UNIVERSITÉ TOULOUSE III PAUL SABATIER Université de Toulouse

Identification et Classification de systèmes ABC

Dans le cadre de l'UE "Fouille de données" par R. Barriot

CHENEL Hugo
GHEZIEL Nadine

M1 "Biologie Informatique et Biologie des Systèmes"

- **INTRODUCTION**
  - **Contexte**
  - **Objectifs du projet**
  - **Informations disponibles**

- **ANALYSE**
  - **Données fournies**
  - **Tables**
    - **Taxonomy, Strain and Chromosome**
    - **Gene**
    - **Conserved_Domain and Functional_Domain**
    - **Orthology**
    - **Protein and Assembly**

- **CONCEPTION**
  - **Matrice individus-variables**
  - **Knime**

- **RÉALISATION, INTERPRÉTATION ET DISCUSSION**

- **BILAN ET PERSPECTIVES**
  - **Gestion du projet**

❑ CONTEXTE

L'objectif de ce projet consiste en la mise en place d'une méthode permettant l'annotation de génomes complets de procaryotes en termes de systèmes ABC.

Les systèmes ABC (ATP Binding Cassette) constituent un large ensemble de protéines transmembranaires. Leur rôle est de transporter unidirectionnellement divers substrats à travers la membrane cytoplasmique. L'hydrolyse de l'ATP est nécessaire comme source d'énergie pour le transport. La libération d'un groupement phosphate et d'ADP est générée: il s'agit d'un transport actif primaire.

Ces systèmes formant une très grande famille multigénique sont retrouvés dans les 3 règnes du vivant (eucaryotes, archées et procaryotes). Chez ces derniers, ils sont la plupart du temps impliqués dans l'efflux de molécules toxiques comme les antibiotiques.

L'architecture des transporteurs est conservée au sein de la majorité de ces systèmes. Cependant, leur organisation en domaine est variable. On observe généralement 2 domaines pour les exportateurs et 3 domaines pour les importeurs :

- MSD : Membrane Spanning Domain. 2 domaines MSD (hétéro ou homo-dimère) forment le pore à travers la membrane.

- NBD : Nucleotide Binding Domain. 2 domaines (hétéro ou homo-dimère) fournissent l'énergie pour le transport actif par hydrolyse de l'ATP.

- SBP : Solute Binding Protein. 1 domaine capture le substrat et l'amène à l'entrée du pore.

De plus, certains domaines peuvent être portés par un même gène ou des gènes différents, mais on peut également retrouver un seul gène pour plusieurs domaines.

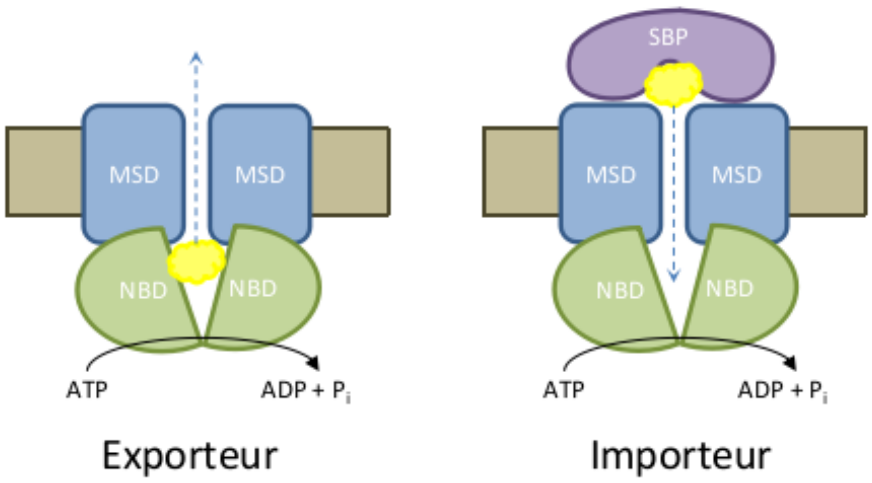


Fig.1 : Architecture des transporteurs ABC
http://silico.biotoul.fr/enseignement/m1/datamining/projet/sujet.html

Par exemple :

- MSD-MSD : un gène contient 2 domaines MSD
- MSD-NBD : un gène arbore un domaine MSD suivi d'un domaine NBD

De plus, la structure peut être homodimérique (2 protéines codées par le même gène) ou hétérodimérique (2 protéines codées par 2 gènes différents).

Ces systèmes étant très anciens, ils se sont fortement diversifiés au cours de l'évolution avec l'accumulation de mutations sur la séquence de leurs gènes. A la suite d'analyses de similarité de séquence**,** il a été possible de les classer en une vingtaine de sous-familles déterminées. Cette similarité de séquence indique que les molécules transportées sont similaires (cela n'est pas transposable à toutes les familles multigéniques). Une seule mutation peut suffire à modifier complètement la fonction.

Les techniques de séquençage génomique ont progressé ces dernières années, permettant d'obtenir les séquences génomiques complètes de différents organismes. La génomique produit des volumes conséquents de données.

