Assignment 2 (Bayesian Decision)

Ahmed Shehata Mahmoud **Sarah Hossam Elmowafy**

Part 1

For class 0:

Mean for Sepal Length = (5.1-+5.0+4.8+5.0)/4 = 4.975

Mean for Sepal Width = (3.4+3.4+3.0+3.3)/4 = 3.275

Mean for Petal length = (1.5+1.5+1.4+1.4)/4 = 1.45

Mean for Petal Width = (0.2+0.2+0.2+0.1)/4 = .175

Variance for Sepal Length = $((5.1-4.975)^2+(5.0-4.975)^2+(4.8-4.975)^2+(5.0-4.975)^2)/4-1 = 0.01583$

Variance for Sepal Width = $((3.4-3.275)^2+(3.4-3.275)^2+(3.0-3.275)^2+(3.3-3.275)^2)/4-1 = 0.03583$

Variance for Petal length = $((1.5-1.45)^2+(1.5-1.45)^2+(1.4-1.45)^2+(1.4-1.45)^2)/4-1 = 0.003$

Variance for Petal Width = $((0.2-.175)^2+(0.2-.175)^2+(0.2-.175)^2+(0.1-.175)^2)/4 = 0.0025$

For Class 1:

Mean for Sepal Length = (4.9+5.7+5.4+5.6)/4 = 5.4

Mean for Sepal Width = (2.4+3.0+3.0+2.5)/4 = 2.725

Mean for Petal length = (3.3+4.2+4.5+3.9)/4 = 4

Mean for Petal Width = (1.0+1.2+1.5+1.1)/4 = 1.2

Variance for Sepal Length = $((4.9-5.4)^2+(5.7-5.4)^2+(5.4-5.4)^2+(5.6-5.4)^2)/4-1 = 0.126$

Variance for Sepal Width = $((2.4-2.725)^2+(3.0-2.725)^2+(3.0-2.725)^2+(2.5-2.725)^2)/4-1 = 0.1025$

Variance for Petal length = $((3.3-4)^2+(4.2-4)^2+(4.5-4)^2+(3.9-4)^2)/4-1 = 0.263$

Variance for Petal Width = $((1.0-1.2)^2+(1.2-1.2)^2+(1.5-1.2)^2+(1.1-1.2)^2)/4-1 = 0.046$

For class 2:

Mean for Sepal Length = (6.5+7.7)/2 = 7.1

Mean for Sepal Width = (3.0+2.6)/2 = 2.8

Mean for Petal length = (5.8+6.9)/2 = 6.35

Mean for Petal Width = (2.2+2.3)/2 = 2.25

Variance for Sepal Length = $((6.5-7.1)^2+(7.7-7.1)^2)/2-1 = 0.72$

Variance for Sepal Width = $((3.0-2.8)^2+(2.6-2.8)^2)/2-1 = 0.08$

Variance for Petal length = $((5.8-6.35)^2+(6.9-6.35)^2)/2-1 = 0.605$

Variance for Petal Width = $((2.2-2.25)^2+(2.3-2.25)^2)/2-1 = 0.005$

This tables shows all the results in a simple way

Class	Mean (Sepal length)	Variance (sepal length)	Mean (sepal width)	Variance (sepal width)	mean(petal length)	Variance (petal length)	Mean (petal width)	variance (petal width)
0	4.975	0.01583	3.275	0.03583	1.45	0.003	.175	0.0025
1	5.4	0.125	2.725	0.1025	4	0.263	1.2	0.046
2	7.1	0.72	2.8	0.08	6.35	0.605	2.25	0.005

Calculating the likelihoods:

$$P(v \mid class) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(v-\mu)^2}{2\sigma^2}\right)$$

For the first point

$$P(\text{sepal length} \mid 0) = \frac{1}{\sqrt{2\pi}*0.01583} \exp\left(\frac{-(5.7-4.975)^2}{2*0.01583}\right) = 1.95*10\text{^-}7$$

$$P(\text{sepal length} \mid 1) = \frac{1}{\sqrt{2\pi}*0.125} \exp\left(\frac{-(5.7-5.4)^2}{2*0.125}\right) = .787$$

$$P(\text{sepal length} \mid 2) = \frac{1}{\sqrt{2\pi}*0.72} \exp\left(\frac{-(5.7-7.1)^2}{2*0.72}\right) = 0.12$$

$$P(\text{sepal width} \mid 0) = \frac{1}{\sqrt{2\pi}*.03583} \exp\left(\frac{-(2.8-3.275)^2}{2*.03583}\right) = 0.0904$$

$$P(\text{sepal width} \mid 1) = \frac{1}{\sqrt{2\pi}*.1025} \exp\left(\frac{-(2.8-2.725)^2}{2*.1025}\right) = 1.21$$

