Model Selection (2)

Evaluating and selecting algorithms and hyperparameters.

Evaluation Metrics and scoring

Keep the end-goal in mind

Evaluation vs Optimization

- Each algorithm optimizes a given objective function (on the training data)
  - E.g. remember L2 loss in Ridge regression

$$L_{ridge} = \sum_i (y_i - \sum_j x_{i,j} w_j)^2 + \alpha \sum_i w_i^2$$

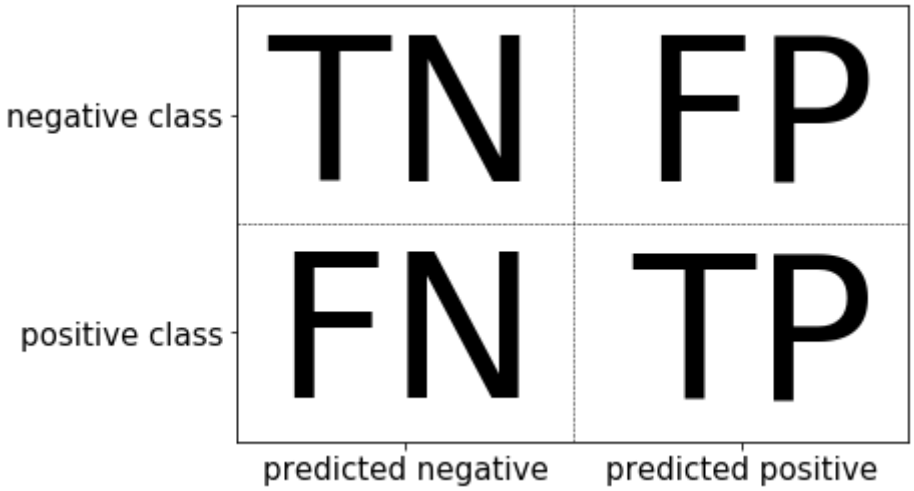
- The choice of function is limited by what can be efficiently optimized
  - E.g. gradient descent requires a differentiable loss function
- We *evaluate* the resulting model with a score that makes sense in the real world
  - E.g. percentage of correct predictions (on a test set)
- We also tune the algorithm's hyperparameters to maximize that score

Binary classification

- We have a positive and a negative class
- 2 different kind of errors:
  - False Positive (type I error): model predicts positive while the true label is negative
  - False Negative (type II error): model predicts negative while the 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 the true classes, columns to the predicted classes
  - Each entry counts how often a sample that belongs to the class corresponding to the row was classified as the class corresponding to the column.
  - For binary classification, we label these true negative (TN), true positive (TP), false negative (FN), false positive (FP)


TN
FP
FN
TP
negative class
positive class
predicted negative
predicted positive

Predictive accuracy

- Accuracy is one of the measures we can compute based on the confusion matrix:

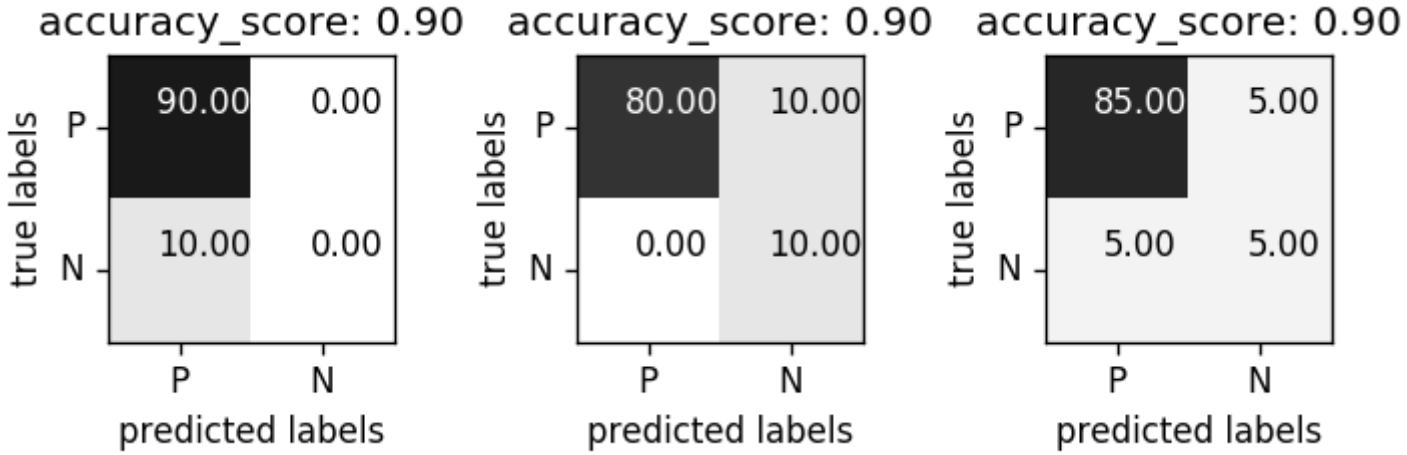
$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- In sklearn: use `confusion_matrix` and `accuracy_score` from `sklearn.metrics`.
- Accuracy is also the default evaluation measure for classification

```
confusion_matrix(y_test, y_pred):
 [[49  4]
 [ 5 85]]
accuracy_score(y_test, y_pred):  0.9370629370629371
model.score(X_test, y_test):  0.9370629370629371
```

The problem with accuracy: imbalanced datasets

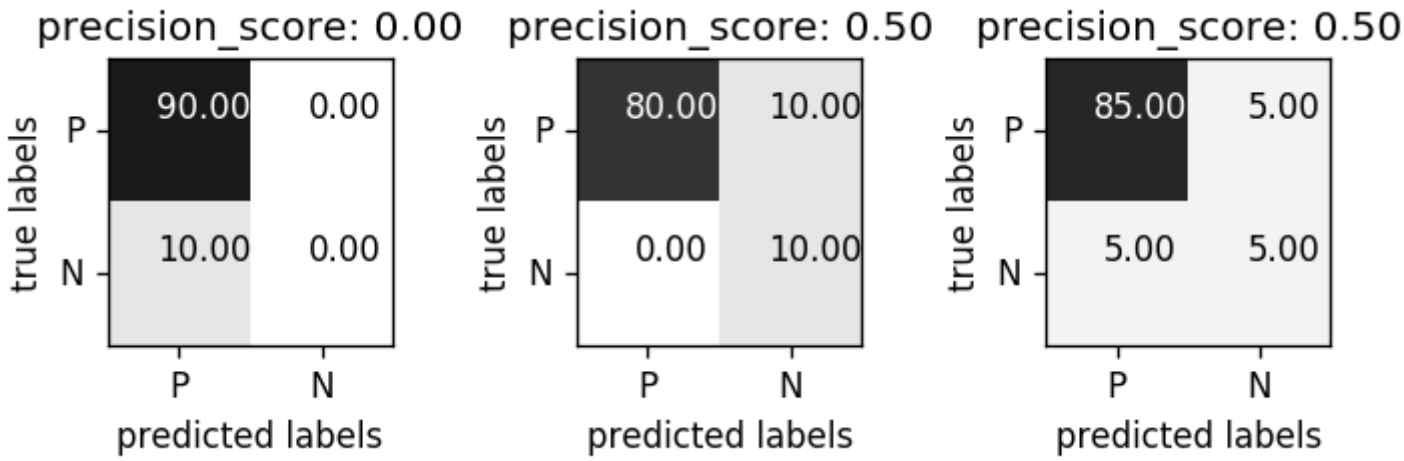
- The type of error plays an even larger role if the dataset is imbalanced
  - One class is much more frequent than the other, e.g. credit fraud
  - Is a 99.99% accuracy good enough?
- Are these three models really equally good?



