Assignment # 1: Logistic Regression, MIMIC-IV, and 48-Hour Mortality Prediction

Fall 2023 BINF 4008 / COMS 4995: Advanced Machine Learning for Health and Medicine

This notebook contains the programming component for Assignment #1 worth 50 points.

As mentioned in the written assignment instructions, your submission should contain:

- `{Your UNI}_assignment_1_written.pdf`: A `PDF` file with your answers for the written questions typeset in $\LaTeX$.
- `{Your UNI}_assignment_1_code.ipynb`: Your answers for the programming questions as a `Jupyter` notebook.
- `{Your UNI}_assignment_1_code.{filetype}`: The same `Jupyter` notebook as either a `PDF` document or `html` file.

Best of luck!

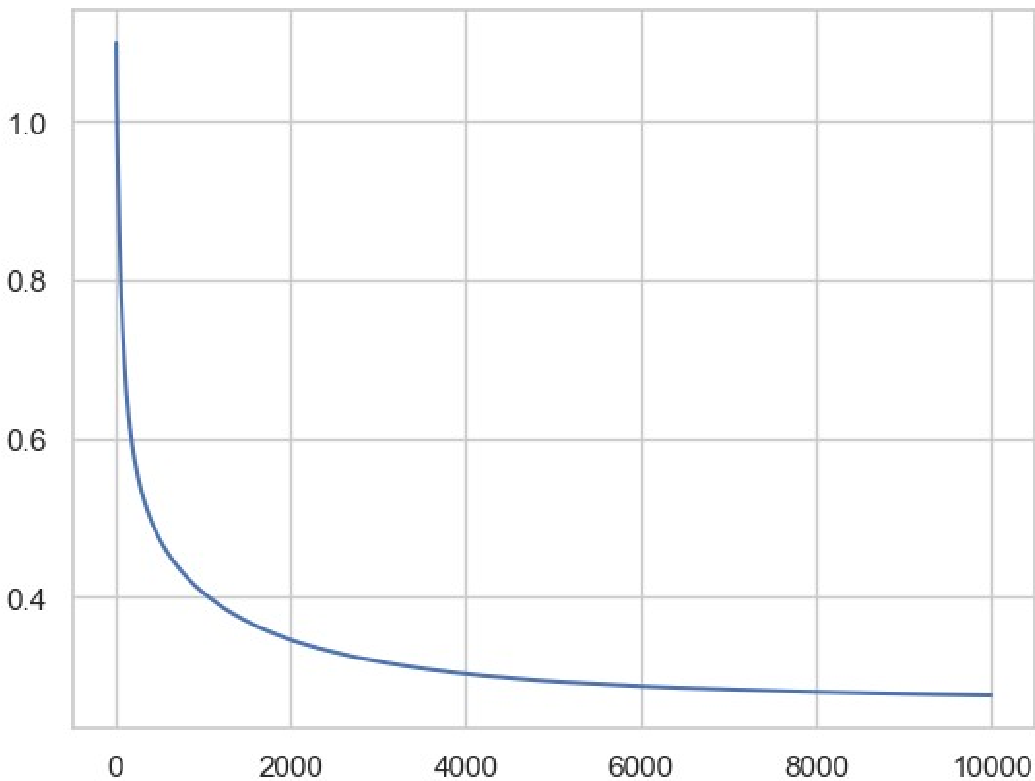
1. Logistic Regression and Gradient Descent

1. Get the `Iris` dataset using the `load_iris()` function from `sklearn.datasets`. Implement gradient descent for logistic regression with L1 regularization using the base `numpy` package and a 80:20 train:test split. Plot the loss curve and ROC curves, and report AUROC and accuracy score for each class.
   - You may use `sklearn.preprocessing` and `sklearn.model_selection` for this question but **not** `sklearn.linear_model`.
   - Don't forget that the `Iris` dataset is a 3-class classification problem and to add a bias term to your model!
2. Run the model for at least 5 additional different regularization strength values. Plot weights using `matplotlib.pyplot.stem`.

```
from sklearn.datasets import load_iris
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve, auc

data = load_iris()
X_train, X_test, y_train, y_test = train_test_split(
    data.data, data.target, test_size=0.2
)
```

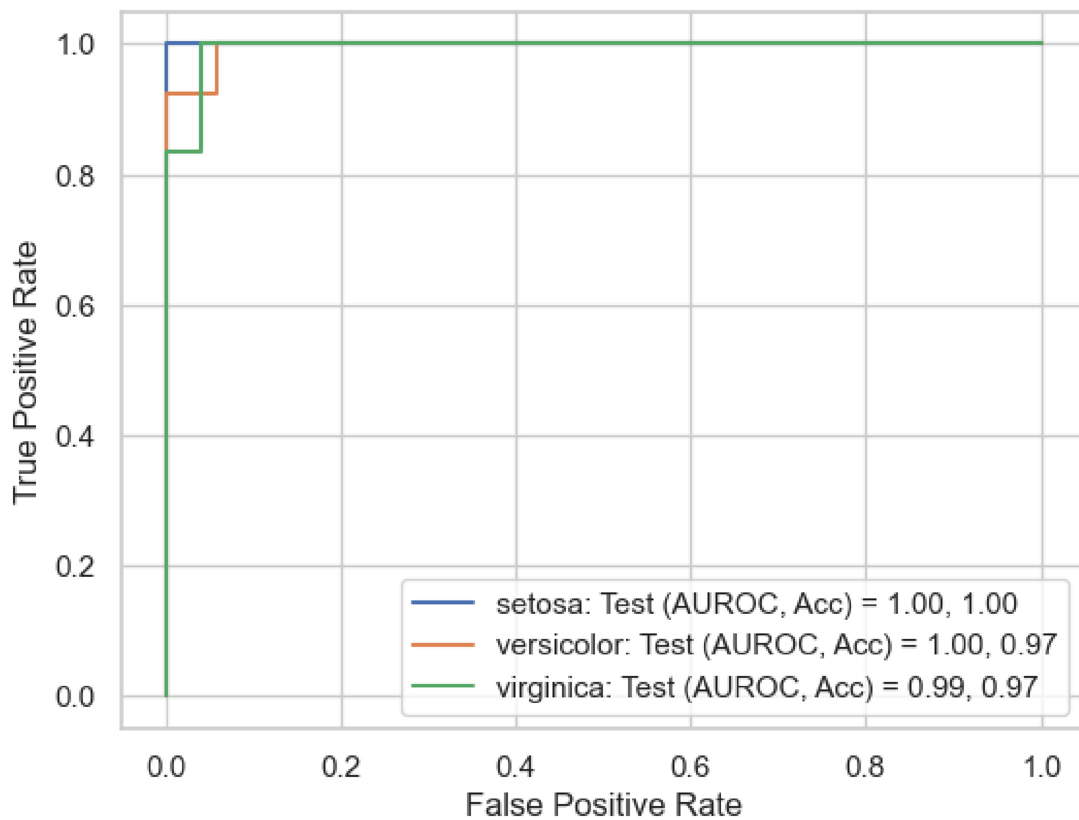
```
X_train = np.hstack([np.ones((X_train.shape[0], 1)), X_train])
X_test = np.hstack([np.ones((X_test.shape[0], 1)), X_test])

class LogisticRegression:
    def __init__(self, lr=0.01, epochs=1000, alpha=0):
        self.lr = lr
        self.epochs = epochs
        self.alpha = alpha
        self.coef_ = None

    def fit(self, X, y):
        num_classes = len(np.unique(y))
        num_features = X.shape[1]
        self.coef_ = np.zeros((num_features, num_classes))
        losses = []
        for epoch in range(self.epochs):
            softmax_scores = self.softmax(X @ self.coef_)
            gradients = self.calculate_gradient(X, y, softmax_scores)
            self.coef_ -= self.lr * gradients
            loss = self.calculate_loss(X, y, softmax_scores)
            losses.append(loss)
        return losses

    def calculate_gradient(self, X, y, softmax_scores):
        m, num_features = X.shape
        m, num_classes = softmax_scores.shape
        gradient = np.zeros((num_features, num_classes))
        for class_idx in range(num_classes):
            y_binary = y == class_idx
            gradient[:, class_idx] = X.T @ (softmax_scores[:,
class_idx] - y_binary)
        l1_gradient = self.alpha * np.sign(self.coef_)
        gradient += l1_gradient
        return gradient

    def calculate_loss(self, X, y, softmax_scores):
        num_rows = X.shape[0]
        loss = 0
        for class_idx in np.unique(y):
            y_binary = y == class_idx
            loss -= np.sum(y_binary * np.log(softmax_scores[:,
class_idx]))
        l1_loss = self.alpha * np.sum(np.abs(self.coef_))
        return (loss + l1_loss) / num_rows

    def predict(self, X):
        return np.argmax(self.softmax(X @ self.coef_), axis=1)

    def softmax(self, xs):
        """Compute softmax values for each sets of scores in x.
```

```
        softmax(np.array([-1, 0, 3, 5])) = [0.0021657, 0.00588697,
0.11824302, 0.87370431]
        """
        e_x = np.exp(xs - np.max(xs, axis=1, keepdims=True))
        return e_x / np.sum(e_x, axis=1, keepdims=True)

log_model = LogisticRegression(lr=0.0001, epochs=10000, alpha=1)
losses = log_model.fit(X_train, y_train)
plt.plot(losses)
plt.show()
print("Train accuracy: ", np.mean(log_model.predict(X_train) ==
y_train))
print("Test accuracy: ", np.mean(log_model.predict(X_test) == y_test))
```




```
Train accuracy:  0.975
Test accuracy:  0.9666666666666667

y_probs = log_model.softmax(X_test @ log_model.coef_)
y_preds = log_model.predict(X_test)

for i in range(y_probs.shape[1]):
    y_binary = y_test == i
    y_pred_binary = y_preds == i
```

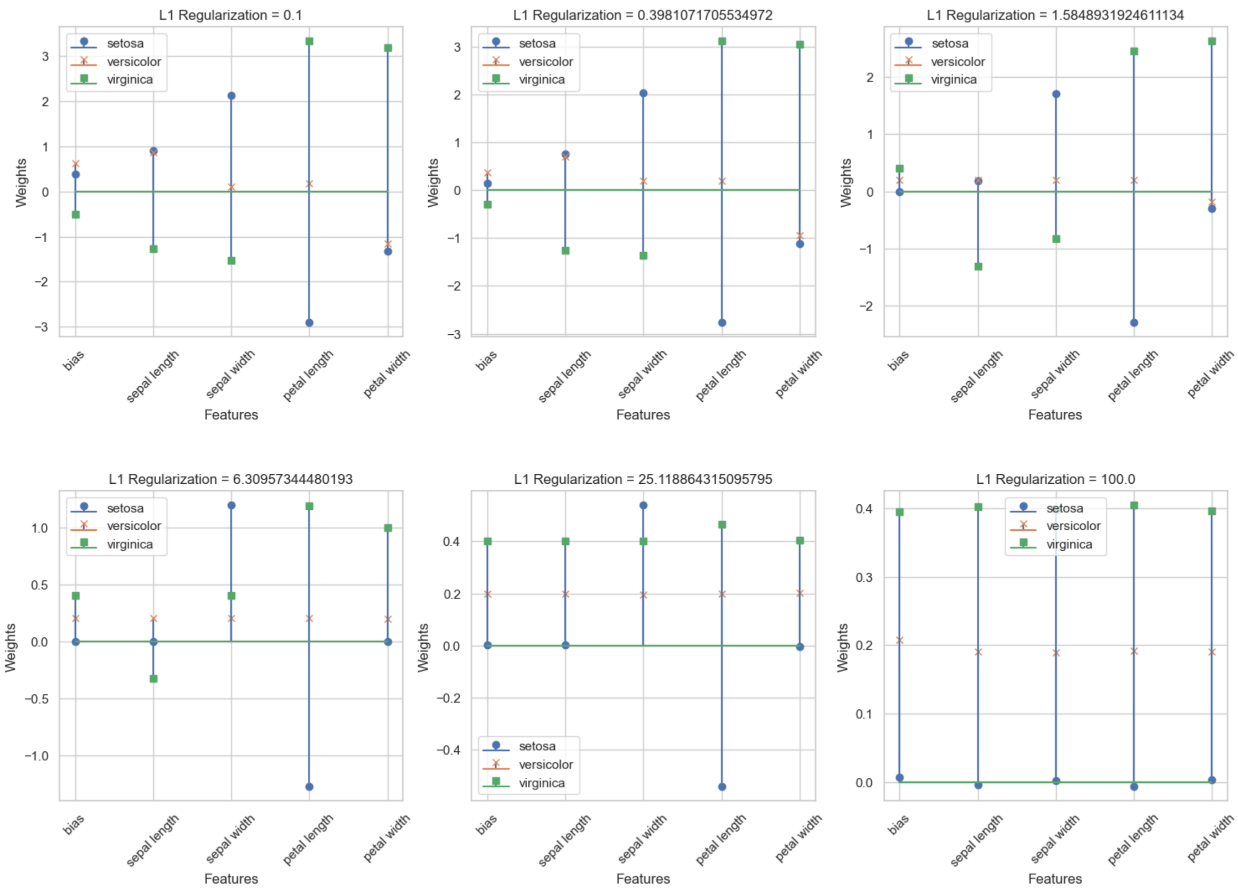
```
    fpr, tpr, _ = roc_curve(y_binary, y_probs[:, i])
    roc_auc = auc(fpr, tpr)
    class_acc = np.mean(y_pred_binary == y_binary)
    plt.plot(
        fpr,
        tpr,
        label=f"{data.target_names[i]}: Test (AUROC, Acc) =
{roc_auc:.2f}, {class_acc:.2f}",
    )
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.show()
```




```
alphas = np.logspace(-1, 2, 6)
weights = {}
test_dict = {}
for alpha in alphas:
    log_model = LogisticRegression(lr=0.0001, epochs=10000,
alpha=alpha)
    losses = log_model.fit(X_train, y_train)
    train_preds = log_model.predict(X_train)
    test_preds = log_model.predict(X_test)
```

```
    test_softmax = log_model.softmax(X_test @ log_model.coef_)
    test_loss = log_model.calculate_loss(X_test, y_test, test_softmax)

    # find the average auc
    y_probs = log_model.softmax(X_test @ log_model.coef_)
    y_preds = log_model.predict(X_test)
    auc_list = []
    for i in range(y_probs.shape[1]):
        y_binary = y_test == i
        y_pred_binary = y_preds == i
        fpr, tpr, _ = roc_curve(y_binary, y_probs[:, i])
        roc_auc = auc(fpr, tpr)
        auc_list.append(roc_auc)
    avg_auc = np.mean(auc_list)
    test_dict[alpha] = {
        "train loss": losses[-1],
        "test loss": test_loss,
        "train acc": np.mean(train_preds == y_train),
        "test acc": np.mean(test_preds == y_test),
        "avg auc": avg_auc,
    }
    weights[alpha] = log_model.coef_
plt.figure(figsize=(18, 12))
for i, (alpha, weight) in enumerate(weights.items()):
    plt.subplot(2, 3, i + 1)
    # make space between plots
    plt.subplots_adjust(hspace=0.5)
    plt.stem(
        weight[:, 0], markerfmt="C0o", basefmt="C0-",
label=f"{data.target_names[0]}"
    )
    plt.stem(
        weight[:, 1] + 0.2,
        markerfmt="C1x",
        basefmt="C1-",
        label=f"{data.target_names[1]}",
    )
    plt.stem(
        weight[:, 2] + 0.4,
        markerfmt="C2s",
        basefmt="C2-",
        label=f"{data.target_names[2]}",
    )
    plt.title(f"L1 Regularization = {alpha}")
    plt.xlabel("Features")
    plt.ylabel("Weights")
    plt.xticks(
        range(5), ["bias", "sepal length", "sepal width", "petal
length", "petal width"]
```

```python
    )
    plt.xticks(rotation=45)
    plt.legend()
plt.show()
```




```python
# print the output of val_dict pretty with 2 decimal places, round the index to 2 decimal places
print("Test Set Results:")
test_dict = pd.DataFrame(test_dict).T
test_dict = test_dict.round(2)
test_dict.index = test_dict.index.round(2)
print(test_dict)
```

```
Test Set Results:
        train loss  test loss  train acc  test acc  avg auc
0.10          0.13       0.22       0.98      0.97     1.00
0.40          0.19       0.41       0.98      0.97     1.00
1.58          0.34       0.87       0.98      0.97     0.99
6.31          0.64       1.40       0.92      0.80     0.99
25.12         0.97       1.73       0.69      0.57     0.91
100.00        1.15       1.40       0.37      0.20     0.78
```

Discuss which parameter you will use for the Iris dataset and why.

I would use the regularization parameter of α = 1.58 because on the unseen test set it has the highest accuracy score and the highest AUROC score, while minimizing the complexity of the model. To make sure this is generalizable, there should be an additional evaluation set that is not used for hyperparameter tuning.

---

2. MIMIC-IV Preprocessing

1. Build your study cohort using your cohort definition from question 2.1 in the written questions. Visualize the distributions for the following:

- Demographic features: age, gender, insurance, racial identity.
- Vitals: Blood pressure, oxygen-levels.
- Lab values.
- Label presence.

1. Remove outlier patients based on out-of-range values. Some resources you may want to explore include:

- Tables for acceptable ranges for physiological variables.
- Prior work on ML-based ICU mortality prediction.

```
import os
from tableone import TableOne
from collections import Counter
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pickle
from tqdm import tqdm
import gc
from datetime import datetime

from preprocess import *
from get_tables import *
```

Problem Definition

We are attempting to train a classifier to predict 48-hour in-hospital mortality using data collected in the first 24-hours of an ICU stay.

Getting File Paths, Organizing Unique Identifiers

From the official Data Description for MIMIC-IV we know we have two sets of folders:

- `hosp`: data from the entire hospital.
- `icu`: data from the iMDSoft system used in BIDMC's ICU units.

Using the `get_files_of_type()` function defined from the block above, we're going to get a `dict` with the file name as the `key` and the filepath as the `value`.

We'll start by using the `admissions` table from the `hosp` and the `icustays` table from the `icu`.

