# RAPPORT CHALLENGE INSA


**Groupe :**

DIALLO Aicha Aminata

GAHN Alioune Badara Ba

NIANG Khaly

SECK Mamadou Moustapha

**Encadrants:**

PAUWELS Edouard

SERRURIER Mathieu

# I. PRÉSENTATION

## a. Le challenge

Le Défi IA est compétition de machine learning regroupant des étudiants venant de plusieurs formations. Il est organisé tous les ans par l'Institut National des Sciences Appliquées. Le thème qui nous a été proposé cette année par Airbus est la détection d'anomalies sur des séries temporelles issues des capteurs d'hélicoptères.

La détection d'anomalies est devenue aujourd'hui un grand défi pour les entreprises. De l'industrie alimentaire aux assurances, chaque entreprise a besoin d'utiliser les nouvelles méthodes d'intelligence artificielle pour prédire et éviter les incidents. C'est dans ce contexte que nous avons contribué dans cette compétition qui avait pour objectif de relever des signaux anormaux sur un ensemble de données de séries chronologiques obtenu à partir des capteurs des hélicoptères d'Airbus.

Dans les lignes suivantes, nous décrirons d'abord notre ensemble de données, puis nous approfondirons notre méthodologie en explicitant les méthodes utilisées et nous terminerons par une discussion des résultats que nous avons obtenus.

## b. L'équipe et l'organisation

Notre équipe DragonFC était composée de quatre personnes. Tout au long du projet, nous avons échangé et avons travaillé en parallèle sur différents méthodes de détection d'anomalies. Notre objectif étant de pouvoir tester chacun de ces méthodes afin d'obtenir une diversité des résultats et de choisir la meilleure méthode de prédiction.

Concernant la communication intra groupe, nous avons utilisé plus utilisé Slack qui nous a permis d'échanger des bouts de code et des liens utiles. Des réunions hebdomadaires avaient également été mis en place pour suivre l'avancé de chacun des membres.

## c. Les données

Lors de cette compétition, nous avons eu à disposition dans un premier temps deux jeux

de données: les données d'entraînement et les données de validation. Les données de test n'ont été disponibles que dans la dernière partie du projet pour appliquer le modèle qui a été choisi.

Ces différents données sont des séquences d'une minute d'accéléromètre à une fréquence de 1024 Hertz. Une séquence correspond à 61440 variables.

Nos données d'entraînement sont composées de 1677 séquences. Ces séquences ne contiennent aucune anomalie et sont donc supposées normales. A partir de nos données d'entraînement, l'objectif est de trouver un modèle capable de trouver sur un autre jeu données les séquences anormales. On est donc confronté à une problématique de détection de nouveauté.

Les données de validation sont composées de 594 séquences. Dans ce jeu de données, nous ne disposons pas du taux de séquences présentant des anomalies. La difficulté étant de trouver ces séquences anormales en les comparant avec les séquences des données d'entraînement et ainsi créer notre modèle.

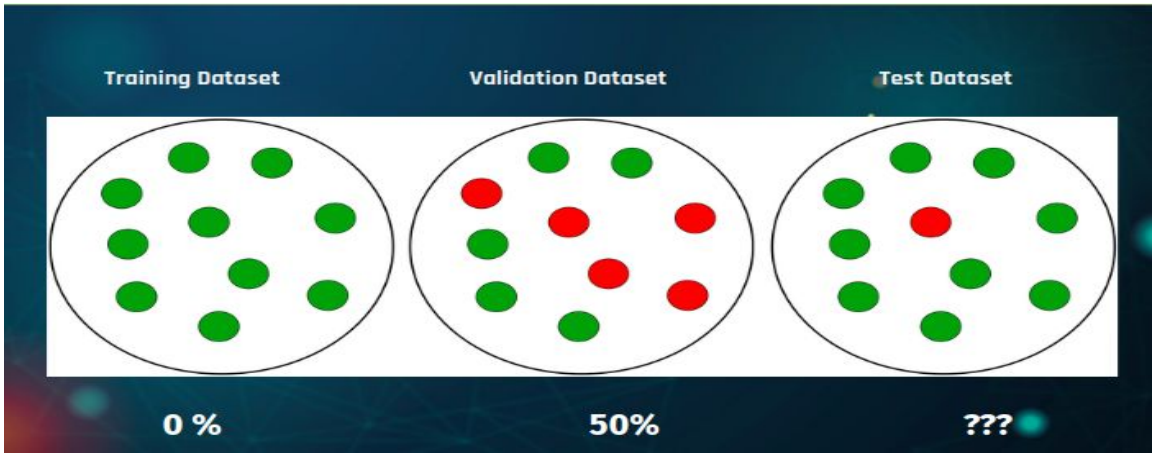



***fig1: composition de notre dataset***

# II. DÉMARCHE ET DÉVELOPPEMENT

## 1. Pré-processing

Dans cette phase, nous décrivons tous les traitements qui ont été effectuées sur les données avant l'utilisation de modèles statistiques, l'objectif de nos méthodes étant de réduire la dimension de notre espace de features afin de permettre à nos algorithmes de mieux fonctionner. Nous avons opté pour cela trois approches:

- Création manuelle de Features statistiques.
- Réduction de dimension grâce à un auto-encoder
- Réduction de dimension grâce à une ACP.

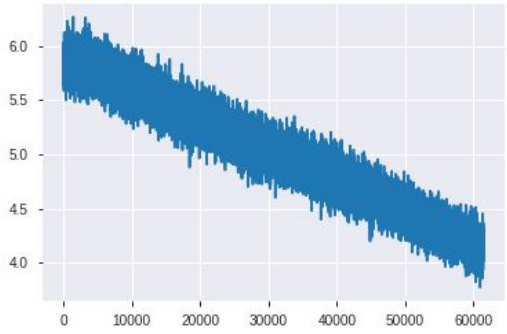
### a. Création manuelle de Features Statistiques:

C'est une des parties les plus importantes de notre démarche car, c'est là ou nous nous sommes le plus concentré. Une première analyse descriptive et visualisation des données nous a montré que toutes les observations de notre échantillon d'entraînement qui n'était composé que de signaux normaux ont des valeurs comprises -5 et 5. A partir de cette remarque, nous avons cherché à trouver des indicateurs statistiques pertinents qui nous permettraient de définir ce qu'était un signal normal ou anormal.

