**Classifying the Users as Spenders and Non-Spenders**

**AIM**:

To know whether the Users while gaming, will potentially spend money or not based on their activities in the game.

**ABSTRACT**:

We perform classification of Users as Spending Users vs Non Spending Users using 3 machine learning techniques.

***DATA***: It was based on the attributes we gathered from attacks.txt, User\_data.txt and alliance chat messages. We focused our calculation for total of 5387 users who were actively fighting in attacks.txt as well as chatting in the alliance chat rooms.

The following are the evaluation metrics by which we evaluate the classifiers.

***Precision*** is the number of Spenders correctly classified to the total number of Users Classified as “Spenders” by the Classifier. In other words, For Logistic Regression, Out of all the Users who are classified as “Spenders”, ***85.9% of them are actual Spenders***.

***Recall*** is the number of Spenders correctly Classified to the total number of actual Spenders. Given a 100 actual spenders in the Data, the Recursive Partition Tree can capture up to 80 of them.

***F1Score:***  It is a measure of classifier’s Accuracy. F1 score is a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Classifier | Description | Precision | Recall | F1 Score |
| Recursive Partition Tree | CART - Decision Tree. | **85.33** | **80.86** | **83.034** |
| Logistic Regression | A simple sigmoid classifier | **85.962** | **77.44** | **81.015** |
| Support Vector Machine with RBF Kernel | Maximum margin classifier. | **81.45** | **89.06** | **83.41** |

**PURPOSE**:

1. The Gamers who spend money on games are our treasures.
2. We need to identify those potential spenders from our gamers to capture the future VIP customers.
3. We have to make sure these potential Spenders are having an enhanced and enriched gaming experience.
4. We need to know if we had made them feel the worth of the money they spent and they are encouraged to pay more. We need to understand their impressions, ideas, likes and dislikes of our game when they play.
5. The interaction logs obtained from the alliance chat rooms give a peek at what the players are communicating to each other.
6. By analyzing the relationship between the User expenditure and the User social interaction gives us a notion of what they feel about our games. Do they want to sustain the relationship/business with us? Do they find their time and money are worth spent? Do they feel excited for our new games /levels/changes/. Are they encouraged to spend more and buy more games from us?

**THEORY:**

**SUPERVISED LEARNING:**

Given is a set of input data, X (set of Independent variables) and its corresponding Output data, Y (Dependent variable). In our problem X corresponds to gamer’s attributes (X) and the Y corresponds to whether he is a spender or not. This set of input data with known output is called ***Training set***. Our goal is, given a training set, to learn a function so that is a good predictor for the corresponding value of Y. The function **H** is called a ***hypothesis***. Supervised learning is the task of creating a mapping function from labeled input ***training*** data to Output. This mapping function can be used to predict Y’ for the new input X’.

**X**

**XX**

**LINEAR REGRESSION**:

To perform supervised learning, we need to represent the hypothesis H which maps the input to the output. Let us start designing the hypothesis as a linear function of input X. X is given by a set of independent variables.

Here, ’s are the parameters (also called weights) parameterizing the space of linear functions mapping from X to Y. From the training set of data, we obtain the values of these weights. Using the training data, the ’s are updated from an arbitrary initial value till convergence based on the ***Euclidean distance*** between true value(Y) and the predicted value . Θ, X represents the corresponding vectors forms of parameters and the input vector. We predict the values of Y using The regression assumes that the error between the hypothesis and the actual output follows Normal Distribution.

**CLASSIFICATION:**

When **Y** can take on only a small number of discrete values, we call it a Classification Problem. In other words, given a training set of input and output data, an inference function is created to map the input to the discrete output values using a learning algorithm, called as ***Classification algorithm.*** Inference function is called as a ***Classifier***. More formally, In a Classification problem, we have a training sample of ***N observations*** on a **class variable** that takes values , and  ***predictor variables****,* ,. Our goal is to find a model for predicting the values of Y from new values. In theory the solution is simply a partition of the X space into disjoint sets, ,, such that

For our problem, we will look into classification task for binary output variable (k =2 ).

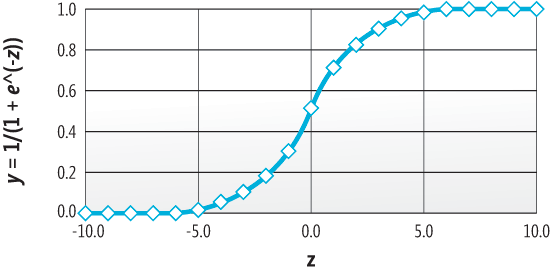
We will look at three learning algorithms.

