Treball de Fi de Màster

Màster Universitari en Enginyeria Industrial (MUEI)

Aplicació de tècniques de 'Data Science' a la predicció de resultats esportius

MEMÒRIA

Autor: Roger Alemany Sadurní
Director: Lluís Talavera Méndez

Convocatòria: Gener 2021

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Resum

En aquest projecte desenvolupa la metodologia CRISP-DM, una metodologia extensament aplicada en l'àmbit de la ciència de dades i mineria de dades, amb l'objectiu de predir resultats esportius de la NBA.

Primer de tot es defineix l'objectiu principal que és determinar amb quin grau de precisió es poden arribar a predir els resultats finals de qualsevol partit de la NBA.

Per obtenir resposta a aquesta pregunta s'han recollit una gran quantitat de dades i estadístiques, de partits i de jugadors. Des de l'edat dels jugadors, la ratxa de victòries en els últims partits, els dies de descans entre partits, lesions dels jugadors, etc. S'han recollit les dades corresponents a les últimes tres temporades disputades

A partir d'aquestes dades recollides s'han processat i preparat. Gràcies a aquesta preparació s'han tret 20 variables diferents que han servit per fer la posterior anàlisi i predicció de resultats.

Seguidament s'han aplicat diversos algoritmes de classificació com Logistic Regression i SVM i també diferents models de "testeig" de les dades com Time Series Cross Validation o Random Cross Validation. Tot amb l'objectiu de aconseguir la màxima eficàcia possible en la predicció de resultats.

Finalment com a conclusions s'ha aconseguit una bona capacitat de previsió de resultats, al voltant del 66%, és a dir dos de cada tres partits predits de forma correcte.

Aquest és un treball de fi de màster, relacionat amb el món de la ciència de dades, i per tant engloba també una part important de estadística, programació i en menor mesura de matemàtiques.

.

Sumari

1. Prefaci

Aquest projecte s'engloba dins el camp de la anomenada ciència de dades o més coneguda com 'Data Science'. Aquest àmbit d'estudi té aplicacions molt diverses, des de la biomedicina o la genètica fins a la compra venta d'accions o la publicitat i el màrqueting. També té àmplies aplicacions en l'esport professional y concretament en el món del bàsquet. En l'anàlisi de dades relacionades amb el bàsquet es on es centrarà bàsicament aquest projecte.

1.1. Origen del projecte

Aquest projecte neix amb l'objectiu de respondre diverses preguntes que ja fa anys que em voltaven pel cap. Des de que vaig sentir a parlar per primera vegada de Big Data, Data Science i Intel·ligència Artificial em vaig preguntar quines aplicacions podria tenir en el món de l'esport i concretament del bàsquet.

1.2. Motivació

Des de petit m'encantava consultar les estadístiques dels partits i extreure'n conclusions, ja fos dels partits que jugava com a jugador amateur com també dels partits d'equips professionals de l'ACB i de la NBA. Es per això que quan el Data Science va començar a aparèixer amb força, em vaig preguntar, si això podria respondre diverses preguntes relacionades amb el bàsquet. Com per exemple, si podria ajudar a predir el resultat d'un partit de bàsquet, quins jugadors són més importants en cada equip, com es pot utilitzar la estadística per analitzar i millorar l'estratègia dels equips alhora de jugar, etc.

Òbviament el ventall de preguntes són infinites, però com que cal acotar molt el problema per tal que sigui realitzable s'han acotat les preguntes a les dades estadístiques de la NBA, ja que a part de ésser la millor lliga de bàsquet del món, és una lliga amb una gran quantitat de estadístiques i dades fàcils de aconseguir.

1.3. Requeriments previs

Per tal de poder realitzar aquest projecte vaig realitzar diversos cursos online sobre Data Science i Machine Learning per tal de ampliar els coneixements adquirits en programació i estadística durant el Grau en enginyeria en Tecnologies Industrials i el Màster en Enginyeria Industrials.

Van ser un total de 9 cursos (total de 180 hores) que tenien el següent contingut:

- Què es la ciència de dades?
- Eines per la ciència de dades.
- Metodologia de la ciència de dades.
- Python per ciència de dades i intel·ligència artificial.
- Bases de dades i SQL per ciència de dades.
- Anàlisis de dades amb Python.
- Visualització de dades amb Python.
- Machine Learning amb Python.
- Projecte Final de ciència de dades aplicant els coneixements adquirits prèviament.

Un cop realitzats aquests cursos vaig tenir una visió més global sobre el món de Data Science i vaig aprendre, a utilitzar diverses metodologies i models que s'utilitzaran en aquest projecte.

1.4. Objectius del projecte

El ventall de maneres de enfocar el projecte és infinit, però en aquest tipus de projectes per tal de realitzar un bon treball, cal acotar molt bé el problema per tal que sigui realitzable i rellevant, dins les limitacions d'un Treball de Fi de Màster.

L'objectiu principal del treball és construir i avaluar diversos models de predicció de resultats de partits de la NBA a partir d'un conjunt d'estadístiques d'equips i jugadors recollides en partits previs al partit que volem predir-ne el resultat. És a dir a partir de resultats que s'han produït anteriorment durant la temporada, el nostre model ha de ser capaç de predir amb la màxima precisió possible els resultats dels propers partits.

És a dir el que volem obtenir és un conjunt de dades que s'actualitzin diàriament i que per tant cada dia tinguem presents les dades dels últims partits disputats. El partit disputat en la data "t" tindrem en compte totes les dades que hem recollit dels partits ja disputats dels partits "t-1" a "t-n", on "n" serà un valor de partits arbitrari que caldrà definir al llarg del projecte.

Aquesta predicció de resultats seria molt útil per els entrenadors i "general managers" dels equips de la lliga ja que els permetria planificar millor, la temporada per tal de planificar descansos de jugadors, en funció dels partits més probables de guanyar o tenir una previsió de com pot anar la temporada en la qüestió dels resultats. Per tant aquesta seria la principal utilitat d'aquest anàlisis de resultats i la seva predicció.

La principal pregunta que busco respondre per tal de intentar obtenir un model acurat i que doni els millors resultats possibles és la següent:

Es pot arribar a predir els resultats dels partits de la NBA? I si és així amb quina precisió es pot fer?

Des del punt de vista personal, l'objectiu principal és aprendre més en els camps del 'Data Science', programació i estadística, mentre s'exploren noves metodologies i models a l'hora de desenvolupar el projecte.

1.5. Abast del projecte

Per tal de respondre a les preguntes caldrà recol·lectar una gran quantitat de dades de tot tipus relacionades amb la NBA. Com més dades de diferents, aspectes del joc siguem capaços de recol·lectar millors prediccions es podran realitzar.

De totes maneres al disposar d'un escenari de temps limitat per la realització d'aquest projecte. S'aplicarà a petita escala però de forma rigorosa la metodologia CRISP-DM (metodologia àmpliament utilitzada en projectes de 'Data Science') i s'implementarà utilitzant les llibreries Pandas i SciKit-learn de Python (llibreries molt orientades a la ciència de dades) entres d'altres. Per tant el llenguatge amb el que es programarà en aquest projecte serà bàsicament Python.

Per acotar mínimament l'abast general del projecte cal definir quines dades es recolliran inicialment tot i que durant la realització del projecte se'n poden afegir en cas que siguin rellevants o necessàries.

La llista de dades que es volen aconseguir de forma inicial són les següents:

- Resultats i data de cada partit de la temporada a analitzar de la NBA.
- Estadístiques de cada jugador per cada partit disputat.
- Estadístiques avançades de cada equip de tota la temporada.
- Estadístiques avançades de cada jugador de tota la temporada.
- Desplaçaments i viatges dels equips.
- Prediccions de victòries de cada equip a l'inici de la temporada

De forma inicial es recolliran aquestes dades per el conjunt de les últimes 3 temporades disputades (temporades 2017-2018, 2018-2019 i 2019-2020).

El projecte també té altres limitacions, ja que l'objectiu de màxims seria crear un sistema automàtic de predicció de resultats, en què es recollissin diàriament les dades necessàries. Per qüestions de temps és inviable automatitzar-lo completament.

El sistema que volem crear tampoc treballarà amb dades a temps real si no que per veure si funciona correctament utilitzarem les dades de les 3 últimes temporades disputades i d'aquesta manera, tenint totes les dades per avançat podrem valorar la precisió en la

predicció de resultats.

Per tant es vol deixar clar que tot i que l'objectiu de màxims seria crear un sistema operatiu que realitzés automàticament les prediccions, el que es farà en aquest treball és una simulació o també es podria anomenar prova de concepte, per veure si els models de predicció sorgits d'aquesta simulació, donen bon resultat i per tant si val la pena en un futur automatitzar-ho i seguir investigant.

2. Introducció

2.1. Conceptes i definicions bàsiques relacionades amb la Data Science

Des de fa ja uns anys, s'ha anat popularitzant nou vocabulari com per exemple "Data Science", "Big Data", "Machine Learning", intel·ligència artificial, etc. Entre tantes paraules es fàcil de confondre conceptes que ens poden provocar errors greus de comprensió, especialment quan hi ha un desconeixement generalitzat que fa que s'usin indistintament per la majoria de la gent.

La intenció d'aquest apartat del treball és aclarir tots aquests conceptes. D'aquesta manera definirem la 'Data Science' entre d'altres conceptes importants.

Cal afegir però que aquesta terminologia està en constant evolució. Cada cop que apareixen noves tecnologies relacionades amb el món de les dades, cal actualitzar i revisar el vocabulari. També és habitual que els experts en la matèria no estiguin totalment d'acord ja que a cada concepte se li poden donar diferents enfocaments.

La terminologia relacionada amb la ciència de dades és una espècie de puzle de peces difícils de encaixar. Que s'intentarà definir a continuació.

"Data Science", com el seu nom indica, és la ciència que estudia les dades. Per tant queda clar que és un terme molt genèric. És a dir 'Data Science' engloba tots els conceptes relacionats amb l'estudi de les dades. És per això que cal definir les seves subdisciplines i conceptes més importants.

Tots els conceptes explicats a partir d'ara pertanyeran a el camp de "Data Science". Primer de tot començarem amb un dels conceptes més importants com és la mineria de dades o més coneguda com 'Data Mining'.

2.1.1. Què és el Data Mining?

"Data Mining" (o mineria de dades) ocupa una part molt gran al món "Data Science": engloba tot el relacionat amb treure informació útil i de valor en dades on en principi no sembla fàcil treure'n informació o conclusions. Normalment, es parteix de dades no estructurades i acaba en unes conclusions o informació de valor, havent passat per diferents tècniques i algoritmes.

La mineria de dades pot ser capaç de respondre preguntes d'àmbits molt diferents, com poden ser les següents.

• És el model de cotxe dels meus clients influent en la seva fidelitat amb la marca?

• Amb quina seguretat puc predir si el nombre de vendes aquest any arribarà al seu objectiu?

• En base a quines característiques podria classificar als meus clients per fer-los arribar un tipus d'anuncis o altres (Màrqueting digital)?

• Com afectaria a les meves vendes que invertís en una ONG i sortís publicat en les xarxes socials?

Per descomptat, es poden estendre a el camp que es vulgui (ex., Medicina, astronomia, psicologia, publicitat ...) i les preguntes podrien ser les mateixes que se li farien a un guru expert en el camp.

2.1.2. Machine Learning

Podríem definir el Machine Learning com el motor que permet automatitzar el procés de la predicció de dades. El Machine Learning té molt en comú amb l'estadística clàssica, ja que utilitza mostres de dades per inferir i fer generalitzacions que en permetin extreure conclusions. Però l'estadística es centra més en el camp descriptiu (malgrat també es pugui utilitzar amb finalitats predictives), en canvi el Machine Learning només s'utilitza com un pas intermedi amb l'objectiu de ser capaç de fer prediccions.

2.1.3. Deep Learning

L'anomenat Deep Learning és un procés, com per exemple el Data Mining, que utilitza una arquitectura de xarxes neuronals, que bàsicament són un tipus particular de algoritmes de Machine Learning.

En els últims anys aquest procés s'ha popularitzat molt, però aquest èxit ha portat a una sobrevaloració excessiva per part de molta gent no experta en la matèria.

El Deep Learning ha demostrat ser una eina molt bona per solucionar problemes de classificació, processament de llenguatge i visió artificial entre d'altres. No obstant avui en dia molta gent creu que aquesta tecnologia té la capacitat de pensar com un cervell humà, la

qual cosa és completament falsa. Per tant el Deep Learning és una eina molt bona però no desplaçarà la resta de algoritmes de Machine Learning ni tècniques de Data Science.

2.1.4. Què és la intel·ligència artificial?

La intel·ligència artificial busca simular i reproduir el raonament lògic tal com ho fem els humans. No només per trobar patrons, classificar i predir esdeveniments a partir d'una informació, sinó també per a generar dades en concordança i lògica a una experiència passada. Alguns exemples d'aplicació són:

- Creació d'un assistent virtual intel·ligent d'atenció a client.
- Generació de logos (o idees de logos) de forma automàtica.
- Càlcul de rutes òptimes d'uns camions de repartiment.

2.1.5. Què és el 'Big Data'?

Big Data és la disciplina que treballa amb grans quantitats de dades. És a dir, el Big Data està present en els projectes potents de Data Science. També tenim lògicament Small Data i Medium Data, tot i que no sonen de forma tan potent, no s'utilitzen tant.

Per a projectes de Small Data, amb utilitzar un Excel o una base de dades petita en un ordinador portàtil és suficient. A més, és molt fàcil de manejar, perquè tota la informació cap en la memòria RAM de l'ordinador i es pot "veure" tota alhora.

El Medium Data treballa amb quantitats més grans de dades, on un Excel no seria suficient per tenir-les totes, encara que sí podríem emmagatzemar-los en una base de dades d'un ordinador. Caldria utilitzar certes tècniques per processar i analitzar la informació sense demanar-la tota de cop, ja que seria massa gran per carregar-lo en memòria.

Els projectes de Big Data són els que necessiten fins i tot diversos ordinadors compartint informació perquè el processament i l'emmagatzematge sigui possible. Això suposa utilitzar tècniques de sincronització i cooperació entre màquines.

2.1.6. Relació entre les diferents terminologies

Per acabar amb la definició d'aquestes terminologies, es volen deixar clars alguns dels

errors de concepte típics que es fan per els que no són experts en la matèria:

El Deep Learning i la inteligencia artificial no són sinònims

La diferencia entre Machine Learning i Data Mining consisteix en, que el Data Mining és un procés, durant el qual s'utilitzen algoritmes de Machine Learning com a eines per extreure patrons útils amagats entre el conjunt de dades.

El Data Mining és un procés, per tant té sentit veure la Data Science com un super conjunt del Data Mining.

El Data Mining pot utilitzar la intel·ligència artificial en les seves fases d'aprenentatge, ja que alguns dels algoritmes de Machine Learning sí que formen part de la intel·ligència artificial.

Amb l'objectiu de resumir s'inclou el següent gràfic [1] que inclou totes les terminologies explicades. Per descomptat el gràfic que apareix a continuació no és una foto exacte de la relació entre termes, ja que aquesta evoluciona constantment. Però permet resumir els conceptes explicats anteriorment.

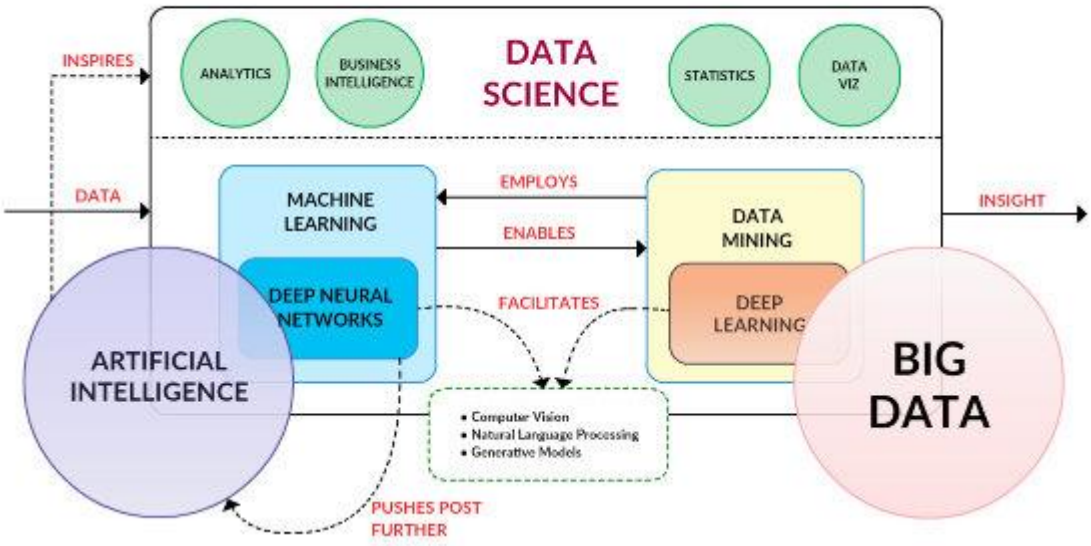


Figura 1. Figura que intenta aclarir la relació entre les diferents terminologies en l'àmbit de Data Science.

2.1.7. Quins són els professionals que treballen en aquestes disciplines?

Normalment és el Data Engineer o enginyer de dades el que s'encarrega de configurar i dissenyar les bases de dades de forma òptima. De fet, l'enginyer de dades està darrere de tota la part informàtica d'un projecte de Data Science; des de la recollida i emmagatzematge de dades, programació de el codi que maneja les dades, fins a realitzar les visualitzacions pertinents.

Un perfil semblant, i que se sol confondre amb l'enginyer de dades, és el científic de dades. Més conegut com Data Scientist.

Tant un enginyer de dades com un científic de dades han de tenir coneixements d'informàtica i estadística i no és estrany que de tant en tant un faci la feina de l'altre. Un Data Scientist té un perfil més matemàtic i menys informàtic que el Data Engineer i el seu treball és dissenyar els algoritmes a utilitzar. També, sol tenir més coneixement sobre de quina manera presentar les dades a el públic perquè siguin útils i fàcilment interpretables.

Perquè s'entengui millor: es podria dir que un Data Scientist i un Data Engineer són comparables a un metge i un infermer: el Data Scientist seria el metge i el Data Engineer l'infermer. O, si es prefereix, es poden comparar amb un arquitecte i un enginyer d'edificació, on el Data Scientist seria l'arquitecte i l'enginyer de dades, l'enginyer d'edificació.

Per tant en aquest projecte es realitzaran les dos feines principals en els projectes de “Data Science”, ja que hi haurà una part corresponent a la recol·lecta de dades i processat que realitzaria l’enginyer de dades i l’altre part seria la corresponent al desenvolupament del model i dels algoritmes que correspondria al científic de dades.

2.2. La metodologia CRISP-DM

La metodologia Cross-Industry Standard Process for Data Mining (CRISP-DM), és un model de procés estàndard obert que descriu enfocaments comuns utilitzats per experts en mineria de dades. És el model d'anàlisi més utilitzat. CRISP-DM divideix el procés de mineria de dades en sis grans fases:

- Business Understanding
- Data Understanding
- Data Preparation
- Modeling
- Evaluation
- Deployment

La seqüència de les fases no és estricta i es mou entre diverses fases, ja que sempre és necessària. Les fletxes del diagrama de procés que apareix a continuació (*Figura 2*) indiquen les dependències més importants i freqüents entre fases. El cercle exterior del diagrama simbolitza la naturalesa cíclica de la pròpia mineria de dades. Un procés de mineria de dades continua després desplegar de una solució. Les lliçons apreses durant el procés poden desencadenar noves qüestions empresarials, sovint més enfocades, i els processos posteriors de mineria de dades es beneficiaran de les experiències anteriors.

Aquesta serà la metodologia utilitzada en cada un dels passos del treball, d'aquesta manera no ens saltarem cap dels passos essencials en un projecte d'aquest tipus.

Figura 2. Representació en un diagrama de la metodologia CRISP-DM.

2.2.1. Bussines i Data Understanding

La part de Business i Data Understanding consisteix bàsicament en plantejar quines preguntes es volen respondre en el projecte, i quines dades es necessitaran per respondre de la forma més precisa la pregunta inicial del projecte. No sempre es disposa de la informació immediatament disponible en una base de dades (o full de càlcul) i la seva obtenció és una part molt important del procés. La recollida de la informació pot realitzar-se mitjançant col·locació de sensors, 'web scraping' (el 'web scraping' es un tècnica que serveix per extreure informació de pàgines web de forma automatitzada). API's (una API és una interfície de programació de aplicacions que estableix com un mòdul de un software es comunica o interactua amb una altre mòdul per complir múltiples funcions), formularis, entre moltes d'altres.

2.2.2. Data Preparation

Un cop es té la informació que es considera necessària per resoldre el nostre model, se li ha d'aplicar un preprocessat. És a dir, sense perdre informació de valor, donar-i un tomb a la informació que ja tenim, per preparar-la per la següent fase. L'objectiu és representar la informació en un format que aconsegueixi reduir el cost de còmput i optimitzar els resultats dels algoritmes.

El preprocessat és una fase crítica, ja que condiciona a la resta del procés i pot causar la diferència entre l'èxit i el fracàs de el model. Per això, aquesta fase i les següents són tan dependents entre si. És molt habitual iterar entre elles fins a trobar la combinació que millor s'ajusti als resultats que es busquen.

2.2.3. Entrenament o modeling

Entrenar el model significa alimentar algoritmes de "Machine Learning" amb les dades ja treballades.

Els algoritmes de "Machine Learning" (o aprenentatge automàtic) són capaços de predir i classificar informació nova, arran d'haver estat entrenats amb informació el passat.

Els algoritmes de "Machine Learning" es poden classificar com d'aprenentatge supervisat o com d'aprenentatge no supervisat. La diferència és que els d'aprenentatge supervisat aprenen a trobar respostes basant-se en casos passats amb les seves respostes ja conegudes, mentre que els d'aprenentatge no supervisat tracten d'aprendre sense tenir les respostes.

2.2.4. Evaluation

En general, no hi ha un algoritme millor que un altre, alguns rendeixen millor en uns casos i altres, en altres casos. I la manera de trobar la millor solució és provant-un a un, amb diferents configuracions, fins a trobar el millor per al nostre cas.

Es segueix una metodologia experimental prova-error perquè, a priori, és molt difícil endevinar quin tipus d'algorisme i configuració pot tenir millors resultats. Només els millors professionals en "Machine Learning" poden tenir una vaga idea de quin tipus d'algoritmes poden funcionar millor amb un set de dades determinat. Però tot i així haurien de provar diverses opcions i testejar.

Hi ha diferents tècniques per validar els resultats d'un algoritme de Machine Learning, és a dir, per mesurar la bondat del classificador. Per als d'aprenentatge supervisat, entre altres tècniques, es reserva una part de les dades per realitzar l'entrenament i la resta s'utilitza per validar o testejar el model. Per als d'aprenentatge no supervisat és més complicat, però també existeixen mètodes que donen una estimació de la bondat de l'algorisme.

2.2.5. Visualitzation i Deployment

Finalment, per poder comunicar-ho a tercers, s'ha de representar d'alguna manera el coneixement obtingut. S'ha de trobar la manera més neta i intuïtiva de visualitzar els

resultats i, amb ajuda de diferents programaris, crear visualitzacions atractives i fàcils de assimilar. Un cop presentats els resultats, sinó han estat del tot satisfactoris, es torna enrere en les fases, per corregir i millorar el model.

