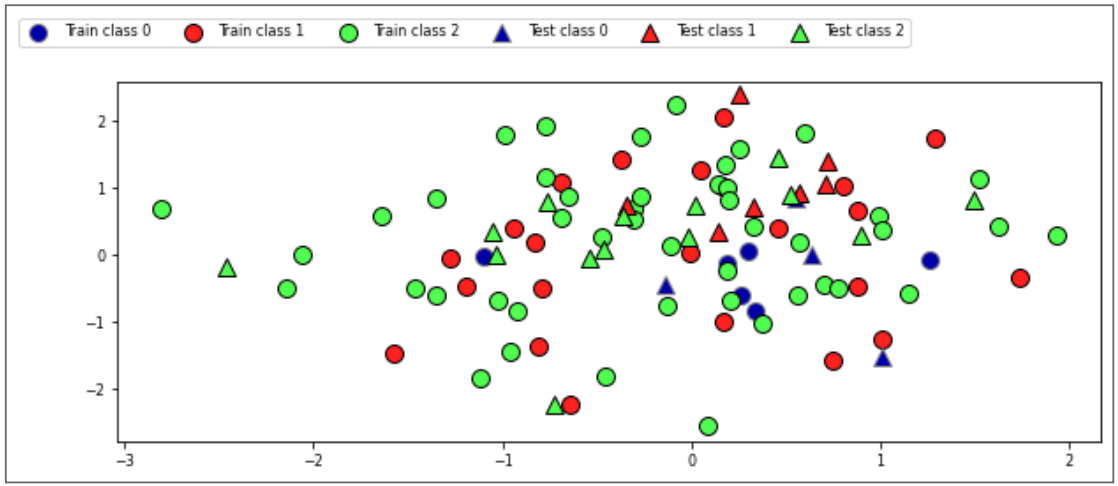
Model Selection

Performance estimation techniques

- Always evaluate models *as if they are predicting future data*
- We do not have access to future data, so we pretend that some data is hidden
- Simplest way: the *holdout* (simple train-test split)

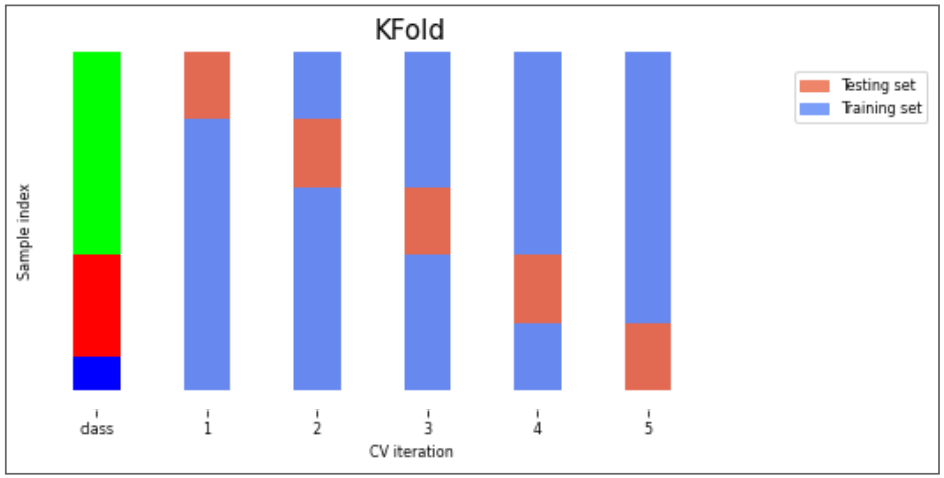
Randomly split data (and labels) into training , Validation and test set (e.g. 60%-20%-20 %)

Train (fit) a model on the training data, minimize error on validation set and score on the test data



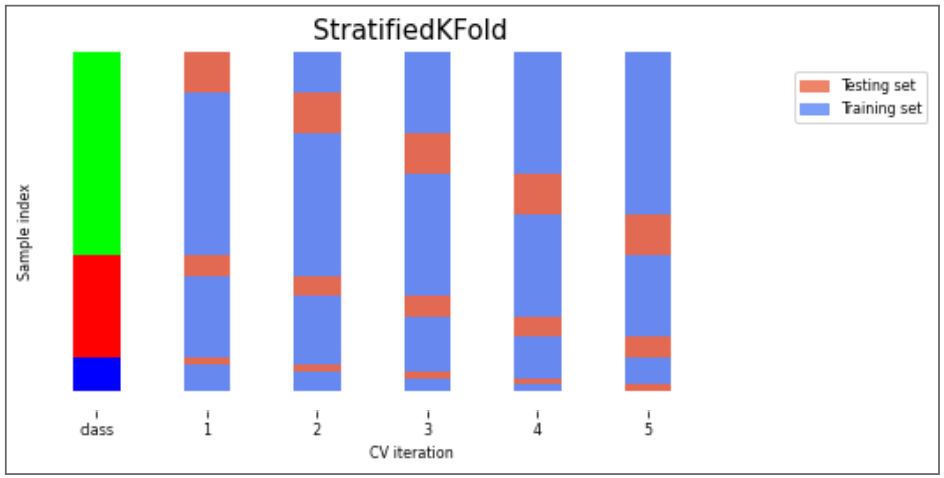
K-fold Cross-validation

- Each random split can yield very different models (and scores)
  - e.g. all easy (of hard) examples could end up in the test set
- Split data into *k* equal-sized parts, called *folds*
  - Create *k* splits, each time using a different fold as the test set
- Compute *k* evaluation scores, aggregate afterwards (e.g. take the mean)
- Large *k* gives better estimates (more training data), but is expensive



