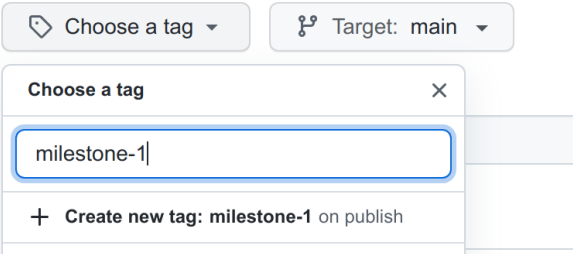


Projet IFT 6758 : Étape 2

Sortie : Oct 19, 2023

Date limite : Nov 13, 2023

Important : Avant de commencer à travailler sur l'étape 2, publiez une version de l'étape 1 en suivant les instructions [ici](#) et ajoutez @ift6758-2023 en tant que collaborateur github. Vous n'avez pas besoin d'ajouter quoi que ce soit dans la description ou de télécharger des binaires. Utilisez `milestone-1` à la fois comme (*tag*) et titre de version *release*, par exemple :



The screenshot shows the GitHub Releases page for a repository. At the top, there are two buttons: 'Choose a tag' and 'Target: main'. Below them, a dropdown menu is open, showing 'Choose a tag' with a close button. The dropdown contains a search bar with 'milestone-1' entered and a button '+ Create new tag: milestone-1 on publish'. Below the dropdown, there are tabs for 'Releases' and 'Tags'. The 'Releases' tab is active. Below the tabs, there is a section for 'milestone-1' with a 'Target: main' button. A message says 'Excellent! This tag will be created from the target when you publish this release.' Below this, there is a text input field with 'milestone-1'. There are tabs for 'Write' and 'Preview'. The 'Write' tab is active. Below the tabs, there is a rich text editor with various formatting options (H, B, I, list, link, code, quote, table, @, image, undo, redo) and a button '+ Auto-generate release notes'. The text area is empty with the placeholder 'Describe this release'. Below the text area, there is a dashed line and a message 'Attach files by dragging & dropping, selecting or pasting them.' Below this, there is a large dashed box with a downward arrow and the message 'Attach binaries by dropping them here or selecting them.' At the bottom, there is a checkbox 'This is a pre-release' which is unchecked. Below the checkbox, there is a message 'We'll point out that this release is identified as non-production ready.' At the very bottom, there are two buttons: 'Publish release' and 'Save draft'.

milestone-1

Write Preview

Describe this release

Attach files by dragging & dropping, selecting or pasting them.

Attach binaries by dropping them here or selecting them.

☐ This is a pre-release
We'll point out that this release is identified as non-production ready.

Publish release Save draft

Le premier étape vous a permis d'acquérir de l'expérience dans la gestion des données, l'analyse exploratoire des données et la création de visualisations. Maintenant, nous allons nous salir les mains avec l'ingénierie des caractéristiques, la sélection des caractéristiques et la modélisation statistique. Vous continuerez à vous appuyer sur les données de jeu acquises à partir de l'[API de statistiques NHL](#) et à résoudre le problème de la prédiction de la probabilité qu'un tir (*shot*) soit un but (*goal*). Vous essayerez un certain nombre de modèles et de caractéristiques différentes et produirez des visualisations pour évaluer l'efficacité de vos modèles. Vous utiliserez [comet.ml](#) pour suivre vos expériences avec votre groupe. Vous présenterez à nouveau votre travail dans un format d'article de blog, en ajoutant un autre article à votre blog existant que vous avez soumis pour l'étape 1.

Une note sur le plagiat	3
Objectifs d'apprentissage	3
Contexte LNH : Buts espérés (xG)	4
Suivi des expériences	5
Configuration de Comet.ml	6
Livrables	7
Détails de la soumission	7
Tâches et questions	8
0. Article de blog	8
1. Suivi des expériences (10%)	8
2. Ingénierie des caractéristiques I (10%)	9
3. Modèles de base (15%)	10
4. Ingénierie des caractéristiques II (15% + bonus 5%)	12
5. Modèles avancés (20%)	15
6. Faites de votre mieux! (20 %)	16
7. Évaluer sur l'ensemble de test (10 %)	17
Évaluations de groupe	18
Comment les scores impactent votre note	19
Références utiles	19
Annexe	21

Une note sur le plagiat

L'utilisation de code/modèles provenant de ressources en ligne est acceptable et c'est courant dans la science des données, mais soyez clair pour citer exactement d'où vous avez pris le code si nécessaire. Un simple extrait d'une ligne qui couvre une syntaxe simple à partir d'un article ou d'un package Stack Overflow ne justifie probablement pas une citation, mais la copie d'une fonction qui effectue un tas de logique qui crée votre figure le fait. Nous espérons que vous pourrez utiliser votre meilleur jugement dans ces cas, mais si vous avez des doutes, vous pouvez toujours citer quelque chose pour être sûr. Nous effectuerons une détection de plagiat sur votre code et vos livrables, et il vaut mieux prévenir que guérir en citant vos références.

L'intégrité est une attente importante de ce projet de cours et tout cas suspect sera poursuivi conformément à la [politique très stricte](#) de l'Université de Montréal. Le texte complet des règlements à l'échelle de l'université peut être trouvé [ici](#). Il est de la responsabilité de l'équipe de s'assurer que cela est suivi rigoureusement et des actions peuvent être prises sur des individus ou sur l'ensemble de l'équipe selon les cas.

Objectifs d'apprentissage

- **Ingénierie des caractéristiques**
 - Créer et sélectionner des caractéristiques à utiliser dans les modèles en aval
 - Encoder/transformer de manière appropriée les caractéristiques qui ne sont pas directement utilisables dans les modèles ML
- **Modélisation statistique et apprentissage automatique**
 - Expérimenter avec des classificateurs couramment utilisés et différentes combinaisons de caractéristiques pour essayer de produire un bon classificateur xG
 - Gagner une compréhension des métriques couramment utilisées telles que les courbes ROC et AUC
 - Produire des figures pour justifier vos résultats
- **Reproductibilité et suivi des expériences**
 - Utiliser de bonnes pratiques avec le suivi des expériences

Contexte LNH : Buts espérés (xG)

Une question courante lors de l'évaluation du sport est « à quel point une équipe a-t-elle bien joué? », ou « quelle est la performance d'un joueur en particulier? ». Pour répondre à de telles questions, nous avons besoin d'une sorte de mesure ou de statistique qui capture la qualité de la performance d'une équipe/joueur. Une mesure évidente pourrait être de prendre en compte le nombre de buts ou de points, mais le problème est que les buts sont des événements rares et « bruyants », qui ne tiennent pas compte d'autres facteurs tels que la performance du gardien de but. Les tirs sont une autre option, mais il n'y a aucune notion de la qualité du tir; c'est-à-dire provient-il d'une zone de haut danger ou est-ce qu'il était éloigné du filet ?

En conséquence, le simple fait de regarder un nombre total de tirs ou de buts peut être trompeur. Pour tenter de dépasser ces problèmes, les analystes se sont tournés vers des techniques de modélisation statistique pour ajouter la notion de « qualité de tir » à chaque tir.

