École Centrale Lyon

UE PRO - PAi
Rapport Final

Projet d'Application industriel n°14

Élève :
Emmie CLUZEL
Antoine EDY
Antoine MISSUE
Léa PIERRAT
Léonardo SANTIAGO
Justin SENN-COCTEAU
Adnane SENNOUNE

Commanditaire :
Philippe ROCHETTE

Tuteur :
Bertrand VILQUIN

22 février 2024

Résumé :

Ce rapport rend compte du travail que nous avons effectué dans le cadre du Projet d'Application Industriel (PAi 14) dont le commanditaire est STMicroelectronics. Ce projet a été réalisé tout au long de la deuxième année du cursus d'ingénieur généraliste de l'École Centrale de Lyon. L'objectif de ce projet est de mettre en place une méthode d'analyse compétitive des composants semi-conducteurs par le biais d'un réseau neuronal. Le livrable de ce projet est une application destinée aux ingénieurs de STMicroelectronics. Ce rapport a pour objectif de détailler les raisonnements, questionnement et évolutions de nos pensées avant d'arriver à l'application finale que nous présentons à ce jour à l'entreprise commanditaire. La restitution orale de ce projet aura lieu le vendredi 28 avril 2023 à Écully, sur le campus de l'école.

Table des matières

Table des figures

1 Introduction

1.1 Mise en contexte

STMicroelectronics est une entreprise multinationale franco-italienne spécialisée dans la conception et la fabrication de semi-conducteurs et de circuits intégrés. Fondée en 1987, elle est aujourd'hui l'un des plus grands fabricants de semi-conducteurs au monde, avec des bureaux et des usines dans plus de 30 pays. Les produits de STMicroelectronics sont utilisés dans une grande variété d'applications, notamment les technologies de l'information et de la communication, l'automobile, l'aérospatiale, l'énergie et la sécurité.

L'un de ses besoins phare est la connaissance du catalogue de ses concurrents, qui est donc primordiale pour préserver cette position de leader sur le marché. Ce besoin est, avec la multiplication des composants disponibles sur le marché, de moins en moins évident à satisfaire et l'entreprise doit déployer de nouvelles méthodes pour parvenir à sa fin.

L'entreprise dispose de bases de données riches de composants disponibles sur le marché, donc de divers constructeurs. Actuellement, chez STMicroelectronics, l'identification des composants concurrents est faite par famille de produits de manière manuelle, c'est-à-dire en ajustant avec les paramètres que l'on souhaite. Or, le catalogue de ST étant très important et le marché évoluant rapidement, il devient nécessaire d'automatiser la comparaison des composants. Il est intéressant également de ne pas seulement pouvoir comparer les composants à un instant t, mais de pouvoir suivre l'évolution du marché au cours du temps, au fil des ajouts (de l'ordre d'une centaine de composants à un millier par semaine).

STMicroelectronics nous a donc fourni une de leurs bases de données contenant les caractéristiques de plusieurs composants semi-conducteurs provenant aussi bien de leur catalogue que de celui d'entreprises concurrentes. Grâce notamment à l'ensemble des propriétés de ces composants, mais aussi à l'expertise des ingénieurs de chez STMicroelectronics, nous tenterons d'identifier de manière automatisée les composants d'autres fabricants qui entrent directement en compétition avec chaque produit du catalogue de STMicroelectronics.

1.2 Problématique et objectifs

Il est relativement aisé de parvenir à classer des composants électroniques en connaissant leurs caractéristiques, et de nombreux algorithmes dits de *clustering* (littéralement de regroupement) peuvent être utilisés (les principes de K-means, de DBSCAN ou de *hierarchical clustering* en sont des exemples généralement performants parmi tant d'autres). Ces regroupements sont purement mathématiques et basés sur des distances : ils ne reflètent pas toute la complexité des composants électroniques et de leurs utilisations multiples. Ce savoir est détenu par les ingénieurs de STMicroelectronics qui, à force d'expérience, connaissent les composants et leurs utilisations et savent des choses que les chiffres ne peuvent décrire. Un réseau complexe devra être mis en place pour prendre en compte ces deux paramètres qui caractérisent chaque composant : les caractéristiques physiques précises des composants et le savoir des ingénieurs qui les utilisent depuis des années.

Les enjeux de notre projet se synthétisent en la problématique suivante : comment mettre en place un algorithme de scoring, basé sur l'intelligence artificielle, qui permettrait de trouver le niveau de compétitivité d'un composant par rapport à un autre ?

Nous avons ainsi établi plusieurs objectifs ; nous devons identifier un algorithme qui permet de trouver les compétiteurs directs d'un composant et d'établir un score pour les comparer. Toute la complexité de cet objectif réside dans le fait de parvenir à prendre en compte efficacement les informations fournies par les ingénieurs de STMicroelectronics.

2 L'avancement du projet

2.1 Les étapes du projet

Nous pouvons décomposer les principales étapes en deux temps :

- Les étapes nécessaires avant le développement de l'application, qui nous ont permis d'en apprendre davantage sur le traitement et la manipulation de données :
  - Suivre la formation "Python appliqué aux Data Sciences",
  - Suivre la formation "Initiez-vous au Machine Learning" et état de l'art des méthodes existantes,
  - Réflexion sur l'environnement de travail adapté pour respecter l'accord de confidentialité signé.
- D'autres, ensuite, sur le traitement des données fournies et le développement de l'application :
  - Extraire et traiter les données intéressantes de la base de données fournie par STMicroelectronics,
  - Mettre en place l'algorithme de classification grâce à des méthodes basées sur l'intelligence artificielle (réseaux neuronaux),
  - Mettre en place une structure d'application qui permet aux ingénieurs de STMicroelectronics d'affiner l'algorithme,
  - Estimer efficacité et coût de l'algorithme.

2.2 GANTT

Ci-dessous le diagramme GANTT de notre projet, qui représente visuellement l'état d'avancement des différentes tâches qui constituent notre PAi.

PAi 14 - Diagramme de GANTT			Octobre			Novembre				Décembre				Janvier				Février				Mars				Avril			
Tâche	Start Date	End Date	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
Suivre la formation « Python appliqué aux Data Sciences » sur OCR	07/10/23	31/10/23			CONGÉS							CONGÉS								CONGÉS							CONGÉS		RESTITUTION
Reflexion sur l'environnement de travail en adéquation avec la confidentialité	15/10/23	31/10/23																											
Formation « Initiez-vous au Machine Learning » sur OCR et état de l'art	02/11/23	15/12/23																											
Extraire et traiter les données intéressantes des bases de données STMicroelectronics	08/11/23	08/12/23																											
Mettre en place l'algorithme de scoring grâce à l'IA	01/12/23	28/02/23																											
Mettre en place l'apprentissage de l'algorithme	20/01/23	12/03/23																											
Estimer l'efficacité de l'algorithme	12/03/23	07/04/23																											
Rédaction du rapport final	15/03/23	17/04/23																											

FIGURE 1 – Diagramme GANTT du projet

2.3 Budget

Le budget total est de 0€. Nous n'avons eu besoin que de formations et ressources gratuites en ligne pour développer cette application, son déploiement à l'échelle de l'entreprise entière n'étant évidemment pas un objectif premier.

3 Le traitement des données brutes

3.1 Traitement de tableaux

Le support de travail que nous avons reçu est une base de données composée d'un peu plus de 4 millions d'entrées. Une étape cruciale de tout projet en data-science est celui de l'analyse de ces données (contenu, format) : elle forme sa branche de métier à part entière, celle de *data cleaner*. Dans un premier temps, il a donc fallu mettre en forme cette base de données brute.

Après avoir suivi une formation en ligne de Nicolas RANGEON [8] nous avons fait le choix d'utiliser la librairie Pandas et ses dataframes pour traiter ces tableaux. Le dataframe est un objet Python qui permet de représenter une base de données sous la forme d'un tableau. Chaque colonne y est explicitement nommée et chaque case (croisement d'une ligne et d'une colonne) contient un objet. Cette organisation simplifie l'accès aux variables, et permet de nombreuses manipulations de données plus ou moins complexes. Le choix de cette bibliothèque a été motivé par les raisons suivantes :

- **Manipulation de données :** Les dataFrames Pandas offrent une multitude de fonctions et de méthodes qui facilitent la manipulation de données : effectuer des opérations de filtrage, tri, groupement, agrégation et fusion de données, ainsi que des calculs statistiques et mathématiques.
- **Flexibilité :** Les dataFrames Pandas sont très flexibles et peuvent gérer différents types de données, notamment les données tabulaires et les séries chronologiques. Ils peuvent également prendre en charge les données manquantes et les valeurs nulles.
- **Intégration aisée :** Les dataFrames Pandas peuvent être facilement intégrés à d'autres bibliothèques Python, telles que Plotly pour la visualisation de données ou Scikit-learn et PyTorch pour l'apprentissage automatique.
- **Gestion des données volumineuses :** Les dataFrames Pandas sont conçus pour gérer des ensembles de données volumineux.
- **Performance :** Les dataFrames Pandas sont optimisés pour la performance et peuvent traiter rapidement des données volumineuses. Les opérations vectorisées et l'utilisation de Numpy sous-jacent pour les calculs garantissent également une performance élevée.
- **Lecture et écriture de fichiers :** Les dataFrames Pandas peuvent être utilisés pour lire et écrire des données à partir de fichiers dans différents formats, tels que CSV, Excel, JSON, SQL et HDF5.

