NEW BET ASSISTANT

Where NBA stats are decrypted for you

ESIEE PARIS – 2014-2015 – Filière DRIO

LY Diane | CAO Claudia | HONG Philippe | CHENG Harrison | COURIVAUD Raphaël | SIM Chan Bora

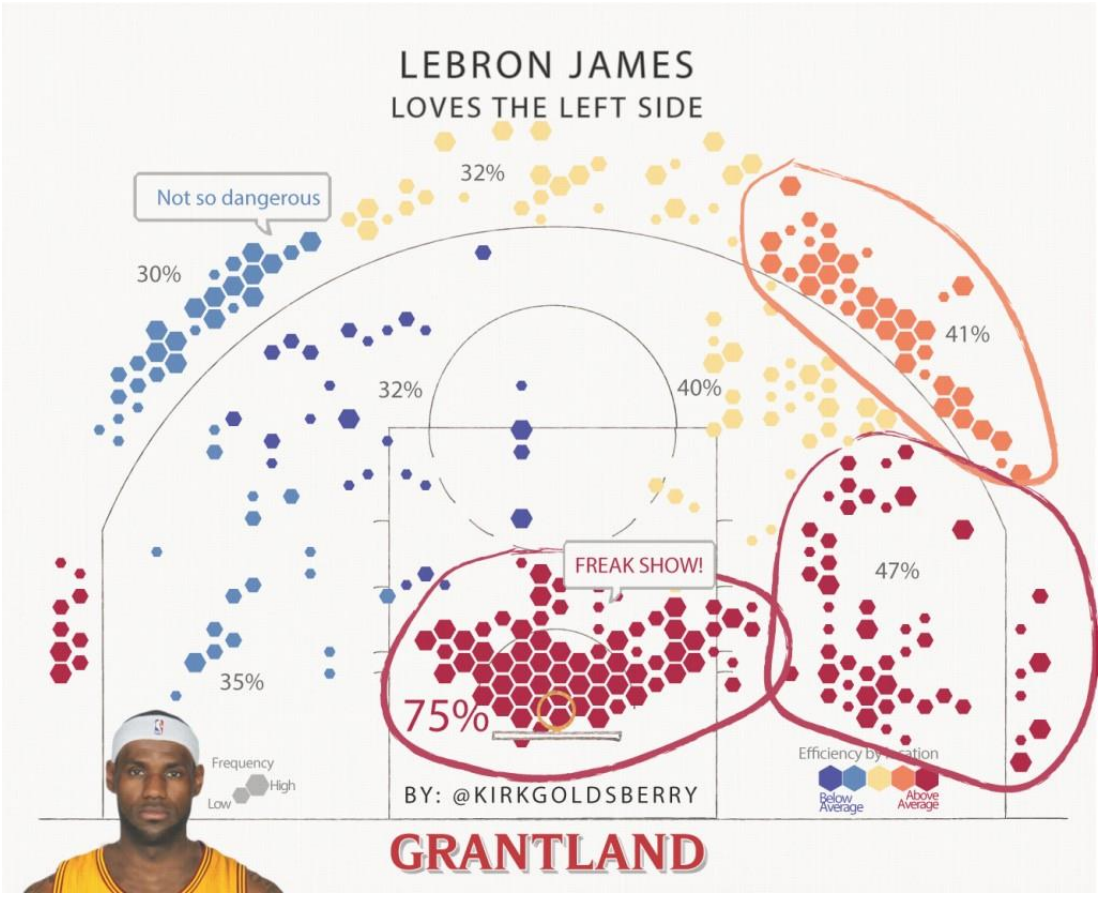
Sommaire

Présentation

Nous sommes 6 étudiants venant de la Classe Préparatoire de Jean Moulin à Torcy ayant intégré l'ESIEE PARIS à la rentrée 2014. Nous étant tous dirigés vers la filière DRIO, une chose était évidente : notre projet de fin de E3 serait axé sur le BIG DATA afin d'approfondir nos compétences sur ce domaine. Étant tous passionnés par le sport, nous avons pensé que les techniques de classifications que nous avons entrevue en cours durant le second semestre seraient très intéressantes appliquées au sport et plus précisément à la NBA (National Basketball Association).

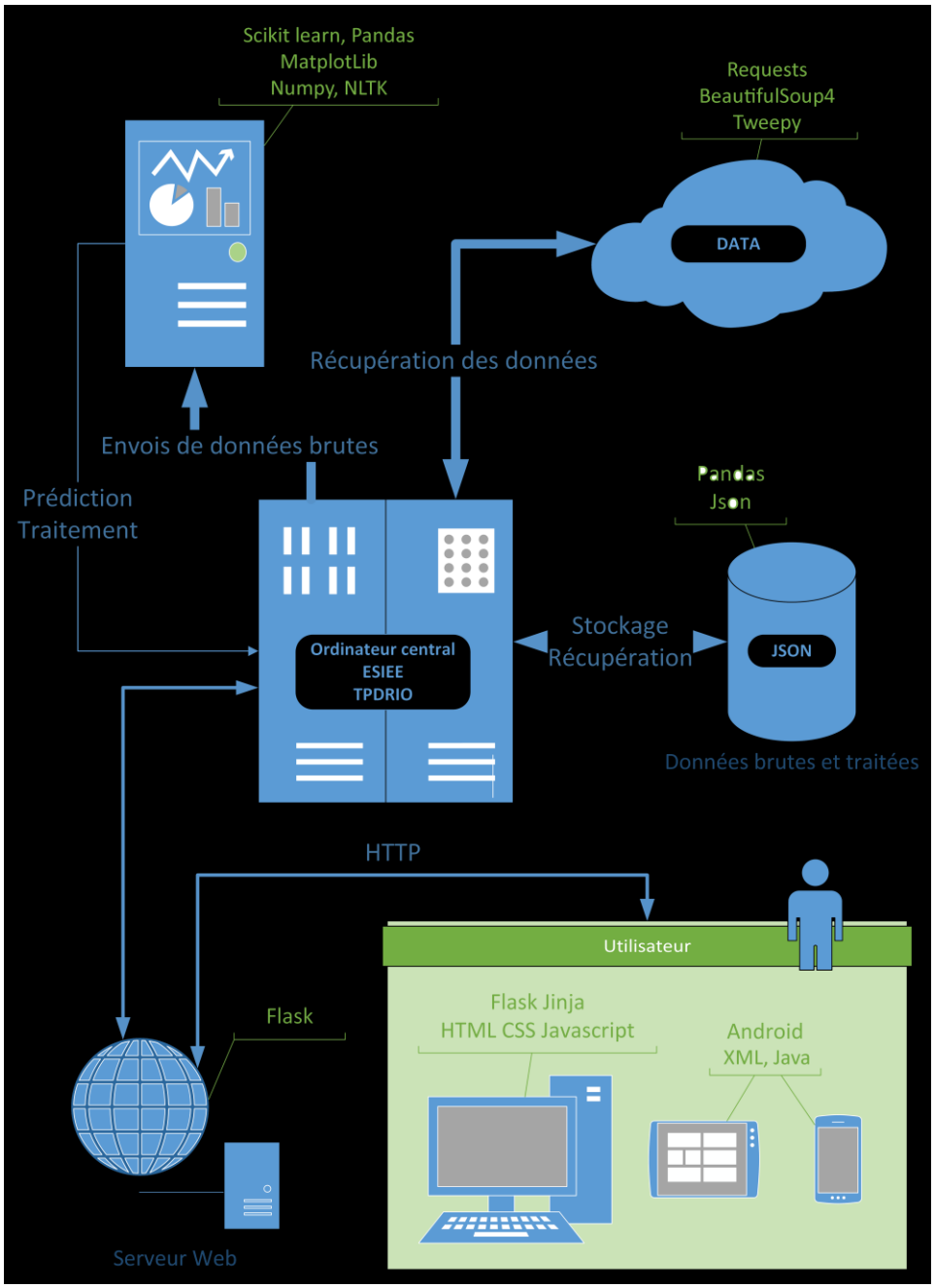
Introduction

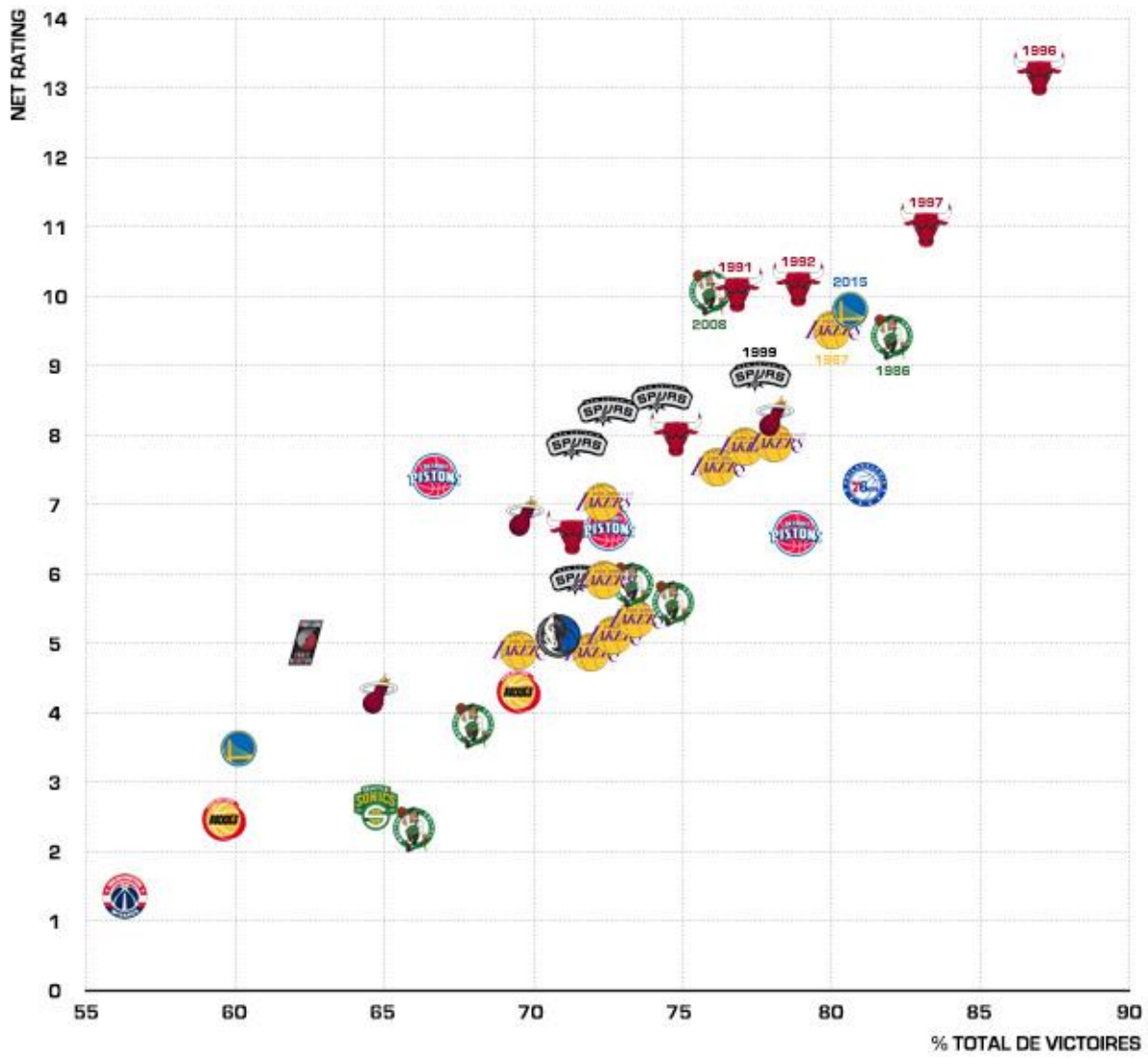
L'objectif de notre projet était de mettre en place une plate-forme complète permettant la récupération, l'analyse, et l'affichage de prédictions de matchs NBA. Nous avons choisi plus particulièrement le basket-ball comme sport car nous sommes tous plus ou moins familier avec celui-ci, ainsi nous possédons un certain savoir sur le fonctionnement et l'influence des différentes statistiques sur le jeu. De plus, c'est un sport américain et les américains sont passionnés de statistiques sportives ce qui résulte dans le fait que nous avions beaucoup de données à notre disposition.



Voulant que la partie d'analyse de nos prédictions et la partie d'affichage soit plus que correcte, nous nous sommes répartis les tâches. Chaque membre de notre groupe s'est occupé d'une ou plusieurs parties de notre projet.

Ce dernier se décompose selon ce schéma :



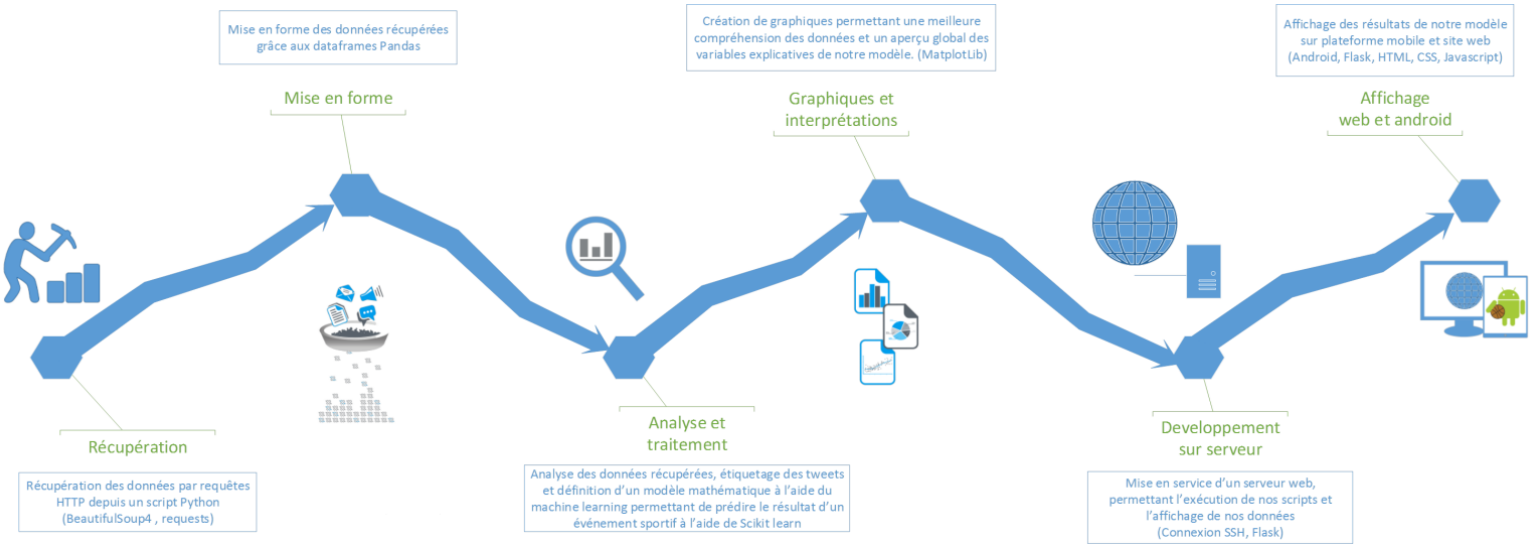


Source : http://www.basketusa.com/

Nous pouvons grâce aux données effectuer de nombreuses comparaisons et trouver des similitudes. Il faut juste savoir les décrypter et en faire bon usage.

Nous avons également voulu montrer et traverser toutes les étapes de l'exploration de données. Dans un premier temps nous devons récupérer des données brutes auprès de différentes sources. Dans notre projet les sources principales seront les sites de la NBA et d'ESPN qui regroupent un nombre

important de données concernant les matchs, les équipes et les joueurs. Dans un deuxième temps il faut les mettre en forme pour qu'elles soient plus faciles à analyser. Elles sont tellement nombreuses qu'il serait impossible de les traiter directement. Ensuite vient la partie mathématique du cheminement des données. C'est là que le Machine Learning et la création du modèle intervient. On définit nos variables explicatives et on essaie de prédire une certaine cible en fonction de ces dernières. C'est la partie la plus délicate, car il faut choisir avec grand soin ces variables pour obtenir une prédiction assez fiable. Nous avons ensuite besoin de montrer aux utilisateurs cette partie mathématique, mais nous devons la rendre plus attractive. Pour cela nous devons créer des graphiques permettant de mettre en œuvre toutes ces données et de les expliquer. Enfin nous devons créer une interface permettant d'afficher nos résultats aux différents utilisateurs. Dans le cadre de ce projet nous avons mis en place une application Android et un site web, pour faciliter l'accès à ces données au plus grand nombre.



Extraction des données et mise en forme

Tout d'abord nous avons dû nous renseigner sur le type de données que nous devions utiliser. Nous avons scruté les différentes sources auxquelles nous avions accès. Les données concernant la ligue Américaine de Basket sont multiples et très diverses. Nous nous sommes concentrés sur deux types de données, les données purement statistiques proposées par la NBA mais aussi des données plus implicites comme l'avis des gens.

Pour récupérer cette dernière donnée nous avons mis en place un traitement de tweets concernant les matchs de la soirée. Nous essayons grâce à des techniques linguistiques de déterminer le sentiment général des personnes concernant un match et plus particulièrement une équipe. Nous avons aussi mis en place un mini-jeu sur plate-forme Android demandant aux joueurs de donner leur avis sur les différents matchs de la soirée.

