Predicting Survival of Patient

@Author :- Priyangshu Sarkar

Import Libraries

```
!pip install -q shap

          |████████████████████████████████| 564 kB 5.3 MB/s
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import joblib

import plotly.express as px
import plotly.offline as py
import plotly.graph_objs as go
import plotly.tools as tls
import plotly.figure_factory as ff

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import (accuracy_score,
                             classification_report,
                             roc_auc_score, roc_curve, auc, precision_recall_curve,
                             confusion_matrix)

from xgboost import XGBClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import StratifiedKFold, KFold

from tqdm import tqdm
import warnings
warnings.filterwarnings('ignore')

import shap

pd.set_option('display.max_rows', 250)
```

Importing Data

```
df=pd.read_csv("Dataset_Patient.csv")
```

```
df
```

	encounter_id	patient_id	hospital_id	hospital_death	age	bmi	electi
0	66154	25312	118	0	68.0	22.730000	
1	114252	59342	81	0	77.0	27.420000	
2	119783	50777	118	0	25.0	31.950000	
3	79267	46918	118	0	81.0	22.640000	
4	92056	34377	33	0	19.0	NaN	
...	...	...	...	...	...	...	
45820	90508	19814	21	0	86.0	24.136910	
45821	64010	80420	21	0	55.0	27.913563	
45822	101166	124288	21	0	59.0	42.581313	
45823	27015	41654	21	0	54.0	42.850872	
45824	124157	60471	21	0	17.0	22.606103	

45825 rows × 186 columns

Exploratory Data Analysis (EDA)

```
df.shape
```

```
(45825, 186)
```

```
df.info(verbose=True, null_counts=True)
```

```
97   h1_temp_min                     38304 non-null  float64
98   d1_albumin_max                  20552 non-null  float64
99   d1_albumin_min                  20552 non-null  float64
100  d1_bilirubin_max                19456 non-null  float64
101  d1_bilirubin_min                19456 non-null  float64
102  d1_bun_max                      40240 non-null  float64
103  d1_bun_min                      40240 non-null  float64
104  d1_calcium_max                  39629 non-null  float64
105  d1_calcium_min                  39629 non-null  float64
106  d1_creatinine_max               40582 non-null  float64
107  d1_creatinine_min               40582 non-null  float64
108  d1_glucose_max                  43288 non-null  float64
109  d1_glucose_min                  43288 non-null  float64
110  d1_hco3_max                     39775 non-null  float64
111  d1_hco3_min                     39775 non-null  float64
112  d1_hemaglobin_max               39630 non-null  float64
113  d1_hemaglobin_min               39630 non-null  float64
114  d1_hematocrit_max               40120 non-null  float64
```

```
 115  d1_hematocrit_min              40120 non-null  float64
 116  d1_inr_max                     17956 non-null  float64
 117  d1_inr_min                     17956 non-null  float64
 118  d1_lactate_max                 12509 non-null  float64
 119  d1_lactate_min                 12509 non-null  float64
 120  d1_platelets_max               38958 non-null  float64
 121  d1_platelets_min               38958 non-null  float64
 122  d1_potassium_max               40978 non-null  float64
 123  d1_potassium_min               40978 non-null  float64
 124  d1_sodium_max                  40371 non-null  float64
 125  d1_sodium_min                  40371 non-null  float64
 126  d1_wbc_max                     39325 non-null  float64
 127  d1_wbc_min                     39325 non-null  float64
 128  h1_albumin_max                 3671 non-null   float64
 129  h1_albumin_min                 3671 non-null   float64
 130  h1_bilirubin_max               3451 non-null   float64
 131  h1_bilirubin_min               3451 non-null   float64
 132  h1_bun_max                     8593 non-null   float64
 133  h1_bun_min                     8593 non-null   float64
 134  h1_calcium_max                 8149 non-null   float64
 135  h1_calcium_min                 8149 non-null   float64
 136  h1_creatinine_max              8625 non-null   float64
 137  h1_creatinine_min              8625 non-null   float64
 138  h1_glucose_max                 22652 non-null  float64
 139  h1_glucose_min                 22652 non-null  float64
 140  h1_hco3_max                    8449 non-null   float64
 141  h1_hco3_min                    8449 non-null   float64
 142  h1_hemaglobin_max              9955 non-null   float64
 143  h1_hemaglobin_min              9955 non-null   float64
 144  h1_hematocrit_max              9783 non-null   float64
 145  h1_hematocrit_min              9783 non-null   float64
 146  h1_inr_max                     17956 non-null  float64
 147  h1_inr_min                     17956 non-null  float64
 148  h1_lactate_max                 3994 non-null   float64
 149  h1_lactate_min                 3994 non-null   float64
 150  h1_platelets_max               8510 non-null   float64
 151  h1_platelets_min               8510 non-null   float64
 152  h1_potassium_max               10183 non-null  float64
 153  h1_potassium_min               10183 non-null  float64
 154  h1_sodium_max                  9756 non-null   float64
 155  h1_sodium_min                  9756 non-null   float64
```

```
df.describe()
```

	encounter_id	patient_id	hospital_id	hospital_death	age	
count	45825.000000	45825.000000	45825.000000	45825.000000	43665.000000	4335
mean	65625.932875	65557.129624	106.995985	0.086219	62.651804	2
std	37774.188036	37832.562156	49.579404	0.280691	16.610883	

```
df.isnull().sum(axis=0).sort_values(ascending=False)
```

```
h1_potassium_min              35642
h1_potassium_max              35642
paco2_apache                  34074
paco2_for_ph_apache           34074
pao2_apache                   34074
ph_apache                     34074
fio2_apache                   34074
d1_lactate_max                33316
d1_lactate_min                33316
d1_pao2fio2ratio_min          31707
d1_pao2fio2ratio_max          31707

d1_diasbp_invasive_max        31596
d1_diasbp_invasive_min        31596
d1_sysbp_invasive_max         31580
d1_sysbp_invasive_min         31580
d1_mbp_invasive_max           31438
d1_mbp_invasive_min           31438
bilirubin_apache              28404
d1_arterial_po2_min           28366
d1_arterial_po2_max           28366
d1_arterial_pco2_max          28346
d1_arterial_pco2_min          28346
d1_arterial_ph_max            28336
d1_arterial_ph_min            28336
h1_inr_min                    27869
h1_inr_max                    27869
d1_inr_min                    27869
d1_inr_max                    27869
albumin_apache                27571
d1_bilirubin_min              26369
d1_bilirubin_max              26369
d1_albumin_max                25273
d1_albumin_min                25273
h1_glucose_min                23173
h1_glucose_max                23173
urineoutput_apache            21481
wbc_apache                    10403
bun_apache                     9537
hematocrit_apache              9374
creatinine_apache              9179
sodium_apache                  9161
h1_temp_min                    7521
h1_temp_max                    7521
hospital_admit_source          6954
d1_platelets_min               6867
d1_platelets_max               6867
d1_wbc_min                     6500
d1_wbc_max                     6500
d1_calcium_min                 6196
```

```
     d1_calcium_max                     6196
     d1_hemaglobin_max                  6195
     d1_hemaglobin_min                  6195
     d1_hco3_min                        6050
     d1_hco3_max                        6050
     d1_hematocrit_max                  5705
     d1_hematocrit_min                  5705
     d1_bun_min                         5585
     d1 bun max                         5585
```

columns that can be dropped: **'encounter_id', 'hospital_admit_source', 'icu_admit_source', 'icu_id', 'icu_stay_type', 'patient_id', 'hospital_id', 'readmission_status'**

