BERZIET UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ENCS3340

Project 2

Emotion prediction(machine learning)

Prepared by:

Omar Daraghmeh 1200162

Mahmoud Shihab 1182143

Supervised by:

Dr. Aziz Qaroush

Feb/23

Table of Contents

Table of Figures:

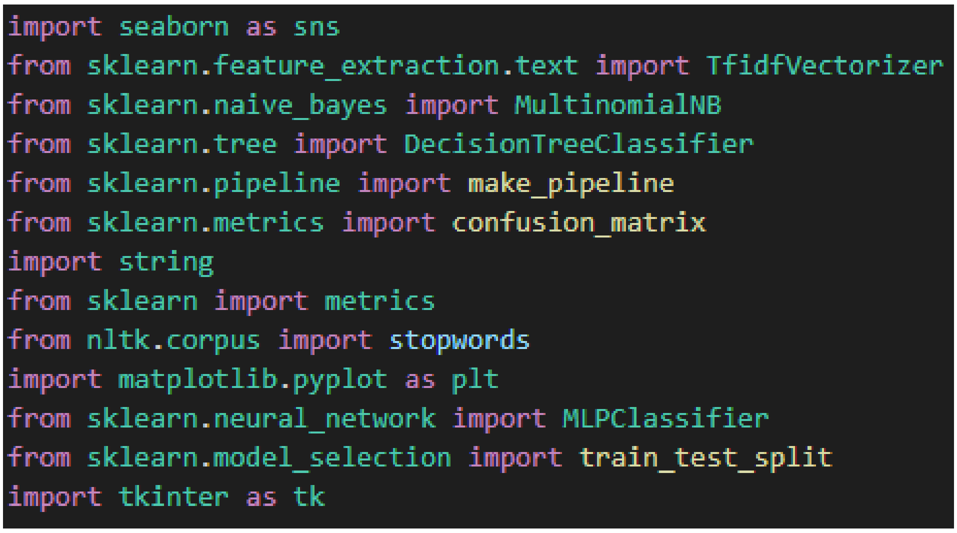
Program implementation:

We used python to implement our project using VScode environment .

for GUI we used a library called tkinter and for the visualizing we used matplot library and seaborn.

For the classifiers we used sklearn libraries, for the naïve bayez we used MultinomialNB, for the neural network we used MLPClassifier, for the tree decision classifier we used DecisionTreeClassifier

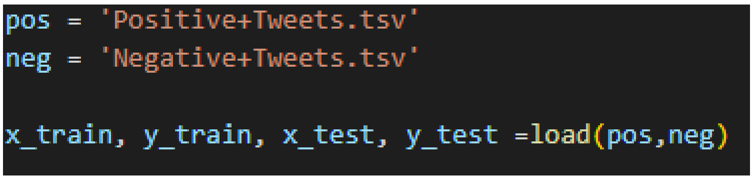
For vectoring the words we used TfidfVectorizer.



```python
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.pipeline import make_pipeline
from sklearn.metrics import confusion_matrix
import string
from sklearn import metrics
from nltk.corpus import stopwords
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
import tkinter as tk
```

Figure 1:implementations

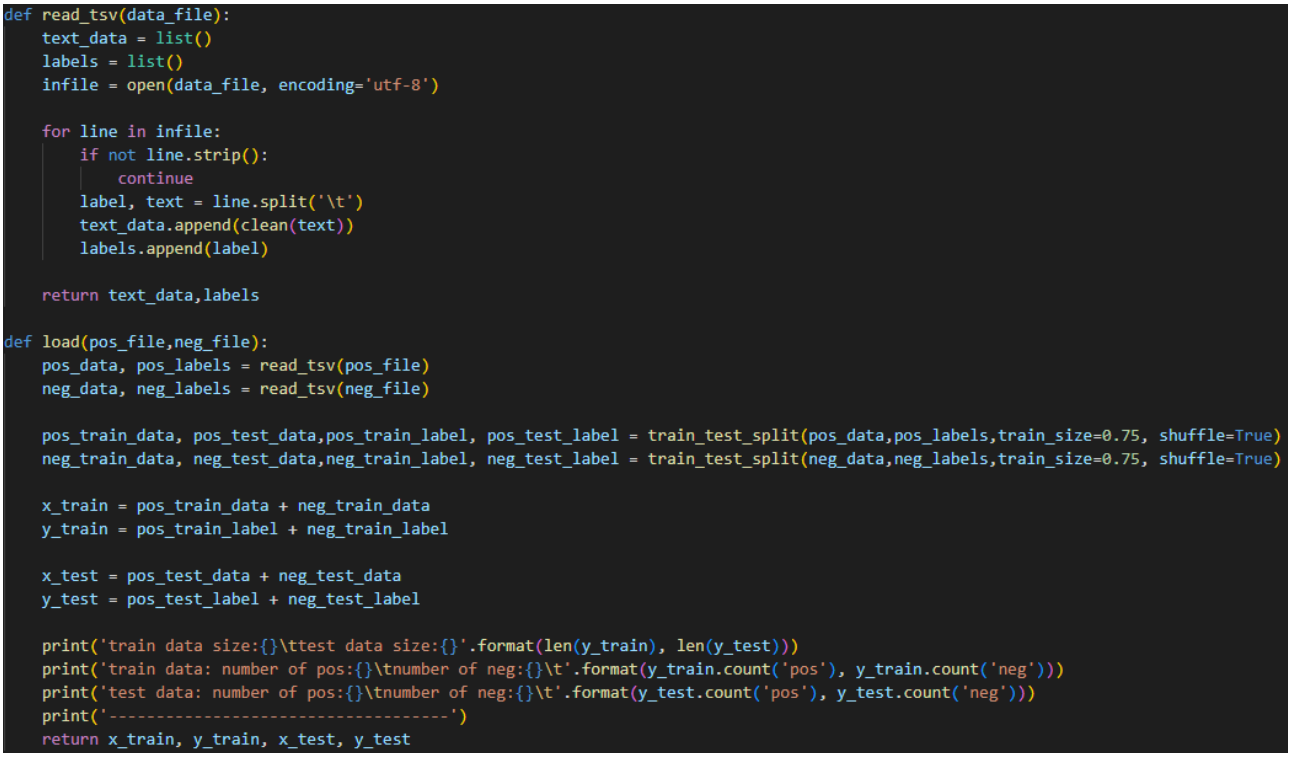
We read the file and put it in lists :



```python
pos = 'Positive+Tweets.tsv'
neg = 'Negative+Tweets.tsv'

x_train, y_train, x_test, y_test =load(pos,neg)
```