P (sepal width | 2) = $\frac{1}{\sqrt{2\pi}*.08}\exp\left(\frac{-(2.8-2.8)^2}{2*.08}\right) = 1.41$

P (petal length | 0) = $\frac{1}{\sqrt{2\pi}*.003}\exp\left(\frac{-(4.5-1.45)^2}{2*.003}\right) = 0$

P (petal length | 1) = $\frac{1}{\sqrt{2\pi}*.263}\exp\left(\frac{-(4.5-4)^2}{2*.263}\right) = .483$

P (petal length | 2) = $\frac{1}{\sqrt{2\pi}*.605}\exp\left(\frac{-(4.5-6.35)^2}{2*.605}\right) = 0.0303$

P (petal width | 0) = $\frac{1}{\sqrt{2\pi}*.0025}\exp\left(\frac{-(1.3-.175)^2}{2*.0025}\right) = 0$

P (petal width | 1) = $\frac{1}{\sqrt{2\pi}*.046}\exp\left(\frac{-(1.3-1.2)^2}{2*.046}\right) = 1.668$

P (petal width | 2) = $\frac{1}{\sqrt{2\pi}*.005}\exp\left(\frac{-(1.3-2.25)^2}{2*.005}\right) = 3.6*10^{-39}$

For the second point:

P (sepal length | 0) = $\frac{1}{\sqrt{2\pi}*0.01583}\exp\left(\frac{-(5.4-4.975)^2}{2*0.01583}\right) = 0.01055$

P (sepal length | 1) = $\frac{1}{\sqrt{2\pi}*0.125}\exp\left(\frac{-(5.4-5.4)^2}{2*0.125}\right) = 1.12$

P (sepal length | 2) = $\frac{1}{\sqrt{2\pi}*0.72}\exp\left(\frac{-(5.4-7.1)^2}{2*0.72}\right) = 0.063$

P (sepal width | 0) = $\frac{1}{\sqrt{2\pi}*.03583}\exp\left(\frac{-(3.9-3.275)^2}{2*.03583}\right) = 9.04*10^{-3}$

P (sepal width | 1) = $\frac{1}{\sqrt{2\pi}*.1025}\exp\left(\frac{-(3.9-2.725)^2}{2*.1025}\right) = 1.48*10^{-3}$

P (sepal width | 2) = $\frac{1}{\sqrt{2\pi}*.08}\exp\left(\frac{-(3.9-2.8)^2}{2*.08}\right) = 7.32*10^{-4}$

P (petal length | 0) = $\frac{1}{\sqrt{2\pi}*.003}\exp\left(\frac{-(1.3-1.45)^2}{2*.003}\right)$ = 5.0059

P (petal length | 1) = $\frac{1}{\sqrt{2\pi}*.263}\exp\left(\frac{-(1.3-4)^2}{2*.263}\right)$ = 7.4*10^-7

P (petal length | 2) = $\frac{1}{\sqrt{2\pi}*.605}\exp\left(\frac{-(1.3-6.35)^2}{2*.605}\right)$ =3.6*10^-10

P (petal width | 0) = $\frac{1}{\sqrt{2\pi}*.0025}\exp\left(\frac{-(.4-.175)^2}{2*.0025}\right)$ = 2.8987

P (petal width | 1) = $\frac{1}{\sqrt{2\pi}*.046}\exp\left(\frac{-(.4-1.2)^2}{2*.046}\right)$ = 1.77*10^-3

P (petal width | 2) = $\frac{1}{\sqrt{2\pi}*.005}\exp\left(\frac{-(.4-2.25)^2}{2*.005}\right)$ = 0

Priors:

- P (0) = .4
- P (1) = .4
- P (2) = .2

Evidence = P(0) P(sepal length | 0) P(sepal width | 0) (petal length | 0) P(petal width | 0) +

P(1) P(sepal length | 1) P(sepal width | 1) (petal length | 1) P(petal width | 1) +

P(2) P(sepal length | 2) P(sepal width | 2) (petal length | 2) P(petal width | 2)

For the first point:

Evidence = (.4 * 0.0904 * 1.95*10^-7 * 0 *0) + (.4 * 1.21 * .787 * 1.668 * .483) + (.2 * 1.41 * 0.12 *3.6*10^-39 * 0.0303) = 0.3068

