Report on ML Classification on a weather dataset

by Giuseppe Insana, January 2022

The dataset

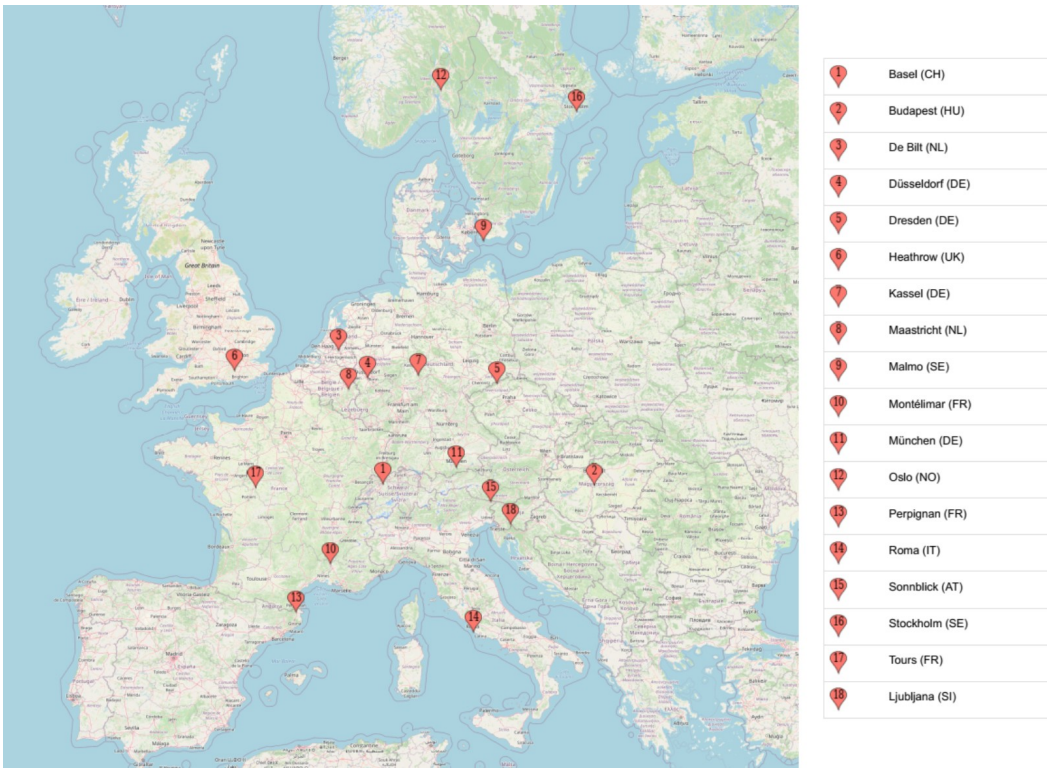
I chose to work on weather data, chiefly because I never worked on this kind of data before (having mostly dealt with biological and linguistic data in the past).

The weather data analysed in this report has been created by Huber Florian from ECA&D data (see section on Data origin at the end for references).

It contains daily weather observations from 18 different European weather stations through the years 2000 to 2010.

The description of the data set says that the minimal set of variables 'mean temperature', 'max temperature' and 'min temperature' are available for all locations. An additional number of measured variables ('cloud_cover', 'wind_speed', 'wind_gust', 'humidity', 'pressure', 'global_radiation', 'precipitation', 'sunshine') are provided, but not for all the locations.

The following map shows the locations which are included in the dataset:



Objective of the analysis

We will try to predict tomorrow's weather for one location based on today's weather measures across all locations. We will focus chiefly on trying to achieve prediction accuracy (in particular good precision and good recall, as condensed in the F1 score) but we will also try where possible to gain insights from the models, to help us in understanding the major factors that guide the classification.

In particular we will try to predict whether tomorrow is going to be sunny in the city of Basel.

Data exploration

A comprehensive exploratory data analysis has been previously conducted on the model and presented in a previous report (attached as a separate document, see file **addonEDAcourseproject_report_Giuseppe_Insana.pdf**). We will here briefly summarise the findings of that report to give a description of the data we will process and to show the rationale behind the actions we will take.

We have analysed the types and amount of the available data, checking ranges and distributions of all observations.

Data attributes

The original data is loaded into a pandas dataframe which has 3654 rows (one per each day) and 165 columns:

- DATE, with integer values from 20000101 to 20100101, corresponding to interval from Jan 1st 2000 to Jan 1st 2010
- MONTH, integer 1 to 12
- and another 163 columns for the weather measurements at the different locations.

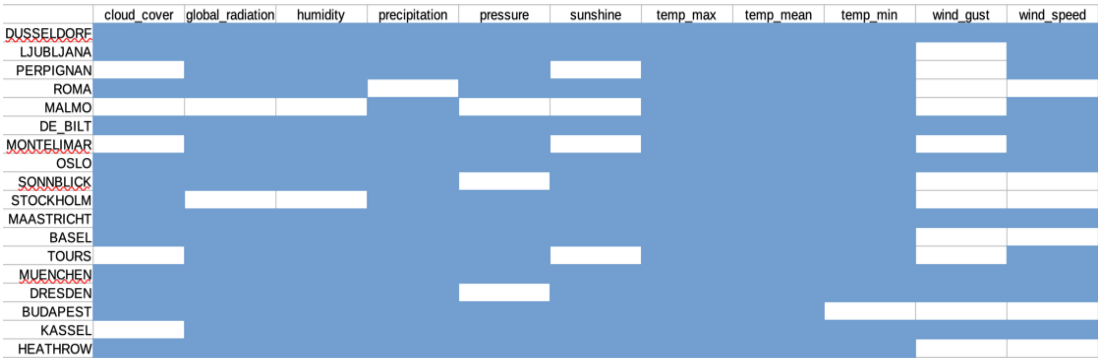
The measurements are labelled as LOCATION_*measure* (e.g. BASEL_cloud_cover, BASEL_pressure, OSLO_precipitation...) with the measured variables being: **cloud_cover, global_radiation, humidity, precipitation, pressure, sunshine, temp_max, temp_mean, temp_min, wind_gust, wind_speed**

All the measurements are loaded as floating point numbers, with the exception of cloud_cover, loaded as integer.

The **physical units** for the variables are described as follows:

> **cloud_cover** in oktas; **wind_speed** and **wind_gust** in m/s; **humidity** in fraction of 100%; **pressure** in 1000 hPa, **global_radiation** in 100 W/m^2; **precipitation** in 10 mm; **sunshine** in 1 Hours; **mean max** and **min temperature** in Celsius degrees.

The following table shows which measures are available for which location:



This can obviously present problems of **unbalanced data**, in particular when extending the model to the prediction for different cities. We will address this issue in the discussion.