Notice the amount of missing values in each row We decide a threshold value to delete some of the attributes from the dataset (25k in this approach) **74 columns** will get deleted in this turn.

```
print("Number of rows with missing values:", df.isnull().any(axis=1).sum())
```

```
     Number of rows with missing values: 45803
```

```
large_missing = df.isnull().sum(axis=0).sort_values(ascending=False)[df.isnull().sum(axis=

print("\nTotal features with more than", 25000, "missing values:", len(large_missing))

df.drop(large_missing.index.tolist() + ['encounter_id', 'hospital_admit_source', 'icu_admi
              axis=1,
             inplace = True)
df
```

```
Total features with more than 25000 missing values: 71
```

	hospital_death	age	bmi	elective_surgery	ethnicity	gender	height
0	0	68.0	22.730000	0	Caucasian	M	180.3
1	0	77.0	27.420000	0	Caucasian	F	160.0

```
df.nunique()
```

```
hospital_death                     2
age                               74
bmi                            23169
elective_surgery                   2
ethnicity                          6
gender                             2
height                           352
icu_type                           8
pre_icu_los_days                7181
weight                          2566
apache_2_diagnosis                44
apache_3j_diagnosis              387
apache_post_operative              2
arf_apache                         2
bun_apache                       127
creatinine_apache               1037
gcs_eyes_apache                    4
gcs_motor_apache                   6
gcs_unable_apache                  2
gcs_verbal_apache                  5
glucose_apache                   548
heart_rate_apache                149
hematocrit_apache                353
intubated_apache                   2
map_apache                       161
resprate_apache                   57
sodium_apache                    119
temp_apache                      124
urineoutput_apache             16703
ventilated_apache                  2
wbc_apache                      2154
d1_diasbp_max                    120
d1_diasbp_min                     78
d1_diasbp_noninvasive_max        120
d1_diasbp_noninvasive_min         78
d1_heartrate_max                 120
d1_heartrate_min                 150
d1_mbp_max                       125
d1_mbp_min                        91
d1_mbp_noninvasive_max           122
d1_mbp_noninvasive_min            91
d1_resprate_max                   79
d1_resprate_min                   49
d1_spo2_max                       31
d1_spo2_min                      101
d1_sysbp_max                     143
d1_sysbp_min                     120
d1_sysbp_noninvasive_max         143
d1_sysbp_noninvasive_min         120
```

```
d1_temp_max                    115
d1_temp_min                    115
h1_diasbp_max                  107
h1_diasbp_min                   92
h1_diasbp_noninvasive_max      108
h1_diasbp_noninvasive_min       93
h1_heartrate_max               119
h1_heartrate_min               109
```

Removing missing values from some of the inter-related columns (bmi, weight and height)

We cut down almost 3000 instances with this process safely

```
df = df[df[['bmi', 'weight', 'height']].isna().sum(axis=1) == 0]
df
```

	hospital_death	age	bmi	elective_surgery	ethnicity	gender	height
0	0	68.0	22.730000	0	Caucasian	M	180.3
1	0	77.0	27.420000	0	Caucasian	F	160.0
2	0	25.0	31.950000	0	Caucasian	F	172.7
3	0	81.0	22.640000	1	Caucasian	F	165.1
5	0	67.0	27.560000	0	Caucasian	M	190.5
...	...	...	...	...	...	...	...
45819	0	87.0	35.133404	0	Caucasian	F	152.4
45820	0	86.0	24.136910	0	Hispanic	M	167.6
45821	0	55.0	27.913563	1	Caucasian	F	162.6
45822	0	59.0	42.581313	0	Caucasian	M	172.7
45823	0	54.0	42.850872	0	Caucasian	M	180.3

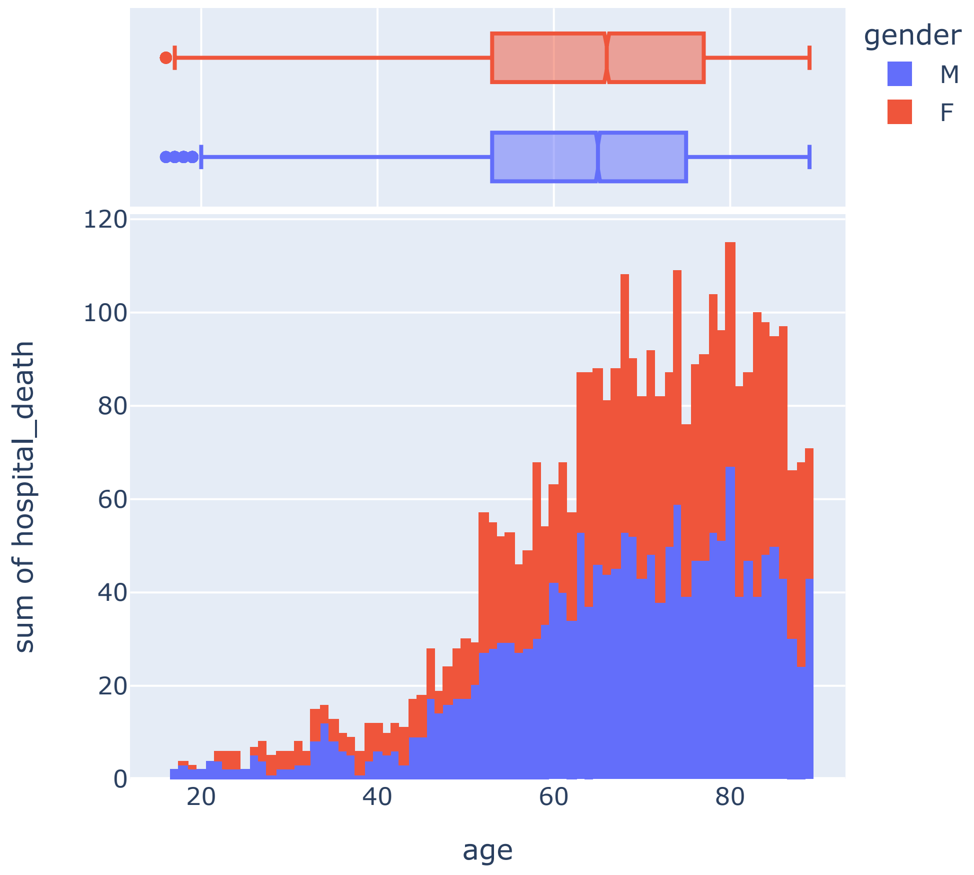
43355 rows × 107 columns

Univariate-Multivariate Analysis

Individual plots seldom doesn't help in large datasets, in this approach let's look at the variation of instances according to each context of the column