L'annotation des génomes permet de trier et organiser ces données afin de leur donner du sens. Il pourrait être judicieux d'utiliser les données disponibles relatives aux systèmes ABC afin d'annoter automatiquement les informations liées aux systèmes ABC d'un génome.

❑ OBJECTIFS

Le data mining, ou Knowledge Discovery in Databases (KDD), est un processus permettant l'extraction des connaissances intéressantes ou des motifs/patterns à partir d'une grande quantité de données. Composante essentielle des techniques d'analyse des grands jeux de données, cette technique permet l'exploration et l'analyse de données volumineuses, la transformation de ces données en information utile, et éventuellement la recherche de relation entre les données.

En d'autres termes, le data mining consiste en la découverte de connaissances dans les bases de données.

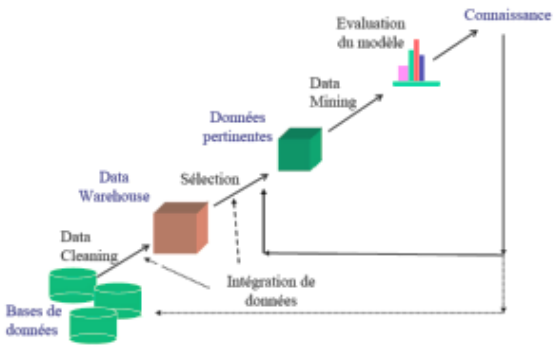


Fig. 2 : Processus d'extraction de connaissances à partir de bases de données

https://touriaelouahabi.wordpress.com/ecbd/concepts-et-principes-du-data-mining/

Une succession d'étapes est impliquée dans le processus de découverte de connaissances:

- sélection des données à l'aide de la création du jeu de données cible (intégration)
- nettoyage et prétraitement des données
- réduction et transformation des données (normalisation)
- choix des fonctionnalités de data mining et des algorithmes
- recherche de motifs intéressants
- représentation de connaissances après évaluation de ces motifs

Cette procédure aboutit à la découverte de connaissances.

Deux types d'apprentissages sont utilisés en fouille de données :

-Le premier est un apprentissage non supervisé, les classes sont alors inconnues. L'objectif est de, à partir de n observations, constituer k groupes tels que ces groupes soient constitués d'observations semblables, mais qu'ils soient le plus différents possibles entre eux. On parle de clustering.
Il existe des méthodes non hiérarchiques, dites par partitionnement, qui consiste en la construction de k partitions qui seront corrigées jusqu'à obtenir une similarité parfaite. L'exemple le plus connu est la méthode du k-means.
Les méthodes hiérarchiques consistent en la création d'une décomposition hiérarchique par agglomération ou division de groupes similaires ou dissimilaires(ex : Hierarchical clustering).

-Le second est un apprentissage supervisé : étant donné un ensemble de classes connues, établir les « meilleures » règles de classement, le jeu de données d'apprentissage fournit donc les classes des objets. Ce type d'apprentissage correspond à la classification.
L'objectif de cet apprentissage est de modéliser la relation entre les observations et la classe d'appartenance, mais également d'identifier la classe d'appartenance d'un objet à partir d'un ensemble de descripteurs. De nombreux outils existent, tels que la méthode des K plus proches voisins, ou celle de l'arbre de décision.

Les arbres de décisions sont des outils d'aide à la décision : une procédure de classification est construite suivant un protocole simple tout en offrant une interprétabilité agréable aux utilisateurs. Les variables discriminantes seront choisies par le logiciel selon une suite de calculs statistiques.

La méthode des K plus proches voisins (ou knn pour "k-nearest neighbors") se base sur un algorithme dit paresseux : il n'apprend rien pendant la phase d'entraînement. La prédiction d'une classe se base sur un concept simple : l'algorithme reçoit une nouvelle donnée et cherche à calculer ses k plus proches voisins (selon une distance euclidienne par exemple). La donnée reçoit la même classe que la classe majoritaire de ses k plus proches voisins.

Il existe plusieurs manières d'évaluer un classificateur :

La validation croisée est une méthode d'estimation de la fiabilité d'un modèle basé sur une technique d'échantillonnage. Son principe consiste à diviser les données en k-partitions puis

utiliser k-1 partitions pour l'apprentissage et la dernière pour le test. Après l'apprentissage, on peut calculer une performance de validation.

Divers autres paramètres permettent l'évaluation d'un classificateur :
-Calcul de la spécificité : capacité à ne détecter que des vrais positifs
-Calcul de la sensibilité : capacité à détecter un maximum de vrais positifs
-Courbe ROC : graphique des performances du modèle, vrais positifs en fonction des faux positifs

-AUC : pour l'aire sous la courbe ROC : entre 0 et 1, c'est une valeur qui fournit la capacité discriminatoire du modèle.

La classification est le type d'apprentissage retenu pour ce projet afin de classer les gènes selon leur sous-type de protéine ABC.

❑ INFORMATIONS DISPONIBLES

ABCdb est une banque de données publique dédiée aux transporteurs ABC. Elle a été encodée à partir de génomes procaryotes intégralement séquencés. On dispose donc de données sur les génomes expertisés permettant la classification automatique.

