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

from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from keras.layers import Dense, BatchNormalization, Dropout, LSTM
from keras.models import Sequential
from keras import callbacks
from sklearn.metrics import precision_score, recall_score,
confusion_matrix, classification_report, accuracy_score, f1_score

import warnings
warnings.filterwarnings('ignore')
```

```
pip install keras
```

```
Requirement already satisfied: keras in c:\users\suravi\anaconda3\lib\
site-packages (3.1.1)
Requirement already satisfied: absl-py in c:\users\suravi\anaconda3\
lib\site-packages (from keras) (2.1.0)
Requirement already satisfied: numpy in c:\users\suravi\anaconda3\lib\
site-packages (from keras) (1.24.3)
Requirement already satisfied: rich in c:\users\suravi\anaconda3\lib\
site-packages (from keras) (13.7.1)
Requirement already satisfied: namex in c:\users\suravi\anaconda3\lib\
site-packages (from keras) (0.0.7)
Requirement already satisfied: h5py in c:\users\suravi\anaconda3\lib\
site-packages (from keras) (3.10.0)
Requirement already satisfied: optree in c:\users\suravi\anaconda3\
lib\site-packages (from keras) (0.11.0)
Requirement already satisfied: ml-dtypes in c:\users\suravi\anaconda3\
lib\site-packages (from keras) (0.3.2)
Requirement already satisfied: typing-extensions>=4.0.0 in c:\users\
suravi\anaconda3\lib\site-packages (from optree->keras) (4.7.1)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\
suravi\anaconda3\lib\site-packages (from rich->keras) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\
suravi\anaconda3\lib\site-packages (from rich->keras) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in c:\users\suravi\
anaconda3\lib\site-packages (from markdown-it-py>=2.2.0->rich->keras)
(0.1.0)
Note: you may need to restart the kernel to use updated packages.
```

```
pip install tensorflow
```

```
Requirement already satisfied: tensorflow in c:\users\suravi\
anaconda3\lib\site-packages (2.16.1)
Requirement already satisfied: tensorflow-intel==2.16.1 in c:\users\
suravi\anaconda3\lib\site-packages (from tensorflow) (2.16.1)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\suravi\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\suravi\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in c:\users\
suravi\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in
c:\users\suravi\anaconda3\lib\site-packages (from tensorflow-
intel==2.16.1->tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\suravi\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in c:\users\suravi\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (3.10.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\suravi\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes~=0.3.1 in c:\users\suravi\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (0.3.2)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\suravi\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (3.3.0)
Requirement already satisfied: packaging in c:\users\suravi\anaconda3\
lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (23.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!
=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in c:\users\suravi\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (4.25.3)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\suravi\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (2.31.0)
Requirement already satisfied: setuptools in c:\users\suravi\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (68.0.0)
Requirement already satisfied: six>=1.12.0 in c:\users\suravi\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\suravi\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\
```

```
suravi\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (4.7.1)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\suravi\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (1.14.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\suravi\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (1.62.1)
Requirement already satisfied: tensorboard<2.17,>=2.16 in c:\users\
suravi\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (2.16.2)
Requirement already satisfied: keras>=3.0.0 in c:\users\suravi\
anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (3.1.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
c:\users\suravi\anaconda3\lib\site-packages (from tensorflow-
intel==2.16.1->tensorflow) (0.31.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in c:\users\
suravi\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (1.24.3)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\suravi\
anaconda3\lib\site-packages (from astunparse>=1.6.0->tensorflow-
intel==2.16.1->tensorflow) (0.38.4)
Requirement already satisfied: rich in c:\users\suravi\anaconda3\lib\
site-packages (from keras>=3.0.0->tensorflow-intel==2.16.1-
>tensorflow) (13.7.1)
Requirement already satisfied: namex in c:\users\suravi\anaconda3\lib\
site-packages (from keras>=3.0.0->tensorflow-intel==2.16.1-
>tensorflow) (0.0.7)
Requirement already satisfied: optree in c:\users\suravi\anaconda3\
lib\site-packages (from keras>=3.0.0->tensorflow-intel==2.16.1-
>tensorflow) (0.11.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\
suravi\anaconda3\lib\site-packages (from requests<3,>=2.21.0-
>tensorflow-intel==2.16.1->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\suravi\
anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorflow-
intel==2.16.1->tensorflow) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\suravi\
anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorflow-
intel==2.16.1->tensorflow) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\suravi\
anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorflow-
intel==2.16.1->tensorflow) (2023.7.22)
Requirement already satisfied: markdown>=2.6.8 in c:\users\suravi\
anaconda3\lib\site-packages (from tensorboard<2.17,>=2.16->tensorflow-
intel==2.16.1->tensorflow) (3.4.1)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0
in c:\users\suravi\anaconda3\lib\site-packages (from
tensorboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow) (0.7.2)
```

```
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\suravi\
anaconda3\lib\site-packages (from tensorboard<2.17,>=2.16->tensorflow-
intel==2.16.1->tensorflow) (2.2.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\suravi\
anaconda3\lib\site-packages (from werkzeug>=1.0.1-
>tensorboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow)
(2.1.1)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\
suravi\anaconda3\lib\site-packages (from rich->keras>=3.0.0-
>tensorflow-intel==2.16.1->tensorflow) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\
suravi\anaconda3\lib\site-packages (from rich->keras>=3.0.0-
>tensorflow-intel==2.16.1->tensorflow) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in c:\users\suravi\
anaconda3\lib\site-packages (from markdown-it-py>=2.2.0->rich-
>keras>=3.0.0->tensorflow-intel==2.16.1->tensorflow) (0.1.0)
Note: you may need to restart the kernel to use updated packages.
```

```
df = pd.read_csv('heart_failure_clinical_records_dataset.csv')
df.head()
```

```
    age  anaemia  creatinine_phosphokinase  diabetes
ejection_fraction  \
0  75.0        0                       582         0
20
1  55.0        0                      7861         0
38
2  65.0        0                       146         0
20
3  50.0        1                       111         0
20
4  65.0        1                       160         1
20

   high_blood_pressure  platelets  serum_creatinine  serum_sodium  sex
\
0                    1  265000.00               1.9           130    1

1                    0  263358.03               1.1           136    1

2                    0  162000.00               1.3           129    1

3                    0  210000.00               1.9           137    1

4                    0  327000.00               2.7           116    0


   smoking  time  DEATH_EVENT
0        0     4            1
```

```
1        0     6             1
2        1     7             1
3        0     7             1
4        0     8             1
```

```
df.size
```

```
3887
```

```
df.shape
```

```
(299, 13)
```

```
df.columns
```

```
Index(['age', 'anaemia', 'creatinine_phosphokinase', 'diabetes',
       'ejection_fraction', 'high_blood_pressure', 'platelets',
       'serum_creatinine', 'serum_sodium', 'sex', 'smoking', 'time',
       'DEATH_EVENT'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 13 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   age                       299 non-null    float64
 1   anaemia                   299 non-null    int64  
 2   creatinine_phosphokinase  299 non-null    int64  
 3   diabetes                  299 non-null    int64  
 4   ejection_fraction         299 non-null    int64  
 5   high_blood_pressure       299 non-null    int64  
 6   platelets                 299 non-null    float64
 7   serum_creatinine          299 non-null    float64
 8   serum_sodium              299 non-null    int64  
 9   sex                       299 non-null    int64  
 10  smoking                   299 non-null    int64  
 11  time                      299 non-null    int64  
 12  DEATH_EVENT               299 non-null    int64  
dtypes: float64(3), int64(10)
memory usage: 30.5 KB
```

```
df.describe().T
```

```
                          count           mean           std      min
\
age                       299.0      60.833893     11.894809     40.0

