Bahria University,
Karachi Campus

LAB EXPERIMENT NO.
08

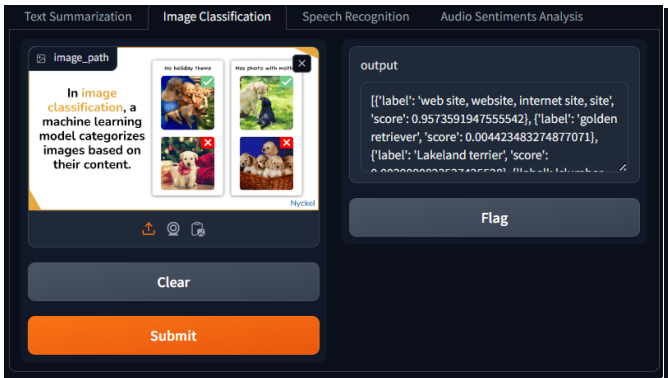
LIST OF TASKS

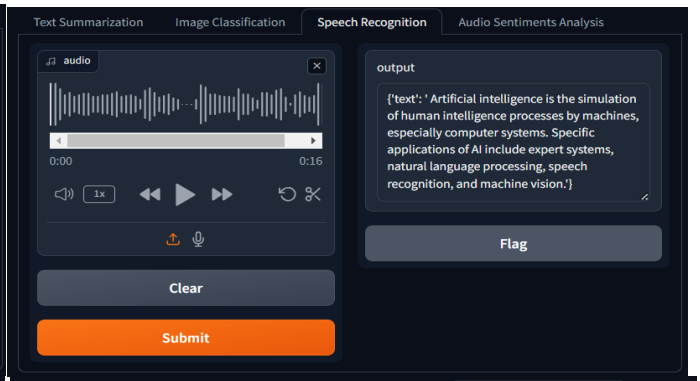
TASK NO	OBJECTIVE
1.	You are tasked with developing a comprehensive GUI-based application that leverages transformer-based models to provide text summarization, image classification, speech recognition, and audio sentiment analysis functionalities in a single application with select model option. The application should offer a seamless user experience and enable users to perform multiple tasks within a single interface. Also, Implement robust error handling mechanisms to handle exceptions gracefully. Display informative error messages to users in case of failures or errors.
2.	You need to develop a user-friendly graphical interface for an Optical Character Recognition (OCR) application using Python's Pytesseract library. •Develop a GUI-based OCR application using Pytesseract. •Allow users to upload images or capture them using a webcam. •Display uploaded or captured images in the application window. •Preprocess images for optimal OCR performance (resize, binarize, etc.). •Extract text from processed images using Pytesseract. •Display extracted text in the application interface. •Provide an option to save extracted text to a file. •Implement error handling for graceful exception handling. •Include tooltips or user guides for easy navigation.

