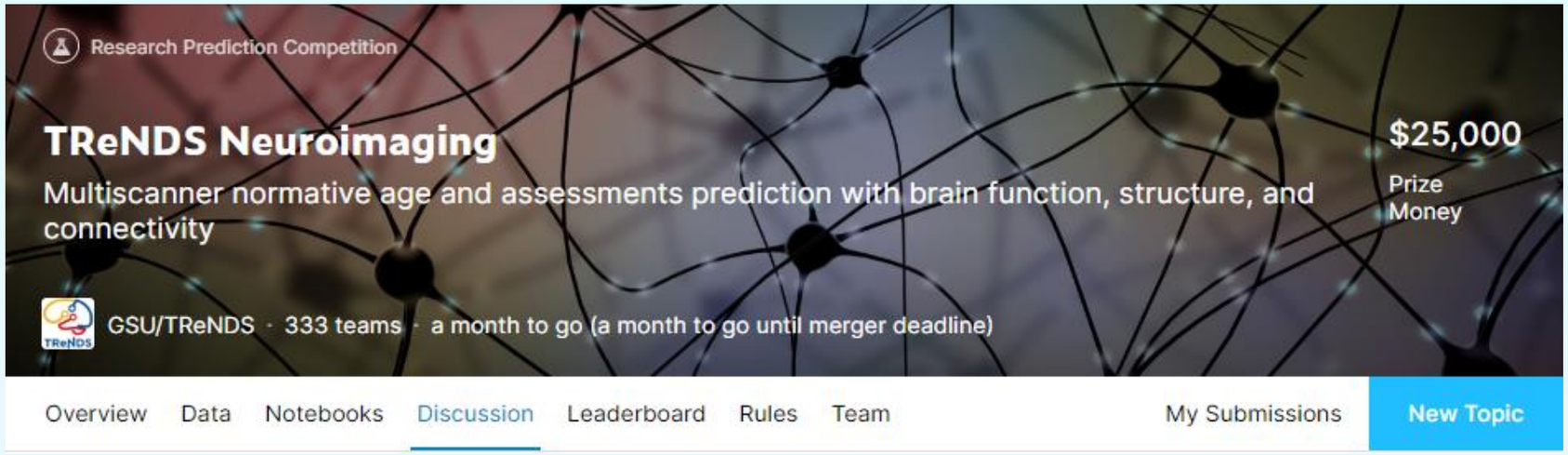
Mehdi Doubiani-25/05/2020

Etudiant Ingénieur Machine Learning OpenClassrooms



COMPÉTITION KAGGLE

IMAGERIE CEREBRALE : 3 APPROCHES

PROJET IML 8

SOMMAIRE

1°) Présentation de Kaggle

- Création : 2010
- Créateur : Anthony Goldbloom
- Principe :
  - Site de competition de data science
  - Site de cours en ligne
  - Forum de discution theme data
  - Réseau social
- Autres :
  - Différents niveaux de novice à master

2°) LA PROBLÉMATIQUE ET SON INTERPRÉTATION

Qu'est-ce que la neuroimagerie ?

- Techniques pour obtenir, des images de la structure, de la fonction ou de la pharmacologie du système nerveux.

La neuroimagerie se divise en deux grandes catégories :

- L'imagerie structurelle, qui traite de la structure du système nerveux et du diagnostic des maladies intracrâniennes (comme les tumeurs) et des blessures graves (à grande échelle).
- L'imagerie fonctionnelle, qui est utilisée pour diagnostiquer des maladies et des lésions métaboliques à une échelle plus fine (comme la maladie d'Alzheimer), ainsi que pour la recherche en psychologie neurologique, cognitive et la construction d'interfaces cerveau-ordinateur.
- **BUT :** Prédire l'âge et d'autres 4 autres variables (domain1-2_var1-2) issues de deux domaines (scanners) différents en utilisant des caractéristiques dérivées des images IRM du cerveau comme données d'entrée. Il faut donc prédire au total 5 variables numériques. Il s'agit donc d'un problème de régression.

2°) LA PROBLÉMATIQUE ET SON INTERPRÉTATION

Quelles sont les attentes vis-à-vis de l'hôte du concours ?

- Les hôtes attendent des modèles qui se généralisent bien sur les données d'un autre scanner/site (site 2). Tous les sujets du site 2 ont été assignés aux données de tests, leurs scores ne sont donc pas disponibles.
- Bien qu'il y ait moins d'individus du site 2 que d'individus du site 1 dans les données de tests, le nombre total d'individus du site 2 ne sera révélé qu'après la fin du concours. Pour rendre les choses plus intéressantes, les identifiants de certains sujets du site 2 ont été révélés.
- Utilisez-les pour que vos modèles prennent en compte les effets du site. Les effets de site sont une forme de biais. Pour bien généraliser, les modèles doivent apprendre des caractéristiques qui ne sont pas liées aux effets de site ou qui ne sont pas en relation avec eux.

2°) LA PROBLÉMATIQUE ET SON INTERPRÉTATION

Métrique utilisée pour la compétition

Submissions are scored using feature-weighted, normalized absolute errors.

$$\text{score} = \sum_f w_f \left(\frac{\sum_i |y_{f,i} - \hat{y}_{f,i}|}{\sum_i \hat{y}_{f,i}} \right)$$

where $y_{f,i}$ is the i^{th} observation of feature f, $\hat{y}_{f,i}$ is the corresponding ground truth for that observation, and w_f is a weighting given to each feature. The weights are `[.3, .175, .175, .175, .175]` corresponding to features `[age, domain1_var1, domain1_var2, domain2_var1, domain2_var2]`.

A small percentage of values are missing from the ground truth. These are skipped in the calculation. You should, though, make predictions for every row in the submission file.

3°) DONNÉES

- fMRI_train - un dossier contenant 53 cartes spatiales 3D pour les échantillons de trains au format [.mat].
- fMRI_test - un dossier contenant 53 cartes spatiales 3D pour des échantillons de test au format [.mat].
- fnc.csv - fonctions de corrélation FNC statique pour les échantillons de train et de test.
- ICN_numbers.csv - numéros de réseau de connectivité intrinsèque pour chaque carte spatiale d'IRMf ; correspond aux noms des FNC
- loading.csv - chargements sMRI SBM pour les échantillons de train et d'essai.
- train_scores.csv - âge et valeurs d'évaluation pour les échantillons de train.
- reveal_ID_site2.csv - une liste d'identifiants de sujets dont les données ont été collectées avec un scanner différent de celui des échantillons d’entrainement.
- fMRI_mask.nii - une carte spatiale binaire en 3D.
- sample_submission.csv - un exemple de fichier de soumission dans le format attendu

- Les fichiers .mat de ce concours peuvent être lus en python en utilisant `h5py`, et le fichier `.nii` peut être lu en python en utilisant nilearn.

4°) Analyse Exploratoire

L'ensemble de données comprend les fichiers importants suivants :

- `train_scores.csv` : Ce fichier contient les variables `age`, `domaine1_var1`, `domaine1_var2`, `domaine2_var1`, `domaine2_var2` comme variables caractéristiques importantes.
- `loading.csv` : Ce fichier contient les variables `IC_01` à `IC_29` comme variables de caractéristiques importantes.

```
display(train_data.head())
print("Shape of train_data :", train_data.shape)
```

executed in 12ms, finished 16:36:13 2020-05-17

	Id	age	domain1_var1	domain1_var2	domain2_var1	domain2_var2
0	10001	57.436077	30.571975	62.553736	53.325130	51.427998
1	10002	59.580851	50.969456	67.470628	60.651856	58.311361
2	10004	71.413018	53.152498	58.012103	52.418389	62.536641
3	10005	66.532630	NaN	NaN	52.108977	69.993075
4	10007	38.617381	49.197021	65.674285	40.151376	34.096421

```
Shape of train_data : (5877, 6)
```

```
display(loading_data.head())
print("Shape of loading_data :", loading_data.shape)
```

executed in 19ms, finished 16:36:18 2020-05-17

	Id	IC_01	IC_07	IC_05	IC_16	IC_26	IC_06	IC_10	IC_09	IC_18	...	IC_08	IC_03	IC_21	IC_28	IC_11
0	10001	0.006070	0.014466	0.004136	0.000658	-0.002742	0.005033	0.016720	0.003484	0.001797	...	0.018246	0.023711	0.009177	-0.013929	0.030696
1	10002	0.009087	0.009291	0.007049	-0.002076	-0.002227	0.004605	0.012277	0.002946	0.004086	...	0.014635	0.022556	0.012004	-0.011814	0.022479
2	10003	0.008151	0.014684	0.010444	-0.005293	-0.002913	0.015042	0.017745	0.003930	-0.008021	...	0.019565	0.030616	0.018184	-0.010469	0.029799
3	10004	0.004675	0.000957	0.006154	-0.000429	-0.001222	0.011755	0.013010	0.000193	0.008075	...	0.002658	0.022266	0.005956	-0.010595	0.024078
4	10005	-0.000398	0.006878	0.009051	0.000369	0.000336	0.010679	0.010352	0.003637	0.004180	...	0.009702	0.017257	0.005454	-0.008591	0.019416

