PLOTS AND EXPLANATIONS FOR HW45R

Student Name: Akshay Nalwaya
Student ID: 200159155

R1. Validation (10 points): Implement Q5 from the [RT] book (under 5.4 exercises)
Note: Implement the function LogisticRegression() in Q1.R. The detailed instruction is in the R file. Also, write the comments for (c) and (d) parts in .pdf file.

Part (c):
Different splits can be made by changing the proportion of dataset used for training and that used for testing.
The three splits were -

- 50% training data, 50% test
  Output:
```
> data_split <- Validation_Split(Default,0.5)
> validation_error <-
LogisticRegression(data_split$trainingset,data_split$validation_set)
> validation_error
[1] 0.0258
```

- 70% training data, 30% test
  Output:
```
> data_split <- Validation_Split(Default,0.7)
> validation_error <-
LogisticRegression(data_split$trainingset,data_split$validation_set)
> validation_error
[1] 0.02333333
```

- 60% training data, 40% test
  Output:
```
> data_split <- Validation_Split(Default,0.6)
> validation_error <-
LogisticRegression(data_split$trainingset,data_split$validation_set)
> validation_error
[1] 0.026
```

Comments:
It is evident that value of error obtained is not constant although we can say that it stays around 2-3% for this data set. This value will change depending on the fraction of data used for training the model.

Part (d):
Including the dummy variable *student* in the training model, we get the below output,
Output:
```
> data_split <- Validation_Split(Default,0.5)
> validation_error <-
LogisticRegression(data_split$trainingset,data_split$validation_set,include_stu
dent = TRUE)
> validation_error
[1] 0.0262
```

Comments:
From the output, we can say that including *student* (dummy) variable does not have much impact on the final output. Although the value of error has changed but this change in error is insignificant (only 1.55% change).

Q4.

(1) List the packages you found (not based on just web search; but install, and explore various examples provided with those packages), describe major functionality of each package, and highlight differences if any.

Ans. Packages that can be used for Bayesian network are:

- **bnlearn** – It is a package for Bayesian network structure learning (via constraint-based, score-based and hybrid algorithms), parameter learning (via ML and Bayesian estimators) and inference. This package implements some algorithms for learning the structure of Bayesian networks.
- **gRain –** The gRain package implements propagation in graphical independence networks. Such networks are also known as probabilistic networks and Bayesian networks. *grain* class in gRain stores a fitted Bayesian network as a list of conditional probability tables and is an important data structure.
- **abn –** Modelling Multivariate Data with Additive Bayesian Networks. An additive Bayesian network model consists of a form of a DAG where each node comprises a generalized linear model, GLM. In this package, model specification is using two matrices (ban and retain).