Posterior (0) = (P(c) P(sepal length | c) P(sepal width | c) (petal length | c) P(petal width | c))/evidence

Posterior (0) = (.4 * 0.0904 * 1.95*10^-7 * 0 *0) / 0.3068 = 0

Posterior (1) = (.4 * 1.21 * .787 * 1.668 * .483) /0.3068 = (.3068) /0.3068 = 1.00

Posterior (2) = (.2 * 1.41 * 0.12 *3.6*10^-39 * 0.0303) / 0.3068 = 1.218*10^38 / 0.3068 =1.2*10^-41

- Class (1) has the higher probability for the first testing record

For the second point:

Evidence = (.4*0.01055*9.04*10^-3*5.0059*2.8987) + (.4*1.12*1.48*10^-3*7.4*10^-7*1.77*10^-3) + (.2*0.063*7.32*10^-4*3.6*10^-10*0) = 5.535*10^-4

Posterior (0) = (9.04*10^-3*0.01055*2.8987*5.0059)/ 5.535*10^-4= 2.493

Posterior (1) = (1.48*10^-3) *(1.12)*(1.77*10^-3)*(7.4*10^-7)) /5.535*10^-4= (2.17*10^-12) /e =3.9*10^-17

Posterior (2) = (7.32*10^-4*0.063*0*3.6*10^-10) / 5.535*10^-4= 0

- Class (0) has the higher probability for the second testing record

These Next calculations are made just in case of needing the mean and variance of each feature regardless of the class, but if the goal here is to calculate them for each class separately to use them in the third question so that already has been calculated after.

Calculate mean of each feature in Table 1:

Mean for Sepal Length = (6.5+4.9+5.7+7.7+5.4+5.1+5.0+5.6+4.8+5.0)/10 = 5.57

Mean for Sepal Width = (3.0+2.4+3.0+2.6+3.0+3.4+3.4+2.5+3.0+3.3)/10 = 2.96

Mean for Petal length = (5.8+3.3+4.2+6.9+4.5+1.5+1.5+3.9+1.4+1.4)/10 = 3.44

Mean for Petal Width = (2.2+1.0+1.2+2.3+1.5+0.2+0.2+1.1+0.2+0.1)/10 = 1

Calculate variance of each feature in Table 1:

Variance for Sepal Length = ((6.5-5.57) ^{2}+ (4.9-5.57) ^{2}+ (5.7-5.57) ^{2}+ (7.7-5.57) ^{2}+ (5.4-5.57) ^{2} + (5.1-5.57) ^{2}+ (5.0-5.57) ^{2}+ (5.6-5.57) ^{2}+ (4.8-5.57) ^{2}+ (5.0-5.57) ^{2}) / 10 = 0.7361

Variance for Sepal Width = ((3.0-2.96) ^{2}+ (2.4-2.96) ^{2}+ (3.0-2.96) ^{2}+ (2.6-2.96) ^{2}+ (3.0-2.96) ^{2} + (3.4-2.96) ^{2}+ (3.4-2.96) ^{2}+ (2.5-2.96) ^{2}+ (3.0-2.96) ^{2}+ (3.3-2.96) ^{2}) / 10 = 0.1164

Variance for Petal Length = ((5.8-3.44) ^{2} + (3.3-3.44) ^{2} + (4.2-3.44) ^{2}+ (6.9-3.44) ^{2} + (4.5-3.44) ^{2} + (1.5-3.44) ^{2} + (1.5-3.44) ^{2} + (3.9-3.44) ^{2} + (1.4-3.44) ^{2} + (1.4-3.44) ^{2}) / 10 = 3.5

Part 2

Introduction:

The goal is to classify Iris data set using Naïve Bayes classifier and trying to increase the model's accuracy by tuning hyperparameters and seeing if using this model with risk-based matrix will affect the overall accuracy of them model.

Data Set:

Loading the dataset:

Iris data set, numerical data with four features and one target (150 columns).

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0

Dropping the petal length and petal width features

Here is the data after dropping the unwanted features.

	sepal length (cm)	sepal width (cm)	target
0	5.1	3.5	0.0
1	4.9	3.0	0.0
2	4.7	3.2	0.0
3	4.6	3.1	0.0
4	5.0	3.6	0.0

Building classifier function

Build_classifier function takes the estimator that will be used, train and test data.

