Classification Error Metrics

METIS

datascope

%54 Democrats, %46 republicans

Classify:

Predict party using votes

%54 Democrats, %46 republicans

Classify:

Predict party using votes

Model performance:

"How many times did I get it right?"

%54 Democrats, %46 republicans

Classify:

Predict party using votes

Model performance:

“How many times did I get it right?”

Accuracy. % correct prediction of all predictions

%54 Democrats, %46 republicans
Classify:
Predict party using votes

Model performance:
"How many times did I get it right?"

Accuracy. % correct prediction of all predictions

95% accuracy: Good job!

%1 have leukemia, %99 are healthy

Classify:

Predict leukemia using health records and tests

%1 have leukemia, %99 are healthy

Classify:

Predict leukemia using health records and tests

"How many times did I get it right?"

Accuracy. % correct prediction of all predictions

%1 have leukemia, %99 are healthy
Classify:
Predict leukemia using health records and tests

"How many times did I get it right?"
Accuracy. % correct prediction of all predictions

Imagine the stupidest predictor:
"Always predict healthy"

%1 have leukemia, %99 are healthy

Classify:

Predict leukemia using health records and tests

"How many times did I get it right?"

Accuracy. % correct prediction of all predictions

Imagine the stupidest predictor:

"Always predict healthy"

What will its accuracy be?

%1 have leukemia, %99 are healthy
Classify:
Predict leukemia using health records and tests

"How many times did I get it right?"
Accuracy. % correct prediction of all predictions

Imagine the stupidest predictor:
"Always predict healthy"

What will its accuracy be?
It will be correct 99% of the time!
You won't catch any sick people. Useless.

Confusion Matrix

	P' (Predicted)	N' (Predicted)
P (Actual)	True Positive	False Negative
N (Actual)	False Positive	True Negative

Confusion Matrix

	Spam (Predicted)	Non-Spam (Predicted)	Accuracy
Spam (Actual)	27	6	81.81
Non-Spam (Actual)	10	57	85.07
Overall Accuracy			83.44

Confusion Matrix

	Spam (Predicted)	Non-Spam (Predicted)	Accuracy	
Spam (Actual)	27	6	81.81	Sensitivity
Non-Spam (Actual)	10	57	85.07	Specificity
Overall Accuracy			83.44	Accuracy

Confusion Matrix

	Spam (Predicted)	Non-Spam (Predicted)	Accuracy	
Spam (Actual)	0	10	0.0	Sensitivity
Non-Spam (Actual)	0	990	100.00	Specificity
Overall Accuracy			99	Accuracy

Confusion Matrix

	P′ (Predicted)	N′ (Predicted)
P (Actual)	True Positive	False Negative
N (Actual)	False Positive	True Negative

Sensitivity = TP / (TP + FN)

Specificity = TN / (TN + FP)

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Confusion Matrix

	P′ (Predicted)	N′ (Predicted)
P (Actual)	True Positive	False Negative
N (Actual)	False Positive	True Negative

Sensitivity = TP / (TP + FN) TRUE POSITIVE RATE

Specificity = TN / (TN + FP) TRUE NEGATIVE RATE

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Precision and recall

Precision and recall

Precision. Out of all cases I predicted as positive, how many times was I right?
(% times I was right when I told somebody they had leukemia)

Recall. Out of all the (few) positive cases, how many did I find?
(% of actual leukemia patients I could catch with my classifier)

Confusion Matrix

	Spam (Predicted)	Non-Spam (Predicted)	Accuracy
Spam (Actual)	27	6	81.81
Non-Spam (Actual)	10	57	85.07
Overall Accuracy			83.44

Precision = 27 / 37 = 73.0%

Recall = 27 / 33 = 81.8%

Confusion Matrix

	Spam (Predicted)	Non-Spam (Predicted)	Accuracy
Spam (Actual)	0	10	0.0
Non-Spam (Actual)	0	990	100.00
Overall Accuracy			99

Precision = 0 / 0 Undefined!

Recall = 2 / 10 = 20%

Confusion Matrix

	P′ (Predicted)	n′ (Predicted)
P (Actual)	True Positive	False Negative
n (Actual)	False Positive	True Negative

Precision = TP / (TP + FP)

Recall = TP / (TP + FN)

Confusion Matrix

	P′ (Predicted)	n′ (Predicted)
P (Actual)	True Positive	False Negative
n (Actual)	False Positive	True Negative

Focusing on a single class (positive: the one with small prevalence) in skewed cases

Precision = TP / (TP + FP)

Recall = TP / (TP + FN)

Confusion Matrix

	P′ (Predicted)	n′ (Predicted)
P (Actual)	True Positive	False Negative
n (Actual)	False Positive	True Negative

Focusing on a single class (positive: the one with small prevalence) in skewed cases

