SCORECARD CREATION

Dr. Aric LaBarr

Institute for Advanced Analytics

can do some variable selection with IVs,

Binning individidual variables

Process Flow

Data Collection
- Variable Selection
- Sample Size
- Sample / Performance Window

Data Cleaning
- Eliminate Duplicates
- Examine / Remove Outliers

Variable Grouping and Selection
- Weights of Evidence (WOE)
- Information Value (IV)
- Gini Criterion

Initial Scorecard Creation
- Logistic Regression
- Accuracy
- Threshold
- Assessment

Reject Inference
- Remove bias resulting from exclusion of rejects

Final Scorecard Creation
- Final Model Assessment

INITIAL SCORECARD CREATION

Initial Scorecard Model

- The scorecard is (typically) based on a logistic regression model:

$$logit(p) = \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k$$

- *p* is the posterior probability of default (PD) given the inputs.

Blasphemy!!!

- Wait, so I’m going through all that math just to throw things back into the logistic regression I was trying to avoid in the first place?!?!?!?!

Different Inputs

- Instead of using the original variables for the model, scorecard models have the binned variables as their foundation.

1 defaulted

predictor

put woe
instead of
binned variable
in logistic reg

Diff obs so they re diff ppl with diff score but based on what you did it is the same bin so model treat them same. How strong is this bin at predicting

Observation	Target	Bureau Score	Bureau Score Bin (R)	Bureau Score WOE (R)
1	0	757	716 – 765	1.0914
2	1	NA	Missing	-0.6972
3	0	626	605 – 629	-0.9586
4	0	693	665 – 716	0.1776
5	0	706	665 – 716	0.1776
6	0	673	665 – 716	0.1776
7	0	730	716 – 765	1.0914

Odds ratio gone now cuz we have coeff of WOE now not coeff of bureau score (inrease by 1)

Different Inputs

- Instead of using the original variables for the model, scorecard models have the binned variables as their foundation.

needed this categorical rep to get WOE values

Observation	Target	Bureau Score	Bureau Score Bin (R)	Bureau Score WOE (R)
1	0	757	716 – 765	1.0914
2	1	NA	Missing	-0.6972
3	0	626	605 – 629	-0.9586
4	0	693	665 – 716	0.1776
5	0	706	665 – 716	0.1776
6	0	673	665 – 716	0.1776
7	0	730	716 – 765	1.0914

This numerical variable that was once just tell me the value of your credit score is now a numerical representation of the strength of the bean,
and those bins were created to best predict the target.
So those numerical values are now directly tied to the target variable in a way that best predicts them.
So we no longer have the original credit score or bureau score values. Have instead best represetnation of score. Variable transformation.

Different Inputs

- Instead of using the original variables for the model, scorecard models have the binned variables as their foundation.

cont to categorical to cont. We dont do this always cuz otherwise lose interpretabiliy.

- Inputs are still treated as continuous.
- All variables now on the same scale.
- Model **coefficients** are desired output for the scorecard.
- **Coefficients** now serve as measures of variable importance.

Bureau Score WOE (R)
1.0914
-0.6972
-0.9586
0.1776
0.1776
0.1776
1.0914

So now not only can I compare every continuous variable because every continuous variables on the same scale,
every categorical variables on the same scale as my continuous variables.
And so all of those betas truly represent a notion of variable importance and variable strength.
I can literally compare any variable I'd like, any variable I'd like, and that's the benefit of this.
This is the underlying piece of putting everything into a point system, right
Because that's what a scorecard is. Everything's on a point system in a scorecard.

Different Inputs

give me result from smbinning funciton

this is a list that calling from. list has result of smbinning.

downside of smbinning is cant put all variables at a time. Have to put in 1 at a time. So looped and put result in a list

```
smbinning.gen(df = train, ivout = result_all_sig$bureau_score,
          chrname = “bureau_score_bin")
```

STeps taken: 1) Did sm binning on bureau score to figure values of categories, 2) smbinning.generate will create those categorical variables for u to put inside data.
if you do not want to build a scorecard,but you want to be able to just put all categorical variables into your model, whatever your model is. if you want to just look at only categorical variables into a logistic regression model, completely fine (Fall 1 binned set of variables, labarr gave us binn and we inputted it)

Different Inputs

```
smbinning.gen(df = train, ivout = result_all_sig$bureau_score,
              chrname = "bureau_score_bin")
```

Observation	Target	Bureau Score	Bureau Score Bin (R)	Bureau Score WOE (R)
1	0	757	716 – 765	1.0914
2	1	NA	Missing	-0.6972
3	0	626	605 – 629	-0.9586
4	0	693	665 – 716	0.1776
5	0	706	665 – 716	0.1776
6	0	673	665 – 716	0.1776
7	0	730	716 – 765	1.0914

Different Inputs

Every obs you are extracting out what bin you are in, once i know what bin i look up row in my dictorionary sort of and i look at value of weight of eveidence column.

sm binning labels missing category as 00 ie 'o'. c
Basically you have dictionary with values, you need to compare your categorical values to that dictionary and just replace with numbers of WOE

```
for (i in 1:nrow(train)) {
    bin_name <- "bureau_score_bin"
    bin <- substr(train[[bin_name]][i], 2, 2)
    woe_name <- "bureau_score_WOE"

    if(bin == 0) {
      bin <- dim(result_all_sig$bureau_score$ivtable)[1] - 1
      train[[woe_name]][i] <- result_all_sig$bureau_score$ivtable[bin, "WoE"]
    } else {
      train[[woe_name]][i] <- result_all_sig$bureau_score$ivtable[bin, "WoE"]
    }
}
```

Different Inputs

this is the goal to get it. Now just do logisitic regression Fall 1. Whether you had categorical or cont variables originally, all of them represnted by WOE column

Observation	Target	Bureau Score	Bureau Score Bin (R)	Bureau Score WOE (R)
1	0	757	716 – 765	1.0914
2	1	NA	Missing	-0.6972
3	0	626	605 – 629	-0.9586
4	0	693	665 – 716	0.1776
5	0	706	665 – 716	0.1776
6	0	673	665 – 716	0.1776
7	0	730	716 – 765	1.0914

Initial Scorecard

smbinning requires non defaulters as flagged as 1 (Good). But we are modelling Default in actually modelling.

8 variable WOE value sonly - 8 cuz got it from IV value. used IV for vairable selection.

```
initial_score <- glm(data = train, bad ~ tot_derog_WOE +
                                         tot_tr_WOE +
                                         age_oldest_tr_WOE +
                                         tot_rev_line_WOE +
                                         rev_util_WOE +
                                         bureau_score_WOE +
                                         down_pyt_WOE +
                                         ltv_WOE,
                     weights = train$weight, family = "binomial")
summary(initial_score)
```

INformamtion value tells you for all the levels of variable how does it predict Y vs WOE is level by level

cuz we have rare event problem. weights assigned during modelling

There are two ways of being able to undo the bias inside of the actual under sampling slash oversampling technique.
The first way is to be able to just adjust the intercept. The second ways to do weighted observations.
If you remember, adjusting the intercept only works if you're completely, 100% sure that those variables are right.
Okay. Weighted observation works better when you do not know if the variables you have are right.
And there was an observer, there was a number of observations limit in there as well.
We have over a thousand observations. We don't know if these variables are 100% right.
We're going to use weighted observations.

in world outside banking world, can remove variable that have high p value. IN banking world, use IV to variable select not p value.

Initial Scorecard

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6969  -0.7432  -0.4273  -0.1679   3.3704

