IFT 6758 Projet: Étape 1

Sortie: Sept 29, 2022
Date échéance: Oct 21, 2022

L'objectif de cette étape est de vous donner une expérience avec le data wrangling et l'analyse exploratoire des données phases d'un projet de science de données, qui sont souvent là où vous passerez la plupart de votre temps au cours d'un projet de science de données. Vous acquerrez de l'expérience avec certains des outils courants utilisés pour récupérer et manipuler des données, ainsi que pour gagner en confiance dans la création d'outils et de visualisations pour vous aider à comprendre les données avant de vous lancer dans une modélisation plus avancée.

De manière générale, le schéma de cette étape est d'utiliser l'API de statistiques de la LNH pour récupérer à la fois des données agrégées (statistiques des joueurs pour une saison donnée) et des données « play-by-play » pour une période de temps spécifique et pour générer des graphiques. Vous commencerez par créer des visualisations simples à partir des données agrégées qui ne nécessitent pas beaucoup de prétraitement, puis passerez à la création de visualisations interactives à partir des données play-by-play, ce qui impliquera plus de travail de préparation. Il y aura un petit nombre de questions qualitatives simples auxquelles répondre tout au long des tâches qui seront liées aux tâches décrites. Enfin, vous présenterez votre travail sous la forme d'une simple page web statique, créée à l' aide de Jekyll.

Notez que le travail que vous faites dans cette étape sera utile pour les futures étapes, alors assurez-vous que votre code est propre et réutilisable - votre futur moi vous en remerciera!

Une note sur le plagiat

L'utilisation de code/modèles à partir de ressources en ligne est acceptable et courante en 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 StackOverflow ou d'une documentation de package 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 pouvez 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.

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.

Données de la LNH

Le sujet de ce projet est les données de hockey, en particulier l'API des statistiques de la LNH. Ces données sont très riches ; il contient des informations sur de nombreuses années dans le passé, allant des métadonnées sur la saison elle-même (par exemple, le nombre de matchs joués), aux classements de la saison, aux statistiques des joueurs par saison, aux données d'événement précises pour chaque match joué, jeu connu **. play-by-play data**. Si vous n'êtes pas familier avec les données play-by-play, la LNH utilise ces données exactes pour générer ses visualisations play-by-play, dont un exemple est présenté ci-dessous. Pour un seul match, environ 200 à 300 événements sont suivis, généralement limités aux mises au jeu, aux tirs, aux buts, aux arrêts et aux coups sûrs (pas de passes ni de localisation de joueur individuel). Notez qu'il existe une manière logique d'attribuer un identifiant unique aux matchs, qui est décrit ici (attention à noter la différence entre les matchs de saison régulière et les matchs éliminatoires!).

Figure 1 : Exemple de données play-by-play pour le jeu **2017020001** *au début de la 1ère période. Chaque événement contient un identifiant (ex. « FACEOFF », « SHOT », etc.), une description de l'événement, les joueurs impliqués, ainsi que l'emplacement de cet événement sur la glace (dessiné sur la patinoire à l'extrême droite). Les données d'événement brutes contiennent plus d'informations que cela. Vous pouvez explorer le play-by-play de jeu ici.*

L'heure de l'événement, le type d'événement, l'emplacement, les joueurs impliqués et d'autres informations sont enregistrés pour chaque événement et les données brutes sont accessibles via l'API play-by-play. Par exemple, les données brutes pour le play-by-play ci-dessus peuvent être trouvées ici :

https://statsapi.web.nhl.com/api/v1/game/**2017020001**/feed/live/

Un extrait des données brutes de l'événement peut être vu dans la figure 2. Vous devrez explorer les données et lire les documents de l'API pour déterminer exactement ce dont vous aurez besoin.

