Students Complaints prediction using NLP

By Salonee Jadhav

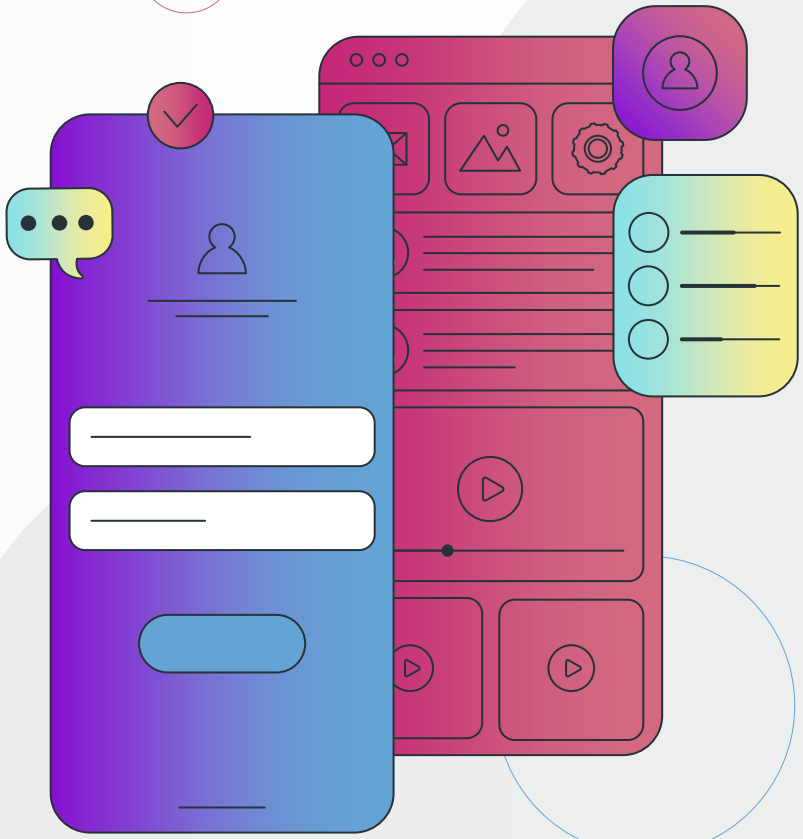


TABLE OF CONTENTS

01

INTRODUCTION

01 INTRODUCTION TO THE DATASET

The dataset is a comprehensive collection of reports and complaints submitted by students in a university setting. From academic grievances to campus safety concerns, this dataset offers a rich trove of insights into the student experience, providing valuable feedback for university administrators and educators. With its diverse range of feedback it offers offers a unique opportunity to gain a better understanding of the needs and concerns of students, and to develop data-driven solutions to enhance the university experience for all.

02

PREPROCESSING & EDA

Predicting department/Genre to which the Reports/Complaints belongs to

Data loading

```
[ ] import pandas as pd
    df= pd.read_csv('/content/Datasetprojpowerbi.csv')
    df.head()
```

	Genre	Reports	Age	Gpa	Year	Count	Gender	Nationality
0	Academic Support and Resources	The limited access to research databases and m...	27	2.18	2	1	M	Egypt
1	Academic Support and Resources	I'm having trouble finding the course material...	23	3.11	2	1	F	Egypt
2	Academic Support and Resources	It's frustrating to have limited access to res...	20	3.68	2	1	F	Egypt
3	Academic Support and Resources	I'm really struggling in one of my classes but...	20	1.30	2	1	F	Egypt
4	Academic Support and Resources	I am really struggling with understanding the...	26	2.50	2	1	F	Egypt

```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1005 entries, 0 to 1004
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Genre        1005 non-null   object
 1   Reports      1005 non-null   object
 2   Age          1005 non-null   int64
 3   Gpa          1005 non-null   float64
 4   Year         1005 non-null   int64
 5   Count        1005 non-null   int64
 6   Gender       1005 non-null   object
 7   Nationality  1005 non-null   object
dtypes: float64(1), int64(3), object(4)
memory usage: 62.9+ KB
```

```
[ ] print(df['Genre'].value_counts())
```

```
Genre
Academic Support and Resources       236
Food and Cantines                    138
Financial Support                     91
Online learning                       90
Career opportunities                  89
International student experiences     86
Athletics and sports                  85
Housing and Transportation            64
Health and Well-being Support         53
Activities and Travelling             40
Student Affairs                       33
Name: count, dtype: int64
```

```
[ ] df.info()
```

