

Faisal_Almokayed_assignment1

November 11, 2025

Practical_Machine_Learning_and_Data_Exploration 2025

Dr.Hajer Saleh

Faisal Ahmed Almokayed - 253001589

Practice on Prediction of Sepsis Dataset

```
[1]: from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
[3]: import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

Dataset Link: <https://www.kaggle.com/datasets/salikhussaini49/prediction-of-sepsis>

```
[4]: df = pd.read_csv('/content/drive/MyDrive/Practical ML Project1/Dataset.csv')
```

```
[5]: df.head()
```

```
[5]: Unnamed: 0  Hour    HR  O2Sat  Temp    SBP    MAP    DBP  Resp  EtCO2  ...  \
0            0      0   NaN     NaN   NaN   NaN   NaN   NaN   NaN   ...
1            1      1  65.0  100.0   NaN   NaN  72.0   NaN  16.5   NaN   ...
2            2      2  78.0  100.0   NaN   NaN  42.5   NaN   NaN   NaN   ...
3            3      3  73.0  100.0   NaN   NaN   NaN   NaN  17.0   NaN   ...
4            4      4  70.0  100.0   NaN  129.0  74.0  69.0  14.0   NaN   ...

      Fibrinogen  Platelets    Age  Gender  Unit1  Unit2  HospAdmTime  ICULOS  \
0            NaN         NaN  68.54    0.0   NaN   NaN         -0.02        1.0
1            NaN         NaN  68.54    0.0   NaN   NaN         -0.02        2.0
2            NaN         NaN  68.54    0.0   NaN   NaN         -0.02        3.0
3            NaN         NaN  68.54    0.0   NaN   NaN         -0.02        4.0
4            NaN        330.0  68.54    0.0   NaN   NaN         -0.02        5.0
```

	SepsisLabel	Patient_ID
0	0.0	17072.0
1	0.0	17072.0
2	0.0	17072.0
3	0.0	17072.0
4	0.0	17072.0

[5 rows x 44 columns]

[6]: df.dtypes

```
[6]: Unnamed: 0      int64
Hour              int64
HR               float64
O2Sat            float64
Temp             float64
SBP              float64
MAP              float64
DBP              float64
Resp             float64
EtCO2            float64
BaseExcess       float64
HCO3             float64
FiO2             float64
pH               float64
PaCO2            float64
SaO2             float64
AST              float64
BUN              float64
Alkalinephos     float64
Calcium          float64
Chloride         float64
Creatinine       float64
Bilirubin_direct float64
Glucose          float64
Lactate          float64
Magnesium        float64
Phosphate        float64
Potassium        float64
Bilirubin_total  float64
TroponinI        float64
Hct              float64
Hgb              float64
PTT              float64
WBC              float64
Fibrinogen       float64
Platelets        float64
```

```

Age                float64
Gender             float64
Unit1             float64
Unit2             float64
HospAdmTime       float64
ICULOS            float64
SepsisLabel       float64
Patient_ID        float64
dtype: object

```

```
[7]: df.columns
```

```

[7]: Index(['Unnamed: 0', 'Hour', 'HR', 'O2Sat', 'Temp', 'SBP', 'MAP', 'DBP',
          'Resp', 'EtCO2', 'BaseExcess', 'HCO3', 'FiO2', 'pH', 'PaCO2', 'SaO2',
          'AST', 'BUN', 'Alkalinephos', 'Calcium', 'Chloride', 'Creatinine',
          'Bilirubin_direct', 'Glucose', 'Lactate', 'Magnesium', 'Phosphate',
          'Potassium', 'Bilirubin_total', 'TroponinI', 'Hct', 'Hgb', 'PTT', 'WBC',
          'Fibrinogen', 'Platelets', 'Age', 'Gender', 'Unit1', 'Unit2',
          'HospAdmTime', 'ICULOS', 'SepsisLabel', 'Patient_ID'],
          dtype='object')

```

```
[8]: df.isnull().sum()
```

```

[8]: Unnamed: 0      0
     Hour           0
     HR          42123
     O2Sat         65804
     Temp        361612
     SBP           83203
     MAP          55596
     DBP          261452
     Resp          53448
     EtCO2         546123
     BaseExcess    489645
     HCO3          502248
     FiO2          469548
     pH           483938
     PaCO2         498561
     SaO2          519443
     AST           537944
     BUN           501673
     Alkalinephos  538173
     Calcium       518937
     Chloride      500815
     Creatinine    509911
     Bilirubin_direct 545305
     Glucose       479815

```

Lactate	527440
Magnesium	503732
Phosphate	518517
Potassium	487114
Bilirubin_total	539405
TroponinI	545435
Hct	481844
Hgb	498055
PTT	519684
WBC	505194
Fibrinogen	541959
Platelets	510532
Age	1
Gender	1
Unit1	271415
Unit2	271415
HospAdmTime	1
ICULOS	1
SepsisLabel	1
Patient_ID	1

dtype: int64

