Title

Generative AI for Humanitarian Cause: Schizophrenia Patient Data Analysis

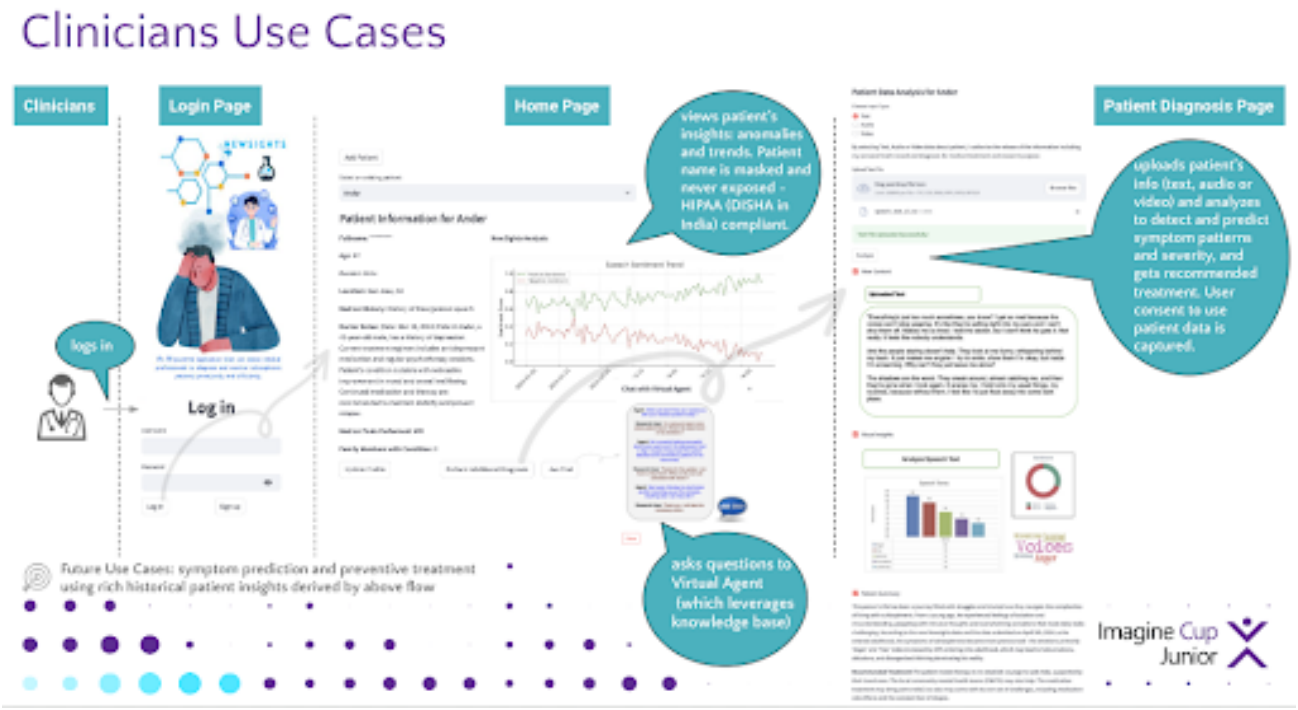
Goal

The primary goal is to develop a Schizophrenia Monitoring & Analysis System that leverages advanced AI technologies to assist medical professionals by providing a more standardized, objective, and efficient method of diagnosing and monitoring schizophrenic patients.
This application will analyze multimodal data inputs - video recordings, speech patterns, and textual information from patient interactions to detect and predict symptom patterns and severity, aiding medical professionals in making informed treatment decisions.

Problem Description

Schizophrenia is a complex and chronic mental health disorder marked by symptoms such as delusions, hallucinations, disorganized thinking and impaired social functioning, often co-occurs with depression, anxiety disorders, and substance abuse.
The symptoms not only significantly impact an individual's functioning of daily life, but also pose a risk of harm to others. Early intervention, comprehensive treatment approaches, and support from healthcare professionals, family, and community resources are essential in addressing the needs of individuals living with schizophrenia, but there is no existing effective solution.

Solution Overview



What is Generative AI ?

Generative AI is a class of artificial intelligence systems that can create new content. This can include text, images, music, and other types of data.

Key Ideas of Generative AI

1. Creating New Stuff: Think of generative AI as an incredibly smart and creative tool. It can write stories, draw pictures, compose music, and even design new products. It’s like having an artist, writer, and inventor all rolled into one, powered by a computer.
2. Learning from Examples: Generative AI learns by studying lots of examples. For instance, if you wanted it to create new paintings, you'd show it thousands of pictures. It analyzes these to understand styles, colors, and shapes, and then uses that knowledge to make its own art.

How It Works

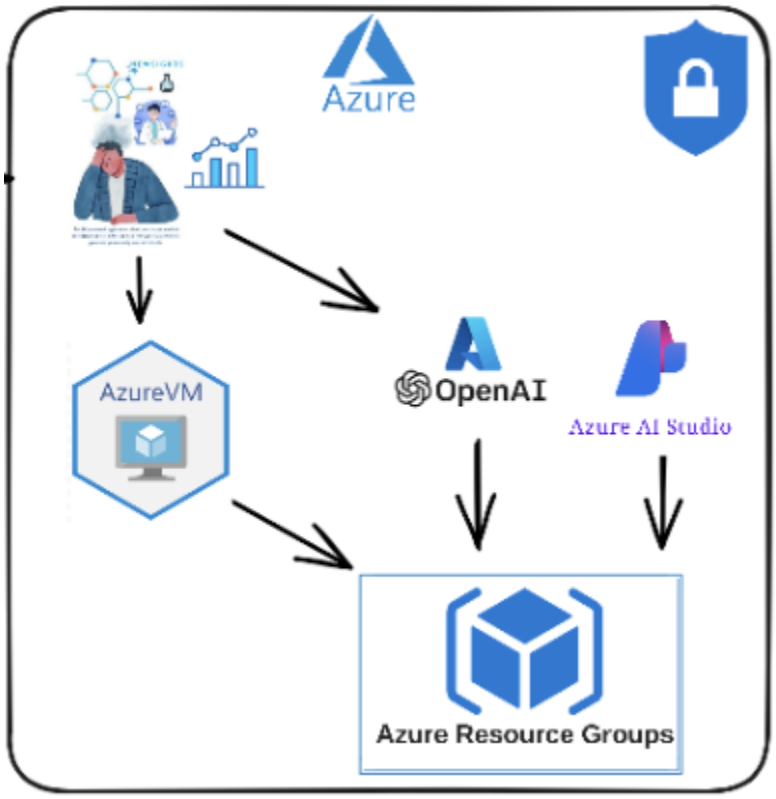
1. Neural Networks: These are computer systems inspired by the human brain. They process information in a way that's somewhat similar to how we do, helping the AI learn and create.
2. Training: The AI is trained on large datasets. If you want it to generate text, you feed it lots of books, articles, and other writings. It picks up on language patterns and styles from these examples.
3. Generation: Once trained, the AI can produce new content. For text, you might give it a starting sentence, and it will continue writing. For images, it might generate a completely new picture based on the styles it has learned.

Everyday Examples

- Text Generation: Tools like chatbots or writing assistants use generative AI to produce human-like text. They can help with drafting emails, writing articles, or even creating poetry.
- Image Creation: AI can create new artwork, design clothing, or even generate realistic photos of people who don't exist (like in deepfakes).
- Music Composition: Generative AI can compose original music, producing new songs that sound like they were created by a human musician.

