# Rapport de projet 4A:

*Etude de la maladie d'Alzheimer à l'aide d'intelligences artificielles*

Lopes Castanheira Marcelo - Peyron Calvin

# Questions préliminaires:

## Définition de Machine Learning :

Le Machine Learning est un champ d'étude utilisé en intelligence artificielle qui consiste à améliorer l'apprentissage des machines au fil du temps de manière autonome, en leur fournissant des données et des informations sous la forme d'observations et d'interactions réelles.

## Différence entre apprentissage supervisé et non-supervisé :

L'apprentissage supervisé fait appel à des données étiquetées ou annotées pour réaliser des prédictions, l'apprentissage non supervisé n'utilise pas d'étiquette. Donc en apprentissage supervisé l'algorithme apprend à associer des entrées à des sorties prédéfinies en utilisant des techniques d'optimisation pour minimiser l'erreur de prédiction, tandis que dans un apprentissage non supervisé l'algorithme apprend à découvrir des structures ou des relations dans les données.

## La différence entre classification et régression :

La différence principale entre la classification et la régression est que, la régression permet de prédire une quantité alors que la classification permet de prédire une catégorie. Les deux problèmes sont des apprentissages supervisés ou encore dit apprentissage automatique supervisée car dans les deux cas nous essayons de prédire une valeur discrète ou continue a partir de donné.

## Définition de clustering et différence avec la classification :

La méthode de clustering cherche à diviser les données en différents groupes ou clusters de sorte que les données dans le même cluster soient similaires entre elles. La différence entre le clustering et la classification est que le clustering va chercher des similitudes entre les données pour pouvoir les mettre dans un même groupe, alors que la classification prédit dans quel groupe appartient chaque donnée . De plus, la méthode du clustering rentre dans l'apprentissage non supervisé contrairement à la classification.

## Exemple de problème possible :

| Problème | Apprentissage supervisé ou non | Classification, régression ou clustering |
|---|---|---|
| Prédire l'âge d'une personne | Apprentissage supervisé | Régression |
| Assigner automatiquement un diagnostic à une personne (« malade » ou « pas malade ») | Apprentissage supervisé | Classification |
| Identifier des groupes de patients similaires | Apprentissage non supervisé | clustering |
| Prédire le score d'une personne à une questionnaire de cognition (entre 0 et 20) | Apprentissage supervisé | régression |
| Prédire le diagnostic futur d'une personne parmi 3 possibilités | Apprentissage supervisé | Classification |
| Prédire l'évolution du volume du petite partie du cerveau | Apprentissage supervisé | Régression |

## Mise en place d'un programme de détection de la maladie d'Alzheimer chez un patient :

Pour mettre en place ce programme, nous utiliserons deux méthodes d'apprentissage supervisé. La première utilisera la classification, tandis que la seconde utilisera la régression pour évaluer le volume de l'hippocampe du patient, qui est un indicateur important pour diagnostiquer la maladie d'Alzheimer.

Pour cela, nous utiliserons une base de données provenant de l'ADNI et qui représente un fichier Excel contenant plus de 12 000 patients et dont pour chaque patient nous avons plus de 1900 informations qui représentent son âge, son sexe, son diagnostic, ces cognitifs tests…

Afin de lire les données de la base de données, nous utiliserons la librairie pandas. Celle-ci nous permet de lire les fichiers CSV, et de modifier certaines valeurs à la lecture. Ce qui nous permet de binariser les résultats des séances (1 pour le patient est malade et 0 pour le patient n'a pas Alzheimer). Les autres paramètres modifiés sont PTETHCAT, PTRACCAT et PTGENDER qui correspondent respectivement à l'ethnicité, la "race" et le sexe du patient. Ces données se doivent d'être modifiées afin que l'IA puisse les comprendre et les traiter. Pour cela, nous avons attribué une valeur numérique à chaque valeur différente d'une colonne ce qui permet d'avoir des données sous forme d'entier.

Finalement, les données communes aux deux types de programmes utilisés sont les résultats de tests cognitifs, l'âge, le genre, le taux d'éducation, l'ethnicité, la "race" ainsi que le diagnostic.

Le diagnostic est utilisé pour entraîner les modèles produits pour la classification tandis que pour la régression il est utilisé comme valeur d'entrée. Les autres données quant à elles ont été sélectionnées car semblaient être celles avec le plus fort impact pour le développement de la maladie d'Alzheimer.

Pour la régression, d’autres données sont utilisées afin de déterminer le volume de l’hippocampe, nous avons choisi d’utiliser le volume total du cerveau, le volume de l'entorhinal, des ventricules, du gyrus fusiforme, du gyrus temporal moyen, le volume intracrânien et le volume de base de l’hippocampe lors du premier test.