Une série d'opérations nous ont permis de passer d'une base de données brute à une base de données ordonnée avec comme clef principale : `KEY_MM` au niveau de la deuxième colonne et toutes les informations associées dans les colonnes suivantes comme le montre la figure ci-dessous :

Base de données brute

	A	B	C	D	E	F	G	H
1	ProdLine,ManufCode,KEY_MM,KEY_FN,FeatName,FeatUnit,FeatMax,FeatMin,FeatValTyp							
2	AC to DC Switching Converters,"D",125962,3089151,"Topology","",,,							
3	AC to DC Switching Converters,"D",125962,3089152,"Soft Start Option","",,,							
4	AC to DC Switching Converters,"D",125962,3089153,"Switching Frequency","kHz",71,62,66.5							
5	AC to DC Switching Converters,"D",125962,3089154,"Minimum Under Voltage Lock Out","V",11.2,11.2,11.2							
6	AC to DC Switching Converters,"D",125962,3089155,"Typical Under Voltage Lock Out","V",12.2,12.2,12.2							
7	AC to DC Switching Converters,"D",125962,3089156,"Maximum Under Voltage Lock Out","V",13.2,13.2,13.2							
8	AC to DC Switching Converters,"D",125962,3089157,"Minimum Supply Voltage","V",-0.4,-0.4,-0.4							
9	AC to DC Switching Converters,"D",125962,3089158,"Maximum Supply Voltage","V",30,30,30							
10	AC to DC Switching Converters,"D",125962,3089159,"Maximum Operating Current","mA",0.4,0.4,0.4							
11	AC to DC Switching Converters,"D",125962,3089160,"Temperature Flag","",,,							
12	AC to DC Switching Converters,"D",125962,3089161,"Minimum Operating Temperature","Â°C",-40,-40,-40							
13	AC to DC Switching Converters,"D",125962,3089162,"Maximum Operating Temperature","Â°C",150,150,150							
14	AC to DC Switching Converters,"D",125962,3089163,"Mounting","",,,							
15	AC to DC Switching Converters,"D",125962,3089164,"PACKAGE_DIMENSION_H","",1.45,1.45,1.45							
16	AC to DC Switching Converters,"D",125962,3089165,"PACKAGE_DIMENSION_L","",5,5,5							
17	AC to DC Switching Converters,"D",125962,3089166,"PACKAGE_DIMENSION_W","",4,4,4							
18	AC to DC Switching Converters,"D",125962,3089167,"PCB","",8,8,8							

Base de données triée

	FeatName	KEY_MM	Maximum Operating Current	Maximum Operating Temperature	Maximum Supply Voltage	Minimum Operating Temperature	Minimum Supply Voltage	PACKAGE_DIMENSION_H	PACKAGE_DIMENSION_L	PACKAGE_DIMENSION_W	P
0	22248	3.5	150.0	23.5	-40.0	8.50	3.3	9.27	6.35		
1	22249	2.5	150.0	23.5	-40.0	8.50	3.3	9.27	6.35		
2	24392	1.5	150.0	23.5	-40.0	11.50	3.3	9.27	6.35		
3	25454	17.0	150.0	32.0	-40.0	3.75	1.3	2.90	1.60		
4	35136	17.0	150.0	32.0	-40.0	3.75	1.3	2.90	1.60		
5	35137	17.0	150.0	32.0	-40.0	3.75	1.3	2.90	1.60		
6	35862	1.5	150.0	30.0	-40.0	4.50	1.5	4.90	3.90	1	
7	35917	1.8	150.0	30.0	-40.0	4.50	1.5	4.90	3.90	1	
8	36445	1.8	150.0	30.0	-40.0	4.50	1.5	4.90	3.90	1	
9	36776	1.2	150.0	30.0	-40.0	4.50	1.5	4.90	3.90	1	

FIGURE 2 – Traitement du tableau de données

Cette structure de base de donnée nous permet de facilement accéder à toutes ses valeurs. Nous devons à présent faire le travail du *data cleaner*, c'est-à-dire nettoyer la base de donnée en ne gardant que les lignes et les colonnes utilisables - donc comportant des valeurs numériques. Il est en effet fréquent dans une base de données de rencontrer des valeurs physiquement incohérentes (des dimensions de 0 par exemple) ou des *Nan*, *Not a Number*, donc des valeurs inutilisables.

Nous avons fait le choix, pour chaque famille, de supprimer les colonnes (donc les caractéristiques des composants) qui comportait plus de 95% de *NaN* ou de 0, ainsi que les lignes (donc les composants) qui comportait plus de 80% de *NaN*. Ces valeurs sont empiriques ; après plusieurs essais, nous avons constaté qu'elles permettaient de conserver un bon nombre des composants présent dans la base de donnée (un peu de 75% de ces composants sont conservés) et de garder les caractéristiques physiques qui importent. Une étude plus précise et une meilleure connaissance des composants électroniques permettrait probablement de sauver encore quelques entrées du dataframe. Comme évoque plus tôt, un métier entier est dédié à cela et notre solution est déjà très satisfaisante vis-à-vis de nos besoins.

3.2 Un outil de visualisation : le PCA

Le PCA, ou *Principal Component Analysis* (Analyse en Composantes Principales en français) est une technique mathématique d'analyse de données qui permet de réduire la dimensionnalité d'un ensemble de données multivariées tout en conservant le maximum d'information possible.

En pratique, la méthode PCA transforme un ensemble de données en un ensemble de vecteurs linéairement indépendants appelés "composantes principales". Ces composantes principales sont triées par ordre d'importance en termes de variance expliquée, ce qui permet de déterminer les dimensions principales du jeu de données. La perte d'information est minimisée lors de ce processus. En visualisant les données dans ce nouvel espace, il est possible de mettre en évidence les relations et les similarités entre les points de données qui étaient auparavant cachées dans les dimensions d'origine. Cette projection permet ainsi de visualiser les données sous une forme plus simple et plus compréhensible et alors de faciliter leur interprétation.

Nous allons nous servir du PCA pour réduire la dimension de notre base de donnée de 11 initialement (les 11 caractéristiques des composants) à 3, ce qui nous permettra de visualiser nos composants dans un espace à trois dimensions, comme sur la figure suivante ou chaque point représente un composant appartenant à la famille *AC to DC converters*.

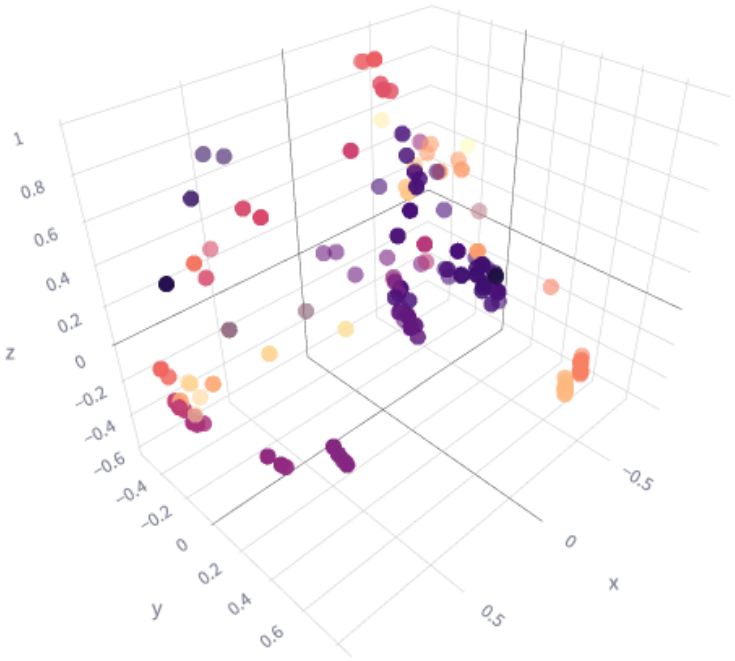

FIGURE 3 – Visualisation des composants de la famille *AC to DC converters* dans un espace à 3 dimensions

Ici, les couleurs représentent les classes trouvées par le réseau neuronal. Cette visualisation nous permet de vérifier que les résultats sont cohérents : deux composants proches dans l'espace, donc de caractéristiques similaires, se retrouvent dans la même famille. Ce

principe sera détaillé par la suite et notamment en section 4.4.

Cependant, cette réduction de dimension s'accompagne forcément d'une perte d'information. Dans notre cas, celle-ci est très faible, de l'ordre de 10%. Cette valeur est faible pour une telle réduction de dimension ($11 \rightarrow 3$), ce qui signifie que de nombreuses caractéristiques sont similaires ou quasiment proportionnelles.

4 Les détails techniques de notre approche

Afin de pouvoir suivre l'évolution du marché autour de différents composants électroniques, il faut être capable de déterminer quels composants appartiennent au même marché, donc quels composants sont similaires, ou au moins répondent aux mêmes besoins clients : se pencher sur la notion de *clustering* semble alors tout à fait naturel. On nommera "classe" un ensemble de composants similaires. Ces classes seront déterminées par le réseau de neurone, dont les détails figurent en section 4.4.

4.1 Des objectifs pratiques

Ce projet a donné lieux à l'exploration de nombreuses pistes autour de la question du *clustering*. Il était essentiel de maitriser les différents outils à notre disposition à ce sujet. Nous avons ainsi expérimenté de nombreuses méthodes avant d'arriver à celle choisie ; cette section du rapport revient sur les principales et explique en quoi elles nous ont aidé à avancer dans notre réflexion.

4.2 Avantages et inconvénients des pistes employées

4.2.1 Qu'est-ce que le clustering ?

Le clustering consiste aux pratiques mis en œuvre pour découvrir des groupes dans un ensemble de données non étiquetées [5]. L'objectif est de trouver, à partir d'un ensemble de données, la meilleure partition pour ces données. Pour rappel, une partition d'un ensemble X est un ensemble de parties, deux à deux disjointes et dont l'union est X. Pour trouver cette "meilleure partition" on peut raisonner au sens de fonction objectif mais également en fonction d'une métrique (distance mathématique entre plusieurs objets).

