

COMP 6321 Machine Learning

Loss Functions & Hyperparameter Search

Computer Science & Software Engineering
Concordia University, Fall 2020



Beyond Accuracy

- Scikit learn classifiers report “accuracy” by default
 - e.g. `my_classifier.score(X, y_true)` returns accuracy
- Depending on application and/or the data set, “accuracy” may not be the best measure.
- Two main aspects to consider:
 - Application-driven asymmetric costs that explicitly assign a different “penalty” for making different kinds of mistakes, based on some knowledge of stakeholder priorities
 - Data-driven asymmetric costs, implicitly caused by class imbalance in the training data, e.g., some classes are easier to collect or to label than other classes.

WANT THIS TO INFLUENCE PREDICTIONS
(Design loss carefully, with stakeholders involved)

DON'T WANT THIS TO INFLUENCE PREDICTIONS
(Try different class weightings as part of hyperparameter search.)

Asymmetric loss and class imbalance

- Examples of application-driven asymmetric loss:
 - In early-stage cancer screening, the cost of “failing to detect a case of cancer” may be higher than the cost of “flagging a patient who turns out not to have cancer.”
 - A bank may decide that the cost of “lending to a person who defaults on their debt” is much higher than the cost of “denying a loan to person who would have paid their debt”
- Examples of class imbalance in the data:
 - 10 million possible binding sites for a protein, but only 10 actual binding sites
 - A face detector scans millions of possible locations in images for every face that exists in the data. If it just predicts “no face” every time, it is “accurate”?

Example from PRML book

Figure 1.25 An example of a loss matrix with elements L_{kj} for the cancer treatment problem. The rows correspond to the true class, whereas the columns correspond to the assignment of class made by our decision criterion.

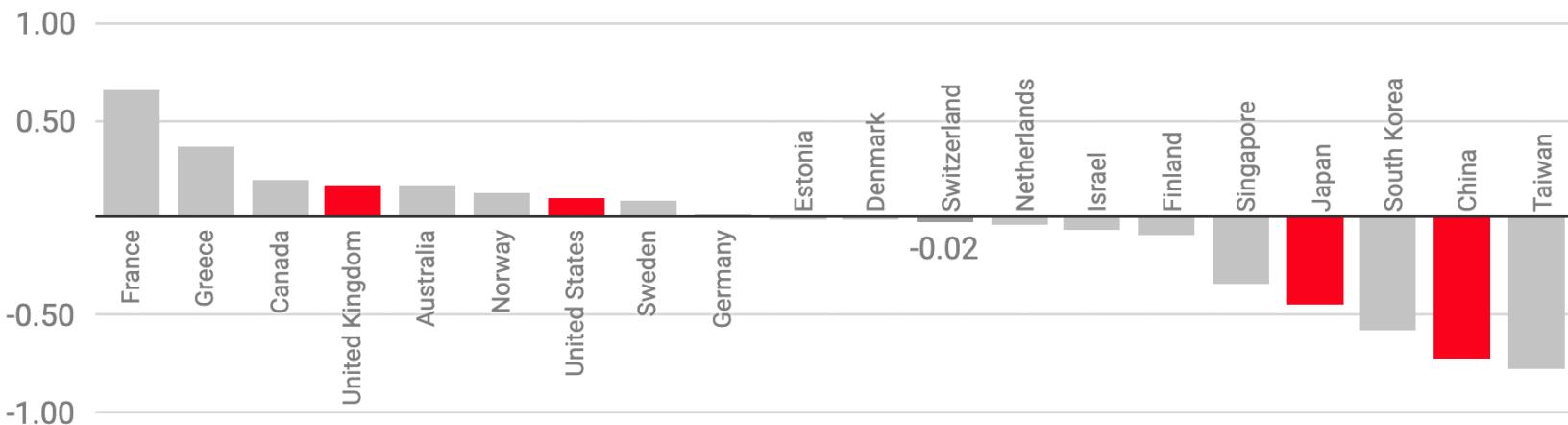
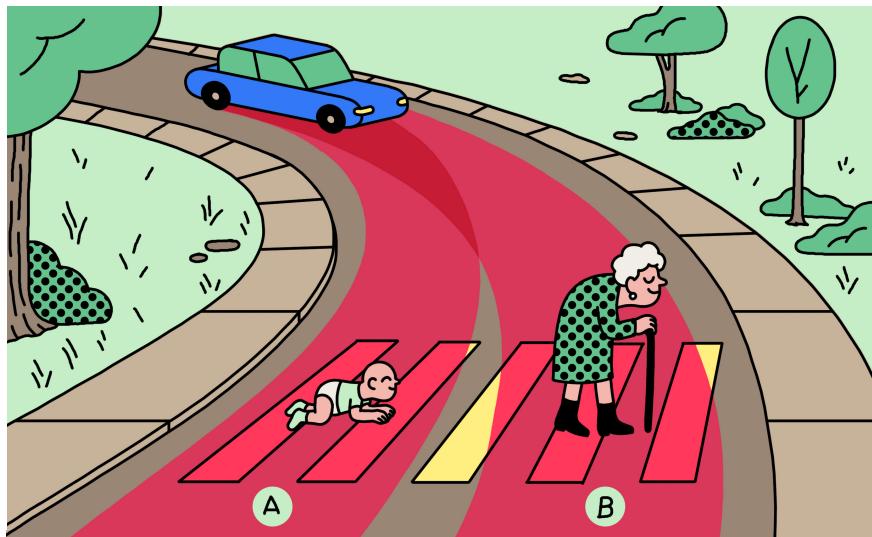
		prediction	
		cancer	normal
truth	cancer	(0	1000)
	normal	1	0

1.5.2 Minimizing the expected loss

For many applications, our objective will be more complex than simply minimizing the number of misclassifications. Let us consider again the medical diagnosis problem. We note that, if a patient who does not have cancer is incorrectly diagnosed as having cancer, the consequences may be some patient distress plus the need for further investigations. Conversely, if a patient with cancer is diagnosed as healthy, the result may be premature death due to lack of treatment. Thus the consequences of these two types of mistake can be dramatically different. It would clearly be better to make fewer mistakes of the second kind, even if this was at the expense of making more mistakes of the first kind.

Should a self-driving car kill the baby or the grandma? Depends on where you're from.

The infamous “trolley problem” was put to millions of people in a global study, revealing how much ethics diverge across cultures.



A comparison of countries piloting self-driving cars: If the bar is closer to 1, respondents placed a greater emphasis on sparing the young; if the bar is closer to -1, respondents placed a greater emphasis on sparing the old; 0 is the global average.

Binary classifiers

2x2 “confusion matrix”

		prediction	
		n	p
known	n	T_n	F_p
	p	F_n	T_p

- Classifier predicts *positive* or *negative* for each example
- **Goal:** Evaluate performance w.r.t. a particular data set
- Relevant quantities of that data set:
 - N_p # known positives in the data set
 - N_n # known negatives in the data set
- Relevant quantities of classifier *applied* to data set:
 - T_p # known positives correctly classified (“*true positives*”)
 - F_n # known positives incorrectly classified (“*false negatives*”)
 - F_p # known negatives incorrectly classified (“*false positives*”)
 - T_n # known negatives correctly classified (“*true negatives*”)
- Obviously $N_p = T_p + F_n$ and $N_n = T_n + F_p$

Also called “recall” or “sensitivity”

Binary classifier performance

- The **true positive rate** (TPR) is important quantity:

$$TPR = \frac{T_p}{T_p + F_n} = \frac{T_p}{N_p}$$

- “The probability that a randomly-selected known positive will be correctly classified as positive.”

$$\mathbf{t} = [0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1]^T$$

$$\hat{\mathbf{t}} = [0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1]^T$$

$$TPR = \frac{5}{5 + 1} = 0.833$$

$p(\text{cries wolf} \mid \text{wolf})$



Chance that boy *cries wolf* when there *is a wolf*

Binary classifier performance

- The ***false positive rate*** (FPR) is also important:

$$FPR = \frac{F_p}{F_p + T_n} = \frac{F_p}{N_n}$$

- “The probability that a randomly-selected known negative will be incorrectly classified as positive.”

$$\mathbf{t} = [0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1]^T$$

$$\hat{\mathbf{t}} = [0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1]^T$$

$$FPR = \frac{2}{2+2} = 0.5$$



$p(\text{cries wolf} \mid \neg\text{wolf})$

Chance that boy *cries wolf* when there *isn't a wolf*

Binary classifier performance

- The ***precision*** (P) is similar to $1 - FPR$, but different:

$$P = \frac{T_p}{T_p + F_p} \leftarrow \text{not equal to } N_p \text{ or } N_n$$

- “The probability that a randomly-selected positive prediction will have been correctly classified.”**

$$\mathbf{t} = [0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1]^T$$

$$\hat{\mathbf{t}} = [0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1]^T$$

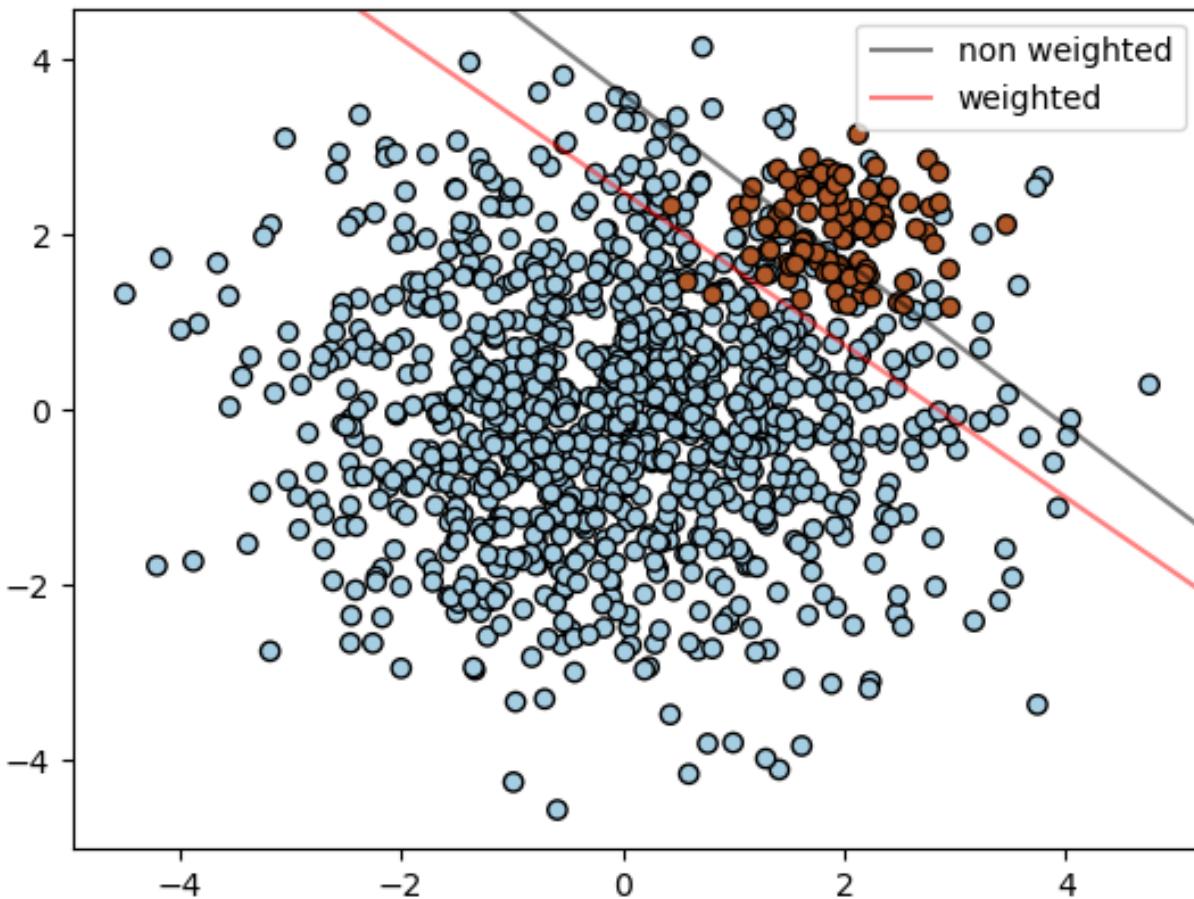
$$P = \frac{5}{5+2} = 0.714$$


 
 $p(\text{wolf} \mid \text{cries wolf})$

Chance that there *is a wolf* given that boy *cries wolf*

Example of unbalanced classes

```
sklearn.svm.SVC(kernel='linear', class_weight={-1: 1.0,  
+1: 10.0})
```



Weight of 10.0 on class +1 means making a *false negative* error in the training set costs 10x more than making a false positive error.

For SVM, implemented by applying different C (slack penalty) for positive than for negative training cases.

$$TPR = \frac{T_p}{T_p + F_n} = \frac{T_p}{N_p} \qquad FPR = \frac{F_p}{F_p + T_n} = \frac{F_p}{N_n}$$

Receiver Operating Characteristic

- The “ROC” curve measures classifier performance using only two computed values (FPR , TPR)

History [edit]

The ROC curve was first used during [World War II](#) for the analysis of radar signals before it was employed in [signal detection theory](#).^[40]

Following the [attack on Pearl Harbor](#) in 1941, the United States army began new research to increase the prediction of correctly detected Japanese aircraft from their radar signals. For these purposes they measured the ability of a radar receiver operator to make these important distinctions, which was called the Receiver Operating Characteristic.^[41]

$$TPR = \frac{T_p}{T_p + F_n} = \frac{T_p}{N_p} \qquad \qquad FPR = \frac{F_p}{F_p + T_n} = \frac{F_p}{N_n}$$

Receiver Operating Characteristic

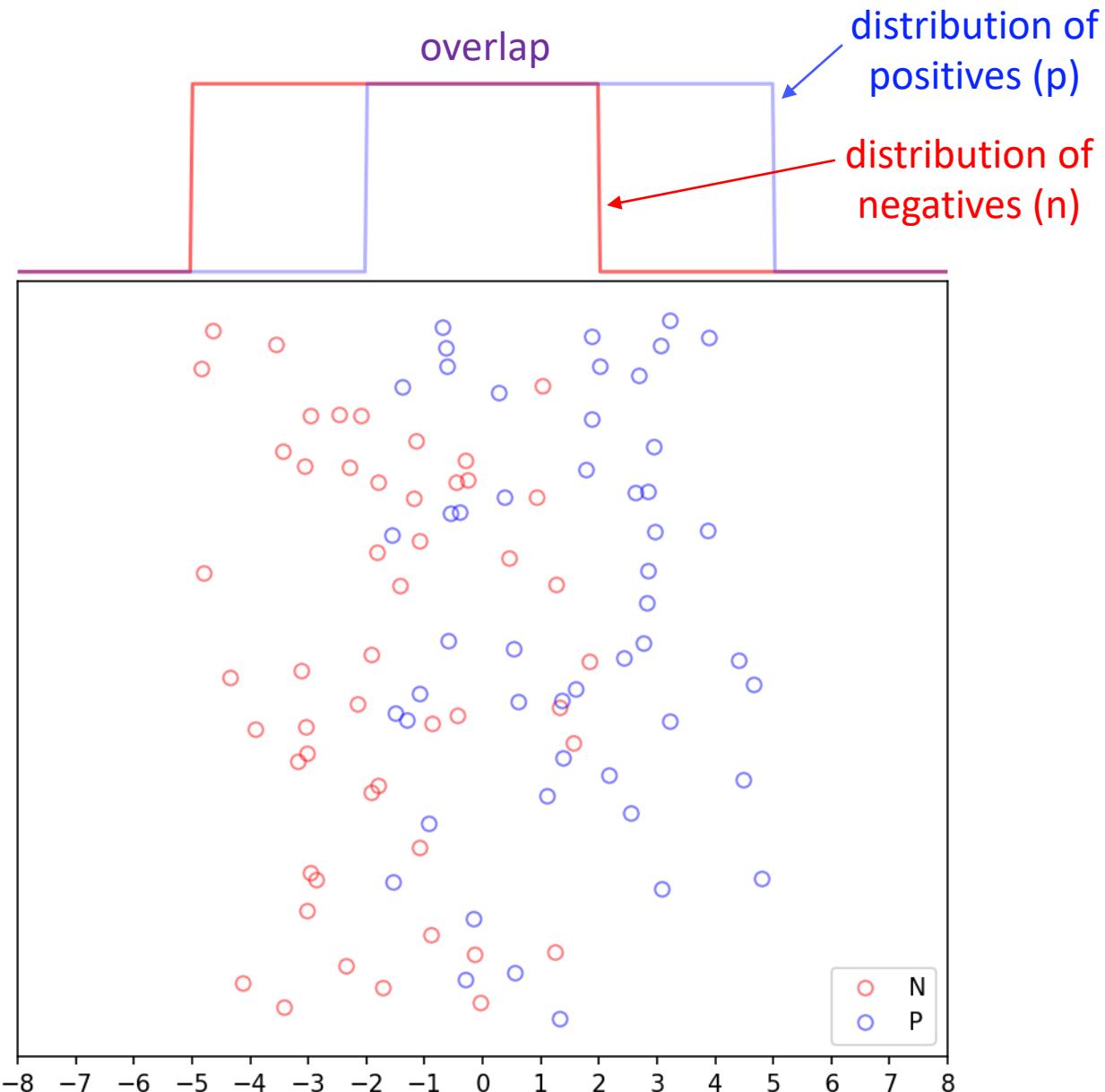
- The “ROC” curve measures classifier performance using only two computed values (FPR , TPR)
- A classifier with perfect performance, no errors of any kind, would be at point $(0, 1)$ in ROC space
 - $FPR = 0$: no known negatives are classified as positives
 - $TPR = 1$: all known positives are classified as positives
- Classifiers often output a *score* that is *thresholded*:
 - SVM score = signed dist from hyperplane, threshold @ 0.0
 - Logistic regression score = probability, threshold @ 0.5
- Other choices of threshold give different (FPR , TPR)
 - threshold @ $-\infty$ gives $(1, 1)$, @ $+\infty$ gives $(0, 0)$

$$TPR = \frac{T_p}{T_p + F_n} = \frac{T_p}{N_p} = R \quad P = \frac{T_p}{T_p + F_p}$$

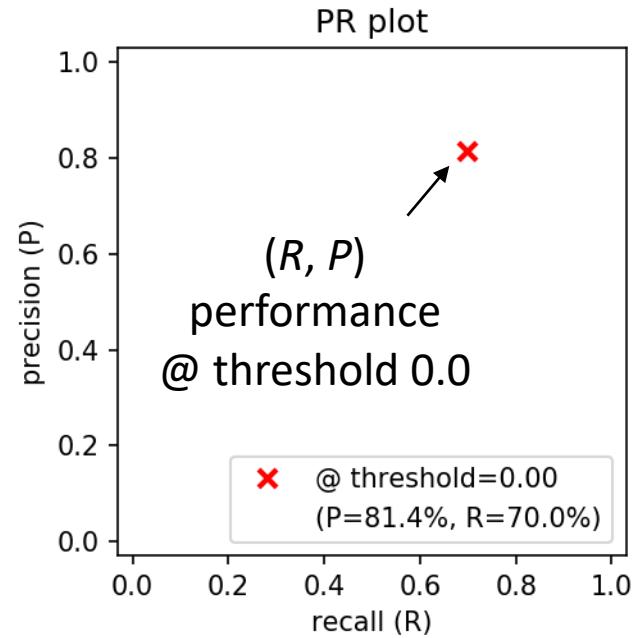
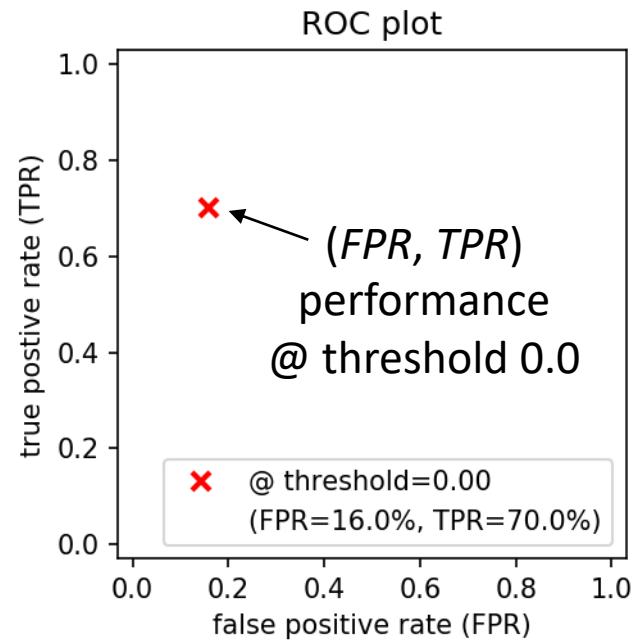
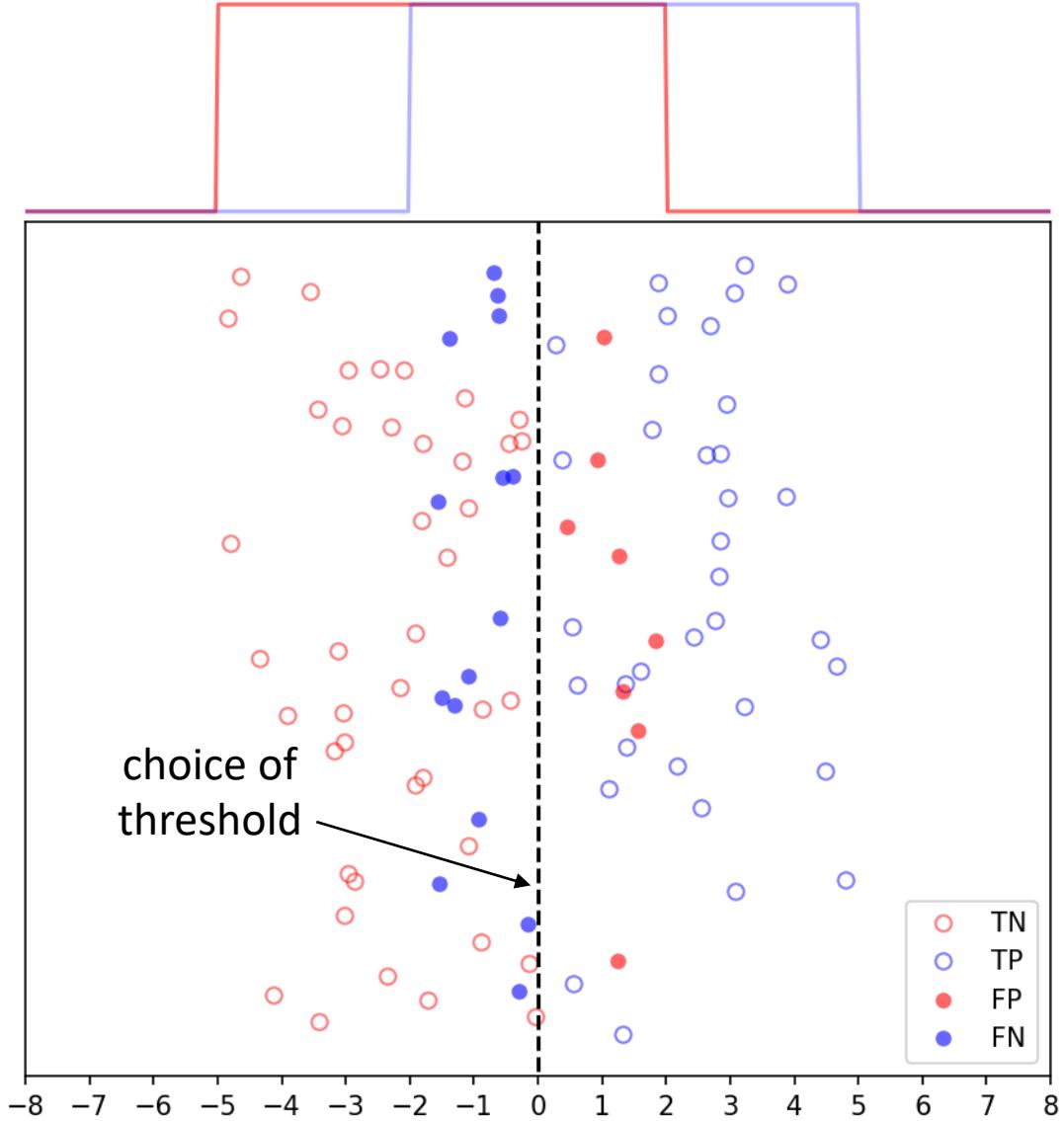
Precision and Recall

- Instead of computing (FPR , TPR) for a classifier we can compute (R , P) instead; same as (TPR , P)
- Perfect performance would be at point at (1, 1)
 - $R = 1$: all known positives are predicted as positive
 - $P = 1$: all predicted positives are known to be positive
- Example: score documents by relevance, threshold as “relevant” (positive) or “irrelevant” (negative)
 - R = fraction of *all relevant documents that were selected*
 - P = fraction of *selected documents were actually relevant*
- Again, different thresholds will give different (R , P)
 - threshold @ $-\infty$ (predict all positive) gives $(1, \frac{N_p}{N})$
 - threshold @ $+\infty$ (predict all negative) gives $(0, \frac{0}{0})$