Null and Out-Of-Range values

The dataset does not contain missing values per se but there are several out-of-range values that can be considered as Null/Missing/Invalid observations:

cloud_cover measures should vary from 0 (sky completely clear) to 8 (sky completely clouded) oktas. But the data contains two data points (both for Stockholm, 20080724 and 20090625) with a value of -99 and one (again Stockholm, 20031108) with a value of 9

pressure: the data contains three entries (again for Stockholm, 20071008, 20000124, 20070603) with value of -0.099 and one entry (Tours, 20081230) with value of 0.0003. These out of range values can again be considered invalid and a decision should be taken for them akin to those mentioned before for cloud_cover.

sunshine: the data contains 29 negative values for hours of sunshine (again for the Stockholm location) which should be treated as invalid/null and dealt appropriately (as mentioned for cloud_cover and pressure). For the location of Oslo there are 24 measurements with more than 18 hours of sunshine, with 20 of them being 24h. While the northern latitude make very long daylight possible, this is for almost 18 hours in midsummer, while these huge values are from Nov-Dec 2006. We must hence treat these as wrong invalid data as well.

Distribution of values

After imputing the out of range values (shown below in the section on Data cleaning and feature engineering), the range, mean and standard deviation for all measures across all locations are the following:

measure	mean	std	range	
cloud_cover:	5.14	2.33	0.00 .. 8.00	*(okta)*
global_radiation:	1.37	0.95	0.01 .. 4.42	*(i.e. 1 to 442 W/m2)*
humidity:	0.75	0.14	0.10 .. 1.00	*(i.e. 1% to 100%)*
precipitation:	0.23	0.58	0.00 .. 16.04	*(i.e. 0 to 160.4 mm)*
pressure:	1.02	0.01	0.96 .. 1.05	*(i.e. 959 to 1016 hPa)*
sunshine:	5	4.41	0.00 .. 17.80	*(hours)*
temp_max:	14.5	9.58	-24.70 .. 41.10	*(°C)*
temp_mean:	10.39	8.41	-26.60 .. 33.10	*(°C)*
temp_min:	6.33	7.58	-30.30 .. 26.30	*(°C)*
wind_gust:	10.06	3.88	1.50 .. 41.00	*(m/s)*
wind_speed:	3.33	1.89	0.00 .. 16.30	*(m/s)*

The dataset was thoroughly explored visually by way of plots, to see the actual distribution of values and to gather insights, plotting the measured features as a whole or grouped spatially or temporally.

Major findings:

- strong dependence of weather measures by month and by location (obviously) for both ranges, mean and variance; for example pressure has smaller variance in summer months compared to winter months;
- the seasonal component appears to be responsible for multimodality in certain features;
- there are also year to year variations, with for example some years being on average warmer or colder; no overall trend was observed but this is probably due to the scale of the dataset (only ten years period);

- precipitation and wind_speed are the most skewed measures (more than 0.75 skew value) across all locations and for some locations humidity and cloud_cover as well. The most strongly skewed features are:

DRESDEN_precipitation	13.077
PERPIGNAN_precipitation	10.781
MONTELIMAR_precipitation	7.479
BUDAPEST_precipitation	5.662
MALMO_precipitation	5.337
MUENCHEN_precipitation	5.206
BASEL_precipitation	4.529
STOCKHOLM_precipitation	4.481
TOURS_precipitation	4.233
LJUBLJANA_precipitation	3.828

Correlations

The correlation between features was analysed, both for measures in a single location or across different locations.

The major insights were:

- related variables have (as expected) very high correlation:
  - temp_mean with temp_min and temp_max
  - wind speed and wind gust
- global radiation has high positive correlation with sunshine
- global radiation and sunshine correlate negatively with humidity
- precipitation appears to have extreme values concentrated where pressure has average values
- locations nearby have measures more highly correlated compared to locations geographically distant

Data cleaning and feature engineering

Out of range values

To clean the dataset we first dealt with the **out-of-range values** identified for the cloud_cover, pressure and sunshine measures.

The number of these invalid values appear in 1.64% of the total rows but as they only affect one location at a time they constitute only 0.01% of the total values. In the previous report we **recommended to not drop** the whole days' measurements but **instead to impute** the invalid values.

To do so we wrote code which gathers the values for the involved measure and location on the 5 days before and 5 days after the date of the out-of-range value. These values are then averaged and the average is used to impute the invalid value. When the invalid values appeared in succession the strategy is modified to use the window of 10 days (5 before and 5 after) but for the year before.

For example, the code identifies the above mentioned out-of-range values for pressure:

idx	DATE	STOCKHOLM_pressure	TOURS_pressure
23	20000124	-0.0990	1.0234
2710	20070603	-0.0990	1.0205
2837	20071008	-0.0990	1.0242
3286	20081230	1.0328	0.0003

and proceeded to impute them as following:

```
** Imputing 3 value(s) for column STOCKHOLM_pressure
idx23: from -0.099 to 1.00449 using mean data from range 18-29
idx2710: from -0.099 to 1.02079 using mean data from range 2705-2716
idx2837: from -0.099 to 1.0203099999999998 using mean data from range 2832-2843

** Imputing 1 value(s) for column TOURS_pressure
idx3286: from 0.0003 to 1.02485 using mean data from range 3281-3292
```

Outliers

The **main outliers** identified in the EDA report were:

- the very low values for humidity, with majority of outliers from Sonnblick and (in much lower proportion) Perpignan
- the precipitation extremes
- the highest recorded wind_speed measures (which are almost all from Perpignan location)

These could represent extreme weather conditions, which may or may not hinder the prediction abilities of the ML models.

Similarly to what said for out-of-range values, the outliers account for a negligible portion of the dataset but non-negligible number of rows. Thus it would be best to impute the single values (using the same strategy as above, averaging over a window of days) rather than excluding the whole rows.

To see the effect of the outliers on the machine learning models, we will be training and comparing the classification models both before and after imputing the outliers.

Feature engineering

The original **dataset does not contain categorical data** which would need to be converted (e.g. by one-hot encoding).

We will be **dropping the** DATE **feature** from the training data and we will check **how important** the MONTH information is for weather prediction.

In the models we are going to train, we will be trying both **transformations** (transforming the skewed variables outlined above) and **scaling** of the features.

We will also be trying the addition of **interaction terms** using PolynomialFeatures.

The **biggest decision** to take is about the features to use as input for the models:

1. whether we want to keep all the features and all the locations
2. whether we'd prefer to have a balanced dataset which holds only a subset of features and a subset of locations so that we would have the same features for each location

We will be training models using both of these strategies so we can understand the impact of such a decision.

We believe that for the prediction of a certain variable at a certain location, knowledge of the weather variables at other locations, even if incomplete (some locations missing some features) could still be important, as long as the model will be able to deal with it.

