Qiutong Shi HW 2

BU.450.760.K1 Fri, 8:30-11:30am

1.a Estimate the first listed specifications using churn indicator as the outcome and implementing f() as the logistic model.

```
> glm.fit1 = glm(churn~tenure+rating+partysize+urban+menu+frequency, family=binomial,data=ds[idx==1,])
> summary(glm.fit1)

Call:
glm(formula = churn ~ tenure + rating + partysize + urban + menu +
    frequency, family = binomial, data = ds[idx == 1, ])

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1377  -0.5587  -0.3117  -0.1440   3.2830

Coefficients:
                       Estimate Std. Error z value Pr(>|z|)
(Intercept)           -0.392650   0.110388  -3.557 0.000375 ***
tenure                -0.050335   0.002345 -21.462  < 2e-16 ***
rating                -0.988663   0.024415 -40.494  < 2e-16 ***
partysize              0.480624   0.021231  22.637  < 2e-16 ***
urban                 -0.772587   0.049817 -15.508  < 2e-16 ***
menuethnic             0.876333   0.056412  15.534  < 2e-16 ***
menuhealthy            0.126405   0.077963   1.621 0.104946
frequencyonce-a-week   0.078563   0.073610   1.067 0.285843
frequencytwice-a-week  0.065966   0.060064   1.098 0.272092
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 14185  on 15641  degrees of freedom
Residual deviance: 10656  on 15633  degrees of freedom
AIC: 10674

Number of Fisher Scoring iterations: 6
```

The first model uses tenure, rating, partysize, urban, menu, and frequency to predict churn rate. Note that menu and frequency are factors pre-handled in the original code.

The training set is the randomly selected 70% of the Blue Apron data set, and will be used in this analysis for all regressions.

1.a Estimate the second listed specifications using churn indicator as the outcome and implementing f() as the logistic model.

The second model uses tenure, rating, partysize, urban, menu, frequency, rating*partysize, rating*urban,partysize*urban, urban*tenure to predict churn rate.

“*” refers to the interaction term between the two variables.

```
> glm.fit2 = glm(churn~tenure+rating+partysize+urban+menu+frequency+rating*partysize+rating*urban+
+                   partysize*urban+urban*tenure, family=binomial,data=ds[idx==1,])
> summary(glm.fit2)

Call:
glm(formula = churn ~ tenure + rating + partysize + urban + menu +
    frequency + rating * partysize + rating * urban + partysize *
    urban + urban * tenure, family = binomial, data = ds[idx ==
    1, ])

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9877  -0.5571  -0.2982  -0.1149   3.2060

Coefficients:
                       Estimate Std. Error z value Pr(>|z|)
(Intercept)            1.540706   0.255247   6.036 1.58e-09 ***
tenure                -0.078102   0.003885 -20.105  < 2e-16 ***
rating                -1.523450   0.096378 -15.807  < 2e-16 ***
partysize              0.070700   0.057629   1.227   0.2199
urban                 -2.281352   0.211483 -10.787  < 2e-16 ***
menuethnic             0.896719   0.056996  15.733  < 2e-16 ***
menuhealthy            0.127131   0.078322   1.623   0.1045
frequencyonce-a-week   0.085123   0.074030   1.150   0.2502
frequencytwice-a-week  0.072116   0.060478   1.192   0.2331
rating:partysize       0.127368   0.020741   6.141 8.21e-10 ***
rating:urban          -0.093344   0.049882  -1.871   0.0613 .
partysize:urban        0.279509   0.044138   6.333 2.41e-10 ***
tenure:urban           0.052169   0.004868  10.717  < 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: 14185  on 15641  degrees of freedom
Residual deviance: 10454  on 15629  degrees of freedom
AIC: 10480

Number of Fisher Scoring iterations: 6
```

1b. Select a model based on predictive performance criteria.

The predictive performance criteria being chosen is true positive rate, ROC and AUC. To calculate true positive rate, we set a threshold of 0.5 for predicting all dataset. Probability > 0.5 will be classified as positive, otherwise negative. Then we construct tables to compare the predicted positive against the actual positive in both the training and testing data set. After that, we call the confusion matrix function to look at the accuracy of the model of in-sample and out-sample data set.

Lastly, we create the graph of ROC and AUC to better understand the performance of each model.

