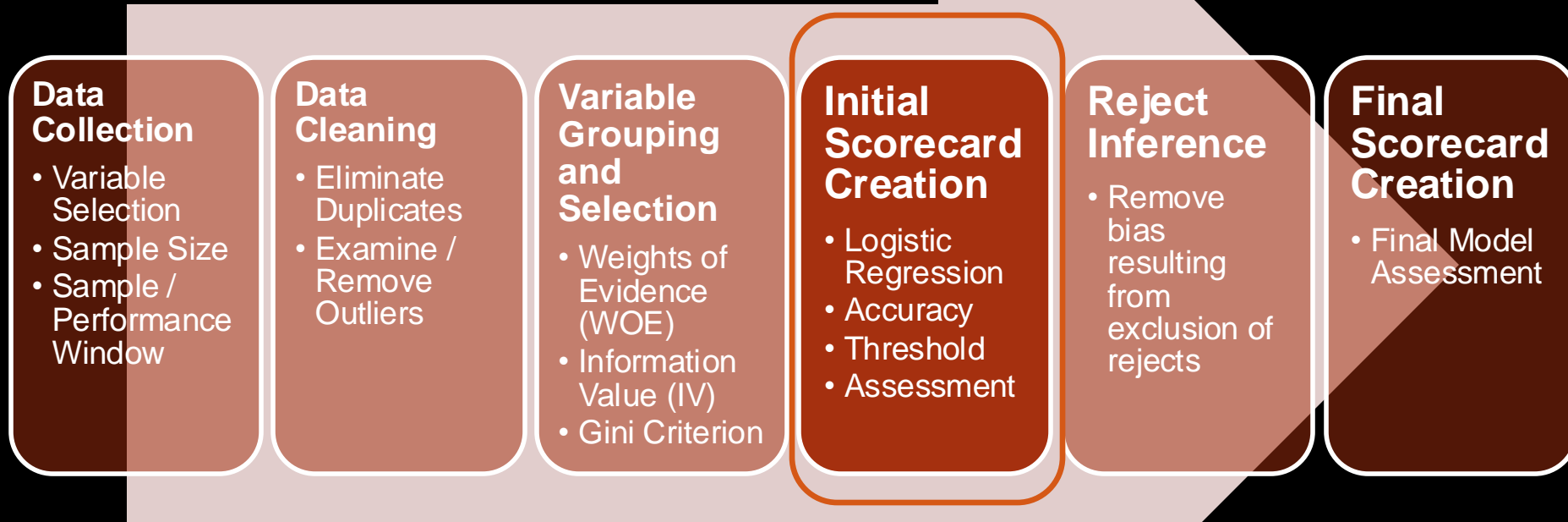


SCORECARD CREATION

Dr. Aric LaBarr

Institute for Advanced Analytics

Process Flow



INITIAL SCORECARD CREATION

Initial Scorecard Model

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

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

- 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.
- 2 Differing Approaches (with similar results):
 1. Weight of Evidence approach – use WOE scores as new variables
 2. Binned approach – use binned variables as new variables

Different Inputs


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

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

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

WOE Approach



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

- Instead of using the original variables for the model, scorecard models have the binned variables as their foundation.
- 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.
- Fewer number of variables (not a ton of categorical variables).

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

Different Inputs

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from optbinning import BinningProcess

colnames = list(train.columns[0:20])

X = train[colnames]
y = train["bad"]

selection_criteria = {"iv": {"min": 0.1, "max": 1}}

bin_proc = BinningProcess(colnames,
                          selection_criteria = selection_criteria,
                          categorical_variables = ["bankruptcy", "purpose", "used_ind"])
```

Different Inputs

```
from optbinning import Scorecard

estimator = LogisticRegression(solver = "lbfgs")

scorecard = Scorecard(binning_process = bin_proc,
                      estimator = estimator,
                      scaling_method = "pdo_odds",
                      scaling_method_params = {"pdo": 20, "scorecard_points": 600, "odds": 50})

scorecard.fit(X, y, sample_weight = train["weight"])
```

Different Inputs

```
from optbinning import Scorecard
estimator = LogisticRegression(solver = "lbfgs")
scorecard = Scorecard(binning_process = bin_proc,
                      estimator = estimator,
                      scaling_method = "pdo_odds",
                      scaling_method_params = {"pdo": 20, "scorecard_points": 600, "odds": 50})
scorecard.fit(X, y, sample_weight = train["weight"])
```

Different Inputs

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

Another Approach



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

- Instead of using the original variables for the model, scorecard models have the binned variables as their foundation.
- Inputs are still treated as **categorical**.
- Need LRT to evaluate p-values.
- Model **coefficients** are desired output for the scorecard.
- Larger number of variables (tons of categorical variables).
- Scorecard creation preprogrammed into a lot of packages.

Bureau Score Bin (R)
716 – 765
Missing
605 – 629
665 – 716
665 – 716
665 – 716
716 – 765

Model Evaluation

- 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 – WOE Approach

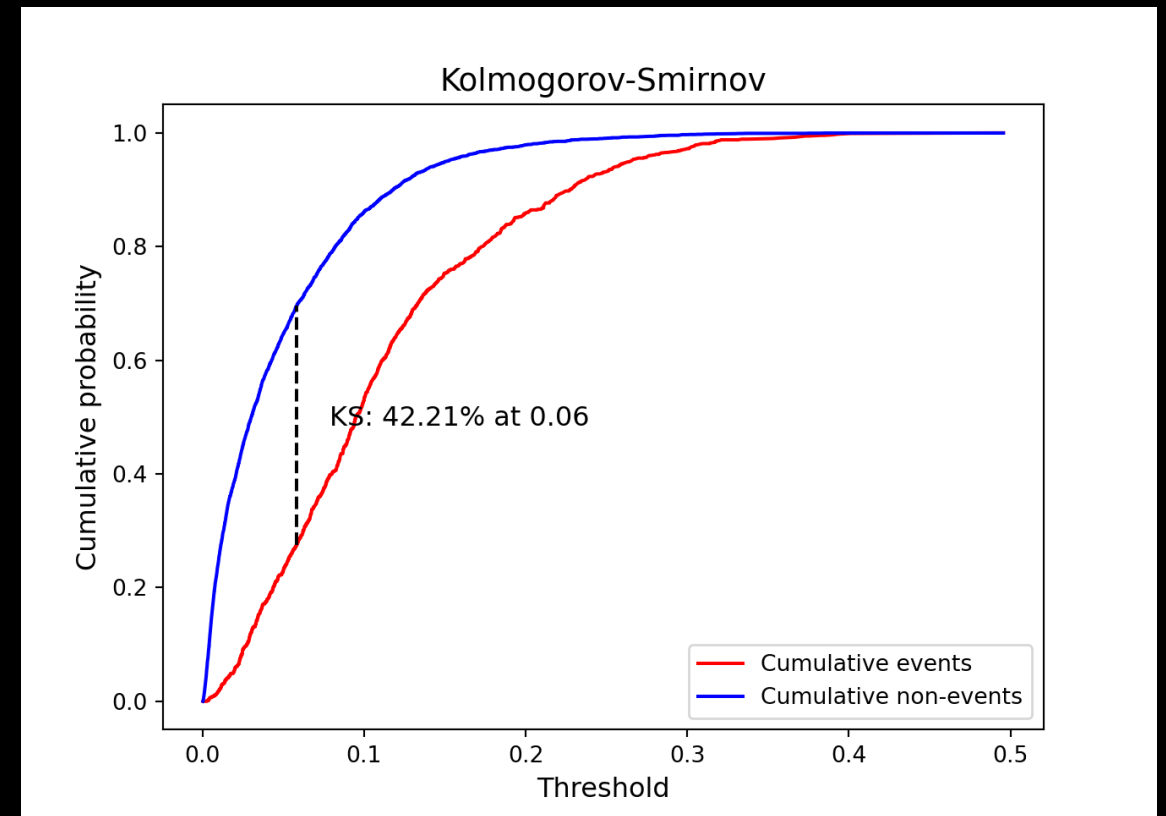
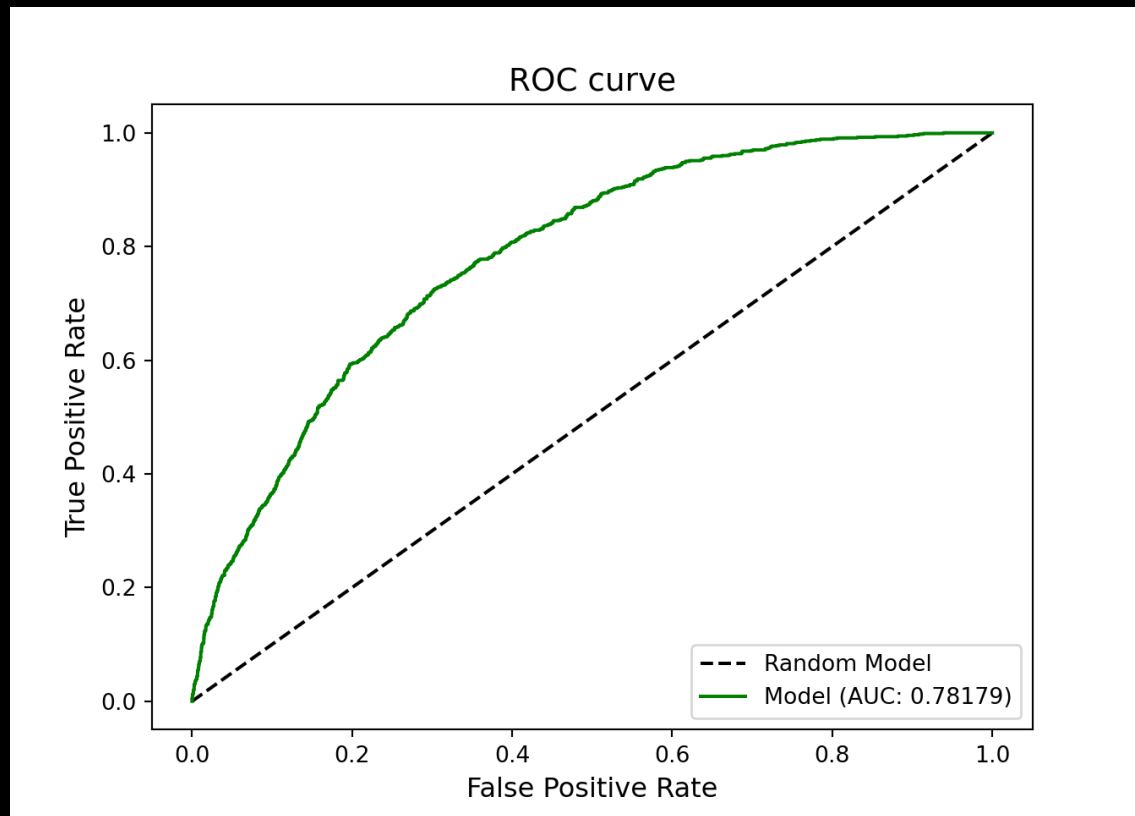
```
from matplotlib import pyplot as plt
from optbinning.scorecard import plot_auc_roc, plot_ks