2.3. Perquè utilitzar Python en Data Science?

El llenguatge de programació Python és el més estès actualment en el món de la ciència de dades tot i que no és l'únic que s'utilitza, per exemple és molt utilitzat també el llenguatge R.

Python és molt útil per diversos motius que es detallen a continuació:

- Simplicitat: Python és un dels llenguatges més simples i par tant més fàcils per iniciar-se al món de la programació, malgrat això Python permet elaborar codis de una gran complexitat.
- Python es gratis i un llenguatge Open Source.
- Té una gran flexibilitat ja que és molt útil per solucionar problemes molt diversos. Desde programació de Apps mòbils fins a càlcul matemàtic avançat entre d'altres.
- Degut a la popularitat del llenguatge, Python disposa de milers de llibreries diferents, que permeten solucionar de forma ràpida i eficaç qualsevol problema relacionat amb qualsevol àmbit de la programació. Python disposa de múltiples llibreries orientades a Data Science.
- Gran comunitat: Python disposa de milions de usuaris a tot el món, que comparteixen les seves experiències i en cas de tenir problemes hi ha moltes webs dedicades a compartir informació per ajudar-se mútuament.
- Diverses feines i creixement: Python es un llenguatge molt potent que obre múltiples oportunitats per tot tipus de feines relacionades amb la programació, entre elles les relacionades amb la ciència de les dades.

Python disposa de una àmplia gamma de eines de programació que ens seran molt útils per poder realitzar aquest projecte.

A continuació s'expliquen alguns conceptes molt bàsics relacionats amb la programació en Python i Data Science que s'utilitzaran en aquest treball.

Primer de tot definirem llibreries de Python utilitzades en l'àmbit de Data Science y Machine Learning.

Les llibreries són:

- **Numpy:** És un package de Python fonamental utilitzat en la programació científica. Conté entre d'altres, un potent objecte (N-dimensional array), funcions sofisticades, eines per integrar en codis de C/C++ i Fortran, funcions d'àlgebra lineal, transformades de Fourier i funcions de creació de numero aleatoris. A part del seu ús per programació científica, NumPy es pot utilitzar com una eina eficient per guardar qualsevol tipus de dades. Això fa que NumPy pugui treballar de manera ràpida i eficient amb una gran varietat de bases de dades. Per aquest projecte serà la funció principal que li donarem a NumPy ja que ens permetrà guardar les dades de forma eficient i també ens permetrà poder manipular aquestes dades de forma fàcil. Numpy treballa amb arrays. Una array és un objecte similar a una llista en Python però amb algunes particularitats que la fan molt útil a l'hora de realitzar càlculs de manera més eficient, això permet estalviar memòria a l'hora de processar dades i temps de càlcul.

- **Pandas**: Es tracta d'una llibreria de Python destinada bàsicament a l'anàlisi de dades, ja que proporciona unes estructures de dades flexibles i que permeten treballar amb les dades de forma molt eficient. Ofereix les següents estructures de dades:

    - **Series**: Són arrays unidimensionals amb indexació (arrays amb índex) i són similars als diccionaris, de fet les series poden generar-se a través de diccionaris o de llistes.

- **DataFrame**: Són estructures de dades similars a les taules de bases de dades relacionals, com s'utilitzen en el extensament conegut llenguatge SQL.

	Date	Visitor	PTSV	Home	PTSH	Overtime	Attend.
0	Tue, Oct 22, 2019	New Orleans Pelicans	122	Toronto Raptors	130	1	20,787
1	Tue, Oct 22, 2019	Los Angeles Lakers	102	Los Angeles Clippers	112	0	19,068
2	Wed, Oct 23, 2019	Chicago Bulls	125	Charlotte Hornets	126	0	15,424
3	Wed, Oct 23, 2019	Detroit Pistons	119	Indiana Pacers	110	0	17,923
4	Wed, Oct 23, 2019	Cleveland Cavaliers	85	Orlando Magic	94	0	18,846
...	...	...	...	...	...	...	...
966	Tue, Mar 10, 2020	Brooklyn Nets	104	Los Angeles Lakers	102	0	18,997
967	Wed, Mar 11, 2020	Detroit Pistons	106	Philadelphia 76ers	124	0	20,172
968	Wed, Mar 11, 2020	New York Knicks	136	Atlanta Hawks	131	1	15,393
969	Wed, Mar 11, 2020	Charlotte Hornets	109	Miami Heat	98	0	19,6
970	Wed, Mar 11, 2020	Denver Nuggets	97	Dallas Mavericks	113	0	20,302

Figura 3. Exemple de Dataframe extret d'unes dades utilitzades pel projecte

- **Scipy**: SciPy és una biblioteca de codi obert d'eines i algoritmes matemàtics per Python. SciPy conté mòduls per optimització, àlgebra lineal, integració, interpolació, funcions especials, FFT, processament de senyals i d'imatge, resolució d'EDOs i altres tasques relacionades amb la ciència i enginyeria. Està dirigida al mateix tipus d'usuaris que els d'aplicacions com MATLAB, GNU Octave, i Scilab.

- **Scikit-learn**: Scikit-learn (també conegut per sklearn) és una extensió del llenguatge Python en forma de biblioteca informàtica que agrega suport en l'àmbit del Machine Learning. Scikit-learn és de codi obert i disposa d'algorismes de classificació estadística, regressió i clustering per a implementar Màquines de vector de suport, random forests, gradient boosting, Algorisme k-means i DBSCAN. Scikit-learn està dissenyat per a integrar-se conjuntament amb les biblioteques numèriques Numpy i SciPy.

- **Itertools**: Itertools es un mòdul de la llibreria estàndard de Python que incorpora funcions que retornen objectes iterables, és dir, estructures de dades basades en elements que poden ser recorreguts seqüencialment i que poden utilitzar-se en processos repetitius (bucles). Aquestes funcions estan dissenyades per una execució ràpida, fent un ús eficient de la memòria, amb la idea de resoldre algoritmes basats en bucles més complicats que aquells que habitualment es solen implementar en un programa per recórrer els elements de una llista, diccionari, etc.

- **MatplotLib**: Matplotlib és una biblioteca de programari per a generar gràfiques a partir de dades contingudes en llistes, o vectors, en el llenguatge de programació Python i en la seva extensió matemàtica NumPy. Proporciona una API, Pylab, dissenyada per ser similar a les funcions gràfiques de MATLAB. Matplotlib és una aplicació de codi obert.

- **Seaborn**: Seaborn es una llibreria de visualització de dades en Python basada en Matplotlib. L'idea de Seaborn és que els científics de dades disposin d'una interfície per fer gràfics estadístics atractius i explicatius: l'objectiu és visualitzar dades complexes de forma senzilla i extreure'n conclusions.

El principal tipus de fitxer que s'utilitzaran seran els fitxers en format **.csv** on es guardaran les dades recol·lectades per aquest projecte.

Els fitxers en format **.csv** (de l'anglès comma-separated values) són un tipus de document en format obert senzill per representar dades en forma de taula, en què les columnes se separen per comes (o punt i coma on la coma és el separador decimal: Catalunya, França, Itàlia...) i les files per salts de línia. Els camps que continguin una coma, un salt de línia o una cometa doble, han de ser tancats entre cometes dobles.

El format **.csv** és molt senzill i no indica un joc de caràcters concret, ni com van situats els bytes, ni el format pel salt de línia. Aquests punts s'han d'indicar molts cops en obrir el fitxer, per exemple, amb un full de càlcul.

La resta de conceptes de programació que s'utilitzaran durant el treball són els que s'ensenyen al grau en tecnologies industrials de la ETSEIB i per tant no es detallaran en aquest projecte, ja que es donen per coneguts.

.

3. Fases I i II. Business Understanding, Data Understanding i Data Preparation.

Aquesta fase inicial s'enfoca la comprensió dels objectius del projecte. Després es converteix aquest coneixement de les dades i tot els aspectes relacionats, en la definició d'un problema de Data Mining i en un pla preliminar per aconseguir els objectius establerts.

Primer de tot recordem l'objectiu fonamental d'aquest projecte. Es tracta de respondre la següent pregunta:

- Es pot arribar a predir els resultats dels partits de la NBA i si és així amb quina precisió es pot fer?

Per respondre aquesta pregunta primer de tot caldrà conèixer el funcionament bàsic de la NBA, ja que totes les dades que utilitzarem corresponen a aquest lliga de bàsquet.

3.1.1. Bases del funcionament de la NBA

Per poder entendre de una forma bàsica el funcionament de la NBA, cal definir algunes de les seves característiques. D'aquesta manera podrem interpretar molt millor les dades recollides, ja que és molt important entendre que ens expliquen les dades que utilitzarem.

La NBA (National Basketball Association, traduït al català: Associació Nacional de Basquetbol) és la principal lliga professional de bàsquet als Estats Units. Se la considera habitualment la lliga més competitiva del món. La lliga es va fundar a la ciutat de Nova York el 6 de juny de 1946 com a Basketball Association of America (BAA). Va canviar el seu nom per la National Basketball Association el 3 d'agost de 1949, després de fusionar-se amb la competència de la National Basketball League (NBL). L'NBA va ser fundada per 11 equips, i després de diverses ampliacions, reduccions i trasllats, actualment la formen 30 equips: 29 dels Estats Units i un del Canadà.

La lliga professional americana de bàsquet està dividida en dues conferències, la Conferència Est i la Conferència Oest (que conté 15 equips cada una), que consten de 3 divisions cada una (3 divisions per conferència de 5 equips cadascuna). La raó de ser d'aquesta divisió és que el EEUU és un país molt gran i es divideixen els equips amb el criteri de proximitat geogràfica.

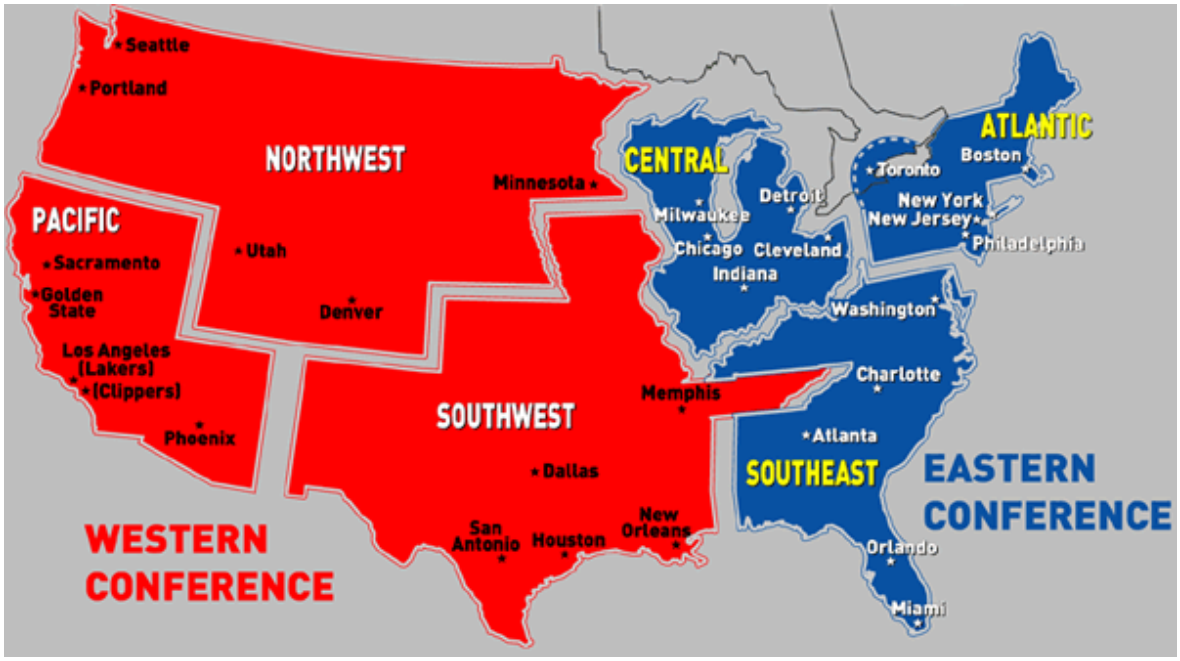


Figura 4. Mapa de la divisió de conferències i divisions dels equips de la NBA.

La competició es desenvolupa de la següent manera, primerament es juguen una sèrie de partits de pretemporada i després la temporada regular, de 82 partits. Finalment els 8 equips més ben classificats de cada Conferència s'enfronten entre ells en eliminatòries al millor de 7 partits, aquest procés és conegut com a Playoffs.

La temporada regular consta de 82 partits. Cada equip juga 41 partits com a local i 41 com a visitant. Els adversaris de cada partit es distribueixen de la següent manera: 4 partits contra els equips de la mateixa divisió, 3 o 4 partits contra els equips de la mateixa conferència, però de diferent divisió i dos partits contra els de l'altra conferència.

Cal deixar clar que en aquest treball s'utilitzaran les dades de la temporada 2019-2020 com a fil conductor del procés del treball, no obstant també es realitzarà el mateix procés posteriormentamb les temporades 2017-2018 i 2018-2019. A l'apartat de anàlisis de resultats si que s'inclouran els resultats de les tres temporades analitzades (Això es degut a que el procés de les tres temporades serà pràcticament igual fins al moment de l'anàlisi de dades).

Conferència Est		Conferència Oest	
Divisió	Equip	Divisió	Equip
Atlàntica	Boston Celtics	Nord-oest	Denver Nuggets
	Brooklyn Nets		Minnesota Timberwolves
	New York Knicks		Portland Trail Blazers
	Philadelphia 76ers		Oklahoma City Thunder
	Toronto Raptors		Utah Jazz
Central	Chicago Bulls	Sud-oest	Dallas Mavericks
	Cleveland Cavaliers		Houston Rockets
	Detroit Pistons		Memphis Grizzlies
	Indiana Pacers		New Orleans Pelicans
	Milwaukee Bucks		San Antonio Spurs
Sud-est	Atlanta Hawks	Pacífic	Golden State Warriors
	Charlotte Hornets		Los Angeles Clippers
	Miami Heat		Los Angeles Lakers
	Orlando Magic		Phoenix Suns
	Washington Wizards		Sacramento Kings

Taula 1. Taula amb el nom dels 30 equips dividits per conferència i divisió.

3.1.2. Factors que afecten al resultat d'un partit

A continuació cal preguntar-se quins factors que puguem conèixer amb anterioritat a la disputa del partit influeixen en el resultat. Aquests factors que es detallaran a continuació sorgeixen del sentit comú i de la experiència en el seguiment per part de l'autor del treball. Poden haver-ne molts més que l'autor no ha esta capaç de identificar.

- L'equip rival al que s'enfronta a l'equip analitzat.
- El descans dels equips entre els últims partits i l'analitzat.
- La dinàmica dels equips, ratxes positives, negatives, etc.
- Els viatges entre ciutats.
- Les lesions dels jugadors.
- Els jugadors dels que disposa cada equip.
- Si l'equip en qüestió juga com a local o com a visitant.

Un cop definits aquests factors totes les dades que es recolliran aniran orientades a analitzar la importància d'aquests factors.

A l'hora de recollir les dades cal tenir en compte que cada temporada es disputen una gran quantitat de partits. En total són 1230 partits, i cada un dels 30 equips de la lliga en disputa un total de 82. A més hi participen un total de 450/500 jugadors cada temporada, això ens donarà una gran quantitat de dades que analitzar i processar.

El pla preliminar per resoldre la pregunta que s'ha plantejat consisteix en els següents passos:

Recol·lecció de les dades i automatització de la seva recol·lecció.

Netejar i adaptar les dades recollides per poder extreure'n les variables que s'utilitzaran per l'anàlisi fàcilment (Data cleaning).

Un cop completada la fase de Business Understading, cal passar a la fase de Data Understanding. La fase de comprensió de les dades comença amb la recol·lecció inicial de dades i continua amb les activitats que permeten familiaritzar-se amb les dades, identificar els problemes de qualitat, coneixement preliminar sobre les dades, o descobrir subconjunts interessants per formular hipòtesis en cas que hi hagi informació oculta.

Pel que fa a la col·lecta de dades l'escenari ideal seria recollir totes les dades de forma automàtica, degut al temps limitat que es disposa en aquest treball, no s'ha pogut aconseguir totes les dades necessàries de forma automàtica. No obstant en alguns dels conjunts de dades que veurem a continuació si que apliquem tècniques de web scraping i la utilització de API's que ajuden molt a la col·lecta de dades.

3.1.3. Data i resultats de cada partit de la temporada

Les dades més importants que cal recollir per respondre a la pregunta inicial són els resultats de tots els partits disputats durant la temporada

Tot seguit emmagatzemarem aquestes dades en un fitxer en format .csv i procedirem a analitzar les dades de les que disposem .

El primer fitxer que importarem conté la següent informació sobre cada partit jugat durant la temporada 19/20 de la NBA a cada columna.

- **Date**: Conté un string amb la data del partit amb format (dia de la setmana, mes dia, any).
- **Start(ET)**: hora en què va començar el partit en hora de la costa est d'Estats Units.
- **Visitor**: Nom complet de l'equip visitant en format string.
- **PTSV**: Punts anotats per l'equip visitant al final del partit en format int.
- **Home**: Nom complet de l'equip local en format string.
- **PTSH**: Punts anotats per l'equip local al final del partit en format int.
- **Overtime**: Indica amb un string 'OT' si hi ha hagut pròrroga en el partit, si no hi ha hagut pròrroga ho indica amb una casella en format NaN.
- **Attend**: Indica l'assistència de públic en format string.

A continuació importem el fitxer i en veiem unes quantes files del mateix.

	Date	Start (ET)	Visitor	PTSV	Home	PTSH	Overtime	Attend.
0	Tue, Oct 22, 2019	8:00p	New Orleans Pelicans	122	Toronto Raptors	130	OT	20,787
1	Tue, Oct 22, 2019	10:30p	Los Angeles Lakers	102	Los Angeles Clippers	112	NaN	19,068
2	Wed, Oct 23, 2019	7:00p	Chicago Bulls	125	Charlotte Hornets	126	NaN	15,424
3	Wed, Oct 23, 2019	7:00p	Detroit Pistons	119	Indiana Pacers	110	NaN	17,923
4	Wed, Oct 23, 2019	7:00p	Cleveland Cavaliers	85	Orlando Magic	94	NaN	18,846
...	...	...	...	...	...	...	...	...
966	Tue, Mar 10, 2020	10:30p	Brooklyn Nets	104	Los Angeles Lakers	102	NaN	18,997
967	Wed, Mar 11, 2020	7:00p	Detroit Pistons	106	Philadelphia 76ers	124	NaN	20,172
968	Wed, Mar 11, 2020	7:30p	New York Knicks	136	Atlanta Hawks	131	OT	15,393
969	Wed, Mar 11, 2020	7:30p	Charlotte Hornets	109	Miami Heat	98	NaN	19,6
970	Wed, Mar 11, 2020	8:00p	Denver Nuggets	97	Dallas Mavericks	113	NaN	20,302

Taula 2. Mostra d'alguna de les columnes del Dataframe que conté els resultats dels partits.

Per tal de preparar les dades per procedir a l'anàlisi primer de tot caldrà manipular aquestes dades de forma que ens puguin ser útils i extreure'n tota la informació possible. Per tant a continuació comencem amb el procés de Data Cleaning.

Primer de tot eliminem les columnes del dataframe data que no ens interessen. En aquest cas es tracta de l'hora del partit que no s'ha considerat que sigui rellevant a l'hora de poder preveure el resultat.

A continuació també cal arreglar el format de la columna Overtime, substituirem els valors NaN per un 0. També canviarem els strings que contenen 'OT' o '2OT', '3OT', '4OT', etc. Per un 1. considerem que tant si hi ha hagut una pròrroga com quatre el mateix cas per tal de simplificar la columna a només 0 o 1.

```
#Substituim 'NaN' a la columna Overtime per un 0 i'OT' per 1

data=data.fillna(0)
data=data.replace(['OT','2OT','3OT','4OT'],1)
data
```

	Date	Visitor	PTSV	Home	PTSH	Overtime	Attend.
0	Tue, Oct 22, 2019	New Orleans Pelicans	122	Toronto Raptors	130	1	20,787
1	Tue, Oct 22, 2019	Los Angeles Lakers	102	Los Angeles Clippers	112	0	19,068
2	Wed, Oct 23, 2019	Chicago Bulls	125	Charlotte Hornets	126	0	15,424
3	Wed, Oct 23, 2019	Detroit Pistons	119	Indiana Pacers	110	0	17,923
4	Wed, Oct 23, 2019	Cleveland Cavaliers	85	Orlando Magic	94	0	18,846
...	...	...	...	...	...	...	...
966	Tue, Mar 10, 2020	Brooklyn Nets	104	Los Angeles Lakers	102	0	18,997
967	Wed, Mar 11, 2020	Detroit Pistons	106	Philadelphia 76ers	124	0	20,172
968	Wed, Mar 11, 2020	New York Knicks	136	Atlanta Hawks	131	1	15,393
969	Wed, Mar 11, 2020	Charlotte Hornets	109	Miami Heat	98	0	19,6
970	Wed, Mar 11, 2020	Denver Nuggets	97	Dallas Mavericks	113	0	20,302

Figura 5. Codi i taula resultant del procés de Data Cleaning

A continuació volem modificar el format de la data de disputa del partit, a un format que sigui igual per totes les dades recollides, (normalment quan s'extreuen dades de diferents fonts les dates estan expressades de forma diferent en cada font) d'aquesta manera ens serà molt més fàcil manipular i creuar dades dels diferents Dataframes. Utilitzarem la funció de Pandas .to_datetime per canviar el format de la data de forma senzilla. Aquest eina canviara el format de la data de cada partit de la forma següent('Tue, Oct 22, 2019' a '2019-10-22').

```
data['Date'] =  pd.to_datetime(data['Date'], format='%a, %b %d, %Y')

data
```

	Date	Visitor	PTSV	Home	PTSH	Overtime	Attend.
0	2019-10-22	New Orleans Pelicans	122	Toronto Raptors	130	1	20,787
1	2019-10-22	Los Angeles Lakers	102	Los Angeles Clippers	112	0	19,068
2	2019-10-23	Chicago Bulls	125	Charlotte Hornets	126	0	15,424
3	2019-10-23	Detroit Pistons	119	Indiana Pacers	110	0	17,923
4	2019-10-23	Cleveland Cavaliers	85	Orlando Magic	94	0	18,846
...	...	...	...	...	...	...	...
966	2020-03-10	Brooklyn Nets	104	Los Angeles Lakers	102	0	18,997
967	2020-03-11	Detroit Pistons	106	Philadelphia 76ers	124	0	20,172
968	2020-03-11	New York Knicks	136	Atlanta Hawks	131	1	15,393
969	2020-03-11	Charlotte Hornets	109	Miami Heat	98	0	19,6
970	2020-03-11	Denver Nuggets	97	Dallas Mavericks	113	0	20,302

Figura 6. Captura del Dataframe amb la data modificada al nou format i la columna overtime també modificada.

A continuació realitzem un petit anàlisis exploratori per extreure les primeres conclusions d'aquest Dataframe.

