USE CASE STUDY REPORT

Group No.: Group 22

Student Names: Harikrishnan Shajil and Saumya Vora

Executive Summary:

The goal of the study was to devise a supervised machine learning algorithm based on prescriptive analytics to predict whether a customer will churn or not.
Telecommunication service providers usually track their customers based on their demographics (area), services signed up such as internet, TV streaming etc. and last but not the least their account charges/billing. These are used to predict behavior of different customers and could there help in deriving insights about them. In terms of data collection, we had extended our search in the Kaggle repositories.

These steps have been taken in the Case Study respectively:

1) Data Preprocessing to eliminate redundant data
2) Data Reduction & Transformation to generate information rich structured data.
3) Exploratory Data Analysis to derive patterns and insights.
4) Data Mining using Machine Learning Algorithms
5) Model Evaluation to compare and pick the best possible classifying model.
6) Interpretation and Course of Action

Data mining techniques used were K- nearest neighbor, Naïve Bayes, CART, Logistic Regression, Artificial Neural Network, Linear Discriminant Analysis, Support Vector Machine. Lift chart, Decile wise Lift chart, Confusion matrix and ROC curve was used to obtain performance measure for each model. Random Forest was observed to the best classifying model for the given dataset.
The Random Forest Algorithm provides us with the best accuracy which is 78.35%.

After classifying customers accurately, the future course of action for customers who are most likely to churn will be to provide them with Priority mails, offers on mobile plans, Discount on calling rates etc.

I. Background and Introduction

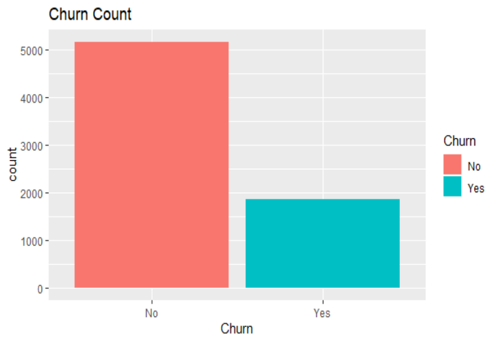
Telecom provider's main concern is "Whether the customer will churn (discontinue phone service)?" Based on the massive dataset, it is extremely difficult to predict individual customer behavior and provide a solution for so by manual statistical computations.
We have taken the initiative to provide a solution to this problem by possibly predicting customer behavior and thereby classifying their churn outcomes accurately using Prescriptive Analytics.
After classifying customers accurately, the future course of action for customers who are most likely to churn will be to provide them with Priority mails, offers on mobile plans, Discount on calling rates etc.

II. Data Exploration and Visualization

Distribution plot

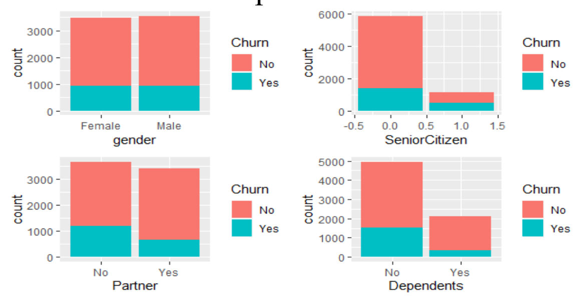
preprocessing part, we are going to remove the missing values rows.

1.Total Churn Count



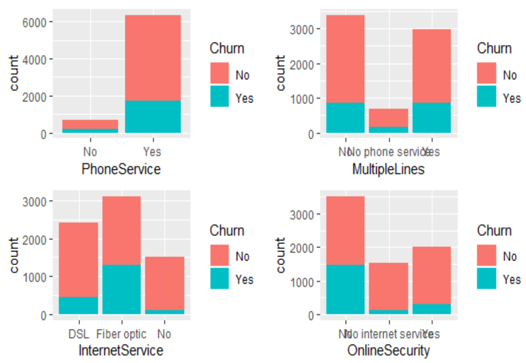
Total Number of Customers changing their telecom company. Nearly 2000 people are churning which is approximately 1/3 of the total customers which is a serious threat for company.

2.Gender, Senior Citizen, Partner, Dependents



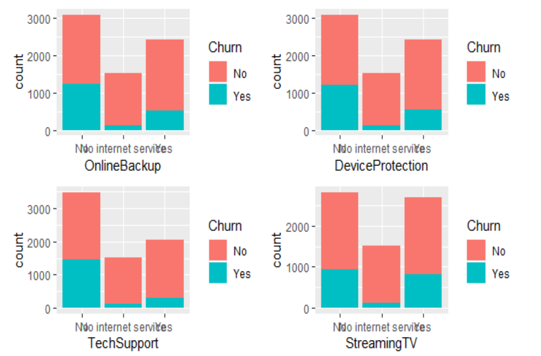
Churning based on the Gender, Senior Citizen, Partner and Dependents. Both the gender shows equal churning. Senior Citizens ,Having Partner and dependents shows less churning.

3.Phoneservice, Multiple lines, Online Security, Internet Service

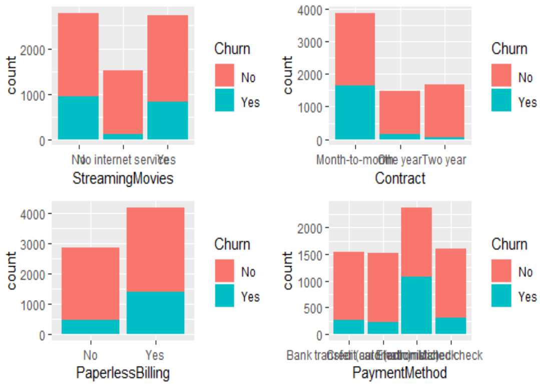


Customers with No internet service and No Online Security are less likely to churn.

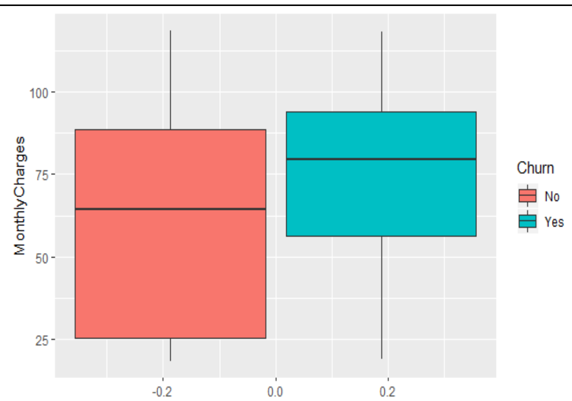
4.Online Backup, Device Protection, Tech Support, Streaming TV



Customers who have Online Backup, Device Protection,Tech Support are less likely to churn.



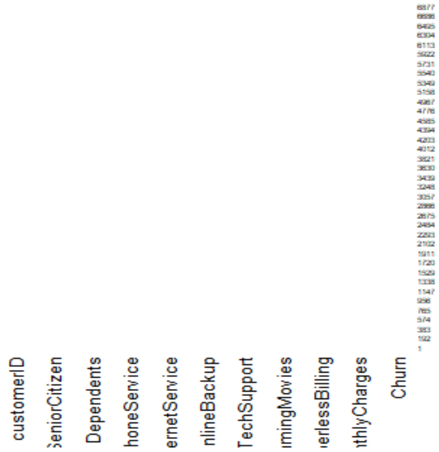
Customers who have opted for one and two year contract ,No paperless billing and are not churning much.



Based on Monthly Charges paid by customers if the range is between 25 to 85 the customers are not churning.

Missing Values

Heatmap clearly shows that really a smaller number of missing data. In the data



Rescaling

It is only done for Monthly Charges and Total Charges in Artificial Neural Networks.

III. Data Preparation and Preprocessing

Data Summary

The Original Dataset Summary

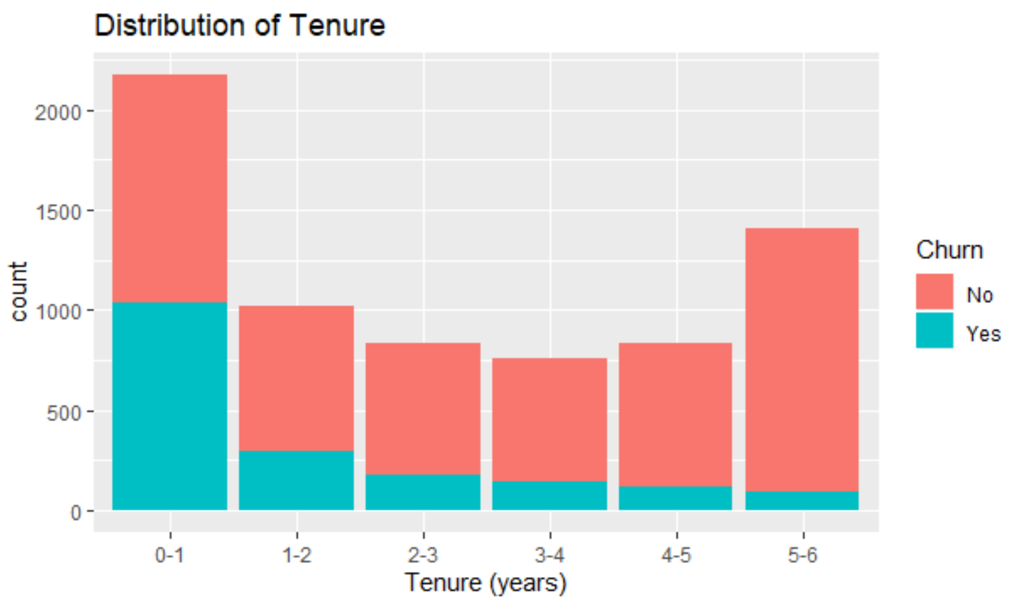
```
> summary(churn)
      customerID      gender     SeniorCitizen      Partner    Dependents
 0002-ORFBO:   1   Female:3488   Min.   :0.0000   No :3641   No :4933
 0003-MKNFE:   1   Male  :3555   1st Qu.:0.0000   Yes:3402   Yes:2110
 0004-TLHLJ:   1                 Median :0.0000
 0011-IGKFF:   1                 Mean   :0.1621
 0013-EXCHZ:   1                 3rd Qu.:0.0000
 0013-MHZWF:   1                 Max.   :1.0000
 (Other)   :7037
     tenure     PhoneService          MultipleLines     InternetService
 Min.   : 0.00   No : 682     No              :3390   DSL        :2421
 1st Qu.: 9.00   Yes:6361     No phone service: 682   Fiber optic:3096
 Median :29.00                Yes             :2971   No         :1526
 Mean   :32.37
 3rd Qu.:55.00
 Max.   :72.00
             OnlineSecurity              OnlineBackup            DeviceProtection
 No                 :3498   No                 :3088   No                 :3095
 No internet service:1526   No internet service:1526   No internet service:1526
 Yes                :2019   Yes                :2429   Yes                :2422


              TechSupport               StreamingTV            StreamingMovies
 No                 :3473   No                 :2810   No                 :2785
 No internet service:1526   No internet service:1526   No internet service:1526
 Yes                :2044   Yes                :2707   Yes                :2732


           Contract    PaperlessBilling                 PaymentMethod
 Month-to-month:3875   No :2872         Bank transfer (automatic):1544
 One year      :1473   Yes:4171         Credit card (automatic)  :1522
 Two year      :1695                    Electronic check         :2365

 MonthlyCharges     TotalCharges     Churn
 Min.   : 18.25   Min.   :  18.8   No :5174
 1st Qu.: 35.50   1st Qu.: 401.4   Yes:1869
 Median : 70.35   Median :1397.5
 Mean   : 64.76   Mean   :2283.3
 3rd Qu.: 89.85   3rd Qu.:3794.7
 Max.   :118.75   Max.   :8684.8
                  NA's   :11
>
```

Dimension Reduction
Variable Converting

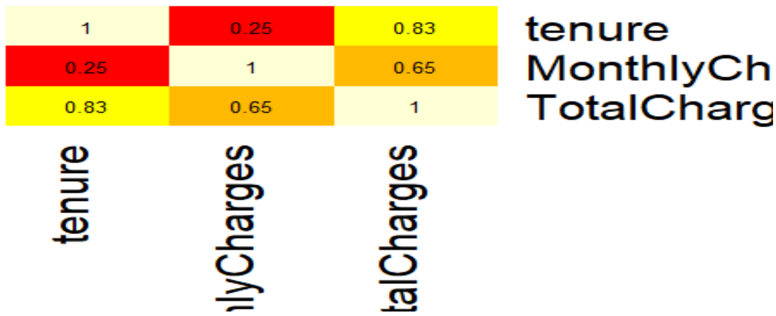
(1) Removing 11 missing values rows.
(2) Applied the function to convert all the rows containing values“ No phone service” to “No”. Other function to convert all the rows containing values “No internet service” to “No”. These functions are applied to have uniformity in the all categorical values of the columns “PhoneService” and “MultipleLines”.
(3) Converting Months to Years for the “Tenure” Variable.
(4) Binned Tenure variable from 0-1,1-2,2-3,3-4,4-5,5-6.
(5) Standardizing columns Monthly Charges and Total Charge
(6) Creating Dummy variables for “Tenure”, ”Monthly Charges” and “Total Charges”.

Visualization after Preprocessing

```
> summary(churn)
      customerID       gender      SeniorCitizen Partner    Dependents tenure
 0002-ORFBO:   1   Female:3483   0:5890        No :3639   No :4933   0-1:2175
 0003-MKNFE:   1   Male  :3549   1:1142        Yes:3393   Yes:2099   1-2:1024
 0004-TLHLJ:   1                                                      2-3: 832
 0011-IGKFF:   1                                                      3-4: 762
 0013-EXCHZ:   1                                                      4-5: 832
 0013-MHZWF:   1                                                      5-6:1407
 (Other)   :7026
 PhoneService MultipleLines     InternetService OnlineSecurity OnlineBackup
 No : 680     No :4065      DSL        :2416   No :5017       No :4607
 Yes:6352     Yes:2967      Fiber optic:3096   Yes:2015       Yes:2425
                            No         :1520


 DeviceProtection TechSupport StreamingTV StreamingMovies           Contract
 No :4614         No :4992    No :4329    No :4301        Month-to-month:3875
 Yes:2418         Yes:2040    Yes:2703    Yes:2731        One year      :1472
                                                          Two year      :1685


 PaperlessBilling                   PaymentMethod  MonthlyCharges
 No :2864         Bank transfer (automatic):1542  Min.   :-1.5472
 Yes:4168         Credit card (automatic)  :1521  1st Qu.:-0.9709
                  Electronic check         :2365  Median : 0.1845
                  Mailed check             :1604  Mean   : 0.0000
                                                  3rd Qu.: 0.8331
                                                  Max.   : 1.7933

  TotalCharges      Churn
 Min.   :-0.9990   No :5163
 1st Qu.:-0.8302   Yes:1869
 Median :-0.3908
```



Variable Selection
All variables except “Customer ID” are selected.
Correlation Analysis



Tenure and Monthly Charges are correlated by 0.25.
Tenure and Total Charges are correlated by 0.83
Monthly Charges and Total Charges are correlated by 0.65

Since Tenure is highly correlated with total Charges ,Tenure is binned and converted to categorical variable.