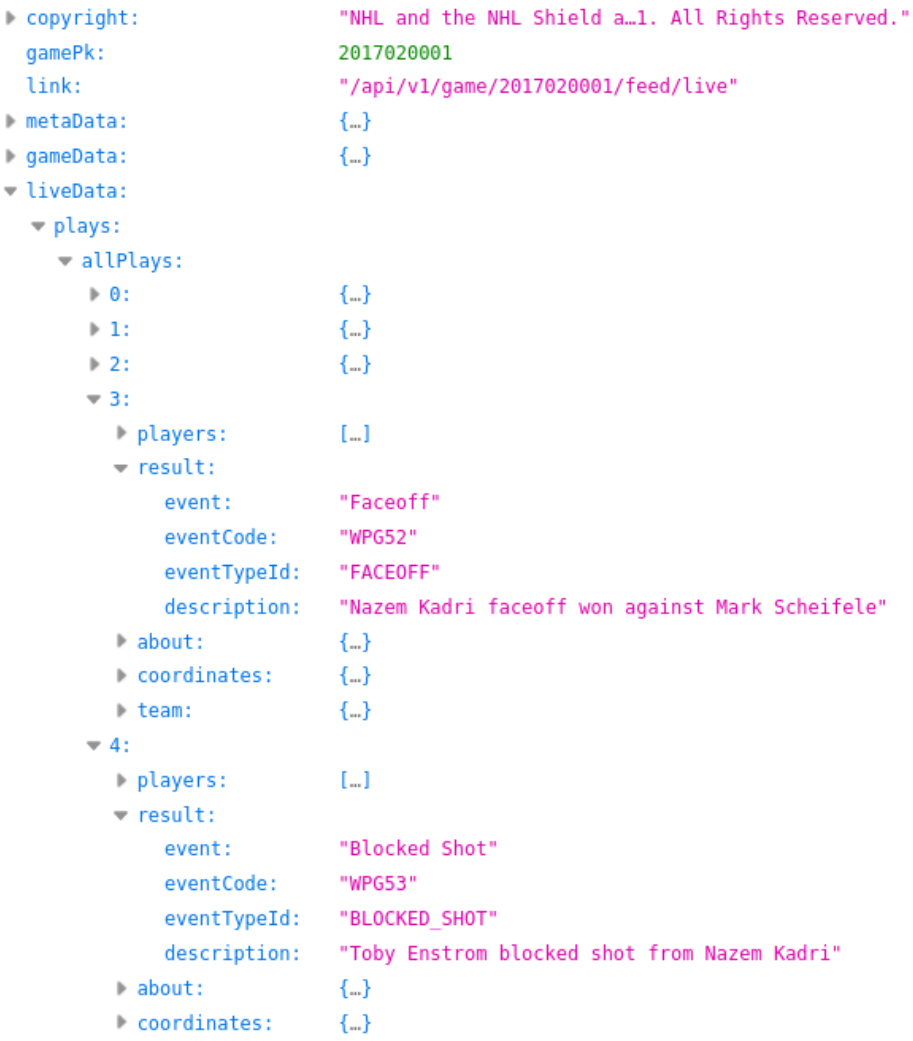


Figure 2 : Données JSON brutes obtenues à partir de l'API de statistiques de la LNH pour les mêmes événements de la figure 1. Notez qu'il existe d'autres événements entre ceux souhaités - que vous pourrez explorer !

Bien que techniquement non documenté, il existe un très détaillé [document API non officiel] maintenu par la communauté, qui devrait être le premier endroit où vous recherchez des informations sur l'API.

Motivation

Bien que nous comprenions que certaines personnes peuvent ne pas être des fans de sport, nous pensons qu'il s'agit d'un ensemble de données passionnant et travailler pour plusieurs raisons :

1. lequel équipes elles-mêmes, et d'autres gèrent leurs propres activités d'analyse.
2. Pendant la saison de hockey, les données sont mises à jour en direct au fur et à mesure que les matchs sont en cours ! Cela donne la possibilité d'interagir fréquemment avec de

nouvelles données, ce qui vous donne un aperçu des raisons pour lesquelles le « pipeline » et l'écriture de code propre et réutilisable sont essentiels à la réussite d'un workflow de science des données.
3. Il est très riche, comme discuté ci-dessus.
4. C'est "propre" dans le sens où l'API est cohérente et vous n'aurez pas à faire face à l'analyse ou au nettoyage de données absurdes.
5. Il est « désordonné » dans le sens où toutes les données brutes sont au format JSON et ne conviennent pas immédiatement à une utilisation dans un flux de travail de science des données. Vous devrez « ranger » les données dans un format utilisable, ce qui constitue une partie importante de nombreux projets de science des données. Étant donné que les données sont déjà fournies dans un format cohérent, nous pensons qu'il s'agit d'un bon équilibre entre vous donner du travail à faire pour nettoyer les données, sans être déraisonnable.
6. Le hockey est souvent un excellent facilitateur de conversation ici au Canada (et particulièrement à Montréal). Si vous êtes nouveau au Canada, c'est une excellente façon d'en apprendre un peu plus sur notre culture :)

Même si vous n'êtes pas un fan de hockey, nous espérons que vous trouverez cette expérience de projet intéressante et éducative. Nous pensons que jouer avec des données du monde réel est plus gratifiant et bien plus représentatif du flux de travail de la science des données que de travailler avec des ensembles de données préparés tels que ceux disponibles sur Kaggle. Si vous êtes particulièrement fier de votre projet, certains des livrables vous apprendront à héberger votre contenu de manière accessible au public (via les pages Github), ce qui pourra vous aider dans vos futures recherches de stage ou d'emploi !

