing data. The text input is a combination of the profile handle, description, and text with commas in between. I used regex to clean punctuation and remove punctuation characters.

Table 1: Breakdown of data set classes

Female	5725
Male	5469
Brand	4328
Unknown	702

I noticed that many of the most common words were the same between the two genders and determined that this would be difficult. Words such as "Love", "Life", and "S" for sarcasm were a few of the examples.

Figure 1: Common words for woman (left). Common words for men (right).

The data was split into a 70-20-10, training-validation-test split. The validation and test spectra are relatively similar splits.

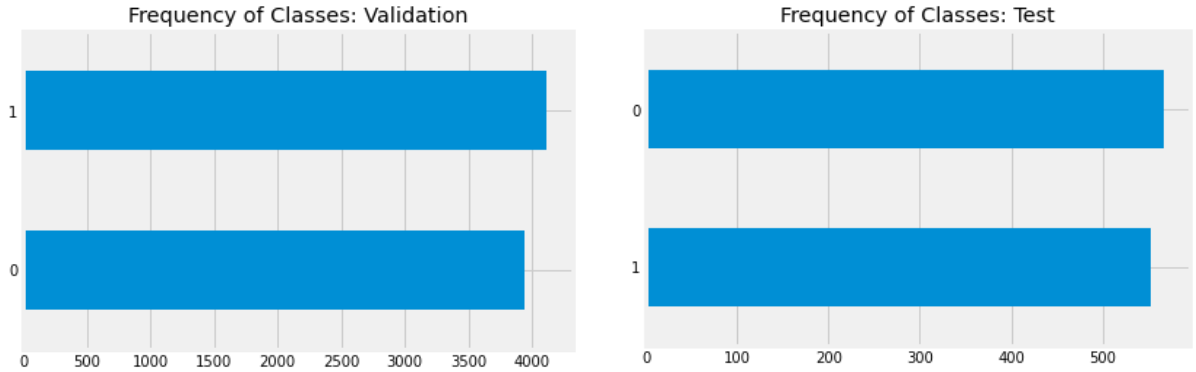


Figure 2: Splits for the validation and test data sets where 1 represents woman and 0 represents men.

Results

The first approach was using the naïve bayes and XG Boost. This method didn't work as well because I didn't remove stop words, and I only used a simple tokenizer instead of GLoVE. The results are shown in Table 2, even with XG Boost, the results are only slightly better than random chance. This gives us a good understanding that this could be a relatively challenging task. However, since the objective of this assignment is to apply transformers, I didn't focus too much on improving this approach.

Table 2: Splits for the validation and test data sets where 1 represents woman and 0 represents

Model	Accuracy	F1
Naïve Bayes Classifier	50.89%	0.42
XG Boost	54.02%	0.51

Next, I fine-tuned the BERT transformer for classification, without changing the dataset at all, I refer to this as the "Vanilla BERT". The results were significantly better than the simple machine learning approaches.

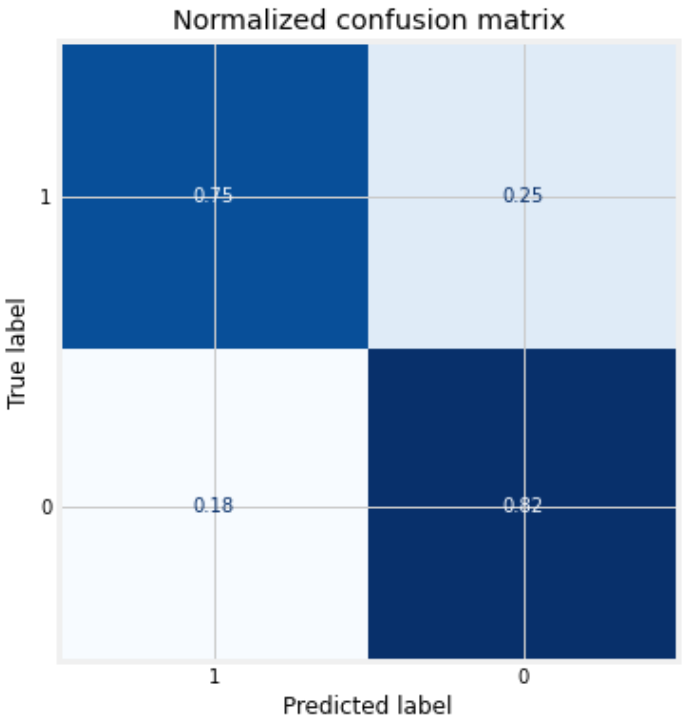


Figure 3: Fine-tuning BERT for gender classification.

