

## RAPPORT DE PROJET

---

# TER 2021 - 073

## Just drag and drop

---

*Réalisé par*

**Nicolas DEMOLIN**

**Ralph EL CHALFOUN**

**Jérémy HIRTH DAUMAS**

**Christel RALALASOA**

*Encadré par*

Peter Sander



### **Remerciements**

Nous tenons à remercier notre encadrant Monsieur Peter SANDER pour son aide tout au long de notre projet, Monsieur Lionel FILLATRE pour ses conseils en science de données et Monsieur Cyril TONIN pour son aide pour le matériel dont nous avons besoin.

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Identification du sujet . . . . .	3
1.2	Présentation du sujet . . . . .	3
1.2.1	Contexte . . . . .	3
1.2.2	Problématique . . . . .	3
1.2.3	L'espace des solutions possible . . . . .	4
1.2.4	Notre projet . . . . .	4
<b>2</b>	<b>Travail réalisé</b>	<b>4</b>
2.1	Data pipeline . . . . .	4
2.1.1	Data visualisation . . . . .	4
2.1.2	Data modèle . . . . .	5
2.1.3	Sélection automatique . . . . .	8
2.2	Web application . . . . .	13
2.2.1	Back-end . . . . .	13
2.2.2	Front-end . . . . .	14
2.3	Sécurité . . . . .	17
2.3.1	Authentification . . . . .	17
2.3.2	Chiffrement . . . . .	19
2.3.3	Sécurité supplémentaire . . . . .	20
2.4	Déploiement, tests et difficultés . . . . .	21
2.4.1	Déploiement . . . . .	21
2.4.2	Difficultés rencontrées . . . . .	21
2.4.3	Tests . . . . .	22
<b>3</b>	<b>Positionnement de la solution par rapport à l'existant</b>	<b>23</b>
<b>4</b>	<b>Conclusion</b>	<b>24</b>
<b>5</b>	<b>Sources</b>	<b>25</b>
<b>6</b>	<b>Annexe</b>	<b>27</b>

# 1 Introduction

## 1.1 Identification du sujet

Notre groupe est composé de 4 membres : Ralph EL CHALFOUN (M2-WIA), Jérémy HIRTH DAUMAS (M2-CASPAR), Nicolas DEMOLIN (MAM5-SD) et Christel RALALASOA (MAM5-SD).

Nous avons en tant qu'encadrant Monsieur Peter SANDER, professeur des Universités.

Afin de valider notre Master Informatique ou notre diplôme d'ingénieur, nous devons effectuer un TER (Travaux d'Etude et de Recherche) lors de notre dernière année en Master/Cycle Ingénieur.

Pour mettre en application toutes les compétences que nous avons acquises au cours de nos années d'études, nous avons choisi un projet de développement regroupant les spécialités de chaque étudiant.

Ce projet a été réalisé au premier semestre de l'année universitaire 2021-2022.

## 1.2 Présentation du sujet

### 1.2.1 Contexte

La Data est au centre du monde du travail. L'essor du Big Data pousse les entreprises à se pencher de plus en plus sur l'analyse de données afin de valoriser la valeur de leurs produits et ainsi augmenter leurs bénéfices.

De ce fait, chaque année, des postes sont ouverts dans chaque entreprise afin de recruter des personnes qui pourront extraire, analyser et apporter des résultats quant aux données collectées.

En effet, le Big Data a donné naissance à de nombreux nouveaux métiers : Data Miner, Data Analyst, Chief Data Officer, Data Architect, Data Scientist... Ces experts sont en mesure de collecter des données, de les organiser, de les traiter, et de les transformer en informations exploitables par tous les départements de l'entreprise. [1]

Encore aujourd'hui, la demande dépasse l'offre ce qui fait que certaines entreprises peuvent se retrouver sans effectif adéquat pour exploiter ses données. De surcroît, cela veut donc aussi dire que le nombre de personnes qualifiées pour ce genre de tâche est encore assez faible.

### 1.2.2 Problématique

Le but est donc de répondre à la problématique qui est de trouver comment permettre à des personnes ou à des entreprises de pouvoir analyser leurs données sans qu'elles aient besoin d'être des expertes dans le domaine. En d'autres termes, comment faciliter l'analyse de données tout en gardant son efficacité ?

### 1.2.3 L'espace des solutions possible

Il existe plusieurs solutions pour répondre à cette problématique. Il est possible de créer un logiciel à installer par l'utilisateur ou encore un site web accessible directement sur internet ou alors une application web qui serait le parfait compromis entre un logiciel et un site web. Nous avons donc décidé de nous pencher vers cette dernière solution qui est pour nous la plus optimale.

### 1.2.4 Notre projet

Notre démarche est donc de développer une application web, nommée Scanylab, qui permettrait à n'importe qui de pouvoir effectuer des analyses de données sur ses bases, en quelques clics et en quelques minutes.

Ainsi, aussi bien une entreprise qu'un étudiant pourrait utiliser cette application et exploiter ses données pour en tirer des informations utiles. Il est également possible d'utiliser l'application pour approfondir ses connaissances en science de données. En effet, pour comprendre l'influence des hyperparamètres d'un modèle de Machine Learning, notre application pourrait être intéressante.

Cette application couvrira donc une partie web, une partie sécurité et une partie Machine Learning qui sont les trois spécialités de nos membres du groupe. De façon plus détaillée, elle permettra à l'utilisateur de créer un compte et de se connecter afin de pouvoir faire différentes analyses sur des bases de données :

- Visualisation de données
- Régression
- Classification

L'utilisateur pourra ensuite enregistrer ses analyses et les télécharger pour les avoir en local. Il pourra également gérer les données sur son compte (supprimer des bases ou des analyses) en toute sécurité.

## 2 Travail réalisé

### 2.1 Data pipeline

#### 2.1.1 Data visualisation

La visualisation des données est une partie très importante dans le traitement des données. En effet, elle permet de mieux comprendre la composition de la base et ainsi, de pouvoir effectuer des analyses qui par extension peuvent amener à des modifications importantes. Elle est aussi importante en fin de processus pour présenter les résultats de manière claire et précise à une quelconque audience.

Ainsi, il est possible à l'utilisateur d'afficher des graphes ou bien des histogrammes des variables qu'il souhaite voir représentées. Il peut également afficher la matrice de corrélation ou être

redirigé vers une page HTML décrivant la base sur laquelle il travaille. Ces fonctionnalités peuvent ainsi l'aider à comprendre l'influence (ou l'importance) de chaque variable et ainsi obtenir des résultats quant à ses analyses.

**Graphes** L'utilisateur choisit au moins deux variables qu'il souhaite afficher de façon à obtenir  $x$  en fonction de  $y$ . Il peut également choisir une troisième variable non numérique dont sa distribution sera représentée en couleur sur les points du graphe. Dans les paramètres avancés, il peut aussi choisir le mode de son graphe (nuage de points ou avec une courbe). Cependant, cela est seulement possible s'il ne choisit pas de troisième variable. ([Annexes 1-2](#))

**Pie chart** Si l'utilisateur a choisi un troisième paramètre, deux pie charts s'affichent. L'un affiche la distribution générale de ce paramètre en couleur et numériquement. ([Annexe 2](#)) L'autre affiche la moyenne de  $x$  parmi  $y$  en fonction de la troisième variable.

**Histogramme** Deux histogrammes s'affichent. L'un représente la distribution de la première variable, l'autre celui de la deuxième, tous les deux en barres verticales. ([Annexe 1-2](#))

**Matrice de corrélation** Située dans les paramètres avancés, cette fonctionnalité peut également être affichée par l'utilisateur. Elle est en couleur mais elle donne aussi accès aux valeurs numériques afin d'avoir une visualisation plus claire de la corrélation entre chaque paire de variables. ([Annexe 3](#))

**Overview** En cliquant sur cette fonction, l'utilisateur a accès à une page HTML décrivant la base sur laquelle il travaille. C'est-à-dire qu'il peut y retrouver les informations suivantes pour chaque variable : le nombre de valeurs, le pourcentage de valeurs uniques, le nombre et le pourcentage de valeurs manquantes, le nombre et le pourcentage de valeurs infinies, la moyenne, le minimum, le maximum, le nombre et le pourcentage de zéros et la taille en mémoire qu'elle prend. Enfin, elle affiche également les dix premières et dernières lignes de la base de données. ([Annexe 4](#))

Toutes ces fonctionnalités, constituant un dashboard, sont basiques mais nécessaires à une analyse de données c'est donc pour cela que nous les avons sélectionnées pour les intégrer dans notre application web.

Afin de pouvoir réaliser cela, nous avons écrit un script en Python qui, en premier lieu, lit le fichier grâce à la fonction `read` de la librairie `pandas` puis utilise les fonctions déjà implémentées de cette même librairie pour récupérer la matrice de corrélation (fonction `corr`) et l'overview (fonction `ProfileReport`). La sortie sur la console est ensuite traitée par JavaScript qui s'occupera de l'affichage côté client. Quant au graphe, à l'histogramme et les pie charts, ils ont été directement traités par la partie Web application (Section 2.2)

### 2.1.2 Data modèle

En addition d'une partie visualisation de données, nous permettons à l'utilisateur de pouvoir faire des prédictions sur sa base de données. En effet, suivant s'il souhaite classer ou effectuer une régression, il a à sa disposition différents modèles/algorithmes de prédiction avec différents paramètres dont il peut choisir les valeurs.

