Rajath Akshay Vanikul

12 October 2018

Task A: Modelling(Classification)

A.1

Logistic regression would be most appropriate to predict whether a loan will be repaid (loan_status) based on the other variables. Reviewing the summary of the data given below, loan_status

- is categorical and has limited number of possible values (Charged Off, Fully paid)
- has no collinearity among the depending variable

Thus, we conclude the most appropriate technique used to predict loan_status is logistic regression.

```
loan <- read.csv("LoanData.csv", header = TRUE) # read the csv to a variable
summary(loan) # summary of the data
```

```
##    loan_amnt             term          int_rate      installment
##  Min.   : 1000    36 months:6580   Min.   : 5.42   Min.   :  22.24
##  1st Qu.: 6000    60 months:3320   1st Qu.: 8.90   1st Qu.: 193.76
##  Median :11200                     Median :12.40   Median : 322.25
##  Mean   :12876                     Mean   :12.39   Mean   : 364.11
##  3rd Qu.:17500                     3rd Qu.:15.20   3rd Qu.: 480.38
##  Max.   :35000                     Max.   :24.10   Max.   :1288.10
##
##  grade     home_ownership   annual_inc          verification_status
##  A:2744   MORTGAGE:4562   Min.   :   6000   Not Verified   :3018
##  B:3077   OWN     : 742   1st Qu.:  42000   Source Verified:3027
##  C:1807   RENT    :4596   Median :  60000   Verified       :3855
##  D:1205                   Mean   :  70328
##  E: 710                   3rd Qu.:  84890
##  F: 291                   Max.   :1782000
##  G:  66
##       loan_status    delinq_2yrs        pub_rec
##  Charged Off:1557   Min.   :0.0000   Min.   :0.00000
##  Fully Paid :8343   1st Qu.:0.0000   1st Qu.:0.00000
##                     Median :0.0000   Median :0.00000
##                     Mean   :0.1332   Mean   :0.04636
##                     3rd Qu.:0.0000   3rd Qu.:0.00000
##                     Max.   :6.0000   Max.   :2.00000
##
```

A.2

Converting the categorical grade variable to numeric will not change the performance of the existing model as logistic regression accepts categorical dependents. Secondly, converting categorical values to numeric will still remain discrete and not continuous to make the model more accurate while employing linear model. Thus, transforming categorical grade variable to numeric will not change the performance of the model.

```
grade_num<-as.numeric(loan$grade) # converting categorical to numeric
summary(grade_num) # summary of the numeric data.

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   2.000   2.515   3.000   7.000
```

A.3

Logistic regression is used to train the model using all the predictors.

```
loan_train <- glm(loan_status ~ ., loan,family = binomial) # logistic
regression to model the output
summary(loan_train) # summary of the trained data

##
## Call:
## glm(formula = loan_status ~ ., family = binomial, data = loan)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0929   0.3358   0.4695   0.6161   1.3103
##
## Coefficients:
##                                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)                        3.600e+00  2.759e-01  13.051  < 2e-16
## loan_amnt                         -2.414e-05  1.557e-05  -1.550  0.12106
## term 60 months                    -3.383e-01  1.112e-01  -3.043  0.00234
## int_rate                          -2.052e-01  3.475e-02  -5.907 3.48e-09
## installment                        5.337e-04  5.503e-04   0.970  0.33210
## gradeB                             3.698e-01  1.657e-01   2.232  0.02562
## gradeC                             5.407e-01  2.517e-01   2.148  0.03168
## gradeD                             7.194e-01  3.381e-01   2.128  0.03334
## gradeE                             9.748e-01  4.132e-01   2.359  0.01831
## gradeF                             9.799e-01  4.884e-01   2.007  0.04479
## gradeG                             1.114e+00  5.940e-01   1.875  0.06082
## home_ownershipOWN                  1.916e-03  1.164e-01   0.016  0.98687
## home_ownershipRENT                -1.446e-01  6.390e-02  -2.262  0.02368
## annual_inc                         9.257e-06  1.009e-06   9.176  < 2e-16
## verification_statusSource Verified 5.297e-02  7.709e-02   0.687  0.49202
## verification_statusVerified        1.414e-01  8.209e-02   1.723  0.08495
## delinq_2yrs                       -1.493e-02  5.858e-02  -0.255  0.79878
## pub_rec                           -3.102e-01  1.178e-01  -2.633  0.00847
```

```
##
## (Intercept)                          ***
## loan_amnt
## term 60 months                       **
## int_rate                             ***
## installment
## gradeB                               *
## gradeC                               *
## gradeD                               *
## gradeE                               *
## gradeF                               *
## gradeG                               .
## home_ownershipOWN
## home_ownershipRENT                   *
## annual_inc                           ***
## verification_statusSource Verified
## verification_statusVerified          .
## delinq_2yrs
## pub_rec                              **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8615.4  on 9899  degrees of freedom
## Residual deviance: 7926.7  on 9882  degrees of freedom
## AIC: 7962.7
##
## Number of Fisher Scoring iterations: 5
```

