Notes Analyse NBA --- 18 / 10 / 2024

Test MP Data - B.E MOULOUD (Notes)

Exploratory data analysis notes

Analyse de forme:

- Notre Target est de prédire la variable qualitative **TARGET_5Yrs** binaire.
- La colonne **Name** n'est pas utile -> A supprimer !!
- Il y'a 12 lignes dupliquées -> Les lignes sont a supprimer !
- On pourrait dire que ce problème est modérément déséquilibré.

```
data["TARGET_5Yrs"].value_counts(normalize=True)
```

```
TARGET_5Yrs
1.0    0.620482
0.0    0.379518
Name: proportion, dtype: float64
```

- Si on dit tout simplement que tous les joueurs vont continuer plus que 5 ans, on aura une performance de **62%**.
- La donnée a une taille de ***(1328, 20)***. Donc **1328** observations, **19** caractéristiques et un **target** comme spécifié dans le descriptif du problème.
- Tous les features sont des valeurs numériques (encodés en float)
- L'analyse des valeurs manquantes nous montre qu'il y'a 10 valeurs qui sont manquantes dans le tableau pour la variable 3P% (Tentative des 3 points).

```
data.isna().sum()
```

```
Name            0
GP              0
MIN             0
PTS             0
FGM             0
FGA             0
FG%             0
3P Made         0
3PA             0
3P%            10
FTM             0
FTA             0
FT%             0
OREB            0
DREB            0
REB             0
AST             0
STL             0
BLK             0
TOV             0
TARGET_5Yrs     0
dtype: int64
```

- La suppression des NaNs n'affecte pas les proportions dans notre **Target**

```
data.dropna()["TARGET_5Yrs"].value_counts(normalize=True)
```

```
TARGET_5Yrs
1.0    0.621396
0.0    0.378604
Name: proportion, dtype: float64
```

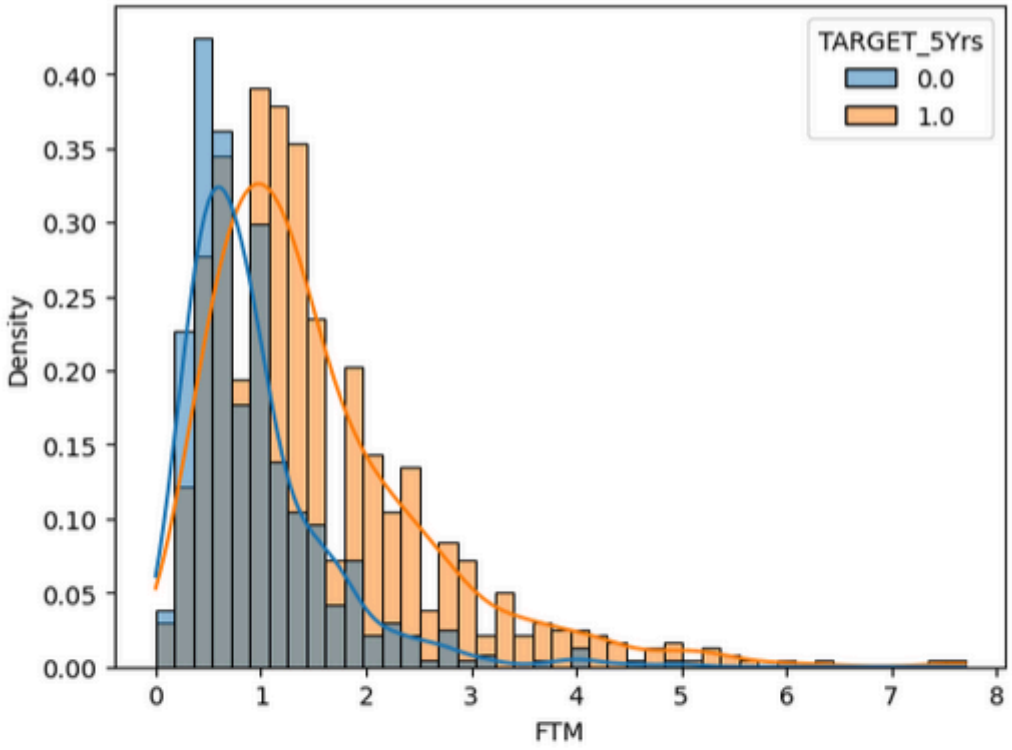
- Les NaNs sont peu nombreux dans notre dataset (**0.75%**) et donc meme leur suppression n'aura pas generallement d'influence sur la performance du modele, on va décider de ce qu'on fait avec les NaNs a la fin de la phase exploratoire.

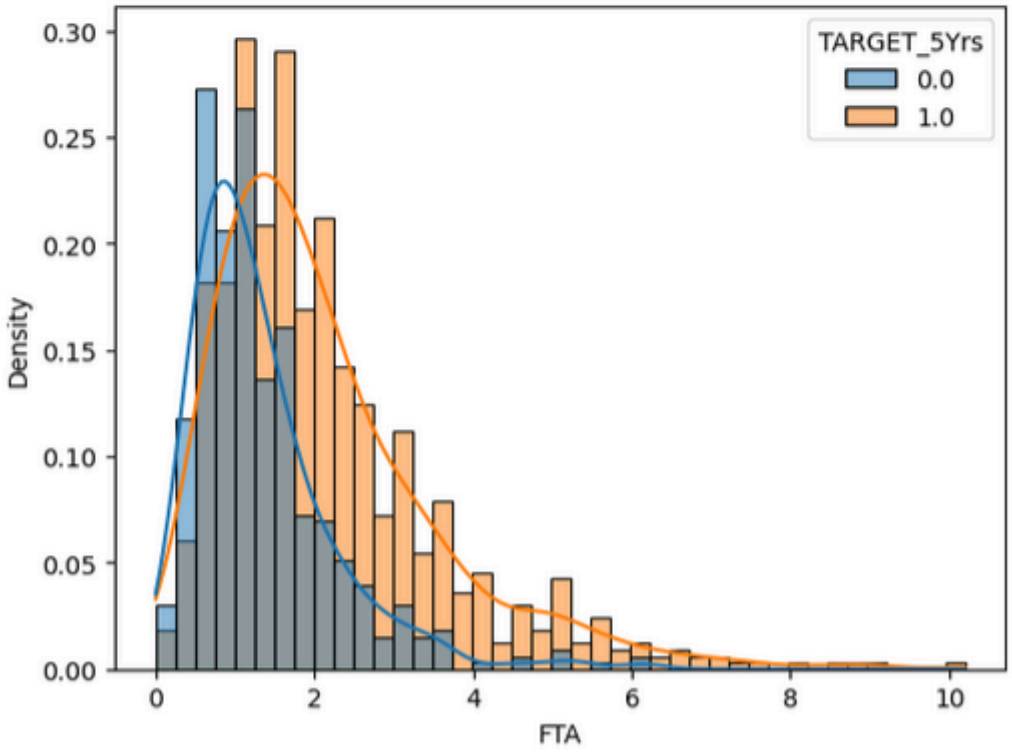
Premier Analyse de fond.

- [x] Visualisation des variables conitnues.
- [x] Relations Target / Variables

- Rien de spécial dans la distribution des variables continues (Distibutions normales ou expo).
- Il y'a des joueurs qui ont joués beaucoup moins que d'autres. Le minimum de temps de jeu est 3 min, la moyenne par joueur etant a environ 18 minutes. Ce joueur a joué 3 min sur 14 matchs. Donc, ca serai interessant par la suite d'ajouter une variable **Min / GP** parcequ'elle peut etre décisive aussi.