```
[9]: missing_percentage = df.isnull().mean() * 100
      # Filter columns with more than 33% missing values
      columns_with_missing_33 = missing_percentage[missing_percentage > 33]
      # Display the columns with more than 33% missing values
      print("Columns with more than 33% missing values:")
      print(columns_with_missing_33)
```

Columns with more than 33% missing values:

Temp	66.214388
DBP	47.874197
EtCO2	100.000000
BaseExcess	89.658374
HCO3	91.966096
FiO2	85.978433
pH	88.613371
PaCO2	91.290973
SaO2	95.114654
AST	98.502352
BUN	91.860808
Alkalinephos	98.544284
Calcium	95.022001
Chloride	91.703700
Creatinine	93.369259
Bilirubin_direct	99.850217
Glucose	87.858413

```

Lactate          96.578976
Magnesium        92.237829
Phosphate        94.945095
Potassium        89.194925
Bilirubin_total  98.769874
TroponinI        99.874021
Hct              88.229941
Hgb              91.198320
PTT              95.158783
WBC              92.505534
Fibrinogen       99.237534
Platelets        93.482970
Unit1            49.698511
Unit2            49.698511
dtype: float64

```

```

[10]: missing_percentage = df.isnull().mean() * 100
      threshold = 33
      # Filter columns with more than the threshold percentage of missing values
      columns_to_drop = missing_percentage[missing_percentage > threshold].index
      # Drop the columns
      df = df.drop(columns=columns_to_drop)
      # Display the resulting dataframe after removing columns with too many missing
      ↪values
      print(f"Columns dropped: {list(columns_to_drop)}")
      print(f"Remaining columns: {df.columns}")

```

```

Columns dropped: ['Temp', 'DBP', 'EtCO2', 'BaseExcess', 'HCO3', 'FiO2', 'pH',
'PaCO2', 'SaO2', 'AST', 'BUN', 'Alkalinephos', 'Calcium', 'Chloride',
'Creatinine', 'Bilirubin_direct', 'Glucose', 'Lactate', 'Magnesium',
'Phosphate', 'Potassium', 'Bilirubin_total', 'TroponinI', 'Hct', 'Hgb', 'PTT',
'WBC', 'Fibrinogen', 'Platelets', 'Unit1', 'Unit2']
Remaining columns: Index(['Unnamed: 0', 'Hour', 'HR', 'O2Sat', 'SBP', 'MAP',
'Resp', 'Age',
'Gender', 'HospAdmTime', 'ICULOS', 'SepsisLabel', 'Patient_ID'],
dtype='object')

```

```

[11]: #Drop Index column ['Unnamed: 0'] and one row where not include patient data
      df = df.dropna(subset=['SepsisLabel'])
      df = df.drop('Unnamed: 0', axis=1)

```

```

[12]: #re-check the missing values after drop columns contains above 33% of missing
      ↪values
      df.isnull().sum()

```

```

[12]: Hour          0
      HR          42123

```

```

O2Sat          65804
SBP            83203
MAP            55596
Resp           53448
Age             0
Gender          0
HospAdmTime    0
ICULOS         0
SepsisLabel    0
Patient_ID     0
dtype: int64

```

```
[13]: df.dtypes
```

```

[13]: Hour          int64
HR              float64
O2Sat           float64
SBP             float64
MAP             float64
Resp            float64
Age             float64
Gender          float64
HospAdmTime     float64
ICULOS          float64
SepsisLabel     float64
Patient_ID      float64
dtype: object

```

No need to RUN for **Pre-processing on Dataset** as all values is numerical data

```

from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
for column in df.select_dtypes(include=['object']).columns:

df[column] = label_encoder.fit_transform(df[column])
df.head()

```

```

[14]: data=df.copy()
def detect_outliers_iqr(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    # Return a boolean series indicating if the value is an outlier
    return (column < lower_bound) | (column > upper_bound)
# Apply the IQR method to each feature in the DataFrame
for col in data.columns:

```

```

data[f'{col}_Outlier'] = detect_outliers_iqr(data[col])
# Filter the DataFrame to display rows where any outlier flag is True
outliers_only = data[data.filter(like='_Outlier').any(axis=1)]
# Display the filtered DataFrame with outliers
outliers_only

```

```

[14]:
      Hour    HR  O2Sat    SBP    MAP  Resp    Age  Gender  HospAdmTime  \
67      8   77.0   98.0  125.0   77.67  33.0  85.57    0.0    -0.03
99      4  147.0   99.0    NaN    NaN   NaN   58.18    0.0    -2.43
103     8  127.0   97.0  155.0  135.00   NaN   58.18    0.0    -2.43
107    12  133.0  100.0  181.0  124.33   NaN   58.18    0.0    -2.43
122    27  114.0  100.0  170.0  124.00  26.0   58.18    0.0    -2.43
...
546117  22  100.0   93.0    NaN   84.00  21.0  70.93    1.0   -313.53
546118  23  141.0   95.0    NaN   79.00  23.0  70.93    1.0   -313.53
546119  24   94.0   98.0    NaN   89.00  21.0  70.93    1.0   -313.53
546120  25  104.0   94.0    NaN   78.00  24.0  70.93    1.0   -313.53
546121  26  104.0   94.0    NaN    NaN  22.0  70.93    1.0   -313.53