y_pred = scorecard.predict_proba(X)[: , 1]

plot_auc_roc(y, y_pred)

Plot_ks(y, y_pred)
```


Model Evaluation – WOE Approach



Model Evaluation – Comparison

Metric	WOE Approach	Binned Approach
AUC	0.7818	0.7947
KS	0.4221	0.4345



SCALING THE SCORECARD

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:

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

$$Score + PDO = Offset + Factor \times \log(2 \times odds)$$

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

- 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:

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

Why people still love logistic regression!

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:

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

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 – WOE Approach

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

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

- $WOE_{i,j}$: 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.

Points Allocation – WOE Approach

```
scorecard.table(style = "summary")
```

Variable Bin Points

0	tot_derog	(-inf, 0.50)	54.556666
1	tot_derog	[0.50, 1.50)	51.180992
2	tot_derog	[1.50, 2.50)	50.051248
3	tot_derog	[2.50, 3.50)	50.031948
4	tot_derog	[3.50, 5.50)	46.972047
..
6	tot_income	[2943.13, 4771.00)	51.911410
7	tot_income	[4771.00, 6554.17)	53.539287
8	tot_income	[6554.17, inf)	57.481580
9	tot_income	Special	51.494166
10	tot_income	Missing	51.494166

```
[109 rows x 3 columns]
```

Points Allocation – WOE Approach

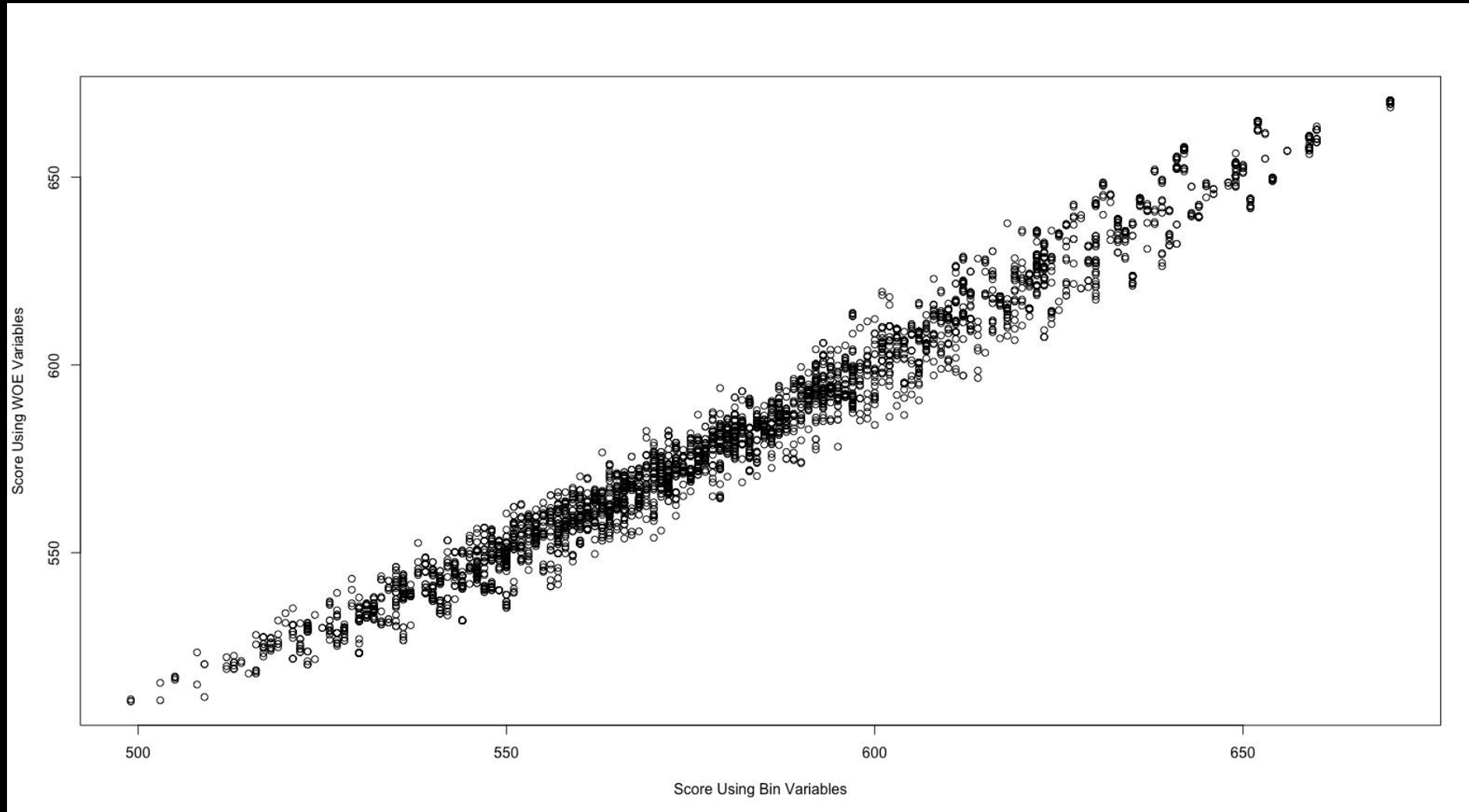
WOE for Bureau Score					
Group	Values	Event Count	Non-event Count	WOE	Points
1	< 607	129	127	-1.367	20.3
2	607 – 640	181	285	-0.898	31.0
3	641 – 653	103	215	-0.616	37.4
4	654 – 667	86	262	-0.238	46.1
...
12	> 786	5	219	2.428	106.9
	MISSING	89	155	-0.797	51.5
Total		900	3,477		

Points Allocation – WOE Approach