Les buts espérés (*expected Goals* - xG) estiment la qualité d'un tir en calculant la probabilité que ce tir aurait abouti à un but, en fonction de divers facteurs (caractéristiques) tels que la position, le type de tir, les événements antérieurs, etc. Cela tente de modéliser comment les téléspectateurs expérimentés qui regardent un jeu peuvent raisonnablement évaluer la qualité d'un tir, en fonction de divers facteurs tels que l'état du jeu, la proximité du tir, etc.

Du point de vue de la science des données, nous pouvons modéliser intuitivement cela comme une tâche de **classification binaire**, où chaque tir a un label indiquant s'il a ou non abouti à un but. Chaque coup peut avoir une variété de caractéristiques allant de caractéristiques simples telles que la distance et l'angle du filet à des caractéristiques plus compliquées telles que l'état du jeu, les événements précédents, les types de coups, etc. C'est important de noter que même si le classifieur produit une étiquette binaire $[0, 1]$ pour indiquer si un tir est prévu ou non comme un but, cela ne nous est en fait pas très utile. Nous sommes plus intéressés par les **probabilités brutes** que le classificateur produit, qui (nous l'espérons) est un proxy approprié pour la « qualité du tir ». Les analystes peuvent ensuite agréger les tirs pondérés par la qualité des tirs dans le but d'évaluer les performances d'une équipe ou d'un individu (bien que cela ne fasse pas partie de cette étape du projet). De plus, un tir qui a une forte probabilité mais qui n'a pas abouti à un but (ou vice-versa) peut ne pas être mal classé. Cela peut être un peu contre-intuitif, mais rappelez-vous que nous essayons d'estimer la **qualité du tir** dans le but d'estimer le nombre total de chances de marquer de haute qualité.

Suivi des expériences



Une situation courante dans laquelle nous nous trouvons lors de l'exécution de nombreuses expériences est de perdre la trace de l'historique du modèle associé à des modèles particuliers. Il existe nombreux outils qui tentent de résoudre ce problème, comme [MLFlow](#), [Weights and Biases](#), [comet.ml](#), [neptune.ai](#), entre autres. Pour les besoins de ce cours, nous utiliserons [comet.ml](#) car il est simple d'utilisation, et gratuit pour les équipes de 5 dans le cas d'un usage académique. Les autres offres sont assez comparables, ce choix était donc largement arbitraire.

Comet.ml (et de nombreuses autres offres) sont des services en ligne qui fournissent des fonctionnalités telles que le suivi de l'historique des modèles, le suivi des expériences/métadonnées, les magasins d'artefacts et les registres de modèles. À un niveau élevé, cela fournit un endroit centralisé pour stocker toutes vos métadonnées d'expérience telles que les configurations, les hyperparamètres, les hashes de jeux de données et les modèles enregistrés. Ces services fournissent généralement un SDK Python très simple qui vous permet d'enregistrer facilement des informations sur leurs serveurs cloud. À un niveau élevé, vous définissez généralement un objet d'*expérimentation*, qui regroupera toutes les données associées que vous avez peut-être enregistrées, telles que les hyperparamètres, les métriques/pertes de modèle, les actifs, les images, etc. Voici un exemple simple pour vous donner un aperçu de quoi ça a l'air de faire un suivi de ses expériences :

```
# Importer comet_ml en haut de votre fichier, avant sklearn !
from comet_ml import Experiment
import os

# Créer une expérience avec votre clé api
exp = Experiment(
    api_key=os.environ.get('COMET_API_KEY'), #ne pas coder en
    dur!
    project_name='milestone_2',
    workspace=<YOUR_WORKSPACE>,
)
# ... Faire de la science des données ...
exp.log_metrics({"auc": auc, "acc": acc, "loss": loss})
```

Pour des informations détaillées, Comet.ml fournit de nombreux guides de démarrage rapide qui seront utiles ; ceux-ci peuvent être trouvés dans leurs [pages de documentation](#). Ils ont des exemples dédiés pour [scikit-learn](#), [pytorch](#), [matplotlib](#), entre autres.

Configuration de Comet.ml

Voici une liste de contrôle rapide pour commencer à utiliser Comet.ml :

1. (Tous les membres de l'équipe) Créez un compte avec votre **courriel académique** via [ce lien](#).
 - a. Assurez-vous de réclamer le *free academic tier* disponible
2. (Tous les membres de l'équipe) Réclamez votre niveau académique gratuit via [ce lien](#).
3. Désignez un membre de l'équipe pour héberger l'espace de travail de votre projet. Ce membre de l'équipe ajoutera `ift6758-2023` (le compte TA) à l'espace de travail comet.ml ; cela peut être trouvé en allant dans les paramètres et en cliquant sur **+ Collaborators**. Vous devriez voir l'invite suivante :

Add Collaborators

Invite by email

Enter email Add

Username Role

ift6758 Member ▼ Add

4. Commencez à utiliser l'espace de travail pour consigner toutes les métadonnées de votre expérience et enregistrez/enregistrez les modèles
5. Reproductibilité !¹

Une note importante : pour écrire dans un espace de travail *comet*, vous devrez spécifier une **api_key**, qui est unique à votre compte *comet*. **NE PAS** coder cela en dur dans votre fichier python; vous pouvez accidentellement le valider dans votre dépôt git, ce qui signifie que n'importe qui peut désormais écrire dans cet espace de travail. Un modèle courant pour gérer les secrets par programmation consiste à enregistrer votre clé API dans une variable d'environnement, qui peut être stockée en toute sécurité sur votre machine, puis à y accéder par programmation via Python via le package `os`. Par exemple, si votre équipe accepte tous de stocker vos clés privées dans la `COMET_API_KEY` variable d'environnement sur chacun de vos ordinateurs, vous pouvez les récupérer en toute sécurité via :

```
my_key = os.environ.get("COMET_API_KEY")
```

Pour cette étape, vous serez demandé d'utiliser Comet.ml pour suivre vos expériences. Nous ne serons pas très pointilleux sur la façon dont tout est formaté exactement dans votre espace de travail de comète, mais nous chercherons à nous assurer que les expériences menant à un modèle pertinent (c'est-à-dire que ce modèle a abouti à des graphiques dans le billet de blog)

¹ D'accord, ce n'est pas si facile, mais le suivi des expériences est un bon pas dans cette direction.

peuvent être trouvées dans votre projet *comet*. Nous mettrons en évidence les exigences dans la section des questions où nous nous attendons à voir des liens vers des entrées de test.

Livrables

Vous devez soumettre **LES DEUX** :

1. Rapport de style de publication de blog sur une **nouvelle publication** (vous pouvez conserver ou omettre la publication pour ÉTAPE1)²
2. La base de code de votre équipe **qui est reproductible**
3. Ajoutez les comptes IFT 6758 pertinents à votre dépôt github et à votre espace de travail comet.ml.

