Naïve Bayesian Classifier

1 What is Naive Bayes?

The Nave Bayes, is a basic algorithm that commonly produces highly accurate and stable models from tiny sample sets. The dimensionality curse is avoided by using Naive Bayes to simplify predictive modelling jobs. Nave Bayesis' central premise is that all independent variable qualities are conditionally independent. The Gaussian Nave Bayes classifier, or simply Nave Bayes, is a straightforward algorithm that commonly produces highly accurate and stable models from tiny sample sets. The dimensionality curse is avoided by using Naive Bayes to simplify predictive modelling jobs. Nave Bayesis' central premise is that all independent variable qualities are conditionally independent.

1.1 Objective

To implement a R program showing Naïve Bayesian Classifier for a training dataset stored as a CSV file. Compute the accuracy of the classifier using the test dataset.

1.2 Algorithm

1. Create a likelihood table by finding the probabilities.
2. Calculate the posterior probability of each feature with respect to the class.
3. If for a certain feature the probability evaluates to zero use feature smoothing for correction.
4. Classify the example into the class for which the probability is highest.

1.3 Procedure

1. Import the dataset into a variable.
2. Perform exploratory analysis on the dataset
3. Split the dataset into train and test with a ratio of 80:20
4. Scale the train and test dataset to avoid variance
5. Fit the model on the training dataset and predict the values using the test dataset.
6. Calculate the accuracy of the model on the test dataset.

1.4 System Requirements

Windows OS/Mac OS/ Linux with R and relevant libraries include: e1071, caTools, caret, ggpubr, kdensity, psych.

1.5 Dataset Summary

For the project, we have utilized the Iris dataset or Fisher's Iris data set is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper. The use of multiple measurements in taxonomic problems as an example of linear discriminant analysis. It includes three iris species with 50 samples each as well as some properties about each flower. It consists of 5 columns:

1. Sepal Length
2. Sepal Width
3. Petal Length
4. Petal Width
5. Species

2 Code and Results

```
print(version)
```

```
##                _
## platform       x86_64-pc-linux-gnu
## arch           x86_64
## os             linux-gnu
## system         x86_64, linux-gnu
## status
## major          4
## minor          1.2
## year           2021
## month          11
## day            01
## svn rev        81115
## language       R
## version.string R version 4.1.2 (2021-11-01)
## nickname       Bird Hippie
```

```
#install.packages("e1071",repos = "http://cran.us.r-project.org")
#install.packages("caTools",repos = "http://cran.us.r-project.org")
#install.packages("caret",repos = "http://cran.us.r-project.org")
#install.packages("ggpubr",repos = "http://cran.us.r-project.org")
#install.packages("kdensity",repos = "http://cran.us.r-project.org")
#install.packages("psych",repos = "http://cran.us.r-project.org")
library(reshape2)
library(e1071)
library(caTools)
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(ggpubr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

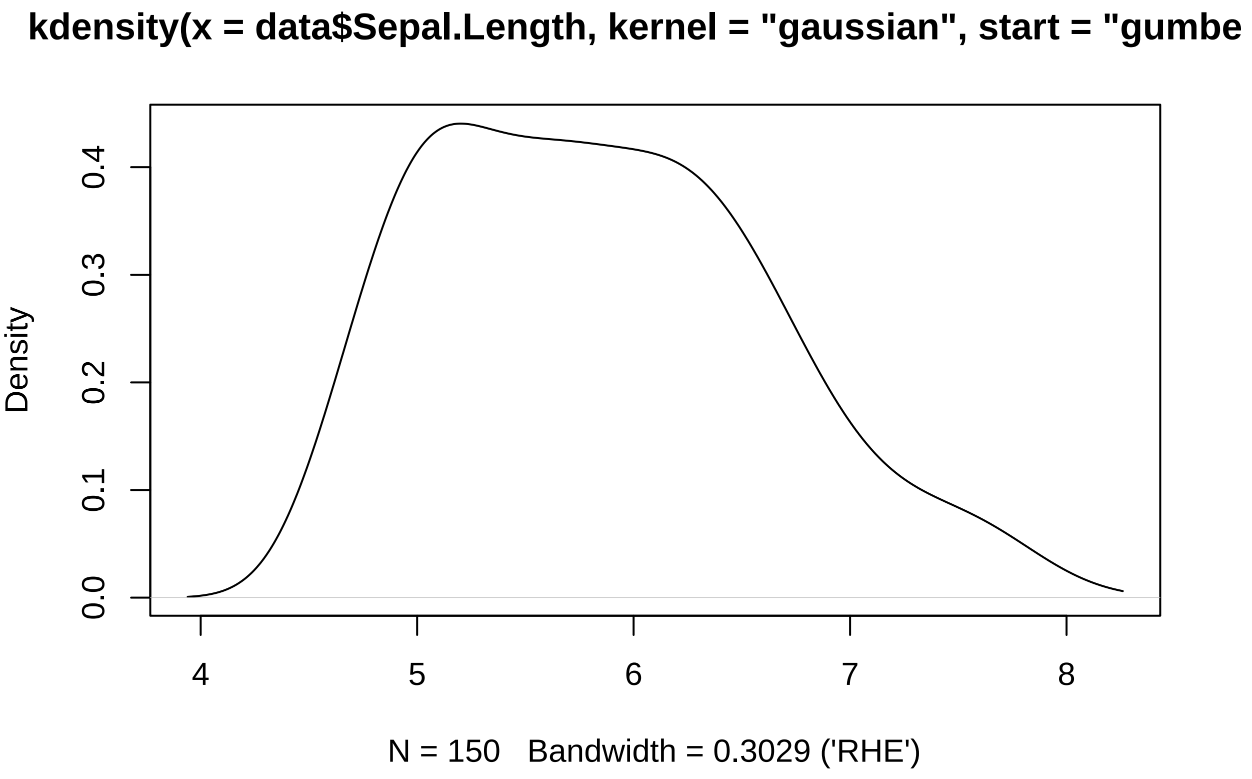
```
library("kdensity")