anaemia                   299.0       0.431438      0.496107      0.0
```

creatinine_phosphokinase	299.0	581.839465	970.287881	23.0
diabetes	299.0	0.418060	0.494067	0.0
ejection_fraction	299.0	38.083612	11.834841	14.0
high_blood_pressure	299.0	0.351171	0.478136	0.0
platelets	299.0	263358.029264	97804.236869	25100.0
serum_creatinine	299.0	1.393880	1.034510	0.5
serum_sodium	299.0	136.625418	4.412477	113.0
sex	299.0	0.648829	0.478136	0.0
smoking	299.0	0.321070	0.467670	0.0
time	299.0	130.260870	77.614208	4.0
DEATH_EVENT	299.0	0.321070	0.467670	0.0

	25%	50%	75%	max
age	51.0	60.0	70.0	95.0
anaemia	0.0	0.0	1.0	1.0
creatinine_phosphokinase	116.5	250.0	582.0	7861.0
diabetes	0.0	0.0	1.0	1.0
ejection_fraction	30.0	38.0	45.0	80.0
high_blood_pressure	0.0	0.0	1.0	1.0
platelets	212500.0	262000.0	303500.0	850000.0
serum_creatinine	0.9	1.1	1.4	9.4
serum_sodium	134.0	137.0	140.0	148.0
sex	0.0	1.0	1.0	1.0
smoking	0.0	0.0	1.0	1.0
time	73.0	115.0	203.0	285.0
DEATH_EVENT	0.0	0.0	1.0	1.0

Checking Missing Values

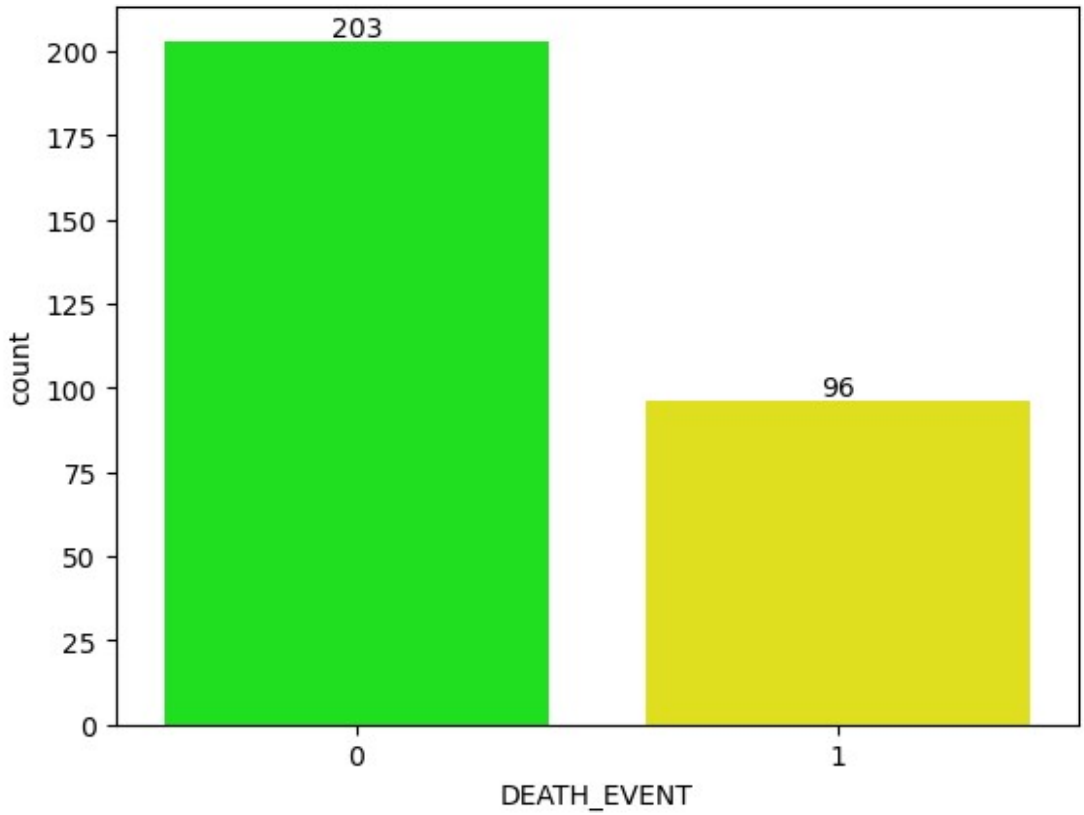
```
df.isnull().sum()
```

```
Missing Values    0
dtype: int64
```

DATA Visualization

Checking Skewness and Data Imbalancing

```
cols = ["#00FF00", "#FFFF00"]

# countplot
ax = sns.countplot(x=df["DEATH_EVENT"], palette=cols)

# Adding label
for container in ax.containers:
    ax.bar_label(container)

plt.show()
```

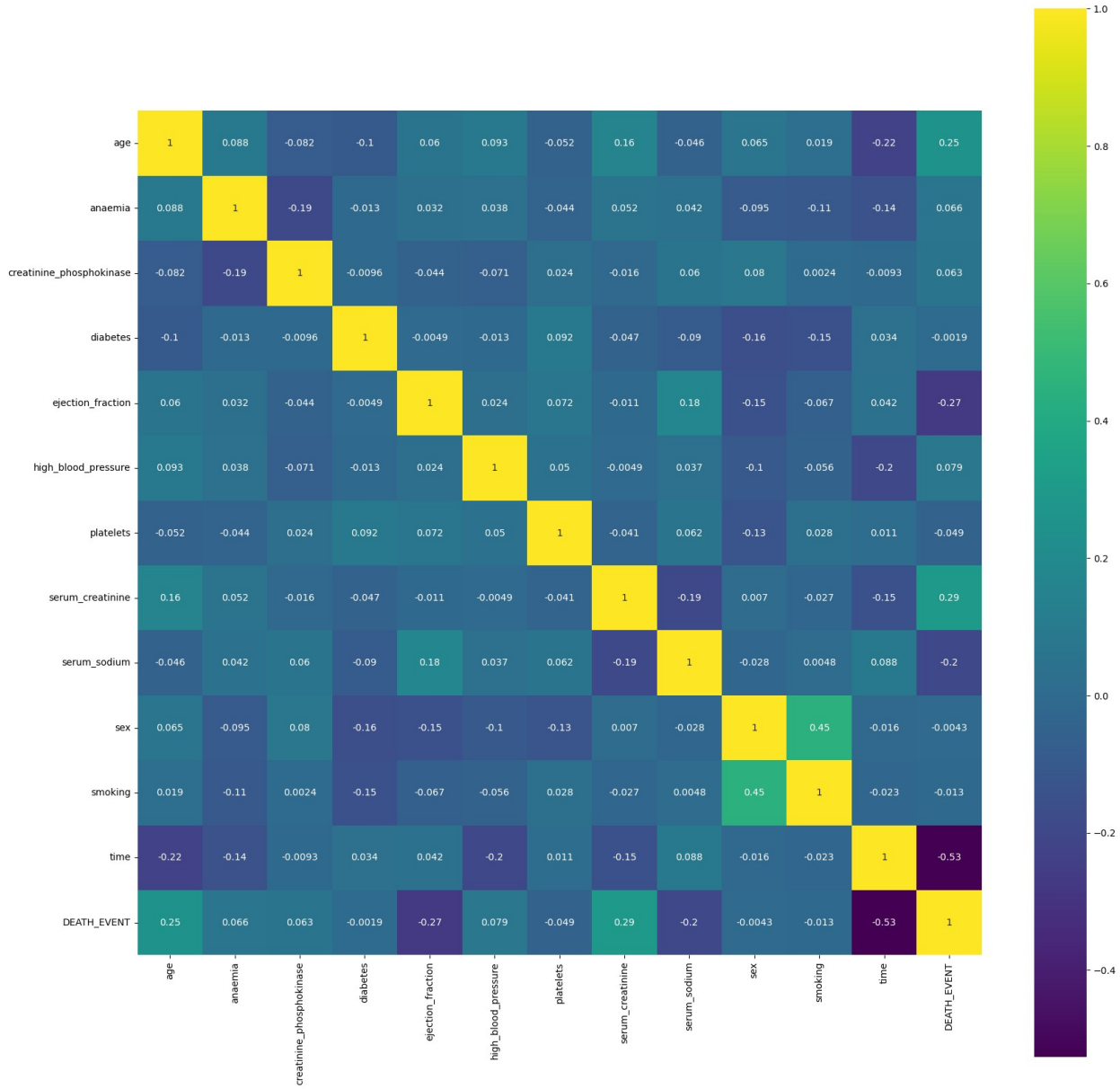


There is an imbalance in the data because the target labels are 203 as opposed to 96.

The features "serum creatinine" and "creatinine_phosphokinase" are wildly skewed.

Correlation Matrix

```
cmap = sns.diverging_palette(2, 165, s=80, l=55, n=9)