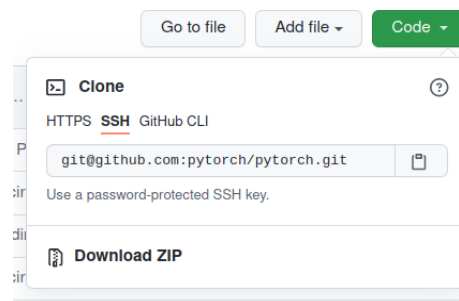
Comme auparavant, au lieu d'un rapport traditionnel écrit en LaTeX, il vous sera demandé de soumettre un article de blog qui contiendra des points de discussion et des figures. Créer **une nouvelle publication/entrée** pour Milestone 2 ; vous pouvez choisir de conserver ou d'omettre la publication d'origine de l'étape1. Pour rappel, bien que nous ne déploierons pas ces pages au public, [il est très simple d'utiliser les pages github pour publier votre contenu](#). Vous êtes plus que bienvenus pour le faire à la fin du cours!

Détails de la soumission

Pour soumettre votre projet, vous devez :

- ☐ Publier votre soumission d'étape finale dans la **milestone-2** branche
 - ☐ Créer une version similaire à ce que vous avez fait pour l'étape 1, mais avec la **milestone-2** balise.
- ☐ Remettre un ZIP de votre article de blog sur [Gradescope](#)
- ☐ Remettre un ZIP de votre base de code sur [Gradescope](#)
- ☐ Ajoutez [le compte GitHub IFT 6758 TA \(@ift6758-2023\)](#) à votre git en tant que *spectateur* ou *contributeur*, si vous ne l'avez pas déjà fait (ou utilisez un nouveau dépôt).
- ☐ Ajouter **ift6758-2023** à votre espace travail comet.ml [+ Collaborators](#) (Paramètres>)

Pour remettre un ZIP de votre dépôt, vous pouvez télécharger via l'interface utilisateur Github:



² Étant donné que la difficulté des visualisations avancées de l'étape 1 peut avoir entraîné des graphiques extrêmement importants (en termes de mémoire), vous êtes permis d'omettre les visualisations avancées de votre article de blog d'origine, si vous choisissez de soumettre votre blog avec les deux articles.

Rappelez-vous que cette méthode ne télécharge pas le dépôt ensemble git, juste la **branche que vous êtes** actuellement. Assurez-vous que tout votre code est dans la version que vous avez créée avant de télécharger les ZIP, et que le zip a la balise de version dans le nom de fichier.

Tâches et questions

Les tâches requises pour l'étape 2 sont décrites ici. La description globale de ce qui est requis est décrite au début de chaque tâche. La **Questions** section de chaque tâche décrit le contenu requis dans le billet de blog. Cela peut être soit des questions d'interprétation, soit vous devrez peut-être produire des figures ou des images à inclure dans le billet de blog. Nous essayons d'écrire la plupart des choses qui sont requises dans le rapport en **gras**, mais assurez-vous de répondre à tout ce qui vous est demandé dans chaque question. Nous n'attendons pas de longues réponses aux questions, dans la plupart des cas quelques phrases suffiront.

0. Article de blog

Encore une fois, créez un autre article de blog à l'aide du modèle fourni contenant toutes les figures, réponses et discussions requises mentionnés dans la section précédente. Vous n'aurez pas besoin de déployer réellement l'article de blog pour qu'il soit accessible au public, mais des instructions seront fournies pour savoir comment procéder si vous souhaitez montrer votre projet génial sur votre CV une fois le cours terminé!

1. Suivi des expériences (10%)

Comme décrit ci-dessus, vous utiliserez Comet.ml pour suivre vos expériences. Suivez les [instructions pour configurer Comet.ml](#) pour votre projet. Une fois configuré, utilisez comet pour suivre vos expériences. L'élaboration d'un workflow qui a du sens pour votre équipe peut prendre un peu de temps, alors n'hésitez pas à jouer avec différentes manières d'organiser vos journaux d'expériences et n'ayez pas peur de générer beaucoup de déchets. Le but de cette section est de vous exposer à l'idée du suivi/enregistrement des expériences, et cela peut sembler un peu lourd au début si vous ne l'avez jamais fait auparavant. **Une suggestion est de ne pas utiliser les notebooks Jupyter pour exécuter vos expériences de modèle.** Les blocs-notes sont parfaits pour le prototypage et la visualisation de choses, mais ils ne sont pas parfaits pour une exécution par programmation. De plus, Comet.ml fonctionne beaucoup mieux avec des scripts Python appropriés qu'avec des cahiers interactifs. Certes, cela peut sembler un peu exagéré pour ce cas d'utilisation particulier, mais il est bon d'acquérir de l'expérience avec de meilleures pratiques avant de travailler dans le domaine de la science des données dans l'industrie ou de poursuivre vos recherches.

Les notes attribuées à cette section sont basées sur la satisfaction des exigences associées au suivi des expériences tout au long de l'étape. Vous êtes libre de choisir la façon dont vous formatez vos journaux d'expériences, mais gardez à l'esprit que l'intérêt d'utiliser cet outil est de faciliter la reproductibilité. Cela signifie qu'il devrait y avoir suffisamment d'informations pour savoir comment un modèle particulier a été créé. Dans les tâches suivantes, nous demanderons

explicitement un lien vers l'expérimentation correspondant à certaines parties de votre article de blog.

Tout au long de l'étape, il vous sera également demandé d'enregistrer des modèles pour les tester sur un ensemble de données d'attente final. Pour faciliter cela, vous utiliserez la [fonction `model registry`](#) pour inscrire et sauvegarder vos modèles. Vous pouvez utiliser l'API `experiment.log_model()` fournie par Comet.ml pour sauvegarder et récupérer les modèles de votre espace de travail. Cela vous obligera généralement à enregistrer d'abord votre modèle sur le disque avant de transmettre le chemin du modèle enregistré à cette fonction : vous pouvez trouver des détails sur l'enregistrement des modèles scikit-learn [ici](#) et des modèles XGBoost [ici](#). Assurez-vous de lire la documentation pertinente de la librairie que vous utilisez pour une tâche particulière.

Questions

- Aucune. Dans les tâches suivantes, il vous sera demandé de référencer des entrées d'expérience dans votre espace de travail comet.ml qui sont liées aux modèles qui ont produit des figures spécifiques.

2. Ingénierie des caractéristiques I (10%)

Utilisant la fonctionnalité créée pour *Étape 1*, acquérez toutes les données *brutes* de jeu par jeu (*play-by-play*) pour la saison 2015/16 jusqu'à la saison 2019/20 (inclus). Notez que les données ordonnées que vous avez créées seront utiles pour les modèles de base, mais vous créerez davantage de caractéristiques qui nécessiteront les données brutes complètes de la Partie 4.

