Lecture 6 - Learning Best Practices for Model Evaluation and Hyperparameter Tuning

Andre E. Lazzaretti

Introduction

- In this lecture, we will learn how to do the following:
  - Obtain unbiased estimates of a model's performance;
  - Diagnose the common problems of machine learning algorithms;
  - Fine-tune machine learning models;
  - Evaluate predictive models using different performance metrics.

Breast Cancer Wisconsin Dataset

- It contains 569 samples of malignant and benign tumor cells.
- The first two columns in the dataset store the unique ID numbers of the samples and the corresponding diagnoses (M = malignant, B = benign), respectively.
- Columns 3-32 contain 30 real-valued features that have been computed from digitized images of the cell nuclei, which can be used to build a model to predict whether a tumor is benign or malignant.
- The Breast Cancer Wisconsin dataset has been deposited in the UCI Machine Learning Repository.

Breast Cancer Wisconsin Dataset

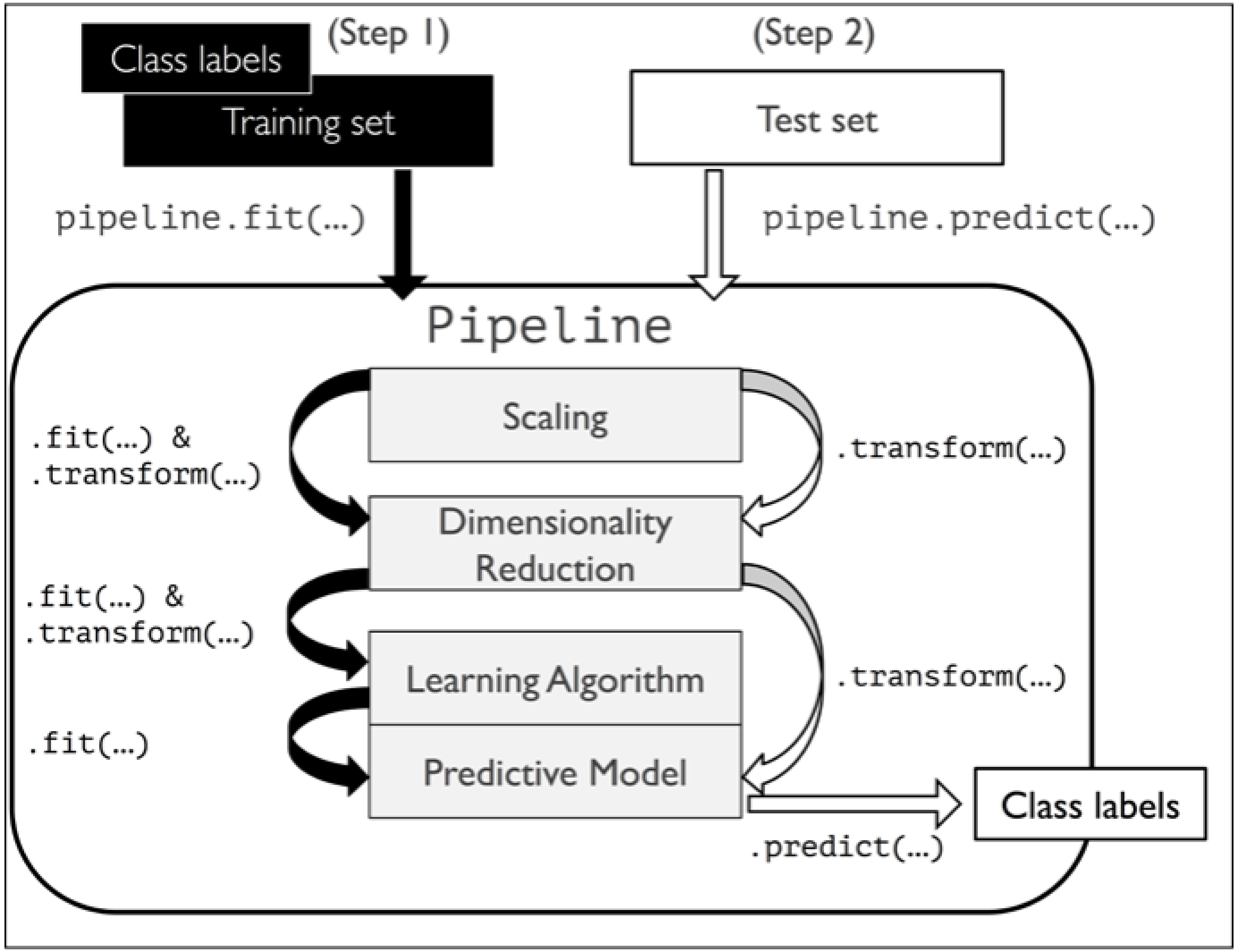
- PYTHON Codes for reading and splitting…

