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

```
data=pd.read_csv(r"C:\Users\mruna\Downloads\
heart_failure_clinical_records_dataset.csv")
```

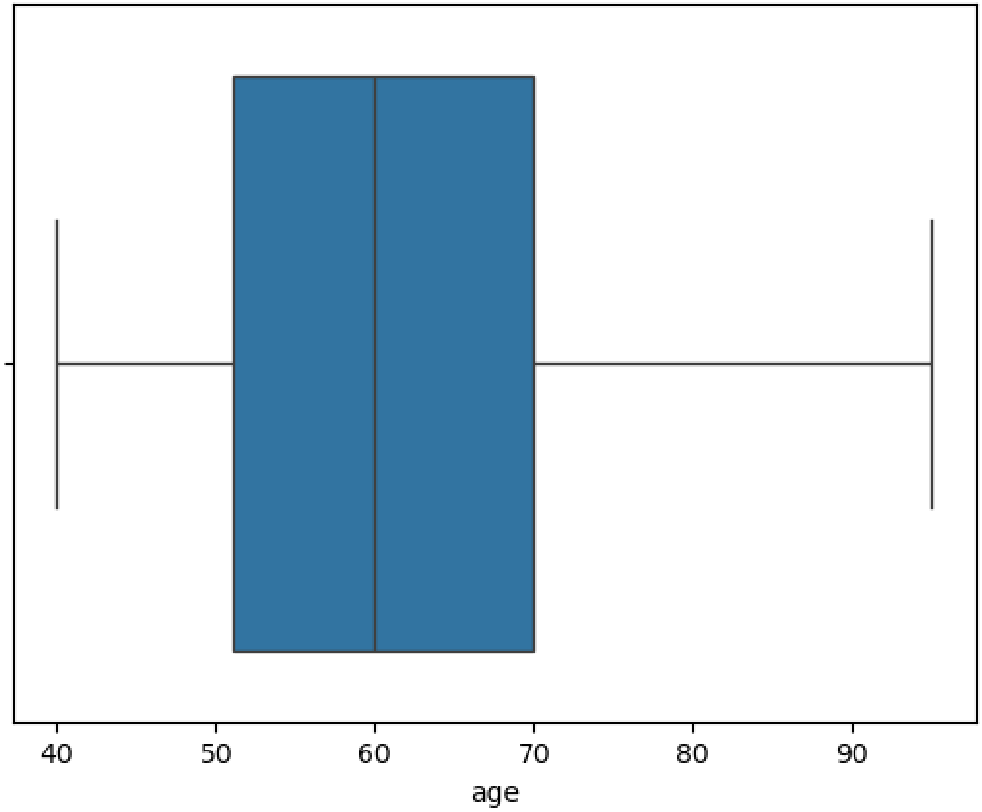
```
data.head()
```

```
    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
```