Extraction de données statistiques

Une majeure partie des données utilisées pour la prédiction des résultats des matchs correspond à des données statistiques récupérées sur internet. Durant chaque match, des personnes présentes sur le bord du terrain scrutent les joueurs afin de récupérer toutes les données possibles comme par exemple le nombre de points marqués, le nombre de faute commises, le différentiel de points (+/-) lorsque le joueur est sur le terrain, etc. Ces données sont affichéesx² sur différentes pages internet. Nous avons choisi de ne garder que deux des sites les plus "officiels" à savoir http://espn.go.com/nba/ et le site officiel de la NBA http://www.nba.com/.

Nous voulions récupérer le maximum de données sur chaque match. Nous avons alors récupéré toutes les statistiques de chaque joueur sur plus de 4500 matchs depuis 2009.

Pour récupérer toutes ces données nous avons utilisé deux modules Python qui permettent de récupérer les données à partir de requêtes HTTP. Le premier est requests (http://docs.python-requests.org/en/latest/) il permet de faire des requêtes HTTP très facilement et par exemple de récupérer le texte d'une page HTML. Ensuite lorsque nous avons récupéré le texte brut nous utilisons BeautifulSoup4 (http://www.crummy.com/software/BeautifulSoup/), un Parser HTML. Ce dernier utilise les spécificités du code html pour pouvoir récupérer les données en fonction des différentes balises, classes, et identifiants.

Maintenant débute un gros travail de recherche des URLs comportant les données intéressantes pour notre projet.

Nous voulons tout d'abord récupérer des données de "base" comme les statistiques moyennes de chaque équipe et de chaque joueur sur la saison. (http://espn.go.com/nba/team/stats/_/name/cle). Nous récupérons sur la page d'accueil les URLs des statistiques de chaque équipe et ensuite bouclons sur cette liste pour récupérer les tableaux pour chaque équipe. Lorsque nous avons le tableau comprenant les statistiques moyennes de chaque équipe sur la saison il faut pouvoir les mettre en forme pour pouvoir les traiter grâce un code python. Nous utilisons ici un autre module python Pandas (http://pandas.pydata.org/) qui permet de stocker des données dans des DataFrame. Ces DataFrame sont très pratiques pour manipuler les données. De nombreuses fonctions sont fournies avec ces DataFrame facilitant leur utilisation, leur traitement, leur analyse et leur affichage.

Vient ensuite la plus grosse partie de nos données. Nous avons décidé dans un premier temps de ne récupérer que les matchs de la saison 2014-2015. Mais nous n'avons récupéré qu'environ 750 matchs sur cette période, ce qui n'est pas beaucoup pour réaliser un traitement convenable par la suite. Nous avons alors décidé d'aller chercher jusqu'au début de la saison 2009-2010. Ce qui nous fait un total de 4500 matchs, ce qui est nettement plus acceptable pour notre modèle mathématique.

Nous avons alors crée un script combinant ces différents modules python permettant d'effectuer plus de 12 000 requêtes HTTP afin de récupérer toutes ces données. Ce script est assez long, il dure environ 3 heures. En effet en

parallèle nous devions récupérer les résultats des matchs et le différentiel de points marqués à chaque quart temps par les deux équipes qui s'affrontent. Nous avons fait la distinction des deux équipes en fonction de laquelle joue à l'extérieur et laquelle à domicile. Nous observons dans tout notre projet une différence de l'équipe jouant à domicile avec celle jouant à l'extérieur. Nous arrivons pour la DataFrame des statistiques pour chaque joueur pour tous les matchs depuis 2009 à un tableau comportant plus de 100 000 lignes.

Pour optimiser ces 12 000 requêtes nous créons plusieurs fichiers d'URLs afin d'éviter les scripts trop long à exécuter. Nous préférons réaliser ce travail en plusieurs étapes, ce qui facilite le débogage du programme, de plus cela empêche les erreurs qui surviennent de stopper totalement notre script et qui nous évite de recommencer tout depuis le début.

	1Q	2Q	3Q	4Q	AwayTeam	HomeTeam	OT	Total	win
date									
20150607	0	2	1	-3	CLE	GS	2	2	1
20150604	10	-7	-3	0	CLE	GS	-8	-8	-1
20150527	5	-11	0	-8	HOU	GS	0	-14	-1
20150526	-12	-5	-8	-5	ATL	CLE	0	-30	-1
20150525	-23	13	-5	2	GS	HOU	0	-13	-1
20150524	3	-2	-6	5	ATL	CLE	-3	-3	-1
20150523	12	13	6	4	GS	HOU	0	35	1
20150522	5	0	13	-6	CLE	ATL	0	12	1
20150521	-8	8	-2	1	HOU	GS	0	-1	-1
20150520	-6	6	7	1	CLE	ATL	0	8	1

	+/-	3PM-A	AST	Against	BLK	DREB	Date	FGM-A	FTM-A	MIN	OREB	PF	PTS	Poste	REB	STL	TO	Team	index
PlayerName																			
Harrison Barnes	+9	2-5	0	Cavaliers	2	5	20150611	4-9	4-4	33	3	2	14	SF	8	0	0	Warriors	0
Draymond Green	+16	1-3	6	Cavaliers	1	7	20150611	6-11	4-7	32	0	5	17	SF	7	2	0	Warriors	1
Stephen Curry	+18	4-7	7	Cavaliers	0	2	20150611	8-17	2-2	41	0	1	22	PG	2	1	4	Warriors	2
Andre Iguodala	+16	4-9	0	Cavaliers	0	7	20150611	8-15	2-2	39	1	4	22	SG	8	1	1	Warriors	3
Klay Thompson	+15	1-5	2	Cavaliers	1	2	20150611	4-9	0-0	39	0	0	9	SG	2	0	0	Warriors	4
David Lee	+4	0-0	3	Cavaliers	0	5	20150611	3-7	3-6	15	0	1	9	PF	5	0	0	Warriors	5
James Michael McAdoo	-1	0-0	0	Cavaliers	0	1	20150611	0-0	0-0	1	0	0	0	SF	1	0	0	Warriors	6
Andrew Bogut	0	0-0	1	Cavaliers	0	1	20150611	0-0	0-0	3	0	3	0	C	1	0	1	Warriors	7
Marreese Speights	0	0-0	0	Cavaliers	0	1	20150611	0-2	1-2	2	1	1	1	C	2	0	0	Warriors	8
Shaun Livingston	+25	0-0	4	Cavaliers	1	7	20150611	2-4	3-4	25	1	3	7	PG	8	1	1	Warriors	9
Justin Holiday	0	0-0	0	Cavaliers	0	0	20150611	0-0	0-0	2	0	0	0	SG	0	0	0	Warriors	10
Leandro Barbosa	+3	0-1	1	Cavaliers	0	0	20150611	1-3	0-0	7	0	1	2	SG	0	0	0	Warriors	11
Warriors		12-30	24	Cavaliers	5	38	20150611	36-77	19-27		6	21	103	Team	44	5	7	Warriors	12
LeBron James	-15	1-4	8	Warriors	0	10	20150611	7-22	5-10	41	2	5	20	SF	12	0	2	Cavaliers	13
Timofey Mozgov	-5	0-0	1	Warriors	0	4	20150611	9-16	10-12	33	6	3	28	C	10	1	3	Cavaliers	14
Tristan Thompson	-13	0-0	0	Warriors	1	7	20150611	6-10	0-0	38	6	1	12	C	13	0	0	Cavaliers	15
Iman Shumpert	-12	1-5	1	Warriors	1	6	20150611	2-9	0-0	39	1	2	5	SG	7	0	0	Cavaliers	16
Matthew Dellavedova	-12	2-9	4	Warriors	0	1	20150611	3-14	2-2	33	0	2	10	SG	1	1	3	Cavaliers	17
James Jones	-15	0-1	0	Warriors	1	3	20150611	0-3	0-0	18	0	1	0	SF	3	0	1	Cavaliers	18
Kendrick Perkins	-2	0-0	0	Warriors	0	1	20150611	0-2	2-2	3	0	1	2	C	1	0	0	Cavaliers	19
Joe Harris	-2	0-0	0	Warriors	0	0	20150611	0-0	1-2	3	0	0	1	SG	0	0	0	Cavaliers	20
Mike Miller	-2	0-0	0	Warriors	0	0	20150611	0-0	0-0	3	0	0	0	SG	0	0	0	Cavaliers	21
J.R. Smith	-27	0-8	2	Warriors	0	1	20150611	2-12	0-0	28	1	4	4	SG	2	0	0	Cavaliers	22
Cavaliers		4-27	16	Warriors	3	33	20150611	29-88	20-28		16	19	82	Team	49	2	9	Cavaliers	23

Nous avons rencontré quelques problèmes dans l'exécution du script. En effet de nombreuses requêtes http sont rejetées par le serveur d'ESPN. Nous gardons tous les URLs pour lesquels nous avons « requêtes rejetées » dans une liste. Mais ces requêtes sont constamment rejetées par le serveur alors qu'elles ne présentent pas de particularités par rapport aux autres URLs. Ces rejets ne nous permettent donc pas d'obtenir tous les matchs de la saison ce qui nous pose beaucoup de problèmes au niveau du nombre de matchs par équipes et de la normalisation des données par rapport à toutes les autres équipes.

De plus, chaque année, à cause des Playoffs, le nombre de matchs joué pour chaque équipe varie beaucoup. Nous n'avons pas un nombre de match fixe et donc les statistiques pour chaque équipe ne sont pas homogènes.

Nous avons dû pour cela réaliser un script très modulable avec de nombreuses exceptions récupérées puisque de nombreux tableaux de données avaient des différences. Soit parce que le poste d'un joueur n'était pas précisé, soit parce que les équipes de NBA rencontraient des équipes européennes donc le format et l'intérêt de ces matchs n'était pas le même.

Nous récupérons ensuite quelques tableaux de statistiques généraux pour les équipes à savoir le nombre de matchs gagné dans la saison, la moyenne du nombre de point encaissé par match, etc… Ces graphiques nous permettent de réaliser une vue globale de la ligue. Et pouvoir faire une comparaison globale des équipes les unes par rapport aux autres.

	CONF	DIV	GB	HOME	L	L 10	PCT	ROAD	STREAK	TeamName	W
0	28-24	12-4	9	21-20	42	8-2	0.488	19-22	W 6	Boston	40
1	12-40	2-14	31	12-29	64	0-10	0.220	6-35	L 10	Philadelphia	18
10	35-17	9-7	1	31-10	27	5-5	0.671	24-17	W 1	Memphis	55
11	29-23	7-9	6	27-14	32	5-5	0.610	23-18	W 1	Dallas	50
12	33-19	11-5	0	27-14	33	7-3	0.598	22-19	W 1	Toronto	49
13	24-28	10-6	11	19-22	44	6-4	0.463	19-22	W 1	Brooklyn	38
14	11-41	5-11	32	10-31	65	3-7	0.207	7-34	L 1	New York	17
15	35-17	11-5	0	31-10	29	7-3	0.646	22-19	W 2	Cleveland	53
16	30-22	7-9	12	23-18	41	5-5	0.500	18-23	L 1	Milwaukee	41
17	23-29	6-10	21	18-23	50	4-6	0.390	14-27	W 1	Detroit	32
18	38-14	12-4	0	35-6	22	5-5	0.732	25-16	L 3	Atlanta	60
19	25-27	6-10	23	20-21	45	4-6	0.451	17-24	W 2	Miami	37
2	33-19	8-8	3	27-14	32	7-3	0.610	23-18	W 4	Chicago	50
20	15-37	4-12	35	13-28	57	3-7	0.305	12-29	L 4	Orlando	25
21	31-21	11-5	0	32-9	31	4-6	0.622	19-22	L 4	Portland	51
22	23-29	9-7	13	21-20	44	7-3	0.463	17-24	L 1	Utah	38
23	7-45	4-12	35	9-32	66	0-10	0.195	7-34	L 12	Minnesota	16
24	42-10	13-3	0	39-2	15	8-2	0.817	28-13	W 4	Golden State	67
25	21-31	6-10	28	22-19	43	1-9	0.476	17-24	L 5	Phoenix	39
26	9-43	2-14	46	12-29	61	2-8	0.256	9-32	L 3	L.A. Lakers	21
27	33-19	8-8	0	30-11	26	7-3	0.683	26-15	W 3	Houston	56
28	32-20	8-8	1	33-8	27	9-1	0.671	22-19	L 1	San Antonio	55
29	29-23	8-8	11	28-13	37	7-3	0.549	17-24	W 2	New Orleans	45
3	28-24	8-8	15	23-18	44	7-3	0.463	15-26	L 1	Indiana	38
4	30-22	10-6	14	29-12	36	6-4	0.561	17-24	L 2	Washington	46
5	25-27	8-8	27	19-22	49	2-8	0.402	14-27	L 6	Charlotte	33
6	25-27	10-6	6	29-12	37	4-6	0.549	16-25	W 2	Oklahoma City	45
7	19-33	6-10	21	19-22	52	3-7	0.366	11-30	L 2	Denver	30
8	37-15	12-4	11	30-11	26	9-1	0.683	26-15	W 7	L.A. Clippers	56
9	18-34	7-9	38	18-23	53	3-7	0.354	11-30	W 2	Sacramento	29

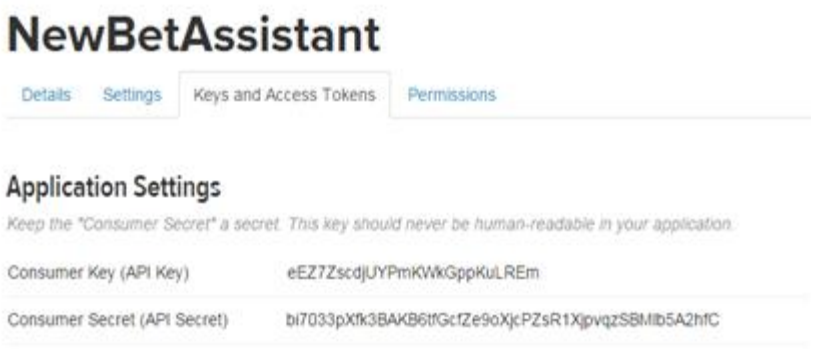
Nous avons besoin maintenant de pouvoir mettre à jour notre base de données sans avoir à recommencer toutes ces démarches très longues. Nous avons donc réalisé un autre script qui permet de faire cette mise à jour de façon très

optimale. Sachant que nous avons récupéré toutes les statistiques de tous les matchs jusqu'à une certaine date, nous pouvons alors récupérer la date maximum et faire les requêtes uniquement sur les matchs qui nous intéressent. Nous avons juste besoin de recréer les URLs correspondant, de récupérer les données, d'ouvrir notre ancienne DataFrame, de la mettre à jour et de la réécrire dans le même fichier. Cela permet de garder une base de données à jour après chaque soirée en optimisant le travail du serveur.

