Bayesian Decision

Hagar Hesham Kamel Ismail Negm
Ekhlas Soliman Khlil Mosa

1 PART 1

1.1 QUESTION 1

Mean Formula: $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, where **n = number of samples in each class** and x_i **= feature x of sample** *i*.

Class 0:

Sepal Length Mean: $\bar{x}_{0SL} = \frac{5.1+5.0+4.8+5.0}{4} = \mathbf{4.975}$

Sepal Width Mean: $\bar{x}_{0SW} = \frac{3.4+3.4+3.0+3.3}{4} = \mathbf{3.275}$

Petal Length Mean: $\bar{x}_{0PL} = \frac{1.5+1.5+1.4+1.4}{4} = \mathbf{1.45}$

Petal Width Mean: $\bar{x}_{0PW} = \frac{0.2+0.2+0.1+0.2}{4} = \mathbf{0.175}$

Class 1:

Sepal Length Mean: $\bar{x}_{1SL} = \frac{4.9+5.7+5.4+5.6}{4} = \mathbf{5.4}$

Sepal Width Mean: $\bar{x}_{1SW} = \frac{2.4+3.0+3.0+2.5}{4} = \mathbf{2.725}$

Petal Length Mean: $\bar{x}_{1PL} = \frac{3.3+4.2+4.5+3.9}{4} = \mathbf{3.975}$

Petal Width Mean: $\bar{x}_{1PW} = \frac{1.0+1.2+1.5+1.1}{4} = \mathbf{1.2}$

Class 2:

Sepal Length Mean: $\bar{x}_{2SL} = \frac{6.5+7.7}{2} = \mathbf{7.1}$

Sepal Width Mean: $\bar{x}_{2SW} = \frac{3.0+2.6}{2} = \mathbf{2.8}$

Petal Length Mean: $\bar{x}_{2PL} = \frac{5.8+6.9}{2} = \mathbf{6.35}$

Petal Width Mean: $\bar{x}_{2PW} = \frac{2.2+2.3}{2} = \mathbf{2.25}$

1.2 QUESTION 2

Variance Formula: $\boldsymbol{\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})}$, where **n = number of samples in each class**, $\boldsymbol{x_i}$ **= feature x of sample** $\boldsymbol{i}$ and $\boldsymbol{\overline{x}}$ **= mean of feature x.**

Class 0:

Sepal Length Variance: $\boldsymbol{\sigma^2}_{0SL} = \frac{(5.1-4.975)^2+(5.0-4.975)^2+(4.8-4.975)^2+(5.0-4.975)^2}{4} =$ **0.011875**

Sepal Width Variance: $\boldsymbol{\sigma^2}_{0SW} = \frac{(3.4-3.275)^2+(3.4-3.275)^2+(3.0-3.275)^2+(3.3-3.275)^2}{4} =$ **0.026875**

Petal Length Variance: $\boldsymbol{\sigma^2}_{0PL} = \frac{(1.5-1.45)^2+(1.5-1.45)^2+(1.4-1.45)^2+(1.4-1.45)^2}{4} =$ **0.0025**

Petal Width Variance: $\boldsymbol{\sigma^2}_{0PW} = \frac{(0.2-0.175)^2+(0.2-0.175)^2+(0.1-0.175)^2+(0.2-0.175)^2}{4} =$ **0.001875**

Class 1:

Sepal Length Variance: $\boldsymbol{\sigma^2}_{1SL} = \frac{(4.9-5.4)^2+(5.7-5.4)^2+(5.4-5.4)^2+(5.6-5.4)^2}{4} =$ **0.095**

Sepal Width Variance: $\boldsymbol{\sigma^2}_{1SW} = \frac{(2.4-2.725)^2+(3.0-2.725)^2+(3.0-2.725)^2+(2.5-2.725)^2}{4} =$ **0.076875**

Petal Length Variance: $\boldsymbol{\sigma^2}_{1PL} = \frac{(3.3-3.975)^2+(4.2-3.975)^2+(4.5-3.975)^2+(3.9-3.975)^2}{4} =$ **0.196875**

