École Nationale de la Statistique et de l'Administration Économique
Mastère Spécialisé Data Science

PROJET : Prévisions dans le Monde des Sports de Combat et Comparaison aux Prévisions de Marché

Kévin Martel

Année universitaire 2016-2017

Introduction :

Le Mixed Martial Art est un sport de contact qui rassemble toutes les disciplines du pied-poing et autorise les phases de combat au sol.
Depuis le début de la décennie il est considéré comme le sport progressant le plus rapidement.
Malgré qu'il soit encore interdit en France, le business qu'il représente Outre-Manche et Outre-Atlantique est gigantesque.
Chaque année de nouveaux records de vente sont atteints et l'année dernière l'UFC, principale organisation de MMA, a vendu près de 9 millions de pay-per-view et encaissé plus de 90 millions d'euros soit 3 fois plus qu'en 2014.
Cela reste toujours 10 fois moins important que les droits TV du football mais la hausse reste impressionnante.

Quant au monde des paris sportifs, il est lui aussi en pleine ébullition : les Français ont misé 45% de plus en 2016 selon une étude du Monde.
Le MMA n'a pas de raison de faire exception à cette folie du pari sportif d'autant plus qu'il est un sport à la mode.

Cependant étant un jeune sport, les modèles de cotes sportives semblent connaître des limites.
De nombreuses surprises adviennent et il serait intéressant de savoir si nous pouvons construire un modèle permettant de prévoir le résultat des matchs et concurrencer les bookmarkers.

Le but de cette étude consistera donc à un construire un modèle prédictif afin de connaître le résultat d'un affrontement entre deux combattants puis de confronter ses résultats au monde des paris sportifs.
Je vous présenterai dans un premier temps la méthodologie que j'ai employé pour construire la base, ensuite je discuterai du modèle et de ses conclusions (qualité de prévision et gains monétaires possibles).
Enfin j'évoquerai les améliorations que je souhaite effectuer lors d'une prochaine étude.

Section 1 : Récupération des données et définition du contexte :

FightMetrics est un provider de données spécialisé dans le MMA.
Pour chaque événement de l'UFC ils visionnent en temps réel les combats et renseignent des informations comme le nombre de coups portés, absorbés, défendus, les plaquages tentés et subis ou encore les soumissions avortées.
Malheureusement ses données ne sont accessibles que par ses clients et quelques doctorants souhaitant étudier le MMA.
N'ayant pas eu de retour de leur part, j'ai décidé de scrapper moi-même les données intéressantes sur le site sherdog.com.
Pour cela je me suis appuyé sur un code existant, qui ne fonctionnait pas, que j'ai débuggé et amélioré.
Le scrapping est disponible à partir du fichier *sherdogspider.py* en lançant la commande *scrapy runspider sherdogspider.py* depuis le terminal.
Ce site ne renseigne pas des statistiques aussi intéressantes que celle de FightMetrics, elles restent très basiques et factuelles.

De plus, afin d'optimiser le temps de scrapping un choix de périmètre et de features a du être effectué. Je me suis donc restreint aux évènements de l'UFC.
J'ai construit deux fichiers un comprenant les caractéristiques des combats et l'autres comprenant les caractéristiques des combattants.

Voici en résumé les features disponibles :

- pour les combats : fichier *fights* :

Combattant A	Combattant B	Resultat	Type resultat	Round de fin	Minute de fin	Date du combat

- pour les combattants : fichier *fighter_file* :

Combattant	Nationalité	Catégorie	Poids	Taille	Date de naissance

Section 2 : Retraitement des données : *mma_analyse_donnees.ipynb*

Une fois les fichiers récupérés avec les données, il a fallu les mettre en forme afin d'avoir pour chaque ligne : le combattant, ses caractéristiques, les caractéristiques de son adversaire et le résultat (victoire ou défaite).
J'en ai donc profité pour créer de nouveaux features représentant le palmarès du combattant au moment du combat.
Puisque j'avais à disposition la liste de tous ces combats à l'UFC, j'ai analysé tous les résultats de ce combattant avant la date du combat et je les ai agrégé.
Il m'a aussi semblé important de ne pas seulement exprimer indépendamment les caractéristiques de A et celles de B mais de les exprimer comparativement.
J'ai donc créé des variables *diff_* qui représentent la comparaison entre les caractéristiques de A et celles de B.
Pour calculer ces variables j'ai appliqué trois fonctions que je détaillerai dans la section 3.