	A4F	ACH	CONS	Conference	DEFF	Division	EDIF	GP	LOSS	OEFF	OPPG	PACE	PDIF	PPG	RSoS	Rank	SAR	SoS	WIN	WIN%	eWIN%	index	pWIN%
Team																							
Atlanta	0.027	0.058	12.9	East	103.1	Southeast	5.8	82	22	108.9	97.1	93.9	5.4	102.5	0		4.60	-1.20	60	0.732	0.674	0	0.678
Boston	-0.020	-0.019	11.7	East	104.5	Atlantic	0.2	82	42	104.7	101.2	95.8	0.2	101.4	0		-0.73	-0.93	40	0.488	0.507	1	0.507
Houston	0.046	0.078	13.2	West	103.4	Southwest	3.5	82	26	106.9	100.5	96.6	3.4	103.9	0		4.01	0.51	56	0.683	0.605	10	0.612
Indiana	0.029	-0.046	12.7	East	103.3	Central	0.3	82	44	103.6	97.0	93.2	0.3	97.3	0		-0.01	-0.31	38	0.463	0.509	11	0.510
LA Clippers	0.037	0.008	15.2	West	105.5	Pacific	6.9	82	26	112.4	100.1	94.7	6.6	106.7	0		6.56	-0.34	56	0.683	0.675	12	0.717
LA Lakers	-0.065	0.005	10.7	West	110.7	Pacific	-7.2	82	61	103.5	105.3	94.0	-6.8	98.5	0		-6.09	1.11	21	0.256	0.251	13	0.276
Memphis	-0.026	0.054	11.8	West	102.2	Southwest	3.5	82	27	105.7	95.1	92.0	3.2	98.3	0		3.88	0.38	55	0.671	0.617	14	0.605
Miami	-0.016	0.033	13.9	East	106.8	Southeast	-2.9	82	45	103.9	97.3	90.9	-2.6	94.7	0		-2.49	0.41	37	0.451	0.418	15	0.414
Milwaukee	0.013	-0.012	12.8	East	102.3	Central	0.4	82	41	102.7	97.4	94.0	0.5	97.9	0		0.23	-0.17	41	0.500	0.512	16	0.516
Minnesota	-0.116	-0.024	12.0	West	112.2	Northwest	-9.3	82	66	102.9	106.5	94.4	-8.7	97.8	0		-7.20	2.10	16	0.195	0.219	17	0.214
New Orleans	0.021	0.027	14.5	West	107.4	Southwest	0.8	82	37	108.2	98.6	91.3	0.8	99.4	0		0.39	-0.41	45	0.549	0.522	18	0.526
New York	-0.064	0.002	12.3	East	110.0	Atlantic	-10.1	82	65	99.9	101.2	91.2	-9.3	91.9	0		-9.43	0.67	17	0.207	0.205	19	0.194
Brooklyn	-0.028	0.037	16.1	East	107.4	Atlantic	-3.0	82	44	104.4	100.9	92.8	-2.9	98.0	0		-2.85	0.15	38	0.463	0.426	2	0.405
Oklahoma City	0.024	-0.020	13.2	West	105.5	Northwest	2.3	82	37	107.8	101.8	95.7	2.2	104.0	0		2.51	0.21	45	0.549	0.569	20	0.572
Orlando	-0.027	0.008	11.5	East	107.7	Southeast	-6.1	82	57	101.6	101.4	93.8	-5.7	95.7	0		-6.02	0.08	25	0.305	0.297	21	0.312
Philadelphia	-0.055	-0.030	13.8	East	104.8	Atlantic	-9.3	82	64	95.5	101.0	95.7	-9.0	92.0	0		-8.04	1.26	18	0.220	0.250	22	0.204
Phoenix	-0.005	0.005	12.5	West	106.2	Pacific	-0.9	82	43	105.3	103.3	96.3	-0.9	102.4	0		-0.89	0.01	39	0.476	0.471	23	0.470
Portland	0.064	-0.031	11.5	West	103.7	Northwest	4.5	82	31	108.2	98.6	94.2	4.2	102.8	0		3.47	-1.03	51	0.622	0.653	24	0.638
Sacramento	0.015	-0.034	13.3	West	109.2	Pacific	-3.8	82	53	105.4	105.0	95.3	-3.7	101.3	0		-2.98	0.82	29	0.354	0.388	25	0.378
San Antonio	0.055	-0.030	12.3	West	102.0	Southwest	6.5	82	27	108.5	97.0	93.8	6.2	103.2	0		6.43	-0.07	55	0.671	0.701	26	0.704
Toronto	-0.014	0.003	13.8	East	107.8	Atlantic	3.3	82	33	111.1	100.9	92.8	3.1	104.0	0		2.88	-0.42	49	0.598	0.595	27	0.602
Utah	0.039	-0.043	14.2	West	104.9	Northwest	0.2	82	44	105.1	94.9	90.4	0.2	95.1	0		0.79	0.59	38	0.463	0.506	28	0.507
Washington	0.050	0.041	15.7	East	102.9	Southeast	0.8	82	36	103.7	97.8	93.7	0.7	98.5	0		-0.21	-1.01	46	0.561	0.520	29	0.523
Charlotte	-0.061	-0.013	15.9	East	103.5	Southeast	-3.4	82	49	100.1	97.3	93.0	-3.1	94.2	0		-3.36	0.04	33	0.402	0.415	3	0.398
Chicago	0.047	0.014	13.1	East	104.4	Central	3.2	82	32	107.6	97.8	92.7	3.0	100.8	0		3.11	-0.09	50	0.610	0.596	4	0.599
Cleveland	0.039	0.030	16.5	East	106.3	Central	4.9	82	29	111.2	98.7	92.3	4.4	103.1	0		4.68	-0.22	53	0.646	0.616	5	0.645
Dallas	-0.028	0.034	16.1	West	106.4	Southwest	3.1	82	32	109.5	102.3	95.2	2.9	105.2	0		3.15	0.05	50	0.610	0.576	6	0.595
Denver	-0.036	-0.039	15.3	West	108.2	Northwest	-3.7	82	52	104.5	105.0	96.1	-3.5	101.5	0		-3.12	0.58	30	0.366	0.405	7	0.385
Detroit	-0.038	-0.078	12.4	East	106.4	Central	-1.0	82	50	105.4	99.5	92.8	-1.0	98.5	0		-0.80	0.20	32	0.390	0.468	8	0.467
Golden State	0.094	0.027	12.8	West	101.3	Pacific	10.3	82	15	111.6	99.9	98.3	10.1	110.0	0		8.79	-1.51	67	0.817	0.790	9	0.833

Extraction de tweets

Pour apporter une plus grande précision à nos prédictions, nous avons décidé de prendre aussi en compte l'avis des personnes. Pour cela, on récupère des données sur un fameux réseau social « Twitter ». Ce dernier nous rend la tâche plus simple puisqu'il nous fournit un outil qui nous permet de récupérer directement des tweets. Pour cela, il nous suffit d'ouvrir un compte développeur sur Twitter, de récupérer ses clés d'accès et écrire un script en utilisant la librairie « Tweepy ».



Nous avons notamment sélectionné les tweets en relation avec une équipe en utilisant la fonction *twitterStream.filter(track=["#leHashTag"])* de Tweepy qui permet de récupérer les tweets en fonction du Hashtag passé en paramètre. Puis, nous avons « nettoyé » les tweets en supprimant certaines informations telles que le pseudonyme de l'utilisateur ou encore la date et l'heure de l'envoie du tweet. Cependant, bien que la tâche soit simple, la récupération des données fût longue. En effet, l'API de Twitter nous limite à 15 requêtes tous les 15 minutes, ce qui est très peu pour notre utilisation. Il nous a

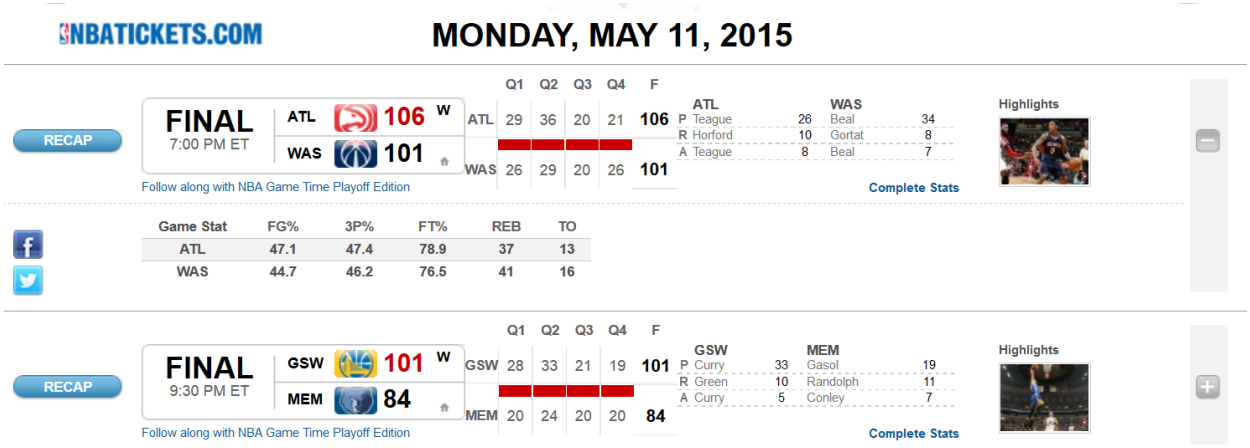
donc fallu plus d'une semaine pour récupérer une base de données qui nous semblait correcte.

Les Tweets récupérés et nettoyés, vient la tâche la plus longue et la plus fastidieuse avant de pouvoir analyser les tweets. En effet, pour pouvoir entrainer notre base, il nous a fallu étiqueter chaque tweet par un « positif », « négatif » ou « neutre » nous même pour plus de précision. En étiquetant, nous nous sommes rendus compte que le nombre de tweets utilisable est très peu par rapport au nombre de tweets récupérés. En effet, plus de la moitié des tweets sont des « retweets », c'est-à-dire des doublons. Nous avons donc décidés d'en récupérer beaucoup plus, et finir à la fin avec une base de données de 600 tweets exploitables.

```
9  "caw caw!!! let's get fired up!!! #hawksvcavs",positive
10 "#hawksvcavs tonight #truetoatlanta",neutre
11 "sadly the cavs taking this one in 5 , fuck lebron",negative
```

Récupération des matchs et résultats de la veille

Nous avons aussi besoin pour notre site web de récupérer les matchs de la soirée ainsi que les résultats de la veille afin de les afficher sur notre application mobile et notre site internet. Pour cela nous récupérons la date d'aujourd'hui et celle d'hier afin de pouvoir recréer les URLs correspondant à nos requêtes. Nous récupérons alors grâce aux méthodes expliquées plus tôt toutes les informations dont nous avons besoin. Soit les équipes qui jouent à domicile et celles qui jouent à l'extérieur ainsi que l'heure locale du match.



Nous récupérons aussi les résultats de la veille avec le score du match. Nous créons tous les jours un fichier .txt au format Json comportant toutes ces informations et qui est très facile de lire à partir du site web. Nous postons aussi ces informations sur notre serveur web pour que notre application mobile puisse y accéder facilement.

Stockage et format de données

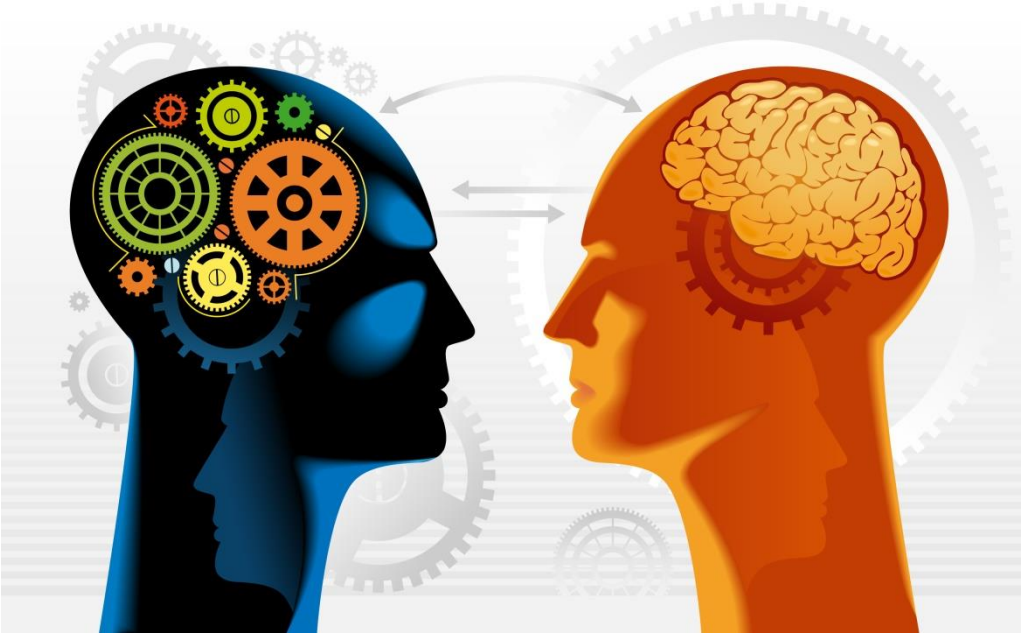
Nous avons décidé pour un souci évident d'optimisation du code de stocker le plus grand nombre de données récupérées en local. Nous créons alors de nombreux fichiers textes comportant le résultat de toutes nos requêtes afin d'éviter de surcharger le serveur et de réaliser des requêtes inutiles.

Nous avons choisi de stocker toutes ces données dans des fichiers .txt mais au format JSON (JavaScript Object Notation). Le format JSON est un des formats de stockage les plus optimaux il est composé d'un ensemble ordonné de clés et de valeurs. Les DataFrame pandas permettent directement de transformer ces dernières en objets JSON que nous écrivons directement dans des fichiers.

Cela nous permet aussi d'avoir un accès très rapide à toutes nos données puisque l'ouverture et lecture d'un fichier local se fait en quelques secondes pour les plus gros. En effet notre plus gros stock de données est d'environ 30Mo et comporte plus de 100 000 lignes par 2 colonnes.

Analyse et traitement

Maintenant que nous avons toutes ces données, il nous faut les analyser afin de réaliser la prédiction des matchs. Nous utiliserons le Machine Learning qui permet de créer un modèle qui va apprendre sur un certain nombre de données de tests et essayer d'en déduire une règle, un seuil permettant de définir si le match est gagné ou perdu.



Exploitation des statistiques

La régression logistique

La technique que nous avons choisie pour déterminer quelle équipe allait gagner ou non est celle de la **régression logistique**. C'est une technique de classification qui permet de pouvoir calculer grâce à certains paramètres choisis appelés **variables explicatives (ou Features),** à quelle classe appartient un match donné. Ces variables sont choisies de sorte que l'on puisse facilement identifier un match gagné d'un match perdu. Pour notre projet, nous avons choisi de séparer en **deux classes** chaque match que nous avons récupéré. Chaque match se voit attribuer -1 ou 1 suivant que l'équipe extérieure (AwayTeam) ou l'équipe domicile (HomeTeam) a gagné. Si l'AwayTeam a gagné, le match est associé au nombre 1. Si l'HomeTeam a gagné, le match est associé au nombre -1. Prenons un exemple similaire : imaginons que l'on veuille déterminer le sexe d'une personne à partir de son poids et de sa taille. Homme et femme sont donc les deux classes que l'on essaye de déterminer, et les variables explicatives ici sont le poids et la taille.

Ainsi la prédiction consiste donc à savoir si un match correspond à la classe "1" ou à la classe "-1" ! Pour ce faire il faut calculer un rapport de chance appelé **odd ratio**. Cet odd ratio prend en compte deux variables : X et Y. X correspond aux variables explicatives et Y correspond aux classes à prédire. Dans notre cas, nous avons Y = {1,-1}.

$$odd(x) = \frac{Pr(Y = 1|X = x)}{Pr(Y = -1|X = x)}$$

L'odd ratio ici est le rapport de probabilité qu'un match soit de la classe 1 ou -1 suivant certains paramètres x. Nous pouvons déduire de ce rapport le résultat du match : s'il est supérieur à 1, le match a plus de probabilité d'être de la classe 1 et inversement s'il est inférieur à 1, il a plus de chance d'être de classe -1.

Posons : $\pi(x) = Pr(Y = 1|X = x)$

On a donc : $Pr(Y = -1|X = x) = 1 - \pi(x)$

Ainsi l'odd ratio devient : $odd(x) = \frac{\pi(x)}{1 - \pi(x)}$

Il nous faut donc trouver ce $\pi(x)$. Pour ce faire, nous utilisons une **hypothèse fondamentale** qui consiste à supposer que le logarithme d'odd ratio est de la forme polynomiale. Dans le cas d'une seule variable explicative x, nous avons :

$$\log(\frac{\pi(x)}{1 - \pi(x)}) = \beta_0 + \beta_1 x$$

Or l'inverse de la fonction $l(t) = \log(\frac{t}{1-t})$ est la fonction **sigmoïde** :

$$s(x) = \frac{1}{1 + e^{-x}}$$

En appliquant la sigmoïde à l'égalité précédente, nous obtenons donc :

$$s(\log(\frac{\pi(x)}{1 - \pi(x)})) = s(\beta_0 + \beta_1 x) \Rightarrow \pi(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

Nous arrivons à une expression où on peut déduire $\pi(x)$ à partir des coefficients β_0 et β_1. Notre but sera de calculer ces coefficients à l'aide de python afin d'en déduire le rapport des chances et donc de savoir à quelle classe appartient un match.

Cependant, dans notre cas à nous, il y a plus d'une variable explicative (assists, rebonds, steals,...). L'hypothèse fondamentale devient donc :

$$\log\left(\frac{\pi(x)}{1-\pi(x)}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

Avec n le nombre de variables explicatives. Ensuite on applique la sigmoïde et on trouve les coefficients bêtas comme précédemment.

Utilisation des fonctions de python pour une régression

Afin d'analyser et traiter les données, nous utilisons **l'apprentissage automatique** (Machine Learning). Comme son nom l'indique, cette méthode donne aux ordinateurs la capacité d'apprendre et de pourvoir classifier ou deviner quelque chose suivant plusieurs features (les variables explicatives).