```
score = scorecard.score(X)
```

Observation	Target	Variables...	Observation Score
1	0	...	667
2	0	...	627
3	0	...	619
4	1	...	544
5	0	...	619
...

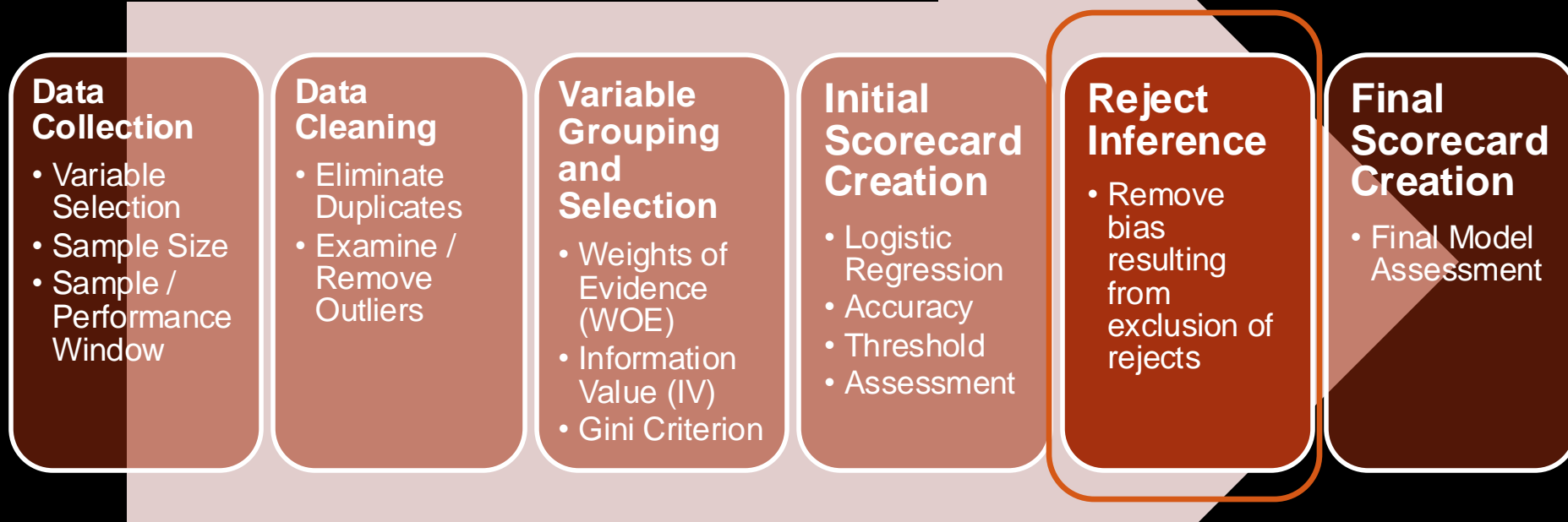
Score Comparison





REJECT INFERENCE

Process Flow



Reject Inference

- **Reject inference** is the process of inferring the status of the rejected applicants based on the accepted applicant 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.

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

Why 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 can generalize better to the entire credit application population.

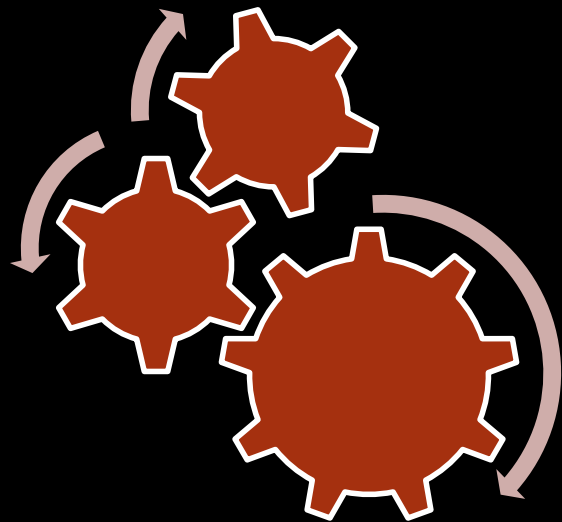
Reject Inference Techniques

- Reject inference is about creating a target variable for a group of observations (reject dataset) that inherently were never given the chance to have a target variable.
- Three common techniques to create target variable with reject inference:
 1. Simple / Hard Cutoff Augmentation
 2. Parceling Augmentation
 3. Fuzzy Augmentation

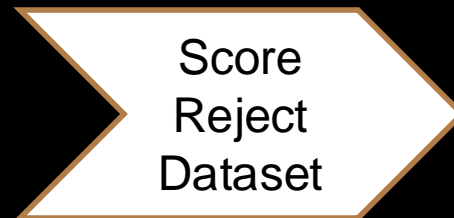
Reject Inference Techniques

- Reject inference is about creating a target variable for a group of observations (reject dataset) that inherently were never given the chance to have a target variable.
- Three common techniques to create target variable with reject inference:
 1. Simple / Hard Cutoff Augmentation
 2. Parceling Augmentation
 3. Fuzzy Augmentation

Simple / Hard Cutoff Augmentation

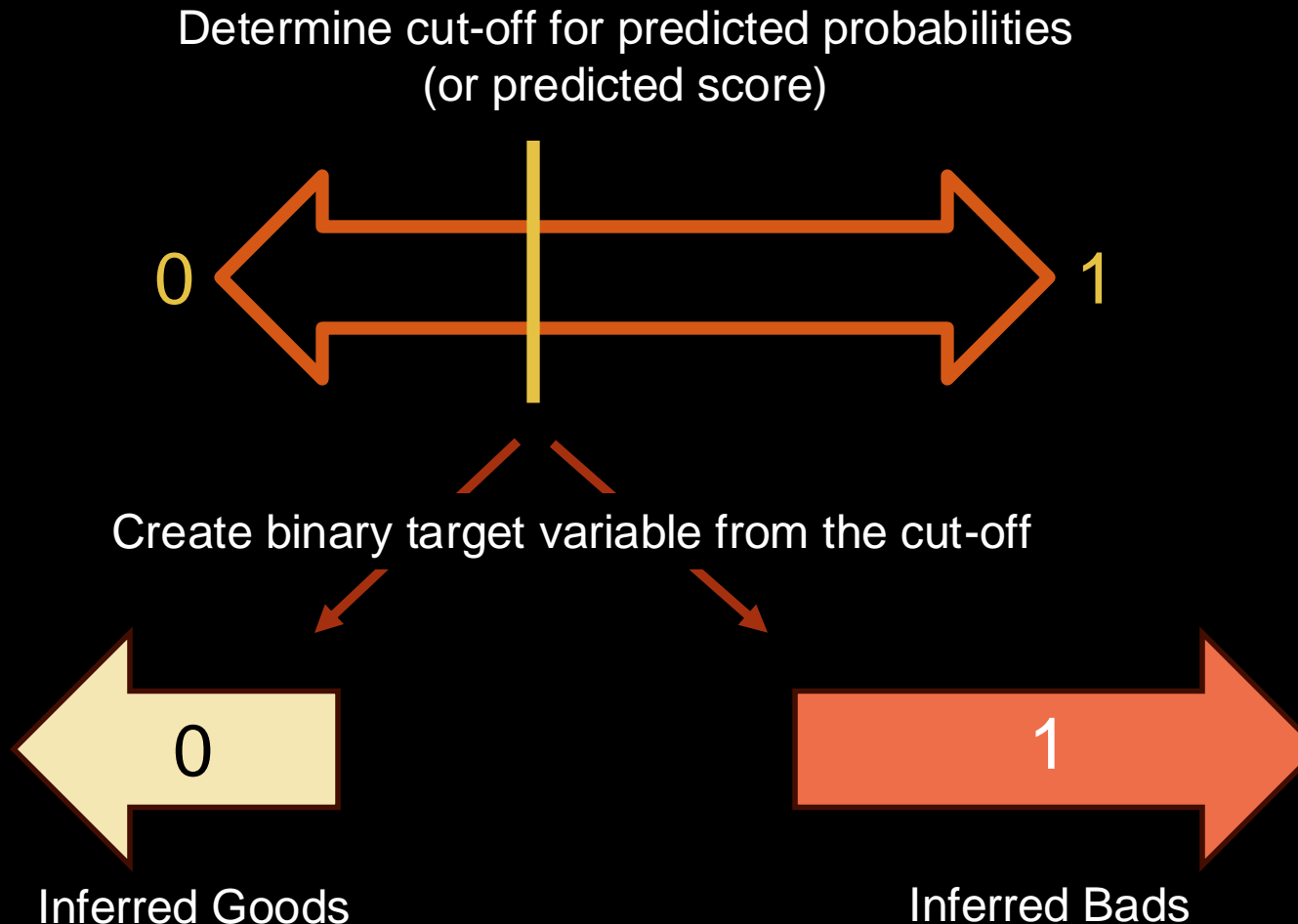


Original credit scoring
model built on accepted
customers



Have predicted probability of **default**
for all of the rejected customers
(can also use predicted score)

Simple / Hard Cutoff Augmentation



Hard Cutoff Augmentation