Estimate column referes to the co-efficient of the point estimate associated with each of the listed variable, also referred to as maximum likelihood estimate. **Standard error(std error)** is known as the standard deviation of the co-efficient of the point estimate.

There more rows in the coefficients table than there are variables in the data because of the categorical values of the variable which also possesses attributes like estimate, standard error, z-value and probability.

A.4

I have used predict() to generate probabilities of the model compared with the test model. I have stored the predicted probabilities in the same dataframe with a different column.

```
library("openxlsx")  # include the library to open the xlsx file
loan_test <- read.xlsx("LoanData_test.xlsx") # read the xlsx to a variable
prob <- predict(loan_train, loan_test, type ='response') # using predict to generate probability of the model
loan_test$prob <- prob # storing the generated probability as an attribute in the same dataframe
```

Investigating the signs of the coefficients of the predictors in the trained model, we can infer that the predictors with positive sign impact the prediction probability of the outcome(loan_status) and the negative sign of the coefficients of the predictors implies the negative impact on the probablility of the outcome. Predictors with positive coefficients like installment,grade,home ownership(OWN) will impact in a posistive prediction of repayment of the loan. However, predictors with negative coefficients like loan amount, 60 month term, interest rate will impact the outcome in negative direction and may result in charged off.

```
summary(loan_train) # summary of the trained data
```

```
##
## Call:
## glm(formula = loan_status ~ ., family = binomial, data = loan)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0929   0.3358   0.4695   0.6161   1.3103
##
## Coefficients:
##                                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)                        3.600e+00  2.759e-01  13.051  < 2e-16
## loan_amnt                         -2.414e-05  1.557e-05  -1.550  0.12106
## term 60 months                    -3.383e-01  1.112e-01  -3.043  0.00234
## int_rate                          -2.052e-01  3.475e-02  -5.907 3.48e-09
## installment                        5.337e-04  5.503e-04   0.970  0.33210
## gradeB                             3.698e-01  1.657e-01   2.232  0.02562
## gradeC                             5.407e-01  2.517e-01   2.148  0.03168
## gradeD                             7.194e-01  3.381e-01   2.128  0.03334
## gradeE                             9.748e-01  4.132e-01   2.359  0.01831
## gradeF                             9.799e-01  4.884e-01   2.007  0.04479
## gradeG                             1.114e+00  5.940e-01   1.875  0.06082
## home_ownershipOWN                  1.916e-03  1.164e-01   0.016  0.98687
## home_ownershipRENT                -1.446e-01  6.390e-02  -2.262  0.02368
## annual_inc                         9.257e-06  1.009e-06   9.176  < 2e-16
## verification_statusSource Verified 5.297e-02  7.709e-02   0.687  0.49202
## verification_statusVerified        1.414e-01  8.209e-02   1.723  0.08495
## delinq_2yrs                       -1.493e-02  5.858e-02  -0.255  0.79878
## pub_rec                           -3.102e-01  1.178e-01  -2.633  0.00847
##
## (Intercept)                       ***
## loan_amnt
## term 60 months                    **
## int_rate                          ***
## installment
## gradeB                            *
## gradeC                            *
## gradeD                            *
## gradeE                            *
## gradeF                            *
```

```
## gradeG                                  .
## home_ownershipOWN
## home_ownershipRENT                      *
## annual_inc                              ***
## verification_statusSource Verified
## verification_statusVerified             .
## delinq_2yrs
## pub_rec                                 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8615.4  on 9899  degrees of freedom
## Residual deviance: 7926.7  on 9882  degrees of freedom
## AIC: 7962.7
##
## Number of Fisher Scoring iterations: 5
```

