Predictive Analytics Modeling
Methodology Document

Campaign Response Modeling

17 – October 2012
Version details

<table>
<thead>
<tr>
<th>Version number</th>
<th>Date</th>
<th>Author</th>
<th>Reviewer name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>16 – October- 2012</td>
<td>Vikash chandra</td>
<td></td>
</tr>
</tbody>
</table>
1. Training Objective

This manual is written principally for new analysts in the Analytics Team at DAS. The subject matter deals expressly with the development of custom marketing analytics models, although much of the material may be applied to other types of model developments as well.

- The course is designed to introduce and familiarize the user with the concepts and process of Predictive Modeling.
- The course will try to give you a realistic perspective of how to do modeling.
- The course lays down the framework on Predictive Modeling in general.
- The successful completion of the course will make you comfortable in handling modeling projects.

We hope that you will be able to leverage our experiences to raise your effectiveness in preparation for your study.

1.1 Process flow:

Here is a high-level project process flow diagram.

![Process Flow Diagram](image-url)
1.2 **Points to remember**

The tool considered for this methodology training is ‘R’. Some of the important pointers to remember are:

- R is case sensitive. (The word “mean” is not same as “MEAN”)
- R allows the = sign to be used for object assignments, however default object for assignment is "<-"
- Comments are preceded by the # symbol
- Need some specific package to run certain commands.

2. **Data Input**

The first task in a project is to import the client data into the R tool. R provides a wide range options for importing data as mentioned below:

2.1 **Reading Data from CSV**

```r
# read comma separated file into memory
> data <- read.csv("C:/Documents and Settings/MyDocuments/modelling.csv")
```

2.2 **Reading Data fromExcel**

```r
# read Excel file into memory
> library(RODBC)
> conn <- odbcConnectExcel("myfile.xls")
> data <- sqlFetch(channel, "mysheet")
> odbcClose(conn)
```

2.3 **Reading Data from RODBC**

```r
# read data from database
> library(RODBC)
> conn <- odbcConnect("mydsn", uid="MT", pwd="password")
> data = sqlQuery(conn,"SELECT * FROM datatable")
> close(conn)
```
> # read comma separated file into memory
> data <- read.csv("C:/Users/ml013161/Desktop/modelling.csv")
> head(data)

Y X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14 X15 X16 X17 X18
1 0.4 6.26 106 1 98 16 1 3 3 1.106 0 6 3 22 1 1
2 0.2 4.12 53 0 0.10 98 98 6 3 53 NA NA 1.50 1 0
3 0.1 1.16 194 0 0 3 2 4 3 1.263 41 2 0 78 0 2
4 0.97 97 19 160 1 0 3 1 2 3 0.160 81 0 0 50 0 0
5 0.4 8 20 92 0.98 2 1 1 1 0.92 100 0 1 33 0 0
6 0.1 1.23 55 0.98 2 1 2 1 0.138 NA 0 0 57 1 0
   X19  X20  X21  X22  X23  X24  X25
1  1210000  72423  68  9 359  0  2604
2  9999999  32245  42  6  0  0 11581
3  68874  32409 149  3 309  0  4568
4  6619  22731  68  2  0  0 1236
5 116000  17069 47  5  0  1  6629
6 162000  12904 71  4  0  1 1471

> # read Excel file into memory
> library(RODBC)
> conn <- odbcConnectExcel("C:/Users/ml013161/Desktop/Modelling.xls")
> data <- sqlFetch(conn, "Sheet1")
> odbcClose(conn)
> head(data)

