Evaluating and Comparing Models

Dr. Fayyaz Minhas

Department of Computer Science

University of Warwick

https://warwick.ac.uk/fac/sci/dcs/teaching/material/cs909/

Objectives of this lecture

- How to compare machine learning models?
  - My classifier is better than yours
- How to select the optimal parameters of a machine learning model?
  - How should I choose “C” or “k”?
- Organization
  - **Philosophical Foundations**
  - **How to evaluate accuracy**
    - **Metrics: Accuracy, FPR, TRP, PPV, ROC, PR-Curves, F-measure**
    - **Cross-Validation and Resampling**

Why Evaluate Models?

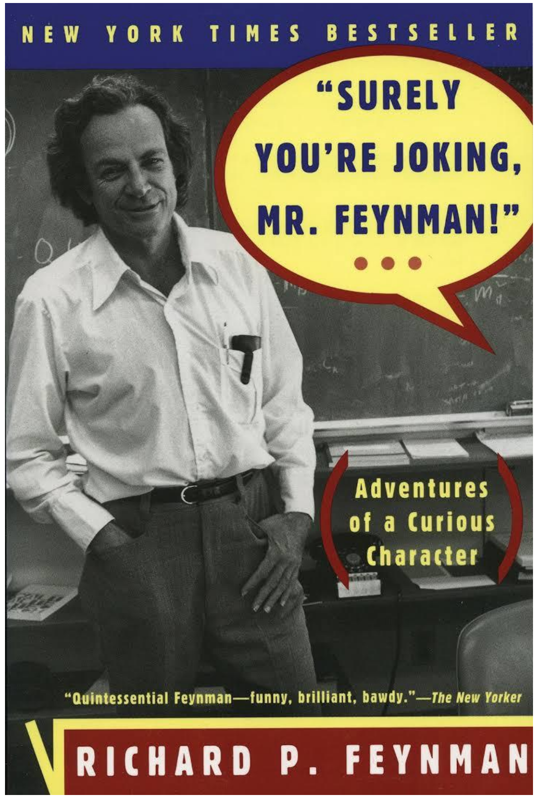
- Guess
- Evaluate consequences
- Compare to Nature / Experiment / Experience

- If it disagrees with nature, it's wrong!

- *"It doesn't matter how beautiful your theory is, it doesn't matter how smart you are. If it doesn't agree with experiment, it's wrong. "*

- **Objective:**
  - We want to find out which model "fits" our data best for use in the real world.
  - AKA "Will it Work?"
  - More often, "Why doesn't it work?"

- Turing Test


NEW YORK TIMES BESTSELLER
"SURELY YOU'RE JOKING, MR. FEYNMAN!"...
Adventures of a Curious Character
"Quintessential Feynman—funny, brilliant, bawdy."—The New Yorker
RICHARD P. FEYNMAN

Richard Phillips Feynman 1918-1988

https://en.wikipedia.org/wiki/Richard_Feynman

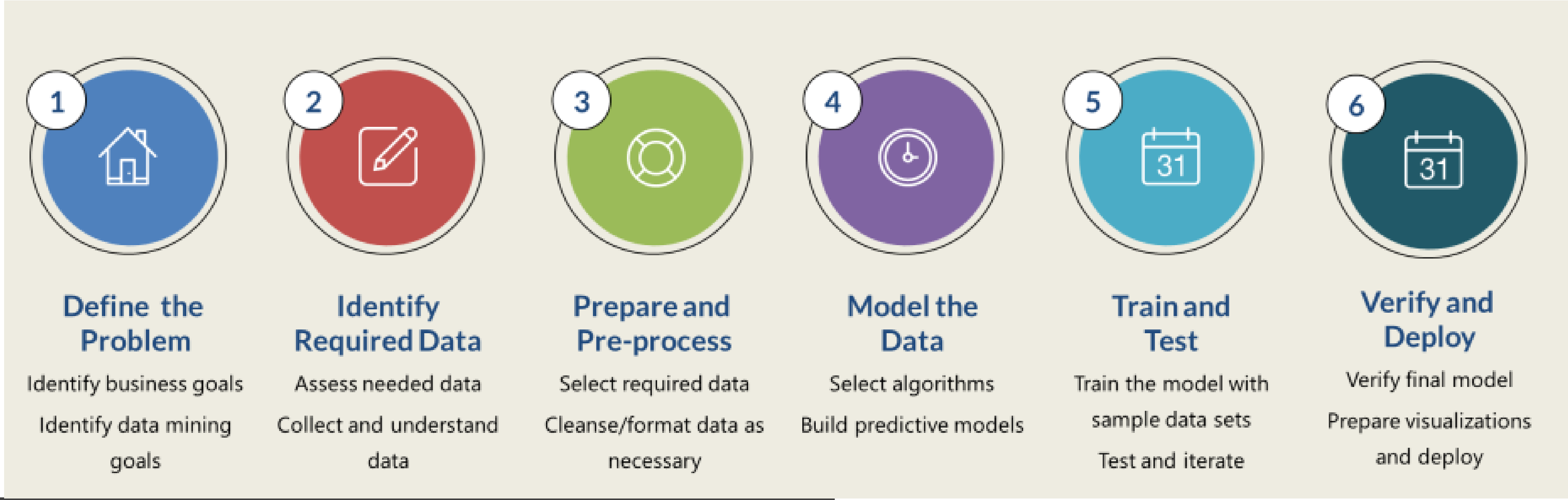
https://www.youtube.com/watch?v=EYPapE-3FRw

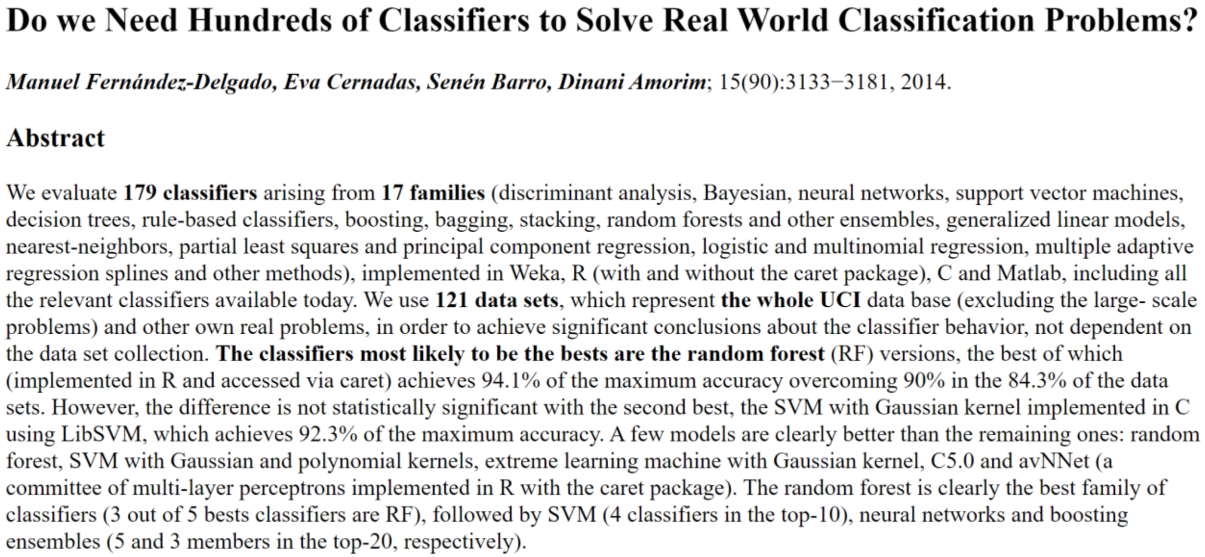
Evaluating & Comparing Models

- **Be clear about the objective of your evaluation**
  - I want to find the best parameters for my model
  - Is my model better on this data?
  - Is this classifier typically better than this other one?
  - Does this model work?
  - This model gives better sensitivity
  - My training time is better than yours
  - The classifier is overfitting/ poor at generalization
  - This model is particular suited for high dimensional data
  - These features work better than these other features
  - When to stop learning?
  - Etc.

This model does not work because …

It fails to capture the structure of the data
It fails to work on discrete data
The data is not linearly separable
The amount of training data is small
The data is imbalanced
The training data is noisy
The test data does not follow the same distribution as the training data
The underlying assumptions of this classifier need revision
The **optimization** algorithm failed
The **representation** is improper
The **evaluation** strategy presented in the paper by Lay man et al. is wrong



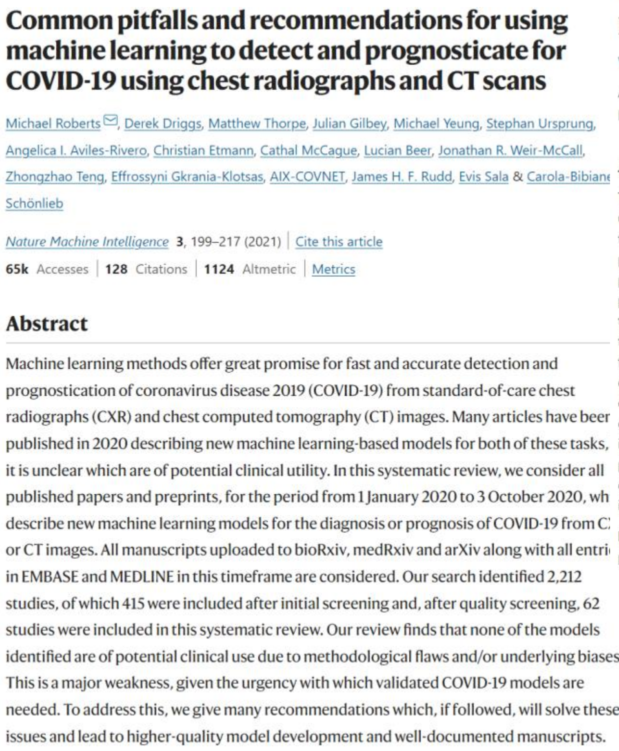


Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?

Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, Dinani Amorim; 15(90):3133−3181, 2014.

Abstract

We evaluate **179 classifiers** arising from **17 families** (discriminant analysis, Bayesian, neural networks, support vector machines, decision trees, rule-based classifiers, boosting, bagging, stacking, random forests and other ensembles, generalized linear models, nearest-neighbors, partial least squares and principal component regression, logistic and multinomial regression, multiple adaptive regression splines and other methods), implemented in Weka, R (with and without the caret package), C and Matlab, including all the relevant classifiers available today. We use **121 data sets**, which represent **the whole UCI** data base (excluding the large- scale problems) and other own real problems, in order to achieve significant conclusions about the classifier behavior, not dependent on the data set collection. **The classifiers most likely to be the bests are the random forest** (RF) versions, the best of which (implemented in R and accessed via caret) achieves 94.1% of the maximum accuracy overcoming 90% in the 84.3% of the data sets. However, the difference is not statistically significant with the second best, the SVM with Gaussian kernel implemented in C using LibSVM, which achieves 92.3% of the maximum accuracy. A few models are clearly better than the remaining ones: random forest, SVM with Gaussian and polynomial kernels, extreme learning machine with Gaussian kernel, C5.0 and avNNet (a committee of multi-layer perceptrons implemented in R with the caret package). The random forest is clearly the best family of classifiers (3 out of 5 bests classifiers are RF), followed by SVM (4 classifiers in the top-10), neural networks and boosting ensembles (5 and 3 members in the top-20, respectively).



Common pitfalls and recommendations for using machine learning to detect and prognosticate for COVID-19 using chest radiographs and CT scans

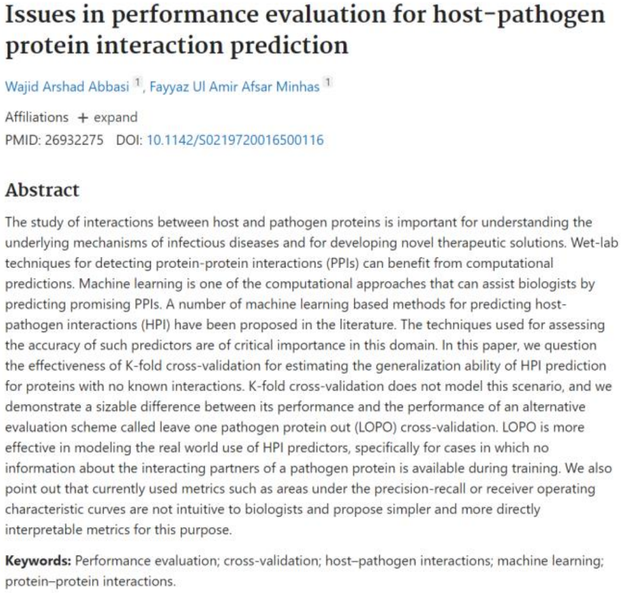
Michael Roberts, Derek Driggs, Matthew Thorpe, Julian Gilbey, Michael Yeung, Stephan Ursprung, Angelica I. Aviles-Rivero, Christian Etmann, Cathal McCague, Lucian Beer, Jonathan R. Weir-McCall, Zhongzhao Teng, Effrossyni Gkrania-Klotsas, AIX-COVNET, James H. F. Rudd, Evis Sala & Carola-Bibiane Schönlieb

Nature Machine Intelligence **3**, 199–217 (2021) | Cite this article

65k Accesses | **128** Citations | **1124** Altmetric | Metrics

Abstract

Machine learning methods offer great promise for fast and accurate detection and prognostication of coronavirus disease 2019 (COVID-19) from standard-of-care chest radiographs (CXR) and chest computed tomography (CT) images. Many articles have beer published in 2020 describing new machine learning-based models for both of these tasks, it is unclear which are of potential clinical utility. In this systematic review, we consider all published papers and preprints, for the period from 1 January 2020 to 3 October 2020, wh describe new machine learning models for the diagnosis or prognosis of COVID-19 from C or CT images. All manuscripts uploaded to bioRxiv, medRxiv and arXiv along with all entri in EMBASE and MEDLINE in this timeframe are considered. Our search identified 2,212 studies, of which 415 were included after initial screening and, after quality screening, 62 studies were included in this systematic review. Our review finds that none of the models identified are of potential clinical use due to methodological flaws and/or underlying biases. This is a major weakness, given the urgency with which validated COVID-19 models are needed. To address this, we give many recommendations which, if followed, will solve these issues and lead to higher-quality model development and well-documented manuscripts.



Issues in performance evaluation for host–pathogen protein interaction prediction

Wajid Arshad Abbasi [1], Fayyaz Ul Amir Afsar Minhas [1]

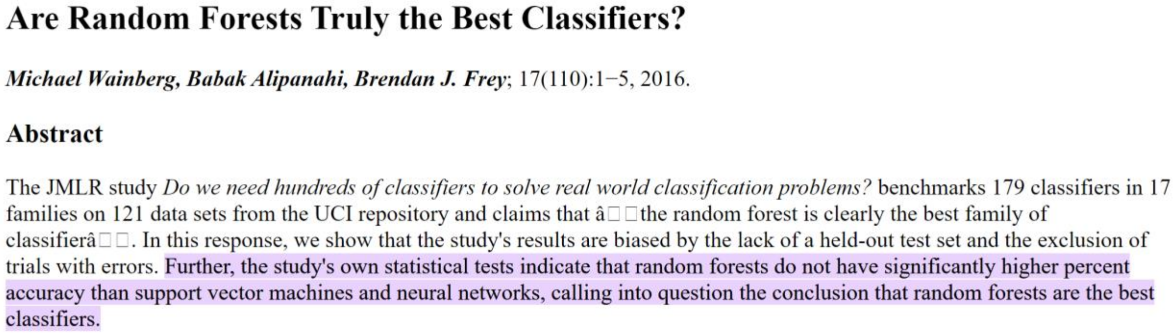
Affiliations + expand

PMID: 26932275 DOI: 10.1142/S0219720016500116

Abstract

The study of interactions between host and pathogen proteins is important for understanding the underlying mechanisms of infectious diseases and for developing novel therapeutic solutions. Wet-lab techniques for detecting protein-protein interactions (PPIs) can benefit from computational predictions. Machine learning is one of the computational approaches that can assist biologists by predicting promising PPIs. A number of machine learning based methods for predicting host-pathogen interactions (HPI) have been proposed in the literature. The techniques used for assessing the accuracy of such predictors are of critical importance in this domain. In this paper, we question the effectiveness of K-fold cross-validation for estimating the generalization ability of HPI prediction for proteins with no known interactions. K-fold cross-validation does not model this scenario, and we demonstrate a sizable difference between its performance and the performance of an alternative evaluation scheme called leave one pathogen protein out (LOPO) cross-validation. LOPO is more effective in modeling the real world use of HPI predictors, specifically for cases in which no information about the interacting partners of a pathogen protein is available during training. We also point out that currently used metrics such as areas under the precision-recall or receiver operating characteristic curves are not intuitive to biologists and propose simpler and more directly interpretable metrics for this purpose.