Pour effectuer une régression, nous avons mis à disposition trois algorithmes de machine learning et certains de leurs paramètres (les plus influents) :

**Gradient Boosting** C'est un type de boost d'apprentissage automatique. Il s'appuie fortement sur la prédiction selon laquelle le prochain modèle réduira les erreurs de prédiction lorsqu'il sera mélangé aux précédents. L'idée principale est d'établir des résultats cible pour ce prochain modèle afin de minimiser les erreurs. [2] Paramètres retenus :

- **learning\_rate** : détermine la taille du pas à chaque itération tout en se déplaçant vers un minimum d'une fonction de perte (loss function) [2]
- **n\_estimators** : le nombre d'étapes de boosting à effectuer [2]
- **max\_depth** : profondeur maximale des estimateurs de régression individuelles [2]
- **min\_samples\_split** : le nombre minimum d'échantillons requis pour diviser un nœud interne [2]

**Random Forest** C'est un méta estimateur qui ajuste un certain nombre d'arbres de décision de classification sur divers sous-échantillons de l'ensemble de données et utilise la moyenne pour améliorer la précision prédictive et contrôler le sur-ajustement/over-fitting (modèle qui correspond trop précisément à une collection particulière d'un ensemble de données). [2] Paramètres retenus :

- **n\_estimators** : le nombre d'arbres [2]
- **max\_depth** : la profondeur maximale de l'arbre. [2]
- **min\_samples\_split** : Le nombre minimum d'échantillons requis pour diviser un nœud interne [2]

**Ridge** Ce modèle résout un modèle de régression où la fonction de perte est la fonction linéaire des moindres carrés et la régularisation est donnée par la norme l2. [2] Paramètres retenus :

- **tol** : précision de la solution [2]
- **solver** : solveur à utiliser dans les routines de calcul [2]
- **alpha** : la force de régularisation [2]

Pour effectuer une classification, nous avons mis à disposition cinq algorithmes de machine learning et certains de leurs paramètres (les plus influents) :

**LinearSVC** La méthode Linear Support Vector Classifier (SVC) applique une fonction de noyau linéaire pour effectuer la classification et fonctionne bien avec un grand nombre d'échantillons. [3] Paramètres retenus :



- **penalty** : spécifie la norme utilisée dans la pénalisation [2]
- **tol** : tolérance pour le critère d'arrêt [2]
- **C** : paramètre de régularisation [2]
- **class\_weight** : poids appliqués aux classes [2]

**AdaBoost** Un classificateur AdaBoost est un méta-estimateur qui commence par ajuster un classificateur sur l'ensemble de données d'origine, puis adapte des copies supplémentaires du classificateur sur le même ensemble de données, mais où les poids des instances mal classées sont ajustés de sorte que les classificateurs suivants se concentrent davantage sur les cas difficiles. [2] Paramètres retenus :

- **n\_estimators** : le nombre maximal d'estimateurs auquel l'amplification est terminée [2]
- **learning\_rate** : poids appliqué à chaque classifieur à chaque itération de boosting [2]

**Gradient Boosting** Description faite précédemment. Paramètres retenus :

- **learning\_rate** : le taux d'apprentissage de chaque arbre [2]
- **n\_estimators** : le nombre d'étapes de boosting à exécuter [2]
- **max\_depth** : profondeur maximale des estimateurs de régression individuelles [2]
- **min\_samples\_split** : le nombre minimum d'échantillons requis pour diviser un nœud interne [2]

**Random Forest** Description faite précédemment. Paramètres retenus :

- **n\_estimators** : le nombre d'arbres dans la forêt [2]
- **max\_depth** : la profondeur maximale de l'arbre [2]
- **min\_samples\_split** : le nombre minimum d'échantillons requis pour diviser un nœud interne [2]
- **class\_weight** : poids appliqués aux classes [2]

**Logistic Regression** Ce modèle modélise les probabilités de problèmes de classification avec deux résultats possibles. C'est une extension du modèle de régression linéaire pour les problèmes de classification. [2] Paramètres retenus :

- **penalty** : spécifie la norme utilisée dans la pénalisation [2]

- **tol** : tolérance pour le critère d'arrêt [2]
- **c** : inverse de la force de régularisation [2]
- **class\_weight** : poids appliqués aux classes [2]
- **max\_iter** : nombre maximal d'itérations nécessaires pour que les solveurs convergent [2]

Pour réaliser cela, nous avons récupéré la sortie sur la console des choix de l'utilisateur et nous les avons traités via un script Python. En effet, nous avons créé une fonction principale qui, tout d'abord, réalise des transformations (normalisation des features et un labelEncoder sur les variables non numériques) puis appelle l'algorithme choisi et applique aux paramètres les valeurs données. Elle retourne sur le terminal :

- S'il s'agit d'une prédiction :

- le R2 score (coefficient utilisé pour évaluer la performance d'un modèle de régression)
- la prédiction et les vraies valeurs (sous forme de dataframe)
- l'influence de chaque variable (sous forme de dataframe)

- S'il s'agit d'une classification :

- le score d'accuracy
- la matrice de confusion (sous forme de dataframe)
- l'influence de chaque variable (sous forme de dataframe)

Ensuite, les données sorties sur le terminal sont traitées par la partie Web application pour l'affichage côté client. (section 2.2)

### 2.1.3 Sélection automatique

Enfin, l'utilisateur peut également laisser l'application choisir le meilleur algorithme pour effectuer la prédiction ou la classification. En effet, nous avons implémenté un algorithme d'auto-Machine Learning (autoML). L'autoML est très populaire dans le monde de la Data Science. Cela consiste à créer une méthode qui va sélectionner le meilleur algorithme et les meilleurs hyperparamètres pour un problème de Machine Learning donné. Différentes méthodes bien connues existent comme :

- GridSearch : cette méthode va tout simplement essayer l'ensemble des possibilités que nous avons rentrées en paramètre. C'est une méthode très gourmande en calcul.
- RandomSearch : cette méthode est semblable au GridSearch mais avec la différence que

les tests d'hyperparamètres sont pris aléatoirement dans notre grille d'hyperparamètres.

- Optimisation bayésienne : cet optimiseur d'hyperparamètres est le plus fiable. En effet, avec cette technique nous sommes certains de récupérer les meilleurs hyperparamètres mais elle est encore une fois très gourmande en calcul.

Toutes ces méthodes sont implémentées sur Scikit-Learn et un algorithme d'autoML basé sur un optimiseur bayésien est aussi implémenté du nom de autoSKLEARN. Nous avons essayé d'implémenter autoSKLEARN dans notre application mais comme attendu, l'optimisateur bayésien est bien trop lent. Nous devons donc créer notre propre algorithme d'autoML et c'est ainsi que MetaScan est né. MetaScan est un algorithme que nous avons créé pour répondre à notre problème en utilisant une technologie très à la mode dans le monde du Machine Learning : **le méta-learning**.

Le méta-learning est une branche du Machine Learning qui consiste à « apprendre à apprendre » à l'aide des métadonnées d'une base. Prenons l'exemple d'un réseau de neurones qui veut résoudre un problème de régression complexe. Lors des premières itérations, le réseau ne converge pas forcément. Le méta-learning est là pour éviter cela.

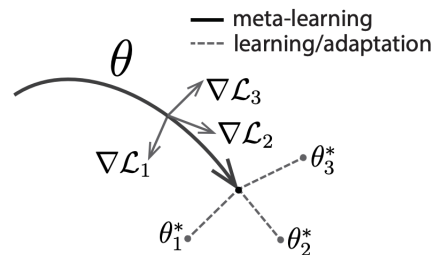


FIGURE 1 – Diagramme représentant l'algorithme de méta-learning par Chelsea Finn, Pieter Abbeel et Sergey Levine [4]

Comme nous pouvons le voir ci-dessus, l'algorithme de MAML va aider la convergence du gradient pour arriver vers le minimum global plus rapidement. Les algorithmes de méta-learning se basent sur la prévisualisation de la base (des métadonnées) pour pouvoir aider les réseaux de neurones à mieux s'adapter et donc à mieux apprendre. Cependant, dans Scanylab il est impossible d'utiliser des réseaux de neurones auto-adaptatifs comme ci-dessus. Les bases que Scanylab reçoit sont toutes extrêmement différentes et sont de petite taille ce qui rend la tâche du réseau trop compliquée. Nous avons donc décidé d'appliquer le méta-learning au Machine Learning d'une manière « originale ». Nous allons utiliser les métadonnées comme une carte d'identité de nos bases de données. Nous allons, grâce à la bibliothèque MFE de Python, calculer l'ensemble des caractéristiques de nos bases de données et cela va servir de base d'apprentissage pour MetaScan. Cependant, il ne faut pas oublier le but premier de MetaScan, l'autoML. Donc après avoir créé la carte d'identité de notre base, nous avons calculé la performance de différents algorithmes de machine learning sur cette même base et nous avons stocké ces scores (R2 score et accuracy) dans notre base d'apprentissage. Ainsi, dans cette dernière, nous avons l'ensemble des caractéristiques de notre base et le score de prédiction des différents algorithmes. Ensuite, il

suffisait d'appliquer un algorithme de Machine Learning afin de prédire le meilleur algorithme pour chaque base/chaque métadonnée. C'est ainsi que nous avons procédé.

Afin de créer notre base d'apprentissage, nous avons réalisé trois scripts majeurs en Python :

- randompick.py
- calculmetafeature.py
- machinelearningprocess.py

Ces trois scripts fonctionnent de pair et notre processus est assez simple : nous rentrons une base dans randompick.py qui va récupérer aléatoirement une variable à prédire d'une base sélectionnée et des features pour réaliser une prédiction. Une fois cela fait, nous demandons à calculmetafeature.py de calculer les méta-features de cet ensemble sélectionné. Il faut savoir qu'une même base de données peut fournir un très grand nombre de méta-features différentes et c'est là toute la puissance de cette méthode. En effet, nous n'avons pas besoin de 10 000 bases pour avoir assez de variabilité dans les méta-features. Lorsque nous prenons ces dernières avec les mêmes variables à prédire mais avec des features différentes, elles sont foncièrement différentes. Ainsi, nous avons créé un ensemble de données extrêmement varié avec seulement 50 bases de données récupérées sur la plateforme web Kaggle.

Après avoir calculé les méta-features de chaque couple feature-prédicat, le script machinelearningprocess.py entre en jeu. Il va réaliser la prédiction sur sept algorithmes : KNN Regressor, Gradient Boosting, Random Forest, Ridge, Lasso, Elasticnet et SGDRegressor, et stocker leur R2\_score qui est notre base d'apprentissage.

