Classification

Machine Learning for Behavioral Data

March 11, 2025

Today's Topic

Week	Lecture/Lab
1	Introduction
2	Data Exploration
3	Regression
4	**Classification**
5	Model Evaluation
6	Time Series Prediction
7	Time Series Prediction
8	Time Series Prediction

Complete pipeline for one use case:

- Data exploration
- Prediction
- Model evaluation

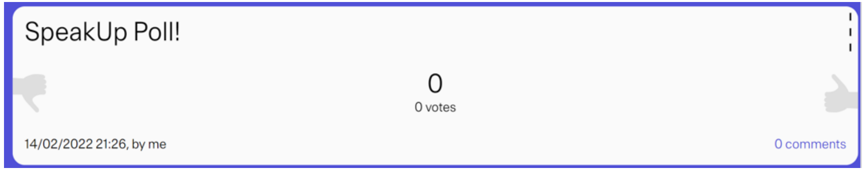
Getting ready for today’s lecture…

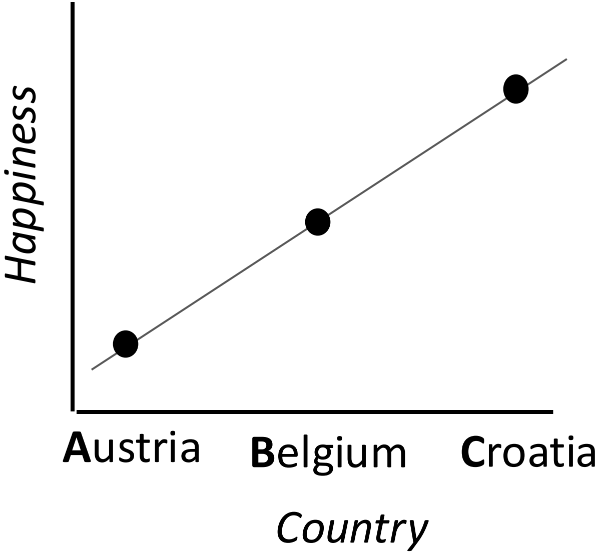
- **If not done yet**: clone the repository containing the Jupyter notebook and data for today’s lecture into your Noto workspace
- SpeakUp room for today’s lecture:

https://go.epfl.ch/speakup-mlbd2025

Short quiz about the past…

- [Exploration] Based on the provided graph, what can you say about the relationship between country and happiness?





a) Happiness increases with country

b) Happiness decreases with country

c) It is not possible to compute a correlation between country and happiness

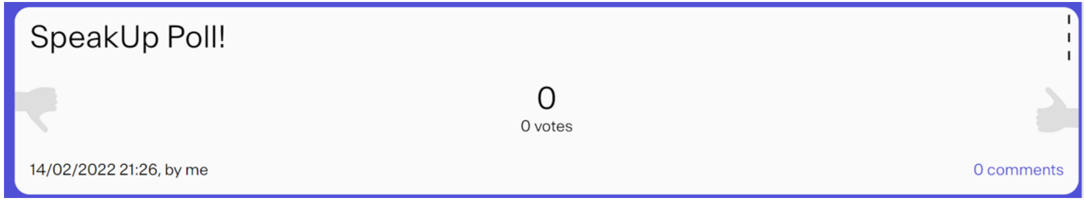
Short quiz about the past...

Which GLM family should you use when the output variable is continuous?

a) Binomial

b) Poisson

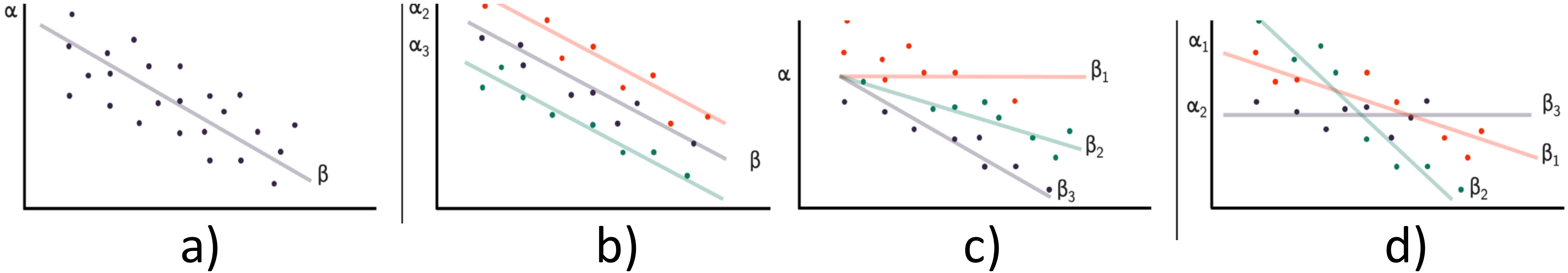
c) Gaussian

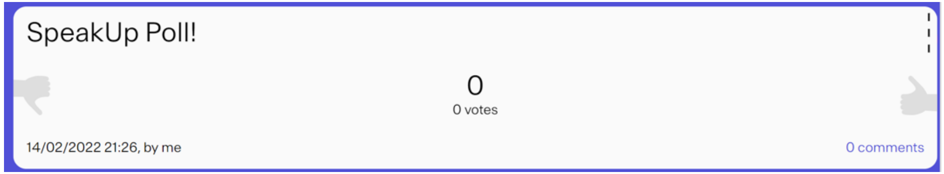



```
model = Lmer("grade ~ (1|user) + submissions_wrong",
             data=df_train, family='??????')
```

Short quiz about the past…

Which of the following are examples of models with **only fixed effects**? In the plots, α denotes intercept, β denotes slope.





https://go.epfl.ch/speakup-mlbd2025

Idea

- In classification, a single aspect of the data (predicted variable) is modeled by some combination of other aspects of the data (features)
  - The predicted variable is categorical (set of classes)
  - Examples:
    - Prediction of dropout in massive open online courses (binary)
    - Exploration of user categories in a simulation (multiclass)

Today’s Use Case: Flipped Classroom Course

- Participants: 288 EPFL students of a course taught in *flipped classroom* mode with a duration of 10 weeks
- Structure:
  - Preparation: watch videos (and solve simple quizzes) on **new content** at home as a preparation for the lecture
  - Lecture: discuss open questions and solve more complex tasks
  - Lab session: solve paper-an-pen assignments
- Data: clickstream data (all interactions of the student with the system)

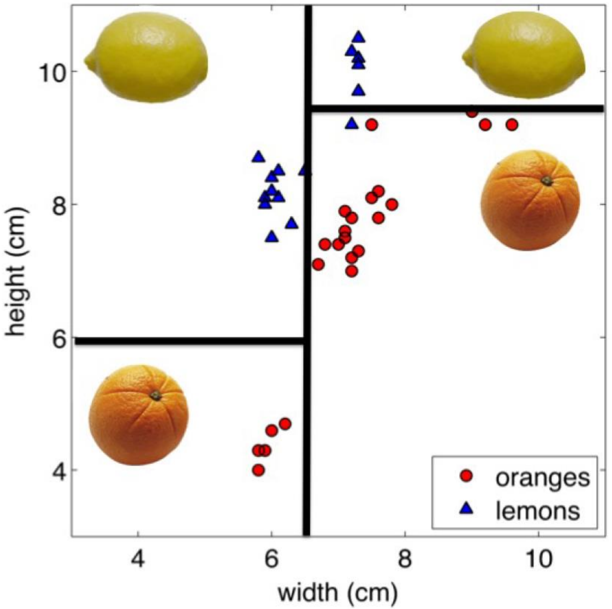
Agenda

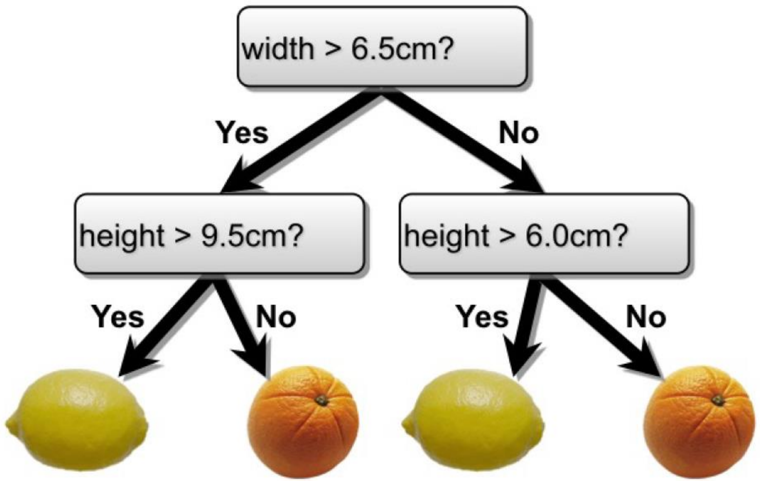
- **Traditional Methods:**
  - **Decision Trees and Random Forest**
  - K-Nearest Neighbor
  - Logistic Regression
- Performance Metrics
- Classification of Time Series

