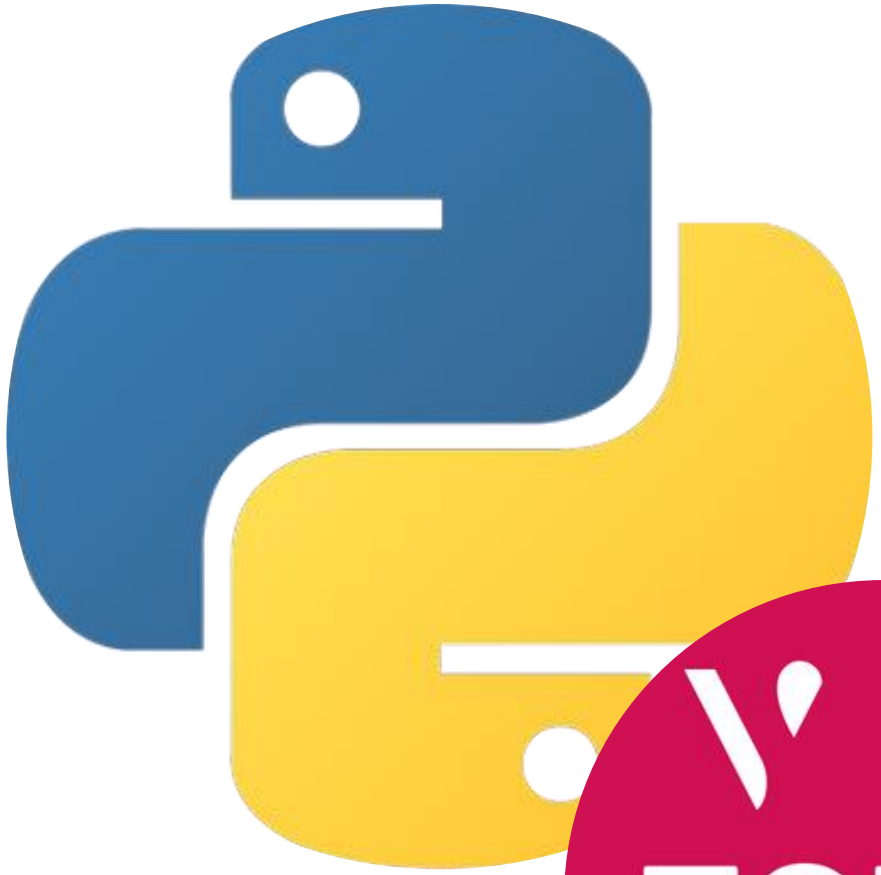
Python for data analysis: project

Florent Drilhon – DIA3



Introduction

- The dataset I had to study is the dataset QSAR biodegradation dataset available here : https://archive.ics.uci.edu/ml/datasets/QSAR+biodegradation

- The purpose of this dataset is to predict if a chemical is ready biodegradable or not given a set of parameters.

So the problem is a binary classification.

- Objectives: As we do not have further information on the problem, our goal will be to maximize not only the accuracy but also the f1-score of the model we will construct. Because the f1-score represent the ratio between precision and recall, so maximizing it is synonym of building a good polyvalent model.

Summary

- Data exploratory:
  - Structural analysis
  - Content analysis
  - Relations/Hypothesis check
- Preprocessing:
  - Structuration of the data
  - Evaluation process
  - Feature engineering
- Model selection:
  - Model comparison
  - Hyper-parameters adjustments
  - Threshold selection

Data exploratory

In this part, I checked the structure of the given dataset in a first time to have a global viewpoint, then I look more in details to have a good comprehension of it.

- Structural analysis:
  - Target variable: experimental class -> binary variable
  - Rows and columns: 1055 rows 42 columns
  - Variable types: 24 parameters are integers, 17 are float and 1 is object (the response)
  - Missing values: No missing values

Data exploratory: Content analysis

Target vizualisation: 34% are ready Biodegradable and 66% are not

Variable signification:

- Continuous variable: most seem to follow Gaussian distribution. We can see links between some of the parameters:
  - Some are scaled on carbon atom: means
  - Some are eigenvalues
  - Some are indexes
  - Some issued from the same matrix: Laplace or Burden

- Integer variables: some are binary, most have less than 15 values. Identification of different types of variable : number of atoms, frequency of bond, presence/absence of bond (binary)

So we have identified different potential relations between the parameters, and some hypothesis we will have to confirm later.

