```
begin
    using Pkg
    Pkg.activate(joinpath(Pkg.devdir(), "MLCourse"))
    using CSV, DataFrames, Distributions, Plots, MLJ, MLJLinearModels, Random,
          Statistics, OpenML, MLJDecisionTreeInterface, MLJFlux, Flux, MLCourse
end
```

Non-Linear Methods

We load the precipitation training and test data from a csv file on the harddisk to a DataFrame. Our goal is to predict whether there is some precipitation (rain, snow etc.) on the next day in Pully, getting measurements from different weather stations in Switzerland.

```
precipitation_training = CSV.read(joinpath(@__DIR__, "..", "data", "project",
"trainingdata.csv"), DataFrame);
```

```
test_data = CSV.read(joinpath(@__DIR__, "..", "data", "project", "testdata.csv"),
DataFrame);
```

First we have to prepare our data set by filling in the missing values with some standard values with the help of `FillImputer` that fills in the median of all values.

```
precipitation_training_med = MLJ.transform(fit!(machine(FillImputer(),
                select(precipitation_training, Not(:precipitation_nextday)))),
                select(precipitation_training, Not(:precipitation_nextday)));
```

```
precipitation_training_med.precipitation_nextday =
    precipitation_training[:,:precipitation_nextday];
```

```
training_data = coerce!(precipitation_training_med, :precipitation_nextday =>
Binary); # with this we tell the computer to interpret the data in column
precipitation_nextday as binary data.
```

Then we standardize our training and test datas.

```
mach_train = machine(Standardizer(features=[:precipitation_nextday,
:ALT_sunshine_4], ignore=true), training_data);
```

```
fit!(mach_train);
```

```
stand_train = MLJ.transform(mach_train, training_data);
```

```
mach_test = machine(Standardizer(features=[:ZER_sunshine_1, :ABO_sunshine_4,
:ALT_sunshine_4, :CHU_sunshine_4, :SAM_sunshine_4], ignore=true), test_data);
```

```
fit!(mach_test);
```

```
stand_test = MLJ.transform(mach_test, test_data);
```