4.2.2 Méthode de clustering supervisé

Dans un algorithme supervisé, les données sont étiquetées en fonction de leur classe ou de leur catégorie. On donne en entrée un ensemble d'objets ainsi qu'une liste d'étiquette, et l'on demande à l'algorithme d'attribuer à chaque objet une étiquette. Un exemple pourrait être le suivant : un individu à un sac rempli de pommes et d'oranges. L'ensemble des objets est l'ensemble des fruits qu'il possède, et les étiquettes sont au nombre de deux, "orange" et "pomme". Il met quelques oranges dans un panier à oranges et quelques pommes dans un panier à pommes ; l'algorithme se charge de remplir les paniers en suivant ce principe d'étiquettes.

4.2.3 Méthode de clustering non supervisé

Dans un algorithme de clustering non supervisé, les données sont regroupées en fonction de leurs similarités. Cette fois-ci, les entrées sont seulement les objets, il n'y a pas d'étiquettes prédéfinies : le résultat sera donc des groupes d'objets similaires sans nom (groupe 1, groupe 2, ... groupe n). Un exemple pourrait être le suivant : on considère un même individu qui possède de nombreux fruits et légumes. Cette fois-ci, il n'y a pas d'étiquettes, c'est l'algorithme qui fait tout de manière autonome. Dans un premier panier, on pourrait retrouver des pommes et des oranges, car ils ont une forme ronde d'environ la

même taille. Dans une deuxième panier, on pourrait retrouver des poires et des aubergines également à cause de leur forme ou de leur poids.

Ces deux premières méthodes de clustering peuvent être simplement mise en place en python, notamment grâce à la bibliothèque très complète de machine-learning scikit-learn [1].

4.2.4 Le clustering semi-supervisé

Ce dernier type de clustering - de plus en plus répandu, car répondant à une complexification des besoins - est une technique de classification qui combine des éléments de clustering non supervisés et supervisés. Avec cette méthode, certaines données sont étiquetées, tandis que d'autres ne le sont pas. Les données étiquetées sont utilisées pour guider le regroupement des données non étiquetées. Cette technique est particulièrement adaptée pour notre projet, car les données peuvent être dans certains cas difficiles à étiqueter. Le clustering semi-supervisé permet de tirer parti de ces exemples étiquetés pour améliorer la précision du clustering global. Pour vulgariser, on peut considérer que la distance mathématique entre deux composants représente la partie non supervisée (ou les composants similaires sont mis dans la même classe) et que l'ingénieur de chez STMicroelectronics va littéralement superviser le clustering, en y forçant plusieurs composants à appartenir à une même classe par exemple.

4.2.5 Méthodes et performance du clustering

Il existe plusieurs méthodes de clustering, chacune ayant ses avantages et ses inconvénients. Voici un graphe comparatif des performances des méthodes les plus courantes et qui ont été utilisées pour tester les performances sur nos données :

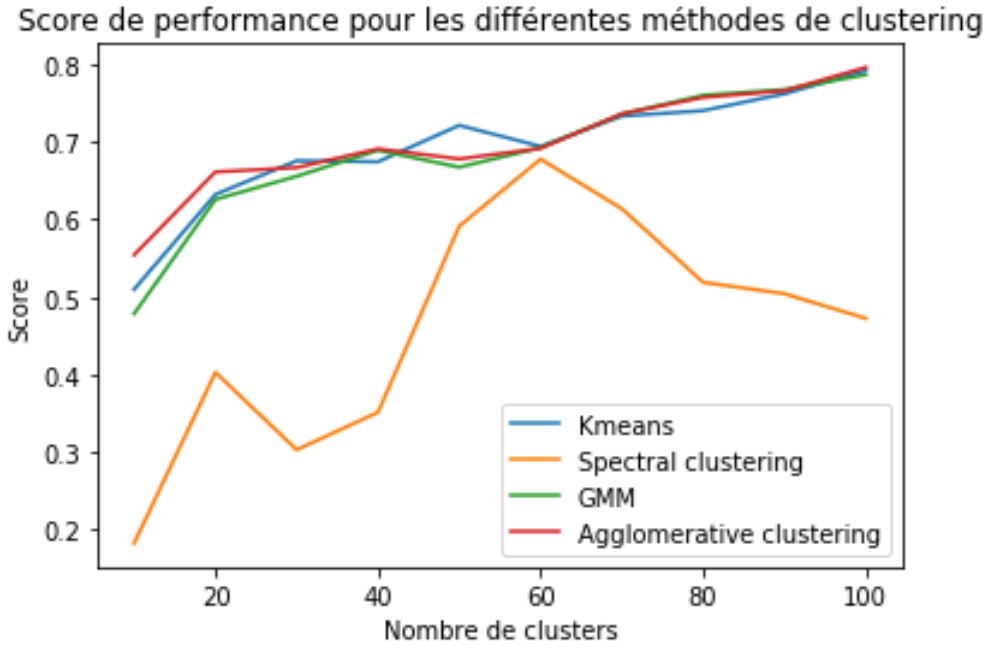

FIGURE 4 – Comparaison de différentes méthodes de clustering

Le score sur l'axe des ordonnées représente le score de silhouette, qui est une mesure de la qualité du clustering. Il évalue à quel point chaque point de données est similaire

aux autres points dans son propre cluster par rapport aux autres clusters. Le score de silhouette peut varier de -1 à 1, où une valeur proche de 1 indique que les points sont bien regroupés dans leur propre cluster et éloignés des autres clusters, tandis qu'une valeur proche de 0 indique que les points sont proches des frontières entre les clusters et qu'ils pourraient être affectés à différents clusters. Un score de silhouette négatif signifie que les points de données sont plus proches des clusters voisins que de leur propre cluster, ce qui est un signe d'un clustering non réussi.

Nous avons observé que les scores donnés par les différentes méthodes sont bons (scores supérieurs à 0.7 à partir d'un nombre de clusters égal à 30) et sensiblement équivalents, sauf pour le clustering spectral. Cela indique que les clusters obtenus avec cette dernière méthode sont moins cohérents et que les points de données sont plus proches des frontières entre les clusters. Cela peut être dû au fait que le clustering spectral fonctionne mieux avec des données linéairement séparables, tandis que les autres méthodes de clustering peuvent gérer des données de formes plus complexes.

Étant donné que K-means est facile à implémenter, efficace pour les grands ensembles de données et fournit des clusters cohérents, il semble être le meilleur choix pour notre cas d'utilisation. Par conséquent, nous avons décidé d'utiliser K-means pour la suite de l'étude.

4.3 Les bibliothèques et langages informatiques utilisés

Python est devenu le langage classique de programmation pour le traitement des données, et ce pour deux principales raisons :

- **Son large écosystème de bibliothèques :** Python dispose d'un grand nombre de bibliothèques spécialisées dans la manipulation, la visualisation et l'analyse de données, telles que NumPy, Pandas, Matplotlib, Scikit-learn ou PyTorch, autant de bibliothèques que nous avons été amenés à utiliser au cours de notre projet.
- **Sa facilité d'intégration :** Python est facilement intégrable avec d'autres langages et plateformes, ce qui en fait un choix idéal pour la data science. Par exemple, il permet de créer facilement des applications de data science grâce à Dash [7] ou Streamlit [2], que nous avons utilisé.

Nous avons utilisé un certain nombre de ces bibliothèques :

- **Pandas** [4] a été utilisé pour le traitement des données, y compris la transformation des données, la manipulation des données manquantes, la normalisation des données et l'analyse exploratoire des données.
- **Scikit-learn** [1] a été utilisé pour la partie non supervisée du projet, en particulier pour l'algorithme de clustering k-means.
- La bibliothèque **Streamlit** [2] a été utile pour créer une interface utilisateur interactive. Streamlit est une bibliothèque Python open-source qui permet de créer facilement des applications web pour les projets de science des données et d'apprentissage automatique. Avec Streamlit, nous avons pu créer une interface utilisateur en quelques lignes de code. Nous avons utilisé les fonctionnalités de Streamlit pour créer des graphiques qui ont permis une exploration interactive des données.
- Enfin, nous avons utilisé **PyTorch** [3] pour mettre en œuvre un réseau neuronal pour la classification. L'utilisation de PyTorch s'est due à une offre d'une grande

flexibilité pour la construction de réseaux de neurones et une grande facilité d'utilisation pour l'entraînement des modèles. Nous détaillerons la structure des réseaux neuronaux utilisés dans la section suivante.

4.4 La méthode de classification : un réseau de neurones

4.4.1 Pourquoi un réseau de neurones ?

Pour mener à bien ce projet, nous passons par l'utilisation d'un réseau de neurones. Les réseaux de neurones sont fréquemment utilisés car ils permettent de résoudre divers problèmes, et notamment, comme dans notre cas, pour mettre en place des méthodes de classification. Les réseaux de neurones peuvent apprendre à reconnaître des motifs dans les données d'entrée et à les utiliser pour effectuer des prédictions précises sur de nouveaux exemples. Il est alors possible de créer un modèle généralisable à partir des données d'apprentissage pour classer de nouveaux exemples avec une grande précision. Cela est particulièrement utile dans l'algorithme pensé ici afin de comparer et conseiller tel ou tel composant en fonction d'un composant cible et de critères précis [6].

L'intérêt du réseau de neurone est donc sa capacité à "apprendre" de manière autonome à partir des données, sans avoir besoin d'une programmation explicite. Cela les rend très efficaces pour traiter des problèmes complexes pour lesquels il est difficile de formuler des règles.

