Week-5

QuickFixDemos

Packages Used Installation

```
# install.packages("e1071")
# install.packages("caTools")
# install.packages("caret")
# install.packages("bnlearn")
```

```
library(bnlearn)
library(e1071)
```

```
##
## Attaching package: 'e1071'
```

```
## The following object is masked from 'package:bnlearn':
##
##     impute
```

```
library(caTools)
library(caret)
```

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

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

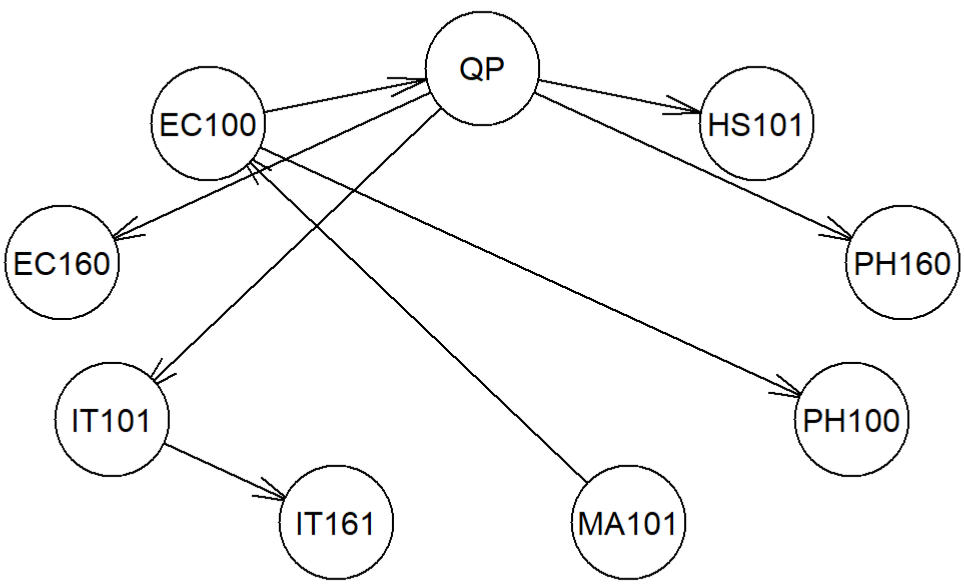
Lab Assignment 5

Learning Objective: Understand the graphical models for inference under uncertainty, build Bayesian Network in R, Learn the structure and CPTs from Data, naive Bayes classification with dependency between features. Problem Statement: A table containing grades earned by students in respective courses is made available to you in (codes folder) 2020_bn_nb_data.txt. Q1: Consider grades earned in each of the courses as random variables and learn the dependencies between courses. ##Solution

```
dataset<-read.table("C:/Users/shiva/OneDrive/Desktop/Semester-6/AILab/Lab5/2020_bn_nb_data.txt",
head=TRUE,stringsAsFactors=TRUE)
dataset_grades=dataset
dataset_net<-hc(dataset_grades,score="k2")
dataset_net
```

```
##
##   Bayesian network learned via Score-based methods
##
##   model:
##    [MA101][EC100|MA101][PH100|EC100][QP|EC100][EC160|QP][IT101|QP][PH160|QP]
##    [HS101|QP][IT161|IT101]
##   nodes:                                 9
##   arcs:                                  8
##     undirected arcs:                     0
##     directed arcs:                       8
##   average markov blanket size:           1.78
##   average neighbourhood size:            1.78
##   average branching factor:              0.89
##
##   learning algorithm:                    Hill-Climbing
##   score:                                 Cooper & Herskovits' K2
##   tests used in the learning procedure:  168
##   optimized:                             TRUE
```

```
plot(dataset_net)
```