IV. Data Mining Techniques and Implementation

MODEL 1: K-Nearest Neighbors (K-NN)
KNN model was applied to the dataset using both oversampled and normally sampled data. We notice that error decreases in the validation set till K=23 and starts increasing again. Thus, the best K is 23. Error at K=23 is 0.2420429.

	train	valid
1	0.00267666	0.2850112
2	0.15310493	0.3000052
3	0.15845824	0.2696783
4	0.19111349	0.2763825
5	0.19325482	0.2661288
6	0.19432548	0.2598588
7	0.19271949	0.2585416
8	0.20396146	0.2541164
9	0.20877944	0.2514820
10	0.21466809	0.2514281
11	0.22216274	0.2463676
12	0.22269807	0.2491346
13	0.22591006	0.2463282
14	0.22483940	0.2529246
15	0.22483940	0.2536604
16	0.22430407	0.2486610
17	0.22376874	0.2438482
18	0.22805139	0.2477314
19	0.22430407	0.2444223

19	0.22430407	0.2444223
20	0.23072805	0.2458795
21	0.22591006	0.2455385
22	0.22269807	0.2433062
23	0.22430407	0.2420429
24	0.22591006	0.2464070
25	0.22912206	0.2452981
26	0.22858672	0.2448566
27	0.23019272	0.2458722
28	0.22912206	0.2426636
29	0.23233405	0.2411133
30	0.23554604	0.2441280

While comparing oversampled data with normally sampled data, accuracy is 74.33 % for normally sampled and 70.92% for oversampled data. However, sensitivity is 85.53% for oversampled and 69.20% for normally sampled. Thus giving balanced accuracy of 74.94% for over sampled and 73.17% for normally sampled.

MODEL 2: Naïve Bayes

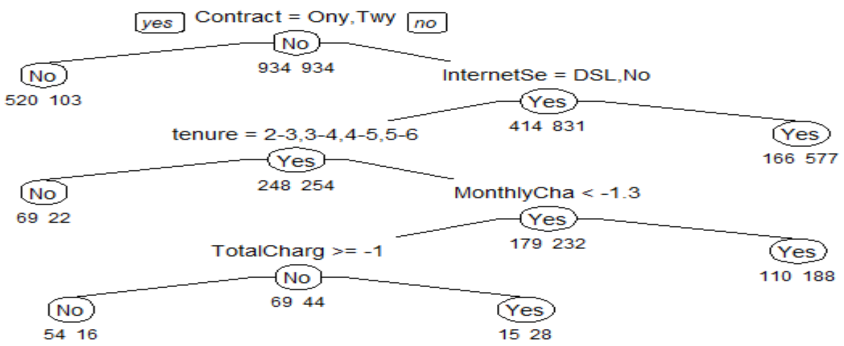
```
> NB_model1

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace, type = "class")

A-priori probabilities:
Y
 No Yes
0.5 0.5

Conditional probabilities:
     gender
Y        Female      Male
  No  0.4957173 0.5042827
  Yes 0.5021413 0.4978587

     SeniorCitizen
Y            0        1
  No  0.866167 0.133833
  Yes 0.745182 0.254818

     Partner
Y           No      Yes
  No  0.503212 0.496788
  Yes 0.643469 0.356531

     Dependents
Y            No       Yes
  No  0.6563169 0.3436831
  Yes 0.8426124 0.1573876

     tenure
Y            0-1        1-2        2-3        3-4        4-5        5-6
  No  0.21092077 0.16274090 0.12955032 0.13169165 0.14989293 0.21520343
  Yes 0.54282655 0.15310493 0.09207709 0.08458244 0.07601713 0.05139186
```

```
     PhoneService
Y             No        Yes
  No  0.10920771 0.89079229
  Yes 0.09100642 0.90899358

     MultipleLines
Y            No       Yes
  No  0.5942184 0.4057816
  Yes 0.5235546 0.4764454

     InternetService
Y            DSL Fiber optic         No
  No  0.36616702  0.35010707 0.28372591
  Yes 0.24197002  0.70877944 0.04925054

     OnlineSecurity
Y            No       Yes
  No  0.6948608 0.3051392
  Yes 0.8404711 0.1595289

     OnlineBackup
Y            No       Yes
  No  0.6627409 0.3372591
  Yes 0.7066381 0.2933619

     DeviceProtection
Y            No       Yes
  No  0.6552463 0.3447537
  Yes 0.7109208 0.2890792

     TechSupport
Y            No       Yes
  No  0.6895075 0.3104925
  Yes 0.8297645 0.1702355

     StreamingTV
Y            No       Yes
  No  0.6434690 0.3565310
  Yes 0.5642398 0.4357602
```

```
     StreamingMovies
Y            No       Yes
  No  0.6434690 0.3565310
  Yes 0.5556745 0.4443255

     Contract
Y     Month-to-month   One year   Two year
  No      0.45182013 0.24625268 0.30192719
  Yes     0.87687366 0.09743041 0.02569593

     PaperlessBilling
Y            No       Yes
  No  0.4743041 0.5256959
  Yes 0.2462527 0.7537473

     PaymentMethod
Y     Bank transfer (automatic) Credit card (automatic) Electronic check Mailed check
  No                  0.2334047               0.2537473        0.2912206    0.2216274
  Yes                 0.1509636               0.1241970        0.5738758    0.1509636

     MonthlyCharges
Y              1          2          3          4          5
  No  0.31370450 0.16809422 0.16702355 0.22376874 0.12740899
  Yes 0.09850107 0.14989293 0.20985011 0.36723769 0.17451820

     TotalCharges
Y               1           2           3           4           5           6           7           8
  No  0.338329764 0.191648822 0.104925054 0.084582441 0.070663812 0.064239829 0.048179872 0.041755889
  Yes 0.532119914 0.149892934 0.083511777 0.068522484 0.038543897 0.037473233 0.039614561 0.027837259
     TotalCharges
Y               9          10
  No  0.036402570 0.019271949
  Yes 0.019271949 0.003211991
```

While comparing oversampled data with normally sampled data, accuracy is 74.23 % for normally sampled and 74.79% for oversampled data. However, sensitivity is 78.50% for oversampled and 73.26% for normally sampled. Thus giving balanced accuracy of 75.98% for over sampled and 74.01% for normally sampled.

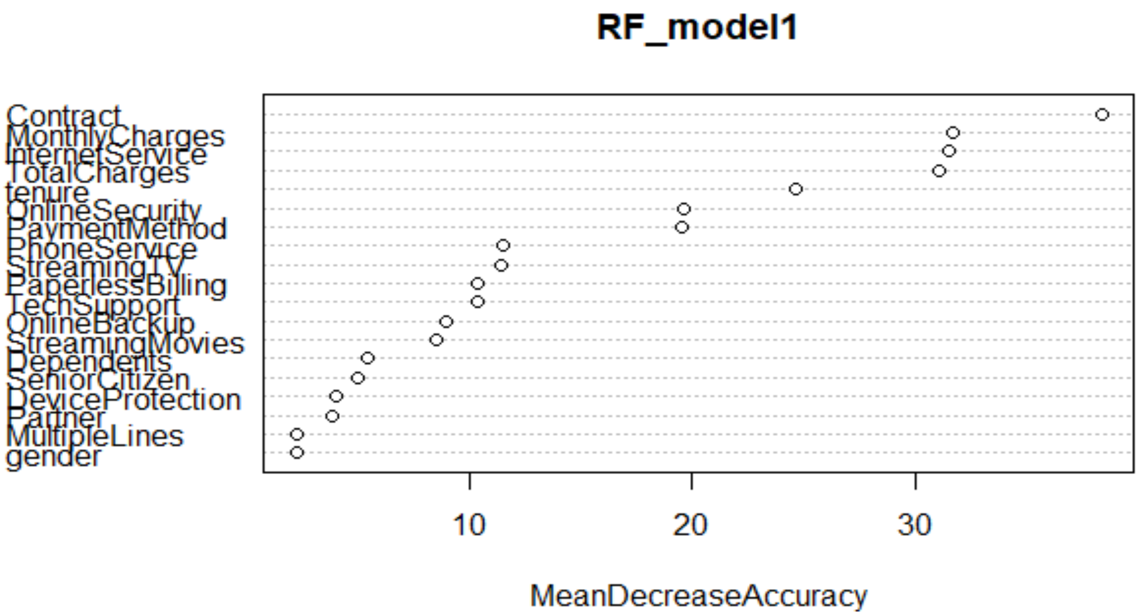
MODEL 3: CART-Random Forest-Boosted Trees
Classification Tree
CART algorithm gives the following rules:



Pruning of the classification tree does not improve the performance.

While comparing oversampled data with normally sampled data, accuracy is 76.32 % for normally sampled and 75.53% for oversampled data. However, sensitivity is 75.51% for oversampled and 60.42% for normally sampled. Thus giving balanced accuracy of 75.53% for over sampled and 72.73% for normally sampled.

Random Forest:



While comparing oversampled data with normally sampled data, accuracy is 78.04 % for normally sampled and 78.35% for oversampled data. However, sensitivity is 76.26% for oversampled and 61.07% for normally sampled. Thus giving balanced accuracy of 77.68% for over sampled and 74.21% for normally sampled.

Boosted Trees:

While comparing oversampled data with normally sampled data, accuracy is 76.74 % for normally sampled and 75.62% for oversampled data. However, sensitivity is 76.47% for oversampled and 62.12% for normally sampled. Thus giving balanced accuracy of 75.89% for over sampled and 73.44% for normally sampled.

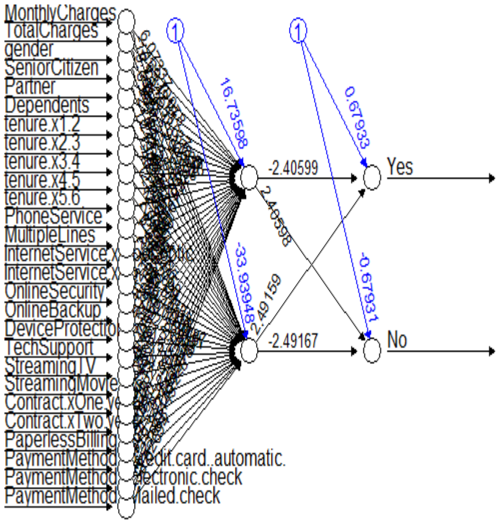
MODEL 4: Logistic Regression

```
Call:  glm(formula = Churn ~ ., family = "binomial", data = train.logit)

Coefficients:
                           (Intercept)                         MonthlyCharges
                             -0.713924                              -1.148212
                          TotalCharges                                 gender
                             -0.140751                              -0.140014
                         SeniorCitizen                                Partner
                              0.305923                              -0.134538
                            Dependents                            tenure.x1.2
                             -0.055375                              -0.593533
                           tenure.x2.3                            tenure.x3.4
                             -1.335914                              -0.640752
                           tenure.x4.5                            tenure.x5.6
                             -1.162276                              -1.539576
                          PhoneService                          MultipleLines
                             -0.003851                               0.428252
          InternetService.xFiber.optic                     InternetService.xNo
                              2.002835                              -1.773130
                        OnlineSecurity                           OnlineBackup
                             -0.377875                               0.126552
                      DeviceProtection                            TechSupport
                              0.027249                               0.174308
                           StreamingTV                        StreamingMovies
                              0.799646                               0.806395
                   Contract.xOne.year                      Contract.xTwo.year
                             -0.776452                              -1.875204
                      PaperlessBilling  PaymentMethod.xCredit.card..automatic.
                              0.211069                              -0.142355
     PaymentMethod.xElectronic.check              PaymentMethod.xMailed.check
                              0.416253                               0.041895

Degrees of Freedom: 1867 Total (i.e. Null);  1840 Residual
Null Deviance:      2590
Residual Deviance: 1752        AIC: 1808
```

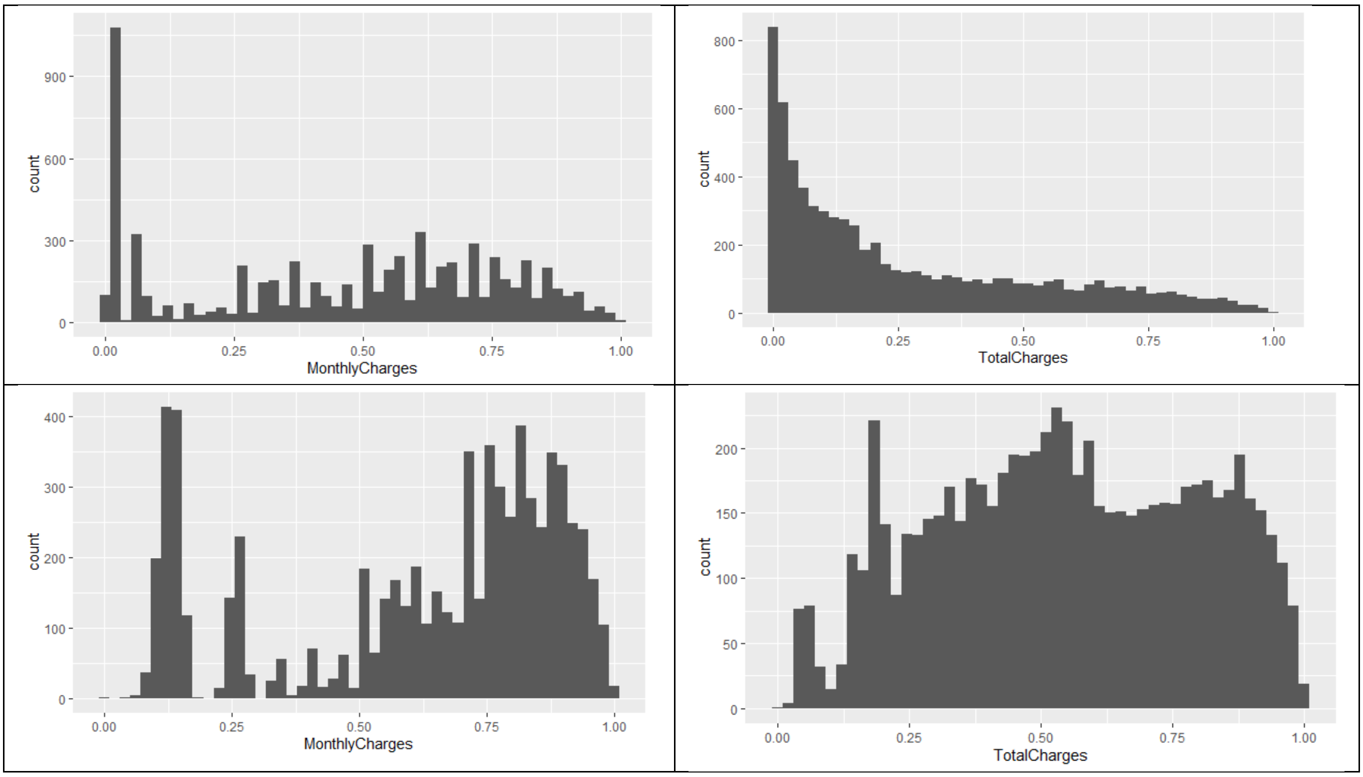
While comparing oversampled data with normally sampled data, accuracy is 76.28 % for normally sampled and 73.71% for oversampled data. However, sensitivity is 78.82% for oversampled and 63.04% for normally sampled. Thus, giving balanced accuracy of 75.34% for over sampled and 73.29% for normally sampled.

