Web scraping

Unai Salaberria Flaño

27 de diciembre de 2022

Índice

Objetivos

El objetivo de este trabajo es aprender sobre las técnicas de extracción de contenidos y datos para posteriormente usarlos a la hora de crear nuestros modelos de aprendizaje automático. Al fin y al cabo, en el mundo de la minería de datos necesitamos, tal y como su propio nombre indica, una gran cantidad de datos. Es por eso que me he interesado en hacer este proyecto, ya que creo que es importante saber cómo conseguir todos esos datos, porque, en muchos casos, no los tendremos a nuestra disposición.

La estructura del trabajo es la siguiente: primero, hago una pequeña introducción al web scraping aprovechando para introducir también el tema de las APIs; después, reflejo el proceso seguido para crear mi primer dataset y, por último, pruebo ese dataset creado por mi mismo. Al final, incluyo algunas conclusiones a las que he llegado tras hacer el proyecto.

Web Scraping

¿Qué es el web scraping?

El web scraping es, a mi forma de entenderlo, una técnica de recopilación de datos.

Pongámonos en contexto. Una empresa de préstamos nos ha encargado un modelo predictivo para saber cuál será la tasa de impagos en el próximo año. Antes de nada, necesitamos los datos con los que entrenaremos ese modelo, pero la empresa solamente nos ha proporcionado una base de datos con el historial de préstamos en los últimos cinco años. ¿Cómo vamos a crear nuestro dataset?

Ahí es donde entra el web scraping. Esta técnica, con el uso de ciertas librerías (que más tarde veremos), nos ayudará a crear nuestro dataset.

¿Es legal?

Una duda que me surgió nada más escuchar sobre el tema es si es legal extraer datos, y, por lo que he visto, es una duda bastante común.

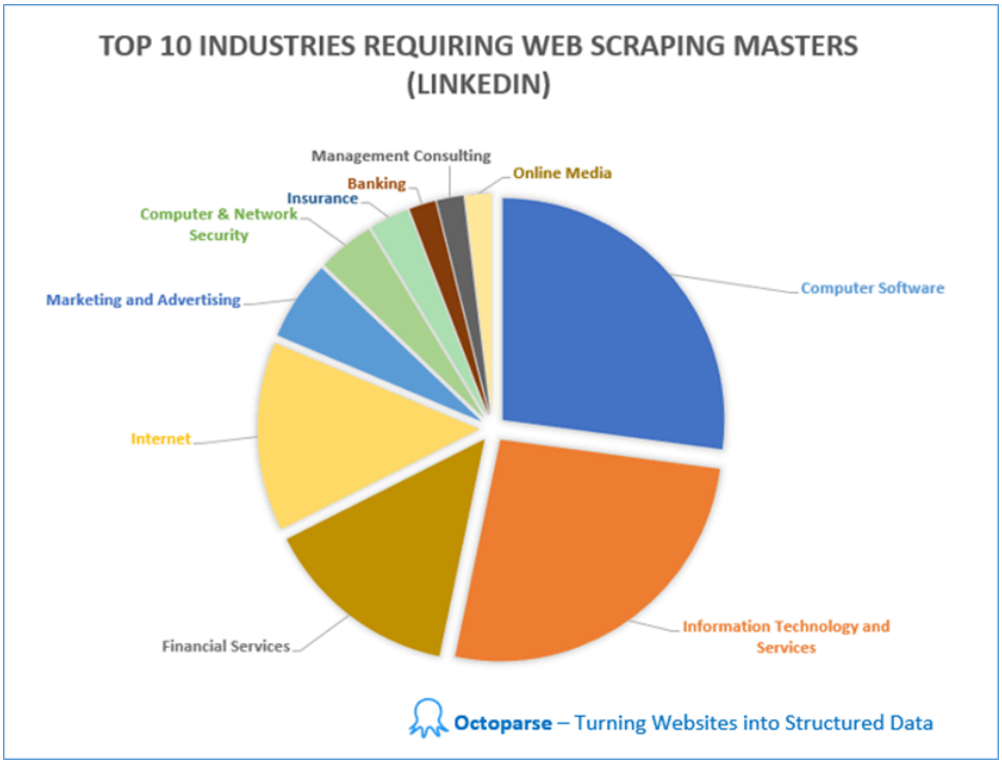
Por lo que he visto, la ley dice que una vez los datos son publicados en Internet pasa a ser legal el scraping de ellos. En cambio, será ilegal obtener esos datos de bases privadas. Además, es conveniente evitar aquellos datos personales que impliquen a una persona concreta sin su previo consentimiento.

¿En qué industrias se utiliza?

El web scraping se utiliza en muchas industrias distintas. Esto es porque, hoy en día, casi todas las industrias hacen uso del análisis de datos. Es de suma importancia saber cómo evoluciona nuestro negocio, y para ello necesitamos datos.

Como es obvio, las industrias dedicadas a la tecnología y el software son aquellas donde más se utiliza esta técnica. Pero hay otras industrias, como la financiera o la de marketing, que también requieren de ella.

Aquí vemos un gráfico de lo que acabamos de comentar:



Web Scraping vs APIs

Diferencias

Una cosa que todavía no he comentado sobre el web scraping es que es capaz de recolectar datos de varios sitios web distintos y luego juntarlo todo, creando así un gran dataset. Es aquí donde encontramos la principal diferencia, y en mi opinión ventaja, sobre las APIs. Una API nos da acceso a los datos de un sistema operativo o aplicación. Para entenderlo más claramente, las APIs son controladas por el propietario de esos datos. Por tanto, será él quien decidirá a quién dar acceso y a quién no, y a cambio de qué.