Comment définir si un gène code ou pas pour un partenaire d'un système ABC ?

Deux informations sont pertinentes parmi celles à notre disposition :

- les domaines présents sur la séquence protéique correspondant au gène à partir de banques de données de domaines
- l'orthologie de type 1:1 entre les gènes des génomes expertisés sachant que l'on connaît leur architecture en domaine et leur sous-famille.

❖ ANALYSE

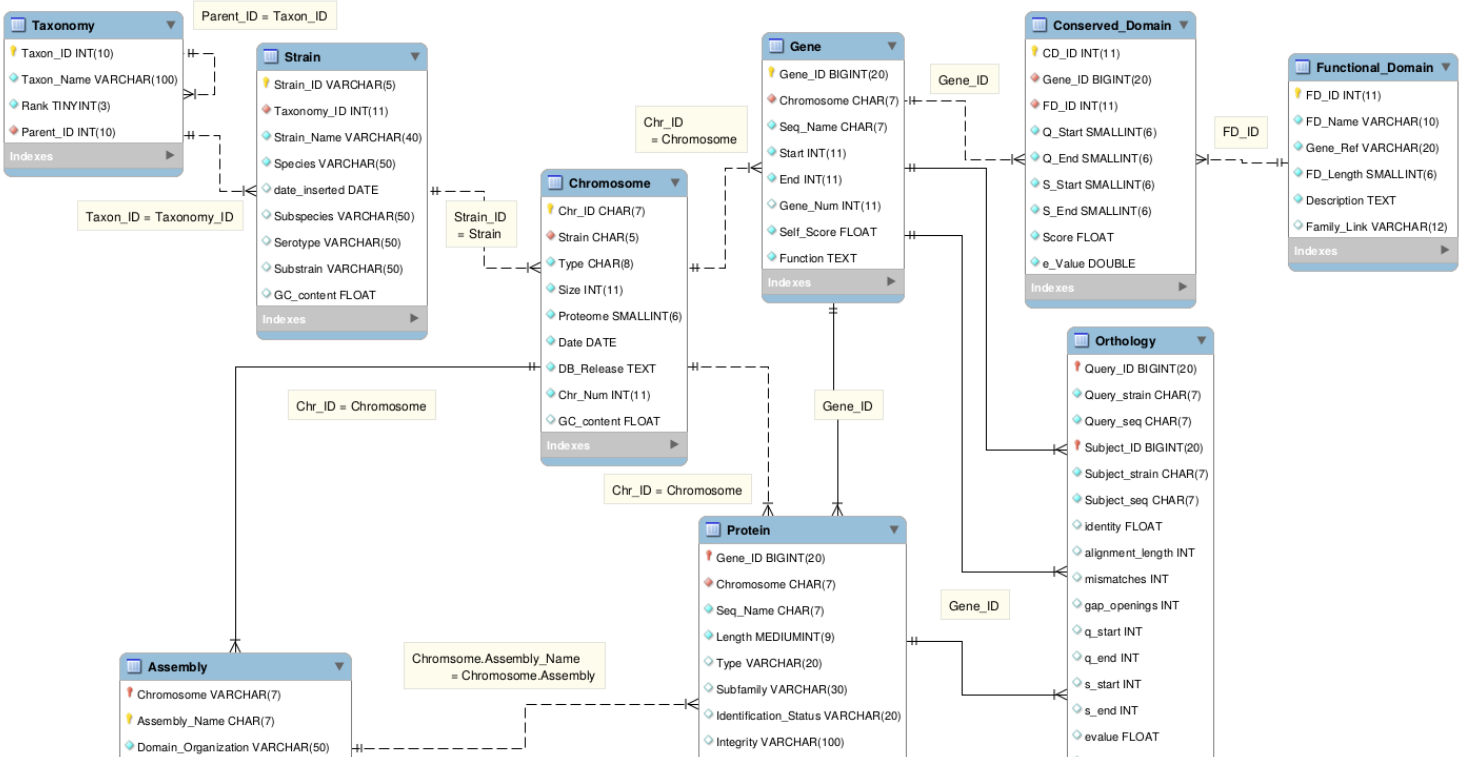


Fig 3 : Schéma Base de données

http://silico.biotoul.fr/enseignement/m1/datamining/projet/sujet.html

Dans un premier temps, il est important de choisir les tables jugées pertinentes à notre analyse. Selon une réflexion menée sur notre problématique, et sur les attributs contenus dans les tables proposées, nous avons portés nos choix sur les suivantes :

- Gene
- Protein
- Conserved_Domain
- Functional_Domain

Les tables contiennent certains attributs pertinents à notre analyse : Gene_ID et FD_ID nous permettront de faire correspondre nos 4 tables d'intérêts.