Coefficients:
                  Estimate Std. Error z value             Pr(>|z|)
(Intercept)       -2.98190    0.04101 -72.706 < 0.0000000000000002 ***
tot_derog_WOE     -0.14478    0.08285  -1.747              0.08055 .
tot_tr_WOE        -0.04041    0.12726  -0.318              0.75084
age_oldest_tr_WOE -0.28207    0.09501  -2.969              0.00299 **
tot_rev_line_WOE  -0.38840    0.07963  -4.878           0.00000107 ***
bureau_score_WOE  -0.77495    0.05833 -13.286 < 0.0000000000000002 ***
rev_util_WOE      -0.23923    0.07643  -3.130              0.00175 **
down_pyt_WOE      -0.39379    0.14828  -2.656              0.00791 **
ltv_WOE           -0.86395    0.10116  -8.541 < 0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 6910.4  on 4376  degrees of freedom
Residual deviance: 6080.4  on 4368  degrees of freedom
AIC: 6185.1
```

How come variabel high p value but still got selected based on IV? So all by itself, that variable has predictive power on Y. But like every model, once we account for everything else in the model, that's what information each variable provides. So remember all of these p values are relative. what we refer to as type three tests - assuming every other variables in the model. What is the significance of this variable

So again, you can do variable selection like we talked about in the fall. Not interpreting coeff values here cuz its of WOE.

You can easily do that. There's no problem here. You can throw this into a lasso, you can do forward, you can do backward, you can do stepwise. But in banking you would not do this all. just output below and done.
In banking treat variables individually not holistically.

Model Evaluation

same as before

Now alll that is left is points column

- Variable significance – review using “standard” output of logistic regression, but don’t forget **business logic**.
- Overall performance of model – AUC (area under ROC curve, also called c) is the most popular criterion.
- This is only a **preliminary scorecard**.
- Final scorecard is created after reject inference is performed.

Model Evaluation

gives all metric results

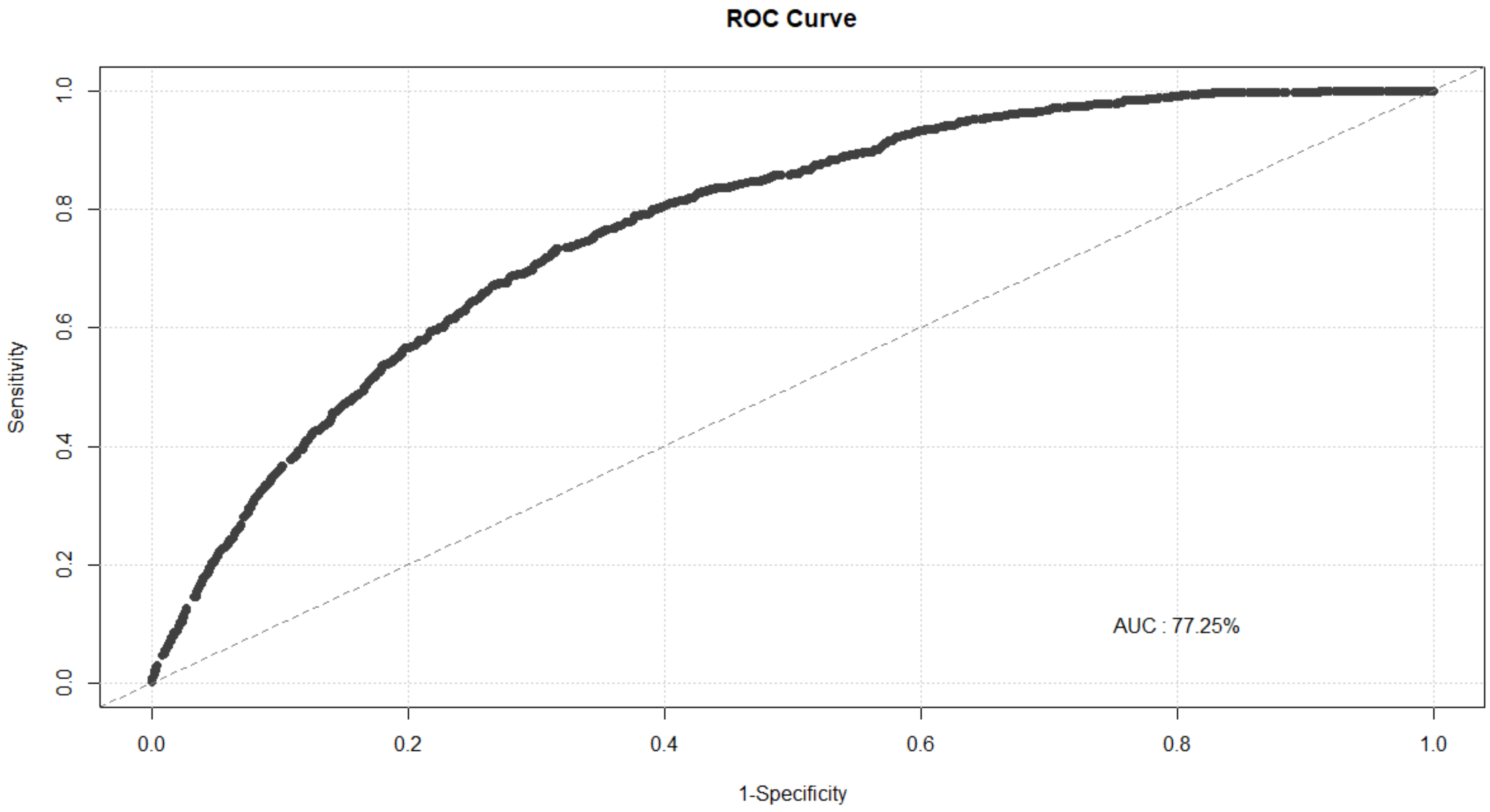
target in my model

get your predicted values prob

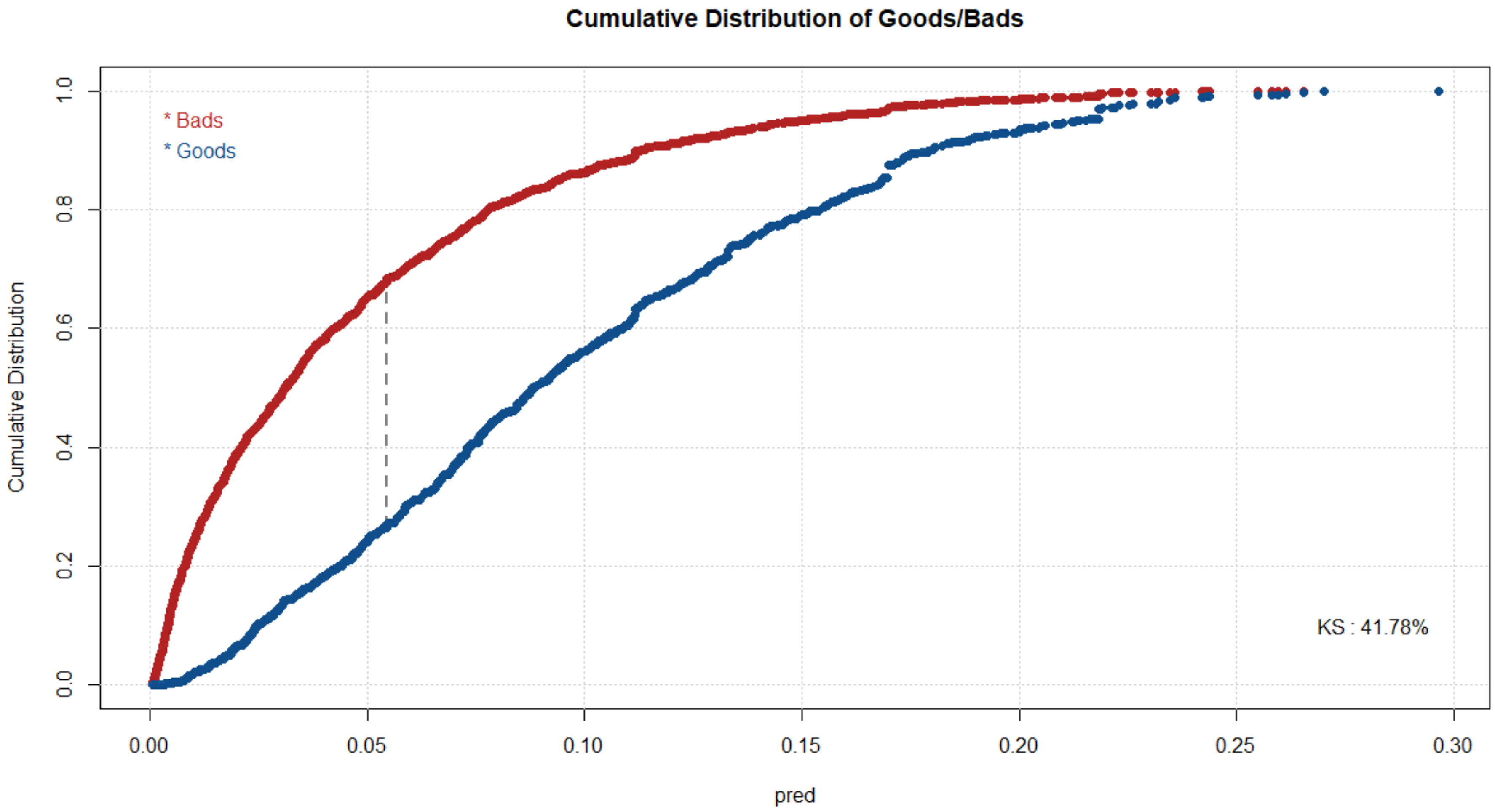
```
smbinning.metrics(dataset = train, prediction = "pred",
                  actualclass = "bad", report = 1)
smbinning.metrics(dataset = train, prediction = "pred",
                  actualclass = "bad", plot = "ks")
smbinning.metrics(dataset = train, prediction = "pred",
                  actualclass = "bad", plot = "auc")
```

1 means everything - AUC, KS, Youden Index, Optimal Cutoff

Model Evaluation



Model Evaluation



To summarize, to build model, i am not using original variables. I am using WOE of variables. If variables are cont, made them categorical and calculated WOE. If categorical i just calculated WOE with dictionary lookup. At end every variable represented by WOE correspondence. That goes into logistic reg, every variable is on same scale. Now all variables cont and cat are on same scale think standardization (where you std both categorical same scale as cont).

So now everything can be compared on a single metric.
That single metric are those beta coefficients from the logistic regression model.
And those beta coefficients are going to underlie the points that we're about to calculate. Thats why we did what we did.

Now, some of the other machine learning techniques we talked about, we have other things other than betas to summarize variable importance. eg Random Forest will give you that. But in logistic regression, we never had that. We never had one number that we could just draw upon to compare cuz things on diff scale.

SCALING THE SCORECARD

score card pt range all depends on you. Why they picked 350 to 850 is arbitraty.

You are gonna pick 2 points and entire scorecard built off of that. Algebra eqn. What are those two points? Essentially what we do is we call them factor and offset.
But that's really what you're looking at. You're looking at a slope and you're looking at an intercept for your point system.

Scaling the Scorecard

- The relationship between odds and scores is represented by a linear function:

$$Score = Offset + Factor \times \log(odds)$$

- If the scorecard is developed using "odds at a certain score" and "points to double the odds" (PDO), $Factor$ and $Offset$ can be calculated using the simultaneous equations:

$$\left.\begin{aligned} Score &= Offset + Factor \times \log(odds) \\ Score + PDO &= Offset + Factor \times \log(2 \times odds) \end{aligned}\right\}$$

2 eqn 2 unknowsn

So how do we do something like that instead of asking someone for an offset and the factor (2 points)? What we do is we ask them for two things. Give me the odds at a specific score. And then give me the points to double the odds (PDO)

Scaling the Scorecard

- Solving the equations for PDO, you get the following results:

$$PDO = Factor \times \log(2)$$

- Therefore,

$$Factor = \frac{PDO}{\log(2)}$$

$$Offset = Score - Factor \times \log(odds)$$

Scaling the Scorecard – Example

- If a scorecard were scaled where the developer wanted odds of 50:1 at 600 points and wanted the odds to double every 20 points (PDO = 20), *Factor* and *Offset* would be:

$$Factor = \frac{20}{\log(2)} = 28.8539$$

$$Offset = 600 - (28.8539 \times \log(50)) = 487.123$$

- Therefore, each score corresponding to each set of odds can be calculated as follows:

$$Score = 487.123 + 28.8539 \times \log(odds)$$

Scaling the Scorecard – Example

Score eqn corresponds to odds.
Get odds from model.
Scorecards can be easily built into any database cuz its just an eqn, once you have eqn you are done.

- If a scorecard were scaled where the developer wanted odds of 50:1 at 600 points and wanted the odds to double every 20 points (PDO = 20), *Factor* and *Offset* would be:

$$Factor = \frac{20}{\log(2)} = 28.8539$$

$$Offset = 600 - (28.8539 \times \log(50)) = 487.123$$

- Therefore, each score corresponding to each set of odds can be calculated as follows:

$$Score = 487.123 + 28.8539 \times \boxed{\log(odds)}$$

this is LHS of logistic reg eqn

Why people still love logistic regression!

Scaling the Scorecard – Example

Domain knowledge picked. ie if their odds were 50:1 , then their score was 600. If their odds were 20:1, their score is 620. Then all find is worst and best person obs, and that gives you your bounds on what you got thats how you ended up with 350 and 800. No One knew going in.

- If a scorecard were scaled where the developer wanted odds of 50:1 at 600 points and wanted the odds to double every 20 points (PDO = 20), *Factor* and *Offset* would be:

$$Factor = \frac{20}{\log(2)} = 28.8539$$

$$Offset = 600 - (28.8539 \times \log(50)) = 487.123$$

- Therefore, each score corresponding to each set of odds can be calculated as follows:

$$Score = 487.123 + 28.8539 \times \boxed{\log(odds)}$$

This is predicted value from logit function.

Scaling the Scorecard – Example

Score	Odds
600	50.0
601	51.8
604	57.4
.	
.	
.	
.	
620	100.0

Points Allocation

each variable each category. How well that category does in separating 1s and 0s. Diff variables stronger than others in predicting Y. So incl Beta j. So those 2 things tell you your points for that category level in that variable. last term is scale things up to add up together.

- The points allocated to attribute *i* of characteristic *j* are computed as follows:

$$Points_{i,j} = -\left(WOE_{i,j} \times \hat{\beta}_j + \frac{\hat{\beta}_0}{L}\right) \times Factor + \frac{Offset}{L}$$

Once we have WOE multiplied by beta, the strenght of category multiplied by strenght of variable, then we need to scale it with times by Factor. Then we add in the OFfset, divide intercept by all points.

- $WOE_{i,j}$: Weight of evidence for attribute *i* of characteristic *j*
- $\hat{\beta}_j$: Regression coefficient for characteristic *j*
- $\hat{\beta}_0$: Intercept term from model
- L: Total number of characteristics
- Points typically rounded to nearest integer.

intercept over L. L is total variables in model. Every variable gets small piece of intercept. There are no intercepts in ML algos. All you need is sth that representeg is beta not, the strenght of variable - eg in RF get variable imp. Thats all u need instead of betas. So again, you can easily put this on top of a machine learning model. You just have to convert this equation to something more machine learning-esq. Instead of beta put variable imp, instead of intercept dont put anything.

Points Allocation