A.5

Previously calculated prediction is converted to categories of "Charged Off" and "Fully Paid" with the margin of 0.5. These predictions are then compared with the actual loan_status attribute to check for correctness of the prediction. The comparision shows that our prediction is **85%** accurate. Below is the demonstration for the procedure in r:

```
pred<- prob > 0.5
pred = as.factor(pred) # categorizing all the values above and below 0.5
levels(pred) <- c("Charged Off","Fully Paid") # naming the category
loan_test$predic <- ifelse(pred == loan_test$loan_status,1,0) # check for
match in actual and predicted outcomes
correct_pred <- (table(loan_test$predic)) # considering the number of
outcomes
correct_pred <- unname(correct_pred, force = FALSE) # deleting the heading
total_rows <- nrow(loan_test) # number of rows in total
pred_percentage <- (correct_pred[2]/total_rows)*100 # percentage calculation

pred_percentage # accuracy(percentage) output

## [1] 85
```

A.6

Considering a model which simply predicts that all loans will be "Fully Paid". Comparing this model with the default given loan_status prediction to check for accuracy of the prediction resulted in **86%** accuracy. This doesnt really mean that we can prefer the simpler model because of higher accuracy, this can be a chance and not a the right model for different set of data as it doesnt consider alternative outcome like "Charged Off".

Thus, eventhough the accuracy is higher with the simpler model, we can not consider this as a better stable model for different data samples.

```
pred1<- prob > 0.5
pred1 <- as.factor(pred1) # categorizing all the values above and below 0.5
levels(pred1) <- c("Charged Off","Fully Paid") # naming the category
loan_test$predic1 <- ifelse(loan_test$loan_status=="Fully Paid",1,0) # check
for match in actual and predicted outcomes
correct_pred <- (table(loan_test$predic1)) # considering the number of
outcomes
correct_pred <- unname(correct_pred, force = FALSE) # deleting the heading
total_rows <- nrow(loan_test) # number of rows in total
pred_percentage <- (correct_pred[2]/total_rows)*100 # percentage calculation

pred_percentage # accuracy(percentage) output

## [1] 86
```

Task B: Modelling - Regression

B.1

```
mpg_train <- read.csv("auto_mpg_train.csv", header = TRUE) # read the csv to
a variable
mpg_train[mpg_train$horsepower == '?',] # check for missing values

##      mpg cylinders displacement horsepower weight acceleration model.year
## 33   25.0         4           98          ?   2046         19.0         71
## 127  21.0         6          200          ?   2875         17.0         74
## 281  40.9         4           85          ?   1835         17.3         80
## 287  23.6         4          140          ?   2905         14.3         80
## 305  34.5         4          100          ?   2320         15.8         81
## 325  23.0         4          151          ?   3035         20.5         82
##     origin            car.name
## 33       1          ford pinto
## 127      1       ford maverick
## 281      2 renault lecar deluxe
## 287      1   ford mustang cobra
## 305      2         renault 18i
## 325      1       amc concord dl
```

I found that there are 6 missing values in the data provided. Since all are horse power data that is missing, it is impossible to calculate horsepower without RPM and time. Thus, I decided to manually impute the value from different sources on the internet.

I have found the horse power of the cars as follows:

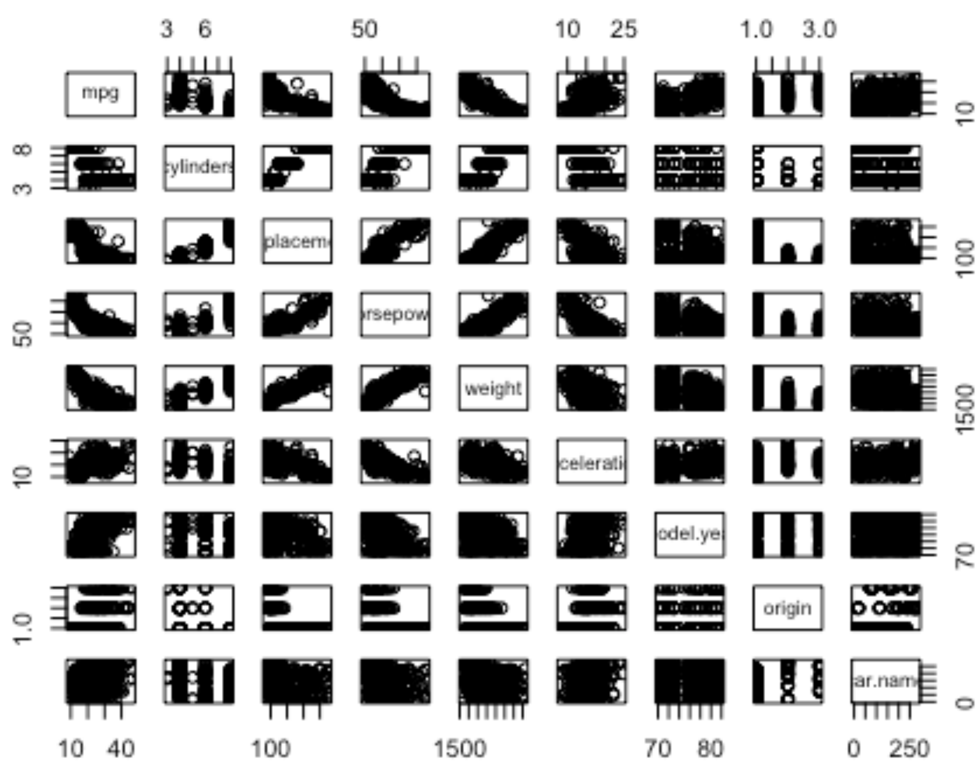
- 75 HP - ford pinto

- 84 HP - ford maverick
- 51 HP - renault lecar deluxe
- 132 HP- ford mustang cobra
- 78 HP - renault 18i
- 82 HP - amc concord dl

Imputing these values in the csv file, we get the complete data set sotored as "auto_mpg_trained.csv".

B.2

```
mpg_edited <- read.csv("auto_mpg_trained.csv", header = TRUE) # read the csv to a variable
plot(mpg_edited) # plot the data
```



from the plot it seems like we have a positive relationship with acceleration, model.year and origin. Thus, including acceleration, model.year and origin in the a multiple linear regression model to predict would give us a better prediction.

Initial set of predictors to use for a multiple linear regression would be **acceleration, model.year and origin**.

B.3

```
mpg_filtered <- mpg_edited[1:8] # Excluding car name column in the data
linear <- lm(mpg ~ acceleration + model.year + origin,mpg_filtered) # linear
regression on the model
summary(linear) # summary of the trained data
```

```
##
## Call:
## lm(formula = mpg ~ acceleration + model.year + origin, data =
mpg_filtered)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.1959  -3.7144  -0.7308   3.3755  13.6644
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -62.21126    5.40363  -11.51  < 2e-16 ***
## acceleration   0.63770    0.10437    6.11  2.7e-09 ***
## model.year     0.91534    0.07501   12.20  < 2e-16 ***
## origin         4.06749    0.35363   11.50  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.121 on 344 degrees of freedom
## Multiple R-squared:  0.5937, Adjusted R-squared:  0.5902
## F-statistic: 167.6 on 3 and 344 DF,  p-value: < 2.2e-16
```

R-squared value ususally explains how much variation in data is explained in the given model. We are interesed in a model that can explain and covering all the variations in the data. Thus, Higher R-squared value from a model is considered favourable.

p-value is often referred to as level of marginal significance within a statistical hypothesis test representing the probability of the occurrence of a given event. A low p-value (often less than 0.05) indecates that the null hypothesis can be rejected.

A good model must possess a low p-value and a high R-squared value.

B.4

The suare of the difference between the given value and the predicted value gives the standard error. Mean of this standard error provides MSE.

$$(given - predicted)^2 = SE$$

$$mean(SE) = MSE$$

```
test_mpg <- read.csv("auto_mpg_test.csv", header = TRUE) # read the csv to a
variable
test_mpg$pred <- predict(linear,test_mpg) # predict using the trained and
```

```
tested.
test_mpg$SE <- (test_mpg$pred - test_mpg$mpg)^2 # calculating the square of the difference
MSE <- mean(test_mpg$SE) # calculating the mean square error
MSE # display the mean square error
```

```
## [1] 19.29113
```

B.5

Desired model is found after examining multiple predictor combination. The final predictor set has improved the model and has a low MLE of 5.373552, low p-vale of 2.2e-16 and a high R-squared value of around 86%.