Nous abordons ainsi trois grandes familles du Machine Learning :

- Les arbres de décision multiple avec les algorithmes : Random Forest et Gradient Boosting qui sont séparés respectivement en deux familles : les algorithmes de type bagging et le boosting.
- Les algorithmes de type neighbors : KNN Regressor
- Les algorithmes linéaires : Ridge, Lasso, Elasticnet et SGDRegressor

Ces algorithmes ont été choisis par leur popularité mais aussi par leur différence de fonctionnement. Il est nécessaire d'avoir différents types d'algorithmes car un certain type peut être très performant sur certaines bases mais pas sur d'autres. L'algorithme universel n'existe pas, c'est pour cela que l'autoML est si important en science de données. L'histogramme ci-dessous illustre très bien ces propos :

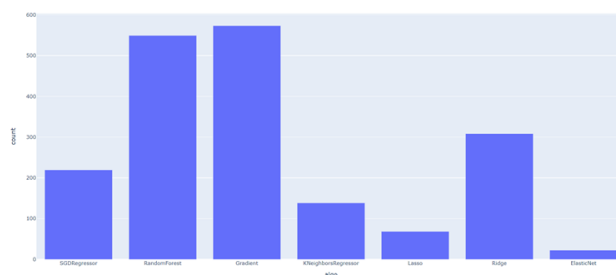


FIGURE 2 – Histogramme représentant le nombre de top 1 par algorithme

Ici, nous avons le nombre de top 1 par algorithme. Nous constatons que les algorithmes Random Forest et Gradient Boosting dominent mais chaque algorithme a sa force.

Ainsi, nous avons réalisé plus de 8000 apprentissages pour avoir assez d'échantillons au niveau des méta-features. Ensuite, énormément de possibilités se sont offertes à nous car cette base de données est très riche d'un point de vue autoML. Dans un premier temps, nous avons essayé de prédire le R2 score de chaque algorithme en fonction des méta-features et donc de déterminer le meilleur algorithme avec celui qui avait le meilleur score prédit. Cette régression n'était pas si facile et les résultats n'étaient pas satisfaisants. En effet, nous arrivions à un R2 score de 0.70 au maximum ce qui n'est pas assez précis.

Par la suite, nous avons décidé de prédire seulement si l'algorithme a eu le meilleur score ou non. Nous avons donc rajouté une colonne à notre base. Cette colonne contient des booléens, True si c'est le meilleur algorithme sur les sept, False sinon. Nous retombons donc sur un problème de classification classique. Ici, nous avons eu des résultats bien plus pertinents et très intéressants (avec une accuracy de 0.85) :

precision	
False	0.90
True	0.55

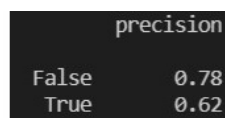
  

False, True
3688, 357
389, 440

FIGURE 3 – Résultats de la classification du meilleur algorithme

Notre base de données n'est pas du tout équilibrée c'est donc pour cela que nous obtenons de tels résultats sur la ligne False. Comme expliqué par la suite, le gros point noir de notre problème est les faux négatifs. De plus, nous constatons quelque chose d'inquiétant. En effet, parfois notre algorithme de classification ne prédit qu'aucun des algorithmes qu'il a eus ne va être le meilleur (Faux négatif). Ceci est problématique et c'est pour cela que nous avons intégré un second algorithme de classification qui va être plus « permissif » au niveau de l'attribution des True. Nous donnons donc en entrée de cet algorithme tous les cas les plus compliqués que notre premier algorithme n'arrive pas à résoudre et nous lui demandons de refaire la prédiction. Ce second algorithme est basé sur le principe de stacking de différents algorithmes de Machine

Learning :



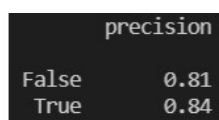
	precision
False	0.78
True	0.62

FIGURE 4 – Résultats de l'algorithme pour les cas compliqués

(avec une accuracy de 0.72). Même si les résultats ne semblent pas satisfaisants au premier abord, ils le sont en réalité grandement. En effet, il ne faut pas oublier que ce sont les algorithmes les plus difficiles à classifier et c'est pour cela que la précision n'est pas aussi bonne que précédemment.

Nous avons donc réussi notre objectif principal, la V1 de MetaScan est née. Nous sommes donc capables de déterminer l'algorithme de Machine Learning à utiliser en fonction des méta-features du problème. MétaScan utilise ensuite l'algorithme choisi pour le problème et réalise un RandomSearchCV (vu précédemment) dessus pour optimiser les hyperparamètres. Par la suite, nous pouvons nous poser une question : est-il possible que l'algorithme détermine si la prédiction va être un succès ou non ? La réponse est oui. Notre base d'apprentissage étant très riche, elle nous le permet. Mais quel serait l'intérêt ?

L'essentiel problème de l'application que nous avons réalisée est la surcharge des serveurs et il faut donc éviter les calculs inutiles. C'est pourquoi nous avons intégré une sécurité à MétaScan. Le principe est de ne pas faire l'optimisation des hyperparamètres si nous savons que la prédiction va être mauvaise à l'aide des méta-features. Si nous reprenons l'analogie avec la carte d'identité, notre algorithme de sécurité va jouer le rôle d'un videur de boîte de nuit ! Pour créer notre videur nous allons utiliser le même principe de boolean que précédemment. Nous ajoutons donc une nouvelle colonne à notre base d'apprentissage : GoodPrediction. True si  $R2\_score > 0.4$ , False sinon. Nous avons appliqué un algorithme de Machine Learning dessus et nous avons eu ces résultats (avec une accuracy de 0.82) :



	precision
False	0.81
True	0.84

FIGURE 5 – Résultats de l'algorithme avec prédiction du R2 score

Ils sont très satisfaisants, notre base de données est bien plus équilibrée que les précédentes. Notre algorithme d'autoML est donc capable de déterminer quel algorithme utiliser pour optimiser ses hyperparamètres mais aussi d'arrêter les calculs si la prédiction risque d'être mauvaise. MétaScan V2 est née.

La V3 serait de réaliser l'optimisation des hyperparamètres de chaque algorithme avec seulement les méta-features et non avec un RandomSearch. Cependant, cela demande de très gros calculs et représente un problème bien plus complexe qui nécessiterait des réseaux de neurones afin d'éviter les biais paramétriques. En effet, un majeur problème des algorithmes de Machine

Learning est qu'ils ne peuvent prédire seulement une colonne à la fois. Il n'est pas possible de demander à un algorithme Random Forest de prédire la valeur de plusieurs hyperparamètres. C'est pour cela que l'on parle de biais paramétrique car avec le Machine Learning nous sommes obligés de réaliser les prédictions une à une et donc le biais paramétrique apparaît. Un réseau de neurones n'a pas ce problème. Nous n'avons pas eu le temps d'effectuer ce travail.

Nous avons réalisé le même procédé pour la sélection automatique pour la classification.

Enfin, comme pour la partie data modèle, nous affichons les mêmes données pour la prédiction et la classification en sortie et en supplément, le meilleur algorithme avec les meilleurs paramètres.

## 2.2 Web application

### 2.2.1 Back-end

Le développement de l'application a donc consisté à implémenter un serveur et un client qui seront capables de proposer toutes les fonctionnalités citées précédemment. Pour cela nous avons d'abord créé un serveur NodeJS (version 17.2.0).

**Déclinaison JavaScript** TypeScript est un sur ensemble de JavaScript développé par Microsoft. Il utilise son propre compilateur pour convertir les fichiers TypeScript (.ts) en fichier JavaScript (.js). L'intérêt de l'utilisation de TypeScript est la vérification de type statique (String, Number, Boolean, Null, Array, Enum, Tuple et Generics). Cette fonctionnalité facilite la lecture du code et évite les bugs. [5] Il est la déclinaison qui apporte le plus d'intérêt et d'utilisation en 2020 (93%) [6], c'est pourquoi nous avons choisi de l'intégrer à notre projet de développement.

**Connexion à une base de données** Notre application web propose la création de compte et pour gérer plusieurs informations (mots de passe, identifiants, activités) sur les clients, nous avons besoin d'une base de données. Pour des raisons de simplicité, nous avons utilisé MongoDB qui propose une version gratuite, qui dans le cadre de notre TER est amplement suffisante. MongoDB est une base de données orientée document. [7] Pour son implémentation nous utilisons le package mongoose. Sur MongoDB, nous stockons les identifiants et mots de passe des clients, ainsi que diverses informations comme le nombre d'analyses réalisées, la taille de stockage disponible et utilisée pour chaque utilisateur, etc. Nous ne stockons cependant pas les bases des clients sur ce système car nous ne requêtons pas directement dessus et leurs tailles nous forceraient à prendre une version payante de MongoDB.

Les bases clientes sont donc stockées localement sur la machine hébergeant le serveur de l'application.

**Tests unitaires** Afin de tester notre code au niveau du backend, nous avons choisi Jest comme outil de compilation des tests. Jest est le framework pour les tests unitaires le plus utilisé en 2020 (68%) devant Mocha, Jasmine et d'autres. [8]

**Communication entre JavaScript et les scripts Python** Toute la partie Machine Learning est implémentée en Python. Pour appeler ces scripts depuis notre serveur, nous utilisons

la commande `exec` du package `child_process`. `Exec` permet d'exécuter une commande comme dans un terminal.

Prenons comme exemple un appel pour obtenir une prédiction :

```
exec('python script.py pathToTheDatabase csv features varToPredict isADemo aesCipherKey aesCipherToEncrypt')
```

Dans cet exemple (hormis Python) tous les paramètres sont dépendants de la requête et de l'utilisateur. Python récupère les paramètres en argument et effectue l'analyse correspondante. Pour retourner le résultat de l'analyse, Python « `print` » les données et nous récupérerons la sortie directement sur le serveur. Cette analyse est ensuite soit enregistrée soit envoyée au client.

**ROUTAGE SERVEUR** En ce qui concerne le routage du serveur, nous avons fait le choix de le structurer de la façon suivante : un script "app.ts" crée un serveur express (du package ExpressJS) et écoute sur différentes routes. En fonction de la requête faite par le client, la demande est redirigée successivement jusqu'à la méthode correspondante qui va traiter la requête.

