Homework 4

We load `heart_processed.csv` which has log-predictors from the [Heart Failure Clinical Records Dataset (https://archive.ics.uci.edu/ml/datasets/Heart%2Bfailure%2Bclinical%2Brecords)](https://archive.ics.uci.edu/ml/datasets/Heart%2Bfailure%2Bclinical%2Brecords) for predicting `DEATH_EVENT` .

In [1]:
```
import pandas as pd
import numpy as np

dataset = pd.read_csv("heart_processed.csv")
X = dataset.drop("DEATH_EVENT", axis=1)
y = dataset["DEATH_EVENT"]

# convert to numpy arrays
X = X.values
y = y.values

# split the data into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra

# print the shapes of the training and testing sets
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(239, 7)
(239,)
(60, 7)
(60,)
```

In [27]:
```
display(dataset)
```

	Unnamed: 0	age	creatinine_phosphokinase	ejection_fraction	platelets	serum_creatinine	s
0	0	4.317488	6.366470	2.995732	12.487485	0.641854	
1	1	4.007333	8.969669	3.637586	12.481270	0.095310	
2	2	4.174387	4.983607	2.995732	11.995352	0.262364	
3	3	3.912023	4.709530	2.995732	12.254863	0.641854	
4	4	4.174387	5.075174	2.995732	12.697715	0.993252	
...	...	...	...	...	...	...	
294	294	4.127134	4.110874	3.637586	11.951180	0.095310	
295	295	4.007333	7.506592	3.637586	12.506177	0.182322	
296	296	3.806662	7.630461	4.094345	13.517105	-0.223144	
297	297	3.806662	7.788626	3.637586	11.849398	0.336472	
298	298	3.912023	5.278115	3.806662	12.886641	0.470004	

299 rows × 8 columns

[10pts] Write a naive Bayes classifier with priors inferred from the dataset and class-conditional densities inferred using `scipy.stats.gaussian_kde` with default bandwidth. Use only the training data to fit the classification model. Print the training accuracy and testing accuracy.

Hint: Recall that naive Bayes classification involves the (naive) assumption that the features of X are independent

In [26]:
```
from scipy.stats import gaussian_kde
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split


#calculate prior probabilities from data
X = dataset.drop("DEATH_EVENT", axis=1)
y = dataset["DEATH_EVENT"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
priors = y_train.value_counts(normalize=True)
print("Prior probabilities: ", priors)

#calculate class conditional densities using scipy.stats.gaussian_kde
kde = {}
for col in X_train.columns[1:]:
    kde[col] = {}
    for label in y_train.unique():
        kde[col][label] = gaussian_kde(X_train.loc[y==label, col])
#print(kde)

#calculate Train posterior probabilities
posterior = pd.DataFrame(index=y_train.index, columns=y_train.unique())
for label in y_train.unique():
    posterior[label] = priors[label]
    for col in X_train.columns[1:]:
        posterior[label] *= kde[col][label](X_train[col])
#print(posterior)

#predict class with highest posterior probability
y_pred = posterior.idxmax(axis=1)
#print(y_pred)

#print accuracy
print("Training Accuracy: ", accuracy_score(y_train, y_pred))

#testing acc
priors_test = y_test.value_counts(normalize=True)
print("Test Prior probabilities: ", priors_test)

#calculate posterior probabilities
posterior_test = pd.DataFrame(index=y_test.index, columns=y_test.unique())
for label in y_test.unique():
    posterior_test[label] = priors_test[label]
    for col in X_test.columns[1:]:
        posterior_test[label] *= kde[col][label](X_test[col])
#print(posterior_test)

#predict class with highest posterior probability
y_pred_test = posterior_test.idxmax(axis=1)
#print(y_pred_test)

#print accuracy
print("Testing Accuracy: ", accuracy_score(y_test, y_pred_test))
```

```
Prior probabilities:  0    0.702929
1    0.297071
Name: DEATH_EVENT, dtype: float64
Training Accuracy:  0.8075313807531381
Test Prior probabilities:  0    0.583333
1    0.416667
Name: DEATH_EVENT, dtype: float64
Testing Accuracy:  0.65
```