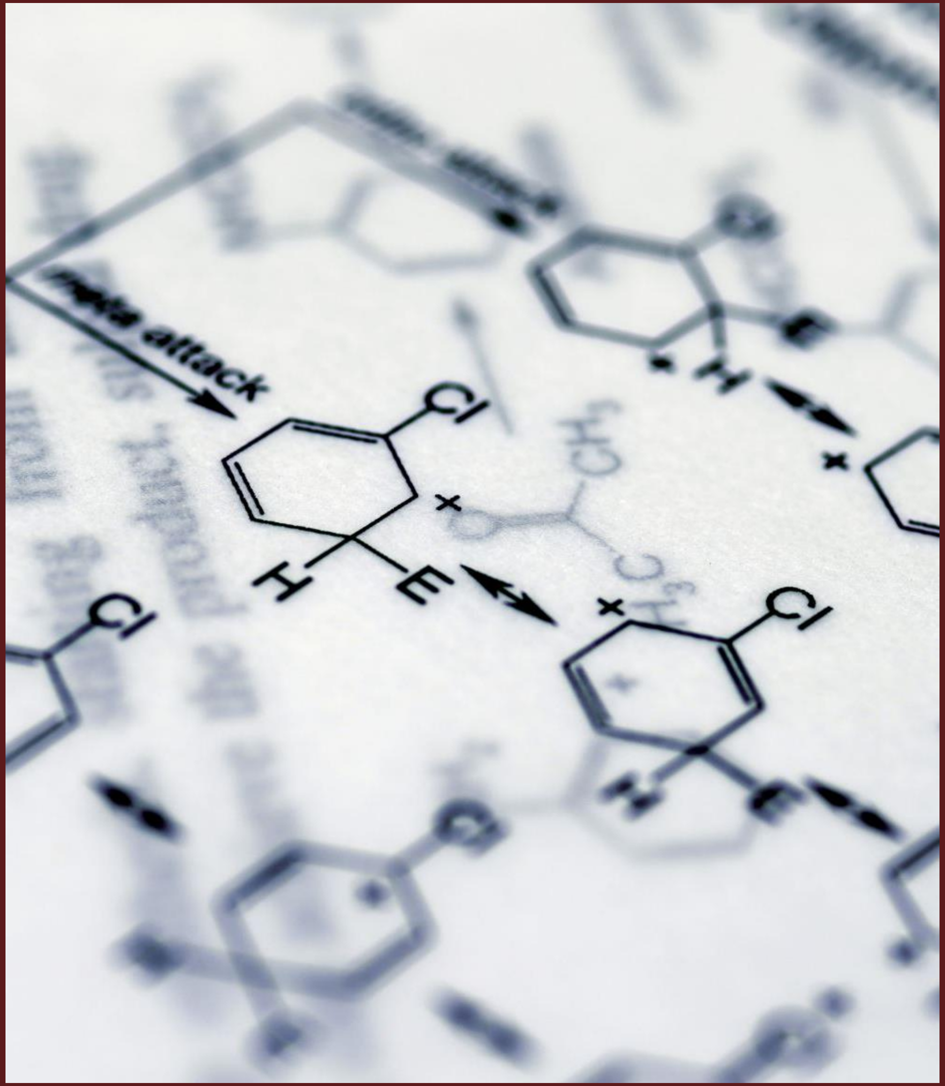
QSAR+BIODEGRADATION

Nader Narcisse



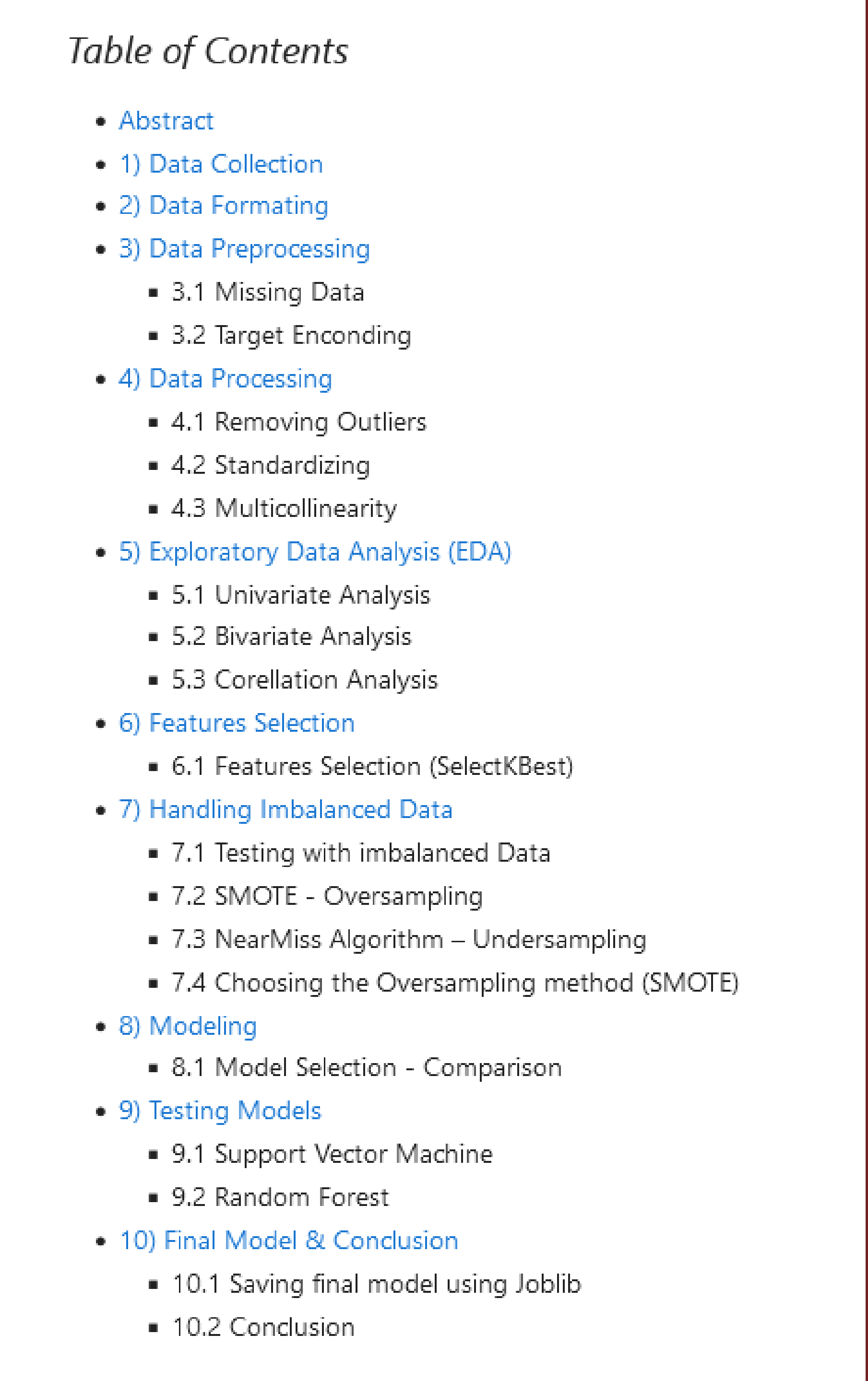


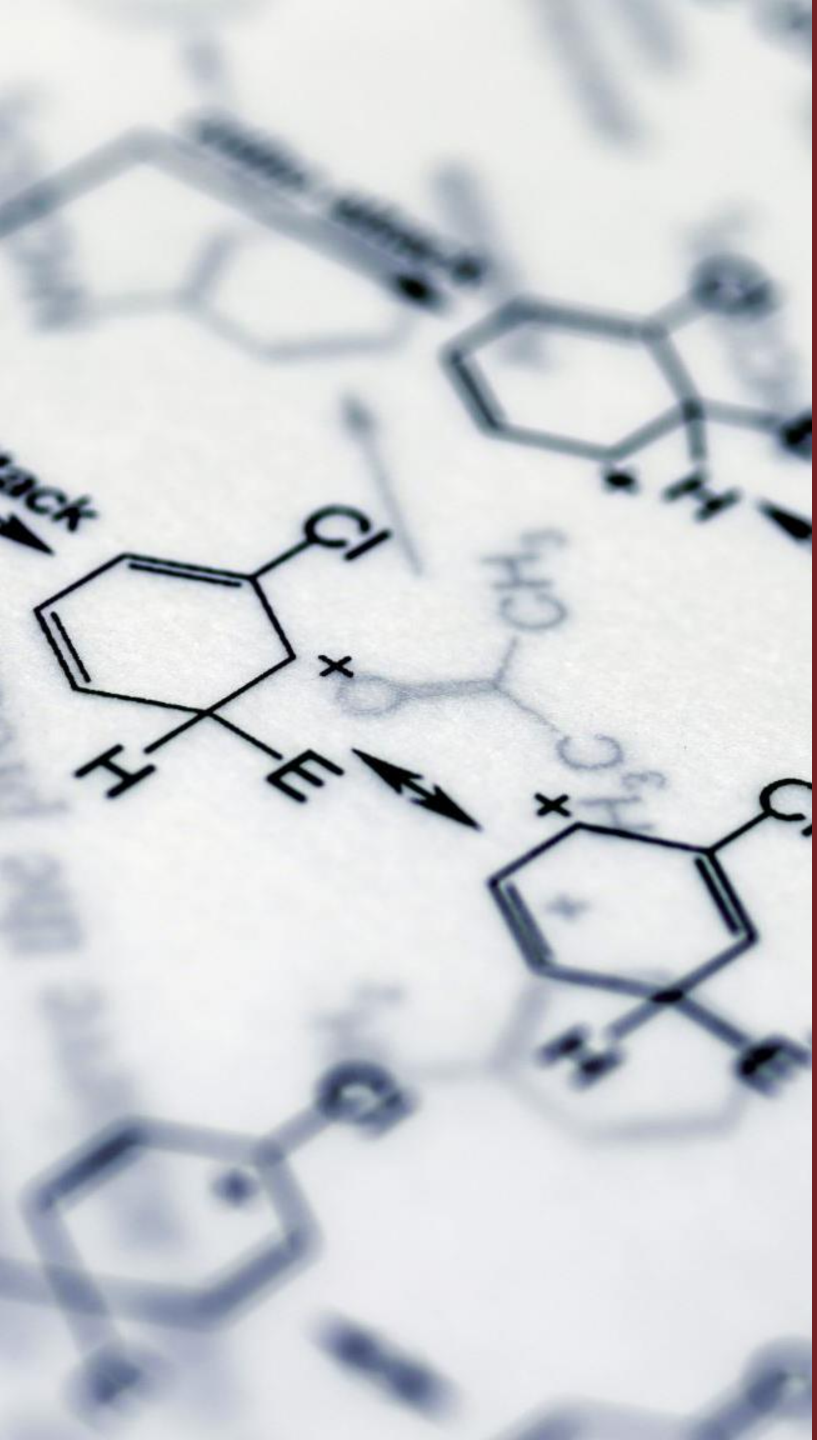
Table of Contents

INTRODUCTION

- Motivation:

School Project from the Data Analysis course in Python, taught as part of the ESILV master's degree.

This presentation is just a support for the notebook explaining the ins and out of the major encounters throughout this project.



ABSTRACT

- The Goal of this project is to **train a reliable model to determine if a substance is biodegradable or not** by using the QSAR biodegradation Data Set provided by UCI.
- Here we are dealing with a **classification problem.** We are trying to predict a discrete value output: *Degradable* and *Not-Degradable.*
- The challenge here personally is that I don't have a chemical engineering background. This project was definitely intriguing for me.

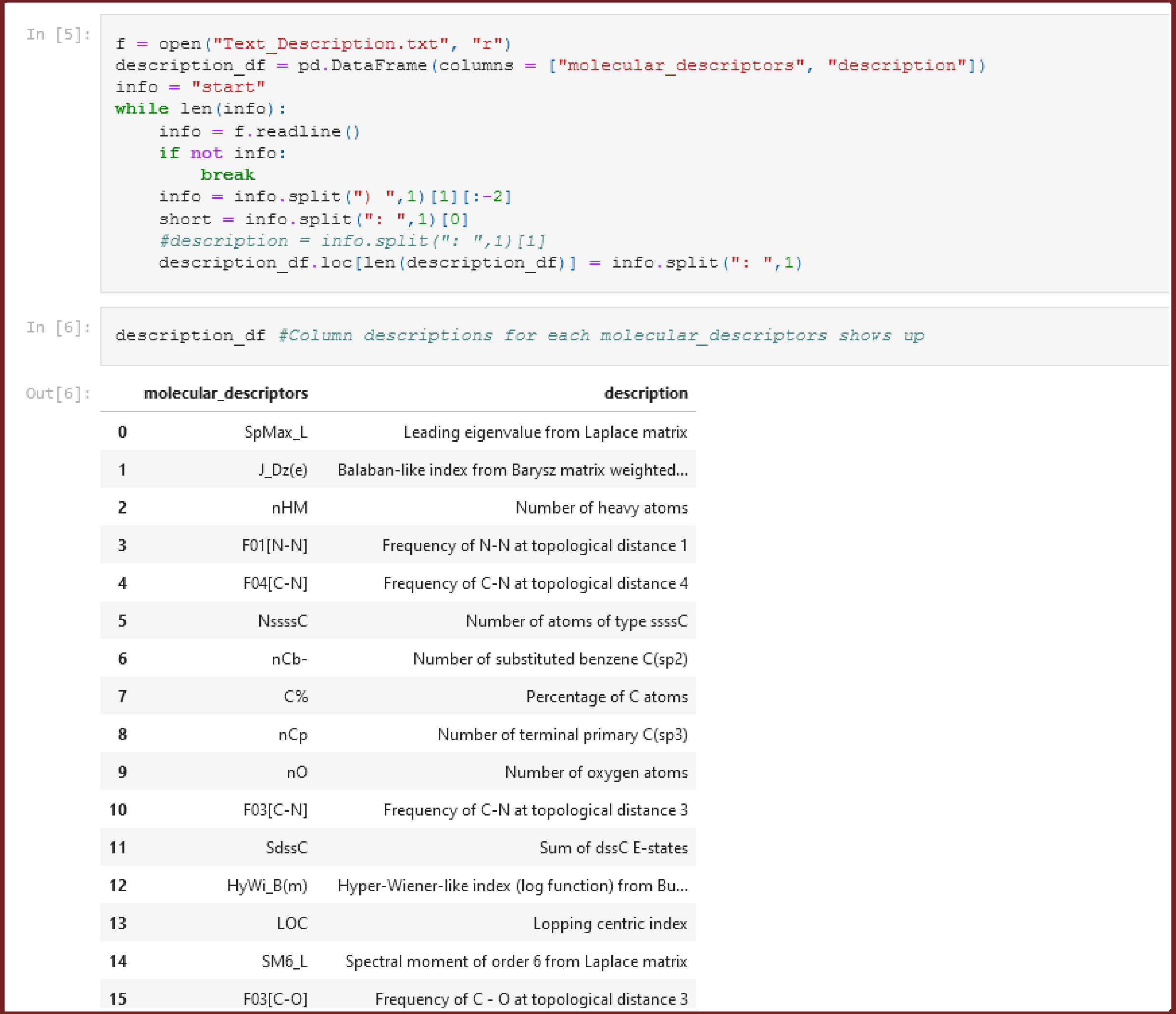
DATA COLLECTION

The QSAR biodegradation Data Set provided by UCI.

DATASET LINK : https://archive.ics.uci.edu/ml/datasets/QSAR+biodegradation

DATA FORMATTING

- I started formatting the data because the DataFrame was missing column descriptions.
- I used the following script that opens the .txt file with the description that has been copied from the UCI web site. Then it was turned into a Panda DataFrame.



```
In [5]:  f = open("Text_Description.txt", "r")
         description_df = pd.DataFrame(columns = ["molecular_descriptors", "description"])
         info = "start"
         while len(info):
             info = f.readline()
             if not info:
                 break
             info = info.split(") ",1)[1][:-2]
             short = info.split(": ",1)[0]
             #description = info.split(": ",1)[1]
             description_df.loc[len(description_df)] = info.split(": ",1)
```

```
In [6]:  description_df #Column descriptions for each molecular_descriptors shows up
```