Y X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14 X15 X16 X17 X18
1 0.4 6.26 106 1 98 16 1 3 3 1.106 0 6 3 22 1 1
2 0.2 4.12 53 0 0.10 98 98 6 3 53 NA NA 1.50 1 0
3 0.1 1.16 194 0 0 3 2 4 3 1.263 41 2 0 78 0 2
4 0.97 97 19 160 1 0 3 1 2 3 0.160 81 0 0 50 0 0
5 0.4 8 20 92 0.98 2 1 1 1 0.92 100 0 1 33 0 0
6 0.1 1.23 55 0.98 2 1 2 1 0.138 NA 0 0 57 1 0
   X19  X20  X21  X22  X23  X24  X25
1  1210000  72423  68  9 359  0  2604
2  9999999  32245  42  6  0  0 11581
3  68874  32409 149  3 309  0  4568
4  6619  22731  68  2  0  0 1236
5 116000  17069 47  5  0  1  6629
6 162000  12904 71  4  0  1 1471
3. Data for Training Exercise

A modeling sample is provided for the response modeling exercise. The project objective is to identify the most likely customers who would apply for a mortgage loan provided we know about their past relationship with the bank. The data contains 4% responders and 96% non-responders captured in target variable. The non-responders are intended as Y=0.

3.1 Data Structure

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Label</th>
<th>Type</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Response to a mailing for loan</td>
<td>Categorical</td>
<td>Target</td>
</tr>
<tr>
<td>X1</td>
<td>Number Inquiries w/in 12 Months</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X2</td>
<td>Number Inquiries w/in 24 Months</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X3</td>
<td>Number Accounts</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X4</td>
<td>Age Oldest Bankcard Account</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X5</td>
<td>Number Revolving Accounts Opened w/in 6 Months</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X6</td>
<td>Number Open Auto Finance Accounts</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X7</td>
<td>Number Open Bankcard Accounts</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X8</td>
<td>Number Open Mortgage Accounts</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X9</td>
<td>Number Open Retail Accounts</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X10</td>
<td>Number Revolving Accounts w/Update w/in 3 Months w/ Balance &gt; $0</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X11</td>
<td>Number Open Revolving Accounts w/Update w/in 3 Months w/ Balance &gt;= 50% Credit Limit/High Credit</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X12</td>
<td>Age of Oldest Bank Revolving Trade</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X13</td>
<td>Ratio of total balance to HC/CL for all personal finance trades</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X14</td>
<td>Ratio of total balance to HC/CL for all retail trades.</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X15</td>
<td>Number of inquiries reported in past 6 months, excluding auto and real estate</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X16</td>
<td>Percent of active trades with balance &gt; $0</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X17</td>
<td>Number of open non-GE bankcard with lines &lt;= $500.</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X18</td>
<td>Number of open revolving retail trades updated within past 3 months with a balance &gt; $0</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X19</td>
<td>Highest limit on closed and opened mortgage loan</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X20</td>
<td>Total amount of credit available after outstanding balance subtracted from credit limit for all open revolving trades</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X21</td>
<td>Average age, in months, of all accounts</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X22</td>
<td>Number of promotional inquiries</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X23</td>
<td>Total revolving retail balance on retail revolving trades rated current:</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X24</td>
<td>Total number of revolving trades closed within the past 6 months.</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
<tr>
<td>X25</td>
<td>Total balance of open bankcard trades:</td>
<td>Numeric</td>
<td>Predictor</td>
</tr>
</tbody>
</table>
3.2 Attaching & Detaching Data

There are 2 ways of programming in R. One is with attaching data another without attaching data. If you are not attaching the data, the $ notation is used. E.g. `table$variable`, which is not at all convenient for huge list of components. Hence the attach() statement attaches the data table temporarily in R work space which makes your life easy. Hence after attaching the table like:

```r
table$variable
```

We can directly view the variables by simply typing

```r
> variable
```

After working on data, we can easily detach data using detach statement

```r
> detach(data)
```

```r
> summary(X1)
Error in summary(X1) : object 'X1' not found
```

```r
> summary(data$X1)

   Min. 1st Qu.  Median    Mean 3rd Qu.   Max. 
  0.000   1.000    2.000   14.350    8.000   99.000 
```

```r
> attach(data)
```

```r
> summary(X1)

   Min. 1st Qu.  Median    Mean 3rd Qu.   Max. 
  0.000   1.000    2.000   14.350    8.000   99.000 
```

```r
> detach(data)
```

> |
4. Descriptive Statistics

Once the data is loaded, the next step would be exploring the data. Descriptive statistics is the discipline of quantitatively describing the main features of the data. R provides a wide range of functions for running summary statistics. Some of the basic statistics are shown below.

