PHASE-2 DATA EXPLORATION AND SOLUTION PLANNING

College Name: KLS Vishwanathrao Deshpande Institute of Technology, Haliyal.

Group Members:

- **Name:** PAVANI T. KHATMAL

  CAN ID Number: CAN_33907639

  Work: Data Exploration (Data Exploration Steps: Text length analysis to Feature Relationships)

  Solution Architecture (NLP and AI model development)
- **Name:** TAHIRA H. SHAIKH

  CAN ID Number: CAN_33907375
- **Name:** SHIFA L. C

  CAN ID Number: CAN_33986589
- **Name:** AISHWARYA J MIRASHI

  CAN ID Number: CAN_33991203

Data Exploration:

Objective:

Understand the structure, quality, and patterns in the dataset to identify challenges and opportunities for model development.

The dataset we used in this project is the “health prescription data.csv” file and the file contains 744 rows and 7 columns**.**

Dataset Overview:

1. Columns and Their Types:

- **SUBJECT_ID:** Unique identifier for patients (integer).
- **ROW_ID:** Row-level unique identifier (integer).
- **HADM_ID:** Hospital admission ID (integer).

- **CATEGORY:** The category of the medical record (e.g., "Discharge summary").
- **ADMISSION_TYPE:** Type of admission (e.g., "EMERGENCY," "ELECTIVE").
- **DIAGNOSIS:** Summary of diagnosis for the record (text).
- **TEXT:** Detailed medical notes (text).

2. Non-Null Counts:
   All columns have complete data (no missing values).

3. Memory Usage:
   Approximately 40.8 KB, making it computationally manageable for processing and analysis.

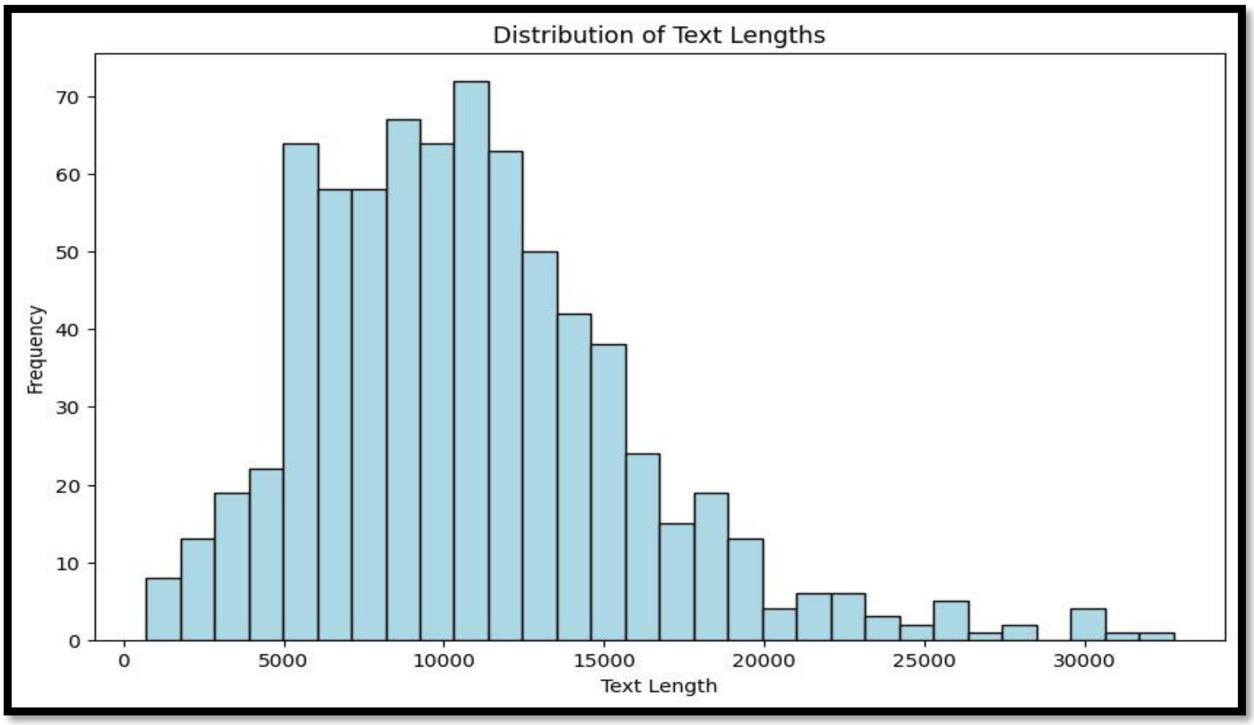
DATA EXPLORATION STEPS:

1. Basic Data Statistics:
   Understanding numerical distributions and unique values in categorical data.
   Code:
   ```
   # Summary statistics for numeric columns
   data.describe()
   #Unique values in categorical columns
   for col in [‘CATEGORY’, ‘ADMISSION_TYPE’]:
   print(f”{col} unique values:\n”, data[col].value_counts())
   ```

2. Missing Data Analysis:
   Visualizing and conforming missing data patterns.
   Code:
   ```
   import seaborn as sns
   import matplotlib.pyplot as plt
   # Visualizing missing data
   plt.figure(figsize=(10,6))
   sns.heatmap(data.isnull(), cbar=False, cmap=’viridis’)
   plt.title(“Missing Data Heatmap”)
   plt.show()
   ```

3. Text Length Analysis:
   Exploring the TEXT and DIAGNOSIS fields for content length.
   Code:

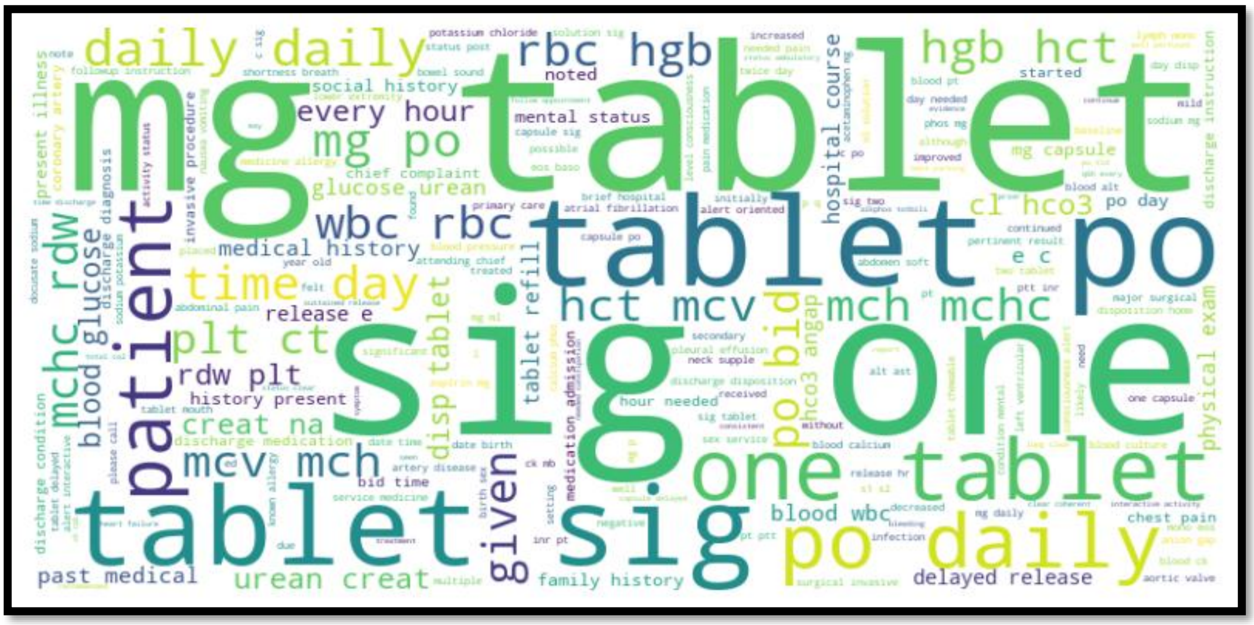
```
# Add columns for text length
data['text_length'] = data['TEXT'].apply(len)
data['diagnosis_length'] = data['DIAGNOSIS'].apply(len)
# Summary statistics for text lengths
print(data[['text_length', 'diagnosis_length']].describe())

# Histogram of text lengths
plt.figure(figsize=(10, 6))
plt.hist(data['text_length'], bins=30, color='lightblue', edgecolor='black')
plt.title("Distribution of Text Lengths")
plt.xlabel("Text Length")
plt.ylabel("Frequency")
plt.show()
```



4. <u>Word Frequency in TEXT and DIAGNOSIS:</u>
Understanding frequently occurring terms.
<u>Code:</u>