The death rate for Male-Female patients is shown below. While the rate of **female deaths** are higher the youngest person to pass away during one of the case was a **Male** of **16yrs** of age

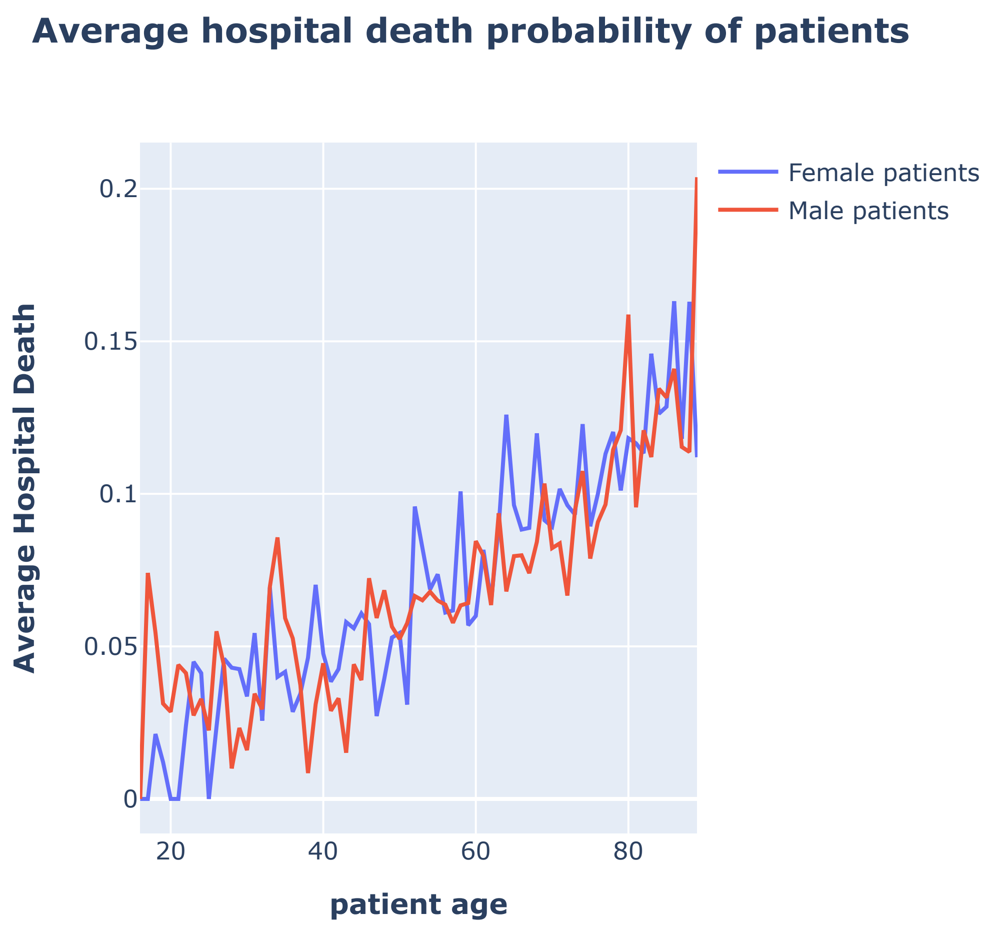
```
fig = px.histogram(df[['age','gender','hospital_death','bmi']].dropna(), x="age", y="hospi
                   marginal="box", # or violin, rug
                   hover_data=df[['age','gender','hospital_death','bmi']].columns)
fig.show()
```



Average hospital death probability of patients

based on age and gender

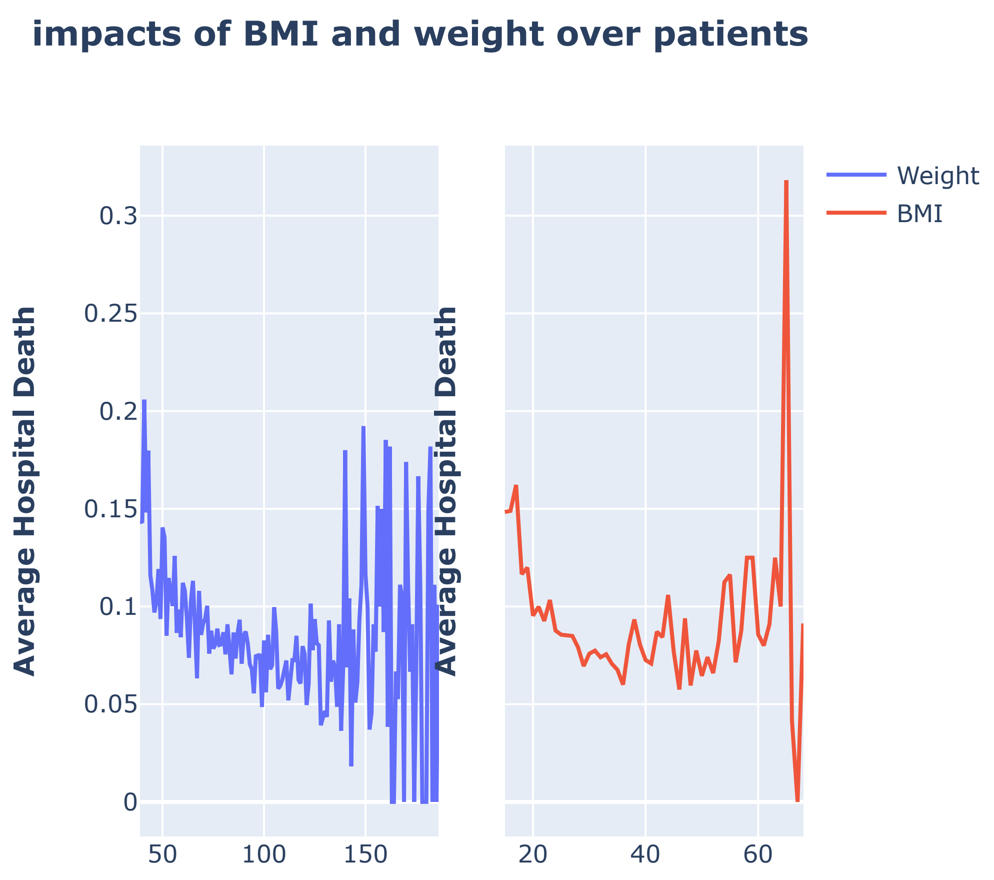
```
age_death_F=df[df['gender']=='F'][['age','hospital_death']].groupby('age').mean().reset_ir
age_death_M=df[df['gender']=='M'][['age','hospital_death']].groupby('age').mean().reset_ir
from plotly.subplots import make_subplots
fig = make_subplots()
fig.add_trace(
    go.Scatter(x=age_death_F['age'], y=age_death_F['hospital_death'], name="Female patient
fig.add_trace(
    go.Scatter(x=age_death_M['age'], y=age_death_M['hospital_death'],name="Male patients")
fig.update_layout(
    title_text="<b>Average hospital death probability of patients<b>")
fig.update_xaxes(title_text="<b>patient age<b>")
fig.update_yaxes(title_text="<b>Average Hospital Death</b>", secondary_y=False)
fig.show()
```