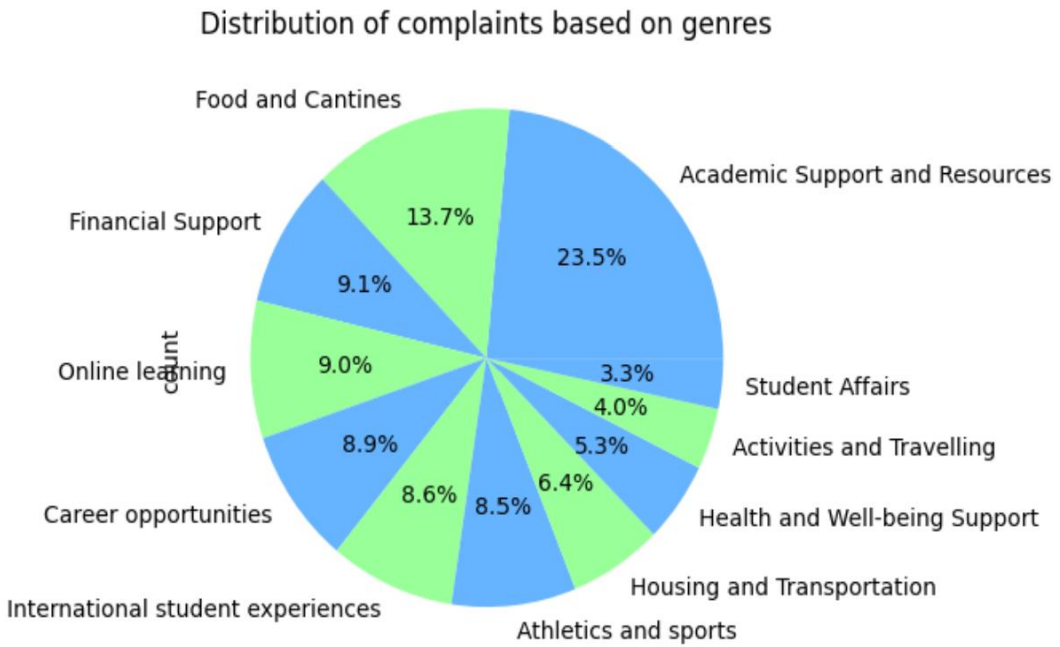
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1005 entries, 0 to 1004
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Genre        1005 non-null   object
 1   Reports      1005 non-null   object
 2   Age          1005 non-null   int64
 3   Gpa          1005 non-null   float64
 4   Year         1005 non-null   int64
 5   Count        1005 non-null   int64
 6   Gender       1005 non-null   object
 7   Nationality  1005 non-null   object
dtypes: float64(1), int64(3), object(4)
memory usage: 62.9+ KB
```

```
[ ] print(df['Genre'].value_counts())
```

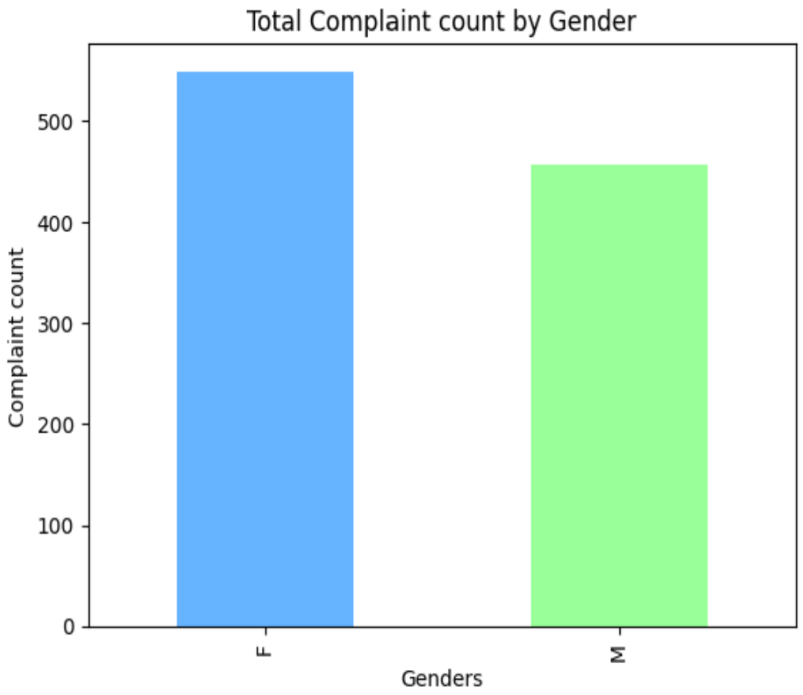
```
Genre
Academic Support and Resources       236
Food and Cantines                    138
Financial Support                     91
Online learning                       90
Career opportunities                  89
International student experiences     86
Athletics and sports                  85
Housing and Transportation            64
Health and Well-being Support         53
Activities and Travelling             40
Student Affairs                       33
Name: count, dtype: int64
```

Exploratory Data Analysis