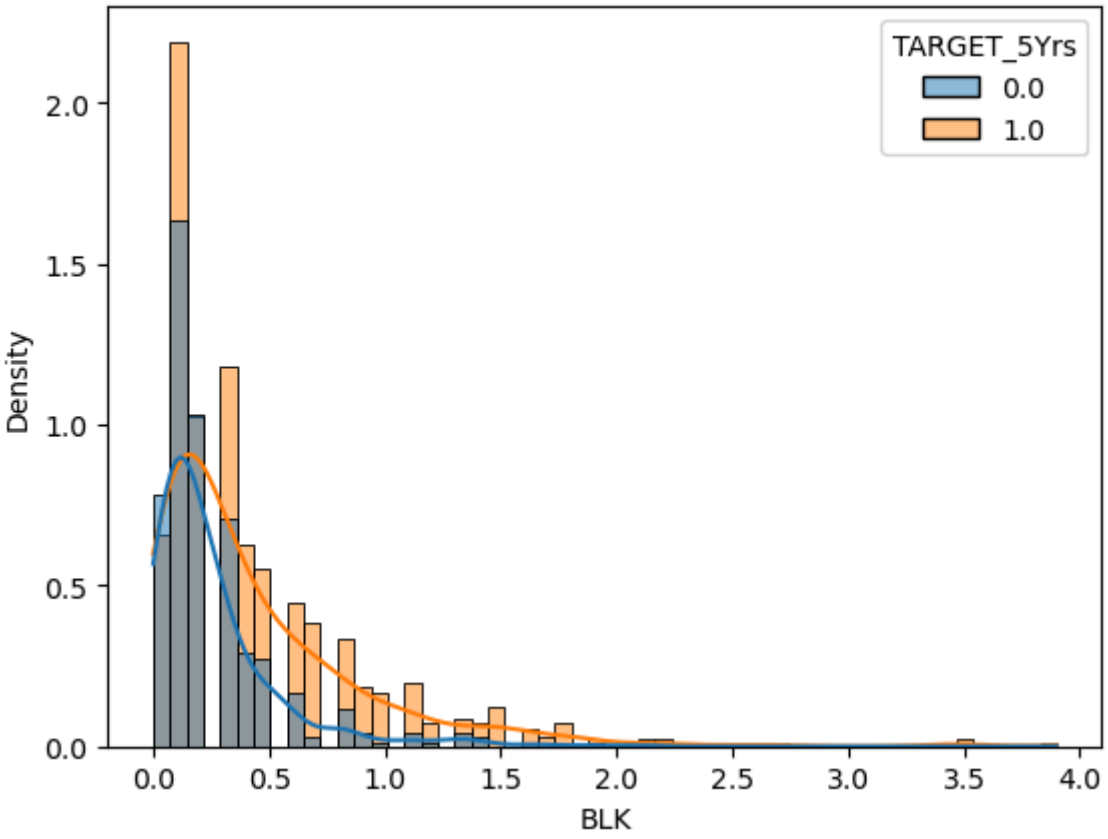
- **Relations Target/Variables**:
  - On voit que la majorité des variables semblent différer d'un joueur qui durera plus de 5 ans en NBA a un autre qui ne durera pas. **Exemple:**

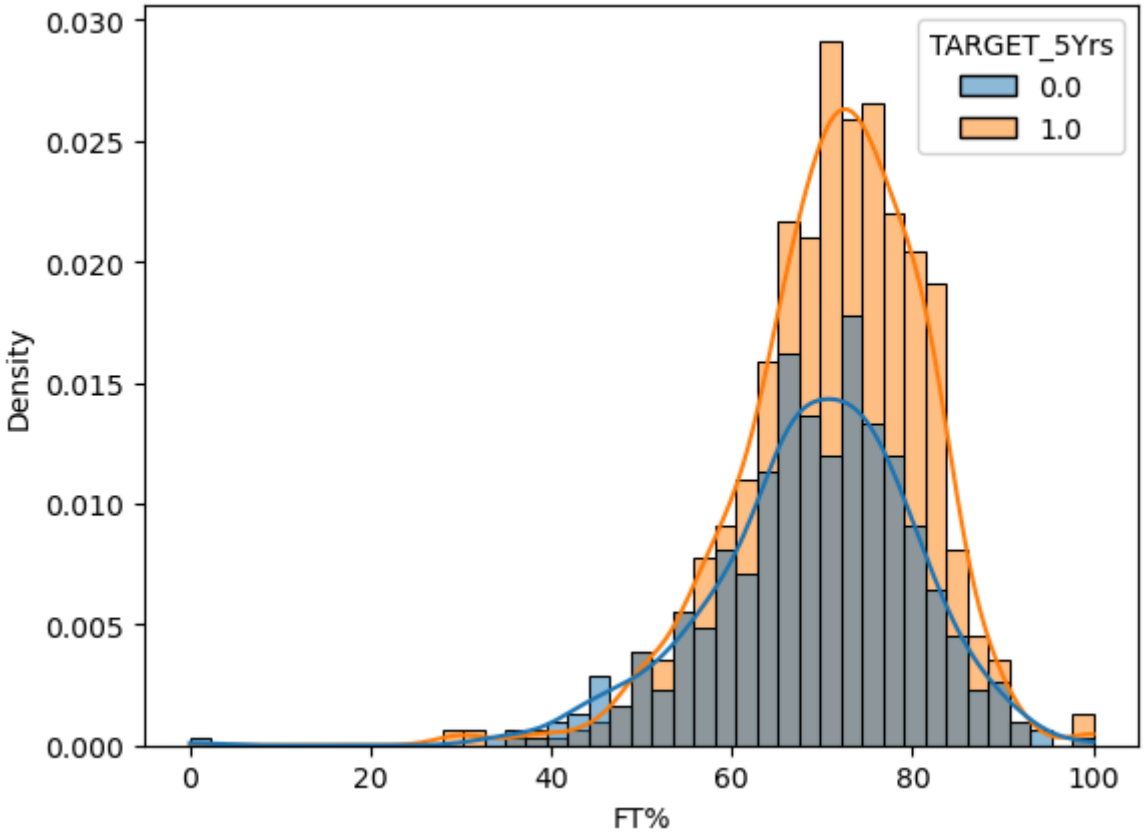




Il va donc falloir tester (**A travers un test statistique de student par exemple**) si ces variables sont significativement différents entre ces deux catégories de joueurs via un test statistique.