Generative AI Application Development Environment

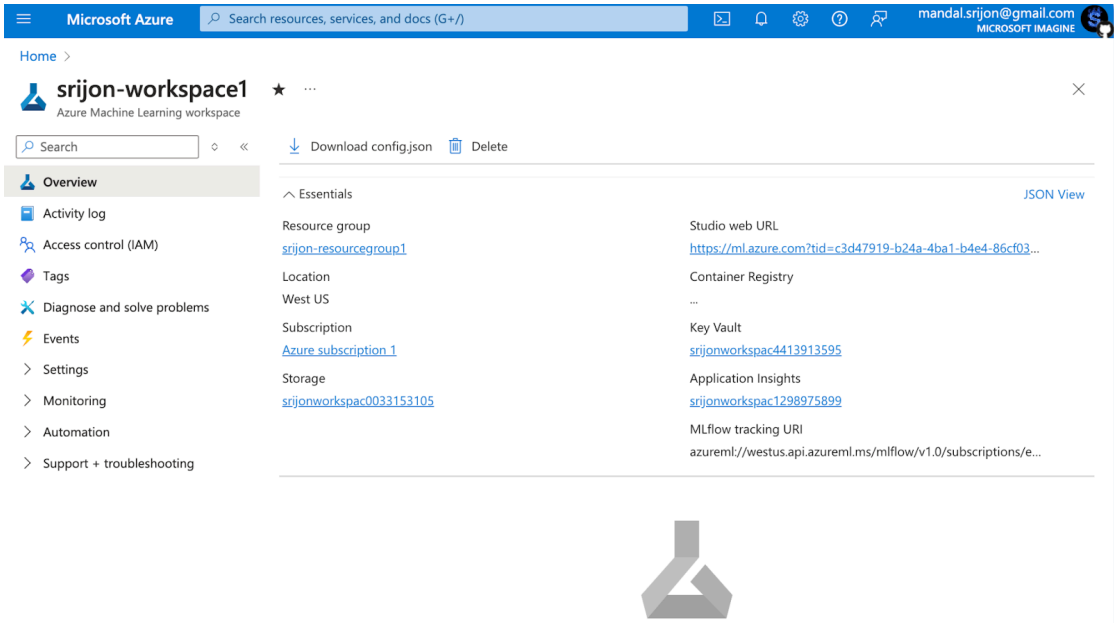
Azure Cloud was chosen as the Development Environment.



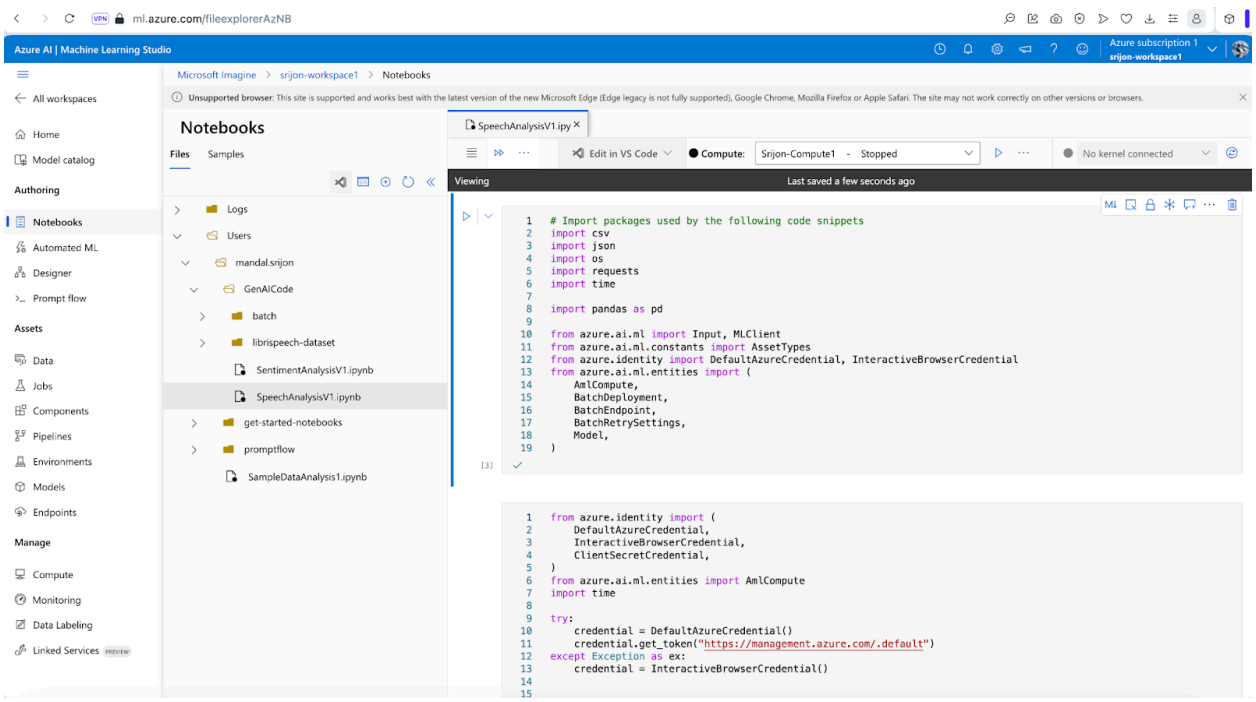
1. Azure Cognitive Services
    - Provides APIs for vision, speech, language, and decision-making capabilities.

    - Learn more
2. Azure Machine Learning
    - Platform for building, training, and deploying machine learning models.
    - Includes automated ML, designer interface, model management, and MLOps.
    - Learn more
3. Azure Bot Services
    - Tools for building, testing, and deploying conversational bots.
    - Integrates with multiple communication channels.
    - Learn more
4. Azure Cognitive Search
    - Cloud search service with AI capabilities for enriched indexing and semantic search.
    - Learn more
5. Azure OpenAI Service
    - Access to OpenAI's language models for natural language processing tasks.
    - Learn more
6. Azure Data and AI Integration
    - Seamless integration with Azure Synapse Analytics, Azure Data Factory, and Azure Databricks.
    - Learn more
7. AI Infrastructure
    - Specialized VMs, containers, AKS, and high-performance computing for AI workloads.
    - Learn more

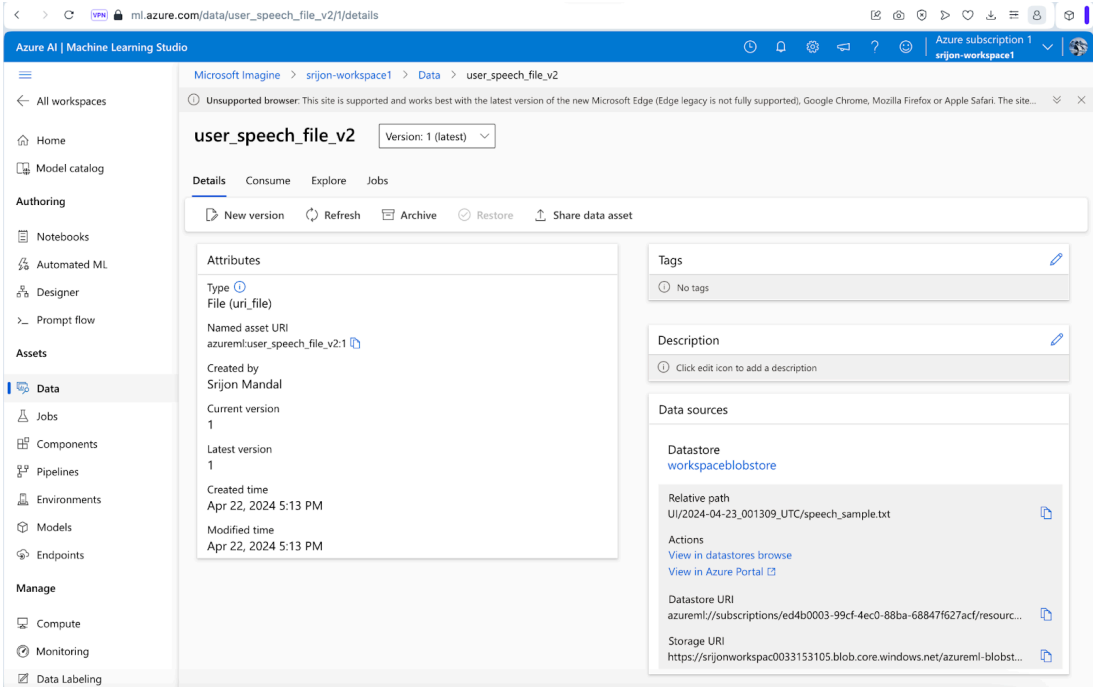
First a resource group and then a workspace need to be created inside Azure.



Next the Azure ML Instance should be created for developing Notebook.



Next the required Text, Audio and Video file data need to be stored in corresponding Datasets.



Generative AI R&D performed in Azure Environment

Generative AI Analysis Techniques

Advanced AI Technologies are used for Text Analysis

- OpenAI Prompt Engineering and Whisper: For processing and understanding natural language inputs and converting speech to text with high accuracy.
- Hugging Face Transformer and OpenAI for Emotion Detection: To interpret emotional states from text, providing insights into the patient's mental state.
- Emotion Detection from Video using YOLOv7 and PyTorch: To analyze facial expressions and body language from video data, offering another layer of emotional and symptomatic analysis.

