Getting Started

GitHub Link

Access these documents and files to run this notebook: clinical-fusion-master

- Includes all required files except the MIMIC-III dataset (see About the Data section for obtaining the data).
- Only a small subset of the Processed data (data/processed) is uploaded due to sizing requirements.
- After copying the repo, you may proceed to the next section of this notebook.

Please keep in mind that the estimated memory for this project at minimum will require 100 GB. Therefore, the team has provided guidance and steps on processing the data (you're welcome to follow along) and the results are/will be provided for your reference (all produced results will be housed in the GitHub link).

Video Link

Access the final video recording saved on our Google Drive.

Important Notices

Original adaptation from Combining structured and unstructured data for predictive models: a deep learning approach (clinical-fusion).

Copyright Notices: For the protection of copyright and to ensure compliance with legal standards, this repository utilizes only the file names and does not host, display, or distribute the actual data. All examples, code, and references within this repository strictly involve file names as placeholders. Users are advised to adhere to copyright laws when using actual data corresponding to these file names.

Mount Notebook to Google Drive

Upload the data, GitHub repo, figures, etc to your Google Drive, then mount this notebook to Google Drive. After that, you can access the resources freely.

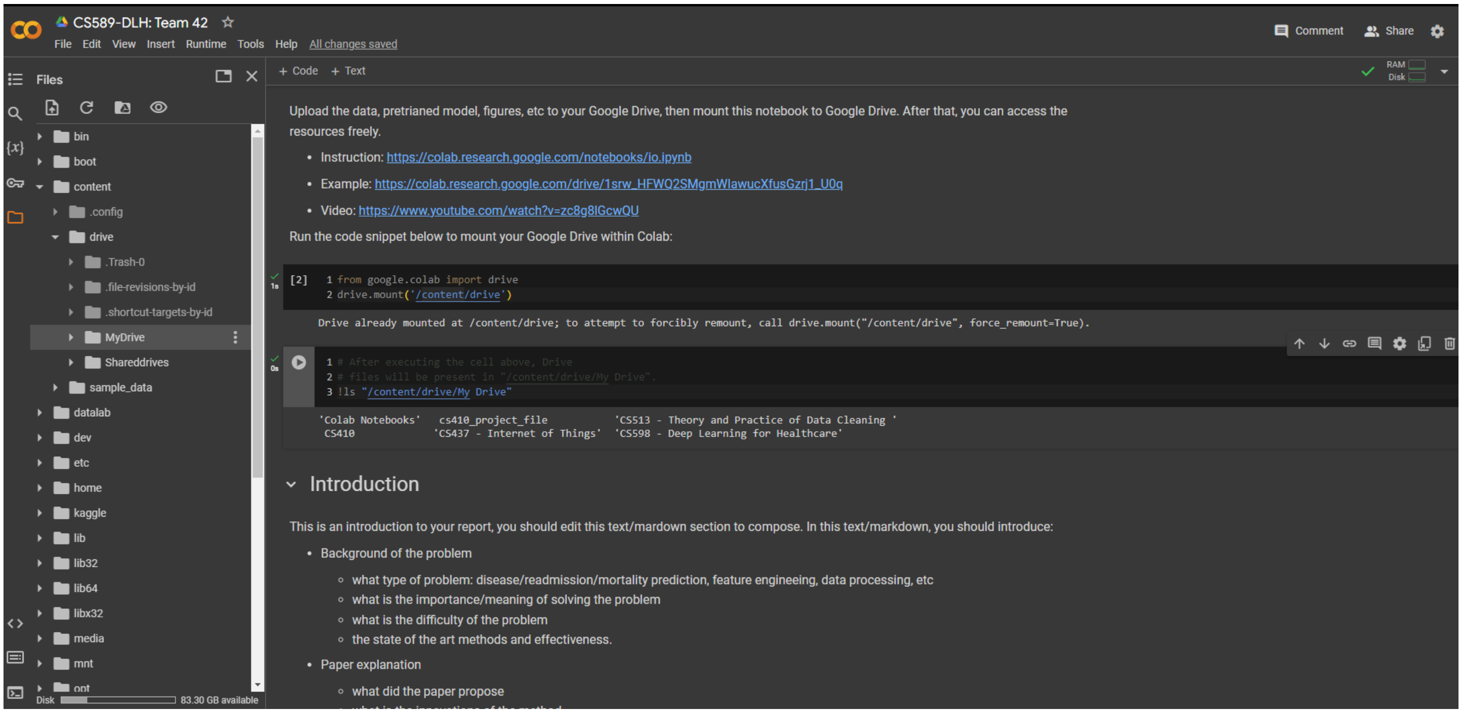
- Instruction: https://colab.research.google.com/notebooks/io.ipynb
- Example: https://colab.research.google.com/drive/1srw_HFWQ2SMgmWIawucXfusGzrj1_U0q
- Video: https://www.youtube.com/watch?v=zc8g8lGcwQU

Run the code snippet below to mount your Google Drive within Colab:

```
from google.colab import drive
drive.mount('/content/drive')


# After executing the cell above, Drive
# files will be present in "/content/drive/My Drive".
!ls "/content/drive/My Drive"
```

You can also use the file browser in the sidebar to browse Drive files.



Upload the data, pretrianed model, figures, etc to your Google Drive, then mount this notebook to Google Drive. After that, you can access the resources freely.

- Instruction: https://colab.research.google.com/notebooks/io.ipynb
- Example: https://colab.research.google.com/drive/1srw_HFWQ2SMgmWIawucXfusGzrj1_U0q
- Video: https://www.youtube.com/watch?v=zc8g8lGcwQU

Run the code snippet below to mount your Google Drive within Colab:

```
[2] from google.colab import drive
    drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
# After executing the cell above, Drive
# files will be present in "/content/drive/My Drive".
!ls "/content/drive/My Drive"

'Colab Notebooks'   cs410_project_file           'CS513 - Theory and Practice of Data Cleaning '
 CS410             'CS437 - Internet of Things'  'CS598 - Deep Learning for Healthcare'
```

Introduction

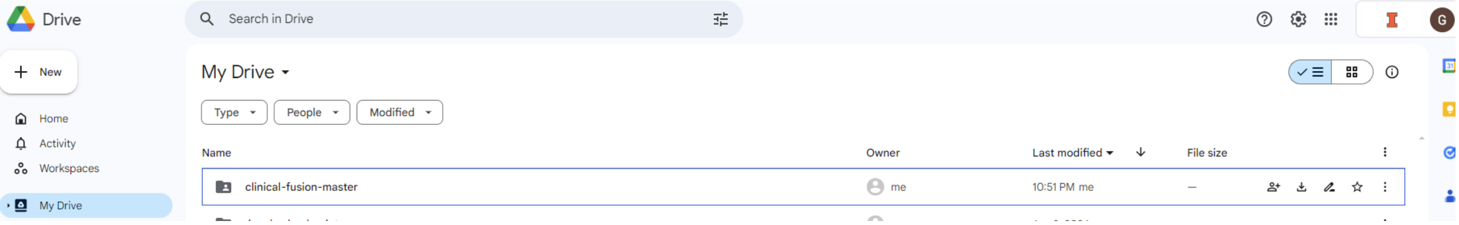
This is an introduction to your report, you should edit this text/mardown section to compose. In this text/markdown, you should introduce:

- Background of the problem
  - what type of problem: disease/readmission/mortality prediction, feature engineeing, data processing, etc
  - what is the importance/meaning of solving the problem
  - what is the difficulty of the problem
  - the state of the art methods and effectiveness.
- Paper explanation
  - what did the paper propose
  - what is the innovations of the method

Prerequisites

Download Repository

Before you begin, make sure you can access the clinical-fusion-master repository and clone the files to your local machine. Upload the repo to your GDrive under "My Drive":



Name	Owner	Last modified	File size
clinical-fusion-master	me	10:51 PM me	—

You should be able to access the repository from this Notebook.

Required Dependencies

Please ensure you have the latest versions of the softwares and libraries to run the code efficiently:

- Python 3.6.10
- Gensim 3.8.0
- NLTK: 3.4.5 (NLTK requires Python versions 3.7, 3.8, 3.9, 3.10 or 3.11.)
- Numpy: 1.14.2
- Pandas: 0.25.3
- Scikit-learn: 0.20.1
- Tqdm: 4.42.1
- PyTorch: 1.4.0
- PostgreSQL (newest version)

Sample commands below to run in your command line or Powershell if you are missing requirements.

```
pip install nltk
```

If you're using a specific version of Python with the Python Launcher:

```
py -3.x -m pip install nltk
```

Introduction

Healthcare industries have access to vast amounts of data that can be used to improve patient outcomes and care. Specifically, EHRs contain a large amount of patient information in both structured (e.g. patient demographics, vital signs, lab tests) and unstructured (e.g. clinical notes) data format. However, most existing predictive modeling research focuses on using only structured data or unstructured data [1]. While structured data is easily quantifiable and analyzable, unstructured data contains nuanced information that, when properly interpreted, offers a deeper insight into patient conditions and outcomes. The challenge, however, lies in effectively integrating these disparate data types to leverage their full potential in enhancing predictive modeling in healthcare.

In this research, the author seeks to address the development of effective methods to fuse and learn from structured and unstructured EHR data to improve patient representation and risk prediction of important clinical outcomes such as in-hospital mortality, 30-day hospital readmission, and long-length of stay prediction [1]. It proposes two innovative deep learning frameworks, Fusion-CNN (Convolutional Neural Network) and Fusion-LSTM (Long Short-Term Memory), designed to harness the complementary strengths of sequential clinical notes (unstructured data) and temporal signals (structured data). By fusing these data modalities, the proposed models aim to capture a more comprehensive picture of patient health trajectories.