Combining transformers and estimators in a pipeline

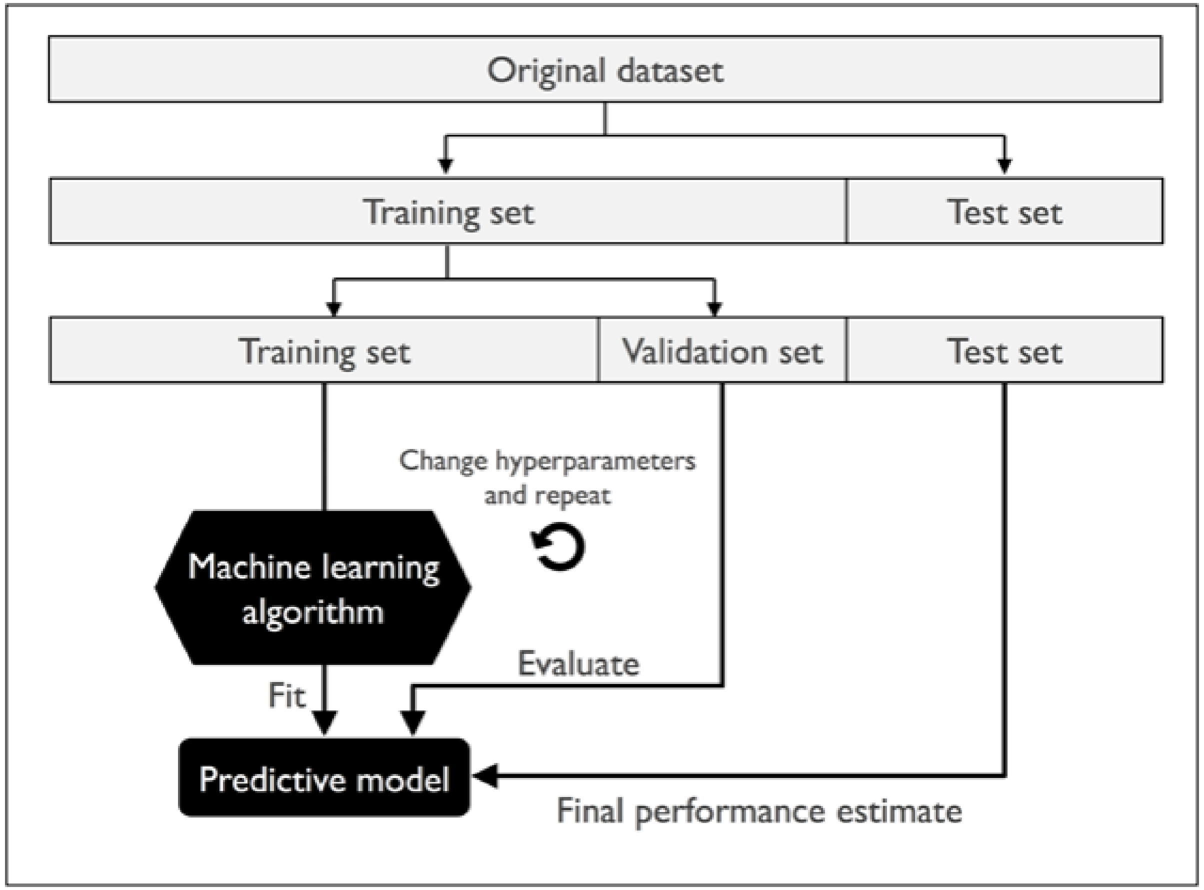
- The make_pipeline function takes an arbitrary number of scikit-learn transformers (objects that support the fit and transform methods as input), followed by a scikit-learn estimator that implements the fit and predict methods.

```
>>> from sklearn.preprocessing import StandardScaler
>>> from sklearn.decomposition import PCA
>>> from sklearn.linear_model import LogisticRegression
>>> from sklearn.pipeline import make_pipeline
>>> pipe_lr = make_pipeline(StandardScaler(),
...                         PCA(n_components=2),
...                         LogisticRegression(random_state=1))
>>> pipe_lr.fit(X_train, y_train)
>>> y_pred = pipe_lr.predict(X_test)
>> print('Test Accuracy: %.3f' % pipe_lr.score(X_test, y_test))
Test Accuracy: 0.956
```