```
X_r = rejects
rejects["y_pred"] = scorecard.predict_proba(X_r)[:, 1]

rejects["bad"] = (rejects["y_pred"] > 0.06).astype(int)
rejects["weight"] = rejects["bad"].apply(lambda x: 4.75 if x == 1 else 1)

rejects = rejects.drop("y_pred", axis=1)

comb_hard = pd.concat([accepts, rejects.head(1876)], ignore_index=True)
```

Hard Cutoff Augmentation

```
X_r = rejects
rejects["y_pred"] = scorecard.predict_proba(X_r)[:, 1]

rejects["bad"] = (rejects["y_pred"] > 0.06).astype(int)
rejects["weight"] = rejects["bad"].apply(lambda x: 4.75 if x == 1 else 1)

rejects = rejects.drop("y_pred", axis=1)

comb_hard = pd.concat([accepts, rejects.head(1876)], ignore_index=True)
```

Hard Cutoff Augmentation

```
X_r = rejects
rejects["y_pred"] = scorecard.predict_proba(X_r)[:, 1]

rejects["bad"] = (rejects["y_pred"] > 0.06).astype(int)
rejects["weight"] = rejects["bad"].apply(lambda x: 4.75 if x == 1 else 1)

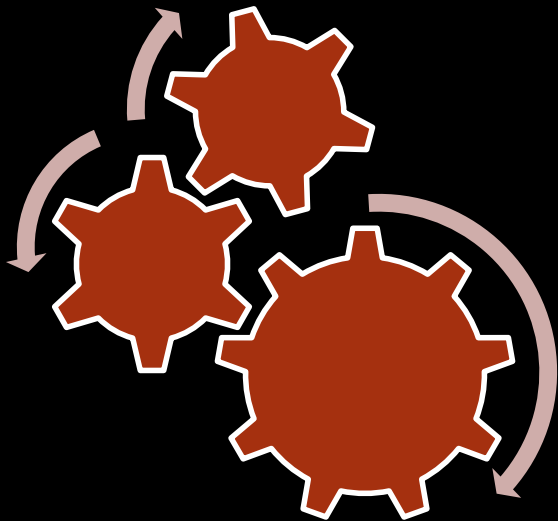
rejects = rejects.drop("y_pred", axis=1)