# correlation matrix
corrmat = df.corr()

# heatmap
plt.figure(figsize=(20, 20))
sns.heatmap(corrmat, cmap='viridis', annot=True, square=True)
plt.show()
```



"Time" is the most critical component because it would have been imperative to have prompt treatment for a cardiovascular condition after an early diagnosis, hence lowering the risk of death. (As the inverse relationship is clear)

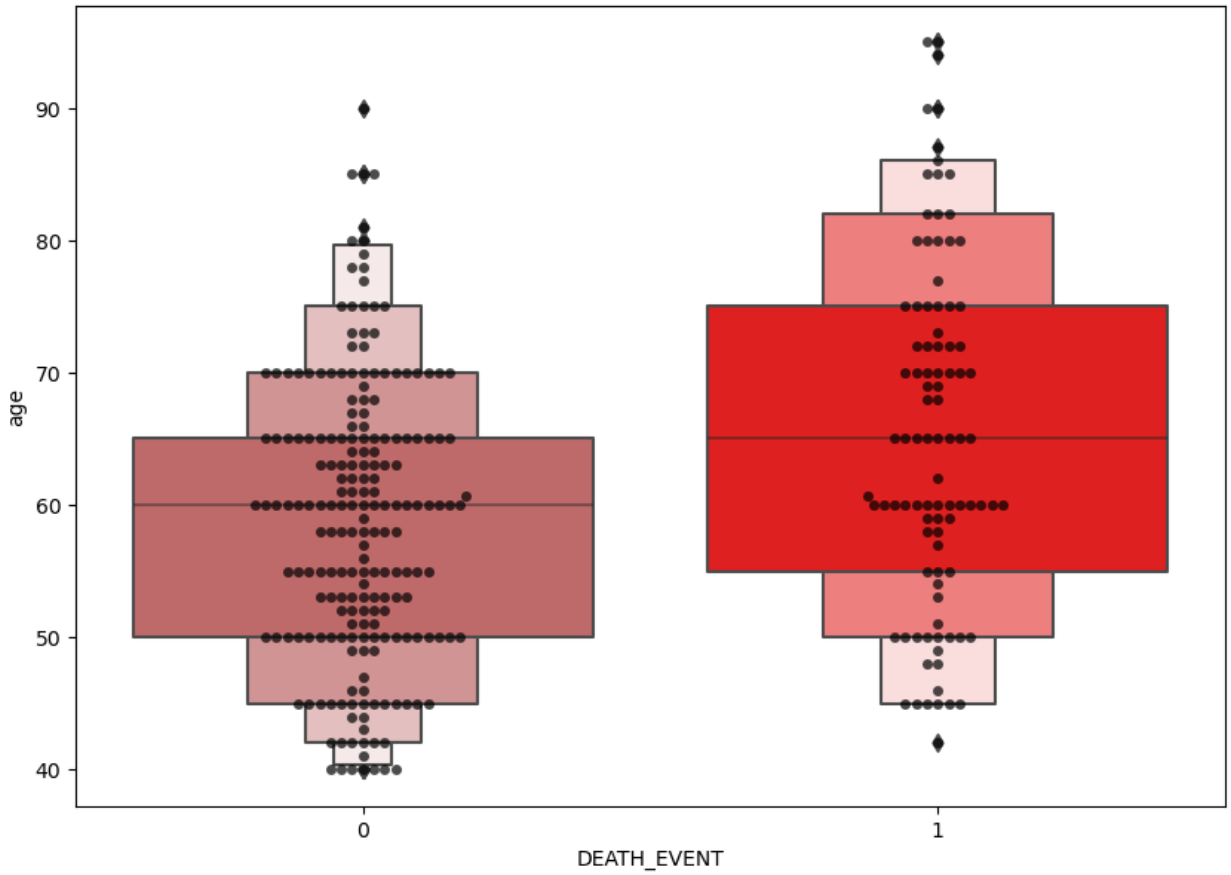
"serum_creatinine" is the next crucial element since the presence of serum, an essential blood component, in blood facilitates cardiac function.

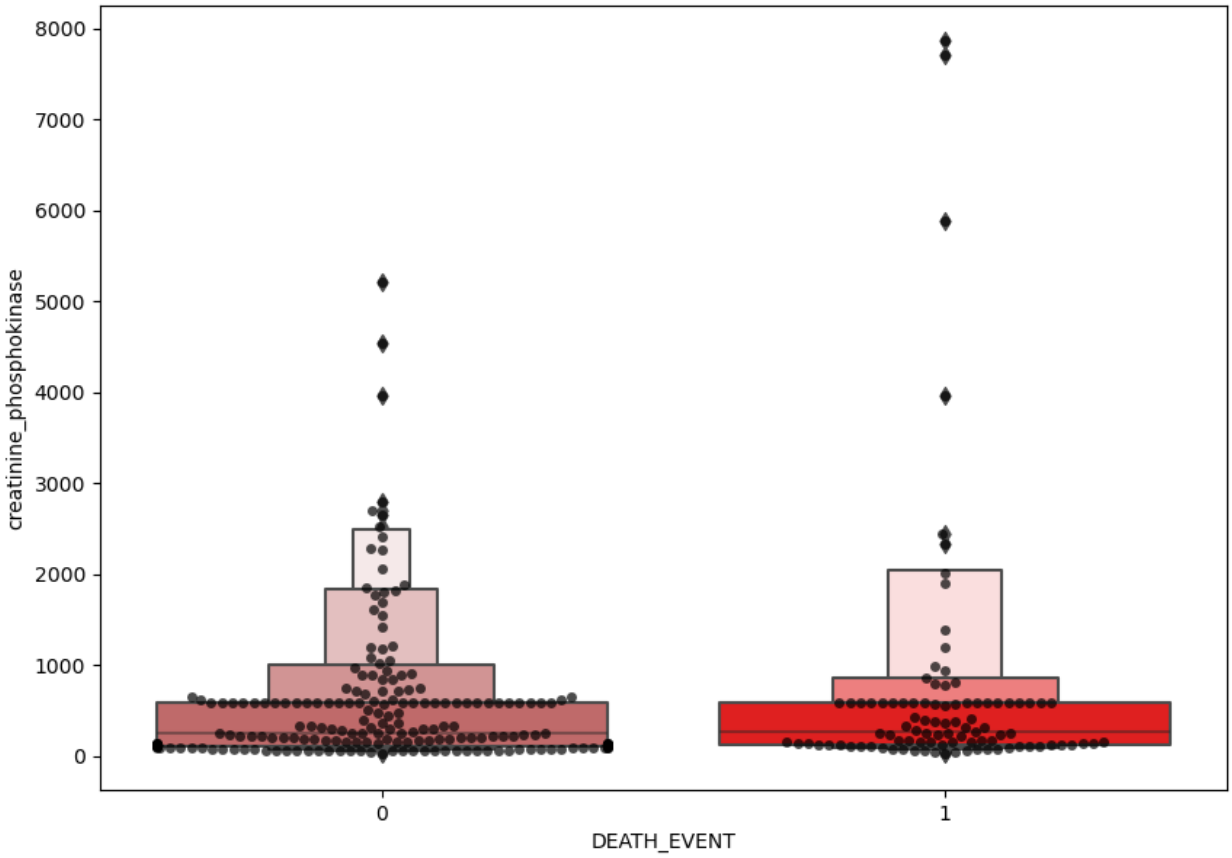
"ejection_fraction" significantly affects the target variable as well, which makes sense given that it essentially represents the heart's efficiency.

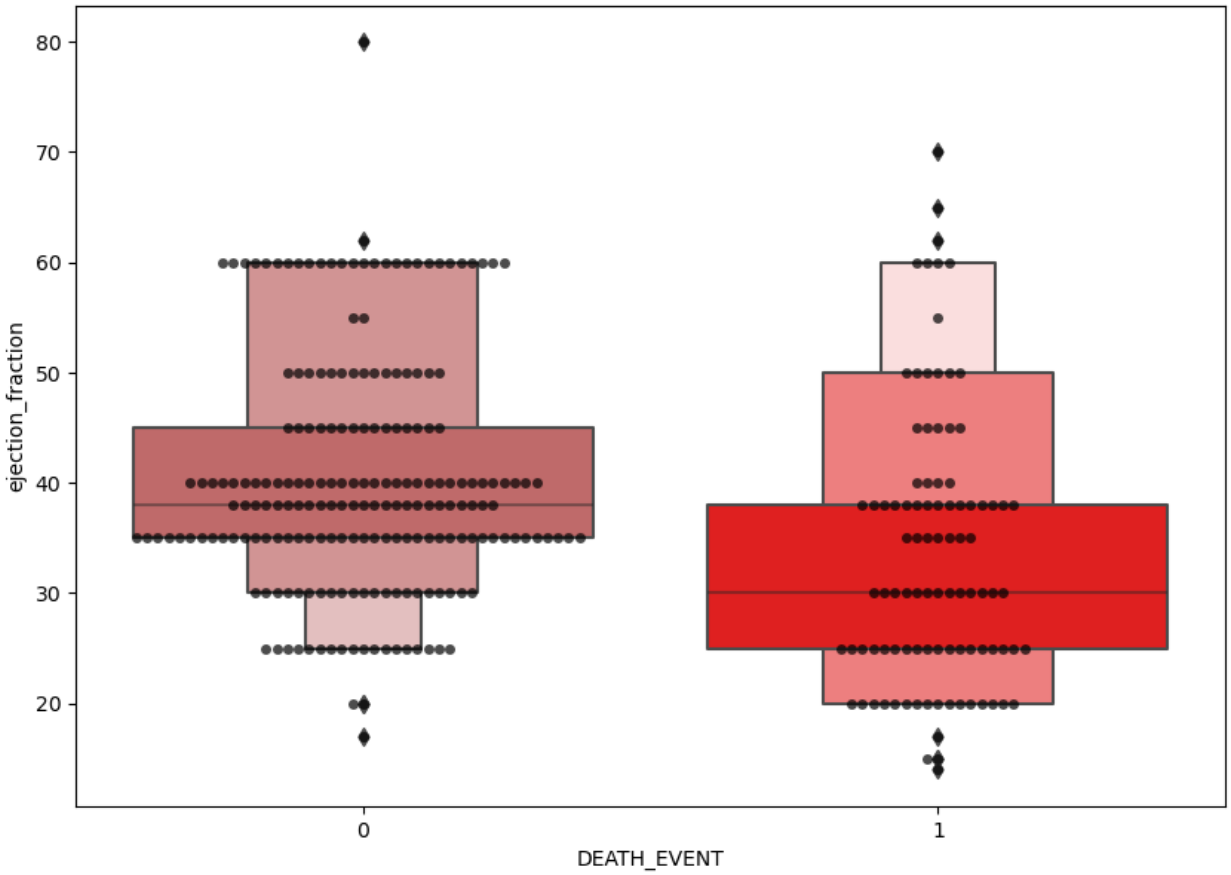
The inverse relation pattern indicates that the heart's capacity to function decreases with age.

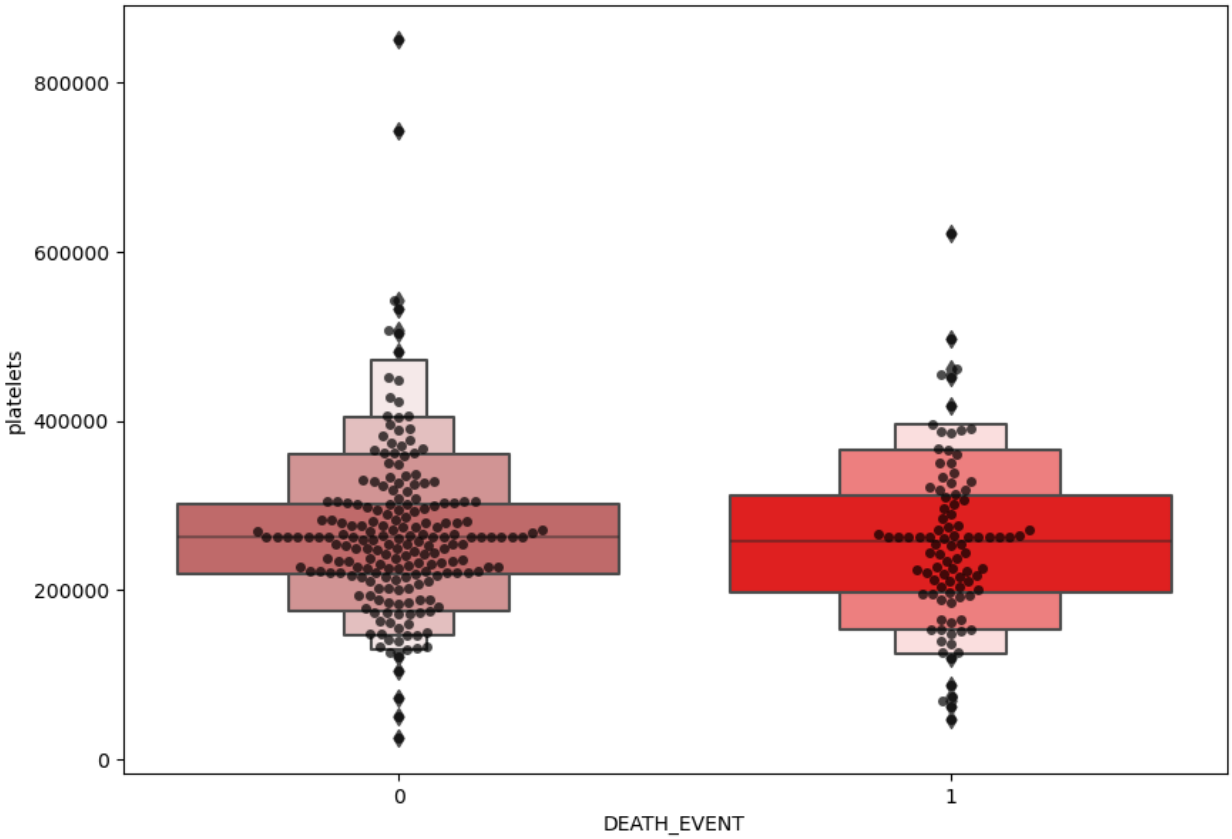
Checking Outliers

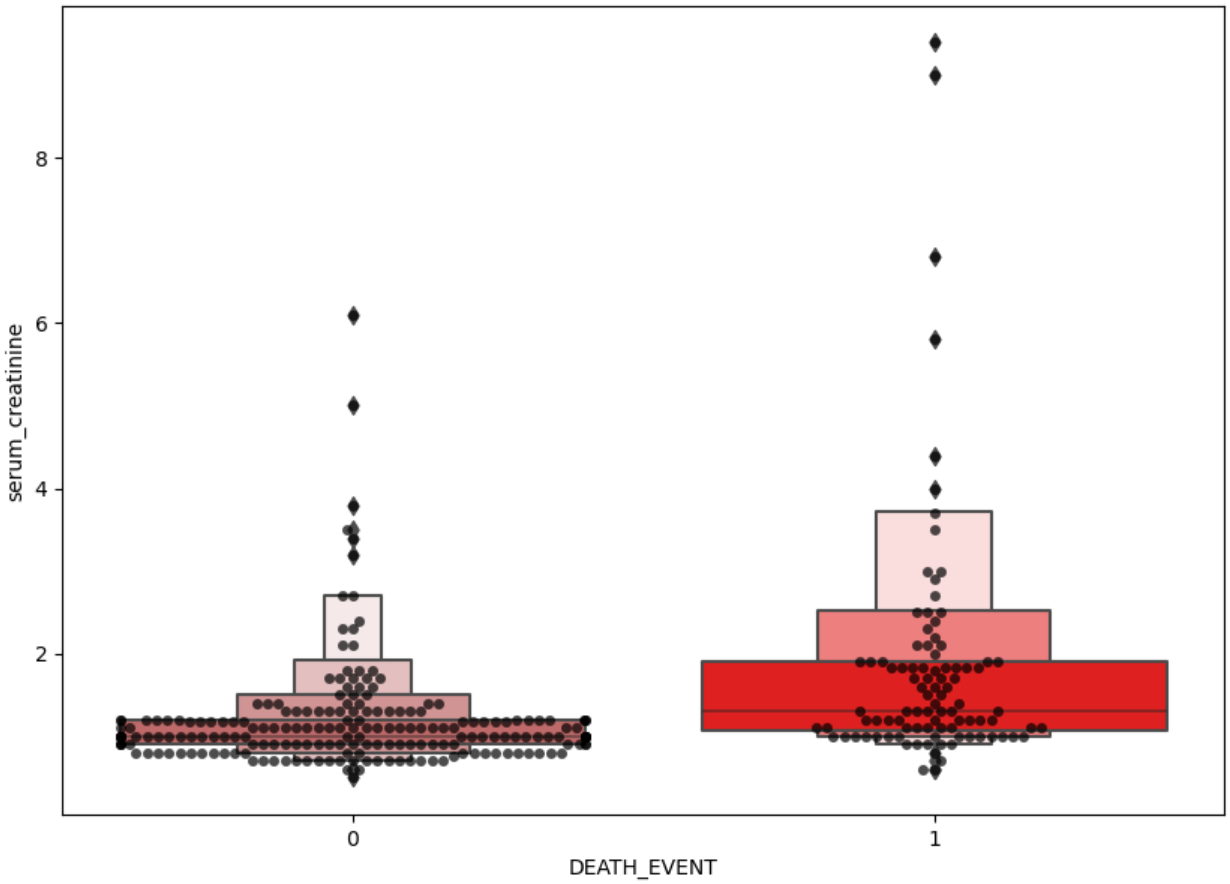
```
features = ["age", "creatinine_phosphokinase", "ejection_fraction",
            "platelets", "serum_creatinine", "serum_sodium", "time"]