      ICULOS  ...  O2Sat_Outlier  SBP_Outlier  MAP_Outlier  Resp_Outlier  \
67      9.0  ...      False      False      False      True
99      5.0  ...      False      False      False      False
103     9.0  ...      False      False      True      False
107    13.0  ...      False      True      True      False
122    28.0  ...      False      False      True      False
...
546117  25.0  ...      False      False      False      False
546118  26.0  ...      False      False      False      False
546119  27.0  ...      False      False      False      False
546120  28.0  ...      False      False      False      False
546121  29.0  ...      False      False      False      False

      Age_Outlier  Gender_Outlier  HospAdmTime_Outlier  ICULOS_Outlier  \
67      False      False      False      False
99      False      False      False      False
103     False      False      False      False
107     False      False      False      False
122     False      False      False      False
...
546117  False      False      True      False
546118  False      False      True      False
546119  False      False      True      False
546120  False      False      True      False
546121  False      False      True      False

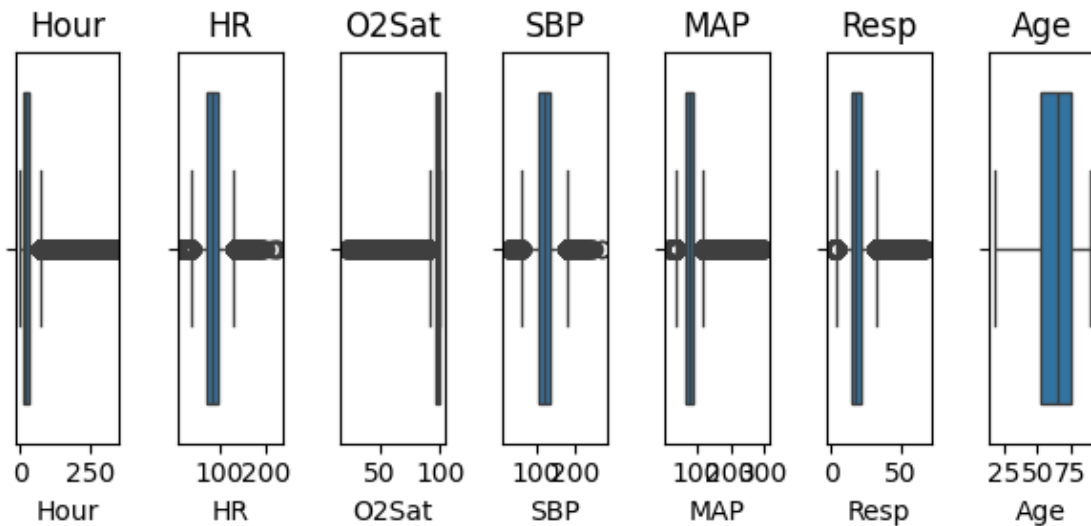
      SepsisLabel_Outlier  Patient_ID_Outlier
67      False      False

```

99	False	False
103	False	False
107	False	False
122	False	False
...	...	...
546117	False	False
546118	False	False
546119	False	False
546120	False	False
546121	False	False

[128123 rows x 24 columns]