Figure 2:reading

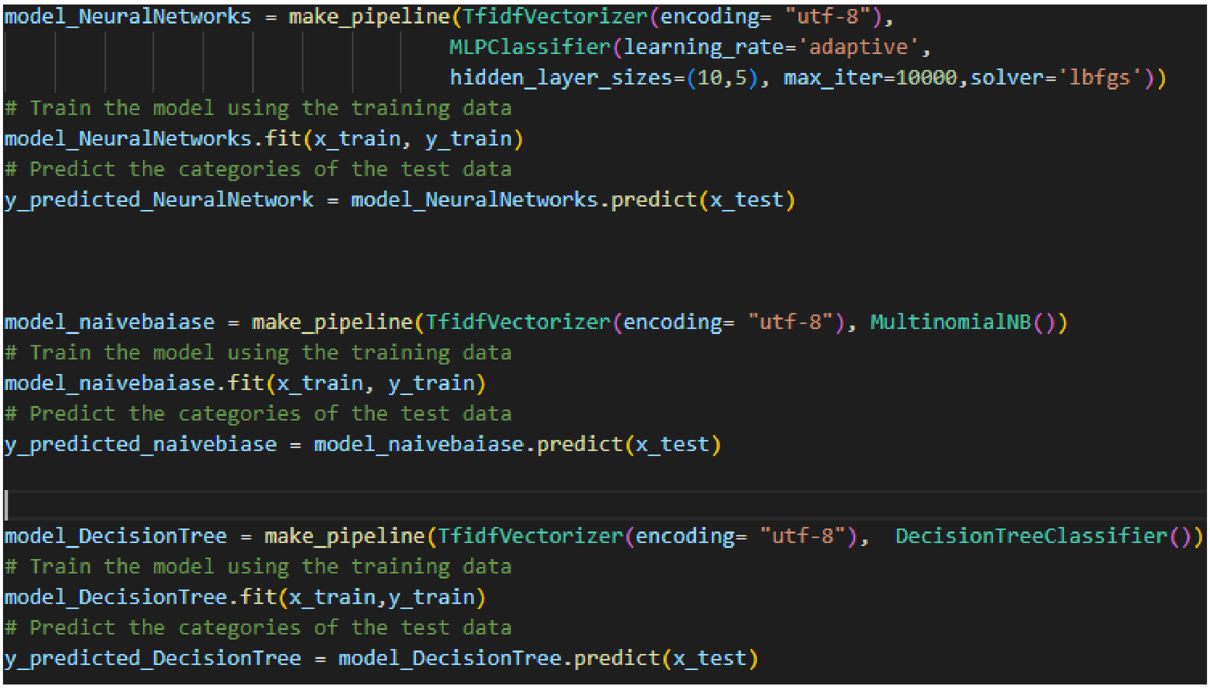
Using those method and we split the data with .75 for the train and .25 for testing :



```python
def read_tsv(data_file):
    text_data = list()
    labels = list()
    infile = open(data_file, encoding='utf-8')

    for line in infile:
        if not line.strip():
            continue
        label, text = line.split('\t')
        text_data.append(clean(text))
        labels.append(label)

    return text_data,labels

def load(pos_file,neg_file):
    pos_data, pos_labels = read_tsv(pos_file)
    neg_data, neg_labels = read_tsv(neg_file)

    pos_train_data, pos_test_data,pos_train_label, pos_test_label = train_test_split(pos_data,pos_labels,train_size=0.75, shuffle=True)
    neg_train_data, neg_test_data,neg_train_label, neg_test_label = train_test_split(neg_data,neg_labels,train_size=0.75, shuffle=True)

    x_train = pos_train_data + neg_train_data
    y_train = pos_train_label + neg_train_label

    x_test = pos_test_data + neg_test_data
    y_test = pos_test_label + neg_test_label

    print('train data size:{}\ttest data size:{}'.format(len(y_train), len(y_test)))
    print('train data: number of pos:{}\tnumber of neg:{}\t'.format(y_train.count('pos'), y_train.count('neg')))
    print('test data: number of pos:{}\tnumber of neg:{}\t'.format(y_test.count('pos'), y_test.count('neg')))
    print('------------------------------------')
    return x_train, y_train, x_test, y_test
```