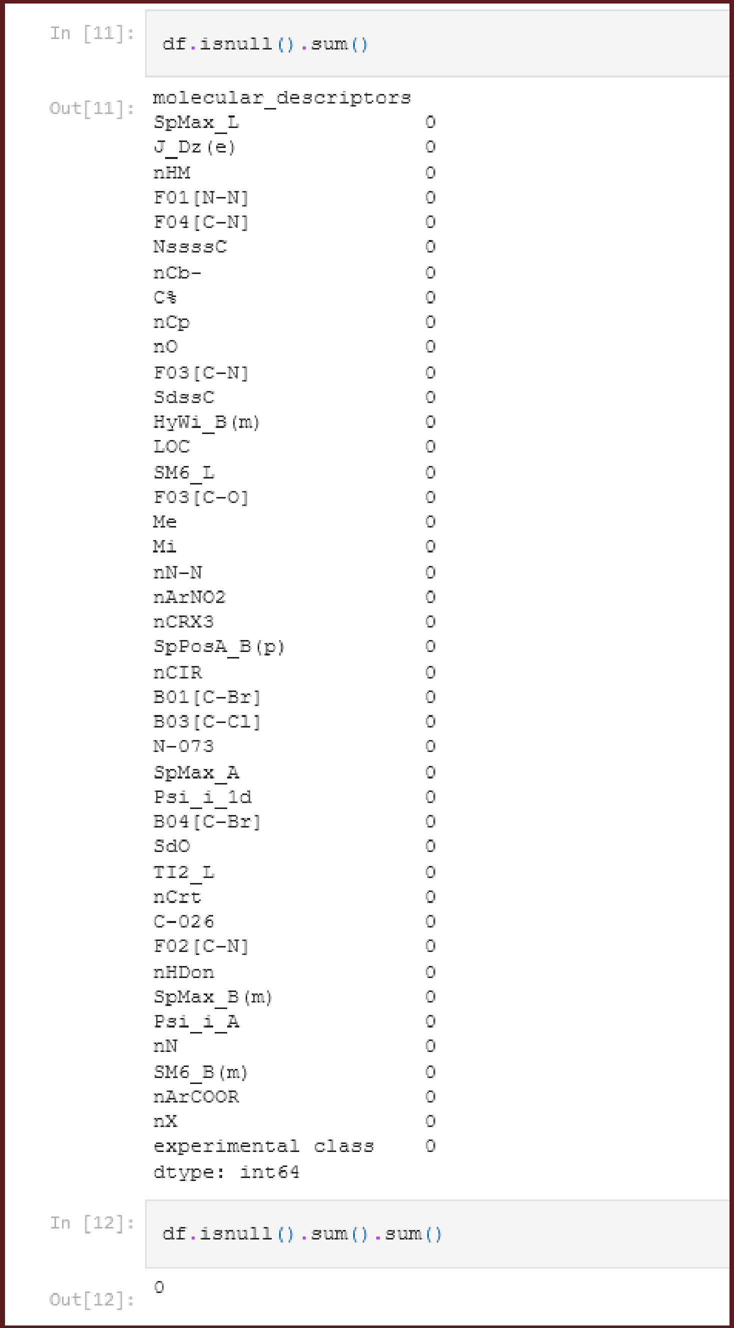
Out[6]:

	molecular_descriptors	description
0	SpMax_L	Leading eigenvalue from Laplace matrix
1	J_Dz(e)	Balaban-like index from Barysz matrix weighted...
2	nHM	Number of heavy atoms
3	F01[N-N]	Frequency of N-N at topological distance 1
4	F04[C-N]	Frequency of C-N at topological distance 4
5	NssssC	Number of atoms of type ssssC
6	nCb-	Number of substituted benzene C(sp2)
7	C%	Percentage of C atoms
8	nCp	Number of terminal primary C(sp3)
9	nO	Number of oxygen atoms
10	F03[C-N]	Frequency of C-N at topological distance 3
11	SdssC	Sum of dssC E-states
12	HyWi_B(m)	Hyper-Wiener-like index (log function) from Bu...
13	LOC	Lopping centric index
14	SM6_L	Spectral moment of order 6 from Laplace matrix
15	F03[C-O]	Frequency of C - O at topological distance 3

DATA PREPROCESSING & PROCESSING

I then made sure we have a clean dataset by :

- - Checking missing Data
- - Target Encoding
- - Removing Outliers
- - Standardizing
- - Checking multicollinearity

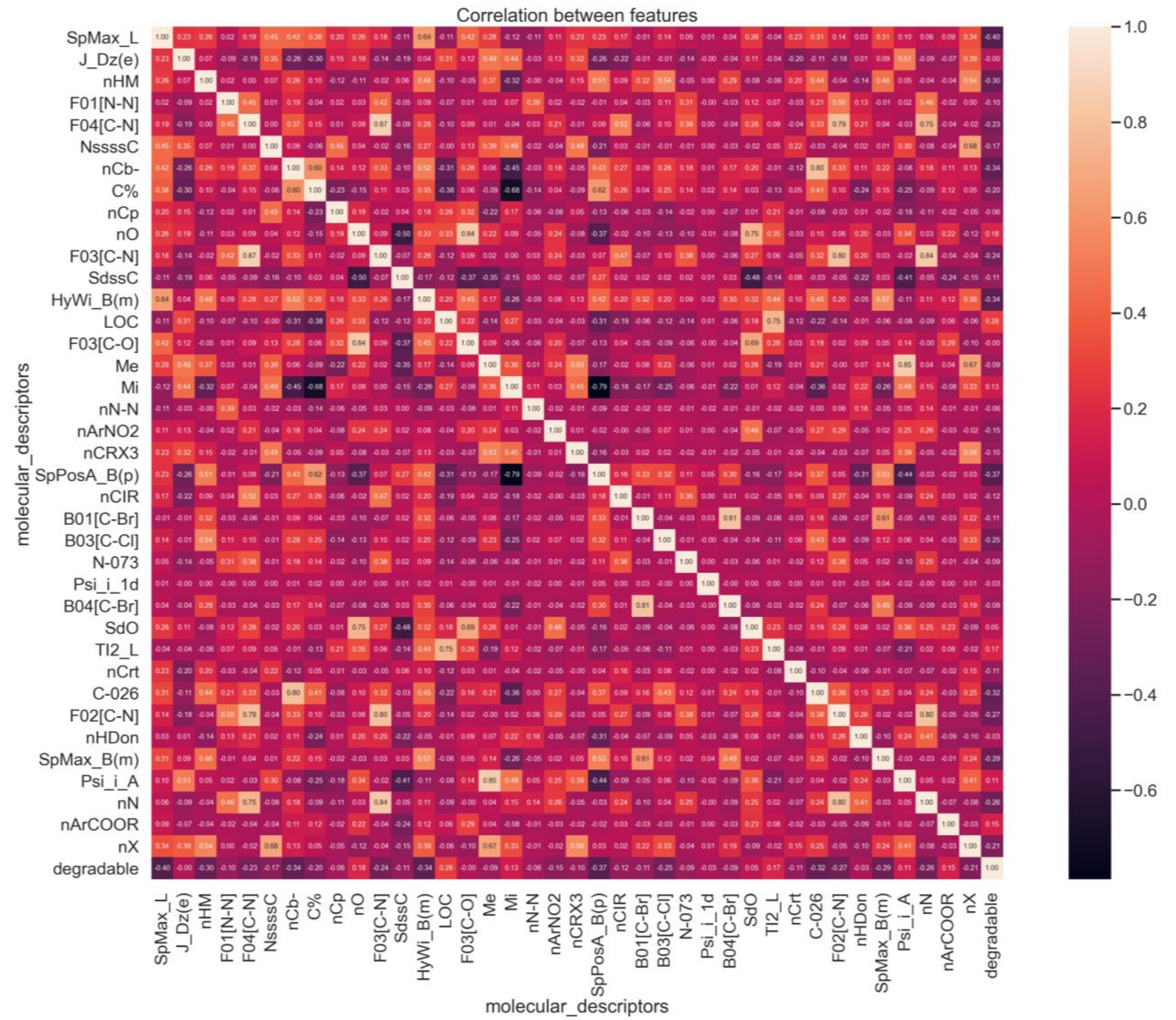


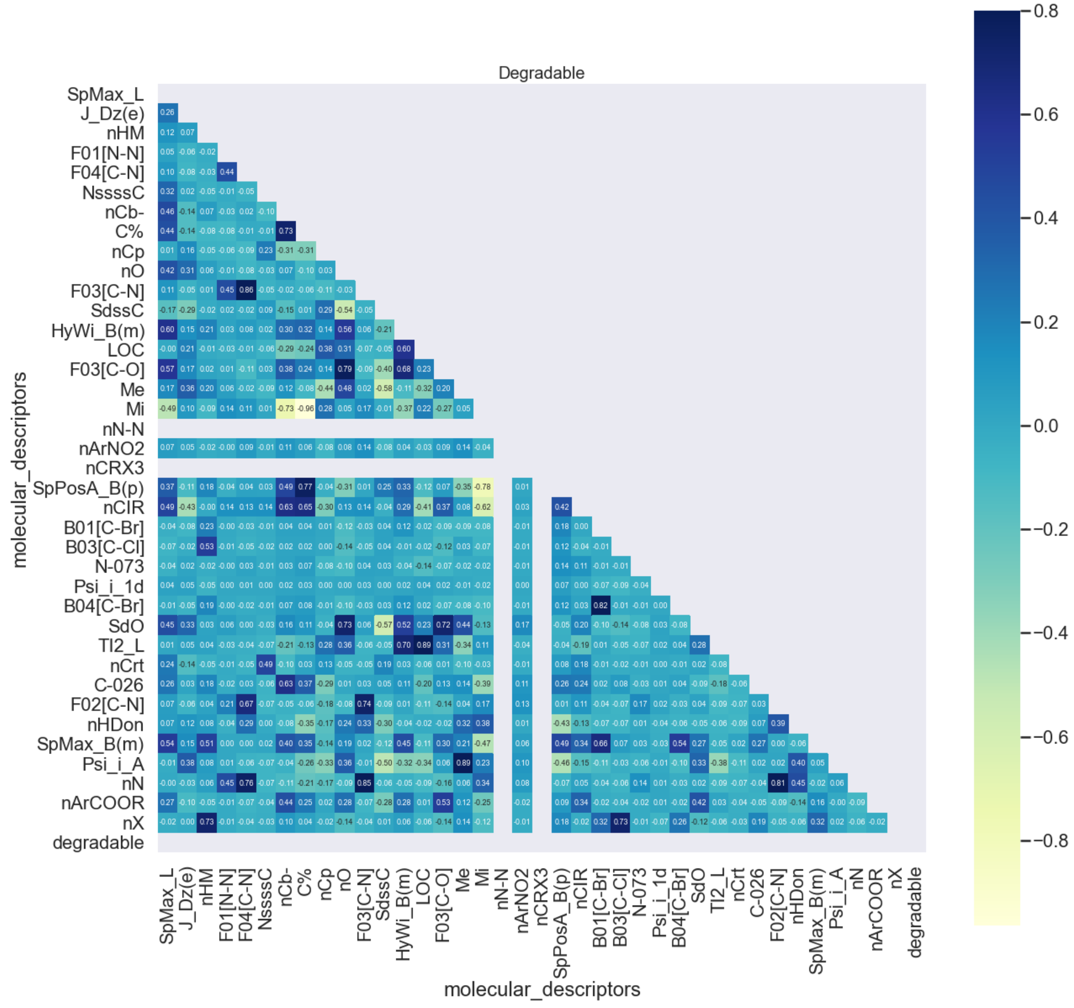
```
In [11]: df.isnull().sum()

Out[11]: molecular_descriptors
         SpMax_L                 0
         J_Dz(e)                 0
         nHM                     0
         F01[N-N]                0
         F04[C-N]                0
         NssssC                  0
         nCb-                    0
         C%                      0
         nCp                     0
         nO                      0
         F03[C-N]                0
         SdssC                   0
         HyWi_B(m)               0
         LOC                     0
         SM6_L                   0
         F03[C-O]                0
         Me                      0
         Mi                      0
         nN-N                    0
         nArNO2                  0
         nCRX3                   0
         SpPosA_B(p)             0
         nCIR                    0
         B01[C-Br]               0
         B03[C-Cl]               0
         N-073                   0
         SpMax_A                 0
         Psi_i_1d                0
         B04[C-Br]               0
         SdO                     0
         TI2_L                   0
         nCrt                    0
         C-026                   0
         F02[C-N]                0
         nHDon                   0
         SpMax_B(m)              0
         Psi_i_A                 0
         nN                      0
         SM6_B(m)                0
         nArCOOR                 0
         nX                      0
         experimental class      0
         dtype: int64

In [12]: df.isnull().sum().sum()

Out[12]: 0
```