Decision Trees - Idea

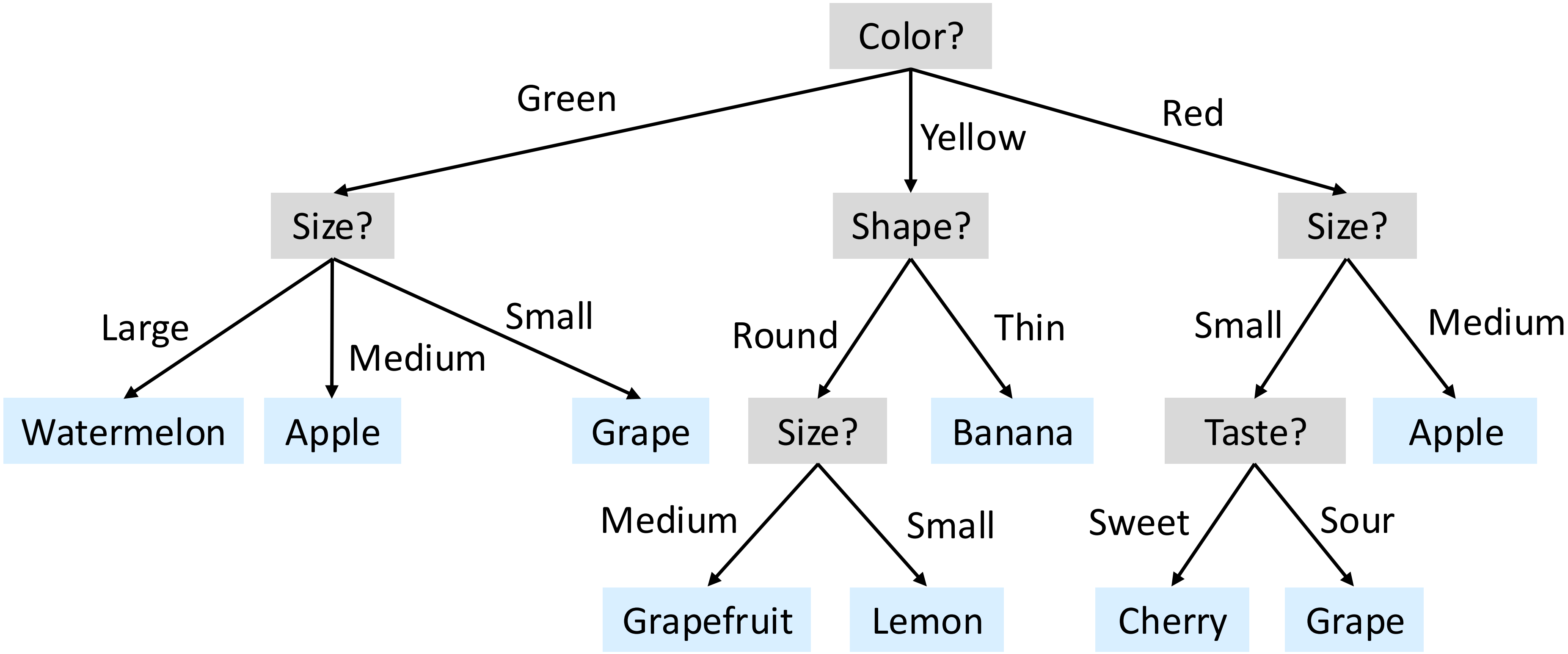
- Pick an attribute, do a simple test
- Conditioned on a choice, pick another attribute, do another test
- In the leaves, assign a class with majority vote
- Do other branches as well

Decision Trees - Example





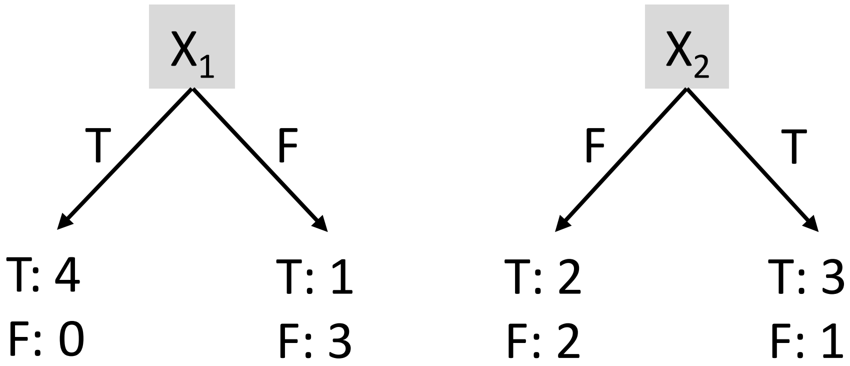
Categorical Features



Construction Algorithm

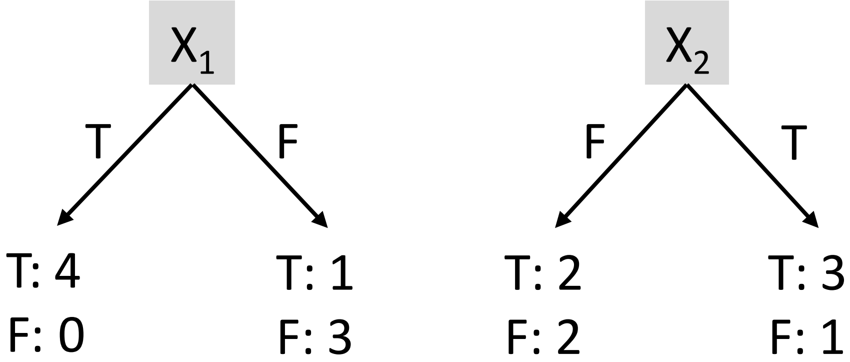
- Learning the simplest (smallest) decision tree is an NP complete problem
- Greedy heuristic:
  1. Pick an attribute to split at a non-terminal node
  2. Split example into groups based on attribute value
  3. For each group:
     - No examples -> return majority of parent node
     - All examples from same class -> return class
     - Else: loop to step 1

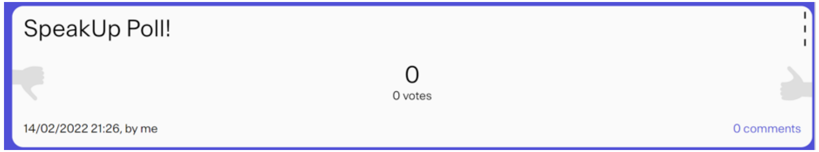
Picking the best attribute



X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

On which attribute would you split?



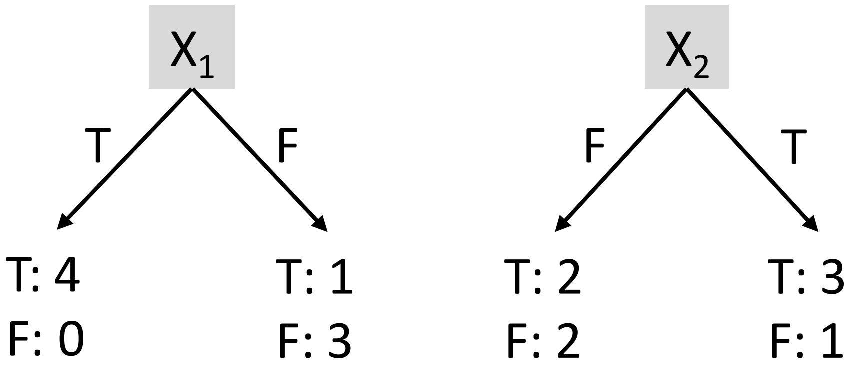


a) X_1
b) X_2

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

https://go.epfl.ch/speakup-mlbd2025

Picking the best attribute



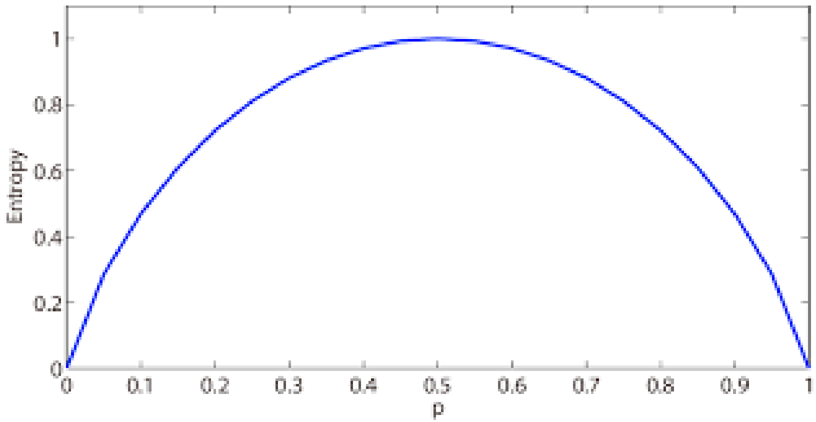
-> Information theory to the rescue!