**LOGISTIC REGRESSION:**

Logistic regression, also called a logit model, is used to model binary outcome variables (Spender or not) for the given set of input variables or predictor variables. In the logit model, the ***log hypothesis of the outcome*** is modeled as a ***linear combination*** of the predictor variables. Let P(Y) is the probability that Y will occur, where Y is a binary object (Y/N, 1/0, true/false). Then

We define the hypothesis through a ***sigmoid function*** instead of a simple linear combination of input vectors.

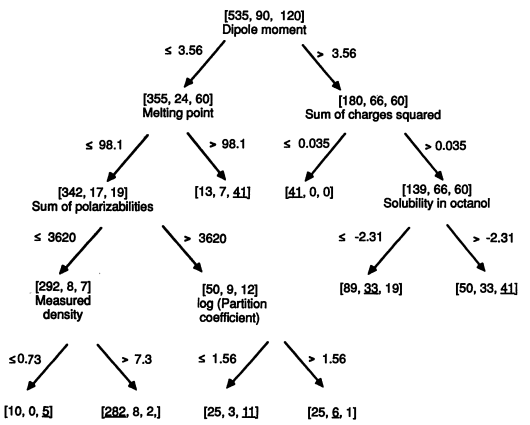
The sigmoid function simply separates the data into two halves the positive and the negative sides of the outcome variable. Using the training data, the is obtained by maximize the log likelihood of the outcome. The error between the hypothesis and the actual output is assumed to follow a ***Bernoulli distribution***. We minimize the error to maximize the likelihood of the outcome. Below is the Sigmoid curve which transcends from 0 to 1 for a given integer input.



Please refer wiki for more details. Let us see the next type of classification Algorithm.

**CLASSIFICATION AND REGRESSION TREE (CART) & RECURSIVE PARTITION (RPART):**

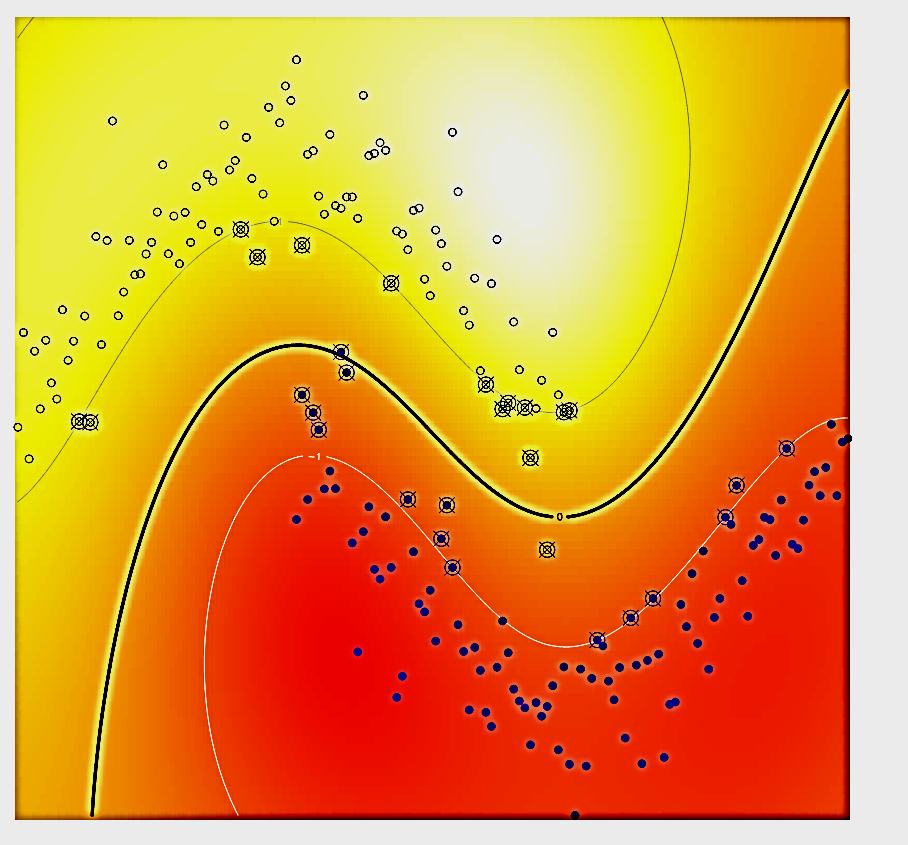
Classification and regression trees are machine-learning methods for constructing prediction models from data. The models are obtained by recursively partitioning the data space and fitting a simple prediction within each partition. It resembles more like a decision tree. Classification trees are designed for dependent variables that take continuous or ordered discrete values, with prediction error typically measured by the squared difference between the observed and predicted values. Classification tree methods yield rectangular sets by recursively partitioning the data set one variable at a time. CART is implemented in R using a **Recursive Partition (RPART)** module. Below is a picture showing how decision trees look like. The ***rpart*** programs build classification or regression models of a very general structure using a two stage procedure; the resulting models can be represented as binary trees. **Rpart** recursively perform univariate splits of the dependent variable based on values on a set of covariates. Rpart employ information measures (such as the ***Gini coefficient***) for selecting the current covariate.



