

Applied Machine Learning

Model Evaluation & Selection

Kevyn Collins-Thompson

Associate Professor of Information & Computer Science
University of Michigan

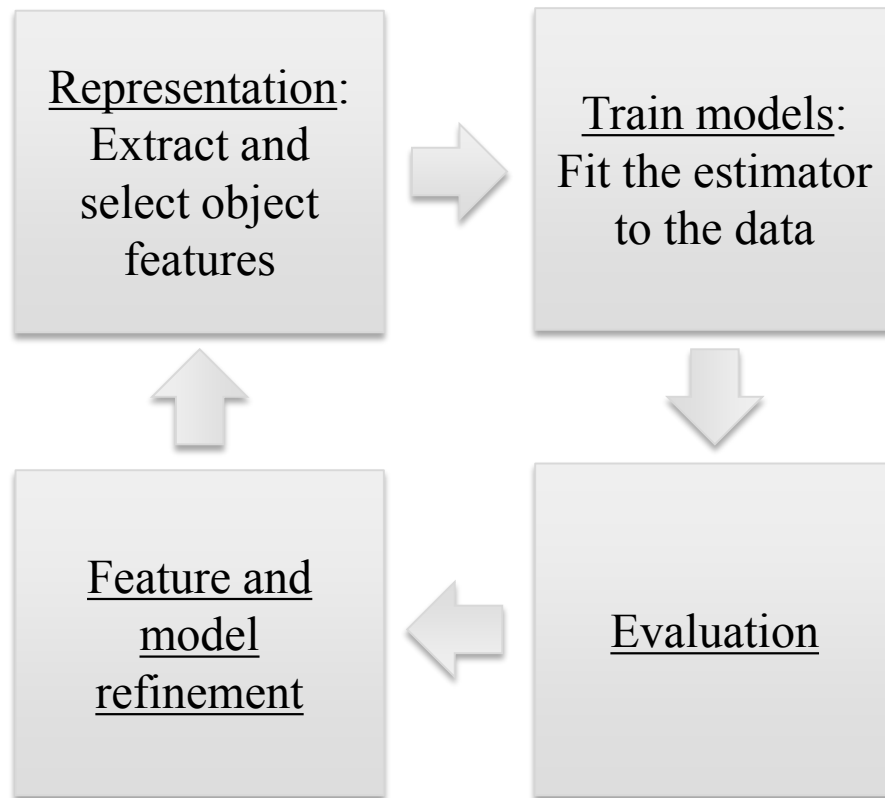
Learning objectives

- Understand why accuracy only gives a partial picture of a classifier's performance.
- Understand the motivation and definition of important evaluation metrics in machine learning.

Learning objectives

- Learn how to use a variety of evaluation metrics to evaluate supervised machine learning models.
- Learn about choosing the right metric for selecting between models or for doing parameter tuning.

Represent / Train / Evaluate / Refine Cycle



Evaluation

- Different applications have very different goals
- Accuracy is widely used, but many others are possible, e.g.:
 - User satisfaction (Web search)
 - Amount of revenue (e-commerce)
 - Increase in patient survival rates (medical)

Evaluation

- It's very important to choose evaluation methods that match the goal of your application.
- Compute your selected evaluation metric for multiple different models.
- Then select the model with 'best' value of evaluation metric.

Accuracy with imbalanced classes

- Suppose you have two classes:
 - Relevant (R): the positive class
 - Not_Relevant (N): the negative class
- Out of 1000 randomly selected items, on average
 - One item is relevant and has an R label
 - The rest of the items (999 of them) are not relevant and labelled N.
- Recall that:

$$\text{Accuracy} = \frac{\text{\#correct predictions}}{\text{\#total instances}}$$

Accuracy with imbalanced classes

- You build a classifier to predict relevant items, and see that its accuracy on a test set is **99.9%**.
- Wow! Amazingly good, right?
- For comparison, suppose we had a "dummy" classifier that didn't look at the features at all, and always just blindly predicted the most frequent class (i.e. the negative N class).

Accuracy with imbalanced classes

- Assuming a test set of 1000 instances, what would this dummy classifier's accuracy be?
- Answer:

$$\text{Accuracy}_{\text{DUMMY}} = 999 / 1000 = 99.9\%$$

Dummy classifiers completely ignore the input data!

- Dummy classifiers serve as a sanity check on your classifier's performance.
- They provide a null metric (e.g. null accuracy) baseline.
- Dummy classifiers should not be used for real problems.

Dummy classifiers completely ignore the input data!

- Some commonly-used settings for the `strategy` parameter for `DummyClassifier` in `scikit-learn`:
 - ***most_frequent*** : predicts the most frequent label in the training set.
 - ***stratified*** : random predictions based on training set class distribution.
 - ***uniform*** : generates predictions uniformly at random.
 - ***constant*** : always predicts a constant label provided by the user.
 - A major motivation of this method is F1-scoring, when the positive class is in the minority.

What if my classifier accuracy is close to the null accuracy baseline?

This could be a sign of:

- Ineffective, erroneous or missing features
- Poor choice of kernel or hyperparameter
- Large class imbalance

Dummy regressors

`strategy` parameter options:

- *mean* : predicts the mean of the training targets.
- *median* : predicts the median of the training targets.
- *quantile* : predicts a user-provided quantile of the training targets.
- *constant* : predicts a constant user-provided value.

Binary prediction outcomes

<u>True</u> negative	TN	FP
<u>True</u> positive	FN	TP
	<u>Predicted</u> negative	<u>Predicted</u> positive

Label 1 = positive class
(class of interest)

Label 0 = negative class
(everything else)

TP = true positive

FP = false positive (Type I error)

TN = true negative

FN = false negative (Type II error)

Confusion matrix for binary prediction task

True negative	TN = 356	FP = 51
	FN = 38	TP = 5
True positive	Predicted negative	Predicted positive

N = 450

- Every test instance is in exactly one box (integer counts).
- Breaks down classifier results by error type.
- Thus, provides more information than simple accuracy.
- Helps you choose an evaluation metric that matches project goals.
- Not a single number like accuracy.. but there are many possible metrics that can be derived from the confusion matrix.