❖ CONCEPTION

❏ Matrice individus-variables

Notre matrice individus-variable a été construite à partir du logiciel R version 3.04. Le gestionnaire de bases de données utilisé est le serveur MariaDB version 10.5.9. L'utilisation du serveur depuis R à l'aide de la librairie RMySQL permet d'effectuer des liens entre ces tables grâce à des requêtes SQL, et ainsi tirer des données pertinentes pour la suite de l'analyse dans le but d'effectuer notre classification.

Gene ID correspondent à nos individus (soient les lignes de la matrice).

Les variables (attributs) choisis pour mener à bien la classification sont :

- Gene_ID depuis Gene
- FD_ID depuis Functional_Domain
- Length depuis Functional_Domain
- S_Start et S_End et Score depuis Conserved_Domain
- Family_Link depuis Functional_Domain

Dans un premier temps, il sera nécessaire de créer une base de données MariaDB contenant la totalité de nos tables. La création des tables à l'aide de requêtes est automatisée grâce à l'utilisation du document "create.and.populate.fouille.db.sql" fourni contenant la totalité des requêtes.

Une fois cette étape réalisée, il faut créer une connexion à MariaDB depuis le document R à l'aide des informations de connexion et de l'adresse IP de la machine utilisée :

```
dbh=dbConnect(MySQL(),
              user="guest",
              password="bioinfo",
              dbname="fouille",
              host="195.220.42.4")
```

La première requête permet de récupérer la majorité des colonnes de notre matrice :

```
query = "SELECT Gene.GENE_ID, Functional_Domain.FD_length,
Functional_Domain.FD_ID, Conserved_Domain.e_Value
              FROM Functional_Domain, Conserved_Domain, Gene
              WHERE Functional_Domain.FD_ID = Conserved_Domain.FD_ID
              AND Conserved_Domain.GENE_ID = Gene.GENE_ID
              AND Gene.Function like 'ABC%'
              AND Conserved_Domain.e_Value < 0.005"
```

La sélection indique que sont récupérées :

- Le Gene_ID depuis la table Gene (soit l'identifiant unique qui permet de rendre unique chaque individu de la matrice) mais également de croiser les tables : c'est la clé primaire et secondaire.
- Length depuis la table Functional_Domain (soit la longueur du domaine fonctionnel)
- Le FD_ID (identifiant unique du domaine fonctionnel retrouvé)
- La e-value depuis Conserved_Domain qui permettra de filtrer les doublons

Les tables Gene et Conserved_Domain se croisent grâce au GENE_ID, Conserved_Domain et Functional_Domain grâce au FD_ID. Un premier filtre est effectué sur les e_value < 0.005 pour optimiser les requêtes, et dans Gene : le gène qui doit coder pour une protéine fonctionnelle ABC.

Il est important de retirer les doublons afin que l'analyse soit menée avec des individus uniques mais également afin de faciliter la commande merge qui suivra plus tard.

```
The_table1 <- df2 %>% group_by(GENE_ID)  %>% filter(e_Value ==
min(e_Value))
```

The_table1 reçoit notre première table (df2) rangée par GENE_ID qui seront filtrés : si des doublons de GENE_ID sont présents, seul l'individu ayant la e_value la plus faible sera conservé dans la matrice afin de garder les résultats les plus significatifs possibles.

Dans un second temps, il faut récupérer le score normalisé également sélectionné pour cette analyse. La table Orthology étant très lourde (plusieurs millions d'individus), il est important de filtrer les résultats dans un but d'efficience du script.

```
query = "SELECT subject_id, norm_Score
          FROM  Orthology
          WHERE evalue <= 0.00005 AND subject_id IN (SELECT Gene_ID
                                                     FROM Protein
                                                     WHERE Type = 'ABC')"
```

Cette requête permet une sélection du subject_id (équivalent au GENE_ID donc qui nous permettra le lien entre les deux tables) et du score_norm d'intérêt pour notre analyse. Les filtres s'effectueront via une e-value stringeante et des types de protéines uniquement ABC.

```
The_table2 <- dft %>% group_by(subject_id)  %>% filter(norm_Score ==
max(norm_Score))

colnames(The_table2)[1] <- "GENE_ID"
```

Là encore, il est nécessaire de retirer les doublons, mais cette fois en ne gardant que les individus dont le score normalisé sera le plus grand.

Enfin, la colonne subject_id sera renommée en GENE_ID afin de faciliter la requête du merge entre nos deux tables.

```
table =  merge(The_table1, The_table2, by = "GENE_ID", all.x = TRUE,
all.y = FALSE)

matrice = na.omit(table)

matrice1 <- matrice1[,-4:-5]
```

Le merge permet de fusionner nos deux tables sur la base du GENE_ID seulement pour les individus présents dans la première table (The_table1). La seconde requête par le *na.omit* permet de retirer les lignes contenant des données NA (not available).

