PTML

Guillet William | Prenleloup Louis | Ribon Axel

Table des matières

1 Supervisé

1.1 Dataset

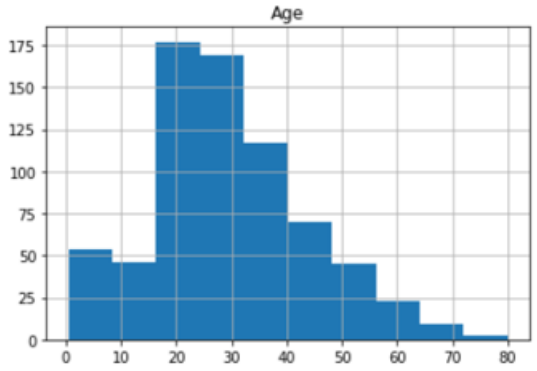
Pour la partie Supervisé de l'application du machine learning nous avons décidé de choisir le dataset du titanic. Ce dataset comprend 891 lignes et 12 colonnes dont les attributs représentent :

— Identifiant du passager
— Booléen déterminant la survie ou non du passager
— Classe du ticket
— Nom
— Genre
— Age
— Nombre de freres et soeurs / épou(x)(ses) sur le Titanic
— Nombre de parents / enfants sur le Titanic
— Numero de ticket
— Prix du ticket
— Numero de cabine
— Port d'embarcation

Suite à l'analyse du dataset nous avons pu analyser sa répartition et comprendre sur quel genre du jeu de données nous nous apprêtions à travailler.

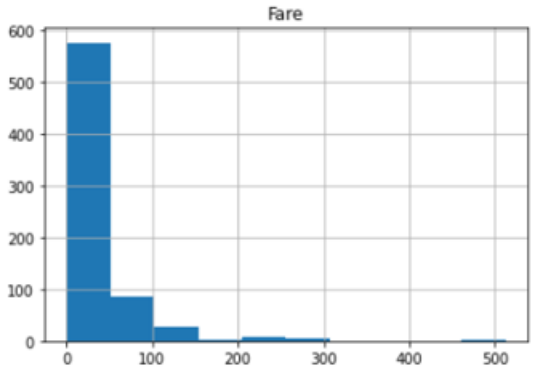
On a pu déterminer par exemple que :

- L'age moyen est de 30 ans



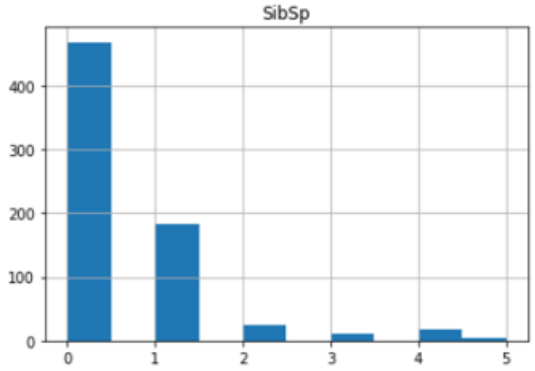

```
count    712.000000
mean      29.642093
std       14.492933
min        0.420000
25%       20.000000
50%       28.000000
75%       38.000000
max       80.000000
Name: Age, dtype: float64
```

- L'écart type du prix est de 52 avec un maximum à 512



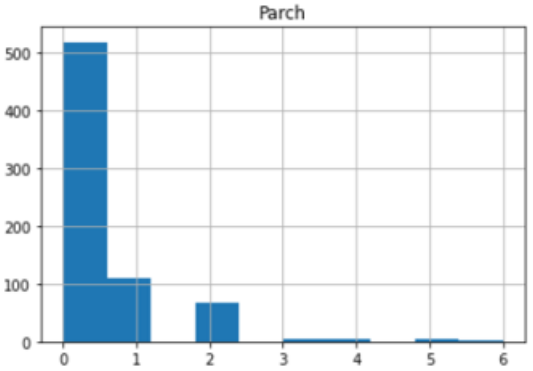

```
count    712.000000
mean      34.567251
std       52.938648
min        0.000000
25%        8.050000
50%       15.645850
75%       33.000000
max      512.329200
Name: Fare, dtype: float64
```