# Iterate over each feature
for feature in features:
    plt.figure(figsize=(10, 7))
    sns.swarmplot(x=df["DEATH_EVENT"], y=df[feature], color="black",
alpha=0.7)
    sns.boxenplot(x=df["DEATH_EVENT"], y=df[feature], palette=cols)
    plt.show()
```

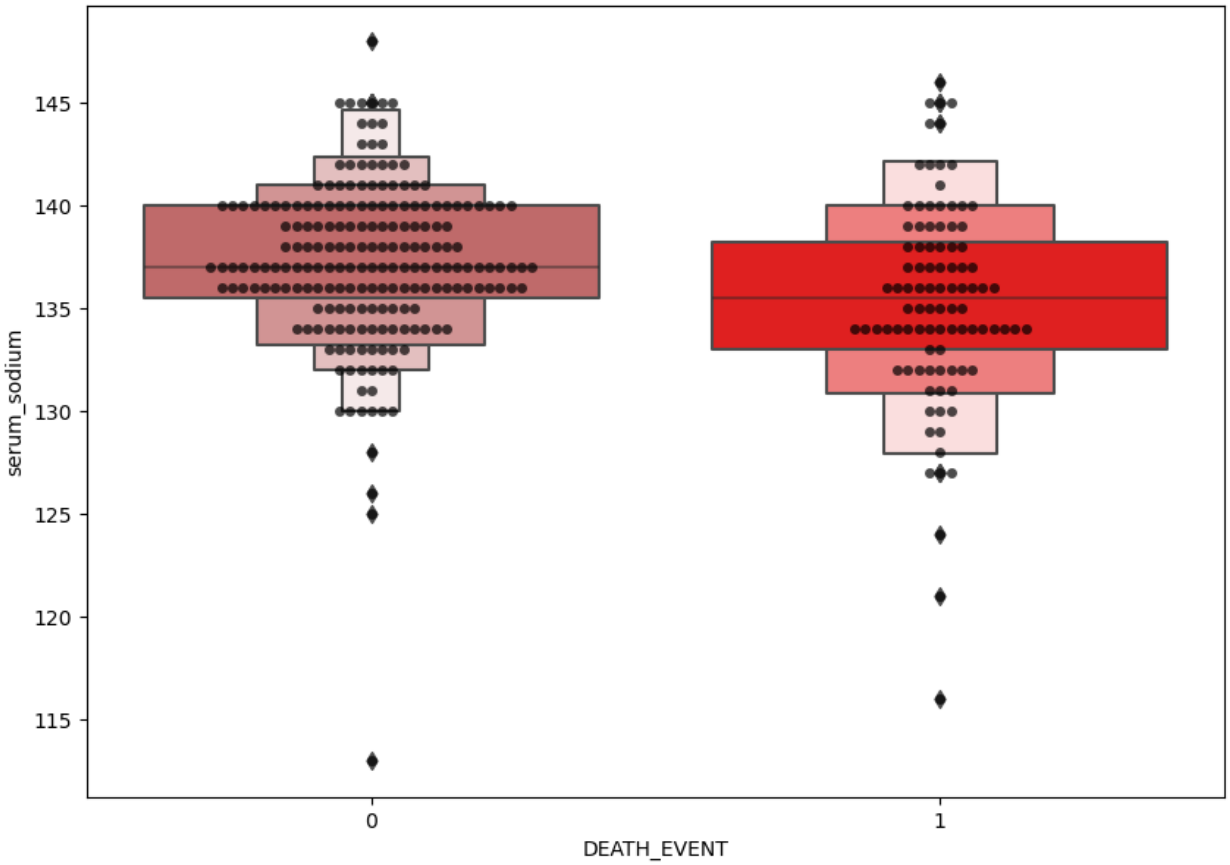

90
80
70
60
50
40
age
0
1
DEATH_EVENT

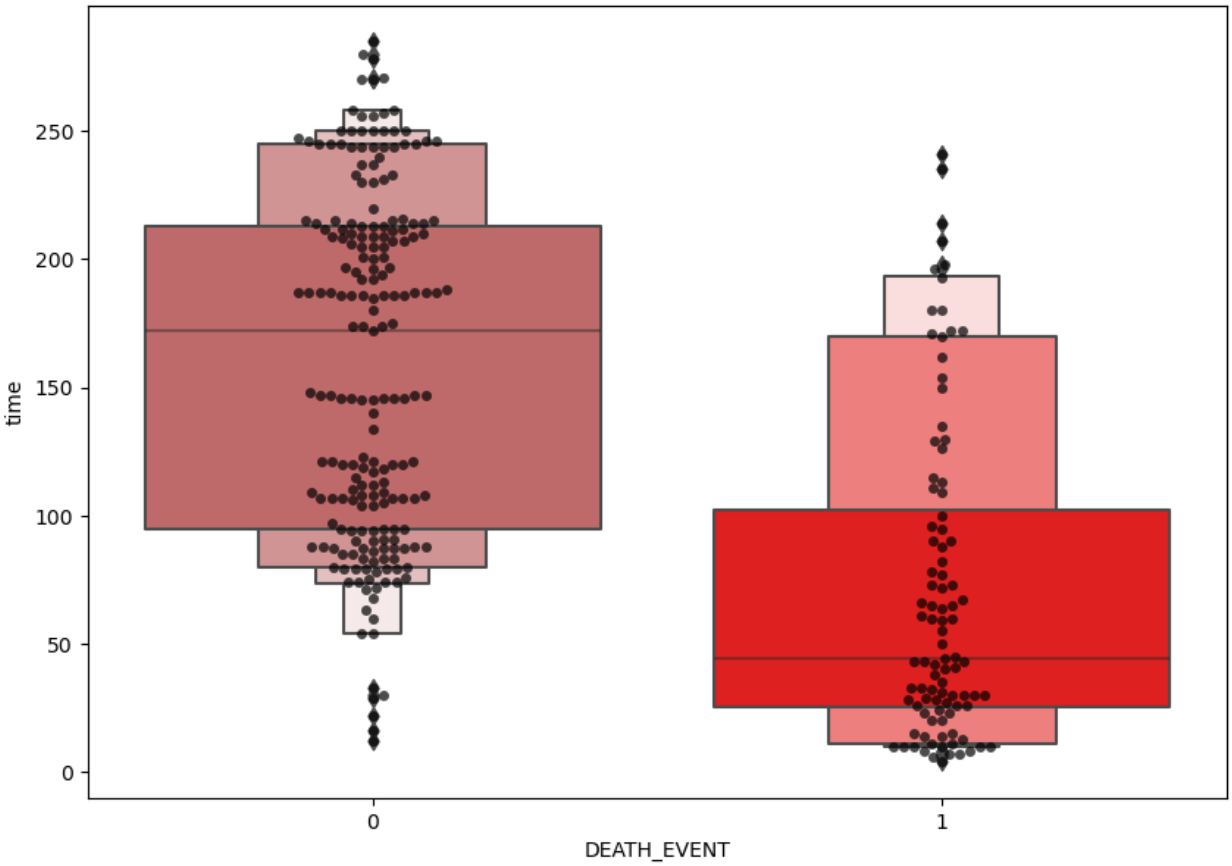

8000
7000
6000
5000
4000
3000
2000
1000
0
creatinine_phosphokinase
0
1
DEATH_EVENT


80
70
60
50
40
30
20
ejection_fraction
0
1
DEATH_EVENT


800000
600000
400000
200000
0
platelets
0
1
DEATH_EVENT


8
6
4
2
serum_creatinine
0
1
DEATH_EVENT


145
140
135
130
125
120
115
serum_sodium
0
1
DEATH_EVENT



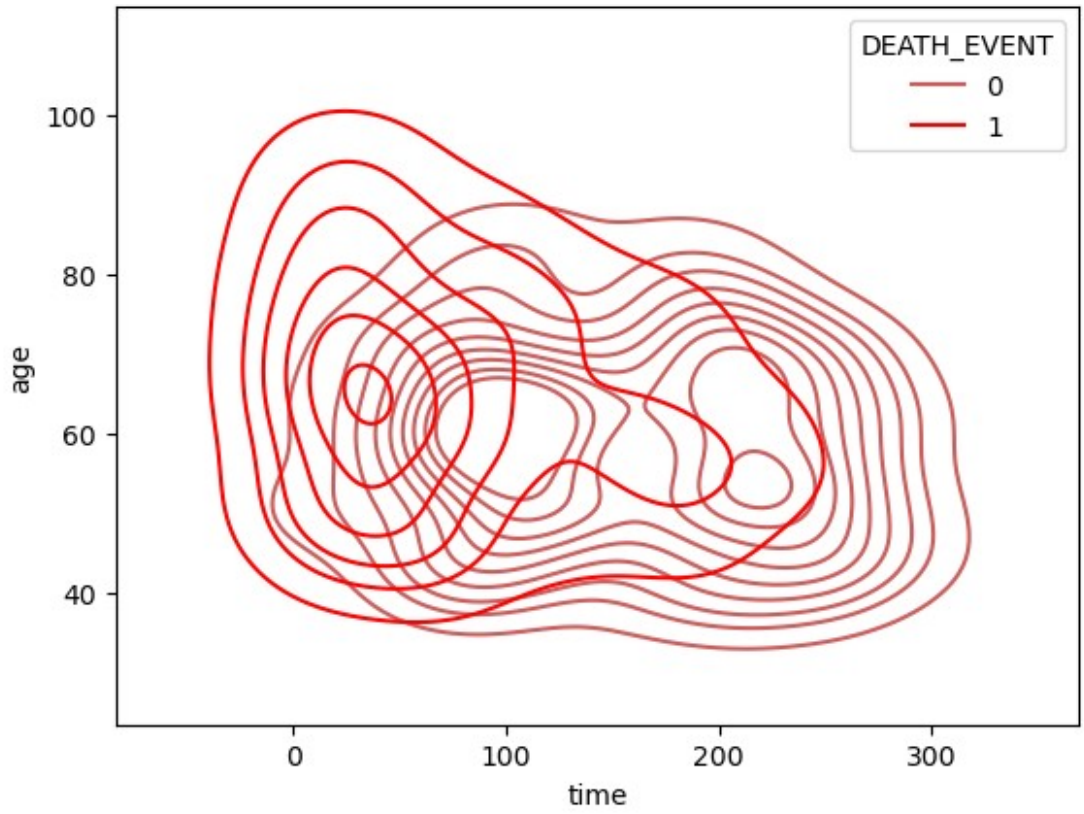
Nearly every feature shows a few outliers.

We won't remove such outliers during data preprocessing because of the dataset's size and relevance, which would prevent any statistical flukes.

KDE Plot