C'est dans cette démarche que nous avons créé un ensemble de features statistiques (27), dont nous vous présentons les plus intéressants ci dessous. (Voir annexe pour plus de détails sur les features)

- Valeur absolue de la corrélation avec le temps:

Pour chaque signal, nous mesurons la valeur absolue de la corrélation du vecteur avec le temps. L'intuition derrière ce feature est que quand la valeur absolue de la corrélation avec le temps est grande, on aura une grande pente pour le vecteur directeur associé au signal. Sachant qu'on a des séries temporelles avec une fréquence de 1200 Hz, on est donc censé se rapprocher d'un signal périodique. Donc plus la pente est importante, plus on est en présence d'anomalies. Les figures suivantes illustrent bien ce feature:



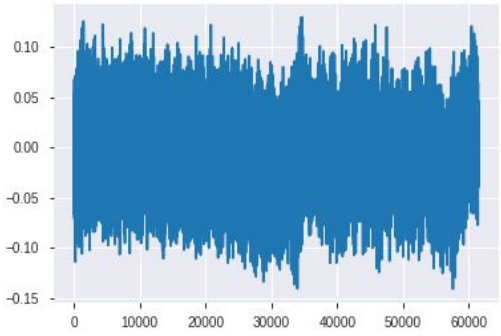




***fig2:corrélation grande: anomalie*** ***fig3:corrélation faible: normale***

- Maximum, Moyenne, Ecart-type des valeurs:

Ces Features ont été calculés naturellement par la suite de la phase de visualisation des données. Nous avons remarqué que toutes les moyennes des observations de notre échantillon d'entraînement sont centrées autours d'une valeur proche de zéro et que toutes les valeurs étaient comprises entre -5 et 5. L'intuition derrière ces features est: toutes les observations dont on a des valeurs supérieures à un certain seuil seraient des anomalies.

- Distance au barycentre des observations normales:

Ce feature se déduit implicitement des moyennes et écart-types calculés précédemment. Après avoir calculé pour chaque observation la moyenne ainsi que l'écart type des coordonnées du vecteur de dimension 61 440, nous obtenons un représentation en deux dimensions (moyenne et écart type) des nos données. Nous calculons ainsi pour uniquement les observations de notre échantillon d'entraînement constitué uniquement de signaux normaux le centre de gravité ou barycentre G. Ainsi, pour toutes les observations du train et validation set, nous calculons la distance de chaque point au barycentre (par rapport à la moyenne et l'écart type).

L'intuition derrière ce feature est: plus un signal s'éloigne du barycentre obtenu en ne considérant que les observations normaux, plus il est susceptible d'être une anomalie.

Les figures suivantes nous permettent de mieux visualiser ce feature:

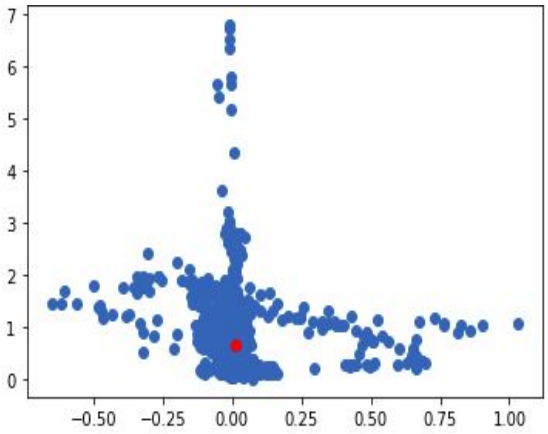



*fig4:données train*

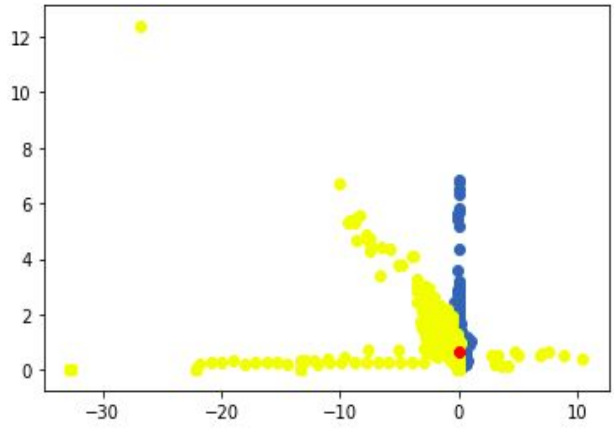



*fig5: données validation + train*

Les figures ci-dessous sont obtenues avec les données d'entraînement (bleu) et les données de validation (jaune). Nous voyons le centre de gravité des données normales(rouge) qui est assez éloignée de certaines observations de nos données de validation. L'intuition derrière ce feature serait de dire que ceux la sont des anomalies.

- Energie du signal:

Un autre feature qu'on a construit durant ce challenge est la mesure de l'énergie de chaque observation:

$$E_S = \int_{-\infty}^{+\infty} |x(t)|^2 dt$$

Ce feature était très corrélé avec le feature Maximum car plus une observation a des valeurs de cordonnées grande, plus l'énergie associée sera importante. C'est pour cela que nous nous en sommes pas servis dans la construction du modèle.

- Score outlier:

Pour chaque observation, nous calculons un feature scoreOutlier désignant le nombre de valeurs supérieures à la moyenne des coordonnées plus 6 écart types, normalisé par le range de ces valeurs.

Ainsi, pour un signal, si on a plusieurs coordonnées au dessus d'un seuil dans un même cluster,

cela sera interprété comme une turbulence donc une anomalie( cas de la figure suivante).

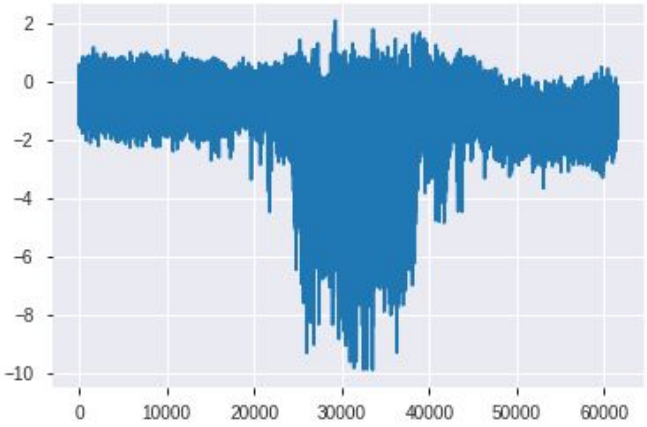



***fig6: Illustration score outlier***

C'est un feature qui a été très important durant ce projet.

- Range de moyennes par fenêtre:

Voici un autre feature qui a été très intéressant durant le challenge. Sachant que chaque signal représente un enregistrement d'une minute sous une fréquence de 1200Hz, nous avons créé 60 fenêtres (61 440 / 1200) afin d'observer ce qui se passait à chaque seconde. Ainsi, nous calculons pour chacune des 60 fenêtres de chaque observation une moyenne des coordonnées. On se retrouve donc avec un vecteur de 60 valeurs de moyennes pour chaque observation. Et notre feature sera le range de ce vecteur (moyenne max - moyenne min).

