Code.Hub

Data Science Bootcamp

powered by Pfizer

IntensiLedge

Medical Information Mart for Intensive Care (MIMIC-III) dataset

September 2020

Abstract:

This document describes the project assigned to the data science bootcamp participants in order to demonstrate the skills and knowledge they have gained throughout the training. A dataset is given concerning information about nearly 50.000 intensive care patients from real life cases. The data will be used to processed and analyzed according to the tasks given to you. An API will be implemented to provide a machine friendly access and management of the data. Also, a web-based user-interface will be built that will allow a user to interact and perform various actions. Finally, a presentation will be given with all the work, decisions, implementations and results that have taken place.

Table of Contents

1 Introduction

You are given a processed version of the Medical Information Mart for Intensive Care (MIMIC-III) dataset that contains information relating to patients admitted to critical care units at a large tertiary care hospital. A detailed description on the data collection protocol and the original dataset can be found here.

The original MIMIC dataset consists of 28 tables with millions of entries. One very popular analysis performed on this dataset is to try to correlate the patients' interactions with the with the hospital with the duration of their hospitalization. For these purposes, we processed the dataset and provide you with an aggregated version of the dataset consisting only of the number of interactions.

The dataset you are given contains nearly 50.000 hospital admissions, including the duration of their hospitalization (which you will later need to predict). You will also be provided with a brief description of the features.

2 Exploratory Data Analysis

Perform an initial Exploratory Data Analysis on the given dataset. The goal of this task is to familiarize you with the dataset. In this step you should:

- Understand what each table/column represents.
- Recognize if there are missing/wrong values in the data.
- Check for outliers in the data.
- Obtain insights through descriptive statistics.
- Identify key relationships within the data.

Hint: visualizations are key for this step!

3 Initial preprocessing

The goal of this task is to bring your data in the proper form to be further analyzed in the next steps. In this step you should:

- Clean the data (handling missing values, fixing errors, outliers, etc.)
- Encode the variables that require you to.
- Perform any feature engineering step you see fit (group features, define KPIs, etc.)

4 Modelling

The goal of this task is to predict the duration that a patient is hospitalized. The targets of this classification are:

- ➔ Day (< 1 day)
- ➔ Week (>= 1 day and < 7 days)
- ➔ TwoWeeks (>= 7 days and < 14 days)
- ➔ Month (>= 14 days and < 30 days)
- ➔ More (>= 30 days)

Note: you will need to generate these labels on your own!

In this task you should:

- Build a model that correctly predicts how long a patient will be hospitalized, according to the labels above.
- You can use any technique you want (heuristic, statistical, machine learning. etc.)

5 Rest API - WIP

Create an API that provides access to the MIMIC-III data.
The goal of this task is to expose the processed data and the created models from the two previous tasks to the world via a RESTful API.

- Create a data store (file, database) for your data both in steps 3 and 4
- Develop an application which will provide two endpoints:
  - /stats: Provide meaningful aggregations based on the data set.

    Examples of meaningful statistics:
    - Plot a histogram of the values of a column (e.g. age)
    - Plot a pie chart of the values of a column (e.g. ethnicity)
    - Plot a bar chart of the mean hospitalization per admission_type)

    Hint 1: these above are just examples. You should plot the variables and relationships that you have found to be ***meaningful*** *during your EDA.*
    Hint 2: some these plots need to be generated from the ***processed*** *data! There is no point to plot categories containing typos, mistakes, etc.*
    Hint 3: these figures are meant to be viewed by a human. Treat them as such. E.g. the categories in a pie chart should be understandable by a human; an encoded/scaled variable loses some of its interpretability.
  - /models: This endpoint should accept the data for an individual patient, in the format of the original file we

provided to you and return the predicted duration of hospitalization for that patient. To do this you should:

i. Perform a check on the validity of the data provided. Have you provided the right amount of features for the model to make a prediction? Does each feature have a valid value?

ii. Bring the data in a format understandable by the model. To do this you must process the data in the same way you did with the original dataset (cleaning, encoding, scaling).

iii. Pass the processed sample to the model and return its prediction to the user.

- Log in a file incoming requests and system behavior you find relevant.

Hints:

- Response from /stats endpoint should comply with how you want to draw your charts in the web application. Your front end should not do any processing on the data.
- Sample /stats response based on examples:

```
{
  "stats": {
    "histogram_data": [
      {
        "age": "30",
        "count": 2
      },
      {
        "age": "35",
        "count": 143
      },
      {
        "age": "40",
        "count": 28
      }
    ],
    "pie_chart_data": [
      {
        "ethnicity": "WHITE",
        "count": 150
      },
      {
        "ethnicity": "HISPANIC OR LATINO",
        "count": 150
      },
      {
        "ethnicity": "BLACK/AFRICAN AMERICAN",
        "count": 50
```

```
      }
    ],
    "bar_char_data": [
      {
        "admission_type": "EMERGENCY",
        "mean_hospitalization": 12.3
      },
      {
        "admission_type": "ELECTIVE",
        "mean_hospitalization": 2.1
      }
    ]
  }
}
```


- Sample /model request and response:

  RQ:


```
{
  "gender": "F",
  "age": 34,
  "los_days": 2,
  "admit_type": "EMERGENCY",
  "admit_location": "CLINIC REFERRAL/PREMATURE",
  "admit_diagnosis": "DIABETIC KETOACIDOSIS",
  "insurance": "Private",
  "religion": "CATHOLIC",
  "marital_status": "DIVORCED",
  "ethnicity": "WHITE",
  "num_callouts": 0.16,
  "num_diagnosis": 2.34,
  "num_procs": 0,
  "admit_procedure": "Endosc control gast hem",
  "num_ctp_events": 0.83,
  "num_input": 13.31,
  "num_labs": 22.93,
  "num_microlabs": 0.62,
  "num_notes": 0.05,
  "num_output": 5.13,
  "num_rx": 7.19,
  "num_proc_events": 0,
  "num_transfers": 0.33,
  "num_chart_events": 1212.2,
  "expired_hospital": 0,
  "toal_num_interact": 845.36,
  "los_group_num": 3
}
```


  RS:


```
{
  "patient_info": {
    "gender": "F",
    "age": 34,
```

```
    "admit_type": "EMERGENCY",
    "admit_location": "CLINIC REFERRAL/PREMATURE",
    "admit_diagnosis": "DIABETIC KETOACIDOSIS",
    "insurance": "Private",
    "religion": "CATHOLIC",
    "marital_status": "DIVORCED",
    "ethnicity": "WHITE"
  },
  "prediction": {
    "hospitalization": "Month "
  }
}
```

Again, those are samples, and you should modify them according to your modeling. (e.g request only the fields that you use for prediction)

6 Web APP

The goal of this task is to consume the RESTful API - developed in the previous step - and present it nicely in a web application. The web app will be responsive and viewable from any device (and screen size) with a modern browser. From a more technical view, the web app will be a SPA (single page application) based on the React library and its extensive ecosystem.

- Create the appropriate layout and components
- Create 2 routes / pages – one for the "**models**" and one for the "**stats**"