AYDIN ADNAN MENDERES UNIVERSITY

ENGINEERING FACULTY
COMPUTER SCIENCE ENGINEERING DEPARTMENT

NER On Medical Text Second Part

CSE431 – Natural Language Processing with Machine Learning 2023/2024

Burak TÜZEL
Talha Alper ASAV

Lecturer:

Asst. Prof. Dr. Fatih SOYGAZİ

Named Entity Recognition on Medical Text Second Part

Installing necessary environment and importing the libraries:

1- Install Jupyter Notebook

```
import spacy
import scispacy
from spacy import displacy
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import string
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.manifold import TSNE
from nltk.tokenize import word_tokenize
from nltk.tokenize import sent_tokenize
from nltk.stem import WordNetLemmatizer
from imblearn.over_sampling import SMOTE
import gc
```

Reading the Medical Text Without NER and Preprocessing:

Here is the medical text link to download: https://www.kaggle.com/datasets/tboyle10/medicaltranscriptions

Firstly we read the medical samples data. Then creating a method to get unique words(vocabulary) and sentence count in a list of text

```
# Reading the csv file
file_path = 'D:/nlp/mtsamples.csv'
df = pd.read_csv(file_path)
```

```
def get_sentence_word_count(text_list):
    sent_count = 0
    word_count = 0
    vocab = {}
    for text in text_list:
        sentences=sent_tokenize(str(text).lower())
        sent_count = sent_count + len(sentences)
        for sentence in sentences:
            words=word_tokenize(sentence)
            for word in words:
                if(word in vocab.keys()):
                    vocab[word] = vocab[word] +1
                else:
                    vocab[word] =1
    word_count = len(vocab.keys())
    return sent_count,word_count
```

Finding the Disease, Drugs and Drugs-Doses Named Entities:

Here we checked the number of samples in each category. Since some categories have less than 50 we will remove them (You can check the full output from the given .html file)

```
df = df[df['transcription'].notna()]
sent_count,word_count= get_sentence_word_count(df['transcription'].tolist())
print("Number of sentences in transcriptions column: "+ str(sent_count))
print("Number of unique words in transcriptions column: "+str(word_count))



data_categories  = df.groupby(df['medical_specialty'])
i = 1
print('==========Original Categories ======================')
for catName,dataCategory in data_categories:
    print('Cat:'+str(i)+' '+catName + ' : '+ str(len(dataCategory)) )
    i = i+1
print('==================================')
```

```
Number of sentences in transcriptions column: 140235
Number of unique words in transcriptions column: 35805
==========Original Categories ======================
Cat:1  Allergy / Immunology : 7
Cat:2  Autopsy : 8
Cat:3  Bariatrics : 18
Cat:4  Cardiovascular / Pulmonary : 371
Cat:5  Chiropractic : 14
Cat:6  Consult - History and Phy. : 516
Cat:7  Cosmetic / Plastic Surgery : 27
Cat:8  Dentistry : 27
Cat:9  Dermatology : 29
Cat:10  Diets and Nutritions : 10
Cat:11  Discharge Summary : 108
Cat:12  ENT - Otolaryngology : 96
```

```
filtered_data_categories = data_categories.filter(lambda x:x.shape[0] > 50)
final_data_categories = filtered_data_categories.groupby(filtered_data_categories['medical_specialty'])
i=1
print('============Reduced Categories ======================')
for catName,dataCategory in final_data_categories:
    print('Cat:'+str(i)+' '+catName + ' : '+ str(len(dataCategory)) )
    i = i+1

print('============ Reduced Categories ======================')
```

```
============Reduced Categories ======================
Cat:1  Cardiovascular / Pulmonary : 371
Cat:2  Consult - History and Phy. : 516
Cat:3  Discharge Summary : 108
Cat:4  ENT - Otolaryngology : 96
Cat:5  Emergency Room Reports : 75
Cat:6  Gastroenterology : 224
```

Preprocessing to Increase the Accuracy:

```
def clean_text(text ):
    text = text.translate(str.maketrans('', '', string.punctuation))
    text1 = ''.join([w for w in text if not w.isdigit()])
    REPLACE_BY_SPACE_RE = re.compile('[/(){}\[\]\|@,;]')
    #BAD_SYMBOLS_RE = re.compile('[^0-9a-z #+_]')

    text2 = text1.lower()
    text2 = REPLACE_BY_SPACE_RE.sub('', text2) # replace REPLACE_BY_SPACE_RE symbols by space in text
    #text2 = BAD_SYMBOLS_RE.sub('', text2)
    return text2

def lemmatize_text(text):
    wordlist=[]
    lemmatizer = WordNetLemmatizer()
    sentences=sent_tokenize(text)

    intial_sentences= sentences[0:1]
    final_sentences = sentences[len(sentences)-2: len(sentences)-1]

    for sentence in intial_sentences:
        words=word_tokenize(sentence)
        for word in words:
            wordlist.append(lemmatizer.lemmatize(word))
    for sentence in final_sentences:
        words=word_tokenize(sentence)
        for word in words:
            wordlist.append(lemmatizer.lemmatize(word))
    return ' '.join(wordlist)
```

Here we grouped the subcategories into their specialties.

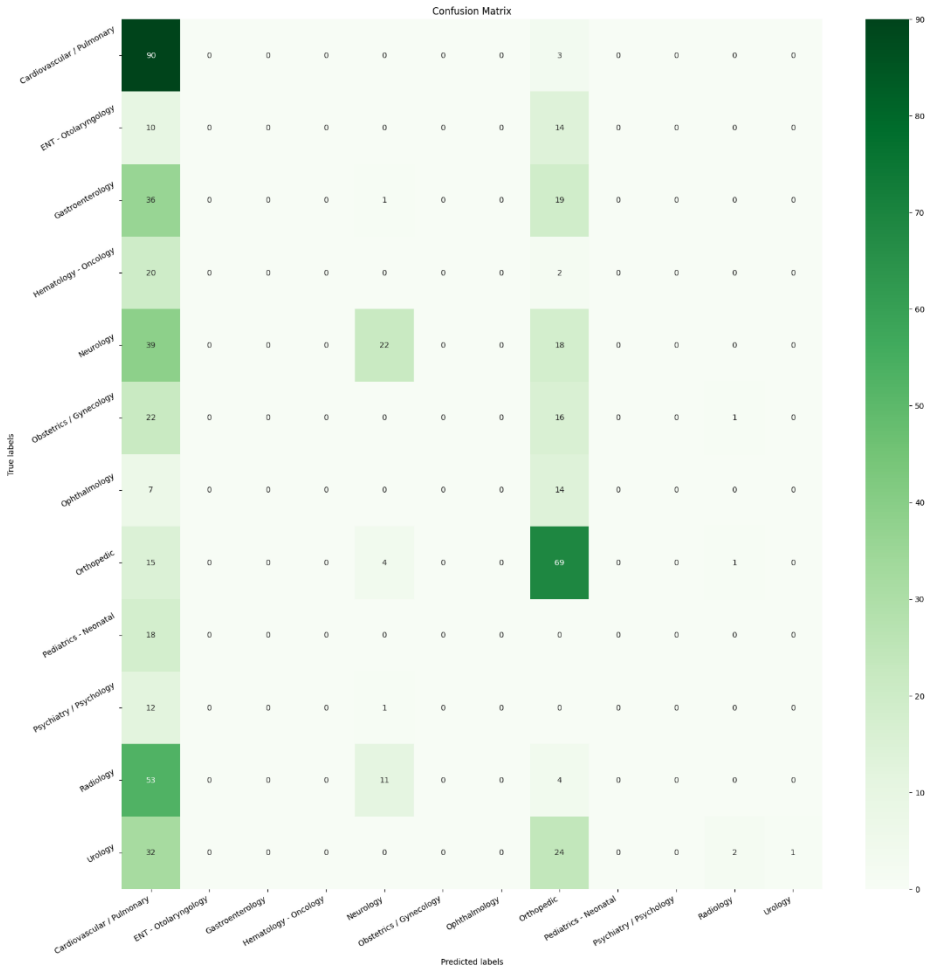
```
filtered_data_categories['medical_specialty'] =filtered_data_categories['medical_specialty'].apply(lambda x:str.strip(x))
mask = filtered_data_categories['medical_specialty'] == 'Surgery'
filtered_data_categories = filtered_data_categories[~mask]
final_data_categories = filtered_data_categories.groupby(filtered_data_categories['medical_specialty'])
mask = filtered_data_categories['medical_specialty'] == 'SOAP / Chart / Progress Notes'
filtered_data_categories = filtered_data_categories[~mask]
mask = filtered_data_categories['medical_specialty'] == 'Office Notes'
filtered_data_categories = filtered_data_categories[~mask]
mask = filtered_data_categories['medical_specialty'] == 'Consult - History and Phy.'
filtered_data_categories = filtered_data_categories[~mask]
mask = filtered_data_categories['medical_specialty'] == 'Emergency Room Reports'
filtered_data_categories = filtered_data_categories[~mask]
mask = filtered_data_categories['medical_specialty'] == 'Discharge Summary'
filtered_data_categories = filtered_data_categories[~mask]

'''
mask = filtered_data_categories['medical_specialty'] == 'Pediatrics - Neonatal'
filtered_data_categories = filtered_data_categories[~mask]
'''
mask = filtered_data_categories['medical_specialty'] == 'Pain Management'
filtered_data_categories = filtered_data_categories[~mask]
mask = filtered_data_categories['medical_specialty'] == 'General Medicine'
filtered_data_categories = filtered_data_categories[~mask]


mask = filtered_data_categories['medical_specialty'] == 'Neurosurgery'
filtered_data_categories.loc[mask, 'medical_specialty'] = 'Neurology'
mask = filtered_data_categories['medical_specialty'] == 'Nephrology'
filtered_data_categories.loc[mask, 'medical_specialty'] = 'Urology'


i=1
print('============Reduced Categories=====================')
for catName,dataCategory in final_data_categories:
    print('Cat:'+str(i)+' '+catName + ' : '+ str(len(dataCategory)) )
    i = i+1

print('============Reduced Categories=====================')


data = filtered_data_categories[['transcription', 'medical_specialty']]
data = data.drop(data[data['transcription'].isna()].index)
data.shape
```

Here We Used TF-IDF Vectorizer and Started Building Our Models:

Here is our vectorizer code. We will split it and feed it to the models.

```
from sklearn.feature_extraction.text import TfidfVectorizer


vectorizer = TfidfVectorizer(analyzer='word', stop_words='english',ngram_range=(1,3), max_df=0.75,min_df=5, use_idf=True, smooth_idf=True,sublinear_tf=T
tfIdfMat  = vectorizer.fit_transform(data['transcription'].tolist() )
feature_names = sorted(vectorizer.get_feature_names_out())
print(feature_names)
```

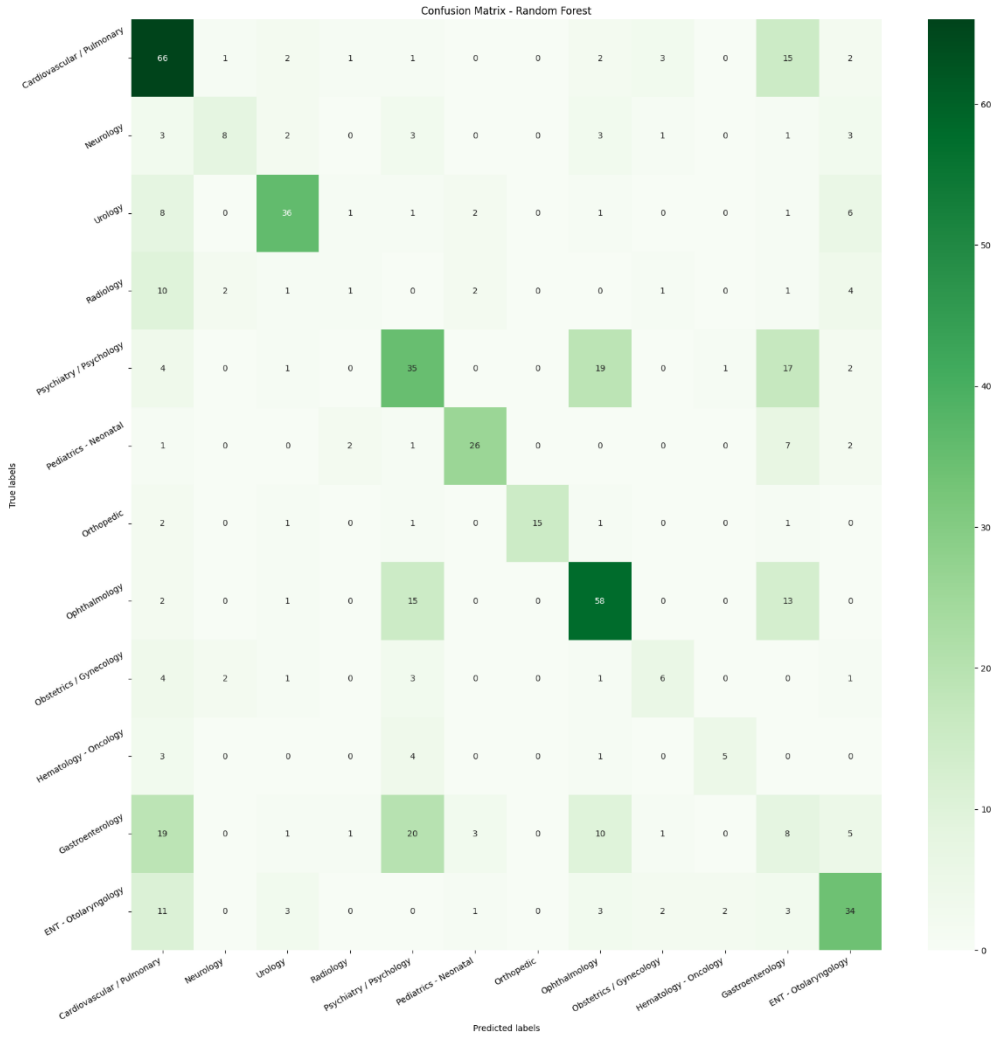
1. Multinomial Naïve Bayes



Accuracy: 0.3132530120481928

	precision	recall	f1-score	support
Cardiovascular / Pulmonary	0.25	0.97	0.40	93
ENT - Otolaryngology	0.00	0.00	0.00	24
Gastroenterology	0.00	0.00	0.00	56
Hematology - Oncology	0.00	0.00	0.00	22
Neurology	0.56	0.28	0.37	79
Obstetrics / Gynecology	0.00	0.00	0.00	39
Ophthalmology	0.00	0.00	0.00	21
Orthopedic	0.38	0.78	0.51	89
Pediatrics - Neonatal	0.00	0.00	0.00	18
Psychiatry / Psychology	0.00	0.00	0.00	13
Radiology	0.00	0.00	0.00	68
Urology	1.00	0.02	0.03	59
accuracy			0.31	581
macro avg	0.18	0.17	0.11	581
weighted avg	0.28	0.31	0.20	581

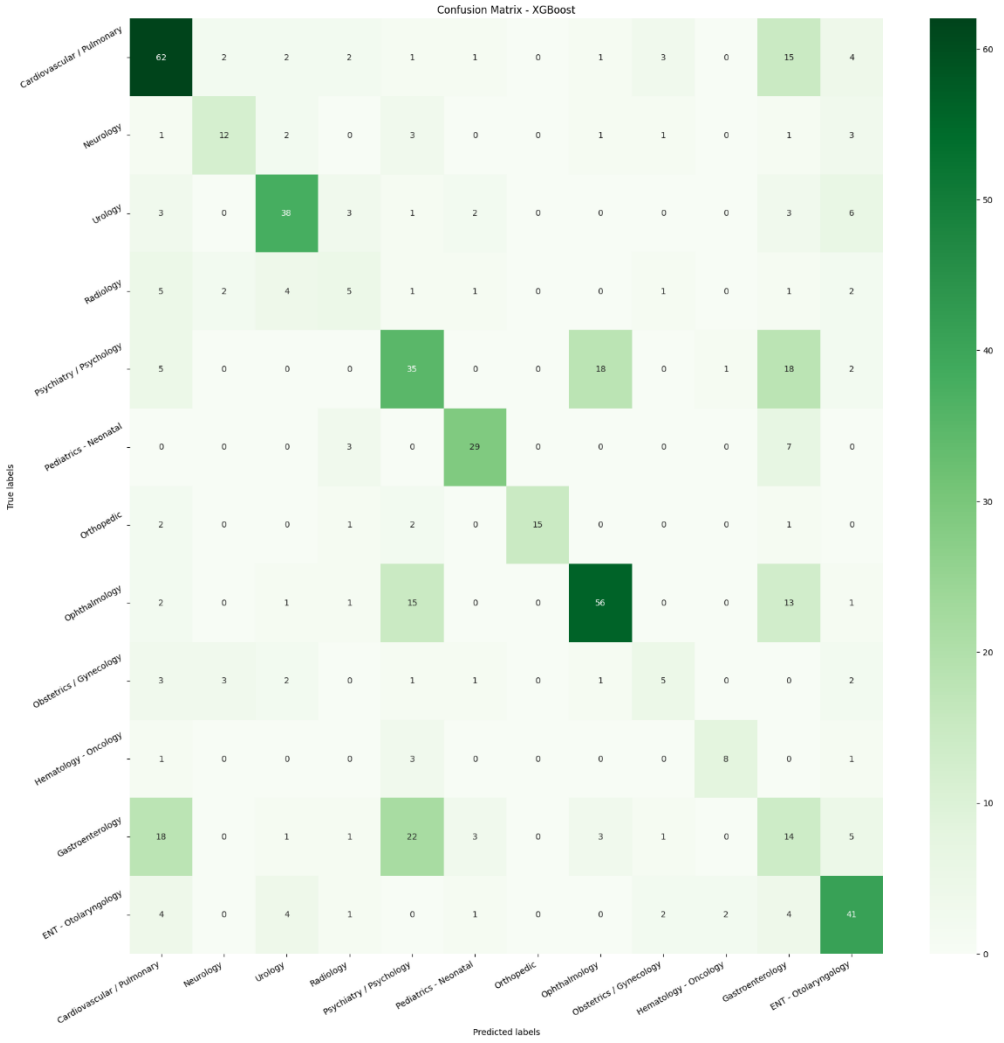
2. Random Forest



Random Forest Classifier Results:
Accuracy: 0.5129087779690189

	precision	recall	f1-score	support
Cardiovascular / Pulmonary	0.50	0.71	0.58	93
Neurology	0.42	0.44	0.43	79
Urology	0.58	0.58	0.58	59
Radiology	0.12	0.12	0.12	68
Psychiatry / Psychology	0.62	0.38	0.48	13
Pediatrics - Neonatal	0.43	0.33	0.38	18
Orthopedic	0.59	0.65	0.62	89
Ophthalmology	1.00	0.71	0.83	21
Obstetrics / Gynecology	0.76	0.67	0.71	39
Hematology - Oncology	0.17	0.05	0.07	22
Gastroenterology	0.73	0.64	0.69	56
ENT - Otolaryngology	0.62	0.33	0.43	24
accuracy			0.51	581
macro avg	0.54	0.47	0.49	581
weighted avg	0.52	0.51	0.51	581