```
[ ] # EDA 1: Distribution of Classes
    import matplotlib.pyplot as plt
    class_distribution = df['Genre'].value_counts()
    class_distribution.plot(kind='pie', autopct='%1.1f%%', colors=['#66b3ff','#99ff99'])
    plt.title('Distribution of complaints based on genres')
    plt.show()
```

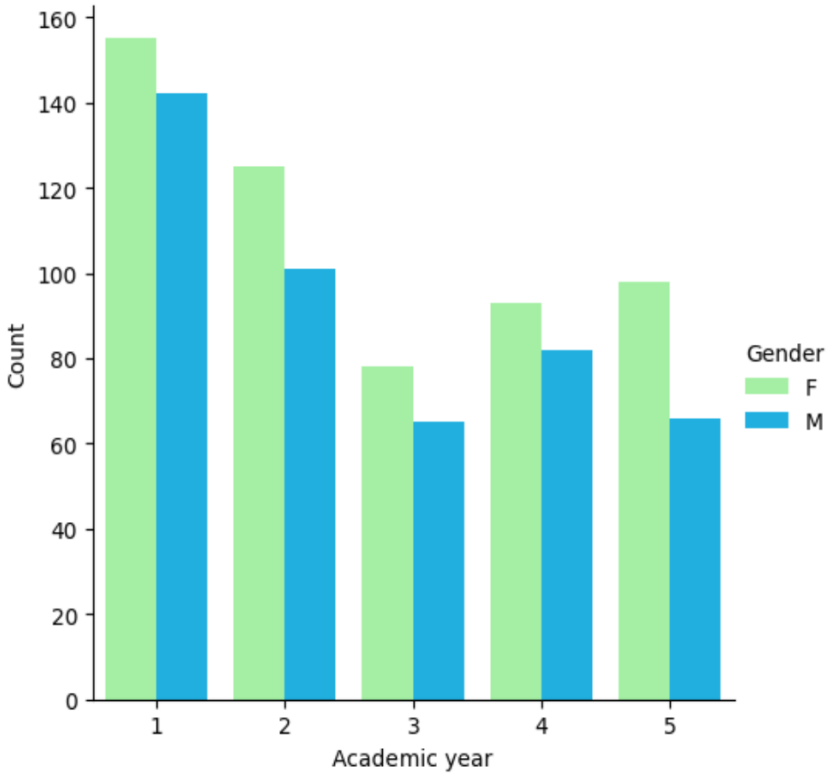


```
[ ] gender_com= df.groupby('Gender')['Count'].sum()
    color=['#66b3ff','#99ff99']
    gender_com.plot(kind='bar',x='Gender',y='Count',color=color)
    # set the title
    plt.title('Total Complaint count by Gender')
    plt.xlabel("Genders")
    plt.ylabel("Complaint count")
    # show the plot
    plt.show()
```



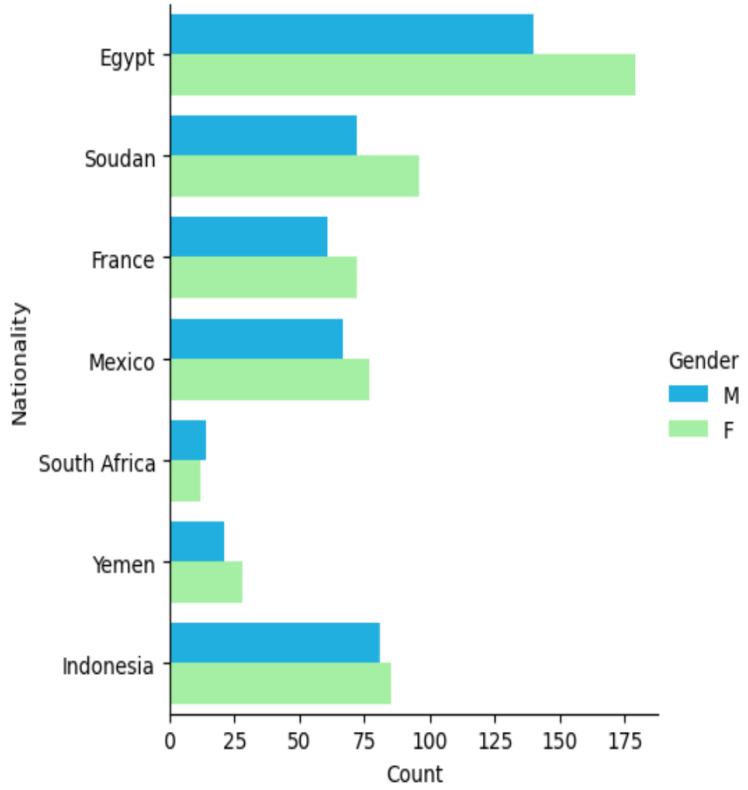

```
import seaborn as sns
gen ={"M": "deepskyblue", "F": "palegreen"}
sns.catplot(
    data=df, x="Year", hue="Gender", kind="count",
    palette=gen).set_axis_labels("Academic year", "Count")
```