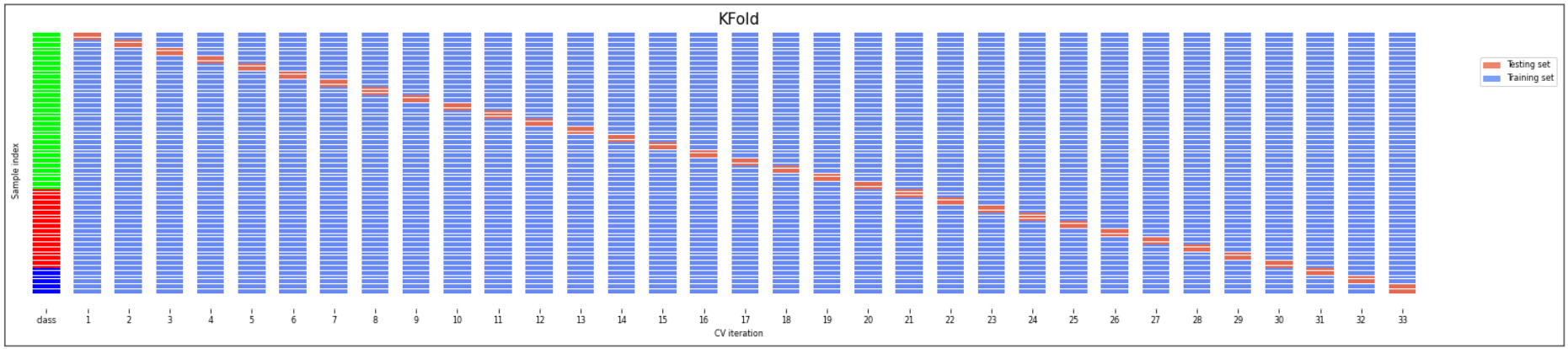
Stratified K-Fold cross-validation

- If the data is unbalanced, some classes have only few samples
- Likely that some classes are not present in the test set
- Stratification: *proportions* between classes are conserved in each fold
    - Order examples per class
    - Separate the samples of each class in *k* sets (strata)
    - Combine corresponding strata into folds



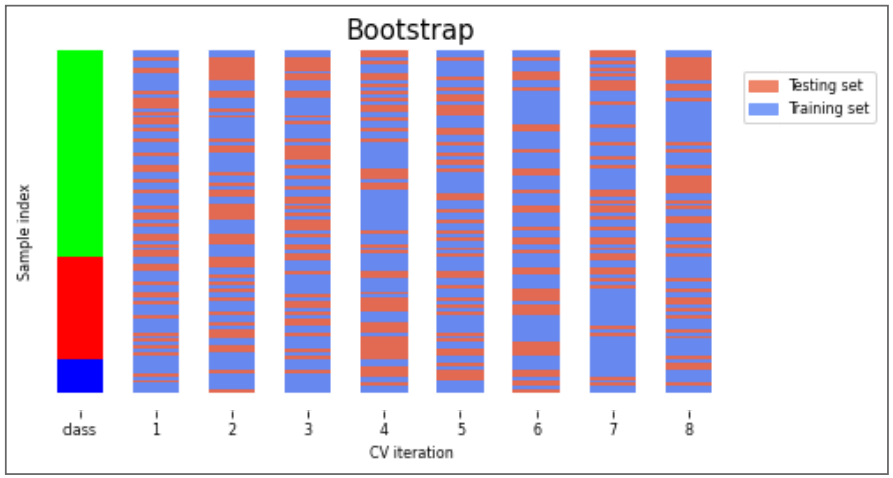
Leave-One-Out cross-validation

- *k* fold cross-validation with *k* equal to the number of samples
- Completely unbiased (in terms of data splits), but computationally expensive
- Actually generalizes *less* well towards unseen data
    - The training sets are correlated (overlap heavily)
    - Overfits on the data used for (the entire) evaluation
    - A different sample of the data can yield different results
- Recommended only for small datasets



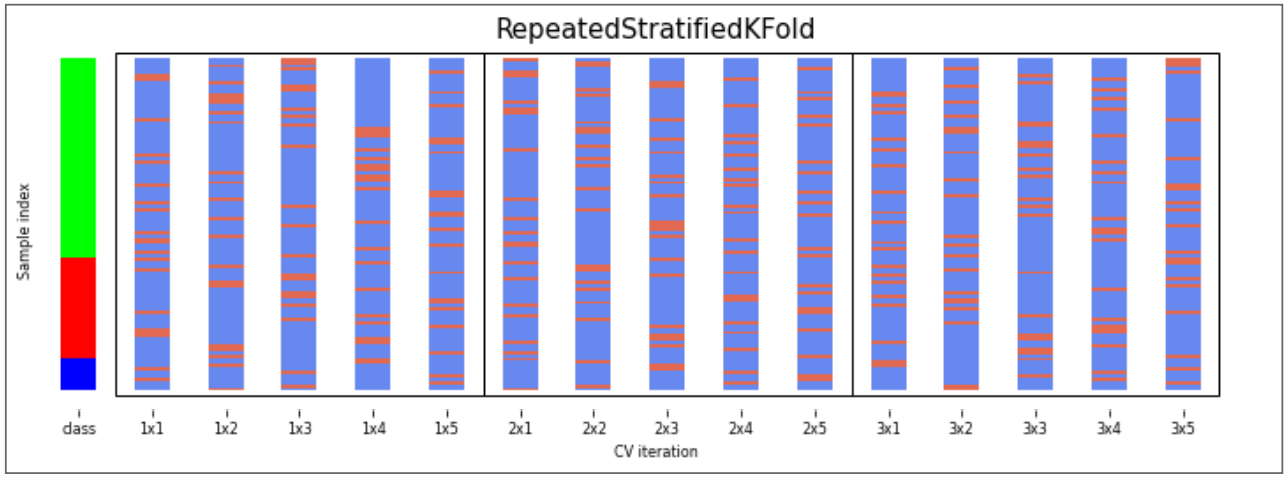
The Bootstrap

- Sample *n* (dataset size) data points, with replacement, as training set (the bootstrap)
    - On average, bootstraps include 66% of all data points (some are duplicates)
- Use the unsampled (out-of-bootstrap) samples as the test set
- Repeat k times to obtain k scores
-



