Ecole Centrale Casablanca

Coding Week
Report

Predicting the cellular localization sites of proteins

Authors :
Mohammed El Barhichi
Aymane Mars
Hiba Nokra
Mohamed Ali Akif
Wadiaa Saadaoui
Abderrazzak Outzoula

Supervised by :
Oumayma Ouedrhiri
Kawtar Zerhouni
Oumayma Banouar

Academic Year
2022/2023

Contents

Abstract

To summarize our work, we started by loading our dataset and visualising it to better understand its problems. We found that the main problem is that our database is imbalanced. Therefore, our project is about a **Multi-class Classification** on an **Imbalanced** Dataset. Then, we prepossessed our dataset by re-scaling it.

As it was requested in the problem statement, we worked essentially with two model: **SVM** and **KNN**, and to resolve the imbalance problem, we worked with SMOTE technique.

In our first approach, we split the imbalanced dataset into a **training** part and a **test** one, then we trained the two given models on the train dataset. This approach does not give good results since the models can't learn very well from this data.

In order to see if we can resolve this problem automatically, we tried to add the hyperparameter **'class weight'** = **'balanced'** that allows the SVM model to balance the train data itself. This approach also does not give significant results since the test data still does not include all the classes.

In a second approach, we tried to balance **only** the **training** data using SMOTE method. We found that the accuracy increased, but this still give the same problem of not detecting some classes since we still always don't have samples from all the classes in the test dataset.

Our third approach was to balance **all** the dataset using SMOTE method before splitting it into the training part and the test one. This lead the test data to have samples from all the classes.

Overall the results that we've got from this approach were very interesting, both models perform well. In comparaison between the performance of the SVM and KNN model, we can affirm that **KNN model is better**.

In order to evaluate the performance of the SMOTE technique, we tried to use other data augmentation methods like the **RandomOverSampling** one. We found that this approach does not give good results. Hense, we can maintain working with SMOTE method on our dataset.

To have a further vision on our results, we tried to work with some other models not given in the problem statement. Finally, we found that the **Random Forest** model is the most performing one for our problem.

1 Introduction

1.1 Contextualization

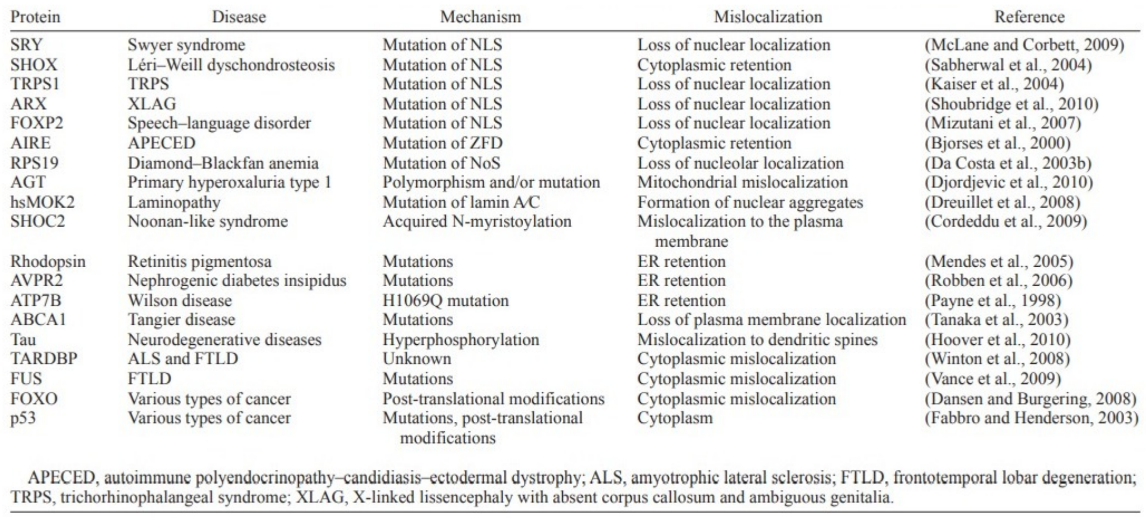
In contemporary times, Artificial Intelligence has undergone a significant evolution, especially with the emergence of ChatGPT and various other AI generators. **Ecole Centrale Casablanca** is committed to following the path of innovation through several programs.

In this regard, the institution organizes a coding week, during which participants engage with diverse computer science topics. The focus of this year's coding week is Machine Learning and its application in various domains, including medicine, with the aim of resolving numerous issues.

1.2 Motivation

Our generation and our continent have suffered greatly from the diseases that have occurred in the last decade. In fact, Illnesses have disrupted our way of life and threatened the lives of many people, including our loved ones, which has sparked great interest in finding a thoughtful and appropriate way to control them.

In this perspective, understanding the cellular localization sites of proteins is a critical aspect of molecular biology research. It provides insight into a protein's function, potential role in disease, and can help identify potential drug targets. Various computational and experimental methods can be used to predict the cellular localization sites of proteins, including sequence-based prediction methods and imaging techniques. Knowing the localization of proteins can provide valuable information for a wide range of research areas, including drug discovery, disease diagnosis, and evolutionary analysis.



Protein	Disease	Mechanism	Mislocalization	Reference
SRY	Swyer syndrome	Mutation of NLS	Loss of nuclear localization	(McLane and Corbett, 2009)
SHOX	Léri–Weill dyschondrosteosis	Mutation of NLS	Cytoplasmic retention	(Sabherwal et al., 2004)
TRPS1	TRPS	Mutation of NLS	Loss of nuclear localization	(Kaiser et al., 2004)
ARX	XLAG	Mutation of NLS	Loss of nuclear localization	(Shoubridge et al., 2010)
FOXP2	Speech–language disorder	Mutation of NLS	Loss of nuclear localization	(Mizutani et al., 2007)
AIRE	APECED	Mutation of ZFD	Cytoplasmic retention	(Bjorses et al., 2000)
RPS19	Diamond–Blackfan anemia	Mutation of NoS	Loss of nucleolar localization	(Da Costa et al., 2003b)
AGT	Primary hyperoxaluria type 1	Polymorphism and/or mutation	Mitochondrial mislocalization	(Djordjevic et al., 2010)
hsMOK2	Laminopathy	Mutation of lamin A/C	Formation of nuclear aggregates	(Dreuillet et al., 2008)
SHOC2	Noonan-like syndrome	Acquired N-myristoylation	Mislocalization to the plasma membrane	(Cordeddu et al., 2009)
Rhodopsin	Retinitis pigmentosa	Mutations	ER retention	(Mendes et al., 2005)
AVPR2	Nephrogenic diabetes insipidus	Mutations	ER retention	(Robben et al., 2006)
ATP7B	Wilson disease	H1069Q mutation	ER retention	(Payne et al., 1998)
ABCA1	Tangier disease	Mutations	Loss of plasma membrane localization	(Tanaka et al., 2003)
Tau	Neurodegenerative diseases	Hyperphosphorylation	Mislocalization to dendritic spines	(Hoover et al., 2010)
TARDBP	ALS and FTLD	Unknown	Cytoplasmic mislocalization	(Winton et al., 2008)
FUS	FTLD	Mutations	Cytoplasmic mislocalization	(Vance et al., 2009)
FOXO	Various types of cancer	Post-translational modifications	Cytoplasmic mislocalization	(Dansen and Burgering, 2008)
p53	Various types of cancer	Mutations, post-translational modifications	Cytoplasm	(Fabbro and Henderson, 2003)

APECED, autoimmune polyendocrinopathy–candidiasis–ectodermal dystrophy; ALS, amyotrophic lateral sclerosis; FTLD, frontotemporal lobar degeneration; TRPS, trichorhinophalangeal syndrome; XLAG, X-linked lissencephaly with absent corpus callosum and ambiguous genitalia.

Figure 1: Mislocalized proteins that have been associated with human diseases

2 Problematic

The manual prediction by radiology represents a less useful and slower methodin face of rapid development and the need to create new medicines, so new technologies play a critical role to assist radiologists (but not replacing them) with artificial intelligence, especially machine learning, presents a real solution for analyzing and extracting the information.

But, usually when scientist want to elaborate databases that present their research results, a problem related to rare cases occurs : It's difficult to predict rare diseases when doing machine learning, especially when relying on **imbalanced dataset**

Question: How can we predict the cellular localization sites of proteins using imbalanced DATA?

In this project we aim to utilise various machine learning algorithms on a given database in order to solve this problem.

3 Machine Learning (ML) algorithms

3.1 Machine Learning and AI

Machine learning is a subset of artificial intelligence (AI) that involves the use of algorithms to enable machines to learn from data and improve their performance on a task over time. Machine learning algorithms are designed to automatically identify patterns in data and make predictions or decisions based on those patterns.

In the context of AI, machine learning plays a critical role in enabling machines to perform complex tasks that would otherwise be difficult or impossible for them to accomplish using traditional rule-based programming. By analyzing and learning from large amounts of data, machine learning algorithms can identify complex relationships and patterns that would be difficult for humans to discern on their own.

Machine learning is used in a variety of applications within AI, including image and speech recognition, natural language processing, autonomous vehicles, and predictive analytics. As data continues to grow in size and complexity **(balanced/imbalanced)**[1], the importance of machine learning in AI is likely to increase, enabling machines to become even more intelligent and capable of performing increasingly complex tasks.

