Final-Project report

TITLE: Develop and operate the message-filter bot for discord and investigate the user evaluation

Student: 202110647 Junhyung-kim

Department: AI•BigData

-Contents-

Abstract

During the summer semester, I worked at a community server as the manager, for the 3 months.

After I became the manager of the Discord community server, I soon faced a lot of accident cases and problems, most of which were about ‘profanity’ from the conflict with the users.

In the other case, they send harmful images and this has a side effect on the other users' mental health. However, there are multiple constraints to the environment and, my schedule.

At this moment, I realized there was a need Auto management system instead of me. This is the purpose of why I developed the message filter bot for management and monitoring in the community server as task automation.

I. Introduction

-Discord: What is the discord-bot

Discord is the group chat platform for global networking[1]. This a community server as a group chat app for Games, and group activity. In Discord, there are many Discord bots to support the users. The company provides the Discord development tool at the portal of ‘discord dev portal’ (https://discord.com/developers/applications)

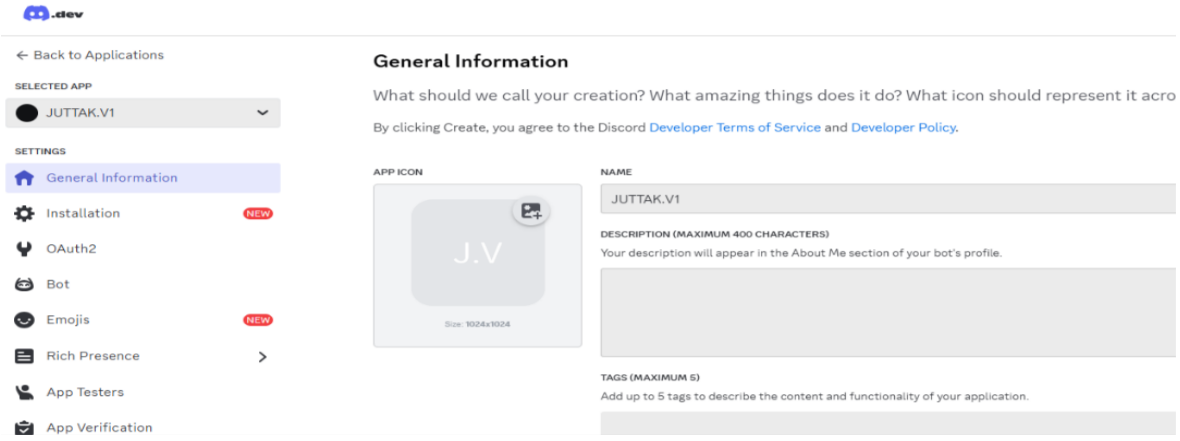
-Setting process-

Firstly, connect to the Discord Developers portal, and get the token for your Discord bot.

In this process, get the ‘token’ to connect to the server, and your bots ‘token’ is instead of login of your accounts.

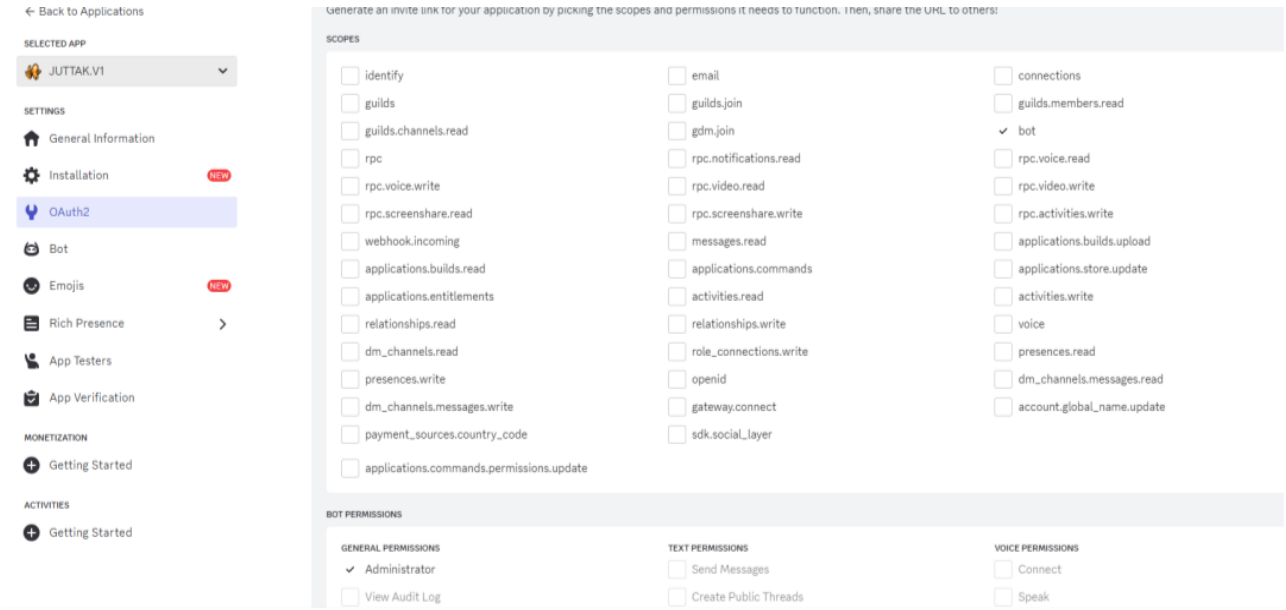
---

[1] **(Anirudh Verma1, Shashikant Tyagi1, Gauri Mathur2, 2016, International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), india, pg: 533)**

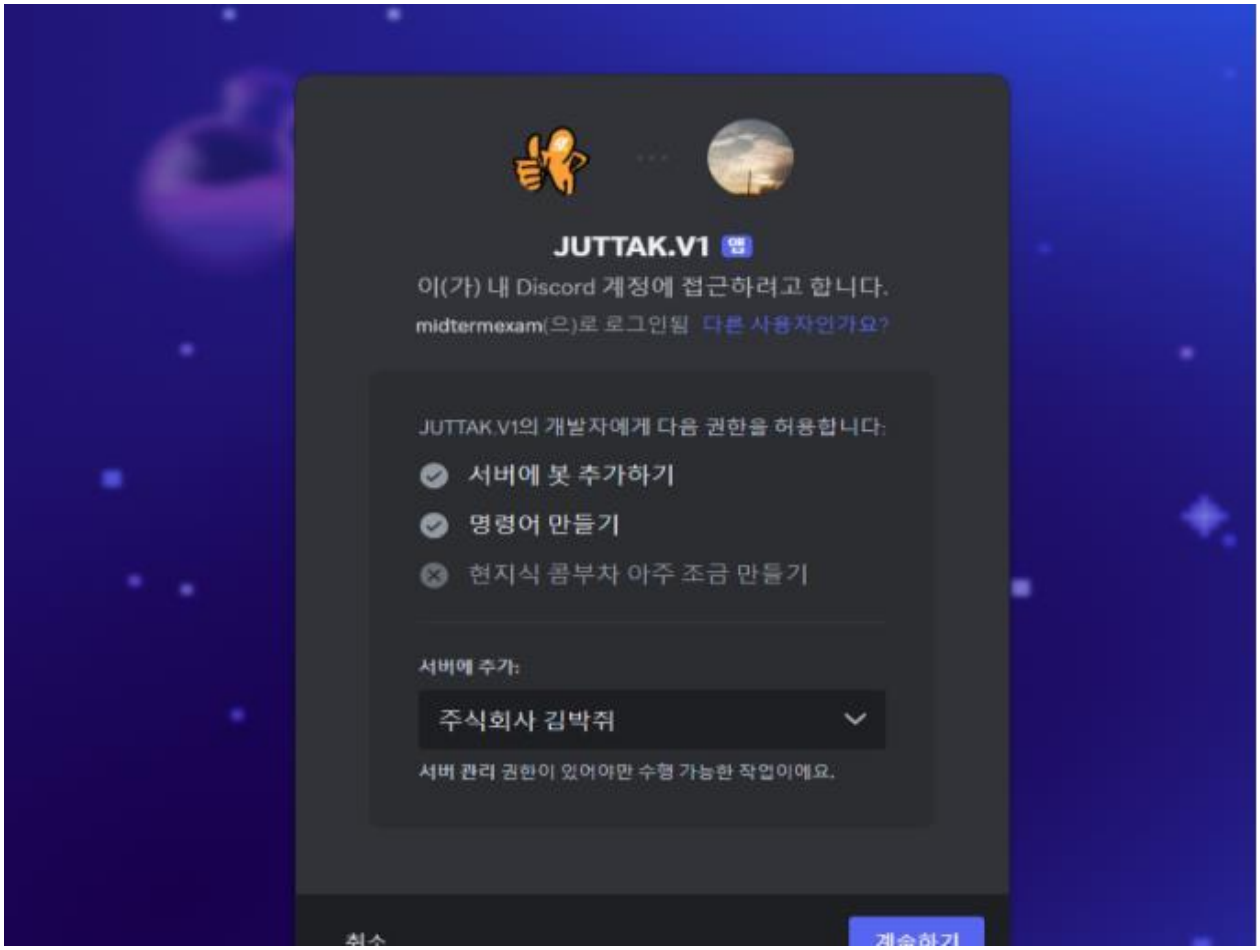


This means the token passes the secure step, so you have to keep secret this if you expose it once then must reissue your token.

Secondly, give your bots authority (permission) for task. If your bot does a work of something on server platform then should have permission. This is simply adding the functions to your bot.



Invite your discord bot to your server using the token, server ID.



-Function: python libraries

The setting discord bot is complete as the preprocess, now we add to function for your bots.

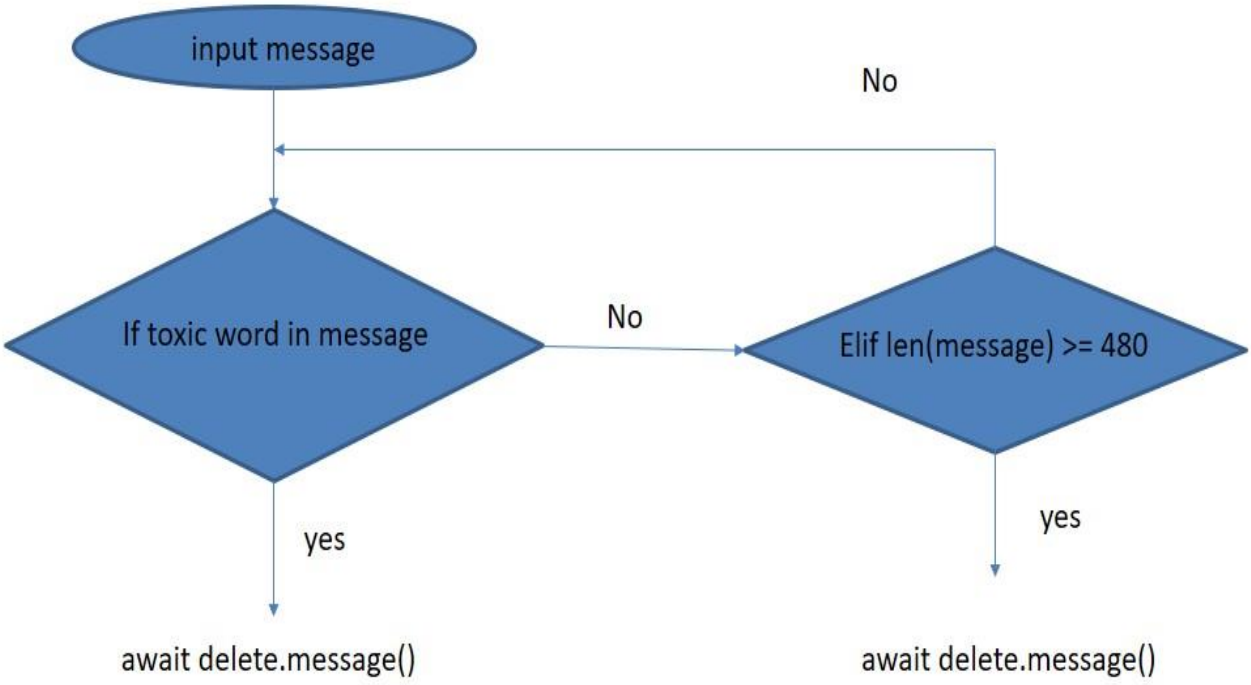
Among the Python libraries, there exists for developing discord bot. These libraries provide the functions for when the discord bot is in the task. This Python library name is ‘discord.py’[2]

I need a function to detect the image and Keywords as decision-by conditions. Then, input data is necessary, to compare the input data from the user message. So using ‘PIL’ for input image data as the target data. The ML model detects the ‘harmful contents’ by comparing the pixel value data if the target image data and the user sends image data have the same pixel value, then the bot deletes the message. Next, show the flow chat for the filter bot model.

[2] (MagzoubZaed, toxic comment classicfication in discord, 2020, pg:5)

-Flow chart-

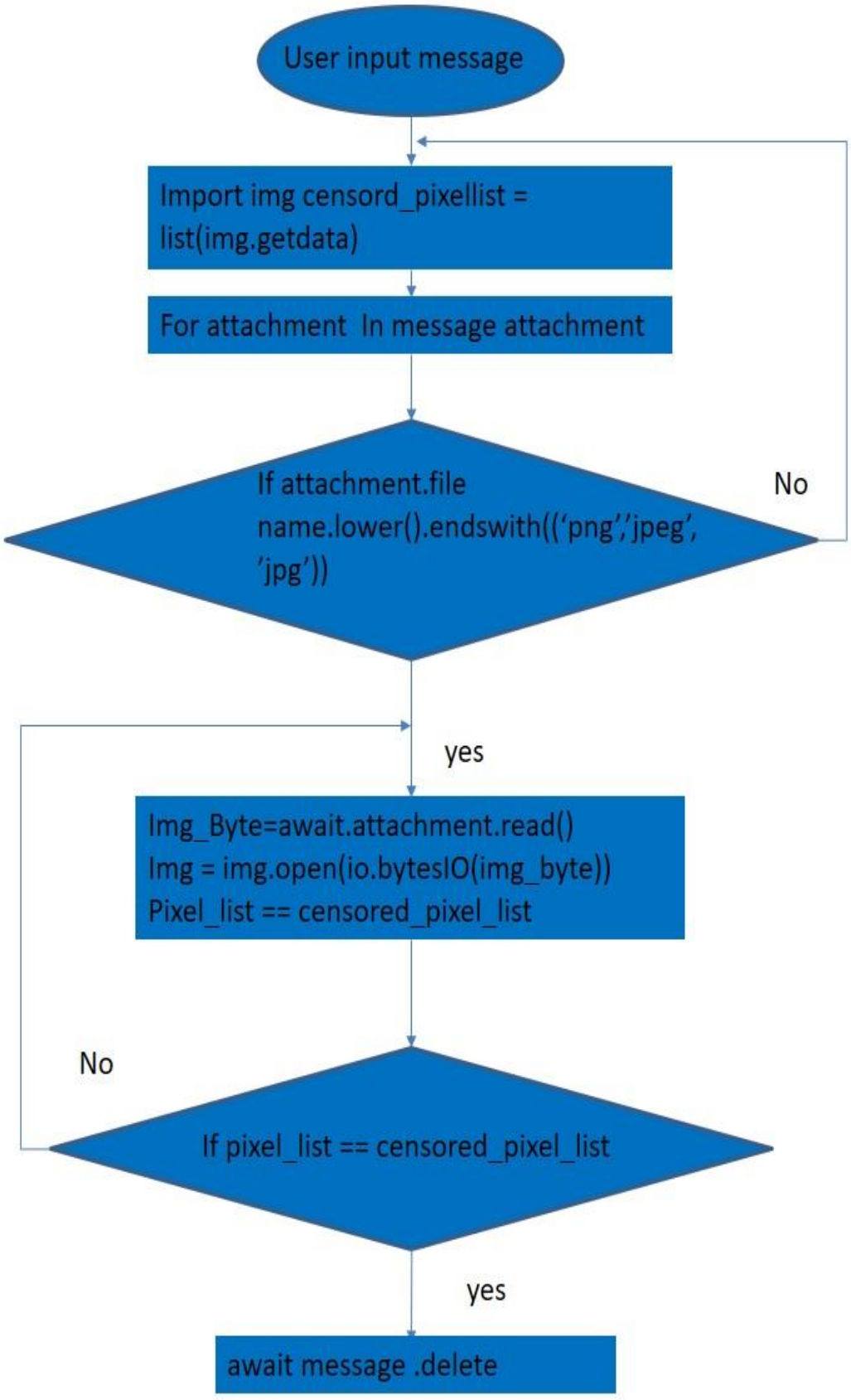
This is the follow-chart (diagram) of the chat filter



If the user sends a message (input message), the 'if-statement' compares the user message with the censored keyword in the array. If is censored keyword exists, 'await delete.message()' function will be running. If there is not exist any censored keyword, then the classification length of the message. If the message is over the 480 types then delete the message.

This is the flowchart (diagram) of the image filter, 'import_img' censored_pixellist' as the input data, I have already uploaded censored target image data on the 'Jupyter Notebook'.

And add the code 'if attachment.filename.lower().ends with (('png',' jpeg',' jpg'))' for consideration when the user sends the image file as the '.png', '.jpeg',' jpg' also, at the decision part as I mentioned before, compare pixel value of the target image data with user input data.


User input message
Import img censord_pixellist = list(img.getdata)
For attachment In message attachment
If attachment.file name.lower().endswith(('png','jpeg', 'jpg'))
No
yes
Img_Byte=await.attachment.read()
Img = img.open(io.bytesIO(img_byte))
Pixel_list == censored_pixel_list
No
If pixel_list == censored_pixel_list
yes
await message .delete

2. Related works

-Develop the filtering bot, and operate in the field (server)-

In this section, I will represent the flow chart as each Python code. Firstly, convert the chat filter system of the censored bot. This uses an array to set the multiple censored keywords and these array variables' names are set as the 'hams' and 'toxic' points of the elements.

```
@bot.event
async def on_message(message):
    message_content = message.content
    harms = ["censord1", "censored2", "censored3", "censored4", "censored5", "censored6", "censored7", "censored8", "censored9"]

    if any(toxic in message_content for toxic in harms):
        await message.channel.send("please note that, this is public place")
        await message.delete()
    elif len(message_content) >= 480:
        await message.delete()
```

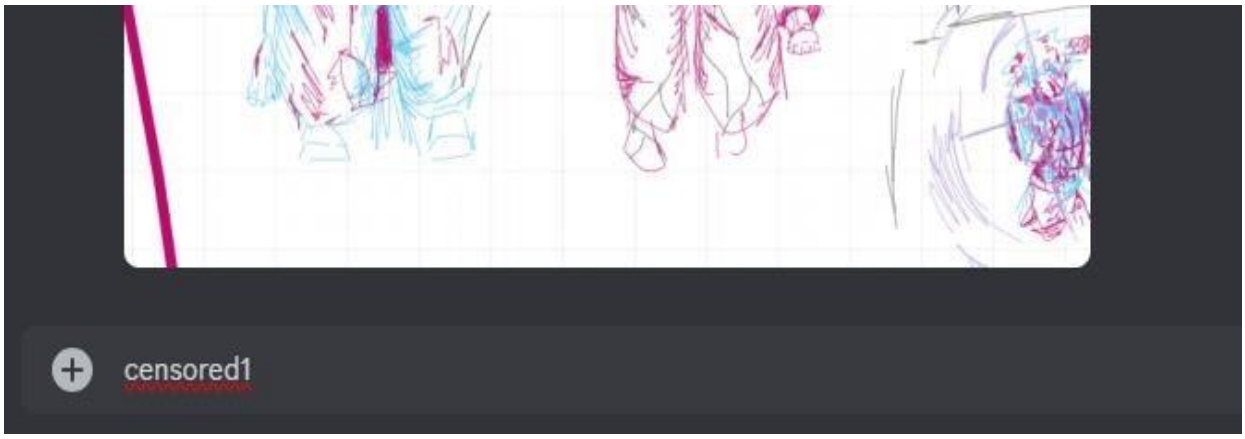
Secondly, convert the image filter system of the censored bot. This uses the PIL library to get image data and compare the pixel value of target image data and user Input(send) data. Target image data should uploaded to Jupiter Notebook.

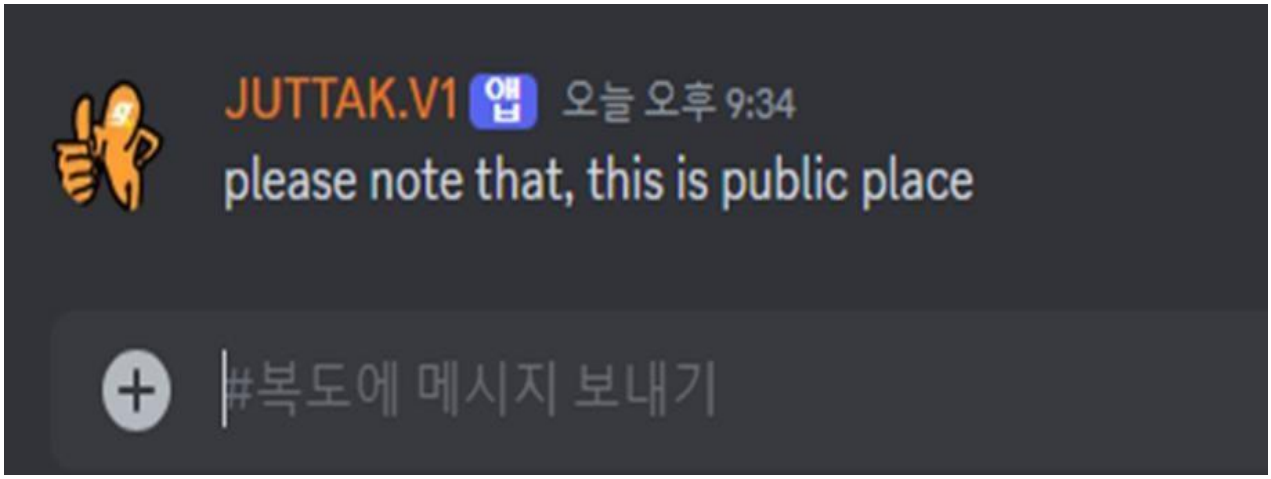
```
import discord
from discord.ext import commands
import asyncio
import nest_asyncio
import numpy as np
from PIL import Image
import io

img = Image.open("Censored.png")
Censored_pixel_list = list(img.getdata())

np.savetxt("pixel_type_data.txt", Censored_pixel_list, fmt='%d', delimiter=" ")
```

```
    for attachment in message.attachments:
        if attachment.filename.lower().endswith(('png', 'jpg', 'jpeg')):
            image_bytes = await attachment.read()
            img = Image.open(io.BytesIO(image_bytes))
            pixel_list = list(img.getdata())

            if pixel_list == Censored_pixel_list:
                await message.channel.send("The harmful contents is detected")
                await message.delete()
                await bot.process_commands(message)
```

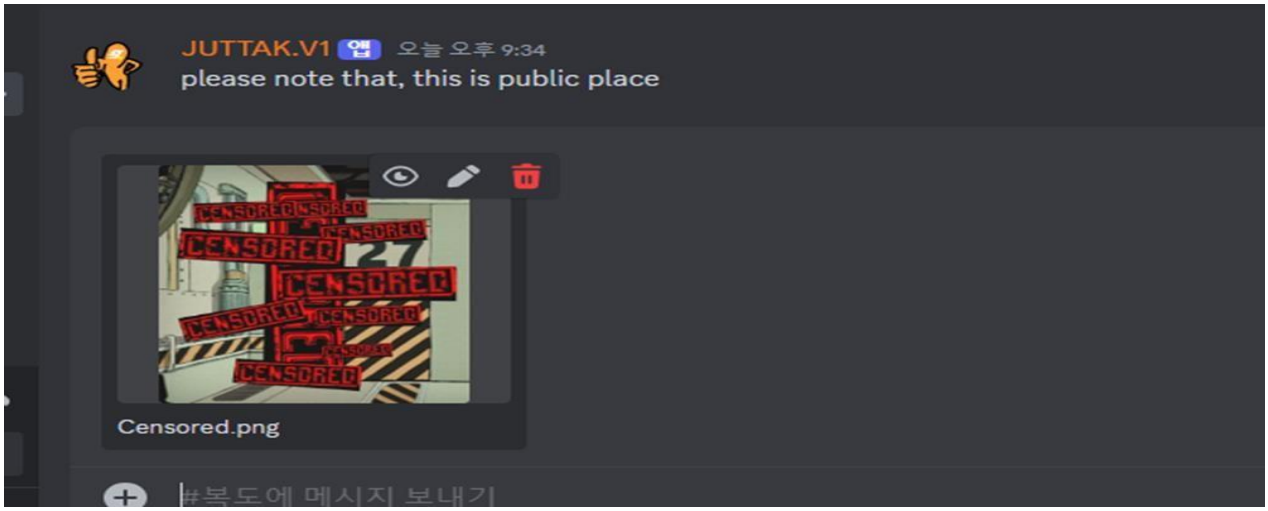
Now demonstrate that each code is working. Firstly, demonstrate the chat filter.