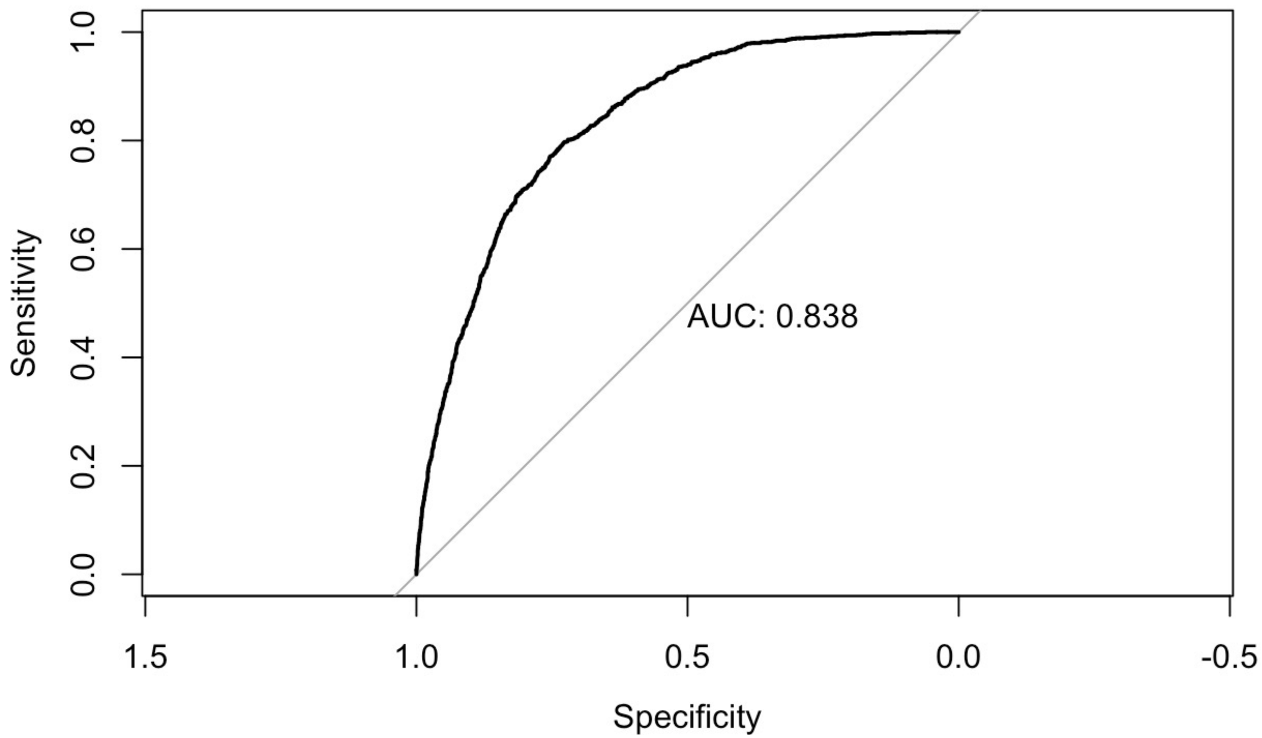
1b. Select a model based on predictive performance criteria.

```
> confusionMatrix(cm1_is, positive = "Response")
Confusion Matrix and Statistics

             actual
predicted      No response Response
  No response        12553     1863
  Response             454      772

               Accuracy : 0.8519
                 95% CI : (0.8462, 0.8574)
    No Information Rate : 0.8315
    P-Value [Acc > NIR] : 2.685e-12

                  Kappa : 0.328

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

            Sensitivity : 0.29298
            Specificity : 0.96510
         Pos Pred Value : 0.62969
         Neg Pred Value : 0.87077
             Prevalence : 0.16846
         Detection Rate : 0.04935
   Detection Prevalence : 0.07838
      Balanced Accuracy : 0.62904

       'Positive' Class : Response
```

```
> confusionMatrix(cm1_os, positive = "Response")
Confusion Matrix and Statistics

             actual
predicted      No response Response
  No response         5380      819
  Response             233      321

               Accuracy : 0.8442
                 95% CI : (0.8353, 0.8528)
    No Information Rate : 0.8312
    P-Value [Acc > NIR] : 0.002059

                  Kappa : 0.3019

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

            Sensitivity : 0.28158
            Specificity : 0.95849
         Pos Pred Value : 0.57942
         Neg Pred Value : 0.86788
             Prevalence : 0.16881
         Detection Rate : 0.04753
   Detection Prevalence : 0.08204
      Balanced Accuracy : 0.62003

       'Positive' Class : Response
```

The screen shot on the left shows us the in-sample accuracy of model 1 is 85.19%; the screen shot on the right shows us the out-sample accuracy of model 1 is 84.42%.