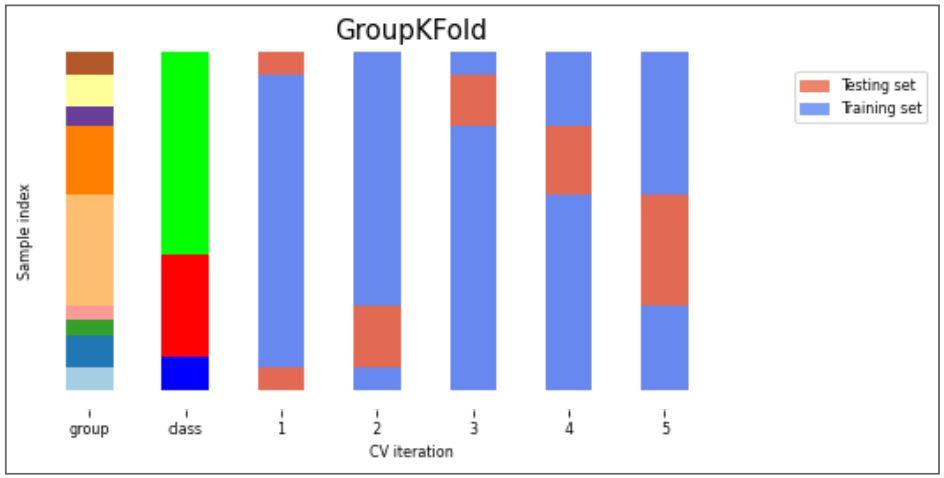
Repeated cross-validation

- Cross-validation is still biased in that the initial split can be made in many ways
- Repeated, or n-times-k-fold cross-validation:
    - Shuffle data randomly, do k-fold cross-validation
    - Repeat n times, yields n times k scores
- Unbiased, very robust, but n times more expensive