Mettez de côté toutes les données 2019/20 comme l'ensemble final des données *test*. Vous ne devez pas y toucher avant d'avoir atteint la fin de cette étape. Vous utiliserez les données 2015/16 - 2018/19 de la *saison régulière* pour créer vos ensembles d'entraînement et de validation. À la fin de cette étape, vous aurez mis en place tout le cadre dont vous avez besoin pour traiter de nouvelles données, il devrait donc être trivial de traiter vos données de test une fois que vous avez terminé toute l'ingénierie des caractéristiques nécessaire. Jusqu'à la Partie 7, **toute référence au «données» se référera exclusivement aux données 2015/16 - 2018/19.**

Questions

1. Utilisant vos données d'entraînement créez un jeu de données propre pour chaque événement SHOT/GOAL, avec les colonnes suivantes (vous pouvez les nommer comme vous voulez):
 - Distance du filet
 - Angle relatif au filet
 - est un but (0 ou 1)
 - Filet vide (0 ou 1, vous pouvez supposer que les NaN sont 0)

Vous pouvez approximer le filet en un seul point (c'est-à-dire que vous n'avez pas besoin de tenir compte de la largeur du filet lors du calcul de la distance ou de l'angle).

Vous devriez être capable de le créer facilement à l'aide de la fonctionnalité que vous avez implémentée pour ranger les nettoyer dans *Étape 1*, car vous n'aurez besoin que des coordonnées (x, y) pour chaque événement de tir. **Créez et intégrez les figures suivants dans votre blogpost et discutez brièvement vos observations (quelques phrases):**

- Un histogramme du nombre de tirs (buts et non-buts séparés), regroupées (*binned*) par distance
- Un histogramme de nombre de tirs (buts et non-buts séparés), *binned* par angle
- Un histogramme 2D où un axe est la distance et l'autre est l'angle. Vous n'avez pas besoin de séparer les buts et les non-buts.
 - *Astuce : consultez les [jointplots](#).*

Comme toujours, assurez-vous que tous vos axes sont correctement étiquetés et que vous faites le bon choix d'échelle d'axe.

2. Maintenant, créez deux autres figures reliant le taux de but, c'est-à-dire $\#buts / (\#pas_de_buts + \#buts)$, à la distance et le taux de but à l'angle du tir. **Incluez ces figures dans votre article de blog et discutez brièvement de vos observations.**
3. Enfin, effectuons quelques vérifications rapides pour voir si nos données ont du sens. Malheureusement, nous n'avons pas le temps de faire une détection automatisée des anomalies, mais nous pouvons utiliser notre « connaissance du domaine » pour des vérifications rapides de l'intégrité ! La connaissance du domaine est qu'*«il est incroyablement rare de marquer un but net non vide sur l'équipe adverse depuis l'intérieur de votre zone défensive³»*. Sachant cela, **créez un autre histogramme**, cette fois de buts uniquement, classés par distance, et séparez les événements nets vides et non vides. **Incluez ce chiffre dans votre article de blog et discutez de vos observations.** Pouvez-vous trouver des événements qui ont des caractéristiques incorrectes (par exemple, de mauvaises coordonnées x/y) ? Si oui, prouvez qu'un événement a des caractéristiques incorrectes.
 - *Astuce : le [gamecenter de LNH](#) propose généralement des clips vidéo des buts pour chaque match.*

3. Modèles de base (15%)

Vous allez maintenant utiliser les caractéristiques de distance et d'angle que vous avez produites dans la section précédente pour produire quelques modèles simples afin d'avoir une idée de ce à quoi vous pouvez vous attendre en tant que performances de base, avant d'approfondir caractéristiques et modèles dans la section suivante. Ne vous embêtez pas à passer beaucoup de temps à accorder vos modèles, le but est de développer une intuition de base sur le problème.

Questions

³ Technically, goals from outside the offensive zone are generally very unlikely, but for the sake of brevity, focus on non-empty net goals on the opposing team taken from within your team's defensive zone.

1. À l'aide de votre ensemble de données (rappelez-vous, ne touchez pas à l'ensemble de test !), créez une division d'entraînement et de validation comme bon vous semble. Utilisant seulement la caractéristique `distance`, entraînez un classifieur [Logistic Regression](#) avec les paramètres complètement par défaut, c'est-à-dire:

```
clf = LogisticRegression()
clf.fit(X, y)
```

Évaluez la précision (accuracy) (c'est-à-dire correctement prédite/totale) de votre modèle sur l'ensemble de validation. Que remarquez-vous? Regardez les prédictions et discutez de vos découvertes. Quel pourrait être un problème potentiel? **Incluez ces discussions dans votre article de blog.**

2. Sur la base de vos conclusions de la première question, nous devrions explorer d'autres façons d'évaluer notre modèle. La première chose à noter est que nous ne nous intéressons pas réellement à la prédiction binaire de savoir si un tir est un but ou non - nous nous intéressons à la probabilité que le modèle lui attribue (rappelons que nous nous intéressons à la notion des **butts espérés**) que Scikit-learn vous le fournit via la méthode [predict_proba\(...\)](#) ; assurez-vous de prendre les probabilités de la classe qui vous intéresse ! Produisez **quatre** graphiques (une courbe par modèle par graphique) pour sonder les performances de notre modèle. Assurez-vous d'utiliser les probabilités obtenues sur l'ensemble de validation:
 - a. Les courbes [Receiver Operating Characteristic \(ROC\)](#) et [la métrique AUC](#) de la courbe ROC. Incluez une ligne de base de classificateur aléatoire, c'est-à-dire que chaque tir a 50 % de chances d'être un but.
 - b. Le taux de buts ($\#buts / (\#non_buts + \#buts)$) comme une fonction du centile de la probabilité de tir donnée par le modèle, c'est-à-dire que si une valeur est au 70e centile, elle est supérieure à 70% des données.
 - c. La proportion cumulée de buts (pas de tirs) comme une fonction du centile de la probabilité de tir donnée par le modèle,
 - d. Le [diagramme de fiabilité](#) (courbe de calibration). Scikit-learn fournit des fonctionnalités pour créer un diagramme de fiabilité en quelques lignes de code ; consultez l'API [CalibrationDisplay](#) (en particulier les méthodes `.from_estimator()` ou `.from_predictions()`) pour plus d'informations.

Un exemple de ce qui est attendu pour (b) et (c) est illustré à la figure 1. N'incluez pas tout cela dans votre article de blog pour le moment, car vous ajouterez quelques courbes supplémentaires dans la section suivante.