1b. Select a model based on predictive performance criteria.



According to the graph of ROC and AUC, the first regression model has a AUC of 0.838

```
Call:
roc.formula(formula = ds$churn[idx == 2] ~ glm1.probs[idx ==     2], plot = TRUE, print.auc = TRUE)

Data: glm1.probs[idx == 2] in 5613 controls (ds$churn[idx == 2] 0) < 1140 cases (ds$churn[idx == 2] 1).
Area under the curve: 0.8375
```

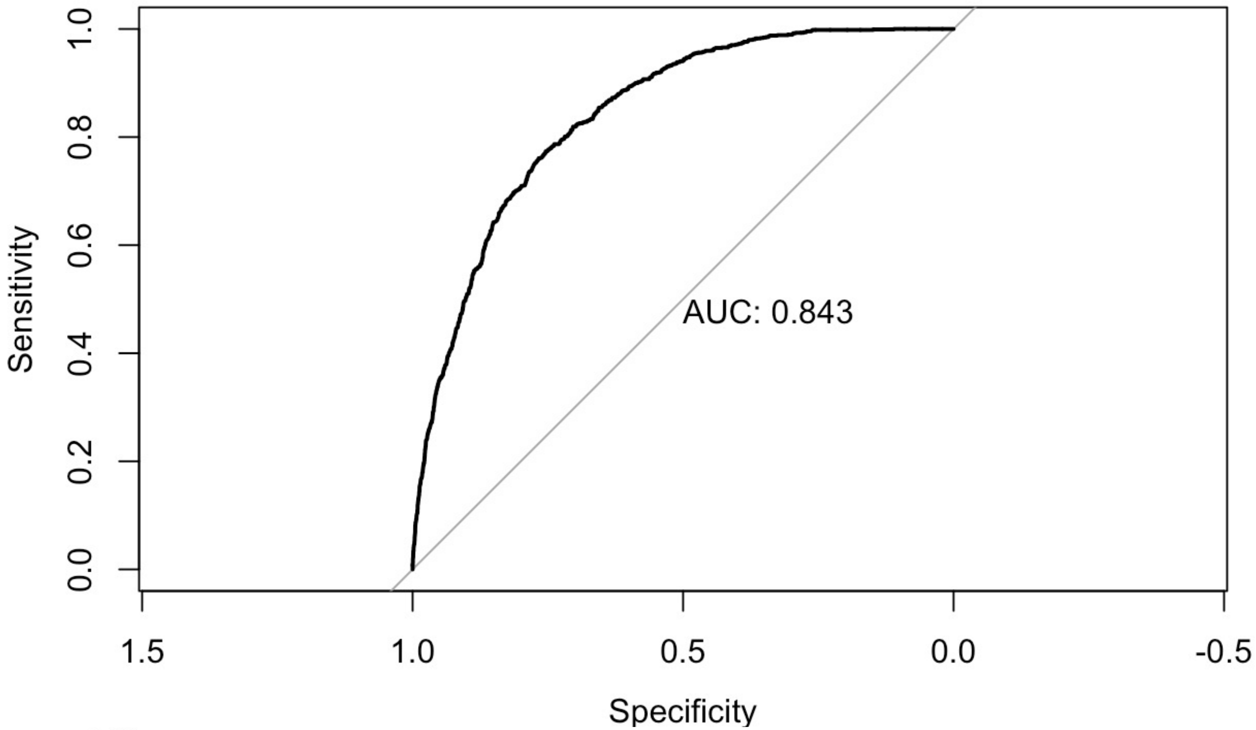
1b. Select a model based on predictive performance criteria.

```
> confusionMatrix(cm2_is, positive = "Response")
Confusion Matrix and Statistics

             actual
predicted      No response Response
  No response        12556     1817
  Response             451      818

               Accuracy : 0.855
                 95% CI : (0.8494, 0.8605)
    No Information Rate : 0.8315
    P-Value [Acc > NIR] : 7.203e-16

                  Kappa : 0.3476

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

            Sensitivity : 0.31044
            Specificity : 0.96533
         Pos Pred Value : 0.64460
         Neg Pred Value : 0.87358
             Prevalence : 0.16846
         Detection Rate : 0.05230
   Detection Prevalence : 0.08113
      Balanced Accuracy : 0.63788

       'Positive' Class : Response
```

```
> confusionMatrix(cm2_os, positive = "Response")
Confusion Matrix and Statistics

             actual
predicted      No response Response
  No response         5393      798
  Response             220      342

               Accuracy : 0.8493
                 95% CI : (0.8405, 0.8577)
    No Information Rate : 0.8312
    P-Value [Acc > NIR] : 3.138e-05

                  Kappa : 0.3268

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

            Sensitivity : 0.30000
            Specificity : 0.96081
         Pos Pred Value : 0.60854
         Neg Pred Value : 0.87110
             Prevalence : 0.16881
         Detection Rate : 0.05064
   Detection Prevalence : 0.08322
      Balanced Accuracy : 0.63040

       'Positive' Class : Response
```