L'intuition de ce feature est: si on a des différences de moyennes trop élevées alors nous avons une potentielle anomalie.

Les figures 7 et 8 ci dessous illustrent bien ce feature:

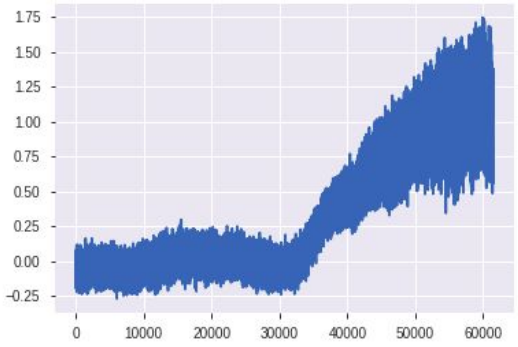



*fig7: grand range : anomalie*

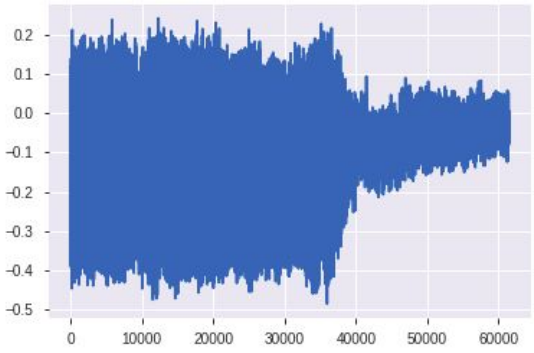



*fig8: petit range: normal*

- Similarité avec le signal moyen:

Pour obtenir ce feature, nous calculons un signal normal moyen sur l'ensemble du training set en faisant la moyenne de toutes les observations colonne par colonne. Une fois qu'on a ce signal moyen qui est censé représenter par excellence un signal normal, nous calculons un score de similarité de ce signal avec toutes les observations de notre dataset. Un signal pour lequel on a une score de similarité proche de 1 sera considéré comme normal et 0 comme anormal.

La mesure de similarité utilisée ici est la similarité cosinus:

$$\cos\theta = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

- Similarités avec le vecteur d'auto-corrélation moyen.

Pour mettre en place feature, nous procédons d'abord une transformation de chaque observation par un vecteur d'autocorrélation permettant de détecter des régularités, des profils répétés dans un signal comme un signal périodique perturbé par beaucoup de bruit, ou bien une fréquence fondamentale d'un signal qui ne contient pas effectivement cette fondamentale, mais l'implique avec plusieurs de ses harmoniques.

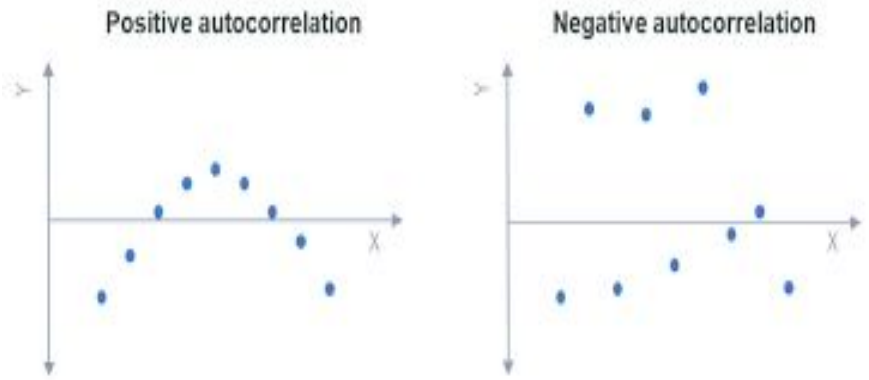



*fig9: Autocorrélation*

Aprés avoir obtenu un vecteur d'autocorrélation pour chaque observation, nous mesurons la similarité avec un vecteur normal moyen par le même procédé que précédemment.

- Balancement des valeurs:

C'est le dernier feature que nous présentons dans ce rapport technique. Il nous permet de mesurer le balancement de chaque observation. Nous obtenons ce feature en faisant la différence pour chaque observation entre le nombre de valeurs au dessus de la moyenne et le nombre de valeurs en dessous de la moyenne. Nous considérons ce feature car nous avons vu durant la phase de visualisation que cette différence est trés petites pour les observations normales du train set.

L'illustration de ce feature peut se faire à l'aide des figures suivantes:

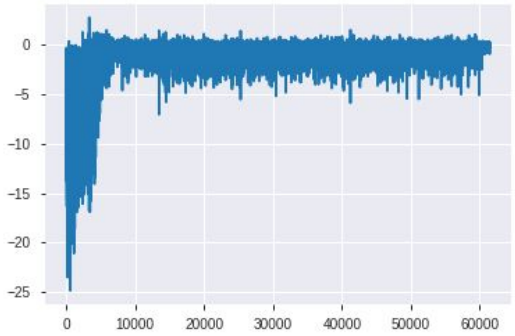



*fig 10: grand écart: anomalie*

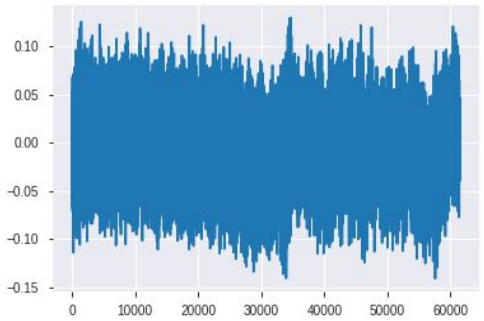



*fig 11: écart faible: normal*

### b. Réduction de dimension: Auto- encoder

Dans cette partie, nous utilisons la technique auto-encoders pour réduire la dimension. Les auto-encoders sont des réseaux de neurones à deux composantes: un encodeur et un décodeur. Le réseau de neurones est conçu pour compresser les données grâce à la composante encodeur. Le décodeur tentera de décompresser les données et de retrouver la représentation initiale.

la technique auto-encoder est utilisé pour synthétiser au mieux l'information pertinente. Elle fonctionne de façon un peu similaire à une analyse en composante principale.

Le réseau est paramétré de façon que l'erreur de reconstruction de notre signal d'entrée soit minimale.