K-Nearest-Neighbor Classification

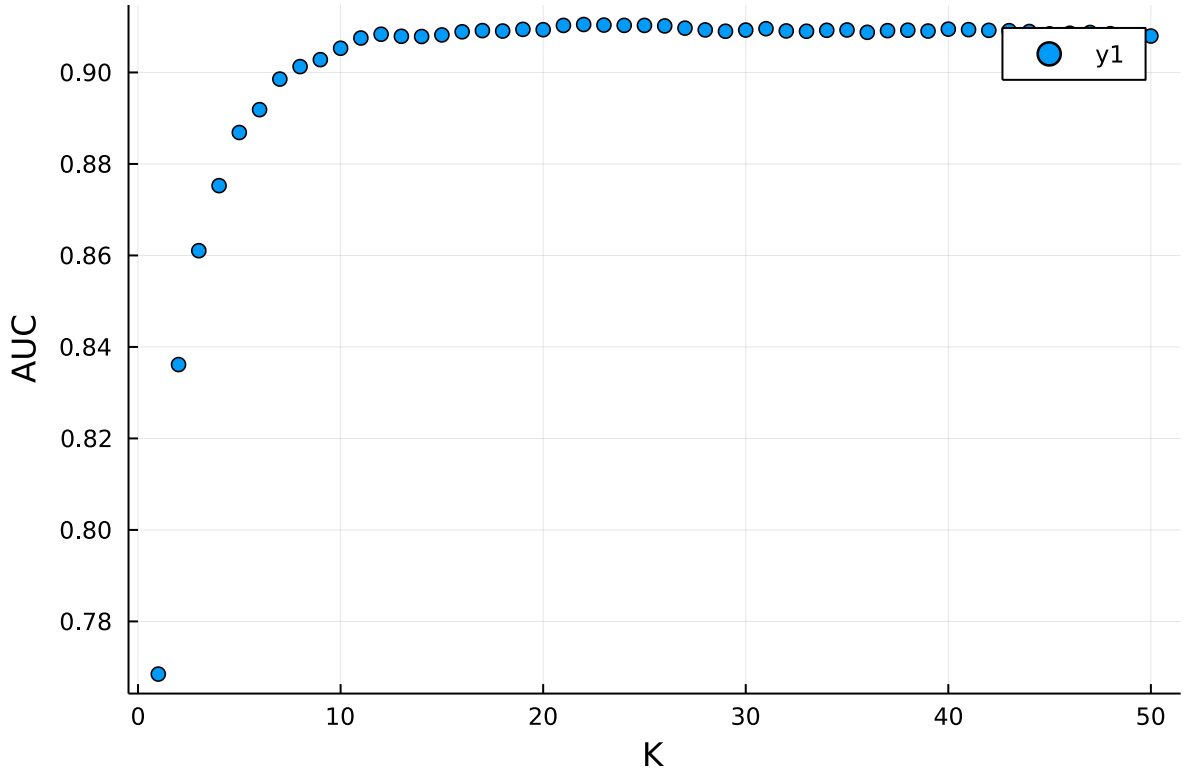
```
using NearestNeighborModels
```

```
Machine{ProbabilisticTunedModel{Grid,…},…} trained 1 time; caches data
  args:
    1:  Source @731 ⏎ `ScientificTypesBase.Table{AbstractVector{ScientificTypesBase.Conti
    2:  Source @279 ⏎ `AbstractVector{ScientificTypesBase.Multiclass{2}}`
```

```
begin
    model = KNNClassifier()
    Random.seed!(10)
    self_tuning_model = TunedModel(model = model,
                                   resampling = CV(nfolds = 5),
                                   tuning = Grid(),
                                   range = range(model, :K, values = 1:50),
                                   measure = auc)
    self_tuning_mach = machine(self_tuning_model,
                               select(stand_train, Not(:precipitation_nextday)),
                               stand_train.precipitation_nextday) |> fit!
end
```

```
rep =
(best_model = KNNClassifier(                , best_history_entry = (model = KNNClassifier
                  K = 22,                                             K = 22,
                  algorithm = :kdtree,                                algorithm
                  metric = Euclidean(0.0),                            metric =
                  leafsize = 10,                                      leafsize
                  reorder = true,                                     reorder =
                  weights = Uniform())                                weights =
```

```
rep = report(self_tuning_mach)
```




```
scatter(reshape(rep.plotting.parameter_values, :),
        rep.plotting.measurements, xlabel = "K", ylabel = "AUC")
```

```
mach = machine(KNNClassifier(K = 22), select(stand_train,
                Not(:precipitation_nextday)), stand_train.precipitation_nextday);
```

```
fit!(mach, verbosity = 2);
```

Let's prepare these results for a submission data set. First we have to load the test set, and apply our machine on it. Then we construct our submission data and download it.

```
pred = predict(mach, stand_test);
```

```
true_pred =
▸ [0.954545, 0.590909, 0.954545, 0.0, 0.863636, 0.0909091, 0.0, 0.954545, 0.272727, 0.0909
```

```
true_pred = pdf.(pred, true)
```

```
submission = DataFrame(id = 1:1200, precipitation_nextday = true_pred);
```

```
CSV.write("../data/project/submission_knn.csv", submission);
```

Tree-Based Methods

We first used a big interval, from 100 to 500. We found that the best model is with n_trees = 254.