Precision is used 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

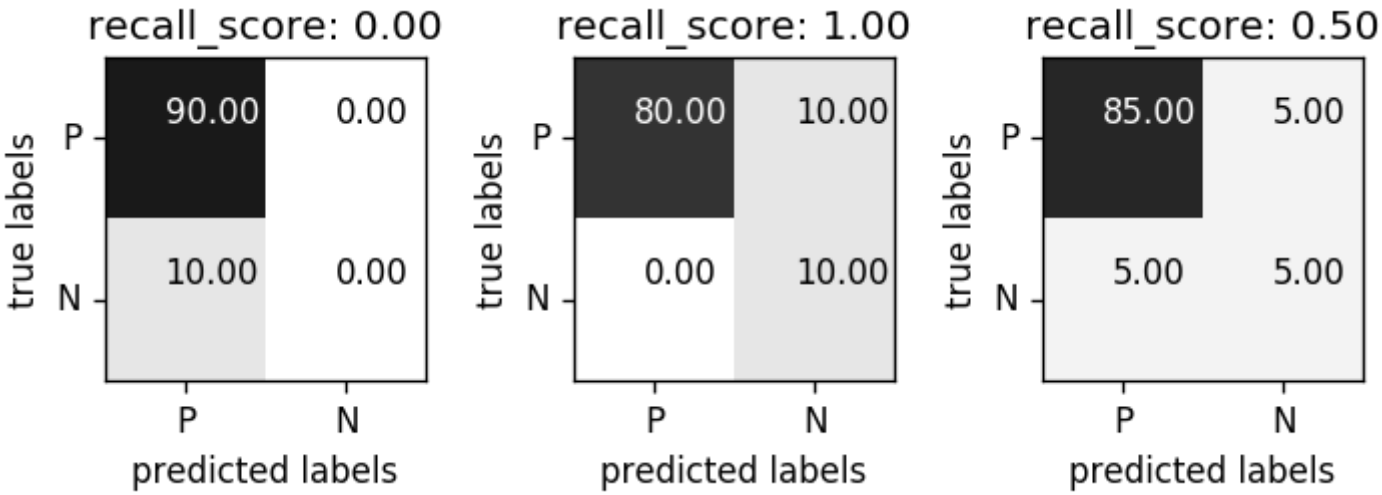
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$



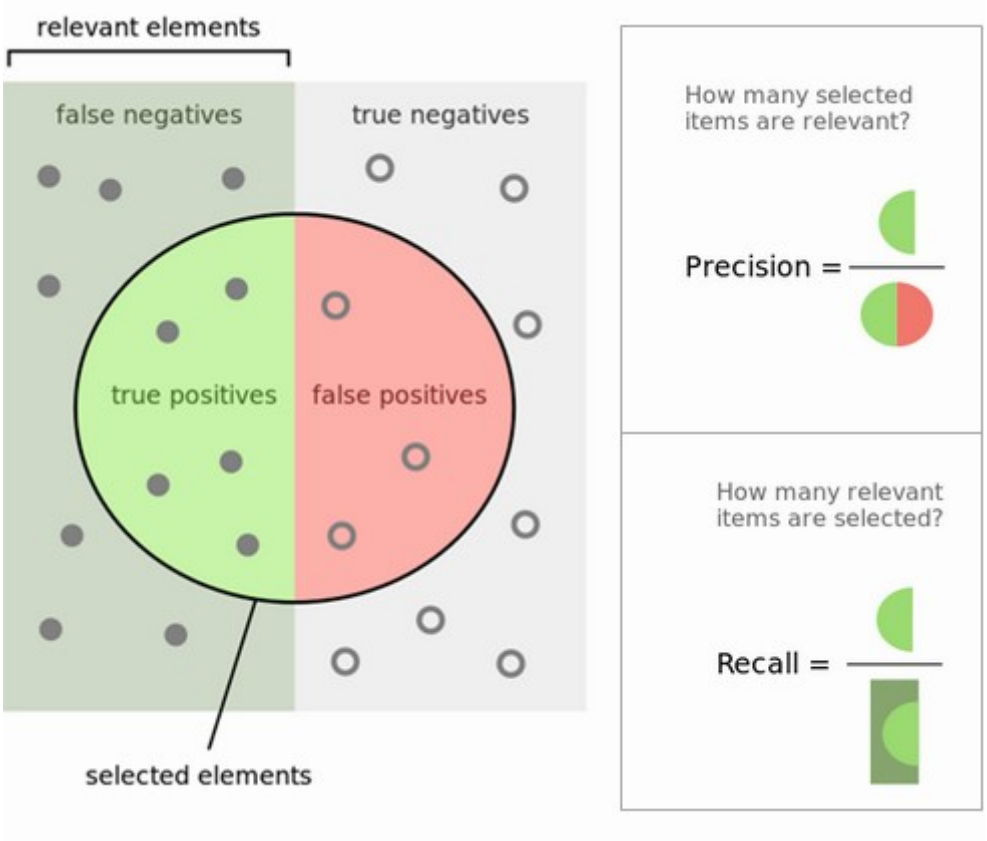
Recall is used when the goal is to limit FNs

- 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)

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

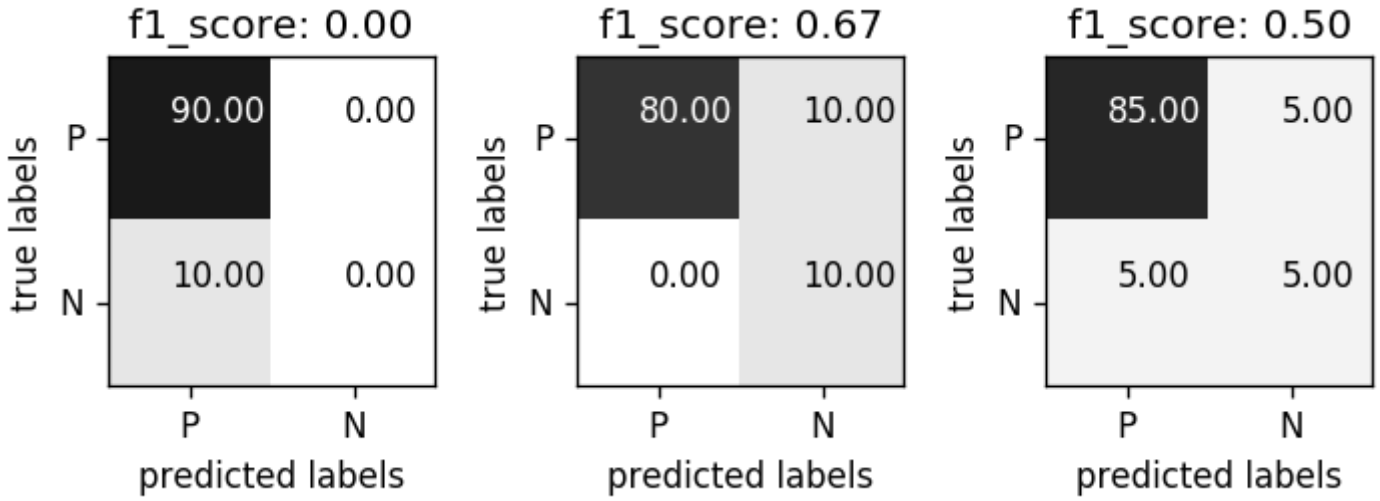


Comparison


relevant elements
false negatives
true negatives
true positives
false positives
selected elements
How many selected items are relevant?
Precision =
How many relevant items are selected?
Recall =

F1-score or F1-measure trades off precision and recall:

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



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 (https://en.wikipedia.org/wiki/Precision_and_recall)"