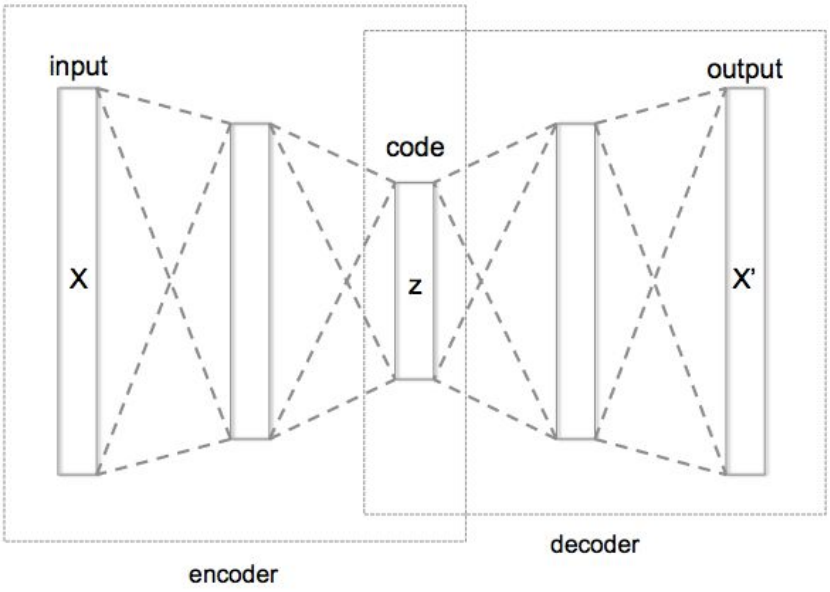



*fig 12: auto-encoder*

La configuration ci-dessous (encoder) est celle qui minimise notre erreur de reconstruction. En ne considérant que la représentation obtenue avec la couche du milieu, elle nous permet de passer de 61441 colonnes à 200 colonnes. Après avoir réduit la dimension, nous centrons et réduisons le résultat afin d'utiliser un algorithme de détection d'anomalies sur le résultat des données centrées et réduites.

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 61441)             0
_________________________________________________________________
dense_1 (Dense)              (None, 14360)             882307120
_________________________________________________________________
dense_2 (Dense)              (None, 7680)              110292480
_________________________________________________________________
dense_3 (Dense)              (None, 5000)              38405000
_________________________________________________________________
dense_4 (Dense)              (None, 2250)              11252250
_________________________________________________________________
dense_5 (Dense)              (None, 2000)              4502000
_________________________________________________________________
dense_6 (Dense)              (None, 1750)              3501750
_________________________________________________________________
dense_7 (Dense)              (None, 1500)              2626500
_________________________________________________________________
dense_8 (Dense)              (None, 1250)              1876250
_________________________________________________________________
dense_9 (Dense)              (None, 1000)              1251000
_________________________________________________________________
dense_10 (Dense)             (None, 750)               750750
_________________________________________________________________
dense_11 (Dense)             (None, 500)               375500
_________________________________________________________________
dense_12 (Dense)             (None, 250)               125250
_________________________________________________________________
dense_13 (Dense)             (None, 200)               50200
=================================================================
Total params: 1,057,316,050
Trainable params: 1,057,316,050
Non-trainable params: 0
```

***fig13: Architecture de notre encoder***

### c. Réduction de dimension: Analyse en composantes principales (ACP)

Compte tenu du nombre important de features qu'on a à notre disposition, il s'est avéré important de faire une réduction de dimension à travers une ACP, comme nous l'avons fait dans la partie de l'Auto-encoder.

Initialement les jeu de données d'entraînement et de validation avaient une taille respective de 1677 lignes et 61440 colonnes contre 594 lignes et 61440 colonnes pour le jeu de données de validation. Ainsi, après réduction des dimensions à travers cette méthode, nous avons passé de 61440 à 191 features.

Pour des raisons d'affichage on a pas pu représenter les individus sur le plan factoriel pour voir comment les individus sont répartis dans le plan principal.

```
[2573.43434516 2322.92823295 1937.62511244 1641.69143799 1605.18516159
 1572.93503002 1369.6546267  1304.74607123 1212.52371497 1128.11142609
  849.52229043  700.99343561  598.21945843  573.21151373  551.23247164
  515.69792365  510.60627462  495.3561018   488.30815036  481.34079887
  478.52105046  465.38597082  433.3840265   415.48196753  409.57873208
  407.72969407  398.22614333  377.89313737  366.42984206  361.32177532
  356.8636498   349.6605613   342.29023596  335.16081124  330.95520314
  327.28240424  318.07840237  314.5072114   305.3519029   302.43360868
  297.9425473   293.99802855  291.03721521  284.75730875  276.69933553
  271.3487213   261.53297251  257.15188555  251.68171894  246.00272742
  243.6108045   242.03517191  235.81741417  230.0623284   221.31068761
  216.98822622  213.61691294  210.53090695  208.02870536  205.33096975
  201.74351527  199.4959821   197.26236974  196.60214393  190.50785277
  189.96422729  185.6631728   183.56316977  181.44860691  179.5589805
  176.6921003   174.76556901  170.82081288  169.7865132   168.6998161
  165.72366362  165.03685162  162.89800188  157.6967772   155.94770319
  152.51774831  151.70199652  150.61975091  146.72033664  144.66246269
  144.5199452   143.29480829  141.34456734  140.90523401  140.41621496
  138.25107024  137.03885391  135.36040348  135.32957198  132.91511975
  131.40138759  130.64324479  130.33474827  127.84021648  126.92709798
  125.86284143  124.52581893  123.32157365  122.33089711  121.0948056
  119.7189396   119.13866413  118.68309564  117.66476128  115.97396873
  115.41739833  114.00561566  113.13627756  112.13324735  110.45637785
```

***fig14:Valeurs propres obtenu après réduction des dimensions.***

