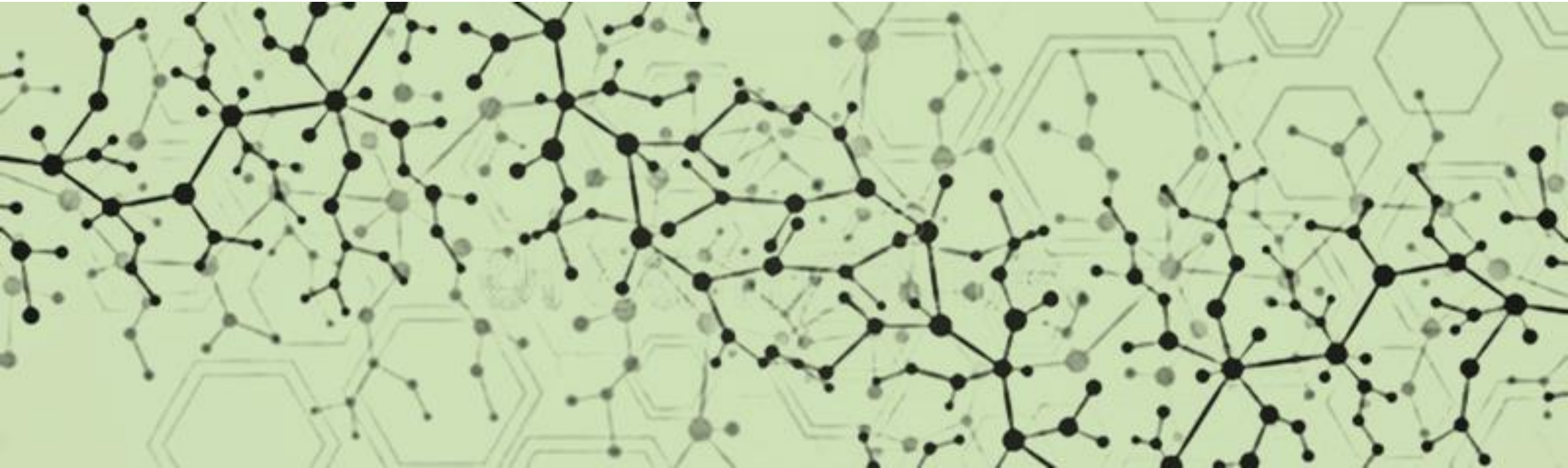
PYTHON FOR DATA ANALYSIS

Project 2023

ESILV DIA 4 : Paul RUNAVOT, Eleonor KIOULOU, Adèle MONTOYA

OBJECTIVE

- Analyze a dataset and choose the best model



PLAN

1. EXPLORE THE DATASET

Cleaning, encoding, normalization, imputation

2. DATA VISUALIZATION

show the link between the variables and the target

3. MODELING

Try several algorithms, change the hyper parameters, do a grid search, compare the results of our models using graphics

4. API

Transformation of the model into an API

1. EXPLORE

OUR DATASET

SUBJECT

Specificities of molecules

EXAMPLE

```
col_names=[
"1) SpMax_L: Leading eigenvalue from Laplace matrix",
"2) J_Dz(e): Balaban-like index from Barysz matrix weighted by Sanderson electronegativity",
"3) nHM: Number of heavy atoms",
"4) F01[N-N]: Frequency of N-N at topological distance 1",
"5) F04[C-N]: Frequency of C-N at topological distance 4",
'6) NssssC: Number of atoms of type ssssC',
'7) nCb-: Number of substituted benzene C(sp2)',
'8) C%: Percentage of C atoms',
'9) nCp: Number of terminal primary C(sp3)',
'10) nO: Number of oxygen atoms',
'11) F03[C-N]: Frequency of C-N at topological distance 3',
'12) SdssC: Sum of dssC E-states',
'13) HyWi_B(m): Hyper-Wiener-like index (log function) from Burden matrix weighted by mass',
'14) LOC: Lopping centric index',
'15) SM6_L: Spectral moment of order 6 from Laplace matrix',
'16) F03[C-O]: Frequency of C - O at topological distance 3',
'17) Me: Mean atomic Sanderson electronegativity (scaled on Carbon atom)',
'18) Mi: Mean first ionization potential (scaled on Carbon atom)',
'19) nN-N: Number of N hydrazines',
```

DIMENSIONS

Number of columns : 42

Number of rows : 1 055

CLEANING

SIMPLIFY COLUMN NAMES

```
col_acr=[]
for a in col_names:
    col_acr.append(re.findall('(?<=\) )(.*?)(?=\:)',a))
col_acr
```

```
[['SpMax_L'],
 ['J_Dz(e)'],
 ['nHM'],
 ['F01[N-N]'],
 ['F04[C-N]'],
 ['NssssC'],
 ['nCb-'],
 ['C%'],
 ['nCp'],
 ['nO'],
 ['F03[C-N]'],
 ['SdssC'],
 ['HyWi_B(m)'],
 ['LOC'],
 ['SM6_L'],
 ['F03[C-O]'],
 ['Me'],
 ['Mi'],
 ['nN-N'],
```