Par exemple une requête de suppression de base de données se fait à la route suivante :

```
'http://localhost:4000/api/upload/deleteData'
```

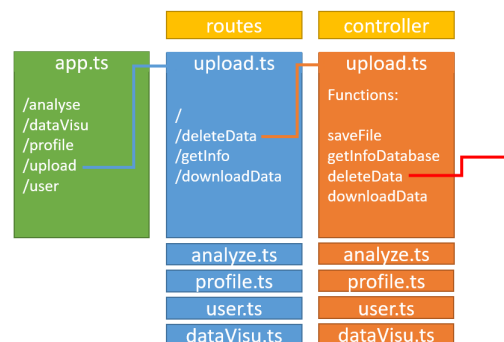


FIGURE 6 – Schema architecture des routes

### 2.2.2 Front-end

**Framework** Le client est la partie frontend de notre application. Pour mener à bien le frontend, nous avons choisi d'utiliser un framework. Ce dernier permet de créer un cadre de travail fournissant un ensemble d'outils et de bibliothèques ayant pour but d'améliorer le développement de l'application. Les frameworks web les plus connus sont ReactJS, Angular et VueJS. VueJS est, parmi les 3 cités, le plus simple d'utilisation car il utilise directement HTML et CSS. Cependant, il offre moins de possibilité que Angular ou ReactJS. Nous avons donc dû choisir entre ReactJS et Angular. Les deux sont assez équivalents en termes d'avantages et inconvénients. C'est donc sous les conseils de notre encadrant et après une analyse de l'utilisation des



frameworks sur les dernières années, que nous avons choisis ReactJS, étant le plus populaire. [9]

Comme pour notre serveur et pour majoritairement les mêmes raisons, nous avons également choisi d'utiliser TypeScript comme langage de programmation pour le client. Les modèles des composants de ReactJS sont en JSX, pour React+TypeScript ils seront donc en TSX.

**Apparence** Pour ce qui est de l'apparence de l'application, nous avons remplacé le CSS (Cascading Style Sheets) par Sass (Syntactically awesome stylesheets). Sass est une extension de CSS qui ajoute une super-puissance qui le rend plus utile, plus productif et qui écrit du code stable. Il est également appelé préprocesseur CSS. Sass permet de partager les propriétés CSS dans tout le code. [11] Sass fournit deux syntaxes distinctes SASS et SCSS, les deux sont similaires et font la même chose, mais ont un style d'écriture différent. Le SCSS est le dernier et considéré comme meilleur que SASS. [12] Nous avons donc utilisé SCSS. À noter que nous avons ajouté un thème jour-nuit à l'application ([Annexe 5](#)), cette fonctionnalité nécessite l'utilisation de variable CSS et non SCSS.

**Affichage des résultats** Le coeur de notre application repose sur l'analyse de base de données, les résultats de ces analyses sont composés en majeure partie de graphiques, nuages de points et matrices. Ainsi, afin d'afficher ces données, nous avons choisi d'utiliser ChartJS. Ce module nous permet d'afficher nos données sous de nombreuses formes. ([Annexe 2](#)) Ces affichages sont en réalité des éléments Canvas de HTML, sur lesquels il est possible de passer le curseur sur les points d'un graphique ou les cellules d'une matrice pour obtenir des informations supplémentaires. ([Annexe 3](#))

**Page d'accueil** Lorsqu'un utilisateur se rend sur l'application, il arrive sur la page d'accueil. Sur celle-ci, nous avons choisi de présenter l'application et notre équipe. Nous avons également crédité tous les outils qui ont permis la réalisation de ce projet. ([Annexe 5](#))

**Barre de navigation** La barre de navigation est un composant, qui grâce au framework React, nous permet de la réutiliser facilement sur les différentes pages de l'application. La navigation change en fonction de l'état de connexion de l'utilisateur :

- Un utilisateur déconnecté a accès à la démonstration "Démo" et aux pages de connexion "Log in" et inscription "Sign in". ([Annexe 6](#))
- Un utilisateur connecté a accès aux pages de gestion de bases "My databases" et "Upload", aux pages d'analyses "New analysis" et "Data visualisation" ainsi qu'à son historique d'analyses "History".

**Interface d'upload de bases** La page "Upload" permet aux utilisateurs connectés d'upload leurs bases de données par glissé/déposé ("drag and drop") ou en important directement le fichier depuis leur ordinateur. Les extensions acceptées sont : .csv, .xlsx, .xls, .json et .txt. À noter que dans le cas d'un fichier CSV, l'utilisateur peut choisir le type de séparateur (par défaut la virgule ","). ([Annexe 10](#)) Depuis la page "My database" les utilisateurs peuvent retrouver les bases déjà uploadées sur l'application ainsi que des informations sur ces bases, les télécharger ou les supprimer. ([Annexe 11](#))

**Réaliser une prédiction (classification/régression)** La page "New Analysis" est consacrée aux régressions et aux classifications. En sélectionnant une base (déjà uploadée), l'utilisateur a le choix parmi toutes les variables de sa base. Une fois la variable choisie, il doit choisir une ou plusieurs feature(s) qui permettront de prédire la variable. Enfin, des algorithmes de régression ou de classification lui sont proposés en fonction du type de la variable (numérique/objet), accompagnés des paramètres modifiables. Que ce soit pour une régression ou une classification, l'utilisateur a la possibilité de choisir l'algorithme de Machine Learning automatique que nous avons implémenté, ce dernier n'a pas besoin de configuration. ([Annexe 7](#)) Également, auprès de chaque algorithme et paramètre, se trouve une indication pouvant aider l'utilisateur à mieux comprendre l'influence de son choix sur l'analyse. Une fois l'analyse demandée et terminée côté serveur, la réponse est envoyée au client et affichée dans un nouvel onglet (sous forme de graphique ou matrice), ([Annexe 12](#)) permettant à l'utilisateur de garder les paramètres entrés et d'affiner son analyse en modifiant que certains facteurs à la fois. Enfin, dans le nouvel onglet il a la possibilité de télécharger le résultat sous forme d'image, et de l'enregistrer dans son historique afin de revisualiser le résultat sur le site plus tard.

**Réaliser une data visualisation** Depuis la page "Data visualisation" il est possible de visualiser deux à trois colonnes d'une base. Nous avons choisi d'afficher ces données sous forme de dashboard. Pour que cela soit plus dynamique et facile à prendre en main, le dashboard est affiché sur cette même page, ce qui permet d'en afficher plusieurs à la suite et également de supprimer un à un les dashboards créés. Également comme pour l'analyse, il est possible d'enregistrer le dashboard au format image. Contrairement aux autres données affichées sur l'application, ici les dashboards sont majoritairement calculés par le client web et non par Python. Le serveur n'a que pour rôle de retourner les colonnes et c'est ensuite sur le client que les différents éléments qui composent le dashboard sont créés. ([Annexe 1-2-3](#))

**Interface d'historique** Les utilisateurs ont à leur disposition une page d'historique où ils peuvent retrouver les analyses qu'ils ont choisies de sauvegarder. Cette page se présente sous la forme de liste de cellules avec une prévisualisation de l'analyse (jpg enregistré et sauvegardé sur le serveur) et le nom de l'analyse. Une fois une cellule cliquée, plus d'informations apparaissent à droite de la page (algorithme, paramètres,...), leur permettant soit de supprimer l'analyse soit de la réafficher (format Chartjs). ([Annexe 15](#))

**Système de démonstration** Comme notre système d'analyse nécessite une création de compte, nous souhaitions donner la possibilité aux visiteurs du site de tester les fonctionnalités sans pour autant passer par le processus d'inscription. ([Annexe 6](#)) La page de démonstration permet donc de choisir entre une data visualisation ou une prédiction, dans les deux cas, l'utilisateur est redirigé vers la page correspondante en mode démo, ce qui signifie qu'il pourra tester les fonctionnalités sur une base pré-enregistrée commune à tous les utilisateurs.

**Système d'authentification** Des pages de connexion et de création de compte sont accessibles depuis la barre de navigation et seront expliquées plus en détails dans la partie suivante : Sécurité. ([Annexe 8-9](#))

## 2.3 Sécurité

### 2.3.1 Authentification

Notre application repose sur un système d'authentification afin que les utilisateurs puissent enregistrer leurs données et ainsi les conserver. Pour commencer, les utilisateurs doivent s'inscrire sur notre application en fournissant des informations, en particulier leur adresse email et leur mot de passe. L'application vérifie ensuite le bon format de l'adresse email, s'il/elle n'est pas déjà dans la base de données Mongo puis la complexité du mot de passe choisi par l'utilisateur. Bien évidemment, ces vérifications sont effectuées côté serveur pour empêcher qu'une personne change la requête d'inscription et ainsi contourne les vérifications. Une fois les vérifications passées avec succès, les informations de l'utilisateur vont être enregistrées dans la base de données dont son mot de passe qui sera haché avec l'algorithme « Argon2id », comme indiqué par les recommandations de l'OWASP [13].

Une fois que l'utilisateur s'est enregistré sur notre application, il peut désormais s'authentifier afin d'ouvrir une session. Mais si le système d'authentification est mal implémenté et présente des failles, une personne mal intentionnée pourrait voler les identifiants d'un autre utilisateur ou alors tout simplement se connecter à un compte quelconque. Un système d'authentification peut présenter entre autres les failles suivantes :

- Les failles XSS (Cross-site scripting) qui permettent d'injecter du contenu dans une page Web, provoquant ainsi des actions sur les navigateurs web visitant la page. Un exemple concret serait le vol de cookies contenant un token d'authentification si ceux-ci ne sont pas HttpOnly, donc accessible via JavaScript.
- Les failles CSRF (Cross-Site Request Forgery) qui consiste à faire exécuter une requête HTTP falsifiée à un utilisateur. Par exemple une requête pour envoyer les cookies contenant un token d'authentification vers le serveur de l'attaquant, et ce, même si les cookies sont HttpOnly.
- Les injections NoSQL qui permettent aux attaquants de placer des commandes arbitraires dans une requête NoSQL afin de récupérer, modifier ou supprimer des données.

Pour gérer notre système d'authentification, nous avons décidé d'utiliser les JSON Web Token (JWT), et il devra donc se prémunir contre ces failles. Pour ce qui est des injections NoSQL, nous effectuons une sanitization de l'entrée de l'utilisateur et nous utilisons les plugins « Mongoose » qui aident à prévenir contre ce type d'attaque. Quant aux failles XSS et CSRF, la partie ci-dessous va nous permettre d'en savoir plus sur comment nous arrivons à les contrer.