Let us look at the third Classification Algorithm, Support Vector Machines.

**SUPPORT VECTOR MACHINE (SVM):**

Support Vector Machines (SVM’s) are a relatively new learning method used for binary classification. The basic idea is to find a ***hyperplane*** which separates the ***d-dimensional data perfectly into its two classes.***  The notion is to separate the data using a decision surface (Hyperplane) with as maximum “***Margin***”{the distance between the nearest data point to the decision boundary”}.However, since example data is often not linearly separable, SVM’s introduce the notion of a “***kernel induced feature space***” which casts the data into a higher dimensional space where the data is separable. This method of construction necessarily means that the decision function for an SVM is fully specified by a (usually small) subset of the data which defines the position of the separator. These points are referred to as the ***support vectors*** (in a vector space, a point can be thought of as a vector between the origin and that point). SVM’s are intuitive, theoretically well- founded, and have shown to be practically successful. SVM’s have also been extended to solve regression tasks (where the system is trained to output a numerical value, rather than “yes/no” classification).



**SUPPORT VECTORS**

**DATA POINTS**

**DECISION BOUNDARY**

**DATA**:

1. *PRIMARY*: User\_data.txt, attacks.txt and alliance\_chat.txt.

2. *DERIVED DATA*: They are secondary data derived from the primary sources.

* User Performance Metrics: User scores, Match Results, Alliance association, match frequencies.
* Alliance Performance Metrics: Alliance Scores, Match results.
* Knight Performance Metrics: Knight Scores, Match results, match frequencies.
* User Social Metrics: User Sentiments, User subjectivity, chat counts.
* User Delight Metrics: Loots, Boosts, Mights, Wall Strengths & XPs.

**DATA DISTRIBUTION:**

* We used data from User\_data to identify Spenders and non-spenders.
* Out of **10215693** User Ids who are registered in User\_data, only **506,707** are the number of User id correspond to Users who spend money on games.
* Very few of the User’s attack activity are given by the two months data of attacks.txt namely **41621** Defending users and **2261** attacking Users**. 2168** users are both Defenders and attackers.
* The alliance\_chat.txt holds only messages from **6069** distinct Users.
* We compiled a list of **5387** Users from this bunch and used it for Classification of Users.

**CLASSIFICATION TASK**: To determine whether the given gamer is likely to spend money on our games or not based on the gamer’s data.

**TRAINING AND TESTING DATA**:

We split 4000 values from the 5387 User data for training the claissifiers.

The remaining 1387 users are for testing.

**FEATURE SET**: The following are the feature set we choose the data in order to determine whether the User is potential Spender or Not.