Voici à quoi ressemble le fichier final :

COMBATTANT	henry
CATEGORIE	Bantamweight
NATIONALITE A	South Korea
NATIONALITE B	Japan
POIDS A	135,00
TAILLE A	175,26
ANNEE A	1989,00
EXPERIENCE A	5,00
VICTOIRE A	0,00
DEFAITE A	5,00
KO A	0,00
SOUMISSION A	0,00
STREAK A	5,00
POIDS B	143,00
TAILLE B	170,18
ANNEE B	1976,00

EXPERIENCE B	10,00
VICTOIRE B	3,00
DEFAITE B	7,00
KO B	1,00
SOUMISSION B	2,00
STREAK B	3,00
DIFF POIDS	0,94
DIFF TAILLE	1,03
DIFF AGE	1,01
DIFF VICTOIRE	0,00
DIFF DEFAITE	0,71
DIFF KO	0,00
DIFF SOUMISSION	0,00
DIFF STREAK	1,67
RESULTAT	0,00
ADVERSAIRE	eddie

Avec en gris les features utilisés, et en rouge l'output que l'on cherche a prédire (0 pour une défaite, 1 pour une victoire).

Section 3 : Le modèle XgBoost et les fonctions de mapping :
XGBoost_.ipynb

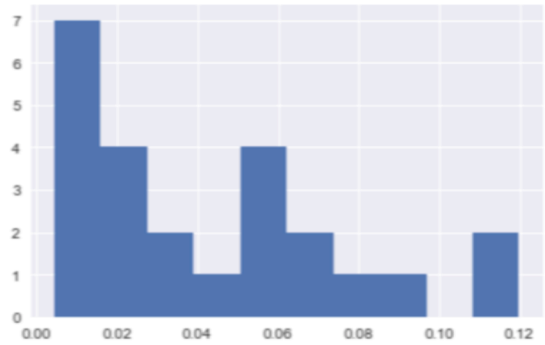
XGBoost est un algorithme de boosting. C'est donc un modèle qui consiste à agréger, séquentiellement, différents modèles simples (weak learner) où chaque nouveau modèle ajouté corrige l'erreur engendrée par les anciens modèles. À la différence d'un modèle de boosting classique, XGBoost utilise la technique de descente du gradient pour minimiser l'erreur. Ce modèle est très utile puisqu'il permet de résoudre à la fois des problèmes de régression ou de classification et très efficace puisqu'il surperforme les modèles classiques de boosting.

L'avantage du modèle XGBoost, c'est qu'il indique les features qui contribuent le plus au modèle. Nous pouvons alors créer une règle de décision très rapide afin de sélectionner les caractéristiques que nous utiliserons dans le modèle. Ainsi, nous avons retenu la caractéristique si celle-ci a une importance de plus de 0,5%.

```
In [161]: # Selection des bons features
          plt.hist(clf.feature_importances_[clf.feature_importances_>0])
          important_indices = np.where(clf.feature_importances_>0.005)[0]
          print(important_indices)
          important_indices.shape

          [ 0  2  3  4  5  6  8  9 10 11 13 14 16 17 18 19 20 21 22 23 25]

Out[161]: (21,)
```




```
In [162]: elements_retenu=np.array(columns)
          elements_retenu[important_indices][:]

Out[162]: array(['POIDS A', 'ANNEE A', 'EXPERIENCE A', 'VICTOIRE A', 'DEFAITE A',
                 'KO A', 'STREAK A', 'POIDS B', 'TAILLE B', 'ANNEE B', 'VICTOIRE B',
                 'DEFAITE B', 'SOUMISSION B', 'STREAK B', 'DIFF POIDS',
                 'DIFF TAILLE', 'DIFF AGE', 'DIFF VICTOIRE', 'DIFF DEFAITE',
```

Nous avons donc créé trois modèles XGBoost, avec pour chacun une fonction de mapping différente.
Pour chaque modèle nous avons scindé notre échantillon en deux : 2/3 pour apprendre et 1/3 pour tester.
Dans cette section nous étudierons les features par modèle puis nous discuterons de la stabilité et de la validité de nos choix dans la section suivante.