data=iris
print("Shape of data")
```

```
## [1] "Shape of data"
```

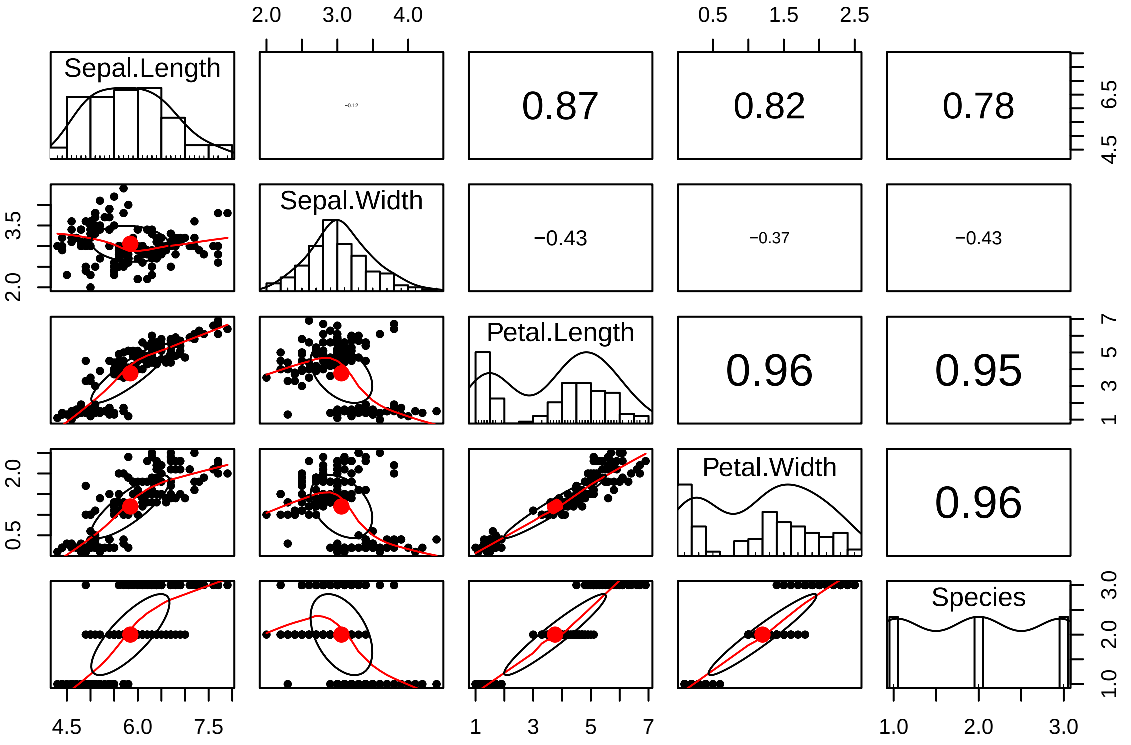
```
print(dim(data))
```

