In [163…
```
import pandas as pd
import numpy as np
import sklearn
import matplotlib.pyplot as plt
from sklearn.model_selection \
import train_test_split
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler , LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
```

Question 1:

1. load the data into Pandas dataframe. Extract two dataframes with the above 4 features: df 0 for surviving patients (DEATH EVENT = 0) and df 1 for deceased patients (DEATH EVENT = 1)

In [164…
```
df=pd.read_csv("heart_failure_clinical_records_dataset.csv")
df_1=df.loc[(df['DEATH_EVENT']==1),['creatinine_phosphokinase','serum_creatinir
df_0=df.loc[(df['DEATH_EVENT']==0),['creatinine_phosphokinase','serum_creatinir
print(df_0.head())
print(df_1.head())
```

```
    creatinine_phosphokinase  serum_creatinine  serum_sodium  platelets  \
14                        80               1.0           138   427000.0
20                        52               1.3           137   276000.0
23                        63               0.8           135   368000.0
33                       159               1.2           138   302000.0
38                      2656               2.3           137   305000.0

    DEATH_EVENT
14            0
20            0
23            0
33            0
38            0
   creatinine_phosphokinase  serum_creatinine  serum_sodium  platelets  \
0                       582               1.9           130  265000.00
1                      7861               1.1           136  263358.03
2                       146               1.3           129  162000.00
3                       111               1.9           137  210000.00
4                       160               2.7           116  327000.00

   DEATH_EVENT
0            1
1            1
2            1
3            1
4            1
```

1. for each dataset, construct the visual representations of correponding correlation matrices M0 (from df 0) and M1 (from df 1) and save the plots into two separate files

In [165…
```
print(df_1.corr())
print(df_0.corr())
```

```
                          creatinine_phosphokinase  serum_creatinine  \
creatinine_phosphokinase                  1.000000         -0.033407
serum_creatinine                         -0.033407          1.000000
serum_sodium                              0.148823         -0.094011
platelets                                 0.078808         -0.029384
DEATH_EVENT                                    NaN               NaN

                          serum_sodium  platelets  DEATH_EVENT
creatinine_phosphokinase      0.148823   0.078808          NaN
serum_creatinine             -0.094011  -0.029384          NaN
serum_sodium                  1.000000   0.141284          NaN
platelets                     0.141284   1.000000          NaN
DEATH_EVENT                        NaN        NaN          NaN
                          creatinine_phosphokinase  serum_creatinine  \
creatinine_phosphokinase                  1.000000         -0.043110
serum_creatinine                         -0.043110          1.000000
serum_sodium                             -0.002474         -0.215651
platelets                                -0.012940         -0.031217
DEATH_EVENT                                    NaN               NaN

                          serum_sodium  platelets  DEATH_EVENT
creatinine_phosphokinase     -0.002474  -0.012940          NaN
serum_creatinine             -0.215651  -0.031217          NaN
serum_sodium                  1.000000   0.001807          NaN
platelets                     0.001807   1.000000          NaN
DEATH_EVENT                        NaN        NaN          NaN
```

1. examine your correlation matrix plots visually and answer the following:

(a) which features have the highest correlation for surviving patients?

the highest correlation for surviving patiets are serum sodium and serum serum creatinine

(b) which features have the lowest correlation for surviving patients?

the lowest correlation for suriving patients is between platelets and serum sodium

(c) which features have the highest correlation for deceased patients?

the highest correlation for deceased patients are between creatinine_phosphokinase and serum_sodium

(d) which features have the lowest correlation for deceased patients?

lowest correlation for deceased patients are between platelets and serum creatinine

(e) are results the same for both cases?

No

Question 2: In this question you will compare a number of different models using linear systems (including linear regres- sion). You choose one feature X as independent variable X and another feature Y as dependent. Your choice of X and Y will depend on your facilitator group as follows:

BUID:U56938298

1. Group 4: X: platelets, Y : serum creatinine

In [166…
```
X_0=df_0['platelets']
Y_0=df_0['serum_creatinine']

X_1=df_1['platelets']
Y_1=df_1['serum_creatinine']
```

1. y = ax + b (simple linear regression)

(a) fit the model on Xtrain

(b) print the weights (a, b, . . .)