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

Entropy

- Describes the level of "uncertainty" about a random variable’s possible outcome

$$H(X) = -\sum_{x \in X} p(x) \cdot \log_2 p(x)$$



Coin flip

Conditional Entropy

P(X,Y)

Y / X	Cloudy	Not Cloudy
Rain	0.24	0.01
No Rain	0.25	0.50

- Specific Conditional Entropy: what is the entropy of cloudiness Y, given that **it is raining**?

$$H(Y|X=x) = -\sum_{y \in Y} p(y|x) \cdot \log_2 p(y|x)$$

Conditional Entropy

P(X,Y)

X \ Y	Cloudy	Not Cloudy
Rain	0.24	0.01
No Rain	0.25	0.50

- Specific Conditional Entropy: what is the entropy of cloudiness Y, given that **it is raining**?

$$H(Y|X=x) = -\sum_{y \in Y} p(y|x) \cdot \log_2 p(y|x)$$

- Expected Conditional Entropy: what is the entropy of cloudiness Y, given "raininess" X?

$$H(Y|X) = -\sum_{x \epsilon X} p(x) \cdot H(Y|X=x)$$

Information Gain

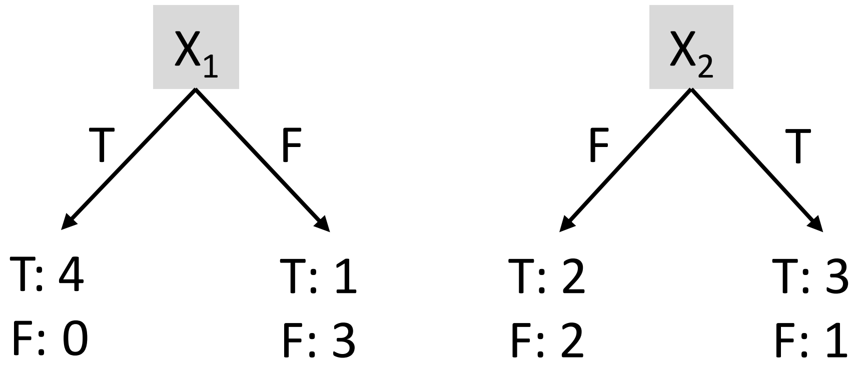
P(X,Y)

X \ Y	Cloudy	Not Cloudy
Rain	0.24	0.01
No Rain	0.25	0.50

- Information Gain: how much information about cloudiness (Y) do we get by discovering whether it is raining (X)?

$$\boxed{I(Y;X) = H(Y) - H(Y|X)}$$

Picking the best attribute



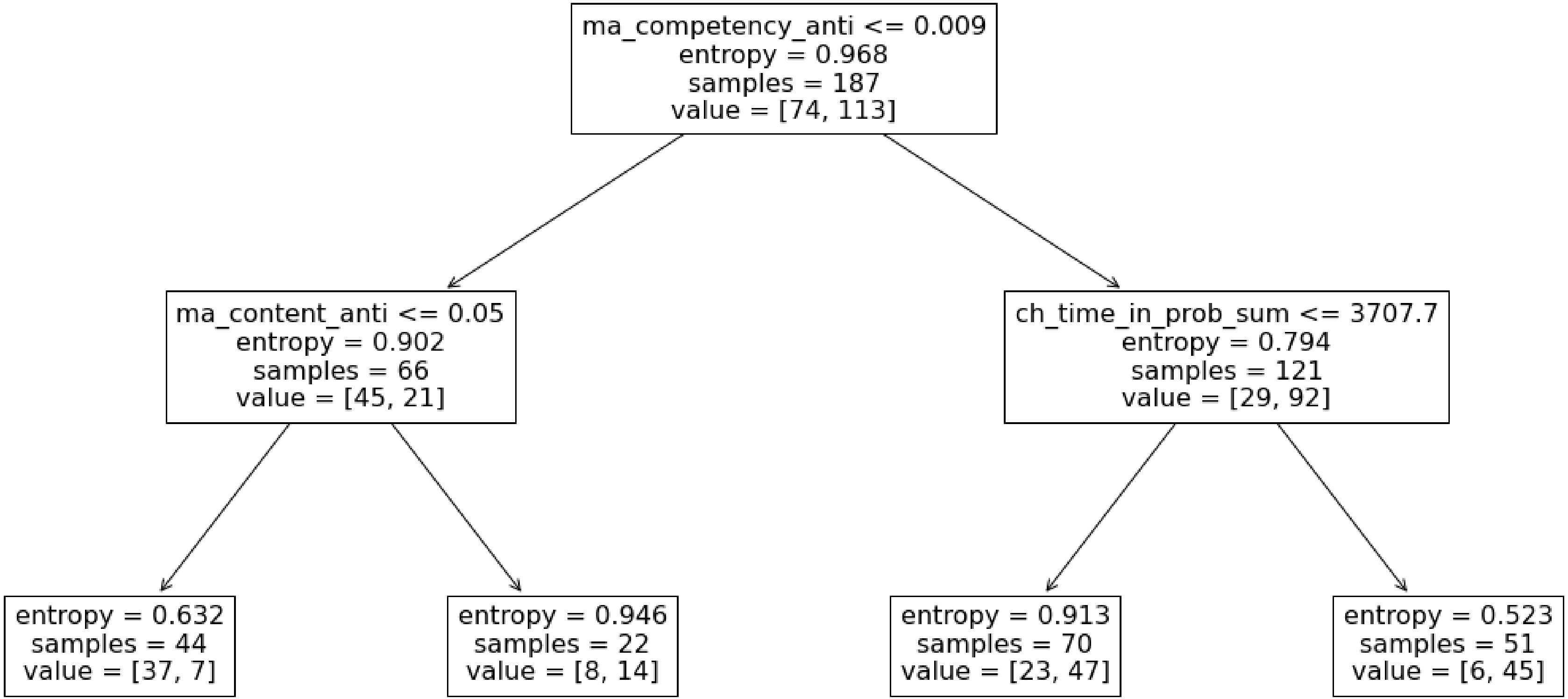
X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

What makes a good tree?

- Not too small: needs to handle important, but possibly subtle distinctions in data
- Not too big:
  - Avoid overfitting to training examples
  - Computational efficiency (avoid redundant, spurious attributes)

➡ Pruning strategies:
  - Stop splitting a node when the number of samples falls below a certain threshold
  - Grow a full tree, do bottom-up cross-validation: two leaves can be merged and labeled with the majority class if classification accuracy (on a validation set!) does not get worse

Decision Tree - Example



Random Forest - Idea

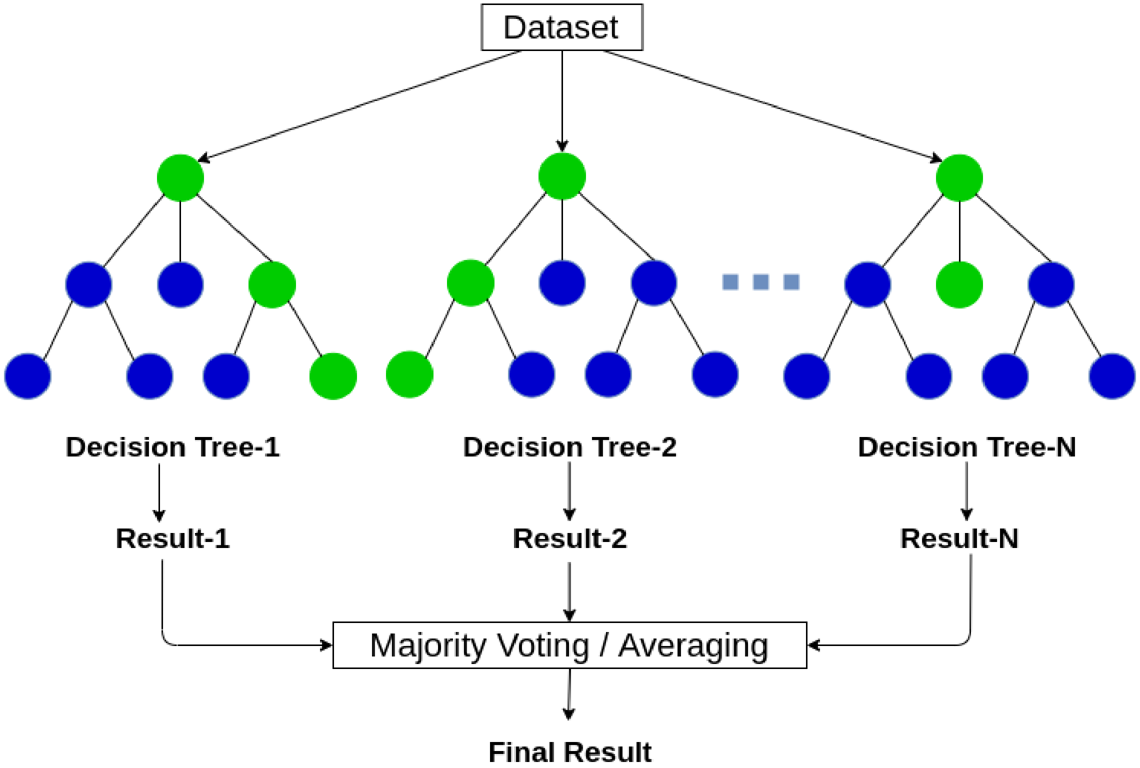
- **Ensemble Method**:
  - Take a collection of weak (simple) learners
  - Combine their predictions to obtain a better result
- **Bagging**: train learners on different samples of the data and then combine their output (e.g., majority vote or average)

Random Forest - Algorithm

Grow **K** trees on datasets sampled from the original data set (size N) with replacement (bootstrap samples), d = number of features.