Keywords: Performance evaluation; cross-validation; host–pathogen interactions; machine learning; protein–protein interactions.

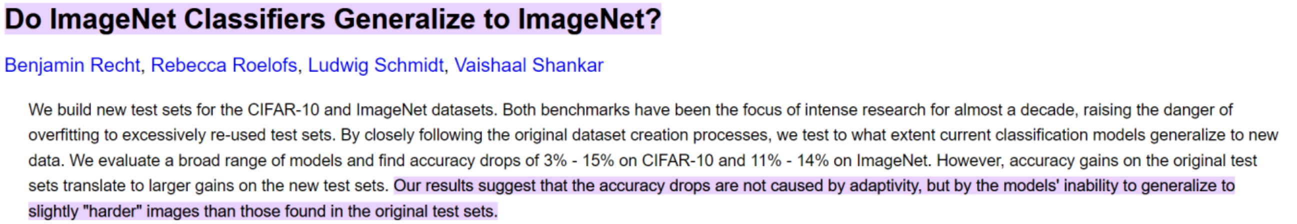


Are Random Forests Truly the Best Classifiers?

Michael Wainberg, Babak Alipanahi, Brendan J. Frey; 17(110):1−5, 2016.

Abstract

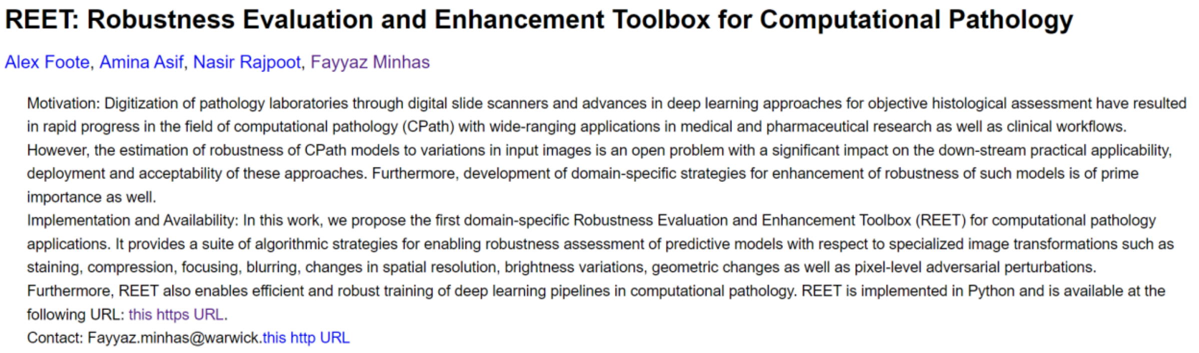
The JMLR study *Do we need hundreds of classifiers to solve real world classification problems?* benchmarks 179 classifiers in 17 families on 121 data sets from the UCI repository and claims that â□□the random forest is clearly the best family of classifierâ□□. In this response, we show that the study's results are biased by the lack of a held-out test set and the exclusion of trials with errors. Further, the study's own statistical tests indicate that random forests do not have significantly higher percent accuracy than support vector machines and neural networks, calling into question the conclusion that random forests are the best classifiers.



Do ImageNet Classifiers Generalize to ImageNet?

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, Vaishaal Shankar

We build new test sets for the CIFAR-10 and ImageNet datasets. Both benchmarks have been the focus of intense research for almost a decade, raising the danger of overfitting to excessively re-used test sets. By closely following the original dataset creation processes, we test to what extent current classification models generalize to new data. We evaluate a broad range of models and find accuracy drops of 3% - 15% on CIFAR-10 and 11% - 14% on ImageNet. However, accuracy gains on the original test sets translate to larger gains on the new test sets. Our results suggest that the accuracy drops are not caused by adaptivity, but by the models' inability to generalize to slightly "harder" images than those found in the original test sets.



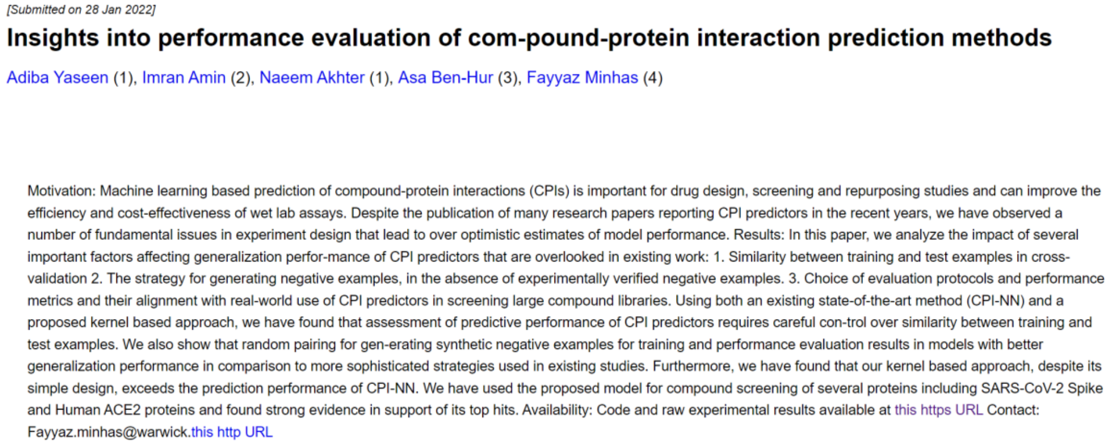
REET: Robustness Evaluation and Enhancement Toolbox for Computational Pathology

Alex Foote, Amina Asif, Nasir Rajpoot, Fayyaz Minhas

Motivation: Digitization of pathology laboratories through digital slide scanners and advances in deep learning approaches for objective histological assessment have resulted in rapid progress in the field of computational pathology (CPath) with wide-ranging applications in medical and pharmaceutical research as well as clinical workflows. However, the estimation of robustness of CPath models to variations in input images is an open problem with a significant impact on the down-stream practical applicability, deployment and acceptability of these approaches. Furthermore, development of domain-specific strategies for enhancement of robustness of such models is of prime importance as well.

Implementation and Availability: In this work, we propose the first domain-specific Robustness Evaluation and Enhancement Toolbox (REET) for computational pathology applications. It provides a suite of algorithmic strategies for enabling robustness assessment of predictive models with respect to specialized image transformations such as staining, compression, focusing, blurring, changes in spatial resolution, brightness variations, geometric changes as well as pixel-level adversarial perturbations. Furthermore, REET also enables efficient and robust training of deep learning pipelines in computational pathology. REET is implemented in Python and is available at the following URL: this https URL.

Contact: Fayyaz.minhas@warwick.this http URL



[Submitted on 28 Jan 2022]

Insights into performance evaluation of com-pound-protein interaction prediction methods

Adiba Yaseen (1), Imran Amin (2), Naeem Akhter (1), Asa Ben-Hur (3), Fayyaz Minhas (4)

Motivation: Machine learning based prediction of compound-protein interactions (CPIs) is important for drug design, screening and repurposing studies and can improve the efficiency and cost-effectiveness of wet lab assays. Despite the publication of many research papers reporting CPI predictors in the recent years, we have observed a number of fundamental issues in experiment design that lead to over optimistic estimates of model performance. Results: In this paper, we analyze the impact of several important factors affecting generalization perfor-mance of CPI predictors that are overlooked in existing work: 1. Similarity between training and test examples in cross-validation 2. The strategy for generating negative examples, in the absence of experimentally verified negative examples. 3. Choice of evaluation protocols and performance metrics and their alignment with real-world use of CPI predictors in screening large compound libraries. Using both an existing state-of-the-art method (CPI-NN) and a proposed kernel based approach, we have found that assessment of predictive performance of CPI predictors requires careful con-trol over similarity between training and test examples. We also show that random pairing for gen-erating synthetic negative examples for training and performance evaluation results in models with better generalization performance in comparison to more sophisticated strategies used in existing studies. Furthermore, we have found that our kernel based approach, despite its simple design, exceeds the prediction performance of CPI-NN. We have used the proposed model for compound screening of several proteins including SARS-CoV-2 Spike and Human ACE2 proteins and found strong evidence in support of its top hits. Availability: Code and raw experimental results available at this https URL Contact: Fayyaz.minhas@warwick.this http URL

Experiment Design Objectives

- Accuracy Evaluation: How good will it be in practice?
- Sensitivity Analysis: Is the classifier sensitive to
  - the choice of the parameters so much so that it will be useless in practice
  - choice of the data
  - Randomness
  - Round-off error
  - Other controllable and non-controllable factors

Evaluation Metrics

- **Training Set:** For training the model
- **Test Set:** For evaluation
- *Under no circumstances are testing labels to be used in training or the training data in evaluation of the generalization performance*
- All evaluation metrics have underlying assumptions and limitations which may or may not suffice for the test that you are trying to perform

Definitions

- True Positive
- True Negative
- False Positive
- False Negative

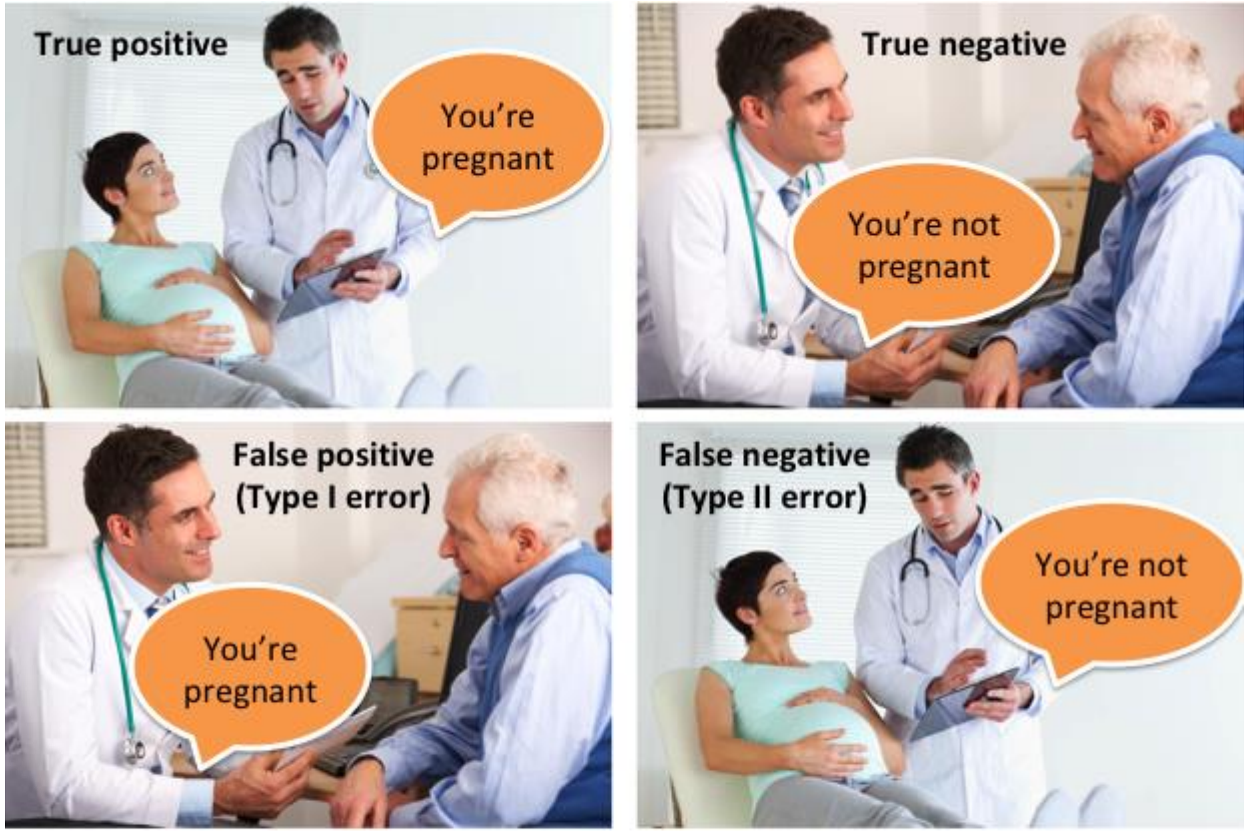


Image from https://neeraj-kumar-vaid.medium.com/statistical-performance-measures-12bad66694b7

Accuracy

- Two-Class Classification
  - Accuracy: Percentage of Correct Predictions

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

- Assumption
  - The data set is balanced
  - Misclassification of any class is equally bad
  - The threshold used for classification is optimal

Classification Performance

- A classifier (or any machine learning model) can be viewed as a function $y = f(x|\theta)$ which generates an output y given the input x and a parameter set θ using a decision function $f(x|\theta)$
- The output of a classifier is typically a real-valued output which is then thresholded to yield classification labels

$$f(x|\theta) > 0 \ \Rightarrow y = +1$$
$$f(x|\theta) < 0 \ \Rightarrow y = -1$$

- Here "0" acts as the threshold
- Thus, the labels can change based on the threshold
- Thus, accuracy of a classifier is parametrized by this threshold

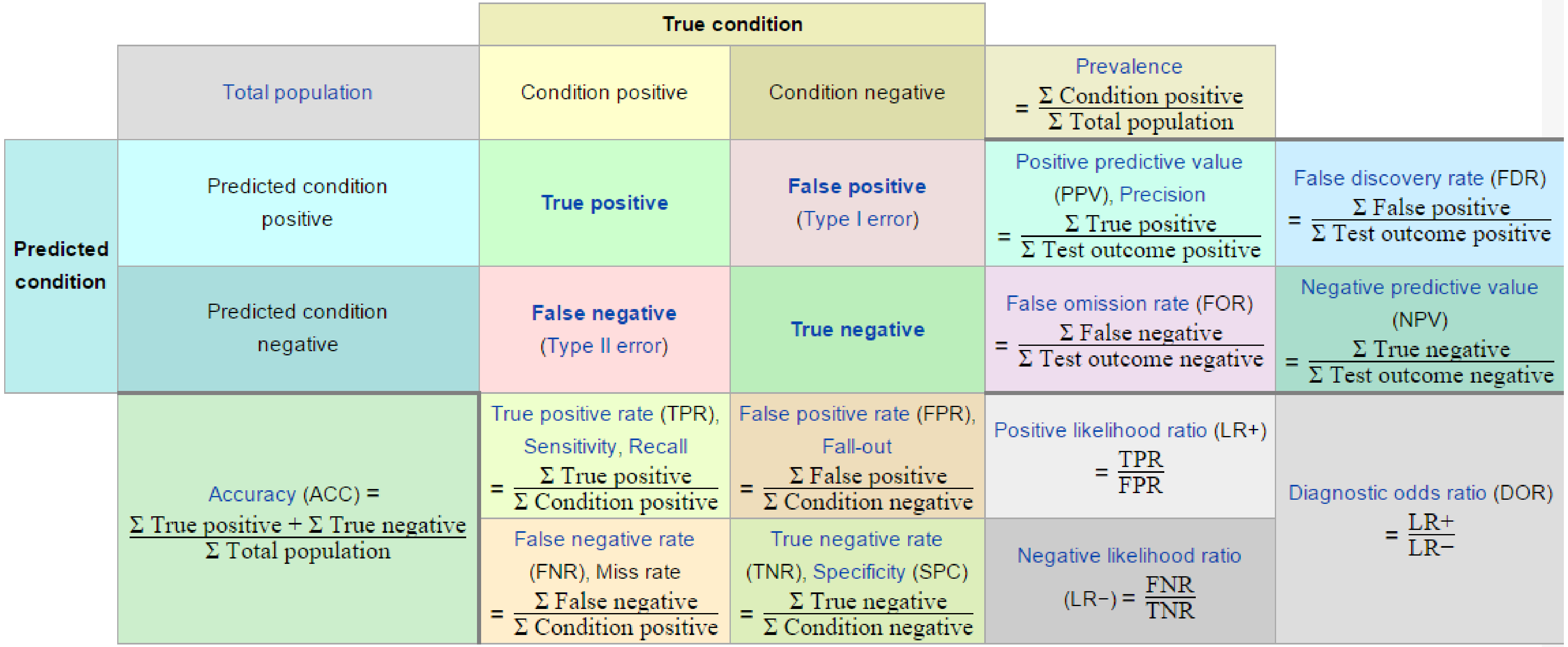
Thought Experiment

- Consider the data
  - Assume that the data is balanced (equal number of positive and negative examples)
  - Consider a random classifier
    - This classifier will generate a random score of any example given as input
    - **What will be its accuracy?**
  - Consider a classifier which generates a score of +1 for all inputs
    - **What will be its accuracy?**