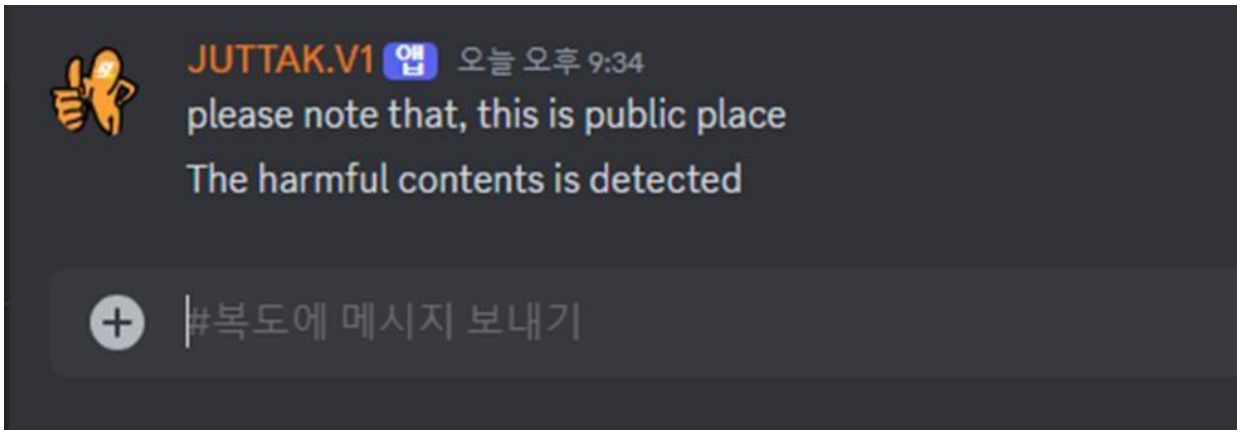
At the discord server, I run the bot, after typing the dummy censored keyword in the harms array elements and sending a message. Then filter bot deletes the message and sends an alert message.





Secondly, now I test about image filter system. Before I contained the real data, uploaded the Dummy data for the image. When I attach the same pixel value of the target image then, bot compare two of value and deletes the attachment image from message.





-Extraction of survey data from users about satisfaction with the filter bot-

Now I need to get evaluation data from the users, about the satisfaction of the censored system. Most of the questions contain numerical data such as the score as 1-100. Firstly, I make the Google form for a survey from users. This survey will become to dataset for the model performance Evaluation, and find the points of improvement. This question is about the performance of the chat filter and Image filter, and the current satisfaction of this process. In addition, I encouraged the limitation of reflecting the user's opinion by adding the answer sheet.

Evaluation for Censored bot

represent your opinion for Improvements

What's your User name ?

단답형 텍스트

How accurate is the word filtering level of that bot? Please grade it (from 1 to 100)

단답형 텍스트

How accurate is the photo(jpeg, png, etc) filtering level of that bot? Please grade it (from 1 to 100)

단답형 텍스트

(Effectiveness of Auto-Delete Feature) how useful the auto - delete feature is for specific keywords. (from 1 to 100)

단답형 텍스트

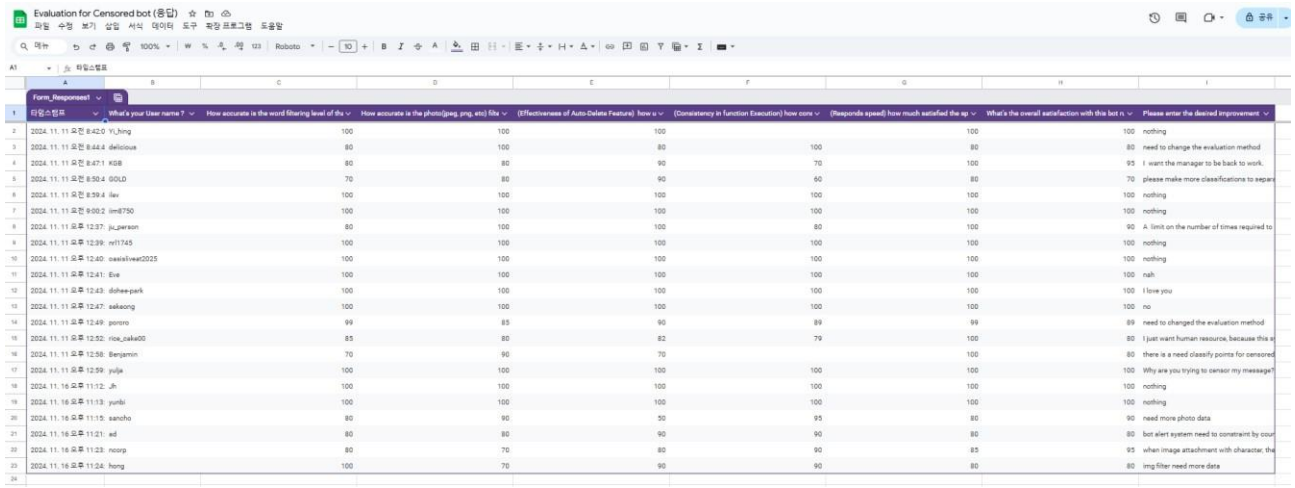
(Consistency in function Execution) how consistency the bot performs its functions? (from 1 to 100)

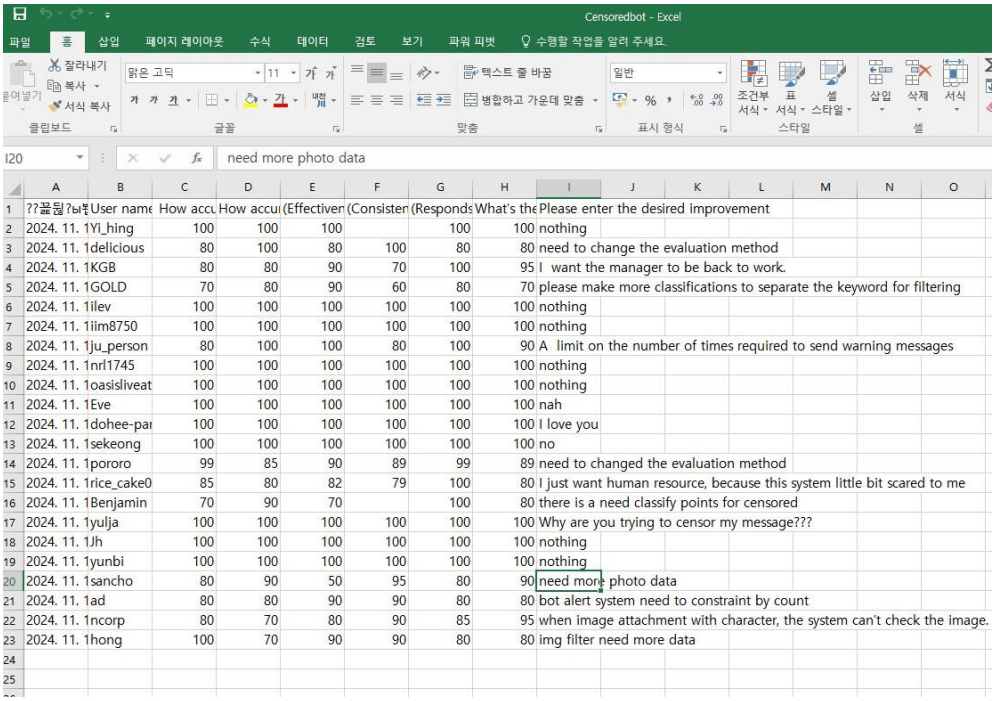
단답형 텍스트

(Responds speed) how much satisfied the speeds which the bot detects and deletes message or images? (from 1 to 100)

단답형 텍스트

Secondly, after collecting the data from users, now I extracted the dataset as the ‘CSV’[3] Connect the Google Forms and connect with the spreadsheets, download the ‘Excel chart’ as a CSV file’, and upload it to Jupyter Notebook. Now we are to extract the dataset.





	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	??꿀딤?뉘붓	User name	How accu	How accu	(Effectiven	(Consisten	(Responds	What's the	Please enter the desired improvement						
2	2024. 11. 1	Yi_hing	100	100	100		100	100	nothing						
3	2024. 11. 1	delicious	80	100	80	100	80	80	need to change the evaluation method						
4	2024. 11. 1	KGB	80	80	90	70	100	95	I want the manager to be back to work.						
5	2024. 11. 1	GOLD	70	80	90	60	80	70	please make more classifications to separate the keyword for filtering						
6	2024. 11. 1	ilev	100	100	100	100	100	100	nothing						
7	2024. 11. 1	iim8750	100	100	100	100	100	100	nothing						
8	2024. 11. 1	ju_person	80	100	100	80	100	90	A limit on the number of times required to send warning messages						
9	2024. 11. 1	nrl1745	100	100	100	100	100	100	nothing						
10	2024. 11. 1	oasisliveat	100	100	100	100	100	100	nothing						
11	2024. 11. 1	Eve	100	100	100	100	100	100	nah						
12	2024. 11. 1	dohee-par	100	100	100	100	100	100	I love you						
13	2024. 11. 1	sekeong	100	100	100	100	100	100	no						
14	2024. 11. 1	pororo	99	85	90	89	99	89	need to changed the evaluation method						
15	2024. 11. 1	rice_cake0	85	80	82	79	100	80	I just want human resource, because this system little bit scared to me						
16	2024. 11. 1	Benjamin	70	90	70		100	80	there is a need classify points for censored						
17	2024. 11. 1	yulja	100	100	100	100	100	100	Why are you trying to censor my message???						
18	2024. 11. 1	Jh	100	100	100	100	100	100	nothing						
19	2024. 11. 1	yunbi	100	100	100	100	100	100	nothing						
20	2024. 11. 1	sancho	80	90	50	95	80	90	need more photo data						
21	2024. 11. 1	ad	80	80	90	90	80	80	bot alert system need to constraint by count						
22	2024. 11. 1	ncorp	80	70	80	90	85	95	when image attachment with character, the system can't check the image.						
23	2024. 11. 1	hong	100	70	90	90	80	80	img filter need more data						
24															
25															

[3] These of data called ‘structured data’ for example, ‘CSV’, ‘Excel’, ‘Data base’ However as like photo data (censored image) is called ‘unstructured data’.

The next section will be data preprocessing with, data reduction.

This process is for the final proposed method as the linear regression. In this section, there must

No overfitting problem.

3. Proposed method

-Represent survey data as PCC-

In the data preprocessing section, need to find the missing value, fill it with numerical data '0', and drop the column of non-numerical data or no-meaning data, for example, there is time series data and user opinion. Because these data can't Quantified for 'PCA'[4].

```
[1]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sn

     data = pd.read_csv('Censoredbot.csv')
     data.sample(10)
```

[1]:

	타임스탬프	What's your User name ?	How accurate is the word filtering level of that bot? Please grade it (from 1 to 100)	How accurate is the photo(jpeg, png, etc) filtering level of that bot? Please grade it (from 1 to 100)	(Effectiveness of Auto-Delete Feature) how useful the auto - delete feature is for specific keywords. (from 1 to 100)	(Consistency in function Execution) how consistency the bot performs its functions? (from 1 to 100)	(Responds speed) how much satisfied the speeds which the bot detects and deletes message or images? (from 1 to 100)	What's the overall satisfaction with this bot running? (from 1 to 100)	Please enter the desired improvement
15	2024. 11. 11 오후 12:59:38	yulja	100	100	100	100.0	100	100	Why are you trying to censor my message???
0	2024. 11. 11 오전 8:42:05	Yi_hing	100	100	100	NaN	100	100	nothing
16	2024. 11. 16 오후 11:12:14	Jh	100	100	100	100.0	100	100	nothing
8	2024. 11. 11 오후 12:40:25	oasisliveat2025	100	100	100	100.0	100	100	nothing
7	2024. 11. 11 오후 12:39:09	nrl1745	100	100	100	100.0	100	100	nothing
4	2024. 11. 11 오전 8:59:45	ilev	100	100	100	100.0	100	100	nothing
3	2024. 11. 11 오전 8:50:44	GOLD	70	80	90	60.0	80	70	please make more classifications to separate t...
	2024. 11.								bot alert system need to

[4] PCA (Principal Component Analysis) is meaning that technique for feature extraction that combines the input variables in a specific way and drops the least important variables from a dataset. This is need to solve the overfitting problem.

[2]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22 entries, 0 to 21
Data columns (total 9 columns):
 #   Column                                                                                                                                  Non-Null Count  Dtype
---  ------                                                                                                                                  --------------  -----
 0   타임스탬프                                                                                                                                  22 non-null     object
 1   What's your User name ?                                                                                                                 22 non-null     object
 2    How accurate is the word filtering level of that bot? Please grade it (from 1 to 100)                                                 22 non-null     int64
 3   How accurate is the photo(jpeg, png, etc) filtering level of that bot? Please grade it (from 1 to 100)                                 22 non-null     int64
 4   (Effectiveness of Auto-Delete Feature)  how useful the auto - delete feature is for specific keywords. (from 1 to 100)                 22 non-null     int64
 5   (Consistency in function Execution) how consistency the bot performs its functions? (from 1 to 100)                                    20 non-null     float64
 6   (Responds speed) how much satisfied the speeds which the bot detects and deletes message or images? (from 1 to 100)                    22 non-null     int64
 7   What's the overall satisfaction with this bot running? (from 1 to 100)                                                                 22 non-null     int64
 8   Please enter the desired improvement                                                                                                    22 non-null     object
dtypes: float64(1), int64(5), object(3)
memory usage: 1.7+ KB
```

[3]:
```
data.isna().sum()
```

```
타임스탬프                                                                                                                                  0
What's your User name ?                                                                                                                 0
 How accurate is the word filtering level of that bot? Please grade it (from 1 to 100)                                                 0
How accurate is the photo(jpeg, png, etc) filtering level of that bot? Please grade it (from 1 to 100)                                 0
(Effectiveness of Auto-Delete Feature)  how useful the auto - delete feature is for specific keywords. (from 1 to 100)                 0
(Consistency in function Execution) how consistency the bot performs its functions? (from 1 to 100)                                    2
(Responds speed) how much satisfied the speeds which the bot detects and deletes message or images? (from 1 to 100)                    0
What's the overall satisfaction with this bot running? (from 1 to 100)                                                                 0
Please enter the desired improvement                                                                                                    0
dtype: int64
```

[19]:
```
Y=data['타임스탬프']
X=data
X.drop(['타임스탬프'], axis=1, inplace=True)
X.sample(10)
```

[19]:

	What's your User name ?	How accurate is the word filtering level of that bot? Please grade it (from 1 to 100)	How accurate is the photo(jpeg, png, etc) filtering level of that bot? Please grade it (from 1 to 100)	(Effectiveness of Auto-Delete Feature) how useful the auto - delete feature is for specific keywords. (from 1 to 100)	(Consistency in function Execution) how consistency the bot performs its functions? (from 1 to 100)	(Responds speed) how much satisfied the speeds which the bot detects and deletes message or images? (from 1 to 100)	What's the overall satisfaction with this bot running? (from 1 to 100)	Please enter the desired improvement
19	ad	80	80	90	90.0	80	80	bot alert system need to constraint by count
17	yunbi	100	100	100	100.0	100	100	nothing
14	Benjamin	70	90	70	NaN	100	80	there is a need classify points for censored
2	KGB	80	80	90	70.0	100	95	I want the manager to be back to work.
20	ncorp	80	70	80	90.0	85	95	when image attachment with character, the syst...
7	nrl1745	100	100	100	100.0	100	100	nothing
16	Jh	100	100	100	100.0	100	100	nothing
6	ju_person	80	100	100	80.0	100	90	A limit on the number of times required to se...
0	Yi_hing	100	100	100	NaN	100	100	nothing
1	delicious	80	100	80	100.0	80	80	need to change the evaluation method

```
[23]: Y2=data['Please enter the desired improvement']
new_data=X
new_data.drop(['Please enter the desired improvement'], axis=1, inplace=True)
new_data.sample(10)
```

[23]:

	What's your User name ?	How accurate is the word filtering level of that bot? Please grade it (from 1 to 100)	How accurate is the photo(jpeg, png, etc) filtering level of that bot? Please grade it (from 1 to 100)	(Effectiveness of Auto-Delete Feature) how useful the auto - delete feature is for specific keywords. (from 1 to 100)	(Consistency in function Execution) how consistency the bot performs its functions? (from 1 to 100)	(Responds speed) how much satisfied the speeds which the bot detects and deletes message or images? (from 1 to 100)	What's the overall satisfaction with this bot running? (from 1 to 100)
15	yulja	100	100	100	100.0	100	100
16	Jh	100	100	100	100.0	100	100
19	ad	80	80	90	90.0	80	80
17	yunbi	100	100	100	100.0	100	100
10	dohee-park	100	100	100	100.0	100	100
1	delicious	80	100	80	100.0	80	80
3	GOLD	70	80	90	60.0	80	70
20	ncorp	80	70	80	90.0	85	95
9	Eve	100	100	100	100.0	100	100
11	sekeong	100	100	100	100.0	100	100

6

5 Drop the column of 'timestamp'

6 Drop the column of 'user opinion'

```
[37]: Y3=data["What's your User name ?"]
new_data2=new_data
new_data2.drop(["What's your User name ?"], axis=1, inplace=True)
new_data.sample(10)
```

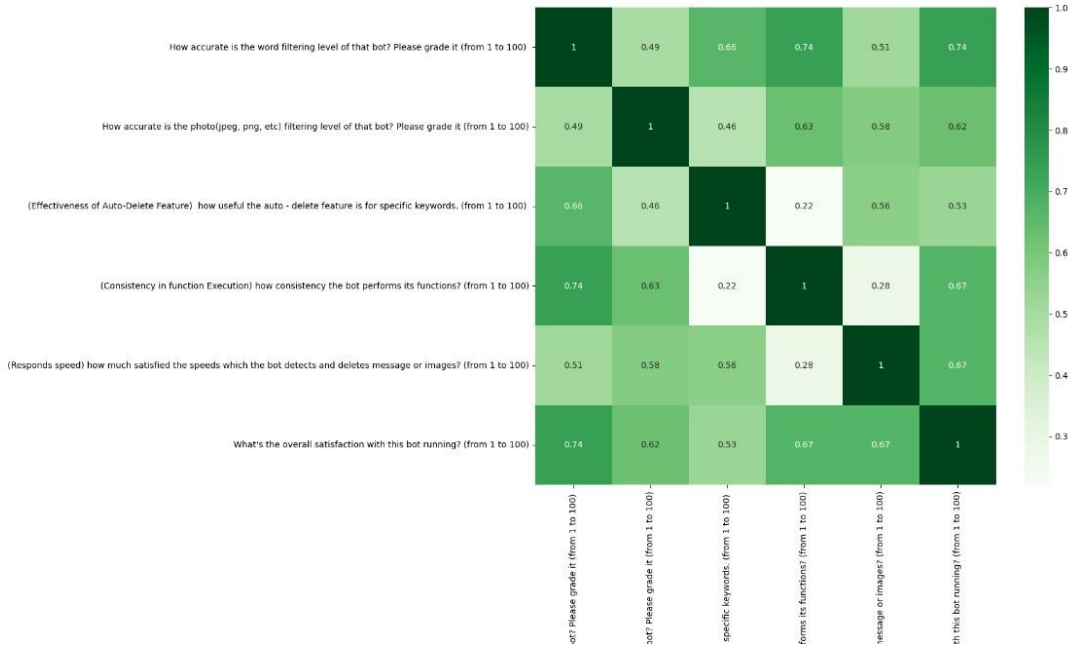
[37]:

	How accurate is the word filtering level of that bot? Please grade it (from 1 to 100)	How accurate is the photo(jpeg, png, etc) filtering level of that bot? Please grade it (from 1 to 100)	(Effectiveness of Auto-Delete Feature) how useful the auto - delete feature is for specific keywords. (from 1 to 100)	(Consistency in function Execution) how consistency the bot performs its functions? (from 1 to 100)	(Responds speed) how much satisfied the speeds which the bot detects and deletes message or images? (from 1 to 100)	What's the overall satisfaction with this bot running? (from 1 to 100)
7	100	100	100	100.0	100	100
0	100	100	100	NaN	100	100
15	100	100	100	100.0	100	100
4	100	100	100	100.0	100	100
12	99	85	90	89.0	99	89
8	100	100	100	100.0	100	100
3	70	80	90	60.0	80	70
17	100	100	100	100.0	100	100
20	80	70	80	90.0	85	95
14	70	90	70	NaN	100	80

7

Before getting the standard deviation value and, the standard scaler, show the PCC[8] of reduced data. This is to find the correction with each column.

```
[39]: data=new_data2
      plt.figure(figsize=(12,10))
      cor=data.corr()
      sn.heatmap(cor, annot = True, cmap=plt.cm.Greens)
      plt.show()
```



[7] Drop the column of the name.

[8] PCC (Person Correlation Coefficient), It finds the direction and strength of a relationship (r) between two variables in a dataset. The range of the possible results of this PCC is -1 to 1, where: 1) 0 indicates no correlation. 2) 1 indicates a strong positive correlation. 3) -1 indicates a strong negative correlation.

The Reason for adding the correlation section is to check each feature evaluation (subdataset), and whole evaluation (overall satisfaction) for the similarity of each sub-dataset to see if there are any out-of-survey results. In addition, this section helps find the improvement points of performance. For example, there is an evaluation of a 'filter system with certain evaluation data' or an 'image data with an overall evaluation.' However, we don't know the Accuracy of this PCC model. So we need to compare two correlation models.

1) Compare the PCC model with a correlation of the "Word-filtering VS overall satisfaction"

Mathematical proofed: Correlation of word filtering & overall evaluation

X=Word-filtering

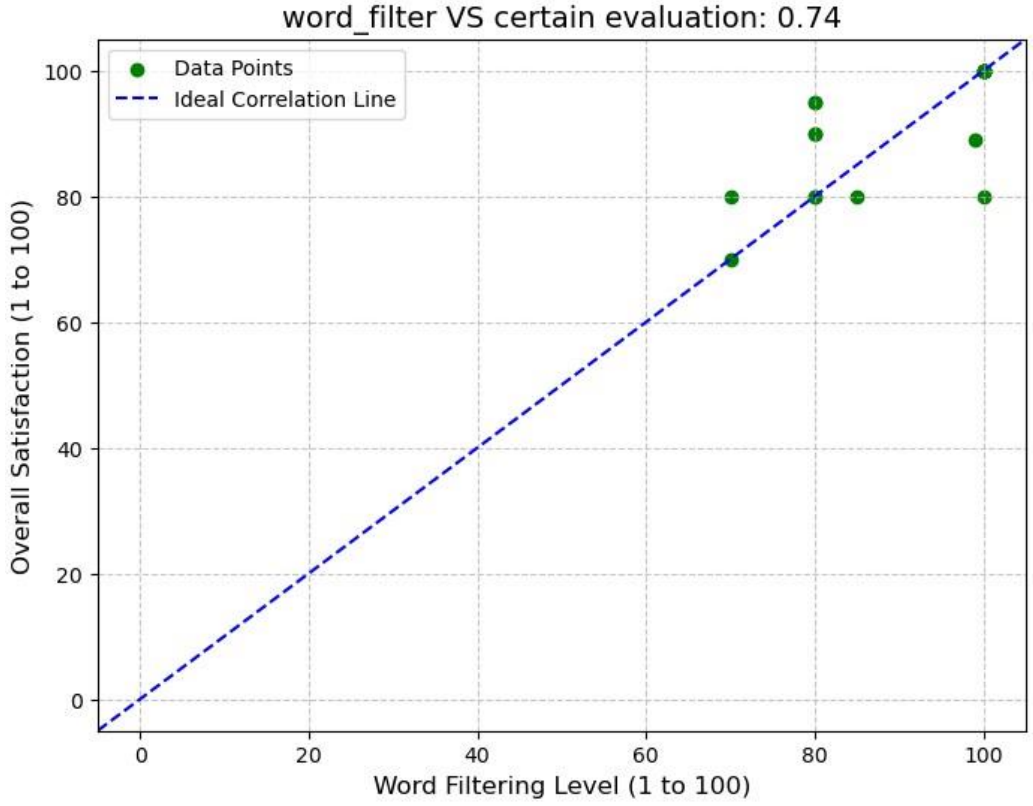
Y=Overall-satisfaction

$$R(X,Y) \quad \frac{Cov(X,Y)}{\sigma X \sigma Y} \quad \frac{\frac{1}{22}\sum_{i=1}^{23}\{(X-mean\ of\ X)\times(Y-mean\ of\ Y)\}}{\sqrt{\frac{1}{22}\sum_{i=1}^{23}(X-mean\ of\ X)}\times\frac{1}{22}\sum_{i=1}^{23}(Y-mean\ of\ Y)} = = 22$$

$$=\frac{\frac{1}{22}(2004-91.09)\times\frac{1}{22}(2025-92.23)}{\sqrt{\frac{1}{22}(2004-91.09)}\times\sqrt{\frac{1}{22}(2025-92.23)}} = \frac{76.71}{11.04\times9.40}=0.74$$

∴The correlation between the word filter and overall evaluations is positive (0.74)