Comme expliqué précédemment, nous commençons par étiqueter chaque match récupéré par 1 ou -1 pour différencier les 2 classes. Voici un exemple de classification linéaire en utilisant une régression logistique :

Il faut tout d'abord créer un objet **LogisticRegression** :

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
```

Ensuite on définit les variables explicatives et cibles (features et target).

AwayAssist	AwayBlock	AwayCoeff	AwayPoints	...	HomeRebound	HomeSteal	win
27	6	0.200000	90.840000	...	45.000000	8.000000	1
16	3	0.342857	96.264706	...	41.000000	5.000000	1
24	3	-0.169925	96.500000	...	48.000000	6.000000	1
17	0	0.372932	95.921875	...	38.000000	6.000000	1
24	3	0.321805	97.437500	...	44.000000	8.000000	-1
20	8	0.580451	101.951220	...	38.000000	9.000000	-1
11	3	-0.536842	96.371429	...	53.000000	7.000000	-1
20	6	-0.795489	97.176471	...	44.000000	10.000000	-1
26	2	0.443609	100.647059	...	37.000000	6.000000	1
20	4	-0.061654	101.214286	...	41.000000	5.000000	-1
19	5	-0.049624	104.242424	...	45.000000	10.000000	-1
22	9	0.040602	95.829268	...	42.000000	4.000000	1
20	6	0.231579	100.025641	...	36.000000	6.000000	-1
27	5	0.503759	103.513514	...	39.000000	9.000000	-1
20	5	-0.419549	101.106383	...	40.000000	8.000000	-1
18	2	0.380451	101.709091	...	40.000000	15.000000	-1
22	3	-0.560902	95.972222	...	34.000000	8.000000	-1
27	1	-0.354887	95.162791	...	40.000000	6.000000	-1
16	6	0.049624	96.000000	...	36.000000	8.000000	1
20	3	-0.406015	101.131579	...	56.000000	7.000000	-1
32	8	0.463158	103.578947	...	35.000000	7.000000	1
16	9	-0.550376	91.780488	...	54.000000	4.000000	-1
34	6	0.224060	99.750000	...	59.000000	9.000000	1
29	4	-0.332331	95.948718	...	46.000000	6.000000	1
17	4	0.127820	103.851064	...	42.000000	6.000000	1
20	5	-0.589474	100.461538	...	52.000000	9.000000	-1
19	4	0.052632	99.937500	...	39.000000	10.000000	1
19	6	0.212030	101.371429	...	41.000000	12.000000	-1
20	3	-0.102256	100.853659	...	44.000000	8.000000	1
32	8	-0.386466	99.289474	...	40.000000	6.000000	1

Features **Target**

Les variables explicatives et cibles sont des données obtenues précédemment sur les joueurs et équipes. Chaque statistique d'un joueur ou d'une équipe correspond à une variable explicative et nous définissons la cible comme le résultat du match car c'est l'output (la sortie) que nous voulons obtenir dans notre cas.

Nous utilisons un module de python nommé sklearn qui contient tous les outils nécessaire à l'analyse des données dont la fonction **train_test_split().** Cette fonction permet de diviser les données en deux parties : **la base**

d'apprentissage et **la base de test**. Habituellement la base d'apprentissage prend 75% des données et la base de test 25%.

La **base d'apprentissage** permet via un algorithme d'apprentissage de "faire apprendre" à la machine les valeurs de données qui font obtenir tel résultat. Par exemple supposons que toutes les équipes marquant plus de 90 points remportent le match alors cette caractéristique sera considérée par la machine comme très favorable pour l'obtention d'un résultat dans la classe "victoire".

La **base de test** permet de tester l'apprentissage de la machine. En effet puisque nous avons les cibles, nous savons exactement quels matchs ont été gagnés par qui. En comparant les résultats de matchs prédits avec la base d'apprentissage et les vrais résultats, nous pouvons ainsi en déduire un **score** de réussite.

En Python, on fait de la manière suivante :

On commence par créer la base de test et d'apprentissage :

```
features_train, features_test, target_train, target_test
= train_test_split(features, target)
```

Ensuite on construit le modèle en faisant un "fit" qui permet de faire apprendre à la machine :

```
classifier.fit(features_train, target_train)
```

Voilà ! Notre modèle est construit. On peut voir à quel point il est performant en regardant le score sur la base de test avec :

```
scoretest=classifier.score(features_test,target_test)
```

Cependant pour pouvoir tester les performances d'un modèle, on utilise généralement la validation croisée et plus particulièrement **la k-fold cross validation**. Cette méthode consiste à diviser les données en k échantillons. Un échantillon est choisi au hasard pour être l'échantillon de test et les autres sont des échantillons d'apprentissage et un modèle est construit depuis ces échantillons. Finalement l'erreur quadratique moyenne est calculée. Ce processus se déroule jusqu'à ce que chaque échantillon soit devenu un échantillon de test. La moyenne des erreurs quadratiques donne l'erreur du modèle original.

La validation croisée nous donne k scores (un pour chaque échantillon), puis nous faisons la moyenne de ces scores. Le score obtenu est celui qui nous permet de voir à quel point notre modèle est performant. En python, il suffit de faire :

```
s=cross_val_score(classifier,features,target,cv=10)
print("score moyen", np.mean(s))
```

Ce qui nous donne le score moyen :

```
('score moyen', 0.86149332815446211)
```

Nous avons réussi à obtenir un **score moyen de 86%** !

A l'aide de quelques fonctions nous pouvons aussi afficher la matrice de confusion ainsi que les diverses métriques qui vont avec :

```
print(confusion_matrix(classifier.predict(features_test),
target_test))

print(metrics.classification_report(classifier.predict(feature
s_test), target_test))
```

```
          TP   FP
[[597   86]
 [ 76 350]]
          FN   TN
```

	precision	recall	f1-score	support
-1	0.89	0.87	0.88	683
1	0.80	0.82	0.81	426
avg / total	0.85	0.85	0.85	1109

TP : True Positive Nombre de matchs où le modèle dit que le match est remporté par l'équipe qui joue à l'extérieur et que c'est le cas en réalité

FN : False Negative Nombre de matchs où le modèle dit que le match est remporté par l'équipe qui joue à l'extérieur alors que c'est l'équipe à domicile qui a gagné en réalité

FP : False Positive Le contraire

TN : True Negative Nombre de matchs où le modèle dit que le match n'est pas remporté par l'équipe qui joue à l'extérieur et que c'est le cas en réalité

Precision = $\frac{nombre\ de\ TP}{nombre\ de\ TP+nombre\ de\ FP}$, la proportion des TP

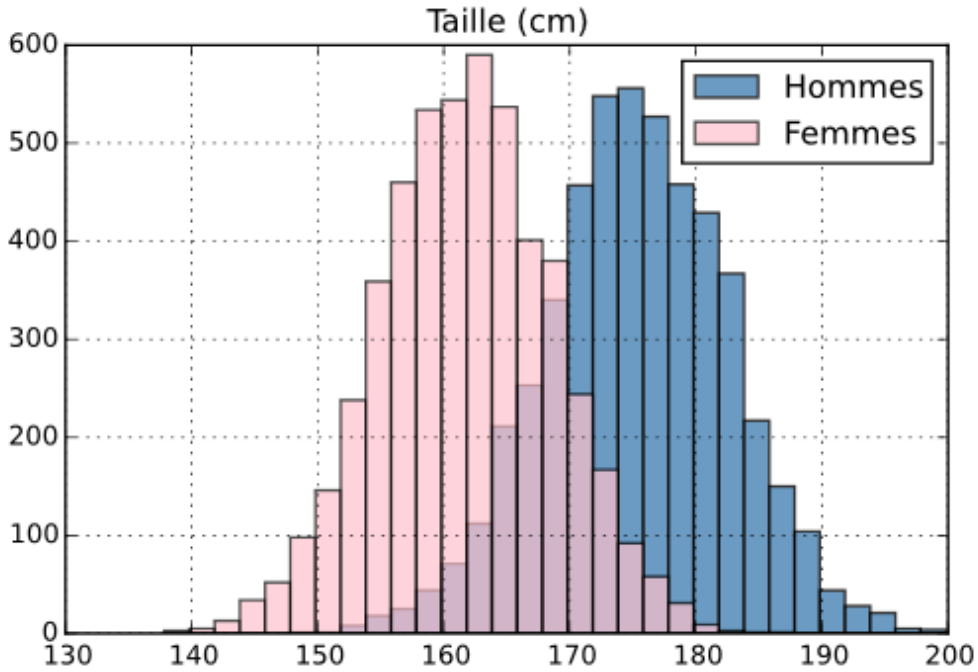
Recall = $\frac{nombre\ de\ TP}{nombre\ de\ TP+nombre\ de\ FN}$, la proportion des TP détectés

$f1 - score = 2 * \frac{precision * recall}{precision + recall}$, moyenne pondérée de la précision et du recall.

Support = nombre d'occurences de chaque classes (victoire et défaite) dans le target_test.

Recherche de bonnes variables explicatives

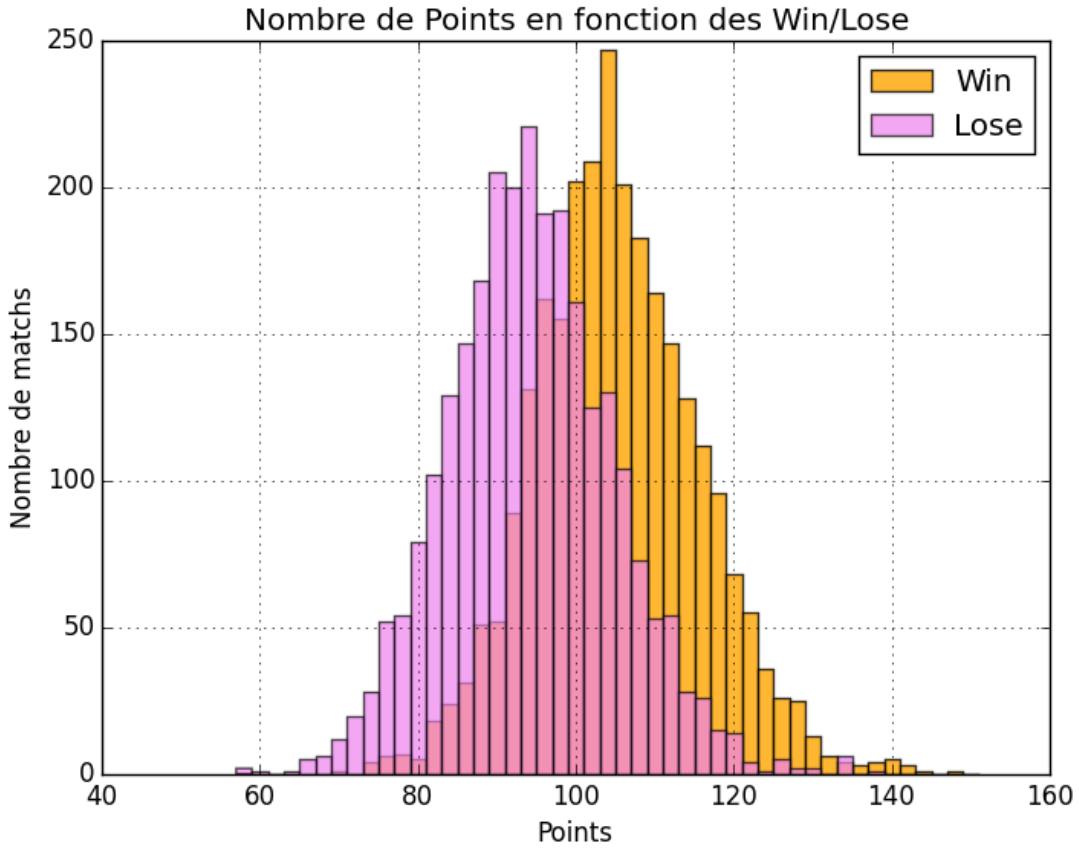
Précédemment pour illustrer une classification, nous avons utilisé l'exemple de prédiction d'homme et femme à partir du poids et de la taille. Ci-dessous, un histogramme montrant la taille en fonction du nombre d'individu et de leur sexe :



Source : http://perso.esiee.fr/~bercherj/IT3007/R%C3%A9gression_logistique.html

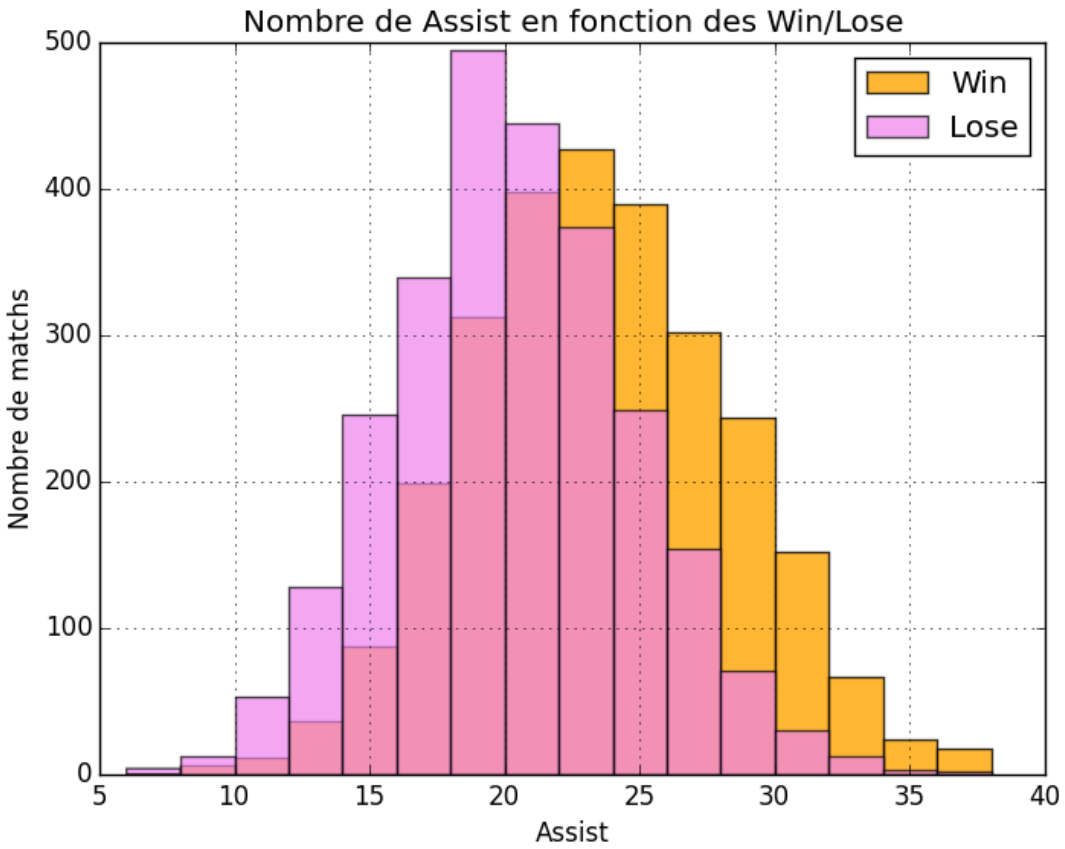
On voit que la majorité des femmes font en moyenne 160 cm alors que les hommes font en moyenne 175 cm. La distinction des deux histogrammes n'est pas parfaite mais reste néanmoins très bonne. La taille est donc un bon critère de distinction entre hommes et femmes. C'est donc une bonne **variable explicative** pour la classification d'homme/femme.

Nous avons donc fait de même avec les matchs de la NBA ! Quelles **statistiques** sont importantes pour qu'une équipe gagne un match ? Plus précisément, à partir de quel **seuil** un match est-il considéré gagné ? Pour le savoir, nous avons créé plusieurs graphiques similaires à celui du dessus :



Ce graphique montre que plus une équipe marque de points, plus elle a de chance de gagner un match. Logique ! Malheureusement, nous ne pouvons pas nous servir du nombre de points pour une régression logistique car le nombre de points est directement lié à la victoire ou à la défaite. Le modèle aurait donc un taux de réussite de 100%...

Nous avons donc cherché d'autres statistiques qui ne seraient pas directement liés à la victoire ou à la défaite, mais qui auraient quand même une influence grande.



Un “assist” correspond à une action où une passe a été effectuée avant qu’une équipe marque un panier. Un assist conduit donc toujours à un panier, mais un panier n’est pas forcément précédé d’un assist ! Forcément, comme dans le graphique précédent, un grand nombre d’assists conduit le plus souvent à une victoire. Cependant les assists sont une statistique indirectement liée à la victoire cette fois ci. Elle correspond un peu à l’importance du jeu d’équipe. C’est pour cela que nous avons choisi d’inclure cette statistique.

Il se peut qu'une variable explicative ne soit en réalité pas très utile pour une régression car trop instable. Dans ce cas, il vaut mieux la retirer car elle pourrait fausser les résultats.

Au départ nous avons ajouté toutes les variables que nous trouvions sur les sites de la NBA ou ESPN : assists, rebonds, vol de ballons, nombre de fautes, etc. En faisant la régression (avec le "classifier" et l'apprentissage avec "fit"), nous pouvions ainsi voir quelles variables étaient utiles ou non à la prédiction. Cela se caractérise par un coefficient que le classifier donne pour chaque variable selon qu'elle ait été utile ou non. Pour voir ces coefficients, on tape la commande :

```
coeffs= classifier.coef_[0]
```

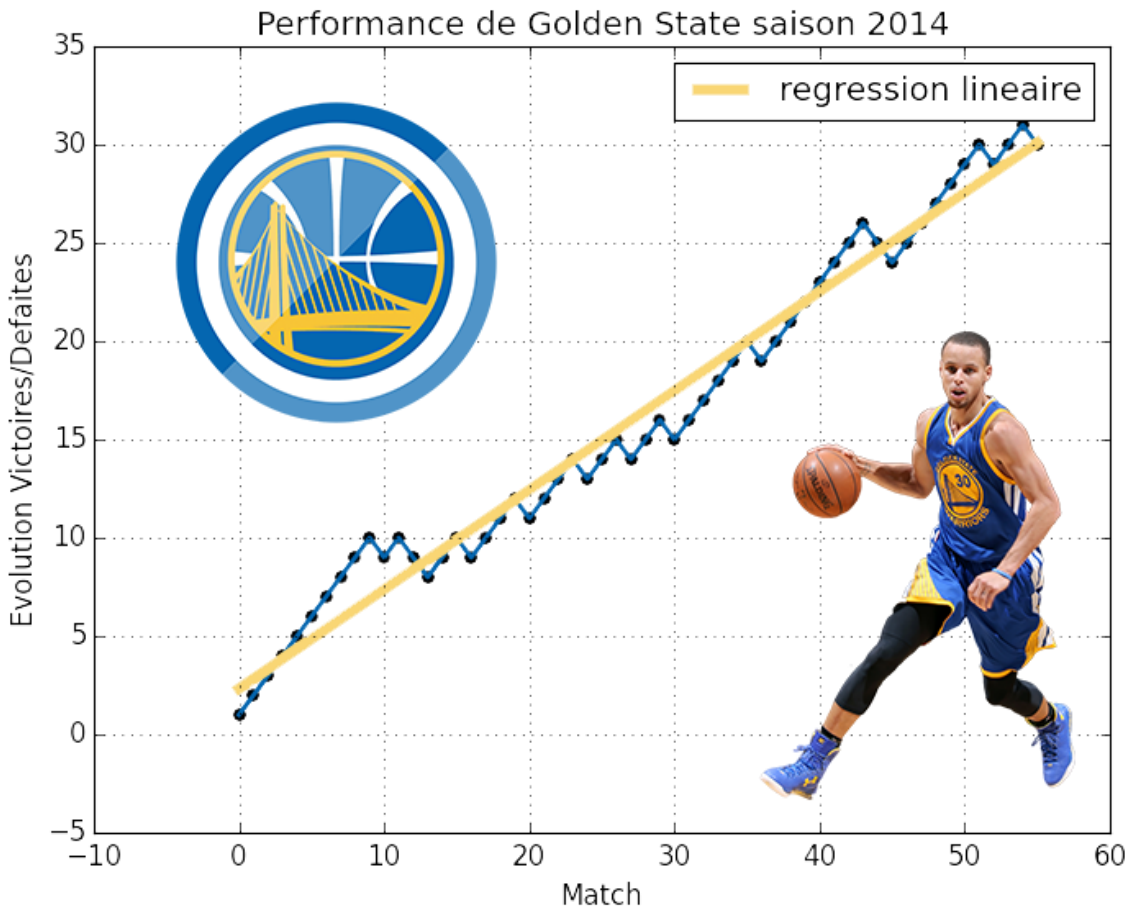
Ce qui nous donne :