Q2: Using the data, learn the CPTs for each course node.

```
dataset_net_bn_fit <- bn.fit(dataset_net, dataset_grades )
print(dataset_net_bn_fit)
```

```
##
##   Bayesian network parameters
##
##   Parameters of node EC100 (multinomial distribution)
##
## Conditional probability table:
##
##      MA101
## EC100         AA         AB         BB         BC         CC         CD
##    AA 0.75000000 0.07692308 0.03846154 0.01851852 0.00000000 0.00000000
##    AB 0.00000000 0.46153846 0.25000000 0.05555556 0.00000000 0.00000000
##    BB 0.25000000 0.23076923 0.32692308 0.22222222 0.04081633 0.00000000
##    BC 0.00000000 0.15384615 0.28846154 0.27777778 0.32653061 0.00000000
##    CC 0.00000000 0.07692308 0.09615385 0.24074074 0.32653061 0.04166667
##    CD 0.00000000 0.00000000 0.00000000 0.12962963 0.26530612 0.33333333
##    DD 0.00000000 0.00000000 0.00000000 0.03703704 0.04081633 0.50000000
##    F  0.00000000 0.00000000 0.00000000 0.01851852 0.00000000 0.12500000
##      MA101
## EC100         DD          F
##    AA 0.00000000 0.00000000
##    AB 0.00000000 0.00000000
##    BB 0.00000000 0.00000000
##    BC 0.00000000 0.00000000
##    CC 0.00000000 0.00000000
##    CD 0.04761905 0.00000000
##    DD 0.19047619 0.00000000
##    F  0.76190476 1.00000000
##
##   Parameters of node EC160 (multinomial distribution)
##
## Conditional probability table:
##
##      QP
## EC160          n          y
##    AA 0.00000000 0.07500000
##    AB 0.00000000 0.10000000
##    BB 0.01388889 0.18750000
##    BC 0.01388889 0.36250000
##    CC 0.15277778 0.22500000
##    CD 0.44444444 0.03125000
##    DD 0.26388889 0.01875000
##    F  0.11111111 0.00000000
##
##   Parameters of node IT101 (multinomial distribution)
##
## Conditional probability table:
##
##      QP
## IT101          n          y
##    AA 0.00000000 0.07500000
##    AB 0.00000000 0.15625000
##    BB 0.04166667 0.19375000
##    BC 0.02777778 0.29375000
```

```
##    CC 0.13888889 0.20000000
##    CD 0.30555556 0.08125000
##    DD 0.31944444 0.00000000
##    F  0.16666667 0.00000000
##
##   Parameters of node IT161 (multinomial distribution)
##
## Conditional probability table:
##
##      IT101
## IT161         AA         AB         BB         BC         CC         CD
##    AA 0.58333333 0.24000000 0.14705882 0.04081633 0.00000000 0.00000000
##    AB 0.16666667 0.40000000 0.29411765 0.02040816 0.04761905 0.00000000
##    BB 0.16666667 0.24000000 0.32352941 0.20408163 0.11904762 0.02857143
##    BC 0.08333333 0.04000000 0.20588235 0.36734694 0.38095238 0.17142857
##    CC 0.00000000 0.04000000 0.00000000 0.24489796 0.33333333 0.31428571
##    CD 0.00000000 0.04000000 0.02941176 0.10204082 0.09523810 0.31428571
##    DD 0.00000000 0.00000000 0.00000000 0.02040816 0.02380952 0.14285714
##    F  0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.02857143
##      IT101
## IT161         DD          F
##    AA 0.00000000 0.00000000
##    AB 0.00000000 0.00000000
##    BB 0.00000000 0.00000000
##    BC 0.00000000 0.00000000
##    CC 0.08695652 0.16666667
##    CD 0.52173913 0.08333333
##    DD 0.39130435 0.58333333
##    F  0.00000000 0.16666667
##
##   Parameters of node MA101 (multinomial distribution)
##
## Conditional probability table:
##          AA         AB         BB         BC         CC         CD         DD
## 0.01724138 0.05603448 0.22413793 0.23275862 0.21120690 0.10344828 0.09051724
##          F
## 0.06465517
##
##   Parameters of node PH100 (multinomial distribution)
##
## Conditional probability table:
##
##      EC100
## PH100         AA         AB         BB         BC         CC         CD
##    AA 0.71428571 0.40909091 0.22857143 0.08333333 0.00000000 0.00000000
##    AB 0.14285714 0.31818182 0.20000000 0.18750000 0.05555556 0.00000000
##    BB 0.00000000 0.18181818 0.31428571 0.29166667 0.13888889 0.03448276
##    BC 0.14285714 0.04545455 0.14285714 0.22916667 0.33333333 0.13793103
##    CC 0.00000000 0.04545455 0.11428571 0.18750000 0.25000000 0.41379310
##    CD 0.00000000 0.00000000 0.00000000 0.02083333 0.19444444 0.31034483
##    DD 0.00000000 0.00000000 0.00000000 0.00000000 0.02777778 0.10344828
##    F  0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##      EC100
## PH100         DD          F
```

```
##    AA 0.00000000 0.00000000
##    AB 0.00000000 0.00000000
##    BB 0.05000000 0.00000000
##    BC 0.00000000 0.00000000
##    CC 0.20000000 0.02857143
##    CD 0.45000000 0.11428571
##    DD 0.20000000 0.45714286
##    F  0.10000000 0.40000000
##
##   Parameters of node PH160 (multinomial distribution)
##
## Conditional probability table:
##
##      QP
## PH160          n          y
##    AA 0.05555556 0.14375000
##    AB 0.09722222 0.15625000
##    BB 0.02777778 0.17500000
##    BC 0.18055556 0.34375000
##    CC 0.29166667 0.13750000
##    CD 0.19444444 0.04375000
##    DD 0.12500000 0.00000000
##    F  0.02777778 0.00000000
##
##   Parameters of node HS101 (multinomial distribution)
##
## Conditional probability table:
##
##      QP
## HS101          n          y
##    AA 0.00000000 0.26250000
##    AB 0.00000000 0.21250000
##    BB 0.05555556 0.22500000
##    BC 0.12500000 0.16875000
##    CC 0.18055556 0.08125000
##    CD 0.19444444 0.03750000
##    DD 0.37500000 0.01250000
##    F  0.06944444 0.00000000
##
##   Parameters of node QP (multinomial distribution)
##
## Conditional probability table:
##
##    EC100
## QP         AA        AB        BB        BC        CC        CD        DD
##   n 0.0000000 0.0000000 0.0000000 0.0000000 0.1388889 0.4482759 0.9500000
##   y 1.0000000 1.0000000 1.0000000 1.0000000 0.8611111 0.5517241 0.0500000
##    EC100
## QP          F
##   n 1.0000000
##   y 0.0000000
```