Petal Width Variance: $\boldsymbol{\sigma^2}_{1PW} = \frac{(1.0-1.2)^2+(1.2-1.2)^2+(1.5-1.2)^2+(1.1-1.2)^2}{4} =$ **0.035**

Class 2:

Sepal Length Variance: $\boldsymbol{\sigma^2}_{2SL} = \frac{(6.5-7.1)^2+(7.7-7.1)^2}{2} =$ **0.36**

Sepal Width Variance: $\boldsymbol{\sigma^2}_{2SW} = \frac{(3.0-2.8)^2+(2.6-2.8)^2}{2} =$ **0.04**

Petal Length Variance: $\boldsymbol{\sigma^2}_{2PL} = \frac{(5.8-6.35)^2+(6.9-6.35)^2}{2}$: = **0.3025**

Petal Width Variance: $\boldsymbol{\sigma^2}_{2PW} = \frac{(2.2-2.25)^2+(2.3-2.25)^2}{2} =$ **0.0025**

1.3 QUESTION 3

Posterior Probability Formula:

$$P(C_k|x = x_i) = \frac{P(x = x_i|C_k) \times P(C_k)}{P(x=x_i)}, \; P(x = x_i|C_k) = \frac{1}{\sqrt{2\pi\sigma_{ki}^2}} e^{\frac{-(x_i - \bar{x}_{ki})^2}{2\sigma_{ki}^2}}$$

Sample 1:

$$P(C_0|SL = 5.7, SW = 2.8, PL = 4.5, PW = 1.3)$$
$$= \frac{P(SL = 5.7|C_0) \times P(SW = 2.8|C_0) \times P(PL = 4.5|C_0) \times P(PW = 1.3|C_0) \times P(C_0)}{P(SL = 5.7, SW = 2.8, PL = 4.5, PW = 1.3)}$$

$$P(SL = 5.7|C_0) \times P(SW = 2.8|C_0) \times P(PL = 4.5|C_0) \times P(PW = 1.3|C_0) \times P(C_0)$$
$$= \frac{1}{\sqrt{2\pi \times 0.011875}} e^{\frac{-(5.7-4.975)^2}{2\times 0.011875}} \times \frac{1}{\sqrt{2\pi \times 0.026875}} e^{\frac{-(2.8-3.275)^2}{2\times 0.026875}} \times \frac{1}{\sqrt{2\pi \times 0.0025}} e^{\frac{-(4.5-1.45)^2}{2\times 0.0025}} \times \frac{1}{\sqrt{2\pi \times 0.001875}} e^{\frac{-(1.3-0.175)^2}{2\times 0.001875}} \times 0.4$$
$$= (8.95 \times 10^{-10}) \times 0.0366 \times 0 \times 0 \times 0.4 = 0$$

$$P(C_1|SL = 5.7, SW = 2.8, PL = 4.5, PW = 1.3)$$
$$= \frac{P(SL = 5.7|C_1) \times P(SW = 2.8|C_1) \times P(PL = 4.5|C_1) \times P(PW = 1.3|C_1) \times P(C_1)}{P(SL = 5.7, SW = 2.8, PL = 4.5, PW = 1.3)}$$

$$P(SL = 5.7|C_1) \times P(SW = 2.8|C_1) \times P(PL = 4.5|C_1) \times P(PW = 1.3|C_1) \times P(C_1)$$
$$= \frac{1}{\sqrt{2\pi \times 0.095}} e^{\frac{-(5.7-5.4)^2}{2\times 0.095}} \times \frac{1}{\sqrt{2\pi \times 0.076875}} e^{\frac{-(2.8-2.725)^2}{2\times 0.076875}} \times \frac{1}{\sqrt{2\pi \times 0.196875}} e^{\frac{-(4.5-3.975)^2}{2\times 0.196875}} \times \frac{1}{\sqrt{2\pi \times 0.035}} e^{\frac{-(1.3-1.2)^2}{2\times 0.035}} \times 0.4$$
$$= 0.806 \times 1.39 \times 0.446 \times 1.85 \times 0.4 = 0.370$$