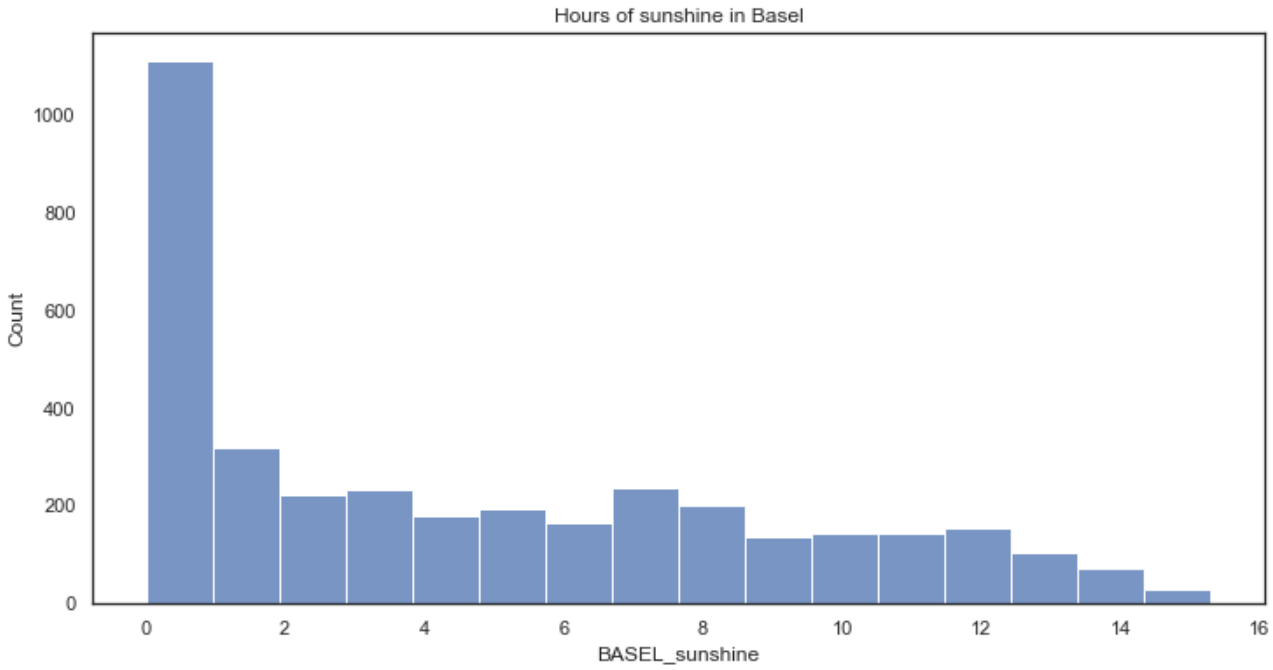
The target class

As mentioned at the beginning, we will try to predict whether tomorrow is going to be sunny in the city of Basel, based on today's weather data (all or subset of it).

The target measure we start with is BASEL_sunshine (which holds number of hours of sunshine), which we transform to a binary class based on the following logic: if **number of sunshine hours** is **higher or equal than 1**, we apply the label "`Sunny`", otherwise we set the label as "`Cloudy`".

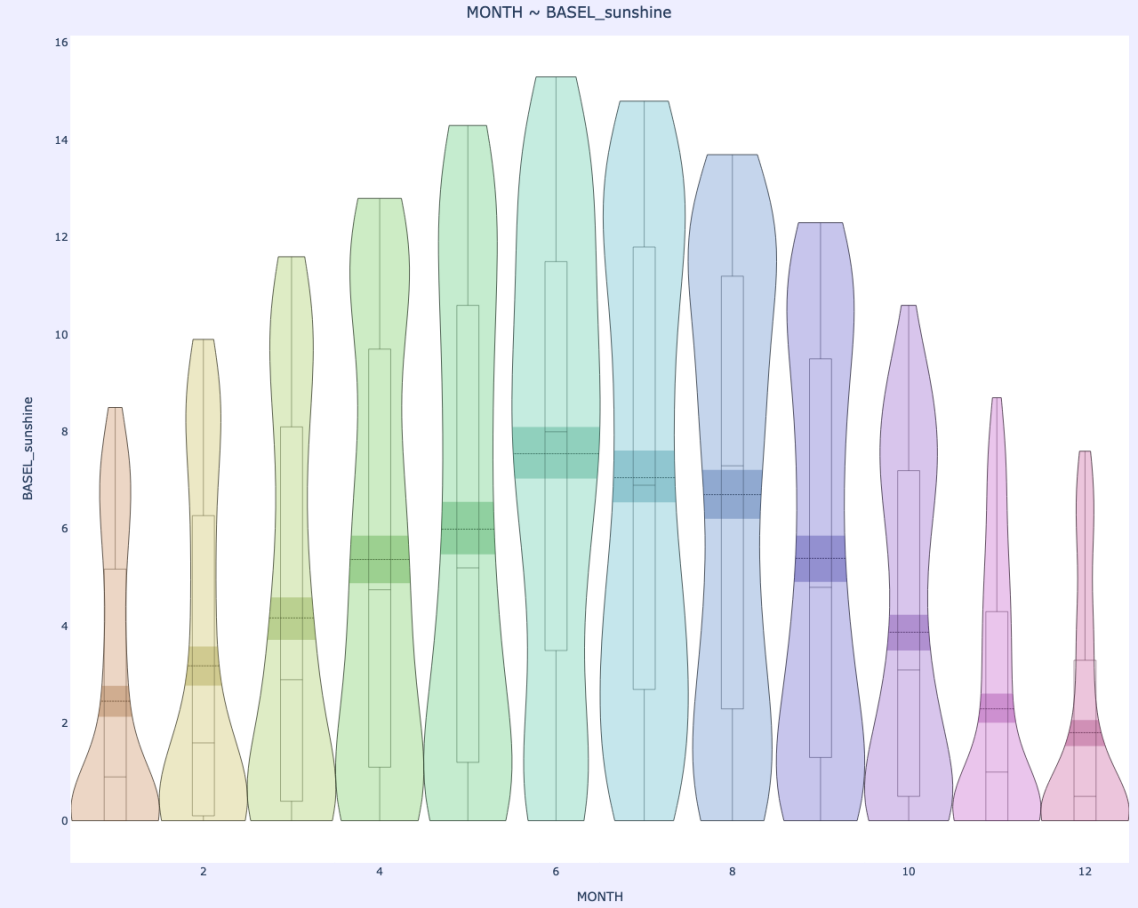
We will show now the rationale behind this choice.

First of all the histogram of sunshine hours shows a great peak close to 0:



In fact there are 641 days without any sunshine in Basel in the 10 years of data (17.5% of the total days) and 1112 days with less than one hour of sunshine (31%).

The sunshine measure has a strong seasonal component, with less hours in average for winter months, as shown in the following plot, where the sunshine hours are plotted by month:



While the mean for the winter months is 2.4h, the median is actually 0.95h.

Choosing **1h** as the threshold creates a balanced classification target for the winter months (about 50% of winter days labelled `Cloudy` and 50% labelled `Sunny`) and a partially unbalanced class when we consider the whole year: 69% labelled `Sunny` and 31% labelled `Cloudy`.

As we are dealing with an unbalanced class we will be using **Stratified** split strategy, both for the initial test/train split and for the cross validation folds. And as the unbalance is connected with the seasonal variance, we will be careful to balance this as well in order to have samples equally distributed across all months (as explained in the next section).

Models training

Procedure

Each model tested (a part from the baseline models described below) will be setup using a *scikit-learn* **Pipeline**, grouping together the list of steps to be applied to the data, with the final one being the estimator and the preceding ones being transforms.

One of the main advantages of this setup is to be able to assemble a series of steps that can be cross-validated together over a series of different parameters.

