slides originally by
Dr. Richard Burns,
modified by
Dr. Stephanie Schwartz

CLASS IMBALANCE

CSCI 452: Data Mining

Class Imbalance Problem

- Datasets with imbalanced class distributions are very common in “real data”
  - *Example*: in CC fraud domain, there are many more legitimate transactions than fraudulent transactions
  - Disproportionate number of instances
- Correct classification of the *rare case* is more important than a correct classification of the majority class
- Imbalance class distribution sometimes problematic for classification algorithms

Evaluation Metric

- How should classifier be quantitatively evaluated?
- If Accuracy is used:
  - *Example:* in CC fraud domain, there are many more legitimate transactions than fraudulent transactions
  - Presume only 1% of transactions are fraudulent
- Then:
  - Classifier that predicts every transaction as non-fraudulent would have 99% accuracy!
  - Seems great, but it's not…

Other Issues

- Measures that guide the learning algorithm (e.g. information gain for decision trees) may need to be modified to focus on the rare case
- (*Accuracy* measure treats every class as equally important.)

Alternative Metrics

- Moving away from accuracy for imbalanced classes
  - *Rare class* is more interesting (more important) than *majority class*
- Notation (for binary classification):
  - + (positive class), rare class
  - - (negative class), majority class

Confusion Matrix: Counts

		Predicted Class	
		+	–
Actual Class	+	f_{++} (True Positive)	f_{+-} (False Negative)
	–	f_{-+} (False Positive)	f_{--} (True Negative)

- True Positive (TP): number of positive examples correctly predicted by model
- False Negative (FN): number of positive examples wrongly predicted as negative
- False Positive (FP): number of negative examples wrongly predicted as positive
- True Negative (TN): number of negative examples correctly predicted

Confusion Matrix: Percentages

		Predicted Class	
		+	–
Actual Class	+	f_{++} (True Positive)	f_{+-} (False Negative)
	–	f_{-+} (False Positive)	f_{--} (True Negative)

- True Positive Rate (TPR): fraction of positive examples correctly predicted by model
  - Also referred to as sensitivity

TPR = TP/(TP+FN)

- False Negative Rate (FNR): fraction of positive examples wrongly predicted as negative

FNR = FN/(TP+FN)

- False Positive Rate (FPR): fraction of negative examples wrongly predicted as positive

FPR = FP/(TN+FP)

- True Negative Rate (TNR): fraction of negative examples correctly predicted
  - Also referred to as specificity

TNR = TN/(TN+FP)

Precision and Recall

- Widely used metrics when successful detection of one class is considered more significant than detection of the other classes

$$\text{Precision}, p = \frac{TP}{TP + FP}$$

$$\text{Recall}, r = \frac{TP}{TP + FN}$$

Precision

- Precision: fraction of records that are positive in the set of records that classifier predicted as positive
- *Interpretation:* higher the precision, the lower the number of false positive errors

Recall

- <u>Recall:</u> fraction of positive records that are correctly predicted by classifier
- *Interpretation:* higher the recall, the fewer number of positive records misclassified as negative class

Baseline Models

- Often naïve models
- *Example #1:* classify every instance as positive (the *rare class*)
  - What is accuracy? Precision? Recall?
  - Precision = poor; Recall = 100%
- Baseline models often maximize one metric (precision, recall) but not the other.
- Key challenge: building a model that performs well with both metrics

F_1 Measure

- F-measure: combines precision and recall into a single metric, using the *harmonic mean*
  - *Harmonic Mean* of two numbers tends to be closer to the smaller of two numbers…
  - …so the only way F_1 is high is for both precision and recall to be high.

$$F_1 = \frac{2rp}{r+p} = \frac{2 \times TP}{2 \times TP + FP + FN}$$

So, we want a high TPR and low FPR.

ROC Curve

- Receiver Operating Characteristic (ROC) Curve:
  - Graphical approach for displaying the tradeoff between *true positive rate* and *false positive rate* of a classifier
- Useful for comparing the relative performance among different classifiers
- Drawn in 2 dimensions
  - X-axis: false positive rate
  - Y-axis: true positive rate

FPR = FP/(TN+FP)
"fraction of negative examples wrongly predicted as positive"

TPR = TP/(TP+FN)
"fraction of positive examples correctly predicted by model"

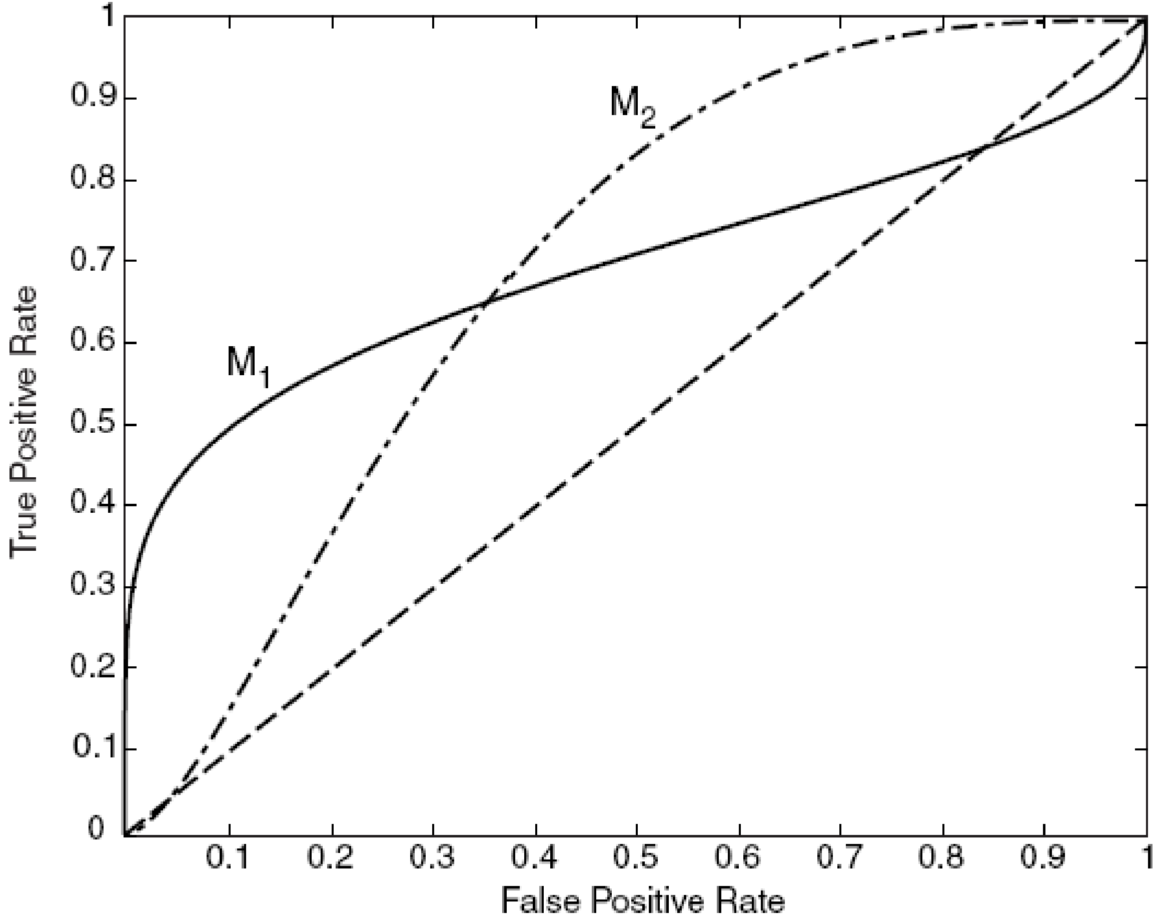


Figure 5.41. ROC curves for two different classifiers.