4.4.2 La bibliothèque PyTorch

PyTorch est une bibliothèque open-source de calculs numériques et de machine learning basée sur Python [3]. Elle fournit des outils pour la création de réseaux de neurones artificiels et permet de définir et d'entraîner des modèles de machine learning en utilisant des techniques telles que la descente de gradient stochastique, les réseaux de neurones convolutionnels, les réseaux de neurones récurrents, et les modèles de transformation de données tels que les auto-encodeurs.

4.4.3 La structure des réseaux neuronaux utilisée

Le principe d'un réseau de neurones est de modéliser les processus cognitifs du cerveau en utilisant des unités de traitement appelées "neurones". Un réseau de neurones est composé de plusieurs couches de neurones interconnectées, qui traitent l'information en la faisant passer de couche en couche.

Le principe de base d'un neurone est le suivant : il reçoit des signaux d'entrée pondérés, les somme, et applique une fonction d'activation pour produire une sortie. La sortie du neurone peut ensuite être transmise à d'autres neurones, créant ainsi des connexions entre les neurones. Le processus de transmission de l'information à travers ces connexions est appelé "propagation avant". L'objectif du réseau de neurones est de trouver la meilleure combinaison de poids de chaque neurone pour produire la sortie souhaitée en réponse à une entrée donnée. Le réseau ajuste les poids de chaque neurone en fonction de l'erreur entre la sortie produite et la sortie souhaitée.

Une fois que le réseau a appris à produire des sorties précises en réponse à des entrées données, il peut être utilisé pour effectuer des tâches telles que la classification, la prédiction, la reconnaissance de formes, la génération de texte et d'images, entre autres. Dans

notre cas, c'est la classification qui nous intéresse.

La structure d'un réseau de neurones est l'un des éléments clés de sa performance, et influe sur sa capacité à apprendre, sur sa rapidité, sur sa capacité à généraliser ou au contraire à "sur-apprendre" les données d'entrainement (on parle alors d'*over fitting*) et sur sa consommation de ressources.

Étudions maintenant la structure choisie. Notre réseau est composé de N_{carac} entrées, d'une couche cachée (*hidden layer*) composée de 15 neurones et de $N_{classes}$ sorties. Les valeurs de N_{carac} et de $N_{classes}$ varient selon la famille de composant sélectionnée, et représentent respectivement le nombre de caractéristiques retenues pour les composants et le nombre de classes. Le nombre de classes pour une famille est calculé comme suit :

$$N_{classes} = \lfloor \frac{N_{composants}}{10} \rfloor$$

avec $N_{composants}$ le nombre de composants de la famille. Cela permet d'avoir une dizaine de composants par classe, ce qui est en accord avec les résultats voulus.

Expliquons cette structure :

— Le nombre d'entrées N_{carac} est donc le nombre de caractéristiques des composants : chacune de ces valeurs sera mise à l'échelle (afin de leur accorder un poids similaire - a priori) donc compris entre -1 et 1. On appellera ces valeurs $E_1, E_2, ...E_{N_{carac}}$.

— Le nombre de couches cachées (15) est choisi arbitrairement. Selon les documentations, il semble que 15 soit adapté à notre situation.

— Le nombre de sorties $N_{classes}$ est le nombre de classes de la famille. On nommera $S_1, S_2, ...S_{N_{classes}}$ ces sorties. La sortie S_i correspond à la probabilité que le composant dont les caractéristiques ont été rentrées en entrée du réseau appartienne à la classe i.

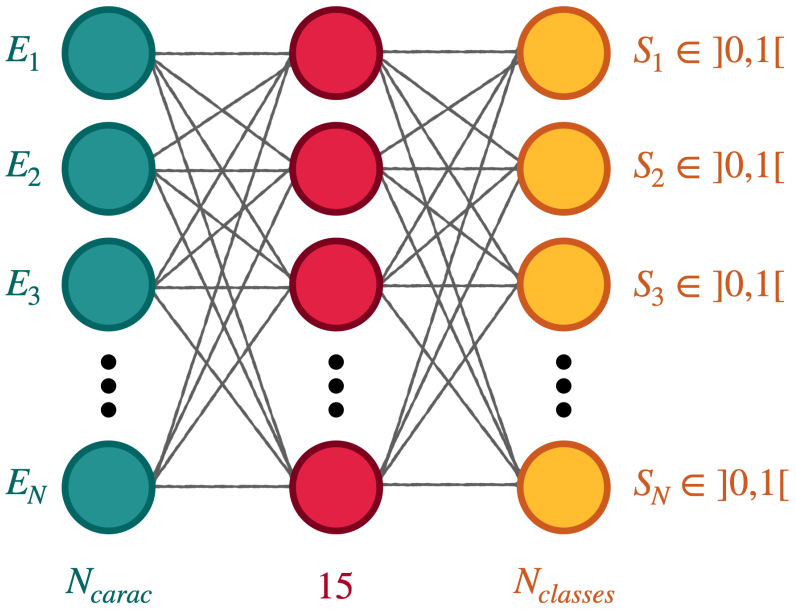

FIGURE 5 – Structure du réseau neuronal choisi

Pour trouver la classe d'un composant, on prendra donc la classe correspondant au maximum des probabilités de sortie. Ainsi, la classe du composant j sera :

$$ClasseA_j = k \text{ tel que } S_k = max_{i \in \{1,2,...,N_{classes}\}}\{S_i\}$$

Il arrive parfois que S_k soit assez faible (entre 0.3 et 0.5). Nous verrons alors que l'ingénieur de chez STMicroelectronics a un rôle à jouer sur ces composants-là spécifiquement, et que l'interface est pensée pour qu'il traite ces cas en priorité. On notera également :

$$\begin{cases} ClasseB_j = k \text{ tel que } S_k = max_{i \in \{1,2,...,N_{classes}\}/\{ClasseA_j\}}\{S_i\} \\ ClasseC_j = k \text{ tel que } S_k = max_{i \in \{1,2,...,N_{classes}\}/\{ClasseA_j, ClasseB_j\}}\{S_i\} \end{cases}$$

$ClasseB_j$ représente ainsi la deuxième classe la plus probable pour le composant j et $ClasseC_j$ la troisième. La définition de ces deux nouvelles classes nous permet d'élargir les composants les plus proches du composant j. On dira que tous les composants appartenant à $ClasseA_j$ appartiennent au même marché que le composant j, que ceux de la $ClasseB_j$ sont dans un marché très proche et ceux de la $ClasseC_j$ dans un marché proche. En termes de scoring, cela revient à attribuer une note de 100% de similarité aux composants de la $ClasseA_j$ par rapport au composant j, 66% de similarité pour la $ClasseB_j$ et 33% de similarité pour la $ClasseC_j$. Ces notions seront primordiales dans l'étude du marché autour d'un composant, principe développé en section 5.2.4.

4.5 Les limites et améliorations possibles de l'algorithme

L'une des limites principales est liée à la perte d'informations causée par la suppression de données au début de l'analyse. En effet, les valeurs *NaN* (*Not a Number*) ont été supprimées de la base de données, ce qui a réduit la quantité de données disponibles pour l'algorithme. Cela peut entraîner des erreurs ou des biais dans les résultats obtenus, car certaines données manquantes pourraient être importantes pour la classification des composants.

En termes d'améliorations possibles, une approche intéressante serait d'utiliser le *reinforcement learning* (basé sur un système de récompense et de malus) pour permettre à l'algorithme de prendre des décisions et d'adapter sa classification des composants semiconducteurs en fonction des tendances observées. Cette approche permettrait d'améliorer l'efficacité de l'algorithme et de réduire la dépendance du tri manuel pour classifier les composants. En outre, l'utilisation de techniques d'apprentissage en continu pourrait également aider à améliorer l'efficacité de l'algorithme en permettant à celui-ci d'apprendre de nouvelles tendances ou de nouveaux schémas à mesure que la base de données évolue au fil du temps.

5 Le livrable : une application web

5.1 Les motivations du développement de l'application

Afin de rendre l'algorithme utilisable par les ingénieurs de STMicroelectronics, nous avons dû développer une application qui est devenue par la suite le livrable de notre projet. Le parcours utilisateur de cette application est détaillé en section 5.2.

Le cahier des charges de l'application est le suivant :
- Permettre à l'utilisateur d'entrainer le modèle de manière intuitive, mais en lui laissant également une liberté totale,
- Permettre à l'utilisateur de visualiser l'évolution de la concurrence autour d'un composant ou d'une famille de composants.

L'ergonomie de l'application est un critère primordial lors de sa conception : les pages sont divisées en sections dont les rôles sont précisés et les graphiques sont rendus interactifs pour une meilleure appréciation des résultats obtenus.

Suite aux différentes réunions que nous avons eues avec le commanditaire, l'idée de pouvoir observer l'évolution des tendances de marché a émergé. L'idée est de pouvoir analyser, pour un composant donné, comment au fil du temps les autres constructeurs font évoluer leur offre dans le cluster du composant de STMicroelectronics. Cela permettrait, par exemple, de se rendre compte qu'un constructeur X essaie de s'approprier le marché pour un composant Y, ou au contraire que les constructeurs délaissent les marchés existants, ce qui témoignerait peut-être de l'émergence d'un nouveau marché sur lequel STMicroelectronics devrait se pencher. Ainsi, STMicroelectronics serait en mesure de réagir face à l'évolution de la concurrence.

Comme expliqué précédemment, l'application sera développée grâce à la bibliothèque Streamlit [2].