$$P(C_2|SL = 5.7, SW = 2.8, PL = 4.5, PW = 1.3) = \frac{P(SL = 5.7|C_2) \times P(SW = 2.8|C_2) \times P(PL = 4.5|C_2) \times P(PW = 1.3|C_2) \times P(C_2)}{P(SL = 5.7, SW = 2.8, PL = 4.5, PW = 1.3)}$$

$$P(SL = 5.7|C_2) \times P(SW = 2.8|C_2) \times P(PL = 4.5|C_2) \times P(PW = 1.3|C_2) \times P(C_2)$$
$$= \frac{1}{\sqrt{2\pi \times 0.36}} e^{\frac{-(5.7-7.1)^2}{2\times 0.36}} \times \frac{1}{\sqrt{2\pi \times 0.04}} e^{\frac{-(2.8-2.8)^2}{2\times 0.04}} \times \frac{1}{\sqrt{2\pi \times 0.3025}} e^{\frac{-(4.5-6.35)^2}{2\times 0.3025}} \times \frac{1}{\sqrt{2\pi \times 0.0025}} e^{\frac{-(1.3-2.25)^2}{2\times 0.0025}} \times 0.2$$
$$= 0.0437 \times 1.99 \times 0.00253 \times (3.25 \times 10^{-78}) \times 0.2 = 1.43 \times 10^{-82}$$

Maximum posterior probability for sample 1 is the posterior probability of class 1**:**
$\frac{0.370}{P(SL=5.7, SW=2.8, PL=4.5, PW=1.3)} = \frac{0.370}{0+0.370+1.43\times 10^{-82}} \approx \mathbf{1}$. Sample 1 label: **1**.

Sample 2:

$$P(C_0|SL = 5.4, SW = 3.9, PL = 1.3, PW = 0.4) = \frac{P(SL = 5.4|C_0) \times P(SW = 3.9|C_0) \times P(PL = 1.3|C_0) \times P(PW = 0.4|C_0) \times P(C_0)}{P(SL = 5.4, SW = 3.9, PL = 1.3, PW = 0.4)}$$

$$P(SL = 5.4|C_0) \times P(SW = 3.9|C_0) \times P(PL = 1.3|C_0) \times P(PW = 0.4|C_0) \times P(C_0)$$
$$= \frac{1}{\sqrt{2\pi \times 0.011875}} e^{\frac{-(5.4-4.975)^2}{2\times 0.011875}} \times \frac{1}{\sqrt{2\pi \times 0.026875}} e^{\frac{-(3.9-3.275)^2}{2\times 0.026875}} \times \frac{1}{\sqrt{2\pi \times 0.0025}} e^{\frac{-(1.3-1.45)^2}{2\times 0.0025}} \times \frac{1}{\sqrt{2\pi \times 0.001875}} e^{\frac{-(0.4-0.175)^2}{2\times 0.001875}} \times 0.4$$
$$= 0.00182 \times 0.00170 \times 0.0886 \times \left(1.26 \times 10^{-5}\right) \times 0.4 = 1.38 \times 10^{-12}$$

$$P(C_1|SL = 5.4, SW = 3.9, PL = 1.3, PW = 0.4) = \frac{P(SL = 5.4|C_1) \times P(SW = 3.9|C_1) \times P(PL = 1.3|C_1) \times P(PW = 0.4|C_1) \times P(C_1)}{P(SL = 5.4, SW = 3.9, PL = 1.3, PW = 0.4)}$$

$$P(SL = 5.4|C_1) \times P(SW = 3.9|C_1) \times P(PL = 1.3|C_1) \times P(PW = 0.4|C_1) \times P(C_1)$$
$$= \frac{1}{\sqrt{2\pi \times 0.095}} e^{\frac{-(5.4-5.4)^2}{2\times 0.095}} \times \frac{1}{\sqrt{2\pi \times 0.076875}} e^{\frac{-(3.9-2.725)^2}{2\times 0.076875}} \times \frac{1}{\sqrt{2\pi \times 0.196875}} e^{\frac{-(1.3-3.975)^2}{2\times 0.196875}} \times \frac{1}{\sqrt{2\pi \times 0.035}} e^{\frac{-(0.4-1.2)^2}{2\times 0.035}} \times 0.4$$