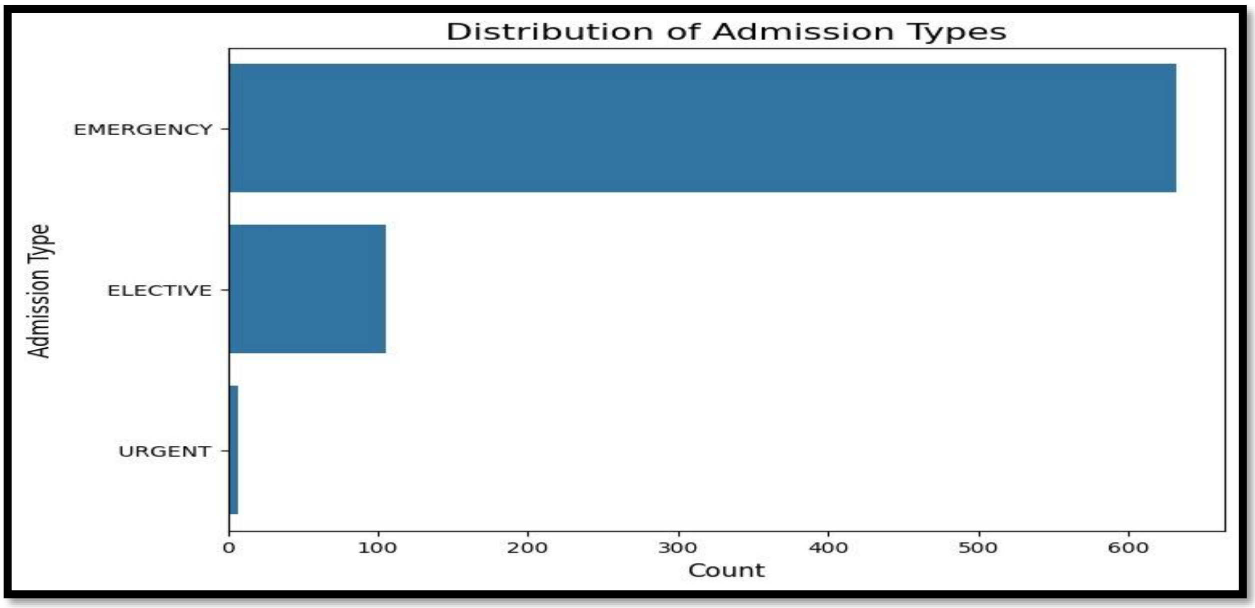
```
from collections import Counter
from wordcloud import WordCloud

# Combine all diagnosis and text for analysis
all_text = ' '.join(data['TEXT'])
```

```
all_diagnosis = ' '.join(data['DIAGNOSIS'])

# Word frequency for `DIAGNOSIS`
diagnosis_counter = Counter(all_diagnosis.split())
print("Most common diagnosis terms:",
diagnosis_counter.most_common(10))

# Word cloud for `TEXT`
wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(all_text)
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title("Word Cloud of Medical Notes")
plt.show()
```



5. <u>Feature Relationships:</u>
   Analysing potential relationships between admission types and diagnosis patterns.
   <u>Code:</u>
   ```
   # Diagnosis count per admission type
   admission_diagnosis=data.groupby('ADMISSION_TYPE')['DIAGNOSIS']
   .count()
   ```

```
# Bar plot for admission types
admission_diagnosis.plot(kind='bar', figsize=(8, 6), color='coral',
edgecolor='black')
plt.title("Diagnosis Count by Admission Type")
plt.xlabel("Admission Type")
plt.ylabel("Diagnosis Count")
plt.show()
```



Insights from Data Exploration:

Text Length Analysis

1. Summary Statistics:
    - Average text length in the TEXT column: 10,805 characters.
    - Average text length in the DIAGNOSIS column: 21 characters.
    - Maximum text length: 32,759 characters.

   Observations:
    - The histogram shows a wide distribution of text lengths, with most medical notes clustered between 7,000 and 15,000 characters.
    - Outliers may exist with very short or very long notes.

2. Diagnosis Count by Admission Type:

- The bar chart illustrates the frequency of diagnoses across different admission types.
- Admission types like "EMERGENCY" likely have higher counts compared to "ELECTIVE."

Solution Architechure:

This architecture outlines the steps and components needed to process, cleanse, and analyse the healthcare data provided.

1. **Data Ingestion Layer:**
   Load and preprocess data for further processing.

   Input**:**
   Source file: health prescription data.csv
   Key columns: TEXT, DIAGNOSIS, CATEGORY,
   and ADMISSION_TYPE

   Components:
   File Storage: Store raw data securely (e.g., on cloud storage or local database).
   Data Loader: Use pandas to load and inspect the dataset.

   Process:
   Verify dataset structure and ensure no missing columns.
   Handle missing values in critical fields (TEXT, DIAGNOSIS).

2. **Data Preprocessing Layer:**
   Clean and transform data for analysis and model training.

   Text Cleaning:
   Remove irrelevant patterns (e.g., [\\.?\\*], metadata like "Admission Date:").
   Tokenization, lemmatization, and removal of stopwords.

   Feature Engineering:
   Combine TEXT and DIAGNOSIS into a single feature (combined_text).
   Extract additional features like text length and diagnosis category distribution.

Anomaly Detection:
Identify and handle outliers in text lengths or unusual patterns.