- Draw *K bootstrap samples* of size N
- Grow each decision tree by selecting a *random set of* ***m*** *out of d features* at each node, and choosing the best feature to split on.
- Aggregate the predictions of the trees (majority vote or average) to produce the final prediction

Random Forest - Algorithm



[Image Source: Google Images]

Random Forest - Algorithm

Grow K trees on datasets sampl[...]et (size N) with replacement (bootstrap[...] features.

Each tree is trained on different data

- Draw *K bootstrap samples* of size N
- Grow each decision tree by selecting a *random set of m out of d features* at each node, and choosing the best feature to split on.
- Aggregate the predictions of the trees (majority vote or average) to produce the final prediction

Random Forest - Algorithm

Grow K trees on datasets sampled from the original data set (size N) w, d = number of featu

Corresponding nodes in different trees will (usually) not use the came feature for splitting

- Dr
- Grow each decision tree by selecting a *random set of m out of d features* at each node, and choosing the best feature to split on.
- Aggregate the predictions of the trees (majority vote or average) to produce the final prediction

Summary – Decision Trees & Random Forests

- Decision trees are simple, but
  - sensitive to small perturbations in the data
  - tend to overfit
- Random forests
  - Reduce overfitting in decision trees and can improve accuracy
  - Are versatile (classification, regression, continuous/categorical variables)
  - Easy to implement and parallelize
  - Most popular algorithm in practice (for dense data)
  - Not so easy to interpret…

Agenda

- **Traditional Methods:**
  - Decision Trees and Random Forest
  - **K-Nearest Neighbor**
  - Logistic Regression
- Performance Metrics
- Classification of Time Series

Instance-based Learning

- Alternative to parametric models are **non-parametric** models
- Learning amounts to simply storing training data
- Test instances are classified using similar training instances

kNN- Algorithm

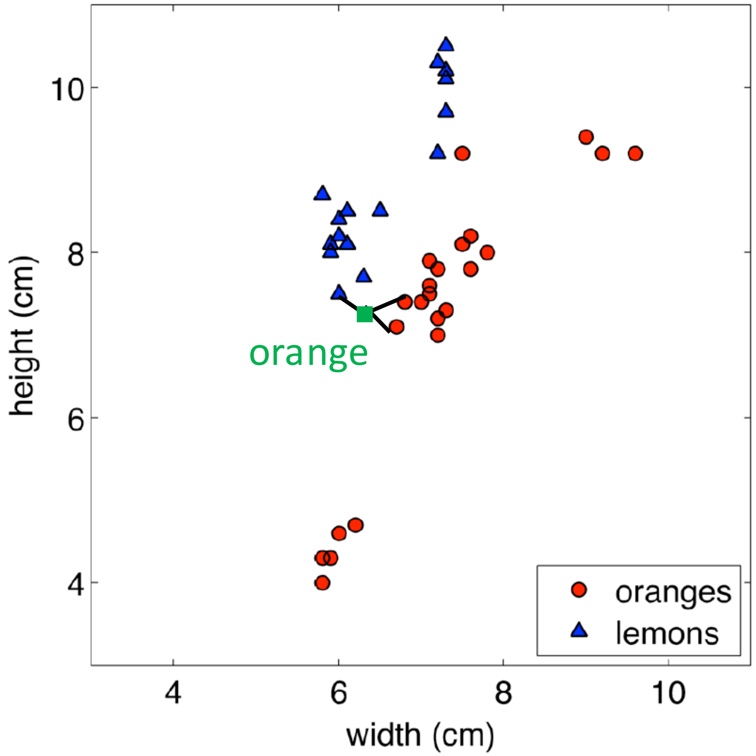
- Training data: samples $\boldsymbol{x}_n$ $(n = 1, \dots, N)$ with class labels c $(c = 1, \dots, C)$
- Classification of test sample $\boldsymbol{x}^*$
  - Find the k nearest neighbors of $\boldsymbol{x}^*$
  - Predicted class $\hat{c}^*$ = majority vote of k nearest neighbors
  - Option: give nearest neighbors different weights

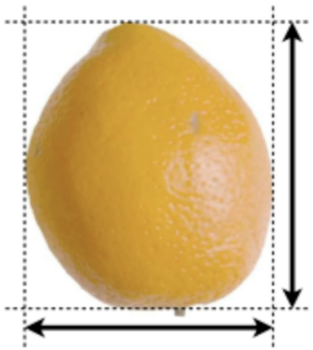
kNN- Algorithm

- Training data: samples $\boldsymbol{x}_n$ $(n = 1, \dots, N)$ with class labels c $(c = 1, \dots, C)$
- Classification of test sample $\boldsymbol{x}^*$
  - Find the k nearest neighbors of $\boldsymbol{x}^*$
  - Predicted class $\hat{c}^*$ = majority vote of k nearest neighbors
  - Option: give nearest neighbors different weights

Often based on distance, i.e. the smaller the distance, the higher the weight

Example: 3-nearest neighbor





Choosing k

- Larger k may lead to better performance
- But if we set k too large we may end up looking at samples that are not neighbors (are far away from the query)
- Find k using leave-one-out cross-validation:
  - For each point x_n in the training data set, find the k nearest neighbors from the set of all *other* training samples
  - Predict $\hat{c}_n$ as the majority vote of the k nearest neighbors
  - Measure the classification accuracy for different values of k
  - Choose k with the highest classification accuracy on the training data set

kNN: what similarity metric could we use?

- Comparing the number of sessions students had in a MOOC:

  u_1: [5,3,7,8,10,2] u_2: [1,9,10,2,3,2] u_3: [4,5,2,8,7,6]

- Comparing users by the movies they have watched:

  u_1: {"Frozen", "The Horse Whisperer", "Follow Me", "Notting Hill" }

  u_2: {"Die Hard 1", "The Father", "Frozen", "Black Panther", "Casablanca" }

  u_3: {The Dark Knight", "Die Hard 1", "Wonder Woman", "Black Panther", "Logan", "Up" }

- Comparing weeks days of users (W: Work, O: Off, S: Sick)

  $u_1: [O, W, W, O, W]$ $u_2: [W, W, W, W, W]$ $u_3: [W, W, S, W, O]$

- Comparing sequences of actions of users

  $u_1: [O, A, P, E, T, F, G, F, G, H, I, O, N, K, U, P, E, L]$ $u_2: [O, S, I, E, P, L]$ $u_3: [O, R, C, C, T, A, A, S, S, P, L]$

- Comparing the relative amount of time users spent on watching videos, solving quizzes, etc.

  $u_1: [0.2, 0.3, 0.1, 0.1, 0.3]$ $u_2: [0.8, 0.1, 0.0, 0.0, 0.1]$ $u_3: [0.1, 0.5, 0.3, 0.0, 0.1]$

kNN: similarity metrics

- Euclidean Distance: simple & fast

$$d(x, y) = \|x - y\|$$

- Jaccard Distance: for set data

$$d(X, Y) = 1 - \frac{|X \cap Y|}{|X \cup Y|}$$

- Hamming distance: for strings (with the same length)

$$d(x, y) = \sum_{i=1}^{n} (x_i \neq y_i)$$

kNN: similarity metrics

- Levensthein distance: minimal number of single character edits (insertion, deletion, substitution) to change one string into the other
- Longest common subsequence (LCS): string similarity measure, find the longest common subsequence between two sequences
- Kullback-Leibler Divergence: measures difference between two probability distributions

$$D_{KL}(P||Q) = \sum_{x \in \mathrm{X}} P(x) \cdot \log(\frac{P(x)}{Q(x)})$$

kNN: what similarity metric could we use?