$$= 1.29 \times (1.81 \times 10^{-4}) \times (1.15 \times 10^{-8}) \times (2.28 \times 10^{-4}) \times 0.4 = 2.45 \times 10^{-16}$$

$$P(C_2|SL = 5.4, SW = 3.9, PL = 1.3, PW = 0.4) = \frac{P(SL = 5.4|C_2) \times P(SW = 3.9|C_2) \times P(PL = 1.3|C_2) \times P(PW = 0.4|C_2) \times P(C_2)}{P(SL = 5.4, SW = 3.9, PL = 1.3, PW = 0.4)}$$

$$P(SL = 5.4|C_2) \times P(SW = 3.9|C_2) \times P(PL = 1.3|C_2) \times P(PW = 0.4|C_2) \times P(C_2)$$

$$= \frac{1}{\sqrt{2\pi \times 0.36}} e^{\frac{-(5.4-7.1)^2}{2\times 0.36}} \times \frac{1}{\sqrt{2\pi \times 0.04}} e^{\frac{-(3.9-2.8)^2}{2\times 0.04}} \times \frac{1}{\sqrt{2\pi \times 0.3025}} e^{\frac{-(1.3-6.35)^2}{2\times 0.3025}} \times \frac{1}{\sqrt{2\pi \times 0.0025}} e^{\frac{-(0.4-2.25)^2}{2\times 0.0025}} \times 0.2$$

$$= 0.0120 \times (5.38 \times 10^{-7}) \times (3.58 \times 10^{-19}) \times 0 \times 0.2 = 0$$

Maximum posterior probability for sample 2 is the posterior probability of class 0**:** $\frac{1.38\times10^{-12}}{P(SL=5.4, SW=3.9, PL=1.3, PW=0.4)} = \frac{1.38\times10^{-12}}{1.38\times10^{-12}+2.45\times10^{-16}+0} \approx \mathbf{1}$. Sample 2 label: **0**.

***<u>Denominator is ignored in the calculations because it is common between all calculations and won't affect the class prediction.</u> ***

2 PART 2

2.1 NAÏVE BAYES CLASSIFIER

```
#plot accuracy, classification_report and model_confusion_matrix
def plot_model(model, X, y, y_pred):
  accuracy = accuracy_score(y, y_pred)
  print('Accuracy : ', accuracy)
  print(classification_report(y, y_pred))
  model_confusion_matrix = confusion_matrix(y, y_pred)
  print(model_confusion_matrix)
  plot_confusion_matrix(model, X, y)
```

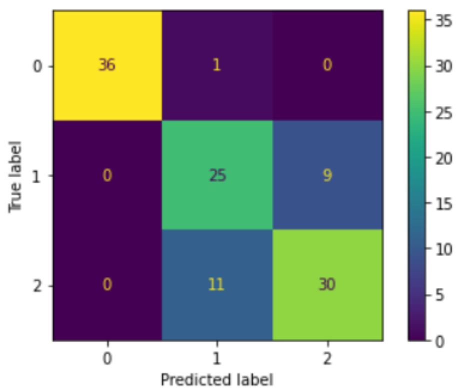
```
nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)
y_pred_train = nb_classifier.predict(X_train)
y_pred_test = nb_classifier.predict(X_test)
```

The Naïve Bayes classifier scored an accuracy of **81%** for the training set and an accuracy of **76%** for the testing set.