Note that the `admissions` table has `subject_id`, `hadm_id` columns while the `icustays` table has columns `subject_id`, `hadm_id`, and `stay_id` columns. The relationships between the UIDs as follows:

- Subjects and `subject_id` have a one-to-one relationship.
- Each `hadm_id` is tied to one admission to the hospital (not necessarily the ICU) for a given patient. One `subject_id` may have several `hadm_id` associated with it.
- Each `stay_id` is tied to one stay at the ICU. One `hadm_id` may have several `stay_id` associated with it.

Therefore, we have to filter out `stay_id` with lengths of stay longer than 24 hours and get their respective timestamps from ICU admission to discharge. Additionally, we have to pull additional data from tables in both in the `hosp` and the `icu` folders. For things like patient demographic information (patient gender, age) we'll match using `subject_id`, while for other values of interest (lab values, comorbidities, etc.) we'll use `hadm_id` and the relative dates of the events.

```
# Get dictionary of paths to csvs
MIMIC_HOSP_PARENT_PATH = os.path.join("D:", "AIDA", "data", "mimic-
iv", "iv", "hosp")
MIMIC_ICU_PARENT_PATH = os.path.join("D:", "AIDA", "data", "mimic-iv",
"iv", "icu")

SAVEDIR = os.getcwd()

assert os.path.isdir(MIMIC_HOSP_PARENT_PATH), "MIMIC hospital folder
path is not valid"
assert os.path.isdir(MIMIC_ICU_PARENT_PATH), "MIMIC icu folder path is
not valid"
hosp_paths = get_files_of_type(MIMIC_HOSP_PARENT_PATH, "csv",
as_dict=True)
icu_paths = get_files_of_type(MIMIC_ICU_PARENT_PATH, "csv",
as_dict=True)
print(f"Hospital tables:\n{sorted(hosp_paths.keys())}\n")
print(f"ICU tables:\n{sorted(icu_paths.keys())}")

SCRATCH = True

if SCRATCH:
    print("Reading in data from scratch")
    # icu stays records
    icustays_df = pd.read_csv(icu_paths["icustays"])
    # hospital admissions records has los (length of stay)
    admissions_df = pd.read_csv(hosp_paths["admissions"])
    # datetime conversion
```

```
    icustays_df = dataframe_datetime(icustays_df)
    admissions_df = dataframe_datetime(admissions_df)

    # check that all hadm_ids for icu stays table are in the
admissions table
    assert
set(icustays_df["hadm_id"]).issubset(set(admissions_df["hadm_id"]))
else:
    icustays_df = pd.read_csv(os.path.join(SAVEDIR, "data",
"icustays.csv"))
```

```
Hospital tables:
['admissions', 'd_hcpcs', 'd_icd_diagnoses', 'd_icd_procedures',
'd_labitems', 'diagnoses_icd', 'drgcodes', 'emar', 'emar_detail',
'hcpcsevents', 'labevents', 'microbiologyevents', 'omr', 'patients',
'pharmacy', 'poe', 'poe_detail', 'prescriptions', 'procedures_icd',
'provider', 'services', 'transfers']

ICU tables:
['caregiver', 'chartevents', 'd_items', 'datetimeevents', 'icustays',
'ingredientevents', 'inputevents', 'outputevents', 'procedureevents']
Reading in data from scratch
```

```
if SCRATCH:
    print("Reading in data from scratch")
    # add additional columns fo 48-hour ICU mortality and hospital
admissions to icu stay table
    # keep only icustays with lengths of stay greater or equal to 24
hours
    ICU_LOS_MIN = 1
    icustays_df = icustays_df[icustays_df["los"] >= ICU_LOS_MIN]
    icu_hadm_ids = set(icustays_df["hadm_id"]) &
set(admissions_df["hadm_id"])
    admissions_df =
admissions_df[admissions_df["hadm_id"].isin(icu_hadm_ids)]
    subjects = set(admissions_df["subject_id"])

    # sourcery skip: identity-comprehension
    hadm_id_deathtime_dict = {
        hadm_id: deathtime
        for hadm_id, deathtime in admissions_df[
            admissions_df["hospital_expire_flag"] == 1
        ].apply(lambda row: (row["hadm_id"], row["deathtime"]), 1)
    }
    admission_time_dict = {
        hadm_id: admittime
        for hadm_id, admittime in admissions_df.apply(
            lambda row: (row["hadm_id"], row["admittime"]), 1
        )
    }
```

```
    icustays_df["admittime"] = icustays_df.apply(
        lambda row: admission_time_dict[row["hadm_id"]], 1
    )
    icustays_df["deathtime"] = icustays_df.apply(
        lambda row: hadm_id_deathtime_dict[row["hadm_id"]]
        if row["hadm_id"] in hadm_id_deathtime_dict
        else np.nan,
        1,
    )
    icustays_df["48_hour_mortality_flag"] = icustays_df.apply(
        lambda row: ((row["deathtime"] - row["intime"]) /
pd.Timedelta(hours=1) <= 48)
        & ((row["deathtime"] - row["intime"]) / pd.Timedelta(hours=1)
>= 24),
        1,
    )
    print(Counter(icustays_df["48_hour_mortality_flag"]))

    print(
        f'Label prevelance: {len(set(icustays_df["hadm_id"]))=},
{len(set(admissions_df["hadm_id"]))=}'
    )
    print(f"Number of admissions: {admissions_df.shape[0]}")

    # get subject age and gender
    patients_df = pd.read_csv(hosp_paths["patients"])
    patients_df =
patients_df[patients_df["subject_id"].isin(subjects)]
    print(patients_df.shape)
    display(patients_df.head(1))

    # Female is 1, Male is 0
    patients_df["gender"] = np.array(patients_df["gender"] ==
"F").astype(int)

    anchor_age_tuples = patients_df.apply(
        lambda row: (row["subject_id"], row["anchor_age"],
row["anchor_year"]), 1
    )

    anchor_age_dict = {
        subject_id: {"anchor_age": anchor_age, "anchor_year":
anchor_year}
        for subject_id, anchor_age, anchor_year in anchor_age_tuples
    }

    clean_mem(anchor_age_tuples)
    gender_dict = dict(zip(patients_df["subject_id"],
patients_df["gender"]))
    icustays_df["age"] = icustays_df.apply(
```

```
        lambda row: anchor_age_dict[row["subject_id"]]["anchor_age"]
        + (row["intime"].year - anchor_age_dict[row["subject_id"]]
["anchor_year"]),
        1,
    )
    clean_mem(patients_df)
    clean_mem(anchor_age_dict)

    # add the age and gender column
    icustays_df["age"] = icustays_df.apply(lambda row: min(row["age"],
90), 1)
    icustays_df["gender"] = icustays_df.apply(
        lambda row: gender_dict[row["subject_id"]], 1
    )

    clean_mem(gender_dict)
    length = icustays_df.shape[0]
    # add the race and insurance columns to the icustays table from
the admissions table
    icustays_df = pd.merge(
        icustays_df,
        admissions_df[["hadm_id", "race", "insurance"]],
        on="hadm_id",
        how="left",
        suffixes=("", "_adm"),
    )

    assert length == icustays_df.shape[0]
icustays_df = dataframe_datetime(icustays_df)
print(icustays_df.shape)
icustays_df.head(1)
```

```
Reading in data from scratch
Counter({False: 56818, True: 916})
Label prevelance: len(set(icustays_df["hadm_id"]))=53034,
len(set(admissions_df["hadm_id"]))=53034
Number of admissions: 53034
(42264, 6)
```

	subject_id	gender	anchor_age	anchor_year	anchor_year_group	dod
40	10001217	F	55	2157	2011 - 2013	NaN

```
(57734, 15)
```

	subject_id	hadm_id	stay_id	first_careunit
0	10001217	24597018	37067082	Surgical Intensive Care Unit (SICU)

	last_careunit	intime
0	Surgical Intensive Care Unit (SICU)	2157-11-20 19:18:02

```
               outtime       los           admittime deathtime  \
0 2157-11-21 22:08:00  1.118032 2157-11-18 22:56:00       NaT

   48_hour_mortality_flag  age  gender   race insurance
0                   False   55       1  WHITE     Other
```

```
# check if file exists, if not create it
if not os.path.exists(os.path.join(SAVEDIR, "data", "icustays.csv")):
    icustays_df.to_csv(os.path.join(SAVEDIR, "data", "icustays.csv"),
index=False)
```

Analyze Missingness in icustays_df

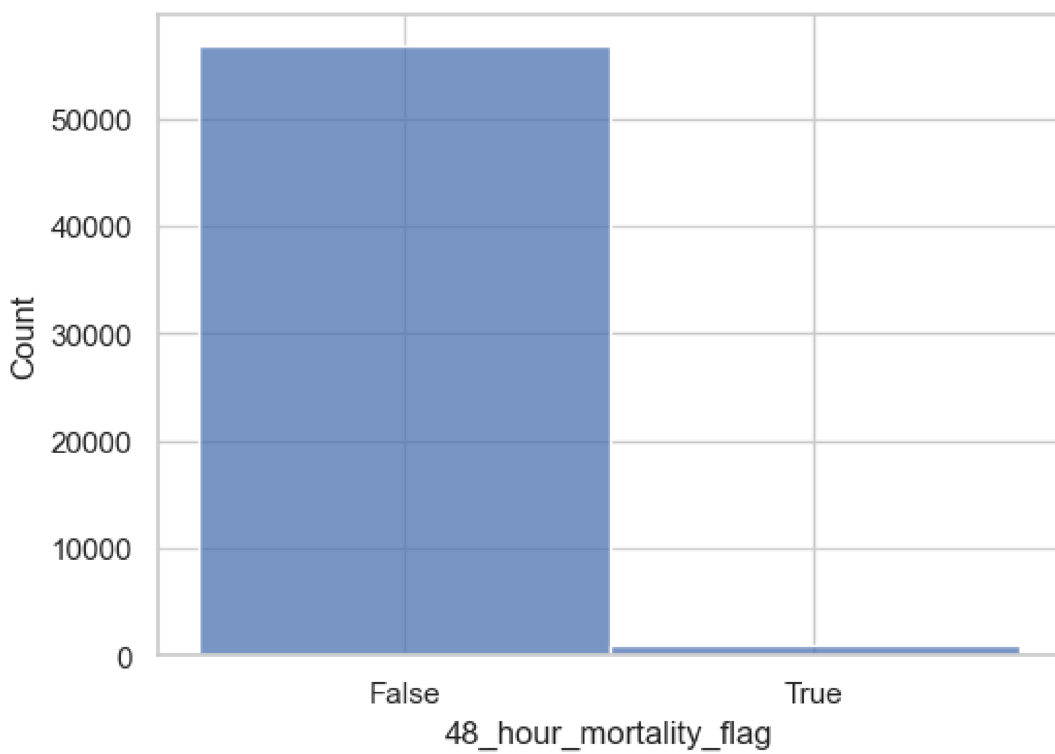
Visualizations for icustays_df

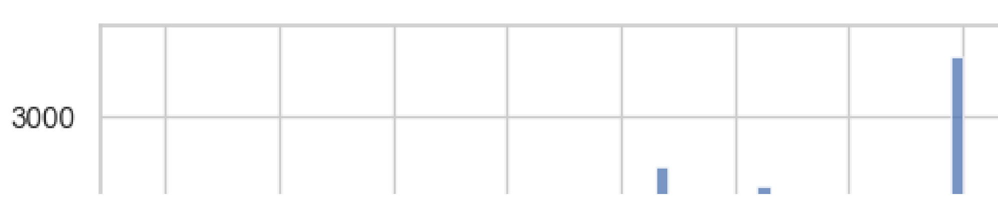
```
#clean up race col
import re
def clean_race(col):
    string = re.sub(r'ASIAN.*', 'ASIAN', col)
    string = re.sub(r'WHITE.*', 'WHITE', string)
    string = re.sub(r'BLACK.*', 'BLACK', string)
    string = re.sub(r'HISPANIC.*', 'HISPANIC', string)
    string = re.sub(r'UNABLE.*', 'UNKNOWN', string)
    string = re.sub(r'PATIENT.*', 'UNKNOWN', string)
    string = re.sub(r'PORT.*', 'SOUTH AMERICAN', string)
    string = re.sub(r'MULTIPLE.*', 'OTHER', string)
    string = re.sub(r'NATIVE.*', 'OTHER', string)
    string = re.sub(r'SOUTH AMERICAN.*', 'OTHER', string)
    string = re.sub(r'AMERICAN.*', 'OTHER', string)

    return string
icustays_df['race']=icustays_df.race.apply(clean_race)

icustays_df.race.value_counts(normalize=True, dropna=False)
```

```
race
WHITE       0.678993
UNKNOWN     0.108584
BLACK       0.104479
OTHER       0.041934
HISPANIC    0.036824
ASIAN       0.029186
Name: proportion, dtype: float64
```

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Set the style of the visualizations
sns.set(style="whitegrid")
# suppress warnings
import warnings

warnings.filterwarnings("ignore")

# Create subplots
fig, axes = plt.subplots(5, 1, figsize=(6, 30))
fig.suptitle("Distribution of 48-hour Mortality")
dem_feats = ["48_hour_mortality_flag", "age", "gender", "insurance",
"race"]
for i, feat in enumerate(dem_feats):
    if feat == "age":
        icustays_df[feat] = icustays_df[feat].astype(int)
        sns.histplot(data=icustays_df, x=feat, ax=axes[i])
        # axes[i].set_xticks(np.arange(0, 100, 10))
        continue
    icustays_df[feat] = icustays_df[feat].astype(str)
    sns.histplot(data=icustays_df, x=feat, ax=axes[i])


# make room between subplots
plt.subplots_adjust(hspace=0.5)
plt.show()
```

Distribution of 48-hour Mortality





Load vital signs data (chartevents)

- and their definitions (d_items)

```
if SCRATCH:
    print("Reading in data from scratch")
    # get vitals table (chartevents)
    chartevent_definitions = pd.read_csv(icu_paths["d_items"])
    chartevent_definitions = chartevent_definitions[
        (chartevent_definitions["linksto"] == "chartevents")
        & (chartevent_definitions["category"] == "Routine Vital 
Signs")
    ]
    routine_vital_items = chartevent_definitions["itemid"].values
    print(f"Number of unique routine vital sign item ids: 
{len(routine_vital_items)}")

    chartevents = pd.read_csv(icu_paths["chartevents"], 
chunksize=10000000)
    icu_stay_ids = set(icustays_df["stay_id"])

    for p in [
        os.path.join(".", "data"),
        os.path.join(".", "data", "mimic_chartevents"),
    ]:
        if not os.path.isdir(p):
            os.mkdir(p)

    mimic_chartevents_parent = os.path.join(".", "data", 
"mimic_chartevents")

    clean_mem(chartevents)

    if len(os.listdir(mimic_chartevents_parent)) == 0:
        for i, chunk in enumerate(chartevents):
            original_size = chunk.shape
            chunk = chunk[
                (chunk["stay_id"].isin(icu_stay_ids))
                & (chunk["itemid"].isin(routine_vital_items))
            ]
            chunk.to_csv(
                os.path.join(mimic_chartevents_parent, 
f"mimic_vitals_{i}.csv"),
                index=False,
            )
            print(f"Chunk {i}: selected {chunk.shape[0]} from 
{original_size[0]} rows.")

    chartevents_df = pd.concat(
        [
            pd.read_csv(
```

```
                path,
                dtype={
                    "subject_id": np.int32,
                    "hadm_id": np.int32,
                    "stay_id": np.int32,
                    "caregiver_id": np.int32,
                    "charttime": str,
                    "storetime": str,
                    "itemid": np.int32,
                    "value": str,
                    "valuenum": np.float64,
                    "valueuom": "category",
                    "warning": bool,
                },
            )
            for path in get_files_of_type(mimic_chartevents_parent,
filetype="csv")
        ],
        axis=0,
        ignore_index=True,
    )
    chartevents_df.reset_index(drop=True, inplace=True)
    chartevents_df = dataframe_datetime(chartevents_df)
    length = chartevents_df.shape[0]

    chartevents_df = pd.merge(
        chartevents_df,
        chartevent_definitions[["itemid", "label"]],
        on="itemid",
        how="left",
    )
    chartevents_df.drop(columns=["caregiver_id"], inplace=True)

    assert length == chartevents_df.shape[0]
else:
    chartevents_df = pd.read_csv(
        os.path.join(SAVEDIR, "data", "chartevents.csv"),
        dtype={
            "subject_id": np.int32,
            "hadm_id": np.int32,
            "stay_id": np.int32,
            "caregiver_id": np.int32,
            "charttime": str,
            "storetime": str,
            "itemid": np.int32,
            "value": str,
            "valuenum": np.float64,
            "valueuom": "category",
            "warning": bool,
```

```
        },
    )
print(f"{chartevents_df.shape=}")
chartevents_df = dataframe_datetime(chartevents_df)
chartevents_df.head(1)
```

```
Reading in data from scratch
Number of unique routine vital sign item ids: 50
chartevents_df.shape=(40665493, 11)

   subject_id   hadm_id   stay_id           charttime
storetime  \
0    10001217  24597018  37067082 2157-11-21 19:00:00 2157-11-21
19:37:00

   itemid value  valuenum valueuom  warning       label
0  220045   101     101.0      bpm    False  Heart Rate
```

