Introduction

Problem Statement

Hospital beds are very limited in many of the countries. In lower income countries, there are approximately 0.1 ICU beds per 100,000 citizens. Number of beds can also be critical in countries where there are ample beds for the population during the times of an outbreak of an epidemic.

Importance of the Solution

When the number of patients reporting for admission is large, a difficult decision needs to be made whether to admit the patient or not especially to the ICU. If the decision is made on certain criteria, it would be an informed decision and not based on an ad-hoc first cum fist serve basis. Chaotic scenes were observed during COVID. Quite a few people could have been saved if hospital beds were available during COVID peaks.

Approach / Pitch

My targeted approach will be to develop two models which will aim to predict mortality; one based on data collected during admission of a patient and another based on data collected during admission and laboratory results typically available within first twenty-four hours of admission.

Various features are available at the time of admission of a patient such as such as demographic characteristics such as age, sex, ethnicity, height, weight, etc., vital signs such as heart rate, systolic blood pressure, diastolic blood pressure, respiratory rate, body temperature, saturation pulse oxygen, etc. In addition, various features are available through laboratory tests within first twenty-four hours of admission such as red blood cells, mean corpuscular hemoglobin concentration, etc.

I am taking he two-stepped approach to achieve the following objectives:

- If the model performance is good with the data available during the admission, it can be used to evaluate whether to admit the patient based on the patient outcome.
- If the model performance is good with the data available during admission together with those available within first twenty-four hours, the patient can be moved to an alternate arrangement and beds made available to patients with positive predicted outcome.
- One model over the other can also be used if the models' performances are significantly different.

Life is important and every effort should be made to preserve a life. Availability of data and the tools available to us provides an opportunity to deviate from norms which are considered actable by the society and develop new norms based on scientific tools developed for justification and acceptance.

Data Source

I have obtained the data from Kaggle, In Hospital Mortality Prediction, In Hospital Mortality Prediction | Kaggle. The dataset has 51 variables. The variables included are: demographic characteristics (age at the time of hospital admission, sex, ethnicity, weight, and height); vital signs (heart rate, (HR), systolic blood pressure [SBP], diastolic blood pressure [DBP], mean blood pressure, respiratory rate, body temperature, saturation pulse oxygen [SPO2], urine output [first 24 h]); comorbidities (hypertension, atrial fibrillation, ischemic heart disease, diabetes mellitus, depression, hypoferric anemia, hyperlipidemia, chronic kidney disease (CKD), and chronic obstructive pulmonary disease [COPD]); and laboratory variables (hematocrit,

red blood cells, mean corpuscular hemoglobin [MCH], mean corpuscular hemoglobin concentration [MCHC], mean corpuscular volume [MCV], red blood cell distribution width [RDW], platelet count, white blood cells, neutrophils, basophils, lymphocytes, prothrombin time [PT], international normalized ratio [INR], NT-proBNP, creatine kinase, creatinine, blood urea nitrogen [BUN] glucose, potassium, sodium, calcium, chloride, magnesium, the anion gap, bicarbonate, lactate, hydrogen ion concentration [pH], partial pressure of CO2 in arterial blood, and LVEF).

Milestones Summary

Milestone #1: EDA

During exploratory Data Analysis or EDA, I created the following four diagrams.

- Frequency histogram of outcome (Alive / Not Alive) by age. Objective was to check whether age had any influence on the outcome (refer **FIGURE 1**).
- Scatterplot of BMI vs Outcome to check whether higher BMI resulted in death in the ICU (refer **FIGURE 2**).
- Frequency histogram of BMI by diabetes to check whether BMI plays a role in a person having diabetes or not as the general belief is that people with higher weight have diabetes (refer **FIGURE 3**).
- Count plot of outcome by gender to check whether a male or a female was more likely to die or survive at the hospital (refer **FIGURE 4**).