Pour classer nos données, nous utiliserons plusieurs modèles de régression/classification et nous les comparerons de façon appropriée. Pour chaque modèle, nous mesurerons le temps nécessaire à l’algorithme pour entraîner et tester les données et évaluer les performances de chaque modèle avec différentes métriques, ce qui nous permettra d'interpréter l’efficacité de chaque modèle.

Afin d’améliorer la vitesse des programmes, nous avons décidé au préalable de créer de nouveaux fichiers csv ne contenant seulement les données dont nous avions besoin. Dans cette optique, nous avons obtenu des fichiers considérablement moins volumineux. car sur plus de 1900 informations à utiliser nous en avons utilisés seulement 7 (classification) et 15 (régression). Pour chaque données, nous vérifions que la valeur existe (qu’elle ne soit pas à NaN), et si elle n’existe pas nous supprimons la ligne correspondante. Finalement, il ne reste que 3678 lignes et 2577 lignes.

# Point commun aux deux programmes:

## Le principe de K-fold :

Le K-fold est une méthode de validation croisée, grâce à elle nous pouvons faciliter l'obtention d'un modèle performant. Elle consiste à diviser les données d'entraînements en K plis égaux. Parmi ces plis, nous utiliserons K-1 pour entraîner notre modèle et le dernier servira à le tester. Toutes les combinaisons sont testées et la meilleure est celle choisie pour être notre modèle. L'utilisation de cette méthode dans notre cas a judicieusement été faite car en effet, lorsque les données d'entraînement sont limitées elle permet d'utiliser l'entièreté des données pour entraîner et tester le modèle.

De plus la méthode K-fold peut être utilisée en effectuant la moyenne des scores de performances de chaque split de modèles pour savoir si le type de modèle utilisé (par exemple SVC, SVR ou encore régression logistique) permet de satisfaire notre problème (que ça soit de la régression ou de la classification).

# Etude du volume de l'hippocampe (Calvin):

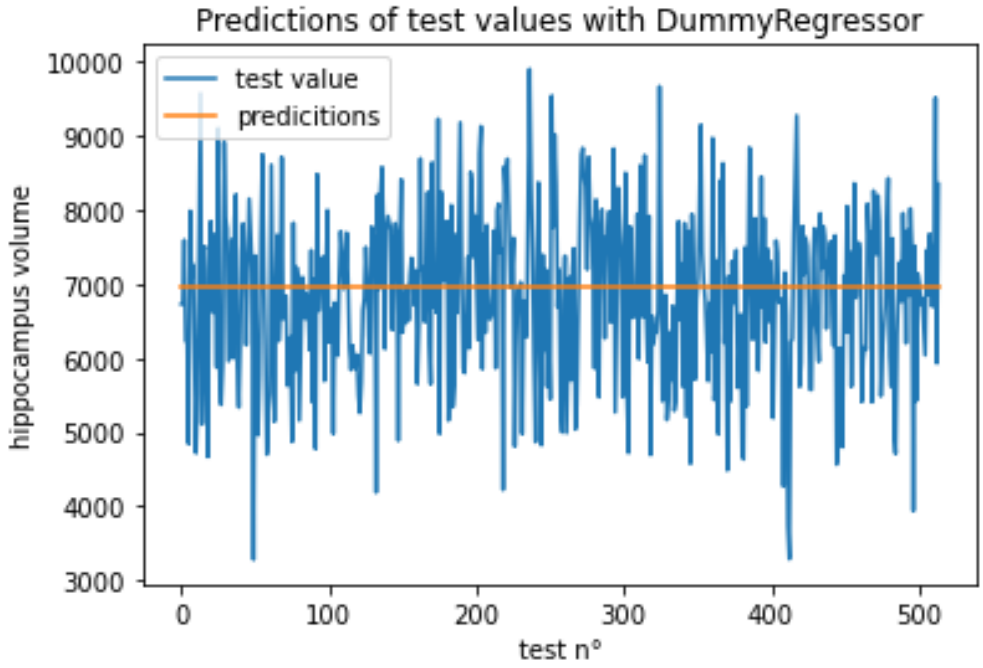
## Définition du problème :

Le volume de l'hippocampe étant une valeur à approcher, le choix de la régression est naturel. Afin de prédire cette valeur, nous avons choisi plusieurs méthodes de régression, les deux méthodes étudiées sont une machine à support de vecteur dédiée à la régression et une tweedie régression. Ces deux méthodes seront comparées à une dummy régression, celle-ci nous permettra de mieux comprendre nos résultats et ainsi produire une IA plus qualitative. Nous avons choisi d'utiliser une machine à support de vecteur car celles-ci sont très communément utilisées dans le cadre de production d'IA, que ce soit à des fins de régression ou de classification. A l'opposé, le choix d'une tweedie régression est plus réfléchie et aboutie, en effet elle a une densité qui suit une courbe exponentielle et est particulièrement précise si ses entrées sont aux alentours de 0. Notre situation convient parfaitement à l'utilisation d'une tweedie régression car au sein de nos données en entrée, nous retrouvons l'état du patient. Or nous avons préalablement binarisés ces valeurs et avons choisi après réflexion que les non-malades présents en plus grande quantité seraient associées à la valeur 0.