3.2 Machine Learning steps

[1] *Dealing with imbalanced data, where one class has significantly more or fewer observations than another, can add complexity to data analysis and model building.*

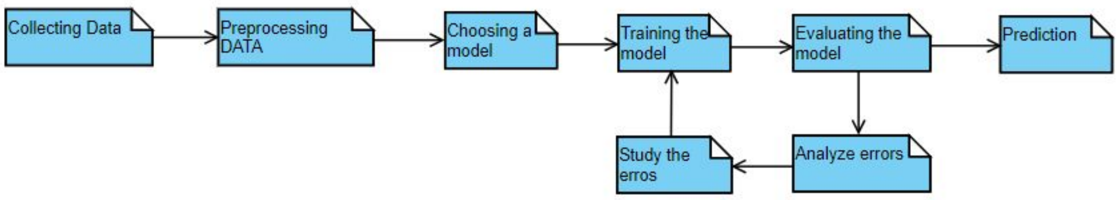
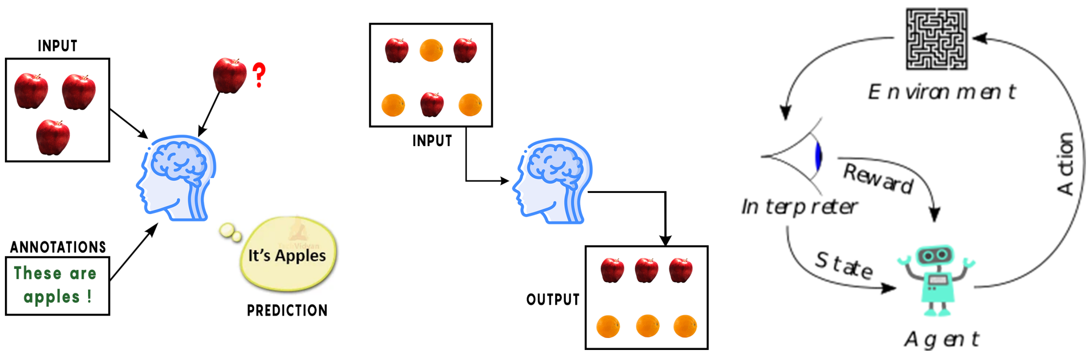


Figure 2: Steps of ML process

This figure displays the steps of a ML model

3.3 Machine Learning algorithms types

There exist many types of Machine Learning algorithms based on the complexity of the dataset being used. However, in this context, we will focus on three main types: supervised learning, unsupervised learning, and reinforcement learning. The figures below display each algorithm type.



(a) Supervised learning (b) Unsupervised learning (c) Reinforcement learning

Figure 3: The 3 main types of machine learning algorithms : supervised, unsupervised and reinforcement learning.

3.4 Machine Learning Application Types

Machine learning applications typically involve several key steps, including data preparation, model selection, training, and testing. One important decision in model selection is whether to use a regression model or a classification model, depending on the nature of the problem being addressed. **Regression models** are used when the output is a continuous variable, while **Classification models** are used when the output is categorical or binary. Other factors that influence the choice of model include the size and quality of the dataset, the complexity of the problem, and the resources available for training and deployment.

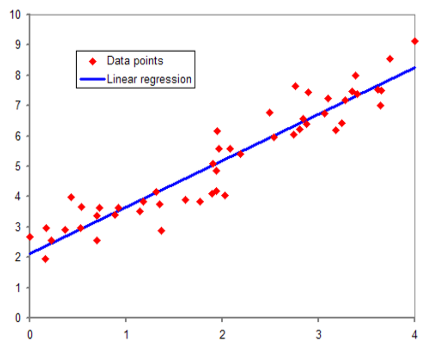


Figure 4: Regression Model

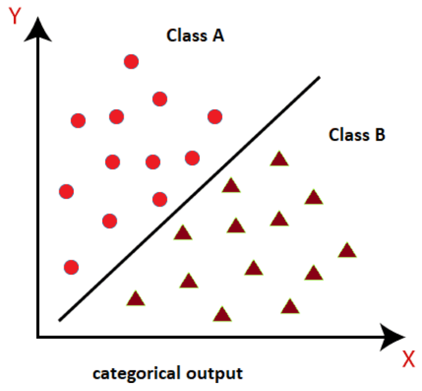


Figure 5: Classification Model

Figure 6: Machine Learning Application Types

3.5 Dataset

Data is information or facts recorded, stored and processed by electronic devices. It can be in various forms including numbers, text, images, audio and video, it can be structured (data/address), semi-structured (XML/JSON format) and unstructured (TXT/IMG/VID/social media posts/emails/customer reviews) based on their organization.

In the field of Machine Learning, data plays a vital role in training models, improving accuracy, identifying patterns and trends, and breaking down bias. Therefore, it is important to ensure that the data used to train the model is accurate.

We can distinguish two types of datasets : **Balanced and Imbalanced dataset**

3.5.1 Balanced vs Imbalanced Dataset

In a **balanced dataset**, the number of instances for each class is roughly the same. For example, if a binary classification problem has two classes, a balanced dataset would have roughly an equal number of instances for each class. Balanced datasets are preferred because they can result in a more accurate and robust model, as the model is exposed to an equal number of instances from each class during training.

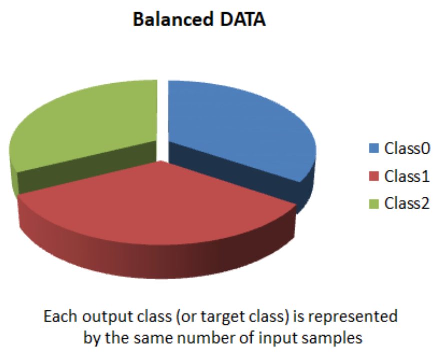


Figure 7: Balanced DATA

In the other hand, an **imbalanced dataset** is one where the number of instances for each class is not equal, and one class has significantly fewer instances than the other. While imbalanced datasets are common in real-world scenarios, they can pose challenges in machine learning. When a model is trained on an imbalanced dataset, it may become biased toward the majority class and struggle to accurately predict the minority class. In such cases, various techniques such as resampling, class weighting, and data augmentation can be used to address the imbalance and improve the model's performance.

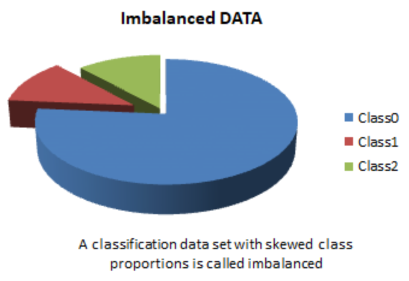


Figure 8: Imbalanced DATA

In a Imbalanced dataset, we find two main classes :

- **Majority Class**: More than half of the examples belong to this class, often the negative or normal case.

- **Minority Class**: Less than half of the examples belong to this class, often the positive or abnormal case.

3.5.2 Balancing the DATASET

A simple approach to using standard machine learning algorithms on an imbalanced dataset is to modify the training dataset to address class imbalance.

-Deleting examples from the majority class, referred to as undersampling[2].

-Adding new examples to the minority class. (Oversampling)[3]

-Cost-sensitive learning[4] is a subfield of machine learning that takes the costs of prediction errors (and potentially other costs) into account when training a machine learning model.

3.5.3 Oversampling techniques

Oversampling techniques in Python are used to balance the class distribution in imbalanced datasets. Some of the popular oversampling techniques in Python are:

-ADASYN (Adaptive Synthetic Sampling)-Borderline-SMOTE.

[2] *The samples are removed randomly from the majority class.*

[3] *The samples are generated from the minority class using some method, such as replication or synthesis.*

[4] *This technique involves assigning different weights to different examples in the training dataset based on their associated costs.*

-Random Minority Oversampling with Replacement (RMOR).

-Synthetic Minority Over-sampling.

-Random Oversampling.

-SMOTE(Synthetic Minority Over-sampling Technique).

In our project we will use:

-**Random Oversampling** : This technique involves randomly duplicating minority class samples until the class distribution is balanced.

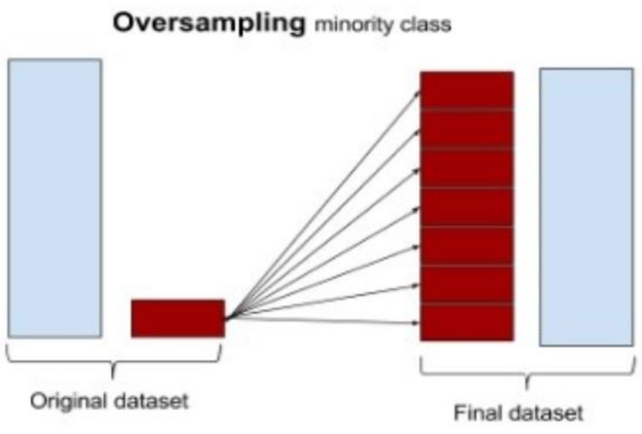


Figure 9: Random Oversampling Method

-**SMOTE (Synthetic Minority Over-sampling Technique)**: This technique involves creating synthetic samples of the minority class by interpolating between existing samples.

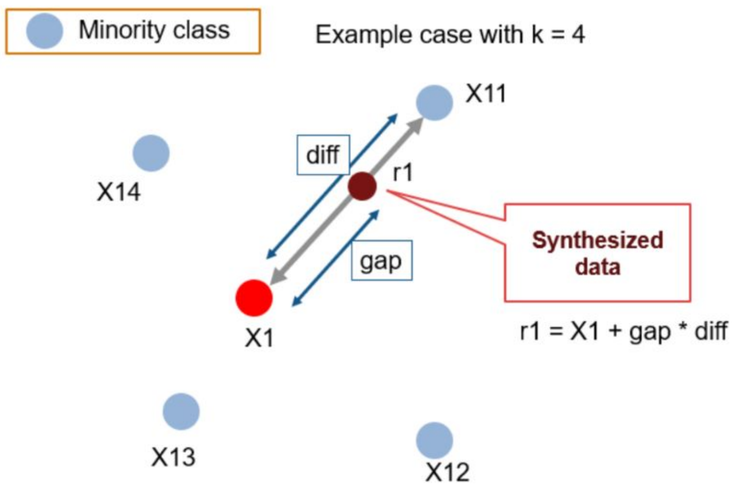


Figure 10: SVM Method

4 Adopted Approach

4.1 Defining the problem

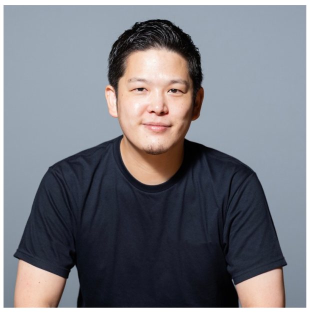


Figure 11: Kentai Nakai

This project will utilize a machine learning dataset called the "E.coli" or "protein localization sites" dataset, which was created by Kenta Nakai and developed by Paul Horton and Kenta Nakai in their 1996 paper. The dataset consists of 336 examples of E.coli proteins each described by seven input variables derived from the protein's amino acid sequence.