- Comparing the number of sessions students had in a MOOC:

  u_1: [5,3,7,8,10,2] u_2: [1,9,10,2,3,2] u_3: [4,5,2,8,7,6]

- Comparing users by the movies they have watched:

  u_1: {"Frozen", "The Horse Whisperer", "Follow Me", "Notting Hill" }

  u_2: {"Die Hard 1", "The Father", "Frozen", "Black Panther", "Casablanca" }

  u_3: {The Dark Knight", "Die Hard 1", "Wonder Woman", "Black Panther", "Logan", "Up" }

- Comparing weeks days of users (W: Work, O: Off, S: Sick)

  $u_1: [O, W, W, O, W]$ $u_2: [W, W, W, W, W]$ $u_3: [W, W, S, W, O]$

- Comparing sequences of actions of users

  $u_1: [O, A, P, E, T, F, G, F, G, H, I, O, N, K, U, P, E, L]$ $u_2: [O, S, I, E, P, L]$ $u_3: [O, R, C, C, T, A, A, S, S, P, L]$

- Comparing the relative amount of time users spent on watching videos, solving quizzes, etc.

  u_1: [0.2, 0.3, 0.1, 0.1, 0.3] u_2: [0.8, 0.1, 0.0, 0.0, 0.1] u_3: [0.1, 0.5, 0.3, 0.0, 0.1]

kNN: similarity metrics

- There are many more similarity metrics (e.g., Cosine distance, Manhattan distance, Mahalanobis distance)
- These are all standard metrics – we will look detailed into metrics for comparing sequences in later lectures

Summary - kNN

- Naturally forms complex decision boundaries
- Works well if we have a lot of samples
- Issues:
  - Scales linearly with the number of samples
  - High-dimensional data

Agenda

- **Traditional Methods:**
  - Decision Trees and Random Forest
  - K-Nearest Neighbor
  - **Logistic Regression**
- Performance Metrics
- Classification of Time Series

Logistic Regression revisited

$$\log\left(\frac{y_n}{1-y_n}\right) = \beta_0 + \beta_1 x_{n,1} + \cdots + \beta_D x_{n,D} = \boldsymbol{\beta}\tilde{\boldsymbol{x}}_{\boldsymbol{n}}$$

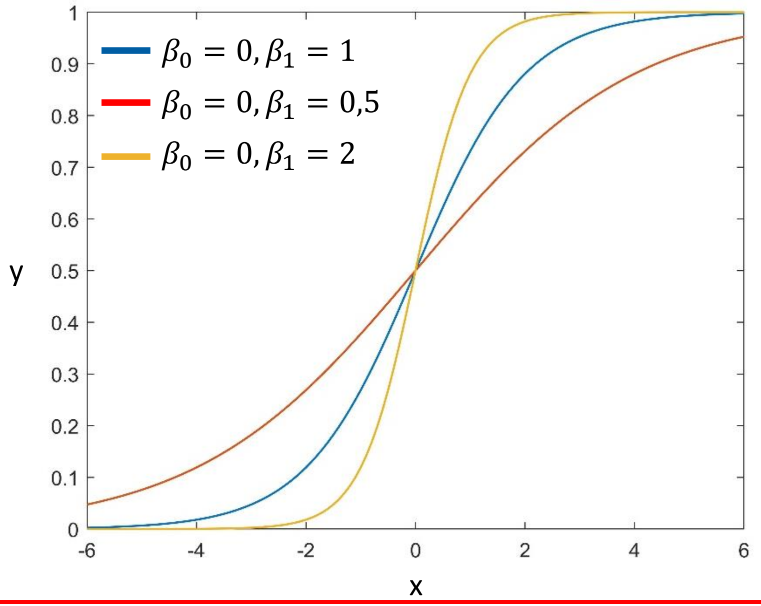
$$\Downarrow$$

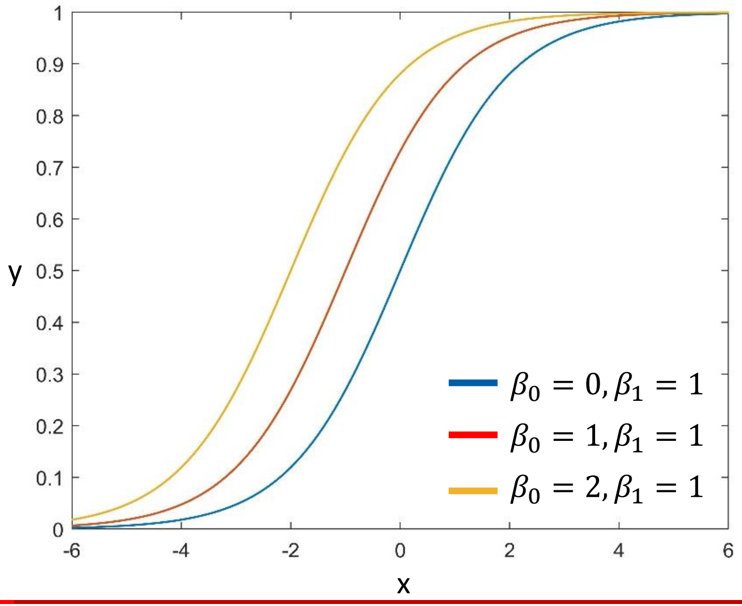
$$y_n = \frac{1}{1+e^{-\boldsymbol{\beta}\tilde{\boldsymbol{x}}_{\boldsymbol{n}}}}$$

Remember: y_n with $n = 1, \ldots, N$ (N samples) are the output variables and $x_{n,d}$ with $d = 1, \ldots, D$ (D dimensions) are the input variables

Example with 1 dimension

$$y_n = \frac{1}{1 + e^{-\beta_0 - \beta_1 x_n}}$$





Logistic regression as a binary classifier

- Probabilistic interpretation: we have a value between 0 and 1, let's use it to model class probability

$$P(C = 1|\boldsymbol{x}) = \frac{1}{1 + \boldsymbol{e}^{-\boldsymbol{\beta}\tilde{\boldsymbol{x}}}} \quad \Rightarrow \quad P(C = 0|\boldsymbol{x}) = 1 - P(C = 1|x)$$

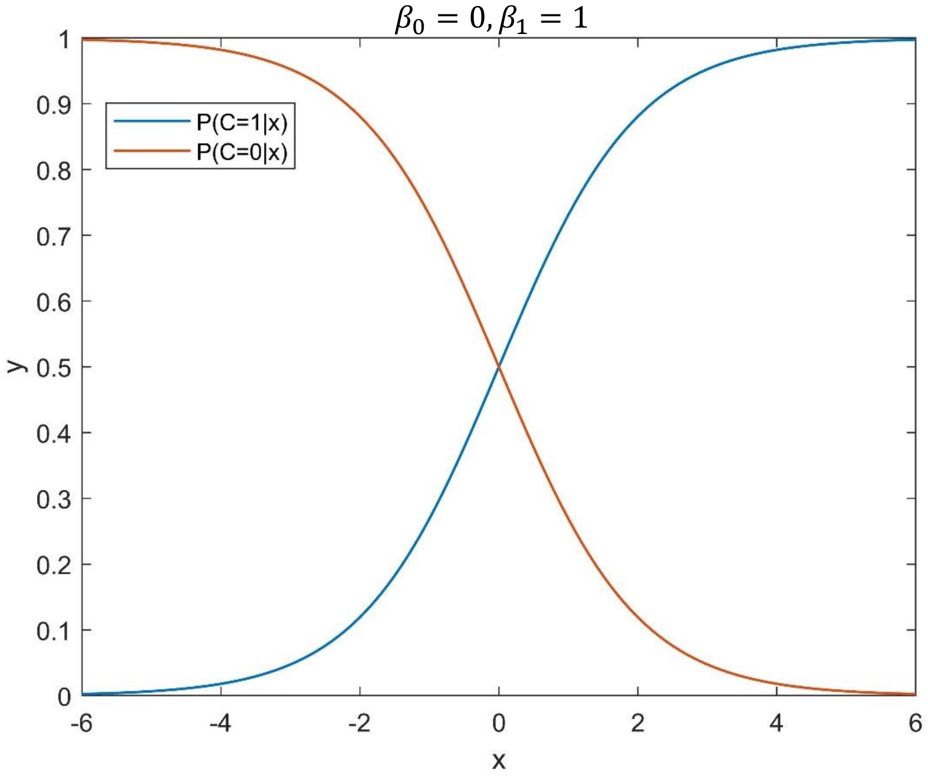
Logistic regression as a binary classifier

- Probabilistic interpretation: we have a value between 0 and 1, let's use it to model class probability