I was interested in understanding if the information required to predict gender was mostly located in the profile description. Without using the handle or a sample tweet I retrained the BERT Transformer, as expected the results were not as good as the previous model but surprisingly good for having such low context. This approach had a F1 score of 0.69, which is still significantly better than the machine learning models.

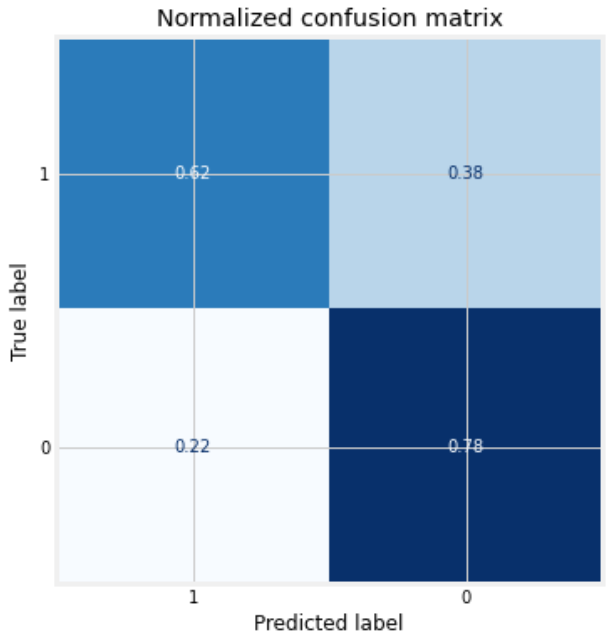


Figure 4: Splits for the validation and test data sets where 1 represents woman and 0 represents men.

Finally, I augmented the training data and fit a final model. To augment the data, I went through each of the text instances (handle, description, and tweet) and generated a number between 0 and 3 to determine if the data should be augmented with synonym replacement, random insert, random swap, random delete or backtranslation. Backtranslation was significantly slow, so I adjusted it to make it not occur as often. Once the data is augmented, the data is combined with the training set only as to keep the same validation and test sets.

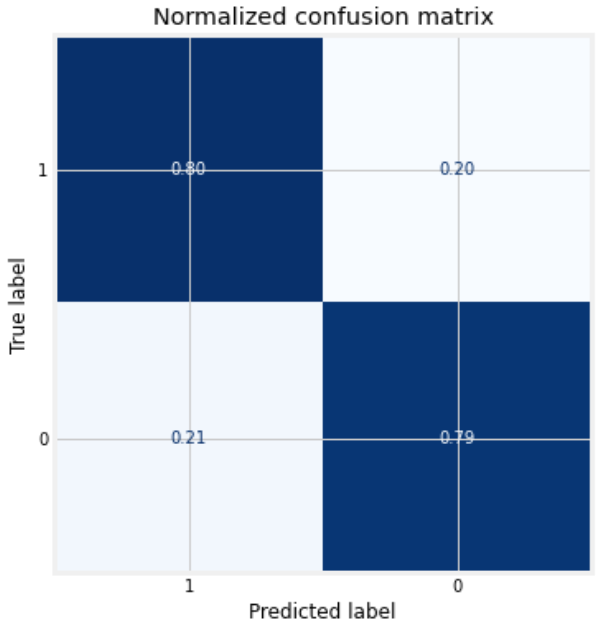


Figure 5: Splits for the validation and test data sets where 1 represents woman and 0 represents men.

The results for the three transformer models are shown in Table 3. As we see the Vanilla BERT does the best. Before developing this approach, the best F1 score was 0.69 for the Vanilla BERT, and didn't receive the large boost until the twitter handle was added to the data. The results for the transformers are shown in Table 3.

Table 3: Splits for the validation and test data sets where 1 represents woman and 0 represents

Model	Accuracy	F1
Vanilla BERT	78.80%	0.79
BERT Description Only	68.73%	0.69
BERT Augmentation	78.56%	0.79

Discussion

To test how well this approach worked I build an API for the BERT Augmented Model. The BERT Augmented Model was selected since it had similar accuracy to the Vanilla BERT but had seen more training examples. I manually derived input for three viral figures on Twitter: Andrew Tate, the epitome of toxic masculinity, Alexandria Ocasio-Cortez, a United States Representative, and Elon Musk. The input was made in the same format as the training data: handle, profile description, sample tweet. The model was able to classify all three examples correctly, but what was interesting was that the model had such high confidence in its prediction given it had seen such few examples.

Table 4: Influential Twitter Users

User	Input	Label	Prediction	P
Andrew Tate	Cobratate, Light-Heavyweight Kickboxing World Champion. Escape the Matrix, Mastery is a funny thing. It's almost as if, on a long enough time, losing simply isn't an option. Such is the way of Wudan	Male	Male	0.995
Alexandria Ocasio-Cortez	AOC, US Representative, NY-14 (BX & Queens). In a modern, moral, & wealthy society, no American should be too poor to live. 💯% People-Funded, no lobbyist💰. She/her.,I see people are rushing out to fill up their cars for this hurricane at the gas station This wouldn't be an issue if they had electric cars.If the power is out for a week how are they going to get gas? We need to start planning ahead and moving forward	Female	Female	0.809
Elon Musk	elonmusk, , twitter deal temporarily on hold pending details supporting calculation that spam/fake accounts	Male	Male	0.996

It was interesting that with AOC, even though her pronouns, “She/her” were present in her tweet, the model had the lowest confidence of the three examples. Perhaps the model was looking heavily at the name of the user. Below is an example of a sample profile with variations of the name “Dan” adjusted to make the gender of the user ambiguous. We see that Dan and Danny, typically Male spellings were classified as Male with extremely high confidence. While Dani and Dany, Female spellings, were classified as Female with extremely high confidence as well. In these situations, it seemed like the description or tweet didn’t matter as much as had been previously shown in the results.

Table 5: Gender Bias of name spellings

Text	Prediction	P
Dani, Live, Laugh, Love, Merry Christmas Everyone.	Female	0.996
Dan, Live, Laugh, Love, Merry Christmas Everyone.	Male	0.986
Dany, Live, Laugh, Love, Merry Christmas Everyone.	Female	0.996
Danny, Live, Laugh, Love, Merry Christmas Everyone.	Male	0.889

Next, I tried making a generic lesbian profile to see if the change in pronouns would confuse the model, but it did not. However, by changing the handle name from BigGay to Big_Gay was all it took to get the model to change it’s mind from Female with 0.994 confidence to Male with 0.799 confidence- despite the rest of the text staying the same and being indicative of the correct classification.

Table 6: LGBTQ Example

Text	Prediction	P
BigGay, Cute, sweet and even funny, Staring at her and thinking, How did a girl like her end up with a girl like me.	Female	0.994
Big_Gay, Cute, sweet and even funny, Staring at her and thinking, How did a girl like her end up with a girl like me.	Male	0.799

This method was applied to my followers on Instagram as well with a similar format: handle, bio, image caption and it was able to correctly classify all 5 users. This shows that it's helpful across platforms.

Conclusion

The power of transformers is displayed with this task, as with very little data and feature engineering we were able to obtain almost 80% accuracy. It is surprising is that there is a distinction between the way that men and women present themselves online, and even more surprising is that these patterns can be distinguished using AI models. At first, I was concerned about the ethics of this problem and my approach since it seemed like I'm perpetuating gender bias to classify profiles online. The fact that I couldn't get higher F1-scores may shed light on the fact that gender may not be cleanly separable. Since both genders speak the same language, therefore the word distribution per class is probably very mixed. However, this is very useful in Marketing, as if we were talking about a identifying people who would be interested in female leggings, it would be inefficient to market to people who identify as Men.