- ROC Curve only needs one classifier drawn
- But multiple classifiers can be included
- Think of M_1 as C_1
  - Classifier #1 instead of Model #1
  - (e.g. your decision tree w/o pruning)

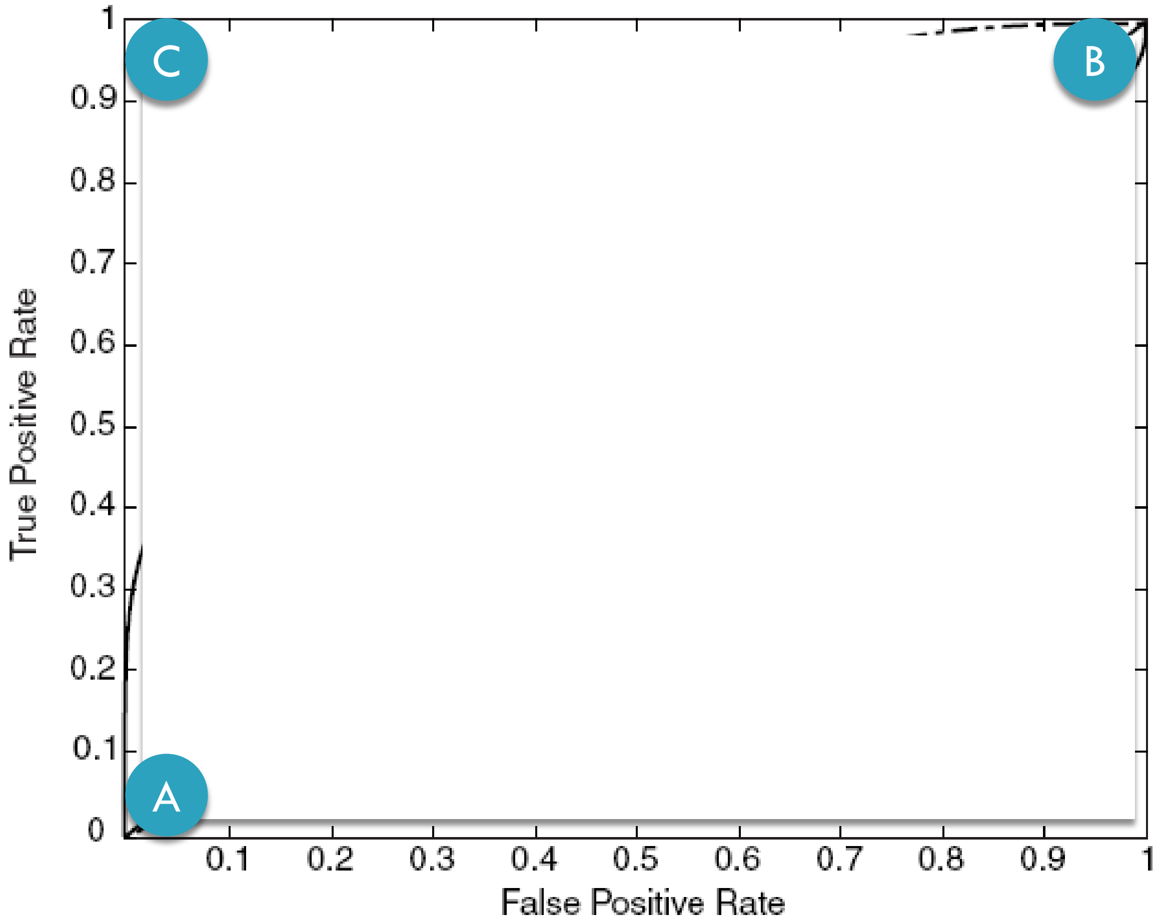


Figure 5.41. ROC curves for two different classifiers.

If an ROC curve passed through these data points, interpretations would be:

- A *(TPR=0, FPR=0)*: model predicts every instances as negative
- B *(TPR=1, FPR=1)*: model predicts every instance as positive
- C *(TPR=1, FPR=0)*: ideal model, no errors

