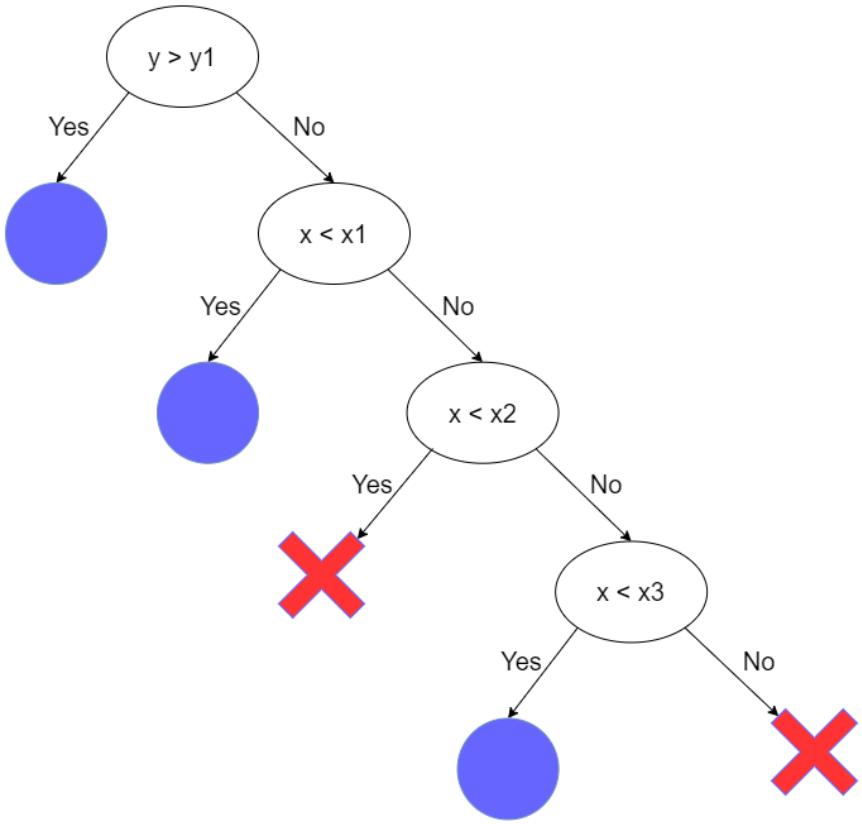
Machine Learning HW #3 answer

1.

Mahalanobis distance is better. Mahalanobis distance is quite effective to find outliers for multivariate data. Euclidean distance is also commonly used to find distance between two points in 2 or more than 2 dimensional space. If we use the Euclidean distance to detect, o1 and o2 will be cluster to C1 which is closer to them. But, it doesn't consider about the dispersion of cluster. MD uses a covariance matrix unlike Euclidean. Because of that, MD works well when two or more variables are highly correlated and even if their scales are not the same. But, when two or more variables are not on the same scale, Euclidean distance results might misdirect.

2.



3.

x	0.9	0.7	1.2	2.4	1.8
$g_1(x)$	0.3970	0.3914	0.3910	0.1497	0.2897
$g_2(x)$	0.2179	0.1714	0.2897	0.3683	0.3910
$B_1(x)$	0.6457	0.6900	0.5744	0.2891	0.4256
$B_2(x)$	0.3543	0.3100	0.4256	0.7109	0.5744

$$\alpha_j = \frac{1}{n}\sum_{i=0}^{n} \beta_j(x_i)$$

$$\mu_j = \frac{\sum_{i=0}^{n} \beta_j(x_i) \cdot x_i}{\sum_{i=0}^{n} \beta_j(x_i)}$$

$$\sigma_j{}^2 = \frac{\sum_{i=0}^{n} \beta_j(x_i) \cdot (x_i - \mu_j)^2}{\sum_{i=0}^{n} \beta_j(x_i)}$$

$$\alpha_1 = \frac{1}{5}(0.6457 + 0.6900 + 0.5744 + 0.2891 + 0.4256) = \frac{2.6248}{5} = 0.52496$$

$$\alpha_2 = \frac{1}{5}(0.3543 + 0.3100 + 0.4256 + 0.7109 + 0.5744) = \frac{2.3752}{5} = 0.47504$$

$$\mu_1 = \frac{0.6457 * 0.9 + 0.6900 * 0.7 + 0.5744 * 1.2 + 0.2891 * 2.4 + 0.4256 * 1.8}{2.6248}$$

$$= \frac{3.2133}{2.6248} = 1.224$$

$$\mu_2 = \frac{0.3543 * 0.9 + 0.3100 * 0.7 + 0.4256 * 1.2 + 0.7109 * 2.4 + 0.5744 * 1.8}{2.3752}$$