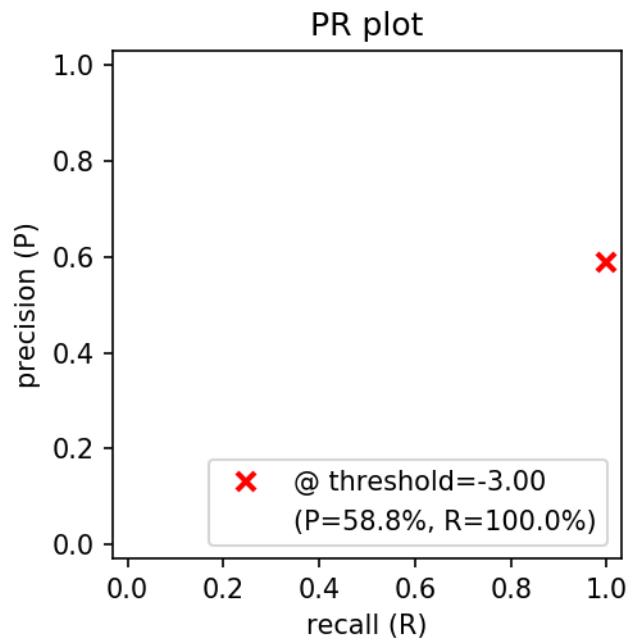
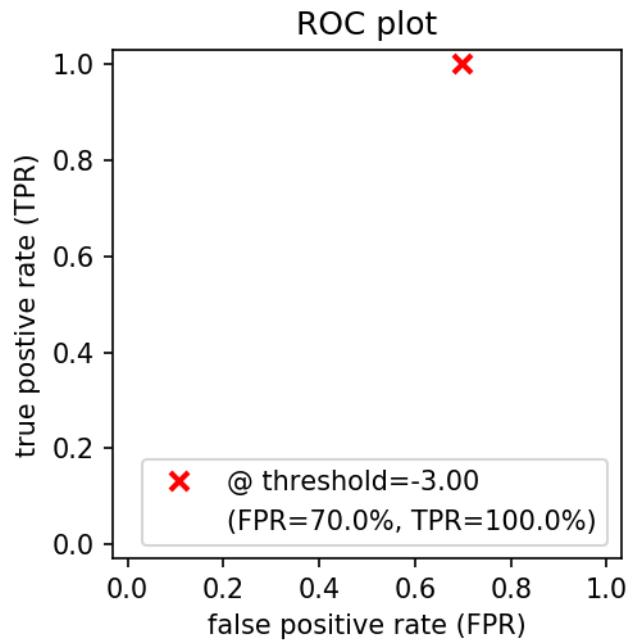
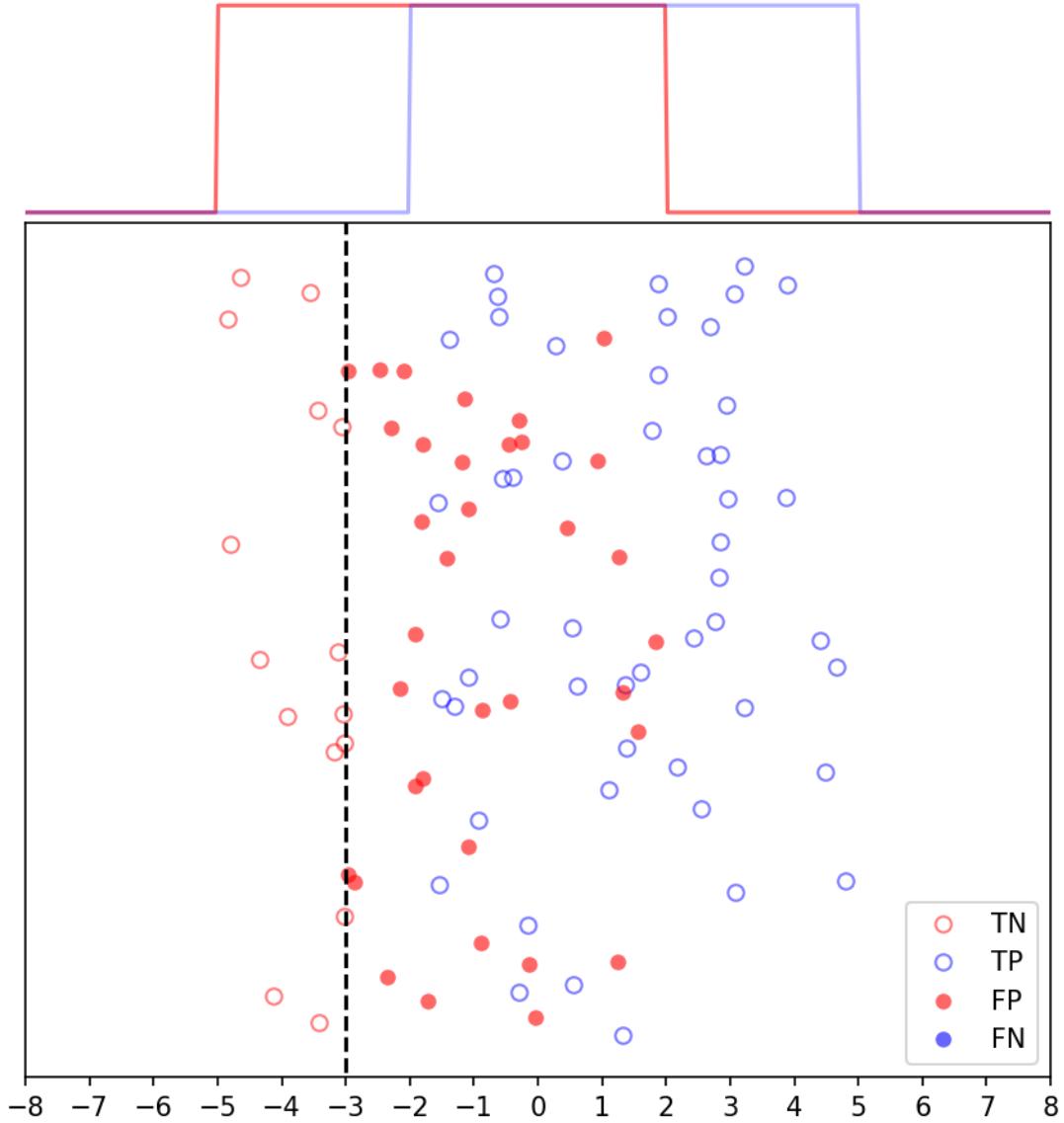
Example of ROC & PR



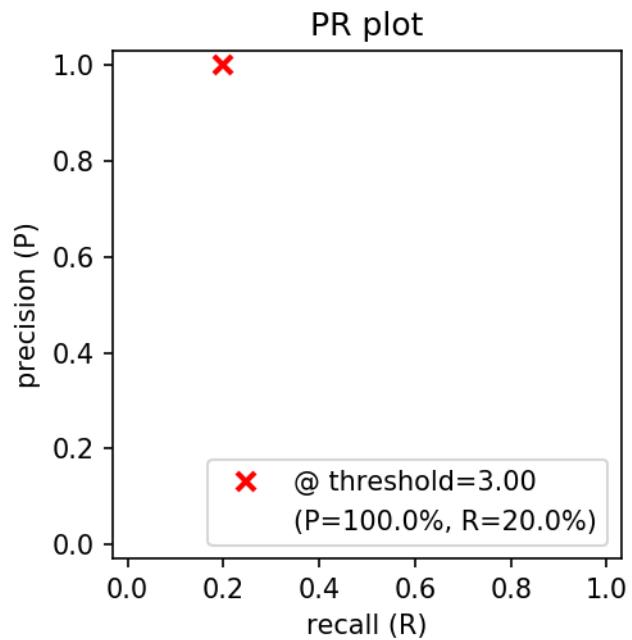
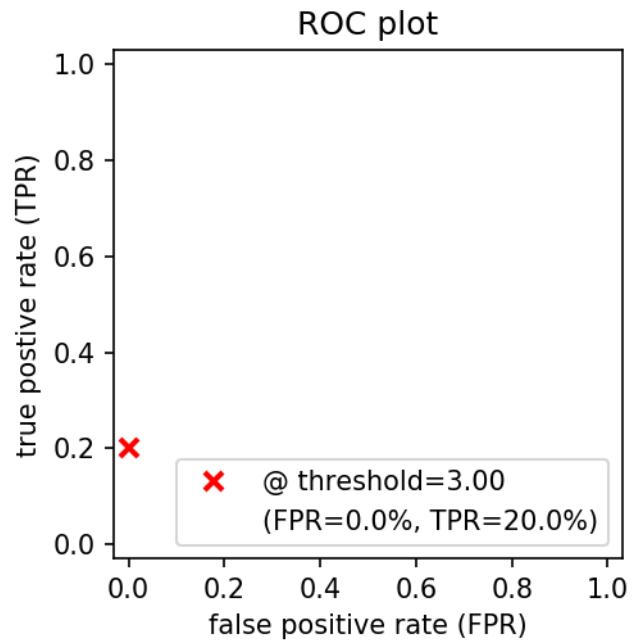
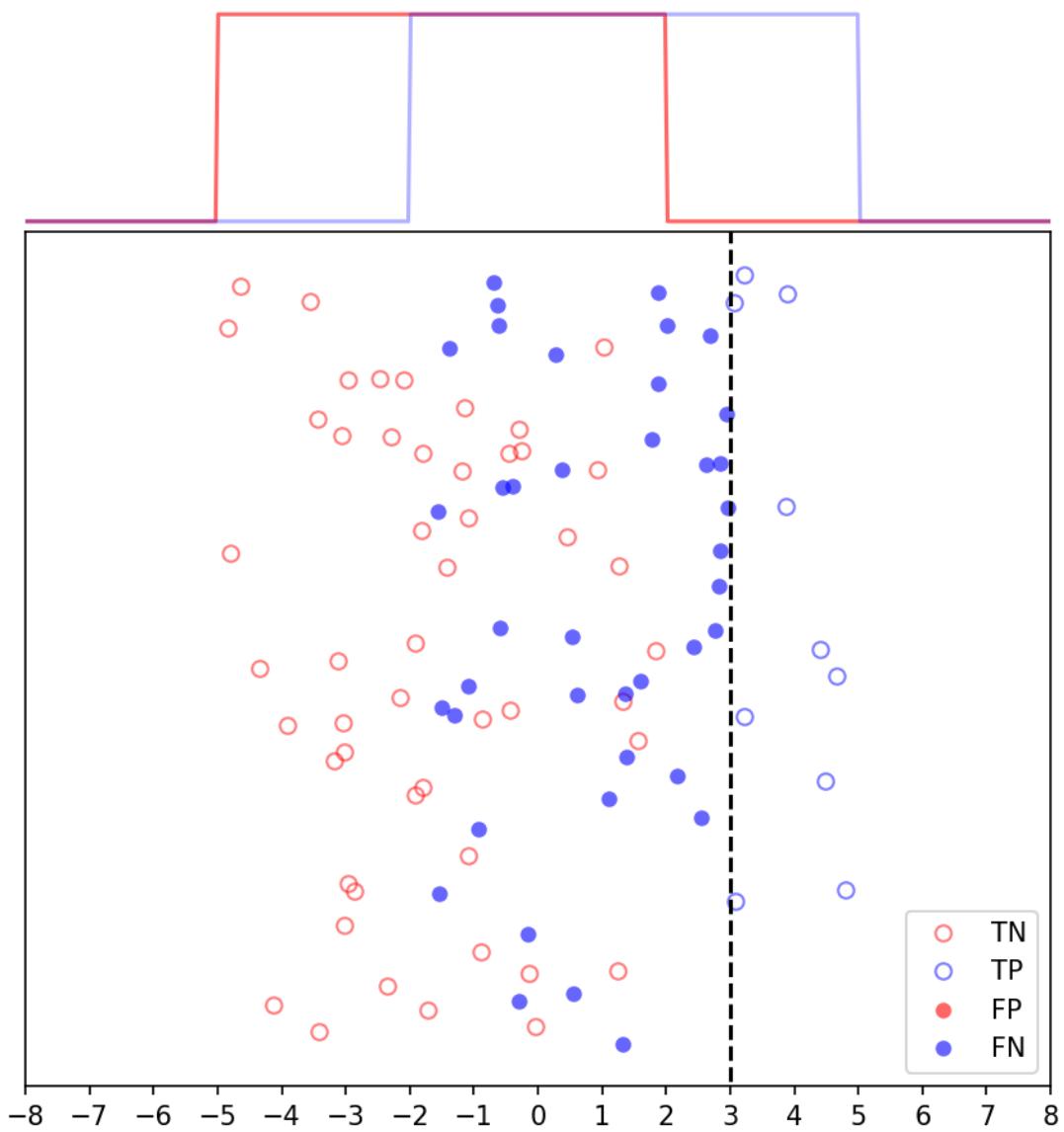
Example of ROC & PR



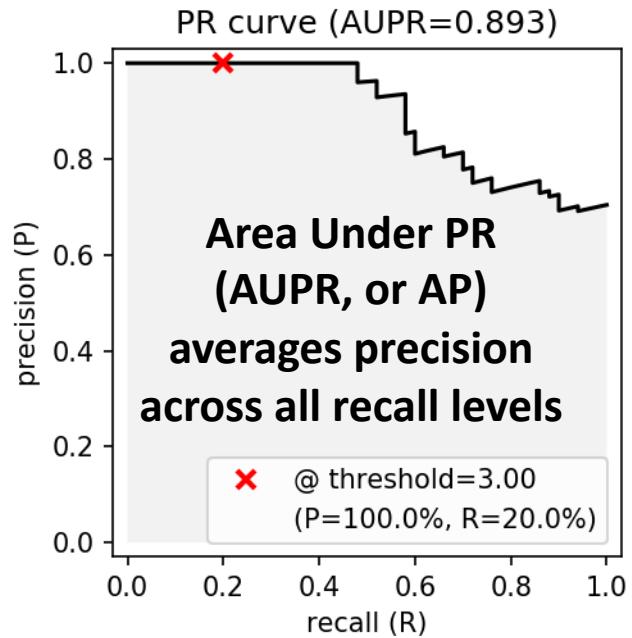
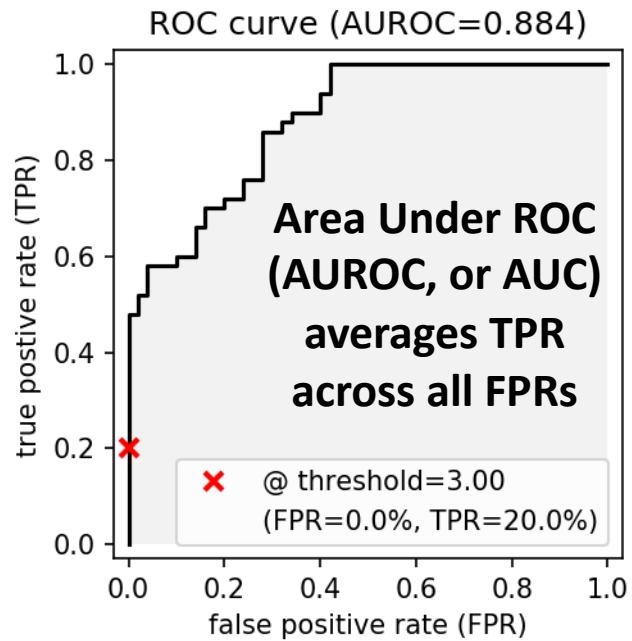
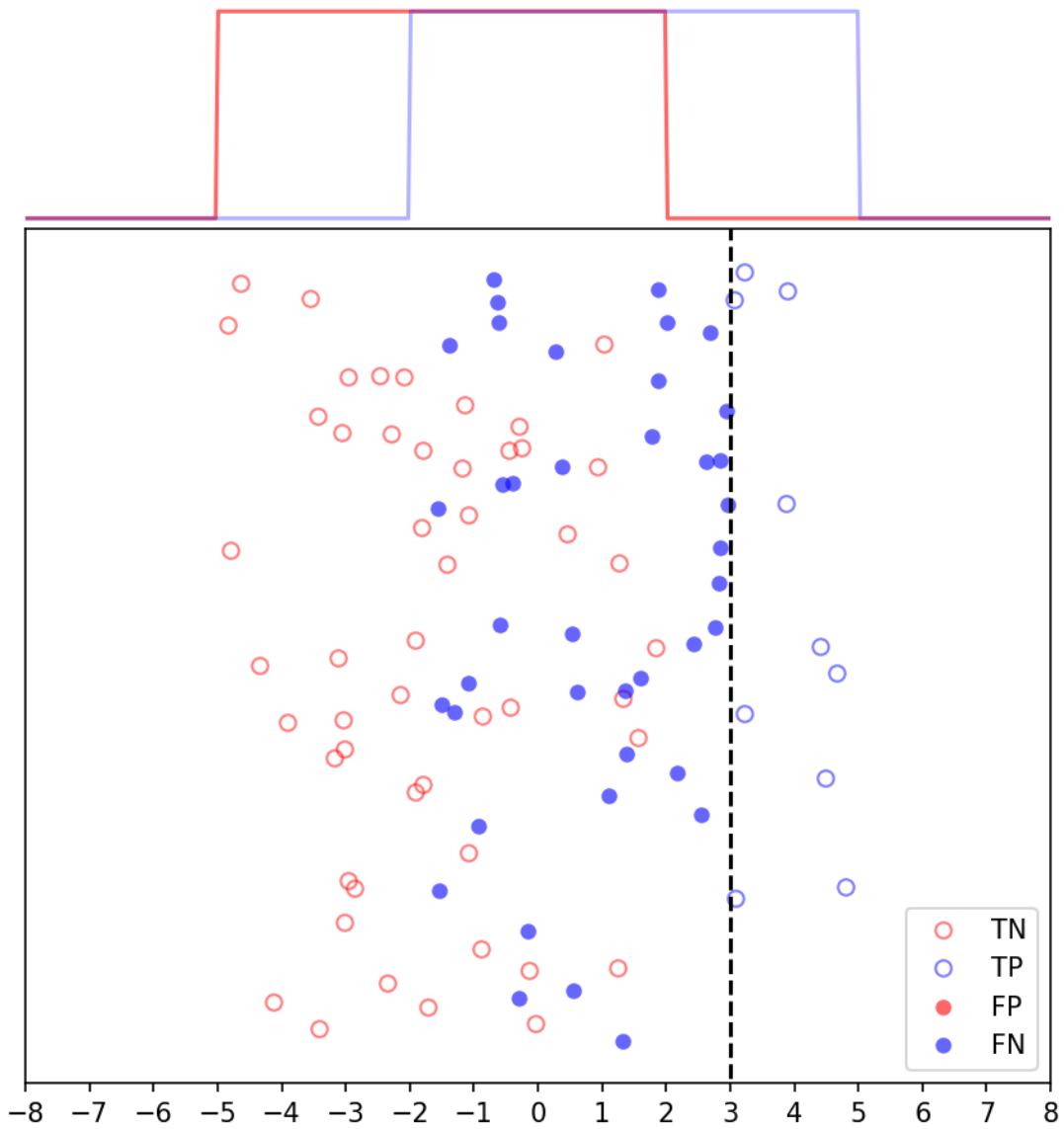
Example of ROC & PR



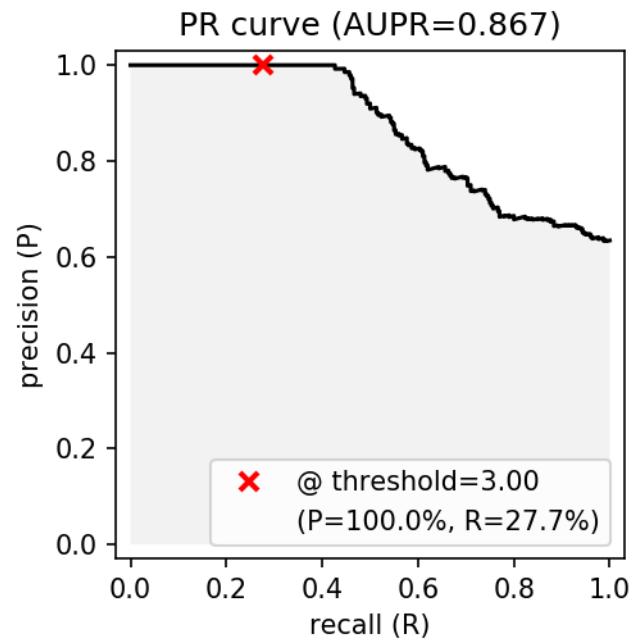
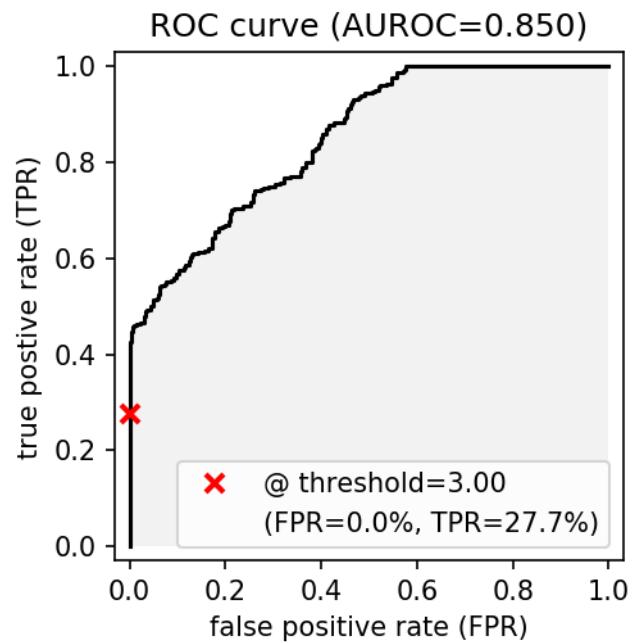
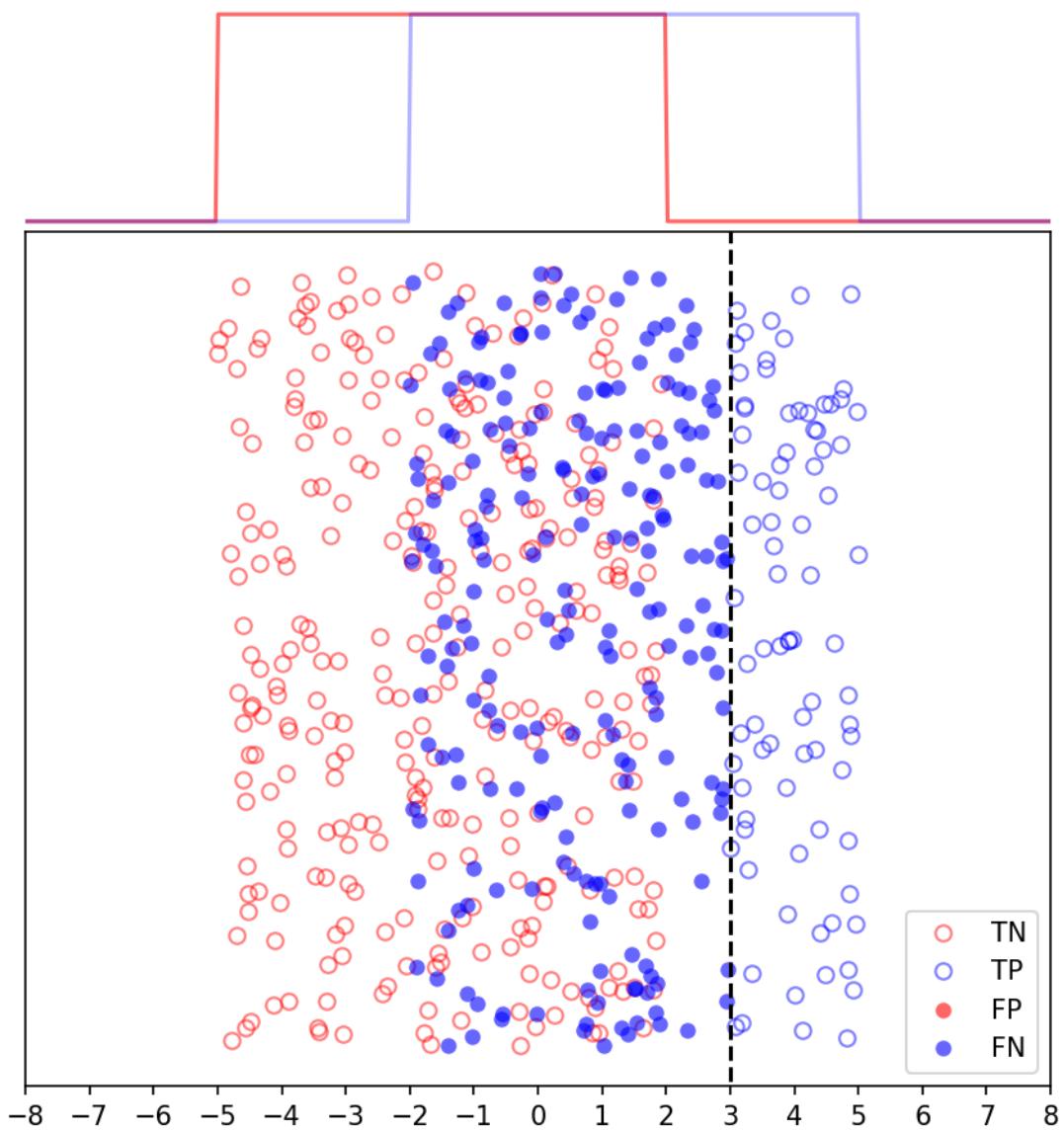
Example of ROC & PR