The screen shot on the left shows us the in-sample accuracy of model 2 is 85.5%; the screen shot on the right shows us the out-sample accuracy if model 3 is 84.93%.

1b. Select a model based on predictive performance criteria.



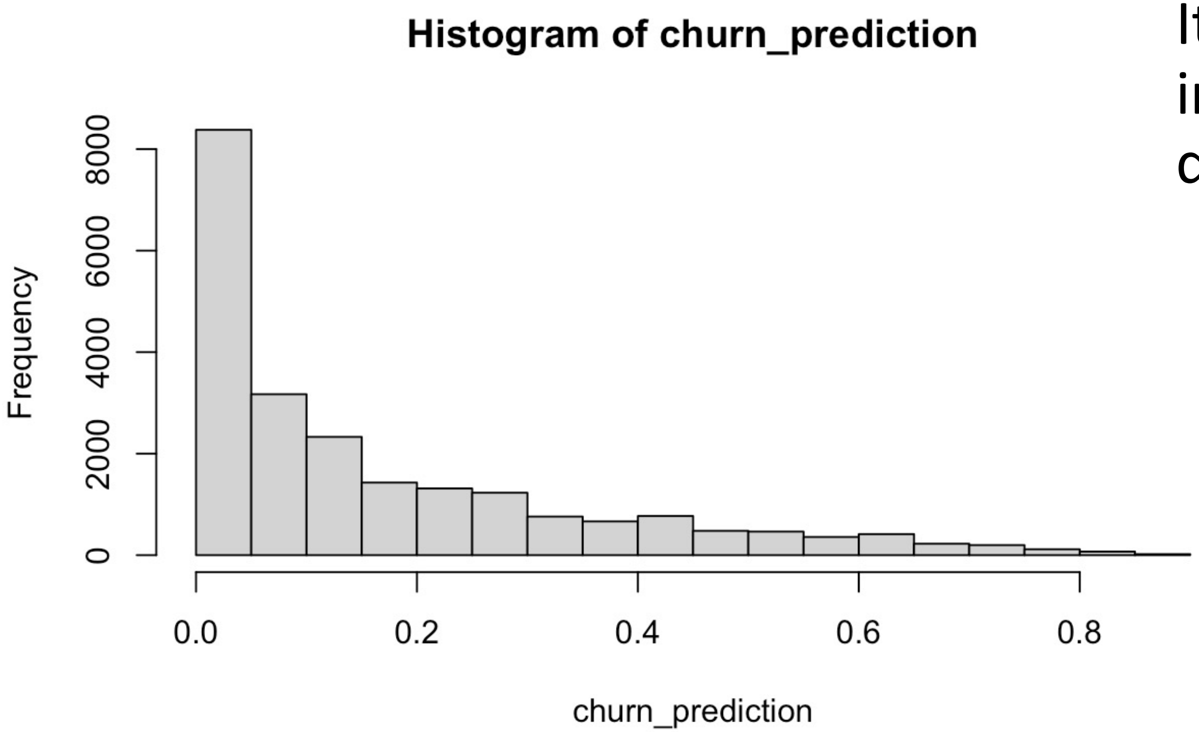
According to the graph of ROC and AUC, the first regression model has a AUC of 0.843

```
Call:
roc.formula(formula = ds$churn[idx == 2] ~ glm2.probs[idx ==     2], plot = TRUE, print.auc = TRUE)

Data: glm2.probs[idx == 2] in 5613 controls (ds$churn[idx == 2] 0) < 1140 cases (ds$churn[idx == 2] 1).
Area under the curve: 0.8425
```

1b. Select a model based on predictive performance criteria.

By the previous comparisons, we could see that model 2 has a slightly better out-sample accuracy, and also a slightly larger AUC. For this reason, model 2 is selected.

1c. Use the selected model to predict churn probabilities for every customer in the sample.