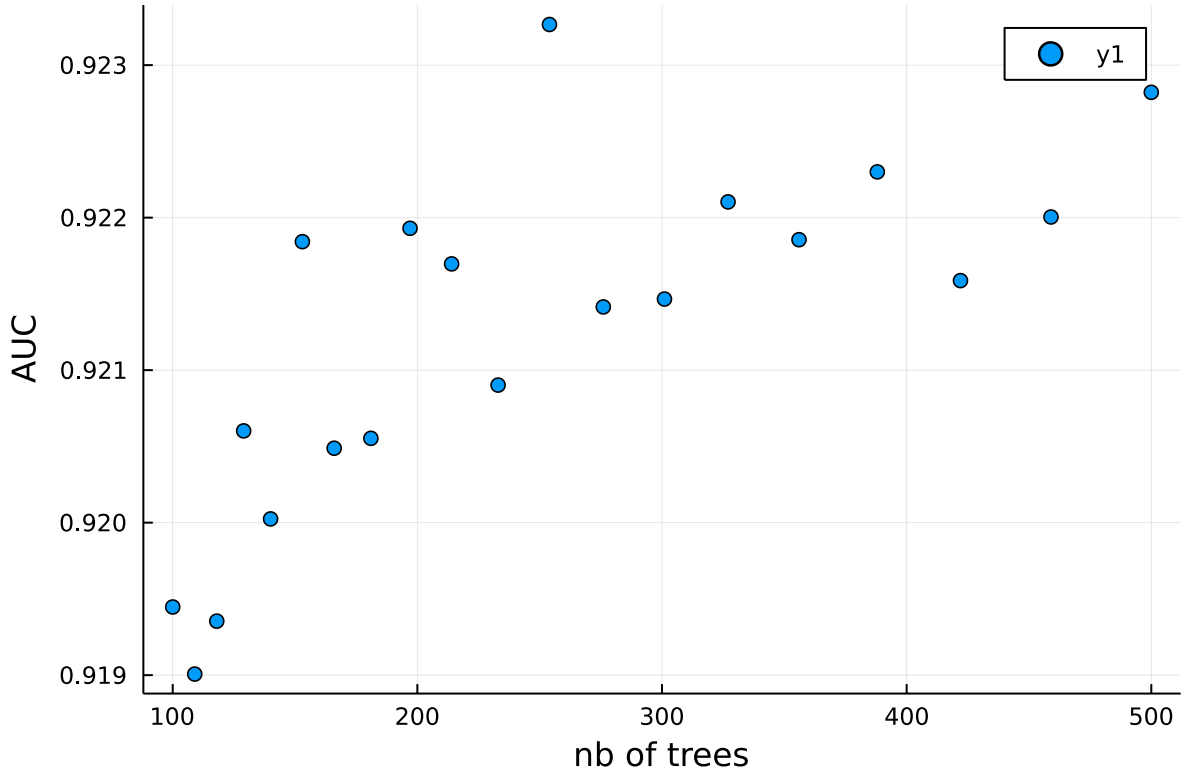
```
Machine{ProbabilisticTunedModel{Grid,…},…} trained 1 time; caches data
  args:
    1:  Source @436 ⏎ `ScientificTypesBase.Table{AbstractVector{ScientificTypesBase.Conti
    2:  Source @635 ⏎ `AbstractVector{ScientificTypesBase.Multiclass{2}}`
```

```
begin
    model1 = RandomForestClassifier()
    Random.seed!(10)
    self_tuning_model1 = TunedModel(model = model1,
                                    resampling = CV(nfolds = 5),
                                    tuning = Grid(goal = 20),
                                    range = range(model1, :n_trees, scale = :log,
                                            lower = 100, upper = 500),
                                    measure = auc)
    self_tuning_mach1 = machine(self_tuning_model1,
                                select(stand_train, Not(:precipitation_nextday)),
                                stand_train.precipitation_nextday) |> fit!
end
```

```
rep1 =
```

```
(best_model = RandomForestClassifier(       , best_history_entry = (model = RandomFores
                  max_depth = -1,                                                max_dep
                  min_samples_leaf = 1,                                          min_sam
                  min_samples_split = 2,                                         min_sam
                  min_purity_increase = 0.0,                                     min_pur
                  n_subfeatures = -1,                                            n_subfe
                  n_trees = 254,                                                 n_trees
                  sampling_fraction = 0.7,                                       samplin
                  pdf_smoothing = 0.0,                                           pdf_smo
                  rng = _GLOBAL_RNG())                                           rng = _
```

```
rep1 = report(self_tuning_mach1)
```




```
scatter(reshape(rep1.plotting.parameter_values, :),
        rep1.plotting.measurements, xlabel = "nb of trees", ylabel = "AUC")
```

