

Project report

Urban IoT & Smart Cities

Summary Prediction using Weather measures

Project realized by:

Alexandre NADIRAS

Table of Contents

1. **Introduction**3
2. **Get and understand the data**4
3. **Exploratory data Analysis**7
4. **Data Preprocessing1**
5. **Machine Learning Results1**
6. **Conclusion1**
7. Introduction

The following report displays my problems, solutions and observations about my Machine Learning project. The goal of my project is to display on a mobile application a summary of the weather in real time. For example, if the weather data and the algorithm I have tell me that the weather is likely to be cloudy, then an image relative to the cloudiness of the weather will be displayed on the screen of the customer’s application.

Now in technical words it means that my target is a multiclass feature composed by a certain number of classes. Which means it is a muticlass classification problem. About the evaluation metric, the best way to make this app work is to make sure that every predictions are correct. What we want to avoid is customers to look at the application saying the weather is clear and when they go out it turns out that it rains (this is an extreme example but this is the principle). So the metric we will use for evaluting our model is precision, we want to avoid at maximum the number of False Positive in our predictions. Ideally we want the precision to be above 90% in weighted and at least 85% for each individual classes.

1. Get and understand the data

Now that our hypothesis is made we can have a look at the dataset we are going to work with. It is available [here](https://www.kaggle.com/taranvee/smart-home-dataset-with-weather-information). This dataset contains features that are measured by sensors. Each row is the average of the measure over a one minute time span. Now let’s have a look at each features of the dataset.

Our dataset is a CSV file that contains the following features:

- Time: this is the index of our dataset, in unix time

Then we have all the features related to the Energy consumption / generation which are all in kW:

- use: total energy consumption

- gen: total energy generated by means of solar or other power generation resources

- House overall: overall house energy consumption

- Dishwasher: energy consumed by specific appliance

- Furnace 1: energy consumed by specific appliance

- Furnace 2: energy consumed by specific appliance

- Home office: energy consumed by specific appliance

- Firdge: energy consumed by specific appliance

- Wine cellar: energy consumed by specific appliance

- Garage door: energy consumed by specific appliance

- Kitchen 12: energy consumption in kitchen 1

- Kitchen 14: energy consumption in Kitchen 2

- Kitchen 38: energy consumption in Kitchen 3

- Barn: energy consumed by specific appliance

- Well: energy consumed by specific appliance

- Microwave: energy consumed by specific appliance

- Living room: energy consumption in Living room

- Solar: solar power generation

And finally all the weather related features with their own unit of measurement:

- temperature: physical quantity used to tell if the air is cold or hot (the unit is not provided)

- humidity: concentration of water in the air (%)

- visibility:

- apparentTemperature: temperature perceived by humans (the unit is not provided)

- pressure: atmospheric pressure (Pa)

- windSpeed: wind's speed (we will consider the unit is in mph)

- cloudCover: the percentage of the sky recovered by clouds (%)

- windBearing: the wind's direction (measured between 0° and 360°)

- dewPoint: atmospheric temperature (the unit is not provided)

- precipProbability: probability of precipitation (no unit)

- precipItensity: the amount of rain that falls over time (no unit provided but we can assume it's in inch per squared meter)

We also have two other features that are probably used by the data collection system, including our target:

- summary: general report of the weather, for example Clear, Mostly Cloudy, etc...

- icon: icon used by the data collection system

Once i downloaded it, i created a pandas dataframe to make my analysis and manipulation on the data. When printing the shape of my dataset, i get the following result : 32 features and more than 500k rows.

By intuition, these number are very large and will probably be hard to handle since my only resource for this project is my personal computer. So we will need to make some feature engineering to reduce the number of features. It will hopefully make our life easier when training models on the dataset. I also think that a good default model for this problem is Random Forest since it generally fits the training set well even if it is likely to overfit.