```
sns.kdeplot(x=df["time"], y=df["age"], hue=df["DEATH_EVENT"],
palette=cols)
```

```
<Axes: xlabel='time', ylabel='age'>
```



Patients frequently died only when they grew older when there were fewer follow-up days.

An increased number of follow-up days increases the likelihood of any mortality.

DATA Preprocessing / Training and Testing of DATA

```
X=df.drop(["DEATH_EVENT"],axis=1)
y=df["DEATH_EVENT"]

col_names = list(X.columns)

# Standardize the features
s_scaler = preprocessing.StandardScaler()
X_scaled = s_scaler.fit_transform(X)

# Create a DataFrame with the scaled features
X_scaled = pd.DataFrame(X_scaled, columns=col_names)

# Display descriptive statistics for the scaled features
X_scaled.describe().T
```

```
                           count          mean       std       min
25%  \
age                        299.0  5.703353e-16  1.001676 -1.754448 -
0.828124
anaemia                    299.0  1.009969e-16  1.001676 -0.871105 -
0.871105
creatinine_phosphokinase   299.0  0.000000e+00  1.001676 -0.576918 -
0.480393
diabetes                   299.0  9.060014e-17  1.001676 -0.847579 -
0.847579
ejection_fraction          299.0 -3.267546e-17  1.001676 -2.038387 -
0.684180
high_blood_pressure        299.0  0.000000e+00  1.001676 -0.735688 -
0.735688
platelets                  299.0  7.723291e-17  1.001676 -2.440155 -
0.520870
serum_creatinine           299.0  1.425838e-16  1.001676 -0.865509 -
0.478205
serum_sodium               299.0 -8.673849e-16  1.001676 -5.363206 -
0.595996
sex                        299.0 -8.911489e-18  1.001676 -1.359272 -
1.359272
smoking                    299.0 -1.188199e-17  1.001676 -0.687682 -
0.687682
time                       299.0 -1.901118e-16  1.001676 -1.629502 -
0.739000