Nous retirons alors les colonnes correspondant à la e_value et au norm_score qui nous ont permis d' améliorer la qualité du filtrage mais ne figureront pas dans l'analyse KNIME.

```
conserved <- conserved[,-1]
conserved <- conserved[,-2:-4]
conserved <- conserved[,-4:-5]

colnames(conserved)[1] <- "GENE_ID"

matrice2 =  merge(matrice1, conserved, by = "GENE_ID", all.x = TRUE,
all.y = FALSE)
```

Nous sélectionnons ensuite depuis la table conserved_domain les positions start et end et le score des séquences ainsi que leur Gene_ID renommé en Gene_ID. Le merge permettra d'ajouter à nos séquences sélectionnées qui codent pour les protéines de type ABC leur séquences start et end, ainsi que le score.

```
functional_domains <- functional_domains[,-2:-5]
functional <- functional_domains %>% group_by(FD_ID)  %>%
filter(Family_Link !="")
matriceFinal = merge(matrice2, functional, by = "FD_ID")
```

Depuis functional_domains, nous sélectionnons les sous-types de familles ABC, soit la colonne Family_link qui nous permettra de construire nos modèles de classifications. Un filtre est effectué afin de supprimer les lignes contenant des cases vides (une majorité de la table). Le merge est alors effectué avec la précédente matrice retenue.

```
matriceFinal <- matriceFinal %>% group_by(GENE_ID)
matriceFinal <- matriceFinal[,-1]
write.table(matriceFinal, file="mymatrix.txt", row.names=FALSE,
col.names=TRUE, sep = '\t', append = FALSE, quote = FALSE)
```

Les dernières étapes consistent en :

- la vérification de l'absence de doublons
- la suppression de la colonne GENE_ID qui ne sera plus utile à l'analyse
- l'écriture de la matrice dans un fichier de sortie

Nous obtenons alors la matrice suivante, composée de 4523 individus :

FD_ID	FD_length	S_Start	S_End	Score	Family_Link
90002	287	91	125	24	M_10ab1
90002	287	238	266	27	M_10ab1
90002	287	9	88	25	M_10ab1
90002	287	12	265	93	M_10ab1
90002	287	4	265	122	M_10ab1
90002	287	3	267	95	M_10ab1
90002	287	3	278	178	M_10ab1
90002	287	475	494	22	M_10ab1
90002	287	1	285	345	M_10ab1
90002	287	304	316	21	M_10ab1
90002	287	11	282	195	M_10ab1
90002	287	1	287	396	M_10ab1
90002	287	152	190	22	M_10ab1

❏ **Knime**

Nous allons maintenant utiliser des méthodes d'apprentissage proposées par le logiciel KNIME, un logiciel open source permettant la construction de workflow précis. Le workflow

sera constitué des différentes étapes nécessaires à sa réalisation depuis les lecture et paramétrage des données, à l'analyse et la visualisation des résultats

KNIME propose l'utilisation d'un nombre étendu de tests statistiques et algorithmes dédiés à la fouilles de données, nous en utiliserons plusieurs afin d'interpréter, mais également de comparer les résultats obtenus avec plusieurs méthodes.

Construction du modèle

Parmi les différents types de classificateurs, les tests ont été effectués sur l'arbre de décision (Decision Tree Learner) et le classificateur bayésien naïf (Naïve Bayes Classifier).

1) Decision Tree Learner

La génération de l'arbre nécessaire à la classification se fait en deux étapes :

- Construction : initialement, les exemples du jeu d'apprentissage se situent à la racine, puis s'effectue une partition récursive des exemples en sélectionnant les attributs.
- Élagage permettant d'identifier supprimer des branches correspondant à des exceptions ou du bruit.

Une mesure de qualité intéressante est le coefficient de Gini. En effet, cette mesure statistique permet d'établir la répartition d'une variable au sein d'une population. Autrement dit, il mesure la dispersion d'une distribution dans la population.

Un piège à éviter dans cette modélisation est l'overfitting ; une sur-modélisation du jeu d'apprentissage dont l'arbre généré risque de trop refléter le jeu d'apprentissage. Pour y remédier, une méthode permet d'évaluer les sous-arbres à élaguer (pruning) : le principe MDL (minimum description length).

2) Classificateur bayésien naïf

Nous sommes dans le cas d'un apprentissage et d'une prédiction probabiliste. Le terme naïf signifie que les descripteurs sont conditionnellement indépendants.

La méthode se construit de la façon suivante :

- Détermination d'un ensemble d'apprentissage
- Détermination des probabilités à priori de chaque classe (données d'observation)
- Application du théorème de Bayes $P(C/X)=(P(X/C)*P(C))/P(X)$ afin d'obtenir la probabilité à posteriori des classes
- Choix de la classe la plus probable

Le workflow se construit selon le modèle suivant pour les 2 types de classification :

- Le nœud *File Reader* permet le chargement des données. On s'assure de configurer correctement les types de nos attributs, en particulier FD_ID en tant que *String* et non *Integer*.
- Le nœud *Decision Tree Learner* ou *Naive Bayes Learner* est configuré de telle sorte que la colonne de classification corresponde à *Family_Link*.

Evaluation d'un classificateur : performances

Si une classe est peu présente dans le jeu de données initial, le partitionnement peut entraîner la création d'un jeu d'apprentissage et d'un jeu de test dont le nombre de classes sera différent. Pour y remédier, on utilisera des validations croisées.