Thought Experiment

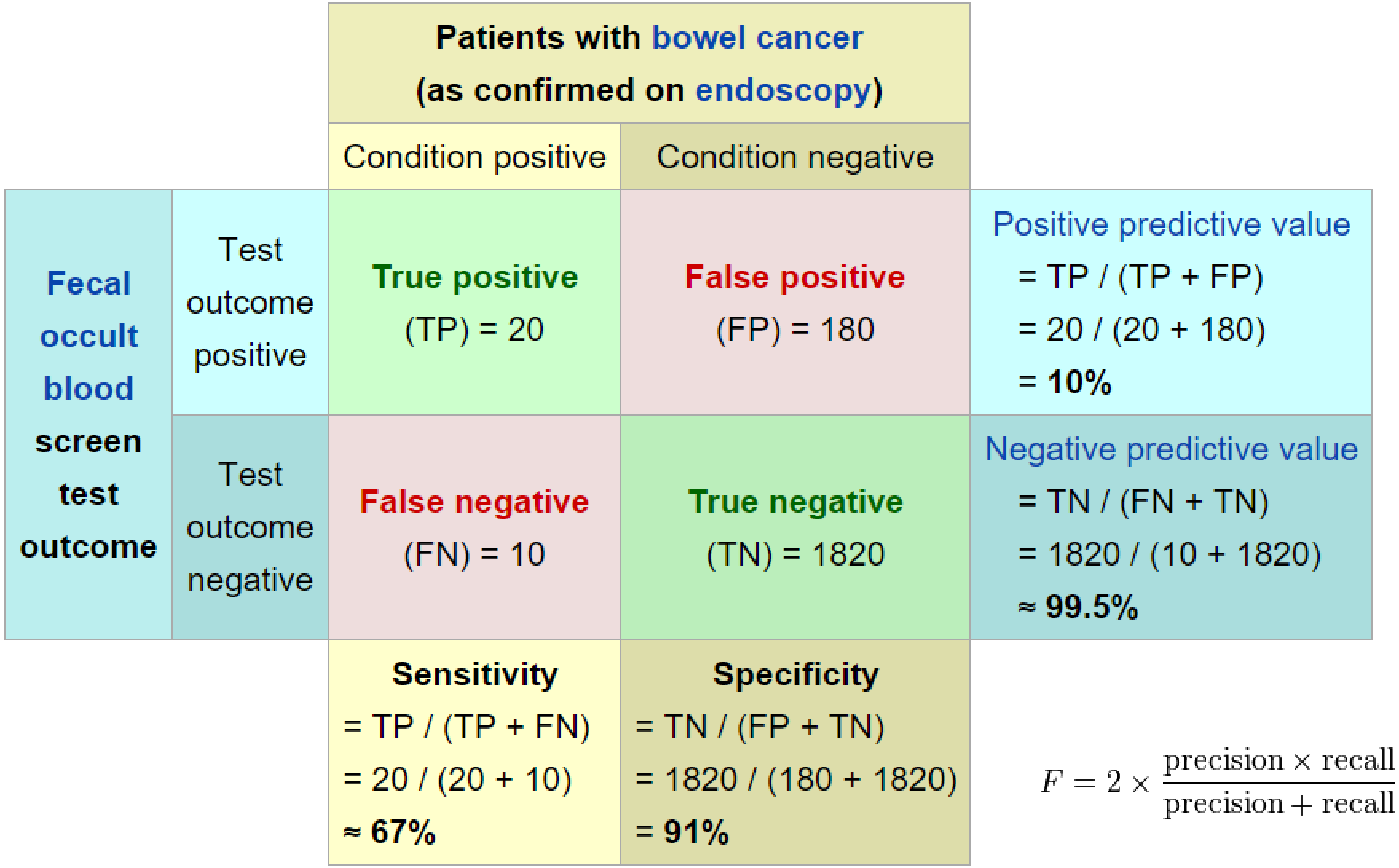
- Consider the data
  - Assume that the data is imbalanced (#Neg>>#Pos)
  - Consider a random classifier
    - This classifier will generate a random score of any example given as input
    - **What will be its accuracy?**
  - Consider a classifier which generates a score of +1 for all inputs
    - **What will be its accuracy?**

Confusion Matrix



		True condition			
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive** (Type I error)	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$	False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Test outcome positive}}$
	Predicted condition negative	**False negative** (Type II error)	**True negative**	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Test outcome negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Test outcome negative}}$
	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	True positive rate (TPR), Sensitivity, Recall $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	True negative rate (TNR), Specificity (SPC) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$	

https://en.wikipedia.org/wiki/Sensitivity_and_specificity



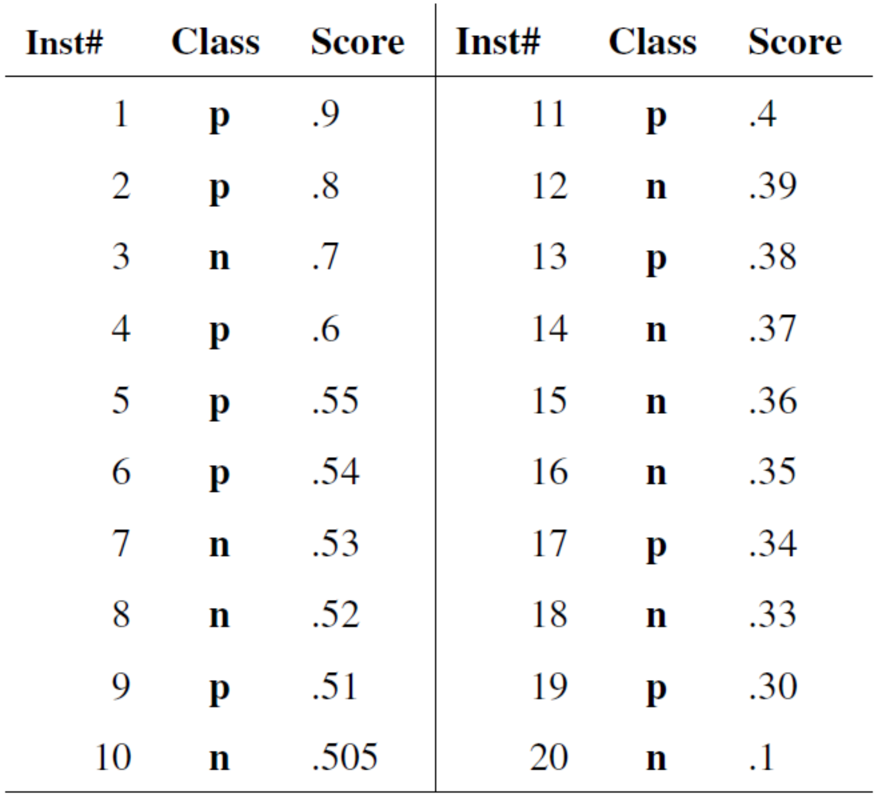
		Patients with bowel cancer **(as confirmed on endoscopy)**		
		Condition positive	Condition negative	
Fecal occult blood screen test outcome	Test outcome positive	**True positive** (TP) = 20	**False positive** (FP) = 180	Positive predictive value = TP / (TP + FP) = 20 / (20 + 180) = **10%**
	Test outcome negative	**False negative** (FN) = 10	**True negative** (TN) = 1820	Negative predictive value = TN / (FN + TN) = 1820 / (10 + 1820) ≈ **99.5%**
		Sensitivity = TP / (TP + FN) = 20 / (20 + 10) ≈ **67%**	**Specificity** = TN / (FP + TN) = 1820 / (180 + 1820) = **91%**	

$$F = 2 \times \frac{\mathrm{precision} \times \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

https://en.wikipedia.org/wiki/Sensitivity_and_specificity

Output of a binary classifier

- At a threshold of 0.5, can you calculate:
  - Accuracy
  - Sensitivity
  - Specificity
  - FPR
  - TPR
  - Precision
  - F-score



Inst#	Class	Score	Inst#	Class	Score
1	**p**	.9	11	**p**	.4
2	**p**	.8	12	**n**	.39
3	**n**	.7	13	**p**	.38
4	**p**	.6	14	**n**	.37
5	**p**	.55	15	**n**	.36
6	**p**	.54	16	**n**	.35
7	**n**	.53	17	**p**	.34
8	**n**	.52	18	**n**	.33
9	**p**	.51	19	**p**	.30
10	**n**	.505	20	**n**	.1

See: https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics

Role of threshold

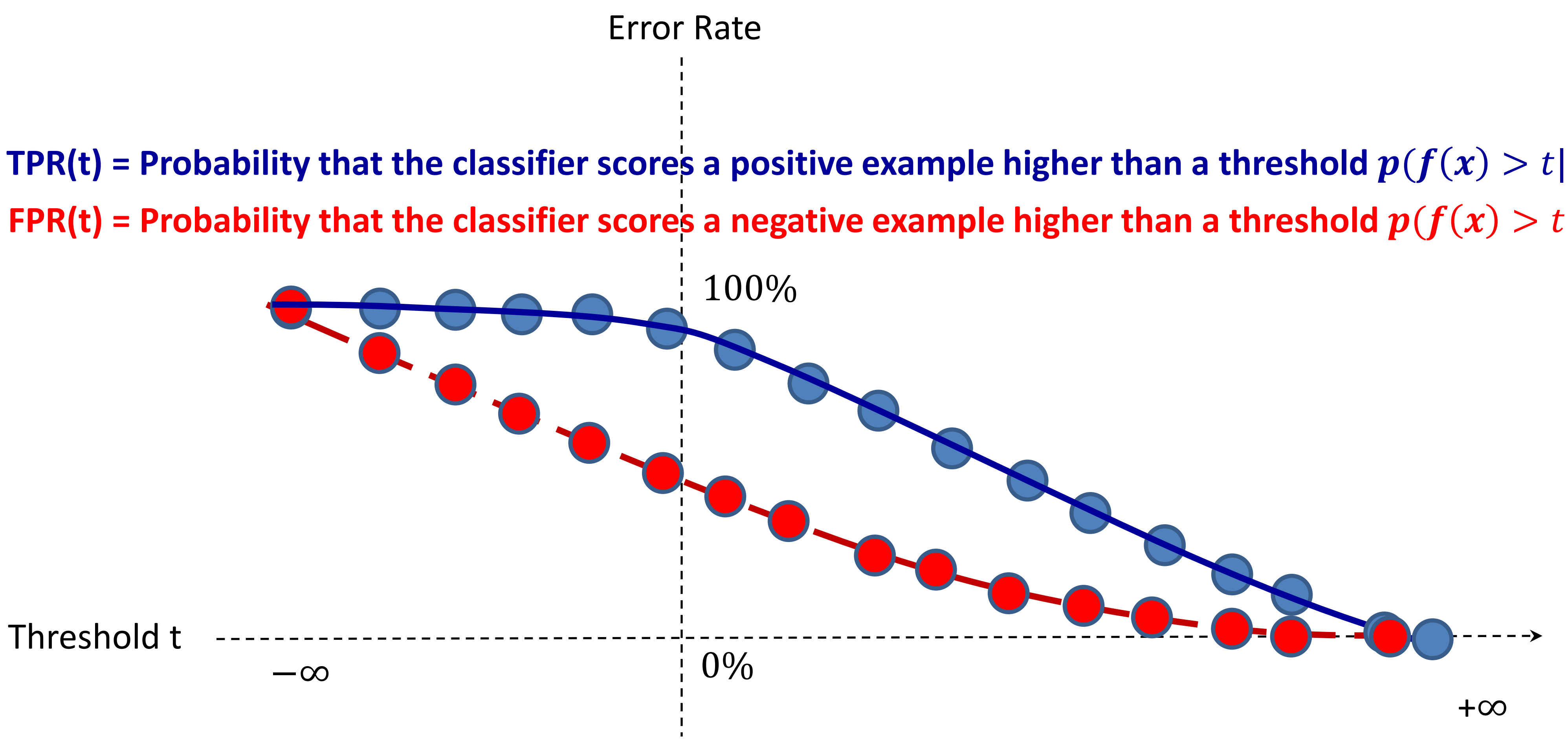
- What will be the behavior of **TPR** with increase in threshold of the classifier?

- How will **FPR** behave?

- How will **Precision** behave?

- Can **TPR** decrease with increase in threshold?

FPR vs. TPR Curve

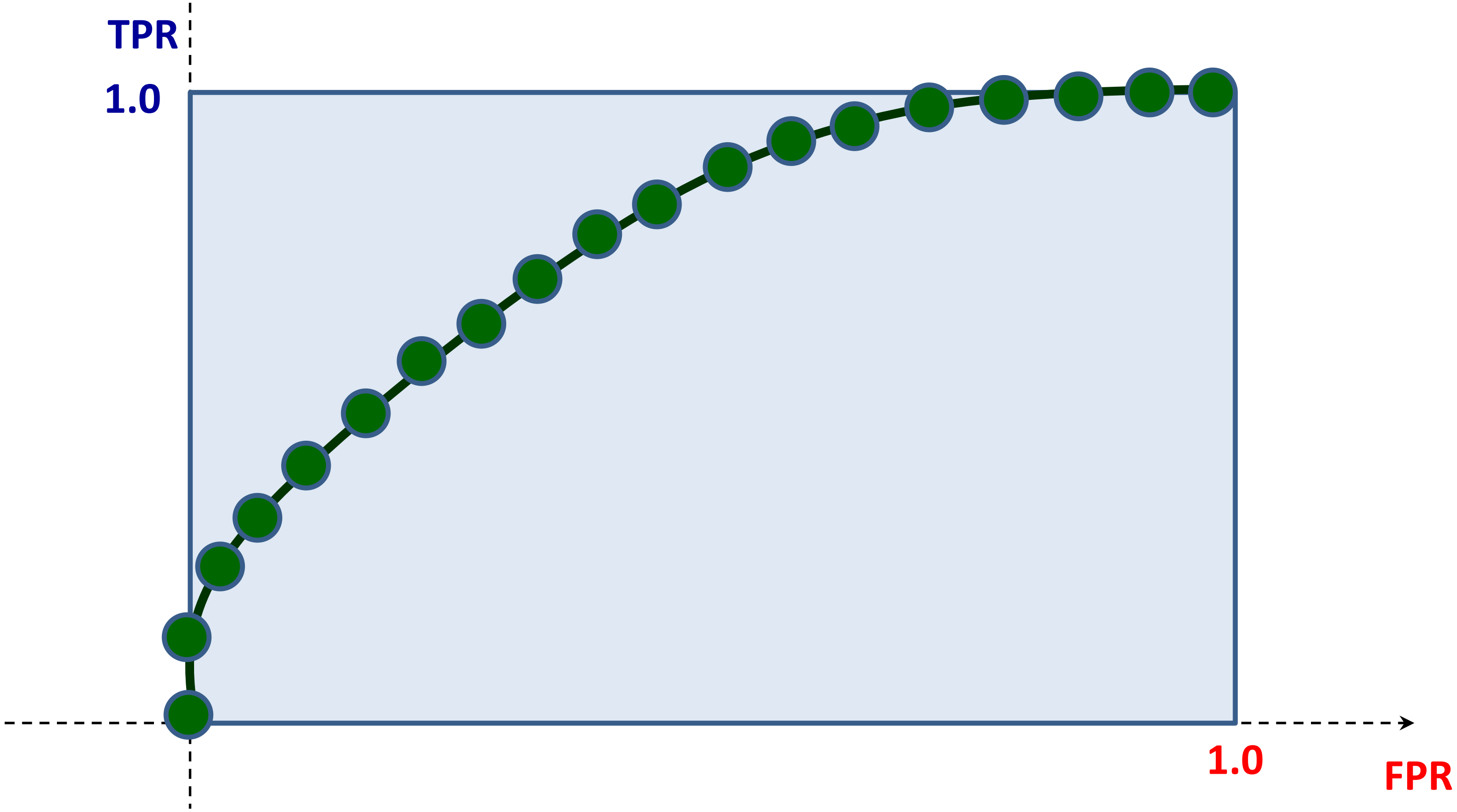


TPR(t) = Probability that the classifier scores a positive example higher than a threshold $p(f(x) > t|y = 1)$

