CLASSIFICATION SUPERVISÉE DE DONNÉES PÉDAGOGIQUES POUR LA RÉUSSITE DANS L'ENSEIGNEMENT SUPÉRIEUR

Saker Amine, Christel Dartigues-Pallez, Rey Gaetan

LABORATOIRE

INFORMATIQUE, SIGNAUX ET SYSTÈMES DE SOPHIA ANTIPOLIS

UMR7271

CLASSIFICATION SUPERVISÉE DE DONNÉES PÉDAGOGIQUES POUR LA RÉUSSITE DANS L'ENSEIGNEMENT SUPÉRIEUR

Saker Amine, Christel Dartigues-Pallez, Rey Gaëtan

EQUIPE **SPARKS**

Rapport de Recherche

Septembre-2019

Laboratoire d'Informatique, Signaux et Systèmes de Sophia-Antipolis (I3S) – UMR7271 - UNS CNRS
2000, route des Lucioles – Les Algorithmes - bât. Euclide B – 06900 Sophia Antipolis – France
http://www.i3s.unice.fr

Membre de UNIVERSITÉ CÔTE D'AZUR

CLASSIFICATION SUPERVISÉE DE DONNÉES PÉDAGOGIQUES

POUR LA RÉUSSITE DANS L'ENSEIGNEMENT SUPÉRIEUR

Saker Amine[1], Christel Dartigues-Pallez[1], Rey Gaëtan[1]

Équipe SPARKS

Septembre-2019 - 45 pages

Abstract :

Research carried out in recent years, particularly in the context of supervised learning, has shown that it is possible to extract relevant knowledge from a set of data representative of a problem.

At the same time, the increase over the years in the supply of university training and the massive arrival of candidates at the gates of the university poses the thorny problem of the orientation of candidates. One of the issues hidden behind this orientation corresponds to the relevance of this orientation and to the personalization / individualization of the courses at the entrance to the university.

In this perspective, this project is interested in finding an adequacy between the data which characterize the candidates for higher education (marks of the candidates and of the classes, high school of origin, etc.) and the marks obtained by these same candidates during of their first year at university.

Key-words : Supervised Learning, Random Forest, Educational Learning

CLASSIFICATION SUPERVISÉE DE DONNÉES PÉDAGOGIQUES POUR LA RÉUSSITE DANS L'ENSEIGNEMENT SUPÉRIEUR

Résumé :

Les recherches effectuées ces dernières années notamment dans le cadre de l'apprentissage supervisé ont montré qu'il était possible d'extraire des connaissances pertinentes à partir d'un ensemble de données représentatives d'un problème.

Dans le même temps, l'augmentation au fil des ans de l'offre de formation universitaire et l'arrivée massive de candidats aux portes de l'université pose l'épineux problème de l'orientation des candidats. Une des problématiques cachées derrière cette orientation correspond à la pertinence de cette orientation et à la personnalisation/individualisation des parcours à l'entrée de l'université.

Dans cette optique, ce projet s'intéresse à trouver une adéquation entre les données qui caractérisent les candidats aux formations du supérieur (notes des candidats et de la classes, lycée d'origine, etc.) et les notes obtenues par ces mêmes candidats lors de leur première année à l'université.

Mots-clefs : Apprentissage supervisé, Forêts aléatoires, Apprentissage éducatif

Table des matières

Introduction Générale

Les recherches effectuées ces dernières années notamment dans le domaine de l'apprentissage supervisé pour la classification de données pédagogiques [3] [4] [5] ont montré qu'il est possible d'extraire des connaissances pertinentes à partir d'un ensemble de données représentatives d'un problème.

C'est dans ce cadre que s'inscrit ce travail. Le présent rapport s'articule de la manière suivante:

- Le premier chapitre comporte une présente le cadre général de ce projet et aborde la méthodologie de gestion de ce projet.
- Le deuxième chapitre expose en premier lieu les concepts fondamentaux clés liés au projet. En second lieu, il décrit l'étude de l'existant, sa critique et décrira l'amélioration envisagée.
- Le troisième chapitre détaille les étapes qui mèneront aux résultats ainsi que les approches envisagées.
- Le quatrième et dernier chapitre présentera dans un premier temps les outils et technologies utilisés pour la réalisation de ce travail et enfin les résultats des approches détaillées dans le chapitre 3.

Nous clôturons ce rapport de recherché par une conclusion, dans laquelle nous évaluerons les résultats atteints et nous exposerons les perspectives éventuelles du présent projet.

Chapitre I Cadre du projet

Dans ce premier chapitre nous nous intéresserons au cadre général de notre projet. Il s'agit d'une présentation de l'organisme d'accueil, du sujet de stage et enfin de la méthodologie de gestion de projet appliquée pour assurer le bon déroulement de ce travail.

1 Contexte du projet

1.1 Présentation du projet

Cette étude s'inscrit dans le cadre d'un projet de recherche dont la finalité est d'apporter une aide, voire un conseil à l'orientation des futurs étudiants via un algorithme de Machine Learning. L'idée consiste à étudier les profils d'anciens candidats qui ont suivi la formation informatique à l'IUT de Nice. Dans cette optique, ce projet s'intéresse à trouver une adéquation entre les données qui caractérisent les candidats aux formations du supérieur (données des élèves anonymisées et enregistrées par ParcourSup[1]) et les notes obtenues par ces mêmes candidats lors de leur première année universitaire (données des étudiants enregistrées par Apogée[2]) en utilisant l'algorithme de Machine Learning "RandomForest". Cette étude porte plus particulièrement sur l'aspect échec/réussite.

1.2 Problématique

Chaque année, en France, près de 800 000 nouveaux bacheliers se présentent aux portes des universités. L'orientation de ces candidats pose un problème majeur au comité de sélection de l'enseignement supérieur, d'autant plus qu'au fil des ans l'offre de formation universitaire ne cesse d'augmenter. Une des problématiques cachées derrière cette orientation correspond à la pertinence de cette orientation et à la personnalisation/individualisation des parcours à l'entrée de l'université.

Est-il possible d'utiliser un algorithme d'apprentissage automatique pour conseiller un étudiant en fonction de ses notes actuelles au lycée ainsi que d'autres informations enregistrées par ParcourSup? Peut-on exploiter toutes ces données? Quelles sont les pré-traitements à effectuer sur ces données?

Pour répondre à ces questions, il est nécessaire d'avoir des connaissances en science des données et en apprentissage automatique. Par conséquent, il a été nécessaire d'effectuer des recherches préliminaires. Dans la suite de ce rapport nous allons essayer de répondre à ces questions en suivant la méthodologie décrite ci-dessus.

2 Bilan

Dans ce chapitre, nous avons présenté le cadre général du travail, et la présentation du projet, le positionnement de la problématique de notre étude et son objectif. Et pour finir, nous avons la méthode qui va guider notre travail tout au long du projet. Dans le chapitre suivant, nous introduirons les concepts fondamentaux ainsi que le support qui serviront de base pour la suite de notre projet.

---

[1] . Parcoursup est la plateforme nationale française de pré-inscription des lycéens en première année d'enseignement supérieur.

[2] . Application Pour l'Organisation et la Gestion des Enseignements et des Étudiants.

Chapitre II Concepts fondamentaux et état de l'art

Ce chapitre introduit les concepts fondamentaux, qui ont servi de support pour notre travail. Nous commencerons par donner une idée générale de l'apprentissage automatique à travers un exemple d'application puis nous exposerons brièvement le travail existant avant de passer à sa critique.

1 Apprentissage automatique

1.1 Définition

Nous pouvons définir l'apprentissage automatique de la façon suivante: Expliquer à une machine comment réaliser une tâche en lui procurant plusieurs exemples. Plus on dispose d'exemples mieux l'algorithme comprendra la structure des données, inférera les paramètres qui vont servir à prédire sur de nouveaux exemples.

1.2 Apprentissage supervisé

L'apprentissage supervisé regroupe les taches de classification, régression et de ranking. Il s'agit en général de traiter un problème de prédiction. L'exemple suivant nous permettra de comprendre le paradigme de l'apprentissage supervisé. Imaginons une base de données de biens immobiliers. Chaque ligne représentant une observation (appartement, maison, studio, etc..) décrite par des variables dites prédictives (les colonnes: type de bien immobilier, nombre de chambres, surface en m^2, proche ou non d'un transport public, etc ...), codées par des chaînes de caractères, des chiffres, des booléens, etc. Le but est de prédire le prix du bien immobilier. Les lignes sont appelées les échantillons (*sample* en anglais), les colonnes: *features* et ce qu'on essaye de prédire, la variable à prédire: *target*. Une fois cette base de données de samples (*dataset* en anglais) dite d'apprentissage (*train* en anglais) à notre disposition, on y récupère des samples sans prix (échantillons de *test*) et on essaye de prédire le prix de ces nouveaux bien immobiliers en ayant appris sur notre dataset train.

Mathématiquement parlant, supposons que l'on dispose d'un ensemble de données D (*dataset* en anglais), décrit par un ensemble de caractéristiques X (*features* en anglais). Un algorithme d'apprentissage supervisé trouve une fonction appelé fonction de mapping ou target function, entre les features en entrée: X et la variable de sortie à prédire: Y.

Cette fonction constitue le modèle de prédiction. (*target* en anglais): $f(X)\ 7\rightarrow\ Y$ la variable de sortie Y se distingue en deux types de catégories de valeurs. Elle peut prendre une valeur continue (valeur infinie parmi un ensemble infini de valeurs), dans ce cas nous traitons une tâche de régression ou une valeur discrète (valeur finie parmi un ensemble fini de valeurs), alors nous traitons une tâche de classification.

1.3 Forêts aléatoires

Nous allons présenter dans cette section le modèle de classification Forêts aléatoires (ou *Random Forest* en anglais). Afin de comprendre le fonctionnement de ce modèle, nous devons commencer par comprendre le fonctionnement d'un arbre de décision (ou *Decision tree* en anglais).

Arbre de décision [6]

Un arbre de décision est un classifieur en forme d'arbre (en sens informatique), possédant une racine, des noeuds, des branches et des feuilles. Lorsqu'on entraîne un modèle de type arbre de

décision, les features du dataset sont testés au niveau des noeud de l'arbre, puis pour chaque valeur possible du feature testé, une branche est formée. La procédure se poursuit jusqu'à ce qu'il n'y ait plus de feature à tester et on obtient ainsi des feuilles représentant les différentes classes de la variable cible (on possède généralement une classe au minimum).

La figure II.1 illustre un exemple d'arbre de décision.

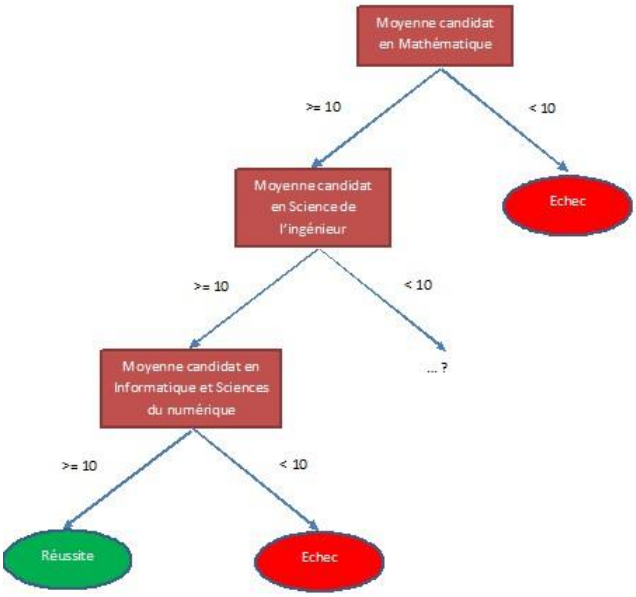


Figure II.1 – Aperçu du fonctionnement d'un arbre de décision

Random Forest[6]

L'algorithme des forêts aléatoires (ou *Random Forest* en anglais) proposé par Leo Breiman en 2001 [7] est l'un des algorithmes les plus utilisés aujourd'hui pour traiter des problème de classification. Il s'agit d'un algorithme qui comme son nom l'indique combine plusieurs arbres de décisions dans une approche de type bagging[3]. L'algorithme entraîne plusieurs arbres de décisions sur des samples sélectionnés aléatoirement. La prédiction de l'algorithme est calculée à partir de la moyenne des des prédictions de chaque arbre de décision construit avec des données sont quantitatives. Pour les données qualitatives, l'algorithme utilise la moyenne des prédiction des modèles indépendants et procède à un vote pour déterminer sa prédiction, comme l'illustre la figure II.2 ci-dessous.

[3] . Bagging: Moyenner les prédictions de plusieurs modèles indépendants afin de réduire l'erreur de prédiction générale.