```
Les coefficients :
('AwayAssist', 0.31862304974958738)
('AwayBlock', 0.084968717383792775)
('AwayCoeff', 0.028993008455812402)
('AwayRebound', -0.079577055785883519)
('AwaySteal', 0.13309888625638741)
('AwayWinStreak', 0.12265751025401672)
('Awaydreb', 0.90887702664509551)
('Awaypfoul', -0.35496529663191112)
('DiffAssist', 0.5803550476816931)
('DiffBlock', 0.1252573588175217)
('DiffCoeff', 0.041257678130135256)
('DiffPoints', 0.18115562105165212)
('DiffRebound', -0.034097856030070636)
('DiffSteal', 0.2218698368679583)
('DiffTo', -1.3938570158256451)
('Diffdreb', 1.2932490502852176)
('Diffpfoul', -0.44580753395375444)
('HomeAssist', -0.46115850834516964)
('HomeBlock', -0.087078461347045208)
('HomeCoeff', -0.026622070133135417)
('HomeRebound', -0.030856999883516582)
('HomeSteal', -0.16875801894971648)
('HomeWinStreak', -0.17956995556895577)
('Homedreb', -0.83425085155395728)
('Homepfoul', 0.17850429039972238)
```

On voit par exemple que la variable AwayAssist (nombre d'assist pour l'équipe extérieure) à un coefficient assez important, donc le modèle donne pour la prédiction une grande importance pour celle-ci.

Nous avons séparé chaque statistique en deux selon qu'elle appartient à l'équipe extérieur (variable "Away") ou domicile (variable "Home"). Nous avons également essayé de coupler ces variables en faisant par exemple la différence d'assist entre les deux teams d'un match (les variables commençant par "Diff").

La tendance d'une équipe est importante en sport car cela accroît la confiance et donc augmente les chances de victoires. Nous avons donc pris en compte celle-ci en tant que variable explicative :



Nous avons également pris en compte les statistiques des joueurs eux-mêmes. Pour commencer nous avons choisi pour chaque équipe les 5 joueurs ayant le plus joué pendant un match (pour chaque poste) donc ayant eu le plus de chance d'avoir un impact sur le match.

Ensuite nous avons calculé un score intermédiaire obtenu avec une opération sur les statistiques de ces joueurs (une fonction du type f(x)=coeff*(stat du joueur de l'équipe à l'extérieur - stat du joueur à domicile) et ce pour tous les postes). Ces scores sont obtenus en prenant en compte l'importance des statistiques selon les postes, par exemple les assists sont plus importants pour les meneurs de jeu que pour les pivots qui eux s'occupent généralement plus des rebonds. Ainsi nous avons des features pour chaque stat des joueurs qui sont ces scores intermédiaires.

Différents types de classification

Nous avons tenté d'utiliser plusieurs types de modèles qui étaient basé sur d'autres classifiers tels que le RandomForestClassifier et l'AdaBoost. Néanmoins au fil du temps le score le plus élevé provenait toujours de la classification avec une régression logistique. Il est possible que notre inexpérience ait joué un rôle car nous n'étions pas familiers avec tous les types de classifiers et le choix des paramètres optimaux, qui sont extrêmement nombreux, ce qui implique des réglages très précis.

Il existe aussi d'autres méthodes pour améliorer le score obtenu par validation croisée tel que le stacking/blending qui est le fait de prendre les résultats de plusieurs modèles puis de refaire un apprentissage sur ces résultats pour pouvoir obtenir de meilleurs résultats que nous n'avons pas implémenté mais nous avons toutefois utilisé le BaggingClassifier mais le score était plus faible car il est apparemment conseillé de l'utiliser dans des modèles où un petit changement dans la base d'apprentissage résulte en un grand changement dans le modèle final. Le bagging est une méthode d'ensemble learning et le BaggingClassifier prend comme modèle un DecisionTree. Le bagging construit plusieurs bases d'apprentissages qui correspondent chacun à un modèle différent. Ensuite les résultats de ces modèles sont soumis à un vote pour créer un résultat final.

La prédiction

Après avoir construit notre modèle et vérifié qu'il correspondait à nos attentes, nous pouvons commencer la prédiction. Pour ce faire, nous prenons les moyennes des statistiques des deux équipes **avant** la date du match. Nous prenons la moyenne sur la saison. Par exemple : imaginons que nous voulions prévoir l'issue du 6e match de la finale des Playoffs 2015, les Cleveland Cavaliers (CLE) contre les Golden State Warriors (GS). Ce match s'est déroulé le 16 juin 2015. Nous prenons donc la moyenne des statistiques sur toute la saison 2014-2015 **avant cette date** pour les deux équipes. Puis nous les rentrons dans une liste (nous l'avons appelée x dans notre code).

Enfin, nous faisons une prédiction avec cette liste à l'aide de la fonction predict : `classifier.predict(x)`. Cette fonction nous donne la classe prédite, -1 ou 1 :

```
Classe predite : [-1]
```

De plus, la fonction `classifier.predict_proba(x)` nous permet de voir avec quel pourcentage de chance il s'agit bien de telle ou telle classe. C'est ce pourcentage que nous affichons sur le site web et sur l'application mobile :

```
('Probas correspondantes', array([[ 0.74023105,  0.25976895]]))
```

A gauche nous avons le pourcentage que le match soit de classe -1, à droite de classe 1.

Pour les 6 matchs de la finale de la saison 2015, nous avions prévu à chaque fois une victoire des Golden State Warriors :

Away Team	Home Team	Date	Prédiction	Vrai vainqueur
CLE	GS	04-juin-15	GS à 81%	GS
CLE	GS	07-juin-15	GS à 78%	CLE
GS	CLE	09-juin-15	GS à 72%	CLE
GS	CLE	11-juin-15	GS à 65%	GS
CLE	GS	14-juin-15	GS à 77%	GS
GS	CLE	16-juin-15	GS à 68%	GS

CLE = Cleveland Cavaliers
GS = Golden State Warriors

Bonne prédiction
Mauvaise prédiction

Exploitation des tweets

L'analyse des tweets est une partie très importante. Nous avons donc réutilisé des éléments du cours en l'adaptant pour notre situation. A l'aide de plusieurs bibliothèques telles que NLTK, Numpy ou encore ScikitLearn, nous avons écrit une fonction qui classifie nos tweets. Tout d'abord, nous avons vectorisé les tweets récupérés en utilisant les fonctions *TfidfVectorizer(), fit(X), transform(X)* qui permettent respectivement de convertir une collection de documentation brut en une matrice de features de Term Frequency Inverse Document Frequency (TF-IDF) en fonction des paramètres que nous lui imposons, d'entrainer le vocabulaire et de transformer le document en une matrice de features. Puis nous avons choisi le classificateur MultiNomial naïve bayesienne qui est adapté pour la classification de caractéristiques spécifiques, entre autre ici, la classification d'un texte. On sépare ensuite nos données avec la fonction *train_test.split()* en deux parties pour tester nos données sur une partie entrainée. Il est alors très facile de construire, à l'aide de Pandas et ScikitLearn, une matrice de confusion ou un Rapport de classification qui permet d'évaluer la qualité d'une classification. Cette matrice est obtenue en comparant les données classées avec des données de référence qui doivent être différentes de celles ayant servi à réaliser la classification.

Par la suite, nous avons testé plusieurs classifications possibles en choisissant différents valeurs pour nos variables :

- Tweet Positif contre Tweet Négatif

Pour cela, on récupère les indices des tweets dont un sentiment est présent, c'est-à-dire, qu'on enlève tous les tweets neutres et on choisit ces tweets pour nos deux valeurs de variable.

```
Positive contre negative

Score =  0.959785522788

Matrice de confusion
====================
Predicted  negative  positive  All
True
negative        235        14  249
positive          1       123  124
All             236       137  373

Rapport de classification
=========================
             precision    recall  f1-score   support

   positive       0.90      0.99      0.94       124
   negative       1.00      0.94      0.97       249

avg / total       0.96      0.96      0.96       373
```

- Tweet avec sentiment contre Tweet neutre

On procède au même processus, en récupérant les tweets sans/avec sentiments. Pour une compréhension plus simple, nous avons étiqueté les tweets sans sentiment par un « No » et les tweets avec sentiments par un « Yes ».

```
Sentiment contre neutre

Score =  0.893867924528

Matrice de confusion
====================
Predicted  No  Yes  All
True
No         39    5   44
Yes        40  340  380
All        79  345  424

Rapport de classification
=========================
             precision    recall  f1-score   support

         No       0.49      0.89      0.63        44
        Yes       0.99      0.89      0.94       380