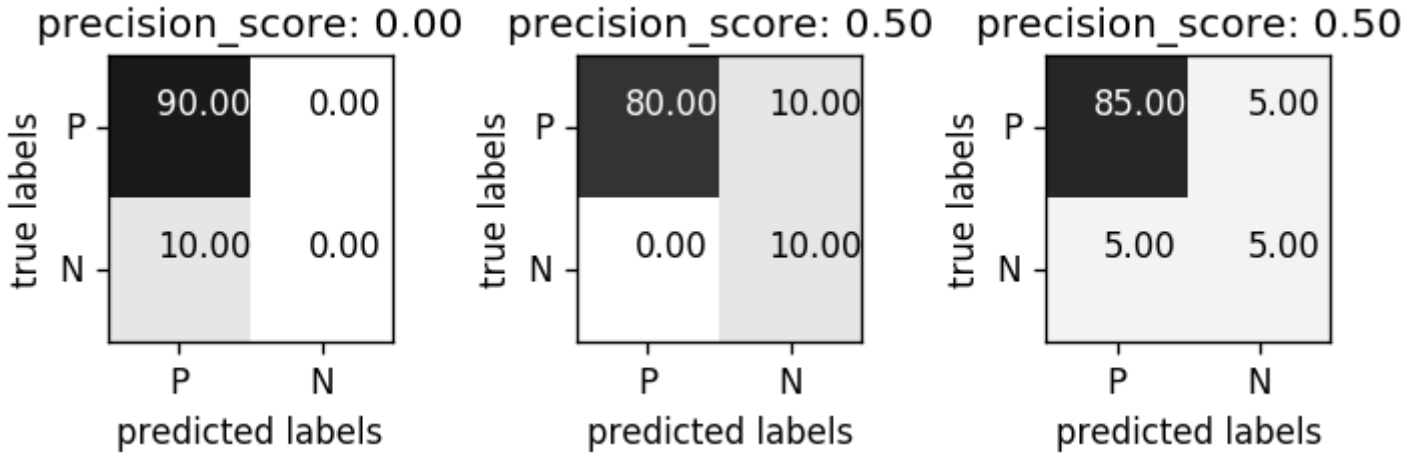
- To study the scores *by class*, use `classification_report`
  - One class viewed as positive, other(s) als negative
  - Support: number of samples in each class
  - Last line: weighted average over the classes (weighted by number of samples in each class)
- Averaging for scoring measure R across C classes (also for multiclass):
  - micro: count total number of TP, FP, TN, FN
  - macro

$$\frac{1}{C}\sum_{c\in C} R(y_c, \hat{y}_c)$$

  - weighted (w_c: ratio of examples of class c)

$$\sum_{c\in C} w_c R(y_c, \hat{y}_c)$$

Example


precision_score: 0.00
90.00
0.00
10.00
0.00
true labels
P
N
P
N
predicted labels
precision_score: 0.50
80.00
10.00
0.00
10.00
true labels
P
N
P
N
predicted labels
precision_score: 0.50
85.00
5.00
5.00
5.00
true labels
P
N
P
N
predicted labels

Matrix 1

	precision	recall	f1-score	support
0	0.90	1.00	0.95	90
1	0.00	0.00	0.00	10
micro avg	0.90	0.90	0.90	100
macro avg	0.45	0.50	0.47	100
weighted avg	0.81	0.90	0.85	100

Matrix 2

	precision	recall	f1-score	support
0	1.00	0.89	0.94	90
1	0.50	1.00	0.67	10
micro avg	0.90	0.90	0.90	100
macro avg	0.75	0.94	0.80	100
weighted avg	0.95	0.90	0.91	100

Matrix 3

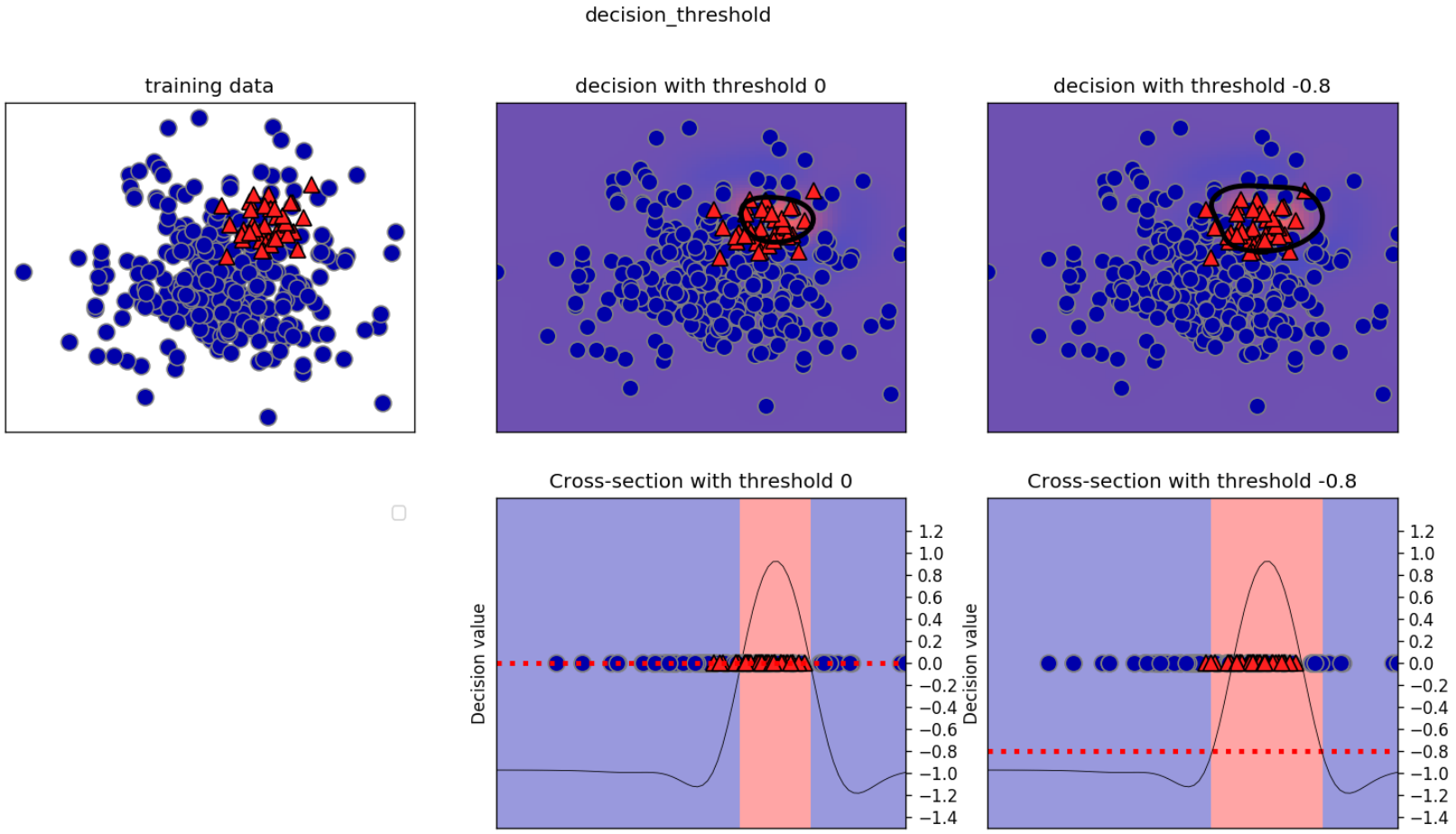
	precision	recall	f1-score	support
0	0.94	0.94	0.94	90
1	0.50	0.50	0.50	10
micro avg	0.90	0.90	0.90	100
macro avg	0.72	0.72	0.72	100
weighted avg	0.90	0.90	0.90	100

Taking uncertainty into account

- Remember that many classifiers actually return a probability per class
  - We can retrieve it with `decision_function` and `predict_proba`
- For binary classification, we threshold at 0 for `decision_function` and 0.5 for `predict_proba` by default
- However, depending on the evaluation measure, you may want to threshold differently to fit your goals
  - For instance, when a FP is much worse than a FN
  - This is called *threshold calibration*

Visualization

- Imagine that we want to avoid misclassifying a red point
- Points within decision boundary (black line) are classified positive (red)
- Lowering the decision treshold (bottom figure): fewer FN, more FP


decision_threshold
training data
decision with threshold 0
decision with threshold -0.8
Cross-section with threshold 0
Cross-section with threshold -0.8
Decision value
Decision value
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4

- Studying the classification report, we see that lowering the threshold yields:
  - higher recall for class 1 (we risk more FPs in exchange for more TP)
  - lower precision for class 1
- We can often trade off precision for recall