4.1 Measures of central tendency

The mean is the average -- the sum of all the values divided by the number of values.

\[ \text{mean}(X4) \]

The median is the exact midpoint of the ordered set of values -- the number where half the values are below it and half are above it.

\[ \text{median}(X4) \]

4.2 Measures of Variance

The range is the simplest measure of variation to find. It is simply the highest value minus the lowest value.

\[ \text{range}(X4) \] #Min & Max value

The deviation from the mean is squared and called the "squared deviation from the mean". This "average squared deviation from the mean" is called the variance.

\[ \text{var}(X4) \]

There is a problem with variances. Recall that the deviations were squared. That means that the units were also squared. To get the units back the same as the original data values, the square root must be taken. This is known as "Standard Deviation".

\[ \text{sd}(X4) \]
4.3 Summary Statistics

Summary statistics are used to summarize a set of observations. There are many R functions designed to provide a range of summary statistics at once. For example:

```r
# mean, median, 25th and 75th quartiles, min, max
> summary(X4)  # summary statistics for single variable in the table
# min, lower-hinge, median, upper-hinge, max
> fivenum(X4)
```
> `summary(data)`  

#summary statistics for the complete dataset

<table>
<thead>
<tr>
<th></th>
<th>Y</th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
<th>X5</th>
<th>X6</th>
<th>X7</th>
<th>X8</th>
<th>X9</th>
<th>X10</th>
<th>X11</th>
<th>X12</th>
<th>X13</th>
<th>X14</th>
<th>X15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>0.00</td>
<td>Min.</td>
<td>0.00</td>
<td>Min.</td>
<td>Min.</td>
<td>0.00</td>
<td>Min.</td>
<td>0.00</td>
<td>Min.</td>
<td>Min.</td>
<td>Min.</td>
<td>Min.</td>
<td>Min.</td>
<td>Min.</td>
<td>Min.</td>
<td>Min.</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>0.00</td>
<td>1st Qu.:</td>
<td>1.00</td>
<td>1st Qu.:</td>
<td>2.00</td>
<td>1st Qu.:</td>
<td>14.00</td>
<td>1st Qu.:</td>
<td>3.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Median</td>
<td>0.00</td>
<td>Median :</td>
<td>2.00</td>
<td>Median :</td>
<td>4.00</td>
<td>Median :</td>
<td>21.00</td>
<td>Median :</td>
<td>4.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>0.04</td>
<td>Mean</td>
<td>14.35</td>
<td>Mean :</td>
<td>16.25</td>
<td>Mean</td>
<td>21.50</td>
<td>Mean :</td>
<td>5.648</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>0.00</td>
<td>3rd Qu.:</td>
<td>5.00</td>
<td>3rd Qu.:</td>
<td>8.00</td>
<td>3rd Qu.:</td>
<td>28.00</td>
<td>3rd Qu.:</td>
<td>6.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max.</td>
<td>1.00</td>
<td>Max. :</td>
<td>99.00</td>
<td>Max. :</td>
<td>99.00</td>
<td>Max. :</td>
<td>99.00</td>
<td>Max. :</td>
<td>99.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>X8</th>
<th>X9</th>
<th>X10</th>
<th>X11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>0.00</td>
<td>Min.</td>
<td>0.00</td>
<td>Min.</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>1.00</td>
<td>1st Qu.:</td>
<td>1.00</td>
<td>1st Qu.:</td>
</tr>
<tr>
<td>Median</td>
<td>1.00</td>
<td>Median :</td>
<td>3.00</td>
<td>Median :</td>
</tr>
<tr>
<td>Mean</td>
<td>30.97</td>
<td>Mean :</td>
<td>14.6</td>
<td>Mean :</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>98.00</td>
<td>3rd Qu.:</td>
<td>6.00</td>
<td>3rd Qu.:</td>
</tr>
<tr>
<td>Max.</td>
<td>99.00</td>
<td>Max. :</td>
<td>99.00</td>
<td>Max. :</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>X12</th>
<th>X13</th>
<th>X14</th>
<th>X15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>1.00</td>
<td>Min.</td>
<td>0.00</td>
<td>Min.</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>108.0</td>
<td>1st Qu.:</td>
<td>31.00</td>
<td>1st Qu.:</td>
</tr>
<tr>
<td>Median</td>
<td>160.0</td>
<td>Median :</td>
<td>59.00</td>
<td>Median :</td>
</tr>
<tr>
<td>Mean</td>
<td>212.0</td>
<td>Mean :</td>
<td>58.64</td>
<td>Mean :</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>220.2</td>
<td>3rd Qu.:</td>
<td>91.00</td>
<td>3rd Qu.:</td>
</tr>
<tr>
<td>Max.</td>
<td>9999.0</td>
<td>Max. :</td>
<td>177.00</td>
<td>Max. :</td>
</tr>
</tbody>
</table>