MODEL 5: Artificial Neural Network

Neural Network model with one hidden layer and two hidden nodes gives the least error in validation set.



Since monthly charges and total charges have right skewed distribution we transform them using square root and cube root respectively.



While comparing oversampled data with normally sampled data, accuracy is 76.28 % for normally sampled and 75.02% for oversampled data. However, sensitivity is 77.11% for oversampled and 62.25% for normally sampled. Thus giving balanced accuracy of 75.69% for over sampled and 73.11% for normally sampled.

MODEL 6: Linear Discriminant Analysis

```
> LDA_model1
Call:
lda(Churn ~ ., data = train.LDA)

Prior probabilities of groups:
 No Yes
0.5 0.5

Group means:
        tenure MonthlyCharges TotalCharges   gender SeniorCitizen  Partner
No   0.1904035     -0.1364461   0.09597908 1.532120      1.132762 1.512848
Yes -0.6136000      0.3150206  -0.35648717 1.487152      1.259101 1.345824
    Dependents PhoneService MultipleLines InternetService OnlineSecurity
No    1.330835     1.910064      1.414347        1.894004       1.345824
Yes   1.162741     1.904711      1.448608        1.799786       1.153105
    OnlineBackup DeviceProtection TechSupport StreamingTV StreamingMovies Contract
No      1.353319         1.349036    1.323340    1.353319        1.350107 1.898287
Yes     1.273019         1.268737    1.180942    1.431478        1.440043 1.134904
    PaperlessBilling PaymentMethod
No          1.524625      2.489293
Yes         1.733405      2.768737

Coefficients of linear discriminants:
                         LD1
tenure           -0.29243909
MonthlyCharges    1.01948429
TotalCharges     -0.29212158
gender           -0.09870775
SeniorCitizen     0.22976116
```

```
Coefficients of linear discriminants:
                         LD1
tenure           -0.29243909
MonthlyCharges    1.01948429
TotalCharges     -0.29212158
gender           -0.09870775
SeniorCitizen     0.22976116
Partner          -0.06218702
Dependents       -0.07981460
PhoneService     -1.23767424
MultipleLines     0.02712665
InternetService   0.03568572
OnlineSecurity   -0.66912553
OnlineBackup     -0.22054275
DeviceProtection -0.28876842
TechSupport      -0.24003644
StreamingTV      -0.02480172
StreamingMovies  -0.05526104
Contract         -0.49520639
PaperlessBilling  0.16558976
PaymentMethod     0.05488869
```

While comparing oversampled data with normally sampled data, accuracy is 75.63 % for normally sampled and 73.88% for oversampled data. However, sensitivity is 80.11% for oversampled and 63.30% for normally sampled. Thus giving balanced accuracy of 75.87% for over sampled and 72.84% for normally sampled.

MODEL 7: Support Vector Machine

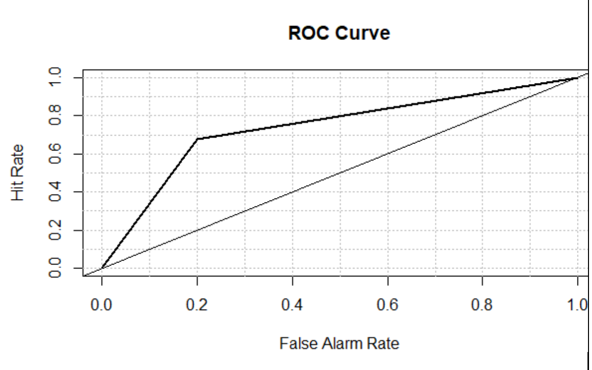
```
> SVM_model1

Call:
svm(formula = Churn ~ ., data = train.SVM)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  1100
```

While comparing oversampled data with normally sampled data, accuracy is 76.00 % for normally sampled and 75.36% for oversampled data. However, sensitivity is 78.50% for oversampled and 64.09% for normally sampled. Thus giving balanced accuracy of 76.36% for over sampled and 73.31% for normally sampled.

V. Performance Evaluation

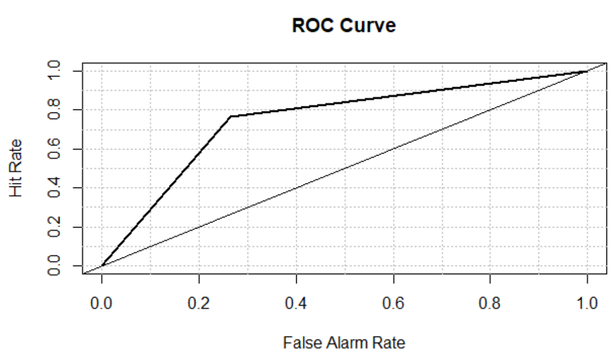
MODEL 1: K-Nearest Neighbors (K-NN)

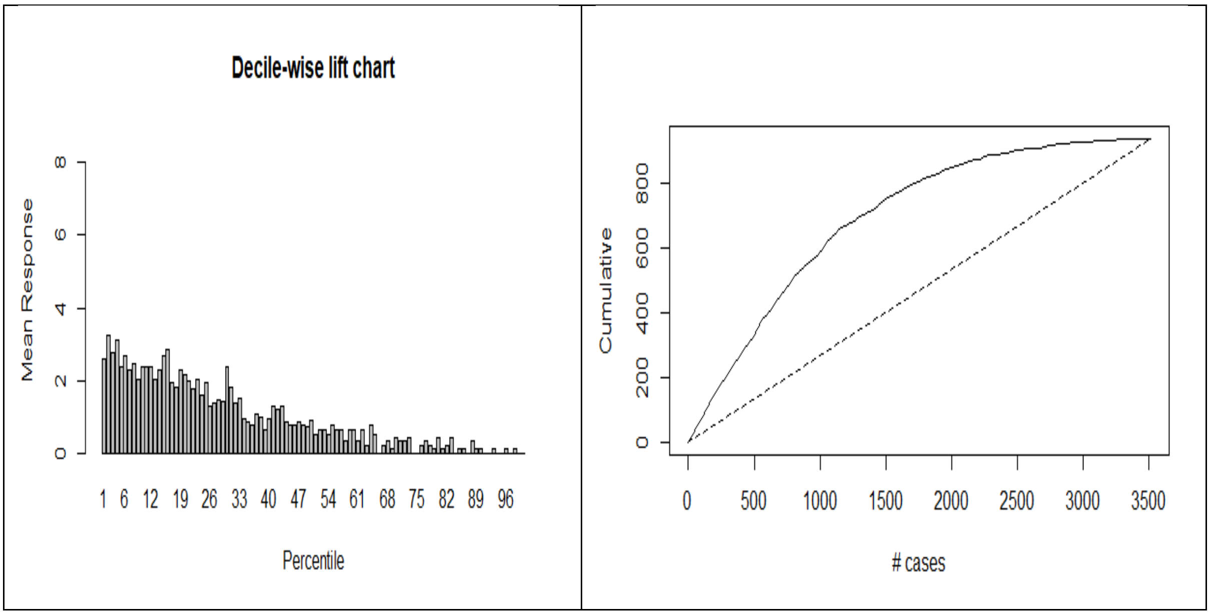
```
> caret::confusionMatrix(knn_model1,valid.knn$Churn,positive = "Yes")
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  1753  151
       Yes  827  784
                                          
               Accuracy : 0.7218          
                 95% CI : (0.7066, 0.7365)
    No Information Rate : 0.734           
    P-Value [Acc > NIR] : 0.9511          
                                          
                  Kappa : 0.4209          
                                          
 Mcnemar's Test P-Value : <2e-16          
                                          
            Sensitivity : 0.8385          
            Specificity : 0.6795          
         Pos Pred Value : 0.4867          
         Neg Pred Value : 0.9207          
             Prevalence : 0.2660          
         Detection Rate : 0.2230          
   Detection Prevalence : 0.4583          
      Balanced Accuracy : 0.7590          
                                          
       'Positive' Class : Yes
```



MODEL 2: Naïve Bayes

```
> caret::confusionMatrix(valid.NB.pred1,valid.NB$Churn,positive = "Yes")
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  1898  219
       Yes  682  716
                                         
               Accuracy : 0.7437         
                 95% CI : (0.7289, 0.758)
    No Information Rate : 0.734          
    P-Value [Acc > NIR] : 0.1001         
                                         
                  Kappa : 0.4331         
                                         
 Mcnemar's Test P-Value : <2e-16         
                                         
            Sensitivity : 0.7658         
            Specificity : 0.7357         
         Pos Pred Value : 0.5122         
         Neg Pred Value : 0.8966         
             Prevalence : 0.2660         
         Detection Rate : 0.2037         
   Detection Prevalence : 0.3977         
      Balanced Accuracy : 0.7507         
                                         
       'Positive' Class : Yes
```

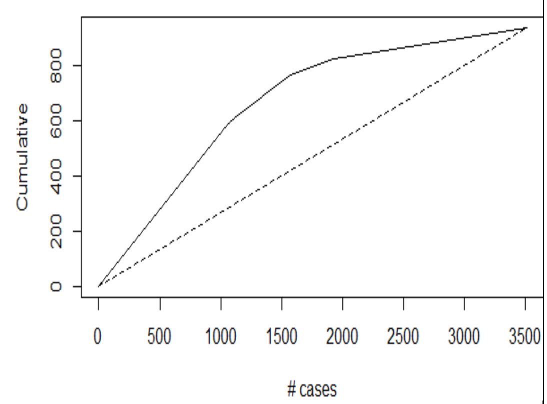


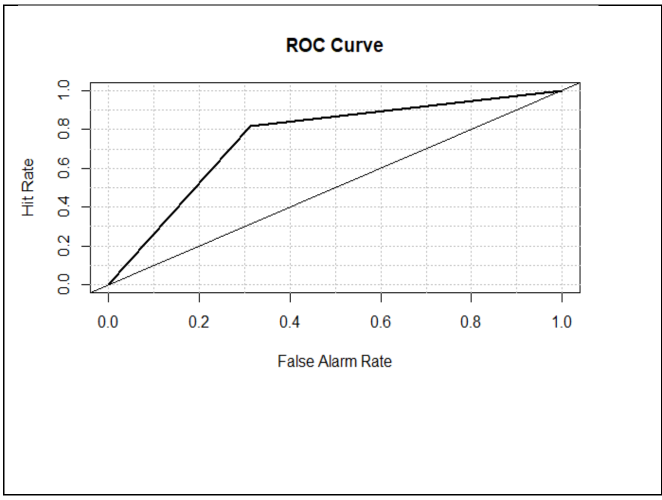


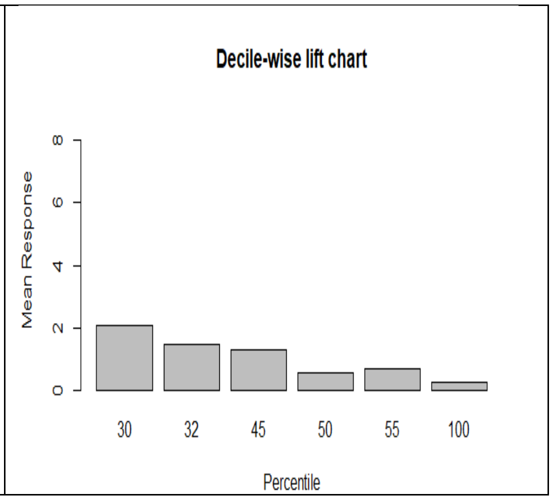
MODEL 3: CART-Random Forest-Boosted Trees

Classification Tree

```
> caret::confusionMatrix(valid.CT.pred1,valid.CT$Churn,positive = "Yes")
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  1775  170
       Yes  805  765

               Accuracy : 0.7226
                 95% CI : (0.7075, 0.7374)
    No Information Rate : 0.734
    P-Value [Acc > NIR] : 0.9385

                  Kappa : 0.4161

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8182
            Specificity : 0.6880
         Pos Pred Value : 0.4873
         Neg Pred Value : 0.9126
             Prevalence : 0.2660
         Detection Rate : 0.2176
   Detection Prevalence : 0.4467
      Balanced Accuracy : 0.7531

       'Positive' Class : Yes
```

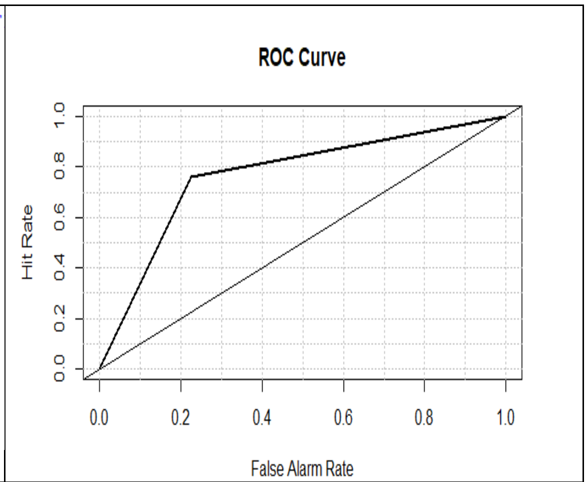


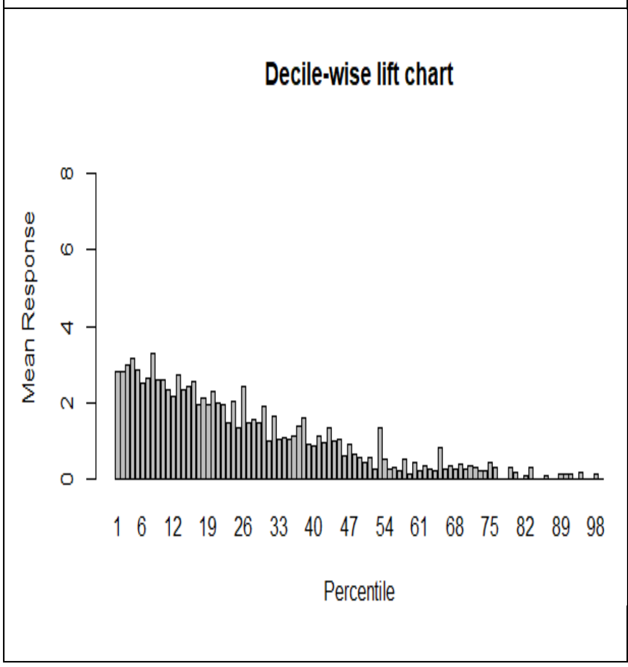


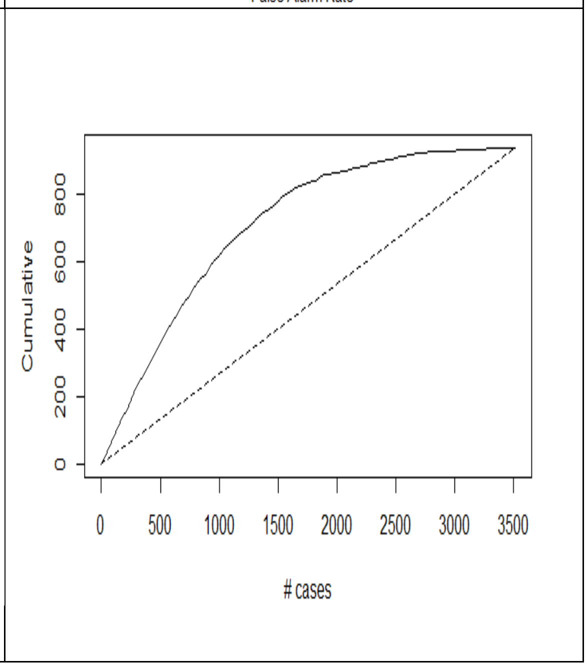


Random Forest

```
> caret::confusionMatrix(valid.CT.pred1,valid.CT$Churn,positive = "Yes"
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  2003  222
       Yes  577  713

               Accuracy : 0.7727
                 95% CI : (0.7585, 0.7865)
    No Information Rate : 0.734
    P-Value [Acc > NIR] : 7.373e-08

                  Kappa : 0.4807

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

            Sensitivity : 0.7626
            Specificity : 0.7764
         Pos Pred Value : 0.5527
         Neg Pred Value : 0.9002
             Prevalence : 0.2660
         Detection Rate : 0.2028
   Detection Prevalence : 0.3670
      Balanced Accuracy : 0.7695

       'Positive' Class : Yes
```