Ventajas del web scraping

Son varias las razones por las que he decidido enfocarme en el web scraping y no en el uso de APIs.

En primer lugar, con el web scraping no tendremos limitaciones a la hora de conseguir los datos. Es decir, podremos recoger todos los datos que queramos dentro de la web, siempre y cuando no sobrepasemos ese límite ético del que hemos hablado antes.

Además, no dependeremos del permiso de nadie, y no tendremos que estar pendientes de recibir su autorización constantemente.

A pesar de todo, hay ciertas páginas web que no autorizan el scraping, y es necesario utilizar APIs para recoger los datos.

Viendo todas estas características creo que conviene centrarse en el web scraping, ya que ofrece mayor flexibilidad y lo veo más interesante para futuros proyectos que pueda desarrollar.

Funcionamiento

Ahora vamos a enfocarnos en entender cómo funciona esta técnica.

Primero de todo, cabe destacar que el web scraping aparece de muchas formas diferentes. Es posible crear nuestro propio web scraper desde cero o utilizar alguno de los softwares que ya han desarrollado otros programadores.

Herramientas

Para recopilar los datos vamos a necesitar dos herramientas. La primera son los crawlers (rastreadores), que son programas que se encargan de ir navegando por la web y guiando a los scrapers. La segunda son los scrapers (recolectores), que, guiados por los crawlers, extraen todos los datos que necesitemos de la página web.

El problema: Dataset de jugadores de la NBA

Descripción

Por lo que he leído hasta ahora, es muy típico utilizar el web scraping para recoger datos y estadísticas en diferentes deportes. Entonces, he decidido crear un dataset que contenga todas las estadísticas de todos los jugadores de la NBA en la temporada 2022-2023, y además, incluir el salario de cada uno de ellos.

El primer paso será buscar todos esos datos que necesito. A priori, había pensado recopilar todos esos datos de la página web oficial de la NBA, pero he imaginado que es posible que los tengan de alguna manera "protegidos" o solamente estén disponibles mediante APIs. Es por eso que finalmente decido tomar como referencia la siguiente página web: https://www.basketball-reference.com/

Navegando por la página me he dado cuenta de que no tenemos en ningún lado todas las estadísticas junto con los salarios, por lo que tendré que buscar la forma de recoger ambos datasets y después combinarlos.

A partir de ahora, toca entrar en materia y empezar a programar. Como incluiré aparte el código fuente utilizado, voy a comentar solamente algunos apartados que considero importantes.

Programa

Primero de todo, hay que importar aquellas librerías que vayamos a necesitar. Además de pandas, que es la librería que introduce los dataframes, tendremos que importar las librerías beautifulsoup4 y requests. BeautifulSoup nos facilita el uso de varias técnicas de web scraping, y Requests es la librería por defecto para realizar peticiones de tipo HTTP, es decir, nos da acceso a aquellas páginas web que queremos scrapear.

Mediante estas dos librerías vamos a hacer la petición de la página web y, después, cargaremos todo su contenido en una variable para tenerlo de forma accesible. Una vez tenemos el contenido vamos a utilizar la función `find_all()` para obtener los datos que queremos. Esto habrá que hacerlo inspeccionando el código fuente de la página web, ya que necesitamos saber qué tipo de id o clase se utiliza para esa información concreta que queremos. En este paso he tenido ciertos problemas, porque en los ejemplos que he estado mirando siempre tenían el id

explícito, pero en esta página no se etiquetaba la información por id, e incluso a veces ni mediante clases.

Antes de continuar, recuerdo que, tal y como he planteado el problema, necesitamos información de dos páginas distintas, es por eso que mucha parte del código aparece repetida dos veces.

El siguiente paso es guardar en un vector todas las estadísticas de los jugadores, y en otro todos los salarios. Una vez hecho esto, necesito tener guardadas también las etiquetas de cada columna, para saber qué información representa cada columna del dataframe.

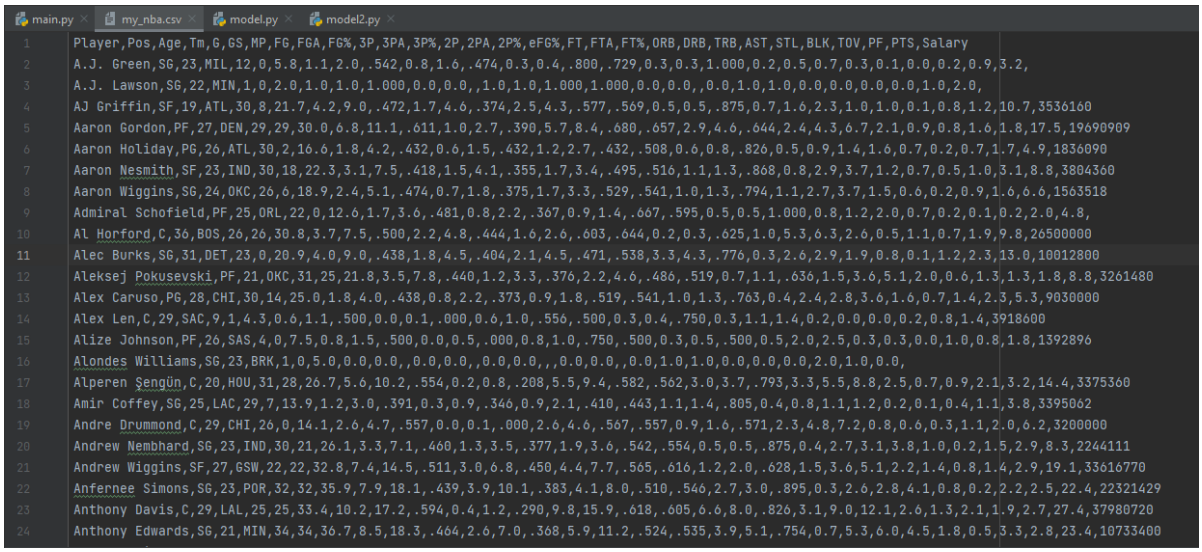
He tenido que limpiar un poco el vector con las etiquetas porque tenía ciertas columnas vacías y, en el caso de las etiquetas de los salarios, había información que no vamos a necesitar, ya que sólo quiero los salarios de esta temporada y no de temporadas futuras.