Comparison of bnlearn and gRain and abn

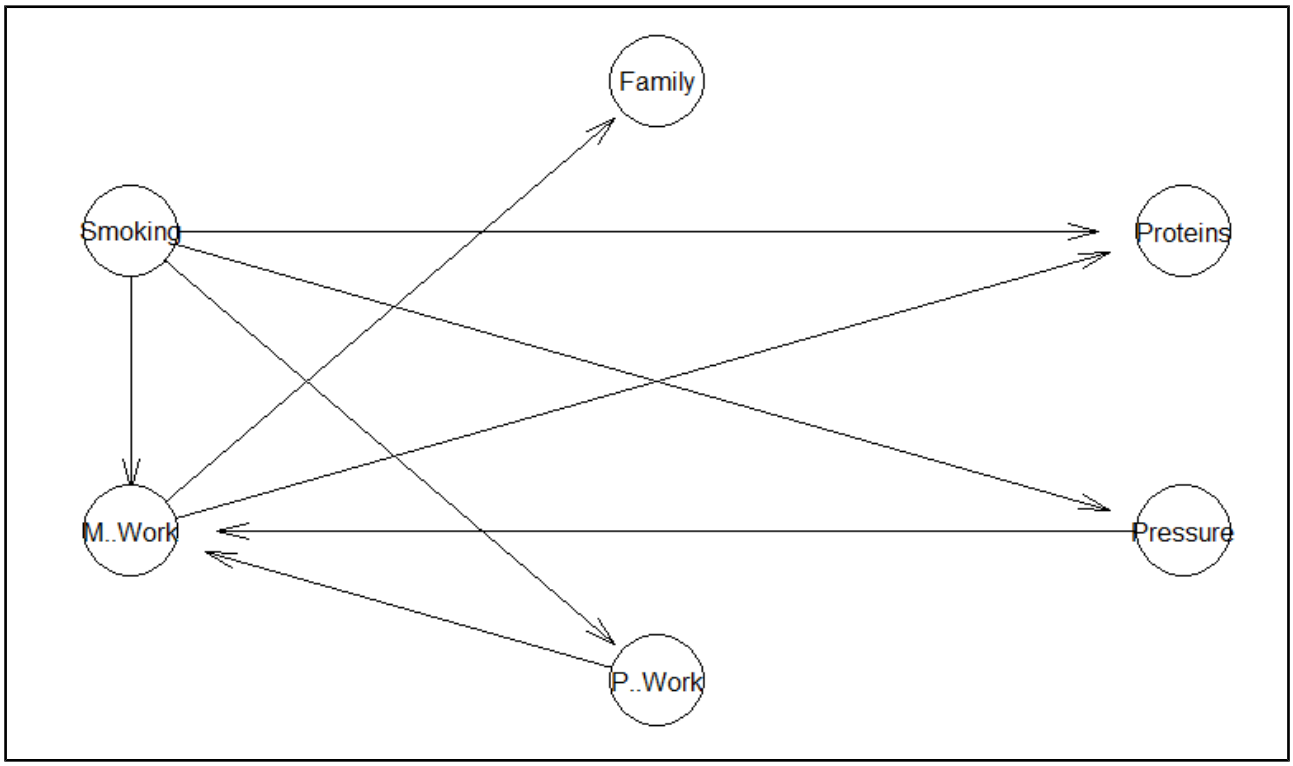
In case of *gRain*, conditional probabilities cannot be NaN. If that happens during execution, then *as.grain()* gives a warning and replaces NaNs with uniform distributions. In such scenarios, *bnlearn* is a better option.

abn does not allow having missing values, we have to either manually remove them or impute them before performing operations.

[Ref.: http://hameddaily.blogspot.com/2015/02/bayesian-network-in-r-introduction.html https://www.r-bloggers.com/bayesian-network-in-r-introduction/]

(2) (a) construct the Bayesian network and draw the resulting network,

RESULTING BAYESIAN NETWORK



(2) (b) compute conditional probabilities for each node (submit resulting conditional probabilities as tables – for each attribute – please format properly so that it's easy to read).

OUTPUT:

```
> bayesian_network()

$Smoking

 Parameters of node Smoking (multinomial distribution)

Conditional probability table:
    no       yes
0.5219989 0.4780011

$M..Work

 Parameters of node M..Work (multinomial distribution)

Conditional probability table:

, , P..Work = no, Pressure = <140

    Smoking
M..Work        no       yes
  no  0.2668919 0.6260504
  yes 0.7331081 0.3739496

, , P..Work = yes, Pressure = <140

    Smoking
M..Work        no       yes
  no  0.8995434 0.8571429
  yes 0.1004566 0.1428571

, , P..Work = no, Pressure = >140

    Smoking
M..Work        no       yes
  no  0.2745902 0.2684564
  yes 0.7254098 0.7315436

, , P..Work = yes, Pressure = >140

    Smoking
M..Work        no       yes
  no  0.8861386 0.8385417
  yes 0.1138614 0.1614583
```

```
$P..Work

 Parameters of node P..Work (multinomial distribution)

Conditional probability table:

    Smoking
P..Work        no       yes
  no  0.5619147 0.4397727
  yes 0.4380853 0.5602273

$Pressure

 Parameters of node Pressure (multinomial distribution)

Conditional probability table:

    Smoking
Pressure        no       yes
  <140 0.5359001 0.6125000
  >140 0.4640999 0.3875000

$Proteins

 Parameters of node Proteins (multinomial distribution)

Conditional probability table:

, , M..Work = no

    Smoking
Proteins        no       yes
   <3 0.6685824 0.6167763
   >3 0.3314176 0.3832237

, , M..Work = yes

    Smoking
Proteins        no       yes
   <3 0.5671982 0.3235294
   >3 0.4328018 0.6764706


$Family

 Parameters of node Family (multinomial distribution)
```

```
Conditional probability table:

   M..Work
Family        no       yes
  neg 0.8814159 0.8227848
  pos 0.1185841 0.1772152

>
```