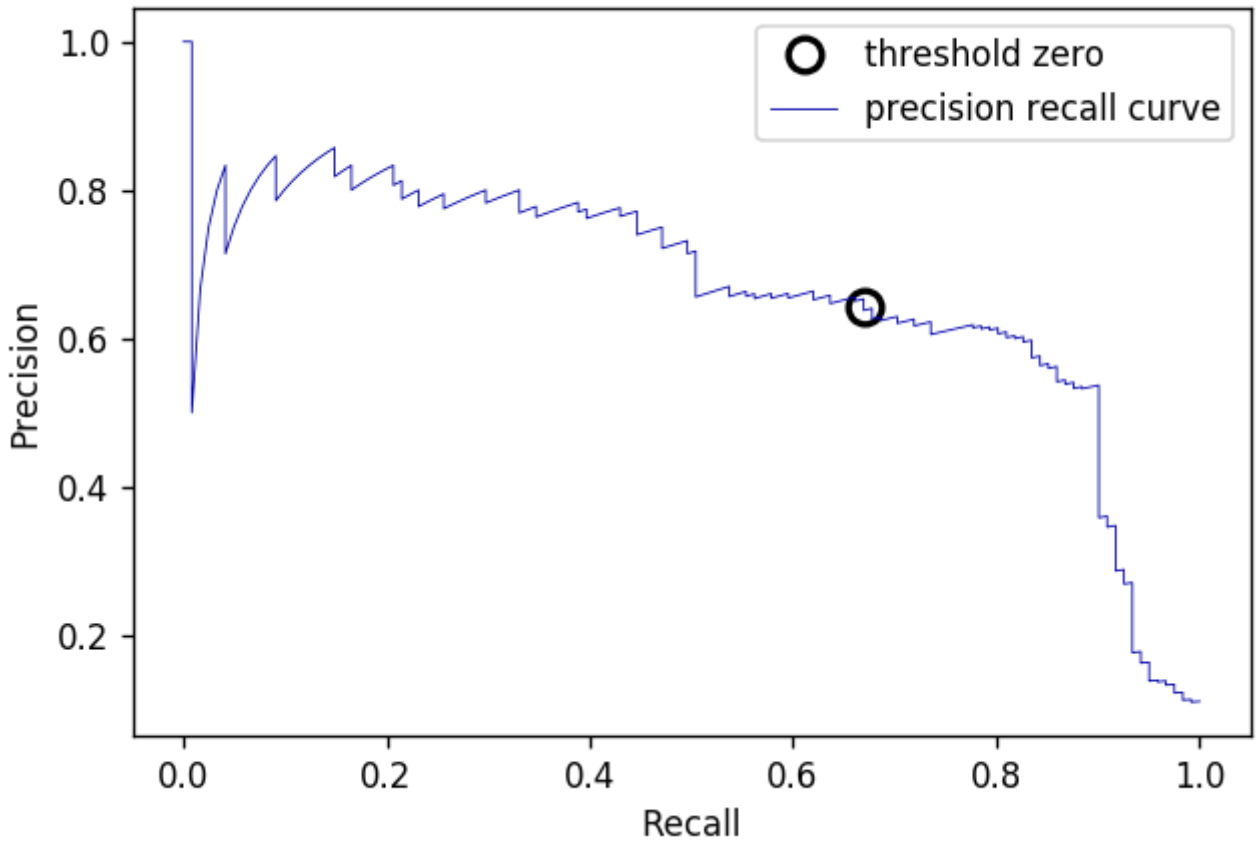
```
Threshold 0
              precision    recall  f1-score   support

           0       0.91      0.96      0.93        96
           1       0.67      0.47      0.55        17

   micro avg       0.88      0.88      0.88       113
   macro avg       0.79      0.71      0.74       113
weighted avg       0.87      0.88      0.88       113


Threshold -0.8
              precision    recall  f1-score   support

           0       0.98      0.92      0.95        96
           1       0.65      0.88      0.75        17

   micro avg       0.91      0.91      0.91       113
   macro avg       0.81      0.90      0.85       113
weighted avg       0.93      0.91      0.92       113
```

Precision-Recall curves

- The best threshold depends on your application, should be driven by real-world goals.
- You can have arbitrary high recall, but you often want reasonable precision, too.
- It is not clear beforehand where the optimale trade-off (or *operating point*) will be, so it is useful to look at all possible thresholds
- Plotting precision against recall for all thresholds yields a **precision-recall curve**

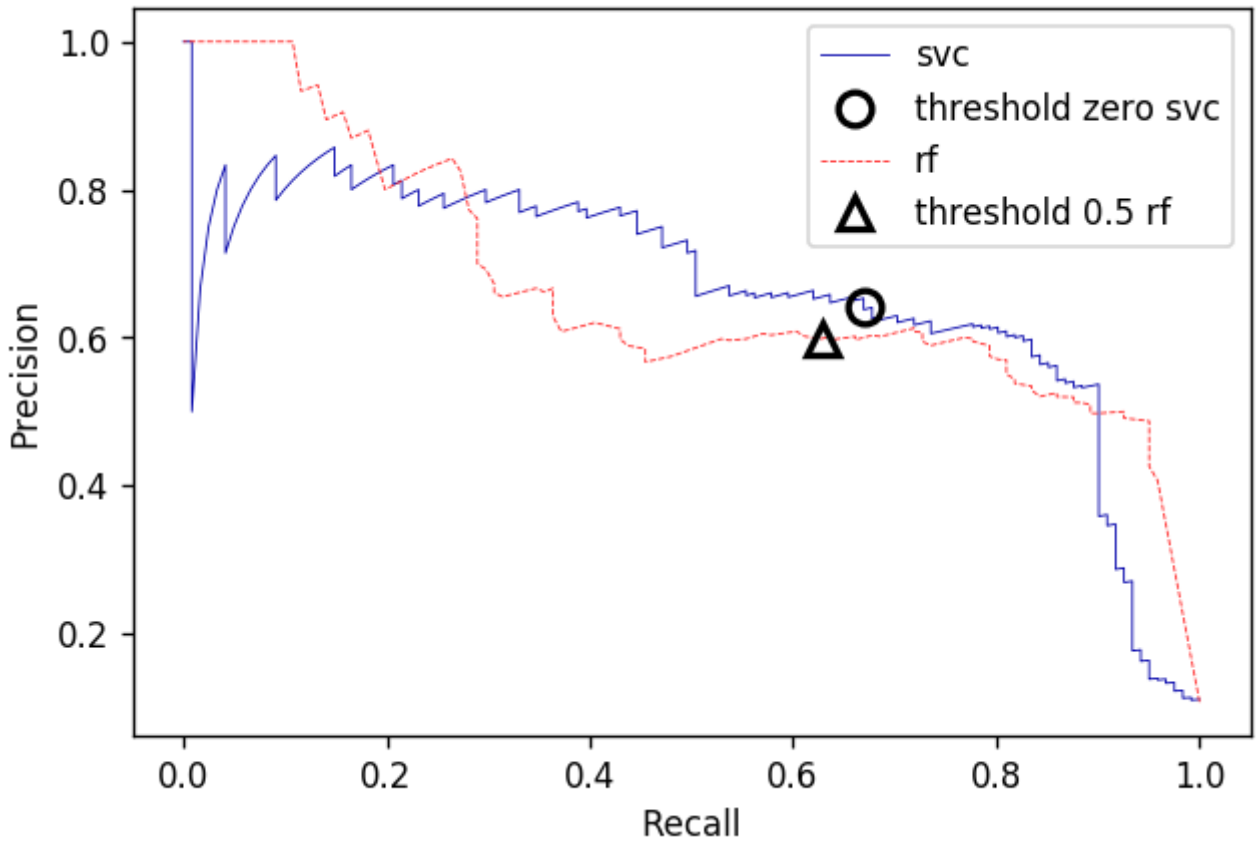
- In scikit-learn, this is included in the `sklearn.metrics` module
- Returns all precision and recall values for all thresholds
  - Vary threshold from lowest to highest decision function score in the predictions
  - Or from highest to lowest class probability

- The default tradeoff (chosen by the `predict` method) is shown as *threshold zero*.
  - Higher threshold, more precision (move left)
  - Lower threshold, more recall (move right)
- The closer the curve stays to the upper-right corner, the better
  - High precision and high recall
- Here, it is possible to still get a precision of 0.5 with high recall


threshold zero
precision recall curve
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0

Model selection

- Different classifiers work best in different parts of the curve (at different operating points)
- RandomForest (in red) performs better at the extremes, SVM better in center
- The area under the precision-recall curve (AUPRC) is often used as a general evaluation measure


svc
threshold zero svc
rf
threshold 0.5 rf
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0

Note that the F1-measure completely misses these subtleties

```
f1_score of random forest: 0.610
f1_score of svc: 0.656
```

- The area under the precision-recall curve is returned by the `average_precision_score` measure
  - It's actually a close approximation of the actual area
- This is a good automatic measure, but also hides the subtleties

```
Average precision of random forest: 0.660
Average precision of svc: 0.666
```