```
if not os.path.exists(os.path.join(SAVEDIR, "data",
"chartevents.csv")):
    chartevents_df.to_csv(os.path.join(SAVEDIR, "data",
"chartevents.csv"), index=False)
```

Analyze Missingness in chartevents_df

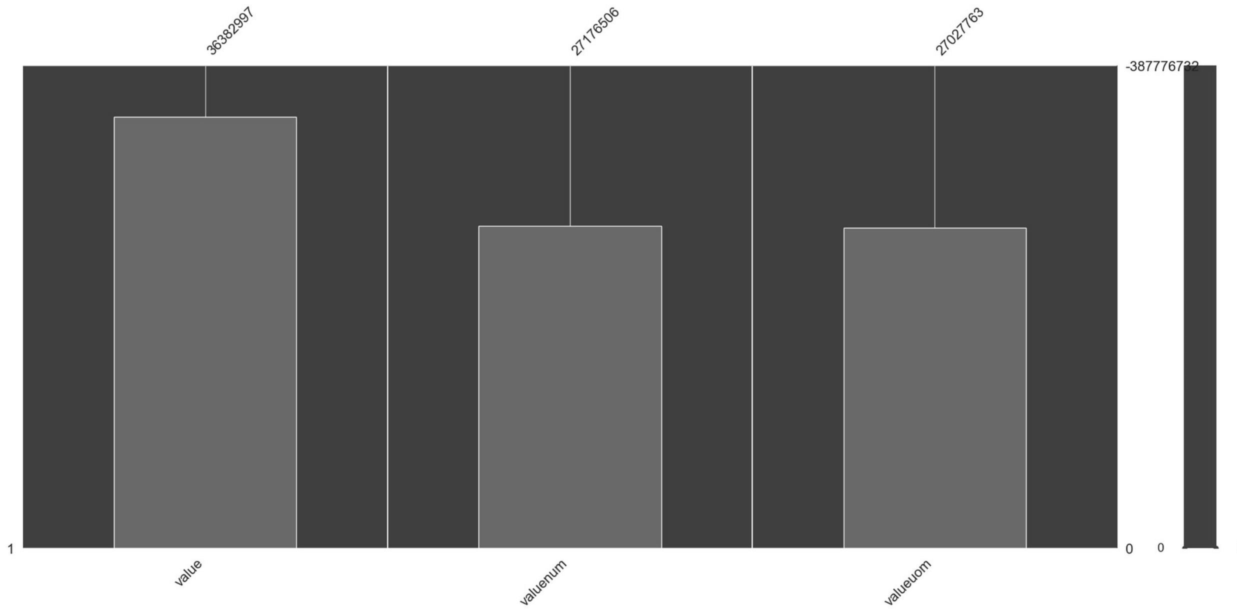
```
chartevents_df.info()
```

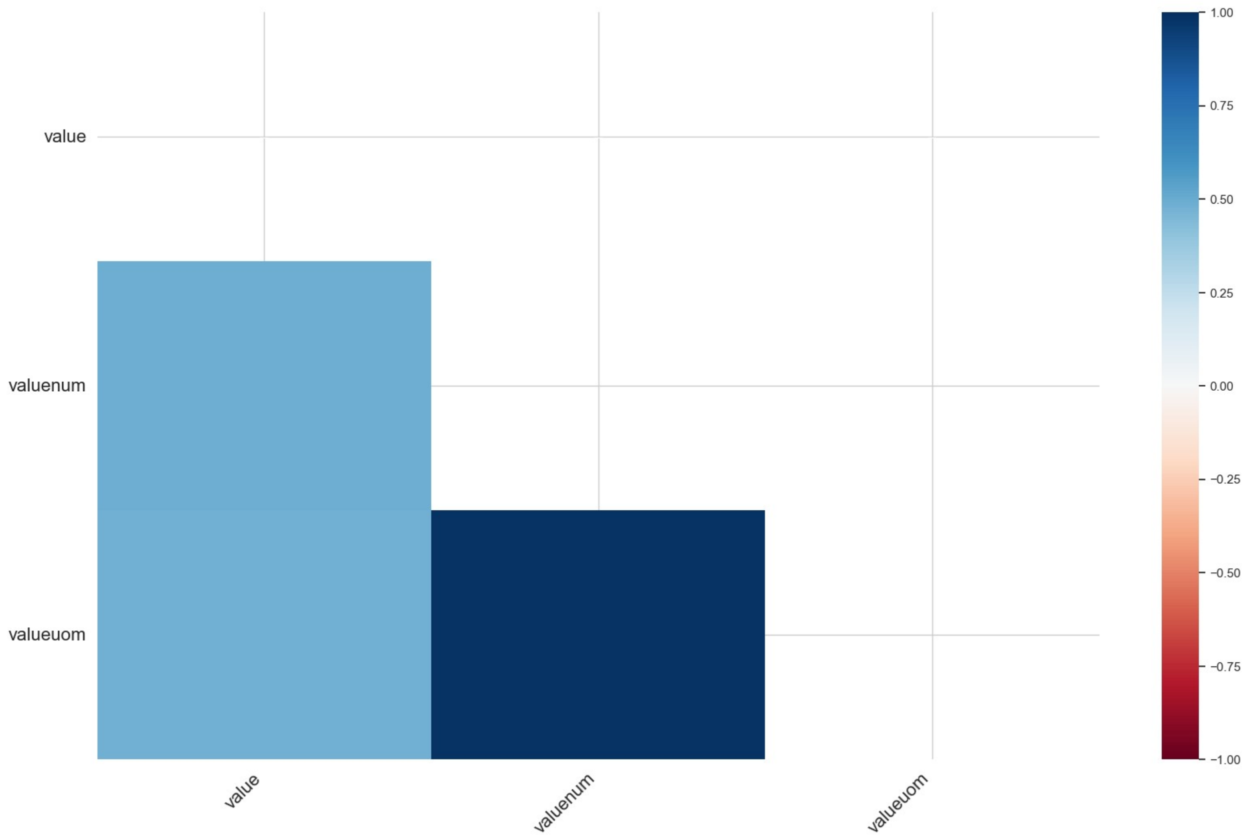
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40665493 entries, 0 to 40665492
Data columns (total 11 columns):
 #   Column      Dtype
---  ------      -----
 0   subject_id  int32
 1   hadm_id     int32
 2   stay_id     int32
 3   charttime   datetime64[ns]
 4   storetime   datetime64[ns]
 5   itemid      int32
 6   value       object
 7   valuenum    float64
 8   valueuom    category
 9   warning     bool
 10  label       object
dtypes: bool(1), category(1), datetime64[ns](2), float64(1), int32(4),
object(2)
memory usage: 2.2+ GB
```

```
from get_tables import *
try:
    missing_charts = missingness(df=chartevents_df)
```

```python
except:
    print('')
```

```
value       0.105310
valuenum    0.331706
valueuom    0.335364
dtype: float64
```

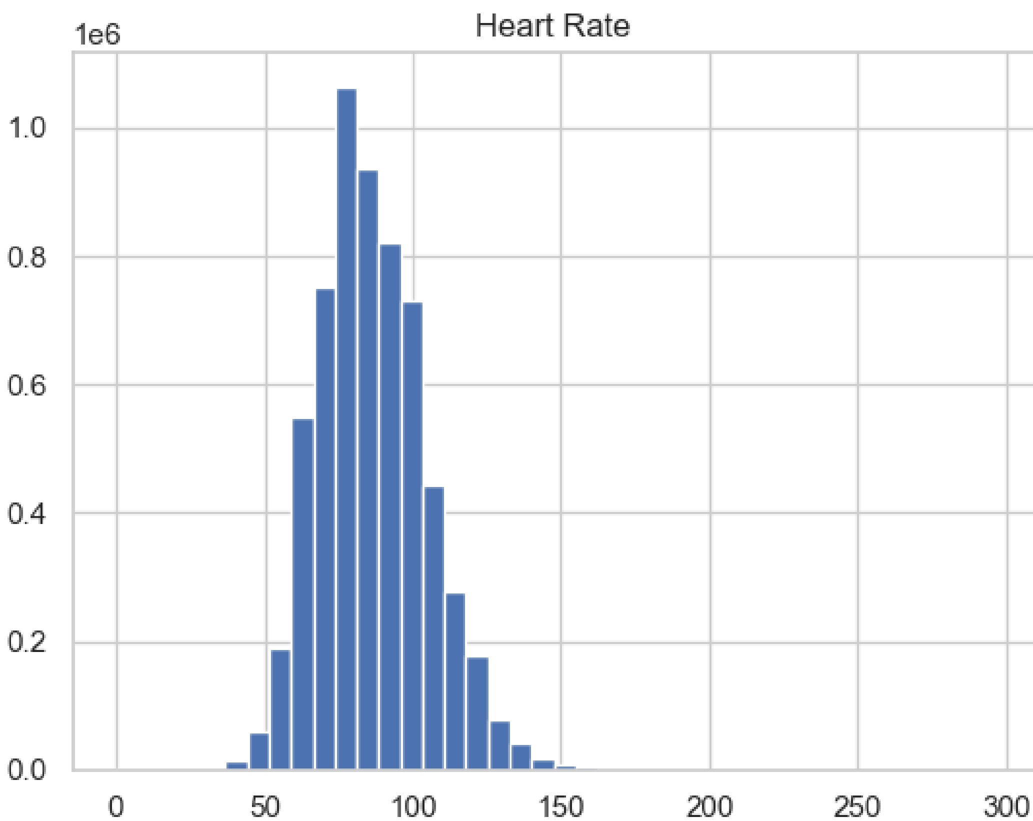




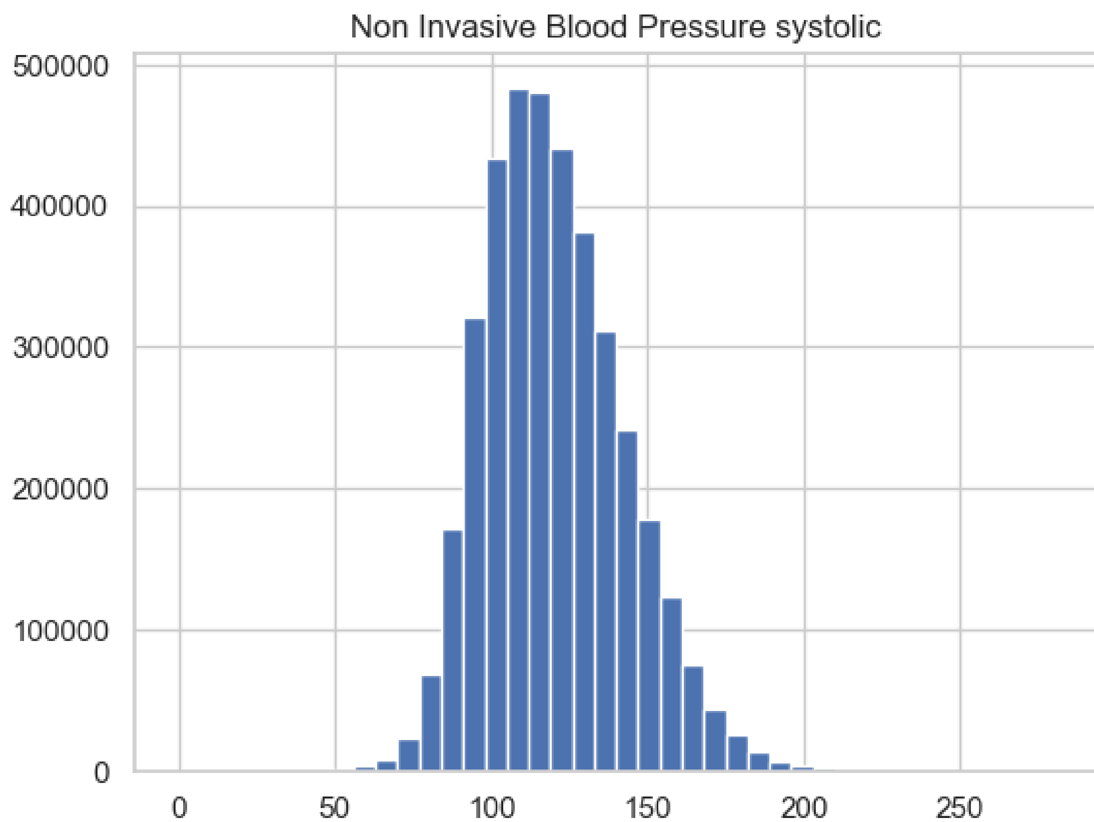

```
# Visualize histograms for label = 'Heart Rate' and label = 'Non
Invasive Blood Pressure systolic' and label = 'Non Invasive Blood
Pressure diastolic', and label = 'Non Invasive Blood Pressure mean'

# Set the style of the visualizations
sns.set(style="whitegrid")

# Create subplots
vital_feats = [
    "Heart Rate",
    "Non Invasive Blood Pressure systolic",
    "Non Invasive Blood Pressure diastolic",
    "Non Invasive Blood Pressure mean",
]
# exclude outliers
temp_df = chartevents_df[
    (chartevents_df["valuenum"] >= 0) & (chartevents_df["valuenum"] <=
300)
]
for feat in vital_feats:
    display(temp_df[temp_df["label"] == feat]["valuenum"].describe())
    plt.hist(temp_df[temp_df["label"] == feat]["valuenum"], bins=40)
    plt.title(feat)
```

```
    plt.show()
```

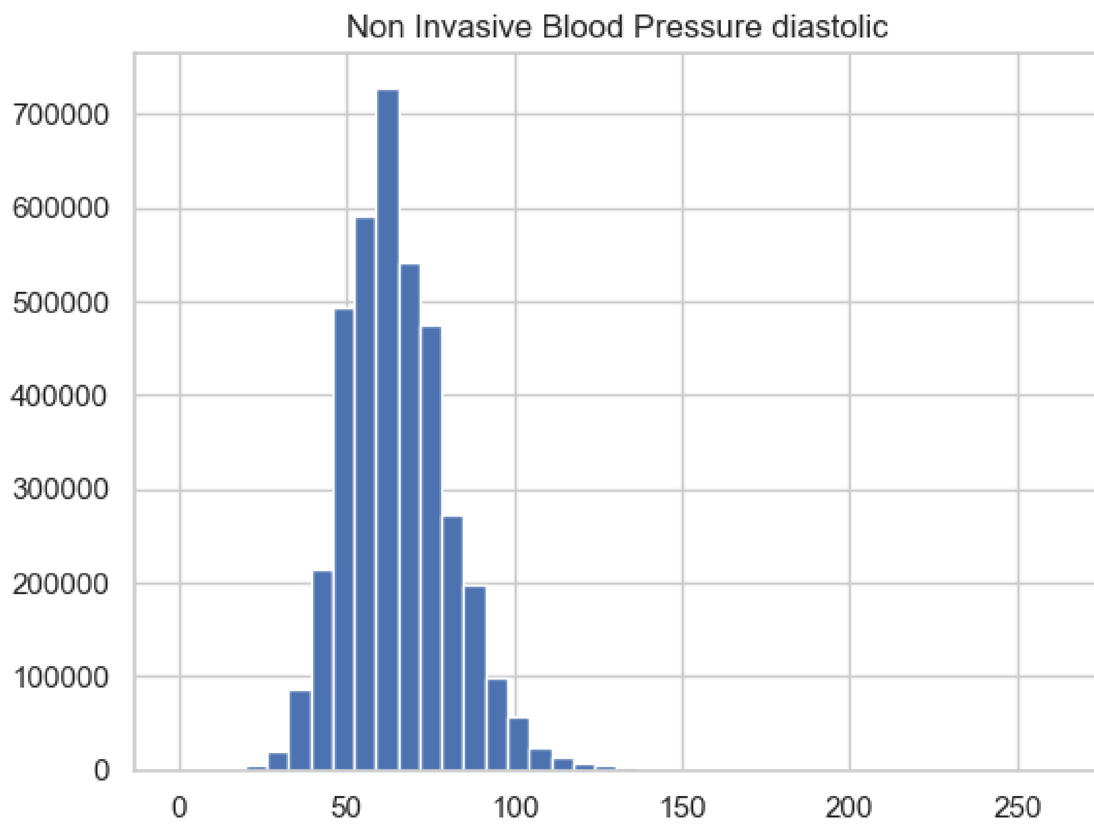
```
count    6.179707e+06
mean     8.633857e+01
std      1.841630e+01
min      0.000000e+00
25%      7.300000e+01
50%      8.500000e+01
75%      9.800000e+01
max      2.950000e+02
Name: valuenum, dtype: float64
```



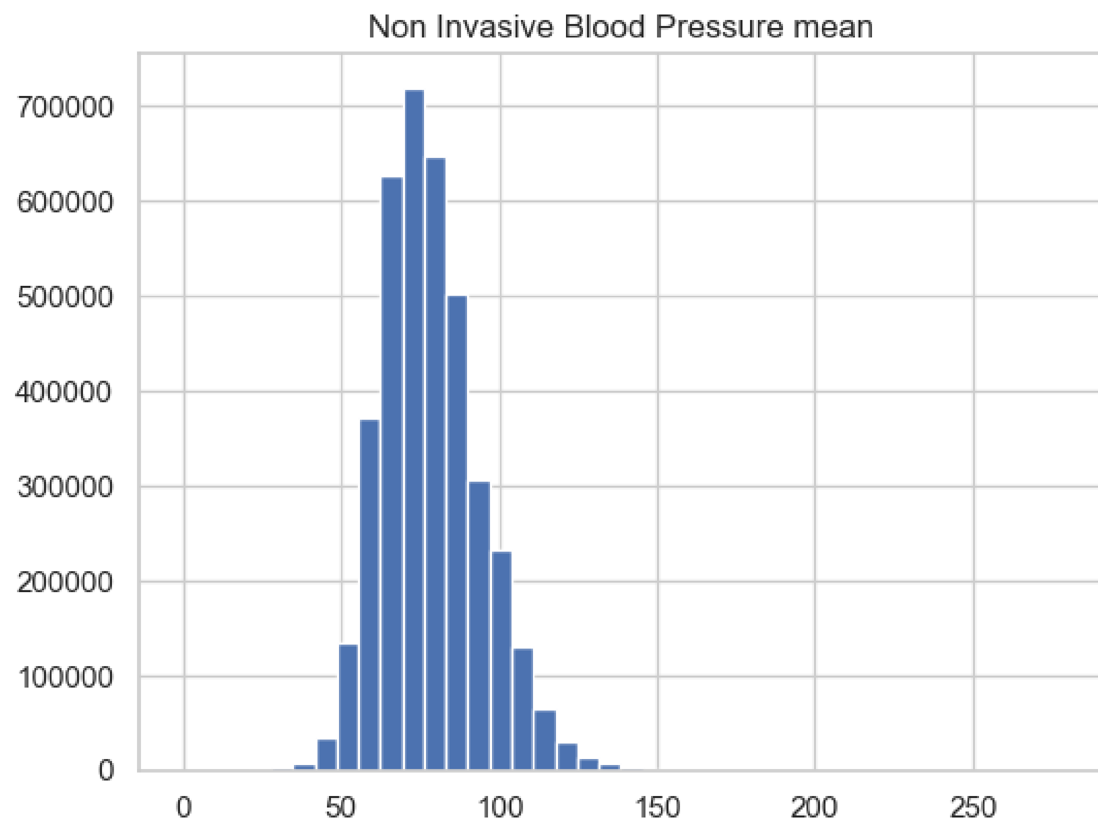

```
count    3.844922e+06
mean     1.196691e+02
std      2.223590e+01
min      0.000000e+00
25%      1.030000e+02
50%      1.170000e+02
75%      1.340000e+02
max      2.800000e+02
Name: valuenum, dtype: float64
```




```
count    3.843863e+06
mean     6.488196e+01
std      1.575905e+01
min      0.000000e+00
25%      5.400000e+01
50%      6.300000e+01
75%      7.400000e+01
max      2.610000e+02
Name: valuenum, dtype: float64
```




```
count    3.842253e+06
mean     7.839710e+01
std      1.589315e+01
min      0.000000e+00
25%      6.700000e+01
50%      7.700000e+01
75%      8.800000e+01
max      2.770000e+02
Name: valuenum, dtype: float64
```




```
# exclude outliers for vitals
chartevents_df
#exclude negative values
chartevents_df = chartevents_df[chartevents_df['valuenum']>=0]
#exclude values above 300
chartevents_df = chartevents_df[chartevents_df['valuenum']<=300]
chartevents_df.to_csv(os.path.join(SAVEDIR, "data",
"chartevents.csv"), index=False)
```