                                 50%       75%       max
age                        -0.070223  0.771889  2.877170
anaemia                    -0.871105  1.147968  1.147968
creatinine_phosphokinase   -0.342574  0.000166  7.514640
diabetes                   -0.847579  1.179830  1.179830
ejection_fraction          -0.007077  0.585389  3.547716
high_blood_pressure        -0.735688  1.359272  1.359272
platelets                  -0.013908  0.411120  6.008180
serum_creatinine           -0.284552  0.005926  7.752020
serum_sodium                0.085034  0.766064  2.582144
sex                         0.735688  0.735688  0.735688
smoking                    -0.687682  1.454161  1.454161
time                       -0.196954  0.938759  1.997038

X_train, X_test, y_train,y_test =
train_test_split(X_scaled,y,test_size=0.30,random_state=25)
```

Model Building

Support Vector Machine (SVM)

```
model1 = svm.SVC()

# Fitting the model
model1.fit(X_train, y_train)

# Predicting the test variables
y_pred = model1.predict(X_test)

# Getting the score and rounding it to decimal places
score = round(model1.score(X_test, y_test), 3)
score
```

```
0.789
```

Classification Report

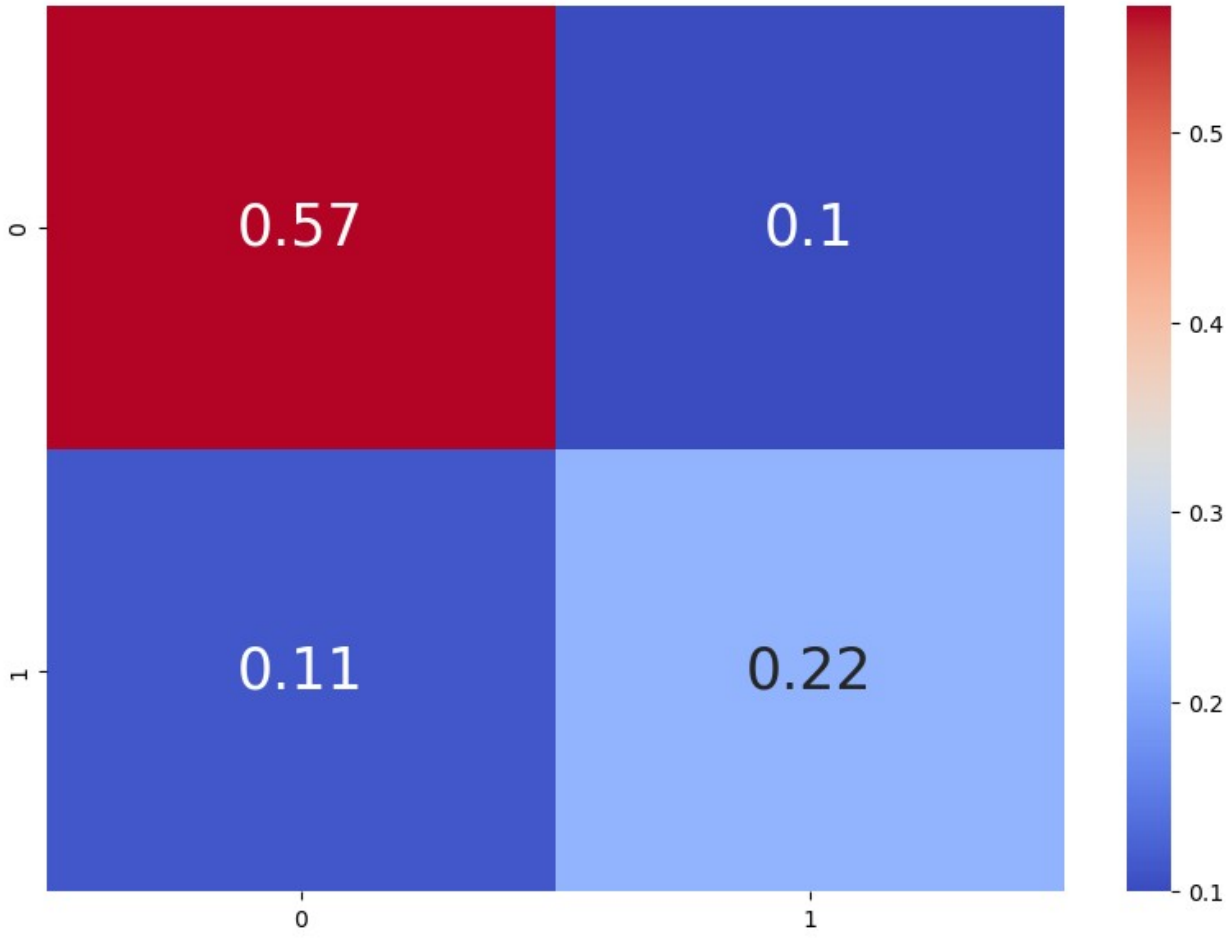
```
report = classification_report(y_test, y_pred, output_dict=True)

# Convert classification report to DataFrame
df_report = pd.DataFrame(report).transpose()

print(df_report)
```

```
              precision  recall  f1-score  support
0                  0.84    0.85      0.84    60.00
1                  0.69    0.67      0.68    30.00
accuracy           0.79    0.79      0.79     0.79
macro avg          0.76    0.76      0.76    90.00
weighted avg       0.79    0.79      0.79    90.00
```

Confusion Matrix

```
cmap1 = 'coolwarm'  # Change the colormap here

# confusion matrix
cf_matrix = confusion_matrix(y_test, y_pred)

# heatmap
plt.subplots(figsize=(10, 7))
sns.heatmap(cf_matrix / np.sum(cf_matrix), cmap=cmap1, annot=True,
```

```
annot_kws={'size': 25})
plt.show()
```



Artificial Neural Network (ANN)

```
# early stopping
early_stopping = callbacks.EarlyStopping(
    min_delta=0.001,  # Minimum amount of change to count as an
improvement
    patience=20,      # How many epochs to wait before stopping
    restore_best_weights=True
)

# Initialize the model
model = Sequential([
    Dense(units=16, kernel_initializer='uniform', activation='relu',
```

```
input_dim=12),
    Dense(units=8, kernel_initializer='uniform', activation='relu'),
    Dropout(0.25),
    Dense(units=8, kernel_initializer='uniform', activation='relu'),
    Dropout(0.5),
    Dense(units=1, kernel_initializer='uniform', activation='sigmoid')
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

# Train the model with early stopping
history = model.fit(X_train, y_train, batch_size=25, epochs=80,
callbacks=[early_stopping], validation_split=0.25)
```

```
Epoch 1/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 4s 78ms/step - accuracy: 0.6586 - loss:
0.6930 - val_accuracy: 0.8302 - val_loss: 0.6908
Epoch 2/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.6114 - loss:
0.6923 - val_accuracy: 0.8302 - val_loss: 0.6889
Epoch 3/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6282 - loss:
0.6914 - val_accuracy: 0.8302 - val_loss: 0.6863
Epoch 4/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.6187 - loss:
0.6907 - val_accuracy: 0.8302 - val_loss: 0.6836
Epoch 5/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.6196 - loss:
0.6897 - val_accuracy: 0.8302 - val_loss: 0.6805
Epoch 6/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6357 - loss:
0.6881 - val_accuracy: 0.8302 - val_loss: 0.6771
Epoch 7/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6738 - loss:
0.6854 - val_accuracy: 0.8302 - val_loss: 0.6733
Epoch 8/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6262 - loss:
0.6858 - val_accuracy: 0.8302 - val_loss: 0.6699
Epoch 9/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6361 - loss:
0.6834 - val_accuracy: 0.8302 - val_loss: 0.6656
Epoch 10/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6803 - loss:
0.6792 - val_accuracy: 0.8302 - val_loss: 0.6598
Epoch 11/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.6507 - loss:
0.6767 - val_accuracy: 0.8302 - val_loss: 0.6539
Epoch 12/80
```

```
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6403 - loss:
0.6747 - val_accuracy: 0.8302 - val_loss: 0.6463
Epoch 13/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.6622 - loss:
0.6671 - val_accuracy: 0.8302 - val_loss: 0.6362
Epoch 14/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.5744 - loss:
0.6709 - val_accuracy: 0.8302 - val_loss: 0.6240
Epoch 15/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6334 - loss:
0.6527 - val_accuracy: 0.8302 - val_loss: 0.6072
Epoch 16/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6405 - loss:
0.6369 - val_accuracy: 0.8302 - val_loss: 0.5880
Epoch 17/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6550 - loss:
0.6250 - val_accuracy: 0.8302 - val_loss: 0.5673
Epoch 18/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6095 - loss:
0.6158 - val_accuracy: 0.8302 - val_loss: 0.5417
Epoch 19/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.6438 - loss:
0.6010 - val_accuracy: 0.8302 - val_loss: 0.5127
Epoch 20/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6153 - loss:
0.6019 - val_accuracy: 0.8302 - val_loss: 0.4847
Epoch 21/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6618 - loss:
0.5614 - val_accuracy: 0.8302 - val_loss: 0.4546
Epoch 22/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7064 - loss:
0.5450 - val_accuracy: 0.8302 - val_loss: 0.4300
Epoch 23/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6169 - loss:
0.5775 - val_accuracy: 0.8302 - val_loss: 0.4095
Epoch 24/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7185 - loss:
0.5351 - val_accuracy: 0.8302 - val_loss: 0.3845
Epoch 25/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.7196 - loss:
0.5268 - val_accuracy: 0.8113 - val_loss: 0.3620
Epoch 26/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7778 - loss:
0.4889 - val_accuracy: 0.8491 - val_loss: 0.3372
Epoch 27/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7277 - loss:
0.4891 - val_accuracy: 0.8491 - val_loss: 0.3168
Epoch 28/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7073 - loss:
```

