Création de dashboards Jupyter

Afin d'analyser les données des machines d'un atelier, Clément a fourni, aux ingénieurs et techniciens de l'atelier, un outil qui facilite l'affichage sous forme graphique de leurs données sans avoir besoin de code SQL bien que l'option soit aussi présente. Cet outil se nomme Metabase. Plusieurs affichages graphiques sont proposés dans cet outil et le requêtage de données sous la forme d'une interface graphique est très bien pensé pour les personnes ne sachant pas utiliser SQL. Les utilisateurs peuvent alors créer des cockpits avec les figures et données qu'ils souhaitent faire apparaitre. Il est aussi possible de filtrer les données selon une date, une valeur spécifique…

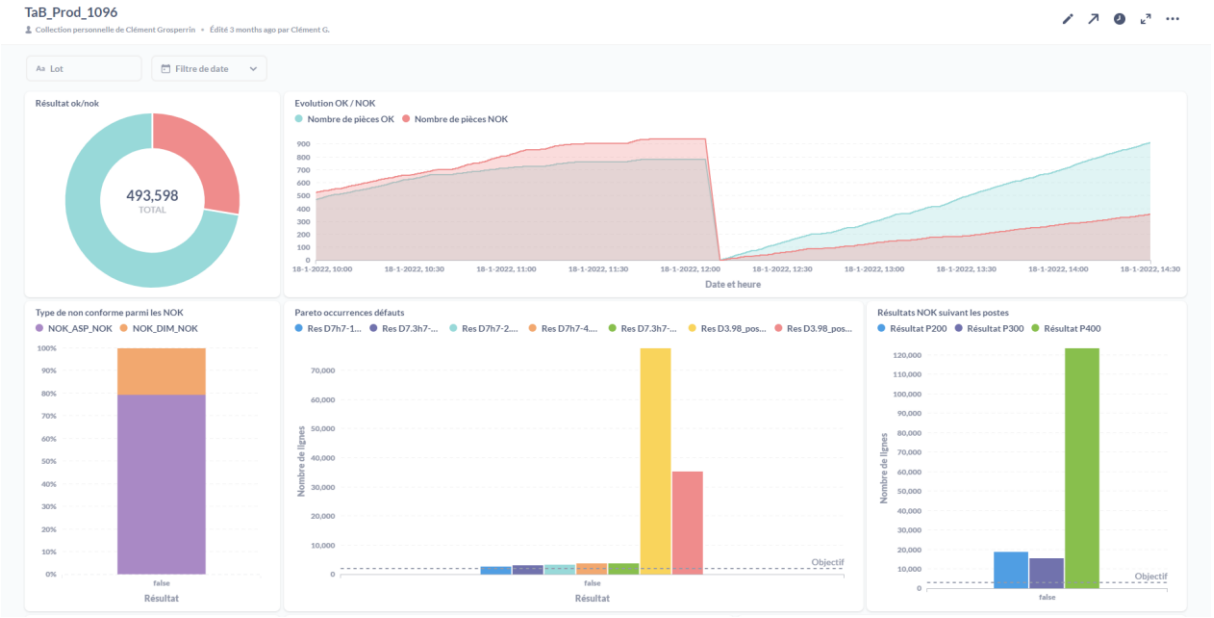


Figure 1 : Cockpit Metabase – OK/NOK et Paretos

Figure 2 : Cockpit Metabase - Cartes de contrôle et distribution

Cependant, pour un usage ou des besoins plus spécifiques, l'outil peut rapidement ne plus suffire. Par exemple, quelques usages impossibles dans Metabase :

- Affichage de bornes de tolérance sur les distributions
- Tri des résultats NOK/OK en tenant en compte les lots repassés (nécessite le développement d'un algorithme)
- Ordonner des paretos selon la taille des barres
- Zoom sur les cartes de contrôle

Le plus dérangeant sans doute dans les points énoncés plus haut est celui concernant l'affichage des résultats OK/NOK qui n'est pas correct. En effet, il compte des pièces passées plusieurs fois et n'est donc pas représentatif du nombre réel de pièces OK et NOK en sortie de machine. Certains lots de pièces sont partiellement repassés si besoin et ainsi, des pièces repassent dans la machine et sont comptabilisés deux fois avec une appellation de lot qui prend un « R-#lot » pour le différencier des lots non repassés. Pour avoir un nombre exact reflétant réellement la performance de la machine, il fallait donc vérifier si une pièce est repassée et compter uniquement le résultat de la pièce pour sa dernière repasse (car une pièce peut être repassée X fois). Par exemple dans Metabase le nombre total de pièces comme on peut le voir sur les figures précédentes avoisine les 500 000 pièces alors que le nombre réel de pièces sorties est bien inférieur.