Application of AI for Prescriptive Analytics:

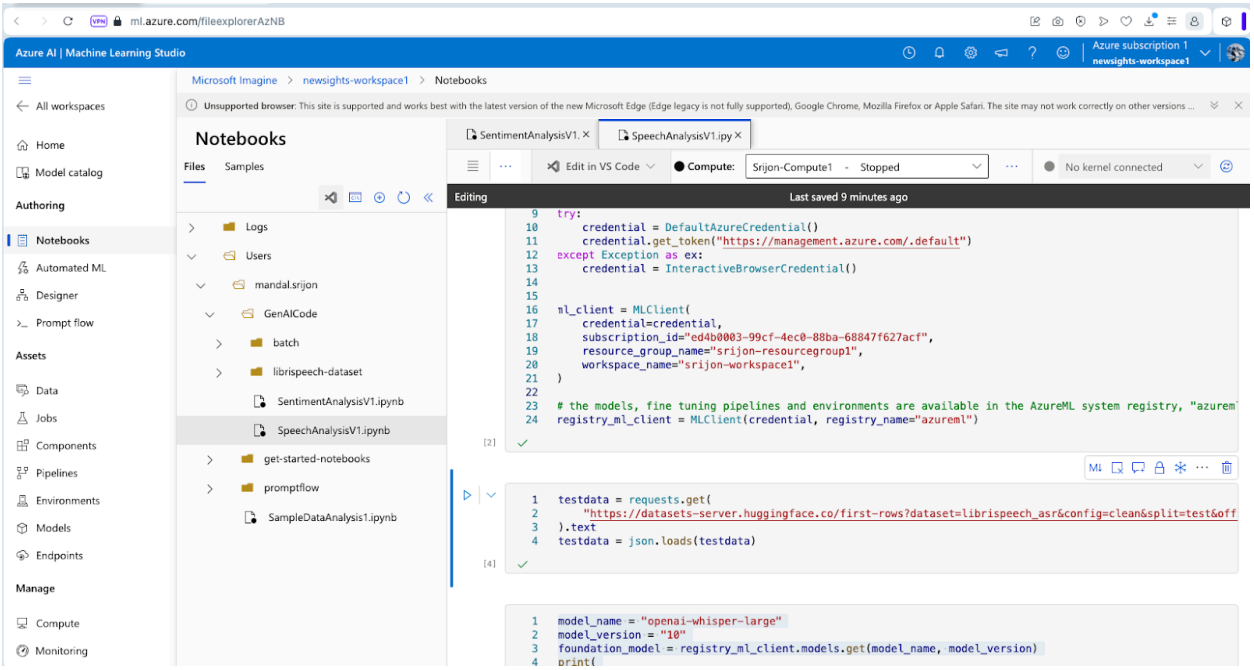
- ML Algorithms will be used to perform predictions and generate recommendations.

AI-driven Knowledge Management:

- Open-AI and RAG based Virtual Agent: It assists researchers and medical professionals by answering queries related to the patient data and broader schizophrenia research, facilitating an interactive, learning-enhanced environment.

Generative AI Notebook Setup

Once the Azure ML Studio instance is up and running in your workspace, its time to create the Notebooks.



Generative AI Programming Examples

Here goes the Example for analyzing sample Audio data in Azure Notebook

- In this example,
- we import required dependencies, create an instance of `MLClient,`
- `read the dataset into a dataframe,`
- `create a batch deployment endpoint`
- perform inference
- store the prediction result
- terminate the inference cluster

```
# Import packages used by the following code snippets
import csv
import json
import os
import requests
import time

import pandas as pd

from azure.ai.ml import Input, MLClient
from azure.ai.ml.constants import AssetTypes
from azure.identity import DefaultAzureCredential,
InteractiveBrowserCredential
from azure.ai.ml.entities import (
   AmlCompute,
   BatchDeployment,
   BatchEndpoint,
   BatchRetrySettings,
   Model,
)
from azure.identity import (
   DefaultAzureCredential,
   InteractiveBrowserCredential,
   ClientSecretCredential,
)
from azure.ai.ml.entities import AmlCompute
import time

try:
   credential = DefaultAzureCredential()
   credential.get_token("https://management.azure.com/.default")
except Exception as ex:
   credential = InteractiveBrowserCredential()


ml_client = MLClient(
   credential=credential,
   subscription_id="ed4b0003-99cf-4ec0-88ba-68847f627acf",
   resource_group_name="srijon-resourcegroup1",
   workspace_name="srijon-workspace1",
)
```

```
# the models, fine tuning pipelines and environments are available in the
AzureML system registry, "azureml"
registry_ml_client = MLClient(credential, registry_name="azureml")


model_name = "openai-whisper-large"
model_version = "10"
foundation_model = registry_ml_client.models.get(model_name,
model_version)
print(
   f"Using model name: {foundation_model.name}, version:
{foundation_model.version}, id: {foundation_model.id} for inferencing."
)

audio_urls_and_text = [
   (row["row"]["audio"][0]["src"], row["row"]["text"]) for row in
testdata["rows"]
]
test_df = pd.DataFrame(data=audio_urls_and_text, columns=["audio",
"text"])
# Define directories and filenames as variables
dataset_dir = "librispeech-dataset"
test_datafile = "test_100.csv"

batch_dir = "batch"
batch_inputs_dir = os.path.join(batch_dir, "inputs")
batch_input_file = "batch_input.csv"
os.makedirs(dataset_dir, exist_ok=True)
os.makedirs(batch_dir, exist_ok=True)
os.makedirs(batch_inputs_dir, exist_ok=True)
pd.set_option(
   "display.max_colwidth", 0
)  # Set the max column width to 0 to display the full text
test_df.head()

batch_df = test_df[["audio", "language"]]

# Divide this into files of 10 rows each
batch_size_per_predict = 10
for i in range(0, len(batch_df), batch_size_per_predict):
   j = i + batch_size_per_predict
```

```
    batch_df[i:j].to_csv(
        os.path.join(batch_inputs_dir, str(i) + batch_input_file),
quoting=csv.QUOTE_ALL
    )

# Check out the first and last file name created
input_files = os.listdir(batch_inputs_dir)
print(f"{input_files[0]} to {str(i)}{batch_input_file}.")

compute_name = "test-cpu-cluster"

compute_cluster = AmlCompute(
    name=compute_name,
    description="An AML compute cluster",
    size="Standard_E4as_v4",
    min_instances=1,
    max_instances=2,
    idle_time_before_scale_down=120,
)   # 120 seconds

ml_client.begin_create_or_update(compute_cluster)

# Endpoint names need to be unique in a region, hence using timestamp to
create unique endpoint name
timestamp = int(time.time())
endpoint_name = "speech-recognition-" + str(timestamp)

endpoint = BatchEndpoint(
    name=endpoint_name,
    description="Batch endpoint for "
    + foundation_model.name
    + ", for automatic-speech-recognition task",
)
ml_client.begin_create_or_update(endpoint).result()

deployment_name = "demo"

deployment = BatchDeployment(
    name=deployment_name,
    endpoint_name=endpoint_name,
    model=foundation_model.id,
    compute=compute_name,
```

```
    error_threshold=0,
    instance_count=1,
    logging_level="info",
    max_concurrency_per_instance=1,
    mini_batch_size=2,
    output_file_name="predictions.csv",
    retry_settings=BatchRetrySettings(max_retries=3, timeout=600),
)
ml_client.begin_create_or_update(deployment).result()

endpoint = ml_client.batch_endpoints.get(endpoint_name)
endpoint.defaults.deployment_name = deployment_name
ml_client.begin_create_or_update(endpoint).wait()

endpoint = ml_client.batch_endpoints.get(endpoint_name)
print(f"The default deployment is {endpoint.defaults.deployment_name}")
input = Input(path=batch_inputs_dir, type=AssetTypes.URI_FOLDER)

job = ml_client.batch_endpoints.invoke(
    endpoint_name=endpoint.name, input=input
)

ml_client.jobs.stream(job.name)

scoring_job = list(ml_client.jobs.list(parent_job_name=job.name))[0]

ml_client.jobs.download(
    name=scoring_job.name, download_path=batch_dir, output_name="score"
)

predictions_file = os.path.join(batch_dir, "named-outputs", "score",
"predictions.csv")

# Load the batch predictions file with no headers into a dataframe and set
your column names
score_df = pd.read_csv(
    predictions_file,
    header=None,
    names=["row_number_per_file", "prediction", "batch_input_file_name"],
)
score_df.head()
```

```
input_df = []
for file in input_files:
    input = pd.read_csv(os.path.join(batch_inputs_dir, file), index_col=0)
    input.reset_index(inplace=True)
    input["batch_input_file_name"] = file
    input.reset_index(names=["row_number_per_file"], inplace=True)
    input_df.append(input)
input_df = pd.concat(input_df)
input_df.set_index("index", inplace=True)
input_df = input_df.join(test_df.drop(columns=["audio", "language"]))

input_df.head()

df = pd.merge(
    input_df, score_df, how="inner", on=["row_number_per_file",
"batch_input_file_name"]
)

# Show the first few rows of the results
df.head(20)
ml_client.batch_endpoints.begin_delete(name=endpoint_name).result()
ml_client.compute.begin_delete(name=compute_name).result()
```

Link:
https://github.com/srijonmandal1/schizophrenia-patient-data-analysis/blob/main/speech-analysis/AzureSpeechAnalysisNotebookV1.ipynb

Sentiment Analysis, keyword Extraction and Summarization using OpenAI

Here goes a detailed example for performing all types of text analysis using OpenAI.