```
[20]: import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(6, 3))
for i, col in enumerate(df.columns[:7]): # Only plotting the first 7 features
    plt.subplot(1, 7, i+1) # Create subplots for each feature
    sns.boxplot(x=df[col])
    plt.title(f"{col}")
plt.tight_layout()
plt.show()
```



No outliers to Be removed nor modified as all showing clinical results.

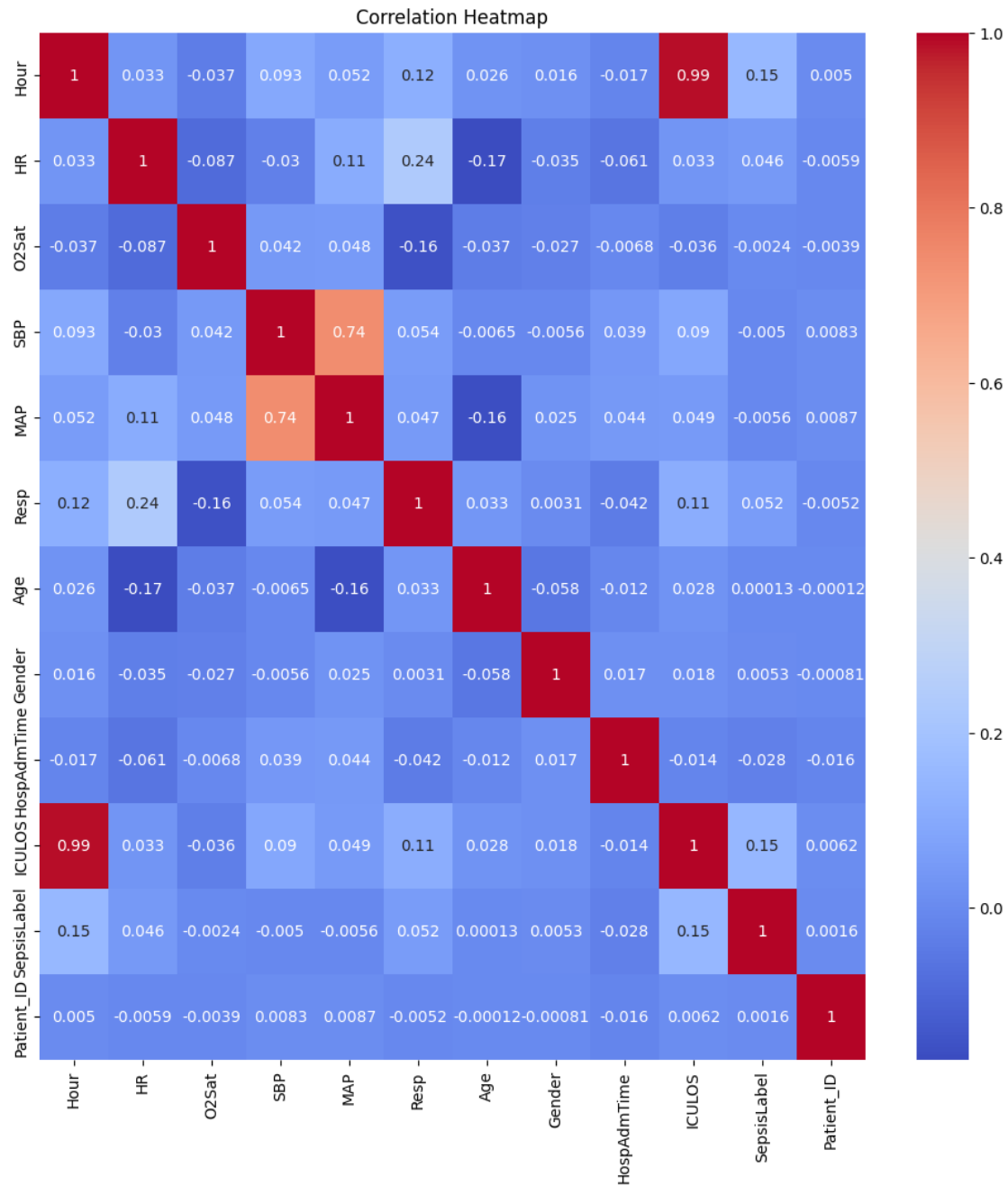
```
[21]: #Fill missing values in columns with mode as HR, O2Sat, SBP, MAP, Resp
↳ (float64)
columns_to_fill = ['HR', 'O2Sat', 'SBP', 'MAP', 'Resp']
for column in columns_to_fill:
```

```
if column in df.columns:
    mean_value = df[column].mean()
    df[column] = df[column].fillna(mean_value)
```

```
[22]: df.isnull().sum()
```

```
[22]: Hour          0
      HR            0
      O2Sat         0
      SBP           0
      MAP           0
      Resp          0
      Age           0
      Gender        0
      HospAdmTime   0
      ICULOS        0
      SepsisLabel   0
      Patient_ID    0
      dtype: int64
```

```
[23]: plt.figure(figsize=(12, 12))
      sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
      plt.title('Correlation Heatmap')
      plt.show()
```



```
[ ]: df['SepsisLabel'].value_counts()
```

```
[ ]: SepsisLabel
0.0    534272
1.0     11850
Name: count, dtype: int64
```

```
[ ]: import seaborn as sns
import matplotlib.pyplot as plt

# Replace gender codes with labels
df['Gender'] = df['Gender'].replace({0.0: 'Female', 1.0: 'Male'})

# Colors and labels
colors = ['lightcoral', 'skyblue']
labels = ['Female', 'Male']
order = ['Female', 'Male']

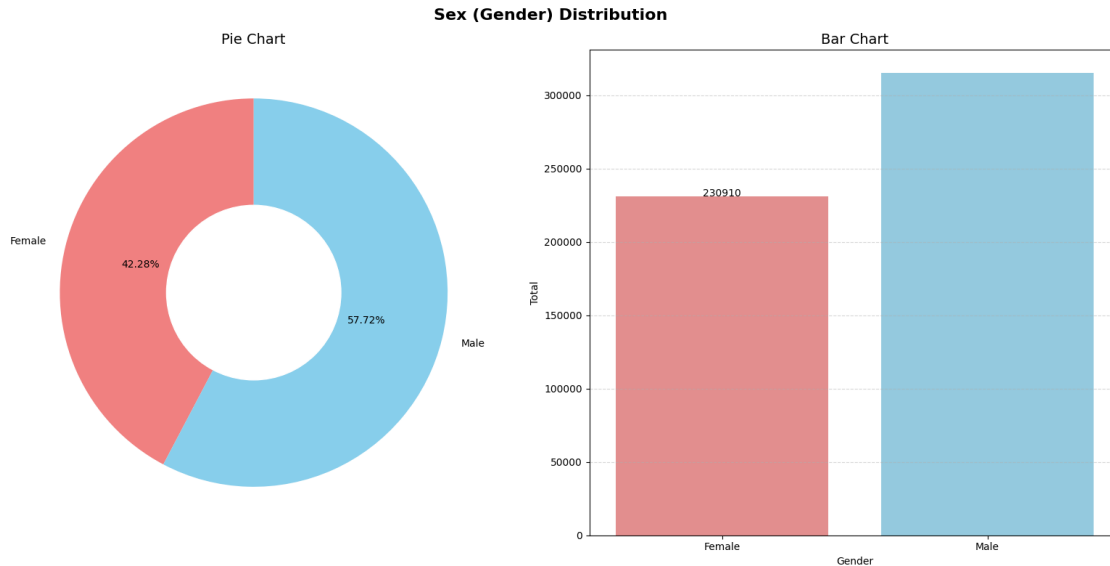
# Set up the figure
plt.figure(figsize=(16, 8))
plt.suptitle('Total Gender Distribution', fontsize=16, fontweight='bold')