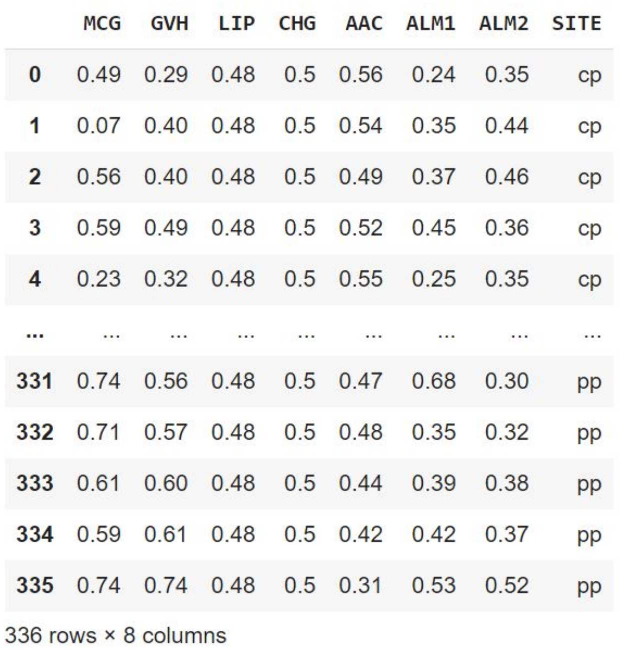
These input features include McGeoch's and von Heijne's methods for signal sequence recognition, scores from the ALOM membrane-spanning region prediction program, and other discriminant analysis scores. The dataset has eight classes, including cytoplasm, inner membrane, periplasm, outer membrane, and various lipoproteins. In their paper, the authors achieved a classification accuracy of 81 percent.

Features	**Meaning**
mcg	McGeoch's method for signal sequence recognition.
gvh	Von Heijne's method for signal sequence recognition.
lip	Von Heijne's Signal Peptidase II consensus sequence score.
chg	Presence of charge on N-terminus of predicted lipoproteins.
aac	Score of discriminant analysis of the amino acid content of outer membrane and periplasmic proteins.
alm1	Score of the ALOM membrane-spanning region prediction program.
alm2	Score of ALOM program after excluding putative cleavable signal regions from the sequence.

There are **eight classes** described as follows:

Classes	**Meaning**
cp	Cytoplasm
im	Inner membrane without signal sequence
pp	Periplasm
imU	Inner membrane, non cleavable signal sequence
om	Outer membrane
omL	Outer membrane lipoprotein
imL	Inner membrane lipoprotein
imS	Inner membrane, cleavable signal sequence

Displaying Data



	MCG	GVH	LIP	CHG	AAC	ALM1	ALM2	SITE
0	0.49	0.29	0.48	0.5	0.56	0.24	0.35	cp
1	0.07	0.40	0.48	0.5	0.54	0.35	0.44	cp
2	0.56	0.40	0.48	0.5	0.49	0.37	0.46	cp
3	0.59	0.49	0.48	0.5	0.52	0.45	0.36	cp
4	0.23	0.32	0.48	0.5	0.55	0.25	0.35	cp
...	...	...	...	...	...	...	...	...
331	0.74	0.56	0.48	0.5	0.47	0.68	0.30	pp
332	0.71	0.57	0.48	0.5	0.48	0.35	0.32	pp
333	0.61	0.60	0.48	0.5	0.44	0.39	0.38	pp
334	0.59	0.61	0.48	0.5	0.42	0.42	0.37	pp
335	0.74	0.74	0.48	0.5	0.31	0.53	0.52	pp

336 rows × 8 columns

Figure 12: our DATASET (336 samples)

So our project aims to resolve a ***multiclass classification*** *problem of a* ***supervised dataset****.*

4.2 Analyzing the DATA

We analysed our dataset to identify any problems that needed to be resolved. Fortunately, we did not find any missing values. We then counted the number of samples in each class and visualised the distribution to gain a better understanding of the data.

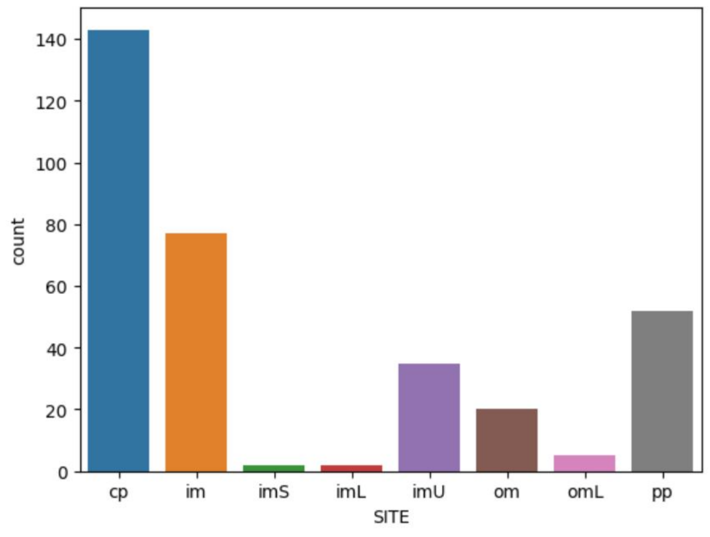


Figure 13: Number of samples of each class

This graph illustrates that our dataset is **imbalanced**. So we need to balance it later in order to get good prediction results.

Correlation

Finally, we want to visualize the correlation between different features of our dataset. To do so, we have plotted the correlation matrix.

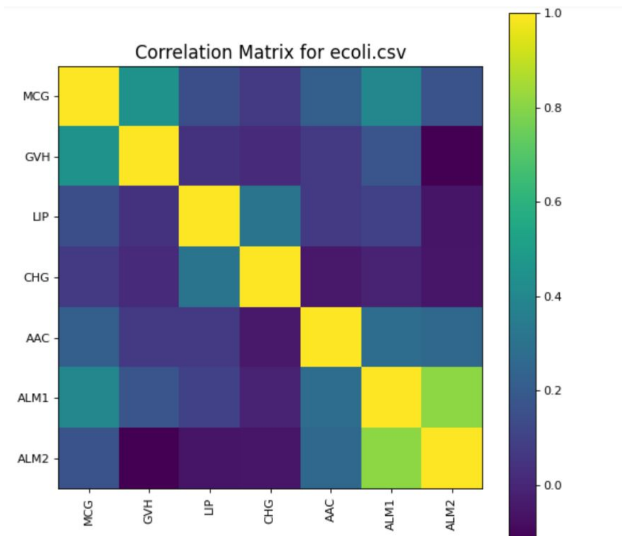


Figure 14: Correlation matrix

Correlation implies a potential connection between two variables, however, it does not establish causation. The observed relationship between the two variables could be influenced by an additional factor or factors beyond the scope of the study.

We found using df.corr() method that the correlation coefficient between '**ALM1**' and '**ALM2**' is: **0.809324**

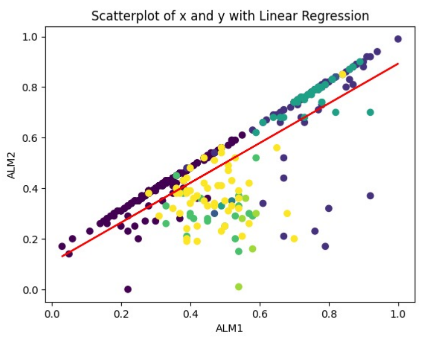


Figure 15: Linear Regression between ALM1 and ALM2

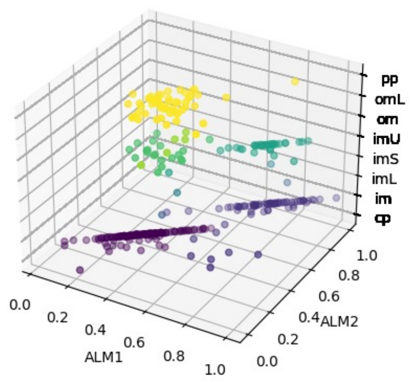


Figure 16: Correlation between ALM1 and ALM2 in 3D

Figure 17: Correlation between our features

The line in red (figure 15), is the regression line represented by the equation:

$$y = a + bx$$

b : The correlation coefficient between the two features (=0.8)

That red line represents the relationship between ALM1 and ALM2 in a linear regression analysis, because it is the line that minimizes the sum of the squared distances between each data point and the line.

This line can be used to make predictions about the dependent variable based on the value of the independent variable. It can also be used to determine the strength and direction of the relationship between the two variables.

Note

This correlation coefficient of 0.8 indicates a positive linear relationship between ALM1 and ALM2. This means that as one variable increases, the other variable tends to increase as well, and vice versa. In case of working with large datasets with many variables, we can drop one of these two variables. ***But in our case, since the dataset is very small, we need to keep all the features that we have.***

Classes distribution as function of different features

A box plot is a graphical technique used to display the distribution of a numerical variable in terms of quartiles. We used box plots to visualize the distribution of our dataset across different features and determine if it was randomly distributed among the classes.