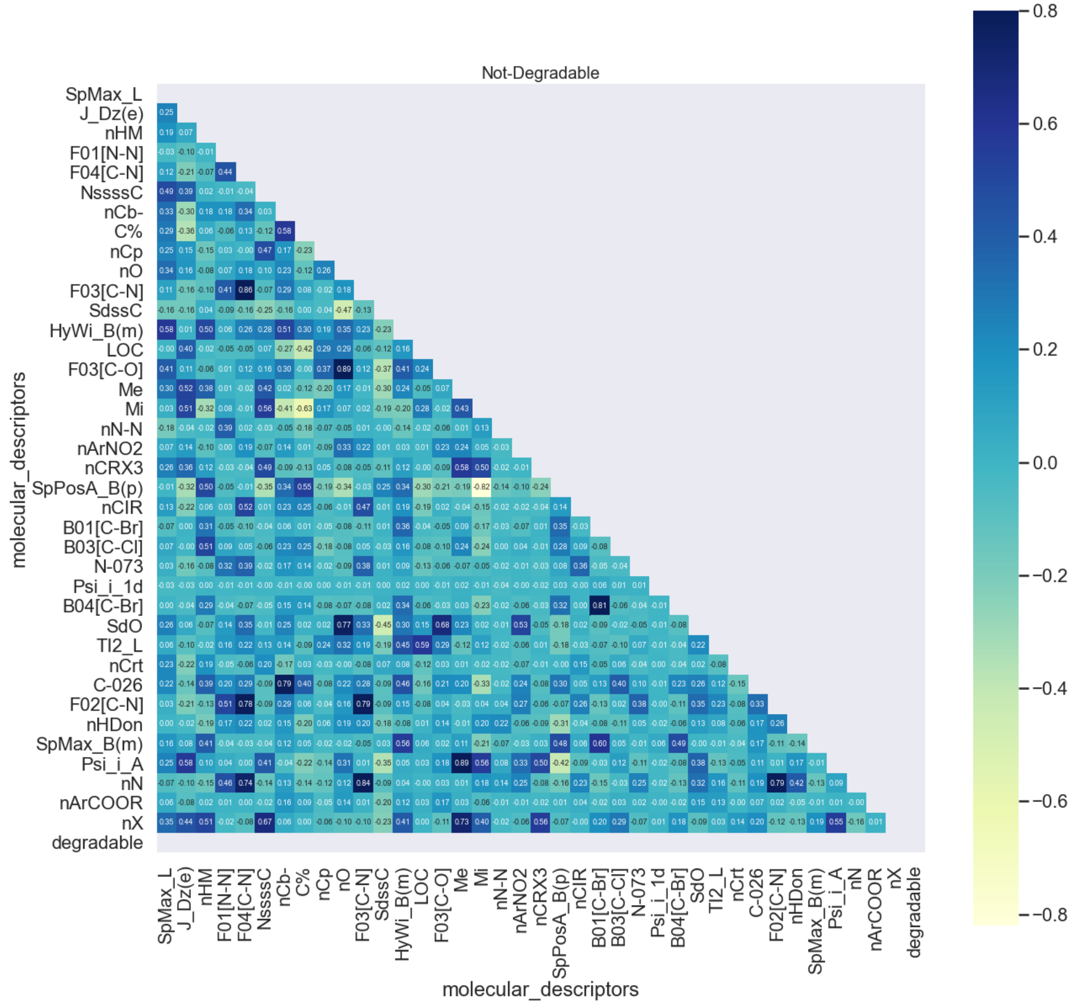
EXPLORATORY DATA ANALYSIS (EDA) *PART-1*

I proceeded with the EDA :

- Univariate Analysis
- Bivariate Analysis
- Correlation Analysis

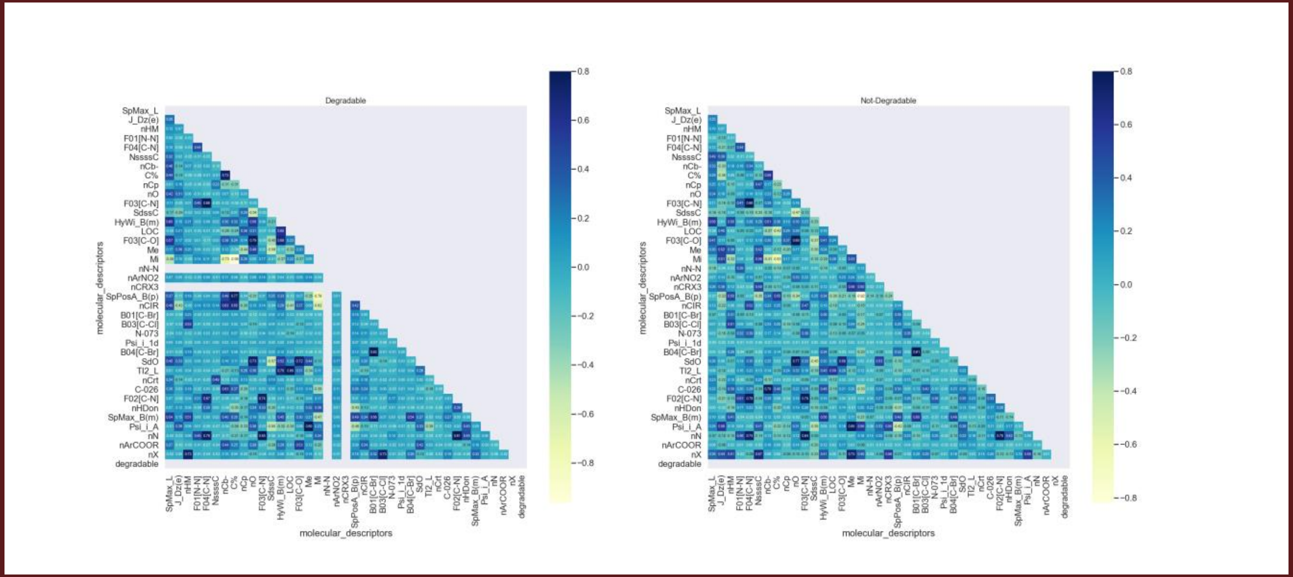



Degradable
molecular_descriptors
molecular_descriptors
SpMax_L
J_Dz(e)
nHM
F01[N-N]
F04[C-N]
NssssC
nCb-
C%
nCp
nO
F03[C-N]
SdssC
HyWi_B(m)
LOC
F03[C-O]
Me
Mi
nN-N
nArNO2
nCRX3
SpPosA_B(p)
nCIR
B01[C-Br]
B03[C-Cl]
N-073
Psi_i_1d
B04[C-Br]
SdO
TI2_L
nCrt
C-026
F02[C-N]
nHDon
SpMax_B(m)
Psi_i_A
nN
nArCOOR
nX
degradable
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8