Applied Machine Learning

Confusion Matrices & Basic Evaluation Metrics

Kevyn Collins-Thompson

Associate Professor of Information & Computer Science
University of Michigan

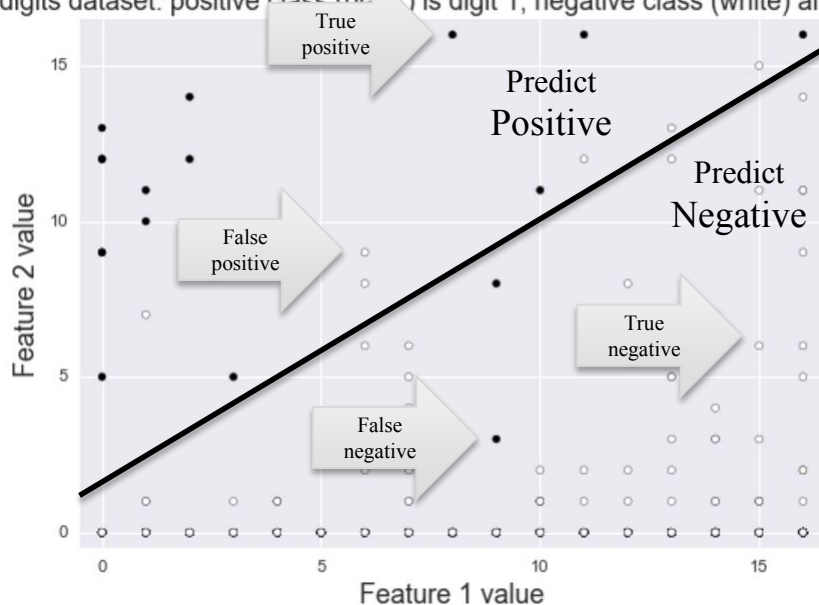
Confusion Matrix for Binary Prediction Task

True negative	TN = 400	FP = 7	
True positive	FN = 17	TP = 26	
	Predicted negative	Predicted positive	$N = 450$

*Always look at the
confusion matrix for
your classifier.*

Visualization of Different Error Types

digits dataset: positive class (black) is digit 1, negative class (white) all others



TN = 429

FP = 6

FN = 2

TP = 13

Accuracy: for what fraction of all instances is the classifier's prediction correct (for either positive or negative class)?

True negative	TN = 400	FP = 7	
True positive	FN = 17	TP = 26	
	Predicted negative	Predicted positive	$N = 450$

$$\begin{aligned}\text{Accuracy} &= \frac{TN+TP}{TN+TP+FN+FP} \\ &= \frac{400+26}{400+26+17+7} \\ &= 0.95\end{aligned}$$

Classification error ($1 - \text{Accuracy}$): for what fraction of all instances is the classifier's prediction incorrect?

True negative	TN = 400	FP = 7	
True positive	FN = 17	TP = 26	
	Predicted negative	Predicted positive	$N = 450$

$$\begin{aligned}\text{ClassificationError} &= \frac{FP + FN}{TN + TP + FN + FP} \\ &= \frac{7+17}{400+26+17+7} \\ &= 0.060\end{aligned}$$

Recall, or True Positive Rate (TPR): what fraction of all positive instances does the classifier correctly identify as positive?

True negative	TN = 400	FP = 7	
True positive	FN = 17	TP = 26	
	Predicted negative	Predicted positive	$N = 450$

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

$$= \frac{26}{26 + 17}$$

$$= 0.60$$

Recall is also known as:

- True Positive Rate (TPR)
- Sensitivity
- Probability of detection

Precision: what fraction of positive predictions are correct?

True negative	TN = 400	FP = 7	
True positive	FN = 17	TP = 26	
	Predicted negative	Predicted positive	$N = 450$

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

$$= \frac{26}{26+7}$$

$$= 0.79$$

False positive rate (FPR): what fraction of all negative instances does the classifier incorrectly identify as positive?

True negative	TN = 400	FP = 7	
True positive	FN = 17	TP = 26	
	Predicted negative	Predicted positive	$N = 450$

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

$$= \frac{7}{400+7}$$

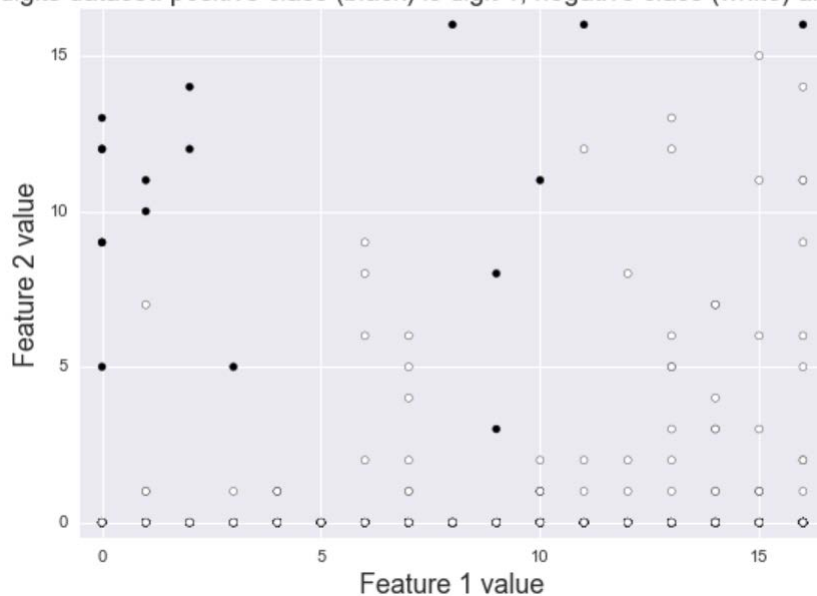
$$= 0.02$$

False Positive Rate is also known as:

- Specificity

A Graphical Illustration of Precision & Recall

digits dataset: positive class (black) is digit 1, negative class (white) all others