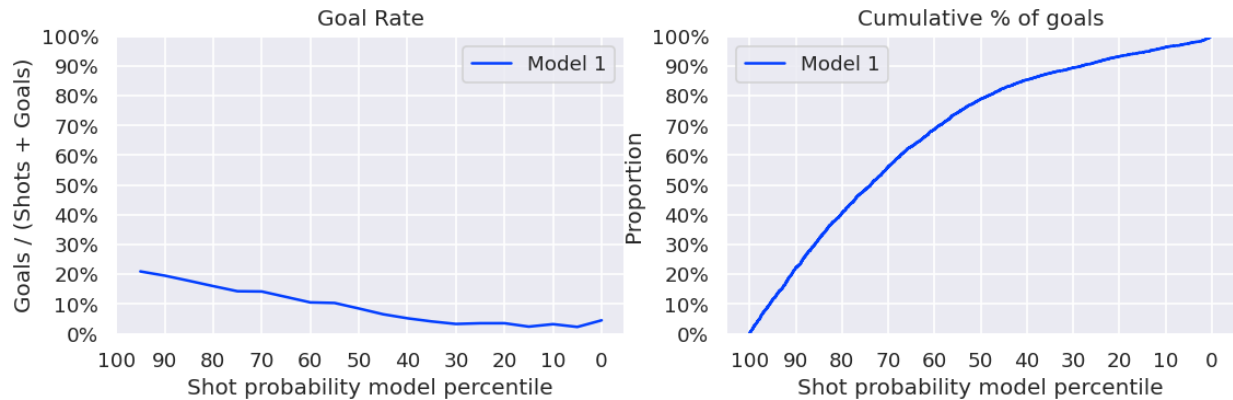


Fig 1 : Taux de buts et proportion cumulée de buts.

3. Entraînez maintenant deux autres classificateurs de régression logistique en utilisant la même configuration que ci-dessus, mais cette fois sur la caractéristique d'angle, puis ensuite à la fois sur les caractéristiques de distance et d'angle. Produisez les trois mêmes courbes que celles décrites dans la section précédente pour chaque modèle. Y compris la ligne de base aléatoire, vous devriez avoir un total de 4 lignes sur chaque figure :
 - Régression logistique, entraînée sur la `distance` uniquement (déjà fait ci-dessus)
 - Régression logistique, entraînée sur l' `angle` uniquement
 - Régression logistique, entraînée sur la `distance` et l' `angle`
 - Ligne de base aléatoire : probabilité prédite est échantillonné à partir d'une distribution uniforme, c'est-à-dire $\tilde{y}_i \sim U(0, 1)$

Incluez ces quatre figures (chacune avec quatre courbes) dans votre article de blog. En quelques phrases, **discutez de votre interprétation de ces résultats.**

4. À côté des figures, **incluez des liens vers les trois entrées d'expérience dans vos projets comet.ml qui ont produit ces trois modèles. Enregistrez les trois modèles dans les trois expériences sur comet.ml (exemple [ici](#))** et ajoutez des tags informatifs, car vous en aurez besoin pour la section finale.

4. Ingénierie des caractéristiques II (15% + bonus 5%)

Maintenant que nous avons des résultats de base, nous pouvons essayer quelques caractéristiques supplémentaires. Vous allez inclure un certain nombre de caractéristiques qui sont déjà disponibles dans votre jeu de données précédemment nettoyées de l'étape 1, ainsi que d'ajouter de nouvelles fonctionnalités qui incluent des informations provenant d'autres caractéristiques.

Questions

Notez que vous pouvez donner à ces caractéristiques les noms informatifs que vous souhaitez.

- À partir des données déjà triées, incluez les caractéristiques suivantes :
 - **Secondes de jeu** (*Game seconds*)
 - **Période de jeu** (*Game period*)
 - **Coordonnées** (x,y, colonnes séparées)
 - **Distance de tir** (*Shot distance*)
 - **Angle de tir** (*Shot angle*)
 - **Type de tir** (*Shot type*)
- Maintenant, à chaque tir, ajoutez des informations sur les événements précédents. Notez qu'il peut s'agir de n'importe quel événement, pas seulement de tirs. Cela signifie que vous devrez probablement mettre à jour votre code qui a produit les données propres pour maintenant également examiner d'autres types d'événements. Comme rappel, cet *endpoint* peut vous être utile :

<https://statsapi.web.nhl.com/api/v1/playTypes>

À chaque tir, ajoutez les informations suivantes de l'événement immédiatement précédent sous forme de quatre nouvelles caractéristiques:

- **Dernier type d'événement** (*Last event type*)
 - **Coordonnées du dernier événement** (x, y, colonnes séparées)
 - **Temps écoulé depuis le dernier événement** (secondes)
 - **Distance depuis le dernier événement** (*Distance from the last event*)
- Avec ces nouvelles informations, nous allons créer quelques caractéristiques supplémentaires qui tentent de quantifier quelques choses plus intéressantes sur l'état du jeu. Créez les trois caractéristiques suivantes :
 - **Rebond** (bool) : Vrai si le dernier événement était aussi un tir, sinon False
 - **Changement d'angle de tir**: Inclure seulement si le tir est un rebond, sinon 0.
 - **«Vitesse»** : définie comme la *distance depuis l'événement précédent*, divisée par le *temps écoulé* depuis l'événement précédent.

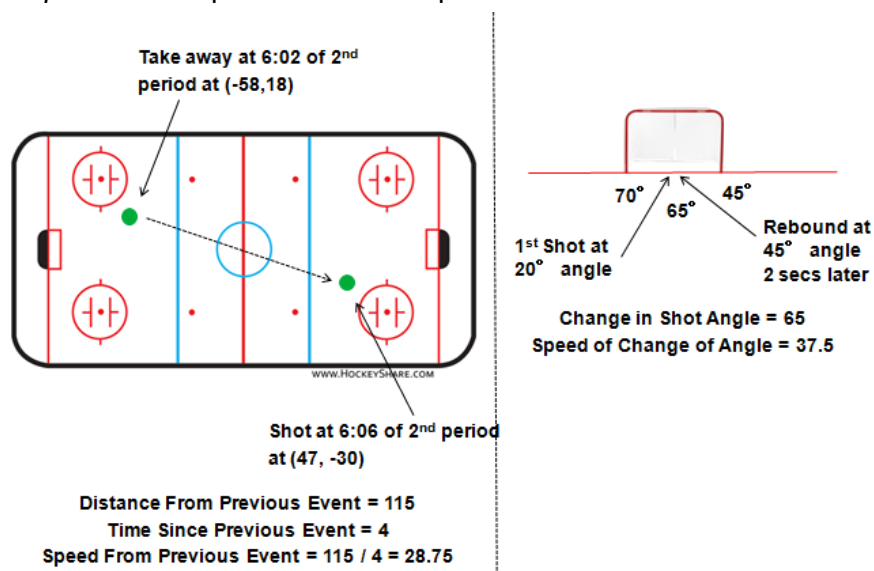


Fig 2: Les caractéristiques de *vitesse* et de *changement d'angle*, de moneypuck.com.

