Sin Cheuk Kit 54418030 cksin8-c@my.cityu.edu.hk

1. Estimation (20 points)

1.1. Using information from the estimation sample only, estimate a logistic regression model of the purchase decision (buytabw), using all customer attributes in the data file (except customer_no and validation_sample indicator) as independent variables. Display and briefly discuss the marginal effects of the customer attributes on the catalog purchase choice using the maBina function in the erer package.

Summary of Logistic Regression Model in Estimation Sample:

```
> # 1.1:logistic regression (only estimation sample)
> logit <- glm(formula = buytabw ~. -customer_no - validation_sample,
+     family = binomial(link="logit"),
+     data=train_df,x=TRUE)
> summary(logit)

Call:
glm(formula = buytabw ~ . - customer_no - validation_sample,
    family = binomial(link = "logit"), data = train_df, x = TRUE)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3642  -0.6189  -0.3644  -0.1242   3.1298

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.009748   0.095528 -10.570  < 2e-16 ***
tabordrs     0.068339   0.014083   4.853 1.22e-06 ***
divsords     0.009726   0.016803   0.579 0.562718
divwords     0.115872   0.008572  13.518  < 2e-16 ***
spgtabord    0.071413   0.020038   3.564 0.000365 ***
moslsdvs    -0.011327   0.002279  -4.970 6.68e-07 ***
moslsdvw    -0.069529   0.005465 -12.723  < 2e-16 ***
moslstab    -0.041403   0.004544  -9.112  < 2e-16 ***
orders      -0.060513   0.006131  -9.870  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 8976.8  on 9999  degrees of freedom
Residual deviance: 7065.2  on 9991  degrees of freedom
AIC: 7083.2

Number of Fisher Scoring iterations: 6
```

All independent variables except "**divwords**" are statistically significant at significance level α = 0.05

There is **no direct meaning** of coefficient. The coefficients are the size that the log odds ratio increases when each independent variable increases by one unit.

Accuracy of Logistic Regression Model in Estimation Sample:

The customer with predicted purchase probability **more or equal to 0.5** will classify as buy from catalog.

Accuracy = 0.8449 (84.49%). Please find the confusion matrix for more detail about classification result.

```
> # confusionMatrix
> predicts <- as.numeric(logit$fitted.values >= 0.5)
> confusionMatrix(table(predicts,train_df$buytabw))
Confusion Matrix and Statistics


predicts    0    1
       0 8138 1345
       1  206  311

               Accuracy : 0.8449
                 95% CI : (0.8377, 0.8519)
    No Information Rate : 0.8344
    P-Value [Acc > NIR] : 0.002309

                  Kappa : 0.2252

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9753
            Specificity : 0.1878
         Pos Pred Value : 0.8582
         Neg Pred Value : 0.6015
             Prevalence : 0.8344
         Detection Rate : 0.8138
   Detection Prevalence : 0.9483
      Balanced Accuracy : 0.5816

       'Positive' Class : 0
```

Marginal effects of Logistic Regression Model in Estimation Sample:

```
> # marginal effect in log model
> logit_eff <- maBina(logit,x.mean = TRUE,rev.dum = TRUE, digits = 4)
Warning message:
In f.xb * coef(w) :
  Recycling array of length 1 in array-vector arithmetic is deprecated.
  Use c() or as.vector() instead.

> logit_eff
             effect  error  t.value p.value
(Intercept) -0.0835 0.0091  -9.2046  0.0000
tabordrs     0.0057 0.0012   4.7582  0.0000
divsords     0.0008 0.0014   0.5784  0.5630
divwords     0.0096 0.0008  11.6223  0.0000
spgtabord    0.0059 0.0017   3.5423  0.0004
moslsdvs    -0.0009 0.0002  -4.9199  0.0000
moslsdvw    -0.0057 0.0003 -16.7455  0.0000
moslstab    -0.0034 0.0004  -9.5769  0.0000
orders      -0.0050 0.0005  -9.2308  0.0000
>
```

1. Tabordrs: 1% increase in the size of total orders from tabloids leads to an **increase** in the probability of buy from catalog by 0.0057 (i.e. 0.57 percentage points less likely).

2. Divsords: 1% increase in the size of orders with shoe division leads to an **increase** in the probability of buy from catalog by 0.0008 (i.e. 0.08 percentage points less likely).

3. Divwords: 1% increase in the size of orders with women's division leads to an **increase** in the probability of buy from catalog by 0.0096 (i.e. 0.96 percentage points less likely).

4. Spgtabord: 1% increase in the size of total spring tabloid orders leads to an **increase** in the probability of buy from catalog by 0.0059 (i.e. 0.59 percentage points less likely).

5. Moslsdvs: 1% increase in the size of months since last shoe order leads to a **decrease** in the probability of buy from catalog by 0.0009 (i.e. 0.09 percentage points less likely).

6. Moslsdvw: 1% increase in the size of months since last women's order leads to a **decrease** in the probability of buy from catalog by 0.0057 (i.e. 0.57 percentage points less likely).

7. Moslstab: 1% increase in the size of months since last tab order leads to a **decrease** in the probability of buy from catalog by 0.0034 (i.e. 0.34 percentage points less likely).

8. Orders: 1% increase in the size of Total orders leads to a **decrease** in the probability of buy from catalog by 0.0050 (i.e. 0.5 percentage points less likely).

1.2. Try one of your most favorite machine learning methods. The purchase decision (buytabw) is a binary outcome. Using "regression" should restrict the outcome to [0, 1]. You can simply change all predicted values below 0 to zero, and all predicted value above 1 to 1.