Ces validations croisées sont intégrées dans Knime (”Cross Validation”). La construction s'effectue de la manière suivante :

- Le nœud X-Partitioner est placé entre le File Reader et le Learner. Son rôle est de diviser le jeu de données en jeu d'apprentissage pour le Learner et en jeu de test pour le Predictor. Nous avons testé différents nombre de validations, les résultats suivants pour toutes les méthodes d'échantillonnages auront des 10 et 50 Cross Validation.
- Le nœud X-Aggregator situé après le Predictor permet une confrontation de la classe permet une confrontation de la classe prédite à la classe connue pour chacun des objets testés.
- en fin de Workflow, le noeud Statistics affiche le taux d'erreur moyen de notre modèle tandis que le noeud Scorer nous apporte des informations complémentaires tel que le nombre de VP, FP, VN, FN et ainsi la spécificité (capacité à ne détecter que des vrais positifs) et la sensibilité (capacité à détecter un maximum de vrais positifs).

La validation croisée “Leave-one-out” (validation croisée avec k=s où k représente le nombre de partitions) n'a pas été effectuée car les temps de calculs étaient trop importants, cela est lié à la taille conséquente de la matrice individu x variable.

❖ RÉSULTATS

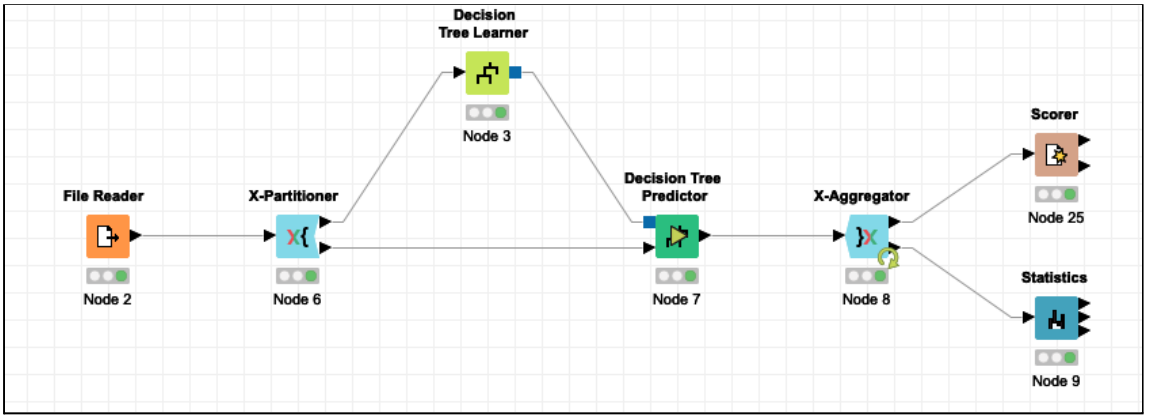


Fig 4 : KNIME *workflow* for Decision Tree model.

Les résultats obtenus pour l'arbre de décision sont représentés dans le tableau ci-dessous :

Sampling method	*Number of validations*	*Parameters*	*Mean percentage error*
Linear sampling	10	Gain ratio / No pruning	6.520 %
		Gain ratio / MDL	6.476 %
		Gini index / No pruning	6.507 %
		Gini index / MDL	6.485 %
	50	Gain ratio / No pruning	2.985 %
		Gain ratio / MDL	3.31 %
		Gini index / No pruning	3.034 %
		Gini index / MDL	3.311 %

Sampling method	*Number of validations*	*Parameters*	*Mean percentage error*
Random sampling	10	Gain ratio / No pruning	0.720 %
		Gain ratio / MDL	0.815 %
		Gini index / No pruning	0.703 %
		Gini index / MDL	0.762 %
	50	Gain ratio / No pruning	0.68 %
		Gain ratio / MDL	0.71 %

		Gini index / No pruning	0.687 %
		Gini index / MDL	0.738 %

Sampling method	*Number of validations*	*Parameters*	*Mean percentage error*
Stratified sampling	10	Gain ratio / No pruning	0.687 %
		Gain ratio / MDL	0.762 %
		Gini index / No pruning	0.676 %
		Gini index / MDL	0.747 %
	50	Gain ratio / No pruning	0.665 %
		Gain ratio / MDL	0.696 %
		Gini index / No pruning	0.676 %
		Gini index / MDL	0.705 %

