SCORING

The Project Team chose to score the Neural Network model by running a new regression that would generate ten most important data points: Loan Amount, Checking Acct, Age, Loan Duration, Credit History, Employment, Savings Account, Installment Rate, Present Residence, and Job.

#Filter variables

```
> {
> dt_f = var_filter(Credit_Data_v4csv, y="DEFAULT")
[INFO] filtering variables ...
```

#Breaking into training and test datasets. I set the threshold at 60% accuracy and the se ed at 42.

```
> dt_list = split_df(dt_f, y="DEFAULT", ratio = 0.6, seed = 42)
> label_list = lapply(dt_list, function(x) x$DEFAULT)
```

Next, we binned the variables that had equal levels using the weight of evidence (WOE) binning technique. The WOE is a simple calculation of the importance of an independent variable to the dependent variable.

$$WOE = \left(\frac{Num\ of\ Good\ risk\ customers}{Num\ bad\ risk\ customers}\right)$$

#woe binning

```
> bins = woebin(dt_f, y="DEFAULT")
[INFO] creating woe binning ...
```

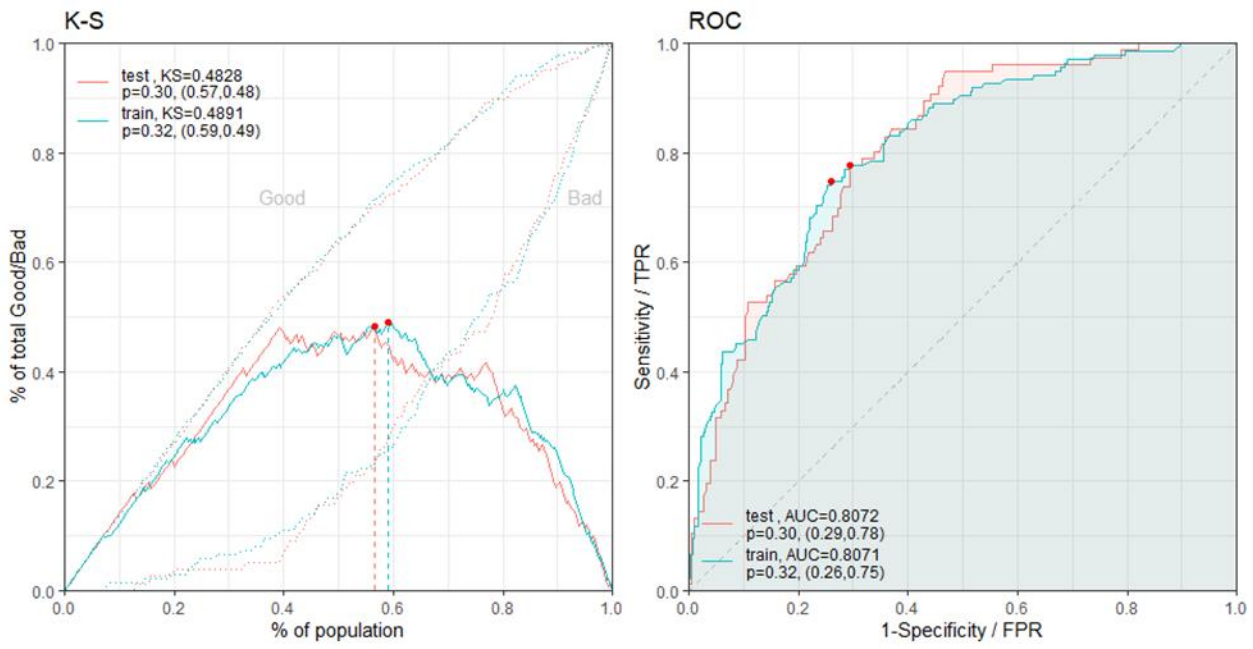
converting train and test into woe values

```
> breaks_adj = list(
+ LOAN_AMOUNT.or.LOAN_DURATION OR EMPLOYMENT=c(0,1,2,3,4),
+ other.AGE. or. INSTALL_RATE=c(1,2,3,4,5),
+ other.PRESENT_RES.OR.JOB=c(0,1,2,3),
+ other.HISTORY.or.CHK_ACCT.or.SAV_ACCT=c(0,1,2))
> bins_adj = woebin(dt_f, y="DEFAULT", breaks_list=breaks_adj)

[INFO] creating woe binning ...
> dt_woe_list = lapply(dt_list, function(x) woebin_ply(x, bins_adj))

[INFO] converting into woe values ...
```

#Ran a regression linear model using DEFAULT as a target on the newly created WOE values.

```
> m1 = glm(DEFAULT ~ ., family = binomial(), data = dt_woe_list$train)
> m_step = step(m1, direction="both", trace = FALSE)
> m2 = eval(m_step$call)
> pred_list = lapply(dt_woe_list, function(x) predict(m2, x, type='response'))

> perf = perf_eva(pred = pred_list, label = label_list)
[INFO] The threshold of confusion matrix is 0.3163.

> card = scorecard(bins_adj, m2)
> score_list = lapply(dt_list, function(x) scorecard_ply(x, card))
> perf_psi(score = score_list, label = label_list)
}
```

The charts below show how well the model performed. It appears that the test and training data are very similar. Both the training and test data proved that the model can discern from a good-risk or bad-risk customer data 80% of the time.