Scope of Reproducibility

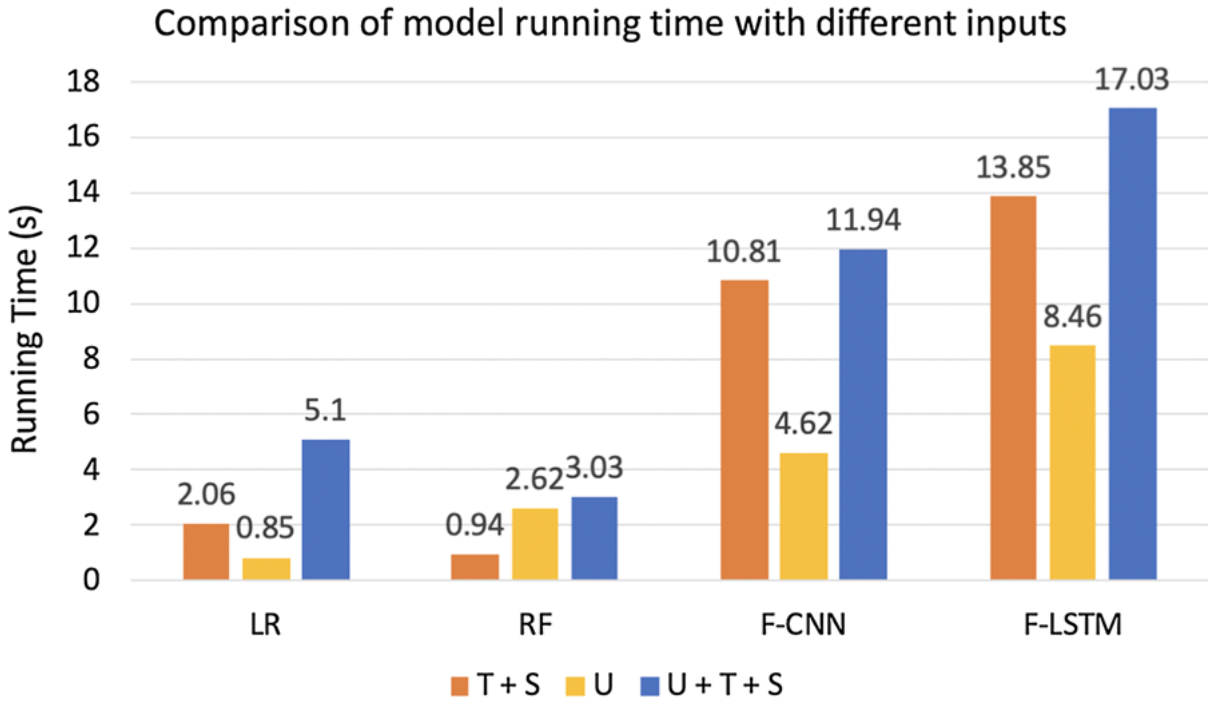
Hypothesis 1: Fusion-CNN and Fusion-LSTM outperform traditional machine learning models

- Experiments:
    - Implement traditional machine learning model baselines, namely logistic regression (LR) and random forest (RF).
    - Compare the performance of traditional ML baselines with Fusion-CNN and Fusion-LSTM on each prediction task using the evaluation metrics
        - F1 score
        - AUROC

        - AUPRC

Hypothesis 2: Models could improve performance on predictions by combining unstructured text and structured data compared to using either data type alone.

- Experiments:
    - Use structured data (temporal signal + static information) to train and evaluate Fusion-CNN and Fusion-LSTM model.
    - Use unstructured data to train and evaluate Fusion-CNN and Fusion-LSTM model.
    - Use structured data and unstructured data to train and evaluate Fusion-CNN and Fusion-LSTM model.
    - For each experiment task, do the following prediction tasks and performance comparison:
        - In-hospital mortality prediction on MIMIC-III
        - Long length of stay prediction on MIMIC-III
        - 30-day readmission prediction on MIMIC-III.
        - Compare the performance of 2 baseline methods (logistic regression, random forest) and Fusion-CNN and Fusion-LSTM when using both data types VS using only structured data or only unstructured data.
    - Evaluation of the models using:
        - F1 score
        - AUROC
        - AUPRC



Methodology

The specific approach proposed in this paper is to use two multimodal deep learning architectures, namely Fusion-CNN and Fusion LSTM, to combine structured and unstructured data from EHRs for patient representation learning and predictive modeling:

- **Dataset**: Medical Information Mart for Intensive Care III (MIMIC-III).
- **Data Inputs**: Static information (demographic information, admission-related information), temporal signals (vital signs, lab tests), and sequential clinical notes.
- **Models:**
    - **Fusion-CNN architecture:** the Fusion-CNN framework is adept at combining unstructured textual data with structured temporal and static data. The approach uses a sophisticated pipeline involving several stages:
        - Unstructured Text Processing with Doc2Vec: The model begins by transforming sequential clinical notes into vector representations using Doc2Vec. This unsupervised algorithm learns fixed-length feature representations from variable-length pieces of texts, such as sentences, paragraphs, and documents. By capturing the semantic meaning of words in context, Doc2Vec provides a rich, condensed encoding of the unstructured notes.
        - Convolutional Layers: These vector representations are then processed through convolutional layers. In the context of text, convolutional neural networks (CNNs) apply a series of filters to the embeddings to capture local features – analogous to identifying phrases or concepts that are significant for prediction tasks within clinical notes.
        - Max-Pooling Layers: Following convolution, max-pooling layers are employed to reduce dimensionality and to extract the most salient features from the convolution layer's output. This step enhances the model’s ability to focus on the most important local features that are indicative of the outcome.
        - Flatten and Concatenation: The output from pooling layers is flattened and concatenated with processed structured data to form a comprehensive patient representation. This includes temporal signals such as vital signs and lab results over time, as well as static structured information like age, gender, or

        specific medical diagnoses.
        - Outcome Prediction: The final patient representation, which now encapsulates a multi-dimensional view of the patient's health state, is passed to the output layers. These layers are responsible for making the final predictions on outcomes such as in-hospital mortality, length of stay, or likelihood of readmission.
    - **Fusion-LSTM architecture**: the Fusion-LSTM framework leverages the power of LSTM networks to handle sequential and time-series data effectively:
        - Unstructured Text Processing with Doc2Vec: Similar to Fusion-CNN, it starts by converting clinical notes into vector representations using Doc2Vec.
        - LSTM Layers: The vectors are then fed into LSTM layers. LSTMs are particularly well-suited for modeling time-series data because of their ability to maintain long-term dependencies. They can remember information for a long period, which is crucial when dealing with temporal patterns in clinical data.
        - Pooling: The LSTM outputs are pooled to condense the information, capturing the most relevant temporal features for the task at hand.
        - Concatenation with Structured Data: The pooled LSTM outputs are combined with the encoded static structured data. This is achieved by encoding the structured data, often through an embedding or encoding layer that transforms static variables into a format that is compatible for fusion with the LSTM outputs.
        - Outcome Prediction: The concatenated representation forms a comprehensive patient profile that is passed to the final prediction layers, generating predictions for the targeted clinical outcomes.
- **Evaluation**: Performance is evaluated and compared against baseline models (logistic regression and random forest), Fusion-CNN and Fusion-LSTM.

About the Data

MIMIC-III is a comprehensive database of deidentified health data from over 40,000 ICU patients at Beth Israel Deaconess Medical Center between 2001-2012. It includes demographics, vital signs, lab tests, medications, and mortality data, supporting a range of studies in epidemiology, clinical decision-making, and tool development. The database integrates data from various sources and is freely available to researchers who complete required training and agree to a data use agreement.

For more detailed information, please refer to the PhysioNet website: MIMIC-III Clinical Database v1.4.

Data Description

MIMIC-III is a relational database consisting of 26 tables. Tables are linked by identifiers which usually have the suffix ‘ID’. For example, SUBJECT_ID refers to a unique patient, HADM_ID refers to a unique admission to the hospital, and ICUSTAY_ID refers to a unique admission to an intensive care unit.

Charted events such as notes, laboratory tests, and fluid balance are stored in a series of ‘events’ tables. For example the OUTPUTEVENTS table contains all measurements related to output for a given patient, while the LABEVENTS table contains laboratory test results for a patient.

Tables prefixed with ‘D_’ are dictionary tables and provide definitions for identifiers. For example, every row of CHARTEVENTS is associated with a single ITEMID which represents the concept measured, but it does not contain the actual name of the measurement. By joining CHARTEVENTS and D_ITEMS on ITEMID, it is possible to identify the concept represented by a given ITEMID.

Developing the MIMIC data model involved balancing simplicity of interpretation against closeness to ground truth. As such, the model is a reflection of underlying data sources, modified over iterations of the MIMIC database in response to user feedback. Care has been taken to avoid making assumptions about the underlying data when carrying out transformations, so MIMIC-III closely represents the raw hospital data.

Broadly speaking, five tables are used to define and track patient stays: ADMISSIONS; PATIENTS; ICUSTAYS; SERVICES; and TRANSFERS. Another five tables are dictionaries for cross-referencing codes against their respective definitions: D_CPT; D_ICD_DIAGNOSES; D_ICD_PROCEDURES; D_ITEMS; and D_LABITEMS. The remaining tables contain data associated with patient care, such as physiological measurements, caregiver observations, and billing information.

In some cases it would be possible to merge tables—for example, the D_ICD_PROCEDURES and CPTEVENTS tables both contain detail relating to procedures and could be combined—but our approach is to keep the tables independent for clarity, since the data sources are significantly different. Rather than combining the tables within MIMIC data model, we suggest researchers develop database views and transforms as appropriate.

Instructions to Download Data:

Total uncompressed size: 6.2 GB.

Prerequisites: Prior to requesting access to MIMIC, you must become a credentialed user on PhysioNet AND complete the required training. Instructions for the credentialing process and the training are provided on PhysioNet.

MIMIC-III is made available via Physionet. Beyond directly downloading the dataset from PhysioNet, there are a few mechanisms for accessing the data:

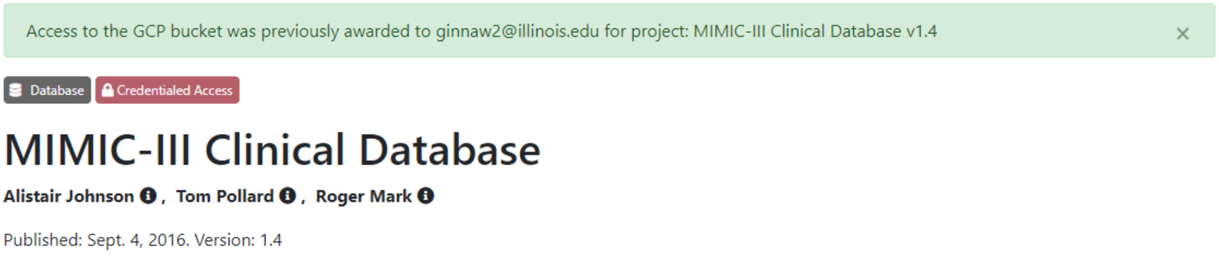
- Accessing the data in BigQuery
- Accessing the data on AWS
- Accessing the data in a Google Cloud storage bucket

Although Physionet recommends accessing the data via BigQuery, due to budget constraints and space limitation which requires a minimum 100 GB of local storage to process the data, the team has opted to access the demo version of the dataset through Google Cloud storage bucket and downloading locally to our machines.

