JOHNS HOPKINS
UNIVERSITY

Track patient recovery in real-time by processing streaming data

BIOMEDICAL DATA DESIGN

TA: Haoyin Xu

Group:Zhenyu Xiao
Haobin Zhou
Yimeng Xu
Emma Cardenas

Content

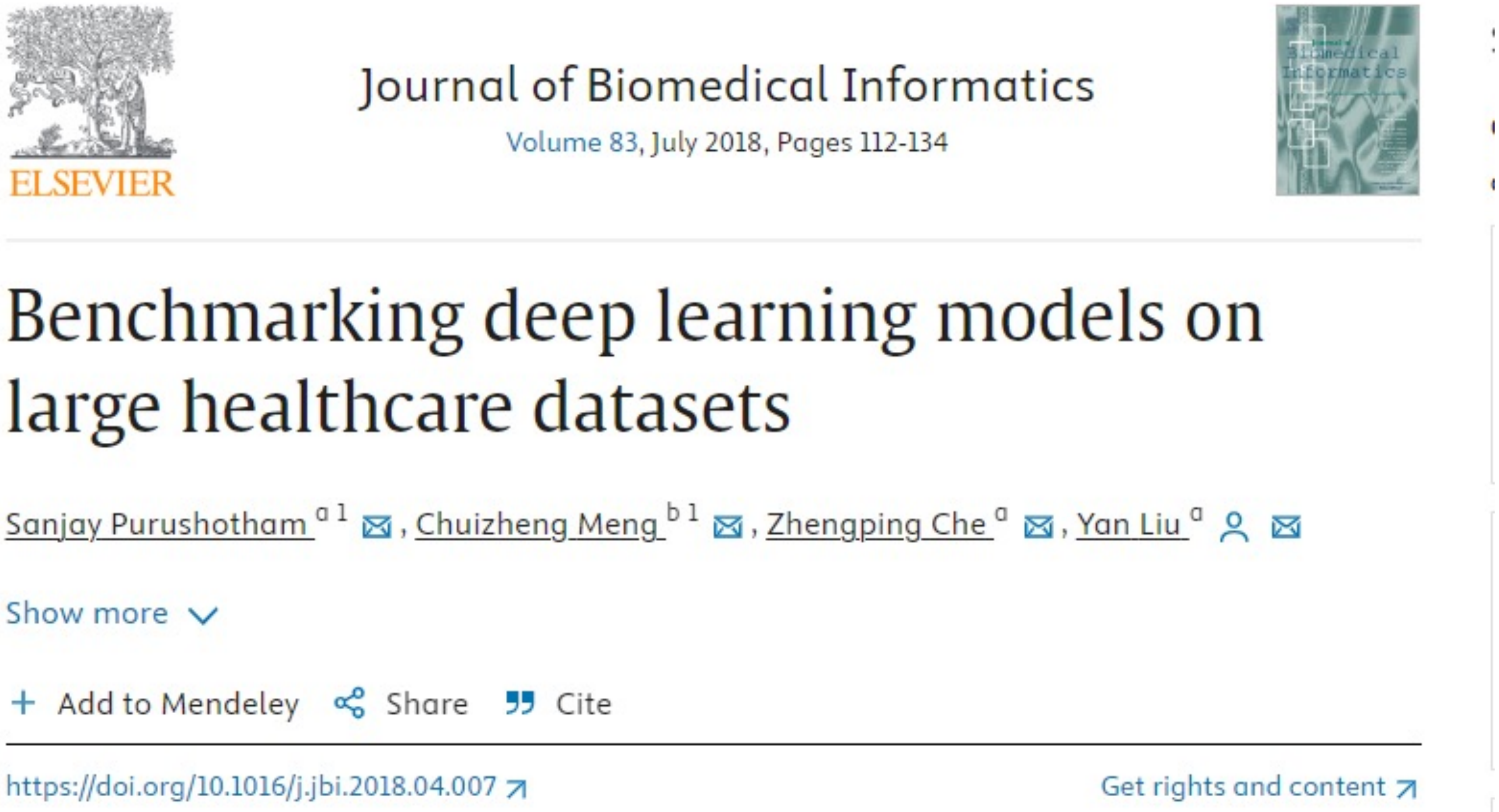
01 **SAPS II: 17 features**

02 **Missing data processing**

01

SAPS II: 17 features

01 17 features



Journal of Biomedical Informatics

Volume 83, July 2018, Pages 112-134