```
0.5500 - val_accuracy: 0.8302 - val_loss: 0.3049
Epoch 29/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.7123 - loss:
0.5112 - val_accuracy: 0.9057 - val_loss: 0.2991
Epoch 30/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.7826 - loss:
0.4683 - val_accuracy: 0.8679 - val_loss: 0.2924
Epoch 31/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.7743 - loss:
0.4725 - val_accuracy: 0.8868 - val_loss: 0.2891
Epoch 32/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7287 - loss:
0.4936 - val_accuracy: 0.8868 - val_loss: 0.2896
Epoch 33/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7695 - loss:
0.4613 - val_accuracy: 0.8868 - val_loss: 0.2923
Epoch 34/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.7535 - loss:
0.4298 - val_accuracy: 0.8868 - val_loss: 0.2871
Epoch 35/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.7648 - loss:
0.4486 - val_accuracy: 0.8868 - val_loss: 0.2814
Epoch 36/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.6766 - loss:
0.4479 - val_accuracy: 0.8868 - val_loss: 0.2723
Epoch 37/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7597 - loss:
0.4271 - val_accuracy: 0.8868 - val_loss: 0.2615
Epoch 38/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6688 - loss:
0.4818 - val_accuracy: 0.8868 - val_loss: 0.2540
Epoch 39/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7032 - loss:
0.4916 - val_accuracy: 0.8868 - val_loss: 0.2510
Epoch 40/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7214 - loss:
0.4186 - val_accuracy: 0.9057 - val_loss: 0.2513
Epoch 41/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.8023 - loss:
0.3915 - val_accuracy: 0.8868 - val_loss: 0.2507
Epoch 42/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7592 - loss:
0.4358 - val_accuracy: 0.9057 - val_loss: 0.2521
Epoch 43/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7834 - loss:
0.4318 - val_accuracy: 0.8868 - val_loss: 0.2515
Epoch 44/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.7284 - loss:
0.4225 - val_accuracy: 0.8868 - val_loss: 0.2471
```

```
Epoch 45/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6947 - loss:
0.4430 - val_accuracy: 0.8868 - val_loss: 0.2470
Epoch 46/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.7055 - loss:
0.4364 - val_accuracy: 0.8868 - val_loss: 0.2463
Epoch 47/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7097 - loss:
0.4391 - val_accuracy: 0.8868 - val_loss: 0.2460
Epoch 48/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7088 - loss:
0.4672 - val_accuracy: 0.8302 - val_loss: 0.2482
Epoch 49/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7563 - loss:
0.4048 - val_accuracy: 0.8302 - val_loss: 0.2478
Epoch 50/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7371 - loss:
0.4659 - val_accuracy: 0.8302 - val_loss: 0.2469
Epoch 51/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7507 - loss:
0.3866 - val_accuracy: 0.8113 - val_loss: 0.2436
Epoch 52/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6993 - loss:
0.4599 - val_accuracy: 0.8302 - val_loss: 0.2409
Epoch 53/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.7465 - loss:
0.4309 - val_accuracy: 0.8491 - val_loss: 0.2416
Epoch 54/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.7065 - loss:
0.3989 - val_accuracy: 0.8302 - val_loss: 0.2396
Epoch 55/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.7461 - loss:
0.4579 - val_accuracy: 0.8491 - val_loss: 0.2379
Epoch 56/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7049 - loss:
0.4060 - val_accuracy: 0.8491 - val_loss: 0.2343
Epoch 57/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7002 - loss:
0.4553 - val_accuracy: 0.8491 - val_loss: 0.2318
Epoch 58/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.7389 - loss:
0.3902 - val_accuracy: 0.8491 - val_loss: 0.2297
Epoch 59/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7360 - loss:
0.3603 - val_accuracy: 0.8491 - val_loss: 0.2288
Epoch 60/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.7090 - loss:
0.4241 - val_accuracy: 0.8679 - val_loss: 0.2247
Epoch 61/80
```

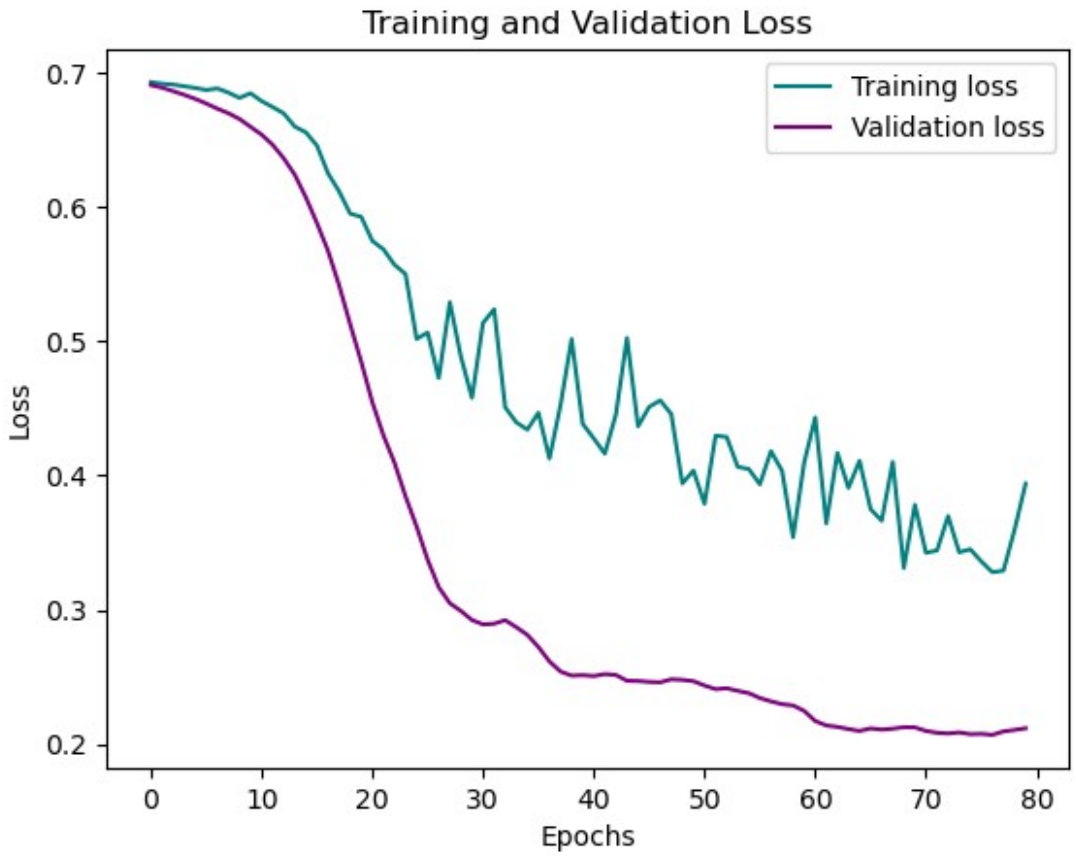
```
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6522 - loss:
0.4268 - val_accuracy: 0.8679 - val_loss: 0.2172
Epoch 62/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7569 - loss:
0.3698 - val_accuracy: 0.8679 - val_loss: 0.2139
Epoch 63/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.6978 - loss:
0.4270 - val_accuracy: 0.8679 - val_loss: 0.2128
Epoch 64/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6830 - loss:
0.4078 - val_accuracy: 0.8868 - val_loss: 0.2112
Epoch 65/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.7574 - loss:
0.3873 - val_accuracy: 0.8868 - val_loss: 0.2098
Epoch 66/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.8046 - loss:
0.3609 - val_accuracy: 0.8679 - val_loss: 0.2117
Epoch 67/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.8450 - loss:
0.3943 - val_accuracy: 0.8491 - val_loss: 0.2108
Epoch 68/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7882 - loss:
0.3772 - val_accuracy: 0.8491 - val_loss: 0.2115
Epoch 69/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.8425 - loss:
0.3624 - val_accuracy: 0.8491 - val_loss: 0.2126
Epoch 70/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.8585 - loss:
0.3462 - val_accuracy: 0.8491 - val_loss: 0.2124
Epoch 71/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.8326 - loss:
0.3418 - val_accuracy: 0.8491 - val_loss: 0.2098
Epoch 72/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.8737 - loss:
0.3248 - val_accuracy: 0.8491 - val_loss: 0.2084
Epoch 73/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.8296 - loss:
0.3702 - val_accuracy: 0.8491 - val_loss: 0.2079
Epoch 74/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.8380 - loss:
0.3893 - val_accuracy: 0.8491 - val_loss: 0.2087
Epoch 75/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.8501 - loss:
0.3979 - val_accuracy: 0.8491 - val_loss: 0.2074
Epoch 76/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.8555 - loss:
0.3476 - val_accuracy: 0.8491 - val_loss: 0.2076
Epoch 77/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.8510 - loss:
```