```
pdo <- 20
score <- 600
odds <- 50
fact <- pdo/log(2)
os <- score - fact*log(odds)
var_names <- names(initial_score$coefficients[-1])

for(i in var_names) {

    beta <- initial_score$coefficients[i]
    beta0 <- initial_score$coefficients["(Intercept)"]
    nvar <- length(var_names)
    WOE_var <- train[[i]]
    points_name <- paste(str_sub(i, end = -4), "points", sep = "")

    train[[points_name]] <- -(WOE_var*(beta) + (beta0/nvar))*fact + os/nvar
}
```

Points Allocation

This table just gives you observation and predicted score. But you want scorecard itself, what variable has what points ? Next slide takes everyone's score and breaks into separate pieces.

Observation	Target	Variables…	Observation Score
1	0	…	599
2	1	…	524
3	0	…	537
4	0	…	561
5	0	…	578
6	0	…	583
7	0	…	672

Points Allocation

You do this for every variable and you can hand over someone a score card. Pick a category for every variable and add the points together to get final.

WOE for Bureau Score					
Group	Values	Event Count	Non-event Count	WOE	**Scorecard Points**
1	< 603	111	112	-1.32	**50.4**
2	604 – 662	378	678	-0.74	**64.1**
3	663 – 699	185	754	0.08	**83.5**
4	700 – 717	74	440	0.46	**92.4**
5	718 – 765	75	824	1.07	**106.9**
6	> 765	15	498	2.18	**133.1**
7	MISSING	80	153	-0.68	**65.5**
Total		**918**	**3,459**		

get these points for every variable (broken out)

everyone below score of 603, get 50.3 points

Now, the latest research is to take SHAPLEY values and use those as scorecards.
So if you think about it, every individual has their own scorecard as compared to a scorecard for all.
Regulator issue cuz you bias ppl out potential.

3 techniques just tell us how to infer the target. Step 1build model, score oyur rejects, make sure everythig is balanced, step 4 is target

At this point you have a model, a scoreacrd on top of mode. Now you can analyze your current customers anyway you like. You have model you have given loans to ppl. What you should not do is apply model to ppl applying to get new loan. (model bias) but we have reject bias. Done here if banks just want to understand current customers.

REJECT INFERENCE

Process Flow

Data Collection
- Variable Selection
- Sample Size
- Sample / Performance Window

Data Cleaning
- Eliminate Duplicates
- Examine / Remove Outliers

Variable Grouping and Selection
- Weights of Evidence (WOE)
- Information Value (IV)
- Gini Criterion

Initial Scorecard Creation
- Logistic Regression
- Accuracy
- Threshold
- Assessment

Reject Inference
- Remove bias resulting from exclusion of rejects

Final Scorecard Creation
- Final Model Assessment

Reject Inference

- **Reject inference** is the process of inferring the status of the rejected applicants based on the accepted applicants model in an attempt to use their information to build a scorecard that is representative of the entire applicant population.
- Reject inference is about solving sample bias so that the development sample is similar to the population to which the scorecard will be applied.

problem is rejected dataset
doesnt have target

Rejected Inference

- Can we develop a scorecard without rejected applications? **YES!**
- Is it **legally permissible** to develop a scorecard without rejected applications? **YES!**
- If yes, then how **biased** would the scorecard model be? **DEPENDS!**

- *"My suggestion is to develop the scorecard using what data you have, but start saving rejected applications ASAP."*
  Raymond Anderson, Head of Scoring at Standard Bank Africa, South Africa

But I can't tell you. Of the people you didn't give loans to, did you make the right decision?

Why Reject Inference?

model is right but underlying data is missing reject inference.

- Initial scorecard used only **known** good and bad loans (accepted applicants only) – also called "behavioral scoring"
- Reduce bias in model and provide risk estimates for the "through-the-door" population – also called "application scoring"
- Comply with regulatory requirements (FDIC, Basel)
- Provide a scorecard that is able to generalize better to the entire credit application population.

bankers get together in Swiss. US pays attention but not follow

Reject Inference Techniques

- Three common techniques for reject inference:
  1. Hard Cutoff Augmentation
  2. Parceling Augmentation
  3. Fuzzy Augmentation (DEFAULT in SAS)

draw a line in sand, like youden index. Boave and below 1 and 0

Take your original model that you built on the accepts.
Yes, it's bias, but it's the only model we've got score the rejects with that model, basically create a target variable, create a target variable for the rejects,
then combine the rejects and the accepts together into one huge data set and rebuild the entire modeling process.
Go all the way back to binning, rebuild your variables, rebuild your model, do all of that again,
and you will notice that the variables start changing y because now you have new information.

Hard Cutoff Augmentation

For your final model, cant have 50% rejects and 50% accepts. Have to reflect population 75-25.

1. Build a scorecard model using the known good/bad population (accepted applications)
2. Score the rejected applications with this model to obtain each rejected applicant's probability of default and their score on the scorecard model.
3. Create weighted cases for the rejected applicants – weight applied is the "rejection rate" which adjusts the number of sampled rejects to accurately reflect the number of rejects from population.

Two undersampling problems (see OneNote screenshot): 1) Initial model, you had 3% default rare eevent that you solved using weights=weight in glm (Rare event). 2) But now after combing reject inference you have 50% reject and 50% accept rows. BUT real population is 75% ppl accepted for loan, 25% reject. So our combined data set should look like real population so you just undersample the reject inference to match 75-25 in population.

You have luxury to just undersample to make it go from 50-50 to 75-25 but initially when you modelled default it waas 97-3, so didnt have luxury to make model data be 97-3. Had to oversample

Hard Cutoff Augmentation

4. Set a cut-off score level above which applicant is deemed good and below applicants deemed bad.
5. Add inferred goods and bads with known goods and bads and rebuild scorecard.

Take your rejects, score them based on your model (score means create target variable), Inffer whetehr they default or not. Guess target for rejects, then combine itall into 1 data set. Then go back to modelling process - binning re bin, rebuild model, you will notice variable cahnge due to o new info.

Hard Cutoff Augmentation

pred probability or score

create new default.bad=1.