1. *UserID*: Userid is an increasing number. Usually IDs are not used as feature set. But surprisingly there is a strong correlation between this number and the user’s behavior for this data set.
2. *LoginCount:* the number of logins made by the user shows how interested he is towards gaming.
3. *UserWinCount*: The number of matches won by the User.
4. *UserLostCount:* The number of matches lost by the User.
5. *UserDrawCount:* the number of matches drawn by the User.
6. *UserDefenseMight*: the scaled might of the User while defending in matches.
7. *UserAttackMight*: The scaled might of the user while he is attacking.
8. *UserDefenseScore*: The Score indicating the defense strength of the user.
9. *UserAttackScore*: The Score indicating the attack strength of the user.
10. *UserAvgDefenceScore*: The average defense score of the User across the matches.
11. *UserAvgAttackScore*: the average attacks score of the User across the matches.
12. *UserKnightWin*: Number of matches won by the user as knight.
13. *UserKnightLost*: Number of matches lost by the user as a knight.
14. *UserKnightDrawCount*: Number of matches drawn by the user as Knight.
15. *UserKnightDef*: The defense score of the user as a knight.
16. *UserKnightAtk*: The attack score of the user as knight.
17. *UserScore* : DefenceScore+AttackScore+WinCount+LostCount+ 0.5\*B.DrawCount+ KnightDefScore+ KnightAtkScore
18. *num\_alliances*: Number of alliances associated by the user.
19. *AllianceAvgWin*: Average number of matches won by the alliance per user.
20. *AllianceTotalWin*: Total number of matches won by the alliances.
21. *AllianceAvgLost*: Average number of matches lost by the alliances of the user.
22. *AllianceTotalLost*:Total number of matches lost by the alliances of the user.
23. *AllianceTotalDraw*: Total number of matches drawn by the alliances of the user.
24. *AllianceAvgDraw:* Average number of matches drawn by the alliances of the user.
25. *AllianceSumDefense:* Sum of Defense scores of the users in the alliance.
26. *AllianceSumAttack*: Sum of Attack scores of the users in the alliance.
27. *AvgAtkPerDay*: Average number of attacks done by the user per day.
28. *AvgDefPerDay*:Average number of defense done by the user per day.
29. *TotalSentiment*: Sum of all sentiments of the user’s chat messages.
30. *AvgSentiment*: Avearage of all sentiments of the user’s chat messages.
31. *TotalSubjectivity*: Sum of all subjectivity scores of the user’s chat messages.
32. AvgSubjectivity:Average of all subjectivity scores of the user’s chat messages.
33. *Msgcount*: Number of chat messages per day by a user.
34. *DailyAvgWordCount*: Daily number of words given by the user in the chat messages.
35. *SumAtkBoost*:Sum of all attack boosts taken by the user.
36. *AvgAtkBoost*: Average of all attack boosts taken by the user.
37. *SumDefBoost*: sum of all defense boosts taken by the user.
38. *AvgDefBoost*: Average of all defense boosts taken by the user.
39. *TotalWall:* Sum of all wall strengths of the defending user during matches.
40. *AvgWall*: Average of all wall strengths of the defending user during matches.
41. *SumXP*:Sum of all XP earned by the user in matches.
42. *AvgLoot*: Average of all the Loot of resources{ Gold , Food, Wood, Ore, Stone}
43. *SumLoot*: Sum of all the loot of the resources{ gold, Food, Wood, Ore, Stone}
44. **Spender : Truth data which consists of ‘1’ if the user has spent money on game and ‘0’ if he has not.**

Machine Learning Language used**: R**

**Data Analysis:**

In order to find the most important features to consider to predict whether the user is a spender or not, we used correlation coefficient between the features and the Spender Truth data.

Correlation Coefficient,Mutual Information with Spender

FEATURES CORRELATION COEFF MUTUAL INFORMATION

userID -0.430595390 0.5206974

LoginCount 0.220187766 0.3891212

UserWinCount 0.116691019 0.0982105

UserLostCount 0.215442359 0.09188361

UserDrawCount 0.164402704 0.1422602

UserDefenseMight 0.007637920 0.002634197

UserAttackMight 0.196327842 0

UserDefenseScore 0.142411209 0.1180316

UserAttackScore -0.081272767 0.1814049

UserAvgDefenceScore 0.065963149 0.004224291

UserAvgAttackScore -0.042212112 0.01618479

UserKnightWin 0.015817044 0.003781667

UserKnightLost 0.015287568 0.0006701847

UserKnightDrawCount 0.030975938 0.002315676

UserKnightDef 0.010530917 0.0003902789

UserKnightAtk -0.024648546 0.003710424

UserScore 0.008783527 0.5206974

num\_alliances 0.138909230 0.02592823

AllianceAvgWin 0.242545697 0.1931967

AllianceTotalWin 0.256607318 0.207764

AllianceAvgLost 0.246871409 0.1797478

AllianceSumLost 0.258180983 0.2162882

AllianceSumDraw 0.255755568 0.2294382

AllianceAvgDraw 0.245933942 0.2063902

AllianceSumDefense 0.101966293 0.2592428

AllianceSumAttack -0.164388208 0.2303018

AvgAtkPerDay 0.209129205 0.0536823

AvgDefPerDay -0.012271459 0.04072304

TotalSentiment 0.181493431 0.3491399

AvgSentiment 0.200167131 0.3766588

TotalSubjetivity 0.161756718 0.3512205

AvgSubjectivity 0.264362487 0.3806916

msgcount 0.143144316 0.1241952

DailyAvgTalkCount 0.141644458 0.0302807

SumAtkBoost 0.049099204 0.01325526

AvgAtkBoost 0.078870813 0

SumDefBoost 0.044089424 0.01189367

AvgDefBoost 0.083525219 0

TotalWall 0.093853515 0.09332666

AvgWall 0.105515404 0.0201931

SumXP 0.087610762 0.06869

AvgLoot 0.081407779 0.007526336

SumLoot 0.099987485 0.1173244

We added quadratic features by hardcoding combinations of features to improve the accuracy of models.

