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

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);
```

First we have to prepare our data set by dropping the missing values and split the datas into a train and a test set.

```
p =
```

	ABO_radiation_1	ABO_delta_pressure_1	ABO_air_temp_1	ABO_sunshine_1	ABO_win
1	-0.166667	-1.2	-5.68333	0.0	2.08333
2	0.333333	0.2	5.16667	0.0	1.43333
3	16.5	-0.3	6.16667	33.0	0.983333
4	5.33333	-0.6	9.01667	0.0	7.6
5	9.83333	0.3	7.38333	0.0	7.98333
6	25.5	-1.11022e-16	4.36667	48.0	0.483333
7	0.0	-1.0	-1.15	0.0	1.85
8	12.3333	0.8	13.45	20.0	0.516667
9	0.166667	-1.3	4.3	0.0	0.066666
10	0.333333	0.3	2.9	0.0	9.35
⋮	more				
1699	17.3333	0.1	14.9333	28.0	4.48333

```
p = dropmissing!(precipitation)
```

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

```
function data_split(data;
                            shuffle = false,
                            idx_train = 1:1275,
                            idx_test = 1276:1699)
             idxs = if shuffle
                       randperm(size(data, 1))
                   else
                        1:size(data, 1)
                   end
            (train = data[idxs[idx_train], :],
               test = data[idxs[idx_test], :])
end
```

```
p1 = coerce!(p, :precipitation_nextday => Binary); # with this we tell the computer
to interpret the data in column precipitation_nextday as multi-class data.
```

```
data1 =
```

(train =

	ABO_radiation_1	ABO_delta_pressure_1	ABO_air_temp_1	ABO_sunshine_1
1	-0.166667	-1.2	-5.68333	0.0
2	0.333333	0.2	5.16667	0.0
3	16.5	-0.3	6.16667	33.0
4	5.33333	-0.6	9.01667	0.0
5	9.83333	0.3	7.38333	0.0
6	25.5	-1.11022e-16	4.36667	48.0
7	0.0	-1.0	-1.15	0.0
8	12.3333	0.8	13.45	20.0
9	0.166667	-1.3	4.3	0.0
10	0.333333	0.3	2.9	0.0
⋯ more				
1275	0.333333	-0.2	8.78333	0.0

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

Multiple Logistic Regression

Now we define a supervised learning machine.

```
mach = machine(LogisticClassifier(penalty = :none),
               select(data1.train, Not(:precipitation_nextday)),
               data1.train.precipitation_nextday);
```

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

	Ground Truth	
Predicted	false	true
false	710	0
true	0	565

```
confusion_matrix(predict_mode(mach, select(data1.train,
Not(:precipitation_nextday))),
                 data1.train.precipitation_nextday)
```

With our simple features, logistic regression can classify the training data correctly. Let us see how well this works for test data.

	Ground Truth	
Predicted	false	true
false	180	40
true	69	135

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

Let us evaluate the fit in terms of commonly used losses for binary classification.

```
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;
```

```
lihood = -3464.55, misclassification_rate = 0.257075, accuracy = 0.742925, auc = 0.797338)
```

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

The test accuracy of linear classification is approximately 74%, and the AUC is 79.7%.

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);
```

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

```
true_pred =
▸ [0.0, 1.0, 0.0651944, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0
```

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

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

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

Multiple Logistic Ridge Regression

```
mach1 = machine(LogisticClassifier(penalty = :l2, lambda = 2e-2),
                select(data1.train, Not(:precipitation_nextday)),
                data1.train.precipitation_nextday);
```

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

	Ground Truth	
Predicted	false	true
false	708	3
true	2	562

```
confusion_matrix(predict_mode(mach1, select(data1.train,
Not(:precipitation_nextday))),
                 data1.train.precipitation_nextday)
```

	Ground Truth	
Predicted	false	true
false	182	44
true	67	131

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

```
elihood = -2439.7, misclassification_rate = 0.261792, accuracy = 0.738208, auc = 0.822031)
```

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

The test accuracy of linear Ridge classification is approximately 74%, and the AUC is 82.2%.

Let's prepare these results for a submission data set, same steps as the Multiple Logistic Regression.

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

```
true_pred1 =
▸ [0.000218071, 1.0, 0.960695, 1.0, 1.0, 1.40411e-11, 0.0, 1.0, 0.999985, 0.0, 4.86534e-12,
```

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

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

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

Regularization

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

```
function tune_model(model, data)
    tuned_model = TunedModel(model = model,
                             resampling = CV(nfolds = 5),
                             tuning = Grid(goal = 20),
                             range = range(model, :lambda,
                                           scale = :log,
                                           lower = 1e-2, upper = 1),
                             measure = auc)
    machine(tuned_model, select(data1.train, Not(:precipitation_nextday)),
data1.train.precipitation_nextday) |> fit!
end
```

```
▸ (best_model = LogisticClassifier(            , best_fitted_params = (classes = Categori
                  lambda = 0.78476,
                  gamma = 0.0,
                  penalty = :l2,
                  fit_intercept = true,
                  penalize_intercept = false,
                  solver = nothing)
```

```
fitted_params(tune_model(LogisticClassifier(), data1))
```

```
mach2 = machine(LogisticClassifier(penalty = :l2, lambda = 0.78476),
            select(data1.train, Not(:precipitation_nextday)),
            data1.train.precipitation_nextday);
```

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

```
lihood = -2127.88, misclassification_rate = 0.261792, accuracy = 0.738208, auc = 0.832312)
```

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

The test accuracy of linear Ridge classification regularized is approximately 74%, and the AUC is 83.2%.

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

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

```
true_pred2 =
```

```
[0.00311153, 1.0, 0.992206, 1.0, 1.0, 2.46293e-11, 2.50565e-13, 1.0, 0.994671, 0.0, 4.09
```

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

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

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