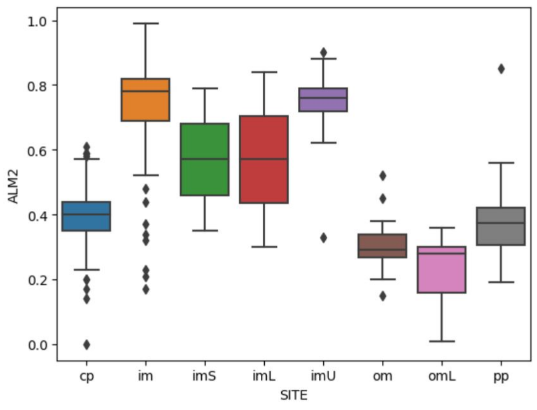


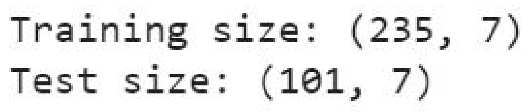
Figure 18: Classes distribution as function of ALM2 feature

This plot shows the distribution of different classes based on the ALM2 feature. So, we will normalize our dataset in order to make all the minimum values 0 and maximum values which is 1

4.3 Preprocessing DATA

After completing the data exploration process, we will prepare the data for the model. To achieve this,we will start by dividing the dataset into two parts: the independent data set (X) (Features) , which is also known as the feature data set, and the dependent data set (Y) (Targets), which is also known as the target data set.

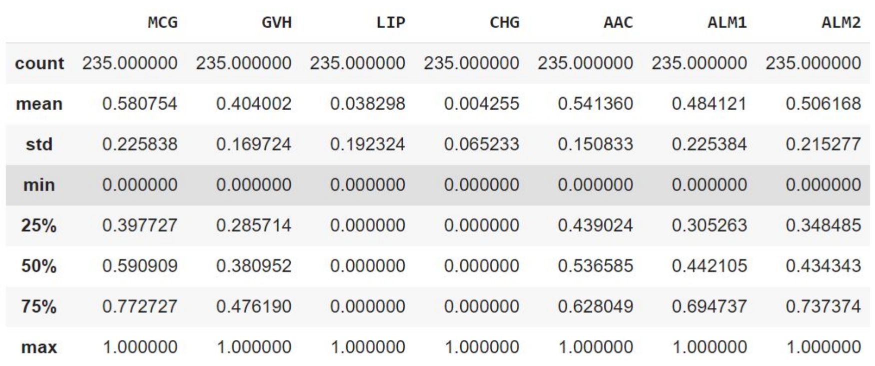
Split the data again, but this time into **70 percent training** and **30 percent testing** data sets because our data is too small (336 samples) to choose a 80/20.



```
Training size: (235, 7)
Test size: (101, 7)
```

Figure 19: Our train and test data size

Since our dataset is clean, we will just rescale the train data between 0 and 1. After the normalization of the features we obtain the maximum and the minimum of values respectively 1 and 0.



	MCG	GVH	LIP	CHG	AAC	ALM1	ALM2
count	235.000000	235.000000	235.000000	235.000000	235.000000	235.000000	235.000000
mean	0.580754	0.404002	0.038298	0.004255	0.541360	0.484121	0.506168
std	0.225838	0.169724	0.192324	0.065233	0.150833	0.225384	0.215277
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.397727	0.285714	0.000000	0.000000	0.439024	0.305263	0.348485
50%	0.590909	0.380952	0.000000	0.000000	0.536585	0.442105	0.434343
75%	0.772727	0.476190	0.000000	0.000000	0.628049	0.694737	0.737374
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Figure 20: Our DATASET Description After Normalization

4.4 Evaluating models' performance

Now we create our algorithms that can incorporate multiple classification models, such as **K Nearest Neighbor**, **Support Vector Machine**, Logistic Regression, Decision Tree Classifier, XGB Classifier and Random Forest Classifier to predict the localization sites of proteins.

To choose the optimal model, many measures has been set to compare the performance of these algorithms :

Confusion Matrix

In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix. A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm. It allows easy identification of confusion between classes e.g. one class is commonly mislabeled as the other. Most performance measures are computed from the confusion matrix.

Accuracy Score

Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right.

$$Accuracy = \frac{TP+TN}{TN+FP+TP+FN}$$

With :
TP : True Positive
TN : True Negative
FP : False Positive
FN : False Negative

Recall Score

The recall is the ratio between TP and TP + FN . It is intuitively the ability of the classifier to find all the positive samples.

$$Recall = \frac{TP}{TP+FN}$$

Precision Score

Precision measures the proportion of correctly predicted positive instances out of all predicted positive instances. It is calculated by dividing the number of true positives by the sum of true positives and false positives.

$$Precision = \frac{TP}{TP+FP}$$

F1 Score

The F1 score is a weighted average of precision and recall.

$$F1\ Score = 2 * \frac{Precision*Recall}{Precision+Recall}$$

4.5 Theoretical Analysis

4.5.1 First model : K Nearest Neighbor

The k-Nearest Neighbors (KNN) algorithm is a type of supervised machine learning algorithm used for classification and regression. The operating principle of the kNN model is based on the idea that similar data points are close to each other in the feature space.

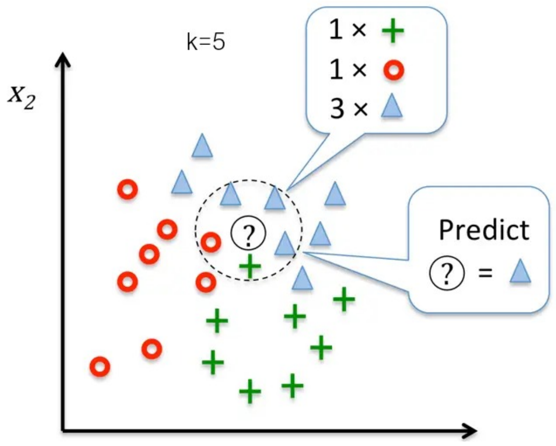


Figure 21: K-Nearest Neighbor Theory

KNN Method

KNN algorithm follow these steps:

1- Choose the number of K nearest neighbors. Calculate the distance between the new data point and each training data point.

$$X = \begin{pmatrix} x_1 \\ . \\ . \\ x_n \end{pmatrix} \quad \text{et} \quad Y = \begin{pmatrix} y_1 \\ . \\ . \\ y_n \end{pmatrix} \qquad d(x, y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

Figure 22: Distance Formula used by KNN Algorithm

2- Select the K nearest neighbors based on the calculated distance.

3- Assign the label of the new data point based on the majority label of the K nearest neighbors.

4.5.2 Second Model : Support Vector Machine (SVM) :

The aim of the SVM algorithm is to identify a hyperplane in an N-dimensional feature space that can effectively separate the data points into their respective classes using a decision boundary

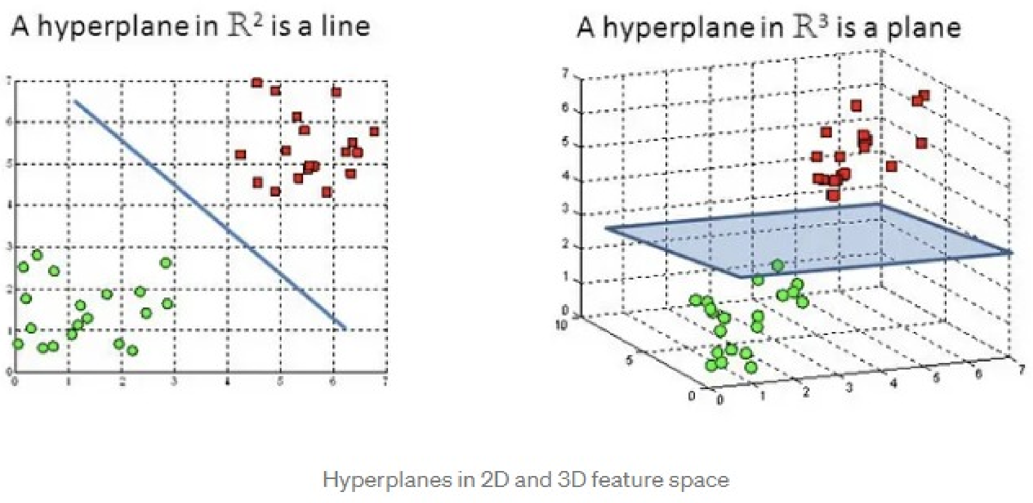


Figure 23: Hyperplanes in 2D and 3D feature space

SVM aims to maximize the distance between data points and the decision boundary. This means that SVM prefers a wider margin (figure below)

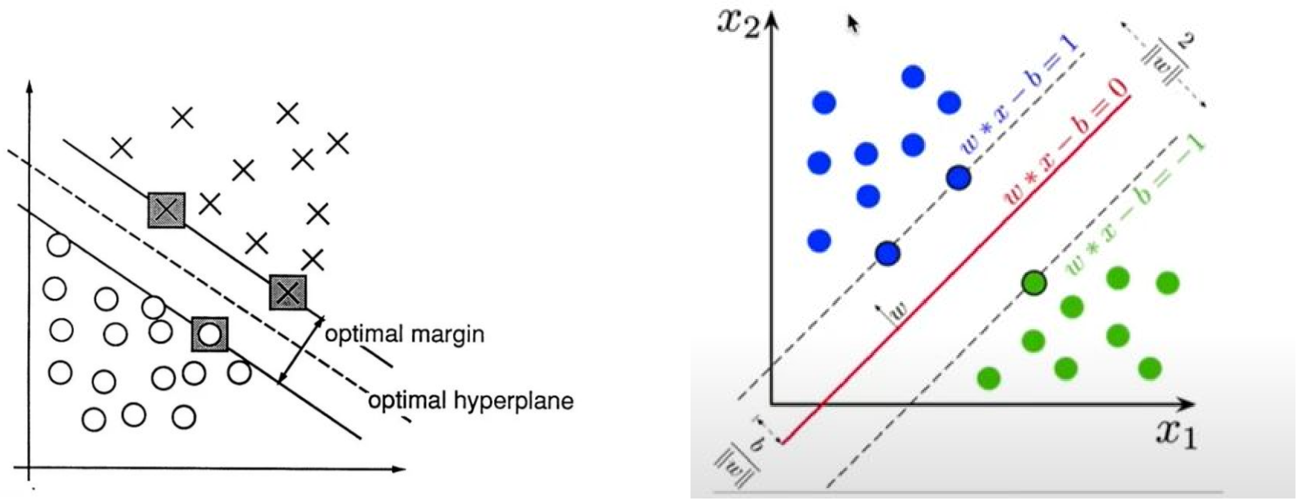


Figure 24: SVM Method

$$
\begin{gathered}
w \cdot x_i - b \geqslant 1 \quad if\ y_i = 1 \\
w \cdot x_i - b \leqslant 1 \quad if\ y_i = -1 \\
y_i(w \cdot x_i - b) \geqslant 1 \quad with\ y_i \in \{-1, 1\}
\end{gathered}
$$