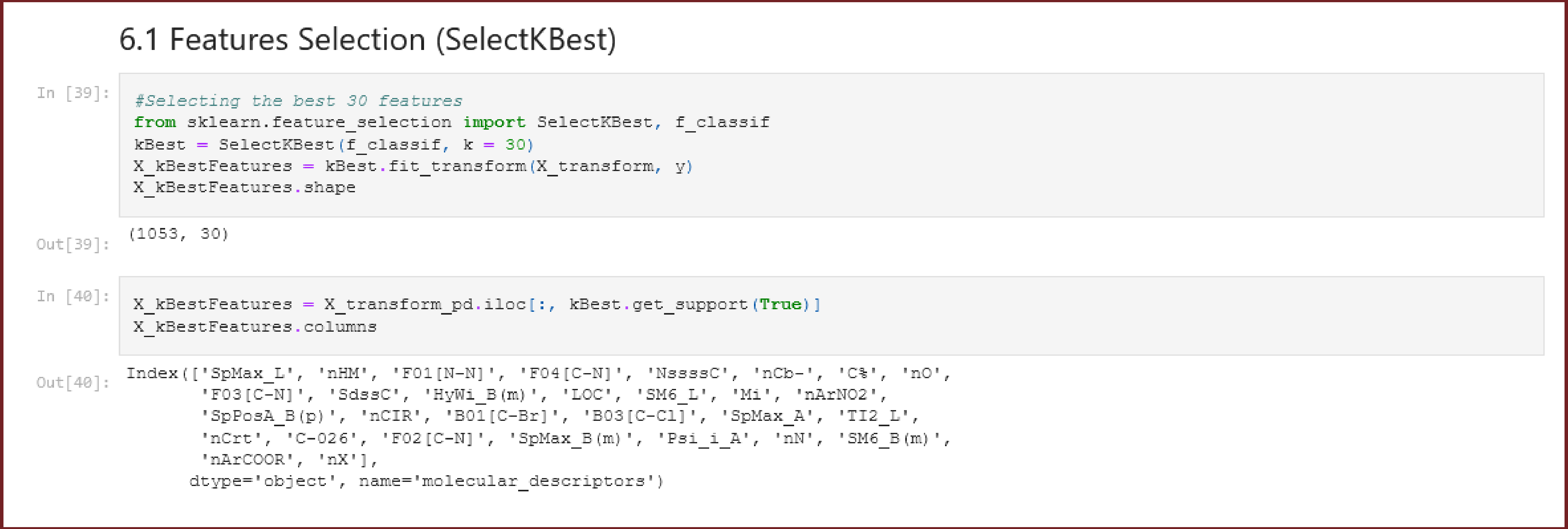

Not-Degradable
molecular_descriptors
molecular_descriptors
SpMax_L
J_Dz(e)
nHM
F01[N-N]
F04[C-N]
NssssC
nCb-
C%
nCp
nO
F03[C-N]
SdssC
HyWi_B(m)
LOC
F03[C-O]
Me
Mi
nN-N
nArNO2
nCRX3
SpPosA_B(p)
nCIR
B01[C-Br]
B03[C-Cl]
N-073
Psi_i_1d
B04[C-Br]
SdO
TI2_L
nCrt
C-026
F02[C-N]
nHDon
SpMax_B(m)
Psi_i_A
nN
nArCOOR
nX
degradable
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8

EXPLORATORY DATA ANALYSIS (EDA) *PART-2*

- If there is no correlation between two variables, it means that the variables do not appear to be statistically related, that the value of one variable doesn't increase or decrease in association with the increase or decrease of the other variable.
- Here we see that nN-N and nCRX3 are two variables not correlated with the other variables for the Degradable class.



FEATURES SELECTION



6.1 Features Selection (SelectKBest)

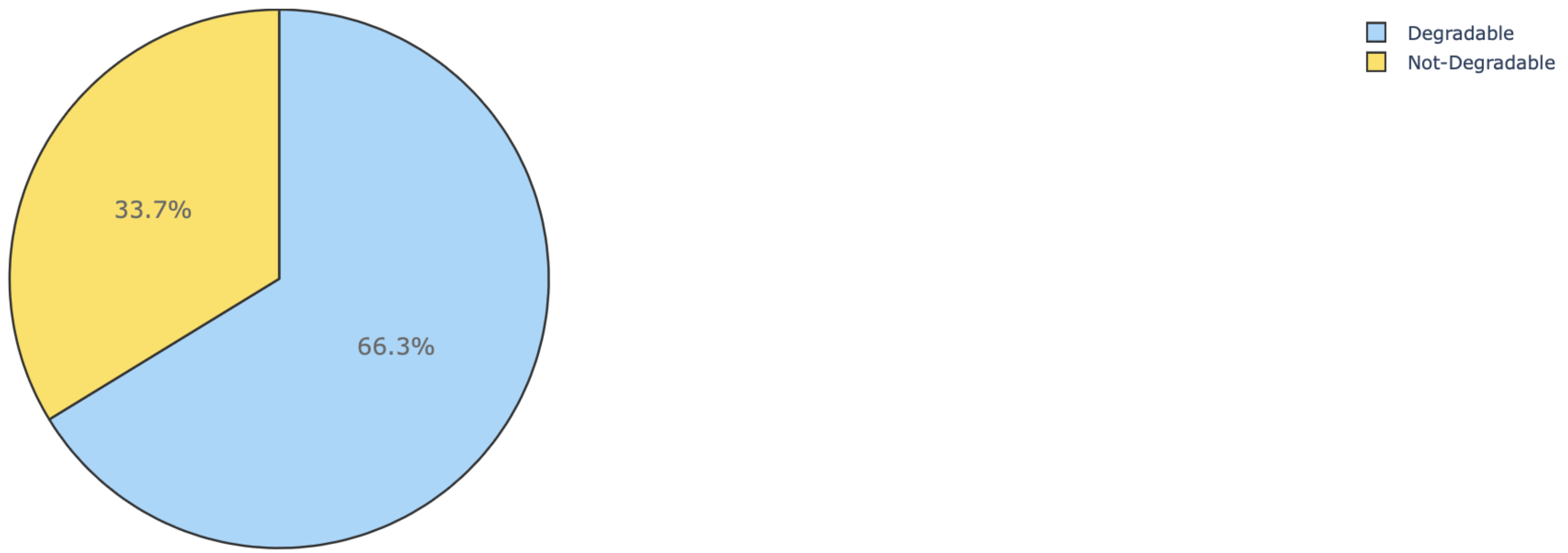
```
In [39]: #Selecting the best 30 features
         from sklearn.feature_selection import SelectKBest, f_classif
         kBest = SelectKBest(f_classif, k = 30)
         X_kBestFeatures = kBest.fit_transform(X_transform, y)
         X_kBestFeatures.shape

Out[39]: (1053, 30)
```

```
In [40]: X_kBestFeatures = X_transform_pd.iloc[:, kBest.get_support(True)]
         X_kBestFeatures.columns

Out[40]: Index(['SpMax_L', 'nHM', 'F01[N-N]', 'F04[C-N]', 'NssssC', 'nCb-', 'C%', 'nO',
                'F03[C-N]', 'SdssC', 'HyWi_B(m)', 'LOC', 'SM6_L', 'Mi', 'nArNO2',
                'SpPosA_B(p)', 'nCIR', 'B01[C-Br]', 'B03[C-Cl]', 'SpMax_A', 'TI2_L',
                'nCrt', 'C-026', 'F02[C-N]', 'SpMax_B(m)', 'Psi_i_A', 'nN', 'SM6_B(m)',
                'nArCOOR', 'nX'],
              dtype='object', name='molecular_descriptors')
```

HANDLING IMBALANCED DATA

- We can see almost 2/3 of them are non-ready degradable while the rest of 1/3 of them are ready degradable.
- There is a significant class imbalance. Class imbalance will lead to a bias towards the majority class. In this case we will perform an oversampling or undersampling method to equalize the data and choose one of them.

