Project 6 : Prediction of chemicals biodegradation

Objectives of the project

The goal and the steps of this project are :

1. Data preprocessing
2. Feature selection for model(s)
3. Use useless features for clustering
4. Use useful features AND cluster for modeling.

1: Data preprocessing

The data have been used to study the relationships between chemical structure and biodegradation of molecules.

The dataset contains:

- 1055 chemicals,
- 41 molecular descriptors,
- 1 column biodegradable or not

Most important metrics : **Precision**

→ *The less False Positive the better*

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1055 entries, 0 to 1054
Data columns (total 42 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   SpMax_L             1055 non-null   float64
 1   J_Dz                1055 non-null   float64
 2   nHM                 1055 non-null   int64
 3   F01                 1055 non-null   int64
 4   F04                 1055 non-null   int64
 5   NssssC              1055 non-null   int64
 6   nCb                 1055 non-null   int64
 7   C%                  1055 non-null   float64
 8   nCp                 1055 non-null   int64
 9   nO                  1055 non-null   int64
 10  F03                 1055 non-null   int64
 11  SdssC               1055 non-null   float64
 12  HyWi_B              1055 non-null   float64
 13  LOC                 1055 non-null   float64
 14  SM6_L               1055 non-null   float64
 15  F03                 1055 non-null   int64
 16  Me                  1055 non-null   float64
 17  Mi                  1055 non-null   float64
 18  nNN                 1055 non-null   int64
 19  nArNO2              1055 non-null   int64
 20  nCRX3               1055 non-null   int64
 21  SpPosA_B            1055 non-null   float64
 22  nCIR                1055 non-null   int64
 23  B01                 1055 non-null   int64
 24  B03                 1055 non-null   int64
 25  N073                1055 non-null   int64
 26  SpMax_A             1055 non-null   float64
 27  Psi_i_1d            1055 non-null   float64
 28  B04                 1055 non-null   int64
 29  SdO                 1055 non-null   float64
 30  TI2_L               1055 non-null   float64
 31  nCrt                1055 non-null   int64
 32  C026                1055 non-null   int64
 33  F02                 1055 non-null   int64
 34  nHDon               1055 non-null   int64
 35  SpMax_B             1055 non-null   float64
 36  Psi_i_A             1055 non-null   float64
 37  nN                  1055 non-null   int64
 38  SM6_B               1055 non-null   float64
 39  nArCOOR             1055 non-null   int64
 40  nX                  1055 non-null   int64
 41  experimental_class  1055 non-null   int64
dtypes: float64(17), int64(25)
memory usage: 346.3 KB
```

3: Function cross validation

Definition of a function which runs all models and compares them with cross validation and computes metrics for dataframe for all models

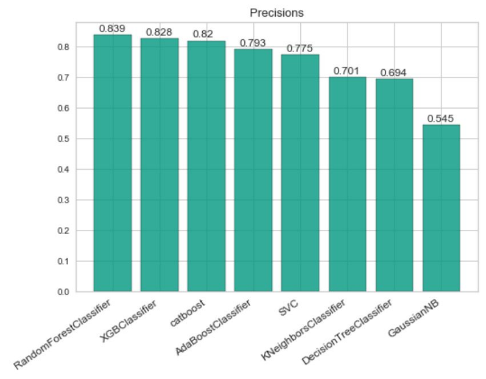
<u>Input</u>:

- Dataframe

<u>Outputs</u>:

- Table with all mean of metrics per model, ordered by precision
- A graphic with precisions per model

```python
def f_cross(df, n_splits=5):

    X=df.drop('experimental_class',axis=1)
    y=df.experimental_class

    res_cross={}
    kfold=KFold(n_splits=n_splits, shuffle=True, random_state=42)


    for model in models:
        list_of_accuracies=[]
        list_of_precision=[]
        list_of_recall=[]
        list_of_f1=[]
        for train, test in kfold.split(X):
            model.fit(X.iloc[train], y.iloc[train])
            list_of_accuracies.append(accuracy_score(y.iloc[test], model.predict(X.iloc[test])))
            list_of_precision.append(precision_score(y.iloc[test], model.predict(X.iloc[test])))
            list_of_recall.append(recall_score(y.iloc[test], model.predict(X.iloc[test])))
            list_of_f1.append(f1_score(y.iloc[test], model.predict(X.iloc[test])))
        res_cross[str(model).split("(")[0].split('.')[0].replace('<','')]=[round(np.mean(list_of_accuracies),3),
                                                     round(np.mean(list_of_precision),3),
                                                    round(np.mean(list_of_recall),3), round(np.mean(list_of_f1),3)]
        list_of_precision2=[round(i,3) for i in list_of_precision]
        print(model, ':', list_of_precision2)