(c) compute predicted values using Xtest

(d) plot (if possible) predicted and actual values in Xtest

(e) compute (and print) the corresponding loss function

In [167…
```
from sklearn.metrics import mean_squared_error
# degree = 1
# #(a)
# weights_0 = np.polyfit(X_0_train,y_0_train, degree)
# #(b)computer weights
# model = np.poly1d(weights_0)
# print('Question (b), wights:',weights_0)
# #(c) Moldel
# prodict=model(X_0_test)
# #print('Question (c)',prodict)
# #(d)
# print("(d)")
# x_points = np.linspace(0.75,9.5, 1000)
# y_points = model(x_points)
# ax, fig = plt.subplots()
# plt.xlim(0, 10)
# plt.ylim(0, 10)
# plt.xlabel('X')
# plt.ylabel('Y', rotation=0)
# plt.plot(x_points , y_points , lw=4, color='blue')
```

```
# plt.scatter(X_0, Y_0, color='black', s=200)
# # for i, txt in enumerate(id_list):
# #     plt.text(x[i]+0.2, y[i]+0.2, txt, fontsize=10)
# plt.show()
# #(e)
# print('(e)',mean_squared_error(y_0_test,prodict)*len(y_0_test))

def sse(X,Y,degree):
    X_train,X_test,y_train,y_test=train_test_split(X,Y, train_size=0.5)
    #(a)
    #(b)computer weights
    weights= np.polyfit(X_train,y_train, degree)
    print('(b) wights:',weights)
    model = np.poly1d(weights)
    #(c) Moldel
    prodict=model(X_0_test)
    ##print('Question (c)',prodict)
    # #(d)
    # print("(d)")
    # x_points = np.linspace(0.75,9.5, 1000)
    # y_points = model(x_points)
    # ax, fig = plt.subplots()
    # plt.xlim(0, 10)
    # plt.ylim(0, 10)
    # plt.xlabel('X')
    # plt.ylabel('Y', rotation=0)
    # plt.plot(x_points , y_points , lw=4, color='blue')
    # plt.scatter(X_0, Y_0, color='black', s=200)
    # # for i, txt in enumerate(id_list):
    # #     plt.text(x[i]+0.2, y[i]+0.2, txt, fontsize=10)
    # plt.show()
    print('(e) the corresponding loss function',mean_squared_error(y_0_test,pro
    return mean_squared_error(y_0_test,prodict)*len(y_0_test)
print('1. y = ax + b (simple linear regression)')
print('when the dgree =1, the suriving patients :')
sse_0_1=sse(X_0,Y_0,1)
print('when the dgree =1, the deceased patients :')
sse_1_1=sse(X_1,Y_1,1)
print('\n')

print('2. y = ax2 + bx + c (quadratic)')
print('when the dgree =2, the suriving patients :')
sse_0_2=sse(X_0,Y_0,2)
print('when the dgree =2, the deceased patients :')
sse_1_2=sse(X_1,Y_1,2)
print('\n')

print('3. y = ax3 + bx2 + cx + d (cubic spline)')
print('when the dgree =3, the suriving patients :')
sse_0_3=sse(X_0,Y_0,3)
print('when the dgree =3, the deceased patients :')
sse_1_3=sse(X_1,Y_1,3)
print('\n')

print('4. y = a log x + b (GLM - generalized linear model)')
print('when the dgree =1, the suriving patients :')
sse_0_log_x=sse(np.log(X_0),Y_0,1)
print('when the dgree =1, the deceased patients :')
sse_1_log_x=sse(np.log(X_1),Y_1,1)
print('\n')
```

```
print('5. log y = a log x + b (GLM - generalized linear model))')
print('when the dgree =1, the suriving patients :')
sse_0_log_xy=sse(np.log(X_0),np.log(Y_0),1)
print('when the dgree =1, the deceased patients :')
sse_1_log_xy=sse(np.log(X_1),np.log(Y_1),1)
print('\n')
```

```
1. y = ax + b (simple linear regression)
when the dgree =1, the suriving patients :
(b) wights: [-4.47796721e-07  1.31842865e+00]
(e) the corresponding loss function 19.376530222100463
when the dgree =1, the deceased patients :
(b) wights: [1.29005515e-06 1.15205306e+00]
(e) the corresponding loss function 34.140438902227274


2. y = ax2 + bx + c (quadratic)
when the dgree =2, the suriving patients :
(b) wights: [ 6.96903533e-12 -3.72291338e-06  1.63265169e+00]
(e) the corresponding loss function 17.978409612848324
when the dgree =2, the deceased patients :
(b) wights: [ 7.38870133e-12 -5.17021092e-06  2.62938093e+00]
(e) the corresponding loss function 72.88730378659686


3. y = ax3 + bx2 + cx + d (cubic spline)
when the dgree =3, the suriving patients :
(b) wights: [ 1.50273956e-17 -1.87249486e-11  5.48271405e-06  9.13705738e-01]
(e) the corresponding loss function 24.880934145001365
when the dgree =3, the deceased patients :
(b) wights: [-1.36959859e-17  2.88674122e-11 -1.63711295e-05  4.20360390e+00]
(e) the corresponding loss function 77.22527996194415


4. y = a log x + b (GLM - generalized linear model)
when the dgree =1, the suriving patients :
(b) wights: [0.0691848  0.29460395]
(e) the corresponding loss function 36843810261.09107
when the dgree =1, the deceased patients :
(b) wights: [-0.50651063  8.16945522]
(e) the corresponding loss function 1974757500318.8518


5. log y = a log x + b (GLM - generalized linear model))
when the dgree =1, the suriving patients :
(b) wights: [ 0.01376523 -0.13532686]
(e) the corresponding loss function 1457718214.1176505
when the dgree =1, the deceased patients :
(b) wights: [-0.03774469  0.85690621]
(e) the corresponding loss function 10967572897.571415
```