Let's understand the different parameters :

w : Represents the weight , the Weight by SVM operator is applied on it to calculate the weights of the attributes('SITE').

b: Represents the bias is, indeed, a special parameter in SVM. Without it, the classifier will always go through the origin. So, SVM does not give you the separating hyperplane with the maximum margin if it does not happen to pass through the origin, unless you have a bias term.

After that we need to add the regulation value to get the new values of weight and bias.

Regularization

$$J = \lambda ||w||^2 + \frac{1}{n}\sum_{i=1}^{n} \max\{0, 1 - y_i(w \cdot x_i - b)\}$$
$$\text{If } y_i \cdot f(x) \geqslant 1 : \qquad J_i = \lambda ||w||^2$$
$$\text{Else : } \qquad J = \lambda ||w||^2 + 1 - y_i(w \cdot x_i - b)$$

So this regulation term must be derived in order to have the derive of weight and bias, so we apply the gradient

Gradient

$$\text{If } y_i \cdot f(x) \geqslant 1 : \quad \frac{\partial J_i}{\partial w_k} = 2\lambda w_k \text{ and } \frac{\partial J_i}{\partial b} = 0$$
$$\text{Else : } \frac{\partial J_i}{\partial w_k} = 2\lambda w_k - y_i \cdot x_{ik} \text{ and } \frac{\partial J_i}{\partial b} = y_i$$

So now we can get the values of weight and bias updated

Update rule

If $y_i \cdot f(x) \geqslant 1$:

$$w = w - \alpha \cdot dw = w - \alpha \cdot 2\lambda w \quad , \quad b = b - \alpha \cdot db = b$$

Else:

$$w = w - \alpha \cdot dw = w - \alpha \cdot (2\lambda w - y_i \cdot x_i) \ ;$$
$$b = b - \alpha \cdot db = b - \alpha \cdot y_i$$

As we can see, there is a new parameter called alpha (α) in the SVM model, which refers to the learning rate used during training.

To conclude, we now have the code that gives us the hyperplane equation. First, we set the number of iterations for training (higher iterations lead to higher accuracy) and fix the learning rate and parameter lambda. We initialize the weights (weights of each attribute) and the bias (which can be initialized to zero), then we run the code. The for loop runs for the fixed number of iterations and updates the values of w and b according to the if condition. Once the values of w and b are returned by the for loop, we obtain the hyperplane equation, and the prediction is given by the sign of this equation. Finally, we can determine whether the sample belongs to the positive class or the negative class.

Now that we know the approach for binary classification using SVM, we can generalize it for multiclass classification. One way to convert an SVM binary classifier to a multiclass classifier is by using techniques such as One-vs-All (OVA) and One-vs-One (OVO).

4.6 Determination of optimal hyperparameters

4.6.1 What is a hyperparameter?

In machine learning, an hyperparameter is a parameter whose value is set before the learning process begins. Hyperparameters are usually set manually by the programmer and determine the behavior and performance of the model. Examples of hyperparameters include learning rate, number of layers, number of neurons per layer, batch size, and activation functions.

4.6.2 KNN hyperparameters

- K : the number of neighbors
- The distance metric : the choice of a distance metric for a KNN model will depend on the type of data you are working with. Generally, the Euclidean distance is the most popular choice since it works best on continuous data such as real values.

4.6.3 SVM hyperparameters

- Regularization Parameter (C): This parameter tells the SVM optimization how much you want to avoid misclassifying each training example.

- Kernel Type: The kernel type determines which type of hyperplane will be used to separate the data. Some common kernel types are linear, polynomial, radial basis function (RBF) and sigmoid. The most commonly used kernel is the RBF.

In order to optimise K in KNN model we decided to use cross-validation function :

4.6.4 Cross-validation

Cross-validation involves splitting the data into training and validation sets and training the model on the training set while evaluating its performance on the validation set. This process is repeated multiple times, with different subsets of the data used for training and validation, to reduce the impact of random variations in the data

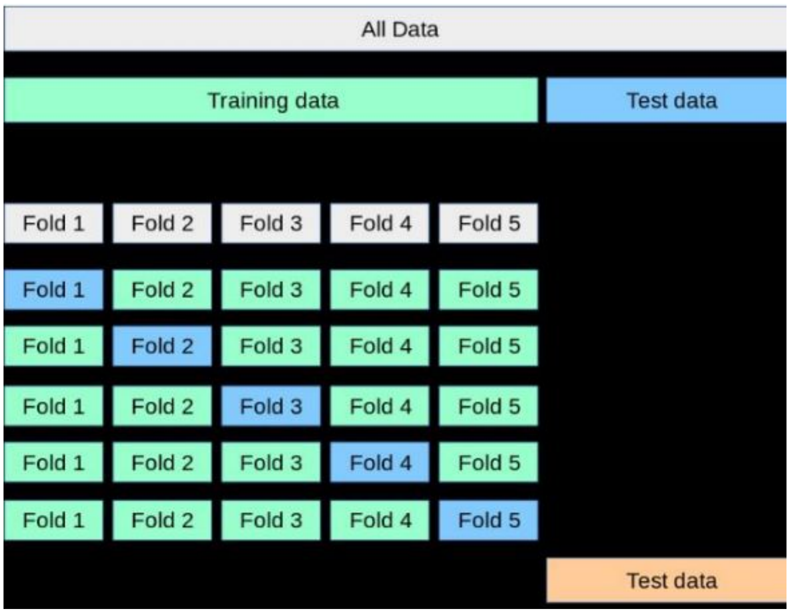
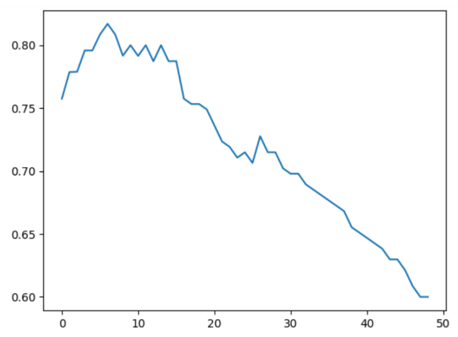


Figure 25: Cross-validation Method

After executing our notebook code we get as result this plot showing cross validation scores as a function of different n neighbors values :



This plot presents a maximum value of the score on the value n_neighbors = 7

Figure 26: Determination of the optimal n neighbor

This figure reveals that the optimal K to choose is 7.

4.6.5 GridSearch

Grid search is a technique used to optimize hyperparameters of a machine learning model.

-In the context of the K-Nearest Neighbors (KNN) model, hyperparameters that can be tuned using grid search include the number of neighbors (K), and the distance metric used to calculate distances between points. 'metric'.

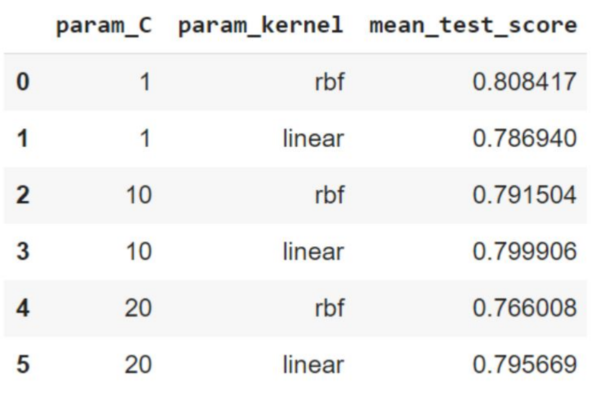
In order to confirm the value that we get with cross validation method, we used the Grid Search Method and we've got the same results:

```
{'metric': 'euclidean', 'n_neighbors': 7}
```

Figure 27: Metric and n neighbors attribution

Hence, for the following use of the k-nearest-neighbor, we will use the parameters 'metric': 'euclidean' and 'n neighbors': 7

-In the context of the Support Vector Machine (SVM) model, hyperparameters that can be tuned using grid search do include the Regularization Parameter (C) and the Kernel Type 'kernel' To choose optimal hyparameters, we applied the GridSearch method and we've got these scores:



	param_C	param_kernel	mean_test_score
0	1	rbf	0.808417
1	1	linear	0.786940
2	10	rbf	0.791504
3	10	linear	0.799906
4	20	rbf	0.766008
5	20	linear	0.795669

