PREDICCIÓN EN TIEMPO REAL DE FUTURAS TRANSACCIONES EN ESTABLECIMIENTOS

SERGI GRAU DALMAU

MÁSTER EN INTELIGENCIA ARTIFICIAL, RECONOCIMIENTO DE FORMAS E IMAGEN DIGITAL

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_____

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Septiembre de 2019

Tutor: Roberto Paredes Palacios

Curso: 2018/2019

Agradecimientos

Antes de empezar, me gustaría agradecer a la Universidad Politécnica de Valencia y en especial a Roberto Paredes, por el año de formación que me han proporcionado y por toda la ayuda que me han facilitado durante todo el curso.

Por otra parte, agradecer a mi familia por estar en todos los momentos buenos y difíciles durante toda mi trayectoria recorrida hasta el momento.

Índice general

1. Introducción al problema

El problema sobre el que tratará esta tesis, consiste en la predicción de asistentes en un cine de la ciudad de Lérida.

Dicho problema surgió en mi empresa, dado que el personal de recursos humanos cada semana se dedicaba a organizar manualmente el horario y los trabajadores necesarios, a una semana vista, en el cine.

Todo esto se estaba realizando basándose en ciertas predicciones según varias relaciones estadísticas estipuladas por la empresa.

En el momento de organizar los horarios y los trabajadores en función de dichas relaciones estadísticas, como estas no eran del todo robustas, la tasa de error en cuanto al acierto de asistentes era bastante alto, ya que se fallaba en una media de 90 asistentes.

Entonces mi jefa me propuso la idea de implementar una red neuronal, la cual fuera capaz de predecir los asistentes en el cine con mayor exactitud de la que estaban obteniendo hasta el momento.

El principal requerimiento, fue que se intentara abstraer lo máximo posible del sector del cine dicha red neuronal, ya que se pretende implementar dicho modelo en otros sectores en un futuro.

Por lo qué, como se verá a continuación, únicamente se utilizará un atributo referente a dicho sector, el cual es del todo necesario, ya que el factor película es uno de los más influyentes en cuanto a la asistencia del cine, por no decir el más influyente.

Como modo de introducción en el mundo de las predicciones, previamente a empezar a abordar el problema, se expondrá el estado del arte en las técnicas predictivas.

2. Estado del arte en técnicas predictivas

2.1. Métodos de regresión:

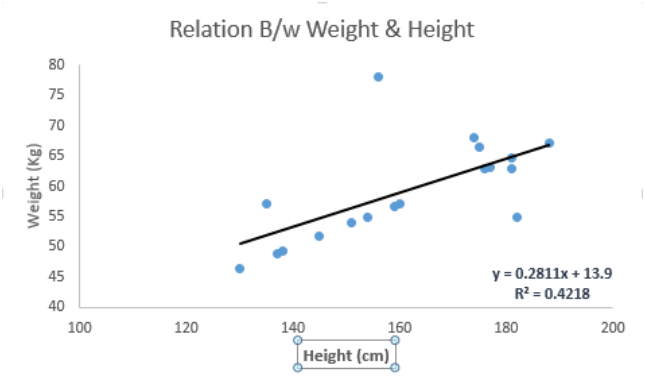
Uno de los aspectos más relevantes de la estadística es el análisis de la relación o dependencia entre variables. Frecuentemente resulta de interés conocer el efecto que una o varias variables pueden causar sobre otra, e incluso predecir en mayor o menor grado valores en una variable a partir de otra.
Por ejemplo, podemos suponer que la altura de los padres influye significativamente en la de los hijos. Entonces podríamos estar interesados en estimar la altura media de los hijos cuyos padres presentan una determinada estatura.
Los métodos de regresión estudian la construcción de modelos para explicar o representar la dependencia entre una variable respuesta o dependiente (y = target) y las variables explicativas o dependientes (x = features).

Para ello, existen diferentes tipos de regresión y a continuación se hará una breve explicación de algunos de estos.

2.1.1. Regresión lineal:

En la regresión lineal, la variable dependiente es continua, las variables independientes pueden ser continuas o discretas y la naturaleza de la línea de regresión es lineal.
La regresión lineal establece una relación entre la variable dependiente (target = y) y una o más variables independientes (features = x) usando una línea de regresión.
Dicha línea es interpretada por una ecuación tal que, $y = a \cdot x + b + e$, donde b es la intersección, a es la pendiente de la línea y e es el término de error.