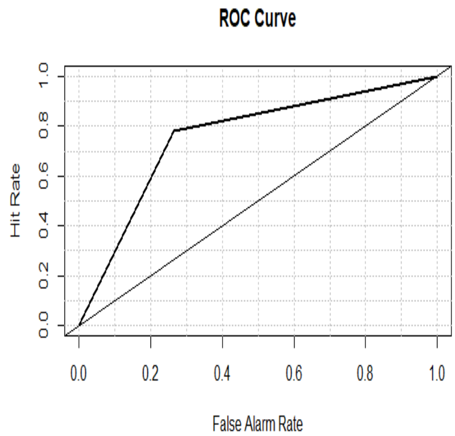


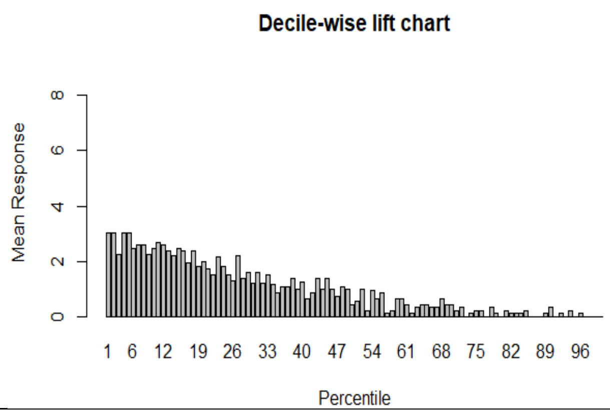
Boosted Trees

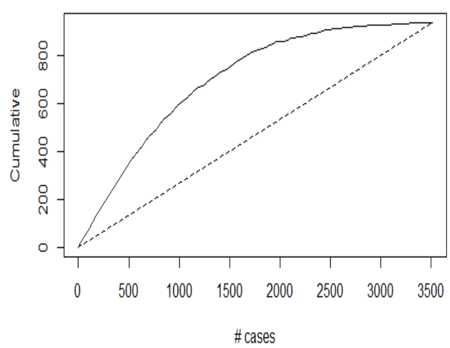
```
> caret::confusionMatrix(valid.CT.pred1,valid.CT$Churn,positive = "Yes")
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  1960  220
       Yes  620  715

               Accuracy : 0.761
                 95% CI : (0.7466, 0.775)
    No Information Rate : 0.734
    P-Value [Acc > NIR] : 0.0001336

                  Kappa : 0.4615

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

            Sensitivity : 0.7647
            Specificity : 0.7597
         Pos Pred Value : 0.5356
         Neg Pred Value : 0.8991
             Prevalence : 0.2660
         Detection Rate : 0.2034
   Detection Prevalence : 0.3798
      Balanced Accuracy : 0.7622

       'Positive' Class : Yes
```

MODEL 4: Logistic Regression

```
> caret::confusionMatrix(valid.logit.pred1,valid.logit$Churn,positive = "Yes")
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  1900  206
       Yes  680  729

               Accuracy : 0.7479
                 95% CI : (0.7332, 0.7622)
    No Information Rate : 0.734
    P-Value [Acc > NIR] : 0.03153

                  Kappa : 0.4443

 Mcnemar's Test P-Value : < 2e-16

            Sensitivity : 0.7797
            Specificity : 0.7364
         Pos Pred Value : 0.5174
         Neg Pred Value : 0.9022
             Prevalence : 0.2660
         Detection Rate : 0.2074
   Detection Prevalence : 0.4009
      Balanced Accuracy : 0.7581

       'Positive' Class : Yes
```

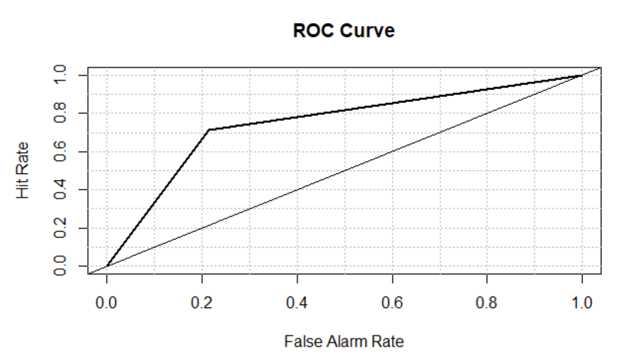


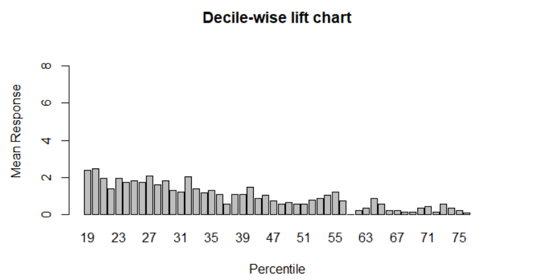


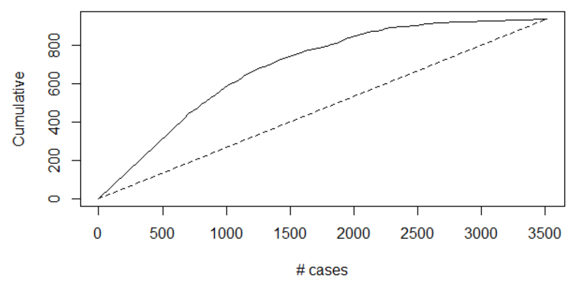


MODEL 5: Artificial Neural Network

```
> caret::confusionMatrix(valid.NN.pred1,valid.NN$Churn,positi
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  2025  269
       Yes  555  666

               Accuracy : 0.7656
                 95% CI : (0.7512, 0.7795)
    No Information Rate : 0.734
    P-Value [Acc > NIR] : 9.706e-06

                  Kappa : 0.453

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

            Sensitivity : 0.7123
            Specificity : 0.7849
         Pos Pred Value : 0.5455
         Neg Pred Value : 0.8827
             Prevalence : 0.2660
         Detection Rate : 0.1895
   Detection Prevalence : 0.3474
      Balanced Accuracy : 0.7486

       'Positive' Class : Yes
```

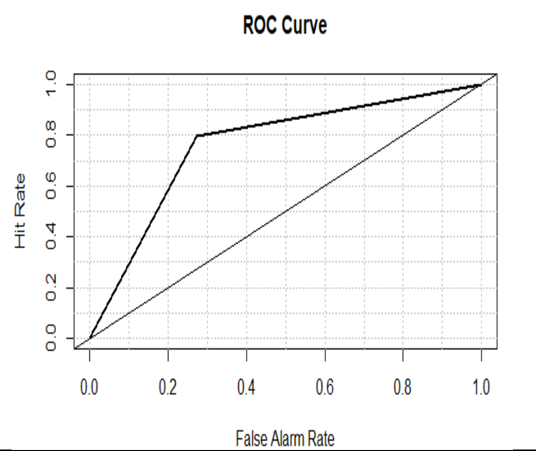


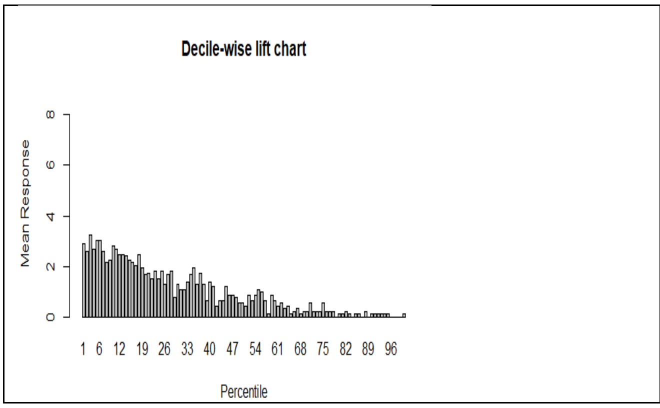


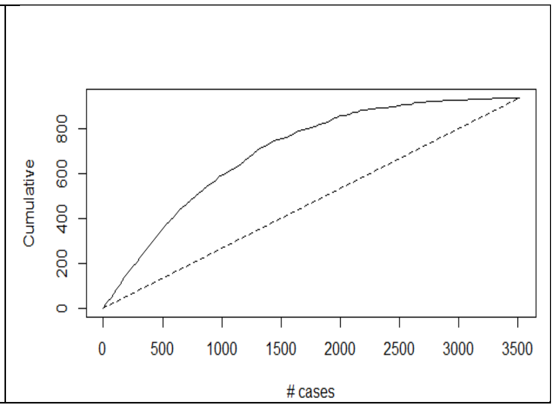


MODEL 6: Linear Discriminant Analysis

```
> caret::confusionMatrix(valid.LDA.pred1,valid.LDA$Churn,positive = "Yes")
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  1876  189
       Yes  704  746

               Accuracy : 0.7459
                 95% CI : (0.7312, 0.7603)
    No Information Rate : 0.734
    P-Value [Acc > NIR] : 0.05606

                  Kappa : 0.4466

 Mcnemar's Test P-Value : < 2e-16

            Sensitivity : 0.7979
            Specificity : 0.7271
         Pos Pred Value : 0.5145
         Neg Pred Value : 0.9085
             Prevalence : 0.2660
         Detection Rate : 0.2122
   Detection Prevalence : 0.4125
      Balanced Accuracy : 0.7625

       'Positive' Class : Yes
```

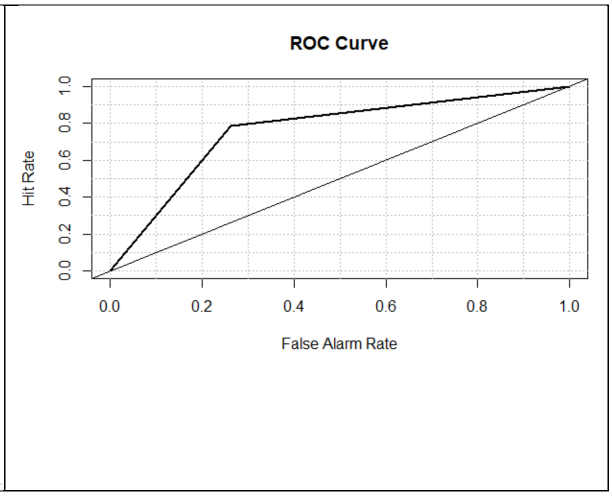






MODEL 7: Support Vector Machine

```
> caret::confusionMatrix(valid.SVM.pred1,valid.SVM$Churn,positive = "Yes")
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  1905  201
       Yes  675  734

               Accuracy : 0.7508
                 95% CI : (0.7361, 0.765)
    No Information Rate : 0.734
    P-Value [Acc > NIR] : 0.01237

                  Kappa : 0.4506

 Mcnemar's Test P-Value : < 2e-16

            Sensitivity : 0.7850
            Specificity : 0.7384
         Pos Pred Value : 0.5209
         Neg Pred Value : 0.9046
             Prevalence : 0.2660
         Detection Rate : 0.2088
   Detection Prevalence : 0.4009
      Balanced Accuracy : 0.7617

       'Positive' Class : Yes
```



VI. Discussion and Recommendation

From the performance evaluation of models discussed above, we observe that Random Forest gives the highest predictive accuracy followed by Boosted Trees. Thus, for the prediction of telecom churn rate, we can successfully implement Random Forest model and predict churn rate with a confidence of 80%. The models have been trained on oversampled data exposing them to higher positive cases thus providing us with a higher sensitivity and balanced accuracy.

Finally, cases predicted as "Churn" are to be analyzed further and provided with a course of action such as offers on mobile plans, discounts and priority emails.

VII. Summary

The case study mainly aims at solving one of the most common business problems faced in many industries, specially telecommunication. The objective of this study is to use prescriptive analytics to model a successful algorithm and recommend the best course of action. This case study involves all methods in data mining from exploration and preprocessing to choosing the best possible model. Finally, we conclude that Random Forest is the best method to model the dataset along with oversampling and the cases which are predicted as "Churn" are to be considered for further analysis and thus provided with the best course of action.