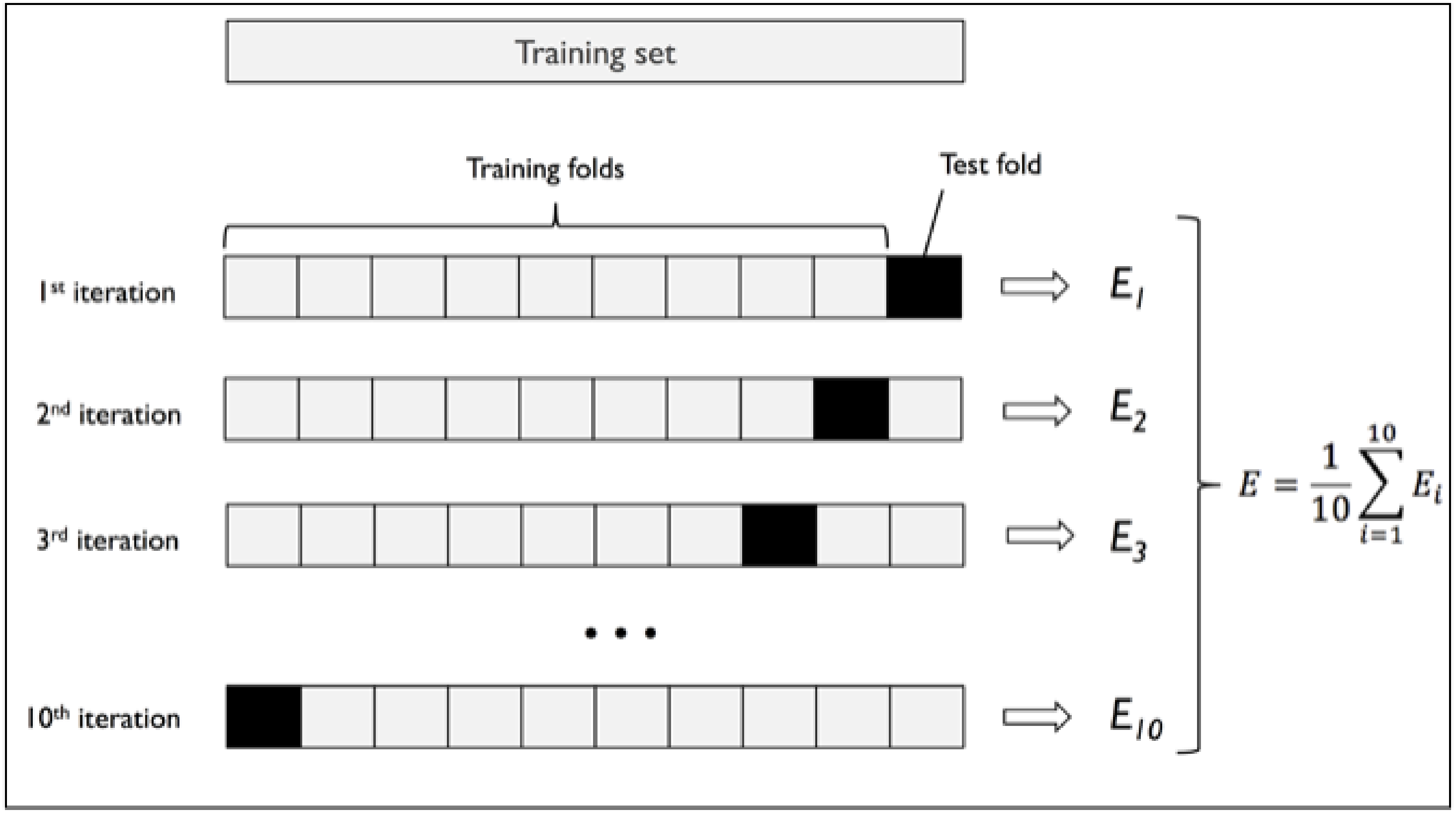
Fit and Predict Pipeline



Holdout cross-validation



K-Fold Cross-Validation



Cross-validation

- PYTHON Codes…

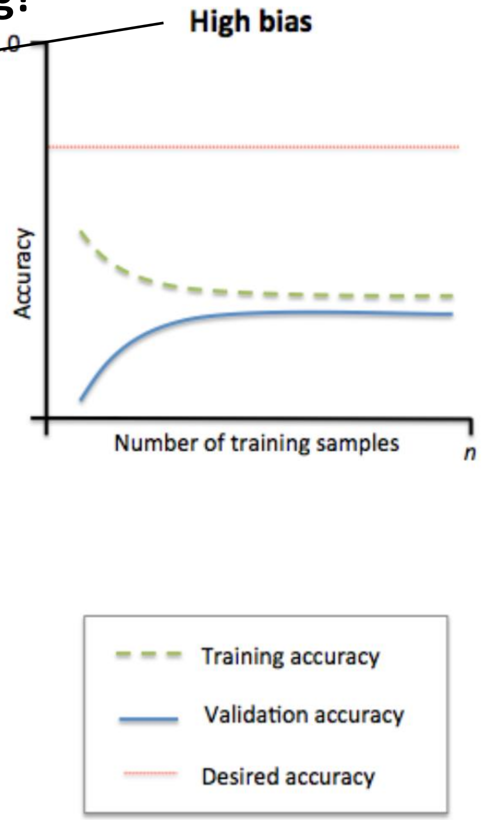
Curve Analysis

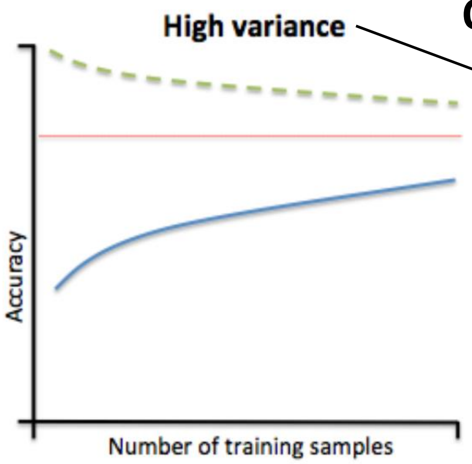
Underfitting!

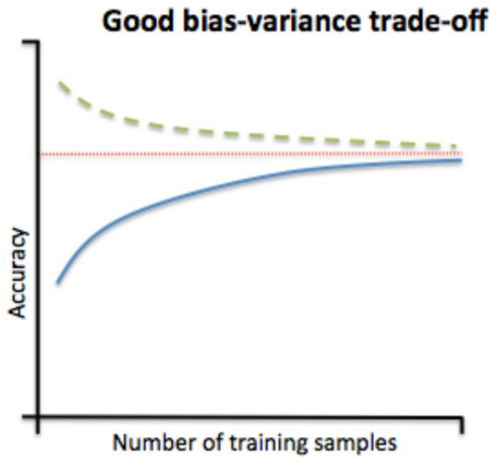
More training data will not significantly help.

Try adding new features.

Check the cross-validation, i.e., increase the variable range.







Overfitting!

More training data will probably help.

Try removing some features.

Check the cross-validation, i.e., decrease the variable range.