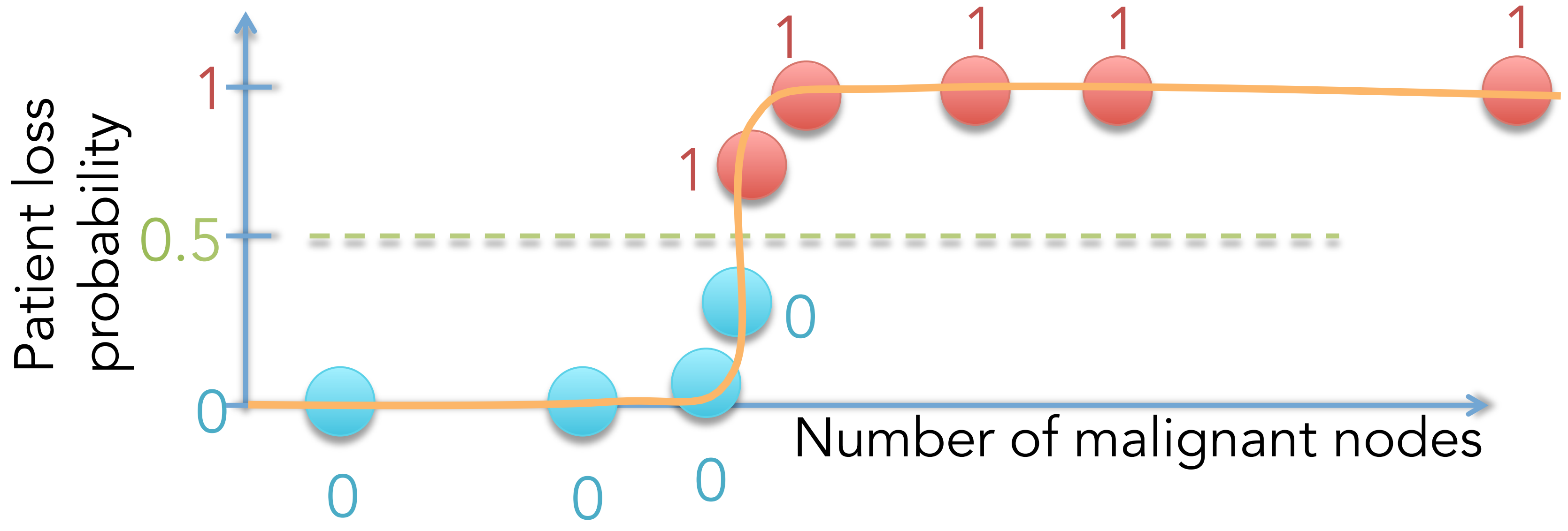
Precision = TP / (TP + FP)

Recall = TP / (TP + FN)

F1 = Their harmonic mean

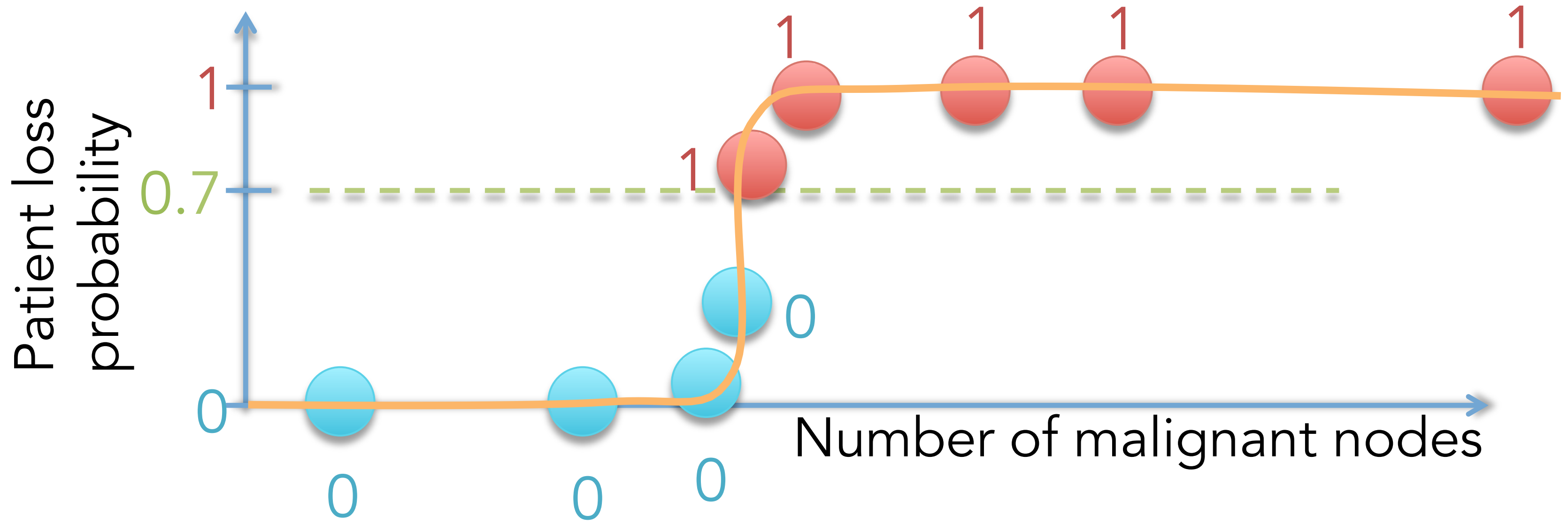
$$F1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Logistic regression

