

# MIMIC LLM

SUJAY GOPINATHAN (SG59258)

# User Story

- ▶ Study how to use ChatGPT for reasoning and inferencing of medical data.
- ▶ COVID-19 synthetic data used (<https://mitre.box.com/shared/static/9iglv8kbs1pfi7z8phjl9sbpjk08spze.zip>)
- ▶ Github: <https://github.com/sujaycloud/aih>

# DataSet

- ▶ Following COVID-19 tables are being used
  - ▶ patients.csv.gz
  - ▶ observations.csv.gz
  - ▶ conditions.csv.gz
  - ▶ procedures.csv.gz

```
patients = pd.read_csv('./10k_synthea_covid19_csv/patients.csv')
conditions = pd.read_csv('./10k_synthea_covid19_csv/conditions.csv')
observations = pd.read_csv('./10k_synthea_covid19_csv/observations.csv')
procedures = pd.read_csv('./10k_synthea_covid19_csv/procedures.csv')
patients.info()
```

✓ 1.4s

# Pre-processing

- ▶ Find only COVID-19 cases (840539006)
- ▶ Merge with the observation table and filter for O2 saturation (2708-6), respiratory rate (9279-1), Ferritin (2276-4) and pivot on the observation column.
- ▶ Get the ground truth label (Ventilator code: 26763009) from the procedures table

```
# Find patients that needed ventilation from procedures
ventilation_procedures = procedures[procedures['CODE'].isin([26763009])]
# Add a column to identify patients needing ventilation
ventilation_procedures = ventilation_procedures[['PATIENT']]
ventilation_procedures['VENTILATOR'] = True
ventilation_procedures.head()
```

```
# filter for specific observation codes (O2 saturation), respiratory rate, ferritin
merged_data = merged_data[merged_data['CODE_y'].isin(['2708-6', '9279-1', '2276-4'])]
merged_data = merged_data.rename(columns={'CODE_y': 'OBSERVATION_CODE'})
merged_data['PATIENT'].nunique()
# for each patient, get the max value of each observation
merged_data = merged_data.groupby(['PATIENT', 'OBSERVATION_CODE']).agg({'VALUE': 'max'}).reset_index()
merged_data = merged_data.merge(patients_subset, left_on='PATIENT', right_on='Id')
merged_data = merged_data.drop(columns=['Id'])
merged_data.info()
merged_data.head()
# Pivot the data to have observations as columns
pivot_data = merged_data.pivot(index='PATIENT', columns='OBSERVATION_CODE', values='VALUE')
pivot_data.head()
```

```
# Find number of unique patients with COVID-19
covid_patients = conditions[conditions['CODE'] == 840539006]
• Click to add a breakpoint covid_patients[['PATIENT', 'CODE']]
covid_patients = covid_patients.merge(patients_subset, left_on='PATIENT', right_on='Id')
covid_patients.head()
```

✓ 0.0s

Python

# Pre-processed dataset

- ▶ Input features:
  - ▶ O2 saturation
  - ▶ Respiratory Rate (RR)
  - ▶ Ferritin
- ▶ Output feature
  - ▶ Ventilator

|    | PATIENT                              | O2 Saturation | RR   | Ferritin | VENTILATOR |
|----|--------------------------------------|---------------|------|----------|------------|
| 11 | 0100f99a-1b5d-4a5b-a73f-559a920412e5 | 88.8          | 39.5 | 982.2    | True       |
| 12 | 0100f99a-1b5d-4a5b-a73f-559a920412e5 | 88.8          | 39.5 | 982.2    | True       |
| 13 | 0100f99a-1b5d-4a5b-a73f-559a920412e5 | 88.8          | 39.5 | 982.2    | True       |
| 14 | 0100f99a-1b5d-4a5b-a73f-559a920412e5 | 88.8          | 39.5 | 982.2    | True       |
| 15 | 0100f99a-1b5d-4a5b-a73f-559a920412e5 | 88.8          | 39.5 | 982.2    | True       |

# Open-API

- ▶ Zero shot request/response
- ▶ Chat based approach
- ▶ Notice how the response changed from a Yes to a No

```
response = openai.ChatCompletion.create(  
    model="gpt-4",  
    messages=[  
        {"role": "system", "content": "You are an expert on Covid diagnosis."},  
        {"role": "user", "content": "Decide in a single word if the patient needs ventilation based on the"}  
    ]  
)  
  
print(response.choices[0].message['content'])
```

✓ 0.4s Python

Yes

```
response = openai.ChatCompletion.create(  
    # use cheaper model for testing  
    model = "gpt-4",  
  
    messages=[  
        {"role": "user", "content": "Decide in a single word if the patient needs ventilation based on the"}  
    ]  
)  
  
print(response.choices[0].message['content'])
```