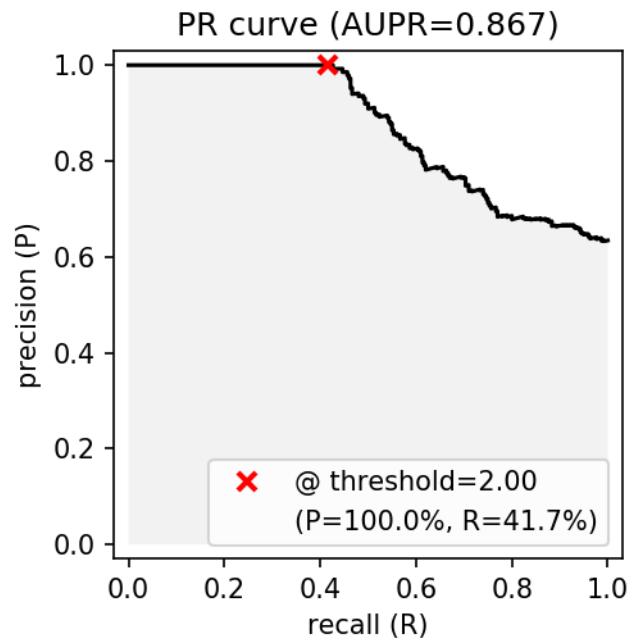
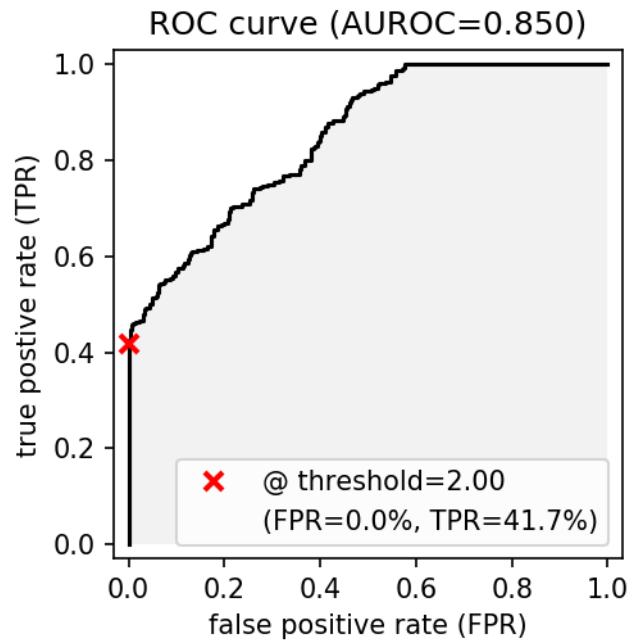
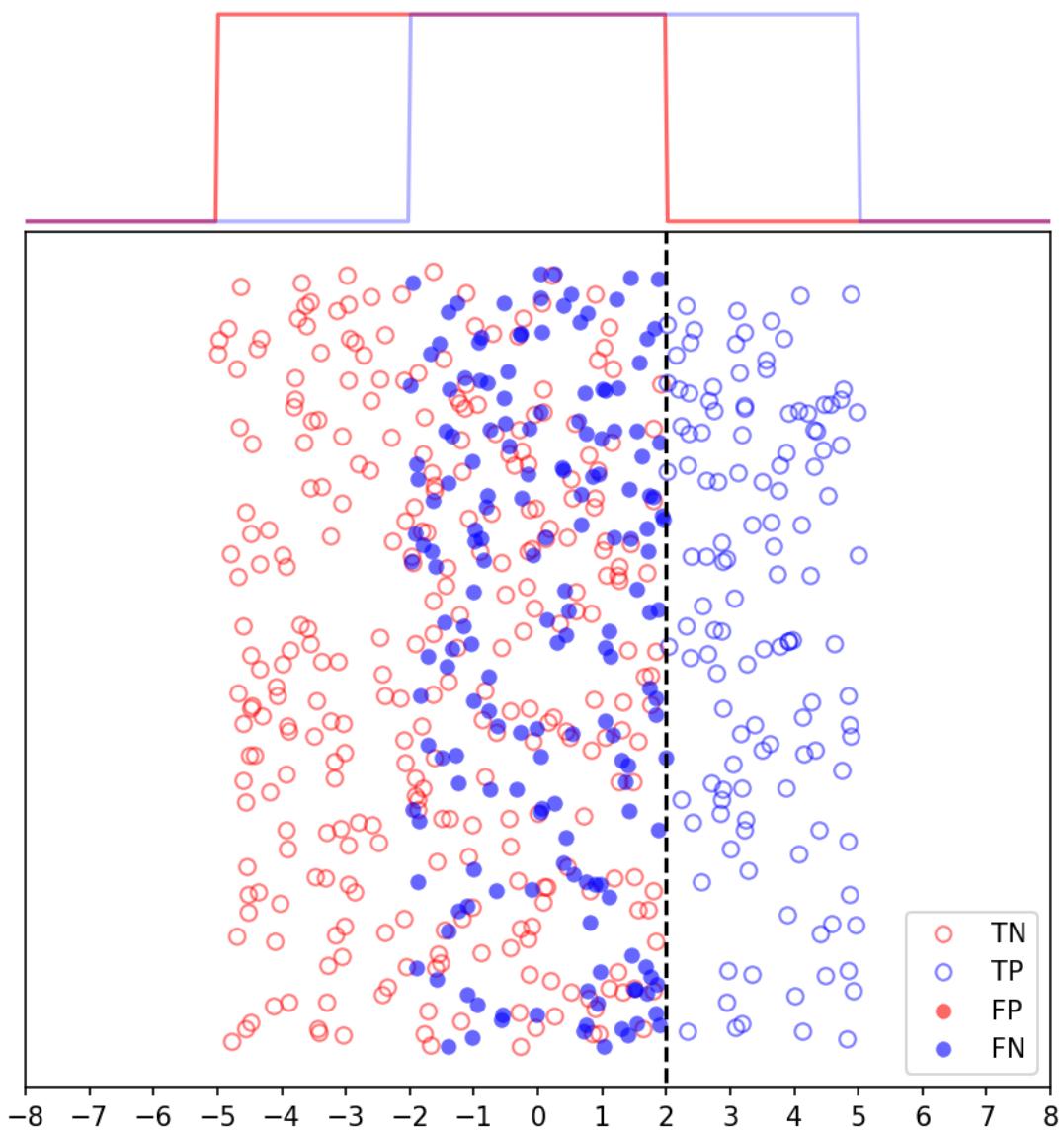
Example of ROC & PR



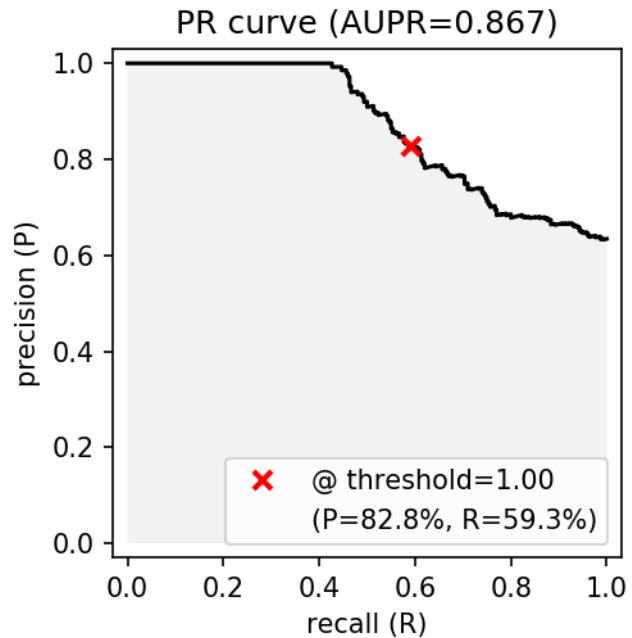
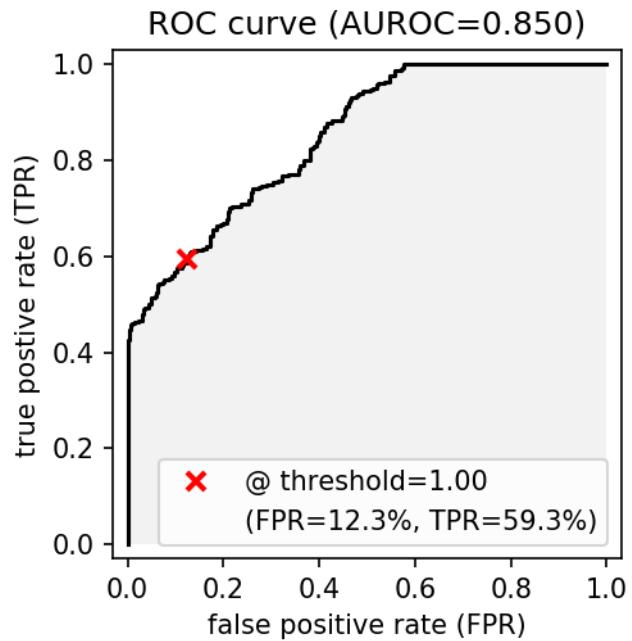
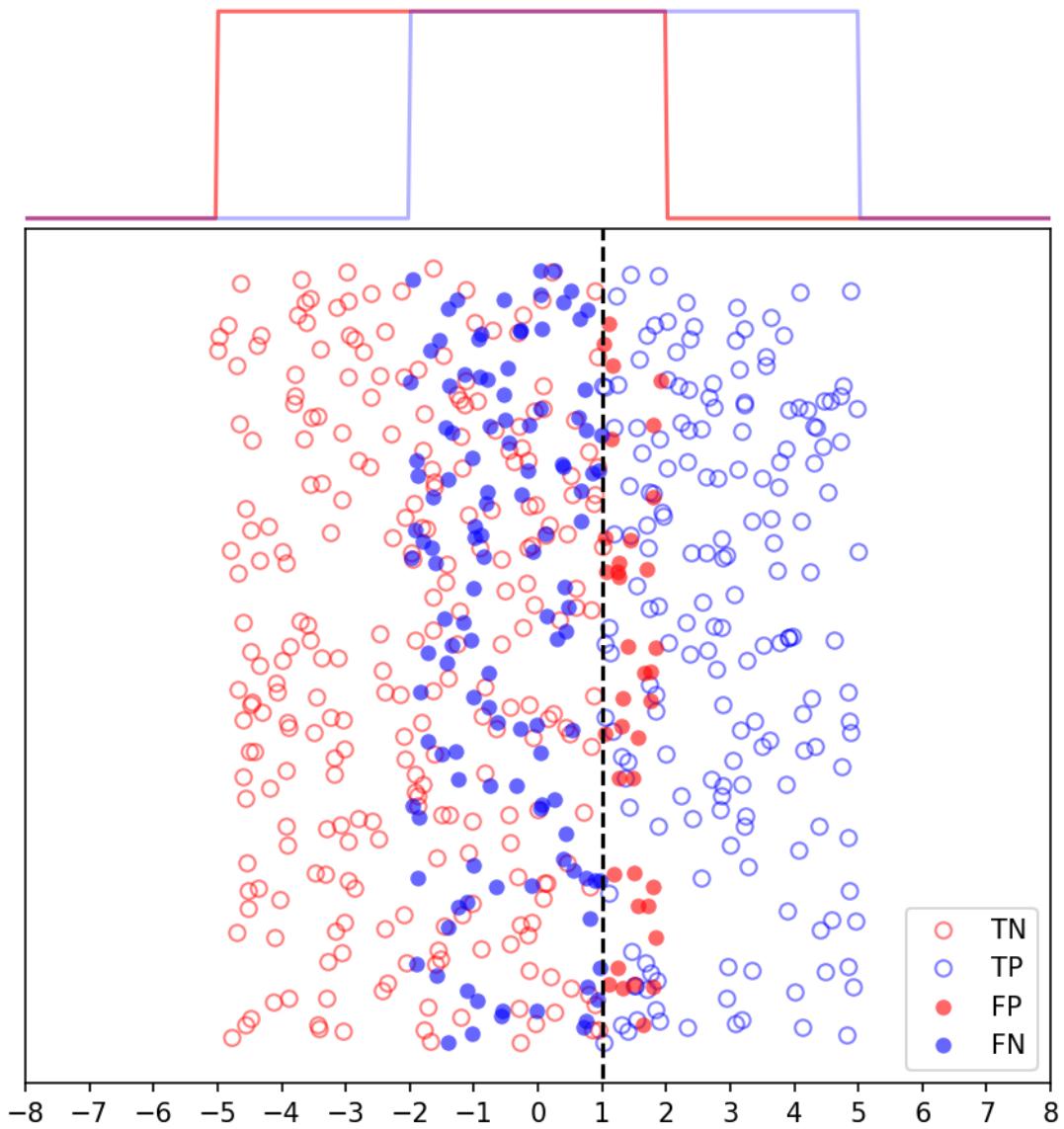
Example of ROC & PR



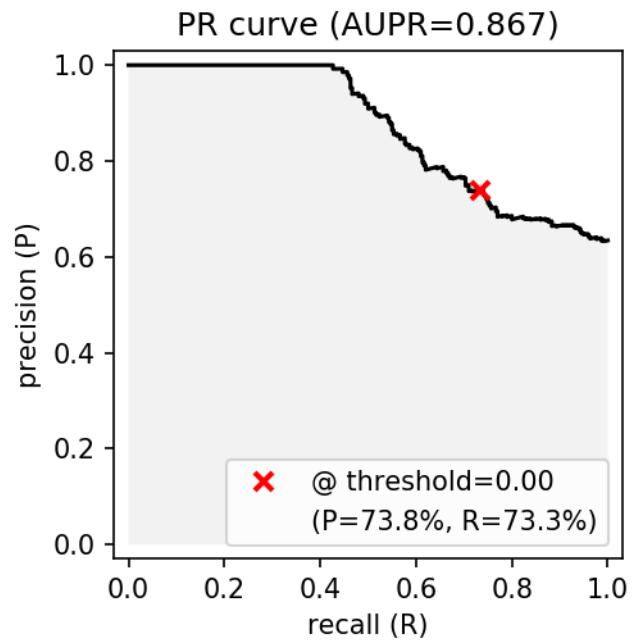
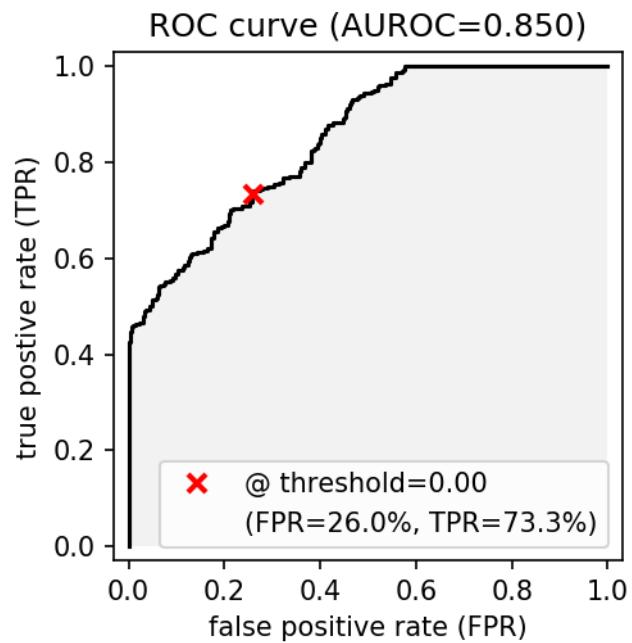
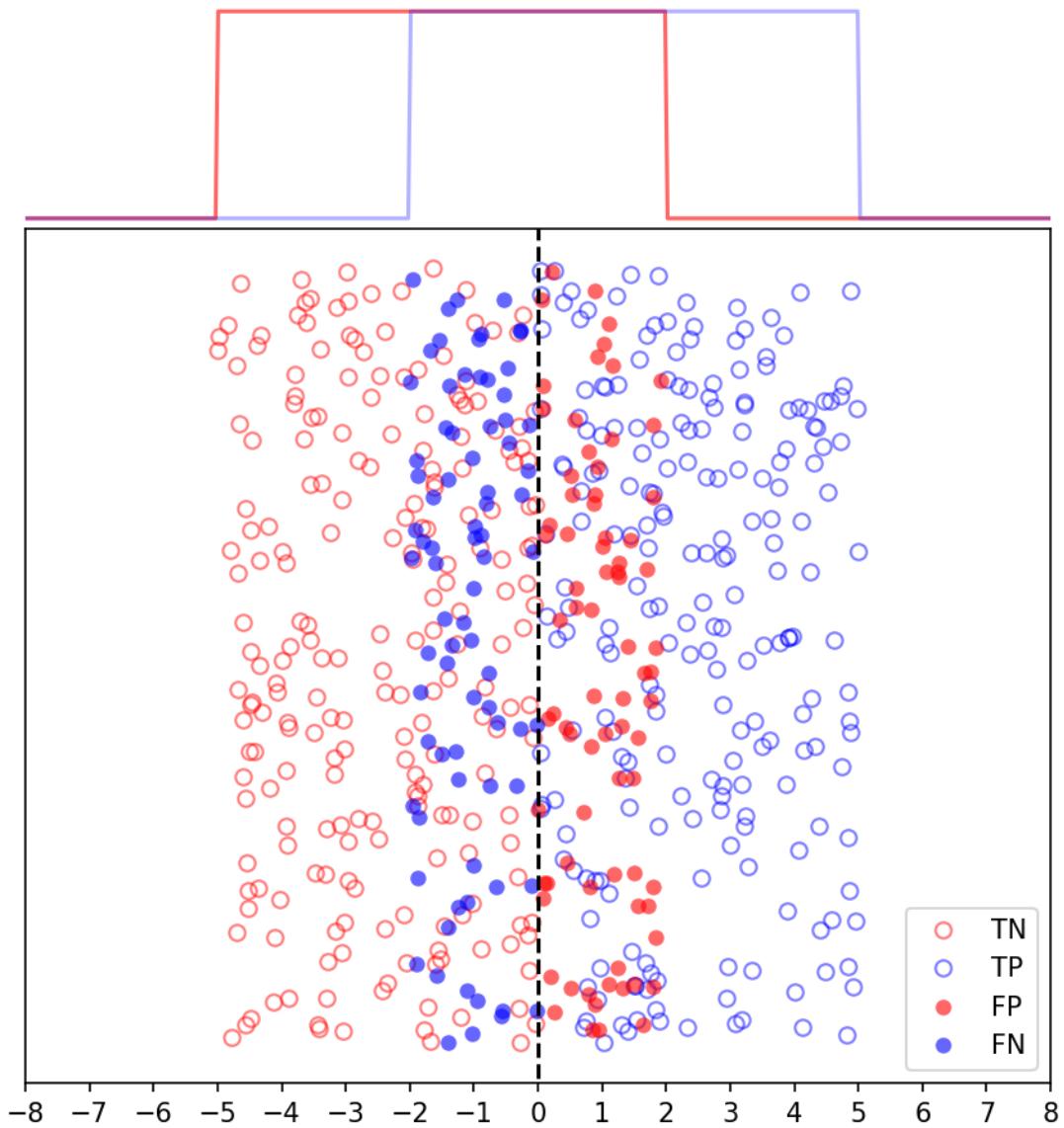
Example of ROC & PR



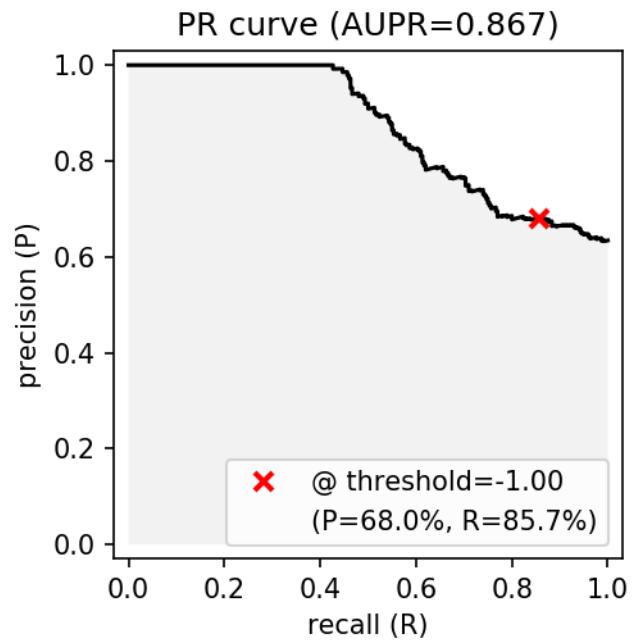
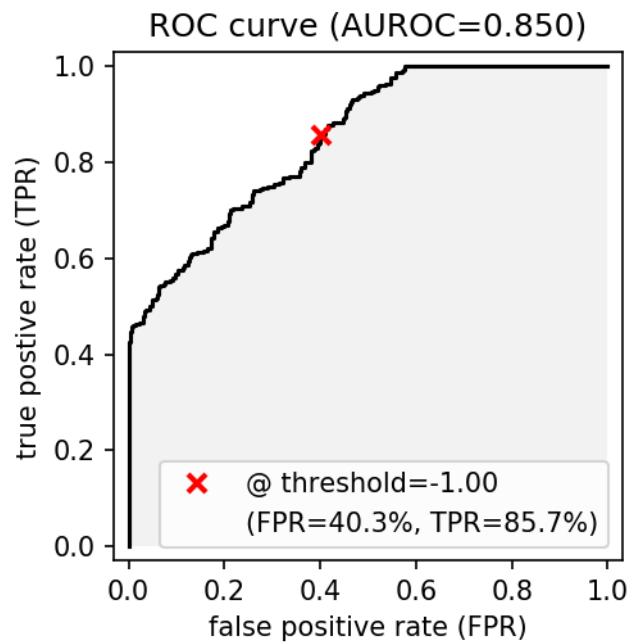
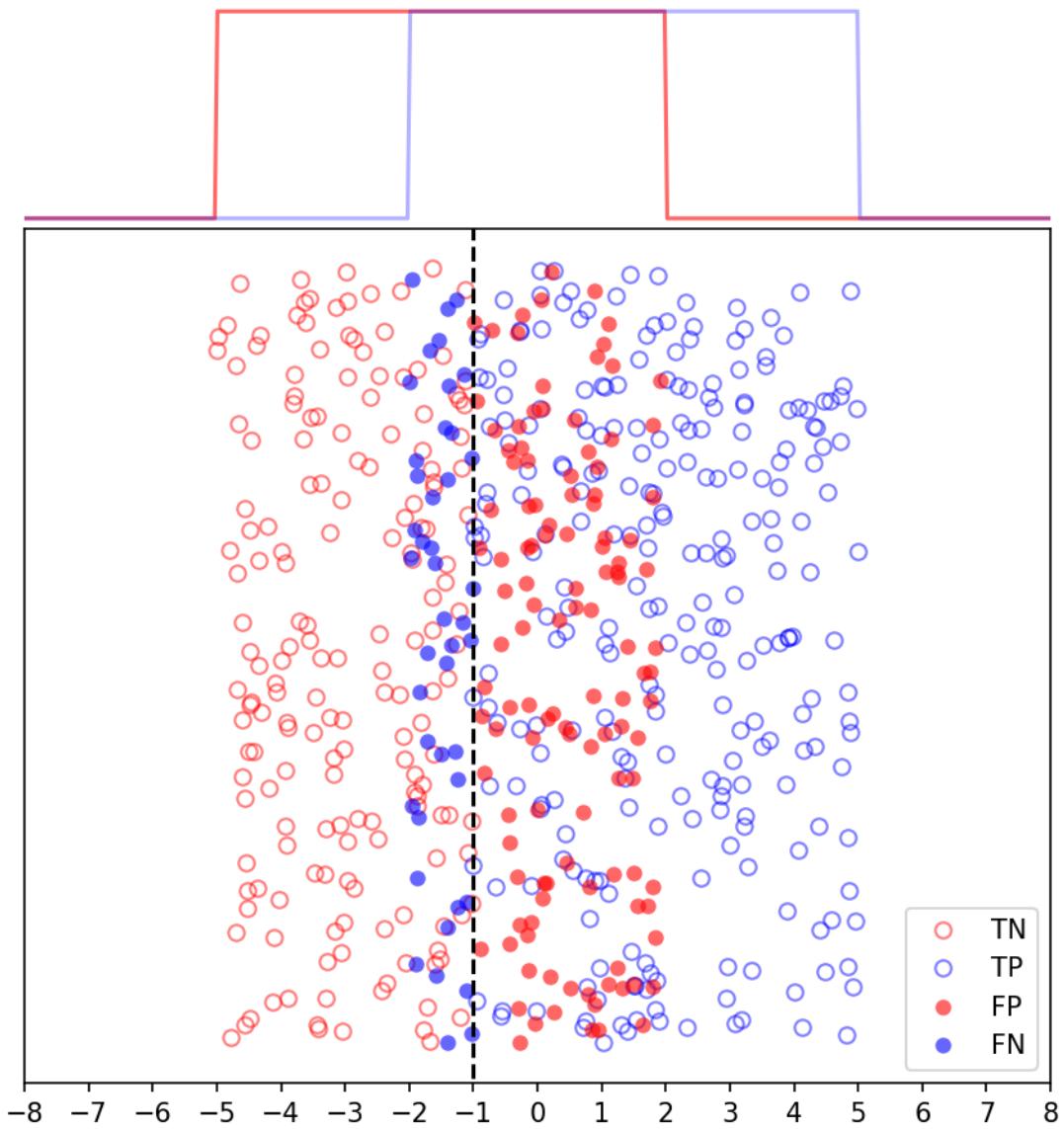
Example of ROC & PR