5 rows × 27 columns

```
Shape of loading_data : (11754, 27)
```

4°) Analyse Exploratoire

Données manquantes :

3.2 Checking for Null values

3.2.0.1 train_data

```
# checking missing data
total = train_data.isnull().sum().sort_values(ascending = False)
percent = (train_data.isnull().sum()/train_data.isnull().count()*100).sort_values(ascending = False)
missing_train_data  = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_train_data.head()
```

executed in 25ms, finished 16:36:27 2020-05-17

	Total	Percent
domain1_var2	438	7.452782
domain1_var1	438	7.452782
domain2_var2	39	0.663604
domain2_var1	39	0.663604
age	0	0.000000

inference

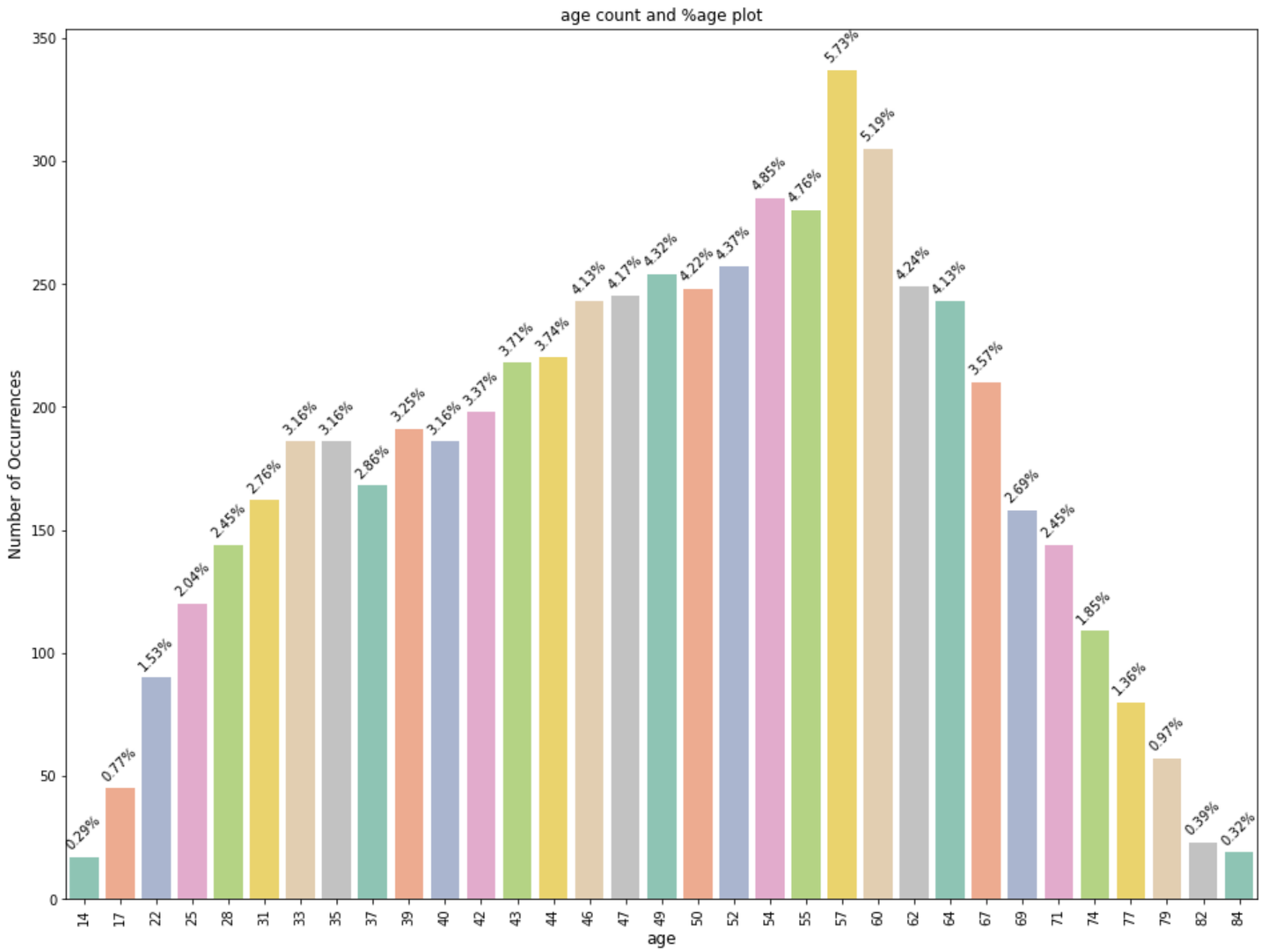
- `domain1_var2` and `domain1_var1` has 438 missing values each.
- `domain2_var2` and `domain2_var1` has 39 missing values each

Âge :

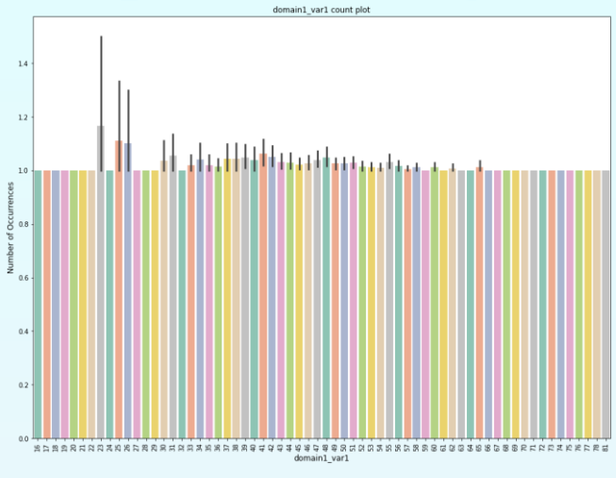
- Les 5 âges les plus fréquents sont 57, 60, 54, 55, 50
- La plupart des patients se situent entre 22 et 77 ans.

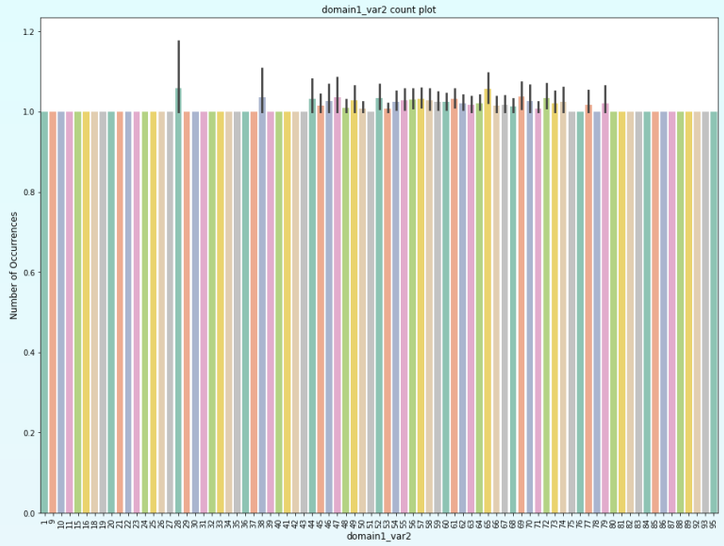
```
plot_bar(train_data, 'age', 'age count and %age plot', show_percent=True, size=4)
```