Relation target/parameter:

For this part, we will look at the distribution of the target following the different features. I gathered the features by the categories identified earlier to facilitate the visualization.

- number of atoms: we can difference in the distribution of the target for the number of oxygen atoms and nitrogen atoms.
- Frequency: No real difference can be inferred.
- Presence/absence: in the 3 cases, the presence of the bond results in most of the cases in a NRB, maybe a link
- Indexes: differences in Hyper-Wiener (log), lopping centric, Lovasz-Pelikan, Intrinsic state pseudoconnectivity (less flagrant)
- Eigenvalues: differences in Laplace matrix, adjency (Lovasz-Pelikan) and Burden matrix
- Others: in percentage of C atoms, spectral moment from Laplace Matrix, Spectral positive sum from Burden matrix, Spectral Moment from Burden matrix
- The variables from the two matrices seems to be representative too.

Data exploratory

In this part I checked for the relations between the different features by looking at the correlations matrices or their distributions in function of the case.

- Parameter/parameter relations
  - Number atom/number atom: halogen relatively correlated to sssssC/CRX3&Heavy atoms
  - Number atom/ frequency of bond
  - Frequency of bond/ frequency of bond : All the bonds C-N seems to be correlated
  - Presence-absence/presence-absence:pratically no atom have both C-Br (dist 1 or 4) and C-C. every of those which have C-Br4 have C-Br1 and 2/3 of thos which have Br1 have Br3
  - Presence-absence/Number atom: Br atom is an halogen so the relation is logical. Difference for the oxygen (-> similar to the relation with the target)
  - Indexes/indexes relations : 75% correlation between Second Mohar Index and Lopping centric index.

Then we want to verify the relations between some feature and the target we identified earlier. To test the dependance I used a Student test for the numerical features, and a chi2 test for the binary one (presence/absence).

- Hypothesis to check:

  Relation target/parameter:
  - number of atoms: difference in the number of oxygen atom, nitrogen atom
    -> threshold 1%, dependance with all the features except terminal primary C, N hydrazines, CRX3, donor atoms
  - Presence/absence: in the 3 cases, the presence of the bond results in most of the cases in a NRB, maybe a link
    -> threshold 1%: H0 rejected for C-Br1 and C-Cl (strongly)

Preprocessing

The data is explored and we have a good comprehension of it, we can initiate the preprocessing phase by structuring the data.

- Structure the data:
  - Creation of test/train set: as the the size of the dataset is not really consequent we choose a spliting rate of 0.2 to avoid losing some precious information.
  - Encryption of the target: as the target is categorical, it could be useful to encrypt it to give it an heavier meaning.
  - No need to deal with missing value because there isn't any
  - Create a first function of preprocessing that applies all of this and returns X and y

Now that we have structured data, we can build a first model that we will use to test all the preprocessing settings we will try. A decision tree is a good model to do so because it is light to compute and very adapted to classification problems.

We also build an evaluation procedure that allows us to see the aspects of a model and how to improve it.

- Evaluation process:
  - To evaluate the models, it is interesting to focus on the accuracy as long as the target is binary, but not only ! We want also to know the sensibility and the recall of the model because these parameters are more significant to improve the model. We will visualize this thanks to the f1-score.
  - Another point is to look at the fitting of the model to adapt it. For that we use learning curves.