- D'un autre coté, certaines variables ne le sont pas



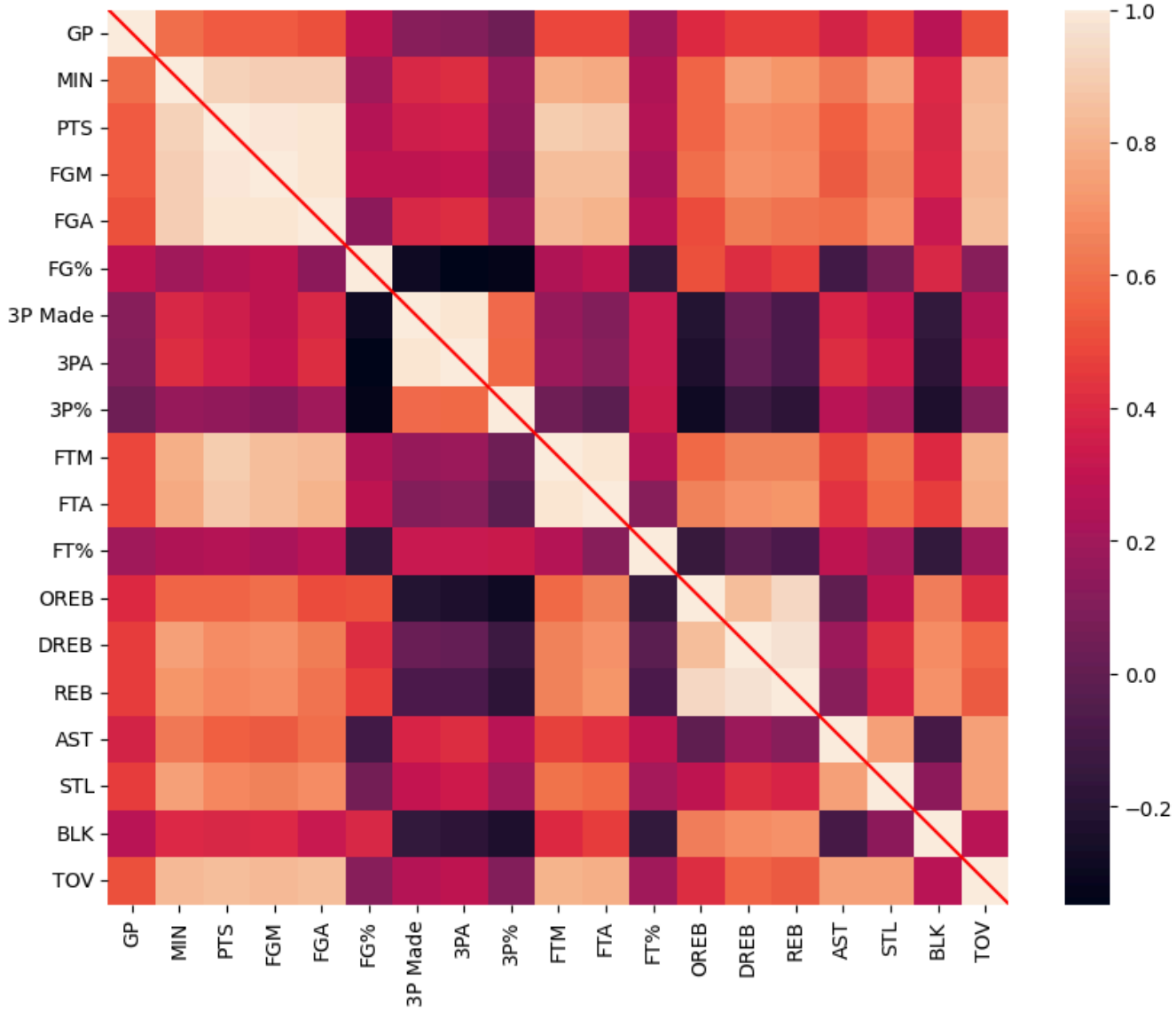


Analyse de fond approfondi

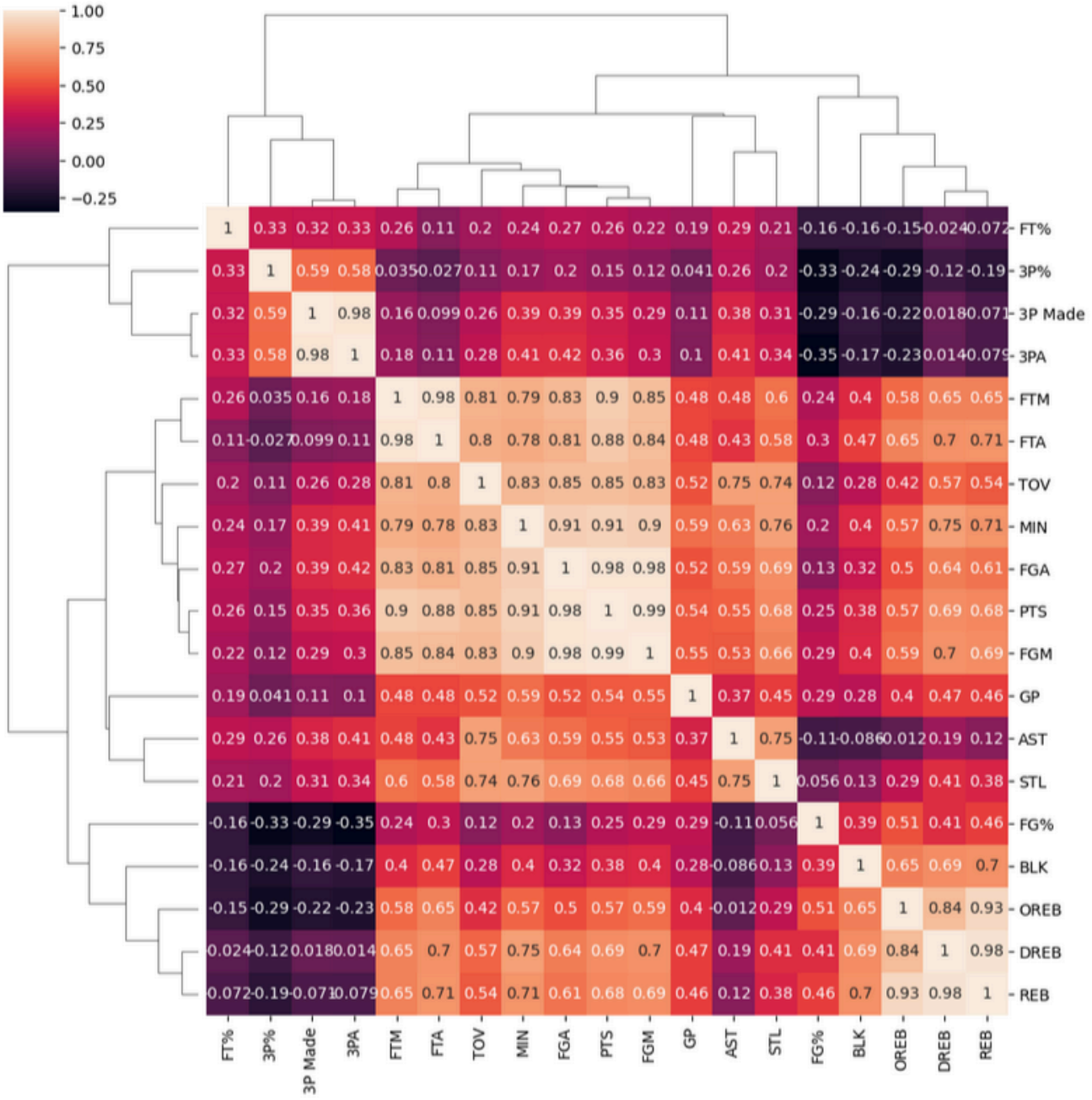
On va essayer de voir maintenant les relations entre les différentes variables et s'il y'a du sens dedans ou pas.

- Le pairplot entre les différentes variables montre qu'il y'a bel et bien des relations de corrélation entre différentes variables de notre donnée (Ce qui est naturel pour cette donnée). Connaitre ces corrélations peut nous etre utile par la suite dans la preparation de la donnée pour la modélisation.
- On va visualizer ces relations dans un tableau de corrélartions de Pearson ou un clustermap pour une visualizarion plus facile des variables corrélées.

  CorrMap



ClusterMap



- Les corrélations observées ici s'imposent automatiquement. Par exemple, dans le premier cluster de corrélation **3P Made** et **3PA** sont naturellement corrélés puisque ce sont les tentatives des 3points qui menent vers les 3points.

Analyse des clusters dans la ClusterMap

1. Cluster basé sur les minutes jouées (MIN), les points (PTS), les tirs réussis (FGM) et les tentatives de tir (FGA)

- **Forte corrélation (r > 0.9) entre MIN, PTS, FGM et FGA** : Plus un joueur passe de temps sur le terrain (MIN), plus il a tendance à marquer de points (PTS), à réussir des tirs (FGM) et à tenter des tirs (FGA). Cela montre une association naturelle entre le temps de jeu et la performance offensive de chaque joueur.
- **Corrélations avec les lancers francs (FTM et FTA)** : Il existe aussi une corrélation élevée entre les points et les lancers francs tentés et réussis (FTA et FTM), ce qui montre que les joueurs qui marquent plus de points ont tendance à obtenir et réussir plus de lancers francs.

2. Cluster de la réussite au tir (FG%), aux rebonds (OREB, DREB, REB) et à la défense

- **Corrélation entre FG%, OREB, DREB et REB** : La précision au tir (FG%) est modérément corrélée avec les rebonds offensifs (OREB) et défensifs (DREB), ainsi qu'avec les rebonds totaux (REB). Cela suggère que les joueurs efficaces dans le tir tendent à capter plus de rebonds, renforçant leur performance globale.
- **Lien avec les contres (BLK) et les vols de ballons (STL)** : Les joueurs qui captent plus de rebonds (surtout défensifs) sont également liés à la capacité à contrer (BLK) et intercepter (STL), montrant une synergie naturelle entre la présence défensive et la capacité à sécuriser les rebonds.

3. Cluster des tirs à trois points (3P Made, 3PA, 3P%)

- **Forte corrélation entre 3P Made et 3PA (r > 0.98)** : Cela montre que les joueurs qui tirent plus à trois points réussissent également plus, ce qui est intuitif.
- **Corrélation modérée avec la précision à trois points (3P%)** : L'efficacité au tir à trois points (3P%) a une corrélation modérée avec le nombre de tirs à trois points réalisés et tentés, ce qui montre que tenter beaucoup de trois points n'implique pas forcément une très haute précision.

4. Cluster des passes décisives (AST), pertes de balle (TOV), et interceptions (STL)

- **Forte corrélation entre AST, TOV et STL** : Les joueurs qui font plus de passes décisives (AST) ont tendance à perdre également plus la balle (TOV), mais sont aussi plus actifs en défense avec des interceptions (STL). Cela peut refléter l'activité des meneurs de jeu qui contrôlent beaucoup la balle.
- **Relation avec le temps de jeu (MIN)** : Une correlation relativement forte (r > 0.63) entre les minutes jouées et les passes décisives (AST), ce qui indique que plus un joueur passe de temps sur le terrain, plus il contribue à la création de jeu.

5. Autres réusltats spécifiques

- **Faible corrélation entre FG% et 3P%** : La precision globale au tir (FG%) est faiblement liée à la precision à trois points (3P%), suggerant que l'efficacité dans ces deux catégories peut être indépendante.
- **Impact des tirs à trois points sur les autres statistiques** : Les tirs à trois points réalisés (3P Made) et tentés (3PA) ont une corrélation négative avec certaines statistiques comme les rebonds (OREB, DREB) et les contres (BLK), montrant que les joueurs axés sur les trois points ont tendance à être moins impliqués dans ces aspects du jeu.

Remarques:

- Il pourrait etre interessant de créer des groupes des variables "defensives" vs "offensives" pour la suite parceque les deux groupes de varaibles ont une influence directe sur notre target.
- On peut envisager de créer d'autres variables aussi a partir des variables présnetes, tels que. A premiere refelexion, les variables qu'on peut envisager sont les suivantes:
    - Points par minute,
    - Taux de réussite des tirs,
    - Taux de réussite des tirs à 3 points,

    - Taux de réussite des lancers francs,
    - Rebounds offensifs par total de rebonds,
    - Rebounds défensifs par total de rebonds,
    - Passes décisives par perte de balle,
    - Interceptions par minute,
    - Contres par minute,
    - Pertes de balle par minute.
- C'est dans la partie pre-processing qu'on va faire des décisions en fonction de la performance d'un premier modele simple déstiné a guider intelligemment le preprocessing.