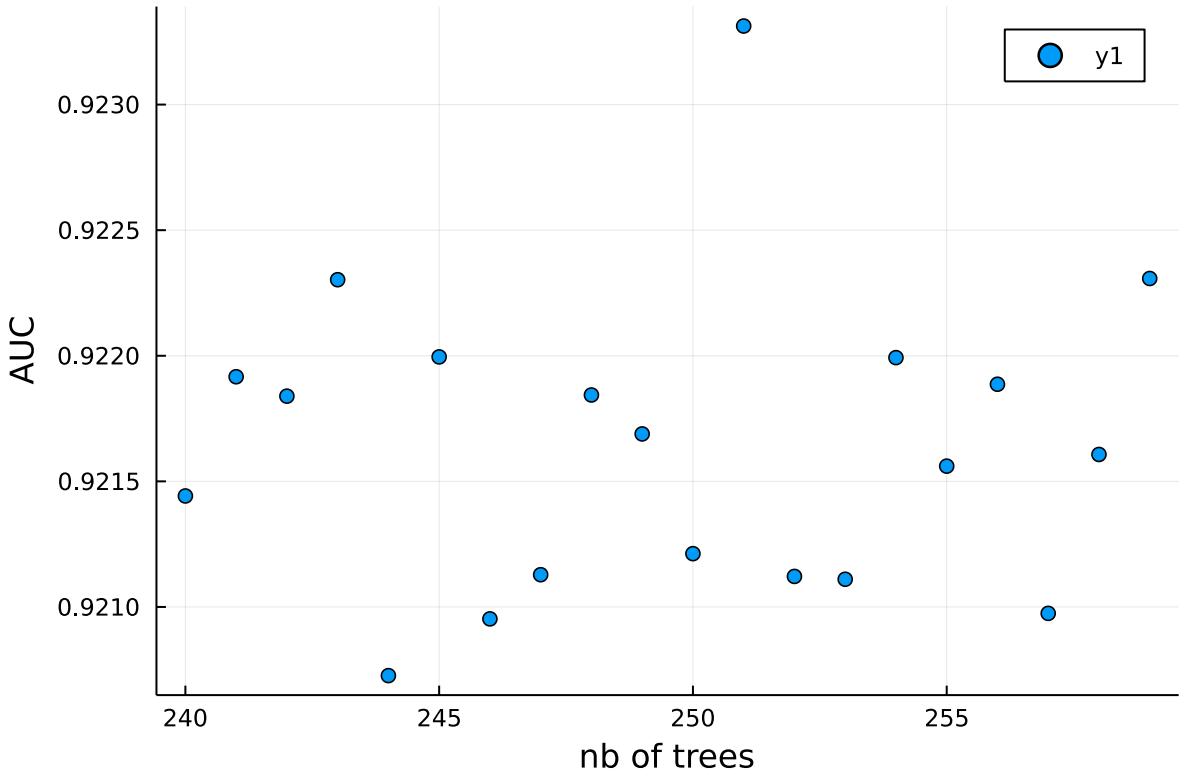
We decided to reduce our interval to find the best model. We obtained n_trees = 251. We continue with this value.

```
Machine{ProbabilisticTunedModel{Grid,…},…} trained 1 time; caches data
  args:
    1:  Source @163 ⏎ `ScientificTypesBase.Table{AbstractVector{ScientificTypesBase.Conti
    2:  Source @464 ⏎ `AbstractVector{ScientificTypesBase.Multiclass{2}}`
```

```
begin
    Random.seed!(10)
    self_tuning_model11 = TunedModel(model = model1,
                                     resampling = CV(nfolds = 5),
                                     tuning = Grid(goal = 20),
                                     range = range(model1, :n_trees, scale = :log,
                                             lower = 240, upper = 259),
                                     measure = auc)
    self_tuning_mach11 = machine(self_tuning_model11,
                                 select(stand_train, Not(:precipitation_nextday)),
                                 stand_train.precipitation_nextday) |> fit!
end
```

```
rep11 =
▸ (best_model = RandomForestClassifier(       , best_history_entry = (model = RandomFores
                  max_depth = -1,                                          max_dep
                  min_samples_leaf = 1,                                    min_sam
                  min_samples_split = 2,                                   min_sam
                  min_purity_increase = 0.0,                               min_pur
                  n_subfeatures = -1,                                      n_subfe
                  n_trees = 251,                                           n_trees
                  sampling_fraction = 0.7,                                 samplin
                  pdf_smoothing = 0.0,                                     pdf_smo
                  rng = _GLOBAL_RNG())                                     rng = _
```

```
rep11 = report(self_tuning_mach11)
```




```
scatter(reshape(rep11.plotting.parameter_values, :),
        rep11.plotting.measurements, xlabel = "nb of trees", ylabel = "AUC")
```

