Classification

Supervised Learning

- A very common method for classification
- The supervised model trains / learns over m labeled instances $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(m)}, y^{(m)}\right)$
  - x is the values of attributes representing the instance and y is the label assigned to x
- After training, the model can then predict the class of unlabeled instances
- Many kinds of supervised classifiers
  - K-nearest neighbours
  - Neural network
  - Naive Bayes
  - Logistic Regression
  - Decision Tree

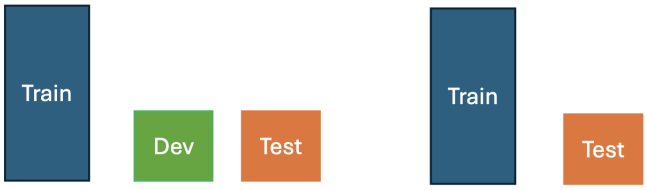
Classification Design

- To train and evaluate a supervised model, we need to partition our dataset into different sets
- Training Set
  - Contains labeled instances
  - Model learns to predict the labels of instances
- Dev / Validation Set
  - Contains labeled instances
  - Used to tune hyperparameters of models
  - Expected to look similar to test set
  - Sometimes a dev set is not used
- Test Set
  - Contains unlabeled instances
    - Although the model does not see / use the labels, during experiments the instances do have labels that we use to measure the performance of the classifier
  - Used to evaluate the performance of a model

Designing the Splits

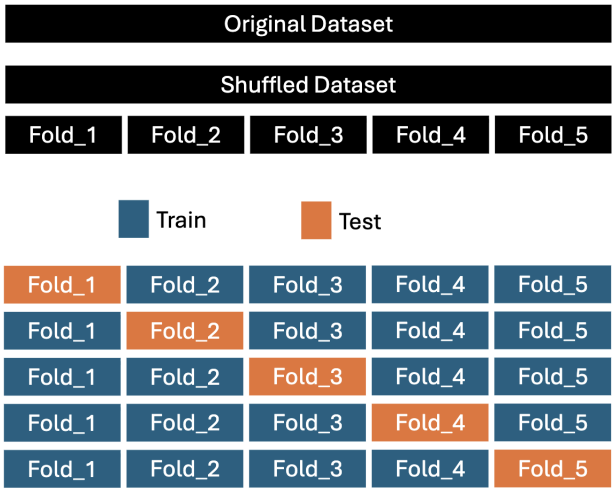
1. Held-out

   - You divide the dataset into Training, Dev, and Test splits based on some predefined percentage
     - Train / Dev / Test = 80 / 10 / 10 or 70 / 20 / 10, etc.
     - If not using a Dev set, then Train / Test = 80 / 20, etc.



3. k-fold cross-validation

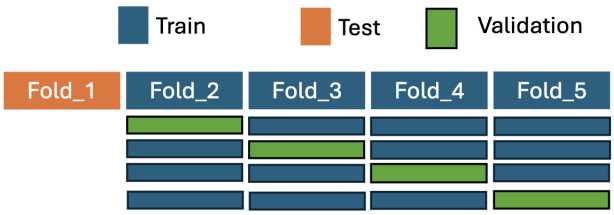
   - This is good for when there are a few instances
   - The order of the instances in the data are shuffled
   - You partition the dataset into k roughly equal sized partitions
   - The model takes turn testing on one partition and training on the other $k-1$ partitions
   - The final score is the average performance on each test fold



4. Nested k-fold

   - Used when there are hyperparameters to tune
   - Start with shuffling the instances and creating k folds, like before
   - Train on $k-2$ folds and validate on validation
   - Used to tune hyperparameters
   - After tuning hyperparameters, train on fold 2 to fold 5 and test on fold 1
   - Repeat this process for each fold, similar to k-fold cross-validation

- The final score is the average performance on each test fold



Types of Learning

- Supervised
  - Learns with access to labeled instances
- Unsupervised
  - Learns without access to labeled instances
- Semi-supervised
  - Learns with some instances being labeled, and some instances that are not labeled

Dataset Example

Sepal Length (cm)	Sepal Width (cm)	Petal Length (cm)	Petal Width (cm)	Class
5.3	3.7	1.5	0.2	Iris-setosa
5.0	3.3	1.4	0.2	Iris-setosa
7.0	3,2	4.7	1.4	Iris-vesicolor
6.4	3.2	5.3	2.3	Iris-virginica

- The class may also be called the ground truth, or the gold standard

Classification Model

- Our model should use the values of the attributes (features) to then predict a class
- Iris dataset used for example

$$\boxed{5.3}\boxed{3.7}\boxed{1.5}\boxed{0.2} \rightarrow \text{Model} \rightarrow \hat{y}$$

Baselines

- Baseline – A simple model that is used for comparison
- Common Baselines
  - All-in-one / Most-frequent Class (MFC)
    - Always predict the most frequent class
  - Random

    - Randomly predicts each class qith equal probability
  - Simple Heuristic
    - Use a simple rule to predict a class

Baseline Example

- Most-Frequent Class (MFC)
- 50 Iris-setosa
- 50 Iris-versicolor
- 50 Iris-virginica
- Since all of the three classes have the same frequency, we can select either one to be our class that we always predict
- If one class had more frequencies, we would select that class

Sepal Length (cm)	Sepal Width (cm)	Petal Length (cm)	Petal Width (cm)	Class	Predictions (MFC)
5.3	3.7	1.5	0.2	Iris-setosa	Iris-setosa
5.0	3.3	1.4	0.2	Iris-setosa	Iris-setosa
7.0	3,2	4.7	1.4	Iris-vesicolor	Iris-setosa
6.4	3.2	5.3	2.3	Iris-virginica	Iris-setosa

Evaluation

- We need some way to measure the performance of our classifier model
- We will first consider a binary classification
  - 2 possible classes
  - We can call one class the positive class and one class the negative class

Confusion Matrix for a Binary Classification

- True Positive (tp) – The number of predicted positive that have the true label as positive
- True Negative (tn) – The number of predicted negative that have the true label as negative
- False Positive (fp) – The number of predicted positive that have the true label as negative
- False Negative (fn) – The number of predicted negative that have the true label as positive

		Actual	
		Class 1	Class 0
Predicted	Class 1	tp	fp
	Class 0	fn	tn

Building the confusion matrix

- Confusion matrices are not often given to us
- We need to build them ourselves from the output of our classifiers

Predicted	Gold / Actual
Class 1	Class 1
Class 0	Class 1
Class 1	Class 0
Class 0	Class 0
Class 0	Class 0
Class 1	Class 0
Class 1	Class 0

		Actual	
		Class 1	Class 0
Predicted	Class 1	1	3
	Class 0	1	2

Apply the Most Frequent Class Baseline: Accuracy = $\frac{5}{7}$

Gold / Actual	Predicted
Class 1	Class 0
Class 1	Class 0
Class 0	Class 0
Class 0	Class 0
Class 0	Class 0
Class 0	Class 0
Class 0	Class 0

- Accuracy – Percentage of all instances that were correctly classified
- Precision – Percentage of the instances predicted as positive that were indeed positive
- Recall – Percentage of actual positives that were predicted as positive

$$\text{Accuracy} = \frac{tp + tn}{tp + fp + tn + fn}$$

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$