Conclusion

Cet EDA a permis de dégager plusieurs insights importants concernant le dataset. La cible principale est la variable binaire **TARGET_5Yrs**. Au total, le dataset comprend 1328 observations et 19 caractéristiques, toutes de type numérique. L'analyse des valeurs manquantes a révélé un faible pourcentage (0,75%) de données manquantes, ce qui nous permet d'envisager la suppression des valeurs nulles sans compromettre l'intégrité du modèle.

Les visualisations ont mis en évidence des relations significatives entre certaines variables et notre cible, suggérant que des tests statistiques supplémentaires seront nécessaires pour confirmer ces différences. Nous avons également identifié des clusters de corrélations, ce qui pourrait nous guider dans la création de nouvelles variables pour mieux modéliser la performance des joueurs.

Checklist des Points à Tester ou à Faire

- [x] Tester les hypothèses formulées durant l'analyse préliminaire à l'aide de tests statistiques par un test de student:
    - [x] **Hypothèses à tester:** Les joueurs ayant la plus de chace d'avoir une longue carriere en NBA ont, moyennement, leurs différentes variables caractéristiques significativement différentes !
    - [x] **Hypothese initiale (nulle)**: Moyennement, les variables sont **égaux** chez les deux catégories de joueurs (alpha=0.01).
    - [x] **Résultat apres test de student**: La majorité des variables sont vraiement significativement différents entre les différentes catégories des joueurs.

```python
for col in test_data.columns[:-1]:
    test_result = t_test(col)
    print(f"{col :-<50}  {test_result}")
```

```
GP-------------------------------------------------  H0 rejetée
MIN------------------------------------------------  H0 rejetée
PTS------------------------------------------------  H0 rejetée
FGM------------------------------------------------  H0 rejetée
FGA------------------------------------------------  H0 rejetée
FG%------------------------------------------------  H0 rejetée
3P Made--------------------------------------------  H0
3PA------------------------------------------------  H0
3P%------------------------------------------------  H0
FTM------------------------------------------------  H0 rejetée
FTA------------------------------------------------  H0 rejetée
FT%------------------------------------------------  H0 rejetée
OREB-----------------------------------------------  H0 rejetée
DREB-----------------------------------------------  H0 rejetée
REB------------------------------------------------  H0 rejetée
AST------------------------------------------------  H0 rejetée
STL------------------------------------------------  H0 rejetée
BLK------------------------------------------------  H0 rejetée
TOV------------------------------------------------  H0 rejetée
```

Preprocessing

Démarche:

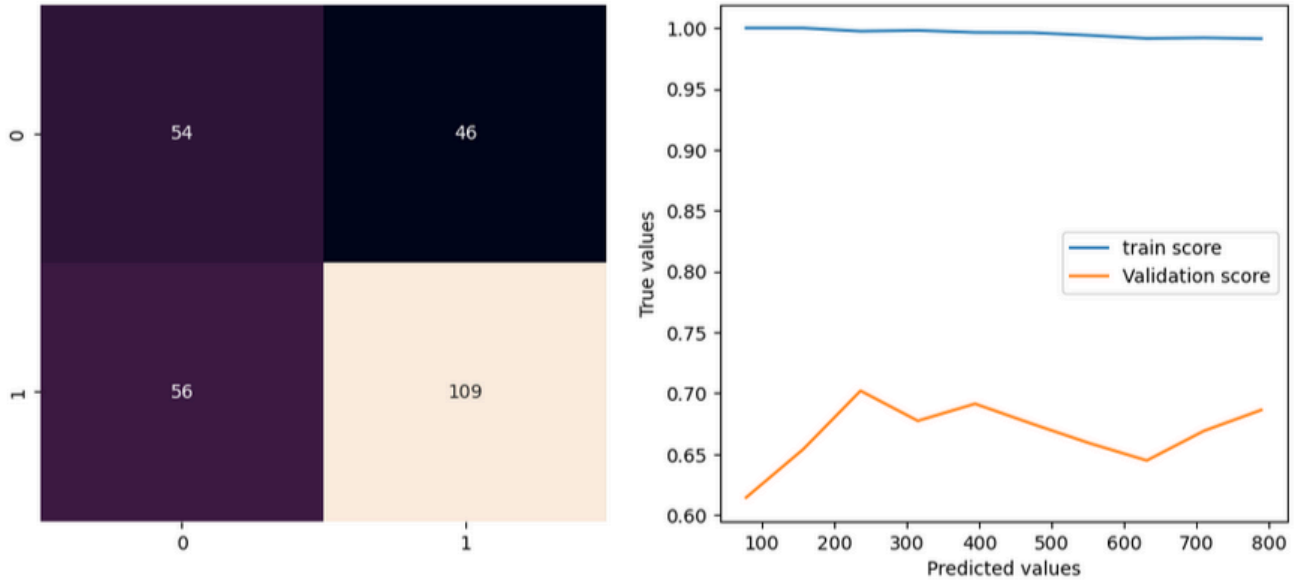
- On va reprendre notre donnée initiale. Nous allons ré-effectuer toutes les opérations que nous avions jugés nécessaires durant l'EDA et le minimum possible de preprocessing. A savoir:
  - [x] Suppression de la colonne **Name**.
  - [x] Suppression des valeurs dupliquées.
  - [x] Imputation des valeurs manquantes.
- On va ensuite créer un premier modele hyper simple avec le minimum de PreProcessing (On commence par un simple Arbre de decision ou un RandomForest ici qu'on peut changer si besoin) pour bien mener notre pre-processing en fonction de la performance de ce premier modele.
- Nous allons créer une approche robuste et reproductible d'evaluation. A ce stade, nous allons nous contenter de la **recall** et **f1-score** pré-implementé dans sklearn. On visualizera le **Recall** précisé dans le `test.py` de MP Data en meme temps.

Pre-processing notes

- On va prendre comme metrique le score **f1** qui est prend un bon compromis entre **précision** et le **recall**.

- On a entrainé un Arbre de decision avec le minmimum possible de preprocessing (Uniquement avec les etapes mentionnés dans les demarches):

```
              precision    recall  f1-score   support

         0.0       0.49      0.54      0.51       100
         1.0       0.70      0.66      0.68       165

    accuracy                           0.62       265
   macro avg       0.60      0.60      0.60       265
weighted avg       0.62      0.62      0.62       265

-------------------- Custom score function from MP DATA
Custom recall on trainning data
[[191. 208.]
 [222. 432.]]
0.6597500750478914

Custom recall on Validation data
[[ 55.  45.]
 [ 54. 111.]]
0.67354979787949
------------------------------------------------------------
```

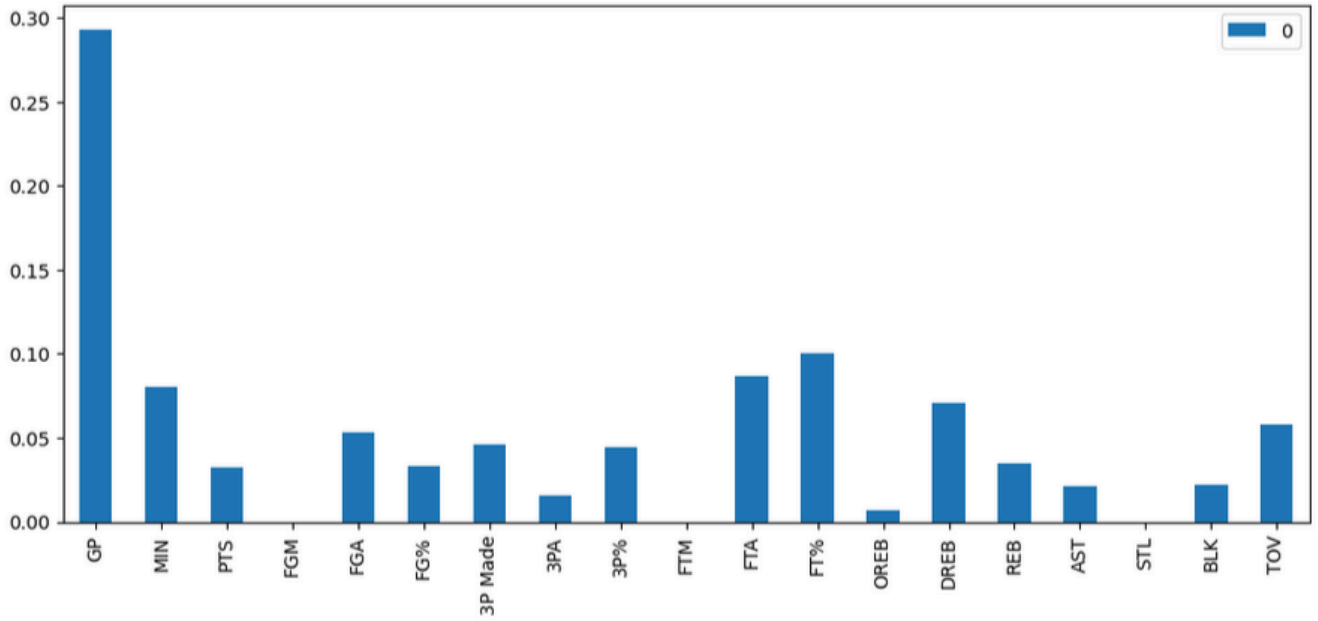


On voit qu'on est un peu en situation d'Overfitting. Maintenant, on va garder notre modele simple, et on va essayer de l'ameliorer avec du preprocessing.