Appendix: R Code for use case study

```
.
#*************************

#Preprocessing & Visualization

#*************************
#Importing Data
churn <- read.csv("new churn.csv")
str(churn)
summary(churn)

#------------------------

#data exploration & visualization
library(ggplot2)
library(dplyr)
library(gplots)
#To detect missing values
heatmap(1*is.na(churn),Rowv = NA,Colv = NA)
#To detect missing values in corresponding rows/columns
lapply(churn,function(x) which(is.na(x)))
#Deleting observations with missing values
churn <- churn[complete.cases(churn),]
#To detect correlation among numerical variables
corr <- cor(churn[,c("tenure","MonthlyCharges","TotalCharges")])
gplots::heatmap.2(corr, Rowv = FALSE, Colv = FALSE, dendrogram = "none",cellnote =
round(corr,2),notecol = "black", key = FALSE, trace = 'none', margins = c(10,10))
#plots
ggplot2::ggplot(churn)+geom_bar(mapping = aes(x = Churn,fill = Churn))+ggtitle("Churn
Count")
ggplot2::ggplot(churn)+geom_boxplot(mapping = aes(y=MonthlyCharges))+ggtitle("Boxplot
of Monthly Charges")
ggplot2::ggplot(churn)+geom_boxplot(mapping = aes(y=TotalCharges))+ggtitle("Boxplot of
Total Charges")
ggplot2::ggplot(churn)+geom_boxplot(mapping = aes(y=tenure))+ggtitle("Boxplot of
Tenure")
ggplot2::ggplot(churn)+geom_bar(mapping = aes(tenure,fill = tenure))+xlab("Tenure
(Month)")+ggtitle("Distribution of Tenure")
g <- ggplot2::ggplot(churn)+geom_bar(mapping = aes(x=gender,fill = Churn))
s <- ggplot2::ggplot(churn)+geom_bar(mapping = aes(x=SeniorCitizen,fill = Churn))
p <- ggplot2::ggplot(churn)+geom_bar(mapping = aes(x=Partner,fill = Churn))
d <- ggplot2::ggplot(churn)+geom_bar(mapping = aes(x=Dependents,fill = Churn))
gridExtra::grid.arrange(g,s,p,d)
ps <- ggplot2::ggplot(churn)+geom_bar(mapping = aes(x=PhoneService,fill = Churn))
ml <- ggplot2::ggplot(churn)+geom_bar(mapping = aes(x=MultipleLines,fill = Churn))
is <- ggplot2::ggplot(churn)+geom_bar(mapping = aes(x=InternetService,fill = Churn))
os <- ggplot2::ggplot(churn)+geom_bar(mapping = aes(x=OnlineSecurity,fill = Churn))
gridExtra::grid.arrange(ps,ml,is,os)
ob <- ggplot2::ggplot(churn)+geom_bar(mapping = aes(x=OnlineBackup,fill = Churn))
```

```
dp <- ggplot2::ggplot(churn)+geom_bar(mapping = aes(x=DeviceProtection,fill = Churn))
ts <- ggplot2::ggplot(churn)+geom_bar(mapping = aes(x=TechSupport,fill = Churn))
st <- ggplot2::ggplot(churn)+geom_bar(mapping = aes(x=StreamingTV,fill = Churn))
gridExtra::grid.arrange(ob,dp,ts,st)
sm <- ggplot2::ggplot(churn)+geom_bar(mapping = aes(x=StreamingMovies,fill = Churn))
c <- ggplot2::ggplot(churn)+geom_bar(mapping = aes(x=Contract,fill = Churn))
pb <- ggplot2::ggplot(churn)+geom_bar(mapping = aes(x=PaperlessBilling,fill = Churn))
pm <- ggplot2::ggplot(churn)+geom_bar(mapping = aes(x=PaymentMethod,fill = Churn))
gridExtra::grid.arrange(sm,c,pb,pm)
ggplot2::ggplot(churn)+geom_boxplot(mapping = aes(y = MonthlyCharges,fill = Churn))
ggplot2::ggplot(churn)+geom_boxplot(mapping = aes(y = TotalCharges,fill = Churn))
ggplot2::ggplot(churn)+geom_histogram(mapping = aes(x=MonthlyCharges))
ggplot2::ggplot(churn)+geom_histogram(mapping = aes(x=TotalCharges))

#-------------------------

#Preprocessing
#Function to change 'No Phone/Internet Service to No'
sub1 <- function(x){
  gsub("No phone service","No",x)
}
sub2 <- function(x){
  gsub("No internet service","No",x)
}
#Applying function sub to data frame
churn <- data.frame(lapply(churn, sub1))
churn <- data.frame(lapply(churn, sub2))
#Converting factor to numeric
churn$tenure <- as.numeric(as.character(churn$tenure))
churn$MonthlyCharges <- as.numeric(as.character(churn$MonthlyCharges))
churn$TotalCharges <- as.numeric(as.character(churn$TotalCharges))
#Function to convert months to years
conv <- function(x){
  x/12
}
churn$tenure <- sapply(churn$tenure,conv)
#Binning tenure
churn$tenure[churn$tenure >= 0 & churn$tenure <=1] = '0-1'
churn$tenure[churn$tenure > 1 & churn$tenure <=2] = '1-2'
churn$tenure[churn$tenure > 2 & churn$tenure <=3] = '2-3'
churn$tenure[churn$tenure > 3 & churn$tenure <=4] = '3-4'
churn$tenure[churn$tenure > 4 & churn$tenure <=5] = '4-5'
churn$tenure[churn$tenure > 5 & churn$tenure <=6] = '5-6'
churn$tenure <- as.factor(churn$tenure)
#Standardizing columns Monthly Charges and Total Charges
churn[,c('MonthlyCharges','TotalCharges')] =
scale(churn[,c('MonthlyCharges','TotalCharges')])
#Visualization after preprocessing
ggplot2::ggplot(churn)+geom_bar(mapping = aes(tenure,fill = Churn))+xlab("Tenure
(years)")+ggtitle("Distribution of Tenure")
```

```
#------------------------
#************************

#KNN

#************************
#Importing Data
churn <- read.csv("new churn.csv")
str(churn)
summary(churn)

#------------------------

#Preprocessing
#Deleting observations with missing values
churn <- churn[complete.cases(churn),]
#Function to change 'No Phone/Internet Service to No'
sub1 <- function(x){
  gsub("No phone service","No",x)
}
sub2 <- function(x){
  gsub("No internet service","No",x)
}
#Applying function sub to data frame
churn <- data.frame(lapply(churn, sub1))
churn <- data.frame(lapply(churn, sub2))
#Converting factor to numeric
churn$tenure <- as.numeric(as.character(churn$tenure))
churn$MonthlyCharges <- as.numeric(as.character(churn$MonthlyCharges))
churn$TotalCharges <- as.numeric(as.character(churn$TotalCharges))
#Function to convert months to years
conv <- function(x){
  x/12
}
churn$tenure <- sapply(churn$tenure,conv)
#Binning tenure
churn$tenure[churn$tenure >= 0 & churn$tenure <=1] = '0-1'
churn$tenure[churn$tenure > 1 & churn$tenure <=2] = '1-2'
churn$tenure[churn$tenure > 2 & churn$tenure <=3] = '2-3'
churn$tenure[churn$tenure > 3 & churn$tenure <=4] = '3-4'
churn$tenure[churn$tenure > 4 & churn$tenure <=5] = '4-5'
churn$tenure[churn$tenure > 5 & churn$tenure <=6] = '5-6'
churn$tenure <- as.factor(churn$tenure)
#Standardizing columns Monthly Charges and Total Charges
churn[,c('MonthlyCharges','TotalCharges')] =
scale(churn[,c('MonthlyCharges','TotalCharges')])

#------------------------
```

```
#Partioning Data
#Original ratio
set.seed(123)
or <- sum(churn$Churn == "Yes")/sum(churn$Churn == "No")
churn.yes.index <- churn$Churn == "Yes"
churn.no.index <- churn$Churn == "No"
churn.yes.df <- churn[churn.yes.index,]
churn.no.df <- churn[churn.no.index,]
#Training/Validation
#Yes
train.yes.index <- sample(c(1:dim(churn.yes.df)[1]),dim(churn.yes.df)[1]/2)
train.yes.df <- churn.yes.df[train.yes.index,]
valid.yes.df <- churn.yes.df[-train.yes.index,]
#No
train.no.index <- sample(c(1:dim(churn.no.df)[1]),dim(churn.yes.df)[1]/2)
train.no.df <- churn.no.df[train.no.index,]
valid.no.df <- churn.no.df[-train.no.index,]
valid.no.index <- sample(c(1:dim(valid.no.df)[1]),(dim(train.yes.df)[1]/or))
valid.no.df <- churn.no.df[valid.no.index,]
#Combining Train/Valid
train.df <- rbind(train.yes.df,train.no.df)
valid.df <- rbind(valid.yes.df,valid.no.df)

#------------------------

#KNN
#Dummy Variable for KNN
#m dummies
#Function to create dummy variable
dum_knn <- function(x){
  model.matrix(~x-1,data = churn)
}
#Categorical Columns & Numerical Columns
cat <- churn[,-c(1,19,20,21)]
num <- churn[,c(1,19,20,21)]
#Creating Dummy Variables
dummy <- data.frame(sapply(cat, dum_knn))
#Combining variables to final dataset
churn.knn <- cbind(num,dummy)
str(churn.knn)

#------------------------

#Oversampling
train.knn <- churn.knn[rownames(train.df),]
valid.knn <- churn.knn[rownames(valid.df),]
#Sampling
churn.sam <- rbind(train.knn,valid.knn)
train.index <- sample(c(1:dim(churn.sam)[1]),0.60*dim(churn.sam)[1])
train.sam <- churn.sam[train.index,]
```

```
valid.sam <- churn.sam[-train.index,]

#------------------------

#KNN
library(class)
i <- 1
error <- data.frame(matrix(ncol = 2,nrow = 0))
error_name <- c("train","valid")
colnames(error) <- error_name
while (i<=30) {
  print(i)
  knn_model1 <- class::knn(train = train.knn[,-c(1,4)],test = valid.knn[,-c(1,4)],cl =
train.knn[,4],k = i)
  knn_model2 <- class::knn(train = train.knn[,-c(1,4)],test = train.knn[,-c(1,4)],cl =
train.knn[,4],k = i)
  cm1 <- caret::confusionMatrix(knn_model1,valid.knn$Churn,positive = "Yes")
  cm2 <- caret::confusionMatrix(knn_model2,train.knn$Churn,positive = "Yes")
  error[i,1] <- 1 - cm2$byClass[11]
  error[i,2] <- 1 - cm1$byClass[11]
  i = i + 1
}
#Oversampled
#K=23
knn_model1 <- class::knn(train = train.knn[,-c(1,4)],test = valid.knn[,-c(1,4)],cl =
train.knn[,4],k = 23)
#Sampled
#K=23
knn_model2 <- class::knn(train = train.sam[,-c(1,4)],test = valid.sam[,-c(1,4)],cl =
train.sam[,4],k = 23)

#------------------------

#Model Performance
library(verification)
library(gmodels)
library(caret)
#Oversampled
#Confusion Matrix
gmodels::CrossTable(knn_model1,valid.knn[,4],prop.r = FALSE,prop.c = FALSE,prop.t =
FALSE,prop.chisq = FALSE)
caret::confusionMatrix(knn_model1,valid.knn$Churn,positive = "Yes")
#ROC Curve
verification::roc.plot(ifelse(valid.knn$Churn == "Yes",1,0),ifelse(knn_model1 == "Yes",1,0))
#Sampled
#Confusion Matrix
gmodels::CrossTable(knn_model2,valid.sam[,4],prop.r = FALSE,prop.c = FALSE,prop.t =
FALSE,prop.chisq = FALSE)
caret::confusionMatrix(knn_model2,valid.sam$Churn,positive = "Yes")
#ROC Curve
```

```
verification::roc.plot(ifelse(valid.sam$Churn == "Yes",1,0),ifelse(knn_model2 ==
"Yes",1,0))

#------------------------
#************************

#Naive Bayes

#************************
#Importing Data
churn <- read.csv("new churn.csv")
str(churn)
summary(churn)

#------------------------

#Preprocessing
#Deleting observations with missing values
churn <- churn[complete.cases(churn),]
#Function to change 'No Phone/Internet Service to No'
sub1 <- function(x){
  gsub("No phone service","No",x)
}
sub2 <- function(x){
  gsub("No internet service","No",x)
}
#Applying function sub to data frame
churn <- data.frame(lapply(churn, sub1))
churn <- data.frame(lapply(churn, sub2))
#Converting factor to numeric
churn$tenure <- as.numeric(as.character(churn$tenure))
churn$MonthlyCharges <- as.numeric(as.character(churn$MonthlyCharges))
churn$TotalCharges <- as.numeric(as.character(churn$TotalCharges))
#Function to convert months to years
conv <- function(x){
  x/12
}
churn$tenure <- sapply(churn$tenure,conv)
#Binning tenure
churn$tenure[churn$tenure >= 0 & churn$tenure <=1] = '0-1'
churn$tenure[churn$tenure > 1 & churn$tenure <=2] = '1-2'
churn$tenure[churn$tenure > 2 & churn$tenure <=3] = '2-3'
churn$tenure[churn$tenure > 3 & churn$tenure <=4] = '3-4'
churn$tenure[churn$tenure > 4 & churn$tenure <=5] = '4-5'
churn$tenure[churn$tenure > 5 & churn$tenure <=6] = '5-6'
churn$tenure <- as.factor(churn$tenure)
#Binning of total charges and Monthly Charges
library(OneR)
churn$MonthlyCharges <- OneR::bin(churn$MonthlyCharges, nbins = 5, labels =
c(1,2,3,4,5))
```

```
churn$TotalCharges <- OneR::bin(churn$TotalCharges, nbins = 10, labels =
c(1,2,3,4,5,6,7,8,9,10))
summary(churn$MonthlyCharges)
summary(churn$TotalCharges)

#------------------------

#Partioning Data
#Original ratio
set.seed(123)
or <- sum(churn$Churn == "Yes")/sum(churn$Churn == "No")
churn.yes.index <- churn$Churn == "Yes"
churn.no.index <- churn$Churn == "No"
churn.yes.df <- churn[churn.yes.index,]
churn.no.df <- churn[churn.no.index,]
#Training/Validation
#Yes
train.yes.index <- sample(c(1:dim(churn.yes.df)[1]),dim(churn.yes.df)[1]/2)
train.yes.df <- churn.yes.df[train.yes.index,]
valid.yes.df <- churn.yes.df[-train.yes.index,]
#No
train.no.index <- sample(c(1:dim(churn.no.df)[1]),dim(churn.yes.df)[1]/2)
train.no.df <- churn.no.df[train.no.index,]
valid.no.df <- churn.no.df[-train.no.index,]
valid.no.index <- sample(c(1:dim(valid.no.df)[1]),(dim(train.yes.df)[1]/or))
valid.no.df <- churn.no.df[valid.no.index,]
#Combining Train/Valid
train.df <- rbind(train.yes.df,train.no.df)
valid.df <- rbind(valid.yes.df,valid.no.df)
#Oversampling
train.NB <- train.df
valid.NB <- valid.df
#Sampling
churn.sam <- rbind(train.NB,valid.NB)
train.index <- sample(c(1:dim(churn.sam)[1]),0.60*dim(churn.sam)[1])
train.sam <- churn.sam[train.index,]
valid.sam <- churn.sam[-train.index,]

#------------------------

#Naive Bayes
library(e1071)