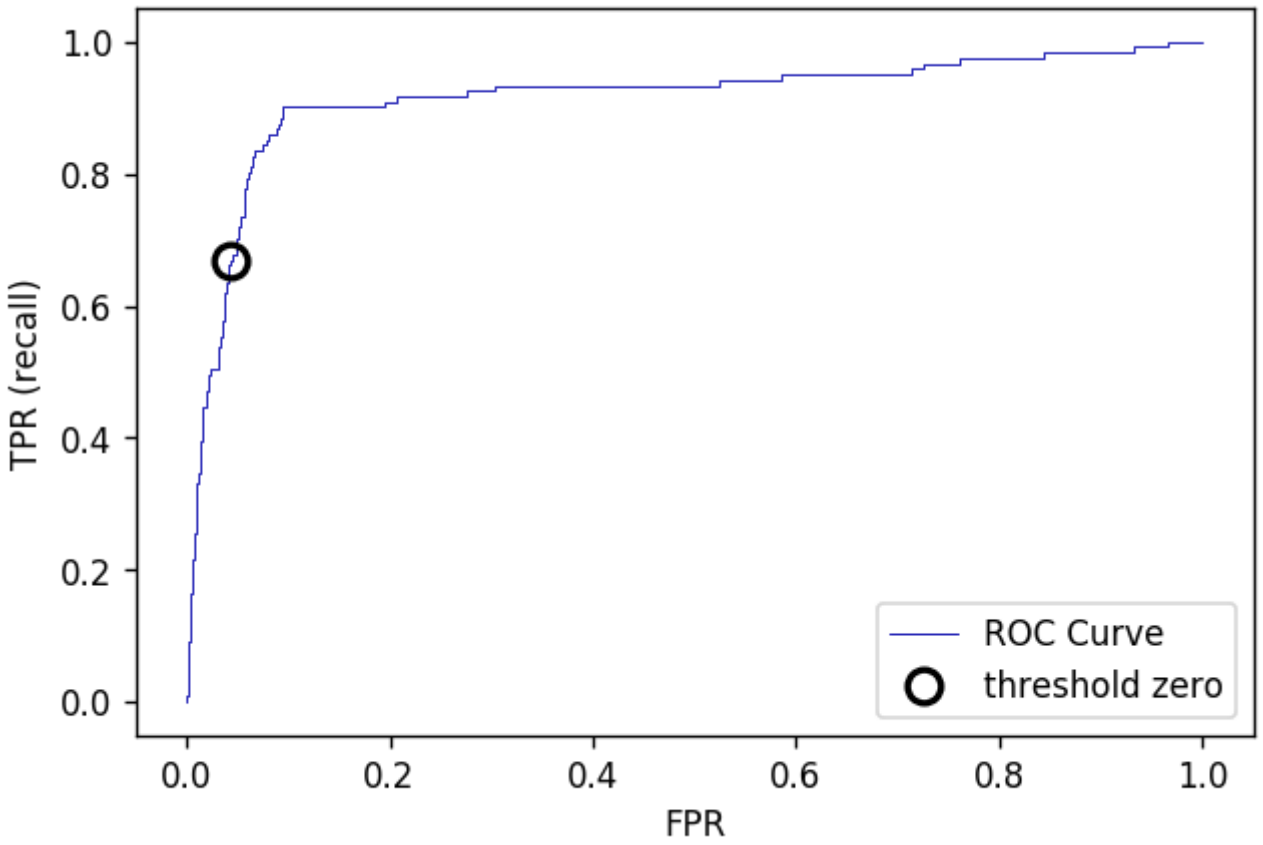
Receiver Operating Characteristics (ROC) and AUC

- There is another trade-off between recall (true positive rate, TPR) and the false positive rate (FPR).
- The 2D space created by TPR and FPR is called the Receiver Operating Characteristics (ROC) space
- A model will be at one point in this ROC space

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

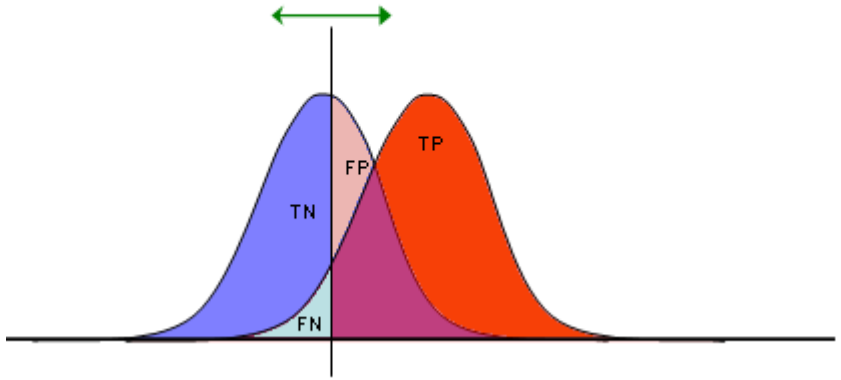
$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

- Varying the decision threshold yields the ROC curve
- It can be computed with the `roc_curve` function
  - Lower threshold, more recall/TPR, move right
  - High threshold, fewer FPs, move left
- Ideal is close to the top left: high recall, low FPR
- Inspect the curve to find the preferred calibration
  - Here, we can get much higher recall with slightly worse FPR

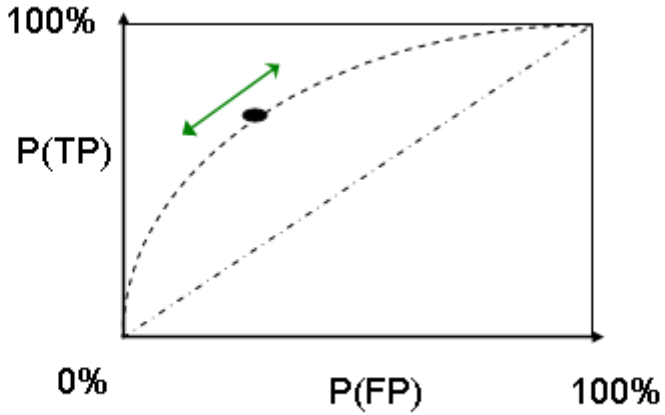

1.0
0.8
0.6
0.4
0.2
0.0
TPR (recall)
0.0
0.2
0.4
0.6
0.8
1.0
FPR
ROC Curve
threshold zero

Visualization

- The blue probability density shows the probability p(x) that the model predicts blue if a data point has a certain predicted probability x to be blue. Same for red.
- In a random classifier the probability densities completely overlap.
- All points with a predicted probability higher than the threshold are predicted positive, others negative
- As we increase the threshold, we'll get fewer FPs, more FNs. We move from right to left along the ROC curve.