```
<seaborn.axisgrid.FacetGrid at 0x7c1970c36440>
```

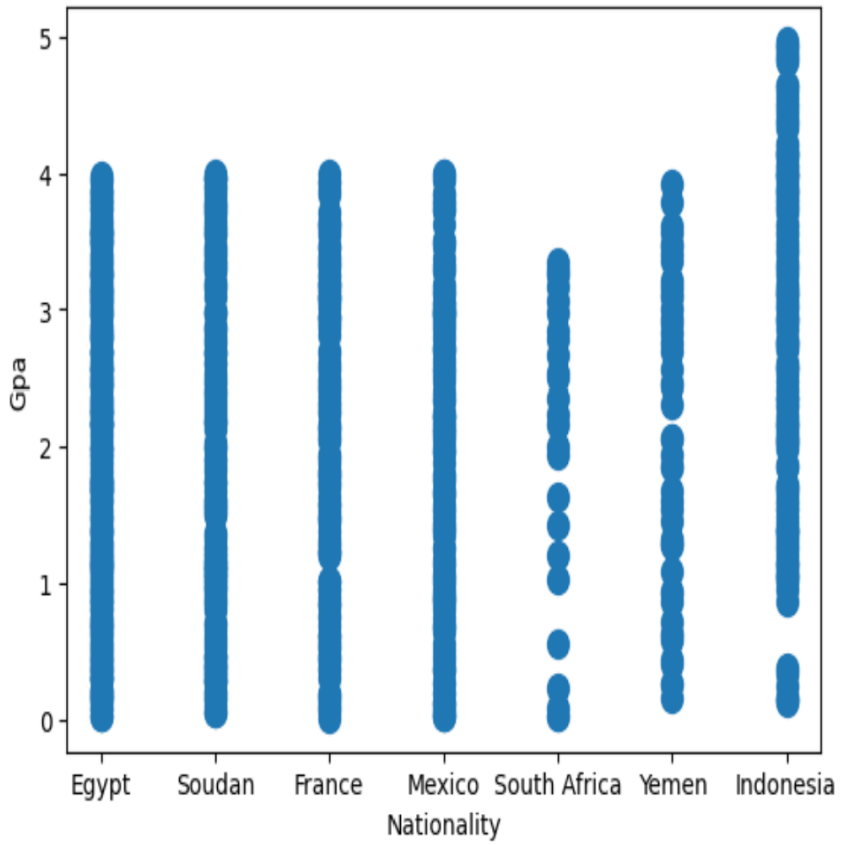


```
gen ={"M": "deepskyblue", "F": "palegreen"}
sns.catplot(data=df, y="Nationality",hue='Gender',kind="count",palette=gen).
set_axis_labels("Count","Nationality")
```

```
<seaborn.axisgrid.FacetGrid at 0x7c1960373190>
```

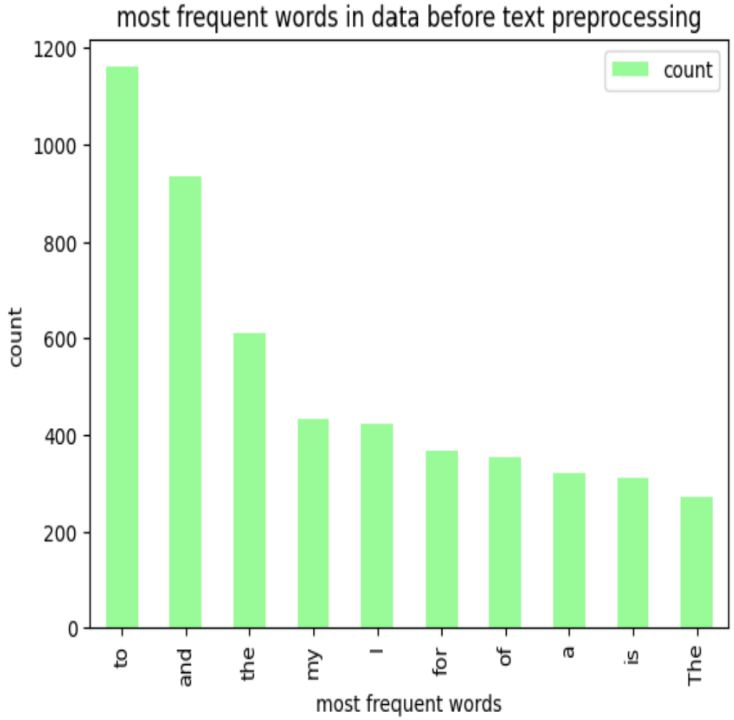



```
df.plot.scatter(x = 'Nationality', y = 'Gpa', s = 100);
```



```
[ ] #The most frequent word used in complaints/ Reports before preprocessing the text data
    df1 = df.Reports.str.split(expand=True).stack().value_counts().head(10)
    df1.plot(kind='bar',color='palegreen')
    plt.title("most frequent words in data before text preprocessing")
    plt.xlabel("most frequent words")
    plt.ylabel("count")
    plt.legend()
```

```
<matplotlib.legend.Legend at 0x7c195f38d900>
```



Text Preprocessing

```
[ ] !pip install emoji
    import emoji
    df['Reports'] = df['Reports'].apply(lambda s: emoji.replace_emoji(s, ''))
    df['Reports']
```

```
Requirement already satisfied: emoji in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: typing-extensions>=4.7.0 in /usr/local/lib/
```

	Reports
0	The limited access to research databases and m...
1	I'm having trouble finding the course material...
2	It's frustrating to have limited access to res...
3	I'm really struggling in one of my classes but...
4	I am really struggling with understanding the...
...	...
1000	26. I have been unable to find food that meets...
1001	27. I have been unable to find food that I can...
1002	28. I have been unable to find food that I enjoy.
1003	29. I have been unable to find food that is he...
1004	30. I have been unable to find food that is co...

1005 rows × 1 columns