TN =

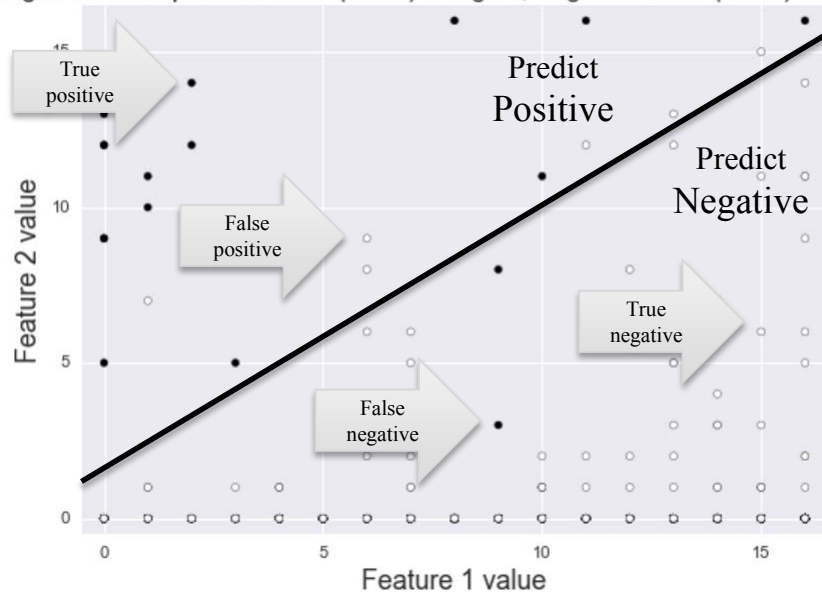
FP =

FN =

TP =

The Precision-Recall Tradeoff

digits dataset: positive class (black) is digit 1, negative class (white) all others



$$TN = 429$$

$$FP = 6$$

$$FN = 2$$

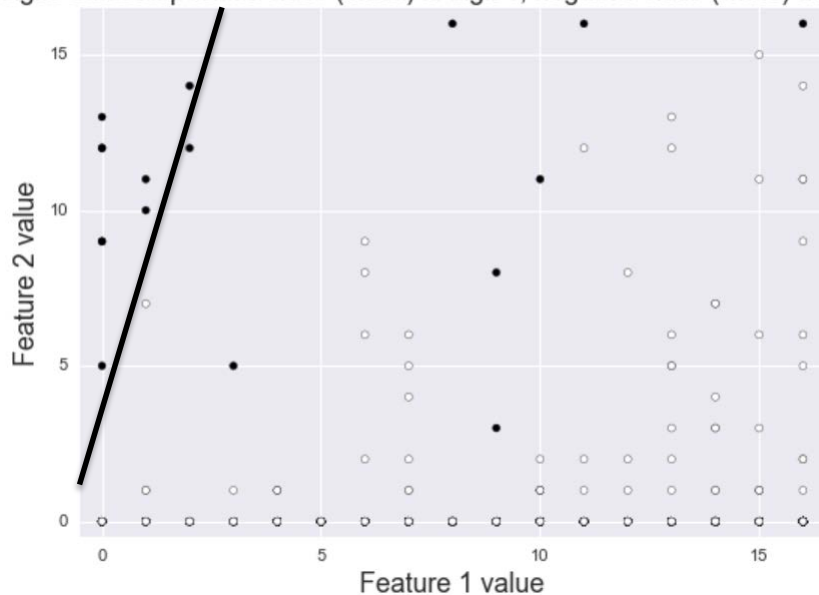
$$TP = 13$$

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{13}{19} = 0.68$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{13}{15} = 0.87$$

High Precision, Lower Recall

digits dataset: positive class (black) is digit 1, negative class (white) all others



$$TN = 435$$

$$FP = 0$$

$$FN = 8$$

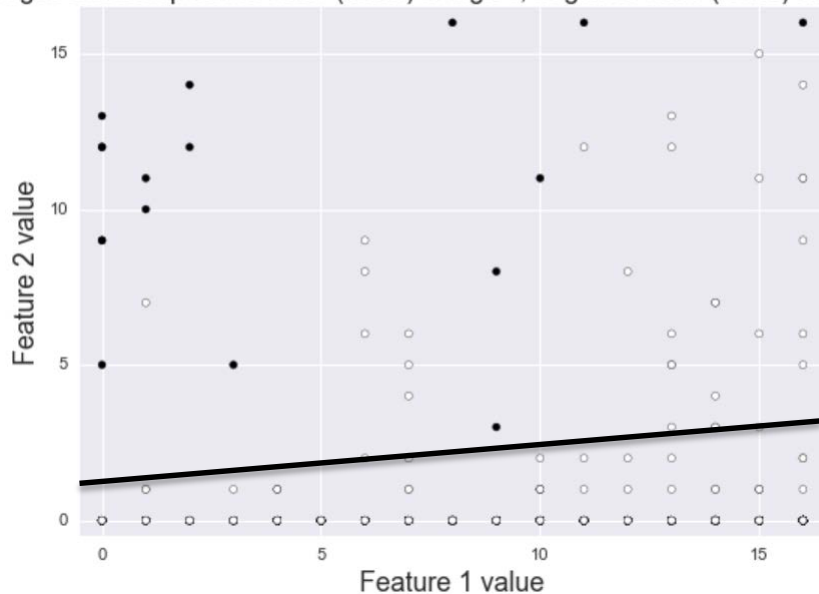
$$TP = 7$$

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{7}{7} = 1.00$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{7}{15} = 0.47$$

Low Precision, High Recall

digits dataset: positive class (black) is digit 1, negative class (white) all others



$$TN = 408$$

$$FP = 27$$

$$FN = 0$$

$$TP = 15$$

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{15}{42} = 0.36$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{15}{15} = 1.00$$

There is often a tradeoff between precision and recall

- Recall-oriented machine learning tasks:
 - Search and information extraction in legal discovery
 - Tumor detection
 - Often paired with a human expert to filter out false positives
- Precision-oriented machine learning tasks:
 - Search engine ranking, query suggestion
 - Document classification
 - Many customer-facing tasks (users remember failures!)

F1-score: combining precision & recall into a single number

$$F_1 = 2 \cdot \frac{\textit{Precision} \cdot \textit{Recall}}{\textit{Precision} + \textit{Recall}} = \frac{2 \cdot TP}{2 \cdot TP + FN + FP}$$

F-score: generalizes F1-score for combining precision & recall into a single number

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \cdot TP}{2 \cdot TP + FN + FP}$$

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}} = \frac{(1 + \beta^2) \cdot TP}{(1 + \beta^2) \cdot TP + \beta \cdot FN + FP}$$

β allows adjustment of the metric to control the emphasis on recall vs precision:

- **Precision-oriented users: $\beta = 0.5$ (false positives hurt performance more than false negatives)**
- **Recall-oriented users: $\beta = 2$ (false negatives hurt performance more than false positives)**

Applied Machine Learning

Classifier Decision Functions

Kevyn Collins-Thompson

Associate Professor of Information & Computer Science
University of Michigan

Decision Functions (decision_function)

- Each classifier score value per test point indicates how confidently the classifier predicts the positive class (large-magnitude positive values) or the negative class (large-magnitude negative values).
- Choosing a fixed decision threshold gives a classification rule.
- By sweeping the decision threshold through the entire range of possible score values, we get a series of classification outcomes that form a curve.