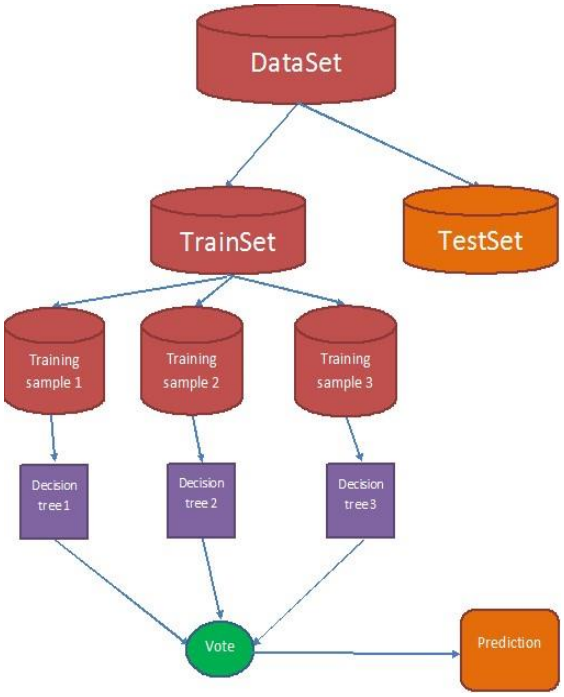


Figure II.2 – Aperçu du fonctionnement de Random Forest

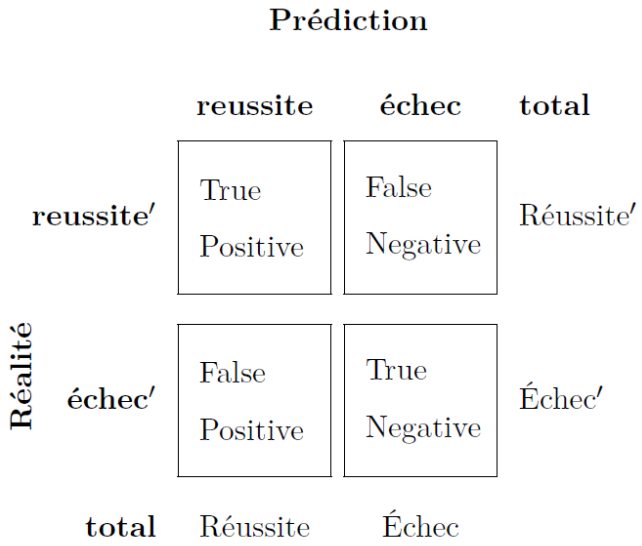
1.4 Évaluation des modèles de classification [2]

Dans le domaine de la recherche, il est courant de devoir comparer des modèles de classification afin de sélectionner le meilleur modèle pour nos données. Nous devons être capable de dire si un modèle est plus performant qu'un autre tout en le justifiant. Pour cela, il existe différents critères d'évaluation. Nous allons définir dans cette section ces critères et en sélectionner deux d'entre eux pour l'évaluation de notre modèle. Pour commencer procédons aux définitions suivantes:

- Un candidat avec une variable cible supérieur à 10 est labelisé "Réussite»: **"Réussite" est une classe positive.**
- Un candidat avec une variable cible inférieur à 10 est labelisé "Échec»: **"Échec" est une classe négative.**

1.4.1 Matrice de confusion

La matrice de confusion (ou *confusion matrix* en anglais) est un tableau à double entrée vérifiant la fréquence des prédictions correctes par rapport à la réalité.



- TP (Vrais Positifs): un vrai positif est un résultat où le modèle prédit correctement la classe positive.
- TN (Vrais Négatifs): un vrai négatif est un résultat où le modèle prédit correctement la classe négative.
- FP (Faux Positifs): un faux positif est un résultat où le modèle prédit incorrectement la classe positive
- FN (Faux Négatifs): un faux négatif est un résultat où le modèle prédit incorrectement la classe négative

La lecture du nombre de prédictions peut se faire en diagonale. Les prédictions correctes: *TP* + *TN* et celles incorrectes: *FN* + *FP*.

1.4.2 Justesse

La justesse, score (ou *accuracy* en anglais) représente la part de prédictions correctes effectuées par un modèle. La formule du score est définie comme suit:

$$Score = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

1.4.3 Courbe ROC et AUC

Une courbe ROC (receiver operating characteristic) est une courbe qui trace le taux de vrais positifs en fonction du taux de faux positifs (FPR) pour différents seuils de classification[8][4].

[4] . Le seuil de classification représente le critère de valeur scalaire appliqué au score d'un modèle dans le but de séparer la classe positive de la classe négative.

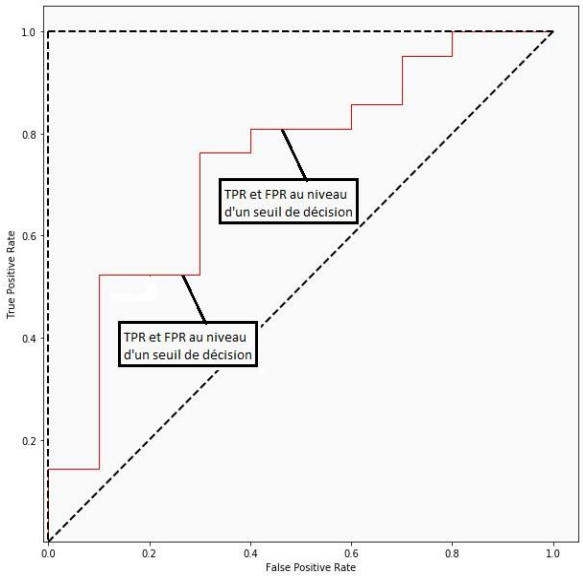


Figure II.3 – Aperçu de la courbe ROC

Aire sous la courbe AUC

L'aire sous la courbe (*Area Under Curve* en anglais) . On peut interpréter l'AUC comme une mesure de la probabilité pour que le modèle classe un exemple positif aléatoire au-dessus d'un exemple négatif aléatoire. Ses valeurs sont comprises dans une plage de 0 à 1. Un modèle dont 100%des prédictions sont erronées a un AUC de 0. Si toutes ses prédictions sont correctes, son AUC est de 1. Pour l'évaluation des modèles de classification, notre choix s'est porté sur les critères d'évaluation: AUC et score.

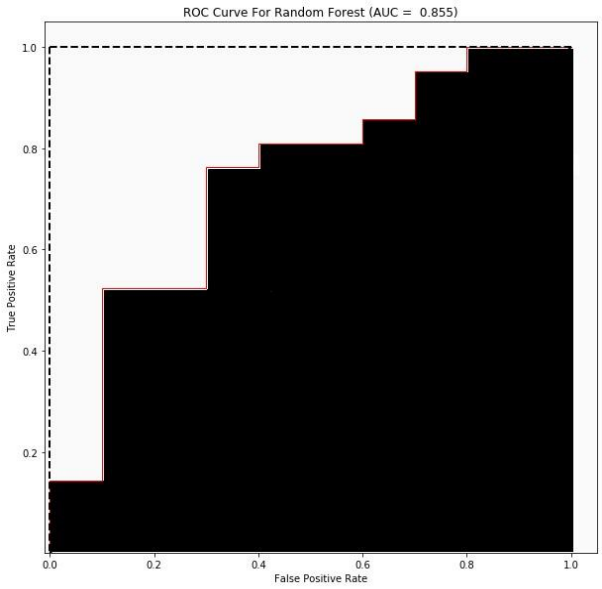


Figure II.4 – Aperçu de l'AUC

1.4.4 Validation croisée

La validation croisée (ou *Cross validation* en anglais, est une méthode de ré-échantillonage qui fonctionne de la manière suivante:

- On divise le DataSet en K blocs.

- On garde un des K blocs comme TestSet et on apprend sur les (K-1) blocks restants et on calcule l'erreur de prédiction
- On répète la procédure K fois en sélectionnant à chaque fois un autre block parmi les (K-1) comme TestSet jusqu'à ce que chaque block soit traité une seul fois.
- On calcul la moyenne des K prédictions pour avoir la prédiction du modèle

2 Étude de l'existant

Le travail antérieur mené par deux étudiants de polytech Nice pour leur projet de PFE est l'existant qui constitue la base de mon stage. Ils ont su aborder le problème et ont mené une conduite de projet d'une manière assez général. Nous allons parcourir brièvement les grandes étapes de leur travail sont les suivantes:

- Pré-traitement des données:
  - Visualisation de quelques données
  - Étude de la différence des colonnes entre les données des deux années
  - Nettoyage des données
- **Extraction de subset:**
  - Cas1: Note obtenues lors de l'année de la terminale: tailles des données 152 lignes, 100 colonnes
  - Cas2: Transformation des notes en moyenne trimestriels: taille des données: 152 lignes, 30 colonnes
  - Cas3: Regroupement de quelques matières pour en constituer une matière de type littéraire: taille des données: 152 lignes, 24 colonnes
  - Cas4: le Cas de référence n'ayant pas subit de pré-traitement particulier: taille des données: 152 lignes, 240 colonnes
- Étude de 4 algorithmes de classification: Random Forest, Support Vector Machines, K-Nearest Neighbors et Multi Layer Perceptron
  - Estimation des paramètres
  - Prédiction
  - Résultats par validation croisée
- Étude générale de plusieurs algorithmes de classification: Je résume dans La figure en A.7:
  - Résultats moyens obtenus selon le cas, le type d'encodage et type de normalisation: Les résultats sont reportées dans le les figures A.2
  - Résultats obtenues pour tous les modèles considérés: Les résultats sont reportées dans le tableaux A.1

3 Critique de l'existant

Nous allons énumérer dans le tableau II.1 les points forts et les points faibles de leur approche.

Pré-traitement des données		Modèle de classification	
Points fort	**Points faibles**	**Points fort**	**Points faibles**
Bonne description des données	Mauvaise imputation des valeurs manquantes	Étude de 4 algorithmes de machine learning	Choix des algorithmes à étudier
Bonne sélection de features (colonnes) avant la fusion des deux années	Pertes d'information (features) lors de la fusion des données des deux années (suppression de colonnes)	Bonne optimisation des algorithmes	Pas d'étude centrée sur un seul algorithme (Random forest par exemple)
Étude de plusieurs type de mise à l'échelle	Mise à l'échelle similaire pour tout type de données	Étude générale de plusieurs modèles de classification	Prédiction avec les paramètres par défaut
Choix du subset	Étude d'un seul subset (subsets résultant de la transformation du subset choisi)		

Tableau II.1 – Points forts et points faibles de l'existant

La reprise de ce travail existant, nous a permis de comprendre les données, leur pré-traitement mais plus particulièrement de mettre en place une nouvelle approche qui pourra déterminer l'échec ou la réussite d'un candidat en utilisant le jeu de données entier, c'est à dire sans sélection de cas, ou de connaissances expertes car pour rappel ce projet est une première étape qui conduira à un outil de sélection et d'aide à l'orientation de futurs candidats pour l'enseignement supérieur et non pour l'IUT Info de Nice.

4 Conclusion

Dans ce chapitre, nous avons d'une part définit les principaux concepts du machine learning avec les termes importants pour pouvoir situer cette étude et d'autre part nous avons analysé l'existant du projet pour pouvoir lui apporter des améliorations, explicitées en détail dans le chapitre suivant. Nous verrons également dans le prochain chapitre la description des approches mises en oeuvre pour la classification de nos données.

Chapitre III Présentation des approches envisagées

Cette partie est primordiale dans le cycle de vie du projet. En effet, toute la base du projet théorique et pratique y est exposée. Par la suite, nous aborderons dans le prochain chapitre, l'exposition de nos résultats et leurs interprétations.

1 Chaîne du machine learning

Comme tout projet de classification, le problème est décomposé en deux grandes parties: le pré-traitement des données puis la création de modèles de machine learning.

1.1 Pré-traitement des données

Avant de pouvoir entraîner les modèles de classification, un pré-traitement des données est nécessaire afin d'éviter de biaiser les performances des modèles. Les données dont on dispose sont brutes, non homogènes, possédant plusieurs valeurs manquantes ainsi que des valeurs aberrantes.

1.1.1 Compréhension des données

Dans un premier temps, nous essayons de comprendre quel type de données nous devons traiter. En effet, la compréhension des données constitue l'étape initiale pour savoir comment aborder un problème d'analyse de données et d'apprentissage automatique.

Description des données

Nous disposons de deux sources de données principales: Les données caractérisant les lycéens et les données du premier semestre de ces mêmes lycéens ayant candidatés à l'IUT Informatique de Nice. Nos avons pu récupérer les données des candidats que sur deux années: 2017 et 2018.

Tailles des données: Le tableau ci dessous représente la taille des données sur les deux années.

Année	Lignes	Colonnes
2017	83	6021
2018	69	6113

Tableau III.1 – Taille des données des deux années

Nous disposons de 83 candidats pour l'année 2017 et de 69 candidats pour l'année 2018. Les features représentant les informations des candidats sont au nombre de 6021 pour l'année 2017 et 6113 pour l'année 2018. Nous pouvons constater une différence dans le nombre de colonnes des deux années. En effet certaines colonnes sont propres à chaque année. Il s'agit principalement de matières présentent dans une année scolaire et non dans l'autre: **276 colonnes** propres au données de 2017 et **184 colonnes** pour les données de 2018. Nous avons donc **5837 colonnes communes** aux deux années. Les colonnes propres à chaque année ne sont pas supprimées d'emblée car on risquerai de rendre par la suite la tâche de nettoyage des colonnes assez difficile, du fait que les colonnes de type notes sont sous un certains ordre. En effet, l'ordre des colonnes notes est le suivant: Moyenne du candidat en matière X / Moyenne classe en matière X / Moyenne plus haute de la classe en matière X / Moyenne plus basse de la classe en matière X; et ce pour toutes les matières (168 matières pour chaque trimestre × 3 trimestres × 4)

Le Taux de remplissage des colonnes dépend des informations du candidat. Par exemple, un candidat ayant effectué une année de terminale en série S, ne remplira pas les champs concernant certaines matières propres aux candidats de série STMG. Par conséquent, un grand travail de nettoyage doit être effectuer sur notre jeu de données. De plus, les données ne sont pas homogènes.