## Modèle de base servant de comparaison :

Le modèle que nous utiliserons et qui nous servira de comparaison pour noter l'efficacité des autres modèles est le dummy regressor qui est présent dans la librairie sklearn en tant que DummyRegressor. Celui-ci a pour principe d'utiliser des règles de prédiction très simples. De cette manière, il est aisé par exemple de ressortir une valeur moyenne du volume de l'hippocampe en donnant la valeur "mean" au paramètre strategy de DummyRegressor. Nous comprenons donc que celui-ci ne doit pas être utilisé pour réellement prédire la valeur de l'hippocampe d'un patient car il produira un résultat trop souvent éloigné de la réalité.

Voici un exemple de ce que ressort un dummy regressor avec nos données en mettant le paramètre strategy à “median”:




Nous pouvons observer que la valeur médiane du volume de l' hippocampe est aux alentours de 7000 unités. Les performances de ce modèle sont les suivantes:

```
Average mean absolute error percentage dummy: 0.15047177923567537.
```

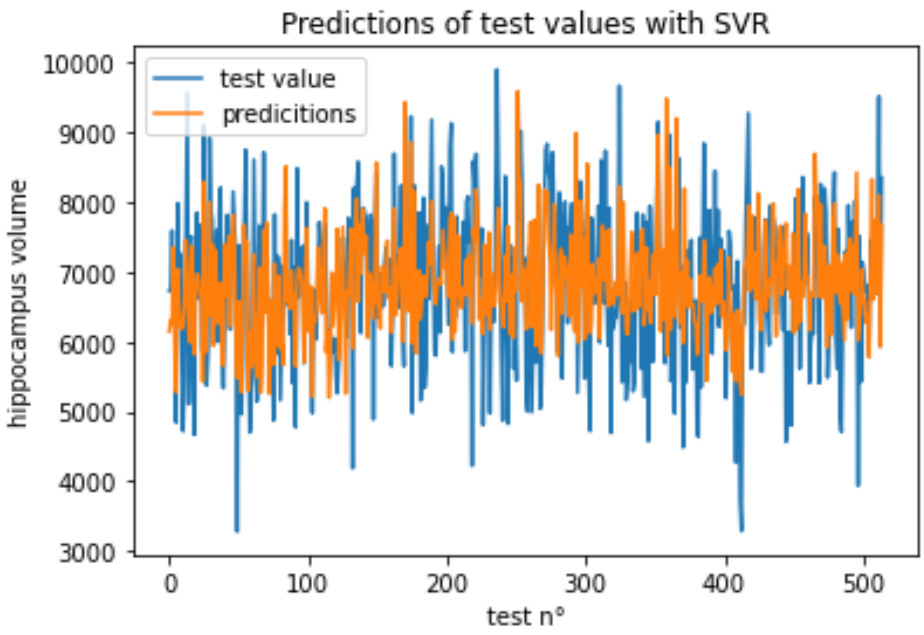
```
Pour un temps de: 0.0019989013671875
```

En conclusion ce modèle est d’une rapidité énorme car sa valeur de sortie est constante. Cependant le pourcentage d’erreur moyen en valeur absolue est aux alentours de 15%, ce qui signifie que la précision est de seulement 85%, le résultat n’est donc pas suffisant et cela se comprend par le fait que ce modèle n’est pas fait pour être efficace.

## Un des modèles les plus couramment utilisés :

Le premier modèle que nous étudions est la machine à support de vecteur (SVR) qui est présente dans la librairie sklearn sous le nom de SVR. Le principe général des modèles de régression linéaire est de minimiser la somme des erreurs en quadrature. Le modèle SVR ne déroge pas à la règle, cependant il permet de gagner en flexibilité en définissant une marge d'erreur dite "acceptable". Celle-ci permet au SVR de ressortir un hyperplan approprié aux données fournies. Cette marge d'erreur est un paramètre à donner en entrée de la fonction SVR, ce paramètre est appelé epsilon. Un autre paramètre à donner à la fonction est C, c'est un paramètre de régularisation qui permet de prendre en compte les points qui sont en dehors de de la marge d'erreur "acceptable", plus le paramètre C sera important plus l'impact des points le sera. Il est bon de noter que le paramètre C prend une valeur strictement supérieure à 0. Un autre paramètre important est kernel, il représente le noyau de la matrice utilisée dans le SVR, par défaut celui-ci est imposé à rbf cependant dans nos test nous utiliserons aussi le mode poly pour comparer les deux noyaux. Le dernier paramètre que nous modifierons est tol qui représente la tolérance pour le critère d'arrêt, il indique à sklearn d'arrêter de rechercher un minimum dès qu'une certaine tolérance est atteinte. D'autres paramètres existent cependant nous avons choisis de ne pas nous en servir car les principaux sont ceux cités plus haut.