Figure 3:loading and cleaning methods

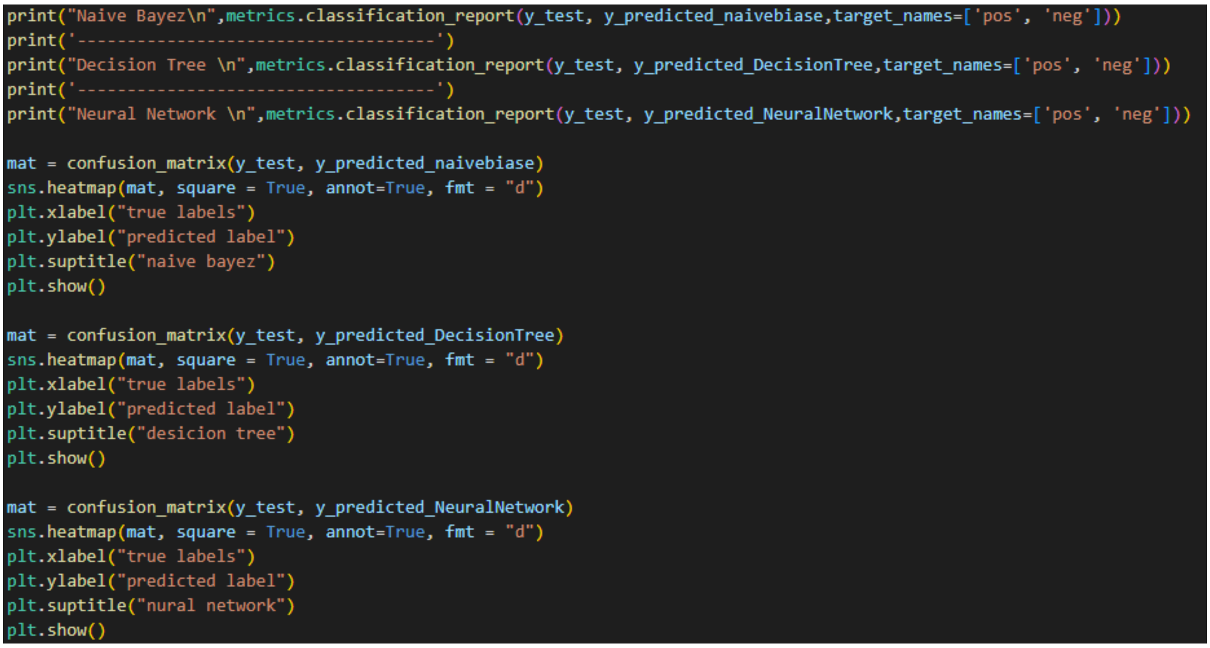
We use the imported models from sklearn to fit the data and start training and testing for the neural network we put the maximum iteration 10000 because it will automatically stop changing the values when there is 10 epochs with no changing for the error :



```python
model_NeuralNetworks = make_pipeline(TfidfVectorizer(encoding= "utf-8"),
                                    MLPClassifier(learning_rate='adaptive',
                                    hidden_layer_sizes=(10,5), max_iter=10000,solver='lbfgs'))
# Train the model using the training data
model_NeuralNetworks.fit(x_train, y_train)
# Predict the categories of the test data
y_predicted_NeuralNetwork = model_NeuralNetworks.predict(x_test)



model_naivebaiase = make_pipeline(TfidfVectorizer(encoding= "utf-8"), MultinomialNB())
# Train the model using the training data
model_naivebaiase.fit(x_train, y_train)
# Predict the categories of the test data
y_predicted_naivebiase = model_naivebaiase.predict(x_test)


model_DecisionTree = make_pipeline(TfidfVectorizer(encoding= "utf-8"),  DecisionTreeClassifier())
# Train the model using the training data
model_DecisionTree.fit(x_train,y_train)
# Predict the categories of the test data
y_predicted_DecisionTree = model_DecisionTree.predict(x_test)
```

Figure 4:starting models

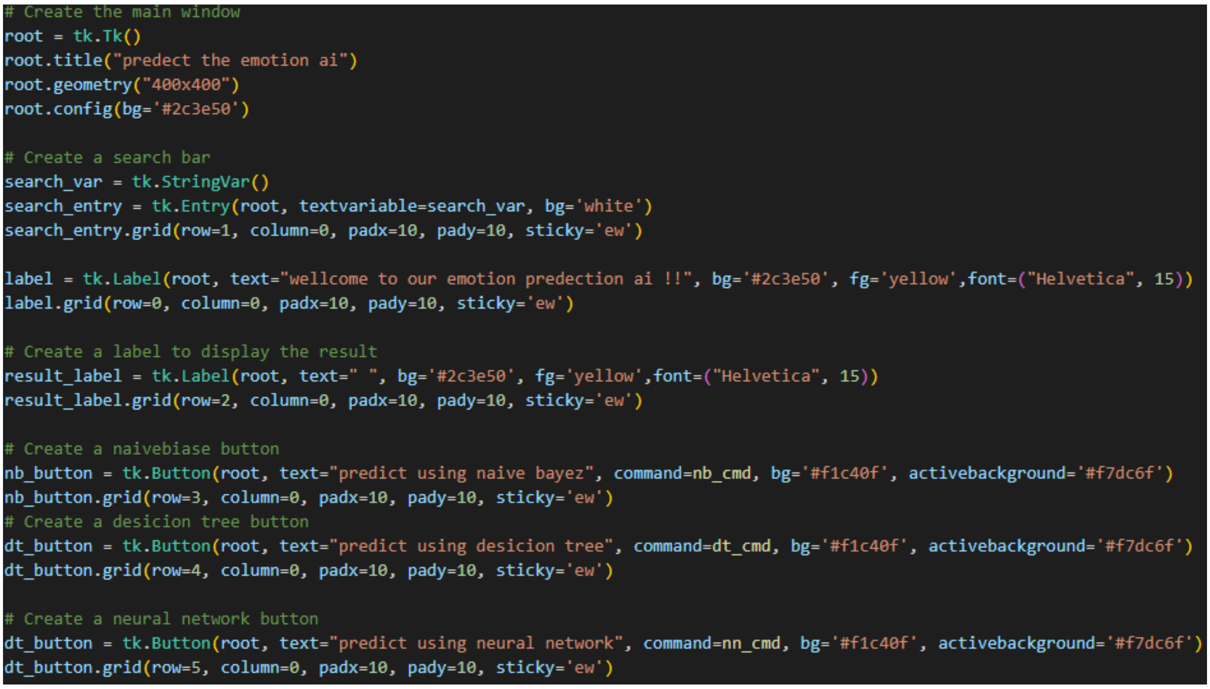
We print the accuracy , the recall , f1score and precision then we visualize the predicted value and the true values:



```
print("Naive Bayez\n",metrics.classification_report(y_test, y_predicted_naivebiase,target_names=['pos', 'neg']))
print('-------------------------------------')
print("Decision Tree \n",metrics.classification_report(y_test, y_predicted_DecisionTree,target_names=['pos', 'neg']))
print('-------------------------------------')
print("Neural Network \n",metrics.classification_report(y_test, y_predicted_NeuralNetwork,target_names=['pos', 'neg']))

mat = confusion_matrix(y_test, y_predicted_naivebiase)
sns.heatmap(mat, square = True, annot=True, fmt = "d")
plt.xlabel("true labels")
plt.ylabel("predicted label")
plt.suptitle("naive bayez")
plt.show()

mat = confusion_matrix(y_test, y_predicted_DecisionTree)
sns.heatmap(mat, square = True, annot=True, fmt = "d")
plt.xlabel("true labels")
plt.ylabel("predicted label")
plt.suptitle("desicion tree")
plt.show()

mat = confusion_matrix(y_test, y_predicted_NeuralNetwork)
sns.heatmap(mat, square = True, annot=True, fmt = "d")
plt.xlabel("true labels")
plt.ylabel("predicted label")
plt.suptitle("nural network")
plt.show()
```