#Oversampled
train.NB <- train.NB[,-1]
valid.NB <- valid.NB[,-1]
NB_model1 <- e1071::naiveBayes(Churn~.,data = train.NB,type="class")
valid.NB.pred1 <- predict(NB_model1,newdata = valid.NB)
pred.prob1 <- predict(NB_model1,newdata = valid.NB,type = "raw")
#Sampled
train.sam <- train.sam[,-1]
```

```
valid.sam <- valid.sam[,-1]
NB_model2 <- e1071::naiveBayes(Churn~.,data = train.sam,type="class")
valid.NB.pred2 <- predict(NB_model2,newdata = valid.sam)
pred.prob2 <- predict(NB_model2,newdata = valid.sam,type = "raw")

#------------------------

#Model Performance
library(gmodels)
library(caret)
library(gains)
library(verification)
#Oversampled
#Confusion Matrix
gmodels::CrossTable(valid.NB.pred1,valid.NB$Churn,prop.r = FALSE,prop.c =
FALSE,prop.t = FALSE,prop.chisq = FALSE)
caret::confusionMatrix(valid.NB.pred1,valid.NB$Churn,positive = "Yes")
#Lift Chart
gain <- gains(ifelse(valid.NB$Churn=="Yes",1,0), pred.prob1[,2], groups=100)
plot(c(0,gain$cume.pct.of.total*sum(valid.NB$Churn=="Yes"))~c(0,gain$cume.obs),
    xlab="# cases", ylab="Cumulative", main="", type="l")
lines(c(0,sum(valid.NB$Churn=="Yes"))~c(0, dim(valid.NB)[1]), lty=2)
#Decile-wise Lift Chart
heights <- gain$mean.resp/mean(ifelse(valid.NB$Churn=="Yes",1,0))
midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),
                 xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")
#ROC Curve
verification::roc.plot(ifelse(valid.NB$Churn=="Yes",1,0),ifelse(valid.NB.pred1 ==
"Yes",1,0))
#Sampled
#Confusion Matrix
gmodels::CrossTable(valid.NB.pred2,valid.sam$Churn,prop.r = FALSE,prop.c =
FALSE,prop.t = FALSE,prop.chisq = FALSE)
caret::confusionMatrix(valid.NB.pred2,valid.sam$Churn,positive = "Yes")
#Lift Chart
gain <- gains(ifelse(valid.sam$Churn=="Yes",1,0), pred.prob2[,2], groups=100)
plot(c(0,gain$cume.pct.of.total*sum(valid.sam$Churn=="Yes"))~c(0,gain$cume.obs),
    xlab="# cases", ylab="Cumulative", main="", type="l")
lines(c(0,sum(valid.sam$Churn=="Yes"))~c(0, dim(valid.sam)[1]), lty=2)
#Decile-wise Lift Chart
heights <- gain$mean.resp/mean(ifelse(valid.sam$Churn=="Yes",1,0))
midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),
                 xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")
#ROC Curve
verification::roc.plot(ifelse(valid.sam$Churn=="Yes",1,0),ifelse(valid.NB.pred2 ==
"Yes",1,0))

#------------------------
#************************
```

```
#CART/Random Forest/Boosted Trees

#************************
#Importing Data
churn <- read.csv("new churn.csv")
str(churn)
summary(churn)

#------------------------

#Preprocessing
#Deleting observations with missing values
churn <- churn[complete.cases(churn),]
#Function to change 'No Phone/Internet Service to No'
sub1 <- function(x){
  gsub("No phone service","No",x)
}
sub2 <- function(x){
  gsub("No internet service","No",x)
}
#Applying function sub to data frame
churn <- data.frame(lapply(churn, sub1))
churn <- data.frame(lapply(churn, sub2))
#Converting factor to numeric
churn$tenure <- as.numeric(as.character(churn$tenure))
churn$MonthlyCharges <- as.numeric(as.character(churn$MonthlyCharges))
churn$TotalCharges <- as.numeric(as.character(churn$TotalCharges))
#Function to convert months to years
conv <- function(x){
  x/12
}
churn$tenure <- sapply(churn$tenure,conv)
#Binning tenure
churn$tenure[churn$tenure >= 0 & churn$tenure <=1] = '0-1'
churn$tenure[churn$tenure > 1 & churn$tenure <=2] = '1-2'
churn$tenure[churn$tenure > 2 & churn$tenure <=3] = '2-3'
churn$tenure[churn$tenure > 3 & churn$tenure <=4] = '3-4'
churn$tenure[churn$tenure > 4 & churn$tenure <=5] = '4-5'
churn$tenure[churn$tenure > 5 & churn$tenure <=6] = '5-6'
churn$tenure <- as.factor(churn$tenure)
#Standardizing columns Monthly Charges and Total Charges
churn[,c('MonthlyCharges','TotalCharges')] =
scale(churn[,c('MonthlyCharges','TotalCharges')])
churn$Churn <- as.factor(churn$Churn)

#------------------------

#Partioning Data
#Original ratio
set.seed(123)
```

```
or <- sum(churn$Churn == "Yes")/sum(churn$Churn == "No")
churn.yes.index <- churn$Churn == "Yes"
churn.no.index <- churn$Churn == "No"
churn.yes.df <- churn[churn.yes.index,]
churn.no.df <- churn[churn.no.index,]
#Training/Validation
#Yes
train.yes.index <- sample(c(1:dim(churn.yes.df)[1]),dim(churn.yes.df)[1]/2)
train.yes.df <- churn.yes.df[train.yes.index,]
valid.yes.df <- churn.yes.df[-train.yes.index,]
#No
train.no.index <- sample(c(1:dim(churn.no.df)[1]),dim(churn.yes.df)[1]/2)
train.no.df <- churn.no.df[train.no.index,]
valid.no.df <- churn.no.df[-train.no.index,]
valid.no.index <- sample(c(1:dim(valid.no.df)[1]),(dim(train.yes.df)[1]/or))
valid.no.df <- churn.no.df[valid.no.index,]
#Combining Train/Valid
train.df <- rbind(train.yes.df,train.no.df)
valid.df <- rbind(valid.yes.df,valid.no.df)
#Oversampling
train.CT <- train.df
valid.CT <- valid.df
#Sampling
churn.sam <- rbind(train.CT,valid.CT)
train.index <- sample(c(1:dim(churn.sam)[1]),0.60*dim(churn.sam)[1])
train.sam <- churn.sam[train.index,]
valid.sam <- churn.sam[-train.index,]

#------------------------

#Classification Tree
library(rpart)
library(rpart.plot)
#Oversampled
train.CT <- train.CT[,-1]
valid.CT <- valid.CT[,-1]
CT_model1 <- rpart::rpart(Churn~.,data = train.CT,method = "class")
rpart.plot::prp(CT_model1,type = 1, extra = 1, split.font = 1, varlen = -10,under = TRUE)
valid.CT.pred1 <- as.factor(predict(CT_model1,valid.CT,type = "class"))
pred.prob1 <- predict(CT_model1,valid.CT,type = "prob")
#Sampled
train.sam <- train.sam[,-1]
valid.sam <- valid.sam[,-1]
CT_model2 <- rpart::rpart(Churn~.,data = train.sam,method = "class")
rpart.plot::prp(CT_model2,type = 1, extra = 1, split.font = 1, varlen = -10,under = TRUE)
valid.CT.pred2 <- as.factor(predict(CT_model2,valid.sam,type = "class"))
pred.prob2 <- predict(CT_model2,valid.sam,type = "prob")

#------------------------
```

```
#Model Performance
library(gmodels)
library(caret)
library(gains)
library(verification)
#Oversampled
#Confusion Matrix
gmodels::CrossTable(valid.CT.pred1,valid.CT$Churn,prop.r = FALSE,prop.c =
FALSE,prop.t = FALSE,prop.chisq = FALSE)
caret::confusionMatrix(valid.CT.pred1,valid.CT$Churn,positive = "Yes")
#Lift Chart
gain <- gains(ifelse(valid.CT$Churn=="Yes",1,0), pred.prob1[,2], groups=100)
plot(c(0,gain$cume.pct.of.total*sum(valid.CT$Churn=="Yes"))~c(0,gain$cume.obs),
    xlab="# cases", ylab="Cumulative", main="", type="l")
lines(c(0,sum(valid.CT$Churn=="Yes"))~c(0, dim(valid.CT)[1]), lty=2)
#Decile-wise Lift Chart
heights <- gain$mean.resp/mean(ifelse(valid.CT$Churn=="Yes",1,0))
midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),
                xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")
#ROC Curve
verification::roc.plot(ifelse(valid.CT$Churn=="Yes",1,0),ifelse(valid.CT.pred1 ==
"Yes",1,0))
#Sampled
#Confusion Matrix
gmodels::CrossTable(valid.CT.pred2,valid.sam$Churn,prop.r = FALSE,prop.c =
FALSE,prop.t = FALSE,prop.chisq = FALSE)
caret::confusionMatrix(valid.CT.pred2,valid.sam$Churn,positive = "Yes")
#Lift Chart
gain <- gains(ifelse(valid.sam$Churn=="Yes",1,0), pred.prob2[,2], groups=100)
plot(c(0,gain$cume.pct.of.total*sum(valid.sam$Churn=="Yes"))~c(0,gain$cume.obs),
    xlab="# cases", ylab="Cumulative", main="", type="l")
lines(c(0,sum(valid.sam$Churn=="Yes"))~c(0, dim(valid.sam)[1]), lty=2)
#Decile-wise Lift Chart
heights <- gain$mean.resp/mean(ifelse(valid.sam$Churn=="Yes",1,0))
midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),
                xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")
#ROC Curve
verification::roc.plot(ifelse(valid.sam$Churn=="Yes",1,0),ifelse(valid.CT.pred2 ==
"Yes",1,0))

#------------------------

#Pruning Classification Tree
#Oversampled
CT_pruned1 <- rpart::prune(CT_model1, cp =
CT_model1$cptable[which.min(CT_model1$cptable[,"xerror"]),"CP"])
rpart.plot::prp(CT_pruned1,type = 1, extra = 1, split.font = 1, varlen = -10,under = TRUE)
valid.CT.pred1 <- as.factor(predict(CT_pruned1,valid.CT,type = "class"))
pred.prob1 <- predict(CT_pruned1,valid.CT,type = "prob")
#Sampled
```

```
CT_pruned2 <- rpart::prune(CT_model2, cp =
CT_model2$cptable[which.min(CT_model2$cptable[,"xerror"]),"CP"])
rpart.plot::prp(CT_pruned2,type = 1, extra = 1, split.font = 1, varlen = -10,under = TRUE)
valid.CT.pred2 <- as.factor(predict(CT_pruned2,valid.sam,type = "class"))
pred.prob2 <- predict(CT_pruned2,valid.sam,type = "prob")

#------------------------

#Model Performance
library(gmodels)
library(caret)
library(gains)
library(verification)
#Oversampled
#Confusion Matrix
gmodels::CrossTable(valid.CT.pred1,valid.CT$Churn,prop.r = FALSE,prop.c =
FALSE,prop.t = FALSE,prop.chisq = FALSE)
caret::confusionMatrix(valid.CT.pred1,valid.CT$Churn,positive = "Yes")
#Lift Chart
gain <- gains(ifelse(valid.CT$Churn=="Yes",1,0), pred.prob1[,2], groups=100)
plot(c(0,gain$cume.pct.of.total*sum(valid.CT$Churn=="Yes"))~c(0,gain$cume.obs),
    xlab="# cases", ylab="Cumulative", main="", type="l")
lines(c(0,sum(valid.CT$Churn=="Yes"))~c(0, dim(valid.CT)[1]), lty=2)
#Decile-wise Lift Chart
heights <- gain$mean.resp/mean(ifelse(valid.CT$Churn=="Yes",1,0))
midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),
                xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")
#ROC Curve
verification::roc.plot(ifelse(valid.CT$Churn=="Yes",1,0),ifelse(valid.CT.pred1 ==
"Yes",1,0))
#Sampled
#Confusion Matrix
gmodels::CrossTable(valid.CT.pred2,valid.sam$Churn,prop.r = FALSE,prop.c =
FALSE,prop.t = FALSE,prop.chisq = FALSE)
caret::confusionMatrix(valid.CT.pred2,valid.sam$Churn,positive = "Yes")
#Lift Chart
gain <- gains(ifelse(valid.sam$Churn=="Yes",1,0), pred.prob2[,2], groups=100)
plot(c(0,gain$cume.pct.of.total*sum(valid.sam$Churn=="Yes"))~c(0,gain$cume.obs),
    xlab="# cases", ylab="Cumulative", main="", type="l")
lines(c(0,sum(valid.sam$Churn=="Yes"))~c(0, dim(valid.sam)[1]), lty=2)
#Decile-wise Lift Chart
heights <- gain$mean.resp/mean(ifelse(valid.sam$Churn=="Yes",1,0))
midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),
                xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")
#ROC Curve
verification::roc.plot(ifelse(valid.sam$Churn=="Yes",1,0),ifelse(valid.CT.pred2 ==
"Yes",1,0))

#------------------------
```

```
#Random Forest
library(randomForest)
#Oversampled
RF_model1 <- randomForest::randomForest(Churn ~ ., data = train.CT, ntree = 500, mtry =
4, nodesize = 5, importance = TRUE)
#Variable Importance Plot
varImpPlot(RF_model1, type = 1)
valid.CT.pred1 <- as.factor(predict(RF_model1,valid.CT,type = "class"))
pred.prob1 <- predict(RF_model1,valid.CT,type = "prob")
#Sampled
RF_model2 <- randomForest::randomForest(Churn ~ ., data = train.sam, ntree = 500, mtry =
4, nodesize = 5, importance = TRUE)
#Variable Importance Plot
varImpPlot(RF_model2, type = 1)
valid.CT.pred2 <- as.factor(predict(RF_model2,valid.sam,type = "class"))
pred.prob2 <- predict(RF_model2,valid.sam,type = "prob")

#------------------------

#Model Performance
library(gmodels)
library(caret)
library(gains)
library(verification)
#Oversampled
#Confusion Matrix
gmodels::CrossTable(valid.CT.pred1,valid.CT$Churn,prop.r = FALSE,prop.c =
FALSE,prop.t = FALSE,prop.chisq = FALSE)
caret::confusionMatrix(valid.CT.pred1,valid.CT$Churn,positive = "Yes")
#Lift Chart
gain <- gains(ifelse(valid.CT$Churn=="Yes",1,0), pred.prob1[,2], groups=100)
plot(c(0,gain$cume.pct.of.total*sum(valid.CT$Churn=="Yes"))~c(0,gain$cume.obs),
    xlab="# cases", ylab="Cumulative", main="", type="l")
lines(c(0,sum(valid.CT$Churn=="Yes"))~c(0, dim(valid.CT)[1]), lty=2)
#Decile-wise Lift Chart
heights <- gain$mean.resp/mean(ifelse(valid.CT$Churn=="Yes",1,0))
midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),
                xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")
#ROC Curve
verification::roc.plot(ifelse(valid.CT$Churn=="Yes",1,0),ifelse(valid.CT.pred1 ==
"Yes",1,0))
#Sampled
#Confusion Matrix
gmodels::CrossTable(valid.CT.pred2,valid.sam$Churn,prop.r = FALSE,prop.c =
FALSE,prop.t = FALSE,prop.chisq = FALSE)
caret::confusionMatrix(valid.CT.pred2,valid.sam$Churn,positive = "Yes")
#Lift Chart