It also lowers the risk of making mistakes with transformations and leaking test information into the train set.

At the beginning of the analysis we set aside a sample of 20% of the entire dataset to be used as **test** for evaluation. The remaining 80% is the data on which we **train** each model on.

The test/train split has been done with the "stratify" strategy due to the unbalance in the class as mentioned in the previous section.

In particular, as the winter months would have a different balance of the Sunny and Cloudy label, to ensure that we keep the correct proportion not only of the target variable but also balanced across all months, we performed a stratification along two dimensions: the classification target and also the MONTH variable. This prevents the skewing of the proportion of data from different months in test and train sets.

Using pandas value_counts() function we can verify that the balance among the labels has been kept:

```
Sunny     69.8%
Cloudy    30.2%
```

And we can also confirm that X_test data is now equally spread across all months:

1	8.35%
2	8.19%
3	8.30%
4	8.21%
5	8.21%
6	8.20%
7	8.52%
8	8.48%
9	8.23%
10	8.62%
11	8.35%
12	8.34%

For **cross validation**, we have used *StratifiedKFold* with **5 splits**. This means that the training data is split in 5 subsets and each fold is then used once as a validation while the remaining folds form the training set for that iteration.

For **scoring**, a series of metrics will be presented for each model: accuracy, precision, recall, F1, AUC.

Accuracy is the proportion of correct predictions, both true positive and true negatives. Thus it is computed as:

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

where TP = True Positives, TN = True Negatives, FP = False Positives, FN = False Negatives.

Note that Accuracy could be partly misleading for an imbalanced data set (like the one we are analysing): a model which would always predict Sunny would get an accuracy of 69.8%.

Precision is the proportion of correctly classified positive results over all those that are classified as positive, so it is also referred as *Positive Predictive Value*; it is computed as

$$\text{Precision} = TP / (TP + FP)$$

Recall is the true positive rate, i.e. the proportion of correctly classified positive results over all the real positive ones and can also be referred as *Sensitivity*; it is computed as

$$\text{Recall} = TP / (TP + FN)$$

F1 score is a measure which combines precision and recall as their harmonic mean

$$F_1 = 2 / (\text{Recall}^{-1} + \text{Precision}^{-1})$$

AUC or *ROC AUC* is the Area Under the Receiver Operating Characteristic Curve (the curve plotting true positive rate against false positive rate) computed by sklearn's *roc_auc_score*.

All the above metrics would be equal to 1 only in case of perfect classification (100% of the predictions correct).

We will also present confusion matrices, where the data is plotted divided in four quadrants which correspond to TN, FP, FN and TP. We will be showing both the confusion matrix relative to the training data and the final one for the test data, as this often offers a clear illustration of the overfitting that plagues some models.

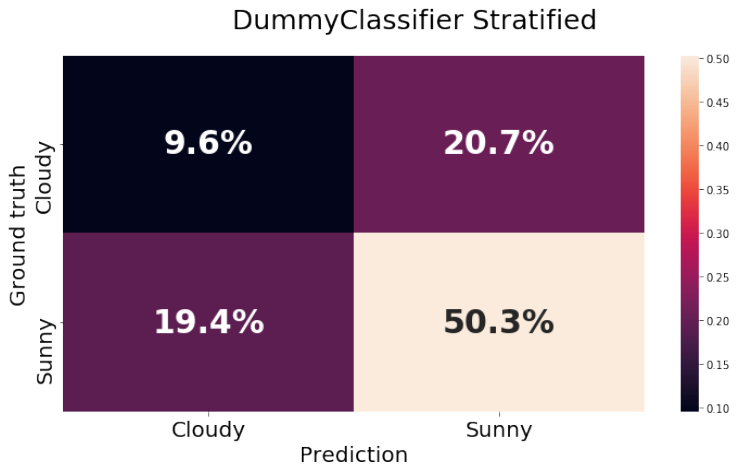
The cross validation procedure ensures a good comparison of the models for the training, and the final scoring on completely unseen test data is best practice to evaluate the predictive ability of the different models.

Baseline

To compare machine learning models it is often useful to have a baseline. The simplest approach is to use a dummy classifier, in our case one with the strategy *"stratified"* which will randomly "predict" a label in the same proportion as that found in the training data.

It obviously performs quite bad:

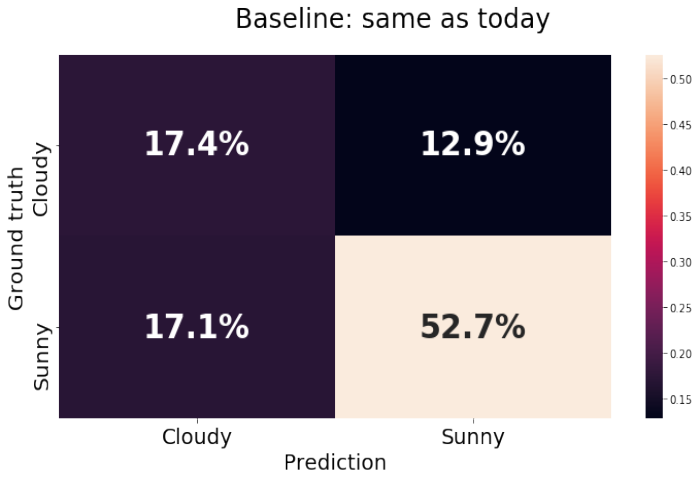
	Accuracy	Precision	Recall	F1	AUC
dummy	0.599	0.709	0.722	0.715	0.519



A better baseline we can devise is to use today's measured observation in order to predict tomorrow's weather. For example, if today the city's sunshine was measured in 4 hours, this baseline model would predict tomorrow's label to be **Sunny**.

It performs much better than the dummy regressor and we can compare the results of our classification models against it:

	Accuracy	Precision	Recall	F1	AUC
same_as_today	0.700	0.804	0.755	0.779	0.665

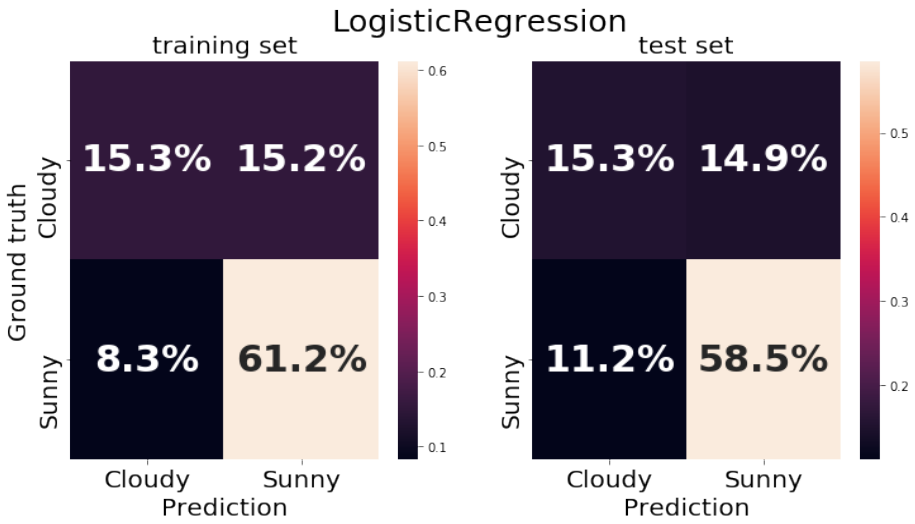


Simple Logistic Regression

We can begin with simple logistic regression, without any particular setup a part from the cross validation explained above.