NA's : 890
5. Data Treatments

To have robust models we need to ensure that the data is robust as well. We should remember the GIGO principle (Garbage In Garbage Out). So the model is as good as the data it is built on. Typical Data problems we should consider for treatment:

- Missing Values
- Extreme Values

Non-Linear Variables

5.1 Missing value Treatment

Missing Values create problems at two stages:

- **Model Construction Stage**: Results are biased and may not represent the true association
- **Model Deployment Stage**: Prediction formula cannot score cases with missing values

These problems can be solved in following ways:

- Complete Case Analysis
- Use Synthetic distribution for missing values
- Use Estimation method to predict the missing values

Test for Missing Values

Summary statistics gives us the counts of missing value in the data along with all other statistics. Other than summary statistic `is.na` gives us the exact counts of missing value.

```r
> summary(data)
> summary(is.na(X13))
```

![R Console output](image)
Excluding Missing Values
If we have enough data for modeling, we can simply drop the missing data cases from the analysis. The option for omitting missing values from the analysis is given below:

```r
# na.omit Options
>nrow(data)
>sum(is.na(data))
>newdata <- na.omit(data)
>nrow(newdata)
>sum(is.na(newdata))
```

Replacing missing values
Most of the cases we don’t want to miss any data from the analysis. Hence we’ll try to impute the missing places by some other values such as mean, median or mode depending upon the need. This sometimes over fits the data, but most of the times help us in developing very good models.

```r
>mean(data$X13) # shows error as data contains missing value
>mean(data$X13, na.rm = TRUE) # removes missing value from calculating mean
# Replacing missing values using various statistical measures
>data$X13[is.na(data$X13)] <- round(mean(data$X13, na.rm = TRUE))
>data$X13[is.na(data$X13)] <- median(data$X13, na.rm = TRUE)
>mean(data$X13)
```
5.2 Outlier treatment

We need to understand the data using univariate analysis to see how the variables are populated. This will help us to identify the unusual or very low and high values (Extreme Values) from the rest of the values.

Extreme Values pose two challenges:

- First, in most real-world applications, the relationship between expected target value and input value does not increase without bound. Standard regression cannot detect such a relationship.
- Second, extreme points have more influence, or leverage, on model fit. This can result in over-fitting the data.

```r
> mean(data$X13)# shows error as data contains missing value
[1] NA
> mean(data$X13, na.rm = TRUE)# removes missing value from calculating mean
[1] 58.56797
> # Replacing missing values using various statistical measures
> data$X13[is.na(data$X13)] <- round(mean(data$X13, na.rm = TRUE))
> data$X13[is.na(data$X13)] <- median(data$X13, na.rm = TRUE)
> mean(data$X13)
[1] 58.6396
```
Test for Skewness
Skewness is a measure of symmetry, or more precisely, the lack of symmetry. A distribution, or data set, is symmetric if it looks the same to the left and right of the center point. A histogram is a graphical representation showing a visual impression of the distribution of data.

```R
> summary(X21)
> hist(X21)  #Plotting histogram
```
Capping Skewed Values