```
[ ] df['Reports'] = df['Reports'].replace("\s+", " ", regex=True).str.strip()
```

```
[ ] import string
    string.punctuation
    def remove_punctuations(text):
      punctuationfree= "".join([i for i in text if i not in string.punctuation])
      return punctuationfree
    df['Reports']=df['Reports'].apply(lambda x:remove_punctuations(x))
```

```
[ ] df['Reports']=df['Reports'].apply(lambda x:x.lower())
    df.head()
```

	Genre	Reports	Age
0	Academic Support and Resources	the limited access to research databases and m...	27
1	Academic Support and Resources	im having trouble finding the course materials...	23
2	Academic Support and Resources	its frustrating to have limited access to rese...	20
3	Academic Support and Resources	im really struggling in one of my classes but ...	20
4	Academic Support and Resources	i am really struggling with understanding the ...	26

```
[ ] df['Reports']=df['Reports'].apply(lambda x: x.split(' '))
```

```
[ ] import nltk
    nltk.download('stopwords')
    stopwords=nltk.corpus.stopwords.words('english')
    def remove_stopwords(text):
      output=[ i for i in text if i not in stopwords]
      return output
    df['Reports']=df['Reports'].apply(lambda x: remove_stopwords(x))
    df.head()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

	Genre	Reports
0	Academic Support and Resources	[limited, access, research, databases, materia...
1	Academic Support and Resources	[im, trouble, finding, course, materials, need...
2	Academic Support and Resources	[frustrating, limited, access, research, datab...
3	Academic Support and Resources	[im, really, struggling, one, classes, cant, g...
4	Academic Support and Resources	[really, struggling, understanding, instructio...

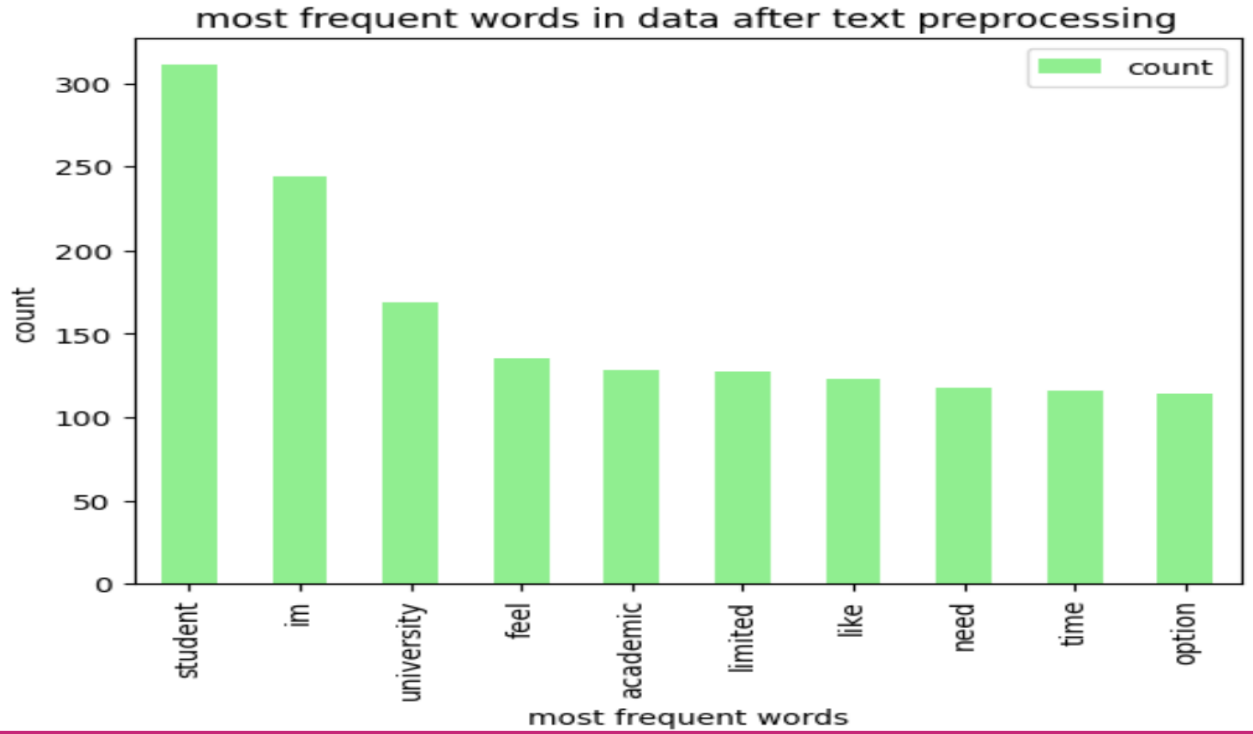
```
[ ] from nltk.stem import WordNetLemmatizer
    nltk.download('wordnet')
    lemma= WordNetLemmatizer()
    #defining the fn for lemmatization
    def lemmatizer(text):
      lemma_text=[lemma.lemmatize(word) for word in text]
      return lemma_text

    df['Reports']= df['Reports'].apply(lambda x: lemmatizer(x))
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
[ ] df['Reports'] = df['Reports'].apply(lambda x: [word for word in x if word.isalpha()])
    df['Reports'] = df['Reports'].apply(lambda x: " ".join(x))
```

```python
#The most frequent word used in complaints/ Reports
df2=df.Reports.str.split(expand=True).stack().value_counts().head(10)
df2.plot(kind='bar',color='lightgreen')
plt.title("most frequent words in data after text preprocessing")
plt.xlabel("most frequent words")
plt.ylabel("count")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7c195ef624a0>
```