Lorsqu'un utilisateur va pour s'authentifier, les étapes ci-dessous se produisent :

- L'utilisateur effectue une demande de connexion, ses identifiants sont alors envoyés au serveur sur la route « /api/auth/login »
- Le serveur compare les identifiants reçus avec ceux dans la base de données. Si les identifiants sont corrects, alors :

1. Un token CSRF est généré.
  2. Un JWT est généré contenant le token CSRF dans son payload.
  3. Un refresh Token est généré ainsi que les dates d'expiration de celui-ci et du JWT.
- Le JWT et le refresh token sont transmis via les cookies, le token CSRF et les dates d'expiration via le corps de la réponse.
  - Le client reçoit donc la réponse de la requête et enregistre les informations comme ceci :
    1. Le token CSRF est enregistré dans le localStorage pour éviter les attaques de type CSRF.
    2. Les cookies contenant le JWT et le refresh token sont enregistrés. Comme les cookies ont l'attribut HttpOnly, cela a pour conséquence d'empêcher les attaques de type XSS avec le vol de cookies.
  - Lors de la prochaine requête de l'utilisateur, les cookies seront inclus dans celle-ci et le token CSRF sera inclus dans le header de la requête.
  - Le serveur vérifie la validité du JWT et la présence du token CSRF dans son payload. Si cela est validé alors l'utilisateur s'est correctement authentifié et le serveur lui délivre le résultat de sa requête.

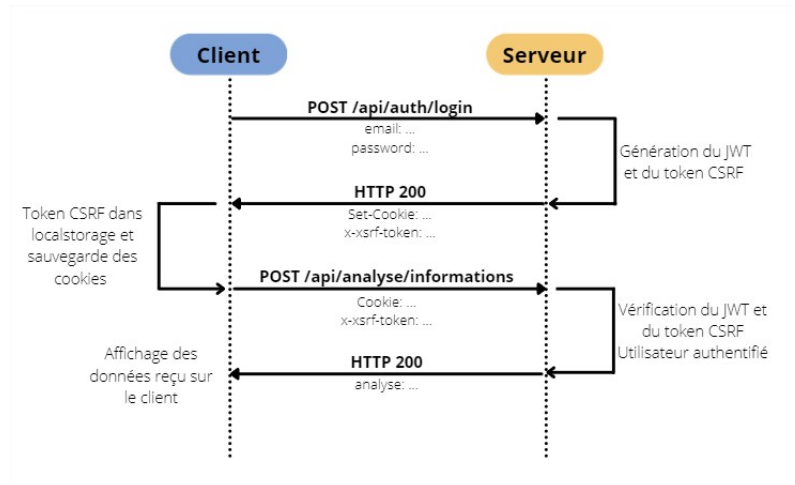


FIGURE 7 – Schéma d'authentification via JWT et token CSRF

Pour résumer le système d'authentification, les informations qui permettent d'authentifier un utilisateur sont divisées en deux parties, une se trouve dans le `localStorage` et l'autre dans les cookies. Car en effet les cookies `HttpOnly` ne sont pas vulnérables aux attaques XSS, et inversement le `localStorage` n'est pas vulnérable aux attaques de type CSRF.

### 2.3.2 Chiffrement

Pour qu'un utilisateur puisse utiliser notre application à des fins d'analyse, il doit obligatoirement téléverser ses bases de données vers nos serveurs. Ceci n'est pas un problème lorsqu'il s'agit de données dont la confidentialité n'est pas requise, comme par exemple des données publiques. Mais lorsqu'il s'agit de données plus personnelles que l'on veut garder secrète, les utilisateurs peuvent se montrer (et se montreront) perplexes quant à la bonne sécurisation de leurs données, et ce à juste titre. Pour pallier ce problème, nous avons implémenté un chiffrement sur toutes les données de l'utilisateur, en particulier sur les bases de données que l'utilisateur téléverse mais aussi sur les analyses qu'il effectue et qu'il enregistre.

Nous avons choisi d'utiliser AES-256-CBC qui est fourni avec le module « crypto » de Node pour effectuer le chiffrement. Nous utilisons la fonction `crypto.scryptSync()`, qui est une fonction de dérivation de clé basée sur un mot de passe pour générer la clé de chiffrement. Comme mot de passe (à passer en argument à la fonction) nous utilisons l'identifiant unique de l'utilisateur tel qu'il est enregistré dans la base de données, ainsi qu'une clé secrète pour le sel (salt) qui est passé au programme Node à l'aide d'une variable d'environnement. Cela a pour conséquence d'avoir une clé de chiffrement unique à chaque utilisateur.

Nous procédons ensuite comme ceci pour effectuer le chiffrement et le déchiffrement :

- Lorsque l'utilisateur téléverse sa base de données sur le serveur, celle-ci est téléchargée par le module « multer » de Node et est directement chiffrée avant même d'être écrite sur le disque. Si la base de données est au format .xlsx, celle-ci sera convertie au préalable en csv et ensuite chiffrée.
- Lors d'une analyse, le programme Node transmet le chemin de la base de données chiffrée ainsi que la clé de déchiffrement au script Python, qui va quant à lui effectuer le déchiffrement de la base avant de travailler dessus. Une fois l'analyse terminée, le programme Node récupère les résultats du script et les envoie au client qui peut décider de les sauvegarder. Si c'est le cas, ils seront alors chiffrés et enregistrés sur le serveur.
- Et enfin, lorsque l'utilisateur veut consulter l'historique de ses analyses ou bien alors télécharger une de ses bases de données sauvegardée, le serveur Node effectue le déchiffrement et envoie les résultats.

L'important est de retenir que quelle que soit l'action de l'utilisateur, aucune de ses données n'est écrite en clair sur le serveur, même temporairement.

Bien que les données soient chiffrées côté serveur, les échanges entre celui-ci et le client doivent eux aussi être chiffrés pour garantir la confidentialité des données. Pour cela, nous avons développé la prise en charge de HTTPS avec un certificat SSL pour notre application. Nous utilisons actuellement un certificat généré par OpenSSL, donc non vérifié par une autorité car notre application n'est pas encore hébergée. Par ailleurs, nous envisageons de prendre un nom de domaine (Scanylab) et d'héberger notre application. Nous pourrions alors nous procurer un certificat SSL vérifié auprès d'une autorité de certification. De plus, l'application met en œuvre une redirection automatique HTTP vers HTTPS, elle n'est donc pas accessible depuis HTTP.

### 2.3.3 Sécurité supplémentaire

Au cours du développement, nous avons également implémenté plusieurs sécurités supplémentaires afin de garantir un niveau de sécurité accrue pour notre application. Nous avons implémenté notamment :

- **Une vérification de l'email de l'utilisateur :** Lorsqu'un utilisateur s'inscrit sur notre application, il est enregistré dans notre base de données où il est lié à un token unique généré par le serveur. Un mail de confirmation lui est alors envoyé. Ce mail comprend un lien avec le token redirigeant vers une page de confirmation de notre site. Le token présent dans le lien sera comparé avec celui enregistré dans la base de données, et s'ils sont identiques, l'utilisateur disposera du statut « confirmé » dans celle-ci. Tant qu'un utilisateur n'a pas confirmé son adresse email, il apparaîtra comme « non confirmé » dans la base de données et ne pourra donc pas accéder à la majeure partie des fonctionnalités de l'application. En particulier, il ne pourra pas téléverser des bases, effectuer des analyses ou visualiser ses bases de données. Il aura uniquement accès à son profil contenant ses informations. Nous avons utilisé le module « nodemailer » pour effectuer l'envoi du mail. La confirmation utilise actuellement une adresse Gmail mais nous envisageons de prendre un nom de domaine ce qui permettra à l'application d'avoir sa propre adresse mail.
- **Une limite de téléversement par utilisateur :** Chaque utilisateur dispose d'une limite d'utilisation de l'application. Ils sont limités à 20 analyses chacun et de 10 mégabytes pour le téléversement de base de données. Cependant, ils peuvent une fois la limite atteinte, supprimer des anciennes analyses ou base de données afin de voir leur taux d'utilisation baisser.
- **Une limitation de débit :** Une limitation de débit a été mise en place afin d'empêcher les attaques par force brute liées à l'authentification. Le module « express-rate-limit » nous a permis d'effectuer ceci en autorisant uniquement 20 requêtes de connexion toutes les heures.
- **L'implémentation d'un captcha :** Lors de l'inscription d'un utilisateur, celui-ci doit obligatoirement remplir le bon captcha pour que son inscription soit validée. ([Annexes 8](#))  
Le captcha fonctionne de la manière suivante :
  1. Lorsque l'utilisateur arrive sur la page d'inscription, le serveur génère un captcha aléatoire et le transmet au client. Le serveur enregistre la solution du captcha liée au bon client grâce à l'identifiant de session inclus dans un cookie.
  2. Lorsque l'utilisateur envoie la requête d'inscription, le serveur compare la proposition de l'utilisateur avec la solution qu'il a au préalable enregistrée. Si le captcha est validé, l'inscription se poursuit côté serveur.
- **Une vérification de toutes les entrées de l'utilisateur :** Nous avons utilisé le middleware « helmet » pour filtrer toutes les entrées de l'utilisateur. Ceci dans le but d'éviter de potentielles attaques XSS ou des injections NoSQL au moment de la connexion.

Malgré toutes les sécurités qui ont été implémentées, nous sommes conscients que le WEB

fait preuve d'une constante évolution et que les sécurités d'aujourd'hui ne seront sans doute pas celles de demain. Cela nous forcera à constamment nous remettre à jour pour assurer la sécurité de l'application.

## 2.4 Déploiement, tests et difficultés

### 2.4.1 Déploiement

Une fois la majeure partie du projet implémentée et fonctionnelle, nous devons essayer de déployer notre application sur un serveur. Cette manipulation nous permet de tester la viabilité du projet. En effet, jusqu'alors nous testions en local. Tous les transferts entre le client et le serveur étaient donc instantanés.

Nous avons donc demandé à M.Sander s'il était possible d'obtenir une machine qui hébergera le serveur. Notre encadrant nous a par la suite mis en relation avec Cyril Tonin, qui nous a gracieusement préparé et prêté une machine sous Linux accessible via le VPN de Polytech.

À titre d'information, la machine possède un puissant processeur (Intel® Xeon® Gold 5218). Les calculs de Machine Learning nécessitant beaucoup de ressources CPU, la puissance de ce processeur aide considérablement à réduire le temps de calcul.

Les étapes du déploiement sur la machine :

- Connexion via SSH
- Clonage du dépôt GitHub
- Installation de NodeJS (v17.2.0) et Python (3.8).
- Installation des packages Python grâce au "requierement.txt"
- Installation des packages sur le Back-end et Front-end grâce à "npm install"
- Build du client via "npm run build"
- Démarrage du serveur via "npm run dev"
- Accès à l'application depuis `https://adresse_ip_de_la_machine/`

### 2.4.2 Difficultés rencontrées

L'apprentissage de nouveaux outils en autonomie (React, TypeScript) fut laborieux car bien que la documentation sur internet soit bien fournie, l'utilisation commune de ces mêmes outils l'est moins. Nous avons eu du mal à trouver des informations sur l'utilisation simultanée de React, TypeScript et Jest. Cela a engendré un temps d'adaptation et de recherche important. L'utilisation de React dans notre projet n'est pas optimale (possibilité d'implémenter de nouveaux composants plus cohérents).