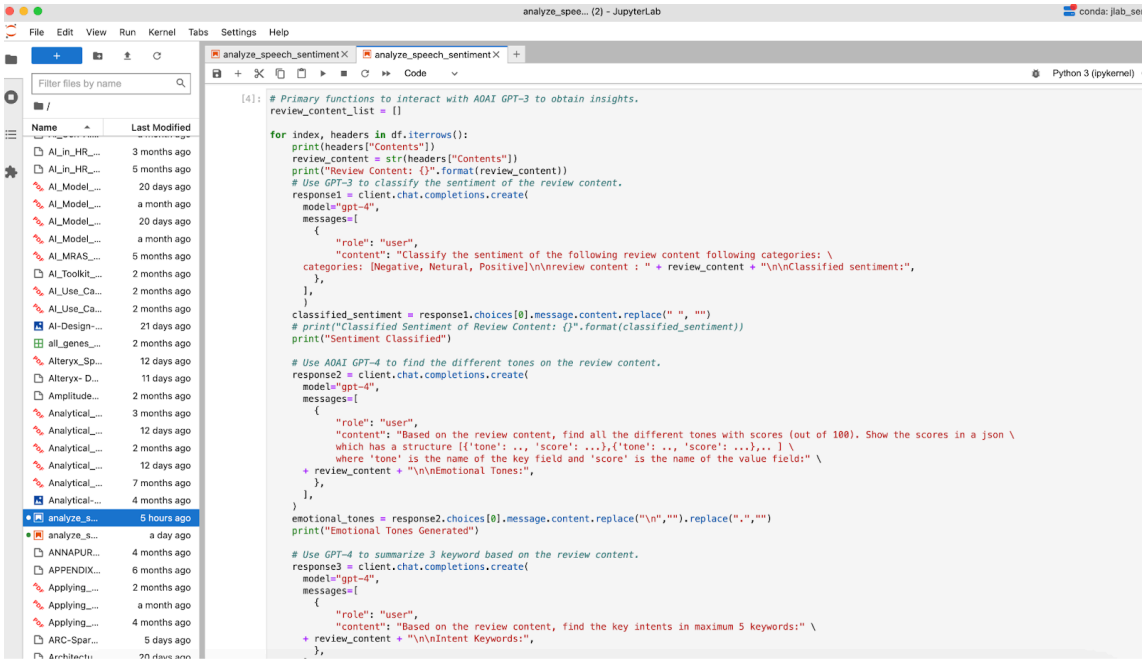
User needs to provide the sample text content.

==========

```
  print("Review Content: {}".format(review_content))
  # Use GPT-4 to classify the sentiment of the review content.
  response1 = client.chat.completions.create(
    model="gpt-4",
    messages=[
     {
        "role": "user",
```

```
            "content": "Classify the sentiment of the following review content following categories: \
        categories: [Negative, Netural, Positive]\n\nreview content : " + review_content +
"\n\nClassified sentiment:",
          },
        ],
        )
    classified_sentiment = response1.choices[0].message.content.replace(" ", "")
    # print("Classified Sentiment of Review Content: {}".format(classified_sentiment))
    print("Sentiment Classified")

    # Use GPT-4 to find the different tones on the review content.
    response2 = client.chat.completions.create(
      model="gpt-4",
      messages=[
        {
            "role": "user",
            "content": "Based on the review content, find all the different tones with scores (out of
100). Show the scores in a json \
            which has a structure [{'tone': .., 'score': ...},{'tone': .., 'score': ...},.. ] \
            where 'tone' is the name of the key field and 'score' is the name of the value field:" \
        + review_content + "\n\nEmotional Tones:",
        },
      ],
    )
    emotional_tones = response2.choices[0].message.content.replace("\n","").replace(".","")
    print("Emotional Tones Generated")

    # Use GPT-4 to summarize the given content.
    response3 = client.chat.completions.create(
      model="gpt-4",
      messages=[
        {
            "role": "user",
            "content": "Based on the review content, find the key intents in maximum 5 keywords:" \
        + review_content + "\n\nIntent Keywords:",
        },
      ],
    )
    summarized_keywords =
response3.choices[0].message.content.replace("\n","").replace(".","")

    # print("Summarize 3 Keywords from the Review Content: {}".format(summarized_keywords))
    print("Intent Keywords Generated")
```

```
    # Append the insights result into a list.
    review_content_list.append([review_content, classified_sentiment, emotional_tones,
summarized_keywords])

# Convert the list of insights into a Pandas dataframe.
review_content_df = pd.DataFrame(review_content_list, columns=['review_content',
'classified_sentiment', 'emotional_tones','intent_keywords'])
=======================
```



Audio Analysis using OpenAI Whisper Model

Following code shows how to transcribe the Audio data

```
==============
from openai import OpenAI

client = OpenAI(api_key=api_key_string)
file_path = "/Users/kaniska/Downloads/test.wav"
audio_file= open(file_path, "rb")
transcription = client.audio.transcriptions.create(
  model="whisper-1",
  file=audio_file
)
print(transcription.text)

import librosa
```

```
waveform, sample_rate = librosa.load(file_path, sr=None)

import matplotlib.pyplot as plt

time_axis = librosa.times_like(waveform, sr=sample_rate)

plt.figure(figsize=(10, 4))
plt.plot(time_axis, waveform)
plt.title('Waveform of Audio')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.show()

content_prompt = "Extract keywords from this text:\n\n"+str(transcription.text)

response = client.chat.completions.create(
  model="gpt-3.5-turbo-0125",
  response_format={"type":"json_object"},
  messages=[
    {"role": "system", "content": "You are a helpful assistant designed to output JSON."},
    {"role": "user", "content": content_prompt}
  ]
)

print(response.choices[0].message.content)

content_prompt = f"Please analyze the sentiment of the following text:{transcription.text}"

response = client.chat.completions.create(
  model="gpt-4",
  messages=[
    {
      "role": "system",
      "content": "You will be provided with a text, and your task is to classify its sentiment as
positive, neutral, or negative."
    },
    {
      "role": "user",
      "content": content_prompt
    }
  ],
  temperature=0.7,
  max_tokens=64,
  top_p=1
```

```
)

#sentiment = response.choices[0].content.strip().lower()

print(response)
==============
```

