Streamlit Application for Healthcare Data Analysis

## Overview

This document outlines the structure and functionalities of a Streamlit application designed for healthcare data analysis, focusing on hospital admissions and emergency department operations. The application integrates data fetching, processing, predictive modeling, and user interaction to provide insights into hospital operations, patient admissions, discharges, transfers, and emergency department metrics.

# Data Connections

**(data\_connections.py​​)**

This module handles connections to various data sources, including AWS S3 and Amazon Redshift. It contains functions for fetching admissions data, mortality features, and ECG features from Redshift, as well as loading pickled Python objects from S3. These functions are crucial for retrieving the data needed for analysis and predictions within the application.

* **Redshift Database Access**: It interacts with a Redshift database to perform SQL queries for fetching admissions, emergency department mortality, and ECG data. The database credentials are hardcoded, which is not a best practice for security reasons.
* **S3 Data Retrieval**: It retrieves serialized objects stored in S3, such as machine learning models or transformer objects, using boto3.

# Pre-Processing and Predictions

**(pre\_processing.py​​ and predict\_and\_decode.py )**

* **pre\_processing.py**: This file contains functions for pre-processing the data retrieved from the data sources. It includes data cleaning, encoding, and preparation steps tailored for mortality predictions and ECG data analysis. Moreover, it uses OpenAI's GPT model for generating insights from admissions, discharges, and transfers forecasted data, integrating natural language processing to derive meaningful interpretations and actions from the data.
* **predict\_and\_decode.py**: Defines a **PredictAndDecode** class that wraps around a machine learning model and its associated label encoder. It's designed to make predictions using the model and then decode these predictions into a human-readable format using the label encoder. This might be used for categorizing input data into predefined classes and providing outputs in a more interpretable form.

# Security and Best Practices

* **Error Handling**: The application includes basic error handling to manage failures in database connections or data processing. However, ensuring comprehensive error handling throughout can improve the robustness and user experience.
* **Code Organization**: The application's functionality is modularized into specific scripts for data connections, pre-processing, and predictions, which aids in maintainability and scalability. However, further refactoring could be done to separate configuration details (e.g., database credentials) from the code logic.

# Application Flow

1. **Data Retrieval**: The application begins by fetching necessary data from AWS S3 and Amazon Redshift, depending on the operation being performed (e.g., admissions analysis, mortality prediction).
2. **Pre-Processing**: The retrieved data is then cleaned, transformed, and encoded to match the input requirements of the predictive models or for direct analysis.
3. **Predictive Modeling and Insight Generation**: Pre-processed data is used for predictions or generating insights. This includes using machine learning models for healthcare predictions and leveraging GPT for text-based insights generation.
4. **User Interface Interaction**: While not directly visible in the provided scripts, interactions with a Streamlit user interface likely serve as the entry point for user requests, displaying predictions and insights derived from the data.

# Admissions

In the "Admissions" section of the Streamlit application, users can interact with data related to hospital admissions. This might include analyzing forecasted admissions, discharges, and transfers, and generating insights to assist in hospital administration and planning.

Workflow:

* **Data Fetching**: The application uses the **data\_connection\_admissions** function from **data\_connections.py** to retrieve admissions data for a specified date range from a Redshift database.
* **Data Processing and Analysis**:
  + The retrieved data may undergo pre-processing using the **pre\_process\_adt\_insights** function from **pre\_processing.py**, which formats the data and possibly generates insights using GPT models.
  + This processed information can provide predictive insights into patient admissions, discharges, and transfers, which are critical for resource allocation and staffing.
* **Insight Generation**: Utilizing the GPT model, the application generates textual insights that are easy for hospital administrators to understand and act upon, based on the forecasted data.
* **User Interface**: Within Streamlit, this section would display the fetched and processed data in a user-friendly format, possibly including tables and visualizations. Additionally, generated insights could be presented in a readable format, offering administrators actionable information.

# 2. Emergency Department Section

In the "Emergency Department" section, users can explore and analyze data related to emergency department visits, including mortality rates, patient demographics, and other relevant metrics.

Workflow:

* **Data Fetching**:
  + **data\_connection\_ed\_mortality** and **data\_connection\_ed\_ecg** functions in **data\_connections.py** could be used to fetch data related to mortality and ECGs in the emergency department, respectively.
* **Data Processing and Feature Preparation**:
  + The **pre\_process\_mortality** and **pre\_process\_ecg** functions in **pre\_processing.py** prepare the mortality and ECG data for analysis. This includes data cleaning, normalization, and feature encoding.
  + **mortality\_features** and **ecg\_features** functions might be used to fetch additional features necessary for analysis or predictive modeling.
* **Predictive Modeling**:
  + For mortality predictions, pre-processed data might be fed into a predictive model loaded from S3. The model could use patient data to predict outcomes, which are then decoded for readability using the **PredictAndDecode** class.
  + Similar processes could be applied to ECG data for predicting relevant health outcomes or anomalies.
* **User Interface**:
  + This section would present the emergency department data and predictions to the user, likely including statistical summaries, charts, and predictive insights.
  + Users could filter data by date range, patient demographics, or other relevant metrics to gain specific insights into emergency department operations and patient outcomes.