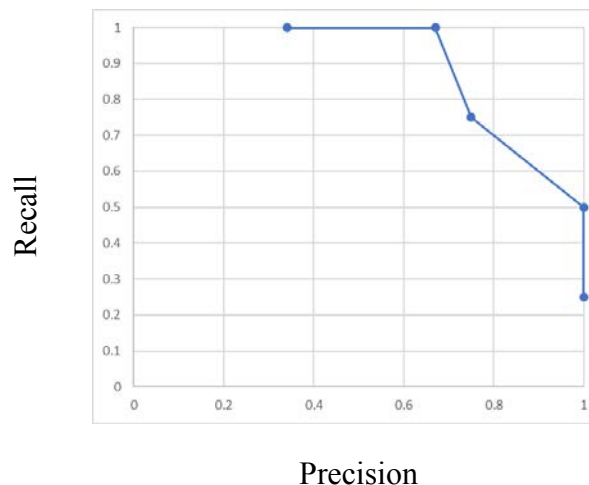
Predicted Probability of Class Membership (predict_proba)

- Typical rule: choose most likely class
 - e.g. class 1 if threshold > 0.50 .
- Adjusting threshold affects predictions of classifier.
- Higher threshold results in a more conservative classifier
 - e.g. only predict Class 1 if estimated probability of class 1 is above 70%
 - This increases precision. Doesn't predict class 1 as often, but when it does, it gets high proportion of class 1 instances correct.
- Not all models provide realistic probability estimates

Varying the Decision Threshold

True Label	Classifier score
0	-27.6457
0	-25.8486
0	-25.1011
0	-24.1511
0	-23.1765
0	-22.575
0	-21.8271
0	-21.7226
0	-19.7361
0	-19.5768
0	-19.3071
0	-18.9077
0	-13.5411
0	-12.8594
1	-3.9128
0	-1.9798
1	1.824
0	4.74931
1	15.234624
1	21.20597

Classifier score threshold	Precision	Recall
-20	$4/12=0.34$	$4/4=1.00$
-10	$4/6=0.67$	$4/4=1.00$
0	$3/4=0.75$	$3/4=0.75$
10	$2/2=1.0$	$2/4=0.50$
20	$1/1=1.0$	$1/4 = 0.25$



Applied Machine Learning

Precision-Recall and ROC curves

Kevyn Collins-Thompson

Associate Professor of Information & Computer Science
University of Michigan

Precision-Recall Curves

X-axis: Precision

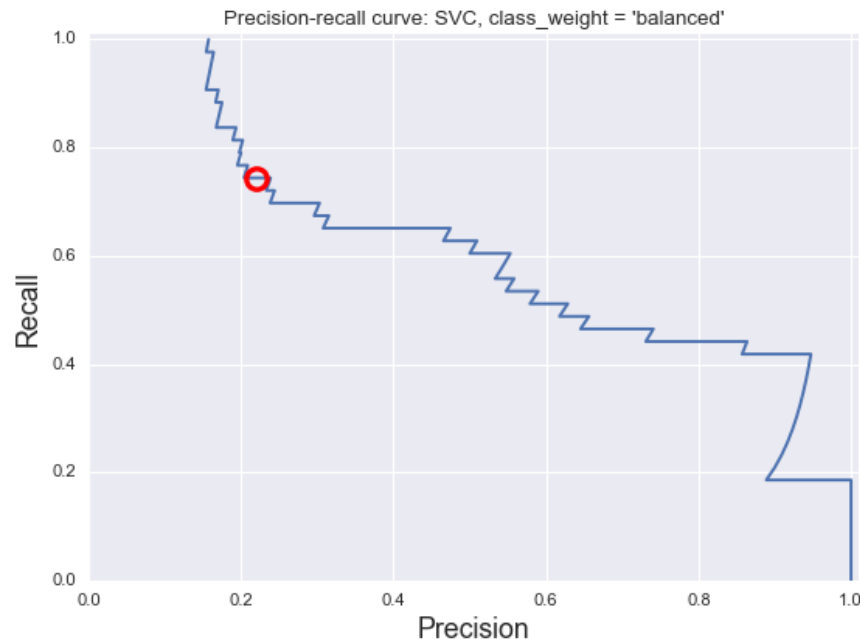
Y-axis: Recall

Top right corner:

- The “ideal” point
- Precision = 1.0
- Recall = 1.0

“Steepness” of P-R curves is important:

- Maximize precision
- while maximizing recall



ROC Curves

X-axis: False Positive Rate

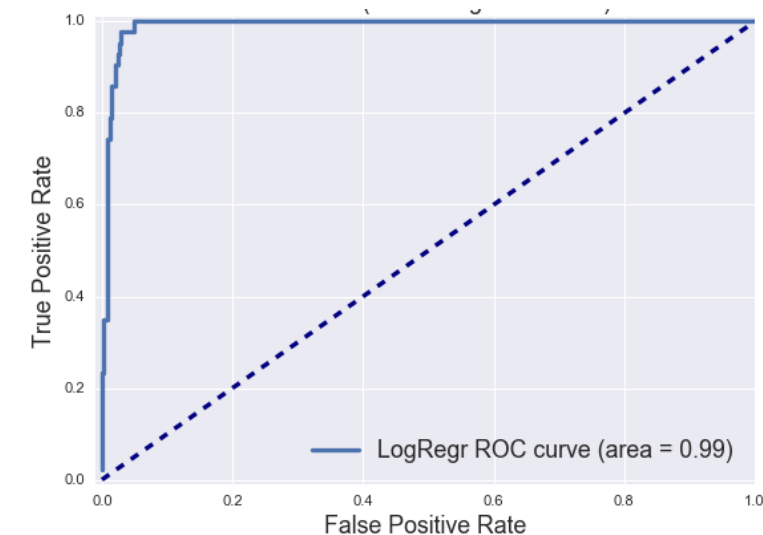
Y-axis: True Positive Rate

Top left corner:

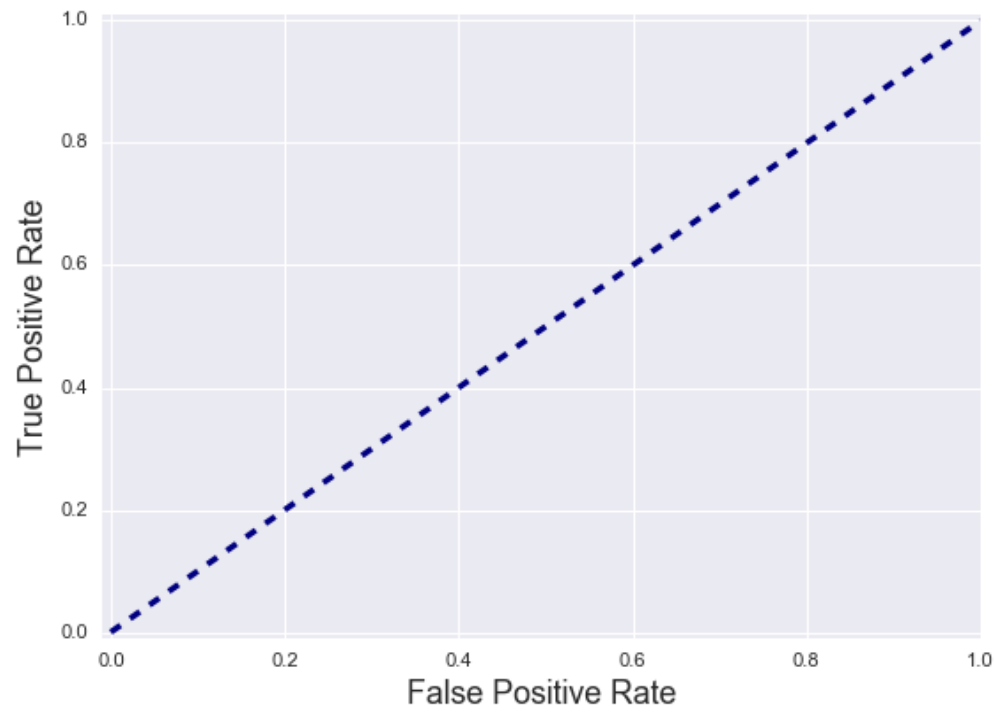
- The “ideal” point
- False positive rate of zero
- True positive rate of one

“Steepness” of ROC curves is important:

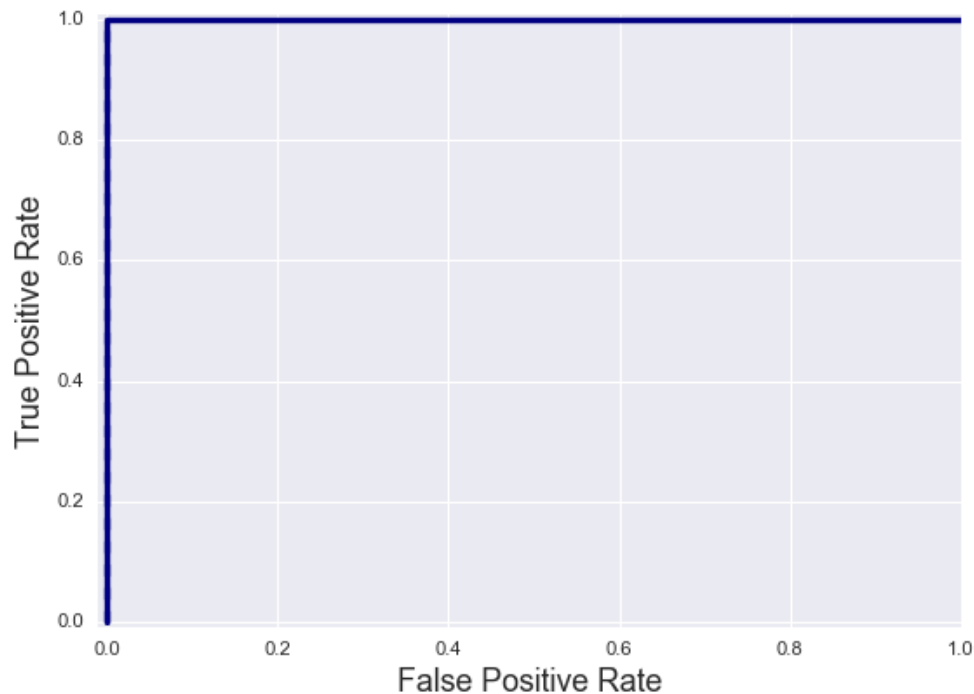
- Maximize the true positive rate
- while minimizing the false positive rate