CHECK NULL VALUES

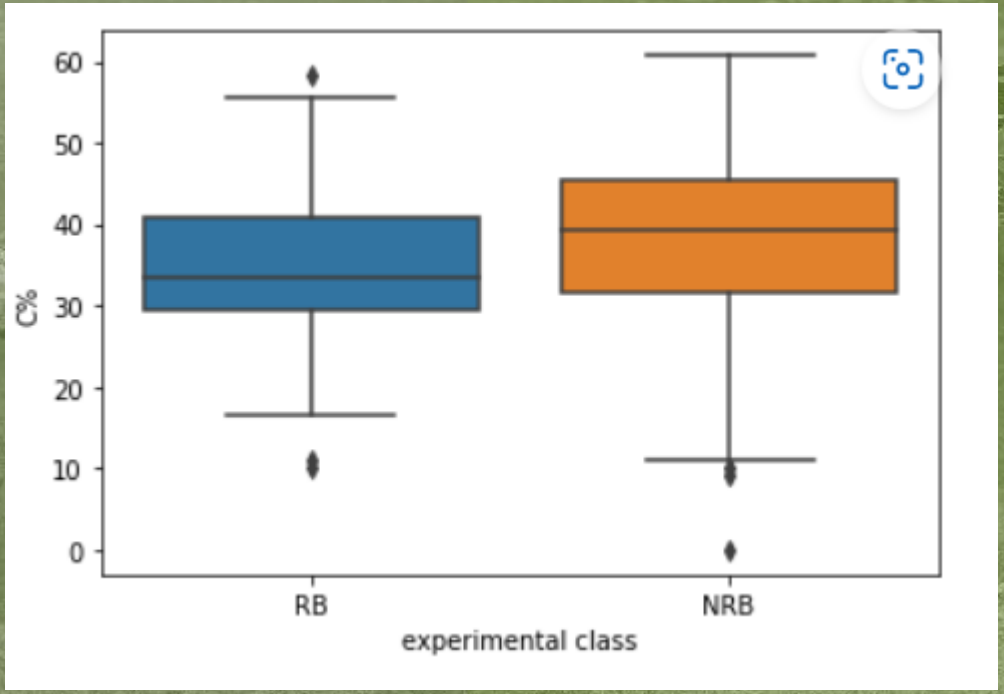
```
RangeIndex: 1055 entries, 0 to 1054
Data columns (total 42 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   SpMax_L             1055 non-null   float64
 1   J_Dz(e)             1055 non-null   float64
 2   nHM                 1055 non-null   int64
 3   F01[N-N]            1055 non-null   int64
 4   F04[C-N]            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[C-N]            1055 non-null   int64
 11  SdssC               1055 non-null   float64
 12  HyWi_B(m)           1055 non-null   float64
 13  LOC                 1055 non-null   float64
```

PROBLEMATIC

Can we predict whether a molecule is biodegradable in light of these characteristics?

2. VISUALIZATION

BOXPLOT



OBJECTIF

Show the link between the variables and the target

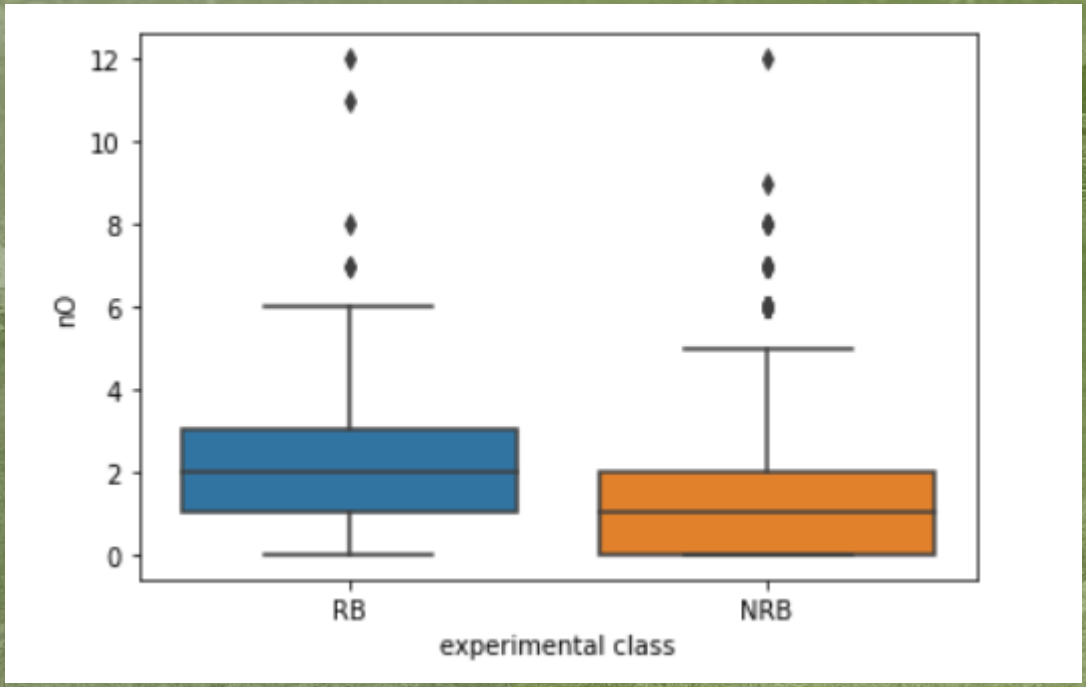
CODE

```
sns.boxplot(df["experimental class"],df['C%'])
plt.show()
```