```
rejects_scored$pred <- predict(initial_score, newdata = rejects_scored,
                               type = 'response')
rejects$bad <- as.numeric(rejects_scored$pred > 0.0545)
rejects$weight <- ifelse(rejects$bad == 1, 2.80, 0.59)
rejects$good <- abs(rejects$bad - 1)

comb_hard <- rbind(accepts, rejects)
```

first model Youden Index (could be F1 too)

add good col, cuz your accepts data set has it.

harder way to do it, just do undersample way. instead of 2 weghts.

NOW repeat all sets form before

Hard Cutoff Augmentation

```
rejects_scored$pred <- predict(initial_score, newdata = rejects_scored,
                               type = 'response')

rejects$bad <- as.numeric(rejects_scored$pred > 0.0545)
rejects$weight <- ifelse(rejects$bad == 1, 2.80, 0.59)
rejects$good <- abs(rejects$bad - 1)

comb_hard <- rbind(accepts, rejects)
```

how you got thi ? Used youden index but can use anything. Can use KS stat or F1 score or anything.

Hard Cutoff Augmentation

```
rejects_scored$pred <- predict(initial_score, newdata = rejects_scored,
                               type = 'response')
rejects$bad <- as.numeric(rejects_scored$pred > 0.0545)
rejects$weight <- ifelse(rejects$bad == 1, 2.80, 0.59)
rejects$good <- abs(rejects$bad - 1)

comb_hard <- rbind(accepts, rejects)
```

watch the video on how to do weights twice. But if you got enough data, then instead just sample down rejects to reflect population.

after rbind 2 sets, re do everything. We are extrapolating accepts model to reject cuz tahts all you got. Now after combining does model change its opinion

smbinning needed 1s to be good and 0s to be bad

Parceling Augmentation

nust change how you get 0s or 1s for target
1-3 steps al lsame.

This is just diff way of doing cutoff. Before no notion of grey area.

1. Build a scorecard model using the known good/bad population (accepted applications)
2. Score the rejected applications with this model to obtain each rejected applicant's probability of default and their score on the scorecard model.
3. Create weighted cases for the rejected applicants – weight applied is the "rejection rate" which adjusts the number of sampled rejects to accurately reflect the number of rejects from population.

parcelling means break into smaller groups

could be pred
prob. high
score is low
prob. work
oppposte.

Parceling Augmentation

4. Define score ranges manually or automatically with simple bucketing.
5. The inferred good/bad status of the rejected applicants will be assigned **randomly** and proportional to the number of goods and bads in the accepted population within each score range.
6. If desired, apply the event rate increase factor to P(bad) to increase the proportion of bads among the rejects (oversampling with the rejects)
7. Add the inferred goods and bads back in with the known goods and bads and rebuild the scorecard.

Parceling Augmentation – Example

	Accepted Applicants				Rejected Applicants		
Score Range	# Bad	# Good	% Bad	%Good	Rejects	# Inferred Bad	# Inferred Good
< 655	0	0	0%	0%	5	?	?

This range is 'never seen before range'. Is there anyone in our applicants that we accepted?
So the stuff the model was built off of that we've ever seen get a score that low?
Nope. Never seen someone with a score that low.
Welcome to the world of extrapolation. We are literally about to score people who we've never seen before in terms of something that low.
But there are on the rejected applicant side, five individuals, five individuals who score that low, even though we've never seen people score that low for the accepted. Okay, so what do we do?

Parceling Augmentation – Example

	Accepted Applicants				Rejected Applicants		
Score Range	# Bad	# Good	% Bad	%Good	Rejects	# Inferred Bad	# Inferred Good
< 655	0	0	0%	0%	5	5	0

Assume bad if no information to prove otherwise

in this range in our model our evidence is 0%, but assume bad cuz no info otherwise to disprove it.

Parceling Augmentation – Example

	Accepted Applicants				Rejected Applicants		
Score Range	# Bad	# Good	% Bad	%Good	Rejects	# Inferred Bad	# Inferred Good
< 655	0	0	0%	0%	5	5	0
655 – 665	300	360	45.5%	54.5%	190	?	?

Parceling Augmentation – Example

	Accepted Applicants				Rejected Applicants		
Score Range	# Bad	# Good	% Bad	%Good	Rejects	# Inferred Bad	# Inferred Good
< 655	0	0	0%	0%	5	5	0
655 – 665	300	360	45.5%	54.5%	190	86	?

somebanks may do reject bump because say if you give rejected applicants loans, then for same score range they would still have defaulted at higher rate. Thats why you artifically inflate bad%

you are applying this same % to rejects based on original data

$0.455 \times 190 \approx 86$

Randomly assign!

190 is total in reject set for that range.
45.5% of that is inferred bad defaulters

Parceling Augmentation – Example

	Accepted Applicants				Rejected Applicants		
Score Range	# Bad	# Good	% Bad	%Good	Rejects	# Inferred Bad	# Inferred Good
< 655	0	0	0%	0%	5	5	0
655 – 665	300	360	45.5%	54.5%	190	86	114

$$190 - 86 = 114$$

Parceling Augmentation – Example

Banks first use external info to make decisions until they get enough info within bank.

	Accepted Applicants				Rejected Applicants		
Score Range	# Bad	# Good	% Bad	%Good	Rejects	# Inferred Bad	# Inferred Good
< 655	0	0	0%	0%	5	5	0
655 – 665	300	360	45.5%	54.5%	190	86	114
665 – 675	450	700	39.1%	60.9%	250	98	152
…	…	…	…	…	…	…	…

```
parc <- seq(500, 725, 25)

accepts_scored$Score_parc <- cut(accepts_scored$Score, breaks = parc)
rejects_scored$Score_parc <- cut(rejects_scored$Score, breaks = parc)

table(accepts_scored$Score_parc, accepts_scored$bad)

parc_perc <- table(accepts_scored$Score_parc, accepts_scored$bad)[,2] /
                   rowSums(table(accepts_scored$Score_parc, accepts_scored$bad))

rejects$bad <- 0

rej_bump <- 1.25

for(i in 1:(length(parc) - 1)) {
    for(j in 1:length(rejects_scored$Score)) {
        if((rejects_scored$Score[j] > parc[i]) &
           (rejects_scored$Score[j] <= parc[i+1]) &
           (runif(n = 1, min = 0, max = 1) < (rej_bump*parc_perc[i]))) {
            rejects$bad[j] <- 1
        }
    }
}

table(rejects_scored$Score_parc, rejects$bad)
rejects$weight <- ifelse(rejects$bad == 1, 2.80, 0.59)
rejects$good <- abs(rejects$bad - 1)

comb_parc <- rbind(accepts, rejects)
```

```
parc <- seq(500, 725, 25)

accepts_scored$Score_parc <- cut(accepts_scored$Score, breaks = parc)
rejects_scored$Score_parc <- cut(rejects_scored$Score, breaks = parc)

table(accepts_scored$Score_parc, accepts_scored$bad)

parc_perc <- table(accepts_scored$Score_parc, accepts_scored$bad)[,2] /
                   rowSums(table(accepts_scored$Score_parc, accepts_scored$bad))

rejects$bad <- 0

rej_bump <- 1.25

for(i in 1:(length(parc) - 1)) {
    for(j in 1:length(rejects_scored$Score)) {
        if((rejects_scored$Score[j] > parc[i]) &
           (rejects_scored$Score[j] <= parc[i+1]) &
           (runif(n = 1, min = 0, max = 1) < (rej_bump*parc_perc[i]))) {
            rejects$bad[j] <- 1
        }
    }
}

table(rejects_scored$Score_parc, rejects$bad)
rejects$weight <- ifelse(rejects$bad == 1, 2.80, 0.59)
rejects$good <- abs(rejects$bad - 1)

comb_parc <- rbind(accepts, rejects)
```

Fuzzy Augmentation

1. Build a scorecard model using the known good/bad population (accepted applications)
2. Score the rejected applications with this model to obtain each rejected applicant's probability of being good, P(Good), and probability of being bad, P(Bad).
3. **Do not assign a reject to a good/bad class** – create **two weighted cases** for each rejected applicant using P(Good) and P(Bad).

Represent good and bad side of Ann Marie. Everyone angel devil same have both in model. but good evan willl have a higher weight. Every obs have good and bad just have diff weights.

Rememer loony tune cartoon story (a devil and angel version of character)

If predicted prob is 0.5, the 1s and 0s for that observation get 50% weights. Weights is what gets adjusted. If pred prob is 0.8, then you get4 times the weight.

Fuzzy Augmentation

if everyone get os and 1s,
then replicating data so it will
change your popilation
sames 75-25

4. Multiply P(Good) and P(Bad) by the user-specific rejection rate to form frequency variables.
5. For each rejected applicant, create **two observations** – one observation has a frequency variable (rejection weight × P(Good)) and a target variable of 0; other observation has a frequency variable (rejection weight × P(Bad)) and a target variable of 1.
6. Add inferred goods and bads back in with the known goods and bads and rebuild the scorecard.

Fuzzy Augmentation

So no longer will your models sit there and be like, I've never seen anybody that look like that, so I'm just going to have to assume they're are bad.
Like, Nope, you've seen everybody look like that, both good and bad.