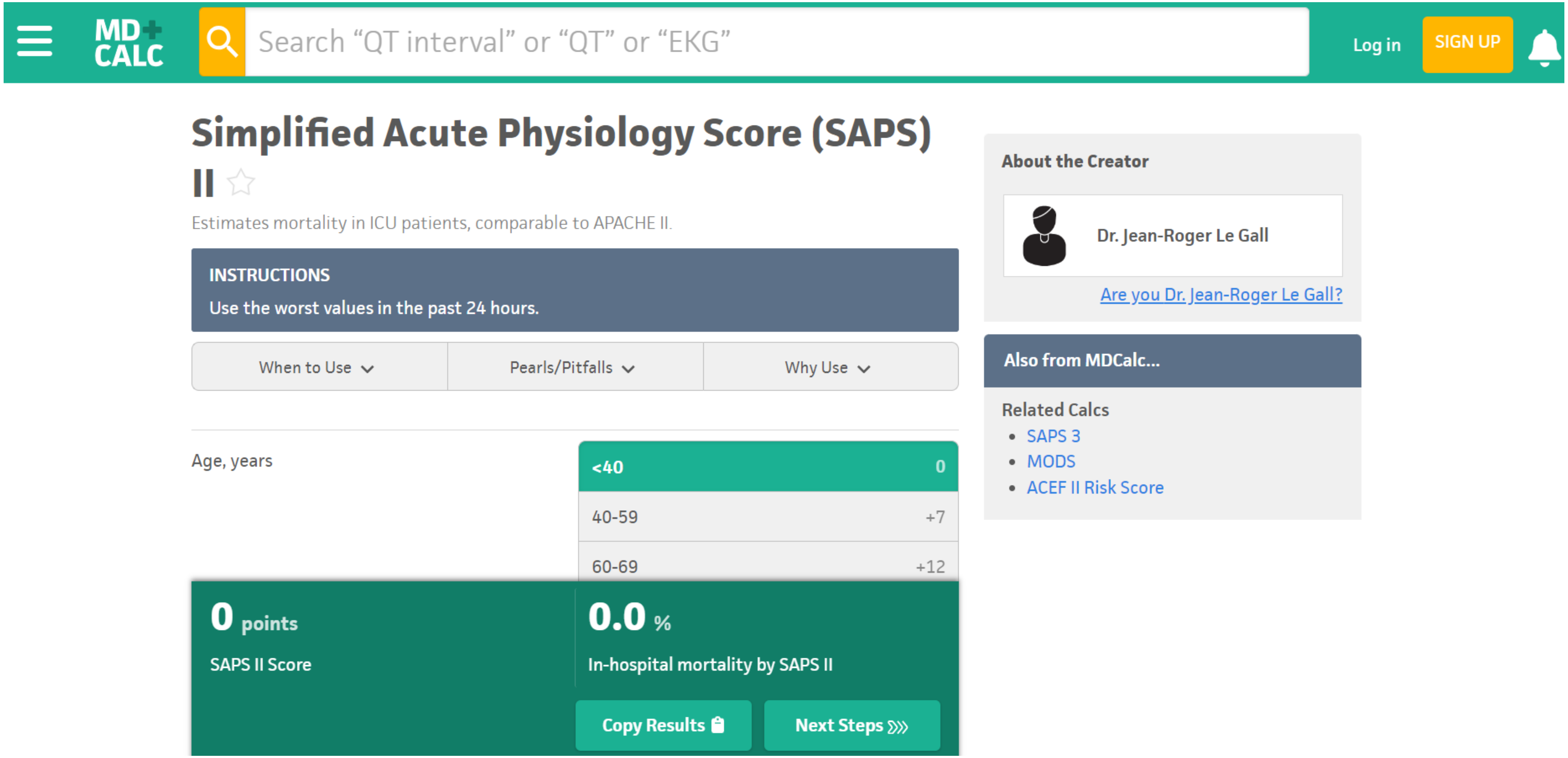
Benchmarking deep learning models on large healthcare datasets

Sanjay Purushotham [a, 1] ✉ , Chuizheng Meng [b, 1] ✉ , Zhengping Che [a] ✉ , Yan Liu [a] 👤 ✉

https://doi.org/10.1016/j.jbi.2018.04.007 ↗

01 **17 features**

How to get Physiology Score



Simplified Acute Physiology Score (SAPS) II ☆

Estimates mortality in ICU patients, comparable to APACHE II.