Classification with **Neural Networks**:

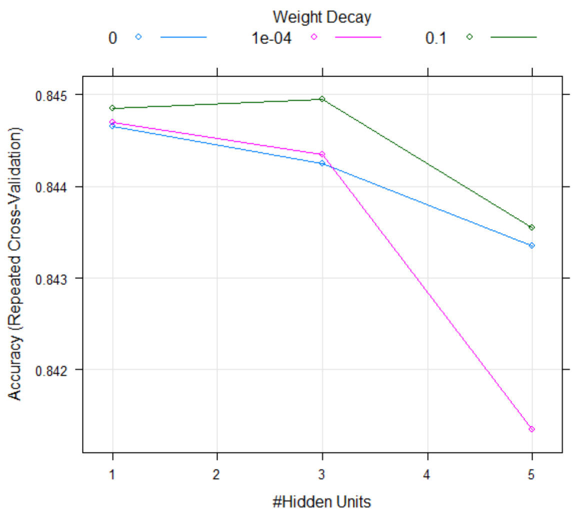
Neural Networks generated by using R package "caret". **To avoid overfitting problem, 5-fold cross-validation will repeat 2 times.**

```
# 1.2:
# NN
TrainingParameters <- trainControl(method = "repeatedcv", number = 5, repeats=2)
NN_model <- train(buytabw ~. -customer_no - validation_sample,data = train_df,
                  method = "nnet",
                  trControl= TrainingParameters,
                  preProcess=c("scale","center"),
                  na.action = na.omit)
```

Summary of Neural Networks in Estimation Sample:

```
Neural Network

10000 samples
   10 predictor
    2 classes: '0', '1'

Pre-processing: scaled (8), centered (8)
Resampling: Cross-Validated (5 fold, repeated 2 times)
Summary of sample sizes: 8001, 8000, 7999, 8000, 8000, 8001, ...
Resampling results across tuning parameters:

  size  decay  Accuracy   Kappa
  1     0e+00  0.8446503  0.2165832
  1     1e-04  0.8447003  0.2167134
  1     1e-01  0.8448502  0.2122834
  3     0e+00  0.8442497  0.2075305
  3     1e-04  0.8443499  0.1950142
  3     1e-01  0.8449501  0.2016102
  5     0e+00  0.8433499  0.2082246
  5     1e-04  0.8413502  0.1995429
  5     1e-01  0.8435499  0.2011000

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were size = 3 and decay = 0.1.
```

Note that accuracy was used to select the optimal model using the largest value. The final values used for the model were size = 3 and decay = 0.1.



Accuracy of Neural Networks in Estimation Sample:

Accuracy = 0.8466 (84.66%).

Please find the confusion matrix for more detail about classification result.

```
> NN_train <-predict(NN_model, train_df)
> # Create confusion matrix
> CM_NN <-confusionMatrix(NN_train, train_df$buytabw)
> CM_NN
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 8205 1395
         1  139  261

               Accuracy : 0.8466
                 95% CI : (0.8394, 0.8536)
    No Information Rate : 0.8344
    P-Value [Acc > NIR] : 0.0004857

                  Kappa : 0.2025

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9833
            Specificity : 0.1576
         Pos Pred Value : 0.8547
         Neg Pred Value : 0.6525
             Prevalence : 0.8344
         Detection Rate : 0.8205
   Detection Prevalence : 0.9600
      Balanced Accuracy : 0.5705

       'Positive' Class : 0
```

In **estimation** dataset, the model accuracy of logistic regression and Neural Networks is 84.49% and 84.66% respectively. It represents that the performance of **Neural Networks** is better than logistic regression.

2. Predicted purchase probability in the validation sample (10 points)

2.1. Predict the purchase probability for all customers in the **validation sample** using the predict function. Verify that the predicted purchase probability variable was created and that it has reasonable values. **(0~1)?**

Fit the validation sample in logistic regression model and Neural Networks which build in part 1.

Logistic Regression Model in Validation Sample: | **Neural Networks in Validation Sample:**

Checking of Predicted Purchase Probability

Logistic Regression Model in Validation Sample:

```
> # 2.1 :fit model in validation data
> probabilities <- logit %>% predict(vaild_df, type = "response")
>
> min(probabilities)
[1] 0.0006788839
> max(probabilities)
[1] 0.9962092
```

All predicted purchase probability between 0 to 1.

Neural Networks in Validation Sample:

```
> NN_valid_prob <- predict(NN_model, valid_df, type='prob')
> min(NN_valid_prob)
[1] 0.001139213
> max(NN_valid_prob)
[1] 0.9988608
```

All predicted purchase probability between 0 to 1.

Accuracy of Model in Validation Sample:

Logistic Regression Model in Validation Sample:

Accuracy = 0.8375 (83.75%).

Please find the confusion matrix for more detail about classification result.

```
> # confusionMatrix
> predicts <- as.numeric(probabilities >= 0.5)
> confusionMatrix(table(predicts,vaild_df$buytabw))
Confusion Matrix and Statistics


predicts    0    1
       0 8035 1408
       1  217  340

               Accuracy : 0.8375
                 95% CI : (0.8301, 0.8447)
    No Information Rate : 0.8252
    P-Value [Acc > NIR] : 0.0005706

                  Kappa : 0.23

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9737
            Specificity : 0.1945
         Pos Pred Value : 0.8509
         Neg Pred Value : 0.6104
             Prevalence : 0.8252
         Detection Rate : 0.8035
   Detection Prevalence : 0.9443
      Balanced Accuracy : 0.5841

       'Positive' Class : 0
```

Neural Networks in Validation Sample:

Accuracy = 0.8373 (83.73%).