First, the model is fit with the training data after that it predicts the output for the test as well as the train data. Predicting on the train data that has been already used in training phase before maybe is used just to show that the model will even misclassify that data that it has learn on because it couldn't understand the data as we will see later there're two overlapping classes that it is hard for the model to classify them. Also, it prints some information as we will see.

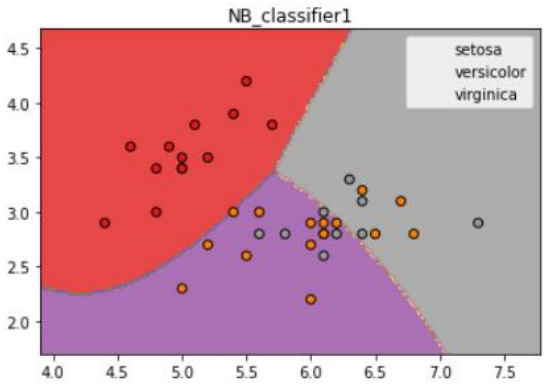
```python
def build_classifier(estimator_obj, X_train, X_test, y_train, y_test, title=''):

    model = estimator_obj.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    y_pred_for_trainData = model.predict(X_train)

    print('\nClassification Report:\n')
    print(classification_report(y_test, y_pred))
    print("-----------------------------------------------\n")

    train_acc = accuracy_score(y_train, y_pred_for_trainData)
    print("Train Accuracy: {:.2f}\n".format(train_acc))

    train_precision = precision_score(y_train, y_pred_for_trainData, average='macro')
    print("precision score for train: {:.2f}\n".format(train_precision))

    train_recall = recall_score(y_train, y_pred_for_trainData, average='macro')
    print("recall score for train: {:.2f}\n".format(train_recall))
    print("-----------------------------------------------\n")

    test_acc = accuracy_score(y_test, y_pred)
    print("Test Accuracy: {:.2f}\n".format(test_acc))

    test_precision = precision_score(y_test, y_pred, average='macro')
    print("precision score for test: {:.2f}\n".format(test_precision))

    test_recall = recall_score(y_test, y_pred, average='macro')
    print("recall score for test: {:.2f}\n".format(test_recall))
    print("-----------------------------------------------")


    print('\nDecision Boundary for test:\n')
    plotDecisionBoundary(X_test, y_test, model, title)

    print('\nConfusion Matrix:\n')
    plot_confusion_matrix(model, X_test, y_test, xticks_rotation='horizontal', cmap="OrRd_r")

    return model, y_pred, train_acc, test_acc
```

Apply Naïve Bayes Classifier to get training and testing accuracy

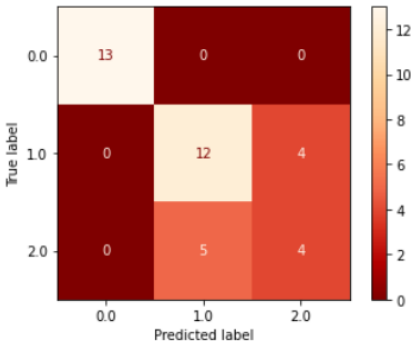
The following results on the train and test accuracy shows that the model is not good enough.

```
Classification Report:

              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00        13
         1.0       0.71      0.75      0.73        16
         2.0       0.50      0.44      0.47         9

    accuracy                           0.76        38
   macro avg       0.74      0.73      0.73        38
weighted avg       0.76      0.76      0.76        38

Train Accuracy: 0.81

precision score for train: 0.81

recall score for train: 0.81
```

```
Test Accuracy: 0.76

precision score for test: 0.74

recall score for test: 0.73
```

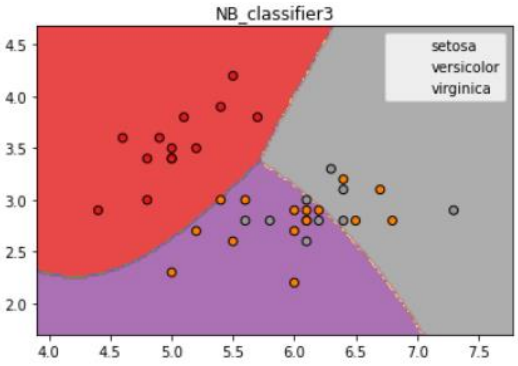


Confusion Matrix:

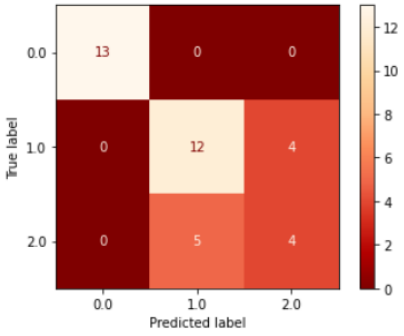


Tuning hyperparameters of Naive Bayes Classifier (i.e., var_smoothing).