Esta ecuación es usada para predecir el valor de la variable dependiente en función de las características dadas. Vamos a suponer que queremos predecir el peso de varias personas en función de su altura, entonces podríamos representar la siguiente línea de regresión:



En este caso solo tenemos una variable independiente ya que estamos hablando de una regresión lineal simple, pero de querer añadir mas factores para poder predecir el peso, utilizaríamos la regresión lineal múltiple.

2.1.2. Regresión logística:

La regresión logística es utilizada para encontrar la probabilidad de un evento exitoso o fallido, o mejor dicho para un evento binario (cualitativo), por lo tanto, se debería de utilizar dicha técnica cuando la naturaleza de la variable dependiente (target) es binaria como 0/1, Verdadero/Falso, Sí/No…
En este caso se debe de transformar la variable dependiente cualitativa a una variable dependiente cuantitativa.
Para ello, la variable se transforma mediante el operador logístico en una probabilidad, logrando de esta forma obtener la misma estructura que la regresión lineal.

$$\mathbf{odds} = \frac{\boldsymbol{p}}{(\mathbf{1}-\boldsymbol{p})} \quad (\mathbf{1})$$

$$\ln(\mathbf{odds}) = \ln\left(\frac{\boldsymbol{p}}{(\mathbf{1}-\boldsymbol{p})}\right) \quad (\mathbf{2})$$

$$\mathbf{logit(p)} = \ln\left(\frac{\boldsymbol{p}}{(\mathbf{1}-\boldsymbol{p})}\right) = \boldsymbol{b0} + \boldsymbol{b1}\cdot\boldsymbol{x1} + \boldsymbol{b2}\cdot\boldsymbol{x2} + \boldsymbol{b3}\cdot\boldsymbol{x3} + \boldsymbol{bk}\cdot\boldsymbol{xk} \quad (\mathbf{3})$$

Dónde *p* es la probabilidad de la presencia de la característica de interés.

2.1.3. Regresión polinomial:

La regresión polinomial es un caso especial de la regresión lineal, ya que esta extiende el modelo lineal al agregar características adicionales, que se obtienen al elevar cada una de las características originales a una potencia. Este enfoque proporciona una forma sencilla de añadir un ajuste no lineal a los datos.
El método estándar para extender la regresión lineal a una relación no lineal entre las variables dependientes e independientes, ha sido reemplazar el modelo lineal con una función polinomial.

En el caso de transformar una regresión lineal simple en polinomial nos quedaría la siguiente ecuación:

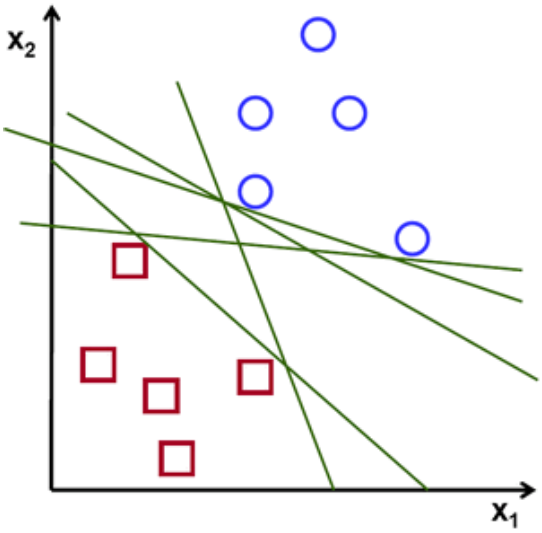
$$\boldsymbol{y} = \boldsymbol{a_1}\cdot\boldsymbol{x_1} + \boldsymbol{b} \qquad \text{->} \qquad \boldsymbol{y} = \boldsymbol{a_1}\cdot\boldsymbol{x_1} + \boldsymbol{a_2}\cdot\boldsymbol{x_1^2} + \boldsymbol{b}$$

Como se puede observar para la regresión polinomial se crean algunas características adicionales que no se encuentran en la regresión lineal.
Por otra parte, el término polinomial, bien sea cuadrático o cúbico, convierte un modelo de regresión lineal en una curva, pero como los datos de *x* son cuadráticos o cúbicos pero el coeficiente *b* no lo es, todavía se califican como un modelo lineal.
Al considerar los ajustes lineales dentro de un espacio de mayor dimensión construido con estas funciones básicas, el modelo tiene la flexibilidad de adaptarse a una gama de datos mucho más amplia.