En testant de nombreuses valeurs, nous avons conclus que les valeurs $C = 1000$, $epsilon = 10^{-8}$ et $tol = 10^{-1}$ étaient suffisantes et que pour obtenir de meilleurs résultats il nous faudrait modifier les autres paramètres. En utilisant seulement ces paramètres là, nous obtenons les résultats suivants:




Nous pouvons observer une nette différence comparé au dummy regressor. En effet, il ne s'agit plus ici d'estimer une valeur constante mais d'approcher la valeur réelle. La précision du modèle est meilleure:

```
Average mean absolute error percentage SVR with polynomial kernel: 0.10973363024452112. Pour un temps moyen de: 0.13238391876220704
```

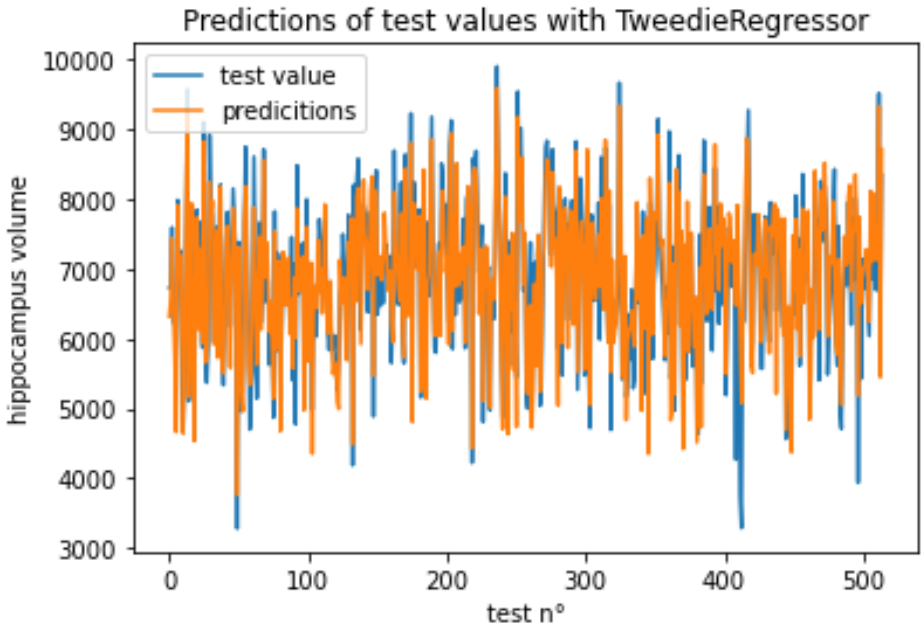
```
Average mean absolute error percentage SVR with rbf kernel: 0.10575866157587085. Pour un temps moyen de: 0.15835604667663575
```

L'erreur moyenne en valeur absolue étant approximativement de 11% qu'importe le noyau utilisé, nous avons une précision de 89%. C'est-à-dire que nous avons amélioré nos résultats par rapport au dummy regressor. Nous avons gagné 3% qui sont non négligeables à ce niveau. Cependant le revers de la médaille est que nous avons perdu en efficacité temporelle. En effet le temps mis pour ce modèle est près de 75 fois supérieur à celui pour le dummy regressor mais cela se comprend par la complexité du SVR et l'énorme simplicité du dummy regressor.

En comparant les résultats des deux noyaux, nous concluons qu'un noyau polynomial est moins précis dans notre cas, une déduction possible est que nos données ne sont pas reliées par une relation polynomiale. Cependant, celui-ci est plus rapide.

## Un modèle pensé pour réussir :

Le dernier modèle est le tweedie regressor qui par la logique utilisée correspond à notre problème. De nombreux paramètres sont changeables dans la fonction donnée par la librairie sklearn, cependant nous n'en modifieront que deux qui sont max_iter représentant le nombre maximum d'itérations pour le solveur et tol semblable au paramètre homonyme du SVR. Après de nombreux test, nous nous sommes rendu compte qu'augmenter max_iter ne donnait plus de résultats suffisant, nous nous sommes donc arrêtés à $max\_iter = 10000$. La tolérance est elle mise à 1e-2. Nous obtenons les résultats suivants:




Nous pouvons observer une nette amélioration par rapport aux deux modèles précédents. En effet cela se confirme par l'erreur moyenne absolue qui est d'à peu près 5%. Soit 95% de précision, ce qui représente une grande amélioration par rapport aux deux modèles précédents en les surperformant de 6% et 9%. Cependant, tout comme pour le SVR, cette amélioration qualitative de la précision se fait au détriment de la temporalité. Le temps mis pour ce modèle est plus de deux fois supérieur au temps mis pour le SVR.

```
Average mean absolute error percentage tweedie: 0.05387409803174453. Pour un temps moyen de: 0.3980122089385986
```