Once if the data is found to be skewed the upper and lower boundary can be capped with quintiles, standard deviation, etc. according to the business needs.

```r
> m <- mean(X21, na.rm = TRUE)  # Mean of the variable X21
> sd <- sd(X21, na.rm = TRUE)    # StdDev of the variable X21
> ul <- m + (2 * sd)             # Upper limit is calculated as Mean+2(StdDev)
> ll <- m - (2 * sd)             # Lower limit is calculated as Mean-2(StdDev)

# Capping values based on Mean & 2(StdDev) measures
> X21 <- as.numeric(
    ifelse(data$X21 <= ll, ll,
            ifelse(data$X21 > ul, ul,
                    X21)))

> hist(X21)
```

![R Console](image1)

![Histogram of X21](image2)
5.3 Binning Values

Sometimes it is very difficult to explain the numeric values. Hence binning such variables can add some business sense and also very easy to explore data.

```r
> summary(X4)
> age <- as.factor(
+    ifelse(X4 <= 250, 'Young',
+            ifelse(X4 <= 350, 'Mature', 'Old')))
> summary(age)
> counts <- table(Y, age)
> counts

# Table View

<table>
<thead>
<tr>
<th></th>
<th>Mature</th>
<th>Old</th>
<th>Young</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counts</td>
<td>483</td>
<td>255</td>
<td>4262</td>
</tr>
</tbody>
</table>
```

# Stacked bar plot

```r
> barplot(counts, 
+    main="Stacked Bar Plot",
+    xlab="Age", ylab="Frequency",
+    col=c("yellow", "green"),
+    legend=rownames(counts))
```
# Grouped bar plot

```r
barplot(counts,
        main="Grouped Bar Plot",
        xlab="Age", ylab="Frequency",
        col=c("yellow", "green"),
        legend=rownames(counts), beside=TRUE)
```
5.4 **Breaking Data into Training and Test Sample**

Let us say that we now have the data which contains the dependent variable along with the independent variables with all exclusions done. The next step before we begin modeling, the data needs to be split into two parts.

One part is used to build the model and the second part serves to validate or test the stability of the model. These are our training and test samples. The following code creates a training data set comprised of randomly selected 80% of the training data and 20% of test data sample.

```r
nrow(data)  # No Of observations in complete data
d = sort(sample(nrow(data), nrow(data) *.8))
# select training sample
train <- data[d,]
test <- data[-d,]
nrow(train)  # No Of observations in training data
nrow(test)  # No Of observations in testing data
```

```
> nrow(data)  # No Of observations in complete data
[1] 5000
> d = sort(sample(nrow(data), nrow(data) *.8))
> # select training sample
> train <- data[d,]
> test <- data[-d,]
> nrow(train)  # No Of observations in training data
[1] 4000
> nrow(test)  # No Of observations in testing data
[1] 1000
```
6. Variable Reduction

Once the data preparation and sampling is done 75% of the work is done for the modeling. The purpose of any modeling exercise is to identify the variables (or attributes) that best predict our target variable. Having too many, or variables that do not add much to the predictability will increase the model implementation cost.

6.1 Collinearity Measure

Multicollinearity between the explanatory variables indicates that these variables yield similar information in explaining the dependent variable. The way out shall be to drop Multicollinear variables from our model in by calculating VIF. Higher the VIF, higher is the auxiliary between the explanatory variables. It means the variable X1 is collinear with other variables.

```r
fit <- glm(Y~., data=train, family=binomial(logit))
vif(fit)
```

6.2 Stepwise Procedures

The logistic regression comes with variable selection features like forward, backward, stepwise which can help in identifying the right set of most predictive variables in a model.

**Backward Elimination**

This is the simplest of all variable selection procedures and can be easily implemented without special software. In situations where there is a complex hierarchy, backward elimination can be run manually while taking account of what variables are eligible for removal.