Figure 1: Frequency Histogram of Age

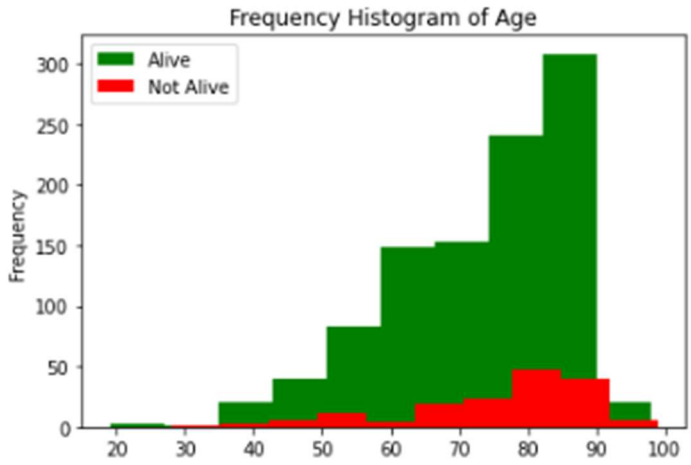


Figure 2: Scatterplot of BMI Vs Outcome

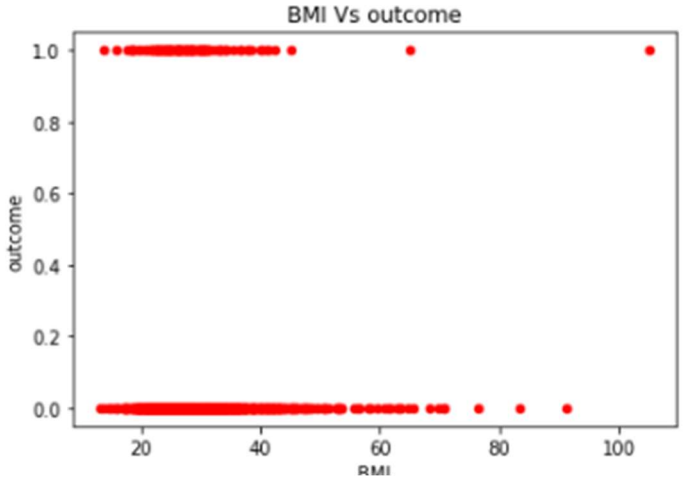


Figure 3: Frequency Histogram of BMI by Diabetes

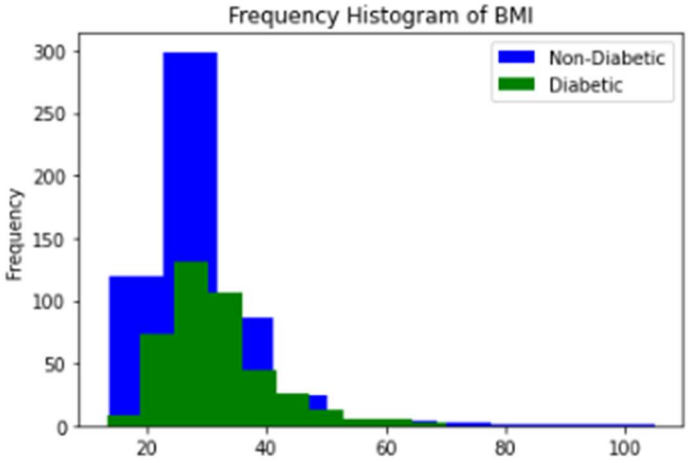
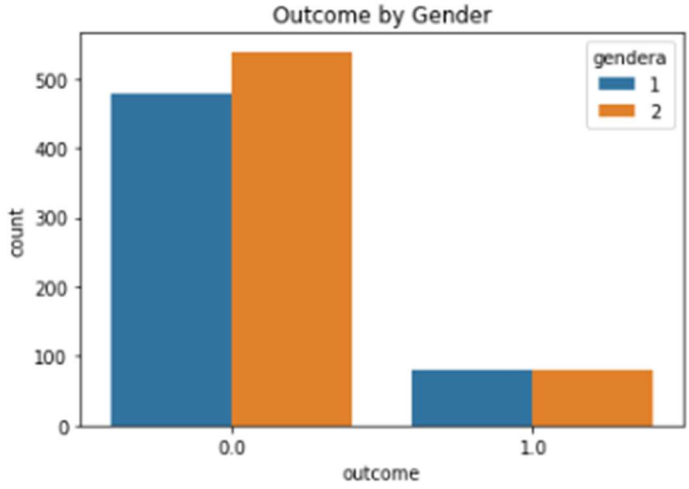


Figure 4: Count of Outcome by Gender



Summary of EDA

Frequency histogram of outcome by Age does not provide conclusive evidence that age was related to the outcome that the person died in the hospital or not.

BMI vs outcome scatterplot tends to indicate that people with high BMI were also likely to come out alive. Therefore no, conclusive evidence.

Frequency histogram of BMI again does not show conclusive evidence that BMI causes diabetes; however, it does indicate that people with BMI of ~ 25 to 35 tend to be diabetic.

Outcome by gender graph does not indicate that gender has any implications on the outcome.

In summary, individually the selected variables do not have any significant impact on the outcome. Also, BMI and diabetes do not show comorbidity per the graph.