gain <- gains(ifelse(valid.sam$Churn=="Yes",1,0), pred.prob2[,2], groups=100)
plot(c(0,gain$cume.pct.of.total*sum(valid.sam$Churn=="Yes"))~c(0,gain$cume.obs),
    xlab="# cases", ylab="Cumulative", main="", type="l")
```

```
lines(c(0,sum(valid.sam$Churn=="Yes"))~c(0, dim(valid.sam)[1]), lty=2)
#Decile-wise Lift Chart
heights <- gain$mean.resp/mean(ifelse(valid.sam$Churn=="Yes",1,0))
midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),
                xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")
#ROC Curve
verification::roc.plot(ifelse(valid.sam$Churn=="Yes",1,0),ifelse(valid.CT.pred2 ==
"Yes",1,0))

#------------------------

#Boosted Trees
library(adabag)
#Oversampled
boost_model1 <- adabag::boosting(Churn~.,data = train.CT)
valid.CT.pred1 <- as.factor(predict(boost_model1,valid.CT)$class)
pred.prob1 <- predict(boost_model1,valid.CT)$prob
#Sampled
boost_model2 <- adabag::boosting(Churn~.,data = train.sam)
valid.CT.pred2 <- as.factor(predict(boost_model2,valid.sam)$class)
pred.prob2 <- predict(boost_model2,valid.sam)$prob

#------------------------

#Model Performance
library(gmodels)
library(caret)
library(gains)
library(verification)
#Oversampled
#Confusion Matrix
gmodels::CrossTable(valid.CT.pred1,valid.CT$Churn,prop.r = FALSE,prop.c =
FALSE,prop.t = FALSE,prop.chisq = FALSE)
caret::confusionMatrix(valid.CT.pred1,valid.CT$Churn,positive = "Yes")
#Lift Chart
gain <- gains(ifelse(valid.CT$Churn=="Yes",1,0), pred.prob1[,2], groups=100)
plot(c(0,gain$cume.pct.of.total*sum(valid.CT$Churn=="Yes"))~c(0,gain$cume.obs),
   xlab="# cases", ylab="Cumulative", main="", type="l")
lines(c(0,sum(valid.CT$Churn=="Yes"))~c(0, dim(valid.CT)[1]), lty=2)
#Decile-wise Lift Chart
heights <- gain$mean.resp/mean(ifelse(valid.CT$Churn=="Yes",1,0))
midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),
                xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")
#ROC Curve
verification::roc.plot(ifelse(valid.CT$Churn=="Yes",1,0),ifelse(valid.CT.pred1 ==
"Yes",1,0))
#Sampled
#Confusion Matrix
gmodels::CrossTable(valid.CT.pred2,valid.sam$Churn,prop.r = FALSE,prop.c =
FALSE,prop.t = FALSE,prop.chisq = FALSE)
```

```
caret::confusionMatrix(valid.CT.pred2,valid.sam$Churn,positive = "Yes")
#Lift Chart
gain <- gains(ifelse(valid.sam$Churn=="Yes",1,0), pred.prob2[,2], groups=100)
plot(c(0,gain$cume.pct.of.total*sum(valid.sam$Churn=="Yes"))~c(0,gain$cume.obs),
    xlab="# cases", ylab="Cumulative", main="", type="l")
lines(c(0,sum(valid.sam$Churn=="Yes"))~c(0, dim(valid.sam)[1]), lty=2)
#Decile-wise Lift Chart
heights <- gain$mean.resp/mean(ifelse(valid.sam$Churn=="Yes",1,0))
midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),
                xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")
#ROC Curve
verification::roc.plot(ifelse(valid.sam$Churn=="Yes",1,0),ifelse(valid.CT.pred2 ==
"Yes",1,0))

#------------------------
#************************

#Logistic Regression

#************************
#Importing Data
churn <- read.csv("new churn.csv")
str(churn)
summary(churn)

#------------------------

#Preprocessing
#Deleting observations with missing values
churn <- churn[complete.cases(churn),]
#Function to change 'No Phone/Internet Service to No'
sub1 <- function(x){
  gsub("No phone service","No",x)
}
sub2 <- function(x){
  gsub("No internet service","No",x)
}
#Applying function sub to data frame
churn <- data.frame(lapply(churn, sub1))
churn <- data.frame(lapply(churn, sub2))
#Converting factor to numeric
churn$tenure <- as.numeric(as.character(churn$tenure))
churn$MonthlyCharges <- as.numeric(as.character(churn$MonthlyCharges))
churn$TotalCharges <- as.numeric(as.character(churn$TotalCharges))
#Function to convert months to years
conv <- function(x){
  x/12
}
churn$tenure <- sapply(churn$tenure,conv)
#Binning tenure
```

```
churn$tenure[churn$tenure >= 0 & churn$tenure <=1] = '0-1'
churn$tenure[churn$tenure > 1 & churn$tenure <=2] = '1-2'
churn$tenure[churn$tenure > 2 & churn$tenure <=3] = '2-3'
churn$tenure[churn$tenure > 3 & churn$tenure <=4] = '3-4'
churn$tenure[churn$tenure > 4 & churn$tenure <=5] = '4-5'
churn$tenure[churn$tenure > 5 & churn$tenure <=6] = '5-6'
churn$tenure <- as.factor(churn$tenure)
#Standardizing columns Monthly Charges and Total Charges
churn[,c('MonthlyCharges','TotalCharges')] =
scale(churn[,c('MonthlyCharges','TotalCharges')])

#------------------------

#Partioning Data
#Original ratio
set.seed(123)
or <- sum(churn$Churn == "Yes")/sum(churn$Churn == "No")
churn.yes.index <- churn$Churn == "Yes"
churn.no.index <- churn$Churn == "No"
churn.yes.df <- churn[churn.yes.index,]
churn.no.df <- churn[churn.no.index,]
#Training/Validation
#Yes
train.yes.index <- sample(c(1:dim(churn.yes.df)[1]),dim(churn.yes.df)[1]/2)
train.yes.df <- churn.yes.df[train.yes.index,]
valid.yes.df <- churn.yes.df[-train.yes.index,]
#No
train.no.index <- sample(c(1:dim(churn.no.df)[1]),dim(churn.yes.df)[1]/2)
train.no.df <- churn.no.df[train.no.index,]
valid.no.df <- churn.no.df[-train.no.index,]
valid.no.index <- sample(c(1:dim(valid.no.df)[1]),(dim(train.yes.df)[1]/or))
valid.no.df <- churn.no.df[valid.no.index,]
#Combining Train/Valid
train.df <- rbind(train.yes.df,train.no.df)
valid.df <- rbind(valid.yes.df,valid.no.df)

#------------------------

#Dummy Variable for other algorithms
#m-1 dummies
#Categorical Columns & Numerical Columns
cat <- churn[,-c(1,19,20,21)]
num <- churn[,c(1,19,20,21)]
#Function to create dummy variable
dum <- function(x){
  model.matrix(~x-1,data = churn)[,-1]
}
#Creating Dummy Variables
dummy <- data.frame(sapply(cat, dum))
#Combining variables to final dataset
```

```
churn.logit <- cbind(num,dummy)
str(churn.logit)

#------------------------

#Oversampling
train.logit <- churn.logit[rownames(train.df),]
valid.logit <- churn.logit[rownames(valid.df),]
#Sampling
churn.sam <- rbind(train.logit,valid.logit)
train.index <- sample(c(1:dim(churn.sam)[1]),0.60*dim(churn.sam)[1])
train.sam <- churn.sam[train.index,]
valid.sam <- churn.sam[-train.index,]

#------------------------

#Logistic Regression
#Oversampled
train.logit <- train.logit[,-1]
valid.logit <- valid.logit[,-1]
logit_model1 <- glm(Churn~.,data = train.logit,family = "binomial")
valid.logit.pred1 <- as.factor(ifelse(predict(logit_model1,valid.logit,type = "response") >
0.50,"Yes","No"))
pred.prob1 <- predict(logit_model1,valid.logit,type = "response")
#Sampled
train.sam <- train.sam[,-1]
valid.sam <- valid.sam[,-1]
logit_model2 <- glm(Churn~.,data = train.sam,family = "binomial")
valid.logit.pred2 <- as.factor(ifelse(predict(logit_model2,valid.sam,type = "response") >
0.50,"Yes","No"))
pred.prob2 <- predict(logit_model1,valid.sam,type = "response")

#------------------------

#Model Performance
library(gmodels)
library(caret)
library(gains)
library(verification)
#Oversampled
#Confusion Matrix
gmodels::CrossTable(valid.logit.pred1,valid.logit$Churn,prop.r = FALSE,prop.c =
FALSE,prop.t = FALSE,prop.chisq = FALSE)
caret::confusionMatrix(valid.logit.pred1,valid.logit$Churn,positive = "Yes")
#Lift Chart
gain <- gains(ifelse(valid.logit$Churn=="Yes",1,0), pred.prob1, groups=100)
plot(c(0,gain$cume.pct.of.total*sum(valid.logit$Churn=="Yes"))~c(0,gain$cume.obs),
     xlab="# cases", ylab="Cumulative", main="", type="l")
lines(c(0,sum(valid.logit$Churn=="Yes"))~c(0, dim(valid.logit)[1]), lty=2)
#Decile-wise Lift Chart
```

```
heights <- gain$mean.resp/mean(ifelse(valid.logit$Churn=="Yes",1,0))
midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),
                     xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")
#ROC Curve
verification::roc.plot(ifelse(valid.logit$Churn=="Yes",1,0),ifelse(valid.logit.pred1 ==
"Yes",1,0))
#Sampled
#Confusion Matrix
gmodels::CrossTable(valid.logit.pred2,valid.sam$Churn,prop.r = FALSE,prop.c =
FALSE,prop.t = FALSE,prop.chisq = FALSE)
caret::confusionMatrix(valid.logit.pred2,valid.sam$Churn,positive = "Yes")
#Lift Chart
gain <- gains(ifelse(valid.sam$Churn=="Yes",1,0), pred.prob2, groups=100)
plot(c(0,gain$cume.pct.of.total*sum(valid.sam$Churn=="Yes"))~c(0,gain$cume.obs),
     xlab="# cases", ylab="Cumulative", main="", type="l")
lines(c(0,sum(valid.sam$Churn=="Yes"))~c(0, dim(valid.sam)[1]), lty=2)
#Decile-wise Lift Chart
heights <- gain$mean.resp/mean(ifelse(valid.sam$Churn=="Yes",1,0))
midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),
                     xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")
#ROC Curve
verification::roc.plot(ifelse(valid.sam$Churn=="Yes",1,0),ifelse(valid.logit.pred2 ==
"Yes",1,0))

#------------------------
#************************

#Neural Nets

#************************
#Importing Data
churn <- read.csv("new churn.csv")
str(churn)
summary(churn)

#------------------------

#Preprocessing
library(scales)
#Deleting observations with missing values
churn <- churn[complete.cases(churn),]
#Function to change 'No Phone/Internet Service to No'
sub1 <- function(x){
  gsub("No phone service","No",x)
}
sub2 <- function(x){
  gsub("No internet service","No",x)
}
#Applying function sub to data frame
churn <- data.frame(lapply(churn, sub1))
```

```
churn <- data.frame(lapply(churn, sub2))
#Converting factor to numeric
churn$tenure <- as.numeric(as.character(churn$tenure))
churn$MonthlyCharges <- as.numeric(as.character(churn$MonthlyCharges))
churn$TotalCharges <- as.numeric(as.character(churn$TotalCharges))
#Function to convert months to years
conv <- function(x){
  x/12
}
churn$tenure <- sapply(churn$tenure,conv)
#Binning tenure
churn$tenure[churn$tenure >= 0 & churn$tenure <=1] = '0-1'
churn$tenure[churn$tenure > 1 & churn$tenure <=2] = '1-2'
churn$tenure[churn$tenure > 2 & churn$tenure <=3] = '2-3'
churn$tenure[churn$tenure > 3 & churn$tenure <=4] = '3-4'
churn$tenure[churn$tenure > 4 & churn$tenure <=5] = '4-5'
churn$tenure[churn$tenure > 5 & churn$tenure <=6] = '5-6'
churn$tenure <- as.factor(churn$tenure)
#Normalizing/Scaling columns Monthly Charges and Total Charges
#Neural Nets Normalizaing/Scaling
churn$MonthlyCharges = scales::rescale(churn$MonthlyCharges)
churn$TotalCharges = scales::rescale(churn$TotalCharges)
#Transforming MonthlyCharges by squareroot & TotalCharges by Cuberoot
#Function for cuberoot
cbrt <- function(x){
  sign(x) * abs(x)^(1/3)
}
ggplot(churn)+geom_histogram(mapping = aes(MonthlyCharges),bins = 50)
ggplot(churn)+geom_histogram(mapping = aes(TotalCharges),bins = 50)
churn$MonthlyCharges = sqrt(churn$MonthlyCharges)
churn$TotalCharges = cbrt(churn$TotalCharges)
ggplot(churn)+geom_histogram(mapping = aes(MonthlyCharges),bins = 50)
ggplot(churn)+geom_histogram(mapping = aes(TotalCharges),bins = 50)

#------------------------

#Partioning Data
#Original ratio
set.seed(123)
or <- sum(churn$Churn == "Yes")/sum(churn$Churn == "No")
churn.yes.index <- churn$Churn == "Yes"
churn.no.index <- churn$Churn == "No"
churn.yes.df <- churn[churn.yes.index,]
churn.no.df <- churn[churn.no.index,]
#Training/Validation
#Yes
train.yes.index <- sample(c(1:dim(churn.yes.df)[1]),dim(churn.yes.df)[1]/2)
train.yes.df <- churn.yes.df[train.yes.index,]
valid.yes.df <- churn.yes.df[-train.yes.index,]
#No
```

```
train.no.index <- sample(c(1:dim(churn.no.df)[1]),dim(churn.yes.df)[1]/2)
train.no.df <- churn.no.df[train.no.index,]
valid.no.df <- churn.no.df[-train.no.index,]
valid.no.index <- sample(c(1:dim(valid.no.df)[1]),(dim(train.yes.df)[1]/or))
valid.no.df <- churn.no.df[valid.no.index,]
#Combining Train/Valid
train.df <- rbind(train.yes.df,train.no.df)
valid.df <- rbind(valid.yes.df,valid.no.df)

#------------------------

#Dummy Variable for other algorithms
#m-1 dummies
#Categorical Columns & Numerical Columns
cat <- churn[,-c(1,19,20,21)]
num <- churn[,c(1,19,20,21)]
#Function to create dummy variable
dum <- function(x){
  model.matrix(~x-1,data = churn)[,-1]
}
#Creating Dummy Variables
dummy <- data.frame(sapply(cat, dum))
#Combining variables to final dataset
churn.NN <- cbind(num,dummy)
str(churn.NN)

#------------------------

#Oversampling
train.NN <- churn.NN[rownames(train.df),]
valid.NN <- churn.NN[rownames(valid.df),]
#Sampling
churn.sam <- rbind(train.NN,valid.NN)