As we see the model accuracy is not satisfying so we will try to tune the hyperparameters that may help us improve the model's accuracy. We tried three different values for smoothing, all of them yield the same next output that is the same as the model's result without tuning.



Confusion Matrix:

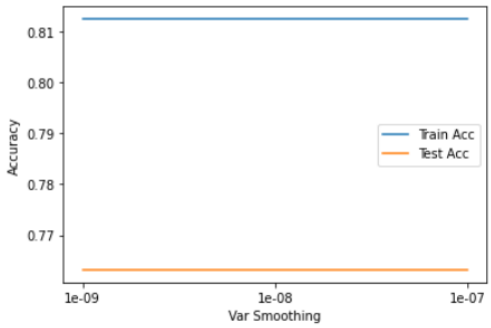



```
Classification Report:

              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00        13
         1.0       0.71      0.75      0.73        16
         2.0       0.50      0.44      0.47         9

    accuracy                           0.76        38
   macro avg       0.74      0.73      0.73        38
weighted avg       0.76      0.76      0.76        38
```

```
Train Accuracy: 0.81

precision score for train: 0.81

recall score for train: 0.81

Test Accuracy: 0.76

precision score for test: 0.74

recall score for test: 0.73
```

Smoothing

The plot below shows that the accuracy in the training and testing set is stable through the different values of var smoothing.



Risk-based Bayesian Decision Theory Classifier (RBDTC)

```python
# 5.1
class BayesianDecisionTheoryClassifier(BaseEstimator, ClassifierMixin):

    # 5.3
    def __init__(self, estimator, utilityMat):
        self.estimator = estimator
        self.utilityMat = utilityMat

    # 5.4
    def fit(self, X, y):
        # 5.2
        # check inputs
        X_checked, y_checked = check_X_y(X, y)
        self.classes_names = np.unique(labels) #names
        self.classes_ = np.unique(y_checked)   #numbers
        self.estimator_ = clone(self.estimator).fit(X_checked, y_checked)
        return self

    # 5.5
    def predict_proba(self, X):
        # check model status
        check_is_fitted(self)
        prob = self.estimator_.predict_proba(X)
        probList = [(prob * self.utilityMat[index]).sum(axis=1).reshape((-1, 1))
                    for index, c in enumerate(self.classes_)]
        prob = np.hstack(probList)
        return prob

    # 5.6
    def predict_labels(self, X):
       pred = self.predict_proba(X).argmin(axis=1)
       pred_name = self.classes_names[pred]
       return pred_name

    def predict(self, X):
        pred = self.predict_proba(X).argmin(axis=1)
        pred_label = self.classes_[pred]
        return pred_label
```

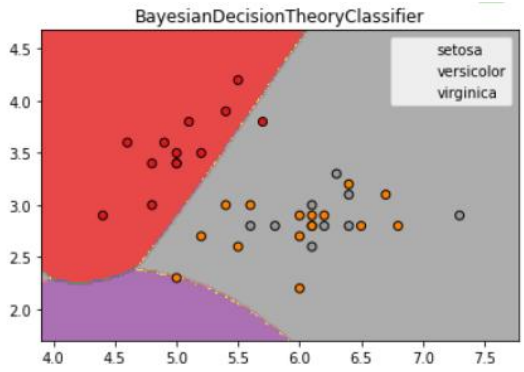
Here the classifier inherits BaseEstimator and ClassifierMinxin, the __init__ function takes risk matrix and estimator as inputs. Fit function takes training dataset as input after that the classifier checks inputs also, we define classes names and labels. by cloning the base estimator is trained while the original bases estimator is as it is. Predict proba checks the model status then, it takes the probabilities and calculate them with risk-based utility matrix and return the final probability. the other two functions return the name and the label for each class.