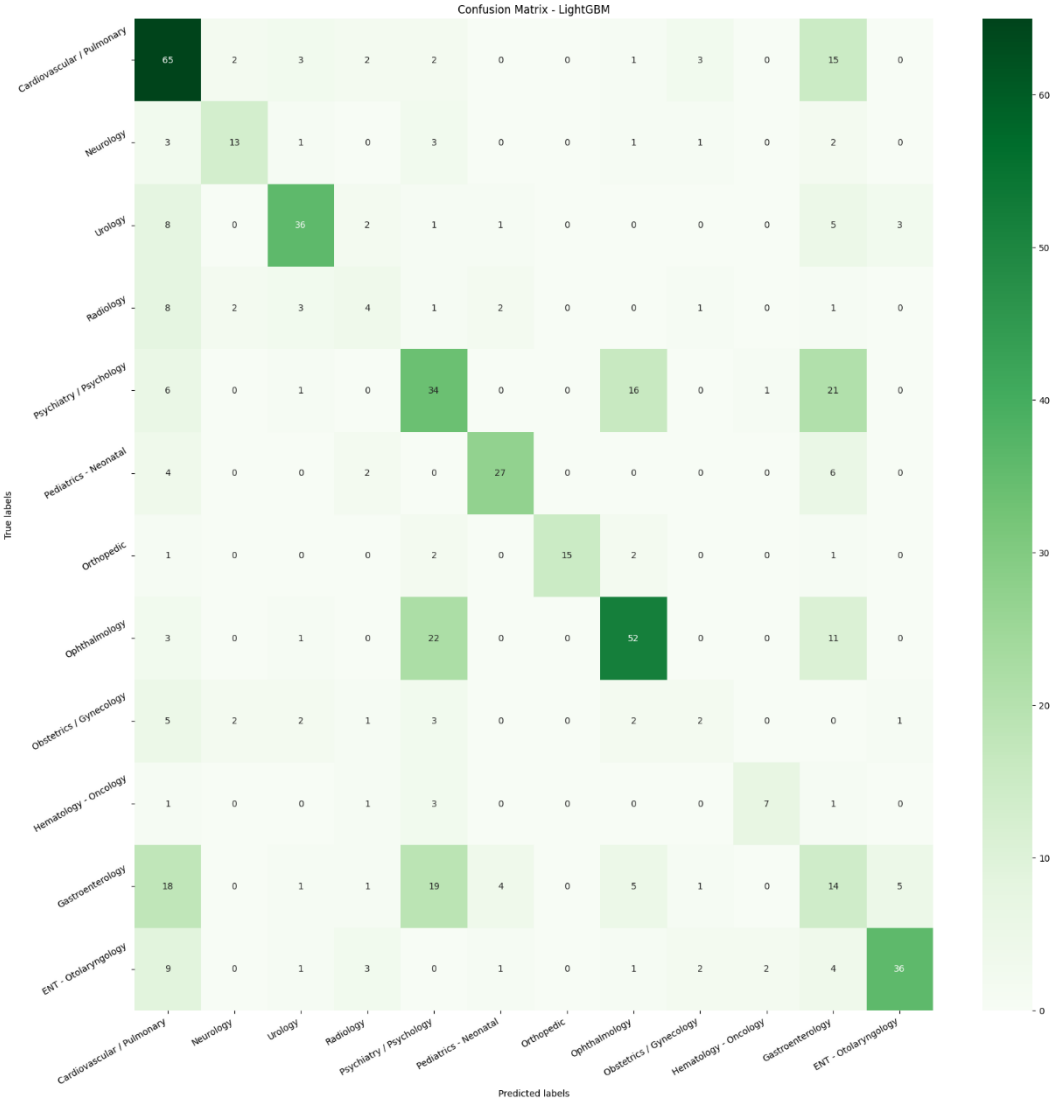
3. XGBoost



XGBoost Classifier Results:
Accuracy: 0.5507745266781411

	precision	recall	f1-score	support
0	0.58	0.67	0.62	93
1	0.63	0.50	0.56	24
2	0.70	0.68	0.69	56
3	0.29	0.23	0.26	22
4	0.42	0.44	0.43	79
5	0.76	0.74	0.75	39
6	1.00	0.71	0.83	21
7	0.70	0.63	0.66	89
8	0.38	0.28	0.32	18
9	0.73	0.62	0.67	13
10	0.18	0.21	0.19	68
11	0.61	0.69	0.65	59
accuracy			0.55	581
macro avg	0.58	0.53	0.55	581
weighted avg	0.56	0.55	0.55	581

4. LightGBM



LightGBM Classifier Results:
Accuracy: 0.5249569707401033

	precision	recall	f1-score	support
Cardiovascular / Pulmonary	0.50	0.70	0.58	93
ENT - Otolaryngology	0.68	0.54	0.60	24
Gastroenterology	0.73	0.64	0.69	56
Hematology - Oncology	0.25	0.18	0.21	22
Neurology	0.38	0.43	0.40	79
Obstetrics / Gynecology	0.77	0.69	0.73	39
Ophthalmology	1.00	0.71	0.83	21
Orthopedic	0.65	0.58	0.62	89
Pediatrics - Neonatal	0.20	0.11	0.14	18
Psychiatry / Psychology	0.70	0.54	0.61	13
Radiology	0.17	0.21	0.19	68
Urology	0.80	0.61	0.69	59
accuracy			0.52	581
macro avg	0.57	0.50	0.52	581
weighted avg	0.55	0.52	0.53	581

5. 1D CNN, LSTM and GRU

Accuracy: 0.5591397849462365

Classification Report:

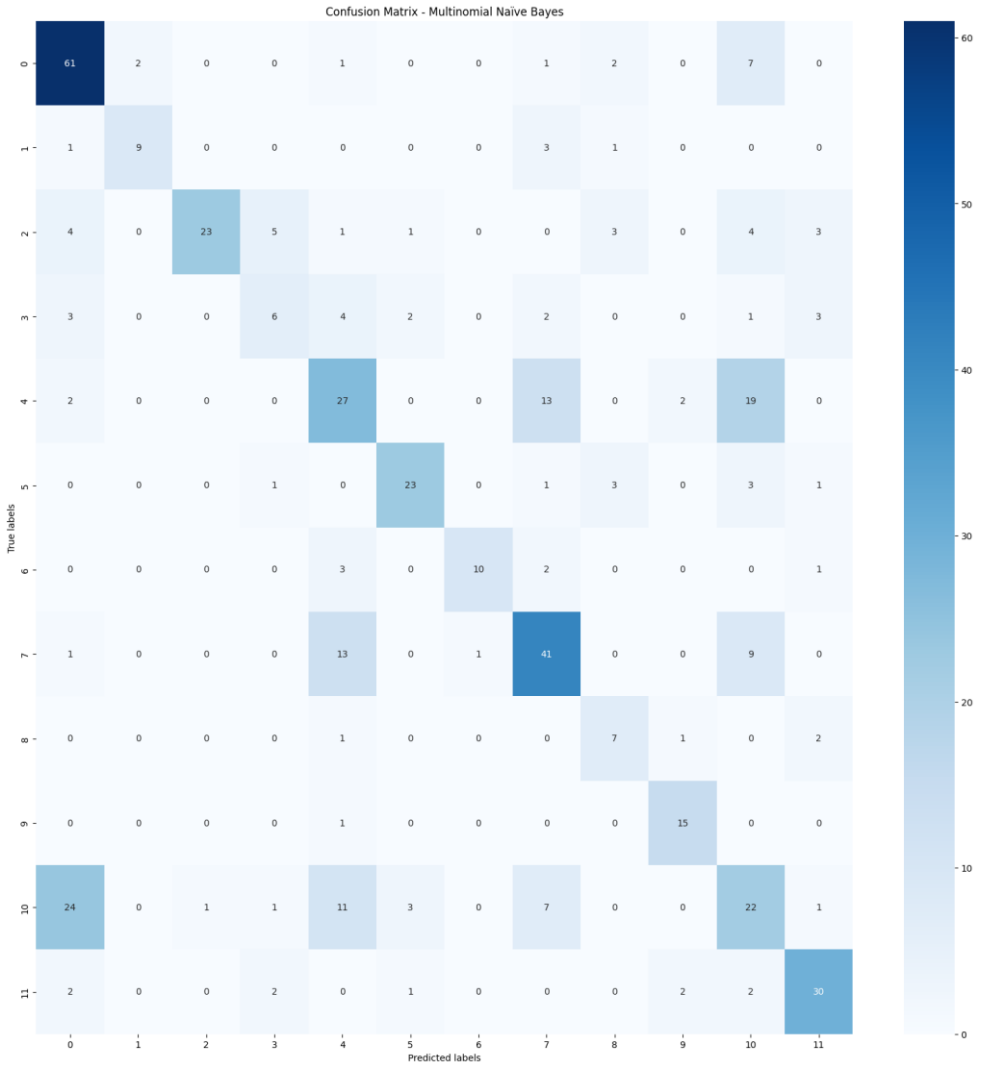
	precision	recall	f1-score	support
0	0.58	0.76	0.65	74
1	0.67	0.57	0.62	14
2	0.93	0.57	0.70	44
3	0.44	0.33	0.38	21
4	0.40	0.33	0.36	63
5	0.73	0.84	0.78	32
6	1.00	0.94	0.97	16
7	0.53	0.68	0.59	65
8	0.46	0.55	0.50	11
9	0.80	0.75	0.77	16
10	0.10	0.09	0.09	70
11	0.85	0.85	0.85	39
accuracy			0.56	465
macro avg	0.62	0.60	0.61	465
weighted avg	0.56	0.56	0.55	465

Here We Used Bag-of-Words (CountVectorizer) and Started Building Our Models:

Here is our vectorizer code. We will split it and feed it to the models.

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

X = data['transcription'].tolist()
y = data['medical_specialty']

# Use CountVectorizer to convert text data to a bag-of-words representation
vectorizer = CountVectorizer()
X_bow = vectorizer.fit_transform(X)

