Safi Ammar Mohammed

C21034767

Credit Risk Scoring

MAT012

Assignment 2021/22

1) Split the dataset into two subsets as follows:

Subset 1: the applicants with Duration <= 12 months

Subset 2: the applicants where Duration > 12 months

Clean the subsets if necessary.

A quick summary of the dataset before the splitting shows that there is one variable that will require cleaning.

The purpose variable requires cleaning since it is supposed to be in numeric categories but returns as a character type variable. On closer inspection, there are 'X' characters present between numeric categories.

Referring to the data dictionary reveals that this 'X' category refers to the 10th category which is supposed to represent 'Other' purposes. All the 'X' characters are converted to '10' and the whole purpose variable is parsed as numeric to avoid issues while modelling.

It is important to note that the purpose variable does not have any data referring to the 7th category representing 'vacations.

```
german["Purpose"][german["Purpose"] == "X"] <- "10"
as.numeric(unique(german$Purpose))
german$Purpose <- as.numeric(german$Purpose)
summary(german)
```

Fig 1: Cleaning the Purpose variable.

Splitting the dataset:

The cleaned dataset is now split according to the following conditions:

L12: This refers to the dataset of applicants whose duration is less than or equal to 12.

M12: This refers to the dataset of applicants whose duration is more than 12.

(Any future mentions of l12, m12 refer to this split of data based on duration.)

```
l12 <- german[german$Duration <= 12,]
m12 <- german[german$Duration > 12,]
```

Fig 2: Splitting the dataset.

2) For each subset, establish a training set and validation set. Explain:

a) What principle you have used to decide on these?

To split the subsets into training and validation sets, random sampling is done, using 'Good" as target variable.

```
> table(l12$Good)

  0   1
 76 283
> table(m12$Good)

  0   1
224 417
```

Fig 3: Tables of 'Good' variable.

The number of 'Goods' out of total applicants in each subset is used as the ratio to split the data into training and testing sets (Fig 3).

```
l12_traintest = split_df(l12, y = "Good", ratio = 0.79, seed = 7)
m12_traintest = split_df(m12, y = "Good", ratio = 0.65, seed = 7)
l12_train <- l12_traintest$train
l12_test <- l12_traintest$test
m12_train <- m12_traintest$train
m12_test <- m12_traintest$test
```

Fig 4: Train Test split of the subsets.

b) Why both training and validation sets are needed?

Training set is used to build the model upon the data. Without the test sets, there is no way of performing an evaluation of the model without bias. If the same training data was used for evaluation, the model performance would be near 100%. Therefore, we split the data into training and validation sets to validate the model either to tune it during the process or to evaluate and measure the performance of the final build. In this assignment it will be used to evaluate the final models (Similar to test set).

c) Any issues encountered during the splitting exercise.

There were no specific issues during the split, but it is good to note that for m12, the good ratio was much less when compared to l12, 0.65 and 0.79 respectively. There is much less data in l12 comparatively.

3) For each training set choose four variables which are suitable for building a scorecard. For each training set the variables must have (i) at least one continuous variable before binning; (ii) at least one categorical variable with more than two categories, so you can see whether categories can be combined.

Explain the rationale behind your choice of variables (using supporting statistics eg chi-square). Should you be unable to choose variables satisfying the above criteria, explain the problem you have encountered and the solution you have chosen to compromise the variable selection.

To get a good idea of which variables have a strong correlation with the target 'Good' variable, the *create_infotables* function in R is used. This function provides the Information Values (IV) of each variable with respect to its correlation with 'Good'.

L12 Train Set:

```
> l12_info <- create_infotables(l12_train, NULL, y="Good")
> l12_info$Summary
    Variable          IV
1   Checking 0.517546175
3    History 0.471511446
12  Property 0.402405646
2   Duration 0.348701339
13       Age 0.317776085
```

Fig 5: L12 Infotable Summary

The top four variables with the highest IV are Checking, History, Property, and Duration. Duration is the only continuous variable among the top four (Fig 5).

The Persons Chi Square Independence Test between the categorical variables and the target variable is used to confirm the correlation of the variables.

```
> with(l12_train, CrossTable(Good, Checking, digits = 1, prop.chisq = F, chisq = T))

   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  283

             | Checking
        Good |         1 |         2 |         3 |         4 | Row Total |
-------------|-----------|-----------|-----------|-----------|-----------|
           0 |        27 |        22 |         4 |        10 |        63 |
             |       0.4 |       0.3 |       0.1 |       0.2 |       0.2 |
             |       0.4 |       0.3 |       0.2 |       0.1 |           |
             |       0.1 |       0.1 |       0.0 |       0.0 |           |
-------------|-----------|-----------|-----------|-----------|-----------|
           1 |        50 |        45 |        20 |       105 |       220 |
             |       0.2 |       0.2 |       0.1 |       0.5 |       0.8 |
             |       0.6 |       0.7 |       0.8 |       0.9 |           |
             |       0.2 |       0.2 |       0.1 |       0.4 |           |
-------------|-----------|-----------|-----------|-----------|-----------|
Column Total |        77 |        67 |        24 |       115 |       283 |
             |       0.3 |       0.2 |       0.1 |       0.4 |           |
-------------|-----------|-----------|-----------|-----------|-----------|

Statistics for All Table Factors


Pearson's Chi-squared test
------------------------------------------------------------
Chi^2 =  24.28621     d.f. =  3     p =  2.176838e-05
```

Fig 6: Chi Sq. Test with CrossTable (Checking L12)

The cross table gives a brief statistic of the relation between ‘Checking’ and ‘Good’ (Fig 6). The Chi Square test statistic value helps compare the different categorical variables performance. The p-value helps deciding if the variable is independent of the target variable or not. If the p-value is more than the significance level of 0.05, the null hypothesis is rejected (H_0: The variable is independent).

Similarly, all the other categorical variables are tested. The higher the Chi Square value, the more prominent the relation with the target variable. All the categorical variables tested have a p-value less than 0.05. The tested variables include Checking, History, Property, and Emploed.

After comparison using Chi Square values, the categorical variables selected are History, Checking, and Property, in order. The continuous variable 'Duration' is selected based on the IV from the infotable. The final variable selection:

- Checking (Categorical)
- History (Categorical)
- Property (Categorical)
- Duration (Continuous)

M12 Train Set:

```
> m12_info <- create_infotables(m12_train, NULL, y="Good")
> m12_info$Summary
    Variable           IV
1   Checking 0.7617589115
6    Savings 0.3485819403
4    Purpose 0.2995884965
13       Age 0.2431246307
3    History 0.2251005841
2   Duration 0.1257788703
```

Fig 7: M12 Infotable Summary

The top four variables with the highest IV are Checking, Savings, Purpose, and Age. Age is the only continuous variable among the top four (Fig 7). According to Siddiqi (2006), by convention the values of IV in credit scoring, any value less than 0.02 is not useful for prediction, 0.02 to 0.1 is weak, 0.3 to 0.5 is strong and anything above 0.5 can be suspicious. Though checking has a very high IV value, this might reduce after coarse classification. One of the interesting points is that duration has more IV in l12 than it has in m12, even though m12 has significantly more Duration values (listendata.com 2015).

The Persons Chi Square Independence Test between the categorical variables and the target variable is used to confirm the correlation of the variables.

```
> with(m12_train, CrossTable(Good, Savings, digits = 1, prop.chisq = F, chisq = T))

   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  408

             | Savings
        Good |         1 |         2 |         3 |         4 |         5 | Row Total |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
           0 |       100 |        19 |         5 |         1 |        11 |       136 |
             |       0.7 |       0.1 |       0.0 |       0.0 |       0.1 |       0.3 |
             |       0.4 |       0.4 |       0.2 |       0.0 |       0.2 |           |
             |       0.2 |       0.0 |       0.0 |       0.0 |       0.0 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
           1 |       144 |        29 |        20 |        19 |        60 |       272 |
             |       0.5 |       0.1 |       0.1 |       0.1 |       0.2 |       0.7 |
             |       0.6 |       0.6 |       0.8 |       0.9 |       0.8 |           |
             |       0.4 |       0.1 |       0.0 |       0.0 |       0.1 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|
Column Total |       244 |        48 |        25 |        20 |        71 |       408 |
             |       0.6 |       0.1 |       0.1 |       0.0 |       0.2 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|

Statistics for All Table Factors

Pearson's Chi-squared test
------------------------------------------------------------
Chi^2 =  26.66399     d.f. =  4     p =  2.32431e-05
```

Fig 8: Chi Sq. Test with CrossTable (Savings M12)

The cross table gives a brief statistic of the relation between 'Savings' and 'Good' (Fig 8). The Chi Square test statistic value helps compare the different categorical variables performance

Similarly, all the other categorical variables are tested. The higher the Chi Square value, the more prominent the relation with the target variable. All the categorical variables tested have a p-value less than 0.05. The tested variables include Checking, Savings, Purpose, and History.