Ahora ya solo queda crear los dataframes y moldearlos como necesite. He creado ambos dataframe con la librería pandas. Como ya he dicho antes, hay cierta información en el dataframe de los salarios que no nos interesa, por lo que he borrado todas esas columnas, y nos quedamos solamente con los nombres de los jugadores y sus salarios en la temporada 2022-2023.

El último paso del problema es ordenar los dataframes (esto no es necesario pero quería comprobar visualmente si tenía todo buen aspecto) y unirlos. Para unir los dataframes he utilizado la función `merge()` que permite unir uno (el de los salarios) a otro (el de las estadísticas) comparando sobre una variable que tengan en común (el nombre de los jugadores).

Por tanto, ya tendríamos nuestro dataset creado y solo falta convertirlo a un archivo de tipo csv, lo cual se hace con la función `to_csv()`.

Este es el aspecto que tiene nuestro dataset:


main.py | my_nba.csv | model.py | model2.py

```
Player,Pos,Age,Tm,G,GS,MP,FG,FGA,FG%,3P,3PA,3P%,2P,2PA,2P%,eFG%,FT,FTA,FT%,ORB,DRB,TRB,AST,STL,BLK,TOV,PF,PTS,Salary
A.J. Green,SG,23,MIL,12,0,5.8,1.1,2.0,.542,0.8,1.6,.474,0.3,0.4,.800,.729,0.3,0.3,1.000,0.2,0.5,0.7,0.3,0.1,0.0,0.2,0.9,3.2,
A.J. Lawson,SG,22,MIN,1,0,2.0,1.0,1.0,1.000,0.0,0.0,,1.0,1.0,1.000,1.000,0.0,0.0,,0.0,1.0,1.0,0.0,0.0,0.0,0.0,0.1,0.2,0,
AJ Griffin,SF,19,ATL,30,8,21.7,4.2,9.0,.472,1.7,4.6,.374,2.5,4.3,.577,.569,0.5,0.5,.875,0.7,1.6,2.3,1.0,1.0,0.1,0.8,1.2,10.7,3536160
Aaron Gordon,PF,27,DEN,29,29,30.0,6.8,11.1,.611,1.0,2.7,.390,5.7,8.4,.680,.657,2.9,4.6,.644,2.4,4.3,6.7,2.1,0.9,0.8,1.6,1.8,17.5,19690909
Aaron Holiday,PG,26,ATL,30,2,16.6,1.8,4.2,.432,0.6,1.5,.432,1.2,2.7,.432,.508,0.6,0.8,.826,0.5,0.9,1.4,1.6,0.7,0.2,0.7,1.7,4.9,1836090
Aaron Nesmith,SF,23,IND,30,18,22.3,3.1,7.5,.418,1.5,4.1,.355,1.7,3.4,.495,.516,1.1,1.3,.868,0.8,2.9,3.7,1.2,0.7,0.5,1.0,3.1,8.8,3804360
Aaron Wiggins,SG,24,OKC,26,6,18.9,2.4,5.1,.474,0.7,1.8,.375,1.7,3.3,.529,.541,1.0,1.3,.794,1.1,2.7,3.7,1.5,0.6,0.2,0.9,1.6,6.6,1563518
Admiral Schofield,PF,25,ORL,22,0,12.6,1.7,3.6,.481,0.8,2.2,.367,0.9,1.4,.667,.595,0.5,0.5,1.000,0.8,1.2,2.0,0.7,0.2,0.1,0.2,2.0,4.8,
Al Horford,C,36,BOS,26,26,30.8,3.7,7.5,.500,2.2,4.8,.444,1.6,2.6,.603,.644,0.2,0.3,.625,1.0,5.3,6.3,2.6,0.5,1.1,0.7,1.9,9.8,26500000
Alec Burks,SG,31,DET,23,0,20.9,4.0,9.0,.438,1.8,4.5,.404,2.1,4.5,.471,.538,3.3,4.3,.776,0.3,2.6,2.9,1.9,0.8,0.1,1.2,2.3,13.0,10012800
Aleksej Pokusevski,PF,21,OKC,31,25,21.8,3.5,7.8,.440,1.2,3.3,.376,2.2,4.6,.486,.519,0.7,1.1,.636,1.5,3.6,5.1,2.0,0.6,1.3,1.3,1.8,8.8,3261480
Alex Caruso,PG,28,CHI,30,14,25.0,1.8,4.0,.438,0.8,2.2,.373,0.9,1.8,.519,.541,1.0,1.3,.763,0.4,2.4,2.8,3.6,1.6,0.7,1.4,2.3,5.3,9030000
Alex Len,C,29,SAC,9,1,4.3,0.6,1.1,.500,0.0,0.1,.000,0.6,1.0,.556,.500,0.3,0.4,.750,0.3,1.1,1.4,0.2,0.0,0.0,0.2,0.8,1.4,3918600
Alize Johnson,PF,26,SAS,4,0,7.5,0.8,1.5,.500,0.0,0.5,.000,0.8,1.0,.750,.500,0.3,0.5,.500,0.5,2.0,2.5,0.3,0.3,0.0,1.0,0.8,1.8,1392896
Alondes Williams,SG,23,BRK,1,0,5.0,0.0,0.0,,0.0,0.0,,0.0,0.0,,,0.0,0.0,,0.0,1.0,1.0,0.0,0.0,0.0,0.0,2.0,1.0,0.0,
Alperen Şengün,C,20,HOU,31,28,26.7,5.6,10.2,.554,0.2,0.8,.208,5.5,9.4,.582,.562,3.0,3.7,.793,3.3,5.5,8.8,2.5,0.7,0.9,2.1,3.2,14.4,3375360
Amir Coffey,SG,25,LAC,29,7,13.9,1.2,3.0,.391,0.3,0.9,.346,0.9,2.1,.410,.443,1.1,1.4,.805,0.4,0.8,1.1,1.2,0.2,0.1,0.4,1.1,3.8,3395062
Andre Drummond,C,29,CHI,26,0,14.1,2.6,4.7,.557,0.0,0.1,.000,2.6,4.6,.567,.557,0.9,1.6,.571,2.3,4.8,7.2,0.8,0.6,0.3,1.1,2.0,6.2,3200000
Andrew Nembhard,SG,23,IND,30,21,26.1,3.3,7.1,.460,1.3,3.5,.377,1.9,3.6,.542,.554,0.5,0.5,.875,0.4,2.7,3.1,3.8,1.0,0.2,1.5,2.9,8.3,2244111
Andrew Wiggins,SF,27,GSW,22,22,32.8,7.4,14.5,.511,3.0,6.8,.450,4.4,7.7,.565,.616,1.2,2.0,.628,1.5,3.6,5.1,2.2,1.4,0.8,1.4,2.9,19.1,33616770
Anfernee Simons,SG,23,POR,32,32,35.9,7.9,18.1,.439,3.9,10.1,.383,4.1,8.0,.510,.546,2.7,3.0,.895,0.3,2.6,2.8,4.1,0.8,0.2,2.2,2.5,22.4,22321429
Anthony Davis,C,29,LAL,25,25,33.4,10.2,17.2,.594,0.4,1.2,.290,9.8,15.9,.618,.605,6.6,8.0,.826,3.1,9.0,12.1,2.6,1.3,2.1,1.9,2.7,27.4,37980720
Anthony Edwards,SG,21,MIN,34,34,36.7,8.5,18.3,.464,2.6,7.0,.368,5.9,11.2,.524,.535,3.9,5.1,.754,0.7,5.3,6.0,4.5,1.8,0.5,3.3,2.8,23.4,10733400
```