**LOGISTIC REGRESSION:**

**Logistic Regression:** We implemented Logistic Regression using R by using a subset of the above feature set.

Below is the summary of the Classifier:

***Deviance*** is a measure of how different is the predicted output compared to the true output.

***Deviance Residuals*** show the distribution of the residual error among the data points.

***Columns Estimate, Std.Error, Z value and Pr(>|z|)*** specify how appropriate are the coefficients for each of the feature we have chosen. Stars show how significant are the coefficients using Z test.

***The Null Deviance*** specify how much deviance will the predicted data produce from the output if only the Intercept is used for predicting the output.

***Residual Deviance*** specify how much deviance will the predicted data produce from the output if features and the Intercept are used for predicting the output. The better the difference between the Residual Deviance and Null Deviance, the better the classifier.

**Degrees of Freedom:** In statistics, the number of degrees of freedom is the number of values in the final calculation of a statistic that are free to vary.

**AIC:** The Akaike information criterion (AIC) is a measure of the relative quality of a statistical model for a given set of data. The higher the better.

Call:

glm(formula = Spender ~ userID + LoginCount + UserAttackScore \* UserDrawCount +

UserAttackScore \* UserLostCount + UserLostCount + UserDrawCount + UserAttackMight +

UserDefenseScore + UserAttackScore + UserAvgDefenceScore +UserAvgAttackScore +

UserKnightWin + UserKnightLost + UserKnightDrawCount + UserKnightDef + UserScore +

AvgSubjectivity \* AvgSubjectivity +AllianceTotalWin + AllianceAvgWin +

AllianceAvgLost + +AllianceSumDraw + AllianceAvgDraw + AllianceSumDefense +

UserAttackScore \* UserAttackScore + +AllianceSumAttack + AvgAtkPerDay + AvgAtkPerDay

\* AvgAtkPerDay +AvgSubjectivity + msgcount + UserAttackScore \* UserKnightLost +

AvgAtkBoost +SumXP + SumDefBoost + AvgDefBoost + TotalWall + AvgWall \*AvgWall

+AvgWall + AvgLoot + SumLoot, family = binomial,data = train)

Deviance Residuals: Min 1Q Median 3Q Max

-3.6322 0.0930 0.3525 0.5973 1.6205

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) 7.423e-01 2.345e-01 3.166 0.00155 \*\*

userID -1.969e-07 1.755e-08 -11.219 < 2e-16 \*\*\*

LoginCount 8.674e-05 1.593e-05 5.446 5.15e-08 \*\*\*

UserAttackScore -3.843e-04 1.511e-04 -2.544 0.01095 \*

UserDrawCount -1.804e-03 4.241e-04 -4.254 2.10e-05 \*\*\*

UserLostCount 5.628e-03 1.868e-03 3.014 0.00258 \*\*

UserAttackMight -1.506e-01 2.149e-01 -0.701 0.48342

UserDefenseScore 5.199e-04 4.175e-04 1.245 0.21306

UserAvgDefenceScore 2.227e-02 4.068e-02 0.547 0.58416

UserAvgAttackScore -7.137e-03 2.522e-02 -0.283 0.77716

UserKnightWin 1.942e-02 2.599e-02 0.747 0.45500

UserKnightLost -2.631e-01 1.614e-01 -1.630 0.10311

UserKnightDrawCount 4.597e-02 4.592e-02 1.001 0.31686

UserKnightDef 2.656e-02 6.131e-02 0.433 0.66485

UserScore -1.559e-05 3.465e-05 -0.450 0.65283

AvgSubjectivity 3.076e-04 4.065e-05 7.567 3.83e-14 \*\*\*

AllianceTotalWin 9.450e-06 1.226e-05 0.771 0.44085

AllianceAvgWin -1.136e-05 2.815e-05 -0.404 0.68648

AllianceAvgLost 1.208e-04 7.346e-05 1.645 0.10002

AllianceSumDraw -1.077e-05 7.955e-06 -1.354 0.17579

AllianceAvgDraw 1.679e-05 1.977e-05 0.849 0.39567

AllianceSumDefense 6.731e-05 8.940e-05 0.753 0.45146

AllianceSumAttack -1.002e-04 1.202e-04 -0.834 0.40420

AvgAtkPerDay 4.727e-02 8.572e-03 5.515 3.49e-08 \*\*\*

msgcount 2.953e-04 1.380e-04 2.141 0.03228 \*

AvgAtkBoost 1.389e+01 9.427e+00 1.474 0.14056

SumXP -3.460e-05 3.019e-05 -1.146 0.25174

SumDefBoost 1.207e-02 2.491e-02 0.485 0.62797

AvgDefBoost 2.122e+01 2.186e+01 0.971 0.33161

TotalWall -1.140e-05 1.792e-05 -0.636 0.52449

AvgWall 3.159e-03 2.210e-03 1.429 0.15292

AvgLoot -1.685e-02 8.620e-03 -1.955 0.05059 .