Load labs data ()

- and their definitions (d_items) `labevent_id`: A unique identifier for each laboratory event. `subject_id`: A unique identifier for each patient. `hadm_id`: Hospital admission ID, a unique identifier for each hospital stay. `specimen_id`: A unique identifier for the specimen being tested. `itemid`: A unique identifier for each item or test conducted. `order_provider_id`: The ID of the healthcare provider who ordered the test. `charttime`: The timestamp when the laboratory test result was charted. `storetime`: The timestamp when the laboratory test result was stored in the database. `value`: The result of the laboratory test as a string (e.g., "Positive", "Negative", or a numeric value as a string). `valuenum`: The result of the laboratory test as a number, if applicable. `valueuom`: The unit of measure for the result (e.g., mg/dL, mmol/L, etc.). `ref_range_lower`: The lower limit of the reference range for the test result. `ref_range_upper`: The upper limit of the reference range for the test result. `flag`: Indicates if the result is abnormal or outside the reference range. `priority`: The priority of the laboratory test order (e.g., STAT, routine, etc.). `comments`: Any additional comments or notes related to the laboratory test or result.

```
labevents_dict = {
    "labevent_id": np.int32,
    "subject_id": np.int32,
    # "hadm_id": np.int32,
    "specimen_id": np.int32,
    "itemid": np.int32,
    "charttime": "str",
    "storetime": "str",
    "value": "str",
    "valuenum": np.float64,
    "valueuom": "str",
    "ref_range_lower": np.float64,
    "ref_range_upper": np.float64,
    "flag": "category",  # change to bool
    "priority": "category",  # change to bool
    "comments": "str",
}
if SCRATCH:
    print("Reading in data from scratch")
    labevents_df = pd.read_csv(
        hosp_paths["labevents"],
        dtype=labevents_dict,
        parse_dates=["charttime", "storetime"],
    )
    labevents_df =
labevents_df[labevents_df["subject_id"].isin(subjects)]
    labevents_df = labevents_df[pd.notnull(labevents_df["hadm_id"])]
else:
    labevents_df = pd.read_csv(
        os.path.join(SAVEDIR, "data", "labevents.csv"),
        dtype=labevents_dict,
        parse_dates=["charttime", "storetime"],
    )
labevents_df = dataframe_datetime(labevents_df)
labevents_df.head(1)
```

```
Reading in data from scratch

      labevent_id  subject_id     hadm_id  specimen_id  itemid  \
8988         9004    10001217  24597018.0     69818655   51790

     order_provider_id           charttime           storetime
value  \
8988               NaN 2157-11-19 02:37:00 2157-11-19 03:19:00    59


      valuenum valueuom  ref_range_lower  ref_range_upper flag
priority  \
```

```
8988      59.0    mg/dL              NaN              NaN  NaN
ROUTINE

     comments
8988      NaN
```

```
if not os.path.exists(os.path.join(SAVEDIR, "data", "labevents.csv")):
    labevents_df = labevents_df.to_csv(
        os.path.join(SAVEDIR, "data", "labevents.csv"), index=False
    )
```

Analyze Missingness in labevents_df

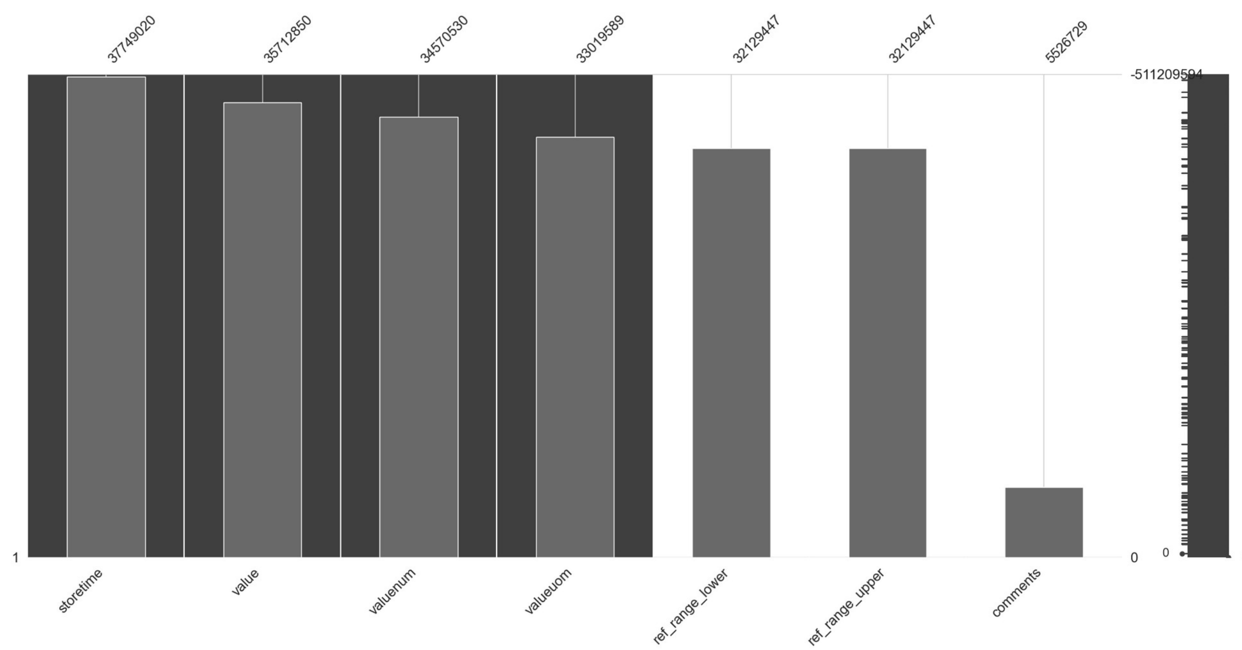
```
labevents_df.head(1)
```

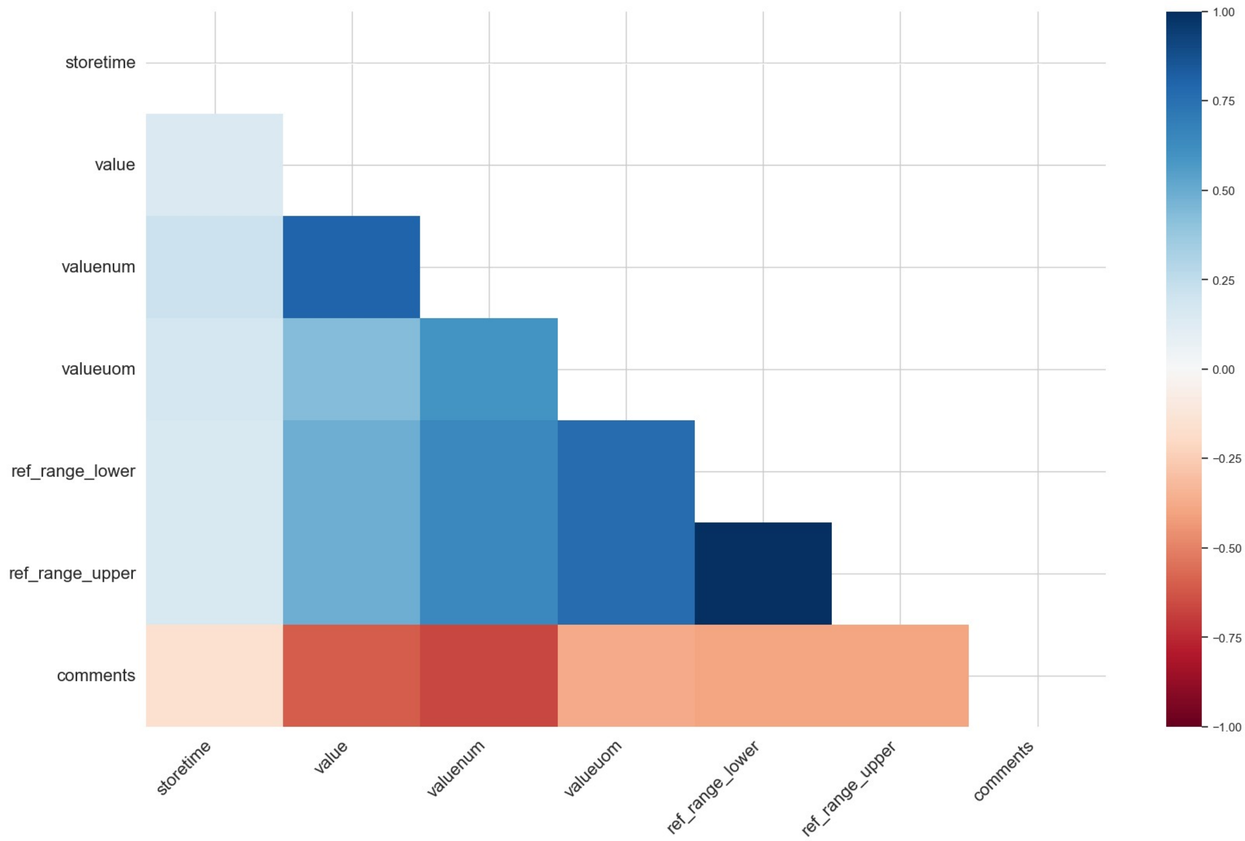
```
      labevent_id  subject_id     hadm_id  specimen_id  itemid  \
8988         9004    10001217  24597018.0     69818655   51790

     order_provider_id           charttime           storetime
value  \
8988               NaN 2157-11-19 02:37:00 2157-11-19 03:19:00    59


      valuenum valueuom  ref_range_lower  ref_range_upper flag
priority  \
8988      59.0    mg/dL              NaN              NaN  NaN
ROUTINE

     comments
8988      NaN
```

```
try:
    labevents_df.drop(columns="order_provider_id", inplace=True)
    labevents_df["abnormal"] = labevents_df["flag"].astype("str")
except:
    print("not in axis")
# rename flag column to abnormal, and convert null values to False,
"abnormal" to True

labevents_df["abnormal"] = labevents_df["abnormal"].fillna(False)
labevents_df["abnormal"] = labevents_df["abnormal"].replace({"nan":
False})
labevents_df["abnormal"] =
labevents_df["abnormal"].replace({"abnormal": True})
labevents_df["abnormal"] = labevents_df["abnormal"].astype("bool")
try:
    labevents_df.drop(columns="flag", inplace=True)
except:
    print("not in axis")
```

```
labevents_df["priority"] = labevents_df["priority"].astype("str")
labevents_df["priority"] = labevents_df["priority"].replace("STAT",
True)
labevents_df["priority"] = labevents_df["priority"].replace("ROUTINE",
False)
#labevents_df["priority"] = labevents_df["priority"].astype("bool")


# replace '--' with NaN
labevents_df = labevents_df.replace("--", np.nan)
try:
    missing = missingness(labevents_df)
    missing = missing.index
except:
    print("")
```

```
storetime          0.004593
value              0.058285
valuenum           0.088407
valueuom           0.129304
ref_range_lower    0.152776
ref_range_upper    0.152776
comments           0.854265
dtype: float64
```






```
# use comments column to fill in missing values
# if the comments column contains a number, use that as the value
import re
labevents_df["comments"] = labevents_df["comments"].astype("str")
def get_number(string):
    """
    If string is a number, or has a number inside of it, return that
number
    """
    #use regex to find number in string
    # the number can include a decimal point

    try:
        return re.search(r"[-+]?\d*\.\d+|\d+", string).group()

    except:
        return np.nan

labevents_df["comments"] = labevents_df["comments"].apply(get_number)
# replace the missing values in valuenum with the comments column
# change the type to float, if it yields a number otherwise it should
return a NaN
labevents_df["valuenum"] =
labevents_df["valuenum"].fillna(labevents_df["comments"])
```

```
labevents_df["valuenum"] = labevents_df["valuenum"].astype("float")


labevents_df.isnull().sum() / labevents_df.shape[0]
```

```
labevent_id          0.000000
subject_id           0.000000
hadm_id              0.000000
specimen_id          0.000000
itemid               0.000000
charttime            0.000000
storetime            0.004593
value                0.058285
valuenum             0.074431
valueuom             0.129304
ref_range_lower      0.152776
ref_range_upper      0.152776
priority             0.000000
comments             0.952932
abnormal             0.000000
dtype: float64
```

```
# drop comments column
labevents_df.drop(columns="comments", inplace=True)
# drop rows with missing values
#labevents_df.dropna(inplace=True)
labevents_df
```

```
           labevent_id  subject_id     hadm_id  specimen_id  itemid  \
8988              9004    10001217  24597018.0     69818655   51790
8989              9005    10001217  24597018.0     69818655   51802
8990              9006    10001217  24597018.0     74137804   52264
8991              9007    10001217  24597018.0     74137804   52272
8992              9008    10001217  24597018.0     74137804   52281
...                ...         ...         ...          ...     ...
118171359    118352498    19999987  23865745.0     85842100   51250
118171360    118352499    19999987  23865745.0     85842100   51265
118171361    118352500    19999987  23865745.0     85842100   51277
118171362    118352501    19999987  23865745.0     85842100   51279
118171363    118352502    19999987  23865745.0     85842100   51301

                    charttime           storetime value  valuenum
valueuom  \
8988      2157-11-19 02:37:00 2157-11-19 03:19:00    59     59.00
mg/dL
8989      2157-11-19 02:37:00 2157-11-19 03:19:00    42     42.00
mg/dL
8990      2157-11-19 02:37:00 2157-11-19 04:55:00   ___    100.00
```

```
%
8991      2157-11-19 02:37:00 2157-11-19 04:55:00     0      0.00
%
8992      2157-11-19 02:37:00 2157-11-19 04:55:00   ___      0.00
%
...                       ...                 ...   ...       ...
...
118171359 2145-11-09 05:30:00 2145-11-09 07:06:00   104    104.00
fL
118171360 2145-11-09 05:30:00 2145-11-09 07:06:00   129    129.00
K/uL
118171361 2145-11-09 05:30:00 2145-11-09 07:06:00  15.4     15.40
%
118171362 2145-11-09 05:30:00 2145-11-09 07:06:00  3.52      3.52
m/uL
118171363 2145-11-09 05:30:00 2145-11-09 07:06:00   5.7      5.70
K/uL

           ref_range_lower  ref_range_upper priority  abnormal
8988                   NaN              NaN    False     False
8989                  15.0             45.0    False     False
8990                   NaN              NaN    False     False
8991                   NaN              NaN    False     False
8992                   NaN              NaN    False     False
...                    ...              ...      ...       ...
118171359             82.0             98.0    False      True
118171360            150.0            440.0    False      True
118171361             10.5             15.5    False     False
118171362              4.2              5.4    False      True
118171363              4.0             11.0    False     False

[37923203 rows x 14 columns]

labevents_df['priority'].value_counts(dropna=False)

priority
True     17538841
False    16034624
nan       4349738
Name: count, dtype: int64

labevents_df.reset_index(drop=True, inplace=True)
# merge labevents_df with labevent_definitions to get the label for
each itemid
labevent_definitions = pd.read_csv(icu_paths["d_items"])
labevents_df = pd.merge(
    labevents_df,
    labevent_definitions[["itemid", "label"]],
    on="itemid",
    how="left",
```

```
)


-----------------------------------------------------------------------
-----
MemoryError                               Traceback (most recent call
last)
c:\Users\david\Desktop\mimic_project\BINF 4008\Assignment 1\
update.ipynb Cell 29 line 3
     <a
href='vscode-notebook-cell:/c%3A/Users/david/Desktop/mimic_project/
BINF%204008/Assignment%201/update.ipynb#X40sZmlsZQ%3D%3D?
line=26'>27</a>     except:
     <a
href='vscode-notebook-cell:/c%3A/Users/david/Desktop/mimic_project/
BINF%204008/Assignment%201/update.ipynb#X40sZmlsZQ%3D%3D?
line=27'>28</a>         return row["valuenum"]
---> <a
href='vscode-notebook-cell:/c%3A/Users/david/Desktop/mimic_project/
BINF%204008/Assignment%201/update.ipynb#X40sZmlsZQ%3D%3D?
line=30'>31</a> labevents_df["z_score"] =
labevents_df.apply(get_z_score, 1)
     <a
href='vscode-notebook-cell:/c%3A/Users/david/Desktop/mimic_project/
BINF%204008/Assignment%201/update.ipynb#X40sZmlsZQ%3D%3D?
line=31'>32</a> labevents_df["outlier"] =
labevents_df["z_score"].apply(lambda x: x > 3)
     <a
href='vscode-notebook-cell:/c%3A/Users/david/Desktop/mimic_project/
BINF%204008/Assignment%201/update.ipynb#X40sZmlsZQ%3D%3D?
line=32'>33</a> labevents_df["outlier"] =
labevents_df["outlier"].astype("bool")

File c:\Programming-Environments\Python3.11.5\Lib\site-packages\
pandas\core\frame.py:10037, in DataFrame.apply(self, func, axis, raw,
result_type, args, by_row, **kwargs)
  10025 from pandas.core.apply import frame_apply
  10027 op = frame_apply(
  10028     self,
  10029     func=func,
   (...)
  10035     kwargs=kwargs,
  10036 )
> 10037 return op.apply().__finalize__(self, method="apply")

File c:\Programming-Environments\Python3.11.5\Lib\site-packages\
pandas\core\apply.py:837, in FrameApply.apply(self)
    834 elif self.raw:
    835     return self.apply_raw()
--> 837 return self.apply_standard()
```

```
File c:\Programming-Environments\Python3.11.5\Lib\site-packages\
pandas\core\apply.py:966, in FrameApply.apply_standard(self)
    963 results, res_index = self.apply_series_generator()
    965 # wrap results
--> 966 return self.wrap_results(results, res_index)

File c:\Programming-Environments\Python3.11.5\Lib\site-packages\
pandas\core\apply.py:1003, in FrameApply.wrap_results(self, results,
res_index)
   1001     result = constructor_sliced(results, dtype=np.float64)
   1002 else:
-> 1003     result = constructor_sliced(results)
   1004 result.index = res_index
   1006 return result

File c:\Programming-Environments\Python3.11.5\Lib\site-packages\
pandas\core\series.py:475, in Series.__init__(self, data, index,
dtype, name, copy, fastpath)
    473         data = data._mgr
    474 elif is_dict_like(data):
--> 475     data, index = self._init_dict(data, index, dtype)
    476     dtype = None
    477     copy = False

File c:\Programming-Environments\Python3.11.5\Lib\site-packages\
pandas\core\series.py:555, in Series._init_dict(self, data, index,
dtype)
    549 if data:
    550     # GH:34717, issue was using zip to extract key and values
from data.
    551     # using generators in effects the performance.
    552     # Below is the new way of extracting the keys and values
    554     keys = tuple(data.keys())
--> 555     values = list(data.values())  # Generating list of values-
faster way
    556 elif index is not None:
    557     # fastpath for Series(data=None). Just use broadcasting a
scalar
    558     # instead of reindexing.
    559     if len(index) or dtype is not None:

MemoryError:
```

```
#Visualize for labevents_df
#make histograms for the
```