Curve Analysis

- PYTHON Codes…

Grid Search

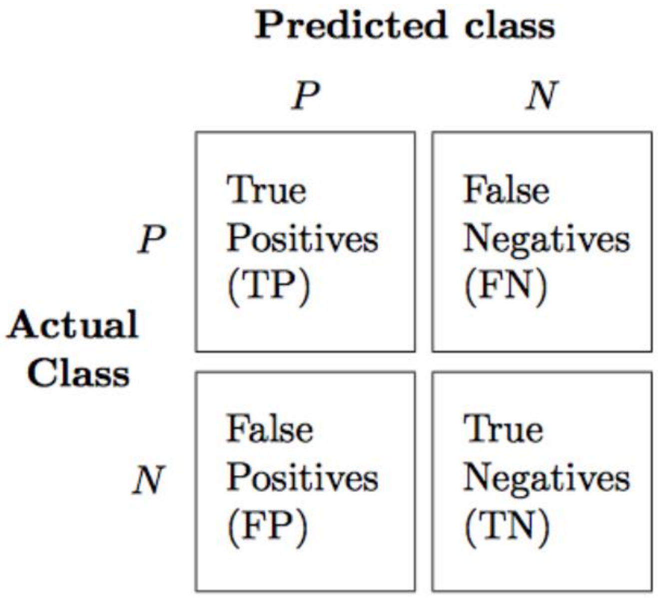
- We have two types of parameters: those that are learned from the training data, for example, the weights in logistic regression, and the parameters of a learning algorithm that are optimized separately.
- The latter are the tuning parameters, also called hyperparameters, of a model, for example, the regularization parameter in logistic regression or the depth parameter of a decision tree.
- The approach of grid search is quite simple; it's a brute-force exhaustive search paradigm where we specify a list of values for different hyperparameters, and the computer evaluates the model performance for each combination of those to obtain the optimal combination of values from this set.

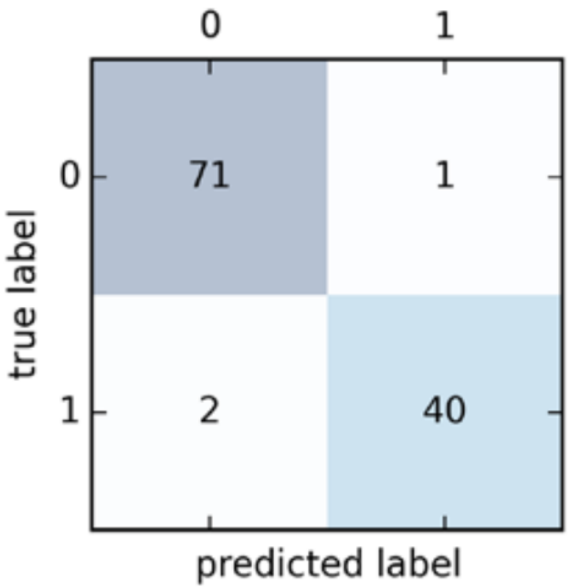
Grid Search

- Python Codes...

Confusion Matrix

- Assuming you are working with a two-class classification problem and have at hand $\mathbf{y}_{pred}$ e $\mathbf{y}_{true}$:





Error, Accuracy, TP, FP, F1

$$ACC = \frac{TP + TN}{FP + FN + TP + TN} = 1 - ERR$$

$$ERR = \frac{FP + FN}{FP + FN + TP + TN}$$

$$TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN}$$

$$PRE = \frac{TP}{TP + FP}$$

$$REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

$$F1 = 2\frac{PRE \times REC}{PRE + REC}$$

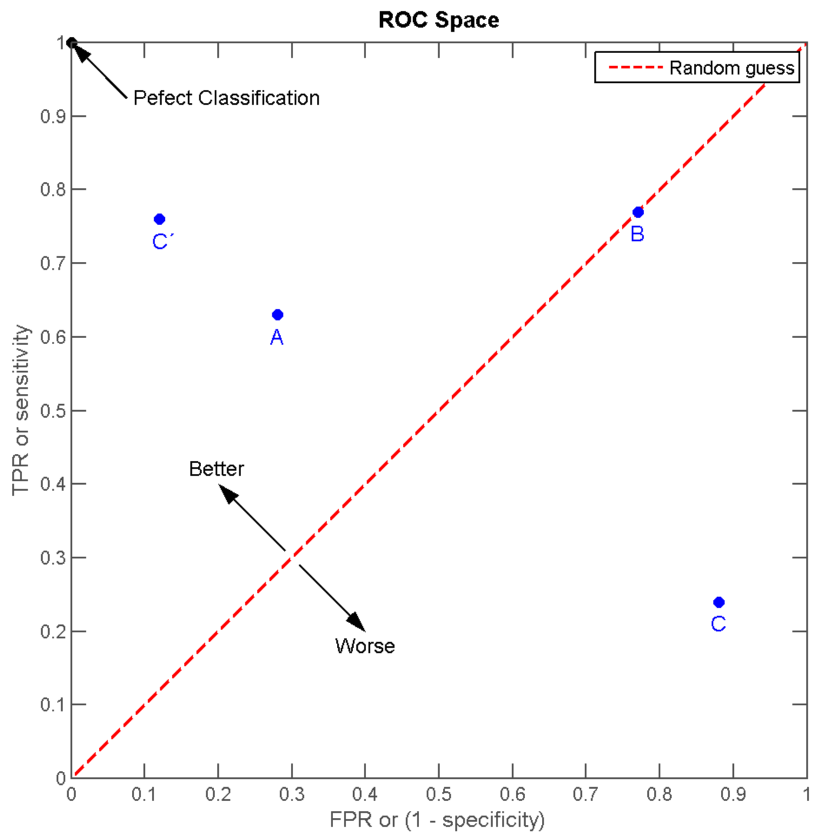
		Predicted class	
		P	N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

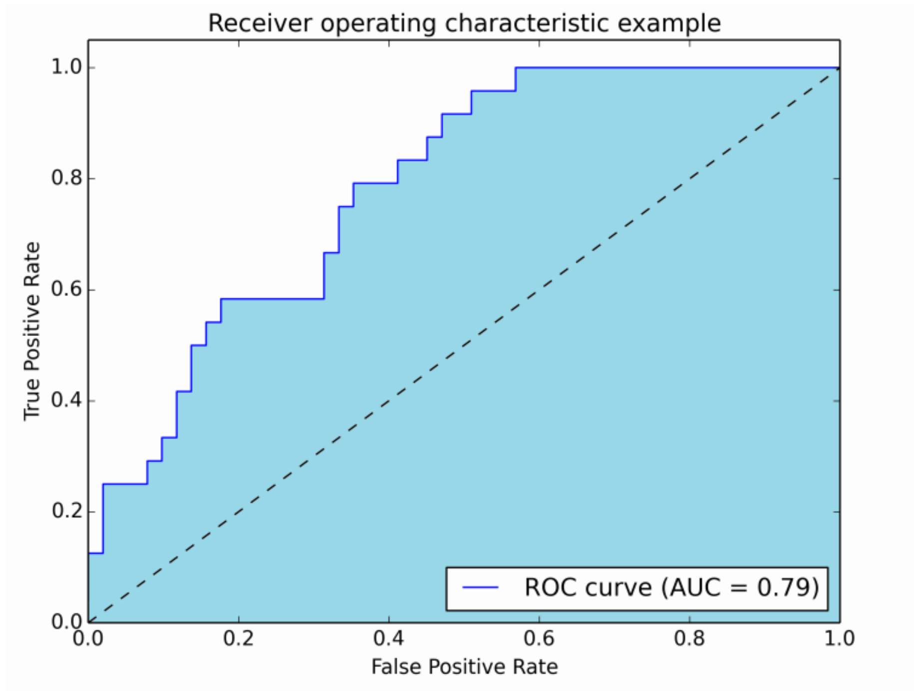
Multiclass case:

$$PRE_{micro} = \frac{TP_1 + \cdots + TP_k}{TP_1 + \cdots + TP_k + FP_1 + \cdots + FP_k}$$