# Split the data
X_train_bow, X_val_bow, y_train, y_val = train_test_split(X_bow, y, test_size=0.2, random_state=42)
```

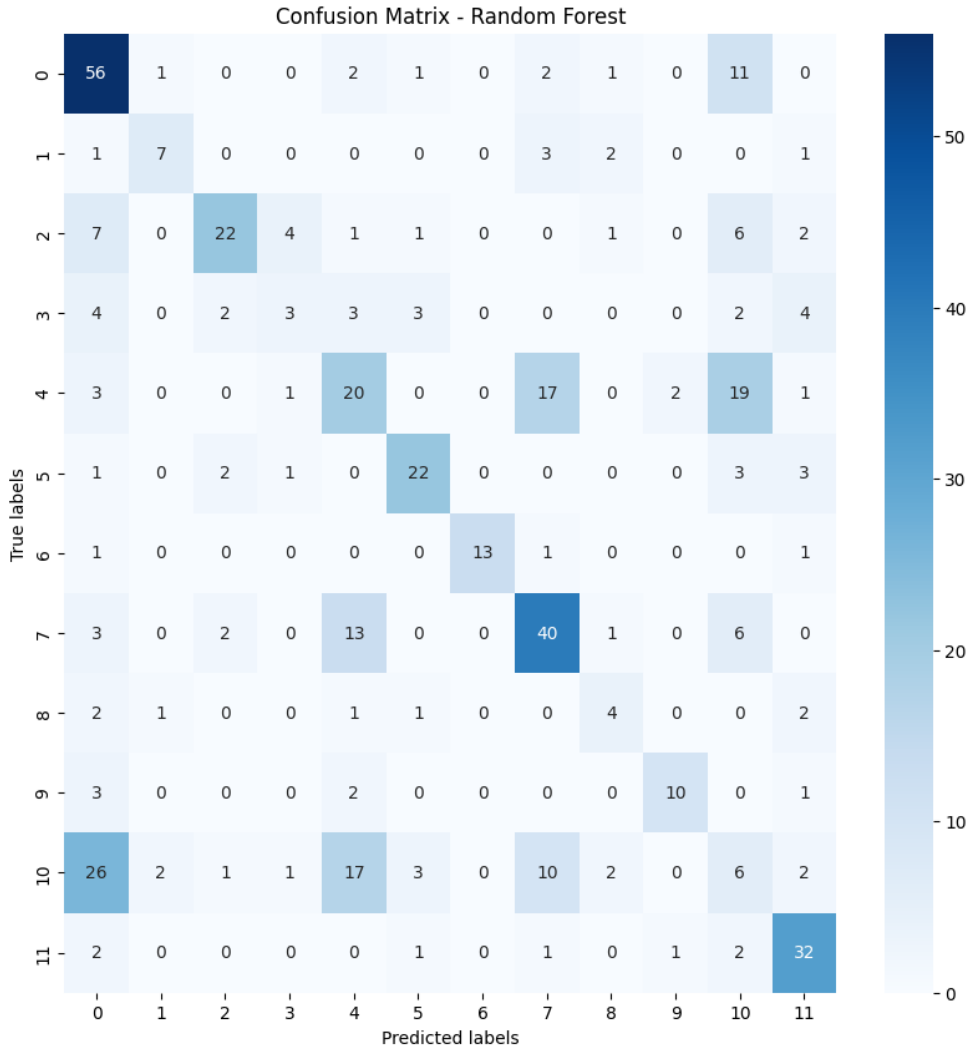
1. Multinomial Naïve Bayes



Multinomial Naïve Bayes Results:
Accuracy: 0.589247311827957

	precision	recall	f1-score	support
Cardiovascular / Pulmonary	0.62	0.82	0.71	74
ENT - Otolaryngology	0.82	0.64	0.72	14
Gastroenterology	0.96	0.52	0.68	44
Hematology - Oncology	0.40	0.29	0.33	21
Neurology	0.44	0.43	0.43	63
Obstetrics / Gynecology	0.77	0.72	0.74	32
Ophthalmology	0.91	0.62	0.74	16
Orthopedic	0.59	0.63	0.61	65
Pediatrics - Neonatal	0.44	0.64	0.52	11
Psychiatry / Psychology	0.75	0.94	0.83	16
Radiology	0.33	0.31	0.32	70
Urology	0.73	0.77	0.75	39
accuracy			0.59	465
macro avg	0.65	0.61	0.62	465
weighted avg	0.60	0.59	0.59	465

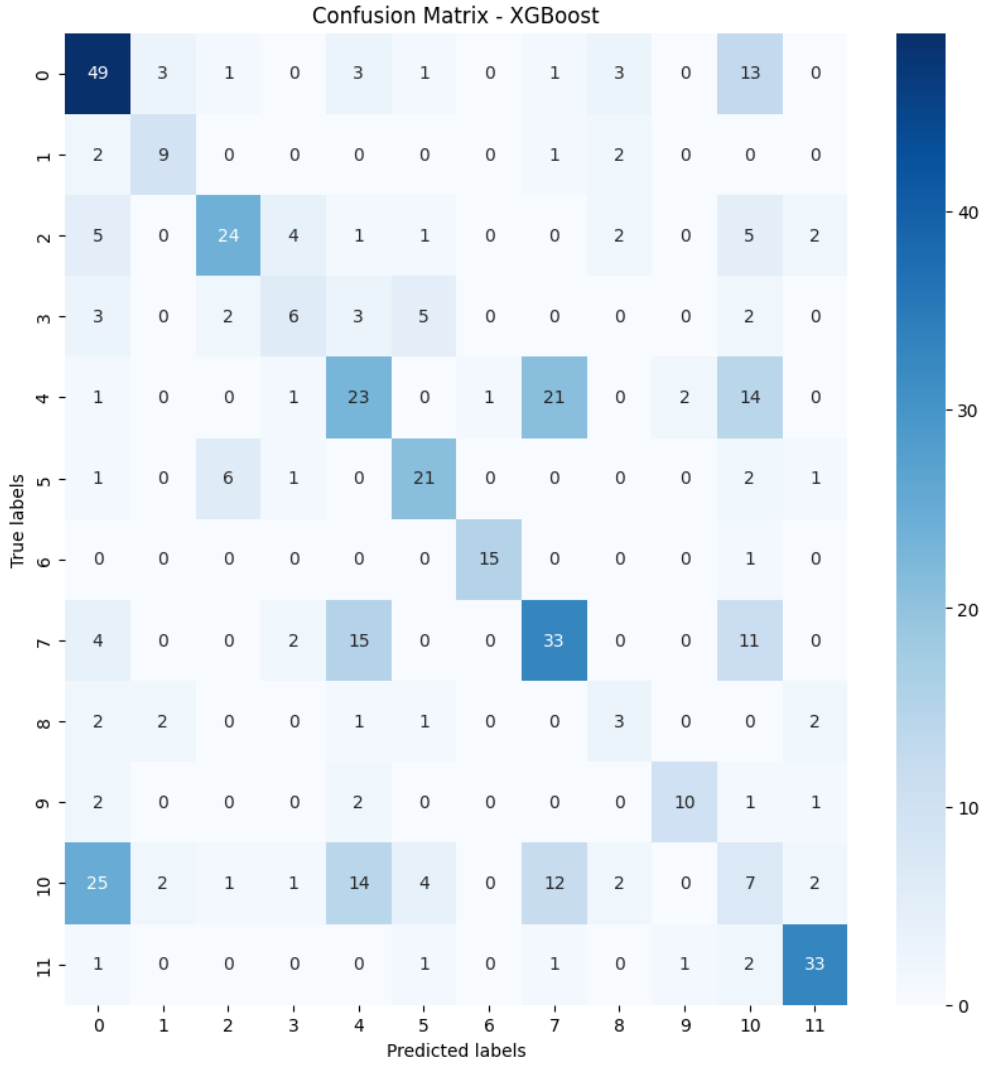
2. , Random Forest



Random Forest Results:
Accuracy: 0.5053763440860215

	precision	recall	f1-score	support
Cardiovascular / Pulmonary	0.51	0.76	0.61	74
ENT - Otolaryngology	0.64	0.50	0.56	14
Gastroenterology	0.76	0.50	0.60	44
Hematology - Oncology	0.30	0.14	0.19	21
Neurology	0.34	0.32	0.33	63
Obstetrics / Gynecology	0.69	0.69	0.69	32
Ophthalmology	1.00	0.81	0.90	16
Orthopedic	0.54	0.62	0.58	65
Pediatrics - Neonatal	0.36	0.36	0.36	11
Psychiatry / Psychology	0.77	0.62	0.69	16
Radiology	0.11	0.09	0.10	70
Urology	0.65	0.82	0.73	39
accuracy			0.51	465
macro avg	0.56	0.52	0.53	465
weighted avg	0.50	0.51	0.49	465

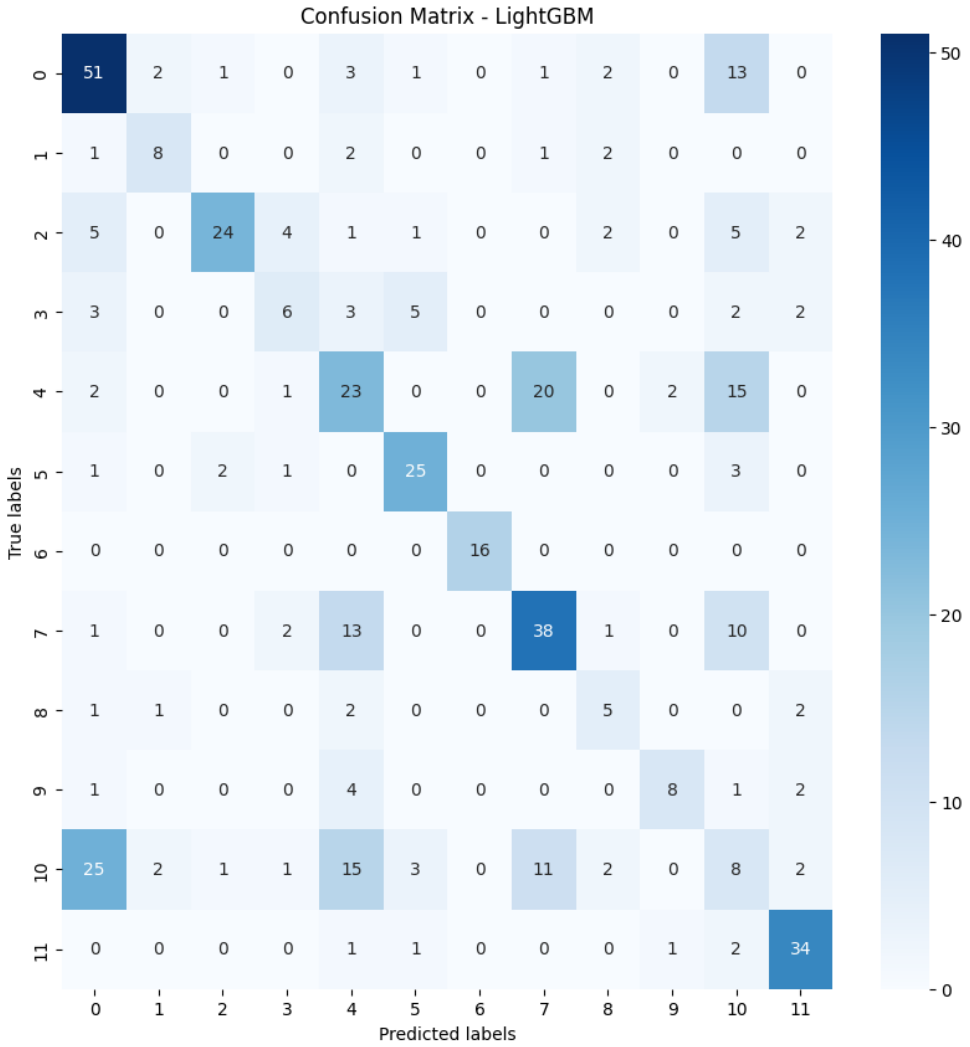
3. XGBoost



XGBoost Results:
Accuracy: 0.5010752688172043

	precision	recall	f1-score	support
0	0.52	0.66	0.58	74
1	0.56	0.64	0.60	14
2	0.71	0.55	0.62	44
3	0.40	0.29	0.33	21
4	0.37	0.37	0.37	63
5	0.62	0.66	0.64	32
6	0.94	0.94	0.94	16
7	0.48	0.51	0.49	65
8	0.25	0.27	0.26	11
9	0.77	0.62	0.69	16
10	0.12	0.10	0.11	70
11	0.80	0.85	0.83	39
accuracy			0.50	465
macro avg	0.54	0.54	0.54	465
weighted avg	0.49	0.50	0.49	465

4. LightGBM