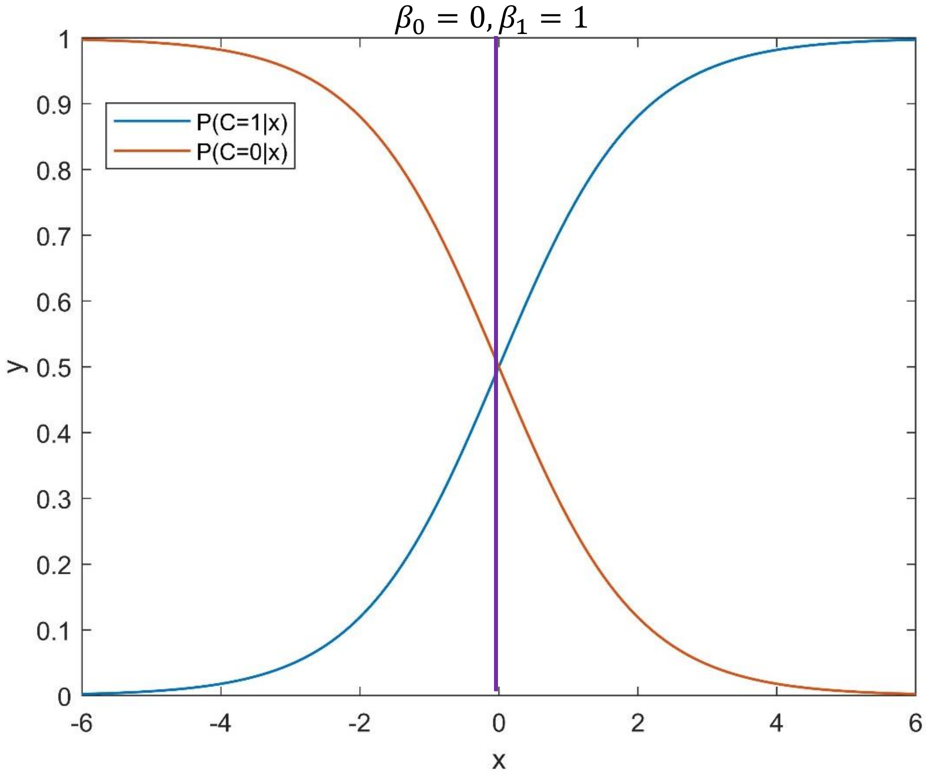
$$P(C=1|\boldsymbol{x}) = \frac{1}{1+\boldsymbol{e}^{-\boldsymbol{\beta}\widetilde{\boldsymbol{x}}}} \Rightarrow P(C=0|\boldsymbol{x}) = 1 - P(C=1|x)$$

- Binary classification: add decision boundary (e.g., we predict $C = 1$ if $P(C=1|\boldsymbol{x}) > 0.5$)

Example with 1 dimension



Example with 1 dimension

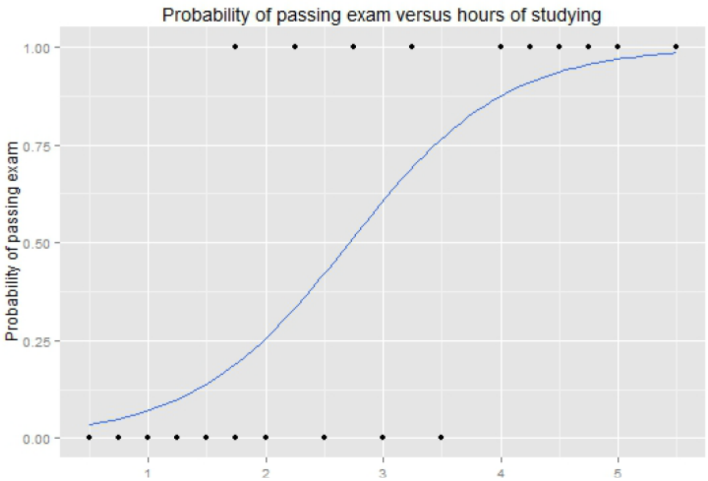


Set threshold to 0.5, i.e. predict $C = 1$ if $P(C = 1|\boldsymbol{x}) > 0.5$

Example: passing the exam

- **Problem**: given the number of hours the student spent learning, will (s)he pass the exam?

Hours	0.50	0.75	1.00	1.25	1.50	1.75	1.75	2.00	2.25	2.50	2.75	3.00	3.25	3.50	4.00	4.25	4.50	4.75	5.00	5.50
Pass	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1



Hours of study	Probability of passing exam
1	0.07
2	0.26
3	0.61
4	0.87
5	0.97

[Slide credit: Zemel, Urtasun, Fidler (UofT)]

Summary – Logistic Regression

- Advantages:
  - Natural probabilistic view of class predictions
  - Quick to train
  - Good accuracy for many simple data sets
  - Interpretability of model coefficients
- Disadvantages:
  - Linear decision boundary (too simple for more complex problems)?

Traditional Methods

- Decision Trees and Random Forest
- K-Nearest Neighbor
- Logistic Regression

These are just examples of classification algorithms, there are others out there (e.g., Naïve Bayes, Support Vector Machines)

Agenda

- Traditional Methods:
  - Decision Trees and Random Forest
  - K-Nearest Neighbor
  - Logistic Regression
- **Performance Metrics**
- Classification of Time Series

Classification: Accuracy

$$\boxed{Acc = \frac{|agreements|}{N}}$$

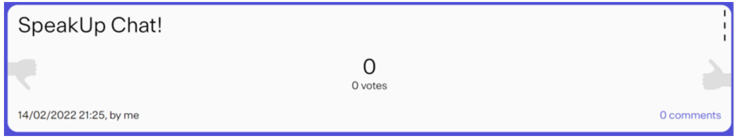
where $agreements$ means that the predicted outcome is equal to the observed outcome

- Easy to interpret
- Works for binary and multi-class
- General agreement across fields: accuracy is not a good standalone performance metric

Classification: Accuracy

$$\boxed{Acc = \frac{|agreements|}{N}}$$

where $agreements$ means that the predicted outcome is equal to the observed outcome



- Easy to interpret
- Works for binary and multi-class
- General agreement across fields: accuracy is not a good standalone performance metric

Why?

Classification: Balanced accuracy

$$\boxed{Acc_{bal} = \frac{1}{|C|} \cdot \sum_{c \in C} Acc_c}$$

where $|C|$ denotes the number of classes

- Easy to interpret: average accuracy over all classes
- Takes into account class imbalance
- Works also for the multi-class case

Classification: Confusion Matrix

		True Outcome	
		Positive	Negative
Predicted Outcome	Positive	True Positive (tp)	False Positive (fp)
	Negative	False Negative (fn)	True Negative (tn)

Confusion matrix for a binary classification task (two classes denoted as positive and negative class)

Classification: Specificity, sensitivity,…

		True Outcome	
		Positive	Negative
Predicted Outcome	Positive	True Positive (tp)	False Positive (fp)
	Negative	False Negative (fn)	True Negative (tn)

$$specificity = \frac{tn}{tn + fp} \quad sensitivity = \frac{tp}{tp + fn}$$

Classification: Specificity, sensitivity,…

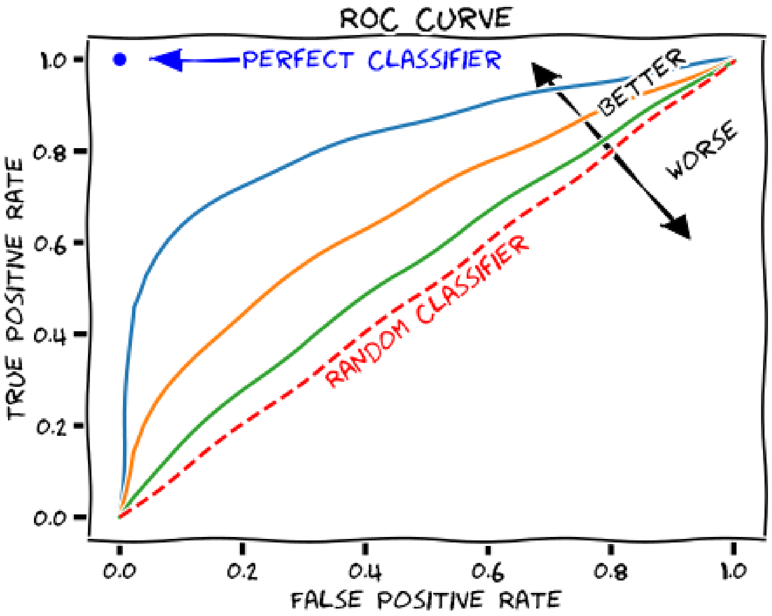
		True Outcome	
		Positive	Negative
Predicted Outcome	Positive	True Positive (tp)	False Positive (fp)
	Negative	False Negative (fn)	True Negative (tn)

$$specificity = \frac{tn}{tn + fp} \quad sensitivity = \frac{tp}{tp + fn}$$

- These metrics are used in addition to accuracy
- Sensitivity and specificity are often used in the fields of Psychology, Learning Sciences, etc.