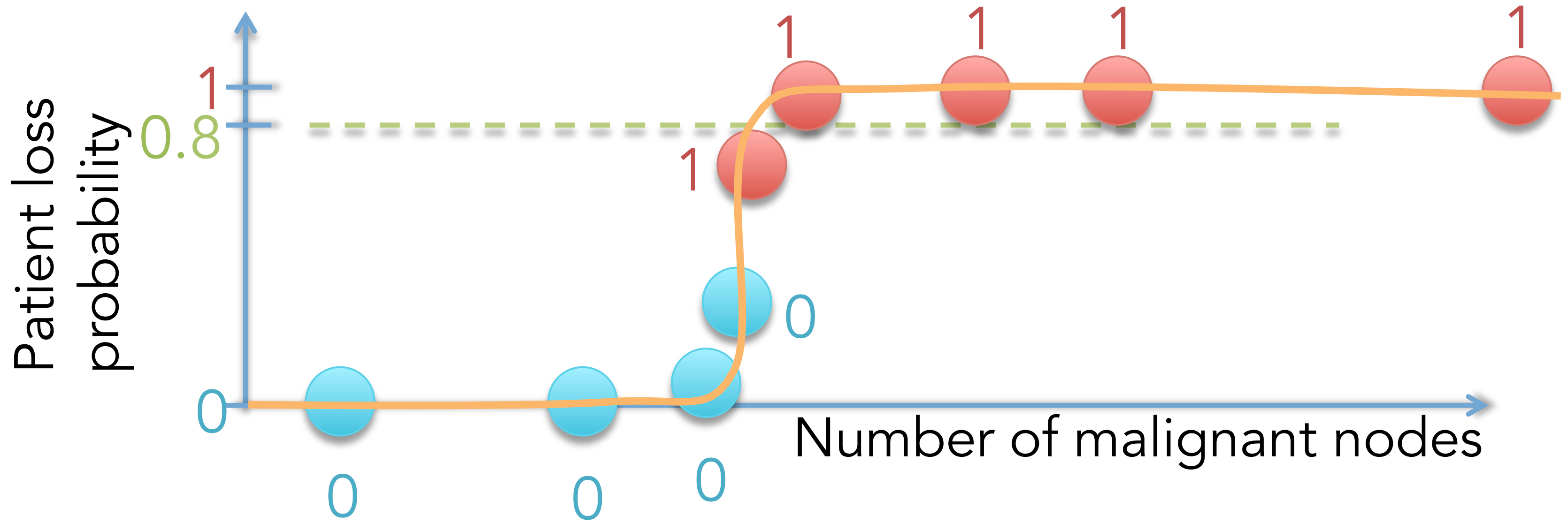


$$y_\beta(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x + \varepsilon)}}$$