- There are three steps to accessing data on the cloud:
    1. Link your cloud account to your PhysioNet profile.
    2. Request access to the cloud resource (see second bullet under *Steps to Access Files*).
    3. Log-in to the appropriate service and navigate to the resource.

Steps to Access Files:

- Download the ZIP file (6.2 GB) located here. Data download is estimated to be around 3 hours. The demo version (13.4 MB) can be found here and is significantly faster to download.
- Request access to the files using the Google Cloud Storage Browser here. Login with a Google account is required.
    - After gaining access you should see a confirmation notice:



Access to the GCP bucket was previously awarded to ginnaw2@illinois.edu for project: MIMIC-III Clinical Database v1.4 ×

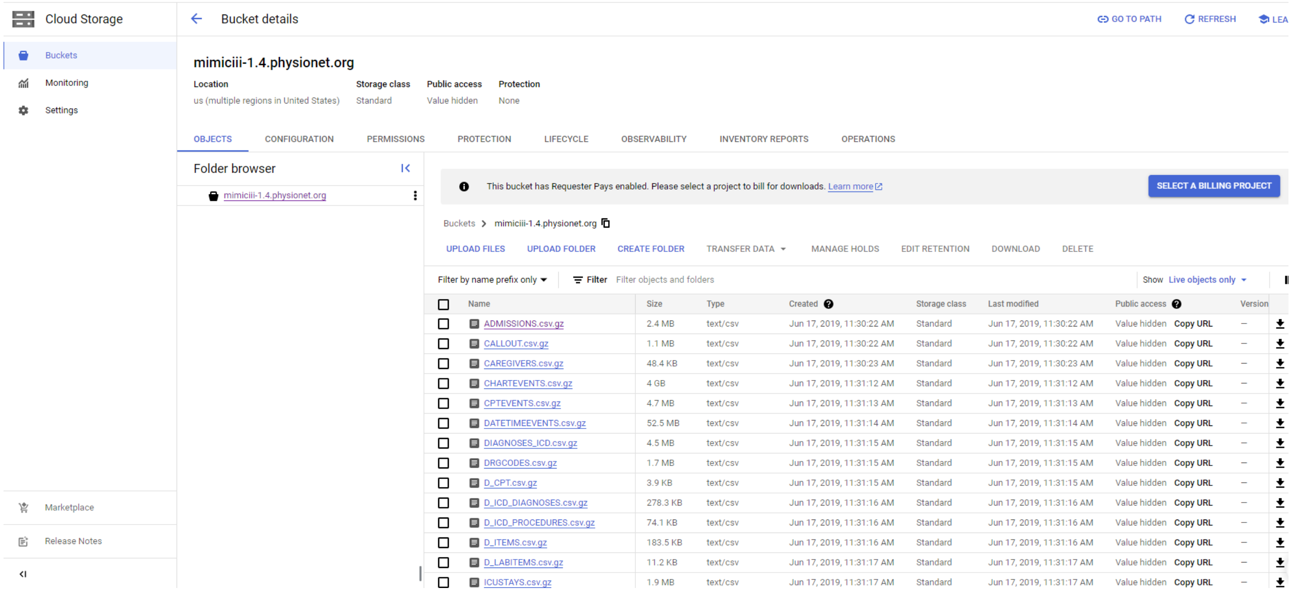
Database Credentialed Access

MIMIC-III Clinical Database

Alistair Johnson, Tom Pollard, Roger Mark

Published: Sept. 4, 2016. Version: 1.4

    - Click the link to the Google Cloud Storage and you should see this page:



mimiciii-1.4.physionet.org

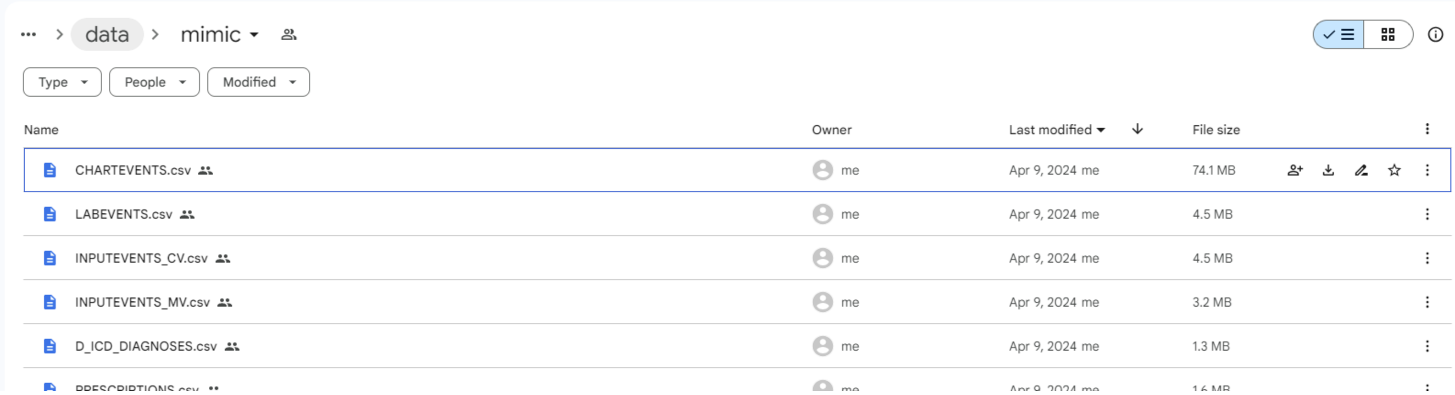
Location	Storage class	Public access	Protection
us (multiple regions in United States)	Standard	Value hidden	None

This bucket has Requester Pays enabled. Please select a project to bill for downloads. Learn more

□	Name	Size	Type	Created	Storage class	Last modified	Public access		Version
□	ADMISSIONS.csv.gz	2.4 MB	text/csv	Jun 17, 2019, 11:30:22 AM	Standard	Jun 17, 2019, 11:30:22 AM	Value hidden	Copy URL	–
□	CALLOUT.csv.gz	1.1 MB	text/csv	Jun 17, 2019, 11:30:22 AM	Standard	Jun 17, 2019, 11:30:22 AM	Value hidden	Copy URL	–
□	CAREGIVERS.csv.gz	48.4 KB	text/csv	Jun 17, 2019, 11:30:23 AM	Standard	Jun 17, 2019, 11:30:23 AM	Value hidden	Copy URL	–
□	CHARTEVENTS.csv.gz	4 GB	text/csv	Jun 17, 2019, 11:31:12 AM	Standard	Jun 17, 2019, 11:31:12 AM	Value hidden	Copy URL	–
□	CPTEVENTS.csv.gz	4.7 MB	text/csv	Jun 17, 2019, 11:31:13 AM	Standard	Jun 17, 2019, 11:31:13 AM	Value hidden	Copy URL	–
□	DATETIMEEVENTS.csv.gz	52.5 MB	text/csv	Jun 17, 2019, 11:31:14 AM	Standard	Jun 17, 2019, 11:31:14 AM	Value hidden	Copy URL	–
□	DIAGNOSES_ICD.csv.gz	4.5 MB	text/csv	Jun 17, 2019, 11:31:15 AM	Standard	Jun 17, 2019, 11:31:15 AM	Value hidden	Copy URL	–
□	DRGCODES.csv.gz	1.7 MB	text/csv	Jun 17, 2019, 11:31:15 AM	Standard	Jun 17, 2019, 11:31:15 AM	Value hidden	Copy URL	–
□	D_CPT.csv.gz	3.9 KB	text/csv	Jun 17, 2019, 11:31:15 AM	Standard	Jun 17, 2019, 11:31:15 AM	Value hidden	Copy URL	–
□	D_ICD_DIAGNOSES.csv.gz	278.3 KB	text/csv	Jun 17, 2019, 11:31:16 AM	Standard	Jun 17, 2019, 11:31:16 AM	Value hidden	Copy URL	–
□	D_ICD_PROCEDURES.csv.gz	74.1 KB	text/csv	Jun 17, 2019, 11:31:16 AM	Standard	Jun 17, 2019, 11:31:16 AM	Value hidden	Copy URL	–
□	D_ITEMS.csv.gz	183.5 KB	text/csv	Jun 17, 2019, 11:31:16 AM	Standard	Jun 17, 2019, 11:31:16 AM	Value hidden	Copy URL	–
□	D_LABITEMS.csv.gz	11.2 KB	text/csv	Jun 17, 2019, 11:31:17 AM	Standard	Jun 17, 2019, 11:31:17 AM	Value hidden	Copy URL	–
□	ICUSTAYS.csv.gz	1.9 MB	text/csv	Jun 17, 2019, 11:31:17 AM	Standard	Jun 17, 2019, 11:31:17 AM	Value hidden	Copy URL	–

- Access the data using the Google Cloud command line tools (please refer to the gsutil documentation for guidance): `gsutil -m -u YOUR_PROJECT_ID cp -r gs://mimiciii-demo-1.4.physionet.org DESTINATION`
- Download the files using your terminal: `wget -r -N -c -np https://physionet.org/files/mimiciii-demo/1.4/`

Congratulations! You've successfully downloaded the dataset! Now, load the dataset to **clinic-fusion-master>data>MIMIC** folder on your Google Drive to access the data from the Notebook.



Name	Owner	Last modified	File size
CHARTEVENTS.csv	me	Apr 9, 2024 me	74.1 MB
LABEVENTS.csv	me	Apr 9, 2024 me	4.5 MB
INPUTEVENTS_CV.csv	me	Apr 9, 2024 me	4.5 MB
INPUTEVENTS_MV.csv	me	Apr 9, 2024 me	3.2 MB
D_ICD_DIAGNOSES.csv	me	Apr 9, 2024 me	1.3 MB
PRESCRIPTIONS.csv	me	Apr 9, 2024 me	1.6 MB

Data Processing

Prerequisites:

1. **Understand SQL** before starting with PostgreSQL, as it's important to have a solid understanding of the language since SQL is the foundation for managing and manipulating databases. PostgreSQL is an SQL-based system, and knowing SQL will greatly aid in effectively using PostgreSQL's features. Familiarize yourself with SQL!
2. **Download PostgreSQL** following the instructions provided. Make sure to download the correct version for your machine.
   - Run the installer file as an administrator. The installer will guide you through several steps. You'll be prompted to choose the install location, the components to install (such as the PostgreSQL Server, pgAdmin, Stack Builder, and command-line tools), and the data directory where your databases will be stored.

     NOTE: *Installation process may take a few minutes*.