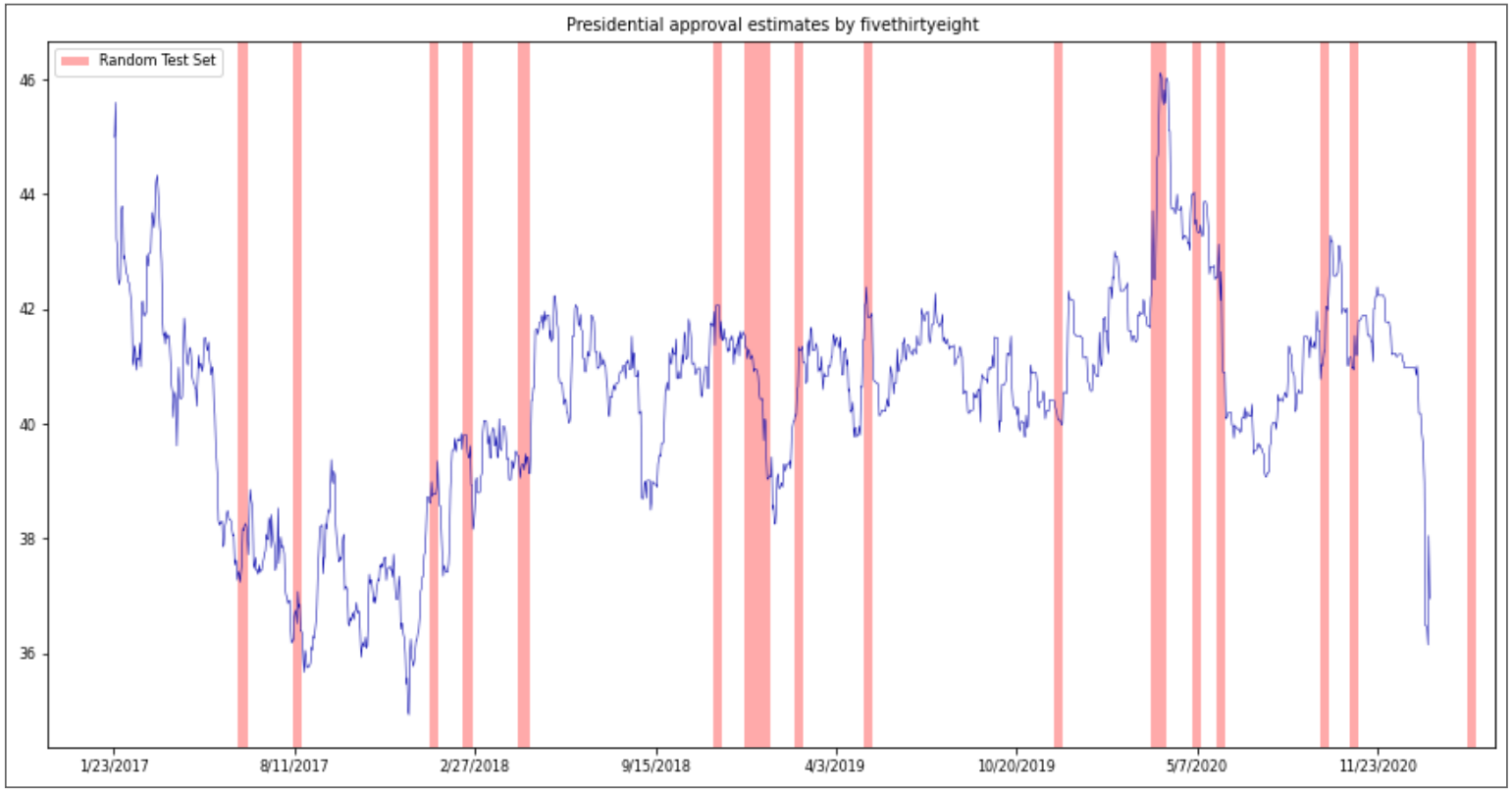
Cross-validation with groups

- Sometimes the data contains inherent groups:
    - Multiple samples from same patient, images from same person,...
- Data from the same person may end up in the training *and* test set
- We want to measure how well the model generalizes to *other* people
- Make sure that data from one person are in *either* the train or test set
    - This is called *grouping* or *blocking*
    - Leave-one-subject-out cross-validation: test set for each subject/group



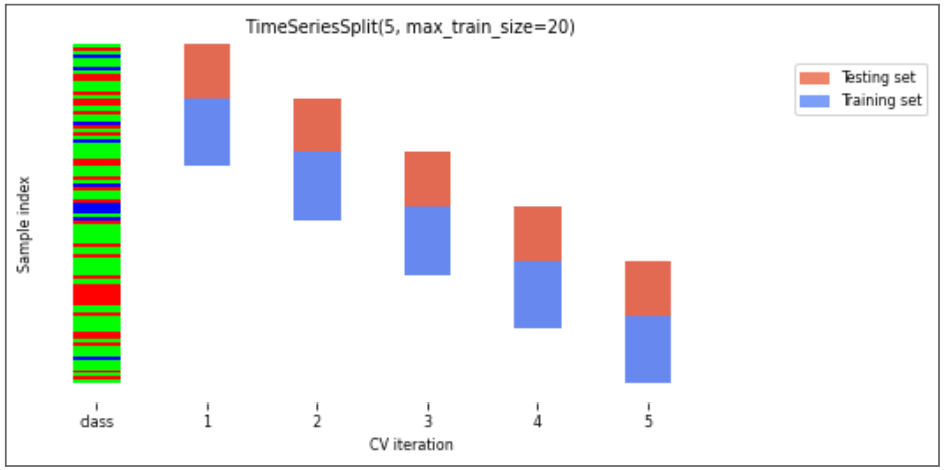
Time series

When the data is ordered, random test sets are not a good idea



Test-then-train (prequential evaluation)

- Every new sample is evaluated only once, then added to the training set
    - Can also be done in batches (of *n* samples at a time)
- `TimeSeriesSplit`
    - In the kth split, the first k folds are the train set and the (k+1)th fold as the Validation set
    - Often, a maximum training set size (or window) is used
        - more robust against concept drift (change in data over time)



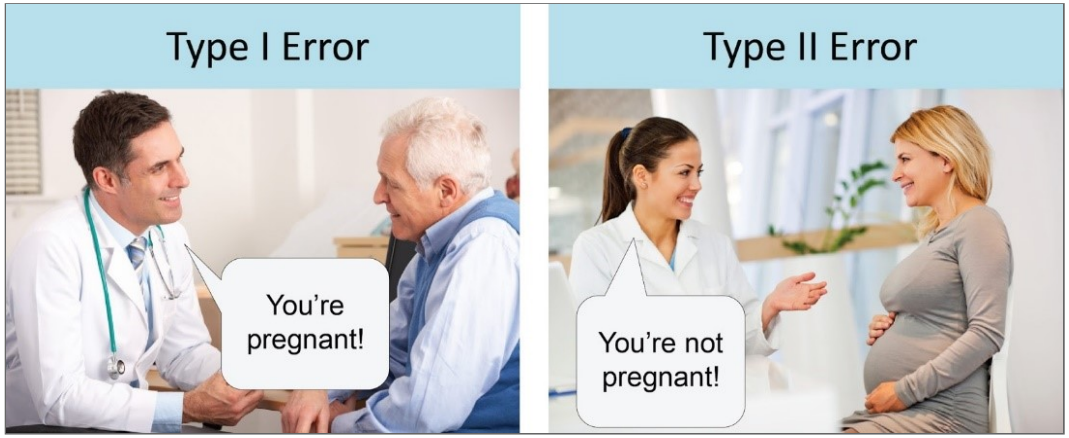
Choosing a performance estimation procedure

No strict rules, only guidelines:

- Always use stratification for classification (sklearn does this by default)
- Use holdout for very large datasets (e.g. >1.000.000 examples)
    - Or when learners don't always converge (e.g. deep learning)
- Choose *k* depending on dataset size and resources
    - Use leave-one-out for very small datasets (e.g. <100 examples)
    - Use cross-validation otherwise
        - Most popular (and theoretically sound): 10-fold CV
        - Literature suggests 5x2-fold CV is better
- Use grouping or leave-one-subject-out for grouped data
- Use train-then-test for time series

Binary classification

- We have a positive and a negative class
- 2 different kind of errors:
    - False Positive (type I error): model predicts positive while true label is negative
    - False Negative (type II error): model predicts negative while true label is positive
- They are not always equally important
    - Which side do you want to err on for a medical test?