```
## [1] 150   5
```

```
kde = kdensity(data$Sepal.Length, start = "gumbel", kernel = "gaussian")
plot(kde)
```



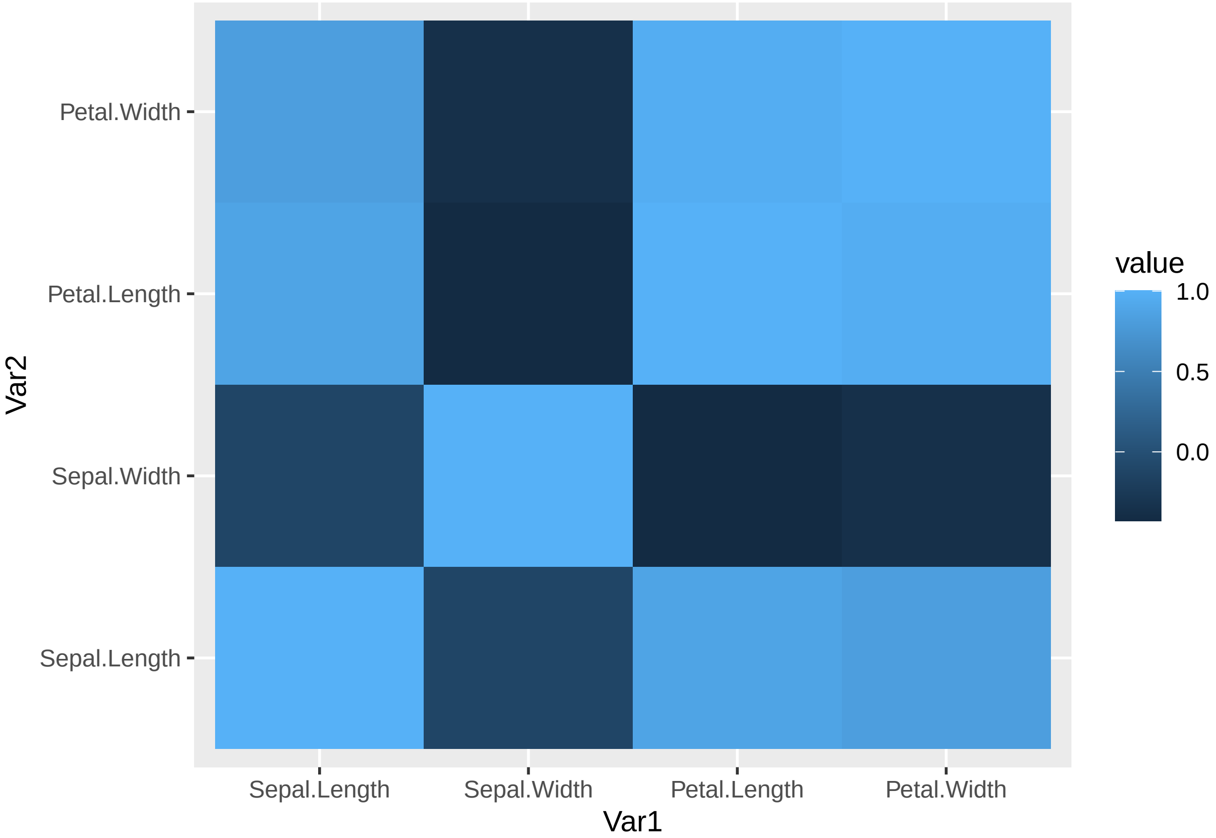
```r
pairs.panels(data, hist.col="white", scale=TRUE)
```



```r
melted_cormat <- melt(cor(data[1:4]))
head(melted_cormat)
```

```
##             Var1          Var2       value
## 1 Sepal.Length Sepal.Length  1.0000000
## 2  Sepal.Width Sepal.Length -0.1175698
## 3 Petal.Length Sepal.Length  0.8717538
## 4  Petal.Width Sepal.Length  0.8179411
## 5 Sepal.Length  Sepal.Width -0.1175698
## 6  Sepal.Width  Sepal.Width  1.0000000
```