```
LightGBM Results:
Accuracy: 0.5290322580645161
              precision    recall  f1-score   support

           0       0.56      0.69      0.62        74
           1       0.62      0.57      0.59        14
           2       0.86      0.55      0.67        44
           3       0.40      0.29      0.33        21
           4       0.34      0.37      0.35        63
           5       0.69      0.78      0.74        32
           6       1.00      1.00      1.00        16
           7       0.54      0.58      0.56        65
           8       0.36      0.45      0.40        11
           9       0.73      0.50      0.59        16
          10       0.14      0.11      0.12        70
          11       0.77      0.87      0.82        39

    accuracy                           0.53       465
   macro avg       0.58      0.56      0.57       465
weighted avg       0.53      0.53      0.52       465
```

5. 1D CNN, LSTM and GRU

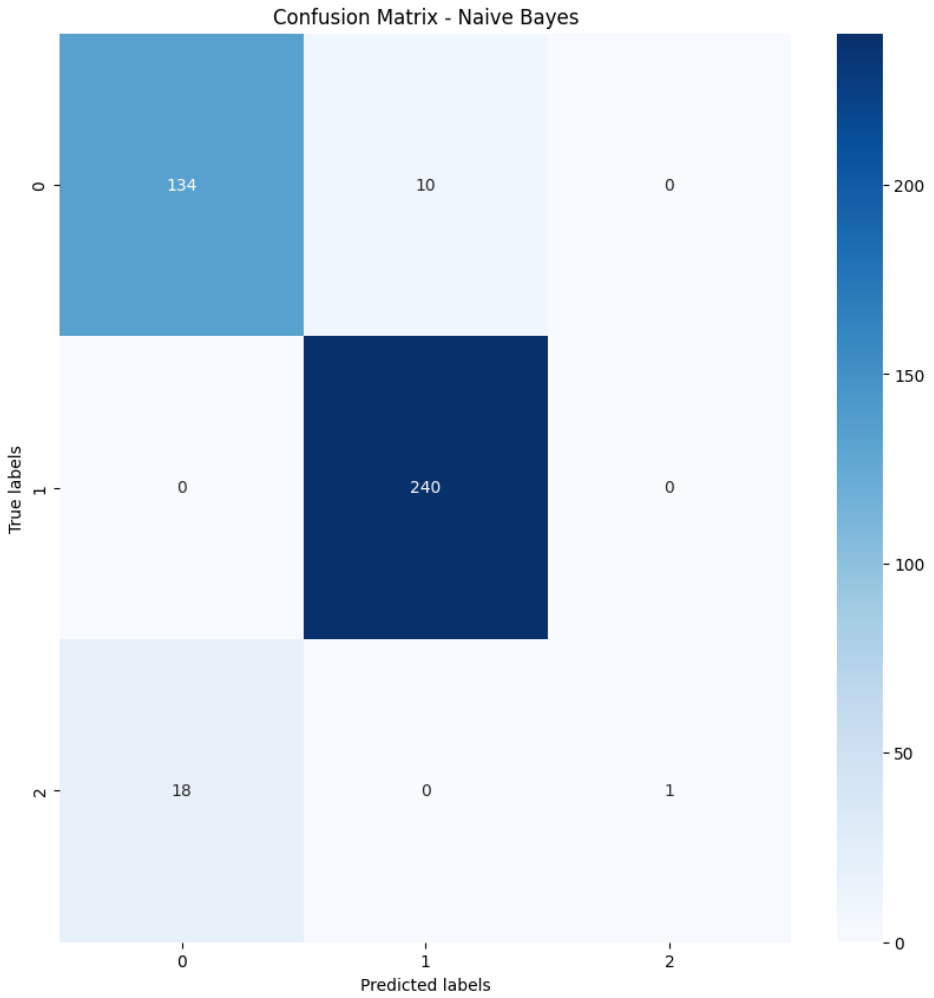
```
Confusion Matrix:
[[53  2  3  0  3  1  0  0  1  0 10  1]
 [ 1  7  0  1  3  0  0  0  2  0  0  0]
 [ 3  0 31  3  1  1  0  0  1  0  4  0]
 [ 2  0  2 12  1  2  0  0  0  0  2  0]
 [ 2  0  0  0 32  0  0 11  0  2 16  0]
 [ 0  0  5  2  0 23  0  0  0  0  2  0]
 [ 0  0  0  2  0  0 14  0  0  0  0  0]
 [ 0  0  0  2 16  0  0 39  0  0  8  0]
 [ 1  1  0  2  0  0  0  0  5  0  0  2]
 [ 0  0  0  0  4  0  0  0  0 11  0  1]
 [27  1  2  1 20  4  0  7  2  0  6  0]
 [ 0  0  1  2  0  1  0  0  0  1  2 32]]

Accuracy: 0.5698924731182796

Classification Report:
              precision    recall  f1-score   support

           0       0.60      0.72      0.65        74
           1       0.64      0.50      0.56        14
           2       0.70      0.70      0.70        44
           3       0.44      0.57      0.50        21
           4       0.40      0.51      0.45        63
           5       0.72      0.72      0.72        32
           6       1.00      0.88      0.93        16
           7       0.68      0.60      0.64        65
           8       0.45      0.45      0.45        11
           9       0.79      0.69      0.73        16
          10       0.12      0.09      0.10        70
          11       0.89      0.82      0.85        39

    accuracy                           0.57       465
   macro avg       0.62      0.60      0.61       465
weighted avg       0.56      0.57      0.56       465
```

Here We Used TF-IDF Vectorizer for the First Part Output and Started Building Our Models:

Here is our vectorizer code. We will split it and feed it to the models.

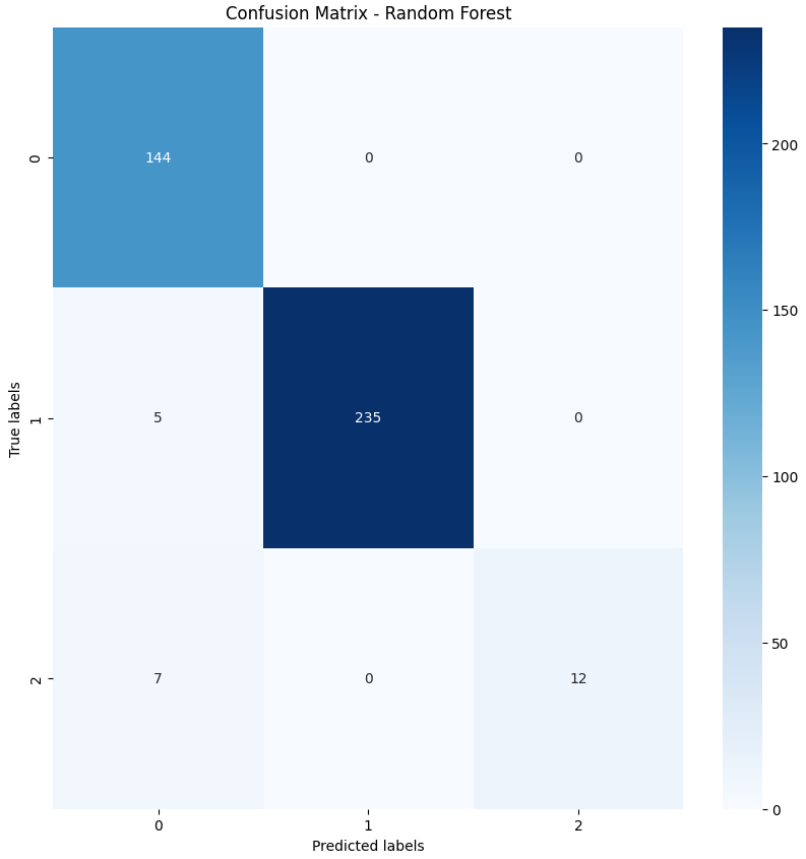
```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer

# Specify the file path
file_path = "C:/spark/output.csv"

# Read the CSV file into a DataFrame
output_df = pd.read_csv(file_path)

# Create a TfidfVectorizer instance
tfidf_vectorizer = TfidfVectorizer()

# Fit and transform the "Text" column
tfidf_matrix = tfidf_vectorizer.fit_transform(output_df['Text'])

# Convert the TF-IDF matrix to a DataFrame
tfidf_df = pd.DataFrame(tfidf_matrix.toarray(), columns=tfidf_vectorizer.get_feature_names_out())