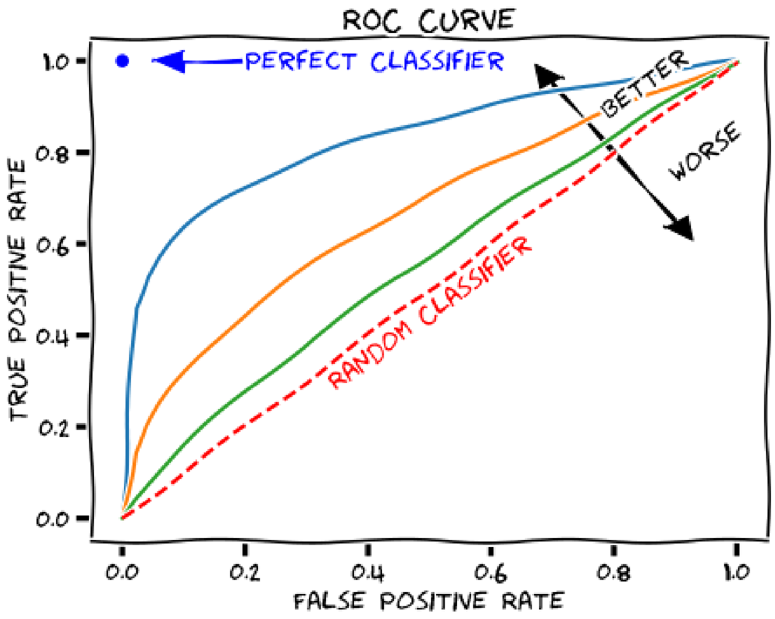
ROC curve

$$TPR = \frac{tp}{P}$$



$$FPR = \frac{fp}{N}$$

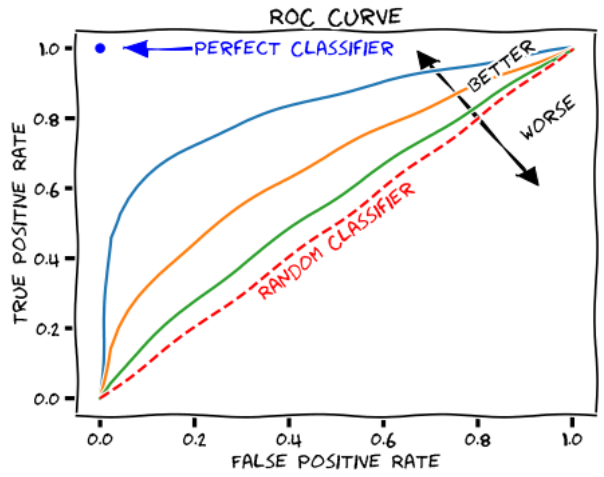
Computing the ROC curve



- Given a classifier $f(\boldsymbol{x})$ predicting some type of score (e.g., a probability)
- Choose a threshold t:
  - $f(x_i) \geq t$: predict positive class
  - $f(x_i) < t$: predict negative class
- Compute TPR and FPR given t
- Repeat for different thresholds (e.g., in case of probabilities choose $t = 0{,}0.1, \dots 1$)

Area under the ROC curve (AUC)

- The AUC denotes the area under the ROC curve
- A perfect classifier has an AUC of 1
- A random classifier has an AUC of 0.5
- Often used as a performance metric in more technical fields (e.g., educational data mining)
- The AUC can be extended to the multi-class case by considering all possible pairs of classes



Classification: Summary

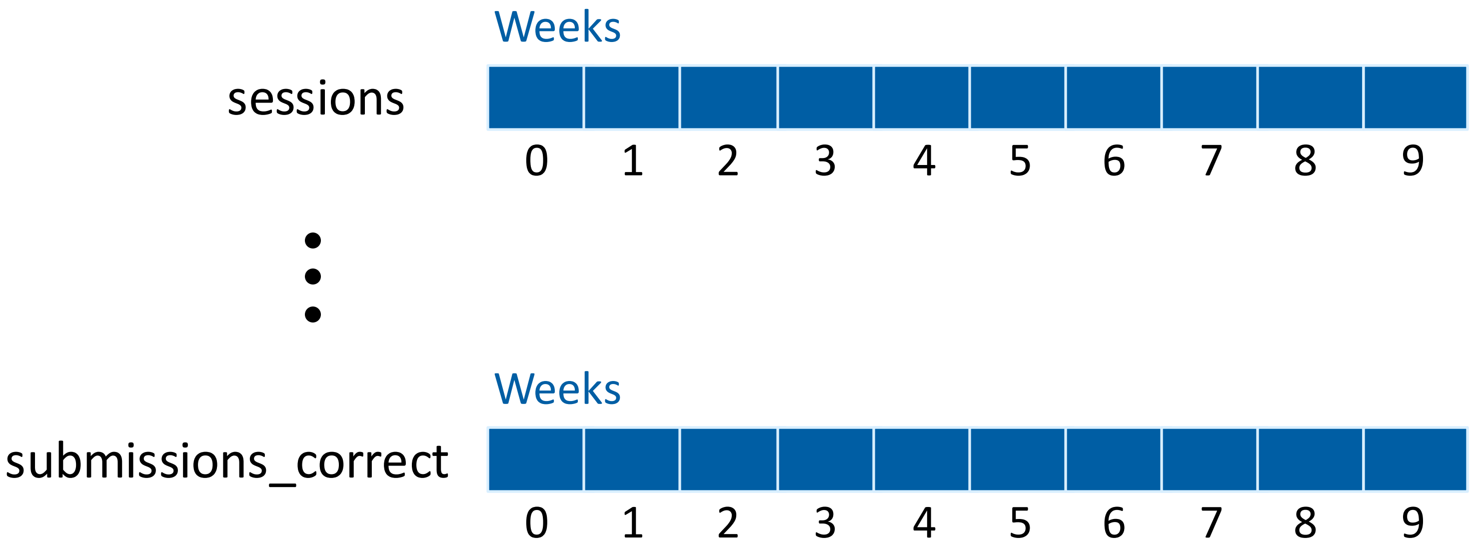
- Do:
  - Carefully think about the choice of metric (or combination)
  - Some ideas:
    - Use accuracy plus sensitivity and specificity [binary]
    - Use (balanced) accuracy plus AUC [binary + multi-class]
    - Use just AUC [binary + multi-class]
- Don’t:
  - Use accuracy as a standalone metric
  - Compute “all” possible metrics that come to your mind

Agenda

- Traditional Methods:
  - Decision Trees and Random Forest
  - K-Nearest Neighbor
  - Logistic Regression
- Performance Metrics
- **Classification of Time Series**

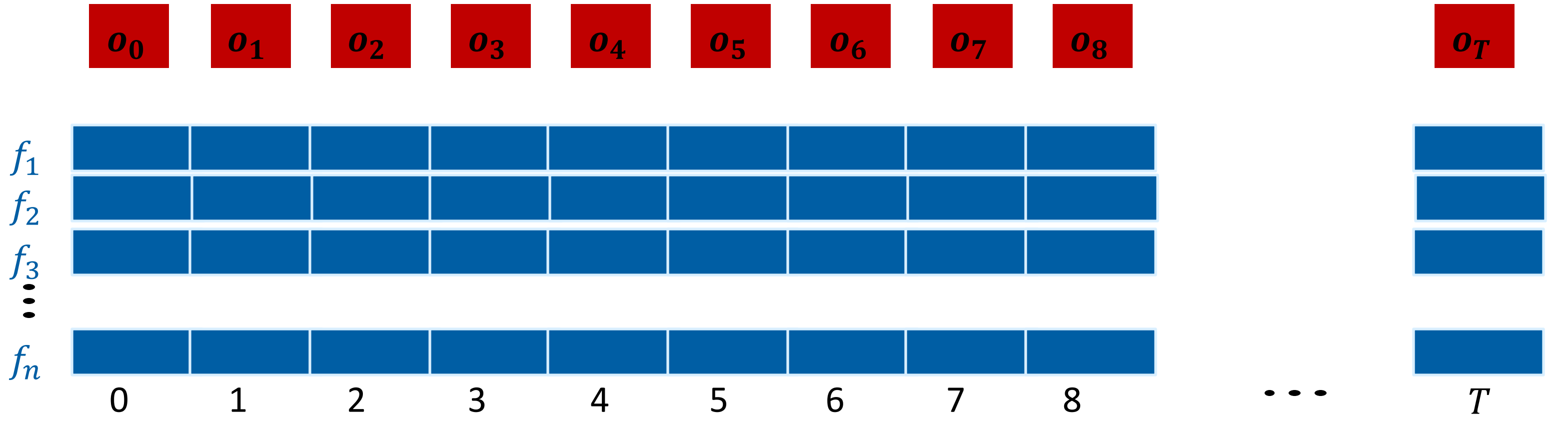
Time Series – Our flipped classroom case

Student i



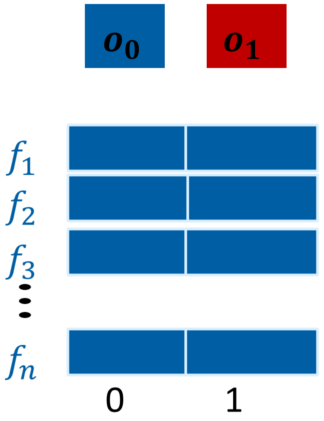
Time Series – Tracing Task

- Prediction of a **categorical** target variable after $t < T$ time steps, where T is the total number of time steps
- Prediction of a variable in time step $t + 1$, based on time steps $0, \ldots, t$.