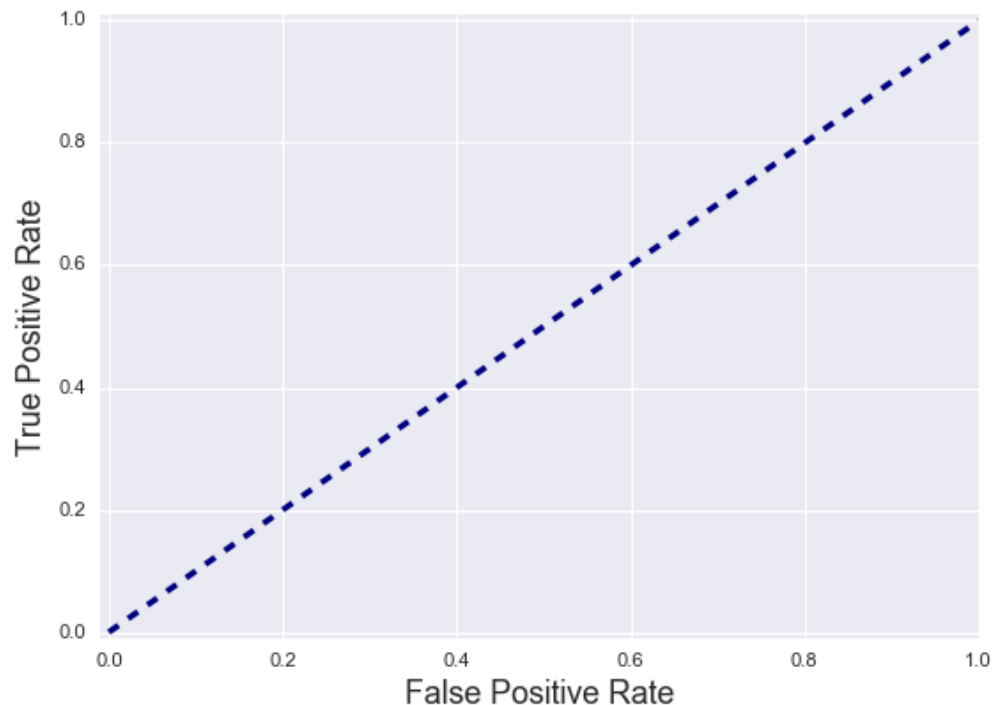
ROC curve examples: random guessing



ROC curve examples: perfect classifier

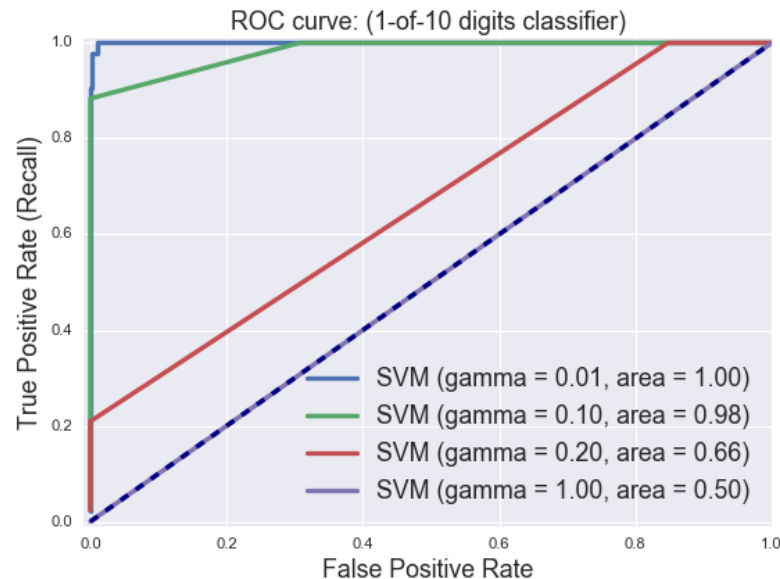


ROC curve examples: bad, okay,



Summarizing an ROC curve in one number: Area Under the Curve (AUC)

- $AUC = 0$ (worst) $AUC = 1$ (best)
- AUC can be interpreted as:
 1. The total area under the ROC curve.
 2. The probability that the classifier will assign a higher score to a randomly chosen positive example than to a randomly chosen negative example.
- **Advantages:**
 - Gives a single number for easy comparison.
 - Does not require specifying a decision threshold.
- **Drawbacks:**
 - As with other single-number metrics, AUC loses information, e.g. about tradeoffs and the shape of the ROC curve.
 - This may be a factor to consider when e.g. wanting to compare the performance of classifiers with overlapping ROC curves.



Applied Machine Learning

Multi-Class Evaluation

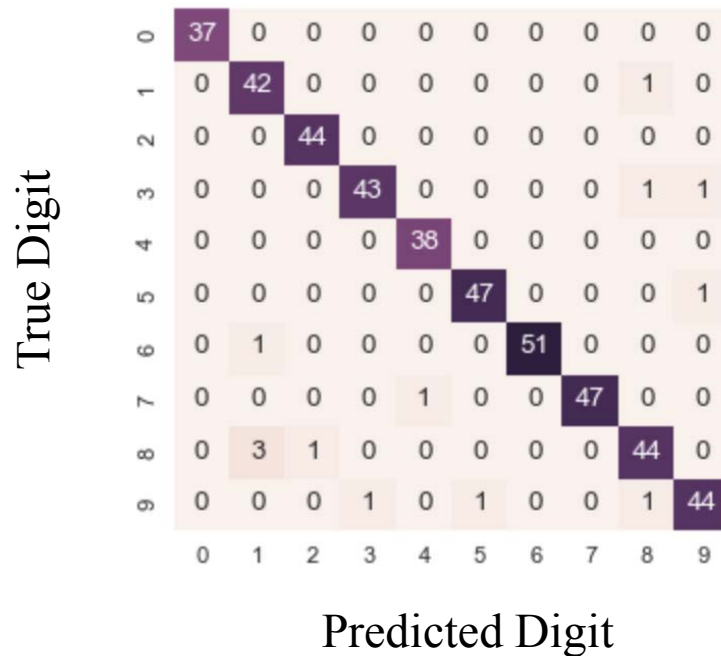
Kevyn Collins-Thompson

Associate Professor of Information & Computer Science
University of Michigan

Multi-Class Evaluation

- Multi-class evaluation is an extension of the binary case.
 - A collection of true vs predicted binary outcomes, one per class
 - Confusion matrices are especially useful
 - Classification report
- Overall evaluation metrics are averages across classes
 - But there are different ways to average multi-class results: we will cover these shortly.
 - The support (number of instances) for each class is important to consider, e.g. in case of imbalanced classes
- Multi-label classification: each instance can have multiple labels (not covered here)

Multi-Class Confusion Matrix



Micro vs Macro Average

Class	Predicted Class	Correct?
orange	lemon	0
orange	lemon	0
orange	apple	0
orange	orange	1
orange	apple	0
lemon	lemon	1
lemon	apple	0
apple	apple	1
apple	apple	1

Macro-average:

- Each class has equal weight.
1. Compute metric within each class
 2. Average resulting metrics across classes

<u>Class</u>	<u>Precision</u>
orange	$1/5 = 0.20$
lemon	$1/2 = 0.50$
apple	$2/2 = 1.00$

Macro-average precision:
 $(0.20 + 0.50 + 1.00) / 3 = \mathbf{0.57}$

Micro vs Macro Average

Class	Predicted Class	Correct?
orange	lemon	0
orange	lemon	0
orange	apple	0
orange	orange	1
orange	apple	0
lemon	lemon	1
lemon	apple	0
apple	apple	1
apple	apple	1

Micro-average:

- Each instance has equal weight.
 - Largest classes have most influence
1. Aggregate outcomes across all classes
 2. Compute metric with aggregate outcomes

Micro-average precision:

$$4 / 9 = \mathbf{0.44}$$

Macro-Average vs Micro-Average

- If the classes have about the same number of instances, macro- and micro-average will be about the same.
- If some classes are much larger (more instances) than others, and you want to:
 - Weight your metric toward the largest ones, use micro-averaging.
 - Weight your metric toward the smallest ones, use macro-averaging.
- If the micro-average is much lower than the macro-average then examine the larger classes for poor metric performance.
- If the macro-average is much lower than the micro-average then examine the smaller classes for poor metric performance.