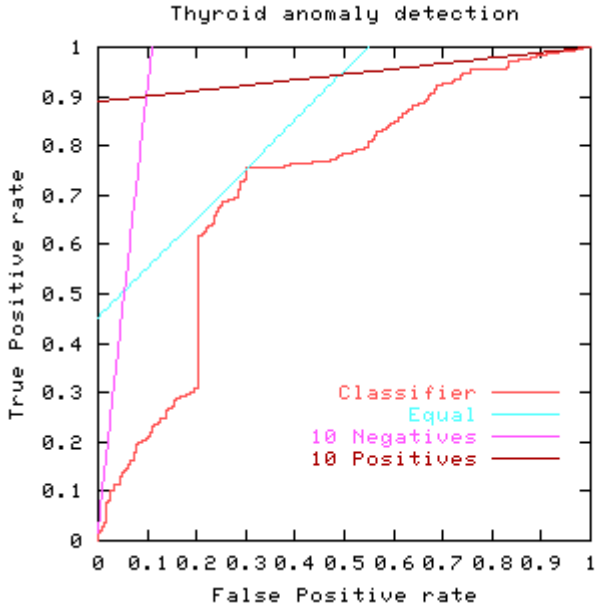


TP	FP
FN	TN
1	1



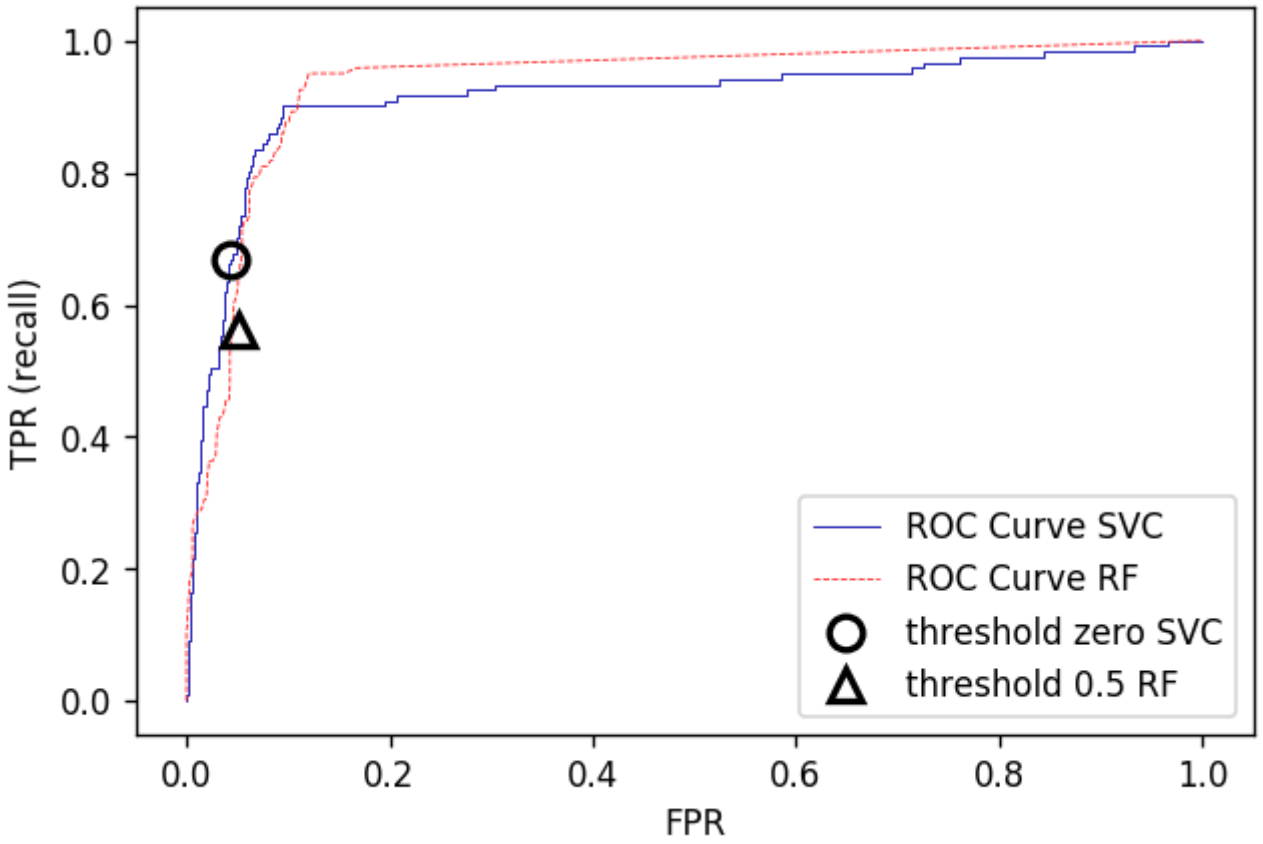
ROC Isometrics

- Different *costs* can be involved for FP and FN
- This yields different *isometrics* (lines of equal cost) in ROC space
- The optimal threshold is the point on the ROC curve where the cost in minimal
  - If a FP and FN are weigthed equally, cost lines follow the diagonal (blue line)
  - If a FP is 10 times worse than a FN: pink line
  - IF a FN is 10 times worse than a FP: red line



Model selection

- Again, we can compare multiple models by looking at the ROC curves
- We can calibrate the threshold depending on whether we need high recall or low FPR
- We can select between algorithms (or hyperparameters) depending on the involved costs.


1.0
0.8
0.6
0.4
0.2
0.0
TPR (recall)
0.0
0.2
0.4
0.6
0.8
1.0
FPR
ROC Curve SVC
ROC Curve RF
threshold zero SVC
threshold 0.5 RF

Area under the ROC curve

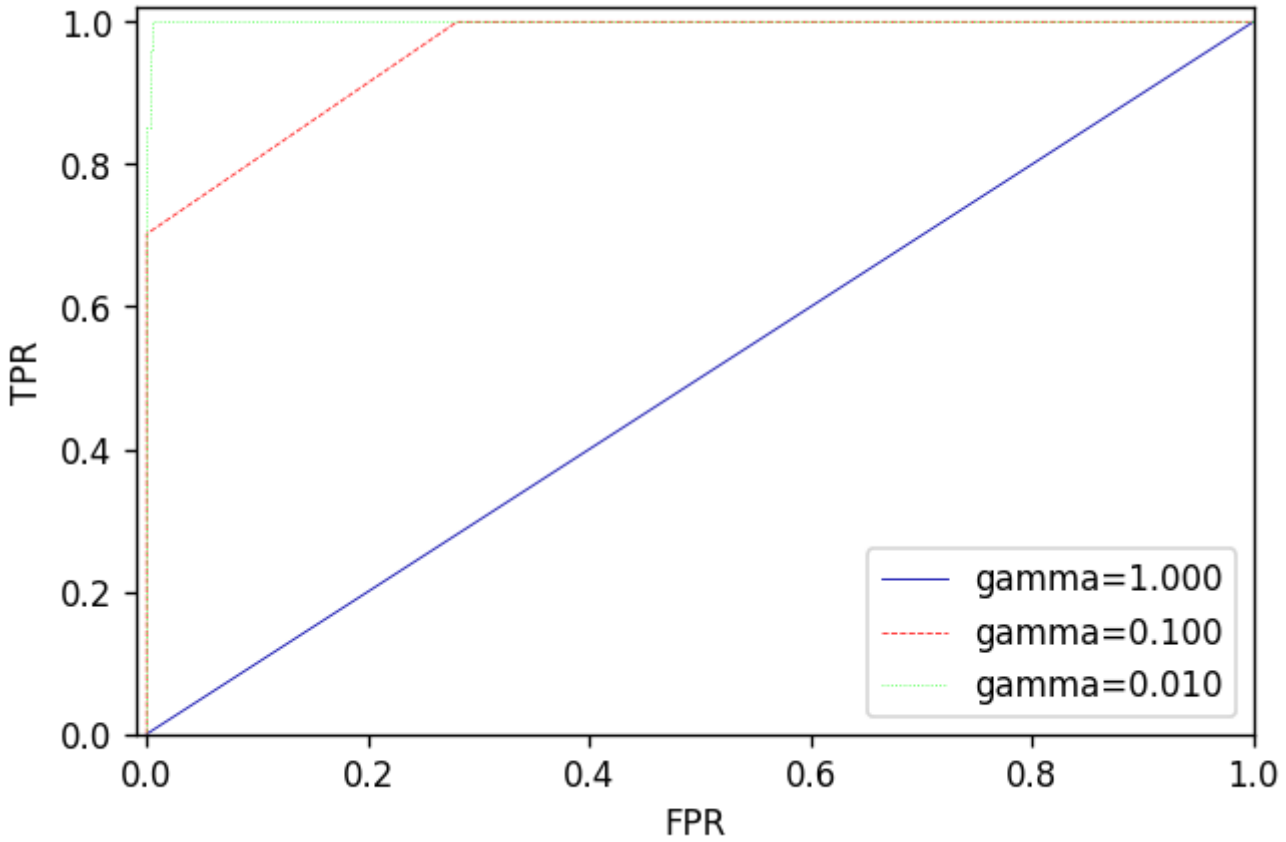
- A good summary measure is the area under the ROC curve (AUROC or AUC)
- Compute using the `roc_auc_score`
  - Don't use `auc` (uses less accurate trapezoidal rule)

```
AUC for Random Forest: 0.937
AUC for SVC: 0.916
```