Logistic regression

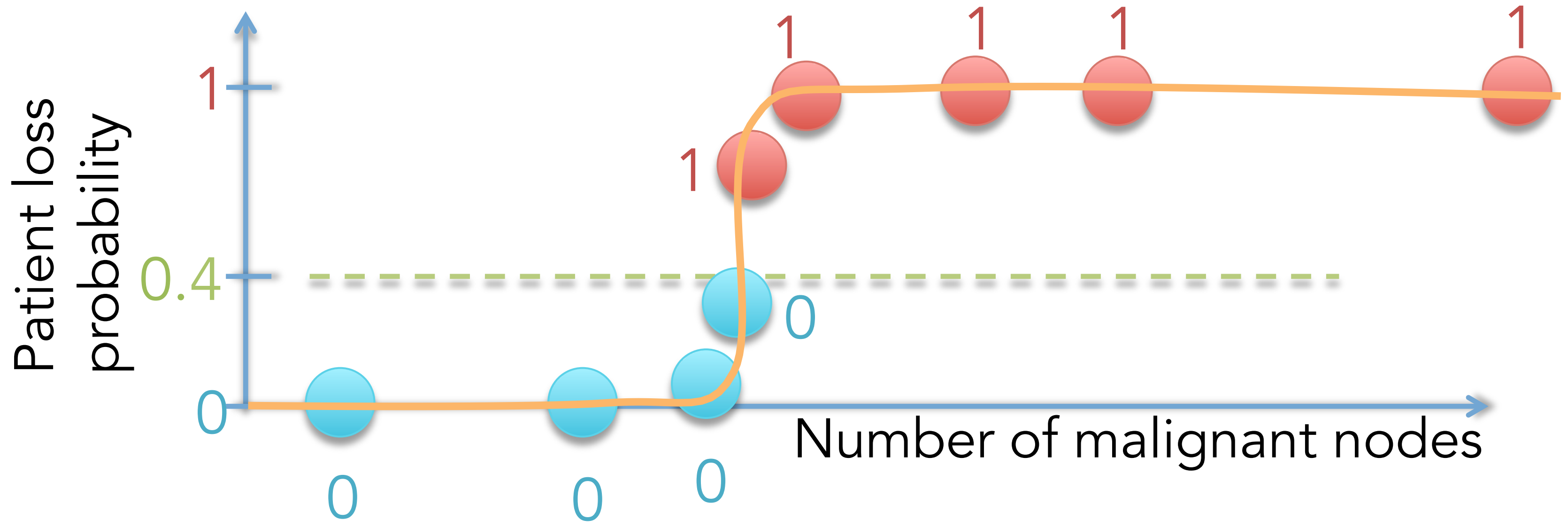


Logistic regression

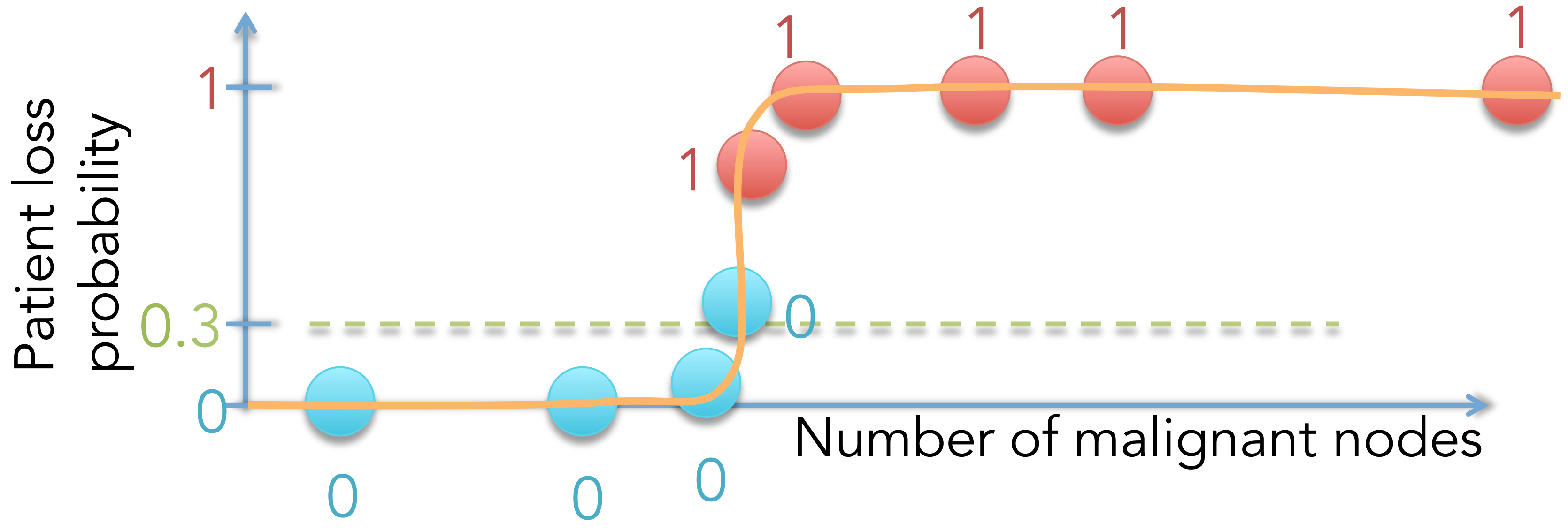


Higher threshold: Only call it positive if absolutely sure
To make most positive calls right, give up on catching all
lower recall, higher precision
lower True Positive Rate, lower False Positive Rate

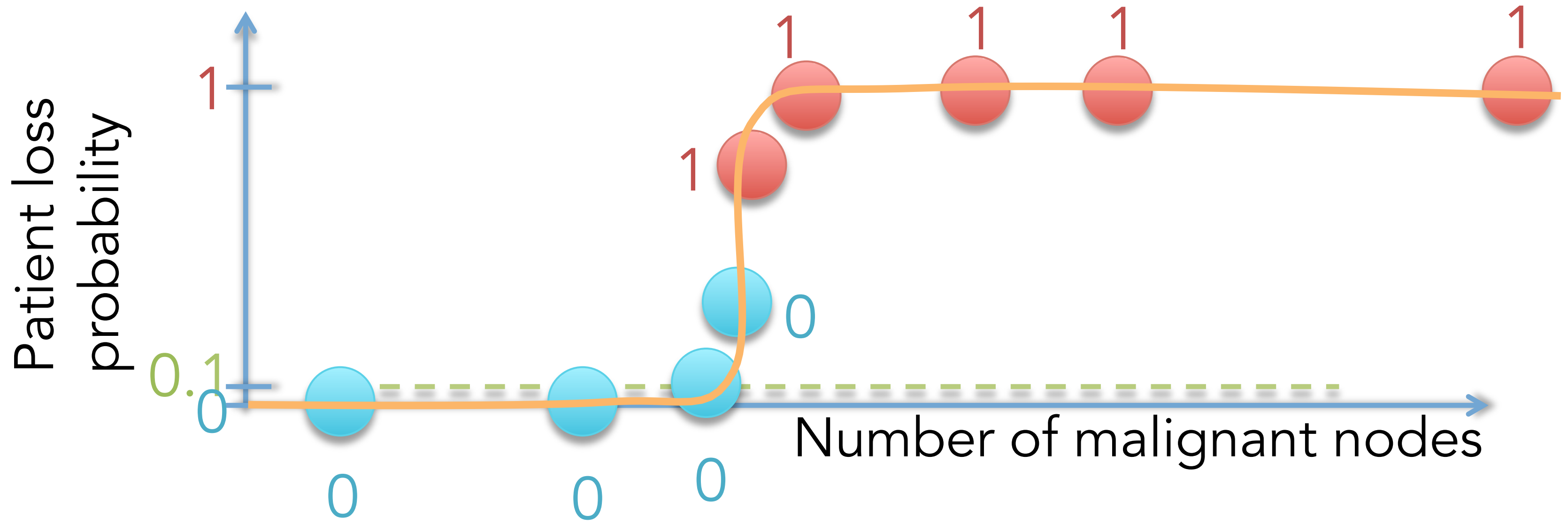
Logistic regression



Logistic regression



Logistic regression



Lower threshold: Ease criteria for calling it positive
Can catch more positives by accepting edge cases
higher recall, lower precision
higher True Positive Rate, higher False Positive Rate

How to tune precision and recall?

Google.
Forget Recall. There are millions of relevant search results. Catching all of them is not important: Most people only look at the first page of results.
Precision way more important. If one of those ten links in the first page is not relevant to the search, you'll lose customers.

How to tune precision and recall?