RESULT

- The carbon level for non-biodegradable molecules is higher.
- It can be assumed that the carbon content has an impact on the bio/non-bio decision

BOXPLOT



OBJECTIF

Show the link between the variables and the target

CODE

```
sns.boxplot(df["experimental class"],df['nO'])
plt.show()
```

RESULT

- The number of oxygen atoms for biodegradable molecules is higher.
- It can be assumed that the oxygen content has an impact on the bio/non-bio decision

3. MODELING

PREPROCESSING

TRAIN TEST SPLIT

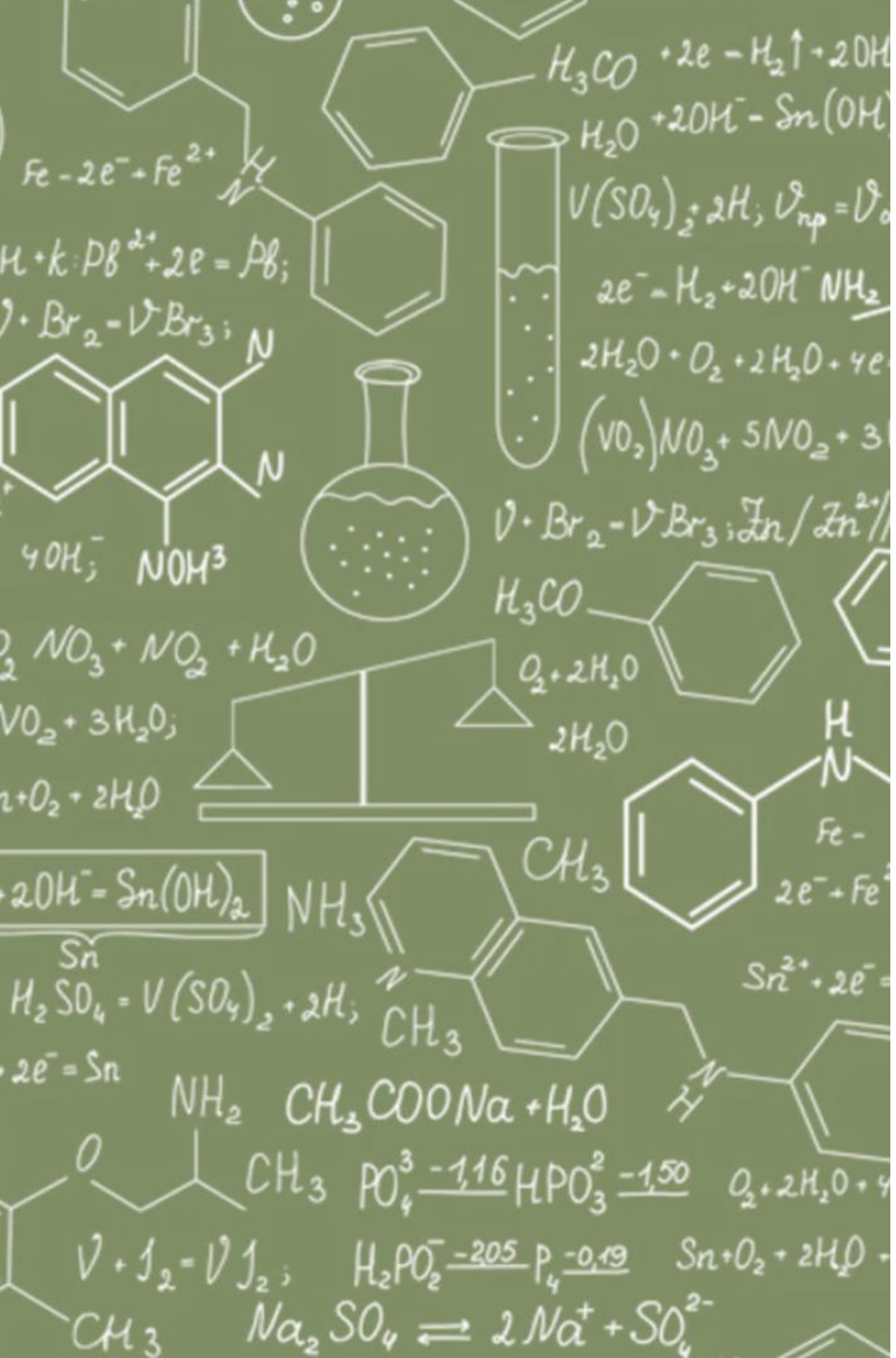
Data separation: train and set (80%/20%)