SumLoot 2.862e-05 3.406e-05 0.840 0.40075

UserAttackScore:UserDrawCount -5.117e-08 5.749e-08 -0.890 0.37339

UserAttackScore:UserLostCount -1.792e-07 8.414e-07 -0.213 0.83138

UserAttackScore:UserKnightLost 3.741e-04 2.300e-04 1.627 0.10384

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Null deviance: 3550.3 on 3999 degrees of freedom

Residual deviance: 2784.8 on 3964 degrees of freedom

AIC: 2856.8 Number of Fisher Scoring iterations: 10

**Analysis of Deviance Table:**  The function will show the change in deviance obtained by adding each of the terms in the order listed in the model formula. Every step shows an improvement in the Residual Deviance and the total degrees of Freedom.

Analysis of Deviance Table Model: binomial, link: logit

Response: Spender Terms added sequentially (first to last)

Df Deviance Resid. Df Resid. Dev

NULL 3999 3550.3

userID 1 312.943 3998 3237.4

LoginCount 1 66.042 3997 3171.4

UserAttackScore 1 7.316 3996 3164.0

UserDrawCount 1 68.478 3995 3095.6

UserLostCount 1 46.239 3994 3049.3

UserAttackMight 1 3.705 3993 3045.6

UserDefenseScore 1 9.300 3992 3036.3

UserAvgDefenceScore 1 1.403 3991 3034.9

UserAvgAttackScore 1 2.210 3990 3032.7

UserKnightWin 1 0.222 3989 3032.5

UserKnightLost 1 1.444 3988 3031.0

UserKnightDrawCount 1 3.717 3987 3027.3

UserKnightDef 1 0.567 3986 3026.8

UserScore 1 1.268 3985 3025.5

AvgSubjectivity 1 115.897 3984 2909.6

AllianceTotalWin 1 12.567 3983 2897.0

AllianceAvgWin 1 25.576 3982 2871.4

AllianceAvgLost 1 2.894 3981 2868.6

AllianceSumDraw 1 0.372 3980 2868.2

AllianceAvgDraw 1 0.884 3979 2867.3

AllianceSumDefense 1 0.549 3978 2866.7

AllianceSumAttack 1 0.504 3977 2866.2

AvgAtkPerDay 1 45.804 3976 2820.4

msgcount 1 7.880 3975 2812.6

AvgAtkBoost 1 9.996 3974 2802.6

SumXP 1 3.232 3973 2799.3

SumDefBoost 1 4.749 3972 2794.6

AvgDefBoost 1 2.224 3971 2792.4

TotalWall 1 0.195 3970 2792.2

AvgWall 1 0.489 3969 2791.7

AvgLoot 1 3.271 3968 2788.4

SumLoot 1 0.795 3967 2787.6

UserAttackScore:UserDrawCount 1 0.778 3966 2786.8

UserAttackScore:UserLostCount 1 0.054 3965 2786.8

UserAttackScore:UserKnightLost 1 1.941 3964 2784.8

Metrics of Measuring Accuracy:

**Confusion Matrix**:  contingency table or an error matrix, is a specific table layout that allows visualization of the performance of an algorithm

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | TESTING\_TRUTH | |
|  |  | Non-Spender | Spender |
| TEST\_PREDICTION | Non-Spender | 398(TN) | 198(FN) |
| Spender | 111 (FP) | 680 (TP) |

TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives.

**Precision**: It is defined as the ratio of number of positive values correctly classified to the total number of positive values. In our problem, it is the ratio of number of Users correctly classified as Spenders to the total number of Users classified as Spenders.

**True Positive Rate or Recall**: It is the ratio of number of positive values correctly classified vs the total number of positive values. In our problem, it is the ratio of number of Users correctly classified as Spenders vs the total number of Spenders.