S'obté la següent informació estadística bàsica de totes les columnes en format numèric del Dataframe en aquest cas PTSV, PTSH.

	PTSV	PTSH	Overtime
count	971.000000	971.000000	971.000000
mean	110.359423	112.533471	0.062822
std	12.147783	12.578728	0.242767
min	76.000000	73.000000	0.000000
25%	102.000000	104.000000	0.000000
50%	110.000000	112.000000	0.000000
75%	119.000000	120.000000	0.000000
max	159.000000	158.000000	1.000000

Taula 3. Taula amb dades estadístiques bàsiques sobre el Dataframe anterior.

Podem observar que el nombre de partits del Dataframe és 971 (ja que la temporada actual no s'ha disputat al complert a causa del COVID-19), també podem observar que la mitjana de punts anotats per els equips que juguen a casa és superior a la dels equips que juguen a fora de casa. A més podem veure els valors màxims i mínims d'anotació i el valor dels quartils. També s'observa que una mica més del 6% dels partits es disputa acaba amb la disputa de pròrroga.

3.1.4. Estadístiques avançades de cada jugador al llarg de la temporada

Per tal de atacar el factor de les lesions i les absències de jugadors en els partits i la seva influencia en el resultat final, es busquen les estadístiques avançades de cada jugador en el global de la temporada.

El següent fitxer de dades que importarem conté una gran quantitat d'estadística avançada de tots els jugadors de la NBA durant aquesta temporada.

Malgrat això només ens interessarà per el projecte quatre columnes que són les següents.

- **Player**: conté el nom complet del jugador en format string.
- **Age:** Conté un enter amb el valor de l'edat del jugador
- **Tm**: conté el nom de l'equip en què juga abreviat amb 3 lletres majúscules, també en format string.
- **STAR**: conté un 1 si el jugador és considerat una estrella de la lliga per el rànking de top 50 jugadors de la NBA que es fa a l'inici de cada temporada. En cas que el jugador no estigui en aquest Top 50 la taula contindrà un 0.

A continuació importem el fitxer i en veiem unes quantes files del mateix.

```
data_players=pd.read_csv("NBA_project/players_stats_19_20.csv")
data_players
```

	Player	Pos	Age	Tm	G	MP	PER	TS%	3PAr	FTr	...	USG%	OWS	DWS	WS	WS/48	OBPM	DBPM
0	James Harden	SG	30	HOU	61	2241	28.4	0.616	0.555	0.519	...	36.4	8.7	2.7	11.5	0.245	7.9	1.2
1	Giannis Antetokounmpo	PF	25	MIL	57	1763	31.6	0.608	0.238	0.500	...	37.4	5.6	4.8	10.4	0.282	7.4	4.1
2	LeBron James	PG	35	LAL	60	2094	26.0	0.582	0.324	0.292	...	31.6	6.1	3.4	9.5	0.218	6.8	1.9
3	Nikola Jokic	C	24	DEN	65	2101	25.0	0.604	0.233	0.277	...	26.6	6.0	3.2	9.2	0.209	5.4	2.3
4	Anthony Davis	PF	26	LAL	55	1889	28.2	0.614	0.195	0.460	...	29.7	6.2	4.1	10.3	0.262	5.8	2.8
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
494	RJ Barrett	SG	19	NYK	56	1704	10.7	0.479	0.271	0.349	...	24.0	-1.6	1.1	-0.5	-15.000	-2.8	-1.5
495	Dillon Brooks	SG	24	MEM	65	1851	11.1	0.508	0.387	0.192	...	25.0	-0.3	1.3	1.0	0.025	-2.8	-1.5
496	De'Andre Hunter	SF	22	ATL	63	2018	8.6	0.521	0.445	0.211	...	17.5	-0.4	0.5	0.1	0.001	-2.8	-1.8
497	Jordan Poole	SG	20	GSW	57	1274	7.2	0.454	0.528	0.237	...	21.1	-1.6	0.4	-1.2	-47.000	-4.4	-2.2
498	Darius Garland	PG	20	CLE	59	1824	8.5	0.498	0.423	0.103	...	20.7	-1.3	0.0	-1.3	-35.000	-2.7	-2.9

Figura 7. Mostra del Dataframe que recull les estadístiques de cada jugador al llarg de la temporada

Ens interessa aquest Dataset sobretot perquè relaciona cada jugador amb l'equip que pertany i també ens indica si el jugador es considerat una estrella de la lliga i per tant un jugador molt important a l'inici de temporada. També ens n'indica la edat que serà un altre tipus de informació que s'utilitzarà.

3.1.5. Distàncies en Km entre les ciutats on es disputen els partits.¶

Un altre factor que hem comentat és la gran quantitat de viatges i kilòmetres que realitzen setmanalment tots els equips per desplaçar-se als partits fora de casa per això volem el següent conjunt de dades.

Per obtenir aquestes dades s'ha creat un codi basat en el "web scraping" per tal d'obtenir les distancies entres les 28 ciutats on es disputa la NBA. No són 30 ja que Nova York i Los Ángeles tenen dos equips cada una. Tarda uns 5 min a calcular-se, ja que s'ha de vigilar no saturar la web amb masses 'calls', no obstant no importa molt en aquest cas, ja que les distancies seran sempre les mateixes i només cal calcular-ho un cop.

En aquest exemple s'han calculat les distàncies entre les ciutats que pertanyen a la conferència Oest. Després cal fer-ho amb les ciutats de la conferència Est i finalment entre les de l'Est i l'Oest. Una vegada ho ajuntem tot en unt fitxer d'Excel obtenim el següent Dataframe amb les següents dades.

L'única manera de obtenir aquestes dades d'una forma eficient era utilitzar el web scraping i la lectura de scripts html utilitzant les llibreries beautifulsoup i urllib, ja que hi ha moltes

ciutats i la quantitat de distàncies a calcular són moltes al voltant de 860.

```
#La API de WolframAlpha ens retorna els Km entre dos localitzacions.
import urllib3
appID = "GUJ6PG-PK9RG8VL6K" #specific to this program
data = pd.DataFrame(columns=('City Origin','City Destiny','Distance (Km)'))
from bs4 import BeautifulSoup

http = urllib3.PoolManager()
def miles_between(city1, city2):
    query = "Distance+between+" + city1 + "+and+" + city2
    url = "http://api.wolframalpha.com/v2/query?input="+query+"&appid="+str(appID)+"&includepodid=Result"
    http = urllib3.PoolManager()
    response = http.request('GET', url)
    response=response.data.decode('utf-8')
    soup = BeautifulSoup(response, 'html.parser')
    s = soup.get_text()
    dis=' '
    distance=''
    for e in s:
        if e!='\n' and e in ['.','0','1','2','3','4','5','6','7','8','9']:
            dis=dis+str(e)

    return   float(dis)
def farthest_cities_within(conference1, conference2):
    cit=[]
    las=[]
    km=[]
    maxDistance = ["city1", "city2", 0]

    while len(conference1) > 0:
        last = conference1.pop()
        for city in conference2:
            d = miles_between(last, city)
            print (last + ", " + city + ": " + str(d))
            las=np.append(las,last)
            cit=np.append(cit,city)
            km=np.append(km,str(d))
            if d > maxDistance[2]:
                maxDistance[0] = last
                maxDistance[1] = city
                maxDistance[2] = d

    return maxDistance,las,cit,km

atlantic = ["Boston", "New+York+City", "Philadelphia", "Toronto"]
central = ["Chicago", "Cleveland", "Auburn+Hills", "Indianapolis", "Milwaukee"]
southeast = ["Atlanta", "Charlotte", "Miami", "Orlando", "Washington+DC"]

northwest = ["Denver", "Minneapolis", "Oklahoma+City", "Portland", "Salt+Lake+City"]
pacific = ["Oakland", "LA", "Phoenix", "Sacramento"]
southwest = ["Dallas", "Houston", "Memphis", "New+Orleans", "San+Antonio"]

east = atlantic + central + southeast
west = northwest + pacific + southwest

maxdis,city1,city2,distance=farthest_cities_within(west, west)
```

Figura 8. Captura del codi que permet recollir les distancies entre totes les ciutats automàticament.

El següent fitxer conté informació de la distància entre totes les ciutats que tenen equips de la NBA en kilòmetres. Conté les següents columnes de dades:

- **City Origin**: Conté un string amb el nom de l'equip de la ciutat origen. Per exemple 'Washington Wizards' es refereix a la ciutat de Washington DC dels EEUU.
- **City Destiny**: Conté un string amb el nom de l'equip de la ciutat destí del viatge. Per exemple 'Denver Nuggets' es refereix a la ciutat de Denver dels EEUU.
- **Distance (km)**: conté la distància en format float dels km en línia recta entre les dues ciutats on es farà el viatge. Aquesta taula s'ha aconseguit utilitzant un altre codi fent web scrapping que ja s'ha explicat anteriorment.

```
data_distance=pd.read_csv("NBA_project/distances_NBA.csv")
data_distance
```

	City Origin	City Destiny	Distance (Km)
0	Washington Wizards	Denver Nuggets	2395.0
1	Washington Wizards	Minnesota Timberwolves	1503.0
2	Washington Wizards	Oklahoma City Thunder	1855.0
3	Washington Wizards	Portland Trail Blazers	3786.0
4	Washington Wizards	Utah Jazz	2976.0
...	...	...	...
872	Los Angeles Lakers	Phoenix Suns	587.5
873	Los Angeles Lakers	Brooklyn Nets	3966.0
874	Los Angeles Clippers	Los Angeles Lakers	0.0
875	Los Angeles Lakers	Los Angeles Clippers	0.0
876	Denver Nuggets	Denver Nuggets	0.0

Figura 9. Dataframe que conté la distància en Km entre dues ciutats.

3.1.6. Victòries de cada equip corresponents a la temporada anterior

Per tal de poder inicialitzar de forma correcte la anàlisis de resultats en cada una de les temporades que volem simular, cal que utilitzem aquest valor de victòries que va aconseguir cada equip la temporada anterior a la analitzada. A principi de temporada, quan encara no hi ha suficient volum de dades, per tal que el model pugui decidir amb precisió. És a dir el nombre de victòries de la temporada anterior sobre un total de 82 partits que disputa cada equip, servirà per predir el resultats dels primers partits de la temporada següent. Guardarem aquestes dades en un diccionari que anomenarem wins_last_year.

Com es pot veure conté com a clau el nom complet de l'equip i com a valor associat un valor 'float' amb el número de victòries sobre un total de 82 possibles.

{'Atlanta Hawks': 29, 'Boston Celtics': 49, 'Brooklyn Nets': 42, 'Charlotte Hornets': 39, 'Chicago Bulls': 22, 'Cleveland Cavaliers': 19, 'Dallas Mavericks': 33, 'Denver Nuggets': 54, 'Detroit Pistons': 41, 'Golden State Warriors': 57, 'Houston Rockets': 53, 'Indiana Pacers': 48, 'Los Angeles Clippers': 48, 'Los Angeles Lakers': 37, 'Memphis Grizzlies': 33, 'Miami Heat': 39, 'Milwaukee Bucks': 60, 'Minnesota Timberwolves': 36, 'New Orleans Pelicans': 33, 'New York Knicks': 17, 'Oklahoma City Thunder': 49, 'Orlando Magic': 42, 'Philadelphia 76ers': 51, 'Phoenix Suns': 19, 'Portland Trail Blazers': 53, 'Sacramento Kings': 39, 'San Antonio Spurs': 48, 'Toronto Raptors': 58, 'Utah Jazz': 50,'Washington Wizards': 32}

Figura 10. Diccionari que conté les victòries de cada equip la temporada anterior.

3.1.7. Estadístiques individuals per cada jugador en cada un dels partits de la temporada.

Per poder saber quins jugadors es van perdre algun partit de la temporada i poder-ho incloure en les dades necessitarem saber tots els partits disputats i la data d'aquest partit per cada un dels jugadors de la lliga això suposarà un data set diferent per cada un dels jugadors. És a dir, tindrem més de 450 datasets en aquesta categoria de dades.

Per aconseguir aquestes estadístiques de forma eficient i ràpida caldrà utilitzar una API de Python que es dedica exclusivament a obtenir dades de la NBA (NBA API). El codi per obtenir les dades que volem de tots els jugadors és el següent (Cal tenir en compte que hi ha més de 450 jugadors que disputen cada temporada i que per tant a vegades cal llençar més de una vegada el càlcul ja que la API es satura si rep masses "calls" seguides).

```
# Busquem la id dels jugadors a la API
from nba_api.stats.static import players

# Dona una llista de diccionaris que conté la Id del jugador i el seu nom entre d'altre. Per exemple:
#{'id': 76410, 'full_name': 'Richard Coffey', 'first_name': 'Richard', 'last_name': 'Coffey', 'is_active': False}
player_dict = players.get_players()

#Ens retorna un diccionari amb els jugadors que estan en actiu actualment ja que la API també té dades
#de jugadors desde fa 20 temporades
def get_active_players_id(player_dict):
    active_players={}
    for player in player_dict:
        if player['is_active']==True:
            active_players[player['full_name']]=player['id']
    return active_players

active_players=get_active_players_id(player_dict)


split_idx = 473
active_players_done = dict(list(active_players.items())[:split_idx])
active_players_not_done= dict(list(active_players.items())[split_idx:])

# Volem les estadístiques de la temporada 2019/2020 per cada jugador en actiu
from nba_api.stats.library.parameters import SeasonAll
from nba_api.stats.endpoints import playergamelog
from nba_api.stats.endpoints import playerdashboardbyclutch
```

Figura 11. Captura del codi que permet recollir el Dataframe per cada jugador.

Aquest codi ens generarà un fitxer com el següent (*Figura 12*) per cada un dels jugadors actius de la NBA. Aquest fitxer conté una gran quantitat de dades de cada partit disputat pel jugador durant la temporada, però el que ens interessarà realment de cada jugador serà les següents columnes:

- **GAME_DATE**: Conté la data del partit en format “string” (mes(“string”), dia, any).
- **MATCHUP** : Conté els equips que juguen en format abreviatiu dels equips en només tres lletres.

Més endavant caldrà adaptar el format de la data de cada partit i també el dels equips que s'enfronten, per fer que sigui igual que el format de data i equips que hem fet per la resta de dades.

```
LBJ= pd.read_csv("NBA_project/players_19_20/games_LeBron James.csv")
LBJ
```

	SEASON_ID	Player_ID	Game_ID	GAME_DATE	MATCHUP	WL	MIN	FGM	FGA	FG_PCT	...	DREB	REB	AST	STL	BLK	TOV	PF	PTS	PLI
0	22019	2544	21900968	MAR 10, 2020	LAL_vs_BKN	L	35	12	22	0.545	...	11	12	9	1	0	3	1	29	7
1	22019	2544	21900948	MAR 08, 2020	LAL_at_LAC	W	35	7	17	0.412	...	6	8	9	0	2	2	3	28	7
2	22019	2544	21900939	MAR 06, 2020	LAL_vs_MIL	W	37	12	21	0.571	...	8	8	8	3	0	4	4	37	8
3	22019	2544	21900915	MAR 03, 2020	LAL_vs_PHI	W	34	9	16	0.563	...	6	7	14	1	2	3	1	22	-2
4	22019	2544	21900900	MAR 01, 2020	LAL_at_NOP	W	36	14	21	0.667	...	12	12	13	2	0	6	2	34	23
5	22019	2544	21900891	FEB 29, 2020	LAL_at_MEM	L	34	8	18	0.444	...	6	8	10	1	1	5	1	19	-12
6	22019	2544	21900861	FEB 25, 2020	LAL_vs_NOP	W	34	17	27	0.630	...	6	8	6	0	1	7	1	40	12
7	22019	2544	21900842	FEB 23, 2020	LAL_vs_BOS	W	35	9	19	0.474	...	8	8	9	0	0	2	4	29	-1
8	22019	2544	21900833	FEB 21, 2020	LAL_vs_MEM	W	36	10	17	0.588	...	3	3	7	0	2	1	0	32	6
9	22019	2544	21900817	FEB 12, 2020	LAL_at_DEN	W	42	15	29	0.517	...	11	12	14	0	0	3	3	32	4
10	22019	2544	21900801	FEB 10, 2020	LAL_vs_PHX	W	30	6	16	0.375	...	8	8	9	1	0	8	2	17	16

Figura 12. Dataframe que conté tots els partits disputats per LeBron James.

3.2. Fase II. Data Preparation. Anàlisis de les dades i selecció de les característiques

La fase de la preparació de dades cobreix totes les activitats necessàries per construir el conjunt final de dades (les dades que s'utilitzaran en les eines de modelatge) a partir de les dades brutes inicialment recollides. Les tasques inclouen la selecció de taules, registres, atributs, així com la transformació i la neteja de dades per a les eines posteriors de modelatge.

Un cop obtingudes totes les dades que utilitzarem cal definir quines seran les variables que extraurem d'aquestes dades, per fer el posterior anàlisi.

La idea principal és preveure els resultats dels partits a partir de tots els resultats que s'han donat en partits anteriors durant la temporada. És a dir el model anirà aprenent a poc a poc durant la temporada dels resultats anteriors que s'han donat. Es per això que totes les variables que extraurem de les dades recollides, es basen en la actualització partit a partit i són acumulatives al llarg de la temporada.

Per posar un exemple molt clar s'explicarà amb la primera variable que es vol obtenir, que és el percentatge de victòries de cada equip fins al moment 't' de la temporada.

És a dir si es vol predir el resultat d'un dels equips en el partit, per exemple, de la jornada 9, el que ens donarà la variable serà el percentatge de victòries de aquell equip en els 8 primers partits de lliga. És a dir si per exemple l'equip en qüestió ha guanyat 6 dels 8 partits ja disputats el valor de la variable per aquell partit serà el 75% de victòries o expressat en tant per u 0,75. Així es farà successivament per les següents jornades. Per exemple la variable de la jornada 10, on per exemple l'equip perd el partit anterior la variable serà 0,66 o 66% ja que en els primers 9 partits l'equip n'haurà guanyat 6 de 9.

En resum aquesta serà la dinàmica pel que fa la obtenció de variables. Sempre busquem tenir els valors de cada variable percentatge de victòries, punts anotats per partit, etc. Actualitzats amb les dades acumulades anteriors a cada partit.

Per tant les funcions que es dissenyaran a continuació per extreure els valors de les variables desitjades, seran funcions que utilitzaran la data de disputa de cada partit per saber en cada moment, quin valor de cada variable tenia cada equip en cada jornada del campionat.

Utilitzar les dades d'aquesta manera ens permet obtenir estadístiques que copsen el

moment de forma de cada equip, ja que si agaféssim aquests valors simplement com els globals de cada temporada, la precisió a la hora de predir resultats podria ser més baixa, ja que no s'estaria tenint en compte la conjuntura de cada moment sinó només la visió global de la temporada. Per exemple no estaria tenint en compte coses tant important en el bàsquet com lesions, acumulació de partits, moments de forma o ratxes positives i negatives.

Totes les funcions que es dissenyen a continuació es fan amb l'objectiu de poder tenir en compte diferents factors més enllà de la pura estadística bàsica, però que tenen un gran impacte en els resultats dels partits.

A continuació hi ha una taula on es detalla les variables que es crearan i quins factors que afecten al resultat, que hem definit anteriorment, van relacionats a cada variable.

		FACTOR					
Nom de la variable	Abreviatura	EQUIP RIVAL	DESCANS	RATXES	VIATGES	LESIONS	LOCAL/VISITANT
Percentatge de victòries	%W	■					
Percentatge de victòries com a local i visitant	%W as home/visitor	■					■
Càlcul del temps de descans entre partits consecutius dels equips	Back to back		■				
Parametritzar la divisió en la que juga cada un dels 30 equips	Division	■					
Classificiació dels equips en funció de la conferència en què juga cada equip	Conference	■					
Percentatge de victòries en els últims N partits	%W last N	■		■			
Càlcul del Net Rating, diferència de punts anotats a favor i en contra	Net rating	■					
Càlcul de si s'ha jugat pròrroga a l'últim partit disputat	OT		■				
Paràmetre d'absència de les estrelles de l'equip	rest_inj					■	
Ratxa de N victòries o derrotes	N wins			■			
Distància recorreguda per cada equip	Km				■		
Edat mitjana dels jugadors de cada equip	Age	■					
Paràmetre que indica l'equip guanyador de cada partit	winner	És el resultat que volem predir					

Taula 4. Variables que es crearan i com afecten als factors que hem definit com a influents en el resultat d'un partit.

També cal diferenciar les variables entre les que s'actualitzen els seus valors amb la disputa de cada partit i les variables que no canviaran en cap moment al llarg de la temporada, ja que són constants.

Nom de la variable	Símbol	Variables constants al llarg de la temporada	Variables que varien cada partit
Percentatge de victòries	%W		
Percentatge de victòries com a local i visitant	%W as home/visitor		
Càlcul del temps de descans entre partits consecutius dels equips	Back to back		
Parametritzar la divisió en la que juga cada un dels 30 equips	Division		
Classificiació dels equips en funció de la conferència en què juga cada equip	Conference		
Percentatge de victòries en els últims N partits	%W last N		
Càlcul del Net Rating, diferència de punts anotats a favor i en contra	Net rating		
Càlcul de si s'ha jugat pròrroga a l'últim partit disputat	OT		
Paràmetre d'absència de les estrelles de l'equip	rest_inj		
Ratxa de N victòries o derrotes	N wins		
Distància recorreguda per cada equip	Km		
Edat mitjana dels jugadors de cada equip	Age		
Paràmetre que indica l'equip guanyador de cada partit	winner	És el resultat que volem predir	

Taula 5. Variables que es modifiquen cada partit vs variables que no canviaran al llarg de la temporada.

A continuació s'explicarà cada una de les variables amb més detall. És a dir què signifiquen i de quines dades provenen. Cal tenir en compte que alguns dels conceptes que s'usaran a continuació, són bastant tècnics relacionats amb el bàsquet i la NBA. Per tant s'explicaran amb el màxim detall possible.

3.2.1. Percentatge de victòries (%W)

La següent funció ens permetrà extreure el primer paràmetre que ens servirà per fer la predicció de resultats. Aquest paràmetre serà la diferència de % de victòries entre l'equip local i l'equip visitant just abans de començar el partit en qüestió (L'explicació detallada de totes les funcions i el codi que les fa possibles, es pot visualitzar a l'Annex 1 en la part de Data Preparation. Allà estan explicades detalladament).

Primer de tot cal explicar en què consisteix el percentatge de victòries abans de cada partit. Per posar un exemple si un equip porta 15 partits jugats i n'ha guanyat 7 el seu percentatge de victòries en aquell moment serà del 46.6% i si per exemple llavors guanya 5 partits seguits en portarà guanyats 12 sobre 20 lo que representa un 60%. Aquest percentatge de victòries actualitzat ens permet monitoritzar el moment de forma de cada equip i el global de la temporada fins aquell moment donant-nos una idea molt més acurada de les possibilitats reals de l'equip de guanyar cada partit en concret.

Per tal d'obtenir aquest percentatge de victòries actualitzat per cada partit i equip haurem de crear els següent diccionari anomenat w_percent. També crearem dos arrays buits on emmagatzemarem els percentatges de victòries actualitzats i els afegirem al Dataframe.

Aquest diccionari contindrà com a clau el nom de l'equip de l'NBA en qüestió i com a valor associat a la clau una llista amb dos 'int' el primer serà el nombre de victòries obtingudes fins el moment i el segon el nombre total de partits disputats fins aquell moment.