5.2 Le parcours utilisateur

Le parcours utilisateur, aussi appelé parcours client, désigne le parcours suivi par un client ou utilisateur au cours de ses différentes interactions avec une application. L'objectif du parcours utilisateur est d'identifier, de représenter et d'évaluer les parcours que les utilisateurs ou les usagers effectueraient lors de l'utilisation du service proposé par l'application développée.

Avant de vous présenter les captures d'écrans de l'application, détaillons les étapes principales de ce parcours client :

1. L'utilisateur lance l'application et se rend sur la page "Entrainer le modèle",
2. Il choisit une famille de composant ainsi qu'un composant de cette famille. À l'aide de ces connaissances sur les différents composants de cette famille, il va valider ou non certains résultats de l'algorithme, en modifiant les classes résultantes du réseau neuronal,
3. Il peut ensuite visualiser l'évolution du marché sur la page "Market tracker" et voir comment se situe STMicroelectronics par rapport à ses concurrents autour de composants ou de familles spécifiques.

Nous présentons ici le parcours d'un utilisateur, donc classiquement d'un ingénieur de STMicroelectronics.

5.2.1 La page d'accueil

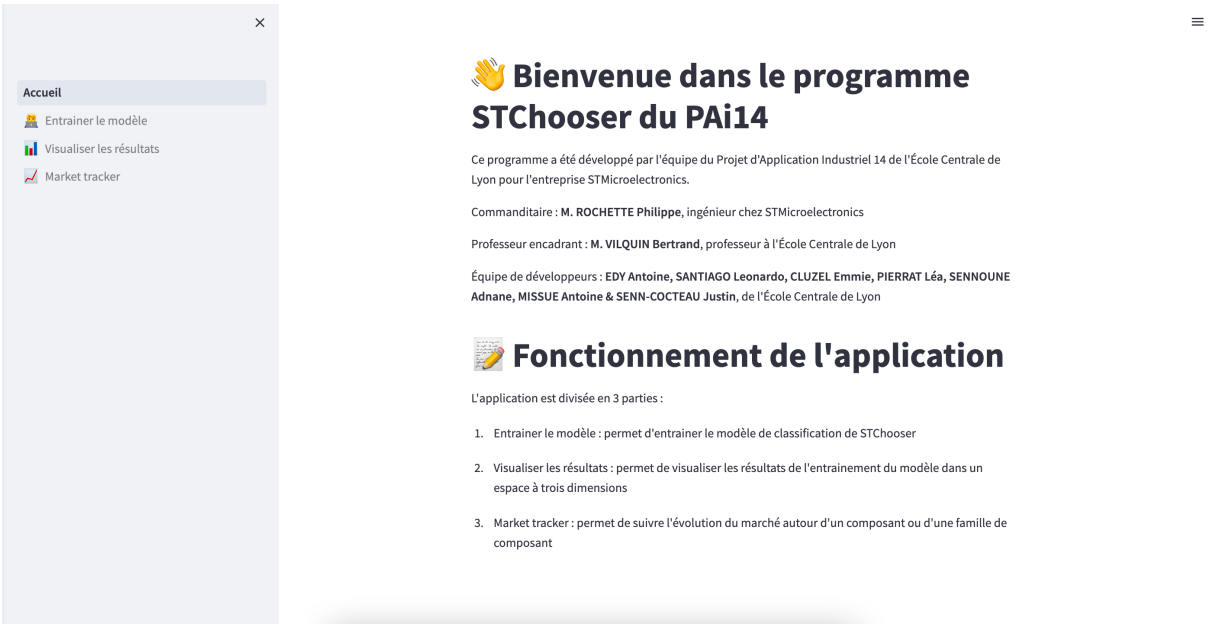

FIGURE 6 – Page d'accueil de l'application

L'utilisateur arrive sur la page d'accueil, qui contient un simple texte d'explication du fonctionnement de l'application.

5.2.2 La page d'entraînement du modèle

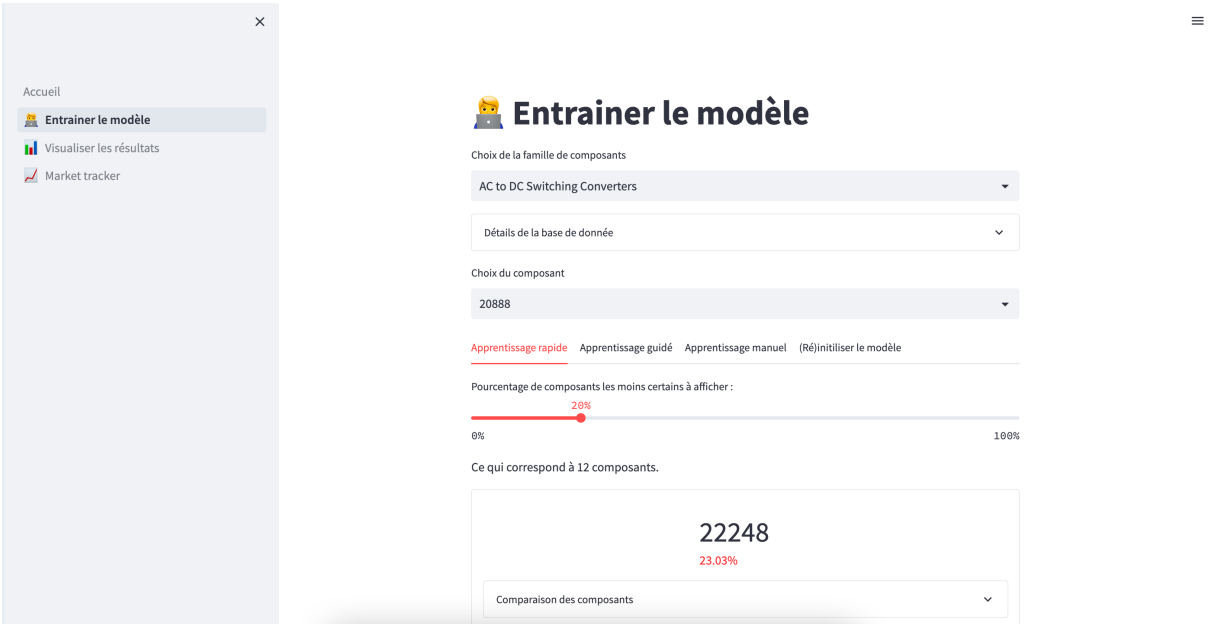

FIGURE 7 – Page d'entraînement du modèle

Ici la page d'entraînement du modèle, qui comporte plusieurs sections que nous allons détailler.

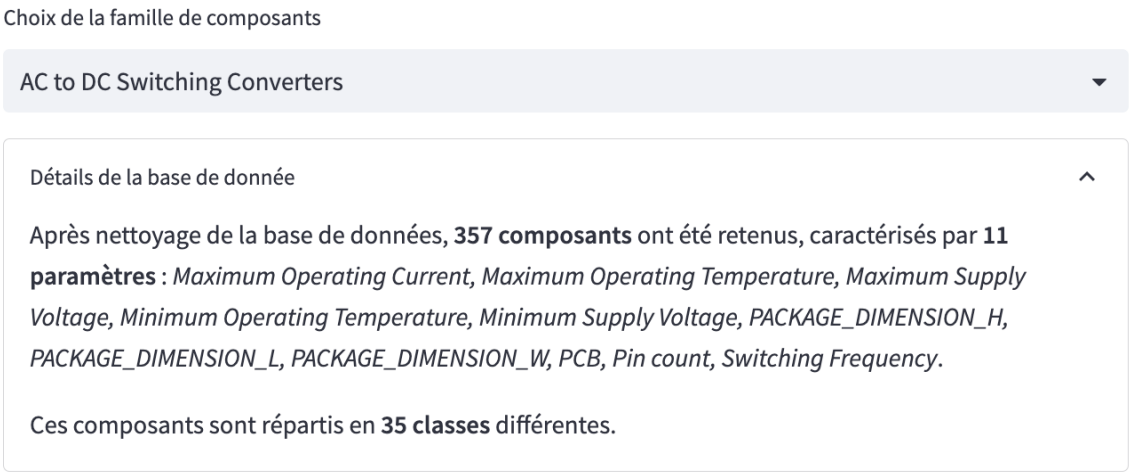

FIGURE 8 – Détails des familles des composants

Dans cette section (réduite de base - comme en figure 7 - que l'on peut développer pour afficher les informations - comme en figure 8), l'utilisateur accède à certaines informations importantes de la base de donnée : les paramètres pris en compte, le nombre de composants de la famille, le nombre de classes de composants.

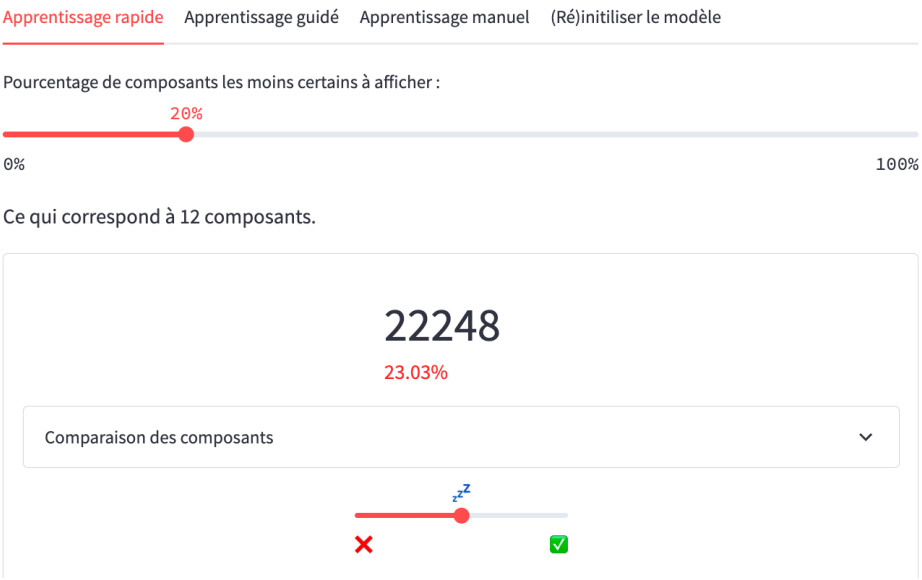

FIGURE 9 – La section dites d'"apprentissage rapide"

La première des quatre possibilités offertes par l'application pour aider l'utilisateur à affiner le modèle est "l'apprentissage rapide". L'utilisateur va pouvoir valider ou non les choix du modèle parmi un certain pourcentage des choix les moins certains, ici 20% - mais ajustable avec le *slider*.