Figure 5:visualizing and printing specs

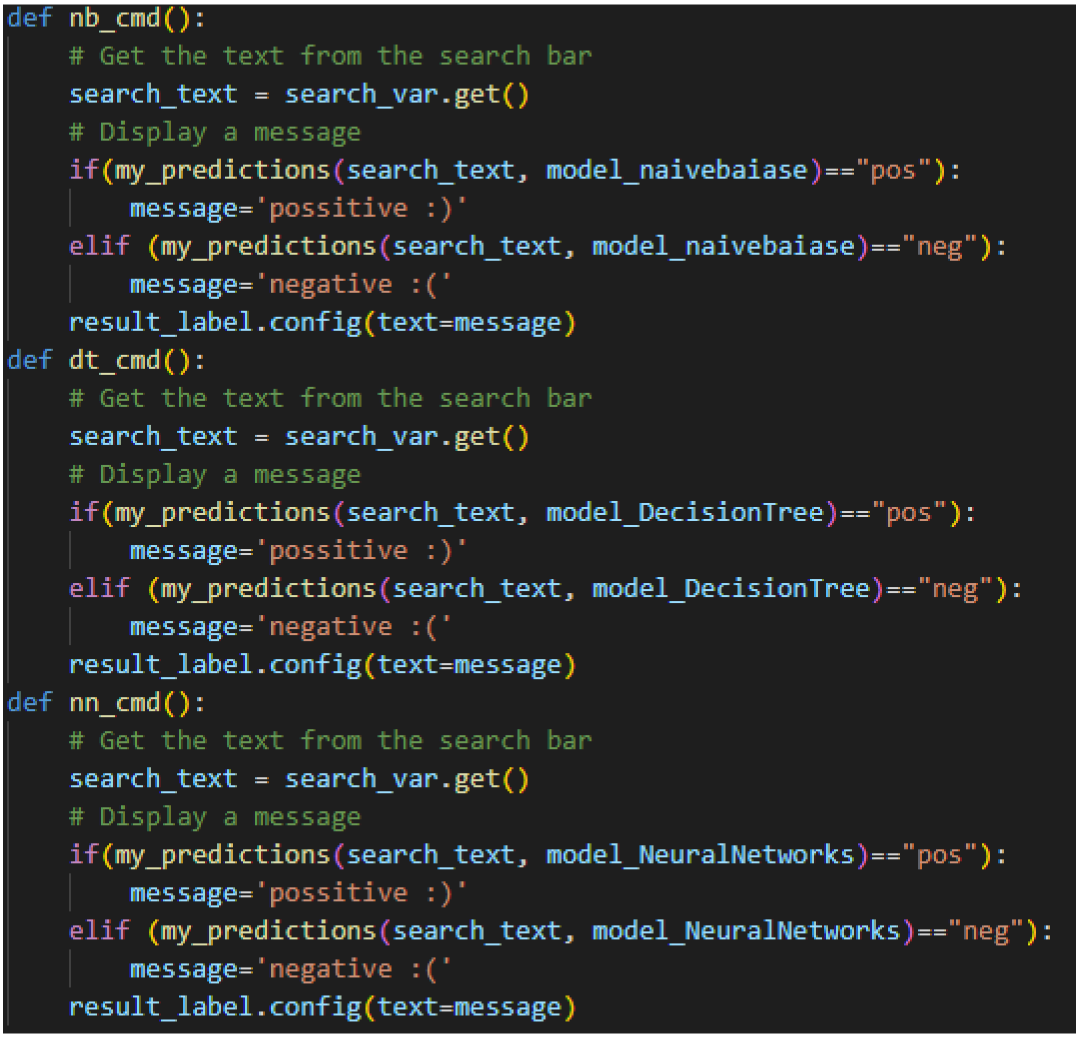
We make a search bar to put a statement to predict and three buttons to select one of the classifiers and we put them in a grid and center it to get the best look and we colored the GUI for better look for the user:



```
# Create the main window
root = tk.Tk()
root.title("predect the emotion ai")
root.geometry("400x400")
root.config(bg='#2c3e50')

# Create a search bar
search_var = tk.StringVar()
search_entry = tk.Entry(root, textvariable=search_var, bg='white')
search_entry.grid(row=1, column=0, padx=10, pady=10, sticky='ew')

label = tk.Label(root, text="wellcome to our emotion predection ai !!", bg='#2c3e50', fg='yellow',font=("Helvetica", 15))
label.grid(row=0, column=0, padx=10, pady=10, sticky='ew')

# Create a label to display the result
result_label = tk.Label(root, text=" ", bg='#2c3e50', fg='yellow',font=("Helvetica", 15))
result_label.grid(row=2, column=0, padx=10, pady=10, sticky='ew')

# Create a naivebiase button
nb_button = tk.Button(root, text="predict using naive bayez", command=nb_cmd, bg='#f1c40f', activebackground='#f7dc6f')
nb_button.grid(row=3, column=0, padx=10, pady=10, sticky='ew')
# Create a desicion tree button
dt_button = tk.Button(root, text="predict using desicion tree", command=dt_cmd, bg='#f1c40f', activebackground='#f7dc6f')
dt_button.grid(row=4, column=0, padx=10, pady=10, sticky='ew')

# Create a neural network button
dt_button = tk.Button(root, text="predict using neural network", command=nn_cmd, bg='#f1c40f', activebackground='#f7dc6f')
dt_button.grid(row=5, column=0, padx=10, pady=10, sticky='ew')
```