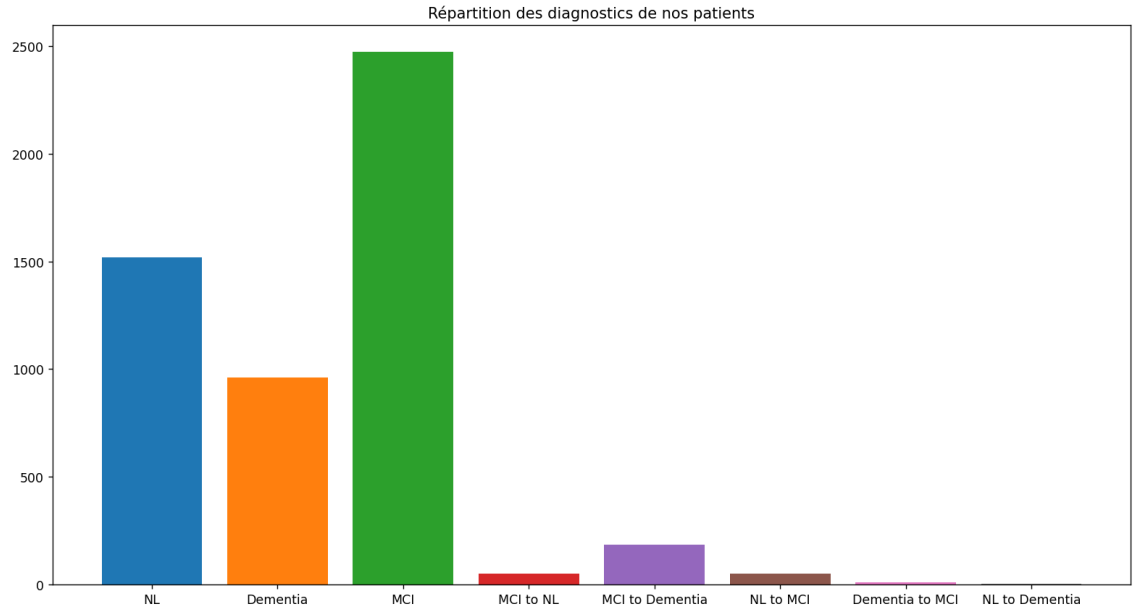
Pour conclure, dans notre cas, en considérant les données qui nous sont offertes et l'objectif, il est préférable d'utiliser le tweedie regressor. En effet, bien que plus lent en moyenne que les deux autres modèles étudiés, son efficacité est bien meilleure. De plus, nos données sont suffisamment restreintes pour que le problème de temporalité ne soit que mineur. Cependant, dans le cadre d'une base de données beaucoup plus conséquente, cela pourrait poser problème.

# Etude de la classification des patients (Marcelo):

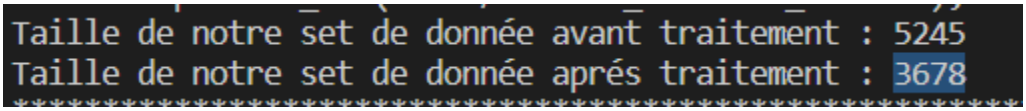
## Analyse et Traitement des données

Avant de commencer à créer nos modèles nous devons traiter les données de notre base de données afin que le modèle puisse mieux s'entraîner et qu'il ne soit pas biaisée par certaines données.

Pour cela, nous avons créé un histogramme avec matplotlib qui représente les différents diagnostics qu'ont pu recevoir les patients ainsi que la répartition des diagnostics :




Sur cet histogramme, nous pouvons voir que nous avons en majorité des patients avec des troubles cognitifs légers (MCI), ce qui aura son importance lorsque nous regarderons la matrice de confusion. Nous avons choisi d'ignorer les valeurs avec comme diagnostic "NL" (qui signifie "normal" ou "sain") et "MCI to NL" afin d'avoir une meilleure répartition des personnes ayant une démence et des personnes n'en ayant pas. Il est cependant à noter qu'en adoptant cette démarche, nous serons amenés à subir des pertes considérables de données, environ 30%, ce qui rendra plus complexe la tâche d'entraînement de nos modèles.




Maintenant que nos données sont traitées,nous pouvons désormais procéder à l'évaluation de différents types de modèles afin de sélectionner celui qui s'avérera le plus approprié pour atteindre les objectifs souhaités.

## Sélection de nos modèles :

Le premier modèle que nous testerons est celui du modèle de régression logistique qui est un des modèles de classification les plus simples et efficaces. Pour cela, nous utiliserons la méthode LogisticRegression de sklearn.linear_model, ou nous laisserons pour l'instant la majorité des options par défaut, sauf pour certains paramètres dont :

- Le paramètre solver qui correspond à l'algorithme utilisé dans le problème d'optimisation et dont dans notre cas nous mettrons "liblinear" qui selon la documentation de scikit learn, représente un bon choix pour les petits ensembles de données.