impacts of BMI and weight over patients

```
weight_df=df[['weight','hospital_death','bmi']]
weight_df['weight']=weight_df['weight'].round(0)
weight_df['bmi']=weight_df['bmi'].round(0)
weight_death=weight_df[['weight','hospital_death']].groupby('weight').mean().reset_index(
bmi_death=weight_df[['bmi','hospital_death']].groupby('bmi').mean().reset_index()
fig = make_subplots(rows=1, cols=2, shared_yaxes=True)
fig.add_trace(
    go.Scatter(x=weight_death['weight'], y=weight_death['hospital_death'], name="Weight"),
   row=1, col=1
)
fig.add_trace(
    go.Scatter(x=bmi_death['bmi'], y=bmi_death['hospital_death'], name="BMI"),
    row=1, col=2
)
fig.update_layout(
    title_text="<b>impacts of BMI and weight over patients<b>"
)
```

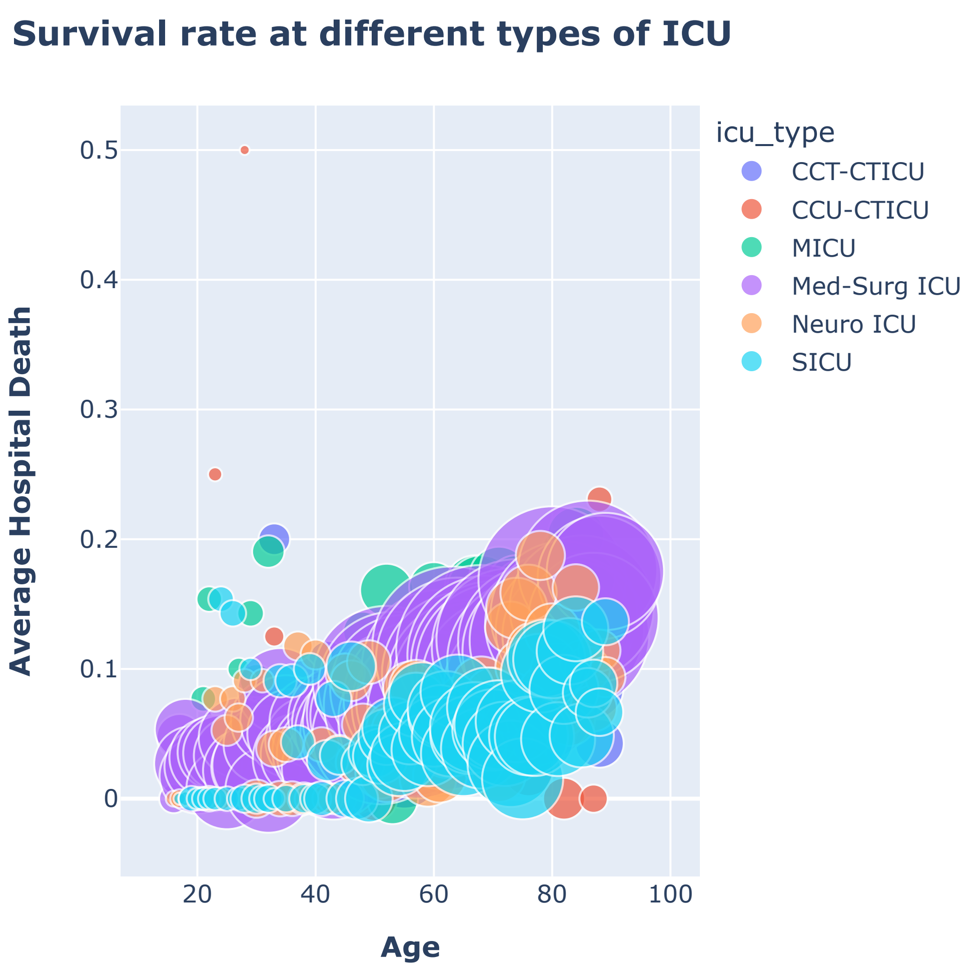
```
fig.update_yaxes(title_text="<b>Average Hospital Death")
fig.show()
```



Survival rate at different types of ICU

some of the ICU wards have higher death probability pertaining to being surgical wards