Milestone 2: Data Preparation

Data preparation is an important step any data science project as it can lead to potential saving in time, reduction or elimination in errors thereby resulting in correct outcomes and the result is better and correct decision making.

Steps taken for data preparation:

1. It is a good practice to eliminate any space between the multi-word variable name and replace the space by an underscore. Therefore, I eliminated the spaces and replaced spaces by underscore to separate words in a column name.
2. I dropped features which are not indicative of acuteness of the patient's current condition and / or are indicative of generic manageable conditions or have features explaining similar characteristics such as: group, ID, depression or Gender which does not affect the outcome per the outcome. In addition, on similar grounds, following features were also dropped:
    - Hyperlipemia - Hyperlipidemia is a condition that incorporates various genetic and acquired disorders that describe elevated lipid levels within the human body.
    - PT - The prothrombin time, sometimes referred to as PT or pro time, test is a test to evaluate blood clotting.
    - RDW - A red cell distribution width (RDW) test measures the differences in the volume and size of your red blood cells (erythrocytes).
    - Similarly Creatine kinase, Urea nitrogen, Blood potassium, Blood sodium, Blood calcium, Chloride, Anion gap, PH, Lactic acid, PCO2, and Bicarbonate.
3. The next step included dropping any feature that had more than 20% of the values as missing in the dataset. Only Basophils was found to have ~22% of the rows with missing values and therefore, subsequently dropped from the analysis.
4. Checked for the data types for object and none of the variables were object type and thereby not requiring change in the type prior to analysis.
5. Checked for duplicate rows and dropped as applicable.
6. For numerical columns, filled in any missing data with the median value as the max percentage of missing rows was observed to be ~18% and is not expected to adversely affect the data.
7. Introduced Urine_output_BMI which is urine output normalized by BMI as body weight can be a factor in the urine output.

8. Created descriptive statistics of all columns to check the statistics for obvious inaccuracies.

Milestone 3: Model Building and Evaluation

Our target variable, outcome (alive or not alive) is binary. Therefore, logistic regression was chosen for model building. Before building the model, training and test data which were obtained from the dataset were standardized using StandardScaler from sklearn.preprocessing.

As stated above, the following two models were built:

- Model 1: Features included data available during admission such as age, BMI, etc. Outcome was the target variable.
- Model 2: Features included in Model 1 along with laboratory test data obtained within first twenty-four hours of admission. Outcome was the target variable.

Model evaluation in both the above cases was done by checking for overfitting. Model accuracy scores were calculated both on training and test data against predicted outcome for both cases. Model accuracies for both models and on training and test data were found to be in similar range. Therefore, it was determined that there is no overfitting in the models.

Codes associated with these milestones are included as **APPENDIX 1**.

Conclusion

Model Result / Analysis Result

The accuracy scores for both models were also observed to be in the similar range 88%; thereby, indicating that addition of laboratory features did not improve our model significantly. Therefore, to be more efficient in decision making and to avoid chaos, Model with parameters available at the time of admission of the patient can be utilized to make the admission / no admission decision.

Model Deployment Status

The model is ready for deployment though only for patients with heart failure condition. The data utilized was specific for the patients with Heart Failure categorizations.

Recommendations / challenges and Opportunities

More general data with general patients with no specific conditions are required to be analyzed to make a general ICU admission decision. In addition, the outcome is required to be monitored for performance and the model be tweaked as necessary to capture the dynamic data available at a hospital which would be more reflective of the people visiting a particular hospital.

My recommendation is that a more generalized collection of data followed by a similar analysis be performed for general ICU decision making. In addition, models for specific conditions should also be created and checked for performance against the general model based on actual data collected after deployment. This will help in determining whether specific health conditions help in model performance or not.

Ethical questions will always remain in the context of not admitting any person based on model prediction and denying him or her an opportunity to fight the condition with hospital support.

Appendix 1

In [1]:
```
# Shashi Bhushan
# MSDS DSC 550, Winter 2022
# Milestone 3
```

In [2]:
```
# Milestone 1
# Importing libraries
import pandas as pd
import matplotlib
from matplotlib import pyplot as plt
import seaborn as sns
```