Confusion matrices

- We can represent all predictions (correct and incorrect) in a confusion matrix
    - n by n array (n is the number of classes)
    - Rows correspond to true classes, columns to predicted classes
    - Count how often samples belonging to a class C are classified as C or any other class.
    - For binary classification, we label these true negative (TN), true positive (TP), false negative (FN), false positive (FP)

	Predicted Neg	Predicted Pos
Actual Neg	TN	FP
Actual Pos	FN	TP

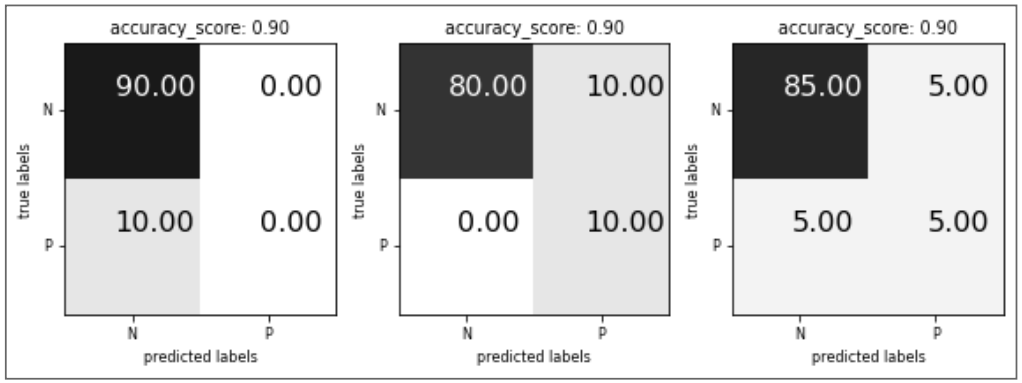
```
confusion_matrix(y_test, y_pred):
 [[48  5]
 [ 5 85]]
```

Predictive accuracy

- Accuracy can be computed based on the confusion matrix
- Not useful if the dataset is very imbalanced
    - E.g. credit card fraud: is 99.99% accuracy good enough?

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1}$$

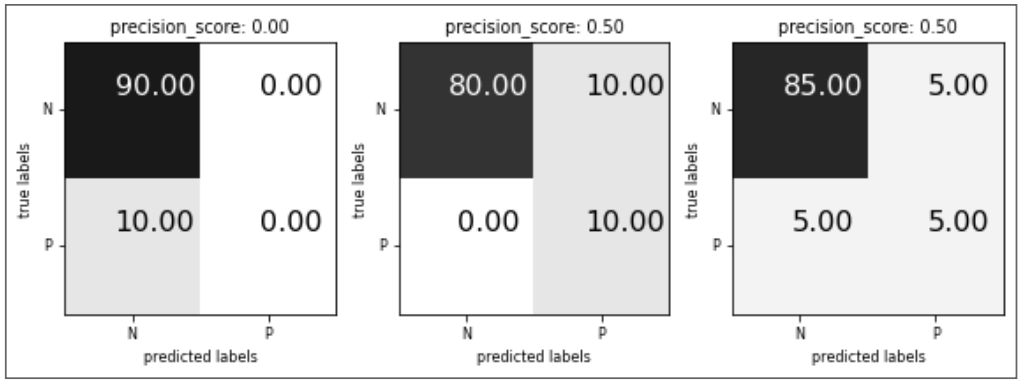
- 3 models: very different predictions, same accuracy:



Precision

- Use when the goal is to limit FPs
    - Clinical trails: you only want to test drugs that really work
    - Search engines: you want to avoid bad search results

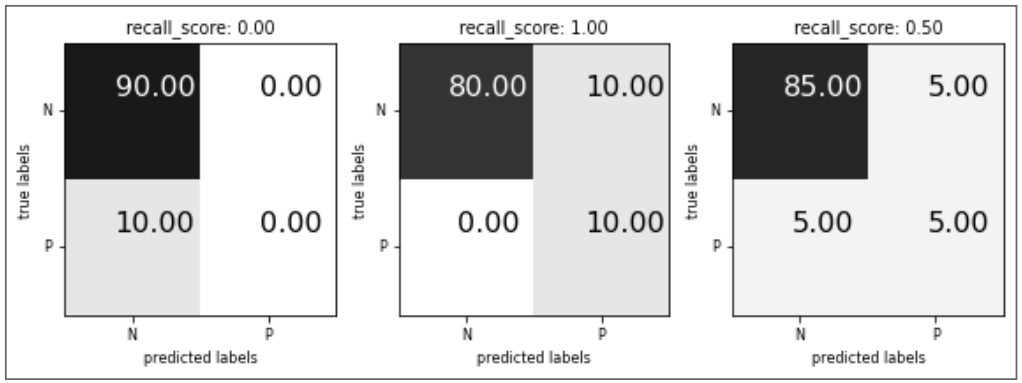
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$



Recall

- Use when the goal is to limit FNs
    - Cancer diagnosis: you don't want to miss a serious disease
    - Search engines: You don't want to omit important hits
- Also know as sensitivity, hit rate, true positive rate (TPR)

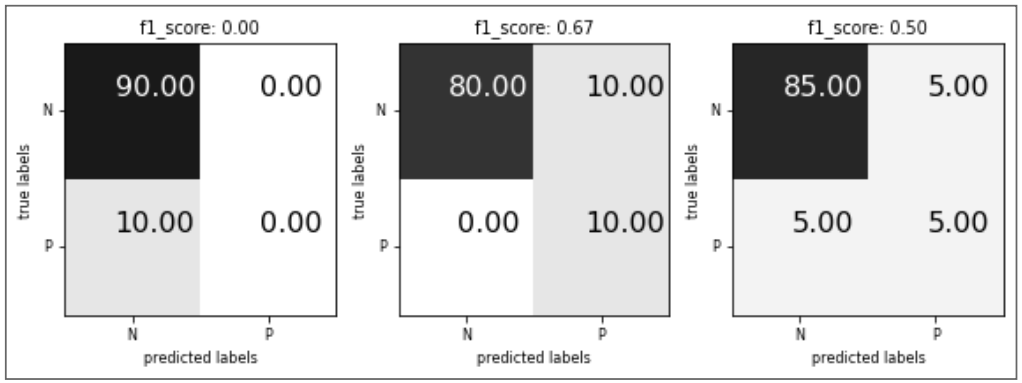
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$