```
rejects_scored$pred <- predict(initial_score, newdata = rejects_scored,
                               type = 'response')

rejects_g <- rejects
rejects_b <- rejects

rejects_g$bad <- 0
rejects_g$weight <- (1 - rejects_scored$pred)*2.80
rejects_g$good <- 1

rejects_b$bad <- 1
rejects_b$weight <- (rejects_scored$pred)*0.59
rejects_b$good <- 0

comb_fuzz <- rbind(accepts, rejects_g, rejects_b)
```

So for this because the way the data set was built, I didn't under sample. i Did double weights instead

Reject Inference Techniques

- Three common techniques for reject inference:
  1. Hard Cutoff Augmentation
  2. Parceling Augmentation
  3. Fuzzy Augmentation (DEFAULT in SAS EM)

- There are other techniques as well, but are not as highly recommended.

Other Reject Inference Techniques

1. Assign all rejects to bads.
2. Assign rejects in the same proportion of goods to bads as reflected in the accepted data set.
3. Similar in-house model on different data.
4. Approve all applicants for certain period of time.
5. Clustering
6. Memory based reasoning

Other Reject Inference Techniques

1. Assign all rejects to bads
   - Appropriate only if approval rate is very high (ex. 97%) and there is a high degree of confidence in adjudication process.
2. Assign rejects in the same proportion of goods to bads as reflected in the accepted data set.
3. Similar in-house model on different data.
4. Approve all applicants for certain period of time.
5. Clustering
6. Memory based reasoning

Other Reject Inference Techniques

1. Assign all rejects to bads.
2. Assign rejects in the same proportion of goods to bads as reflected in the accepted data set.
   - Assignment done completely at random!
   - Valid only if current system has no consistency.
3. Similar in-house model on different data.
4. Approve all applicants for certain period of time.
5. Clustering
6. Memory based reasoning

Other Reject Inference Techniques

1. Assign all rejects to bads.