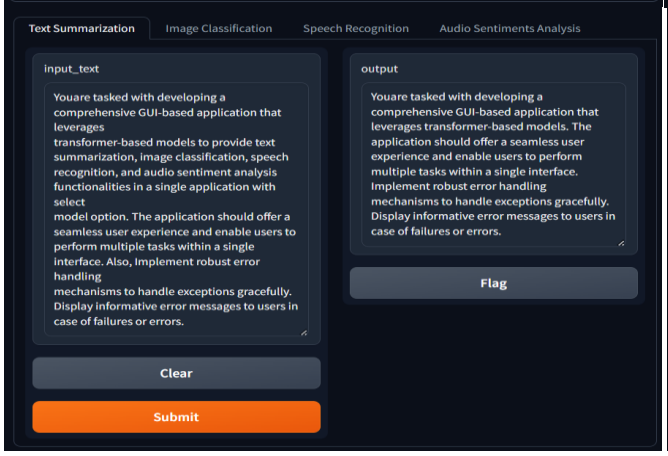
Submitted On:
6/10/2024

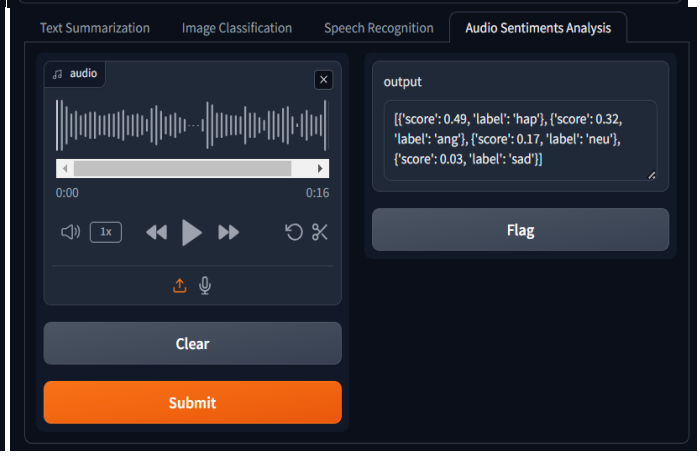
TASK NO 1: GUI-based application that leverages transformer-based models to provide text summarization, image classification, speech recognition, and audio sentiment analysis.

```
import gradio as gr
from transformers import BartForConditionalGeneration,
BartTokenizer
import torch
from transformers import pipeline
import torchvision.transforms as transforms
from PIL import Image
from transformers import ViTForImageClassification,
ViTFeatureExtractor
def generate_summary(input_text):
    model_name = "facebook/bart-large-cnn"
    model =
BartForConditionalGeneration.from_pretrained(model_name)
    tokenizer = BartTokenizer.from_pretrained(model_name)
    input_ids = tokenizer.encode(input_text, return_tensors="pt",
max_length=1024, truncation=True)
    summary_ids = model.generate(input_ids, num_beams=4,
max_length=150, early_stopping=True)
    summary = tokenizer.decode(summary_ids[0],
skip_special_tokens=True)
    return summary
def classify_image(image_path):
    classifier = pipeline (task="image-classification" )
    return classifier(image_path)
def speech_recg(audio):
    transcriber = pipeline (task="automatic-speech-recognition",
model="openai/whisper-small")
    result = transcriber(audio)
    return result
def audio_sent(audio):
    classifier = pipeline (task="audio-classification",
model="superb/hubert-base-superb-er")
    preds = classifier(audio)
    preds = [{"score": round(pred["score"], 2), "label": pred["label"]}
for pred in preds]
    return preds
text_summarization = gr.Interface(fn=generate_summary,
inputs=["text"], outputs=["text"])
image_classification = gr.Interface(fn=classify_image,
inputs=gr.Image(), outputs="text")
speech_recognition = gr.Interface(fn=speech_recg ,
inputs=gr.Audio(), outputs="text")
audio_sentiment_analysis = gr.Interface(fn= audio_sent,
inputs=gr.Audio(), outputs="text")
demo = gr.TabbedInterface([text_summarization,
image_classification,
speech_recognition,audio_sentiment_analysis], ["Text
Summarization ", "Image Classification","Speech
Recognition","Audio Sentiments Analysis"])
if __name__ == "__main__":
    demo.launch()
```









TASK NO 2: You need to develop a user-friendly graphical interface for an Optical Character.

```
import pytesseract
import shutil
```

```
image_path_in_colab= "/content/image-classification.png"
extractedInformation = pytesseract.image_to_string(Image.open(image_path_in_colab))
print(extractedInformation)
```

```
In
,a
machine learning
model categorizes
images based on
their content.

No holiday theme

Has photo with mother
```

Suggested code may be subject to a license | SkafteNicki/dtu_mlops

```
import gradio as gr
from PIL import Image
from transformers import ViTForImageClassification, ViTFeatureExtractor
def imagetotext(image):
  extractedInformation = pytesseract.image_to_string(image)
  return extractedInformation

text_summarization = gr.Interface(fn=imagetotext, inputs=gr.Image(), outputs=["text"])
demo = gr.TabbedInterface([text_summarization], ["Image to text"])

if __name__ == "__main__":
    demo.launch()
```