J'ai alors proposé à Clément, pour remédier à ces problèmes, de reproduire le cockpit créé par Monsieur Grosperrin, l'un des ingénieurs de l'atelier MU32 (Composants Métalliques), grâce à l'outil Jupyter et ainsi leur fournir une solution complète. J'ai participé à plusieurs réunions avec lui pour bien cibler ses besoins et comprendre certains points propres aux deux machines pour lesquels j'allais créer un dashboard d'analyse. J'ai ensuite pu commencer la conception.

Tout d'abord, il est proposé à l'utilisateur de filtrer les données qu'il souhaite requêter en fonction des lots et de la date. De plus, il peut ajuster la moyenne mobile pour les cartes de contrôle afin de gagner en lisibilité si jamais un grand nombre de données est requêté.

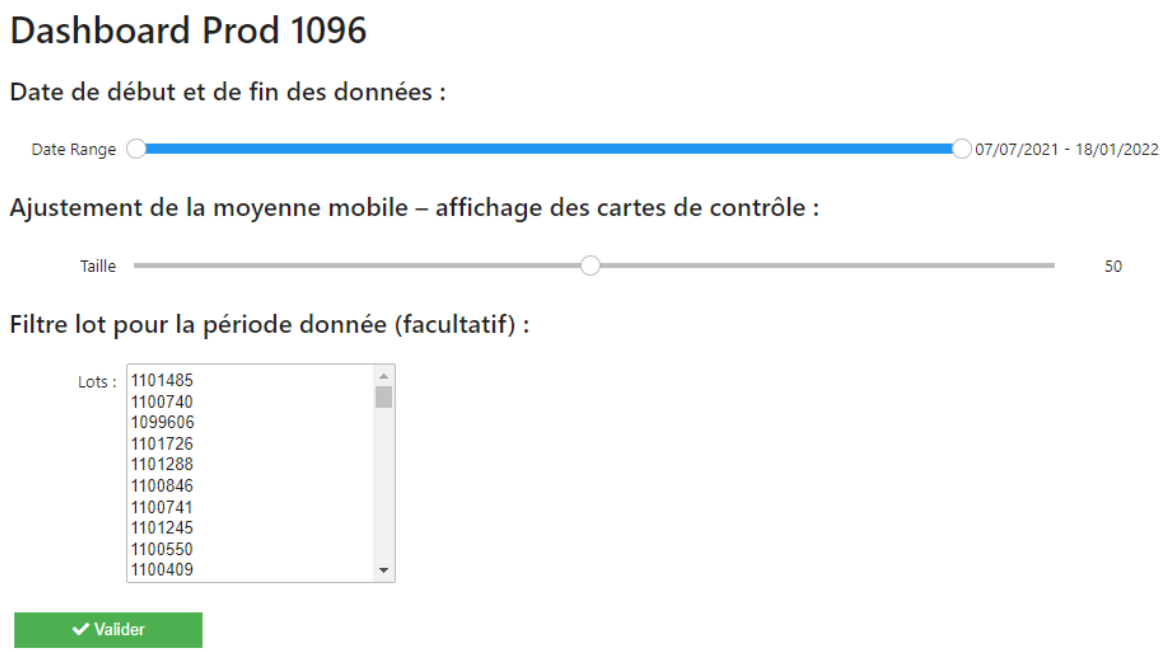
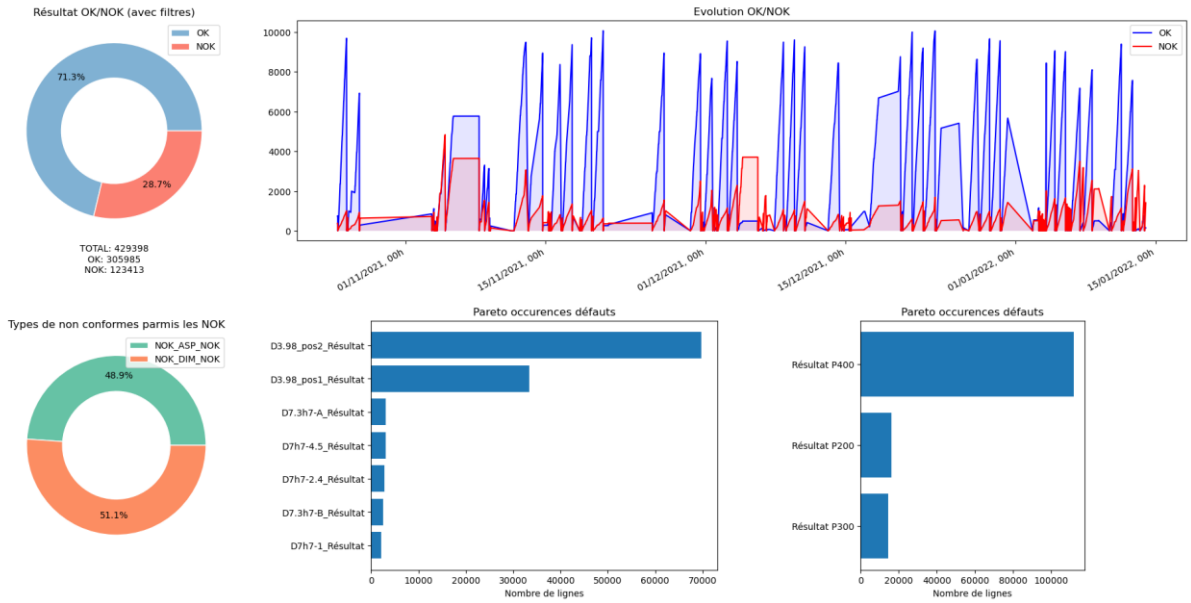