The results are encouraging. We have a much better predictive power than the baseline models, but we can probably do better.

	Accuracy	Precision	Recall	F1	AUC
LR	0.739	0.797	0.839	0.818	0.673



Adding Column Transformations

We tested adding the transformer *log1p* to specific columns (using sklearn's *ColumnTransformer*), first to all columns for the most skewed measures of precipitation, wind_gust and wind_speed and then only to the columns effectively more skewed (with skew value more than 0.75). This second strategy proved more successful.

It slightly improved the model, with about a 0.01 increase across all metrics:

	Accuracy	Precision	Recall	F1	AUC
CT_LR	0.750	0.803	0.849	0.826	0.685

Adding Scaling

We then added scaling using sklearn's *StandardScaler*. As expected, there was no change in the performance of the basic LR model, but it is important for the other models.

Putting all features on the same scale allows also to better interpret which features have more influence in predicting our target variable. For example we can check which features correspond to the fitted coefficients having a highest absolute value:

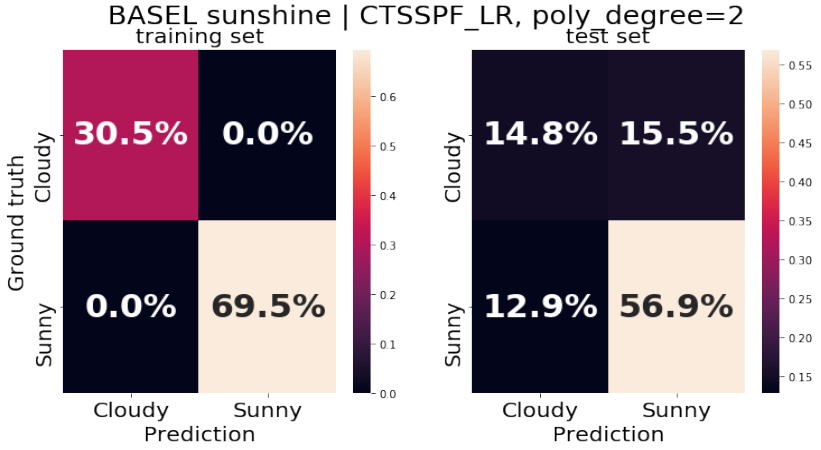
```
      Sum of coefficients: 29.47
Coefficients set to zero: 0/164 (0.00%)
    Highest (absolute) coefficients:
LJUBLJANA_pressure               -1.02
MAASTRICHT_temp_min               0.99
DUSSELDORF_precipitation          0.64
MAASTRICHT_humidity              -0.61
OSLO_pressure                     0.61
OSLO_sunshine                    -0.57
OSLO_temp_min                     0.55
KASSEL_global_radiation           0.54
DUSSELDORF_temp_max              -0.51
```

Adding PolynomialFeatures

The addition of polynomial features of second degree brings the number of features (and hence of parameters to train) from 164 to 13694 (the original measures, the measures squared and all the combinations of feature pairs).

The results show the model overfitting. The model trained on the whole set makes excellent classification on the training set (as can be seen in the perfect confusion matrix on the left) but performs very bad on the test set.

	Accuracy	Precision	Recall	F1	AUC
CTSSPF_LR	0.717	0.786	0.816	0.801	0.652



Adding regularisation: L1 & L2

As shown by the above metrics, the addition of PolynomialFeatures creates drastic overfitting. To cope with that, regularisation techniques need to be added, introducing a cost component to the coefficients.

We used both L1 and L2, with and without PolynomialFeatures of degree two while keeping the previously introduced transforms (*log1p* to skewed features and *StandardScaler*).

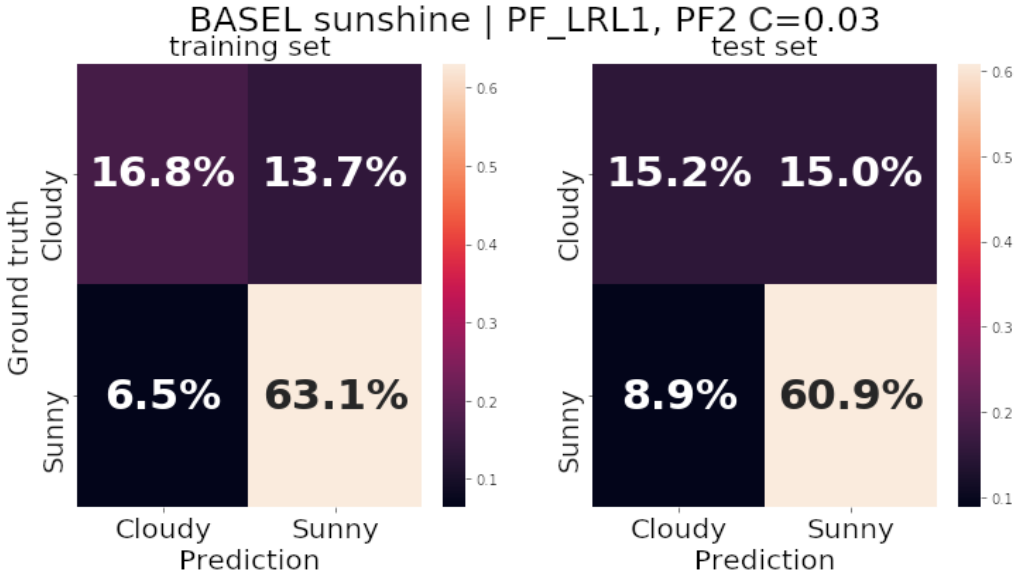
To tune the C hyper-parameter for regularisation, sklearn's *GridSearchCV* was used, with the same cross validation setup explained above.

	Accuracy	Precision	Recall	F1	AUC
LR L1	0.752	0.796	0.867	0.830	0.678
PF_LR L1	0.761	0.802	0.873	0.836	0.687
LR L2	0.761	0.822	0.839	0.830	0.709
PF_LR L2	0.747	0.795	0.859	0.826	0.674

L1 regularisation has the advantage of setting coefficients to zero, so selectively removing the least important features from the model. Only 25 features out of 164 were kept for LR L1 (zeroing 84.76% of the coefficients) and only 184 features out of 13694 for PF_LR L1 (with polynomial features of second degree):

```
L1, C=0.03
Sum of coefficients: 2.06
Coefficients set to zero: 139/164 (84.76%)

PF_LR L1, C=0.03
Sum of coefficients: 5.34
Coefficients set to zero: 13510/13694 (98.66%)
```



We will discuss below the most important features retained.

The confusion matrix also shows that the models with polynomial features suffer more from overfitting. For example PF_LR L1 on training data (without cross validation) would get an F1 score of 0.92.