Let's prepare these results for a submission data set.

```
mach1 = machine(RandomForestClassifier(n_trees = 251),
             select(stand_train, Not(:precipitation_nextday)),
                stand_train.precipitation_nextday);
```

```
fit!(mach1, verbosity = 2);
```

```
pred1 = predict(mach1, stand_test);
```

```
true_pred1 =
```

```
▸ [0.944223, 0.685259, 0.768924, 0.199203, 0.864542, 0.322709, 0.131474, 0.924303, 0.37051
```

```
true_pred1 = pdf.(pred1, true)
```

```
submission1 = DataFrame(id = 1:1200, precipitation_nextday = true_pred1);
```

```
CSV.write("../data/project/submission_trees.csv", submission1);
```

Neural Networks

Two-layers

```
mach3 = machine(NeuralNetworkClassifier(builder= MLJFlux.@builder(Chain(Dense(n_in,
50, relu), Dense(50, 50, relu),Dense(50, n_out))), batch_size = 32, epochs = 10,
rng=Random.seed!(10)), select(stand_train, Not(:precipitation_nextday)),
stand_train.precipitation_nextday);
```

```
fit!(mach3, verbosity = 2);
```

Let's prepare these results for a submission data set.

```
pred3 = predict(mach3, stand_test);
```

```
true_pred3 =
```

```
▸ [0.99189, 0.496788, 0.998705, 0.00056508, 0.989927, 0.0113627, 0.0003867, 0.99943, 0.249
```

```
true_pred3 = pdf.(pred3, true)
```

```
submission3 = DataFrame(id = 1:1200, precipitation_nextday = true_pred3);
```

```
CSV.write("../data/project/submission_nn_2_50.csv", submission3);
```

Full-connected three-layer network

```
md"#### Full-connected three-layer network"
```

```
mach2 = machine(NeuralNetworkClassifier(builder =
            MLJFlux.Short(n_hidden = 128, dropout = 0.1, σ = relu),
                                batch_size = 32,
                                epochs = 30, rng=Random.seed!(10)),
             select(stand_train, Not(:precipitation_nextday)),
                 stand_train.precipitation_nextday);
```

```
fit!(mach2, verbosity = 2);
```

Let's prepare these results for a submission data set.

```
pred2 = predict(mach2, stand_test);
```

```
true_pred2 =
```

[0.999983, 0.998141, 0.999953, 2.10731e-5, 1.0, 0.130032, 4.15502e-10, 0.999993, 0.05653

```
true_pred2 = pdf.(pred2, true)
```

```
submission2 = DataFrame(id = 1:1200, precipitation_nextday = true_pred2);
```

```
CSV.write("../data/project/submission_nn_128.csv", submission2);
```