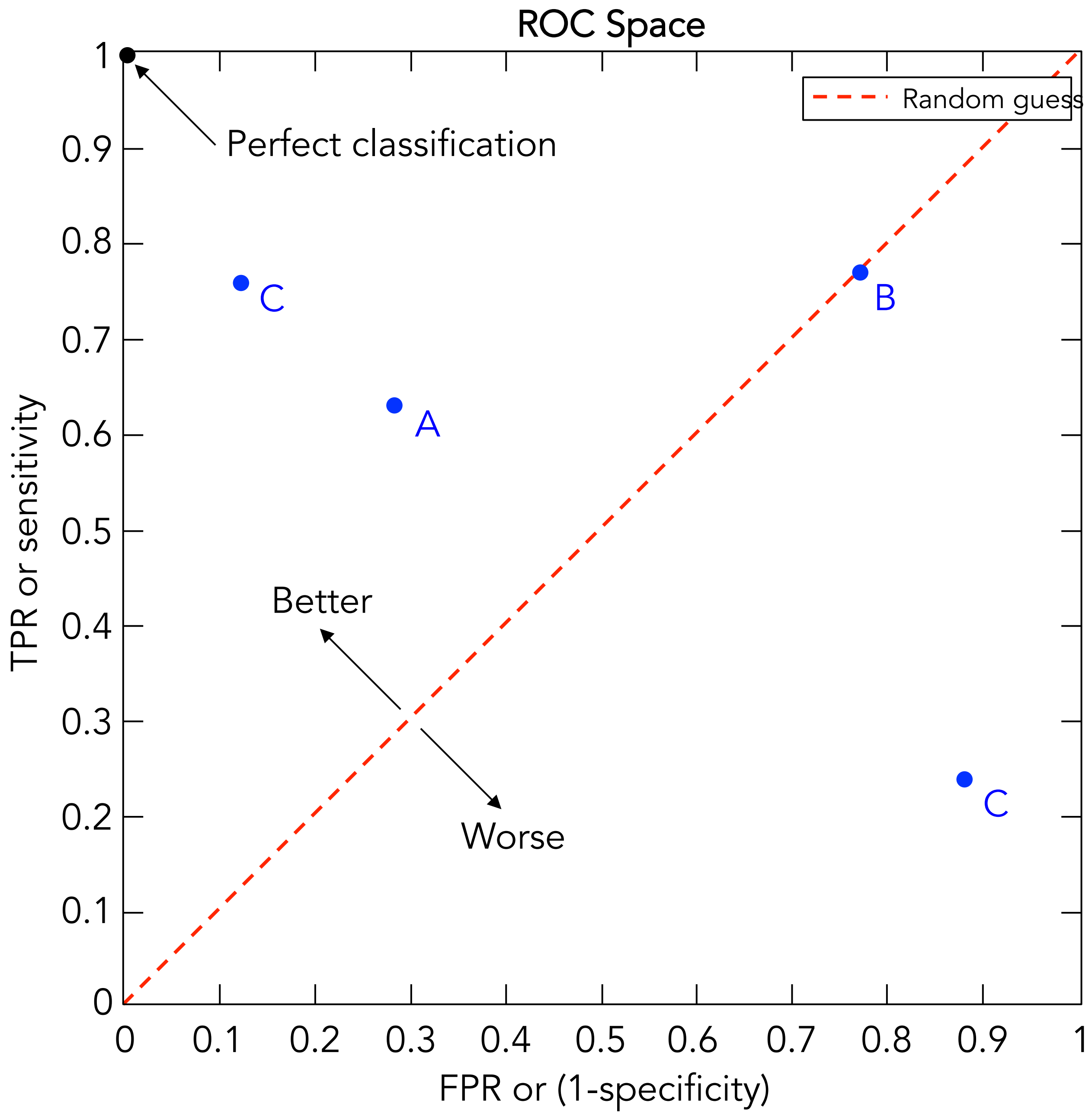
HIV Tests.
Recall very very important. We want to catch all sick people. Imagine an HIV+ person taking the test and being told they are HIV-. Worst outcome. Nope.
Precision also important. Telling someone that they are HIV+ when they are not is a devastating mistake. <u>Not as bad</u> as letting HIV+ slip through, but we still cannot turn the knob all the way to recall without a care about this, either.

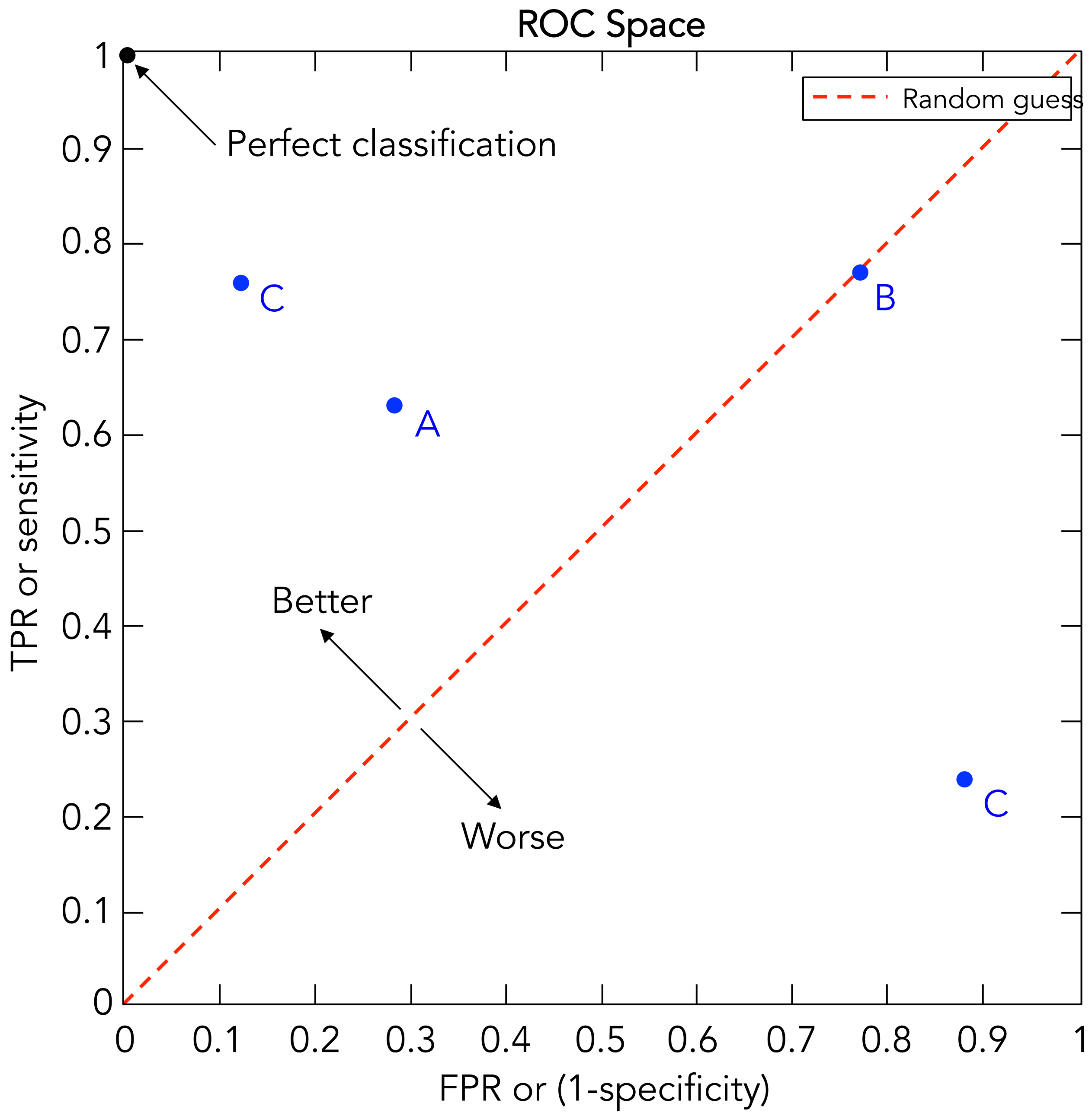
How to tune precision and recall?