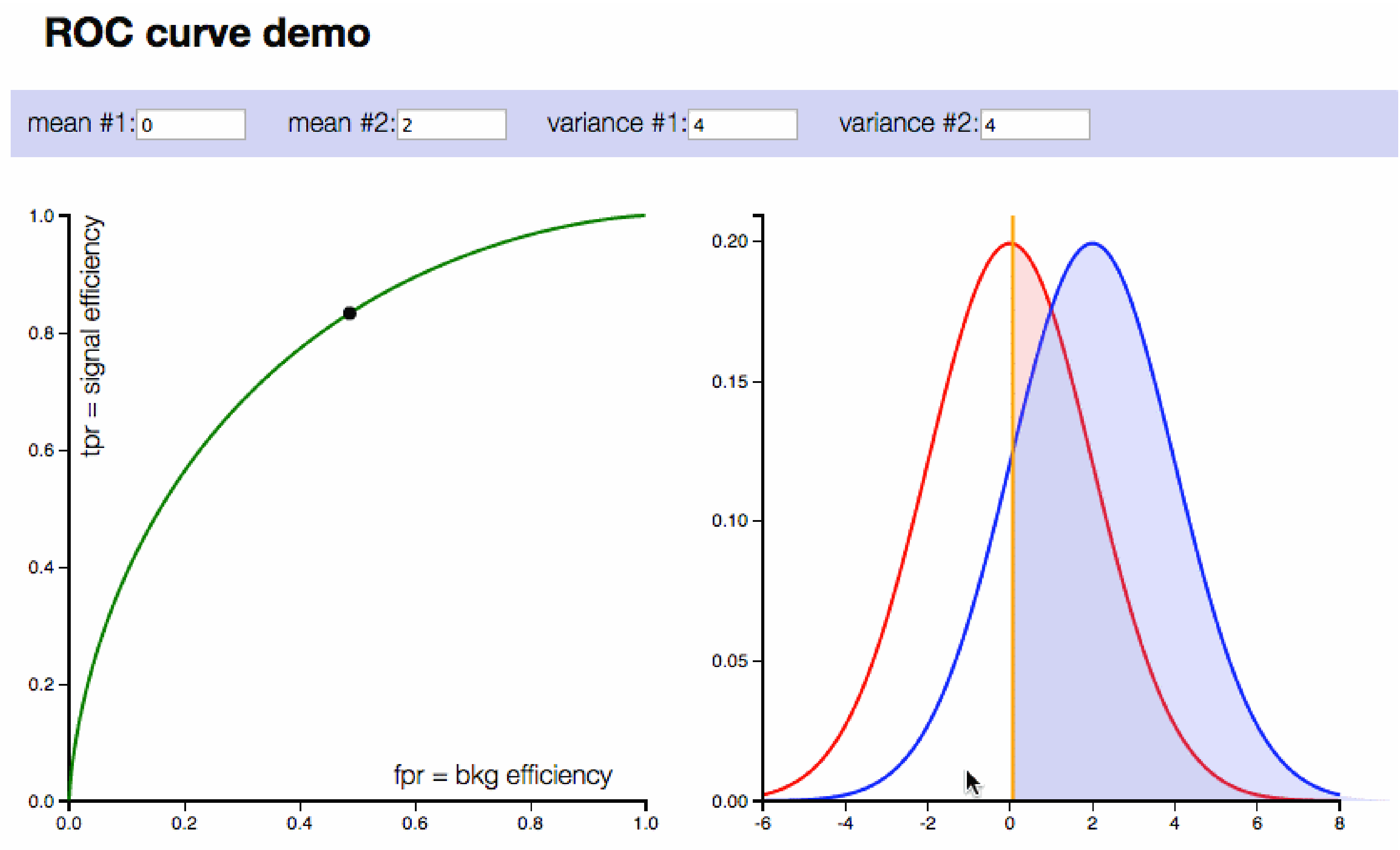
$$PRE_{macro} = \frac{PRE_1 + \cdots + PRE_k}{k}$$

Receiver Operating Curve (ROC)





Receiver Operating Curve (ROC)



Other remarks

- If you are an expert on the problem, make a visual inspection of the data and see if you can differentiate what the classifier is doing wrong;
- Check the annotation (labels) of the data;
- Check the main errors of the confusion matrix;

Metrics

- PYTHON Codes...