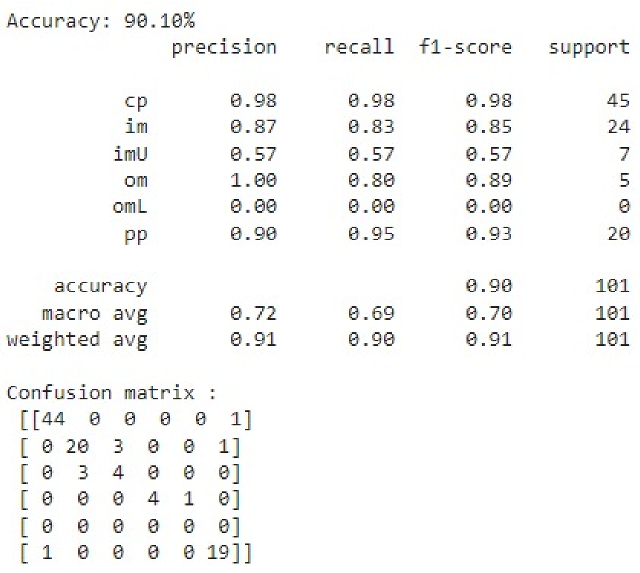
Figure 28: Regularization Parameter (C) and Kernel Type

Hence, for the following use of the Support Vector Machines model, we will use the parameters C=1 and kernel: 'rbf'

4.7 Project WorkFlow

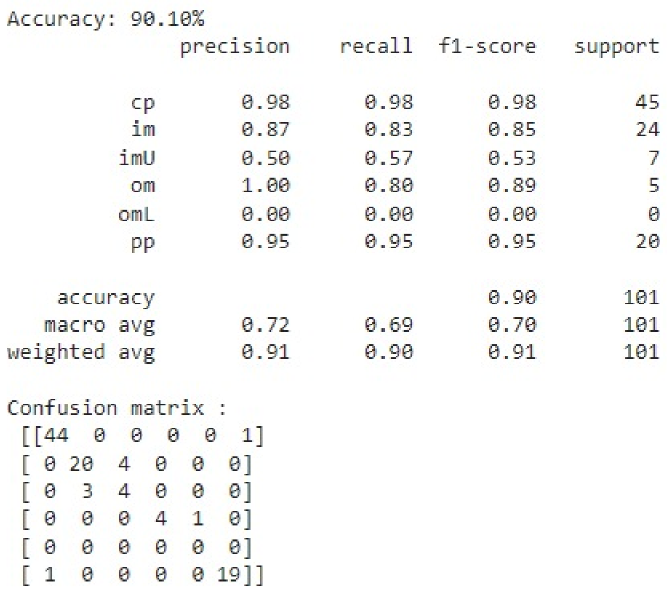
Working on the imbalanced Data

Let's begin by examining the problem statement, addressing the issue of imbalanced data, and finally presenting the outcome. As you can see in the notebook code, we can conclude that the comparison between the two concerned models based on accuracy is not enough. The model does not perform very well in the omL class.



```
Accuracy: 90.10%
              precision    recall  f1-score   support

          cp       0.98      0.98      0.98        45
          im       0.87      0.83      0.85        24
         imU       0.57      0.57      0.57         7
          om       1.00      0.80      0.89         5
         omL       0.00      0.00      0.00         0
          pp       0.90      0.95      0.93        20

    accuracy                           0.90       101
   macro avg       0.72      0.69      0.70       101
weighted avg       0.91      0.90      0.91       101

Confusion matrix :
 [[44  0  0  0  0  1]
 [ 0 20  3  0  0  1]
 [ 0  3  4  0  0  0]
 [ 0  0  0  4  1  0]
 [ 0  0  0  0  0  0]
 [ 1  0  0  0  0 19]]
```

Figure 29: Results using the SVM model on imbalanced data



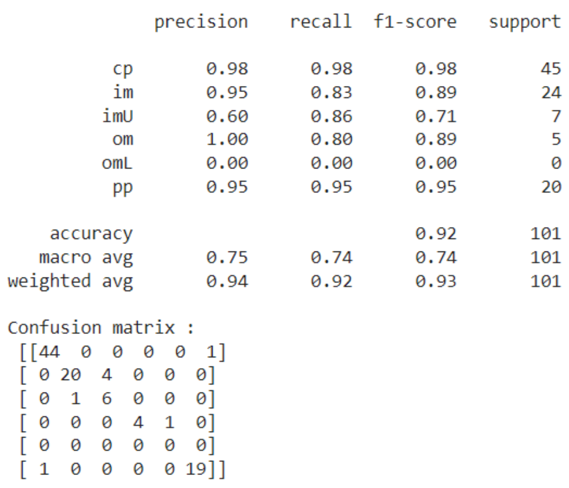
```
Accuracy: 90.10%
              precision    recall  f1-score   support

          cp       0.98      0.98      0.98        45
          im       0.87      0.83      0.85        24
         imU       0.50      0.57      0.53         7
          om       1.00      0.80      0.89         5
         omL       0.00      0.00      0.00         0
          pp       0.95      0.95      0.95        20

    accuracy                           0.90       101
   macro avg       0.72      0.69      0.70       101
weighted avg       0.91      0.90      0.91       101

Confusion matrix :
 [[44  0  0  0  0  1]
 [ 0 20  4  0  0  0]
 [ 0  3  4  0  0  0]
 [ 0  0  0  4  1  0]
 [ 0  0  0  0  0  0]
 [ 1  0  0  0  0 19]]
```

Figure 30: Results using the KNN model on imbalanced data

Thus, we introduce the confusion matrix to show the number of cases that are not predicted . We realized that the source of the problem is : test Data does not contain the minority class.

Balancing our Data automatically

Our goal now is to balance the dataset in order to get more performing results. To minimize the path, we thought to make our data balanced automatically by adding a new hyperparameter class weight in SVM model .



```
              precision    recall  f1-score   support

          cp       0.98      0.98      0.98        45
          im       0.95      0.83      0.89        24
         imU       0.60      0.86      0.71         7
          om       1.00      0.80      0.89         5
         omL       0.00      0.00      0.00         0
          pp       0.95      0.95      0.95        20

    accuracy                           0.92       101
   macro avg       0.75      0.74      0.74       101
weighted avg       0.94      0.92      0.93       101

Confusion matrix :
 [[44  0  0  0  0  1]
 [ 0 20  4  0  0  0]
 [ 0  1  6  0  0  0]
 [ 0  0  0  4  1  0]
 [ 0  0  0  0  0  0]
 [ 1  0  0  0  0 19]]
```

Figure 31: Results using SVM model with a new hyperparameter: class weight

In the diagonal of the confusion matrix we have the number of sites correctly predicted. Otherwise, they are wrongly predicted .We can conclude that , this technique does not resolve the problem, it just increased the accuracy and this is noticed in the site imU where we have 2 other cases exactly predicted. It also decreased the number of not predicted cases from 10 in imbalanced dataset to 8.

Working on balanced train Data

Now, we will try to balance our dataset before training the different models
We started by balancing only the **training** dataset after the splitting using the SMOTE technique .

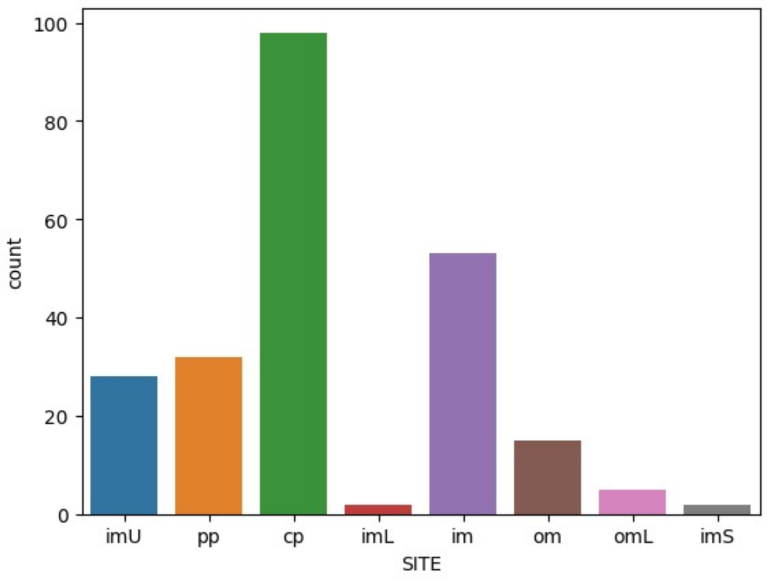


Figure 32: Visualization of the training dataset **before** using SMOTE method

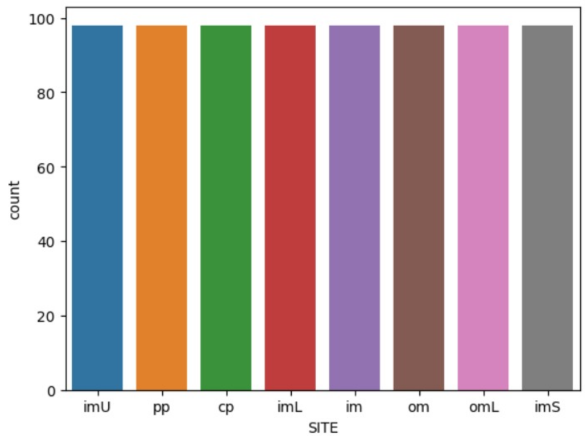
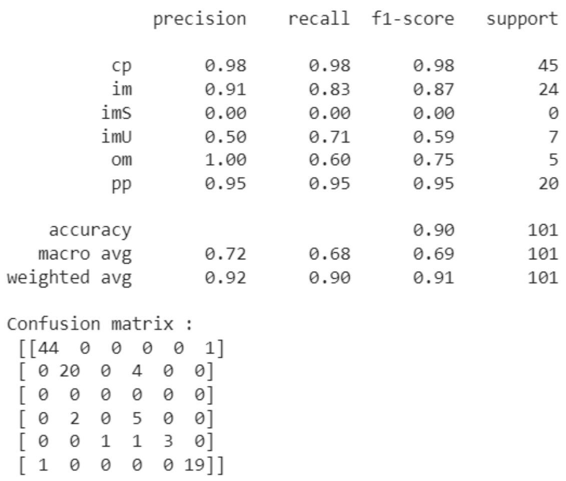


Figure 33: Visualization of the training dataset **after** using SMOTE method

We obtained an augmentation of the number of data in an equal way, 98 for each class.