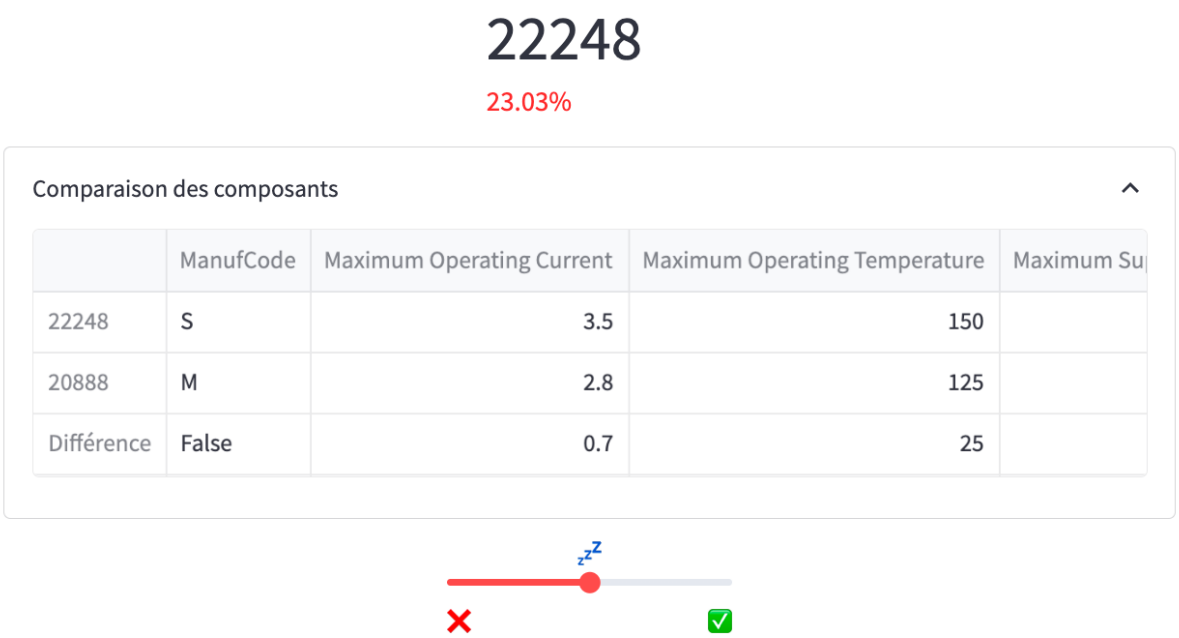

FIGURE 10 – Un tableau de comparaison des composants

L'utilisateur peut comparer les caractéristiques de chaque composant avec celui sélectionné précédemment en faisant dérouler la "comparaison des composants".

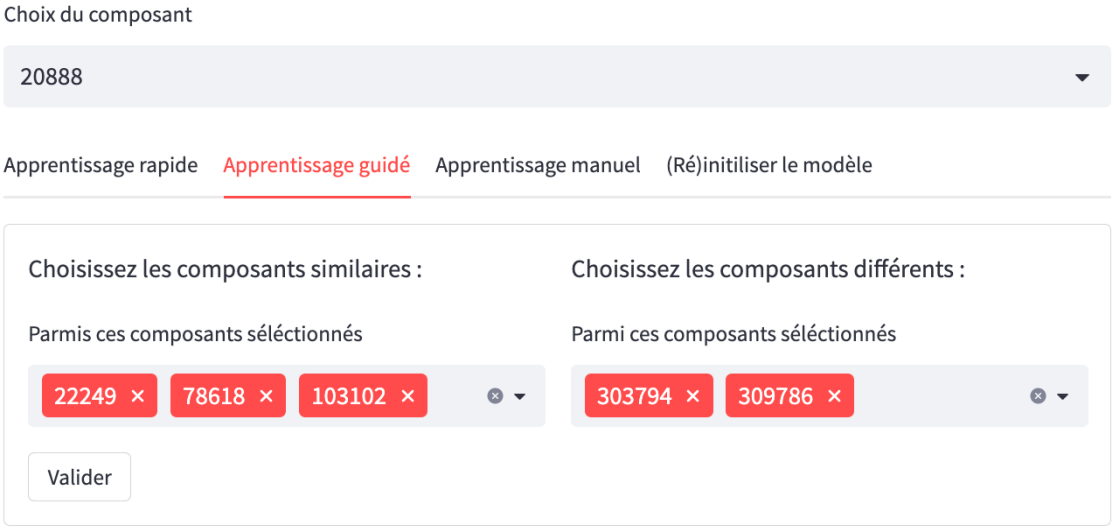

FIGURE 11 – La section dites d'"apprentissage guidé"

La partie "apprentissage guidé" permet à l'utilisateur de valider ou de désapprouver les choix du modèle : pour chaque composant, il pourra dire quels composants appartenant à sa classe se doit d'y appartenir (colonne de gauche) ou non (colonne de droite).

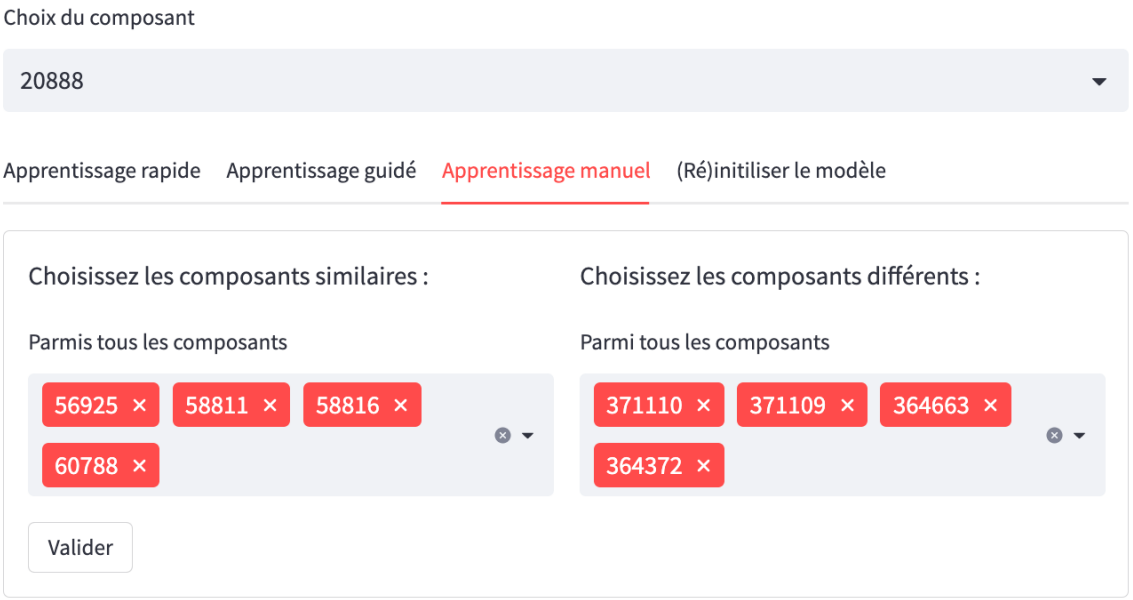

FIGURE 12 – La section dites d'"apprentissage manuel"

Cette partie répond au besoin de l'utilisateur à forcer deux composants à se situer dans la même classe. Ici, l'utilisateur (l'ingénieur de STMicroelectronics) peut forcer deux composants parmi tous à être dans la même classe ou à être dans des classes différentes.

FIGURE 13 – La section de (ré)initialisation

Cette dernière partie permet de réinitialiser les groupes au sein de la famille. Le réseau neuronal est donc initialisé avec un système de cluster classique construit par distances. Chaque famille doit être initialisée avec ce bouton-là également.

FIGURE 14 – La barre de chargement indique le temps des calculs de la mise à jour du réseau neuronal

Une fois l'une des méthodes choisies, l'utilisateur valide son choix avec le bouton "Valider" situé en bas de page. Le modèle prend donc en compte les modifications et se

modifie en fonction. Ce calcul reste court (moins d'une minute) pour la grande majorité des familles. De manière générale, plus une famille comporte de nombreux composants et plus le chargement sera long.

5.2.3 La page de visualisation des résultats

L'intérêt de cette page pour les ingénieurs de STMicroelectronics est limité. Elle nous a principalement servi à vérifier les premiers résultats de notre modèle.

FIGURE 15 – Visualisation des résultats dans un espace à 3 dimensions

Cette page permet alors de visualiser, dans un espace à trois dimensions, les différents composants.