```
[0.04188532 0.03780808 0.03153687 0.02672024 0.02612606 0.02560116
 0.02229256 0.0212361  0.01973509 0.01836119 0.01382686 0.0114094
 0.00973664 0.00932961 0.00897188 0.00839352 0.00831065 0.00806244
 0.00794772 0.00783432 0.00778843 0.00757464 0.00705378 0.0067624
 0.00666632 0.00663623 0.00648155 0.0061506  0.00596403 0.00588089
 0.00580833 0.00569109 0.00557113 0.00545509 0.00538664 0.00532686
 0.00517706 0.00511893 0.00496992 0.00492242 0.00484933 0.00478512
 0.00473693 0.00463472 0.00450357 0.00441648 0.00425672 0.00418541
 0.00409638 0.00400395 0.00396502 0.00393937 0.00383817 0.0037445
 0.00360206 0.00353171 0.00347684 0.00342661 0.00338588 0.00334198
 0.00328359 0.003247   0.00321065 0.0031999  0.00310071 0.00309187
 0.00302186 0.00298768 0.00295327 0.00292251 0.00287585 0.00284449
 0.00278029 0.00276345 0.00274577 0.00269733 0.00268615 0.00265133
 0.00256668 0.00253821 0.00248239 0.00246911 0.00245149 0.00238803
 0.00235453 0.00235221 0.00233227 0.00230053 0.00229338 0.00228542
 0.00225018 0.00223045 0.00220313 0.00220263 0.00216333 0.00213869
 0.00212635 0.00212133 0.00208073 0.00206587 0.00204855 0.00202679
 0.00200719 0.00199106 0.00197094 0.00194855 0.00193911 0.00193169
 0.00191512 0.0018876  0.00187854 0.00185556 0.00184141 0.00182509
 0.00179779 0.00178629 0.00177448 0.00176399 0.00174912 0.00173754
 0.00173136 0.001709   0.00169523 0.00169477 0.00168244 0.00167718
 0.00166429 0.00165207 0.00162577 0.00160942 0.00159634 0.00158257
 0.00157931 0.00156259 0.00155553 0.00153411 0.00152582 0.00150351
```

***Fig15: Variance expliquée dans les dimensions***

Afin d'avoir le nombre de dimension adéquat pour représenter nos données, nous choisissons les composantes principales qui contribuent le plus à l'inertie. Le graphe suivant nous montre qu'il serait judicieux de choisir 191 dimensions car c'est ce qui nous donne un pourcentage de variance expliquée avoisinant les 80%.

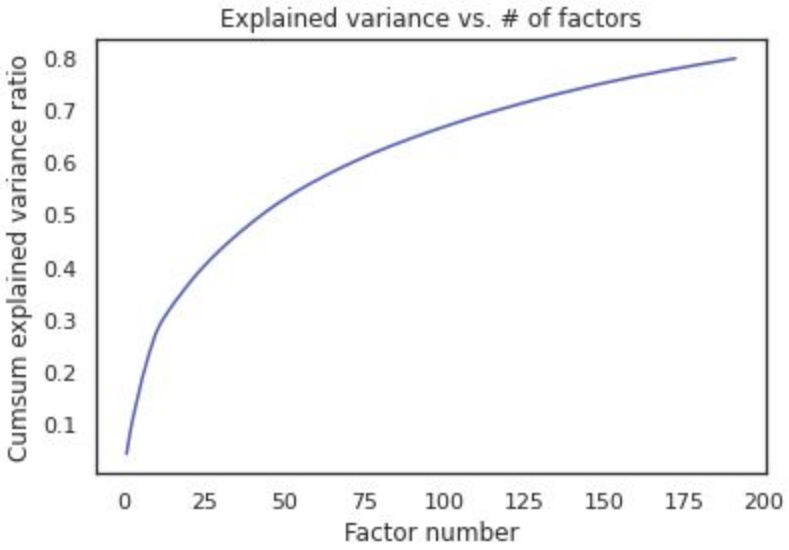



***fig16: Variance expliquée en fonction de la dimension***

## 2. Détection d'anomalies

Une fois que nous avons nos données bien pré-traitées et les dimensions réduites, nous pouvons maintenant passer à la phase de construction de modèles de machine learning afin de détecter les anomalies dans notre dataset. Pour ce faire, nous avons adopté une démarche de novelty detection induite par les données dont nous disposions en testant 3 algorithmes différents:

- Kernel Density Estimation
- Isolation Forest
- Vote de nos 3 meilleurs Features

L'intérêt de ces méthodes par rapport à notre jeu de données est que le score d'anomalie est obtenu sur la base d'un calcul de distance sachant que lorsque notre espace de feature est de grande dimension, les algorithmes à base de distance ne marchent pas. D'ou l'intérêt d'appliquer ces algorithmes de machine learning sur un dataset de dimension réduite obtenue grâce à la phase précédente.

### a. Kernel Density Estimation

C'est une méthode permettant de trouver une fonction de densité de probabilité estimée à partir de notre échantillon d'apprentissage. L'estimation tente d'inférer les caractéristiques de la population, sur la base d'un ensemble de données obtenues à la phase de réduction de dimension.