# --- Pie Chart ---
plt.subplot(1, 2, 1)
plt.title('Pie Chart', fontsize=14)
plt.pie(df['Gender'].value_counts()[order], labels=labels, colors=colors,
        autopct='%.2f%%', startangle=90)
centre = plt.Circle((0, 0), 0.45, color='white')
plt.gca().add_artist(centre)

# --- Bar Chart ---
plt.subplot(1, 2, 2)
plt.title('Bar Chart', fontsize=14)
ax = sns.countplot(x='Gender', data=df, palette=colors, order=order)

# Add count labels only for bar chart
for bar in ax.containers[0]:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width() / 2, height + 1, int(height),
            ha='center', fontsize=10)

plt.xlabel('Gender')
plt.ylabel('Total')
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```



```
[ ]: import matplotlib.pyplot as plt
from matplotlib.patches import Patch

####Sort SepsisLabel counts by index
sepsis_counts = df['SepsisLabel'].value_counts().sort_index()

# Create the bar chart
plt.figure(figsize=(8, 5))
ax = sepsis_counts.plot(kind='bar', color=['green', 'red'])

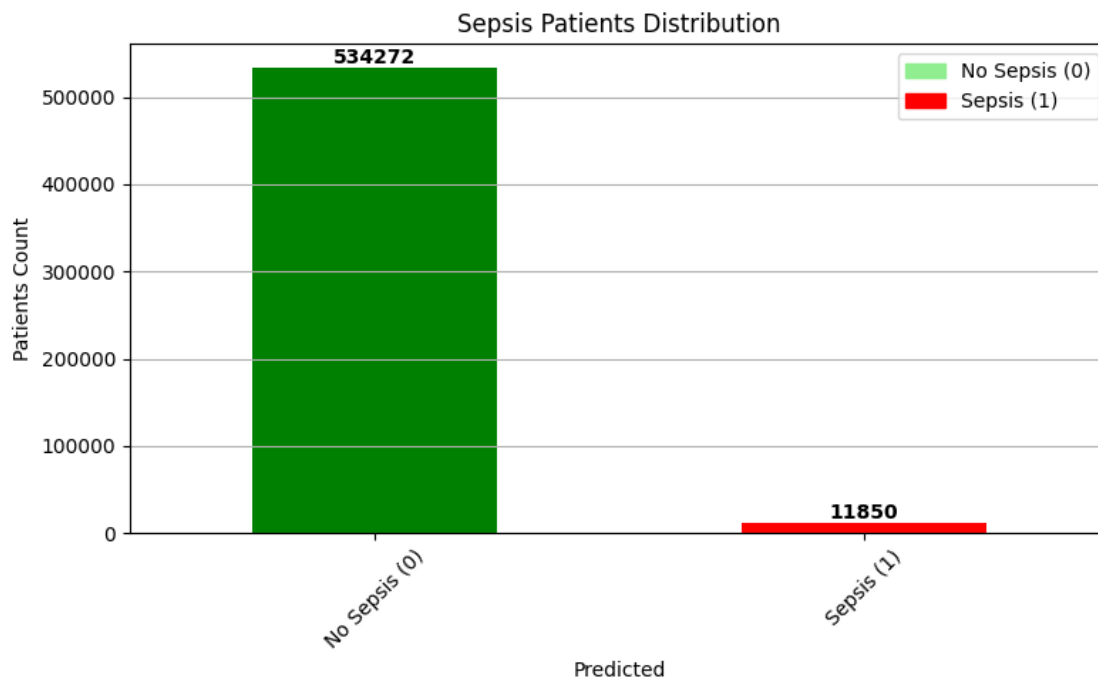
# Customize ticks and labels
plt.xticks([0, 1], ['No Sepsis (0)', 'Sepsis (1)'], rotation=45)
plt.title('Sepsis Patients Distribution')
plt.xlabel('Predicted')
plt.ylabel('Patients Count')
plt.grid(axis='y')

#### Add count labels on top of each bar

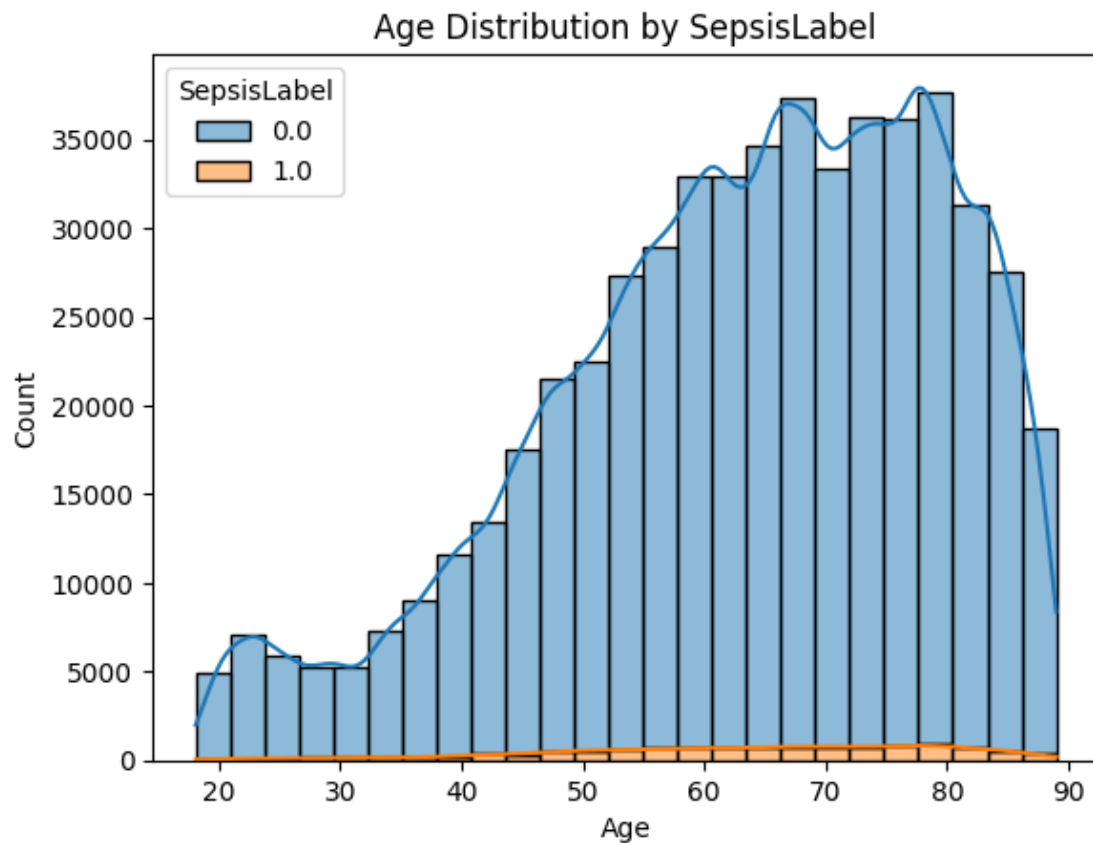
for i, count in enumerate(sepsis_counts):
    ax.text(i, count + 5, str(count), ha='center', va='bottom',
            fontweight='bold')

# color legend
legend_labels = [Patch(color='lightgreen', label='No Sepsis (0)'),
                  Patch(color='red', label='Sepsis (1)')]
plt.legend(handles=legend_labels)
```