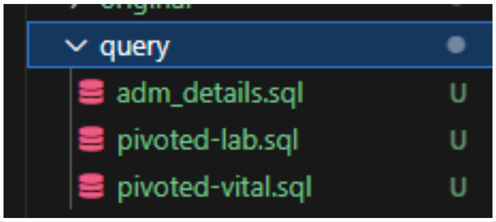
Preprocessing

Optional:

To begin processing the data for Clinical-Fusion, we need to build the MIMIC-III dataset locally using Postgres. You can follow the instructions provided by Physionet here to examine the dataset further prior to building the dataset. The instructions tutorial provides an introduction to the structure and content of the MIMIC-III database. Although it is not necessary to go through the postgre queries in the link, you should make sure you have a good understanding of the MIMIC-III dataset.

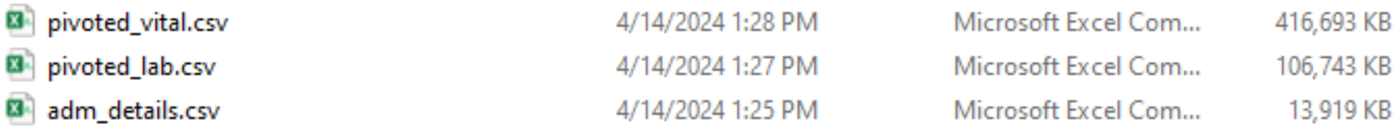
Step 1: Use PostgreSQL to Build adm_details.csv, pivoted-lab.csv, and pivoted-vital.csv

Process portion of this code requires you to run each python sequentially in order to process the MIMIC-III dataset and train the model. Before running these files, you will need run the following queries to generate dependencies files.



```
> original
v query
    adm_details.sql      U
    pivoted-lab.sql      U
    pivoted-vital.sql    U
```

Launch Postgre to run the adm_details.sql, pivoted-lab.sql, and pivoted-vital.sql. You should see three new cvs files generated in data folder.



pivoted_vital.csv	4/14/2024 1:28 PM	Microsoft Excel Com...	416,693 KB
pivoted_lab.csv	4/14/2024 1:27 PM	Microsoft Excel Com...	106,743 KB
adm_details.csv	4/14/2024 1:25 PM	Microsoft Excel Com...	13,919 KB

Step 2: Run Preprocessing Python Scripts Below

Run the files below in sequential order. A series of files will be generated under the "processed" folder. You may need to generate a new folder "imgs" to allow the script to save graphs produced (this is required for 01_get_signals.py).

This step may take some time especially when running *05_preprocess.py*.

```
$ python 00_define_cohort.py # define patient cohort and collect labels
$ python 01_get_signals.py # extract temporal signals (vital signs and laboratory
$ python 02_extract_notes.py --firstday # extract first day clinical notes
$ python 03_merge_ids.py # merge admission IDs
$ python 04_statistics.py # run statistics
$ python 05_preprocess.py # run preprocessing
$ python 06_doc2vec.py --phase train # train doc2vec model
$ python 06_doc2vec.py --phase infer # infer doc2vec vectors
```

Step 3: Build the Model

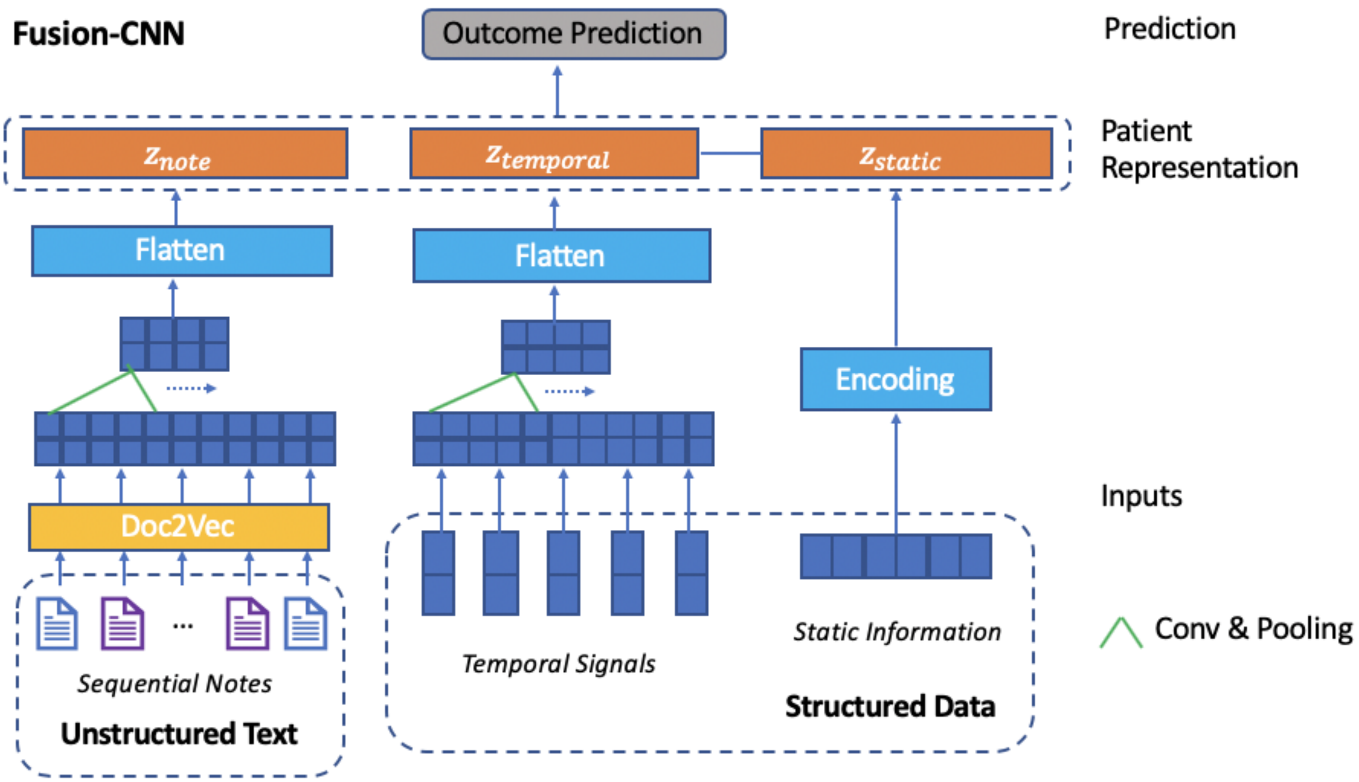
Refer to the Model section for understanding and building the model.

Model

Fusion-CNN:

- **Model Architecture:** The Fusion-CNN model is a deep learning model that combines structured EHR data and unstructured clinical text records to learn a unified patient representation. It contains five main components: static information encoder, temporal signal embedding, sequential notes representation, patient representation, and output layer.
    - Static information encoder: encodes static categorical features as one-hot vectors. The output of the encoder z_static=[z_demo; z_amd] is part of the final patient representation.
    - Temporal signal representation: Fusion-CNN uses a 2-layer CNN and max-pooling to extract deep features from temporal signals.
    - Sequential notes representation: Fusion-CNN uses Doc2Vec to map each clinical note to a vector. The sequence of document vectors is then passed through convolutional layers and max-pooling layers to learn the text representation.
    - Patient representation: the final patient representation z_p=[z_static; z_temporal; z_text] .
    - Output layer: the patient representation is fed into the final output layer and makes predictions.
- **Training Objective:**
    - Loss function: binary cross entropy loss.
    - Optimizer: Adam optimizer with a learning rate of 0.0001.

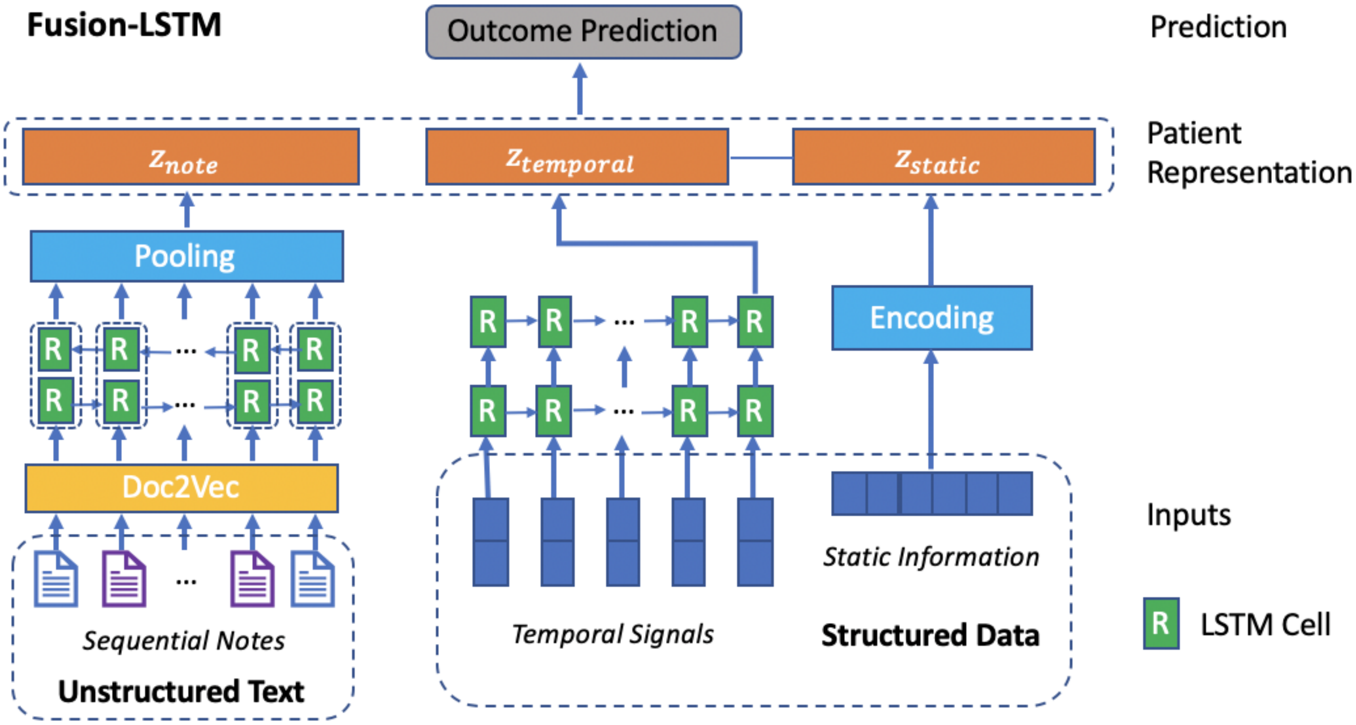
- **Others:**
    - No explicit use of pre-trained models (except for Doc2Vec).