executed in 417ms, finished 16:36:43 2020-05-17

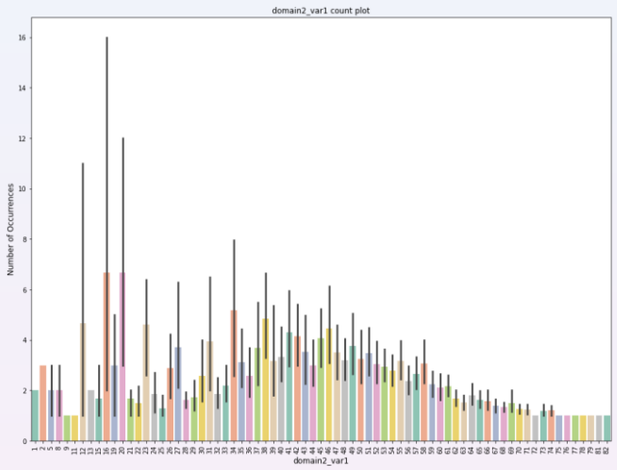


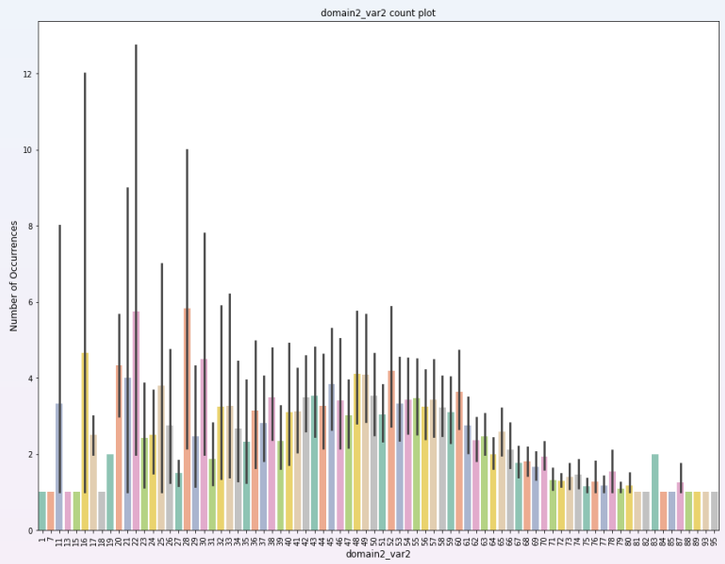
4°) Analyse Exploratoire





- `domaine1_var1` a 64 valeurs uniques lorsqu'il est arrondi aux entiers les plus proches, 23 est le plus fréquent.
- `domaine1_var2` a 81 valeurs uniques lorsqu'il est arrondi aux nombres entiers les plus proches, 28 est le plus fréquent.
- domaine2_var1 a 76 valeurs uniques lorsqu'il est arrondi à l'entier le plus proche, 16 étant la valeur la plus fréquente.
- `domaine2_var2` a 83 valeurs uniques lorsqu'il est arrondi aux nombres entiers les plus proches, 22 est le plus fréquent.





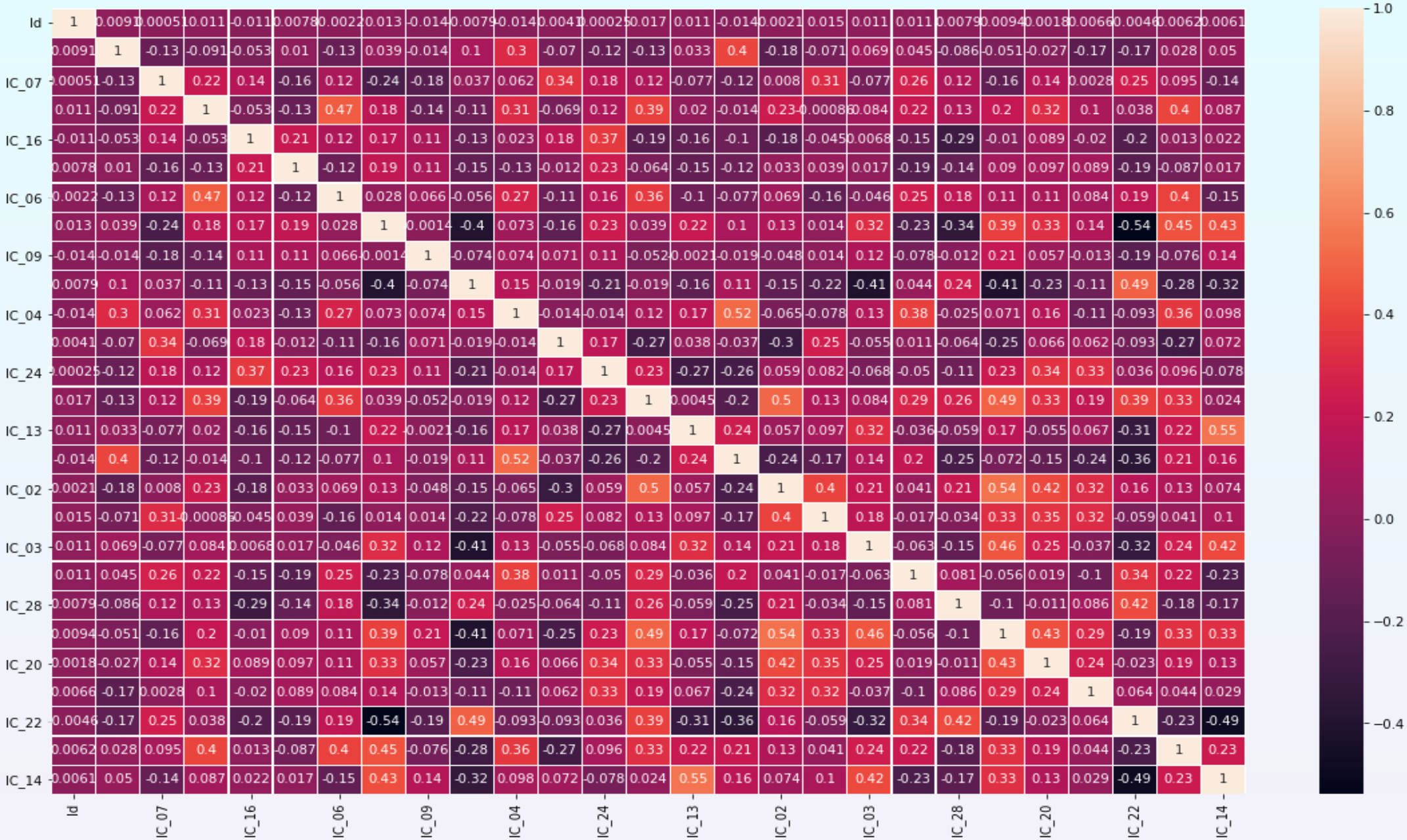
4°) Analyse Exploratoire

CORRELATIONS ENTRE VARIABLES CIBLES

L'"âge" et le "domaine1_var1" ont une valeur de corrélation de 0,33, ce qui est assez significatif et montre une corrélation positive entre ces deux variables.