After building a first model we remark that the model is overfitting so there are several solutions :

- We can augment the size of the train set -> we don't have enough value to do that
- We can do a feature selection -> that seems to be a good solution

Preprocessing: feature engineering

- Feature transformation:
  - As we have a lot of continuous parameters, it can be interesting to standardize them for building models -> better results
  - We can also try to encode the presence/absence values to give them more significance as they are considered as continuous variables. -> good but don't know how to plot the learning curves.
  - We remark also that the « number of atoms » features are not significant so we try to create a new feature «total number of atoms » which is the sum of all these variables -> not convincing either
  - After the vizualisation of the importance of the features, we notice that the variables about the presence/absence are not significant to the model, we can try to combine them at first. But the result is not convincing

- Feature standardization:

  As we have a lot of continuous variable, and we will try several models in the following part, it is important to construct a transformer that will scale the selected features when needed. But we have to be careful, standardized values can lower the performances of some models like tree-based models for example.

- Feature selection:

  We try feature selection with a recursive feature elimination with cross validation (RFECV) selector by importance scoring. For the RFE CV selector, we I choosed the Stochastic Gradient Descent classifier, because this is a model that provides a good approach to deal with the features importance.

  By setting the parameters to 4 minimum features and a cross-validation of 8, 16 features are selected.

  To test these new changes, we change the model we use for a Logistic regression to integrate the transformer.

  The selector seems to be not very useful but can show an importance with other models so we will keep it warm.

Model selection

Now that we have a clean dataset and a good preprocessing, we can test different models and select the most promising one.

To test the different models, I differentiated two different types of preprocessor to integrate in the pipeline, one for the tree-based models without the scaler and one for the other models with she scaler.

- Model selection:

That step is pretty simple, we choose several models that can be adapted to our situation and compare their results and their learning curves thanks to our wonderful evaluation function. The goal is not only to select the model with the best results but also to consider the improvement capacity of the model.

- For this step I selected several model to test :
  - Random Forest: very adaptable model
  - Adaboost: similar to random forest but more subtile, could be more intersting with its feature selection
  - Support Vector Machine: model which could be very interesting with a medium/small dataset which is our case here.
  - KNN: simple model of classification but very effective when the situation is favourable
  - Logistique regression: Once again very simple but could be interesting because we have a lot of continuous parameters
  - Tree: If it works well we can find other tree-based model to boost
  - SGD: This model usually works well with big dataset, but who knows, it could work in our case too.

- Results:

  After analysing the results (available on the notebook), it can be inferred that the tree based model are not that effective, except for Adaboost which behave as if it not tree-based (this is why I used the other preprocessor by the way). The support vector machine result obtain good results too. The logistic regression and KNN are not bad but their learning curves seem to reach a stagning point. For the SGD the results are good but the learning curves show a very unstable model.

  After analysing the results, I chose 2 models to focus on : SVM and Adaboost.

Model improvement

- Support Vector Machine:

To improve the support vector machine model, first we identify the hyper-parameters on which we will be able to play to boost the model.

I chose to act on the hyper parameters:

- C : regularization parameter which can variate a lot
- Gamma: which is the kernel coefficient

To modify the parameters, I used Randomized search CV several times, I chose this method because of the parameter C which can variate a lot, so a GridSearch could not be enough powerful to finf the best C.

After reducing the range of the parameters, the best couple I obtained is : C= 1481 and Gamma = 0.001

- AdaBoost:

To improve the Adaboost model, first we identify the hyper-parameters on which we will be able to play to boost the model.

I chose to act on the hyper parameters:

- Learning rate: rate that shrinks the contribution of each classifier.
- Maximimum number of estimators.

Again, I used a RandomizedSearchCV to test the different parameters.

After a few tries, no results were convincing, I presume the AdaBoost model needs a bigger dataset to train itself if we want to reach high performances.

Threshold selection

As our case is a binary classification problem, we can improve the performances by defining a decision threshold ourselves. In fact, the threshold of all scikit-learn models is predifined and cannot be changed.

I consider this part as a bonus because I was not able to integrate the threshold in the final model so I did not include it in the model used for the flask app.

Our goal was to reach the higher f1-score possible, so thanks to the precision/recall curve we can easily find the breaking point where the f1-score (which is the rapport between precision and recall) is maximium.

We just have to identifiate that point and create another simple function to deal with the results.

Conclusion

After trying different types of models/parameters/preprocessing, the best model I built is a Support Vector Machine model.