Fusion-LSTM:

- **Architecture:** The Fusion-LSTM model is another deep learning model that combines structured EHR data and unstructured clinical texts. Similar to Fusion-CNN, it has five main components: static information encoder, temporal signal embedding, sequential notes representation, patient representation, and output layer.
    - Static information encoder: encodes static categorical features as one-hot vectors. The output of the encoder z_static=[z_demo; z_amd] is part of the final patient representation.
    - Temporal signal representation: Fusion-LSTM uses a 2-layer LSTM to extract the representations of temporal signals.
    - Sequential notes representation: Fusion-LSTM starts by mapping clinical notes using Doc2Vec, and then passes the sequence of document vectors through a bidirectional layer (BiLSTM). A max-pooling layer aggregates the hidden states of BiLSTM layer to create the final text representation.
    - Patient representation: same as Fusion-CNN, z_p=[z_static; z_temporal; z_text]
    - Output layer: same as Fusion-CNN.
- **Training Objective:**

    - Loss function: binary cross entropy loss.
    - Optimizer: Adam optimizer with a learning rate of 0.0001.
- **Others:**
    - No explicit use of pre-trained models (except for Doc2Vec).



Metrics

F1 Score: A measure of a test's accuracy on a classification task. It considers both the precision (the proportion of positive predictions that were correct) and recall (the proportion of actual positives that were identified correctly) of the model. A higher F1 score indicates better overall performance.

AUROC (Area Under the Receiver Operating Characteristic Curve): A performance metric for classification tasks that uses a ROC curve. The ROC curve plots the **true positive rate (TPR)** against the **false positive rate (FPR)** at all classification thresholds. The AUROC represents the total area underneath the ROC curve. A higher AUROC indicates better performance, with an AUROC of 1 corresponding to a perfect test.

AUPRC (Area Under the Precision-Recall Curve): Another performance metric for classification tasks that uses a Precision-Recall (PR) curve. The PR curve plots the precision against the recall for all classification thresholds. The AUPRC represents the total area underneath the PR curve. A higher AUPRC indicates that the model is better at ranking the cases with the most positive outcomes at the top.

Model Implementation

We have implemented two deep learning models:

- **Fusion-CNN**: A Convolutional Neural Network for capturing spatial dependencies.
- **Fusion-LSTM**: A Long Short-Term Memory network for capturing long-term dependencies.

Both models were built using the PyTorch library, known for its flexibility and efficiency in training deep learning models.

Hyperparameters

To train our models, we fine-tuned the following hyperparameters:

- **Activation Function**: We used the Rectified Linear Unit (ReLU) for non-linear transformations within the neural networks.
- **Batch Size**: Our models were trained with a batch size of 64, which offered a good trade-off between training speed and model performance.
- **Epochs**: Training was conducted for 10 epochs as opposed to 50 in the original experiment due to limited time for this project.
- **Learning rate**: We used a learning rate of 0.0001 for training our models.

Computational Requirements

The training of our models was carried out on Google Colab, which provides a robust and accessible platform for running computationally intensive tasks.

- **Environment**: We utilized the Google Compute Engine backend, which comes with a pre-configured Python 3 environment.
- **Memory**: The virtual machines on Google Colab provided us with 12.7 GB of RAM, ensuring that our models were trained without memory constraints.
- **GPU Utilization**: For our experiments, GPU resources were not required, demonstrating the efficiency of our models and training procedure.

Baseline Evaluation

The `baselines.py` script implements Logistic Regression and RandomForest classifiers from the scikit-learn library as baseline models. These models serve as a comparison standard for more complex approaches like Fusion-CNN and Fusion-LSTM. The script includes code for parsing arguments, setting up hyperparameters for the models, and applying grid search with cross-validation to find the best hyperparameter settings. It also calculates and prints out model performance metrics after training and testing.

```
$ python baselines.py --model [model] --task [task] --inputs [inputs]
```

The `run_baselines.sh` bash script is designed to facilitate the execution of baseline models for various clinical prediction tasks. It uses `baselines.py` to perform the tasks with different data inputs. The tasks include:

- mortality
- length of stay (los_bin)
- readmission prediction

Each task is run with different input combinations signified by the --inputs argument, which indicates the type of data used (e.g., temporal (T), unstructured (U), structured (S)):

- 3: temporal (T) + structured (S)
- 4: unstructured (U)
- 7: temporal (T) + structured (S) + unstructured (U)

The script showcases how to systematically evaluate baseline models against different data features and prediction tasks. It returns a tuple of three performance metrics based on the true labels and the predicted probabilities: F1-score, AUC (Area Under the Receiver Operating Characteristic curve), and AUPRC (Area Under the Precision-Recall Curve).

F1 Score

- F1 score is a machine learning evaluation metric to measure the accuracy of a model. It provides a balanced measure of model performance, taking both false positives and false negatives into account. The F1 score is a useful secondary metric that provides a single number to summarize model performance.
- F1 Score range: 0 ~ 1, where 1 represents perfect precision and recall.
- The F1 score is defined based on the precision and recall scores, which are

mathematically defined as follows:

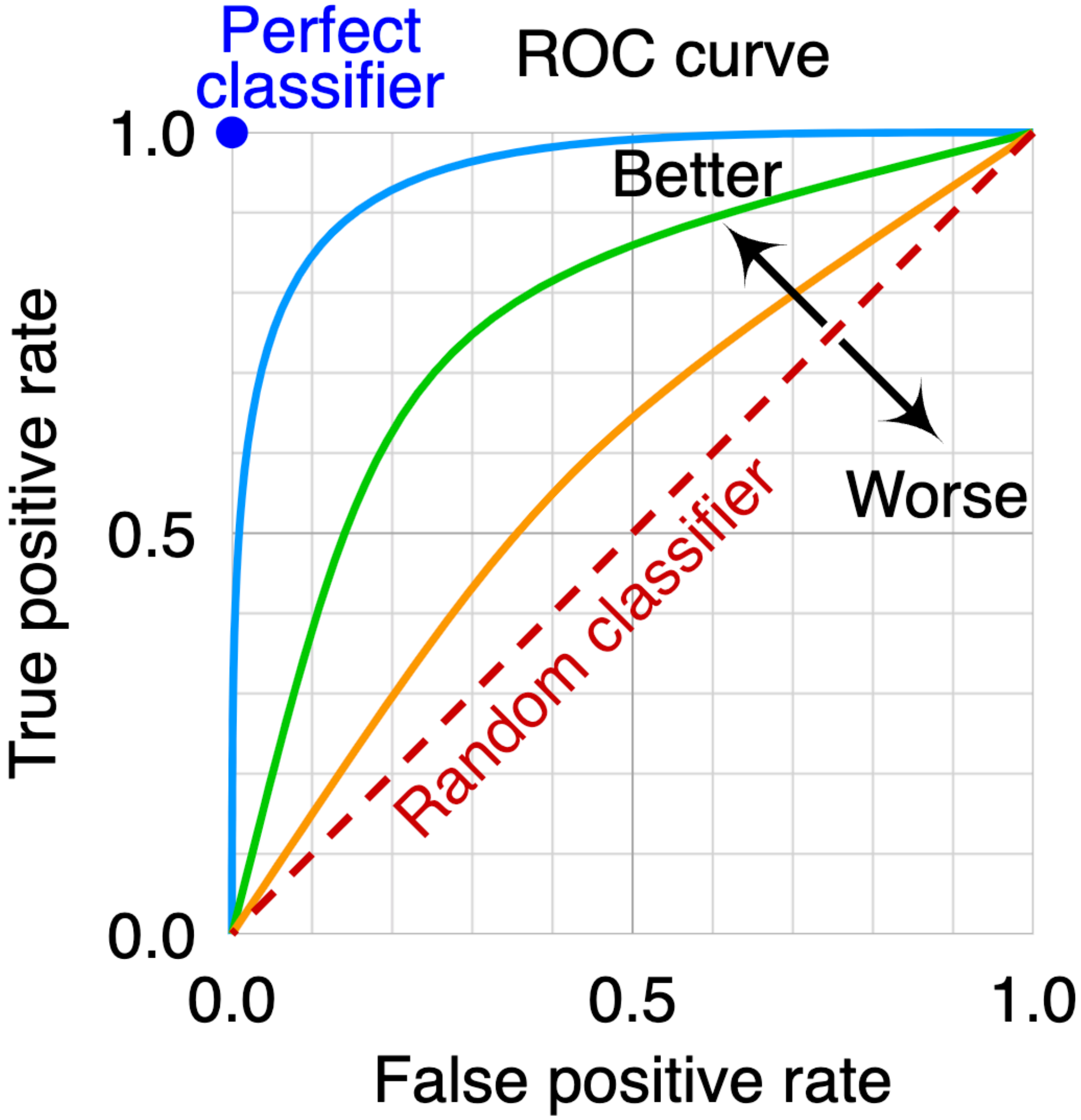
$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1 Score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

AUROC (area under the receiver operating characteristic)

- AUROC is a performance metric to evaluate classification models. For binary classification problems, AUROC measures the ability of a model to discriminate between positive cases and negative cases.
- AUROC range: 0 ~ 1, where a higher AUROC value indicates better overall discrimination performance.
  - 0.5 represents a random classifier, the worst case.
  - < 0.7 is sub-optimal performance.
  - 0.7 ~ 0.8 is good performance.
  - < 0.8 indicates excellent performance.
  - 1.0 represents a perfect classifier, the best case.



AUPRC (area under the precision-recall curve)

- AUPRC is another performance metric for binary classification that combines precision (positive predictive value) and recall into a single score. AUPRC is useful for imbalanced datasets where you care more about positive classes. It emphasizes the model's ability to correctly identify positive examples while minimizing false positives.
- AUPRC range: 0 ~ 1, where a higher AUPRC value indicates better performance. Different classes have different AUPRC baselines (unlike AUROC whose baseline is 0.5).

The team ran the bash script to evalute the baseline model. The results produced offers insights into which data types are most predictive for mortality in this context and how each model handles different combinations of data. The highest values among the F1-score, AUC, and AUPRC indicate the best performing model under the given configuration.