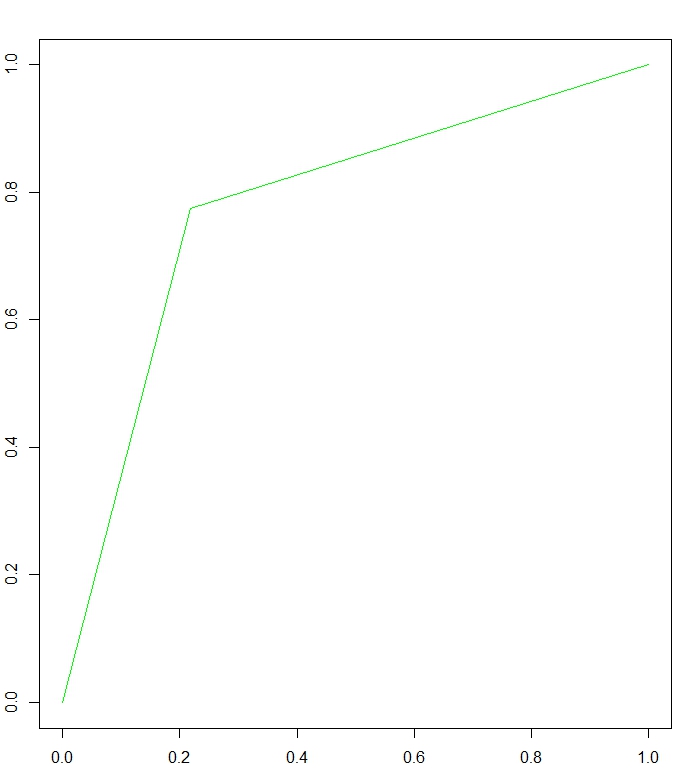
**Accuracy**: It is the ratio of total number of correctly classified items to the total number of items classified. In our problem, it corresponds to number of Users correctly classified as Spenders and Non-Spenders to the total number of users classified.

**F1 Score**: It is a measure of classifier’s Accuracy. F1 score is a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0.

**Specificity**: It is the proportional of correctly classified negative values to the total number of negative items.

**False Positive Rate or Fall-out:** It is the proportion of incorrectly classified negative values to the total number of negative items. It is simply 1-specificity.

**ROC Curve:** In signal detection theory, a ***Receiver Operating Characteristic*** (ROC), or simply ROC curve, is a graphical plot which illustrates the performance of a binary classifier system as its discrimination threshold is varied. It is created by plotting the fraction of true positives out of the total actual positives (**Recall**) vs. the fraction of false positives out of the total actual negatives (**Fall Out**), at various threshold settings.



(FPR =0.21, TPR= 0.77)

**TRUE**

**POSITIVE**

**RATE (Recall)**

**FALSE POSITIVE RATE (Fall Out)**

**CLASSIFICATION AND REGRESSION TREE USING RPART:**

We performed decision tree classification using rpart package from R.

rpart(formula = Spender ~ UserWinCount + UserLostCount + UserDrawCount +

UserAttackMight + UserDefenseScore + UserAttackScore + UserAvgDefenceScore +

UserKnightLost + UserKnightDrawCount + UserKnightDef + UserKnightAtk +

num\_alliances + AllianceTotalWin + AllianceAvgLost + AllianceSumDraw +

AllianceAvgDraw + AllianceSumDefense + AllianceSumAttack +

AvgAtkPerDay + TotalSubjetivity + AvgSubjectivity + msgcount +

SumAtkBoost + AvgAtkBoost + SumXP + userID + LoginCount +

AvgDefBoost + AvgWall + SumXP + AvgLoot + SumLoot, data = train,

control = rpart.control(xval = 10, minbucket = 35, cp = 0.00015))

***Xval Cross Validation***: It is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set.

**Minbucket:** the minimum number of observations in any leaf node in the decision tree.

**CP:Complexity Parameter:** Any split that does not decrease the overall lack of fit by a factor of CP is not attempted.