Data encoding

```
[ ] from sklearn.preprocessing import LabelEncoder
    obj=df[['Gender', 'Nationality','Genre']]
    for col in obj:
      encoder= LabelEncoder()
      df[col]= encoder.fit_transform(df[col])
```

Transforming text into tabular form

```
[ ] from sklearn.feature_extraction.text import CountVectorizer
    vectorizer= CountVectorizer()
    X= vectorizer.fit_transform(df['Reports'])
    x= pd.DataFrame(X.toarray(), columns= vectorizer.get_feature_names_out(),index= df['Reports'])
    x.head()
```

Reports									
limited									
access									
research									
database									
material									
causing lot	0	1	0	0	1	0	0	0	0
frustration									
among									
student need									
better access									
able succeed									
academically									

03

MODEL TRAINING & SELECTION

Data Splitting

```
[ ] from sklearn.model_selection import train_test_split
    xtrain,xtest,ytrain,ytest= train_test_split(x,y,train_size=0.8,random_state=40)
```

```
[ ] DF= pd.DataFrame(columns=['Model_Name','Training_score','Testing_score'])
```

Model Training and Selection

```
[ ] from sklearn.metrics import f1_score
    def evaluation_model(model, xtrain, ytrain, xtest, ytest):
      model.fit(xtrain, ytrain)
      trainpred = model.predict(xtrain)
      testpred = model.predict(xtest)
      return f1_score(ytrain, trainpred,average='micro'),f1_score(ytest, testpred,average='micro')
```

KNN WITH N NEIGHBORS

```python
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)
row=[]
row.extend(['KNN(3)'])
row.extend(evaluation_model(model, xtrain, ytrain, xtest, ytest))
DF.loc[len(DF.index)] = row
result=evaluation_model(model, xtrain, ytrain, xtest, ytest)
print('score_train:',result[0])
print('score_test:',result[1])
```

```
score_train: 0.8296019900497511
score_test: 0.746268656716418
```

```python
model = KNeighborsClassifier(n_neighbors=5)
row=[]
row.extend(['KNN(5)'])
row.extend(evaluation_model(model, xtrain, ytrain, xtest, ytest))
DF.loc[len(DF.index)] = row
result=evaluation_model(model, xtrain, ytrain, xtest, ytest)
print('score_train:',result[0])
print('score_test:',result[1])
```

```
score_train: 0.7574626865671642
score_test: 0.6865671641791045
```

LOGISTIC REGRESSION

```python
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(class_weight='balanced')
row=[]
row.extend(['LogisticRegression(class_weight=balanced)'])
row.extend(evaluation_model(model, xtrain, ytrain, xtest, ytest))
DF.loc[len(DF.index)] = row
result=evaluation_model(model, xtrain, ytrain, xtest, ytest)
print('score_train:',result[0])
print('score_test:',result[1])
```

```
score_train: 0.9975124378109452
score_test: 0.945273631840796
```

```python
model = LogisticRegression()
row=[]
row.extend(['LogisticRegression'])
row.extend(evaluation_model(model, xtrain, ytrain, xtest, ytest))
DF.loc[len(DF.index)] = row
result=evaluation_model(model, xtrain, ytrain, xtest, ytest)
print('score_train:',result[0])
print('score_test',result[1])
```

```
score_train: 1.0
score_test 0.9402985074626865
```

DECISION TREE

```
[ ] from sklearn.tree import DecisionTreeClassifier
    model = DecisionTreeClassifier(class_weight='balanced')
    row=[]
    row.extend(['DecisionTreeClassifier(class_weight=balanced)'])
    row.extend(evaluation_model(model, xtrain, ytrain, xtest, ytest))
    DF.loc[len(DF.index)] = row
    result=evaluation_model(model, xtrain, ytrain, xtest, ytest)
    print('score_train:',result[0])
    print('score_test:',result[1])
```

```
score_train: 1.0
score_test: 0.8756218905472637
```

```
[ ] model = DecisionTreeClassifier(max_depth=3,criterion='entropy')
    row=[]
    row.extend(['DecisionTreeClassifier(criterion=entropy)'])
    row.extend(evaluation_model(model, xtrain, ytrain, xtest, ytest))
    DF.loc[len(DF.index)] = row
    result=evaluation_model(model, xtrain, ytrain, xtest, ytest)
    print('score_train:',result[0])
    print('score_test:',result[1])
```

```
score_train: 0.43905472636815923
score_test: 0.42288557213930356
```