Risk-based utility matrix

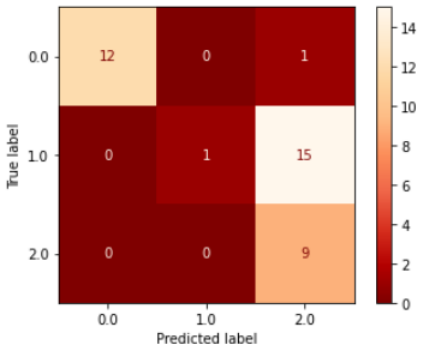
This matrix is composed of three columns and rows, every time the list of three probabilities is multiplied by one of them and then calculated to get the final probability.

```
# 6
utilityMat = np.array([
      [-10, -5, -5],
      [-5, -10, -5],
      [-5, -5, -100],])

bdtc = BayesianDecisionTheoryClassifier(NB1, utilityMat)
```

For virginica the risk is -100 which means that we need the estimator to take each point that might belong to this class like if the risk to misclassify its point to another class is high even if this affects the other class's accuracy, For Sesota, it will be affected by classifying one of its points to virginica because of the risk value there and for Vercicolor it will be extremely affected by misclassifying almost all points for Verginica. Using risk-based utility matrix here decreased the training accuracy for the model although it increased the recall of the virginica to be 100%, also decreased the recall of Versicolor badly.



Confusion Matrix:



Classification Report:

	precision	recall	f1-score	support
0.0	1.00	0.92	0.96	13
1.0	1.00	0.06	0.12	16
2.0	0.36	1.00	0.53	9
accuracy			0.58	38
macro avg	0.79	0.66	0.54	38
weighted avg	0.85	0.58	0.50	38

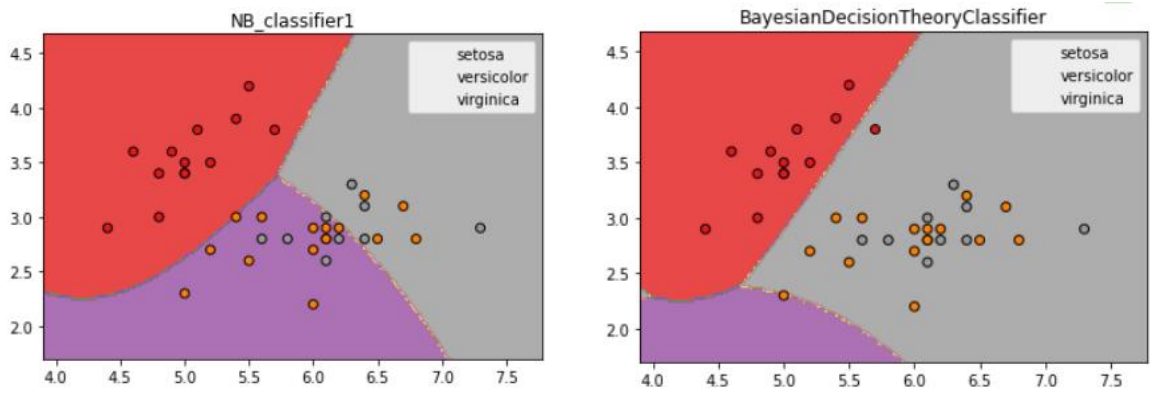
```
Train Accuracy: 0.66

precision score for train: 0.68

recall score for train: 0.63
```

```
Test Accuracy: 0.58

precision score for test: 0.79

recall score for test: 0.66
```

NB and RBDTC

Naïve Bayes classifier test or even train accuracy are not good enough to be used, although the accuracy for one of its classes Sesota is 100% but it still suffers with the other two classes as they are very similar so we can use it if we care much about classifying Sesota correctly from the other two classes.

The left plot shows 100% precision and recall for Sesota, around 75% for Versicolor but lower than 50% with Verginica than we might care a lot about so if we care to classify any other class correctly, we can use the risk-based utility matrix just to get sure that the recall of this class will be high.



As we see in the plot on the right increasing the recall for virginica badly affects the recall of Versicolor also decreases the precision of Verginica itself and all of this badly affects the model's accuracy to 66, 58 for train and test data which is pretty lower than the previous accuracy 81 and 76 without risk matrix.

Conclusion

As we saw above Naïve Bayes classifier is not the optimal model to be used if we want to classify these three classes, it's not suitable with such overlapped feature's characteristics even with tuning its parameters, it doesn't give enough accuracy, so it's not recommended to b used with such problems, but it might be used with other problems where its assumptions of independence of features holds true, actually it can perform better than other models also, it requires much less training data which will be helpful in many cases.

Using risk-based utility matrix with naïve bayes classifier would help us to increase the recall of some specific class that holds high risk if its points was falsely classified as negative as we saw with the example above, caring about Virginica leaded us to increase its recall without taking into considerations that Versicolor recall or score might be affected. Such case would be used in medical situations as we care for the malignant tumors recall being high even if this will affect the recall of benign tumor, but the risk will not be huge in this case by using this with the suitable model.