FPR(t) = Probability that the classifier scores a negative example higher than a threshold $p(f(x) > t|y = -1)$

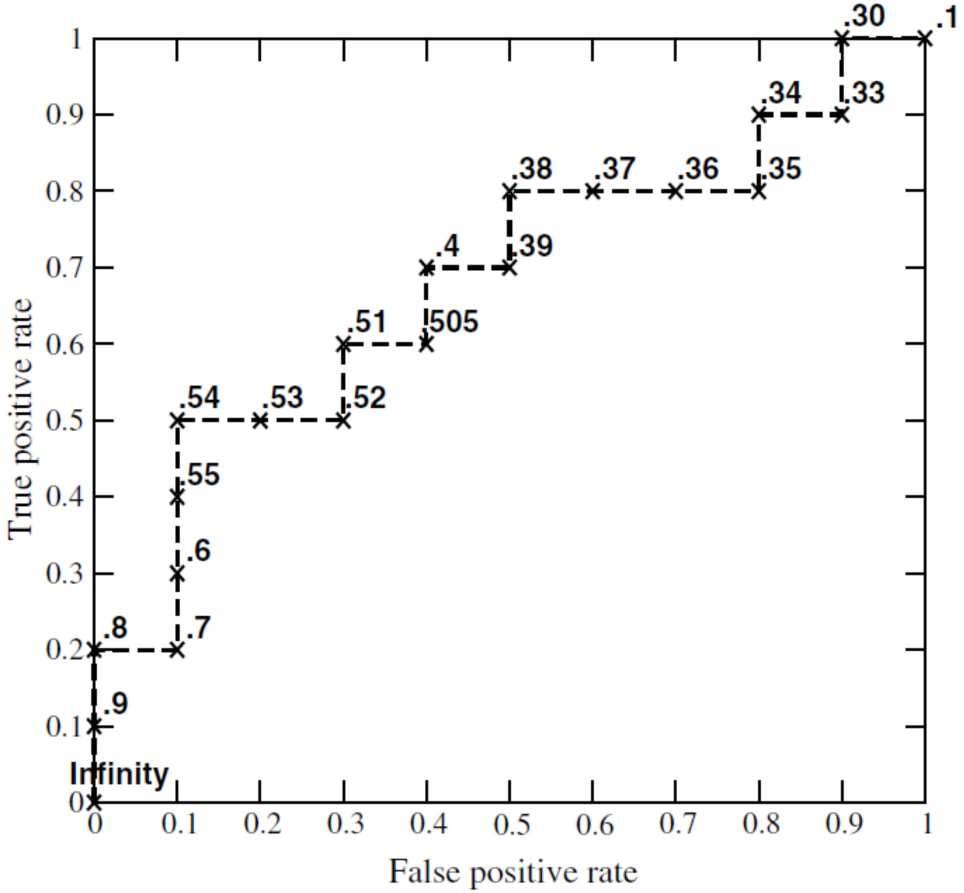
Receiver Operating Characteristics Curve

- A plot of TPR vs FPR

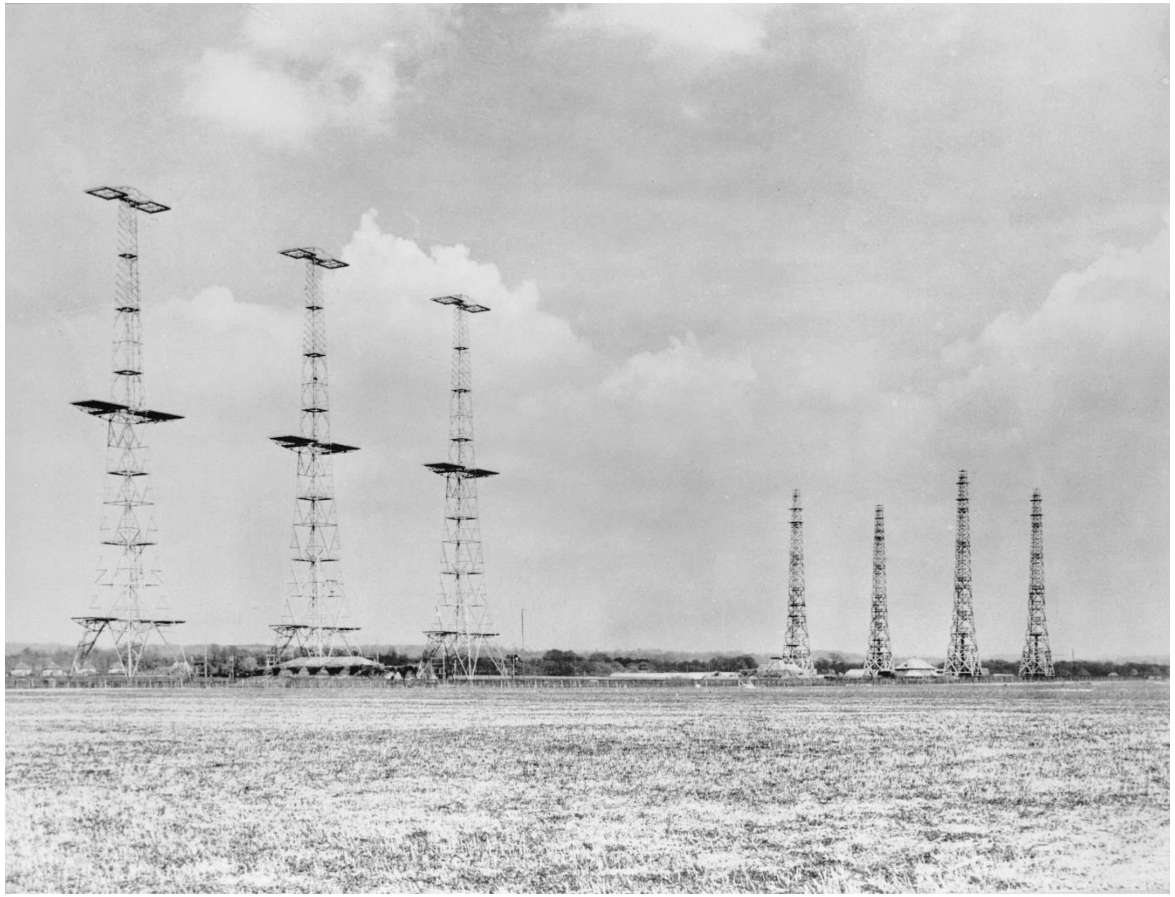


Making the ROC Curve

Inst#	Class	Score	Inst#	Class	Score
1	**p**	.9	11	**p**	.4
2	**p**	.8	12	**n**	.39
3	**n**	.7	13	**p**	.38
4	**p**	.6	14	**n**	.37
5	**p**	.55	15	**n**	.36
6	**p**	.54	16	**n**	.35
7	**n**	.53	17	**p**	.34
8	**n**	.52	18	**n**	.33
9	**p**	.51	19	**p**	.30
10	**n**	.505	20	**n**	.1



Origin of the ROC



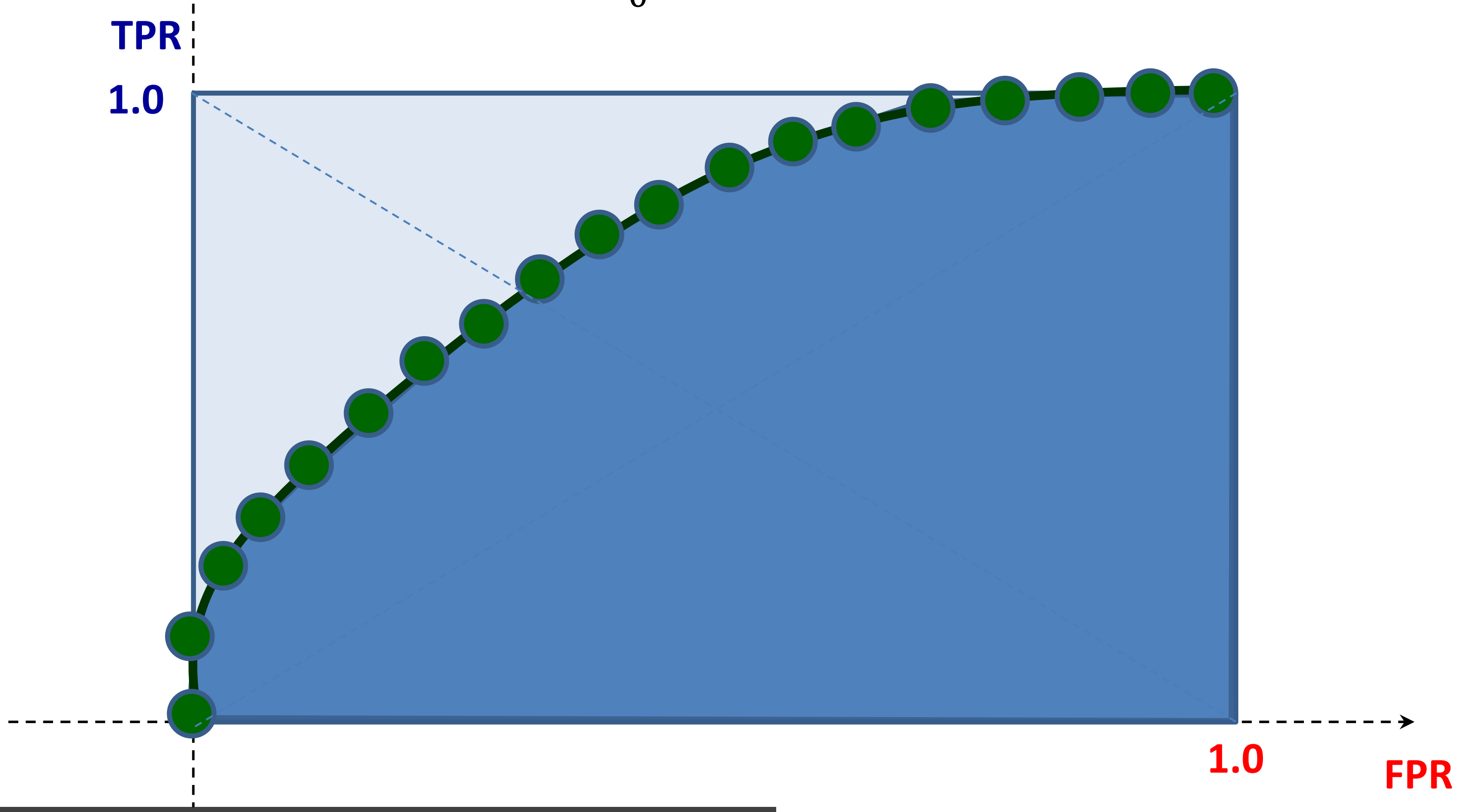
ROC

- What will be ROC curve for a perfect classifier?
- What will the ROC Curve of a random classifier look like?
- What will the ROC curve of a classifier that always predicts the positive class look like?
- What are the underlying assumptions of the ROC curve?
- What part of the ROC curve is the most important?

ROC and AUC-ROC

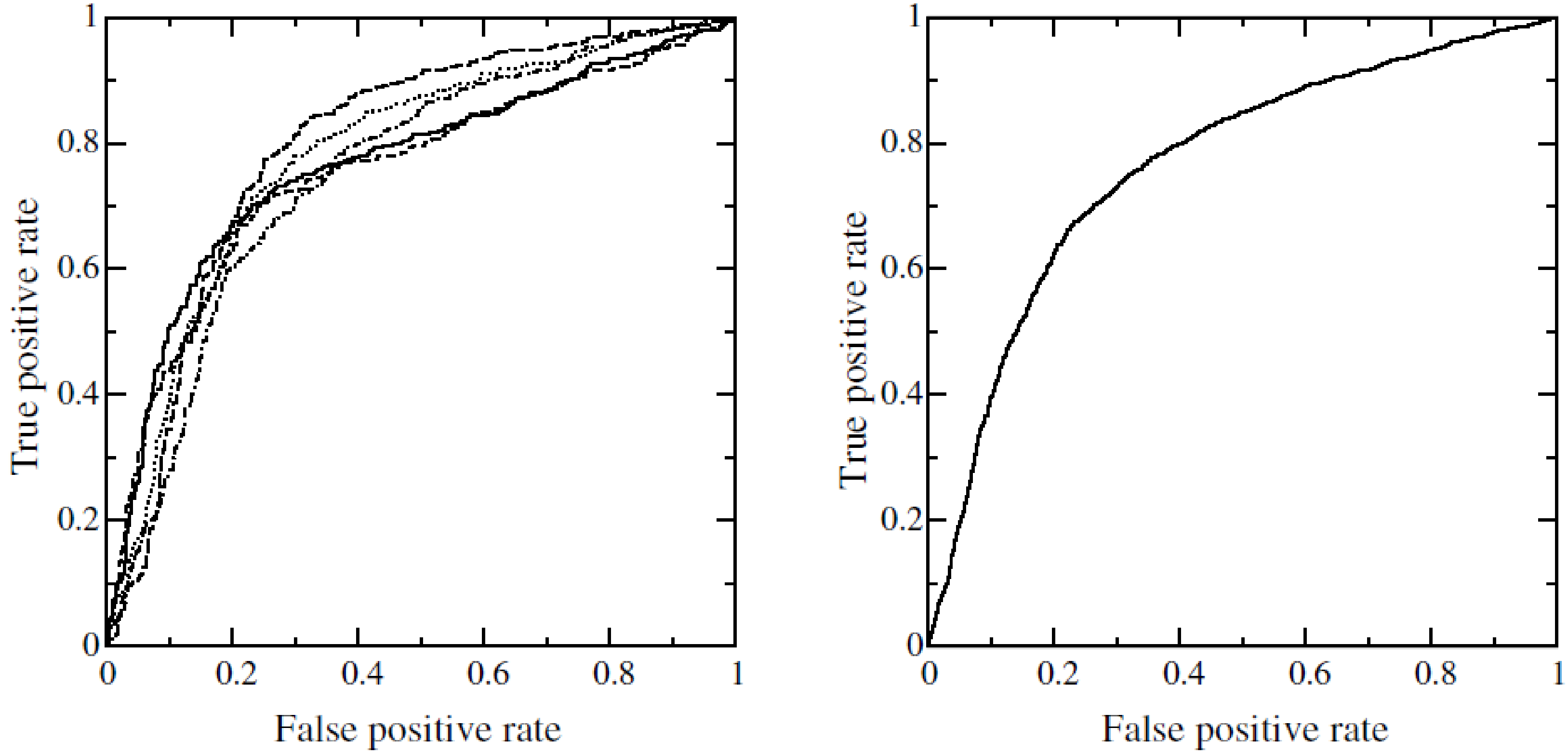
- The area under the ROC curve is a quality metric