Training Accuracy

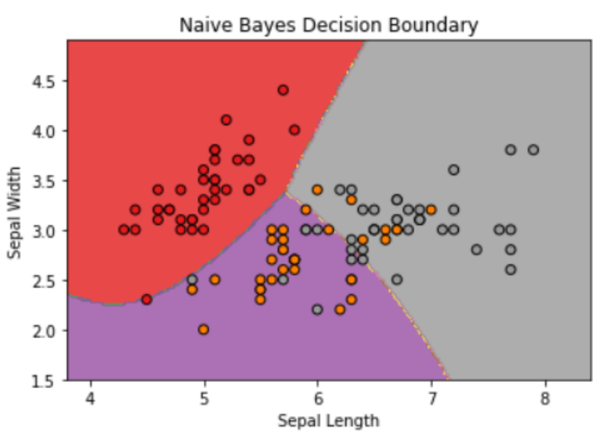
```
[ ]  1 plot_model(nb_classifier, X_train, y_train, y_pred_train)
```

```
Accuracy :  0.8125
              precision    recall  f1-score   support

           0       1.00      0.97      0.99        37
           1       0.68      0.74      0.70        34
           2       0.77      0.73      0.75        41

    accuracy                           0.81       112
   macro avg       0.81      0.81      0.81       112
weighted avg       0.82      0.81      0.81       112

[[36  1  0]
 [ 0 25  9]
 [ 0 11 30]]
```



Plot Decision Boundary

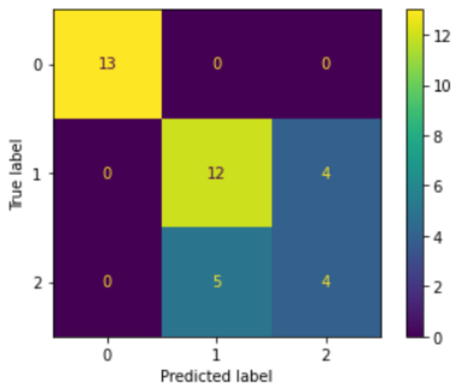
```
[108]  1 plotDecisionBoundary(X_train, y_train, nb_classifier, 'Naive Bayes Decision Boundary')
```



Testing Accuracy

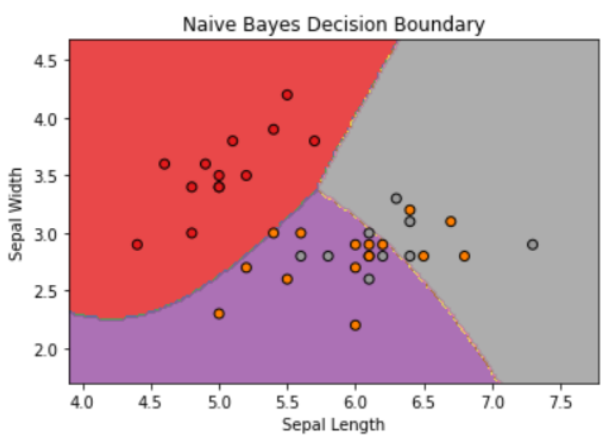
```
[ ]  1 plot_model(nb_classifier, X_test, y_test, y_pred_test)
```

```
Accuracy :  0.7631578947368421
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        13
           1       0.71      0.75      0.73        16
           2       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

[[13  0  0]
 [ 0 12  4]
 [ 0  5  4]]
```