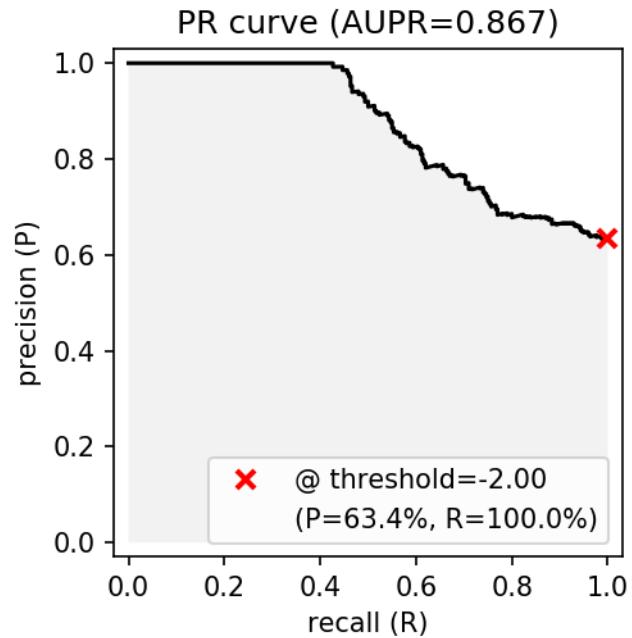
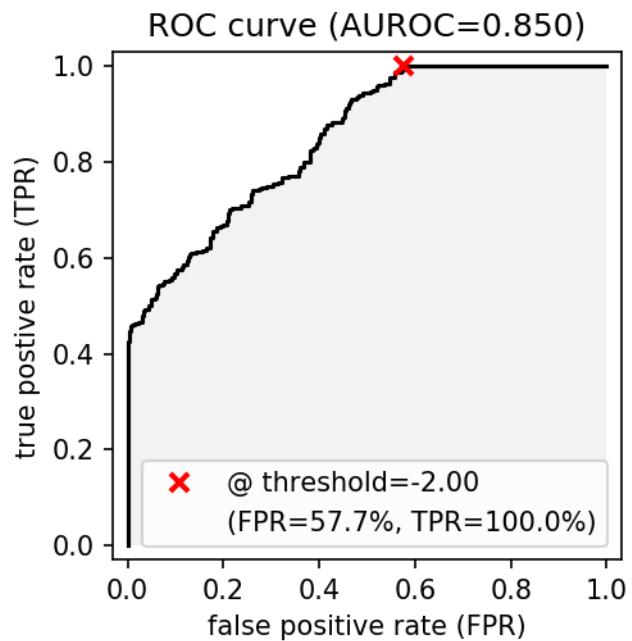
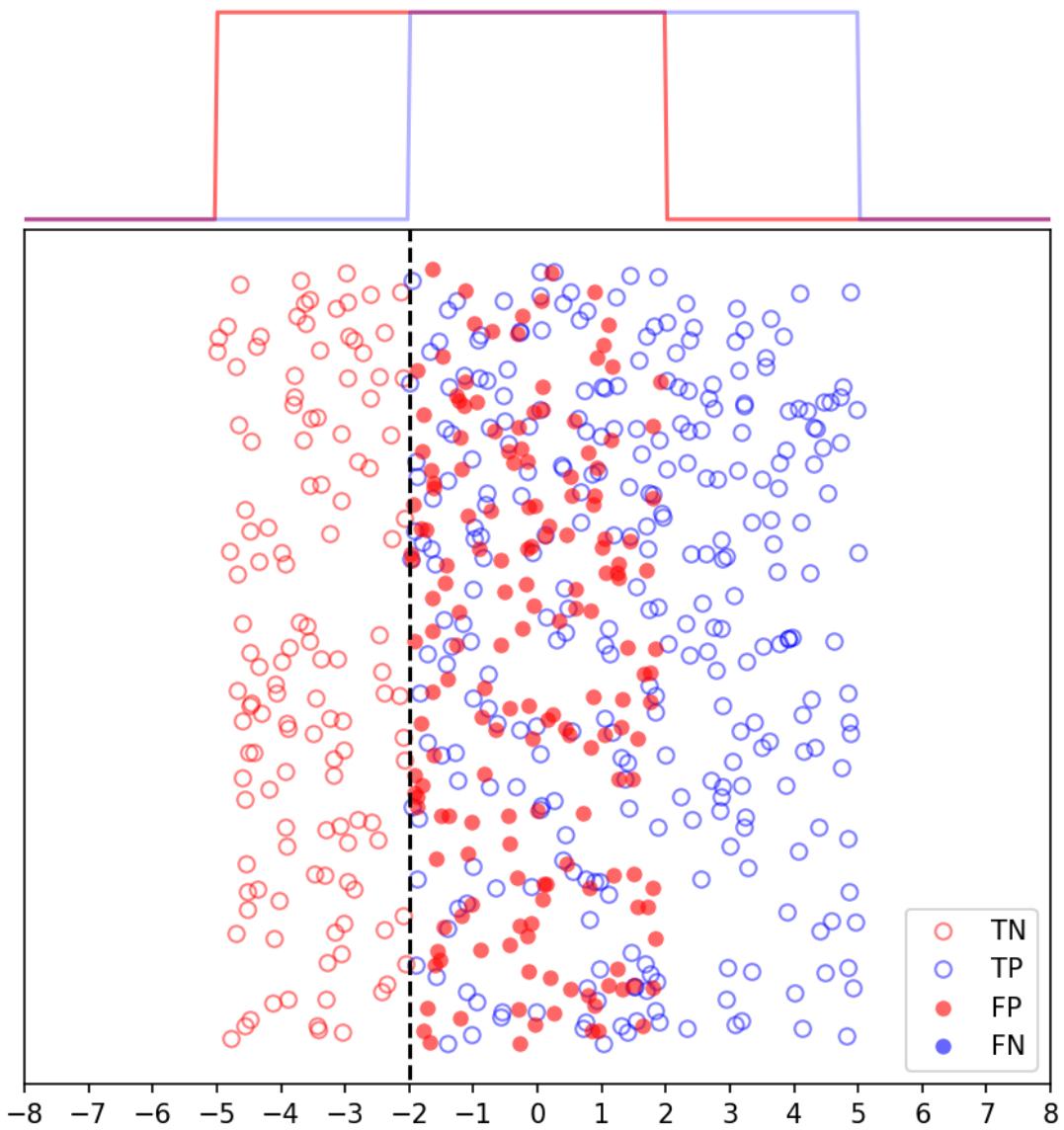
Example of ROC & PR



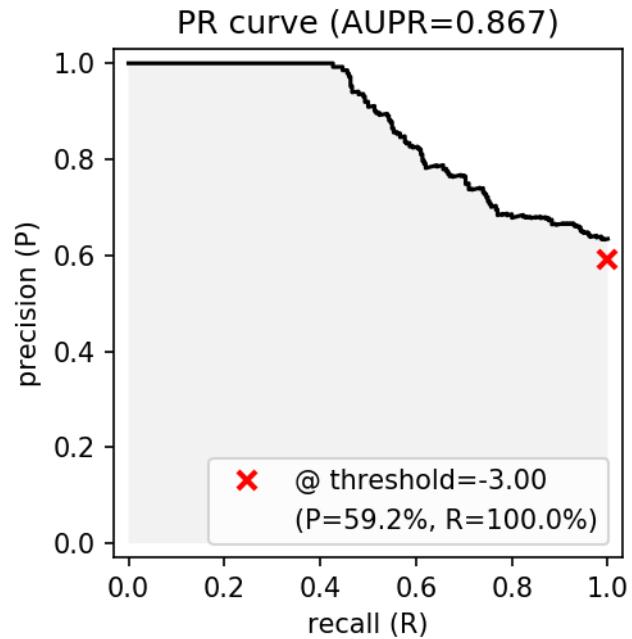
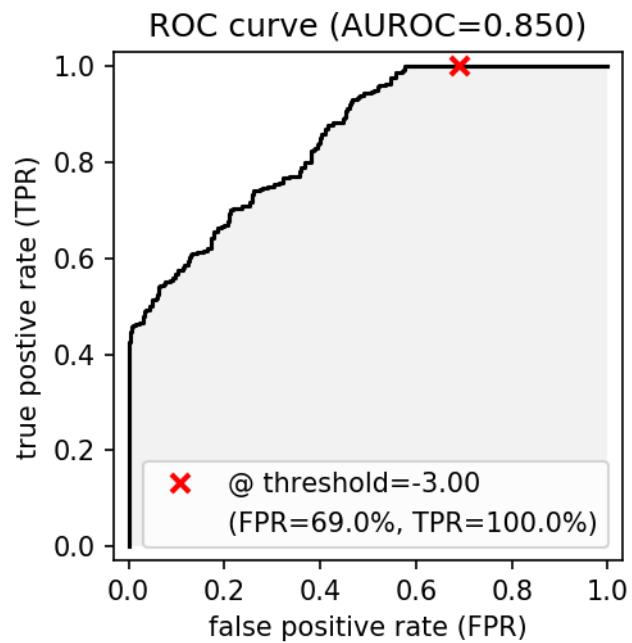
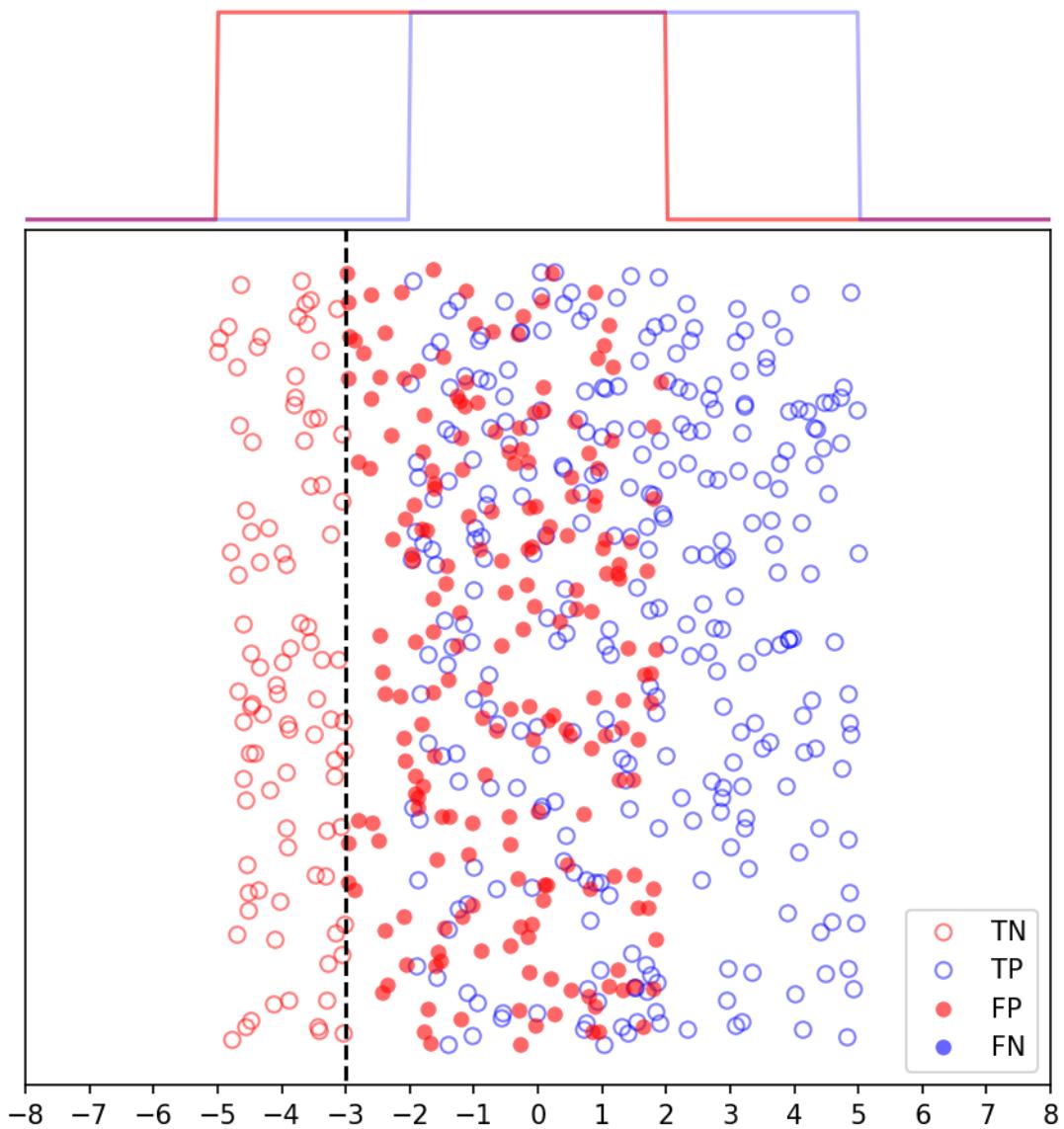
Example of ROC & PR



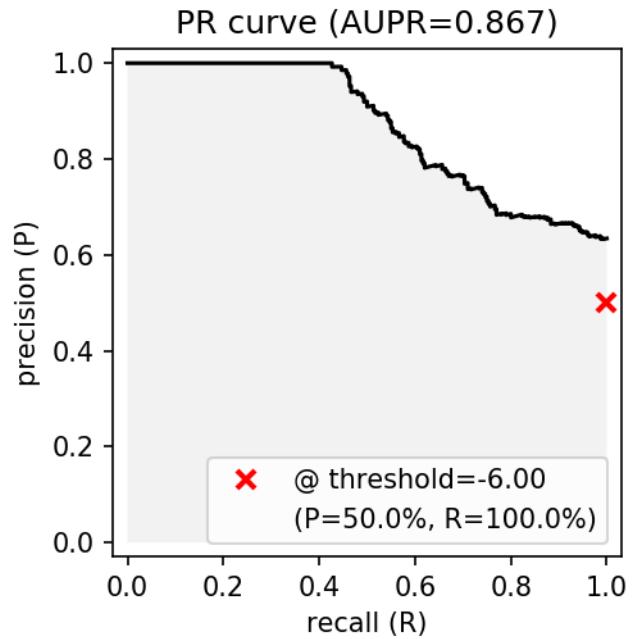
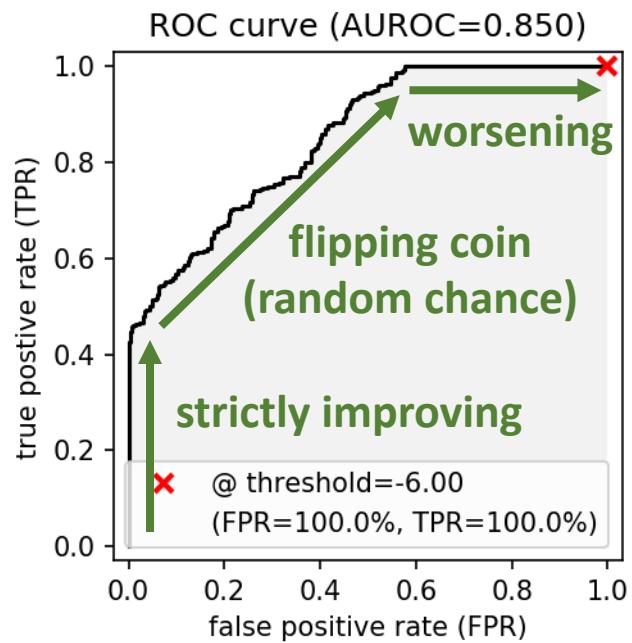
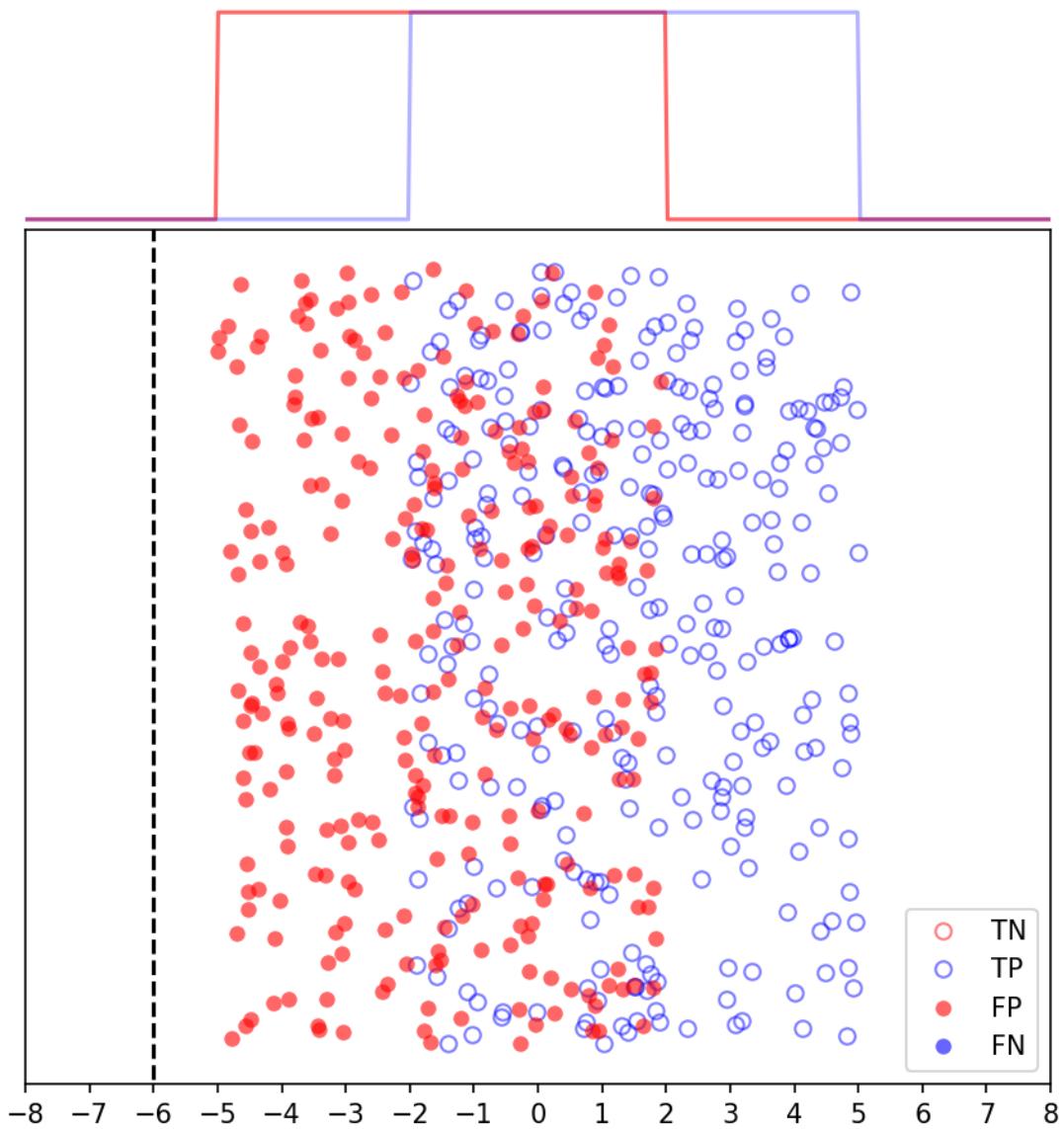
Example of ROC & PR



Example of ROC & PR

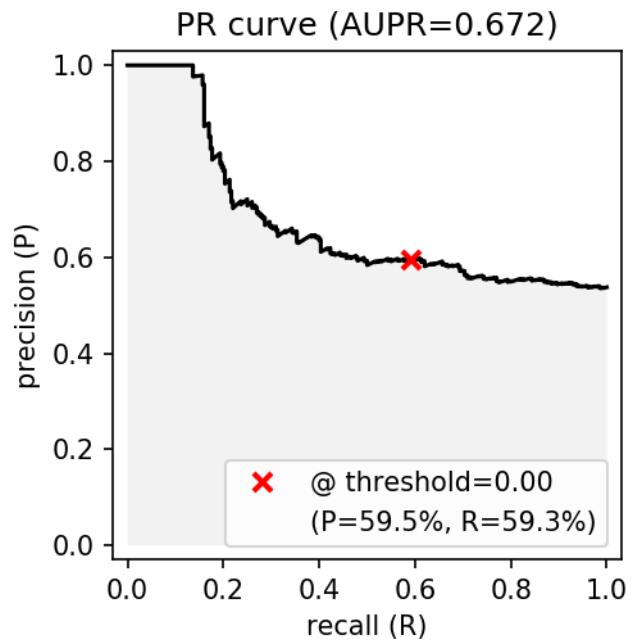
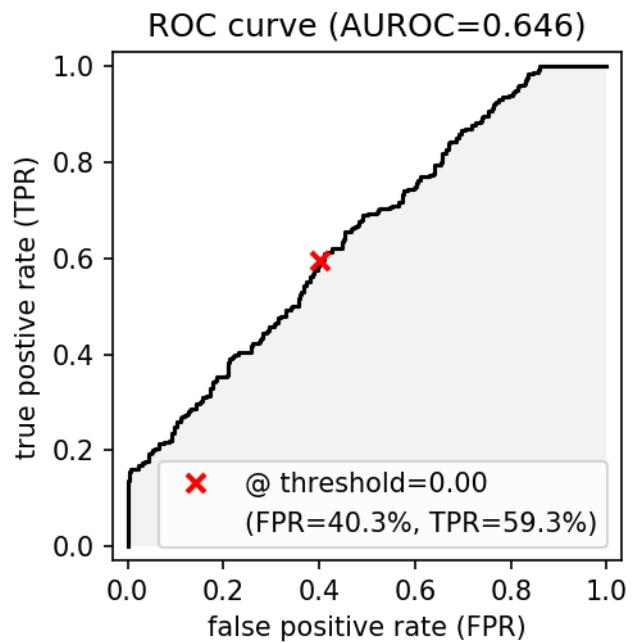
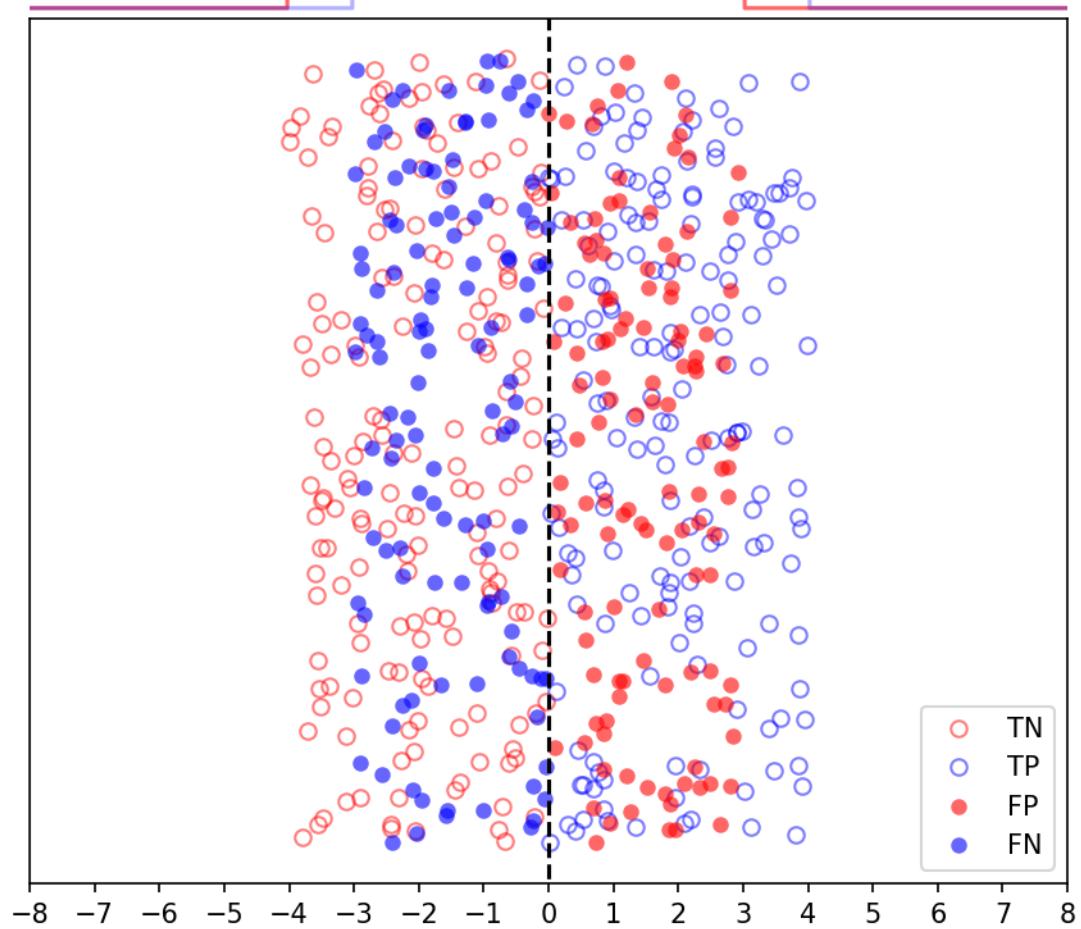