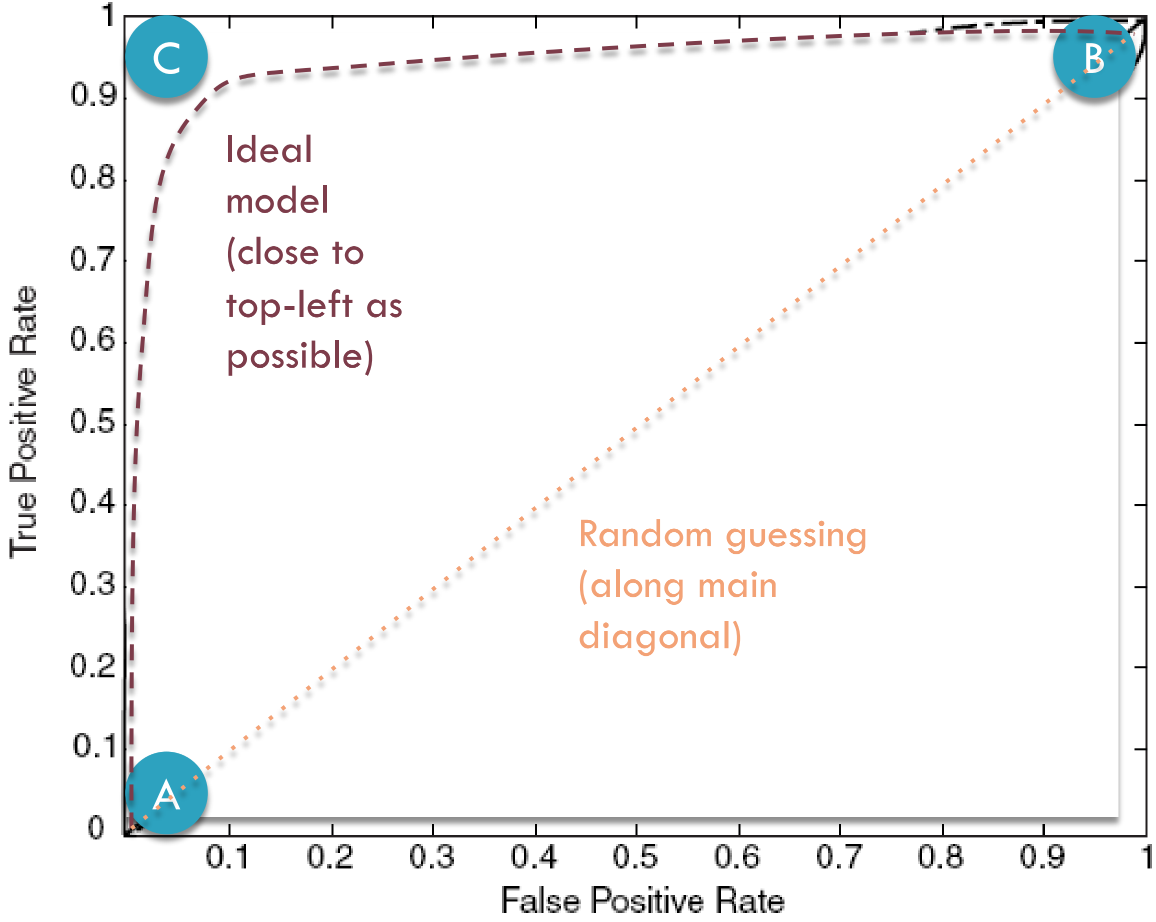


Figure 5.41. ROC curves for two different classifiers.

If an ROC curve passed through these data points, interpretations would be:

- A *(TPR=0, FPR=0)*: model predicts every instances as negative
- B *(TPR=1, FPR=1)*: model predicts every instance as positive
- C *(TPR=1, FPR=0)*: ideal model, no errors

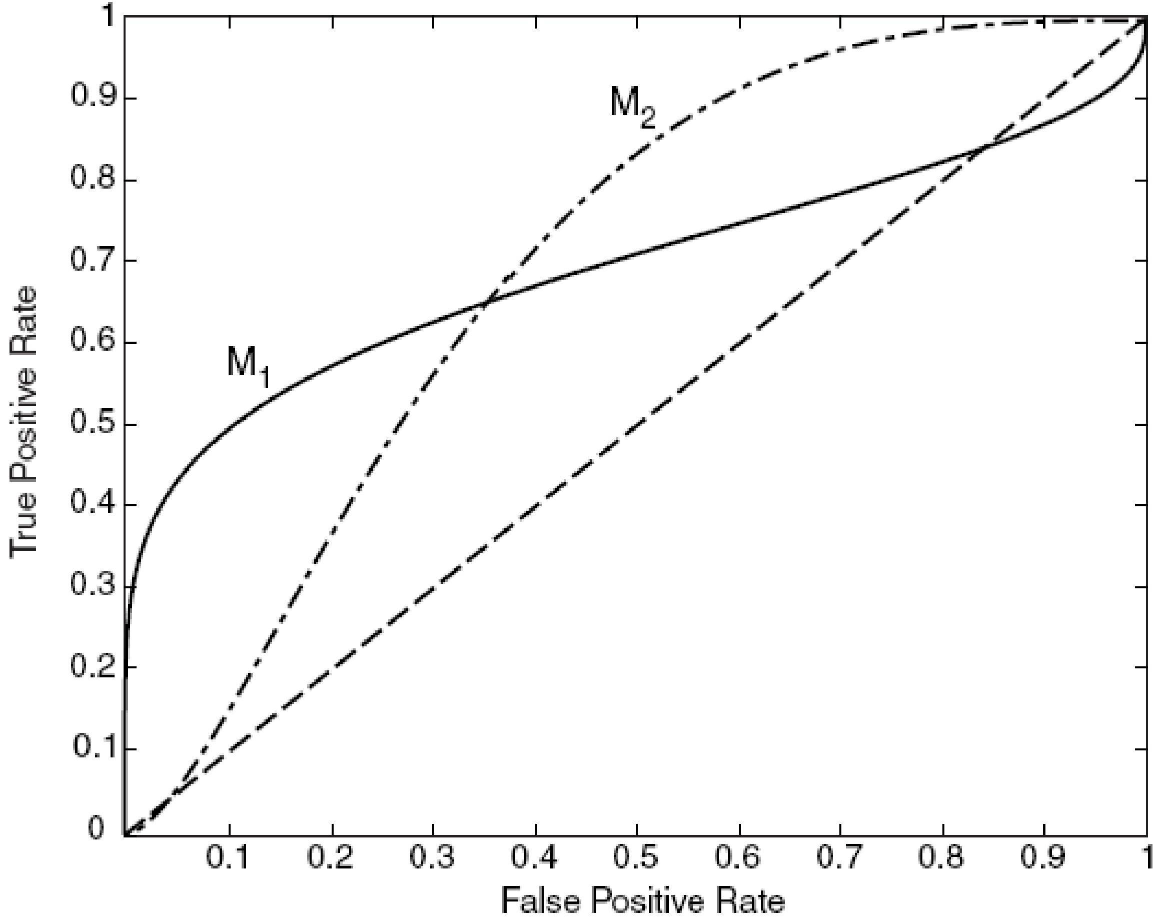


Figure 5.41. ROC curves for two different classifiers.

- ROC Curves are useful to compare performance of two classifiers
- This graph:
  - M_1 is better when *FPR* < *0.36*
  - else, M_2 is superior