Concernant les fonctions de mapping, fonctions permettant de créer les variables *diff_* pour représenter la comparaison entre deux combattants, nous avons choisi les fonctions suivantes :

- $f_1(a,b) = a - b$
- $f_2(a,b) = \frac{a}{b}$
- $f_3(a,b) = \frac{a-b}{a+b}$

Voici les features sélectionnés par modèle :

Features	f_1	f_2	f_3
ANNEE A			X
ANNEE B			X
DEFAITE A	X	X	X
DEFAITE B	X	X	X
DIFF AGE	X	X	X
DIFF DEFAITE	X	X	X
DIFF KO	X		X
DIFF POIDS	X	X	X
DIFF SOUMISSION	X		
DIFF STREAK	X		X
DIFF TAILLE	X		X
DIFF VICTOIRE	X	X	X
EXPERIENCE A	X	X	X
EXPERIENCE B	X		
KO A			X
POIDS A	X		X
POIDS B	X	X	X
SOUMISSION B			X
STREAK A	X	X	X
STREAK B	X	X	X
TAILLE A	X		
TAILLE B	X		X
VICTOIRE A	X	X	X
VICTOIRE B	X	X	X

On remarque que la fonction 2 exclut un grand nombre des variables *diff_*, or par construction ce sont les variables qui nous intéressent.
Cela s'explique par notre décision de fixer à 0 les variables dont le dénominateur est égal à 0, cela induit un biais, et ce cas se présente fréquemment sur notre échantillon.
Quelque soit la performance de prévision de ce modèle nous décidons donc de l'exclure pour la suite de cette étude.

Les fonctions 1 et 3 sont quant à elles assez cohérentes, elles contiennent pratiquement toutes les variables *diff_* et lorsqu'une variable est significative pour un combattant, elle l'est aussi pour l'autre combattant dans la plupart des cas.
Etudions la validité de ces modèles.

Section 4 : Validité des modèles

Pour valider notre modèle nous avons appliqué la méthode de validation croisée par 3-ensembles. Ce test consiste à diviser notre échantillon initial en 3 sous-échantillons (la proportion des classes dans les trois sous-échantillons étant homogène grâce à la fonction *stratifiedk*), puis à choisir un des 3 sous-échantillons comme ensemble de validation et les 2 autres comme ensemble

d'apprentissage. On calcule ensuite une erreur de prévision. Puis on répète l'opération afin que chaque sous-échantillon ait été utilisé une fois comme ensemble de validation. La moyenne des 3 erreurs est enfin calculée pour estimer l'erreur de prédiction.

```
In [9]: #Cross validation
clf = XGBClassifier(max_depth=5, base_score=0.005)
cv = StratifiedKFold(y, n_folds=3)
preds = np.ones(y.shape[0])
for i, (train, test) in enumerate(cv):
    preds[test] = clf.fit(X[train], y[train]).predict_proba(X[test])[:,1]
    print("fold {}, ROC AUC: {:.5f}".format(i, roc_auc_score(y[test], preds[test])))
print(roc_auc_score(y, preds))
```

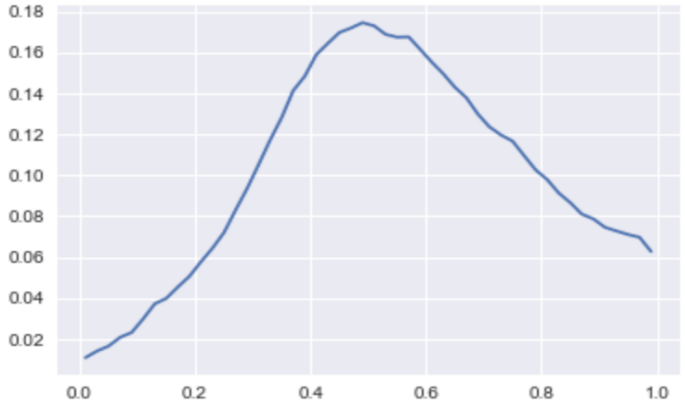
ROC AUC	f_1	f_3
fold 0	0,589	0,645
fold 1	0,627	0,653
fold 2	0,594	0,634
Moyenne	0,602	0,644

On constate que les deux modèles sont stables puisque leur ROC est proche pour les 3 sous échantillons. Et surtout, on peut conclure que notre modèle apprend puisque le ROC sur l'échantillon total est compris entre 60% et 64%.