After applying the SVM model on this dataset, we have got a slight increase from 8 to 10 in wrongly predicted cases .



```
              precision    recall  f1-score   support

          cp       0.98      0.98      0.98        45
          im       0.91      0.83      0.87        24
         imS       0.00      0.00      0.00         0
         imU       0.50      0.71      0.59         7
          om       1.00      0.60      0.75         5
          pp       0.95      0.95      0.95        20

    accuracy                           0.90       101
   macro avg       0.72      0.68      0.69       101
weighted avg       0.92      0.90      0.91       101

Confusion matrix :
 [[44  0  0  0  0  1]
 [ 0 20  0  4  0  0]
 [ 0  0  0  0  0  0]
 [ 0  2  0  5  0  0]
 [ 0  0  1  1  3  0]
 [ 1  0  0  0  0 19]]
```

Figure 34: Results using SVM model on the balanced train data

Overall, the scores that we got are better than those that we got by using the imbalanced dataset because of the fact that certain cases in the minority class are predicted this time. But we did not achieve the desired results.

Working on balanced initial Database (Using SMOTE method)

Now, we will balance the dataset before splitting it into the train part and the test one despite this can lead to an overfitting problem.

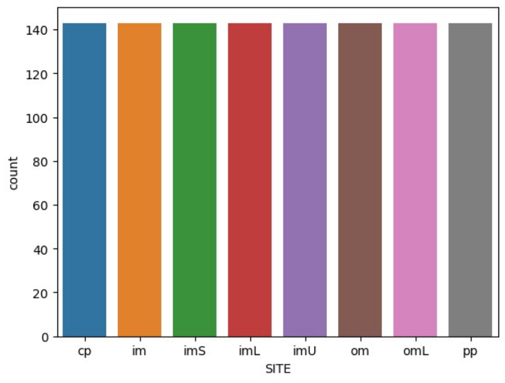
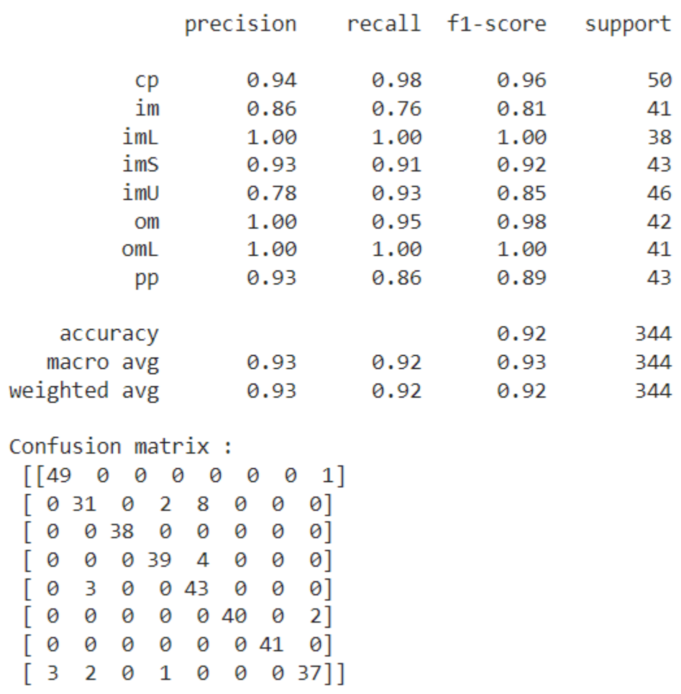


Figure 35: Visualization of the dataset after using SMOTE method

Now, all the dataset was augmented using the SMOTE method, we have 143 data for each site .

We split this dataset into a train part and a test one, then we train the two models on the train part. We've got these results for SVM model:



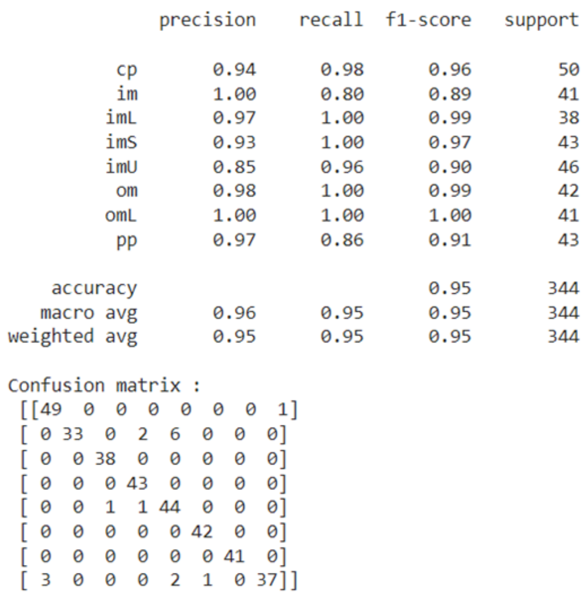
```
              precision    recall  f1-score   support

          cp       0.94      0.98      0.96        50
          im       0.86      0.76      0.81        41
         imL       1.00      1.00      1.00        38
         imS       0.93      0.91      0.92        43
         imU       0.78      0.93      0.85        46
          om       1.00      0.95      0.98        42
         omL       1.00      1.00      1.00        41
          pp       0.93      0.86      0.89        43

    accuracy                           0.92       344
   macro avg       0.93      0.92      0.93       344
weighted avg       0.93      0.92      0.92       344

Confusion matrix :
 [[49  0  0  0  0  0  0  1]
 [ 0 31  0  2  8  0  0  0]
 [ 0  0 38  0  0  0  0  0]
 [ 0  0  0 39  4  0  0  0]
 [ 0  3  0  0 43  0  0  0]
 [ 0  0  0  0  0 40  0  2]
 [ 0  0  0  0  0  0 41  0]
 [ 3  2  0  1  0  0  0 37]]
```

Figure 36: Results using SVM model

We have significant results, the confusion matrix which passed from 6x6 dimension to 8x8 due to the fact that y test contains the minority class this time.

We can see the line of omL respectively precision recall F1-score is [1.00 1.00 1.00]

And for the KNN model:



```
              precision    recall  f1-score   support

          cp       0.94      0.98      0.96        50
          im       1.00      0.80      0.89        41
         imL       0.97      1.00      0.99        38
         imS       0.93      1.00      0.97        43
         imU       0.85      0.96      0.90        46
          om       0.98      1.00      0.99        42
         omL       1.00      1.00      1.00        41
          pp       0.97      0.86      0.91        43

    accuracy                           0.95       344
   macro avg       0.96      0.95      0.95       344
weighted avg       0.95      0.95      0.95       344

Confusion matrix :
 [[49  0  0  0  0  0  0  1]
 [ 0 33  0  2  6  0  0  0]
 [ 0  0 38  0  0  0  0  0]
 [ 0  0  0 43  0  0  0  0]
 [ 0  0  1  1 44  0  0  0]
 [ 0  0  0  0  0 42  0  0]
 [ 0  0  0  0  0  0 41  0]
 [ 3  0  0  0  2  1  0 37]]
```

Figure 37: Results using KNN

The results that we got are very interesting; we can see from the confusion matrix that we got less prediction errors despite the fact that this SMOTE technique that we used can cause overfitting. But, doing this is better than neglecting some classes (as it is the case when we augmented only the train data)

In comparaison between SVM model and KNN one, we can affirme that overall, KNN is more performant.

Working on balanced initial Database (Using RandomOverSampling method)

In the upcoming section, we will attempt augmentation using the Random Over Sampling technique, in order to evaluate its effectiveness compared to the SMOTE method. This time, the augmentation is not equal.

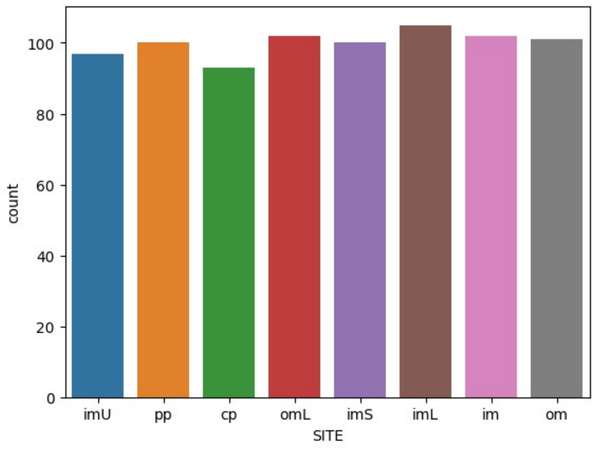
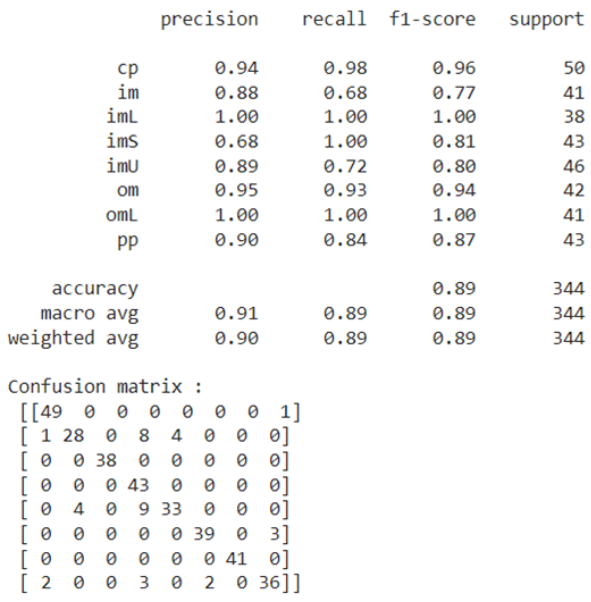