4. **(Bonus 5%)** En prime, calculez quelques caractéristiques concernant la situation de jeu de puissance (*power-play*). Incluez :
- **temps écoulé depuis le début du jeu de puissance** (secondes), réinitialisez-le à 0 lorsque le power play expire.
 - Notez qu'il existe des pénalités de deux minutes [mineures](#), 4 minutes [mineure double](#), et 5 minutes [majeures](#). Lisez la documentation pour comprendre comment chaque pénalité expire.
 - Les power-plays peuvent s'empiler, c'est-à-dire que si l'équipe pénalisée subit une autre pénalité, elle perd 2 joueurs jusqu'à l'expiration de la pénalité, puis 1 jusqu'à l'expiration de la seconde. Dans ce cas, ne réinitialisez pas la minuterie ; continuez simplement à compter jusqu'à ce que toutes les pénalités expirent.
 - **Nombre de patineurs non-gardiens amicaux sur la glace**
 - **Nombre de patineurs non-gardiens adverses sur la glace**

Ce sera un peu délicat, car vous devrez suivre chaque pénalité et déterminer quand la pénalité expire pour réinitialiser le nombre de joueurs sur la glace ainsi que le temps écoulé depuis le début du jeu de puissance. Vous avez accès à une liste de tous les événements de pénalité par index dans les données brutes:

```
https://statsapi.web.nhl.com/api/v1/game/[GAME_ID]/feed/live/
> liveData
  > plays
    > penaltyPlays
```

5. Comet.ml fournit certaines fonctionnalités pour suivre les ensembles de données, soit en tant qu'artefacts bruts que vous enregistrez manuellement, soit en téléchargeant automatiquement des DataFrame/tables pour vous. Il dispose également de fonctionnalités intéressantes pour calculer les hashes de jeux de données, vous permettant de vérifier si vous vous entraînez sur les mêmes données que les autres. Vous pouvez trouver plus d'informations sur cette fonctionnalité dans leur [documentation](#). Vous n'êtes pas obligé d'utiliser cette fonctionnalité pour la majorité de votre projet (vous pouvez si vous le souhaitez !). Cependant, aux fins de cette section, nous vous demandons de soumettre un petit sous-ensemble (un jeu spécifique) de votre cadre de données final avec toutes vos caractéristiques finales. Filtrez votre ensemble de données sur le [match Winnipeg vs Washington le 12 mars 2018](#). Ce jeu a l'ID de jeu "2017021065". **Téléchargez le DataFrame filtré avec toutes les caractéristiques que vous avez créées en tant que CSV à l'aide de la méthode `log_dataframe_profile(...)` ; gardez le nom comme `'wpg_v_wsh_2017021065'`.** Le code ici est un exemple de comment upload un jeu de données:

```
experiment = Experiment(
    api_key=os.environ.get('COMET_API_KEY'),
```

```

project_name='feature_engineering_data',
workspace=<YOUR_WORKSPACE>,
)
experiment.log_dataframe_profile(
    subset_df,
    name='wpg_v_wsh_2017021065', # keep this name
    dataframe_format='csv' # ensure you set this flag!
)

```

Dans votre article de blog, ajoutez une liste de toutes les caractéristiques que vous avez créées pour cette section. Répertoriez chaque caractéristique à la fois par le nom de la colonne dans votre cadre de données ET par une explication simple et lisible par un humain (par exemple game_sec: nombre total de secondes écoulées dans le jeu). Si vous avez créé de nouvelles caractéristiques, décrivez brièvement ce qu'elles sont. Ajoutez un lien vers l'expérience qui stocke l'artefact DataFrame filtré pour le jeu spécifié. Notez qu'il ne doit y avoir qu'un seul DataFrame inscrit avec ce nom. Voir l' [annexe](#) pour un exemple de ce que nous recherchons.

5. Modèles avancés (20%)

Maintenant que nous avons de nombreuses caractéristiques pour travailler avec, voyons si cela nous permet d'améliorer sur nos modèles de régression logistique simples dans la partie 3. Nous nous concentrerons sur les modèles [XGBoost](#) pour cette section; vous aurez le champ libre pour essayer ce que vous voulez dans la section suivante.

Questions

Pour chacune des questions suivantes, les quatre mêmes figures que dans la partie 3 seront utilisés comme mesures quantitatives :

- ROC/AUC
- Taux de buts vs percentile de probabilité
- Proportion cumulée de buts vs percentile de probabilité
- Courbe de fiabilité

Lorsqu'une question vous demande d'enregistrer les métriques du modèle, ajouter une courbe correspondante à chacune de ces quatre figures. Notez qu'il s'agit de quatre nouvelles figures, c'est-à-dire que vous n'avez pas besoin de superposer les courbes sur les mêmes figures que vous avez produites pour la partie 3.

1. Entraînez un classificateur XGBoost en utilisant le même ensemble de données en utilisant uniquement les caractéristiques de `distance` et d' `angle` (similaire à la partie 3). Ne vous inquiétez pas encore du réglage des hyperparamètres, cela servira simplement de comparaison avec la ligne de base avant d'ajouter plus de caractéristiques. **Ajoutez les courbes correspondantes aux quatre figures à votre article de blog. Discutez brièvement (quelques phrases) de votre configuration d'entraînement/validation et comparez les résultats à la référence de régression logistique. Incluez un lien vers l'entrée comet.ml appropriée pour cette**

expérience, mais vous n'avez pas besoin de consigner ce modèle dans le registre des modèles.

2. Maintenant, entraînez un classificateur XGBoost en utilisant toutes les caractéristiques que vous avez créées dans la [Partie 4](#) et effectuez quelques réglages d'hyperparamètres pour essayer de trouver le modèle le plus performant avec toutes ces caractéristiques. **Dans votre article de blog, discutez de votre configuration de réglage des hyperparamètres et incluez des figures pour justifier votre choix d'hyperparamètres.** Par exemple, vous pouvez sélectionner les métriques appropriées et effectuer une recherche par grille avec validation croisée. **Une fois réglé, intégrez les courbes correspondant au meilleur modèle aux quatre figures de votre article de blog et comparez brièvement les résultats au baseline XGBoost de la première partie. Incluez un lien vers l'entrée comet.ml appropriée pour cette expérience et enregistrez ce modèle dans le registre des modèles.**
3. Enfin, explorez l'utilisation de certaines techniques de sélection de caractéristiques pour voir si vous pouvez simplifier vos caractéristiques d'entrée. Un certain nombre de caractéristiques contiennent des informations corrélées, vous pouvez donc essayer de voir si certaines d'entre elles sont redondantes. Vous pouvez essayer certaines des techniques de sélection de caractéristiques discutées en classe; beaucoup d'entre eux sont [implémentés pour vous par scikit-learn](#). Vous pouvez également utiliser une librairie comme [SHAP](#) pour essayer d'interpréter les caractéristiques sur lesquelles votre modèle repose le plus. **Discutez des stratégies de sélection de caractéristiques que vous avez essayées et de l'ensemble de caractéristiques le plus optimal que vous avez proposé. Incluez quelques figures pour justifier vos affirmations.** Une fois que vous avez trouvé l'ensemble optimal de caractéristiques via le réglage des hyperparamètres /validation croisée, si l'ensemble de caractéristiques est différent de celui utilisé pour la Q2 de cette section, **incluez les courbes correspondant au meilleur modèle aux quatre figures de votre article de blog, et comparer brièvement les résultats à la référence XGBoost. Incluez un lien vers l'entrée comet.ml appropriée pour cette expérience et enregistrez ce modèle dans le registre des modèles.**

6. Faites de votre mieux! (20 %)

C'est maintenant à votre tour de trouver le meilleur modèle possible pour prédire les buts espérés ! Pour cette section, vous êtes libre d'essayer toutes sortes de modèles que vous souhaitez. Voici quelques choses potentielles que vous pouvez essayer (nous vous encourageons à essayer vos propres idées !) :

- Différents types de modèles tels que les réseaux de neurones, les arbres de décision, le clustering, etc.
- Des stratégies de sélection de caractéristiques plus avancées
- Une division plus réfléchie des données d'entraînement
 - Par exemple , motivé par le fait que les équipes commencent souvent en parfaite santé mais perdent ensuite des joueurs sur blessure au fil de la saison, vous

pouvez omettre les 20 premiers matchs par équipe ou sélectionner les sets d'entraînement/validation en alternant les matchs.