4°) Analyse Exploratoire

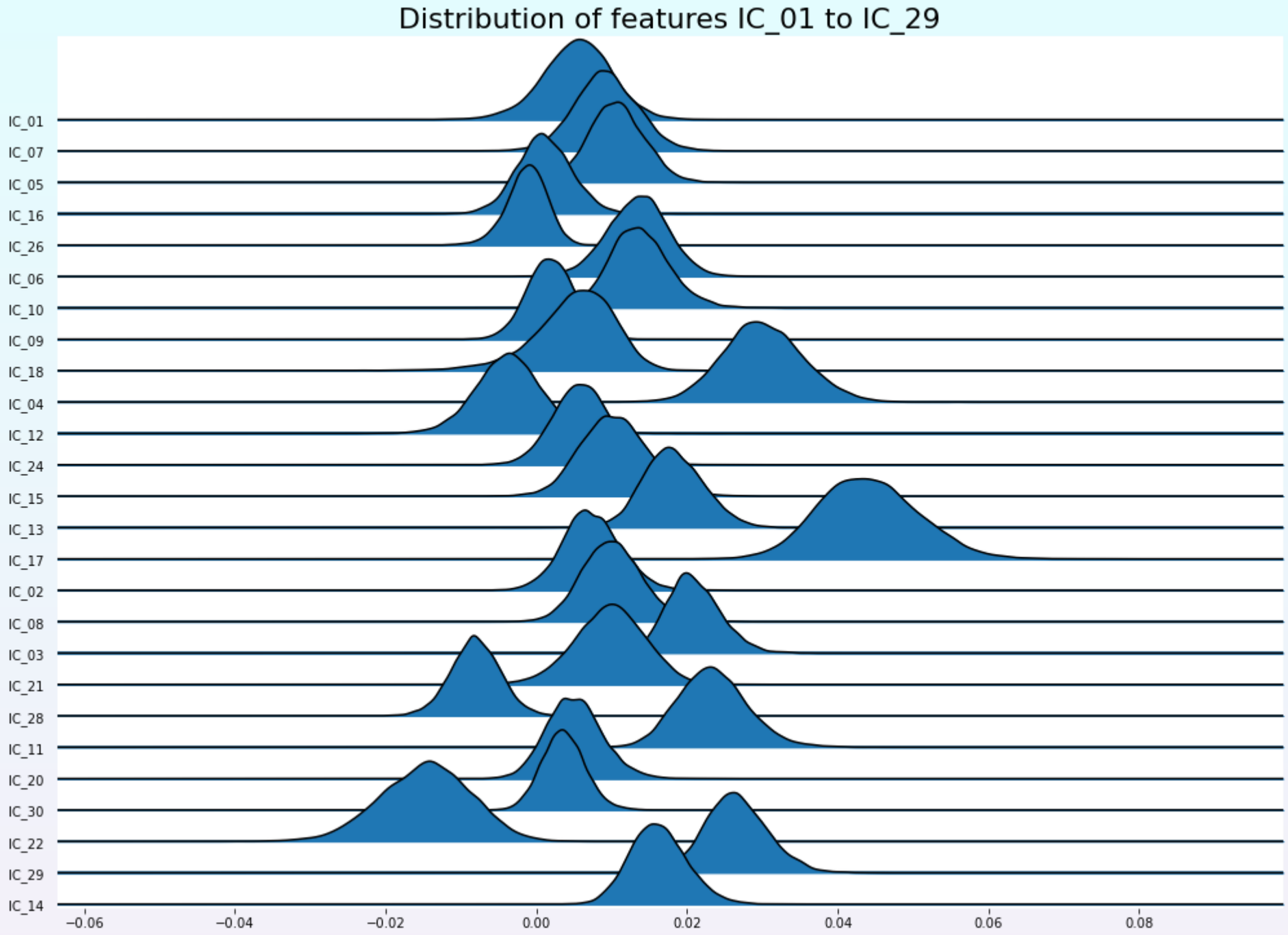
CORRELATIONS ENTRE VARIABLES EXPLICATIVES



La carte thermique ci-dessus montre une très forte corrélation entre certaines variables. Par exemple

- "IC_13" et "IC_14" ont une valeur de corrélation élevée de 0,55
- `IC_10` et `IC_22` ont également une corrélation très négative de -0,54

4°) Analyse Exploratoire



Graphique montrant la distribution des Variables de loading_data

5°) Préparation des données

X_TRAIN

```
X
```

executed in 24ms, finished 14:08:54 2020-05-18

	Id	IC_01	IC_07	IC_05	IC_16	IC_26	IC_06	IC_10	IC_09	IC_18	...	CBN(13)_vs_DMN(94)	CBN(18)_vs_DMN(94)	C
0	10001.0	0.006070	0.014466	0.004136	0.000658	-0.002742	0.005033	0.016720	0.003484	0.001797	...	-0.149279	0.552841	
1	10002.0	0.009087	0.009291	0.007049	-0.002076	-0.002227	0.004605	0.012277	0.002946	0.004086	...	-0.214216	-0.039792	
2	10004.0	0.004675	0.000957	0.006154	-0.000429	-0.001222	0.011755	0.013010	0.000193	0.008075	...	-0.130339	0.309540	
3	10005.0	-0.000398	0.006878	0.009051	0.000369	0.000336	0.010679	0.010352	0.003637	0.004180	...	-0.139525	0.394932	
4	10007.0	0.005192	0.010585	0.012160	-0.000920	-0.002255	0.011416	0.013838	0.001929	0.003051	...	-0.150218	0.408926	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
5872	21746.0	-0.001115	0.007108	0.008652	0.003596	0.000950	0.016314	0.017090	0.003513	0.004217	...	-0.249481	0.205351	
5873	21747.0	0.007263	0.016489	0.012704	0.004357	-0.005044	0.013909	0.019284	-0.006267	-0.000456	...	-0.119170	0.201846	
5874	21750.0	0.005996	0.003873	0.012353	0.000242	-0.002159	0.020201	0.020931	0.003684	-0.002458	...	-0.103786	0.375065	
5875	21752.0	0.000627	0.011407	0.010957	0.000534	-0.000347	0.013499	0.010541	0.001867	0.007447	...	0.229712	0.431489	
5876	21754.0	0.010670	0.010670	0.006662	-0.002215	-0.001773	0.006544	0.010900	0.000563	0.002995	...	0.341059	0.390722	

5877 rows × 1405 columns

TEST

```
test
```

executed in 24ms, finished 14:09:05 2020-05-18

	Id	IC_01	IC_07	IC_05	IC_16	IC_26	IC_06	IC_10	IC_09	IC_18	...	CBN(13)_vs_DMN(94)	CBN(18)_vs_DMN(94)	CE
0	10003	0.008151	0.014684	0.010444	-0.005293	-0.002913	0.015042	0.017745	0.003930	-0.008021	...	-0.154941	0.136850	
1	10006	0.000334	0.005311	0.010053	0.006920	-0.000065	0.015310	0.016543	0.004794	0.003982	...	-0.053606	0.240957	
2	10010	0.007103	0.006144	0.009770	-0.002884	-0.001346	0.015651	0.011613	-0.003291	0.013423	...	-0.244332	0.272077	
3	10011	0.004362	0.010240	0.010167	0.004492	-0.001623	0.017381	0.014680	0.007453	0.008786	...	-0.099726	0.557121	
4	10012	-0.007521	-0.003918	0.008434	-0.001145	0.002017	0.015065	0.019616	0.004140	-0.003744	...	-0.025230	0.203298	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
5872	21745	0.005406	0.006275	0.012252	0.003518	0.001400	0.015054	0.015373	0.001532	0.003546	...	0.179080	0.580813	
5873	21748	0.004240	0.009213	0.010981	0.000443	-0.003072	0.010702	0.014673	0.005523	0.005780	...	-0.106345	0.234340	
5874	21749	0.004783	0.017910	0.012128	-0.005683	-0.011613	0.017000	0.007230	0.001315	0.008788	...	-0.165575	0.170154	
5875	21751	0.003835	0.015067	0.015428	-0.002030	0.001205	0.012396	0.011026	-0.001491	0.005310	...	-0.087604	0.131902	
5876	21753	0.007431	0.021419	0.014143	-0.005623	-0.006399	0.008602	0.006831	-0.001018	0.014972	...	-0.147730	0.206375	