F1-SCORE

- Trades off precision and recall:

$$\mathrm{F1} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} \tag{4}$$



Classification measure Zoo

		True condition				
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$	F_1 score = $\frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$		

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

Multi-class classification

- Train models *per class* : one class viewed as positive, other(s) als negative, then average
    - micro-averaging: count total TP, FP, TN, FN (every sample equally important)
        - micro-precision, micro-recall, micro-F1, accuracy are all the same

$$\text{Precision:} \frac{\sum_{c=1}^{C} \text{TP}_c}{\sum_{c=1}^{C} \text{TP}_c + \sum_{c=1}^{C} \text{FP}_c} \xrightarrow{c=2} \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Other useful classification metrics

- Cohen's Kappa
    - Measures 'agreement' between different models (aka inter-rater agreement)
    - To evaluate a single model, compare it against a model that does random guessing
        - Similar to accuracy, but taking into account the possibility of predicting the right class by chance
    - Can be weighted: different misclassifications given different weights
    - 1: perfect prediction, 0: random prediction, negative: worse than random
    - With p_0 = accuracy, and p_e = accuracy of random classifier:

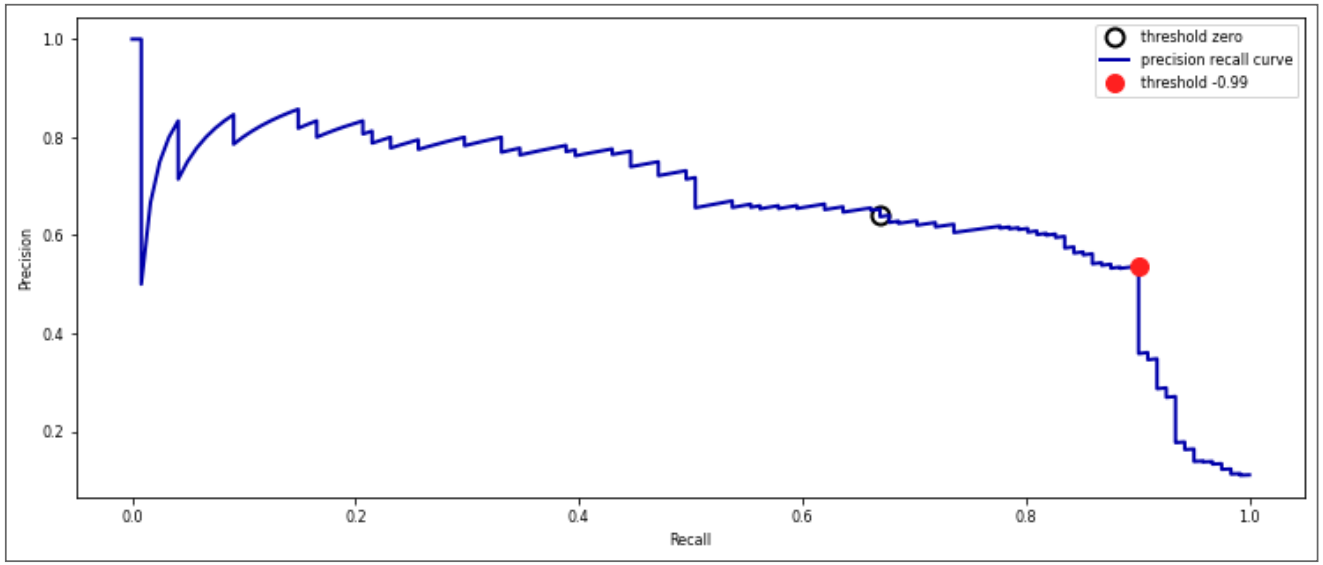
$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

- Matthews correlation coefficient
    - Corrects for imbalanced data, alternative for balanced accuracy or AUROC
    - 1: perfect prediction, 0: random prediction, -1: inverse prediction

$$MCC = \frac{tp \times tn - fp \times fn}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}}$$