No obstant durant els primers partits de la temporada utilitzarem el percentatge de victòries de la temporada anterior que hem guardat al diccionari wins_last_year, d'aquesta manera evitem errors en els primers partits per manca de dades. un cop disputats com a mínim 8 partits cada equip el percentatge de victòries obtingut ja començarà a ser més rellevant.

La variable %W que obtenim és el valor de la resta entre el %W de l'equip local i l'equip visitant en aquella jornada. Aquesta resta ens dona un valor entre 1 i -1 que ens indicarà quin és el favorit per guanyar el partit si ens basem només en el percentatge de victòries. Si el valor de %W és més gran que 0 voldrà dir que l'equip local té un percentatge de victòries fins aquell moment més gran que l'equip visitant i per tant és més favorit per guanyar el partit, en cas de que el valor del %W sigui més petit que 0 la conclusió que en podem extreure serà la inversa.

	Visitor	PTSV	Home	PTSH	Overtime	Attend.	New Date	%W
0	New Orleans Pelicans	122	Toronto Raptors	130	1	20,787	22/10/19	0.073
1	Los Angeles Lakers	102	Los Angeles Clippers	112	0	19,068	22/10/19	0.049
2	Chicago Bulls	125	Charlotte Hornets	126	0	15,424	23/10/19	-0.079
3	Detroit Pistons	119	Indiana Pacers	110	0	17,923	23/10/19	0.134
4	Cleveland Cavaliers	85	Orlando Magic	94	0	18,846	23/10/19	0.201
...	...	...	...	...	...	...	...	...
966	Brooklyn Nets	104	Los Angeles Lakers	102	0	18,997	10/03/20	0.330
967	Detroit Pistons	106	Philadelphia 76ers	124	0	20,172	11/03/20	0.286
968	New York Knicks	136	Atlanta Hawks	131	1	15,393	11/03/20	-0.005
969	Charlotte Hornets	109	Miami Heat	98	0	19,6	11/03/20	0.297
970	Denver Nuggets	97	Dallas Mavericks	113	0	20,302	11/03/20	-0.081

Taula 6. Dataframe base amb la variable de percentatge de victòries incorporada.

3.2.2. Percentatge de victòries com a local i visitant (%W as local / visitor)

La següent funció ens permetrà extreure un altre paràmetre. Aquest paràmetre serà la diferència de % de victòries com a local de l'equip local i el % de victòries de l'equip visitant com a visitant just abans de començar el partit en qüestió.

Primer cal definir en què consisteix el percentatge de victòries com a local de l'equip local i el percentatge de victòries de l'equip visitant com a visitant. Com ja s'ha explicat cada partit hi ha un equip local i un de visitant. El que farem serà calcular el percentatge de victòries de cada equip tant com a local com a visitant. Per tant cada equip tindrà associat un %W com a local i un %W com a visitant. Per exemple el percentatge de victòries com a local seria de la forma següent, si un equip porta 10 partits jugats com a local i n'ha guanyat 7 el seu percentatge de victòries en aquell moment serà del 70% i per exemple el mateix equip en pot haver jugat 12 com a visitant i haver-ne guanyat 6 i tenir un 50% de victòries com a visitant.

Aquest percentatge de victòries com a local i visitant actualitzat hauríem de ser capaços de predir millor els resultats ja que té en conte no el global dels partits sinó també com ho fa cada equip a fora de casa i a casa.

Com hem observat anteriorment en la anàlisi prèvia del Dataframe dels resultats dels partits la mitjana de punts anotats per els equips que juguen a casa és superior a la dels equips que juguen a fora de casa. Això ens indica que segurament els equips de casa guanyen més partits i que per tant diferenciar el comportament dels equips quan juguen a casa o fora pot ser rellevant a l'hora de predir els resultats.

Per tal de obtenir aquest percentatge de victòries actualitzat per cada partit com a local i visitant de crear els següents diccionaris anomenats w_percent-as_local i w_percent_as_visitor. També crearem dos arrays buits on emmagatzemarem els percentatges de victòries actualitzats i els afegirem al dataframe.

Els dos diccionaris tindran com a clau el nom de l'equip de l'NBA en qüestió i com a valor associat a la clau una llista amb dos integers el primer serà el nombre de victòries obtingudes com a local/visitant fins el moment i el segon el nombre total de partits disputats com a local/ visitant fins aquell moment de la temporada.

No obstant durant els primers partits de la temporada utilitzarem el percentatge de victòries de la temporada anterior que hem guardat al diccionari "wins_last_year".

3.2.3. Càlcul del temps de descans entre partits consecutius dels equips

La següent funció ens permetrà extreure un paràmetre molt rellevant, que serà els dies de descans que ha tingut cada equip des del seu últim partit. Cal tenir en compte que la NBA té un calendari molt atapeït cada equip ha de disputar 82 partits en aproximadament 5 mesos i mig (aproximadament 165 dies), un partit cada dos dies de mitjana. Això fa que tots els equips més de 10 o 15 vegades a la temporada juguin dos partits en 2 dies o 3 en 4 dies. El cansament que això produeix pot influir de forma important en el resultat del partit.

Per tant el que volem calcular és quants dies de descans ha tingut tant l'equip local i visitant abans d'aquell partit. Per exemple si l'equip local va jugar el dia abans el valor serà 1, si va jugar fa dos dies el paràmetre serà 2 i així successivament.

Per tal de obtenir aquest nombre de dies de descans de cada equip en cada partit crearem el següent diccionari anomenat last_game. També crearem dos arrays buits on emmagatzemarem els dies de descans tant per l'equip local com per l'equip visitant.

El diccionari tindrà com a clau el nom de l'equip de l'NBA en qüestió i com a valor associat a la clau un string que contindrà la data de l'últim partit que ha disputat cada equip.

3.2.4. Parametritzar la divisió en la que juga cada un dels 30 equips

La següent funció ens permetrà extreure un paràmetre que indicarà la divisió on juga cada equip.

Primer de tot cal indicar què són les divisions. Les divisions són grups de 5 equips en els que es subdivideix la lliga, per tant hi ha 6 divisions de 5 equips cada una. Aquestes divisions es fan per proximitat geogràfica ja que Estats Units és un país enorme. Per evitar fer tants viatges els equips de cada divisió juguen més partits entre ells que amb els altres equips de la lliga. Cada equip juga 5 partits contra els equips de la mateixa divisió 4 contra els de la mateixa conferència i 2 contra els de la altre conferència.

Per poder realitzar aquesta classificació s'ha creat un diccionari que conté com a clau el nom de l'equip i com a valor associat un enter del 1 al 6 cada un dels enters correspon a una de les 6 divisions de la lliga.

```
Div={'Toronto Raptors':1, 'Boston Celtics':1, 'Philadelphia 76ers':1, 'Brooklyn Nets':1, 'New York Knicks':1,  'Milwaukee Bucks':2, 'Indiana Pacers':2, 'C
hicago Bulls':2,'Detroit Pistons':2, 'Cleveland Cavaliers':2, 'Miami Heat':3, 'Orlando Magic':3,'Washington Wizards':3,'Charlotte Hornets':3,'Atlanta Haw
ks':3, 'Denver Nuggets':4,'Utah Jazz':4,'Oklahoma City Thunder':4,'Portland Trail Blazers':4, 'Minnesota Timberwolves':4, 'Los Angeles Lakers':5,'Los
Angeles Clippers':5,'Sacramento Kings':5,'Phoenix Suns':5,'Golden State Warriors':5, 'Houston Rockets':6,'Dallas Mavericks':6,'Memphis Grizzlies':6,'
New Orleans Pelicans':6,'San Antonio Spurs':6}
```

Figura 13. Diccionari que conté la divisió a la que pertany cada equip.

3.2.5. Volem classificar els equips en funció de si juguen a la conferència Est o Oest

La següent funció ens permetrà extreure un paràmetre que indicarà si cada un dels equips juga a la conferència Est o Oest.

Primer de tot cal indicar què són les conferències. La NBA està dividida en dos conferències formades per 15 equips cada una, per tant hi ha 2 conferències de 15 equips cada una. Aquestes conferències es fan per proximitat geogràfica ja que Estats Units és un país enorme. Per evitar fer tants viatges llargs els equips de cada conferència juguen més partits entre ells que amb els altres equips de la lliga. Cada equip juga 4 partits contra els de la mateixa conferència i 2 contra els de la altre conferència.

Per poder realitzar aquesta classificació s'ha creat un diccionari que conté com a clau el nom de la conferència i com a valor associat el nom de els equips que hi pertanyen.

Crearem el diccionari codi que conté una codificació de 0 o 1 per cada una de les conferències (conferència Oest =1 i conferència Est=0). També una array conf_visitor i conf_local on guardarem els valors.

```
Conferences={'Eastern Conference':['Toronto Raptors', 'Boston Celtics', 'Philadelphia 76ers', 'Brooklyn Nets', 'New York Knicks','Milwaukee Bucks',
'Indiana Pacers', 'Chicago Bulls','Detroit Pistons', 'Cleveland Cavaliers','Miami Heat', 'Orlando Magic','Washington Wizards','Charlotte
Hornets','Atlanta Hawks'],'Western Conference':['Denver Nuggets','Utah Jazz','Oklahoma City Thunder','Portland Trail Blazers', 'Minnesota
Timberwolves','Los Angeles Lakers','Los Angeles Clippers','Sacramento Kings','Phoenix Suns','Golden State Warriors','Houston Rockets','Dallas
Mavericks','Memphis Grizzlies','New Orleans Pelicans','San Antonio Spurs']}
```

Figura 14. Diccionari que conté la conferència a la que pertany cada equip.

3.2.6. Percentatge de victòries en els últims N partits (%W)

La següent funció ens permetrà extreure el percentatge de victòries en els últims N partits de l'equip local i l'equip visitant just abans de començar el partit en qüestió.

A diferència de les dues altres dades que utilitzen el percentatge de victòries aquesta funció només calcula el percentatge en els últims N partits. Per exemple en el cas que N=10 només tindrem en compte els resultats dels últims 10 partits de cada equip.

El motiu de buscar aquest paràmetre és que permet tenir en compte el moment de forma de cada equip en els últims partits. Ja que si un equip per exemple guanya molts partits al principi de temporada i llavors en perd molts el percentatge de victòries pot resultar enganyós per l'estat de forma actual de l'equip.

Per tal de obtenir aquest percentatge de victòries en els últims N partits haurem de crear els següent diccionari anomenat last_N_games. També crearem dos arrays buits on emmagatzemarem els percentatges de victòries actualitzats i els afegirem al dataframe (last_N_games_visitor, last_N_games_home).

Aquest diccionari contindrà com a clau el nom de l'equip de l'NBA en qüestió i com a valor associat a la clau una llista amb N strings que seran 'L' o 'W' en funció de si l'equip ha guanyat o perdut l'últim partit. Un diccionari per N=10 seria del següent format.

```
{'Los Angeles Lakers': ['W', 'W', 'W', 'L', 'W', 'W', 'W', 'W', 'L']}
```

Figura 15. Diccionari "last_N_games"

No obstant durant els primers partits de la temporada utilitzarem el percentatge de victòries corresponents a la temporada anterior que hem guardat al diccionari wins_last_year.

3.2.7. Càlcul del Net Rating

La següent funció ens permetrà extreure el paràmetre anomenat Net Rating, aquest paràmetre és el valor que té en compte la diferència entre punts a favor i en contra de cada equip dividit per el nombre de partits totals disputats. Això permet veure si un equip sol guanyar o perdre per molts o pocs punts de diferència.

Per tal de obtenir aquesta diferència mitjana de punts per partit crearem el següent diccionari anomenat net_rat={}. També crearem dos arrays buits on emmagatzemarem el

net rating tant per l'equip local com per l'equip visitant (net_rat_visitor, net_rat_local).

El diccionari tindrà com a clau el nom de l'equip de l'NBA en qüestió i com a valor associat a la clau una llista de dos valors el primer contindrà la resta de punts a favor menys punts en contra, el segon valor serà el nombre de partits disputats, a continuació en veiem un exemple:

{New York Knicks': [-417, 66]}

Figura 16. Diccionari "net_rat"

3.2.8. Càlcul de si s'ha jugat pròrroga a l'últim partit disputat

La següent funció ens permetrà extreure un paràmetre que indicarà si l'equip en qüestió ha disputat una pròrroga a l'últim partit o no.

Primer de tot cal indicar que els partits a la NBA duren 48 minuts i que en cas d'empat les pròrrogues són de 5 minuts extres. En cas de pròrroga això suposa un desgast extra per l'equip en qüestió que pot afectar al seu rendiment al partit següent.

Per poder obtenir aquest paràmetre s'ha creat un diccionari anomenat overtime que conté com a clau el nom de l'equip i com a valor associat un 0 si no ha disputat pròrroga l'últim partit i un 1 si l'ha disputat.

{'Toronto Raptors': 0, 'New Orleans Pelicans': 0, 'Los Angeles Clippers': 0, 'Los Angeles Lakers': 0, 'Charlotte Hornets': 0, 'Chicago Bulls': 0, 'Indiana Pacers': 0, 'Detroit Pistons': 0, 'Orlando Magic': 0, 'Cleveland Cavaliers': 0, 'Brooklyn Nets': 0, 'Minnesota Timberwolves': 0, 'Miami Heat': 0, 'Memphis Grizzlies': 0, 'Philadelphia 76ers': 0, 'Boston Celtics': 0, 'Dallas Mavericks': 0, 'Washington Wizards': 0, 'San Antonio Spurs': 0, 'New York Knicks': 1, 'Utah Jazz': 0, 'Oklahoma City Thunder': 0, 'Phoenix Suns': 0, 'Sacramento Kings': 0, 'Portland Trail Blazers': 0, 'Denver Nuggets': 0, 'Atlanta Hawks': 1, 'Houston Rockets': 0, 'Milwaukee Bucks': 0, 'Golden State Warriors': 0}

Figura 17 Diccionari "overtime"

3.2.9. Absència de jugadors estrell de la lliga en un partit

A continuació s'expliquen el conjunt de funcions que permetran obtenir aquesta variable.

3.2.9.1. Jugadors que són estrelles de la NBA

A la NBA hi ha jugadors que tenen influència important en els partits i que la seva simple presència en el partit o no pot canviar les possibilitats d'un equip de guanyar un partit. Aquests jugadors s'han escollit a partir de una llista els 50 millors jugadors de la lliga que es fa al principi de cada temporada [2].

Aquesta funció seleccionarà entre el Dataframe que conté tots els jugadors que han disputat la temporada els que tenen un 1 en la columna STAR i els emmagatzema en un diccionari.

Crearem un diccionari anomenat STARS que contindrà com a clau el nom dels jugadors que són considerats entre els 50 millors a l'inici de la temporada i com a valor associat un 1.

{'James Harden': 1, 'Giannis Antetokounmpo': 1, 'LeBron James': 1, 'Nikola Jokic': 1, 'Anthony Davis': 1, 'Damian Lillard': 1, 'Luka Doncic': 1, 'Kawhi Leonard': 1, 'Jimmy Butler': 1, 'Chris Paul': 1, 'Rudy Gobert': 1, 'Jayson Tatum': 1, 'Ben Simmons': 1, 'Karl-Anthony Towns': 1, 'Khris Middleton': 1, 'Nikola Vucevic': 1, 'Kemba Walker': 1, 'Bradley Beal': 1, 'Kyle Lowry': 1, 'Joel Embiid': 1, 'Danilo Gallinari': 1, 'Devin Booker': 1, 'Jrue Holiday': 1, 'Donovan Mitchell': 1, 'Steven Adams': 1, 'Paul George': 1, 'Al Horford': 1, 'Pascal Siakam': 1, 'DeMar DeRozan': 1, 'Kevin Love': 1, 'Russell Westbrook': 1, 'Eric Bledsoe': 1, 'Kyrie Irving': 1, 'CJ McCollum': 1, 'LaMarcus Aldridge': 1, 'Tobias Harris': 1, 'Andre Drummond': 1, "De'Aaron Fox": 1, 'Jamal Murray': 1, 'Kristaps Porzingis': 1, "D'Angelo Russell": 1, 'Marc Gasol': 1, 'Myles Turner': 1, 'Paul Millsap': 1, 'Mike Conley': 1, 'Draymond Green': 1, 'Stephen Curry': 1, 'Victor Oladipo': 1, 'Gary Harris': 1, 'Blake Griffin': 1}

Figura 18. Diccionari "STARS"

3.2.9.2. Partits disputats per els jugadors que són estrelles de la NBA

A la NBA hi ha jugadors que tenen influència important en els partits i que la seva simple presència en el partit o no pot canviar les possibilitats d'un equip de guanyar un partit. Per tal de saber quins partits han jugat i quins han sigut baixa realitzarem la següent funció.

Aquesta funció rep com a paràmetre el diccionari creat a la funció anterior. El que farem serà crear un diccionari que contingui com a clau el nom del jugador i com a valor associat una llista amb la data de tots els partits disputats per cada jugador durant la temporada. Aquesta dada la traurem de les estadístiques individuals de cada jugador per cada partit.

Crearem un diccionari anomenat games_players on guardarem les dades mencionades anteriorment. Així per cada un dels 50 jugadors que són estrelles.

{'James Harden': [Timestamp('2020-03-10 00:00:00'), Timestamp('2020-03-08 00:00:00'), Timestamp('2020-03-07 00:00:00'), Timestamp('2020-03-05 00:00:00'), Timestamp('2020-03-02 00:00:00'), Timestamp('2020-02-29 00:00:00'), Timestamp('2020-02-26 00:00:00'), Timestamp('2020-02-24 00:00:00'), Timestamp('2020-02-22 00:00:00'), Timestamp('2020-02-20 00:00:00'), Timestamp('2020-02-11 00:00:00'), Timestamp('2020-02-09 00:00:00'), Timestamp('2020-02-07 00:00:00'), Timestamp('2020-02-06 00:00:00'), Timestamp('2020-02-04 00:00:00'), Timestamp('2020-02-02 00:00:00'), Timestamp('2020-01-31 00:00:00'), Timestamp('2020-01-29 00:00:00'), Timestamp('2020-01-24 00:00:00'), Timestamp('2020-01-22 00:00:00'), Timestamp('2020-01-20 00:00:00'), Timestamp('2020-01-18 00:00:00'), Timestamp('2020-01-15 00:00:00'), Timestamp('2020-01-14 00:00:00'), Timestamp('2020-01-11 00:00:00'), Timestamp('2020-01-09 00:00:00'), Timestamp('2020-01-08 00:00:00'), Timestamp('2020-01-03 00:00:00'), Timestamp('2019-12-31 00:00:00'), Timestamp('2019-12-28 00:00:00'), Timestamp('2019-12-25 00:00:00'), Timestamp('2019-12-23)], 'Giannis Antetokounmpo': [Timestamp('2020-03-06 00:00:00'), Timestamp('2020-03-04 00:00:00'), etc..

Figura 19. Diccionari "games_players"

3.2.9.3. Diccionari que conté els jugadors estrella i el seu equip

Ens cal assignar també cada jugador al equip al que pertany. Amb aquest objectiu volem obtenir un diccionari que tingui com a clau el nom del jugador i com a valor el nom de l'equip on juga.

No obstant a la NBA es poden realitzar fitxatges durant la temporada el que suposa un canvi d'equip. En cas que això es produeixi ho emmagatzemarem amb una llista associada a la clau que contindrà l'equip que pertanyia abans del fitxatge, l'equip pel que ha fitxat i la data en la que s'ha produït el traspàs.

{"D'Angelo Russell": ['MIN', 'GSW', 'FEB 05, 2020']}

Figura 20. Exemple de diccionari que conté un traspàs de jugador

Per obtenir aquest diccionari ho farem de la següent manera:

Crearem un diccionari anomenat players_teams on guardarem les dades mencionades anteriorment.

```
{'James Harden': ['HOU'], 'Giannis Antetokounmpo': ['MIL'], 'LeBron James': ['LAL'], 'Nikola Jokic':
['DEN'], 'Anthony Davis': ['LAL'], 'Damian Lillard': ['POR'], 'Luka Doncic': ['DAL'], 'Kawhi Leonard':
['LAC'], 'Jimmy Butler': ['MIA'], 'Chris Paul': ['OKC'], 'Rudy Gobert': ['UTA'], 'Jayson Tatum': ['BOS'],
'Ben Simmons': ['PHI'], 'Karl-Anthony Towns': ['MIN'], 'Khris Middleton': ['MIL'], 'Nikola Vucevic':
['ORL'], 'Kemba Walker': ['BOS'], 'Bradley Beal': ['WAS'], 'Kyle Lowry': ['TOR'], 'Joel Embiid': ['PHI'],
'Danilo Gallinari': ['OKC'], 'Devin Booker': ['PHO'], 'Jrue Holiday': ['NOP'], 'Donovan Mitchell': ['UTA'],
'Steven Adams': ['OKC'], 'Paul George': ['LAC'], 'Al Horford': ['PHI'], 'Pascal Siakam': ['TOR'], 'DeMar
DeRozan': ['SAS'], 'Kevin Love': ['CLE'], 'Russell Westbrook': ['HOU'], 'Eric Bledsoe': ['MIL'], 'Kyrie
Irving': ['BRK'], 'CJ McCollum': ['POR'], 'LaMarcus Aldridge': ['SAS'], 'Tobias Harris': ['PHI'], 'Andre
Drummond': ['CLE', 'DET', Timestamp('2020-02-05 00:00:00')], "De'Aaron Fox": ['SAC'], 'Jamal Murray':
['DEN'], 'Kristaps Porzingis': ['DAL'], "D'Angelo Russell": ['MIN', 'GSW', Timestamp('2020-02-05
00:00:00')], 'Marc Gasol': ['TOR'], 'Myles Turner': ['IND'], 'Paul Millsap': ['DEN'], 'Mike Conley':
['UTA'], 'Draymond Green': ['GSW'], 'Stephen Curry': ['GSW'], 'Victor Oladipo': ['IND'], 'Gary Harris':
['DEN'], 'Blake Griffin': ['DET']}
```

Figura 21. Diccionari "players_teams"

3.2.9.4. Paràmetre d'absència de les estrelles de l'equip

La següent funció ens permetrà extreure el paràmetre d'absència de les estrelles dels dos equips que disputen el partit. Totes les funcions anteriors estan orientades a poder obtenir aquest paràmetre. Bàsicament indicarà si algun dels jugadors que pertanyen al top 50 no ha disputat el partit en qüestió. Això ens servirà per analitzar amb detall l'efecte de les absències dels jugadors considerats estrelles en cada partit.

Per tal d'obtenir aquest paràmetre crearem dos arrays buits on emmagatzemarem les baixes de les estrelles tant per l'equip local com per l'equip visitant (stars_inj_visitor, stars_inj_local).

3.2.10. Ratxa de N victòries o derrotes

La següent funció ens permetrà extreure la ratxa de victòries en els últims n partits de l'equip local i l'equip visitant just abans de començar el partit en qüestió.

Aquesta funció calcula la ratxa de partits guanyats o perduts consecutius. Per exemple en el cas que n=10 només tindrem en compte només les ratxes de 10 o més victòries o derrotes. És a dir si un equip ha guanyat 11 partits seguits l'equip rebrà un paràmetre 11 si un equip ha guanyat 4 seguits el valor serà 0 ja que està per sota del llindar fixat per N i si un equip ha perdut 12 consecutius el paràmetres serà -12 amb signe negatiu per indicar que es tracta d'una derrota.