Combining regularisation techniques: Elastic Net

Elastic net uses both the L2 and the L1 penalties.

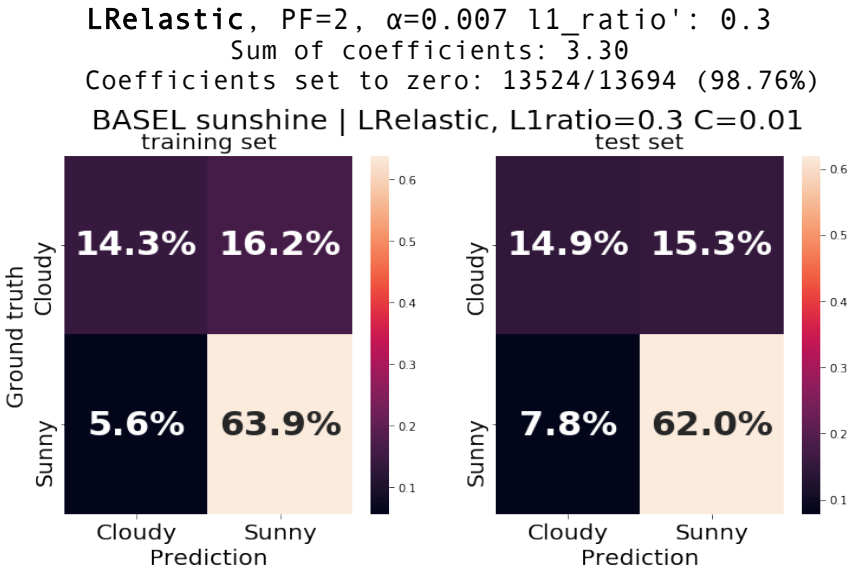
Elastic net is also considered useful when there are multiple features correlated with one another.

It has the drawback of taking much longer to fit, both because of the solver used (*saga*) and because of the new parameter for the grid search (the ratio between L2 and L1 penalties).

Combining the two penalties produces slightly better results:

	Accuracy	Precision	Recall	F1	AUC
LRelastic	0.769	0.802	0.888	0.843	0.691

The best hyper-parameters for ElasticNet were C of 0.007 and an L1 ratio of 0.3. It kept 170 features with a non-zero coefficient:



Feature importance and insights

The top features identified by the LogisticRegression models trained so far are more or less the same, with minor variations in the order. The top 20 were the following:

DUSSELDORF_pressure	-0.13000
ROMA_sunshine	0.11935
MUENCHEN_temp_min	0.11187
TOURS_precipitation	0.11020
OSLO_wind_speed	0.09572
MONTELIMAR_temp_max	0.08974
TOURS_temp_max	0.08896
DE_BILT_precipitation	-0.08787
DRESDEN_wind_speed	-0.06409
MAASTRICHT_global_radiation	0.06190
HEATHROW_pressure	-0.05900
TOURS_pressure	-0.05405
BUDAPEST_cloud_cover	-0.05381
ROMA_pressure	0.04891
DUSSELDORF_temp_mean	0.04518
ROMA_humidity	0.04447
MONTELIMAR_temp_min	0.04398
OSLO_pressure	0.03948
LJUBLJANA_temp_min	-0.03808
DUSSELDORF_sunshine	0.03636

Interestingly BASEL_sunshine is not among the retained features, revealing even more (like above seen with the baseline model) how today's sunshine hours is not the best indication of tomorrow's sunshine.

In fact there are no BASEL measures in this list, revealing how the weather measures at other locations have a stronger predictive power than the one at the location.

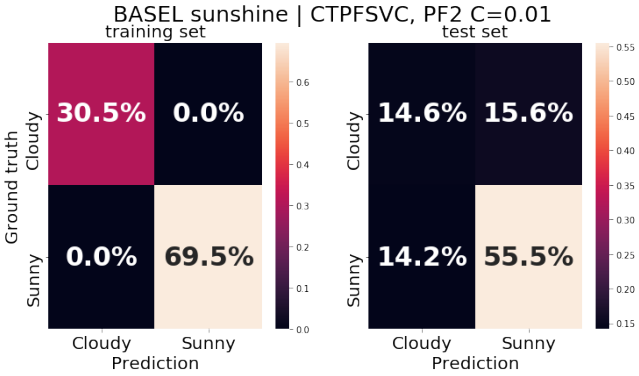
Also to note is the fact that the MONTH feature does not appear. A naive hypothesis was that MONTH would be a good predictor for the weather, due to the seasonal component previously noted.

Support Vector Machine Classifiers

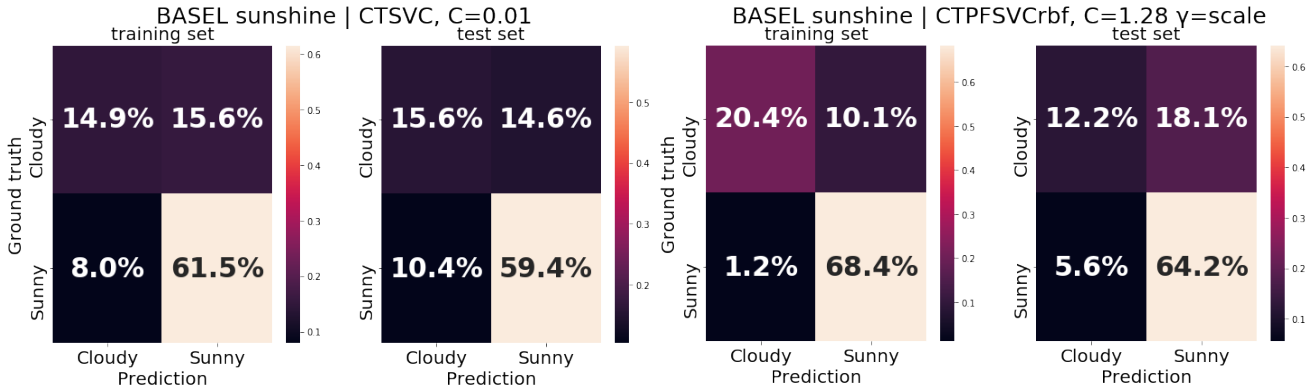
A first alternative to LogisticRegression is to use Support Vector Classifiers. We trained both linear model (sklearn's *LinearSVC*) and *SVC* with rbf kernels, with and without PolynomialFeatures. The results are comparable or better than the best ones achieved by tuned LogisticRegression models:

	Accuracy	Precision	Recall	F1	AUC
SVC	0.750	0.802	0.851	0.826	0.683
PF SVC	0.702	0.781	0.796	0.788	0.640
SVC rbf	0.767	0.800	0.888	0.842	0.689
PF SVC rbf	0.763	0.780	0.920	0.844	0.661

The interaction terms from PolynomialFeatures apparently causes high overfitting which is then penalised during the final scoring on test data:



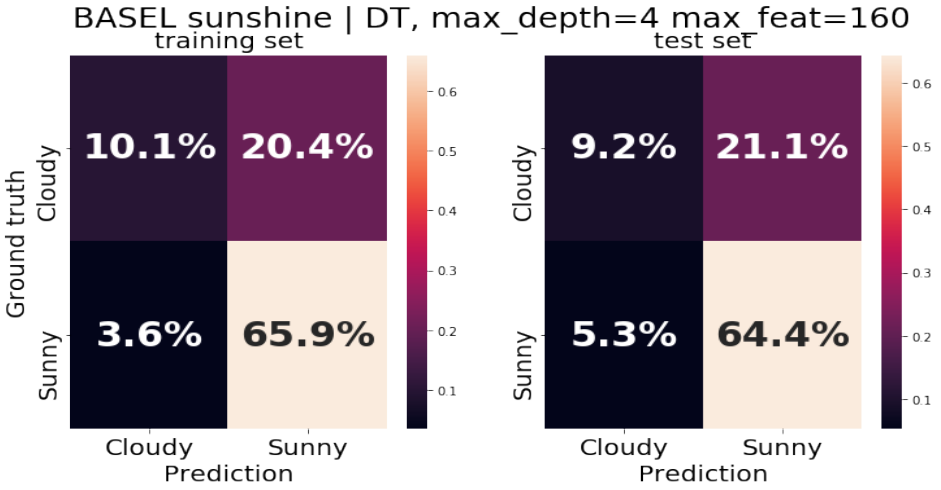
The use of kernels also appears to cause overfitting. It improved Accuracy and Recall at the cost of Precision, tending to classify too many results as Sunny and hence having more False Positives:



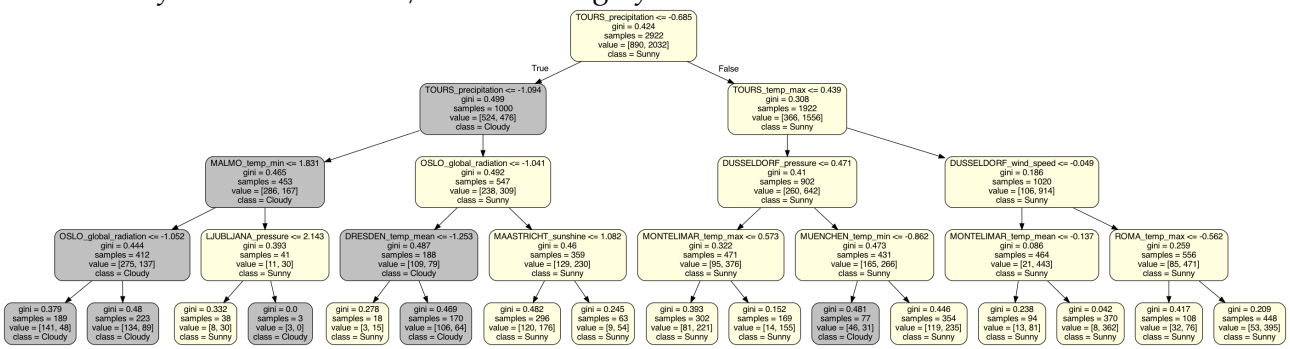
Decision Trees

A completely different approach is to train decision trees. We used sklearn's *DecisionTreeClassifier* with the same setup as for the other models (Pipeline, trained with stratified cross validation, final comparison against holdout test data). Like previously observed for SVC with kernels, the Decision tree ends up with higher recall but lower precision compared with LogisticRegression models. The F1 score is higher but we are getting too many False Positives:

	Accuracy	Precision	Recall	F1	AUC
DT	0.736	0.754	0.924	0.830	0.613



The following is a visualization of one DecisionTree, where nodes with majority of Sunny are coloured in yellow while Cloudy nodes are in grey:

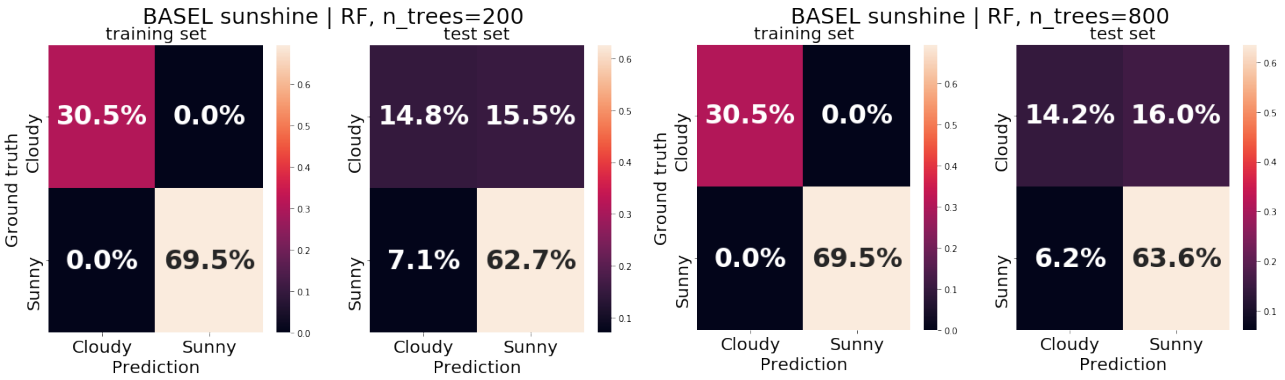


Ensemble method: RandomForest

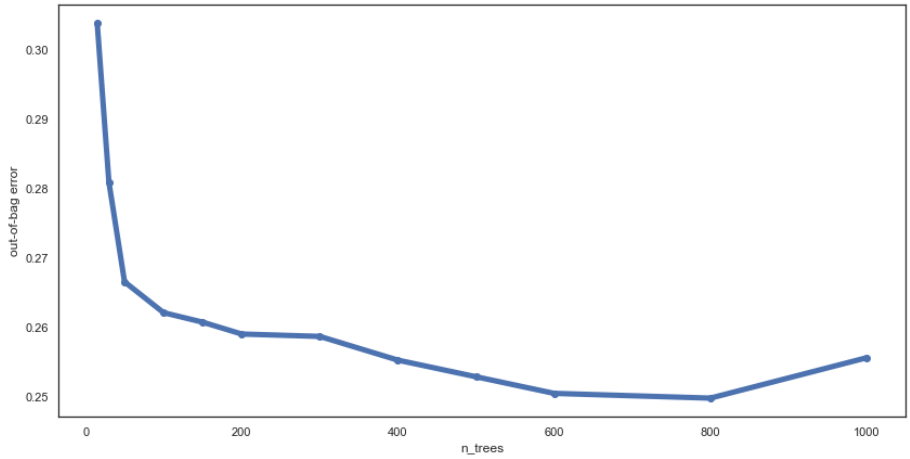
The Random Forest model **randomly** samples the training data (bootstrap samples) and the samples are fed as training data to many Decision Trees (DTs) and trained separately. The result of the ensemble model is determined by counting a majority vote from all the DTs (*Bagging*: Bootstrap Aggregation). After training the DTs, the left over samples not used (Out Of Bag) can be used for computation of OOB scores.

The *ensemble* method RandomForest we trained, using sklearn's *RandomForestClassifier*, manages to get the highest Accuracy and F1 score of all the models examined, lowering the number of False Positives compared with DecisionTree and increasing the Precision.