- Ajustement des hyperparamètres, stratégies de validation croisée
- Régularisation
- Explorez de nouvelles métriques pour cette tâche

Ne passez pas trop de temps à essayer de rendre cette partie parfaite. Le but ici est d'essayer une grande variété de choses plutôt que d'essayer d'hyper-optimiser une seule approche. Pour obtenir la note maximale, nous nous attendons à voir au moins 3 à 4 tentatives différentes.

Questions

1. **Dans votre article de blog, discutez des différentes techniques et méthodes que vous avez essayées. Incluez les quatre mêmes figures que dans la [Partie 3](#)** (courbe ROC/AUC, taux de buts par rapport au centile de probabilité, proportion cumulée de buts par rapport au centile de probabilité et courbe de fiabilité). Les mesures quantitatives ne sont nécessaires que pour quelques ensembles d'expériences, vous n'avez donc besoin d'inclure que quelques courbes sur chaque figure (par exemple, des choses que vous avez trouvées intéressantes ou des modèles qui ont particulièrement bien fonctionné). **Assurez-vous d'inclure et de mettre en évidence ce que vous considérez comme votre meilleur modèle «final».** Pour les méthodes qui n'ont pas été très efficaces ou intéressantes, vous pouvez simplement les inclure sous forme de brève discussion qualitative.
2. À côté des figures, **incluez des liens vers l'entrée d'expérience dans vos projets comet.ml pour lesquels vous avez inclus des métriques quantitatives** (environ 3-4). **Connectez les modèles aux expériences sur comet.ml (exemple [ici](#))** et inscrivez-les avec des tags informatives.

7. Évaluer sur l'ensemble de test (10 %)

Enfin, lorsque vous avez terminé de créer et d'enregistrer les modèles des sections 3, 5 et 6, évaluez ces modèles sur l'ensemble de test que vous avez mis de côté dans la partie 1. Étant donné que vous avez enregistré vos modèles finaux dans le registre de modèles comet.ml, vous pouvez désormais extraire tous ces modèles et les évaluer par programmation; cela peut être fait en utilisant l' [API Python](#).

Remarque : une fois que vous avez évalué vos modèles sur l'ensemble de test, vous devriez avoir *terminé*. Vous ne devriez jamais revenir en arrière et affiner vos modèles pour maximiser vos scores sur l'ensemble de test. Vous ne perdez pas des points si vos modèles fonctionnent moins bien sur l'ensemble de test!

Questions

Pour chacune des questions suivantes, vous devrez évaluer un total de cinq modèles :

- Les trois modèles de régression logistique que vous avez enregistrés dans la [Partie 3](#)
- Le meilleur modèle XGBoost que vous avez enregistré dans la [Partie 5](#)
- Votre meilleur modèle global que vous avez enregistré dans la [Partie 6](#)

et produisez les mêmes quatre figures que dans la partie 3 (graphique unique pour chacun, avec 5 courbes):

- ROC/AUC
- Taux de but par rapport au centile de probabilité (*Goal rate vs probability percentile*)
- Proportion cumulée de buts par rapport au centile de probabilité
- Courbe de fiabilité (*Reliability curve*)

Notez que vous êtes plus que bienvenus d'évaluer plus que les cinq spécifiés ci-dessus si vous le souhaitez! Nous vous demandons seulement d'évaluer les 5 modèles spécifiés afin d'obtenir la note maximale. Pour les deux questions ci-dessous, nous nous attendons à 3 à 5 phrases chacune, mais vous pouvez discuter plus si vous avez des sujets intéressants à discuter.

1. Testez vos 5 modèles sur l'ensemble de données intact de la saison régulière 2019/20. **Dans votre article de blog, incluez les quatre figures décrites ci-dessus. Discutez de vos résultats et de vos observations sur l'ensemble de données test. Vos modèles fonctionnent-ils aussi bien sur l'ensemble de test que sur votre ensemble de validation lors de la construction de vos modèles? Certains modèles fonctionnent-ils mieux ou moins bien que prévu?**
2. Testez vos 5 modèles sur les matchs des séries éliminatoires/playoff 2019/20 bruts. **Dans votre article de blog, incluez les quatre figures décrites ci-dessus. Discutez de vos résultats et de vos observations sur cet ensemble de tests. Y a-t-il des différences par rapport à l'ensemble de tests de la saison régulière ou obtenez-vous des performances de « généralisation » similaires?**

Évaluations de groupe

En plus de la notation décrite ci-dessus, pour chaque étape, il vous sera demandé de noter dans quelle mesure vous pensez que chacun a contribué à cette étape et de fournir des commentaires constructifs à vos coéquipiers, en fonction de leurs forces et de leurs domaines d'amélioration potentiels. Les étudiants qui ne contribuent pas beaucoup pourraient obtenir une mauvaise note et, dans des cas extrêmes, pourraient être retirés du groupe et invités à réaliser les projets individuellement.

Pour une équipe de taille **N**, chaque membre de l'équipe aura **N x 20** points à répartir entre tous les membres de votre groupe (vous compris). Vous attribuerez ensuite à chacun une note comprise entre 10 et 40, où 10 correspond à "demi-effort" et 40 correspond à "double effort". Dans une situation idéale, tout le monde contribuera au projet de manière égale et ainsi chacun attribuera 20 points à chaque coéquipier. Cependant, dans le cas où certaines personnes ont moins contribué que d'autres, vous pouvez leur attribuer moins de points et donner ces points à ceux qui, selon vous, ont plus contribué. Les cas extrêmes entraîneront un suivi par un instructeur auprès de l'équipe pour résoudre toute difficulté potentielle. Cela peut inclure une vérification de l'historique git. Toute tentative de déjouer le système sera gérée manuellement et défavorablement!