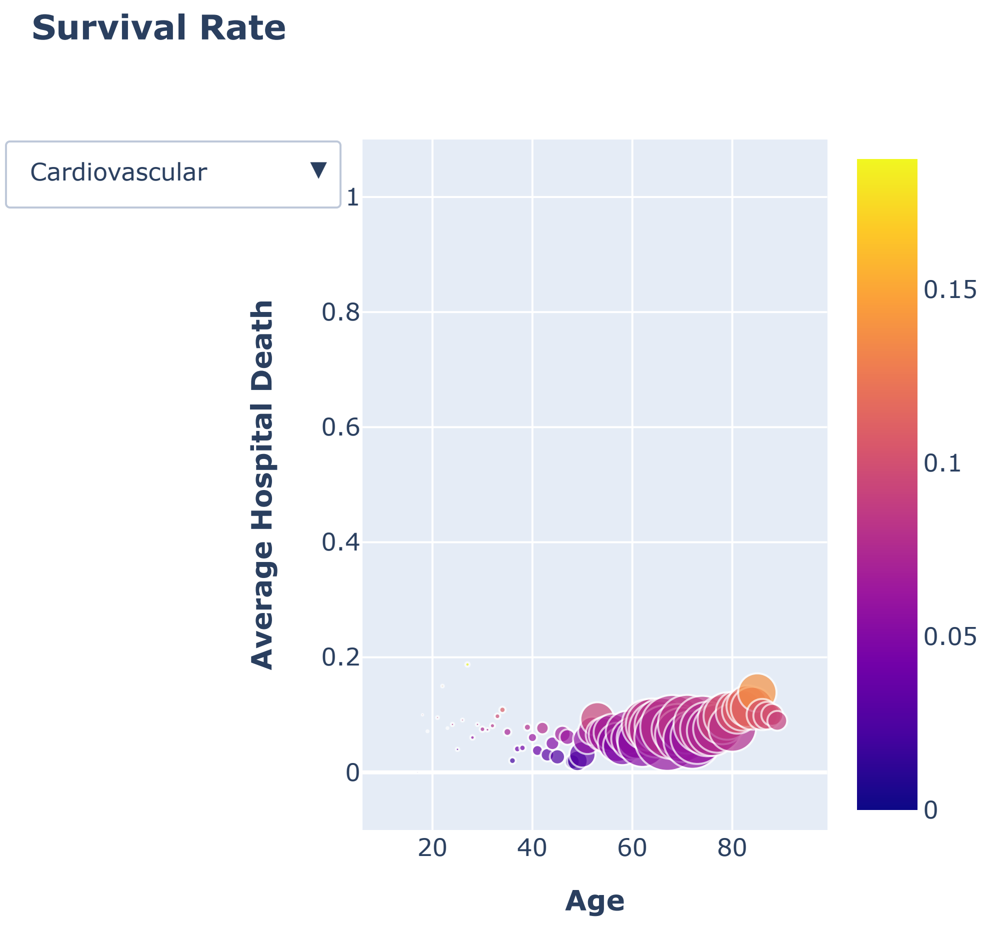
```
ICU_type=df[['icu_type','age','hospital_death']]
ICU_type['icu_type']=ICU_type['icu_type'].replace({'CTICU':'CCU-CTICU',
                                               'Cardiac ICU':'CCT-CTICU',
                                               'CTICU':'CCT-CTICU',
                                               'CSICU':'SICU'})
#ICU_type['pre_icu_los_days']=ICU_type['pre_icu_los_days'].round(0)
ICU_df=ICU_type.groupby(['icu_type','age']).mean().reset_index()
ICU_df['count']=ICU_type.groupby(['icu_type','age']).count().reset_index()['hospital_deatl

fig = px.scatter(ICU_df, x="age", y="hospital_death", size="count", color="icu_type",
           hover_name="icu_type", log_x=False, size_max=60,)
fig.update_layout(
    title_text="<b>Survival rate at different types of ICU<b>"
)
fig.update_yaxes(title_text="<b>Average Hospital Death<b>")
fig.update_xaxes(title_text="<b>Age<b>")
fig.show()
```



Hospital Death Rate, by age and Medical condition

```
apache3=df[['age','apache_3j_bodysystem','hospital_death']]
apache3=apache3.groupby(['apache_3j_bodysystem','age']).agg(['size','mean']).reset_index(

apache3['size']=apache3['hospital_death']['size']
apache3['mean']=apache3['hospital_death']['mean']

apache3.drop('hospital_death',axis=1,inplace=True)

systems =list(apache3['apache_3j_bodysystem'].unique())
data = []
list_updatemenus = []
for n, s in enumerate(systems):
    visible = [False] * len(systems)
    visible[n] = True
    temp_dict = dict(label = str(s),
                 method = 'update',
                 args = [{'visible': visible},
                         {'title': '<b>'+s+'<b>'}])
    list_updatemenus.append(temp_dict)
```

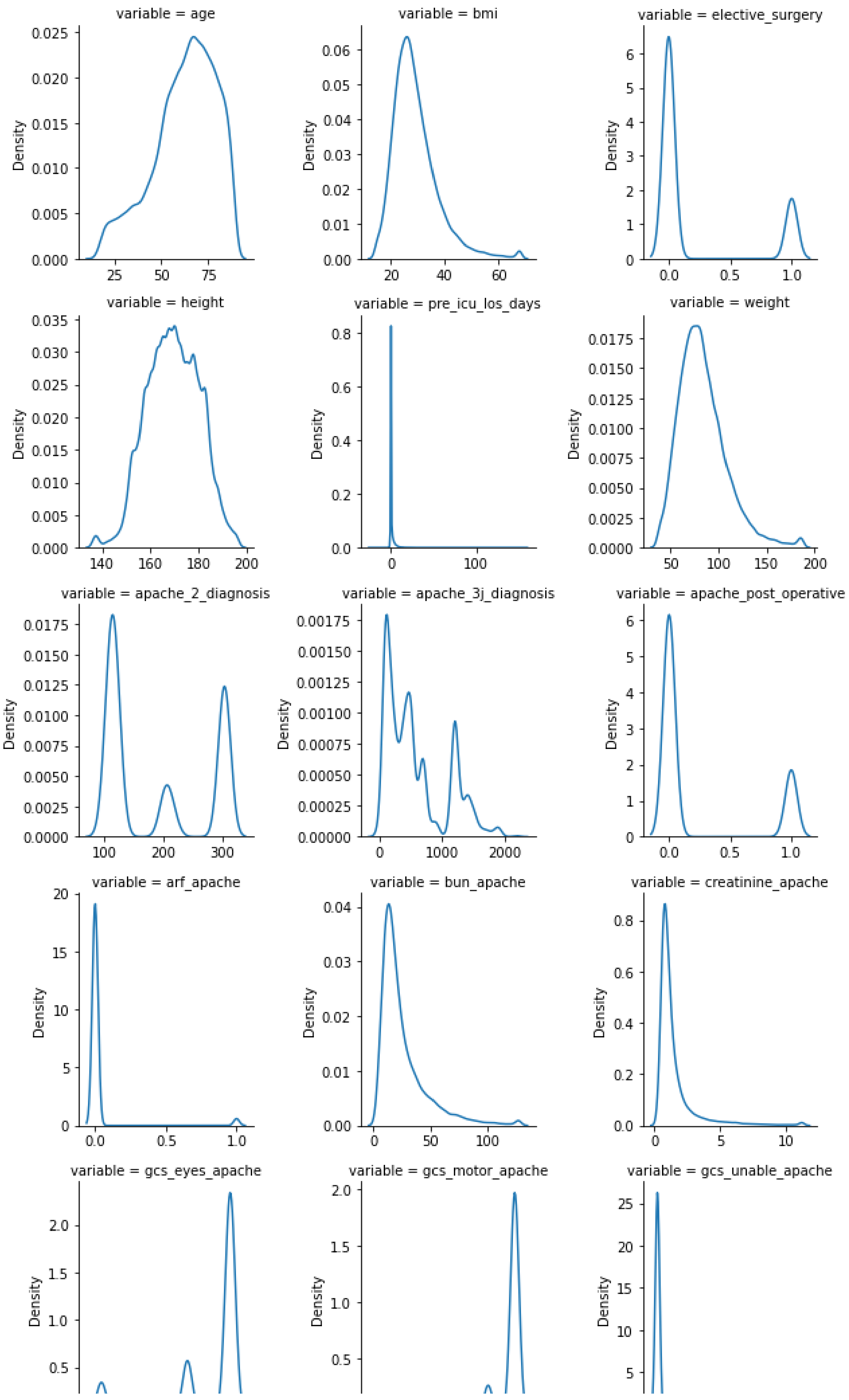
```
for s in systems:
    mask = (apache3['apache_3j_bodysystem'].values == s)
    trace = (dict(visible = False,
        x = apache3.loc[mask, 'age'],
        y = apache3.loc[mask, 'mean'],
        mode = 'markers',
        marker = {'size':apache3.loc[mask, 'size']/apache3.loc[mask,'size'].sum()*1000,
                 'color':apache3.loc[mask, 'mean'],
                 'showscale': True})
                   )
    data.append(trace)

data[0]['visible'] = True

layout = dict(updatemenus=list([dict(buttons= list_updatemenus)]),
              xaxis=dict(title = '<b>Age<b>', range=[min(apache3.loc[:, 'age'])-10, max(ap
              yaxis=dict(title = '<b>Average Hospital Death<b>', range=[min(apache3.loc[:,
              title='<b>Survival Rate<b>' )
fig = dict(data=data, layout=layout)
py.iplot(fig, filename='update_dropdown')
```