comb_hard = pd.concat([accepts, rejects.head(1876)], ignore_index=True)
```

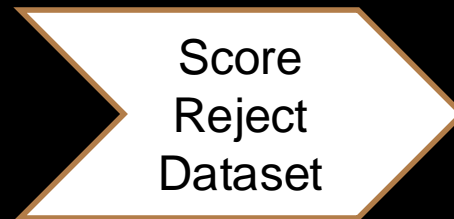
Reject Inference Techniques

- Reject inference is about creating a target variable for a group of observations (reject dataset) that inherently were never given the chance to have a target variable.
- Three common techniques to create target variable with reject inference:
 1. Simple / Hard Cutoff Augmentation
 2. Parceling Augmentation
 3. Fuzzy Augmentation

Parceling Augmentation

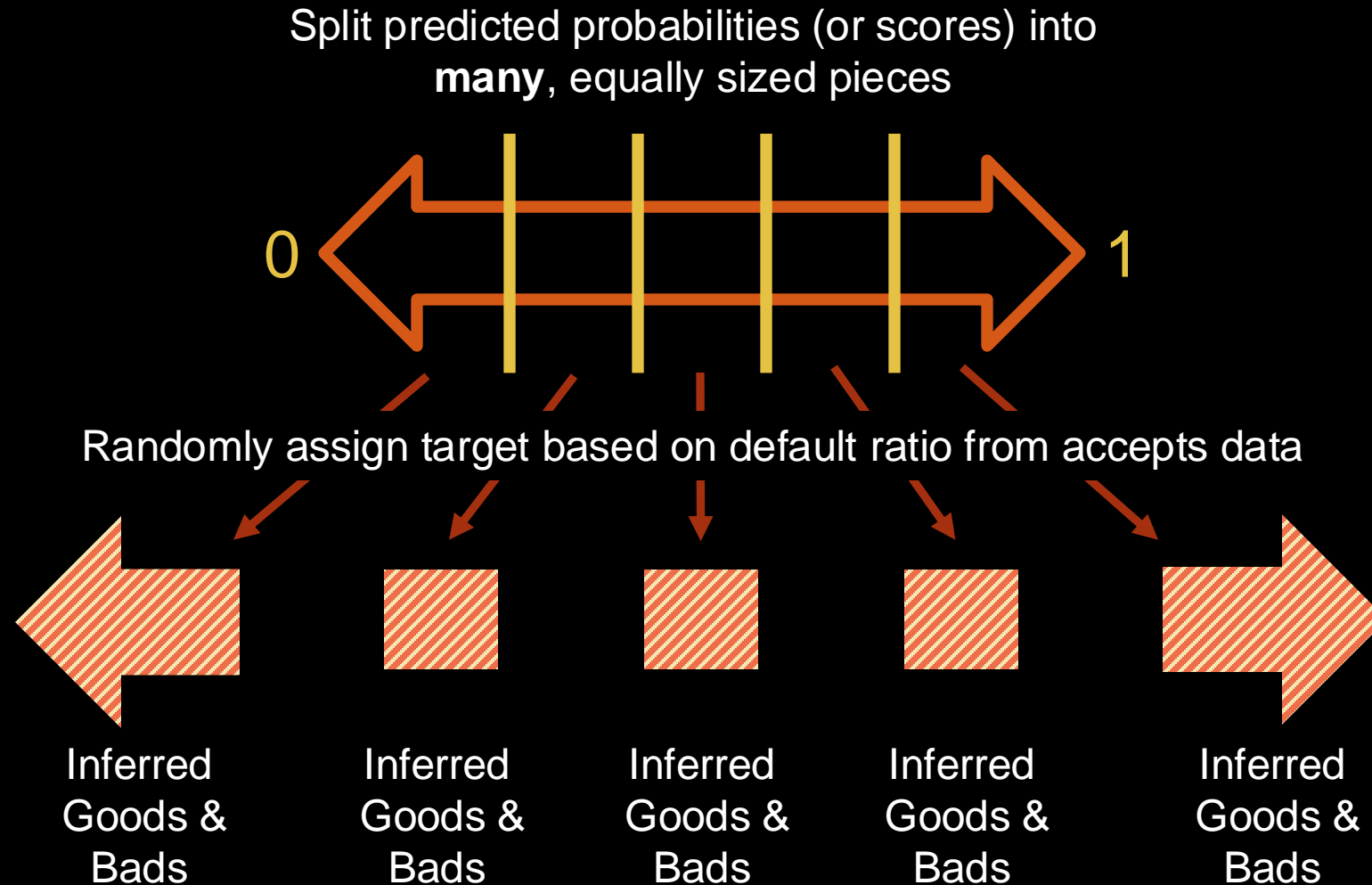


Original credit scoring
model built on accepted
customers



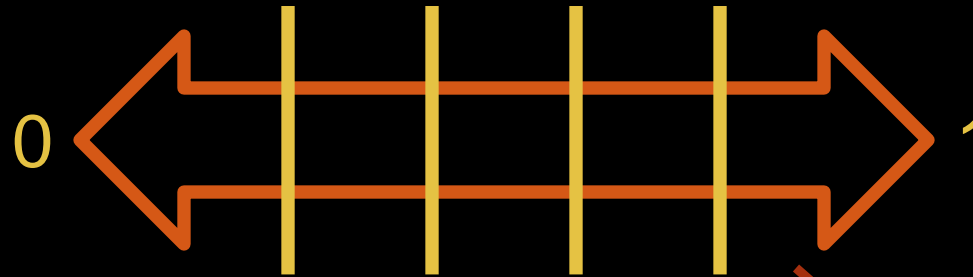
Have predicted probability of **default**
for all of the rejected customers
(can also use predicted score)

Parceling Augmentation



Parceling Augmentation

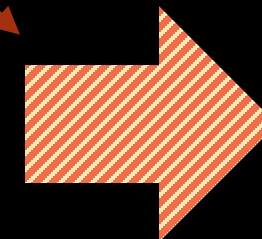
Split predicted probabilities (or scores) into
many, equally sized pieces



Randomly assign target based on default ratio from accepts data

In accepts data we see
25% default in this range
of probabilities.

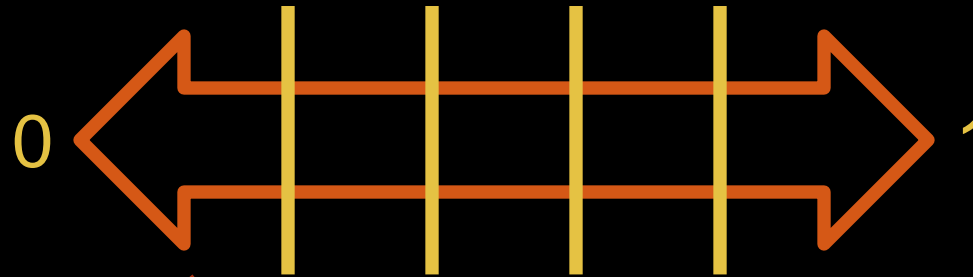
Randomly assign 25% of
observations in this group
to default



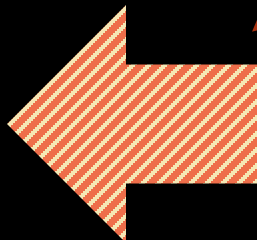
Inferred
Goods &
Bads

Parceling Augmentation

Split predicted probabilities (or scores) into
many, equally sized pieces



Randomly assign target based on default ratio from accepts data