Please find the confusion matrix for more detail about classification result.

```
> # NN
> NN_valid <-predict(NN_model, valid_df)
> # Create confusion matrix
> CM_NN <-confusionMatrix(NN_valid, valid_df$buytabw)
> CM_NN
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 8083 1458
         1  169  290

               Accuracy : 0.8373
                 95% CI : (0.8299, 0.8445)
    No Information Rate : 0.8252
    P-Value [Acc > NIR] : 0.0006883

                  Kappa : 0.205

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9795
            Specificity : 0.1659
         Pos Pred Value : 0.8472
         Neg Pred Value : 0.6318
             Prevalence : 0.8252
         Detection Rate : 0.8083
   Detection Prevalence : 0.9541
      Balanced Accuracy : 0.5727

       'Positive' Class : 0
```

In **validation** dataset, the model accuracy of logistic regression and Neural Networks is 83.75% and 83.73% respectively. It represents that the performance of **logistic regression** is better than Neural Networks.

2.2. From now on, you should only work with observations in the validation sample. Make sure that you do not accidentally include observations from the estimation sample in the analysis!

3. Box plot of predicted purchase probabilities (10 points)

Show box plots of the predicted purchase probabilities using the boxplot function, separately for customers who made a purchase after receiving the catalog and those who did not respond:
Do the box plots indicate that the model has some power to predict who is likely to purchase in the validation sample?

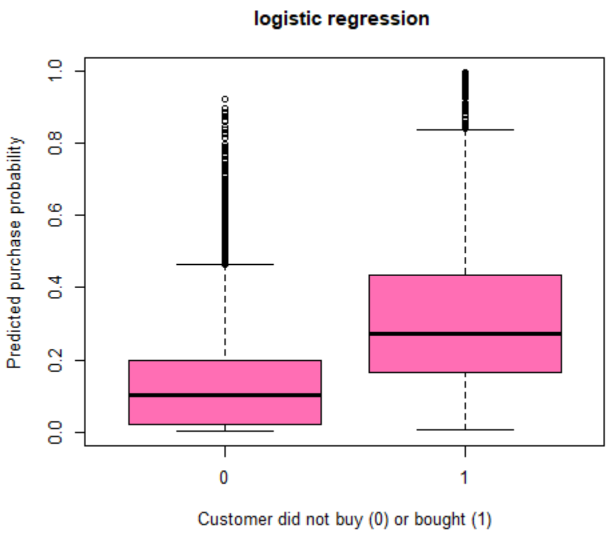
The heads of validation sample:

validation_sample	logit_prob	NN_prob.0	NN_prob.1	NN_pred
1	0.267008250	0.6818141	0.318185917	0
1	0.039837971	0.9964398	0.003560167	0
1	0.124983806	0.8467510	0.153248991	0
1	0.168890988	0.8199524	0.180047641	0
1	0.074569725	0.9255397	0.074460327	0

Box plots show that both of logistic regression model and neural networks have **almost some power** to predict who is likely to purchase in the validation sample.

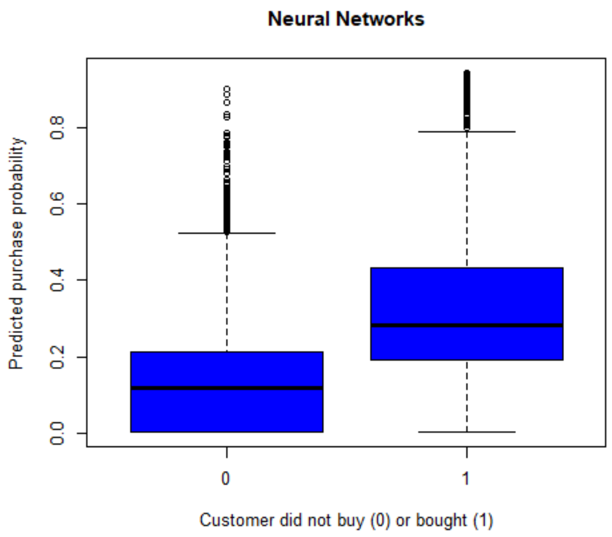
Logistic Regression Model

```
# logit
par(cex = 0.80)
boxplot(logit_prob ~ buytabw, data = valid_df,
        col = "hotpink1",
        xlab = "Customer did not buy (0) or bought (1)",
        ylab = "Predicted purchase probability",
        main = "logistic regression ")
```



Neural Networks

```
# NN
par(cex = 0.80)
boxplot(valid_df$NN_prob[,2] ~ buytabw, data = valid_df,
        col = "blue",
        xlab = "Customer did not buy (0) or bought (1)",
        ylab = "Predicted purchase probability",
        main = "Neural Networks")
```



4. Scoring and segmentation (10 points)

Score the customers and segment the customers into ten deciles, where score = 1 corresponds to the customers with the lowest predicted purchase probabilities and score = 10 corresponds to the customers with the highest predicted purchase probabilities. Employ the createBins function for this task. Now create a summary data set, score_DF, that contains some key summary statistics separately for each segment (score). Include these summary statistics: • Number of observations in segment • Number of buyers in segment • Mean *predicted* purchase probability • Mean *observed* purchase rate (based on buytabw)