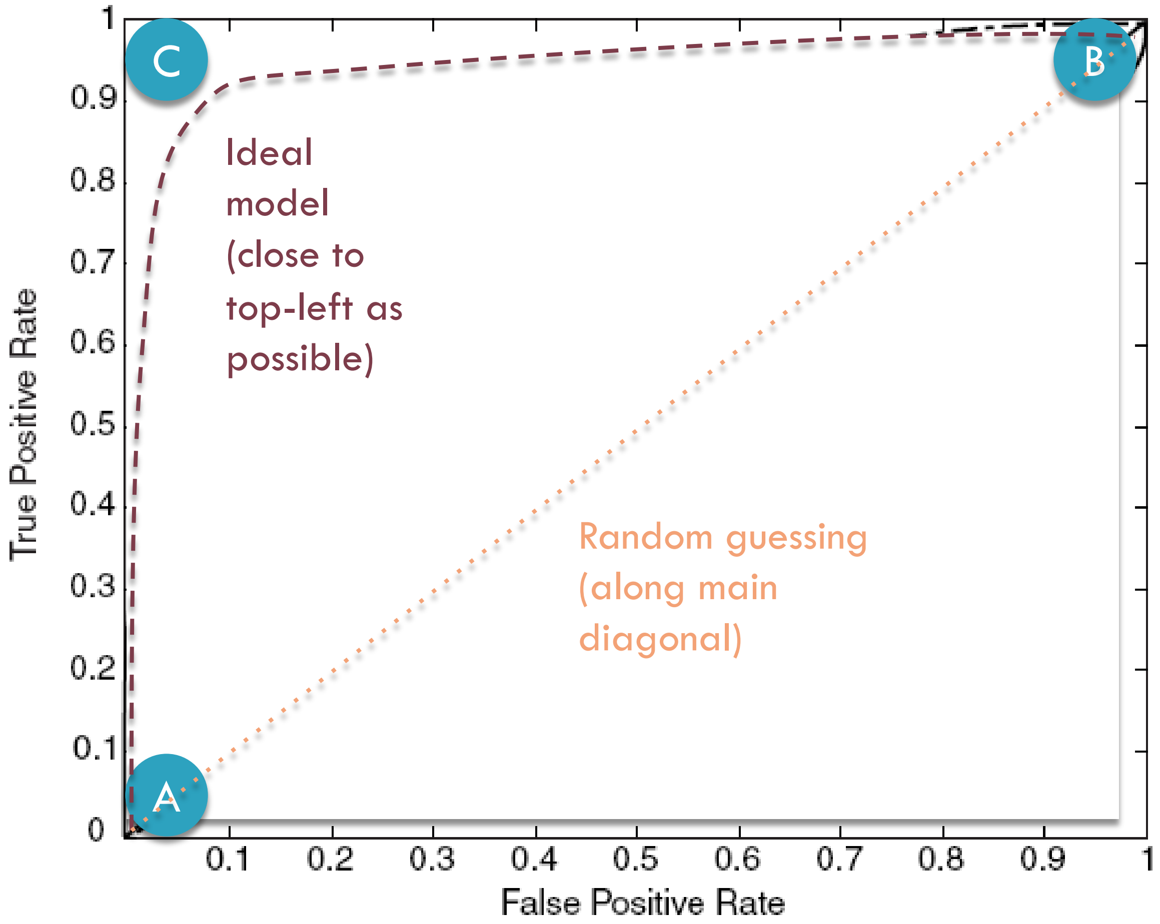


Figure 5.41. ROC curves for two different classifiers.

<u>Area Under ROC Curve (AUC) Metric:</u>

- Ideal model
  - *AUC = 1*
- Random guessing
  - *AUC = 0.5*

Generating an ROC Curve

- *What's necessary?*
  - Classifier needs to produce <u>continuously-valued output</u> that can be used *to rank* its predictions
    - From instance that is most likely positive to instance that is least likely positive
- *Classifiers that do this:*
  - Naïve Bayes
  - Support Vector Machines
  - Classification Trees in R that output probabilities

Test Instance #	Model's Probability Output of Instance being +
1	0.25
2	0.85
3	0.93
4	0.43
5	0.85
6	0.53

Generating an ROC Curve

Actual Class	-	+	+	-	+	+	-	-	-	+
Model Output	0.76	0.93	0.95	0.85	0.85	0.25	0.85	0.87	0.43	0.53

1. Sort the test records in increasing order of their output values

Actual Class	+	-	+	-	-	-	+	-	+	+
Model Output	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95

Generating an ROC Curve

2. Select the lowest ranked test record. Assign the selected record and those ranked above it to the positive class. Assign TP and FP for the current record.

Actual Class	+	-	+	-	-	-	+	-	+	+
Model Output	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95
Assign	+	+	+	+	+	+	+	+	+	+
TP	5									
FP	5									

(Equivalent to classifying all records as +.)

Generating an ROC Curve

3. Select the next test record. Classify it and those above it as positive. Classify those below it as negative. Assign TP and FP counts for the current record.

Actual Class	+	-	+	-	-	-	+	-	+	+
Model Output	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95
Assign	-	+	+	+	+	+	+	+	+	+
TP	5	4								
FP	5	5								

Generating an ROC Curve

4. Repeat for all test records.

Actual Class	+	-	+	-	-	-	+	-	+	+
Model Output	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95
Assign	-	-	+	+	+	+	+	+	+	+
TP	5	4	4							
FP	5	5	4							

Generating an ROC Curve

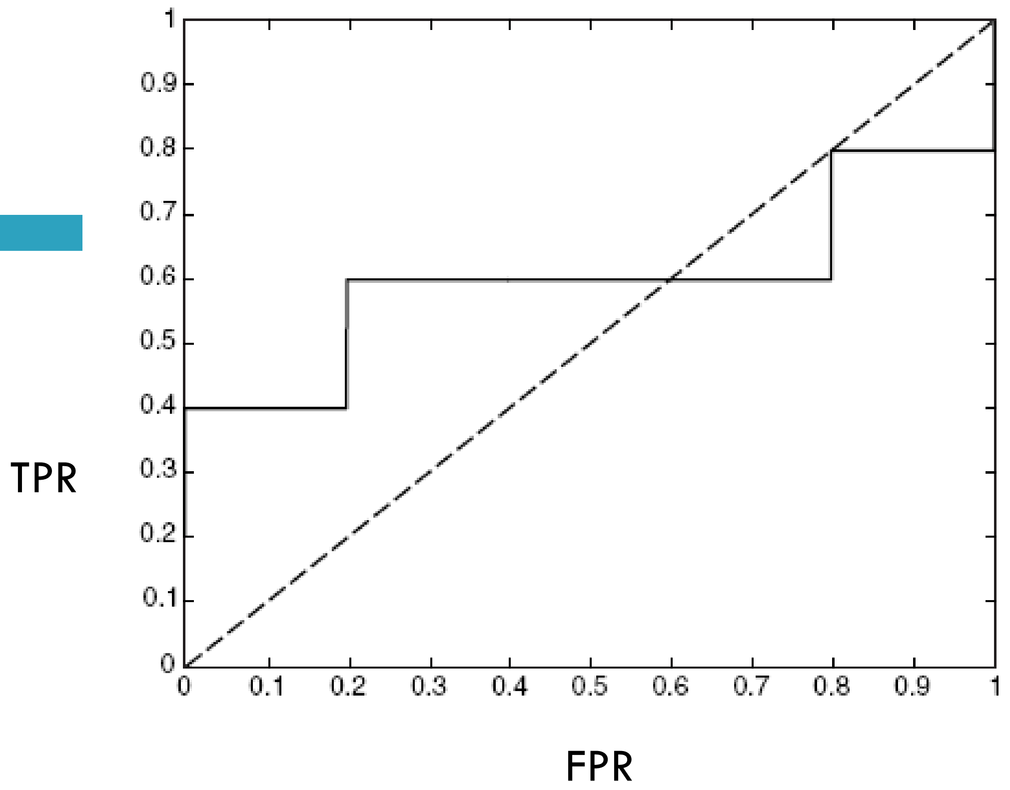
4. Repeat for all test records.