El motiu de buscar aquest paràmetre és que permet tenir en compte el moment de forma de cada equip ja que cal tenir en compte si un equip porta una mala o bona dinàmica en els últims partits. Aquest fet pot tenir una influència important en el resultat i cal tenir-la en compte. Una altre cosa que s'ha de tenir en compte és a partir de quants N partits es considera una ratxa.

Per tal d'obtenir aquesta ratxa de victòries haurem de crear els següent diccionari anomenat streak_N_games. També crearem dos arrays buits on emmagatzemarem els valors de les ratxes i els afegirem al dataframe (streak_N_visitor, streak_N_home).

Aquest diccionari contindrà com a clau el nom de l'equip de l'NBA en questió i com a valor associat a la clau una llista amb N strings que seran 'L' o 'W' en funció de si l'equip ha guanyat o perdut. Un diccionari per N=30 seria del següent format.

{'New Orleans Pelicans': ['W', 'L', 'W', 'W', 'L', 'W', 'W', 'L', 'W', 'L', 'L', 'W', 'W', 'W', 'L', 'L', 'W', 'W', 'W', 'L', 'W', 'W', 'L', 'W', 'L', 'L', 'L', 'W', 'W']

Figura 22. Exemple diccionari "streak_N_games"

3.2.11. Distància recorreguda per cada equip

Aquesta funció té com a objectiu calcular la distància en Km recorreguda per cada equip anteriorment al partit analitzat. D'aquesta manera som capaços de monitoritzar quin dels dos equips ha recorregut menys Km i per tant arriba menys fatigat al partit. Pot ser un factor rellevant i tenir un efecte similar al de la funció del Back to Back que hem fet anteriorment.

Cal destacar que Estats Units és un país enorme amb grans distàncies entre les ciutats on es disputen els partits per exemple la distància entre Los Àngeles i Boston és de 4800 km. Per tant al disputar tants partits en pocs dies els equips passen una part important del seu

temps de descans a l'avió. Per tots aquests motius s'ha realitzat la funció següent.

Crearem 3 diccionaris un es dirà travels i contindrà les distàncies que ja hem obtingut anteriorment al Dataframe de distàncies entre ciutats. La clau serà la ciutat origen i la ciutat destí i el valor associat un float que contindrà la distància en Km. El diccionari last_match guardarà la ciutat on s'hagi disputat l'últim partit per cada equip i el diccionari kilometers guardarà les distàncies acumulades totals per cada equip. També crearem com sempre dos arrays per guardar els km de l'equip local i visitant.

3.2.12. Paràmetre que calcula la mitjana d'edat dels jugadors de cada equip

Aquesta variable ens retornarà la mitjana d'edat de tots els jugadors de la plantilla de cada equip. D'aquesta manera es podrà veure si afecta la edat mitjana dels jugadors de cada equip al resultat dels partits, és a dir si per exemple un equip té jugadors joves i poc experimentats o jugadors molt vells pot ser un factor que afecti al rendiment de l'equip i per tant en els resultats dels partits.

3.2.13. Paràmetre que indica l'equip guanyador de cada partit

Aquesta funció ens retorna el paràmetre que indicarà si el partit l'ha guanyat l'equip local, d'aquesta manera es retornarà un 1 i si guanya l'equip visitant un 0.

Crearem una array 'Winner' on guardarem els valors del guanyador de cada partit.

Finalment per acabar aquesta fase cal remarcar que en cada partit analitzat tenim el valor de les variables, tant per l'equip local com per l'equip visitant. Però al final ens quedem amb un valor només, que és resultat del valor de la variable de l'equip local menys el resultat de la variable de l'equip visitant. La motivació principal és reduir el nombre de variables a analitzar ja que aquesta variable que resulta de la resta de les altres dues ja conté tota la informació que se'n vol extreure.

La taula de dades que s'obté després de extreure totes les variables és la següent:

Visitor	PTSV	Home	PTSH	Overtime	Attend.	Back to back	Div_Visitor	Div_Home	Division	Conference_Visitor	Conference_Home	Conference	Last N	Net Rating	OT last match	Rest/Injuries Visitor Stars	Rest/Injuries Home Stars	Rest/Injuries Stars	Win Streak	Km	Age_visitor	Age_home	Age	Winner
New Orleans Pelicans	122	Toronto Raptors	130	1	20,787	0.00	6	1	-5	0	1	1	0.31	0.000	0.00	0.00	0.00	0.00	0.00	-1789.00	24.875000	25.470588	0.595588	1
Los Angeles Lakers	102	Los Angeles Clippers	112	0	19,068	0.00	5	5	0	0	0	0	0.14	0.000	0.00	0.00	1.00	1.00	0.00	0.00	28.357143	26.357143	-2.000000	1
Chicago Bulls	125	Charlotte Hornets	126	0	15,424	0.00	2	3	1	1	1	0	0.21	0.000	0.00	0.00	0.00	0.00	0.00	-947.10	24.176471	24.307692	0.131222	1
Detroit Pistons	119	Indiana Pacers	110	0	17,923	0.00	2	2	0	1	1	0	0.09	0.000	0.00	1.00	1.00	0.00	0.00	-403.40	24.571429	25.125000	0.553571	0
Cleveland Cavaliers	85	Orlando Magic	94	0	18,846	0.00	2	3	1	1	1	0	0.28	0.000	0.00	0.00	0.00	0.00	0.00	-1436.00	24.307692	25.882353	1.574661	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Brooklyn Nets	104	Los Angeles Lakers	102	0	18,997	0.00	1	5	4	1	0	-1	0.45	8.400	0.00	1.00	0.00	-1.00	0.00	-1688.56	25.526316	28.357143	2.830827	0
Detroit Pistons	106	Philadelphia 76ers	124	0	20,172	1.00	2	1	-1	1	1	0	0.33	5.230	0.00	1.00	1.00	0.00	0.00	4296.74	24.571429	25.666667	1.095238	1
New York Knicks	136	Atlanta Hawks	131	1	15,393	1.00	1	3	2	1	1	0	0.11	-1.410	1.00	0.00	0.00	0.00	0.00	229.58	24.533333	25.142857	0.609524	0
Charlotte Hornets	109	Miami Heat	98	0	19,6	1.00	3	3	0	1	1	0	0.34	10.210	-1.00	0.00	1.00	1.00	0.00	1506.34	24.307692	26.294118	1.986425	0
Denver Nuggets	97	Dallas Mavericks	113	0	20,302	-1.00	4	6	2	0	0	0	0.00	2.650	0.00	0.00	1.00	1.00	0.00	-4856.11	25.583333	27.000000	1.416667	1

4. Fase III. Modeling.

En aquesta fase, es seleccionen y s'utilitzen les tècniques de modelatge que siguin pertinents al problema i es calibren els seus paràmetres a valors òptims. Típicament hi ha diverses tècniques per el mateix problema de mineria de dades. Algunes tècniques tenen requeriments específics sobre la forma de les dades. Per tant casi sempre en qualsevol projecte s'acaba tornant a la fase de preparació de dades.

De la fase anterior corresponent a la Data Preparation s'ha obtingut el data set que serà el que utilitzarem per realitzar l'anàlisi de de dades. Com ja sabem el data set conté per cada un dels partits de la temporada, el valor corresponent de les variables que s'han extret anteriorment en la fase de la preparació de dades.

Primer de tot abans de entrar en la fase de modelat cal definir una sèrie de algoritmes i conceptes, per poder comprendre el procés que es farà posteriorment.

4.1. Algoritmes a utilitzar en el modelatge

Un cop tenim les variables que utilitzarem per fer les prediccions definirem els diferents algoritmes que s'utilitzaran. És important provar-ne de diferents ja que cadascun funciona millor amb diversos tipus de dades i per tant amb el mateix data set podem obtenir millors resultats amb un algoritme en comparació amb altres.

El primer de tots que s'explica a continuació és l'anomenada Logistic Regression.

4.1.1. Logistic Regression

Normalment la regressió lineal és efectiva per estimar variables continues, per exemple estimar el preu de les vivendes, però no és la millor eina per predir la classe o classificar un conjunt de dades. Amb l'objectiu d'estimar cada conjunt de dades necessitem ajuda per determinar la classe més probable per el conjunt de dades analitzat. Per realitzar aquesta classificació s'utilitza l'anomenada Logistic Regression [3].

Com és sabut la regressió lineal busca una funció que relacioni una variable dependent continua Y amb variables predictores (variables independents, x1, x2, etc.). Per exemple la regressió lineal simple assumeix la següent forma.

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots \quad \textit{(Equació 1)}$$

I troba els valors dels paràmetres θ_0, θ_1, θ_2, etc. On el terme θ_0 es l'ordenada a l'origen i pot ser generalment ensenyat com:

$$h_0(x) = \theta^T X \qquad \textit{(Equació 2)}$$

La Logistic Regression és una variació de la regressió lineal, la diferència principal entre les dues és que la variable depenent observada es categòrica. Aquesta regressió produeix una fórmula que preveu la probabilitat de que pertanyi a una classe categòrica o a una altre en funció de les variables dependents.

La Logistic Regression bé determinada per una corba especial "s-sharped", tot agafant la regressió lineal i transformant el valor numèric estimat, en una probabilitat amb la següent funció, que s'anomena, funció sigmoide.

$$h_0(x) = \theta^T X = \frac{e^{(\theta_0+\theta_1 x_1+\theta_2 x_2+\cdots)}}{1+e^{(\theta_0+\theta_1 x_1+\theta_2 x_2+\cdots)}} \qquad \textit{(Equació 3)}$$

També ho podem expressar com:

$$Probabilitat\ d'una\ classe = P(Y=1) = \sigma(\theta^T X) = \frac{e^{(\theta^T X)}}{1+e^{(\theta^T X)}}$$

(Equació 4)

En aquesta equació $\theta^T X$ és el resultat de la regressió (la suma de les variables balancejades amb els coeficients) i exp és la funció exponencial.

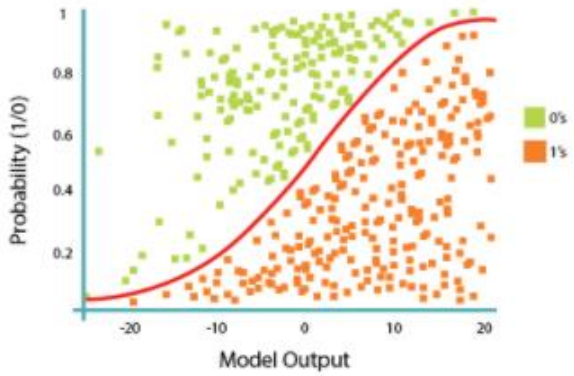


Figura 23. Exemple visual de com classifica la Logistic Regression.

En resum, la Logistic Regression converteix el input a través de la funció sigmoide en una probabilitat que un conjunt de dades pertanyi a una classe categòrica concreta.

L'objectiu de l'algoritme de la Logistic Regression, consisteix en trobar els paràmetres θ, de forma que el model sigui capaç de predir en cada cas la seva classe categòrica de la millor manera possible.

4.1.2. Anàlisi a partir de l'algoritme de classificació SVM

L'altre algoritme que s'utilitzarà són les Màquines de Suport Vectorial (Support Vector Machines o SVM's). Són un conjunt d'algoritmes d'aprenentatge supervisat que desenvolupen mètodes relacionats amb els problemes de classificació i regressió.

Com en la majoria dels mètodes de classificació supervisada, les dades d'entrada són vistos com un vector p-dimensional (una llista de p números). Donat un conjunt de punts com un subconjunt de un conjunt major (espai), en el que cada un d'ells pertany a una de les dues possibles categories, de manera que un algoritme basat en SVM construeix un model capaç de predir si un punt nou (la seva categoria desconeixem) pertany a una categoria o la altre.

La SVM, intuïtivament, és un model que partint d'un conjunt d'exemples de entrenament, podem etiquetar-lo en diferents classes i representar les mostres en punts en l'espai per tractar de separar les diferents classes mitjançant un espai el més ampli possible, perquè quan les noves mostres dels casos del test es posin amb correspondència amb el model puguin ser classificades correctament en funció de la seva proximitat.

En aquest concepte de separació òptima és on resideix la característica fonamental de les SVM: aquest tipus d'algoritmes busquen l'hiperplà que tingui la màxima distancia amb el punts que estiguin més a prop de ell mateix. Per això també a vegades se les coneix a les SVM com classificadors de marge màxim. D'aquesta manera els punts del vector que són etiquetats amb una categoria estaran a un costat de l'hiperplà i els casos que es troben en l'altre categoria estaran al altre costat.

La manera més simple de realitzar al separació és mitjançant una línia recta, un pla recte o un hiperplà N-dimensional, però els universos a classificar no es solen presentar en l'ideal de les dues dimensions, sinó que un algoritme SVM ha de tractar amb més de dues variables predictores, corbes no lineals de separació, casos on els conjunts de dades no poden ser completament separats, classificacions en més de dues categories.

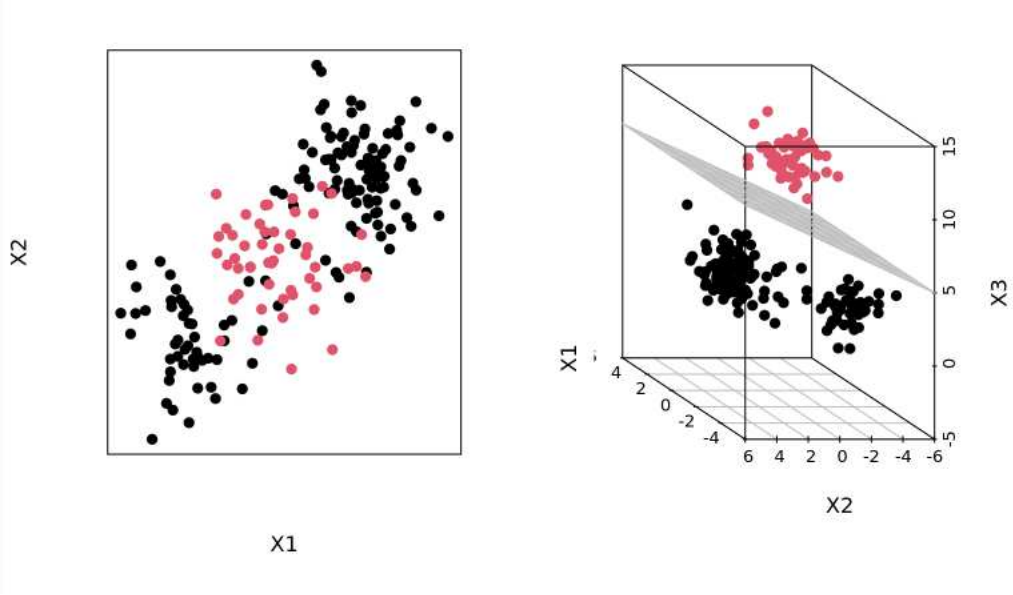


Figura 24. Visualització separació de punts en algoritme SVM.

La representació per mitja de funcions nucli o Kernel ofereix una solució a aquest problema, projectant la informació a un espai de característiques de major dimensió per tant un Kernel aplica una transformació a les dades cap a un altre espai a on les categories son linealment separables.

4.2. Mètriques d'avaluació de resultats

Per tal d'analitzar els resultats cal primer de tot definir algunes de les mètriques que s'utilitzaran per avaluar-los.

4.2.1. Confusion matrix

La Confusion Matrix serveix per avaluar la precisió de la classificació donada per el model. Per definició la Confusion Matrix C és tal que Cij és igual al nombre de observacions que estan al grup i en canvi s'ha predit que estarien en el grup j.

Aquesta classificació binaria el recompte de negatius certs es C_{00}, negatius falsos és C_{10}, positiu cert és C_{11}, positiu fals és C_{01} [2].

4.2.2. Precision

Primer de tot analitzarem la mètrica anomenada Precision. Com es pot intuir la mètrica Precision, ens dona informació sobre la precisió del model, per exemple, quants dels positius predits, són realment positius.

La mètrica precision és útil per determinar quan el cost de un falç positiu és alt. Per exemple, detecció d'emails spam. Ja que un fals positiu voldria dir que un email que no que no fos spam (actual negative) és identificat com spam i que per tant es perdrien e-mails importants si la precisió del model de detecció de spam no és molt elevada.

4.2.3. Recall

El paràmetre Recall es calcula de la següent manera:

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad \textit{(Equació 5)}$$

El paràmetre Recall representa quants del positius actuals captura el nostre model tot classificant-los com a positiu (Positiu certs). Seguint el mateix raonament sabem que el paràmetre Recall hauria de ser el model de mètrica que fem servir per seleccionar el nostre millor model quan hi ha un elevat cost associat al Fals negatiu.

Per exemple, en la detecció de frau o detecció de malalties. Si una transacció fraudulenta (Actual Positive) es prediu com a no fraudulenta (Negatiu predit) la conseqüència pot ser

molt dolenta pel banc.

De forma similar en detecció de malalties. Si el pacient malalt (Actual Positive) passa el test i es preveu com que no està malalt (Predicted Negative). El cost associat amb Falç negatiu serà extremament alt si la malaltia és contagiosa.

4.2.4. F1 Score

Un altre paràmetre d'anàlisis a tenir en compte és la mesura F1, és una funció de les altres dues (Precision i Recall). La fórmula per calcular F1 és la següent:

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} \qquad \textit{(Equació 6)}$$

El paràmetre F1 és necessari quan es vol obtenir un balanç entre Precision i Recall. Anteriorment hem vist que la precisió esta molt influenciada per un gran nombre de True Negatives i Recall per False Positive. Per tant F1 serà una millor mesura per si volem obtenir un equilibri entre Precission i Recall.

4.2.5. Log Loss

El paràmetre Log Loss és útil en la classificació basada en les probabilitats com és el cas de la Logisitic Regression. És difícil a vegades de interpretar però segueix essent una molt bona mètrica per comparar models. Per a qualsevol problema un valor baix de Log-loss significa bona capacitat de predicció.

El paràmetre Log Loss, utilitza l'anomenada Likelihood Function. De fet Log Loss és -1* el logaritme de la Likelihood function.

La Likelihood function, respon a la pregunta quant probable és el model de predir amb el menor cost d'error possible

Un model qualsevol prediu les següents probabilitats [0.8, 0.4, 0.1] per tres cases. Les dues primeres van ser venudes i l'altre no. Per tant els resultat final seria representat per els valors següents: [1, 1, 0].

Ara entrem a analitzar aquestes prediccions per veure el procediment del càlcul de la Likelihood function. La primera venda, i el model deia que la probabilitat de venda era del 80%, per tant la Likelihood funcition mirant només aquesta predicció val 0.8. La segona casa venuda diu que la probabilitat era del 40%. Hi ha una regla de probabilitat que diu que la

probabilitat total de múltiples variables independent es el producte de les seves probabilitats individuals. Per tant el valor de Likelihood combinat de les dues primeres previsions seria la següent. Serà 0.8 * 0.4, que dona 0.32.

Finalment anem a la tercera predicció. Que la casa no s'havia venut. El model deia que la probabilitat de vendre-la era del 10% i per tant de un 90% de no vendre-la. Per tant el valor que s'utilitzarà per calcular el valor de la Likelihood function serà 0.9. Per tant multipliquem els resultats anteriors 0.32 per 0.9.

Fent això per totes les prediccions del model acabarem obtenim el valor de la funció de Likelihood global.

4.2.6. De Likelihood a Log Loss

Cada predicció és un valor entre 0 i 1. Si es multipliquen suficients nombres en aquest rang, el resultat es torna tant petit que els ordinadors no poden seguir calculant-lo. Es per aquest motiu que per un motiu computacional utilitzem el logaritme del valor de likelihood. El fet de multiplicar-ho per -1 és degut a que volem mantenir la convenció de que els valors baixos de Log Loss són per bones prediccions.

4.3. Mètodes de validació de resultats

4.3.1. Over-fitting

Over-fitting és un problema molt comú en el Data Science i es produeix quan el nostre model aprèn les dades d'entrenament massa bé, i que per tant no es capaç de generalitzar Com a conseqüència d'això quan li arribin les noves dades, s'obtindran mals resultats. En el gràfic que apareix a continuació es pot veure de forma més clara en què consisteix.

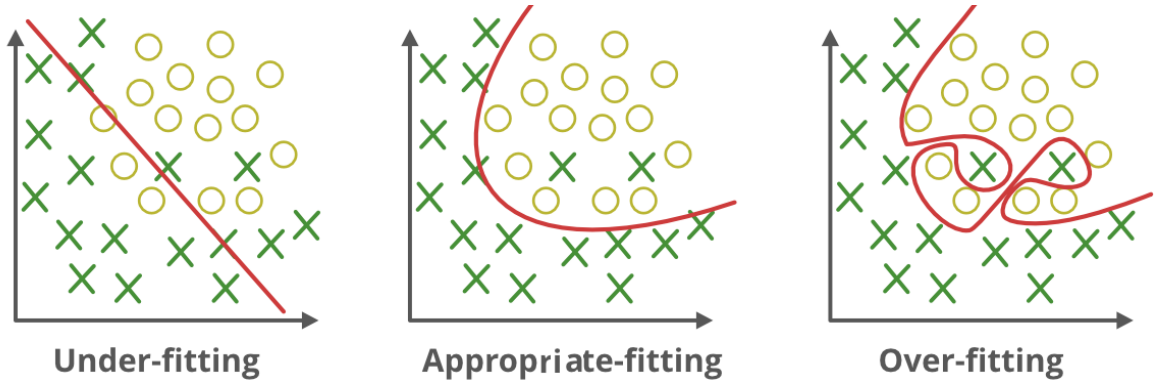


Figura 25. Representació gràfica del concepte de over-fitting

Per tal de evitar els problemes d'over-fitting, tenim les solucions següents:

- Dividir les dades entre entrenament, validació i testing.
- Obtenir un major número de dades.
- Ajustar els paràmetres dels models.
- Utilitzar models més simples.
- Que les dades vinguin de diferents distribucions.
- Baixar el nombre d'iteracions en els algoritmes iteratius.

Un cop s'ha definit el problema de 'over-fitting', cal explicar els diferents mètodes de validació que ens ajudaran a evitar aquest problema.

Primer de tot cal indicar que els mètodes de validació parteixen tots de dividir el conjunt que tenim de dades en dos grups. L'anomenat training set que són les dades que el model utilitzarà per aprendre i l'anomenat test set, que corresponen a la resta de dades que no s'han utilitzat pel training. Aquestes dades serveixen per comprovar si les prediccions que s'han fet, són correctes. És a dir el model s'entrena amb les dades del training set i realitza les prediccions amb les dades del test set. Finalment es comparen les prediccions realitzades a partir el test set amb els resultats reals.

4.3.2. Concepte de Hold out

El mètode de validació anomenat Hold out, consisteix en dividir en dos conjunts complementaris les dades de la mostra. Amb el primer subconjunt realitzar l'entrenament o training i en l'altre subconjunt prova o test validar la anàlisis.