Figure 6:gui

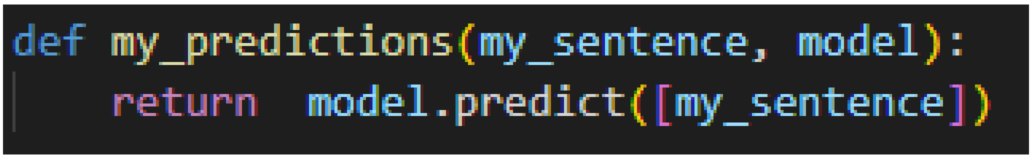
When we press one of the buttons we call one of these three methods that use the selected classifier and predict the outcome :



```
def nb_cmd():
    # Get the text from the search bar
    search_text = search_var.get()
    # Display a message
    if(my_predictions(search_text, model_naivebaiase)=="pos"):
        message='possitive :)'
    elif (my_predictions(search_text, model_naivebaiase)=="neg"):
        message='negative :('
    result_label.config(text=message)
def dt_cmd():
    # Get the text from the search bar
    search_text = search_var.get()
    # Display a message
    if(my_predictions(search_text, model_DecisionTree)=="pos"):
        message='possitive :)'
    elif (my_predictions(search_text, model_DecisionTree)=="neg"):
        message='negative :('
    result_label.config(text=message)
def nn_cmd():
    # Get the text from the search bar
    search_text = search_var.get()
    # Display a message
    if(my_predictions(search_text, model_NeuralNetworks)=="pos"):
        message='possitive :)'
    elif (my_predictions(search_text, model_NeuralNetworks)=="neg"):
        message='negative :('
    result_label.config(text=message)
```

Figure 7:buttons when pressed

We use this method to predict :

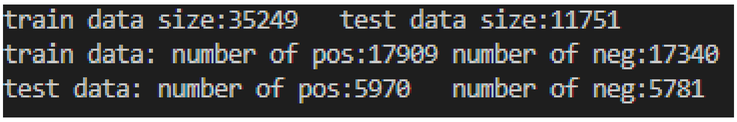


```
def my_predictions(my_sentence, model):
    return  model.predict([my_sentence])
```

Figure 8:prediction method

Program run:

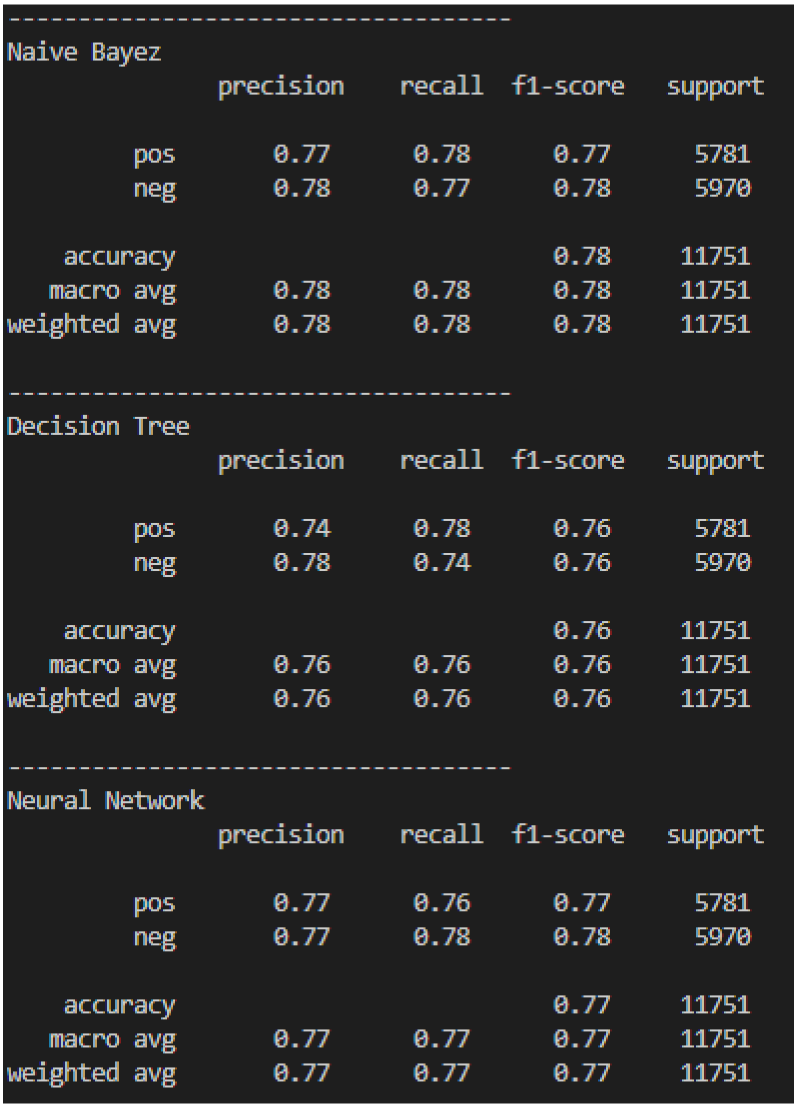
When we run the program firstly it prints the data size and the count of the data used for the train and the count of the data used for test:



```
train data size:35249   test data size:11751
train data: number of pos:17909 number of neg:17340
test data: number of pos:5970   number of neg:5781
```

Figure 9:the data specs

For all classifiers this is the report :



```
------------------------------------
Naive Bayez
              precision    recall  f1-score   support

         pos       0.77      0.78      0.77      5781
         neg       0.78      0.77      0.78      5970

    accuracy                           0.78     11751
   macro avg       0.78      0.78      0.78     11751
weighted avg       0.78      0.78      0.78     11751

------------------------------------
Decision Tree
              precision    recall  f1-score   support

         pos       0.74      0.78      0.76      5781
         neg       0.78      0.74      0.76      5970

    accuracy                           0.76     11751
   macro avg       0.76      0.76      0.76     11751
weighted avg       0.76      0.76      0.76     11751

------------------------------------
Neural Network
              precision    recall  f1-score   support

         pos       0.77      0.76      0.77      5781
         neg       0.77      0.78      0.78      5970

    accuracy                           0.77     11751
   macro avg       0.77      0.77      0.77     11751
weighted avg       0.77      0.77      0.77     11751
```