It seems that as churn rate increases, frequency of meal delivery decreases

2a. Estimate the two listed models using MonthlyAddons as the outcome and implementing f() as linear regression.

```
> glm.fit3 = glm(monthlyaddons~tenure+rating+partysize+urban+menu+frequency, family="gaussian",data=ds[id
x==1,])
> summary(glm.fit3)

Call:
glm(formula = monthlyaddons ~ tenure + rating + partysize + urban +
    menu + frequency, family = "gaussian", data = ds[idx == 1,
    ])

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-32.586  -12.622   -4.134   12.187   53.334

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)           -8.49052    0.65658 -12.931  < 2e-16 ***
tenure                -0.01775    0.01023  -1.734 0.082952 .
rating                 6.82303    0.11623  58.700  < 2e-16 ***
partysize              0.89476    0.11713   7.639 2.32e-14 ***
urban                  0.95024    0.28749   3.305 0.000951 ***
menuethnic            -0.30368    0.34928  -0.869 0.384615
menuhealthy           -0.62450    0.44925  -1.390 0.164518
frequencyonce-a-week   0.67788    0.43688   1.552 0.120770
frequencytwice-a-week  0.45660    0.35073   1.302 0.192986
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 320.2316)

    Null deviance: 6130448  on 15641  degrees of freedom
Residual deviance: 5006181  on 15633  degrees of freedom
AIC: 134640

Number of Fisher Scoring iterations: 2
```

Model 3 uses tenure, rating, partysize, urban, menu, and frequency to predict monthlyaddons. Note that menu and frequency are factors pre-handled in the original code, and the linear regressions would be using the same testing and training data set.

2a. Estimate the two listed models using MonthlyAddons as the outcome and implementing f() as linear regression.

```
> glm.fit4 = glm(monthlyaddons~tenure+rating+partysize+urban+menu+frequency+rating*partysize+rating*urban
+
+                  partysize*urban+urban*tenure, family="gaussian",data=ds[idx==1,])
> summary(glm.fit4)

Call:
glm(formula = monthlyaddons ~ tenure + rating + partysize + urban +
    menu + frequency + rating * partysize + rating * urban +
    partysize * urban + urban * tenure, family = "gaussian",
    data = ds[idx == 1, ])

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-34.565  -12.665   -3.266   12.184   54.237

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           -3.606484   1.482733  -2.432 0.015013 *
tenure                -0.008164   0.015359  -0.532 0.595039
rating                 5.169406   0.418791  12.344  < 2e-16 ***
partysize             -0.137279   0.336741  -0.408 0.683522
urban                 -0.483492   1.244731  -0.388 0.697702
menuethnic            -0.295312   0.349138  -0.846 0.397660
menuhealthy           -0.630419   0.449054  -1.404 0.160374
frequencyonce-a-week   0.706338   0.436771   1.617 0.105859
frequencytwice-a-week  0.452711   0.350627   1.291 0.196671
rating:partysize       0.339967   0.095199   3.571 0.000357 ***
rating:urban           0.525917   0.233373   2.254 0.024239 *
partysize:urban        0.024638   0.235368   0.105 0.916634
tenure:urban          -0.017522   0.020591  -0.851 0.394827
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 319.9305)

    Null deviance: 6130448  on 15641  degrees of freedom
Residual deviance: 5000194  on 15629  degrees of freedom
AIC: 134630

Number of Fisher Scoring iterations: 2
```

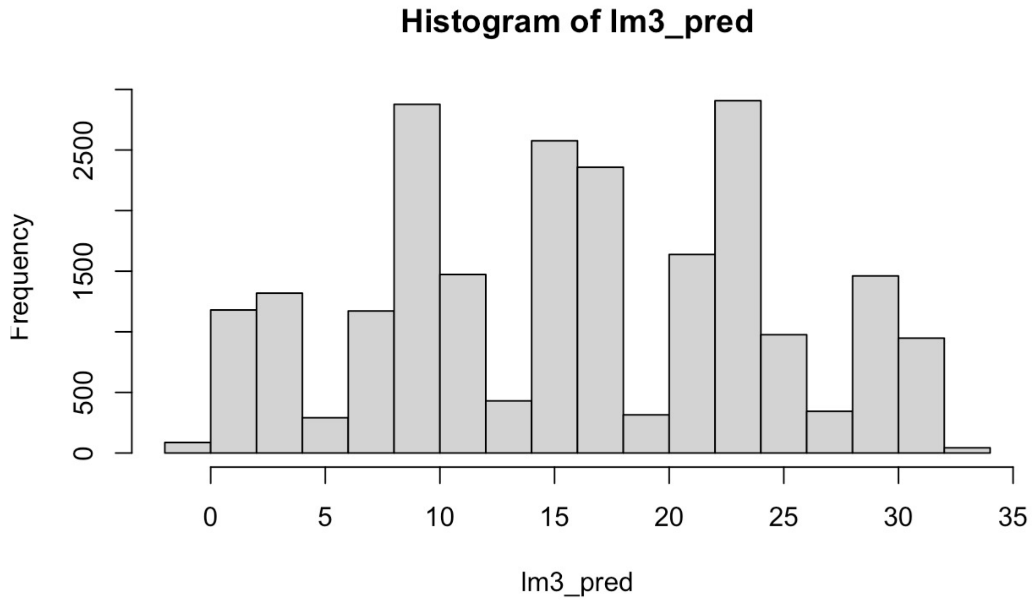
Model 4 uses tenure, rating, partysize, urban, menu, frequency, rating*partysize, rating*urban,partysize*urban, urban*tenure to predict churn rate.