- Le paramètre max_iter qui correspond au nombre maximal d'itérations nécessaires pour que les solveurs convergent, nous avons décidé de fixer sa valeur à 10 000 car après plusieurs tests nous avons estimé que cette valeur se rapprochait de la limite produisant un résultat optimal.

Le second modèle que nous testerons est le modèle de machines à vecteurs de support (SVC). Pour utiliser ce modèle, nous utiliserons la méthode SVC de sklearn.svm. Pour les paramètres de la méthode, nous les laisserons par défaut à l'exception de probability que nous mettons à true, car il permet d'utiliser la méthode predict_proba qui nous sera utile plus tard. Il est important de savoir que nous n'avons pas modifié le paramètre kernel, et donc par défaut le type de noyau utilisé pour modéliser les données est "rbf".

Et enfin, le dernier modèle que nous utiliserons est le modèle des k plus proches, voisins. Il sera important de déterminer le nombre de voisins K, car si cette valeur est trop haute, nous pouvons obtenir de l'overfitting. Ce qui signifie que notre modèle ne sera efficace uniquement sur nos données d'entraînement et donnera une mauvaise prédiction sur des nouvelles données. Par défaut, la valeur de K est 5.

## Choix de nos critères de performance :

Pour évaluer ces modèles, nous utiliserons 4 types de critère qui sont:

- Le temps d'entraînement
- le pourcentage de précision qui correspond à la précision d'un modèle de classification et qui utilise la formule suivante :

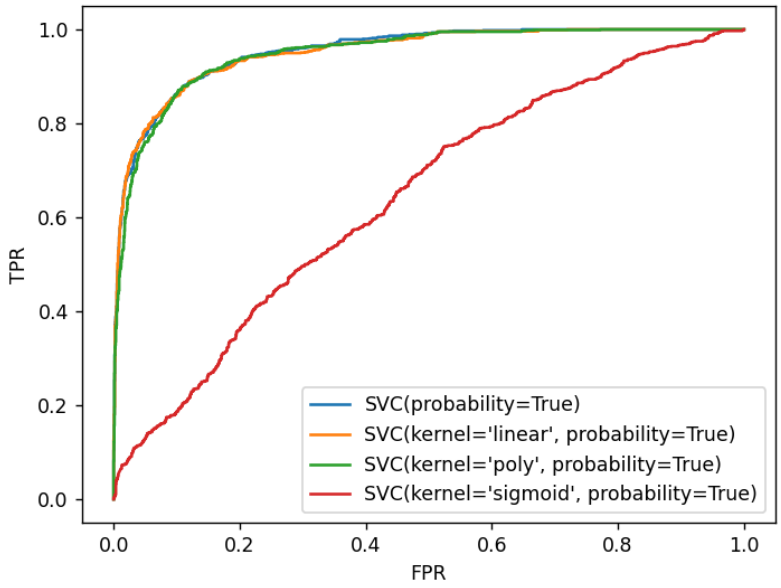
$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

  avec y les valeurs réelles et ŷ les valeurs prédites

- La matrice de confusion qui nous permettra de visualiser les erreurs commises par le modèle lors de la prédiction des classes
- La courbe ROC qui est un graphique qui permet de visualiser les performances d'un modèle en comparant le taux de vrais positifs (True Positive Rate) à celui de faux positifs (False Positive Rate)

Le choix du paramètre du noyau pour le modèle SVC est un bon exemple qui permet de montrer en quoi les critères de performance permettent de choisir les paramètres de notre modèle.

Effectivement si nous commençons par effectuer une courbe ROC avec les différents type de noyau disponible nous avons ceci :




En observant cette courbe, nous réalisons que le modèle qui utilise la sigmoid comme noyau se rapproche plus de l'aléatoire que d'une réelle classification de nos données. Et donc cela nous permet de retirer le noyau sigmoid des choix de paramètre pour le modèle SVC.

De plus, si nous restons sur ce test, seulement avec la courbe ROC, nous pouvons penser que le noyau linéaire et le noyau polynomial sont quasiment identique, cependant lorsque nous regardons la matrice de confusion :

```
**********************************************************************
Modele : SVC(kernel='linear', probability=True)
Temps d'entrainement pour le modèle : 0.047004938125610352
Pécision du modèle : 0.8985200845665962
Matrice de confusion :
[[2149  126]
 [ 210  826]]
l'aire sous la courbe ROC est de : 0.9521451907166193
**********************************************************************
Modele : SVC(kernel='poly', probability=True)
Temps d'entrainement pour le modèle : 0.009001255035400391
Pécision du modèle : 0.8021745696164301
Matrice de confusion :
[[2264   11]
 [ 644  392]]
l'aire sous la courbe ROC est de : 0.9499828164113878
**********************************************************************
```

Nous remarquons que le modèle avec un noyau linéaire a plus de faux négatifs que celui avec un noyau polynomial. Cela signifie qu'il y a une probabilité plus grande de ne pas détecter la présence de symptômes de la maladie d'Alzheimer chez un individu.