1. Exploratory Data Analysis

There is a list of tasks that need to be done in this part :

* Checking feature’s type
* Checking missing values in the dataset
* Make some relevant changes to the features
* Make some descriptive statistics

The first step is to check every feature’s type. Given the fact most of the features are measures from sensors it is likely that most of them are float type. After checking, only 4 feautres are not float by objects : Time, cloudCover, Icon and Summary. The fact that Icon and Summary are object is not surprising because the values they take are strings.

However cloudCover should not be an object because it should take numerical values so we need to check why.

Also the Time is not a relevant feature in our study case so we will drop it.

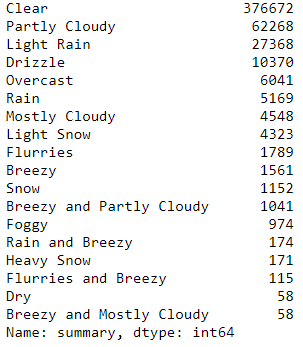
On top of this, the feature use[kW] is the exact same as the House overall [kW] feature so we can remove it too.

Back to the cloudCover feature, i checked the value that took this feature and it appears that in 58 instances it takes the value « cloudCover », which makes its type become object. This could be explained by sensor failure or a miss communication in the network. I decided to delete all instances relative to this mistakes because even if it lowers the variance of the dataset i still have a lot of instances to train on. Then i swapped its type to float.

Now it is time to deal with missing values, which tells me that all the columns have only one instance with a null value. It is likely that one row is totally womposed of NaN values. After checking it appears that le very last row was the on full of NaN values so I deleted it.

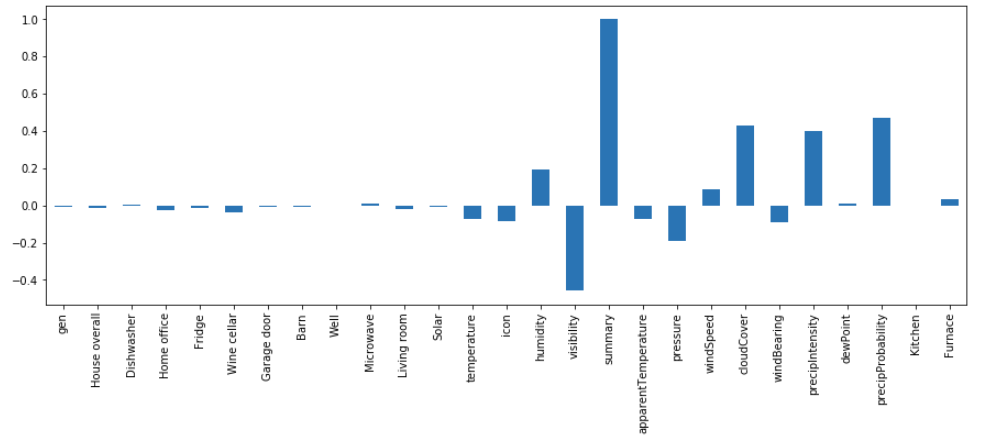
Now that our dataset is clean, it is time to make some changes to the features. The first thing to do in order to reduce the number of features is to group the similar features in a relevant way. For example I grouped together the features "Kitchen 12", "Kitchen 14" and "Kitchen 38" into one feature called « Kitchen » which is the mean of the other Kitchen value. And i did the same thing for the "Furnace 1" and "Furnace 2" features. This allowed me to have a dataset of 27 columns and still more than 500k rows before going into the categorical features encoding.

As we saw previously we have 2 categorical features in the dataset : « Icon » and « Summary », which is our target. So to make some analysis on them i need to make them numerical. There are many ways of encoding categorical features but the one i choose for this part at least is every class corresponds to one number. The Icon is composed of 9 classes and the Summary feature is composed of 18 classes.