avg / total       0.93      0.89      0.91       424
```

Pour atteindre une meilleure précision, nous avons procédé à deux autres méthodes. La première a été de réaliser un second nettoyage en remplaçant les abréviations en anglais par leur forme conventionnelle. Pour cela, rien de plus simple que d'écrire une liste d'abréviations souvent rencontrées accompagnées de leur forme non abrégées. La deuxième a été d'utiliser une méthode de NLTK pour insérer une valeur intrinsèque à nos mots. Cette deuxième méthode donne au mot son genre et une définition, ainsi qu'un certain poids (positif ou négatif) au mot. En comparant les résultats obtenus avant et après ces deux méthodes appliquées, on constate seulement une légère hausse de la précision, qui est tout de même significative étant donné notre pourcentage de précision déjà élevé.

Après toute cette phase de préparation, nous avons testé notre base en analysant une centaine de tweets récupérés la veille d'un match de NBA. On

s'est vite rendus compte que le nombre de tweets étiquetés avec sentiment négatif est peu comparé au tweets neutres ou positifs.

Nous avons donc complété notre base en lui rajoutant une autre base qui contient plus de 1000 tweets étiquetés. Notre programme nous donne alors la nature, accompagnée de leur pourcentage, des tweets récupérés dans un fichier.

Nous obtenons un score qui s'approche de 0,90 de précision.

Par la suite, nous avons écrit une petite fonction pour récupérer un pourcentage sur les nouveaux tweets récupérés marqué comme positif ou négatif, afin de l'intégrer dans notre analyse statistique. Cela nous permet d'avoir l'avis des gens en général sur un match.

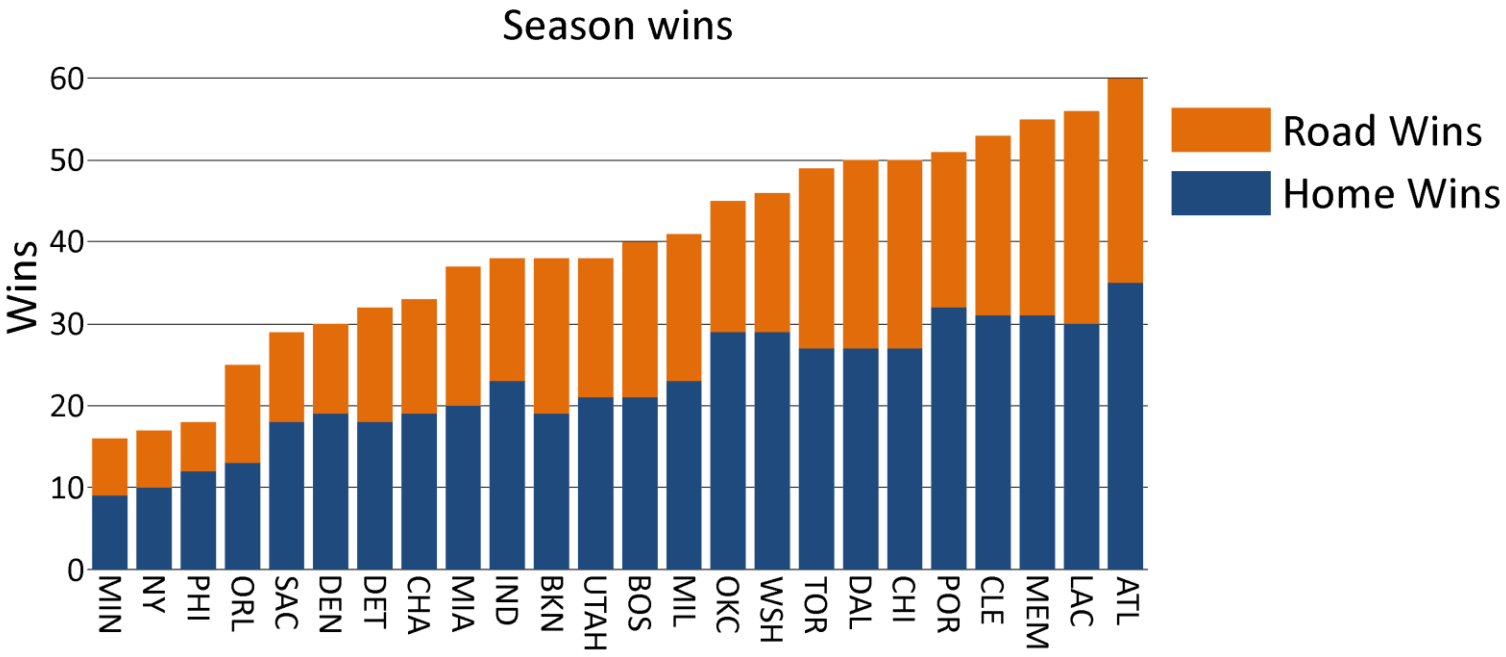
Cet avis peut nous être très utile pour pouvoir déterminer le pourcentage de chance qu'une équipe à de gagner. En effet, plus les gens parlent d'une équipe positivement, plus on peut penser que c'est un facteur indicatif de la victoire de cette équipe. Cette donnée n'est pas sûre à 100% mais nous avons pensé que ce serait sûrement intéressant de la placer dans notre modèle.

Graphiques et interprétation

Maintenant que la partie traitement et analyse des données à proprement parler est terminée, nous devons essayer de créer et d'afficher nos données dans un cadre pertinent afin de mieux comprendre les différentes relations entre les statistiques que nous avons récupérées.

Nous essayons tout d'abord de représenter une vue globale de la NBA. Nous commençons tout d'abord par afficher le nombre de victoire sur la saison courante. Ces victoires sont partagées entre victoires à domicile et victoires à l'extérieur. Pour toute cette partie nous utilisons la bibliothèque la plus connue de python en ce qui concerne l'affichage de graphiques à savoir MatplotLib (http://matplotlib.org/). Elle nous permet de réaliser à peu près tous les graphiques dont nous avons besoin, tout en ayant un grand contrôle sur les paramètres, ainsi que sur la gestion de l'affichage de notre graphique.

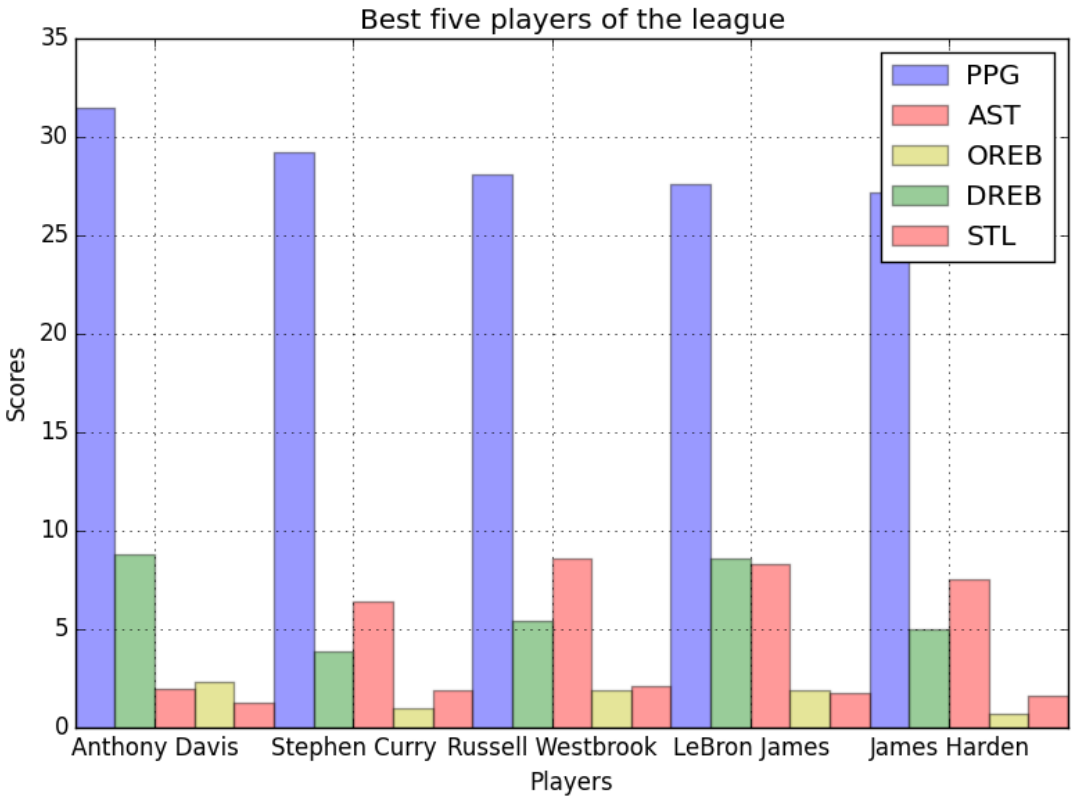
Pour certains graphiques nous avons utilisé le module plotly (similaire à mpld3) afin de pouvoir afficher des graphiques interactifs sur le site internet. Ce module permet de convertir des graphiques matplotlib en graphiques plotly. Cette interactivité est traduite par un affichage des données exact lorsque nous survolons un endroit précis du graphique. Ces derniers sont intégrables au site car plotly nous fournit le code html pour les utiliser. Cela permet d'avoir une attractivité supplémentaire pour les utilisateurs.



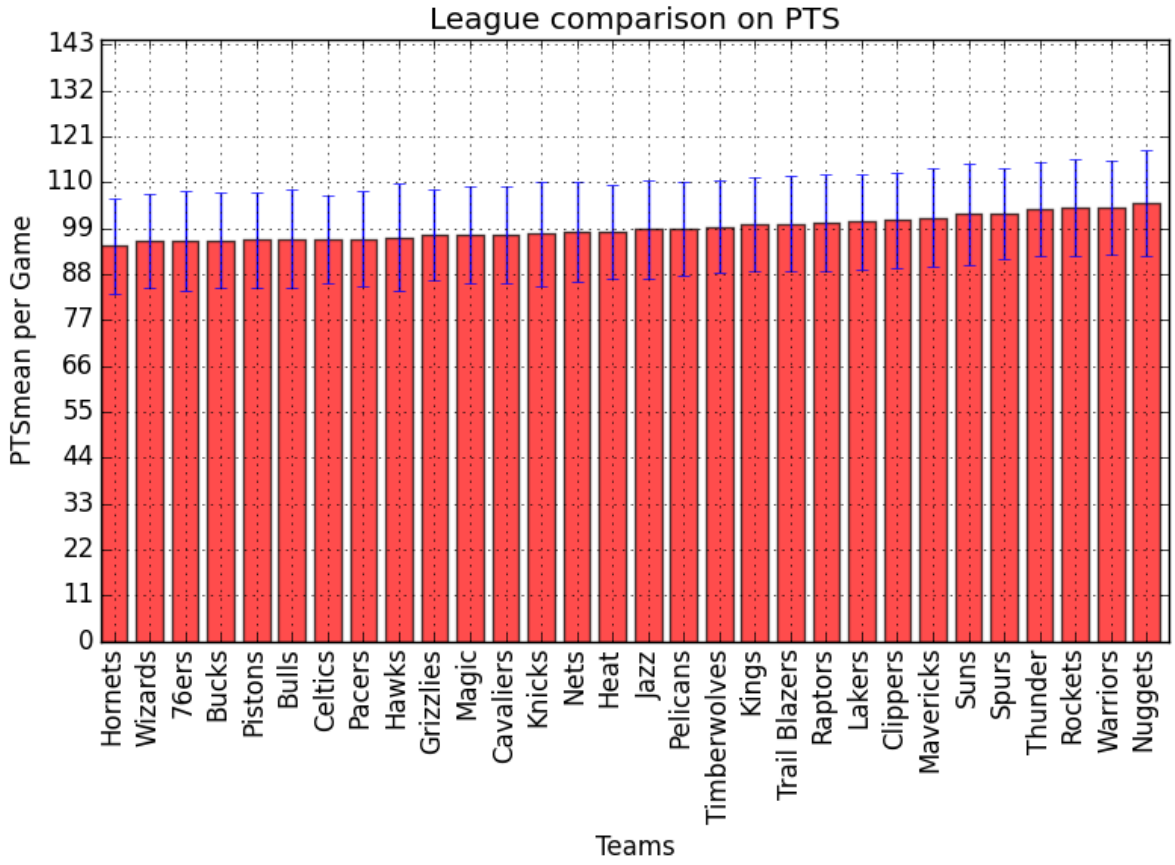
Nous avons aussi récupéré de nombreuses données avancées concernant les équipes. Ce sont des données non pas brutes, mais calculées à partir de données brutes avec des métriques très intéressantes. Elles nous permettent en effet d'obtenir d'autres variables explicatives et de réaliser de nouveaux graphiques permettant de séparer chaque équipe. Nous avons en effet deux données, l'efficacité offensive $\left(100 * \frac{Points\ Scored}{Possessions}\right)$ et l'efficacité défensive

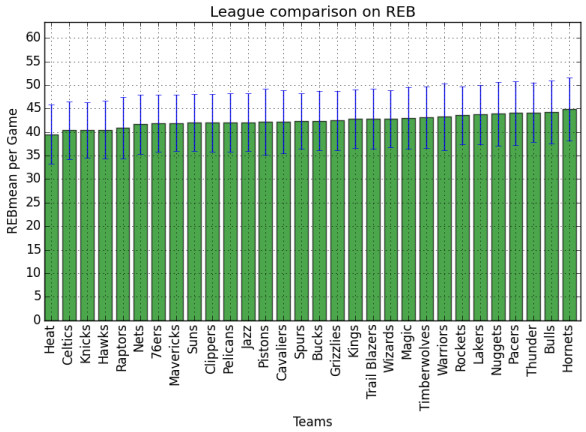
$\left(100 * \frac{Points\ Allowed}{Possessions}\right)$. Ces deux variables nous permettent de créer un graphique assez intéressant qui montre bien la capacité d'une équipe à défendre et à attaquer.

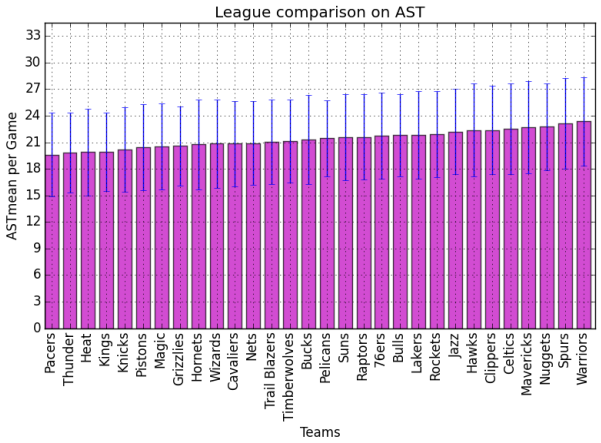
Pour continuer sur notre lancée de décrire un peu la NBA, nous avons décidé de retrouver les cinq joueurs les plus importants. Nous récupérons à partir des bases de données que nous avons créées les cinq meilleurs marqueurs de la Ligue et affichons les différentes caractéristiques de ces derniers. Nous pouvons alors suivre tout au long de la saison l'évolution de ce graphique car nous le créons chaque jour avec les nouvelles données que nous avons récupérées sur les matchs de la veille.

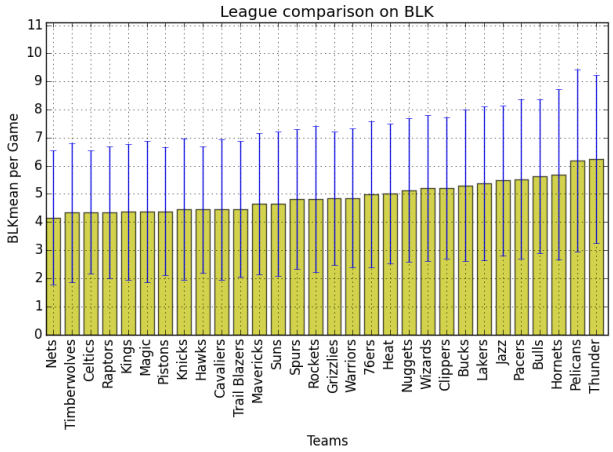


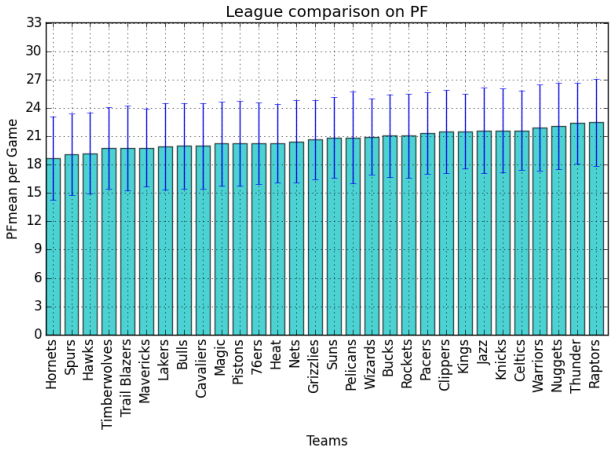
Ensuite nous voulions voir la différence de chaque équipe au niveau des cinq statistiques les plus intéressantes. Nous avons alors choisis de comparer toutes les équipes de la ligue selon leur moyenne de points par match depuis le début de la saison mais aussi leur nombre de passes, leur nombre de contres, leur nombre de rebonds, et leur nombre de fautes. Nous avons alors créé automatiquement cinq graphiques avec des échelles de valeurs qui s'adaptent automatiquement aux valeurs des moyennes de chaque caractéristique car nous ne pouvions pas mettre les points et les contres sur la même échelle puisque la moyenne des points est d'environ 100 points par match alors que le nombre de contres n'est que de 5.



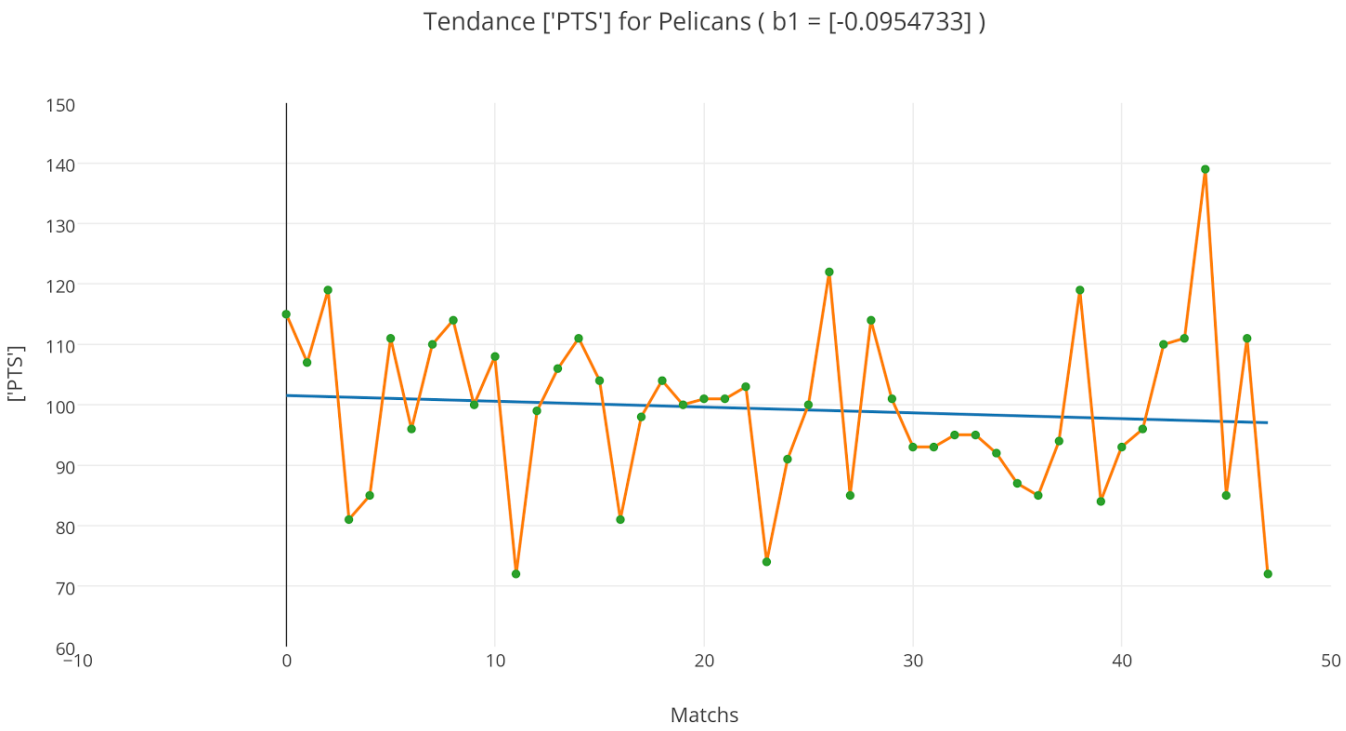








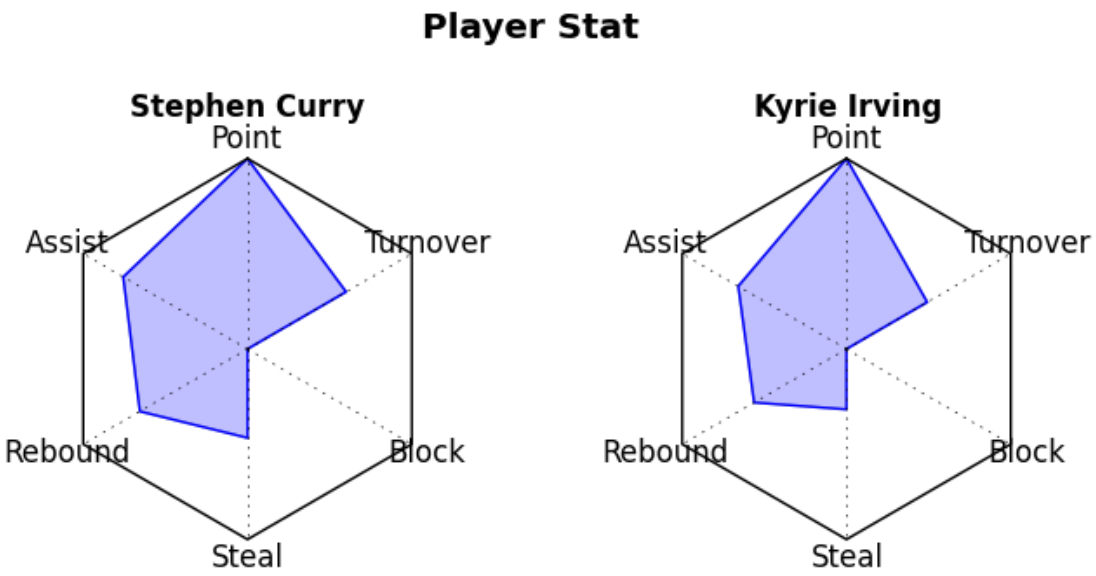
Nous voulons aussi afficher des données plus spécifiques, les statistiques les plus importantes pour l'ensemble des équipes présentes sur la saison. (Points par matchs, Rebonds, passes, fautes, contres). Nous créons alors automatiquement chaque jour un nouveau graphique prenant en compte tous les matchs de la saison et les nouveaux. Cela nous permet d'avoir des données toujours à jour.



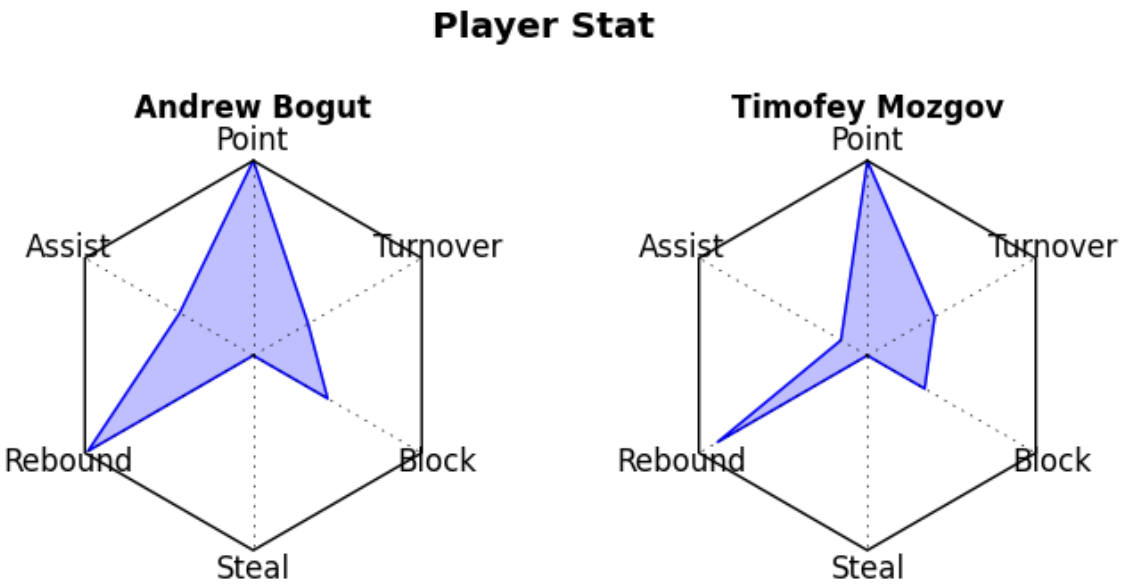
Nous avons aussi réalisé une régression linéaire afin de trouver la courbe qui détermine le mieux notre nuage de points. Elle permet de donner une tendance générale de l'équipe suivant une caractéristique spécifique. Elle nous permet de voir si en moyenne cette équipe est dans une bonne période (C'est à dire si elle marque plus de point, prends plus de rebonds, en fonction de l'avancement de la saison).

Un autre graphique intéressant est le radar chart qui permet de voir d'un coup d'oeil les forces et faiblesses d'un joueur. Néanmoins il n'est pas très précis car nous utilisons une échelle logarithmique.

Voici un exemple de radar chart entre les 2 meneurs de Cleveland et Golden State :



Comme on peut le constater on voit que ces meneurs sont offensifs et passeurs.



Ici ce sont les pivots et nous voyons bien qu'ils se spécialisent dans le rebond et block. Andrew Bogut marque autant de points qu'il prend de rebonds.

Mise en fonction du serveur

Pour agencer toutes ces technologies nous avons eu besoin d'un ordinateur central permettant à la fois l'exploration des données qui se trouvent sur le web, mais aussi l'exécution de scripts Python permettant la mise en place de notre modèle mathématiques. Nous avons pu aussi à partir de cette plate-forme lancer un serveur web à l'aide de Flask.

Cet ordinateur central est en fait une machine virtuelle, accessible uniquement depuis l'ESIEE. Elle nous permet grâce à une connexion ssh, de télécharger nos fichiers python et de les exécuter directement.

Nous avons cinq types de scripts:

- Permettant la récupération de données sur le web. Ils nous permettent aussi de mettre en forme ces données. Nous utilisons comme expliquer précédemment des DataFrame Pandas que nous stockons en local au format JSON pour optimiser la taille de nos fichiers.
- Nos scripts permettant la réalisation de notre modèle mathématique. Ils prennent pour données brutes les différentes DataFrame et grâce à différentes méthodes de classification.

- Ensuite, nous avons des scripts qui nous créent tous les graphiques, les DataFrames sous forme d'images. Cela nous permet de les récupérer très facilement et de les afficher sur notre site et sur notre application mobile.
- Nous avons aussi deux scripts permettant de récupérer les matchs de la soirée et les résultats de la veille et les poster à un url précis. Cela nous permet de récupérer toutes ces informations depuis notre application mobile, grâce à des requêtes HTTP.
- Enfin le dernier type est celui qui nous permet de mettre en place concrètement le serveur web. Il utilise Flask qui crée une application web. Ce module python utilise des templates Jinja, HTML, CSS et Javascript. Cela nous permet d'avoir une page web ou l'on peut afficher tous nos résultats.

Nous avons besoin pour avoir toutes ces données à jour à tous moments de la saison de pouvoir lancer ces tâches automatiquement. Nous utilisons pour cela Supervisor. Il nous permet de lancer nos programmes python en tâche de fond quand on le veut. Cela nous permet de scruter automatiquement les pages web, de mettre en forme nos DataFrames de créer nos modèles mathématiques à partir de données fraîchement récupérées pour qu'ils soient les plus justes possibles, de créer tous nos graphiques et tableaux que l'on affiche sur le site. Cela permet aussi que tous les matchs affichés sur la page principale du site soient les matchs de la soirée.

Malgré toutes les fonctionnalités que la machine virtuelle propose, la modification de fichiers n'est pas très facile. Nous devons alors développer notre code en local et ensuite les transférer sur le serveur de l'ESIEE. Nous utilisons pour cela Midnight Commander (https://www.midnight-commander.org/) qui est un gestionnaire de fichier permettant de faire le lien entre la machine locale et la machine virtuelle.

Nous pouvons aussi utiliser un terminal pour lancer tous nos programmes à la main. Grâce à une connexion SSH nous arrivons directement sur une machine virtuelle linux comme notre machine locale. Nous retrouvons toutes les fonctionnalités et les commandes du terminal Linux.

Affichage des données

Bien que l'analyse des données et l'interprétation de celles-ci soient les éléments-clés ainsi que le cœur de notre projet, l'affichage de ces données joue également un rôle vital. En effet, le formatage et le style sont les facteurs qui permettront de capter l'attention du lecteur. Ainsi un site web classique qui ne comporte que du HTML, même si son contenu est excellent, pourra en décourager plus d'un. Il a donc été important de s'investir dans le design du site web, en essayant de créer un site dynamique, responsive, aéré qui permettrait de maximiser nos chances d'attirer des lecteurs.

Design

Afin de créer et retoucher des images pour le site web et les différentes applications Android, nous avons eu besoin de nous familiariser avec un logiciel d'édition d'image, Adobe Photoshop CS6 et un logiciel de diagrammes et de synoptique : Microsoft Visio 2013. Le logiciel Microsoft Publisher 2010 a également été nécessaire à la réalisation du poster et de la brochure (Annexe).

Adobe Photoshop CS6

Photoshop est un outil d'édition d'images très performant qui peut également être très complexe tant les fonctionnalités sont nombreuses. C'est pour cela que nous nous sommes concentrés sur les outils qui étaient nécessaires à la réalisation des images dont nous avions besoin.

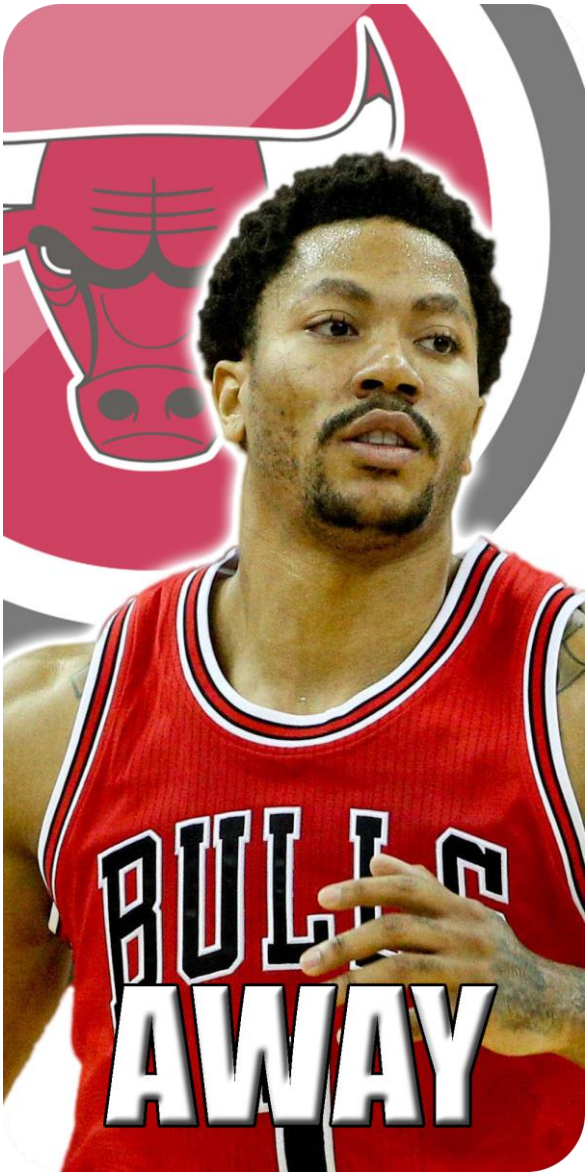
Premièrement, nous avons eu besoin de trouver des images, de la matière pour pouvoir travailler dessus. Ainsi, une des premières tâches a été de rechercher des images. La récupération des logos de chaque équipe à été faite assez aisément et assez rapidement. Cependant, un de nos objectifs était de mettre en avant deux matchs sur le site web. Nous avons donc pensé à créer des images avec les joueurs les plus importants (les plus populaires) de chaque équipe. Trouver des images pour les joueurs les plus médiatisés était une tâche facile, mais pour d'autres joueurs évoluant dans des formations moins

populaires auprès du grand public, la tâche était très ardue. En effet, la principale difficulté lors de la recherche d'image était de trouver des images aux résolutions assez grandes afin d'obtenir une qualité optimisée, et faciliter la création de Renders (Images de résolution d'au moins 1920 x 1080).

L'outil de sélection rapide a été très utile pour créer des Renders. Par exemple, on prend une image avec un fermier devant son champ de maïs, créer un Render sur cette image consiste à récupérer le fermier uniquement, et à le mettre sur un fond transparent.

➔

Grâce à cet outil, nous avons pu également mettre les logos sur fond transparent et en jouant un peu sur d'autres fonctionnalités du logiciel ainsi qu'une certaine inspiration, on parvient à créer ce genre d'images :



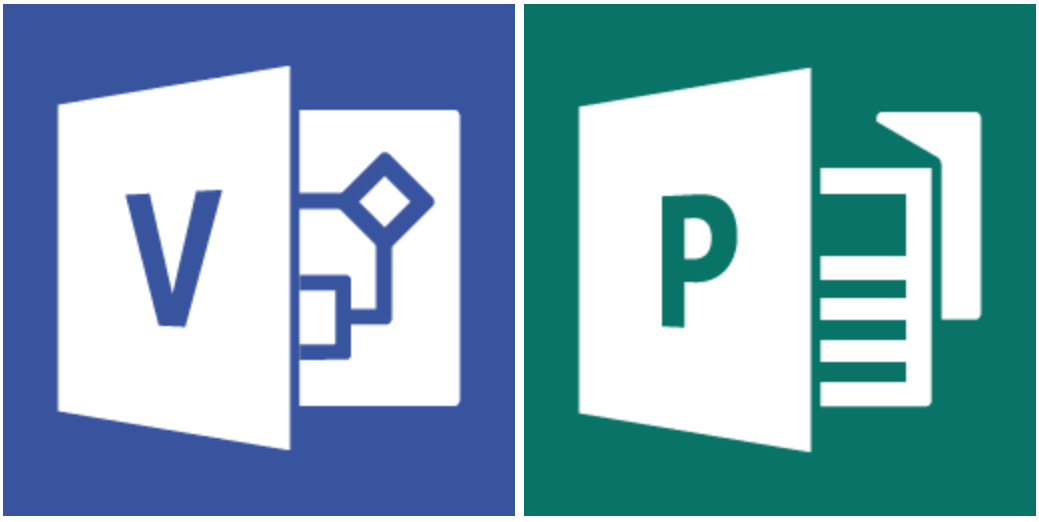
NEW BET ASSISTANT

Where NBA stats are decrypted for you

Visio-Publisher

Microsoft Visio nous a permis de créer un synoptique du projet, notamment pour le poster, et Microsoft Publisher a été utile pour la mise en page du poster et la réalisation de la brochure.

La prise en main et l'utilisation de ces deux logiciels n'a pas été difficile en soi. La partie la plus difficile a été de mettre les données en forme. En effet, il a fallu réfléchir aux points les plus importants, afin d'écrire des descriptions brèves mais précises, trouver des titres qui permettent d'accrocher le lecteur et surtout de gérer l'espace disponible.



Web

Flask

Il faut maintenant utiliser les images retouchées et les afficher sur notre site web. Pour cela, nous avons décidé d'utiliser Flask (http://flask.pocoo.org/) un environnement de programmation de site web en Python utilisant les templates Jinja, HTML et CSS. Il nous permet de créer la structure du site web, de récupérer les fichiers stockés sur le serveur et de les afficher intelligemment à l'aide de Python.

Une fois bien pris en main, l'utilisation de Flask a été plutôt simple. Les fichiers HTML sont stockés dans un dossier appelé « Templates » et les fichiers annexes (comprenant CSS, images, JavaScript) sont stockés dans un dossier « Static ».

Tout se lance à partir d'un fichier Python, dans lequel nous avons créé tous les liens de notre site web à l'aide de @app.route() qui prends en argument le lien créé. Nous avons ensuite une fonction qui va lancer le template HTML correspondant au lien créé : render_template(). Cette fonction peut prendre en argument des éléments que l'on peut ensuite utiliser dans le fichier HTML à l'aide du langage Jinja.

Flask possède une autre fonctionnalité très intéressante que nous avons utilisée : il nous permet de créer des liens dynamiquement. (voir ci-dessous)