Tools:
NLP Libraries: NLTK or SpaCy for text preprocessing.
Visualization: Matplotlib and Seaborn for anomaly and distribution insights.

3. Exploratory Data Analysis (EDA):

Understand data patterns, distributions, and potential anomalies.
Visualize text length distribution, word clouds, and token frequency.
Analyse relationship between CATEGORY and ADMISSION_TYPE with diagnoses.
Identify anomalies in text quality (e.g., excessively long or short notes).

4. NLP and AI Model Development:

Build models to detect anomalies and cleanse data.

NLP Techniques:

a. Text Vectorization:
Use TF-IDF, Word2Vec, or pre-trained embeddings (e.g., BERT) to represent text.

b. Anomaly Detection:
Apply Isolation Forest or clustering techniques (e.g., DBSCAN) to detect unusual patterns.

c. Named Entity Recognition (NER):
Extract domain-specific entities like symptoms, medications, and diagnoses.

AI Models:

a. Supervised Learning:
Train models like Random Forest or Logistic Regression on labeled anomalies.

b. Unsupervised Learning:
Use clustering to identify common and anomalous patterns.

5. **Data Cleansing and Transformation:**
Automatically cleanse and enhance the dataset for downstream applications.

Techniques:
Correct spelling and format inconsistencies.
Standardize medical terminology (e.g., SNOMED or ICD codes).
Remove duplicate or redundant entries

6. **Visualization and Monitoring Layer:**
Provide insights and track cleansing performance.

Dashboards:
Visualize cleansing results (e.g., anomaly counts, data quality scores).
Monitor key performance metrics (e.g., model accuracy, runtime efficiency).

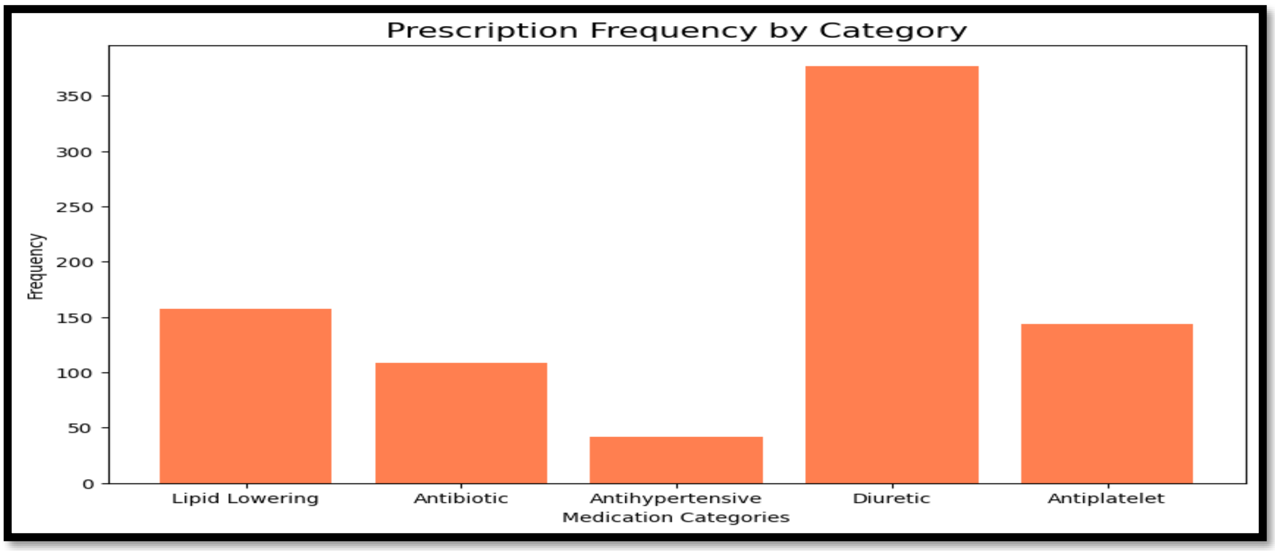
Tools:
Dash or Streamlit for interactive dashboards.
Matplotlib and Seaborn for detailed analysis.

7. **Deployment and Automation:**
Ensure scalable and automated data processing.

Deployment Options:
Local Server: Deploy scripts for batch processing.
Cloud: Use AWS Lambda, Google Cloud Functions, or Azure Logic Apps for scalability.
API Integration: Expose the cleansing solution as an API for real-time processing.

Automation:
Schedule periodic runs to cleanse incoming data.
Generate alerts for anomalies or critical issues.

Visualizations:



Workflow:

1. Store Raw Data: Import the CSV data into the database.
2. Data Cleansing: Perform preprocessing and update records directly in the database.
3. Export for Model Training: Use filtered and processed data for NLP model development.