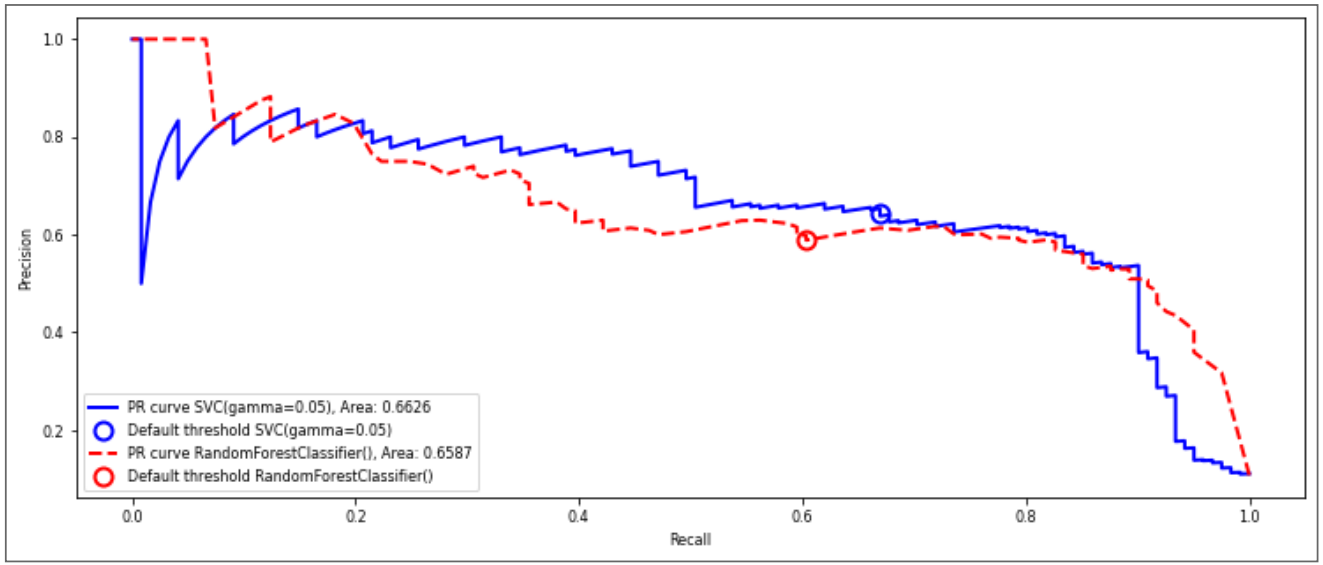
Precision-Recall curve

- The best trade-off between precision and recall depends on your application
    - You can have arbitrary high recall, but you often want reasonable precision, too.
- Plotting precision against recall *for all possible thresholds* yields a **precision-recall curve**
    - Change the treshold until you find a sweet spot in the precision-recall trade-off
    - Often jagged at high thresholds, when there are few positive examples left



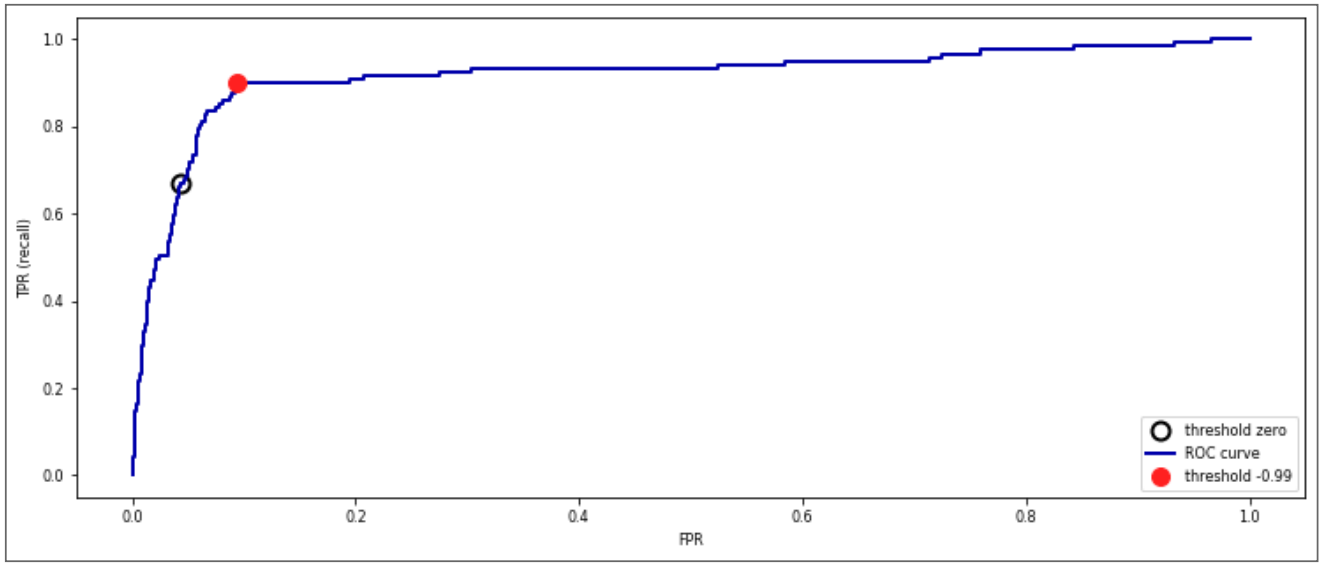
Model selection

- Some models can achieve trade-offs that others can't
- Your application may require very high recall (or very high precision)
    - Choose the model that offers the best trade-off, given your application
- The area under the PR curve (AUPRC) gives the *best overall* model



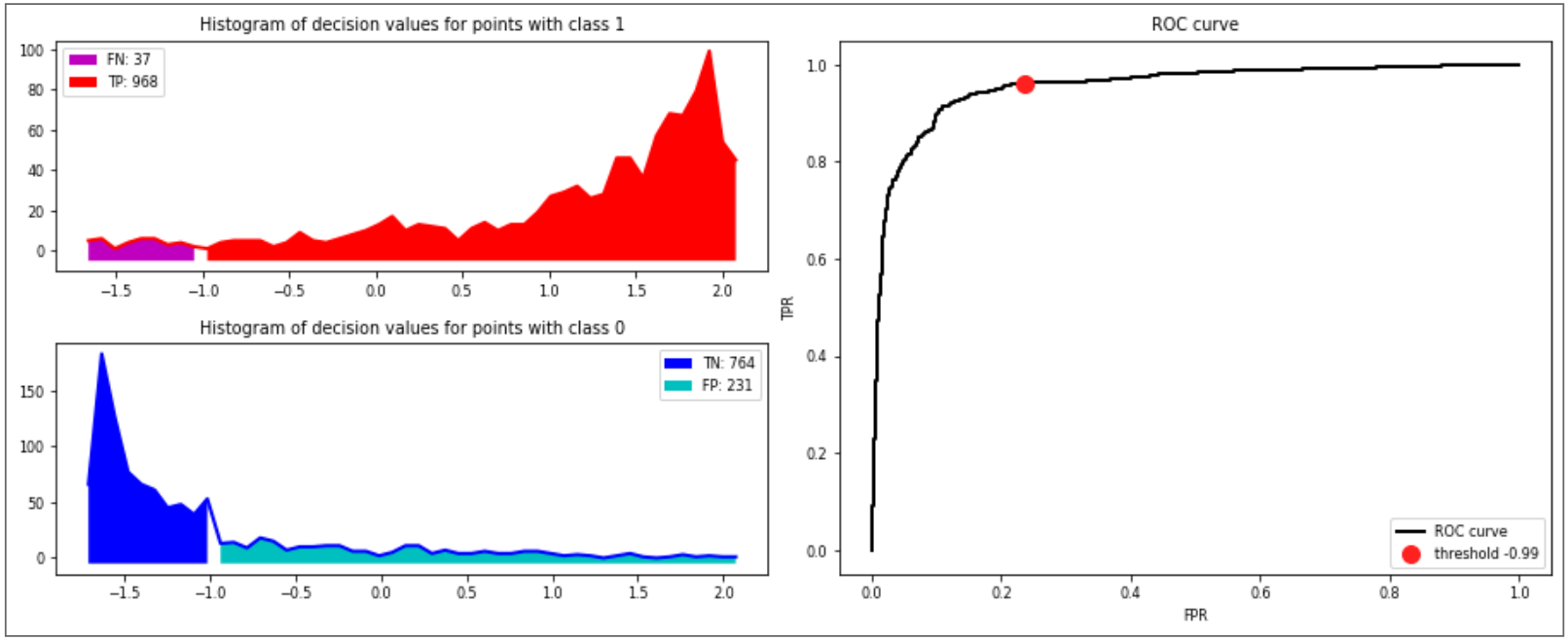
Receiver Operating Characteristics (ROC)

