```
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);
```

Tree-Based Methods

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

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

```
0.8136792452830188
```

```
mean(predict_mode(mach, select(data1.test, Not(:precipitation_nextday))) .==
data1.test.precipitation_nextday)
```

The test accuracy of a random forest with 500 trees is approximately 81%, better than with the linear methods (74% and ...).

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.

```
md" 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);
```

```
pred =
```

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

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

```
true_pred =
```

```
[0.96, 0.716, 0.794, 0.268, 0.776, 0.364, 0.118, 0.94, 0.212, 0.172, 0.356, 0.016, 0.018,
```

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

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

```
"../data/project/submission_tree.csv"
```

```
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);
```

```
0.7735849056603774
```

```
mean(predict_mode(mach1, select(data1.test, Not(:precipitation_nextday))) .==
data1.test.precipitation_nextday)
```

```
pred1 =
```

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

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

```
true_pred1 =
```

```
[1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 1.0,
```

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

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

```
"../data/project/submission_neural.csv"
```

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