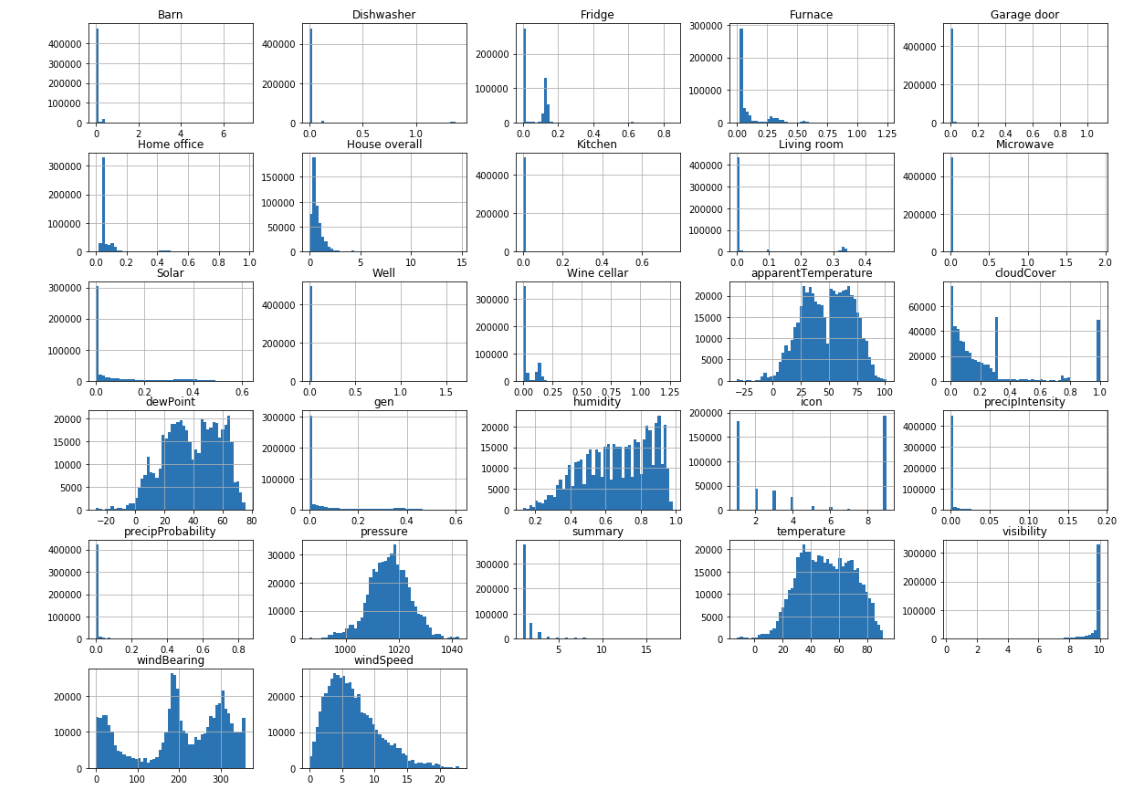
And this is where the main problem of this project is : our main target is largely unbalanced. If we wanted to put this dataset in our model for training, it is very likely that our model will perform very well on the most represented feature but very poorly (close to 0% accuracy) on all the other features. So we need to find ways to deal with this problem. For now let’s make our categorical feature take numerical values.

Once it is done we can start makin analysis because all of our features are numerical. The first thing i want to do is see if some faetures are very correlated to our target so we can think about removing them.



Unfortunatly it doens’t look like any of the features are correlated enought to my target to remove them. So we will keep them as they are. We can also notice that the data relative the the house energy consumption don’t have that much predictive power, which makes sense.

The last thing to check is the distribution of our features.



As we can see we have a lot of outlied features such as the gen, visivility or precipItensity. The presence of outliers in the dataset can be a problem, but the fact that i don't personally know how the measure where done and if an outlier is a miss measurment or just a presence of variance in the dataset make me think that i should let them be here and treated as variance. We also have some features that follow a normal distribution with the pressure and the windSpeed.