In [3]:
```
df = pd.read_csv('data01.csv')
df.head()
```

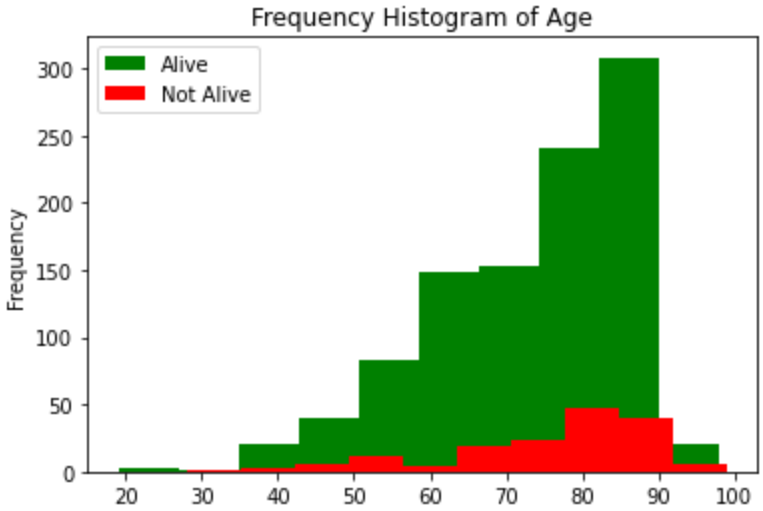
Out[3]:

	group	ID	outcome	age	gendera	BMI	hypertensive	atrialfibrillation	CHD with no MI	diabetes	...	Blo sodi
0	1	125047	0.0	72	1	37.588179	0	0	0	1	...	138.750(
1	1	139812	0.0	75	2	NaN	0	0	0	0	...	138.888
2	1	109787	0.0	83	2	26.572634	0	0	0	0	...	140.714
3	1	130587	0.0	43	2	83.264629	0	0	0	0	...	138.500(
4	1	138290	0.0	75	2	31.824842	1	0	0	0	...	136.666

5 rows × 51 columns

In [4]:
```
# Creating histogram of age by outcome
age1=df.loc[df.outcome==1.0, 'age']
age0=df.loc[df.outcome==0.0, 'age']
plt.hist(age0,  color='g', label='Alive')
plt.hist(age1,  color='r', label='Not Alive')
plt.gca().set(title='Frequency Histogram of Age', ylabel='Frequency')
plt.legend()
```

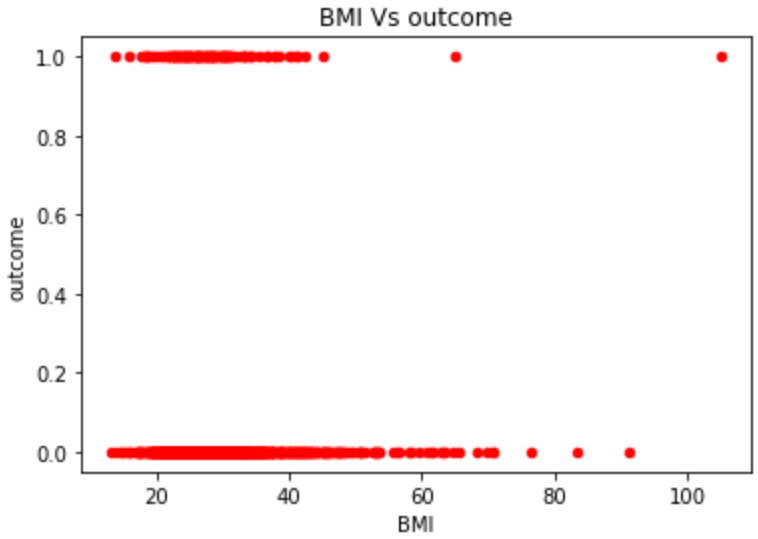
Out[4]:
```
<matplotlib.legend.Legend at 0x231fd6984c0>
```



In [5]:
```
# Scatterplot of BMI and outcome
df.plot(kind='scatter', x='BMI',y='outcome',color='red')
plt.title("BMI Vs outcome")
```

```
Text(0.5, 1.0, 'BMI Vs outcome')
```

Out[5]:

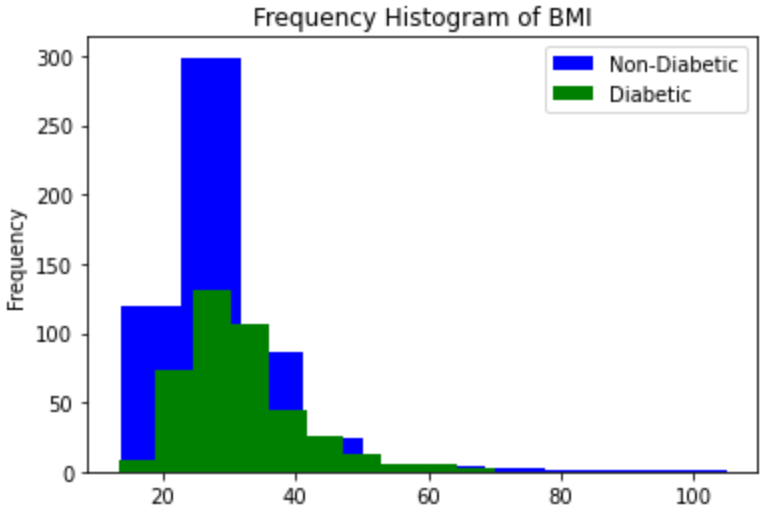


In [6]:

```
# histogram of BMI and diabetes
dia1=df.loc[df.diabetes==1.0, 'BMI']
dia0=df.loc[df.diabetes==0.0, 'BMI']
plt.hist(dia0,  color='b', label='Non-Diabetic')
plt.hist(dia1,  color='g', label='Diabetic')
plt.gca().set(title='Frequency Histogram of BMI', ylabel='Frequency')
plt.legend()
```

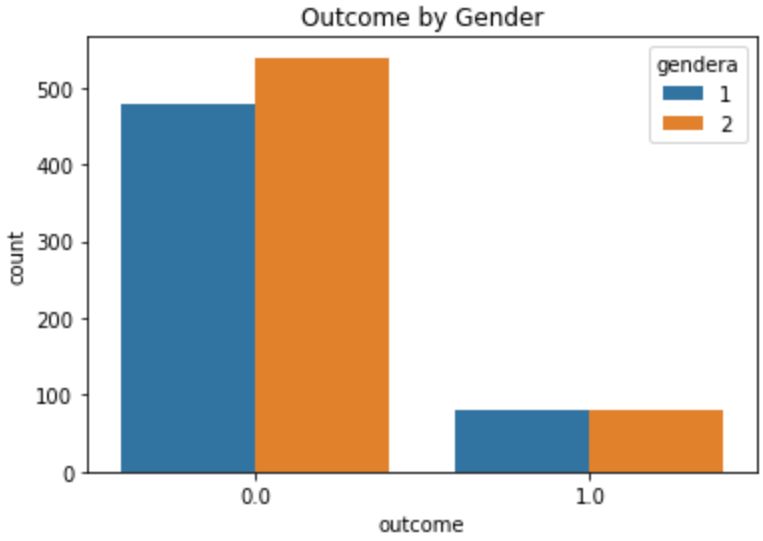
Out[6]:

```
<matplotlib.legend.Legend at 0x231fde95d90>
```



In [7]:

```
plt.title("Outcome by Gender")
sns.countplot(x ='outcome', hue = "gendera", data = df)
plt.show()
```



In [8]:
```
# Summary
#Frequency histogram of outcome by Age does not provide a conclusive evidence that
# age was related to the outcome that the person died in the hospital or not.
# BMI vs outcome scatterplot tends to indicate that people with high BMI were also likel
# Frequency histogram of BMI again does not show conclusive evidence that BMI causes dia
# however, it does indicate that people with BMI of ~ 25 to 35 tend to be diabetic.
# Outcome by gender graph does not indicate that gender has any implications on the outc
# In summary, individually the selected variables do not have any significant impact on
# Also, BMI and diabetes do not show comorbidity per the graph.
```

In [9]:
```
# Milestone 2
```

In [10]:
```
#  Changing Column Names to introduce underscope between names with two or more words ha
# This is to avoid using '' between column names having names with two or more words
df.columns = df.columns.map(lambda x : x.replace(" ", "_"))
```

In [11]:
```
df.describe()
```

Out[11]:

	group	ID	outcome	age	gendera	BMI	hypertensive	atrialfibrillati
count	1177.000000	1177.000000	1176.000000	1177.000000	1177.000000	962.000000	1177.000000	1177.0000
mean	1.299065	150778.120646	0.135204	74.055225	1.525064	30.188278	0.717927	0.4511
std	0.458043	29034.669513	0.342087	13.434061	0.499584	9.325997	0.450200	0.4978
min	1.000000	100213.000000	0.000000	19.000000	1.000000	13.346801	0.000000	0.0000
25%	1.000000	125603.000000	0.000000	65.000000	1.000000	24.326461	0.000000	0.0000
50%	1.000000	151901.000000	0.000000	77.000000	2.000000	28.312474	1.000000	0.0000
75%	2.000000	176048.000000	0.000000	85.000000	2.000000	33.633509	1.000000	1.0000
max	2.000000	199952.000000	1.000000	99.000000	2.000000	104.970366	1.000000	1.0000