Ce qui permet d'assimiler ce qu'est une observation normale et de donner un score d'anomalie à une observation en comparant sa densité avec celle apprise durant la phase d'entraînement. Dans notre situation, le fine tuning du modèle sur l'hyper paramètre bandwith est effectué grâce à un Gridsearch.

En bref, la technique permet de créer une courbe lisse étant donné un ensemble de données aléatoires.

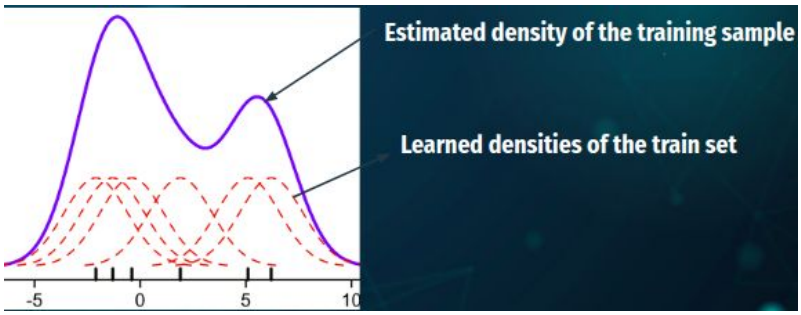



*fig17: illustration KDE*

### b. Isolation Forest

Cet algorithme non supervisé de machine learning permet de détecter des anomalies dans un jeu de données. Il isole les données atypiques, autrement dit celles qui sont trop différentes de la plupart des autres données en calculant, pour chaque donnée du jeu, un score d'anomalie qui reflète à quel point la donnée en question est atypique.

Afin de calculer ce score, l'algorithme isole la donnée en question de manière récursive : il choisit un descripteur et un "seuil de coupure" au hasard, puis il évalue si cela permet d'isoler la donnée en question ; si tel est le cas, l'algorithme s'arrête, sinon il choisit un autre descripteur et un autre point de coupure au hasard,

Le nombre k d'arbres utilisés est un des hyper paramètres du modèle. Nous choisissons la valeur optimale de K par Gridsearch.

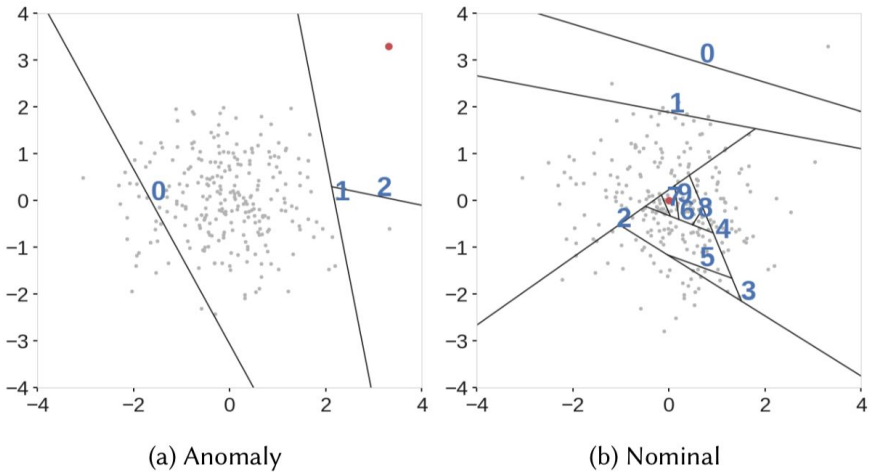



*fig18:Illustration Isolation Forest*

### c. Vote de Features

Cette approche est différente des autres méthodes de machine learning. Ici nous faisons voter nos 3 meilleurs features. En effet, en choisissant pour chacune des features un bon seuil de détection d'anomalies, on obtient en considérant chacune d'elles indépendamment des autres des F1-scores avoisinant 0.93 sur l'échantillon de validation.

Nous décidons donc de les faire voter sans utiliser de modèles de machine learning sur les données. Ceci a eu son intérêt dans notre démarche.

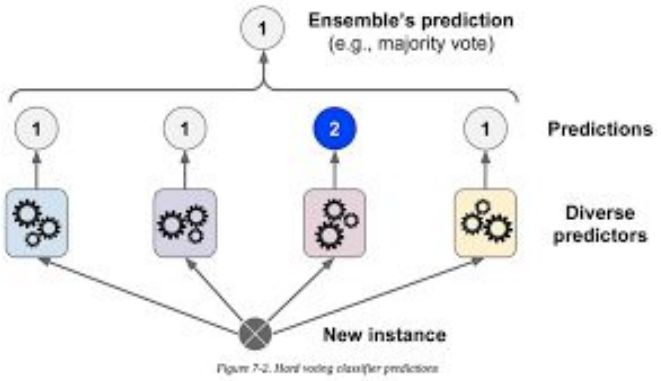



*fig19: Illustration Vote*

## 3. Détermination du seuil de détection d'anomalie:

Afin de déterminer le seuil de détection d'anomalie, nous utilisons l'information selon laquelle il y a 50% d'anomalies dans l'échantillon de validation.

Nous choisissons donc assez naturellement pour chacune des 3 méthodes explicitées précédemment le seuil qui prédit 50% d'anomalies dans notre échantillon de validation.

Nous avons essayé une autre approche consistant à choisir le seuil nous minimisant le nombre d'anomalies dans le training set mais à chaque la première approche donnait le meilleur score sur le leaderboard. C'est donc celui que nous avons utilisé.

## 4. Evaluation et Validation de modèles

Une fois que nous avons construit nos modèles, il est nécessaire d'avoir une méthode d'évaluation et de validation afin de pouvoir se décider sur quoi soumettre durant la dernière phase du Challenge. Pour cela, nous optons pour deux métriques:

- Le F1-score qui s'impose naturellement car étant le critère d'évaluation sur le leaderboard