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df.drop('experimental class',axis=1),df['experimental class'],test_size=0.2, random_state=42)
```

SCALING

Scaling, which helps standardize data and can help speed up algorithm computations

```
from sklearn import preprocessing

scaler=preprocessing.StandardScaler().fit(X_train)
X_train_scale=scaler.transform(X_train)
X_test_scale=scaler.transform(X_test)
```



ETAPES MODELING

1. CHOOSE A CLASSIFER MODEL

Choose an adapted model, and try to optimize their parameters.

2. SCORE

We chose the accuracy metrics,
To evaluate our models.

3. VISUALISATION

A method for graphically summarizing statistical data in order to show the links between data sets

4. CONCLUSION

All analyses and graphs need an explanation to be understood. Conclusions are then drawn to move forward in the process

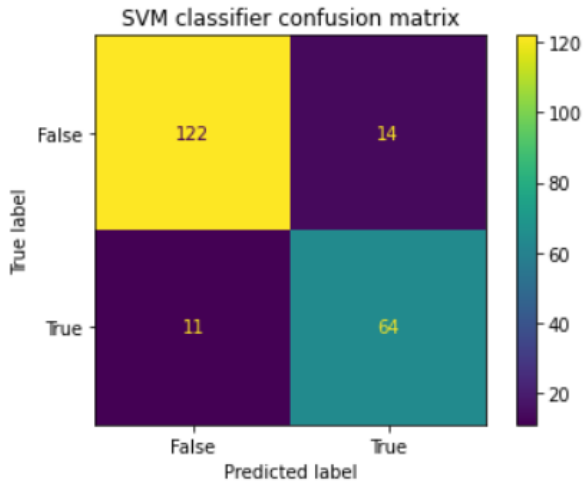
SUPPORT VECTOR MACHINE CLASSIFIER

SCORE – MODEL WITH OPTIMIZED PARAMETERS AFTER GRIDSEARH

```
cl_svm=svm.SVC(kernel='rbf',C=1,coef0=0.01,degree=1,gamma='auto')
cl_svm.fit(X_train_scale,y_train)

print(cl_svm.score(X_train_scale,y_train))
print(cl_svm.score(X_test_scale,y_test))
```

```
0.9028436018957346
0.8815165876777251
```

PLOTTING CONFUSION MATRIX :

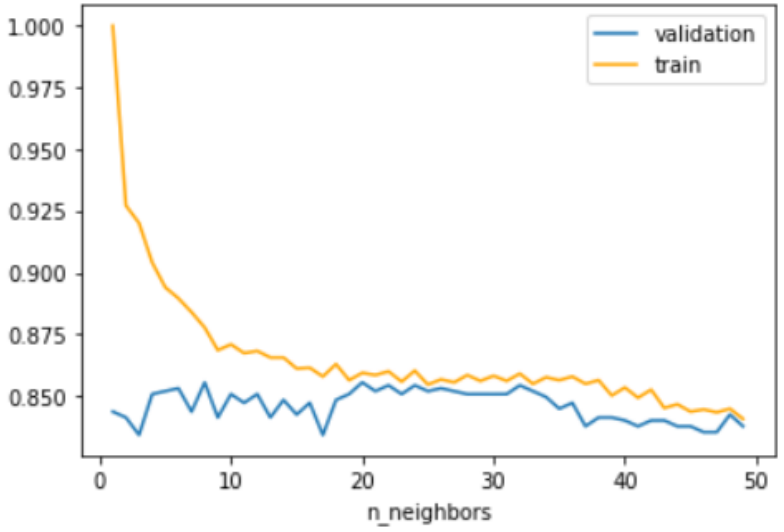


CONCLUSION :

Somewhat satisfactory result

K-NEAREST NEIGHBORS CLASSIFIER

Validation Curve



KNN SCORE

```
cl_knn= KNeighborsClassifier(n_neighbors=20)
cl_knn.fit(X_train_scale,y_train)

print(cl_knn.score(X_train_scale,y_train))
print(cl_knn.score(X_test_scale,y_test))
```