1. Start with all the predictors in the model
2. Remove the predictor with highest p-value greater than acrit
3. Refit the model and go to 2
4. Stop when all p-values are less than .

The is sometimes called the .p-to-remove and does not have to be 5%. If prediction performance is the goal, then a 15-20% cut-off may work best, although methods designed more directly for optimal prediction should be preferred.

```r
stepb <- step(fit, direction="backward")  # backward elimination
stepb$anova  # display results
```
Forward Selection
This just reverses the backward method.
1. Start with no variables in the model.
2. For all predictors not in the model, check their p-value if they are added to the model. Choose the one with lowest p-value less than .
3. Continue until no new predictors can be added.

```r
stepf<- step(fit, direction="forward")  #forward selection
stepf$anova # display results
```

```
> stepf$anova
   Step Df Deviance Resid. Df Resid. Dev   AIC
1  NA    NA       3974    1123.471 1175.471
2  - X23 1 0.02215401      3975    1123.493 1173.493
3  - X7 1 0.11039437      3976    1123.603 1171.603
4  - X15 1 0.10999223      3977    1123.713 1169.713
5  - X14 1 0.49528913      3978    1124.208 1168.208
6  - X8 1 0.64971839      3979    1124.858 1166.858
7  - X24 1 0.75700096      3980    1125.615 1165.615
8  - X10 1 0.88713116      3981    1126.502 1164.502
9  - X11 1 1.14886015      3982    1127.651 1163.651
10 - X6 1 1.62811846      3983    1129.279 1163.279
```
Stepwise Regression
This is a combination of backward elimination and forward selection. This addresses the situation where variables are added or removed early in the process and we want to change our mind about them later. At each stage a variable may be added or removed and there are several variations on exactly how this is done.

```r
stepw<- step(fit, direction="both") #Stepwise selection
stepw$anova # display results
```

7. Running Model & Interpreting Output

After selecting right variable, modeling starts. Here we go with Logistic regression which is part of a category of statistical models called generalized linear models. It allows one to predict a discrete outcome from a set of variables that may be continuous, discrete, dichotomous, or a mix of any of these. Generally, the dependent or response variable is dichotomous, such as presence/absence or success/failure.

The code below estimates a logistic regression model using the `glm` (generalized linear model) function from variable selected by backward selection method.

```r
> lgs<- glm(Y~X1+X2+X3+X4+X5+X9+X12+X13+X16+X17+X18+X20+X21+X22+X25,data=train,family=binomial(logit))
```

Since we gave our model a name (`lgs`), R will not produce any output from our regression. In order to get the results we use the `summary` command:

```r
> summary(lgs)
```
```r
> lgrs<-glm(Y~X1+X2+X3+X4+X5+X9+X12+X13+X16+X17+X18+X19+X20+X21+X22+X25, +
> data=train, family=binomial(logit))
> summary(lgrs)

Call:
glm(formula = Y ~ X1 + X2 + X3 + X4 + X5 + X9 + X12 + X13 + X16 +
    X17 + X18 + X19 + X20 + X21 + X22 + X25, family = binomial(logit),
    data = train)

Deviance Residuals:
     Min       1Q   Median       3Q      Max
-1.2619   -0.2979  -0.2081   -0.1389   3.6107

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  -4.406e+00  5.778e-01  -7.625  2.43e-14 ***
     X1       -7.500e-02  3.925e-02  -1.931  0.053450 .
     X2        7.875e-02  3.964e-02   1.987  0.046911 *
     X3        2.558e-02  1.121e-02   2.282  0.022498 *
     X4       -2.255e-03  2.288e-03  -0.986  0.324362
     X5        1.152e-01  8.239e-02   1.398  0.162101
     X9        4.264e-03  2.809e-03   1.518  0.128981
     X12       1.038e-03  2.281e-03   0.455  0.649052
     X13       4.926e-03  3.030e-03   1.626  0.104037
     X16       1.597e-02  5.301e-03   3.012  0.002597 **
     X17       5.944e-01  1.310e-01   4.536  5.72e-06 ***
     X18       1.753e-02  6.420e-02   2.730  0.006335 **
     X19       8.164e-02  2.232e-02   3.658  0.000255 ***
     X20      -4.755e-06  3.371e-06  -1.410  0.158426
     X21      -1.905e-02  4.436e-03  -4.294  1.75e-05 ***
     X22      -4.616e-02  1.610e-02  -2.868  0.004135 **
     X25       4.326e-05  1.045e-05   4.140  3.47e-05 ***

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

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1292.3  on 3999  degrees of freedom
Residual deviance: 1129.3  on 3983  degrees of freedom
AIC: 1163.3

Number of Fisher Scoring iterations: 11
```
We can use the `confint` function to obtain confidence intervals for the coefficient estimates.