    #The datafrane res_cross contains all the metrics value for all the models
    res_cross=pd.DataFrame(res_cross).T
    res_cross.columns=['accuracy', 'precision', 'recall', 'f1' ]
    res_cross.sort_values(by='precision', ascending=False, inplace=True)


    #display precision for each model
    x=np.arange(len(res_cross.index))
    plt.bar(x, res_cross.precision, color = (0,0.6,0.5,0.8), edgecolor='black')
    plt.xticks(x, res_cross.index, rotation=35,horizontalalignment='right', fontsize=12)
    plt.title('Precisions', fontsize=13)
    for i in range(len(res_cross.precision)):
        plt.text(x = i-0.3, y = res_cross.precision[i]+0.01, s = res_cross.precision[i])
    plt.show()

    return res_cross
```



	precision	accuracy	recall	f1
RandomForestClassifier	0.839	0.863	0.731	0.780
XGBClassifier	0.828	0.865	0.753	0.788
catboost	0.820	0.864	0.759	0.788
AdaBoostClassifier	0.793	0.856	0.774	0.782
SVC	0.775	0.807	0.599	0.675
KNeighborsClassifier	0.701	0.808	0.748	0.722
DecisionTreeClassifier	0.694	0.797	0.712	0.702
GaussianNB	0.545	0.714	0.937	0.688

2: Function cluster

Function which :

- makes clusters using K-means with the useless features
- Allows to choose the best number of clusters according to 3 metrics

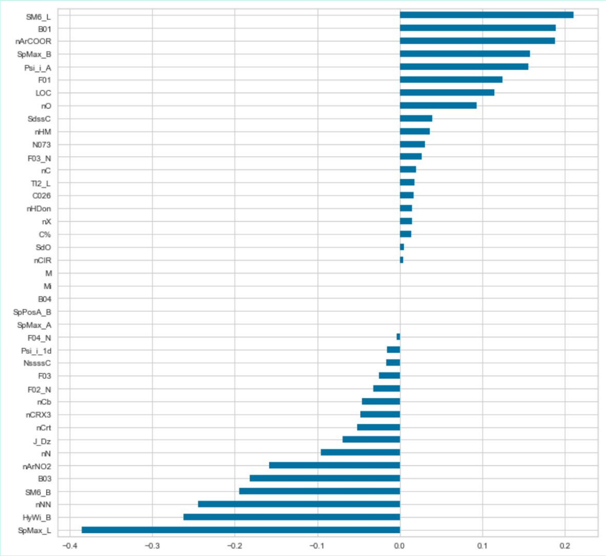
Output:

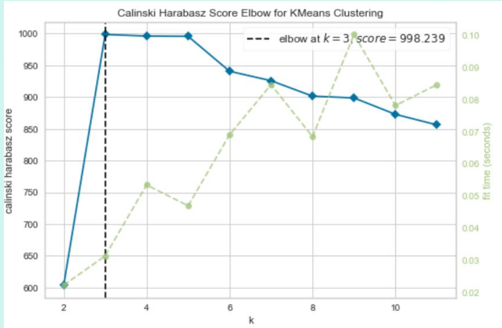
- Returns a new dataframe with important features & a column cluster.

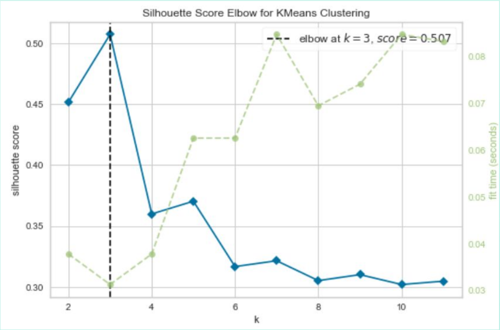
```
def f_cluster(df, useless_col):

    #creation of a new dataframe df_cluster with useless features
    df_cluster=df[[c for c in df.columns if c in useless_col]]
    model_c=KMeans()

    df2=df.drop(useless_col, axis=1)