A l'inverse, le modèle avec un noyau polynomial a plus de faux positifs que le premier modèle. Ce qui signifie que le modèle polynomial a plus de chance de donner un diagnostic positif alors que cela est faux.

Il est donc préférable d'utiliser le second modèle car s'il indique qu'une personne a la maladie alors qu'elle ne l'a pas, elle sera suivie par des médecins qui pourront s'en rendre compte, alors qu'une personne qui a la maladie mais qui est indiquée comme n'ayant pas les symptômes ne sera pas prise en charge.

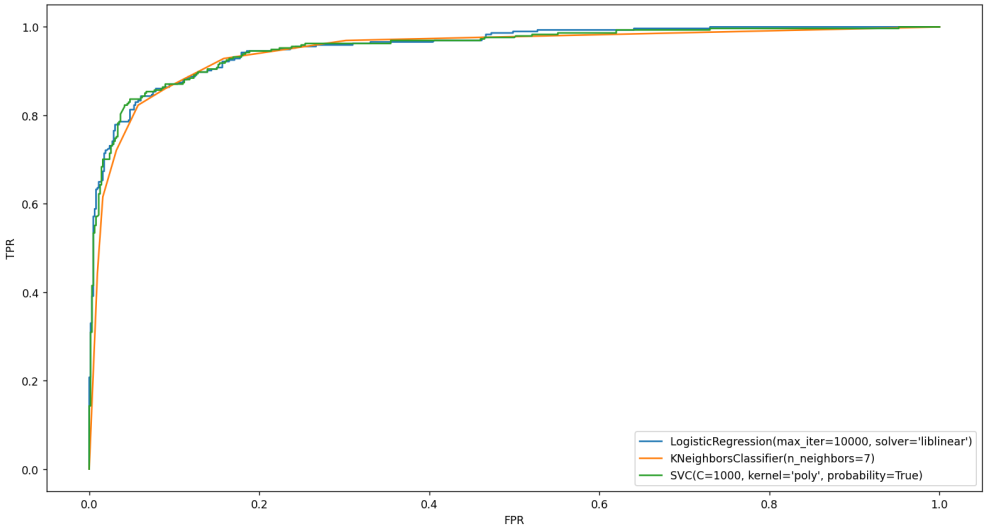
Et donc cet exemple nous montre l'importance de la matrice de confusion lors du choix de nos modèles.

## Choix final de nos modèles :

Après avoir effectué plusieurs tests afin de définir les paramètre les plus pertinents, voici les différents paramètres choisis pour chaque méthode :

```
tabModel = []
tabModel.append(LogisticRegression(solver="liblinear", max_iter=10000))
tabModel.append(KNeighborsClassifier(7))
tabModel.append(SVC(probability=True,kernel="poly",C=1000))
```

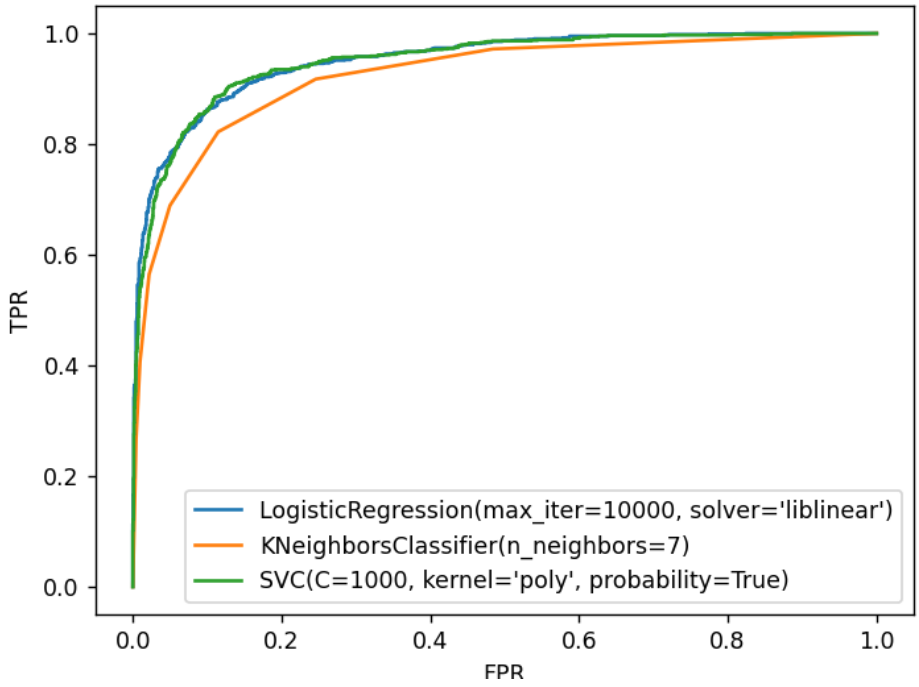
Ce qui nous donne ceci comme courbe ROC :