Actual Class	+	-	+	-	-	-	+	-	+	+	
Model Output	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0

TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

Generating an ROC Curve

5. Plot the TPR against the FPR.

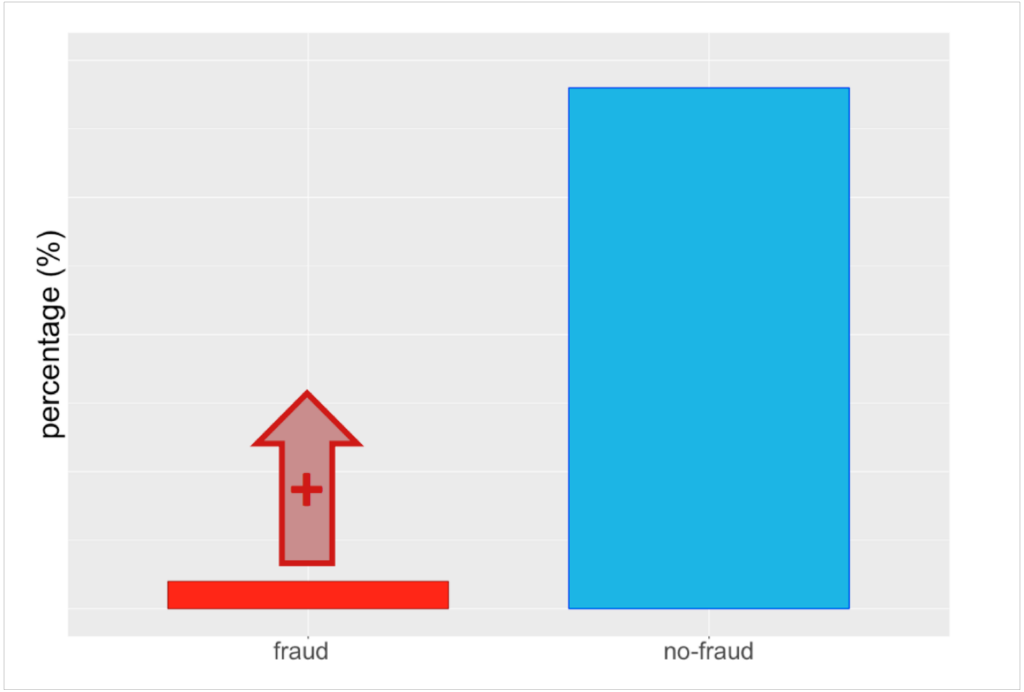


TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2		0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0		0

Sampling Techniques to Address Class Imbalance

- Undersampling
- Oversampling
- Synthetic Generation of Samples

Oversampling



Random Oversampling

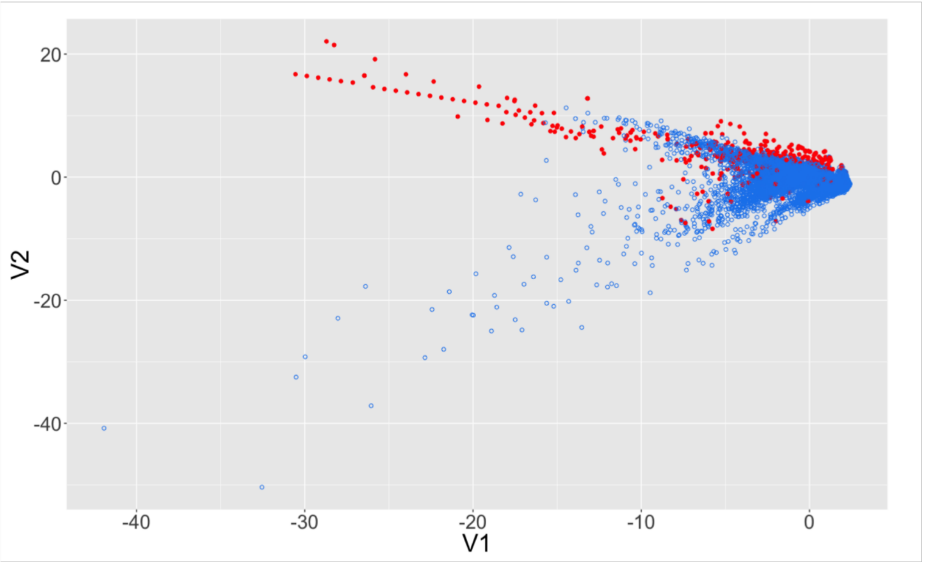
Original data

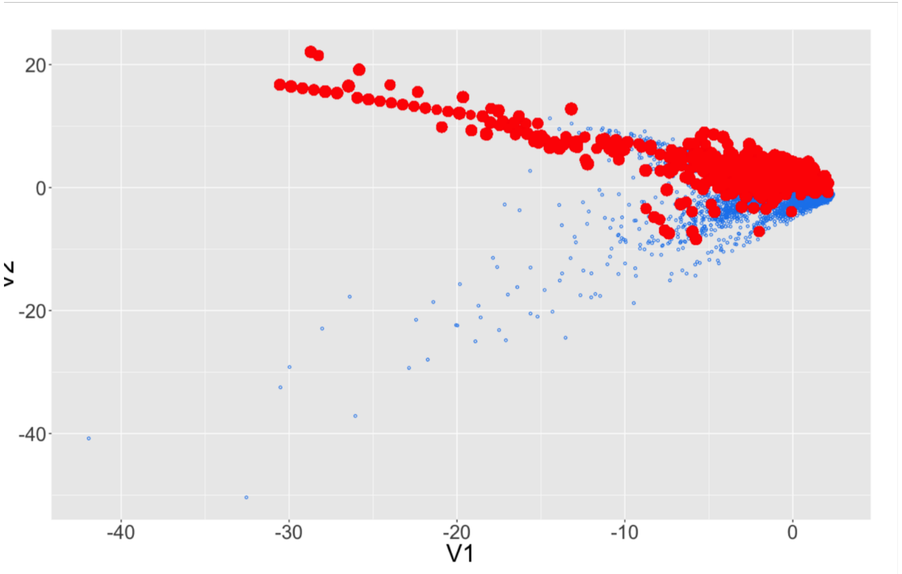
	ID	Variables	Class
Train	1	···	Fraud
	2	···	No fraud
	3	···	No fraud
	4	···	Fraud
	5	···	No fraud
	6	···	No fraud
Test	7	···	No fraud
	8	···	No fraud
	9	···	Fraud
	10	···	No fraud