Split into 80% train, and 20% test set

```
# split into train and test sets
from sklearn.model_selection import train_test_split

# 80% train, 20% test
# icustays_df = icustays_df.sort_values(by="intime")


# split chronologically, sine we cannot have a patient in the test set
who has an earlier stay number in the train set
train_ids, test_ids = train_test_split(
    icustays_df["subject_id"], test_size=0.2, shuffle=True
)
icustays_df["data_split"] = icustays_df["subject_id"].isin(train_ids)
icustays_df["data_split"] = icustays_df["data_split"].replace(
    {True: "train", False: "test"}
)
icustays_df.to_csv(
    os.path.join(SAVEDIR, "data", "mimic_subject_split.csv"),
index=False
)
print(icustays_df.shape)
icustays_df.head(1)
```

```
(57734, 16)
```

```
   subject_id   hadm_id   stay_id                       first_careunit  
\
0    10001217  24597018  37067082  Surgical Intensive Care Unit (SICU)


                         last_careunit              intime  \
0  Surgical Intensive Care Unit (SICU) 2157-11-20 19:18:02

              outtime       los           admittime deathtime  \
0 2157-11-21 22:08:00  1.118032 2157-11-18 22:56:00       NaT

  48_hour_mortality_flag  age gender   race insurance data_split
0                  False   55      1  WHITE     Other      train
```

```
table_one = icustays_df[['48_hour_mortality_flag','race', 'age',
'insurance', 'gender', 'stay_id', 'hadm_id', 'subject_id']]
table_one
```

```
      48_hour_mortality_flag     race  age insurance gender   stay_id  
\
0                      False    WHITE   55     Other      1  37067082

1                      False    WHITE   46     Other      1  31205490
```

```
2                      False     BLACK   77  Medicare      1  37510196
3                      False     OTHER   57  Medicare      1  39060235
4                      False     WHITE   81     Other      1  33685454
...                      ...       ...  ...       ...    ...       ...
57729                  False     OTHER   42     Other      0  37364566
57730                  False     WHITE   43  Medicaid      0  32336619
57731                  False     WHITE   48     Other      1  36075953
57732                  False     WHITE   58     Other      0  38978960
57733                  False   UNKNOWN   57     Other      1  36195440

        hadm_id  subject_id
0      24597018    10001217
1      25563031    10001725
2      26184834    10001884
3      23581541    10002013
4      23822395    10002155
...         ...         ...
57729  21439025    19999297
57730  26785317    19999442
57731  25744818    19999828
57732  21033226    19999840
57733  23865745    19999987

[57734 rows x 8 columns]
```

```python
def describe_continuous(series):
    mean = series.mean()
    std = series.std()
    return f"{mean:.2f} ({std:.2f})"

def describe_categorical(series):
    counts = series.value_counts()
    total = len(series)
    descriptions = [f"{val} ({count/total*100:.1f}%)" for val, count
in counts.items()]
    return ", ".join(descriptions)

def create_table_1(df, stratify_var):
    # Define variables
    continuous_vars = ['age']
    categorical_vars = ['gender', 'race', 'insurance', 'hadm_id',
'stay_id']
```

```
    # Define stratified columns
    strata = df[stratify_var].unique()
    strata_labels = [f"{stratify_var}: {s}" for s in strata]
    columns = ["Overall"] + strata_labels

    # Initialize Table 1
    table_1 = pd.DataFrame(index=["N"] + continuous_vars +
categorical_vars, columns=columns)

    # Overall statistics
    table_1.loc["N", "Overall"] = len(df)
    for var in continuous_vars:
        table_1.loc[var, "Overall"] = describe_continuous(df[var])
    for var in categorical_vars:
        table_1.loc[var, "Overall"] = describe_categorical(df[var])

    # Stratified statistics
    for stratum, label in zip(strata, strata_labels):
        strata_data = df[df[stratify_var] == stratum]
        table_1.loc["N", label] = len(strata_data)
        for var in continuous_vars:
            table_1.loc[var, label] =
describe_continuous(strata_data[var])
        for var in categorical_vars:
            table_1.loc[var, label] =
describe_categorical(strata_data[var])

    return table_1
#convert boolean gender to Female = 1
#table_one.gender = table_one.gender.map({'F':True, 'M':False})
#display(table_one.gender.value_counts())
create_table_1(table_one, '48_hour_mortality_flag')
```

```
                                                          Overall  \
N                                                           57734
age                                                 65.00 (16.34)
gender                                       0 (56.5%), 1 (43.5%)
race       WHITE (67.9%), UNKNOWN (10.9%), BLACK (10.4%),...
insurance   Other (46.9%), Medicare (45.8%), Medicaid (7.3%)
hadm_id    23344494 (0.0%), 24307798 (0.0%), 26879479 (0....
stay_id    37067082 (0.0%), 39993968 (0.0%), 35755099 (0....

                                48_hour_mortality_flag: False  \
N                                                           56818
age                                                 64.90 (16.34)
gender                                       0 (56.6%), 1 (43.4%)
race       WHITE (68.0%), UNKNOWN (10.7%), BLACK (10.5%),...
insurance   Other (47.1%), Medicare (45.7%), Medicaid (7.3%)
hadm_id    23344494 (0.0%), 26543049 (0.0%), 26879479 (0....
```

```
stay_id    37067082 (0.0%), 34427349 (0.0%), 31445224 (0....

                                48_hour_mortality_flag: True
N                                                        916
age                                            71.19 (15.26)
gender                                 0 (52.0%), 1 (48.0%)
race       WHITE (61.0%), UNKNOWN (21.2%), BLACK (8.4%), ...
insurance   Medicare (55.0%), Other (39.4%), Medicaid (5.6%)
hadm_id    22942076 (0.1%), 29846851 (0.1%), 21548105 (0....
stay_id    34617352 (0.1%), 38649297 (0.1%), 36943198 (0....
```

```
table_one = table_one.groupby(['subject_id',
'48_hour_mortality_flag']).size()
table_one = table_one.reset_index()
#table_one[table_one['48_hour_mortality_flag']==False]
table_one
```

```
       subject_id 48_hour_mortality_flag  0
0        10001217                  False  1
1        10001725                  False  1
2        10001884                  False  1
3        10002013                  False  1
4        10002155                  False  2
...           ...                    ... ..
42497    19999297                  False  1
42498    19999442                  False  1
42499    19999828                  False  1
42500    19999840                  False  1
42501    19999987                  False  1

[42502 rows x 3 columns]
```

```
# TO-DO: MAKE YOUR OWN SPLITS FOR MIMIC_SUBJECT_SPLIT

# Get train subjects and test subjects
split_df_path = os.path.join(".", "data", "mimic_subject_split.csv")
split_df = pd.read_csv(split_df_path)
train_subjects, test_subjects = (
    split_df[split_df["data_split"] == "train"]["subject_id"].values,
    split_df[split_df["data_split"] == "test"]["subject_id"].values,
)
assert set(train_subjects) & set(test_subjects) == set()

train_stay_ids =
icustays_df[icustays_df["subject_id"].isin(train_subjects)][
    "stay_id"
].values

if SCRATCH:
    lab_itemids = itemids_with_minimum_uid_counts(
        labevents_df[labevents_df["subject_id"].isin(train_subjects)],
```

```
        uid_column="subject_id",
        key_column="itemid",
        MIN_UID_THRESHOLD=0.9,
    )
    chartevents_itemids = itemids_with_minimum_uid_counts(

chartevents_df[chartevents_df["stay_id"].isin(train_stay_ids)],
        uid_column="stay_id",
        key_column="itemid",
        MIN_UID_THRESHOLD=0.9,
    )

    labevents_df =
labevents_df[labevents_df["itemid"].isin(lab_itemids)]
    labevents_df.to_csv(os.path.join(SAVEDIR, "data",
"labevents.csv"), index=False)
    chartevents_df =
chartevents_df[chartevents_df["itemid"].isin(chartevents_itemids)]
    chartevents_df.to_csv(os.path.join(SAVEDIR, "data",
"chartevents.csv"), index=False)

icustays_df = icustays_df.drop(columns=["first_careunit",
"last_careunit"])
icustays_df.head(1)

   subject_id   hadm_id   stay_id              intime
outtime  \
0    10001217  24597018  37067082 2157-11-20 19:18:02 2157-11-21
22:08:00

        los           admittime deathtime 48_hour_mortality_flag  age
gender  \
0  1.118032 2157-11-18 22:56:00       NaT                  False   55
1

    race insurance data_split
0  WHITE     Other      train

def get_feature_dict_from_dataframe(df, uid, uid_column, intime,
time_column, key_column, value_column) :

    first_24_hours = (intime, intime + pd.DateOffset(hours = 24))
    current_df = df[df[uid_column] == uid].copy()
    current_df =
current_df[current_df[time_column].between(*first_24_hours)]
    current_df[value_column] = pd.to_numeric(current_df[value_column],
errors = 'coerce')
    current_df = current_df[pd.notnull(current_df[value_column])]
    current_df = current_df[[key_column,
value_column]].groupby([key_column]).agg(np.mean).reset_index()
```

```
#value_column = 'valuenum', key_column = 'itemid'
    return dict(zip(current_df[key_column], current_df[value_column]))

# only feed in the columns that we need for chartevents and labevents
chartevents_min_df = pd.read_csv(
    os.path.join(SAVEDIR, "data", "chartevents.csv"),
    usecols=["stay_id", "charttime", "itemid", "valuenum", 'warning'],
    dtype={
        "stay_id": np.int32,
        "charttime": str,
        "itemid": np.int32,
        "valuenum": np.float64,
    },
    parse_dates=["charttime"],
)
labevents_min_df = pd.read_csv(
    os.path.join(SAVEDIR, "data", "labevents.csv"),
    usecols=["subject_id", "charttime", "itemid", "valuenum",
'priority', 'abnormal'],
    dtype={
        "stay_id": np.int32,
        "charttime": str,
        "itemid": np.int32,
        "valuenum": np.float64,
    },
    parse_dates=["charttime"],
)
icustays = pd.read_csv(
    os.path.join(SAVEDIR, "data", "icustays.csv"),
    usecols=["stay_id", "intime", "subject_id"],
    dtype={"stay_id": np.int32, "intime": str, "subject_id":
np.int32},
    parse_dates=["intime"],
)

labevents_min_df.size
```

```
95518724
```

```
from tqdm import tqdm

icustays_dicts = icustays.to_dict("records")
icustays_dicts = np.array_split(icustays_dicts, 10)

processed_parent_path = os.path.join(".", "data", "mimic_processed2")
if not os.path.isdir(processed_parent_path):
    os.mkdir(processed_parent_path)

if len(os.listdir(processed_parent_path)) == 0:
    for i, icu_stay_dict_array in enumerate(icustays_dicts):
```

```
        data = {}

        for _, icustay_dict in tqdm(enumerate(icu_stay_dict_array)):

            vital_features = get_feature_dict_from_dataframe(
                df=chartevents_min_df,
                uid=icustay_dict["stay_id"],
                uid_column="stay_id",
                intime=icustay_dict["intime"],
                time_column="charttime",
                key_column="itemid",
                value_column="valuenum",
            )
            lab_features = get_feature_dict_from_dataframe(
                df=labevents_min_df,
                uid=icustay_dict["subject_id"],
                uid_column="subject_id",
                intime=icustay_dict["intime"],
                time_column="charttime",
                key_column="itemid",
                value_column="valuenum",
            )
            lab_priority = get_feature_dict_from_dataframe(
                df=labevents_min_df,
                uid=icustay_dict["subject_id"],
                uid_column="subject_id",
                intime=icustay_dict["intime"],
                time_column="charttime",
                key_column="itemid",
                value_column="priority",
            )
            vital_warning = get_feature_dict_from_dataframe(
                df=chartevents_min_df,
                uid=icustay_dict["stay_id"],
                uid_column="stay_id",
                intime=icustay_dict["intime"],
                time_column="charttime",
                key_column="itemid",
                value_column="warning",
            )
            lab_abnormal = get_feature_dict_from_dataframe(
                df=labevents_min_df,
                uid=icustay_dict["subject_id"],
                uid_column="subject_id",
                intime=icustay_dict["intime"],
                time_column="charttime",
                key_column="itemid",
                value_column="abnormal",
            )
```

```
            # print(vital_features)
            vital_features = {f"vital_{k}": v for k, v in
vital_features.items()}
            lab_features = {f"lab_{k}": v for k, v in
lab_features.items()}
            lab_priority = {f"lab_priority_{k}": v for k, v in
lab_priority.items()}
            vital_warning = {f"vital_warning_{k}": v for k, v in
vital_warning.items()}
            lab_abnormal = {f"lab_abnormal_{k}": v for k, v in
lab_abnormal.items()}
            data[icustay_dict["stay_id"]] = {
                **icustay_dict,
                **vital_features,
                **lab_features,
                **lab_priority,
                **vital_warning,
                **lab_abnormal,
            }

        with open(
            os.path.join(processed_parent_path,
f"processed_mimic_{i}.pickle"), "wb"
        ) as handle:
            pickle.dump(data, handle,
protocol=pickle.HIGHEST_PROTOCOL)

        # data = pd.DataFrame(data)
        # data.to_csv(os.path.join(processed_parent_path,
f'processed_mimic_{i}.csv'), index = False)
        print(f"{i}: Dict with {len(data)} stay_ids saved.")


data = {}

for pickle_path in get_files_of_type(processed_parent_path,
filetype="pickle"):
    with open(pickle_path, "rb") as handle:
        current_data = pickle.load(handle)

    data = {**data, **current_data}

len(data)
```

```
5774it [09:29, 10.14it/s]

0: Dict with 5774 stay_ids saved.

5774it [09:28, 10.16it/s]

1: Dict with 5774 stay_ids saved.
```

```
5774it [09:28, 10.16it/s]

2: Dict with 5774 stay_ids saved.

5774it [09:28, 10.16it/s]

3: Dict with 5774 stay_ids saved.

5773it [10:29,  9.17it/s]

4: Dict with 5773 stay_ids saved.

5773it [11:05,  8.68it/s]

5: Dict with 5773 stay_ids saved.

5773it [10:17,  9.35it/s]

6: Dict with 5773 stay_ids saved.

5773it [10:02,  9.59it/s]

7: Dict with 5773 stay_ids saved.

5773it [09:29, 10.14it/s]

8: Dict with 5773 stay_ids saved.

5773it [09:29, 10.14it/s]

9: Dict with 5773 stay_ids saved.

57734
```

```
data = {}

for pickle_path in get_files_of_type(processed_parent_path,
filetype="pickle"):