2. Assign rejects in the same proportion of goods to bads as reflected in the accepted data set.
3. Similar in-house model on different data.
   - Performance on similar products used as proxy.
   - Hard to pass by regulators.
4. Approve all applicants for certain period of time.
5. Clustering
6. Memory based reasoning

Other Reject Inference Techniques

1. Assign all rejects to bads.
2. Assign rejects in the same proportion of goods to bads as reflected in the accepted data set.
3. Similar in-house model on different data.
4. Approve all applicants for certain period of time.
   - Provides actual performance of rejects instead of inferred.
   - Might be “legal” problems…
5. Clustering
6. Memory based reasoning

Other Reject Inference Techniques

If bank is discriminating by rejecting for loan, then including reject inference to help fight those things. There to help with them. sometimes they may have 2 models - one they show regualtor, one they actual use.

1. Assign all rejects to bads.
2. Assign rejects in the same proportion of goods to bads as reflected in the accepted data set.
3. Similar in-house model on different data.
4. Approve all applicants for certain period of time.
5. Clustering
6. Memory based reasoning

need to have some accepts and bad in every cluster to be able to model

FINAL SCORECARD CREATION

Process Flow

Data Collection
- Variable Selection
- Sample Size
- Sample / Performance Window

Data Cleaning
- Eliminate Duplicates
- Examine / Remove Outliers

Variable Grouping and Selection
- Weights of Evidence (WOE)
- Information Value (IV)
- Gini Criterion

Initial Scorecard Creation
- Logistic Regression
- Accuracy
- Threshold
- Assessment

Reject Inference
- Remove bias resulting from exclusion of rejects

Final Scorecard Creation
- Final Model Assessment

Final Scorecard Creation

- The mechanics of building the final scorecard model are identical with the initial scorecard creation except that analysis is performed **after reject inference**.