United States Legal System.
Higher Precision at the cost of Recall. Burden of proof. Reasonable doubt = Acquittal. These mean that the “threshold” for classifying someone with the ‘criminal’ label is set very high.
High precision means making sure everyone sent to jail is indeed guilty.
High recall means making sure all criminals go to jail.
The legal system is ready to pay the cost of letting some criminals go to ensure no innocents go to jail.

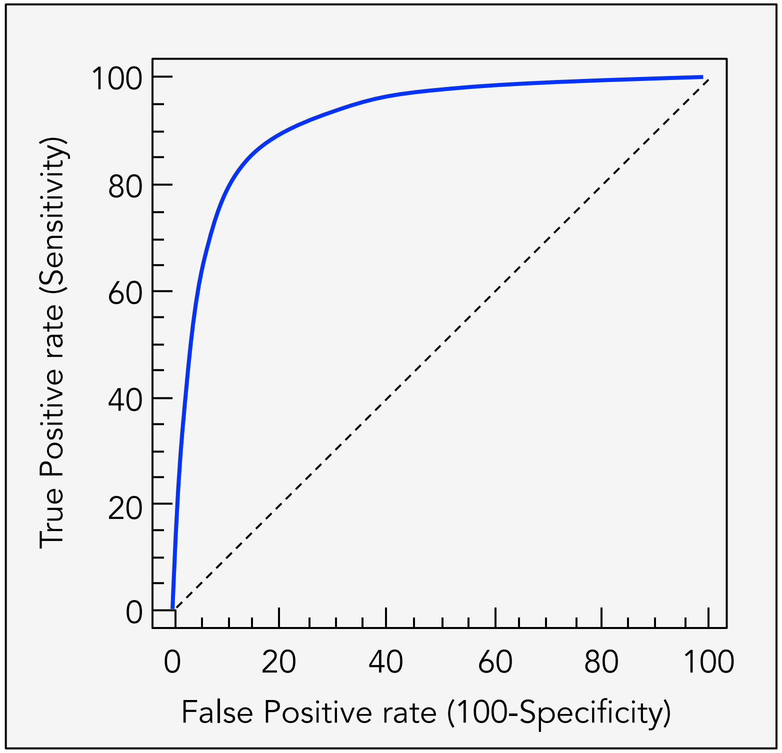
Each threshold is a different model

Plot their True Positive Rate & False Positive Rate