Q3: What grade will a student get in PH100 if he earns DD in EC100, CC in IT101 and CD in MA101.

```
grade_list <- list("AA","AB","BB","BC","CC","CD","DD","F")
probability <- 0.0
result=""
for(grade in grade_list) {
  prob <- cpquery(dataset_net_bn_fit, event = (PH100== grade), evidence = (EC100=="DD" & IT101==
"CC" & MA101=="CD"))
  if(probability<prob){
    probability=prob;
    result=grade
  }
}
sprintf("The max probability of resultant grade is %f",probability)
```

```
## [1] "The max probability of resultant grade is 0.493827"
```

```
sprintf("The max grade obtained with given ecidence is %s ",result)
```

```
## [1] "The max grade obtained with given ecidence is CD "
```

Q4(a): The last column in the data file indicates whether a student qualifies for an internship program or not. From the given data, take 70 percent data for training and build a naive Bayes classifier (considering that the grades earned in different courses are independent of each other) which takes in the student's performance and returns the qualification status with a probability. Test your classifier on the remaining 30 percent data. Repeat this experiment for 20 random selection of training and testing data. Report results about the accuracy of your classifier.

```
dataset_grades=dataset
split <- sample.split(dataset_grades, SplitRatio = 0.7)
train <- subset(dataset_grades, split == "TRUE")
test <- subset(dataset_grades, split == "FALSE")
naive_bayes_classifier<- naiveBayes(QP ~ ., data = train)
y_train=predict(naive_bayes_classifier, newdata = train)
y_prediction <- predict(naive_bayes_classifier, newdata = test)
cm_train<- table(train$QP, y_train)
accuracy_train = (cm_train[1,1]+cm_train[2,2])/sum(cm_train)
print(round(cbind("Train Accuracy" =accuracy_train), 4))
```

```
##      Train Accuracy
## [1,]         0.9935
```

```
cm_test <- table(test$QP, y_prediction)
accuracy_test = (cm_test[1,1]+cm_test[2,2])/sum(cm_test)
print(round(cbind("Test Accuracy" =accuracy_test), 4))
```

```
##      Test Accuracy
## [1,]        0.9359
```

