```
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.

```
md" # 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);
```

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

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

	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.

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

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

```
0.7429245283018868
```

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

The test accuracy of linear classification is approximately 74%.

	Ground Truth	
Predicted	false	true
false	180	40
true	69	135

```julia
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.

```julia
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;
```

```
▶(loglikelihood = -9.35433, misclassification_rate = 0.0, accuracy = 1.0, auc = 1.0)
```

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

```
▶(loglikelihood = -3464.55, misclassification_rate = 0.257075, accuracy = 0.742925, auc
```

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

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.

```julia
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."
```

```julia
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)
```

```
_pred =
.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, 1.0
```

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

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

```
"../data/project/submission_regression.csv"
```

```
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!;
```

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

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

	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)
```

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

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

```
0.7382075471698113
```

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

The test accuracy of linear Ridge classification is approximately 74%.

	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)
```

We see that the test misclassification rate with regularization is lower than in our original fit without regularization. The misclassification rate on the training set is higher. This indicates that unregularized logistic regression is too flexible for our data set.

```
(loglikelihood = -33.0821, misclassification_rate = 0.00392157, accuracy = 0.996078, au
```

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

```
(loglikelihood = -2439.7, misclassification_rate = 0.261792, accuracy = 0.738208, auc =
```

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

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

```
pred1 =
MLJBase.UnivariateFiniteVector{ScientificTypesBase.Multiclass{2}, Bool, UInt32, Float64
```

```
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);
```

```
"../data/project/submission_ridge_regression.csv"
```

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