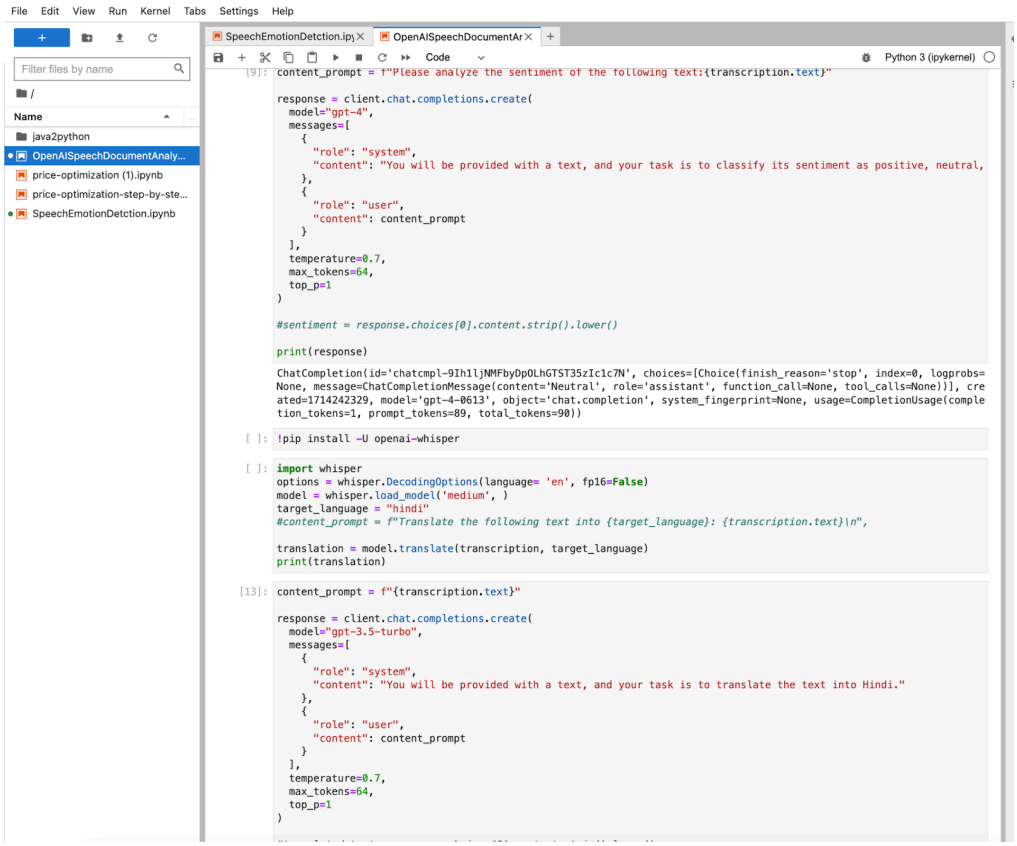
Translation using OpenAI Model

Following code shows how to use prompt and openai to analyze the transcription data generated from the above code

```
==============


import whisper
options = whisper.DecodingOptions(language= 'en', fp16=False)
model = whisper.load_model('medium', )
target_language = "hindi"
#content_prompt = f"Translate the following text into {target_language}: {transcription.text}\n",

translation = model.translate(transcription, target_language)
print(translation)


content_prompt = f"{transcription.text}"

response = client.chat.completions.create(
  model="gpt-3.5-turbo",
  messages=[
    {
      "role": "system",
      "content": "You will be provided with a text, and your task is to translate the text into Hindi."
    },
    {
      "role": "user",
      "content": content_prompt
    }
  ],
  temperature=0.7,
  max_tokens=64,
  top_p=1
)
```

#translated_text = response.choices[0].content.strip().lower()

print(response)



==========

Audio Waveform Analysis using HuggingFace Model

Following code shows how to use Huggingface library and model for Audio data analysis

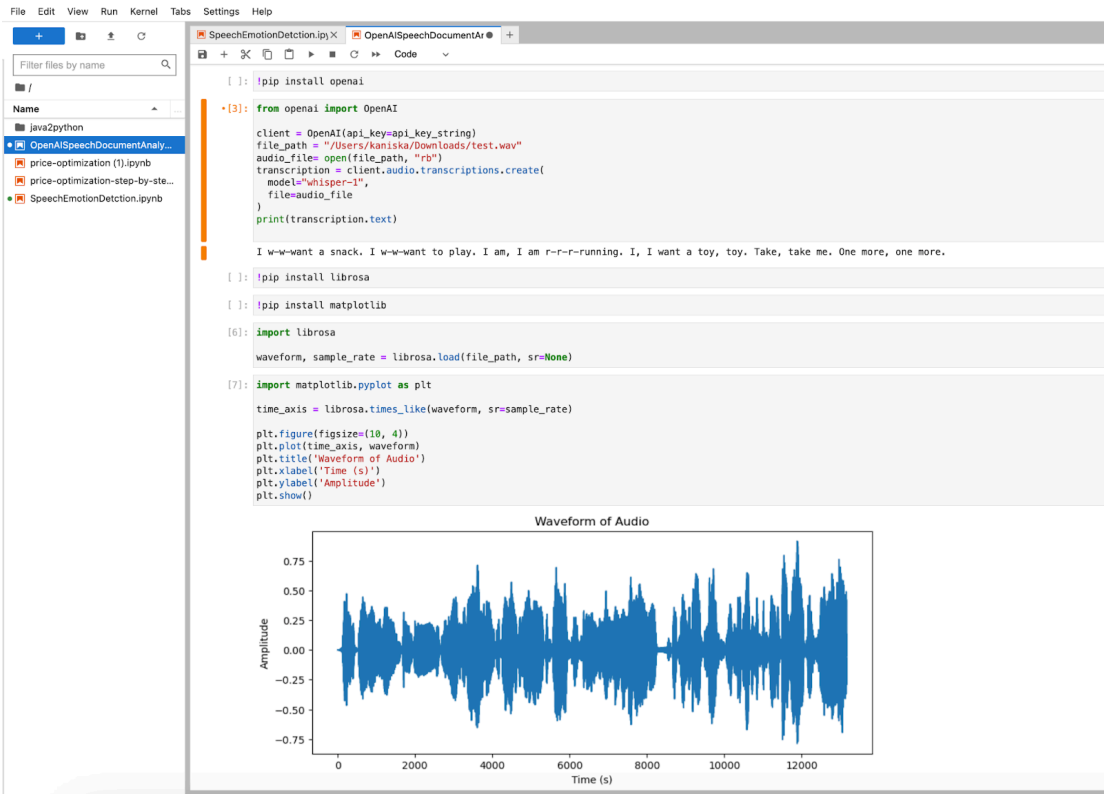
==========

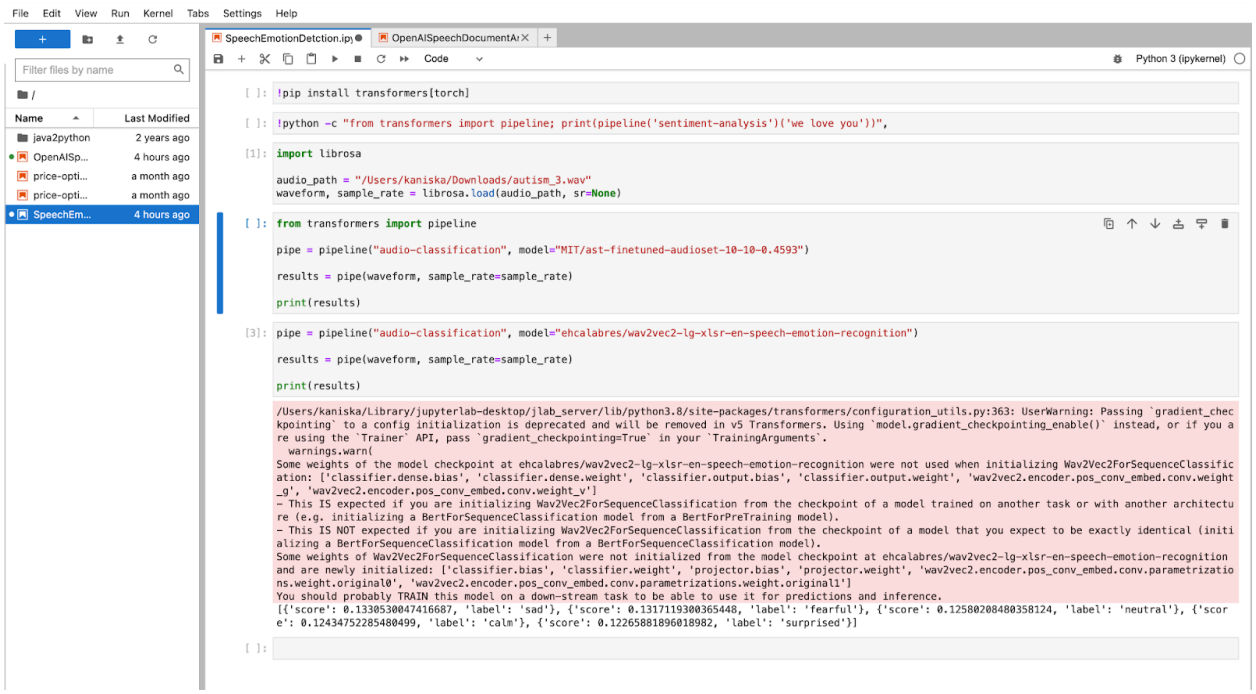
```
from transformers import pipeline

pipe = pipeline("audio-classification", model="MIT/ast-finetuned-audioset-10-10-0.4593")

results = pipe(waveform, sample_rate=sample_rate)

print(results)
```

pipe = pipeline("audio-classification",
model="ehcalabres/wav2vec2-lg-xlsr-en-speech-emotion-recognition")

results = pipe(waveform, sample_rate=sample_rate)

