# Model Evaluation and Selection

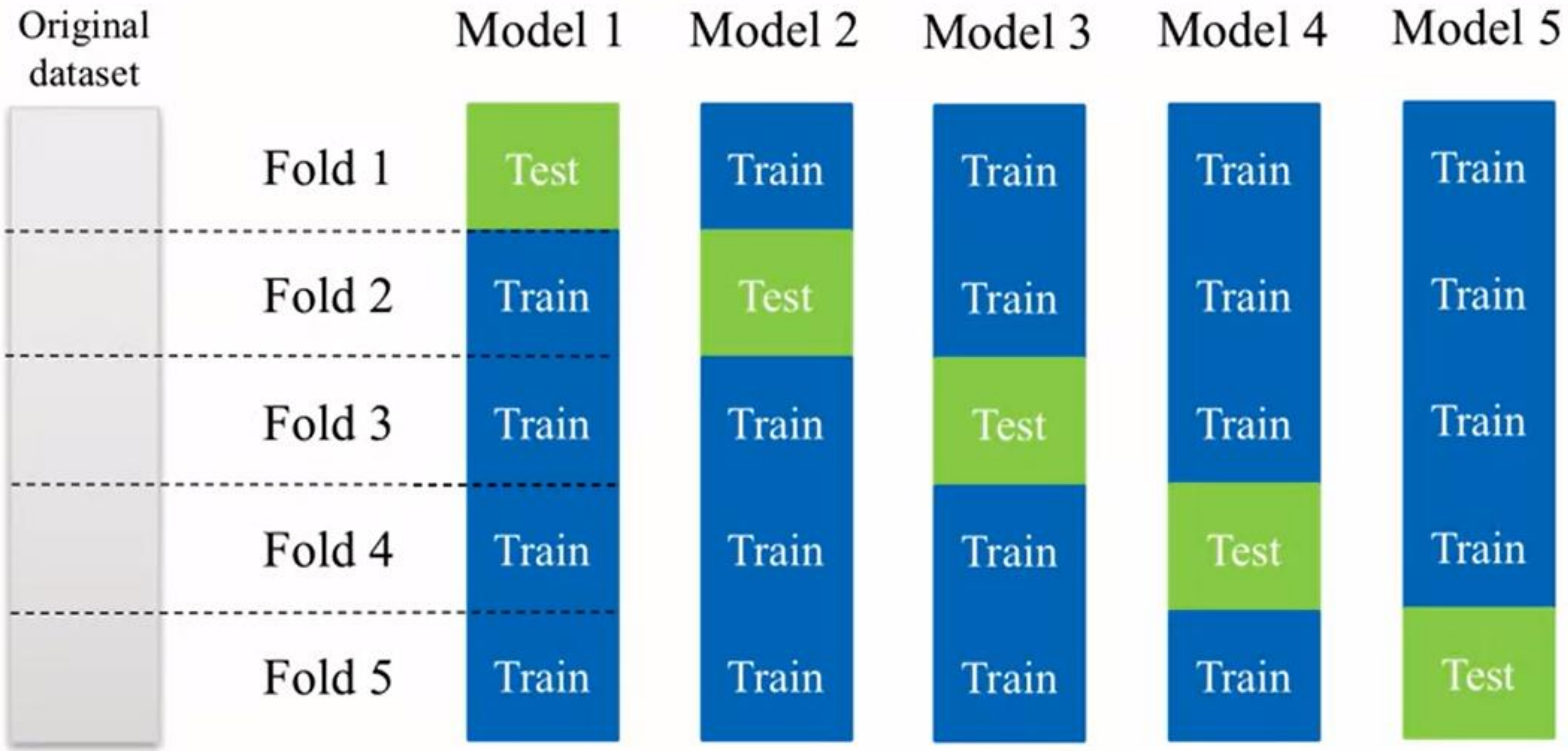
*Part 1*

## Cross Validation

## Cross Validation

- Cross-validation is a method that goes beyond evaluating  a single model using a single Train/Test split of  the data by using **multiple** Train/Test splits,  each of which is used to train and evaluate a separate model.
- You may have noted that by choosing different values for the random state seed parameter in the Train/Test split function,  when you're working on some examples or assignments,  that the accuracy score you get from running a classifier can vary quite a bit  just by chance depending on  the specific samples that happen to end up in the training set.
- Cross-validation basically gives more stable and reliable estimates  of how the classifiers likely to  perform on average by running  multiple different training test splits and then averaging the results,  instead of relying entirely on a single particular training set.
- You need roughly k times more time.

## Cross-validation Example (5-fold)



```
RBF-kernel SVC (with MinMax scaling) test set accuracy: 0.96
```

## Cross-validation

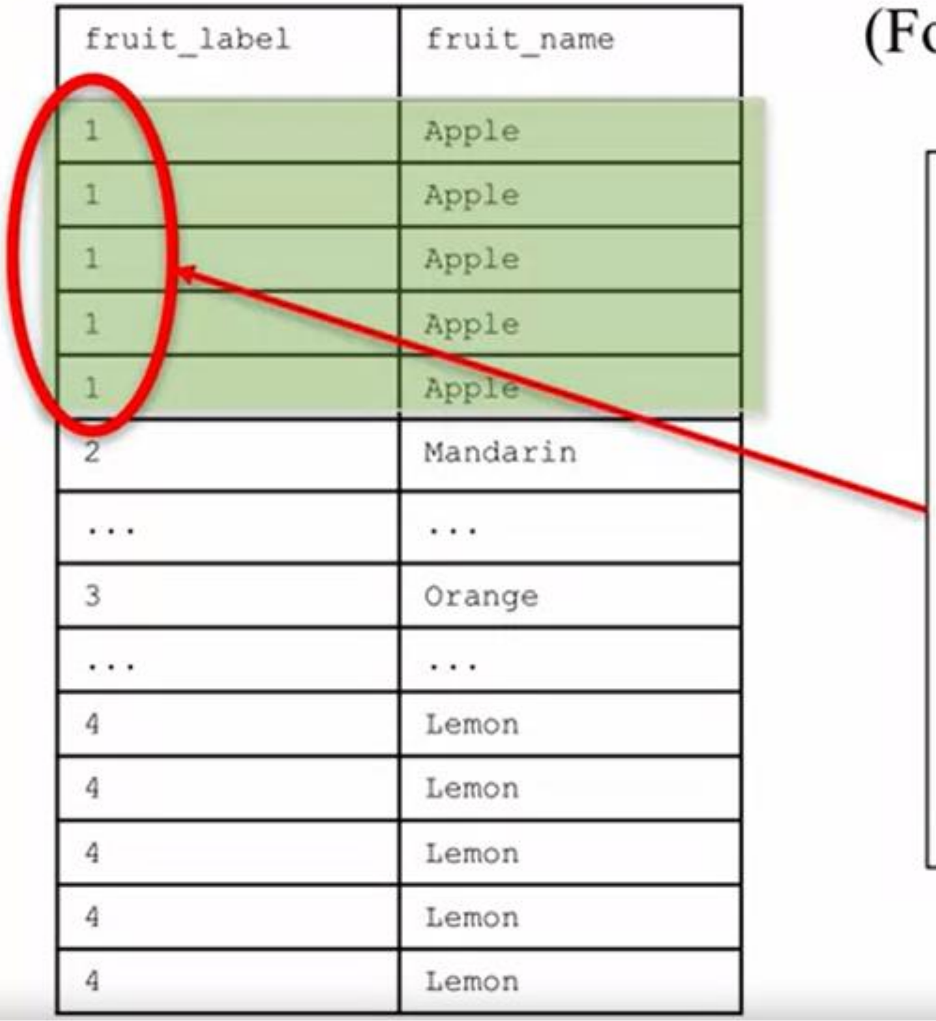
### Example based on k-NN classifier with fruit dataset (2 features)

In [30]:
```python
from sklearn.model_selection import cross_val_score

clf = KNeighborsClassifier(n_neighbors = 5)
X = X_fruits_2d.as_matrix()
y = y_fruits_2d.as_matrix()
cv_scores = cross_val_score(clf, X, y)

print('Cross-validation scores (3-fold):', cv_scores)
print('Mean cross-validation score (3-fold): {:.3f}'
     .format(np.mean(cv_scores)))
```

```
Cross-validation scores (3-fold): [ 0.77  0.74  0.83]
Mean cross-validation score (3-fold): 0.781
```

In [ ]:

## Stratified Cross-validation



| fruit_label | fruit_name |
|---|---|
| 1 | Apple |
| 1 | Apple |
| 1 | Apple |
| 1 | Apple |
| 1 | Apple |
| 2 | Mandarin |
| ... | ... |
| 3 | Orange |
| ... | ... |
| 4 | Lemon |
| 4 | Lemon |
| 4 | Lemon |
| 4 | Lemon |
| 4 | Lemon |

(Folds and dataset shortened for illustration purposes.)

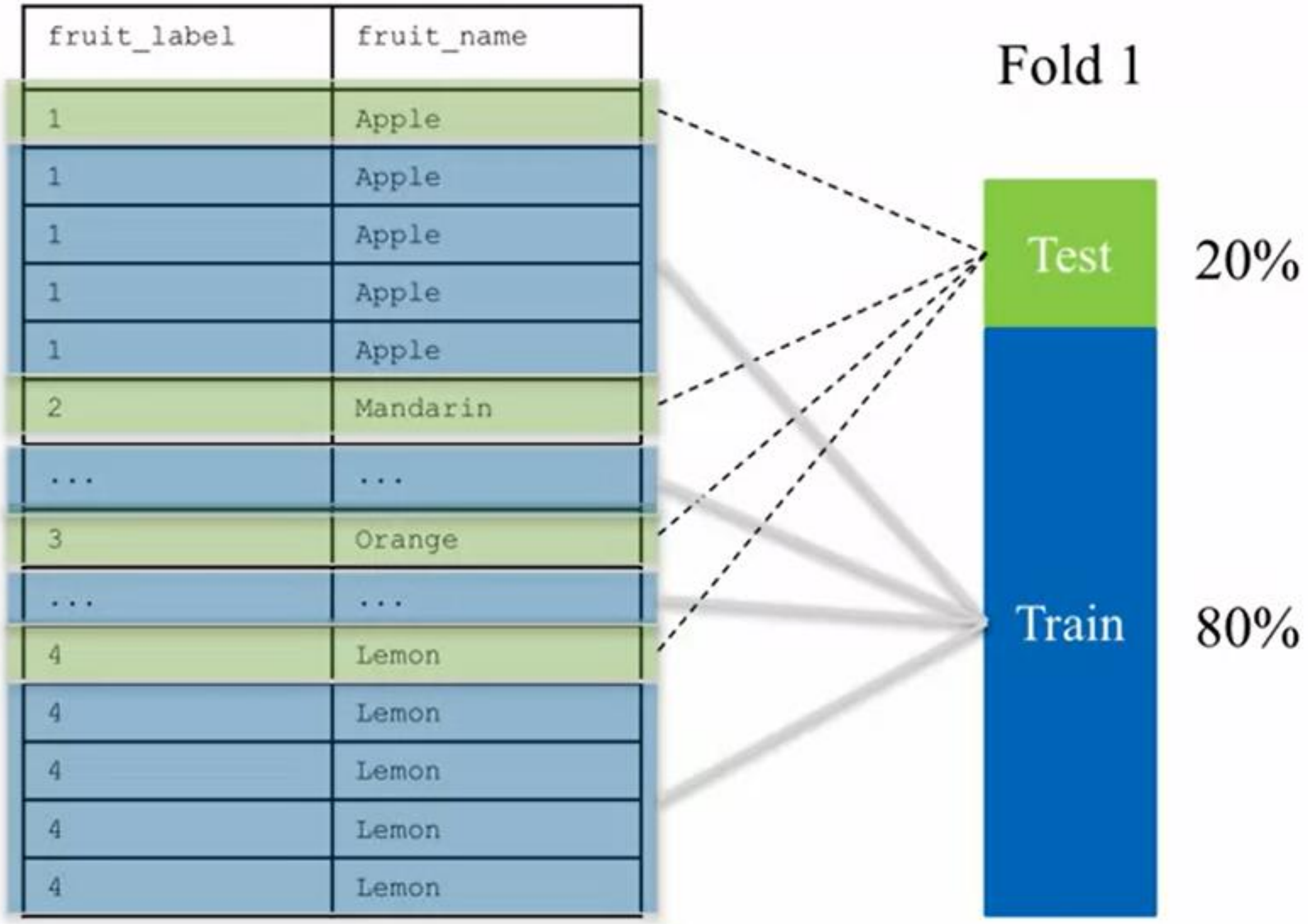
Example has 20 data samples
= 4 classes with 5 samples each.

5-fold CV: 5 folds of 4 samples each.

Fold 1 uses the first 20% of the dataset as the test set, which only contains samples from class 1.

Classes 2, 3, 4 are missing entirely from test set and so will be missing from the evaluation.

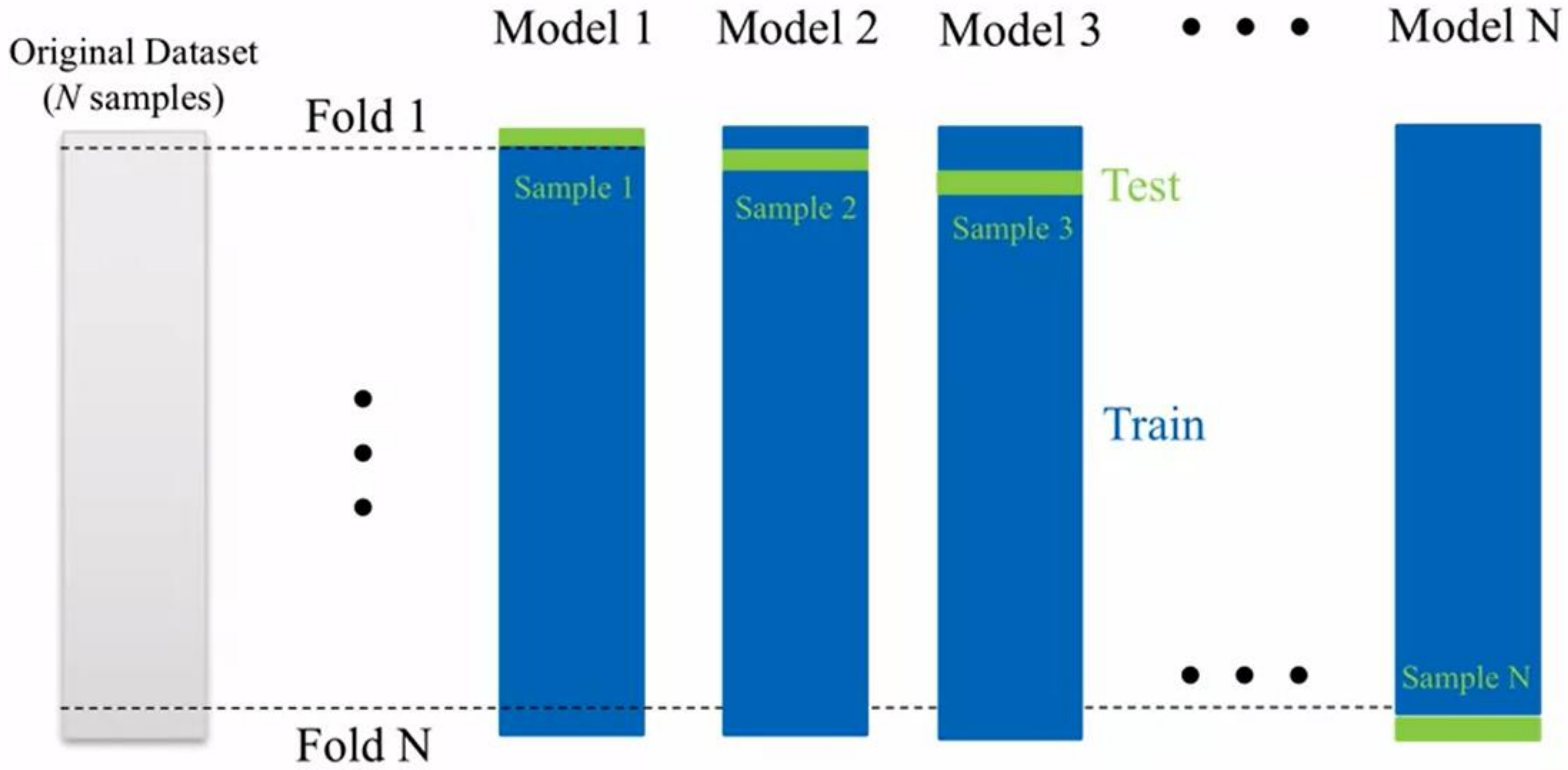
# Stratified Cross-validation



| fruit_label | fruit_name |
|---|---|
| 1 | Apple |
| 1 | Apple |
| 1 | Apple |
| 1 | Apple |
| 1 | Apple |
| 2 | Mandarin |
| ... | ... |
| 3 | Orange |
| ... | ... |
| 4 | Lemon |
| 4 | Lemon |
| 4 | Lemon |
| 4 | Lemon |
| 4 | Lemon |

Stratified folds each contain a proportion of classes that matches the overall dataset. Now, all classes will be fairly represented in the test set.

## Leave-one-out cross-validation (with *N* samples in dataset)



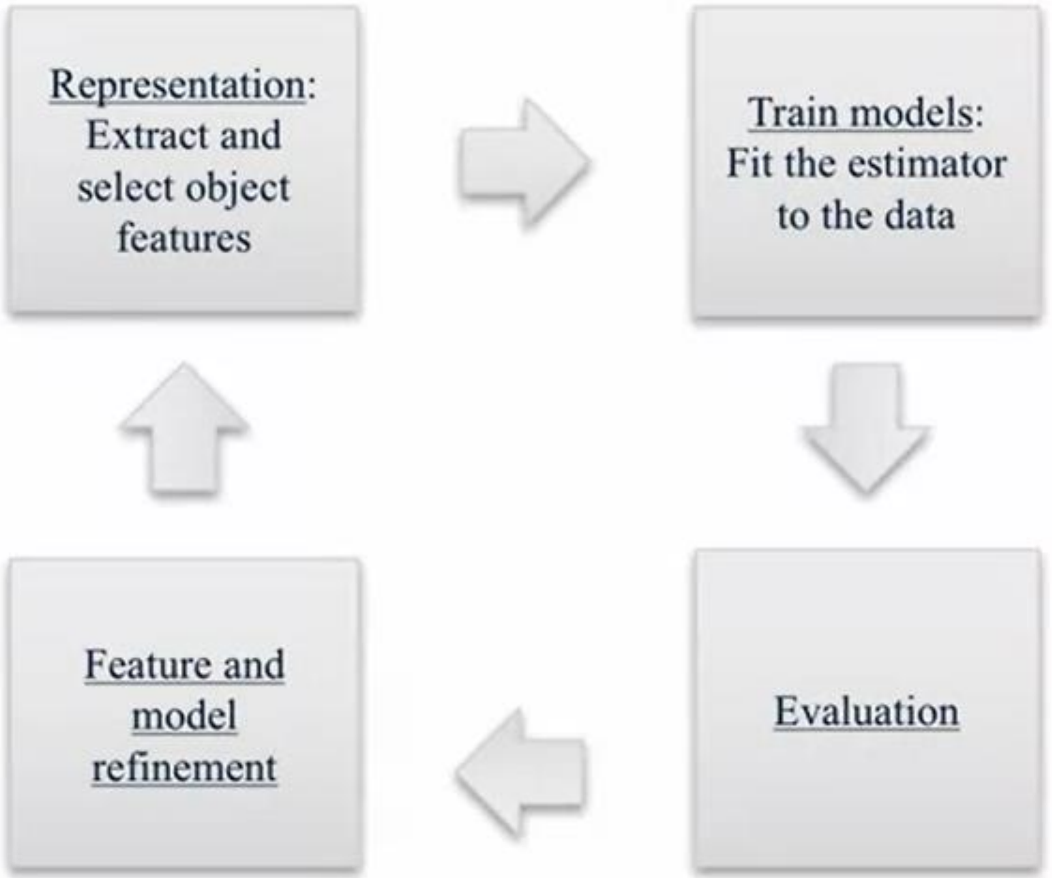
## Evaluation Metrics for Classification

## 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


Representation: Extract and select object features
Train models: Fit the estimator to the data
Evaluation
Feature and model refinement

## 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.

## Some examples

- Credit card transactions
- Web search
- Cancer prediction

## Confusion Matricies

## Binary Prediction Outcomes

| | **Predicted** negative | **Predicted** positive |
|---|---|---|
| **True** negative | TN | FP |
| **True** positive | FN | TP |

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

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

N = 450

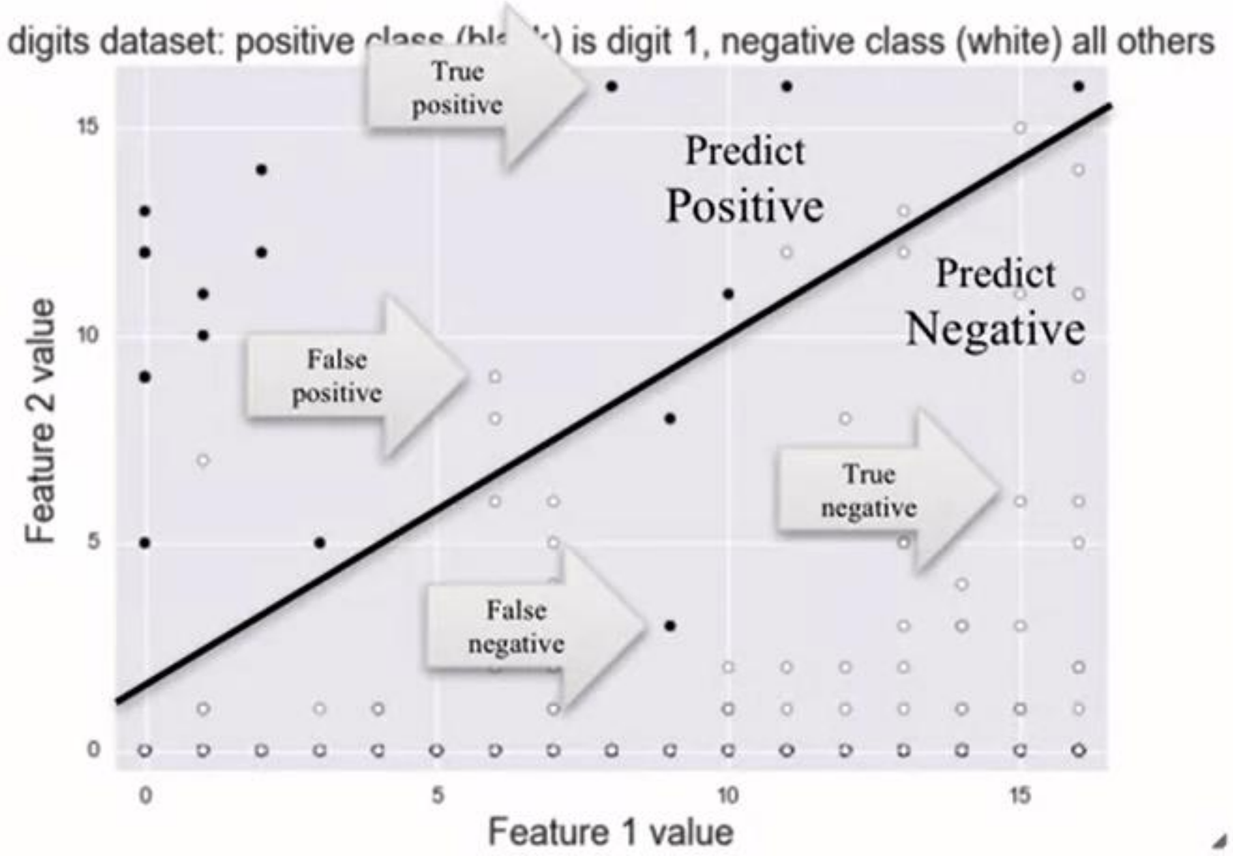
- Every test instance is in exactly one box (integer counts).
- Breaks down classifier results by error type.
- Thus, provides more information than simple accuracy.
- Helps you choose an evaluation metric that matches project goals.
- Not a single number like accuracy.. but there are many possible metrics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

## 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



| | |
|---|---|
| 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 | $N = 450$ |

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

$$= \frac{400+26}{400+26+17+7}$$

$$= 0.95$$

## Classification error (1 – Accuracy): for what fraction of all instances is the classifier's prediction <u>incorrect</u>?

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

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

$$= \frac{7+17}{400+26+17+7}$$

$$= 0.060$$

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

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

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

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

$$= 0.60$$

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 | $N = 450$ |

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

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

$$= 0.79$$

Query suggestions…

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

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

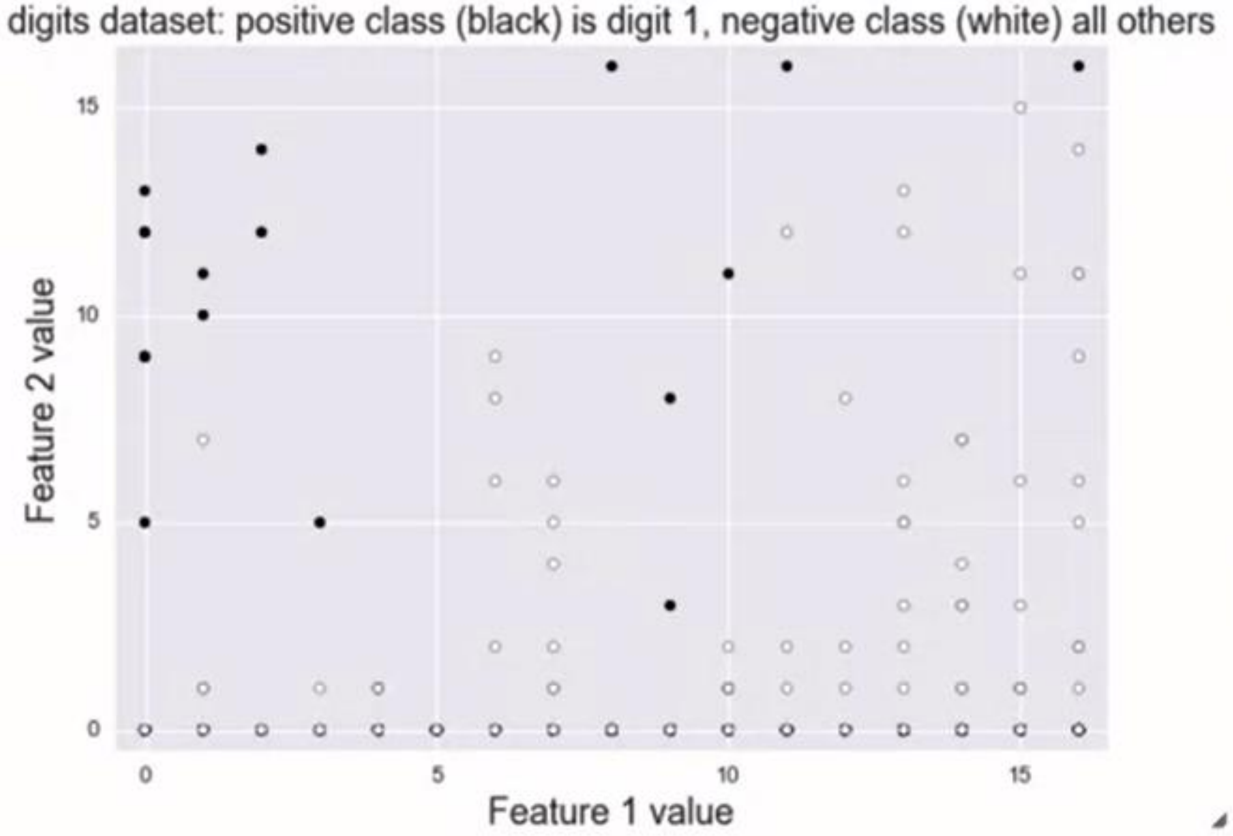
$$FPR = \frac{FP}{TN+FP}$$

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

$$= 0.02$$

False Positive Rate is also known as:
- Specificity

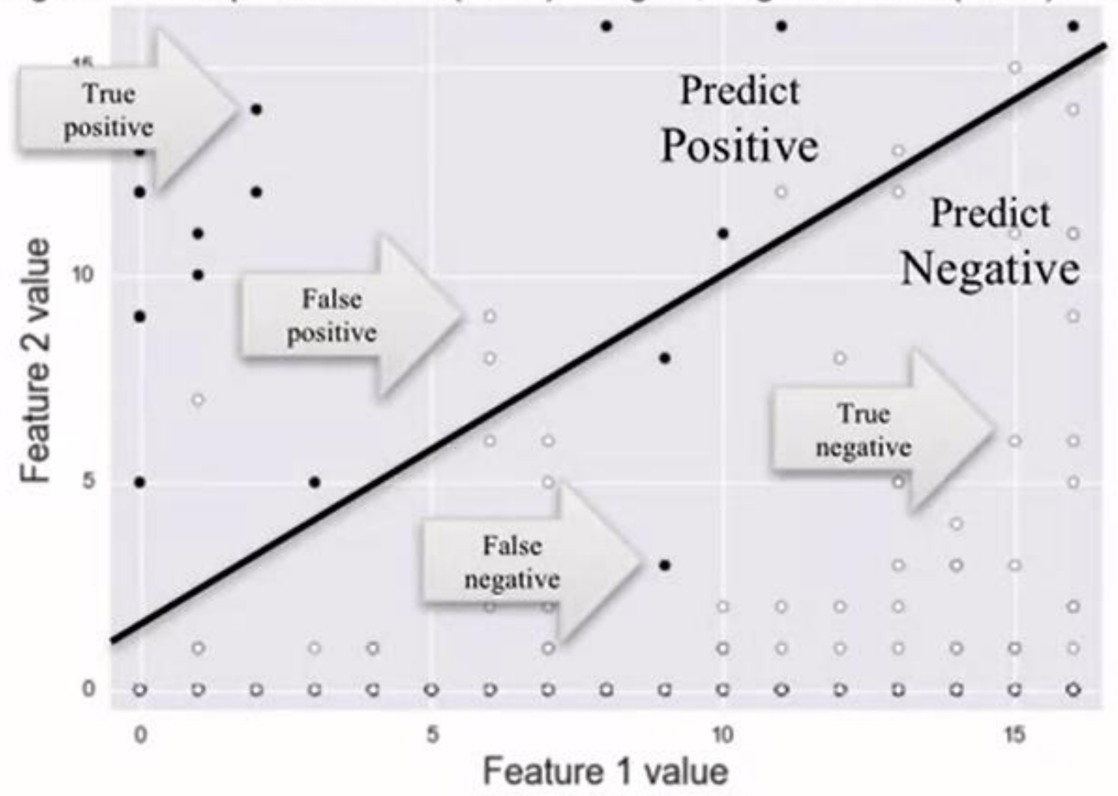
## A Graphical Illustration of Precision & Recall



| | |
|---|---|
| TN = | FP = |
| FN = | TP = |

# The Precision-Recall Tradeoff

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

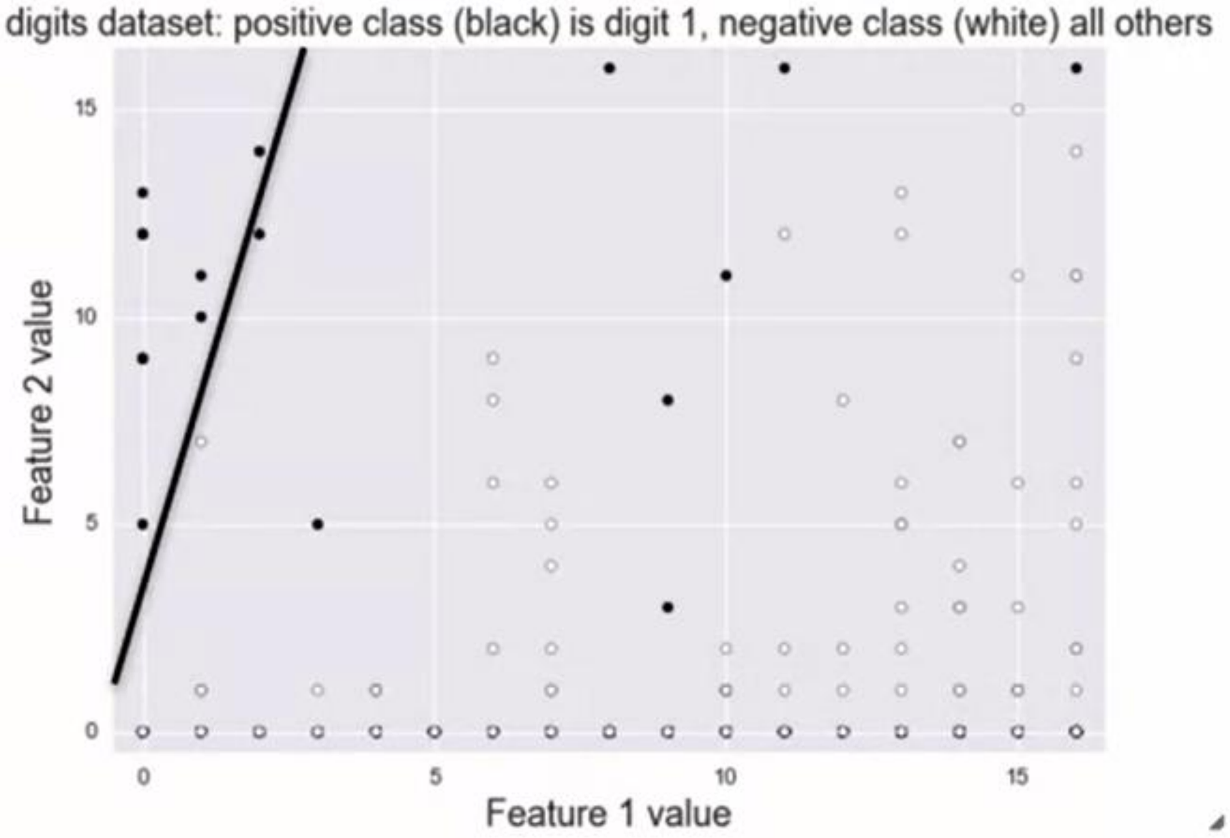


| | |
|---|---|
| TN = 429 | FP = 6 |
| FN = 2 | TP = 13 |

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

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

## High Precision, Lower Recall



|  |  |
|---|---|
| TN = 435 | FP = 0 |
| FN = 8 | TP = 7 |

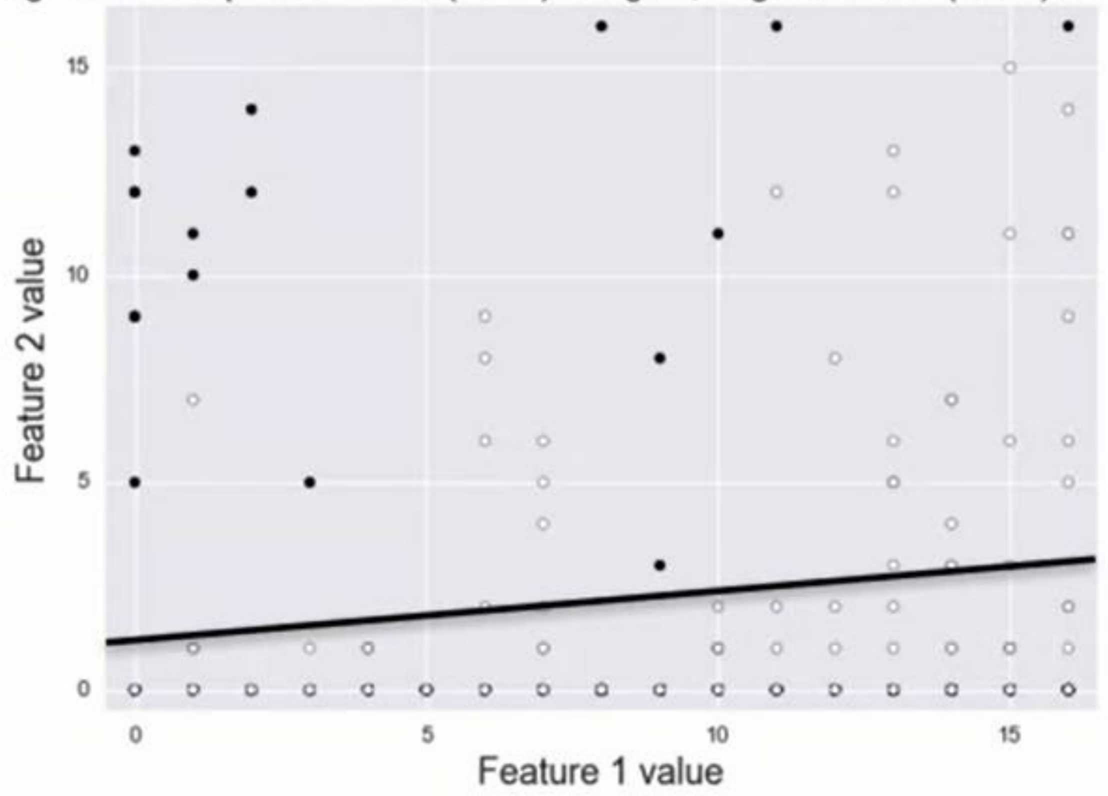
$$\text{Precision} = \frac{TP}{TP+FP} = \frac{7}{7} = 1.00$$

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

Query results.

## Low Precision, High Recall

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



Tumor Prediction

| | |
|---|---|
| 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

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

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

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

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

$\beta$ 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)**