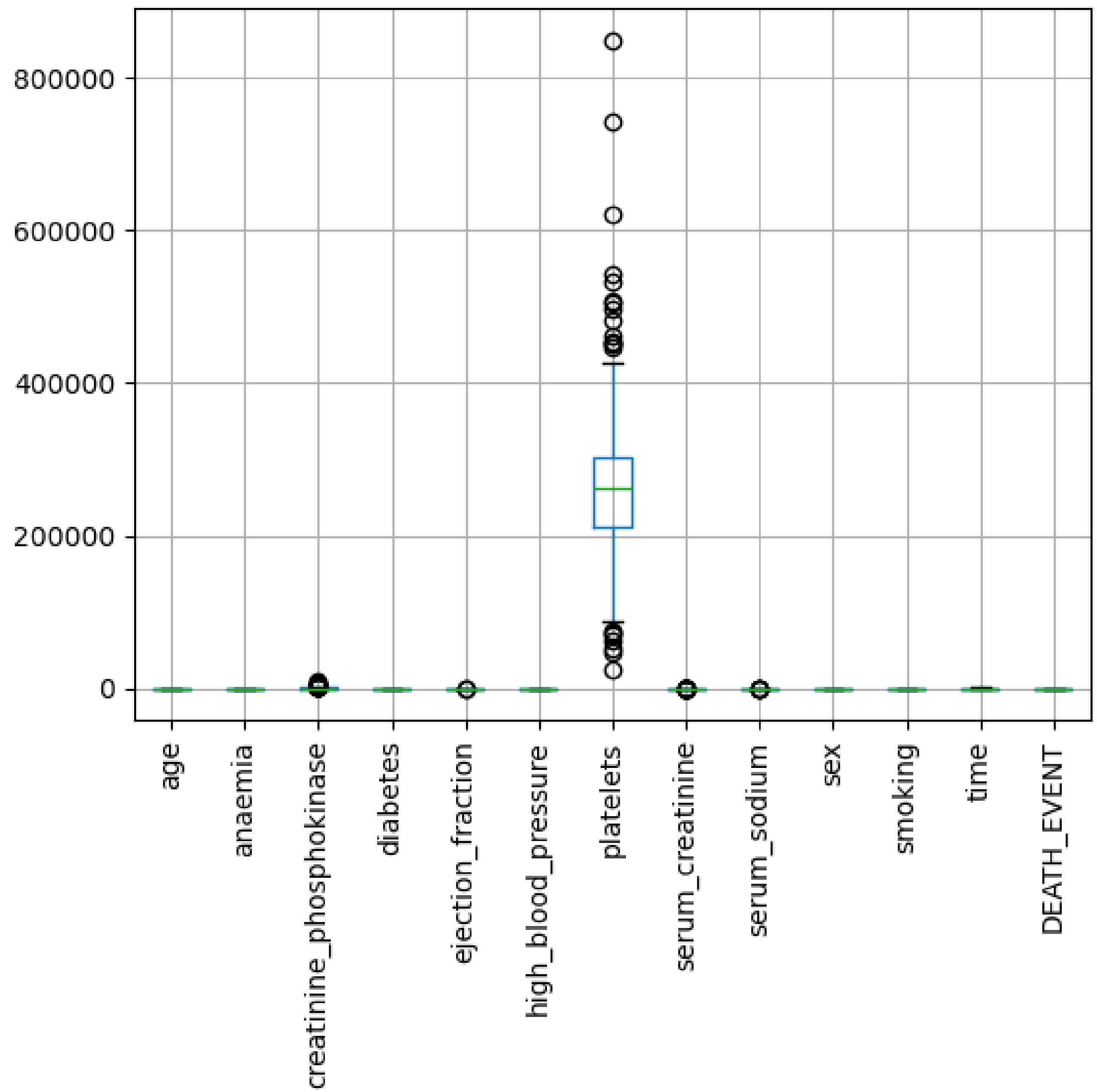
```
numeric_col=data.select_dtypes(include=['int','float']).columns
```

```
sns.boxplot(data=data,x='age')
```

```
<Axes: xlabel='age'>
```