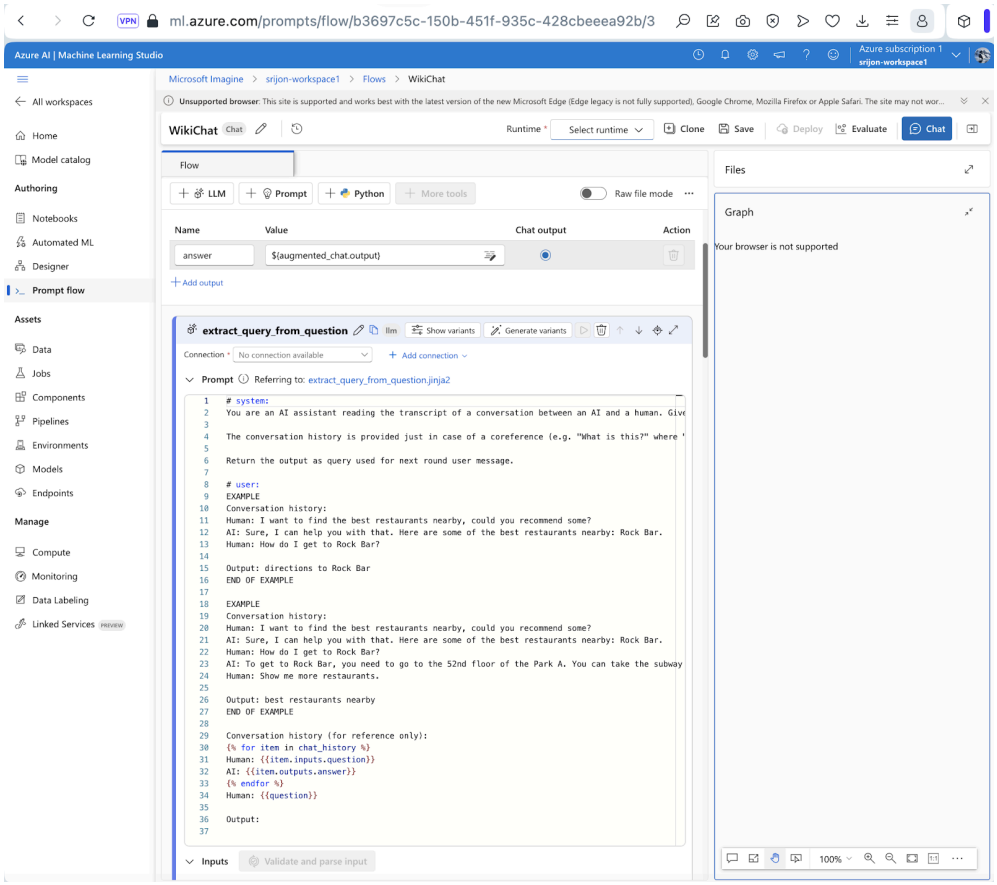
print(results)





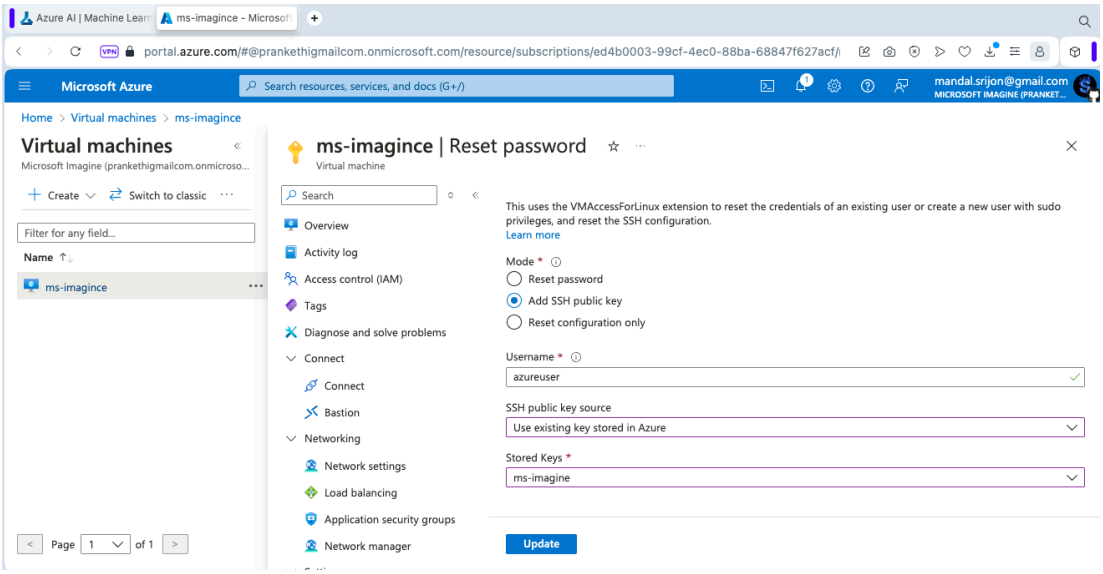
==========

Create a Chat Agent using Intelligent Prompts in Azure

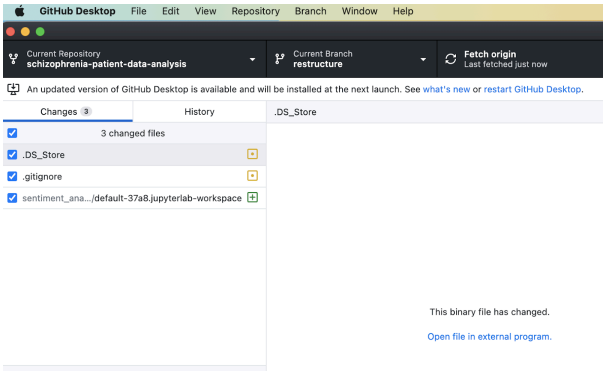


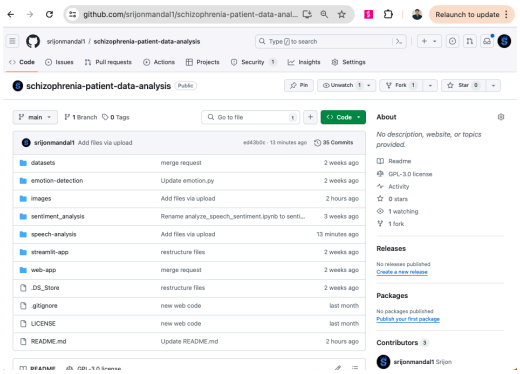
Setup a Develop Environment in local Machine and Push Code to Azure

Provision a Virtual Machine and import Code from Github



- one can download the .pem file and login to the remote VM using SSH
- one can download the code from github into local machine and SCP to the remote VM





Run Application

Run App

- Go to the app folder
- cd streamlit-app

- Set OPENAI API Key
  Export OPENAI_API_KEY in the VM
  Set api_key_string in Prototype.py