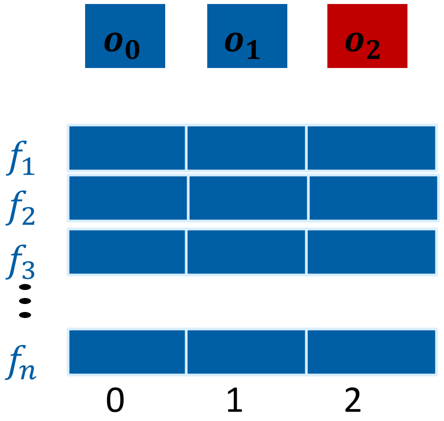
Time Series – Tracing Task

- Prediction of a **categorical** target variable after $t < T$ time steps, where T is the total number of time steps
- Prediction of a variable in time step $t + 1$, based on time steps $0, \dots, t$.



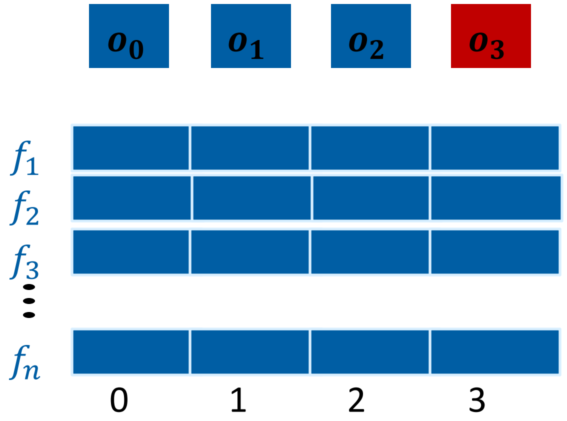
Time Series – Tracing Task

- Prediction of a **categorical** target variable after $t < T$ time steps, where T is the total number of time steps
- Prediction of a variable in time step $t + 1$, based on time steps $0, \ldots, t$.



Time Series – Tracing Task

- Prediction of a **categorical** target variable after $t < T$ time steps, where T is the total number of time steps
- Prediction of a variable in time step $t + 1$, based on time steps $0, \dots, t$.



Time Series – Tracing Task

Last Week:

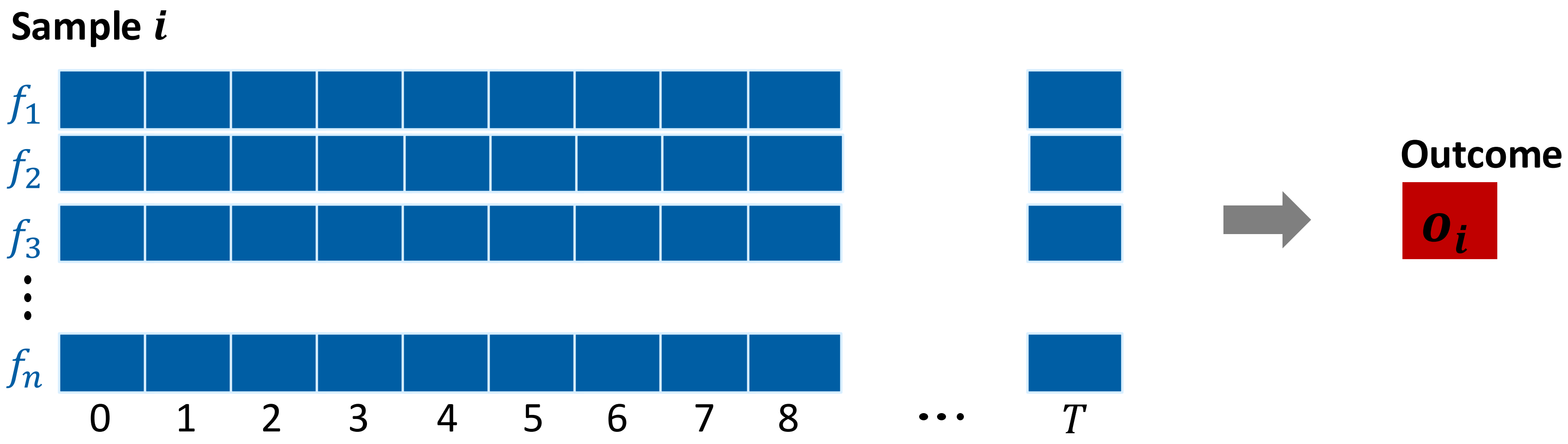
- we have solved this task for a *numerical* target variable (students' performance in weekly quizzes) using a linear mixed effect model

This Week:

- for a *binary* target variable, this task can be solved using a logistic (mixed effect) model
- The other presented algorithms (last week) are not suitable for this task (they are suitable for regression tasks).

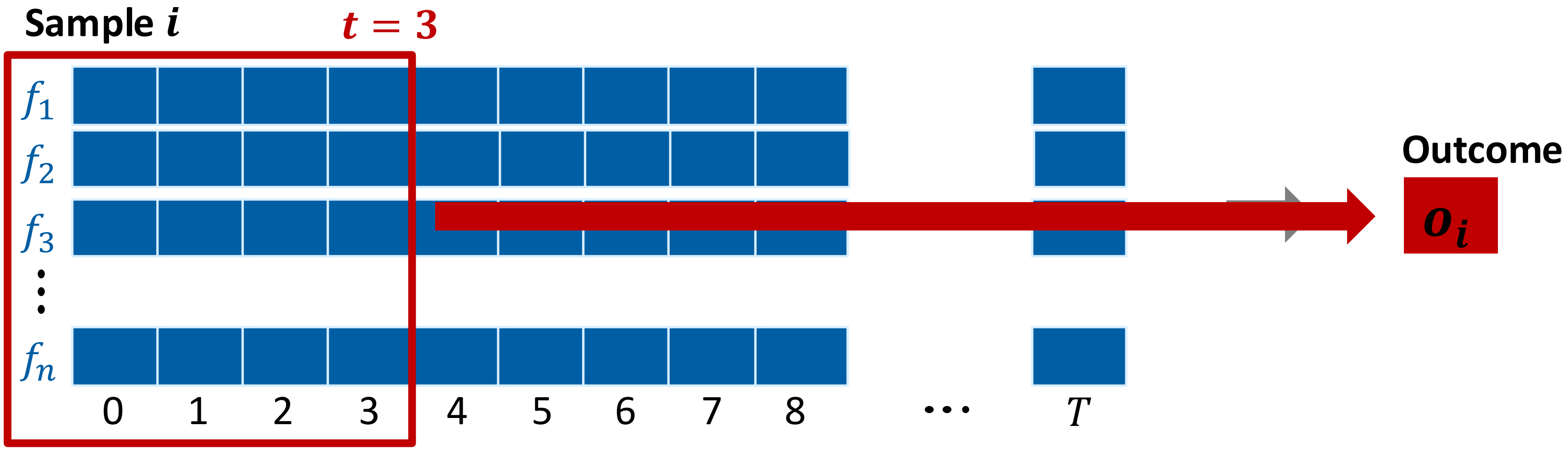
Time Series – Prediction Task

- Prediction of a **binary** target variable after $t < T$ time steps, where T is the total number of time steps



Time Series – Prediction Task

- Prediction of a **binary** target variable after $t < T$ time steps, where T is the total number of time steps



Your Turn

- Predict whether students will pass the course after $t = 5$ weeks (i.e. after half of the course)
- We provide you the train-test split to use in the Jupyter Notebook
- You can choose the classifier: Decision Tree, Random Forest, or k-Nearest Neighbor

Your Turn – Some Hints

- For Decision Tree and Random Forest you will need to use one of the following:
  - Flattening
  - Aggregation (hint: we have aggregated features last week)
- For k-Nearest Neighbor, you can compute pairwise distances between vectors
  - If you have several features, compute a pairwise distance matrix separately for each feature and then sum the distance matrices up
  - Distance matrices can have different scales (hint: MinMaxScaler from *sklearn*)

Your Turn – Feedback

Do you want feedback or have questions?
Upload your Jupyter Notebook here:
https://go.epfl.ch/mlbd-activities

Reminder

M2 is due on March 14 (at 23:59).
Submission through Moodle.