After comparison using Chi Square values, the categorical variables selected are Checking, Savings, and Purpose in order. The continuous variable 'Duration' is selected based on the IV from the infotable. The final variable selection:

- Checking (Categorical)
- Savings (Categorical)
- Purpose (Categorical)
- Age (Continuous)

```
# Let us create training subsets using the variables we selected.
l12_trainsubset <- l12_train[,c('Checking', 'History', 'Property', 'Duration', 'Good')]
m12_trainsubset <- m12_train[,c('Checking', 'Savings', 'Purpose', 'Age', 'Good')]
```

Fig 9: Training subsets with selected variables.

Problems Encountered:

For some of the Chi Sq. Tests, there is a warning that the Chi Sq. value is not accurate. This is because there are too few data points in some of the categories in the variables. In such cases, the combination of observation of IV and the Chi Sq. Test results is relied upon.

4) Using the binary variables obtained from the coarse classification in the above exercise to build two scorecards for each training set (so, two scorecards for those applicants with Duration <= 12 months; another two for those with Duration > 12 months), one using linear regression and one using logistic regression.

Coarse Classification:

Weight of Evidence (WoE) is checked to decide how many bins for each continuous variables are going to be the best. This is done using the *woebin* function in R. It divides categorical and continuous variables into buckets and calculates weights of evidence and information value for each bucket. The aim is to have a monotone relationship between the weight of evidence values of each bucket.

```
> woebin(l12_trainsubset, y = "Good", method="chimerge", positive="bad|0")$Duration
[INFO] creating woe binning ...
   variable       bin count count_distr neg pos    posprob        woe     bin_iv  total_iv breaks
1: Duration  [-Inf,7)    66   0.2332155  59   7 0.10606061 -0.8811345 0.13840041 0.2957422      7
2: Duration    [7,10)    44   0.1554770  32  12 0.27272727  0.2696636 0.01214070 0.2957422     10
3: Duration   [10,12)    31   0.1095406  28   3 0.09677419 -0.9830994 0.07830748 0.2957422     12
4: Duration [12, Inf)   142   0.5017668 101  41 0.28873239  0.3489444 0.06689359 0.2957422    Inf
```

Fig 10: Binning of Duration in L12.

For example, the default binning for Duration by the *woebin* function does not provide a monotone WoE (Fig 10). Therefore, experimenting with the buckets, changing the breaks, a good binning solution is obtained.

```
> breaks_list = list(Checking = c(2, 3), History = c(3, 4), Property = c(3, 4),Duration = c(7, 12))
> woebin(l12_trainsubset, y = "Good", method="chimerge", positive="bad|0", breaks_list = breaks_list)$Duration
[INFO] creating woe binning ...
   variable       bin count count_distr neg pos   posprob        woe      bin_iv  total_iv breaks
1: Duration  [-Inf,7)    66   0.2332155  59   7 0.1060606 -0.8811345 0.138400415 0.2099971      7
2: Duration    [7,12)    75   0.2650177  60  15 0.2000000 -0.1358015 0.004703084 0.2099971     12
3: Duration [12, Inf)   142   0.5017668 101  41 0.2887324  0.3489444 0.066893592 0.2099971    Inf
```

Fig 11: Binning Solution for Duration in L12.

In this case, decreasing the number of bins using the *breaks_list* attribute in the *woebin* function, buckets with monotone WoE's are obtained. These buckets can be used to divide the Duration variable into binary variables (Fig 11).

```
> woebin(l12_trainsubset, y = "Good", method="chimerge", positive="bad|0")$Checking
[INFO] creating woe binning ...
   variable      bin count count_distr neg pos    posprob        woe      bin_iv  total_iv breaks
1: Checking [-Inf,2)    77  0.27208481  50  27 0.35064935  0.6343067 0.127685111 0.5655794      2
2: Checking    [2,3)    67  0.23674912  45  22 0.32835821  0.5348728 0.077375175 0.5655794      3
3: Checking    [3,4)    24  0.08480565  20   4 0.16666667 -0.3589451 0.009841207 0.5655794      4
4: Checking [4, Inf)   115  0.40636042 105  10 0.08695652 -1.1008824 0.350677919 0.5655794    Inf
```

Fig 12: Binning of Checking in L12.

Checking has a good WoE monotony, but we can try to combine some categories and achieve similar results (Fig 12).

```
> breaks_list = list(Checking = c(3, 4), History = c(2, 3), Property = c(2, 3),Duration = c(7, 12))
> woebin(l12_trainsubset, y = "Good", method="chimerge", positive="bad|0", breaks_list = breaks_list)$Checking
[INFO] creating woe binning ...
   variable      bin count count_distr neg pos    posprob        woe      bin_iv  total_iv breaks
1: Checking [-Inf,3)   144  0.50883392  95  49 0.34027778  0.5884362 0.203575159 0.5640943      3
2: Checking    [3,4)    24  0.08480565  20   4 0.16666667 -0.3589451 0.009841207 0.5640943      4
3: Checking [4, Inf)   115  0.40636042 105  10 0.08695652 -1.1008824 0.350677919 0.5640943    Inf
```

Fig 13: Binning Solution for Checking in L12.

The WoE obtained provides the best breaks for classifying the Checking variable (Fig 13). Hence, we merge '1' and '2' categories of Checking. This means that according to the data dictionary, the status of existing checking account below 0 DM and between 0 and 200 DM will be combined into one category.

Similarly, for continuous variables, buckets are reduced, or breaks are changed until a good WoE solution is obtained using *woebin*. For categorical variables, buckets are combined, or breaks are changed until a good WoE solution is obtained.

After multiple experimentations with the training subsets, the following breaks and merges are obtained:

L12 Train Subset:

- Checking (Categorical): Breaks at 3, 4 are best. Hence, merging '1' and '2'.
- History (Categorical): Breaks at 2, 3 are best. Hence, merging '3' and '4'.
- Property (Categorical): Breaks at 3, 4 are best. Hence, merging '1' and '2'.
- Duration (Continuous): Best class split at 7, 12.

M12 Train Subset:

- Checking (Categorical): Breaks at 2, 3, 4 are best.
- Savings (Categorical): Breaks at 3, 4 are best. Hence, merging (1,2), and (4,5).
- Purpose (Categorical): Breaks at 1, 3 are best. Hence, merging (1,2), and (3...9).
- Age (Continuous): Best class split at 24, 30, 35.

```
> # Creating Binary Variables according to the established buckets.
> l12_trainsubset <-
+   l12_trainsubset %>%
+   mutate(Checking12 = ifelse(Checking==1|Checking==2, 1, 0),
+          Checking3 = ifelse(Checking==3, 1, 0),
+          History01 = ifelse(History==0|History==1, 1 ,0),
+          History2 = ifelse(History==2, 1, 0),
+          Property12 = ifelse(Property==1|Property==2, 1, 0),
+          Property3 = ifelse(Property==3, 1, 0),
+          Duration06 = ifelse(Duration<=6, 1, 0),
+          Duration711 = ifelse(Duration>6&Duration<=11, 1, 0))
> l12_trainsubset <- select(l12_trainsubset, -c(Checking, History, Property, Duration))
> head(l12_trainsubset)
   Good Checking12 Checking3 History01 History2 Property12 Property3 Duration06 Duration711
1:    1          0         0         0        0          1         0          0           0
2:    1          0         0         0        1          1         0          0           0
3:    0          1         0         0        1          0         1          0           0
4:    1          1         0         0        1          0         1          0           0
5:    1          0         0         0        0          0         1          0           1
6:    1          1         0         0        0          1         0          0           1
```

Fig 14: L12 Coarse Classification.

