1. For calculating outliers in clusters, using the Mahalanobis distance method would be better if there is a belief that the dimensions are correlated and have different variances.

2.

3.

4.

```
[1]: # 4
# import dataset and sklearn module
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.mixture import GaussianMixture

iris = datasets.load_iris()
X = iris.data
y = iris.target
```

```
[2]: # split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# create gmm model for each class using sklearn
gmm_0 = GaussianMixture(n_components=2, random_state=42)
gmm_1 = GaussianMixture(n_components=2, random_state=42)
gmm_2 = GaussianMixture(n_components=2, random_state=42)

# Fit each GMM model on its corresponding class in the training data
gmm_0.fit(X_train[y_train == 0])
gmm_1.fit(X_train[y_train == 1])
gmm_2.fit(X_train[y_train == 2])
```

```
[2]: ▾ GaussianMixture
GaussianMixture(n_components=2, random_state=42)
```

```
[3]: # Predict the class of each data point in the test set
y_pred_0 = gmm_0.score_samples(X_test)
y_pred_1 = gmm_1.score_samples(X_test)
y_pred_2 = gmm_2.score_samples(X_test)

# Combine the predicted classes into a single array
y_pred = []
for i in range(len(y_pred_0)):
    if y_pred_0[i] > y_pred_1[i] and y_pred_0[i] > y_pred_2[i]:
        y_pred.append(0)
    elif y_pred_1[i] > y_pred_0[i] and y_pred_1[i] > y_pred_2[i]:
        y_pred.append(1)
    else:
        y_pred.append(2)
```

```
[4]: from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       1.00      0.83      0.91        18
           2       0.80      1.00      0.89        12

    accuracy                           0.93        45
   macro avg       0.93      0.94      0.93        45
weighted avg       0.95      0.93      0.93        45
```

5.

```python
# Problem 5
from sklearn.datasets import load_breast_cancer
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import RFE
from sklearn.metrics import accuracy_score
import numpy as np

data = load_breast_cancer()
X = data.data[:, [0, 2, 3]]  # Keep only the first three features
y = data.target

X
```

```
array([[  17.99,  122.8 , 1001.  ],
       [  20.57,  132.9 , 1326.  ],
       [  19.69,  130.  , 1203.  ],
       ...,
       [  16.6 ,  108.3 ,  858.1 ],
       [  20.6 ,  140.1 , 1265.  ],
       [   7.76,   47.92,  181.  ]])
```

```python
# data spliting
X_train_fs, X_test_fs, y_train_fs, y_test_fs = train_test_split(X, y, test_size=0.2, stratify=y)
X_train, X_fs, y_train, y_fs = train_test_split(X_train_fs, y_train_fs, test_size=0.33, stratify=y_train_fs)
```

```python
# using  Recursive Feature Elimination (RFE) for feature selection
nb = GaussianNB()
rfe = RFE(nb, n_features_to_select=3)
rfe.fit(X_fs, y_fs)
```

```
▸ RFE
▸ estimator: GaussianNB
    ▸ GaussianNB
```

```python
# use the selected features for training and testing
X_train_sel = X_train[:, rfe.support_]
X_test_sel = X_test_fs[:, rfe.support_]

nb_sel = GaussianNB()
nb_sel.fit(X_train_sel, y_train)

y_pred_sel = nb_sel.predict(X_test_sel)
accuracy_sel = accuracy_score(y_test_fs, y_pred_sel)
```

```python
n_runs = 10
accuracies = np.zeros(n_runs)

for i in range(n_runs):
    X_train_fs, X_test_fs, y_train_fs, y_test_fs = train_test_split(X, y, test_size=0.2, stratify=y, random_state=i)
    X_train, X_fs, y_train, y_fs = train_test_split(X_train_fs, y_train_fs, test_size=0.33, stratify=y_train_fs, random_state=i)

    nb = GaussianNB()
    rfe = RFE(nb, n_features_to_select=3)
    rfe.fit(X_fs, y_fs)

    X_train_sel = X_train[:, rfe.support_]
    X_test_sel = X_test_fs[:, rfe.support_]

    nb_sel = GaussianNB()
    nb_sel.fit(X_train_sel, y_train)

    y_pred_sel = nb_sel.predict(X_test_sel)
    accuracy_sel = accuracy_score(y_test_fs, y_pred_sel)

    accuracies[i] = accuracy_sel

mean_accuracy_sel = np.mean(accuracies)

mean_accuracy_sel
```

```
0.8921052631578948
```

```python
X_train_full, X_test_full, y_train_full, y_test_full = train_test_split(data.data, data.target, test_size=0.2, stratify=data.target, random_state=42)

nb_full = GaussianNB()
nb_full.fit(X_train_full, y_train_full)

y_pred_full = nb_full.predict(X_test_full)
accuracy_full = accuracy_score(y_test_full, y_pred_full)
```

```python
print("Average accuracy with 3 selected features: {:.3f}".format(mean_accuracy_sel))
print("Accuracy with full set of 9 features: {:.3f}".format(accuracy_full))
```

```
Average accuracy with 3 selected features: 0.892
Accuracy with full set of 9 features: 0.939
```