```
import matplotlib.pyplot as plt

word_filtering= np.array([100,80,80,70,100,100,80,100,100,100,100,100,99,85,70,100,100,100,80,80,80,100])
overall_satisfaction= np.array([100,80,95,70,100,100,90,100,100,100,100,100,89,80,80,100,100,100,90,80,95,80])

correlation = np.corrcoef(word_filtering, overall_satisfaction)[0, 1]

plt.figure(figsize=(8, 6)),95
plt.scatter(word_filtering, overall_satisfaction, color='green', label='Data Points')
plt.title(f'word_filter VS certain evaluation: {correlation:.2f}', fontsize=14)
plt.xlabel('Word Filtering Level (1 to 100)', fontsize=12)
plt.ylabel('Overall Satisfaction (1 to 100)', fontsize=12)
plt.axline((0, 0), slope=1, color='blue', linestyle='--', label='Ideal Correlation Line')
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend(fontsize=10)
plt.show()
```



The correlation of the word filter VS overall evaluations is ‘Positive’

∴PCC model of the ‘word-filtering VS overall evaluation” is correct.

2) compare the PCC model with the Correlation of the ‘image’ filter VS overall evaluation

Before the normalized data, just one more need to classify the Correlation of the ‘image’ filter and overall evaluations.

Mathematical proofed: Correlation of image filtering & overall evaluation

X=image filter

Y=overall evaluation

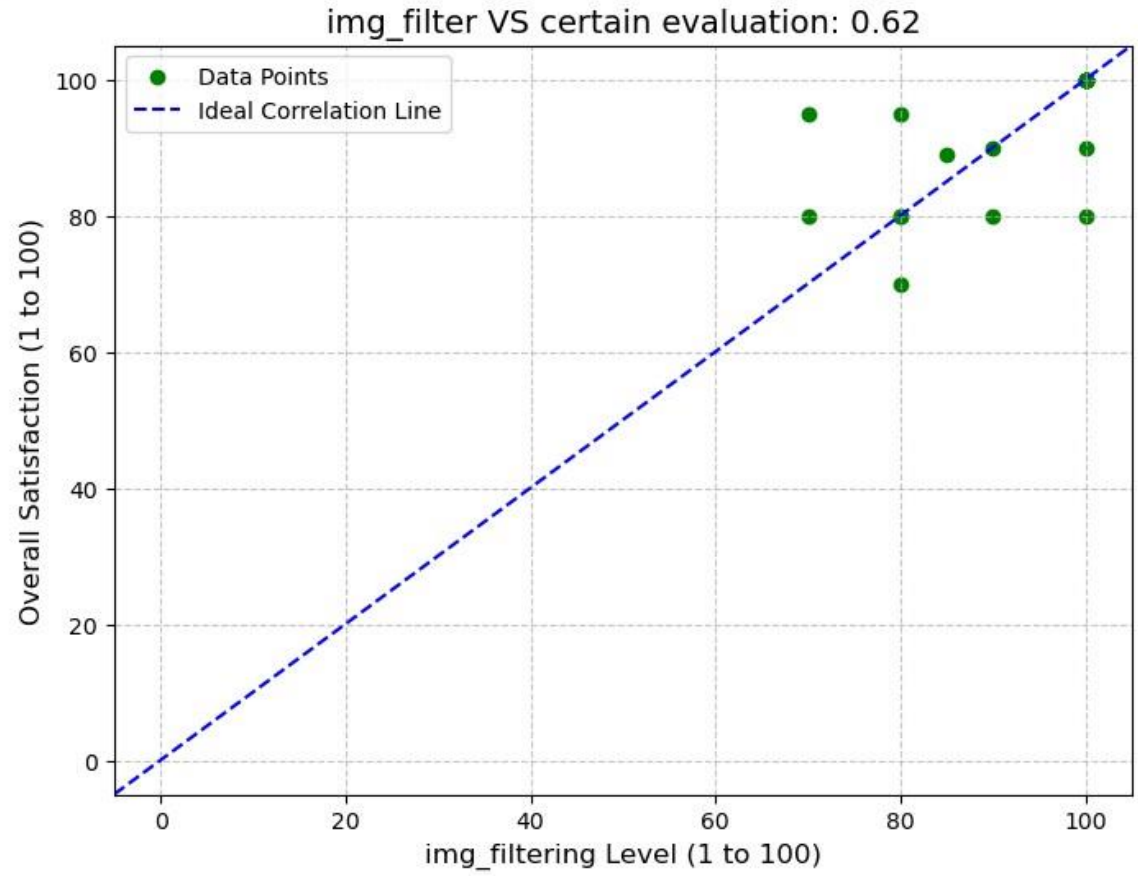
R (image filtering, overall evaluation) = $\frac{Cov(X,Y)}{\sigma X \sigma Y} =$

$$\frac{\frac{1}{22}\sum_{i=1}^{23}\{(X-mean\ \ of\ \ X)\times(Y-mean\ \ of\ \ Y)\}}{\sqrt{\frac{1}{22}\sum_{i=1}^{23}(X-mean\ \ of\ \ X)}\times\frac{1}{22}\sum_{i=1}^{23}(Y-mean\ \ of\ \ Y)} = \frac{\frac{1}{22}(2005-92.05)\times\frac{1}{22}(2029-92.23)}{\sqrt{\frac{1}{22}(2005-92.05)}\times\sqrt{\frac{1}{22}(2029-92.23)}} =$$

$$\frac{61.58}{10.52\times9.40} = 0.62$$

∴ correlation of the image filter and overall evaluations is positive (0.62)

```python
import numpy as np
import matplotlib.pyplot as plt

img_filtering= np.array([100,100,80,80,100,100,100,100,100,100,100,100,85,80,90,100,100,100,90,80,70,70])
overall_satisfaction= np.array([100,80,95,70,100,100,90,100,100,100,100,100,89,80,80,100,100,100,90,80,95,80])

correlation = np.corrcoef(img_filtering, overall_satisfaction)[0, 1]

plt.figure(figsize=(8, 6))
plt.scatter(img_filtering, overall_satisfaction, color='green', label='Data Points')
plt.title(f'img_filter VS certain evaluation: {correlation:.2f}', fontsize=14)
plt.xlabel('img_filtering Level (1 to 100)', fontsize=12)
plt.ylabel('Overall Satisfaction (1 to 100)', fontsize=12)
plt.axline((0, 0), slope=1, color='blue', linestyle='--', label='Ideal Correlation Line')
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend(fontsize=10)
plt.show()
```



∴PCC model of the ‘Image-filtering VS overall evaluation” is correct.

-Train the data of each class (features)

Until we only depended on the law data, but now we need to use training data for the evaluation. This is to prevent the overfitting[5] problem. Because some raw datasets have

Data redundancy problem. [6]For example, my survey dataset has a similar proportion of scores. That reason, I have to divide the datasets as the input data, and output data.

The Input data will be the predicted data of the model, and I will use KNN[7] to predict

The model to select. The score finds the most weight of the percentage of the scores given by the users and represents it with the performance evaluation model.

[228]:

	How accurate is the word filtering level of that bot? Please grade it (from 1 to 100)	How accurate is the photo(jpeg, png, etc) filtering level of that bot? Please grade it (from 1 to 100)	(Effectiveness of Auto-Delete Feature) how useful the auto - delete feature is for specific keywords. (from 1 to 100)	(Consistency in function Execution) how consistency the bot performs its functions? (from 1 to 100)	(Responds speed) how much satisfied the speeds which the bot detects and deletes message or images? (from 1 to 100)	What's the overall satisfaction with this bot running? (from 1 to 100)
0	100	100	100	0.0	100	100
1	80	100	80	100.0	80	80
2	80	80	90	70.0	100	95
3	70	80	90	60.0	80	70
4	100	100	100	100.0	100	100
5	100	100	100	100.0	100	100
6	80	100	100	80.0	100	90
7	100	100	100	100.0	100	100
8	100	100	100	100.0	100	100
9	100	100	100	100.0	100	100
10	100	100	100	100.0	100	100
11	100	100	100	100.0	100	100
12	99	85	90	89.0	99	89
13	85	80	82	79.0	100	80
14	70	90	70	0.0	100	80
15	100	100	100	100.0	100	100
16	100	100	100	100.0	100	100
17	100	100	100	100.0	100	100
18	80	90	50	95.0	80	90
19	80	80	90	90.0	80	80
20	80	70	80	90.0	85	95
21	100	70	90	90.0	80	80

[5] Overfitting is meaning situation about learned overate data.

[6] Reference(2020)박해선 혼자공부하는 머신 러닝,딥러닝, 서울:한빛미디어 pg:122

[7] The K-nearest neighbor (KNN) algorithm looks at the proportion of other data in the dataset and uses the largest percentage as the answer.

```
[66]: from sklearn.model_selection import train_test_split
      chatfilter_input_train, chatfilter_input_test, chatfilter_output_train, chatfilter_output_test =train_test_split(chatfilter_input, chatfilter_output, te

[39]: imgfilter_input_train, imgfilter_input_test, imgfilter_output_train, imgfilter_output_test = train_test_split(imgfilter_input, imgfilter_output, test_si

[40]: useful_input_train, useful_input_test, useful_output_train, useful_output_test = train_test_split(useful_input, useful_output, test_size=0.3, random_sta

[41]: Consistency_input_train, Consistency_input_test, Consistency_output_train, Consistency_output_test = train_test_split(Consistency_input, Consistency_out

[42]: RespondSpeed_input_train, RespondSpeed_input_test, RespondSpeed_output_train, RespondSpeed_output_test=train_test_split(RespondSpeed_input, RespondSpeed_

[43]: overallevaluation_input_train, overallevaluation_input_test, overallevaluation_output_train, overallevaluation_output_test=train_test_split(overallevalu

[44]: #model predict - by knn
      from sklearn.neighbors import KNeighborsClassifier
      kn = KNeighborsClassifier()
      kn.fit(chatfilter_input_train,chatfilter_output_train)
      kn.score(chatfilter_input_test, chatfilter_output_test)
      chatfilter_pred=kn.predict(chatfilter_input_test)

      kn.fit(imgfilter_input_train,imgfilter_output_train)
      kn.score(imgfilter_input_test, imgfilter_output_test)
      imgfilter_pred=kn.predict(imgfilter_input_test)

      kn.fit(useful_input_train,useful_output_train)
      kn.score(useful_input_test, useful_output_test)
      useful_pred=kn.predict(useful_input_test)

      kn.fit(Consistency_input_train,Consistency_output_train)
      kn.score(Consistency_input_test, Consistency_output_test)
      Consistency_pred=kn.predict(Consistency_input_test)

      kn.fit(RespondSpeed_input_train, RespondSpeed_output_train)
      kn.score(RespondSpeed_input_test, RespondSpeed_output_test)
      RespondSpeed_pred=kn.predict(RespondSpeed_input_test)

      kn.fit(overallevaluation_input_train, overallevaluation_output_train)
      kn.score(overallevaluation_input_test, overallevaluation_output_test)
```

```
[46]: overallevaluation_input=data.values
      overallevaluation_output=data.iloc[:,5].values
      overallevaluation_output

[46]: array([100,  80,  95,  70, 100, 100,  90, 100, 100, 100, 100, 100,  89,
              80,  80, 100, 100, 100,  90,  80,  95,  80], dtype=int64)

[92]: import train_test_split
      filter_input_test, chatfilter_output_train, chatfilter_output_test =train_test_split(chatfilter_input,chatfilter_output, test_size=0.3, random_state = 42

[62]: imgfilter_input_train, imgfilter_input_test, imgfilter_output_train, imgfilter_output_test =train_test_split(imgfilter_input, imgfilter_output, test_siz

[64]: ul_input_train, useful_input_test, useful_output_train, useful_output_test = train_test_split(useful_input, useful_output, test_size=0.3, random_state=42

[76]: Consistency_input_train, Consistency_input_test, Consistency_output_train, Consistency_output_test = train_test_split(Consistency_input, Consistency_out

[78]: RespondSpeed_input_train, RespondSpeed_input_test, RespondSpeed_output_train, RespondSpeed_output_test=train_test_split(RespondSpeed_input, RespondSpeed_

[82]: overallevaluation_input_train, overallevaluation_input_test, overallevaluation_output_train, overallevaluation_output_test=train_test_split(overallevalu
```

```
[40]: data=new_data2
      chatfilter_input=data.values
      chatfilter_output=data.iloc[:,0].values
      chatfilter_output

[40]: array([100,  80,  80,  70, 100, 100,  80, 100, 100, 100, 100, 100,  99,
              85,  70, 100, 100, 100,  80,  80,  80, 100], dtype=int64)

[42]: imgfilter_input=data.values
      imgfilter_output=data.iloc[:,1].values
      imgfilter_output

[42]: array([100, 100,  80,  80, 100, 100, 100, 100, 100, 100, 100, 100,  85,
              80,  90, 100, 100, 100,  90,  80,  70,  70], dtype=int64)

[14]: useful_input=data.values
      useful_output=data.iloc[:,2].values
      useful_output

[14]: array([100,  80,  90,  90, 100, 100, 100, 100, 100, 100, 100, 100,  90,
              82,  70, 100, 100, 100,  50,  90,  80,  90], dtype=int64)

[15]: Consistency_input=data.values
      Consistency_output=data.iloc[:,3].values
      Consistency_output

[15]: array([  0., 100.,  70.,  60., 100., 100.,  80., 100., 100., 100., 100.,
             100.,  89.,  79.,   0., 100., 100., 100.,  95.,  90.,  90.,  90.])

[44]: RespondSpeed_input=data.values
      RespondSpeed_output=data.iloc[:,4].values
      RespondSpeed_output

[44]: array([100,  80, 100,  80, 100, 100, 100, 100, 100, 100, 100, 100,  99,
             100, 100, 100, 100, 100,  80,  80,  85,  80], dtype=int64)

[46]: overallevaluation_input=data.values
      overallevaluation_output=data.iloc[:,5].values
      overallevaluation_output

[46]: array([100,  80,  95,  70, 100, 100,  90, 100, 100, 100, 100, 100,  89,
              80,  80, 100, 100, 100,  90,  80,  95,  80], dtype=int64)