- Le nombre de frères ou époux moyen est de 0.5



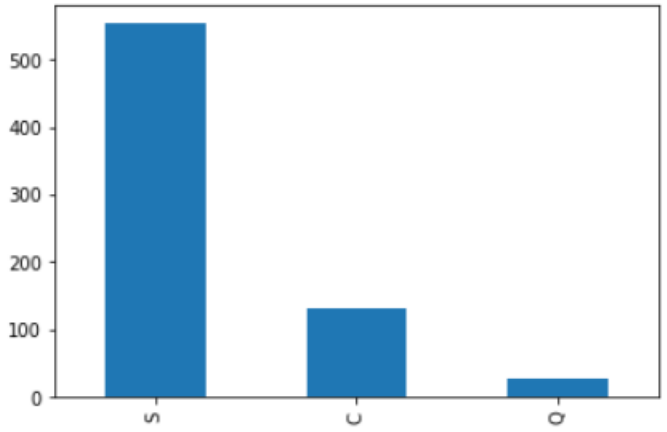

```
count    712.000000
mean       0.514045
std        0.930692
min        0.000000
25%        0.000000
50%        0.000000
75%        1.000000
max        5.000000
Name: SibSp, dtype: float64
```

- Le nombre de parents ou d'enfants moyen est de 0.4

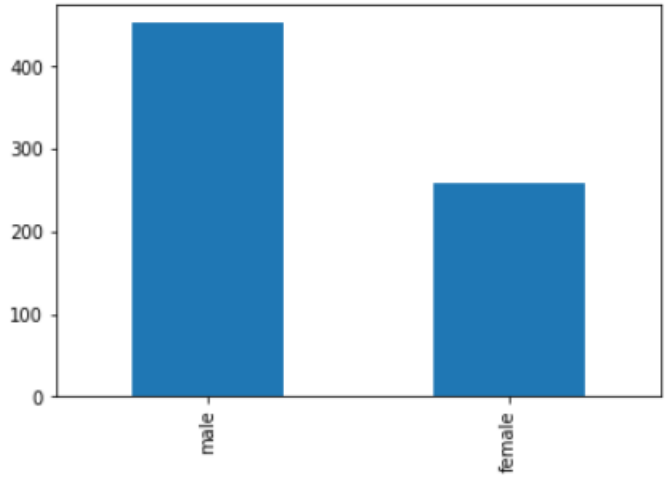



```
count    712.000000
mean       0.432584
std        0.854181
min        0.000000
25%        0.000000
50%        0.000000
75%        1.000000
max        6.000000
Name: Parch, dtype: float64
```

- Sachant C = Cherbourg, Q = Queenstown, S = Southampton ; nous pouvons affirmer que la majorité des passagers ont embarqué à Southampton.



- Il y a près de deux fois plus d'homme que de femme



Nous avons aussi pu étudier les relations entre les attributs grace à la matrice de corrélation.

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	zscore
PassengerId	1.000000	0.029526	-0.035609	0.033681	-0.082704	-0.011672	0.009655	-0.034679
Survived	0.029526	1.000000	-0.356462	-0.082446	-0.015523	0.095265	0.266100	0.190850
Pclass	-0.035609	-0.356462	1.000000	-0.365902	0.065187	0.023666	-0.552893	-0.159166
Age	0.033681	-0.082446	-0.365902	1.000000	-0.307351	-0.187896	0.093143	-0.087336
SibSp	-0.082704	-0.015523	0.065187	-0.307351	1.000000	0.383338	0.139860	0.563231
Parch	-0.011672	0.095265	0.023666	-0.187896	0.383338	1.000000	0.206624	0.652539
Fare	0.009655	0.266100	-0.552893	0.093143	0.139860	0.206624	1.000000	0.620600
zscore	-0.034679	0.190850	-0.159166	-0.087336	0.563231	0.652539	0.620600	1.000000

Ce tableau nous montre que la survie dépendait le plus de la classe et du prix du ticket, que la classe était lié à l'age et au prix, que l'age avait un lien fort avec la présence de familles à bord et que lorsqu'on a de la famille c'est souvent en grand nombre (avec parents/enfants/frère/époux)

Sur ce tableau nous voyons aussi une colonne zscore qui représente notre analyse des outliers. On voit ici qu'ils sont fortement liés au prix du ticket et à la présence de familles.