La principal avantatge d'aquest mètode és que es molt ràpid, pel que fa al temps de càlcul. No obstant, aquest mètode no és gaire precís. Els resultats de la avaluació solen dependre massa de com s'ha fet la divisió entre dades de entrenament i test. Per tant poden ser significativament diferents en funció de com es realitzi aquesta divisió. Degut a aquestes deficiències es va crear la validació creuada, que és el següent mètode que s'explicarà.

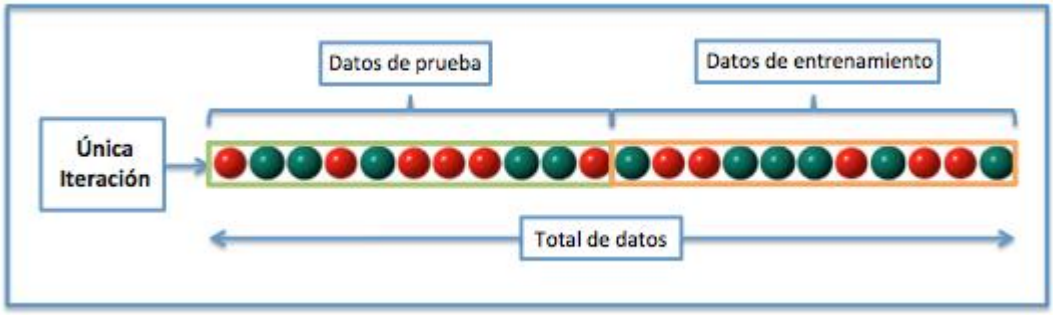


Figura 26. Exemple gràfic de Hold out.

4.3.3. Cross validation

Un dels mètodes més utilitzats per dividir les dades entre entrenament i test és la validació encreuada de K iteracions o K-fold cross-validation. Les dades de mostra es divideixen en K subconjunts. Un dels subconjunts s'utilitza com a dades de prova i la resta (K-1) com a dades d'entrenament. El procés de validació encreuada és repetit durant k iteracions, amb cada un dels possibles subconjunts de dades de prova.

Finalment es realitza la mitjana aritmètica dels resultats de cada iteració per a obtenir un únic resultat. Aquests mètode és molt precís, ja que avaluem a partir de K combinacions de dades d'entrenament i de prova, però tot i així té un desavantatge, i és que, és lent des del punt de vista computacional. A la pràctica, l'elecció del nombre d'iteracions depèn de la mida del conjunt de dades. El més comú és utilitzar la validació encreuada de 10 iteracions (10-fold cross-validation).

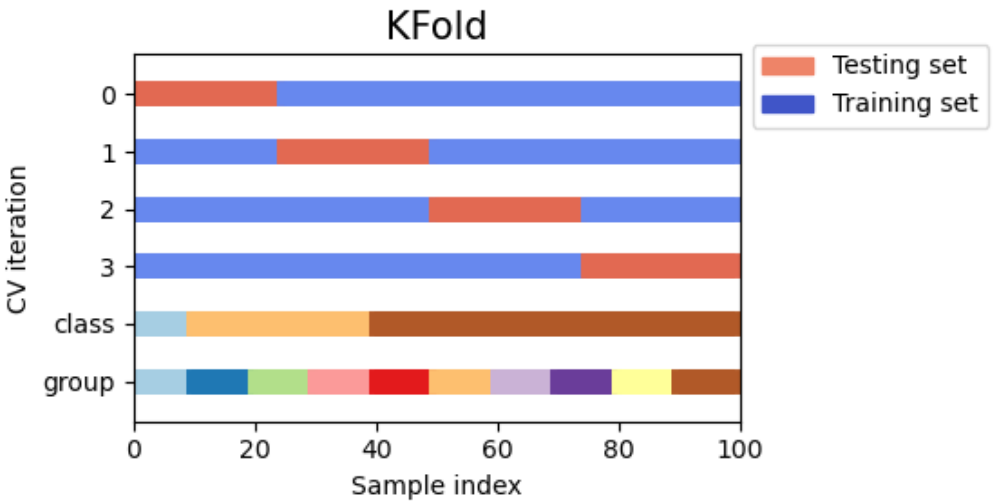


Figura 27. Exemple visual de la Cross Validaton KFold

4.3.4. Time Series Split

Com que les dades que tenim contenen dates en forma de temps, la Cross Validation aplicada en series temporals serà un tipus de validació de resultats molt útil.

Les series temporals de dades estan caracteritzada amb la correlació entre observacions que estan properes en el temps (autocorrelació). No obstant les tècniques clàssiques de Cross Validation com poden ser la KFold explicada anteriorment assumeixen que les dades són independents i distribuïdes idènticament. I resultarien en una correlació errònia entre el training i el test (provocant males estimacions i generalització d'errors) en una time series data.

Per aquest motiu és molt important avaluar el nostre model de time series data per utilitzar d'una forma més adequada les dades més recents. I veure com aprèn el model al llarg de la temporada . Per aconseguir això existeix el mètode Time Series Split.

Time Series Split és una variació de el mètode K fold. Cal tenir en compte que a diferència de la Cross Validation, els training sets són supersets que venen abans de aquests. A més a més afegeix un extra de dades a la primera partició que es fa servir sempre per entrenar el model.

Aquesta tipus de divisió s'utilitza per analitzar dades que poden estar afectades per esdeveniments temporals i que per tant en funció de les dates podem esperar un resultats diferents.

A continuació apareix una visualització del comportament de la Time Series Split que s'ha explicat.

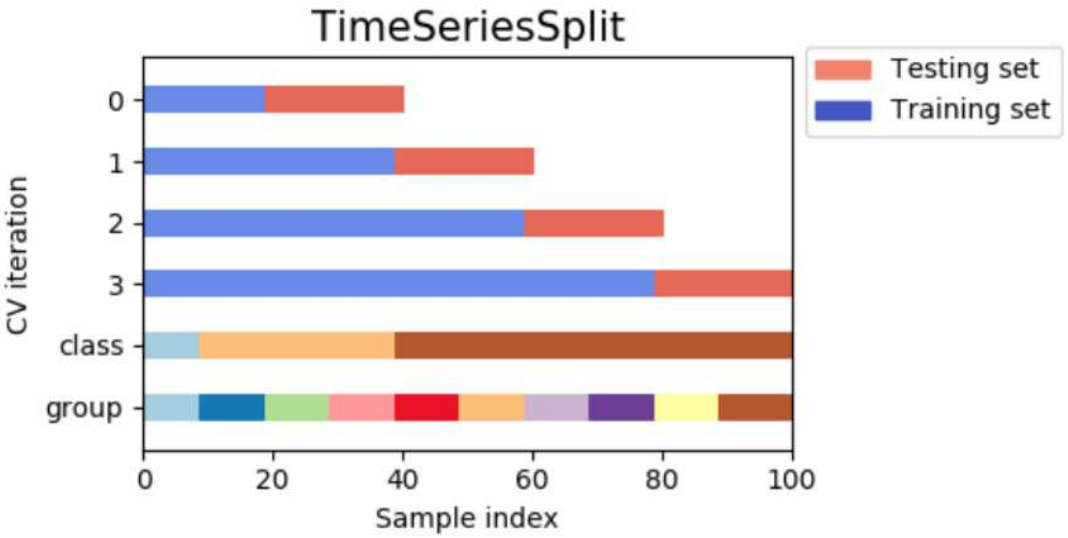


Figura 28. Exemple visual de la Cross Validaton Time Series Split

4.3.5. Random permutation cross validation (Shuffle & Split)

Aquest mètode de validació és molt simple. Consisteix en que en cada iteració de la Cross Validation, la part de les dades que pertanyen al train set i al test set, es decideix a partir de números aleatoris. És a dir no es segueix ni cap lògica temporal ni cap ordre. En el gràfic que tenim a continuació es pot apreciar clarament com funciona.

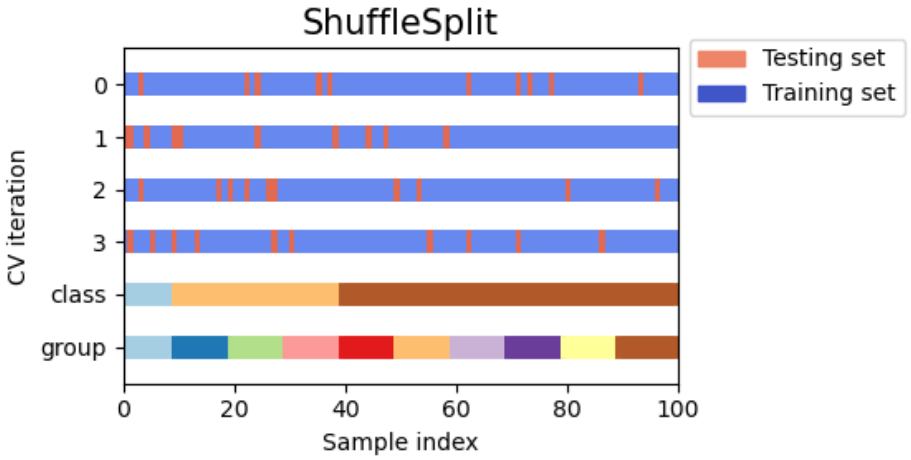


Figura 29. Exemple visual de la Random Cross Validaton

Aquest tipus de divisió ens serà útil per poder saber si el nostre model està realment afectat per la conjuntura temporal o la precisió d'encert és constant al llarg de la temporada. Ja que al realitzar les prediccions sense cap ordre cronològic ens porta a suposar que l'ordre en què es juguen els partits no és rellevant a l'hora de fer prediccions. No obstant en una aplicació no experimental, aquest mètode de avaluació no tindria sentit ja que estem utilitzant dades que per exemple a la meitat de la temporada no sabríem.

4.4. Procés de modelatge

Una vegada definits tots els conceptes necessaris es pot explicar ja el desenvolupament del model. (L'explicació detallada de totes les funcions i el codi que les fa possibles, es pot visualitzar a l'Annex 2. Allà estan explicades detalladament).

4.4.1. Anàlisis inicial del data set

Primer de tot analitzem la correlació entre variables mitjançant el següent mapa de calor, que ens proporciona la llibreria Seaborn de Python.

Aquest visualització té l'objectiu de realitzar una exploració inicial de dades per detectar si hi ha alguna forta correlació entre la variable Winner (que és la variable que volem predir) i alguna de les altres variables creades. Això podria indicar, que són variables que poden influir bastant en el resultat final d'un partit.

Es pot observar que les variables %W, Net Rating o Age tenen una forta correlació amb la variable Winner. Per tant cal esperar que siguin variables que ens ajudin a fer bones prediccions. En canvi n'hi ha d'altres que sembla que no ajudaran tant ja que la correlació que s'observa amb la variable Winner és propera a zero.

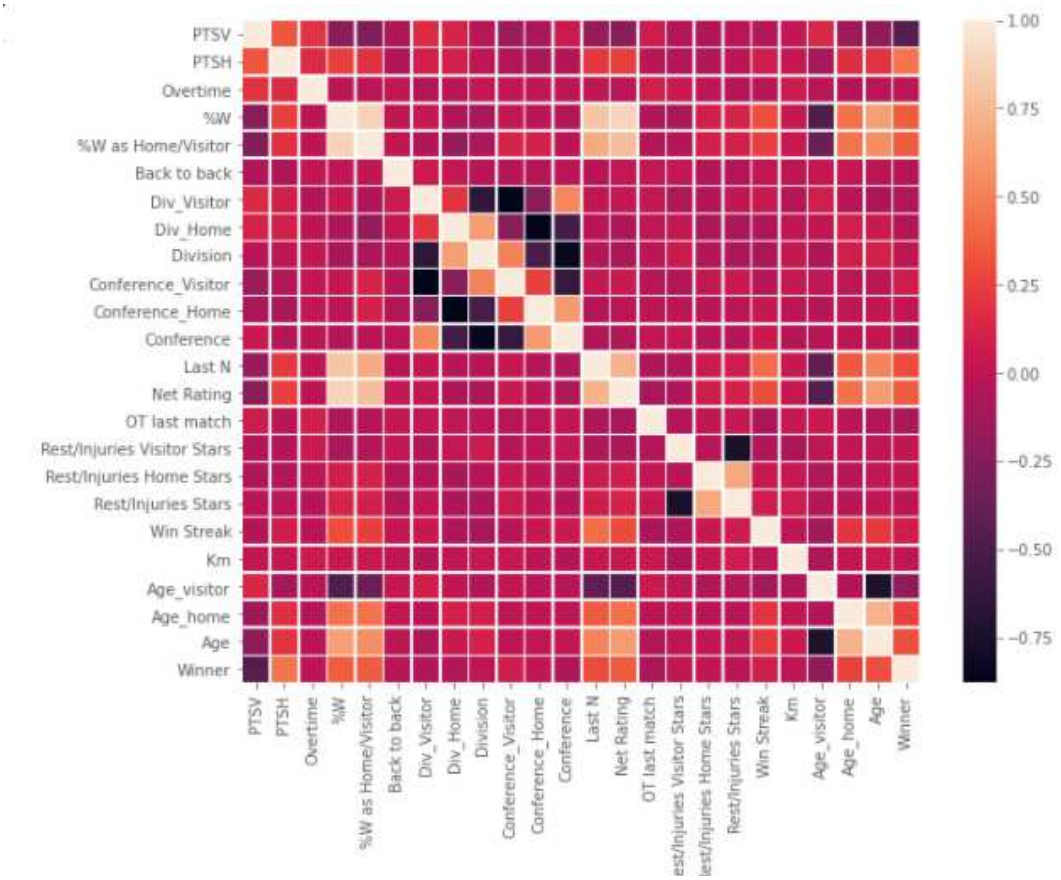


Figura 30. Correlació entre variables

4.4.2. Variables més rellevants

Primer de tot comencem amb el Data set que hem obtingut de la fase de Data Preparation. Aquest Dataset conté moltes variables per analitzar i per tant cal escollir quines són les variables que cal utilitzar per obtenir una predicció de resultats òptima. En proves preliminars de la fase de modelatge es va provar de introduir totes les variables al model. Els resultats eren clarament inferiors en termes de precisió que si s'introduïen una quantitat de variables més petita, és per això que primer de tot cal buscar les variables més rellevants, amb l'objectiu de que les que s'utilitzin donin el millor resultat possible.

Com s'ha vist el data set a analitzar disposa de un total de 20 variables. És un nombre bastant elevat de variables i per tal d'evitar els problemes de 'over-fitting' que s'han descrit anteriorment, caldrà reduir el nombre de variables que s'analitzaran per fer una bona predicció.

Primer de tot volem saber quines són les variables que ens poden proporcionar la informació més rellevant per tal de obtenir les prediccions. La següent eina de la llibreria Scikit Learn ens proporcionarà un ranking de les variables que poden proporcionar una millor predicció amb el mètode d'eliminació recursiva. D'aquest manera eliminem directament algunes variables que seran poc útils.

El mètode que utilitzarem és l'anomenada RFE que consisteix en un estimador extern que assigna pesos a les variables, l'objectiu de la Recursive Feature Elimination (RFE) és seleccionar variables, tot tenint en compte, cada cop grups més petits de variables, ja que deixem de tenir en compte les que anem eliminant. Primer l'estimador s'entrena amb el grup inicial de variables, la importància de cada variable s'obté mitjançant el coef_atribute o feature/importances atribute. Llavors les variables menys importants són purgades del conjunt de variables. Aquest procés es repeteix recursivament fins que el número de variables que queden coincideix amb nombre de variables que nosaltres volem. S'ha obtingut el següent rànquing.

	Features	Support	Ranking
0	%W	True	1
1	%W as Home/Visitor	False	2
11	OT last match	False	3
9	Last N	False	4
6	Conference_Visitor	False	5
17	Age	False	6
12	Rest/Injuries Visitor Stars	False	7
8	Conference	False	8
5	Division	False	9
7	Conference_Home	False	10
14	Rest/Injuries Stars	False	11
2	Back to back	False	12
3	Div_Visitor	False	13
4	Div_Home	False	14
10	Net Rating	False	15
15	Win Streak	False	16
13	Rest/Injuries Home Stars	False	17
16	Km	False	18

Figura 31. Exemple classificació variables donat per RFE

No obstant cal destacar que aquestes variables que seleccionem com a les més prometedores, no seran sempre les millors. Com que aquest procés es tracta d'una prova de concepte utilitzem aquesta simplificació per treure les variables més rellevants. Però les variables més adequades varien en funció de molts paràmetres, com per exemple l'algoritme que s'utilitza (Logistic Regression, SVM, etc.), també de el tipus de divisió del train i test que fem.

Per exemple si fem un anàlisi amb Time Series les variables no les estem seleccionant de la forma més adequada, ja que l'RFE utilitza les dades completes de tota la temporada per estimar les variables i en canvi en les Time Series només estem utilitzant les dades per ordre cronològic. Malgrat aquest fet, per tal de evitar una extrema complexitat a l'hora de realitzar la anàlisis de resultats per haver de canviar les variables adequades per cada petit subconjunt, que escapa de l'abast del treball, es farà d'aquesta manera. Però es un aspecte a millorar en un futur.

Un cop sabem quines variables són aparentment les més rellevants, escollim les variables independents (X) que estan en un rang millor classificades en l'anàlisi RFE que hem realitzat anteriorment. També afegim la variable dependent (y) per el nostre

model.

```
X = np.asarray(dataframe[['%W','%W as Home/Visitor','Conference_Visitor','OT last match','Last N','Age']])
X[0:5]
```

```
array([[ 0.073     ,  0.176     ,  0.        ,  0.        ,  0.07      ,
         0.59558824],
       [ 0.049     ,  0.18      ,  0.        ,  0.        ,  0.05      ,
        -2.        ],
       [-0.079     , -0.013     ,  1.        ,  0.        , -0.08      ,
         0.13122172],
       [ 0.134     ,  0.239     ,  1.        ,  0.        ,  0.13      ,
         0.55357143],
       [ 0.201     ,  0.28      ,  1.        ,  0.        ,  0.2       ,
         1.57466063]])
```

```
y = np.asarray(dataframe['Winner'])
y [0:5]
```

```
array([1, 1, 1, 0, 1])
```

Figura 32. Codi que agafa les variables seleccionades en el ranking anterior

A continuació es processen i s'escalen totes les dades per tenir-les ben preparades per el training i el test.

```
X = preprocessing.StandardScaler().fit(X).transform(X)
X[0:5]
```

```
array([[ 0.30729595,  0.15830714, -1.00723524,  0.0178806 ,  0.26586807,
         0.30587088],
       [ 0.21359698,  0.17208254, -1.00723524,  0.0178806 ,  0.20115063,
        -0.94893253],
       [-0.28613085, -0.49258047,  0.99281674,  0.0178806 , -0.21951279,
         0.08137892],
       [ 0.54544749,  0.37526968,  0.99281674,  0.0178806 ,  0.46002042,
         0.2855584 ],
       [ 0.80702377,  0.51646753,  0.99281674,  0.0178806 ,  0.68653148,
         0.77919072]])
```

Figura 33. Codi de l'escalat de les diverses variables de X.

4.4.3. Obtenció de les prediccions a partir de Time Series Cross Validation

Un dels objectius del projecte és observar com varia la precisió en la predicció al llarg de la temporada. Teòricament es podria esperar que la precisió augmentés a mesura que avança la temporada, ja que el model hauria de anar aprenent, però caldrà comprovar-ho amb resultats.

Per tal de realitzar aquest anàlisis, utilitzarem la Cross Validation of Time Series que consisteix en el següent procés.

Primer de tot indiquem el nombre de divisions temporals que realitzarem al llarg de la temporada. Després es crea un bucle amb l'eina 'for' que genera una predicció per cada una de les particions ja esmentades.

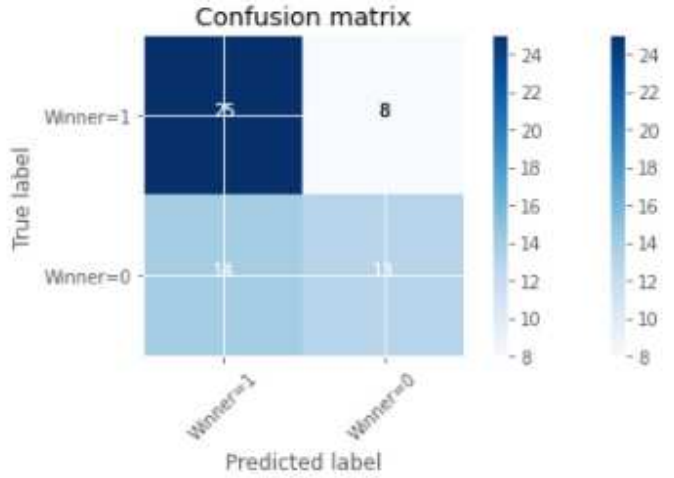
Es realitzarà la simulació tant amb l'algoritme Logistic Regression com també amb l'algoritme SVM.

```
tscv = TimeSeriesSplit(n_splits=15)

correct=[]

for train_index, test_index in tscv.split(y):
    #print("%s %s" % (train_index, test_index))
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    LR = LogisticRegression(C=0.1, solver='liblinear').fit(X_train,y_train)
    yhat = LR.predict(X_test)
    yhat_prob = LR.predict_proba(X_test)

    jaccard_score(y_test, yhat)
    # Compute confusion matrix
    cnf_matrix = confusion_matrix(y_test, yhat, labels=[1,0])
    np.set_printoptions(precision=2)

    # Plot non-normalized confusion matrix
    plt.figure()
    plot_confusion_matrix(cnf_matrix, classes=['Winner=1','Winner=0'],normalize= False,  title='Confusion matrix')
    cor=plot_confusion_matrix(cnf_matrix, classes=['Winner=1','Winner=0'],normalize= False,  title='Confusion matrix')
    correct=correct+[cor/float(len(y_test))]
    plt.show()

    print (classification_report(y_test, yhat))

    log_loss(y_test, yhat_prob)

    LR2 = LogisticRegression(C=0.1, solver='liblinear').fit(X_train,y_train)
    yhat_prob2 = LR2.predict_proba(X_test)
    print ("LogLoss: : %.2f" % log_loss(y_test, yhat_prob2))
```

Figura 34. Codi corresponent al mètode de Time Series Cross Validation.

Finalment s'imprimeix la matriu de confusió per cada una de les iteracions realitzades i a partir d'aquest moment ja es podran analitzar els resultats obtinguts, per la simulació.

Com també es pot veure en el codi creem també una llista anomenada 'correct' que contindrà per cada iteració l'enter que indiqui el nombre d'encerts total. Això ens permetra veure la evolució dels resultats durant la temporada.



	precision	recall	f1-score	support
0	0.62	0.48	0.54	27
1	0.64	0.76	0.69	33
accuracy			0.63	60
macro avg	0.63	0.62	0.62	60
weighted avg	0.63	0.63	0.63	60

Figura 35. Confusion matrix i valors de les mètriques d'avaluació, en una iteració del model.

Finalment, cal tenir en compte que la divisió ideal en forma de Time Series seria fer un Split de n-1 vegades, on n és el nombre total de partits de la temporada. Ja que d'aquesta manera no perdríem informació de cap dels partits anteriors. No obstant això té un cost de càlcul molt elevat i en l'àmbit d'aquest projecte, que és una prova de concepte, s'ha decidit dividir en Splits de aproximadament 10 partits, que és aproximadament el nombre de partits que es juga cada dia en la NBA.