1. Data Preprocessing

In this part of the report we will :

* Reduce the number of category in the target feature
* Encode the categorical features
* Make some feature scaling

The problem i need to solve in this part is the unbalance of my dataset. There are many ways to deal with this problem but here is how i di dit. First i decided to regroup some classes that have similair behaviour in reality. I made differents groups such as :

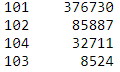
101) Clear, Dry

102) Partly Cloudy , Drizzle, Overcast, Mostly Cloudy, Breezy and Mostly Cloudy, Breezy and Partly Cloudy, Breezy

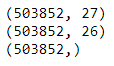
103) Flurries, Snow, Heavy Snow, Foggy, Light Snow, Flurries and Breezy

104) Light Rain, Rain, Rain and Breezy

These groups helped me reducing the number of class to 4 and have the following repartition :



Once this was done i splitted my dataset to have one with my target variable and one with the other features. The shape of each dataset is like following :



The next step is to scale the features because some algorithms may perform badly with unscaled features. I decided to use a MinMaxScaler because the feature that are not scaled don’t follow a normal distribution.

Then for better encoding pratices i decided to use a OneHotEncoder on the feature « Icon » to create 9 features.

After this, i randomized the indexing of my dataset in order to prevent the pattern creation that is naturally happening.

Lastly, to solve my problem of unbalanced, i decided to use the oversampling method which allowed me to create some new instances based on the k neighbours algorithm. This is likely to make my models overfit the data for the less represented classes but the final results are better than when i tested undersampling.



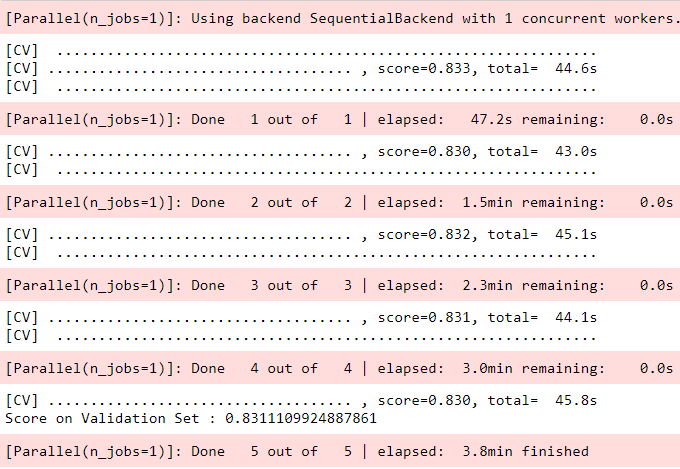
This is result i finally get, with 4 classes having 376730 instances. However it is likely that a lot of the created instances are close or even clones of existing instances. That might lead to a good performance of my model on the training set but overall bad performance on a predictive point of view.

Now that this is done we can start the machine learning part of the project.

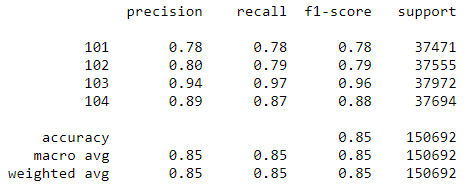
1. Machine Learning

The first thing to do here is to split our dataset into a training and a test set. I choose to take a test set representing 10% of the original dataset.

I decided to train 2 algorithms, first a default random forest using a 5 fold cross validation. This is the result i got :

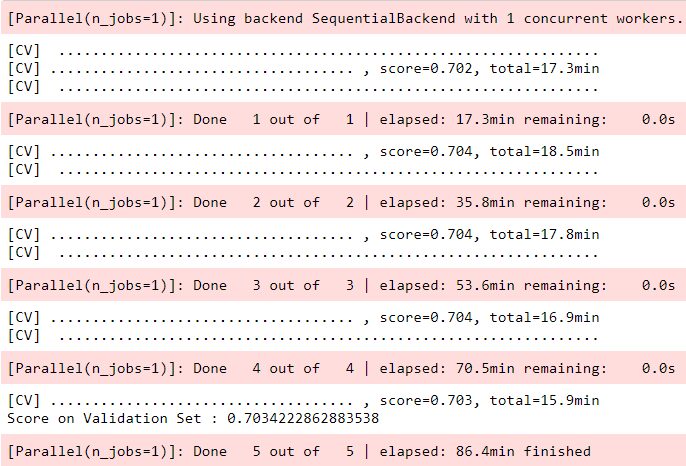


This is a pretty good performance on the training set as I expected , and it didn’t took that long to execute now let’s see how it performs on the test set.