Q4(b): Repeat this experiment for 20 random selection of training and testing data. Report results about the accuracy of your classifier.

```
dataset_grades=dataset
dataset_grades=dataset_grades[sample(nrow(dataset_grades), 20), ]
split <- sample.split(dataset_grades, SplitRatio = 0.7)
train <- subset(dataset_grades, split == "TRUE")
test <- subset(dataset_grades, split == "FALSE")
naive_bayes_classifier<- naiveBayes(QP ~ ., data = train)
y_train=predict(naive_bayes_classifier, newdata = train)
y_prediction <- predict(naive_bayes_classifier, newdata = test)
cm_train<- table(train$QP, y_train)
accuracy_train = (cm_train[1,1]+cm_train[2,2])/sum(cm_train)
print(round(cbind("Train Accuracy" =accuracy_train), 4))
```

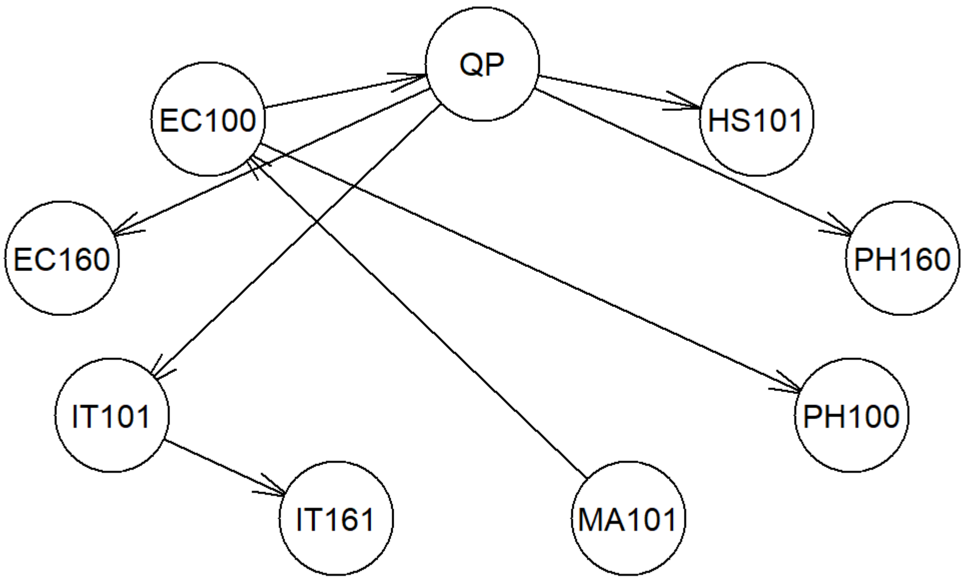
```
##      Train Accuracy
## [1,]              1
```

```
cm_test <- table(test$QP, y_prediction)
accuracy_test = (cm_test[1,1]+cm_test[2,2])/sum(cm_test)
print(round(cbind("Test Accuracy" =accuracy_test), 4))
```

```
##      Test Accuracy
## [1,]             1
```