Project:

```
!pip install xgboost
!pip install lightgbm
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('fivethirtyeight')
import warnings
warnings.filterwarnings('ignore')
import nltk
import re
from nltk.stem import PorterStemmer # for stemming
from nltk.stem import WordNetLemmatizer # for lemmatization
from nltk.corpus import stopwords
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: xgboost in /usr/local/lib/python3.8/dist-packages (0.90)
Requirement already satisfied: scipy in /usr/local/lib/python3.8/dist-packages (from xgboost) (1.7.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (from xgboost) (1.21.6)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: lightgbm in /usr/local/lib/python3.8/dist-packages (2.2.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (from lightgbm) (1.21.6)
Requirement already satisfied: scipy in /usr/local/lib/python3.8/dist-packages (from lightgbm) (1.7.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.8/dist-packages (from lightgbm) (1.0.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from scikit-learn->lightgbm) (3.1.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.8/dist-packages (from scikit-learn->lightgbm) (1.2.0)
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
```

```
%matplotlib inline
```

```
data = pd.read_csv('/content/gender-classifier.csv', encoding = 'latin1')
data.head()
```

	_unit_id	_golden	_unit_state	_trusted_judgments	_last_judgment_at	gender
0	815719226	False	finalized	3	10/26/15 23:24	male
1	815719227	False	finalized	3	10/26/15 23:30	male

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20050 entries, 0 to 20049
Data columns (total 26 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   _unit_id               20050 non-null  int64  
 1   _golden                20050 non-null  bool   
 2   _unit_state            20050 non-null  object 
 3   _trusted_judgments     20050 non-null  int64  
 4   _last_judgment_at      20000 non-null  object 
 5   gender                 19953 non-null  object 
 6   gender:confidence      20024 non-null  float64
 7   profile_yn             20050 non-null  object 
 8   profile_yn:confidence  20050 non-null  float64
 9   created                20050 non-null  object 
 10  description            16306 non-null  object 
 11  fav_number             20050 non-null  int64  
 12  gender_gold            50 non-null     object 
 13  link_color             20050 non-null  object 
 14  name                   20050 non-null  object 
 15  profile_yn_gold        50 non-null     object 
 16  profileimage           20050 non-null  object 
 17  retweet_count          20050 non-null  int64  
 18  sidebar_color          20050 non-null  object 
 19  text                   20050 non-null  object 
 20  tweet_coord            159 non-null    object 
 21  tweet_count            20050 non-null  int64  
 22  tweet_created          20050 non-null  object 
 23  tweet_id               20050 non-null  float64
 24  tweet_location         12566 non-null  object 
 25  user_timezone          12252 non-null  object 
dtypes: bool(1), float64(3), int64(5), object(17)
memory usage: 3.8+ MB
```

```
# Drop unique attribute columns and redundant
df = data[['gender', 'description', 'text', "name"]]
df.head()
```

	gender	description	text	name
0	male	i sing my own rhythm.	Robbie E Responds To Critics After Win Against...	sheezy0
1	male	I'm the author of novels filled with family dr...	ÛÏIt felt like they were my friends and I was...	DavdBurnett
2	male	louis whining and squealing and all	i absolutely adore when louis starts the songs...	lwtprettylaugh
3	male	Mobile guy. 49ers, Shazam, Google, Klein... Re	Hi @JordanSpieth - Looking at the ... do you	douggarland

```
# Check for Null Values
print(df.isna().sum())
df.dropna(axis=0, inplace=True)
```

```
gender           97
description    3744
text              0
name              0
dtype: int64
```

```
# Explore gender counts and strata
df['gender'].value_counts()
```

```
female     5725
male       5469
```

```
brand      4328
unknown     702
Name: gender, dtype: int64
```

```
# Parse only Male and Female
df = df[(df['gender'] == "male") | (df['gender'] == "female")]
df.head()
```

	gender	description	text	name
0	male	i sing my own rhythm.	Robbie E Responds To Critics After Win Against...	sheezy0
1	male	I'm the author of novels filled with family dr...	ÛÏIt felt like they were my friends and I was...	DavdBurnett
2	male	louis whining and squealing and all	i absolutely adore when louis starts the songs...	lwtprettylaugh
3	male	Mobile guy. 49ers, Shazam, Google, Kleiner Pe...	Hi @JordanSpieth - Looking at the url - do you...	douggarland