Nous avons plusieurs colonnes de type date (ex: date de validation du bac), chaîne de caractères (ex: libelle établissement pour une année scolaire donnée ou bien mention du diplôme obtenue), catégorielles (ex: Série S, STMG, STI2D), numériques (notes du candidats), etc. Des transformation de certaines colonnes sont donc nécessaires.

1.1.2 Décomposition des données

Après analyse du jeu de données, nous avons pensé à regrouper certaines colonnes entres elles et constituer ainsi cinq groupes de données que nous nommons subsets. Les subsets sont constitués de la même manière pour les deux années, contenant les mêmes types d'information mais ne contiennent pas nécessairement le même nombre de colonnes. La figure III.1 illustre la formation des subsets finaux avec le nombre total de candidats (152 sample au total).

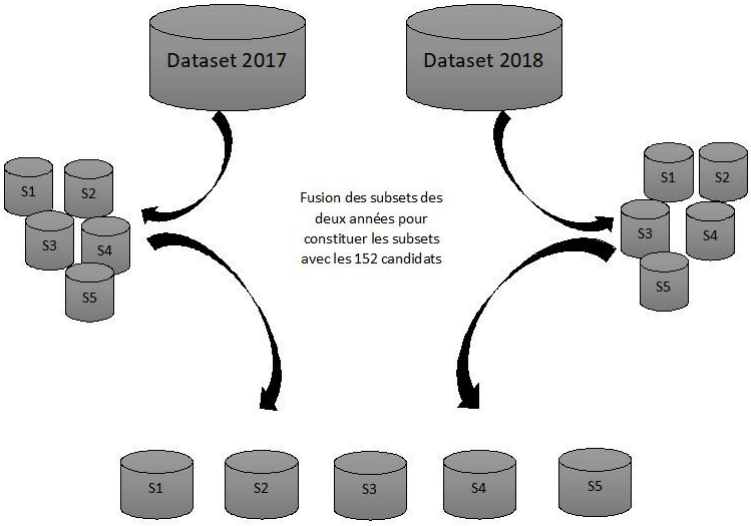


Figure III.1 – Décomposition des datasets des deux années

Le découpage des cinq subsets pour les deux années est réalisé de la manière suivante:

- Notes candidat: Le subset Notes candidat regroupe des colonnes de types notes mais aussi des colonnes à valeurs numériques comme l'effectif d'élèves pour chaque matière, le classement de chaque élève, etc. Nous avons pour ce subset les:
  - Notes des épreuves anticipées.
  - Notes de la première (3 trimestres)
  - Notes de la terminale (2 trimestres)
  - Moyenne des notes obtenues lors de l'année de la terminale.
  - Effectif des élèves dans chaque matière
  - Classement de l'élève dans chaque matière

Nous ne prenons en compte que les notes des deux premiers trimestres de l'année de la terminale car les candidats soumettent leurs voeux pour l'enseignement supérieur avant d'obtenir les résultats du troisième trimestre.

- Profil candidat: Le subset Profil candidat regroupe des colonnes de type chaînes de caractères, colonnes à valeurs catégorielles et quelques colonnes numériques. Nous avons pour ce subset:
  - les données démographiques: Civilité, ville, département et pays de naissance, nationalité, etc.
  - Type de scolarisation

    - D'autres données anonymisées sur le candidat.
- Infos années antérieurs: Le subset Infos années antérieurs regroupe des colonnes de même type que le subset Profil candidat. Nous avons pour ce subset des:
- Infos
    - Informations liées à l'établissement fréquenté: Libellé établissement et localisation de l'établissement (ville département, pays)
    - Informations liées à la formation suivie (type de formation, spécialité choisie, série suivie, matière dominante, etc.)

Ces informations s'étalent sur cinq années, pouvant aller jusqu'à l'année de troisième. Un candidat peut avoir des informations différentes pour une même formation dans le cas où il s'agit d'un redoublant ou bien s'il est déjà inscrit dans le supérieur mais se porte candidat pour une nouvelle formation dans le supérieur.

- Infos bac: Le subset Infos bac antérieurs regroupe des colonnes de type date, chaînes de caractères, colonnes à valeurs catégorielles et quelques colonnes numériques.
    - Informations liées à au diplôme du bac: type de diplôme, date d'obtention du bac, Série diplôme, Spécialité diplôme, Mention diplôme, LV1, LV2, Option du bac, etc.
    - Informations liées à l'académie du bac:
- Dossier candidat: Le subset Dossier candidat regroupe des colonnes de type date, colonnes à valeurs catégorielles et quelques colonnes numériques. Information liées au dossier de candidature su candidat (ex: date de validation du dossier).

La taille de chaque subset est reportée dans le tableau III.2 ci-dessous:

	Notes candidat	Profil candidat	Infos années antérieurs	Infos bac	Dossier candidat
2017	(83, 4780)	(83, 32)	(83, 116)	(83, 22)	(83, 11)
2018	(69, 4848)	(69, 32)	(69, 116)	(69, 22)	(69, 11)

Tableau III.2 – Tailles des subsets avant pré-traitement

1.1.3 Nettoyage des données

Les mêmes opérations de nettoyage de données sont appliquées aux subsets des deux années afin de ne pas perdre de l'information. Les étapes de nettoyage des subsets sont:

- Suppression des colonnes sans importance au sein de chaque groupe formé tel que: Dossier dématérialisé (colonne à valeurs booléennes)
- Suppression des colonnes vides si cette même colonne est vide dans les deux années.

Cas particulier pour le subset Note candidat car comme nous l'avons expliqué précédemment, il existe un certain ordre de colonnes. Pour ce groupe, nous ne supprimons la colonne vide que si les trois autres colonnes qui suivent sont également vides.

1.1.4 Transformation des données

Afin de rendre les colonnes plus ou moins homogènes, certains features au sein des groupes formés ont subit des transformations:

- Transformation des colonnes de type date en numérique: Par exemple afin d'anonymiser les informations concernant le profil du candidat comme sa date de naissance qui est transformée en nombre de mois (nombre de mois depuis le mois de naissance au mois

d'obtention du bac): Supposons qu'on dispose de la date de naissance 19/04/1997 et date d'obtention du diplôme 20/06/2014. La nouvelle date de naissance devient alors:

$(2014 - 1997) \times 12 + (12 - 04) + (12 - 06) = 218$ *mois*

- Transformation des colonnes de type chaîne de caractères en numérique: Les colonnes comme 'libellé établissement', 'ville de naissance', 'département établissement', etc, sont transformées en utilisant l'encodage LabelEncoding (voir section 1.1.8 à la page 21).
- Imputation des valeurs manquantes selon le feature (moyenne des valeurs dans la plupart des cas)

1.1.5 Génération de nouvelles features

- Génération de features à partir de colonnes existantes: Par exemple, on génère une nouvelle colonne que l'on nomme Rang (matière X) à partir de l'effectif d'élève pour une matière donnée et le classement de cet élève dans cette matière:

$$\text{Rang (matière X)} = 1 - \frac{\text{Effectif (matière X)}}{\text{Classement (matière X)}}$$

- Ajout de nouvelles colonnes pour compléter les informations sur les Lycées fréquentés par les candidats: Taux de réussite au bac, Taux de réussite par filière, Note attribuée au Lycée, etc

La taille de chaque subset après pré-traitement est reportée dans le tableau III.2 ci-dessous:

	Notes candidat	Profil candidat	Infos années antérieurs	Infos bac	Dossier candidat
2017	(83, 819)	(83, 16)	(83, 71)	(83, 12)	(83, 7)
2018	(69, 819)	(69, 16)	(69, 71)	(69, 12)	(69, 7)

Tableau III.3 – Tailles des subsets après pré-traitement

1.1.6 Fusion des subsets

Après pré-traitement des subsets des deux années, nous avons constitué les cinq subsets en fusionnant les subsets équivalents des deux années qui ne différent que dans le nombre de candidats (lignes).

Nous avons ainsi 152 candidats pour chacun des cinq subsets, dont la taille est reportée dans le tableau III.4 ci-dessous.

Notes candidat	Profil candidat	Infos années antérieurs	Infos bac	Dossier candidat
(152, 819)	(152, 16)	(152, 71)	(152, 12)	(152, 7)

Tableau III.4 – Taille des subsets après fusion des deux années

Nous supprimons ensuite les colonnes dont le taux de remplissage est inférieur à 70%. Nous obtenons ainsi la taille suivante des subsets:

Notes candidat	Profil candidat	Infos années antérieurs	Infos bac	Dossier candidat
(152, 200)	(152, 10)	(152, 47)	(152, 11)	(152, 2)

Tableau III.5 – Taille des subsets après suppression des colonnes à moins de 70% de remplissage

1.1.7 Variable cible

La variable cible de nos données est calculée à partir de la moyenne des résultats obtenus par les lycéens lors du première semestre à l'IUT Info de Nice. Nous avons huit modules au total. Ensuite, la variable cible est mappée en considérant l'équilibrage suivant: Réussite (valeur 1) si la moyenne des modules est supérieur ou égale à 10. Échec (Valeur 0) si la moyenne des modules est inférieur à 10.

La distribution des classes de la variable cible est la suivante:

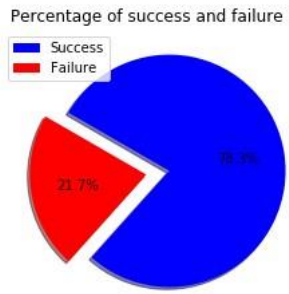


Figure III.2 – Distribution des classes de la variable cible

1.1.8 Encodage des données

Après pré-traitement, l'encodage des données est une étape cruciale dans la préparation de données pour la modélisation. Pour cette étude, nous avons choisi d'encoder nos données sous deux types d'encodage: LabelEncoding et OneHotEncoding.

- LabelEncoding:

L'encodage LabelEncoding consiste à attribuer aux valeurs catégorielles d'une colonne des classes. Par exemple pour la colonne 'Mention diplôme', les valeurs catégorielles possibles sont: 'Sans mention', 'assez bien', 'Bien' et 'Très bien'. Après encodage les valeurs deviennent respectivement '0', '1', '2', '3'. Dans cet exemple on peut dire qu'il existe un ordre de grandeur entre les différentes valeurs catégorielles. Cependant pour d'autres colonnes (ex: LV1), il n'existe pas de relation entre les classes attribués, l'algorithme considérera cet ordre.

- oneHotEncoding:

L'encodage oneHotEncoder consiste à transformer une colonne en n colonnes selon le nombre n de valeurs catégorielles présentent dans cette colonne. Chaque colonne créée correspond alors aux valeurs différentes de la colonne d'origine. Les valeurs des colonnes créées sont binaires. Par exemple pour la colonne 'Nationalité', les valeurs catégorielles sont 'FR', 'UE' ou 'Hors UE'. Les nouvelles colonnes créées sont alors Nationalité_FR, Nationalité_UE et Nationalité_HorsUE. Les lignes qui avaient pour valeur 'FR' dans la colonne d'origine auront dans la colonne Nationalité_FR comme valeur '1'. Les autres valeurs sont remplacées par des '0'. Avec ce type d'encodage, il n'y aura pas de relation d'ordre. L'augmentation du nombre de colonne est un avantage dans notre cas où l'on veut augmenter le nombre de colonnes.

1.1.9 Mise à l'échelle des données

En machine Learning, les features peuvent avoir des ordres de grandeurs différents. La différence d'échelle peut conduire à des performances moindres. La mise à l'échelle des données (ou *feature scaling* en anglais) peut remédier à ce problème. Elle comprend la standardisation et la

Normalisation. Pour notre étude, nous avons sélectionné six types de mise à l'échelle de données dont nous allons détailler trois d'entre elles dans les sous sections suivantes.

- *MinMaxScaler:* Le MinMaxScaler transforme chaque feature en mettant à l'échelle chacune de ses valeurs dans un intervalle fixe [0,1]. Il fonctionne mieux dans les cas où la distribution n'est pas gaussienne ou avec un écart-type est très faible.

$$\frac{X - X_{min}}{X + X_{max}}$$

- *MaxAbsScaler:* fonctionne de manière très similaire que le MinMaxScaler. La différence est qu'il met automatiquement les données à l'échelle sur un intervalle fixe [-1,1] basée sur le maximum absolu du feature.

$$\frac{X}{\max |X|}$$

- *Normalizer:* ou par défaut normalisation L2 est appliquée à chaque ligne de sorte que ses valeurs ont une norme unitaire[5].
- StandardScaler:

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardS

- QuantilTransformer
- Binarizer

En plus des mises à l'échelle citées ci-dessus, nous rajoutons le cas ou les données ne sont pas normalisées. Nous avons donc au total sept mises à l'échelle.

1.2 Modèle de classification

Contrairement à l'existant qui compare l'étude de quatre modèles de classification, notre étude est centrée sur l'algorithme des forêts aléatoires. Nous comparons trois configurations de l'algorithme, expliquées dans les sous-sections suivantes:

1.2.1 Random forest par défaut

Dans un premier temps nous avons utilisé le modèle random forest avec les paramètres par défaut suivants:

RandomForestClassifier(n_estimators = warn, criterion = gini, max_depth = None, min_samples_split = 2, min_samples_leaf = 1, min_weight_fraction_leaf =

0.0, max_features = auto,

max_leaf_nodes = None, min_impurity_decrease = 0.0, min_impurity_split =

None, bootstrap = True, oob_score = False, n_jobs = None, random_state = None,

verbose = 0, warm_start = False, class_weight = None)

Il s'agit de la configuration de base de l'algorithme avec les paramètres par défaut. Nous entraînons nos données en utilisons cette configuration afin de voir le comportement de nos données.