train.index <- sample(c(1:dim(churn.sam)[1]),0.60*dim(churn.sam)[1])
train.sam <- churn.sam[train.index,]
valid.sam <- churn.sam[-train.index,]

#------------------------

#Neural Nets
library(neuralnet)
#Oversampling
train.NN <- train.NN[,-1]
valid.NN <- valid.NN[,-1]
NN_model1 <- neuralnet::neuralnet(Churn~.,data = train.NN,linear.output = FALSE,hidden
= 2)
plot(NN_model1,rep = "best")
valid.NN.pred1 <-
as.factor(ifelse(apply(neuralnet::compute(NN_model1,valid.NN)$net.result,1,which.max)-1
== 1,"Yes","No"))
```

```
pred.prob1 <- predict(NN_model1,valid.NN,type = "response")
#Sampling
train.sam <- train.sam[,-1]
valid.sam <- valid.sam[,-1]
NN_model2 <- neuralnet::neuralnet(Churn~.,data = train.sam,linear.output = FALSE,hidden
= 2)
plot(NN_model2,rep = "best")
valid.NN.pred2 <-
as.factor(ifelse(apply(neuralnet::compute(NN_model2,valid.sam)$net.result,1,which.max)-1
== 1,"Yes","No"))
pred.prob2 <- predict(NN_model2,valid.sam,type = "response")

#------------------------

#Model Performance
library(gmodels)
library(caret)
library(gains)
library(verification)
#Oversampling
#Confusion Matrix
gmodels::CrossTable(valid.NN.pred1,valid.NN$Churn,prop.r = FALSE,prop.c =
FALSE,prop.t = FALSE,prop.chisq = FALSE)
caret::confusionMatrix(valid.NN.pred1,valid.NN$Churn,positive = "Yes")
#Lift Chart
gain <- gains(ifelse(valid.NN$Churn=="Yes",1,0), pred.prob1[,2], groups=100)
plot(c(0,gain$cume.pct.of.total*sum(valid.NN$Churn=="Yes"))~c(0,gain$cume.obs),
    xlab="# cases", ylab="Cumulative", main="", type="l")
lines(c(0,sum(valid.NN$Churn=="Yes"))~c(0, dim(valid.NN)[1]), lty=2)
#Decile-wise Lift Chart
heights <- gain$mean.resp/mean(ifelse(valid.NN$Churn=="Yes",1,0))
midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),
                xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")
#ROC Curve
verification::roc.plot(ifelse(valid.NN$Churn=="Yes",1,0),ifelse(valid.NN.pred1=="Yes",1,0)
)
#Sampling
#Confusion Matrix
gmodels::CrossTable(valid.NN.pred2,valid.sam$Churn,prop.r = FALSE,prop.c =
FALSE,prop.t = FALSE,prop.chisq = FALSE)
caret::confusionMatrix(valid.NN.pred2,valid.sam$Churn,positive = "Yes")
#Lift Chart
gain <- gains(ifelse(valid.sam$Churn=="Yes",1,0), pred.prob2[,2], groups=100)
plot(c(0,gain$cume.pct.of.total*sum(valid.sam$Churn=="Yes"))~c(0,gain$cume.obs),
    xlab="# cases", ylab="Cumulative", main="", type="l")
lines(c(0,sum(valid.sam$Churn=="Yes"))~c(0, dim(valid.sam)[1]), lty=2)
#Decile-wise Lift Chart
heights <- gain$mean.resp/mean(ifelse(valid.sam$Churn=="Yes",1,0))
midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),
                xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")
```

```
#ROC Curve
verification::roc.plot(ifelse(valid.sam$Churn=="Yes",1,0),ifelse(valid.NN.pred2=="Yes",1,0)
)

#------------------------
#************************

#Linear Discriminant Analysis

#************************
#Importing Data
churn <- read.csv("new churn.csv")
str(churn)
summary(churn)

#------------------------

#Preprocessing
#Deleting observations with missing values
churn <- churn[complete.cases(churn),]
#Function to change 'No Phone/Internet Service to No'
sub1 <- function(x){
  gsub("No phone service","No",x)
}
sub2 <- function(x){
  gsub("No internet service","No",x)
}
#Applying function sub to data frame
churn <- data.frame(lapply(churn, sub1))
churn <- data.frame(lapply(churn, sub2))
#Converting factor to numeric
churn$tenure <- as.numeric(as.character(churn$tenure))
churn$MonthlyCharges <- as.numeric(as.character(churn$MonthlyCharges))
churn$TotalCharges <- as.numeric(as.character(churn$TotalCharges))
#Function to convert months to years
conv <- function(x){
  x/12
}
churn$tenure <- sapply(churn$tenure,conv)
#Standardizing columns Monthly Charges and Total Charges
churn[,c('tenure','MonthlyCharges','TotalCharges')] =
scale(churn[,c('tenure','MonthlyCharges','TotalCharges')])
#Categorical Columns & Numerical Columns
cat <- churn[,-c(1,6,19,20,21)]
num <- churn[,c(1,6,19,20,21)]
#Converting Categorical to Numerical
cat <- data.frame(lapply(cat,as.numeric))
#Combining variables to final dataset
churn.LDA <- cbind(num,cat)
str(churn.LDA)
```

```
#-------------------------

#Partioning Data
#Original ratio
set.seed(123)
or <- sum(churn$Churn == "Yes")/sum(churn$Churn == "No")
churn.yes.index <- churn$Churn == "Yes"
churn.no.index <- churn$Churn == "No"
churn.yes.df <- churn[churn.yes.index,]
churn.no.df <- churn[churn.no.index,]
#Training/Validation
#Yes
train.yes.index <- sample(c(1:dim(churn.yes.df)[1]),dim(churn.yes.df)[1]/2)
train.yes.df <- churn.yes.df[train.yes.index,]
valid.yes.df <- churn.yes.df[-train.yes.index,]
#No
train.no.index <- sample(c(1:dim(churn.no.df)[1]),dim(churn.yes.df)[1]/2)
train.no.df <- churn.no.df[train.no.index,]
valid.no.df <- churn.no.df[-train.no.index,]
valid.no.index <- sample(c(1:dim(valid.no.df)[1]),(dim(train.yes.df)[1]/or))
valid.no.df <- churn.no.df[valid.no.index,]
#Combining Train/Valid
train.df <- rbind(train.yes.df,train.no.df)
valid.df <- rbind(valid.yes.df,valid.no.df)

#-------------------------

#Oversampling
train.LDA <- churn.LDA[rownames(train.df),]
valid.LDA <- churn.LDA[rownames(valid.df),]
#Sampling
churn.sam <- rbind(train.LDA,valid.LDA)
train.index <- sample(c(1:dim(churn.sam)[1]),0.60*dim(churn.sam)[1])
train.sam <- churn.sam[train.index,]
valid.sam <- churn.sam[-train.index,]

#-------------------------

#Linear Discriminant Analysis
library(MASS)
#Oversampling
train.LDA <- train.LDA[,-1]
valid.LDA <- valid.LDA[,-1]
LDA_model1 <- MASS::lda(Churn~.,data = train.LDA)
valid.LDA.pred1 <- as.factor((predict(LDA_model1,valid.LDA)$class))
pred.prob1 <- predict(LDA_model1,valid.LDA)$posterior
#Sampling
train.sam <- train.sam[,-1]
valid.sam <- valid.sam[,-1]
```

```
LDA_model2 <- MASS::lda(Churn~.,data = train.sam)
valid.LDA.pred2 <- as.factor((predict(LDA_model2,valid.sam)$class))
pred.prob2 <- predict(LDA_model2,valid.sam)$posterior

#------------------------

#Model Performance
library(gmodels)
library(caret)
library(gains)
library(verification)
#Oversampling
#Confusion Matrix
gmodels::CrossTable(valid.LDA.pred1,valid.LDA$Churn,prop.r = FALSE,prop.c =
FALSE,prop.t = FALSE,prop.chisq = FALSE)
caret::confusionMatrix(valid.LDA.pred1,valid.LDA$Churn,positive = "Yes")
#Lift Chart
gain <- gains(ifelse(valid.LDA$Churn=="Yes",1,0), pred.prob1[,2], groups=100)
plot(c(0,gain$cume.pct.of.total*sum(valid.LDA$Churn=="Yes"))~c(0,gain$cume.obs),
     xlab="# cases", ylab="Cumulative", main="", type="l")
lines(c(0,sum(valid.LDA$Churn=="Yes"))~c(0, dim(valid.LDA)[1]), lty=2)
#Decile-wise Lift Chart
heights <- gain$mean.resp/mean(ifelse(valid.LDA$Churn=="Yes",1,0))
midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),
                     xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")
#ROC Curve
verification::roc.plot(ifelse(valid.LDA$Churn=="Yes",1,0),ifelse(valid.LDA.pred1=="Yes",1
,0))
#Sampling
#Confusion Matrix
gmodels::CrossTable(valid.LDA.pred2,valid.sam$Churn,prop.r = FALSE,prop.c =
FALSE,prop.t = FALSE,prop.chisq = FALSE)
caret::confusionMatrix(valid.LDA.pred2,valid.sam$Churn,positive = "Yes")
#Lift Chart
gain <- gains(ifelse(valid.sam$Churn=="Yes",1,0), pred.prob2[,2], groups=100)
plot(c(0,gain$cume.pct.of.total*sum(valid.sam$Churn=="Yes"))~c(0,gain$cume.obs),
     xlab="# cases", ylab="Cumulative", main="", type="l")
lines(c(0,sum(valid.sam$Churn=="Yes"))~c(0, dim(valid.sam)[1]), lty=2)
#Decile-wise Lift Chart
heights <- gain$mean.resp/mean(ifelse(valid.sam$Churn=="Yes",1,0))
midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),
                     xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")
#ROC Curve
verification::roc.plot(ifelse(valid.sam$Churn=="Yes",1,0),ifelse(valid.LDA.pred2=="Yes",1,
0))

#------------------------
#************************

#Support Vector Machine
```

```r
#************************
#Importing Data
churn <- read.csv("new churn.csv")
str(churn)
summary(churn)

#------------------------

#Preprocessing
#Deleting observations with missing values
churn <- churn[complete.cases(churn),]
#Function to change 'No Phone/Internet Service to No'
sub1 <- function(x){
  gsub("No phone service","No",x)
}
sub2 <- function(x){
  gsub("No internet service","No",x)
}
#Applying function sub to data frame
churn <- data.frame(lapply(churn, sub1))
churn <- data.frame(lapply(churn, sub2))
#Converting factor to numeric
churn$tenure <- as.numeric(as.character(churn$tenure))
churn$MonthlyCharges <- as.numeric(as.character(churn$MonthlyCharges))
churn$TotalCharges <- as.numeric(as.character(churn$TotalCharges))
#Function to convert months to years
conv <- function(x){
  x/12
}
churn$tenure <- sapply(churn$tenure,conv)
#Binning tenure
churn$tenure[churn$tenure >= 0 & churn$tenure <=1] = '0-1'
churn$tenure[churn$tenure > 1 & churn$tenure <=2] = '1-2'
churn$tenure[churn$tenure > 2 & churn$tenure <=3] = '2-3'
churn$tenure[churn$tenure > 3 & churn$tenure <=4] = '3-4'
churn$tenure[churn$tenure > 4 & churn$tenure <=5] = '4-5'
churn$tenure[churn$tenure > 5 & churn$tenure <=6] = '5-6'
churn$tenure <- as.factor(churn$tenure)
#Standardizing columns Monthly Charges and Total Charges
churn[,c('MonthlyCharges','TotalCharges')] =
scale(churn[,c('MonthlyCharges','TotalCharges')])

#------------------------

#Partioning Data
#Original ratio
set.seed(123)
or <- sum(churn$Churn == "Yes")/sum(churn$Churn == "No")
churn.yes.index <- churn$Churn == "Yes"
```

```
churn.no.index <- churn$Churn == "No"
churn.yes.df <- churn[churn.yes.index,]
churn.no.df <- churn[churn.no.index,]
#Training/Validation
#Yes
train.yes.index <- sample(c(1:dim(churn.yes.df)[1]),dim(churn.yes.df)[1]/2)
train.yes.df <- churn.yes.df[train.yes.index,]
valid.yes.df <- churn.yes.df[-train.yes.index,]
#No
train.no.index <- sample(c(1:dim(churn.no.df)[1]),dim(churn.yes.df)[1]/2)
train.no.df <- churn.no.df[train.no.index,]
valid.no.df <- churn.no.df[-train.no.index,]
valid.no.index <- sample(c(1:dim(valid.no.df)[1]),(dim(train.yes.df)[1]/or))
valid.no.df <- churn.no.df[valid.no.index,]
#Combining Train/Valid
train.df <- rbind(train.yes.df,train.no.df)
valid.df <- rbind(valid.yes.df,valid.no.df)

#------------------------

#Dummy Variable for other algorithms
#m-1 dummies
#Categorical Columns & Numerical Columns
cat <- churn[,-c(1,19,20,21)]
num <- churn[,c(1,19,20,21)]
#Function to create dummy variable
dum <- function(x){
  model.matrix(~x-1,data = churn)[,-1]
}
#Creating Dummy Variables
dummy <- data.frame(sapply(cat, dum))
#Combining variables to final dataset
churn.SVM <- cbind(num,dummy)
str(churn.SVM)

#------------------------

#Oversampling
train.SVM <- churn.SVM[rownames(train.df),]
valid.SVM <- churn.SVM[rownames(valid.df),]
#Sampling
churn.sam <- rbind(train.SVM,valid.SVM)
train.index <- sample(c(1:dim(churn.sam)[1]),0.60*dim(churn.sam)[1])
train.sam <- churn.sam[train.index,]
valid.sam <- churn.sam[-train.index,]

#------------------------

#Support Vector Machine
library(e1071)
```

```
#Oversampling
train.SVM <- train.SVM[,-1]
valid.SVM <- valid.SVM[,-1]
SVM_model1 <-  e1071::svm(Churn~.,data = train.SVM)
valid.SVM.pred1 <- as.factor(predict(SVM_model1,valid.SVM))
#Sampling
train.sam <- train.sam[,-1]
valid.sam <- valid.sam[,-1]
SVM_model2 <-  e1071::svm(Churn~.,data = train.sam)
valid.SVM.pred2 <- as.factor(predict(SVM_model2,valid.sam))

#-------------------------

#Model Performance
library(gmodels)
library(caret)
library(verification)
#Oversampling
#Confusion Matrix
gmodels::CrossTable(valid.SVM.pred1,valid.SVM$Churn,prop.r = FALSE,prop.c =
FALSE,prop.t = FALSE,prop.chisq = FALSE)
caret::confusionMatrix(valid.SVM.pred1,valid.SVM$Churn,positive = "Yes")
#ROC Curve
verification::roc.plot(ifelse(valid.SVM$Churn=="Yes",1,0),ifelse(valid.SVM.pred1=="Yes",
1,0))
#Sampling
#Confusion Matrix
gmodels::CrossTable(valid.SVM.pred2,valid.sam$Churn,prop.r = FALSE,prop.c =
FALSE,prop.t = FALSE,prop.chisq = FALSE)
caret::confusionMatrix(valid.SVM.pred2,valid.sam$Churn,positive = "Yes")
#ROC Curve
verification::roc.plot(ifelse(valid.sam$Churn=="Yes",1,0),ifelse(valid.SVM.pred2=="Yes",1,
0))

#-------------------------
```