Multi-class Evaluation Metrics via the "Average" Parameter for a Scoring Function

- Micro: Metric on aggregated instances
- Macro: Mean per-class metric, classes have equal weight
- Weighted: Mean per-class metric, weighted by support
- Samples: for multi-label problems only

Applied Machine Learning

Regression Evaluation

Kevyn Collins-Thompson

Associate Professor of Information & Computer Science
University of Michigan

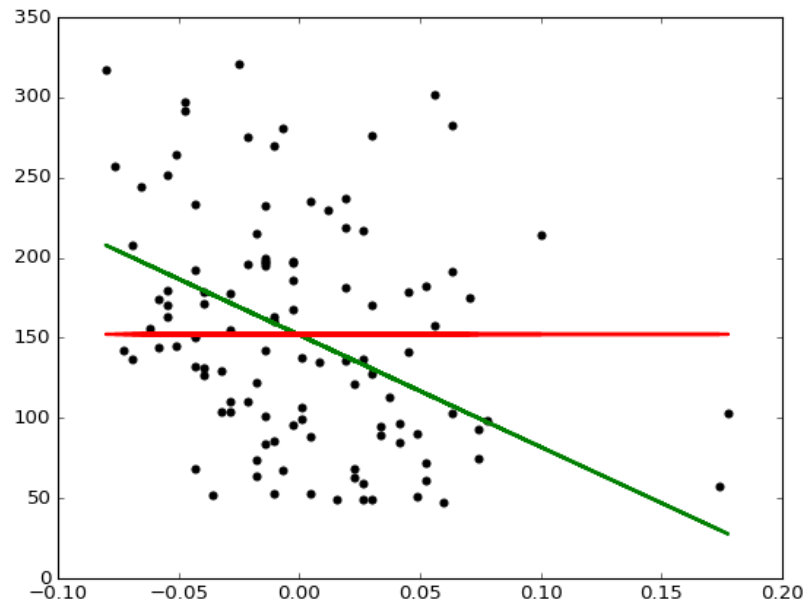
Regression metrics

- Typically `r2_score` is enough
 - Reminder: computes how well future instances will be predicted
 - Best possible score is 1.0
 - Constant prediction score is 0.0
- Alternative metrics include:
 - `mean_absolute_error` (absolute difference of target & predicted values)
 - `mean_squared_error` (squared difference of target & predicted values)
 - `median_absolute_error` (robust to outliers)

Dummy regressors

As in classification, comparison to a 'dummy' prediction model that uses a fixed rule can be useful.

For this, `scikit.learn` provides dummy regressors.

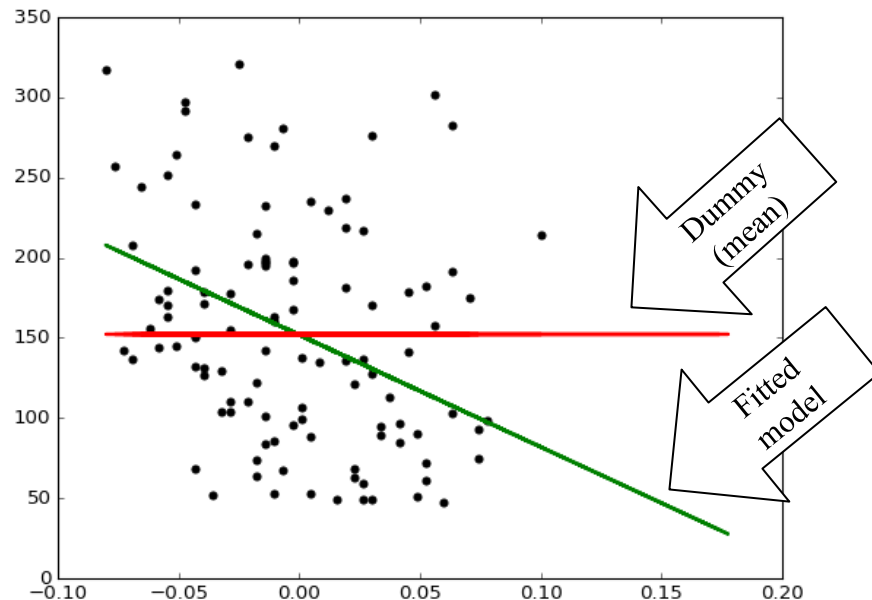


Dummy regressors

As in classification, comparison to a 'dummy' prediction model that uses a fixed rule can be useful.

For this, `scikit.learn` provides dummy regressors

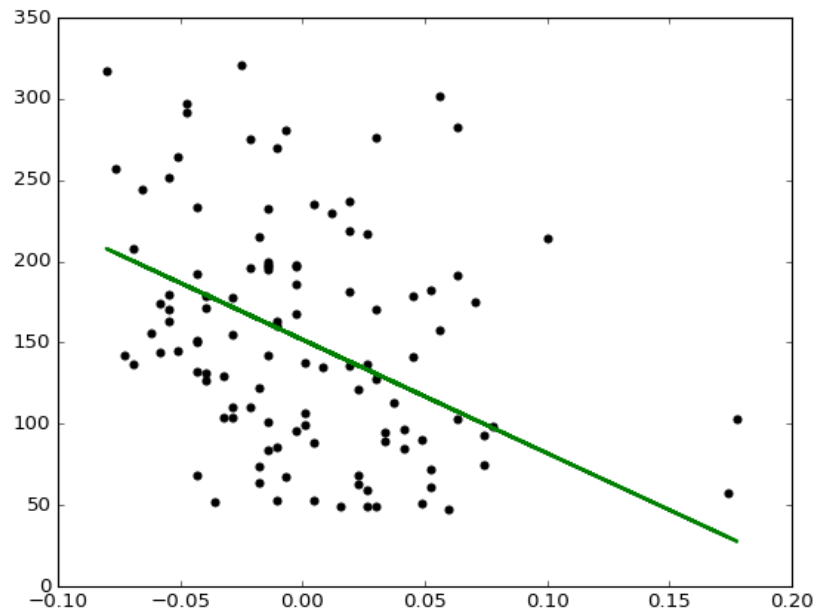
```
Linear model, coefficients: [-698.80206267]  
Mean squared error (dummy): 4965.13  
Mean squared error (linear model): 4646.74  
r2_score (dummy): -0.00  
r2_score (linear model): 0.06
```



Dummy regressors

The `DummyRegressor` class implements four simple baseline rules for regression, using the `strategy` parameter:

- `mean` predicts the mean of the training target values.
- `median` predicts the median of the training target values.
- `quantile` predicts a user-provided quantile of the training target values (e.g. value at the 75th percentile)
- `constant` predicts a custom constant value provided by the user.



Applied Machine Learning

Optimizing Classifiers for Different Evaluation Metrics

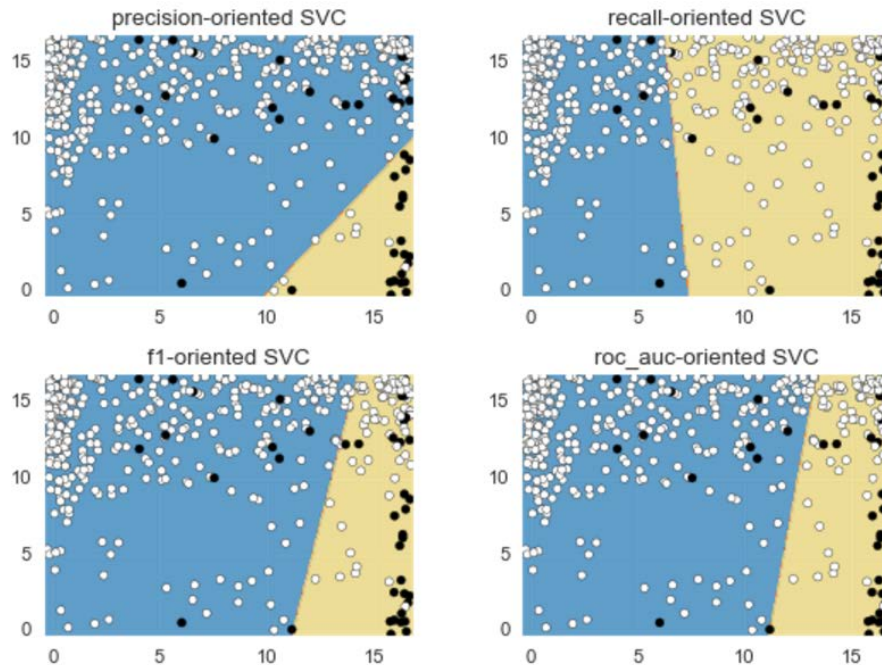
Kevyn Collins-Thompson

Associate Professor of Information & Computer Science
University of Michigan

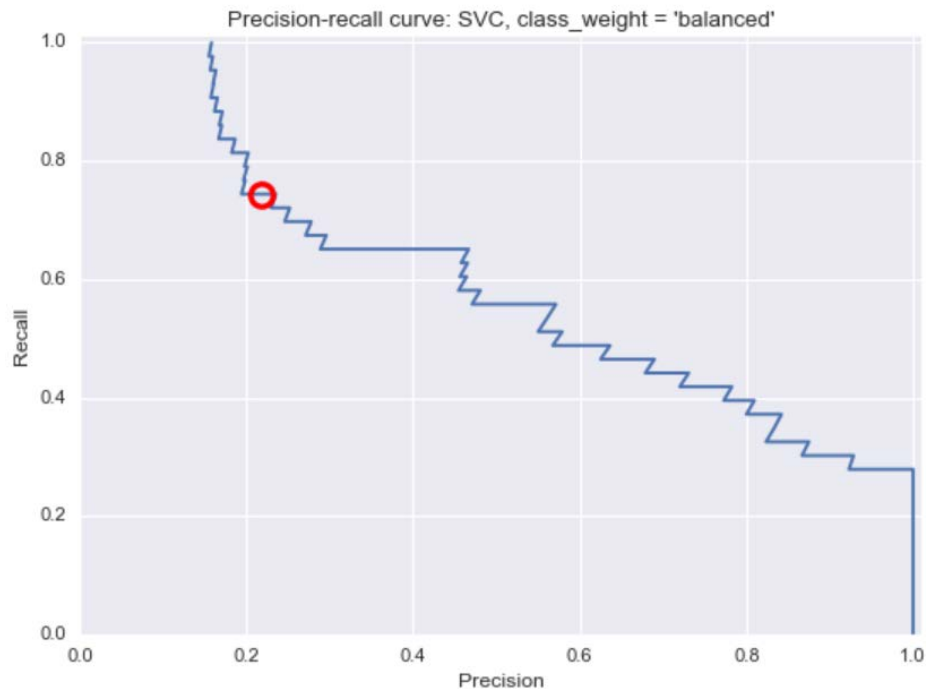
Model Selection Using Evaluation Metrics

- Train/test on same data
 - Single metric.
 - Typically overfits and likely won't generalize well to new data.
 - But can serve as a sanity check: low accuracy on the training set may indicate an implementation problem.
- Single train/test split
 - Single metric.
 - Speed and simplicity.
 - Lack of variance information
- K-fold cross-validation
 - K train-test splits.
 - Average metric over all splits.
 - Can be combined with parameter grid search: GridSearchCV (def. cv = 3)

Example: Optimizing a Classifier Using Different Evaluation Metrics



Example: Precision-Recall Curve of Default Support Vector Classifier



Training, Validation, and Test Framework for Model Selection and Evaluation

- Using only cross-validation or a test set to do model selection may lead to more subtle overfitting / optimistic generalization estimates
- Instead, use three data splits:
 1. Training set (model building)
 2. Validation set (model selection)
 3. Test set (final evaluation)
- In practice:
 - Create an initial training/test split
 - Do cross-validation on the training data for model/parameter selection
 - Save the held-out test set for final model evaluation

Concluding Notes

- Accuracy is often not the right evaluation metric for many real-world machine learning tasks
 - False positives and false negatives may need to be treated very differently
 - Make sure you understand the needs of your application and choose an evaluation metric that matches your application, user, or business goals.
- Examples of additional evaluation methods include:
 - Learning curve: How much does accuracy (or other metric) change as a function of the amount of training data?
 - Sensitivity analysis: How much does accuracy (or other metric) change as a function of key learning parameter values?