$$AUC - ROC = \int_{0}^{1} TPR\left(FPR^{-1}(u)\right)du$$



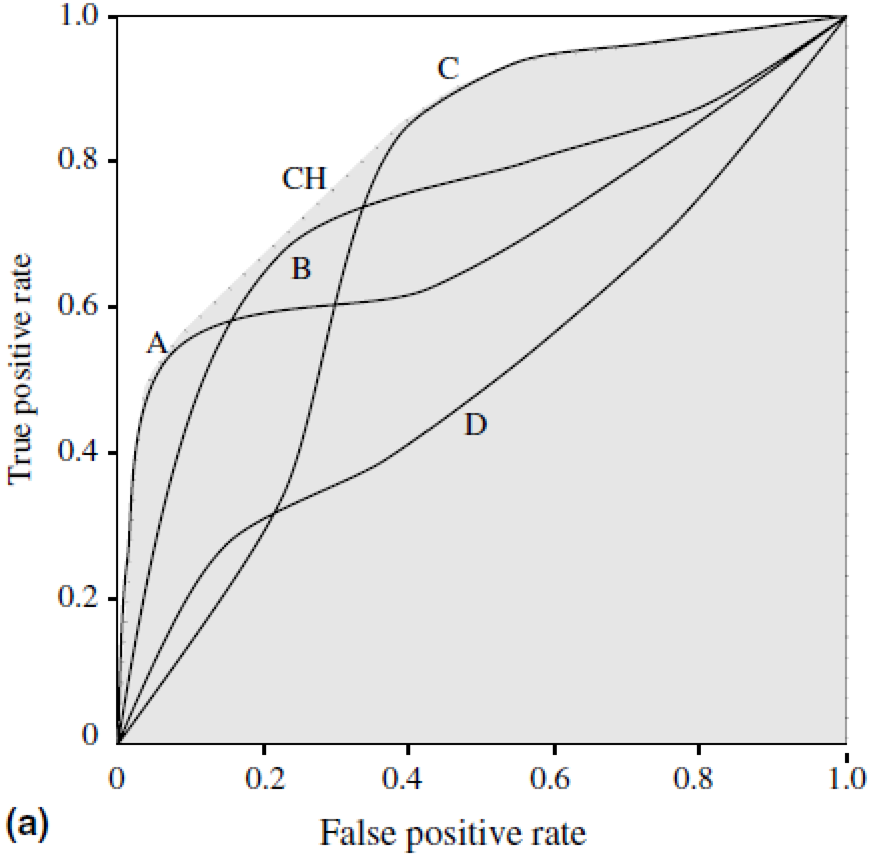
Averaging ROC Curves

- ROC curves can be vertically averaged



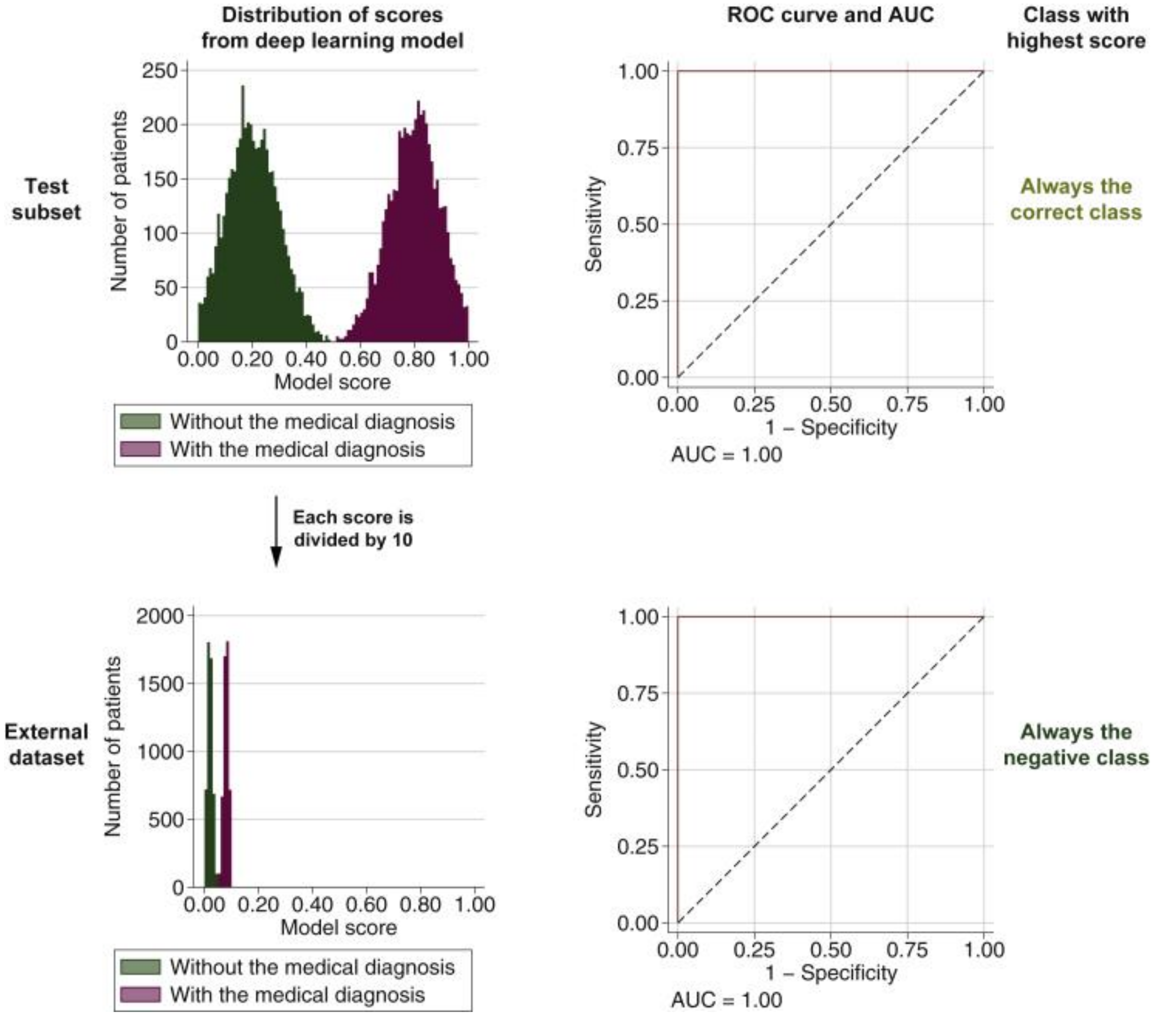
ROC Convex Hull

- Scores of two classifiers can be combined through a weighted combination to result in an optimal classifier
- This can be done using the ROC convex hull



ROC Properties

- **What does AUC-ROC measure?**
  - The probability with which a randomly-chosen positive example is ranked more highly than a randomly-chosen negative example by a classifier [1]
  - This also allows us to generalize this idea to “paired evaluation” [1]
- When to Use?
  - When there isn’t much class imbalance
  - To compare classifiers across all operating points
- Caveats
  - Class imblanace
  - Multi-class ROC curves: Watch out for Imbalance
  - LHS of the plot matters more
  - Threshold selection: Threshold on one dataset may not work for another (see image on the right). ROC curves are not affected by calibration of the classifier.
- Recommendations
  - Always plot histogram of prediction scores to identify potential bias
  - Know its limitations

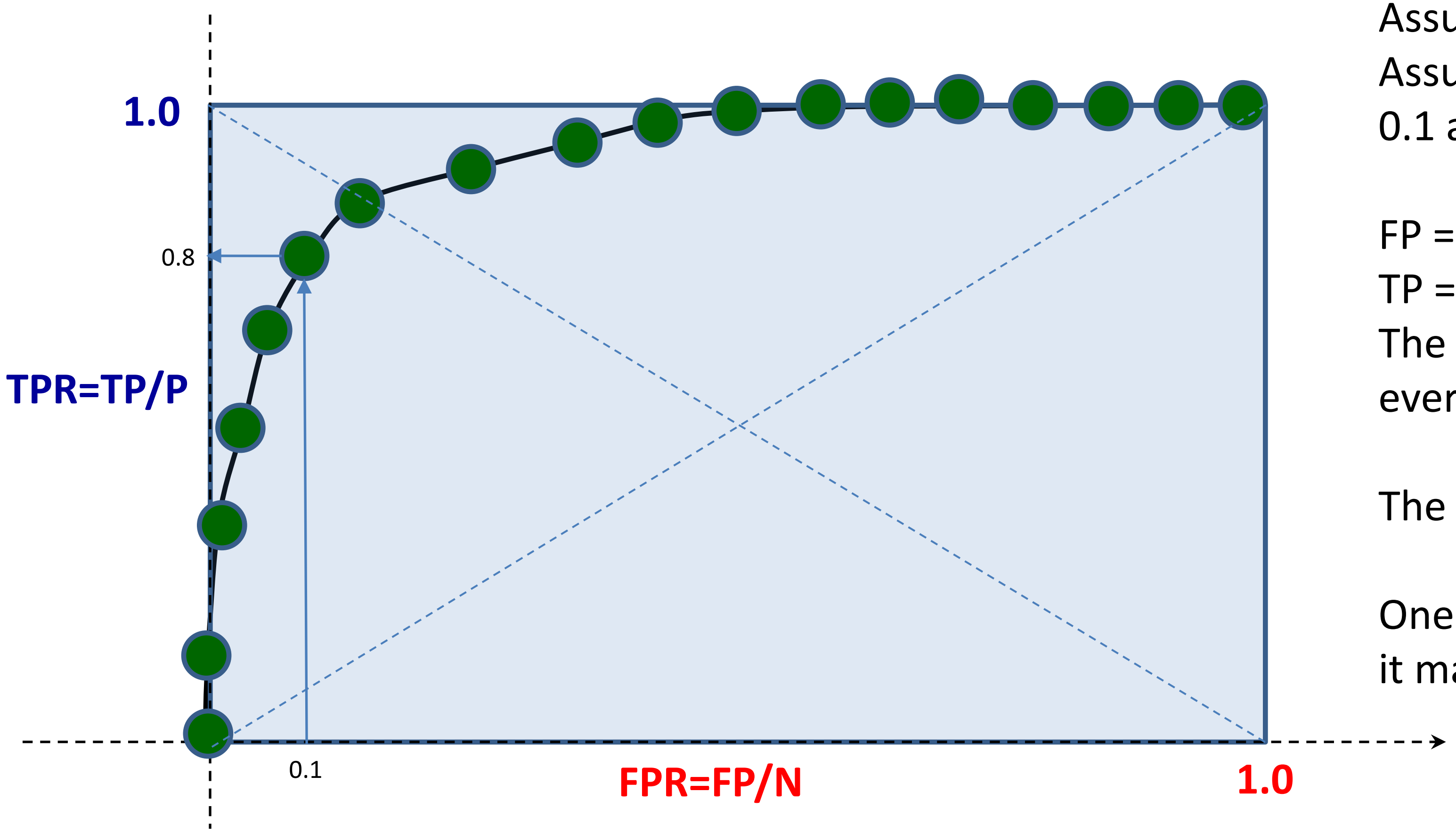


A method that can give a high AUC-ROC on a validation or test subset may still generate inaccurate labels depending upon the threshold.

See: Kleppe, A. “Area under the Curve May Hide Poor Generalisation to External Datasets.” *ESMO Open* 7, no. 2 (April 2022): 100429. https://doi.org/10.1016/j.esmoop.2022.100429.

[1] Nariya, Maulik K., Caitlin E. Mills, Peter K. Sorger, and Artem Sokolov. “Paired Evaluation of Machine-Learning Models Characterizes Effects of Confounders and Outliers.” *Patterns* 0, no. 0 (July 7, 2023). https://doi.org/10.1016/j.patter.2023.100791.

Interpreting ROC curves when N>>P



A small FPR signifies a significant number of FPs when N>>P

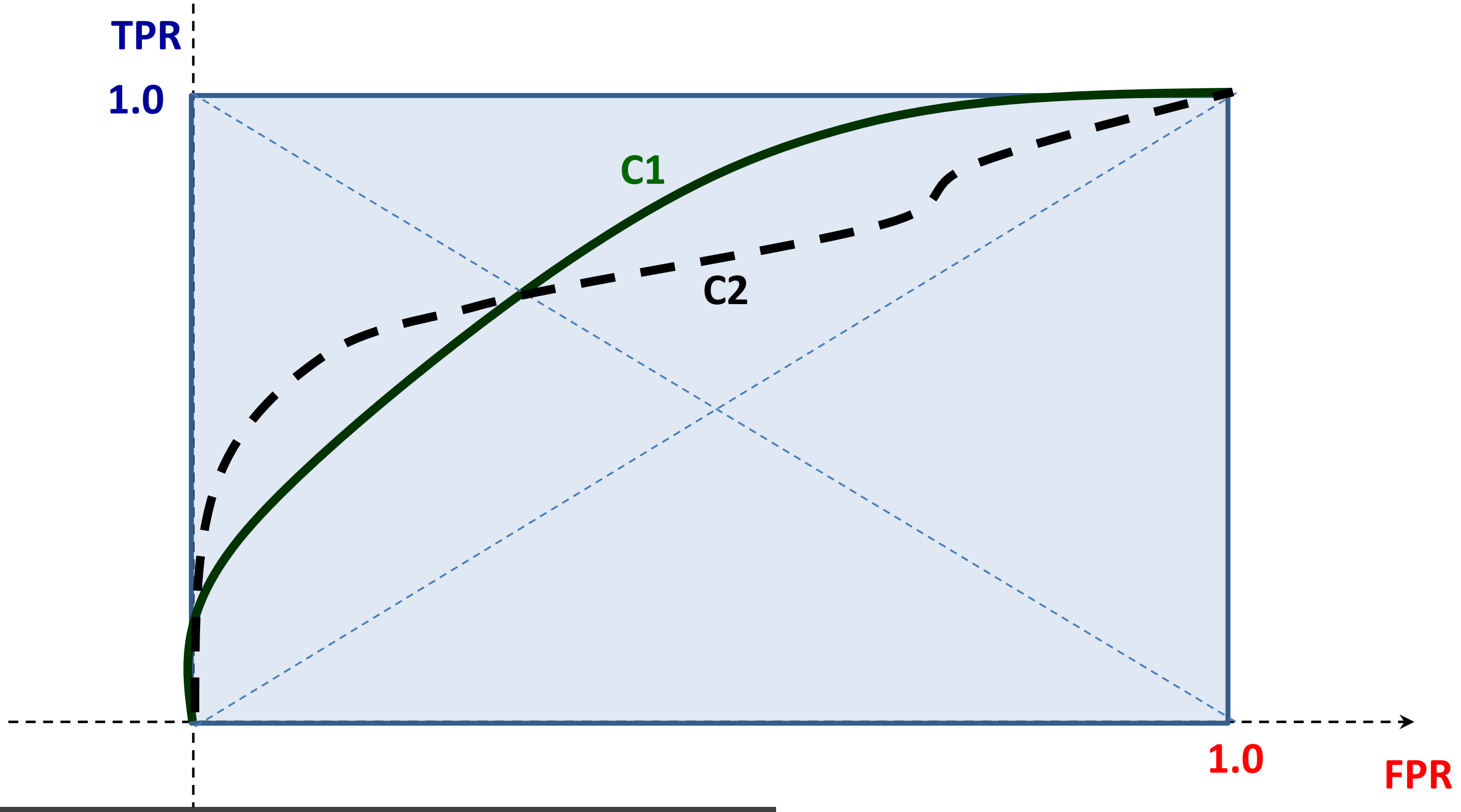
Assume P = 100, N = 10,000
Assume that at a certain threshold you get FPR = 0.1 and TPR of 0.8, i.e., TPR(FPR=0.1) = 0.8

FP = 1,000
TP = 80
The classifier is generating 80 true positives for every 1,000 false positives.

The classifier is not very precise.

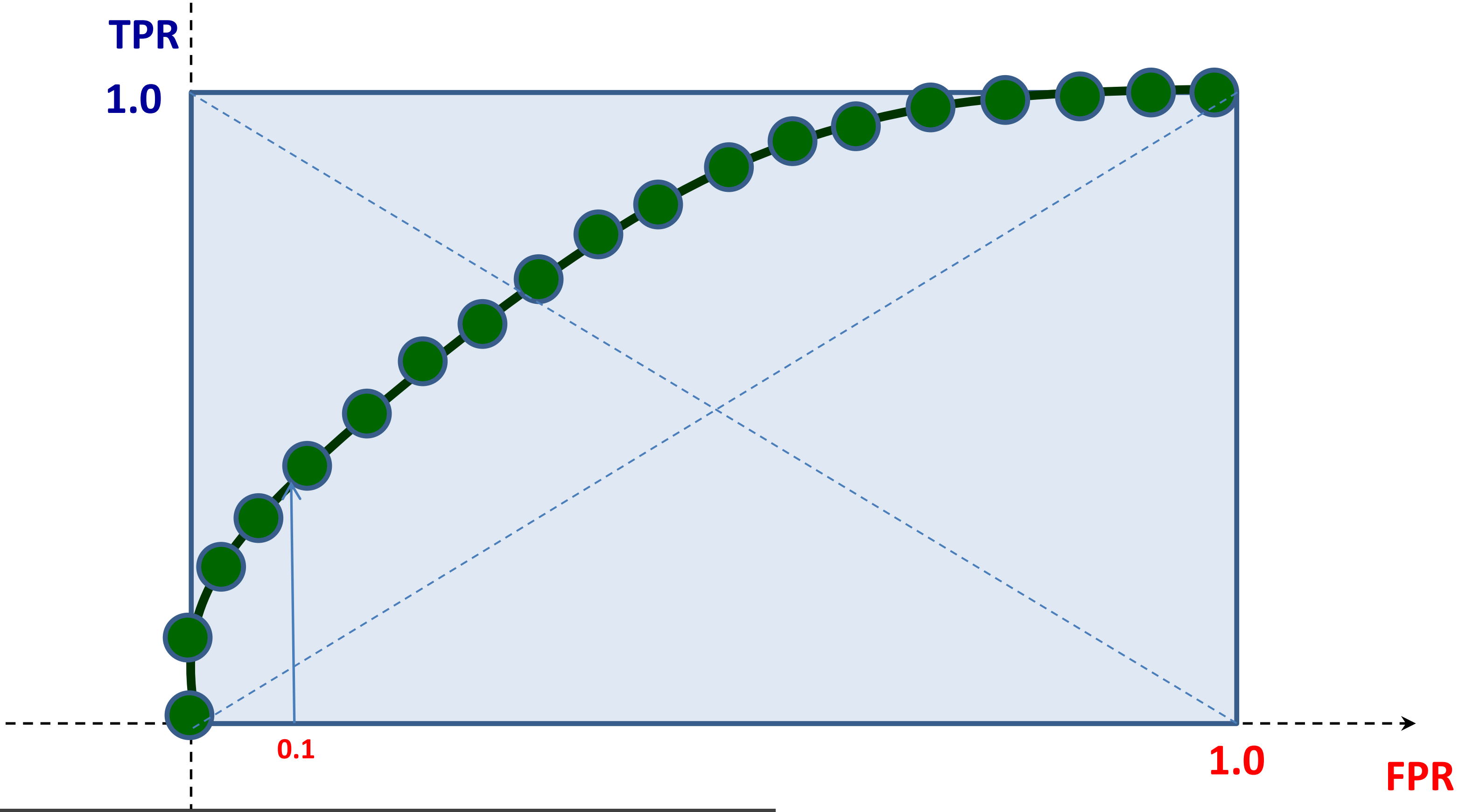
One may get high AUC-ROC but, in case of N>>P, it may not be a good metric.