```
@app.route('/team/<team_name>')
def team(team_name='Not found'):
return render_template("team.html", team = team_name)
```

Le terme team_name entre les < et > est remplacé par une string et nous pouvons directement renvoyer cette string dans la fonction `render_template()`. Au lieu de créer 10 liens pour afficher les statistiques de 10 équipes, il suffit d'en créer un seul et de changer les éléments de la page en fonction de la string dans team_name.

Template Jinja

Le template Jinja (http://jinja.pocoo.org/) est un langage de programmation très particulier que nous avons découvert avec Flask. Il nous permet d'utiliser les données sous forme de listes ou autre conteneur en python, directement dans le fichier HTML. Nous pouvons également travailler avec des variables, des boucles et des conditions, ce qui créé un site web très dynamique et qui s'adapte en fonction des données en temps réel fournies (voir ci-dessus)

Voici quelques particularités de Jinja :

Les variables se créent de la manière suivante : `{% set chiffre=5%}`

Lorsque l'on veut les utiliser, il faut les mettre entre accolades comme ceci : `{{chiffre}}`

Les boucles (for, while) et les conditions (if, else if, else) s'écrivent de cette manière : `{% if chiffre = 5%}`

N'ayant pas de délimiteur, il faut fermer les boucles ou conditions avec :

`{% endfor%}` ou `{%endif%}`

Certaines fonctions que nous avons utilisées sont nécessaires au développement sur l'environnement de Flask/Jinja :

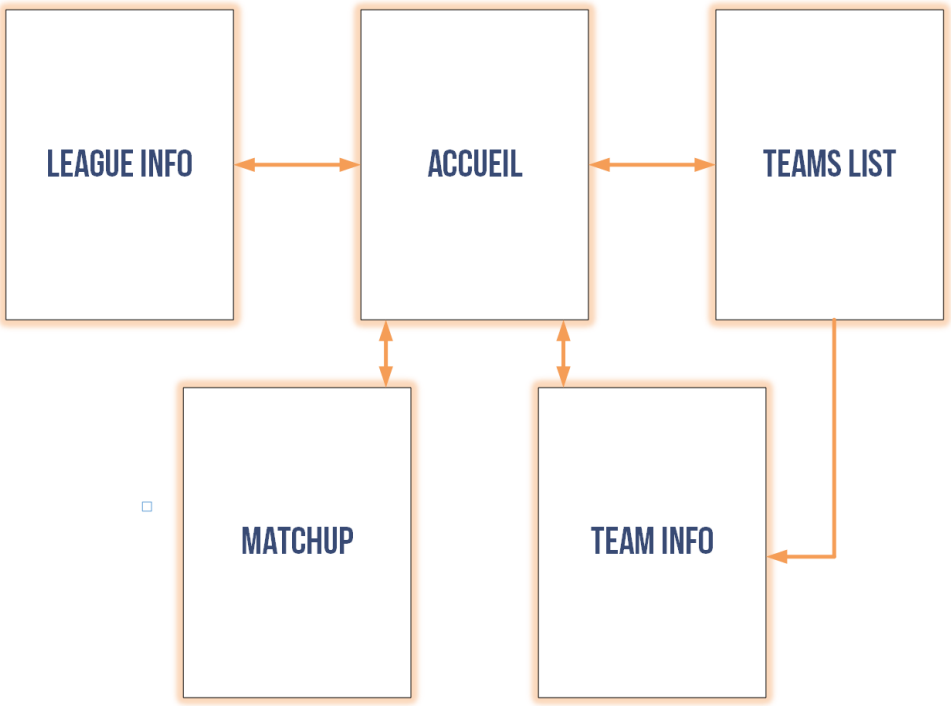
```
url_for('static', filename='images/golden-star.png')
```

`url_for()` renvoie le chemin de filename se trouvant dans le dossier static (là où tous les fichiers annexes sont stockés). Il permet également de faire un lien sur une autre page (nous l'avons notamment utilisé lors du renvoi vers la page d'accueil :`url_for('accueil') )`

HTML, CSS

Structure du site

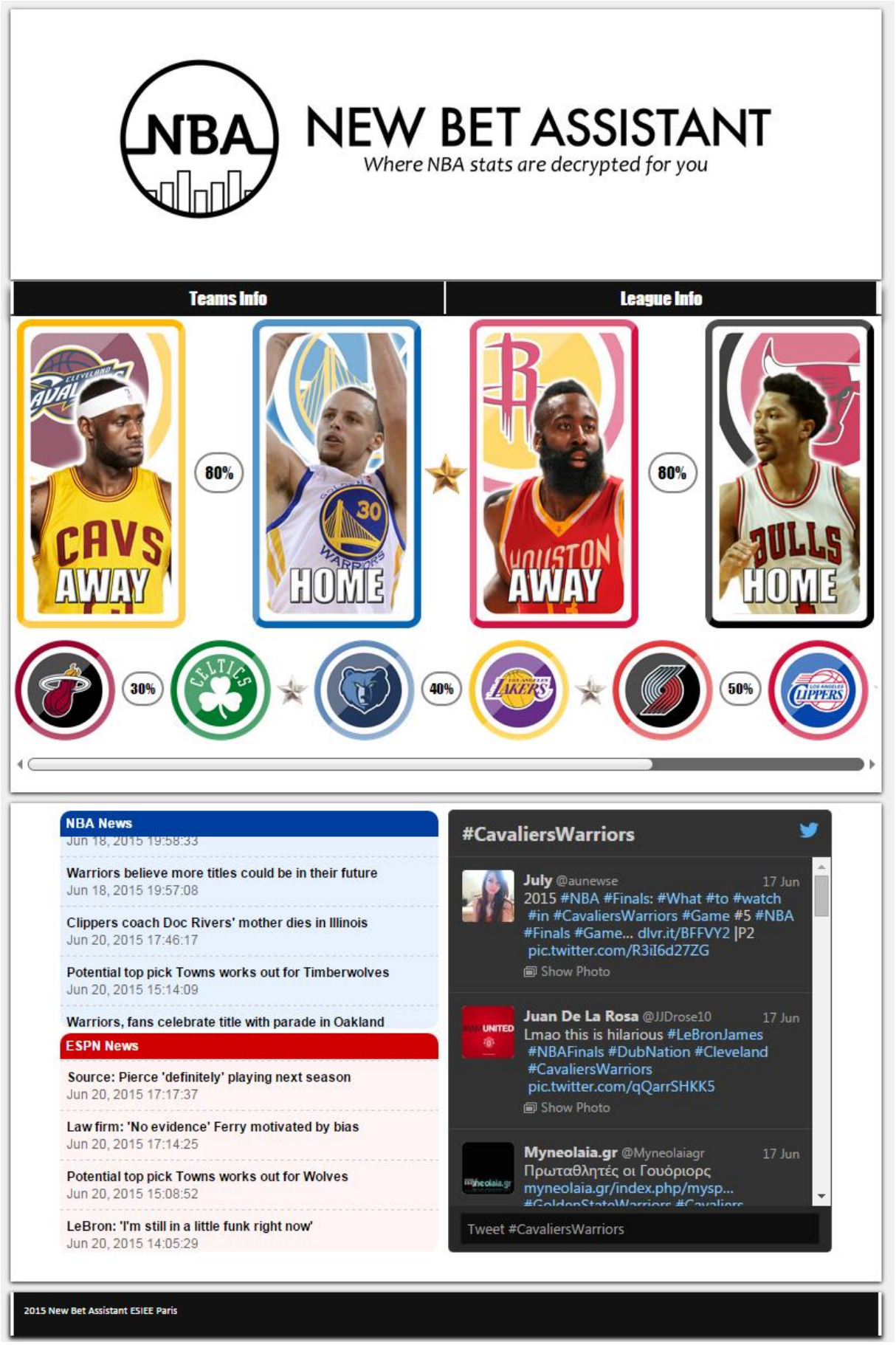
Notre site web se compose de 5 pages HTML différentes dont une page d'accueil, une page de choix d'équipe et 3 autres pages d'informations statistiques (League Info, Matchup et Team Info).



La page d'accueil est formatée différemment des autres pages, en particulier au niveau de la bannière, qui est plus grande. Nous avons ensuite un menu (« Teams List » et « League Info ») suivi des scores : Ici, Cleveland a 80% de chance de gagner contre Golden State. Juste en dessous on trouve une zone de fil info et une zone de fil twitter. Et puis enfin, le bas de page avec le nom de notre projet et l'année de création.

Pour ce qui est des autres pages, nous avons utilisé un squelette type : header prenant toute la largeur de la fenetre, body de 800px de largeur et bas de page similaire à celui de la page d'accueil.

Nous voulions que le site ait toujours la même proportion, quel que soit la taille de l'écran de l'utilisateur. C'est la raison pour laquelle le body et le footer ont une largeur de taille fixe : 800px. Tous les autres éléments sont alors placés en fonction de cette mesure.


NBA
NEW BET ASSISTANT
Where NBA stats are decrypted for you
Teams Info
League Info
CAVS
AWAY
80%
HOME
HOUSTON
AWAY
80%
BULLS
HOME
30%
40%
50%
NBA News
Warriors believe more titles could be in their future
Jun 18, 2015 19:57:08
Clippers coach Doc Rivers' mother dies in Illinois
Jun 20, 2015 17:46:17
Potential top pick Towns works out for Timberwolves
Jun 20, 2015 15:14:09
Warriors, fans celebrate title with parade in Oakland
ESPN News
Source: Pierce 'definitely' playing next season
Jun 20, 2015 17:17:37
Law firm: 'No evidence' Ferry motivated by bias
Jun 20, 2015 17:14:25
Potential top pick Towns works out for Wolves
Jun 20, 2015 15:08:52
LeBron: 'I'm still in a little funk right now'
Jun 20, 2015 14:05:29
#CavaliersWarriors
July @aunewse 17 Jun
2015 #NBA #Finals: #What #to #watch #in #CavaliersWarriors #Game #5 #NBA #Finals #Game... dlvr.it/BFFVY2 |P2 pic.twitter.com/R3iI6d27ZG
Show Photo
Juan De La Rosa @JJDrose10 17 Jun
Lmao this is hilarious #LeBronJames #NBAFinals #DubNation #Cleveland #CavaliersWarriors pic.twitter.com/qQarrSHKK5
Show Photo
Myneolaia.gr @Myneolaiagr 17 Jun
Πρωταθλητές οι Γουόριορς myneolaia.gr/index.php/mysp...
Tweet #CavaliersWarriors
2015 New Bet Assistant ESIEE Paris

Contenu

Dans les différentes pages du site, nous pouvons trouver des graphiques interactifs, des statistiques et donc des informations concernant chaque équipe individuellement, les équipes entre elles et également 2 équipes qui vont s'affronter.

Etant un site créé dans le but d'aider d'éventuels parieurs, nous n'avons affiché que les matchs se déroulant le jour même.

Outils utilisés

Le site web a été codé en HTML, CSS et Javascript. Afin de créer un site plus attractif, nous avons décidé d'intégrer des hovers, ce qui donne une dimension moderne et esthétique à notre site.

Certaines parties du site ont été codées à l'aide de Javascript, notamment la scrollbar permettant de voir les matchs différents matchs moins médiatisés de la soirée et le fil twitter.

Illustration du hover du site web

Application android

Afin de pouvoir afficher nos résultats sous une forme plus design et plus attractive, nous avons décidé d'incorporer une application sous Android dans notre projet.

L'application a été entièrement codée sous le logiciel « Android Studio » qui utilise à la fois le langage Java et XML. Ce logiciel est très bien optimisé, nous avons seulement eu besoin de quelques jours pour en comprendre son utilisation.

Une première étape a été de préparer les différentes parties selon notre besoin. Une première aurait la fonction de présenter notre projet ainsi que montrer les informations des équipes, puis une autre qui permettrait d'afficher le résultat de nos analyses, et enfin une partie qui permettrait aux utilisateurs de donner leur avis en leur permettant de choisir, selon eux, quelle équipe serait gagnante dans les prochains matchs.

Prise en main d'Android Studio

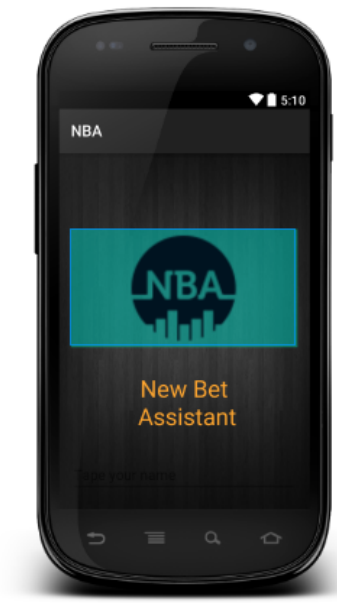
Pour chaque ''page'' de l'application, nous avons créé une activité accompagnée de son fichier XML qui permet de réaliser la mise en forme. Le fichier XML fonctionne dans le même principe que le HTML combiné à du CSS.

Pour rendre l'application plus optimale et plus design, le logiciel nous fournit des outils pour pouvoir créer des boutons « Button », des tables de navigation « TabHost » ou encore, des listes déroulantes « Spinner ». Une fois ces outils crées dans le fichier XML grâce aux balises du même nom « <TextView>, <Button>, <Spinner> », il faut lui indiquer ses caractéristiques, telles que le Width (largeur) et le Height (hauteur). Ils sont aussi personnalisables en modifiant la couleur du texte, de l'interface ou la taille de l'outil dans le fichier XML. Pour donner une fonction et une action propre à chaque outil, il faut le programmer dans la partie Java. Nous avons dû tout d'abord récupérer l'identifiant de l'outil, puis faire appel à des méthodes fournies dans les nombreuses bibliothèques que contient ce logiciel.

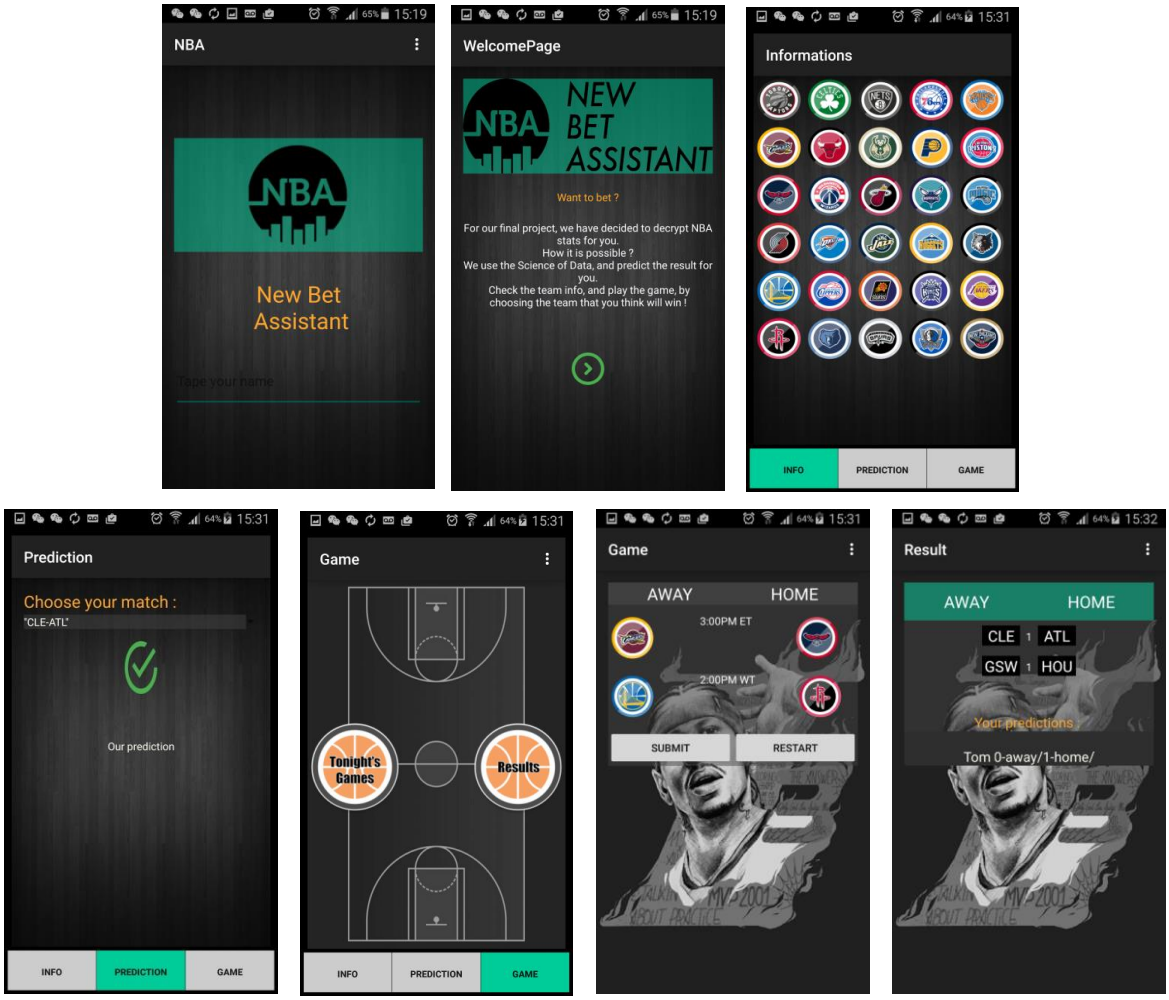
```
</FrameLayout>