- **Premiere tentative**: On va observer les variables qui sont les plus significatives d'apres notre modele d'arbre de decision.

```
feature_importance = pd.DataFrame(prep_step_model.feature_importances_, index=X_train.columns)
```

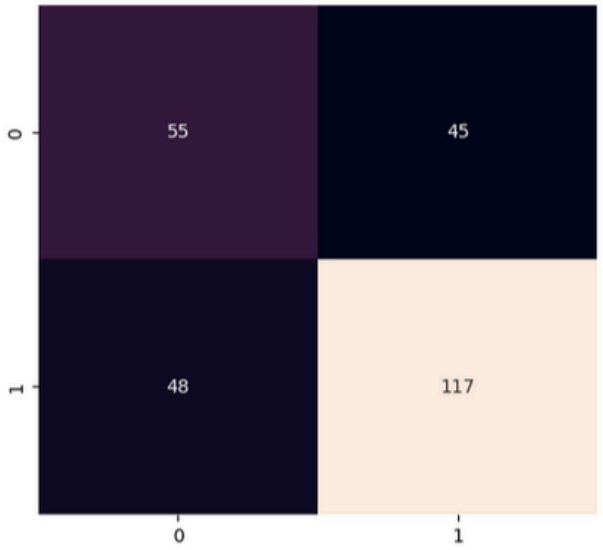
```
fig, ax = plt.subplots(figsize=[12,5])
feature_importance.plot.bar(fig=fig, ax=ax)
```

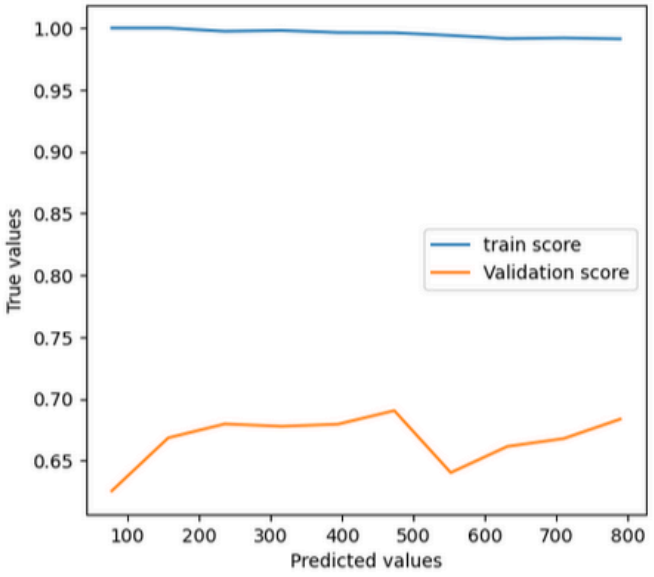


- Il y'a beaucoup de conclusions qu'on peut tirer de ce graphique simple.
    - La premiere immediate c'est que les variables **FGM**, **FTM** et **STL** ne contribuent pratiquement pas a la performance globale du modele. On va donc tenter de modifier la fonction

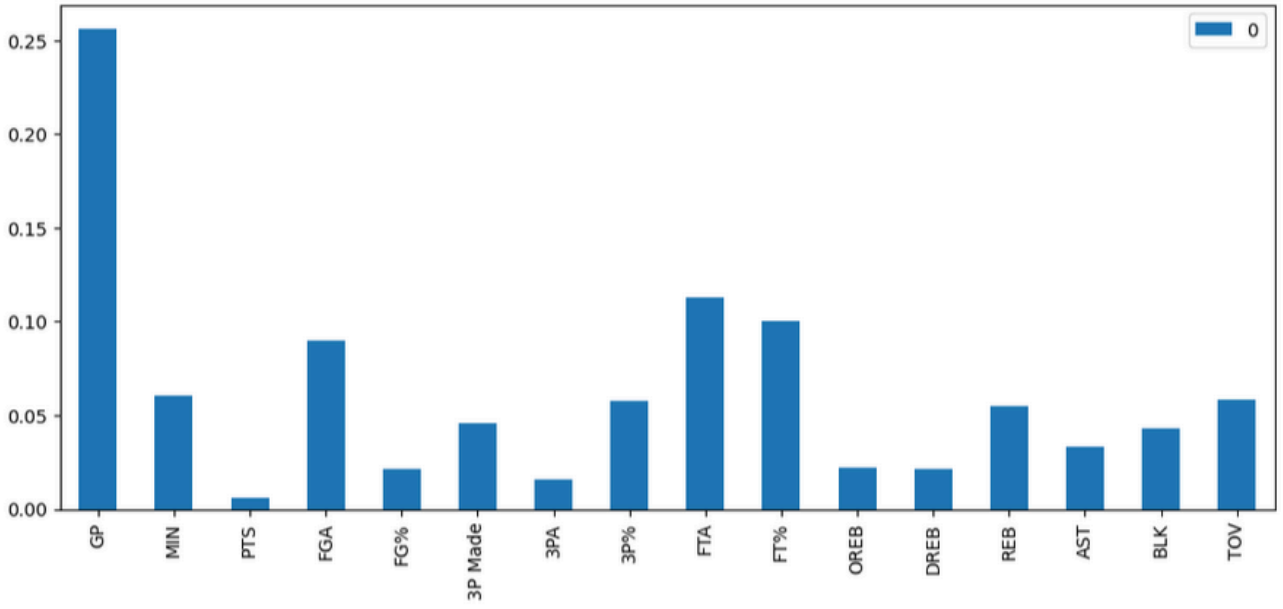
`feature_engineering` en abondant ces 3 variables.

```
              precision    recall  f1-score   support

         0.0       0.53      0.55      0.54       100
         1.0       0.72      0.71      0.72       165

    accuracy                           0.65       265
   macro avg       0.63      0.63      0.63       265
weighted avg       0.65      0.65      0.65       265

-------------------- Custom score function from MP DATA
Custom recall on trainning data
[[205. 194.]
 [217. 437.]]
0.6678704309760558

Custom recall on Validation data
[[ 49.  51.]
 [ 37. 128.]]
0.7757615880753915
------------------------------------------------------------
```





- Clairement, les resultats sans ces 3 colonnes sont bien meilleures avec toutes les metriques. On a un overfitting qui persiste par contre. Notre nouveau graphique d'importance des variables est le suivant:



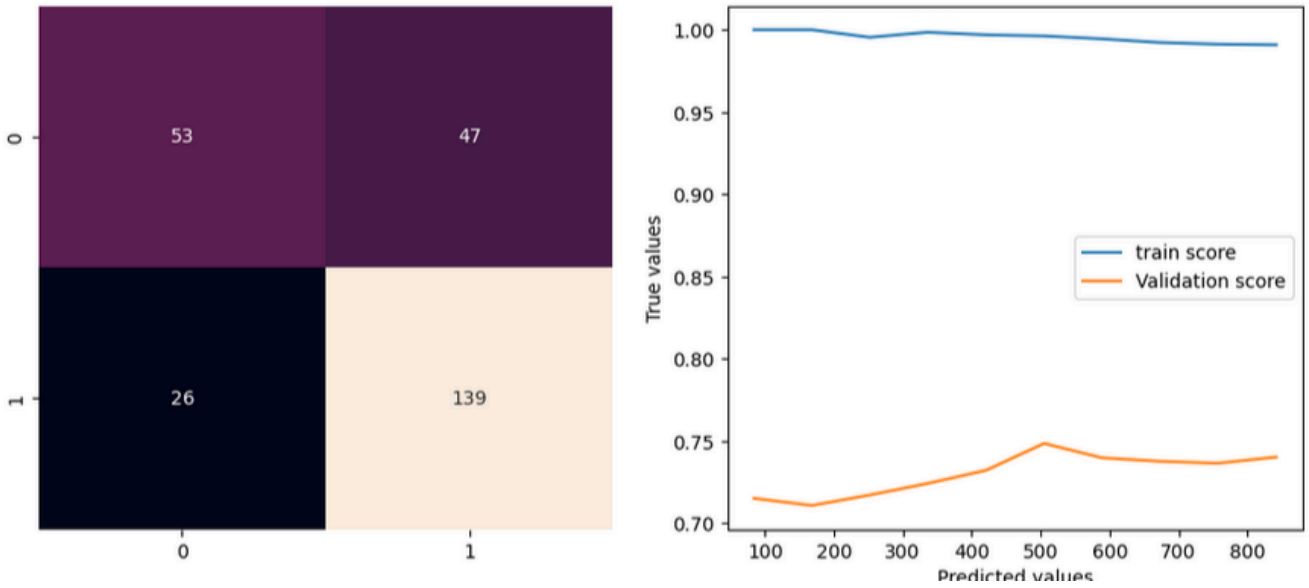
- Contrairement a FTM, FGM, FTA et FGA sont significatifs. Apparemment, on s'intéresse plus à la prise d'initiative (tentatives) qu'à la réussite en soi, car les lancers francs réussis (FTM) et les

tirs réussis (FGM) n'influencent pas autant la performance du modèle que les tentatives de lancers francs (FTA) et de tirs (FGA). Cela montre que les joueurs qui prennent beaucoup d'initiatives et créent des opportunités de tir, même s'ils ne les convertissent pas toujours, sont plus valorisés. Leur implication offensive et leur potentiel à améliorer leur efficacité en font peut etre des profils prometteurs à développer.