8 rows × 51 columns

In [12]:
```
# Dropping features which are not indicative of acuteness of the patient's currernt cond
# of generic manageable conditions or have features explaining similar characteristics s
## group, ID, depression or Gender which does not affect the outcome per the outcome

## Hyperlipemia - Hyperlipidemia is a condition that incorporates various genetic and ac
## that describe elevated lipid levels within the human body.

## PT - The prothrombin time, sometimes referred to as PT or pro time, test is a test to
```

```
## RDW - A red cell distribution width (RDW) test measures the differences in the
## volume and size of your red blood cells (erythrocytes).

## Similarly Creatine kinase, Urea nitrogen, Blood potassium, Blood sodium, Blood calciu
## Lactic acid, PCO2, and Bicarbonate variables were dropped

df.drop(labels=['group', 'ID', 'gendera', 'depression', 'Hyperlipemia', 'PT', 'RDW', 'Cr
                'Urea_nitrogen', 'Blood_potassium', 'Blood_sodium', 'Blood_calcium', 'Ch
               'Anion_gap', 'PH', 'Lactic_acid','PCO2', 'Bicarbonate'], axis=1, inplace=
df.describe()
```

Out[12]:

	outcome	age	BMI	hypertensive	atrialfibrillation	CHD_with_no_MI	diabetes	deficie
count	1176.000000	1177.000000	962.000000	1177.000000	1177.000000	1177.000000	1177.000000	
mean	0.135204	74.055225	30.188278	0.717927	0.451147	0.085811	0.421410	
std	0.342087	13.434061	9.325997	0.450200	0.497819	0.280204	0.493995	
min	0.000000	19.000000	13.346801	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	65.000000	24.326461	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	77.000000	28.312474	1.000000	0.000000	0.000000	0.000000	
75%	0.000000	85.000000	33.633509	1.000000	1.000000	0.000000	1.000000	
max	1.000000	99.000000	104.970366	1.000000	1.000000	1.000000	1.000000	

8 rows × 33 columns

In [13]:
```
# finding percentage of missing values in each feature
import numpy as np
for column in df.columns:
    print('{} has {} % missing values'.format(column,np.round(df[column].isnull().sum()/
```

```
outcome has 0.085 % missing values
age has 0.0 % missing values
BMI has 18.2668 % missing values
hypertensive has 0.0 % missing values
atrialfibrillation has 0.0 % missing values
CHD_with_no_MI has 0.0 % missing values
diabetes has 0.0 % missing values
deficiencyanemias has 0.0 % missing values
Renal_failure has 0.0 % missing values
COPD has 0.0 % missing values
heart_rate has 1.1045 % missing values
Systolic_blood_pressure has 1.3594 % missing values
Diastolic_blood_pressure has 1.3594 % missing values
Respiratory_rate has 1.1045 % missing values
temperature has 1.6143 % missing values
SP_O2 has 1.1045 % missing values
Urine_output has 3.0586 % missing values
hematocrit has 0.0 % missing values
RBC has 0.0 % missing values
MCH has 0.0 % missing values
MCHC has 0.0 % missing values
MCV has 0.0 % missing values
Leucocyte has 0.0 % missing values
Platelets has 0.0 % missing values
Neutrophils has 12.2345 % missing values
Basophils has 22.0051 % missing values
Lymphocyte has 12.3195 % missing values
INR has 1.6992 % missing values
NT-proBNP has 0.0 % missing values
Creatinine has 0.0 % missing values
```

```
glucose has 1.5293 % missing values
Magnesium_ion has 0.0 % missing values
EF has 0.0 % missing values
```

In [14]:
```
# Drop any features that are missing more than 20% of their values.Only Basophils will d
df=df.drop(df.columns[df.isnull().mean()>0.20],axis=1)
df.describe()
```

Out[14]:

	outcome	age	BMI	hypertensive	atrialfibrillation	CHD_with_no_MI	diabetes	deficie
count	1176.000000	1177.000000	962.000000	1177.000000	1177.000000	1177.000000	1177.000000	
mean	0.135204	74.055225	30.188278	0.717927	0.451147	0.085811	0.421410	
std	0.342087	13.434061	9.325997	0.450200	0.497819	0.280204	0.493995	
min	0.000000	19.000000	13.346801	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	65.000000	24.326461	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	77.000000	28.312474	1.000000	0.000000	0.000000	0.000000	
75%	0.000000	85.000000	33.633509	1.000000	1.000000	0.000000	1.000000	
max	1.000000	99.000000	104.970366	1.000000	1.000000	1.000000	1.000000	