[92]: import train_test_split
      filter_input_test, chatfilter_output_train, chatfilter_output_test =train_test_split(chatfilter_input,chatfilter_output, test_size=0.3, random_state = 42
```

- Find the problem statements using the Evaluation method-

-confusion matrix-

It is enough to prove the accuracy of the PCC model, however, we have not yet found improvement points in the filtering systems. (word, Image) On the other hand, there I can use the evaluation method. This method is for feedback and measuring the performance of the model. And, it will be represented by a ‘confusion matrix’

		Actual correct answer	
		True(1)	False(0)
Predicted Correct answer	Positive (1) == True	TP(True + positive)	FP(false + positive)
	Negative(0) == False	FN (False + Negative)	TN (True + Negative)

A confusion matrix contains 4 cases, each ‘True Positive’, ‘False Positive’, ‘True Negative’, and ‘False Negative’.

Firstly, True positive means when the model predicted the correct answer and the actual answer has the correct answer.

Secondly, False Positive means when the model predicted the correct answer but the actual answer is incorrect.

Thirdly, False Negative means when the model predicted the incorrect answer but the actual answer is the correct answer.

Lastly, True Negative means when the model predicted the incorrect answer and the actual answer has the wrong answer.

-Precision, Recall and Accuracy, F1-score-

Precision is the percentage of correct answers among those predicted by the model.

$$\text{Precision} = \frac{TP}{TP+FP}$$

Recall also was the percentage of what the model predicts to be the correct answer among the actual answers. (True Positive Rate)

$$\text{Recall(TPR)} = \frac{TP}{TP+FN}$$

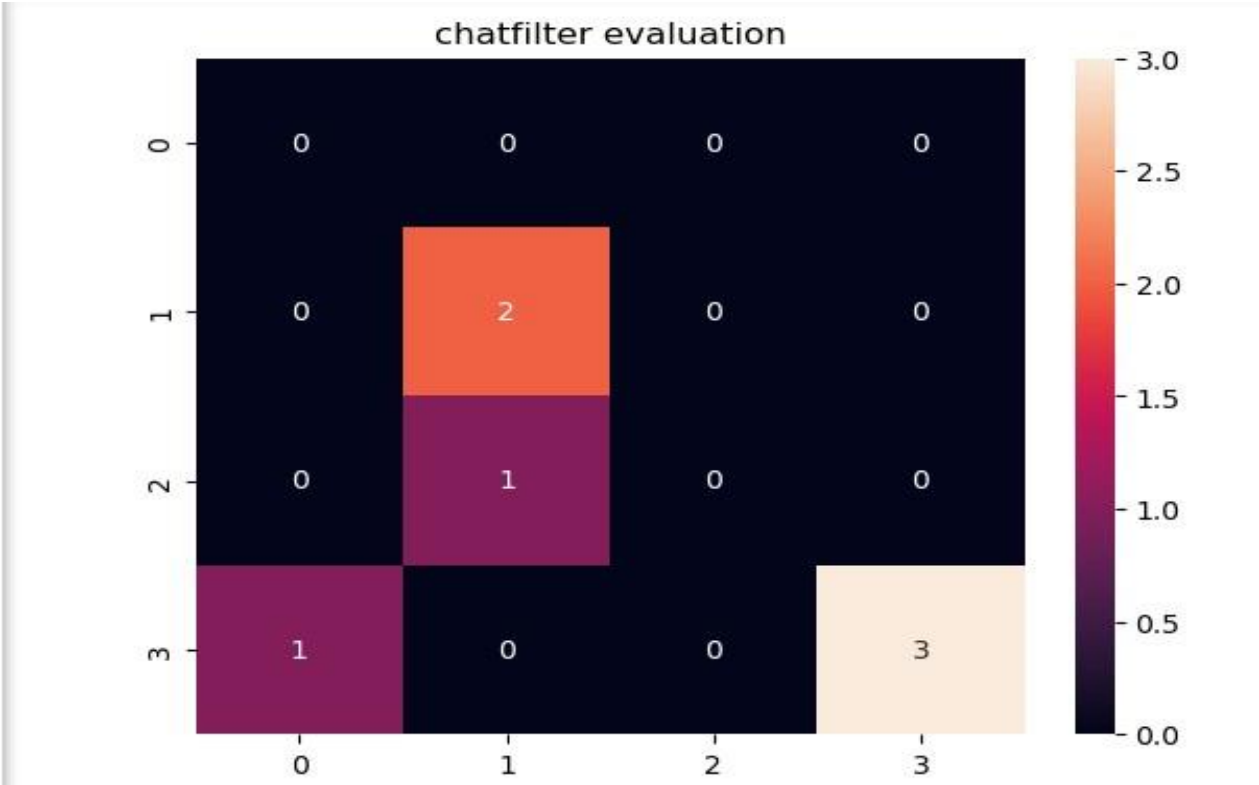
Accuracy is the number of correct predictions to the total number of input samples.

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$$

F1-score represents the Harmonic Mean between Precision and Recall and it tells how precise the ML model is. This means there is not only accuracy and precision in all of the efficient measures for classification tasks. F1-score is the most important thing in evaluation methods.

$$\text{F1-score} = \frac{2(precision \times recall)}{(precision+recall)}$$

- Precision, Recall, and Accuracy, F1-score of chat filter-



	precision	recall	f1-score	support
70	0.00	0.00	0.00	0
80	0.67	1.00	0.80	2
85	0.00	0.00	0.00	1
100	1.00	0.75	0.86	4
accuracy			0.71	7
macro avg	0.42	0.44	0.41	7
weighted avg	0.76	0.71	0.72	7

TP (Positive + Positive, True + True) = 3 + 2 =5

FP (Negative + True) = 1

FN (Positive + False) = 1

TN =0

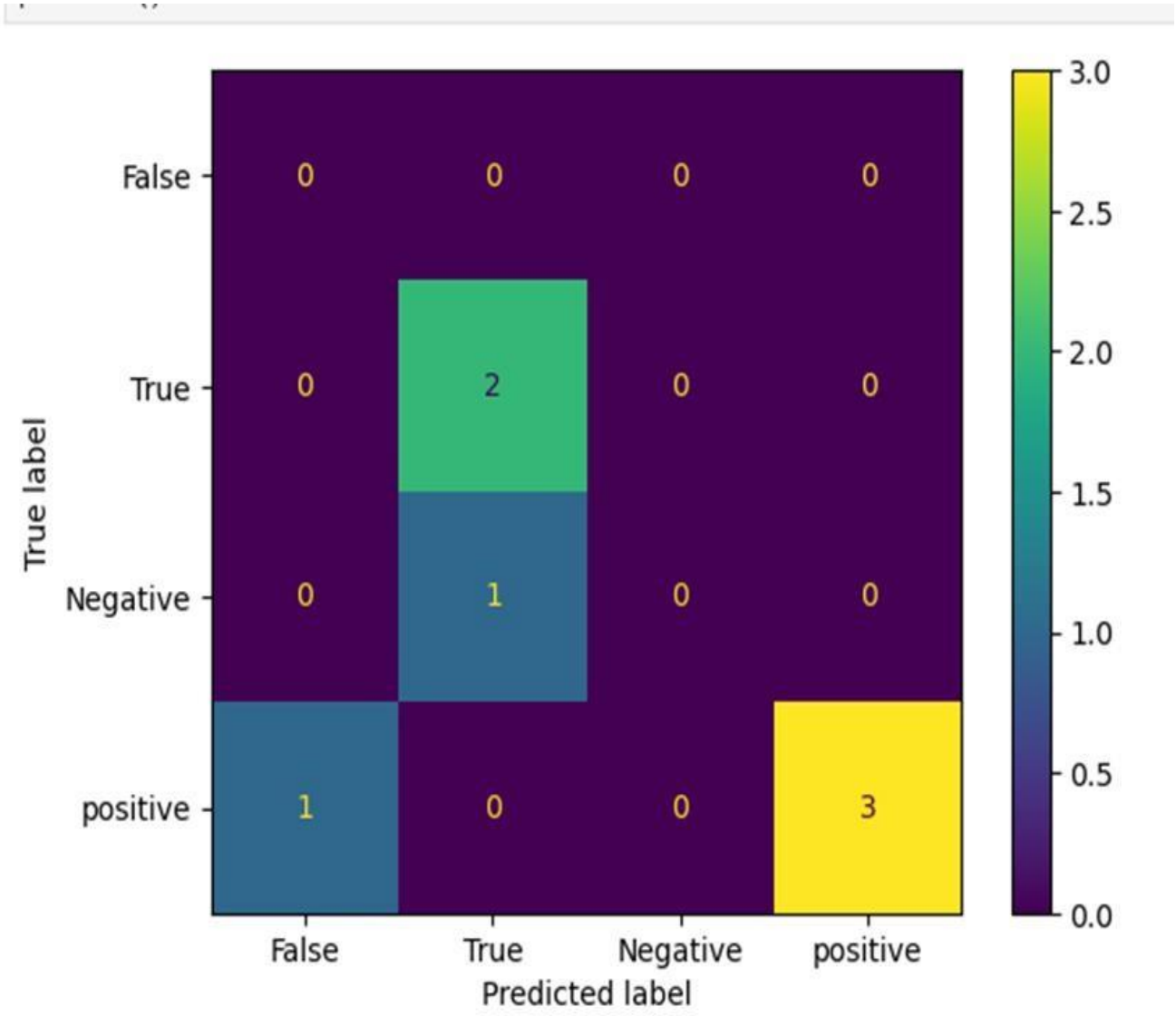
$$\therefore \text{Precision} = \frac{TP}{TP+FP} = \frac{5}{5+1} = \frac{5}{6} \approx 0.833$$

$$\therefore \text{Recall(TPR)} = \frac{TP}{TP+FN} = \frac{5}{5+1} = \frac{5}{6} \approx 0.833$$

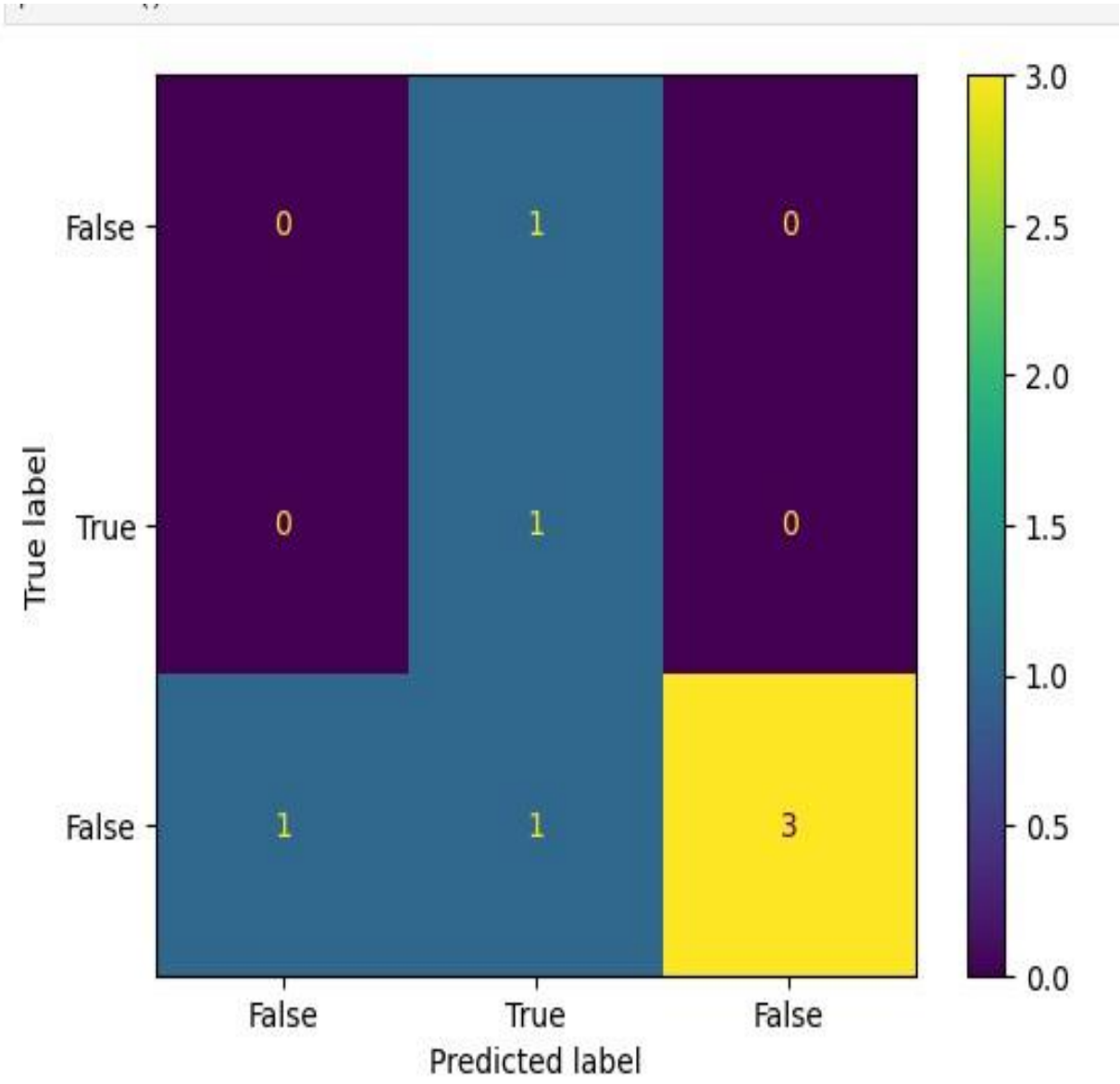
$$\therefore \text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN} = \frac{5+0}{5+1+0+1} = \frac{5}{7} \approx 0.714$$

$$\therefore \text{F1-score} = \frac{2(precision \times recall)}{(precision+recall)} = \frac{2(80\% \times 80\%)}{(80\%+80\%)} = = 0.8 = 80\%$$

Therefore, the chat-filter performance evaluation gets 80 scores.



- Precision, Recall, and Accuracy, F1-score of Image filter-



TP (True + True) = 1

FP (False, True + False, True) = 1 + 1 =2

TN (False, False + False, False) =1 + 3 =4

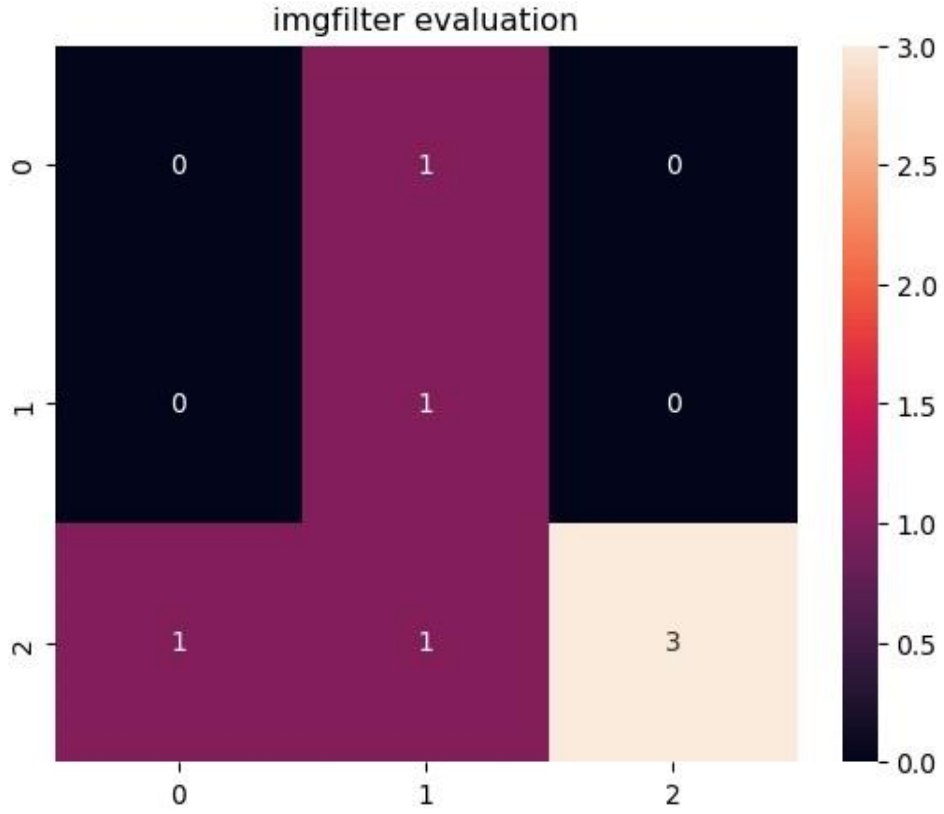
FN = 0

$\therefore$ Precision = $\frac{TP}{TP+FP} = \frac{1}{1+2} = \frac{1}{3} \approx 0.333$

$$\therefore \text{Recall(TPR)} = \frac{TP}{TP+FN} \quad \frac{1}{1+0} = = 1$$

$$\therefore \text{Accuracy} = \frac{}{TP+FP+TN+FN} \quad \frac{TP+TN}{1+2+4+0} = \frac{1+4}{7} = 5 \approx 0.714$$

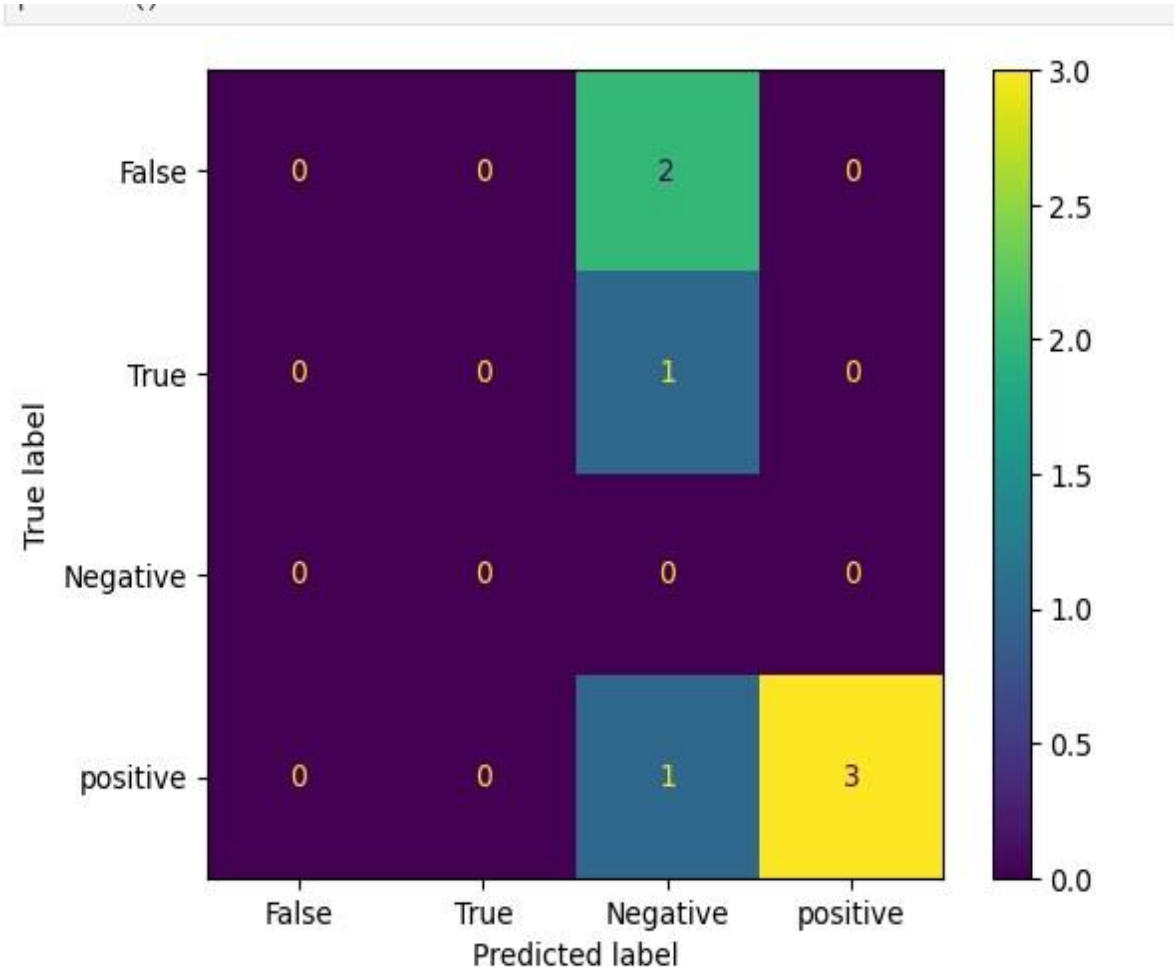
$$\therefore \text{F1-score} = \frac{2(precision \times recall)}{(precision+recall)} \quad \frac{2(30\% \times 100\%)}{(30\%+100\%)} = = 0.4615 \approx 0.5 = 50\%$$



	precision	recall	f1-score	support
70	0.00	0.00	0.00	1
80	0.33	1.00	0.50	1
100	1.00	0.60	0.75	5
accuracy			0.57	7
macro avg	0.44	0.53	0.42	7
weighted avg	0.76	0.57	0.61	7

Therefore, the image-filter performance gets 50 scores.

score of useful evaluation-



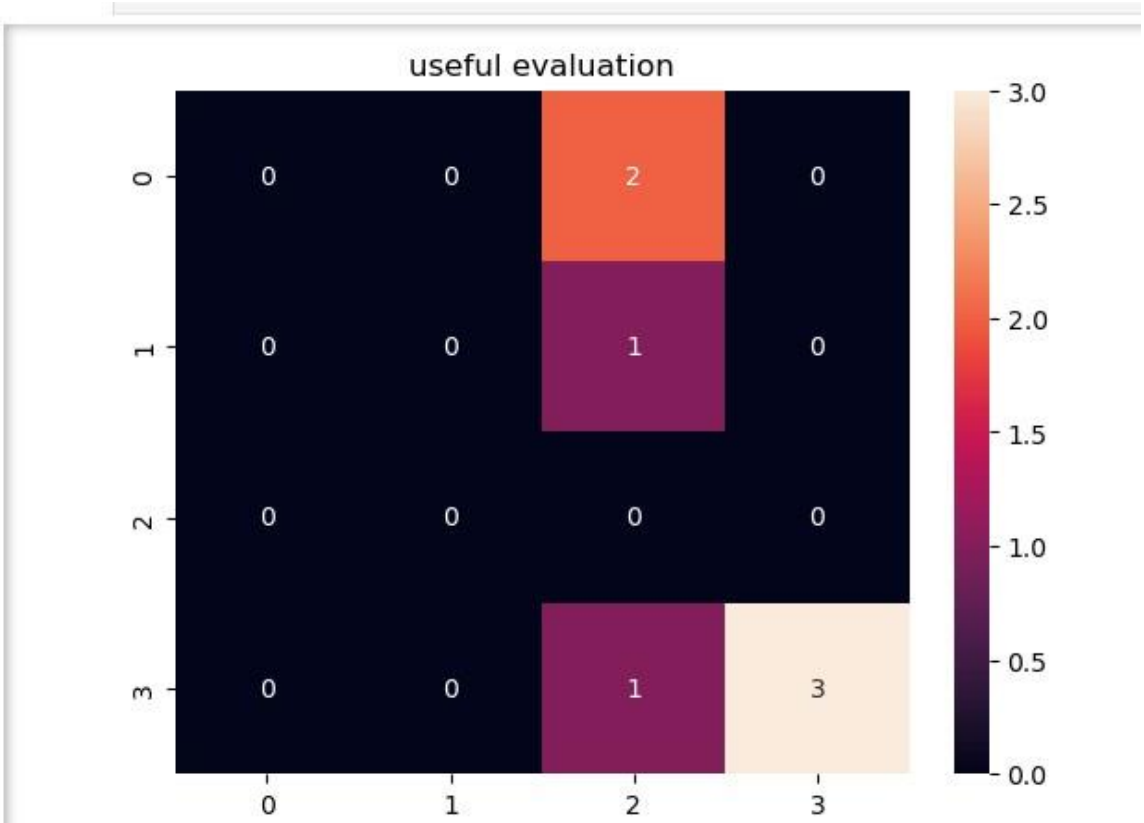
TP= (Positive + Positive) = 3

FP = 0

TN= (False + Negative) = 2

- Precision, Recall, and Accuracy, F1-

FN= (True, Negative + Positive, Negative) =1 + 1 = 2



	precision	recall	f1-score	support
80	0.00	0.00	0.00	2
82	0.00	0.00	0.00	1
90	0.00	0.00	0.00	0
100	1.00	0.75	0.86	4
accuracy			0.43	7
macro avg	0.25	0.19	0.21	7
weighted avg	0.57	0.43	0.49	7

$$\therefore \text{Precision} = \frac{TP}{TP+FP} = \frac{3}{3+0} = \frac{3}{3} = 1$$

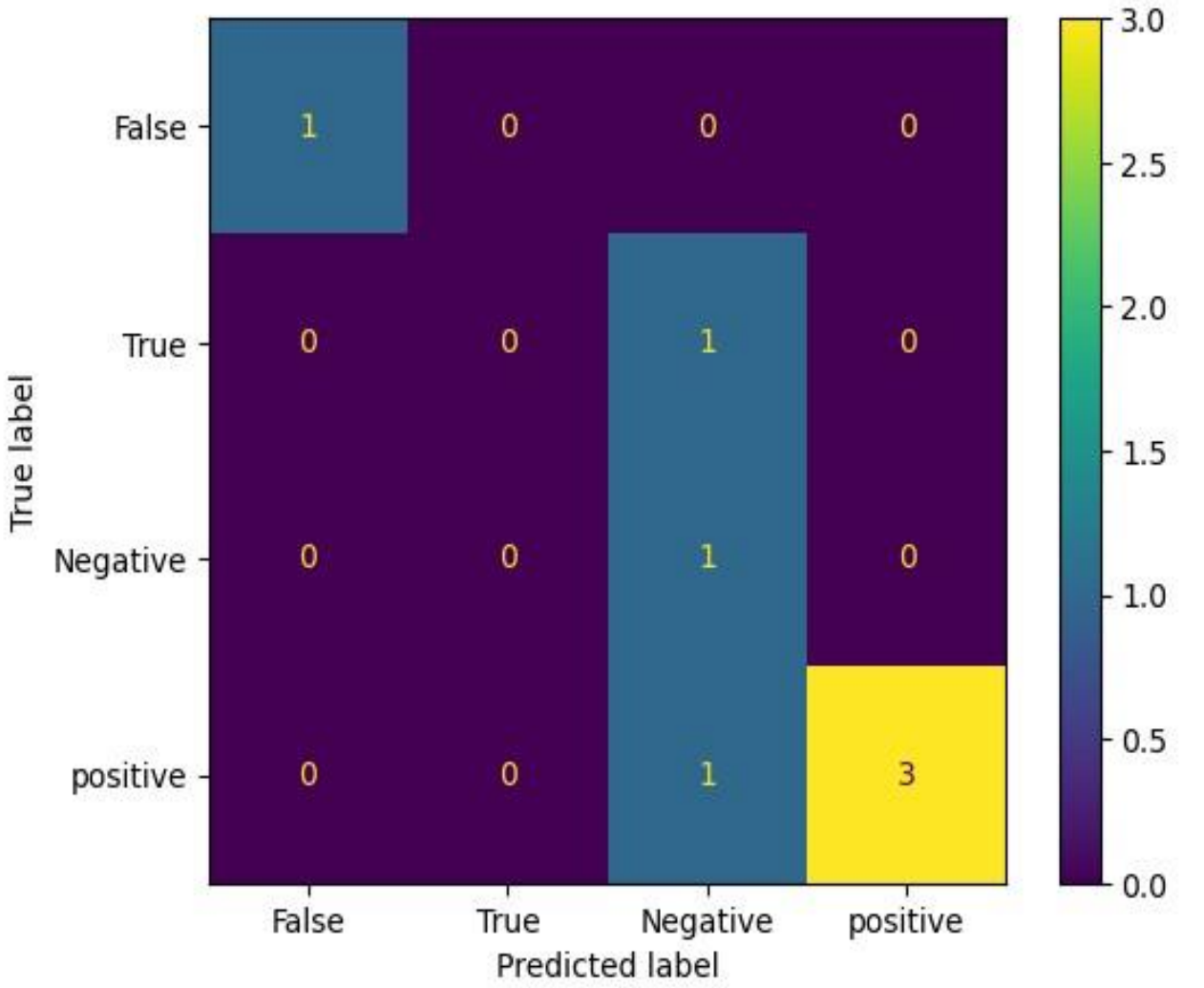
$$\therefore \text{Recall(TPR)} = \frac{TP}{TP+FN} = \frac{3}{3+2} = \frac{3}{5} = 0.6$$

$$\therefore \text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN} = \frac{3+2}{3+0+2+2} = \frac{5}{7} \approx 0.714$$

$$\therefore \text{F1-score} = \frac{2(precision \times recall)}{(precision+recall)} \quad \frac{2(100\% \times 60\%)}{(100\%+60\%)} = = 0.75 \approx 0.8 = 80\%$$

Therefore, the useful evaluation gets 80 scores.

score of Consistency evaluation-



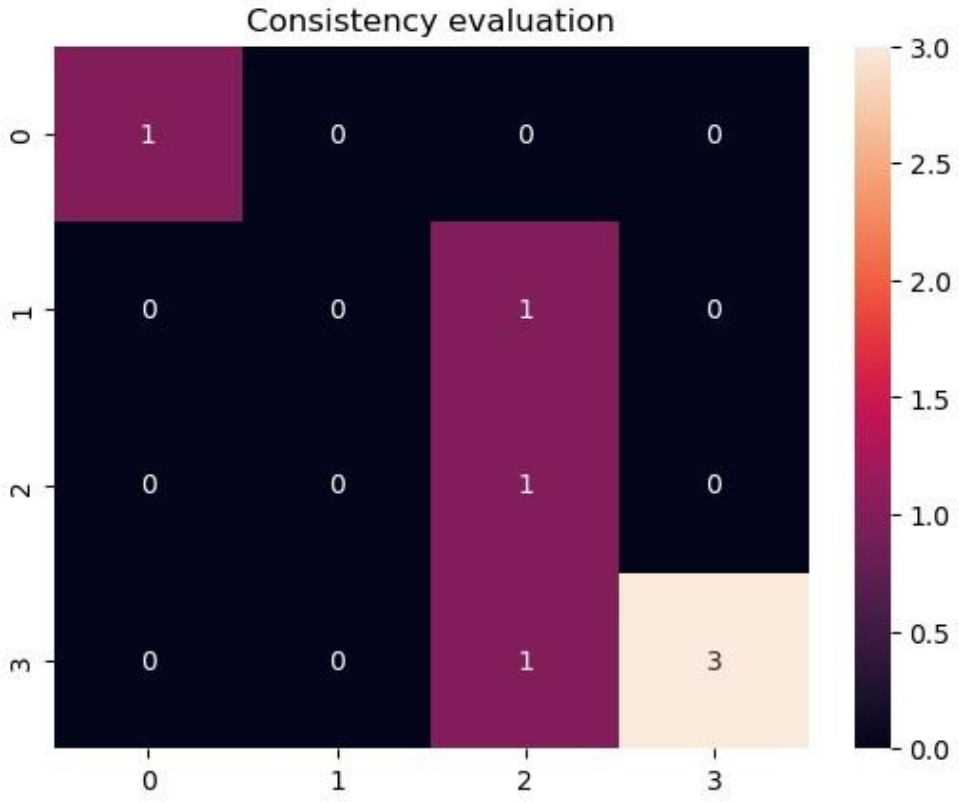
TP= (Positive + Positive) = 3

FP = 0

- Precision, Recall, and Accuracy, F1-

TN= (Negative, Negative + False, False) =1 + 1 = 2

FN = (True, Negative + Positive, Negative) = 1 + 1 = 2



	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	1
79.0	0.00	0.00	0.00	1
90.0	0.33	1.00	0.50	1
100.0	1.00	0.75	0.86	4
accuracy			0.71	7
macro avg	0.58	0.69	0.59	7
weighted avg	0.76	0.71	0.70	7

Precision = $\frac{TP}{TP+FP} = \frac{3}{3+0} = \frac{3}{3} = 1$ ∴

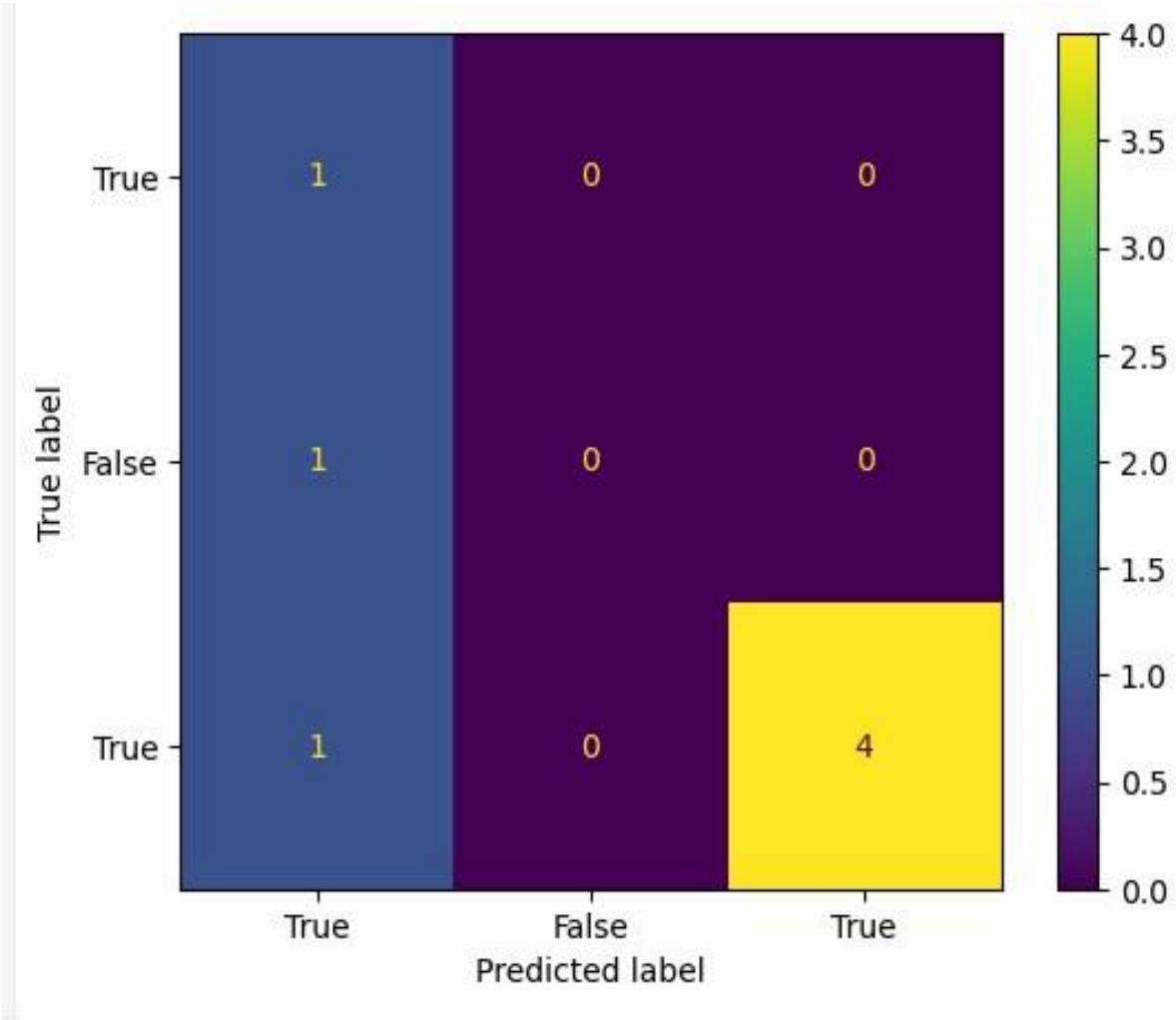
∴ Recall(TPR) = $\frac{TP}{TP+FN} = \frac{3}{3+2} = \frac{3}{5} = 0.6$

$$\therefore \text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN} = \frac{3+2}{3+0+2+2} = \frac{5}{7} \approx 0.714$$

$$\therefore \text{F1-score} = \frac{2(precision \times recall)}{(precision+recall)} = \frac{2(100\% \times 60\%)}{(100\%+60\%)} = 0.75 \approx 0.8 = 80\%$$

Therefore, the Consistency evaluation gets 80 scores.

score of Respond Speed-



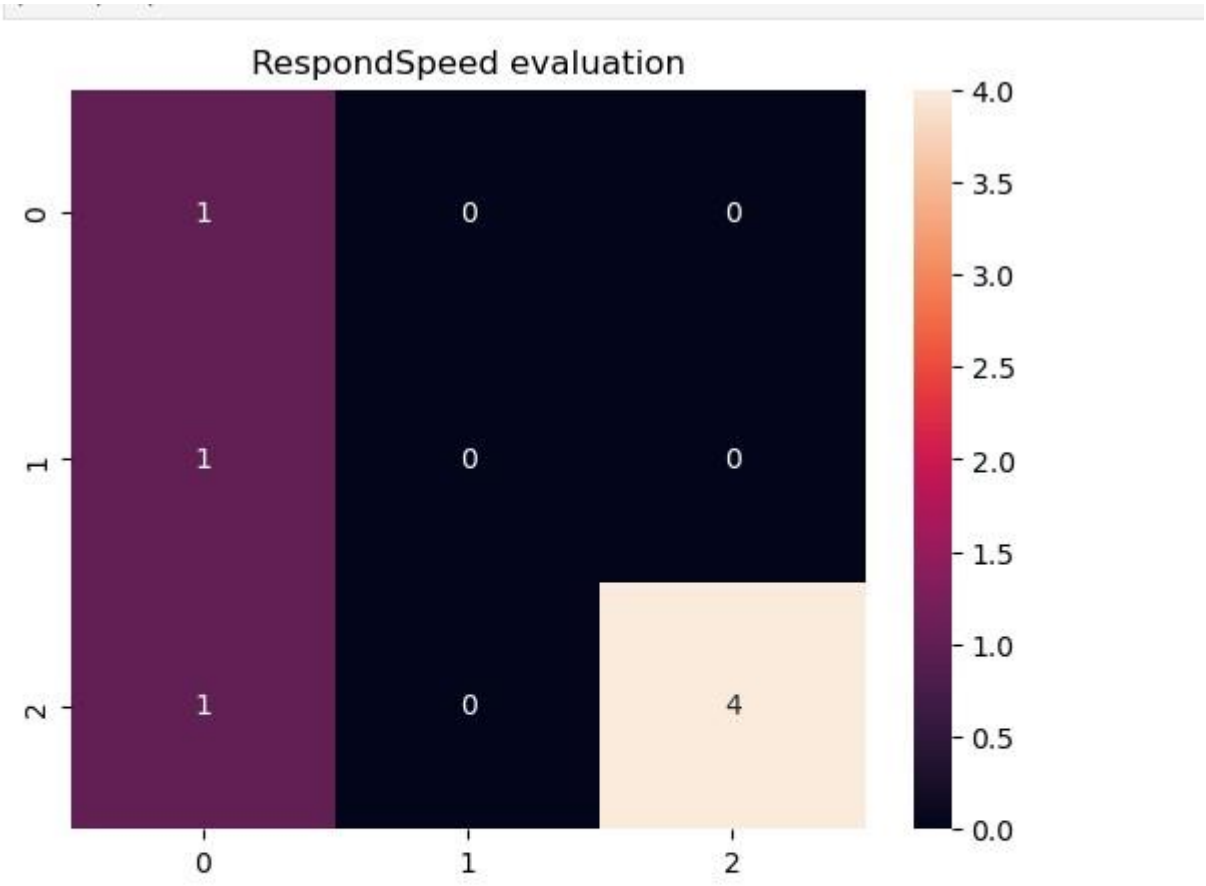
TP = (True, True + True, True + True, True) = 1 + 1 + 4 = 6

- Precision, Recall, and Accuracy, F1-

FP = (False + True) = 1

FN = 0

TN = 0



	precision	recall	f1-score	support
80	0.33	1.00	0.50	1
85	0.00	0.00	0.00	1
100	1.00	0.80	0.89	5
accuracy			0.71	7
macro avg	0.44	0.60	0.46	7
weighted avg	0.76	0.71	0.71	7

$$\therefore \text{Precision} = \frac{TP}{TP+FP} = \frac{6}{6+1} = \frac{6}{7} = 0.857$$

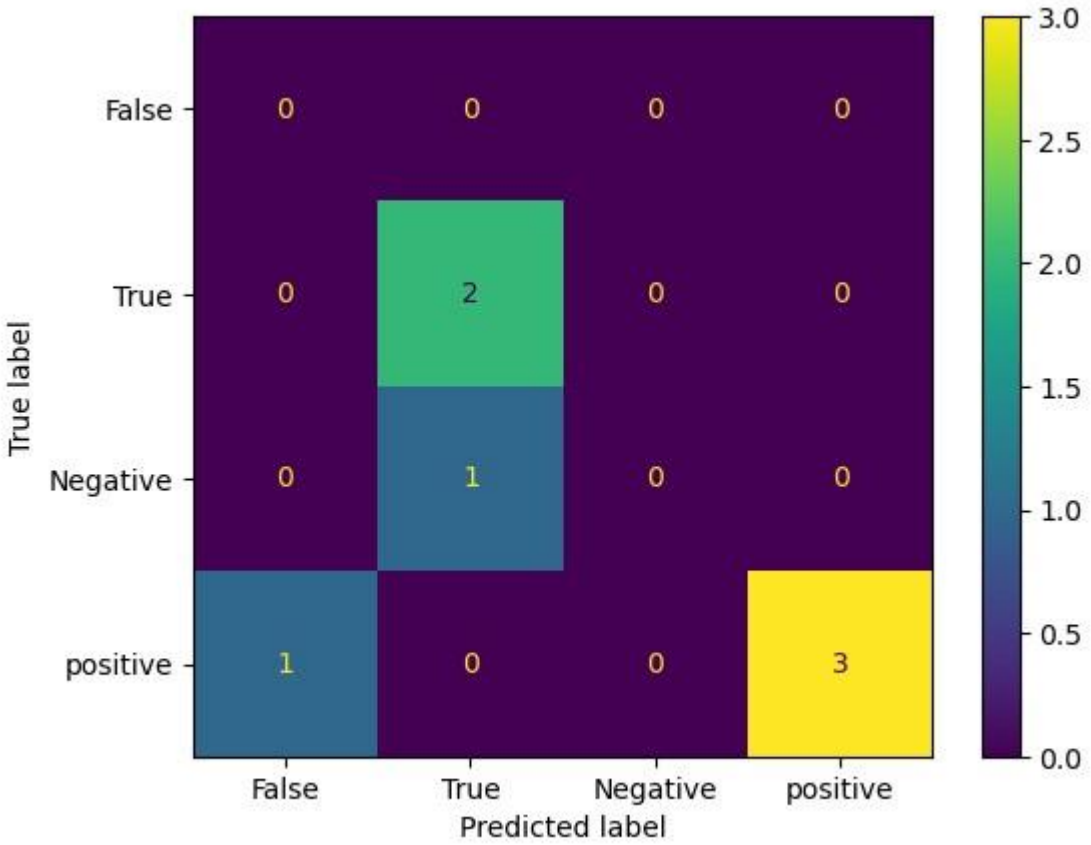
$$\therefore \text{Recall(TPR)} = \frac{TP}{TP+FN} = \frac{6}{6+0} = \frac{6}{6} = 1$$

$$\therefore \text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN} = \frac{6+0}{6+1+0+0} = \frac{6}{7} \approx 0.857$$

$$\therefore \text{F1-score} = \frac{2(precision \times recall)}{(precision+recall)} = \frac{2(0.857\times1)}{(0.857+1)} = = 0.926 \approx 0.9 = 90\%$$

Therefore, the response speed evaluation gets 90 scores.

score of overall evaluation-



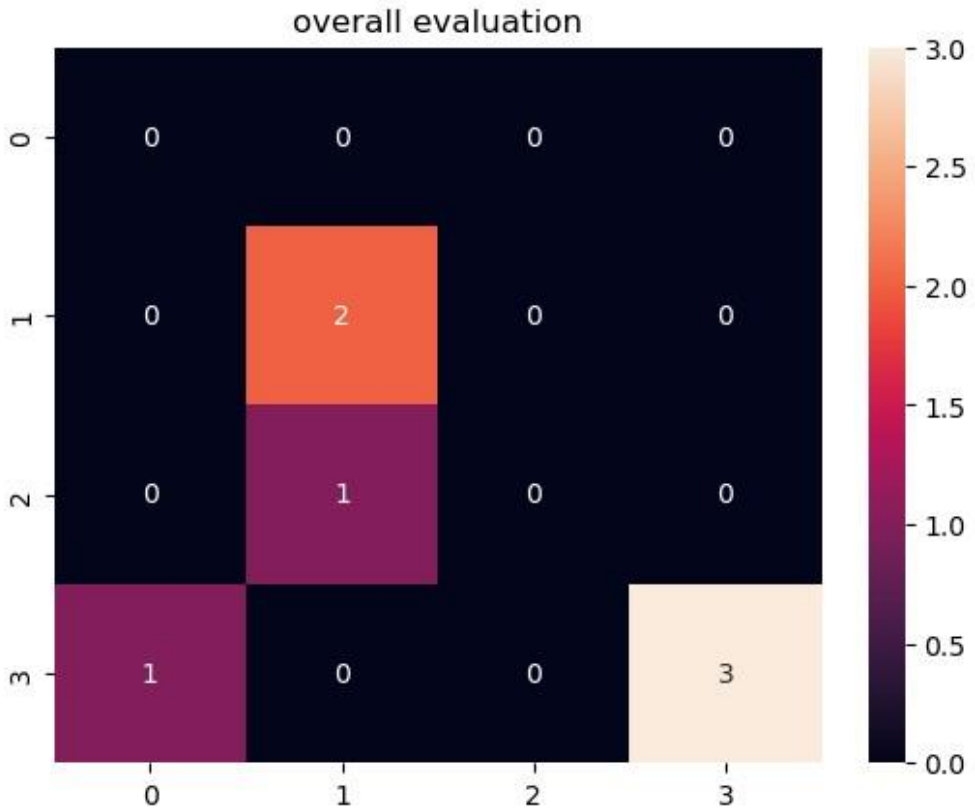
TP = (Positive, Positive + True, True) = 3 + 2 = 5

- Precision, Recall, and Accuracy, F1-

FP = (Negative + Positive) = 1

TN= 0

FN= (Positive + False) = 1



	precision	recall	f1-score	support
70	0.00	0.00	0.00	0
80	0.67	1.00	0.80	2
95	0.00	0.00	0.00	1
100	1.00	0.75	0.86	4
accuracy			0.71	7
macro avg	0.42	0.44	0.41	7
weighted avg	0.76	0.71	0.72	7

$\therefore$ Precision $= \frac{TP}{TP+FP} = \frac{5}{5+1} = \frac{5}{6} = 0.8333 = 0.8$

$\therefore$ Recall(TPR) $= \frac{TP}{TP+FN} = \frac{5}{5+1} = \frac{5}{6} = 0.8333 = 0.8$

$\therefore$ Accuracy $= \frac{TP+TN}{TP+FP+TN+FN} = \frac{5+0}{5+1+0+1} = \frac{5}{7} \approx 0.714$

$2(precision \times recall) \quad 2(0.8333 \times 0.8333)$

- Precision, Recall, and Accuracy, F1-

$$\therefore \text{F1-score} = \frac{}{(precision+recall)} \quad \frac{}{(0.8333+0.8333)} = = 0.833 \approx 0.8 = 80\%$$

Therefore, the overall evaluation is 80 scores.

So, compare the score of each rate

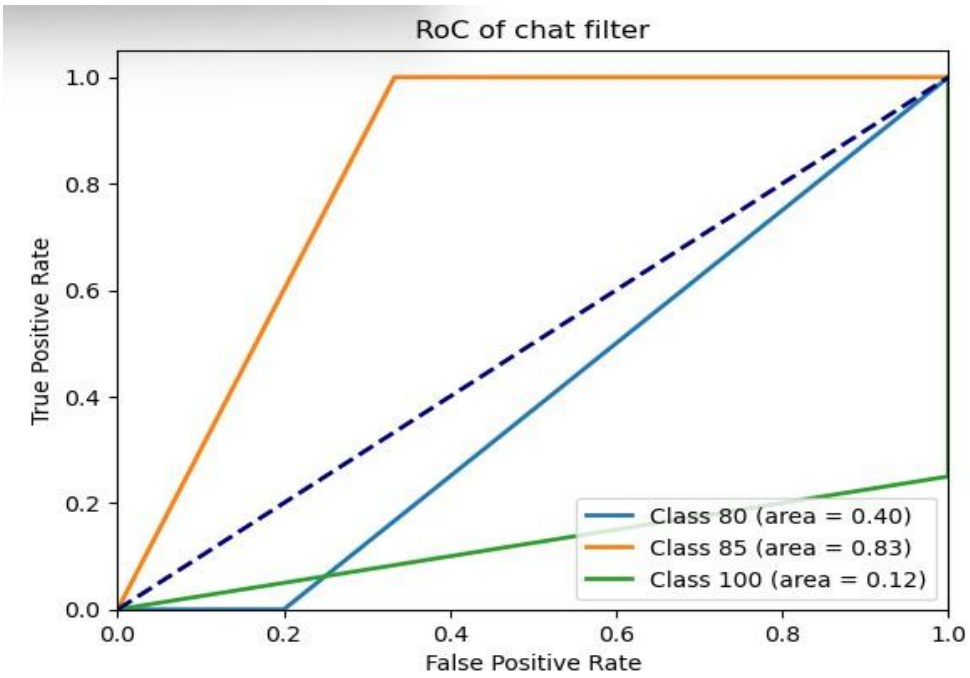
Chat filter= 80, Image-filter= 50, Useful evaluation =80,

Consistency evaluation =80, Respond Speed = 90

∴image filter mostly needs improvements.

-ROC (Receiver Operating Characteristics) of each evaluation class-

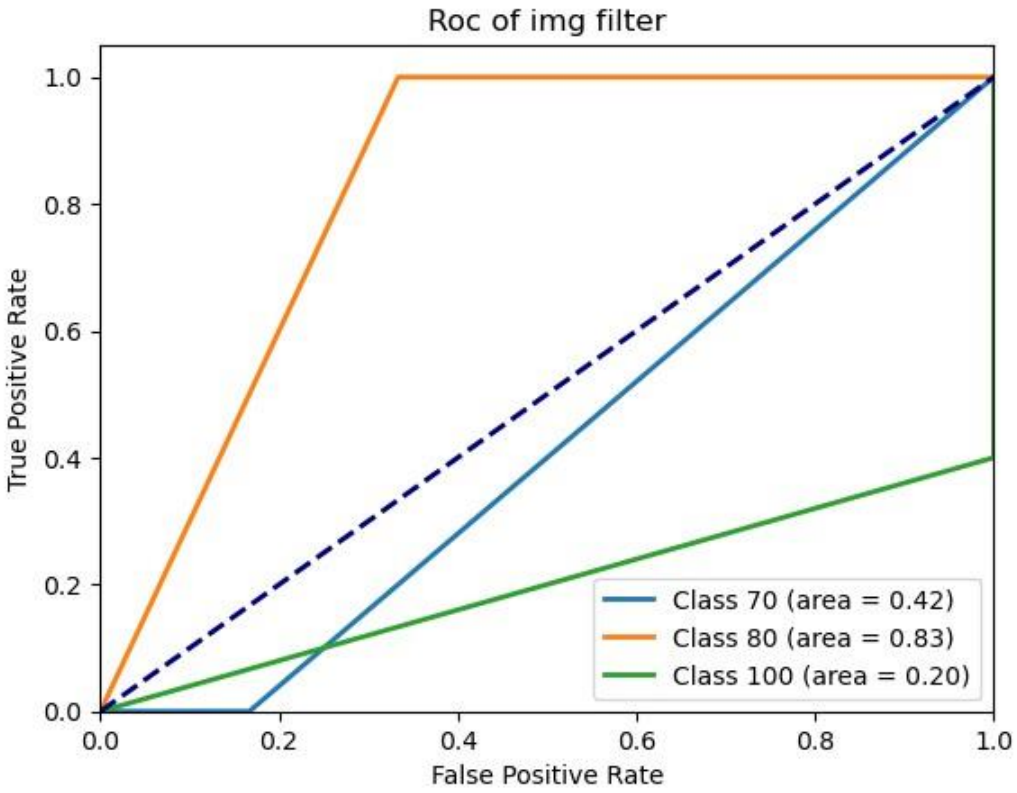
We have already found the improvement point. However, there is one more step for inspecting the data. This is a visualized class as the curve of the graph. The mostly external curve is mostly a better class. In this graph, set the 'X' axis as the FPR and the 'Y' axis as the TPR to show which class most users choose and make intuitive. -Roc of the chat-filter-



In this case, the class with a score of 85 has the highest curve, indicating that most users rated the chat filter evaluation at 85. However, only a few users gave the filter system a perfect score of 100. The dashed line in the middle represents random classification, meaning any curves below this line indicate poor performance. Furthermore, since the AUC is low, the model is not very accurate in predicting users belonging to Class 80. A significant number of users may be incorrectly classified either into or out of this class, reflecting low confidence in the model's

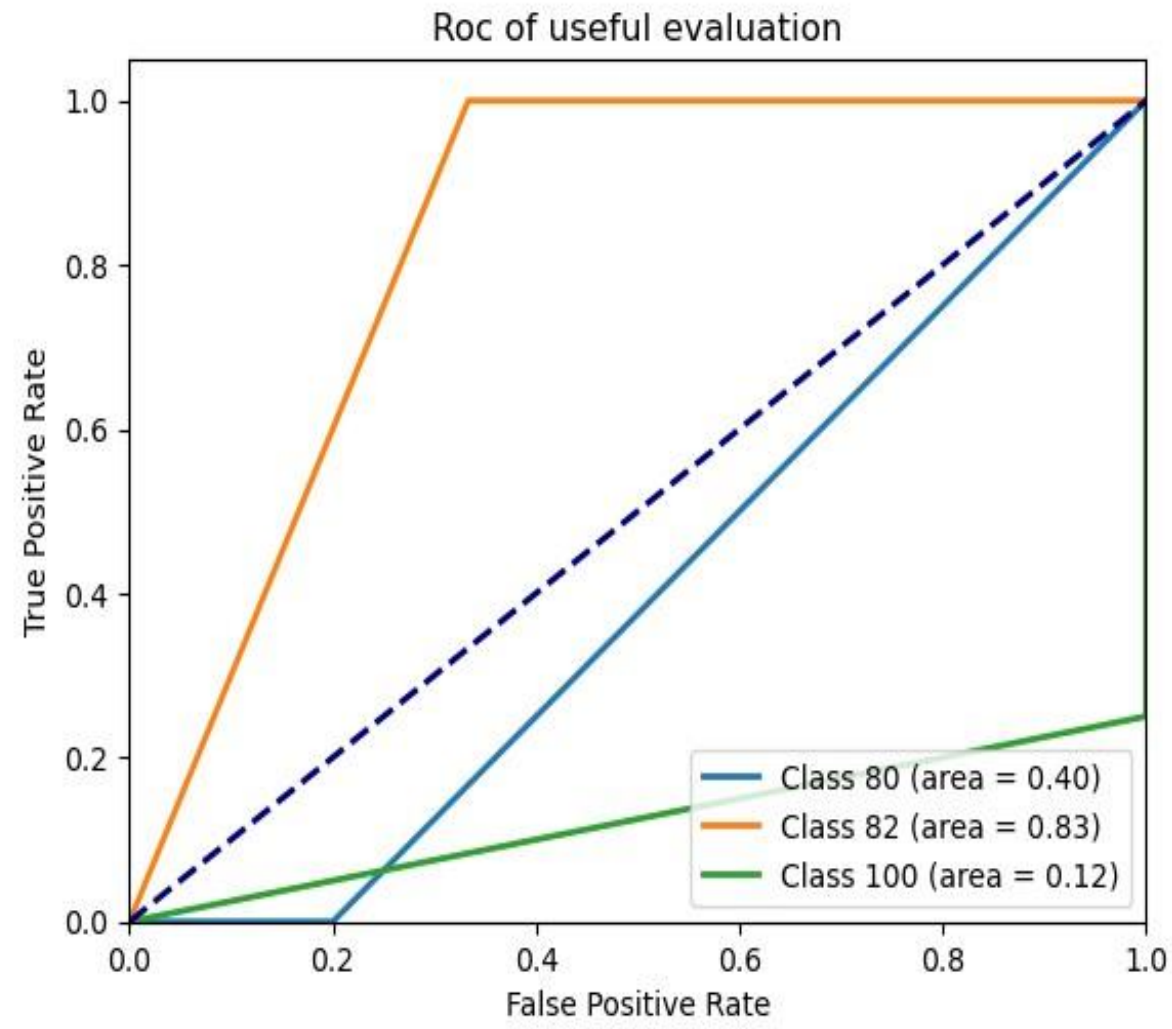
-

performance to distinguish between users accurately. This results could be an over rate of incorrect classifications, whether false positives or false negatives. -Roc of image filter-



In this graph, many users rated the image filter system with a score of 80, with a score of 70 was given for the new data elements. Although fewer users gave a score of 70, it still holds a larger proportion compared to the score of 100. Most users perceive the model's performance as poor, with about 60-70% rating it as unsatisfactory or ineffective, due to the low AUC, which is significantly below 0.5, indicating performance similar to random guessing. On the other hand, around 80-90% of users would rate the model's performance for Class 80 positively, as an AUC above 0.8 generally signifies high-quality predictions. However, about 10-20% of users may still see room for improvement, possibly due to the need for even greater accuracy or domain-specific requirements demanding near-perfect performance.

Roc of useful evaluation-



In the graph, the low AUC for Class 80 suggests that most users would think of the censored bot.

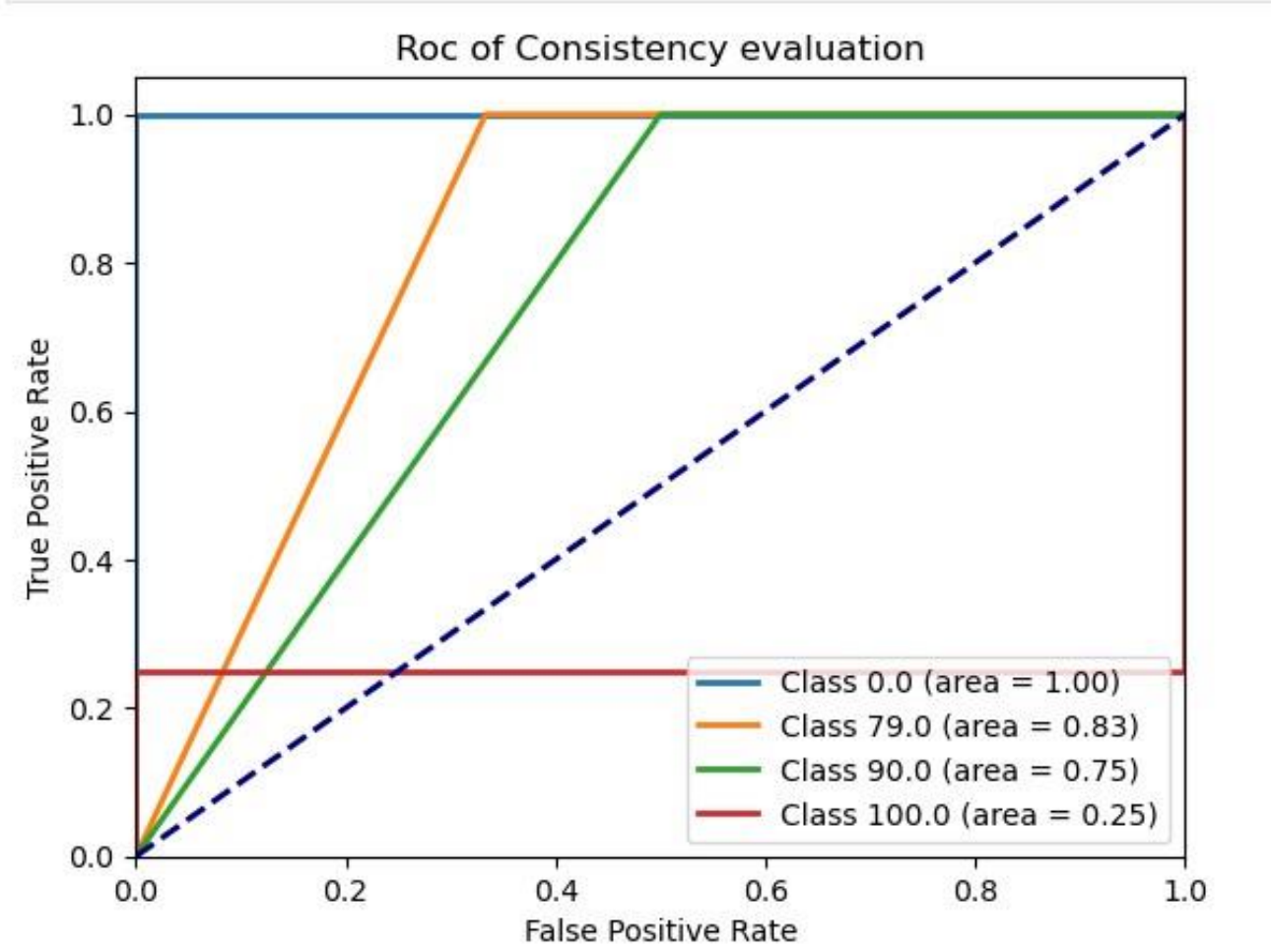
Isn't performing well, with about 60-70% likely rating it poorly, as it barely outperforms random guessing.

In contrast, the high AUC for Class 82 indicates that 80-90% of users would be satisfied with the model, as an AUC above 0.8 generally means it makes accurate predictions.

For Class 100, only a few users gave this score, and the very low AUC suggests that users still find the model unreliable for this class.