Objectifs d'apprentissage

- **Acquisition et nettoyage de données**
  - Comprendre ce qu'est une API REST
  - Télécharger par programmation des données depuis Internet à l'aide de Python
  - Formater les données brutes en tableaux de données utiles
  - Familiarisez-vous avec l'idée de « pipeliner » votre travail ; c'est-à-dire créer des composants logiquement séparés tels que :
    - Télécharger et enregistrer des données
    - Charger des données brutes
    - Traiter des données brutes dans un certain format
- **Exploration des données**
  - Explorer les données brutes et comprendre à quoi elles ressemblent
  - Construire des outils interactifs simples pour vous aider à travailler avec les données plus efficacement
- **Visualisation et science de données exploratoire**
- acquérir certaine intuition et répondre aux quelques questions simples sur les données en regardant les visualisations
  - utiliser Matplotlib et Seaborn pour créer des belles figures
  - créer chiffres interactifs pour communiquer vos résultats plus efficacement

LIVRABLES

Vous devez soumettre:

1. Blog rapport post style
2. codebase de votre équipe **qui est reproductible**

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 chiffres (interactifs !). Nous vous fournirons un modèle et des instructions par Sep. 22, 2021 , alors ne vous inquiétez pas d'avoir à tout découvrir par vous-même. À un niveau élevé, vous utiliserez Jekyll pour créer une page Web statique à partir de Markdown. Il s'agit d'un moyen très simple de créer de belles pages et pourrait vous être très utile pour créer des articles de blog à l'avenir si vous êtes intéressé ou si vous souhaitez améliorer votre CV dans une recherche d'emploi. Bien que nous ne déploierons pas ces pages au public[1], 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!

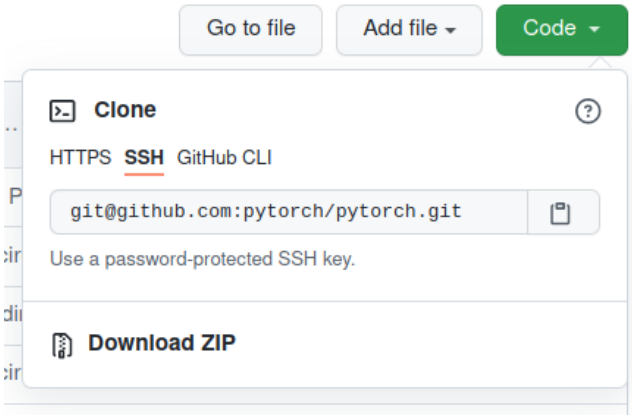
Détails de Présentation

Pour soumettre votre projet, vous devez:

- ☐ Publier votre soumission étape finale au `master` ou `main` branche (Vous devez le faire avant de télécharger les ZIPs!)
- ☐ Proposez un fichier ZIP de votre blog à gradescope
- ☐ Envoyer un fichier ZIP de votre base du code à gradescope
- ☐ Ajoutez le compte IFT6758 TA Github (`@ift-6758-a22`) à votre repo en tant que *viewer*

Pour soumettre un ZIP de votre référentiel, vous pouvez le télécharger via l'interface utilisateur Github du référentiel

[1] Une mise en garde concernant les pages Github est que même si votre dépôt est privé, toutes les pages publiées sont publiques. Vous ne pourrez peut-être même pas publier une page à partir d'un dépôt privé si vous n'avez pas obtenu votre compte étudiant gratuit Github Pro. Parce que nous ne voulons pas que les groupes aient leurs pages visibles les uns aux autres pendant qu'ils travaillent sur le projet, vous ne devez pas publier les pages que vous créez et les rendre localement.



N'oubliez pas que cette méthode ne télécharge pas l'intégralité git, mais uniquement la branche principale ou principale. Assurez-vous que tout votre code est validé dans la branche master avant de télécharger les ZIP.

Tâches et questions

Les tâches requises pour l'étape 1 sont décrites ici. La description globale de ce qui est requis est décrite au début de chaque tâche. La section **Questions** de chaque tâche décrira le contenu requis dans le billet de blog. Cela peut être soit des questions d'interprétation, soit vous devrez peut-être produire des chiffres 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 pour les questions, dans la plupart des cas, quelques phrases suffiront.

0. Article de blog

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

Vous DEVEZ le soumettre dans un format d'article de blog !

Nous vous suggérons de commencer à créer un environnement de travail pour l'article de blog dès le début, car il sera beaucoup plus facile de répondre aux questions et d'ajouter des graphiques pendant que vous travaillez sur le projet, plutôt que d'essayer de le configurer juste avant la date limite! Une fois que votre environnement est opérationnel, il est très simple de travailler avec!

1. Acquisition de données (25 %)

Créez une fonction ou une classe pour télécharger les données play-by-play de la LNH pour la saison régulière et les séries éliminatoires. Le point final principal d'intérêt est :

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

Vous devrez lire le document API non officiel pour comprendre comment **`GAME_ID`** est formé. Vous pouvez ouvrir le point de terminaison dans votre navigateur pour consulter le JSON brut pour l'explorer un peu (Firefox a une belle visionneuse JSON intégrée).

Utilisez votre outil pour télécharger les données de la saison 2016-17 jusqu'à la saison 2020-21. Vous pouvez l'implémenter comme vous le souhaitez, mais si vous avez besoin de conseils, voici quelques conseils :

1. Il s'agit d'une API publique, et en tant que telle, vous devez être conscient que quelqu'un d'autre paie pour les demandes. Vous devez télécharger les données brutes et les enregistrer localement, puis utiliser cette copie locale pour en dériver des ensembles de données ordonnés/utilisables.