Task Mortality

```
# mortality:  U
!python baselines.py
# mortality: T+S
!python baselines.py --inputs 3
# mortality: U+T+S
!python baselines.py --inputs 7
```

Outputs:

Mortality: U

```
Running task mortality using inputs 4...
Loading notes...
Done.
Start training Logistic Regression:
Running time: 5.936492204666138
(0.3704396632366698, 0.8194338825517532, 0.351511879169403)
Start training Random Forest:
Running time: 1594.4834170341492
(0.31451899183528575, 0.7538361422137547, 0.2517496407128295)
```

Mortality: T+S

```
Running task mortality using inputs 3...
Loading temporal data...
Loading demographics...
Done.
Start training Logistic Regression:
Running time: 10.263913869857788
(0.3265426631237083, 0.7728912317902655, 0.3202270307520234)
Start training Random Forest:
Running time: 697.0606007575989
(0.32880755608028334, 0.7795339671365924, 0.33893859409516924)
```

Mortality: U+T+S

```
Running task mortality using inputs 7...
```

```
Loading notes...
Loading temporal data...
Loading demographics...
Done.
Start training Logistic Regression:
Running time: 14.972199201583862
(0.3714285714285714, 0.8362021317532203, 0.39873421821281746)
Start training Random Forest:
Running time: 1849.498405456543
(0.3429937257245294, 0.8060038193239133, 0.36047925501163297)
```

In the context for Mortality, the Logistic Regression and Random Forest models exhibit different performances across the various input types.

For Logistic Regression:

- The highest AUC is with U+T+S inputs.
- The highest F1-score is also with U+T+S inputs.
- The AUPRC is highest with U+T+S inputs.

For Random Forest:

- The highest AUC is again with U+T+S inputs.
- The F1-score is highest with U+T+S inputs.
- The AUPRC is highest with U+T+S inputs.

In this scenario, **the combination of unstructured, temporal, and static data (U+T+S) yields the best performance for both Logistic Regression and Random Forest models** in terms of all three metrics: F1-score, AUC, and AUPRC. This suggests that a more comprehensive data input leads to improved predictive capabilities for mortality prediction.

Task Readmit

```
# readmit:  U
!python baselines.py --task readmit
# readmit: T+S
!python baselines.py --task readmit --inputs 3
# readmit: U+T+S
!python baselines.py --task readmit --inputs 7
```

Readmit: U

```
Running task readmit using inputs 4...
Loading notes...
Done.
Start training Logistic Regression:
Running time: 1.6387195587158203
(0.15126528291157235, 0.6432312651087833, 0.10359691905165995)
Start training Random Forest:
Running time: 1628.2646622657776
(0.12429111531190926, 0.5934241691695358, 0.07607032807726372)
```

Readmit: T+S

```
Running task readmit using inputs 3...
Loading temporal data...
Loading demographics...
Done.
Start training Logistic Regression:
Running time: 2.254403829574585
(0.13841201716738197, 0.6174883402925306, 0.08623456415049702)
Start training Random Forest:
Running time: 518.0668940544128
(0.12080536912751676, 0.5694347788928773, 0.07809514683851808)
```

Readmit: U+T+S

```
Running task readmit using inputs 7...
Loading notes...
Loading temporal data...
Loading demographics...
Done.
Start training Logistic Regression:
Running time: 4.281747102737427
(0.1555292726197516, 0.6526686079163919, 0.1043344450312436)
Start training Random Forest:
Running time: 1787.5334012508392
(0.1343115124153499, 0.6079237174321783, 0.09930980010022672)
```

Looking at these results, the following observations can be made for the readmission prediction task:

- Logistic Regression generally performs better than Random Forest in terms of AUC across all input types.
- The combination of unstructured, temporal, and static data (U+T+S) seems to give the best results for Logistic Regression.
- For the Random Forest model, the F1 score and AUC are also highest when combining all types of data (U+T+S), though the improvement in AUPRC is marginal compared to unstructured data alone.

The **U+T+S combination provides the best metrics**, indicating that integrating all types of data can potentially improve the model's predictive performance for readmission.

Task Long Length Of Stay

```
# llos:  U
!python baselines.py --task llos
# llos: T+S
!python baselines.py --task llos --inputs 3
# llos: U+T+S
!python baselines.py --task llos --inputs 7
```

Los_bin: U

```
Running task llos using inputs 4...
Loading notes...
Done.
Start training Logistic Regression:
Running time: 2.032042980194092
(0.6756174794173526, 0.7305319366871751, 0.7042684601983227)
Start training Random Forest:
Running time: 1042.985583782196
(0.6656093692814445, 0.7009457624164281, 0.6749054867878506)
```

Los_bin: T+S

```
Running task llos using inputs 3...
Loading temporal data...
Loading demographics...
Done.
Start training Logistic Regression:
Running time: 2.261179208755493
(0.6603386351975371, 0.7240891231854905, 0.7162329942831149)
Start training Random Forest:
Running time: 416.05641627311707
(0.6653465346534654, 0.736769408342121, 0.734146670966267)
```

Los_bin: U+T+S

```
Running task llos using inputs 7...
Loading notes...
Loading temporal data...
Loading demographics...
Done.
Start training Logistic Regression:
Running time: 4.478009223937988
(0.6916645553585002, 0.750640901330107, 0.7337056359283238)
Start training Random Forest:
Running time: 1085.791122674942
(0.6816331889195222, 0.7586284901279937, 0.7372561813045976)
```

LOS: U (Unstructured Data)

- Logistic Regression: Appears to have the highest AUC among the LR models.
- Random Forest: Comparable AUC to the LR model but with a different F1-score and AUPRC.

LOS: T+S (Temporal and Static Data)

- Logistic Regression: Lower AUC than the U model.
- Random Forest: The metrics are not as high as in the U data set.

LOS: U+T+S (Unstructured, Temporal, and Static Data Combined)

- Logistic Regression: Lower AUC than the U model.
- Random Forest: Slightly lower metrics than the U model.

It seems that for **Logistic Regression, unstructured data alone (U) provided the highest AUC**. For **Random Forest, while the differences are not dramatic, the unstructured data alone still seems to edge out slightly over the combined data inputs** in terms of AUC and AUPRC.

The performance of a model on a task like Length of Stay (los_bin) prediction depends on how well the features represent the underlying patterns associated with longer or shorter stays. Unstructured data often contains rich, detailed narratives that might include subtle indicators of patient health not captured in structured data. It's possible that the model can pick up on these nuances more effectively, leading to better performance when only unstructured data is used.

On the other hand, combining unstructured data with temporal and static data might introduce noise or irrelevant information that could slightly confuse the model, making it harder to find the truly predictive signals within the unstructured data. This might explain why the models trained on unstructured data alone performed better for los_bin prediction. It also underscores the importance of feature selection and the possibility that more data isn't always better for predictive accuracy.

Baseline Analysis

Task	Inputs	Training Model	Running Time	Output
Mortality: U	4	Logistic Regression	5.93649	(0.3704396, 0.819433, 0.351518)
Mortality: U	4	Random Forest	1594.48431	(0.31451899, 0.7538364, 0.2517496)
Mortality: T+S	3	Logistic Regression	10.26391	(0.32654266, 0.7728912, 0.3220278)
Mortality: T+S	3	Random Forest	697.0606007	(0.3288075, 0.7795339, 0.3389859)
Mortality: U+T+S	7	Logistic Regression	14.972199	(0.37142857, 0.8362021, 0.3987342)
Mortality: U+T+S	7	Random Forest	1849.496845	(0.3429937, 0.8060038, 0.3604792)

Task	Inputs	Training Model	Running Time	Output
Readmit: U	4	Logistic Regression	1.6387915	(0.15162528, 0.6432216, 0.1035969)
Readmit: U	4	Random Forest	1628.26466	(0.12429111, 0.5934421, 0.0760703)
Readmit: T+S	3	Logistic Regression	2.2544038	(0.1384121, 0.6174884, 0.0862345)
Readmit: T+S	3	Random Forest	518.0668940	(0.12080536, 0.5694347, 0.0780951)
Readmit: U+T+S	7	Logistic Regression	4.287147	(0.15559227, 0.6526668, 0.1043344)
Readmit: U+T+S	7	Random Forest	1787.5334025	(0.1343115, 0.6079237, 0.0993989)

Task	Inputs	Training Model	Running Time	Output
LOS_bin: U	4	Logistic Regression	2.02324	(0.6756741, 0.7305193, 0.7042485)
LOS_bin: U	4	Random Forest	1042.98558	(0.6656093, 0.7009457, 0.6749054)
LOS_bin: T+S	3	Logistic Regression	2.26178	(0.6603386, 0.7240891, 0.7162329
LOS_bin: T+S	3	Random Forest	416.05462	(0.6653465, 0.7367694, 0.7341466)
LOS_bin: U+T+S	7	Logistic Regression	4.478009	(0.6916645, 0.7506409, 0.7337056)
LOS_bin: U+T+S	7	Random Forest	1085.79112	(0.6816331, 0.7586284, 0.7372561)

Our first overarching goal for **Hypothesis 1** was to determine if Fusion-CNN and Fusion-LSTM models could outperform traditional machine learning models (Logistic Regression and Random Forest) on three clinical prediction tasks: F1 score, AUC, and AUPRC.

Our findings indicated:

- Mortality prediction showed the highest performance metrics when using the combination of unstructured, temporal, and static data (U+T+S) for both Logistic Regression and Random Forest.
- Readmission prediction also had higher performance metrics with U+T+S inputs for both models.
- Length of Stay prediction demonstrated that models trained on unstructured data alone (U) had slightly better or comparable performance metrics.

In each case, Logistic Regression and Random Forest models exhibit variations in their performance based on the data inputs, with no single model consistently outperforming the other across all tasks and metrics. To conclude which model is better overall, we would need to analyze the specific metric priorities for each task and compare them to the performance of the Fusion-CNN and Fusion-LSTM models. If Fusion-CNN and Fusion-LSTM models can surpass these performance metrics, it would support the hypothesis that they outperform traditional machine learning models on these clinical prediction tasks.