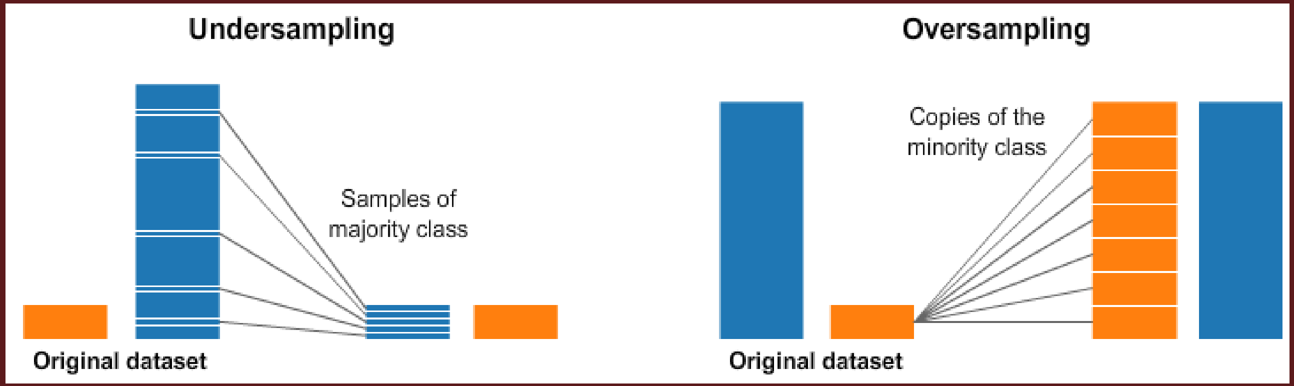
Distribution of target variable



HANDLING IMBALANCED DATA

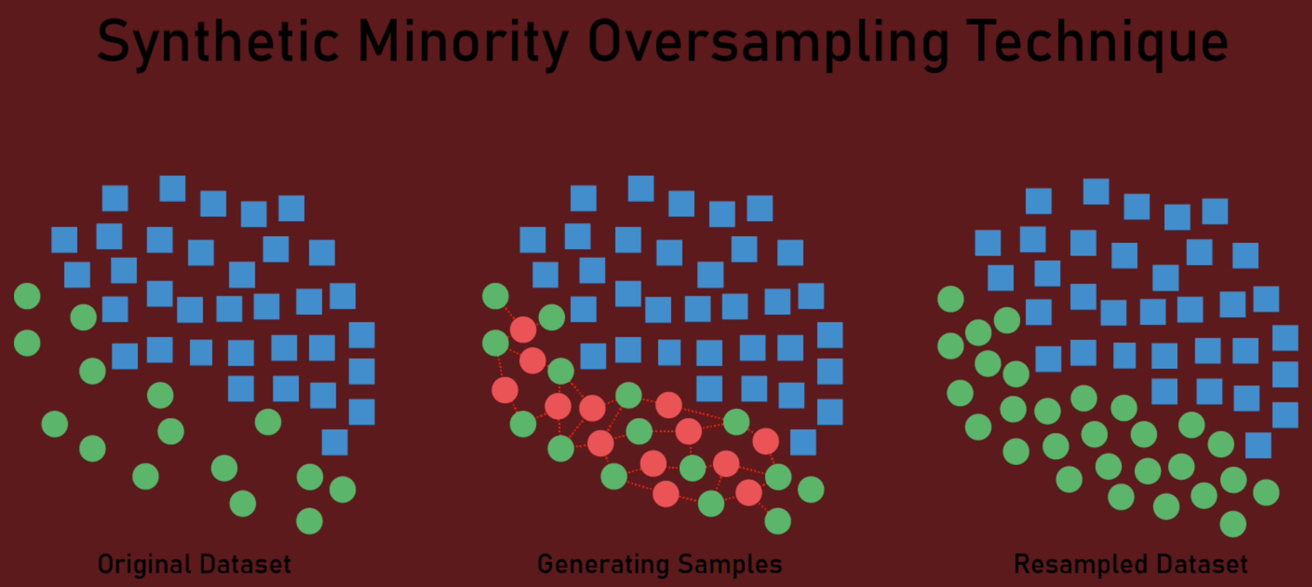
Why are we doing this ?

- With a greater imbalanced ratio, the decision function favor the class with the larger number of samples, usually referred as the majority class. Roughly speaking, weight of class begins depending on count samples.
- Also, we can't use some metrics, like accuracy, if we have disproportion of samples. The objective here is to try to get the best accuracy on a reliable model.



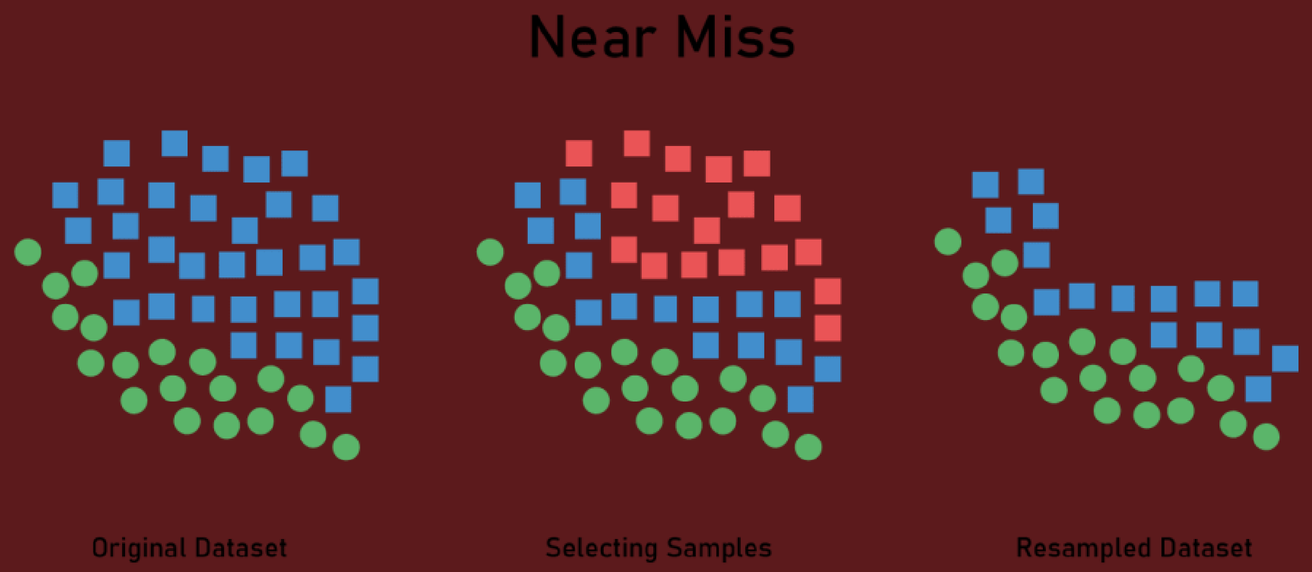
HANDLING IMBALANCED DATA
-
SMOTE - OVERSAMPLING

- Synthetic Minority Oversampling Technique (SMOTE) is an intelligent alternative to oversampling: rather than creating duplicates of the minority class, it creates synthetic data points that are relatively similar to the original ones. Using SMOTE, the model start detected more cases of the minority class, which will result in an increased recall, but a decreased precision.



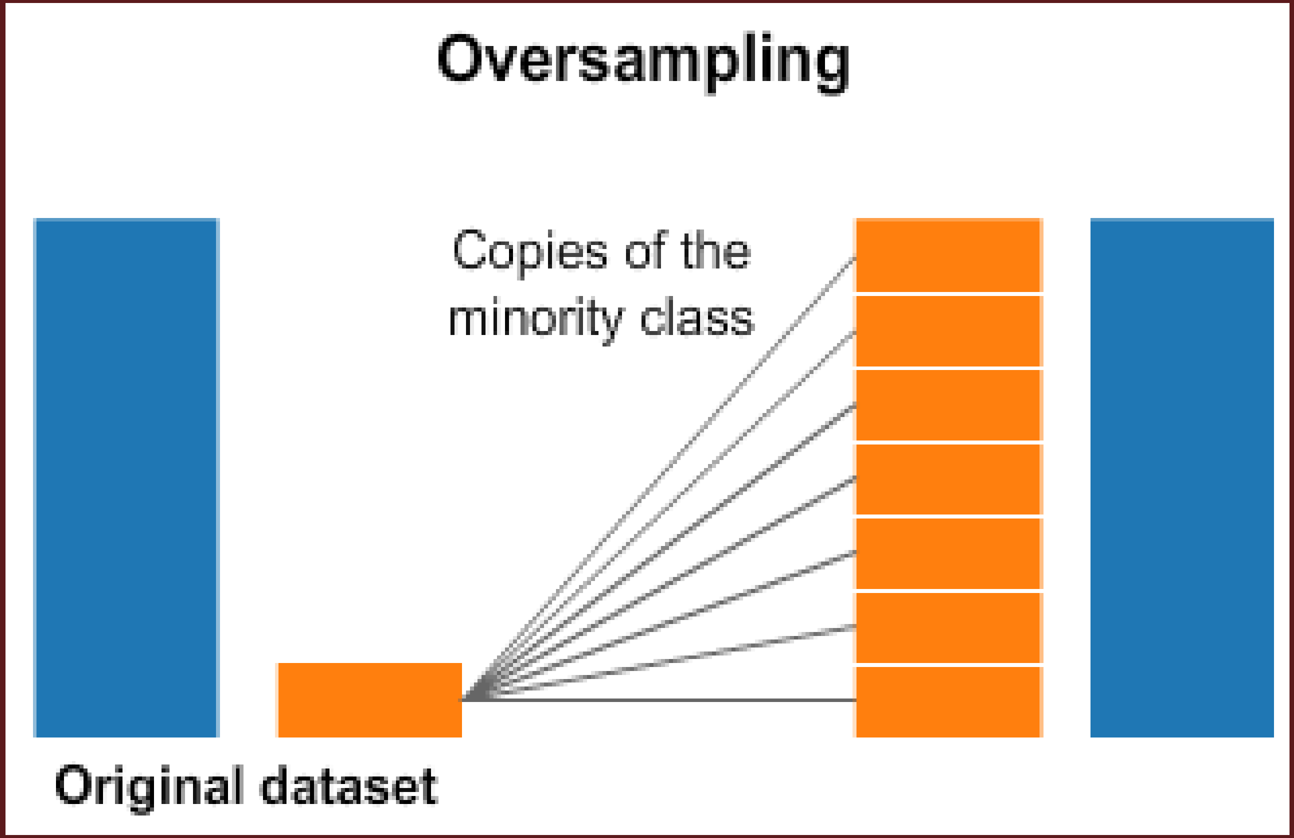
HANDLING IMBALANCED DATA - NEARMISS ALGORITHM - UNDERSAMPLING

- Near-miss is an algorithm that can help in balancing an imbalanced dataset. It can be grouped under undersampling algorithms and is an efficient way to balance the data. The algorithm does this by looking at the class distribution and randomly eliminating samples from the larger class.