In [168…

```
print(X_0,np.log(X_0))
```

```
14     427000.0
20     276000.0
23     368000.0
33     302000.0
38     305000.0
         ...
294    155000.0
295    270000.0
296    742000.0
297    140000.0
298    395000.0
Name: platelets, Length: 203, dtype: float64 14     12.964539
20     12.528156
23     12.815838
33     12.618182
38     12.628067
         ...
294    11.951180
295    12.506177
296    13.517105
297    11.849398
298    12.886641
Name: platelets, Length: 203, dtype: float64
```

Question 3: Summarize your results from question 2 in a table like shown below:

In [169…

```
Q3_res=pd.DataFrame({
    'Model':["y = ax + b","y = ax2 + bx + c","y = ax3 + bx2 + cx + d",'y = a lo
    'SSE (death event=0)':[sse_0_1,sse_0_2,sse_0_3,sse_0_log_x,sse_0_log_xy],
    '(death event=1)':[sse_1_1,sse_1_2,sse_1_3,sse_1_log_x,sse_1_log_xy]
})
print(Q3_res)
```

```
                   Model  SSE (death event=0)  (death event=1)
0             y = ax + b         1.937653e+01     3.414044e+01
1         y = ax2 + bx + c         1.797841e+01     7.288730e+01
2  y = ax3 + bx2 + cx + d         2.488093e+01     7.722528e+01
3          y = a log x + b         3.684381e+10     1.974758e+12
4      log y = a log x + b         1.457718e+09     1.096757e+10
```

1. which model was the best (smallest SSE) for surviving pa- tients? for deceased patients?

Y=ax2+bx+c is best

1. which model was the worst (largest SSE) for surving pa- tients? for deceased patients?

Y=alogx+b is worst