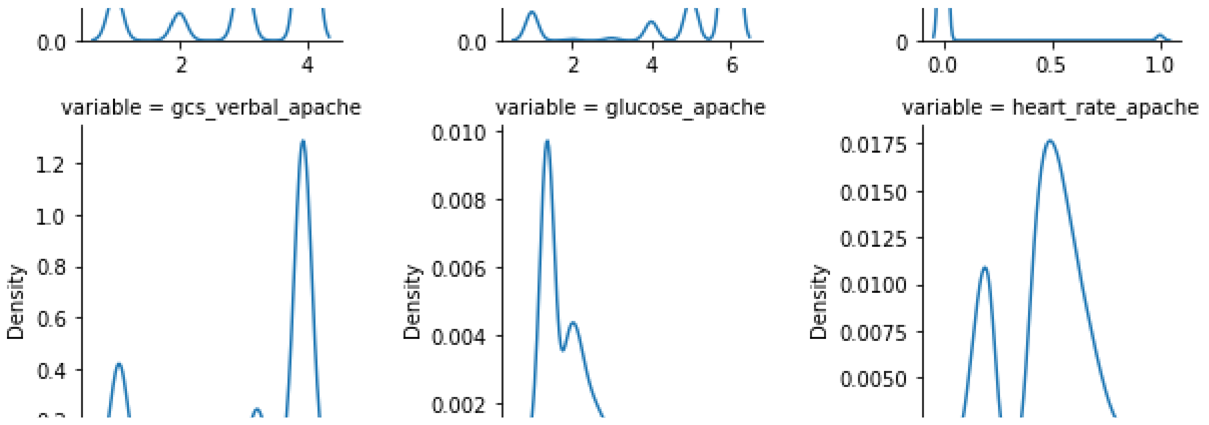


Density Distribution for numerical columns

```
[illegible]
```

```
unpivot = pd.melt(df, df.describe().columns[0],·df.describe().columns[1:])

g·=·sns.FacetGrid(unpivot,·col="variable",·col_wrap=3,·sharex=False,·sharey=False)
g.map(sns.kdeplot,·"value")

plt.show()
```


variable = age
variable = bmi
variable = elective_surgery
variable = height
variable = pre_icu_los_days
variable = weight
variable = apache_2_diagnosis
variable = apache_3j_diagnosis
variable = apache_post_operative
variable = arf_apache
variable = bun_apache
variable = creatinine_apache
variable = gcs_eyes_apache
variable = gcs_motor_apache
variable = gcs_unable_apache
Density


0.0
2
4
0.0
2
4
6
0
0.0
0.5
1.0
variable = gcs_verbal_apache
variable = glucose_apache
variable = heart_rate_apache
Density
1.2
1.0
0.8
0.6
0.4
0.010
0.008
0.006
0.004
0.002
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050