Example of ROC & PR

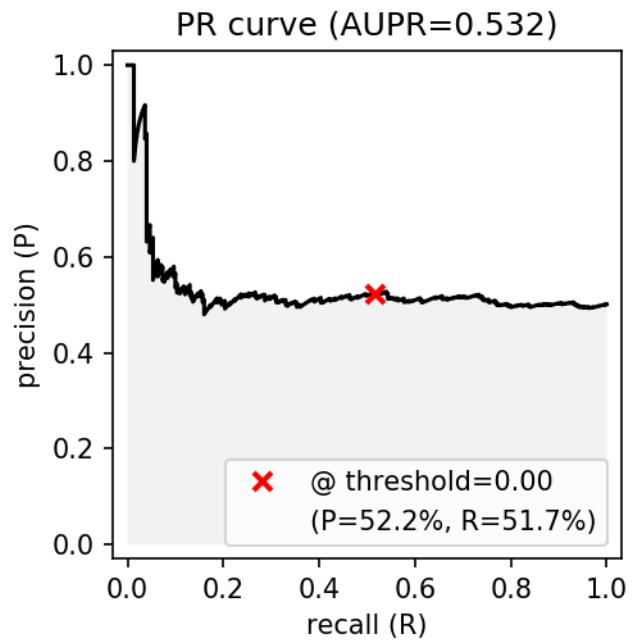
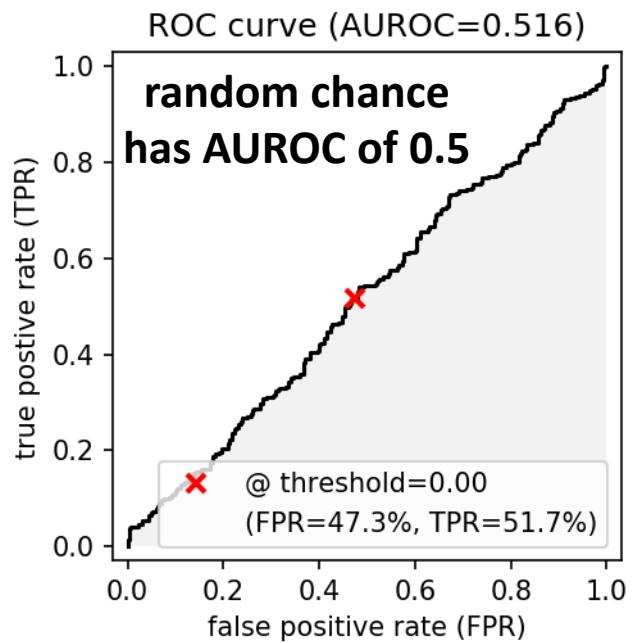
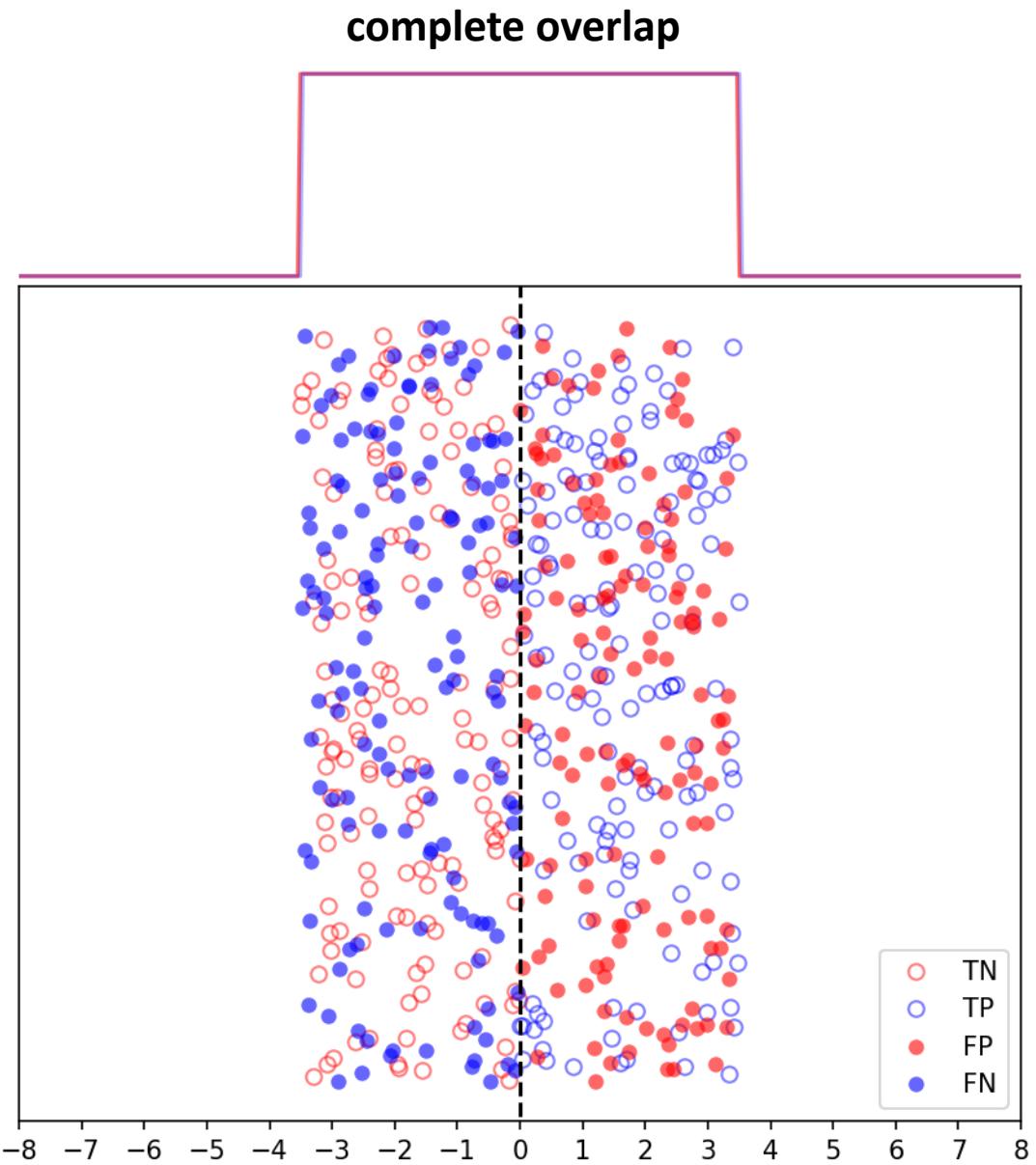


Example of ROC & PR

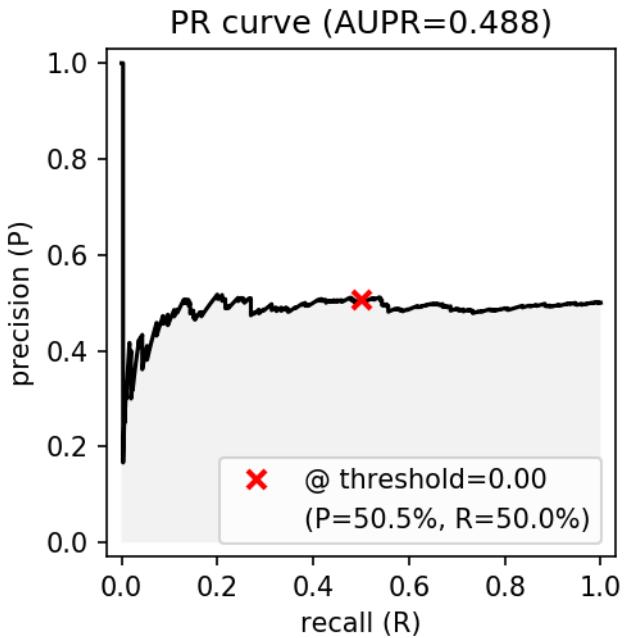
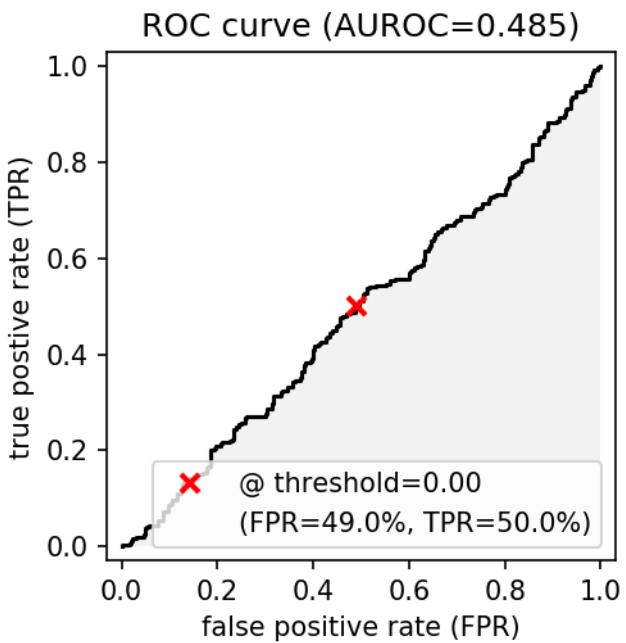
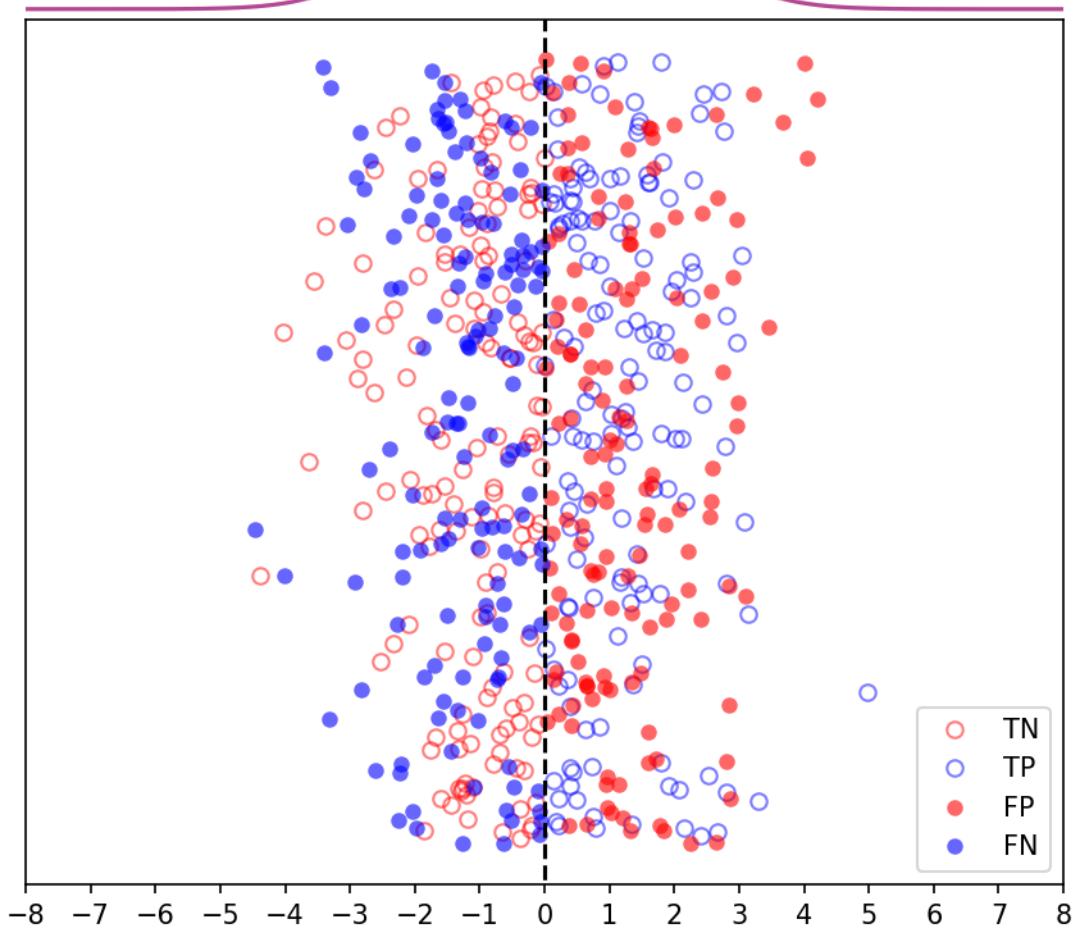
more overlap



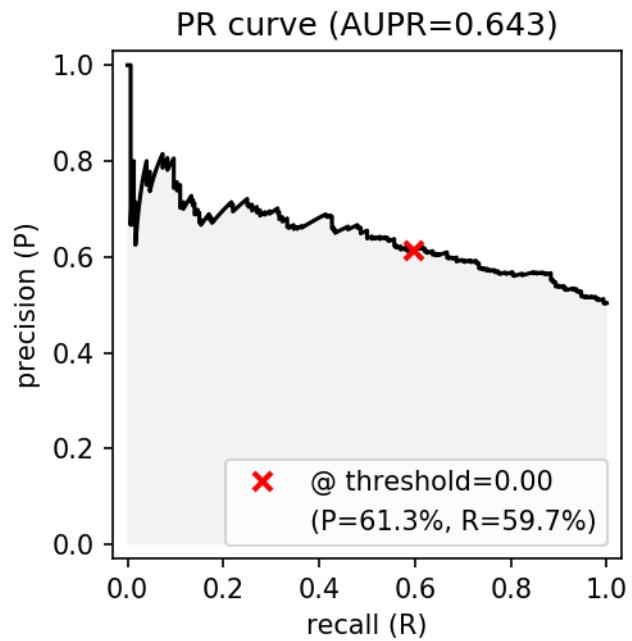
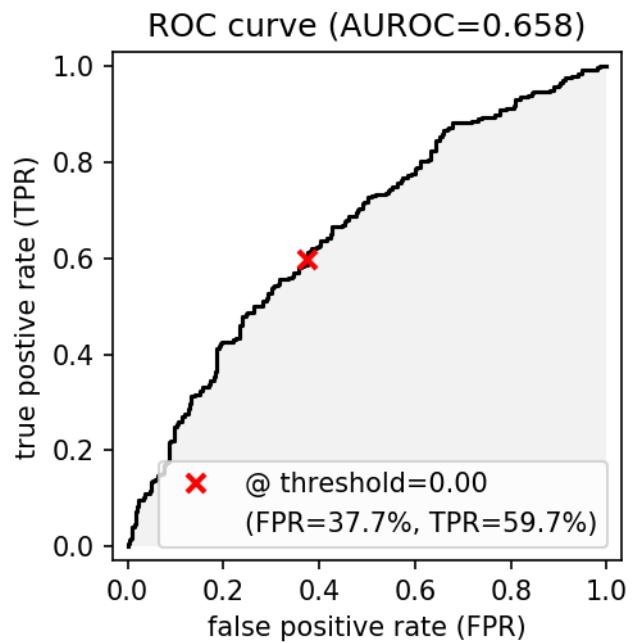
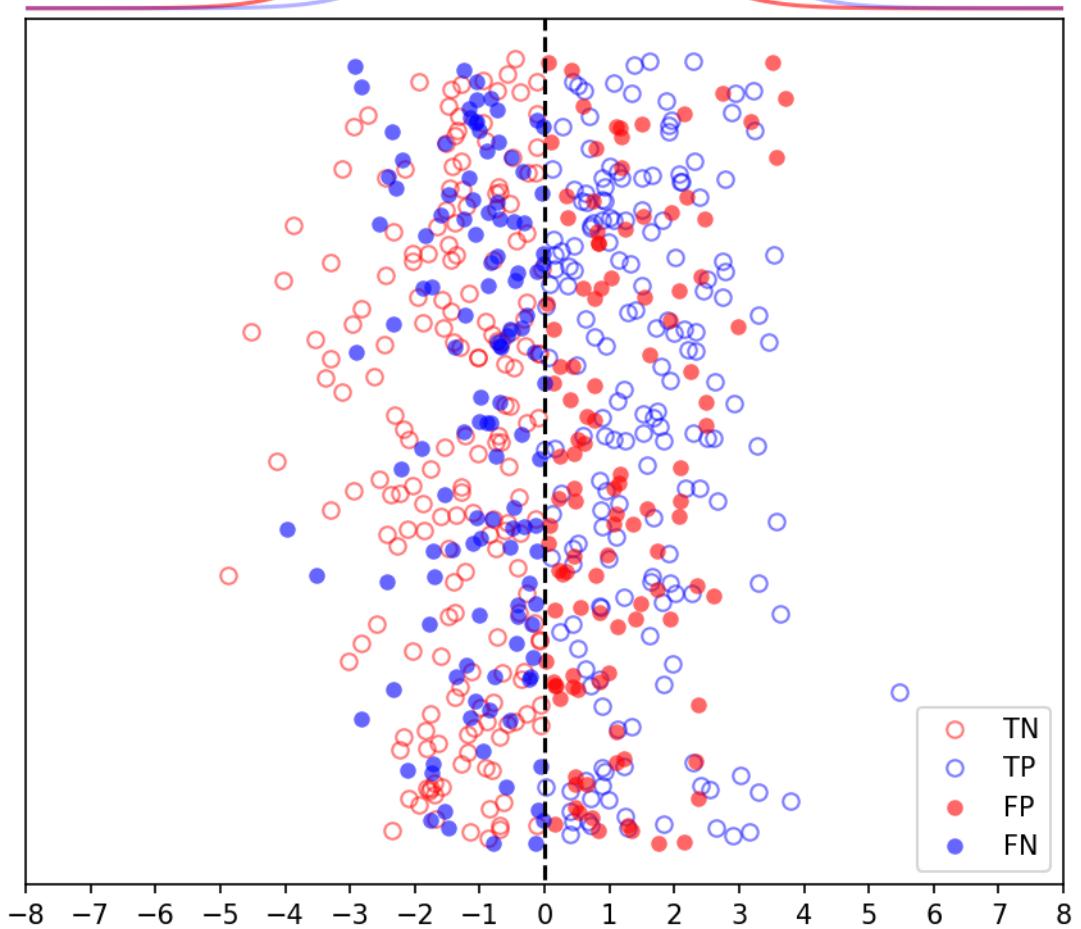
Example of ROC & PR



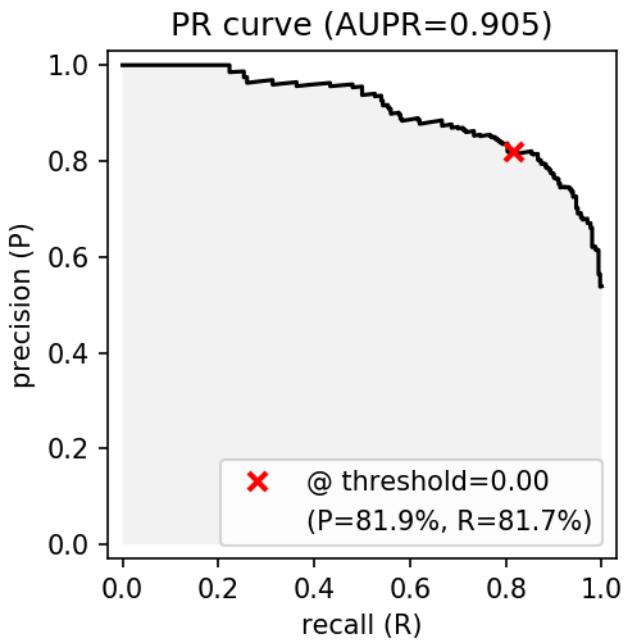
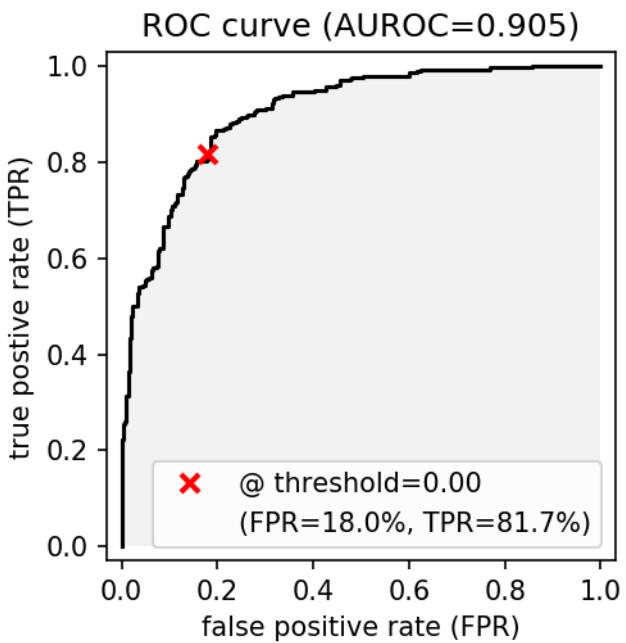
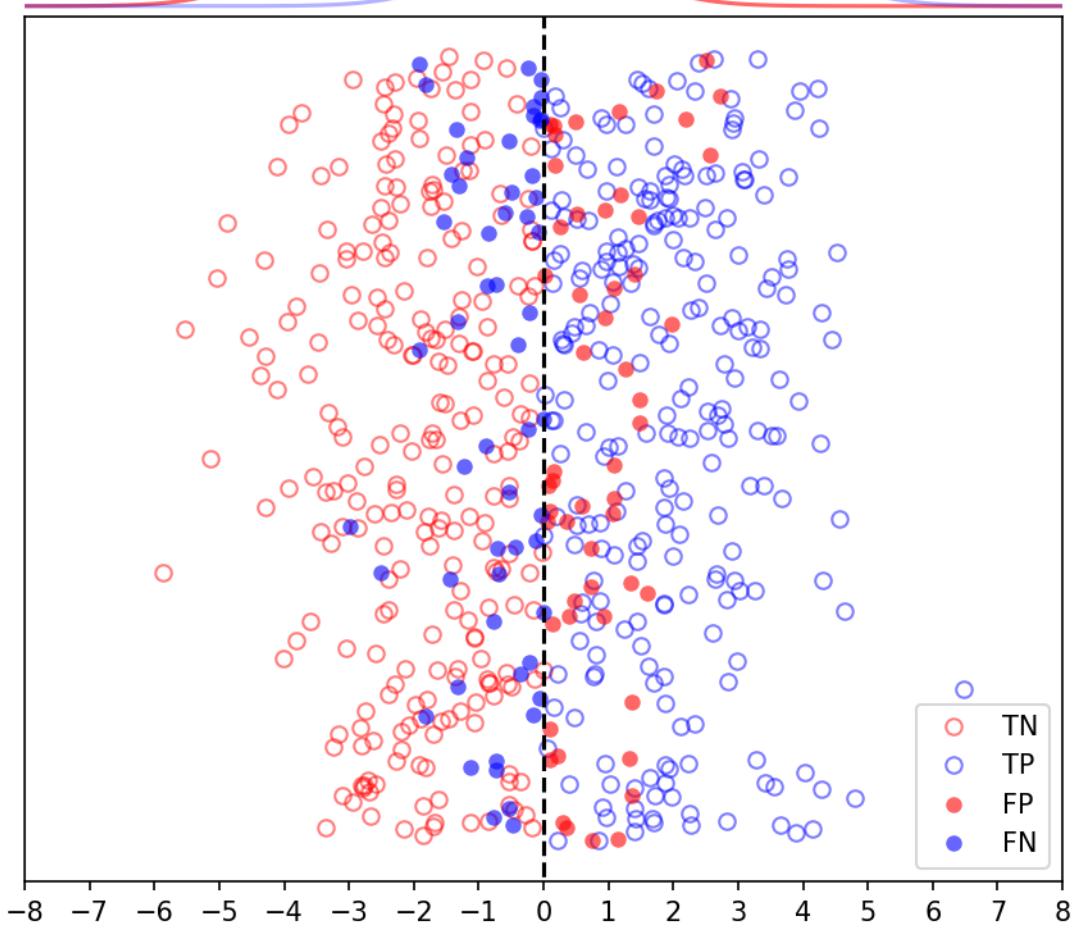
Example of ROC & PR



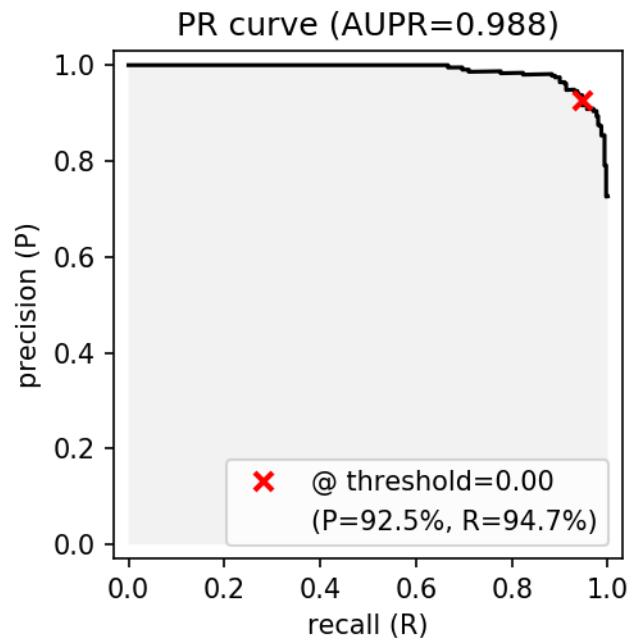
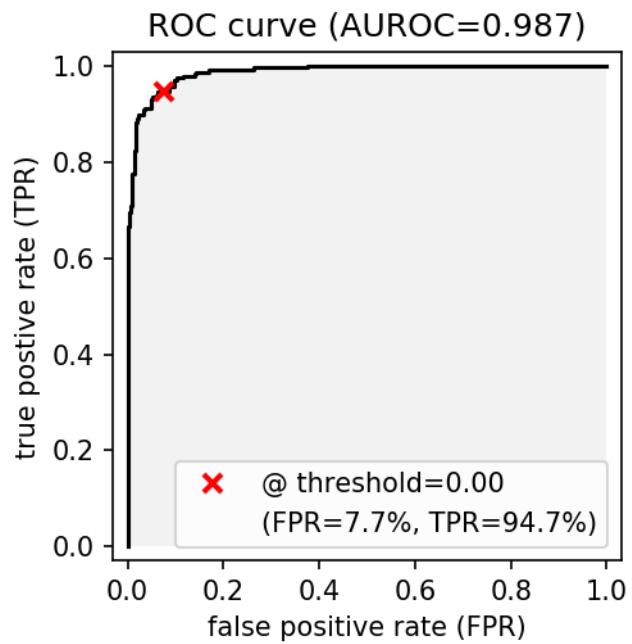
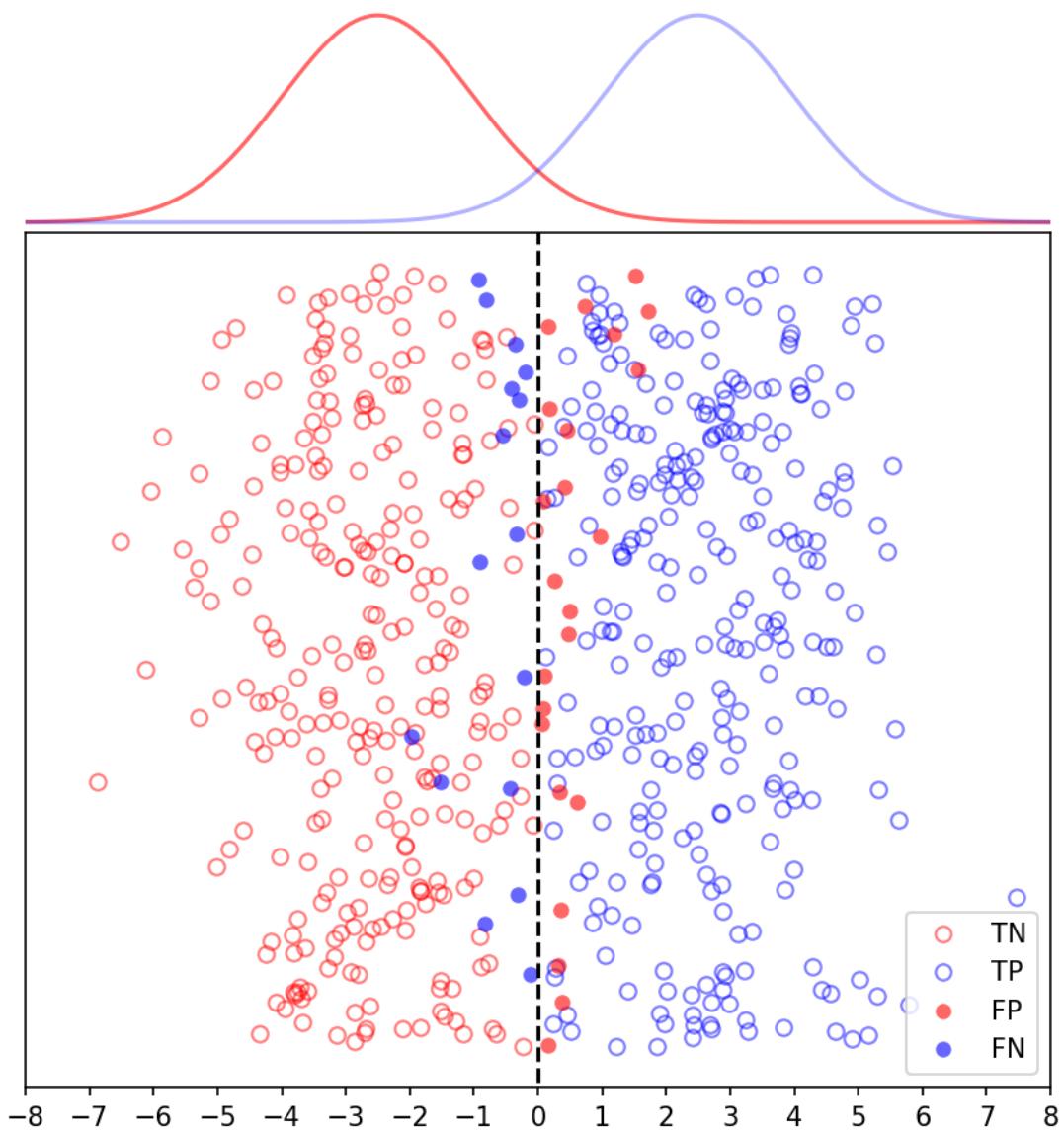
Example of ROC & PR