8 rows × 32 columns

In [15]:
```
df.dtypes
```

Out[15]:
```
outcome                    float64
age                          int64
BMI                        float64
hypertensive                 int64
atrialfibrillation           int64
CHD_with_no_MI               int64
diabetes                     int64
deficiencyanemias            int64
Renal_failure                int64
COPD                         int64
heart_rate                 float64
Systolic_blood_pressure    float64
Diastolic_blood_pressure   float64
Respiratory_rate           float64
temperature                float64
SP_O2                      float64
Urine_output               float64
hematocrit                 float64
RBC                        float64
MCH                        float64
MCHC                       float64
MCV                        float64
Leucocyte                  float64
Platelets                  float64
Neutrophils                float64
Lymphocyte                 float64
INR                        float64
NT-proBNP                  float64
Creatinine                 float64
glucose                    float64
Magnesium_ion              float64
EF                           int64
dtype: object
```

In [16]:
```
# We see that none of the features are object type
```

In [17]:
```
# dropping duplicate rows if any
df.drop_duplicates(inplace=True)
df.describe()
```

Out[17]:

	outcome	age	BMI	hypertensive	atrialfibrillation	CHD_with_no_MI	diabetes	deficie
count	1176.000000	1177.000000	962.000000	1177.000000	1177.000000	1177.000000	1177.000000	
mean	0.135204	74.055225	30.188278	0.717927	0.451147	0.085811	0.421410	
std	0.342087	13.434061	9.325997	0.450200	0.497819	0.280204	0.493995	
min	0.000000	19.000000	13.346801	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	65.000000	24.326461	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	77.000000	28.312474	1.000000	0.000000	0.000000	0.000000	
75%	0.000000	85.000000	33.633509	1.000000	1.000000	0.000000	1.000000	
max	1.000000	99.000000	104.970366	1.000000	1.000000	1.000000	1.000000	

8 rows × 32 columns

In [18]:
```
# For numerical columns, fill in any missing data with the median value as the max perce
# and is not expected to advsersely affect the data
df = df.fillna(df.median(numeric_only=True))
df.describe()
```

Out[18]:

	outcome	age	BMI	hypertensive	atrialfibrillation	CHD_with_no_MI	diabetes	defic
count	1177.000000	1177.000000	1177.000000	1177.000000	1177.000000	1177.000000	1177.000000	
mean	0.135089	74.055225	29.845629	0.717927	0.451147	0.085811	0.421410	
std	0.341964	13.434061	8.461626	0.450200	0.497819	0.280204	0.493995	
min	0.000000	19.000000	13.346801	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	65.000000	25.276974	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	77.000000	28.312474	1.000000	0.000000	0.000000	0.000000	
75%	0.000000	85.000000	32.101349	1.000000	1.000000	0.000000	1.000000	
max	1.000000	99.000000	104.970366	1.000000	1.000000	1.000000	1.000000	

8 rows × 32 columns

In [19]:
```
# Introducing Urine_output_BMI normalized by BMI as body weight can be a factor in the u
df['Urine_output_BMI'] = df['Urine_output']/df['BMI']
df.describe()
```

Out[19]:

	outcome	age	BMI	hypertensive	atrialfibrillation	CHD_with_no_MI	diabetes	defic
count	1177.000000	1177.000000	1177.000000	1177.000000	1177.000000	1177.000000	1177.000000	
mean	0.135089	74.055225	29.845629	0.717927	0.451147	0.085811	0.421410	
std	0.341964	13.434061	8.461626	0.450200	0.497819	0.280204	0.493995	
min	0.000000	19.000000	13.346801	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	65.000000	25.276974	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	77.000000	28.312474	1.000000	0.000000	0.000000	0.000000	
75%	0.000000	85.000000	32.101349	1.000000	1.000000	0.000000	1.000000	

max	1.000000	99.000000	104.970366	1.000000	1.000000	1.000000	1.000000

8 rows × 33 columns

In [20]:
```
#  Validating data by Showing descriptive statistics of all columns
df.describe(include = 'all')
```

Out[20]:

	outcome	age	BMI	hypertensive	atrialfibrillation	CHD_with_no_MI	diabetes	defic
count	1177.000000	1177.000000	1177.000000	1177.000000	1177.000000	1177.000000	1177.000000	
mean	0.135089	74.055225	29.845629	0.717927	0.451147	0.085811	0.421410	
std	0.341964	13.434061	8.461626	0.450200	0.497819	0.280204	0.493995	
min	0.000000	19.000000	13.346801	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	65.000000	25.276974	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	77.000000	28.312474	1.000000	0.000000	0.000000	0.000000	
75%	0.000000	85.000000	32.101349	1.000000	1.000000	0.000000	1.000000	
max	1.000000	99.000000	104.970366	1.000000	1.000000	1.000000	1.000000	