2. **Ne pas** valider les données (ou les gros blobs binaires) dans votre référentiel GitHub. C'est une mauvaise pratique, git est pour le code, pas pour le stockage de fichiers. Bien que cela soit possible pour l'ensemble de données avec lequel vous travaillerez, ce ne sera probablement pas le cas lorsque vous travaillerez sur des projets à plus grande échelle dans l'industrie ou le milieu universitaire. Les grands dépôts git deviennent lents à cloner et à utiliser. Notez qu'en raison de la façon dont git fonctionne, une fois que vous avez validé et poussé un fichier, le simple fait de le supprimer et de valider la suppression ne supprimera pas réellement le fichier ; vous devrez en fait réécrire l'historique de git, ce qui devient risqué. Un bon moyen d'éviter accidentellement des fichiers est d'utiliser un valider des.gitignore fichier et d'ajouter le modèle de fichier que vous souhaitez (comme `*.npy` ou `*.pkl`).

3. Un bon modèle pourrait être de définir une fonction qui accepte l'année cible et un chemin de fichier comme argument, puis recherche dans le chemin de fichier spécifié un fichier correspondant à l'ensemble de données que vous allez télécharger. S'il existe, il pourrait immédiatement ouvrir le fichier et renvoyer le contenu enregistré. Sinon, il pourrait télécharger le contenu de l'API REST et l'enregistrer dans le fichier avant de renvoyer les données. Cela signifie que la première fois que vous exécutez cette fonction, elle téléchargera et mettra en cache automatiquement les données localement, et la prochaine fois que vous exécuterez la même fonction, elle chargera à la place les données locales. Pensez à utiliser des variables d'environnement pour permettre à chaque coéquipier de spécifier des emplacements différents, et à ce que votre fonction récupère automatiquement l'emplacement spécifié par la variable d'environnement afin que vous n'ayez pas à vous battre pour les chemins dans votre référentiel git.

4. Si vous vouliez devenir encore plus sophistiqué, vous pourriez envisager de pousser cette logique dans une classe qui implémente la logique suggérée dans le point précédent. Cela se prête bien à la façon dont les données sont séparées par les saisons de hockey, et cela vous permettrait d'ajouter une logique qui se généraliserait à toute autre saison que vous souhaiterez peut-être analyser de manière propre et évolutive. Pour être *encore plus sophistiqué*, vous pouvez envisager de surcharger l'« add » (`__add__` opérateur) sur cette classe pour vous permettre d'ajouter les données entre les saisons à une structure de données commune, ce qui vous permet d'agréger les données entre les saisons. Ce n'est absolument *pas* obligatoire, ce ne sont que quelques idées pour vous inspirer ! Nous vous encourageons à faire preuve de créativité et à appliquer vos anciennes connaissances sur les structures de données/POO à la science des données - cela peut vous rendre la vie beaucoup plus facile !

5. Écrire des docstrings pour vos fonctions est une bonne habitude à prendre.

Questions

1. **Rédigez un bref tutoriel** sur la façon dont votre équipe a téléchargé l'ensemble de données. Imaginez :) que vous cherchiez un guide sur la façon de télécharger les données play-by-play ; votre guide devrait vous faire dire "Parfait - c'est exactement ce que je cherchais!". Inclure votre fonction/classe et un exemple de comment l'utiliser.

2. Outil de débogage interactif (bonus 10%)

Lorsque vous travaillez avec de nouvelles données, il est souvent utile de créer des outils interactifs simples pour vous aider à parcourir les données et les implémentations de prototypes. Un outil utile est ipywidgets, qui vous permet de créer très rapidement et facilement des widgets HTML dans une cellule de bloc-notes Jupyter. Un cas d'utilisation courant de ces widgets consiste à les appliquer en tant que décorateurs et à les utiliser pour spécifier des arguments de fonction. Par exemple, si vous souhaitez récupérer des informations qui résident dans un élément d'un tableau, vous pouvez utiliser un `IntSlider` pour contrôler l'index qui est passé à cette fonction. Vous pouvez alors définir une logique dans cette fonction pour afficher votre image ; si votre liste est une liste de chemins d'images, vous pouvez charger l'image et l'afficher via matplotlib. Ces widgets peuvent également être imbriqués, vous permettant un degré élevé de flexibilité avec très peu d'effort.

Questions

1. **Implémentez un ipywidget** qui vous permet de parcourir tous les événements, pour chaque match d'une saison donnée, avec la possibilité de changer entre la saison régulière et les séries éliminatoires. **Dessinez les coordonnées de l'événement sur l'image de la patinoire fournie**, similaire à l'exemple ci-dessous (vous pouvez simplement imprimer les données de l'événement lorsqu'il n'y a pas de coordonnées).