$$= \frac{3.7867}{2.3752} = 1.594$$

$$\sigma_1{}^2 = \frac{0.6457 * 0.105 + 0.6900 * 0.275 + 0.5744 * 0.0006 + 0.2891 * 1.383 + 0.4256 * 0.332}{2.6248}$$

$$= \frac{0.7990}{2.6248} = 0.3044$$

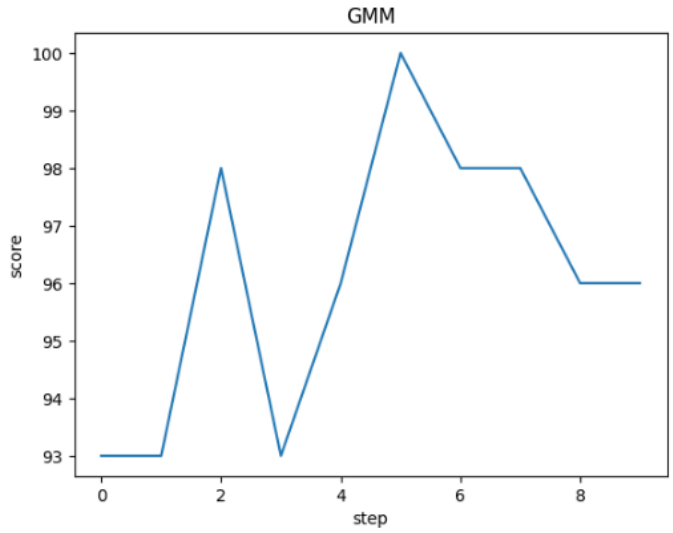
$$\sigma_2{}^2 = \frac{0.3543 * 0.482 + 0.3100 * 0.799 + 0.4256 * 0.155 + 0.7109 * 0.65 + 0.5744 * 0.042}{2.3752}$$

$$= \frac{0.9706}{2.3752} = 0.4087$$

4.

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.mixture import GaussianMixture

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

iris_dataset = datasets.load_iris()

df_X = iris_dataset.data[:,:4]
df_y = iris_dataset.target
```

```
GMM_scores = []

def GMM_classifier(X_train, X_test, y_train, y_test):
    s=[[], [], []]
    for i in range(3):
        X_train_list = X_train[y_train==i, :]
        gmm = GaussianMixture(n_components=2)
        gmm.fit(X_train_list)
        s[i] = gmm.score_samples(X_test)
    predict_0 = np.logical_and((s[0]>s[1]), (s[0]>s[2]))
    predict_1 = np.logical_and((s[1]>s[0]), (s[1]>s[2]))
    predict_2 = np.logical_and((s[2]>s[1]), (s[2]>s[0]))

    accuracy_0 = np.logical_and(predict_0, y_test==0)
    accuracy_1 = np.logical_and(predict_1, y_test==1)
    accuracy_2 = np.logical_and(predict_2, y_test==2)

    return round((sum(accuracy_0)+sum(accuracy_1)+sum(accuracy_2)) / len(y_test) * 100)

for i in range(10):
        X_train, X_test, y_train, y_test = train_test_split(df_X, df_y, test_size=0.3)

        GMM_scores.append(GMM_classifier(X_train, X_test, y_train, y_test))

plt.plot(GMM_scores)
plt.title('GMM')
plt.xlabel('step')
plt.ylabel('score')
plt.show()

print(f"GMM Score = ", GMM_scores)
print(f"GMM Score average: {np.mean(GMM_scores)}")
```




```
GMM Score = [93, 93, 98, 93, 96, 100, 98, 98, 96, 96]
GMM Score average: 96.1
```

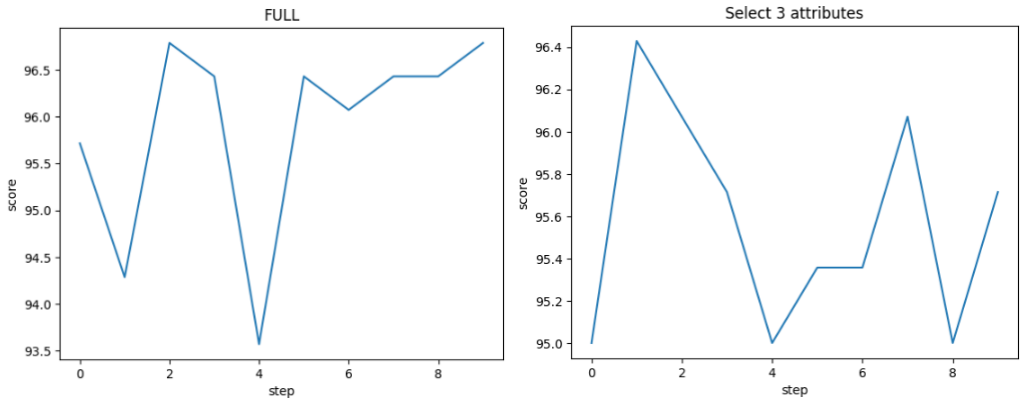
5.