For Logistic Regression:

```
# logit
logit_bin <- createBins(valid_df$logit_prob,10)
valid_df$logit_bin <- logit_bin
# SS refer to Scoring and segmentation
SS_logit_bin <-  valid_df %>%
  group_by(logit_bin) %>%
  summarize(No_of_observation = n(),
            No_of_buyers = sum(buytabw==1),
            Mean_predicted_purchase_prob = mean(logit_prob),
            Mean_observed_purchase_rate=sum(buytabw==1)/n() )
# Sorting
SS_logit_bin <- SS_logit_bin[order(-SS_logit_bin$logit_bin),]
```

	logit_bin	No_of_observation	No_of_buyers	Mean_predicted_purchase_prob	Mean_observed_purchase_rate
1	10	1000	520	0.565831210	0.520000000
2	9	1000	336	0.325663006	0.336000000
3	8	998	249	0.240438986	0.249498998
4	7	1002	210	0.187161925	0.209580838
5	6	1000	196	0.147128833	0.196000000
6	5	1000	151	0.110500668	0.151000000
7	4	999	81	0.069533363	0.081081081
8	3	1001	4	0.032928973	0.003996004
9	2	1000	0	0.012973255	0.000000000
10	1	1000	1	0.004172469	0.001000000

"logit_bin" refer to score

For Neural Networks

```
# NN
NN_bin <- createBins(valid_df$NN_prob[,2],10)
valid_df$NN_bin <- logit_bin
# SS refer to Scoring and segmentation
SS_NN_bin <-  valid_df %>%
  group_by(NN_bin) %>%
  summarize(No_of_observation = n(),
            No_of_buyers = sum(buytabw==1),
            Mean_predicted_purchase_prob = mean(NN_prob[,2]),
            Mean_observed_purchase_rate=sum(buytabw==1)/n() )

# Sorting
SS_NN_bin <- SS_NN_bin[order(-SS_NN_bin$NN_bin),]
```

	NN_bin	No_of_observation	No_of_buyers	Mean_predicted_response_rate	Mean_observed_purchase_rate
1	10	1000	520	0.524484426	0.520000000
2	9	1000	336	0.326582254	0.336000000
3	8	998	249	0.245674002	0.249498998
4	7	1002	210	0.209600277	0.209580838
5	6	1000	196	0.181783519	0.196000000
6	5	1000	151	0.137442290	0.151000000
7	4	999	81	0.061721551	0.081081081
8	3	1001	4	0.007083829	0.003996004
9	2	1000	0	0.001558897	0.000000000
10	1	1000	1	0.001223480	0.001000000

"NN_bin" refer to score

5. Lift and gains (10 points)

Create a table indicating the lift, cumulative lift, and cumulative gains from the **predictive model**. Plot the lift, cumulative lift, and cumulative gains chart. Interpret and discuss the lifts and gains: Is the predictive model useful for targeting purposes?

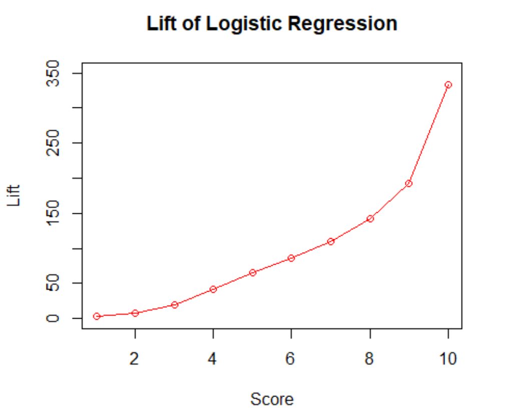
For Logistic Regression:

```
> # 5. Lift and gains
> # logic
> logit_avg_pre_purchase_prob <- mean(valid_df$logit_prob)
> logit_avg_pre_purchase_prob
[1] 0.169619
> # LG refer to Lift and gains
> LG_logit <- SS_logit_bin
> # lift
> LG_logit$lift = 100 * LG_logit$Mean_predicted_purchase_prob / logit_avg_pre_purchase_prob
> # cum lift
> logit_cum_pre_purchase_prob <- vector()
> for (i in 1:10){
+   logit_cum_pre_purchase_prob[i] <- mean(LG_logit$Mean_predicted_purchase_prob[1:i])
+ }
> LG_logit$cum_lift = 100 * logit_cum_pre_purchase_prob / logit_avg_pre_purchase_prob
> # cum gain
> logit_cum_gain <- vector()
> for (i in 1:10){
+   logit_cum_gain[i] <- LG_logit$cum_lift[i] * (i/10)
+ }
> LG_logit$cum_gain = logit_cum_gain
```

Average predicted purchase probability of logistic regression model: **0.1696**

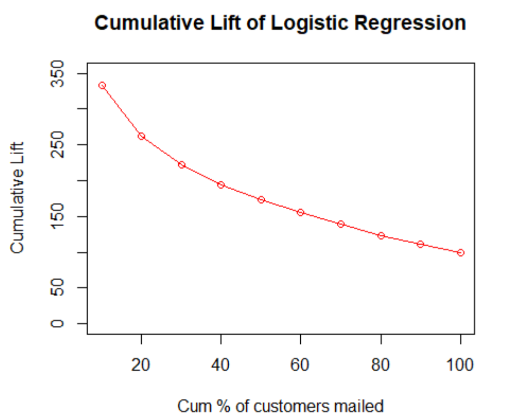
	logit_bin	No_of_observation	No_of_buyers	Mean_predicted_purchase_prob	Mean_observed_purchase_rate	lift	cum_lift	cum_gain
1	10	1000	520	0.565831210	0.520000000	333.589614	333.5896	33.35896
2	9	1000	336	0.325663006	0.336000000	191.996826	262.7932	52.55864
3	8	998	249	0.240438986	0.249498998	141.752429	222.4463	66.73389
4	7	1002	210	0.187161925	0.209580838	110.342578	194.4204	77.76814
5	6	1000	196	0.147128833	0.196000000	86.740798	172.8844	86.44222
6	5	1000	151	0.110500668	0.151000000	65.146416	154.9281	92.95687
7	4	999	81	0.069533363	0.081081081	40.993864	138.6518	97.05625
8	3	1001	4	0.032928973	0.003996004	19.413498	123.7470	98.99760
9	2	1000	0	0.012973255	0.000000000	7.648470	110.8472	99.76245
10	1	1000	1	0.004172469	0.001000000	2.459908	100.0084	100.00844

```
#plot lift
plot(LG_logit$logit_bin,LG_logit$lift,type='o', col='red',
     ylim=c(0,350), xlab = "Score", ylab='Lift',
     main="Lift of Logistic Regression")
```