Over-sampled data

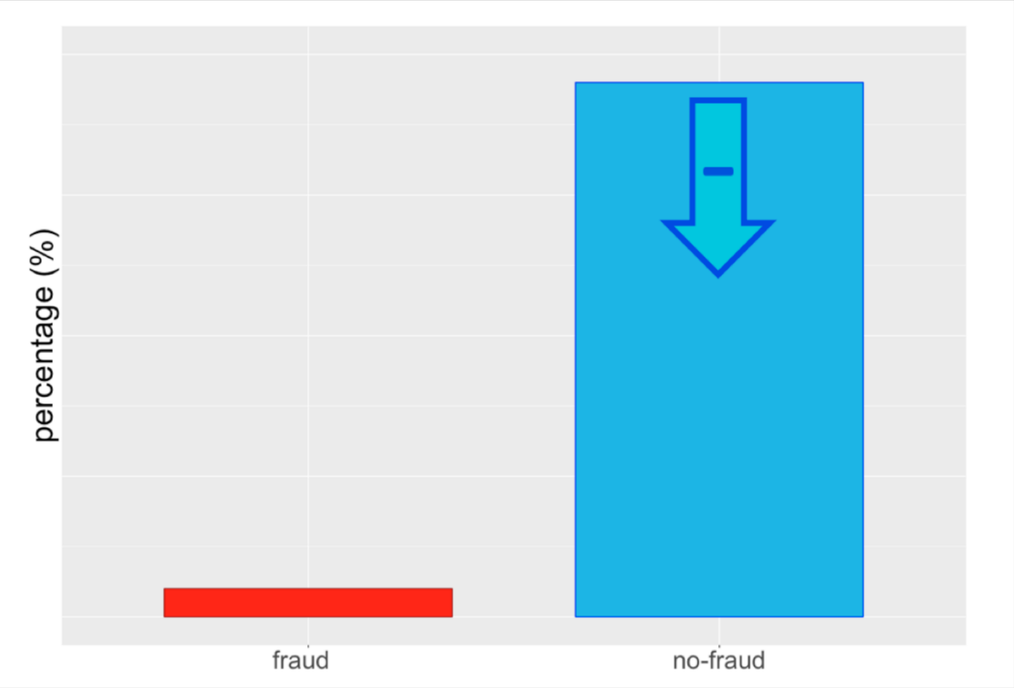
	ID	Variables	Class
Train	1	···	Fraud
	1	···	Fraud
	2	···	No fraud
	3	···	No fraud
	4	···	Fraud
	4	···	Fraud
	5	···	No fraud
	6	···	No fraud
Test	7	···	No fraud
	8	···	No fraud
	9	···	Fraud
	10	···	No fraud

Visualization





Random Undersampling



Random Undersampling

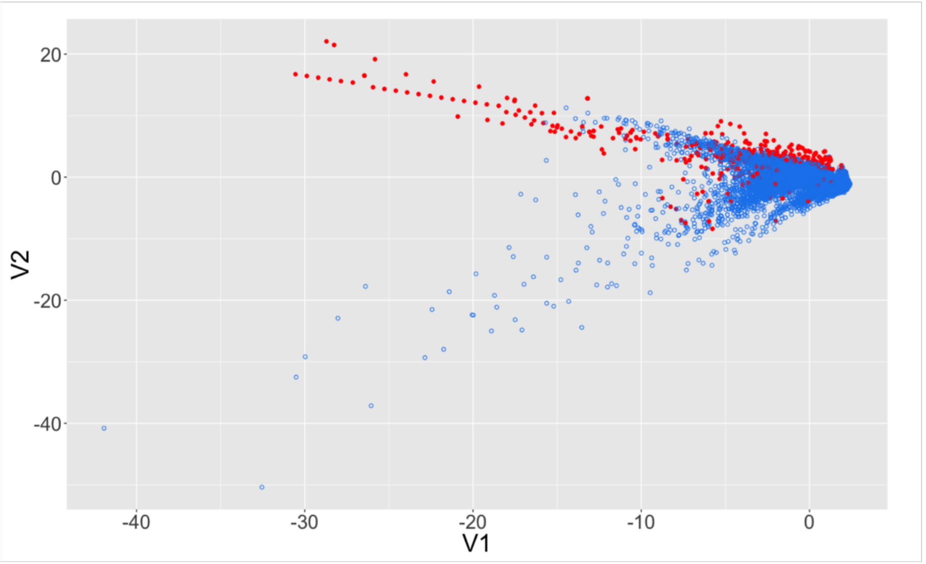
Original data

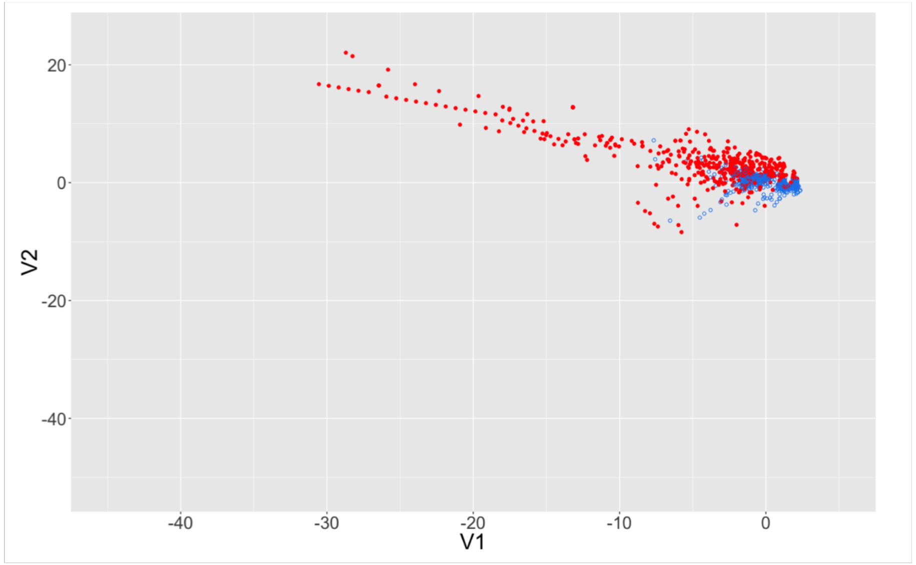
	ID	Variables	Class
	1	…	Fraud
	~~2~~	~~…~~	~~No Fraud~~
Train	3	…	No fraud
	4	…	Fraud
	~~5~~	~~…~~	~~No Fraud~~
	6	…	No fraud
	7	…	No fraud
Test	8	…	No fraud
	9	…	Fraud
	10	…	No fraud