    - Les points marquées par match est peu influant apparemment. Le potentiel offensif et défensif ainsi que la présence dans les matchs sont les plus significatifs.

- Si on regarde un RandomForest non-optimisé, on obtient des résultats plus satisfaisants:

```
              precision    recall  f1-score   support

         0.0       0.67      0.53      0.59       100
         1.0       0.75      0.84      0.79       165

    accuracy                           0.72       265
   macro avg       0.71      0.69      0.69       265
weighted avg       0.72      0.72      0.72       265

-------------------- Custom score function from MP DATA
Custom recall on trainning data
[[201. 198.]
 [148. 506.]]
0.7746486868940076

Custom recall on Validation data
[[ 59.  41.]
 [ 23. 142.]]
0.8600523607474949
------------------------------------------------------------
```

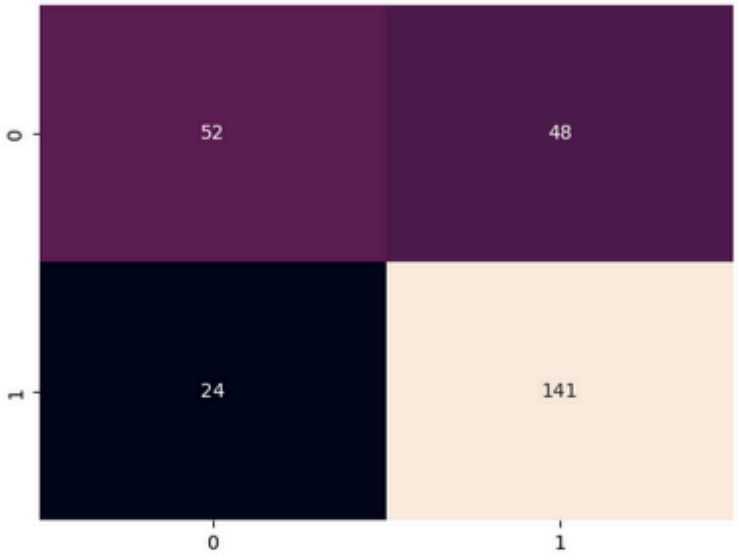


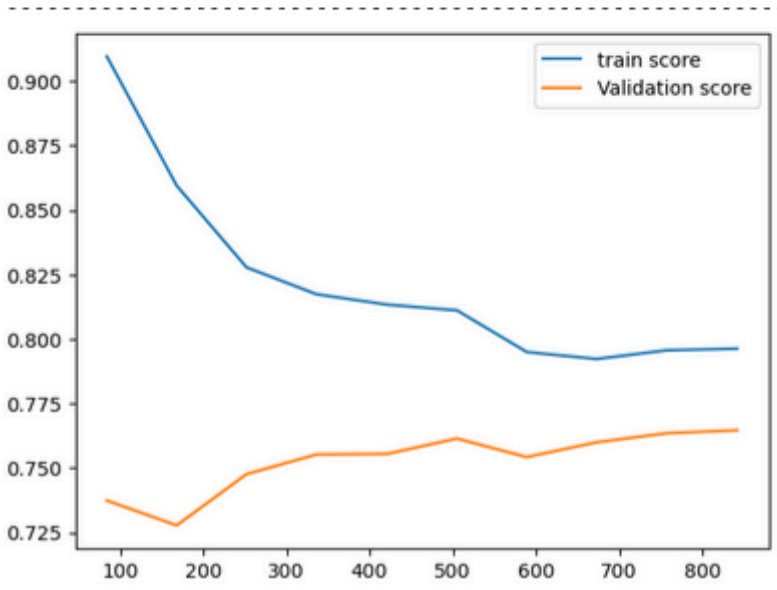
.

- On est legerement meilleur en termes de Recall, mais nous avons toujours un overfitting modéré. On va essayer ici de voir l'effet du pre-pruning en changeant seulement la profondeur des arbres (`max_depth`):

```
Pipeline
  StandardScaler
  RandomForestClassifier
RandomForestClassifier(max_depth=3, random_state=0)
```

```
              precision    recall  f1-score   support

         0.0       0.65      0.49      0.56       100
         1.0       0.73      0.84      0.78       165

    accuracy                           0.71       265
   macro avg       0.69      0.67      0.67       265
weighted avg       0.70      0.71      0.70       265

-------------------- Custom score function from MP DATA
Custom recall on Test data
MP data recall: 0.8537630525713942
```





- Dans ce cas, on voit que les scores sur le test et sur la validation se rapporche et le gap est devenu plus petit. Dans ce cas on ne peut plus parler d'Overfitting.
- Le pre-pruning permettra eventuellement de lutter contre l'Overfitting dans les **Arbre de decision** et les **RandomForest**

- On va s'arreter a ce stade dans notre preprocessing et on va essayer d'optimiser plusieurs modeles différents, a savoir, un **Arbre de decision**, **RandomForest**, **AdaBoost**, **gradBoost**, **SVM** et **KNN**.

Optimisation

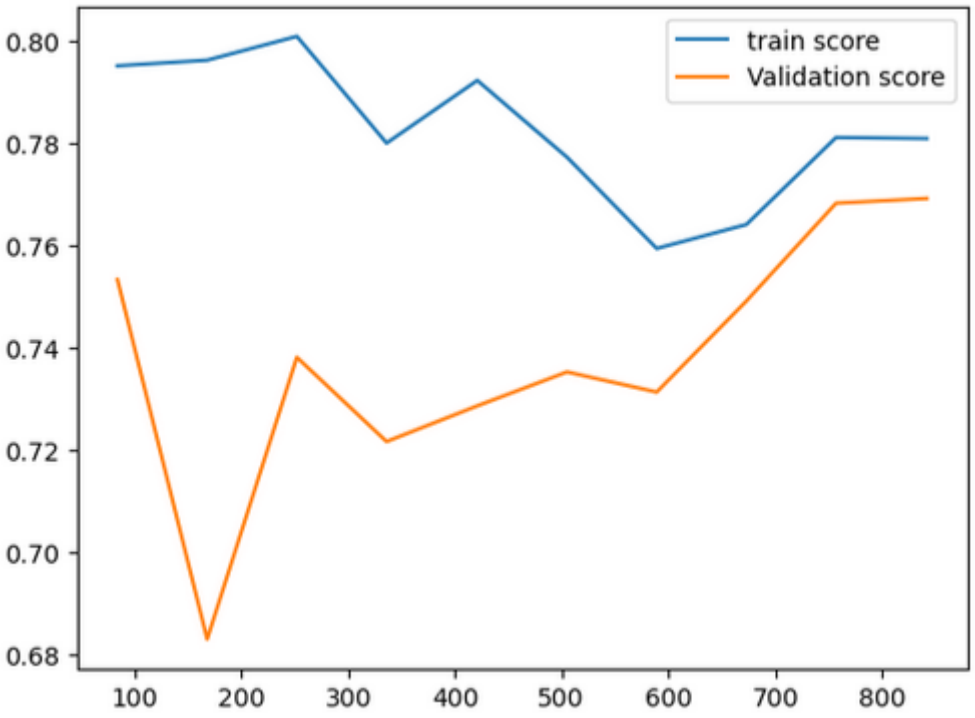
- Nous avons optmisé tous les modeles **Arbre de decision**, **RandomForest**, **AdaBoost**, **gradBoost**, **SVM** et **KNN**.
- Nous avons illustons dans ce tableau un recap de toutes les performances avec et sans modification de la frontiere de decision.

