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

Non-Linear Methods

We load the precipitation 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 = CSV.read(joinpath(@__DIR__, "..", "data", "project",
"trainingdata.csv"), DataFrame);
```

```
p = dropmissing(precipitation);
```

```
data_split (generic function with 1 method)
```

```
p1 = coerce!(p, :precipitation_nextday => Binary);
```

```
data1 = data_split(p1);
```

```
function losses(machine, input, response)
    (loglikelihood = -sum(log_loss(predict(machine, input), response)),
     misclassification_rate = mean(predict_mode(machine, input) .!= response),
     accuracy = accuracy(predict_mode(machine, input), response),
     auc = MLJ.auc(predict(machine, input), response)
    )
end;
```

K-Nearest-Neighbor Classification

```
using NearestNeighborModels
```

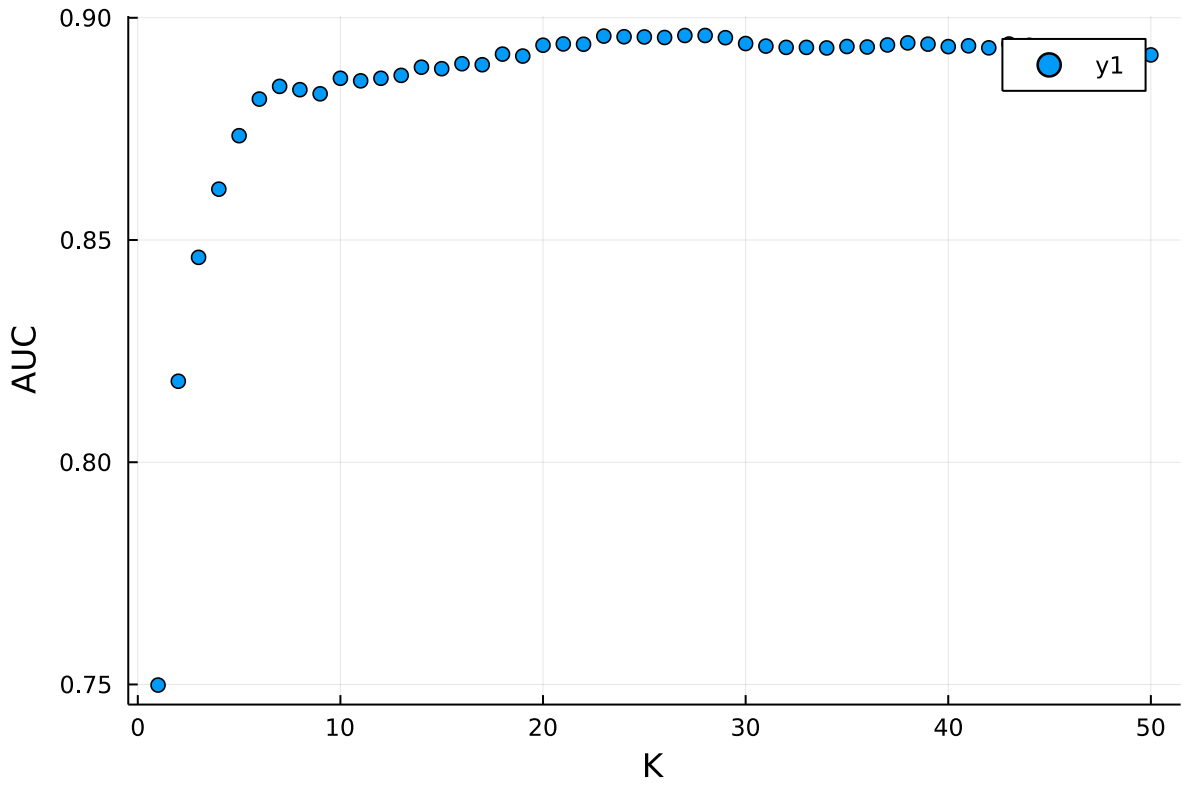
```
Machine{ProbabilisticTunedModel{Grid,…},…} trained 1 time; caches data
  args:
    1:  Source @732 ⏎ `ScientificTypesBase.Table{AbstractVector{ScientificTypesBase.Conti
    2:  Source @135 ⏎ `AbstractVector{ScientificTypesBase.Multiclass{2}}`
```

```
begin
    model = KNNClassifier()
    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(data1.train, Not(:precipitation_nextday)),
                               data1.train.precipitation_nextday) |> fit!
end
```

```
rep =
```

```
(best_model = KNNClassifier(              , best_history_entry = (model = KNNClassifier
                  K = 28,                                           K = 28,
                  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")
```

```
mach2 = machine(KNNClassifier(K = 28), select(data1.train,
                Not(:precipitation_nextday)), data1.train.precipitation_nextday);
```

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

```
lihood = -207.892, misclassification_rate = 0.195755, accuracy = 0.804245, auc = 0.887413)
```

```
losses(mach2, select(data1.test, Not(:precipitation_nextday)),
        data1.test.precipitation_nextday)
```

The test accuracy of KNN is approximately 80%, and the AUC is 88.7%.

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.

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

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

```
true_pred2 =
▸ [0.964286, 0.857143, 0.892857, 0.178571, 0.892857, 0.392857, 0.0, 1.0, 0.107143, 0.07142
```

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

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

```
CSV.write("../data/project/submission_knn2.csv", submission2);
```

Tree-Based Methods

```
mach = machine(RandomForestClassifier(n_trees = 500),
               select(data1.train, Not(:precipitation_nextday)),
                   data1.train.precipitation_nextday);
```

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

```
lihood = -166.857, misclassification_rate = 0.186321, accuracy = 0.813679, auc = 0.910086)
```

```
losses(mach, select(data1.test, Not(:precipitation_nextday)),
        data1.test.precipitation_nextday)
```

The test accuracy of a random forest with 500 trees is approximately 81%, and the AUC is 91%.

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

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

```
true_pred =
▸ [0.944, 0.73, 0.84, 0.224, 0.836, 0.364, 0.12, 0.898, 0.292, 0.16, 0.36, 0.02, 0.014, 0.0
```

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

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

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

Neural Networks

```
mach1 = machine(NeuralNetworkClassifier(builder = MLJFlux.Short(n_hidden = 128,
                                                                dropout = 0.1,
                                                                σ = relu),
                                        batch_size = 32,
                                        epochs = 30),
                select(data1.train, Not(:precipitation_nextday)),
                       data1.train.precipitation_nextday);
```

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

```
elihood = -3536.7, misclassification_rate = 0.233491, accuracy = 0.766509, auc = 0.807022)
```

```
losses(mach1, select(data1.test, Not(:precipitation_nextday)),
        data1.test.precipitation_nextday)
```

The test accuracy of a neural network is approximately 77%, and the AUC is 80.7%.

```
pred1 =
```

```
▸ MLJBase.UnivariateFiniteVector{ScientificTypesBase.Multiclass{2}, Bool, UInt32, Float64
```

```
pred1 = predict(mach1, precipitation_test)
```

```
true_pred1 =
```

```
▸ [1.0, 1.0, 1.0, 3.86834e-123, 1.0, 1.0, 3.89698e-219, 1.0, 9.93578e-201, 0.0, 2.17075e-1
```

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

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

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