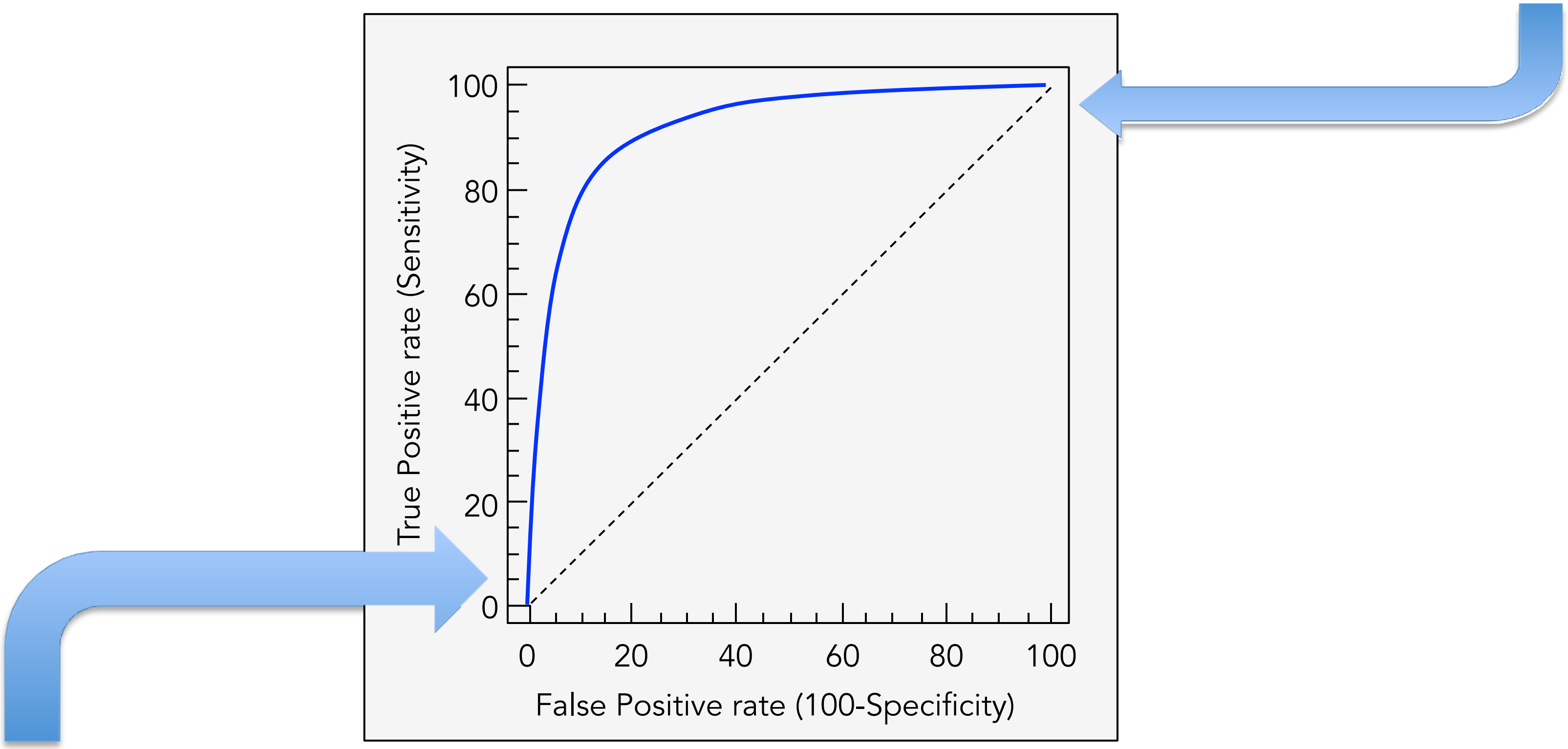

ROC Space
Random guess
Perfect classification
C
A
B
C
Better
Worse
TPR or sensitivity
FPR or (1-specificity)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



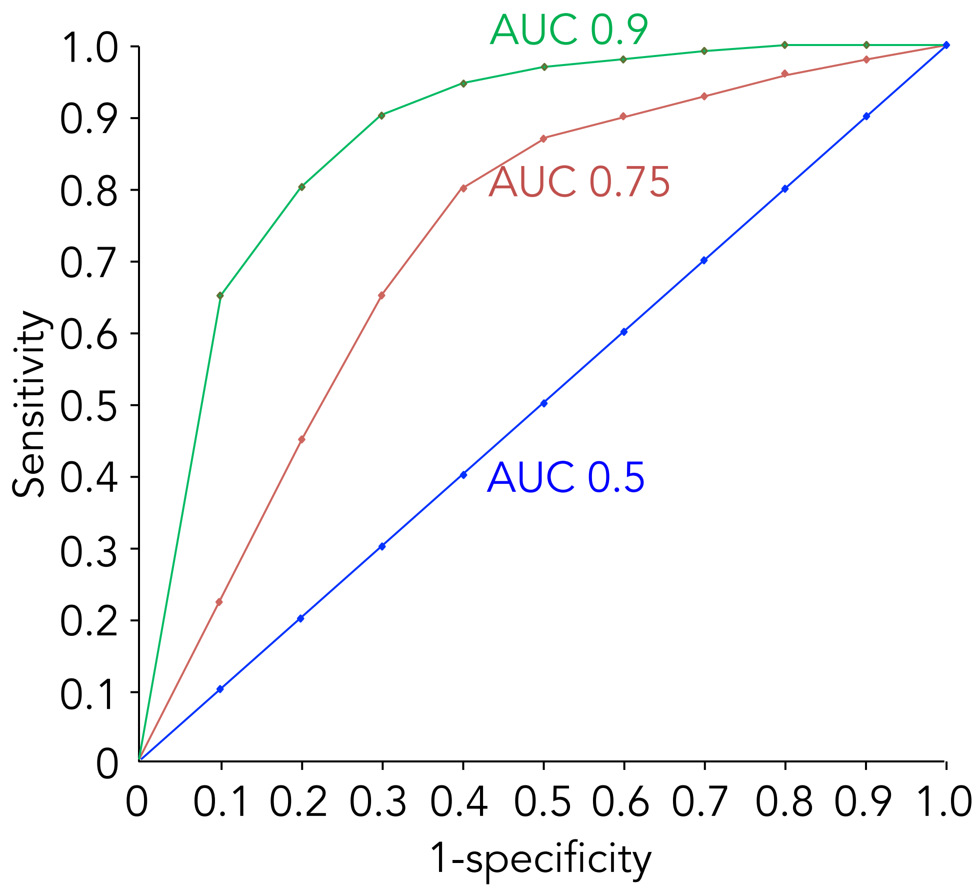
Receiver Operating Characteristic


100
80
60
40
20
0
True Positive rate (Sensitivity)
0
20
40
60
80
100
False Positive rate (100-Specificity)

Lower threshold: Ease criteria for calling it positive
Can catch more positives by accepting edge cases
higher recall, lower precision
higher True Positive Rate, higher False Positive Rate