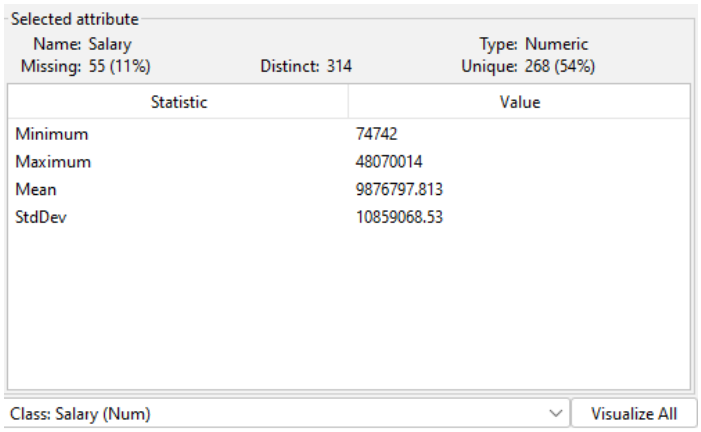
Una cosa que se me ha olvidado comentar es que uno de los pasos finales, antes de conseguir el dataset como lo acabamos de ver, es cambiar el formato de la variable salarios. En un principio, esta variable era un string de la siguiente forma '$1,234,567', así que he tenido que quitar el símbolo de dólar y las comas para pasar la variable a numérica.

Probando el dataset

Primero de todo, he decidido ver qué tal funciona mi dataset en WEKA. Para ello, he convertido el archivo a un arff mediante el ArffViewer de WEKA. Esta funcionalidad nos permite convertir cualquier base de datos en un arff, que es lo que usaremos dentro del programa.

Una vez abierto el archivo en WEKA vemos que tenemos 499 instancias y 30 atributos. Antes de pasar a la pestaña de Classify, hay ciertas variables que no vamos a necesitar, ya que no influyen en nada. Esas variables son el nombre del jugador, el equipo y la posición. Por tanto, nos quedamos con 27 variables, 26 predictoras y la clase 'salary'.

Antes de "jugar" con el dataset, vamos a ver la información sobre la clase:



Hay 55 casos en los que no tenemos información sobre el salario. Por lo que he visto, aquellos jugadores incorporados a mitad de temporada no cuentan con esta información. Además, vemos que la media está en torno a los 10M de dólares, y la desviación estándar es muy alta, casi de 11M. Esto es lógico viendo cómo se distribuyen los salarios en la gráfica de abajo.

Vamos ahora a probar ciertos clasificadores. Como se trata de un problema con clase numérica tendremos que utilizar modelos de regresión.

En primer lugar voy a probar con el algoritmo “LinearRegression”, ya que he leído que es el método más simple entre los de regresión. Al no conocer mucho sobre el algoritmo he dejado los distintos atributos con los valores por defecto. Para la evaluación he usado un 10-fold cross-validation.