Now, based on the established buckets, binary variables are created using the *mutate* function in R (Fig 14). The training subset is then rid of the original non-binary variables.

```
> m12_trainsubset <-
+   m12_trainsubset %>%
+   mutate(Checking1 = ifelse(Checking==1, 1, 0),
+          Checking2 = ifelse(Checking==2, 1, 0),
+          Checking3 = ifelse(Checking==3, 1, 0),
+          Savings12 = ifelse(Savings==1|Savings==2, 1 ,0),
+          Savings3 = ifelse(Savings==3, 1, 0),
+          Purpose0 = ifelse(Purpose==0, 1, 0),
+          Purpose12 = ifelse(Purpose==1|Purpose==2, 1, 0),
+          Age023 = ifelse(Age<=23, 1, 0),
+          Age2429 = ifelse(Age>23&Age<=29, 1, 0),
+          Age3034 = ifelse(Age>29&Age<=34, 1, 0))
> m12_trainsubset <- select(m12_trainsubset, -c(Checking,Savings, Purpose, Age))
> head(m12_trainsubset)
   Good Checking1 Checking2 Checking3 Savings12 Savings3 Purpose0 Purpose12 Age023 Age2429 Age3034
1:    0         1         0         0         1        0        1         0      0       0       0
2:    1         0         0         0         0        0        0         0      0       0       0
3:    1         0         1         0         1        0        0         1      0       0       0
4:    0         0         1         0         1        0        1         0      0       1       0
5:    0         1         0         0         1        0        0         0      0       1       0
6:    0         1         0         0         1        0        1         0      0       0       0
```

Fig 15: M12 Coarse Classification.

On inspection of the IV values on these processed datasets, it can be seen how coarse classification has shifted the ranks of variables (Fig 16).

```
> l12_infosubset <- create_infotables(l12_trainsubset, NULL, y="Good")
> l12_infosubset$Summary
     Variable          IV
1  Checking12 0.528349431
3   History01 0.312312425
7  Duration06 0.168941266
5  Property12 0.139840390
6   Property3 0.066919631
4    History2 0.023955510
2   Checking3 0.010655847
8 Duration711 0.006314166
```

Fig 16: IV of processed L12.

Linear Regression Models:

```
# Linear Regression L12
l12_linr <- lm(Good~., data=l12_trainsubset)
vif(l12_linr, merge_coef = TRUE)
# None of the VIF is above 5. This means there is no co-linearity between variables.
l12_linrstep <- step(l12_linr, direction = 'both', trace = FALSE)
l12_linr <- eval(l12_linrstep$call)
summary(l12_linr)
```

Fig 17: Linear Regression Model Code

The linear regression models are built using the *lm* function in R. The variance inflation factor is calculated for each variable using the *vif* function. This helps in checking co-linearity between variables. Standard rule of thumb is that if VIF value is greater than 5, there might be co-linearity between variables (Fig17) (Sthda.com 2018). None of the VIF values are greater than 5 for this model. After creating a model, the co-efficients of variables are obtained as shown (Fig18).

```
> # Linear Regression L12
> l12_linr <- lm(Good~., data=l12_trainsubset)
> summary(l12_linr)

Call:
lm(formula = Good ~ ., data = l12_trainsubset)

Residuals:
    Min      1Q  Median      3Q     Max
-1.0310 -0.1075  0.1096  0.2581  0.8007

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.80386    0.08251   9.743  < 2e-16 ***
Checking12  -0.22497    0.04870  -4.620 5.91e-06 ***
Checking3   -0.07213    0.08548  -0.844   0.3995
History01   -0.41624    0.09666  -4.306 2.32e-05 ***
History2    -0.07643    0.04937  -1.548   0.1228
Property12   0.16301    0.07518   2.168   0.0310 *
Property3    0.03665    0.08431   0.435   0.6641
Duration06   0.14060    0.05757   2.442   0.0152 *
Duration711  0.06258    0.05434   1.152   0.2505
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3788 on 274 degrees of freedom
Multiple R-squared:  0.1973,    Adjusted R-squared:  0.1738
F-statistic: 8.416 on 8 and 274 DF,  p-value: 3.174e-10
```

Fig 18: Initial Linear Regression L12.

Next, a step function eliminates any variables which are not significant to the model based on the p-value obtained after building the model. This presents the final Linear Regression model for L12 (Fig 19).

```
> l12_linrstep <- step(l12_linr, direction = 'both', trace = FALSE)
> l12_linr <- eval(l12_linrstep$call)
> summary(l12_linr)

Call:
lm(formula = Good ~ Checking12 + History01 + History2 + Property12 +
    Duration06, data = l12_trainsubset)

Residuals:
     Min       1Q   Median       3Q      Max
-1.01649 -0.09462  0.09843  0.23626  0.80201

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.84187    0.05454  15.437  < 2e-16 ***
Checking12  -0.21421    0.04618  -4.638 5.42e-06 ***
History01   -0.42967    0.09603  -4.474 1.12e-05 ***
History2    -0.07813    0.04913  -1.590  0.11288
Property12   0.13783    0.04815   2.863  0.00452 **
Duration06   0.11492    0.05399   2.129  0.03416 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3782 on 277 degrees of freedom
Multiple R-squared:  0.1912,    Adjusted R-squared:  0.1766
F-statistic: 13.09 on 5 and 277 DF,  p-value: 1.882e-11
```

Fig 19: Linear Regression Model L12.

Using the same principles and steps, a linear regression model for M12 is obtained (Fig 20).

```
> # Linear Regression M12
> m12_linr <- lm(Good~., data=m12_trainsubset)
> vif(m12_linr, merge_coef = TRUE)
       variable    Estimate Std. Error t value Pr(>|t|)     gvif
 1: (Intercept)  1.02644534     0.0573 17.9094   0.0000       NA
 2:   Checking1 -0.35803856     0.0562 -6.3666   0.0000 1.384937
 3:   Checking2 -0.27333216     0.0522 -5.2357   0.0000 1.285271
 4:   Checking3 -0.08667358     0.0982 -0.8830   0.3778 1.105837
 5:   Savings12 -0.18915882     0.0535 -3.5383   0.0005 1.308463
 6:    Savings3 -0.05908307     0.0973 -0.6075   0.5438 1.223996
 7:    Purpose0 -0.16633208     0.0555 -2.9958   0.0029 1.143901
 8:   Purpose12  0.01453685     0.0501  0.2904   0.7717 1.182110
 9:      Age023 -0.12393527     0.0797 -1.5557   0.1206 1.148828
10:     Age2429 -0.01267558     0.0520 -0.2438   0.8075 1.237374
11:     Age3034  0.02642010     0.0582  0.4537   0.6503 1.213707
> # None of the VIF is above 5. This means there is no co-linearity between variables.
> m12_linrstep <- step(m12_linr, direction = 'both', trace = FALSE)
> m12_linr <- eval(m12_linrstep$call)
> summary(m12_linr)

Call:
lm(formula = Good ~ Checking1 + Checking2 + Savings12 + Purpose0 +
    Age023, data = m12_trainsubset)

Residuals:
    Min      1Q  Median      3Q     Max
-1.0132 -0.4014  0.1679  0.3366  0.8164

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.01322    0.04423  22.908  < 2e-16 ***
Checking1   -0.34642    0.05347  -6.479 2.70e-10 ***
Checking2   -0.26191    0.05036  -5.201 3.16e-07 ***
Savings12   -0.18112    0.04766  -3.800 0.000167 ***
Purpose0    -0.16875    0.05188  -3.253 0.001239 **
Age023      -0.13333    0.07478  -1.783 0.075346 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4241 on 402 degrees of freedom
Multiple R-squared:  0.2025,    Adjusted R-squared:  0.1926
F-statistic: 20.42 on 5 and 402 DF,  p-value: < 2.2e-16
```

Fig 20: Linear Regression Model M12.