# CIs using profiled log-likelihood

```r
> confint(lgrs)
Waiting for profiling to be done...
2.5 % 97.5 %
(Intercept) -5.555720e+00 -3.289227e+00
X1 -1.608906e-01 3.321004e-03
X2 -1.313037e-03 1.545029e-01
X3  3.704466e-03 4.766850e-02
X4 -6.472719e-03 -5.463772e-04
X5 -1.661206e-02 2.736886e-01
X6 -1.466398e-03 9.591575e-03
X12  8.279944e-05 5.350424e-03
X13 -8.264550e-04 1.095948e-02
X16  5.604385e-03 2.640109e-02
X17  3.292670e-01 8.456151e-01
X18  4.518944e-02 2.933803e-01
X19  3.800200e-08 1.255920e-07
X20 -1.167055e-05 1.442323e-06
X21 -2.793143e-02 -1.058351e-02
X22 -7.827981e-02 -1.614118e-02
X25  2.243669e-05 6.315751e-05
```

There were 35 warnings (use `warnings()` to see them)

# CIs using standard errors

```r
> confint.default(lgrs)
2.5 % 97.5 %
(Intercept) -5.537980e+00 -3.273233e+00
X1 -1.527198e-01 1.126171e-03
X2  1.076053e-03 1.564810e-01
X3  3.607731e-03 4.754235e-02
X4 -6.739798e-03 2.229659e-03
X5 -4.629620e-02 2.766611e-01
X6 -1.241074e-03 9.765696e-03
X12 -3.432274e-03 5.508122e-03
X13 -1.018222e-03 1.086500e-02
X16  5.576213e-03 2.635655e-02
X17  3.573742e-01 8.511387e-01
X18  4.953236e-02 3.011037e-01
X19  3.789717e-08 1.255852e-07
X20 -1.156257e-05 1.655816e-06
X21 -2.774272e-02 -1.035396e-02
X22 -7.770815e-02 -1.451087e-02
X25  2.277380e-05 6.373150e-05
```
8. Model Evaluation And Validation

Now having got the Model Performance metrics on the development sample, we need to verify whether the model will hold well on overall population. To do this we will use the Validation Sample kept aside earlier.

8.1 Generate the confusion matrix showing counts

Misclassification refers to measurement error. It gives us the matrix containing different counts such as True Positive, False Positive, True Negative and False Negative.

```r
> train$pr <- as.vector(ifelse(predict(lgrs, type="response", train) > 0.5, "1", "0"))
> table(train$Y, train$pr, dnn=c("Actual", "Predicted"))

> test$pr <- as.vector(ifelse(predict(lgrs, type="response", test) > 0.5, "1", "0"))
> table(test$Y, test$pr, dnn=c("Actual", "Predicted"))
```

RGui (32-bit) - [R Console]

```
> # Generate the confusion matrix showing counts for Training Set
> train$pr <- as.vector(ifelse
> (predict(lgrs, type="response", train) > 0.5, "1", "0"))
> table(train$Y, train$pr, dnn=c("Actual", "Predicted"))

Actual    0    1
 0 3847  1
 1  149  3

> # Generate the confusion matrix showing counts for Testing Set
> test$pr <- as.vector(ifelse
> (predict(lgrs, type="response", test) > 0.5, "1", "0"))
> table(test$Y, test$pr, dnn=c("Actual", "Predicted"))

Actual    0    1
 0   952  0
 1   47  1
```
8.2 Calculating ROC Curve & Lift for model