Vamos a analizar la información que nos ofrece.

Primero, tenemos que ha creado el siguiente polinomio con las distintas variables predictoras:

```
Linear Regression Model

Salary =

 766196.1217 * Age +
 -97060.3484 * G +
 -92401.0201 * MP +
7116520.4893 * FG +
-4008775.7676 * 3P +
-5376193.529 * 3P% +
-4093305.4626 * 2P +
-802383.4808 * 2PA +
-5733391.6661 * 2P% +
 814926.6146 * FTA +
 888399.7878 * DRB +
1153071.1774 * AST +
2580890.3331 * BLK +
-1125027.9977 * PF +
-13992986.6733
```

Los resultados han sido los siguientes:

```
=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.7804
Mean absolute error                5024736.5503
Root mean squared error            6796220.9099
Relative absolute error                 61.1846 %
Root relative squared error             62.5631 %
Total Number of Instances              444
Ignored Class Unknown Instances                55
```

El parámetro que más me interesa es el mean absolute error, que nos indica que, de media, la predicción se aleja 5M del salario real. No son unos resultados fantásticos, así que voy a probar con otros métodos de regresión.

Una vez probados unos cuantos métodos, finalmente uno de los mejores modelos, teniendo en cuenta el error absoluto medio, ha sido el K-NN.

He utilizado una K=3 y distance weighting de 1/distance.

```
Correlation coefficient                  0.7951
Mean absolute error                4376643.5414
Root mean squared error            6616061.1759
Relative absolute error                 53.293  %
Root relative squared error             60.9046 %
Total Number of Instances              444
Ignored Class Unknown Instances                55
```

Hemos conseguido bajar bastante el error medio, a 4.7M de dólares.

Ahora, vamos a crear un modelo en Python, para cambiar un poco el entorno de trabajo.

Para crear este modelo me he basado en este vídeo: Weather Prediction With Python And Machine Learning [W/Code]

Como he hecho antes, voy a comentar lo más importante del código e incluiré junto a este archivo el código fuente.

El modelo que voy a crear está incluido en la librería sklearn, así que tenemos que importarlo antes de nada. Este se llama Ridge, que es un algoritmo de regresión lineal.

Antes de crear el modelo, he eliminado las filas con algún valor perdido, ya que tengo entendido que Ridge da problemas a la hora de tratar con estos. Además, he visualizado la correlación entre variables, ya que Python nos lo ofrece de forma muy sencilla. Aquí la tenemos:

	Age	G	GS	...	PF	PTS	Salary
Age	1.000000	-0.006195	0.041503	...	0.019766	0.055703	0.387717
G	-0.006195	1.000000	0.529196	...	0.365901	0.362801	0.193472
GS	0.041503	0.529196	1.000000	...	0.590318	0.718588	0.508958
MP	0.087804	0.504860	0.832775	...	0.689256	0.862717	0.615932
FG	0.042650	0.374040	0.730258	...	0.565046	0.991428	0.699401
FGA	0.042588	0.354383	0.715502	...	0.529242	0.977964	0.681823
FG%	0.005516	0.178087	0.137416	...	0.242536	0.144243	0.098910
3P	0.108721	0.293807	0.505028	...	0.301949	0.688541	0.458902
3PA	0.098688	0.278057	0.507498	...	0.315326	0.705596	0.475149
3P%	0.083881	0.141493	0.082195	...	0.045841	0.175015	0.073560
2P	0.003779	0.327939	0.673263	...	0.561182	0.911612	0.654482
2PA	-0.003488	0.321992	0.684700	...	0.545006	0.928349	0.657151
2P%	0.001250	0.122311	0.043392	...	0.185970	0.061133	0.019311
eFG%	0.058683	0.221099	0.123380	...	0.186468	0.139350	0.085696
FT	0.050187	0.251390	0.580218	...	0.463260	0.891994	0.648085
FTA	0.037056	0.257459	0.581842	...	0.483398	0.881787	0.645717
FT%	0.080111	0.096782	0.157327	...	0.086068	0.251445	0.140915
ORB	-0.036421	0.304376	0.393497	...	0.537210	0.276504	0.202189
DRB	0.068961	0.389323	0.642778	...	0.642305	0.687528	0.562286
TRB	0.042283	0.392513	0.615776	...	0.658033	0.615171	0.495765
AST	0.133835	0.214201	0.544303	...	0.416333	0.705944	0.603875
STL	0.017282	0.318401	0.555718	...	0.492054	0.576275	0.423132
BLK	0.000234	0.264150	0.392307	...	0.464612	0.313671	0.284269
TOV	0.040851	0.312889	0.658976	...	0.566325	0.852361	0.643089
PF	0.019766	0.365901	0.590318	...	1.000000	0.547316	0.378416
PTS	0.055703	0.362801	0.718588	...	0.547316	1.000000	0.705699
Salary	0.387717	0.193472	0.508958	...	0.378416	0.705699	1.000000

De esta forma podemos ver cuales son las variables predictoras que más afectarán en el resultado final.

Ahora, creo una función que se encargará de ejecutar el modelo, además de calcular la diferencia entre las predicciones y el salario real.

Por último, también de la librería sklearn he importado la función `mean_absolute_error()` que nos calculará el error en las predicciones.

El resultado es:

```
5826312.0
```

Por tanto, el resultado obtenido con este modelo no es mejor que el obtenido en WEKA utilizando el K-NN.

Lo siguiente que voy a hacer es convertir nuestro problema en uno de clasificación, es decir, voy a convertir la clase a categórica. Esto con el objetivo de probar otros modelos que hemos visto durante el año.