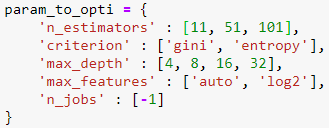
This result shows that as I expected the model performs pretty good on the less represented classes which can be explain by the mothod i used to oversample my dataset. But still, the model preforms pretty well on the classes that have the most variance. Given our performance metric, we are not satisfied with this flat model, so we will try another one and compare them, and then mae some hyperparameters tuning.

Let’s try our second model a KNN classification :



As we can see, this model was much longer to train and has an overall lower performance. So i decided not to try to optimize it. It may be a mistakes but given the fact that i don’t have a lot of resources to train my models i prefered not to waste time trying to optimize a model that is likely to perform lower than the first one.

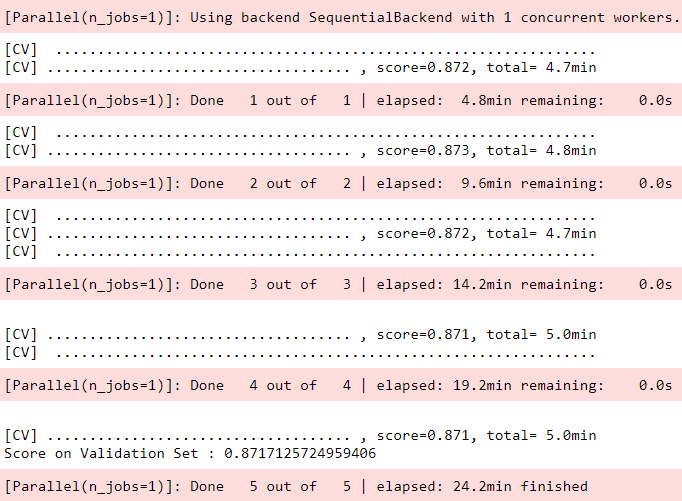
Let’s try to optimize the Random Forest model to see if we can make it perform better. I tried with the following parameters  to optimize the overall precision of the model with a GridSearchCV :



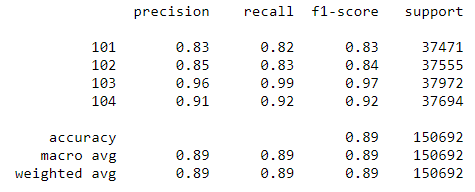
This is the result that gives me the best estimated parameters :



After this tuning i ran the new optimized random forest which gave me this result on a 5fold cross validation :



After this i tested this model on the test set and got his result :



We will consider this result to be acceptable as we have a precision for each class that is very close to 90% which was our target value.

1. Conclusion and areas of improvement

Throught this project i made a multiclass classification with a very unbalanced dataset. I managed to get to a relatively good result that doesn’t meet our expectations, however we have to put these results into perspective, we made a lot of transformations to our dataset before feeding it in our model so we might be biased when thinking about the right model to use.

Furthermore this project can be optimized, here is a list of areas of improvements :

* Using more models and comparing the performances using GridSearchCV on more hyperparameters, indeed, we managed to improve our performance but nothing too impressive. I could have make a better job at predicting the most represented class.
* Dealing differently with the missing values
* Dealing differently with the problem of unbalanced classes, like providing more data, indeed our final model has good performance, but on a dataset that is not representative of the reality.
* Maybe using Deep Learning models such as Neural Network for a better precision, but i have no guarantee on the final result.