Vous pouvez également imprimer toutes les informations que vous jugez utiles, telles que les métadonnées du jeu/boxscores et les résumés d'événements (mais ce n'est pas obligatoire). **Prenez une capture d'écran de l'outil et ajoutez-la à l'article de blog**, accompagnée du **code de l'outil** et d'une **brève description (1-2 phrases)** de ce que fait votre outil. Vous n'avez pas à vous soucier de l'intégration de l'outil dans le blogpost.

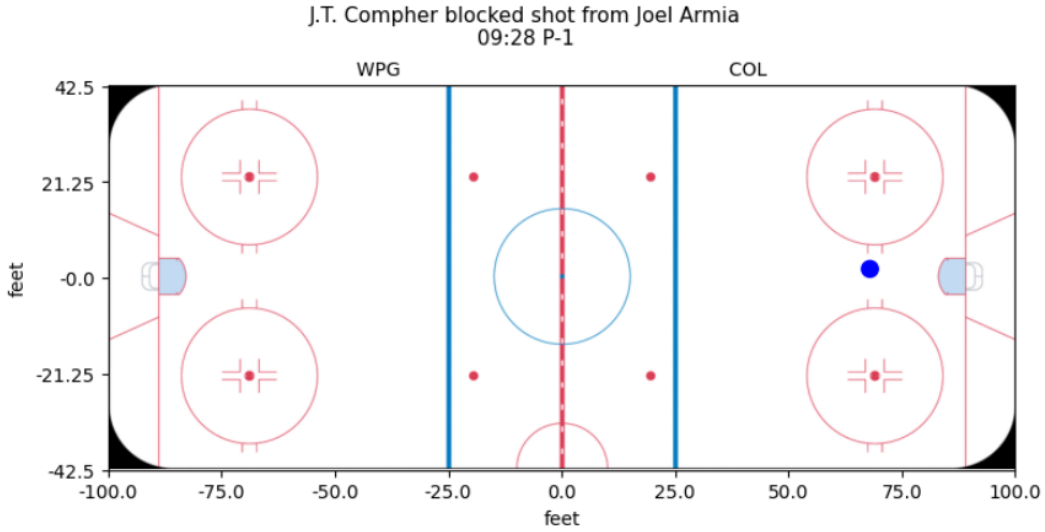
Remarque : *Un bon test de cohérence consiste à vérifier un jeu spécifique avec les données disponibles sur le site Web de la LNH, dont un exemple peut être trouvé ici. Vous remarquerez que les coordonnées de l'événement sont également dessinées, ce qui vous permet de confirmer si vos chiffres sont valides. Pour vous inspirer, une capture d'écran de celle que j'ai rapidement créée se trouve ci-dessous. Au-delà de la figure, vous n'avez pas besoin de copier cette mise en page ; n'hésitez pas à ajouter toutes les informations que vous jugez utiles !*

game_id 379

```
2017-11-30T02:30:00Z
Game ID: 379; COL (home) vs WPG (away)

       OT
                     Home    Away
       Teams:        COL     WPG
       Goals:        3       2
       SoG:          26      33
       SO Goals:     None    None
       SO Attempts:  None    None
```

i 46




```
{'about': {'dateTime': '2017-11-30T02:52:22Z',
           'eventId': 77,
           'eventIdx': 46,
           'goals': {'away': 1, 'home': 0},
           'ordinalNum': '1st',
           'period': 1,
           'periodTime': '09:28',
           'periodTimeRemaining': '10:32',
           'periodType': 'REGULAR'},
 'coordinates': {'x': 68.0, 'y': 2.0},
 'players': [{'player': {'fullName': 'J.T. Compher',
                         'id': 8477456,
                         'link': '/api/v1/people/8477456'},
              'playerType': 'Blocker'},
             {'player': {'fullName': 'Joel Armia',
                         'id': 8476469,
                         'link': '/api/v1/people/8476469'},
              'playerType': 'Shooter'}],
 'result': {'description': 'J.T. Compher blocked shot from Joel Armia',
            'event': 'Blocked Shot',
            'eventCode': 'COL77',
            'eventTypeId': 'BLOCKED_SHOT'},
 'team': {'id': 21,
          'link': '/api/v1/teams/21',
          'name': 'Colorado Avalanche',
          'triCode': 'COL'}}
```


Un exemple de widget interactif que vous pouvez créer pour explorer les données. Cela a été créé à l'aide de simples ipywidgets et de matplotlib stock.