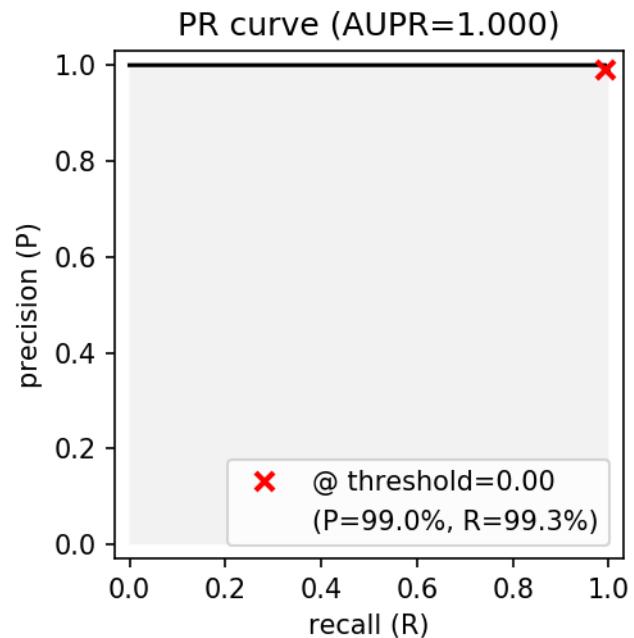
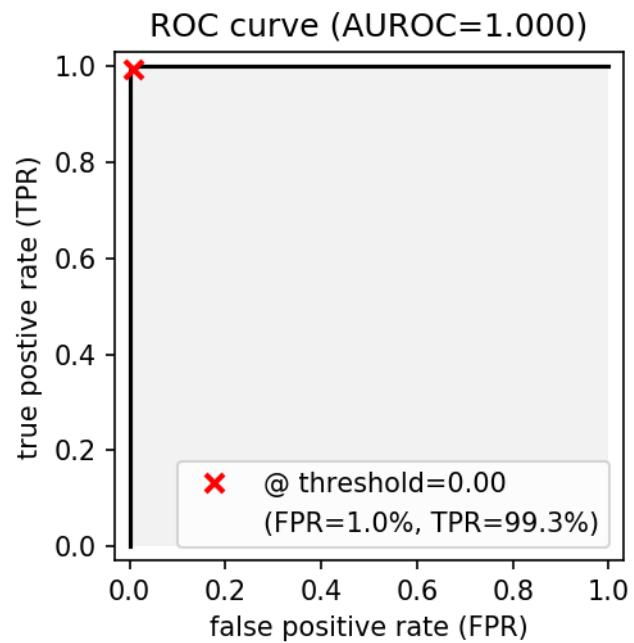
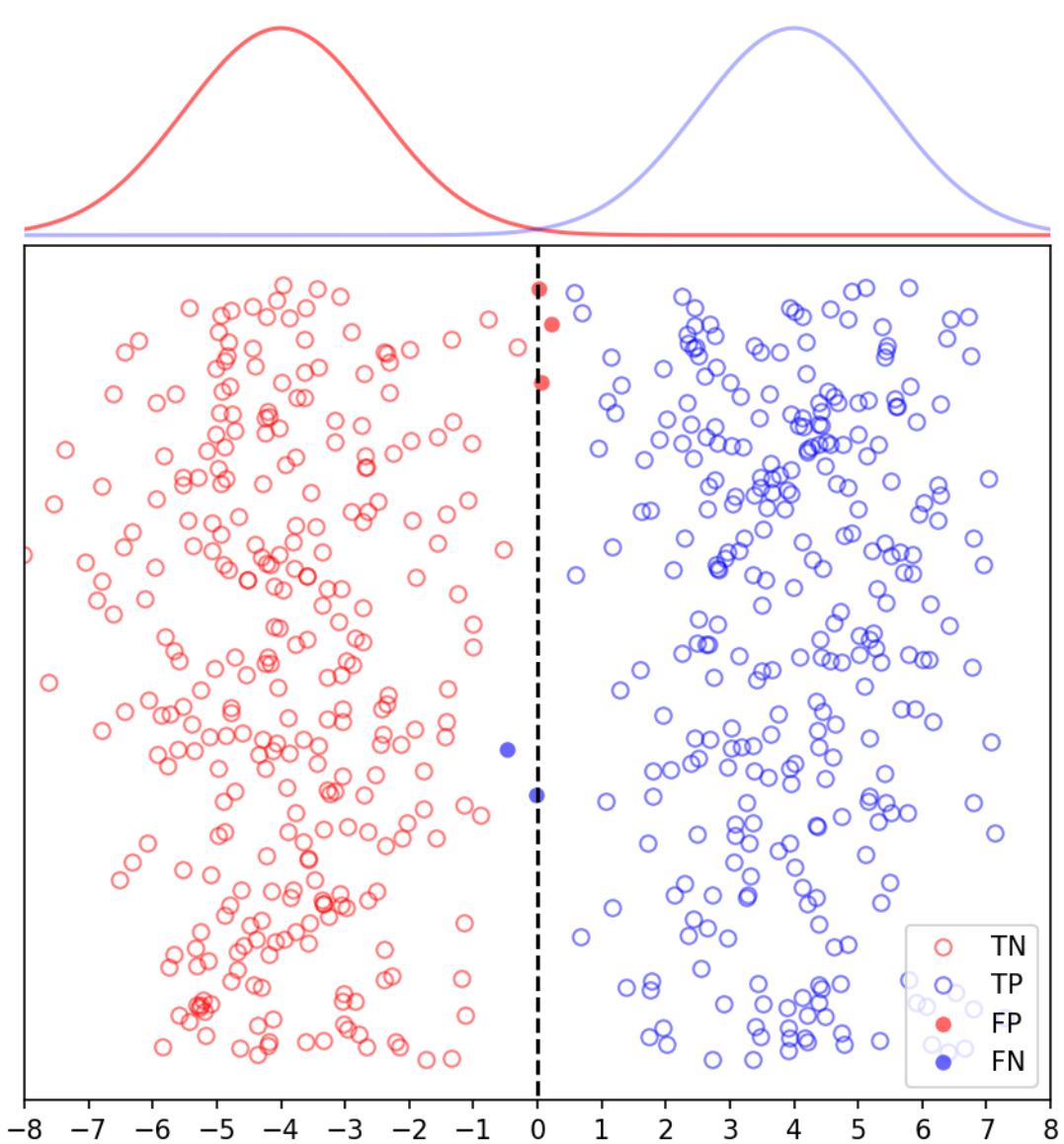
Example of ROC & PR



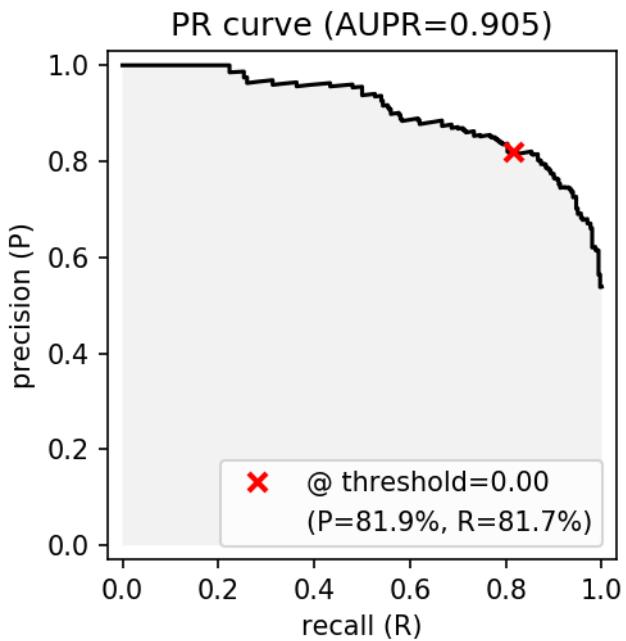
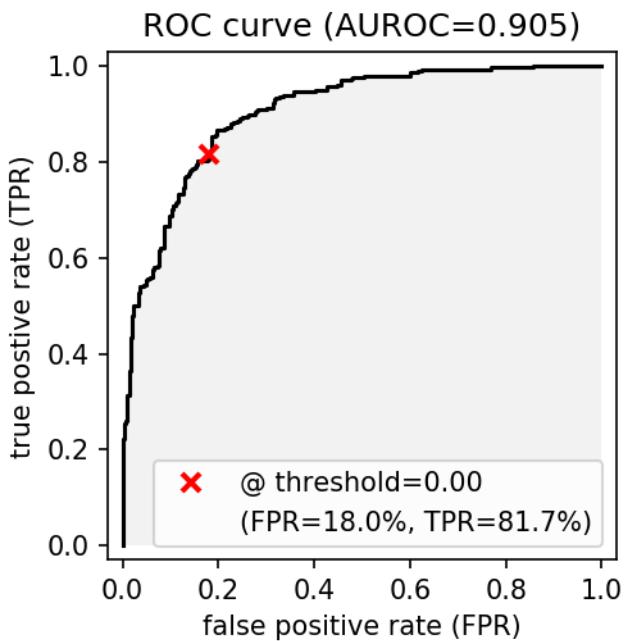
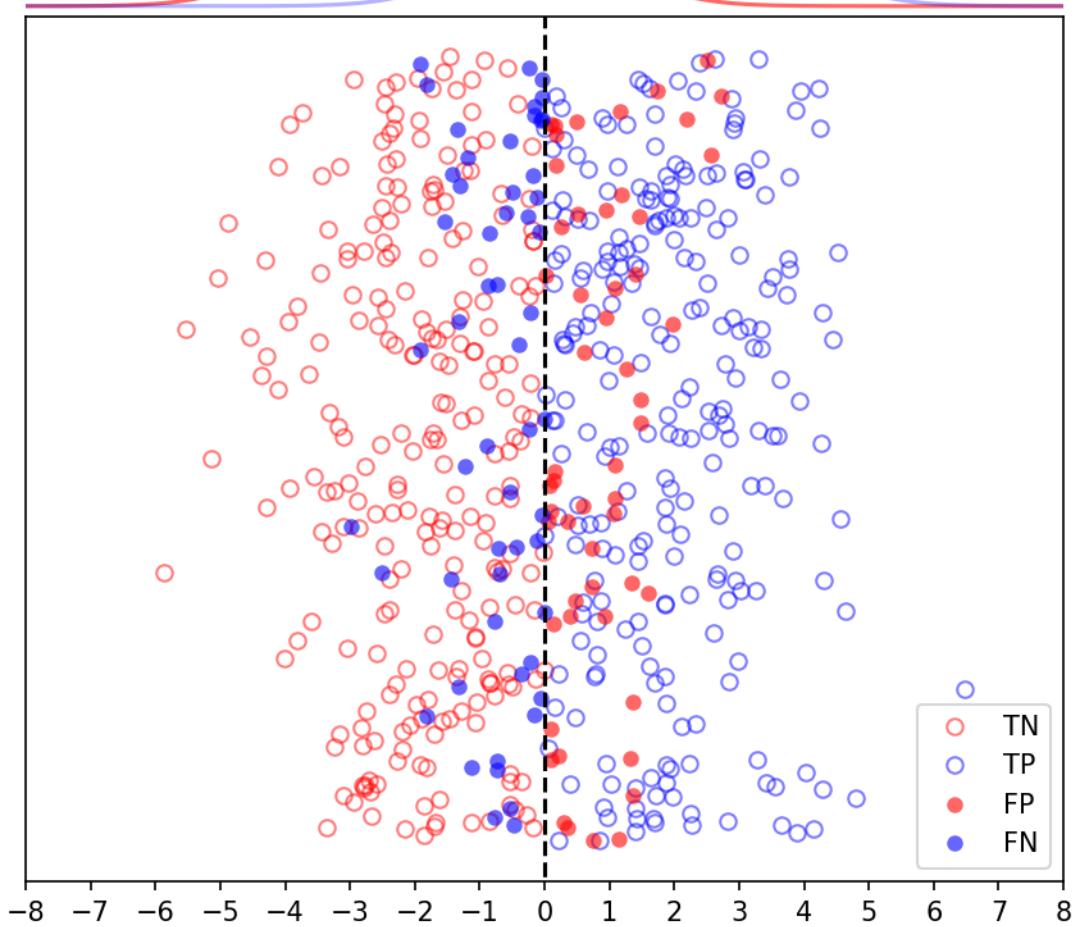
Example of ROC & PR



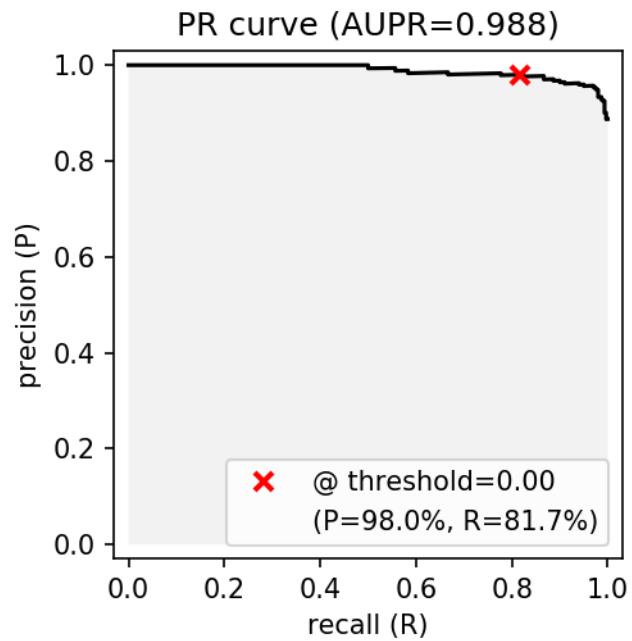
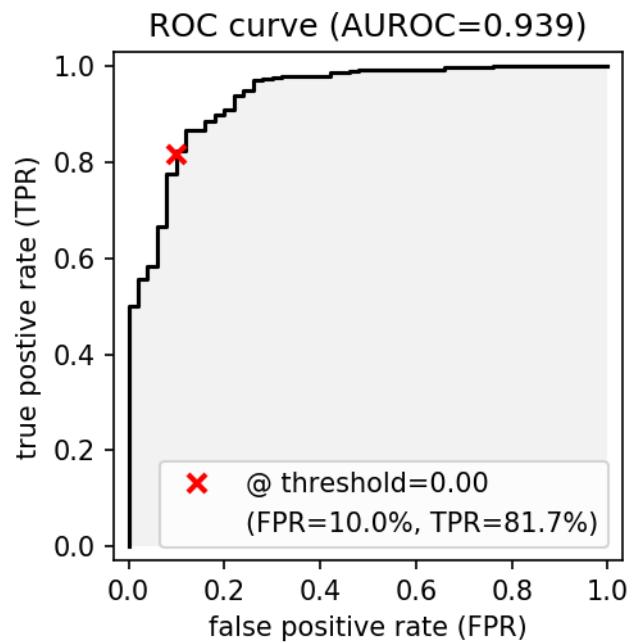
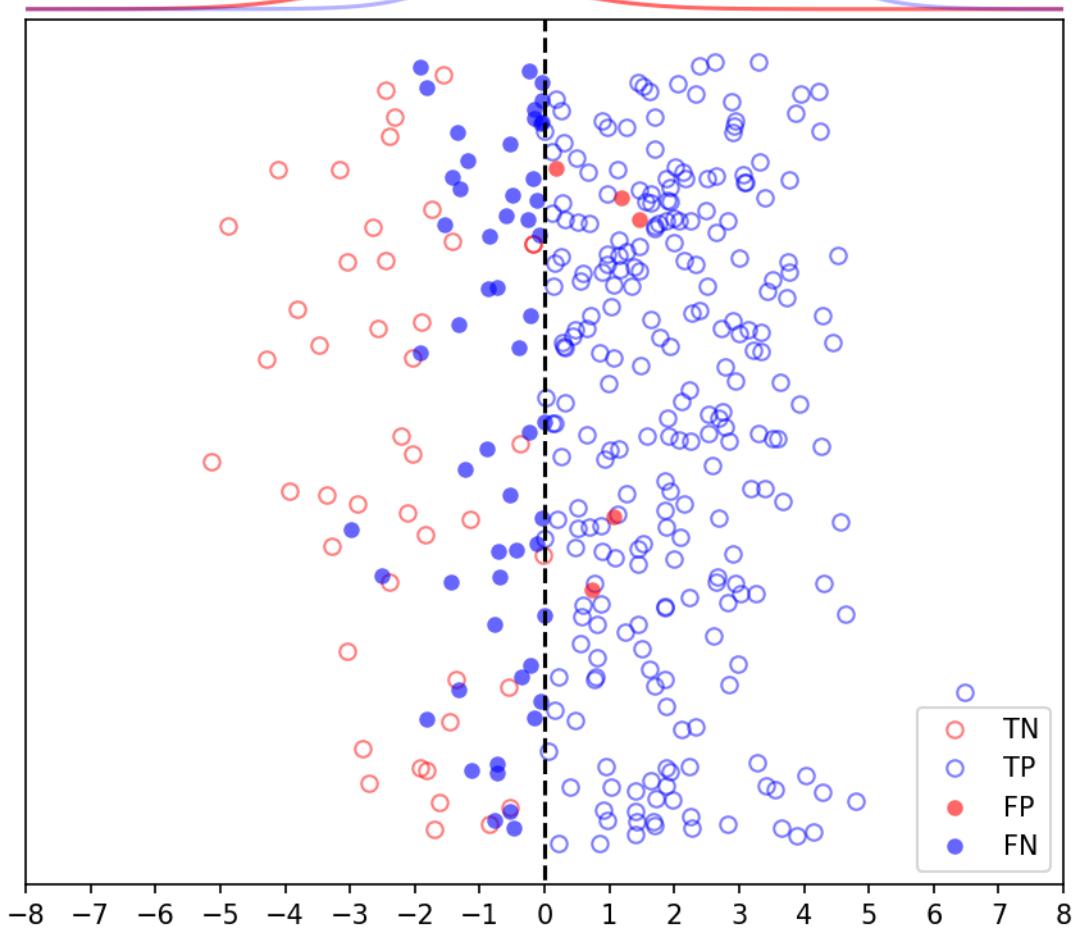
Example of ROC & PR



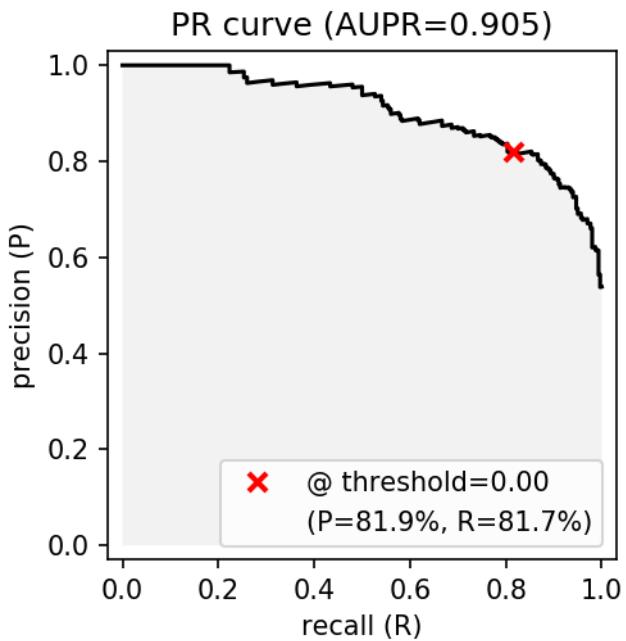
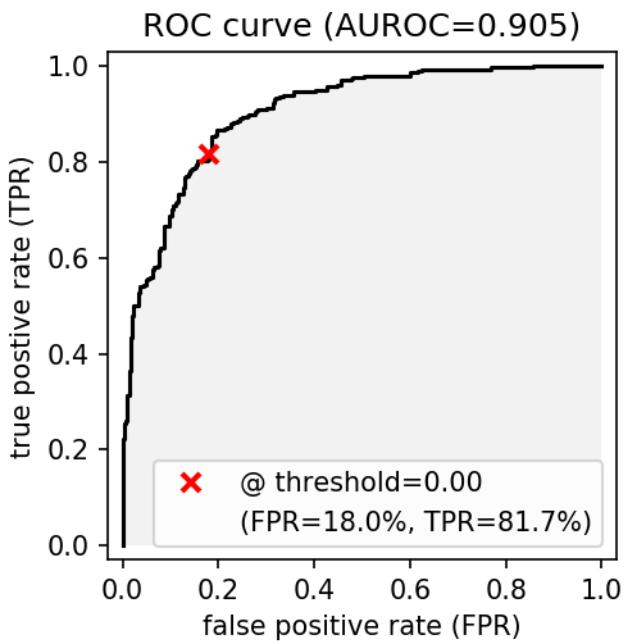
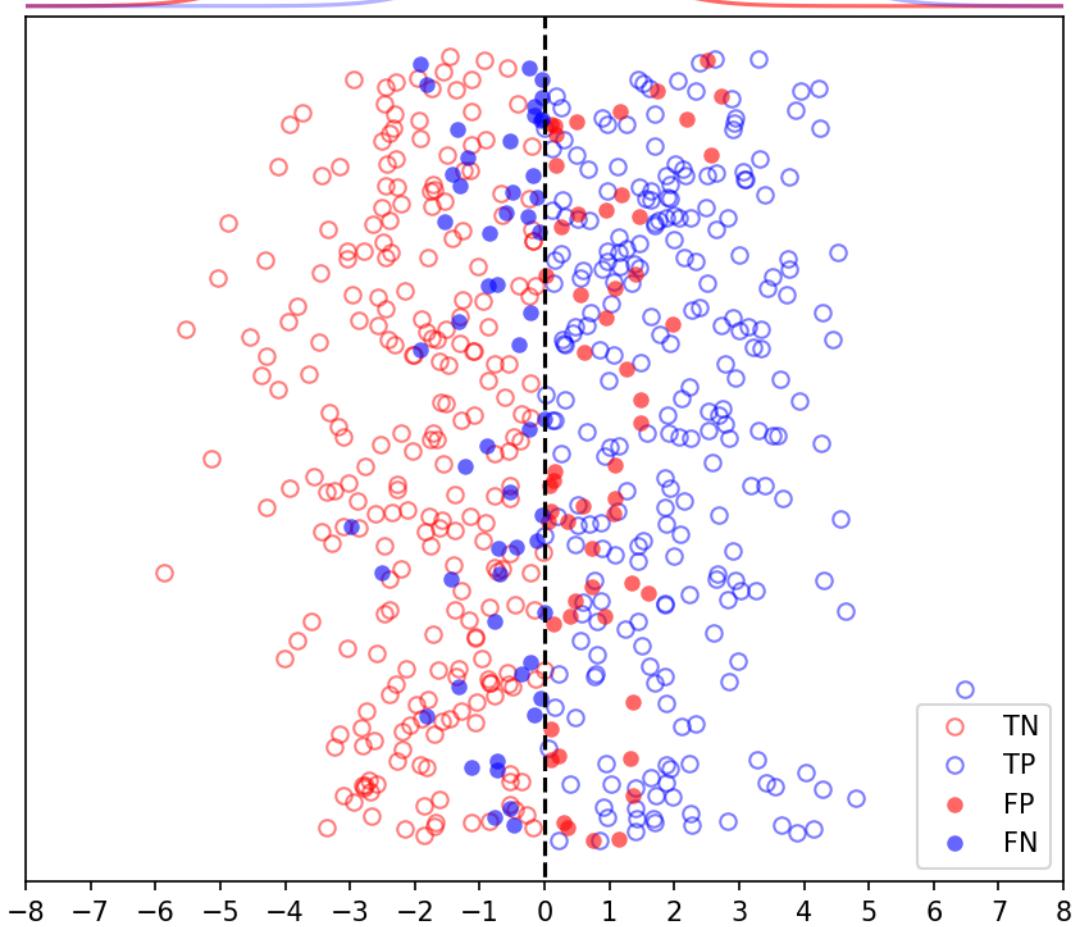
Example of ROC & PR