“*” refers to the interaction term between the two variables.

2b. Select a model based on predictive performance criteria.

The predictive performance criteria selected for linear regression models is RMSE. We first use the trained model to predict monthlyaddons in every observation of the blue apron data set, then compare the predictions with the actual in-sample and out-sample monthlyaddons. Note that rows with empty churn rate is eliminated as predicting the amonunt spent on spend add-ons for observatioin without churn response makes the analysis lose its real-world importance for marketing decisions.

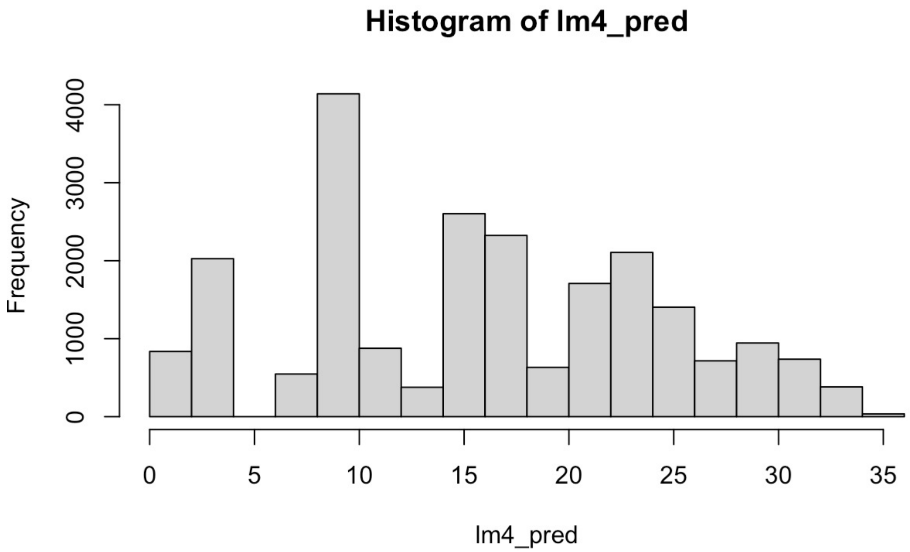
2b. Select a model based on predictive performance criteria.



Model 3 has an in-sample MRSE of 17.92 and out-sample MRSE of 17.80.

```
> postResample(pred = lm3_pred[idx==1 & is.na(ds$churn)==FALSE], obs = ds$churn[idx==1 & is.na(ds$churn)=
=FALSE])
      RMSE   Rsquared        MAE
17.9177491  0.1020603 15.7257600
> postResample(pred = lm3_pred[idx==2 & is.na(ds$churn)==FALSE], obs = ds$churn[idx==2 & is.na(ds$churn)=
=FALSE])
       RMSE    Rsquared         MAE
17.80063514  0.09797234 15.63796813
```

2b. Select a model based on predictive performance criteria.



Model 4 has an in-sample MRSE of 17.93 and out-sample MRSE of 17.81.

```
> postResample(pred = lm4_pred[idx==1 & is.na(ds$churn)==FALSE], obs = ds$churn[idx==1 & is.na(ds$churn)=
=FALSE])
     RMSE   Rsquared        MAE 
17.9289928  0.1035628 15.7163934 
> postResample(pred = lm4_pred[idx==2 & is.na(ds$churn)==FALSE], obs = ds$churn[idx==2 & is.na(ds$churn)=
=FALSE])
     RMSE   Rsquared        MAE 
17.8100437  0.1008991 15.6254420 
```

2b. Select a model based on predictive performance criteria.