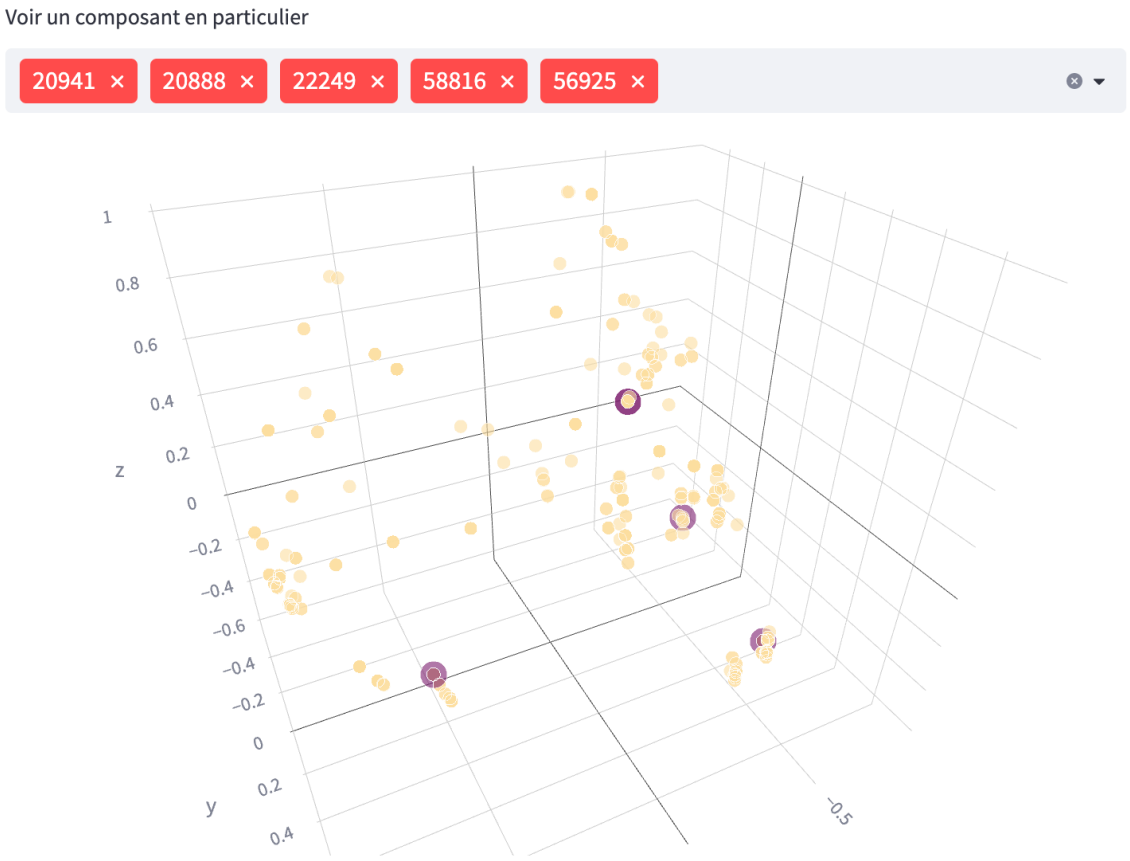

FIGURE 16 – Visualisation des résultats - highlight de certains composants selctionnés

L'utilisateur peut voir où se situe un ensemble de composant qu'il aura choisi. En passant sa souris sur un point, il y verra le nom du composant.

5.2.4 La page de visualisation temporelle de l'évolution du marché

Cette page est la finalité du projet : elle permet à l'utilisateur de suivre l'évolution de la concurrence autour d'un composant ou d'une famille de composants.

On notera que nous ne disposions que d'une base de donnée à un instant t. Ainsi, nous avons simulé les évolutions temporelles en effectuant une division de notre base de donnée et en ajoutant une partie de plus en plus importante. À l'instant 0, le premier cinquième des composants de la base de donnée sont traités, puis les deux premiers cinquièmes et ainsi de suite, jusqu'à arriver à la base complète.

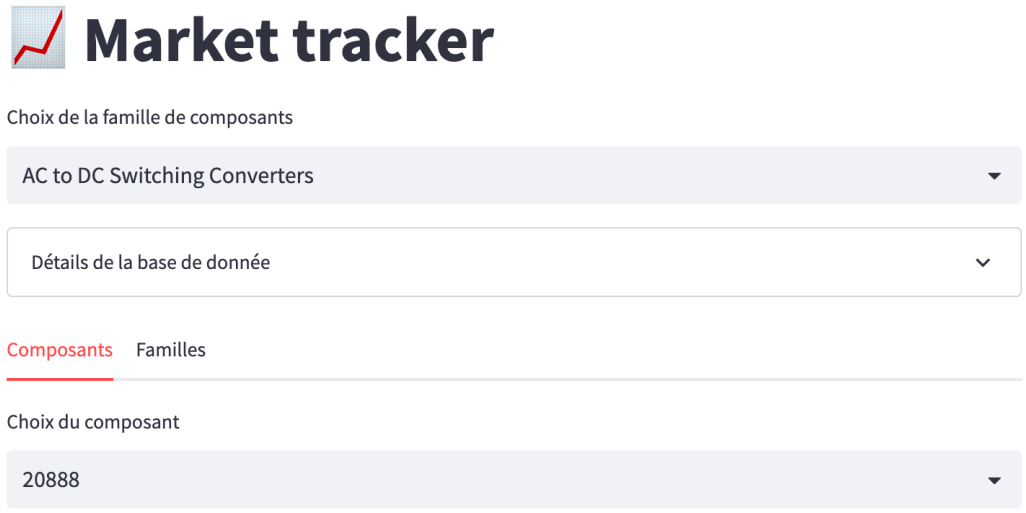

FIGURE 17 – Market tracker - haut de page

Ici le haut de cette page, reprenant la possibilité de choisir une famille de composant. Deux possibilités s'offrent ensuite à nous : l'étude du marché autour d'un composant ou pour une famille entière.

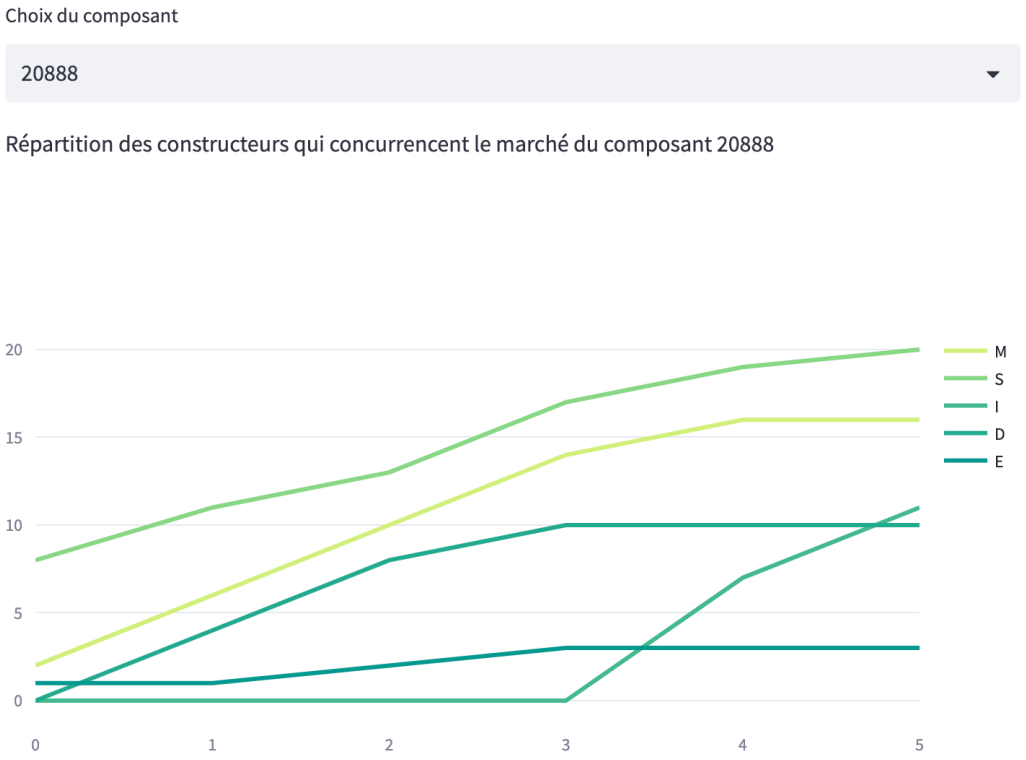

FIGURE 18 – Évolution du marché autour d'un composant

Cette première partie permet de visualiser l'évolution du marché autour d'un composant sélectionné.

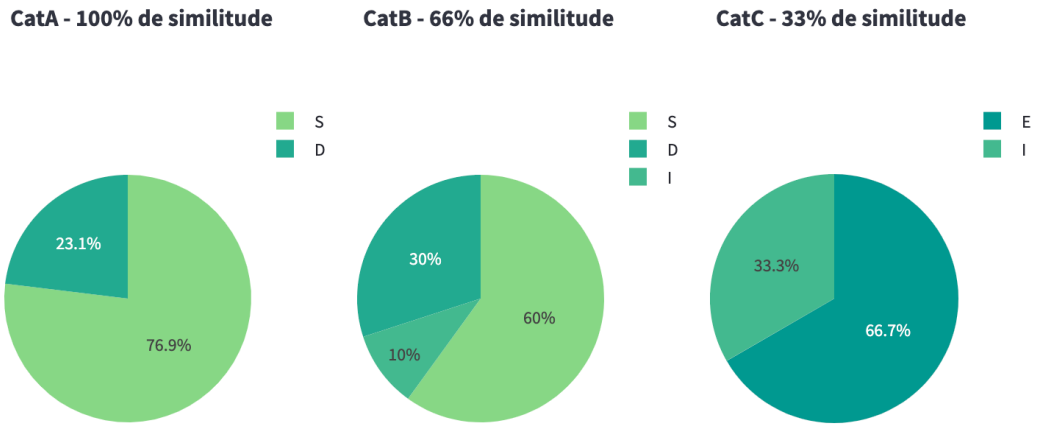

FIGURE 19 – Part des constructeurs dans les composants proches (appartenant à la même classe (catégorie A) ou aux classes voisines (catégories B et C) de celui sélectionné

Ces figures représentent les parts des constructeurs dans les composants proches (appartenant à la même classe (catégorie A) ou aux classes voisines (catégories B et C) de celui sélectionné.