Under-sampled data

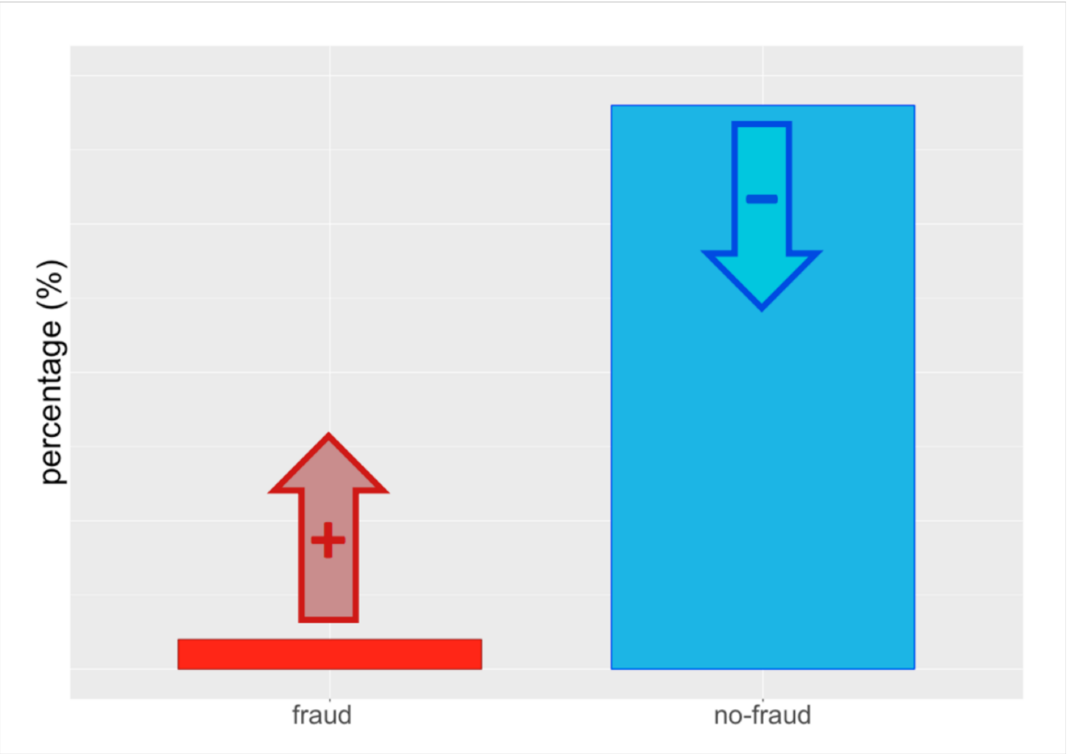
	ID	Variables	Class
	1	…	Fraud
Train	3	…	No fraud
	4	…	Fraud
	6	…	No fraud
	7	…	No fraud
Test	8	…	No fraud
	9	…	Fraud
	10	…	No fraud

Visualization





…Or Do Both!

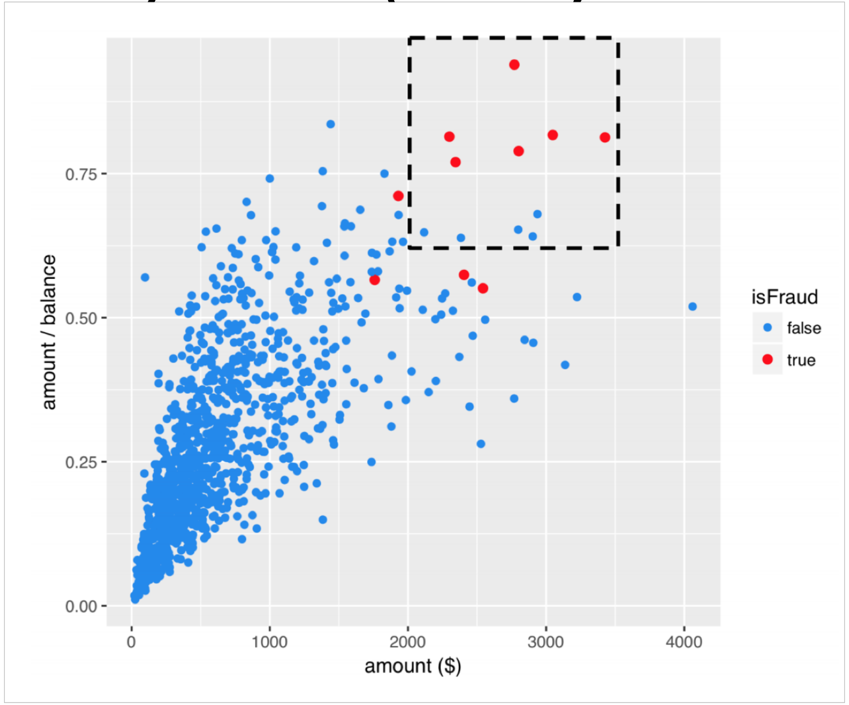


Synthetic Oversampling

- SMOTE: Synthetic Minority Oversampling TEchnique (Chawla et al., 2002)
- Over-sample minority class by creating synthetic minority cases