# Concatenate the TF-IDF DataFrame with the original DataFrame
output_df = pd.concat([output_df, tfidf_df], axis=1)
```

1. Multinomial Naïve Bayes



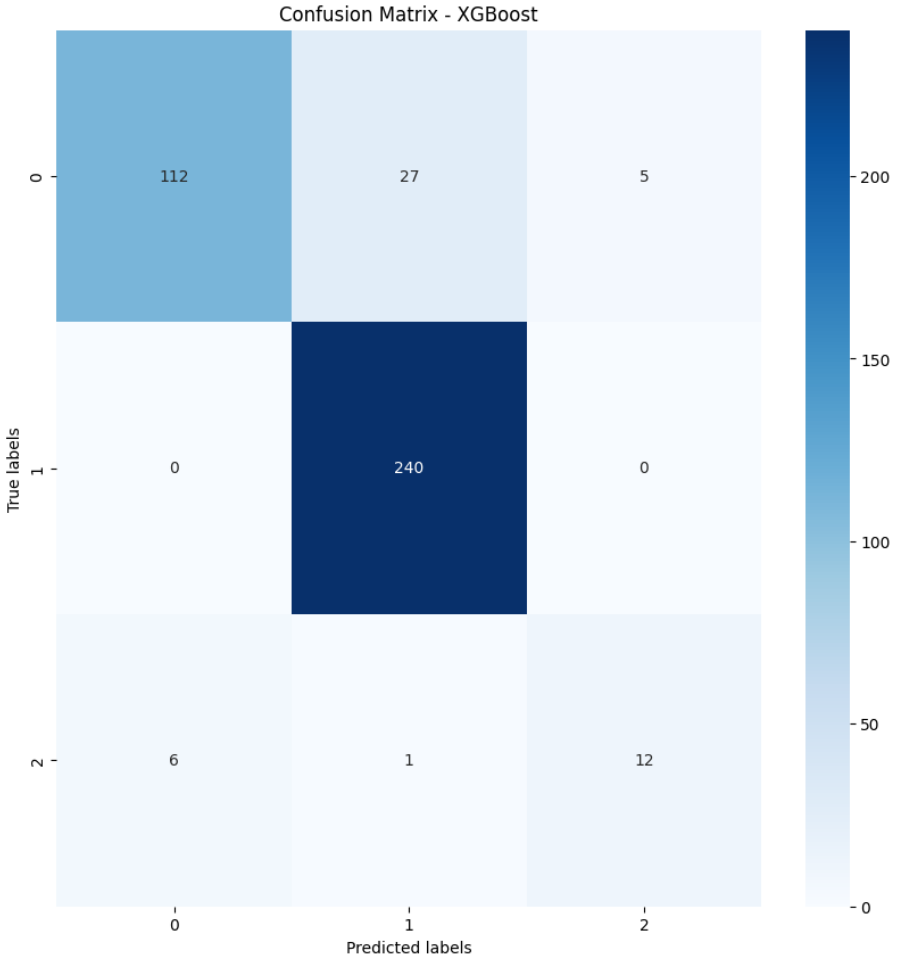
```
Naive Bayes Results:
Accuracy: 0.9305210918114144
              precision    recall  f1-score   support

    CHEMICAL       0.88      0.93      0.91       144
     DISEASE       0.96      1.00      0.98       240
  DRUG_DOSE        1.00      0.05      0.10        19

    accuracy                           0.93       403
   macro avg       0.95      0.66      0.66       403
weighted avg       0.93      0.93      0.91       403
```

2. Random Forest



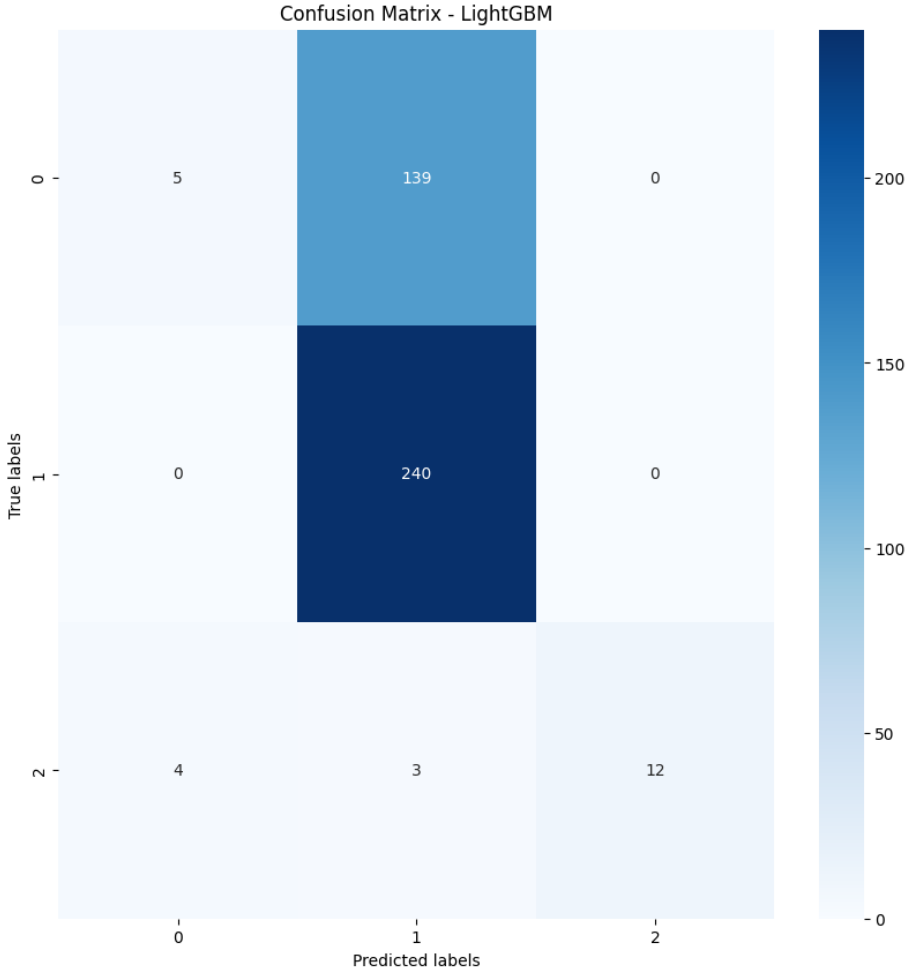

```
Random Forest Results:
Accuracy: 0.9702233250620348
              precision    recall  f1-score   support

    CHEMICAL       0.92      1.00      0.96       144
     DISEASE       1.00      0.98      0.99       240
  DRUG_DOSE        1.00      0.63      0.77        19

    accuracy                           0.97       403
   macro avg       0.97      0.87      0.91       403
weighted avg       0.97      0.97      0.97       403
```

3. XGBoost



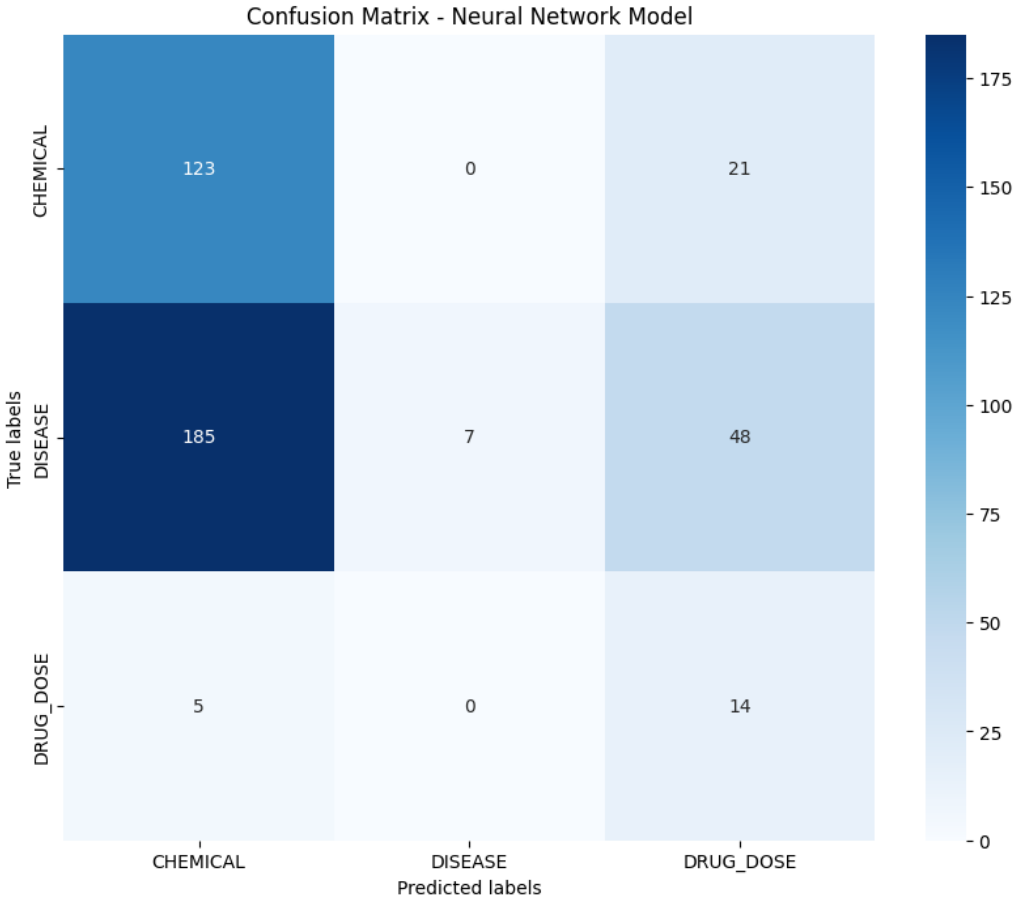

```
XGBoost Results:
Accuracy: 0.9032258064516129
              precision    recall  f1-score   support

           0       0.95      0.78      0.85       144
           1       0.90      1.00      0.94       240
           2       0.71      0.63      0.67        19

    accuracy                           0.90       403
   macro avg       0.85      0.80      0.82       403
weighted avg       0.91      0.90      0.90       403
```

4. LightGBM




```
LightGBM Results:
Accuracy: 0.6377171215880894
              precision    recall  f1-score   support