HANDLING IMBALANCED DATA
-
CHOOSING THE OVERSAMPLING METHOD SMOTE

- Since undersampling may discard the useful information which could be important for building good classifiers. I went with the Oversampling method with SMOTE.
- Undersampling gets you less data, and most classifiers' performance suffers with less data.

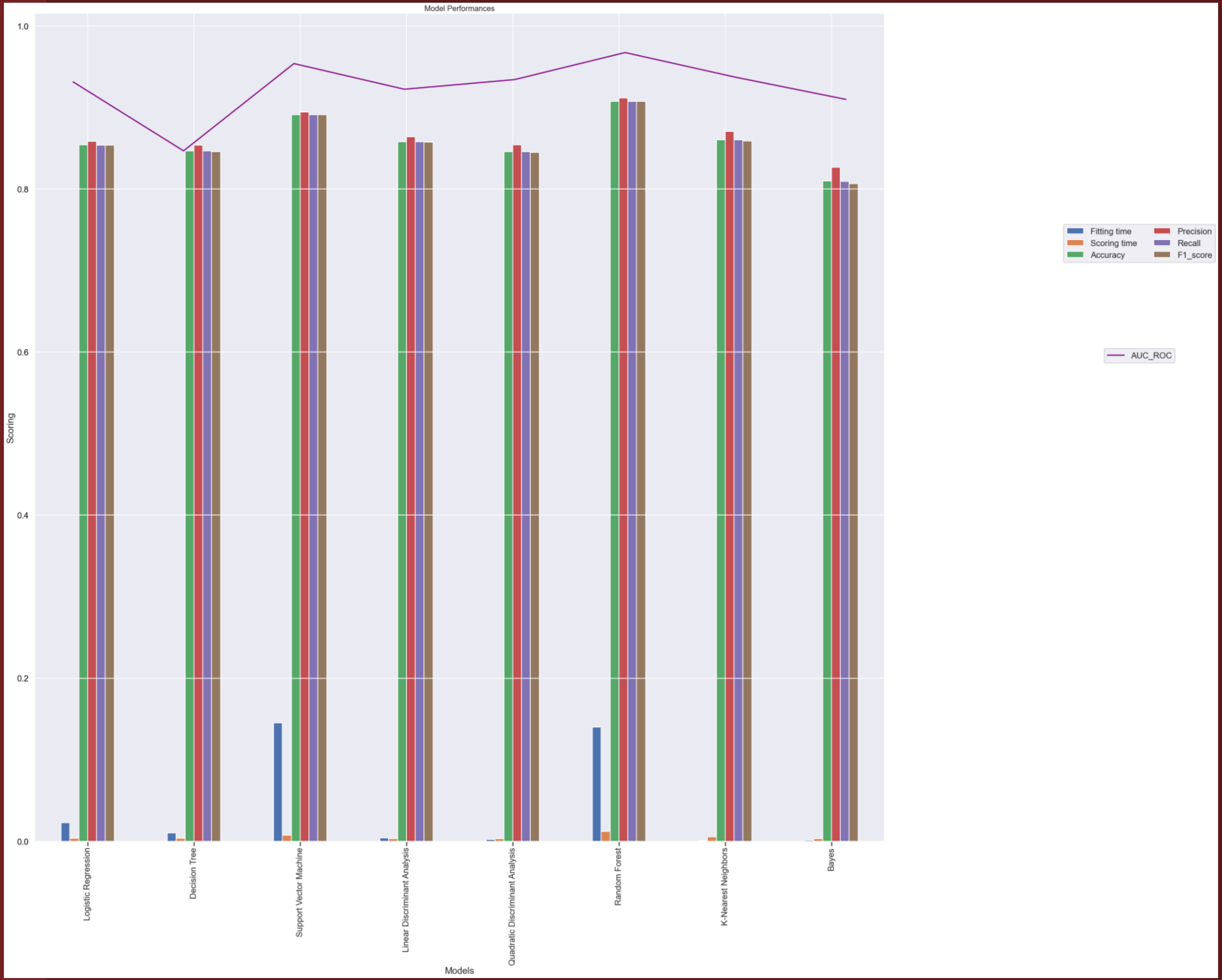


HANDLING IMBALANCED DATA - IMPORTANT NOTE

- I first used SMOTE and then split the data into train test split. The results were enthusiastic, the confusion matrix and classification report were pleasing. As I understood the problem with this approach is that the new synthetically created observations from the minority class in the training dataset might end up in the testing dataset. This in a way allows the algorithm to "cheat" since it learned from something similar and now is testing on almost very similar data points.
- Which is understandable because if I first used SMOTE and then split the data into train test split is that the original sample in training (testing) and the synthetic sample (that was created based on this original sample) in the test (training) set.
- When the model is in production, it's predicting on unseen data. The main point of model validation is to estimate how the model will generalize to new data. If the decision to put a model into production is based on how it performs on a validation set, it's critical that oversampling is done correctly.
- *To summarize:* When you use any sampling technique you divide your data first and then apply synthetic sampling on the training data only. After you do the training, you use the test set which contains only original samples to evaluate.

MODEL SELECTION - COMPARISON

- As we can see in overall performance, the *Random Forest* and *Support Vector Machine* models are the best ones in this selection process, in the next step we will be using them for testing.
- Note :
- We are not necessarily looking for the best fit time but we see that the 2 best models in terms of performance are the ones with a bigger fit time.