Et voici les autres critères pour chaque modèle :

```
////////////////////////////////////////////////////////////
Utilisation du KFold avec K = 5
**************************************************************************
Modele : LogisticRegression(max_iter=10000, solver='liblinear')
Temps d'entrainement pour le modèle : 0.007324075698852539
score du model pour le train :  0.8996739777965805
score du model :  0.8980468056787932
l'aire sous la courbe ROC est de :  0.9516220570248726
Moyenne des matrices de confusion pour le test:
[[478.2 28.0]
 [47.0 182.4]]
**************************************************************************
Modele : KNeighborsClassifier(n_neighbors=7)
Temps d'entrainement pour le modèle : 0.0051997661590576175
score du model pour le train :  0.9255030487487967
score du model :  0.9018574386276249
l'aire sous la courbe ROC est de :  0.9419454888161954
Moyenne des matrices de confusion pour le test:
[[480.0 26.2]
 [46.0 183.4]]
**************************************************************************
Modele : SVC(C=1000, kernel='poly', probability=True)
Temps d'entrainement pour le modèle : 1.1662020683288574
score du model pour le train :  0.9022567231973552
score du model :  0.8953283052351375
l'aire sous la courbe ROC est de :  0.9468068349543188
Moyenne des matrices de confusion pour le test:
[[480.2 26.0]
 [51.0 178.4]]
PS C:\Users\manso\Desktop\cours\Projet 4A\code>
```

Nous pouvons observer dans un premier temps que certaines courbes ROC sont saccadées au lieu d'être lisse ce qui s'explique par le fait que nous avons seulement 25% de nos données qui sont testé ce qui correspond à 920 valeurs. Nous pouvons rendre cette courbe plus linéaire en augmenter le nombre de valeur à mettre dans la partie train, ce qui nous donne cette courbe ROC :




Cependant, nous avons moins de données pour la partie train et donc plus de chance d'avoir de l'overfitting. Ce qui explique en partie le fait que la courbe du modèle KNN soit moins bonne. Ce modèle est moins stable que les autres, il n'est donc pas pertinent de le choisir.

Ainsi, le modèle de régression logistique semble être un bon choix car bien plus rapide que le SVC et le KNN. De plus, nous portons de l'importance au nombre de vrais positifs et lorsque nous regardons la matrice de confusion, le modèle de régression logistique a plus de vrais positifs que le SVC.

Cependant, il ne faut pas oublier que comme montré au début de cette étude nous n'avons pas beaucoup de données et donc il est possible que certains modèles fassent de l'overfitting, si nous avions plus de données, il est fort possible que les résultats obtenus ne soient pas les mêmes.

# Conclusion générale:

En résumé, notre étude montre qu'il est possible d'utiliser des techniques de machine learning pour mesurer le volume de l'hippocampe et pour détecter la maladie d'Alzheimer chez les patients. Les méthodes de régression ont permis de donner une estimation assez précise du volume de l'hippocampe même si plusieurs voies d'améliorations sont disponibles. Une IA donnant de telles performances ne pourrait pas être utilisée dans le domaine médical, en effet, les prédictions bien que bonnes en moyenne possèdent une erreur moyenne de 5% ce qui n'est absolument pas négligeable dans le cadre de la santé, surtout quand cela met des vies en jeu. Pour la méthode de classification des k plus proche voisin à permis de détecter les patients atteints de la maladie d'Alzheimer avec une précision d'environ 90% ce qui est assez bas pour une utilisation dans le domaine médical. Les principales améliorations que nous pourrons effectuer sont de tester nos modèles avec une plus grande base de données ainsi que de continuer à tester de nouveaux modèles, de plus il est aussi possible d'améliorer les résultats en augmentant notre compréhension des différents paramètres des fonctions utilisées. La suite de notre étude serait de pouvoir prédire l'évolution du volume de l'hippocampe d'une personne ainsi que de savoir si une personne est à risque de développer la maladie d'Alzheimer.

# Conclusions personnelles:

Ce projet nous aura surpris à bien des égards. En effet, nous avons pu grandement approfondir nos connaissances sur le sujet de l'intelligence artificielle. Grâce à ce projet et aux cours d'intelligence artificielle, nous avons réutilisé le langage de programmation Python qui fut l'un des premiers langages de programmation que nous avions appris. Si nous avions à refaire un tel projet, alors nous passerions plus de temps sur la planification, en effet, nous avons consacré la majorité de notre temps à rechercher la signification des paramètres de nos modèles et de plus nous manquions de connaissances sur les événements à venir, subséquemment nous n'avons que très peu suivis le planning prévisionnel.