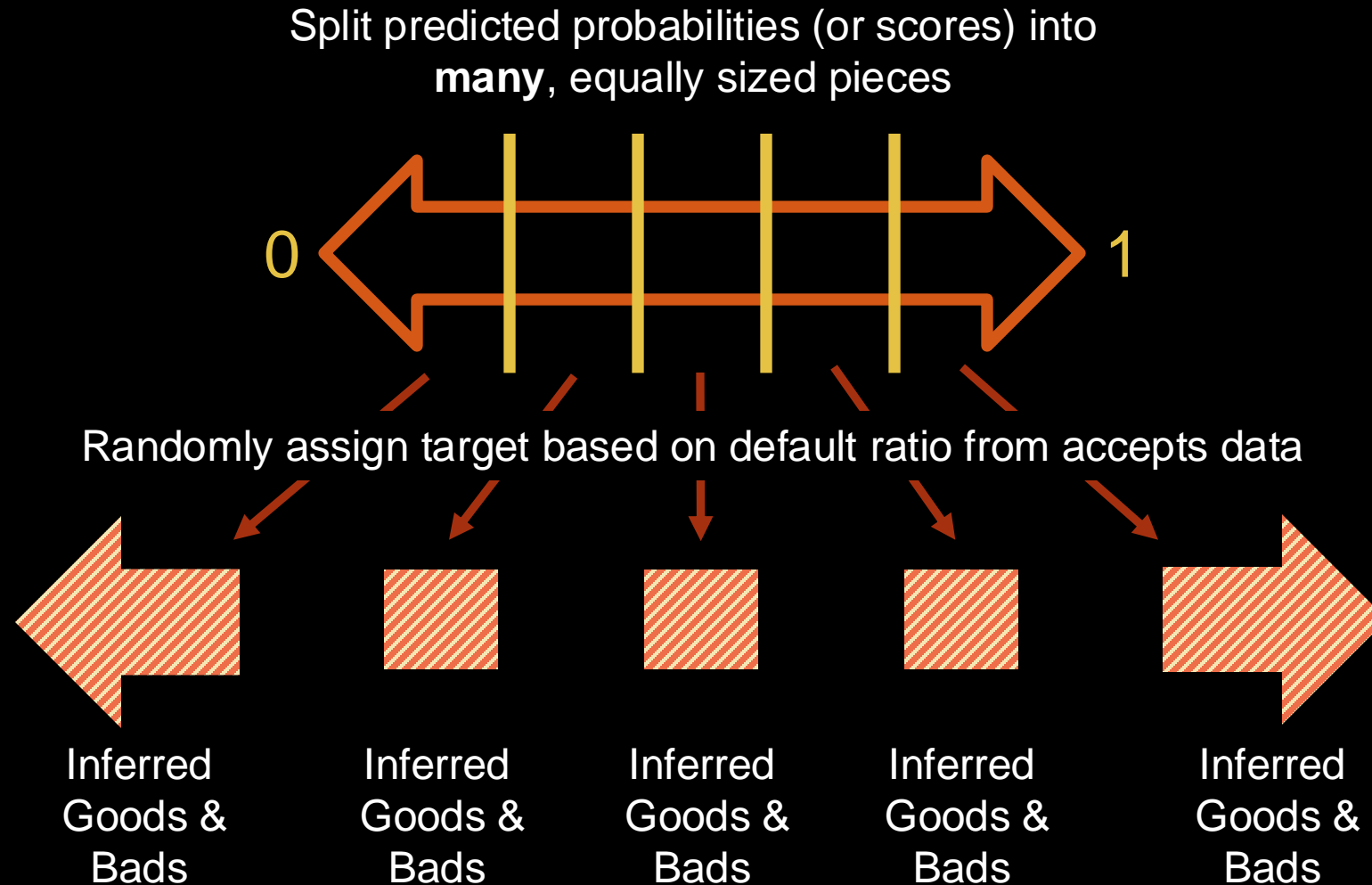


Inferred
Goods &
Bads

In accepts data we see
1% default in this range
of probabilities.

Randomly assign 1% of
observations in this group
to default

Parceling Augmentation



Parceling Augmentation – Example

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

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

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	?


$$0.455 \times 190 \approx 86$$

Randomly assign!

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	104


$$190 - 86 = 104$$

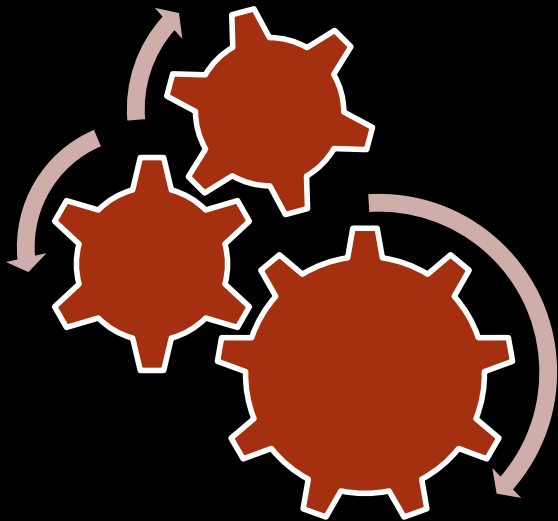
Parceling Augmentation – Example

	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	104
665 – 675	450	700	39.1%	60.9%	250	98	152
...

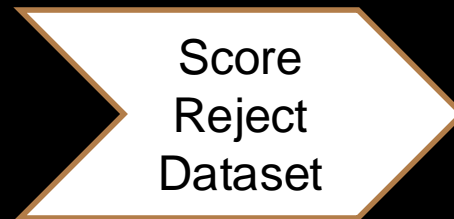
Reject Inference Techniques

- Reject inference is about creating a target variable for a group of observations (reject dataset) that inherently were never given the chance to have a target variable.
- Three common techniques to create target variable with reject inference:
 1. Simple / Hard Cutoff Augmentation
 2. Parceling Augmentation
 3. Fuzzy Augmentation

Fuzzy Augmentation



Original credit scoring
model built on accepted
customers



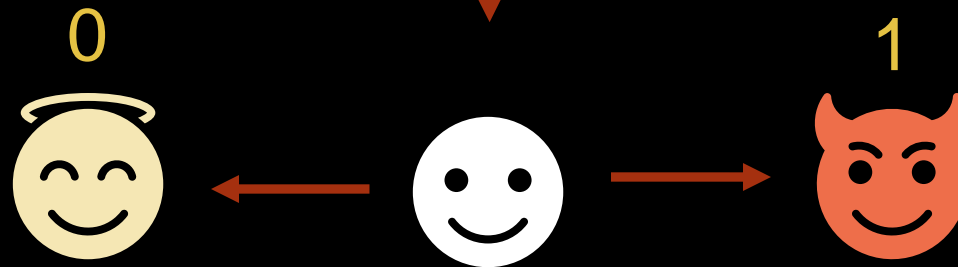
Have predicted probability of **default**
for all of the rejected customers