Final set of predictors used is **acceleration, weight*horsepower, cylinders*displacement, model.year and origin**

```
mpg_edited <- read.csv("auto_mpg_trained.csv", header = TRUE) # read the csv to a variable
linear_imp <- lm(mpg ~ acceleration + weight*horsepower + cylinders*displacement + model.year + origin ,mpg_edited) # linear regression on the model
summary(linear_imp) # summary of the trained data
```

```
##
## Call:
## lm(formula = mpg ~ acceleration + weight * horsepower + cylinders *
##     displacement + model.year + origin, data = mpg_edited)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.3169 -1.6627 -0.1953  1.5524 11.7567
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             6.764e+00  4.856e+00   1.393  0.16456
## acceleration           -9.962e-02  9.573e-02  -1.041  0.29880
## weight                 -1.068e-02  9.927e-04 -10.762  < 2e-16 ***
## horsepower             -2.208e-01  2.807e-02  -7.864 5.04e-14 ***
## cylinders              -6.013e-01  5.073e-01  -1.185  0.23672
## displacement           -1.272e-02  1.752e-02  -0.726  0.46817
## model.year              7.521e-01  4.703e-02  15.993  < 2e-16 ***
## origin                  7.563e-01  2.769e-01   2.731  0.00664 **
## weight:horsepower       4.848e-05  7.199e-06   6.734 7.10e-11 ***
## cylinders:displacement  3.391e-03  2.327e-03   1.457  0.14595
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3 on 338 degrees of freedom
```

```
## Multiple R-squared:  0.863,  Adjusted R-squared:  0.8593
## F-statistic: 236.6 on 9 and 338 DF,  p-value: < 2.2e-16
```

```
test_mpg$pred <- predict(linear_imp,test_mpg) # predict using the trained and tested.
test_mpg$SE <- (test_mpg$pred - test_mpg$mpg)^2 # calculating the square of the difference
MSE <- mean(test_mpg$SE) # calculating the mean square error
MSE # display the mean square error
```

```
## [1] 5.373552
```

Task C: Sampling

C.1

I'm using **inverse sampling** to model the histogram.

Condidering the PDF : $ p(x) = 2e^{???2x} $ for $ x???0 $

integrarting the PDF from 0 to t to arive at CDF.

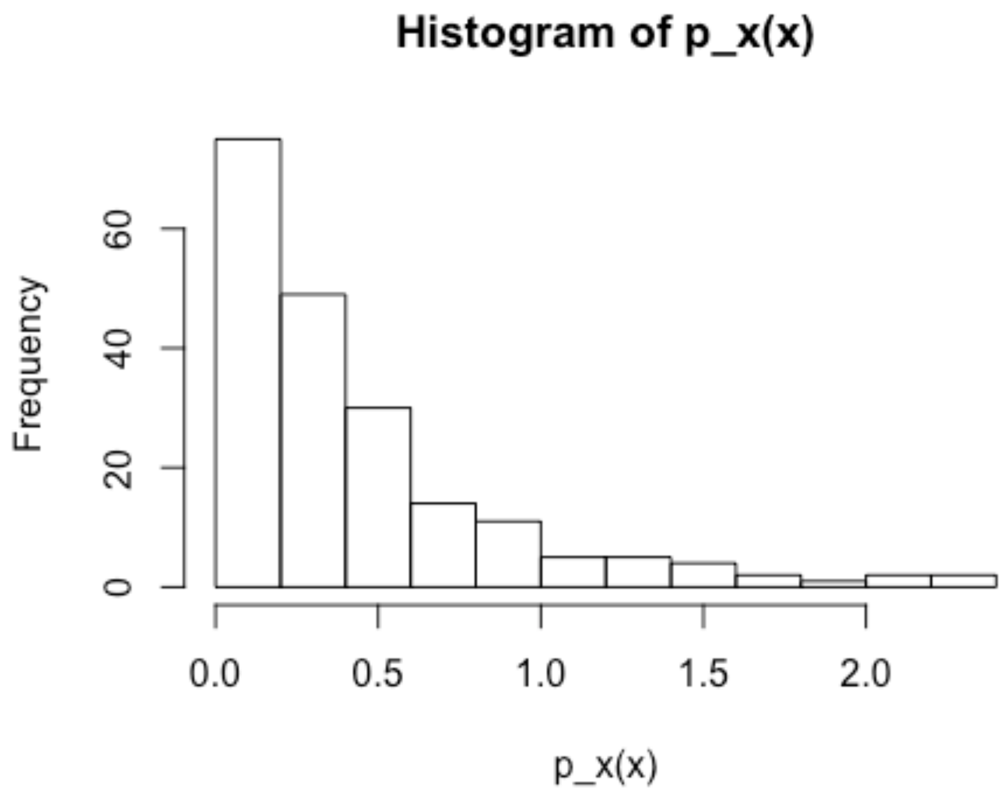
We get a CDF : $x = 1 - e^{-2t}$

We will need to feed in random number sampling to the inverse of CDF.