In a ROC curve the true positive rate (Sensitivity) is plotted in function of the false positive rate (Specificity) for different cut-off points of a parameter. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold.

```r
# ROC Chart: requires the ROCR package
library(ROCR)

#score train data set
> train$score <- predict(lgrs,type='response',train)
> predtrain <- prediction(train$score, train$Y)
> perftrain <- performance(predtrain, "tpr", "fpr")

#score test data set
> test$score <- predict(lgrs,type='response',test)
> predtest <- prediction(test$score, test$Y)
> perftest <- performance(predtest, "tpr", "fpr")

# Plotting ROC Curve
> plot(perftrain, main="ROC Curve", col="green")
> par(new=TRUE)
> plot(perftest, col="red", lty=2)
> legend("bottomright", c("Train", "Test"), cex=0.8,
  col=c("green","red"), lty=1:2)
```
```r
# Load data set
> train$score <- predict(lgrs, type='response', train)
> predtrain <- prediction(train$score, train$Y)
> perftrain <- performance(predtrain, "tpr", "fpr")

# Score test data set
> test$score <- predict(lgrs, type='response', test)
> predtest <- prediction(test$score, test$Y)
> perftest <- performance(predtest, "tpr", "fpr")

# Plot ROC Curve
> plot(perftrain, main="ROC Curve", col="green")
> par(new=TRUE)
> plot(perftest, col="red", lty=2)
> legend("bottomright", c("Train", "Test"), cex=0.8,
+ col=c("green", "red"), lty=1:2)
```
8.3 **Lift Chart**

A lift chart identifies the gain in performance offered by the model.

- Easiest to interpret metric in validation
- Cumulative distribution of target records against cumulative distribution of total sample with score
- Shows the ability of the score to capture target records (i.e. ‘Bad’ or ‘Response’)

```r
# Convert rate of positive predictions to percentage.
lifttrain <- performance(predtrain, "lift", "rpp")
lifttest <- performance(predtest, "lift", "rpp")

# Plot the lift chart.
plot(lifttrain, col="green", lty=1, xlab="Caseload (%)", add=FALSE, main="Lift Chart")
par(new=TRUE)
plot(lifttest, col="red", lty=2, xlab="Caseload (%)", add=FALSE)
legend("topright", c("Train", "Test"), cex=0.8,
       col=c("green", "red"), lty=1:2)
```

![Lift Chart](image)
The area under the ROC curve (AUC) is a measure of how well a parameter can distinguish between two diagnostic groups (Response Vs. Non-Response).

```r
# the following line computes the area under the curve for model performance (pred,"auc")
> performance(predtrain,"auc")
An object of class "performance"
Slot "x.name":
[1] "None"
Slot "y.name":
[1] "Area under the ROC curve"
Slot "alpha.name":
[1] "none"
Slot "x.values":
list()
Slot "y.values":
[[1]]
[1] 0.7834538
Slot "alpha.values":
list()
> performance(predtest,"auc")
An object of class "performance"
Slot "x.name":
[1] "None"
Slot "y.name":
[1] "Area under the ROC curve"
Slot "alpha.name":
[1] "none"
Slot "x.values":
list()
Slot "y.values":
[[1]]
[1] 0.7638524
Slot "alpha.values":
list()
```
8.5 *Calculating KS Statistic*

The KS statistic shows the maximum ability of the score to separate target records (i.e. ‘Bad’ or ‘Response’) to non-target (i.e. ‘Good’ or ‘Non response’) in a point / range of score.

#this code builds on ROCR library by taking the max delta between cumulative bad and good rates being plotted by ROCR
> max (attr(perftrain,'y.values')[[1]]-attr(perftrain,'x.values')[[1]])
> max (attr(perftest,'y.values')[[1]]-attr(perftest,'x.values')[[1]])