Logistic Regression Models:

```
# Logistic Regression M12
m12_logr <- glm(Good ~ ., family= binomial(), data = m12_trainsubset)
vif(m12_logr, merge_coef = TRUE)
# None of the VIF is above 5. This means there is no co-linearity between variables.
m12_logrstep <- step(m12_logr, direction = 'both', trace = FALSE)
m12_logr <- eval(m12_logrstep$call)
summary(m12_logr)
```

Fig 21: Logistic Regression Model Code.

The logistic regression models are built using the *glm* function in R. The family is set as binomial since logistic regression produces continuous outputs. The variance inflation factor is calculated for each variable using the *vif* function. None of the VIF values are greater than 5 for this model. After creating a model, the co-efficientS of variables are obtained as shown (Fig22).

```
> # Logistic Regression M12
> m12_logr <- glm(Good ~ ., family= binomial(), data = m12_trainsubset)
> summary(m12_logr)

Call:
glm(formula = Good ~ ., family = binomial(), data = m12_trainsubset)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2270  -0.9214   0.5113   0.7944   1.8837

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  3.06651    0.42212   7.265 3.74e-13 ***
Checking1   -1.94299    0.32990  -5.890 3.87e-09 ***
Checking2   -1.60863    0.31703  -5.074 3.89e-07 ***
Checking3   -0.67428    0.58674  -1.149 0.250475
Savings12   -1.24096    0.35660  -3.480 0.000501 ***
Savings3    -0.48468    0.64534  -0.751 0.452622
Purpose0    -0.83413    0.30216  -2.761 0.005771 **
Purpose12    0.11032    0.28644   0.385 0.700125
Age023      -0.63668    0.41606  -1.530 0.125951
Age2429     -0.06798    0.28770  -0.236 0.813198
Age3034      0.14325    0.33697   0.425 0.670763
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 519.4  on 407  degrees of freedom
Residual deviance: 425.6  on 397  degrees of freedom
AIC: 447.6

Number of Fisher Scoring iterations: 5
```

Fig 22: Initial Logistic Regression M12.

Next, a step function eliminates any variables which are not significant to the model based on the p-value obtained after building the model. This presents the final Logistic Regression model for M12 (Fig 23).

```
> m12_logrstep <- step(m12_logr, direction = 'both', trace = FALSE)
> m12_logr <- eval(m12_logrstep$call)
> summary(m12_logr)

Call:
glm(formula = Good ~ Checking1 + Checking2 + Savings12 + Purpose0 +
    Age023, family = binomial(), data = m12_trainsubset)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4352  -0.9384   0.5641   0.8251   1.9028

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   2.9122     0.3433   8.482  < 2e-16 ***
Checking1    -1.8243     0.3030  -6.020 1.74e-09 ***
Checking2    -1.4946     0.2958  -5.052 4.37e-07 ***
Savings12    -1.1546     0.3079  -3.750 0.000177 ***
Purpose0     -0.8550     0.2814  -3.038 0.002379 **
Age023       -0.7099     0.3872  -1.834 0.066711 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 519.40  on 407  degrees of freedom
Residual deviance: 427.92  on 402  degrees of freedom
AIC: 439.92

Number of Fisher Scoring iterations: 4
```

Fig 23: Logistic Regression Model M12.

Using the same principles and steps, a logistic regression model for M12 is obtained (Fig 24).

```
> # Logistic Regression L12
> l12_logr <- glm(Good ~ ., family= binomial(), data = l12_trainsubset)
> vif(l12_logr, merge_coef = TRUE)
      variable   Estimate Std. Error z value Pr(>|z|)     gvif
1: (Intercept)  1.8265282     0.6005  3.0419   0.0024       NA
2:  Checking12 -1.6994639     0.4036 -4.2105   0.0000 1.306611
3:   Checking3 -0.6529783     0.6922 -0.9433   0.3455 1.249986
4:   History01 -2.3161030     0.6282 -3.6872   0.0002 1.291259
5:    History2 -0.6112979     0.3952 -1.5466   0.1220 1.324059
6:  Property12  1.1698766     0.5116  2.2867   0.0222 2.430912
7:   Property3  0.2991542     0.5550  0.5390   0.5899 2.388395
8:  Duration06  1.1656148     0.4955  2.3524   0.0187 1.113720
9: Duration711  0.3995308     0.3810  1.0486   0.2944 1.073593
> # None of the VIF is above 5. This means there is no co-linearity between variables.
> l12_logrstep <- step(l12_logr, direction = 'both', trace = FALSE)
> l12_logr <- eval(l12_logrstep$call)
> summary(l12_logr)

Call:
glm(formula = Good ~ Checking12 + History01 + History2 + Property12 +
    Duration06, family = binomial(), data = l12_trainsubset)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6244   0.1867   0.4218   0.6573   2.0305

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   2.0524     0.4388   4.678 2.90e-06 ***
Checking12   -1.5652     0.3624  -4.319 1.57e-05 ***
History01    -2.4124     0.6244  -3.863 0.000112 ***
History2     -0.6300     0.3909  -1.612 0.107070
Property12    0.9525     0.3356   2.838 0.004537 **
Duration06    1.0365     0.4815   2.153 0.031353 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 300.09  on 282  degrees of freedom
Residual deviance: 243.14  on 277  degrees of freedom
AIC: 255.14

Number of Fisher Scoring iterations: 5
```

Fig 24: Logistic Regression Model L12.

5) Derive ROC curves for all scorecards using the validation set applicable to each, showing in detail how sensitivity and specificity have been calculated. Estimate the Gini coefficient and KS values for each. Explain and comment on your results.

Processing the Validation Sets:

The validation sets have been untouched after train-test split. But to evaluate the model using these sets, they must be coarse classified in the same manner as training sets. Both the L12, and M12 test sets are processed using similar code (Fig 25).

```
# Processing the test sets to prepare for evaluation.
l12_testsubset <- l12_test[,c('Checking', 'History', 'Property', 'Duration', 'Good')]
m12_testsubset <- m12_test[,c('Checking', 'Savings', 'Purpose', 'Age', 'Good')]
l12_testsubset <-
  l12_testsubset %>%
  mutate(Checking12 = ifelse(Checking==1|Checking==2, 1, 0),
         Checking3 = ifelse(Checking==3, 1, 0),
         History01 = ifelse(History==0|History==1, 1 ,0),
         History2 = ifelse(History==2, 1, 0),
         Property12 = ifelse(Property==1|Property==2, 1, 0),
         Property3 = ifelse(Property==3, 1, 0),
         Duration06 = ifelse(Duration<=6, 1, 0),
         Duration711 = ifelse(Duration>6&Duration<=11, 1, 0))
head(l12_testsubset)

l12_testsubset <- select(l12_testsubset, -c(Checking, History, Property, Duration))
```

Fig 25: Test Set Processing

Using a performance evaluation (*perf_eva*) function from the *scorecard* package of R, a ROC curve graph along with KS, and Gini coefficients is obtained (Fig 26). This function calculates

the sensitivity and specificity from a confusion matrix of the predicted variables. From the confusion matrix, the True Positives (The positives that the model predicted correctly) are divided by the total actual positives to give the sensitivity. Similarly, specificity is calculated by dividing the True Negative with the total actual negative (Rdrr.io 2021).

Sensitivity = True Positive/Total Actual Positive

Specificity = True Negative/Total Actual Negative

The ROC is then obtained by plotting Sensitivity on the y-axis and 1-Specificity on the x-axis.

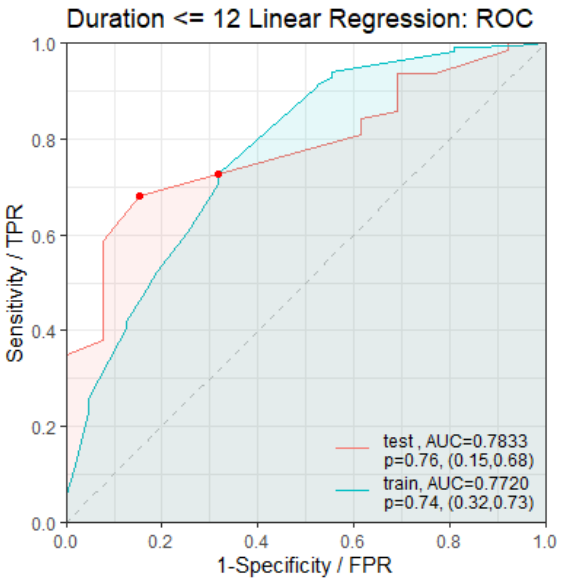
```
l12_linr_pred <- lapply(l12_traintestsubset, function(x) predict(l12_linr, x, type='response'))
l12_linr_perf <- perf_eva(pred = l12_linr_pred, label=l12_label, binomial_metric = c('ks','gini'),
                 show_plot = 'roc', title = 'Duration <= 12 Linear Regression', confusion_matrix = TRUE)
l12_linr_perf
```

Fig 26: Performance Evaluation Code

The same code is used to obtain ROC, KS, and Gini for all the scorecards.