Which is better?



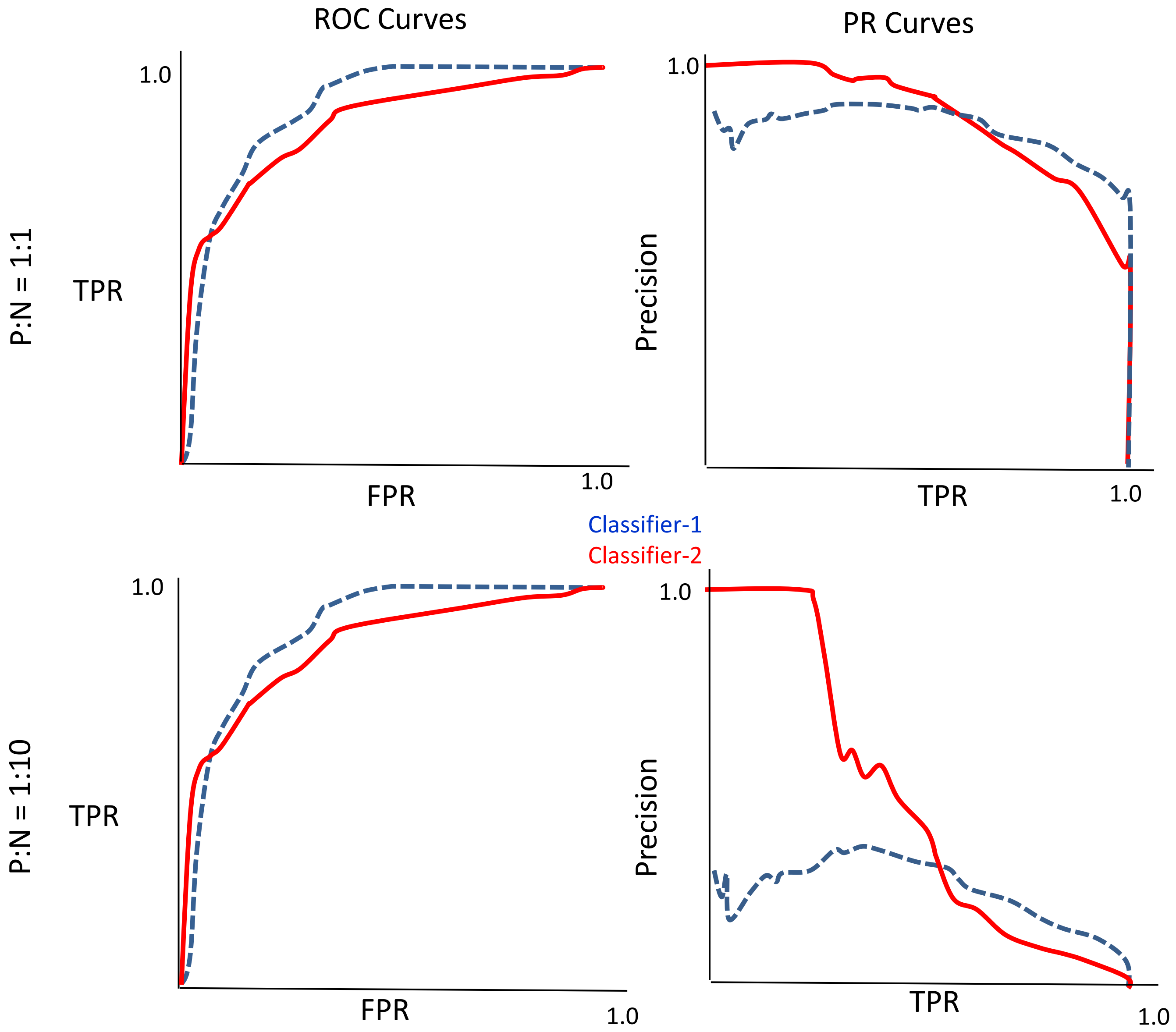
AUC ROC-N

- In cases with class imbalance or where high false positive rates are unacceptable or it is useless to evaluate TPR at high FPR values, we can use AUC-ROC N
- Area under the ROC curve up to the first N False Positives
  - N = 50
  - N = 10%



Precision-Recall Curves

- Plot of Precision vs. Recall
- AUC-PR is a performance metric
- Useful in cases of class-imbalance or in which precision is a requirement



Relationship between ROC & PR Curves

- One-to-One correspondence between the two curves
- If a curve dominates in ROC space then it dominates in PR space.
- If a curve dominates in PR space then it dominates in ROC space.

- What will be the PR curve for a random classifier?
- What part of an ROC curve impacts the PR curve more?

Reading

- Recommended
  - Davis, Jesse, and Mark Goadrich. 2006. “The Relationship Between Precision-Recall and ROC Curves.” In *Proceedings of the 23rd International Conference on Machine Learning*, 233–40. ICML ’06. New York, NY, USA: ACM. doi:10.1145/1143844.1143874.
  - Fawcett, Tom. 2006. “An Introduction to ROC Analysis.” *Pattern Recogn. Lett.* 27 (8): 861–74. doi:10.1016/j.patrec.2005.10.010.

- Required
  - Alpaydin 2010, Section 19.7

ROC and PR Curves in Scikit-Learn

- http://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
- http://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html#example-model-selection-plot-roc-py

```
from sklearn.metrics import *
P,R = precision_recall_curve(Y,Z)
AUCPR = average_precision_score(Y,Z)
```

Very Important Exercise and associated Questions
https://github.com/foxtrotmike/CS909/blob/master/evaluation_example.ipynb

Metrics

Classification		
‘accuracy’	metrics.accuracy_score	
‘balanced_accuracy’	metrics.balanced_accuracy_score	
‘top_k_accuracy’	metrics.top_k_accuracy_score	
‘average_precision’	metrics.average_precision_score	
‘neg_brier_score’	metrics.brier_score_loss	
‘f1’	metrics.f1_score	for binary targets
‘f1_micro’	metrics.f1_score	micro-averaged
‘f1_macro’	metrics.f1_score	macro-averaged
‘f1_weighted’	metrics.f1_score	weighted average
‘f1_samples’	metrics.f1_score	by multilabel sample
‘neg_log_loss’	metrics.log_loss	requires predict_proba support
‘precision’ etc.	metrics.precision_score	suffixes apply as with ‘f1’
‘recall’ etc.	metrics.recall_score	suffixes apply as with ‘f1’
‘jaccard’ etc.	metrics.jaccard_score	suffixes apply as with ‘f1’
‘roc_auc’	metrics.roc_auc_score	
‘roc_auc_ovr’	metrics.roc_auc_score	
‘roc_auc_ovo’	metrics.roc_auc_score	
‘roc_auc_ovr_weighted’	metrics.roc_auc_score	
Matthews correlation coefficient	metrics.matthews_corrcoef	
‘roc_auc_ovo_weighted’	metrics.roc_auc_score	

You are expected to know at least those in **bold**. Also the difference between micro and macro averaging.
https://scikit-learn.org/stable/modules/model_evaluation.html

Try understanding why “MCC is better than F-measures”:
https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-019-6413-7

Regression	
‘explained_variance’	metrics.explained_variance_score
‘max_error’	metrics.max_error
‘neg_mean_absolute_error’	metrics.mean_absolute_error
‘neg_mean_squared_error’	metrics.mean_squared_error
‘neg_root_mean_squared_error’	metrics.root_mean_squared_error
‘neg_mean_squared_log_error’	metrics.mean_squared_log_error
‘neg_root_mean_squared_log_error’	metrics.root_mean_squared_log_error
‘neg_median_absolute_error’	metrics.median_absolute_error
‘r2’	metrics.r2_score
‘neg_mean_poisson_deviance’	metrics.mean_poisson_deviance
‘neg_mean_gamma_deviance’	metrics.mean_gamma_deviance
‘neg_mean_absolute_percentage_error’	metrics.mean_absolute_percentage_error
‘d2_absolute_error_score’	metrics.d2_absolute_error_score
‘d2_pinball_score’	metrics.d2_pinball_score
‘d2_tweedie_score’	metrics.d2_tweedie_score
Spearman, Pearson Corr and Kendall tau	scipy.stats.pearsonr scipy.stats.spearmanr scipy.stats.kendalltau

Clustering	
‘adjusted_mutual_info_score’	metrics.adjusted_mutual_info_score
‘adjusted_rand_score’	metrics.adjusted_rand_score
‘completeness_score’	metrics.completeness_score
‘fowlkes_mallows_score’	metrics.fowlkes_mallows_score
‘homogeneity_score’	metrics.homogeneity_score
‘mutual_info_score’	metrics.mutual_info_score
‘normalized_mutual_info_score’	metrics.normalized_mutual_info_score
‘rand_score’	metrics.rand_score
‘v_measure_score’	metrics.v_measure_score

Measurement of Generalization Performance

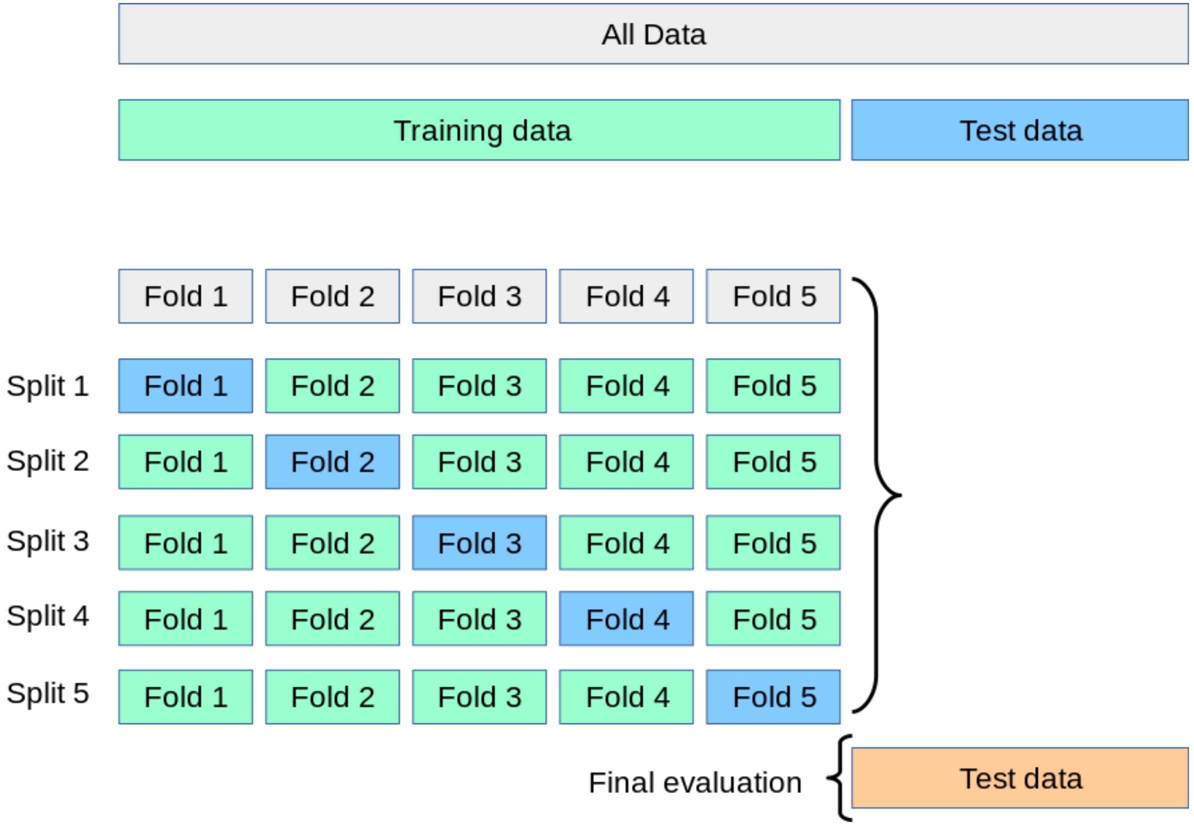
- Typically we do not have access to real world test examples
- Use the given “training” set for approximating the generalization performance
- Guidelines
  - There should be “enough” training examples left
  - Test labels should not be used, directly or indirectly, during training
    - Test data (without labels) can be used
  - You should be clear about the intended use and application of the system
  - You should be clear about the objective of performance evaluation

Issues

- The variance of our estimate increases as the size of the test set decreases.
- A small increase in the pessimistic bias when we decrease the size of the training set

Cross-Validation: K-fold

- Measurement of Generalization Performance
- For estimation of variation
- Divide the data into K folds
  - For k = 1…K
    - Train on K-1 sets leaving the k^th set out for validation
    - Validate on the k^th set and obtain the performance metrics
  - Report the average and the variation in the performance



All Data

Training data | Test data

Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5

Split 1 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5

Split 2 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5

Split 3 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5

Split 4 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5

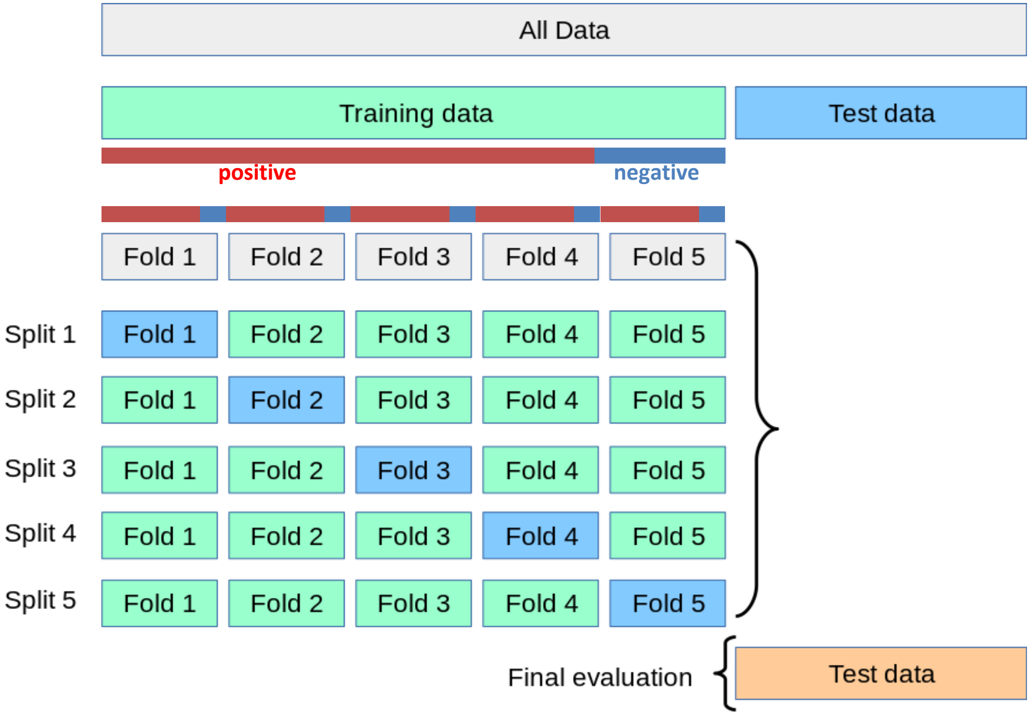
Split 5 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5

Final evaluation | Test data

https://github.com/foxtrotmike/CS909/blob/master/evaluation_example.ipynb

Cross-Validation

- If K = Number of examples then this extreme case is called Leave One Out CV (LOOCV)
  - Useful if the amount of data is small
- **Stratification** (Stratified cross-validation)
  - Make sure that each fold contains the same number of examples as the overall data
    - If a class has 20 percent examples in the whole dataset, in all samples drawn from the dataset, it should also have approximately 20 percent examples.
- What will be the impact on approximated performance with increase in K?



https://github.com/foxtrotmike/CS909/blob/master/evaluation_example.ipynb

So what to use?