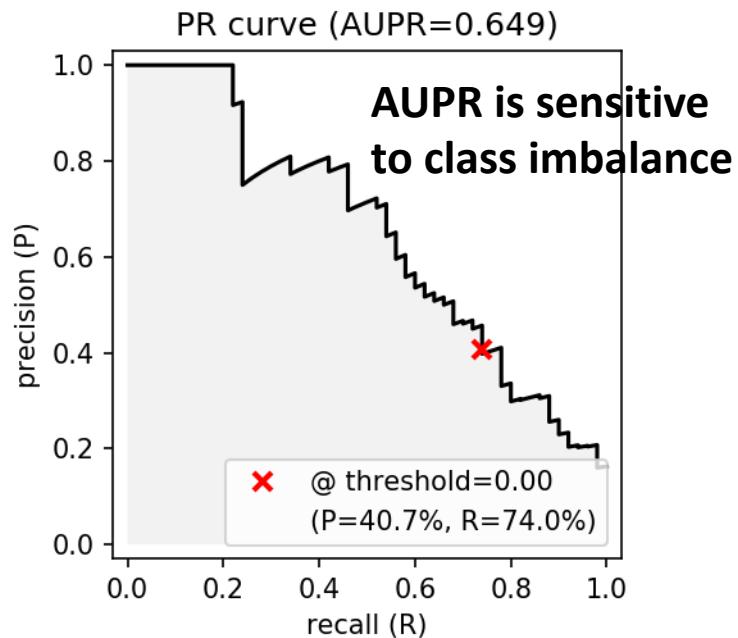
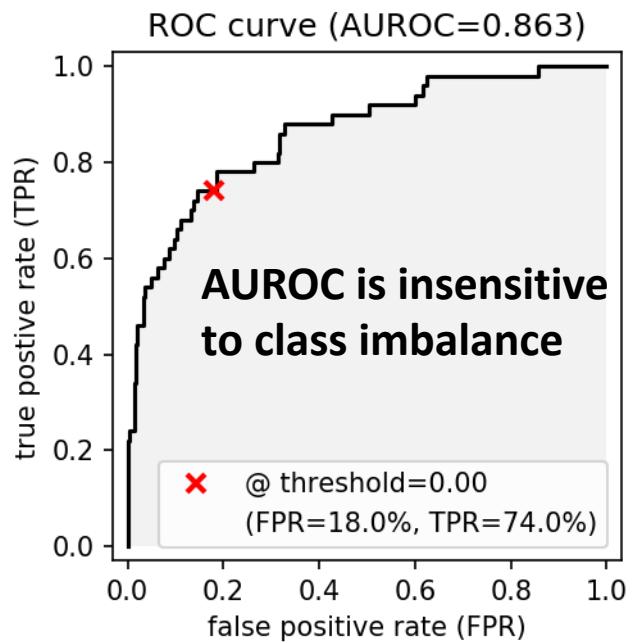
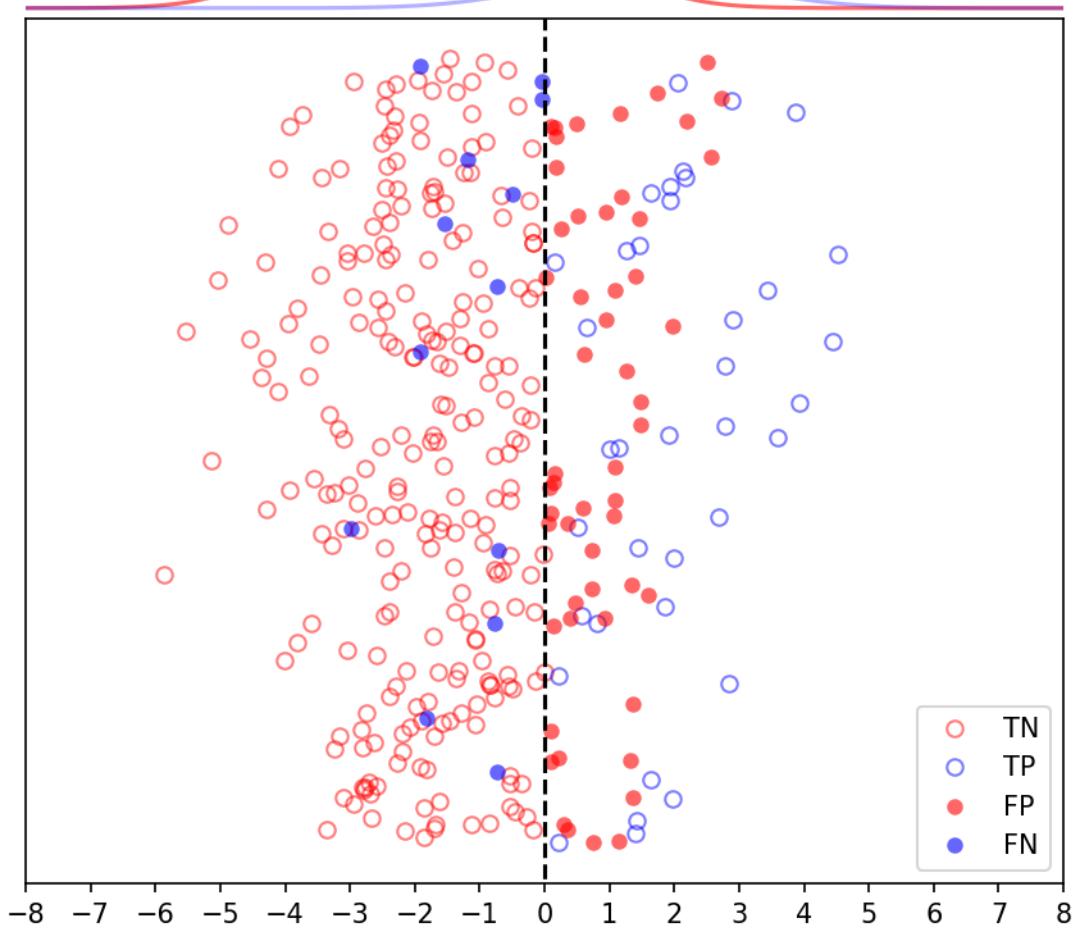
Example of ROC & PR



Example of ROC & PR



Example of ROC & PR



Comparing binary classifiers

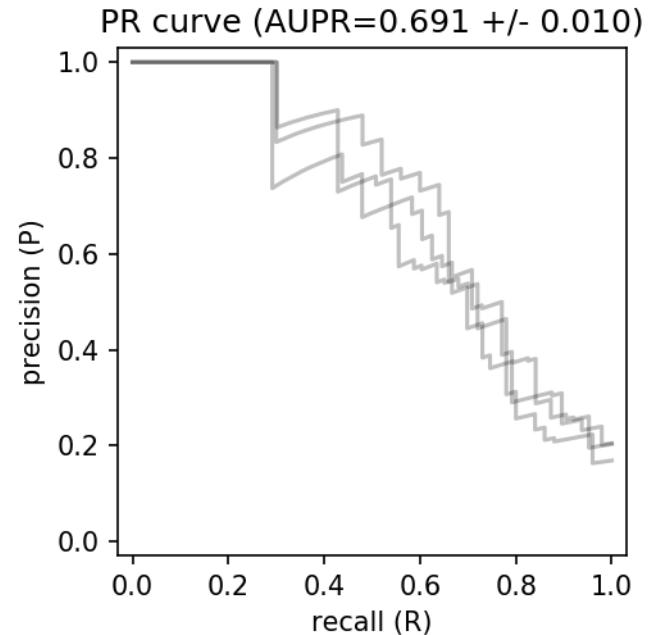
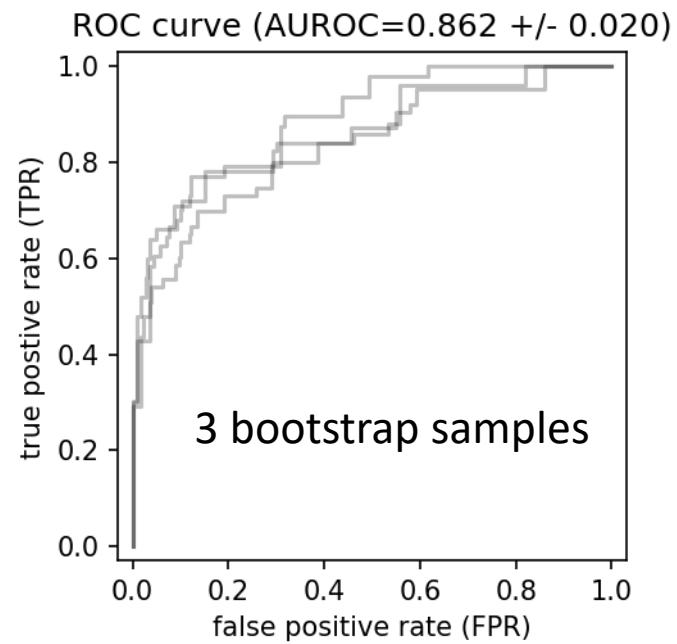
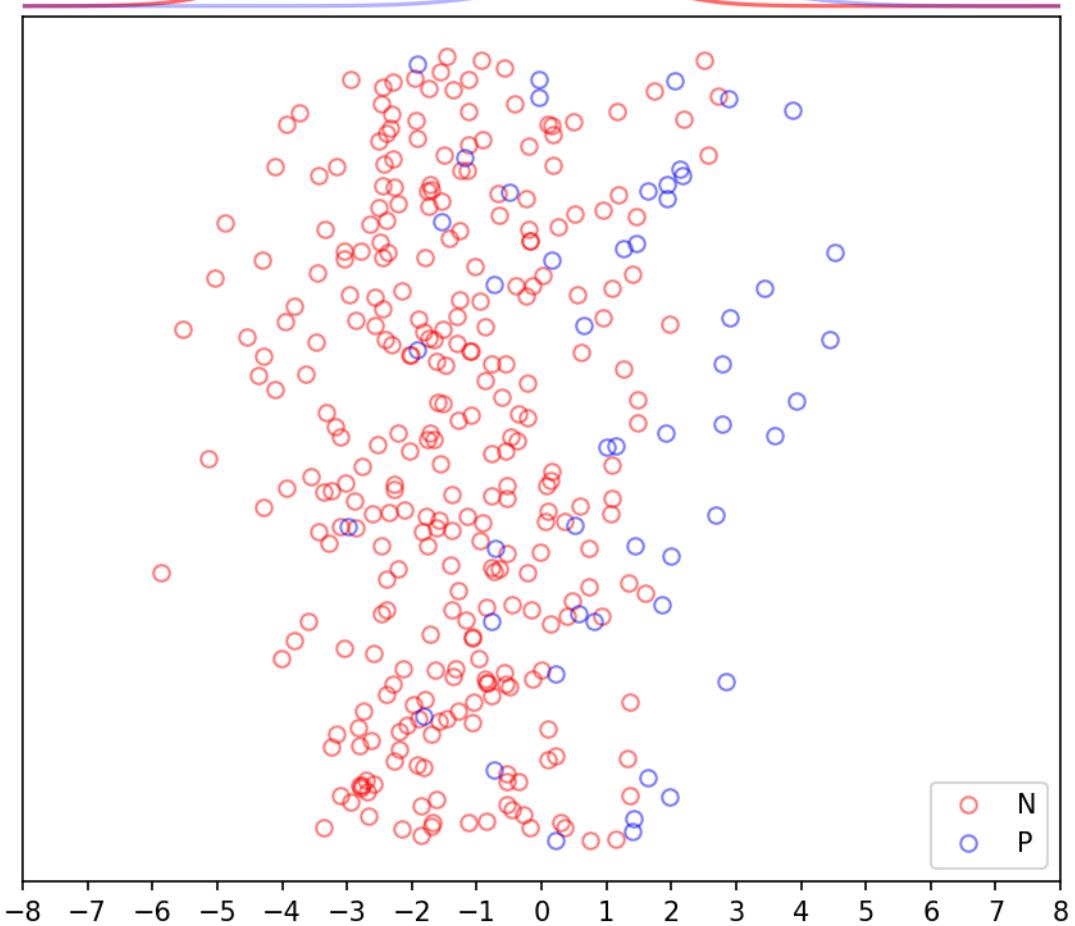
- But how do we compare classifiers A and B when each classifier has *two* numbers?
 - If stakeholders want a particular FPR or FPR or P , easy!
e.g. “choose best cancer diagnostic @ 20% FPR ”
 - Otherwise, no strict ordering over pairs (FPR, TPR) or (R, P)
- 1st idea: ***directly compare the curves***
- Example for ROC curves:
 - If for each FPR the corresponding TPR for classifier A is always greater than the corresponding TPR for classifier B , then “the TPR of A dominates TPR of B . ”
 - This does **not** mean that A and B are compared using the *same threshold*. It means that separate thresholds are chosen to give ***same FPR*** for each classifier.

Comparing binary classifiers

- What if the curves cross, and neither dominates?
- 2nd idea: **compare the *area* under each curve**
- If **AUROC** of A is greater than AUROC of B , then we can expect A to have greater TPR on average across all choices of threshold.
- If **AUPR** of A is greater than AUPR of B , then we can expect A to have greater precision P on average across all choices of recall (all choices of TPR).
- For highly imbalanced classes, consider using AUPR to assess performance, as it is sensitive to imbalance

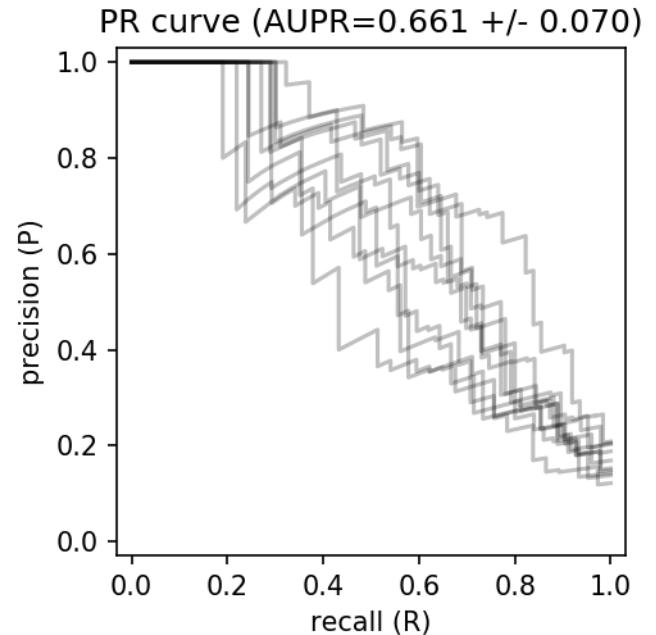
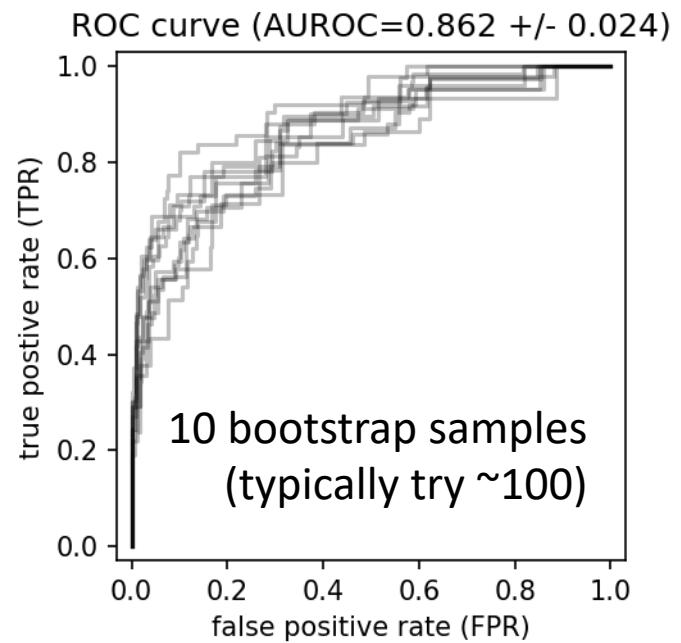
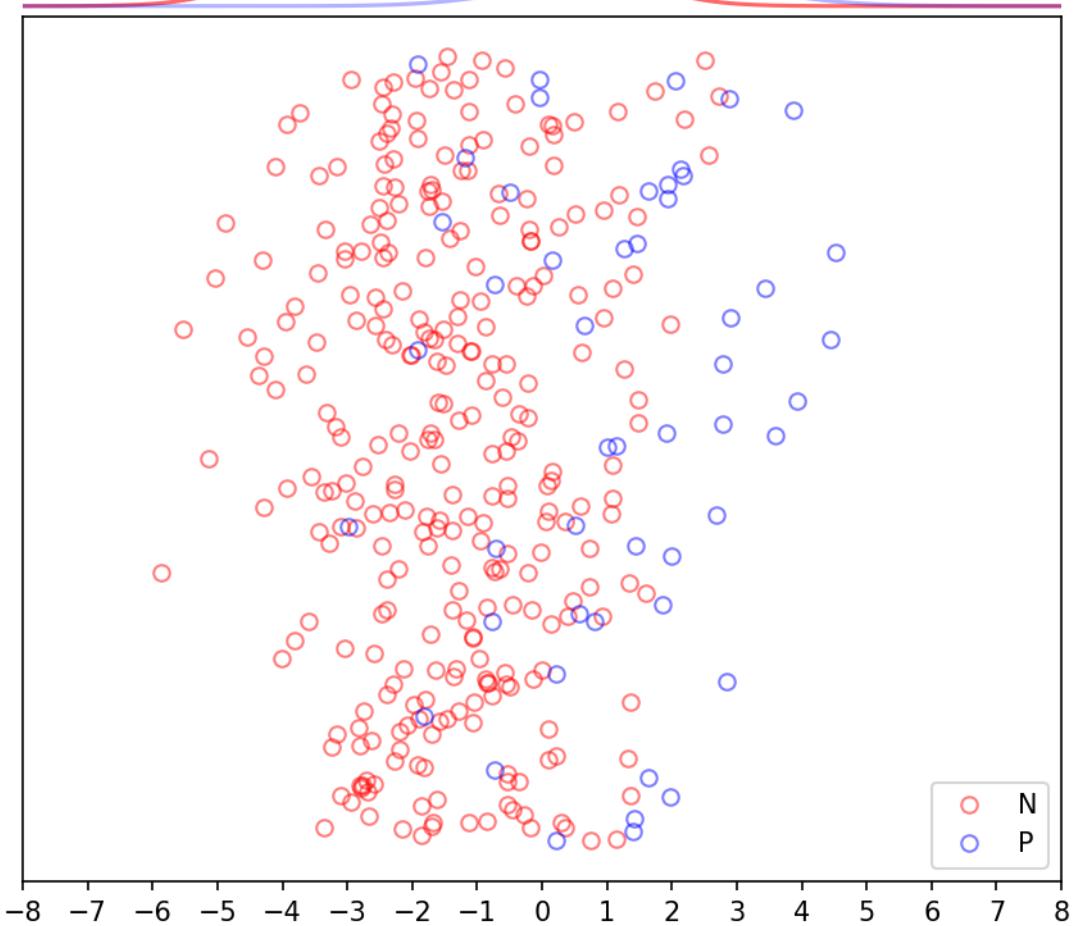
Can bootstrap ROC & PR

Idea: your data set is just one sample from possible data sets that could have been sampled, so use bootstrap to see how sensitive your estimate is to the particular data sample you are using!



Can bootstrap ROC & PR

Idea: your data set is just one sample from possible data sets that could have been sampled, so use bootstrap to see how sensitive your estimate is to the particular data sample you are using!



Viola-Jones cascade: Classic paper using asymmetric loss for *training*

20,000 citations!

ACCEPTED CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION 2001

Rapid Object Detection using a Boosted Cascade of Simple Features

Paul Viola

viola@merl.com

Mitsubishi Electric Research Labs
201 Broadway, 8th FL
Cambridge, MA 02139

Michael Jones

mjones@crl.dec.com
Compaq CRL
One Cambridge Center
Cambridge, MA 02142

- Trains an early-stage detector to quickly reject image patches that “obviously not a face”
 - Loss penalizes false-negatives heavily, false positives only lightly
- Later stage classifier only sees the “hard” cases, trained on a more balanced loss function

Viola-Jones face detector

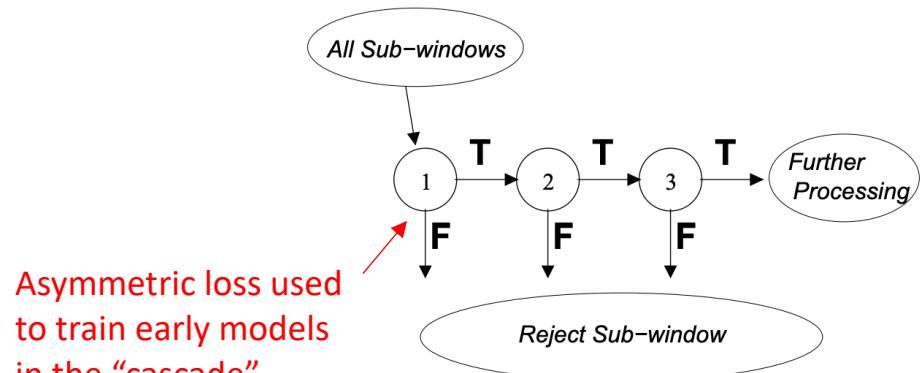


Figure 4: Schematic depiction of a the detection cascade.

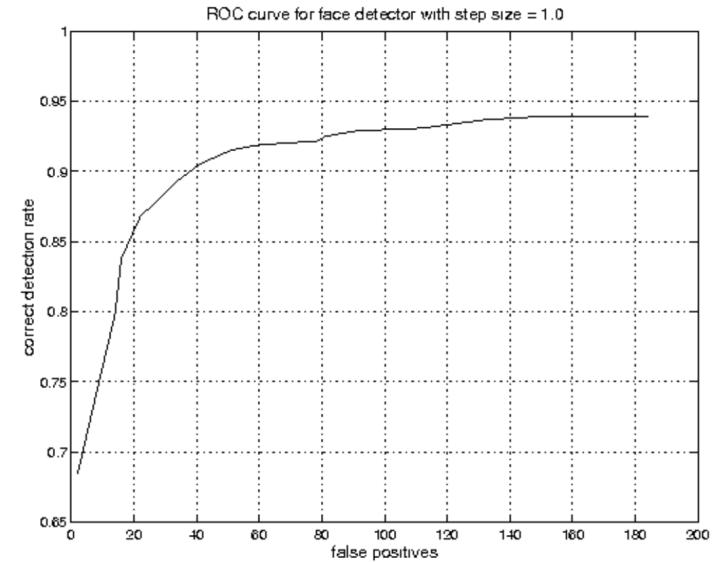
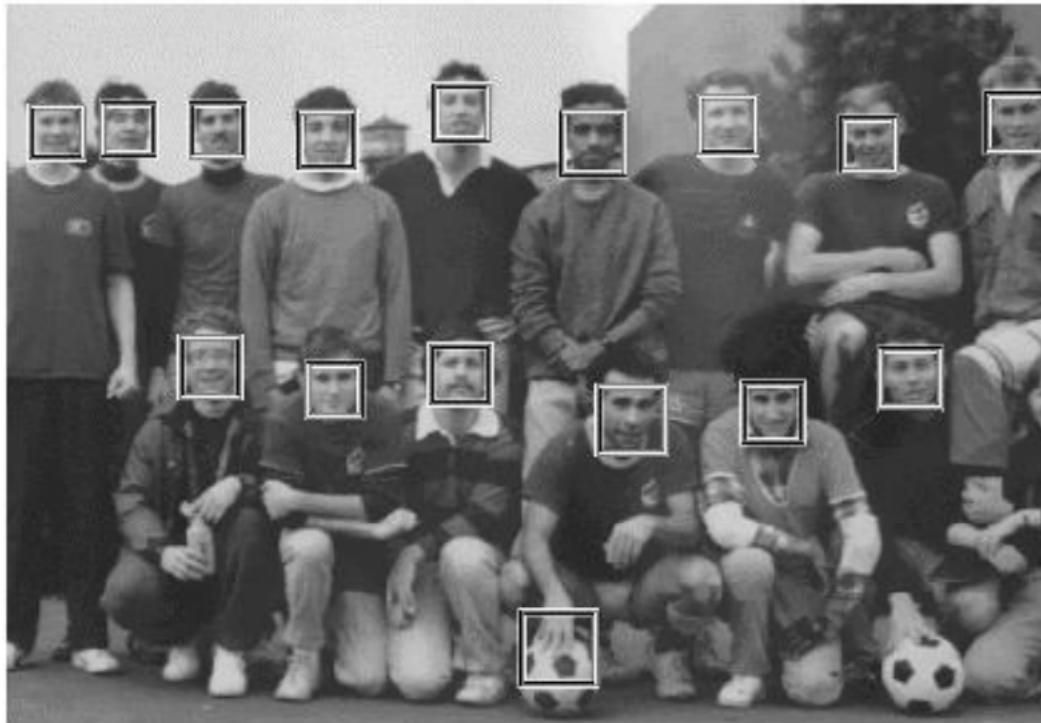
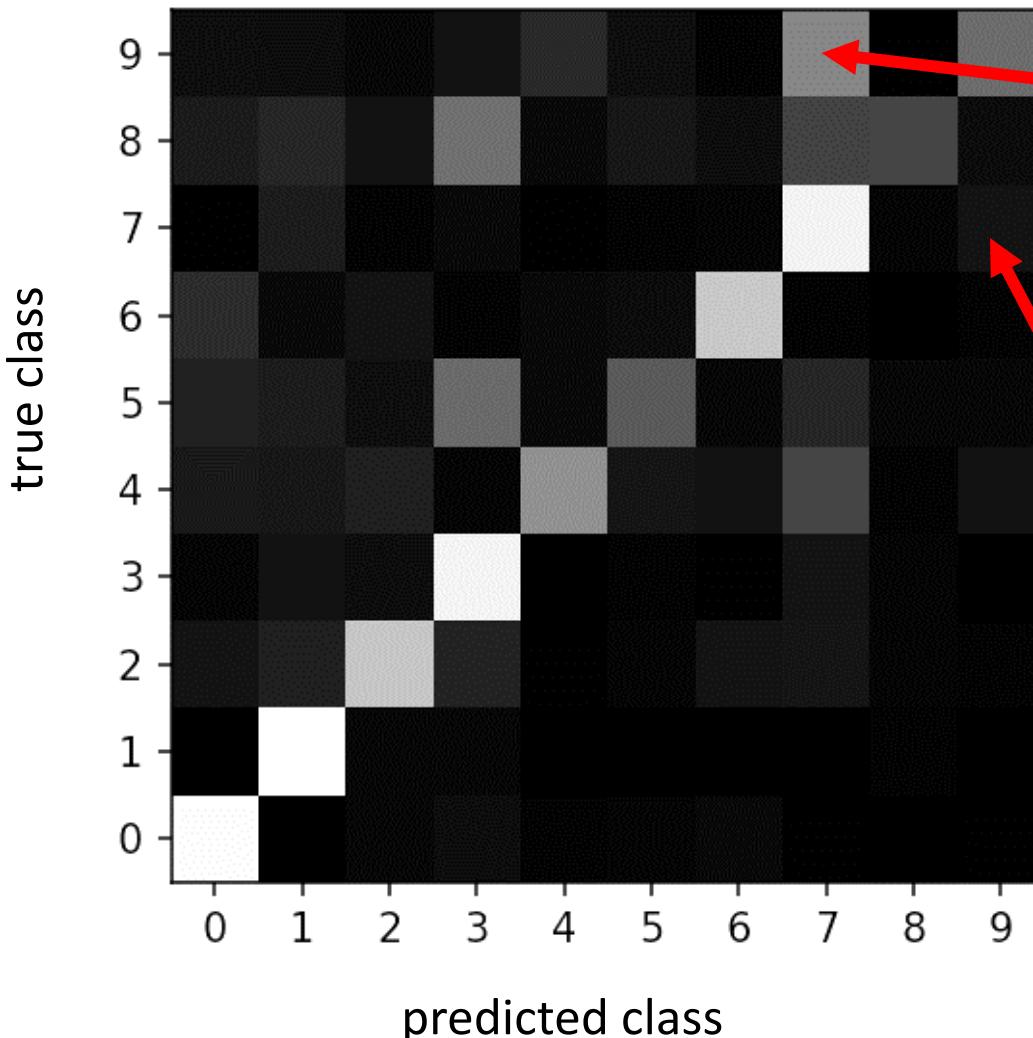


Figure 6: ROC curve for our face detector on the MIT+CMU test set. The detector was run using a step size of 1.0 and starting scale of 1.0 (75,081,800 sub-windows scanned).