Figure 3 : Dashboard Jupyter - Accueil avec choix des filtres

Une fois qu'il valide, un message invite l'utilisateur à patienter pendant que le module pyobdc requête toutes les données nécessaires dans la base SQL. Une fois ce chargement terminé, l'affichage du dashboard complet débute.



On peut déjà remarquer ici une figure supplémentaire par rapport au cockpit Metabase, c'est un affichage des résultats OK/NOK qui tient en compte les filtres et un global. De plus, les barres des paretos sont maintenant classées de la plus grande à la plus petite.

Figure 4 : Dashboard Jupyter - OK/NOK et Paretos

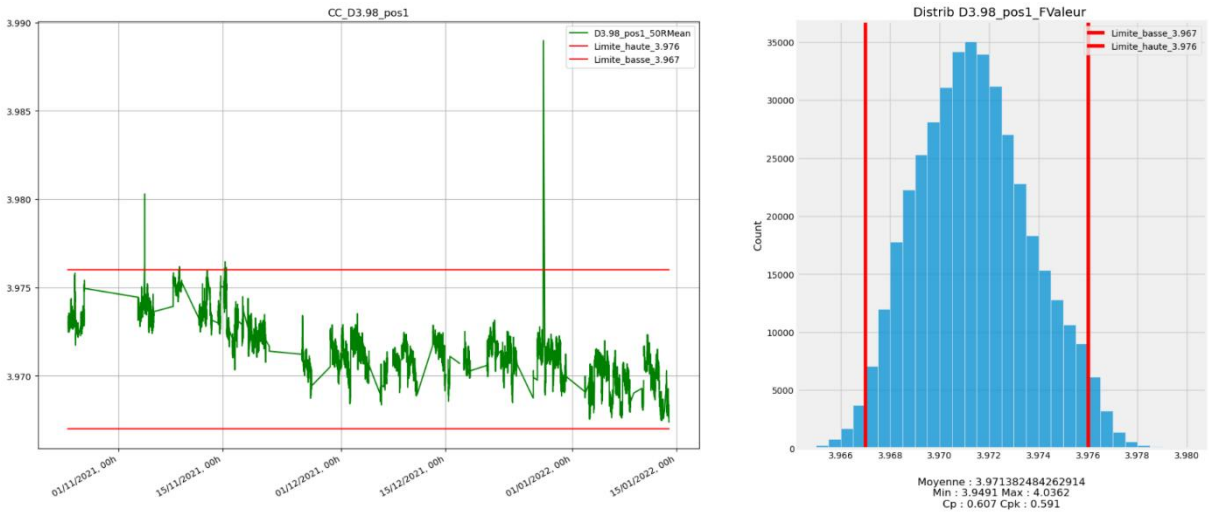


Figure 5 : Dashboard Jupyter - Carte de contrôle et distribution avec tolérances

Sur la figure ci-dessus, on peut remarquer l'ajout de bornes de tolérance sur la distribution contrairement au cockpit de Metabase où cela était impossible. On peut maintenant mieux voir si la répartition des valeurs est bonne ou non selon les tolérances. Aussi, la carte de contrôle est maintenant tracée grâce à une moyenne mobile ce qui réduit le nombre de points pour gagner en lisibilité.