- Le nombre d'anomalies prédites sur le training set.

a) <u>F1-score:</u>

C'est une moyenne harmonique constituant un bon compromis entre la précision et le rappel.

Il est obtenu grâce à la formule suivante:

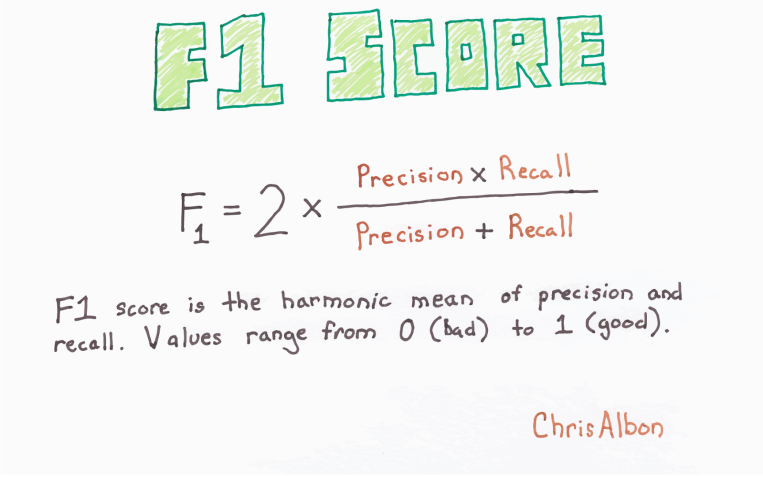



***fig20: F1-score***

b) Pourcentage d'anomalies dans le train:

Le pourcentage d'observations prédites comme anomalies dans le dataset d'entraînement qui n'est constitué que de signaux normaux est un bon indicateur pour déterminer à quel point notre modèle a assimilé ce qu'était une observation normale. Pour la déterminer, nous procédons par cross-validation.

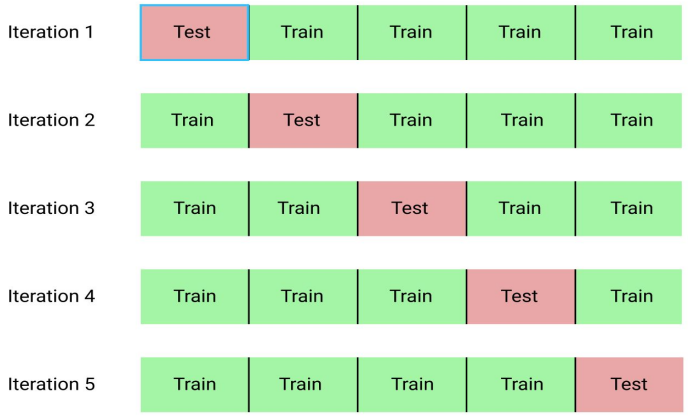



***fig21: Cross-validation***

Nous nous sommes donc basés sur ces deux méthodes pour évaluer et valider nos modèles.

## 3. Analyse des résultats

Le tableau suivant est un récapitulatif des résultats obtenus durant ce challenge:

| pre-processing | model | F1-score | % anomalies train |
|---|---|---|---|
| auto-encoder | KDE | 85 | 7 |
| ACP | Isolation Forest | 89 | 5 |
| Feature statistique | KDE | 96 | 15 |
| Feature statistique | Isolation Forest | 92 | 11 |
| Feature statistique | Feature Voting | 94 | 4 |

***fig22: tableau des résultats***

L'analyse du tableau de résultats ci dessus nous montre qu'il y a une corrélation négative entre le F1-score et le pourcentage d'anomalies prédites dans notre échantillon d'entraînement qui n'est composé que de signaux normaux.

Cependant on observe un meilleur F1-score (96%) avec un KDE sur certains des features statistiques présentés précédemment et un plus faible taux d'anomalies dans notre échantillon d'entraînement avec le vote de nos 3 meilleurs features.

Nous voyons cependant que les autres méthodes de réduction de dimension ( ACP et Auto-encoder) donnent les plus faibles F1-score sur le dataset de validation. Ceci s'explique par le fait que les algos choisis deviennent fragiles en grande dimension (200 pour l'Auto-coder et 191 pour l'ACP). Il faudrait pour ces méthodes changer d'approche (calculer un score d'anomalie en fonction de l'erreur de reconstruction du modèle pour les auto-encoders par exemple) au lieu d'utiliser des algorithmes basés sur la distance.

## 4. Stratégie pour la phase finale

Pour la phase finale, une brève analyse descriptive nous a montré qu'on a pas une uniformité de la distribution des données dans les échantillons de test et de validation. Ceci va nous permettre de bien choisir les features sur lesquelles nous allons construire nos modèles.

Sachant que nous n'avions droit qu'à deux soumissions durant la dernière étape du challenge et en prenant compte des résultats présentés dans la section précédente, nous avons décidé de soumettre les prédictions effectuées par le modèle de KDE et le vote des features car c'est ceux qui nous donnent respectivement le meilleur F1 score sur la partie validation et le plus faible taux d'anomalies obtenu dans l'échantillon d'entraînement.

Ce qui nous a donné sur le dataset de test un F1-score de 32%, une précision de 31% et un rappel de 33%.