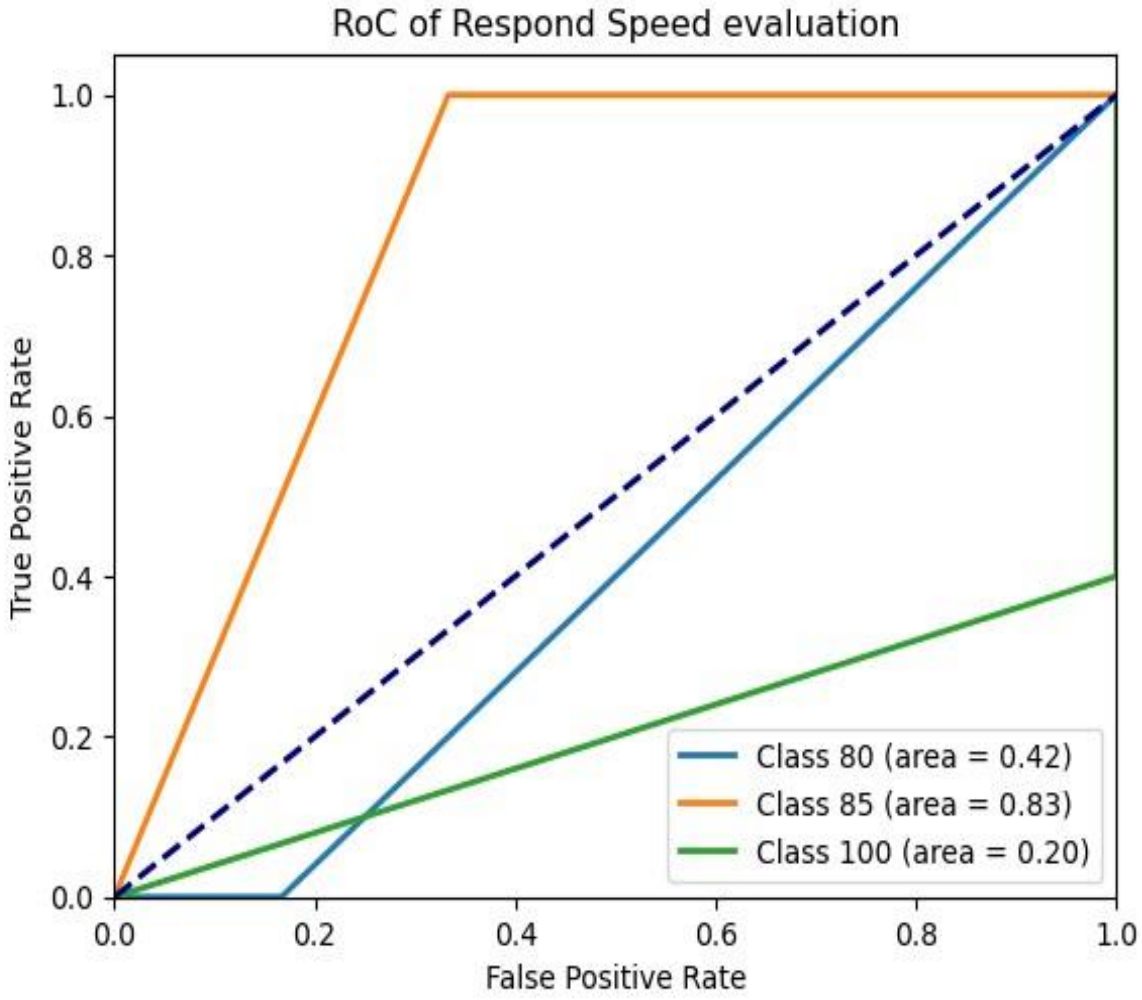
-

Roc of Consistency Evaluation-



in this graph, Class 0.0, the AUC is 1.00, meaning the model performs perfectly, always getting it right. Most users would rate this class very highly. For Class 79.0, the AUC is 0.83, indicating good performance, though not perfect. Most users, about 80-90%, would be satisfied with it, though there are still some errors. For Class 90.0, the AUC is 0.75, which is decent but could be better, and around 70-80% of users would rate it positively. However, Class 100.0 has an AUC of just 0.25, meaning the model struggles to classify users in this category correctly, leading to frequent errors. Most users, about 90-95%, would rate this class poorly due to its unreliable performance.

Roc of Respond Speed Evaluation-



This graph shows how well the model classifies each class in terms of response speed.

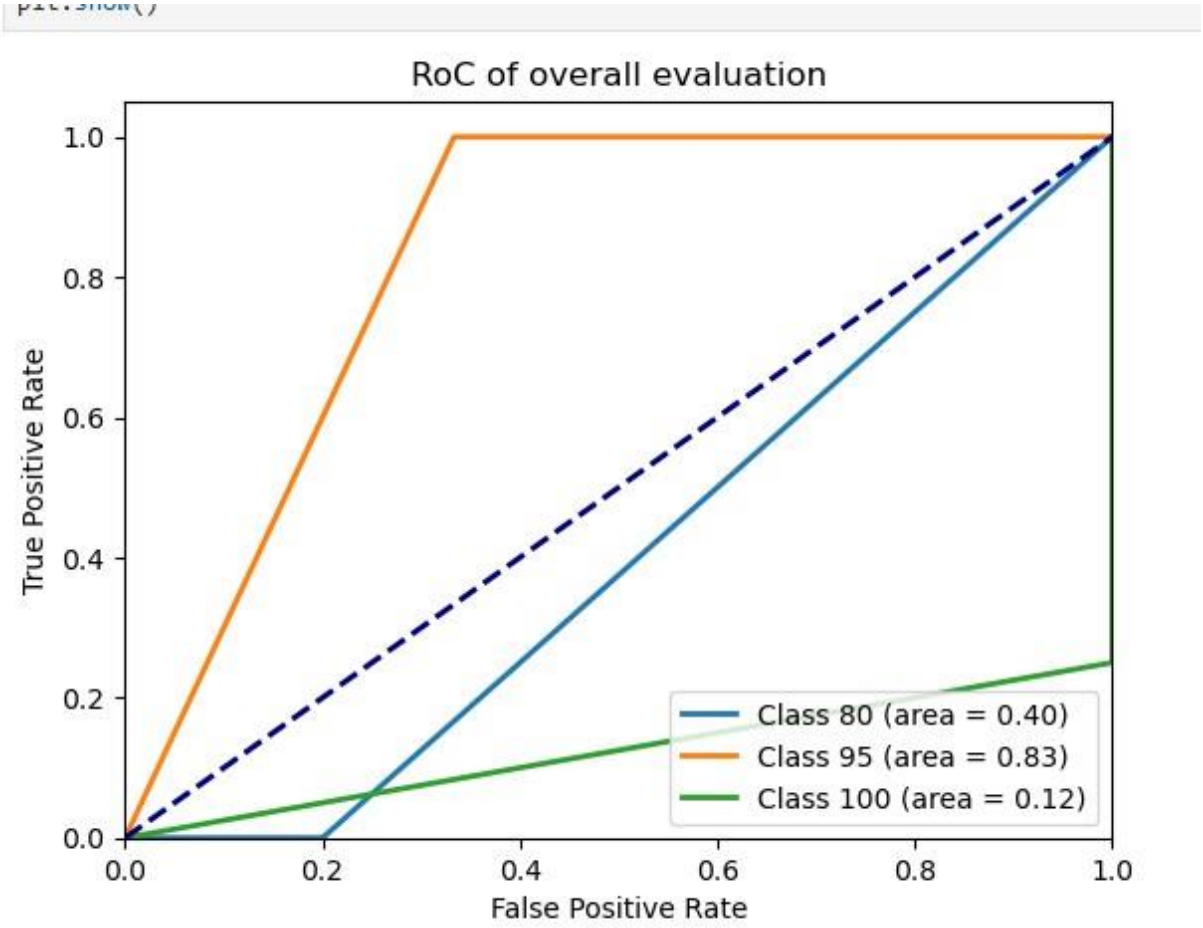
For Class 85, the AUC is 0.83, indicating that the model does a good job of predicting this class. Most users around 80-90% would likely be satisfied with its performance.

On the other hand, Class 80 has an AUC of 0.42, which is quite low, suggesting that the model's performance for this class isn't great. Around 60-70% of users might find it lacking. For Class 100, the AUC is 0.20, which is very poor. This means that the model struggles significantly to accurately predict this class and 90-95% of users would likely rate its performance as unsatisfactory.

In summary, the model performs well for Class 85, earning higher user satisfaction, while Class 80 and Class 100 need improvement to reach acceptable levels of accuracy.

-

-Roc of Overall evaluation-



For Class 95, the AUC is 0.83, indicating that the model is doing a good job here. Most users—around 80-90%—would be satisfied with how accurately the model predicts this class, as it shows strong performance. Class 80, on the other hand, has an AUC of 0.40, which is below average. This means the model is struggling with this class, performing only slightly better than random guessing. Because of this, about 60-70% of users would likely feel that the performance is not good enough.

Lastly, Class 100 has an AUC of 0.12, which is very poor. The model has a hard time accurately classifying users into this category, often resulting in incorrect predictions. Therefore, most users—around 90-95%—would likely be unhappy with the model's performance for this class. In summary, the model works quite well for Class 95, with most users finding it reliable, while Class 80 and Class 100 fall short of expectations, needing substantial improvements to boost user satisfaction.

-

-Problem statements and challenges-

I've been working on training the dataset, dividing it into input, output, and prediction sets, but I'm still facing a data imbalance issue. To address this, I need to tackle the sampling bias problem, which means finding a more balanced dataset. There are several possible solutions, like using SMOTE [8]to augment the data. However, this can lead to overfitting and conflicts with the number of KNN classes. It's a tricky balance—reducing the class count can lead to under fitting, while expanding it too much risks over fitting.

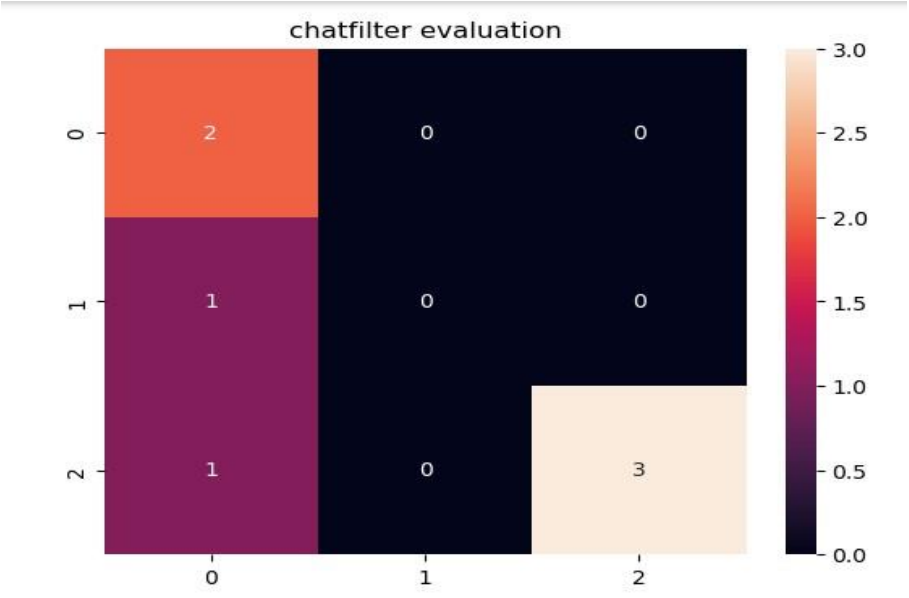
Instead of SMOTE, I'm considering using Support Vector Machines (SVM)[9], which is a powerful approach for mapping data points into higher-dimensional spaces to enable better classification or regression. In particular, I'll be using an SVC (Support Vector Classifier), which maps data points into a high-dimensional space and identifies hyperplanes that maximize the margin between different classes. After we get more classify data as better than before we using 'K-Nearest Neighbor' machine learning model.

---

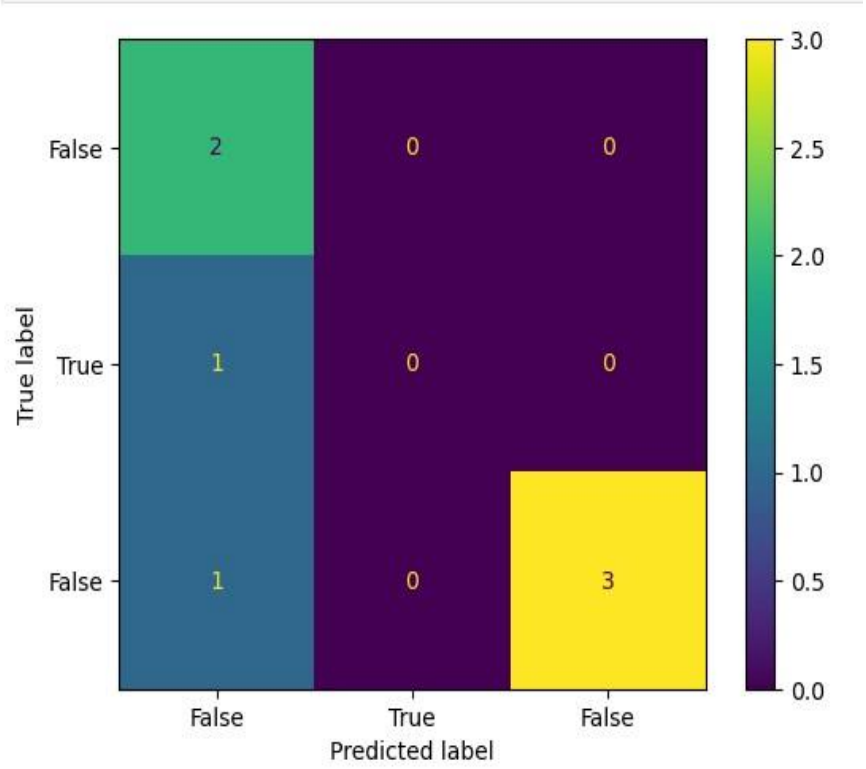
[8] 'SMOTE' is measures of oversampling, extended dataset for solve the data imbalance.

[9] Referance: https://ice-ice-bear.github.io/machine-learning/%ED%98%BC%EA%B3%B5%EB%A8%B8%EC%8B%A0-3-1-3/

New chat filter evaluation-



	precision	recall	f1-score	support
80	0.50	1.00	0.67	2
85	0.00	0.00	0.00	1
100	1.00	0.75	0.86	4
accuracy			0.71	7
macro avg	0.50	0.58	0.51	7
weighted avg	0.71	0.71	0.68	7



-

-Class 0: False-

TP (True Positive): Cases correctly predicted as Class 0 → 2

FP (False Positive): Other classes incorrectly predicted as Class 0 → 1 + 1 = 2

FN (False Negative): Class 0 incorrectly predicted as another class → 0

TN (True Negative): Other classes (Class 1 and 2) correctly excluded → 0 + 3 = 3

$$\text{Precision} = \frac{2}{2+2} = 0.5$$

$$\text{Recall} = \frac{2}{2+0} = 1.0$$

$$\text{F1-score: } \frac{2(Precision \times recall)}{(Precision + recall)} \quad \frac{2(0.5 \times 1.0)}{(0{,}5+1.0)} = = 0.6666 = 0.7$$

-Class 1: True –

TP: Cases correctly predicted as Class 1 → 0

FP: Other classes incorrectly predicted as Class 1 → 0

FN: Class 1 incorrectly predicted as another class → 1

TN: Other classes (Class 0 and 2) correctly excluded → 2 + 3 = 5

$$\text{Precision} = \frac{0}{0+0} = 0$$

$$\text{Recall} = \frac{0}{0+1} = 0$$

$$\frac{2(Precision \times recall)}{} \quad \frac{2(0 \times 0)}{}$$

F1-score: $\frac{}{(Precision + recall)} \quad \frac{}{(0+0)} = = 0$

Class 2: False-

TP: Cases correctly predicted as Class 2 → 3

FP: Other classes incorrectly predicted as Class 2 → 0

FN: Class 2 incorrectly predicted as another class → 1

TN: Other classes (Class 0 and 1) correctly excluded → 2 + 0 = 2

Precision=$\frac{3}{3+0}$ = 1

Recall =$\frac{3}{3+1}$ = 0.75

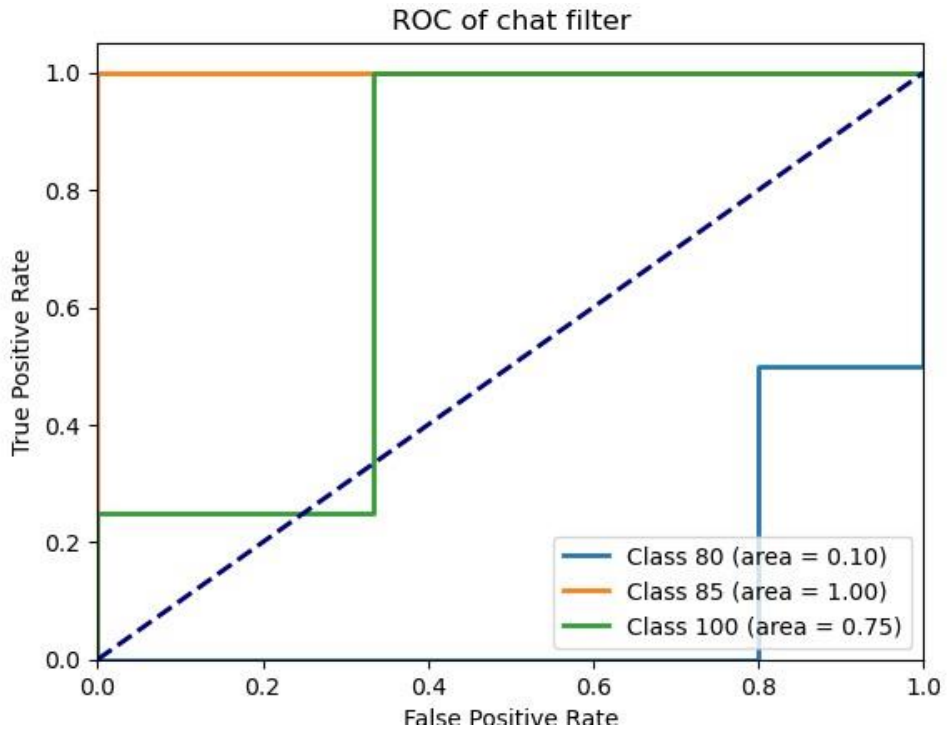
F1-score: $\frac{2(Precision \times recall)}{(Precision + recall)} \quad \frac{2(1.0 \times 0.75)}{(1.0+0.75)} = = 0.857$

-Overall accuracy of chat filter-

Accuracy: $\frac{2+0+3+3}{7} = \frac{8}{7} \approx 0.71$ (i.e., 71%)

-New AUC Roc of the chat filter-

-



Class 80 (Blue Line, AUC = 0.10):

The ROC curve for Class 80 has a very small area under it, which means the model performs poorly in distinguishing this class. An AUC of 0.10 is far below the optimal value, indicating that it is almost as bad as random guessing.

Class 85 (Orange Line, AUC = 1.00):

The ROC curve for Class 85 shows a perfect horizontal line at the top of the graph (True Positive Rate = 1.0), which means the model perfectly classifies all samples in this class. An AUC of 1.00 indicates a perfect classifier with no errors for this specific class.

Class 100 (Green Line, AUC = 0.75):

The ROC curve for Class 100 shows a reasonable shape with an AUC value of 0.75. This indicates that the model has a fair capability of distinguishing this class. While it is not perfect, it's performing above the random chance level (0.5), which means it is making meaningful predictions for this class.

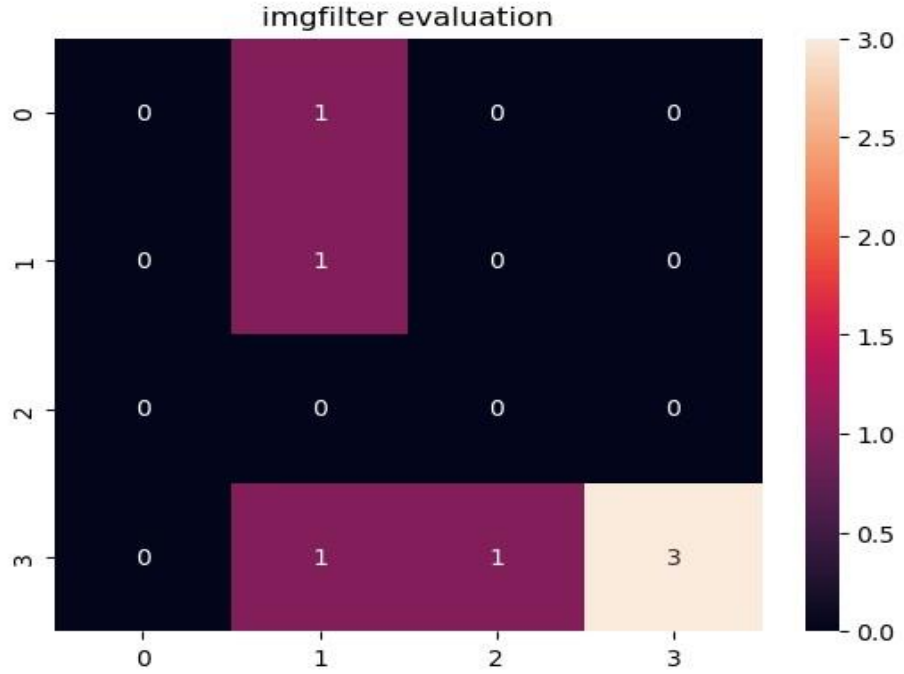
Class 85: Perfect classification, no false positives or negatives.

Class 100: Above average classification, reasonable performance.

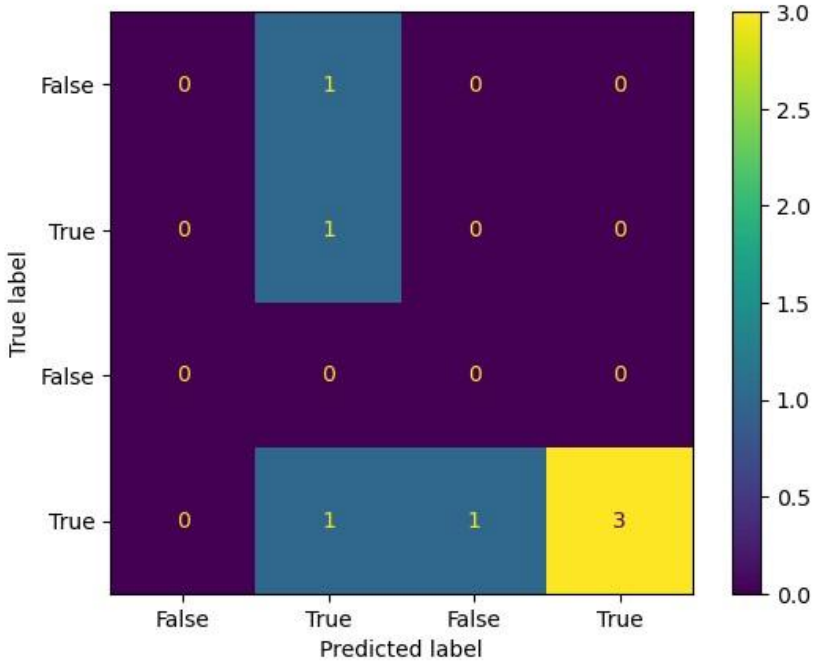
Class 80: Poor performance, close to random guessing.

New image filter evaluation-

-



	precision	recall	f1-score	support
70	0.00	0.00	0.00	1
80	0.33	1.00	0.50	1
90	0.00	0.00	0.00	0
100	1.00	0.60	0.75	5
accuracy			0.57	7
macro avg	0.33	0.40	0.31	7
weighted avg	0.76	0.57	0.61	7



Class False 0

TP (True Positive): Cases correctly predicted as Class 70 → 0

FP (False Positive): Other classes incorrectly predicted as Class 70 → 0

FN (False Negative): Class 70 incorrectly predicted as another class → 1

TN (True Negative): Remaining correctly classified as not Class 70 → 1 + 0 + 1 + 3 = 5

Precision=$\frac{0}{0+0}$ = 0

Recall =$\frac{0}{0+1}$=0

F1-score: $\frac{2(Precision \times recall)}{(Precision + recall)}$ $\frac{2(0 \times 0)}{(0 + 0)}$ = = 0

Class True 1

TP: Cases correctly predicted as Class 80 → 1

FP: Other classes incorrectly predicted as Class 80 → 1

FN: Class 80 incorrectly predicted as another class → 0

TN: Remaining correctly classified as not Class 80 → 0 + 0 + 1 + 3 = 4

Precision=$\frac{1}{1+0}$ = 0.5

Recall =$\frac{1}{1+0}$=1.0

F1-score: $\frac{2(Precision \times recall)}{(Precision + recall)}$ $\frac{2(0.5 \times 1.0)}{(0.5 + 1.0)}$ = = 0.6666=0.7

-

Class False 2

TP: Cases correctly predicted as Class 90 → 0

FP: Other classes incorrectly predicted as Class 90 → 1

FN: Class 90 incorrectly predicted as another class → 0

TN: Remaining correctly classified as not Class 90 → 0 + 1 + 1 + 3 = 5

Precision=$\frac{0}{0+1}$ = 0.0

Recall =$\frac{0}{0+0}$= 0.0

F1-score: $\frac{2(Precision \times recall)}{(Precision + recall)}$ $\frac{2(0 \times 0)}{(0 + 0)}$ = = 0

Class True 3

TP: Cases correctly predicted as Class 100 → 3

FP: Other classes incorrectly predicted as Class 100 → 0

FN: Class 100 incorrectly predicted as another class → 2

TN: Remaining correctly classified as not Class 100 → 1 + 1 + 0 = 2

Precision=$\frac{3}{3+1}$ = 0.75

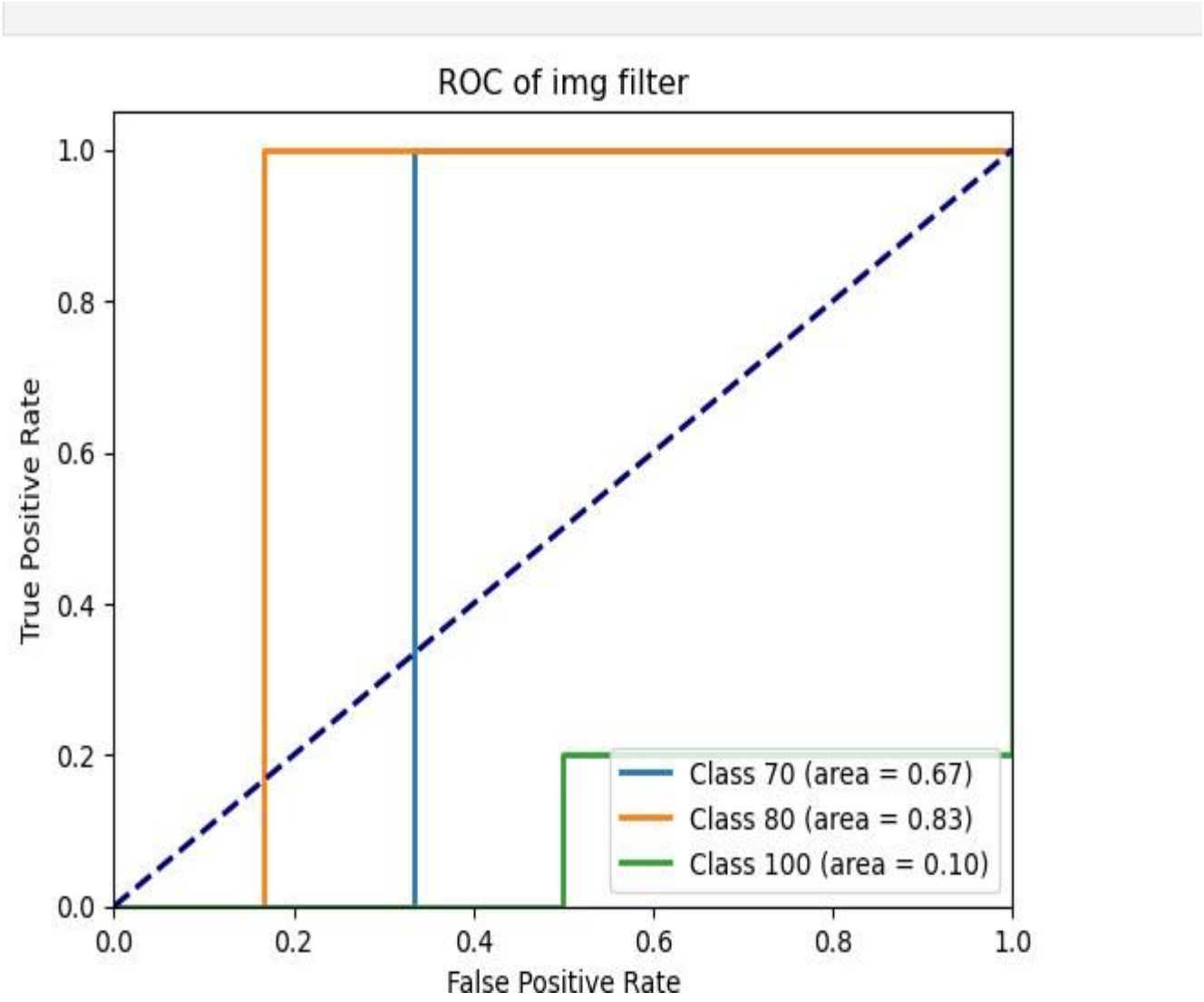
Recall =$\frac{3}{3+2}$= 0.6

F1-score: $\frac{2(Precision \times recall)}{(Precision + recall)}$ $\frac{2(0.75 \times 0.6)}{(0.75 + 0.6)}$ = = 0.666 =0.7

-Overall accuracy of image filter-

Accuracy: $\frac{0+1+0+3}{7} = \frac{4}{7} \approx 0.57$ (i.e., 57%)

-New AUC Roc of image filter-



Class 70 (Blue Line, AUC = 0.67):

The ROC curve for Class 70 is quite steep initially but has a limited area under it. An AUC of 0.67 suggests the model performs moderately well for distinguishing Class 70 from the other classes. It is better than random guessing (AUC of 0.5), but there's still significant room for improvement.

Class 80 (Orange Line, AUC = 0.83):

The ROC curve for Class 80 has a near-vertical rise, reaching a True Positive Rate (TPR) of 1.0 at a relatively low False Positive Rate (FPR). The AUC value of 0.83 indicates that the model does a good job distinguishing this class from the others. This is significantly better than random guessing and suggests relatively strong performance for this class.

Class 100 (Green Line, AUC = 0.10):

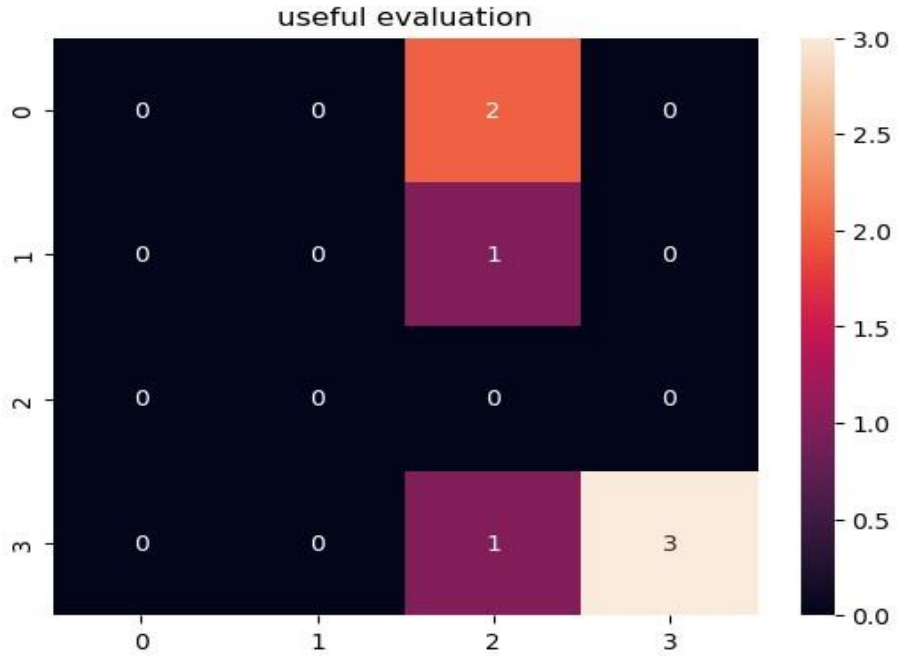
The ROC curve for Class 100 stays low along the y-axis, and the AUC value of 0.10 is very poor, indicating that the model struggles to classify Class 100 effectively. The performance is well below the random guess level, implying that it has serious issues in correctly predicting this class.

Class 80 has the best performance (AUC = 0.83), indicating a good discriminative ability.

Class 70 shows moderate performance (AUC = 0.67), and improvements are possible.

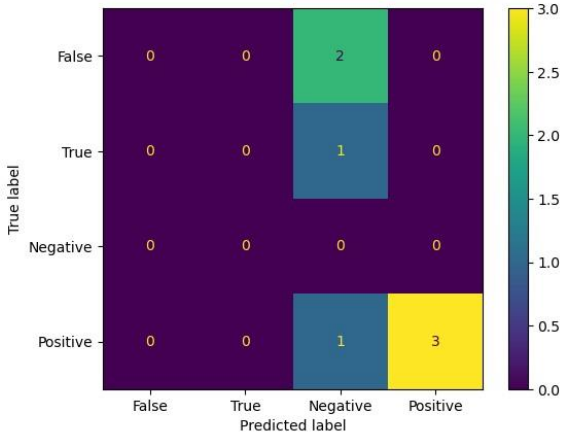
Class 100 shows poor performance (AUC = 0.10), requiring significant attention to improve model effectiveness.

-New useful evaluation-



	precision	recall	f1-score	support
80	0.00	0.00	0.00	2
82	0.00	0.00	0.00	1
90	0.00	0.00	0.00	0
100	1.00	0.75	0.86	4
accuracy			0.43	7
macro avg	0.25	0.19	0.21	7
weighted avg	0.57	0.43	0.49	7

[18]:
```
#useful evaluation
import numpy as np
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