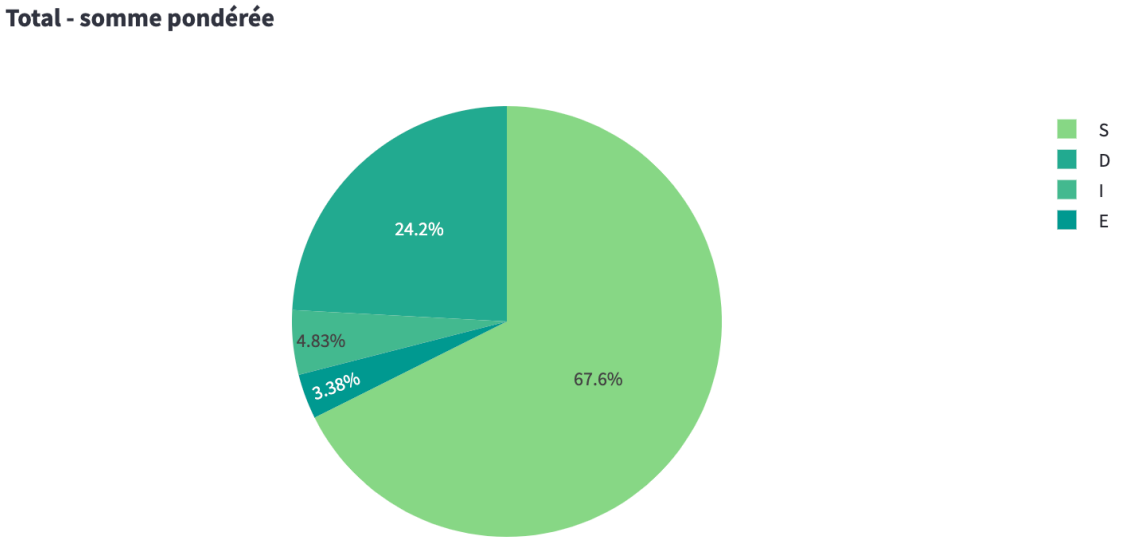

FIGURE 20 – La somme pondérée des catégories A, B et C

Cette figure est la somme pondérée des trois précédentes (figure 19) pondérée suivant la formule suivante :

$$Total = \frac{3 \cdot Cat_A + 2 \cdot Cat_B + 1 \cdot Cat_C}{6}$$

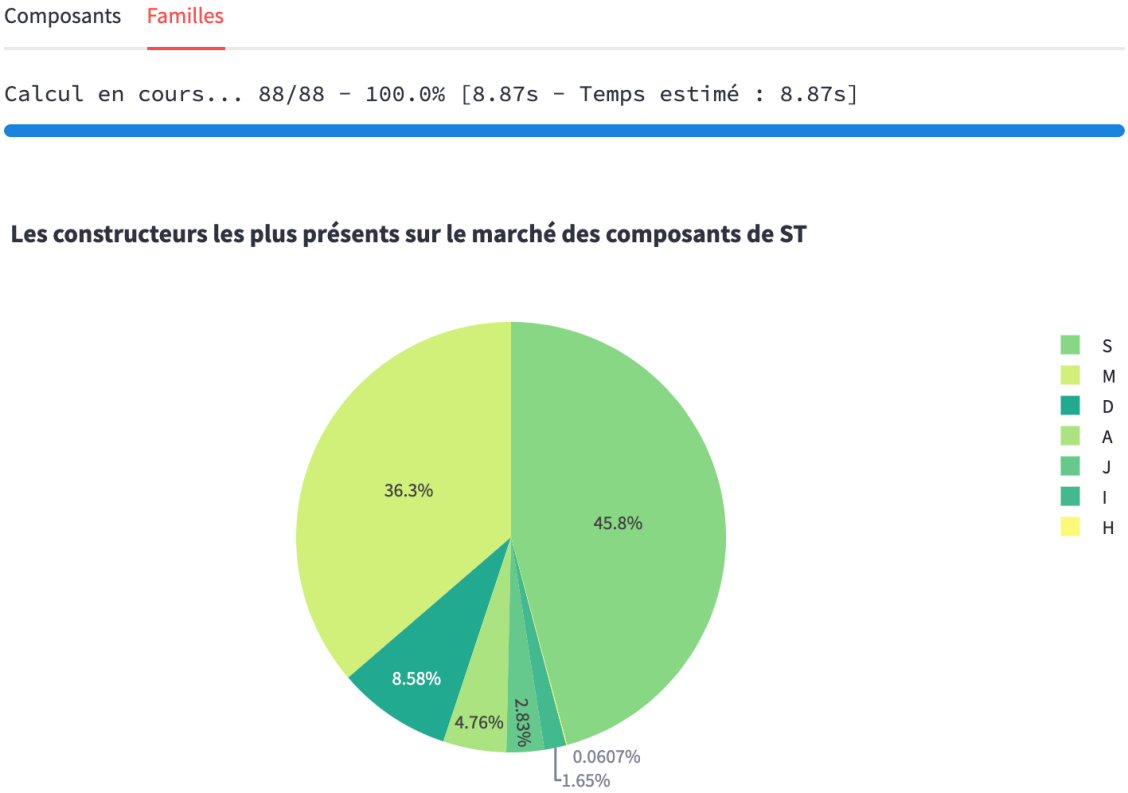

FIGURE 21 – Market tracker - résultat par famille

La deuxième partie de cette section est dédiée à l'étude du marché non plus autour d'un composant, mais autour de la famille complète. On retrouve, pour une famille donnée, la part des constructeurs qui possèdent des composants proches de ceux de STMicroelectronics.

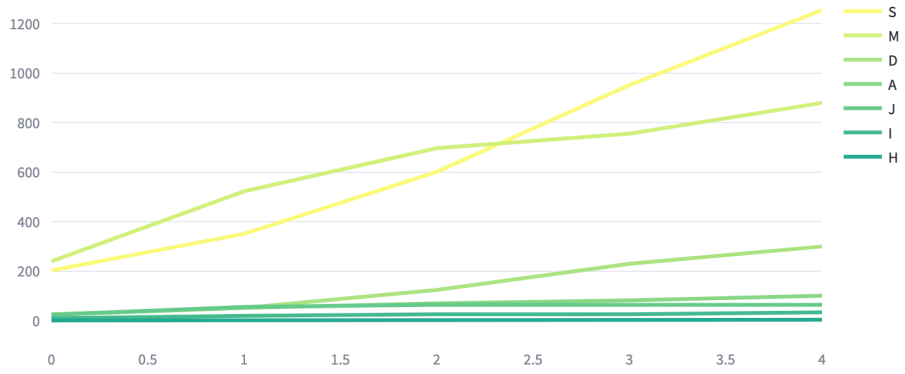

FIGURE 22 – Évolution temporelle de la part de marchés de différents constructeurs pour une famille donnée

On retrouve également l'évolution temporelle du marché, autour cette fois-ci d'une famille de composants.

6 Conclusion

Ce projet de développement d'une application web et d'un algorithme qui vise à identifier les concurrents directs de STMicroelectronics et à établir leur niveau de compétitivité nous a permis d'initier ce que l'on espère être une solution pour surmonter les défis imposés par la forte concurrence du marché des composants électroniques.

Nous pensons que ce projet a un avenir qui s'étend au-delà de notre collaboration avec l'entreprise qui a eu lieu le temps de quelques mois. Les échéances nous ont empêché de réaliser l'étape clef qui est celle du test de l'application par un groupe d'ingénieur de l'entreprise et la prise en compte de leurs différents retours. Ce projet représente ainsi une première étape vers une automatisation plus poussée de l'analyse comparative de produits, et nous espérons que les résultats obtenus seront utilisés par STMicroelectronics dans le futur.

7 Bibliographie

Références

[1] Scikit learn AUTHORS et COMMUNITY. *Scikit-learn documentation*. URL : https://scikit-learn.org/stable/user_guide.html.

[2] Streamlit CREATORS. *Streamlit documentation*. URL : https://docs.streamlit.io.

[3] The Linux FOUNDATION. *PyTorch documentation*. URL : https://pytorch.org/docs/stable/index.html.

[4] NumFocus INC. *Pandas documentation*. URL : https://pandas.pydata.org/docs/.

[5] Nicolas LABROCHE. *Apprentissage actif pour le clustering semi-supervisé*. URL : http://www.vincentlemaire-labs.fr/CluCo2014/cluco_labroche_2014.pdf. Atelier Clustering and Co-clustering (CluCo), EGC 2014.

[6] Kimia NADJAHI. *Openclassrooms : Implémentez votre premier réseau de neurones*. URL : https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5097666-tp-implementez-votre-premier-reseau-de-neurones-avec-kerass. (accessed : 01.09.2016).

[7] PLOTLY. *Dash documentation*. URL : https://dash.plotly.com.

[8] Nicolas RANGEON. *Découvrez les libairies Python pour la Data Science*. URL : https://openclassrooms.com/fr/courses/7771531-decouvrez-les-librairies-python-pour-la-data-science/7857178-creez-votre-premier-data-frame-avec-pandas. Nicolas Rangeon, Data scientist, instructor Computer engineer (Université de Technologie de Compiègne).