           0       0.56      0.03      0.07       144
           1       0.63      1.00      0.77       240
           2       1.00      0.63      0.77        19

    accuracy                           0.64       403
   macro avg       0.73      0.56      0.54       403
weighted avg       0.62      0.64      0.52       403
```

5. 1D CNN, LSTM and GRU



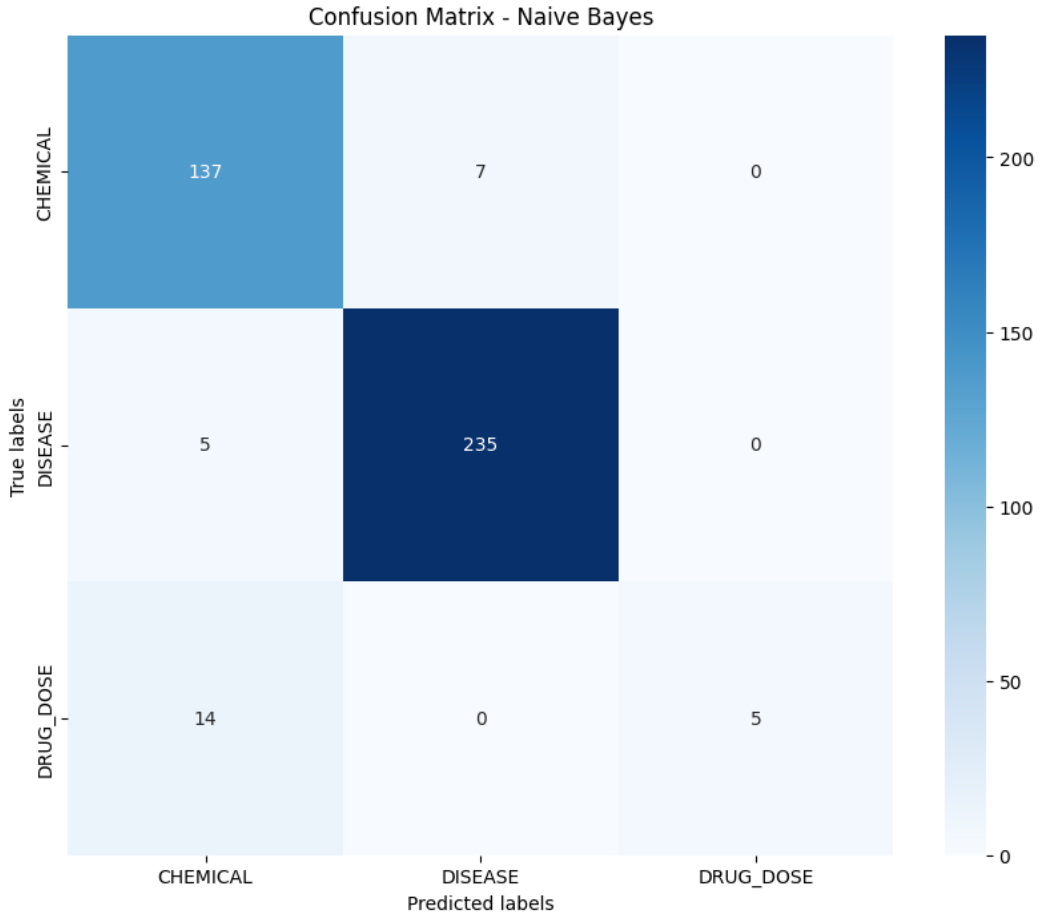

```
Accuracy: 0.3573200992555831
Classification Report:
              precision    recall  f1-score   support

    CHEMICAL       0.39      0.85      0.54       144
     DISEASE       1.00      0.03      0.06       240
  DRUG_DOSE        0.17      0.74      0.27        19

    accuracy                           0.36       403
   macro avg       0.52      0.54      0.29       403
weighted avg       0.74      0.36      0.24       403
```

Here We Used CountVectorizer for the First Part Output and Started Building Our Models:

Here is our vectorizer code. We will split it and feed it to the models.

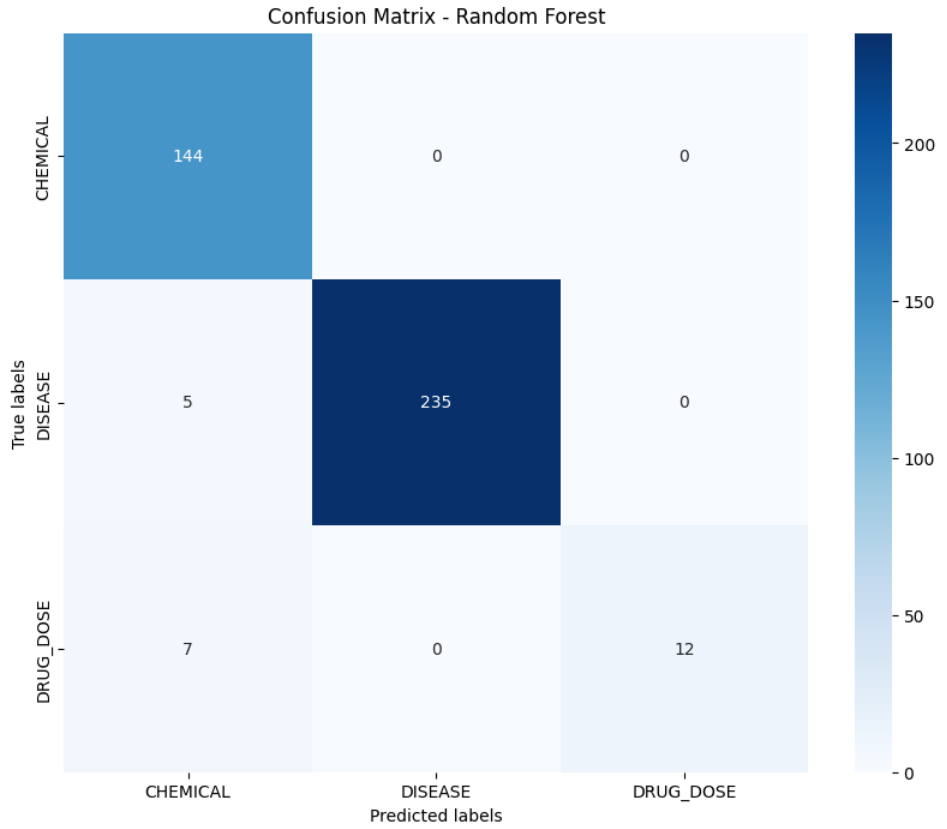
```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split

# Read the CSV file into a DataFrame
file_path = "C:/spark/output.csv"
output_df = pd.read_csv(file_path)

text_data = output_df['Text']
target_variable = output_df['Entity Type']
# Create a CountVectorizer instance
vectorizer = CountVectorizer()

# Fit and transform the text data
X_text_vectorized = vectorizer.fit_transform(text_data)
```

1. Multinomial Naïve Bayes



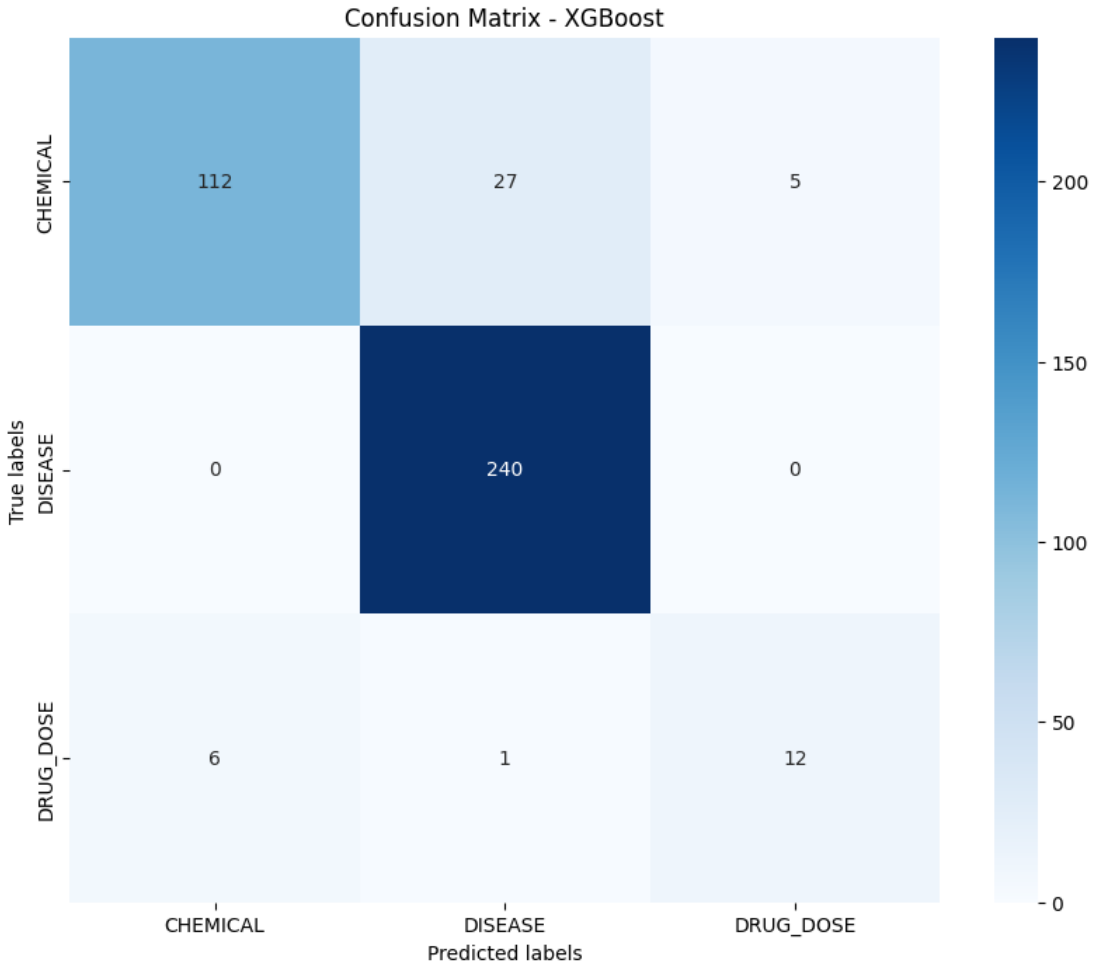
```
Accuracy: 0.9354838709677419
Classification Report:
              precision    recall  f1-score   support

    CHEMICAL       0.88      0.95      0.91       144
     DISEASE       0.97      0.98      0.98       240
  DRUG_DOSE        1.00      0.26      0.42        19

    accuracy                           0.94       403
   macro avg       0.95      0.73      0.77       403
weighted avg       0.94      0.94      0.93       403
```

2. Random Forest




```
Accuracy: 0.9702233250620348
Classification Report:
              precision    recall  f1-score   support