```
df['gender'].value_counts()
```

```
female    5725
male      5469
Name: gender, dtype: int64
```

```
print("Number of instances: ", len(df))
```

```
Number of instances:  11194
```

```
# Encoding Gender Labels
label_map = {"female":1, "male":0}
df["label"] = df["gender"].map(label_map)
df = df.drop(["gender"], axis=1)
df.head()
```

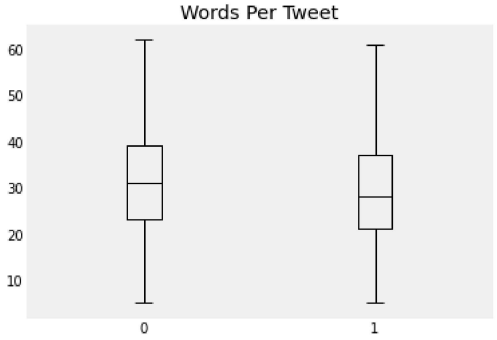
	description	text	name	label
0	i sing my own rhythm.	Robbie E Responds To Critics After Win Against...	sheezy0	0
1	I'm the author of novels filled with family dr...	ÛÏIt felt like they were my friends and I was...	DavdBurnett	0
2	louis whining and squealing and all	i absolutely adore when louis starts the songs...	lwtprettylaugh	0
3	Mobile guy. 49ers, Shazam, Google, Kleiner Pe...	Hi @JordanSpieth - Looking at the url - do you...	douggarland	0

```
#df["text"] = df["description"] + ", " + df["text"]
#df = df.drop(["description"], axis=1)
df["text"] = df["name"] + ", " + df["description"] + ", " + df["text"]
df = df.drop(["description", "name"], axis=1)
df.head()
```

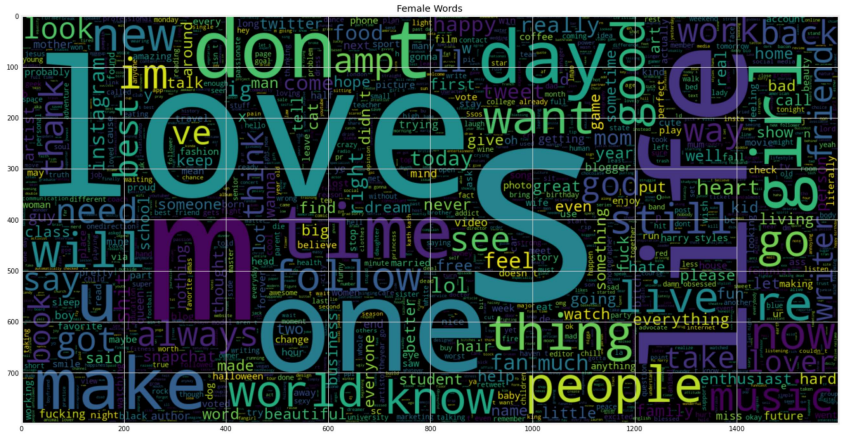
	text	label
0	sheezy0, i sing my own rhythm., Robbie E Respo...	0
1	DavdBurnett, I'm the author of novels filled w...	0
2	lwtprettylaugh, louis whining and squealing an...	0
3	douggarland, Mobile guy. 49ers, Shazam, Googl...	0
4	WilfordGemma, Ricky Wilson The Best FRONTMAN/K...	1

```
# Data Preprocessing
text_cleaning_re = "@\S+|https?:\S+|http?:\S|[^A-Za-z0-9]+"

def preprocessing(regex, text):
  text = re.sub(regex, ' ', str(text).lower()).strip()
  return text

df.text = df.text.apply(lambda x: preprocessing(text_cleaning_re, x))
```

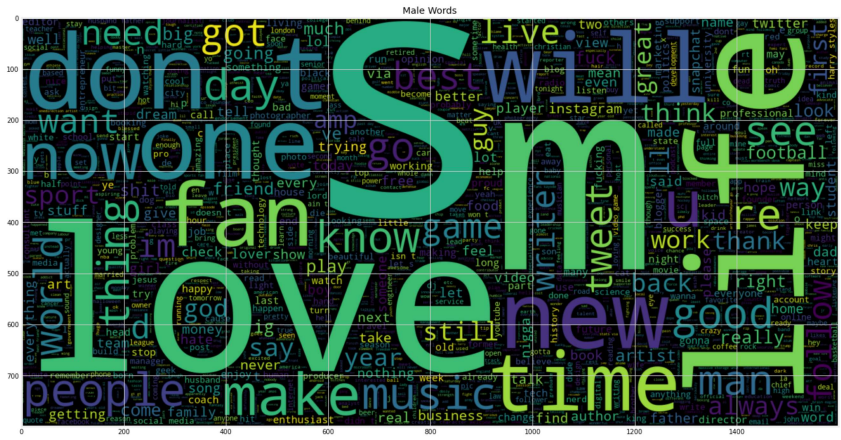
```
df.loc[:, "Words Per Tweet"] = df["text"].str.split().str.len()
df.boxplot("Words Per Tweet", by="label", grid=False, showfliers=False,
           color="black")
plt.suptitle("")
plt.xlabel("")
plt.show()
df = df.drop(["Words Per Tweet"], axis=1)
```