Le déploiement de l'application sur une machine Linux a posé des problèmes dûs à une incompatibilité de version d'un package Python (problème non présent sous Windows et MacOS).

La création d'une base d'apprentissage conséquente pour le méta-learning fut très longue car les méta-features sont parfois capricieuses et difficiles à calculer.

Comme l'application est hébergée sur une machine accessible uniquement via le VPN de Polytech, il nous est impossible d'acquérir un nom de domaine pour le rediriger vers celle-ci, ainsi la vérification du certificat SSL nécessaire pour la validation HTTPS est suspendue tant que notre projet est hébergé sur cette machine.

Le build du code TypeScript en JavaScript a posé un problème au niveau du Front-end, certaines fonctionnalités de notre application avaient une politique de sécurité de contenu trop élevée et empêchaient notre application de fonctionner. Avec quelques recherches nous avons trouvé une solution qui marche dans notre cas et qui consiste à ajouter `INLINE_RUNTIME_CHUNK = false` à l'exécution du client [14]. Cette solution n'est peut-être pas optimale, une alternative serait également un certificat SSL vérifié.

Notre fonctionnalité de vérification de l'adresse email lors de la création d'un compte a dû être temporairement désactivée également en raison du manque de nom de domaine. L'adresse Gmail jusqu'ici utilisée a été restreinte par Google pour cause d'envois depuis différentes adresses IP, et les mails de confirmation prenaient parfois plusieurs heures avant d'être réceptionnés.

### 2.4.3 Tests

**Test Back-end Jest** Grâce à l'outil Jest sur le serveur TypeScript, nous avons pu réaliser des tests pour nos fonctions principales. La plupart de nos tests consistent à simuler des requêtes du client avec selon la route des données dans le corps de la requête. Une fois la demande simulée, nous observons la réponse du serveur. Si le résultat est celui que nous attendions, alors nous considérons que la fonction testée est correctement implémentée pour la requête faite. En revanche, si la réponse n'est pas correcte, nous corrigeons la méthode jusqu'à ce que les tests passent. Cette méthode de travail s'appelle l'automatisation de tests et permet d'adapter le développement aux tests et non l'inverse. En plus de ces tests, nous nous sommes servi de Postman qui permet d'exécuter rapidement et simplement des requêtes et d'en observer la réponse sous plusieurs formats. Nos tests serveur ne couvrent pas l'intégralité des fonctions implémentées mais les plus importantes, notamment les méthodes d'authentification pour assurer une sécurité valide ainsi que celles intervenant dans le processus d'analyse car ce sont des méthodes exécutant des scripts Python demandant beaucoup de paramètres à vérifier.

**Tests Front-End** Pour les tests au niveau du client, nous observions principalement dans le navigateur si les données affichées étaient bien celles attendues. Nous nous aidions également de `console.log()` pour afficher dans la console du navigateur la valeur de certaines variables lors du développement. Postman a également servi à développer le client car il nous a permis de trouver la bonne syntaxe pour les requêtes.

**Tests Python** Pour nos scripts Python, nous les avons d'abord testés en local, sur notre propre éditeur, afin de vérifier que chacune de nos fonctions fonctionnait. En effet, nous avons



simulé les appels que ferait le serveur Node pour récupérer les paramètres en les appelant directement dans nos fonctions pour les tester. Puis, une fois rattachées à l'application, nous avons fait de nouveaux tests pour vérifier que la connexion entre le code TypeScript et celui de Python fonctionnait. Nous avons constaté que nos tests en local ne donnaient pas forcément le même résultat que ceux avec l'application. Nous avons donc dû modifier nos scripts afin qu'ils s'adaptent correctement avec les exécutions Python du serveur. Au final, nous avons pu connecter nos deux langages afin qu'ils fonctionnent en harmonie. Nos scripts récupèrent bien les paramètres, appellent les bonnes fonctions et renvoient ce qui est attendu pour assurer l'affichage côté client, traité par TypeScript.

### 3 Positionnement de la solution par rapport à l'existant

Il existe en effet des produits sur le marché pour faciliter l'analyse de données de la même manière que le fait notre application. Cependant, ces produits sont des logiciels à installer destinés aux entreprises comme Tada, le produit créé par MyDataModels, une start-up qui a mis au point une plateforme qui fonctionne également à l'aide du machine learning et des outils d'intelligence artificielle pour permettre aux entreprises d'analyser leurs jeux de données et d'obtenir des modèles prédictifs [15] ou encore comme Dataiku qui propose le même type de produit.

Notre application web, qui est un logiciel applicatif hébergé sur un serveur et accessible via un navigateur web (comme Google Chrome) [16], permet une plus grande accessibilité puisqu'il n'est pas nécessaire de l'installer. Les fonctionnalités de Scanylab ne requièrent aucun devis, donc aucun délai d'attente. C'est de ce fait une évolution comparée aux deux précédents concurrents car nous enlevons la contrainte de l'installation et celle d'affiliation à une entreprise. Ainsi, quiconque souhaitant établir une étude analytique ou souhaitant simplement en apprendre davantage de façon autonome sur la science de données pourra librement se servir de Scanylab.

Il suffit donc seulement de s'inscrire et de se connecter à notre application pour pouvoir utiliser ses fonctionnalités sans quelconque limite. Bien évidemment, si l'application venait à rencontrer un grand succès, il faudrait restreindre certaines fonctionnalités comme la quantité de stockage ou d'analyses disponibles gratuitement mais son utilisation resterait malgré tout plus accessible qu'en passant par des logiciels. De surcroît, si nous venions à améliorer notre produit, l'utilisateur n'aura pas besoin d'effectuer une mise à jour contrairement aux logiciels qui en nécessitent.

Toutefois, il existe une contre-partie. Une sécurité plus importante est nécessaire et une limitation de la bande passante (upload/download). Également, la puissance du serveur doit être assez forte pour réussir à gérer plusieurs requêtes simultanées.

## 4 Conclusion

Nous avons pu développer une application web qui remplit entièrement notre cahier des charges initial. Nous pouvons premièrement conclure que ce projet est une réussite. La plate-forme Scanylab permet de :

- Visualiser
- Prédire
- Automatiser les predictions

Les résultats de l'algorithme autoScan sont au-dessus de nos espérances. Nous avons peur du temps de compilation trop long mais le multithreading de Python est assez performant pour éviter les temps d'attente à rallonge. Cependant, comme prévu, les analyses sur des bases trop volumineuses prennent beaucoup de temps et sont très gourmandes. C'est pourquoi nous avons instauré des limites sur les tailles des bases et sur le nombre d'analyses. Les interactions entre Python et TypeScript nous ont permis de nous organiser et d'avoir une vraie méthode de travail et une logique dans le projet. Ce projet fut aussi très intéressant car il regroupait plusieurs spécialités. Nous devions travailler en équipe et comprendre ce que chaque spécialité faisait et ce fût très pédagogique. Il est nécessaire de continuer ce projet pour aller encore plus loin au niveau de l'analyse et de l'automatisation. En effet, nous avons pensé à un grand nombre de fonctionnalités qu'il est possible d'implémenter :

- Automatisation de tout ce qui est pré-processing (gestion des outliers (valeurs aberrantes), différentes normalisations...)
- Réaliser une V4 d'autoScan qui permettrait d'optimiser les hyperparamètres à l'aide des méta-features.
- Implémenter plus d'algorithmes
- Proposer des études sur les séries temporelles (ARIMA, SARIMA,...)

Nous avons cependant rencontré beaucoup de difficultés comme expliqués dans la partie 2.4.2 et nous avons fait de notre maximum pour les résoudre. Nous tenons réellement à ce projet et nous aimerions le continuer afin de le stabiliser et que tous les étudiants de Polytech'Nice Sophia puissent y avoir accès et mieux comprendre le monde de la data. Cela pourrait notamment les aider dans leur choix de spécialité pour les élèves en Mathématiques Appliquées et Modélisation et en Science de l'Informatique.