    with open(pickle_path, "rb") as handle:
        current_data = pickle.load(handle)

    data = {**data, **current_data}

print(len(data))
```

```
57734
```

```
data = pd.DataFrame(data).T
print(data.shape)
data.reset_index(inplace=True, drop=True)
display(data.head(1))
```

```
(57734, 82)
```

```
  subject_id   stay_id               intime vital_220045 vital_220179
\
0   10001217  37067082  2157-11-20 19:18:02         93.2       135.32


  vital_220180 vital_220181 vital_223761 lab_50868 lab_50882  ...  \
0        80.96    93.041667      99.0625      15.0      23.0  ...

  lab_abnormal_51237 lab_abnormal_51248 lab_abnormal_51249
lab_abnormal_51250  \
0                1.0                0.0                0.0
0.0

  lab_abnormal_51265 lab_abnormal_51274 lab_abnormal_51275
lab_abnormal_51277  \
0                0.0                1.0                0.0
0.0

  lab_abnormal_51279 lab_abnormal_51301
0                1.0                1.0

[1 rows x 82 columns]
```

```
data = data.merge(icustays_df, on=['subject_id',"stay_id"],
how="left")

data.head(1)
```

```
  subject_id   stay_id             intime_x vital_220045 vital_220179
\
0   10001217  37067082  2157-11-20 19:18:02         93.2       135.32


  vital_220180 vital_220181 vital_223761 lab_50868 lab_50882  ...  \
0        80.96    93.041667      99.0625      15.0      23.0  ...

              outtime       los           admittime deathtime  \
0 2157-11-21 22:08:00  1.118032 2157-11-18 22:56:00       NaT

  48_hour_mortality_flag age gender   race insurance data_split
0                  False  55      1  WHITE     Other      train

[1 rows x 94 columns]
```

```
data.columns
```

```
Index(['subject_id', 'stay_id', 'intime_x', 'vital_220045',
'vital_220179',
       'vital_220180', 'vital_220181', 'vital_223761', 'lab_50868',
       'lab_50882', 'lab_50893', 'lab_50902', 'lab_50912',
'lab_50931',
```

```
        'lab_50960', 'lab_50970', 'lab_50971', 'lab_50983',
'lab_51006',
        'lab_51221', 'lab_51222', 'lab_51237', 'lab_51248',
'lab_51249',
        'lab_51250', 'lab_51265', 'lab_51274', 'lab_51275',
'lab_51277',
        'lab_51279', 'lab_51301', 'lab_priority_50868',
'lab_priority_50882',
        'lab_priority_50893', 'lab_priority_50902',
'lab_priority_50912',
        'lab_priority_50931', 'lab_priority_50960',
'lab_priority_50970',
        'lab_priority_50971', 'lab_priority_50983',
'lab_priority_51006',
        'lab_priority_51221', 'lab_priority_51222',
'lab_priority_51237',
        'lab_priority_51248', 'lab_priority_51249',
'lab_priority_51250',
        'lab_priority_51265', 'lab_priority_51274',
'lab_priority_51275',
        'lab_priority_51277', 'lab_priority_51279',
'lab_priority_51301',
        'vital_warning_220045', 'vital_warning_220179',
'vital_warning_220180',
        'vital_warning_220181', 'vital_warning_223761',
'lab_abnormal_50868',
        'lab_abnormal_50882', 'lab_abnormal_50893',
'lab_abnormal_50902',
        'lab_abnormal_50912', 'lab_abnormal_50931',
'lab_abnormal_50960',
        'lab_abnormal_50970', 'lab_abnormal_50971',
'lab_abnormal_50983',
        'lab_abnormal_51006', 'lab_abnormal_51221',
'lab_abnormal_51222',
        'lab_abnormal_51237', 'lab_abnormal_51248',
'lab_abnormal_51249',
        'lab_abnormal_51250', 'lab_abnormal_51265',
'lab_abnormal_51274',
        'lab_abnormal_51275', 'lab_abnormal_51277',
'lab_abnormal_51279',
        'lab_abnormal_51301', 'hadm_id', 'intime_y', 'outtime', 'los',
        'admittime', 'deathtime', '48_hour_mortality_flag', 'age',
'gender',
        'race', 'insurance', 'data_split'],
       dtype='object')
```

```
data = data.drop(columns=["intime_x", 'outtime', 'admittime',
'intime_y', 'deathtime', 'admittime', 'los'])
data = pd.get_dummies(data,columns=['race', 'insurance'])
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 57734 entries, 0 to 57733
Data columns (total 95 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   subject_id              57734 non-null  object
 1   stay_id                 57734 non-null  object
 2   vital_220045            57664 non-null  object
 3   vital_220179            52084 non-null  object
 4   vital_220180            52077 non-null  object
 5   vital_220181            52084 non-null  object
 6   vital_223761            53568 non-null  object
 7   lab_50868               56832 non-null  object
 8   lab_50882               56901 non-null  object
 9   lab_50893               52254 non-null  object
 10  lab_50902               56929 non-null  object
 11  lab_50912               56920 non-null  object
 12  lab_50931               56658 non-null  object
 13  lab_50960               54840 non-null  object
 14  lab_50970               52417 non-null  object
 15  lab_50971               56886 non-null  object
 16  lab_50983               56925 non-null  object
 17  lab_51006               56912 non-null  object
 18  lab_51221               56738 non-null  object
 19  lab_51222               56630 non-null  object
 20  lab_51237               48375 non-null  object
 21  lab_51248               56614 non-null  object
 22  lab_51249               56617 non-null  object
 23  lab_51250               56621 non-null  object
 24  lab_51265               56646 non-null  object
 25  lab_51274               48372 non-null  object
 26  lab_51275               47982 non-null  object
 27  lab_51277               56597 non-null  object
 28  lab_51279               56622 non-null  object
 29  lab_51301               56645 non-null  object
 30  lab_priority_50868      56832 non-null  object
 31  lab_priority_50882      56902 non-null  object
 32  lab_priority_50893      52256 non-null  object
 33  lab_priority_50902      56932 non-null  object
 34  lab_priority_50912      56920 non-null  object
 35  lab_priority_50931      56658 non-null  object
 36  lab_priority_50960      54841 non-null  object
 37  lab_priority_50970      52417 non-null  object
 38  lab_priority_50971      56888 non-null  object
 39  lab_priority_50983      56926 non-null  object
 40  lab_priority_51006      56914 non-null  object
 41  lab_priority_51221      56747 non-null  object
 42  lab_priority_51222      56649 non-null  object
```

```
 43  lab_priority_51237      48504 non-null  object
 44  lab_priority_51248      56639 non-null  object
 45  lab_priority_51249      56640 non-null  object
 46  lab_priority_51250      56639 non-null  object
 47  lab_priority_51265      56667 non-null  object
 48  lab_priority_51274      48504 non-null  object
 49  lab_priority_51275      48139 non-null  object
 50  lab_priority_51277      56638 non-null  object
 51  lab_priority_51279      56639 non-null  object
 52  lab_priority_51301      56656 non-null  object
 53  vital_warning_220045    57664 non-null  object
 54  vital_warning_220179    52084 non-null  object
 55  vital_warning_220180    52077 non-null  object
 56  vital_warning_220181    52084 non-null  object
 57  vital_warning_223761    53568 non-null  object
 58  lab_abnormal_50868      56832 non-null  object
 59  lab_abnormal_50882      56902 non-null  object
 60  lab_abnormal_50893      52256 non-null  object
 61  lab_abnormal_50902      56932 non-null  object
 62  lab_abnormal_50912      56920 non-null  object
 63  lab_abnormal_50931      56658 non-null  object
 64  lab_abnormal_50960      54841 non-null  object
 65  lab_abnormal_50970      52417 non-null  object
 66  lab_abnormal_50971      56888 non-null  object
 67  lab_abnormal_50983      56926 non-null  object
 68  lab_abnormal_51006      56914 non-null  object
 69  lab_abnormal_51221      56747 non-null  object
 70  lab_abnormal_51222      56649 non-null  object
 71  lab_abnormal_51237      48504 non-null  object
 72  lab_abnormal_51248      56639 non-null  object
 73  lab_abnormal_51249      56640 non-null  object
 74  lab_abnormal_51250      56639 non-null  object
 75  lab_abnormal_51265      56667 non-null  object
 76  lab_abnormal_51274      48504 non-null  object
 77  lab_abnormal_51275      48139 non-null  object
 78  lab_abnormal_51277      56638 non-null  object
 79  lab_abnormal_51279      56639 non-null  object
 80  lab_abnormal_51301      56656 non-null  object
 81  hadm_id                 57734 non-null  int64
 82  48_hour_mortality_flag  57734 non-null  object
 83  age                     57734 non-null  int32
 84  gender                  57734 non-null  object
 85  data_split              57734 non-null  object
 86  race_ASIAN              57734 non-null  bool
 87  race_BLACK              57734 non-null  bool
 88  race_HISPANIC           57734 non-null  bool
 89  race_OTHER              57734 non-null  bool
 90  race_UNKNOWN            57734 non-null  bool
 91  race_WHITE              57734 non-null  bool
```

```
 92  insurance_Medicaid      57734 non-null  bool
 93  insurance_Medicare      57734 non-null  bool
 94  insurance_Other         57734 non-null  bool
dtypes: bool(9), int32(1), int64(1), object(84)
memory usage: 38.2+ MB
```

```
data['48_hour_mortality_flag'] = data['48_hour_mortality_flag'] ==
'True'
data['48_hour_mortality_flag'].value_counts()
```

```
48_hour_mortality_flag
False    56818
True       916
Name: count, dtype: int64
```

```
#if column is bool, change to int
for col in data.columns:
    if data[col].dtype == bool:
        data[col] = data[col].astype(np.int32)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 57734 entries, 0 to 57733
Data columns (total 95 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   subject_id              57734 non-null  object
 1   stay_id                 57734 non-null  object
 2   vital_220045            57664 non-null  object
 3   vital_220179            52084 non-null  object
 4   vital_220180            52077 non-null  object
 5   vital_220181            52084 non-null  object
 6   vital_223761            53568 non-null  object
 7   lab_50868               56832 non-null  object
 8   lab_50882               56901 non-null  object
 9   lab_50893               52254 non-null  object
 10  lab_50902               56929 non-null  object
 11  lab_50912               56920 non-null  object
 12  lab_50931               56658 non-null  object
 13  lab_50960               54840 non-null  object
 14  lab_50970               52417 non-null  object
 15  lab_50971               56886 non-null  object
 16  lab_50983               56925 non-null  object
 17  lab_51006               56912 non-null  object
 18  lab_51221               56738 non-null  object
 19  lab_51222               56630 non-null  object
 20  lab_51237               48375 non-null  object
 21  lab_51248               56614 non-null  object
 22  lab_51249               56617 non-null  object
 23  lab_51250               56621 non-null  object
```

```
 24  lab_51265              56646 non-null  object
 25  lab_51274              48372 non-null  object
 26  lab_51275              47982 non-null  object
 27  lab_51277              56597 non-null  object
 28  lab_51279              56622 non-null  object
 29  lab_51301              56645 non-null  object
 30  lab_priority_50868     56832 non-null  object
 31  lab_priority_50882     56902 non-null  object
 32  lab_priority_50893     52256 non-null  object
 33  lab_priority_50902     56932 non-null  object
 34  lab_priority_50912     56920 non-null  object
 35  lab_priority_50931     56658 non-null  object
 36  lab_priority_50960     54841 non-null  object
 37  lab_priority_50970     52417 non-null  object
 38  lab_priority_50971     56888 non-null  object
 39  lab_priority_50983     56926 non-null  object
 40  lab_priority_51006     56914 non-null  object
 41  lab_priority_51221     56747 non-null  object
 42  lab_priority_51222     56649 non-null  object
 43  lab_priority_51237     48504 non-null  object
 44  lab_priority_51248     56639 non-null  object
 45  lab_priority_51249     56640 non-null  object
 46  lab_priority_51250     56639 non-null  object
 47  lab_priority_51265     56667 non-null  object
 48  lab_priority_51274     48504 non-null  object
 49  lab_priority_51275     48139 non-null  object
 50  lab_priority_51277     56638 non-null  object
 51  lab_priority_51279     56639 non-null  object
 52  lab_priority_51301     56656 non-null  object
 53  vital_warning_220045   57664 non-null  object
 54  vital_warning_220179   52084 non-null  object
 55  vital_warning_220180   52077 non-null  object
 56  vital_warning_220181   52084 non-null  object
 57  vital_warning_223761   53568 non-null  object
 58  lab_abnormal_50868     56832 non-null  object
 59  lab_abnormal_50882     56902 non-null  object
 60  lab_abnormal_50893     52256 non-null  object
 61  lab_abnormal_50902     56932 non-null  object
 62  lab_abnormal_50912     56920 non-null  object
 63  lab_abnormal_50931     56658 non-null  object
 64  lab_abnormal_50960     54841 non-null  object
 65  lab_abnormal_50970     52417 non-null  object
 66  lab_abnormal_50971     56888 non-null  object
 67  lab_abnormal_50983     56926 non-null  object
 68  lab_abnormal_51006     56914 non-null  object
 69  lab_abnormal_51221     56747 non-null  object
 70  lab_abnormal_51222     56649 non-null  object
 71  lab_abnormal_51237     48504 non-null  object
 72  lab_abnormal_51248     56639 non-null  object
```

```
 73  lab_abnormal_51249      56640 non-null  object
 74  lab_abnormal_51250      56639 non-null  object
 75  lab_abnormal_51265      56667 non-null  object
 76  lab_abnormal_51274      48504 non-null  object
 77  lab_abnormal_51275      48139 non-null  object
 78  lab_abnormal_51277      56638 non-null  object
 79  lab_abnormal_51279      56639 non-null  object
 80  lab_abnormal_51301      56656 non-null  object
 81  hadm_id                 57734 non-null  int64
 82  48_hour_mortality_flag  57734 non-null  int32
 83  age                     57734 non-null  int32
 84  gender                  57734 non-null  object
 85  data_split              57734 non-null  object
 86  race_ASIAN              57734 non-null  int32
 87  race_BLACK              57734 non-null  int32
 88  race_HISPANIC           57734 non-null  int32
 89  race_OTHER              57734 non-null  int32
 90  race_UNKNOWN            57734 non-null  int32
 91  race_WHITE              57734 non-null  int32
 92  insurance_Medicaid      57734 non-null  int32
 93  insurance_Medicare      57734 non-null  int32
 94  insurance_Other         57734 non-null  int32
dtypes: int32(11), int64(1), object(83)
memory usage: 39.4+ MB

data['train'] = data['data_split'] == 'train'

data['gender'] = data['gender'] == '1'

data.to_csv(os.path.join(SAVEDIR, "data", "mimic_subject_split.csv"),
index=False)
```