```
0.8649289099526066
0.8483412322274881
```

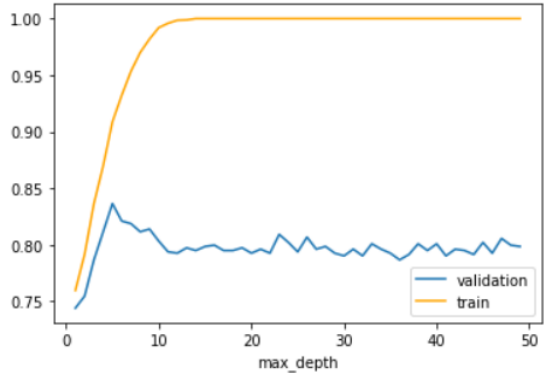
DECISION TREE CLASSIFIER

```
from sklearn.tree import DecisionTreeClassifier

cl_dt= DecisionTreeClassifier()
cl_dt.fit(X_train_scale,y_train)

print(cl_dt.score(X_train_scale,y_train))
print(cl_dt.score(X_test_scale,y_test))
```

```
1.0
0.8246445497630331
```

Parameters optimization allowed us to reduce a lot overfitting and made our accuracy better.

```
model=DecisionTreeClassifier()
k=np.arange(1,50)
train_scores,val_scores=validation_curve(model,X_train_scale,y_train,param_name='max_depth',param_range=k,cv=5)
plt.plot(k,val_scores.mean(axis=1),label='validation')
plt.plot(k,train_scores.mean(axis=1),c='orange',label='train')
plt.xlabel('max_depth')
plt.legend()
plt.show()
```




```
cl_dt= DecisionTreeClassifier(criterion='entropy', max_depth=5, splitter='random')
cl_dt.fit(X_train_scale,y_train)

print(cl_dt.score(X_train_scale,y_train))
print(cl_dt.score(X_test_scale,y_test))
```

```
0.816350710900474
0.8341232227488151
```

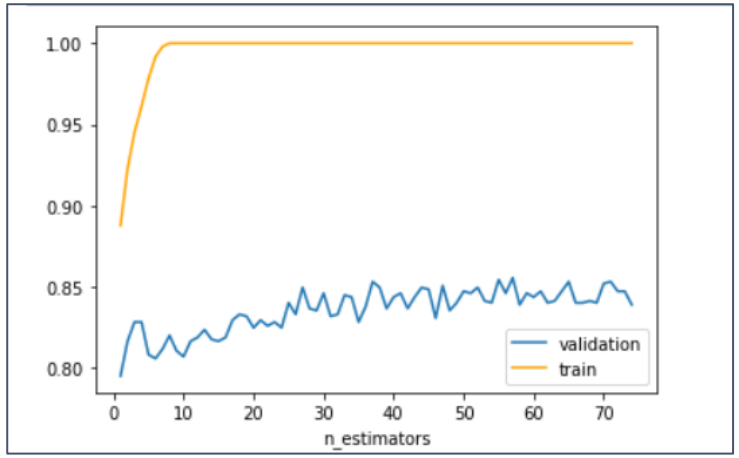
DECISION TREE CLASSIFIER WITH ADABOOSTING

SCORE

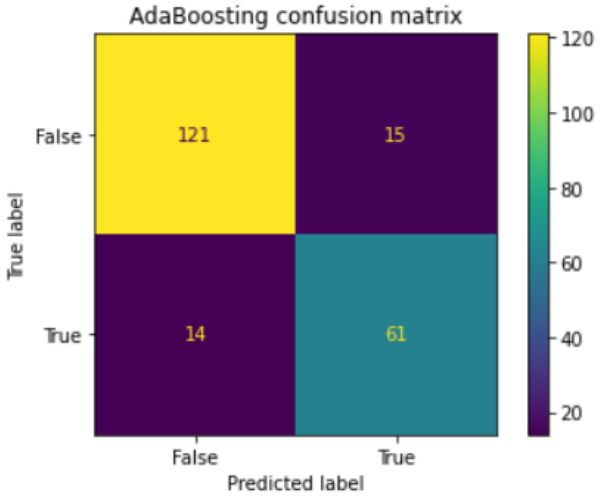
```
cl_ada = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(criteri
cl_ada.fit(X_train_scale,y_train)

print(cl_ada.score(X_train_scale,y_train))
print(cl_ada.score(X_test_scale,y_test))
```