Para ello, he creado un código que cambie los valores de la clase a “por encima de la media” y “por debajo de la media”. Es decir, calculamos la media de salarios y con esto etiquetamos aquellos casos que estén por encima y por debajo de la media. He utilizado el mismo proceso que seguimos en el laboratorio del K-NN que hicimos en Python.

Ahora, tenemos otro dataset, que lo he llamado “my_nba_categorical”, para probar otros algoritmos de clasificación y ver otras métricas de evaluación.

Vamos a ver la información sobre los salarios en este caso:

Ahora la distribución es binaria, y tenemos aproximadamente el 67% de los casos por debajo de la media y el restante por encima.

He probado primero a crear un modelo Naive Bayes, y estos han sido los resultados con un 10-fold cross-validation:

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         362               81.5315 %
Incorrectly Classified Instances        82               18.4685 %
Kappa statistic                          0.5831
Mean absolute error                      0.1844
Root mean squared error                  0.4175
Relative absolute error                 42.0349 %
Root relative squared error             89.1887 %
Total Number of Instances              444
Ignored Class Unknown Instances                55

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,853    0,264    0,871      0,853    0,862      0,583    0,674     0,669     Below mean
                 0,736    0,147    0,707      0,736    0,721      0,583    0,892     0,758     Above mean
Weighted Avg.    0,815    0,226    0,818      0,815    0,816      0,583    0,745     0,698

=== Confusion Matrix ===

   a   b   <-- classified as
 256  44 |   a = Below mean
  38 106 |   b = Above mean
```

Hemos obtenido resultados bastante decentes.

El último clasificador que voy a utilizar va a ser el Multilayer Perceptron, ya que es de los últimos que hemos visto en clase.

Resultados:

```
Time taken to build model: 0.92 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         370               83.3333 %
Incorrectly Classified Instances        74               16.6667 %
Kappa statistic                          0.6183
Mean absolute error                      0.1613
Root mean squared error                  0.3714
Relative absolute error                 36.7854 %
Root relative squared error             79.3345 %
Total Number of Instances              444
Ignored Class Unknown Instances                55

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,880    0,264    0,874      0,880    0,877      0,618    0,724     0,741     Below mean
                 0,736    0,120    0,746      0,736    0,741      0,618    0,902     0,842     Above mean
Weighted Avg.    0,833    0,217    0,833      0,833    0,833      0,618    0,782     0,774