4. Rangez les données (10 %)

Maintenant que vous avez obtenu et exploré un peu les données, nous devons formater les données de manière à faciliter la science des données (c'est-à-dire ranger les données) ! Nous souhaitons généralement travailler avec de belles trames de données Pandas plutôt que des données brutes, vous êtes donc ici chargé de traiter les données d'événement brutes de chaque jeu en trames de données qui seront utilisables pour les tâches suivantes. Vous pouvez trouver ce point de terminaison utile :

```
https://statsapi.web.nhl.com/api/v1/playTypes
```

Créez une fonction pour convertir tous les événements de chaque jeu en une trame de données pandas. Pour ce jalon, vous voudrez inclure des événements de type « coups » et « objectifs ». Vous pouvez ignorer les tirs manqués ou les tirs bloqués pour le moment. Pour chaque événement, vous voudrez inclure comme fonctionnalités (au minimum) : les informations sur l'heure/la période de jeu, l'identifiant du jeu, les informations sur l'équipe (quelle équipe a tiré), l'indicateur s'il s'agit d'un tir ou d'un but, les coordonnées sur la glace, le nom du tireur et du gardien de but (ne vous inquiétez pas des aides pour l'instant), le type de tir, si c'était sur un filet vide, et si un but était à force égale, en désavantage numérique ou en avantage numérique.

Des questions

1. Dans votre article de blog, **incluez un petit extrait** de votre cadre de données final (par exemple, en utilisant `head(10)`). Vous pouvez simplement inclure une capture d'écran plutôt que de vous battre pour que les tableaux soient soigneusement formatés en HTML/markdown.

2. Vous remarquerez que le champ de « force » (c.-à-d. égal, avantage numérique, en désavantage numérique) n'existe que pour les buts, pas pour les tirs. De plus, il n'inclut pas la force réelle des joueurs sur la glace (c'est-à-dire 5 contre 4, ou 5 contre 3, etc.). **Discutez de la** façon dont vous pourriez ajouter les informations sur la force réelle (c'est-à-dire 5 contre 4, etc.) aux *tirs et aux buts*, compte tenu des autres types d'événements (au-delà des tirs et des buts) et des fonctionnalités disponibles. Vous n'avez pas besoin de l'implémenter pour ce jalon.

3. En quelques phrases, **discutez d'au moins 3** fonctionnalités supplémentaires que vous pourriez envisager de créer à partir des données disponibles dans cet ensemble de données. Nous ne cherchons pas de réponses particulières, mais si vous avez besoin d'inspiration, un tir ou un but pourrait-il être classé comme un rebond/tir dans le rush (expliquez comment déterminer entre ceux-ci?

5. Visualisations simples (25 %)

Utilisons maintenant les données rangées pour créer des graphiques simples sur les données agrégées. Dans chacune des questions ci-dessous, vous devez faire un choix approprié de

graphique pour montrer la relation en question. Il existe généralement plusieurs façons correctes de le faire - si votre graphique raconte la bonne histoire, vous obtiendrez tous les points.

Questions

1. **Produisez un graphique comparant** les types de tirs sur toutes les équipes dans une saison de votre choix (i.e. agrégez juste sur tous les tirs). Superposez le nombre de buts sur le nombre de tirs. Quel semble être le type de tir le plus dangereux ? Le type de tir le plus courant? Pourquoi est-ce que vous avez choisi ce type de graphique. Ajoutez cette figure et cette discussion à votre article de blog.

2. Quelle est la relation entre la distance à laquelle un tir a été effectué et la chance qu'il s'agisse d'un but ? **Produisez un graphique** pour chaque saison entre 2018-19 et 2020-21 pour répondre à cette question, et ajoutez-le à votre article de blog avec quelques phrases décrivant votre silhouette. Y a-t-il eu beaucoup de changements au cours des trois dernières saisons? Pourquoi est-ce que vous avez choisi ce type de graphique?

3. Combinez les informations des sections précédentes pour **produire un graphique** qui montre le pourcentage de buts (# buts / # tirs) en fonction à la fois de la distance par rapport au filet et de la catégorie de types de tirs (vous pouvez choisir une seule saison de votre choix) brièvement **Discutez de** vos conclusions ; Par exemple, quels pourraient être les types de tirs les plus dangereux ?

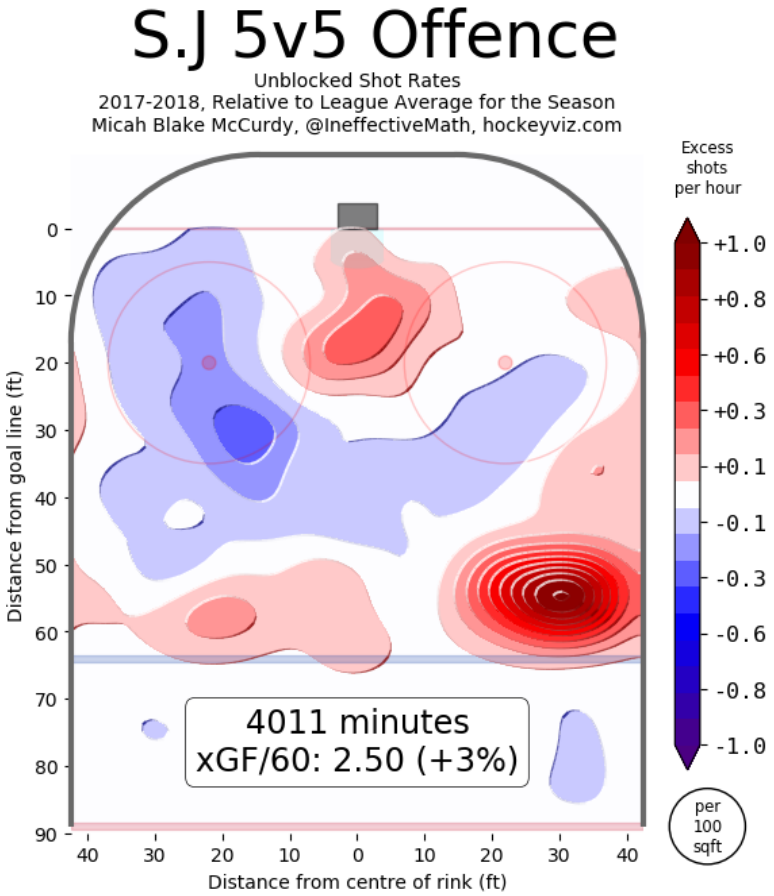
6. Visualisations avancées:(30 %)