Figure 38: Visualization of the dataset after using RandomOverSampling method

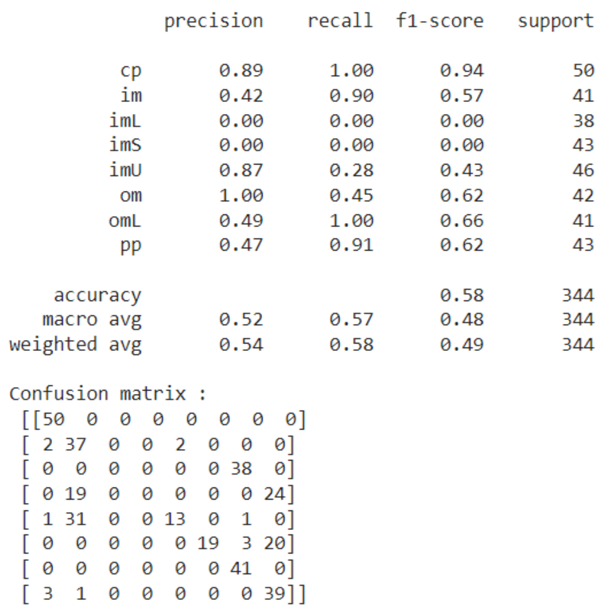
SMOTE is an oversampling technique that generates synthetic examples for the minority class by interpolating the features of neighboring samples, while Random Oversampling simply replicates the same samples of the minority class.

After training the two models with this data augmented using RandomOverSampling, we found this results:



```
              precision    recall  f1-score   support

          cp       0.94      0.98      0.96        50
          im       0.88      0.68      0.77        41
         imL       1.00      1.00      1.00        38
         imS       0.68      1.00      0.81        43
         imU       0.89      0.72      0.80        46
          om       0.95      0.93      0.94        42
         omL       1.00      1.00      1.00        41
          pp       0.90      0.84      0.87        43

    accuracy                           0.89       344
   macro avg       0.91      0.89      0.89       344
weighted avg       0.90      0.89      0.89       344

Confusion matrix :
 [[49  0  0  0  0  0  0  1]
 [ 1 28  0  8  4  0  0  0]
 [ 0  0 38  0  0  0  0  0]
 [ 0  0  0 43  0  0  0  0]
 [ 0  4  0  9 33  0  0  0]
 [ 0  0  0  0  0 39  0  3]
 [ 0  0  0  0  0  0 41  0]
 [ 2  0  0  3  0  2  0 36]]
```

Figure 39: Results using SVM model



```
              precision    recall  f1-score   support

          cp       0.89      1.00      0.94        50
          im       0.42      0.90      0.57        41
         imL       0.00      0.00      0.00        38
         imS       0.00      0.00      0.00        43
         imU       0.87      0.28      0.43        46
          om       1.00      0.45      0.62        42
         omL       0.49      1.00      0.66        41
          pp       0.47      0.91      0.62        43

    accuracy                           0.58       344
   macro avg       0.52      0.57      0.48       344
weighted avg       0.54      0.58      0.49       344

Confusion matrix :
 [[50  0  0  0  0  0  0  0]
 [ 2 37  0  0  2  0  0  0]
 [ 0  0  0  0  0  0 38  0]
 [ 0 19  0  0  0  0  0 24]
 [ 1 31  0  0 13  0  1  0]
 [ 0  0  0  0  0 19  3 20]
 [ 0  0  0  0  0  0 41  0]
 [ 3  1  0  0  0  0  0 39]]
```

Figure 40: Results using KNN model

This time we have a total decrease in accuracy especially for the KNN model.

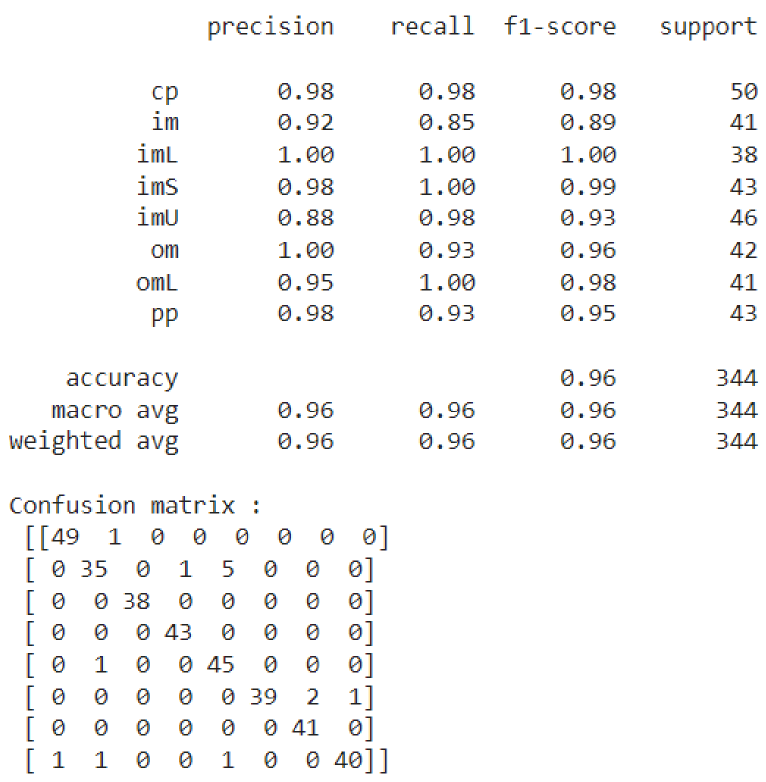
4.8 Discussion

After this study, the final conclusion is that the model which performs the best is when we choose to do **SMOTE** method to the **initial** dataset while using the **K Nearest Neighbor** algorithm.

To have a further vision on our results, we decided to look for other multiclass-class classification models that have better performance.

We tried these four models: Logistic Regression, Decision Tree, XGBClassifier and Random Forest.

Finally we found that the most performing one is the Random Forest model which come out the following good results:



```
              precision    recall  f1-score   support

          cp       0.98      0.98      0.98        50
          im       0.92      0.85      0.89        41
         imL       1.00      1.00      1.00        38
         imS       0.98      1.00      0.99        43
         imU       0.88      0.98      0.93        46
          om       1.00      0.93      0.96        42
         omL       0.95      1.00      0.98        41
          pp       0.98      0.93      0.95        43

    accuracy                           0.96       344
   macro avg       0.96      0.96      0.96       344
weighted avg       0.96      0.96      0.96       344

Confusion matrix :
 [[49  1  0  0  0  0  0  0]
 [ 0 35  0  1  5  0  0  0]
 [ 0  0 38  0  0  0  0  0]
 [ 0  0  0 43  0  0  0  0]
 [ 0  1  0  0 45  0  0  0]
 [ 0  0  0  0  0 39  2  1]
 [ 0  0  0  0  0  0 41  0]
 [ 1  1  0  0  1  0  0 40]]
```

Figure 41: Results using Random Forest model

5 Conclusion and Perspectives

In conclusion, the project aimed to develop a multi-class classification model to predict the localization sites of proteins. The main challenge encountered was the imbalance in the dataset, which was addressed by using the SMOTE technique to balance the data. The SVM and KNN models were used for classification, and the results showed that KNN outperformed SVM. Furthermore, the Random Forest model was found to be the most effective. The use of other data augmentation techniques, such as RandomOverSampling, did not yield satisfactory results, which highlights the effectiveness of SMOTE in this project.

In terms of perspectives, this project could be further improved by exploring more advanced techniques to address the problem of imbalanced data. Also, the use of more diverse models, such as deep learning-based ones, could be explored to improve the accuracy of predictions. Another potential direction could be the integration of domain knowledge in the form of features to further enhance the accuracy. Finally, the dataset used in this project could be expanded to include more proteins.

References

[1] Starter: Ecoli UCI Dataset aaf72f77-a, https://kaggle.com/code/kerneler/starter-ecoli-uci-dataset-aaf72f77-a, Accessed on March 28, 2023.

[2] PYTHON SKLEARN - MODEL SELECTION: Train test split, Cross Validation, GridSearchCV (21/30) - YouTube, https://www.youtube.com/watch?v=w_bLGK4Pteo&feature=youtu.be, Accessed on April 1, 2023.

[3] Pandas Tutorial, https://www.w3schools.com/python/pandas/default.asp, Accessed on April 1, 2023.

[4] Ecoli Classification, https://kaggle.com/code/bryangray/ecoli-classification, Accessed on April 1, 2023.

[5] Bioinformatics, Coursera, https://www.coursera.org/specializations/bioinformatics, Accessed on April 1, 2023.

[6] 7 Ways to Handle Missing Values in Machine Learning, https://towardsdatascience.com/7-ways-to-handle-missing-values-in-machine-learning-1a6326adf79e, Accessed on April 1, 2023.

[7] Cost-sensitive learning – Reproducible Machine Learning for Credit Card Fraud detection - Practical handbook, https://fraud-detection-handbook.github.io/fraud-detection-handbook/Chapter_6_ImbalancedLearning/CostSensitive.html, Accessed on March 28, 2023.

[8] Turkoglu, I. and Kaymaz, E.D. (2009). A hybrid method based on artificial immune system and k-NN algorithm for better prediction of protein cellular localization sites. *Applied Soft Computing*, 9(2), 497-502. doi: 10.1016/j.asoc.2008.07.003.

[9] Özen, B. (2021). Multi-class Classification on Imbalanced Data using Random Forest Algorithm in Spark. *Medium*. Retrieved on: April 1, 2023. Available at: https://burakozen.medium.com/multi-class-classification-on-imbalanced-data-using-random-forest-\algorithm-in-spark-5b3d0af9b93f.