The ROC curve depicts how the model performed with the training data in comparison with how it performed using the testing data performance. It also gives the Area Under the Curve (AUC) for each. It is a standard that larger the AUC, the better a model performs. The Gini coefficient indicates the effectiveness of the model in differentiating between "bad" borrowers, who will default in the future, and "good" borrowers, who won't default in the future (Towardsdatascience.com 2020). The KS statistic is used as the deciding metric to judge the efficacy of models in credit scoring. The higher the KS statistic, the more efficient the model is at capturing the responders (Discuss.analyticsvidhya.com 2016).

Scorecards for Duration <=12:

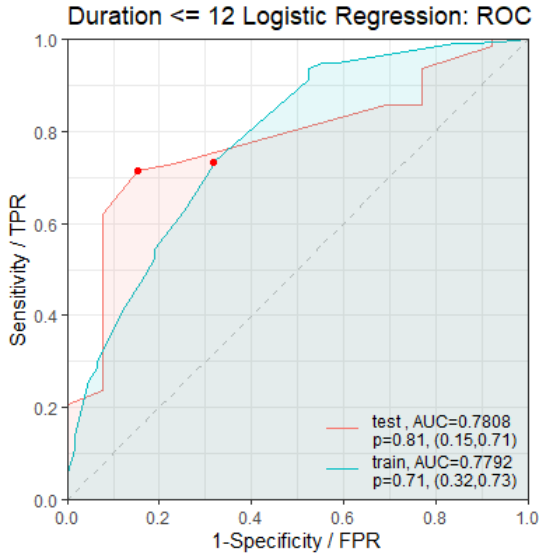



```
> l12_linr_perf
$binomial_metric
$binomial_metric$train
          KS      Gini
1: 0.4098124 0.5439394

$binomial_metric$test
          KS      Gini
1: 0.5286935 0.5665446


$confusion_matrix
$confusion_matrix$train
   label pred_0 pred_1      error
1:     0     28     35 0.55555556
2:     1     13    207 0.05909091
3: total     41    242 0.16961131

$confusion_matrix$test
   label pred_0 pred_1      error
1:     0      3     10 0.76923077
2:     1      4     59 0.06349206
3: total      7     69 0.18421053
```

Fig 27: Performance Evaluation Linear Regression L12.

The AUC of the model for test dataset is 0.78 which is very good, and the ROC of test set retains a similar shape to the ROC of train set. The KS statistic suggests that the model is

moderate at capturing responders, meaning it has a decent efficacy. The Gini coefficient suggest that the model is decent when it comes to bad future defaults vs good future non defaults. The model can perform much better if there are more data points in each of the variables (Fig 27).

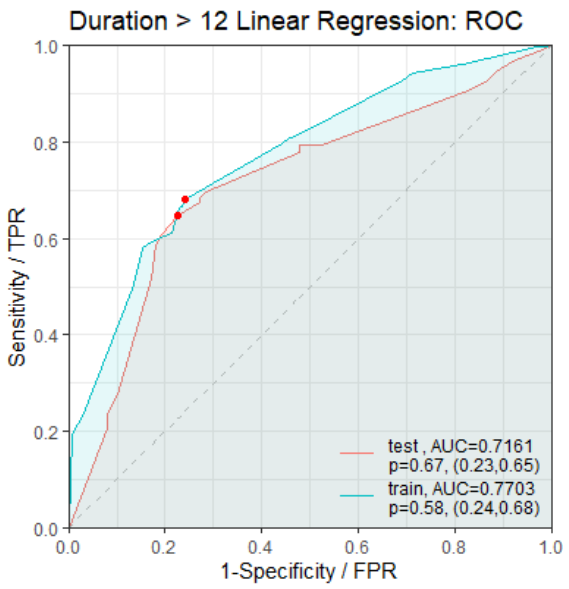



```
> l12_logr_perf
$binomial_metric
$binomial_metric$train
          KS      Gini
1: 0.4143579 0.5583694

$binomial_metric$test
          KS      Gini
1: 0.5604396 0.5616606


$confusion_matrix
$confusion_matrix$train
   label pred_0 pred_1     error
1:     0     28     35 0.5555556
2:     1     11    209 0.0500000
3: total     39    244 0.1625442

$confusion_matrix$test
   label pred_0 pred_1      error
1:     0      3     10 0.76923077
2:     1      4     59 0.06349206
3: total      7     69 0.18421053
```

Fig 28: Performance Evaluation Logistic Regression L12.

The AUC of the model for test dataset is 0.78 which is very good, and the ROC of test set retains a similar shape to the ROC of train set. The KS statistic and the Gini coefficient are very similar to the linear regression model suggesting that the model has a decent efficacy and is decent when it comes to bad future defaults vs good future non defaults. The model can perform much better if there are more data points in each of the variables. The data split for duration seems to have a negative effect on the scorecards involving data of applicants with Duration<12 (Fig 28).

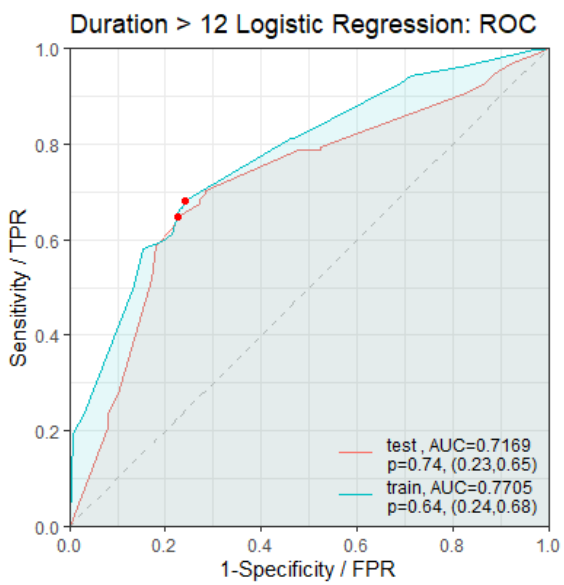
Scorecards for Duration >12:




```
> m12_linr_perf
$binomial_metric
$binomial_metric$train
        KS      Gini
1: 0.4375 0.5405493

$binomial_metric$test
          KS    Gini
1: 0.4210031 0.43221


$confusion_matrix
$confusion_matrix$train
   label pred_0 pred_1      error
1:     0     39     97 0.71323529
2:     1     16    256 0.05882353
3: total     55    353 0.27696078

$confusion_matrix$test
   label pred_0 pred_1      error
1:     0     12     76 0.86363636
2:     1     11    134 0.07586207
3: total     23    210 0.37339056
```

Fig 29: Performance Evaluation Linear Regression M12.

The AUC of the model for test dataset is 0.71 which is decent compared to the train set, and the ROC of test set retains a much more similar shape to the ROC of train set when compared to L12 models. The KS statistic suggests that the model is weaker at capturing responders, meaning it has a lesser efficacy. The Gini coefficient suggest that the model is slightly feeble when it comes to bad future defaults vs good future non defaults. The model performs with a higher error margin. This can be improved by either increasing the size of the dataset or selecting more than four variables to build the model (Fig 29).




```
> m12_logr_perf
$binomial_metric
$binomial_metric$train
        KS    Gini
1: 0.4375 0.54109

$binomial_metric$test
          KS      Gini
1: 0.4210031 0.4337774


$confusion_matrix
$confusion_matrix$train
   label pred_0 pred_1       error
1:     0     39     97 0.71323529
2:     1     16    256 0.05882353
3: total     55    353 0.27696078

$confusion_matrix$test
   label pred_0 pred_1       error
1:     0     12     76 0.86363636
2:     1     11    134 0.07586207
3: total     23    210 0.37339056
```

Fig 30: Performance Evaluation Logistic Regression M12.