Preprocessing

converting categorical values tranforming numerical columns and removing nulls

Null values for categories are replaced by **mode**, and those for numerical are replaced by **mean**.

```
numerical_cat = [
 'elective_surgery',
 'apache_post_operative',
 'arf_apache',
 'gcs_unable_apache',
 'intubated_apache',
 'ventilated_apache',
 'aids',
 'cirrhosis',
 'diabetes_mellitus',
 'hepatic_failure',
 'immunosuppression',
 'leukemia',
 'lymphoma',
 'solid_tumor_with_metastasis']

categorical = ['ethnicity',
 'gender',
 'icu_type',
 'apache_3j_bodysystem',
 'apache_2_bodysystem']


df.nunique()[df.nunique() == 2].index.tolist()
```

```
     ['hospital_death',
      'elective_surgery',
      'gender',
      'apache_post_operative',
      'arf_apache',
      'gcs_unable_apache',
      'intubated_apache',
      'ventilated_apache',
      'aids',
      'cirrhosis',
      'diabetes_mellitus',
      'hepatic_failure',
      'immunosuppression',
      'leukemia',
      'lymphoma',
      'solid_tumor_with_metastasis']
```

```
df.select_dtypes(include='O').columns.values.tolist()
```

```
['ethnicity',
 'gender',
 'icu_type',
 'apache_3j_bodysystem',
 'apache_2_bodysystem']
```

```
not_numeric = df[numerical_cat + categorical + ['hospital_death']].columns.tolist()
numeric_only = df.drop(not_numeric,axis=1).columns.tolist()
numeric_only
```

```
 'd1_mbp_noninvasive_max',
 'd1_mbp_noninvasive_min',
 'd1_resprate_max',
 'd1_resprate_min',
 'd1_spo2_max',
 'd1_spo2_min',
 'd1_sysbp_max',
 'd1_sysbp_min',
 'd1_sysbp_noninvasive_max',
 'd1_sysbp_noninvasive_min',
 'd1_temp_max',
 'd1_temp_min',
 'h1_diasbp_max',
 'h1_diasbp_min',
 'h1_diasbp_noninvasive_max',
 'h1_diasbp_noninvasive_min',
 'h1_heartrate_max',
 'h1_heartrate_min',
 'h1_mbp_max',
 'h1_mbp_min',
 'h1_mbp_noninvasive_max',
 'h1_mbp_noninvasive_min',
 'h1_resprate_max',
 'h1_resprate_min',
 'h1_spo2_max',
 'h1_spo2_min',
 'h1_sysbp_max',
 'h1_sysbp_min',
 'h1_sysbp_noninvasive_max',
 'h1_sysbp_noninvasive_min',
 'h1_temp_max',
 'h1_temp_min',
 'd1_bun_max',
 'd1_bun_min',
 'd1_calcium_max',
 'd1_calcium_min',
 'd1_creatinine_max',
 'd1_creatinine_min',
 'd1_glucose_max',
 'd1_glucose_min',
 'd1_hco3_max',
 'd1_hco3_min',
 'd1_hemaglobin_max',
  'd1_hemaglobin_min',
 'd1_hematocrit_max',
 'd1_hematocrit_min',
 'd1_platelets_max',
 'd1_platelets_min',
 'd1_potassium_max',
 'd1_potassium_min',
 'd1_sodium_max',
 'd1_sodium_min',
 'd1_wbc_max',
 'd1_wbc_min',
 'h1_glucose_max',
 'h1_glucose_min',
 'apache_4a_hospital_death_prob',
 'apache_4a_icu_death_prob']
```

```
for col in numerical_cat:
  df[col] = df[col].astype('Int64')

for col in numerical_cat:
    df[col] = df[col].fillna(df[col].mode()[0])


df[numeric_only].isna().sum(axis=0).sort_values(ascending=False)
```

```
h1_glucose_min                   21686
h1_glucose_max                   21686
urineoutput_apache               19517
wbc_apache                        9812
bun_apache                        8887
hematocrit_apache                 8839
creatinine_apache                 8572
sodium_apache                     8518
h1_temp_max                       7057
h1_temp_min                       7057
d1_platelets_min                  6436
d1_platelets_max                  6436
d1_wbc_max                        6077
d1_wbc_min                        6077
d1_hemaglobin_min                 5800
d1_hemaglobin_max                 5800
d1_calcium_max                    5717
d1_calcium_min                    5717
d1_hco3_max                       5575
d1_hco3_min                       5575
d1_hematocrit_max                 5335
d1_hematocrit_min                 5335
d1_bun_max                        5134
d1_bun_min                        5134
h1_mbp_noninvasive_min            5044
h1_mbp_noninvasive_max            5044
d1_sodium_max                     5007
d1_sodium_min                     5007
d1_creatinine_min                 4835
d1_creatinine_max                 4835
apache_4a_hospital_death_prob     4574
apache_4a_icu_death_prob          4574
d1_potassium_max                  4511
d1_potassium_min                  4511
glucose_apache                    4482
```

```
h1_diasbp_noninvasive_min         3828
h1_diasbp_noninvasive_max         3828
h1_sysbp_noninvasive_min          3820
h1_sysbp_noninvasive_max          3820
d1_glucose_min                    2329
d1_glucose_max                    2329
age                               2045
h1_mbp_max                        1862
h1_mbp_min                        1862
h1_spo2_max                       1656
h1_spo2_min                       1656
h1_resprate_max                   1562
h1_resprate_min                   1562
gcs_eyes_apache                   1218
gcs_motor_apache                  1218
gcs_verbal_apache                 1218
h1_diasbp_max                     1122
h1_diasbp_min                     1122
h1_sysbp_max                      1115
h1_sysbp_min                      1115
d1_mbp_noninvasive_max            1011
d1_mbp_noninvasive_min            1011
temp_apache                        911
```

```
split_one = df[numeric_only].isna().sum(axis=0).sort_values()[df[numeric_only].isna().sum(
split_two = df[numeric_only].isna().sum(axis=0).sort_values()[df[numeric_only].isna().sum(
```

```
split_two
```

```
['urineoutput_apache', 'h1_glucose_max', 'h1_glucose_min']
```

```
for col in split_two:
    df[col] = df[col].fillna(df[col].mean())

process_data = df.dropna(axis=0)
```

```
process_data[categorical].nunique()
```

```
ethnicity                 6
gender                    2
icu_type                  8
apache_3j_bodysystem     11
apache_2_bodysystem      10
dtype: int64
```

```
icu_data = pd.get_dummies(process_data,
    prefix='isin',
    prefix_sep='_',
    columns=categorical,
    drop_first=False)
icu_data.reset_index(drop = True, inplace = True)
icu_data
```

	hospital_death	age	bmi	elective_surgery	height	pre_icu_los_days
0	0	68.0	22.730000	0	180.30	0.541667
1	0	77.0	27.420000	0	160.00	0.927778
2	0	67.0	27.560000	0	190.50	0.000694
3	0	46.0	25.845717	0	167.60	0.000000
4	0	87.0	21.963763	0	180.30	5.046528
...	...	...	...	...	...	...
18771	0	74.0	26.096991	0	177.80	0.115278
18772	0	79.0	23.159226	0	162.56	0.059028
18773	0	85.0	18.943655	0	172.70	0.083333
18774	0	71.0	23.250046	0	177.80	0.195833
18775	0	59.0	42.581313	0	172.70	0.060417

18776 rows × 139 columns

```
icu_data.columns = [x.lower() for x in icu_data.columns.tolist()]
icu_data = icu_data.loc[:,~icu_data.columns.duplicated()]
```

```
t = icu_data['arf_apache'].dtype
for col in tqdm(icu_data.columns.tolist()):
    if icu_data[col].values.dtype == 'uint8' or t == icu_data[col].values.dtype:
        icu_data[col] = icu_data[col].astype(int)
```

```
100%|██████████| 133/133 [00:00<00:00, 4166.11it/s]
```

```
icu_data.dtypes
```

```
hospital_death                   int64
age                            float64
bmi                            float64
elective_surgery                 int64
height                         float64
pre_icu_los_days               float64
weight                         float64
apache_2_diagnosis             float64
apache_3j_diagnosis            float64
apache_post_operative            int64
arf_apache                       int64
bun_apache                     float64
creatinine_apache              float64
gcs_eyes_apache                float64
gcs_motor_apache               float64
gcs_unable_apache                int64
gcs_verbal_apache              float64
```

```
glucose_apache                 float64
heart_rate_apache              float64
hematocrit_apache              float64
intubated_apache                 int64
map_apache                     float64
resprate_apache                float64
sodium_apache                  float64
temp_apache                    float64
urineoutput_apache             float64
ventilated_apache                int64
wbc_apache                     float64
d1_diasbp_max                  float64
d1_diasbp_min                  float64
d1_diasbp_noninvasive_max      float64
d1_diasbp_noninvasive_min      float64
d1_heartrate_max               float64
d1_heartrate_min               float64
d1_mbp_max                     float64
d1_mbp_min                     float64
d1_mbp_noninvasive_max         float64
d1_mbp_noninvasive_min         float64
d1_resprate_max                float64
d1_resprate_min                float64
d1_spo2_max                    float64
d1_spo2_min                    float64
d1_sysbp_max                   float64
d1_sysbp_min                   float64
d1_sysbp_noninvasive_max       float64
d1_sysbp_noninvasive_min       float64
d1_temp_max                    float64
d1_temp_min                    float64
h1_diasbp_max                  float64
h1_diasbp_min                  float64
h1_diasbp_noninvasive_max      float64
h1_diasbp_noninvasive_min      float64
h1_heartrate_max               float64
h1_heartrate_min               float64
h1_mbp_max                     float64
h1_mbp_min                     float64
h1_mbp_noninvasive_max         float64
```