Train XGBoost Model

```
from sklearn.metrics import (
    accuracy_score,
    roc_auc_score,
    average_precision_score,
    precision_recall_curve,
    confusion_matrix,
    ConfusionMatrixDisplay,
    auc,
    KFold
)
```

```python
from sklearn.preprocessing import StandardScaler
from xgboost import XGBClassifier

split_df_path = os.path.join(SAVEDIR, "data",
"mimic_subject_split.csv")
split_df = pd.read_csv(split_df_path)
train_subjects, test_subjects = (
    split_df[split_df["data_split"] == "train"]["subject_id"].values,
    split_df[split_df["data_split"] == "test"]["subject_id"].values,
)
assert set(train_subjects) & set(test_subjects) == set()



feature_columns = [
    c for c in data.columns if c not in (set(icustays_df.columns) -
{'age', 'gender', 'race'})
]
#feature_columns.remove('data_split')

print(feature_columns)
X_train, y_train = data[data["subject_id"].isin(train_subjects)][
    feature_columns
], data[data["subject_id"].isin(train_subjects)][
    "48_hour_mortality_flag"
].values.astype(
    int
)
X_test, y_test = data[data["subject_id"].isin(test_subjects)][
    feature_columns
].values, data[data["subject_id"].isin(test_subjects)][
    "48_hour_mortality_flag"
].values.astype(
    int
)
# find strings in X_train
for i, col in enumerate(feature_columns):
    if isinstance(X_train.loc[0][i], str):
        print(col)
        print(X_train.loc[0][i])
print(X_train.columns)

MISSINGNESS_THRESHOLD = 0.2
passes_missingness_threhsold = (
    data.isnull().sum() / data.shape[0] <= MISSINGNESS_THRESHOLD
)
feature_columns = list(

set(passes_missingness_threhsold[passes_missingness_threhsold].index.v
alues)
```

```
    - {"48_hour_mortality_flag", "subject_id", "hadm_id", "stay_id"}
)
feature_columns.remove('data_split')
print(feature_columns)
X_train, y_train = data[data["subject_id"].isin(train_subjects)][
    feature_columns
].values, data[data["subject_id"].isin(train_subjects)][
    "48_hour_mortality_flag"
].values.astype(
    int
)
X_test, y_test = data[data["subject_id"].isin(test_subjects)][
    feature_columns
].values, data[data["subject_id"].isin(test_subjects)][
    "48_hour_mortality_flag"
].values.astype(
    int
)


def prc_auc(y_true, y_pred):
    precision, recall, _ = precision_recall_curve(y_true, y_pred)
    return auc(recall, precision)


scaler = StandardScaler()
scaler.fit(X_train)

X_train, X_test = scaler.transform(X_train), scaler.transform(X_test)
print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
display(X_train)
xgb = XGBClassifier(
    tree_method="hist",
    early_stopping_rounds=100,
    scale_pos_weight=sum(y_train == 0) / sum(y_train),
    use_label_encoder=False,
    learning_rate=0.01,
    n_estimators=1000,
    device="cuda",
    max_depth=30,
    objective="binary:logistic",
    nthread=4,
    #eval_metric=prc_auc,
)
xgb.fit(X_train, y_train, eval_set=[(X_test, y_test)], verbose=True)
y_preds = xgb.predict(X_test)
y_pred_probs = xgb.predict_proba(X_test)


print(accuracy_score(y_preds, y_test))
```

```
print(roc_auc_score(y_test, y_pred_probs[:, 1]))
print(average_precision_score(y_test, y_pred_probs[:, 1]))

ConfusionMatrixDisplay(confusion_matrix(y_preds, y_test)).plot()
```

```
['vital_220045', 'vital_220179', 'vital_220180', 'vital_220181',
'vital_223761', 'lab_50868', 'lab_50882', 'lab_50893', 'lab_50902',
'lab_50912', 'lab_50931', 'lab_50960', 'lab_50970', 'lab_50971',
'lab_50983', 'lab_51006', 'lab_51221', 'lab_51222', 'lab_51237',
'lab_51248', 'lab_51249', 'lab_51250', 'lab_51265', 'lab_51274',
'lab_51275', 'lab_51277', 'lab_51279', 'lab_51301',
'lab_priority_50868', 'lab_priority_50882', 'lab_priority_50893',
'lab_priority_50902', 'lab_priority_50912', 'lab_priority_50931',
'lab_priority_50960', 'lab_priority_50970', 'lab_priority_50971',
'lab_priority_50983', 'lab_priority_51006', 'lab_priority_51221',
'lab_priority_51222', 'lab_priority_51237', 'lab_priority_51248',
'lab_priority_51249', 'lab_priority_51250', 'lab_priority_51265',
'lab_priority_51274', 'lab_priority_51275', 'lab_priority_51277',
'lab_priority_51279', 'lab_priority_51301', 'vital_warning_220045',
'vital_warning_220179', 'vital_warning_220180',
'vital_warning_220181', 'vital_warning_223761', 'lab_abnormal_50868',
'lab_abnormal_50882', 'lab_abnormal_50893', 'lab_abnormal_50902',
'lab_abnormal_50912', 'lab_abnormal_50931', 'lab_abnormal_50960',
'lab_abnormal_50970', 'lab_abnormal_50971', 'lab_abnormal_50983',
'lab_abnormal_51006', 'lab_abnormal_51221', 'lab_abnormal_51222',
'lab_abnormal_51237', 'lab_abnormal_51248', 'lab_abnormal_51249',
'lab_abnormal_51250', 'lab_abnormal_51265', 'lab_abnormal_51274',
'lab_abnormal_51275', 'lab_abnormal_51277', 'lab_abnormal_51279',
'lab_abnormal_51301', 'age', 'gender', 'race_ASIAN', 'race_BLACK',
'race_HISPANIC', 'race_OTHER', 'race_UNKNOWN', 'race_WHITE',
'insurance_Medicaid', 'insurance_Medicare', 'insurance_Other']
Index(['vital_220045', 'vital_220179', 'vital_220180', 'vital_220181',
       'vital_223761', 'lab_50868', 'lab_50882', 'lab_50893',
'lab_50902',
       'lab_50912', 'lab_50931', 'lab_50960', 'lab_50970',
'lab_50971',
       'lab_50983', 'lab_51006', 'lab_51221', 'lab_51222',
'lab_51237',
       'lab_51248', 'lab_51249', 'lab_51250', 'lab_51265',
'lab_51274',
       'lab_51275', 'lab_51277', 'lab_51279', 'lab_51301',
       'lab_priority_50868', 'lab_priority_50882',
'lab_priority_50893',
       'lab_priority_50902', 'lab_priority_50912',
'lab_priority_50931',
       'lab_priority_50960', 'lab_priority_50970',
'lab_priority_50971',
       'lab_priority_50983', 'lab_priority_51006',
'lab_priority_51221',
       'lab_priority_51222', 'lab_priority_51237',
```

```
'lab_priority_51248',
       'lab_priority_51249', 'lab_priority_51250',
'lab_priority_51265',
       'lab_priority_51274', 'lab_priority_51275',
'lab_priority_51277',
       'lab_priority_51279', 'lab_priority_51301',
'vital_warning_220045',
       'vital_warning_220179', 'vital_warning_220180',
'vital_warning_220181',
       'vital_warning_223761', 'lab_abnormal_50868',
'lab_abnormal_50882',
       'lab_abnormal_50893', 'lab_abnormal_50902',
'lab_abnormal_50912',
       'lab_abnormal_50931', 'lab_abnormal_50960',
'lab_abnormal_50970',
       'lab_abnormal_50971', 'lab_abnormal_50983',
'lab_abnormal_51006',
       'lab_abnormal_51221', 'lab_abnormal_51222',
'lab_abnormal_51237',
       'lab_abnormal_51248', 'lab_abnormal_51249',
'lab_abnormal_51250',
       'lab_abnormal_51265', 'lab_abnormal_51274',
'lab_abnormal_51275',
       'lab_abnormal_51277', 'lab_abnormal_51279',
'lab_abnormal_51301', 'age',
       'gender', 'race_ASIAN', 'race_BLACK', 'race_HISPANIC',
'race_OTHER',
       'race_UNKNOWN', 'race_WHITE', 'insurance_Medicaid',
       'insurance_Medicare', 'insurance_Other'],
      dtype='object')
['lab_priority_51265', 'vital_warning_223761', 'lab_abnormal_50868',
'insurance_Other', 'lab_abnormal_50971', 'lab_50983',
'lab_abnormal_51275', 'lab_priority_51275', 'lab_priority_50902',
'lab_abnormal_51248', 'lab_50912', 'vital_warning_220180',
'lab_51248', 'lab_priority_50970', 'lab_priority_51279',
'lab_priority_51237', 'lab_abnormal_51221', 'lab_50902',
'insurance_Medicare', 'lab_priority_51249', 'lab_abnormal_51279',
'lab_priority_50931', 'lab_50931', 'lab_51265', 'lab_51277',
'lab_50970', 'lab_abnormal_51222', 'lab_51221', 'lab_priority_51222',
'lab_50893', 'gender', 'lab_abnormal_51006', 'race_BLACK',
'insurance_Medicaid', 'lab_priority_51250', 'lab_51274',
'lab_abnormal_50960', 'vital_223761', 'lab_priority_51006',
'lab_abnormal_51249', 'lab_abnormal_50902', 'lab_priority_50960',
'lab_abnormal_51237', 'lab_50882', 'lab_priority_51221',
'lab_abnormal_50983', 'lab_priority_51248', 'lab_51275',
'vital_warning_220179', 'lab_priority_50893', 'lab_51279',
'lab_51249', 'lab_abnormal_51265', 'lab_priority_51274',
'vital_220180', 'lab_priority_50983', 'lab_51006', 'lab_51222',
'lab_51237', 'race_WHITE', 'lab_priority_50971', 'lab_abnormal_51277',
```

```
'lab_51250', 'lab_priority_51277', 'lab_abnormal_50970', 'lab_50868',
'vital_220181', 'lab_50960', 'lab_abnormal_50882',
'lab_priority_51301', 'lab_abnormal_50931', 'lab_priority_50882',
'lab_abnormal_50912', 'lab_priority_50868', 'lab_abnormal_51250',
'lab_51301', 'vital_220179', 'vital_warning_220045', 'lab_50971',
'lab_abnormal_51301', 'vital_warning_220181', 'lab_abnormal_50893',
'lab_priority_50912', 'age', 'vital_220045', 'lab_abnormal_51274',
'race_ASIAN', 'race_UNKNOWN', 'race_HISPANIC', 'race_OTHER']
(50626, 90) (50626,) (7108, 90) (7108,)

array([[ 0.93709829, -0.21062632, -0.41861132, ..., -0.34003848,
         -0.19687319, -0.20931682],
       [-0.31269483, -0.21062632, -0.41861132, ..., -0.34003848,
         -0.19687319, -0.20931682],
       [ 0.93709829,         nan, -0.41861132, ..., -0.34003848,
         -0.19687319, -0.20931682],
       ...,
       [ 0.93709829, -0.21062632, -0.41861132, ..., -0.34003848,
         -0.19687319, -0.20931682],
       [-0.31269483, -0.21062632, -0.41861132, ..., -0.34003848,
         -0.19687319, -0.20931682],
       [-1.56248795, -0.21062632, -0.41861132, ...,  2.9408436 ,
         -0.19687319, -0.20931682]])

[0]	validation_0-logloss:0.68426
[1]	validation_0-logloss:0.67555
[2]	validation_0-logloss:0.66701
[3]	validation_0-logloss:0.65865
[4]	validation_0-logloss:0.65044
[5]	validation_0-logloss:0.64238
[6]	validation_0-logloss:0.63448
[7]	validation_0-logloss:0.62671
[8]	validation_0-logloss:0.61908
[9]	validation_0-logloss:0.61159
[10]	validation_0-logloss:0.60427
[11]	validation_0-logloss:0.59707
[12]	validation_0-logloss:0.58998
[13]	validation_0-logloss:0.58301
[14]	validation_0-logloss:0.57616
[15]	validation_0-logloss:0.56944
[16]	validation_0-logloss:0.56282
[17]	validation_0-logloss:0.55636
[18]	validation_0-logloss:0.54997
[19]	validation_0-logloss:0.54368
[20]	validation_0-logloss:0.53750
[21]	validation_0-logloss:0.53143
[22]	validation_0-logloss:0.52541
[23]	validation_0-logloss:0.51953
[24]	validation_0-logloss:0.51375
[25]	validation_0-logloss:0.50806
```

```
[26]	validation_0-logloss:0.50246
[27]	validation_0-logloss:0.49693
[28]	validation_0-logloss:0.49153
[29]	validation_0-logloss:0.48618
[30]	validation_0-logloss:0.48092
[31]	validation_0-logloss:0.47569
[32]	validation_0-logloss:0.47058
[33]	validation_0-logloss:0.46557
[34]	validation_0-logloss:0.46062
[35]	validation_0-logloss:0.45576
[36]	validation_0-logloss:0.45093
[37]	validation_0-logloss:0.44620
[38]	validation_0-logloss:0.44155
[39]	validation_0-logloss:0.43696
[40]	validation_0-logloss:0.43246
[41]	validation_0-logloss:0.42803
[42]	validation_0-logloss:0.42366
[43]	validation_0-logloss:0.41938
[44]	validation_0-logloss:0.41513
[45]	validation_0-logloss:0.41095
[46]	validation_0-logloss:0.40683
[47]	validation_0-logloss:0.40277
[48]	validation_0-logloss:0.39876
[49]	validation_0-logloss:0.39482
[50]	validation_0-logloss:0.39092
[51]	validation_0-logloss:0.38712
[52]	validation_0-logloss:0.38333
[53]	validation_0-logloss:0.37962
[54]	validation_0-logloss:0.37591
[55]	validation_0-logloss:0.37226
[56]	validation_0-logloss:0.36871
[57]	validation_0-logloss:0.36517
[58]	validation_0-logloss:0.36167
[59]	validation_0-logloss:0.35824
[60]	validation_0-logloss:0.35483
[61]	validation_0-logloss:0.35152
[62]	validation_0-logloss:0.34823
[63]	validation_0-logloss:0.34500
[64]	validation_0-logloss:0.34179
[65]	validation_0-logloss:0.33862
[66]	validation_0-logloss:0.33553
[67]	validation_0-logloss:0.33245
[68]	validation_0-logloss:0.32943
[69]	validation_0-logloss:0.32642
[70]	validation_0-logloss:0.32347
[71]	validation_0-logloss:0.32054
[72]	validation_0-logloss:0.31762
[73]	validation_0-logloss:0.31481
[74]	validation_0-logloss:0.31198
```

```
[75]	validation_0-logloss:0.30921
[76]	validation_0-logloss:0.30646
[77]	validation_0-logloss:0.30375
[78]	validation_0-logloss:0.30109
[79]	validation_0-logloss:0.29844
[80]	validation_0-logloss:0.29581
[81]	validation_0-logloss:0.29324
[82]	validation_0-logloss:0.29068
[83]	validation_0-logloss:0.28817
[84]	validation_0-logloss:0.28570
[85]	validation_0-logloss:0.28323
[86]	validation_0-logloss:0.28083
[87]	validation_0-logloss:0.27843
[88]	validation_0-logloss:0.27606
[89]	validation_0-logloss:0.27373
[90]	validation_0-logloss:0.27141
[91]	validation_0-logloss:0.26917
[92]	validation_0-logloss:0.26692
[93]	validation_0-logloss:0.26471
[94]	validation_0-logloss:0.26252
[95]	validation_0-logloss:0.26037
[96]	validation_0-logloss:0.25824
[97]	validation_0-logloss:0.25613
[98]	validation_0-logloss:0.25405
[99]	validation_0-logloss:0.25201
[100]	validation_0-logloss:0.24998
[101]	validation_0-logloss:0.24800
[102]	validation_0-logloss:0.24601
[103]	validation_0-logloss:0.24405
[104]	validation_0-logloss:0.24213
[105]	validation_0-logloss:0.24021
[106]	validation_0-logloss:0.23834
[107]	validation_0-logloss:0.23647
[108]	validation_0-logloss:0.23464
[109]	validation_0-logloss:0.23280
[110]	validation_0-logloss:0.23103
[111]	validation_0-logloss:0.22923
[112]	validation_0-logloss:0.22747
[113]	validation_0-logloss:0.22575
[114]	validation_0-logloss:0.22403
[115]	validation_0-logloss:0.22236
[116]	validation_0-logloss:0.22067
[117]	validation_0-logloss:0.21903
[118]	validation_0-logloss:0.21741
[119]	validation_0-logloss:0.21577
[120]	validation_0-logloss:0.21419
[121]	validation_0-logloss:0.21260
[122]	validation_0-logloss:0.21106
[123]	validation_0-logloss:0.20951
```

```
[124]	validation_0-logloss:0.20799
[125]	validation_0-logloss:0.20648
[126]	validation_0-logloss:0.20503
[127]	validation_0-logloss:0.20355
[128]	validation_0-logloss:0.20211
[129]	validation_0-logloss:0.20066
[130]	validation_0-logloss:0.19924
[131]	validation_0-logloss:0.19784
[132]	validation_0-logloss:0.19644
[133]	validation_0-logloss:0.19506
[134]	validation_0-logloss:0.19371
[135]	validation_0-logloss:0.19237
[136]	validation_0-logloss:0.19108
[137]	validation_0-logloss:0.18978
[138]	validation_0-logloss:0.18850
[139]	validation_0-logloss:0.18722
[140]	validation_0-logloss:0.18596
[141]	validation_0-logloss:0.18470
[142]	validation_0-logloss:0.18346
[143]	validation_0-logloss:0.18223
[144]	validation_0-logloss:0.18103
[145]	validation_0-logloss:0.17982
[146]	validation_0-logloss:0.17864
[147]	validation_0-logloss:0.17747
[148]	validation_0-logloss:0.17632
[149]	validation_0-logloss:0.17520
[150]	validation_0-logloss:0.17405
[151]	validation_0-logloss:0.17294
[152]	validation_0-logloss:0.17184
[153]	validation_0-logloss:0.17077
[154]	validation_0-logloss:0.16969
[155]	validation_0-logloss:0.16863
[156]	validation_0-logloss:0.16756
[157]	validation_0-logloss:0.16653
[158]	validation_0-logloss:0.16552
[159]	validation_0-logloss:0.16447
[160]	validation_0-logloss:0.16348
[161]	validation_0-logloss:0.16248
[162]	validation_0-logloss:0.16148
[163]	validation_0-logloss:0.16051
[164]	validation_0-logloss:0.15953
[165]	validation_0-logloss:0.15857
[166]	validation_0-logloss:0.15761
[167]	validation_0-logloss:0.15668
[168]	validation_0-logloss:0.15576
[169]	validation_0-logloss:0.15484
[170]	validation_0-logloss:0.15395
[171]	validation_0-logloss:0.15304
[172]	validation_0-logloss:0.15217
```

```
[173]	validation_0-logloss:0.15129
[174]	validation_0-logloss:0.15042
[175]	validation_0-logloss:0.14958
[176]	validation_0-logloss:0.14872
[177]	validation_0-logloss:0.14788
[178]	validation_0-logloss:0.14704
[179]	validation_0-logloss:0.14623
[180]	validation_0-logloss:0.14541
[181]	validation_0-logloss:0.14462
[182]	validation_0-logloss:0.14383
[183]	validation_0-logloss:0.14304
[184]	validation_0-logloss:0.14227
[185]	validation_0-logloss:0.14149
[186]	validation_0-logloss:0.14074
[187]	validation_0-logloss:0.13999
[188]	validation_0-logloss:0.13925
[189]	validation_0-logloss:0.13853
[190]	validation_0-logloss:0.13780
[191]	validation_0-logloss:0.13710
[192]	validation_0-logloss:0.13639
[193]	validation_0-logloss:0.13570
[194]	validation_0-logloss:0.13501
[195]	validation_0-logloss:0.13434
[196]	validation_0-logloss:0.13366
[197]	validation_0-logloss:0.13301
[198]	validation_0-logloss:0.13236
[199]	validation_0-logloss:0.13172
[200]	validation_0-logloss:0.13108
[201]	validation_0-logloss:0.13044
[202]	validation_0-logloss:0.12981
[203]	validation_0-logloss:0.12918
[204]	validation_0-logloss:0.12854
[205]	validation_0-logloss:0.12794
[206]	validation_0-logloss:0.12733
[207]	validation_0-logloss:0.12672
[208]	validation_0-logloss:0.12615
[209]	validation_0-logloss:0.12557
[210]	validation_0-logloss:0.12498
[211]	validation_0-logloss:0.12443
[212]	validation_0-logloss:0.12387
[213]	validation_0-logloss:0.12331
[214]	validation_0-logloss:0.12277
[215]	validation_0-logloss:0.12223
[216]	validation_0-logloss:0.12169
[217]	validation_0-logloss:0.12117
[218]	validation_0-logloss:0.12065
[219]	validation_0-logloss:0.12014
[220]	validation_0-logloss:0.11963
[221]	validation_0-logloss:0.11913
```

```
[222]	validation_0-logloss:0.11863
[223]	validation_0-logloss:0.11814
[224]	validation_0-logloss:0.11765
[225]	validation_0-logloss:0.11717
[226]	validation_0-logloss:0.11669
[227]	validation_0-logloss:0.11624
[228]	validation_0-logloss:0.11578
[229]	validation_0-logloss:0.11530
[230]	validation_0-logloss:0.11484
[231]	validation_0-logloss:0.11436
[232]	validation_0-logloss:0.11388
[233]	validation_0-logloss:0.11341
[234]	validation_0-logloss:0.11294
[235]	validation_0-logloss:0.11248
[236]	validation_0-logloss:0.11203
[237]	validation_0-logloss:0.11158
[238]	validation_0-logloss:0.11114
[239]	validation_0-logloss:0.11070
[240]	validation_0-logloss:0.11028
[241]	validation_0-logloss:0.10984
[242]	validation_0-logloss:0.10942
[243]	validation_0-logloss:0.10899
[244]	validation_0-logloss:0.10862
[245]	validation_0-logloss:0.10824
[246]	validation_0-logloss:0.10787
[247]	validation_0-logloss:0.10747
[248]	validation_0-logloss:0.10708
[249]	validation_0-logloss:0.10673
[250]	validation_0-logloss:0.10634
[251]	validation_0-logloss:0.10596
[252]	validation_0-logloss:0.10559
[253]	validation_0-logloss:0.10521
[254]	validation_0-logloss:0.10488
[255]	validation_0-logloss:0.10452
[256]	validation_0-logloss:0.10416
[257]	validation_0-logloss:0.10380
[258]	validation_0-logloss:0.10346
[259]	validation_0-logloss:0.10311
[260]	validation_0-logloss:0.10278
[261]	validation_0-logloss:0.10247
[262]	validation_0-logloss:0.10214
[263]	validation_0-logloss:0.10181
[264]	validation_0-logloss:0.10149
[265]	validation_0-logloss:0.10117
[266]	validation_0-logloss:0.10086
[267]	validation_0-logloss:0.10056
[268]	validation_0-logloss:0.10025
[269]	validation_0-logloss:0.09995
[270]	validation_0-logloss:0.09965
```

```
[271]	validation_0-logloss:0.09935
[272]	validation_0-logloss:0.09905
[273]	validation_0-logloss:0.09877
[274]	validation_0-logloss:0.09850
[275]	validation_0-logloss:0.09824
[276]	validation_0-logloss:0.09798
[277]	validation_0-logloss:0.09773
[278]	validation_0-logloss:0.09748
[279]	validation_0-logloss:0.09723
[280]	validation_0-logloss:0.09698
[281]	validation_0-logloss:0.09677
[282]	validation_0-logloss:0.09652
[283]	validation_0-logloss:0.09624
[284]	validation_0-logloss:0.09602
[285]	validation_0-logloss:0.09577
[286]	validation_0-logloss:0.09556
[287]	validation_0-logloss:0.09530
[288]	validation_0-logloss:0.09505
[289]	validation_0-logloss:0.09485
[290]	validation_0-logloss:0.09461
[291]	validation_0-logloss:0.09437
[292]	validation_0-logloss:0.09415
[293]	validation_0-logloss:0.09391
[294]	validation_0-logloss:0.09371
[295]	validation_0-logloss:0.09348
[296]	validation_0-logloss:0.09325
[297]	validation_0-logloss:0.09304
[298]	validation_0-logloss:0.09282
[299]	validation_0-logloss:0.09261
[300]	validation_0-logloss:0.09240
[301]	validation_0-logloss:0.09220
[302]	validation_0-logloss:0.09200
[303]	validation_0-logloss:0.09179
[304]	validation_0-logloss:0.09159
[305]	validation_0-logloss:0.09139
[306]	validation_0-logloss:0.09119
[307]	validation_0-logloss:0.09099
[308]	validation_0-logloss:0.09080
[309]	validation_0-logloss:0.09061
[310]	validation_0-logloss:0.09042
[311]	validation_0-logloss:0.09023
[312]	validation_0-logloss:0.09005
[313]	validation_0-logloss:0.08990
[314]	validation_0-logloss:0.08972
[315]	validation_0-logloss:0.08953
[316]	validation_0-logloss:0.08935
[317]	validation_0-logloss:0.08918
[318]	validation_0-logloss:0.08901
[319]	validation_0-logloss:0.08885
```

```
[320]	validation_0-logloss:0.08868
[321]	validation_0-logloss:0.08853
[322]	validation_0-logloss:0.08835
[323]	validation_0-logloss:0.08817
[324]	validation_0-logloss:0.08800
[325]	validation_0-logloss:0.08783
[326]	validation_0-logloss:0.08766
[327]	validation_0-logloss:0.08749
[328]	validation_0-logloss:0.08734
[329]	validation_0-logloss:0.08719
[330]	validation_0-logloss:0.08704
[331]	validation_0-logloss:0.08689
[332]	validation_0-logloss:0.08674
[333]	validation_0-logloss:0.08661
[334]	validation_0-logloss:0.08648
[335]	validation_0-logloss:0.08636
[336]	validation_0-logloss:0.08622
[337]	validation_0-logloss:0.08610
[338]	validation_0-logloss:0.08596
[339]	validation_0-logloss:0.08582
[340]	validation_0-logloss:0.08571
[341]	validation_0-logloss:0.08559
[342]	validation_0-logloss:0.08546
[343]	validation_0-logloss:0.08535
[344]	validation_0-logloss:0.08524
[345]	validation_0-logloss:0.08514
[346]	validation_0-logloss:0.08503
[347]	validation_0-logloss:0.08492
[348]	validation_0-logloss:0.08482
[349]	validation_0-logloss:0.08473
[350]	validation_0-logloss:0.08462
[351]	validation_0-logloss:0.08451
[352]	validation_0-logloss:0.08442
[353]	validation_0-logloss:0.08432
[354]	validation_0-logloss:0.08422
[355]	validation_0-logloss:0.08411
[356]	validation_0-logloss:0.08402
[357]	validation_0-logloss:0.08393
[358]	validation_0-logloss:0.08383
[359]	validation_0-logloss:0.08375
[360]	validation_0-logloss:0.08366
[361]	validation_0-logloss:0.08358
[362]	validation_0-logloss:0.08349
[363]	validation_0-logloss:0.08341
[364]	validation_0-logloss:0.08332
[365]	validation_0-logloss:0.08323
[366]	validation_0-logloss:0.08314
[367]	validation_0-logloss:0.08307
[368]	validation_0-logloss:0.08299
```