Imbalanced classes

- AUC is popular because it is insensitive to class imbalance
  - Random guessing always yields TPR=FPR
  - All points are on the diagonal line, hence an AUC of 0.5
  - Hint: use the visualization of TPR,FPR to see this
- Example: unbalanced digits
  - 3 models, ACC is the same, AUC not
  - If we optimize for ACC, our model could be just random guessing

```
gamma = 1.000  accuracy = 0.90  AUC = 0.5000
gamma = 0.100  accuracy = 0.90  AUC = 0.9582
gamma = 0.010  accuracy = 0.90  AUC = 0.9995
```



Take home message

- AUC is highly recommended, especially on imbalanced data
- Remember to calibrate the threshold to your needs

Multi-class classification

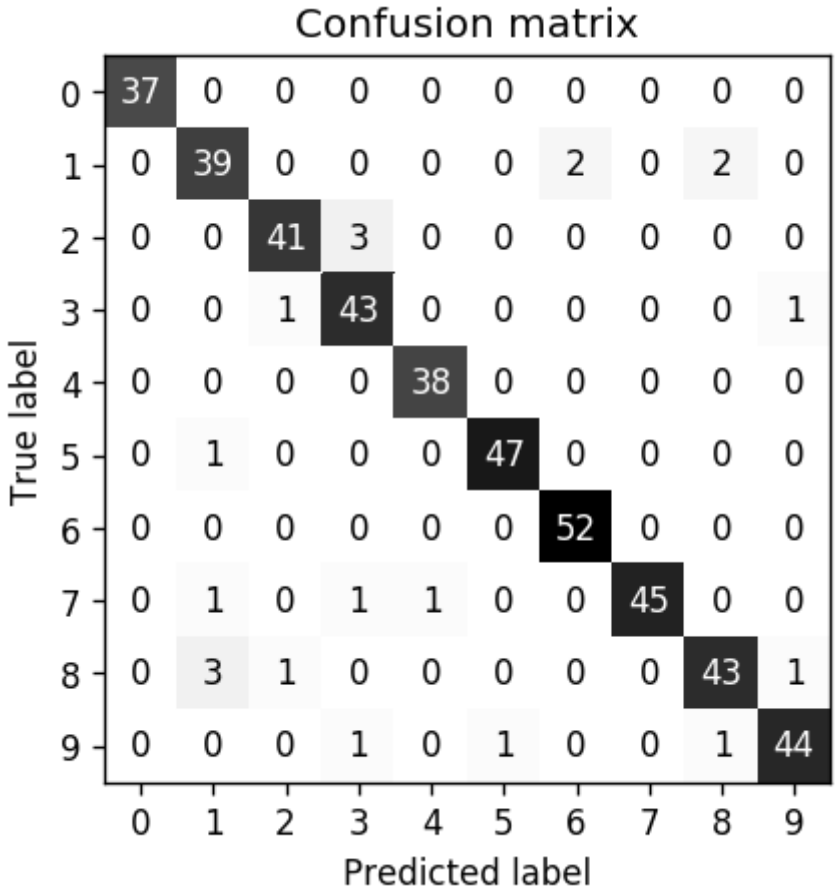
- Multiclass metrics are derived from binary metrics, averaged over all classes
- Let's consider the full (10-class) handwritten digit recognition data

Confusion matrix

```
Accuracy: 0.953
Confusion matrix:
[[37  0  0  0  0  0  0  0  0  0]
 [ 0 39  0  0  0  0  2  0  2  0]
 [ 0  0 41  3  0  0  0  0  0  0]
 [ 0  0  1 43  0  0  0  0  0  1]
 [ 0  0  0  0 38  0  0  0  0  0]
 [ 0  1  0  0  0 47  0  0  0  0]
 [ 0  0  0  0  0  0 52  0  0  0]
 [ 0  1  0  1  1  0  0 45  0  0]
 [ 0  3  1  0  0  0  0  0 43  1]
 [ 0  0  0  1  0  1  0  0  1 44]]
```

Visualized as a heatmap

- Which digits are easy to predict? Which ones are confused?



Precision, recall, F1-score now yield 10 per-class scores

	precision	recall	f1-score	support
0	1.00	1.00	1.00	37
1	0.89	0.91	0.90	43
2	0.95	0.93	0.94	44
3	0.90	0.96	0.92	45
4	0.97	1.00	0.99	38
5	0.98	0.98	0.98	48
6	0.96	1.00	0.98	52
7	1.00	0.94	0.97	48
8	0.93	0.90	0.91	48
9	0.96	0.94	0.95	47
micro avg	0.95	0.95	0.95	450
macro avg	0.95	0.95	0.95	450
weighted avg	0.95	0.95	0.95	450

Different ways to compute average

- macro-averaging: computes unweighted per-class scores: $\frac{\sum_{i=0}^{n} score_i}{n}$
  - Use when you care about each class equally much
- weighted averaging: scores are weighted by the relative size of the classes (support): $\frac{\sum_{i=0}^{n} score_i weight_i}{n}$
  - Use when data is imbalanced
- micro-averaging: computes total number of FP, FN, TP over all classes, then computes scores using these counts: $recall = \frac{\sum_{i=0}^{n} TP_i}{\sum_{i=0}^{n} TP_i + \sum_{i=0}^{n} FN_i}$
  - Use when you care about each sample equally much

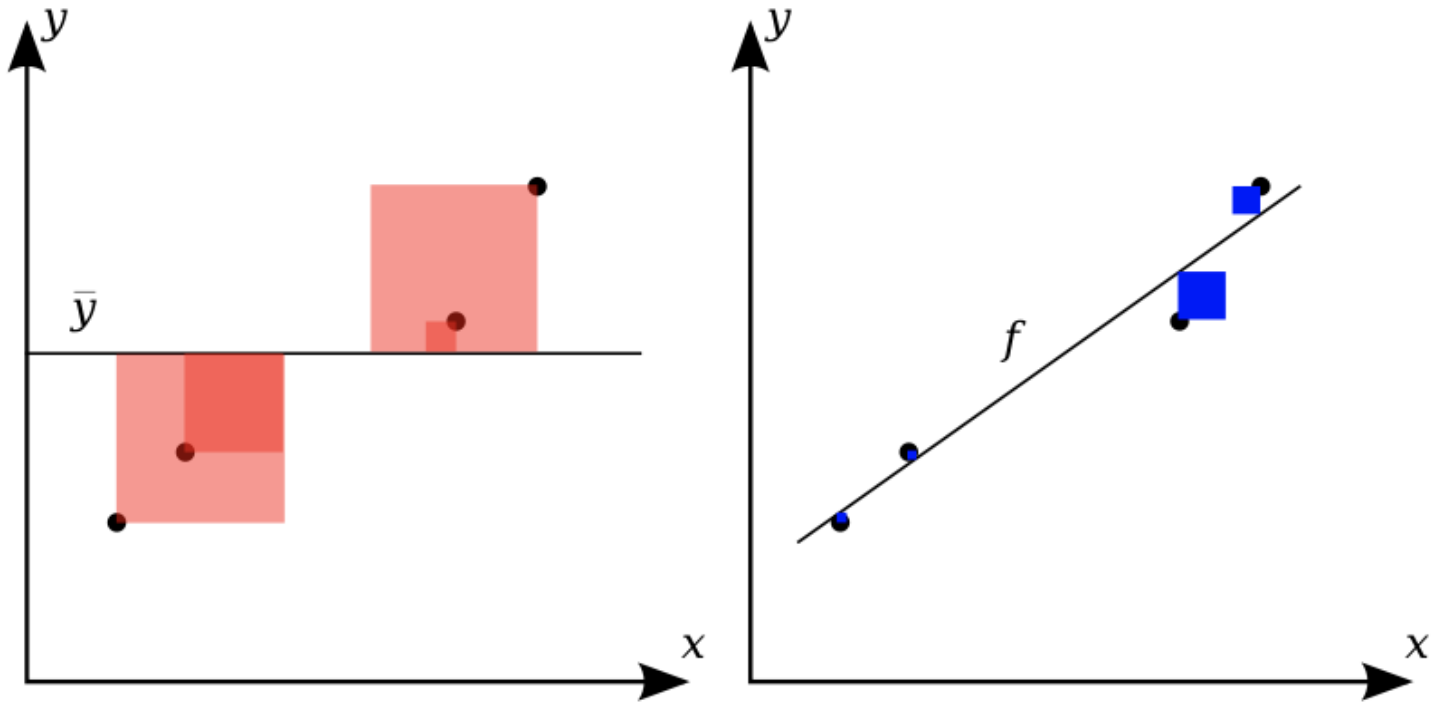
```
Micro average f1 score: 0.953
Weighted average f1 score: 0.953
Macro average f1 score: 0.954
```