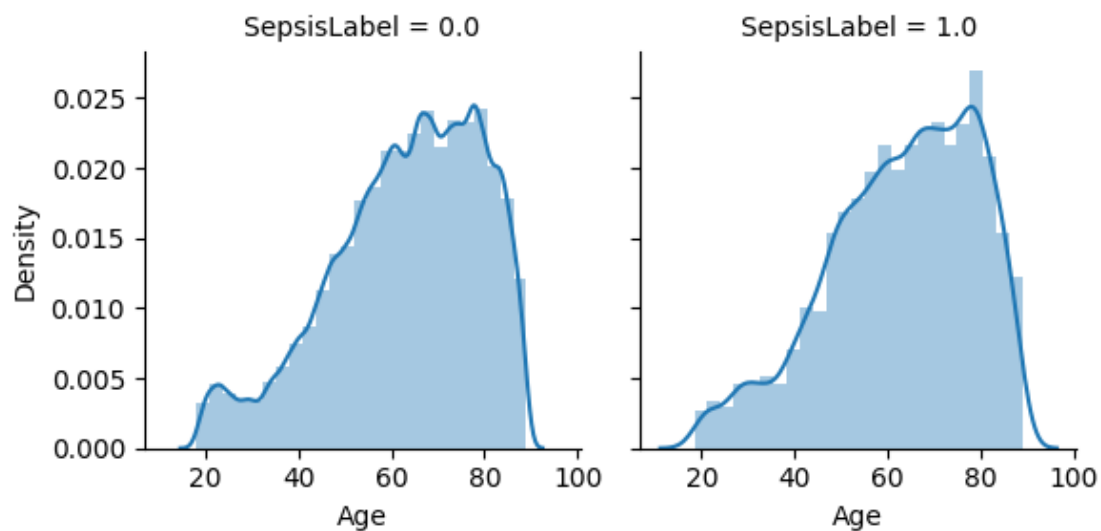
```
plt.tight_layout()
plt.show()
```



```
[ ]: sns.histplot(data=df, x='Age', hue='SepsisLabel', bins=25, kde=True,
↳ multiple='stack')
plt.title("Age Distribution by SepsisLabel")
plt.xlabel("Age")
plt.ylabel("Count")
plt.show()
```



```
[ ]: g = sns.FacetGrid(df,col="SepsisLabel")  
g.map(sns.distplot,"Age", bins=25)  
plt.show()
```

