**NB Classifier & Model Accuracy**

In this tutorial we are looking at five different methods to implement a NB classifier that you can use to estimate model accuracy. They are as follows and each will be described in turn:

* Data Split
* Bootstrap
* k-fold Cross Validation
* Repeated k-fold Cross Validation
* Leave One Out Cross Validation

Generally, Repeated k-fold Cross Validation is recommended, but each method has its features and benefits, especially when the amount of data or space and time complexity are considered. Consider which approach best suits your problem.

**Data Split**

Data splitting involves partitioning the data into an explicit training dataset used to prepare the model and an unseen test dataset used to evaluate the models performance on unseen data. It is useful when you have a very large dataset so that the test dataset can provide a meaningful estimation of performance, or for when you are using slow methods and need a quick approximation of performance.

# load the libraries

library(caret)

library(klaR)

# load the iris dataset

data(iris)

# define an 80%/20% train/test split of the dataset

split=0.80

trainIndex <- createDataPartition(iris$Species, p=split, list=FALSE)

data\_train <- iris[ trainIndex,]

data\_test <- iris[-trainIndex,]

# train a naive bayes model

model <- NaiveBayes(Species~., data=data\_train)

# make predictions

x\_test <- data\_test[,1:4]

y\_test <- data\_test[,5]

predictions <- predict(model, x\_test)

# summarize results

confusionMatrix(predictions$class, y\_test)

**Bootstrap**

Bootstrap resampling involves taking random samples from the dataset (with re-selection) against which to evaluate the model. Aggregating, the results provide an indication of the variance of the models performance. Typically, large number of resampling iterations are performed (thousands or tends of thousands).The following example uses a bootstrap with 10 resamples to prepare a Naive Bayes model.

# load the library

library(caret)

# load the iris dataset

data(iris)

# define training control

train\_control <- trainControl(method="boot", number=100)

# train the model

model <- train(Species~., data=iris, trControl=train\_control, method="nb")

# summarize results

print(model)

**k-fold Cross Validation**

The k-fold cross validation method involves splitting the dataset into k-subsets. For each subset is held out while the model is trained on all other subsets. This process is completed until accuracy is determine for each instance in the dataset, and an overall accuracy estimate is provided.

It is a robust method for estimating accuracy, and the size of k and tune the amount of bias in the estimate, with popular values set to 3, 5, 7 and 10.

The following example uses 10-fold cross validation to estimate Naive Bayes on the iris dataset.

# load the library

library(caret)

# load the iris dataset

data(iris)

# define training control

train\_control <- trainControl(method="cv", number=10)

# fix the parameters of the algorithm

grid <- expand.grid(.fL=c(0), .usekernel=c(FALSE))

# train the model

model <- train(Species~., data=iris, trControl=train\_control, method="nb")

# summarize results

print(model)

**Repeated k-fold Cross Validation**

The process of splitting the data into k-folds can be repeated a number of times, this is called Repeated k-fold Cross Validation. The final model accuracy is taken as the mean from the number of repeats.

The following example uses 10-fold cross validation with 3 repeats to estimate Naive Bayes on the iris dataset.

# load the library

library(caret)

# load the iris dataset

data(iris)

# define training control

train\_control <- trainControl(method="repeatedcv", number=10, repeats=3)

# train the model

model <- train(Species~., data=iris, trControl=train\_control, method="nb")

# summarize results

print(model)

**Leave One Out Cross Validation**

In Leave One Out Cross Validation (LOOCV), a data instance is left out and a model constructed on all other data instances in the training set. This is repeated for all data instances. The following example demonstrates LOOCV to estimate Naive Bayes on the iris dataset.

# load the library

library(caret)

# load the iris dataset

data(iris)

# define training control

train\_control <- trainControl(method="LOOCV")

# train the model

model <- train(Species~., data=iris, trControl=train\_control, method="nb")

# summarize results

print(model)

**Exercise-Sample Code to Consider:** What is the purpose of the Code Below?

library(tidyverse)

library(e1071)

data(iris)

split=0.80

trainIndex <- createDataPartition(iris$Species, p=split, list=FALSE)

data\_train <- iris[ trainIndex,]

data\_test <- iris[-trainIndex,]

model <- naiveBayes(Species ~ Petal.Length + Petal.Width, data = data\_train)

print(model)

data\_train$Species <- predict(model, data\_train)

print(data\_train$Species)

data\_test$Species <- predict(model, data\_test)

print(data\_test$Species)

y\_test <- data\_test[,5]

confusionMatrix(data\_test$Species,y\_test)