    #cluster with useless features, choice of k value after viz elbow visualiser
    viz=KElbowVisualizer(model_c, k=(2,12))
    viz.fit(df_cluster)
    viz.show()
    viz=KElbowVisualizer(model_c, k=(2,12), metric='calinski_harabasz')
    viz.fit(df_cluster)
    viz.show()
    viz=KElbowVisualizer(model_c, k=(2,12), metric='silhouette')
    viz.fit(df_cluster)
    viz.show()
    k = input()
    k=int(k)
    kmeans=KMeans(k)
    kmeans.fit(df_cluster)
    kmeans.predict(df_cluster)
    df2['cluster']=kmeans.predict(df_cluster)
    return df2
```

4: Feature selection & cluster : Lasso

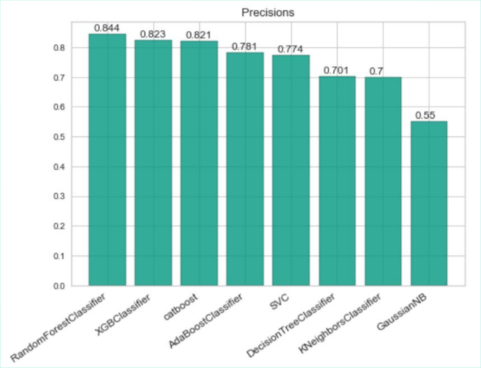
- 1st test with Lasso:
  - 5 useless columns : ['M','Mi','B04','SpPosA_B','SpMax_A']
  - Those 5 columns are used to build cluster with k-means => 3 clusters







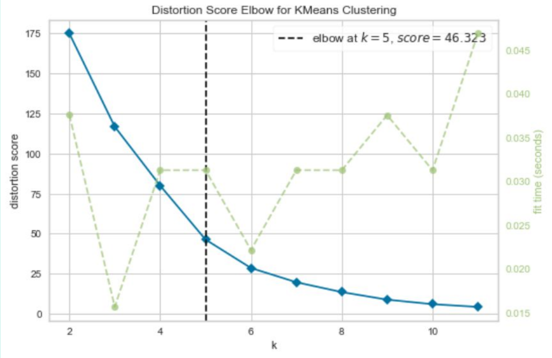
	precision	accuracy	recall	f1
RandomForestClassifier	0.844	0.869	0.747	0.792
XGBClassifier	0.823	0.862	0.748	0.783
catboost	0.821	0.864	0.763	0.790
AdaBoostClassifier	0.781	0.848	0.765	0.772
SVC	0.774	0.806	0.596	0.673
DecisionTreeClassifier	0.701	0.799	0.709	0.704
KNeighborsClassifier	0.700	0.807	0.745	0.720
GaussianNB	0.550	0.718	0.937	0.692

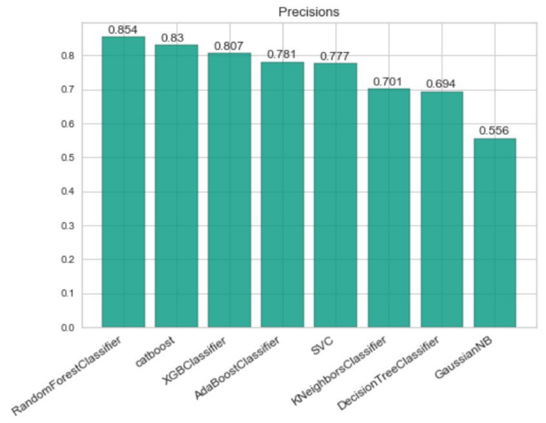


4: Feature selection & cluster : RFE & Random Forest

- 1st test with Lasso:
  - 5 useless columns : ['nArNO2', 'nCRX3', 'B01', 'N073', 'B04']
  - Those 5 columns are used to build cluster with k-means => 5 clusters

	precision	accuracy	recall	f1
RandomForestClassifier	0.854	0.867	0.730	0.786
catboost	0.830	0.865	0.754	0.790
XGBClassifier	0.807	0.860	0.766	0.785
AdaBoostClassifier	0.781	0.857	0.799	0.789
SVC	0.777	0.808	0.602	0.677
KNeighborsClassifier	0.701	0.808	0.748	0.722
DecisionTreeClassifier	0.694	0.798	0.714	0.703
GaussianNB	0.556	0.726	0.928	0.696





5: To conclude

- Best model : **Random Forest**
- Best method of feature selection : **RFE**

Learnings / Improvements

1. Learnings:
    a. Cross validation
    b. Feature selection
    c. Cluster

2. Improvements:
    a. Visualisation of clusters

THANKS

Do you have any questions?