5877 rows × 1405 columns

Y_TRAIN

```
y
```

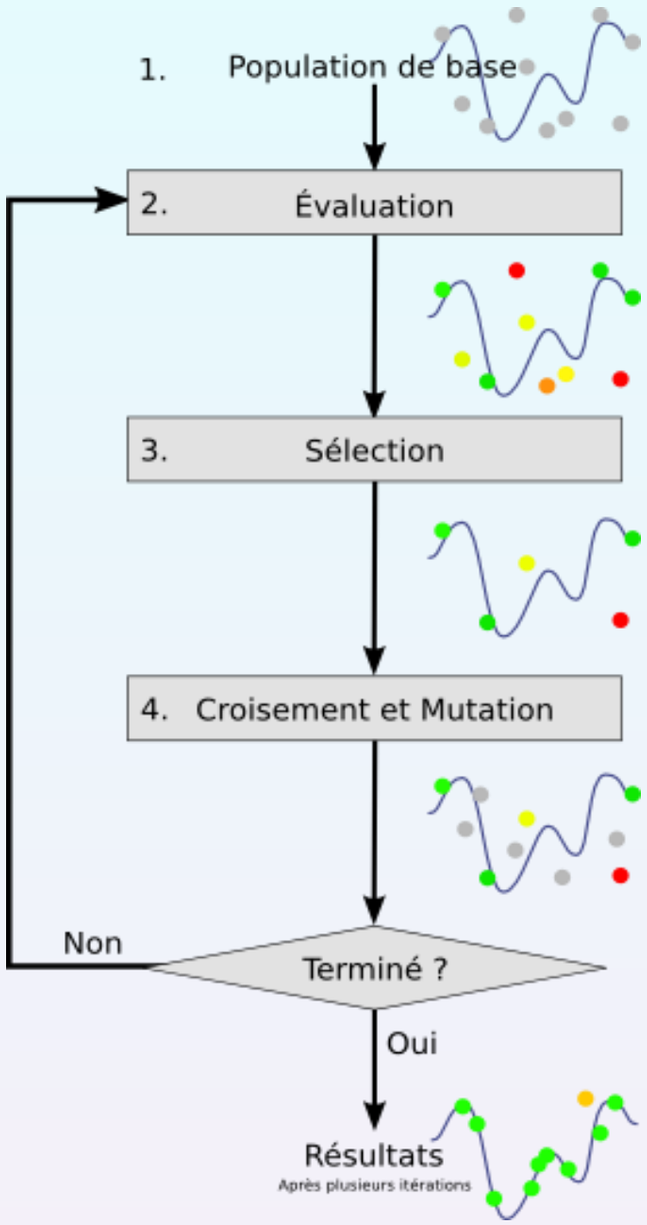
executed in 13ms, finished 14:08:56 2020-05-18

	Id	age	domain1_var1	domain1_var2	domain2_var1	domain2_var2
0	10001.0	57.436077	30.571975	62.553736	53.325130	51.427998
1	10002.0	59.580851	50.969456	67.470628	60.651856	58.311361
2	10004.0	71.413018	53.152498	58.012103	52.418389	62.536641
3	10005.0	66.532630	50.822224	55.305739	52.108977	69.993075
4	10007.0	38.617381	49.197021	65.674285	40.151376	34.096421
...	...	...	...	...	...	...
5872	21746.0	14.257265	21.358872	61.165998	51.778483	54.640179
5873	21747.0	55.456978	68.169675	29.907995	55.349257	54.019517
5874	21750.0	48.948756	55.114811	60.878271	38.617246	50.679885
5875	21752.0	66.532630	59.844808	72.303110	55.458281	46.870235
5876	21754.0	68.820928	56.594193	34.605868	49.922535	50.383078

5877 rows × 6 columns

6°) Modélisation

Approche Algorithme Génétique



1. La population est un ensemble de plusieurs équations constituée au hasard avec des opérateurs de base (addition, soustraction, division, multiplication)
2. Un individu est donc une seule équation parmi la population générée
3. Les gènes sont les opérations de base constituants les équations
4. La mutation d'un gène correspond à la modification d'un opérateur de base.
5. La métrique d'évaluation sera la « mean average error ».

6°) Modélisation

Approche Algorithme Génétique

Code (with annotations):

```
# Import modules
from gplearn.genetic import SymbolicRegressor
from sklearn.utils.random import check_random_state
import numpy as np
import graphviz

# Create 100 observations of (x0, x1, y) according to
# y = sin(x0) - log(x0+x1) + error
rng = check_random_state(0)

X_train = rng.uniform(1, 10, 200).reshape(100,2)
error = np.random.normal(0,0.1,100)
y_train = np.sin(X_train[:,0]) - np.log(X_train[:,0] + X_train[:,1]) + error[:]

# Symbolic Regression
est_gp = SymbolicRegressor(
            population_size=5000,
            generations=20,
            metric='rmse',                          } [1]
            stopping_criteria=0.01,
            function_set=('add','sub','mul','div','sin','cos','log'),} [2]
            p_crossover=0.7,
            p_subtree_mutation=0.1,
            p_hoist_mutation=0.05,                  } [3]
            p_point_mutation=0.1,
            max_samples=0.9,                        } [4]
            verbose=1,
            parsimony_coefficient=0.01,             } [5]
            random_state=0,
            n_jobs=2,
            low_memory=True)

est_gp.fit(X_train, y_train)
```

[1] The `metric` option is the fitness function, set to RMSE. The `stopping_criteria` is the value that the RMSE must drop to for the program to terminate early. We have set this unreasonably low so that we can see how the program evolves over 20 generations.

[2] We have specified a small `function_set` that includes the ones the program will need to complete the task. In the `gplearn` package we also have the option to define our own custom functions.

[3] The `p_:` options give us control over the mix of crossover and mutation. Here there are 3 types of mutation; as well as point mutation, there are some more complex subtree mutations available.

[4] `max_samples` is a proportion of the data that the fitness of each program is evaluated on. Setting this to <1 will affect the output as shown below.

[5] The `parsimony_coefficient` is an added penalty component to the fitness function that penalises the length of the program.

Notre objectif est de choisir des critères de fin qui évitent s'arrêter trop tôt, par exemple à un maximum local qui n'est pas de qualité satisfaisante, mais évite aussi de trop dépenser de longues heures de recherches lorsqu'aucune amélioration ne sera apportée dans un délai raisonnable.

Nous disposons de plusieurs options pour les critères de fin, notamment :

1. Un modèle satisfaisant est trouvé
2. Il y a eu un nombre prédéterminé de nouvelles générations sans qu'aucune amélioration ne soit apportée aux meilleurs modèles dans la population
3. Un nombre maximum déterminé de générations a été créé

Une bonne idée est d'utiliser une combinaison de plusieurs critères, car le processus peut être coûteux en termes de calcul avec une grande population.

6°) Modélisation

Approche Algorithme Génétique

AVANTAGES

- La régression symbolique présente des avantages évidents de par sa conception : elle est conçue pour explorer un large espace de modèles possibles.
- La probabilité qu'il en trouve une qui corresponde bien aux données est donc très élevée, et elle peut être réglée par la fonction de fitness pour sanctionner de manière appropriée le surapprentissage en données de formation.
- L'inclusion de fonctions telles que log () et sin () donne à l'algorithme la possibilité d'expérimenter diverses transformations de chaque variable, ce qui, sans le contexte des données, peut être très difficile à trouver.
- Si notre objectif est de sélectionner le meilleur modèle, nous pouvons prendre le meilleur individu de la dernière génération comme dans l'exemple utilisant la librairie gplearn.
- Toutefois, il est également possible de mettre en place un programme qui retient l'ensemble de la population et d'envisager de ne retenir que les meilleurs modèles de la dernière génération.

INCONVÉNIENTS

- La régression symbolique est très coûteuse en termes de calcul, c'est pourquoi il est judicieux d'inclure le nombre de générations ou la durée d'exécution du programme dans les critères de fin.
- Il y a également de nombreux paramètres qui doivent être testés et qui peuvent influencer la vitesse d'apprentissage, tels que l'équilibre entre la mutation et le crossover, qui nécessitent de l'expérience pour s'accorder correctement.

6°) Modélisation

Approche Algorithme Génétique
RÉSULTATS

Pour la variable Âge, la Mean Average Error est de 10.918846772701464
Pour la variable Domain1_var1, la Mean Average Error est de 7.650291769242
Pour la variable Domain1_var2, la Mean Average Error est de 8.270192160941
Pour la variable Domain2_var1, la Mean Average Error est de 8.761652683147
Pour la variable Domain2_var2, la Mean Average Error est de 9.248416619867

Après soumission à la plateforme Kaggle, La feature weighted normalized error (métrique du concours) est de : 0.188 ce qui en fait un résultat correct mais sans plus par rapport aux autres méthodes et aux autres candidats à la compétition.