Figure 10:classifiers report

For the naïve bayez this is the visualize for the data:

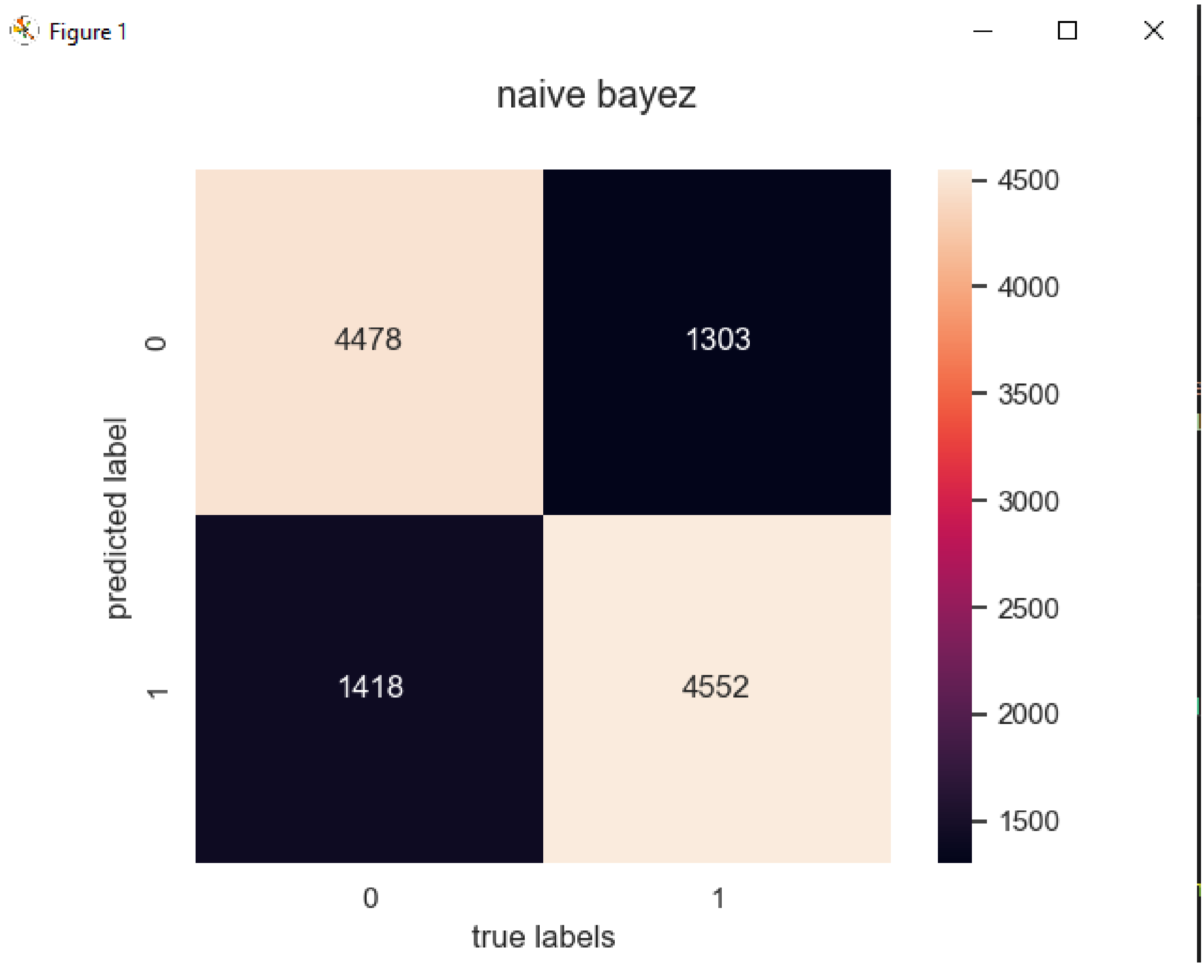


Figure 11:naive bayez

For the decision tree this is the visualize for the data:

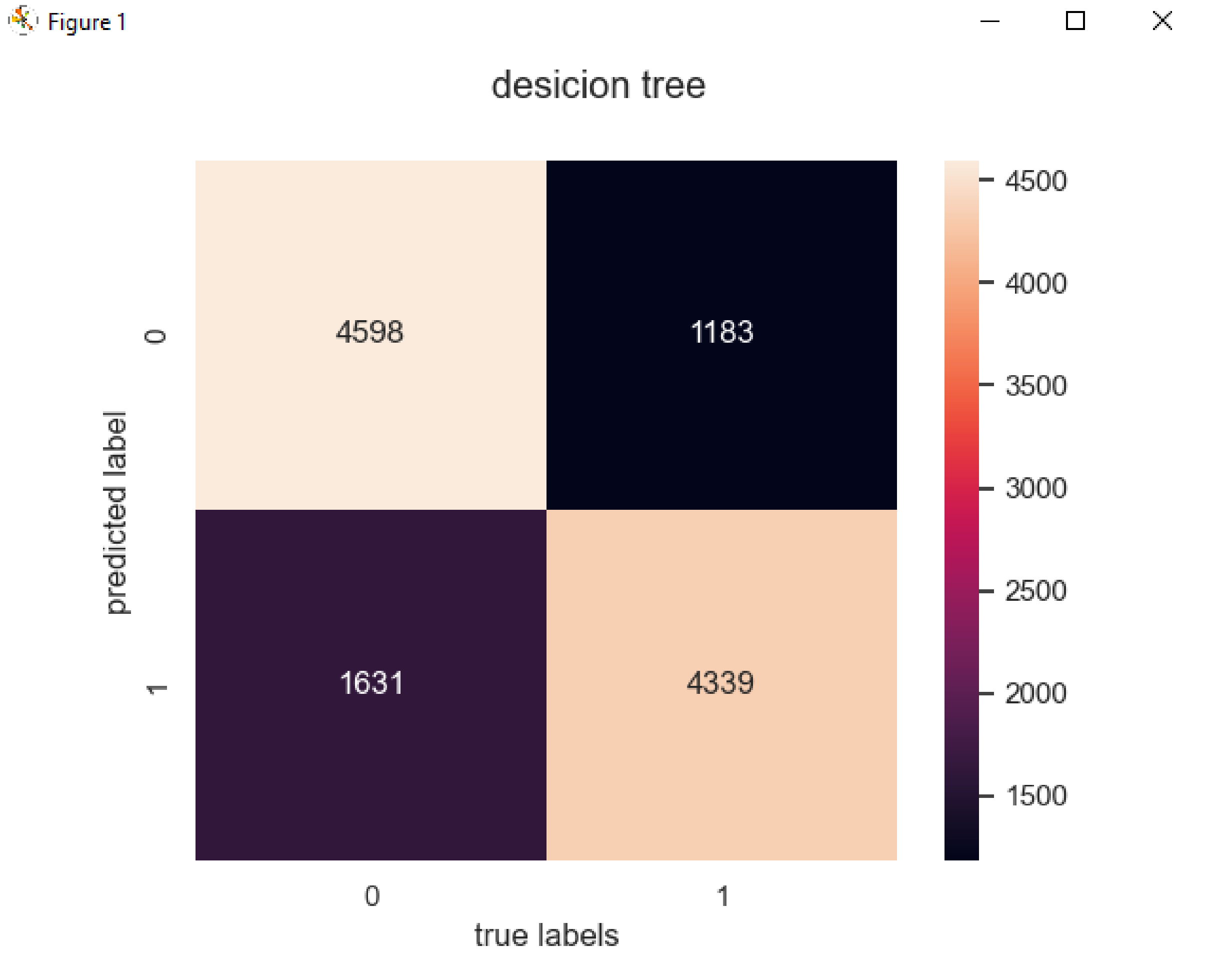


Figure 12:decision tree

For the neural network this is the visualize for the data:

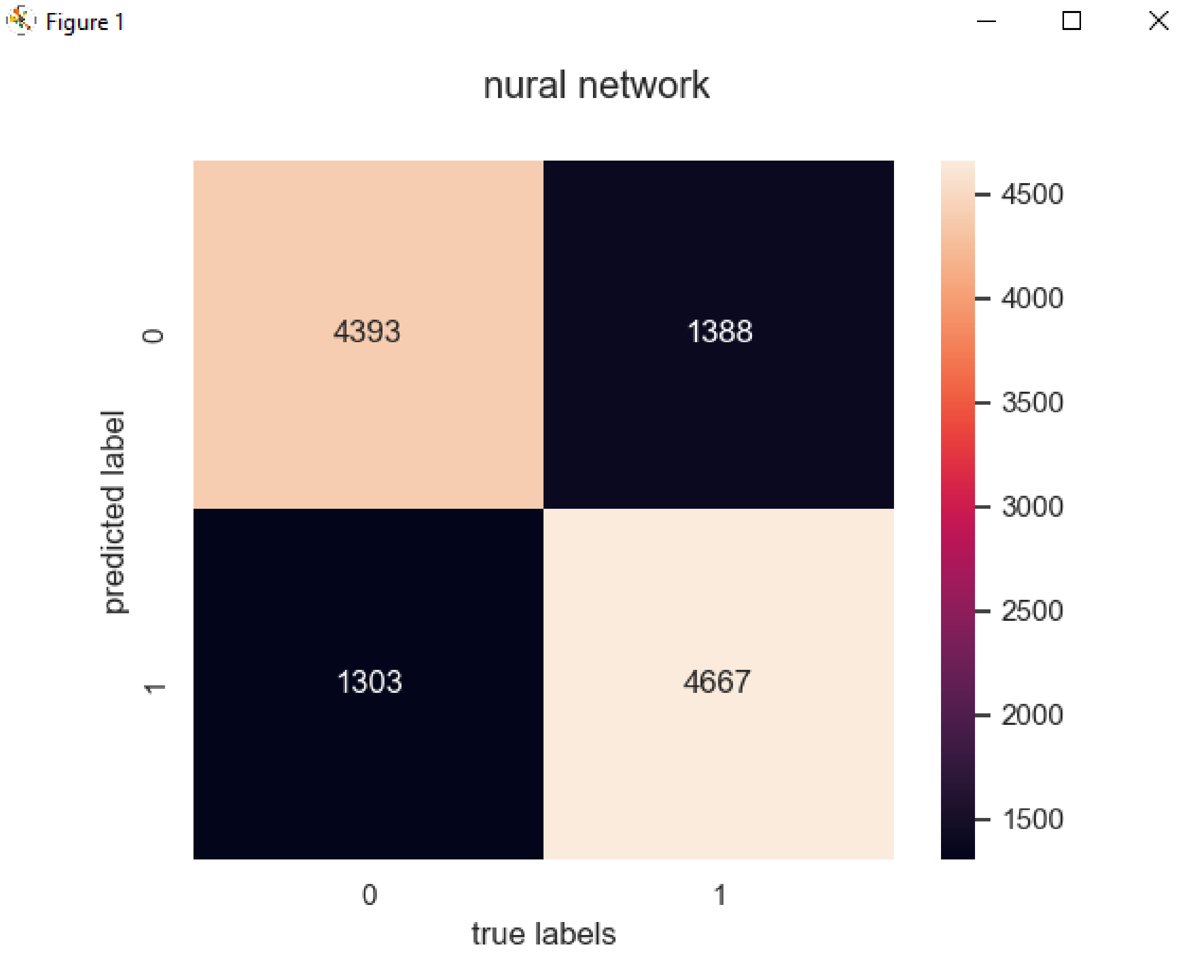
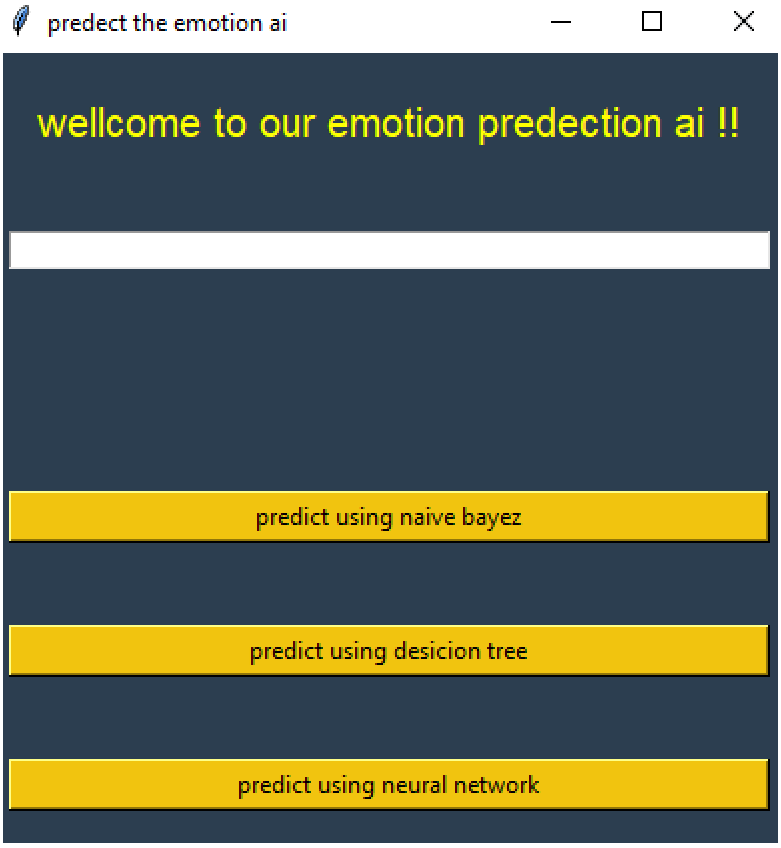