Lift plot shows that when the score of customers is higher (high predicted purchase probability), the lift will be increased.

```
#plot cum lift
plot(seq(10,100,by=10),LG_logit$cum_lift,type='o', col='red',
     ylim=c(0,350), xlab = "CUm % of customers mailed", ylab='Cumulative Lift',
     main="Cumulative Lift of Logistic Regression")
```



Cumulative lift plot shows that when cumulative lift increase, the cumulative lift will be decreased.

```
#plot cum gain
plot(seq(10,100,by=10),LG_logit$cum_gain,type='o', col='red',
     ylim=c(0,100), xlab = "CUm % of customers mailed", ylab='Cumulative Gain',
     main="Cumulative Gain of Logistic Regression")
lines(seq(10,100,by=10),seq(10,100,by=10), col='blue')
legend("bottomright", legend=c("% of customers captured by target mailing","random mailing"),
       col=c("red",'blue'), lty=1, cex=0.75)
```

Cumulative gain plot shows most of potential customers can be skimmed off by targeting the top scores according to logistic regression model.

There is clear evidence that **logistic regression model** can predict buying behaviour in validation sample.

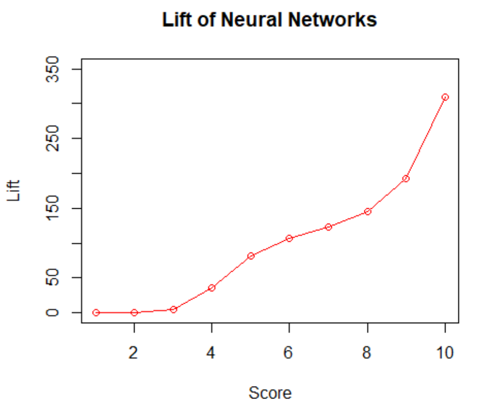
For Neural Networks:

```
> # NN
> NN_avg_pre_purchase_prob <- mean(valid_df$NN_prob[,2])
> NN_avg_pre_purchase_prob
[1] 0.1697028
> # LG refer to Lift and gains
> LG_NN <- SS_NN_bin
> # lift
> LG_NN$lift = 100 * LG_NN$Mean_predicted_purchase_prob / NN_avg_pre_purchase_prob
> # cum lift
> NN_cum_pre_purchase_prob <- vector()
> for (i in 1:10){
+   NN_cum_pre_purchase_prob[i] <- mean(LG_NN$Mean_predicted_purchase_prob[1:i])
+ }
> LG_NN$cum_lift = 100 * NN_cum_pre_purchase_prob / NN_avg_pre_purchase_prob
> # cum gain
> NN_cum_gain <- vector()
> for (i in 1:10){
+   NN_cum_gain[i] <- LG_NN$cum_lift[i] * (i/10)
+ }
> LG_NN$cum_gain = NN_cum_gain
```

Average predicted purchase probability of neural networks: **0.1697**

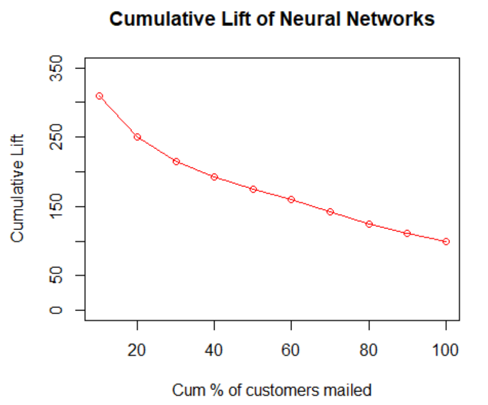
	NN_bin	No_of_observation	No_of_buyers	Mean_predicted_purchase_prob	Mean_observed_purchase_rate	lift	cum_lift	cum_gain
1	10	1000	520	0.524484426	0.520000000	309.0606085	309.0606	30.90606
2	9	1000	336	0.326582254	0.336000000	192.4436744	250.7521	50.15043
3	8	998	249	0.245674002	0.249498998	144.7672284	215.4238	64.62715
4	7	1002	210	0.209600277	0.209580838	123.5102246	192.4454	76.97817
5	6	1000	196	0.181783519	0.196000000	107.1187668	175.3801	87.69005
6	5	1000	151	0.137442290	0.151000000	80.9900078	159.6484	95.78905
7	4	999	81	0.061721551	0.081081081	36.3703842	142.0373	99.42609
8	3	1001	4	0.007083829	0.003996004	4.1742566	124.8044	99.84352
9	2	1000	0	0.001558897	0.000000000	0.9186041	111.0393	99.93538
10	1	1000	1	0.001223480	0.001000000	0.7209548	100.0075	100.00747

```
#plot lift
plot(LG_NN$NN_bin,LG_NN$lift,type='o', col='red',
     ylim=c(0,350), xlab = "Score", ylab='Lift',
     main="Lift of Neural Networks")
```