	Model name	Recall - Train	F1 - Train	Accuracy - Train	Recall - Test	F1 - Test	Accuracy - Test	Decision boundary	Threshold
0	best_decisionTree_estimator	0.831861	0.769150	0.690440	0.975758	0.813131	0.720755	False	NaN
1	best_randomForest_estimator	0.865449	0.771120	0.680943	0.933333	0.819149	0.743396	False	NaN
2	best_gradBoost_estimator	0.873083	0.783419	0.699946	0.951515	0.853261	0.796226	False	NaN
3	best_adaBoost_estimator	0.824146	0.778026	0.708459	0.890909	0.862170	0.822642	False	NaN
4	best_svm_estimator	0.827258	0.774038	0.699959	0.836364	0.826347	0.781132	False	NaN
5	best_knn_estimator	0.793611	0.753452	0.678104	0.830303	0.810651	0.758491	False	NaN
6	best_decisionTree_estimator	0.831861	0.769150	0.690440	NaN	NaN	NaN	True	NaN
7	best_randomForest_estimator	0.865449	0.771120	0.680943	NaN	NaN	NaN	True	NaN
8	best_gradBoost_estimator	0.873083	0.783419	0.699946	0.842424	0.844985	0.807547	True	0.253623
9	best_adaBoost_estimator	0.824146	0.778026	0.708459	0.854545	0.854545	0.818868	True	0.012582
10	best_svm_estimator	0.827258	0.774038	0.699959	0.824242	0.824242	0.781132	True	0.021276
11	best_knn_estimator	0.793611	0.753452	0.678104	NaN	NaN	NaN	True	NaN

Le Recall représente la capacité du modèle à identifier correctement les talents prometteurs (minimiser les faux négatifs), mais il est également crucial de garder un équilibre avec le F1-score et l'accuracy pour éviter de surestimer les faux talents (minimiser les faux positifs).

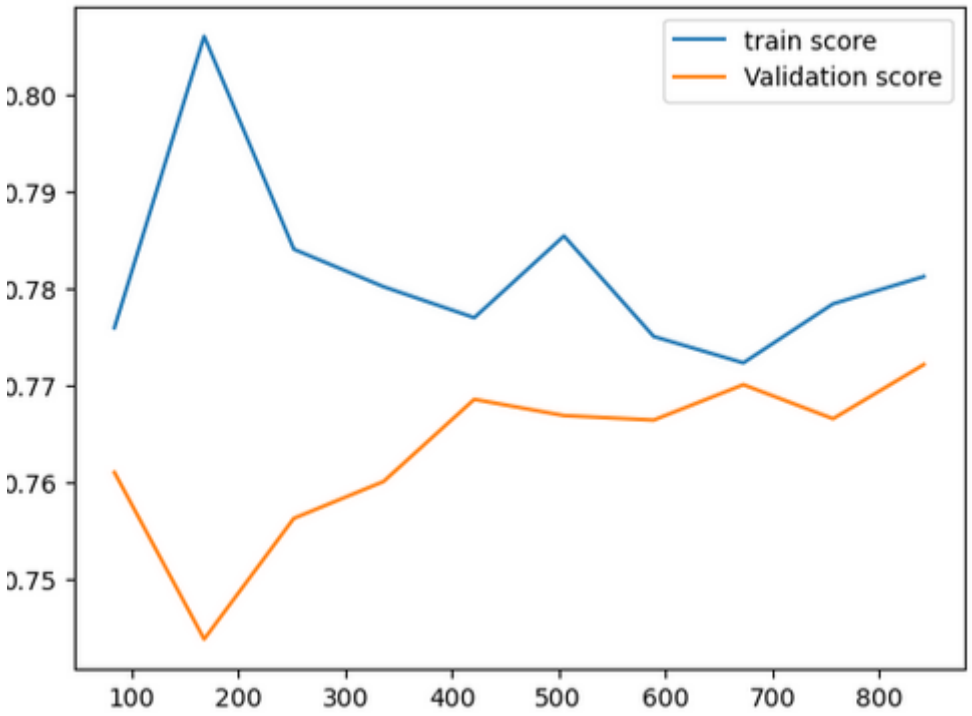
1. **Decision Tree**
    - Train : Recall de 83,2%, F1-score de 76,9%, Accuracy de 69,0%.
    - Test : Recall très élevé de 97,6%, mais un F1-score de 81,3%, et une accuracy de 72,1%.
    - **Analyse** : Le Recall sur l'ensemble de test est exceptionnellement élevé (97,6%), mais l'accuracy est relativement faible (72,1%). Cela suggère que le modèle surestime probablement beaucoup de talents qui ne seront pas réussis (faux positifs), ce qui gonfle artificiellement le Recall.

Conclusion : Le modèle pourrait générer trop de faux positifs, ce qui n'est pas idéal.



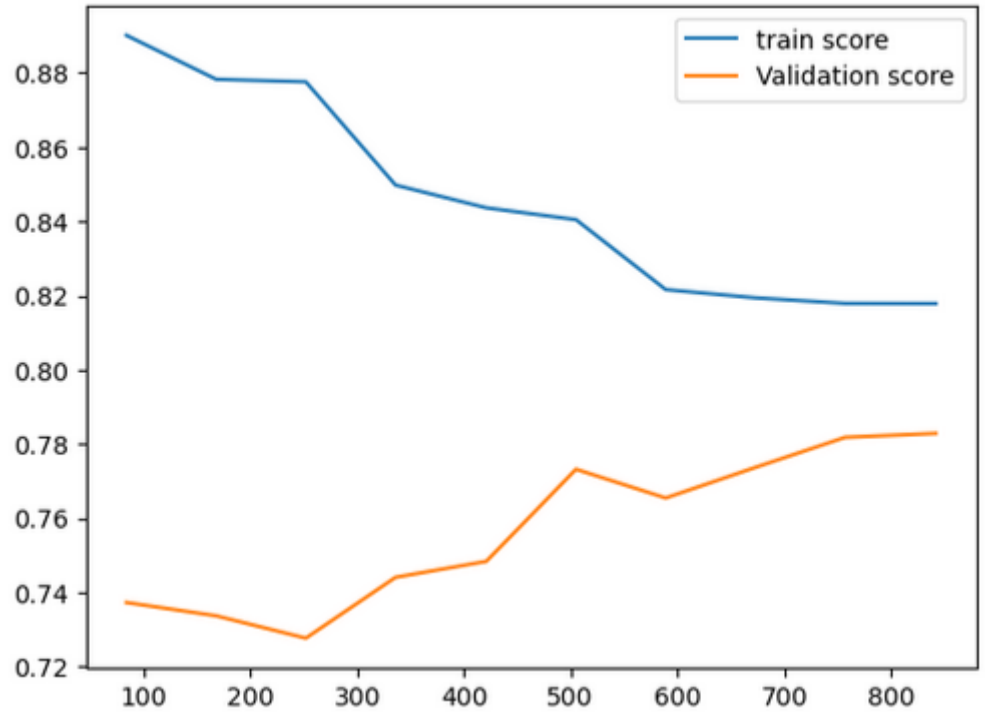
2. **Random Forest**
    - Train : Recall de 86,5%, F1-score de 77,1%, Accuracy de 68,1%.
    - Test : Recall de 93,3%, F1-score de 81,9%, Accuracy de 74,3%.
    - **Analyse** : Le modèle Random Forest est un peu mieux équilibré que le Decision Tree. Le Recall sur l'ensemble de test est très bon (93,3%), et l'accuracy est légèrement supérieure (74,3%). Toutefois, l'écart entre le Recall du train et du test (86,5% vs. 93,3%) suggère également un risque de surajustement, bien que ce soit moins prononcé que dans le cas de l'arbre de décision.

Conclusion : C'est un bon modèle pour maximiser le Recall, mais il pourrait encore surestimer quelques faux talents.



3. **Gradient Boosting**
    - Train : Recall de 87,3%, F1-score de 78,3%, Accuracy de 70,0%.
    - Test : Recall de 95,2%, F1-score de 85,3%, Accuracy de 79,6%.
    - **Analyse** : Le Gradient Boosting offre un excellent compromis. Le Recall sur le test est élevé (95,2%) avec un F1-score de 85,3%, ce qui montre que ce modèle équilibre mieux la détection des talents tout en minimisant les faux talents (faux positifs). L'accuracy de 79,6% suggère également que ce modèle est globalement plus fiable.

