Homework 4

Assignment Info

Homework #: HW4
Description: Naive Bayes Classifier
Course: EN.553.636 Introduction to Data Science
Semester: Spring 2023, Homewood Campus
Instructor: Tamas Budavari
TA: Matthew Tivnan
Date: March 15, 2023

Student Info

Name: Amir Hossein Daraie
JHED-ID: adaraie1
Email: adaraie1@jhu.edu (mailto:adaraie1@jhu.edu)

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) for predicting `DEATH_EVENT`.

Naïve Bayes

y is class.

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

"Naive" assumption that vars are indep. given the class:

$$P(X|y) = P(x_1|y) \cdot P(x_2|y) \cdots P(x_6|y)$$

$$P(y|X) \propto P(x_1|y) \cdot P(x_2|y) \cdots P(x_6|y) \cdot P(y) = P(y) \cdot \prod_{i=1}^{6} P(x_i|y)$$

$$y = \underset{y}{\operatorname{argmax}}\ P(y) \cdot \prod_{i=1}^{6} P(x_i|y)$$

Gaussian NB assumes $P(x_i|y) = \frac{1}{(2\pi\sigma_y^2)^{1/2}} \exp\left(\frac{-1}{2\sigma_y^2}(x_i - \mu_y)^2\right)$

$y \in \{0, 1\}$:

For $y=0$: $P(y=0) \cdot \prod_{i=1}^{6} P(x_i|y=0)$

$$= P(y=0) \cdot \prod_{i=1}^{6} \frac{1}{(2\pi\sigma_y^2)^{1/2}} \exp\left(\frac{-1}{2\sigma_y^2}(x_i - \mu_{y=0})^2\right)$$

$$= P(y=0) \cdot \prod_{i=1}^{6} K_0(x_i)$$

(x_i: train, test; K_0: train)

For $y=1$:

$$P(y=1) \cdot \prod_{i=1}^{6} P(x_i|y=0)$$

$$= P(y=1) \cdot \prod_{i=1}^{6} K_1(x_i)$$

(x_i: "; K_1: train)

Note that multiplication $\prod_{i=1}^{6}$ is done in python's gaussian_kde.

So $y = \underset{y}{\operatorname{argmax}}\ P(y) \cdot K(X)$ → A single $[x_1, \ldots, x_6]$ row of data.

→ Gaussian_kde with params from Train data.

In [1]:

```
import matplotlib.pyplot as plt
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

# drop the first column which are the patient IDs
X = X[:, 1:]

# 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, random_stat

# 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, 6)
(239,)
(60, 6)
(60,)
```

[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 [2]:

```
from scipy.stats import gaussian_kde

# Compute class probabilities P(y=0) and P(y=1)
n = y_train.shape[0]
p_0 = sum(y_train==0) / n
p_1 = sum(y_train==1) / n
```

In [3]:

```
# Compute Gaussian Kernel Density function using variance and mean of our prior data
kde_0 = gaussian_kde(X_train[y_train==0].T)
kde_1 = gaussian_kde(X_train[y_train==1].T)

# Compute the likelihoods for both classes y=0,1
p = np.zeros((239,2))
for i in range(239):
    pi_0 = kde_0.evaluate(X_train[i]) * p_0
    pi_1 = kde_1.evaluate(X_train[i]) * p_1
    p[i,:] = [pi_0[0], pi_1[0]]

# Compute the argmax of likelihood to convert probabilities to classes
y_hat_train = np.argmax(p, axis=1)

# Compute training accuracy
acc_train = np.average(y_train == y_hat_train)
print(f"Training accuracy is {acc_train:.4f}")
```

```
Training accuracy is 0.8996
```

In [4]:

```
# Note: Do not change kde_0 & kde_1 for testing

# Compute the likelihoods for both classes y=0,1
p = np.zeros((60,2))
for i in range(60):
    pi_0 = kde_0.evaluate(X_test[i]) * p_0
    pi_1 = kde_1.evaluate(X_test[i]) * p_1
    p[i,:] = [pi_0[0], pi_1[0]]

# Compute the argmax of likelihood to convert probabilities to classes
y_hat_test = np.argmax(p, axis=1)

# Compute testing accuracy
acc_test = np.average(y_test == y_hat_test)
print(f"Testing accuracy is {acc_test:.4f}")
```

```
Testing accuracy is 0.6167
```