Q5(a): Repeat 4, considering that the grades earned in different courses may be dependent.

```
dataset_grades=dataset
split <- sample.split(dataset_grades, SplitRatio = 0.7)
train <- subset(dataset_grades, split == "TRUE")
test <- subset(dataset_grades, split == "FALSE")
train.hc=suppressWarnings(hc(train, score="k2"))
plot(train.hc)
```




```
naive_bayes_classifier<- suppressWarnings(bn.fit(train.hc, train))
y_train <- predict(naive_bayes_classifier,node="QP", data = train)
y_prediction <- predict(naive_bayes_classifier,node="QP", data = test)
cm_train<- table(train$QP, y_train)
accuracy_train = (cm_train[1,1]+cm_train[2,2])/sum(cm_train)
print(round(cbind("Train Accuracy" =accuracy_train), 4))
```

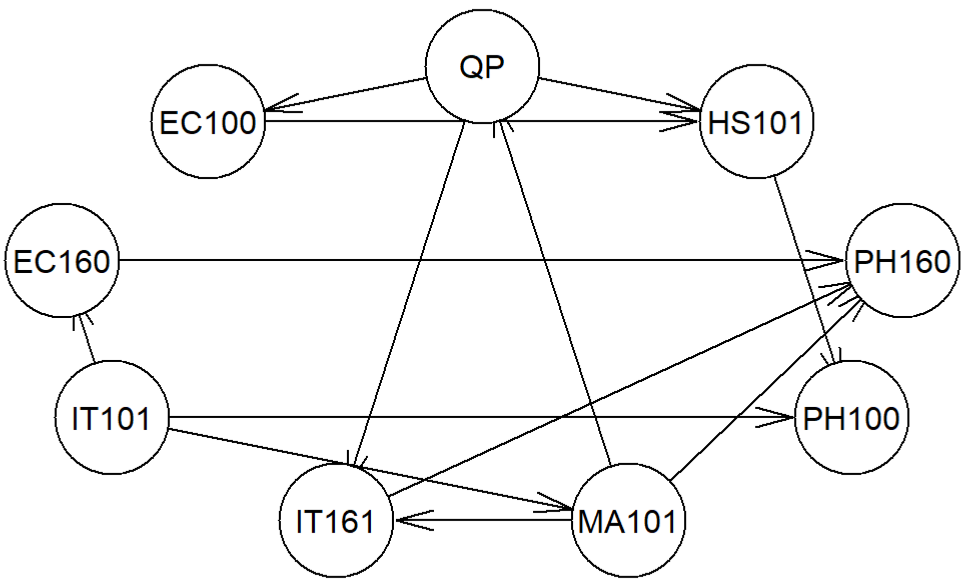
```
##      Train Accuracy
## [1,]         0.8896
```

```
cm_test <- table(test$QP, y_prediction)
accuracy_test = (cm_test[1,1]+cm_test[2,2])/sum(cm_test)
print(round(cbind("Test Accuracy" =accuracy_test), 4))
```

```
##      Test Accuracy
## [1,]        0.8974
```

Q5(b): Repeat this experiment for 20 random selection of training and testing data. Report results about the accuracy of your classifier.

```
dataset_grades=dataset
dataset_grades=dataset_grades[sample(nrow(dataset_grades), 20), ]
split <- sample.split(dataset_grades, SplitRatio = 0.7)
train <- subset(dataset_grades, split == "TRUE")
test <- subset(dataset_grades, split == "FALSE")
train.hc=suppressWarnings(hc(train, score="k2"))
plot(train.hc)
```




```
naive_bayes_classifier<- suppressWarnings(bn.fit(train.hc, train))
y_train <- predict(naive_bayes_classifier,node="QP", data = train)
y_prediction <- predict(naive_bayes_classifier,node="QP", data = test)
cm_train<- table(train$QP, y_train)
accuracy_train = (cm_train[1,1]+cm_train[2,2])/sum(cm_train)
print(round(cbind("Train Accuracy" =accuracy_train), 4))
```

```
##      Train Accuracy
## [1,]         0.9231
```

```
cm_test <- table(test$QP, y_prediction)
accuracy_test = (cm_test[1,1]+cm_test[2,2])/sum(cm_test)
print(round(cbind("Test Accuracy" =accuracy_test), 4))
```

```
##      Test Accuracy
## [1,]             1
```