**Regression hackathon**

**Authors:** Pablo García, Javier Ríos, Andrés Martínez

**Data preprocessing:**

(Check the programm script)

**Model comparison:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Structure** | **Inputs** | **Error training** | **Error test** | **Error CV** |
| SVR | {'regressor\_\_C': 10, 'regressor\_\_gamma': 0.01, 'regressor\_\_kernel': 'rbf'} | {  'regressor\_\_kernel' : ['linear', 'rbf', 'sigmoid'],  'regressor\_\_C' : [1,5,10],  'regressor\_\_gamma': [0.001, 0.01, 1]  } | 20.75 | 18.28 | 211.86 |
| KNN | {'regressor\_\_n\_neighbors': 2} | {  'regressor\_\_n\_neighbors': [2,3,5,7,10],  } | 7.749 | 13.166 | 21.2945 |
| TREE | {'regressor\_\_max\_depth': 7, 'regressor\_\_min\_samples\_leaf': 5, 'regressor\_\_splitter': 'random'} | {  'regressor\_\_splitter': ['best','random'],  'regressor\_\_max\_depth': [3,5,7,8],  'regressor\_\_min\_samples\_leaf': [1,5,10]  } | 17.826 | 21.338 | 25.256 |
| SLR | {'regressor\_\_positive': False} | {  'regressor\_\_positive': [True, False]  } | 21.344 | 19.99 | 22.458 |
| BR | {'regressor\_\_bootstrap': False, 'regressor\_\_bootstrap\_features': False, 'regressor\_\_max\_features': 0.5, 'regressor\_\_max\_samples': 0.6, 'regressor\_\_n\_estimators': 10} | {  'regressor\_\_n\_estimators': [5, 10, 15],  'regressor\_\_max\_samples': [0.4, 0.5, 0.6],  'regressor\_\_max\_features': [0.3, 0.4, 0.5],  'regressor\_\_bootstrap': [True, False],  'regressor\_\_bootstrap\_features': [True, False]  } | 7.619 | 15.38 | 21.87 |

**Conclusions:**

After testing this list of models, K-Nearest Neighbors comes out as the best model on this particular dataset. At first, this didn’t make much sense to us, as it is surprising how, a relatively simple model as KNN, can outperform much more complex ones such as SVR.

Then, after giving it a thought, we came to the conclusion that this case is optimal for KNN for three reasons: First, we don’t have too many features and the ratio “data-points” to features is pretty good (as we have 1538 hours of data), so that the Curse of Dimensionality (one of the main problems with KNN) doesn’t really affect this case. Second, KNN excels when the data is well-behaved and the relationships between the features and the target is simple. In this case, the data checks both conditions, as it comes from a real-world phenomenon that we can expect to have a natural behavior. Last, this data may be pretty noisy, as even though the trend in variables (temperature, wind direction and, specially, wind speed) may be clear over long periods of time, individual measurements may happen to be far away from it. Given that this model performs really well, comparatively, on noisy datasets, we can see why it’s the best.

Additionally, the more complex algorithms may need a larger amount of information to optimally recreate the reality.