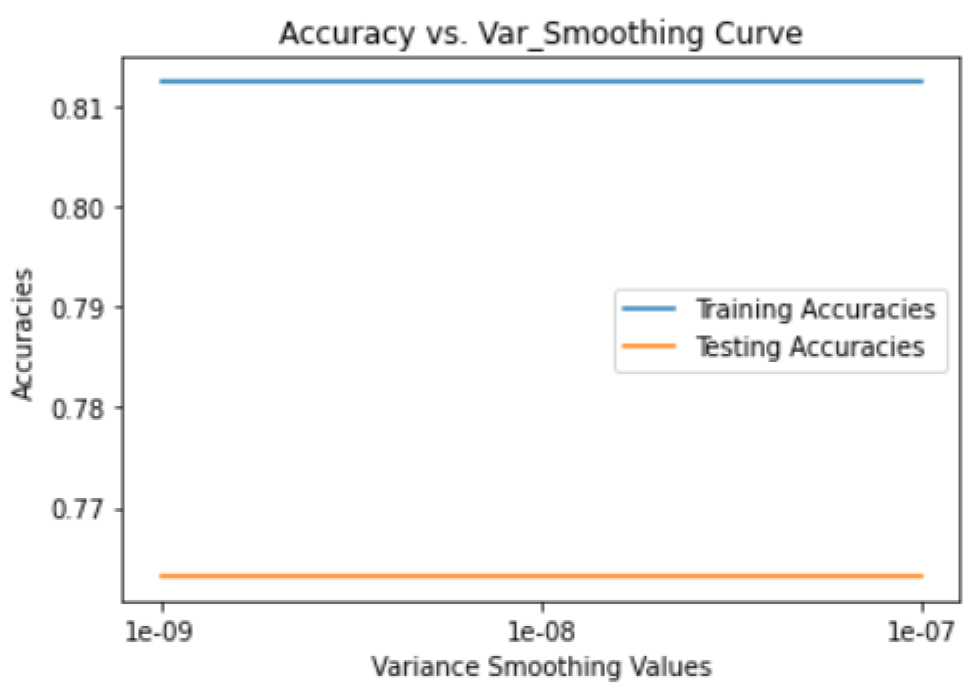
Plot Decision Boundary

```
[110]  1 plotDecisionBoundary(X_test, y_test, nb_classifier, 'Naive Bayes Decision Boundary')
```



2.2 Naïve Bayes Classifier: Accuracy vs. Var_Smoothing Curve

```
var_smoothing_vals = [1e-9, 1e-8, 1e-7]
train_accuracies = []
test_accuracies = []
for val in var_smoothing_vals:
  nb_var = GaussianNB(var_smoothing=val)
  nb_var.fit(X_train, y_train)
  y_pred_train = nb_var.predict(X_train)
  train_accuracy = accuracy_score(y_train, y_pred_train)
  train_accuracies.append(train_accuracy)
  y_pred_test = nb_var.predict(X_test)
  test_accuracy = accuracy_score(y_test, y_pred_test)
  test_accuracies.append(test_accuracy)

values = range(len(var_smoothing_vals))
plt.plot(values, train_accuracies, label = "Training Accuracies")
plt.plot(values, test_accuracies, label = "Testing Accuracies")
plt.xlabel("Variance Smoothing Values")
plt.xticks(values, var_smoothing_vals)
plt.ylabel("Accuracies")
plt.title("Accuracy vs. Var_Smoothing Curve")
plt.legend()
plt.show()
```



2.3 Risk-Based Bayesian Decision Theory Classifier

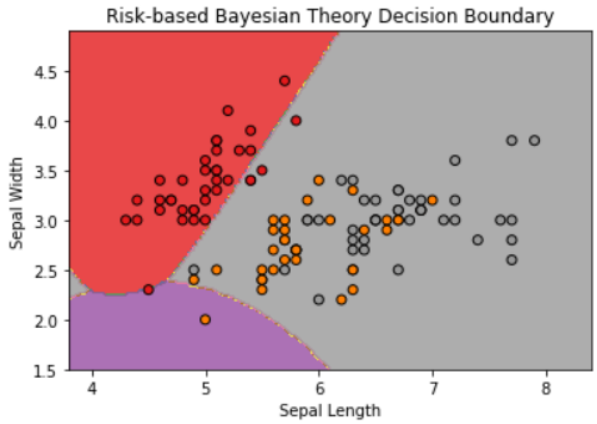
```python
class BayesianDecisionTheoryClassifier(BaseEstimator, ClassifierMixin):
    def __init__(self, estimator, riskMat, classMapping):
        self.estimator = estimator
        self.riskMat = riskMat
        self.classMapping = list(classMapping.items())

    def fit(self, X, y):
        X_checked, y_checked = check_X_y(X, y)
        self.classes_ = np.unique(y_checked)
        self.classes_names = np.array(self.classMapping)
        self.estimator_ = clone(self.estimator).fit(X_checked, y_checked)
        return self

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

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

    def predict_names(self, X):
        pred = self.predict_proba(X).argmin(axis=1)
        return self.classes_names[pred, 1]
```

```python
riskMat = np.array([
      [-10, -5, -5],
      [-5, -10, -5],
      [-5, -5, -100],
  ])
class_mapping = {
    0: "Setosa",
    1: "Versicolor",
    2: "Virginica"
}
bdtc = BayesianDecisionTheoryClassifier(nb_classifier, riskMat, class_mapping)
bdtc.fit(X_train, y_train)
bdtc_yPred_train = bdtc.predict(X_train)
bdtc_yPred_test = bdtc.predict(X_test)
bdtc_yPred_test_names = bdtc.predict_names(X_test)
print(bdtc_yPred_test)
print(bdtc_yPred_test_names)
```

```
[2 2 0 2 0 2 0 2 2 2 2 2 2 2 2 0 2 2 0 0 2 2 0 0 2 0 0 2 1 0 2 2 0 2 2 2 2
 2]
['Virginica' 'Virginica' 'Setosa' 'Virginica' 'Setosa' 'Virginica'
 'Setosa' 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Setosa' 'Virginica' 'Virginica'
 'Setosa' 'Setosa' 'Virginica' 'Virginica' 'Setosa' 'Setosa' 'Virginica'
 'Setosa' 'Setosa' 'Virginica' 'Versicolor' 'Setosa' 'Virginica'
 'Virginica' 'Setosa' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
 'Virginica']
```