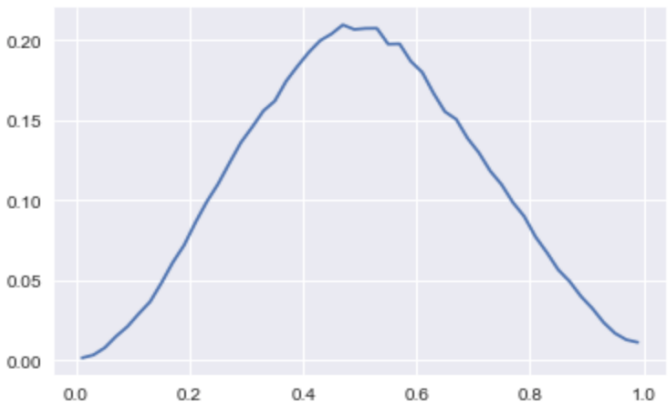
On détermine ensuite le seuil optimal qui nous permettra de réaliser des prédictions correctes. Pour cela, on applique la métrique *matthews correlation coefficient* puisque c'est une métrique couramment utilisée dans les compétitions kaggle pour évaluer la performance d'un modèle.

```
# on récupéere le meilleur seuil pour avoir le meilleur prédicteur selon le critère du coefficient de Mattews
thresholds = np.linspace(0.01, 0.99, 50)
mcc = np.array([matthews_corrcoef(y, preds>thr) for thr in thresholds])
plt.plot(thresholds, mcc)
best_threshold = thresholds[mcc.argmax()]
print(mcc.max())
```

MCC en fonction du seuil pour le modèle 1



MCC en fonction du seuil pour le modèle 3



Cette analyse confirme notre pressenti puisque le modèle 3 affiche un coefficient maximum de 0,21 contre 0,17 pour le modèle 1.
Nous choisirons donc le modèle 3 pour effectuer nos prévisions.

Section 5 : Exemple de prévisions et comparaison au marché des paris :

Grâce au seuil défini par le coefficient de corrélation Matthews on peut effectuer les prévisions du résultat des combats sur notre échantillon de test.
On compare ensuite ces prévisions aux vrais résultats. On obtient un ROC de 61% ce qui reste très satisfaisant dans le milieu sportif.

L'intérêt de cette étude consiste à comparer nos résultats, non pas à la réalité mais aux anticipations du marché afin de faire des gains sur les paris sportifs.
Pour effectuer cette comparaison nous avons extrait 10 combats dont les cotes sont disponibles.
Grâce aux cotes on en déduit les pronostics des bookmarkers : on considère le favori comme celui qui a la cote d'ouverture la plus basse.
On met en place un modèle de marché qui consiste à parier sur le favori des bookmakers.
Enfin on compare les gains dégagés par notre modèle et par le modèle de marché.

Voici le tableau récapitulatif : (tableau complet en annexes)

Combattant A	Combattant B	Prévision Modèle*	Prevision Marché**	Vainqueur	Gain Modèle***	Gain Marché****
Viscardi Andrade	Richard Walsh	Richard Walsh	Richard Walsh	Viscardi Andrade	-100	-100
Mike Bronzoulis	Justin Reiswerg	Mike Bronzoulis	Mike Bronzoulis	Mike Bronzoulis	40	140
Alan Belcher	Michael Bisping	Michael Bisping	Alan Belcher	Michael Bisping	60	-100
Carlos Condit	Martin Kampmann	Martin Kampmann	Martin Kampmann	Carlos Condit	-100	-100
Vitor Belfort	Jon Jones	Jon Jones	Jon Jones	Jon Jones	525	625
Antonio Rogerio Nogueira	Ryan Bader	Ryan Bader	Antonio Rogerio Nogueira	Ryan Bader	250	-100
Urijah Faber	Ivan Menjivar	Ivan Menjivar	Ivan Menjivar	Urijah Faber	-100	-100
John Dodson	Manny Gamburyan	Manny Gamburyan	Manny Gamburyan	John Dodson	-100	-100
Conor McGregor	Nate Diaz	Conor McGregor	Nate Diaz	Conor McGregor	70	-100
Tim Kennedy	Rafael Natal	Rafael Natal	Rafael Natal	Tim Kennedy	-100	-100
Rashad Evans	Ryan Bader	Rashad Evans	Ryan Bader	Ryan Bader	-100	175
				Somme	345	140

* Prévision du modèle : prévision estimée grâce au modèle XgBoost et la fonction de mapping
** Prévision du marché : on considère le gagnant comme celui qui a la cote la plus basse sur le marché des paris
*** Gain Modèle : on gagne le bénéfice du pari si le pronostic du modèle est bon ou on perd la mise s'il est mauvais
**** Gain du marché : on gagne le bénéfice du pari si le pronostic du marché est bon, on perd la mise sinon

En jaune, nos prévisions qui diffèrent des prévisions du marché. On remarque que dans 3 cas sur 4 nos différences sont justifiées.
Le gain dégagé par notre modèle est supérieur à celui dégagé en suivant le marché.