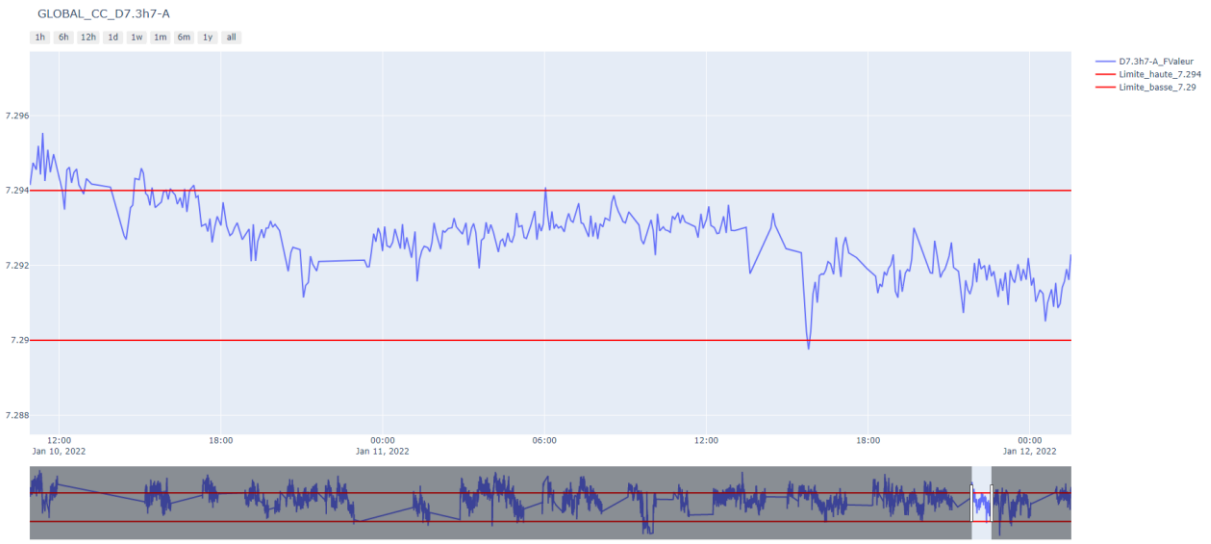


Figure 6 : Dashboard Jupyter - Carte de contrôle interactive

Les deux dernières figures (cartes de contrôle et distribution) sont créées pour chaque valeur de mesure de la machine.

Les deux machines de l'atelier pour lesquelles je devais créer un dashboard sont relativement similaires et diffèrent seulement pour les noms des mesures. Il m'a donc été très rapide de créer le deuxième dashboard.

Pour me permettre d'exporter mon dashboard Jupyter sans les morceaux de code, afin de créer une vraie interface utilisateur, j'ai utilisé la librairie « voila » qui vient charger tout le code du notebook avant de se lancer. Pour les widgets utilisés au début, j'ai utilisé la librairie ipywidgets. Pour toutes les figures graphiques et cartes de contrôle, j'ai utilisé les librairies matplotlib, plotly et seaborn. Et enfin, pour le traitement des données et les calculs, j'ai utilisé numpy et pandas.

Pour ce qui est des résultats OK/NOK qui prennent en compte les lots repassés, j'ai créé une fonction qui permettait cela et qui se basait sur les noms des lots, qui analysait la chaine de caractères qui vérifiait si un « R » était placé avant le numéro de lot, etc… Malheureusement, quand un lot était repassé, ce sont les opérateurs qui inscrivent le nom du lot qui va être repassé et il n'a été établi aucune règle de nommage sur ces lots repassés. Par exemple, voici les différentes valeurs que l'on pouvait retrouver pour un même lot :

- R-1101485
- r1_1101485
- R2_1101485

Ainsi, il était en réalité impossible de pouvoir identifier avec exactitude via un algorithme quel lot est le bon lot repassé, etc… J'ai donc commenté cette fonction dans le code car sans règle de nommage établie au préalable. Le résultat en sortie n'était pas correct non plus car les lots repassés étaient mal identifiés par l'algorithme. Une règle de nommage va donc être mise en place à l'avenir dans l'atelier pour régler ce problème.