```
0.9834123222748815
0.8672985781990521
```

Not bad, we again have overfitting



PLOTTING CONFUSION MATRIX



We didn’t manage how to reduce overfitting here.

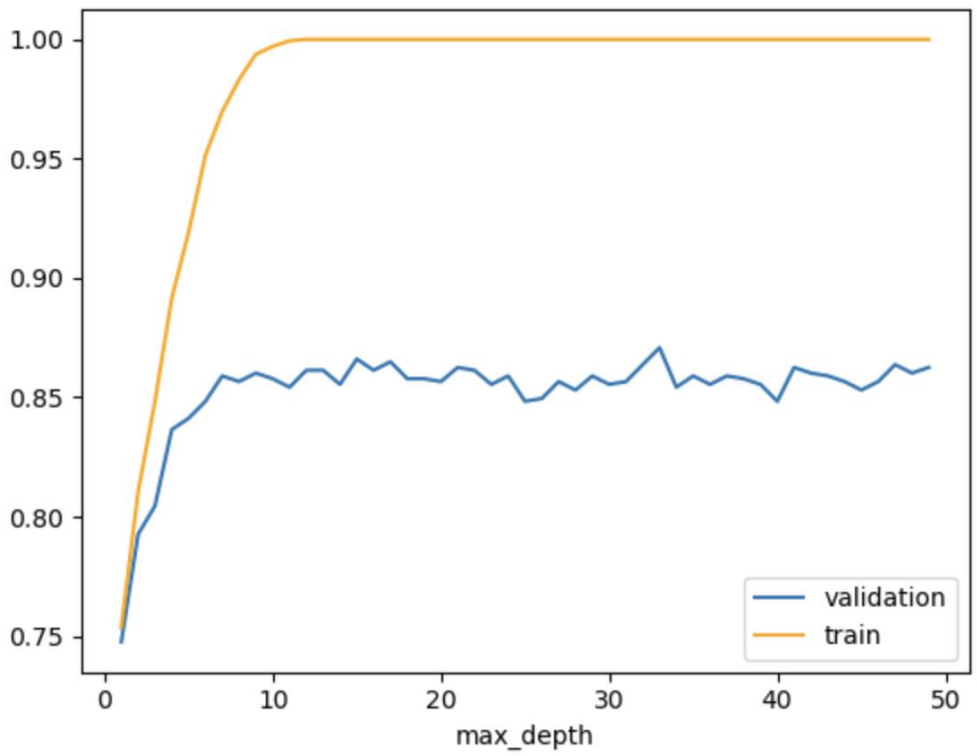
RANDOM FOREST CLASSIFIER

SCORE

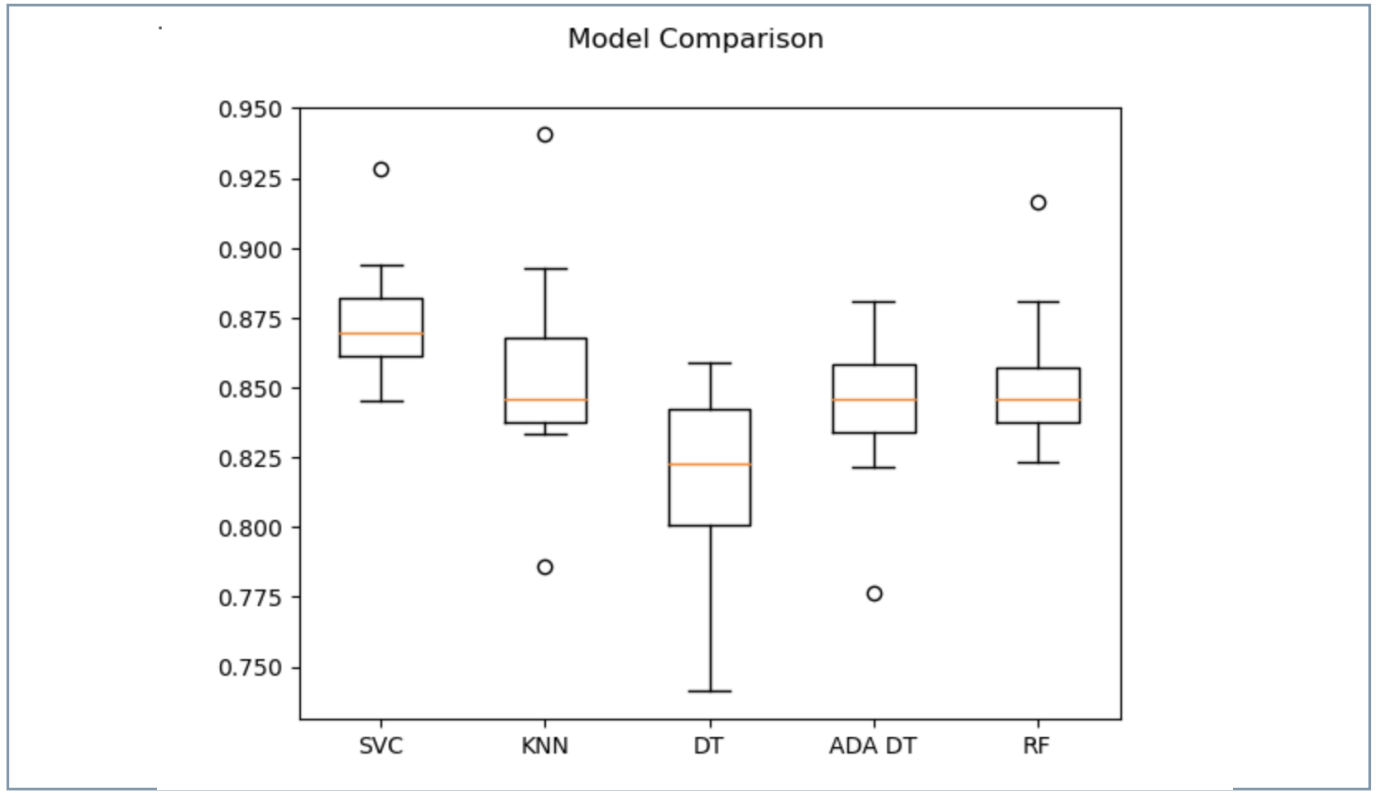
```
cl_rf= RandomForestClassifier()
cl_rf.fit(X_train_scale,y_train)