CP nsplit rel error xerror xstd

1 0.087692308 0 1.0000000 1.0000000 0.03589515

2 0.049230769 1 0.9123077 0.9538462 0.03521363

3 0.024615385 2 0.8630769 0.8953846 0.03430869

4 0.004615385 3 0.8384615 0.8676923 0.03386279

5 0.001923077 5 0.8292308 0.8907692 0.03423517

6 0.001538462 9 0.8215385 0.8953846 0.03430869

7 0.000150000 14 0.8138462 0.9030769 0.03443054

The following are the Predictive variables and their importance

Variable importance

LoginCount 45

userID 9

AvgSubjectivity 8

UserLostCount 6

TotalSubjetivity 4

AllianceAvgLost 3

AllianceAvgDraw 3

UserDefenseScore 3

AllianceSumDraw 3

UserDrawCount 3

SumXP 2

msgcount 2

SumLoot 2

AvgAtkPerDay 2

AllianceTotalWin 2

UserWinCount 1

AllianceSumAttack 1

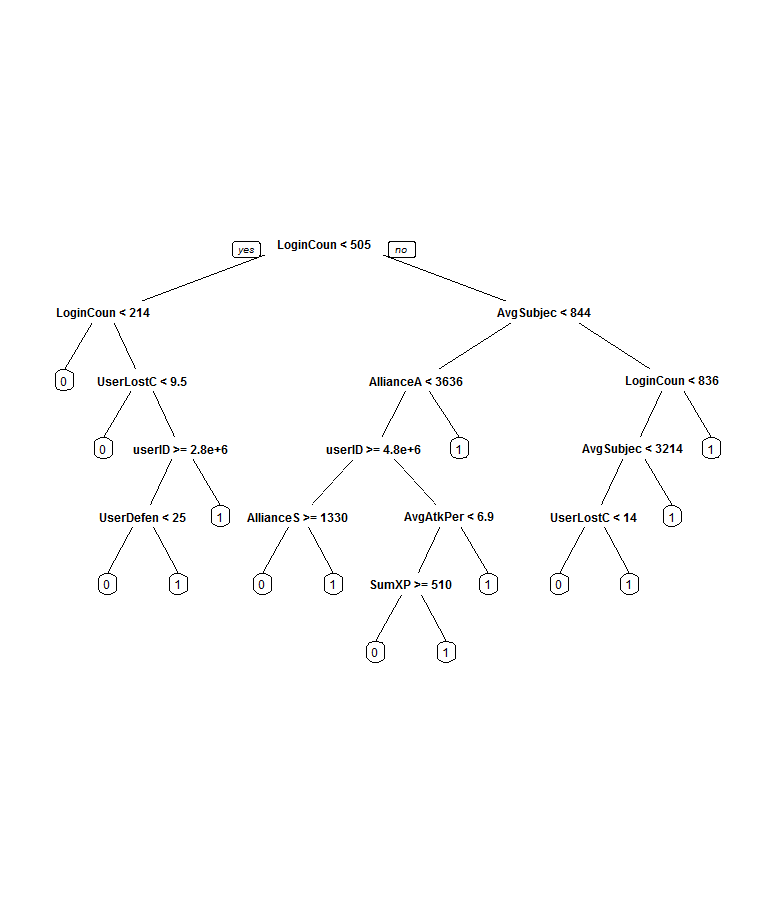
UserAttackScore 1

UserAttackMight 1

AvgLoot 1

Below is the Decision tree which shows the division performed by the model to classify the users into **Spender (1) and Non-Spender (0).**

**DECISION TREE FOR RECURSIVE PARTITION TREE**



**Metrics of Performance for RPART Tree Classifier:**

**Confusion Matrix**:  contingency table or an error matrix, is a specific table layout that allows visualization of the performance of an algorithm

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | TESTING\_TRUTH | |
|  |  | Non-Spender | Spender |
| TEST\_PREDICTION | Non-Spender | 387(TN) | 168(FN) |
| Spender | 122 (FP) | 710 (TP) |

TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives.

|  |  |  |
| --- | --- | --- |
| **Metric** | **Formula** | **Value** |
| Precision | TP/(TP+FP) | 85.33% |
| True Positive Rate or Recall | TP/(TP+FN) | 80.86% |
| Accuracy |  | 79.09% |
| F1 Score |  | 83.034% |
| Specificity |  | 76.03% |
| False Positive Rate or Fall-Out |  | 23.96% |

**ROC Curve for Recursive Partition Tree:**

**CONCLUSION:**

There is a ***NO strong/conclusive relationship*** between the Location and the User Performance Parameters. Correlation Coefficients are negligible almost showing that they are almost independent. The ***Kologomorov smirnoff*** tests between the samples are ***inconclusive***. The Mutual Information shows a *weak positive information share* between ***Draw count, Lose count and Defense score on the Location.***

**REFERENCES:**

1. http://nlp.stanford.edu/IR-book/html/htmledition/support-vector-machines-the-linearly-separable-case-1.html
2. <http://www.scholarpedia.org/article/Mutual_information>
3. <http://stattrek.com/statistics/correlation.aspx>
4. <http://data.princeton.edu/R/glms.html>
5. <http://www.stat.wisc.edu/~loh/treeprogs/guide/wires11.pdf>