[5] . La norme unitaire avec L2 signifie que si chaque élément était au carré et additionné, le total serait égal à 1.

1.2.2 Random forest optimisé

Nous avons ensuite optimisé l'algorithme par défaut en adaptant les paramètres à nos données. Cette configuration de l'algorithme conduit donc à des résultats plus correctes.

Estimation des meilleurs paramètres

Pour déterminer les meilleurs paramètres du modèle, il est possible de tester différentes combinaisons des paramètres listés ci-dessus. La méthode utilisée est la méthode GridSearchCV. Elle consiste à effectuer une recherche sur grille ou balayage de différents paramètres et d'en ressortir les plus adaptés au dataset.

- e *n_estimators:* Nombre d'arbres dans la forêt e *criterion:* Métrique pour évaluer l'homogénéité des divisions de chaque noeud important ainsi que des variables qui étaient importantes à chaque division. Les deux paramètres utilisés sont l'impureté de *gini* et l'*entropie*.
- e *max$_f$eatures:* Nombre maximum de features à prendre en compte à chaque séparation au niveau des noeuds de l'arbre.
- e *max_depth:* Profondeur maximum de chaque arbre de décision.
- e *min_samples$_s$plit:* Profondeur maximum de chaque arbre de décision. e *min_samples_leaf:* Nombre minimum de sample sur une feuille de l'arbre.
- e *bootstrap:* Méthode d'équilibrage de sample.

OOB error rate

En général, lors de la création de la forêt seulement 2/3 des données sont pris en compte pour former chaque arbre et 1/3 reste donc non-utilisées. Par la suite, ces dernières peuvent être utilisées de manière avantageuse pour nos mesures de précision sans entrainer un coût de calcul supplémentaire. Lors du calcul des OOB, deux paramètres doivent être modifiés: *warm$_s$tart* = *Trueetoob$_s$core* = *True*. Dés lors, il est possible de calculer le taux d'erreur OOB et l'utiliser pour évaluer le nombre d'arbres appropriées pour notre modèle.

Random forest avec validation croisée

Notre troisième configuration applique simplement la validation croisée à la configuration optimisée de l'algorithme.

2 Approches générales

Les approches auxquelles nous avons réfléchi consistent en l'étude de la performance des trois configurations de l'algorithme des forêts aléatoires selon l'encodage et la mise en échelle des données. Les subsets sont d'abord encodés sous un des deux encodages (LabelEncoding ou OneHotEncoding) avant d'être mis à l'échelle et inférés au modèle. La taille des subsests ne varie pas sous l'encodage LabelEncoding car il n'y a pas de création de nouvelles colonnes contrairement à l'encodage OneHotEncoding.

Taille des subsets après OneHotEncoding:

Notes candidat	Profil candidat	Infos années antérieurs	Infos bac	Dossier candidat
(152, 200)	(152, 95)	(152, 275)	(152, 60)	(152, 3)

Tableau III.6 – Taille des subsets après encodage: OneHotEncoding

Dans les sous-sections suivantes, nous allons définir ces approches et les illustrer dans les figures rapportées dans la partie annexe IV.

2.1 Approche 1: Traiter séparément les subsets

La première approche illustrée par la figure annexe A.9 comprend 2 étapes, comme suit: Recherche du meilleur dataset puis entraînement du modèle avec ce dataset. La démarche à suivre est la même pour chaque modèle.

- **Constitution des datasets:** On constitue un seul dataset pour chaque modèle. On choisit une mise à l'échelle parmi les 7, ensuite on entraîne un modèle avec ce subset on en retient la meilleure normalisation en se basant sur les valeurs du couple (AUC, Score) retourné par ce modèle. Enfin on constitue le dataset qui résulte de la fusion des subsets avec la meilleur normalisation, devenant ainsi notre dataset à inférer au modèle. Cette démarche teste pour les 5 subsets × 7 normalisations. Donc 35 subsets testés avec 10 tours de boucles afin d'avoir des résultats du modèle stables (le paramètre *random_state* = *None* pour chaque tour de boucle). Au total cela revient à tester 350 subset avec chaque modèle.
- Entraînement des modèles avec ce dataset: On entraîne ensuite notre modèle avec le meilleur dataset constitué.

2.2 Approche 2: Traiter conjointement les subsets

La deuxième approche illustrée par la figure annexe A.10 comprend les trois étapes suivantes:

- **Constitution des datasets:** On crée d'abord $7^5 = 16807$ datasets à partir de la combinaison des sept mises à l'échelle appliquées à nos 5 subsets.
- **Choix du meilleur dataset:** On entraîne ensuite chaque modèle avec les 16807 datasets et on ne retient que celui qui conduit au plus grand couple (AUC, Score). De la même manière que la première approche, on teste sur 10 tours de boucles pour stabiliser les résultats. Ce qui revient à traiter 168070 datasets.
- Entraînement des modèles avec ce dataset: On entraîne ensuite notre modèle avec le meilleur dataset.

La deuxième approche permet certes de retrouver la meilleure combinaison possible de mises à l'échelle des subsets mais reste très coûteuse en temps d'exécution. De plus, nous devons tester deux types d'encodage, ce qui multiplie par deux le temps d'exécution de chaque approche. Nous avons donc réfléchis à comment réduire le nombre de datasets à tester tout en améliorant les performances des modèles. La méthode est explicitée dans l'approche 3.

2.3 Approche 3: Séparer les types de variables

Dans les deux approches citées précédemment, les types de features (catégorielles ou numériques) des subsets ont été pré-traités de la même manière. Cette nouvelle approche permet de construire deux subsets à partir de nos cinq subsets pré-traités auparavant mais pas encore encodés. la figure rapportée en annexe A.8 illustre la constitution de ces subsets.

- subset 1: formé à partir des variables catégorielles des cinq subsets
- subset 2: formé à partir des variables numériques des cinq subsets

Le subset 2 n'est pas encodé car toutes ses variables sont numériques (il ne reste donc plus qu'à tester les mises à l'échelle). La taille des deux subsets sans encodage et avec OneHotEncoding (du subset catégoriel) est reportée dans le tableau suivant:

	Subset1_catégoriel	Subset2_numérique
Sans encodage	(152, 76)	(152, 205)
Sans encodage	(152,)	(152, 205)

Tableau III.7 – Taille des deux subsets: catégoriel et numérique

L'approche 3 reprend ensuite la même démarche que l'approche 2 en remplaçant les cinq subsets par les deux subsets créés avec $7^2 = 49$ datasets à tester au lieu de 16807.

Conclusion

Dans ce chapitre, nous avons définit la chaîne de machine learning mise en oeuvre pour notre projet. Nous avons exposé quelques pré-traitement des données, puis nous avons expliqué la démarche adoptée pour la réalisation de la seconde étape de cette chaîne à savoir la modélisation des données. Nous disposons donc des données pré-traitées, il nous reste plus qu'a exécuter les trois approches. Dans le chapitre suivant, nous allons exposer les résultats obtenus.

Chapitre IV Réalisation

Dans les chapitres précédents, nous avons détaillé la méthodologie suivie pour notre travail ainsi que la définition de notre approche. Dans l'étape suivante qui suit, nous procéderons à la présentation des différents outils et technologies utilisés, et nous conclurons enfin par les principaux résultats obtenus selon nos différentes approches mises en place, au travers de tableaux représentatifs.

1 Outils et technologies utilisés

1.1 Outils

Les principaux logiciels utilisés sont:

- **Microsoft Office 2016:** Word pour la rédaction des rapport et Excel pour une meilleure visualisation des données.
- **Sublime Text 2:** pour le développement ou la modification de quelques scripts.
- **TexMaker:** pour la génération du rapport de stage.
- **Putty:** Putty [9] pour établir une connexion SSH au serveur du laboratoire afin d'exécuter plus rapidement les scripts Python.

1.2 Technologies

Plusieurs choix techniques ont été faits. Dans ce qui suit, nous illustrons les plus importants:

- **Python 3.7.1:** Le choix du langage de programmation est Python [10]. Il s'agit d'un langage orienté objet, facile à prendre en main, l'un des plus utilisé pour faire du machine learning. Python possède un grand nombre de librairies: d'analyse de données, de calcul scientifique, de visualisation et d'apprentissage automatique.
- **Scikit-Learn:** Scikit-learn [11] est l'un des projets les plus prépondérant sur Github derrière (TensorFlow). Il s'agit d'une librairies d'apprentissage automatique qui permet de faire de la classification, régression, du clustering, de la réduction de dimension, de la sélection de modèle et du pré-traitement de données.
- **Pandas:** Pandas [12] est une librairie Python qui nous a permis de manipuler facilement nos données pour l'analyse et le pré-traitement.
- **NumPy:** La bibliothèque NumPy [13] permet d'effectuer des calculs numériques avec Python.
- **Matplotlib:** Matplotlib [14] est une bibliothèque Python qui permet de tracer et de visualiser des données sous plusieurs formes de graphiques.
- **Seaborn:** Seaborn [15] est une librairie qui vient compléter certaines fonctionnalités non accessibles avec Matplotlib.
- **Jupyter Notebook:** Nous avons utilisé les notebooks Jupyter [16], cahiers électroniques qui, dans le même document, peuvent rassembler du texte, des images, des formules mathématiques et du code Python exécutable.
- **PyCharm:** Nous avons également utilisé PyCharm [17], un IDE pour la programmation Python, simple d'utilisation. Il nous a permis de déboguer des scripts afin de pouvoir nous situer dans une boucle 'for' par exemple.
- **Anaconda3:** Nous avons utilisé Anaconda [18], une plateforme libre qui intègre un grand nombre de packages dont des librairies Python, pour l'installation des librairies citées ci-dessus.

2 Présentation des résultats

Dans cette dernière partie, nous allons présenter à travers quelques tableaux les principaux résultats de nos approches. Ces résultats sont obtenus en utilisant les deux types d'encodages cités à la section 1.1.8 à la page 21. Pour l'approche 1 et 3 nous avons pu avoir les résultats des performances pour les trois configurations de l'algorithme Random forest. Pour l'approche 2, nous n'avons les résultats que pour la configuration de l'algorithme par défaut. Tous les résultats ne sont pas reportés dans ce rapport car comme expliqué précédemment lors de la présentation des approches dans la section 2 à la page 25, nous disposons de beaucoup de datasets à inférer aux différentes optimisations de l'algorithme. Leur temps d'exécution est important malgré les solutions que l'on a trouvé pour remédier à leur résolution, à savoir, l'exécution des scripts sur une machine virtuelle de 16Go de RAM avec 8 CPU.

2.1 Résultats approche 1

2.1.1 Constitution des datasets

Random forest par défaut

LabelEncoding

Le tableau IV.6 ci-dessous affiche les meilleures mises à l'échelle des subsets sous l'encodage LabelEncoding, retournées par le modèle Random Forest par défaut.

Subset	Meilleur normalisation	AUC	SCORE
Notes Candidat	MinMaxScaler	0.7186363636363635	0.6838709677419356
Profil candidat Bac	MaxAbsScaler	0.7288636363636363	0.6451612903225807
Infos années antérieurs	StandardScaler	0.0.8395454545454546	0.7741935483870969
Infos Bac	QuantileTransformer	0.7713636363636363	0.6870967741935485
Dossier candidat	StandardScaler	0.5979545454545454	0.6580645161290324

Tableau IV.1 – Meilleurs normalisations des cinq subsets avec LabelEncoding

Meilleur dataset =

MinMaxScaler (Notes Candidat) + MaxAbsScaler(Profil candidat) + StandardScaler(Infos années antérieurs) + QuantileTransformer(Infos Bac) + StandardScaler(Dossier candidat)

OneHotEncoding

Le tableau IV.6 ci-dessous affiche les meilleures mises à l'échelle des subsets sous l'encodage oneHotEncoding retournées par le modèle Random Forest par défaut.

Subset	Meilleur normalisation	AUC	SCORE
Notes Candidat	Normalizer	0.6913636363636364	0.6806451612903227
Profil candidat Bac	StandardScaler	0.7402727272727726	0.6580645161290324
Infos années antérieurs	QuantileTransformer	0.7963636363636363	0.7322580645161291
Infos Bac	StandardScaler	0.7247727272727273	0.6612903225806452
Dossier candidat	Normalizer	0.636818181818182	0.6645161290322582

Tableau IV.2 – Meilleurs normalisations des cinq subsets avec OneHotEncoding

Meilleur dataset =

Normalizer (Notes Candidat) + StandardScaler(Profil candidat) + QuantileTransformer(Infos années antérieurs) + StandardScaler(Infos Bac) + Normalizer(Dossier candidat)

Random forest optimisé

LabelEncoding

Le tableau IV.6 ci-dessous affiche les meilleurs mises à l'échelle des subsets sous l'encodage

LabelEncoding retournées par le modèle Random forest optimisé.