The AUC of the model for test dataset is 0.72 which is decent compared to the train set, and the ROC of test set retains a much more similar shape to the ROC of train set when compared to L12 models. The KS statistic and the Gini coefficient are very similar to the linear regression model. The KS statistic suggests that the model is weaker at capturing responders, meaning it has a lesser efficacy than the L12 models. The Gini coefficient suggest that the model is slightly feeble when it comes to bad future defaults vs good future non defaults. Like the linear regression model, this model performs with a higher error margin. This can be improved by either increasing the size of the dataset or selecting more than four variables to build the model (Fig 30).

REFERENCES

Investopedia.com, 2021. Basel Accord [Online]. Available at: https://www.investopedia.com/terms/b/basel_accord.asp [Accessed: 22 March 2022].

Investopedia.com, 2021. Basel I [Online]. Available at: https://www.investopedia.com/terms/b/basel_i.asp#toc-requirements-for-basel-i [Accessed: 22 March 2022].

Investopedia.com, 2021. Basel II Definition [Online]. Available at: https://www.investopedia.com/terms/b/baselii.asp#toc-basel-ii-requirements [Accessed: 22 March 2022].

Investopedia.com, 2021. Basel III [Online]. Available at: https://www.investopedia.com/terms/b/basell-iii.asp#toc-leverage-and-liquidity-measures [Accessed: 22 March 2022].

Investopedia.com, 2021. Basel IV [Online]. Available at: https://www.investopedia.com/basel-iv-5218598#toc-what-is-basel-iv [Accessed: 22 March 2022].

Businessinsider.com, 2014. Credit Scoring Is An Outdated System Holding Back The American Economy [Online]. Available at: https://www.businessinsider.com/credit-scoring-outdated-2014-12?r=US&IR=T [Accessed: 25 March 2022].

Plend.co.uk, 2021. CREDIT SCORING – THE OLD, THE BROKEN, AND WHY THE FUTURE IS OPEN [Online]. Available at: https://plend.co.uk/2022/02/17/credit-scoring-the-old-the-broken-and-why-the-future-is-open/ [Accessed: 25 March 2022]

Fool.com, 2021. How Gen Z, Millennials, Gen X, and Baby Boomers Use Credit Cards [Online]. Available at: https://www.fool.com/the-ascent/research/how-gen-z-millennials-gen-x-baby-boomers-use-credit-cards/ [Accessed: 26 March 2022]

Listendata.com, 2015. Weight Of Evidence (WOE) AND Information Value (IV) Explained [Online]. Available at: https://www.listendata.com/2015/03/weight-of-evidence-woe-and-information.html [Accessed: 27 March 2022]

REFERENCES

Sthda.com, 2018. Multicollinearity Essentials and VIF in R [Online]. Available at: http://www.sthda.com/english/articles/39-regression-model-diagnostics/160-multicollinearity-essentials-and-vif-in-r/
[Accessed: 27 March 2022]

Rdrr.io, 2021. perf_eva: Binomial Metrics [Online]. Available at: https://rdrr.io/cran/scorecard/man/perf_eva.html
[Accessed: 27 March 2022]

Discuss.analyticsvidhya.com, 2016. Logistic Regression Model Validation in layman terms [Online]. Available at: https://discuss.analyticsvidhya.com/t/logistic-regression-model-validation-in-layman-terms/532/2
[Accessed: 27 March 2022]

Towardsdatascience.com, 2020. Using the Gini coefficient to evaluate the performance of credit score models [Online]. Available at: https://towardsdatascience.com/using-the-gini-coefficient-to-evaluate-the-performance-of-credit-score-models-59fe13ef420#:~:text=The%20Gini%20coefficient%20is%20a,t%20default%20in%20the%20future.
[Accessed: 27 March 2022]

APPENDIX A

The binary variables and the coefficients for each variable calculated in each regression:

1) Linear Regression (Duration<=12)

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.80386	0.08251	9.743	< 2e-16
Checking12	-0.22497	0.0487	-4.62	0.00000591
Checking3	-0.07213	0.08548	-0.844	0.3995
History01	-0.41624	0.09666	-4.306	0.0000232
History2	-0.07643	0.04937	-1.548	0.1228
Property12	0.16301	0.07518	2.168	0.031
Property3	0.03665	0.08431	0.435	0.6641
Duration06	0.1406	0.05757	2.442	0.0152
Duration711	0.06258	0.05434	1.152	0.25

2) Linear Regression (Duration>12)

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	1.02645	0.05731	17.909	< 2e-16
Checking1	-0.35804	0.05624	-6.367	5.34e-10
Checking2	-0.27333	0.05220	-5.236	2.67e-07
Checking3	-0.08667	0.09816	-0.883	0.37776
Savings12	-0.18916	0.05346	-3.538	0.00045
Savings3	-0.05908	0.09725	-0.608	0.54385
Purpose0	-0.16633	0.05552	-2.996	0.00291
Purpose12	0.01454	0.05007	0.290	0.77170
Age023	-0.12394	0.07967	-1.556	0.12059
Age2429	-0.01268	0.05199	-0.244	0.80750
Age3034	0.02642	0.05823	0.454	0.65026

3) Logistic Regression (Duration<=12)

	Estimate	Std Error	z value	Pr(>|z|)
(Intercept)	5.16284	5.65941	0.912	0.36163
Checking12	-1.62648	1.12388	-1.447	0.14784
Checking3	-0.68982	0.82622	-0.835	0.40377
History01	-2.37771	2.70837	-0.878	0.37999
History2	-0.58741	1.55316	-0.378	0.70528
Property12	-2.56114	1.23037	-2.082	0.03738
Property3	-1.18474	0.71995	-1.646	0.09985
Duration06	2.59474	2.21676	1.171	0.2418
Duration711	0.78578	1.05808	0.743	0.4577

4) Logistic Regression (Duration>12)

	Estimate	Std Error	z value	Pr(>|z|)
(Intercept)	3.06651	0.42212	7.265	3.74E-13
Checking1	-1.94299	0.3299	-5.89	3.87E-09
Checking2	-1.60863	0.31703	-5.074	3.89E-07
Checking3	-0.67428	0.58674	-1.149	0.250475
Savings12	-1.24096	0.3566	-3.48	0.000501
Savings3	-0.48468	0.64534	-0.751	0.452622
Purpose0	-0.83413	0.30216	-2.761	0.005771
Purpose12	0.11032	0.28644	0.385	0.700125
Age023	-0.63668	0.41606	-1.53	0.125951
Age2429	-0.06798	0.2877	-0.236	0.813198
Age3034	0.14325	0.33697	0.425	0.670763