Modelling

```
X = icu_data.drop(['hospital_death'], axis=1)
y = icu_data['hospital_death']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,
                                                    random_state=11,
                                                    stratify = y)
```

```
icu_data.to_csv('/content/Dataset_Patient.csv', index=False)
X_test.to_csv("/content/Dataset_Patient.csv", index=False)
```

```
y_train.value_counts()
```

```
     0    11922
     1     1221
     Name: hospital_death, dtype: int64
```

```
y_test.value_counts()
```

```
     0    5110
     1     523
     Name: hospital_death, dtype: int64
```

```
def modelling(X_train, y_train, X_test, y_test, **kwargs):
    scores = {}
    models = []
    if 'xgb' in kwargs.keys() and kwargs['xgb']:
        xgb = XGBClassifier()
        xgb.fit(X_train._get_numeric_data(), np.ravel(y_train, order='C'))
        y_pred = xgb.predict(X_test._get_numeric_data())
        scores['xgb']= [accuracy_score(y_test, y_pred), roc_auc_score(y_test, y_pred)]
#         scores['xgb']['roc_auc'] = roc_auc_score(y_test, y_pred)

    if 'rf' in kwargs.keys() and kwargs['rf']:
        rf = RandomForestClassifier(n_estimators=200)
        rf.fit(X_train, y_train)
        y_pred = rf.predict(X_test)
        scores['rf']= [accuracy_score(y_test, y_pred), roc_auc_score(y_test, y_pred)]
#         scores['rf']['roc_auc'] = roc_auc_score(y_test, y_pred)
        models.append(rf)

    if 'extree' in kwargs.keys() and kwargs['extree']:
        extree = ExtraTreesClassifier()
        extree.fit(X_train, y_train)
        y_pred = extree.predict(X_test)
        scores['extree'] = [accuracy_score(y_test, y_pred), roc_auc_score(y_test, y_pred)
#         scores['extree']['roc_auc'] = roc_auc_score(y_test, y_pred)
        models.append(extree)

    return scores
```

```
modelling(X_train,y_train, X_test, y_test, xgb=True, rf=True, extree=True)
```

```
     {'extree': [0.9227303446722235, 0.6084453975078975],
      'rf': [0.9249831043027709, 0.6350888225888226],
      'xgb': [0.9247578283397162, 0.6482187341562342]}
```

```
def model_performance(model, y_test, y_hat) :
    conf_matrix = confusion_matrix(y_test, y_hat)
    trace1 = go.Heatmap(z = conf_matrix  ,x = ["0 (pred)","1 (pred)"],
                        y = ["0 (true)","1 (true)"],xgap = 2, ygap = 2,
                        colorscale = 'Viridis', showscale  = False)

    #Show metrics
    tp = conf_matrix[1,1]
    fn = conf_matrix[1,0]
    fp = conf_matrix[0,1]
    tn = conf_matrix[0,0]
    Accuracy  =  ((tp+tn)/(tp+tn+fp+fn))
    Precision =  (tp/(tp+fp))
    Recall    =  (tp/(tp+fn))
    F1_score  =  (2*(((tp/(tp+fp))*(tp/(tp+fn)))/((tp/(tp+fp))+(tp/(tp+fn)))))

    show_metrics = pd.DataFrame(data=[[Accuracy , Precision, Recall, F1_score]])
    show_metrics = show_metrics.T

    colors = ['gold', 'lightgreen', 'lightcoral', 'lightskyblue']
    trace2 = go.Bar(x = (show_metrics[0].values),
                   y = ['Accuracy', 'Precision', 'Recall', 'F1_score'], text = np.round_(s
                    textposition = 'auto',
                   orientation = 'h', opacity = 0.8,marker=dict(
            color=colors,
            line=dict(color='#000000',width=1.5)))

    #Roc curve
    model_roc_auc = round(roc_auc_score(y_test, y_hat) , 3)
    fpr, tpr, t = roc_curve(y_test, y_hat)
    trace3 = go.Scatter(x = fpr,y = tpr,
                        name = "Roc : " + str(model_roc_auc),
                        line = dict(color = ('rgb(22, 96, 167)'),width = 2), fill='tozeroy
    trace4 = go.Scatter(x = [0,1],y = [0,1],
                        line = dict(color = ('black'),width = 1.5,
                        dash = 'dot'))

    # Precision-recall curve
    precision, recall, thresholds = precision_recall_curve(y_test, y_hat)
    trace5 = go.Scatter(x = recall, y = precision,
                        name = "Precision" + str(precision),
                        line = dict(color = ('lightcoral'),width = 2), fill='tozeroy')

        #plots
    model = model

    #Subplots
    fig = tls.make_subplots(rows=2, cols=2, print_grid=False,
                          specs=[
#                                [{'colspan': 2}, None],
                                 [{}, {}],
                                 [{}, {}],

#                                   [{'colspan': 2}, None]
                                ],
                          subplot_titles=('Confusion Matrix',
                                          'Metrics',
                                          'ROC curve'+" "+ '('+ str(model_roc_auc)+')',
                                          'Precision - Recall curve',
                                          ))

    fig.append_trace(trace1,1,1)
    fig.append_trace(trace2,1,2)
    fig.append_trace(trace3,2,1)
    fig.append_trace(trace4,2,1)
    fig.append_trace(trace5,2,2)

    fig['layout'].update(showlegend = False, title = '<b>Model performance report</b><br>'
                        autosize = False, height = 1500,width = 830,
                        plot_bgcolor = 'rgba(240,240,240, 0.95)',
                        paper_bgcolor = 'rgba(240,240,240, 0.95)',
                        margin = dict(b = 195))
    fig["layout"]["xaxis2"].update((dict(range=[0, 1])))
    fig["layout"]["xaxis3"].update(dict(title = "false positive rate"))
    fig["layout"]["yaxis3"].update(dict(title = "true positive rate"))
    fig["layout"]["xaxis4"].update(dict(title = "recall"), range = [0,1.05])
    fig["layout"]["yaxis4"].update(dict(title = "precision"), range = [0,1.05])
    fig["layout"]["xaxis5"].update(dict(title = "Percentage contacted"))
    fig["layout"]["yaxis5"].update(dict(title = "Percentage positive targeted"))
    fig.layout.titlefont.size = 14

    py.iplot(fig)
```

Parameter Tuning

```
gkf = KFold(n_splits=3, shuffle=True, random_state=42).split(X=X_train, y=y_train)

fit_params_of_xgb = {
    "early_stopping_rounds":100,
    "eval_metric" : 'auc',
    "eval_set" : [(X_test, y_test)],
    'verbose': 100,
}


# A parameter grid for XGBoost
params = {
    'booster': ["gbtree"],
    'learning_rate': [0.1],
    'n_estimators': range(100, 500, 100),
    'min_child_weight': [1],
    'gamma': [0],
    'subsample': [0.8],
    'colsample_bytree': [0.8],
    'max_depth': [5],
    "scale_pos_weight": [1]
}

xgb_estimator = XGBClassifier(
    objective='binary:logistic',
    # silent=True,
)

gsearch = GridSearchCV(
```