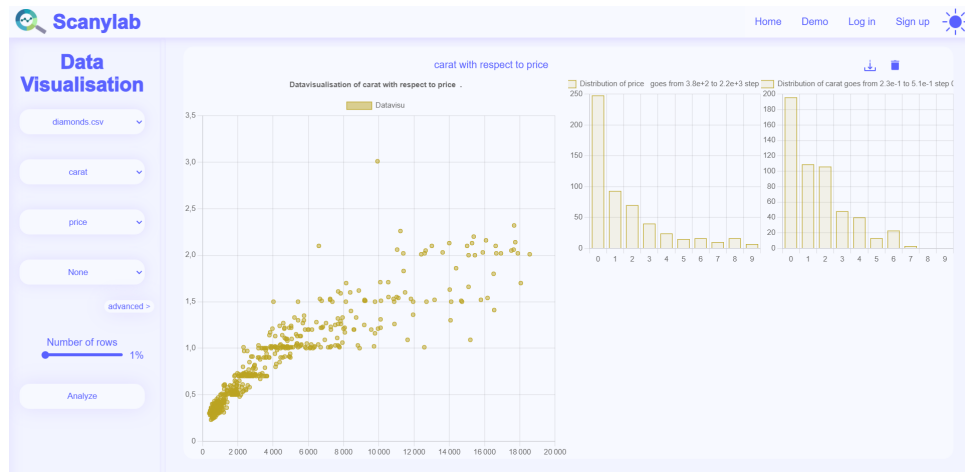
## 5 Sources

### Références

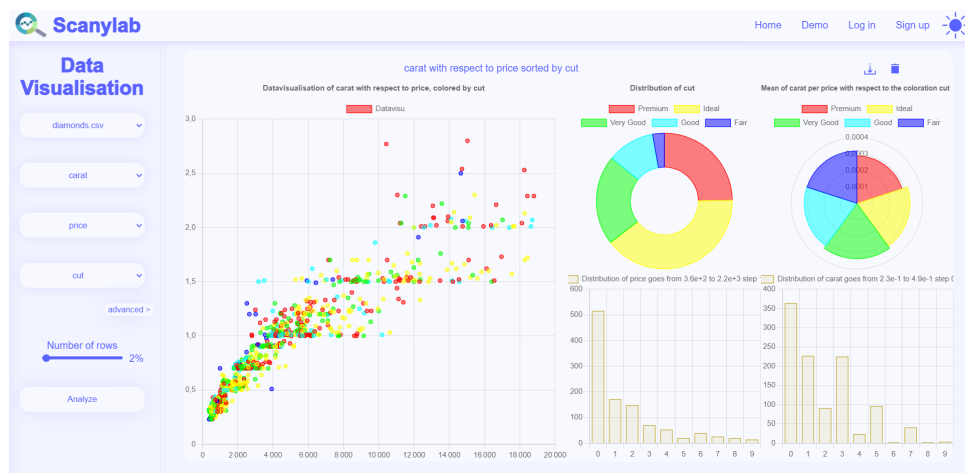
- [1] Emplois Big Data : <https://www.lebigdata.fr/emplois-big-data>
- [2] Scikit-learn : <https://scikit-learn.org/>
- [3] Classification Example with Linear SVC in Python par DataTechNotes le 7/01/2020 : <https://www.datatechnotes.com/2020/07/classification-example-with-linearsvm-in-python.html>
- [4] Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks par Chelsea FINN, Pieter ABBEEL et Sergey LEVINE, 18 juillet 2017 : [https://paperswithcode.com/method/maml?fbclid=IwAR0fKTJ2J7KuDecoKUJDGv-8sKTZtsqlE-i9F1QQm2KEHRnFKaPRS\\_lmqGO](https://paperswithcode.com/method/maml?fbclid=IwAR0fKTJ2J7KuDecoKUJDGv-8sKTZtsqlE-i9F1QQm2KEHRnFKaPRS_lmqGO)
- [5] WayToLearnX - Différence entre JavaScript et TypeScript, 24 mars 2019 : <https://waytolearnx.com/2019/03/difference-entre-JavaScript-et-TypeScript.html>
- [6] JavaScript Flavors 2020 : <https://2020.stateofjs.com/en-US/technologies/JavaScript-flavors/>
- [7] Analytcis & Insights - MongoDB : avantages et inconvénients, 28 février : <https://analytcisinsights.io/mongodb-avantages-inconvenients/>
- [8] Testing JavaScript in 2020 : <https://2020.stateofjs.com/en-US/technologies/testing/>
- [9] Codeur Mag - React, Angular, Vue : quel framework JavaScript choisir ? par Kévin DANGU, 25 février 2022 : <https://www.codeur.com/blog/choisir-framework-JavaScript/>
- [10] JavaScript Front-end Frameworks in 2020 : <https://2020.stateofjs.com/en-US/technologies/front-end-frameworks/>
- [11] WayToLearnX - Différence entre SASS et CSS, 6 avril 2019 : <https://waytolearnx.com/2019/04/difference-entre-sass-et-css.html>
- [12] WayToLearnX - Différence entre SASS et SCSS, 6 avril 2019 : <https://waytolearnx.com/2019/04/difference-entre-sass-et-scss.html>
- [13] OWASP- Recommandation sur le stockage de mot de passe : [https://cheatsheetseries.owasp.org/cheatsheets/Password\\_Storage\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html)
- [14] How-to-use-react-without-unsafe-inline-JavaScript-css-code, 20 mars 2019 : <https://stackoverflow.com/questions/55160698/how-to-use-react-without-unsafe-inline-JavaScript-css-code>

- 
- [15] Les Echos Entrepreneurs - Small data : MyDataModels lève 2,5 millions pour doper les performances par Adrien LELIEVRE, le 09/12/2020 <https://business.lesechos.fr/entrepreneurs/financer-sa-creation/0610094484083-mydatamodels-veut-doper-les-performances-des-professionnels-grace-au-small-data-341083.php>
- [16] Ideematic, Application web, définition d'une application web : <https://www.ideematic.com/dictionnaire-digital/application-web/>

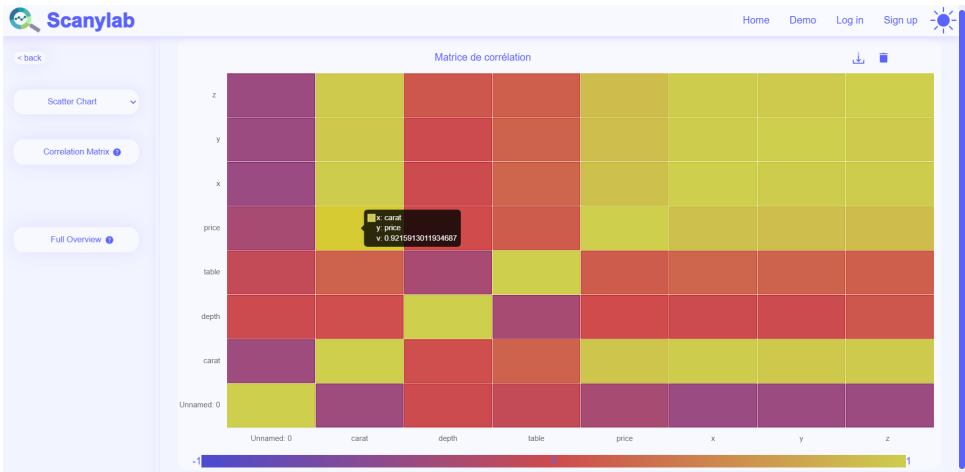
## 6 Annexe



Annexe 1. Exemple de dashboard avec deux paramètres choisis.



Annexe 2. Exemple de dashboard avec trois paramètres choisis.



Annexe 3. Exemple de matrice de corrélation.

### Overview

Overview

Reproduction

Warnings 1

Dataset statistics

Number of variables	11
Number of observations	53940
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	12.6 MiB
Average record size in memory	245.4 B

Variable types

NUM	8
CAT	3

Unnamed: 0

Real number (R<sub>64</sub>)

UNIQUE

Distinct count

53940

Unique (%)

100.0%

Missing

0

Mean

26970.5

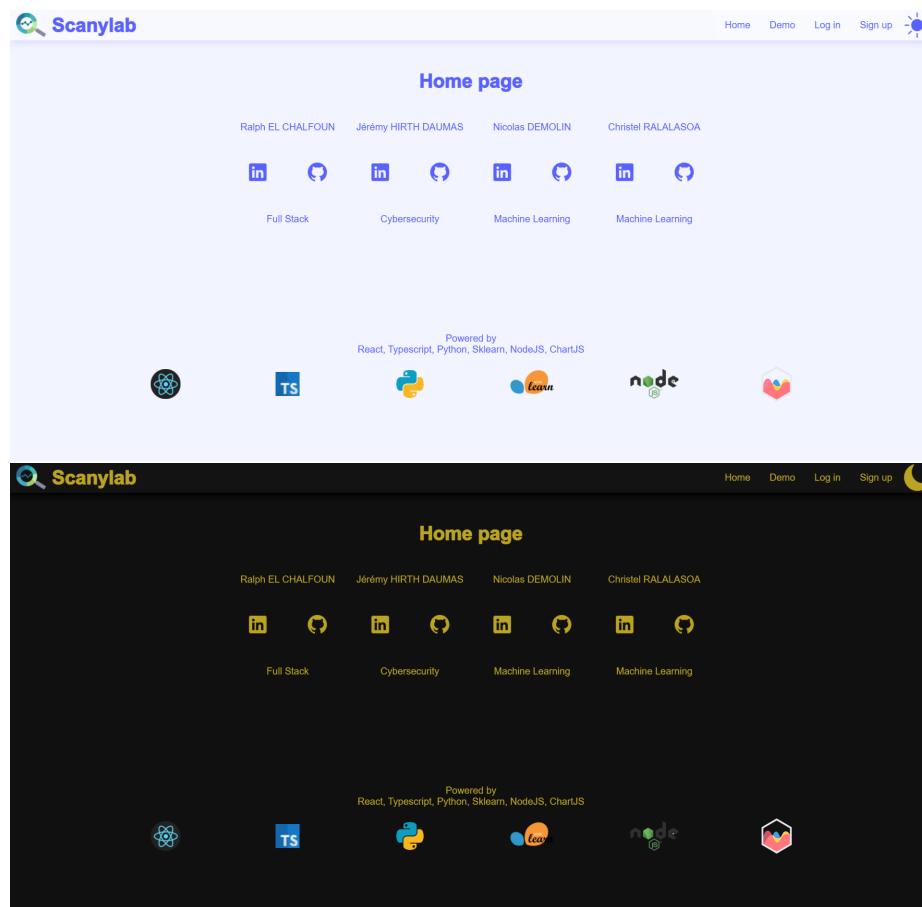
Minimum

1

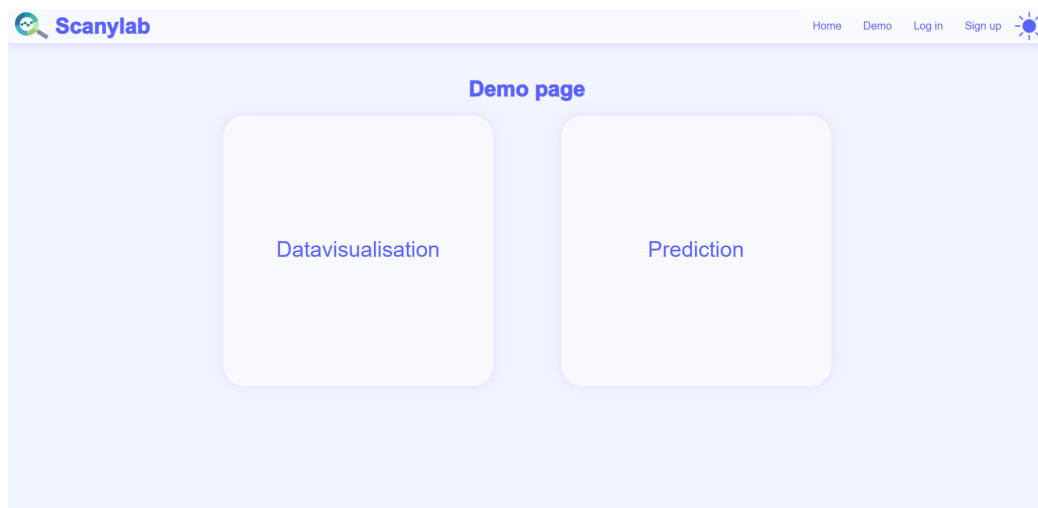
Maximum

53940

Annexe 4. Exemple de page HTML Overview.