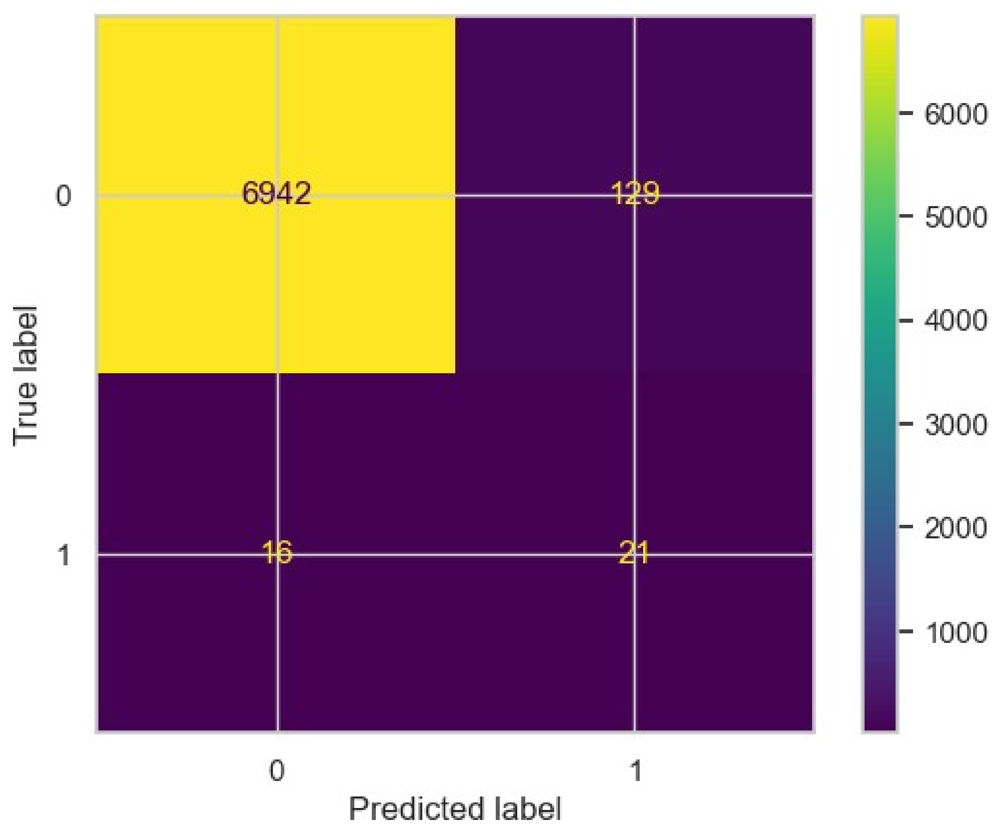
```
[369]	validation_0-logloss:0.08291
[370]	validation_0-logloss:0.08283
[371]	validation_0-logloss:0.08276
[372]	validation_0-logloss:0.08268
[373]	validation_0-logloss:0.08261
[374]	validation_0-logloss:0.08253
[375]	validation_0-logloss:0.08246
[376]	validation_0-logloss:0.08239
[377]	validation_0-logloss:0.08232
[378]	validation_0-logloss:0.08224
[379]	validation_0-logloss:0.08218
[380]	validation_0-logloss:0.08211
[381]	validation_0-logloss:0.08204
[382]	validation_0-logloss:0.08197
[383]	validation_0-logloss:0.08188
[384]	validation_0-logloss:0.08182
[385]	validation_0-logloss:0.08176
[386]	validation_0-logloss:0.08170
[387]	validation_0-logloss:0.08163
[388]	validation_0-logloss:0.08157
[389]	validation_0-logloss:0.08149
[390]	validation_0-logloss:0.08143
[391]	validation_0-logloss:0.08135
[392]	validation_0-logloss:0.08128
[393]	validation_0-logloss:0.08123
[394]	validation_0-logloss:0.08117
[395]	validation_0-logloss:0.08112
[396]	validation_0-logloss:0.08106
[397]	validation_0-logloss:0.08102
[398]	validation_0-logloss:0.08097
[399]	validation_0-logloss:0.08093
[400]	validation_0-logloss:0.08088
[401]	validation_0-logloss:0.08083
[402]	validation_0-logloss:0.08079
[403]	validation_0-logloss:0.08075
[404]	validation_0-logloss:0.08071
[405]	validation_0-logloss:0.08066
[406]	validation_0-logloss:0.08062
[407]	validation_0-logloss:0.08058
[408]	validation_0-logloss:0.08054
[409]	validation_0-logloss:0.08049
[410]	validation_0-logloss:0.08045
[411]	validation_0-logloss:0.08042
[412]	validation_0-logloss:0.08038
[413]	validation_0-logloss:0.08035
[414]	validation_0-logloss:0.08031
[415]	validation_0-logloss:0.08027
[416]	validation_0-logloss:0.08024
[417]	validation_0-logloss:0.08020
```

```
[418]	validation_0-logloss:0.08017
[419]	validation_0-logloss:0.08014
[420]	validation_0-logloss:0.08011
[421]	validation_0-logloss:0.08008
[422]	validation_0-logloss:0.08006
[423]	validation_0-logloss:0.08002
[424]	validation_0-logloss:0.08000
[425]	validation_0-logloss:0.07996
[426]	validation_0-logloss:0.07993
[427]	validation_0-logloss:0.07991
[428]	validation_0-logloss:0.07988
[429]	validation_0-logloss:0.07984
[430]	validation_0-logloss:0.07981
[431]	validation_0-logloss:0.07981
[432]	validation_0-logloss:0.07977
[433]	validation_0-logloss:0.07975
[434]	validation_0-logloss:0.07973
[435]	validation_0-logloss:0.07972
[436]	validation_0-logloss:0.07969
[437]	validation_0-logloss:0.07967
[438]	validation_0-logloss:0.07964
[439]	validation_0-logloss:0.07963
[440]	validation_0-logloss:0.07961
[441]	validation_0-logloss:0.07960
[442]	validation_0-logloss:0.07957
[443]	validation_0-logloss:0.07954
[444]	validation_0-logloss:0.07953
[445]	validation_0-logloss:0.07950
[446]	validation_0-logloss:0.07948
[447]	validation_0-logloss:0.07947
[448]	validation_0-logloss:0.07944
[449]	validation_0-logloss:0.07941
[450]	validation_0-logloss:0.07940
[451]	validation_0-logloss:0.07938
[452]	validation_0-logloss:0.07936
[453]	validation_0-logloss:0.07934
[454]	validation_0-logloss:0.07933
[455]	validation_0-logloss:0.07931
[456]	validation_0-logloss:0.07930
[457]	validation_0-logloss:0.07929
[458]	validation_0-logloss:0.07927
[459]	validation_0-logloss:0.07925
[460]	validation_0-logloss:0.07924
[461]	validation_0-logloss:0.07924
[462]	validation_0-logloss:0.07922
[463]	validation_0-logloss:0.07921
[464]	validation_0-logloss:0.07921
[465]	validation_0-logloss:0.07920
[466]	validation_0-logloss:0.07919
```

```
[467] validation_0-logloss:0.07919
[468] validation_0-logloss:0.07919
[469] validation_0-logloss:0.07918
[470] validation_0-logloss:0.07918
[471] validation_0-logloss:0.07918
[472] validation_0-logloss:0.07918
[473] validation_0-logloss:0.07918
[474] validation_0-logloss:0.07917
[475] validation_0-logloss:0.07917
[476] validation_0-logloss:0.07915
[477] validation_0-logloss:0.07915
[478] validation_0-logloss:0.07915
[479] validation_0-logloss:0.07914
[480] validation_0-logloss:0.07913
[481] validation_0-logloss:0.07914
[482] validation_0-logloss:0.07913
[483] validation_0-logloss:0.07915
[484] validation_0-logloss:0.07915
[485] validation_0-logloss:0.07915
[486] validation_0-logloss:0.07914
[487] validation_0-logloss:0.07915
[488] validation_0-logloss:0.07917
[489] validation_0-logloss:0.07917
[490] validation_0-logloss:0.07918
[491] validation_0-logloss:0.07918
[492] validation_0-logloss:0.07916
[493] validation_0-logloss:0.07918
[494] validation_0-logloss:0.07918
[495] validation_0-logloss:0.07918
[496] validation_0-logloss:0.07919
[497] validation_0-logloss:0.07920
[498] validation_0-logloss:0.07921
[499] validation_0-logloss:0.07921
[500] validation_0-logloss:0.07921
[501] validation_0-logloss:0.07922
[502] validation_0-logloss:0.07922
[503] validation_0-logloss:0.07924
[504] validation_0-logloss:0.07926
[505] validation_0-logloss:0.07926
[506] validation_0-logloss:0.07926
[507] validation_0-logloss:0.07925
[508] validation_0-logloss:0.07925
[509] validation_0-logloss:0.07927
[510] validation_0-logloss:0.07928
[511] validation_0-logloss:0.07929
[512] validation_0-logloss:0.07931
[513] validation_0-logloss:0.07932
[514] validation_0-logloss:0.07933
[515] validation_0-logloss:0.07935
[516] validation_0-logloss:0.07937
```

```
[517] validation_0-logloss:0.07939
[518] validation_0-logloss:0.07940
[519] validation_0-logloss:0.07942
[520] validation_0-logloss:0.07945
[521] validation_0-logloss:0.07947
[522] validation_0-logloss:0.07950
[523] validation_0-logloss:0.07951
[524] validation_0-logloss:0.07954
[525] validation_0-logloss:0.07956
[526] validation_0-logloss:0.07957
[527] validation_0-logloss:0.07959
[528] validation_0-logloss:0.07961
[529] validation_0-logloss:0.07963
[530] validation_0-logloss:0.07964
[531] validation_0-logloss:0.07966
[532] validation_0-logloss:0.07969
[533] validation_0-logloss:0.07971
[534] validation_0-logloss:0.07973
[535] validation_0-logloss:0.07975
[536] validation_0-logloss:0.07976
[537] validation_0-logloss:0.07978
[538] validation_0-logloss:0.07980
[539] validation_0-logloss:0.07982
[540] validation_0-logloss:0.07985
[541] validation_0-logloss:0.07987
[542] validation_0-logloss:0.07989
[543] validation_0-logloss:0.07990
[544] validation_0-logloss:0.07993
[545] validation_0-logloss:0.07994
[546] validation_0-logloss:0.07996
[547] validation_0-logloss:0.07997
[548] validation_0-logloss:0.07999
[549] validation_0-logloss:0.08000
[550] validation_0-logloss:0.08002
[551] validation_0-logloss:0.08005
[552] validation_0-logloss:0.08006
[553] validation_0-logloss:0.08008
[554] validation_0-logloss:0.08010
[555] validation_0-logloss:0.08012
[556] validation_0-logloss:0.08015
[557] validation_0-logloss:0.08016
[558] validation_0-logloss:0.08017
[559] validation_0-logloss:0.08019
[560] validation_0-logloss:0.08022
[561] validation_0-logloss:0.08023
[562] validation_0-logloss:0.08025
[563] validation_0-logloss:0.08027
[564] validation_0-logloss:0.08030
[565] validation_0-logloss:0.08033
```

```
[566] validation_0-logloss:0.08035
[567] validation_0-logloss:0.08039
[568] validation_0-logloss:0.08041
[569] validation_0-logloss:0.08043
[570] validation_0-logloss:0.08046
[571] validation_0-logloss:0.08049
[572] validation_0-logloss:0.08051
[573] validation_0-logloss:0.08054
[574] validation_0-logloss:0.08058
[575] validation_0-logloss:0.08061
[576] validation_0-logloss:0.08063
[577] validation_0-logloss:0.08065
[578] validation_0-logloss:0.08068
[579] validation_0-logloss:0.08071
0.9796004501969612
0.8929989460572961
0.24859749027989134

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x229453e6ed0>
```




```
from sklearn.model_selection import StratifiedKFold, cross_validate
from sklearn.metrics import make_scorer, accuracy_score,
```

```python
roc_auc_score, average_precision_score, precision_score
from xgboost import XGBClassifier

xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss',
device='cuda')


eval_metrics = {
    'accuracy': make_scorer(accuracy_score),
    'roc_auc': make_scorer(roc_auc_score),
    'pr_auc': make_scorer(average_precision_score),
    'ppv': make_scorer(precision_score)
}

cv = StratifiedKFold(n_splits=5, shuffle=True)

params = {'scale_pos_weight': sum(y_train == 0) / sum(y_train),
'max_depth': 10, 'learning_rate': 0.01, 'n_estimators': 10000}

xgb.set_params(**params)

cv_results = cross_validate(xgb, X_train, y_train, cv=cv,
scoring=eval_metrics, return_train_score=False)
cv_results_df = pd.DataFrame(cv_results).mean()

cv_std = pd.DataFrame(cv_results).std()
results = pd.concat([cv_results_df, cv_std], axis=1)
results.rename(columns={0: 'mean', 1: 'std'}, inplace=True)

results
```

	mean	std
fit_time	62.309404	0.989764
score_time	0.329799	0.013141
test_accuracy	0.984237	0.000627
test_roc_auc	0.524761	0.008169
test_pr_auc	0.034180	0.010465
test_ppv	0.367929	0.133334

```python
results.to_latex()
```

```
'\\begin{tabular}{lrr}\n\\toprule\n & mean & std \\\\\n\\midrule\
nfit_time & 17.941201 & 0.392995 \\\\\nscore_time & 0.080000 &
0.001000 \\\\\ntest_accuracy & 0.984139 & 0.000679 \\\\\ntest_roc_auc
& 0.522141 & 0.008549 \\\\\ntest_pr_auc & 0.031618 & 0.011197 \\\\
ntest_ppv & 0.334947 & 0.141804 \\\\\n\\bottomrule\n\\end{tabular}\n'
```

```python
clean_mem(admissions_df)
clean_mem(admission_time_dict)
clean_mem(anchor_age_dict)
clean_mem(anchor_age_tuples)
```

```
clean_mem(axes)
clean_mem(chartevent_definitions)
clean_mem(dem_feats)
clean_mem(patients_df)
clean_mem(temp_df)
```