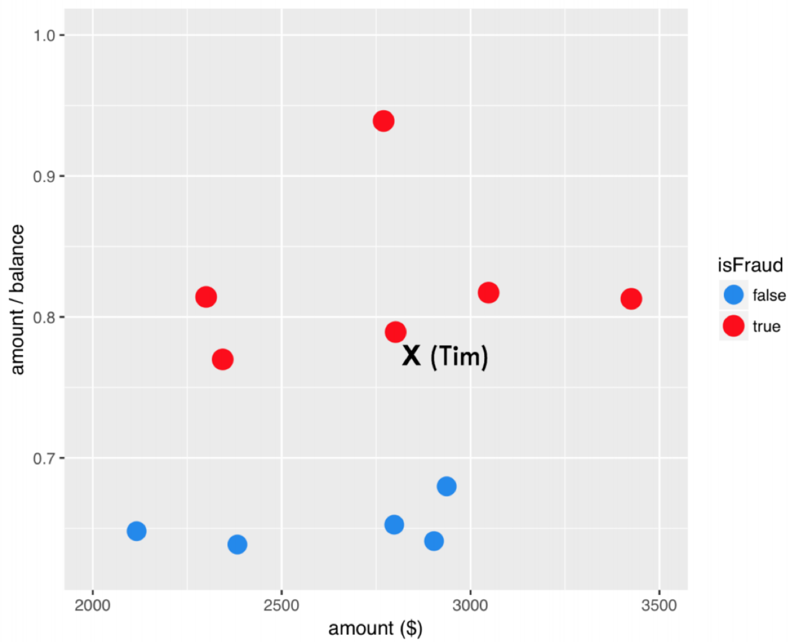
SMOTE

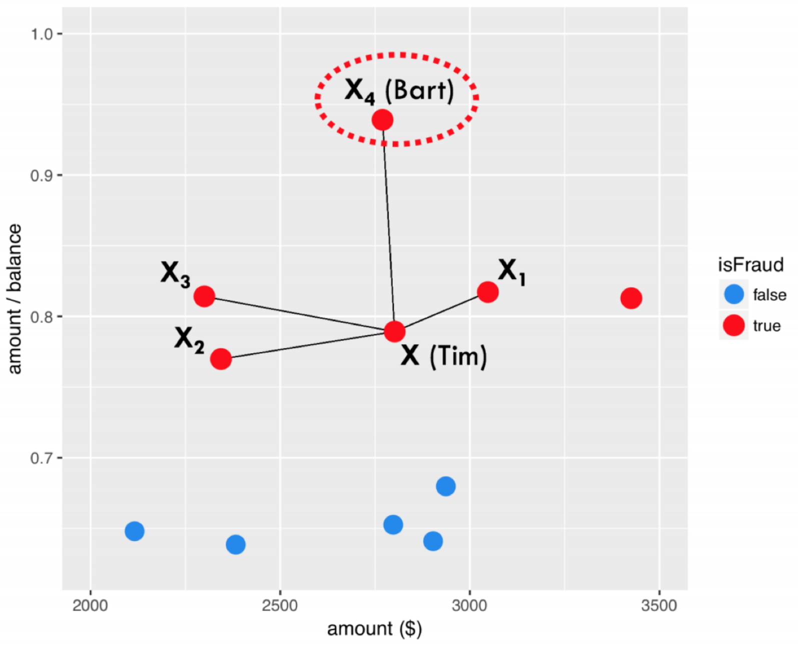
- Focus on minority cases (fraud, in this example)



SMOTE

- Given a record, find the k nearest neighbors and select one





SMOTE

- Create a synthetic example

X (Tim)

Amount	Ratio
2800	0.79

X_4 (Bart)

Amount	Ratio
2770	0.94

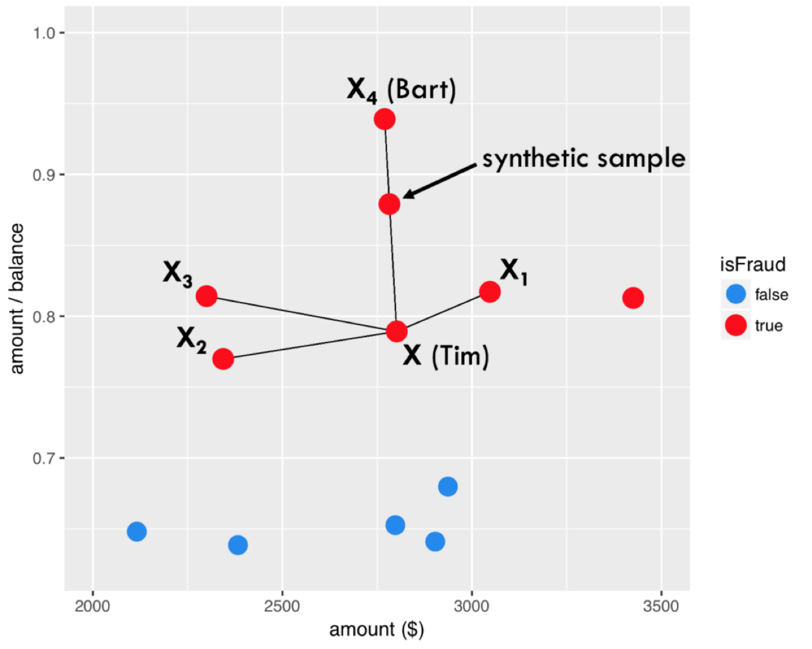
Choose random number between 0 and 1, e.g. **0.6**

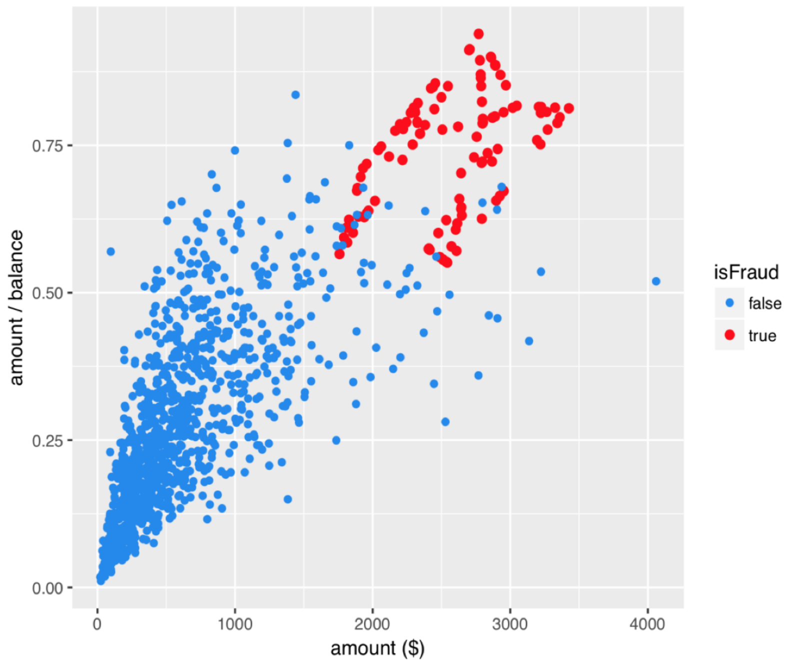
Synthetic sample

Amount	Ratio
2800 + **0.6** * (2770 - 2800) = **2782**	0.79 + **0.6** * (0.94 – 0.79) = **0.88**

SMOTE

- Repeat for desired number of synthetic examples





Ideas?

Multiclass Problems

- *Scenario:* target class is <u>more than 2 categories</u>
- *Motivation:* some machine learning algorithms are designed for binary classification
  - *Example:* Support Vector Machines (SVM)
- How to extend *binary classifiers* to handle *multiclass problems?*

Multiclass: One-Against-Rest

- Assume multiclass dataset with *K* target classes
- Decompose into *K* binary problems
- *Idea:* For each target class y_i create a single binary problem, with classifier C_i:
  - Class y_i: positive
  - All other classes: negative
- *Training:*
  - use all instances; each instance used in training each of the C_i classifiers
- *Testing:*
  - run testing instance through each classifier
  - record votes for each y_i class (negative prediction is a vote for all other classes)
  - class with most votes is the predicted class

Multiclass: One-Against-One

- Assume multiclass dataset with K target classes
- Train $K(K-1)/2$ binary classifiers (*many more than One-Against-Rest*)
- *Idea:* Each classifier distinguishes between pair of classes (y_i, y_j)
  - Classifier ignores records that don't belong to y_i or y_j
- *Training: use all instances; each instance only used in training "relevant classifiers" (K of them)*
- *Testing:*
  - run testing instance through each classifier
  - record votes for each y_i class
  - class with most votes is the predicted class

References

- *Introduction to Data Mining*, 1st edition, Tan et al.