Annexe 5. Thème jour et nuit sur la page d'accueil.



Annexe 6. Page de démo.




The screenshot shows the 'Analyze page' of the Scanylab interface. At the top, there is a navigation bar with 'Home', 'Demo', 'Database', 'Analyze', and 'Disconnect' links. The main content area is titled 'Analyze page'. It features two dropdown menus: 'diamonds.csv' and 'price'. Below these, a 'Choose other parameters' section lists several features: 'carat' (checked), 'cut' (checked), 'color' (unchecked), 'clarity' (unchecked), 'depth' (unchecked), 'table' (checked), 'x' (unchecked), 'y' (unchecked), and 'z' (unchecked). Underneath, there are four buttons for different analysis methods: 'AUTOMATIC' (highlighted in blue), 'GRADIENT BOOSTING', 'RANDOM FOREST', and 'RIDGE'. A note states: 'This is option chooses the best algorithm to get the best result.' Below the buttons, there is a text input field for 'Analyze name...', a status indicator 'Sending request, please wait...' with a circular progress bar, and an 'Analyze' button.

**Annexe 7.1.** Exemple de page d'analyse. (Ici requête d'une régression Automatique.)



The screenshot shows the 'Analyze page' of the Scanylab interface, configured for a Random Forest classification. The navigation bar is the same as in the previous image. The 'diamonds.csv' dropdown is unchanged, but the target variable dropdown is now set to 'cut'. The 'Choose other parameters' section is collapsed. Below, the analysis method buttons are 'AUTOMATIC', 'LINEARSVC', 'ADABOOST', 'GRADIENT BOOSTING', 'RANDOM FOREST' (highlighted in blue), and 'LOGISTIC REGRESSION'. A descriptive text for Random Forest is provided: 'A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.' Below this, there are four parameter controls: 'n\_estimators' (set to 100), 'max\_depth' (set to 3), 'min\_samples\_split' (set to 2), and 'class\_weight' (set to 'none'). At the bottom, there is a 'classification' label, the same 'Sending request, please wait...' status indicator with a progress bar, and an 'Analyze' button.

**Annexe 7.2.** Exemple de page d'analyse. (Ici requête d'une classification Random Forest.)

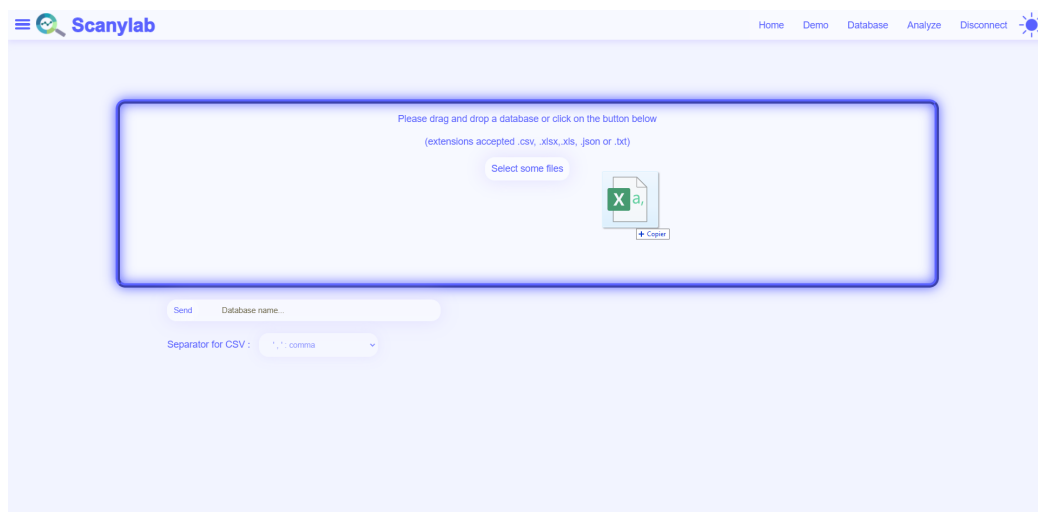


Home Demo Log in Sign up   
**Welcome !**  
Sign up to create an account  
  
ex : Alexandre  
  
ex : Dubois  
  
example@mail.com  
  
\*\*\*\*\*  
  
\*\*\*\*\*  
   

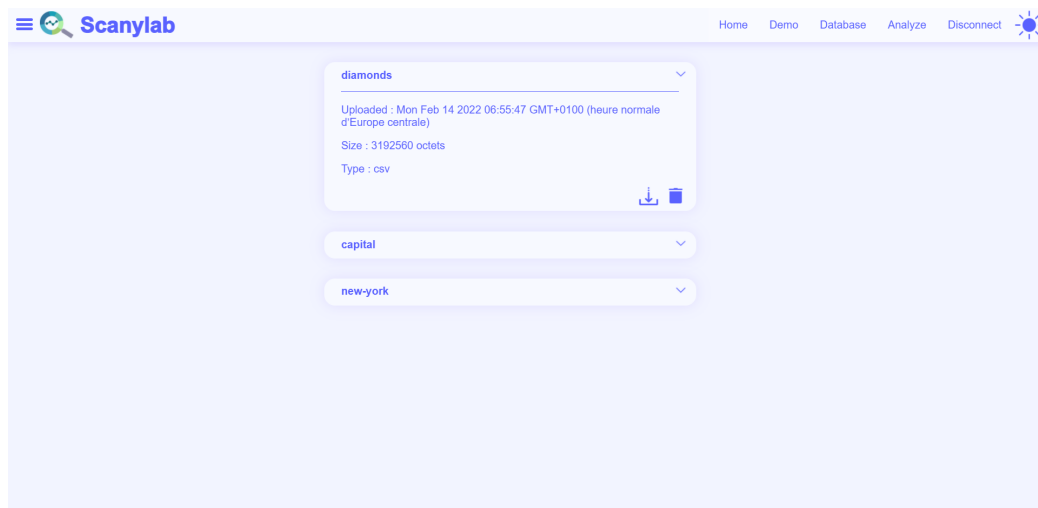
Annexe 8. Page d'inscription.

Home Demo Log in Sign up   
**Welcome !**  
Log in to your account  
  
example@mail.com  
  
\*\*\*\*\*  
  
Or  

Annexe 9. Page de connexion.



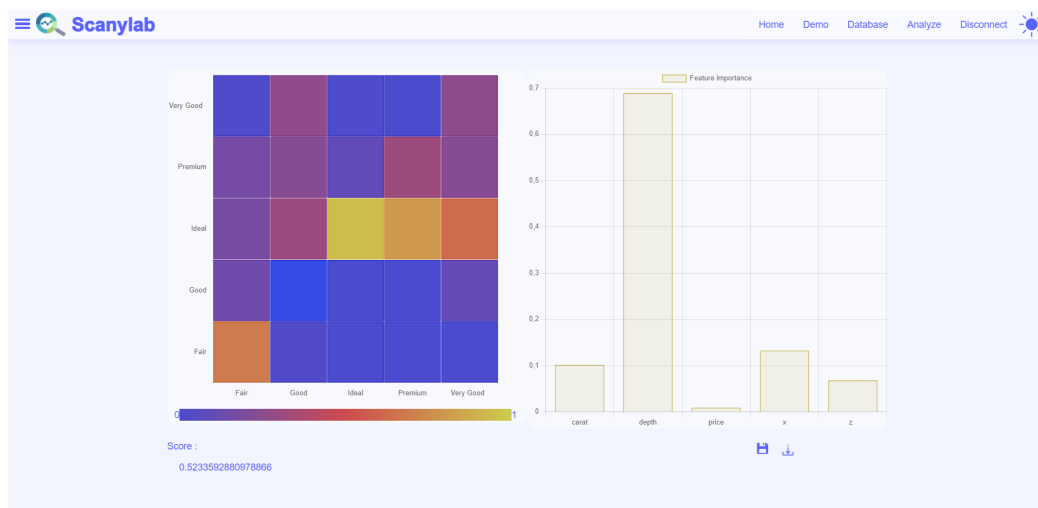
Annexe 10. Page d'upload.



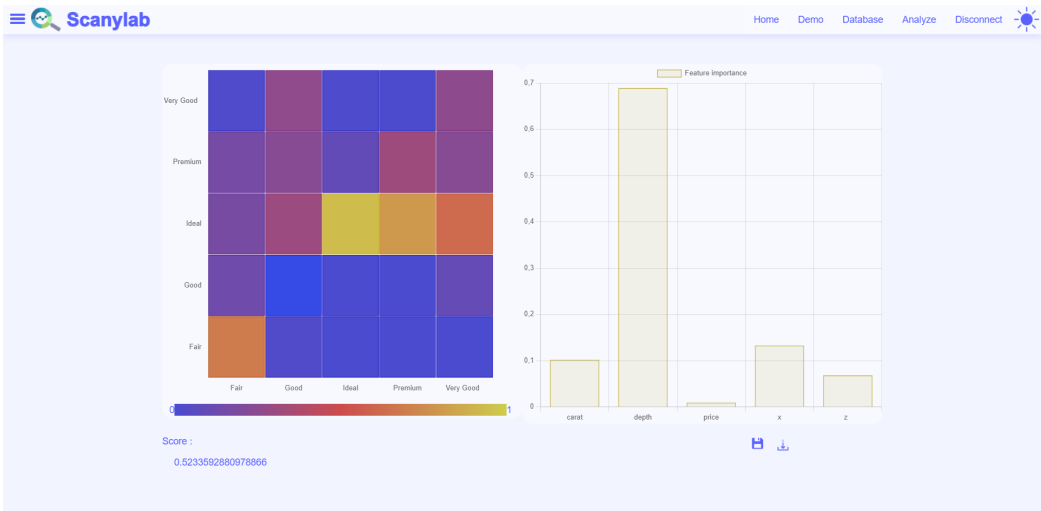
Annexe 11. Exemple de page My Databases.



**Annexe 12.** Exemple de résultat d'une classification avec l'algorithme automatique.



**Annexe 13.** Exemple de résultat d'une régression.



Annexe 14. Exemple de résultat d’une classification.



Annexe 15. Exemple de page historique d’analyse.