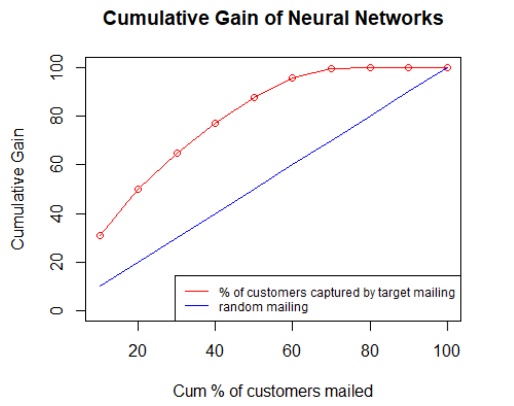
Lift plot shows that when the score of customers is higher (high predicted purchase probability), the lift will be increased.

```
#plot cum lift
plot(seq(10,100,by=10),LG_NN$cum_lift,type='o', col='red',
     ylim=c(0,350), xlab = "Cum % of customers mailed", ylab='Cumulative Lift',
     main="Cumulative Lift of Neural Networks")
```



Cumulative lift plot shows that when cumulative lift increase, the cumulative lift will be decreased.

```
#plot cum gain
plot(seq(10,100,by=10),LG_NN$cum_gain,type='o', col='red',
     ylim=c(0,100), xlab = "Cum % of customers mailed", ylab='Cumulative Gain',
     main="Cumulative Gain of Neural Networks")
lines(seq(10,100,by=10),seq(10,100,by=10), col='blue')
legend("bottomright", legend=c("% of customers captured by target mailing","random mailing"),
       col=c("red",'blue'), lty=1, cex=0.75)
```



Cumulative gain plot shows most of potential customers can be skimmed off by targeting the top scores according to neural networks.

There is clear evidence that **Neural Networks** can predict buying behaviour in validation sample.

6. Profitability analysis (10 points) **Note that the profitability analysis base on validation sample only**

From now on work again with the customer-level data in catalog_DF. Use the following data:

• Based on past data, the average dollar margin per customer is $ 26.90 • The cost of printing and mailing one tabloid is $ 1.40

Using the predicted purchase probability, calculate expected profits. Provide a histogram of the expected profits variable. Discuss the graph. Calculate the fraction of customers who are expected to be profitable, i.e. have positive expected profits. Now rank customers according to their expected profitability. Then calculate realized profits, based on the observed purchase decision of each customer.

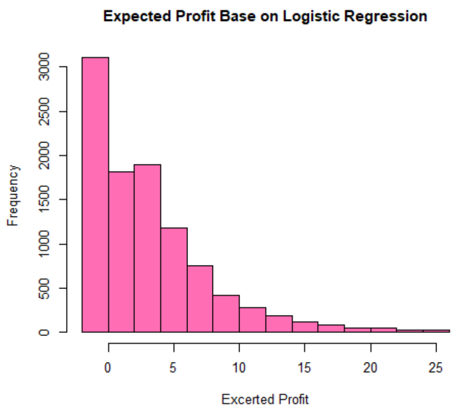
Calculate the cumulative sum of realized profits for a targeting strategy where customers are targeted in descending order of expected profits. Plot the cumulative realized profits on the y-axis versus the percent of customers mailed on the x-axis. Discuss your findings.

$$\text{E(profit)} = \text{Prob(response)} \times \text{margin} - \text{cost of contact} = \text{Prob(response)} \times \$26.9 - \$1.4$$

$$\text{Realized Profit} = \textbf{buy decision} \times \text{margin} - \text{cost of contact} = \text{buy decision} \times \$26.9 - \$1.4$$
$$\text{if buy, } \textbf{buy decision} = 1, \text{ if not buy, } \textbf{buy decision} = 0$$

For Logistic Regression:

Expected Profit:

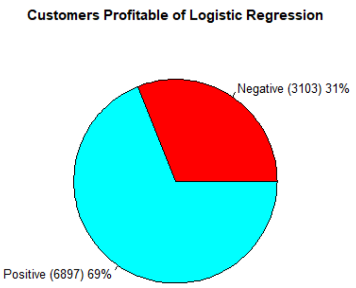



```
# 6: Profitability analysis
# expected profit
# logit
valid_df$logit_expected_profit = (valid_df$logit_prob * 26.9) - 1.4
# plot
hist(valid_df$logit_expected_profit,col='hotpink1',
     xlab = "Excerted Profit", ylab='Frequency',
     main="Expected Profit Base on Logistic Regression")
```

```
> summary(valid_df$logit_expected_profit)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
-1.3817 -0.5198  2.0660  3.1627  5.0224 25.3980 
```

Based on logistic regression model, expected profit **follows right skewed distribution**. The **median** of expected profit ($2.0630) less than **mean** of expected profit ($3.1627), it represents that there are more than half customers with expected profit less than $3.1627.

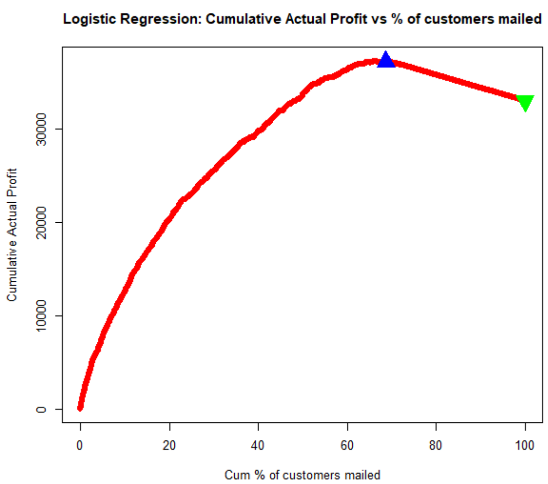
Fraction of customers who are expected to be profitable:




```
> # + or - profit?
> logit_sign_profit <- table(sign(valid_df$logit_expected_profit))
> logit_sign_profit

  -1    1 
3103 6897 
> 
> # Pie Chart with Percentages
> slices <- c(logit_sign_profit[1],logit_sign_profit[2])
> lbls <- c("Negative (3103)", "Positive (6897)")
> pct <- round(slices/sum(slices)*100)
> lbls <- paste(lbls, pct) # add percents to labels
> lbls <- paste(lbls,"%",sep="") # ad % to labels
> pie(slices,labels = lbls, col=rainbow(length(lbls)),
+     main="Customers Profitable of Logistic Regression")
```