Subset	Meilleure normalisation	AUC	SCORE
Notes Candidat	Normalizer	0.696590909090909	0.6451612903225807
Profil candidat Bac	MaxAbsScaler	0.7043181818181817	0.6451612903225807
Infos années antérieurs	QuantileTransformer	0.8252727272727729	0.6870967741935485
Infos Bac	MinMaxScaler	0.8159090909090908	0.6806451612903227
Dossier candidat	MaxAbsScaler	0.6690909090909091	0.6451612903225807

Tableau IV.3 – Meilleurs normalisations

Meilleur dataset =

Normalizer(Notes Candidat) + MaxAbsScaler(Profil candidat) + QuantileTransformer(Infos années antérieurs) + MinMaxScaler(Infos Bac) + MaxAbsScaler(Dossier candidat)

OneHotEncoding

Le tableau IV.6 ci-dessous affiche les meilleures mises à l'échelle des subsets sous l'encodage

OneHotEncoding retournées par le modèle Random forest optimisé.

Meilleur dataset =

Subset	Meilleur normalisation	AUC	SCORE
Notes Candidat	Binarizer	0.6679545454545455	0.6451612903225807
Profil candidat Bac	MaxAbsScaler	0.5965909090909092	0.6451612903225807
Infos années antérieurs	Normalizer	0.7552727272727729	0.6451612903225807
Infos Bac	Normalizer	0.7647727272727272	0.6774193548387097
Dossier candidat	QuantileTransformer	0.6547727272727272	0.6451612903225807

Tableau IV.4 – Meilleurs normalisations

Binarizer (Notes Candidat) + MaxAbsScaler (Profil candidat) + Normalizer (Infos années antérieurs)

+ Normalizer (Infos Bac) + QuantileTransformer (Dossier candidat)

Random forest optimisé avec Validation croisée

LabelEncoding

Le tableau IV.6 ci-dessous affiche les meilleurs mises à l'échelle des subsets sous l'encodage LabelEncoding retournées par le modèle Random forest optimisé avec validation croisée.

Subset	Meilleur normalisation	AUC	SCORE
Notes Candidat	MinMaxScaler	0.9974535268652915	0.7827380952380952
Profil candidat Bac	Normalizer	0.9050165520753757	0.7710714285714285
Infos années antérieurs	StandardScaler	0.9251336898395722	0.7369047619047618
Infos Bac	MaxAbsScaler	0.913037942449707	0.7764880952380951
Dossier candidat	MinMaxScaler	0.7760376878023937	0.7835714285714286

Tableau IV.5 – Meilleurs normalisations

Meilleur dataset =

MinMaxScaler(Notes Candidat) + Normalizer(Profil candidat) + StandardScaler(Infos années antérieurs)

+ MaxAbsScaler(Infos Bac) + MinMaxScaler(Dossier candidat)

OneHotEncoding

Le tableau IV.6 ci-dessous affiche les meilleurs mises à l'échelle des subsets sous l'encodage OneHotEncoding retournées par le modèle Random forest optimisé avec validation croisée.

Subset	Meilleur normalisation	AUC	SCORE
Notes Candidat	StandardScaler	0.9961802902979373	0.7698214285714285
Profil candidat Bac	Normalizer	0.8920295390883626	0.7835714285714286
Infos années antérieurs	MinMaxScaler	0.9441049146931499	0.7769047619047619
Infos Bac	QuantileTransformer	0.8761140819964349	0.7702380952380952
Dossier candidat	MinMaxScaler	0.8269671504965623	0.7835714285714286

Tableau IV.6 – Meilleurs normalisations

Meilleur dataset =

StandardScaler (Notes Candidat) + Normalizer (Profil candidat) + MinMaxScaler (Infos années antérieurs)

+ QuantileTransformer (Infos Bac) + MinMaxScaler (Dossier candidat)

2.1.2 Entraînement des modèles

Lorsqu'on entraîne les modèles avec les datasets ci-dessus, on obtient les performances résultats suivantes:

Random forest par défaut

LabelEncoding

- AUC: 0.6636363636363636
- Score: 0.6838709677419355
- F1 score: 0.6284399622945032
- Rappel: 0.6838709677419355
- Précision: 0.6356206035961319

La courbe ROC associée est reportée en figure annexe A.1

OneHotEncoding

- AUC: 0.7106818181818182
- Score: 0.6806451612903227
- F1 score: 0.6094290668769136
- Rappel: 0.6806451612903227
- Précision: 0.6702500132422268

La courbe ROC associée est reportée en figure annexe A.3

Random forest optimisé

LabelEncoding

- AUC: 0.6516949149883933
- Score: 0.770967741935484
- F1 score: 0.6988360935081425
- Rappel: 0.770967741935484
- Précision: 0.882565789473684

La courbe ROC associée est reportée en figure annexe A.1

OneHotEncoding

- AUC: 0.7513141319228276
- Score: 0.7903225806451613
- F1 score: 0.7214925567326504
- Rappel: 0.7903225806451613
- Précision: 0.6732187581695995

La courbe ROC associée est reportée en figure annexe A.3

Random forest optimisé avec validation croisée

LabelEncoding

- AUC: 0.9880315762668703
- Score: 0.7769047619047619
- F1 score: 0.8352062588904694
- Rappel: 0.8618421052631579
- Précision: 0.882565789473684

La courbe ROC associée est reportée en figure annexe A.1

OneHotEncoding

- AUC: 0.9775910364145659
- Score: 0.7835714285714286
- F1 score: 0.8252024291497976
- Rappel: 0.8552631578947368
- Précision: 0.8778462112728631

La courbe ROC associée est reportée en figure annexe A.3

2.2 Résultats Approche 2

2.2.1 Constitution du dataset

Random forest par défaut

LabelEncoding

Après entraînement du modèle avec les 16807 datasets résultants de toutes les combinaisons des mises à l'échelle des subsets, nous avons ressorti le meilleur dataset suivant:

Meilleur dataset =

Normalizer(Notes Candidat) + StandardScaler(Profil candidat) + Binarizer(Infos années antérieurs)

+ Normalizer(Infos Bac) + Normalizer(Dossier candidat)

2.2.2 Entraînement du modèle

Random forest par défaut

LabelEncoding

Lorsqu'on entraîne le modèle avec le dataset ci-dessus on obtient les résultats suivants:

- AUC: 0.7561363636363635
- Score: 0.7419354838709679
- F1 score: 0.7024827551358952
- Rappel: 0.7419354838709679
- Précision: 0.7778561424809671

La courbe ROC associée est reportée en figure annexe A.4

2.3 Résultats Approche 3

2.3.1 Constitution du dataset

Après entraînement du modèle avec les 49 datasets résultants de toutes les combinaisons des mises à l'échelle des deux subsets, nous avons ressorti le meilleur dataset avec les deux types d'encodages pour chaque configuration.

Random forest par défaut

LabelEncoding

Meilleur dataset =

Normalizer(subset1_catégorielles + subset2_numériques)

Lorsqu'on entraîne le modèle avec le dataset ci-dessus, on obtient les résultats suivants:

- AUC: 0.6345454545454545
- Score: 0.6419354838709678
- F1 score: 0.5879492916446967
- Rappel: 0.6419354838709678
- Précision: 0.5928431242057494

La courbe ROC associée est reportée en figure annexe A.5

OneHotEncoding

Meilleur dataset =

Normalizer(subset1_catégorielles) + Binarizer(subset2_numériques)

Lorsqu'on entraîne le modèle avec le dataset ci-dessus, on obtient les résultats suivants:

- AUC: 0.8327272727272728
- Score: 0.7322580645161292
- F1 score: 0.6954713247565418
- Rappel: 0.7322580645161292
- Précision: 0.7569156590124333

La courbe ROC associée est reportée en figure annexe A.6

Random forest optimisé

LabelEncoding

Meilleur dataset =

StandardScaler(subset1_catégorielles) + Normalizer(subset2_numériques)

- AUC: 0.7658811743811744
- Score: 0.7580645161290324
- F1 score: 0.7071908750466626
- Rappel: 0.7580645161290324
- Précision: 0.6927940666228728

La courbe ROC associée est reportée en figure annexe A.5

OneHotEncoding

Meilleur dataset =

MinMaxScaler(subset1_catégorielles + subset2_numériques)

- AUC: 0.8555107147281061
- Score: 0.8064516129032258
- F1 score: 0.7789038421528683
- Rappel: 0.8064516129032258
- Précision: 0.7798522107922189

La courbe ROC associée est reportée en figure annexe A.6

Random forest optimisé avec validation croisée

LabelEncoding

Meilleur dataset =

StandardScaler(subset1_catégorielles) + MinMaxScaler(subset2_numériques)

- AUC: 0.9941431117901707
- Score: 0.783154761904762
- F1 score: 0.9296628668445308
- Rappel: 0.9342105263157895
- Précision: 0.9393104855161158

La courbe ROC associée est reportée en figure annexe A.5

OneHotEncoding

Meilleur dataset =

QuantileTransformer(subset1_catégorielles) + Normalizer(subset2_numériques)

- AUC: 0.9989814107461166
- Score: 0.7569047619047619
- F1 score: 0.9518733273862622
- Rappel: 0.9539473684210527
- Précision: 0.9565058479532164

La courbe ROC associée est reportée en figure annexe A.6

2.4 Comparaison et interprétation des résultats des différentes approches

D'après les résultats des performances des approches 1, 2 et 3, nous pouvons dire que l'étude la mise en échelle des subsets en traitement séparé (approche 1) est moins intéressante que le cas ou toutes les combinaisons de mise à l'échelle sont traitées (approche 2 et 3). En effet nous pouvons remarquer que les différentes performances du modèle varient d'une configuration à l'autre pour chaque approche. (voir les tableaux IV.7 et IV.8 ci-dessous)

Nous constatons une augmentation de

LabelEncoding

	Random Forest Défaut	Random Forest optimisé	Random Forest optimisé avec CV
Approche 1	(0.664, 0.684)	(0.652, 0.771)	(0.989, 0.777)
Approche 2	(0.756, 0.742)		
Approche 3	(0.635, 0.642)	(0.766, 0.758)	(0.994, 0.783)

Tableau IV.7 – Comparaison des résultats des modèles pour les approches 1, 2 et 3, sous l'encodage

LabelEncoding.

OneHotEncoding

	Random Forest Défaut	Random Forest optimisé	Random Forest optimisé avec CV
Approche 1	(0.711, 0.681)	(0.751, 0.790)	(0.978, 0.784)
Approche 3	(0.833, 0.732)	(0.856, 0.806)	(0.999, 0.754)

Tableau IV.8 – Comparaison des résultats des modèles selon les approches 1et 3 sous l'encodage OneHotEncoding.

Nous remarquons également que le type d'encodage qui conduit en moyenne au plus grand couple (AUC, Score) est le 'OneHotEncoding'. En effet, l'encodage 'LabelEncoding' peut fausser les prédictions du modèle s'il n'existe pas de relation d'autre entre les différentes valeurs catégorielles encodées. Par contre, l'encodage OnehotEncoding en créant de nouvelles colonnes augmente les performances du modèle, plus spécifiquement dans notre cas ou on utilise les forêts aléatoires.

Pour finir, nous pouvons dire que d'après les résultats présents. Les meilleurs performances sont obtenues par l'approche 3 en entraînant notre dataset par le modèle Random Forest optimisé. Le

dataset constitué des: variables catégorielles encodées sous 'OneHotEncoding', mises à l'échelle en utilisant le 'MinMaxScaler' et les variables numériques avec la même mise à l'échelle.

2.5 Chronogramme

Ce travail a été réalisé durant une période de 20 semaines. La répartition des tâches durant ce stage est illustrée par le diagramme de gantt réel de la figure IV.1 ci dessus:

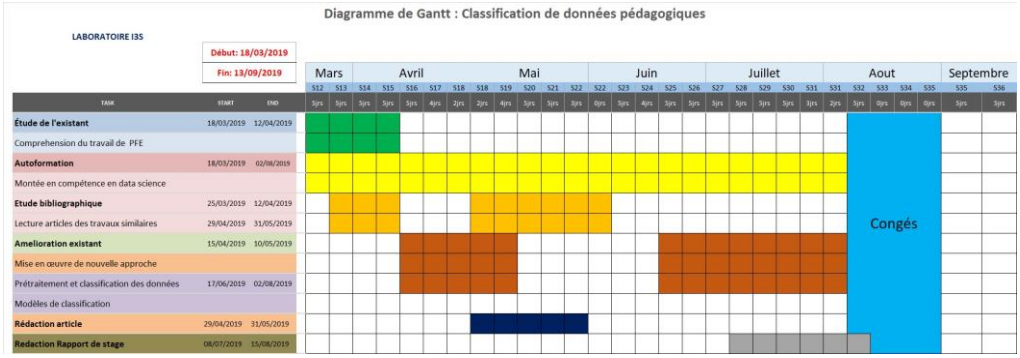


Figure IV.1 – Diagramme de gantt réel

Conclusion

A travers ce chapitre, nous avons justifié le choix des outils et technologies utilisés et présenter les résultats des démarches expliquées au chapitre 3.

Conclusion Générale et Perspectives

La prédiction de la performance des étudiants est aujourd'hui un sujet de recherche très fréquent au sein des universités. Le problème peut être traité en utilisant des modèles d'apprentissage supervisé sur des données bien préparées. Dans ce rapport, nous avons abordé un sujet de classification de données pédagogiques et envisagé des solutions de prédiction de la réussite des candidats dans le supérieur.

Pour mener à bien ce projet, nous avons réfléchis à des approches qui consistent en l'étude de l'impact de la mise à l'échelle des données sur la performance de l'algorithme de Random Forest. Nous avons appliqué 6 types de normalisations sur un dataset issu de deux années d'enregistrement par ParcourSup de données de lycéens et nous avons comparé les différentes performances de l'algorithme des Forêts aléatoires suivant les différentes optimisations apportées.