from sklearn import metrics
confusion_matrix = metrics.confusion_matrix(useful_output_test, useful_pred)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True, 'Negative', 'Positive'])
cm_display.plot()
plt.show()
```



[19]:
```
#Consistency pred
```

Class False 0

TP (True Positive): Cases correctly predicted as Class 0 → 0

FP (False Positive): Other classes incorrectly predicted as Class 0 → 0

FN (False Negative): Class 0 incorrectly predicted as another class → 2

TN (True Negative): Remaining correctly classified as not Class 0 → 1 + 0 + 1 + 3 = 5

$$\text{Precision} = \frac{0}{0+0} = 0.0$$

$$\text{Recall} = \frac{0}{0+2} = 0.0$$

$$\text{F1-score: } \frac{2(Precision \times recall)}{(Precision + recall)} \quad \frac{2(0 \times 0)}{(0 + 0)} = = 0$$

Class True 1

TP: Cases correctly predicted as Class 1 → 1

FP: Cases from other classes incorrectly predicted as Class 1 → 2

FN: Cases from Class 1 incorrectly predicted as other classes → 0

TN: All other correctly classified cases → 0 + 0 + 1 + 3 = 4

$$\text{Precision} = \frac{1}{1+2} = 0.33$$

$$\text{Recall} = \frac{1}{1+0} = 1.0$$

$$\text{F1-score: } \frac{2(Precision \times recall)}{(Precision + recall)} \quad \frac{2(0.33 \times 1.0)}{(0.33 + 1.0)} = = 5.0$$

Class Negative 2

TP: Cases correctly predicted as Class 2 → 0

FP: Cases from other classes incorrectly predicted as Class 2 → 1

FN: Cases from Class 2 incorrectly predicted as other classes → 0

TN: All other correctly classified cases → 0 + 2 + 1 + 3 = 6

Precision = $\frac{0}{0+1}$ = 0.0

Recall = $\frac{0}{0+0}$ = 0.0

F1-score: $\frac{2(Precision \times recall)}{(Precision + recall)}$ $\frac{2(0.0 \times 0.0)}{(0.0 + 0.0)}$ = = 0.0

Class Positive 3

TP: Cases correctly predicted as Class 3 → 3

FP: Cases from other classes incorrectly predicted as Class 3 → 1

FN: Cases from Class 3 incorrectly predicted as other classes → 1

TN: All other correctly classified cases → 2 + 1 + 0

Precision = $\frac{3}{3+1}$ = 0.75

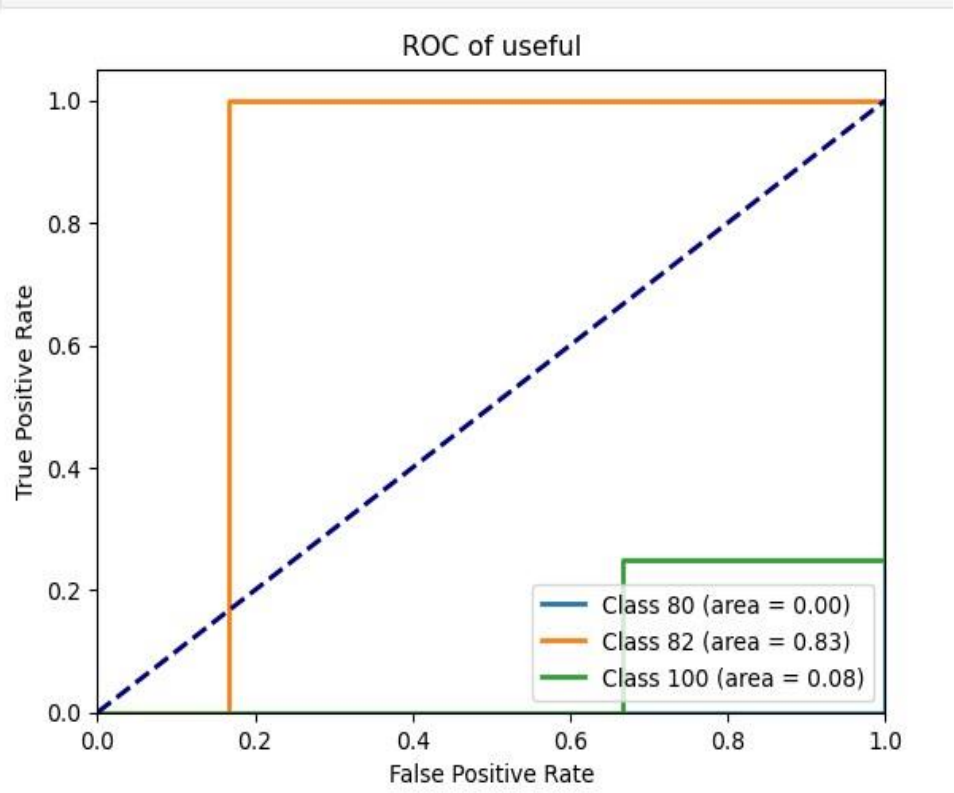
Recall = $\frac{3}{3+1}$ =0.75

F1-score: $\frac{2(Precision \times recall)}{}$ $\frac{2(0.75 \times 0.75)}{}$ = = 0.75

$(Precision + recall)$ $(0.75 + 0.75)$

Overall Accuracy Calculation

$$\text{Accuracy} = \frac{0+1+0+3+4}{7} = \frac{4}{7} = 0.57 \text{ (i.e., 57\%)}$$



Class 80 (AUC = 0.00):

The model completely fails to distinguish this class. Essentially, the model is making entirely incorrect predictions for this class. Additional attention might be needed for data quality, more data quantity, or better features for this class.

Class 82 (AUC = 0.83):

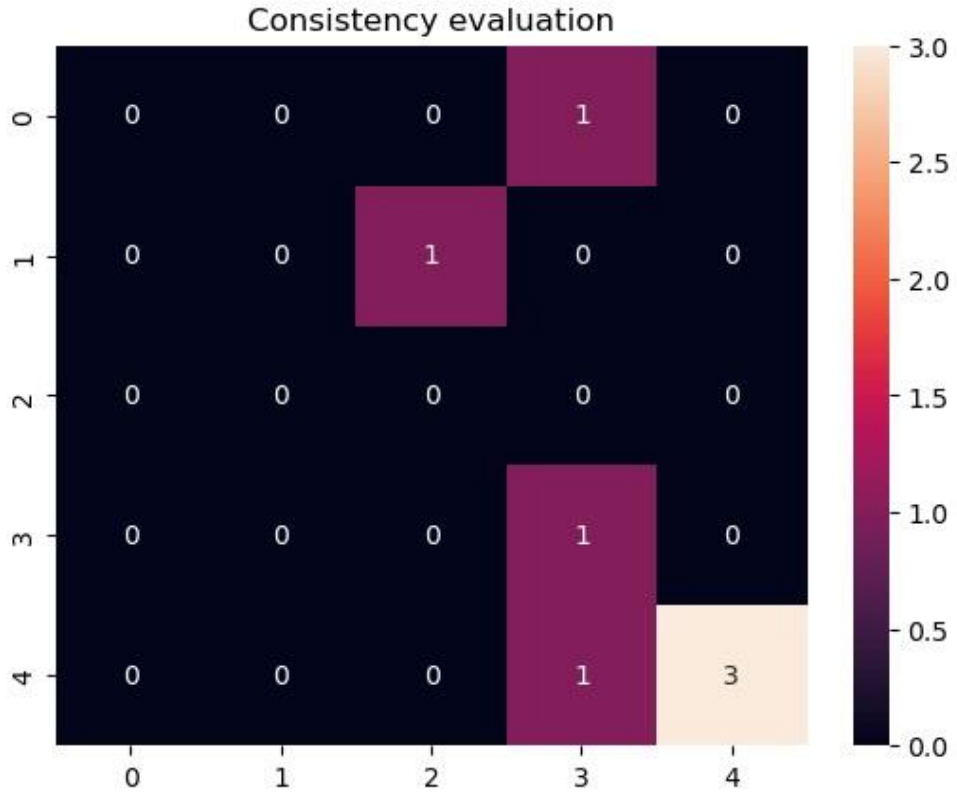
The model performs well for this class, with an AUC of 0.83. This means the model effectively separates positive cases of Class 82 from the negatives. This performance is satisfactory, though there is still some room for improvement.

Class 100 (AUC = 0.08):

An AUC of 0.08 suggests that the model performs extremely poorly for this class. There is almost no capability of distinguishing positive instances for Class 100. This likely indicates a lack of training data, data imbalance, or the need for improved feature engineering.

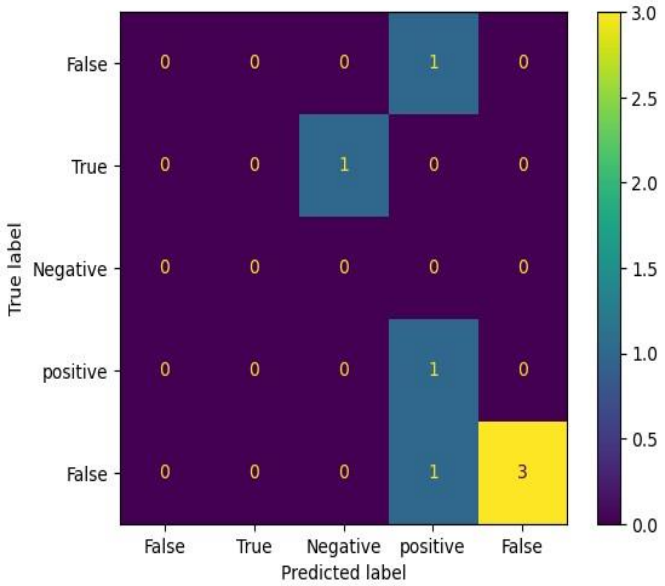
-New Consistency evaluation-