Conclusion :

Même si statistiquement notre modèle n'est pas optimal, il semble meilleur que celui du marché des paris sportifs. Il existe donc des opportunités de gains sur le marché.
Afin de s'en assurer, il aurait été intéressant de scrapper les cotes d'ouverture pour tous les combats prédits afin de calculer le gain global de notre modèle comparativement aux gains obtenus en suivant les favoris du marché.
Une autre amélioration souhaitée serait de construire un programme rassemblant les fonctions principales de notre étude afin de fournir un outil prêt à l'emploi pour un utilisateur.
Cet outil permettrait de construire la base de données (scrapping et retraitement), d'estimer le modèle (notebook *XgBoost_*) puis d'effectuer nos prévisions et les comparer aux cotes du marché.

Sources :

http://www.sherdog.com/
https://www.bestfightodds.com/
https://github.com/andosa/MMAinfer/blob/master/sherdogspider.py

Annexes :

1. Récapitulatif des étapes informatiques et codes associés :

   Etape 1 :
   Scrapping en lançant la commande `scrapy runspider sherdogspider.py` depuis le terminal.
   Générations des fichiers fights.csv et fighter_file.csv.
   Etape 2 :
   Retraitement des données des fichiers scrappés en lançant le notebook python *mma_analyse_donnees*.
   Génération des fichiers stats_fight_mma_.
   Etape 3 :
   Construction du modèle en lançant le notebook python *XGBoost_mapping* sur des fichiers retraités.
   Etape 4 :
   Comparaison des cotes manuellement.

2. Tableau de résultats complets

Combattant A	Combattant B	Probabilité de Gain A	Probabilité de Gain B	Prevision	Prevision marché ******	Vainqueur	Cote A*	Cote B*	Probabilité du marché A**	Probabilité du marché B**	Gain Potentiel***	Gain Réalisé****	Gain marché***
Viscardi Andrade	Richard Walsh	0,284	0,716	Richard Walsh	Richard Walsh	Viscardi Andrade	135	105	0,4375	0,5625	5	-100	-100
Mike Bronzoulis	Justin Reiswerg	0,594	0,406	Mike Bronzoulis	Mike Bronzoulis	Mike Bronzoulis	140	180	0,5625	0,4375	40	40	140
Alan Belcher	Michael Bisping	0,457	0,543	Michael Bisping	Alan Belcher	Michael Bisping	125	160	0,561403509	0,438596491	60	60	-100
Carlos Condit	Martin Kampmann	0,447	0,553	Martin Kampmann	Martin Kampmann	Carlos Condit	180	150	0,454545455	0,545454545	50	-100	-100
Vitor Belfort	Jon Jones	0,456	0,544	Jon Jones	Jon Jones	Jon Jones	925	625	0,403225806	0,596774194	525	525	625
Antonio Rogerio Nogueira	Ryan Bader	0,482	0,518	Ryan Bader	Antonio Rogerio Nogueira	Ryan Bader	250	350	0,583333333	0,416666667	250	250	-100
Urijah Faber	Ivan Menjivar	0,279	0,721	Ivan Menjivar	Ivan Menjivar	Urijah Faber	180	140	0,4375	0,5625	40	-100	-100
John Dodson	Manny Gamburyan	0,465	0,535	Manny Gamburyan	Manny Gamburyan	John Dodson	505	335	0,398809524	0,601190476	235	-100	-100
Conor McGregor	Nate Diaz	0,536	0,464	Conor McGregor	Nate Diaz	Conor McGregor	170	145	0,46031746	0,53968254	70	70	-100
Tim Kennedy	Rafael Natal	0,200	0,800	Rafael Natal	Rafael Natal	Tim Kennedy	310	230	0,425925926	0,574074074	130	-100	-100
Rashad Evans	Ryan Bader	0,627	0,373	Rashad Evans	Ryan Bader	Ryan Bader	245	175	0,416666667	0,583333333	145	-100	175
										Somme	1550	345	140

* Cote : open du site https://www.bestfightodds.com/
** Probabilite du marche A = cote A/(cote A + cote B)
*** Gain potentiel : bénéfice avec 100 euros d'investissement
**** Gain réalisé : on gagne le bénéfice si le pronosticdu modèle est bon ou on perd la mise s'il est mauvais
***** Gain du marché : on gagne le bénéfice si le pronostic du marché est bon, on perd la mise sinon
****** Prévision du marché : on considère le gagnant comme celui qui a la cote la plus basse sur le marché