Dans le présent rapport, nous avons détaillé les étapes de pré-traitement de données et de classification de ces données pour aboutir à des performances discutables.

En perspective, quelques pistes d'améliorations peuvent être envisagées. En effet, nous recommandons de travailler sur le pré-traitement des données, la constitution des différents groupes et également sur la rapidité d'exécution des scripts. En effet, une approche réfléchie non mise en oeuvre ne peut pas être discutée tant qu'on ne dispose pas des résultats.

Bibliographie

[1] http://www.i3s.unice.fr/node/34. [En ligne; consulté le 18-03-2019]. ii, 2

[2] https://developers.google.com/machine-learning/. [En ligne; consulté le 17-06-2019]. ii, 6, 9

[3] Brijesh Kumar Baradwaj ET Saurabh Pal. *Mining Educational Data to Analyze Students' Performance*. (publié en Octobre 2011). 1

[4] Surjeet Kumar Yadav ET Saurabh Pal. Data Mining: A Prediction for Performance Improvement of Engineering Students using Classification. (publié le 17 Mars 2012). 1

[5] Mushtaq Hussain ET Raza Abidi. Using machine learning to predict student difficulties from learning session data. (publié Février 2018). 1

[6] https://dataanalyticspost.com/Lexique/random-forest/. [En ligne; consulté le 10-07-2019]. 7, 8

[7] Leo Breiman. *Random Forests*. (publié Janvier 2001). 8

[8] https://datafranca.org/wiki/index.php?title=Seuil_de_classification. [En ligne; consulté le 13-06-2019]. 11

[9] https://www.putty.org/. [En ligne; consulté le 02-09-2019]. 29

[10] https://www.python.org/downloads/. [En ligne; consulté le 25-03-2019]. 29

[11] https://scikit-learn.org/stable/. [En ligne; consulté le 22-04-2019]. 29

[12] https://pandas.pydata.org/. [En ligne; consulté le 25-03-2015]. 29

[13] https://www.numpy.org/. [En ligne; consulté le 25-03-2019]. 29

[14] https://matplotlib.org/. [En ligne; consulté le 25-03-2019]. 29

[15] https://seaborn.pydata.org/. [En ligne; consulté le 25-03-2019]. 29

[16] https://jupyter.org/. [En ligne; consulté le 08-04-2019]. 29

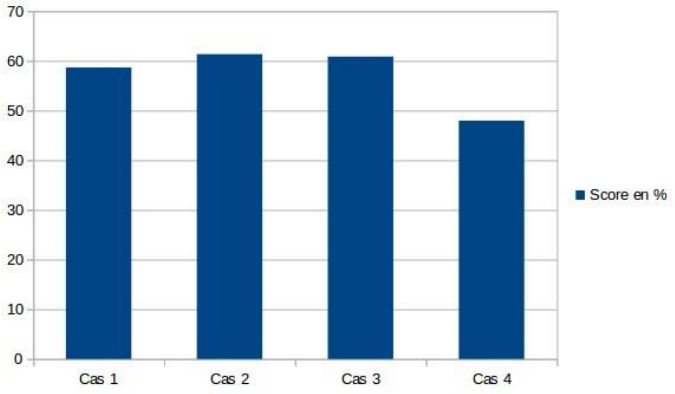
[17] https://www.jetbrains.com/pycharm/. [En ligne; consulté le 08-04-2019]. 29

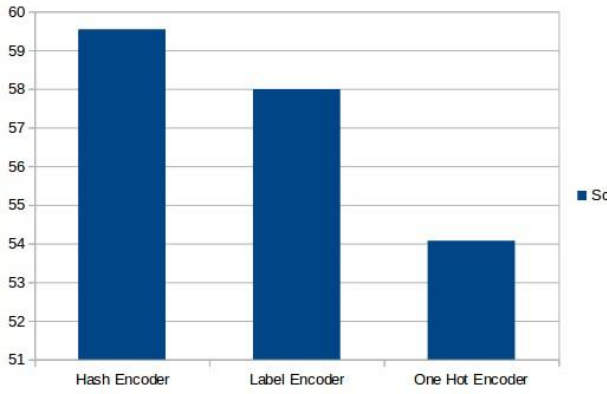
[18] https://www.anaconda.com/. [En ligne; consulté le 25-05-2019]. 29

Annexes

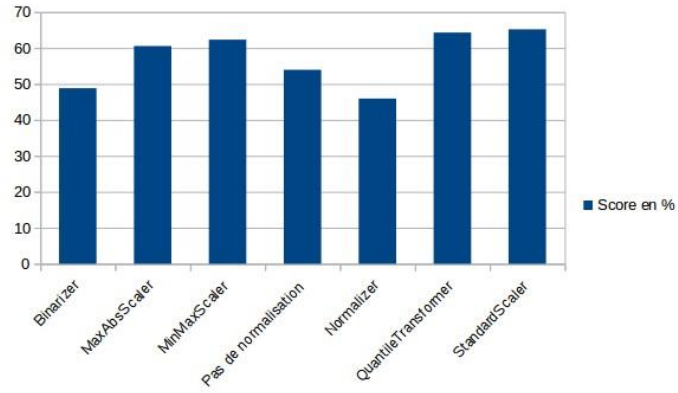
Rang	Score	Cas	Encodage	Normalisation	Modèle
1	0.9571	case4	Hash encoding	QuantileTransformer	DecisionTreeClassifier
2	0.9571	Cas4	Hash encoding	QuantileTransformer	GradientBoostingClassifier
3	0.8642	Cas2	Label encoding	MaxAbsScaler	PassiveAggressiveClassifier
4	0.8428	Cas4	One hot encoding	QuantileTransformer	LinearDiscriminantAnalysis
5	0.8333	Cas3	Hash encoding	StandardScaler	SVM - LinearSVC
6	0.8309	Cas2	Label encoding	MaxAbsScaler	GaussianNB
7	0.8309	Cas2	Label	encoding	No normalization GaussianNB
8	0.8309	Cas2	Label encoding	MinMaxScaler GaussianNB	
9	0.8309	Cas2	Label encoding	StandardScaler GaussianNB	
10	0.8285	Cas2	Hash encoding	QuantileTransformer	PassiveAggressiveClassifier
11	0.8285	Cas3	Hash encoding	StandardScaler	MLPClassifier
12	0.8285	Cas3	Hash encoding	MaxAbsScaler	GaussianNB
13	0.8285	Cas3	Hash encoding	MinMaxScaler	GaussianNB
14	0.8285	Cas3	Hash encoding	StandardScaler	GaussianNB
15	0.8166	Cas3	Hash encoding	StandardScaler	RandomForestClassifier
16	0.8166	Cas3	Hash encoding	StandardScaler	PassiveAggressiveClassifier
17	0.8166	Cas3	Label encoding	QuantileTransformer	SVM - NuSVC
18	0.8166	Cas2	Label encoding	MaxAbsScaler	SVM - NuSVC
19	0.8142	Cas2	Hash encoding	MaxAbsScaler	GaussianNB
20	0.8142	Cas2	Hash encoding	MinMaxScaler	GaussianNB

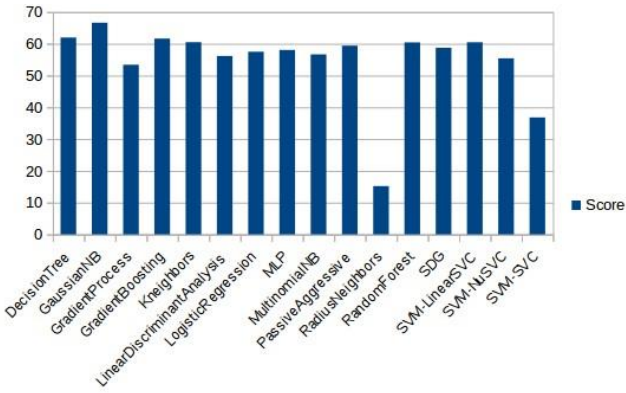
Tableau A.1 – Résultats des modéles avec combinaisons Cas/Encodage/Normalisation





(a) fig 1: Score selon le Cas **(b)** fig 2: Score selon l'encodage

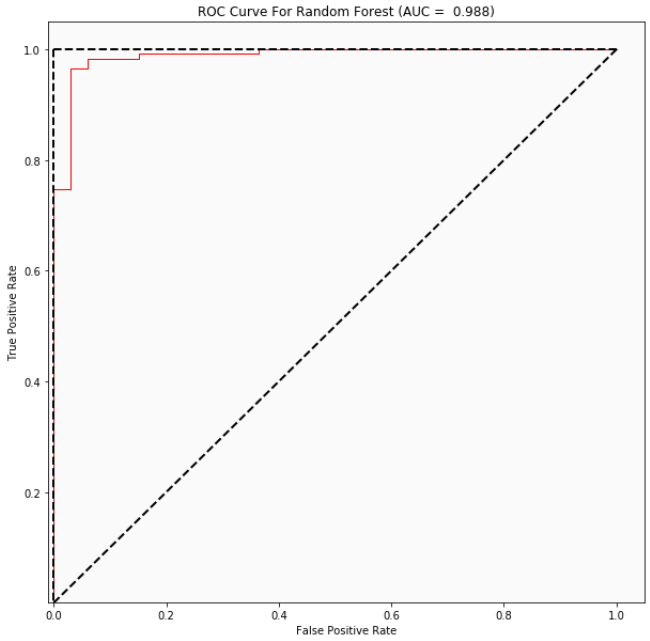




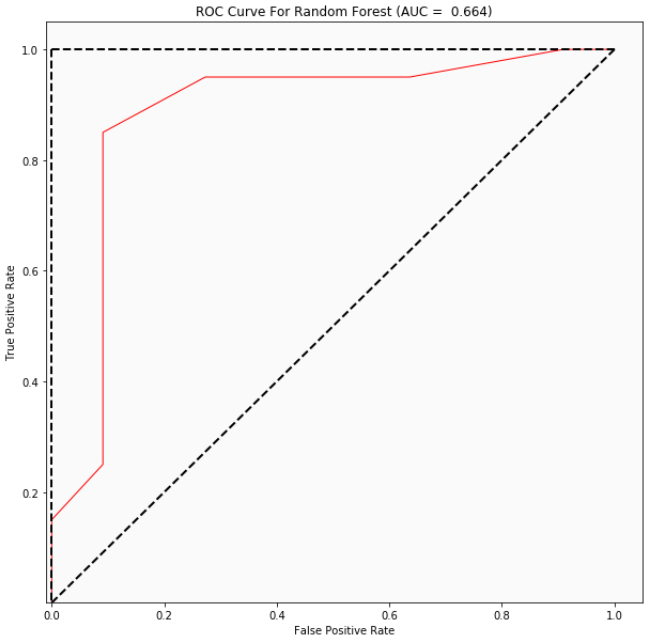
(c) fig 3: Score selon la mise à échelle **(d)** fig 4: Score des classifieurs

Figure A.1 – Résultats des score selon le cas, l'encodage, la mise à l'échelle et le modèle

(a) fig 1: Random forest défaut. **(b)** fig 2: Random forest optimisé.



(c) fig 3: Random forest optimisé avec Validation croisée.