```
#Consistency evaluation
cm = metrics.confusion_matrix(Consistency_output_test,Consistency_pred)
ax= plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax);
ax.set_title('Consistency evaluation');
plt.show()
mat = metrics.classification_report(Consistency_output_test, Consistency_pred)
print(mat)
```



	precision	recall	f1-score	support
0.0	0.00	0.00	0.00	1
79.0	0.00	0.00	0.00	1
89.0	0.00	0.00	0.00	0
90.0	0.33	1.00	0.50	1
100.0	1.00	0.75	0.86	4
accuracy			0.57	7
macro avg	0.27	0.35	0.27	7
weighted avg	0.62	0.57	0.56	7

[41]:
```
#Consistency evaluation
import numpy as np
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

from sklearn import metrics
confusion_matrix = metrics.confusion_matrix(Consistency_output_test,Consistency_pred)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True, 'Negative', 'positive', False])
cm_display.plot()
plt.show()
```



[42]: `#RespondSpeed_pred`

Class False (0)

TP (True Positive): Cases correctly predicted as "False" → 0

FP (False Positive): Other classes incorrectly predicted as "False" → 0

FN (False Negative): "False" cases incorrectly predicted as another class → 1

TN (True Negative): All other correctly classified cases → 1+1+1+3=6

$$\text{Precision} = \frac{0}{0+0} = 0$$

$$\text{Recall} = \frac{0}{0+1} = 0$$

$$\text{F1-score: } \frac{2(Precision \times recall)}{(Precision + recall)} \quad \frac{2(0 \times 0)}{(0+0)} = = 0$$

Class True (1)

TP: Cases correctly predicted as "True" → 1

FP: Other classes incorrectly predicted as "True" → 0

FN: "True" cases incorrectly predicted as another class → 0

TN: All other correctly classified cases → 0+0+1+3=4

Precision = $\frac{1}{1+0} = 1.0$

Recall = $\frac{1}{1+0} = 1.0$

F1-score: $\frac{2(Precision \times recall)}{(Precision + recall)} \quad \frac{2(1.0 \times 1.0)}{(1.0 + 1{,}0)} = = 1.0$

Class Negative (2)

TP: Cases correctly predicted as "Negative" → 0

FP: Other classes incorrectly predicted as "Negative" → 1

FN: "Negative" cases incorrectly predicted as another class → 0

TN: All other correctly classified cases → 0+1+1+3=5

Precision = $\frac{0}{0+1} = 0$

Recall = $\frac{0}{0+0} = 0$

F1-score: $\frac{2(Precision \times recall)}{(Precision + recall)} \quad \frac{2(0 \times 0)}{(0 + 0)} = = 0$

Class Positive (3)

TP: Cases correctly predicted as "Positive" → 1

FP: Other classes incorrectly predicted as "Positive" → 0

FN: "Positive" cases incorrectly predicted as another class → 1

TN: All other correctly classified cases → 1+0+1+3=5

$$\text{Precision} = \frac{1}{1+0} = 1.0$$

$$\text{Recall} = \frac{1}{1+1} = 0.5$$

$$\text{F1-score: } \frac{2(Precision \times recall)}{(Precision + recall)} \quad \frac{2(1.0 \times 0.5)}{(1.0 + 0.5)} = = 0.67$$

$$\text{Overall accuracy} = \frac{\frac{0+1+0+1+3}{7}}{} = \frac{5}{7} = 0.71 = 71\%$$

Class "True":

This class is predicted perfectly, with a precision, recall, and F1-score of 1.0.

Class "Positive":

This class has good precision but moderate recall, indicating some missed predictions.

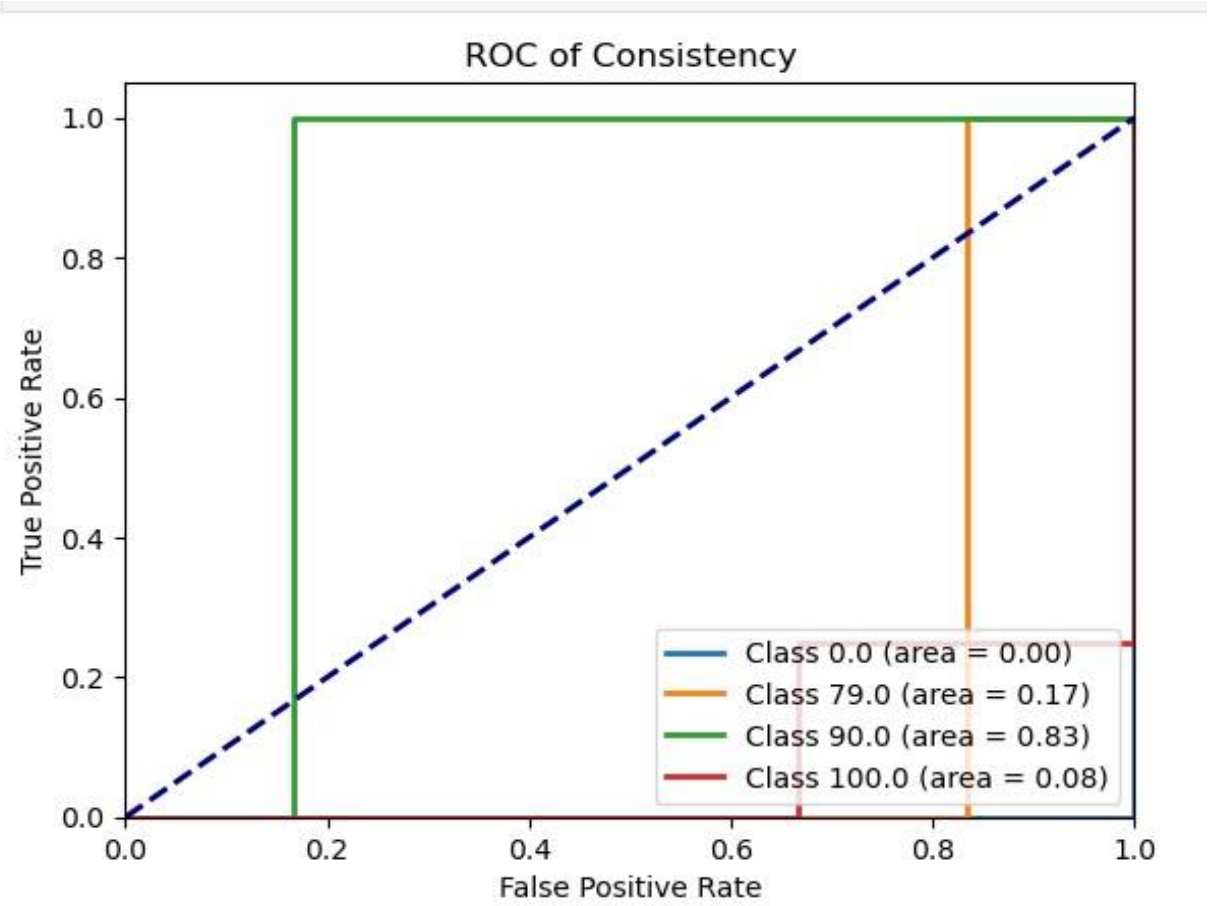
Class "False":

This class is completely missed, with no true positives or precision/recall.

Class "Negative":

No predictions or actual instances for this class, so performance is 0.

-New AUC-Roc of Consistency-



Class 0.0 (AUC = 0.00):

The model is completely ineffective for Class 0.0, as indicated by the AUC of 0.00. There are no correct classifications, which may suggest issues with data imbalance, insufficient training data, or inadequate features for this class.

Class 79.0 (AUC = 0.17):

The model struggles with this class, achieving an AUC of 0.17. This indicates poor performance, only slightly better than random guessing. Further investigation may be needed to determine why the model cannot effectively learn the characteristics of this class.

Class 90.0 (AUC = 0.83):

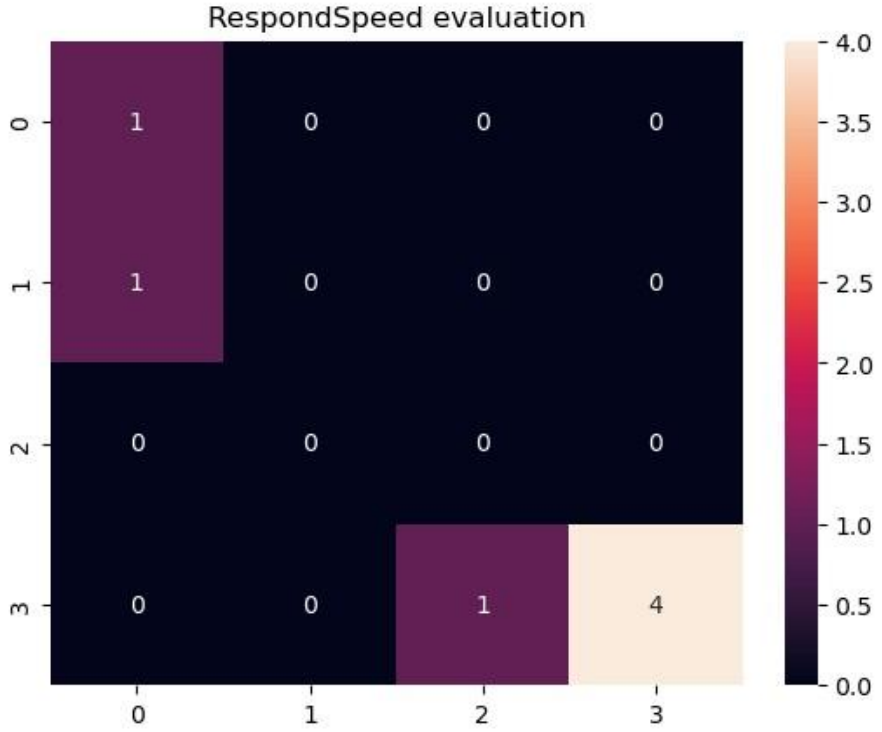
This class shows good model performance, with an AUC of 0.83. The model is effective at distinguishing Class 90.0 from other classes, suggesting that the features used for this class are well-suited for classification and that the model is learning well.

Class 100.0 (AUC = 0.08):

An AUC of 0.08 suggests that the model's performance for Class 100.0 is almost as poor as it can get. The model is unable to effectively distinguish positives from negatives, which could indicate the need for better feature engineering or more balanced data for this class.

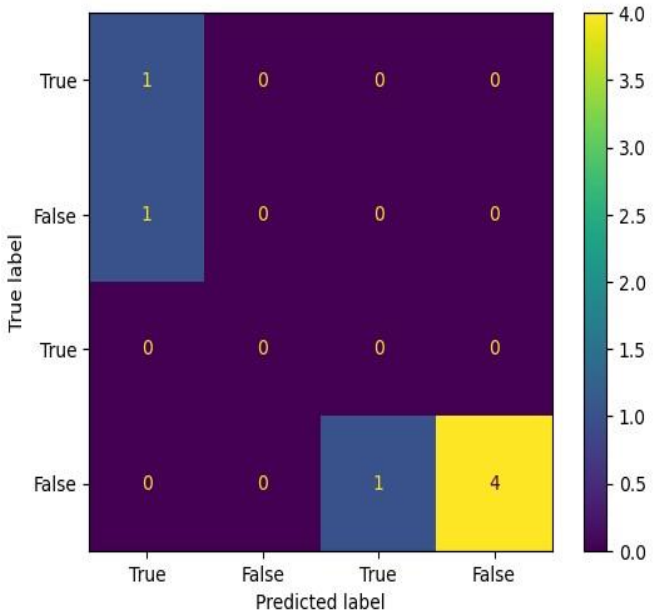
-New Respond Speed evaluation-