- Accuracy Measurements:
  - Repeat review of the logistic model estimated parameters, life, KS, ROC, etc.

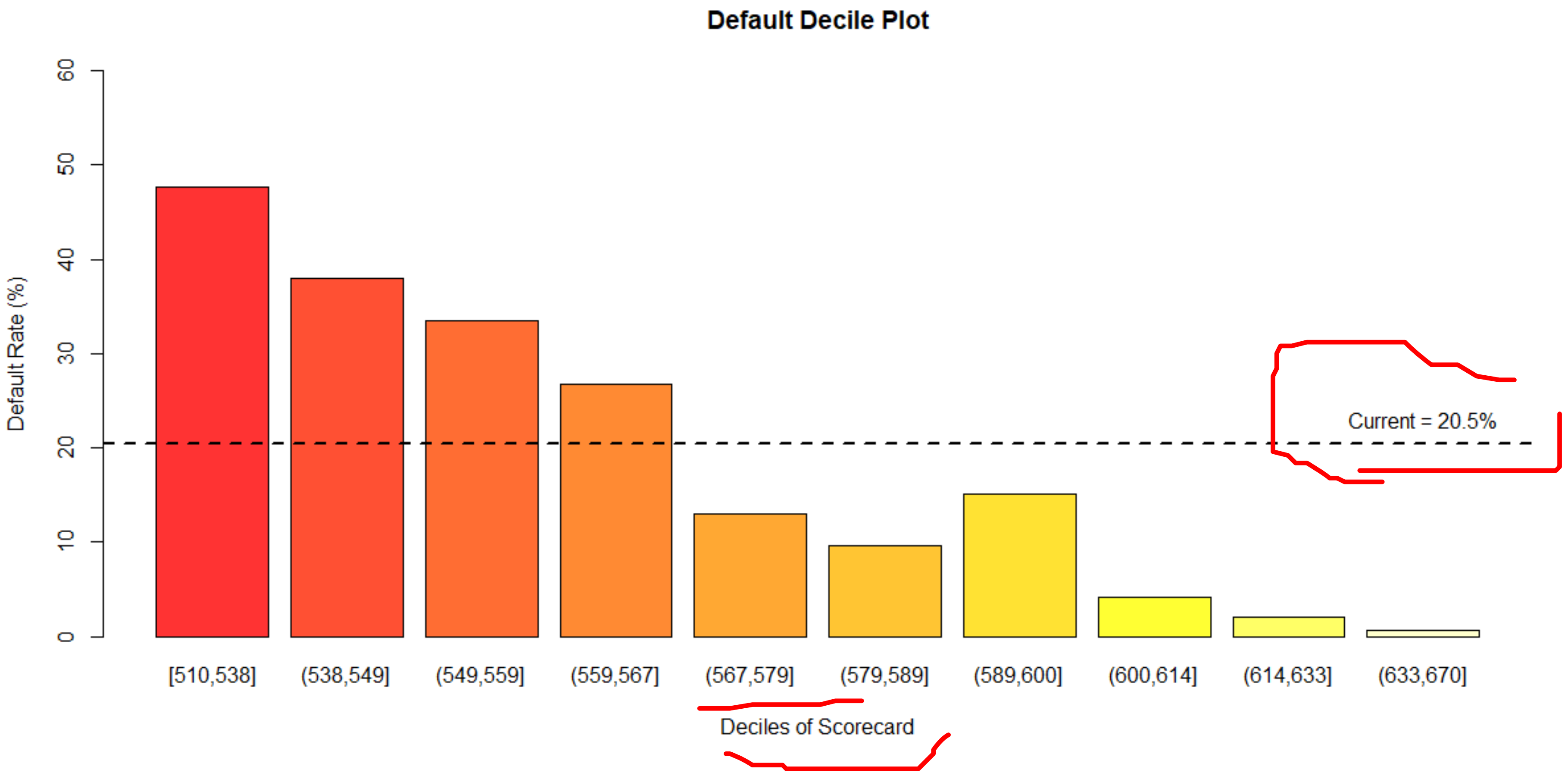
Defining Cut-off

dont just use any cutoff, evaluate the cost

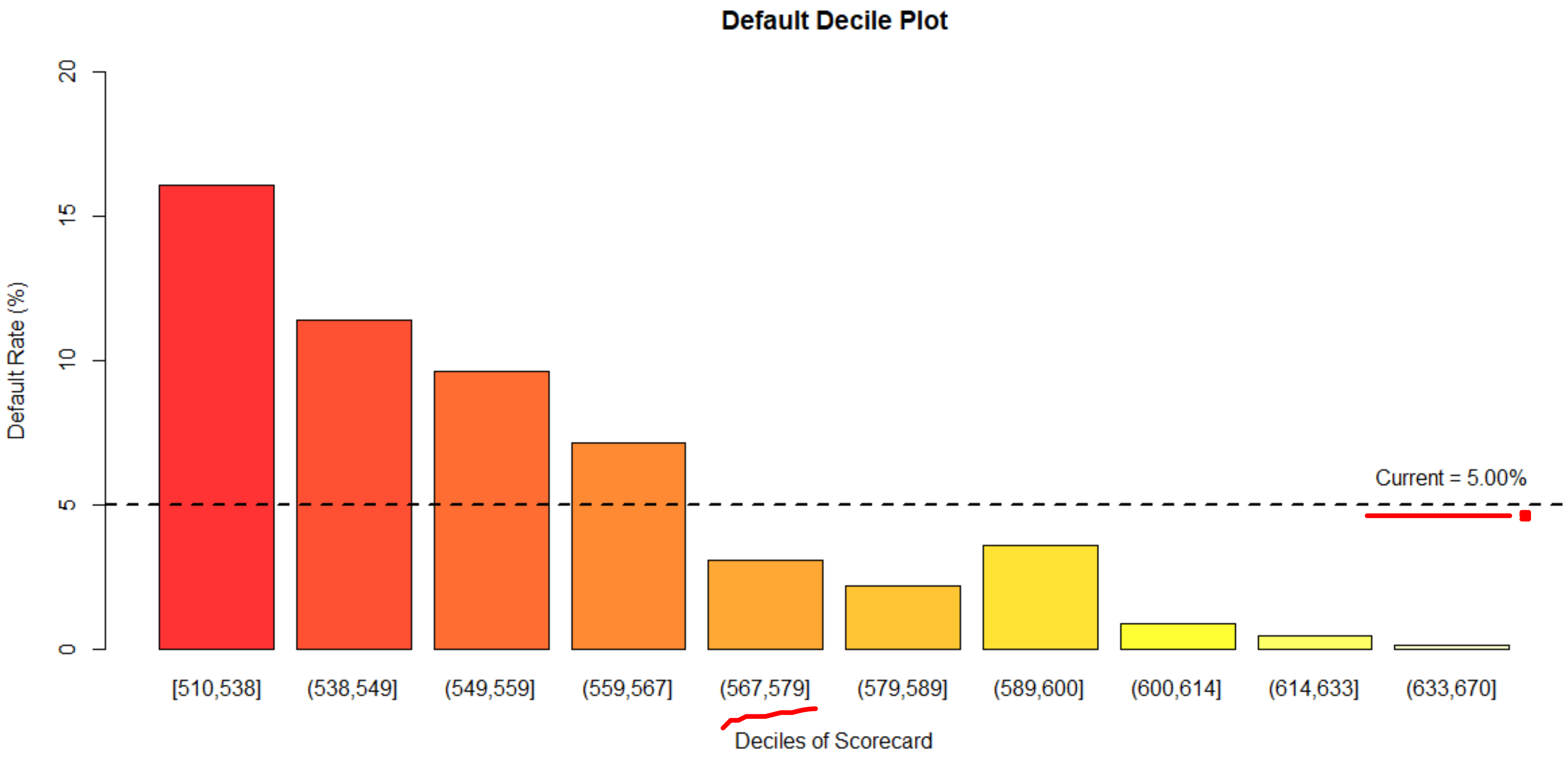
- A new scored should be better than the last in terms of one of the following:
  - Lower bad rate for the same approval rate.
  - Higher approval rate while holding the bad rate constant.

Default Decile Plot

this is how they do cutoff
based on cost. break down
in to 10 equal decile groups.



Default Decile Plot



Defining Cut-off

- Trade-off Plots:
  - The reference lines of approval rate and event (bad) rate are predefined by analyst.
  - How much risk are you willing to take on?

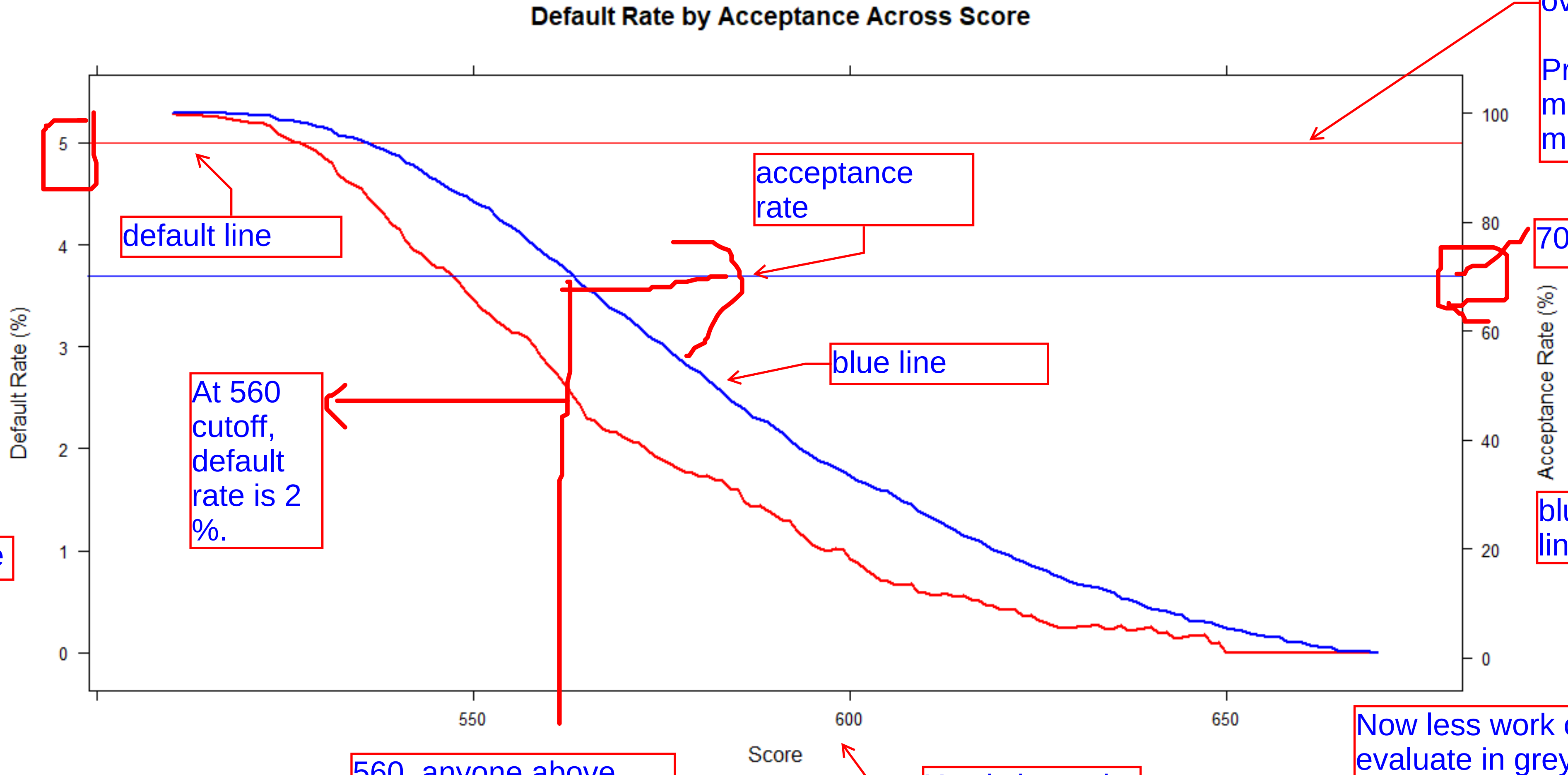
Defining Cut-off

Dual axis plot

What does the bank want to do? Does the bank want to lower its default? maximize profit. If you want to lower your default, I can keep your acceptance rate the same.Does the bank want to increase its customer base and they're okay with their default rate?

current default rate at bank is 5%. If want to keep this same at bank, then anyone over 500 accept them

Profit is i know how much each loan makes me $



default line

acceptance rate

70%

blue line

At 560 cutoff, default rate is 2 %.

red line

blue line

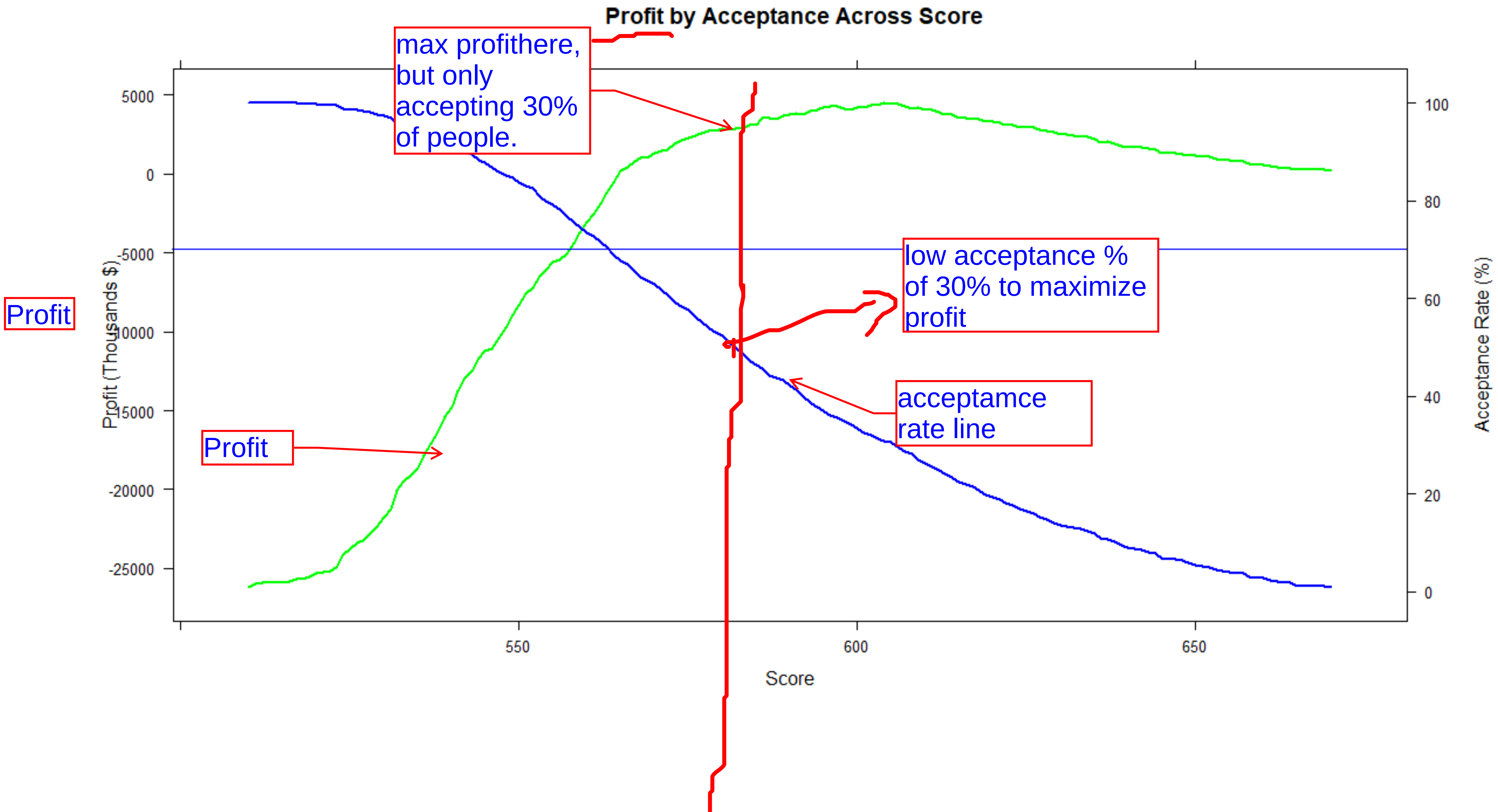
560, anyone above 560 you accept to keep same acceptance %

X axis is pred score from model (target)

Now less work cuz only have to evaluate in grey area ppl in middle (given default rate and acceptance rate of bank). So bankers only eval ppl in middle.

Defining Cut-off

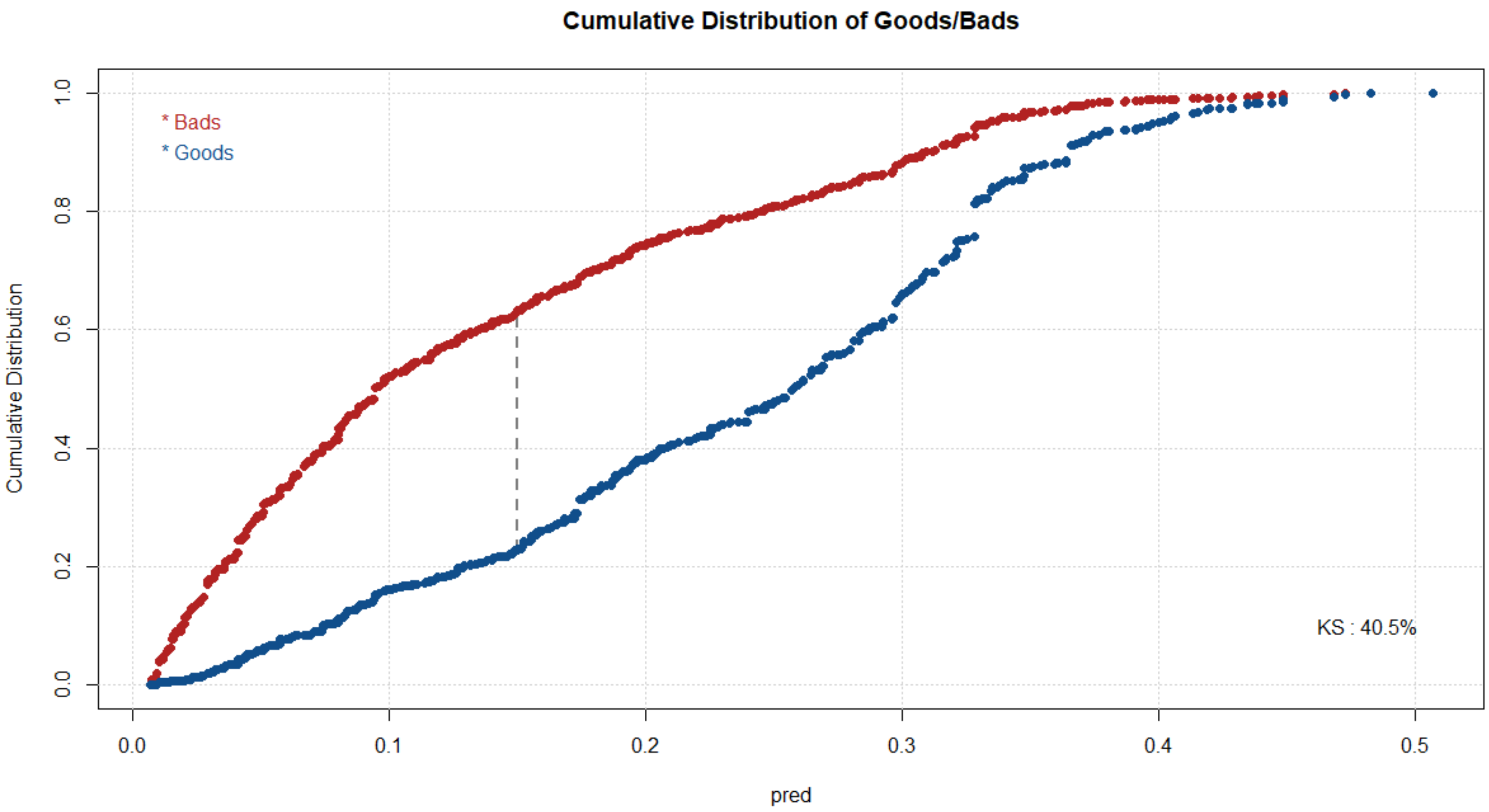
I know how much each loan makes me $, how much i lose when someone defaults.



Defining Cut-off

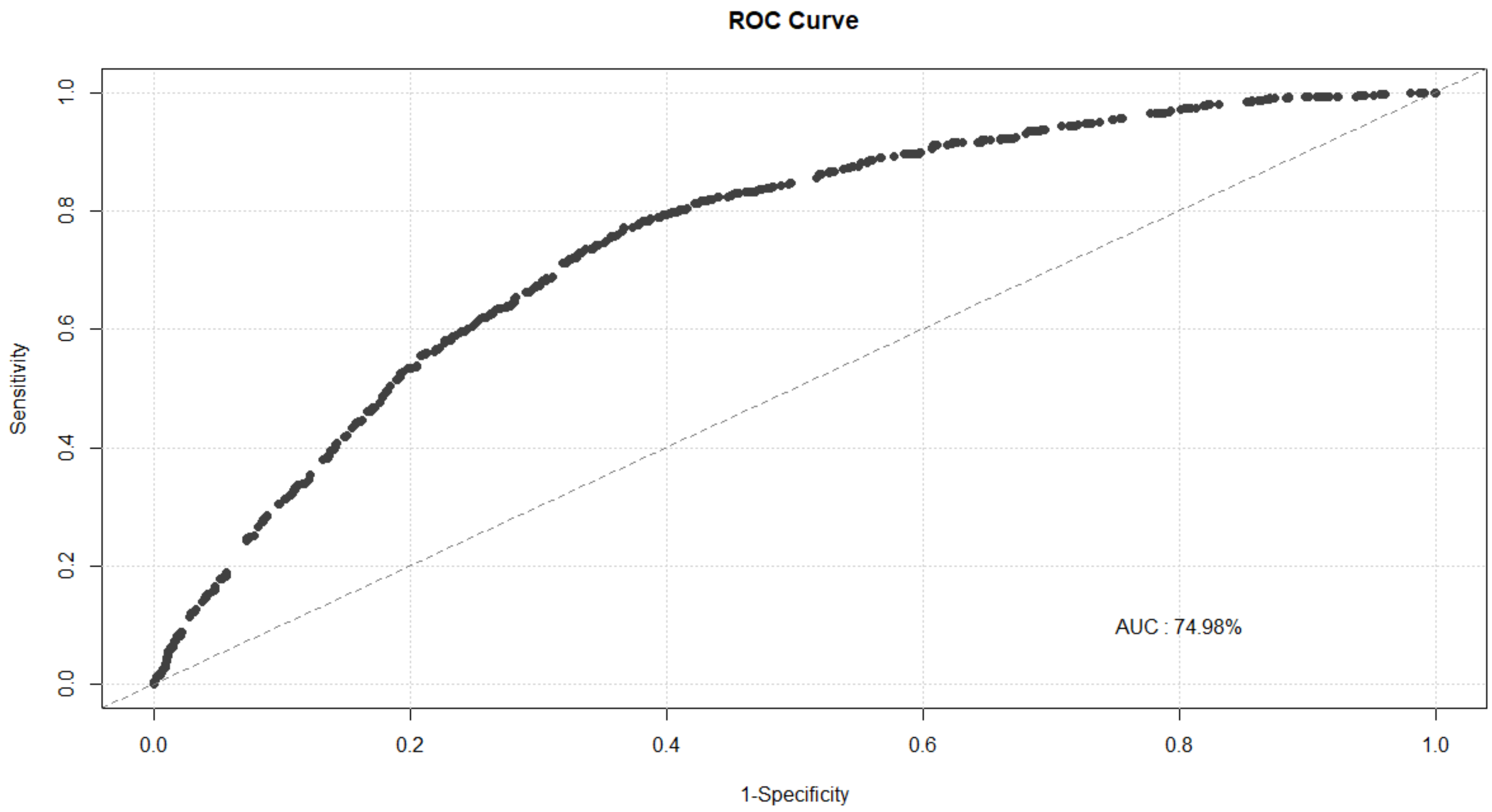
- Setting Multiple Cut-offs Example:
  - Anyone who scores above 210 points is accepted automatically.
  - Anyone who scores below 190 is declined.