3103 (31%) customers had negative expected profit. 6897 (69%) customers had positive expected profit.

```
> #plot cumulative realized profits
> plot(seq(0.01,100,by=0.01),cumsum(valid_df$logit_actual_profit),type='o', col='red',
+          xlab = "Cum % of customers mailed", ylab='Cumulative Actual Profit',
+          main="Logistic Regression: Cumulative Actual Profit vs % of customers mailed")
> points(x=68.7,y=max(cumsum(valid_df$logit_actual_profit)),col="blue",bg="blue",pch=24,cex=2.5)
> points(x=100,y=cumsum(valid_df$logit_actual_profit)[10000],col="green",bg="green",pch=25,cex=2.5)
> 
> # maximum point
> max(cumsum(valid_df$logit_actual_profit))
[1] 37246
> # tail point
> cumsum(valid_df$logit_actual_profit)[10000]
[1] 33021.2
```

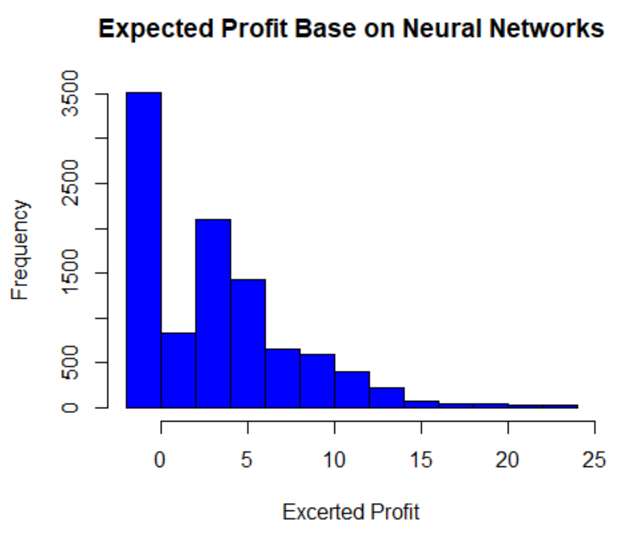


Since cumulative actual profit computed by descending order, the drop down of cumulative actual profit will start from **69%** customers mailed.

For more details, the **maximum** point of cumulative actual profit is $37246 when 69% customers mailed. Then, the cumulative actual profit has been falling to $33021.2 when all customers mailed.

For Neural Networks:

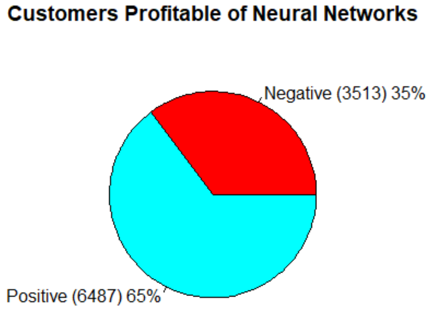
Expected Profit:




```
> # expected profit
> # NN
> valid_df$NN_expected_profit = (valid_df$NN_prob[,2] * 26.9) - 1.4
> # plot
> hist(valid_df$NN_expected_profit,col='blue',
+      xlab = "Excerted Profit", ylab='Frequency',
+      main="Expected Profit Base on Neural Networks")
>
> summary(valid_df$NN_expected_profit)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 -1.369  -1.322   2.880   3.165   5.242  23.778 
```

Based on neural networks model, expected profit follows **right skewed distribution**. The **median** of expected profit ($2.880) less than **mean** of expected profit ($3.165), it represents that there are more than half customers with expected profit less than $3.165.

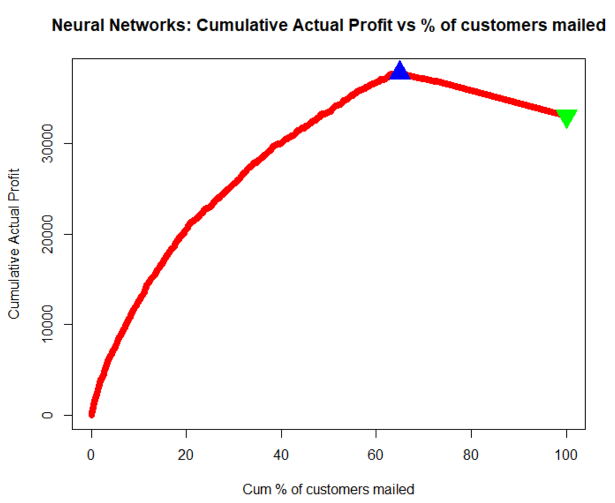
Fraction of customers who are expected to be profitable:




```
> # + or - profit?
> NN_sign_profit <- table(sign(valid_df$NN_expected_profit))
> NN_sign_profit

  -1    1 
3513 6487 
>
> # Pie Chart with Percentages
> slices <- c(NN_sign_profit[1],NN_sign_profit[2])
> lbls <- c("Negative (3513)", "Positive (6487)")
> pct <- round(slices/sum(slices)*100)
> lbls <- paste(lbls, pct) # add percents to labels
> lbls <- paste(lbls,"%",sep="") # ad % to labels
> pie(slices,labels = lbls, col=rainbow(length(lbls)),
+     main="Customers Profitable of Neural Networks")
```