```
]: #RespondSpeed evaluation
cm = metrics.confusion_matrix(RespondSpeed_output_test,RespondSpeed_pred)
ax= plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax);
ax.set_title('RespondSpeed evaluation');
plt.show()

mat = metrics.classification_report(RespondSpeed_output_test, RespondSpeed_pred)
print(mat)
```




```
              precision    recall  f1-score   support

          80       0.50      1.00      0.67         1
          85       0.00      0.00      0.00         1
          99       0.00      0.00      0.00         0
         100       1.00      0.80      0.89         5

    accuracy                           0.71         7
   macro avg       0.38      0.45      0.39         7
weighted avg       0.79      0.71      0.73         7
```

[45]:
```
#RespondSpeed evaluation
import numpy as np
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

from sklearn import metrics
confusion_matrix = metrics.confusion_matrix(RespondSpeed_output_test, RespondSpeed_pred)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [True, False, True, False])
cm_display.plot()
plt.show()
```



Class True (0):

TP (True Positive): Correctly predicted as Class 0 → 1

FP (False Positive): Other classes incorrectly predicted as Class 0 → 1

FN (False Negative): Class 0 incorrectly predicted as another class → 0

TN (True Negative): Correctly classified as not Class 0 → 0+0+1+4=5

Precision = $\frac{1}{1+1}$ = 0.5

Recall = $\frac{1}{1+0} = 1.00$

F1-score: $\frac{2(Precision \times recall)}{(Precision + recall)} \quad \frac{2(0.5 \times 1.00)}{(0.5+1.00)} = = 0.67$

Class False (1):

TP: Correctly predicted as Class 1 → 0

FP: Other classes incorrectly predicted as Class 1 → 0

FN: Class 1 incorrectly predicted as another class → 1

TN: Correctly classified as not Class 1 → 1+0+1+4=6

Precision = $\frac{0}{0+0} = 0$

Recall = $\frac{0}{0+1} = 0$

F1-score: $\frac{2(Precision \times recall)}{(Precision + recall)} \quad \frac{2(0 \times 0)}{(0+0)} = = 0$

Class True (2):

TP: Correctly predicted as Class 2 → 0

FP: Other classes incorrectly predicted as Class 2 → 1

FN: Class 2 incorrectly predicted as another class → 0

TN: Correctly classified as not Class 2 → 1+0+0+4=5

Precision = $\frac{0}{0+1} = 0$

$$\text{Recall} = \frac{0}{0+0} = 0$$

$$\text{F1-score: } \frac{2(Precision \times recall)}{(Precision + recall)} \qquad \frac{2(0\times0)}{(0+0)} = = 0$$

Class False (3):

TP: Correctly predicted as Class 3 → 4

FP: Other classes incorrectly predicted as Class 3 → 1

FN: Class 3 incorrectly predicted as another class → 1

TN: Correctly classified as not Class 3 → 1+0+0=3

$$\text{Precision} = \frac{4}{4+1} = 0.80$$

$$\text{Recall} = \frac{4}{4+1} = 0.80$$

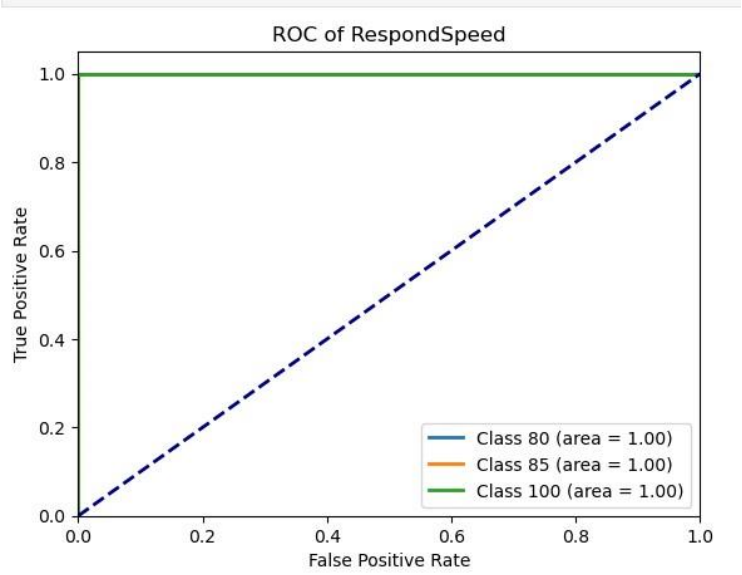
$$\text{F1-score: } \frac{2(Precision \times recall)}{(Precision + recall)} \qquad \frac{2(0.80\times0.80)}{(0.80+0.80)} = = 0.89$$

$$\text{Overall Accuracy} = \frac{1+6+5+3=15}{7} = 0.71 = 71\%$$

The ROC curves for Class 80, Class 85, and Class 100 all have an AUC of 1.00, which implies that the model is perfectly classifying all instances for these classes.

The performance is ideal in this ROC evaluation, and no improvements are required for the current dataset.

-New AUC-roc of Respond Speed-



Class 80 (Blue Line, AUC = 1.00):

The ROC curve for Class 80 reaches the top of the graph at a True Positive Rate of 1.0, which means that the model correctly classified all positive instances without any false negatives. An AUC of 1.00 indicates perfect classification performance for this class, meaning the model can perfectly distinguish between positive and negative instances.

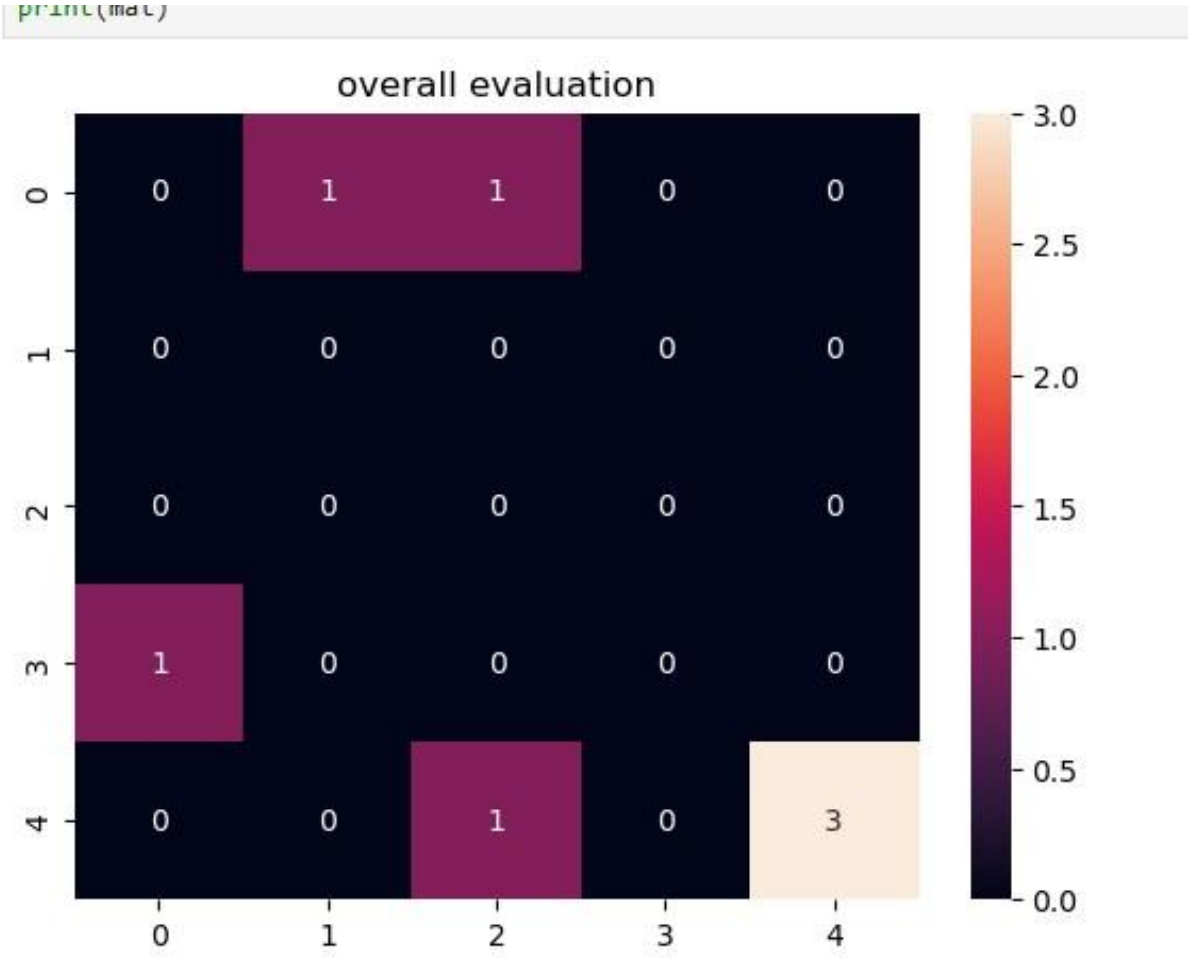
Class 85 (Orange Line, AUC = 1.00):

The ROC curve for Class 85 also reaches the top of the graph, showing a True Positive Rate of 1.0 and indicating perfect classification. An AUC of 1.00 means that the model has achieved perfect performance in predicting instances for Class 85.

Class 100 (Green Line, AUC = 1.00):

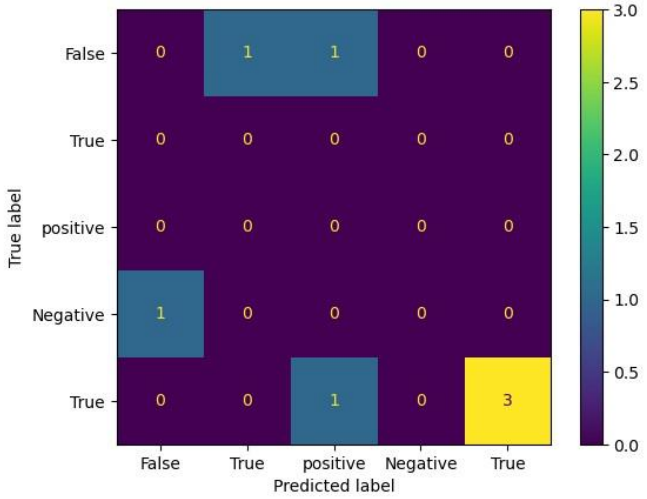
Similarly, the ROC curve for Class 100 shows a True Positive Rate of 1.0 at a very low False Positive Rate, indicating that the model has perfectly distinguished between positive and negative instances for this class. The AUC value of 1.00 again indicates flawless performance.

-New overall satisfaction evaluation-



	precision	recall	f1-score	support
80	0.00	0.00	0.00	2
89	0.00	0.00	0.00	0
90	0.00	0.00	0.00	0
95	0.00	0.00	0.00	1
100	1.00	0.75	0.86	4
accuracy			0.43	7
macro avg	0.20	0.15	0.17	7
weighted avg	0.57	0.43	0.49	7

```
53]: #overall evaluation
import numpy as np
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

from sklearn import metrics
confusion_matrix = metrics.confusion_matrix(overallevaluation_output_test, overallevaluation_pred)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True, "positive", "Negative", True])
cm_display.plot()
plt.show()
```



Class False (0):

TP (True Positive): Correctly predicted as Class 0 → 0

FP (False Positive): Other classes incorrectly predicted as Class 0 → 1

FN (False Negative): Class 0 incorrectly predicted as another class → 2

TN (True Negative): Correctly classified as not Class 0 → 0+0+0+1+0+3=4

$$\text{Precision} = \frac{0}{0+1} = 0$$

$$\text{Recall} = \frac{0}{0+2} = 0$$

$$2(Precision \times recall) \qquad 2(0\times0)$$

F1-score: $\frac{}{(Precision + recall)} \quad \frac{}{(0+0)} = = 0$

Class True (1):

TP: Correctly predicted as Class 1 → 0

FP: Other classes incorrectly predicted as Class 1 → 1

FN: Class 1 incorrectly predicted as another class → 0

TN: Correctly classified as not Class 1 → 0+0+1+1+3=5

Precision = $\frac{0}{0+1} = 0$

Recall = $\frac{0}{0+2} = 0$

F1-score: $\frac{2(Precision \times recall)}{(Precision + recall)} \quad \frac{2(0\times0)}{(0+0)} = = 0$

Class Positive (2):

TP: Correctly predicted as Class 2 → 0

FP: Other classes incorrectly predicted as Class 2 → 2

FN: Class 2 incorrectly predicted as another class → 0

TN: Correctly classified as not Class 2 → 0+0+1+1+3=5

Precision = $\frac{0}{0+2}$ = 0.00

Recall = $\frac{0}{0+0}$ = 0.00

F1-score: $\frac{2(Precision \times recall)}{(Precision + recall)}$ $\frac{2(0\times0)}{(0+0)}$ = = 0

Class Negative (3):

TP: Correctly predicted as Class 3 → 0

FP: Other classes incorrectly predicted as Class 3 → 0

FN: Class 3 incorrectly predicted as another class → 1

TN: Correctly classified as not Class 3 → 1+0+2+3=6

Precision = $\frac{0}{0+0}$ = 0

Recall = $\frac{0}{0+1}$ = 0

F1-score: $\frac{2(Precision \times recall)}{(Precision + recall)}$ $\frac{2(0\times0)}{(0+0)}$ = = 0

Class True (4):

TP: Correctly predicted as Class 4 → 3

FP: Other classes incorrectly predicted as Class 4 → 0

FN: Class 4 incorrectly predicted as another class → 1

TN: Correctly classified as not Class 4 → 1+0+2+0=3

Precision = $\frac{3}{3+0}$ = 1.00

Recall = $\frac{3}{3+1} = 0.75$

F1-score: $\frac{2(Precision \times recall)}{(Precision + recall)}$ $\frac{2(1.00\times0.75)}{(1.00+0.75)} = = 0.857 = 0.86$

Overall accuracy = $\frac{0+5+5+6+3=19}{7}$ = 0.43 = 43%

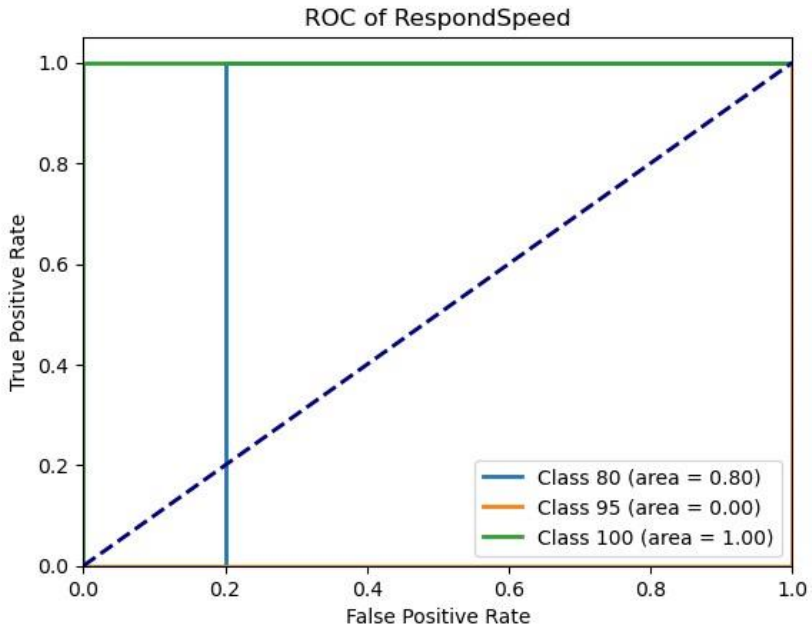
Overall Metrics:

Accuracy: 43%

Macro Average: Precision = 0.20, Recall = 0.15, F1-Score = 0.17

Weighted Average: Precision = 0.57, Recall = 0.43, F1-Score = 0.49

```
svm.fit(overallevaluation_input_test, overallevaluation_output_test)

# Binarize the labels for one-vs-all computation
y_test_binarized = label_binarize(overallevaluation_output_test, classes=classes)
y_score = svm.predict_proba(overallevaluation_input_test)  # Assuming `svm` is your

# Plot the ROC curve for each class
plt.figure()
for i in range(n_classes):
    fpr, tpr, _ = roc_curve(y_test_binarized[:, i], y_score[:, i])
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, lw=2, label=f'Class {classes[i]} (area = {roc_auc:.2f})')

# Plot settings
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC of RespondSpeed')
plt.legend(loc="lower right")
plt.show()
```



Class 80 (AUC = 0.80):

The model performs quite well for Class 80, with an AUC of 0.80. This means that the model correctly identifies a high proportion of true positives while keeping false positives relatively low. However, there is still room for improvement to increase the accuracy to closer to 1.00.

Class 95 (AUC = 0.00):

The model fails completely for Class 95, as indicated by the AUC of 0.00. This suggests that the model is making incorrect predictions for nearly all instances of this class. Improvements could include rebalancing the dataset or adding more meaningful features.

Class 100 (AUC = 1.00):

Class 100 is perfectly classified by the model, with an AUC of 1.00. This indicates that the model is able to distinguish all positive samples correctly with no false positives, reflecting an ideal performance.

Class 100 shows perfect classification performance (AUC = 1.00), meaning the model predicts every instance correctly without any errors.

Class 80 has a good performance (AUC = 0.80), indicating the model's effectiveness in correctly identifying this class with a relatively high accuracy.

Class 95 performs very poorly (AUC = 0.00), suggesting that significant improvements are needed to help the model distinguish this class effectively.

-Pros and Cons of KNN and SVM-

-K-Nearest Neighbors (KNN)-

Pros:

1. Simplicity: KNN is straightforward to implement and easy to understand.

2. No Training Phase: KNN is a lazy learning algorithm, meaning it does not have an explicit training phase, making it useful for quick evaluations.

3. Effective for Small, Balanced Datasets: Performs well when the dataset is small and balanced.

4. Flexible Distance Metrics: Can use different distance metrics (e.g., Euclidean, Manhattan) to suit different types of data.

Cons:

1. Sensitive to Noise: KNN can be significantly affected by noise, leading to frequent misclassifications.

2. Computationally Expensive: Requires calculating the distance to all training points for each prediction, making it inefficient for large datasets.

3. Data Scaling Requirement: Sensitive to feature scales, necessitating feature normalization or scaling.

4. Poor Performance with Imbalanced Data: Heavily influenced by dominant classes, resulting in higher rates of false positives and false negatives.

-Support Vector Machine (SVM)-

Pros:

1. Effective for High-Dimensional Data: SVM works well with datasets that have a large number of features.

2. Handles Non-Linear Boundaries: The use of kernel functions allows SVM to effectively handle non-linear classification tasks.

3. Maximizes Margin: SVM focuses on maximizing the margin between classes, which can lead to better generalization.

4. Less Sensitive to Overfitting: Especially with appropriate kernel and regularization parameters.

Cons:

1. High Training Time: Training an SVM can be computationally expensive, particularly for large datasets.

2. Not Suitable for Very Large Datasets: The training complexity makes SVM less practical for extremely large datasets.

3. Limited Probability Estimates: SVM does not provide direct probability estimates without additional techniques like Platt Scaling.

Contribution

1) Promoting a Cleaner Community Culture:

By filtering out harmful and toxic content on social media, we are contributing to making online communities safer and healthier. This moderation can help reduce the spread of hate speech, bullying, and inappropriate content, leading to more positive user interactions.

2) Advancing Understanding of Classification Techniques:

Our study and implementation of models like KNN and SVM helped deepen our understanding of how different algorithms perform on real-world data. This provides

valuable insights not only for us but also for others in the field who may benefit from our findings, particularly regarding which models are more effective under various conditions.

3) Gathering User Feedback for Improvement:

By conducting surveys and collecting user feedback during the early testing phases, we gained an understanding of how end-users perceive automated moderation. This feedback is crucial for improving the user experience and ensuring that the bot addresses user needs effectively.

4) Personal Skill Development:

On a personal level, working through the challenges of implementing and comparing KNN and SVM has significantly enhanced our technical skills in machine learning, data analysis, and model evaluation. Additionally, developing a functional moderation bot improved our understanding of natural language processing and realtime system integration.

Conclusion

During this project, I developed a censored bot model and deployed it on a Discord server. After collecting survey data from users, I used machine learning models to analyze the data and identify improvement areas. The first model I used was K-nearest neighbor (KNN), which categorized each evaluation score to compare deviations among nearby classes. However, this approach faced issues like underfitting and data imbalance. To address these, I employed a Support Vector Machine (SVM), which effectively mitigated underfitting and provided more detailed feedback for each class. Interestingly, both models highlighted the image filter system

as an area for improvement, with KNN suggesting it needed the most work, while SVM provided more nuanced insights on specific issues.

One of the most difficult parts of the project was splitting the dataset correctly to ensure meaningful results, but I found the data reduction and preprocessing stages to be the most enjoyable and informative. Looking ahead, my future plan focuses on improving the image filter system by exploring advanced techniques and addressing the identified weaknesses.

Overall, this project has been a valuable learning experience in understanding model performance, data handling, and refining machine learning applications for practical use.

Reference

http://essay.utwente.nl/94373/1/Magzoub-BA-EEMCS.pdf

-discord bot censored systems

https://playground.naragara.com

– How to using PIL

혼자 공부하는 머신러닝 + 딥러닝 pg.286

-Deep leanring model for pixel value

(2020)박해선 혼자공부하는 머신 러닝,딥러닝, 서울:한빛미디어 pg:122

-about the over fitting, under fitting problem-

[1] **(Anirudh Verma1, Shashikant Tyagi1, Gauri Mathur2, 2016, International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), india, pg: 533)** -discord word filtering bot-

https://ice-ice-bear.github.io/machine-learning/%ED%98%BC%EA%B3%B5%EB%A8%B8%EC%8B%A0-3-1-3/

-Explain of SVM model-