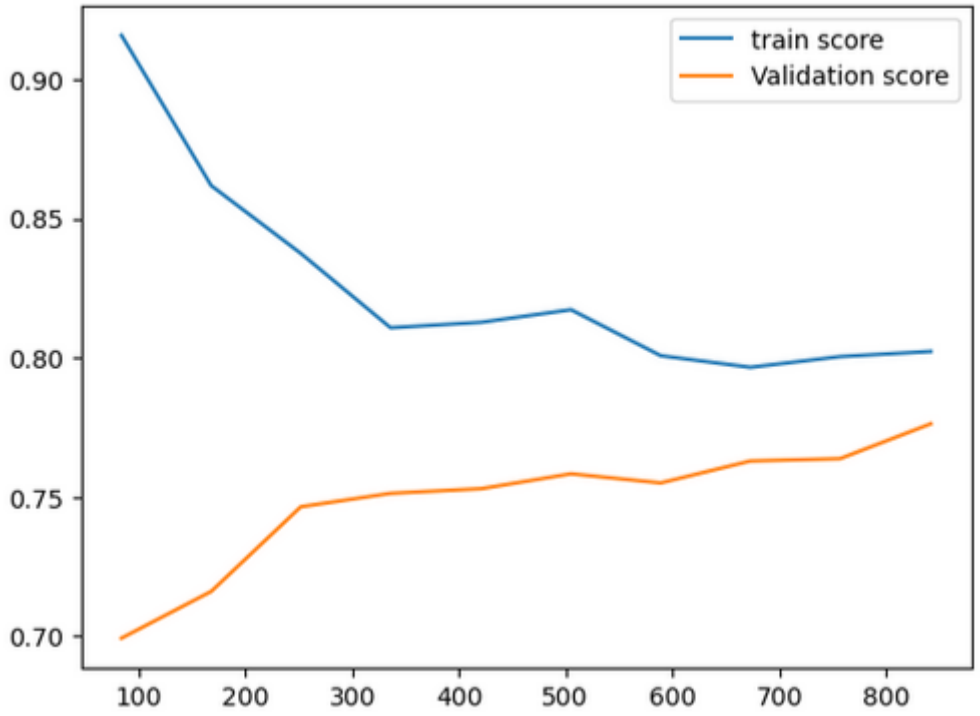
    Conclusion : Le Gradient Boosting semble bien répondre à l'objectif d'un Recall élevé tout en réduisant les faux talents.



4. **AdaBoost**

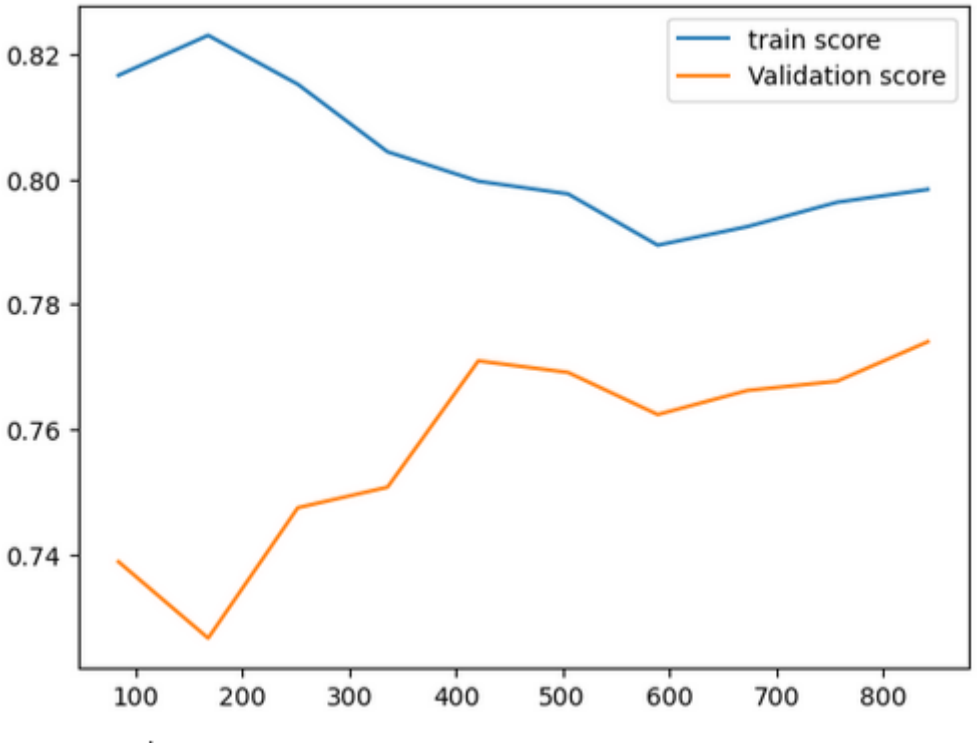
- Train : Recall de 82,4%, F1-score de 77,8%, Accuracy de 70,8%.
- Test : Recall de 89,1%, F1-score de 86,2%, Accuracy de 82,3%.
- **Analyse** : AdaBoost montre des résultats solides avec un F1-score élevé sur le test (86,2%) et une bonne accuracy (82,3%). Le Recall est de 89,1%, ce qui est un peu inférieur à celui de Gradient Boosting, mais l'accuracy est légèrement meilleure. Cela pourrait indiquer que le modèle est moins biaisé vers la surestimation des faux talents, tout en maintenant une bonne capacité à détecter les talents prometteurs.

Conclusion : AdaBoost offre une performance équilibrée avec un bon F1-score, mais un Recall légèrement inférieur par rapport à Gradient Boosting.



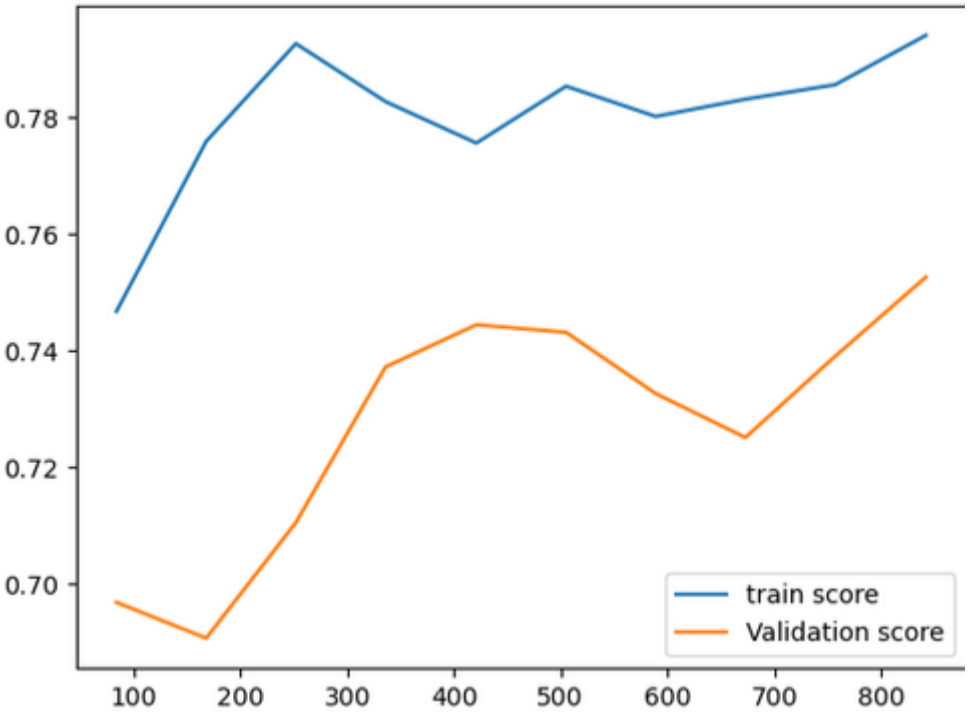
5. **SVM**
   - Train : Recall de 82,7%, F1-score de 77,4%, Accuracy de 70,0%.
   - Test : Recall de 83,6%, F1-score de 82,6%, Accuracy de 78,1%.
   - **Analyse** : Le SVM montre des résultats plus modérés. Le Recall est plus faible (83,6%) par rapport aux autres modèles, et bien que le F1-score soit bon (82,6%), il est plus proche de l'équilibre entre Recall et précision. Cela signifie que le modèle est moins agressif dans la détection des talents (moins de faux positifs), mais cela peut aussi indiquer qu'il manque certains talents.

Conclusion : Ce modèle pourrait être moins adapté si l'objectif est de maximiser le Recall.



6. **KNN**
   - Train : Recall de 79,4%, F1-score de 75,3%, Accuracy de 67,8%.
   - Test : Recall de 83,0%, F1-score de 81,1%, Accuracy de 75,8%.
   - **Analyse** : Le modèle KNN montre un Recall plus faible (83,0%) et un F1-score modéré (81,1%). Bien que ce modèle fonctionne correctement, il ne semble pas aussi performant que les autres modèles pour maximiser la détection des talents prometteurs.

Conclusion : KNN est un modèle moins adapté pour cet objectif.



Remarque

La majorité des modeles on leurs courbes d'apprentissage en hausse et ont un potentiel de continuer a

augmenter en rajoutant plus de données. Plus de données pourront potentiellement ameliorer la performance de ces différents modeles.

Conclusion

Le modèle le plus satisfaisant, selon les resultats obtenus, est le **AdaBoost** avec une frontiere de decision à **-0.033131**.

Caractéristiques :

- **Recall élevé:** Le Recall sur le jeu de test est d'environ 94,5%. Cela signifie que le modèle capture efficacement la majorité des bons talents, ce qui est le critère principal.
- **F1-score élevé:** Le F1-score est de 86,4%, ce qui indique que ce modèle équilibre bien la détection des talents (Recall) tout en limitant les faux talents surestimés (précision). C'est le meilleur F1-score parmi toutes les options, ce qui montre un bon compromis entre la précision et le Recall.
- **Accuracy solide:** L'accuracy sur le test atteint 81,5%. Cela montre que le modèle a une bonne capacité à généraliser sur de nouvelles données.