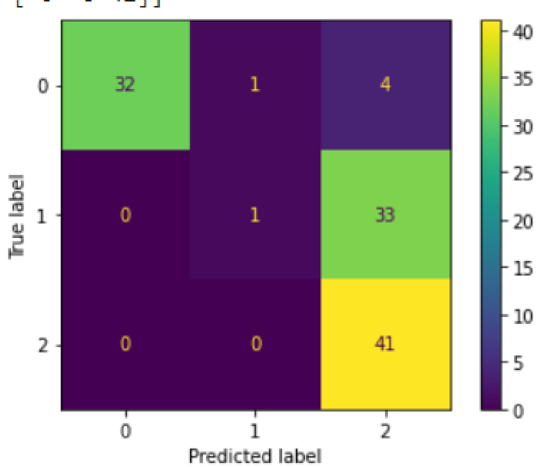
Training Set

```
1 plotDecisionBoundary(X_train, y_train, bdtc, 'Risk-based Bayesian Theory Decision Boundary')
```



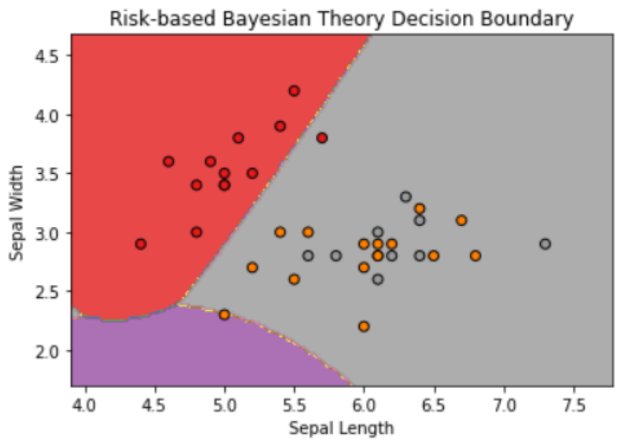

```
1 plot_model(bdtc, X_train, y_train, bdtc_yPred_train)
```

```
Accuracy :  0.6607142857142857
              precision    recall  f1-score   support

           0       1.00      0.86      0.93        37
           1       0.50      0.03      0.06        34
           2       0.53      1.00      0.69        41

    accuracy                           0.66       112
   macro avg       0.68      0.63      0.56       112
weighted avg       0.67      0.66      0.58       112

[[32  1  4]
 [ 0  1 33]
 [ 0  0 41]]
```