✓ 1.0s Python

No

# Open-API (Reasoning)

- Chat conversation with reasoning

```
messages = [
    {"role": "system", "content": "You are an expert on Covid diagnosis."},
    {"role": "user", "content": "Decide in a single word if the patient needs ventilation based on the fo"},
    {"role": "assistant", "content": response.choices[0].message['content']},
    {"role": "user", "content": "Can you provide details why you made that decision?"},
]

response = openai.ChatCompletion.create(
    model="gpt-4",
    messages=messages
)

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

✓ 6.6s

Python

While an oxygen saturation level of 95% falls within the normal range (94–100%), the patient's elevated respi

# Open-API (Custom dataset)

- ▶ Create a dataset class
- ▶ Use ChatGPT to get the response from the dataset

```
from torch.utils.data import Dataset

class VentilatorDataset(Dataset):
    def __init__(self, df):
        self.df = df

    def __len__(self):
        return len(self.df)

    def __getitem__(self, index):
        column_names = [
            ["O2 Saturation", "The first observation is oxygen saturation at "],
            ("RR", "The second observation is respiratory rate at "),
            ("Ferritin", "The third observation is ferritin level at ")
        ]

        x_strs = [f"{col_desc}{self.df.iloc[index][col]}" for col, col_desc in column_names]
        x_str = ''.join(x_strs)
        x_str = x_str.replace('\n', ' ')
        x_str = 'Decide in a single word if the patient needs ventilation: True or False '+x_str

        return x_str
```

✓ 0.0s

Python

```
test_ds = VentilatorDataset(df.iloc[test_index])
```

✓ 0.0s

Python

Use ChatGPT to get the response to the prompts from the dataset

```
from tqdm import tqdm
import time

results = []
for prompt in tqdm(test_ds):
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[
            {"role": "user", "content": prompt},
        ]
    )
    results.append(response.choices[0].message['content'])
    time.sleep(3)
```

✓ 41m 2.4s

Python

100% | 722/722 [41:02<00:00, 3.41s/it]



# Open-API (Custom dataset)

- ▶ Accuracy
  - ▶ AUROC = 49%
  - ▶ AUPRC = 55%

```
from sklearn.metrics import roc_auc_score, average_precision_score
results = [1 if r.strip().lower() == 'true' else 0 for r in results]
test_labels = df.iloc[test_index]['VENTILATOR'].tolist()
auroc = roc_auc_score(test_labels, results)
auprc = average_precision_score(test_labels, results)
print("AUROC:", auroc)
print("AUPRC:", auprc)
```

✓ 0.0s

Python

AUROC: 0.49155275148966  
AUPRC: 0.5568181502874134

# Open-API (embeddings)

- Generate embeddings after training the custom dataset

```
def generate_embeddings(texts, model="text-embedding-ada-002"):
    embeddings = []
    for text in tqdm(texts):
        text = text.replace("\n", " ")
        response = openai.Embedding.create(input = [text], model=model)['data']
        embeddings.append(response[0]['embedding'])
    return np.array(embeddings)
```

✓ 0.0s Python

Generate + Code + Markdown

• Get the embeddings of the training dataset Add Code Cell (%Enter)

```
train_ds = VentilatorDataset(df.iloc[train_index])
embeddings = generate_embeddings(train_ds)
```

✓ 13m 51.1s Python

100%|██████████| 2884/2884 [13:50<00:00, 3.47it/s]

```
np.shape(embeddings)
```

✓ 0.0s Python

(2884, 1536)

# Logistic Regression

- ▶ Use logistic regression with embeddings from ChatGPT
- ▶ Accuracy improvements
  - ▶ AUROC: 92%
  - ▶ AUOPRC: 93%

```
test_embeddings = generate_embeddings(test_ds)
test_labels = list(df.iloc[test_index]['VENTILATOR'])

test_pred = model.predict_proba(test_embeddings)[:,-1]
auroc = roc_auc_score(test_labels, test_pred)
auprc = average_precision_score(test_labels, test_pred)
print('\nAUROC:', auroc, '\nAUPRC', auprc)
```

✓ 3m 18.5s

Python

100%|██████████| 722/722 [03:18<00:00, 3.64it/s]

AUROC: 0.9218911866651088

AUPRC 0.9341415761558569

# Conclusion

- ▶ Using proper prompts with ChatGPT is critical
- ▶ Simple logistic regression with embeddings from ChatGPT improves prediction accuracy of the custom test dataset from ~50% to ~93%