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="\nNew Bet\n Assistant\n"
    android:gravity="center"
    android:textSize="30dp"
    android:id="@+id/textView"
    android:textColor="#FFA62F"
    android:layout_gravity="center_horizontal" />

<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="textPersonName"
    android:hint="Tape your name"
    android:ems="10"
    android:id="@+id/editText"
    android:layout_gravity="center_horizontal"
    android:layout_weight="0.14" />
```



L’application plus en détail



La **première activité** de l’application nous donne le nom de notre projet et notre logo et récupère le nom de l’utilisateur.

La **deuxième activité** résume notre projet et explique brièvement les principales fonctions de l’application. Sur cette page, un bouton a été créé afin

d'accéder à la prochaine activité lorsque l'on clique dessus. Pour obtenir ce résultat, nous avons eu besoin d'écrire une fonction qui récupère la classe de la prochaine activité, et de la lancer lorsque l'on fait appel à la méthode `start()`.

```
public void goResult(View v){
    Intent vI = new Intent(this, Tab.class);
    startActivity(vI);

}
```

La **troisième activité** est plus spéciale, puisqu' elle se présente sous la forme de trois tables de navigation (TabHost). Pour établir la connexion entre les tabulations et le contenu qu'il affiche, il a fallu créer chaque activité séparément et les appeler lorsque l'on clique sur la tabulation correspondant.

- La **première tabulation** donne sur l'activité qui permet d'obtenir les informations des équipes. Afin de rendre cette page design et aérée, nous avons créé pour chaque équipe un ImageButton (bouton dont le fond est une image) en utilisant les logos des équipes spécialement réalisés. Ces ImageButtons sont placés selon deux LinearLayout, un, horizontal et l'autre verticale : cela nous permet d'obtenir des boutons bien alignés, tel un tableau. Lorsqu'on clique sur un des boutons, il nous renvoie vers la page de profil de l'équipe de notre site : pour cela nous avons utilisé une WebView, un autre outil d'Android. Nous récupérons

tout d'abord le nom de l'équipe sélectionné et l'envoyons sur l'activité contenant la WebView. Puis selon, le nom reçu, on envoie un URL spécifique à la WebView.

- La **deuxième tabulation** donne sur le résultat de nos prédictions. L'utilisateur fait dérouler un Spinner qui affiche une liste des prochains matchs, puis choisit le match voulu. La prédiction s'affiche une fois son choix validé. La liste est créé dans le code Java, puisqu'il nécessitait de récupérer les données sur le serveur. (*voir récupération des données du serveur*). De plus, pour que notre application accède à internet, il a fallu lui donner les permissions dans le fichier *manifest.xml*.

- La **troisième tabulation** donne sur le « mini jeu », qui permet aux utilisateurs d'exprimer leur avis sur les gagnants des futurs matchs. Il se décompose en deux parties :

La *première* permet aux utilisateurs de choisir l'équipe gagnant selon lui. Pour cela, en fonction des prochains matchs que l'on récupère sur le serveur, le programme crée, pour chaque match, deux ImageButtons correspondant aux deux équipes récupérées, un pour l'équipe à domicile et l'autre pour l'équipe extérieur.

```
for (int i = 0; i < vMatchupArray.size();i++){

    Log.d("Matchup : " + i, vMatchupArray.get(i));
    final ImageButton vButtonExt = new ImageButton(this.getApplicationContext());
    LinearLayout.LayoutParams paramsLinear0 = new LinearLayout.LayoutParams(LinearLayout.LayoutParams.FILL_PARENT, LinearLayout.LayoutParams.WRAP_CONTENT);
    vButtonExt.setLayoutParams(paramsLinear0);
    vButtonExt.setBackgroundColor(Color.BLACK);
```

Nous avons aussi récupéré la date et l'horaire du match. Une fois sa sélection effectuée, l'utilisateur a le choix entre valider son choix, en sélectionnant le bouton « submit » ou de le remettre à zéro et de recommencer, en sélectionnant le bouton « restart ». Comme nous voulons par la suite exploiter les avis émis par les joueurs, nous avons rajouté un bout de code qui enregistre son choix dans une base de données et dans un fichier créé dans la carte SD du smartphone. Bien entendu, nous prenons seulement en compte la sélection validée. A chaque restart, la sélection est effacée et mise à jour.

L'autre partie permet aux utilisateurs de vérifier si sa sélection était correcte ou non, en regardant le résultat des matchs joués.

Pour l'instant ces données ne sont pas exploitées, car nous n'avons pas encore mis en place un système permettant de récupérer directement les pronostics des joueurs. Nous voulons par la suite utiliser ces données dans notre modèle pour affiner encore plus notre prédiction. En effet, on peut penser qu'inconsciemment, les gens parieront sur l'équipe la plus apte à gagner. Cela peut nous permettre d'avoir des données supplémentaires si le nombre de vote pour une équipe est significativement supérieur à celui de l'autre équipe.

Récupération des données du serveur

Nous avons besoin pour afficher les matchs de la soirée que notre application mobile soit d'une certaine façon reliée au serveur pour qu'elle puisse récupérer

les informations dont elle a besoin pour afficher les résultats des prédictions, les résultats des matchs de la veille.

Nous utilisons pour cela Volley, une librairie disponible sur Android qui nous permet de faire des requêtes des objets JSON que nous avons posté sur notre site web.

```
String url_result = "http://courivar.pythonanywhere.com/result";

JsonObjectRequest vJSONObjRequest2 = new JsonObjectRequest(Request.Method.GET, url_result, null, (response) → {
        Log.d("RESPONSE","You've got a Response");
        // TODO Auto-generated method stub
        aResultArray = new ArrayList<String>();
        Log.e("FULL RESPONSE :" , response.toString());
        JSONArray array = null;
        try {
            array = response.getJSONArray("result");
            Log.e("COUCOU :", array.toString());
        } catch (JSONException e) {
            e.printStackTrace();
        }

        if (array != null) {
            int len = array.length();
            for (int i=0;i<len;i++){
                try {
                    aResultArray.add(array.get(i).toString());
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
```

Lorsque nous récupérons l'objet JSON il nous suffit maintenant de le transformer en ArrayList JAVA et de parcourir ces ArrayList pour pouvoir récupérer les données dont nous avons besoin.

Nous avons vu que le format JSON est un ensemble de clé-valeur qui nous permet d'associer à chaque nom d''équipe une heure pour les matchs de la soirée et le résultat de notre prédiction mais aussi le score pour l'affichage des résultats de la veille.

Conclusion