PERFORMANCE EVALUATION

The performance evaluation (perf_psi) calculates the stability for the score and variables. It is used to run ongoing tests on the model to ensure the data received from the population does not change. This is important because overtime the data will change and the model should be tested and calibrated often. The formula is as follows:

$$PSI = \sum((Actual\% - Expected\%) * (\ln(\frac{Actual\%}{Expected\%})))$$

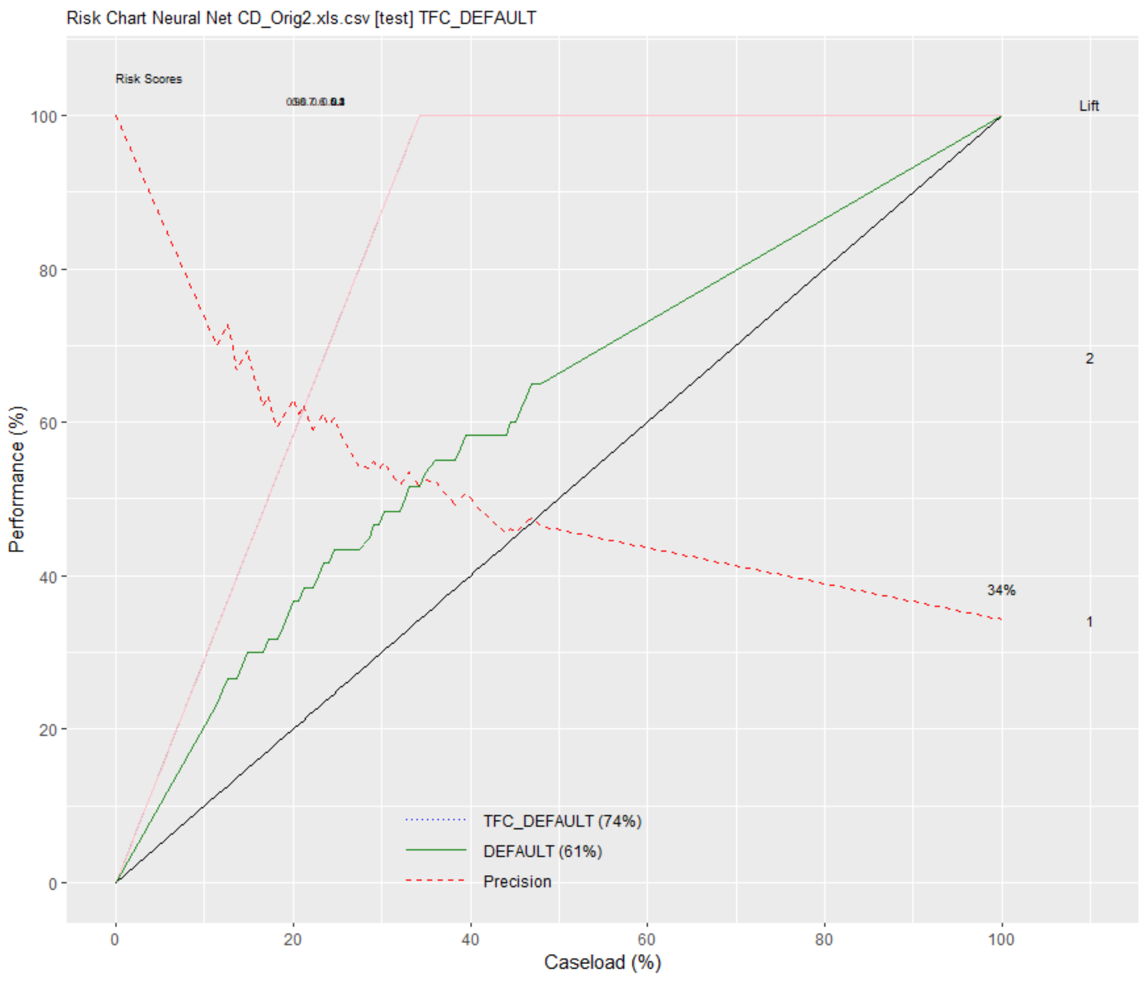
When the PSI is less than 0.1 there is no real change in new data. When the PSI reaches 0.1 - 0.25 there is some minor change, and the model should be evaluated. Anything greater than 0.25 signifies a major change and model will need to be changed.

```
{
$pic
$pic$score


$psi
}

   variable    dataset        psi
1:    score train_test 0.08494946
```

The PSI score for our model is fine at this time, however as previously stated, this should be tested often.

Risk Chart

The use of a risk chart is most helpful when evaluating models of fraud. As seen below:


Risk Chart Neural Net CD_Orig2.xls.csv [test] TFC_DEFAULT
Risk Scores
Lift
Performance (%)
Caseload (%)
TFC_DEFAULT (74%)
DEFAULT (61%)
Precision