Random Forest

```
[ ] from sklearn.ensemble import RandomForestClassifier
    model= RandomForestClassifier(n_estimators=90)
    row=[]
    row.extend(['RandomForestClassifier(n_estimators=90)'])
    row.extend(evaluation_model(model, xtrain, ytrain, xtest, ytest))
    DF.loc[len(DF.index)] = row
    result=evaluation_model(model, xtrain, ytrain, xtest, ytest)
    print('score_train:',result[0])
    print('score_test:',result[1])
```

```
score_train: 1.0
score_test: 0.9154228855721394
```

```
[ ] model= RandomForestClassifier(n_estimators=50)
    row=[]
    row.extend(['RandomForestClassifier(n_estimators=50)'])
    row.extend(evaluation_model(model, xtrain, ytrain, xtest, ytest))
    DF.loc[len(DF.index)] = row
    result=evaluation_model(model, xtrain, ytrain, xtest, ytest)
    print('score_train:',result[0])
    print('score_test:',result[1])
```

```
score_train: 1.0
score_test: 0.8955223880597015
```

EXTRA TREES

```
[ ] from sklearn.ensemble import ExtraTreesClassifier
    model= ExtraTreesClassifier(n_estimators=150)
    row=[]
    row.extend(['ExtraTreesClassifier(n_estimators=150)'])
    row.extend(evaluation_model(model, xtrain, ytrain, xtest, ytest))
    DF.loc[len(DF.index)] = row
    result=evaluation_model(model, xtrain, ytrain, xtest, ytest)
    print('score_train:',result[0])
    print('score_test:',result[1])
```

```
score_train: 1.0
score_test: 0.9402985074626865
```

```
[ ] model= ExtraTreesClassifier(n_estimators=50)
    row=[]
    row.extend(['ExtraTreesClassifier(n_estimators=50)'])
    row.extend(evaluation_model(model, xtrain, ytrain, xtest, ytest))
    DF.loc[len(DF.index)] = row
    result=evaluation_model(model, xtrain, ytrain, xtest, ytest)
    print('score_train:',result[0])
    print('score_test:',result[1])
```

```
score_train: 1.0
score_test: 0.945273631840796
```

NAIVE BAYS

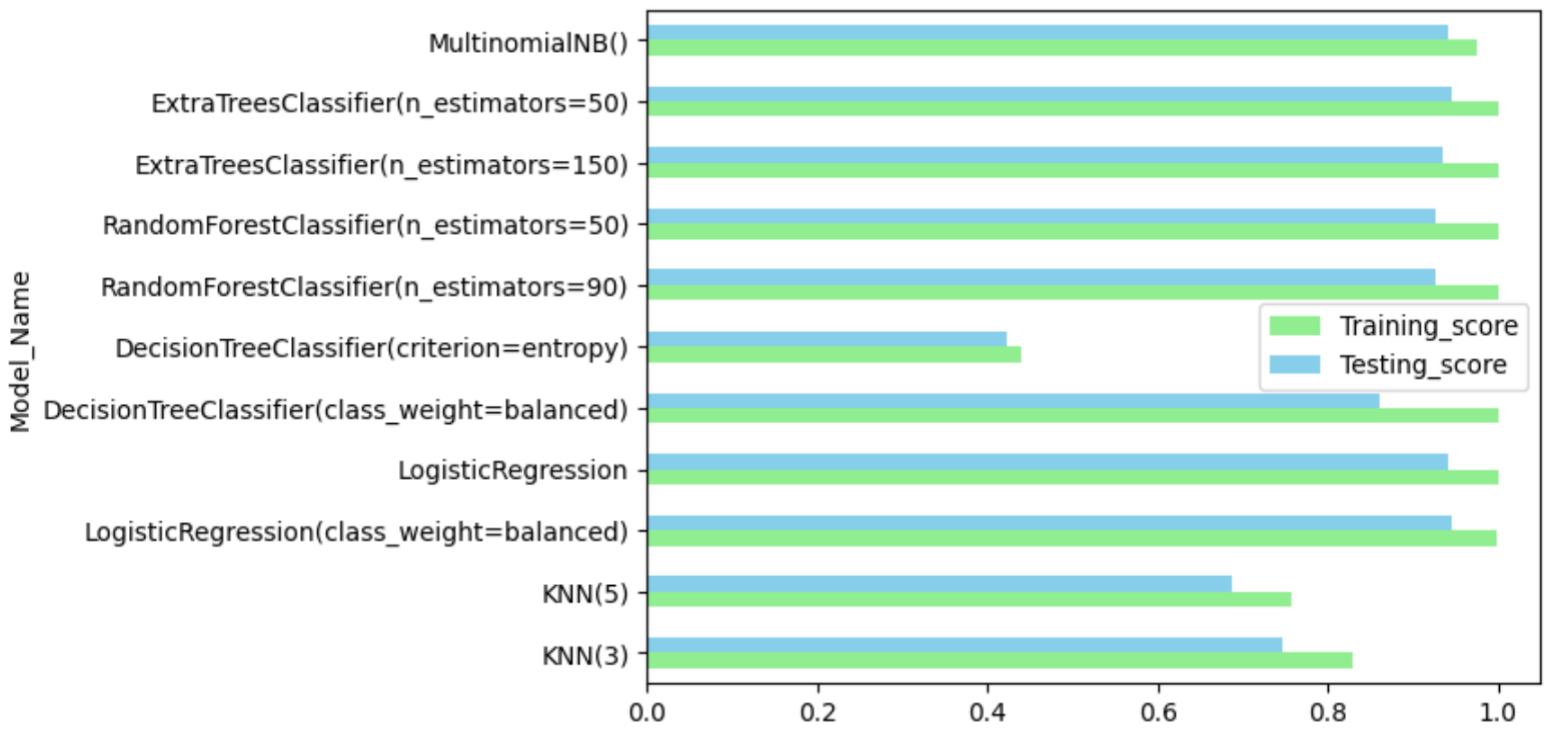
```
[ ] from sklearn.naive_bayes import MultinomialNB
    model= MultinomialNB()
    row=[]
    row.extend(['MultinomialNB()'])
    row.extend(evaluation_model(model, xtrain, ytrain, xtest, ytest))
    DF.loc[len(DF.index)] = row
    result=evaluation_model(model, xtrain, ytrain, xtest, ytest)
    print('score_train:',result[0])
    print('score_test:',result[1])
```

```
score_train: 0.9751243781094527
score_test: 0.9402985074626865
```