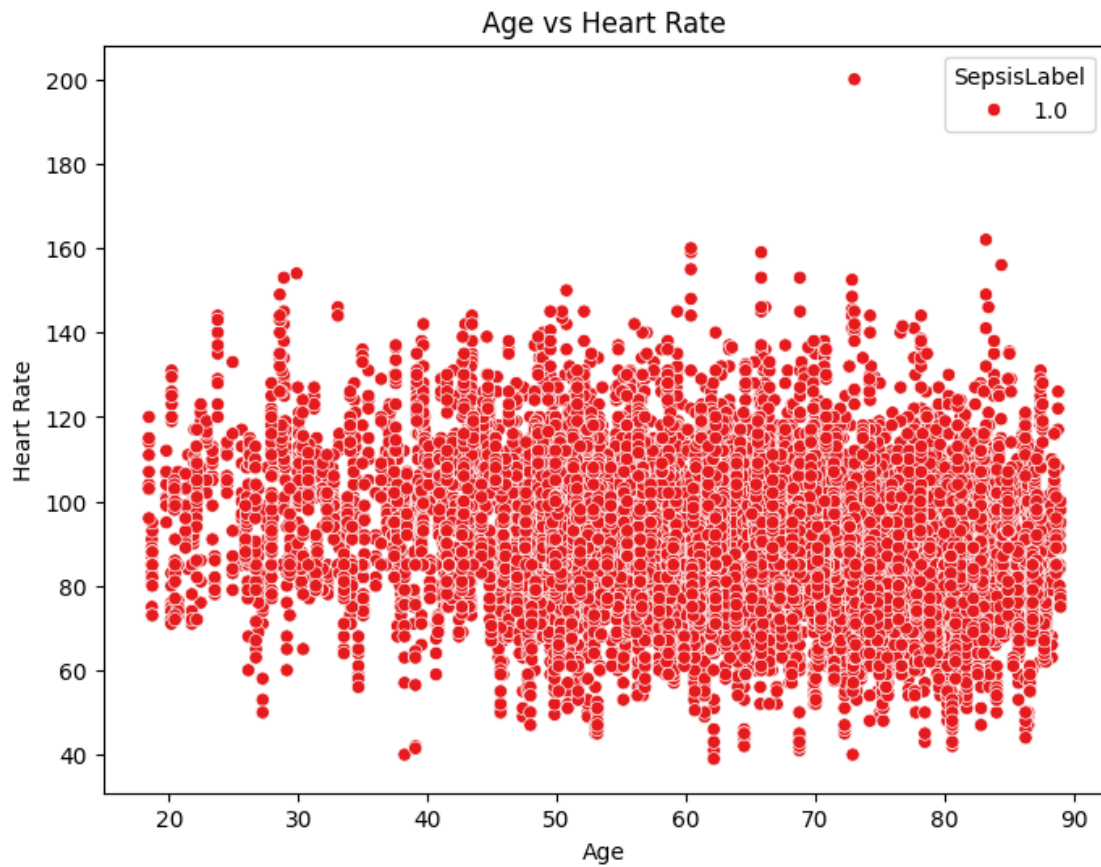


```
[ ]: df.dtypes
```

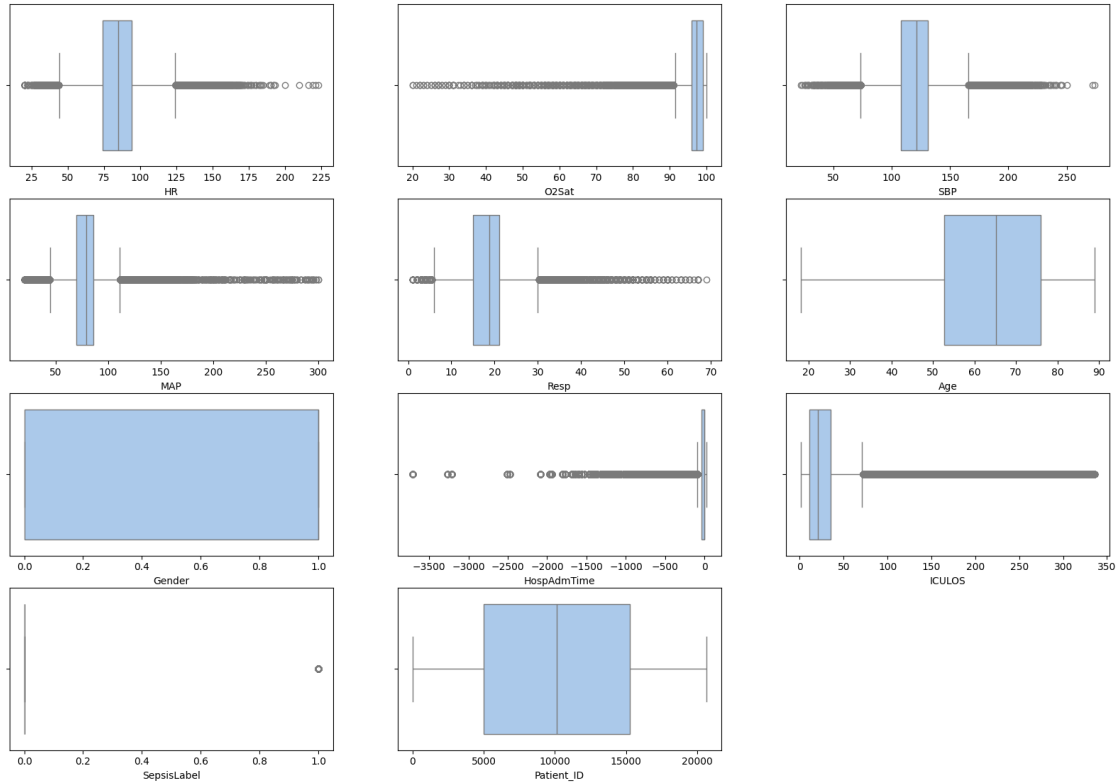
```
[ ]: Hour          int64
      HR           float64
      O2Sat        float64
      SBP          float64
      MAP          float64
      Resp         float64
      Age          float64
      Gender       float64
      HospAdmTime  float64
      ICULOS       float64
      SepsisLabel  float64
      Patient_ID   float64
      dtype: object
```

```
[ ]: plt.figure(figsize=(8, 6))

      # Filter data for SepsisLabel = 1
      sepsis_df = df[df['SepsisLabel'] == 1]
      sns.scatterplot(x='Age', y='HR', data=sepsis_df, hue='SepsisLabel',
                      palette='Set1')
      plt.title('Age vs Heart Rate')
      plt.xlabel('Age')
      plt.ylabel('Heart Rate')
      plt.show()
```



```
[ ]: plt.figure(figsize=(20,35))
      plotnumber =1
      for column in df.columns[1:]:
          ax = plt.subplot(10,3,plotnumber)
          sns.boxplot(data = df, x = column, palette='pastel')
          plt.xlabel(column)
          plotnumber+=1
      plt.show()
```



```
[ ]: import matplotlib.pyplot as plt
import pandas as pd