L'ensemble final de visualisations que vous créerez sont des plans de tir pour une équipe de la LNH donnée, pour une année et une saison données. Un excellent exemple de ces graphiques, avec une description détaillée de la façon de les lire, se trouve sur le site Web de hockeyviz (qui est une excellente ressource pour de nombreuses choses sur la science des données du hockey). Notez que vous devrez créer ces figures à partir de zéro ; pour ce jalon, vous ne pouvez pas utiliser de bibliothèque qui génère des chiffres spécifiques au domaine (hockey) pour vous. Vous recevrez un exemple d'image de patinoire qui a le bon rapport.

Pour créer ces figures, vous devez :

1. vous assurer que vous pouvez travailler correctement avec les coordonnées de l'événement. Cela inclut de s'assurer que les tirs sont du bon côté de la patinoire (en raison de changements de période ou de commencer de différents côtés pendant un match), ainsi que de pouvoir mapper des coordonnées physiques aux coordonnées en pixels sur la figure.
2. Calculez les statistiques agrégées des emplacements de tir dans l'ensemble de la ligue pour calculer le *taux de tir moyen par heure de la ligue.* Vous pouvez faire quelques hypothèses simplificatrices :

   a. Vous pouvez supposer que tous les coups sont de force égale ; cela signifie que vous pouvez simplement agréger tous les coups plutôt que d'avoir à déterminer si un coup était un coup de force égale ou non (rappelez-vous Q 4.2).
   b. Vous pouvez supposer que chaque jeu dure 60 minutes.
3. Regroupez les tirs par équipe et utilisez les *moyennes de ligue du taux de tir moyen par heure* calculées ci-dessus pour calculer le *taux de tirs excédentaires par heure*. Vous pouvez choisir de représenter cela soit comme une différence brute de buts entre les équipes, soit comme un pourcentage.
4. Faites des choix appropriés pour regrouper vos données lors de leur affichage. Vous pouvez également envisager d'utiliser des techniques de lissage pour rendre vos plans de prise de vue plus lisibles. Une stratégie courante consiste à utiliser l'estimation de densité de noyau avec un noyau gaussien.
5. Rendez le graphique interactif, avec des options pour sélectionner l' **équipe**. La façon la plus simple de le faire est d'utiliser quelque chose comme plotly ou bokeh. Une belle démo simple de ce que vous pouvez faire avec plotly peut être trouvée ici.
6. Produisez un graphique interactif pour chaque saison de 2016-17 à 2020-2021. (compris). Il vous suffit de créer les figures de la zone offensive ; vous n'avez rien à faire pour la zone défensive. Comme mentionné ci-dessus, vous pouvez également ignorer le fait d'essayer de déterminer si un événement s'est produit lors d'un avantage numérique ou d'un désavantage numérique.



Source : www.hockeyviz.com. Exemple de carte de tir offensif pour les Sharks de San Jose, sur la période 2017-2018 Vous n'avez pas besoin de calculer les xGF pour/60, ou le nombre de minutes dans la zone offensive.

Questions

1. Exportez les 4 tracés de zone offensive au format HTML et **intégrez-les dans votre article de blog**. Votre parcelle doit permettre aux utilisateurs de sélectionner n'importe quelle équipe au cours de la saison sélectionnée.
   ***Note**: Parce que vous pouvez trouver ces chiffres sur internet, répondre à ces questions sans produire ces chiffres* ne *vous rapportera pas de points !*

2. **Discutez** (en quelques phrases) de ce que vous pouvez interpréter à partir de ces graphiques.

3. Considérez l'Avalanche du Colorado; jetez un œil à leur carte de tir au cours de la saison 2016-17. **Discutez de** ce que vous pourriez dire sur l'équipe au cours de cette saison. Regardez maintenant la carte des plans de l'Avalanche du Colorado pour la saison 2020-21 et **discutez de** ce que vous pourriez conclure de ces différences. Est-ce que ça a du sens? *Astuce : regardez le classement*.