INSTRUCTIONS

Use the worst values in the past 24 hours.

Age, years		
	<40	0
	40-59	+7
	60-69	+12

0 points

SAPS II Score

0.0 %

In-hospital mortality by SAPS II

About the Creator

Dr. Jean-Roger Le Gall

Are you Dr. Jean-Roger Le Gall?

Also from MDCalc...

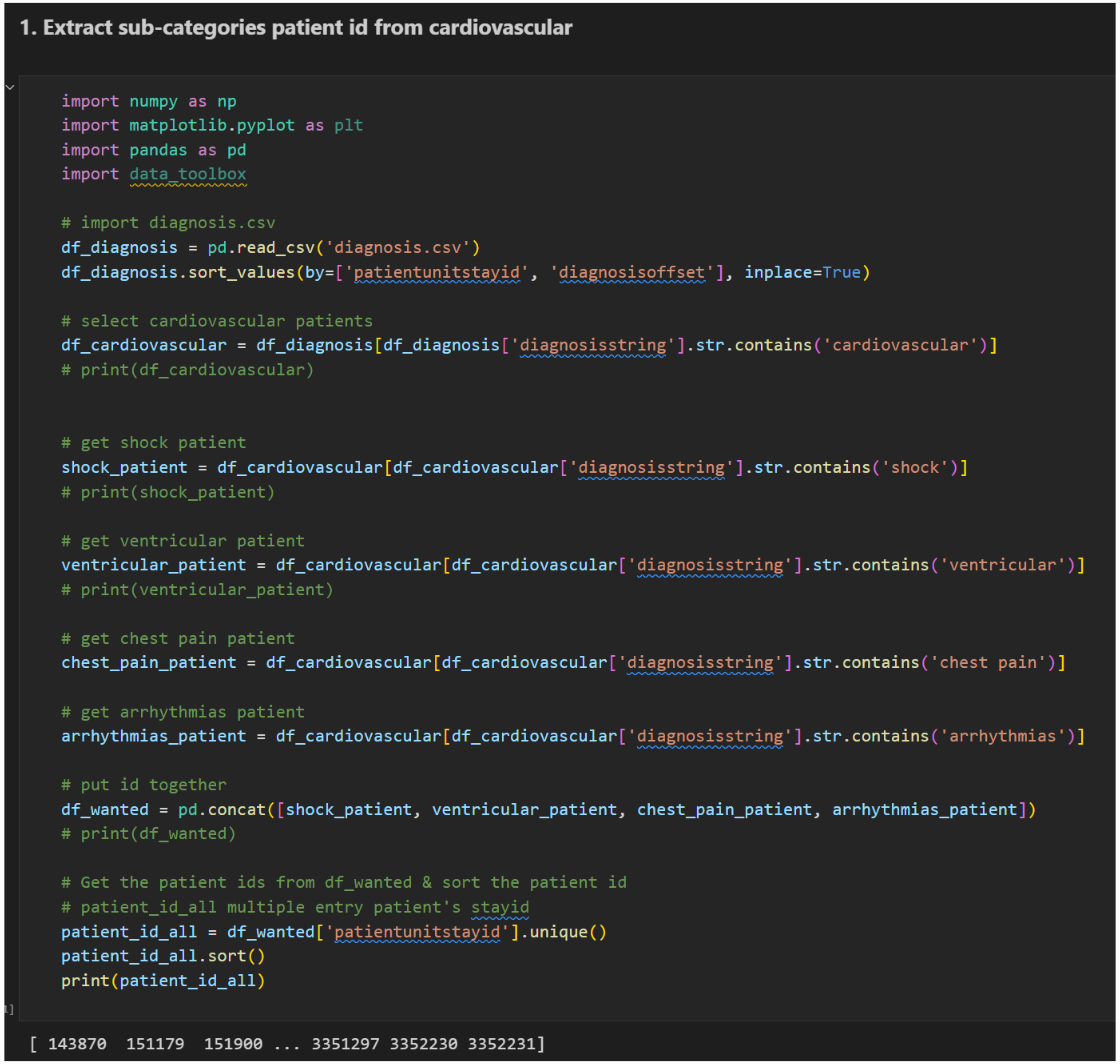
Related Calcs

- SAPS 3
- MODS
- ACEF II Risk Score

02

Missing data processing

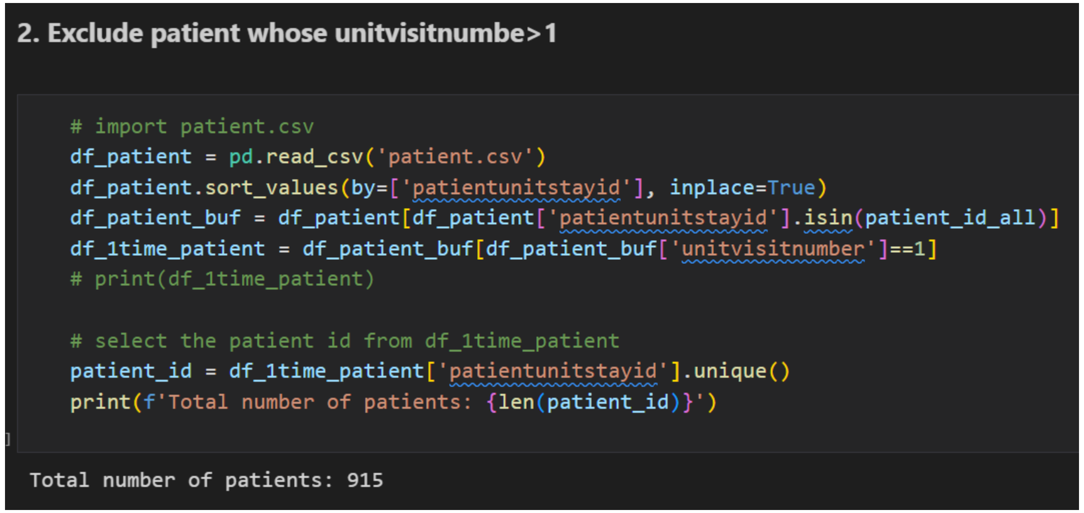
02 Data Preprocessing(Heart rate)



1. Extract sub-categories patient id from cardiovascular

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import data_toolbox

# import diagnosis.csv
df_diagnosis = pd.read_csv('diagnosis.csv')
df_diagnosis.sort_values(by=['patientunitstayid', 'diagnosisoffset'], inplace=True)

# select cardiovascular patients
df_cardiovascular = df_diagnosis[df_diagnosis['diagnosisstring'].str.contains('cardiovascular')]
# print(df_cardiovascular)


# get shock patient
shock_patient = df_cardiovascular[df_cardiovascular['diagnosisstring'].str.contains('shock')]