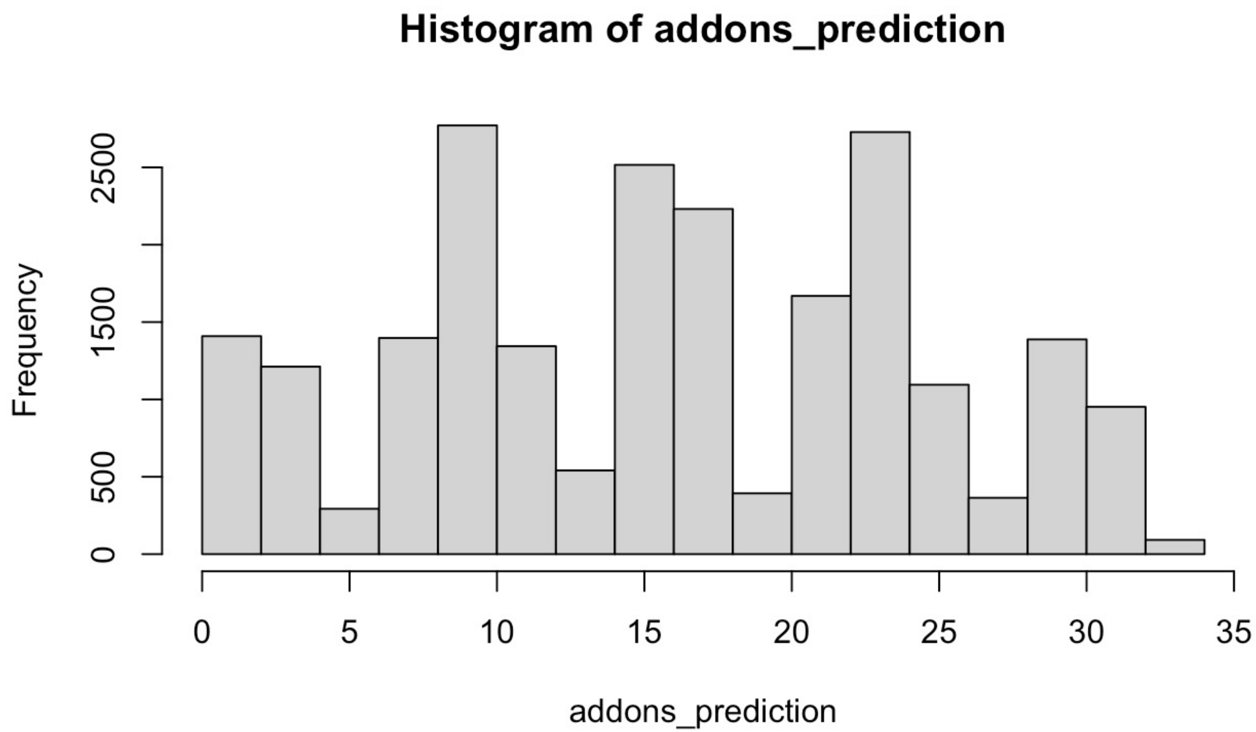
Since model 3 has a slilghtly lower MRSE than model 4, model 3 is selected as the final model for linear regression.

2c. Use the selected model to predict MonthlyAddons for every customer in the sample.

From 2b, we select model 3. Since monthlyaddons have unit in $, we need to replace all negative values in the predicted monthlyaddons vector with 0, so that this vector can be added to the csv file as a column that makes sense in real world for our calculation. Then we use an if to check if the condition “all values in addons_prediction <0” is correct.

```
> #since monthlyaddons cannot be negative, replacing all negative monthlyaddons with "0"
> addons_prediction[addons_prediction<0]<-0
> if (all(addons_prediction >= 0)){
+   print("All values are non-negatives!")
+ }
[1] "All values are non-negatives!"
```