Un altre enfocament que s'ha donat a partir de la anàlisis a través de Time Series és realitzar el mateix tipus de anàlisis però per cada un dels 30 equips de la lliga. L'objectiu d'aquest enfocament és analitzar quins equips obtenen millors prediccions per separat i també veure si separant per equips el model preveu millor o pitjor els resultats que utilitzant

totes les dades en conjunt..

En aquesta ocasió tindrem 30 Dataframes diferents que contindran la informació de tots els partits disputats de cada equip. Per realitzar la predicció el més realista possible realitzarem N-1 splits, (on N és el numero total de partits disputats per un equip) d'aquest manera no perfem informació de cap partit i es preveuran els resultats un per un.

```
for key in teams:
  s=[]
  accuracy=[]
  X = np.asarray(teams[key][['%W','%W as Home/Visitor','Conference_Visitor','OT last match','Last N','Age']])
  y = np.asarray(teams[key]['Winner'])


  X = preprocessing.StandardScaler().fit(X).transform(X)

  tscv = TimeSeriesSplit(n_splits=len(X)-1)

  #correct=[]

  for train_index, test_index in tscv.split(y):
    #print("%s %s" % (train_index, test_index))
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    if len(X_test)>=2:
      clf = svm.SVC(kernel='linear',C=1)
      clf.fit(X_train, y_train)
      yhat = clf.predict(X_test)
    else:
      pass
```

Figura 36. Codi corresponent al mètode utilitzat per l'anàlisi Time Series per cada un dels equips.

Es provarà aquest model només amb l'algoritme SVM i amb les dades de la temporada 2019-2020, ja que aquest anàlisis generarà una gran quantitat de dades analitzar i si es realitzés amb tots els casos possibles, suposaria un cost de càlcul i de processat de dades, que escapa de l'abast del projecte.

4.4.4. Obtenció de prediccions a partir de Random Permutation Cross Validation

L'objectiu d'aquest tipus de simulació és obtenir dades de la precisió en la predicció, oblidant-nos completament de l'ordre temporal dels partits. Com ja hem explicat la Random Cross Validation agafa aleatòriament els partits que pertanyeran al train i al test. Per tant el que busquem és veure si podríem utilitzar el model per predir de forma igual de precisa sense haver de tenir en compte en quin punt de la temporada estem. Això ens donarà una visió clara de si realment és útil utilitzar el Time Series Split com semblaria lògic o en canvi no es rellevant l'ordre cronològic dels partits a l'hora de obtenir bons resultats del model. També ens pot servir per saber quin seria el sostre de precisió del model, ja que aquí en totes les iteracions ja fem servir les dades de tota la temporada.

El procés de modelatge serà molt similar al anterior, però canvien algunes coses importants que s'expliquen a continuació:

La primera diferència que trobem amb el procediment anterior correspon a la forma com es realitza el 'split'. A la setena línia del codi s'observa que es donen a la funció que realitzar el 'split' dos paràmetres que són test_size i random_state.

```
correct=[]
correct_average=[]
random=[]
N=200
adjustment=0
for m in range(N):
    X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.07, random_state=m+280)

    print ('Train set:', X_train.shape,  y_train.shape)
    print ('Test set:', X_test.shape,  y_test.shape)

    LR = LogisticRegression(C=0.9, solver='liblinear').fit(X_train,y_train)
    yhat = LR.predict(X_test)
    yhat_prob = LR.predict_proba(X_test)

    jaccard_score(y_test, yhat)

    # Compute confusion matrix
    cnf_matrix = confusion_matrix(y_test, yhat, labels=[1,0])
    np.set_printoptions(precision=2)

    # Plot non-normalized confusion matrix
    plt.figure()
    plot_confusion_matrix(cnf_matrix, classes=['Winner=1','Winner=0'],normalize= False,  title='Confusion matrix')
    ad=plot_confusion_matrix(cnf_matrix, classes=['Winner=1','Winner=0'],normalize= False,  title='Confusion matrix')
    adjustment=adjustment+ad
    random=np.append(random,m)
    correct=np.append(correct,ad)
```

Figura 37. Codi corresponent al mètode utilitzat per l'anàlisi Random Cross Validation.

El paràmetre test_size indica el percentatge en tant per 1 de partits que correspondran en el test, en aquest cas serà un 7%.

El paràmetre random_state ens permet indicar el conjunt de nombres aleatoris que es vulguin per realitzar el 'split'. S'hi posa el valor 'm' que correspondrà a un enter diferent en cada iteració començant des de 0 fins a 'N' que serà el valor del nombre de iteracions, que en aquest cas s'ha preestablert a 200. És a dir es faran un total de 200 iteracions obtenint en cada una de elles un set de train i de test completament aleatori i únic.

Un altre punt que canvia i que cal explicar són els paràmetres que ens guardem per realitzar la anàlisis de resultats. A part de la confusion matrix i la llista 'correct', que ja hem vist com s'extreien anteriorment, volem obtenir més variables per poder analitzar correctament com són la mitjana de encerts acumulada i la desviació estàndard acumulada.

Aquests dos paràmetres els volem obtenir per tal de veure si s'estabilitza en algun número de iteracions el percentatge d'encert de resultats.

Finalment també comentar que el mateix procediment s'aplicarà per obtenir els resultats de Random Cross Validation per l'algoritme SVM.

5. Anàlisi dels resultats

Una vegada s'ha aplicat el procés de modelatge per les dades de la temporada 2019-2020, es repeteix el procediment de forma anàloga amb les dades de les temporades 2017-2018 i 2018-2019. D'aquesta manera podrem analitzar si en funció de la temporada varien els resultats obtinguts.

Primer de tot caldrà buscar una llindar de percentatge d'encerts a partit del qual podrem afirmar que el model realitzat és útil.

Per trobar el valor de "base line" o llindar s'utilitzarà la variable %W, d'aquesta manera el llindar serà predir els resultats en funció només de quin dels dos equips tingui el millor percentatge de victòries en aquell moment de la temporada. En la taula següent es poden observar els llindars obtinguts per les diferents temporades analitzades.

TEMPORADA	Precisió "Base-line o llindar"
2019-2020	63,12%
2018-2019	62,51%
2017-2018	62,43%

Taula 7. Llindar de precisió per cada temporada analitzada.

A partir d'aquí caldrà comprovar si els diversos models aplicats permeten superar aquest llindar.

5.1. Comparació SVM i LR en predicció en Time Series Cross Validation

En els objectius del projecte es deixava clar que la principal pregunta que es volia respondre, era amb quina precisió era capaç de predir els resultats de la temporada el nostre model. Es per aquest motiu que del procés de modelatge n'extraiem com es pot veure a la taula que apareix a continuació, la precisió en la predicció i la desviació estàndard.

La precisió correspon al percentatge de partits que el nostre algoritme ha estat capaç de predir correctament.

La desviació estàndard ens indica la variabilitat en la predicció per cada Time Series Split que s'ha realitzat.

TEMPORADA	Precisió SVM	Desviació Estàndard SVM	Precisió LR	Desviació Estàndard LR
2019-2020	65,77%	15,61%	65,88%	14,75%
2018-2019	63,91%	13,81%	65,29%	13,01%
2017-2018	63,66%	15,33%	65,66%	15,15%

Taula 8. Precisió de precisió i desviació estàndard per algoritme i temporada analitzada.

Un cop observats els valors de la taula es pot treure la conclusió que l'algoritme Logistic Regression ha funcionat millor en les tres temporades analitzades, ja que en les tres ha donat una precisió entre el 65 i el 66%. En canvi l'algoritme SVM té un descens considerable de la precisió sobretot en les temporades 2018-2019 i 2017-2018.

Pel que fa la desviació estàndard es mou en tots els casos entre el 13 i el 16%, per tant la precisió té una gran variabilitat, és a dir el nostre model no es capaç de predir amb una precisió constant en cada iteració.

Un altre dels factors que es volia analitzar en aquest procés de modelatge era si el nostre algoritme era capaç de aprendre i millorar la precisió en les prediccions al llarg de la temporada. Amb l'objectiu de poder extreure conclusions en aquest sentit s'han elaborat els gràfics que apareixen a continuació. Aquests gràfics corresponen a la evolució de la precisió al llarg de la temporada. En verd veiem la evolució de la precisió en la predicció per cada Time Series Split, i en vermell la mitjana mòbil de tota la temporada. La mitjana mòbil en un període de temps "t", es calcula de la següent manera:

$$SMA_t = \frac{x_t + x_{t-1} + x_{t-2} + \cdots + x_{M-(t-1)}}{M}$$

(Equació 7)

Aquest paràmetre ens permet treure una línia de tendència dins de mostres amb molta variabilitat. D'aquesta manera podem analitzar el nivell d'aprenentatge que té el model al llarg de la temporada.

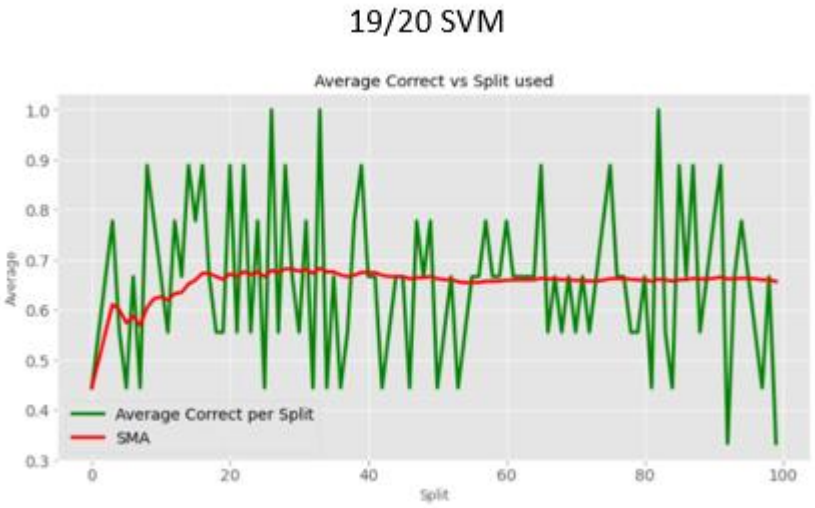


Figura 38. Mitjana d'encerts vs quantitat de splits SVM 19/20

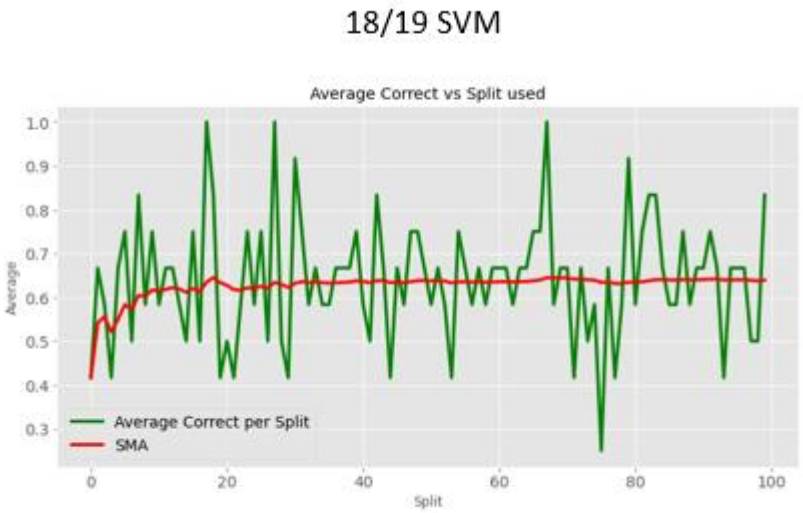


Figura 39. Mitjana d'encerts vs quantitat de splits SVM 18/19.

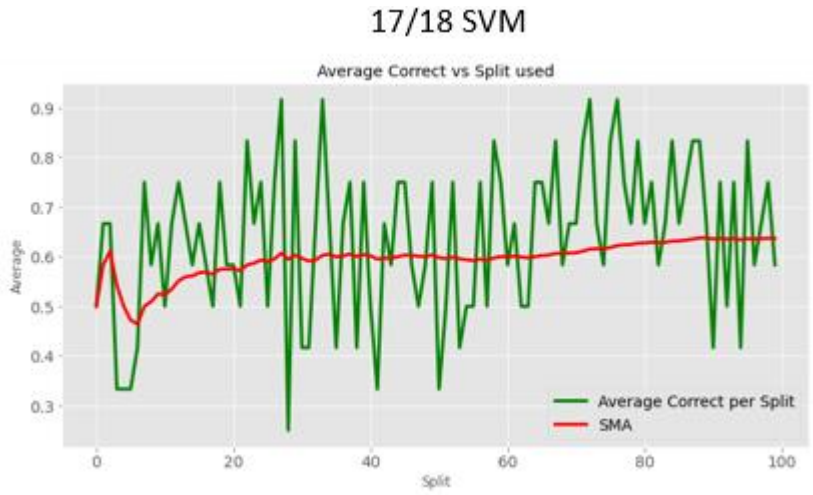


Figura 40. Mitjana d'encerts vs quantitat de splits SVM 17/18.

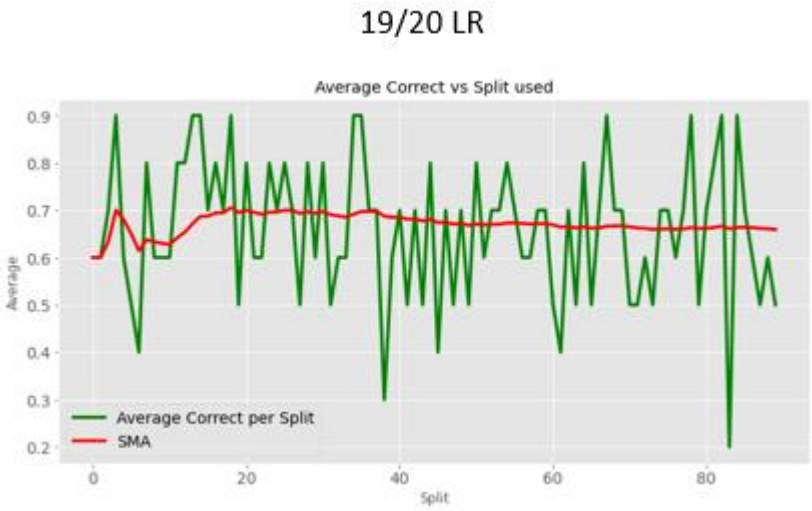


Figura 41. Mitjana d'encerts vs quantitat de splits LR 19/20

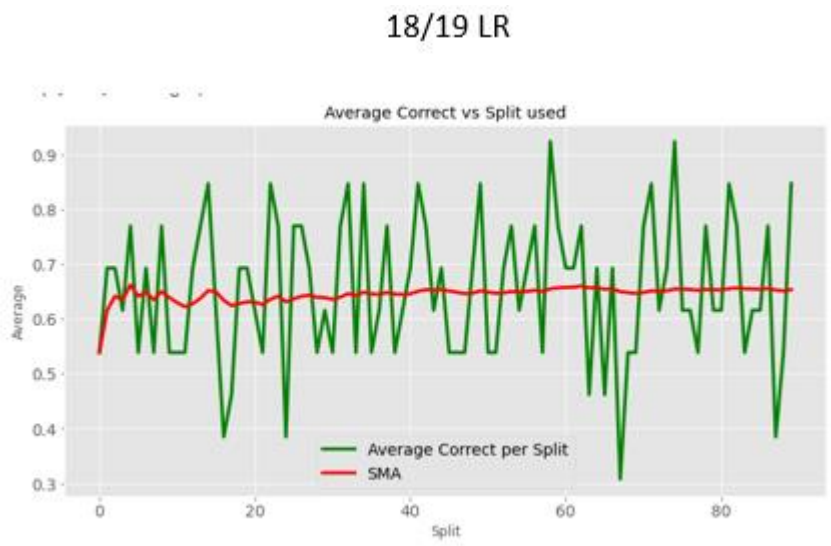


Figura 42. Mitjana d'encerts vs quantitat de splits LR 18/19

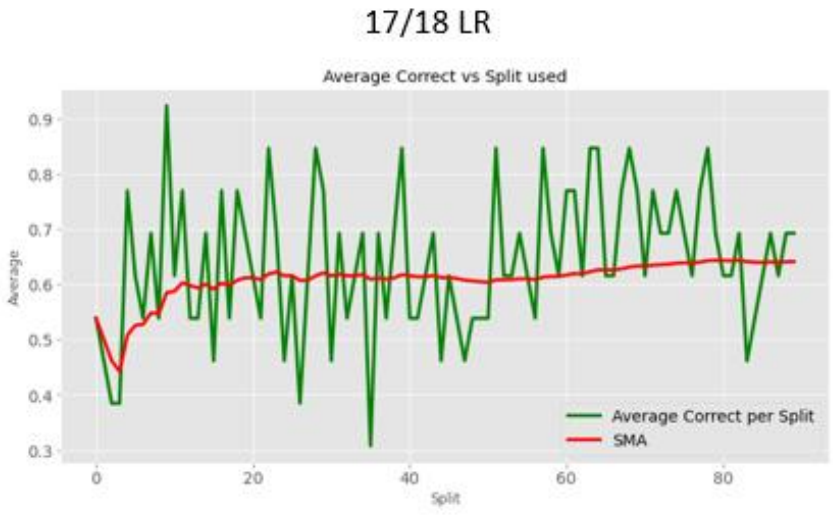


Figura 43. Mitjana d'encerts vs quantitat de splits LR 17/18

En la majoria de gràfics es pot observar que l'encert en la predicció millora al llarg de la temporada. No obstant en la majoria de casos s'estabilitza molt a partir del Split 20 aproximadament. Això probablement és degut a que al disposar de poques dades en les primeres prediccions la precisió a cau a causa de la falta de entrenament de l'algoritme. Un cop l'algoritme ja té suficients dades per predir de forma més precisa, la precisió en les prediccions incrementa de forma molt més lenta.

Un altre part important a tenir en compte, és la gran variabilitat en la precisió de les prediccions, ja que com es pot observar hi ha splits en els que s'encerta un 85% dels resultats i d'altres en els que s'encerta menys del 50%. Aquest fet té una explicació molt simple, el petit tamany de cada Split. Al dividir les nostres dades en 100 splits això significa que cada Split només conté 12 o 13 partits i per tant la aleatorietat influeix molt més que si realitzéssim splits molt més grans de per exemple 100 partits.

5.2. Comparació SVM i LR en predicció per Random Cross Validation

Aquest model basat en la Random Cross Validation s'ha creat amb dos objectius principals. Ser capaços de veure la precisió que pot aconseguir el model si ens oblidem de l'organització cronològica de les dades. És a dir en la realització de splits no es tindrà en compte la data en la que s'ha disputat cada partit si no que en cada Split els partits que van al traint i al test són seleccionats de forma aleatòria.

A més en aquest model a més s'han realitzat Splits amb un 70% de les dades en el train i un 30% de les dades en el test és a dir que es treballa amb volums de test molt més grans que en la time series Split.

TEMPORADA	Precisió SVM	Desviació Estàndard SVM	Precisió LR	Desviació Estàndard LR
2019-2020	66,45%	5,75%	66,21%	5,81%
2018-2019	64,98%	4,96%	65,51%	5,03%
2017-2018	64,81%	5,01%	64,31%	4,93%

Taula 9. Precisió de precisió i desviació estàndard per algoritme i temporada analitzada.

Un cop observats els valors obtinguts a la taula podem destacar que la precisió en la predicció és pràcticament la mateixa utilitzant els dos algoritmes. També podem veure que la precisió en les prediccions ha augmentat en general al voltant de un 1% en totes les temporades si es comparen amb les de Time Series Split. Probablement aquesta diferència es deu a que l'algoritme no passa per un procés de aprenentatge ja que ja des del primer test disposa de una gran quantitat de dades amb les que entrenar-se.

Per exemple si es disposen de 1230 partits, el train correspon a un 93% (1143 partits) en canvi en el primer Split de la Time Series, tenim 12 partits com a train ja que dividim la mostra entre 100.

Un altre aspecte que mostra un canvi notable respecte la Time Series és el valor de la desviació estàndard. Passem de valors del voltant del 14 o 15% a valors entre el 4 i el 6%.

Això és degut també a la gran quantitat de test (87 partits) que tenim per cada test Split en comparació amb la Time Series en el que cada test Split té un total de 12 partits.

No obstant no és la única conclusió que es pot extreure d'aquest model. Com ja s'ha comentat anteriorment per obtenir un percentatge fiable de precisió en la predicció de resultats. No ens podem quedar només amb una simulació train i test ja que com hem vist, la precisió en les prediccions estan exposades a unes desviacions estàndard al voltant del 5%. És per això que en aquest model s'ha realitzat el procés 200 vegades obtenint un nombre d'encerts per a cada iteració. En el següent gràfic podem observar el nombre d'encerts en cada una de les 200 iteracions en funció del nombre de vegades que s'ha donat cada valor d'encerts.

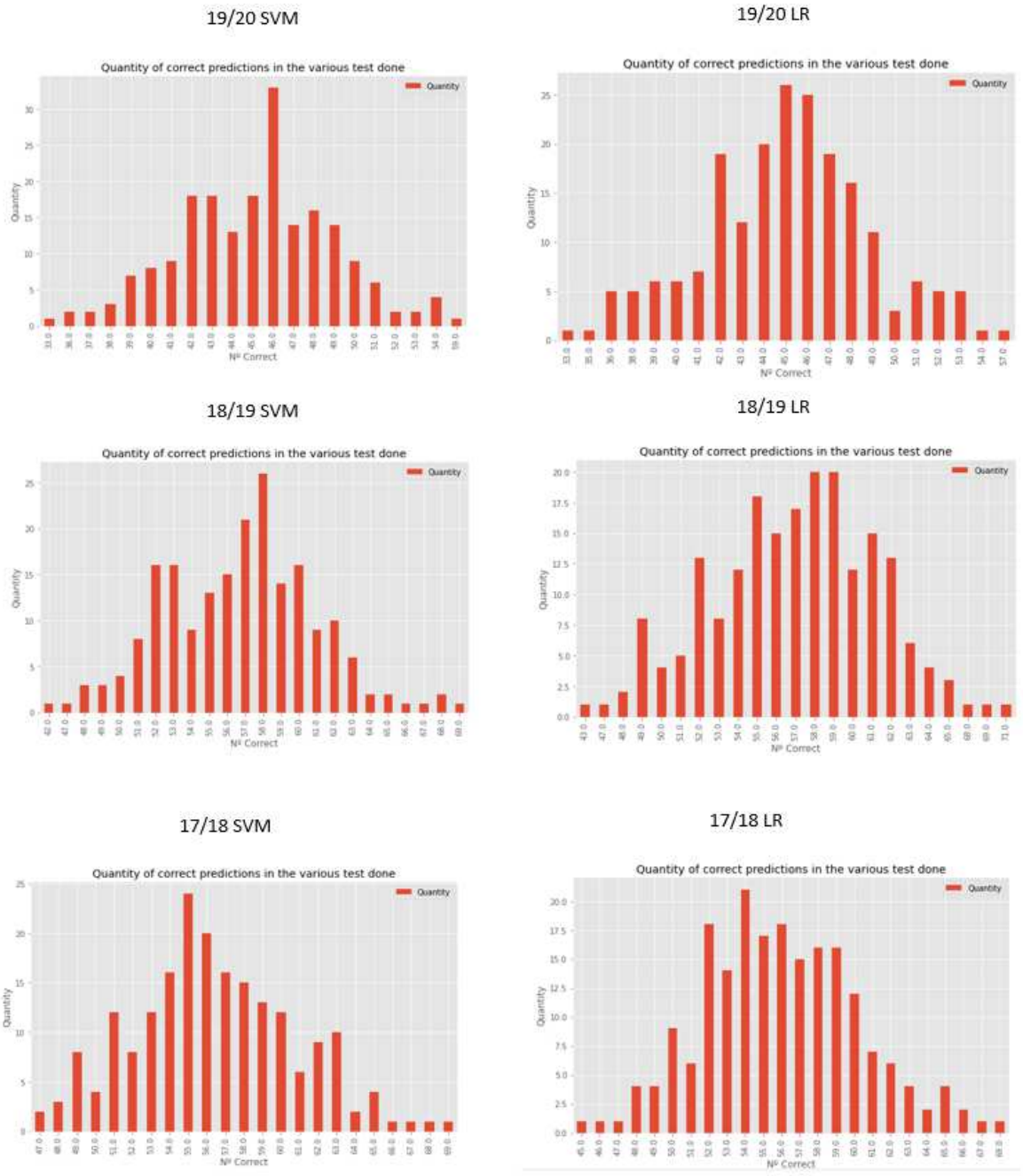


Figura 44. Quantitat de prediccions correctes en les 200 iteracions fetes.