- Trade off *true positive rate* $TPR = \frac{TP}{TP+FN}$ with *false positive rate* $FPR = \frac{FP}{FP+TN}$
- Plotting TPR against FPR *for all possible thresholds* yields a *Receiver Operating Characteristics curve*
    - Change the treshold until you find a sweet spot in the TPR-FPR trade-off
    - Lower thresholds yield higher TPR (recall), higher FPR, and vice versa



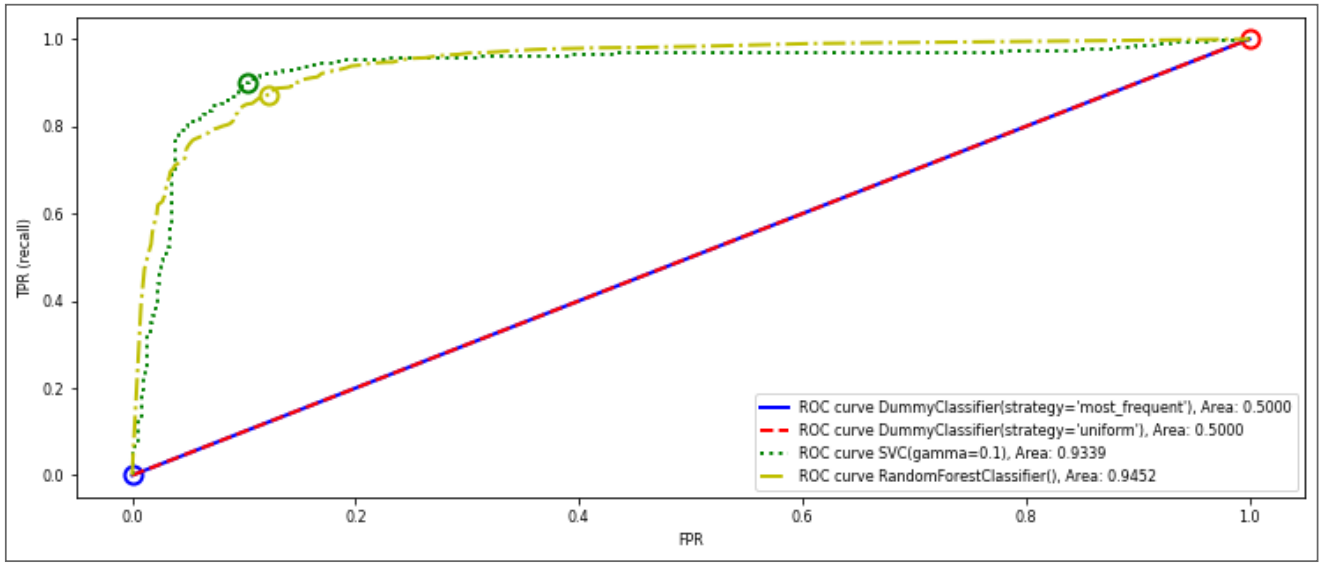
Visualization

- Histograms show the amount of points with a certain decision value (for each class)
- $TPR = \frac{TP}{TP+FN}$ can be seen from the positive predictions (top histogram)
- $FPR = \frac{FP}{FP+TN}$ can be seen from the negative predictions (bottom histogram)



Model selection

- Again, some models can achieve trade-offs that others can't
- Your application may require minizing FPR (low FP), or maximizing TPR (low FN)
- The area under the ROC curve (AUROC or AUC) gives the *best overall* model
    - Frequently used for evaluating models on imbalanced data
    - Random guessing (TPR=FPR) or predicting majority class (TPR=FPR=1): 0.5 AUC



Regression metrics

Most commonly used are

- mean squared error: $\frac{\sum_i (y_{pred_i} - y_{actual_i})^2}{n}$
    - root mean squared error (RMSE) often used as well
- mean absolute error: $\frac{\sum_i |y_{pred_i} - y_{actual_i}|}{n}$
    - Less sensitive to outliers and large errors

R squared

- $R^2 = 1 - \frac{\sum_i (y_{pred_i} - y_{actual_i})^2}{\sum_i (y_{mean} - y_{actual_i})^2}$
  - Ratio of variation explained by the model / total variation
  - Between 0 and 1, but *negative* if the model is worse than just predicting the mean
  - Easier to interpret (higher is better).