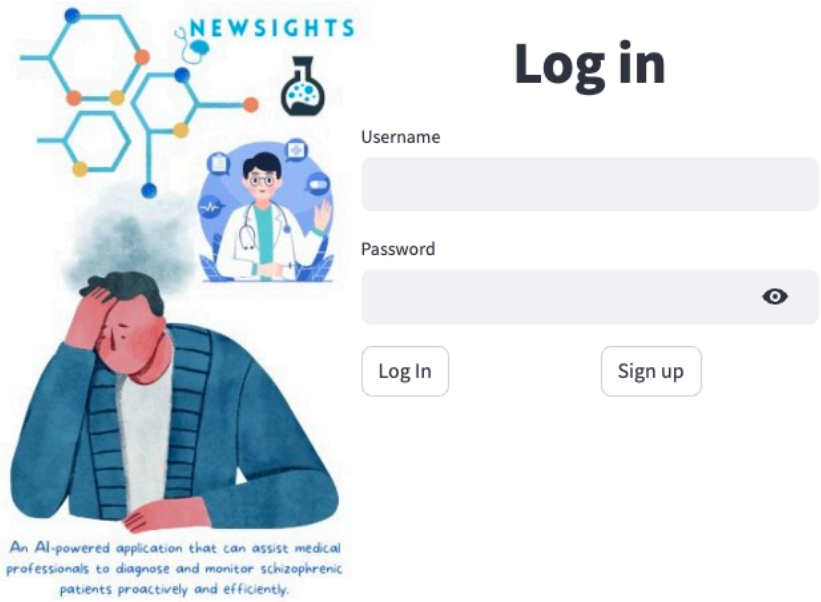
- Run the Prototype

  nohup python3 Prototype.py &
  tail -f nohup.out

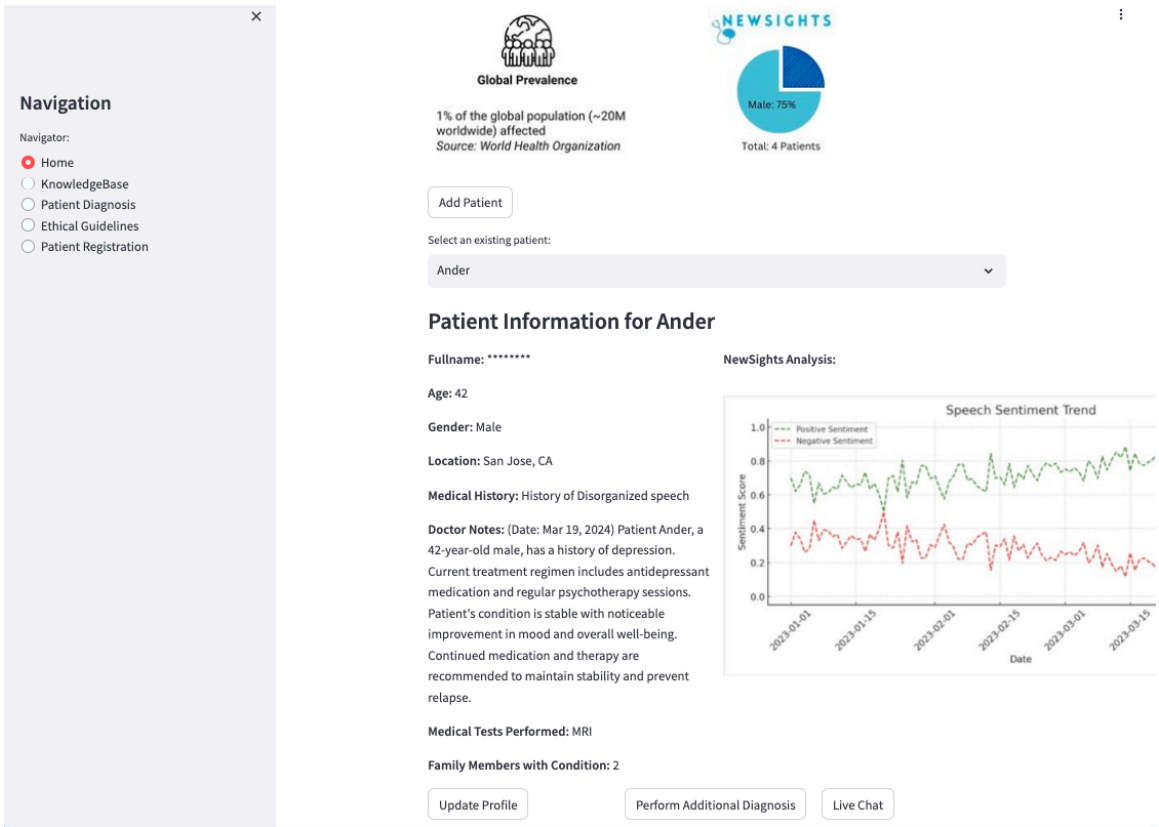
- View the App Use the localhost or IP Address assigned to the VM. For example, the below URL can be accessed by anyone. This Demo VM is running in Azure.

User Interaction

This is a prototype where a Lab User can login using admin credential.



The Lab User will be able to look up the overall patient status and insights in the Homepage. He will also be able to chat with a virtual agent which can search all the patient details and knowledgebase. This page also provides the overall patient stats.

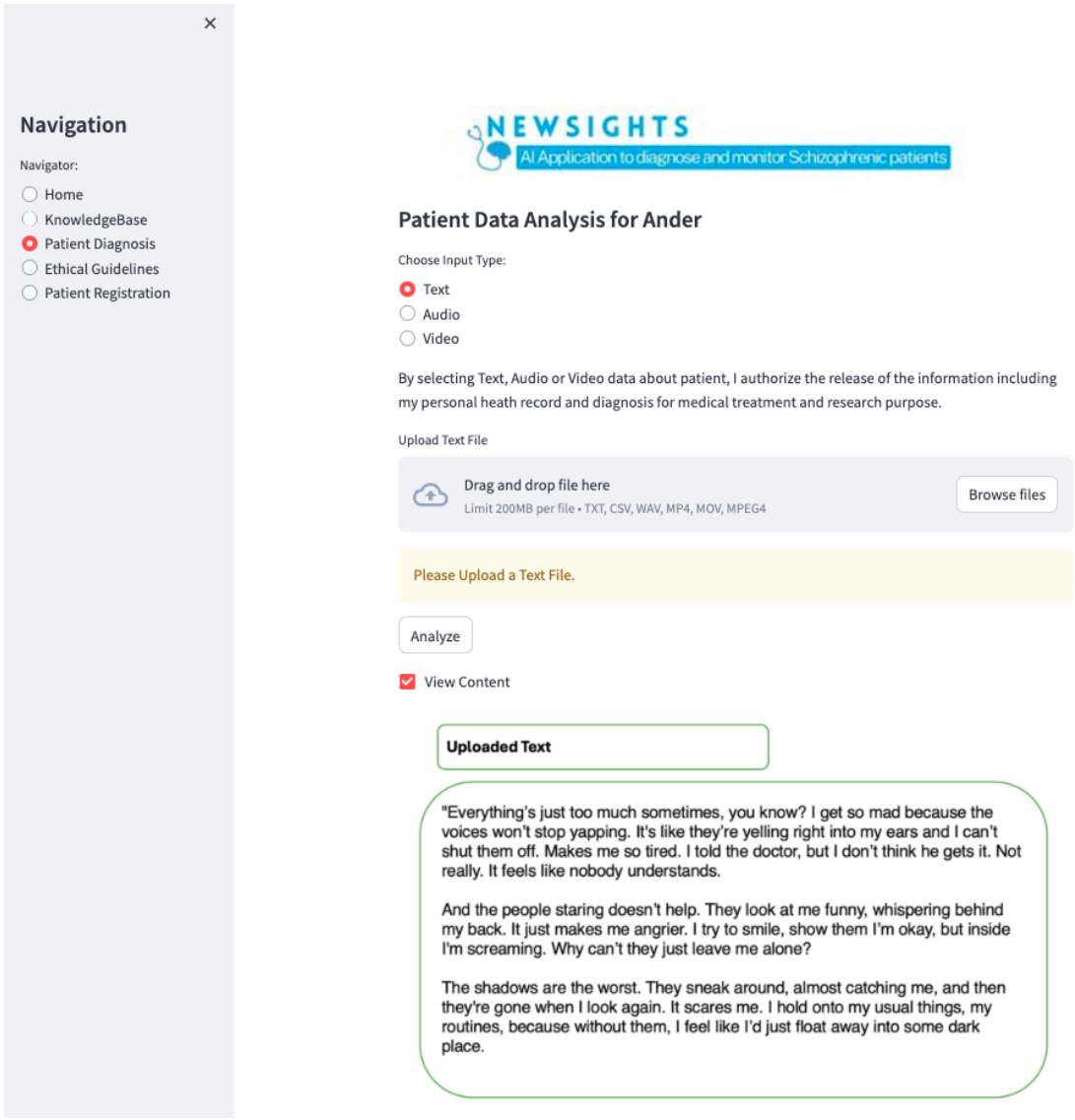


The Lab User will be able to ask Questions.