6°) Modélisation

Package PyCARET : Test de l'AutoML

Introduction à PyCaret

PyCaret est une bibliothèque d'apprentissage machine en Python,
à code source ouvert et à faible niveau de codage, qui vise à réduire le temps
du passage entre l'hypothèse et la compréhension jusqu'au déploiement
du modèle en production

2. Compare Model

La fonction `compare_models` utilise tous les modèles de la bibliothèque de modèles et les note en utilisant la validation croisée K-fold. La sortie imprime une grille de score qui montre les MAE, MSE, RMSE, R2, RMSLE et MAPE par fold (CV par défaut = 10 folds) de tous les modèles disponibles dans la bibliothèque de modèles.

Entrée [66]:
```
compare_models(
    blacklist = blacklist_models,
    fold = 10,
    sort = 'MAE', ## competition metric
    turbo = True
)
```

executed in 2h 30m 27s, finished 00:08:48 2020-05-09

Out[66]:

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Bayesian Ridge	7.183200	82.956100	9.105200	0.540700	0.207100	0.168800
1	Orthogonal Matching Pursuit	7.447600	89.283300	9.444900	0.505800	0.213700	0.174100
2	CatBoost Regressor	7.534600	90.904400	9.531700	0.496800	0.217100	0.178900
3	Gradient Boosting Regressor	7.758100	96.638900	9.828400	0.464900	0.223300	0.184700
4	Light Gradient Boosting Machine	7.760200	97.021000	9.848400	0.462800	0.223300	0.184000
5	Extreme Gradient Boosting	7.760500	95.883700	9.790000	0.469100	0.222500	0.184600
6	Extra Trees Regressor	8.164300	106.361100	10.311000	0.411000	0.233600	0.195200
7	Random Forest	8.248600	109.082400	10.441700	0.396100	0.235700	0.196800
8	Linear Regression	8.373300	112.346700	10.589100	0.377600	0.235500	0.191300
9	AdaBoost Regressor	8.374200	111.657700	10.564600	0.381800	0.235400	0.195500
10	Ridge Regression	9.941700	153.357800	12.383000	0.150900	0.275000	0.239700
11	Lasso Least Angle Regression	10.872100	180.829400	13.446600	-0.001300	0.294900	0.261300
12	Elastic Net	10.876900	180.951000	13.451100	-0.001900	0.295000	0.261400
13	Lasso Regression	10.876900	180.950800	13.451100	-0.001900	0.295000	0.261400
14	Support Vector Machine	10.884600	181.343400	13.465600	-0.004100	0.296100	0.263300
15	Random Sample Consensus	11.290900	202.462200	14.217000	-0.121800	0.335600	0.254500
16	K Neighbors Regressor	11.814300	215.419200	14.673400	-0.192700	0.316600	0.280200
17	Decision Tree	12.132300	234.108700	15.296300	-0.296500	0.340700	0.276500
18	Huber Regressor	13.542800	283.683600	16.833100	-0.570800	0.361000	0.307000
19	Least Angle Regression	15.634200	855.317400	19.701100	-3.742500	0.355100	0.351600
20	Passive Aggressive Regressor	24.783800	1165.538800	29.088100	-5.460400	0.516700	0.579200

1. Setup

```
Setup Succesfully Completed!
```

	Description	Value
0	session_id	5310
1	Transform Target	False
2	Transform Target Method	None
3	Original Data	(5877, 1406)
4	Missing Values	False
5	Numeric Features	1405
6	Categorical Features	0
7	Ordinal Features	False
8	High Cardinality Features	False
9	High Cardinality Method	None
10	Sampled Data	(5877, 1406)
11	Transformed Train Set	(4701, 1405)
12	Transformed Test Set	(1176, 1405)
13	Numeric Imputer	mean
14	Categorical Imputer	constant
15	Normalize	False
16	Normalize Method	None
17	Transformation	False
18	Transformation Method	None
19	PCA	False
20	PCA Method	None
21	PCA Components	None
22	Ignore Low Variance	False
23	Combine Rare Levels	False
24	Rare Level Threshold	None
25	Numeric Binning	False
26	Remove Outliers	False
27	Outliers Threshold	None
28	Remove Multicollinearity	False
29	Multicollinearity Threshold	None
30	Clustering	False
31	Clustering Iteration	None
32	Polynomial Features	False
33	Polynomial Degree	None
34	Trignometry Features	False
35	Polynomial Threshold	None
36	Group Features	False
37	Feature Selection	False
38	Features Selection Threshold	None
39	Feature Interaction	False
40	Feature Ratio	False
41	Interaction Threshold	None

6°) Modélisation

Package PyCARET : Test de l'AutoML

3. Create Model

La fonction `create_model` crée un modèle et le note en utilisant la validation croisée K-fold. (par défaut = 10 fois).
La sortie imprime une grille de score qui montre MAE, MSE, RMSE, RMSLE, R2 et MAPE. Cette fonction renvoie un objet modèle entrainé.
La fonction setup() doit être appelée avant d'utiliser create_model()

Entrée [67]:
```
br_age = create_model(
    estimator='br',
    fold=10
)
```
executed in 1m 0.04s, finished 07:32:41 2020-05-09

	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	7.0647	80.0074	8.9447	0.5572	0.1945	0.1594
1	7.1649	79.9125	8.9394	0.5509	0.2030	0.1666
2	7.2793	85.8612	9.2661	0.5326	0.2202	0.1786
3	7.1389	81.7431	9.0412	0.5386	0.1947	0.1617
4	6.7749	74.4708	8.6296	0.5895	0.1938	0.1556
5	7.4633	89.2604	9.4478	0.5051	0.2127	0.1721
6	7.4320	87.8510	9.3729	0.5378	0.2201	0.1809
7	7.1628	82.4321	9.0792	0.5375	0.2040	0.1647
8	7.0269	82.9781	9.1092	0.5291	0.2134	0.1710
9	7.3239	85.0439	9.2219	0.5281	0.2145	0.1775
Mean	7.1832	82.9561	9.1052	0.5407	0.2071	0.1688
SD	0.1942	4.1013	0.2264	0.0210	0.0099	0.0082

4. Tuning Model

La fonction `tune_model` règle les hyperparamètres d'un modèle et le classe en utilisant la validation croisée K-fold. La sortie imprime la grille de score qui indique MAE, MSE, RMSE, R2, RMSLE et MAPE par folds (par défaut = 10 folds). Cette fonction renvoie un objet modèle entrainé.

Entrée [69]:
```
# here we are tuning the above created model
tuned_br_age = tune_model(
    estimator='br',
    fold=10,
    optimize = 'mae',
    n_iter=50
)
```
executed in 11m 40s, finished 07:47:03 2020-05-09

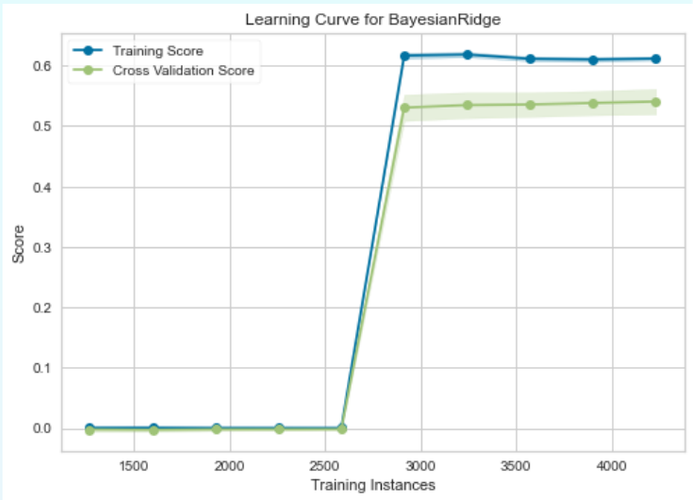
	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	7.0647	80.0071	8.9447	0.5572	0.1945	0.1594
1	7.1649	79.9121	8.9394	0.5509	0.2030	0.1666
2	7.2793	85.8608	9.2661	0.5326	0.2202	0.1786
3	7.1390	81.7433	9.0412	0.5386	0.1947	0.1617
4	6.7749	74.4700	8.6296	0.5895	0.1938	0.1556
5	7.4633	89.2601	9.4478	0.5051	0.2127	0.1721
6	7.4320	87.8500	9.3728	0.5378	0.2201	0.1809
7	7.1627	82.4320	9.0792	0.5375	0.2040	0.1647
8	7.0270	82.9784	9.1092	0.5291	0.2134	0.1710
9	7.3239	85.0433	9.2219	0.5281	0.2145	0.1775
Mean	7.1832	82.9557	9.1052	0.5407	0.2071	0.1688
SD	0.1942	4.1013	0.2264	0.0210	0.0099	0.0082