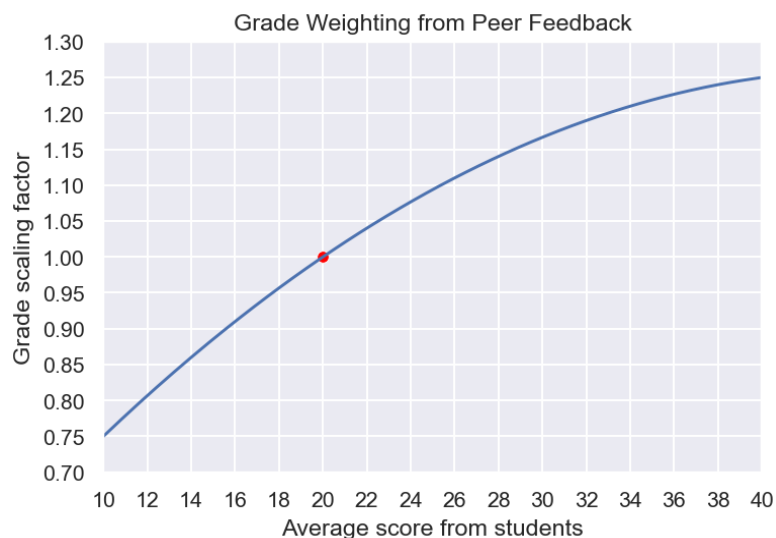
En plus du score, vous donnerez également de **breifs commentaires à vos pairs**, à la fois sur les points forts et sur les domaines d'amélioration potentiels. Cette rétroaction sera donnée de manière privée et anonyme à chaque étudiant. Vous pouvez donner votre avis sur plusieurs axes différents, tels que leur **travail d'équipe** (communication, fiabilité) et leur **travail technique** (leur contribution et la qualité de leur travail/code). Si vous donnez une note inférieure à 20, vous devez donner des commentaires constructifs sur les domaines qu'ils peuvent améliorer.

Comment les scores impactent votre note

- x est la note moyenne d'un élève (hors sa propre note), qui se trouve entre 10 ("demi-effort") et 40 ("double effort").
- Le score d'un étudiant pour le milestone sera multiplié par le facteur de pondération suivant:

$$\text{Scaling}(x) = 0.41667 + 0.0375x - 0.00041667x^4$$

- Ce système a été adopté par [Brian Fraser \(SFU\)](#), qui à son tour est basé sur les travaux de [Bill Gardner](#).

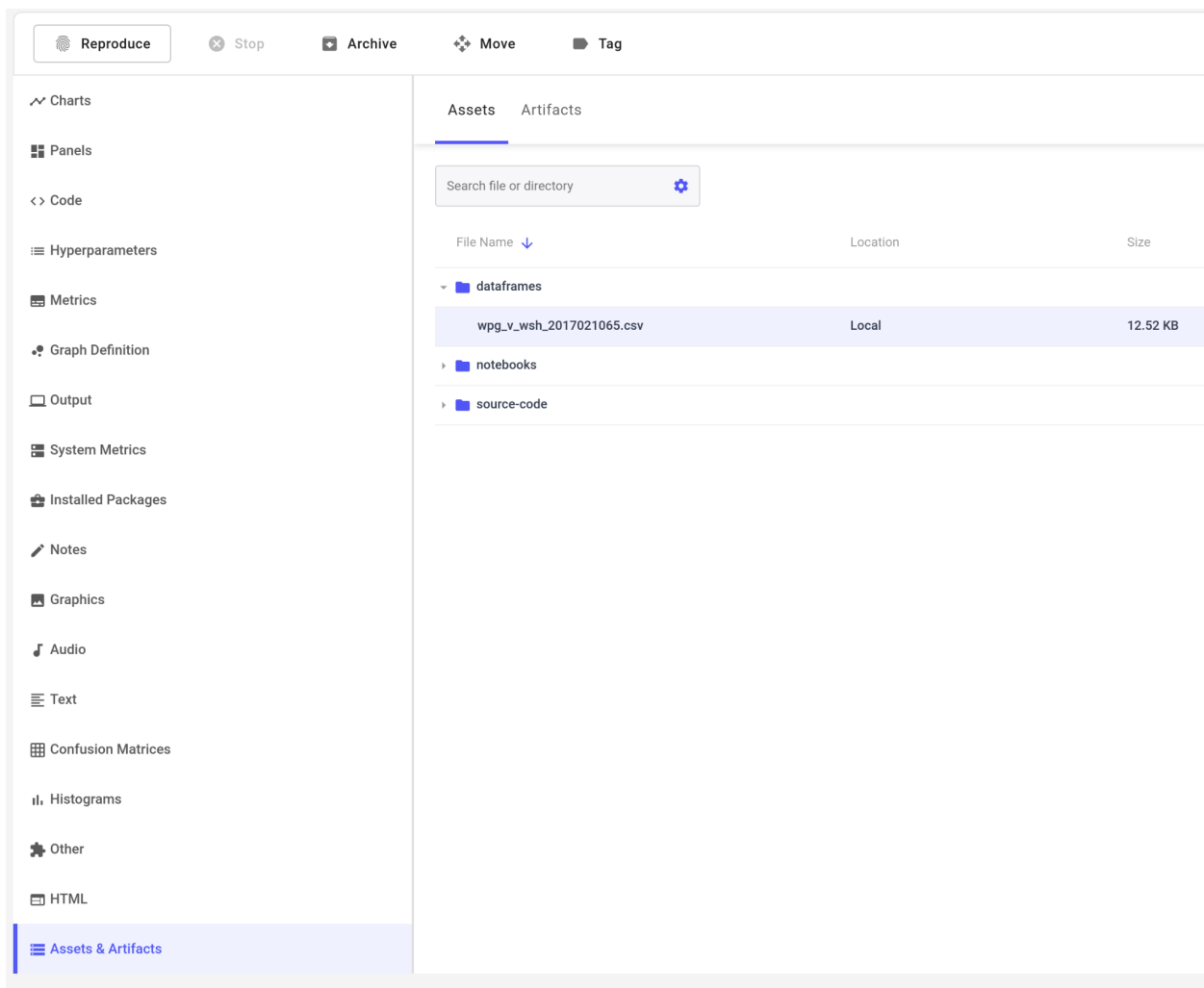


⁴ The coefficients are obtained by fitting the quadratic $y(x) = a \cdot x^2 + bx + c$ to the points $x=10 \rightarrow \text{weight}=0.75$; $x=20 \rightarrow \text{weight}=1.0$; $x=40 \rightarrow \text{weight}=1.25$.

Références utiles

- [IFT 6758 Hockey Primer](#)
- [Documentation non officielle de l'API NHL](#)
- [Cookiecutter Data Science](#) (Un outil utile pour vous aider à modéliser vos dépôts git)
- [Pandas](#)
 - [Cut](#) (utile pour le regroupement ou *binning*)
 - [Pivot](#)
 - [Groupby \(split-apply-combine\)](#)
- [scikit-learn](#)
 - [User Guide](#) (beaucoup d'informations sur les différents modèles)
 - [ROC curve](#) (courbe ROC)
 - [Area Under Curve](#) (aire sous la courbe)
 - [Calibration Curves](#) (courbes de calibration)
 - [Feature Selection](#) (sélection des caractéristiques)
- [XGBoost](#)
- [Seaborn](#)
 - [JointPlot](#)
- [SHAP](#) (Un outil utile pour l'explicabilité du modèle)
- [Comet.ml Python SDK](#)

Annexe



Un exemple de ce que nous rechercherons dans l'expérience que vous liez pour la [Partie 4](#). Assurez-vous qu'il **n'existe qu'un seul DataFrame** avec le nom spécifié dans l'expérience que vous liez et assurez-vous qu'il est de **format CSV**.