Deep Models Evaluation

We will now execute the scripts below to train and evaluate the deep learning models:

- Fusion-CNN
- Fusion-LSTM

These models utilize advanced neural network architectures to learn from both structured and unstructured data, aiming to improve prediction accuracy by capturing complex patterns in the data. Both models are implemented using PyTorch.

```
!python main.py --model [model] --task [task] --use_unstructure [input] # train F
!python main.py --model [model] --task [task] --use_unstructure [input] --phase t
```

Training for Fusion-CNN model:

```
# use_unstructure=0: temporal (T) + structured (S)

# use_unstructure=1: temporal (T) + structured (S) + unstructured (U)
!python main.py --model cnn --epochs 10 --task mortality --use_unstructure 0 --ph
!python main.py --model cnn --epochs 10 --task mortality --use_unstructure 1 --ph

!python main.py --model cnn --epochs 10 --task llos --use_unstructure 0 --phase t
!python main.py --model cnn --epochs 10 --task llos --use_unstructure 1 --phase t

!python main.py --model cnn --epochs 10 --task readmit --use_unstructure 0 --phas
!python main.py --model cnn --epochs 10 --task readmit --use_unstructure 1 --phas
```

Training for Fusion-LSTM model:

```
# use_unstructure=0: temporal (T) + structured (S)

# use_unstructure=1: temporal (T) + structured (S) + unstructured (U)
!python main.py --model lstm --epochs 10 --task mortality --use_unstructure 0 --p
!python main.py --model lstm --epochs 10 --task mortality --use_unstructure 1 --p

!python main.py --model lstm --epochs 10 --task llos --use_unstructure 0 --phase
!python main.py --model lstm --epochs 10 --task llos --use_unstructure 1 --phase

!python main.py --model lstm --epochs 10 --task readmit --use_unstructure 0 --pha
!python main.py --model lstm --epochs 10 --task readmit --use_unstructure 1 --pha
```

CNN model evaluation for *length of stay (los_bin), mortality, and readmit* using **temporal (T) + structured (S) + unstructured (U)** as input data.

```
!python main_eval.py --model cnn --task llos --use_unstructure 1 --phase test --m
```

```
test
  0% 0/124 [00:00<?, ?it/s]/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeW
  self.pid = os.fork()
100% 124/124 [16:55<00:00,  8.19s/it]
1 (0.4862935101364274, 0.48101387094626735, 0.49116222985861185)
Avg [0.4862935101364274, 0.48101387094626735, 0.49116222985861185]

test Epoch 0 (lr 0.000100)
loss: 0.6936
```

```
!python main_eval.py --model cnn --task mortality --use_unstructure 1 --phase tes
```

```
test
  0% 0/124 [00:00<?, ?it/s]/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeW
  self.pid = os.fork()
100% 124/124 [00:33<00:00,  3.72it/s]
1 (0.14223107569721113, 0.4460207901103011, 0.08789246676541324)
Avg [0.14223107569721113, 0.4460207901103011, 0.08789246676541324]

test Epoch 0 (lr 0.000100)
loss: 0.6955
```

```
!python main_eval.py --model cnn --task readmit --use_unstructure 1 --phase test
```

```
test
  0% 0/124 [00:00<?, ?it/s]/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeW
  self.pid = os.fork()
100% 124/124 [00:33<00:00,  3.65it/s]
1 (0.10887974836680378, 0.5249684089077723, 0.06560159980317876)
Avg [0.10887974836680378, 0.5249684089077723, 0.06560159980317876]

test Epoch 0 (lr 0.000100)
loss: 0.6790
```

LSTM model evaluation for *length of stay (los_bin), mortality, and readmit* using **temporal (T) + structured (S) + unstructured (U)** as input data.

```
!python main_eval.py --model lstm --task llos --use_unstructure 1 --phase test --
```

```
test
  0% 0/124 [00:00<?, ?it/s]/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeW
  self.pid = os.fork()
100% 124/124 [00:41<00:00,  3.01it/s]
1 (0.5040920716112531, 0.5094735433834098, 0.5083240528744528)
Avg [0.5040920716112531, 0.5094735433834098, 0.5083240528744528]

test Epoch 0 (lr 0.000100)
loss: 0.6956
```

```
!python main_eval.py --model lstm --task mortality --use_unstructure 1 --phase te
```

```
test
  0% 0/124 [00:00<?, ?it/s]/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeW
  self.pid = os.fork()
100% 124/124 [00:41<00:00,  2.96it/s]
1 (0.16039891959276958, 0.46776797382982754, 0.08888998537965924)
Avg [0.16039891959276958, 0.46776797382982754, 0.08888998537965924]
```

```
!python main_eval.py --model lstm --task readmit --use_unstructure 1 --phase test
```

```
test
  0% 0/124 [00:00<?, ?it/s]/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeW
  self.pid = os.fork()
100% 124/124 [00:40<00:00,  3.03it/s]
1 (0.10575573233504913, 0.49376327741557396, 0.05319525265878469)
Avg [0.10575573233504913, 0.49376327741557396, 0.05319525265878469]

test Epoch 0 (lr 0.000100)
loss: 0.6805
```

CNN model evaluation for *length of stay (los_bin), mortality, and readmit* using **temporal (T) + structured (S)** as input data.

```
!python main_eval.py --model cnn --task llos --use_unstructure 0 --phase test --m
```

```
test
  0% 0/124 [00:00<?, ?it/s]/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeW
  self.pid = os.fork()
100% 124/124 [53:36<00:00, 25.94s/it]
1 (0.42039935240151105, 0.4321146286689378, 0.45786858777398387)
Avg [0.42039935240151105, 0.4321146286689378, 0.45786858777398387]

test Epoch 0 (lr 0.000100)
loss: 0.6944
```

```
!python main_eval.py --model cnn --task mortality --use_unstructure 0 --phase tes
```

```
test
  0% 0/124 [00:00<?, ?it/s]/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeW
  self.pid = os.fork()
100% 124/124 [00:37<00:00,  3.27it/s]
1 (0.15944609297725026, 0.4759799687314158, 0.09685476554069887)
Avg [0.15944609297725026, 0.4759799687314158, 0.09685476554069887]

test Epoch 0 (lr 0.000100)
loss: 0.6966
```

```
!python main_eval.py --model cnn --task readmit --use_unstructure 0 --phase test
```

```
 test
   0% 0/124 [00:00<?, ?it/s]/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeW
   self.pid = os.fork()
 100% 124/124 [00:37<00:00,  3.33it/s]
 1 (0.1203852327447833, 0.5673918882621541, 0.07274826334883697)
 Avg [0.1203852327447833, 0.5673918882621541, 0.07274826334883697]

 test Epoch 0 (lr 0.000100)
 loss: 0.6781
```

LSTM model evaluation for *length of stay (los_bin), mortality, and readmit* using **temporal (T) + structured (S)** as input data.

```
!python main_eval.py --model lstm --task llos --use_unstructure 0 --phase test --
```

```
 test
   0% 0/124 [00:00<?, ?it/s]/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeW
   self.pid = os.fork()
 100% 124/124 [00:39<00:00,  3.18it/s]
 1 (0.48167673910248165, 0.4849326038150168, 0.4928208491938085)
 Avg [0.48167673910248165, 0.4849326038150168, 0.4928208491938085]

 test Epoch 0 (lr 0.000100)
 loss: 0.6938
```

```
!python main_eval.py --model lstm --task mortality --use_unstructure 0 --phase te
```

```
 test
   0% 0/124 [00:00<?, ?it/s]/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeW
   self.pid = os.fork()
 100% 124/124 [00:39<00:00,  3.11it/s]
 1 (0.1475119979792877, 0.45927935152221316, 0.08868583375865752)
 Avg [0.1475119979792877, 0.45927935152221316, 0.08868583375865752]
```

```
!python main_eval.py --model lstm --task readmit --use_unstructure 0 --phase test
```

```
test
  0% 0/124 [00:00<?, ?it/s]/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeW
  self.pid = os.fork()
100% 124/124 [00:39<00:00,  3.16it/s]
1 (0.09274106175514626, 0.43490006837105954, 0.04665630858931094)
Avg [0.09274106175514626, 0.43490006837105954, 0.04665630858931094]

test Epoch 0 (lr 0.000100)
loss: 0.6806
```

Deep Model Analysis

The results table highlights performance metrics for the Fusion-CNN and Fusion-LSTM models across three tasks (Mortality, Readmission, and Length of Stay) on MIMIC-III. These results compare models trained with combinations of inputs:

- Temporal + Static (T+S)
- Temporal + Static + Unstructured (T+S+U)

Our current evaluation does not cover the analysis of unstructured data, which is essential for fully appreciating the capabilities of deep learning models in clinical environments. Unstructured data, including free-text clinical notes, typically holds detailed information that can greatly improve prediction accuracy. Our team faced challenges in assembling performance metrics due to errors in processing this unstructured data. We aim to utilize analyses that combine structured and unstructured data (T+S and T+S+U) to indirectly assess the unstructured data (U).

The data shows that the performance generally **improves or stays consistent when moving from using temporal and static data (T+S) to including unstructured data (T+S+U)**. Specifically:

- Mortality Prediction: There is a slight improvement in AUROC and F1 score when unstructured data is included, though AUPRC decreases somewhat.
- Readmission Prediction: Incorporating unstructured data leads to a slight decrease in all performance metrics. This might indicate that the unstructured data introduces noise or complexity that the models struggle to beneficially integrate for this specific task.
- Length of Stay Prediction: Shows improvement across all metrics when unstructured data

is added, suggesting that this type of data may contain valuable additional signals for predicting LOS.

For the deep models (Fusion cnn and Fusion lstm), across all tasks, the inclusion of unstructured data alongside temporal and static features consistently enhances predictive accuracy especially for fusion lstm model across all tasks. This trend is less obvious for fusion cnn, which only shows improvement in performance for task length of stay. According to the paper, we are expecting to observe the same trend for fusion cnn and fusion lstm across all tasks. Our result differences could be due to smaller epochs used to train the models compared to the paper. Furthermore, the choice of model architecture plays a significant role in performance, with Fusion LSTM generally exhibiting superior predictive capabilities compared to Fusion CNN, particularly when considering the richer feature set enabled by incorporating unstructured data.

Results

Result Tables

- Table 1 shows in-hospital mortality prediction on MIMIC-III. U, T, S represents unstructured data, temporal signals, and static information respectively.

  Baseline models: Logistic Regression, Random Forest.

Task	Inputs	Running Time	F1	AUROC	AUPRC
Logistic Regression	T + S	10.26391	0.32654266	0.7728912	0.3220278
Logistic Regression	U	5.93649	0.3704396	0.819433	0.351518
Logistic Regression	T + S + U	14.972199	0.37142857	0.8362021	0.3987342
Random Forest	T + S	697.0606007	0.3288075	0.7795339	0.3389859
Random Forest	U	1594.48431	0.31451899	0.7538364	0.2517496
Random Forest	T + S + U	1849.496845	0.3429937	0.8060038	0.3604792

- Table 2 shows long length of stay prediction on MIMIC-III. U, T, S represents unstructured data, temporal signals, and static information respectively.

  Baseline models: Logistic Regression, Random Forest.

Task	Inputs	Running Time	F1	AUROC	AUPRC
Logistic Regression	T + S	2.26178	0.6603386	0.7305193	0.7042485
Logistic Regression	U	2.02324	0.6756741	0.819433	0.351518
Logistic Regression	T + S + U	4.478009	0.6916645	0.7506409	0.7337056
Random Forest	T + S	[illegible]	[illegible]	[illegible]	[illegible]

Random Forest	T + S	416.05462	0.6653465	0.7367694	0.7341466
Random Forest	U	1042.98558	0.6656093	0.7009457	0.6749054
Random Forest	T + S + U	1085.79112	0.6816331	0.7586284	0.7372561

- Table 3 shows 30-day readmission prediction on MIMIC-III. U, T, S represents unstructured data, temporal signals, and static information respectively.