1.2 Choix des algorithmes et leurs hyperparamètres

Venons en maintenant à la partie du machine learning. Pour notre modèle de machine learning nous avons décidé d'utiliser l'algorithme de regression logistique issue de la bibliothèque "sklearn.linear_model". C'est un algorithme très souvent utilisé dans des cas de classifications. C'était donc adéquat à notre problème étant donné que nous devions déterminer si une personne avait survécu ou non en fonction de ses caractéristiques.

Pour le choix des hyperparamètres, nous avons ajusté l'hyperparamètre "max_iter" qui représente le nombre maximum d'itération pour que le résolveur converge. Il pouvait arriver que notre modèle ait du mal à converger avec notre input donc nous avons augmenté "max_iter" à 1000. Cela a permis d'augmenter la régularité de notre modèle en évitant des valeurs qui pouvait parfois s'éloigner des 80% de précision.

1.3 Résultats

Dans notre cas nous avons décidé de séparer notre jeu de donné avec 80% pour l'entrainement et 20% pour le test. Avec cela nous obtenons un score qui varie autour des 80% de prédictions correctes.

2 Non Supervisé

2.1 Dataset

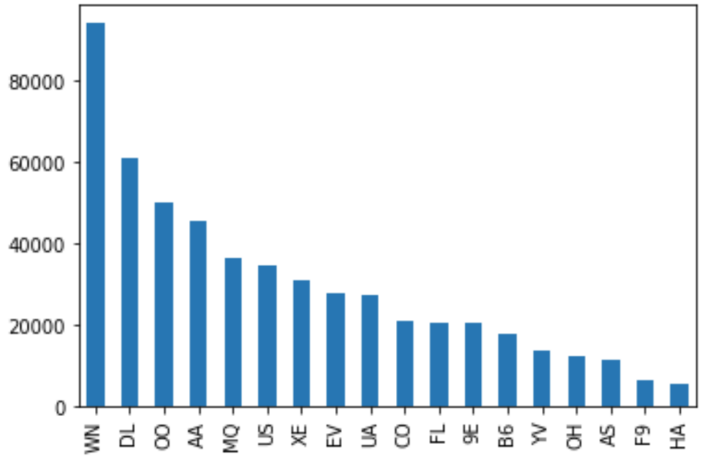
Le jeu de donnée qui comprend 539383 instances et 8 caractéristiques différentes. Le but de ce jeu de donnée est de prédire s'il y a un retard sur un vol ou non. Ainsi, l'objectif derrière le choix de ce jeu de donnée était de voir s'il était possible de passer par du non supervisé pour déterminer si un vol va être retardé ou non. Ainsi, à travers cela, on aimerait trouver deux groupes ("cluster") dont l'un correspondrait au retard et l'autre aux vols qui arrivent dans les temps.

Ce jeu de donnée provient de "Data Expo competition". Les auteurs sont Albert Bifet, Elena Ikonomovska. Il a également été récupéré sur Kaggle.

Ce jeu de données est composées de neuf colonnes.

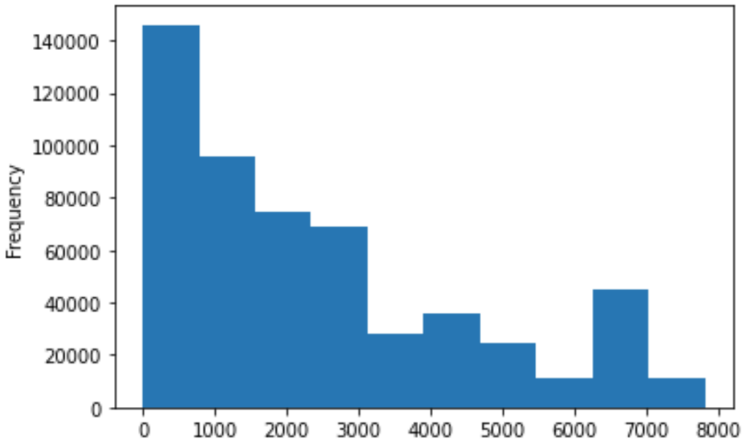
— id
— Companie aérienne "*Airline*" : Type de companie aérienne commerciale
— Vol "*Flight*" : Type d'avion
— Aéroport de départ "*AirportFrom*"
— Aéroport d'arrivée "*AirportTo*"
— Jour de la semaine "*DayOfWeek*"
— Temps pris en minutes "*Time*"
— Durée de vol en minutes "*Length*"
— Retard : "*Delay*"

2.1.1 Companie aérienne



On peut voir sur ce graphique que WN (Southwest Airlines) est la companie la mieux représentée dans ce jeu de données. A l'opposé HA (Hawaiian Airlines) est la moins utilisée. Cela semble plutôt cohérent. Cela pourrait être intéressant d'inclure cette colonne l'entrainement. En effet, la companie peut très souvent jouer sur le retard d'un avion.