  - Any scores in between 190 and 210 are referred to manual adjudication.

Final Scorecard – Example



Final Scorecard – Example

ROC curve good or bad? No idea, it depends. Need sth to compare with. Exisitng



CREDIT SCORING MODEL EXTENSIONS

Lack of Interactions

- Benefits of tree based algorithms are inherent interactions of every split of the tree.
  - Also a detriment to interpretation.

Multi-stage Model

- Benefits of tree based algorithms are inherent interactions of every split of the tree.
  - Also a detriment to interpretation.

- Multi-stage model:
  1. Decision Tree to initially get a couple of layers of splits.
  2. Build logistic regression based scorecard in each of the splits.
  3. Interpretation is now **within** a split (sub-group) of the data.

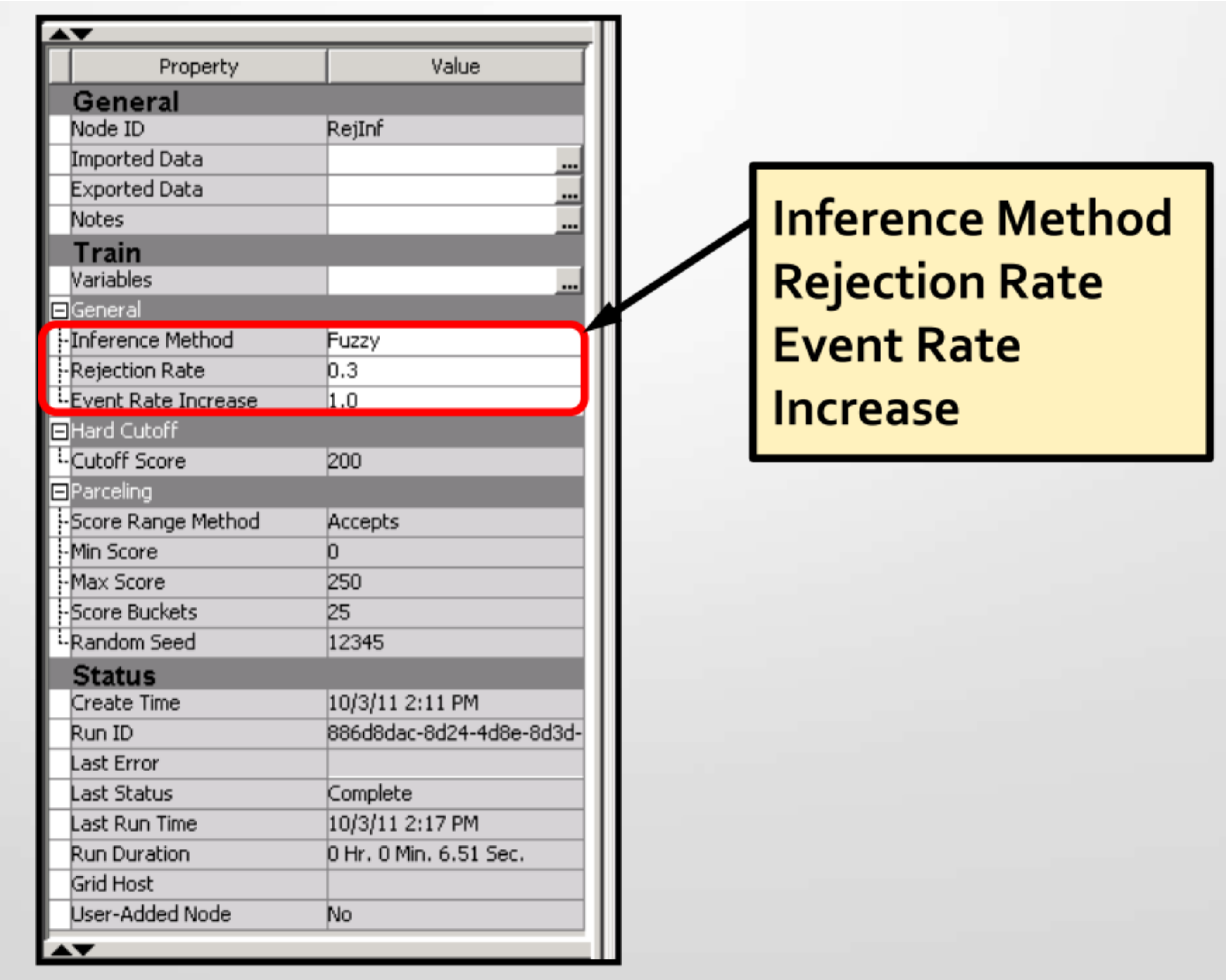
Machine Learning

- Model interpretation is **KEY** in the world of credit scoring.
- Scorecard layer may help drive interpretation of machine learning algorithms, but regulators are still hesitant.

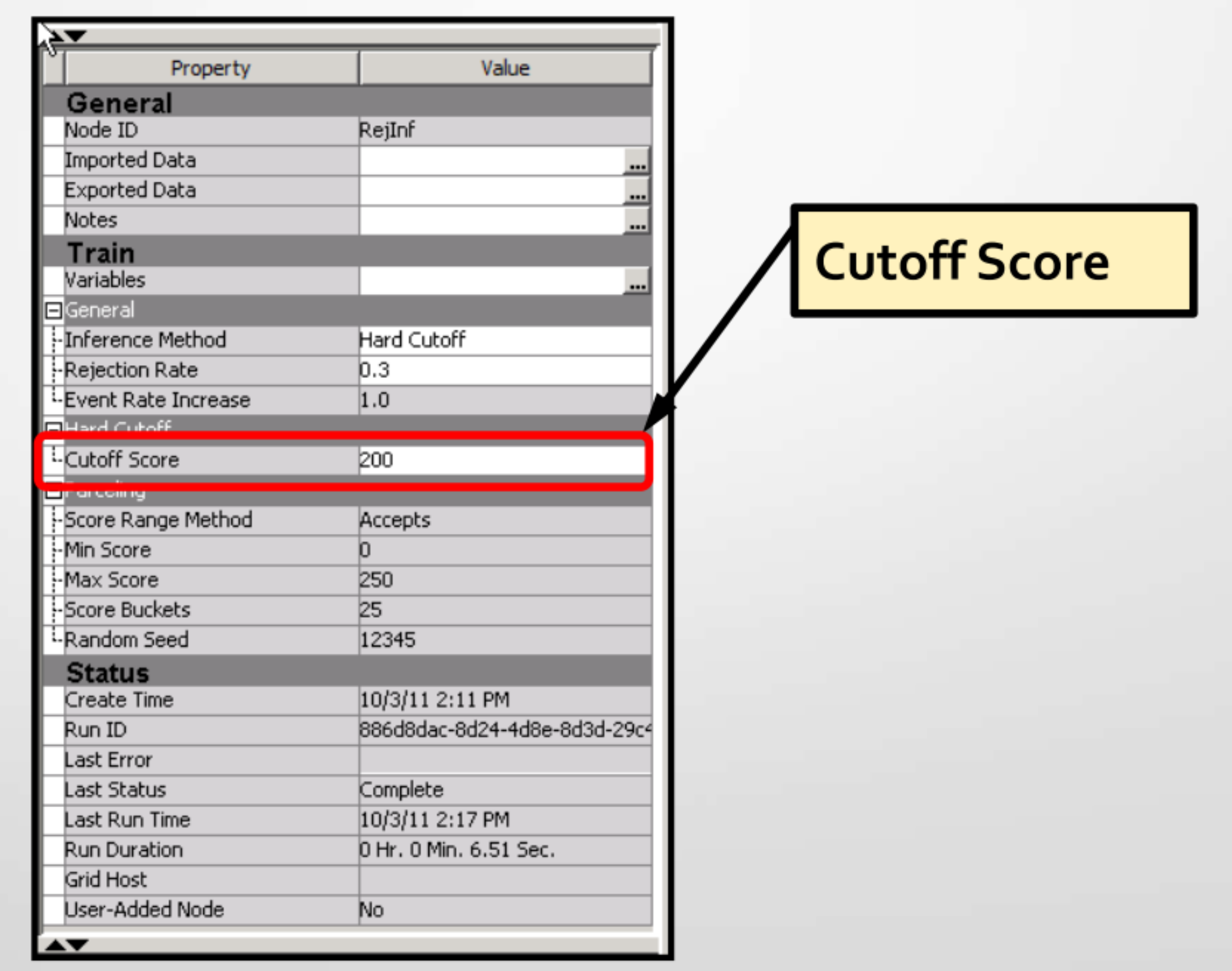
- Great for internal comparison and variable selection.
  - Build a neural network, tree based algorithm, etc. to see if model is statistically different than logistic regression scorecard.
  - Empirical examples have shown WOE based logistic regressions perform very well in comparison to more complicated approaches.

REJECT INFERENCE NODE IN SAS EM

General Options



Hard Cut-off Options



Property	Value
General	
Node ID	RejInf
Imported Data	...
Exported Data	...
Notes	...
Train	
Variables	...
General	
Inference Method	Hard Cutoff
Rejection Rate	0.3
Event Rate Increase	1.0
Hard Cutoff	
Cutoff Score	200
Parceling	
Score Range Method	Accepts
Min Score	0
Max Score	250
Score Buckets	25
Random Seed	12345
Status	
Create Time	10/3/11 2:11 PM
Run ID	886d8dac-8d24-4d8e-8d3d-29c4
Last Error	
Last Status	Complete
Last Run Time	10/3/11 2:17 PM
Run Duration	0 Hr. 0 Min. 6.51 Sec.
Grid Host	
User-Added Node	No

Parceling Options