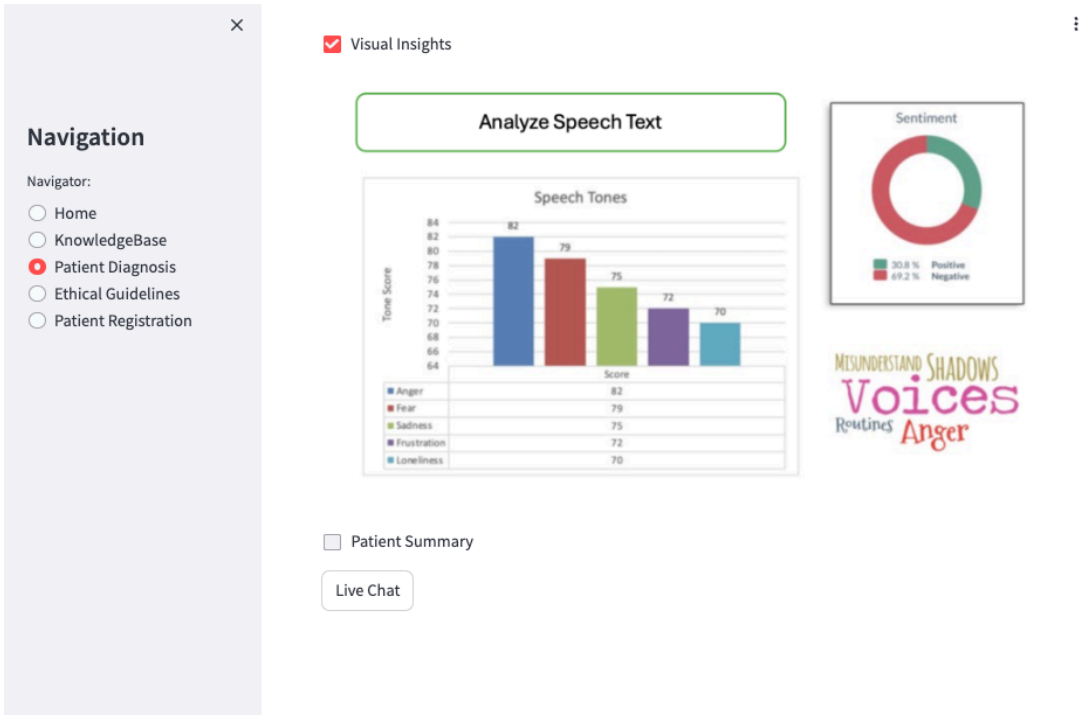
The Lab user can select a patient and analyze specific Texts, Video and Speech Transcription data as shown below.

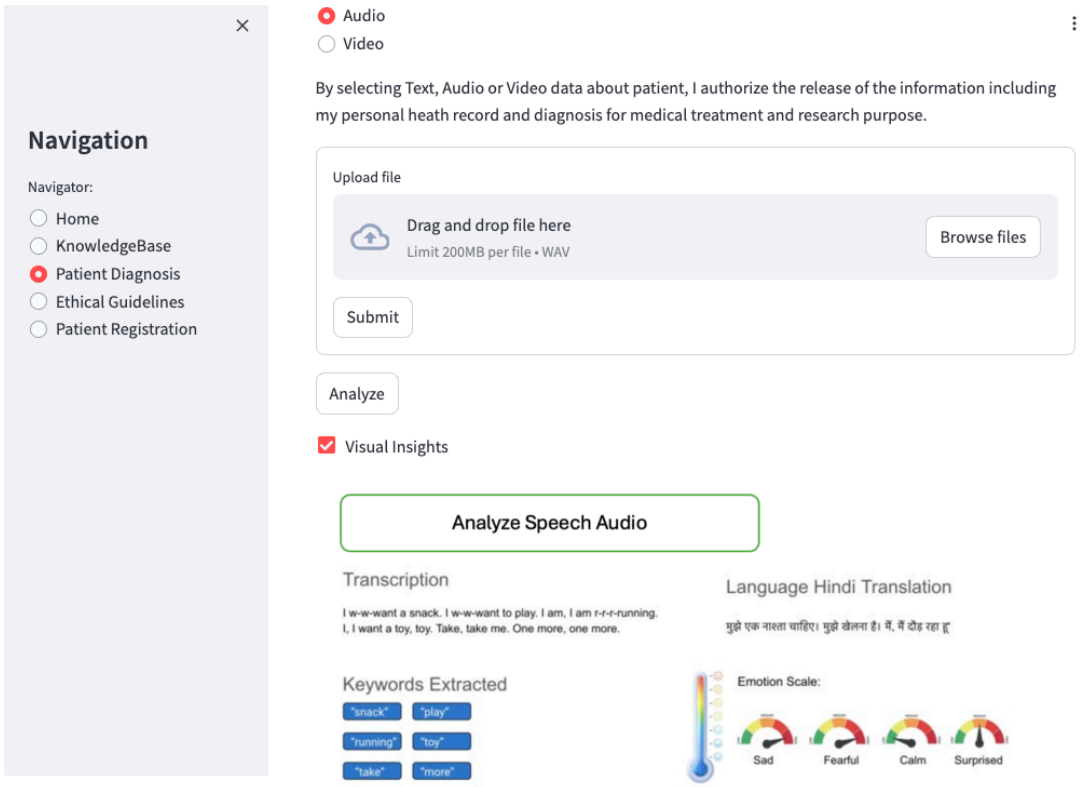
The Lab can perform extensive analysis on patient's data by leveraging the Generative AI functionalities explained in the previous sections. All the insights will be saved into a RAG database for Q&A and will also be saved in a Datastore for historical analysis and predictive analytics.

Analyze Textual Data

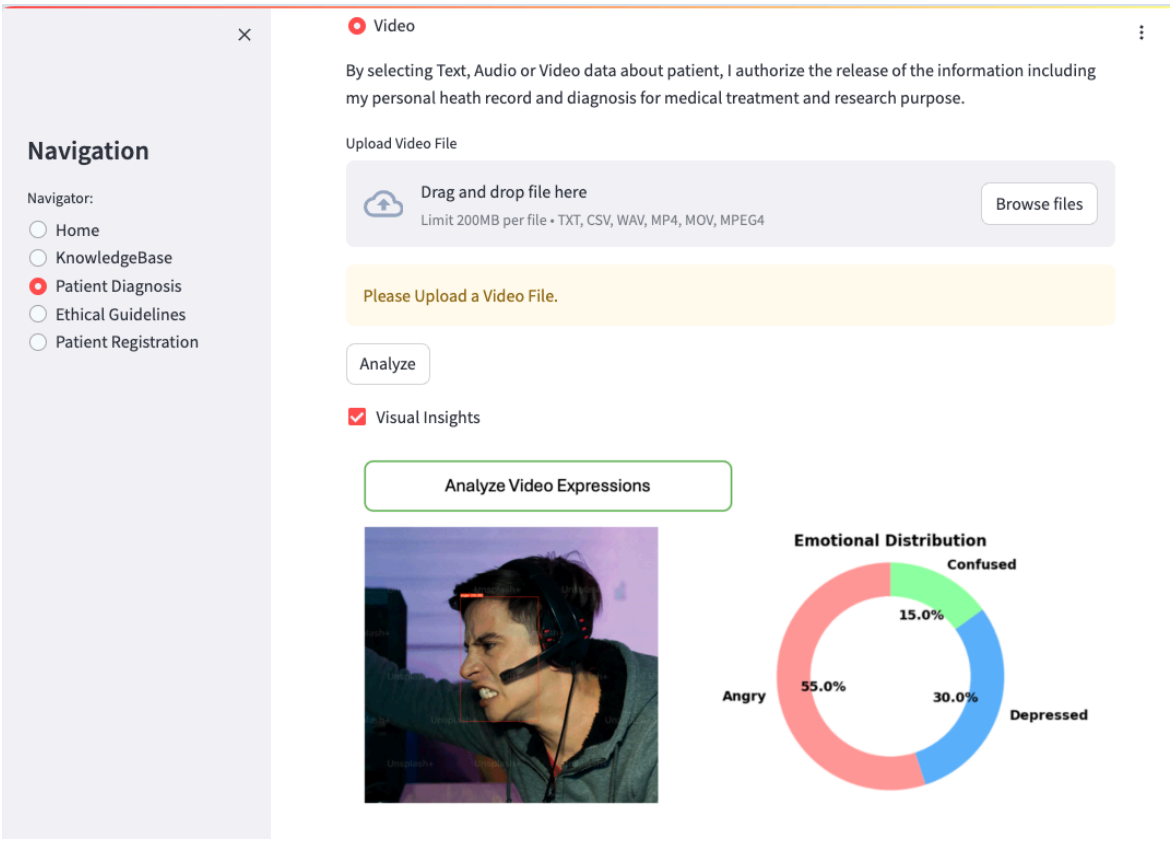


Analyze Audio Data





Analyze Video Data



Reference Codebase: https://github.com/srijonmandal1/schizophrenia-patient-data-analysis

Learn More

- Text Classification using Azure AI
    - https://github.com/Azure/azureml-examples/blob/main/sdk/python/foundation-models/system/inference/text-classification/entailment-contradiction-batch.ipynb
- Using Huggingface Model
    - https://github.com/balakreshnan/Samples2021/blob/main/AzureML/hugginface1.md