```
ggplot(data = melted_cormat, aes(x=Var1,y=Var2, fill = value)) +
geom_tile()
```



```
split <- sample.split(data, SplitRatio = 0.8)
train <- subset(data, split == "TRUE")
test <- subset(data, split == "FALSE")
print('Shape of Train')
```

```
## [1] "Shape of Train"
```

```
print(dim(train))
```

```
## [1] 120   5
```

```
print('Shape of Test')
```

```
## [1] "Shape of Test"
```

```
print(dim(test))
```

```
## [1] 30  5
```

```
train_scale <- scale(train[,1:4])
test_scale <- scale(test[,1:4])
```

```r
set.seed(100)
nbmodel <- naiveBayes(x=train_scale,y=train$Species,usekernel=T)
print(nbmodel)
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = train_scale, y = train$Species, usekernel = T)
##
## A-priori probabilities:
## train$Species
##     setosa versicolor  virginica
##  0.3333333  0.3333333  0.3333333
##
## Conditional probabilities:
##              Sepal.Length
## train$Species       [,1]      [,2]
##    setosa     -1.0192453 0.4314743
##    versicolor  0.1457462 0.6215688
##    virginica   0.8734991 0.7803298
##
##              Sepal.Width
## train$Species       [,1]      [,2]
##    setosa      0.8246049 0.9009516
##    versicolor -0.6312493 0.7769612
##    virginica  -0.1933556 0.7010372
##
##              Petal.Length
## train$Species      [,1]      [,2]
##    setosa     -1.302426 0.0886023
##    versicolor  0.302174 0.2714300
##    virginica   1.000252 0.3318735
##
##              Petal.Width
## train$Species       [,1]      [,2]
##    setosa     -1.2554594 0.1461884
##    versicolor  0.1680536 0.2741950
##    virginica   1.0874058 0.3104803
```

```r
y_pred  <- predict(nbmodel, test_scale)
print(y_pred)
```

```
##  [1] setosa     setosa     setosa     setosa     setosa     setosa
##  [7] setosa     setosa     setosa     setosa     versicolor versicolor
## [13] versicolor versicolor versicolor versicolor versicolor versicolor
## [19] versicolor versicolor virginica  virginica  virginica  versicolor
## [25] virginica  virginica  virginica  virginica  virginica  virginica
## Levels: setosa versicolor virginica
```

```r
cm <- table(test$Species, y_pred)
print(cm)
```

```
##             y_pred
##              setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         10         0
##   virginica       0          1         9
```

```
print(confusionMatrix(cm))
```

```
## Confusion Matrix and Statistics
##
##             y_pred
##              setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         10         0
##   virginica       0          1         9
##
## Overall Statistics
##
##                Accuracy : 0.9667
##                  95% CI : (0.8278, 0.9992)
##     No Information Rate : 0.3667
##     P-Value [Acc > NIR] : 4.476e-12
##
##                   Kappa : 0.95
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.9091           1.0000
## Specificity                 1.0000            1.0000           0.9524
## Pos Pred Value              1.0000            1.0000           0.9000
## Neg Pred Value              1.0000            0.9500           1.0000
## Prevalence                  0.3333            0.3667           0.3000
## Detection Rate              0.3333            0.3333           0.3000
## Detection Prevalence        0.3333            0.3333           0.3333
## Balanced Accuracy           1.0000            0.9545           0.9762
```

3 Interpretation and Conclusion

3.1 Interpretation of output

From the output, we can interpret that:

- The naive bayes algorithm will first create probabilities for each case and using the bayes theorem predict the probability for each species and based on the highest probability will predict the species. The model gives an accuracy of 96.67
- In this case, one species was misclassified as Virginica. The model gives a sensitivity of 1 for setosa which means that it has classified all the setosa flowers correctly. The specificity of setosa is also 1 which means that it has correctly classified all the flowers that do not belong to this category.
- The model gives sensitivity and specificity of 1 and 0.9524 for versicolor and 0.9091 and 1 for virginica due to 1 misclassification

3.2 Conclusion

From the experiment conducted, it can be concluded that:

- A probabilistic machine learning model called a Naive Bayes classifier is utilized for classification tasks. The Bayes theorem lies at the heart of the classifier
- This method makes the assumption that all predictors have the same effect on the result. Naive Bayes also makes the assumption that these predictors are independent.
- Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one-dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality.