CS-EJ3211 Machine Learning with Python

Session 4 - Classification

Shamsi Abdurakhmanova

Aalto University
FITech

24.02.22

Classification

- Find $h(x)$ that predicts label y based on feature vector $\mathbf{x} = (x_1, ..., x_n)$
- Labels y encode different classes or categories
- Examples:
  - images {"cat", "dog"}
  - tumor {"malignant", "benign"}
  - clothes {"shirt", "trousers", "hat"}
- Two classes classification - **binary** classification
- More then two classes classification - **multiclass** classification

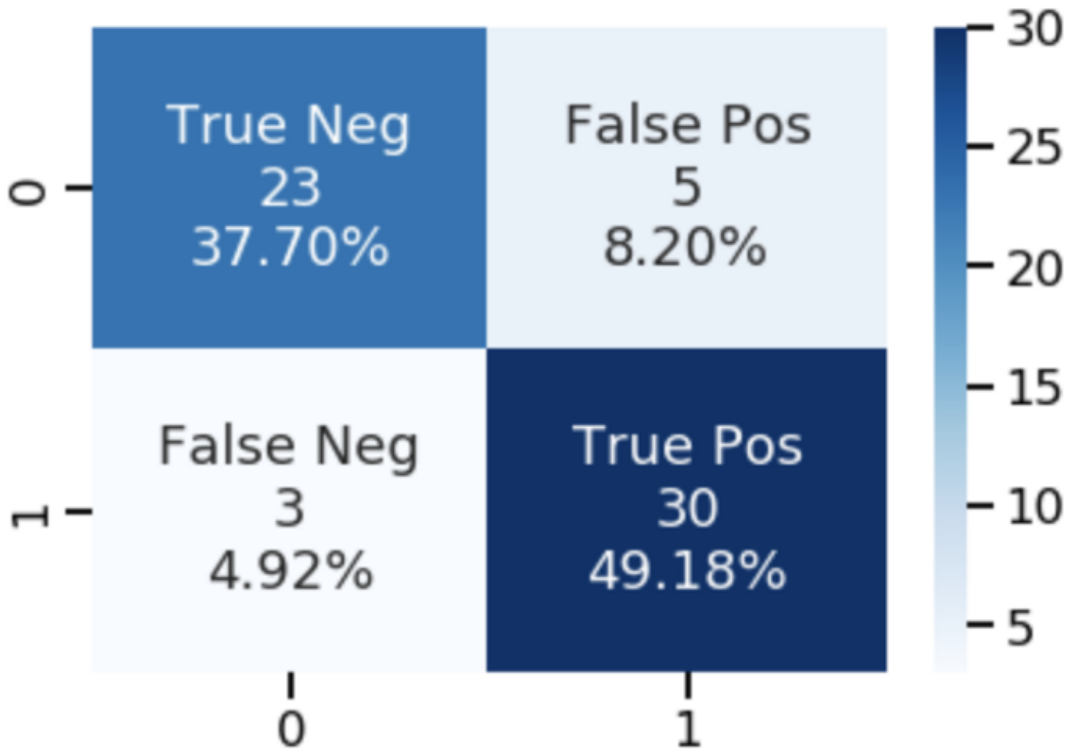
Classification performance

- $y = 1$ positive class/ category
- $y = 0$ negative class/ category

Possible outcomes of binary classification:

- $y = 0, \hat{y} = 0$ True Negative (TN)
- $y = 0, \hat{y} = 1$ False Positive (FP)
- $y = 1, \hat{y} = 1$ True Positive (TP)
- $y = 1, \hat{y} = 0$ False Negative (FN)

Confusion matrix



Classification performance

		Predicted label	
		Negative	Positive
True label	Negative	TN ✓	FP ✗
	Positive	FN ✗	TP ✓

Accuracy - fraction of correctly predicted labels

$$\frac{TP+TN}{TP+FP+TN+FN}$$

Classification performance

		Predicted label: Negative	Predicted label: Positive
True label	Negative	TN ✓	FP ✗
	Positive	FN ✗	TP ✓

Precision - fraction of correctly predicted positive class among all predicted positive. Use when the cost of false positive is high (positive result from "cancer test", which then requires further invasive procedures)

$$\frac{TP}{TP+FP}$$

Classification performance

		Predicted label: Negative	Predicted label: Positive
True label	Negative	TN ✓	FP ✗
	Positive	FN ✗	TP ✓

Recall (sensitivity) - fraction of correctly predicted positive class among all with true label positive. Use when the cost of false negative is high (prediction of an equipment failure, which is easy to check, but very costly to repair)

$$\frac{TP}{TP+FN}$$

Classification performance

		Predicted label: Negative	Predicted label: Positive
True label	Negative	TN ✓	FP ✗
	Positive	FN ✗	TP ✓

F1 score - combination of precision and recall. High F1 score implies low FP and low FN.

$$2\frac{precision*recall}{precision+recall}$$

Classification performance

Consider various scenarios: Blog post "Data Science in Medicine — Precision & Recall or Specificity & Sensitivity?"

Logistic Regression

- **Assumption:** log-odds are linearly dependent on features and weights

$$\log \frac{P(y=1)}{P(y=0)} = w_0 + w_1 x_1 + ... + w_n x_n$$

- **Model parameters:**

$$w_0, w_1, ..., w_n$$

- Log-odds are used to predict labels of data point with features $\mathbf{x}$:

$$\hat{y} = 1 \text{ if } w_0 + \mathbf{w}^\mathbf{T}\mathbf{x} \geq 0 \;\; i.e., P(y=1) \geq P(y=0)$$
$$\hat{y} = 0 \text{ if } w_0 + \mathbf{w}^\mathbf{T}\mathbf{x} < 0 \;\; i.e., P(y=1) < P(y=0)$$

Logistic Regression

$$\log \frac{P(y=1)}{P(y=0)} = w_0 + \mathbf{w}^\mathbf{T}\mathbf{x}$$

Example 1:

- $P(y=1) = 0.5, P(y=0) = 0.5$
- $\log \frac{P(y=1)}{P(y=0)} = 0$
- $w_0 + \mathbf{w}^\mathbf{T}\mathbf{x} = 0$
- $\hat{y} = 1$

Logistic Regression

$$\log \frac{P(y=1)}{P(y=0)} = w_0 + \mathbf{w}^\mathbf{T}\mathbf{x}$$

Example 2:

- $P(y=1) = 0.9 > P(y=0) = 0.1$
- $\log \frac{P(y=1)}{P(y=0)} \approx 2.2$
- $w_0 + \mathbf{w}^\mathbf{T}\mathbf{x} > 0$
- $\hat{y} = 1$

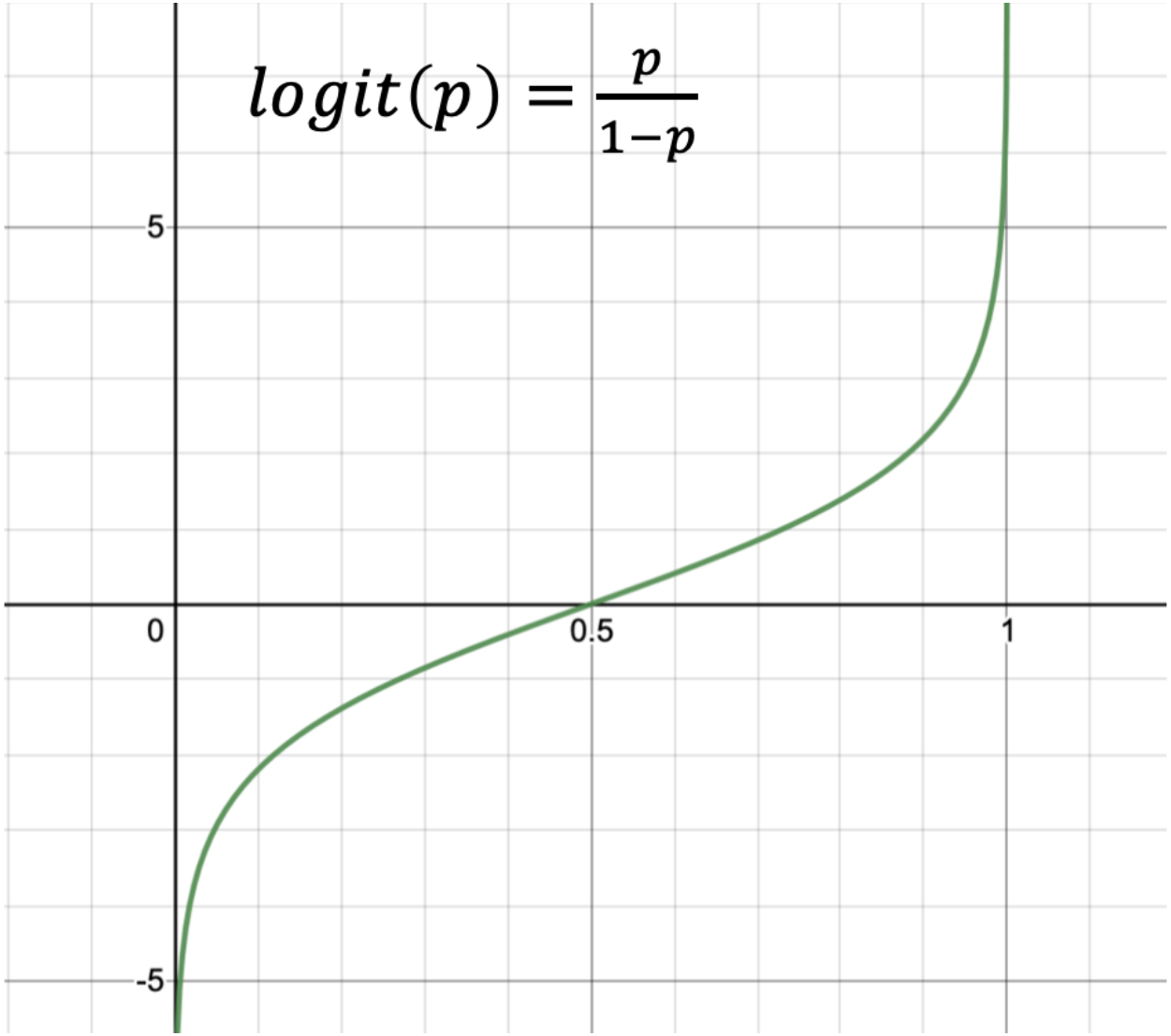
Logistic Regression

$$\log \frac{P(y=1)}{P(y=0)} = w_0 + \mathbf{w}^\mathbf{T}\mathbf{x}$$

Example 3:

- $P(y=1) = 0.1 < P(y=0) = 0.9$
- $\log \frac{P(y=1)}{P(y=0)} \approx -2.2$
- $w_0 + \mathbf{w}^\mathbf{T}\mathbf{x} < 0$
- $\hat{y} = 0$

Logistic Regression

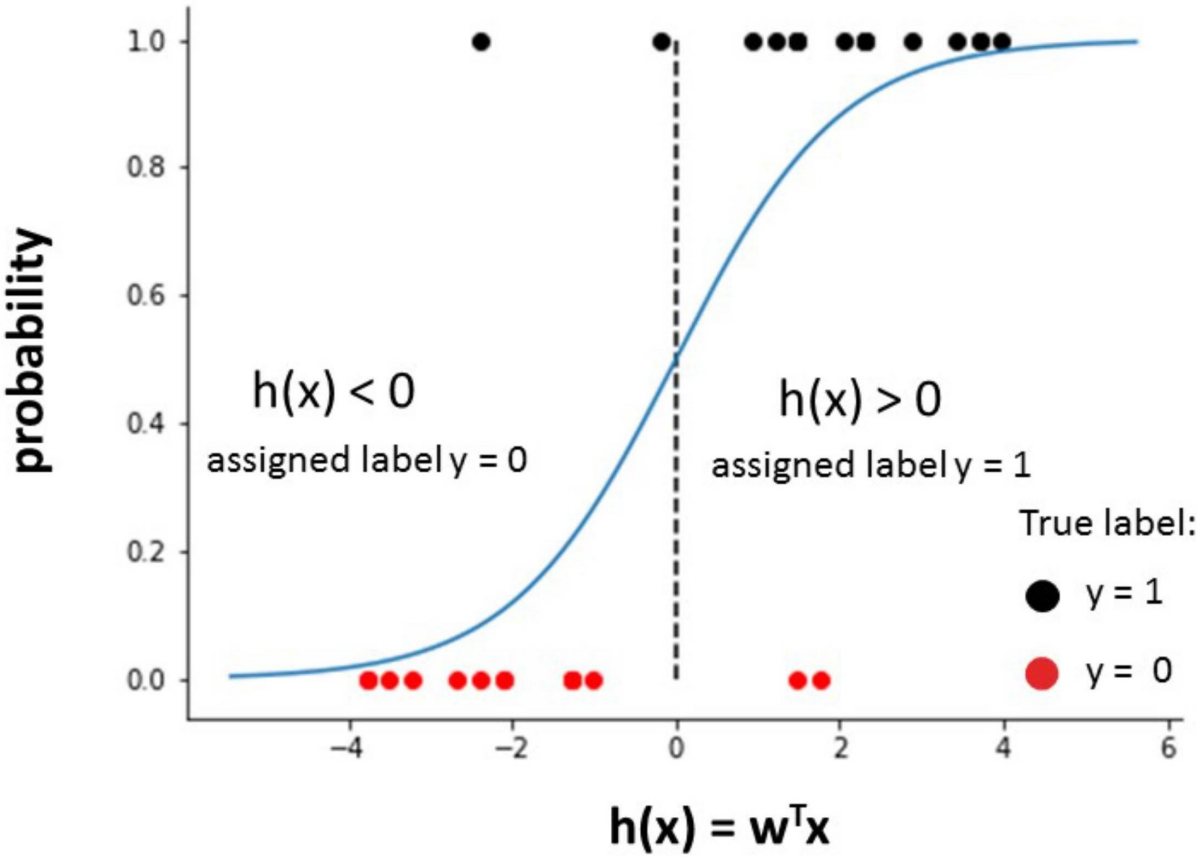


Logistic Regression

$$\log\frac{P(y=1)}{P(y=0)} = \log\frac{P(y=1)}{1-P(y=1)} = w_0 + \mathbf{w}^\mathbf{T}\mathbf{x}$$

$$P(y=1) = \frac{1}{1+exp(-(w_0+\mathbf{w}^\mathbf{T}\mathbf{x}))}$$

Logistic Regression



Fitting Logistic Regression

Problem:

- The usual performance metrics for classification (accuracy, F1, etc.) have no corresponding differentiable loss functions
- $\rightarrow$ We cannot efficiently find a predictor that minimizes these directly

Solution:

- Fit the model using a well-behaved loss function with (hopefully) sufficient carryover to performance metric
- Validate and select model using the actual metric of interest (e.g, accuracy)

Logistic Loss

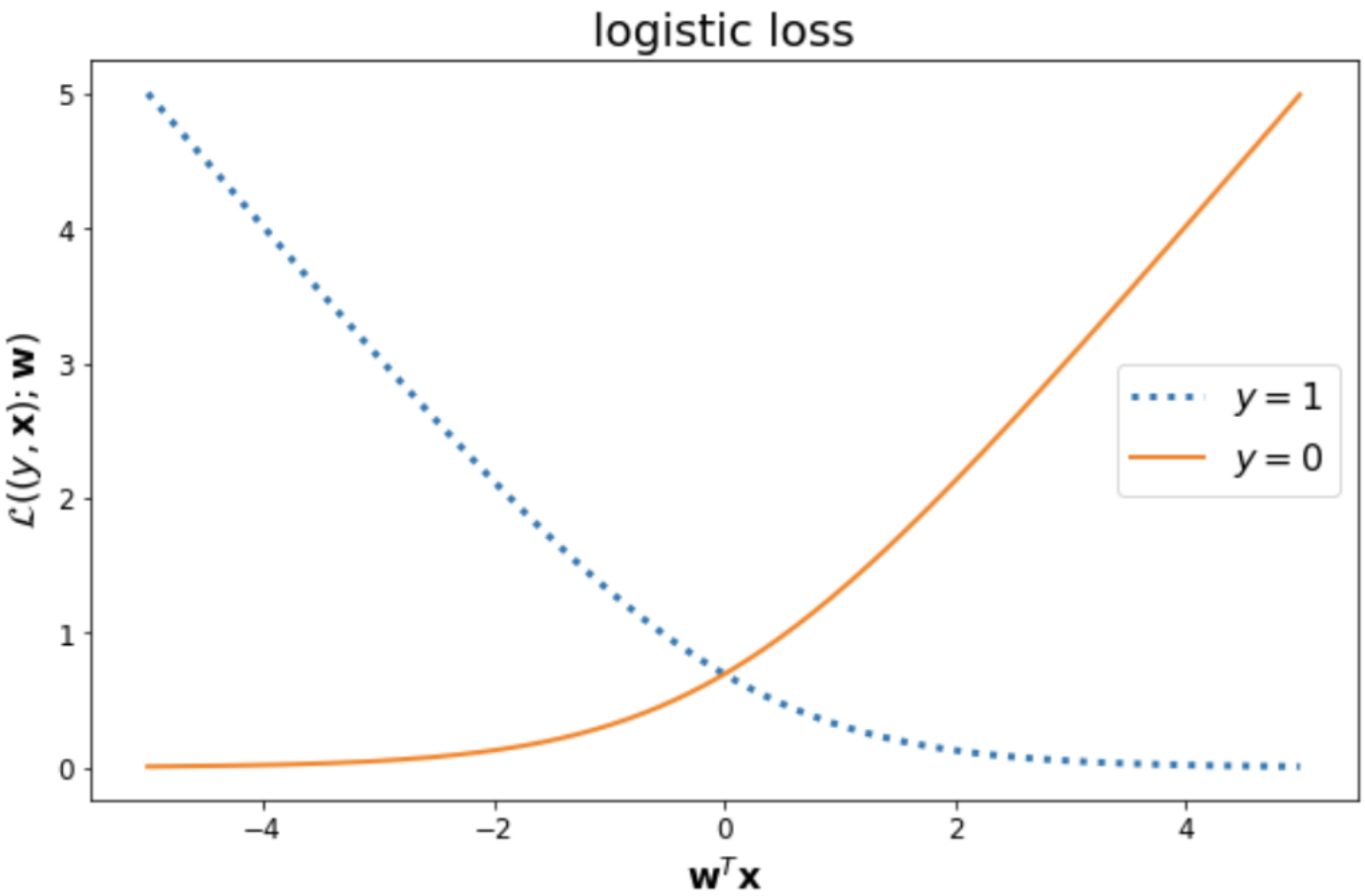
- Logistic regression model is fitted using Maximum Likelihood Estimation (MLE). Goal of MLE is to find $w_0, w_1, ..., w_n$, such it will maximize the probability of the observed labels y
- This is equivalent to minimimizing the average logistic loss:

$$(1/m) \sum_{i=1}^{m} \left[- y^{(i)} \ln \left(P(y = 1) \right) - (1 - y^{(i)}) \ln \left(P(y = 0) \right) \right]$$

- The incurred loss depends on the true label of the data point:

  For $y = 1$ the loss is $-ln(P(y = 1))$
  For $y = 0$ the loss is $-ln(P(y = 0))$

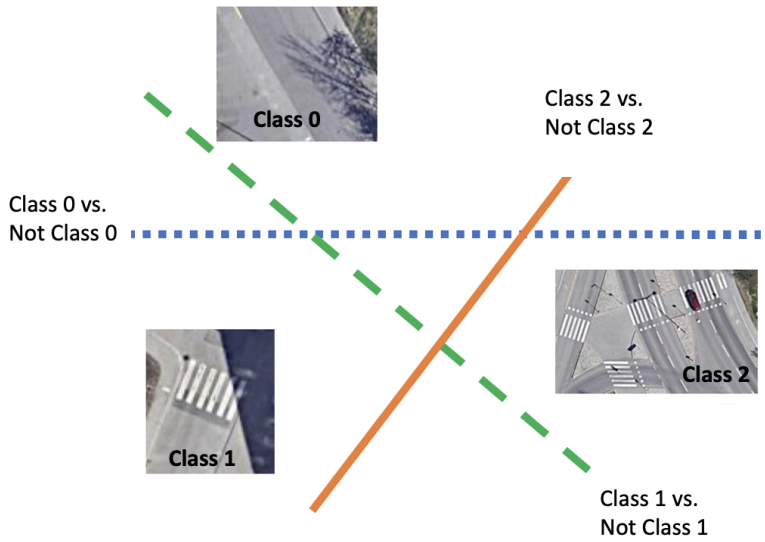
Logistic Loss



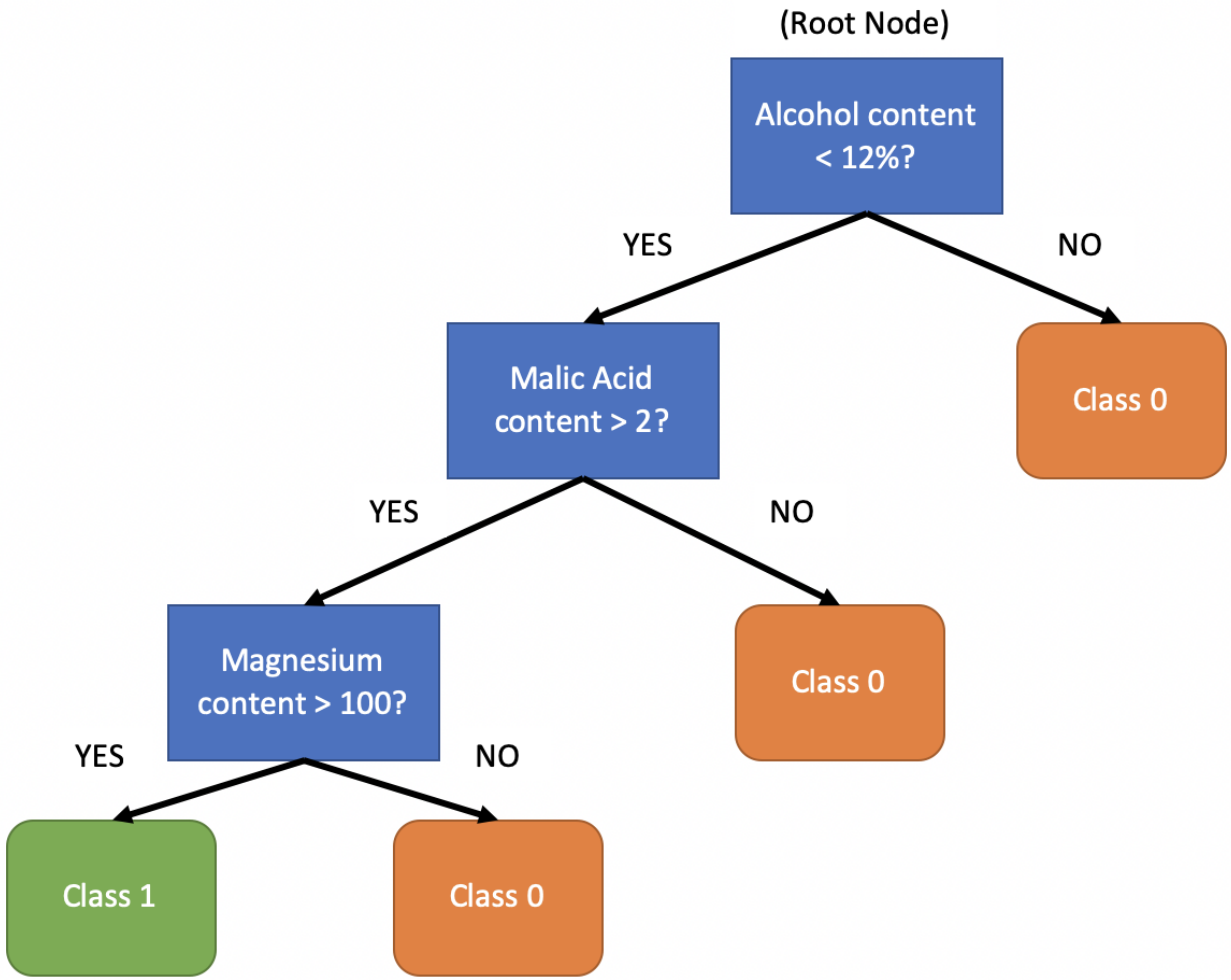
Multiclass Classification - "One-vs-rest"

- Simple extension of binary classification
- For each label, fit classification model for "label vs. not label", and assign the label with the highest probability to each data point

There are also other approaches, which often work better, but are more technical. E.g., multinomial logistic regression.

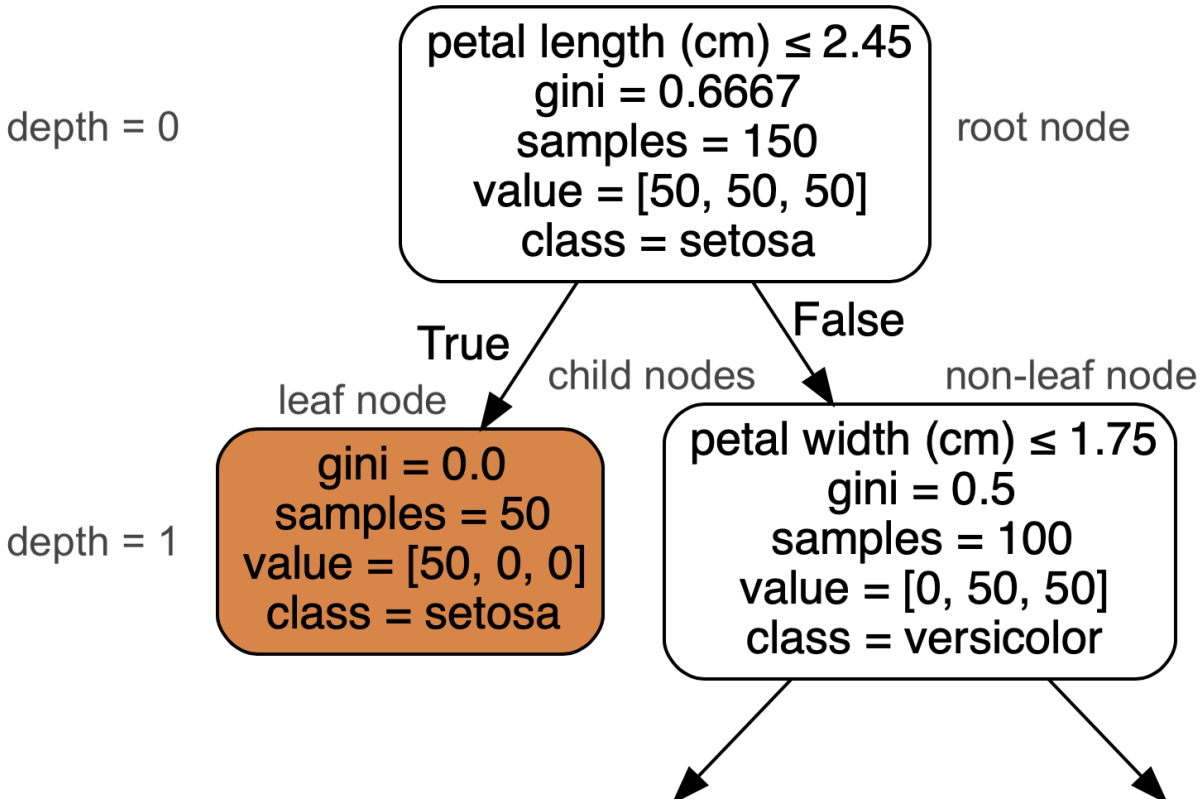


Decision Tree



Decision Tree

Gini criterion for node i: $G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2$



Decision Tree

Pros:

- Very interpretable
- No need for data preparation (standardization, etc.)
- Flexible hypothesis space, including complex predictors

Cons:

- Prone to severe overfitting
- Non-robust. Small changes in data result in very different models
- Training not as efficient, and globally optimum solution is not guaranteed

Decision Tree - Regularization

Hyperparameter tuning:

- Reduce the maximum depth of the tree
- Increase minimum number of data points in leaf nodes
- And so forth ... (check sklearn docs)

Random forest:

- Ensemble model with multiple decision trees
- A data point is classified using a consensus based on the predictions of all decision trees in the "forest"