DF

	Model_Name	Training_score	Testing_score
0	KNN(3)	0.829602	0.746269
1	KNN(5)	0.757463	0.686567
2	LogisticRegression(class_weight=balanced)	0.997512	0.945274
3	LogisticRegression	1.000000	0.940299
4	DecisionTreeClassifier(class_weight=balanced)	1.000000	0.860697
5	DecisionTreeClassifier(criterion=entropy)	0.439055	0.422886
6	RandomForestClassifier(n_estimators=90)	1.000000	0.925373
7	RandomForestClassifier(n_estimators=50)	1.000000	0.925373
8	ExtraTreesClassifier(n_estimators=150)	1.000000	0.935323
9	ExtraTreesClassifier(n_estimators=50)	1.000000	0.945274
10	MultinomialNB()	0.975124	0.940299

```
[ ]  ax = DF.plot.barh('Model_Name',color={"Training_score": "lightgreen", "Testing_score": "skyblue"})
```



Trainig the data with selected model

```
[ ] model1= ExtraTreesClassifier(n_estimators=50)
    model1.fit(xtrain,ytrain)
```

```
ExtraTreesClassifier
ExtraTreesClassifier(n_estimators=50)
```

```
[ ] testpred1= model1.predict(xtest)
```

```
[ ] trainpred1= model1.predict(xtrain)
```

04

EVALUATION

Evaluation

for testing data

```
[ ] #evaluate the model
    from sklearn.metrics import classification_report
    print("classification report:\n",classification_report(ytest,testpred1))
```

```
classification report:
               precision    recall  f1-score   support

           0       0.86      1.00      0.93        44
           1       1.00      0.67      0.80         6
           2       1.00      1.00      1.00        16
           3       0.91      0.95      0.93        21
           4       1.00      0.88      0.94        25
           5       0.94      1.00      0.97        31
           6       0.83      0.83      0.83         6
           7       1.00      0.85      0.92        13
           8       1.00      0.93      0.96        14
           9       1.00      0.89      0.94        18
          10       1.00      1.00      1.00         7

    accuracy                           0.94       201
   macro avg       0.96      0.91      0.93       201
weighted avg       0.95      0.94      0.94       201
```

for training data

```
print("classification report:\n",classification_report(ytrain,trainpred1))
```

```
classification report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00       192
           1       1.00      1.00      1.00        34
           2       1.00      1.00      1.00        69
           3       1.00      1.00      1.00        68
           4       1.00      1.00      1.00        66
           5       1.00      1.00      1.00       107
           6       1.00      1.00      1.00        47
           7       1.00      1.00      1.00        51
           8       1.00      1.00      1.00        72
           9       1.00      1.00      1.00        72
          10       1.00      1.00      1.00        26

    accuracy                           1.00       804
   macro avg       1.00      1.00      1.00       804
weighted avg       1.00      1.00      1.00       804
```

05

CONCLUSIONS

CONCLUSIONS

- Most complaints are targeted towards Academic source and resources where as least complaints are targeted towards students affairs.
- Out of total complaints made by students , most of the complaints are registered by female students.
- Students from academic year 1, registered highest number of complaints.
- Most of the students present in this data are coming from Egypt country followed Indonesia.
- Students belonging to the country Indonesia have highet GPA score.
- Before preprocessing, the word ‘ to ’ is occurring the most in complaints.
- After preprocessing, the word ‘ Student’ is occurring the most in complaints.
- After model selection process, the best model of the data is extra tress classifier with n=50 with f1 score 94 %.

THANKS!

Happy Analysis!!