APPENDIX B

```
install.packages("xlsx")

install.packages("scorecard")

install.packages("Information")

install.packages("gmodels")

library("gmodels")

library("scorecard")

library("Information")

library("xlsx")

library("dplyr")


german <- read.xlsx("German.xlsx",1)

head(german)

nrow(german)

ncol(german)

summary(german)


# The 'Purpose'' variable might require some cleaning.

# There are unrecognized 'X' characters.

# Referring to the data dictionary, this can be the last 'other' category that is supposed to be represented by '10'.

# Let us change all the 'X' to '10' and convert the variable into an numerical categorical variable.

# Purpose does not have any data for the '7'(vacation) category.


german["Purpose"][german["Purpose"] == "X"] <- "10"

as.numeric(unique(german$Purpose))

german$Purpose <- as.numeric(german$Purpose)

summary(german)
```

```
# The purpose variable is now usable in building models.

l12 <- german[german$Duration <= 12,]
m12 <- german[german$Duration > 12,]
summary(l12)
summary(m12)

# Throughout the code 'l12' refers to credit data that has Duration <= 12.
# Throughout the code 'm12' refers to credit data that has Duration > 12.

table(l12$Good)
table(m12$Good)

l12_traintest = split_df(l12, y = "Good", ratio = 0.79, seed = 7)
m12_traintest = split_df(m12, y = "Good", ratio = 0.65, seed = 7)
l12_train <- l12_traintest$train
l12_test <- l12_traintest$test
m12_train <- m12_traintest$train
m12_test <- m12_traintest$test

# 2a) Random Sampling based on good/bad ratio, using Good as target variable.
# 2b) Simple answer is to measure the performance of the models, for example using ROC.
# 2c) There were no specific issues, but it is good to note that for m12, the good to bad ratio
was much less when compared to l12.

table(l12_train$Good)
table(m12_train$Good)
table(m12_test$Good)
table(l12_test$Good)
```

```
l12_info <- create_infotables(l12_train, NULL, y="Good")

m12_info <- create_infotables(m12_train, NULL, y="Good")

l12_info

m12_info


# One of the interesting points is that duration has more IV in l12 than it has in m12, even though m12 has significantly more Duration values.


l12_info$Summary

# Checking -> Categorical

# History -> Categorical

# Property -> Categorical

# Duration -> Continuous


# Lets check and compare the correlation of these top categorical variables selected based on the IV using Chi Sq Independence test with the target "Good" variable.

# Duration is the only continuous variable in the top 4.


with(l12_train, CrossTable(Good, Checking, digits = 1, prop.chisq = F, chisq = T))

with(l12_train, CrossTable(Good, History, digits = 1, prop.chisq = F, chisq = T))

with(l12_train, CrossTable(Good, Property, digits = 1, prop.chisq = F, chisq = T))


# All three variables are correlated to the target variable, since we reject null hypothesis(Variables are independent) in all three Chi Square tests.

# There is a warning that the History Chi Square value may be wrong.

# The warning we are getting when doing the Chi Sq tests is because there are too few data points in some of the categories.

# Hence, the test is not completely accurate. We can rely on a combination of observation of IV and the Chi Sq Test results in such cases.


with(l12_train, CrossTable(Good, Emploed, digits = 1, prop.chisq = F, chisq = T))
```

Eploed has a weak correlation with the target variable when compared to Checking, History and Property.

Hence our choice of categorical variables is final.

Any IV below 0.2 is considered pretty weak according to a rule of thumb proposed by Siddiqui.

m12_info$Summary

Checking -> Categorical

Savings -> Categorical

Purpose -> Categorical

Age -> Continuous

History -> Categorical

According to the info table these 5 variables have the highest IV. Lets Chi Sq test the categorical variables.

with(m12_train, CrossTable(Good, Checking, digits = 1, prop.chisq = F, chisq = T))

Chi Sq value for 'Checking' in m12 is much higher for checking than l12.

with(m12_train, CrossTable(Good, Savings, digits = 1, prop.chisq = F, chisq = T))

with(m12_train, CrossTable(Good, Purpose, digits = 1, prop.chisq = F, chisq = T))

with(m12_train, CrossTable(Good, History, digits = 1, prop.chisq = F, chisq = T))

All 4 are related to the target varibale, but Checking, Savings and Purpose have a higher Chi Sq value than History.

Hence, based on the IV and Chi Sq tests we select Checking, Savings, Purpose, and Age as the four variables.

NOTE: For continuous variables, we use the IV values from the info table to measure the correlation with the target variable.

The final selection for l12: Checking, History, Property, and Duration.

The final selection for m12: Checking, Savings, Purpose, and Age.

```
# Let us create training subsets using the variables we selected.

l12_trainsubset <- l12_train[,c('Checking', 'History', 'Property', 'Duration', 'Good')]

m12_trainsubset <- m12_train[,c('Checking', 'Savings', 'Purpose', 'Age', 'Good')]


# We will check WoE to decide on how many bins for each continuous variables is going to be the best.


woebin(l12_train, y = "Good", method="chimerge", positive="bad|0")


# Binning for l12.


woebin(l12_trainsubset, y = "Good", method="chimerge", positive="bad|0")


# Checking what class division is best for Duration.

woebin(l12_trainsubset, y = "Good", method="chimerge", positive="bad|0", var_skip = c('Checking', 'History', 'Property'), breaks_list = list( Duration = c(9,11,12)))

woebin(l12_trainsubset, y = "Good", method="chimerge", positive="bad|0", var_skip = c('Checking', 'History', 'Property'), breaks_list = list( Duration = c(5,8,12)))

woebin(l12_trainsubset, y = "Good", method="chimerge", positive="bad|0", var_skip = c('Checking', 'History', 'Property'), breaks_list = list( Duration = c(5,9,12)))

woebin(l12_trainsubset, y = "Good", method="chimerge", positive="bad|0", var_skip = c('Checking', 'History', 'Property'), breaks_list = list( Duration = c(5,7,12)))

woebin(l12_trainsubset, y = "Good", method="chimerge", positive="bad|0", var_skip = c('Checking', 'History', 'Property'), breaks_list = list( Duration = c(4,7,12)))


# There is no Duration less than 5. Duration might do better if we lessen the bins.

woebin(l12_trainsubset, y = "Good", method="chimerge", positive="bad|0", breaks_list = list( Duration = c(7,12)))


# Checking has a good WoE monotonity, but we can try to combine some categories and achieve similar results.
```

History has a weak monotonic relationship with the target. We can try to make that better by combining some categories.

Property has similar problems to history.

Trial and Error with Woebins untill a good WoE satisfied solution is obtained.

```
breaks_list = list(Checking = c(2, 3), History = c(3, 4), Property = c(3, 4),Duration = c(7, 12))

woebin(l12_trainsubset, y = "Good", method="chimerge", positive="bad|0", breaks_list = breaks_list)

breaks_list = list(Checking = c(3, 4), History = c(2, 3), Property = c(2, 3),Duration = c(7, 12))

woebin(l12_trainsubset, y = "Good", method="chimerge", positive="bad|0", breaks_list = breaks_list)$Checking

breaks_list = list(Checking = c(2, 4), History = c(2, 3, 4), Property = c(2, 4),Duration = c(7, 12))

woebin(l12_trainsubset, y = "Good", method="chimerge", positive="bad|0", breaks_list = breaks_list)

breaks_list = list(Checking = c(2, 3), History = c(2, 4), Property = c(2, 4),Duration = c(7, 12))

woebin(l12_trainsubset, y = "Good", method="chimerge", positive="bad|0", breaks_list = breaks_list)

l12_breaks_list = list(Checking = c(3, 4), History = c(2, 3), Property = c(3, 4),Duration = c(7, 12))

woebin(l12_trainsubset, y = "Good", method="chimerge", positive="bad|0", breaks_list = l12_breaks_list)
```

Checking: Breaks at 3,4 are best according to WoE.Hence we are merging 1 and 2 categories of Checking. (Check meaning in real data)

History: Breaks at 2,3 are best according to WoE.Hence we are merging 3 and 4 categories of History. (Check meaning in real data)

Property: Breaks at 3,4 are best according to WoE.Hence we are merging 1 and 2 categories of Property. (Check meaning in real data)

Duration: Best split at 7,12.

Binning for m12.

```
woebin(m12_trainsubset, y = "Good", method="chimerge", positive="bad|0")
```

Checking already has a very good WoE monotony as well as a good total IV, hence we are going to keep the 2,3,4 split.

Savings has a skewed monotonic relationship. We can try to combine some categories to achieve better results.

Purpose does not have monotony at all.

Age does not have monotony as well. We will try different classes till we get better results.

Trial and Error with Woebins untill a good WoE satisfied solution is obtained.

```
m12_breaks_list = list(Savings = c(3, 4), Purpose = c(2, 5, 7, 9), Age = c(22, 26, 30, 34, 38, 42))

woebin(m12_trainsubset, y = "Good", method="chimerge", positive="bad|0", breaks_list = m12_breaks_list)

m12_breaks_list = list(Savings = c(2, 3), Purpose = c(3, 5, 7),Age = c(25, 30, 34, 40, 50))

woebin(m12_trainsubset, y = "Good", method="chimerge", positive="bad|0", breaks_list = m12_breaks_list)

m12_breaks_list = list(Savings = c(2, 3), Purpose = c(4, 5, 7),Age = c(25, 30, 34, 45))

woebin(m12_trainsubset, y = "Good", method="chimerge", positive="bad|0", breaks_list = m12_breaks_list)

m12_breaks_list = list(Savings = c(3, 4), Purpose = c(4, 7),Age = c(25, 35, 40, 45))

woebin(m12_trainsubset, y = "Good", method="chimerge", positive="bad|0", breaks_list = m12_breaks_list)

m12_breaks_list = list(Savings = c(3, 4), Purpose = c(1, 3),Age = c(25, 35, 40))

woebin(m12_trainsubset, y = "Good", method="chimerge", positive="bad|0", breaks_list = m12_breaks_list)

m12_breaks_list = list(Savings = c(3, 4), Purpose = c(1, 3),Age = c(24, 30, 35))

woebin(m12_trainsubset, y = "Good", method="chimerge", positive="bad|0", breaks_list = m12_breaks_list)
```

Checking: Breaks at 2,3,4 are best according to WoE.