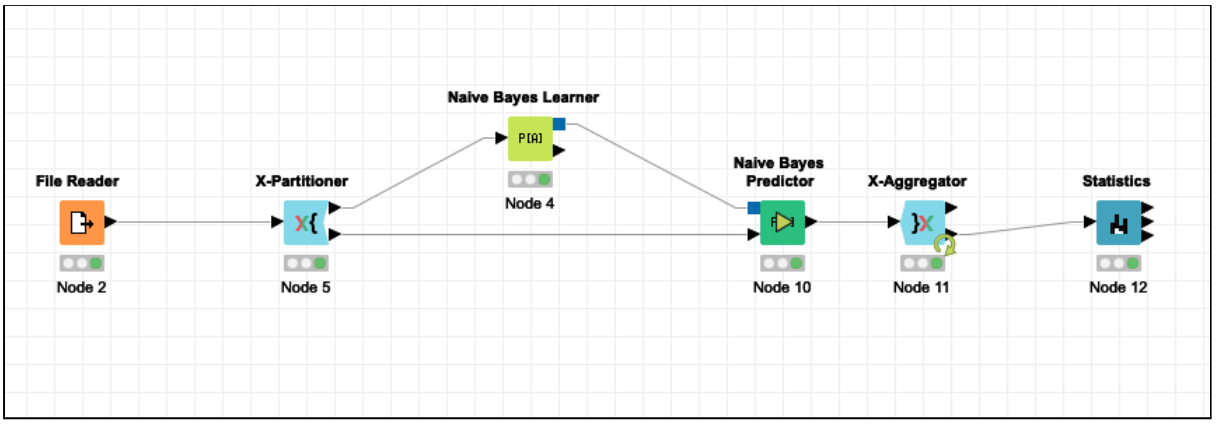


Fig 5 : KNIME *workflow* for **Naïve Bayes model**.

Les résultats obtenus pour classification *naïve* bayésienne sont représentés dans le tableau ci-dessous :

Sampling method	*Number of validations*	*Mean percentage error*
Linear sampling	10	13.098 %
	50	5.402 %

Sampling method	*Number of validations*	*Mean percentage error*
Random sampling	10	1.316 %
	50	1.312 %

Sampling method	*Number of validations*	*Mean percentage error*
Stratified sampling	10	1.314 %
	50	1.309 %

❖ DISCUSSION

L'ensemble des résultats est satisfaisant. L'arbre de décision et le classificateur bayésien semblent être des types de classification pertinent pour notre jeu de données. En effet, hormis pour le linear sampling, le taux d'erreur de ces deux types de classification est inférieur à 2% peu importe les paramètres sélectionnés.

Plusieurs observations :

- Le random et stratified sampling permettent d'obtenir un taux d'erreur moyen bien inférieur à celui obtenu à l'aide d'un linear sampling.
- Plus le nombre de validations est élevé, plus la performance du modèle est grande.
- Au niveau des paramètres, on ne peut pas déterminer l'influence du coefficient de Gini sur le taux d'erreur. Toutefois, on peut remarquer que les arbres n'ayant pas subi d'élagage ont un taux d'erreur plus faible.

Le taux d'erreur moyen le plus faible est obtenu lorsqu'on utilise un arbre de décision avec un stratified sampling, 50 validations croisées, gain ratio et no pruning. Le taux d'erreur moyen est alors de 0.665 %.

Afin d'effectuer une classification sur notre jeu de données, l'arbre de décision semble être une méthode plus judicieuse que le classificateur bayésien.

Cela est probablement lié au fait que ce dernier est basé sur un apprentissage probabiliste. Il calcule explicitement les probabilités des hypothèses. On considère une assomption naïve qui concerne l'indépendance des attributs. Cette hypothèse d'indépendance rend le calcul possible et, en cas de vérification, conduit à des classificateurs optimaux.

Cependant, cette hypothèse d'indépendance est rarement vérifiée. On pourrait émettre l'hypothèse de l'existence d'éventuelles corrélations entre les attributs de notre jeu de

données. Comment résoudre ce problème ? Les réseaux Bayésiens permettent de combiner le raisonnement Bayésien avec la relation causale entre les attributs.

Le nombre de validations croisées semble avoir un impact significatif sur le taux d'erreur (corrélation négative entre le nombre de validations croisées et le taux d'erreur). Cela est lié au fait que le jeu de données est divisé en un nombre plus important de partitions, augmentant ainsi la taille du jeu d'apprentissage; un nombre de validation croisées de 50 permet donc d'obtenir une plus grande précision mais augmente les temps de calcul.

L'élagage (MDL) a pour but d'augmenter la précision pour éviter que l'arbre généré ne reflète trop le jeu d'apprentissage en supprimant les branches pouvant représenter des anomalies. Néanmoins, on observe un taux d'erreur plus grand en présence d'élagage. Ceci peut sembler paradoxal à première vue, on pourrait émettre l'hypothèse que l'élagage supprime des branches nécessaires à la classification. L'élagage pourrait donc être intéressant pour un jeu de données de taille considérable mais n'est pas adapté pour le nôtre.

❖ BILAN ET PERSPECTIVES

La majorité du travail s'est concentrée sur la construction de la matrice individus-variables. Il a fallu préalablement sélectionner des attributs pertinents biologiquement afin de classer les gènes selon leur sous-type de protéine ABC.

Il nous a semblé logique de sélectionner les identifiants des domaines ("FD_ID") pour réussir à classer les gènes selon les sous-types de protéines pour lesquelles ils codent, supposant que les types de domaines fonctionnels présents pourraient discriminer les classes.

Après quelques recherches bibliographiques, il nous paraissait judicieux de sélectionner les tailles des domaines et les positions END et START : un même domaine pourrait être positionné à des endroits différents chez des protéines de deux sous-types différents.