categorical_cols = ['ICULOS', 'Hour', 'MAP']
# Number of categorical columns
n = len(categorical_cols)
nrows, ncols = (3, 3) # Adjust grid size as needed

# Create subplots
fig, axes = plt.subplots(nrows=nrows, ncols=ncols, figsize=(18, 12))
axes = axes.flatten()

# Loop through each categorical column
for i, col in enumerate(categorical_cols):
    ct = pd.crosstab(df[col], df['Gender'], normalize='index')

    # Plot one pie chart per feature (first category only)
    for category in ct.index:
        sizes = ct.loc[category]
        labels = [f'{label} ({sizes[label]*100:.1f}%)' for label in sizes.index]

        ax = axes[i]
```

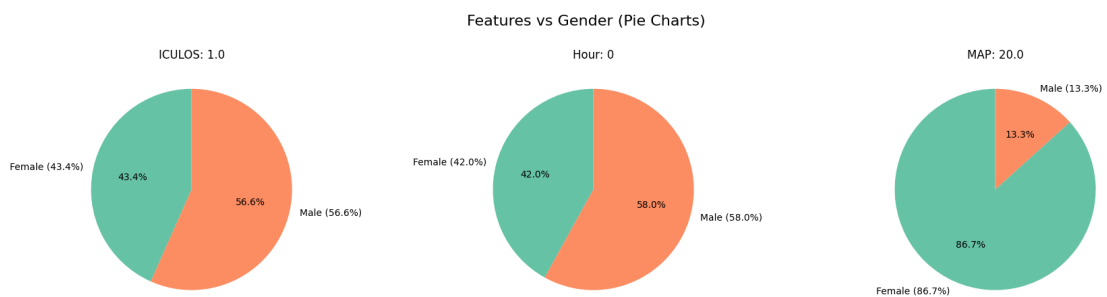
```

    ax.pie(
        sizes,
        labels=labels,
        autopct='%1.1f%%',
        startangle=90,
        colors=plt.cm.Set2.colors
    )
    ax.set_title(f'{col}: {category}')
    break # Remove this line to plot all categories separately

# Remove unused axes
for j in range(len(categorical_cols), len(axes)):
    fig.delaxes(axes[j])

# Final layout
plt.tight_layout()
plt.suptitle('Features vs Gender (Pie Charts)', fontsize=16, y=1.03)
plt.show()

```



END

Thank you