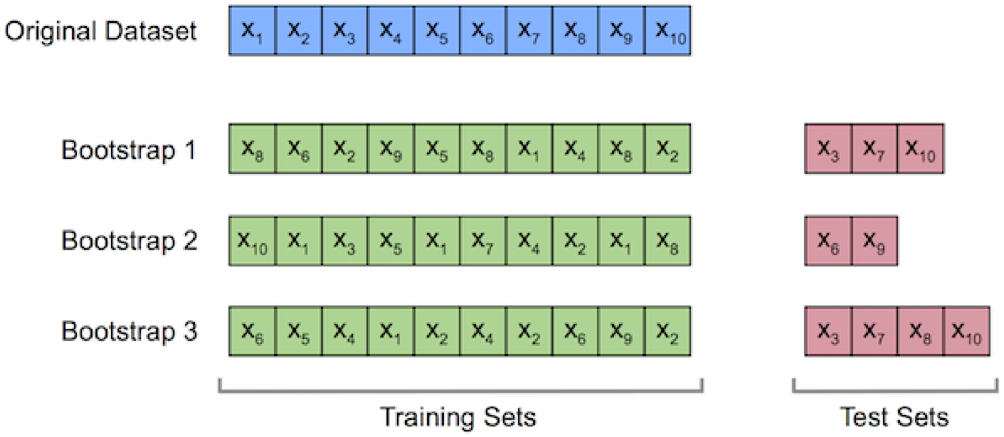
- 10 Fold Stratified Cross-Validation is good
  - May give overly optimistic values. However, its okay to use it for comparison of classifiers.
- However, for small sample sizes, it can have a large variance in which case you can use LOOCV or the .632 or the .632+ bootstrap
- For comparison of multiple classifiers: Cochran Test, F-Test
- SCIKIT-LEARN
- MLXTEND

http://rasbt.github.io/mlxtend/
http://scikit-learn.org/stable/model_selection.html

Kohavi, Ron. 1995. "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection." In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, 1137–43. IJCAI'95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. http://dl.acm.org/citation.cfm?id=1643031.1643047.

Bootstrapping

- Bootstrapping is any test or metric that uses random sampling with replacement (e.g. mimicking the sampling process), and falls under the broader class of resampling methods.
- Basic idea (Out of Bag Bootstrap)
  - – Take the entire data set of N examples
  - – For multiple iterations
    - Sample N examples from it with replacement
      - – You will get a total of N samples but some may be the same example
      - – You train on this set
      - – Test on the set of remaining examples and compute metrics
  - – Report the average metric and its standard deviation
- Should be stratified
- Code example
- Other variants such as .632 and .632+ bootstrap also available



Original Dataset x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10}

Bootstrap 1 x_8 x_6 x_2 x_9 x_5 x_8 x_1 x_4 x_8 x_2 | x_3 x_7 x_{10}

Bootstrap 2 x_{10} x_1 x_3 x_5 x_1 x_7 x_4 x_2 x_1 x_8 | x_6 x_9

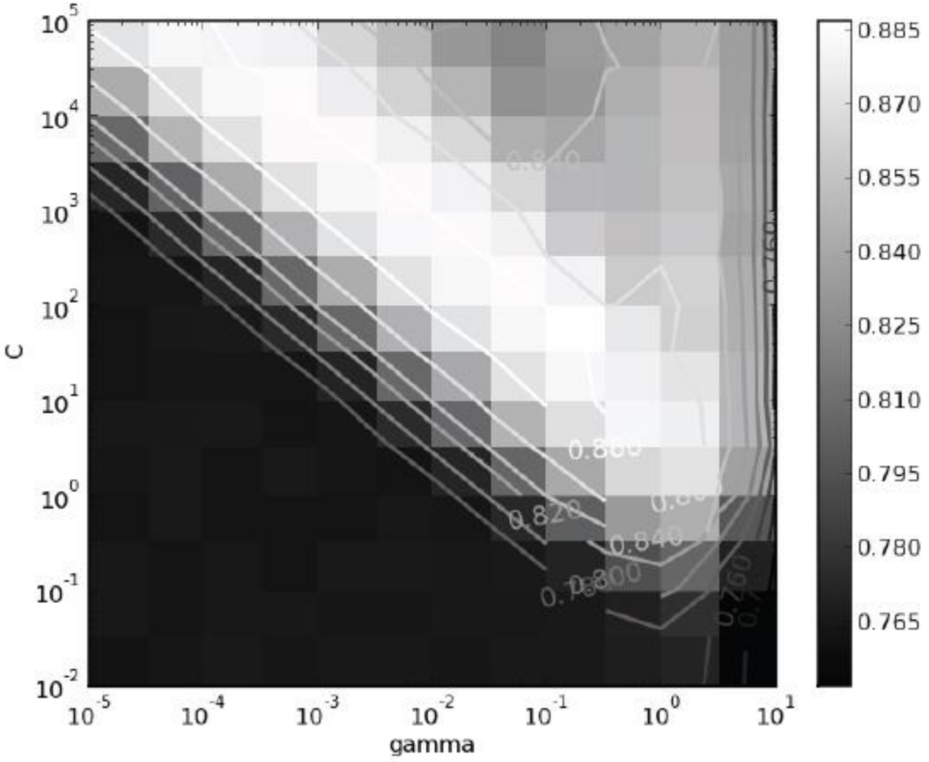
Bootstrap 3 x_6 x_5 x_4 x_1 x_2 x_4 x_2 x_6 x_9 x_2 | x_3 x_7 x_8 x_{10}

Training Sets

Test Sets

Model Hyperparameter Selection

- Grid Search
  - Exhaustive Search through Cross-Validation
    - Recommended: Nested Cross-Validation or separate test set
- There can be a range of parameter values that yield optimal values and these equivalent points in the parameter space fall along a ridge



Ben-Hur, Asa, and Jason Weston. 2010. "A User's Guide to Support Vector Machines." In *Data Mining Techniques for the Life Sciences*, edited by Oliviero Carugo and Frank Eisenhaber, 223–39. Methods in Molecular Biology 609. Humana Press. http://dx.doi.org/10.1007/978-1-60327-241-4_13.

Searching for optimal parameters

- Regularization Path Finding
- Gradient Based Approaches
- Evolutionary approaches

- Grid Search in Scikit-learn
  - http://scikit-learn.org/stable/modules/grid_search.html

Parameter Selection

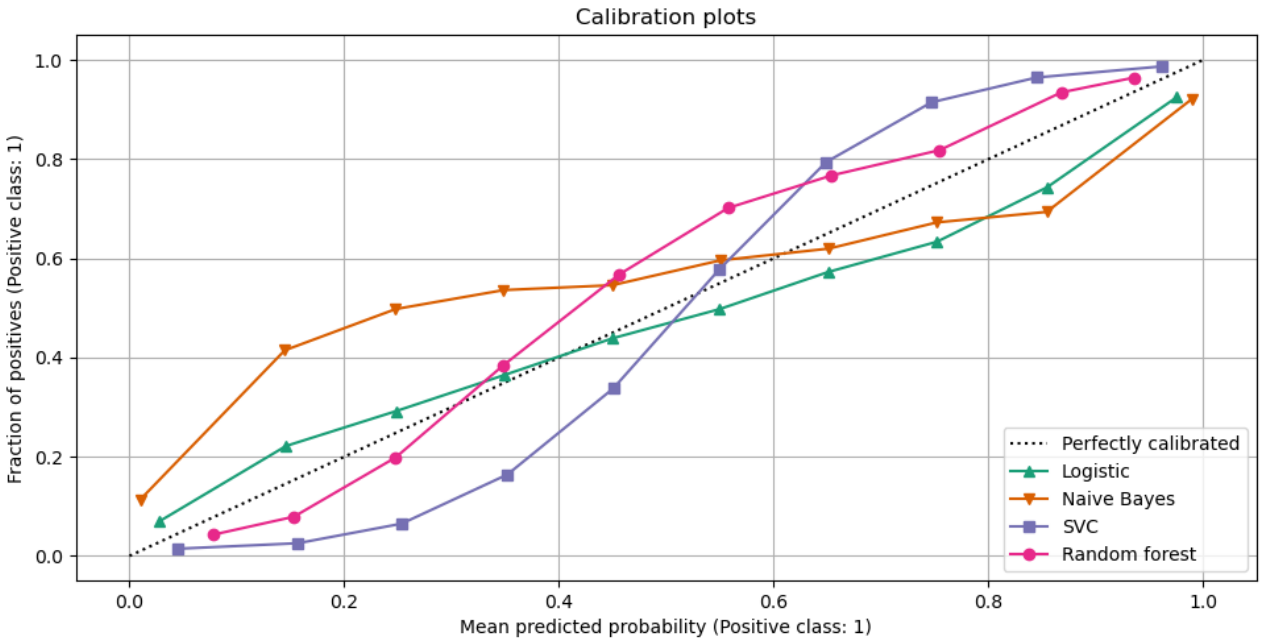
- http://hyperopt.github.io/
- http://hyperopt.github.io/hyperopt-sklearn/
- https://automl.github.io/auto-sklearn/stable/api.html
- https://en.wikipedia.org/wiki/Xgboost
- http://www.kdnuggets.com/2017/01/current-state-automated-machine-learning.html

“Other” ways of selecting parameters

- Selecting gamma
  - Visualize the spread
- Ensuring robustness to parameter changes

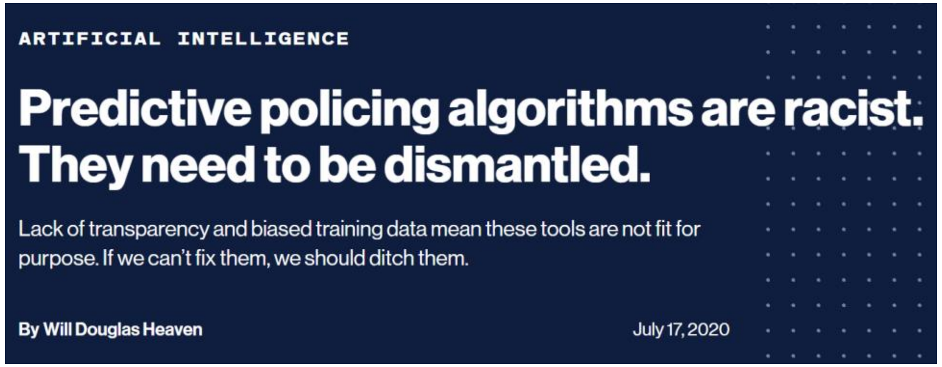
(Very) Important Notes

- Performance evaluation results in “estimates”
- Metrics tell you what you ask of them. It is upto you to choose the appropriate metric
- Having a high-performance metric is no guarantee that the classifier is good for use
  - A model may have hidden biases, be unfair or fail in other ways
  - Can be affected by confounding factors (remember tank classification?)
  - May not be robust or transparent
  - A model may not be well-calibrated, i.e., its output can be interpreted probabilistically or used for uncertainty quantification
    - Example: Without careful calibration, it cannot be assumed that if the prediction score of an example is 0.9, the probability of it being positive is 0.9
    - https://scikit-learn.org/stable/modules/calibration.html
    - Platt Scaling
    - Uncertainty quantification with conformal prediction



THE NEURAL NET TANK URBAN LEGEND

AI folklore tells a story about a neural network trained to detect tanks which instead learned to detect time of day; investigating, this probably never happened.



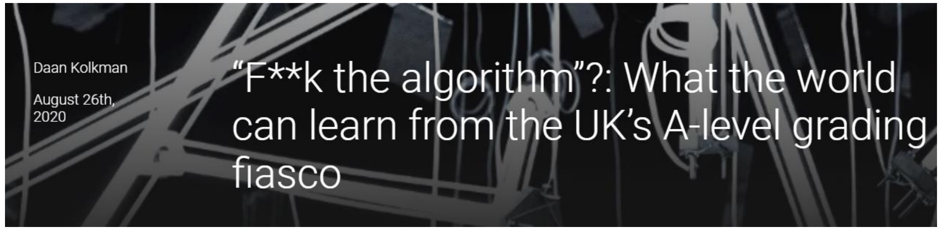
ARTIFICIAL INTELLIGENCE

Predictive policing algorithms are racist. They need to be dismantled.

Lack of transparency and biased training data mean these tools are not fit for purpose. If we can't fix them, we should ditch them.

By Will Douglas Heaven

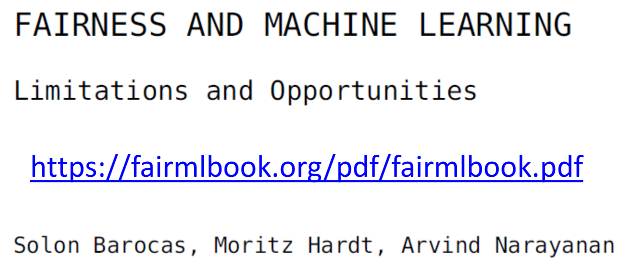
July 17, 2020



Daan Kolkman

August 26th, 2020

“F**k the algorithm”?: What the world can learn from the UK’s A-level grading fiasco



FAIRNESS AND MACHINE LEARNING

Limitations and Opportunities

https://fairmlbook.org/pdf/fairmlbook.pdf

Solon Barocas, Moritz Hardt, Arvind Narayanan

Some papers

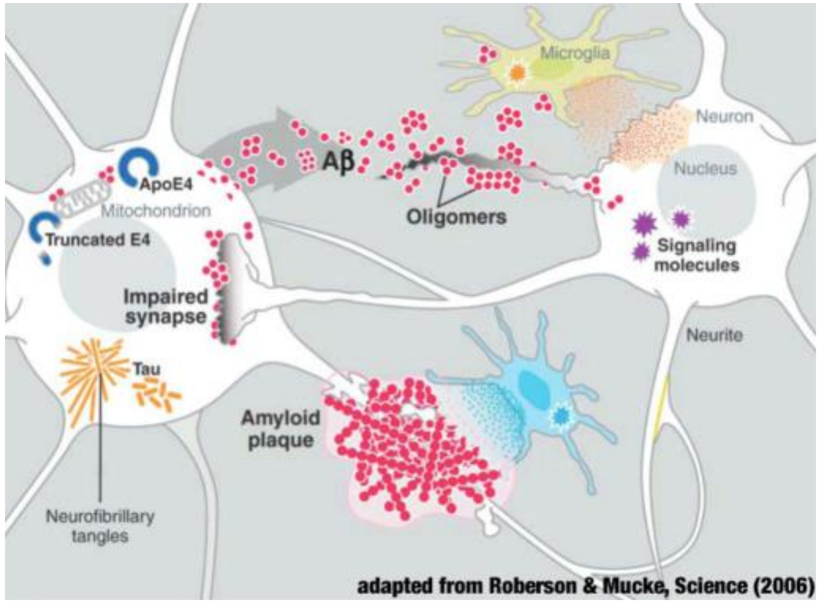
- Chapter 19 “Design and Analysis of Machine Learning Experiments” Alpaydin, Ethem. 2010. *Introduction to Machine Learning*. Cambridge, Mass.: MIT Press.