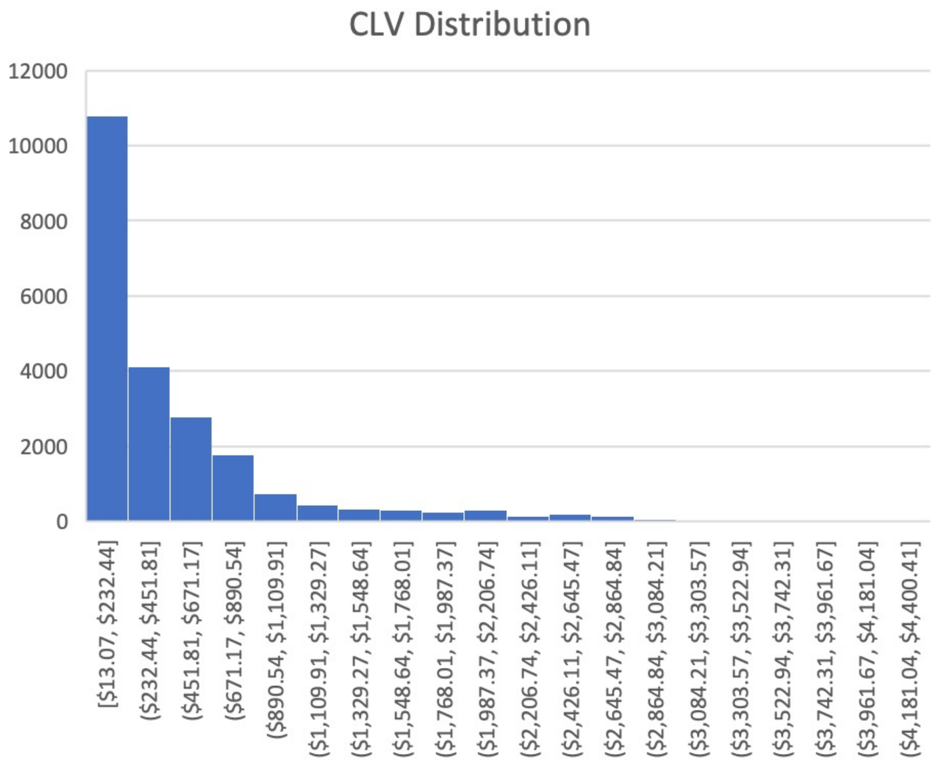
2c. Use the selected model to predict MonthlyAddons for every customer in the sample.



3. Export the full dataset to a csv file.

```
> #export file
> export_df = cbind(ds,churn_prediction,addons_prediction)
> write.csv(export_df,"blueapron_predicted.csv")
```

4a. Compute baseline CLV values for each customer in the initial scenario



We could see from the histogram that as CLV increases, frequency decreases, meaning that a majority of the customers in the data set has a CLV between $13.07 to $232.44.

initial CLV		
multiplier	monthly net contribution	baseline CLV
21.326983	$68.74	$1,466.01
6.8145445	$73.88	$503.45
25.030697	$24.30	$608.16
5.6076932	$53.89	$302.19
26.36497	$18.98	$500.32
7.0888744	$12.19	$86.38
12.182129	$47.04	$573.04

4b. Determine the optimal targeting policy.

To find the group of customers to target, we need to calculate the net profitability each customer is expected to bring after the $20 expenditure. Mathematically, this can be obtained by substracting $20 from the original net monthly contribution calculated for part 4a. We classify a customer as worth targeting if profitability > 1. Eventually, we want to target 11610 customers who will cost $232,200.

optimal targeting(unlimited budget)			
monthly net profitability after targeting	target?	summary	
$48.74	1	#targeted	11610
$53.88	1	total expenditure	$232,200.00
$4.30	1		
$33.89	1		
-$1.02	0		
-$7.81	0		
$27.04	1		

4c. Compute the total financial gains/losses derived from implementing the campaign as the before/after difference between the total CLV values in the entire portfolio of customers.

The CLV of a customer would remain the same if the customer is not targeted. What matters is the targeted customers, whose churn rate is decreased by 0.01 from the $20 campaign. Meanwhile, the monthly net contribution of targeted customers would also remain the same. We use the conditional statement if to update churn,multiplier, and CLV, if targeted, and to use the previous values if not targeted.

after-targeting churn	after-targeting multiplier
0.017437711	26.96220373
0.119331582	7.302204274
0.010358107	33.16645642

4c. Compute the total financial gains/losses derived from implementing the campaign as the before/after difference between the total CLV values in the entire portfolio of customers.

Eventually, we calculate between the baseline CLV and the after-target CLV. Untargeted customer will have a 0 in this term. Then we add the difference up and observe a financial gain of $2,926,335.39 by summing up the differences.

CLV comparison				
after-targeting churn	after-targeting multiplier	after-targeting CLV	difference	loss/gain?
0.017437711	26.96220373	$1,853.38	$387.36	$2,926,335.39
0.119331582	7.302204274	$539.48	$36.03	
0.010358107	33.16645642	$805.84	$197.67	