=== Confusion Matrix ===

   a   b   <-- classified as
 264  36 |   a = Below mean
  38 106 |   b = Above mean
```

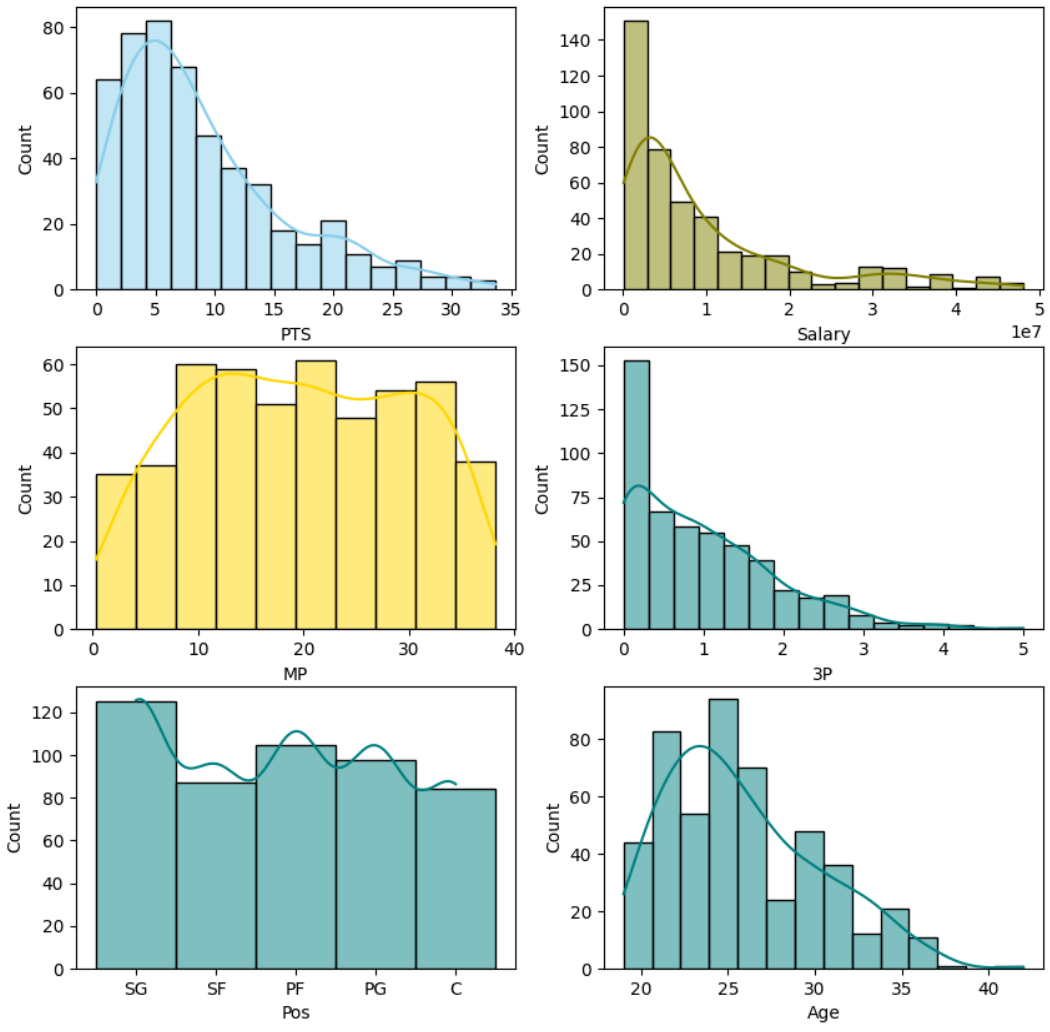
Hemos mejorado un poco los resultados con este modelo.

Visualización de los datos

Por último, vamos a tratar de visualizar los datos que tenemos en distintos gráficos que nos pueden resultar interesantes.

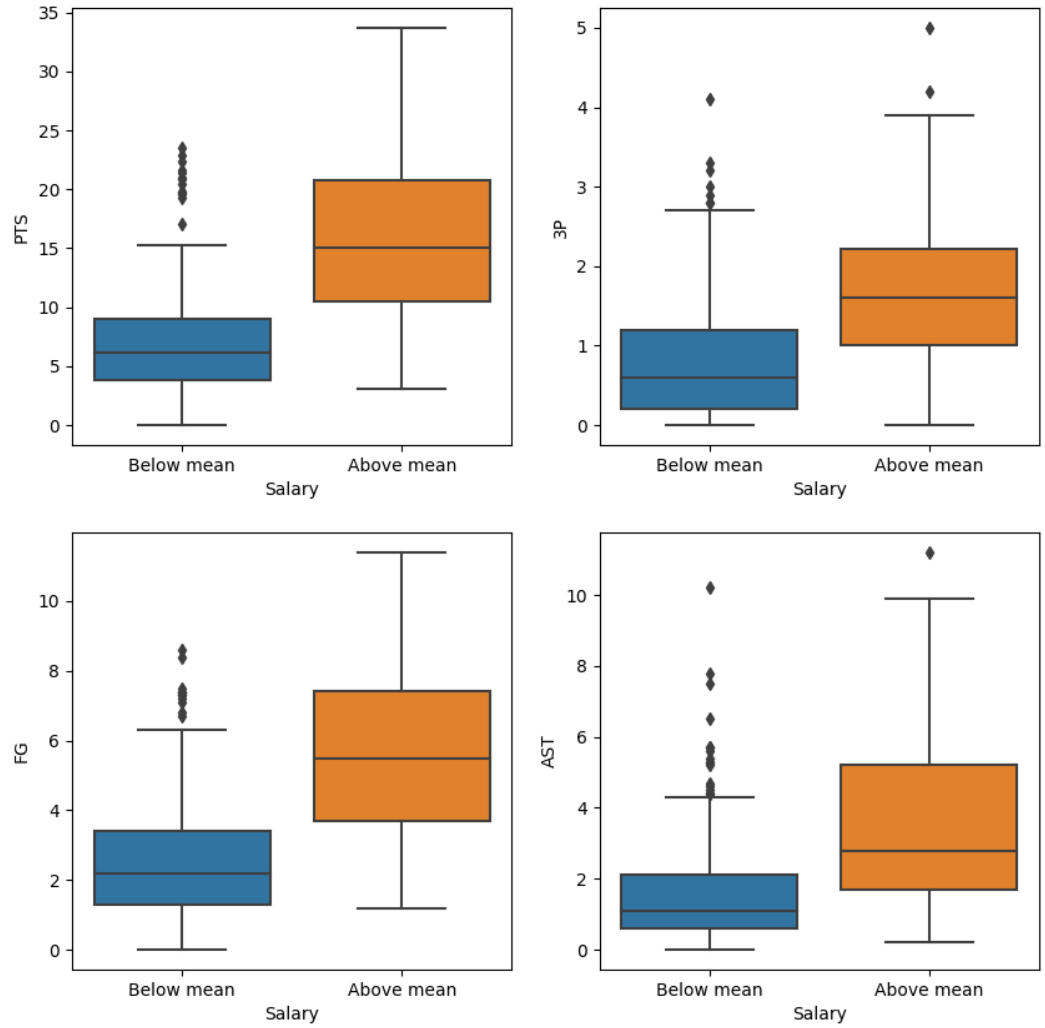
Para ello, he creado otro archivo de código en Python en el que vamos a ver las distintas figuras.

Primero, vamos a visualizar la distribución que siguen algunas de las variables que hemos visto en anteriores pasos.



Algo que me ha llamado la atención es que los puntos, los triples y el salario tienen todos una distribución muy parecida.

Lo siguiente que he hecho es, con el dataset que tiene la clase transformada a categórica, visualizar diagramas de cajas con ciertas variables interesantes comparadas con los salarios.



Haciendo un poco la función de experto, viendo los gráficos podríamos interesarnos por ciertos jugadores con estadísticas muy buenas que tienen salarios por debajo de la media. Hay un caso que me ha sorprendido mucho, y es que hay un jugador con un salario por debajo de la media pero que tiene el segundo mayor promedio de asistencias.

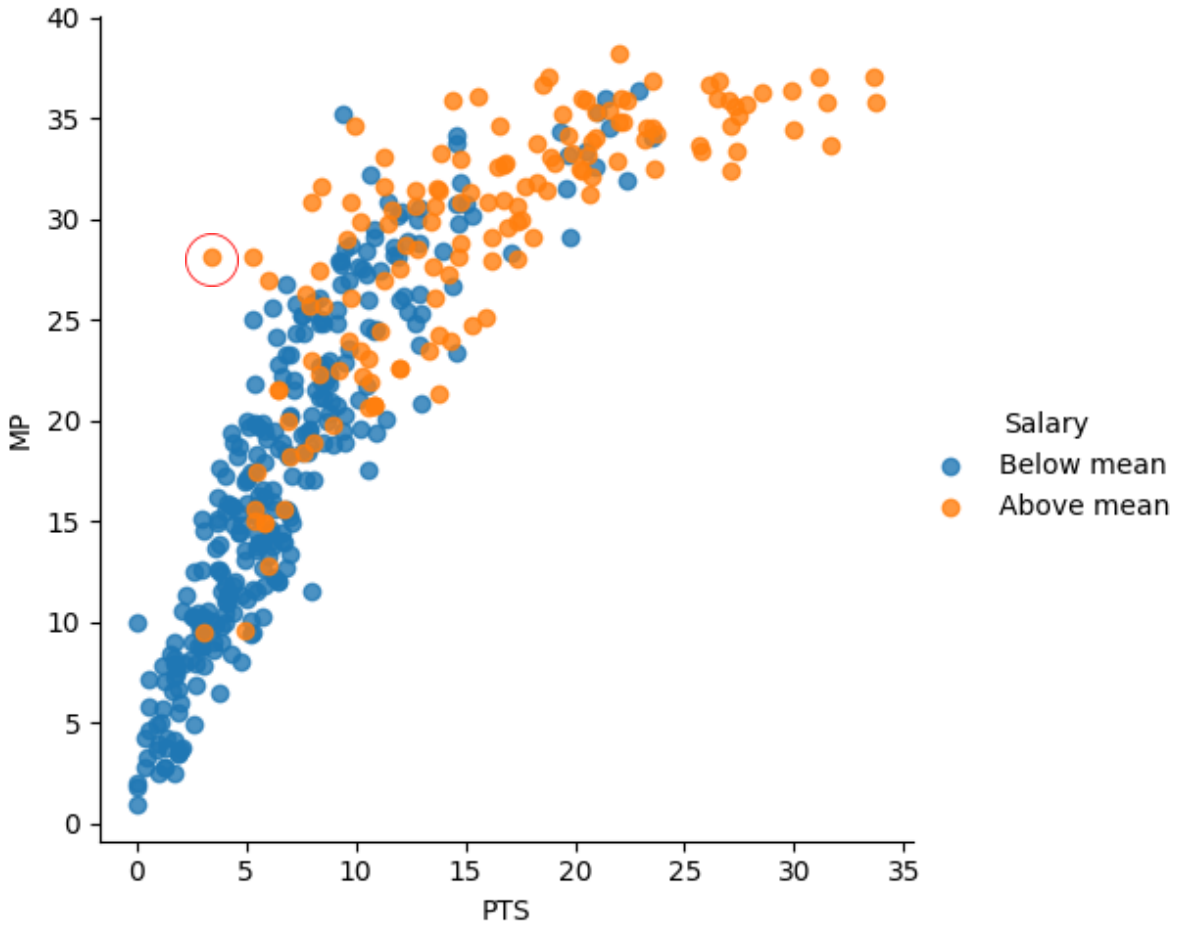
He decidido buscar este caso haciendo una operación de búsqueda en la que filtro aquellos jugadores con un promedio de asistencias mayor a 10. Y este ha sido el resultado:

	Player	Pos	Age	Tm	G	...	BLK	TOV	PF	PTS	Salary
207	James Harden	SG	33	PHI	19	...	0.6	3.7	2.2	22.0	Above mean
473	Tyrese Haliburton	PG	22	IND	33	...	0.4	2.7	1.1	20.5	Below mean

Ese jugador que nos interesa es Tyrese Halliburton, que además, es uno de los pocos jugadores que promedian más de 20 puntos con un salario por debajo de la media.