2.1.2 Type d'avion



Sur les 539383 instances présentent dans ce jeu de données, il est répertorié environ 6585 types d'avions. Il pourrait être intéressant d'ajouter cette catégorie car la maintenance d'un avion peut varier d'un avion à un autre. Cela pourrait laisser entendre qu'il s'agirait d'un indicateur supplémentaire pour indiquer le retard d'un avion. Toutefois, cette donnée est de nature qualitative. Cela augmenterait donc beaucoup trop le nombre de dimensions d'ajouter une telle colonne.

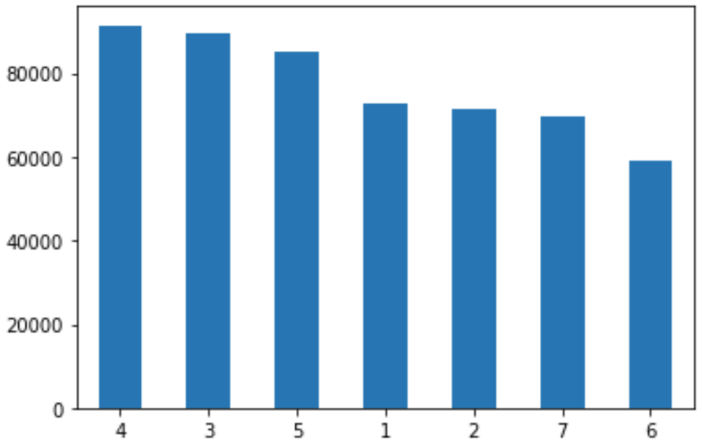
2.1.3 Aéroport de départ

Cette colonne contient 293 catégories. Encore une fois, si on numérise cette colonne, le nombre de dimension risque d'être trop élevé. Cependant, on peut se douter qu'un retard peut être corrélé à l'endroit depuis lequel un avion décolle. Si, par exemple, l'aéroport est trop petit pour un traffic élevé, il se peut que la plupart des avions mettent beacoup plus de temps à en décoller que prévu. cela peut également être dû au personnel de l'aéroport ou à de nombreux autres facteurs. Quoi qu'il en soit cela peut également être un bon indicateur.

2.1.4 Aéroport d'arrivée

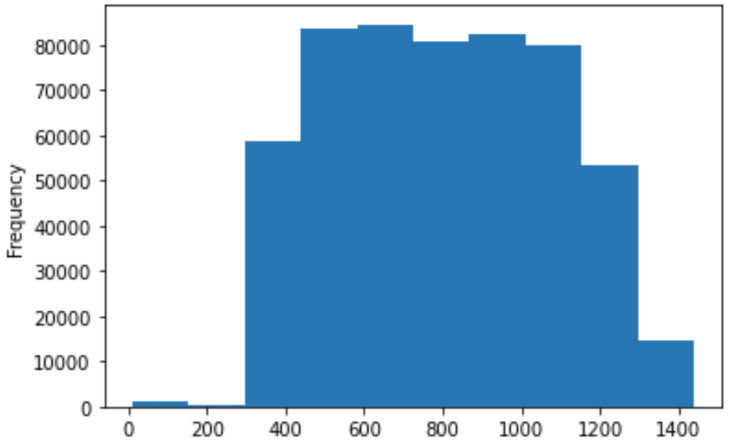
Cette colonne contient également 293 catégorie. Il s'agit également de la même réflexion que pour la partie précédente à la seule différence, qu'ici on parlera plus d'autorisation à attérir dans les raisons récurrentes pouvant engendrer un retard.