Figure 13:neural network

This is the GUI for our program after training and testing:



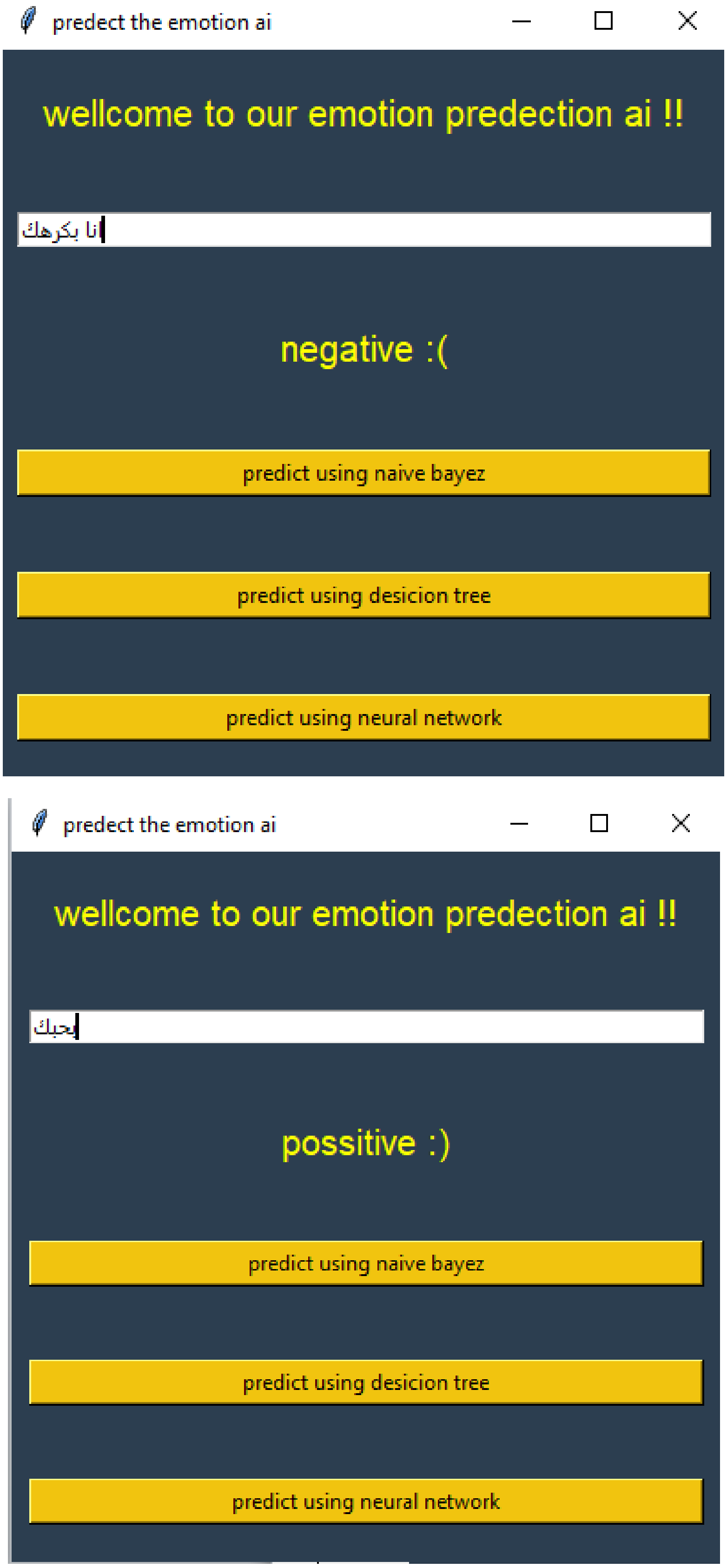
predect the emotion ai

wellcome to our emotion predection ai !!

predict using naive bayez
predict using desicion tree
predict using neural network

Figure 14:GUI

We tried the GUI using positive statements and negative and the result for all classifiers were true



predect the emotion ai

wellcome to our emotion predection ai !!

انا بكرهك

negative :(

predict using naive bayez
predict using desicion tree
predict using neural network

predect the emotion ai

wellcome to our emotion predection ai !!

بحبك

possitive :)

predict using naive bayez
predict using desicion tree
predict using neural network

Figure 15:testing the program for the user