6°) Modélisation

Package PyCARET : Test de l'AutoML

5. Plot Model

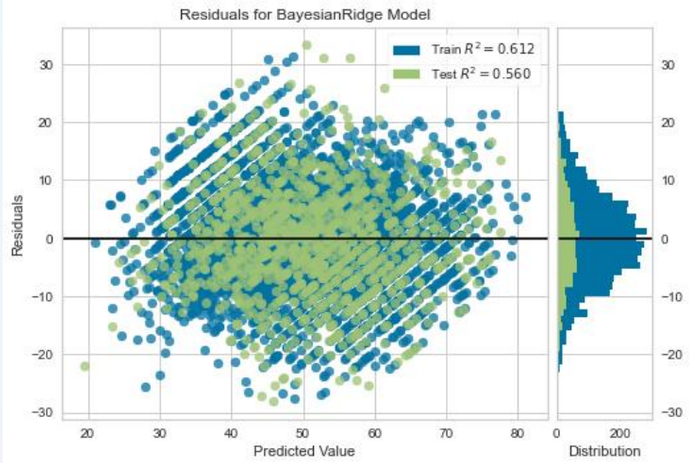
La fonction 'plot_model' prend un objet modèle entraîné et retourne un tracé basé sur l'ensemble test / hold-out. Dans certains cas, le processus peut nécessiter un nouvel entrainement du modèle. Le modèle doit être créé à l'aide de la fonction create_model() ou tune_model()

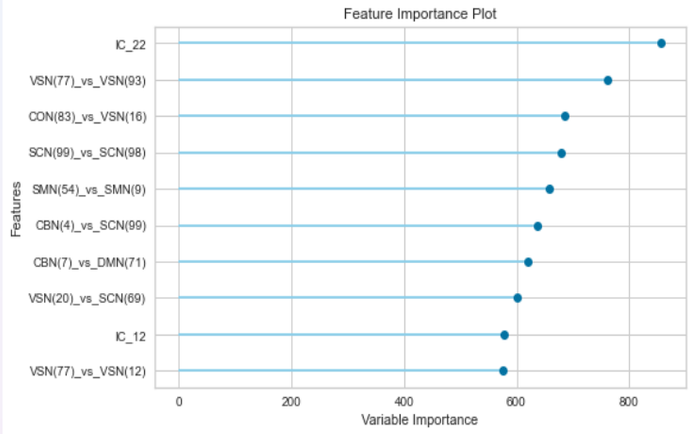


6. Predict Model

	Id	SCN(53)_vs_SCN(69)	SCN(98)_vs_SCN(69)	SCN(99)_vs_SCN(69)	SCN(45)_vs_SCN(69)	ADN(21)_vs_SCN(69)	ADN(56)_vs_SCN(69)	SMN(3)_vs_SCN(69)
0	10003	0.000687	0.000220	0.001483	0.001157	-0.001353	-0.000874	-0.000591
1	10006	0.000647	0.000234	0.001281	0.000641	-0.000639	-0.000606	-0.000428
2	10010	-0.000453	-0.000087	0.000669	0.000638	-0.000342	-0.000519	-0.000279
3	10011	0.000335	0.000332	0.001042	0.000810	-0.000311	-0.000076	-0.000104
4	10012	0.001039	0.000594	0.001191	0.001549	-0.001016	-0.000164	0.000077

La fonction `predict_model` est utilisée pour prédire de nouvelles données en utilisant un estimateur entrainé. Elle accepte un estimateur créé à l'aide d'une des fonctions de pycaret qui renvoie un objet modèle entrainé ou une liste d'objets modèles entrainés créés à l'aide de stack_models() ou de create_stacknet(). Les nouvelles données non vues peuvent être transmises aux paramètres de données sous forme de pandas Dataframe. Si les données ne sont pas passées, le jeu de test / hold-out séparé au moment du setup() est utilisé pour générer des prédictions.





6°) Modélisation

Package PyCARET : Test de l'AutoML

En observant de près les comparaisons des modèles ci-dessus, nous avons fait les observations suivantes :

- **"âge" : Le modèle Bayesian Ridge a le MAE minimum : 7.228200**
- **`domaine1_var1` : Le régresseur CatBoost a le MAE minimum : 7.888200**
- **`domaine1_var2` : Les machines à vecteurs de soutien ont le MAE minimum : 8.982500**
- **`domaine2_var1` : Le régresseur CatBoost a le MAE minimum : 8.748100**
- **`domaine2_var2` : Le régresseur CatBoost a le MAE minimum : 9.239200**

Après soumission des prédictions sur la plateforme KAGGLE, la feature-weighted normalized error (Métrique de la compétition est de : 0.161) ce qui est le meilleur résultat des 3 approches (PyCARET, Algo Génétique et Deep Learning)

6°) Modélisation

Modèle Deep Learning (AutoKeras / AutoML)

Conception de réseau neurones automatiquement

La recherche d'architecture neuronale (NAS) utilise l'apprentissage machine pour automatiser la conception des RNA (Réseaux de Neurones Artificiels). Diverses approches de la NAS ont permis de concevoir des réseaux qui se comparent bien aux systèmes conçus à la main. L'algorithme de recherche de base consiste à proposer un modèle candidat, à l'évaluer par rapport à un ensemble de données et à utiliser les résultats comme retour d'information pour apprendre le réseau NAS. Les systèmes disponibles comprennent AutoML et AutoKeras.

Les questions de conception comprennent le nombre, le type et la connectivité des couches du réseau, ainsi que la taille de chacune et le type de connexion (complète, groupée, ...).

Les hyperparamètres doivent également être définis dans le cadre de la conception (ils ne sont pas appris), régissant des questions telles que le nombre de neurones dans chaque couche, le taux d'apprentissage, le pas, la foulée, la profondeur, le champ de réception et le padding (pour les CNN), etc.

6°) Modélisation

Modèle Deep Learning (AutoKeras / AutoML)

Implémentation

Elle est très simple :

L'avantage de cette technique est qu'elle permet d'avoir une prédiction unique pour nos 5 variables en une seule et même prédiction (régression multi-variables)

L'inconvénient est que ce package est toujours en cours de développement et ne peut donc être utilisé de manière optimale pour l'instant (bugs d'export de modèles…)

Entrée [*]:

```
import autokeras as ak
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.model_selection import KFold,cross_val_score
from tensorflow.keras.layers import Dropout
from keras.callbacks import History, ModelCheckpoint, EarlyStopping
from tensorflow.keras import backend as K

#Custom Metric/Loss Function For keras
weight_factor = K.constant([.3, .175, .175, .175, .175],dtype='float32',shape=(5,1))
def tf_fwnae(y_true, y_pred): #feature-weighted, normalized absolute errors.
    diff = K.abs((y_true - y_pred) / K.clip(K.abs(y_true),
                                            K.epsilon(),
                                            None))
    return K.dot(diff,weight_factor)

regressor  = ak.StructuredDataRegressor(max_trials=100,loss=tf_fwnae,metrics=['mape'], seed=42,output_dim=5)
regressor.fit(x=X.iloc[:,1:], y=y.iloc[:,1:],validation_split=0.25,
                              callbacks=[History(),
                              EarlyStopping(monitor='val_loss',
                              patience=30,
                              restore_best_weights=True,
                              verbose=1)])
```