(can also use predicted score)

Fuzzy Augmentation

From this scored dataset we will take each observation and duplicate it



For example, predicted probability of default of 0.78



Weight of 0.22 multiplied
by old weight

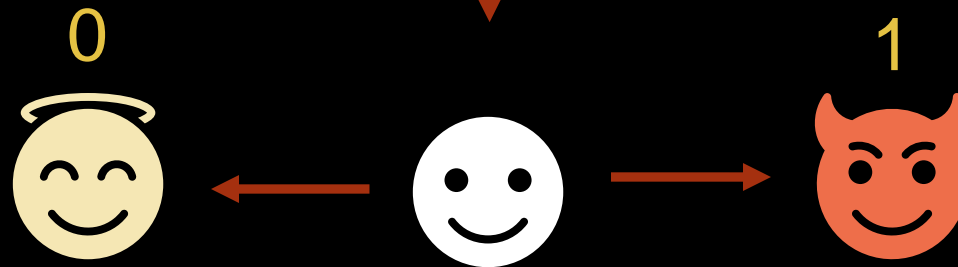
Weight of 0.78 multiplied
by old weight

Fuzzy Augmentation

From this scored dataset we will take each observation and duplicate it



For example, predicted probability of default of 0.03



Weight of 0.97 multiplied
by old weight

Weight of 0.03 multiplied
by old weight

Fuzzy Augmentation

Reject dataset



0



1



⋮

⋮

⋮

⋮



Reject Inference Techniques

- Three common techniques for reject inference:
 1. Simple / Hard Cutoff Augmentation
 2. Parceling Augmentation
 3. Fuzzy Augmentation
- Once you have a target variable, you add the rejected dataset back into the accepted dataset to perform an overall analysis.
- Still take same process as before where you split into training and testing, etc.
- Careful about the weights! Try to keep accepted / rejected balance the same as before (don't want to overweight either group compared to population).

Reject Inference Techniques

- Three common techniques for reject inference:
 1. Simple / Hard Cutoff Augmentation
 2. Parceling Augmentation
 3. Fuzzy Augmentation
- 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.