***fig23: Résultat final***

# CONCLUSION

Pour conclure, ce challenge sur un problème de détection d'anomalies nous a permis de travailler à la fois sur des données réelles et une problématique rencontrée aujourd'hui pas toutes les entreprises allant de l'industrie au secteur assurantiel.

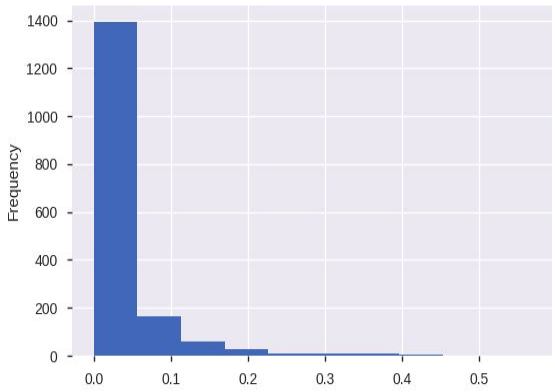
Cela nous a aussi donné l'opportunité de nourrir notre créativité dans la mise en place et le choix de nos features et surtout de pouvoir acquérir une expérience dans le traitement des séries temporelles dans une problématique de machine learning.

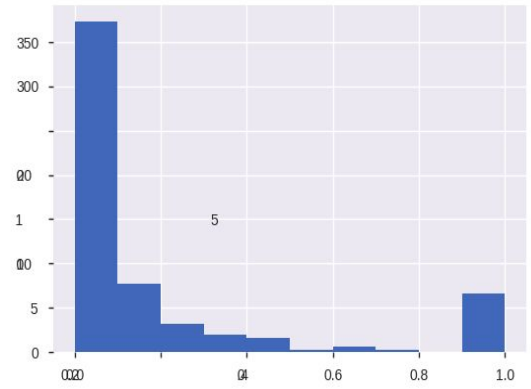
Techniquement parlant, à travers ce challenge, nous avons eu à faire l'inventaire des méthodes et techniques de détection d'outliers même si nous ne les avons pas tous testé.

Nous terminons donc cette compétition par un bilan positif par rapport aux acquis et compétences développés durant ce projet même si nous aurions aimé avoir encore plus de contexte métier pour une meilleure modélisation du problème.

# ANNEXE: Distribution des features selon l'échantillon

- corrélation

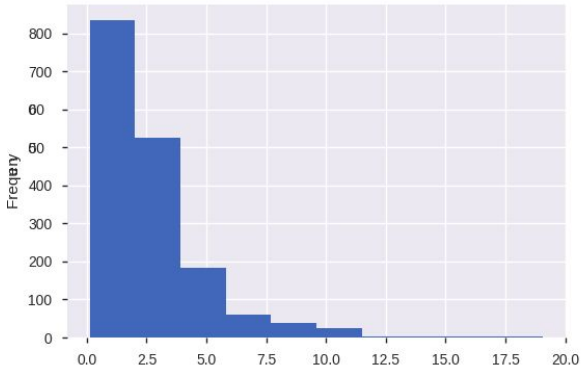


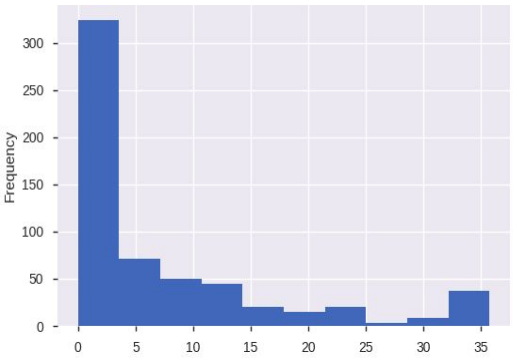




training set: validation set:

Ici, nous voyons qu'on a plus d'observations qui sont corrélées avec le temps dans le validation set. Celles ci seront potentiellement des anomalies

- max

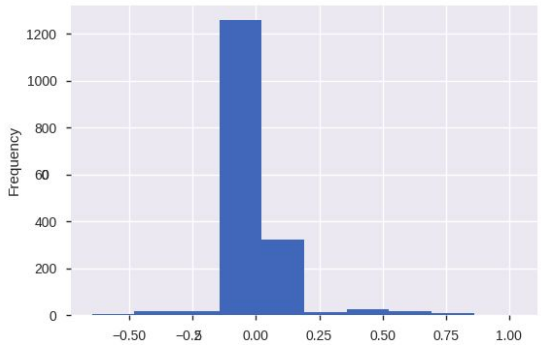


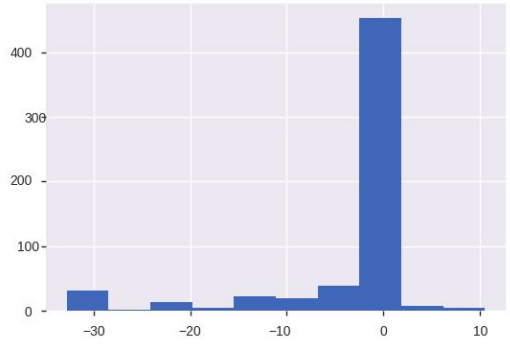




training set: validation set:

Ici nous pouvons voir qu'on a beaucoups d'observations de notre validation set qui ont des valeurs maximales assez élevées. Ce qui est en différence des signaux de notre training set. C'est donc un feature important.

- **mean**

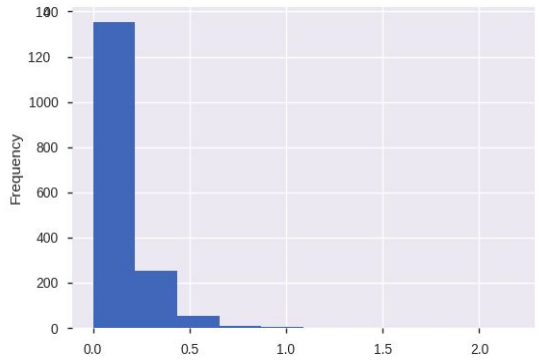


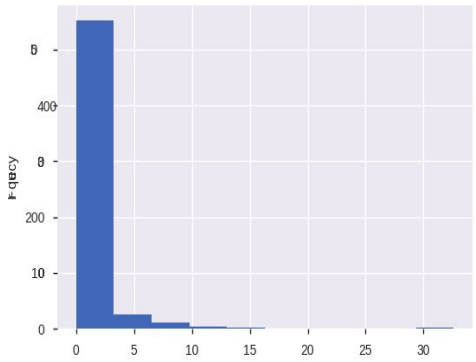




**training set:** **validation set:**

Nous avons ici aussi le même fléau que précédemment. Les moyennes des observations des deux échantillons sont centrées autour de 0 mais cependant, dans le validation set on peut trouver des signaux centrés autour de -30; de potentielles anomalies.

- **Range de moyenne par fenêtre:**

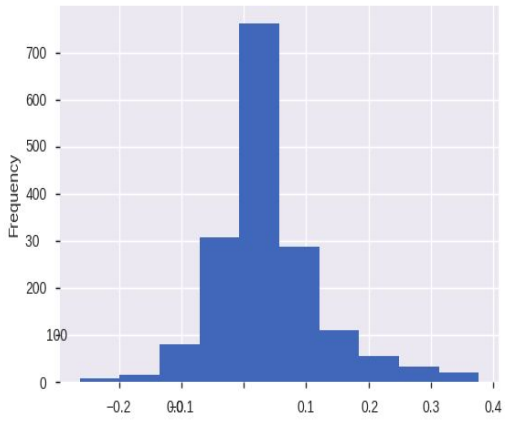


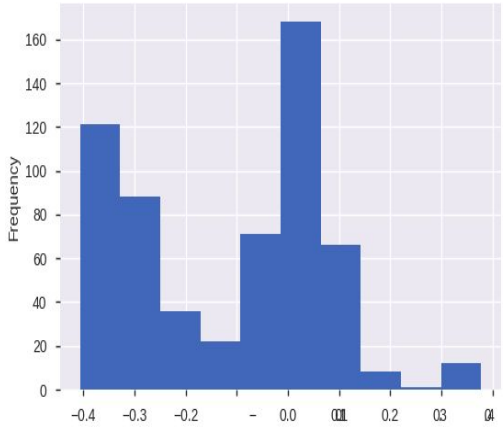

**training set:** **validation set:**

L'analyse de ces histogrammes nous montre que le maximum de la valeur obtenue pour le range de la moyenne par fenêtre de 1s vaut 1 dans le training set. Cependant, dans notre échantillon de validation, on a des écarts de moyennes qui peuvent aller jusqu'à 30.

- similarité p/r au signal moyen:

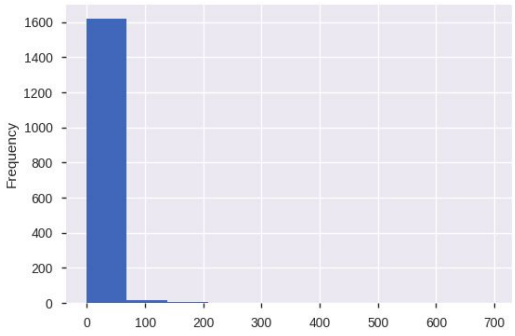



training set:

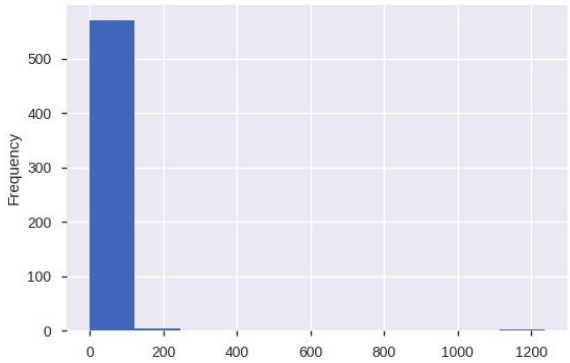



validation set:

- Score Outlier:

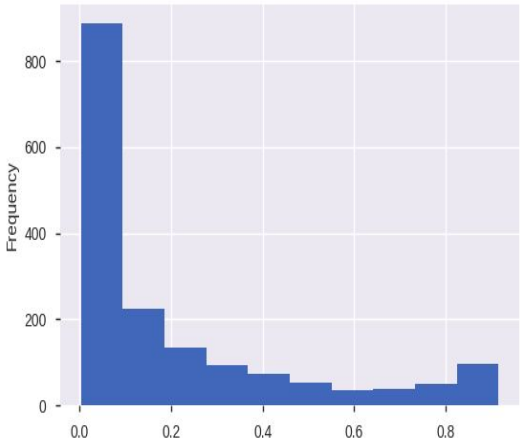



training set:

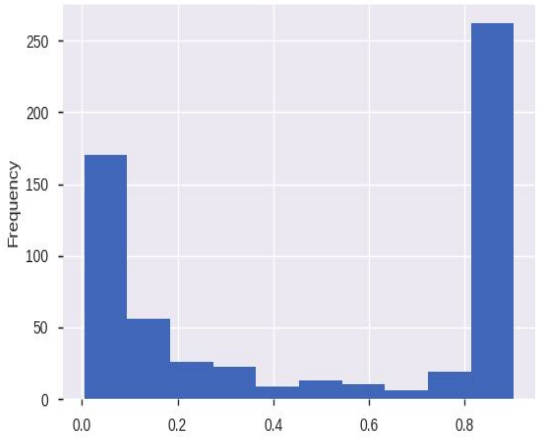



validation set:

- Similarité sur l'autocorrélation:




training set:




validation set:

Pour les 3 features précédents, nous observons des distributions dissimilaires en fonction de l'échantillon considérée. Nous avons espéré que ces dissimilarités soient assez significatives et discriminantes afin de nous permettre de détecter les outliers.