print(cl_rf.score(X_train_scale,y_train))
print(cl_rf.score(X_test_scale,y_test))
```

```
1.0
0.8909952606635071
```

After trying to optimize parameters, we find out that default parameters were the best for our problem.

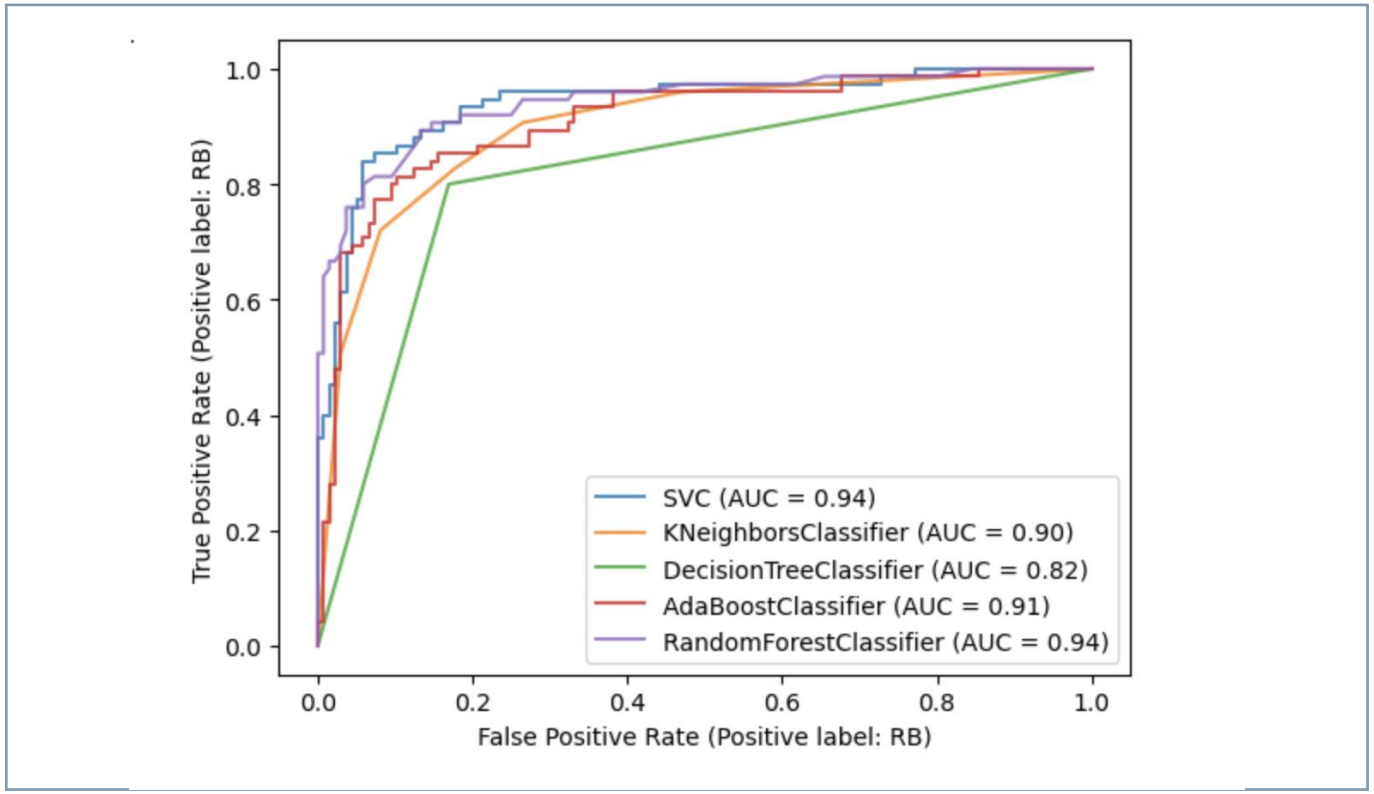


COMPARISON



Accuracy comparison between our models on validation sets

COMPARISON



ROC curve plotting of our models

CHOSEN MODEL

SUPPORT VECTOR MACHINE CLASSIFIER

WHY ?

- Best average validation score
- Best accuracy on test set
- Best ROC curve
- No overfitting

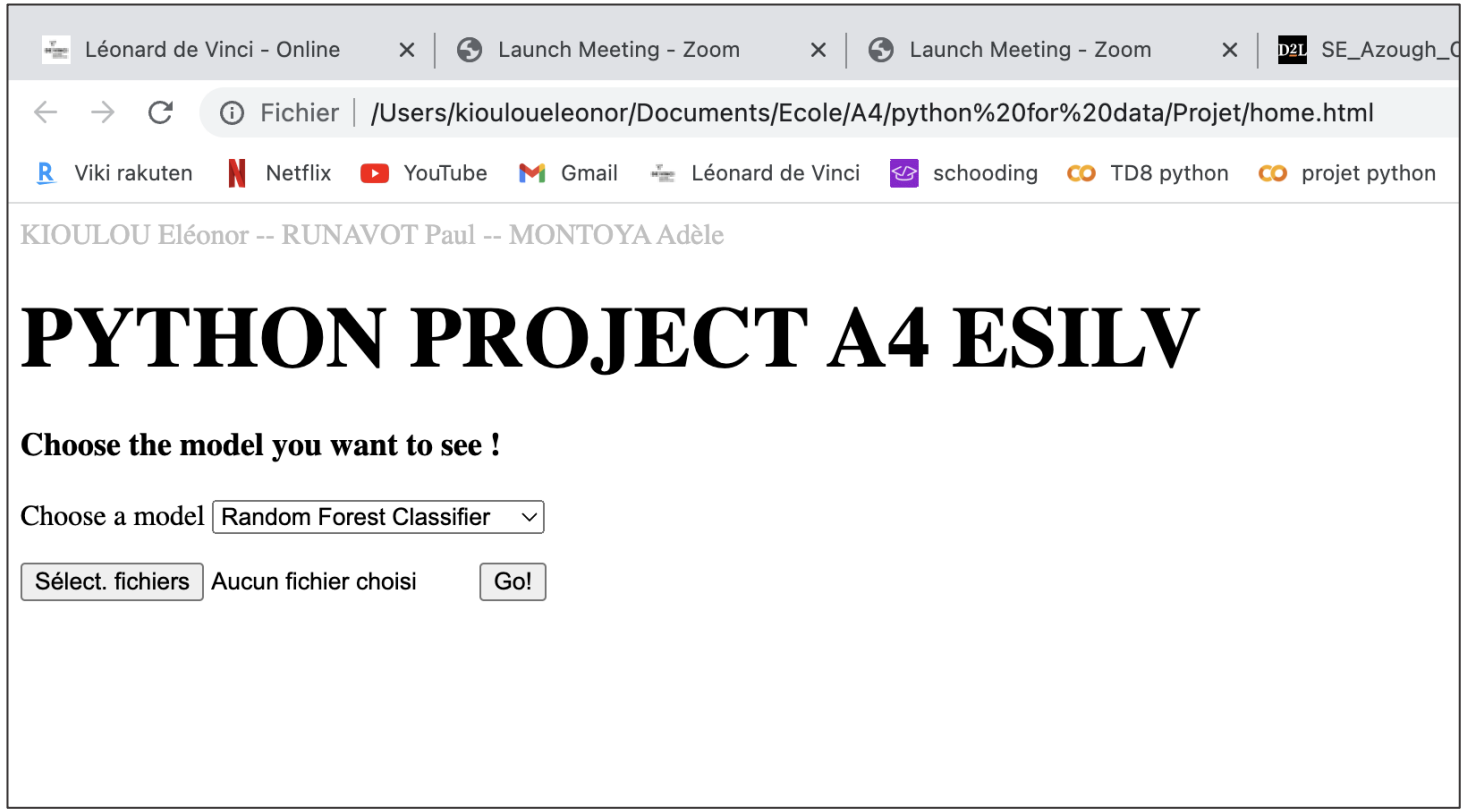
```
cl_svm=svm.SVC(kernel='rbf',C=1,coef0=0.01,degree=1,gamma='auto')
cl_svm.fit(X_train_scale,y_train)

print(cl_svm.score(X_train_scale,y_train))
print(cl_svm.score(X_test_scale,y_test))
```

```
0.9028436018957346
0.8815165876777251
```

4. API

API

RAPPEL API :
Programming interface that provides access to support (data or functionality) provided by a third-party system. It is therefore a matter of making software programs easily dialogue between a service-consuming application and another application that will produce this service.



CONCLUSION

OUR SUBJECT

Thanks to our model, we can say that we can predict whether a molecule is biodegradable or not, by looking at these characteristics.

WHAT WE WOULD HAVE LIKED TO DO MORE

If we had had more time, we would have liked to go deeper into the API part.