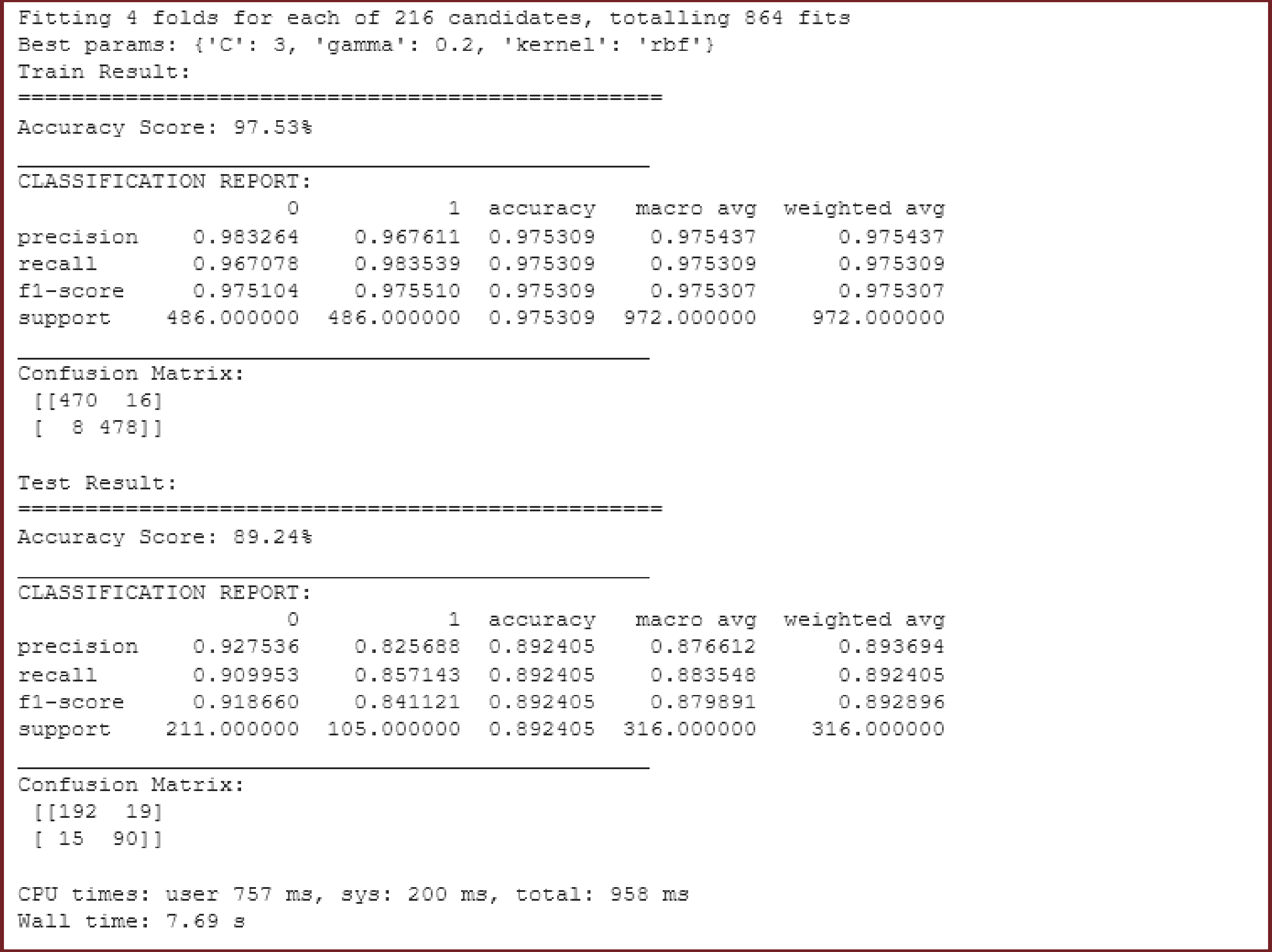


TESTING MODELS

- In this process, we modified the hyperparameter of each models to see how it performed by using GridSearchCV()
- I am satisfied by the overall result of the Support Vector Machine (SVM) model with an accuracy of 89.24%

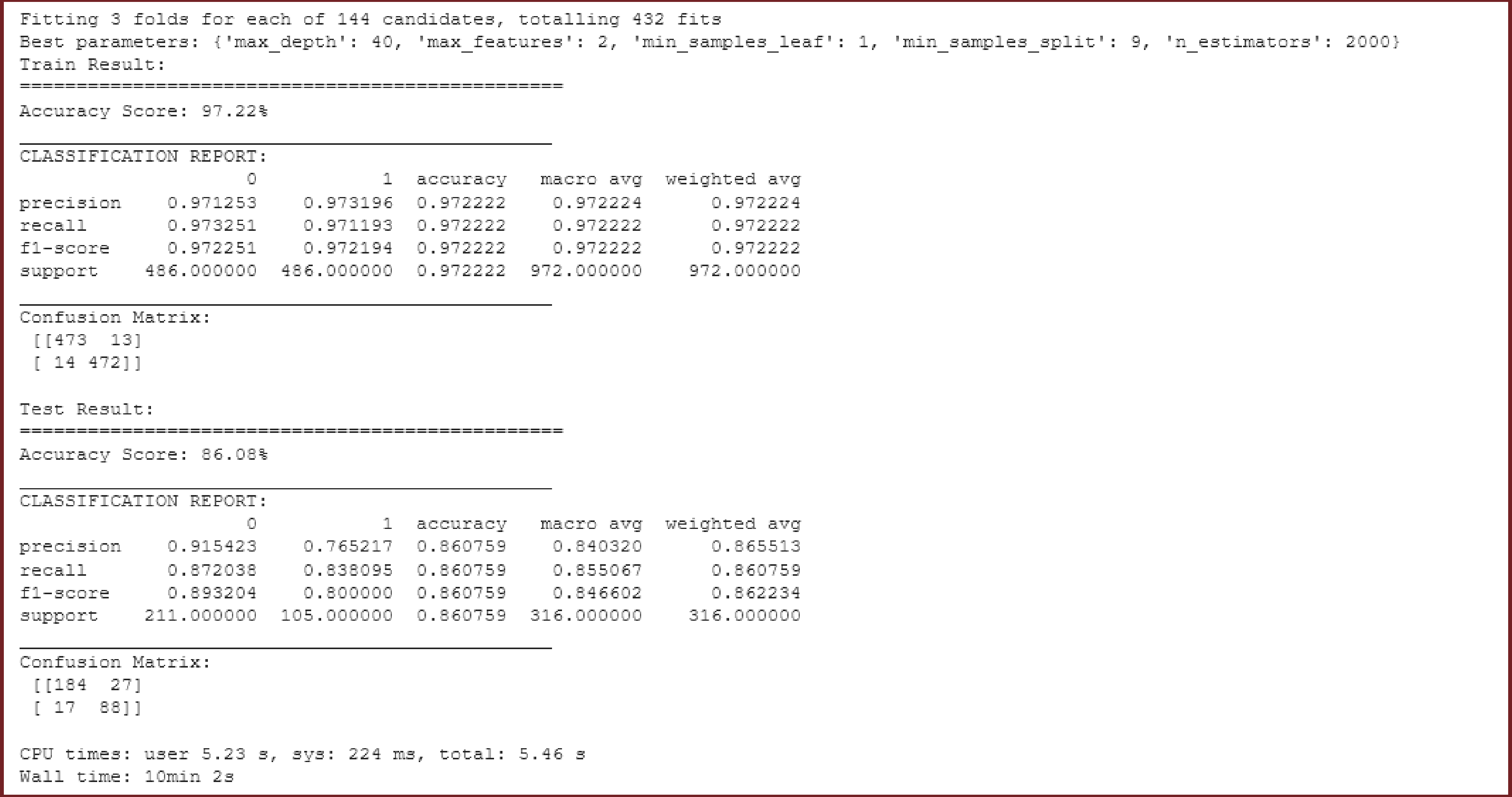
- Note :
- We need to check the accuracy difference between train and test set for each fold result. If my model gives me high training accuracy but low test accuracy my model is overfitting. On the other hand, if it does not give good training accuracy, my model is underfitting.

TESTING RESULTS - SUPPORT VECTOR MACHINE



```
Fitting 4 folds for each of 216 candidates, totalling 864 fits
Best params: {'C': 3, 'gamma': 0.2, 'kernel': 'rbf'}
Train Result:
================================================
Accuracy Score: 97.53%
_______________________________________________
CLASSIFICATION REPORT:
                    0           1  accuracy   macro avg  weighted avg
precision    0.983264    0.967611  0.975309    0.975437      0.975437
recall       0.967078    0.983539  0.975309    0.975309      0.975309
f1-score     0.975104    0.975510  0.975309    0.975307      0.975307
support    486.000000  486.000000  0.975309  972.000000    972.000000
_______________________________________________
Confusion Matrix:
 [[470  16]
 [  8 478]]

Test Result:
================================================
Accuracy Score: 89.24%
_______________________________________________
CLASSIFICATION REPORT:
                    0           1  accuracy   macro avg  weighted avg
precision    0.927536    0.825688  0.892405    0.876612      0.893694
recall       0.909953    0.857143  0.892405    0.883548      0.892405
f1-score     0.918660    0.841121  0.892405    0.879891      0.892896
support    211.000000  105.000000  0.892405  316.000000    316.000000
_______________________________________________
Confusion Matrix:
 [[192  19]
 [ 15  90]]

CPU times: user 757 ms, sys: 200 ms, total: 958 ms
Wall time: 7.69 s
```

TESTING RESULTS - RANDOM FOREST



```
Fitting 3 folds for each of 144 candidates, totalling 432 fits
Best parameters: {'max_depth': 40, 'max_features': 2, 'min_samples_leaf': 1, 'min_samples_split': 9, 'n_estimators': 2000}
Train Result:
================================================
Accuracy Score: 97.22%
_______________________________________________
CLASSIFICATION REPORT:
                    0           1  accuracy   macro avg  weighted avg
precision    0.971253    0.973196  0.972222    0.972224      0.972224
recall       0.973251    0.971193  0.972222    0.972222      0.972222
f1-score     0.972251    0.972194  0.972222    0.972222      0.972222
support    486.000000  486.000000  0.972222  972.000000    972.000000
_______________________________________________
Confusion Matrix:
 [[473  13]
 [ 14 472]]

Test Result:
================================================
Accuracy Score: 86.08%
_______________________________________________
CLASSIFICATION REPORT:
                    0           1  accuracy   macro avg  weighted avg
precision    0.915423    0.765217  0.860759    0.840320      0.865513
recall       0.872038    0.838095  0.860759    0.855067      0.860759
f1-score     0.893204    0.800000  0.860759    0.846602      0.862234
support    211.000000  105.000000  0.860759  316.000000    316.000000
_______________________________________________
Confusion Matrix:
 [[184  27]
 [ 17  88]]

CPU times: user 5.23 s, sys: 224 ms, total: 5.46 s
Wall time: 10min 2s
```

CONCLUSION

In this project, our objective was to make sure we have a reliable model that can be used in order to determine if a substance is biodegradable or not.

I conclude that the best type of model for this problem is a Support Vector Machine (SVM) model with an accuracy of **89.24%**

Model is trained on a **balanced class dataset**, using Oversampling with the Synthetic Minority Oversampling Technique (SMOTE)

This model could be used as a Proof of Concept for the use of simulation on compounds to determine whether it is biodegradable or not.