Enfin, il était nécessaire de sélectionner les noms des sous-types protéiques ("Family_link"), attribut qui permet l'apprentissage et également la classification recherchée.

L'ajout du score s'est fait tardivement dans le processus de construction de la matrice et a permis d'améliorer les résultats de prédiction.

La présence de doublons dans les tables et de valeurs manquantes a rendu la construction des premières matrices ardues, une maîtrise et une compréhension forte des données permet alors un exercice plus aisé et plus efficace. L'utilisation de la fonction *merge* a présenté certaines difficultés dans la prise en main des choix des arguments optionnels et l'utilisation de table très différentes (en taille, en doublons, en attributs…).

Les résultats issus de la classification utilisant l'arbre de décision avec une méthode Stratified sampling ou Random sampling sont excellents avec un pourcentage d'erreurs moyen inférieurs à 1%. Ces résultats sont significatifs et permettent de conclure quant à la pertinence des attributs choisis. Ainsi, les gènes codent pour des protéines ABC et peuvent être classifiés selon leurs domaines fonctionnels, la taille et la position de la séquence.

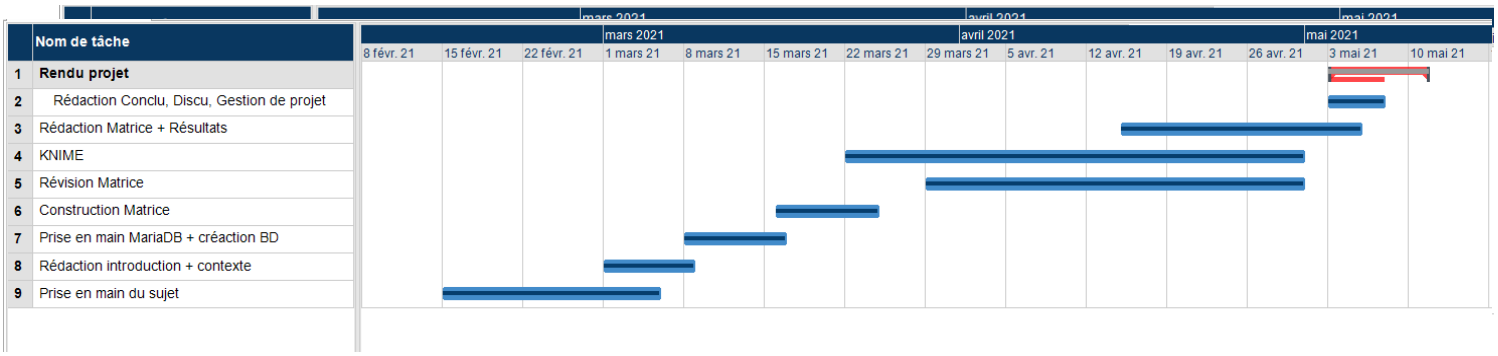
Il aurait cependant été intéressant d'utiliser d'autres méthodes de classification en vue de comparer leurs performances avec les méthodes abordées dans ce projet.

Éventuellement, nous aurions pu tester la validation croisée où k=s (Leave-one-out). Néanmoins, cette dernière est nettement plus coûteuse en calcul que la K validation croisée sans être nécessairement meilleure. De plus, il existe une sous-estimation de l'erreur en cas de sur-apprentissage.

Il pourrait également être intéressant de tester les mêmes arbres avec un plus grand nombre de cross validation, 100 par exemple, et d'observer une éventuelle amélioration des résultats.

❑ GESTION DE PROJET

La gestion de ce projet fut déterminée début février selon le diagramme de GANTT suivant :



L'appréhension du sujet et la prise en main des tables fut plus longue que prévue, les données fournies étant très nombreuses et le nombre d'individus très variés entre les tables, Hugo prit donc de l'avance sur la rédaction de l'introduction et du contexte.
Le choix du sujet s'est fait rapidement après, mais le choix des attributs et des tables étaient plus arbitraires dans un premier temps, il était évident que la matrice serait révisée au fur et à mesure des essais sur KNIME.

Nous avons choisis de travailler nos matrices à l'aide de requêtes SQL avec MariaDB, seulement, l'accessibilité à notre base de données et au logiciel nous a été possible uniquement depuis la P0 sur le campus. La semaine de vacances fin février nous a permis de rattraper notre retard, mais également de prendre de l'avance sur la construction de la première matrice. Les requêtes SQL étaient simples à réaliser, notre formation étant solide et récente dans ce domaine. La difficulté principale que nous avons rencontré concerne les différentes manières de merger nos matrices (présence de doublons).

Les retours sur la matrice nous ont pris la majorité du temps dédié à ce projet : afin de gagner en efficacité, un premier binôme révisait les matrices pendant que l'autre la lançait sur KNIME et analysait les résultats. Malgré une organisation se voulant la plus efficace possible, les retours sur la matrice jusqu'à donner des résultats acceptables nous ont pris plus de 70% du temps accordé au projet.

La rédaction du projet s'est maintenue tout au long du travail ce qui nous a permis de gagner du temps sur la fin du projet.