3513 (35%) customers had negative expected profit. 6487 (65%) customers had positive expected profit.

```
> #plot cumulative realized profits
> plot(seq(0.01,100,by=0.01),cumsum(valid_df$NN_actual_profit),type='o', col='red',
+      xlab = "Cum % of customers mailed", ylab='Cumulative Actual Profit',
+      main="Neural Networks: Cumulative Actual Profit vs % of customers mailed")
> points(x=64.87,y=max(cumsum(valid_df$NN_actual_profit)),col="blue",bg="blue",pch=24,cex=2.5)
> points(x=100,y=cumsum(valid_df$NN_actual_profit)[10000],col="green",bg="green",pch=25,cex=2.5)
>
> # maximum point
> max(cumsum(valid_df$NN_actual_profit))
[1] 37763.4
> # tail point
> cumsum(valid_df$NN_actual_profit)[10000]
[1] 33021.2
```



Since cumulative actual profit computed by descending order, the drop down of cumulative actual profit will start from 65% customers mailed.

For more details, the **maximum point** of cumulative actual profit is $37763 when 65% customers mailed. Then, the cumulative actual profit has been falling to $33021.2 when all customers mailed.

Neural networks can provide higher actual profit ($37763) than logistic regression ($37246).

7. Recommended targeting strategy (20 points)
What mailing strategy do you recommend? Compare the actual profitability from your proposed strategy to

1. The expected profitability based on your model.

Compute expected profitability of logistic regression model, neural networks and actual.

```
# 7 Recommended targeting strategy
# Actual Expected profit
actual_expected_profit <- round(mean((as.numeric(valid_df$buytabw)-1) * 26.9 - 1.4),4)
actual_expected_profit
# logit Expected profit
logit_expected_profit <- round(mean(valid_df$logit_expected_profit),4)
logit_expected_profit
# NN Expected profit
NN_expected_profit <- round(mean(valid_df$NN_expected_profit),4)
NN_expected_profit

# Create data frame to compare expected profit among 2 models and actual
compare_profit_df <- data.frame(Model=c("Actual","Logistic Regression","Neural Networks"),
                                Expected_Profit=c(actual_expected_profit,logit_expected_profit,NN_expected_profit))
compare_profit_df$Different_From_Actual = abs(compare_profit_df$Expected_Profit - actual_expected_profit)
```

	Model	Expected_Profit	Different_From_Actual
1	Actual	3.3021	0.0000
2	Logistic Regression	3.1627	0.1394
3	Neural Networks	3.1650	0.1371

The expected profit of actual, logistic regression model and neural network is 3.3021, 3.1627 and 3.1650 respectively.
In addition, expected profit of **neural network** is closer to actual.

Take expected profit as the criterion, **neural network** recommended to be mailing strategy.

2. A mass mailing strategy where each customer receives a catalog.

Estimate response model separately for treated and untargeted customers. W = 0,1 is the targeting indicator.

For Logistic Regression:

```
# logit conditional prob
logit_prob_df <- valid_df %>%
  group_by(buytabw) %>%
  summarize(con_prob = mean(logit_prob))
```

	buytabw	con_prob
1	0	0.1353981
2	1	0.3311697

$$\text{Prob}(\text{response}|\text{x}, \text{W} = 1) = 0.3312$$
$$\text{Prob}(\text{response}|\text{x}, \text{W} = 0) = 0.1354$$

Predict incremental volume:

$$\Delta\text{Prob} = \text{Prob}(\text{response}|\text{x}, \text{W} = 1) - \text{Prob}(\text{response}|\text{x}, \text{W} = 0) = 0.3312 - 0.1354 = 0.1958$$

Causal effect of targeting vs not targeting

Attribute incremental volume to targeting effort:

$$\text{E(ROI)} = \frac{\Delta\text{Prob} \times \text{margin} - \text{cost of contact}}{cost\ of\ contact}$$

$$\text{E(ROI)} = \frac{0.1958 \times 26.9 - 1.4}{1.4} = 2.7616$$

For Neural Networks

```
# NN conditional prob
NN_prob_df <- valid_df %>%
  group_by(buytabw) %>%
  summarize(con_prob = mean(NN_prob[,2]))
```

	buytabw	con_prob
1	0	0.1352813
2	1	0.3322006

$$\text{Prob}(\text{response}|\text{x}, \text{W} = 1) = 0.3322$$
$$\text{Prob}(\text{response}|\text{x}, \text{W} = 0) = 0.1353$$

Predict incremental volume:

$$\Delta\text{Prob} = \text{Prob}(\text{response}|\text{x}, \text{W} = 1) - \text{Prob}(\text{response}|\text{x}, \text{W} = 0) = 0.3322 - 0.1353 = 0.1969$$

Causal effect of targeting vs not targeting

Attribute incremental volume to targeting effort:

$$\text{E(ROI)} = \frac{\Delta\text{Prob} \times \text{margin} - \text{cost of contact}}{cost\ of\ contact}$$

$$\text{E(ROI)} = \frac{0.1969 \times 26.9 - 1.4}{1.4} = 2.7837$$

The expected ROI of **neural network** is higher, neural network should be used to focus marketing efforts on customers.

What is the percent improvement in profits from your recommended strategy relative to a mass mailing strategy?

Neural network improve 2.7837% profits relative to a mass mailing strategy.