# print(shock_patient)

# get ventricular patient
ventricular_patient = df_cardiovascular[df_cardiovascular['diagnosisstring'].str.contains('ventricular')]
# print(ventricular_patient)

# get chest pain patient
chest_pain_patient = df_cardiovascular[df_cardiovascular['diagnosisstring'].str.contains('chest pain')]

# get arrhythmias patient
arrhythmias_patient = df_cardiovascular[df_cardiovascular['diagnosisstring'].str.contains('arrhythmias')]

# put id together
df_wanted = pd.concat([shock_patient, ventricular_patient, chest_pain_patient, arrhythmias_patient])
# print(df_wanted)

# Get the patient ids from df_wanted & sort the patient id
# patient_id_all multiple entry patient's stayid
patient_id_all = df_wanted['patientunitstayid'].unique()
patient_id_all.sort()
print(patient_id_all)
```

```
[ 143870  151179  151900 ... 3351297 3352230 3352231]
```



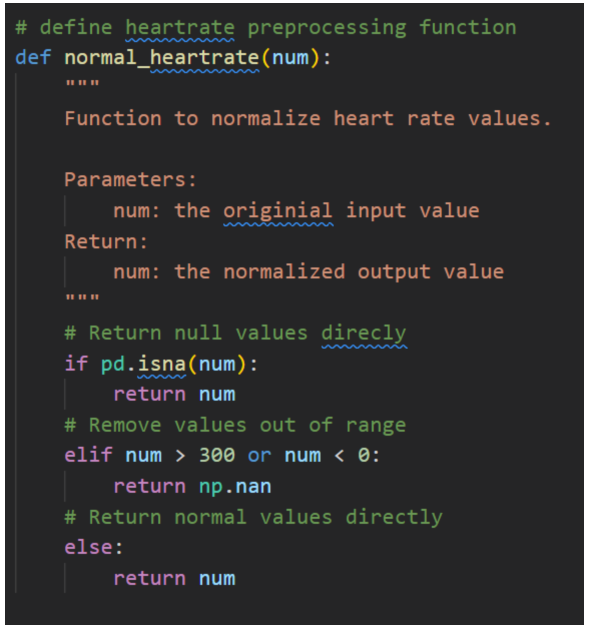
2. Exclude patient whose unitvisitnumbe>1