```
0.3451 - val_accuracy: 0.8491 - val_loss: 0.2069
Epoch 78/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.8400 - loss:
0.3393 - val_accuracy: 0.8491 - val_loss: 0.2094
Epoch 79/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.8605 - loss:
0.3589 - val_accuracy: 0.8491 - val_loss: 0.2105
Epoch 80/80
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.8726 - loss:
0.3208 - val_accuracy: 0.8491 - val_loss: 0.2118
```

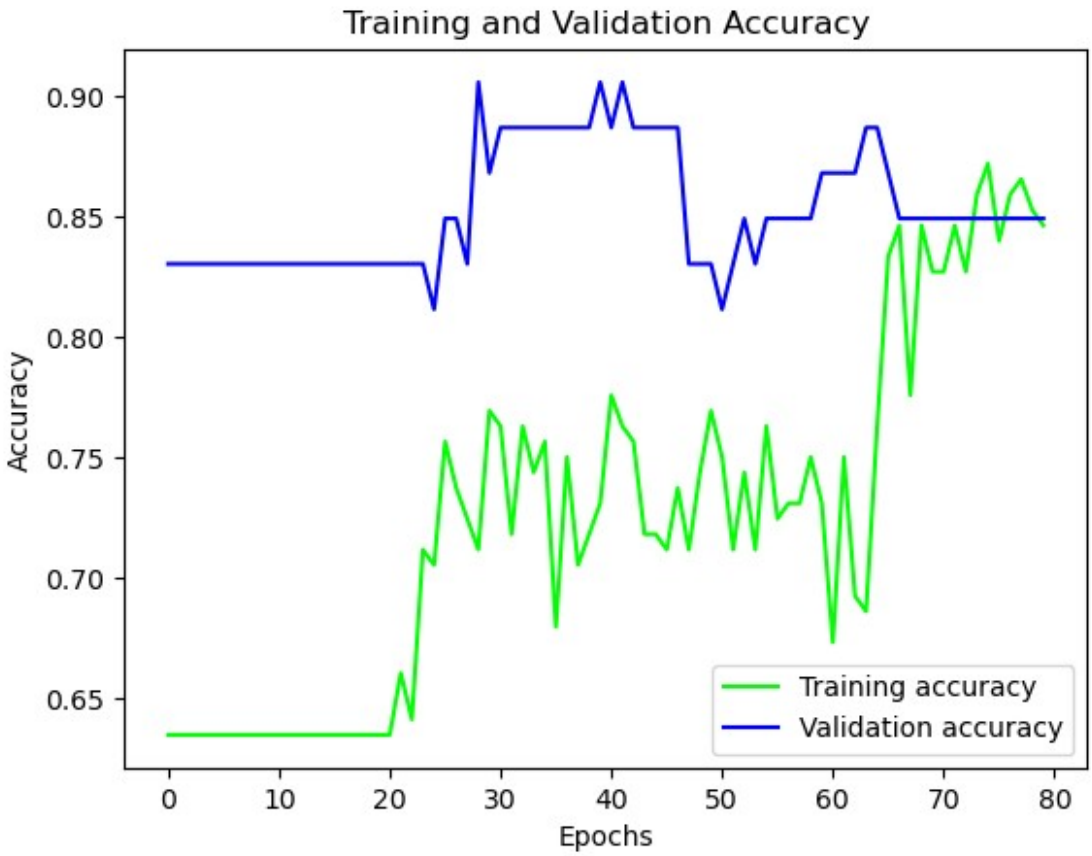
```
val_accuracy = np.mean(history.history['val_accuracy'])
print("\n%s: %.2f%%" % ('val_accuracy is', val_accuracy*100))
```

```
val_accuracy is: 85.26%
```

```
history_df = pd.DataFrame(history.history)

# training and validation loss
plt.plot(history_df['loss'], "#008080", label='Training loss')  # Teal
color
plt.plot(history_df['val_loss'], "#800080", label='Validation loss')
# Purple color
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(loc="best")
plt.show()
```



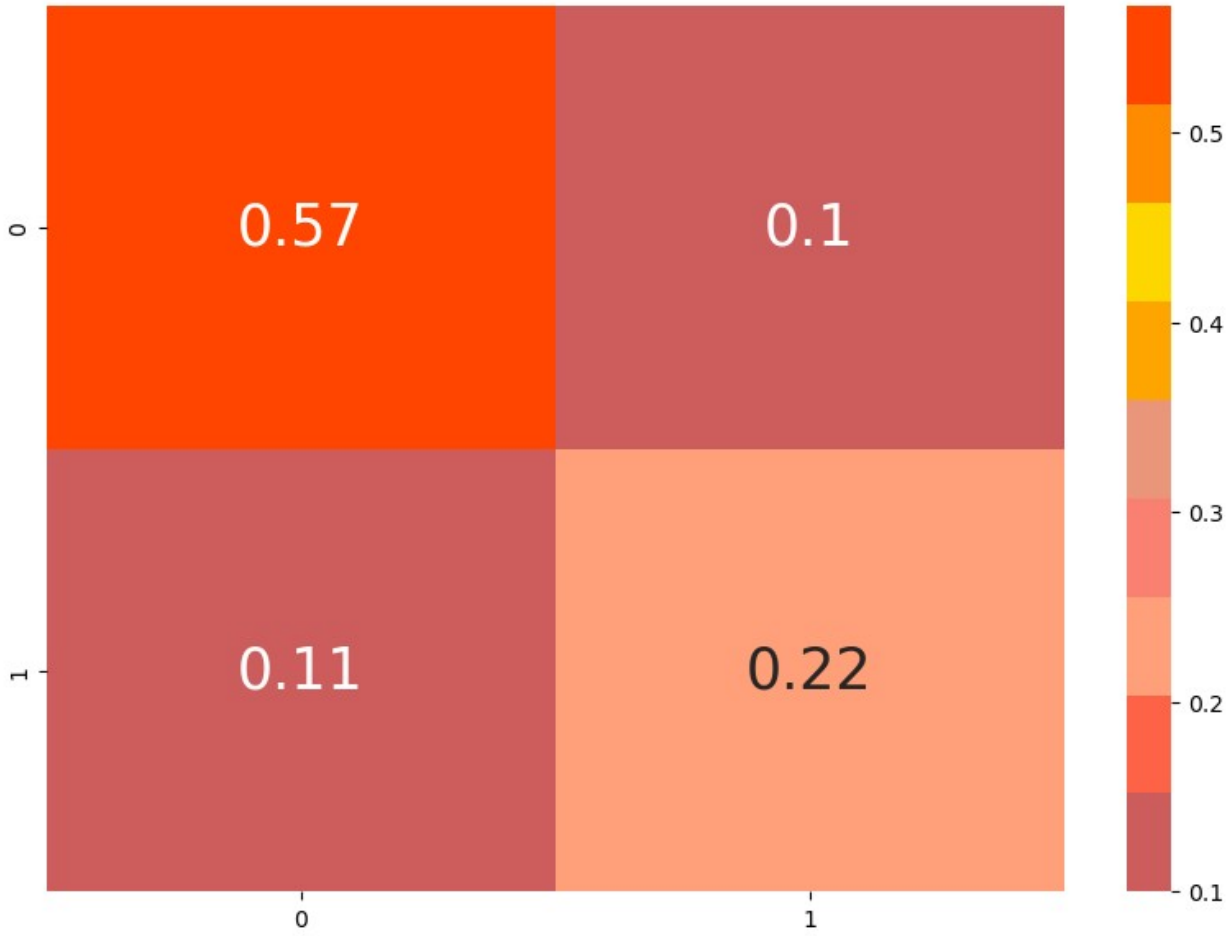

```
history_df = pd.DataFrame(history.history)

# training and validation accuracy
plt.plot(history_df['accuracy'], "#00FF00", label='Training accuracy')
# Green color
plt.plot(history_df['val_accuracy'], "#0000FF", label='Validation
accuracy')  # Blue color
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```




```
y_pred = model.predict(X_test)
y_pred = (y_pred > 0.4)
np.set_printoptions()
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step
```

```
custom_colors = ["#CD5C5C", "#FF6347", "#FFA07A", "#FA8072",
"#E9967A", "#FFA500", "#FFD700", "#FF8C00", "#FF4500"]

# confusion matrix
plt.subplots(figsize=(10, 7))
sns.heatmap(cf_matrix / np.sum(cf_matrix), cmap=custom_colors,
annot=True, annot_kws={'size': 25})
plt.show()
```




```
# classification report
report = classification_report(y_test, y_pred, output_dict=True)

# classification report to DataFrame
report_df = pd.DataFrame(report).transpose()

print(report_df)
```

```
              precision  recall  f1-score  support
0                  0.85    0.75      0.80    60.00
1                  0.59    0.73      0.66    30.00
accuracy           0.74    0.74      0.74     0.74
macro avg          0.72    0.74      0.73    90.00
weighted avg       0.76    0.74      0.75    90.00
```