Hemos comprobado que visualizar los datos de distintas formas es muy útil, ya que nos ofrece puntos de vista que no habíamos visto antes.

Vamos a ver un último gráfico y sacar ciertas conclusiones de él.



Marco en el gráfico un punto que llama mucho la atención. Un jugador que promedia más de 25 minutos, menos de 5 puntos y tiene un salario por encima de la media. Miremos de quién se trata. Con las condiciones que hemos visto sólo tenemos un jugador:

	Player	Pos	Age	Tm	G	GS	...	STL	BLK	TOV	PF	PTS	Salary
385	P.J. Tucker	PF	37	PHI	33	33	...	0.6	0.3	0.8	2.6	3.4	Above mean

Conclusiones

Para terminar, quería reflexionar acerca de lo importante que es la tarea de recopilar los datos y organizarlos correctamente, ya que es principalmente en lo que se basa este trabajo.

Es imposible crear modelos correctamente o visualizar los datos de forma clara sin antes haber creado un buen dataset.

Creo que el haber hecho este trabajo me ha ayudado mucho a entender de donde salen todos esos datasets que durante el curso hemos visto y que recopilar todos esos datos no es tan fácil como aparenta.

Referencias

¿Qué Es el Web Scraping? Cómo Extraer Legalmente el Contenido de la Web

Web Scraping vs API: What's the best way to extract data

Perspectiva de Datos: 54 Industrias que Usan Web Scraping | Octoparse

How To Use Regression Machine Learning Algorithms in Weka - MachineLearningMastery.com

Python Graph Gallery

Predict NBA Games With Python And Machine Learning

Python Tutorial: Working with multiple dataframes in Pandas - Concat and Merge in 9 Minutes

Cleaning NBA Stats Data With Python And Pandas: Data Project [part 2 of 3]

Amazon Web Scraping Using Python | Data Analyst Portfolio Project

Learn Web Scraping With Python: Full Project - HTML, Save to CSV, Pagination

Web Scraping with Python - Beautiful Soup Crash Course