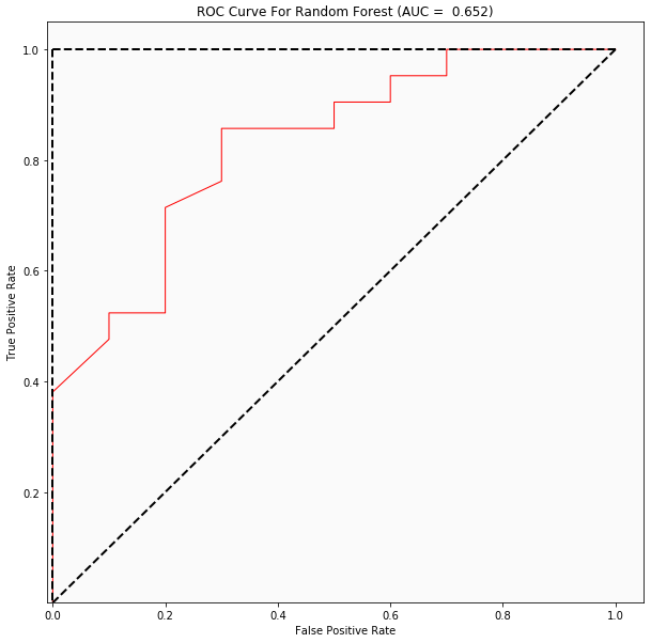
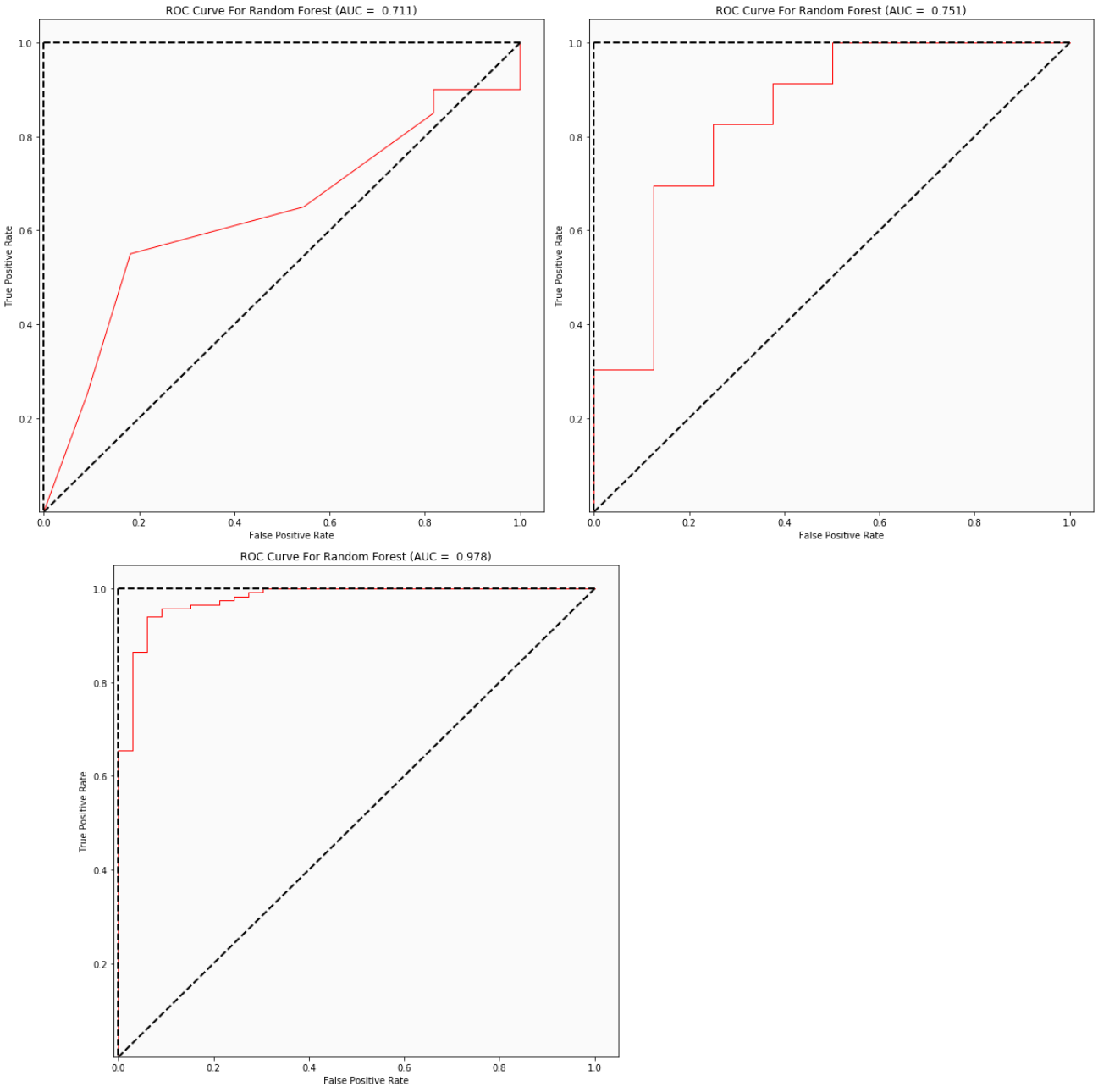


Figure A.2 – Courbe ROC et AUC de l'approche 1 selon l'encodage LabelEncoding avec les différents modèles. **(a)** fig 1: Random forest défaut. **(b)** fig 2: Random forest optimisé.



(c) fig 3: Random forest optimisé avec Validation croisée.

Figure A.3 – Courbe ROC et AUC de l'approche 1 selon l'encodage OneHotEncoding avec les différents modèles.

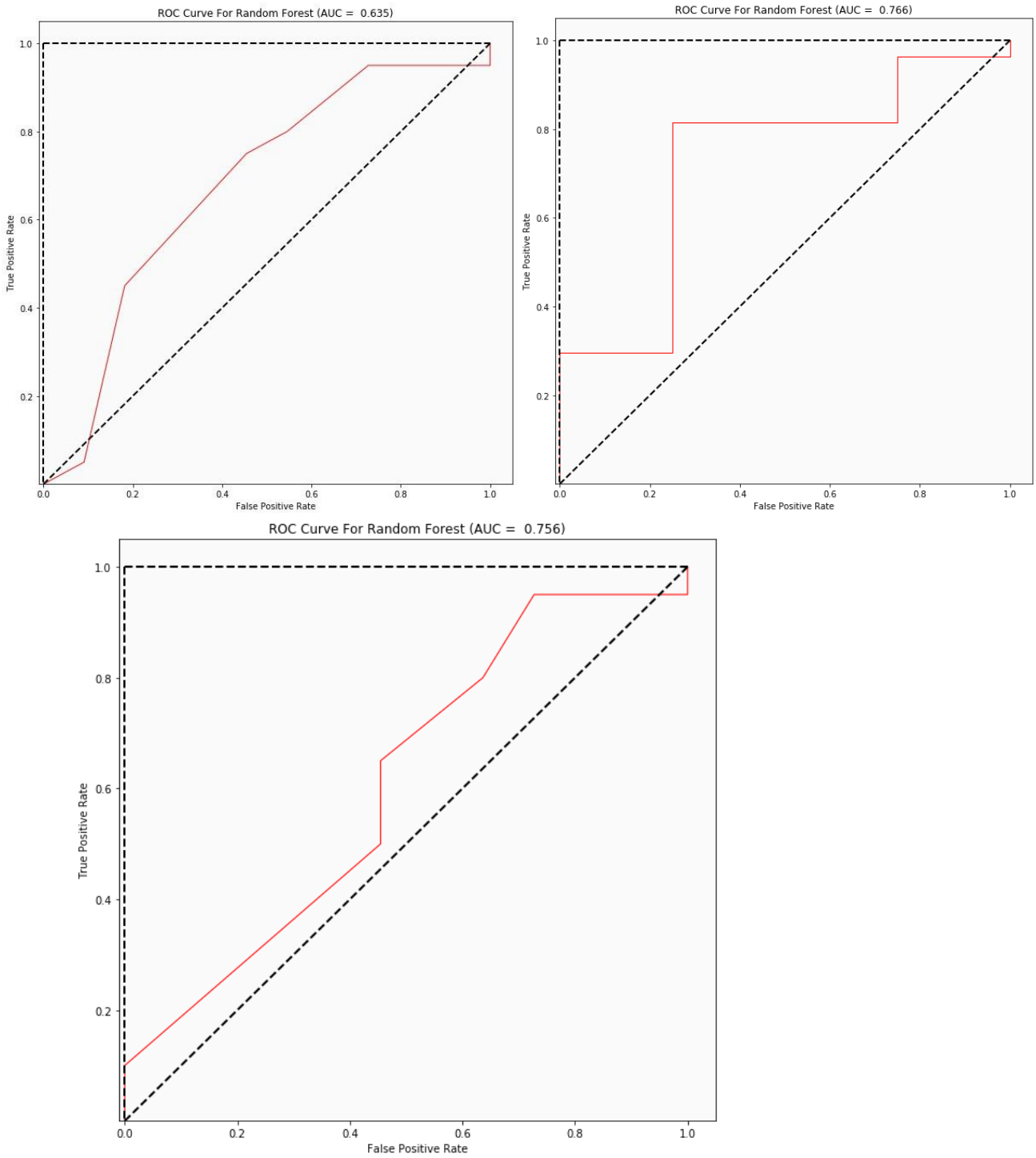
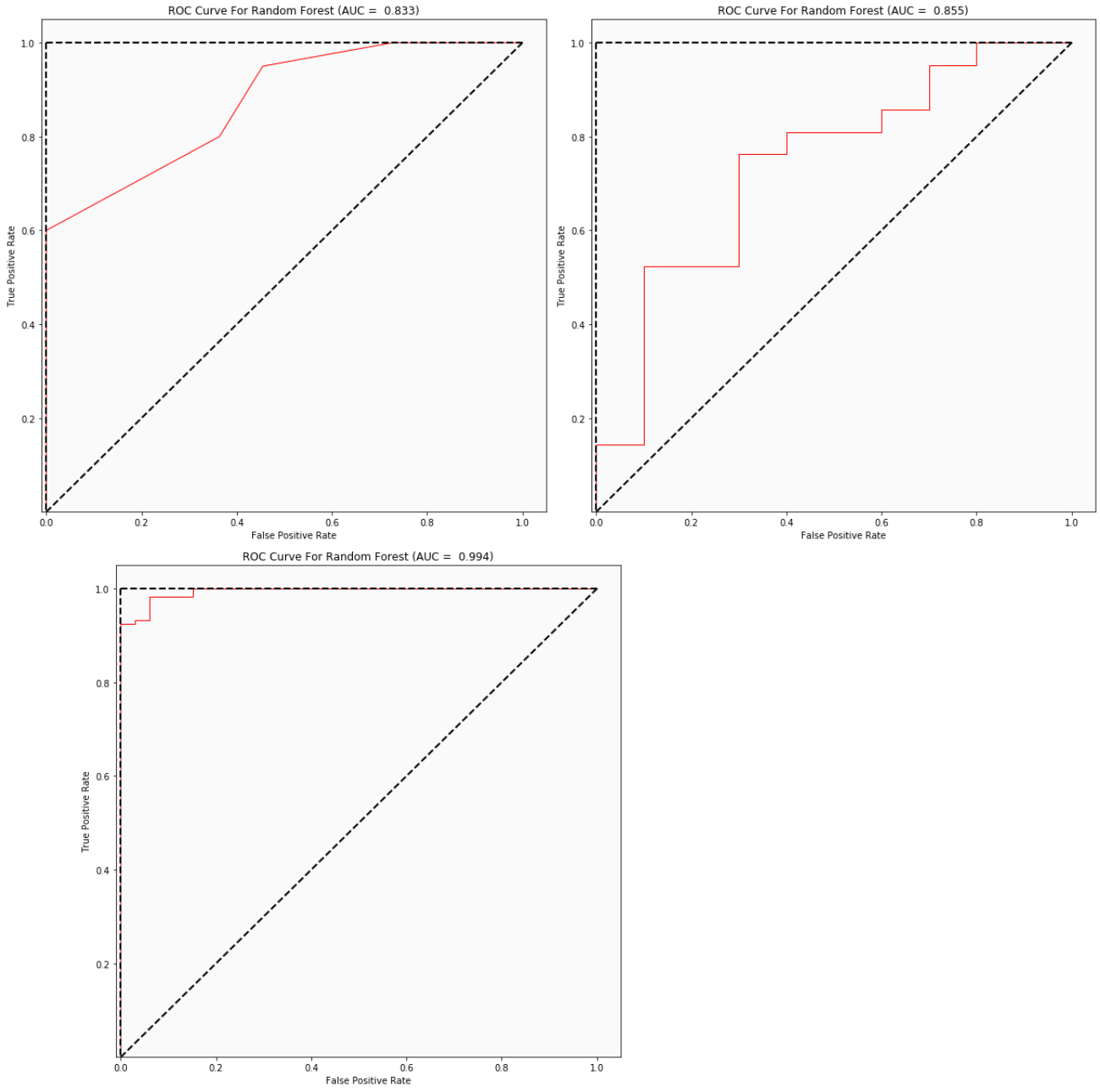
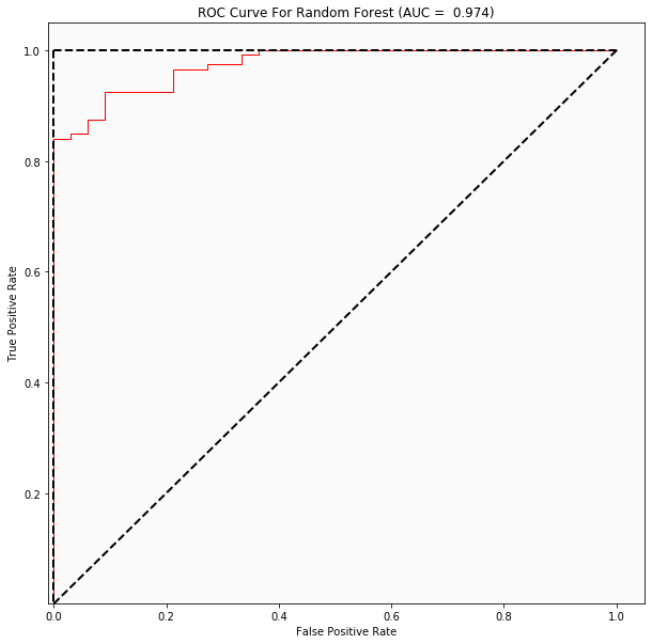


Figure A.4 – Courbe ROC et AUC du modèle Random forest par défaut pour l'approche 2. **(a)** fig 1: Random forest défaut. **(b)** fig 2: Random forest optimisé.



(c) fig 3: Random forest optimisé avec Validation croisée.

Figure A.5 – Courbe ROC et AUC de l'approche 3 selon l'encodage LabelEncoding avec les différents modèles. **(a)** fig 1: Random forest défaut. **(b)** fig 2: Random forest optimisé.



(c) fig 3: Random forest optimisé avec Validation croisée.

Figure A.6 – Courbe ROC et AUC de l'approche 3 selon l'encodage OneHotEncoding avec les différents modèles.