```
from wordcloud import WordCloud

plt.figure(figsize = (20,20))
wc = WordCloud(max_words = 2000 , width = 1600 , height = 800).generate(" ".join(df[df.label == 1].text))
plt.imshow(wc , interpolation = 'bilinear')
plt.title("Female Words")
plt.show()
```




```
plt.figure(figsize = (20,20))
wc = WordCloud(max_words = 2000 , width = 1600 , height = 800).generate(" ".join(df[df.label == 0].text))
plt.imshow(wc , interpolation = 'bilinear')
plt.title("Male Words")
plt.show()
```




```
df_xg = df.copy()
df_xg = df_xg[["label", "text"]]
```

```
train_data, test_data = train_test_split(df_xg, test_size = 0.1, random_state=7)
train_data, val_data = train_test_split(train_data, test_size=0.20, random_state=7)

print("Train Data size:", len(train_data))
print("Val Data size", len(val_data))
print("Test Data size", len(test_data))
```

```
Train Data size: 8059
Val Data size 2015
Test Data size 1120
```

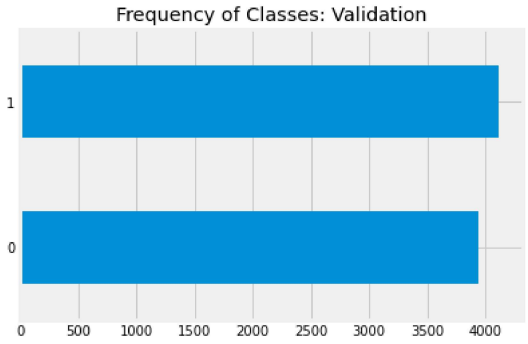
```
# Print Sample
train_data.head(10)
```

	label	text
10822	1	bestdateever relationships couples dating can ...
24	0	jhurkett bsc economics graduate coys james bon...
4066	1	tamrynseale 5sos magcon 1d the vamps the tide ...
6465	0	ity17 god family football if i aint the best j...
11470	0	poeboy412 you came here for a reason just foll...
18634	0	warriorbob9 the 9 is silent my most and last f...
3852	0	alexclegg93 northumbria university 22 and to m...
16737	1	tayedris the world isn t as cruel as you take ...
7444	0	zelakto broadcaster on twitch linux server adm...
14575	1	goddardtara urban studies phd student research...

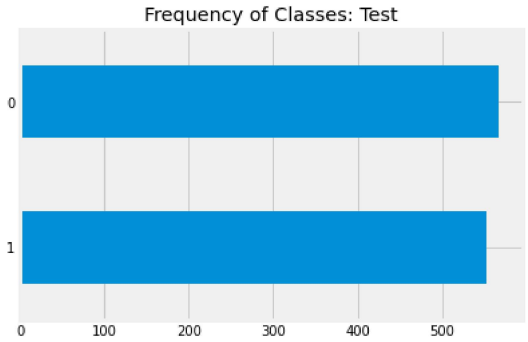
```
# Split the data into features and labels
y_train = train_data["label"]
y_val = val_data["label"]
y_test = test_data["label"]
```

```
# Features
x_train = train_data.drop(["label"], axis=1)
x_val = val_data.drop(["label"], axis=1)
x_test = test_data.drop(["label"], axis=1)
```

```
y_train.value_counts(ascending=True).plot.barh()

plt.title("Frequency of Classes: Validation")
plt.show()
```




```
y_test.value_counts(ascending=True).plot.barh()
plt.title("Frequency of Classes: Test")
plt.show()
```



Machine Learning Approaches

[ ] ↳ *12 cells hidden*

Transformer Model Approach with Bert

[ ] ↳ *17 cells hidden*

Attempting to classify using only the profile description

[ ] ↳ *9 cells hidden*

Augmenting data

[ ] ↳ *14 cells hidden*