Regression metrics

Most commonly used are

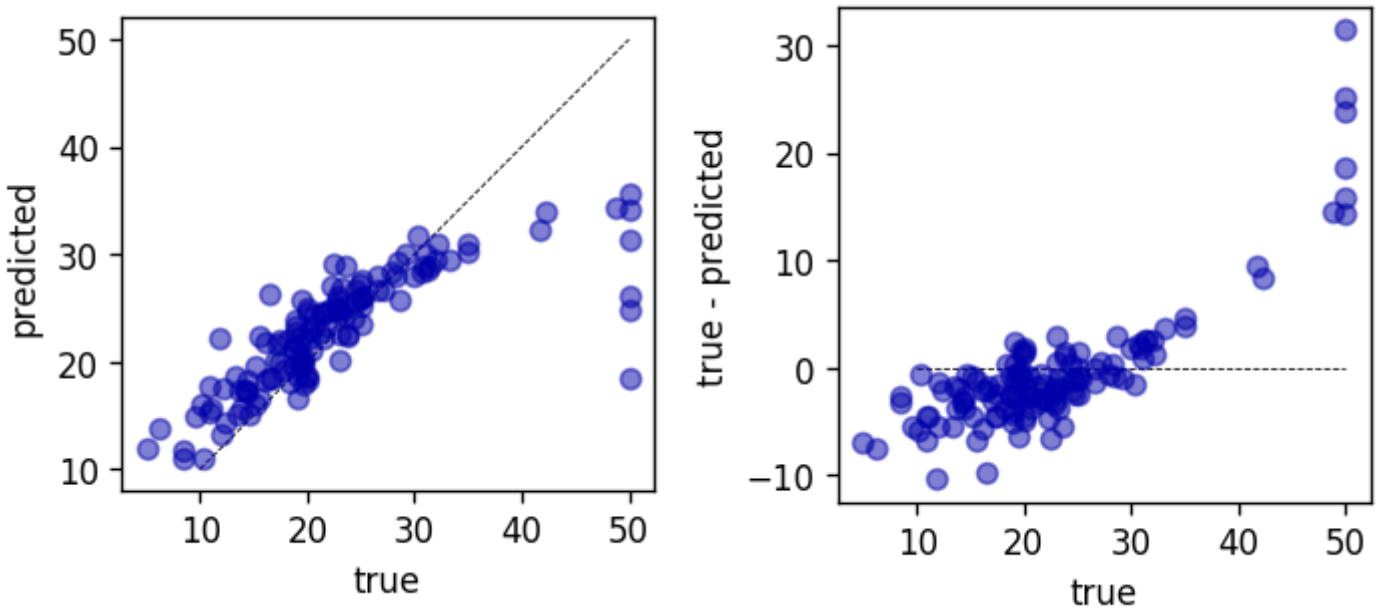
- (root) mean squared error: $\frac{\sum_i (y_{pred_i} - y_{actual_i})^2}{n}$
- mean absolute error: $\frac{\sum_i |y_{pred_i} - y_{actual_i}|}{n}$
  - Less sensitive to outliers and large errors
- R squared (r2): $1 - \frac{\sum_i (y_{pred_i} - y_{actual_i})^2}{\sum_i (y_{mean} - y_{actual_i})^2}$
  - 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).

- R squared: 1 - ratio of $\sum_i (y_{pred_i} - y_{actual_i})^2$ (blue) and $\sum_i (y_{mean} - y_{actual_i})^2$ (red)



Visualizing errors

- Prediction plot (left): predicted vs actual target values
- Residual plot (right): residuals vs actual target values
  - Over- and underpredictions can be given different costs



Using evaluation metrics in model selection

- You typically want to use AUC or other relevant measures in `cross_val_score` and `GridSearchCV` instead of the default accuracy.
- scikit-learn makes this easy through the `scoring` argument
  - But, you need to need to look the mapping between the scorer and the metric (http://scikit-learn.org/stable/modules/model_evaluation.html#model-evaluation)

Scoring	Function	Comment
Classification		
‘accuracy’	`metrics.accuracy_score`	
‘average_precision’	`metrics.average_precision_score`	
‘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’
‘roc_auc’	`metrics.roc_auc_score`	
Clustering		
‘adjusted_rand_score’	`metrics.adjusted_rand_score`	
Regression		
‘neg_mean_absolute_error’	`metrics.mean_absolute_error`	
‘neg_mean_squared_error’	`metrics.mean_squared_error`	
‘neg_median_absolute_error’	`metrics.median_absolute_error`	
‘r2’	`metrics.r2_score`	

Or simply look up like this:

```
Available scorers:
['accuracy', 'adjusted_mutual_info_score', 'adjusted_rand_score', 'average
_precision', 'balanced_accuracy', 'brier_score_loss', 'completeness_scor
e', 'explained_variance', 'f1', 'f1_macro', 'f1_micro', 'f1_samples', 'f1_
weighted', 'fowlkes_mallows_score', 'homogeneity_score', 'mutual_info_scor
e', 'neg_log_loss', 'neg_mean_absolute_error', 'neg_mean_squared_error',
'neg_mean_squared_log_error', 'neg_median_absolute_error', 'normalized_mut
ual_info_score', 'precision', 'precision_macro', 'precision_micro', 'preci
sion_samples', 'precision_weighted', 'r2', 'recall', 'recall_macro', 'reca
ll_micro', 'recall_samples', 'recall_weighted', 'roc_auc', 'v_measure_scor
e']
```

Cross-validation with accuracy and AUC

```
Default scoring: [0.9 0.9 0.9]
Explicit accuracy scoring: [0.9 0.9 0.9]
AUC scoring: [0.994 0.99  0.996]
```

Grid Search with accuracy and AUC

- With accuracy, gamma=0.0001 is selected
- With AUC, gamma=0.01 is selected
  - Actually has better accuracy on the test set

```
Grid-Search with accuracy
Best parameters: {'gamma': 0.0001}
Best cross-validation score (accuracy)): 0.970
Test set AUC: 0.992
Test set accuracy: 0.973

Grid-Search with AUC
Best parameters: {'gamma': 0.01}
Best cross-validation score (AUC): 0.997
Test set AUC: 1.000
Test set accuracy: 1.000
```

Final thoughts

- There exist techniques to correct label imbalance
  - Undersample the majority class, or oversample the minority class
  - SMOTE (Synthetic Minority Oversampling TEchnique) adds artificial *training* points by interpolating existing minority class points
    - Think twice before creating 'artificial' training data
- Cost-sensitive classification (not in sklearn)
  - *Cost matrix*: a confusion matrix with a costs associated to every possible type of error
  - Some algorithms allow optimizing on these costs instead of their usual loss function
  - Meta-cost: builds ensemble of models by relabeling training sets to match a given cost matrix
    - Black-box: can make any algorithm cost sensitive (but slower and less accurate)

Final thoughts

- There are many more metrics to choose from
  - Cohen's Kappa: accuracy, taking into account the possibility of predicting the right class by chance
    - 1: perfect prediction, 0: random prediction, negative: worse than random
    - With p_0 = accuracy, and p_e = accuracy of random classifier:

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

  - Balanced accuracy: accuracy where each sample is weighted according to the inverse prevalence of its true class
    - Identical to macro-averaged recall
  - Matthews correlation coefficient: another measure that can be used on imbalanced data
    - 1: perfect prediction, 0: random prediction, -1: inverse prediction

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

Summary

- Real-world data is often imbalanced
- False positives may be much worse than false negatives (or vise-versa)
- Binary classification
  - Select metrics that can distinguish different types of errors (precision, recall, f1-score, AUC,...)
  - Calibrate decision thresholds to the task at hand
  - Precision-Recall and ROC curves: choose the best threshold or take area under the curve
- Multiclass classification
  - Macro/Micro/weighted average of per-class scores (one-vs-all)
- Regression
  - (Root) mean squared/absolute error from 0..Inf
  - R2 easier to interpret
- All measures can be used in cross-validation or grid/random search
- Cost-sensitive classification: optimize for any cost matrix or cost function