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.

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.

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.

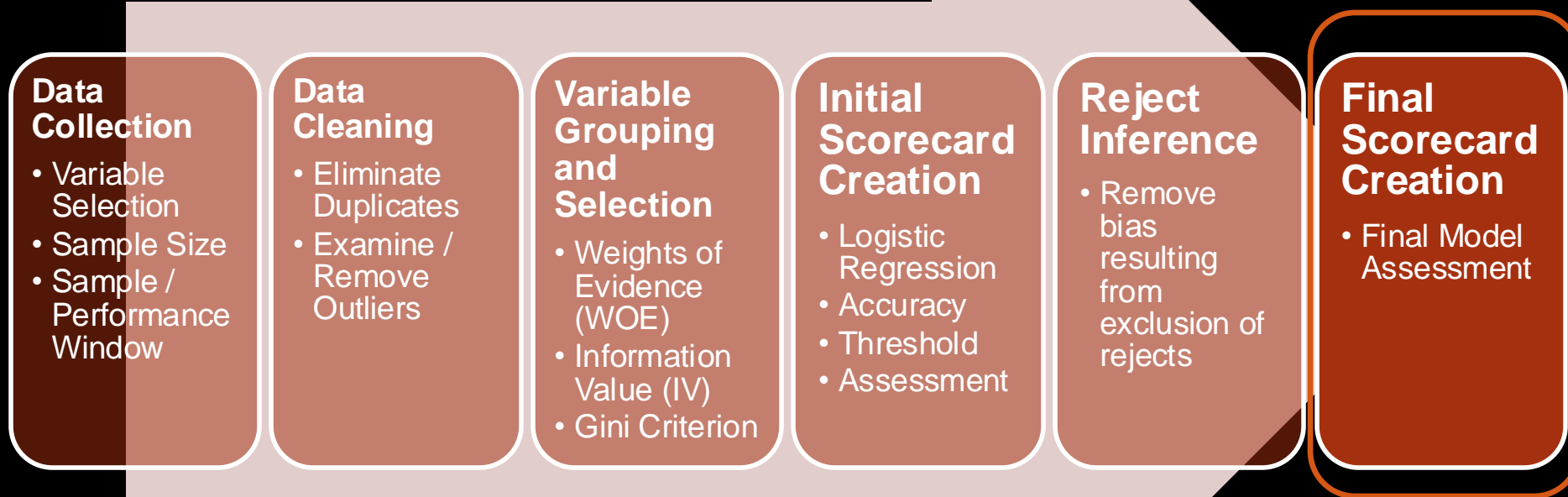
Other Reject Inference Techniques

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...



FINAL SCORECARD CREATION

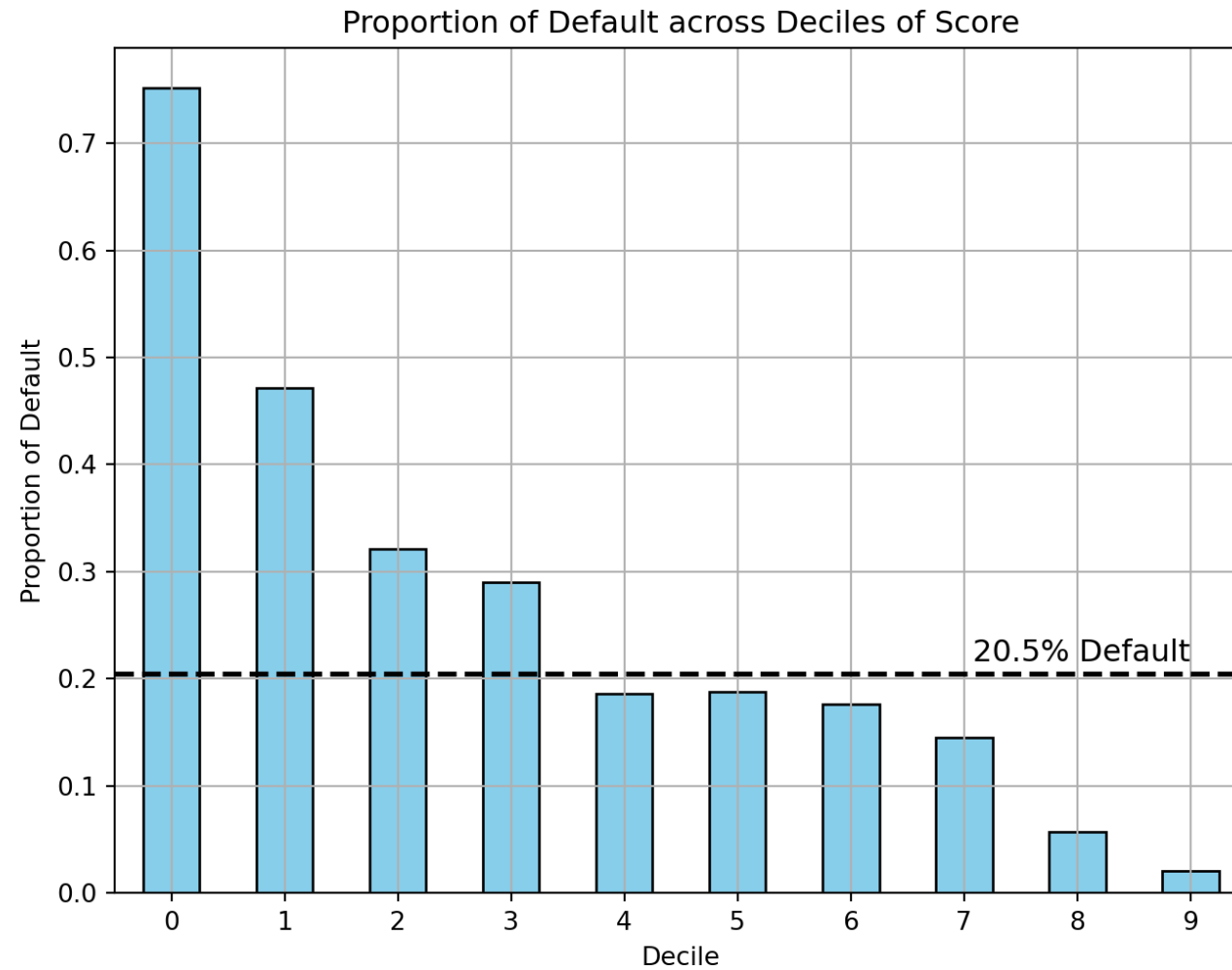
Process Flow



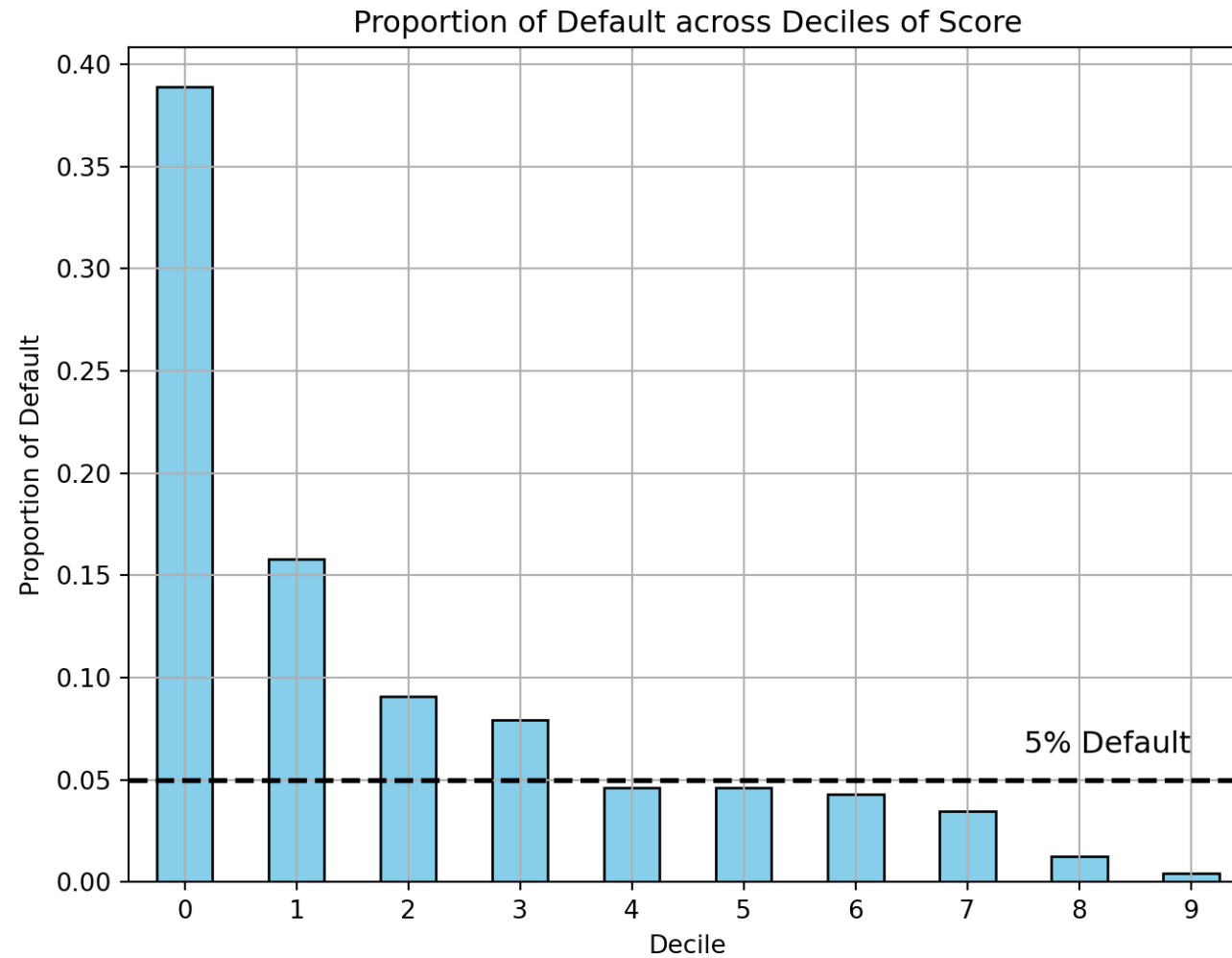
Final Scorecard Creation

- 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.

Default Decile Plot



Default Decile Plot



Defining Cut-off

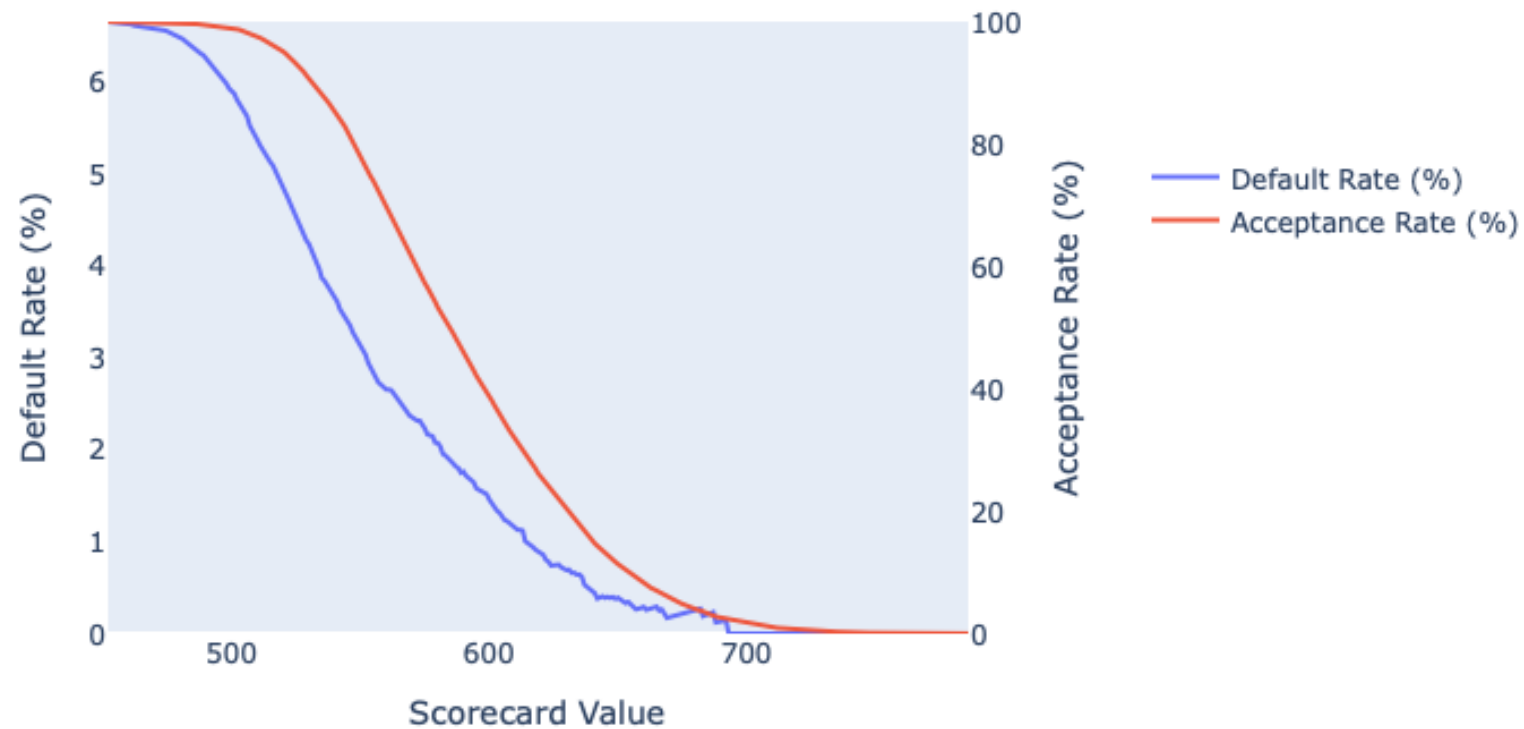
- A new scorecard should be better than the last in terms of one of the following:
 - Lower bad rate for the same approval rate.
 - Higher approval rate while holding the bad rate constant.

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

Default Rate by Acceptance Across Score



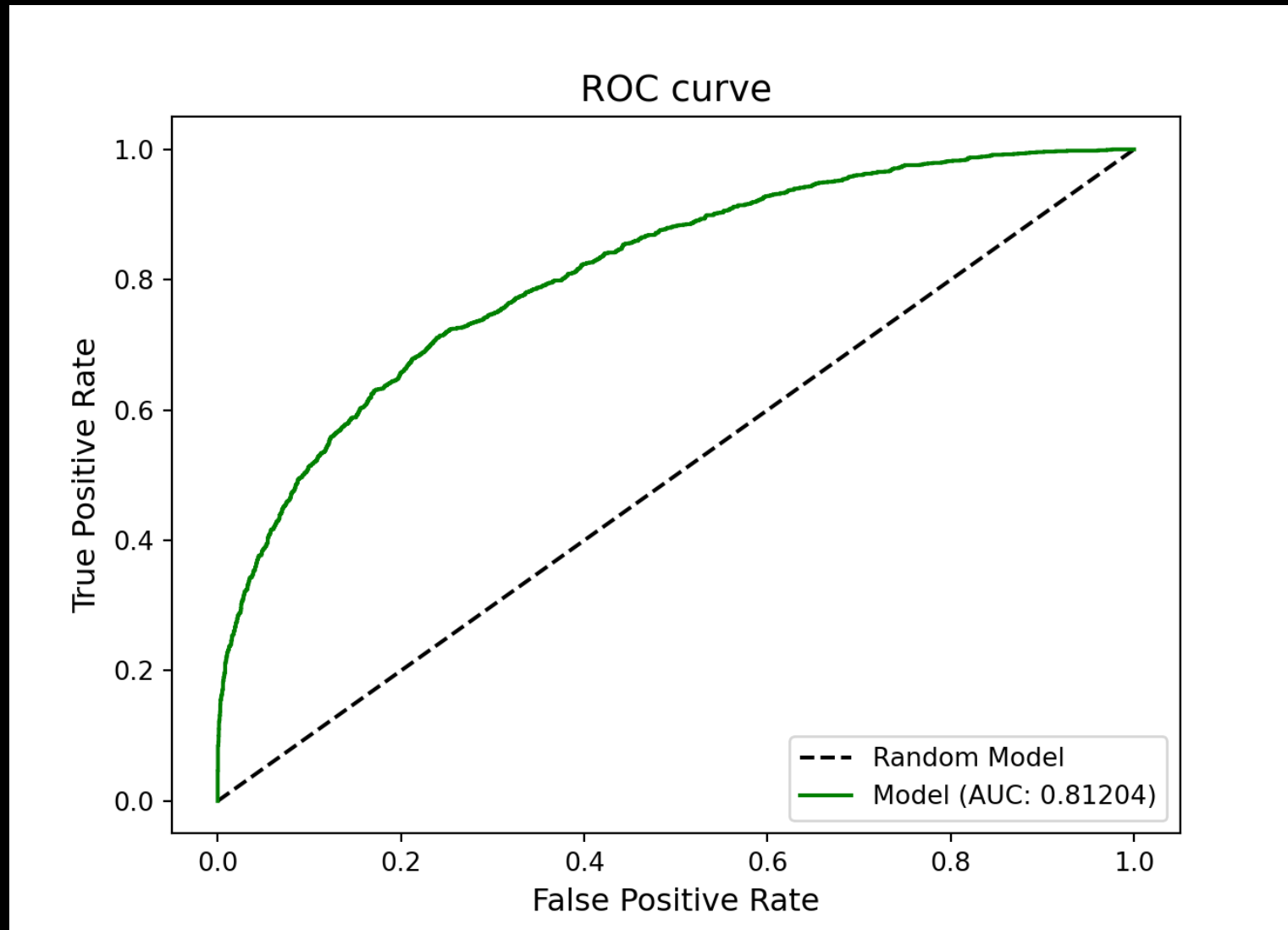
Defining Cut-off



Defining Cut-off

- Setting Multiple Cut-offs Example:
 - Anyone who scores above 606 points is accepted automatically because that maximizes profit.
 - Anyone who scores below 517 is declined because that is our current default rate of 5%.
 - Any scores in between 517 and 606 are referred to manual adjudication.

Final Scorecard – Example





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.