4. Considérez les Sabres de Buffalo, une équipe qui a connu des difficultés ces dernières années, et comparez-les au Lightning de Tampa Bay, une équipe qui a remporté le Stanley deux années consécutives. Regardez les plans de tir de ces deux équipes des saisons 2018-19, 2019-20 et 2020-21. **Discutez des** observations que vous pouvez faire. Y a-t-il quelque chose qui pourrait expliquer le succès du Lightning, ou les luttes des Sabres ? À quel point une image est-elle complète selon vous ?

***Remarque**: le but de cet exercice est de vous familiariser avec l'utilisation des libraries Python standard pour créer des visualisations. Vous ne pouvez pas utiliser d'outil qui crée pour vous des visualisations spécifiques à un domaine (par exemple, le hockey). Vous êtes libre de vous fier aux bibliothèques de stock (matplotlib, seaborn, plotly, bokeh, etc.) pour générer ces tracés.*

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 **brefs 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.

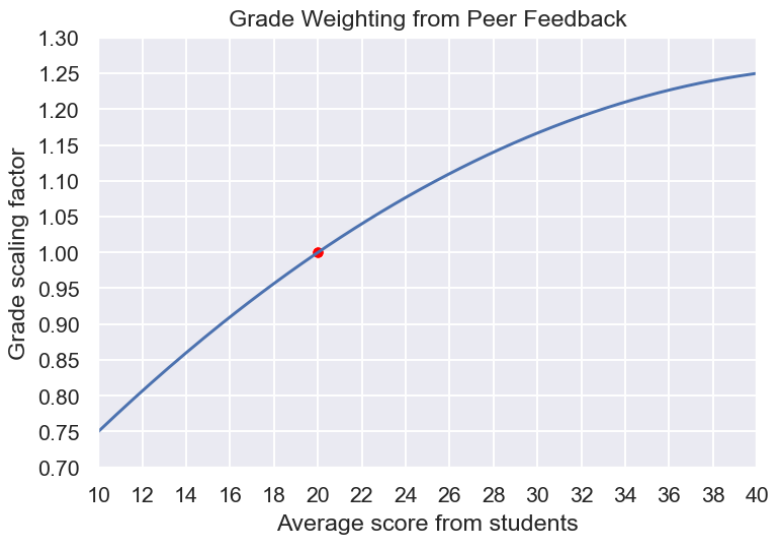
Comment les scores impactent votre note

- **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:

$$\texttt{Scaling}(\mathbf{x}) = 0.41667 + 0.0375\mathbf{x} - 0.00041667\mathbf{x}^2$$

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

[2] *The coefficients are obtained by fitting the quadratic $y(x) = a*\mathbf{x}^2 + b\mathbf{x} + c$ to the points $\mathbf{x}=10 \rightarrow weight=0.75$; $\mathbf{x}=20 \rightarrow weight=1.0$; $\mathbf{x}=40 \rightarrow weight=1.25$.*



Votre note finale sera alors mise à l'échelle par ce facteur final. À titre d'exemple, nous montrons un cas non idéal où la charge de travail n'était pas équitablement répartie dans le groupe. Prenez un groupe de 4 personnes (**A**, B, C et D) où le score final du projet était de 95 % et chacun a attribué **20 points à la personne A**. Leur note ne serait pas affectée (rappelez-vous que vous excluez votre propre score) :

$$\texttt{Scaling}((20 + 20 + 20) / 3) = 1 * 95\% = 95\%$$

Cependant pour ce même groupe, si la personne B n'a pas contribué autant, sa note finale peut en souffrir :

$$\texttt{Scaling}((15 + 14 + 15) / 3) = 0.88 * 95\% = 83.3\%$$

Peut-être que les personnes C et D ont pris le relais, et leurs scores le reflètent - leurs notes finales peuvent être augmentées en conséquence. Cet exemple est résumé dans le tableau ci-dessous.

En général, nous ne nous attendons pas à ce que les équipes aient des problèmes. Nous espérons qu'en établissant une méthode claire pour s'évaluer mutuellement et comment cela peut affecter directement votre note, les gens sont incités à coopérer et à contribuer de manière égale au projet. En cas de conflit ou de préoccupation avec le groupe, nous vous encourageons à essayer de le résoudre le plus rapidement possible. Si vous avez besoin de l'aide des instructeurs pour résoudre des problèmes ou des préoccupations, veuillez nous contacter le plus tôt possible pour les résoudre.

Tableau 1: Exemple d'évaluations de groupe pour une équipe de 95% pour ce milestone.

Personne qui a donné le score	Personne à qui le score est assigné			
	A	**B**	**C**	**D**
A	20	15	21	24
B	20	15	22	23
C	20	14	22	24
D	20	15	21	24
Facteur	1.0	0.88	1.03	1.07
Score Final	95%	83.3%	97.6%	101.7%

Références utiles

- IFT6758 Hockey Primer
- Documentation non officielle de l'API NHL
- Cookiecutter Data Science modéliser vos dépôts (Un outil utile pour vous aider à git)