- Demšar, Janez. 2006. “Statistical Comparisons of Classifiers over Multiple Data Sets.” *J. Mach. Learn. Res.* 7 (December): 1–30.
- Salvador Garcí, and Francisco Herrera. 2008. “An Extension on ‘Statistical Comparisons of Classifiers over Multiple Data Sets’ for All Pairwise Comparisons.” *Journal of Machine Learning Research* 9 (Dec.): 2677–94.
- Chapelle, Olivier, Vladimir Vapnik, Olivier Bousquet, and Sayan Mukherjee. 2002. “Choosing Multiple Parameters for Support Vector Machines.” *Machine Learning* 46 (1-3): 131–59. doi:10.1023/A:1012450327387.
- Hastie, Trevor, Saharon Rosset, Robert Tibshirani, and Ji Zhu. 2004. “The Entire Regularization Path for the Support Vector Machine.” *J. Mach. Learn. Res.* 5 (December): 1391–1415.

- Fernández-Delgado, Manuel, Eva Cernadas, Senén Barro, and Dinani Amorim. 2014. “Do We Need Hundreds of Classifiers to Solve Real World Classification Problems?” *Journal of Machine Learning Research* 15: 3133–81
- Forman, George, and Ira Cohen. 2004. “Learning from Little: Comparison of Classifiers Given Little Training.” In *Knowledge Discovery in Databases: PKDD 2004*, edited by Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, 161–72. Lecture Notes in Computer Science 3202. Springer Berlin Heidelberg. http://link.springer.com/chapter/10.1007/978-3-540-30116-5_17.
- Salperwyck, C., and V. Lemaire. 2011. “Learning with Few Examples: An Empirical Study on Leading Classifiers.” In *The 2011 International Joint Conference on Neural Networks (IJCNN)*, 1010–19. doi:10.1109/IJCNN.2011.6033333.

CASE STUDY

- Amyloid Prediction by Farzeen
  - Alzheimer’s, Parkinson’s, Huntington’s, ALS, Type-II Diabetes, Cataracts
- Given:
  - Labeled Peptide Sequences of length 6
- Output:
  - Predict if a protein contains a sequence that can form amyloids
- Preprocessing
  - Normalization
- Training Data
- Test Data



adapted from Roberson & Mucke, Science (2006)

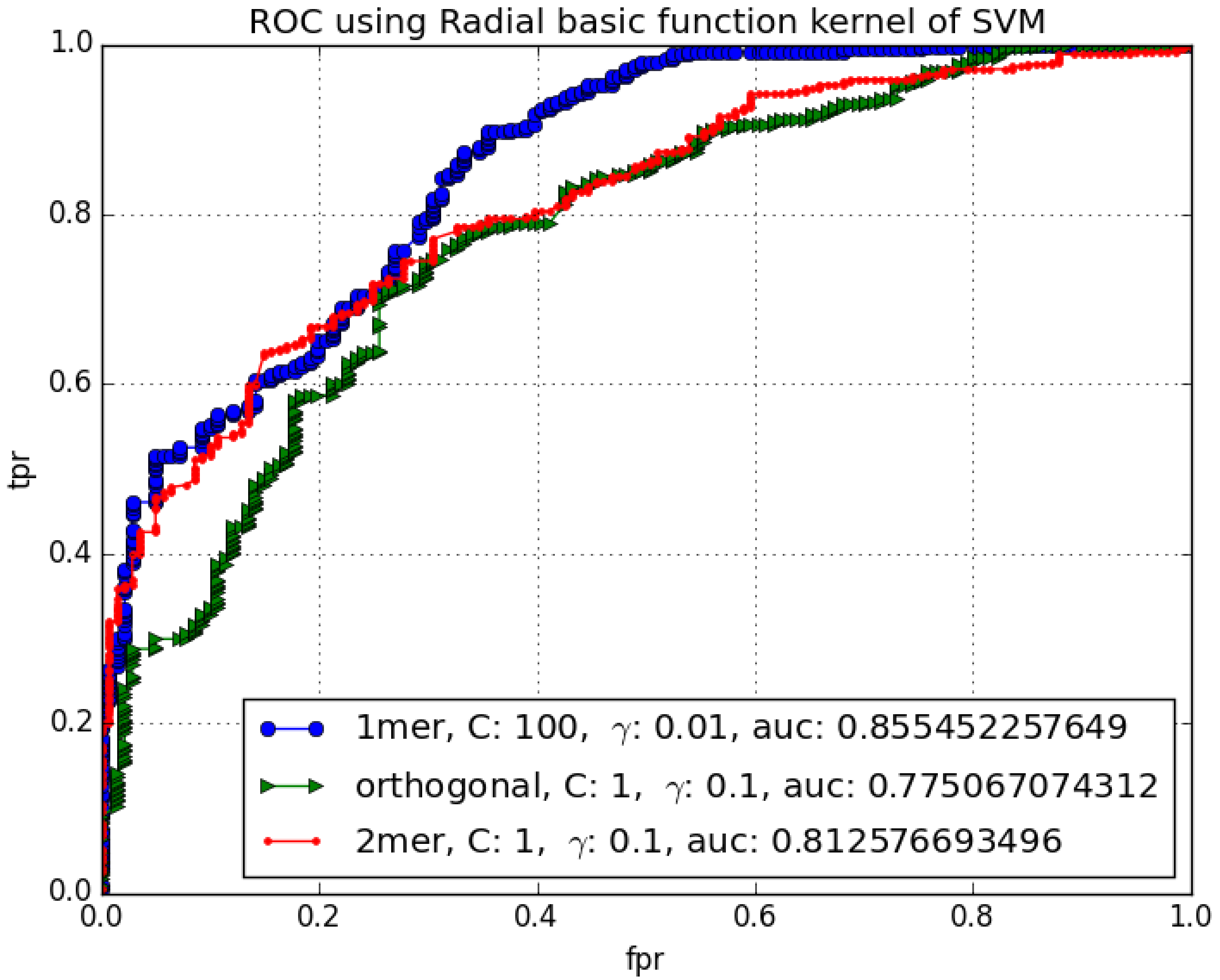
http://www.cmu.edu/biolphys/smsl/research/topics/amyloids.html

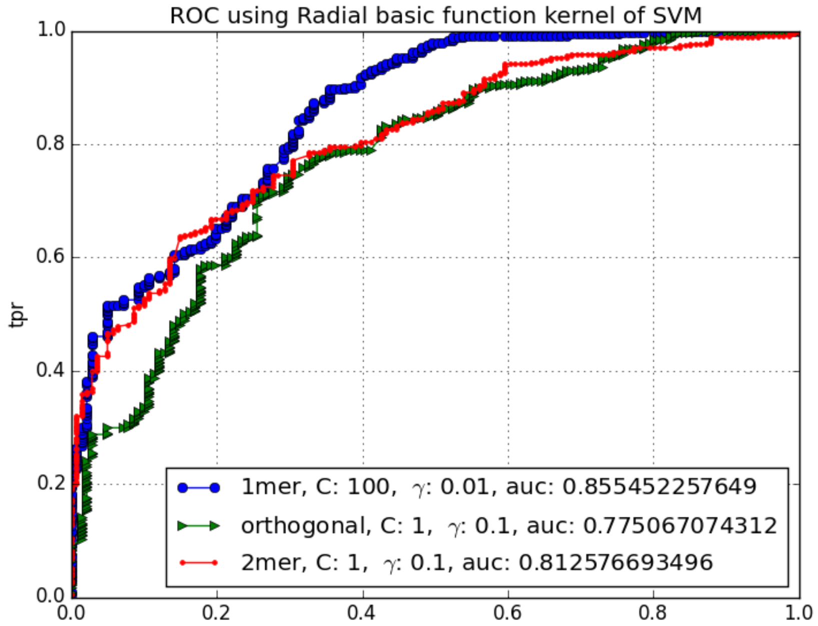
Selecting “C”

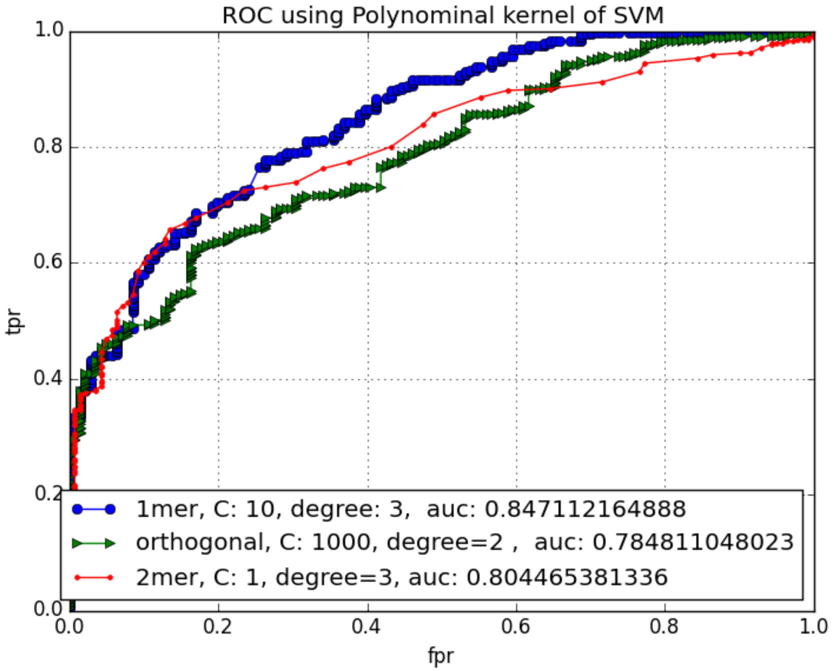
C, 1 mer	1st fold of training	2nd fold of training	3rd fold of training	average
0.0001	0.78	0.65	0.822	0.75
0.001	0.78	0.67	0.824	0.75
0.01	0.78	0.67	0.823	0.757
0.1	0.79	0.705	0.87	0.788
1	0.801	0.75	0.84	**0.797**
10	0.809	0.76	0.809	0.7926
100	0.809	0.76	0.809	0.792

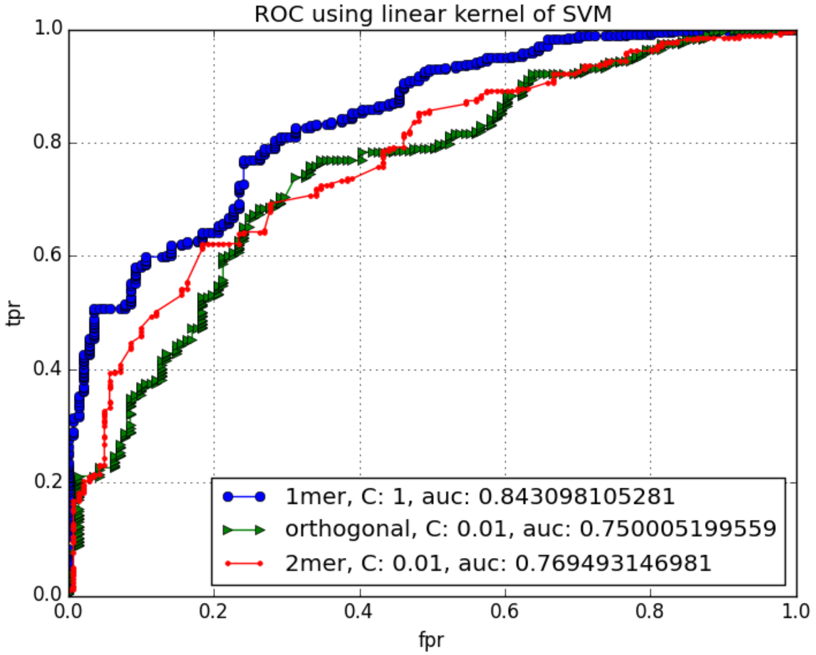
Grid Search

C	gamma=0.001	gamma=0.01	gamma=0.1	gamma=1	gamma=10	gamma=100	gamma=1000
0.0001	0.734	0.733	0.734	0.731	0.525	0.525	0.525
0.001	0.734	0.734	0.734	0.734	0.55	0.546	0.546
0.01	0.734	0.732	0.735	0.737	0.553	0.546	0.546
0.1	0.736	0.7329	0.735	0.704	0.553	0.546	0.546
1	0.724	0.733	**0.758**	0.745	0.597	0.546	0.546
10	0.7340	0.750	0.725	0.751	0.597	0.546	0.546
100	0.746	0.687	0.725	0.751	0.597	0.546	0.546
1000	0.746	0.687	0.725	0.751	0.597	0.546	0.546


ROC using Radial basic function kernel of SVM
tpr
fpr
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1mer, C: 100, γ: 0.01, auc: 0.855452257649
orthogonal, C: 1, γ: 0.1, auc: 0.775067074312
2mer, C: 1, γ: 0.1, auc: 0.812576693496

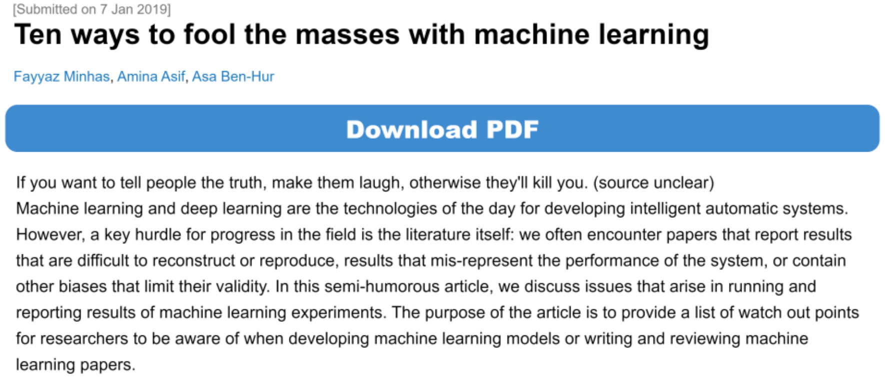

ROC using Radial basic function kernel of SVM
tpr
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1mer, C: 100, γ: 0.01, auc: 0.855452257649
orthogonal, C: 1, γ: 0.1, auc: 0.775067074312
2mer, C: 1, γ: 0.1, auc: 0.812576693496


ROC using Polynominal kernel of SVM
tpr
fpr
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1mer, C: 10, degree: 3, auc: 0.847112164888
orthogonal, C: 1000, degree=2 , auc: 0.784811048023
2mer, C: 1, degree=3, auc: 0.804465381336


ROC using linear kernel of SVM
tpr
fpr
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1mer, C: 1, auc: 0.843098105281
orthogonal, C: 0.01, auc: 0.750005199559
2mer, C: 0.01, auc: 0.769493146981

Ten simple rules to fool the masses with machine learning

- Choose a biased accuracy metric or a metric irrelevant to the problem domain OR Fail to relate the accuracy to impact on the problem domain
- Choose hyper-parameters that maximize the performance metric
- Do not analyze the sensitivty of your model to changes in data, hyper-parameter values or randomness
- Use labeled validation data in training
- Forget that examples may not be independent of each other: use statistical tests even if they might not be applicable
- Do not compare with a simple baseline classifier
- Compare your model with unoptimized versions of other models or ones that have been trained using different data or a different evaluation protocol
- Forget about reproducibility
  - Do not provide detailed performance results, codes or a webserver, Is a model that fits better, better? A model should know when it doesn't know!
- Only publishing matters: Forget about deployment or generalization



[Submitted on 7 Jan 2019]

Ten ways to fool the masses with machine learning

Fayyaz Minhas, Amina Asif, Asa Ben-Hur

If you want to tell people the truth, make them laugh, otherwise they'll kill you. (source unclear) Machine learning and deep learning are the technologies of the day for developing intelligent automatic systems. However, a key hurdle for progress in the field is the literature itself: we often encounter papers that report results that are difficult to reconstruct or reproduce, results that mis-represent the performance of the system, or contain other biases that limit their validity. In this semi-humorous article, we discuss issues that arise in running and reporting results of machine learning experiments. The purpose of the article is to provide a list of watch out points for researchers to be aware of when developing machine learning models or writing and reviewing machine learning papers.

A useful baseline: Naïve Bayes

- Naïve Bayes methods are a set of supervised learning algorithms, works quite well!
  - Apply Bayes’ theorem with the naïve assumption
    - Features are independent of each other
  - The Bayes theorem states that: $P(y \mid x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots x_n \mid y)}{P(x_1, \dots, x_n)}$
  - Independence assumption implies that

$$P(x_i|y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i|y),$$

  - Thus,

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

$$P(y \mid x_1, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

What are the:

Representation
Evaluation
Optimization

For a Naïve Bayes Classifier?

H. Zhang (2004). The optimality of Naive Bayes. Proc. FLAIRS.

End of Lecture

if you can't measure it, you can't manage it.

You get what you measure!