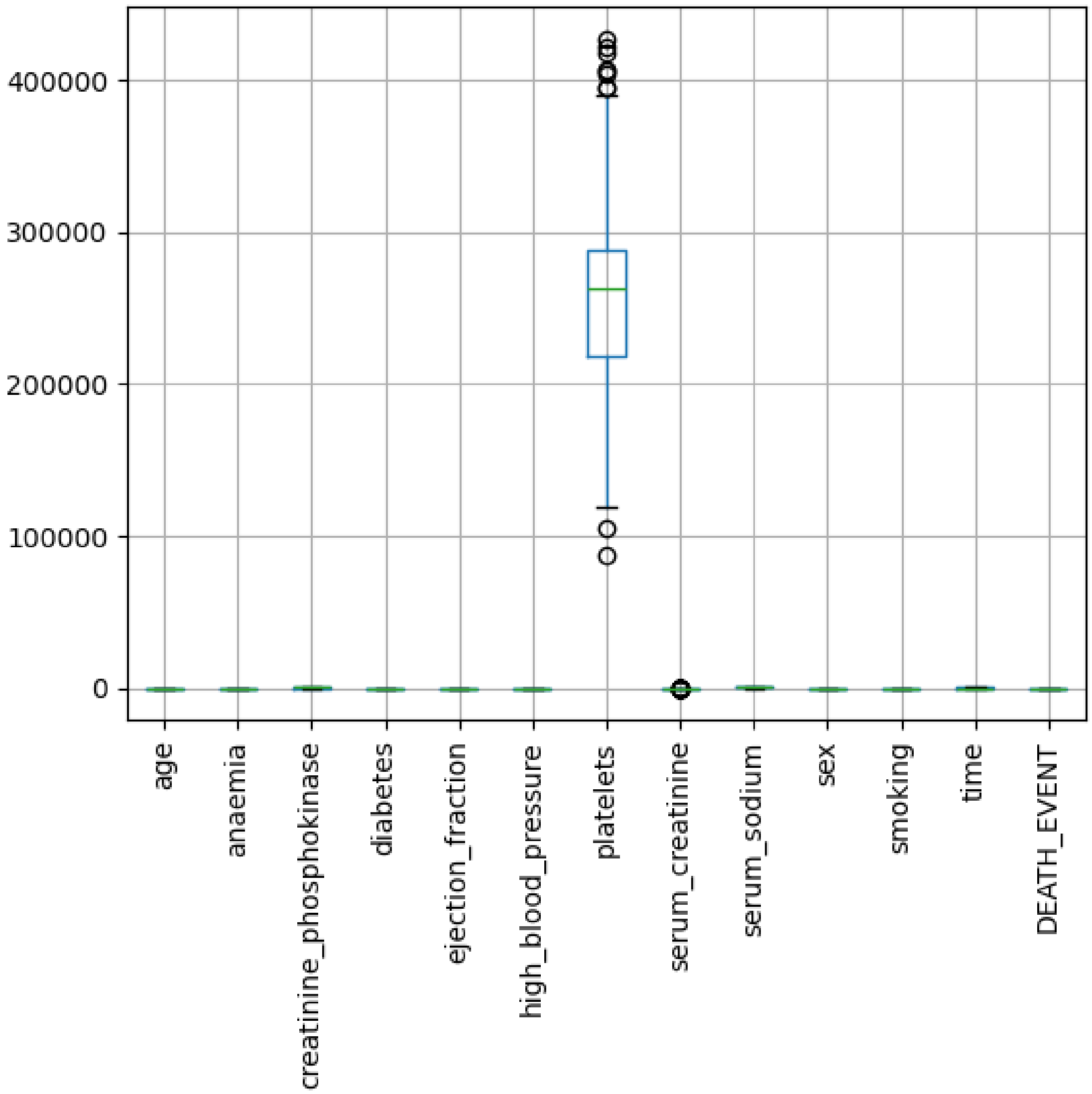


```
data.boxplot()
plt.xticks(rotation=90)
plt.show()
```



```
#Outlier Treatment

def OT(data,col):
    Q1=data[col].quantile(0.25)
    Q3=data[col].quantile(0.75)
    IQR=Q3-Q1
    UW=Q3+1.5*IQR
    LW=Q1-1.5*IQR
    upper_outlier=data[col]>UW
    lower_outlier=data[col]<LW
    data.loc[upper_outlier,col]=data[col].median()
    data.loc[lower_outlier,col]=data[col].median()
    return data

for i in data.select_dtypes(['int','float']):
    OT(data,i)
```

```
data.boxplot()
plt.xticks(rotation=90)
plt.show()
```



```
#Null Value Teatment

data.isnull().sum()
```

```
age                         0
anaemia                     0
creatinine_phosphokinase    0
diabetes                    0
ejection_fraction           0
high_blood_pressure         0
platelets                   0
```

```
serum_creatinine            0
serum_sodium                0
sex                         0
smoking                     0
time                        0
DEATH_EVENT                 0
dtype: int64
```

```
#Skewness
```

```
data.skew()
```

```
age                         0.423062
anaemia                     0.278261
creatinine_phosphokinase    1.157500
diabetes                    0.333929
ejection_fraction           0.441554
high_blood_pressure         0.626732
platelets                   0.172135
serum_creatinine            1.041033
serum_sodium               -0.127243
sex                        -0.626732
smoking                     0.770349
time                        0.127803
DEATH_EVENT                 0.770349
dtype: float64
```

```
#Building the model
```

```
X=data.drop('DEATH_EVENT',axis=1)
y=data.DEATH_EVENT
```

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(X,y,train_size=.70,rand
om_state=42)
```

```
#Classification Model
```

```
from sklearn.linear_model import LogisticRegression
```

```
LR=LogisticRegression()
```

```
LR.fit(x_train,y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\
_logistic.py:469: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression()
```

```
Predictions=LR.predict(x_test)
```

```
from sklearn.metrics import accuracy_score
```

```
accuracy_score(y_test,Predictions)
```

```
0.8
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test,Predictions))
```

```
              precision    recall  f1-score   support

           0       0.77      0.94      0.85        53
           1       0.88      0.59      0.71        37

    accuracy                           0.80        90
   macro avg       0.82      0.77      0.78        90
weighted avg       0.81      0.80      0.79        90
```