Com es pot observar clarament en totes les simulacions la mostra ha seguit una distribució normal pel que fa al nombre d'encerts, on els encerts més propers a la mitjana es repeteixen en moltes més ocasions que en el nombre d'encerts allunyats de la mitjana.

Un altre paràmetre que s'ha cregut rellevant és quantes simulacions havíem de realitzar amb aquest model, per tal de obtenir una precisió fiable, és a dir a partit de quantes N simulacions la precisió mitjana ja no varia de forma rellevant.

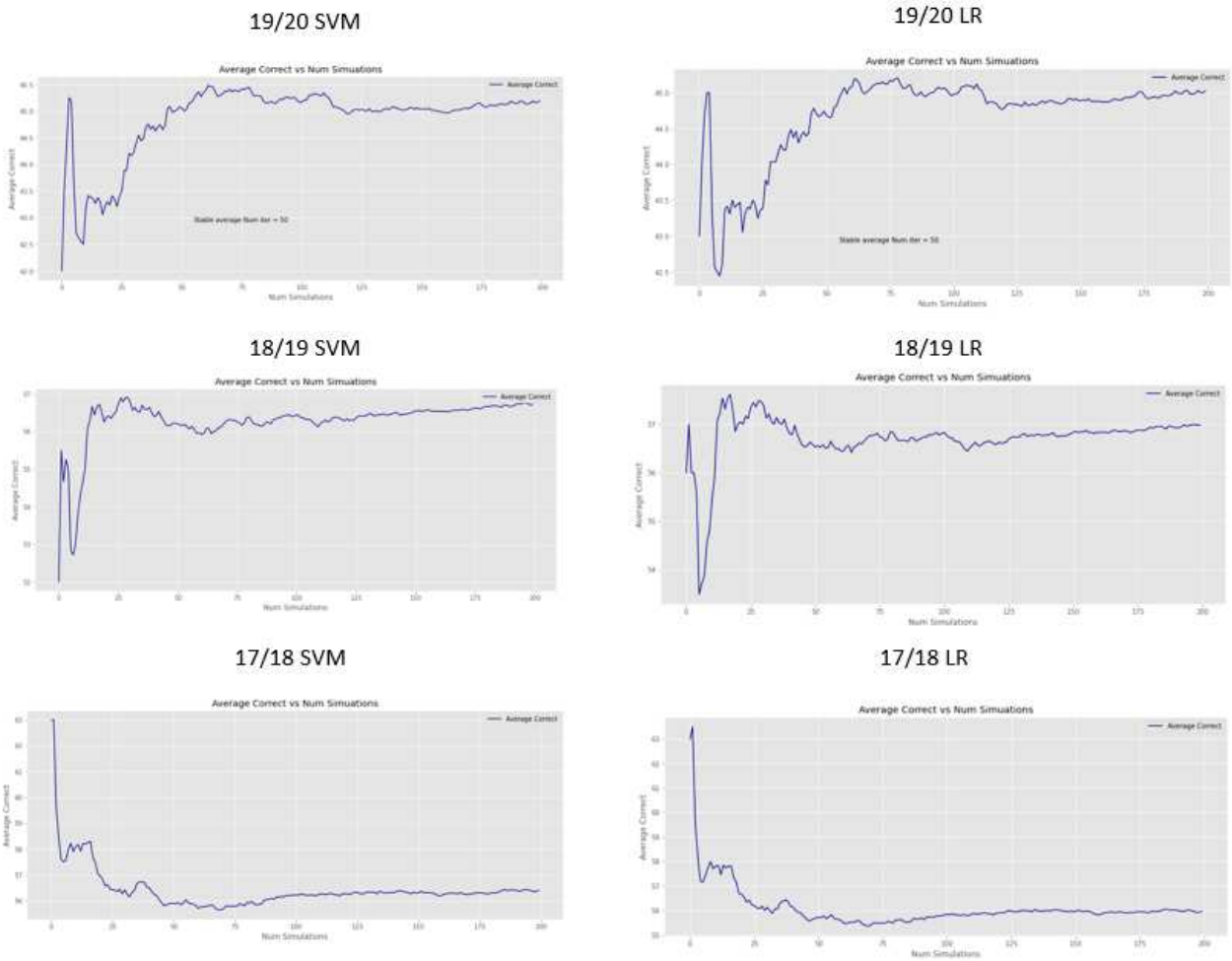


Figura 45. Mitjana de prediccions correctes vs quantitat de iteracions fetes.

Com es pot veure en totes les gràfiques a partir de unes 125 o 150 simulacions la precisió en la predicció ja no varia de forma rellevant, per tant podem concloure que només cal executar al voltant de 150 simulacions per tenir un valor fiable de precisió. D'aquesta manera evitarem temps de càlcul realitzant simulacions extres que no aportarien cap variació en el resultat final.

5.3. Análisis Time Series per equips

Finalment s'ha volgut analitzar breument com canviarien els resultats si només apliquéssim el model Time Series als partits d'un equip en concret. El principal motiu com ja s'ha dit era veure si varia molt la precisió en les prediccions en funció de l'equip analitzat. Com que hi ha 30 equips diferents a la NBA i això suposaria una gran quantitat de gràfiques a analitzar s'ha limitat la anàlisis als 30 equips de la lliga per només per la temporada 2019-2020 utilitzant l'algoritme SVM. A continuació veiem els resultats obtinguts per a 8 dels equips de la lliga, que s'han considerat més rellevants. Els valors de predicció dels 30 equips es poden observar a l'annex.

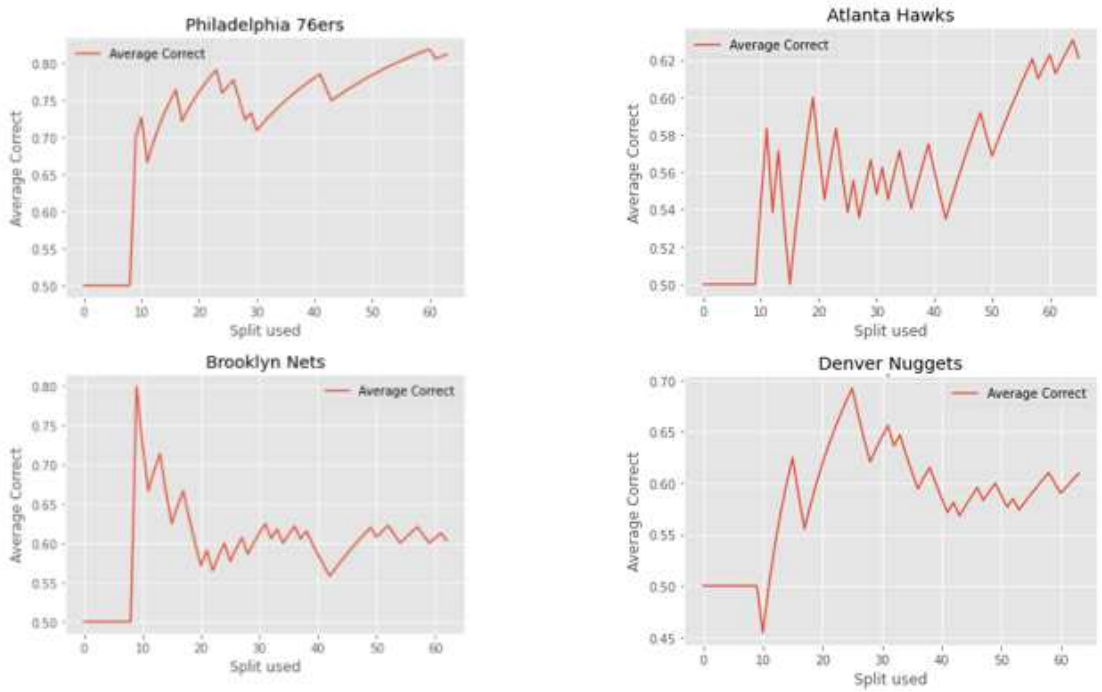


Figura 46. Mitjana acumulada de precisió per quantitat de splits, per diversos equips.

Com es pot observar en aquests quatre primers equips analitzats a excepció dels Philadelphia 76Sixers que obtenim un elevat percentatge d'encert del 80% al llarg de tota la temporada. La majoria d'equips es mouen al voltant del 60% d'encert. Aquest al percentatge d'encert en els 76Sixers es pot atribuir a que durant la temporada 2019-2020, tenien un gran percentatge de victòries a casa i un percentatge molt baix de victòries a fora de casa. Per tant el model a través d'aquesta variable ha estat capaç de predir molt bé els resultats.

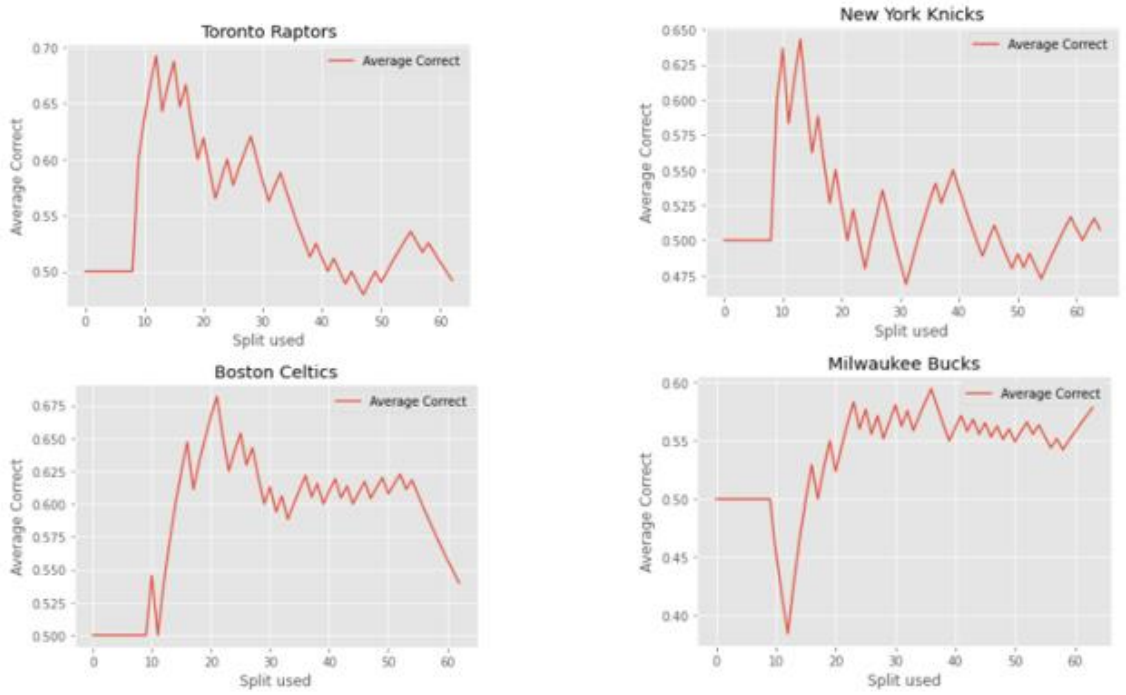


Figura 47. Mitjana acumulada de precisió per quantitat de splits, per diversos equips.

En aquest altres casos podem observar com en canvi tenim un empitjorament en les prediccions sobretot al final de la temporada.

Per tant no es pot concloure que predir els resultats utilitzant només les dades corresponents als partits de cada equip millori el resultat de les prediccions.

Probablement un dels factors més importants és la poca quantitat de dades de la que disposa el model per entrenar-se ja que dividim les dades en splits de 1 partit. I cada equip per exemple en la predicció del quinzè partit disputat, només disposarà de 14 partits en el train i 1 en el test. Això fa que l'algoritme tardi molt a aprendre a interpretar les dades i a més provoca una elevada variabilitat en el percentatge d'encert en les prediccions.

6. Aspectes a millorar

En aquest apartat s'explicaran simplificacions que s'han realitzat amb l'objectiu d'evitar excessives complicacions o aspectes a millorar, que amb el temps limitat del que es disposa en aquest treball no s'han pogut executar. No obstant si es volgués en un futur implementar aquest procés no només en forma de prova de concepte sinó amb aplicacions reals caldria ampliar.

La primera de les millores a realitzar seria l'automatització de la captació de dades a través de web scrapping. El funcionament ideal seria que executant cada dia el codi s'actualitzessin tots el fitxers de dades de forma automàtica. D'aquesta manera es podrien fer prediccions diàries amb dades actualitzades diàriament.

També es podria estendre l'anàlisi a molts altres aspectes del joc en la fase de captació de dades i de Data Preparation, com per exemple analitzar altres estadístiques com rebots i assistències, els resultats anteriors entre els dos equips que disputen el partit, o els enfrontaments anteriors entre els jugadors més importants. La captació i preparació de dades sempre ha de evolucionar i captar noves dades si es vol millorar la precisió del model.

Es podrien buscar algoritmes diferents de classificació que permetessin obtenir millors resultats com per exemple algoritmes basats en arbres de decisió.

Un altre aspecte que caldria millorar clarament és la selecció de variables més adequades per predir els resultats. En aquest projecte s'ha fet mitjançant l'anomenada RFE que fa una classificació de les variables més rellevants utilitzant tot el set de dades, això fa que aquesta decisió no sigui del tot real ja que a la realitat al principi de la temporada no disposaríem de totes les dades, amb les que finalment fem la selecció.

En un futur caldria buscar altres maneres de seleccionar-les i probablement també caldria tenir en compte que en diferents moments de la temporada, potser caldria canviar les variables que s'utilitzen, ja que pot ser que al llarg de la temporada canviïn les que són més rellevants.

Aquest aspecte es podria ampliar encara més en la divisió de models per equips. Ja que probablement cada equip respondria al llarg de la temporada millor a diverses variables. Això s'ha pogut observar en l'apartat de Time Series Split per equips on s'ha vist que la precisió en la predicció dels Philadelphia 76Sixers era molt elevada. I s'ha vist que era degut a que precisament havien estat molt afectats per la variable %W as home/visitor. Que justament s'havia inclòs en el model. Per tant si s'aconsegueix trobar la combinació de

variables més adequada per cada equip es poden millorar molts resultats en aquest tipus de model.

Estendre l'anàlisi del model, a altres paràmetres d'anàlisis a part de la precisió global i la desviació estàndard. Es podria estendre a altres paràmetres com per exemple, Recall o F1 score per veure en què es falla més si en predicció de victòries locals o visitants o el paràmetre Log Loss en el cas de la Logistic Regression que ens permetria saber el cost associat dels errors és a dir si les prediccions fetes que acaben en error s'equivoquen per molt o no.

També es podria augmentar la recollida de dades a més temporades. En el moment que estigués totalment automatitzada la recollida de dades, seria factible obtenir totes les dades des de la temporada 1990-1991 fina a la actual 2020-2021, ja que les webs d'on s'extreuen les dades disposen de totes les dades necessàries a partir de la temporada 1990-1991, per tant es podrien analitzar dades de 30 temporades.

Finalment també caldria millorar la forma de realitzar el Time Series Split, ja que en aquest projecte s'ha fet de forma que en cada Split contingui aproximadament 12 partits. Però la manera ideal de realitzar-ho com ja s'ha explicat seria fer N-1 splits per N partits. D'aquesta manera no es perdria informació de cap partit per el camí per realitzar les prediccions.

7. Cost del projecte acadèmic

En aquest apartat, s'analitza el cost de la realització del projecte. La part corresponent als costos d'enginyeria (en aquest cas, estudiant d'enginyeria) s'inclouen en aquest pressupost.

Això inclou la recerca teòrica sobre tot allò relacionat amb la problemàtica a resoldre: recerca d'exemples, informació sobre el projecte a realitzar. A nivell teòric també s'hi inclou el temps invertit en els cursos online relacionats amb Data Science i els costos econòmics dels mateixos. Els softwares que s'han utilitzat són gratuïts per tant no suposaran cap cost a afegir a aquest pressupost acadèmic.

	Descripció	**Preu/hora (€/h)**	**Temps (hores)**	**Preu (€)**
Treball teòric	Coursera Professional Certificate Data Science	20	180	**3.600**
	Recerca teòrica programació	20	30	**600**
	Redacció de la memòria	20	100	**2.000**
Treball pràctic	Programació del codi	20	150	**3.000**
Material	**Descripció**	**Preu unitat**	**Unitats**	**Preu (€)**
	Material d'oficina	50	1	**50**
Curs Online Data Science	Coursera Professional Certificate Data Science	35	1	**35**
Subtotal				**9.285**
IVA				**1.949,85**
Total + IVA				**11.234,85**

Taula 10. Cost econòmic del projecte.

Conclusions

D'aquest projecte se'n poden extreure diverses conclusions des del punt de vista de la anàlisis estricte del treball.

La principal conclusió és veure si s'ha estat capaç de respondre amb exactitud la pregunta inicial del treball. La pregunta consistia en saber amb quina precisió es podrien predir els resultats en els partits de la NBA. Com és evident per encertar el resultat d'un partit de bàsquet en cas de no disposar de cap informació la probabilitat de encertar el resultat és del 50%, també s'ha definit un llindar, basat en la variable %W a partir del qual considerem que els resultats són positius. S'ha vist que el llindar se situa al voltant del 62 o 63% d'encert. Gràcies als models creats aquesta precisió s'ha situat en el 66% en el millor dels casos és a dir propera a 2 de cada 3 partits es preveuen correctament. Per tant com conclusió es podria qualificar de positiva la precisió aconseguida, ja que s'ha aconseguit superar el llindar amb un percentatge rellevant.

Per tant l'eina sembla útil a la hora de ajudar als general managers i entrenadors de cada franquícia en saber quins partits són més probables de guanyar i quins no. No obstant sembla també evident que la precisió en les prediccions té un petit marge de millora si es realitzen algunes de les accions de futur explicades en l'apartat de millores de futur. No obstant no serà fàcil millorar molt més, ja que la NBA és una lliga molt igualada, el pitjor equip es perfectament capaç de guanyar un partit al millor equip i en general es fa molt difícil predir els resultats, ja que molts cops el factor de l'atzar hi té una gran incidència.

Pel que fa a la part de modelatge, ha quedat palès que la Time Series Split és el mètode més realista i adequat per predir els resultats ja que ens permet controlar l'aprenentatge del model al llarg de la temporada de forma cronològica. A més si es volgués aplicar la metodologia a temps real en un futur seria la manera com es realitzarien les prediccions.

També cal comentar que malgrat pot semblar que la fase més important del treball i a la que s'ha dedicat més temps és la del modelatge. La realitat és que totes són molt importants però tant la elecció de les dades que es capten i també com es preparen aquestes dades són fases importantíssimes que poden modificar molt el resultat final. Ja que sense unes bones de dades, per molt bon modelatge que es faci no s'obtindran bons resultats.

També es vol remarcar que aquest treball té un component molt elevat de programació que no es reflexa especialment en la memòria però que es pot veure en detall a l'annex on està bona part del codi utilitzat per realitzar aquest treball.

Com a conclusió final es vol destacar que aquest projecte es podria ampliar de moltes

maneres diferents i que les possibilitats de millora són il·limitades degut a la gran quantitat de dades disponibles existents de la NBA i també en la predicció no només dels resultats dels partits. Models similars es podrien aplicar per exemple per predir riscos de possibles lesions de jugadors o predicció d'altres aspectes més relacionats amb el rendiment individual de cada jugador.

En aquest treball de final de màster he assolit una gran quantitat de nous coneixements relacionats amb l'enginyeria de dades i el món de Data Science. Quan vaig tenir la idea de realitzar aquest projecte els meus coneixements en aquest àmbit eren molt limitats, i estaven relacionats a la estadística i la programació apresa al grau i al màster en enginyeria industrial. Per tant a nivell d'aprenentatge personal ha estat enormement profitós, i m'agradaria seguir ampliant coneixements en aquest àmbit en un futur.

Agraïments

Vull agrair especialment al meu tutor Lluís Talavera, el suport que m'ha donat durant tota la elaboració del treball. Gràcies a la seva ajuda he pogut aprendre molt i he aconseguit que augmenti encara més el meu interès per el món de la ciència de dades.

Per últim, als membres de la meva família i amics per la paciència demostrada en els bons i mals moments que s'han succeït durant la realització d'aquest treball.

Bibliografia

Referències bibliogràfiques

[1] Informació sobre entre els diferents termes relacionats amb l'àmbit de la Data Science

https://www.kdnuggets.com/2016/03/data-science-puzzle-explained.html

[2] Informació sobre les diferents mètriques d'avaluació

https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

[3] JAKE VAN DER PLAS *. *Python Data Science Handbook*. Londres, O'REYLLI , 2016..

Bibliografia complementària

Obtenció de dades dels millors jugadors de cada temporada

https://www.interbasket.net/news/2019/list-espn-top-100-best-nba-players-2019-20-season-nbarank

Obtenció de dades dels partits disputats.

https://www.basketball-reference.com/leagues/NBA_2020_games.html

Obtenció de dades d'estadístiques avançades dels jugadors.

https://www.basketball-reference.com/leagues/NBA_2020_advanced.html

Obtenció de dades de jugadors en cada partit.

https://pypi.org/project/nba-api/

Informació sobre com canviar el format de les dates dels partits.

https://www.programiz.com/python-programming/datetime/strptime

Informació sobre l'algoritme SVM.

https://scikit-learn.org/stable/modules/svm.html

Informació sobre l'algoritme SVM.

https://www.cienciadedatos.net/documentos/34_maquinas_de_vector_soporte_support_vector_machines

Pàgina web del curs on-line de 180 hores sobre Data Science que vaig fer per preparar-me pel projecte.

https://www.coursera.org/professional-certificates/ibm-data-science

Informació sobre la metodologia Cross Validation.

https://scikit-learn.org/stable/modules/cross_validation.html

Informació bàsica sobre el funcionament de la NBA.

https://es.wikipedia.org/wiki/National_Basketball_Association

Informació sobre el terme Data Science.

https://aukera.es/blog/data-science-que-es-y-que-no-es/