8 rows × 33 columns

In [21]:
```
df.dtypes
```

Out[21]:
```
outcome                    float64
age                          int64
BMI                        float64
hypertensive                 int64
atrialfibrillation           int64
CHD_with_no_MI               int64
diabetes                     int64
deficiencyanemias            int64
Renal_failure                int64
COPD                         int64
heart_rate                 float64
Systolic_blood_pressure    float64
Diastolic_blood_pressure   float64
Respiratory_rate           float64
temperature                float64
SP_O2                      float64
Urine_output               float64
hematocrit                 float64
RBC                        float64
MCH                        float64
MCHC                       float64
MCV                        float64
Leucocyte                  float64
Platelets                  float64
Neutrophils                float64
Lymphocyte                 float64
INR                        float64
NT-proBNP                  float64
Creatinine                 float64
glucose                    float64
Magnesium_ion              float64
EF                           int64
Urine_output_BMI           float64
dtype: object
```

In [22]:
```
# Data does not have any missing values. All feature object types are ax expected.Theref
```

In [23]:
```
# Milestone #3
```

In [24]:
```
# As noted in Milestone 1, I plan to develop a model which will predict mortalality base
# admission of the patient (step 1) and also together with data which is generated typic
# tests done upon admission (step 2).Outcome is a binary variable; therefore, I will be
```

In [32]:
```
# Importing the train_test_split Function
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
```

In [33]:
```
# step 1
x_1 = df[['age', 'BMI', 'hypertensive', 'atrialfibrillation', 'CHD_with_no_MI', 'diabete
        'Renal_failure', 'COPD', 'heart_rate', 'Systolic_blood_pressure', 'Diastolic_blo
        'temperature', 'SP_O2']]
y_1 = df['outcome']
```

In [34]:
```
x_1_train, x_1_test, y_1_train, y_1_test = train_test_split(x_1,y_1,test_size=0.2)
```

In [35]:
```
#we use the standard scaler function to scale the values into a common range.Then we bui
scaler = StandardScaler()
lr = LogisticRegression()
scaler.fit(x_1_train)
x_1_train_scaler = scaler.transform(x_1_train)
x_1_test_scaler = scaler.transform(x_1_test)
```

In [36]:
```
# Train Logistic Regression on the training data
lr.fit(x_1_train_scaler,y_1_train)
```

Out[36]:
```
LogisticRegression()
```

In [37]:
```
#Evaluationg the model using accuracy score to check accuracy and overfitting.
```

In [38]:
```
x_1_test_pred = lr.predict(x_1_test_scaler)
accuracy = accuracy_score( y_1_test, x_1_test_pred)
accuracy
```

Out[38]:
```
0.8813559322033898
```

In [39]:
```
# Accuracy is 88%
```

In [40]:
```
#Accuracy on model data
```

In [41]:
```
x_1_train_pred =lr.predict(x_1_train_scaler)
accuracy = accuracy_score( y_1_train, x_1_train_pred)
accuracy
```

Out[41]:
```
0.8629117959617428
```

In [42]:
```
#Accuracy is similar to that calculated above. Therefore no overfitting.
```

In [43]:
```
# Step 2 with all variables
x = df.loc[:,df.columns != 'outcome']
y = df['outcome']
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2)
```

In [44]:
```
scaler.fit(x_train)
```

```
x_train_scaler = scaler.transform(x_train)
x_test_scaler = scaler.transform(x_test)
```

In [45]:
```
lr.fit(x_train_scaler,y_train)
```

Out[45]:
```
LogisticRegression()
```

In [46]:
```
x_test_pred = lr.predict(x_test_scaler)
accuracy = accuracy_score( y_test, x_test_pred)
accuracy
```

Out[46]:
```
0.8983050847457628
```

In [47]:
```
# Therefore, we see that even with all variables, the accuracy score is similar to that
```

In [48]:
```
x_train_pred = lr.predict(x_train_scaler)
accuracy = accuracy_score( y_train, x_train_pred)
accuracy
```

Out[48]:
```
0.8831030818278427
```

In [49]:
```
# No overfitting in this case as well.
```

In [50]:
```
# As noted above, the model predicts the outcome even with the basic health condition da
# Therefore, hospitals can decide to admit the patient based on data obained during admi
```