    CHEMICAL       0.92      1.00      0.96       144
     DISEASE       1.00      0.98      0.99       240
  DRUG_DOSE        1.00      0.63      0.77        19

    accuracy                           0.97       403
   macro avg       0.97      0.87      0.91       403
weighted avg       0.97      0.97      0.97       403
```

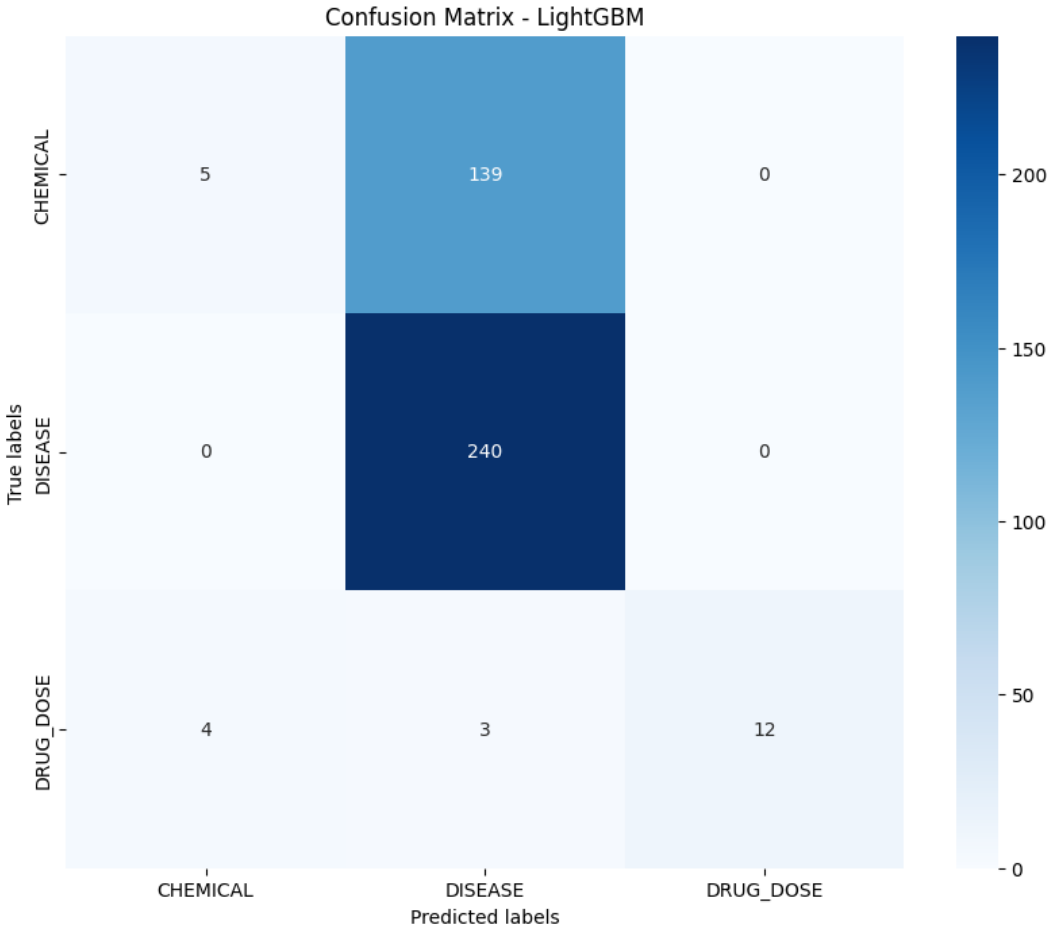
3. XGBoost



Accuracy: 0.9032258064516129
Classification Report:

	precision	recall	f1-score	support
CHEMICAL	0.95	0.78	0.85	144
DISEASE	0.90	1.00	0.94	240
DRUG_DOSE	0.71	0.63	0.67	19
accuracy			0.90	403
macro avg	0.85	0.80	0.82	403
weighted avg	0.91	0.90	0.90	403

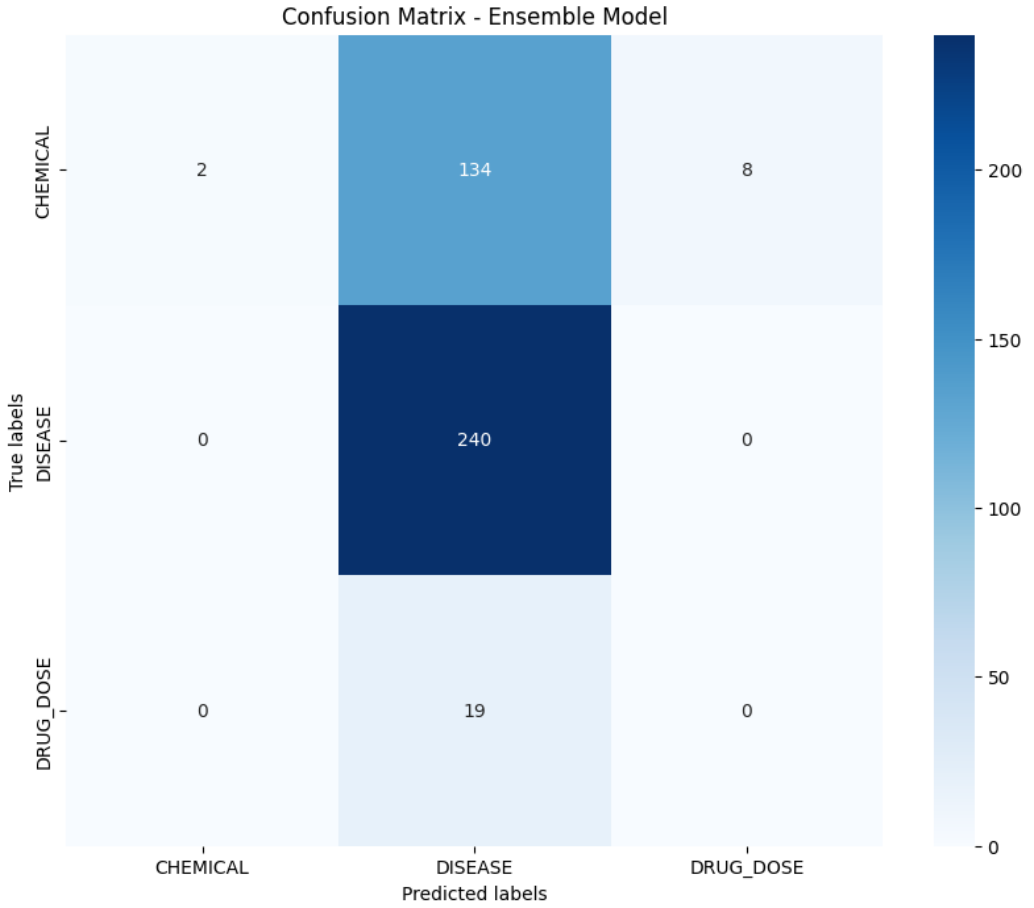
4. LightGBM



Accuracy: 0.6377171215880894
Classification Report:

	precision	recall	f1-score	support
CHEMICAL	0.56	0.03	0.07	144
DISEASE	0.63	1.00	0.77	240
DRUG_DOSE	1.00	0.63	0.77	19
accuracy			0.64	403
macro avg	0.73	0.56	0.54	403
weighted avg	0.62	0.64	0.52	403

5. 1D CNN, LSTM and GRU



Accuracy: 0.6004962779156328
Classification Report:

	precision	recall	f1-score	support
CHEMICAL	1.00	0.01	0.03	144
DISEASE	0.61	1.00	0.76	240
DRUG_DOSE	0.00	0.00	0.00	19
accuracy			0.60	403
macro avg	0.54	0.34	0.26	403
weighted avg	0.72	0.60	0.46	403

And Finally Applying SMOTE to Our Best Models:

For the first input our best model was Multinominal Naïve Bayes with the CountVectorizer.

Here is before oversampling scores.

Multinomial Naïve Bayes Results:
Accuracy: 0.589247311827957

	precision	recall	f1-score	support
Cardiovascular / Pulmonary	0.62	0.82	0.71	74
ENT - Otolaryngology	0.82	0.64	0.72	14
Gastroenterology	0.96	0.52	0.68	44
Hematology - Oncology	0.40	0.29	0.33	21
Neurology	0.44	0.43	0.43	63
Obstetrics / Gynecology	0.77	0.72	0.74	32
Ophthalmology	0.91	0.62	0.74	16
Orthopedic	0.59	0.63	0.61	65
Pediatrics - Neonatal	0.44	0.64	0.52	11
Psychiatry / Psychology	0.75	0.94	0.83	16
Radiology	0.33	0.31	0.32	70
Urology	0.73	0.77	0.75	39
accuracy			0.59	465
macro avg	0.65	0.61	0.62	465
weighted avg	0.60	0.59	0.59	465

And here is after oversampling scores.

Multinomial Naïve Bayes Results:
Accuracy: 0.578494623655914

	precision	recall	f1-score	support
Cardiovascular / Pulmonary	0.65	0.76	0.70	74
ENT - Otolaryngology	0.77	0.71	0.74	14
Gastroenterology	0.96	0.50	0.66	44
Hematology - Oncology	0.32	0.38	0.35	21
Neurology	0.44	0.33	0.38	63
Obstetrics / Gynecology	0.75	0.75	0.75	32
Ophthalmology	0.93	0.81	0.87	16
Orthopedic	0.63	0.63	0.63	65
Pediatrics - Neonatal	0.31	0.73	0.43	11
Psychiatry / Psychology	0.71	0.94	0.81	16
Radiology	0.32	0.33	0.32	70
Urology	0.70	0.72	0.71	39
accuracy			0.58	465
macro avg	0.62	0.63	0.61	465
weighted avg	0.60	0.58	0.58	465

And for the second input our best model was Random Forest. So we applied SMOTE and compared results for it.

Here are the results before oversampling.

```
Accuracy: 0.9702233250620348
Classification Report:
              precision    recall  f1-score   support

    CHEMICAL       0.92      1.00      0.96       144
     DISEASE       1.00      0.98      0.99       240
   DRUG_DOSE       1.00      0.63      0.77        19

    accuracy                           0.97       403
   macro avg       0.97      0.87      0.91       403
weighted avg       0.97      0.97      0.97       403
```

And here are the results after oversampling.

```
Accuracy: 0.9305210918114144
Classification Report:
              precision    recall  f1-score   support

    CHEMICAL       0.95      0.89      0.92       144
     DISEASE       1.00      0.98      0.99       240
   DRUG_DOSE       0.36      0.63      0.46        19

    accuracy                           0.93       403
   macro avg       0.77      0.83      0.79       403
weighted avg       0.95      0.93      0.94       403
```

While we saw some improvements on the first input side, for the second input side it got worse.

REFERENCES

1. https://www.kaggle.com/datasets/tboyle10/medicaltranscriptions
2. https://www.kaggle.com/code/ritheshsreenivasan/clinical-text-classification