Savings: Breaks at 3,4 are best according to WoE.Hence we are merging (1,2), and (4,5) categories of Savings. (Check meaning in real data)

Purpose: Breaks at 1,3 are best according to WoE.Hence we are merging (1,2), and (3...9) categories of Purpose. (Check meaning in real data)

```
# Age: Best class split at 24, 30, 35. This is turning out to be a weak IV variable.

# Creating Binary Variables according to the established buckets.
l12_trainsubset <-
  l12_trainsubset %>%
  mutate(Checking12 = ifelse(Checking==1|Checking==2, 1, 0),
         Checking3 = ifelse(Checking==3, 1, 0),
         History01 = ifelse(History==0|History==1, 1 ,0),
         History2 = ifelse(History==2, 1, 0),
         Property12 = ifelse(Property==1|Property==2, 1, 0),
         Property3 = ifelse(Property==3, 1, 0),
         Duration06 = ifelse(Duration<=6, 1, 0),
         Duration711 = ifelse(Duration>6&Duration<=11, 1, 0))
head(l12_trainsubset)

l12_trainsubset <- select(l12_trainsubset, -c(Checking, History, Property, Duration))

m12_trainsubset <-
  m12_trainsubset %>%
  mutate(Checking1 = ifelse(Checking==1, 1, 0),
         Checking2 = ifelse(Checking==2, 1, 0),
         Checking3 = ifelse(Checking==3, 1, 0),
         Savings12 = ifelse(Savings==1|Savings==2, 1 ,0),
         Savings3 = ifelse(Savings==3, 1, 0),
         Purpose0 = ifelse(Purpose==0, 1, 0),
         Purpose12 = ifelse(Purpose==1|Purpose==2, 1, 0),
         Age023 = ifelse(Age<=23, 1, 0),
         Age2429 = ifelse(Age>23&Age<=29, 1, 0),
         Age3034 = ifelse(Age>29&Age<=34, 1, 0))
```

```
head(m12_trainsubset)

m12_trainsubset <- select(m12_trainsubset, -c(Checking,Savings, Purpose, Age))

l12_infosubset <- create_infotables(l12_trainsubset, NULL, y="Good")
m12_infosubset <- create_infotables(m12_trainsubset, NULL, y="Good")
l12_infosubset$Summary
m12_infosubset

# The IV has shifted based on the binary variables.

# Linear Regression L12
l12_linr <- lm(Good~., data=l12_trainsubset)
vif(l12_linr, merge_coef = TRUE)
# None of the VIF is above 5. This means there is no co-linearity between variables.
l12_linrstep <- step(l12_linr, direction = 'both', trace = FALSE)
l12_linr <- eval(l12_linrstep$call)
summary(l12_linr)

# Linear Regression M12
m12_linr <- lm(Good~., data=m12_trainsubset)
vif(m12_linr, merge_coef = TRUE)
# None of the VIF is above 5. This means there is no co-linearity between variables.
m12_linrstep <- step(m12_linr, direction = 'both', trace = FALSE)
m12_linr <- eval(m12_linrstep$call)
summary(m12_linr)

# Logistic Regression L12
l12_logr <- glm(Good ~ ., family= binomial(), data = l12_trainsubset)
```

```
vif(l12_logr, merge_coef = TRUE)
# None of the VIF is above 5. This means there is no co-linearity between variables.
l12_logrstep <- step(l12_logr, direction = 'both', trace = FALSE)
l12_logr <- eval(l12_logrstep$call)
summary(l12_logr)

# Logistic Regression M12
m12_logr <- glm(Good ~ ., family= binomial(), data = m12_trainsubset)
vif(m12_logr, merge_coef = TRUE)
# None of the VIF is above 5. This means there is no co-linearity between variables.
m12_logrstep <- step(m12_logr, direction = 'both', trace = FALSE)
m12_logr <- eval(m12_logrstep$call)
summary(m12_logr)

# Processing the test sets to prepare for evaluation.
l12_testsubset <- l12_test[,c('Checking', 'History', 'Property', 'Duration', 'Good')]
m12_testsubset <- m12_test[,c('Checking', 'Savings', 'Purpose', 'Age', 'Good')]
l12_testsubset <-
  l12_testsubset %>%
  mutate(Checking12 = ifelse(Checking==1|Checking==2, 1, 0),
         Checking3 = ifelse(Checking==3, 1, 0),
         History01 = ifelse(History==0|History==1, 1 ,0),
         History2 = ifelse(History==2, 1, 0),
         Property12 = ifelse(Property==1|Property==2, 1, 0),
         Property3 = ifelse(Property==3, 1, 0),
         Duration06 = ifelse(Duration<=6, 1, 0),
         Duration711 = ifelse(Duration>6&Duration<=11, 1, 0))
head(l12_testsubset)
```

```
l12_testsubset <- select(l12_testsubset, -c(Checking, History, Property, Duration))


m12_testsubset <-
  m12_testsubset %>%
  mutate(Checking1 = ifelse(Checking==1, 1, 0),
         Checking2 = ifelse(Checking==2, 1, 0),
         Checking3 = ifelse(Checking==3, 1, 0),
         Savings12 = ifelse(Savings==1|Savings==2, 1 ,0),
         Savings3 = ifelse(Savings==3, 1, 0),
         Purpose0 = ifelse(Purpose==0, 1, 0),
         Purpose12 = ifelse(Purpose==1|Purpose==2, 1, 0),
         Age023 = ifelse(Age<=23, 1, 0),
         Age2429 = ifelse(Age>23&Age<=29, 1, 0),
         Age3034 = ifelse(Age>29&Age<=34, 1, 0))
head(m12_testsubset)


m12_testsubset <- select(m12_testsubset, -c(Checking,Savings, Purpose, Age))


l12_traintestsubset <- list(train = l12_trainsubset,test = l12_testsubset)
m12_traintestsubset <- list(train = m12_trainsubset,test = m12_testsubset)
l12_label <- lapply(l12_traintestsubset, function(x) x$Good)
m12_label <- lapply(m12_traintestsubset, function(x) x$Good)


l12_linr_pred <- lapply(l12_traintestsubset, function(x) predict(l12_linr, x, type='response'))
l12_linr_perf <- perf_eva(pred = l12_linr_pred, label=l12_label, binomial_metric =
c('ks','gini'),
             show_plot = 'roc', title = 'Duration <= 12 Linear Regression', confusion_matrix =
TRUE)
l12_linr_perf
```

```
l12_logr_pred <- lapply(l12_traintestsubset, function(x) predict(l12_logr, x,
type='response'))

l12_logr_perf <- perf_eva(pred = l12_logr_pred, label=l12_label, binomial_metric =
c('ks','gini'),

        show_plot = 'roc', title = 'Duration <= 12 Logistic Regression', confusion_matrix =
TRUE)

l12_logr_perf


m12_linr_pred <- lapply(m12_traintestsubset, function(x) predict(m12_linr, x,
type='response'))

m12_linr_perf <- perf_eva(pred = m12_linr_pred, label=m12_label, binomial_metric =
c('ks','gini'),

        show_plot = 'roc', title = 'Duration > 12 Linear Regression', confusion_matrix =
TRUE)

m12_linr_perf


m12_logr_pred <- lapply(m12_traintestsubset, function(x) predict(m12_logr, x,
type='response'))

m12_logr_perf <- perf_eva(pred = m12_logr_pred, label=m12_label, binomial_metric =
c('ks','gini'),

        show_plot = 'roc', title = 'Duration > 12 Logistic Regression', confusion_matrix =
TRUE)

m12_logr_perf



#The End!
```