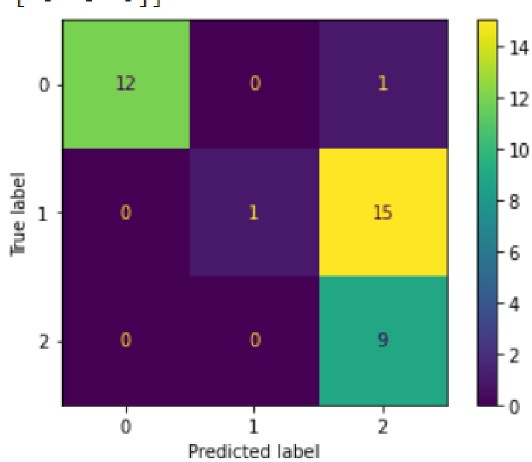
Testing Set

```
1 plotDecisionBoundary(X_test, y_test, bdtc, 'Risk-based Bayesian Theory Decision Boundary')
```




```
1 plot_model(bdtc, X_test, y_test, bdtc_yPred_test)
```

```
Accuracy :  0.5789473684210527
              precision    recall  f1-score   support

           0       1.00      0.92      0.96        13
           1       1.00      0.06      0.12        16
           2       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

[[12  0  1]
 [ 0  1 15]
 [ 0  0  9]]
```



2.4 Comparison and Analysis of Performance

The Naïve Bayes classifier was able to fully separate the Setosa class from the Versicolor and Virginica classes since it is linearly separable. The decision boundary separates all instances of the Setosa class; hence it has a precision and a recall of 100% - no instance of Setosa was mistaken for another class and no instance of other classes were mistaken for Setosa. However, the classifier was unsuccessful in separating the Versicolor class from the Virginica since they are not linearly separable, hence it achieved an accuracy of 76% on the testing set. The Versicolor class has a precision of 71% since 12/17 of the instances predicted to be of the Versicolor class were correct, and a recall of 75% since 12/16 of the class instances were predicted correctly. The Virginica class has a precision of 50% and a recall of 44%.

The Risk-based Bayesian Decision Theory classifier was not able to fully separate any single class and it achieved a testing accuracy of 58%. The classifier was able to recall 12/13 of the Setosa class instances and hence achieved a recall score of 92% for the Setosa and a precision score of 100% since all instances classified as Setosa were actually Setosa. The Versicolor class scored a precision of 100% because the only instance classified as a Versicolor was correct. However, only 1/16 of its instances was correctly recalled hence it achieved a recall of 6%. The Virginica class achieved a recall score of 100% since the decision boundary was in its favor and the classifier was able to correctly classify all its instances. However, it achieved a precision of 36% since it predicted 15 instances of the Versicolor class incorrectly as Virginica.

The overall accuracy of the RBDTC was lower mostly due to the decision boundary between the Versicolor and the Virginica classes. The decision boundary was more in favor of the Virginica class since it had a much lower risk factor than the Versicolor class, and hence the Virginica class had a higher recall score but lower precision since most of the Versicolor class were also predicted to be Virginica.

The overall accuracy of the Naïve Bayes classifier was higher since no risk matrix was used and hence there were no factors playing in favor of one class over the other.

2.5 Conclusion

In Part 1, we calculated the mean and variance of each feature per class. This was done to calculate the posterior probabilities of each class given an unseen set of features. The unseen instance is then classified to be part of the class with the maximum posterior probability. The Gaussian distribution function was used since the set of features given had continuous values. It was used to calculate the likelihood of each feature given a certain class using the previously calculated mean and variance values in addition to the newly given feature values. It was then concluded through the calculation of the posterior probabilities that the first sample is an instance of class 1 (Versicolor) and the second sample is an instance of class 0 (Setosa).

In Part 2, we trained a Gaussian Naïve Bayes classifier on our dataset and obtained a training accuracy of 81% and a testing accuracy of 76%. We then tried to tune the var_smoothing hyperparameter using values of 1e-9, 1e-8, and 1e-7. They all achieved the same accuracy as the default classifier for both training and testing sets since the values are very close to one another and very close to the default value of the var_smoothing hyperparameter. We then implemented the Risk-based Bayesian Decision Theory classifier class and trained it on our dataset and scored a training accuracy of 66% and a testing accuracy of 58%. We used the argmin function in our prediction function since we want to predict the class with lowest risk factor.