```python
# import patient.csv
df_patient = pd.read_csv('patient.csv')
df_patient.sort_values(by=['patientunitstayid'], inplace=True)
df_patient_buf = df_patient[df_patient['patientunitstayid'].isin(patient_id_all)]
df_1time_patient = df_patient_buf[df_patient_buf['unitvisitnumber']==1]
# print(df_1time_patient)

# select the patient id from df_1time_patient
patient_id = df_1time_patient['patientunitstayid'].unique()
print(f'Total number of patients: {len(patient_id)}')
```

```
Total number of patients: 915
```

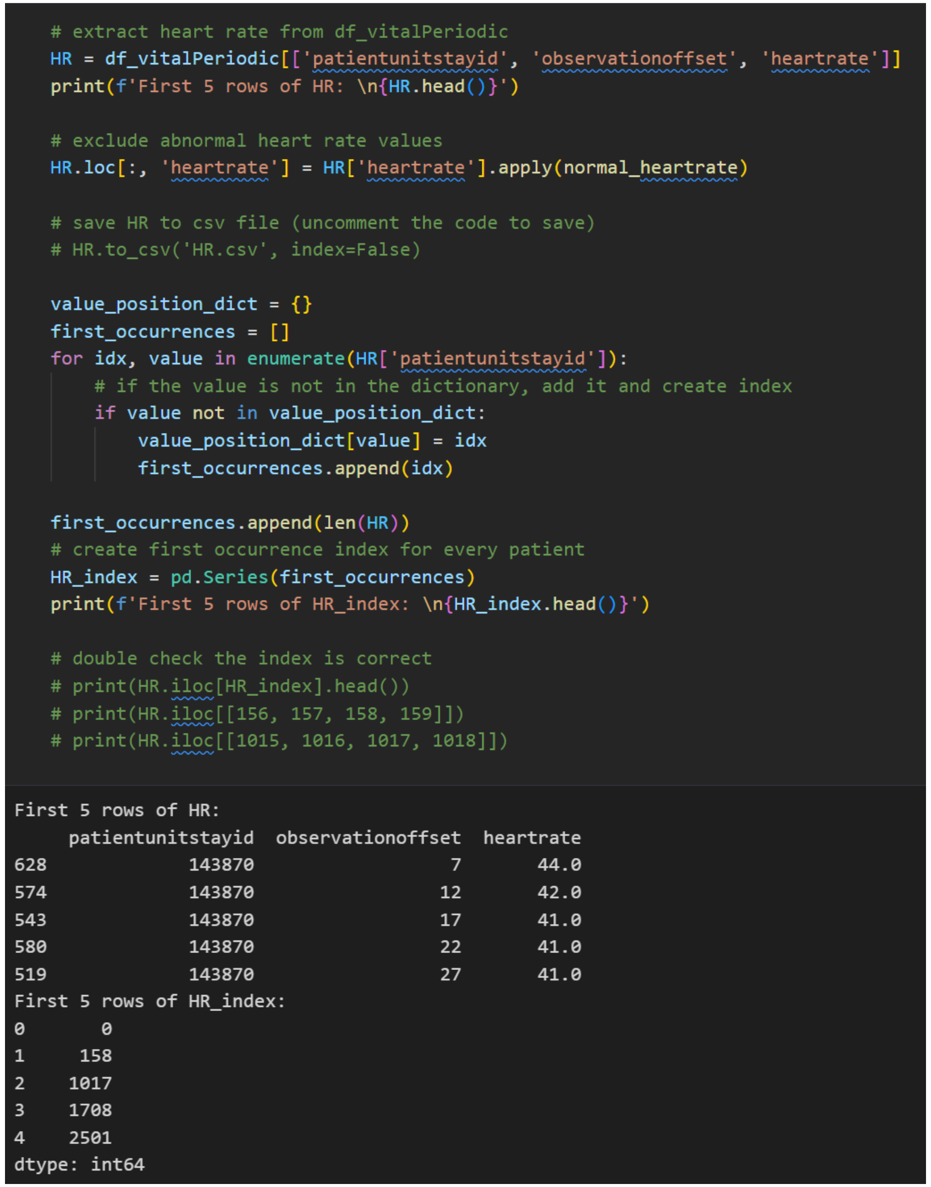
915 Patients valid

02 Data Preprocessing(Heart rate)



```
# define heartrate preprocessing function
def normal_heartrate(num):
    """
    Function to normalize heart rate values.