2.1.5 Jour de la semaine



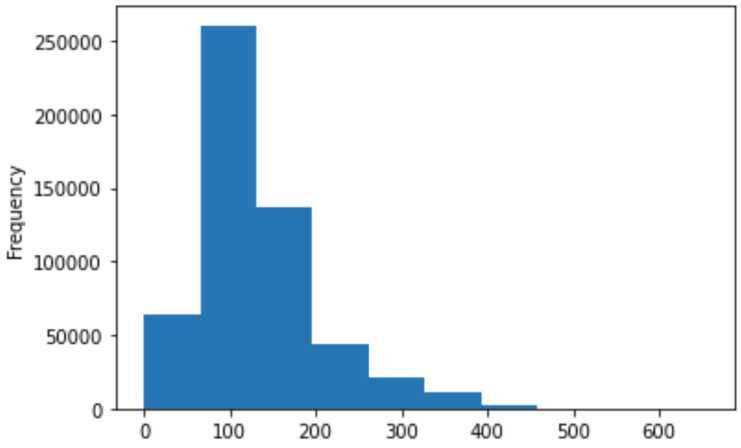
Cette colonne est un peu plus importante que les deux précédentes puisqu'on peut se douter qu'en week-end le traffic est plus élevé qu'en semaine retardant ainsi un certain nombre d'avions. Toutefois on remarque que dans ce jeu de données la répartition à l'air plutôt uniforme sur tous les jours. Ainsi, on peut supposer qu'une telle colonne de fera pas ressortir nos deux catégories voulues.

2.1.6 Temps pris



On peut observer que ce qu'ils appellent le temps pris (*Time taken* oscille entre 0 et 1440. Si on convert 1440 en heure, on se rend compte que cela correspond à 24 heures. Cependant, le sens de cette colonne reste toutefois un peu incomprenhisible.

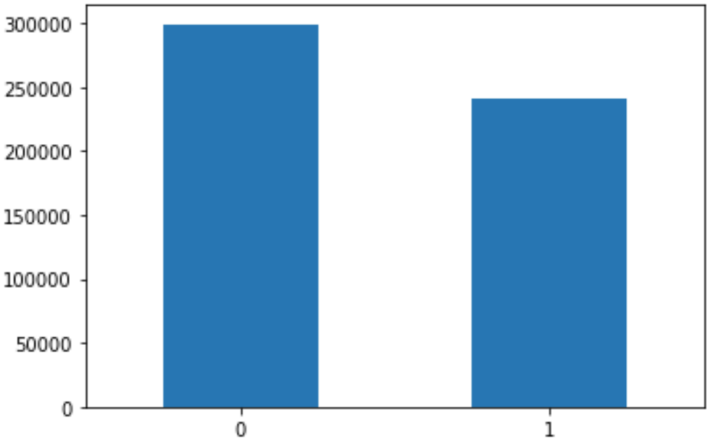
2.1.7 Longueur du vol



Cette colonne semble correspondre à la durée du vol. En effet, les valeurs sont incluses entre 0 et 655. Or 655 représente environ 10 heures en minutes. S'il on se renseigne sur internet, on se rend compte qu'il faut un peu plus de 5 à 6 heures (soit entre 300 à 360 minutes) pour aller d'une côte à une autre des Etats-Unis. Le seul voyage qui prends plus de 10 heures est celui entre Hawai et la côte Ouest/Est des Etats-Unis. Or d'après le première graphique Hawai Airlines est la companie la moins représentée dans ce jeu de données. Cela semble coincider avec ce graphique sur la longeur des vols.

De plus, la longeur d'un vol pourrait impacter le retard d'un avion. Par exemple, les vols longs demanderaient plus de vérifications avant de partir. Ils sont également soumis à plus de facteurs exgtérieurs aléatoires pendant leur trajet étant donné qu'ils sont plus longs.

2.1.8 Retard



Enfin la dernière colonne nous dit si un vol a été en retard ou non. C'est une des raisons pour lesquelles ce jeu de données a été choisit. Cela nous permettra de dire si notre algorithme a réussi à repérer 2 catégories corrélées avec les retard et si la répartition est un peu près la même.

2.1.9 Corrélation

	Flight	DayOfWeek	Time	Length	Delay
Flight	1.000000	0.000416	-0.005750	-0.341481	-0.046175
DayOfWeek	0.000416	1.000000	0.001273	0.013397	-0.026199
Time	-0.005750	0.001273	1.000000	-0.020612	0.150454
Length	-0.341481	0.013397	-0.020612	1.000000	0.040489
Delay	-0.046175	-0.026199	0.150454	0.040489	1.000000

On peut voir ici, qu'une variable ne semble être corrélées avec le retard. Cela n'est pas forcement bon signe puisqu'aucune ne permettra à notre algorithme de bien séparer deux catégories dépendamment des retard. En revanche ce tableau peut nous permettre de mieux expliquer les catégories que nous obtiendrons à la fin.