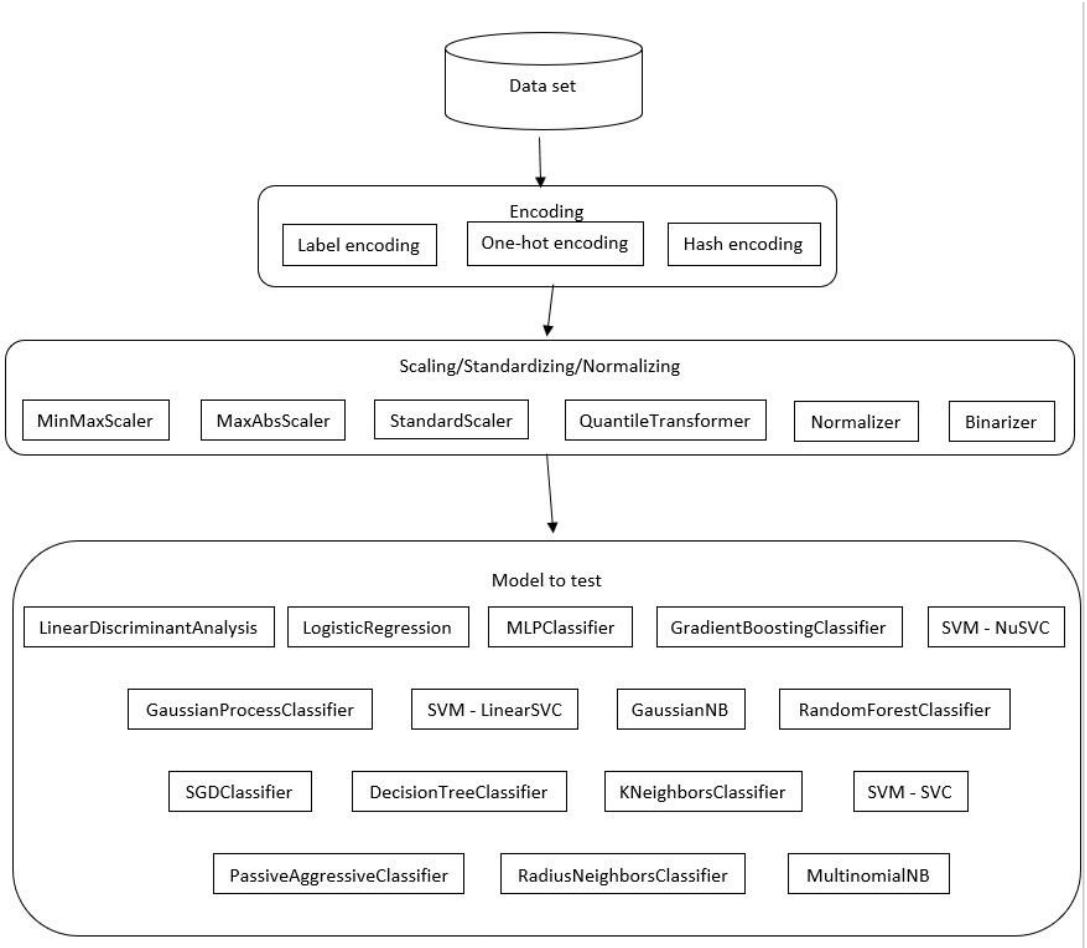


Figure A.7 – Schéma général de pré-traitement des données pour les modèles

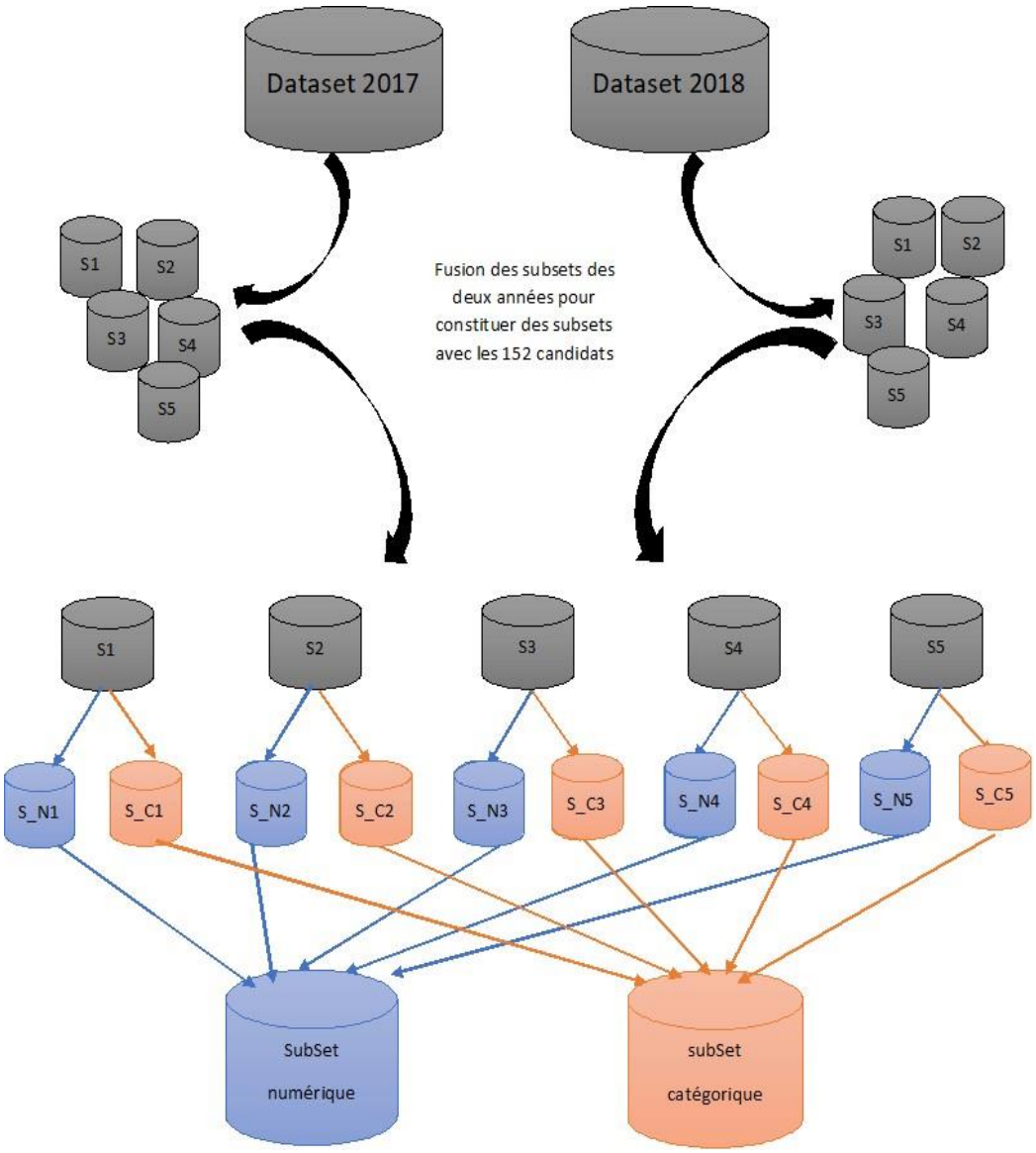
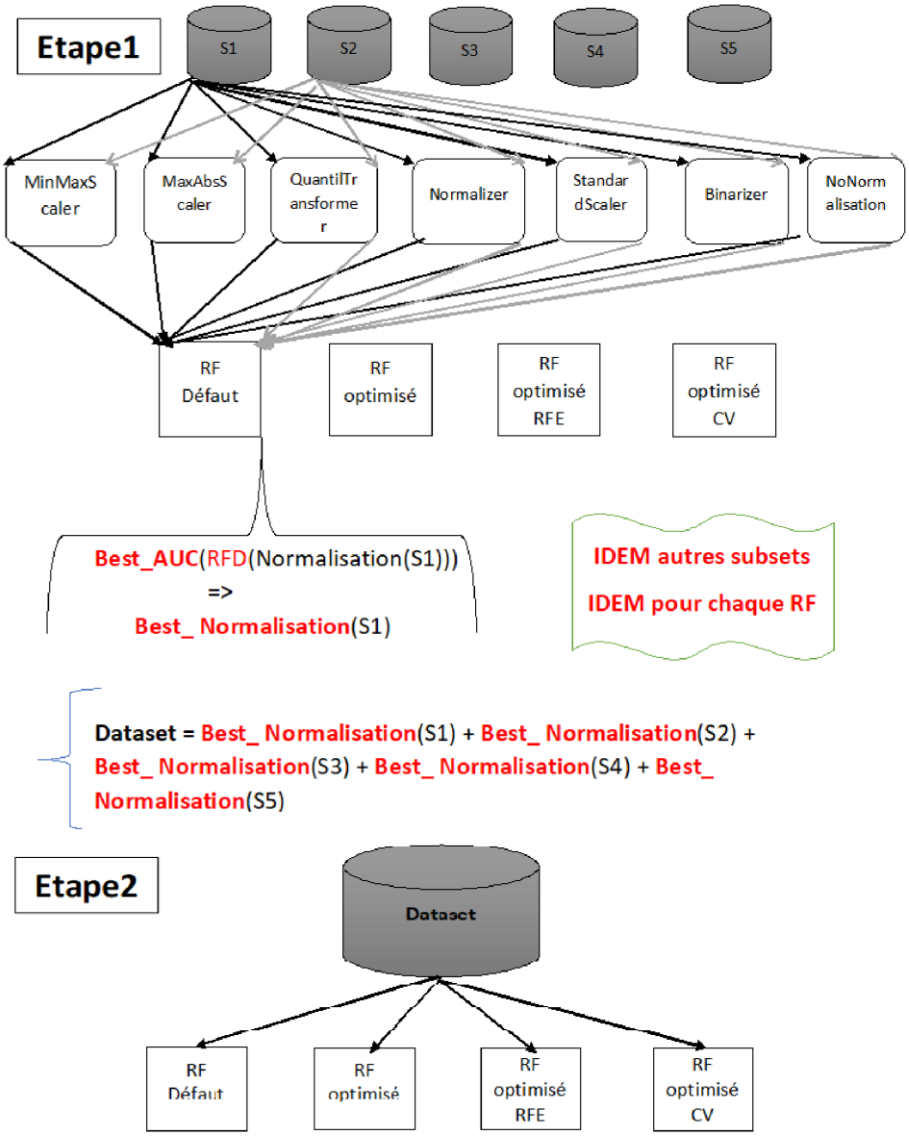

Dataset 2017
Dataset 2018
S1
S2
S3
S4
S5
Fusion des subsets des deux années pour constituer des subsets avec les 152 candidats
S1
S2
S3
S4
S5
S1
S2
S3
S4
S5
S_N1
S_C1
S_N2
S_C2
S_N3
S_C3
S_N4
S_C4
S_N5
S_C5
SubSet numérique
subSet catégorique

Figure A.8 – Schéma explicatif de la création des datasets pour les approches 3 et 4 **Figure A.9** – Schéma explicatif de l'approche 1



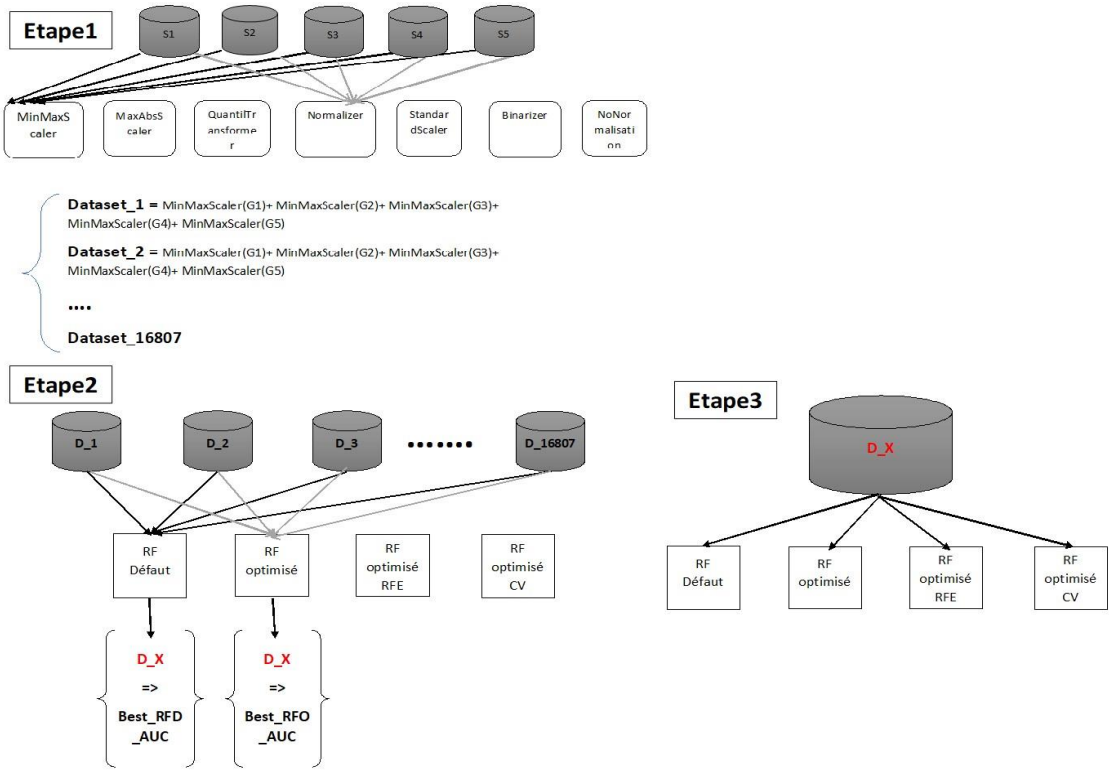


Figure A.10 – Schéma explicatif de l'approche 2