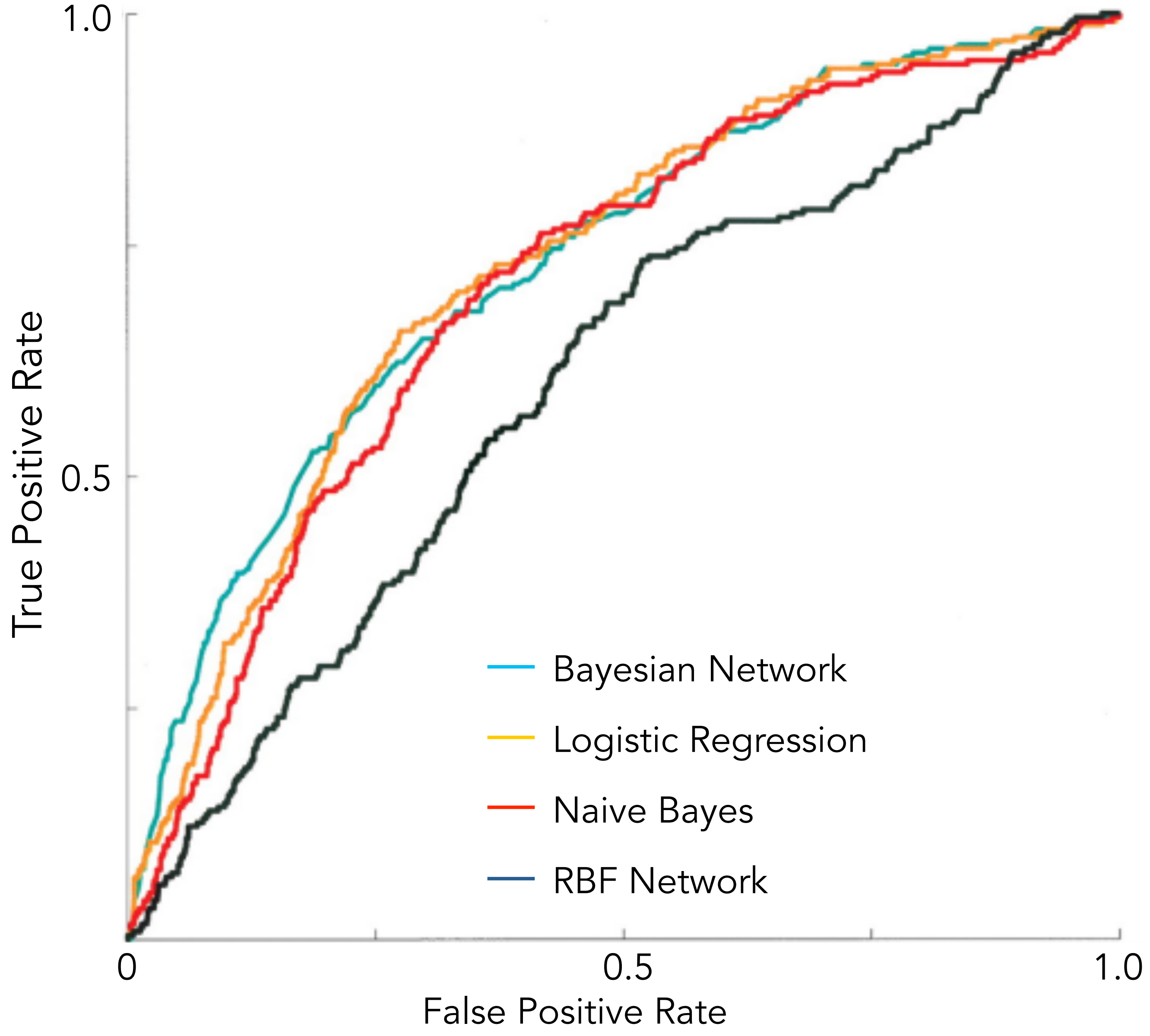


Higher threshold: Only call it positive if absolutely sure
To make most positive calls right, give up on
lower recall, higher precision
lower True Positive Rate, lower False Positive Rate



Area under curve (AUC)

An evaluation of a classification algorithm (including all possible thresholds)


1.0
0.5
True Positive Rate
0
0.5
1.0
False Positive Rate
Bayesian Network
Logistic Regression
Naive Bayes
RBF Network

from sklearn.metrics import ......

Classification metrics

See the *Classification metrics* section of the user guide for further details.

metrics.accuracy_score(y_true, y_pred[, ...])	Accuracy classification score.
metrics.auc(x, y[, reorder])	Compute Area Under the Curve (AUC) using the trapezoidal rule
metrics.average_precision_score(y_true, y_score)	Compute average precision (AP) from prediction scores
metrics.classification_report(y_true, y_pred)	Build a text report showing the main classification metrics
metrics.confusion_matrix(y_true, y_pred[, ...])	Compute confusion matrix to evaluate the accuracy of a classification
metrics.fl_score(y_true, y_pred[, labels, ...])	Compute the F1 score, also known as balanced F-score or F-measure
metrics.fbeta_score(y_true, y_pred, beta[, ...])	Compute the F-beta score
metrics.hamming_loss(y_true, y_pred[, classes])	Compute the average Hamming loss.
metrics.hinge_loss(y_true, pred_decision[, ...])	Average hinge loss (non-regularized)
metrics.jaccard_similarity_score(y_true, y_pred)	Jaccard similarity coefficient score
metrics.log_loss(y_true, y_pred[, eps, ...])	Log loss, aka logistic loss or cross-entropy loss.
metrics .matthews_corrcoef(y_true, y_pred)	Compute the Matthews correlation coefficient (MCC) for binary classes
metrics .precision_recall_curve(y_true, ...)	Compute precision-recall pairs for different probability thresholds
metrics.precision_recall_fscore_support(...)	Compute precision, recall, F-measure and support for each class
metrics.precision_score(y_true, y_pred[, ...])	Compute the precision
metrics.recall_score(y_true, y_pred[, ...])	Compute the recall
metrics.roc_auc_score(y_true, y_score[, ...])	Compute Area Under the Curve (AUC) from prediction scores
metrics.roc_curve(y_true, y_score[, ...])	Compute Receiver operating characteristic (ROC)
metrics.zero_one_ioss(y_true, y_pred[, ...])	Zero-one classification loss.

Always remember,

Fit the model to a **training set**,

Calculate performance
(accuracy, precision, recall, f1, AUC, etc.)
on a **test set**

or (better) on a k-fold **cross validation** scheme http://scikit-learn.org/stable/modules/model_evaluation.html

```
from sklearn.cross_validation import cross_val_score

KNN = KNearestNeighbors( 10 )

cross_val_score( KNN, scoring="recall" )
```