  Baseline models: Logistic Regression, Random Forest.

Task	Inputs	Running Time	F1	AUROC	AUPRC
Logistic Regression	T + S	2.2544038	0.1384121	0.6174884	0.0862345
Logistic Regression	U	1.6387915	0.15162528	0.6432216	0.1035969
Logistic Regression	T + S + U	4.287147	0.15559227	0.6526668	0.1043344
Random Forest	T + S	518.0668940	0.12080536	0.5694347	0.0780951
Random Forest	U	1628.26466	0.12429111	0.5934421	0.0760703
Random Forest	T + S + U	1787.5334025	0.1343115	0.6079237	0.0993989

- Table 4 shows Long length of stay, mortality, and readmit predictions on MIMIC-III. U, T, S represents unstructured data, temporal signals, and static information respectively.

  Fusion-CNN and Fusion-LSTM Models

Task	Inputs	F1	AUROC	AUPRC
Mortality	T+S	Fusion cnn	0.15944609297725026	0.4759799687314158
Mortality	T+S+U	Fusion cnn	0.14223107569721113	0.4460207901103011
Mortality	T+S	Fusion lstm	0.15944609297725026	0.4759799687314158
Mortality	T+S+U	Fusion lstm	0.16039891959276958	0.46776797382982754

Task	Inputs	F1	AUROC	AUPRC
Readmit	T+S	Fusion cnn	0.1203852327447833	0.5673918882621541
Readmit	T+S+U	Fusion cnn	0.10887974836680378	0.5249684089077723
Readmit	T+S	Fusion lstm	0.09274106175514626	0.43490006837105954
Readmit	T+S+U	Fusion lstm	0.10575573233504913	0.49376327741557396

Task	Inputs	F1	AUROC	AUPRC
LOS_Bin	T+S	Fusion cnn	0.42039935240151105	0.4321146286689378
LOS_Bin	T+S+U	Fusion cnn	0.4862935101364274	0.48101387094626735
LOS_Bin	T+S	Fusion lstm	0.48167673910248165	0.4849326038150168
LOS_Bin	T+S+U	Fusion lstm	0.5040920716112531	0.5094735433834098

Please note: Table is missing evaluation for Unstructured (U) data.

The evaluation of both traditional machine learning models (Logistic Regression and Random Forest) and advanced deep learning models (Fusion-CNN and Fusion-LSTM) across various clinical prediction tasks provides significant insights. For Hypothesis 1, which posits that Fusion-CNN and Fusion-LSTM outperform traditional machine learning models, the analysis shows nuanced results. Although deep learning models sometimes exhibit lower AUROC scores compared to baseline models, they consistently achieve higher F1 scores and AUPRC. This indicates that deep models may not always rank positive cases highest but are more effective at correctly identifying and classifying cases, which is crucial in clinical settings where accuracy in positive case identification can directly impact patient outcomes. For example, in mortality prediction, deep models display superior precision and recall, as evidenced by their high F1 and AUPRC scores compared to traditional models. This pattern of results was although not anticipated based on the literature and our findings differ slightly, however, this is possibly due to the fewer epochs used in our model training compared to what was described in the original paper.

In regards to Hypothesis 2, which suggests that combining unstructured text and structured data improves model performance over using either data type alone, the findings are supportive. The deep learning models show improved or maintained performance when unstructured data is added to structured data, particularly evident in tasks such as Length of Stay and Mortality prediction. This enhancement in performance metrics suggests that the integration of data types provides a richer, more comprehensive dataset for models to learn from, enhancing their predictive accuracy.

Overall, the evidence supports that advanced machine learning frameworks, specifically Fusion-CNN and Fusion-LSTM, provide significant advantages over traditional logistic regression and random forest models, especially when handling complex data integrations. The incorporation of unstructured data into these models not only supports improved performance but also confirms the potential of deep learning in effectively utilizing diverse data types for clinical predictions. Future efforts should focus on optimizing these models to harness the full potential of unstructured data, ensuring they are finely tuned to meet the intricate demands of healthcare data analytics.

Issues Encountered

In our attempt to process data using the clinical-fusion and MIMIC-III buildmimic/postgres GitHub repositories, we encountered several challenges that hindered the successful execution of the project.

Environment Setup Issues: While the team ensured all necessary software and libraries were installed, which included the correct version of PostgreSQL and Python libraries. Compatibility issues among these dependencies were a significant hurdle due to different machines or required depdendencies for the project dependences causing delays and long troubleshooting hours.

- Example: For some members using a different version of Python generated compatibility issues with the version of the PostgreSQL for this project. This also led to updating required libraries to the correct version (downgrading for some cases as well).

Poorly Documented Instructions: Although this can vary across different machines and levels of experience working with this type of model, the Github for the clinical-fusion provided an overly simplified instructions that requires intrepretation on building the dataset. The repo instructed users to process the data leveraging several python files; in actuality, users have to PROCESS the data first by following a separate repo. Although the instructions did mention under Step 2 to "build the MIMIC-III dataset locally using Postgres," the repo for bulding the dataset was unclear and was up to the team to decipher what the next steps should be.

- Example: Repo should list major dependencies (Make) especially if the dependencies are required for building the dataset.

Script-Specific Challenges: The team encountered errors in the SQL scripts, including syntax errors and incorrect data type assumptions, which led to faulty executions. The Python scripts had their own set of issues, including compatibility problems with the version of Python the team was using and other logical errors that prevented proper execution.

Data Size and Resource Limitations:

- Hardware Limitations: The large size of the MIMIC-III dataset posed a challenge due to our limited hardware capabilities, particularly with memory and disk space (it was estimated to use up to 100 GB for this project at minimum).
- Timeouts and Overloads: Some scripts ran for an extended period, causing timeouts or system overloads, which made it difficult to proceed without interruptions (i.e., *06_doc2vec.py*)

Documentation and Misconfiguration: At times, the documentation did not seem up-to-date, which led to some confusion and incorrect configuration. There were instances where members misfollowed setup instructions or data preparation steps, contributing to the issues encountered which led to more lost of time to complete this draft.

Trainset data missing: The data defined by the authors for doc2vec model training is missing from the input (i.e., *earlynotes.csv*) file.

Important File Modifications: Several files required modifications to remediate errors encountered during data processing.

- **06_doc2Vec.py**: During data processing, the team converted the data column `hadm_id` to data type **int** to match the `train_id` data type; this allowed the following [-10,-4] to be the correct file ID that is required for processing (updated line: `(train_ids = list(map(lambda x: safe_convert_to_int(x[-10:-4]), train_ids)))`. The code was also updated to remove EMPTY or NONE values in the `train_ids` column.
- **baseline.py**: Columns `trains_ids and test_ids` were converted to data type **int**.
- **lstm.py**: File path correction required.
- **cnn.py**: File path correction required.
- **Other modifications due to compatibility or data issues:**
    - **01_get_signals.py**
    - **02_extract_notes.py**
    - **04_statistics.py**
    - **06_doc2vec.py**
    - **main.py**
    - **parse.py**

Discussions

Our study significantly advances the exploration of multimodal deep learning architectures, specifically Fusion-CNN and Fusion-LSTM, in healthcare predictive modeling. We utilized combinations of structured and unstructured data to predict patient outcomes. The findings suggest that incorporating unstructured data generally enhances or maintains performance levels compared to using only temporal and static data. Notably, the Fusion-LSTM model consistently demonstrated superior performance across most metrics and tasks, highlighting its robustness in integrating and processing complex data types.

The replication encountered substantial hurdles predominantly in the technical sphere, specifically the setup of the environment and the management of software dependencies. A major hurdle was compatibility issues across different versions of PostgreSQL and Python, resulting in significant conflicts that the documentation did not foresee. Windows-specific challenges, particularly the execution of make commands, highlighted the assumption of a Unix-like environment by the original authors. Additionally, the Windows-based issues were further compounded by a lack of detailed setup instructions for this operating system, leading to extended debugging sessions.The simplicity of accessing materials and understanding the procedures was counterbalanced by the complexity of executing the setup process. This complexity was exemplified in the handling of environment configuration, software dependencies, and particularly the execution of database setup tasks in Windows—tasks that proved to be nontrivial.

In essence, the study aimed to replicate and validate previous findings; however, certain deviations were observed. These variations could be attributed to differences in computational environments or slight alterations in model configuration such as smaller epoch used due to machine limits. Our results underscore the significant impact that factors such as training durations and model architecture have on performance, pointing out that even minor changes can significantly influence outcomes.

With regards to handling unstructured data, particularly free-text clinical notes, posed significant challenges. The preprocessing and integration of this data type introduced complexity and noise, which was particularly noticeable in the readmission prediction task where performance slightly declined. Conversely, managing to maintain or improve performance with the addition of unstructured data in most tasks was a significant achievement, demonstrating the models' capabilities in handling complex datasets.

To enhance the reproducibility of such studies, we recommend providing extensive documentation on all aspects of the model architecture, data preprocessing, and training procedures. Standardizing computational environments can minimize variations in performance due to environmental differences. Additionally, sharing detailed settings such as epochs and batch sizes, along with the rationale for their selection, will align more closely with practices described in the literature.

In conclusion, our research confirms the potential of deep learning models to effectively leverage multimodal data. By addressing the challenges noted and adhering to our recommendations, future efforts will not only assess the reproducibility of the paper but also serve as a case study for refining reproducibility practices in computational research. This contribution is hoped to facilitate a more seamless experience for subsequent researchers aiming to validate and build upon the published work.

References

1. Zhang, Dongdong, et al. "Combining structured and unstructured data for predictive models: a deep learning approach." BMC medical informatics and decision making 20 (2020): 1-11.
2. F1 Score vs ROC AUC vs Accuracy vs PR AUC: Which Evaluation Metric Should Your Choose? neptune.ai/blog/f1-score-accuracy-roc-auc-pr-auc
3. F1 Score in Machine Learning: Intro & Calculation v7labs.com/blog/f1-score-guide
4. Measuring Performance: AUC (AUROC) glassboxmedicine.com/2019/02/23/measuring-performance-auc-auroc
5. Understanding AUC-ROC Curve towardsdatascience.com
6. Measuring Performance: AUPRC and Average Precision glassboxmedicine.com
7. The wrong and right way to approximate Area Under Precision-Recall Curve towardsdatascience.com