	Accuracy	Precision	Recall	F1	AUC
RF 200	0.774	0.802	0.898	0.847	0.693
RF 800	0.778	0.799	0.912	0.852	0.691



To examine the influence of the number of estimators on the ensemble model, we collected and plotted the out of bag errors for different number of trees:



Interestingly it appears like there are two first elbow points, one around 200 trees and one around 700-800 trees. As shown in the confusion matrices above (and also as mentioned in the discussion of other models) there is a tradeoff between precision and recall, with models tending to yield better recall at the expense of precision.

The best advantage over DecisionTree is seen in the higher proportion of True Negatives.

Alternative strategies

These are alternative strategies we have tried and compared to what presented above:

1. Pre-selecting only a subset of features and cities (to create a balanced feature set), i.e. using only the measures 'cloud_cover', 'global_radiation', 'precipitation', 'pressure', 'sunshine' for the cities 'BASEL', 'BUDAPEST', 'DE_BILT', 'DUSSELDORF', 'HEATHROW', 'LJUBLJANA', 'MAASTRICHT', 'MUENCHEN', 'OSLO';

2. Removing the MONTH feature, as it did not seem to be have high coefficients fit to it by the models trained;

3. Imputing the biggest outliers.

Results and comparison of the alternative strategies

For (1), using the balanced set with less input features resulted overall in an overall worse performance across all models tested, with usually lower accuracy and precision but usually higher recall.

For example:

	Accuracy	Precision	Recall	F1	AUC
LRL1	0.747	0.771	0.906	0.833	0.643
PF_LRL1	0.748	0.774	0.902	0.833	0.648
LRL2	0.750	0.793	0.869	0.829	0.672
PF_LRL2	0.748	0.796	0.859	0.826	0.676
SVC	0.752	0.786	0.886	0.833	0.665
SVC rbf	0.767	0.793	0.902	0.844	0.680
DT	0.705	0.720	0.943	0.817	0.549
RF	0.763	0.791	0.898	0.841	0.675

For (2), removing MONTH information resulted in negligible changes. This confirmed the observation made above (in the section on Feature importance and insights) of the minimal contribution of this information for the classification.

Finally, for (3) we also tried imputing the most extreme outliers for precipitation (369 values with more than 30 mm of daily precipitation), humidity (29 values lower than 25%) and wind_speed (30 with more than 12.5 m/s). This resulted in very similar performance, in the range of up or down 1 percent or same for the various metrics.
For example:

	Accuracy	Precision	Recall	F1	AUC
LRL1	0.752	0.784	0.890	0.834	0.662
PF_LRL1	0.762	0.802	0.875	0.837	0.688
LRL2	0.750	0.802	0.851	0.826	0.683
PF_LRL2	0.748	0.794	0.863	0.827	0.674
SVC	0.761	0.805	0.867	0.835	0.691
SVC rbf	0.770	0.803	0.888	0.844	0.693
DT	0.718	0.784	0.824	0.803	0.649
RF	0.773	0.000	0.904	0.847	0.687

Final model recommendation

After testing extensively the models described above (with and without Transforms, PolynomialFeatures, Regularisation) for all the strategies we outlined, we recommend using the **RandomForest** model, with standard **scaling** and **log1p** transformation of the most skewed features, addition of **PolynomialFeatures** of 2nd degree, **without** necessarily having to impute the most extreme outliers.

The best F1 score for the final **RandomForest** model was 0.852 and accuracy of 0.778, with hyper-parameters: `n_estimators=800, max_features=132`.

As our objective was to aim for the highest predictive power, we think this is the recommended model. If we were more interested in interpretation, then a LR L1 model could prove more straightforward to interpret.

Key findings and insights

We determined that although it is a very hard problem to predict tomorrow's weather, we can do much better with appropriate models compared to baseline assumptions.

It it also clear that regularisation is fundamental to control overfitting.

Scaling and transformation of skewed distributions help in achieving better predictions even for Logistic Regression.

Feature importance analyses showed that a small set of features are necessary and sufficient for the model to achieve its best predictive power.

It also showed how today's measure was very low in the rank of feature importance and that the month information was not necessary.

Interestingly, most of the top coefficients are for measures from locations surrounding the target city, like Dusseldorf, Roma, Muenchen, Tours.

The *RandomForestClassifier* proved to be he one with the best metrics.

Imputing the most extreme outliers offered some improvement, but not consistent and not significant. Being an extra step which could be debatable (in choosing the threshold of what is considered outlier) and as it does not offer significant improvement, we didn't recommend it.

All models seem to err more in over-predicting the positive value (Sunny) and hence with a high number of False Positives. This is consistent to what had been previously observed (in a previous project) when training *Regression* models on the same dataset, where the models made the worst predictions when the true value was 0 sunshine hours while they instead predict a certain positive number of sunshine hours.

Next steps

It would be both straightforward and interesting to retrain all models changing the target measure. In particular to see which ones are easier to predict and which ones are harder.

For example we could predict precipitation, binning precipitation levels in at least three classes, like "dry day" (less than 1mm of precipitation, cfr. the weather statistics for Basel on wikipedia) "some rain" (between 1 and 5 mm of precipitation) and "rainy day" (more than 5mm of precipitation)

Another interesting extension would be to work with a time frame of observations to predict tomorrow's weather. Instead of using only today's measure across all locations, we could use a series of measures for subsequent days (e.g. the weather of the past *week*) to make a classification prediction for tomorrow's weather.

It would also be good to explore more ensemble methods and in particular combining LogisticRegression with DecisionTrees.

Conclusions

The dataset looked like a very promising one to use for machine learning with limited amount of data cleaning necessary.

The major initial drawback was that only two features are available at all locations (temp_mean and temp_max), but it would be too restrictive to only consider these two. To achieve a balanced dataset it is better to instead drop a combination of some features and some locations in order to minimise the quantity of data left. Nevertheless the models tested appeared to be quite able to cope with the unbalanced information and limiting to a subset actually worsened the predictions.

Although the main objective was classification accuracy, we also managed to gather important interpretation insights by looking at feature importance and also by testing alternative strategies for choosing input data.

Finally we can confirm that weather prediction is definitely not an easy task.

Methods and Materials

Data manipulation and analysis was performed on a MacOS laptop using own written code in python language, working in a jupyter notebook and taking advantage of the following python libraries: pandas, seaborn, scikit-learn, cmplot

Data origin acknowledgement

Dataset compiled by Huber, Florian (2021); Zenodo. https://doi.org/10.5281/zenodo.4770937

ORIGINAL DATA TAKEN FROM:

EUROPEAN CLIMATE ASSESSMENT & DATASET (ECA&D), file created on 22-04-2021 THESE DATA CAN BE USED FREELY PROVIDED THAT THE FOLLOWING SOURCE IS ACKNOWLEDGED:

Klein Tank, A.M.G. and Coauthors, 2002. Daily dataset of 20th-century surface air temperature and precipitation series for the European Climate Assessment. Int. J. of Climatol., 22, 1441-1453. Data and metadata available at http://www.ecad.eu