2.2 Choix des algorithmes et leurs hyperparamètres

2.2.1 Choix de l'algorithme et son fonctionnement

L'algorithme qui semble le plus approprié à notre cas de figure est l'agorithme Kmeans. Le principe de l'algorithme est simple. Il cherche "n" groupes où "n" représente le nombre de groupe spécifié.

Fonctionnement de l'algorithme :

1. L'algorithme commence par sélectionner deux points au hasard. Ces points correspondent aux centres de chaque groupe.
2. Puis pour chaque autre point, il cherche le centre le plus proche afin de les associer à un groupe.
3. Une fois les groupes formés, de nouveaux centres sont recalculés via le barycentre de chaque groupe.
4. Puis, on reconstruit des groupes à travers ces nouveaux centres.
5. On réitère l'opération jusqu'à ce que les centres ne bougent plus ou que le nombre d'itérations maximales soit excédé.

Remarque : Il est important de préciser qu'en fonction des centres de départ choisit on peut tomber sur de meilleurs ou de moins bons groupes à l'arrivée. Quoi qu'il en soit le résultat a de fortes chances de ne pas être le même.

Dans notre code, nous avons décidé d'utiliser le **sklearn.cluster.KMeans** provenant de la librairie Python "Scikit Learn".

2.2.2 Choix des hyperparamètres

random_state Le choix des groupes pouvant ne pas être les mêmes d'un appel à un autre, ce paramètre nous permet de choisir une "seed" afin que l'on puisse retrouver la même génération de groupes d'appel en appel. En effet, ce dernier permet de générer les mêmes centres de départ quoi qu'il advienne. Nous initialiserons ce dernier à **42** pour la référence.

n_clusters Ce paramètre permet de sélectionner le nombre de groupe que l'on souhaite. Dans notre cas, cela parait assez évident. On souhaite obtenir deux groupes. On initialise donc ce paramètre à **2**.

n_init Ce paramètre sert à initialiser les premiers centres. Ici, nous préférons laisser ce choix à Scikit Learn de trouver les centres de départ les plus optimaux. Nous n'initilisons donc pas ce paramètre.

n$_i nit$ Nombre de fois que l'algorithme k-means sera exécuté avec différentes graines ("seed") de centroïdes ("les points centraux sélectionnésé). Le résultat final sera la meilleure sortie de n_init exécutions consécutives. Nous avons initilisé ce paramètre à taton en fonction des résultats que l'on obtenait.

max_iter Il s'agit du nombre d'itérations maximums possible pour chaque période. Cette valeur a aussi été sélectionné en fonction des résultats.

2.3 Résultats

On observe dans les résultats ci-dessous que les meilleurs résultats sont obetnues bien evidemment lorsque l'on spécifie le résultat en entrée (colonne **Delay**). Toutefois, dès qu'on la retire les résultats sont très faibles. Les retard et les groupes trouvés ne semblent pas avoir beaucoup de rapport. Toutefois, les meilleurs résultats ont été trouvé dans un premier temps avec uniquement des valeurs numériques. Puis dans un second temps, étrangement, en retirant uniquement les "companies aériennes". Enfin, dès qu'on tente d'ajouter les types d'avions, le nombre de catégories étant tellement énorme, cela fait planter le noyau (sans surprise).

Airline	Flight	AirportTo	AirportFrom	DayOfWeek	Time	Len	Delay	Accuracy
							X	1.0
					X		X	1.0
					X	X	X	1.0
				X	X	X	X	1.0
X				X	X	X	X	0.0
X			X	X	X	X	X	1.0
					X			0.431
					X	X		0.5687
				X	X	X		0.462
X				X	X	X		0.462
X			X	X	X	X		0.462
			X	X	X	X		0.538
		X	X	X	X	X		0.538
	X	X	X	X	X	X		Crash

2.4 Conclusion

En conclusion, d'après nos résultats, il est très peu concluant de chercher à trouver des retard à travers du non supervisé sur ce jeu de données. Cela est sans doute dû au nombre trop important de catégorie.