Confusion matrix

MNIST confusion matrix



Number of times a “9” digit was mistakenly predicted to be a “7”



Number of times a “7” digit was mistakenly predicted to be a “9”



sklearn.metrics.confusion_matrix

```
sklearn.metrics. confusion_matrix(y_true, y_pred, labels=None, sample_weight=None)
```

[source]

Compute confusion matrix to evaluate the accuracy of a classification

By definition a confusion matrix C is such that $C_{i,j}$ is equal to the number of observations known to be in group i but predicted to be in group j .

Thus in binary classification, the count of true negatives is $C_{0,0}$, false negatives is $C_{1,0}$, true positives is $C_{1,1}$ and false positives is $C_{0,1}$.

Read more in the [User Guide](#).

Parameters:

y_true : array, shape = [n_samples]

Ground truth (correct) target values.

y_pred : array, shape = [n_samples]

Estimated targets as returned by a classifier.

labels : array, shape = [n_classes], optional

List of labels to index the matrix. This may be used to reorder or select a subset of labels. If none is given, those that appear at least once in `y_true` or `y_pred` are used in sorted order.

sample_weight : array-like of shape = [n_samples], optional

Sample weights.

Returns:

C : array, shape = [n_classes, n_classes]

Classification metrics

See the [Classification metrics](#) section of the user guide for further details.

→	<code>metrics.accuracy_score (y_true, y_pred[, ...])</code>	Accuracy classification score.
→	<code>metrics.auc (x, y[, reorder])</code>	Compute Area Under the Curve (AUC) using the trapezoidal rule.
→	<code>metrics.average_precision_score (y_true, y_score)</code>	Compute average precision (AP) from prediction scores.
→	<code>metrics.balanced_accuracy_score (y_true, y_pred)</code>	Compute the balanced accuracy.
→	<code>metrics.brier_score_loss (y_true, y_prob[, ...])</code>	Compute the Brier score.
→	<code>metrics.classification_report (y_true, y_pred)</code>	Build a text report showing the main classification metrics.
→	<code>metrics.cohen_kappa_score (y1, y2[, labels, ...])</code>	Cohen's kappa: a statistic that measures inter-rater agreement for categorical items.
→	<code>metrics.confusion_matrix (y_true, y_pred[, ...])</code>	Compute confusion matrix to evaluate the quality of a classification.
→	<code>metrics.f1_score (y_true, y_pred[, labels, ...])</code>	Compute the F1 score, also known as balanced F-score.
→	<code>metrics.fbeta_score (y_true, y_pred, beta[, ...])</code>	Compute the F-beta score.
→	<code>metrics.hamming_loss (y_true, y_pred[, ...])</code>	Compute the average Hamming loss.
→	<code>metrics.hinge_loss (y_true, pred_decision[, ...])</code>	Average hinge loss (non-regularized).
→	<code>metrics.jaccard_score (y_true, y_pred[, ...])</code>	Jaccard similarity coefficient score.
→	<code>metrics.log_loss (y_true, y_pred[, eps, ...])</code>	Log loss, aka logistic loss or cross-entropy loss.
→	<code>metrics.matthews_corrcoef (y_true, y_pred[, ...])</code>	Compute the Matthews correlation coefficient (MCC).
→	<code>metrics.multilabel_confusion_matrix (y_true, ...)</code>	Compute a confusion matrix for each class.
→	<code>metrics.precision_recall_curve (y_true, ...)</code>	Compute precision-recall pairs for different probability thresholds.
→	<code>metrics.precision_recall_fscore_support (...)</code>	Compute precision, recall, F-measure and support for each class.
→	<code>metrics.precision_score (y_true, y_pred[, ...])</code>	Compute the precision.
→	<code>metrics.recall_score (y_true, y_pred[, ...])</code>	Compute the recall.
→	<code>metrics.roc_auc_score (y_true, y_score[, ...])</code>	Compute Area Under the Receiver Operating Characteristic curve.
→	<code>metrics.roc_curve (y_true, y_score[, ...])</code>	Compute Receiver operating characteristic curve.
→	<code>metrics.zero_one_loss (y_true, y_pred[, ...])</code>	Zero-one classification loss.

Surrogate loss functions

- When training a classifier to be cost-sensitive, we tend to use *weighted* version of its “natural” loss function
 - Logistic regression / neural nets: weighted version of NLL

$$-\sum_{i=1}^N C_1 y_i \ln \hat{y}_i + C_0 (1 - y_i) \ln(1 - \hat{y}_i)$$

- SVM: weighted version of margin violation penalty
$$+ C_1 \sum_{i : t_i = +1} \xi_i + C_0 \sum_{i : t_i = -1} \xi_i$$
- Used *even if test performance measured by AUC/AUPR*
 - When our training loss doesn't match the test metric we plan to use, it is called a **surrogate loss** or **proxy loss**.

Surrogate loss functions

- If AUC or AUPR is what we care about, why not train to maximize it *directly*? Why a surrogate?
-

AUC Optimization vs. Error Rate Minimization

Corinna Cortes* and Mehryar Mohri
AT&T Labs – Research

Abstract

The *area under an ROC curve* (AUC) is a criterion used in many applications to measure the quality of a classification algorithm. However, the objective function optimized in most of these algorithms is the error rate and not the AUC value. We give a detailed statistical analysis of the relationship between the AUC and the error rate, including the first exact expression of the expected value and the variance of the AUC for a fixed error rate. Our results show that the average AUC is monotonically increasing as a function of the classification accuracy, but that the standard deviation for uneven distributions and higher error rates is noticeable. Thus, algorithms designed to minimize the error rate may not lead to the best possible AUC values. We show that, under certain conditions, the global function optimized by the RankBoost algorithm is exactly the AUC. We report the results of our experiments with RankBoost in several datasets demonstrating the benefits of an algorithm specifically designed to globally optimize the AUC over other existing algorithms optimizing an approximation of the AUC or only locally optimizing the AUC.

Because optimization of an AUC loss is very hard

- Gradients hard to compute
- Non-convex optimization

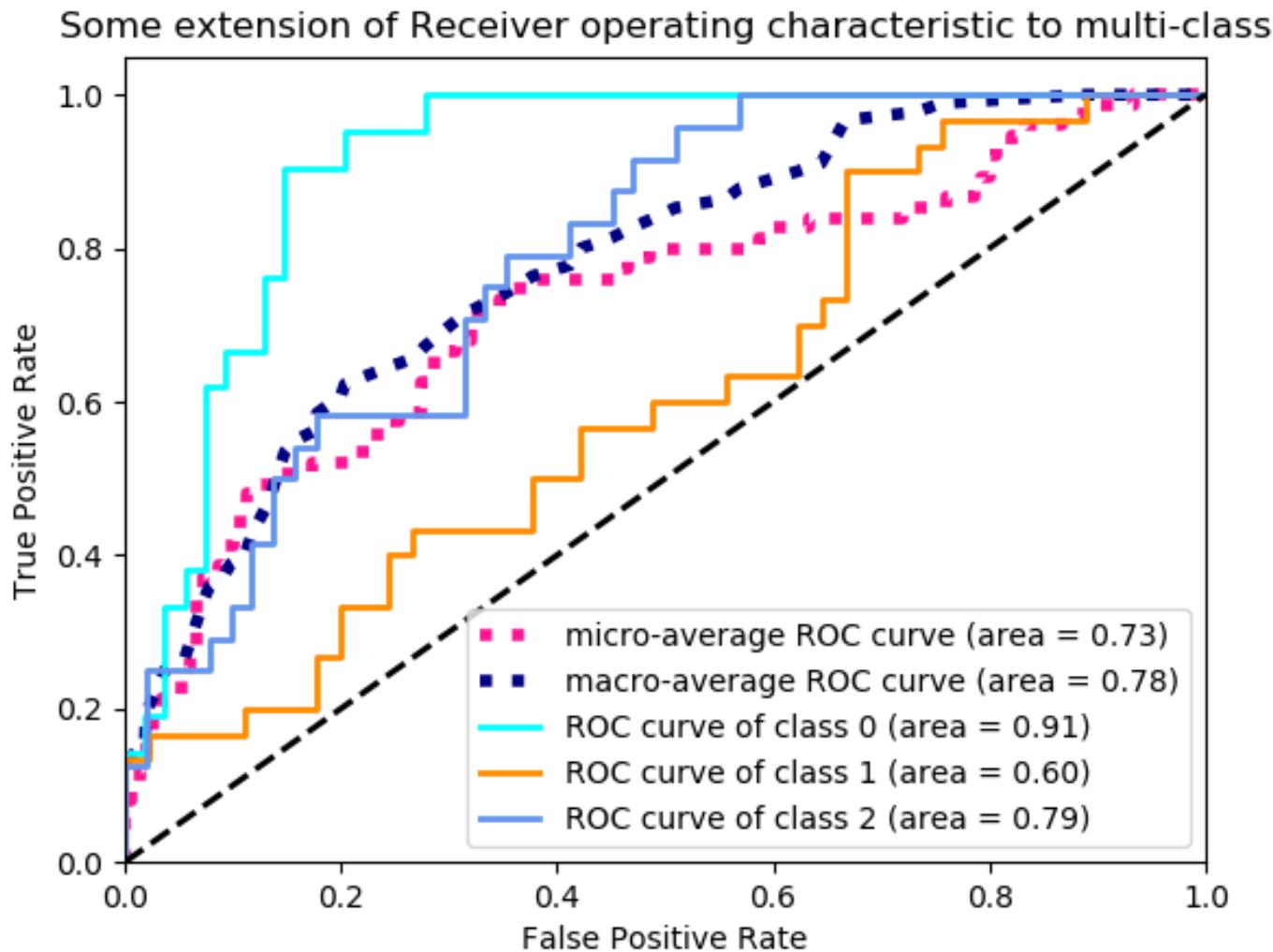
Still, smart people have tried and with some success!



ROC and PR curves for multi-class?

- Multiclass ROC/PR an area of active research, e.g.
<http://proceedings.mlr.press/v97/kleiman19a/kleiman19a.pdf>
- Simplest multiclass extensions treat each of the K predictions as “One-vs-Rest” binary, then compute an ROC or PR curve from $K*N$ binary predictions
 - ***Micro average***: treat each binary prediction \hat{y}_{ik} and binary target y_{ik} as independent, and compute standard ROC or PR curve (and AUROC or AUPR).
 - ***Macro average***: compute K separate curves (ROC or PR) and average their y-axis value at each unique x-axis point (TPR for ROC, P for PR) to get one ‘average’ curve. (Note: class imbalances now averaged out even for PR.)

Example of 3-class ROC averaging



Cross Validation

- ***Cross validation*** is a method for *estimating* the test-time performance of a predictor,
 - *i.e.*, its performance on held-out data or future data.
- **Model selection** is about choosing the model that is expected to perform the best on test data.
- **Hyperparameter search** is a particular aspect of model selection, where choices that are normally held fixed during training (*max_depth*, etc.) are explicitly adapted so as to maximize *expected test-time* performance (not training performance).
 - Must never “peek” at the test data!

3. Model selection and evaluation

3.1. Cross-validation: evaluating estimator performance

- ▪ 3.1.1. Computing cross-validated metrics
- ▪ 3.1.2. Cross validation iterators
- ▪ 3.1.3. A note on shuffling
- 3.1.4. Cross validation and model selection

3.2. Tuning the hyper-parameters of an estimator

- ▪ 3.2.1. Exhaustive Grid Search
- ▪ 3.2.2. Randomized Parameter Optimization
- 3.2.3. Tips for parameter search
- 3.2.4. Alternatives to brute force parameter search

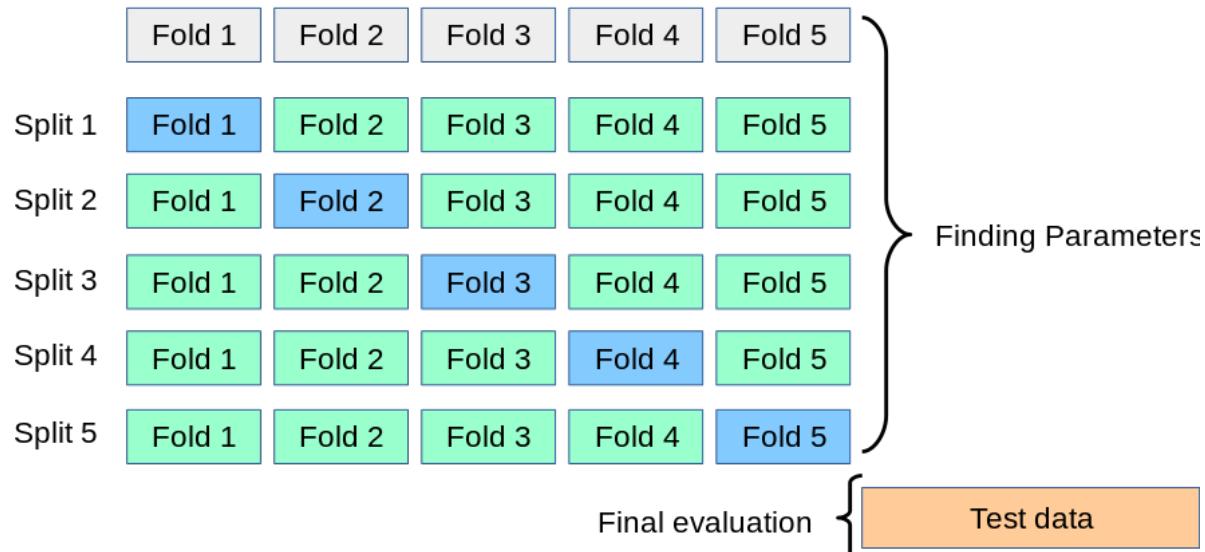
3.3. Metrics and scoring: quantifying the quality of predictions

- 3.3.1. The **scoring** parameter: defining model evaluation rules
- ▪ 3.3.2. Classification metrics
- 3.3.3. Multilabel ranking metrics
- ▪ 3.3.4. Regression metrics
- 3.3.5. Clustering metrics
- ▪ 3.3.6. Dummy estimators

K-Fold Cross Validation

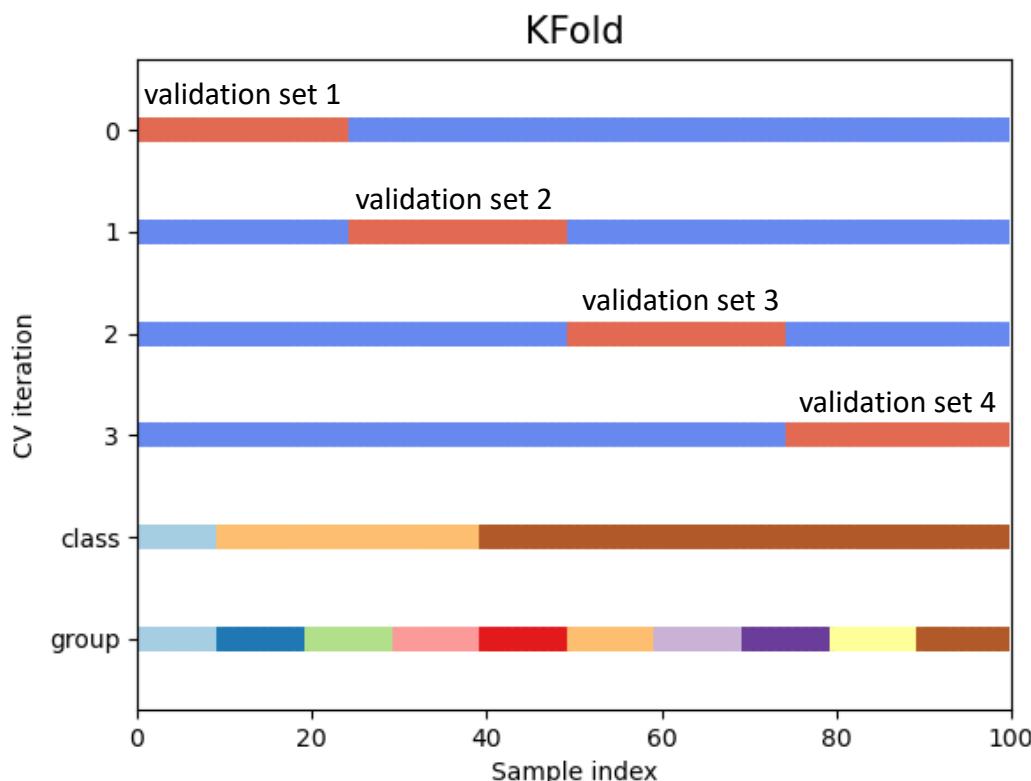
- This is your go-to cross validation method when the data is expected to be independently sampled

You train K separate models, each on a different subset of the training data



K-Fold Cross Validation

- This is your go-to cross validation method when the data is expected to be independently sampled



Advantage over doing K independent random splits: each data point gets a chance to be in exactly one validation set

sklearn.model_selection.KFold

```
class sklearn.model_selection.KFold(n_splits=5, *, shuffle=False, random_state=None) ¶
```

[source]

K-Folds cross-validator

Provides train/test indices to split data in train/test sets. Split dataset into k consecutive folds (without shuffling by default).

Each fold is then used once as a validation while the k - 1 remaining folds form the training set.

Read more in the [User Guide](#).

Parameters: **n_splits : int, default=5**

Number of folds. Must be at least 2.

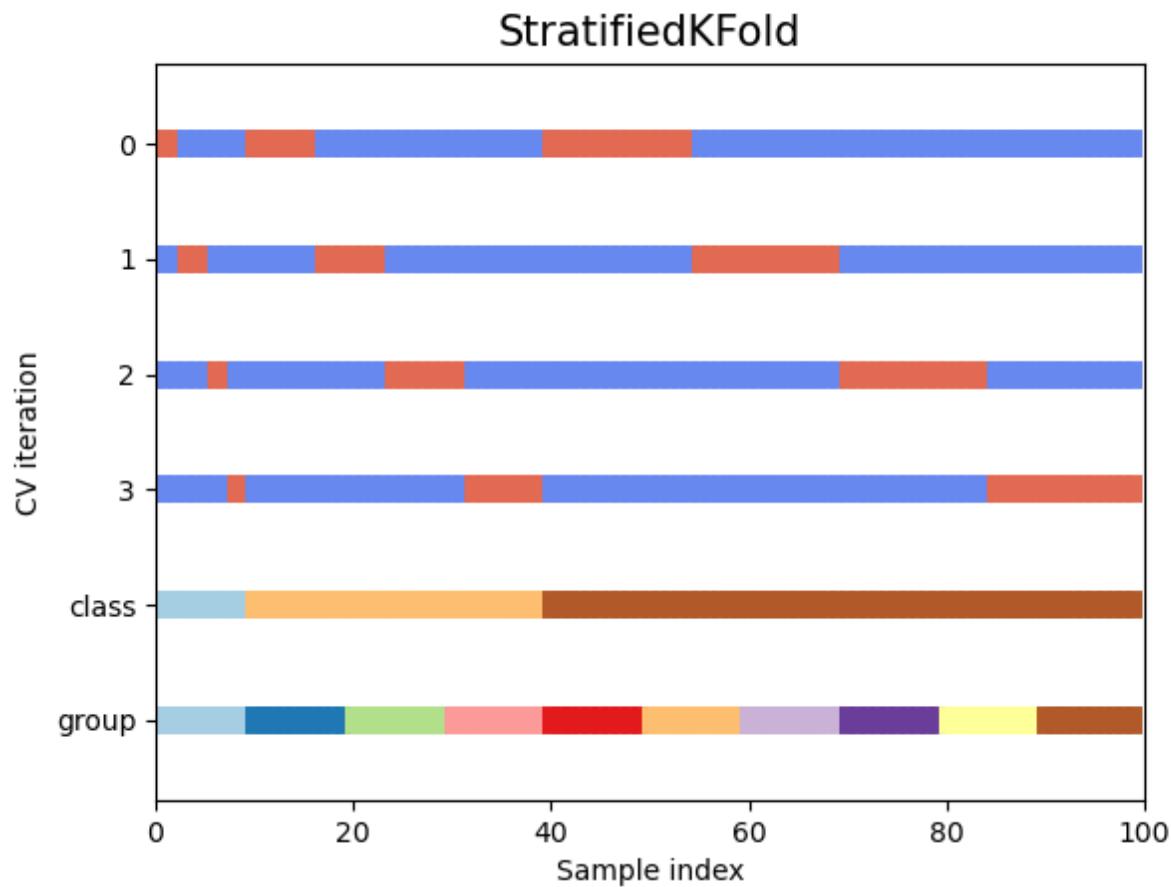
Changed in version 0.22: n_splits default value changed from 3 to 5.

shuffle : bool, default=False

Whether to shuffle the data before splitting into batches. Note that the samples within each split will not be shuffled.

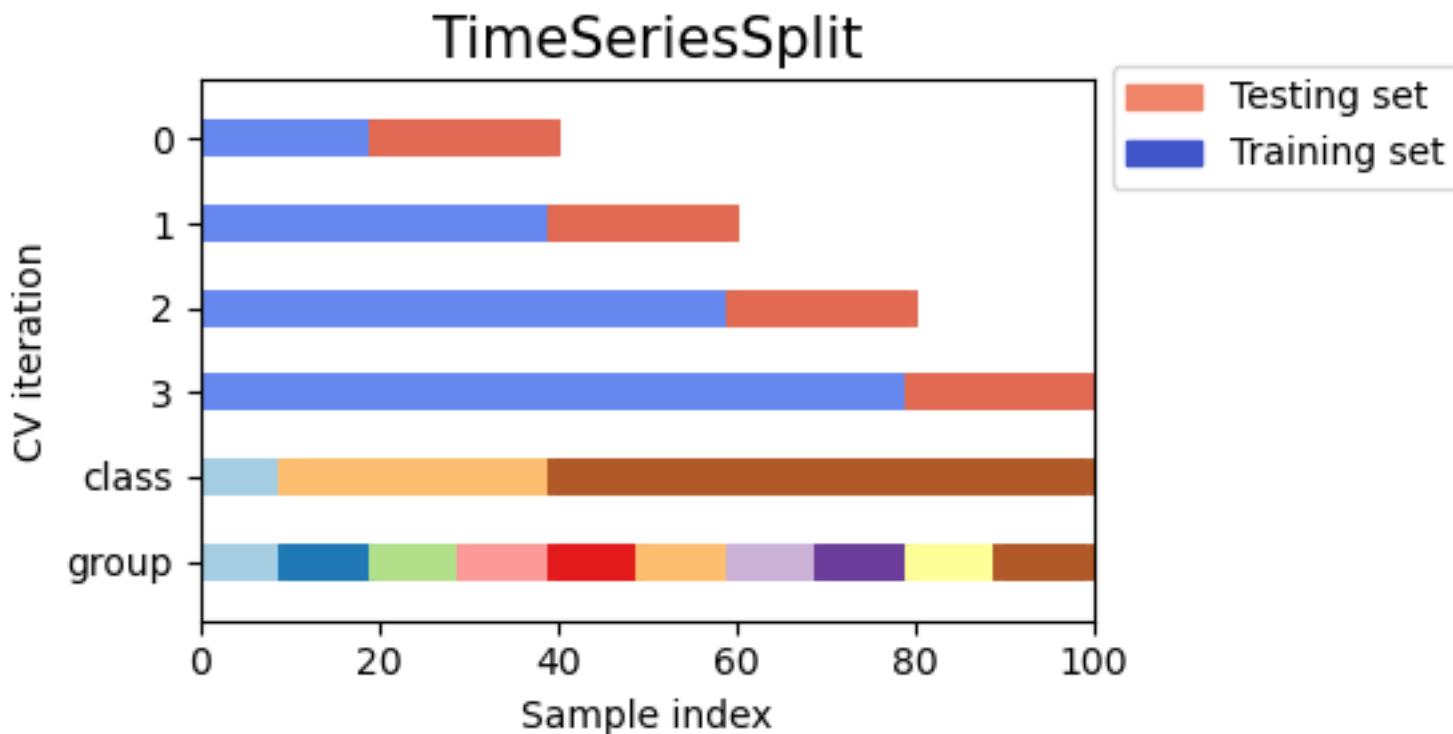
Stratified K-Fold

- Stratified K-fold balances the classes within each fold.



Time Series Split

- Each fold comprises only historical data, never data from the “future”



Hyperparameter search

- (Placeholder)

PRML Readings

§1.5.2 Minimizing the expected loss

§1.5.5 Equation (1.91) and Figure 1.29 ONLY

Scikit-learn documentation does a great job of explaining classification metrics and cross-validation schemes.