```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.naive_bayes import GaussianNB
from sklearn.feature_selection import SequentialFeatureSelector
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

```
def get_mean_of_missing_values_attr(df):
    numbers = 0
    total = 0
    for value in df:
        if value != '?':
            total += int(value)
            numbers += 1
    return total / numbers

def GaussianNB_classifier(X_train, X_test, y_train, y_test):
    classifier = GaussianNB()
    classifier.fit(X_train, y_train)
    return round(accuracy_score(classifier.predict(X_test), y_test) * 100, 3)

def select_top3_attributes(X_val, y_val):
    sfs = SequentialFeatureSelector(GaussianNB(), n_features_to_select=3)
    sfs.fit(X_val, y_val)
    return sfs.get_support()
```

```
breast_cancer = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data', header=None)
breast_cancer.columns = ['Id', 'Clump Thickness', 'Uniformity of Cell Size', 'Uniformity of Cell Shape', 'Marginal Adhesion',
                         'Single Epithelial Cell Size', 'Bare Nuclei', 'Bland Chromatin', 'Normal Nucleoli', 'Mitoses', 'Class']
breast_cancer = breast_cancer.drop(['Id'], axis=1)
breast_cancer.replace(to_replace = '?', value = get_mean_of_missing_values_attr(breast_cancer.iloc[:,9]), inplace = True)
breast_cancer = breast_cancer.astype('int64')

df_X = breast_cancer.iloc[:,:9].values
df_y = breast_cancer.iloc[:,9].values

full_scores = []
select_scores = []

for i in range (10):
    X_train, X_test, y_train, y_test = train_test_split(df_X, df_y, test_size=0.4)
    X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.5)

    full_scores.append(GaussianNB_classifier(X_train, X_test, y_train, y_test))

    selected = select_top3_attributes(X_val, y_val)
    select_scores.append(GaussianNB_classifier(X_train[:, selected], X_test[:, selected], y_train, y_test))

    print(f"{i+1} time(s) select attributes are", end='')
    for k in range(len(selected)):
        if selected[k]:
            print(f"[{breast_cancer.columns[k]}], ", end='')
    print(end='\n')

plt.plot(full_scores)
plt.title('FULL')
plt.xlabel('step')
plt.ylabel('score')
plt.show()

plt.plot(select_scores)
plt.title('Select 3 attributes')
plt.xlabel('step')
plt.ylabel('score')
plt.show()

print(f"Full Score = ", full_scores)
print(f"Full Score average: {np.mean(full_scores)}")
print(f"Select 3 attributes Score = ", select_scores)
print(f"Select 3 attributes Score average: {np.mean(select_scores)}")
```

```
1 time(s) select attributes are[Uniformity of Cell Shape], [Bare Nuclei], [Mitoses],
2 time(s) select attributes are[Clump Thickness], [Uniformity of Cell Size], [Bare Nuclei],
3 time(s) select attributes are[Uniformity of Cell Size], [Uniformity of Cell Shape], [Bare Nuclei],
4 time(s) select attributes are[Clump Thickness], [Uniformity of Cell Size], [Uniformity of Cell Shape],
5 time(s) select attributes are[Clump Thickness], [Uniformity of Cell Size], [Bare Nuclei],
6 time(s) select attributes are[Uniformity of Cell Size], [Bland Chromatin], [Normal Nucleoli],
7 time(s) select attributes are[Clump Thickness], [Bare Nuclei], [Normal Nucleoli],
8 time(s) select attributes are[Uniformity of Cell Size], [Bare Nuclei], [Normal Nucleoli],
9 time(s) select attributes are[Clump Thickness], [Uniformity of Cell Size], [Normal Nucleoli],
10 time(s) select attributes are[Clump Thickness], [Uniformity of Cell Shape], [Normal Nucleoli],
```




```
Full Score =  [95.714, 94.286, 96.786, 96.429, 93.571, 96.429, 96.071, 96.429, 96.429, 96.786]
Full Score average: 95.893
Select 3 attributes Score =  [95.0, 96.429, 96.071, 95.714, 95.0, 95.357, 95.357, 96.071, 95.0, 95.714]
Select 3 attributes Score average: 95.5713
```