    Parameters:
        num: the originial input value
    Return:
        num: the normalized output value
    """
    # Return null values direcly
    if pd.isna(num):
        return num
    # Remove values out of range
    elif num > 300 or num < 0:
        return np.nan
    # Return normal values directly
    else:
        return num
```

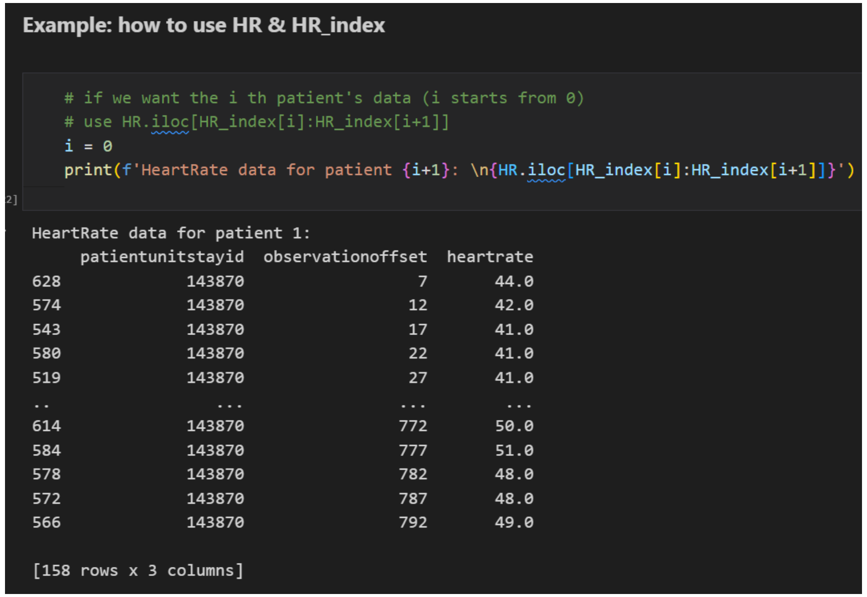
Filter the abnormal ones



```
# extract heart rate from df_vitalPeriodic
HR = df_vitalPeriodic[['patientunitstayid', 'observationoffset', 'heartrate']]
print(f'First 5 rows of HR: \n{HR.head()}')

# exclude abnormal heart rate values
HR.loc[:, 'heartrate'] = HR['heartrate'].apply(normal_heartrate)

# save HR to csv file (uncomment the code to save)
# HR.to_csv('HR.csv', index=False)

value_position_dict = {}
first_occurrences = []
for idx, value in enumerate(HR['patientunitstayid']):
    # if the value is not in the dictionary, add it and create index
    if value not in value_position_dict:
        value_position_dict[value] = idx
        first_occurrences.append(idx)

first_occurrences.append(len(HR))
# create first occurrence index for every patient
HR_index = pd.Series(first_occurrences)
print(f'First 5 rows of HR_index: \n{HR_index.head()}')

# double check the index is correct
# print(HR.iloc[HR_index].head())
# print(HR.iloc[[156, 157, 158, 159]])
# print(HR.iloc[[1015, 1016, 1017, 1018]])
```

```
First 5 rows of HR:
     patientunitstayid  observationoffset  heartrate
628             143870                  7       44.0
574             143870                 12       42.0
543             143870                 17       41.0
580             143870                 22       41.0
519             143870                 27       41.0
First 5 rows of HR_index:
0       0
1     158
2    1017
3    1708
4    2501
dtype: int64
```

Sorted by a 2×1 vector



Example: how to use HR & HR_index

```
# if we want the i th patient's data (i starts from 0)
# use HR.iloc[HR_index[i]:HR_index[i+1]]
i = 0
print(f'HeartRate data for patient {i+1}: \n{HR.iloc[HR_index[i]:HR_index[i+1]]}')
```

```
HeartRate data for patient 1:
     patientunitstayid  observationoffset  heartrate
628             143870                  7       44.0
574             143870                 12       42.0
543             143870                 17       41.0
580             143870                 22       41.0
519             143870                 27       41.0
..                 ...                ...        ...
614             143870                772       50.0
584             143870                777       51.0
578             143870                782       48.0
572             143870                787       48.0
566             143870                792       49.0

[158 rows x 3 columns]
```

Index to search certain patients

Thank you