execution queued 14:09:14 2020-05-18

```
7 - mape: 25.780 - ETA: 3s - loss: 0.2603 - mape: 25.960 - ETA: 3s - loss: 0.2603 - mape: 25.965 - ETA: 3s - loss: 0.2598 - m
ape: 25.919 - ETA: 3s - loss: 0.2600 - mape: 25.935 - ETA: 3s - loss: 0.2599 - mape: 25.924 - ETA: 3s - loss: 0.2597 - mape:
25.901 - ETA: 3s - loss: 0.2597 - mape: 25.904 - ETA: 3s - loss: 0.2601 - mape: 25.942 - ETA: 3s - loss: 0.2600 - mape: 25.92
2 - ETA: 2s - loss: 0.2601 - mape: 25.938 - ETA: 2s - loss: 0.2596 - mape: 25.888 - ETA: 2s - loss: 0.2593 - mape: 25.855 - E
TA: 2s - loss: 0.2593 - mape: 25.868 - ETA: 2s - loss: 0.2591 - mape: 25.845 - ETA: 2s - loss: 0.2587 - mape: 25.804 - ETA: 2
s - loss: 0.2590 - mape: 25.832 - ETA: 2s - loss: 0.2587 - mape: 25.805 - ETA: 2s - loss: 0.2591 - mape: 25.843 - ETA: 2s - l
oss: 0.2589 - mape: 25.814 - ETA: 2s - loss: 0.2601 - mape: 25.953 - ETA: 2s - loss: 0.2602 - mape: 25.954 - ETA: 2s - loss:
0.2601 - mape: 25.954 - ETA: 1s - loss: 0.2601 - mape: 25.960 - ETA: 1s - loss: 0.2600 - mape: 25.948 - ETA: 1s - loss: 0.259
8 - mape: 25.927 - ETA: 1s - loss: 0.2599 - mape: 25.935 - ETA: 1s - loss: 0.2598 - mape: 25.916 - ETA: 1s - loss: 0.2598 - m
ape: 25.921 - ETA: 1s - loss: 0.2599 - mape: 25.927 - ETA: 1s - loss: 0.2594 - mape: 25.881 - ETA: 1s - loss: 0.2595 - mape:
25.888 - ETA: 1s - loss: 0.2592 - mape: 25.857 - ETA: 1s - loss: 0.2592 - mape: 25.854 - ETA: 1s - loss: 0.2591 - mape: 25.84
9 - ETA: 1s - loss: 0.2590 - mape: 25.834 - ETA: 0s - loss: 0.2588 - mape: 25.813 - ETA: 0s - loss: 0.2585 - mape: 25.787 - E
TA: 0s - loss: 0.2582 - mape: 25.758 - ETA: 0s - loss: 0.2579 - mape: 25.733 - ETA: 0s - loss: 0.2579 - mape: 25.734 - ETA: 0
s - loss: 0.2575 - mape: 25.704 - ETA: 0s - loss: 0.2576 - mape: 25.715 - ETA: 0s - loss: 0.2579 - mape: 25.746 - ETA: 0s - l
oss: 0.2580 - mape: 25.759 - ETA: 0s - loss: 0.2580 - mape: 25.756 - ETA: 0s - loss: 0.2578 - mape: 25.738 - ETA: 0s - loss:
0.2579 - mape: 25.742 - 14s 103ms/step - loss: 0.2575 - mape: 25.7200 - val_loss: 0.2525 - val_mape: 25.4174
Epoch 257/1000

138/138 [==============================] - ETA: 31s - loss: 0.2682 - mape: 26.48 - ETA: 21s - loss: 0.2528 - mape: 25.32 - ET
A: 17s - loss: 0.2614 - mape: 26.21 - ETA: 15s - loss: 0.2650 - mape: 26.60 - ETA: 14s - loss: 0.2663 - mape: 26.72 - ETA: 13
```

6°) Modélisation

Modèle Deep Learning (AutoKeras / AutoML)

Résultats

La MAE avec un réseau neuronal structuré est de 8.8129, ceci pour l'ensemble des prévisions des 5 variables. Cependant après soumission sur la plateforme,
le résultat avec le feature weighted normalized error (la métrique de la compétition) n'est que de 0.194, soit le plus mauvais résultat des 3 approches.

```
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.model_selection import KFold,cross_val_score
from tensorflow.keras.layers import Dropout
from keras.callbacks import History, ModelCheckpoint, EarlyStopping

model = Sequential()
model.add(Dense(1404,input_dim=1404,kernel_initializer='normal',activation='relu'))
model.add(Dense(702,kernel_initializer='normal',activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(702,kernel_initializer='normal',activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(702,kernel_initializer='normal',activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(5,kernel_initializer='normal'))

model.compile(loss='mean_absolute_error',optimizer='adam', metrics = ['accuracy'])

model.summary()
```

executed in 146ms, finished 13:27:29 2020-05-12

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_35 (Dense)             (None, 1404)              1972620
_________________________________________________________________
dense_36 (Dense)             (None, 702)               986310
_________________________________________________________________
dropout_18 (Dropout)         (None, 702)               0
_________________________________________________________________
dense_37 (Dense)             (None, 702)               493506
_________________________________________________________________
dropout_19 (Dropout)         (None, 702)               0
_________________________________________________________________
dense_38 (Dense)             (None, 702)               493506
_________________________________________________________________
dropout_20 (Dropout)         (None, 702)               0
_________________________________________________________________
dense_39 (Dense)             (None, 5)                 3515
=================================================================
Total params: 3,949,457
Trainable params: 3,949,457
Non-trainable params: 0
_________________________________________________________________
```

CONCLUSIONS

- L'AutoML (PyCARET) permet d'arriver à un très bon résultat rapidement et avec peu de code et de temps entre l'idée et le résultat de la modélisation.
- Par ailleurs, PyCARET permet de générer le code de chaque étape afin que l'on puisse intervenir manuellement et faire les ajustements nécessaires pour améliorer encore plus le résultat ou l'intégration dans un pipeline encore plus important.
- Donc des 3 approches voici le classement des méthodes avec la métrique **feature weighted normalized error** :
  1. PyCARET (0.161)
  2. Algorithme Génétique (0.188)
  3. Deep Learning (0.194)

A titre comparatif, en date du 25 Mai 2020 le leader de la compétition a un score **feature weighted normalized error** de 0.158 avec une approche Deep Learning Convolutionnelle 3D sur les scans

ATTENTION : Les scores de la métrique de la compétition sont des scores dits « publiques » car évalués sur 48% des données de test , les 52% des données restant seront évalués avec les modèles soumis après la fin de la compétition, cela donnera le score dit « privés »

CONCLUSIONS

	Âge		Domain1_var1		Domain1_var2		Domain2_var1		Domain2_var2	
	Public score Feature Weighted Normalized	Mean Average Error	Public score Feature Weighted Normalized	Mean Average Error	Public score Feature Weighted Normalized	Mean Average Error	Public score Feature Weighted Normalized	Mean Average Error	Public score Feature Weighted Normalized	Mean Average Error
PyCARET	0.161	Bayesian Ridge 7.228200	0.161	CatBoost 7.888200	0.161	SVM 8.982500	0.161	CatBoost 8.748100	0.161	CatBoost 9.239200
Symbolic Regressor	0.188	10.9188	0.188	7.6502	0.188	8.2701	0.188	8.7616	0.188	9.2484
Deep Learning	0.194	8.8129	0.194	8.8129	0.194	8.8129	0.194	8.8129	0.194	8.8129

AXES D'AMELIORATION

- Faire plus d'explorations avec les algorithmes génétiques en faisant un **GridSearchCV** par exemple.
- Utiliser la fonction **Symbolic Transformer** de gplearn pour créer des variables non liées linéaire à la variable cible et/ou explicatives afin d'avoir de nouvelles variables à utiliser dans notre modèle d'algorithme génétique.
- Faire de **l'optimisation d'hyperparamètres** encore plus poussée sous **PyCARET**
- Faire de **l'optimisation d'hyperparamètres** encore plus poussée pour l'approche **Deep Learning**
- Faire une approche de prévision avec l'aides **caractéristiques 3D spatiales** des scans fournis (écartée dans mes approches)
- Faire à la main des **modèles multi-variables cibles de régression (MultiOutputRegressor)**
- Faire des ensembles de modèles **(stacking ,bagging ,etc…)**
- Réutiliser les approches **PyCARET et Symbolic Regression ainsi que l'approche Deep Learning** avec comme fonction d'optimisation la métrique **Feature weighted normalized error** (celle de la compétition)

MERCI DE VOTRE ATTENTION