inverse of CDF : $t = -log(1 - x)/2$

So, we generate random samples using runif and feed it into the inverse of CDF to plot the histogram as shown below.

```
p_x <- function(x){-log(1-x)/2} #create a function
x <- runif(200) # generate random samples
hist(p_x(x),breaks=10) # plot the function with random samples
```



C.2

1. The joint probability distribution is given by

$$p(C = cloudy, S = sprinkler, R = rain, W = wetgrass) = P(C) * P(R|C) * P(S|C) * P(W|R, S)$$

2. **Sprinkler** is conditionally independent to rain in the given model as the probability of sprinkler does not directly change with the change in probabilty of rain. However, they possess a same parent and indirectly related, this relation is considered to be conditionally independent in a Bayesian network.

C.3

We will find the probability of WetGrass(W) given Cloudy(C), Sprinkler(S) and Rain(R). We know that WetGrass(W) in conditionally independent of Cloudy(C) and depends only on Cloudy(C) and Sprinkler(S). This we can conclude that $p(W|C, S, R) = p(W|S, R)$ Below is a program that takes an input of True(T) or False(F) for Sprinkler(S), Rain(R) and WetGrass(W) respectively and outputs the conditional probability accordingly. This is again normalised by dividing it by the sum of all the values in the given row (W=T or W=F).

```
cpt_c <- c(0.5, 0.5) # hard code the probability
cpt_s_given_c <- matrix(c(0.5, 0.5, 0.9, 0.1), 2, 2, byrow = F ) # hard code
```

```
the probability
cpt_r_given_c <- matrix(c(0.8, 0.2, 0.2, 0.8), 2, 2, byrow = F ) # hard code
the probability
cpt_w_given_sr <- matrix(c(1, 0.1, 0.1, 0.01, 0, 0.9, 0.9, 0.99), 2, 4, byrow
= T ) # hard code the probability


row_sum <- rowSums(cpt_w_given_sr)

# define a function to take in values and output normalised probabilty.
p_w_given_crw <- function(S,R,W){
  if (S == T & R == T & W == T) print(paste('The probability
is:',cpt_w_given_sr[2,4]/row_sum[2]))
  if (S == T & R == F & W == F) print(paste('The probability
is:',cpt_w_given_sr[1,2]/row_sum[1]))
  if (S == T & R == T & W == F) print(paste('The probability
is:',cpt_w_given_sr[1,4]/row_sum[1]))
  if (S == T & R == F & W == T) print(paste('The probability
is:',cpt_w_given_sr[2,2]/row_sum[2]))
  if (S == F & R == F & W == T) print(paste('The probability
is:',cpt_w_given_sr[2,1]/row_sum[2]))
  if (S == F & R == T & W == T) print(paste('The probability
is:',cpt_w_given_sr[2,3]/row_sum[2]))
  if (S == F & R == T & W == F) print(paste('The probability
is:',cpt_w_given_sr[1,3]/row_sum[1]))
  if (S == F & R == F & W == F) print(paste('The probability
is:',cpt_w_given_sr[1,1]/row_sum[1]))
  }

p_w_given_crw(S=T,R=T,W=F)

## [1] "The probability is: 0.00826446280991736"
```

C.4

Gibbs sampling to estimate $p(C = T|W = T)$ is performed as follows:

$$p(C = T|W = T) = p(C = T) * p(R = r|C = T) * p(S = s|C = T) * p(W = T|R = r, S = s)/p(W = T))$$

We know, $p(W = T) = p(W = T|C = c, S = s, R = r)$ Therefore,

$$p(C = T|W = T) = p(C = T) * p(R = r|C = T) * p(S = s|C = T)$$

- Initialise the variables "C","S","R" and "W" to either "T" or "F"
- using a loop we then generate a sequence of "C","S","R" and "W"
- We compute p(C|W),p(R|C) and p(S|C) by going through the samples repeatedly.
- Since this forms a Markov chain which converges to the stationary distribution, we generate a large number of sample.

These samples are then used to calsulate the conditional distribution.