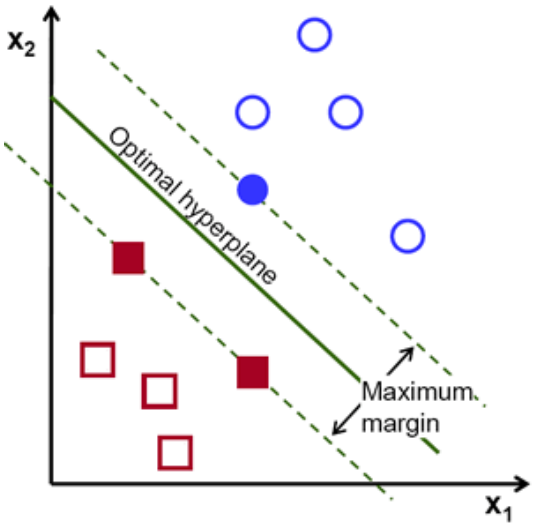
Pero desafortunadamente, la regresión polinomial también tiene un buen número de desventajas, ya que a medida que aumentamos la complejidad de la fórmula, el número de características aumenta, por lo que a veces se convierte en difícil de manejar.

2.2. SVM (Support Vector Machine):

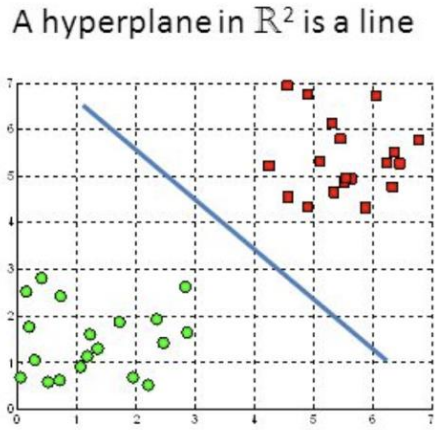
El objetivo del algoritmo SVM es encontrar un hiperplano en un espacio N-Dimensional (donde N es el número de características) que sea capaz de separar los diferentes datos en sus correctas clases, tal y como se puede observar en la siguiente imagen:

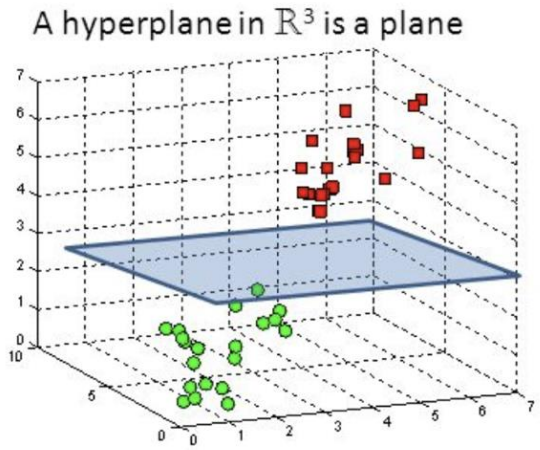


Pero para separar, en este caso las dos clases, existen diferentes hiperplanos a escoger, por lo que el objetivo del SVM es encontrar el plano que maximice el margen, es decir que se encuentre a la máxima distancia posible entre los datos de las dos clases, ya que esta maximización hace más robustas las futuras clasificaciones de datos.



Los hiperplanos son límites de decisión que permiten, tal y como se ha comentado anteriormente, clasificar diferentes puntos de datos, ya que los puntos de datos que caen en una parte u otra del hiperplano son atribuidos a la clase correspondiente a dicha parte.
También es necesario añadir que, según la dimensionalidad del problema, los hiperplanos tienen diferente forma, ya que por ejemplo para un problema de dimensión igual a 2 el hiperplano se convierte en una línea. En cambio, para un problema de dimensión igual a 3 se nos convierte en un plano.





Los vectores de soporte son puntos de datos que están cerca del hiperplano e influyen en la posición y la orientación de este. Gracias al uso de estos vectores, es como se logra la maximización del margen del clasificador.

En dicho algoritmo, se toma la salida de la función lineal y si esta salida es mayor que 1, se identifica como una clase y en cambio sí es igual a -1 se identifica como la otra clase. Dado que los valores de umbral se cambian a 1 y -1 en SVM, obtenemos el siguiente rango de valores de refuerzo que actúan como margen:

[-1,1]

Tal y como se ha comentado, se busca maximizar el margen entre los puntos de datos y el hiperplano y esto se consigue mediante la función de perdida *Hinge loss*:

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases}$$

Donde la pérdida es igual a 0 si el valor predicho y el valor real son del mismo signo. En caso de no serlo, se calcula el valor de la pérdida.

También, para equilibrar la maximización del margen y la pérdida, se agrega un parámetro de regularización. Quedando de la siguiente forma la función de pérdida:

$$min_w \lambda \parallel w \parallel^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle)_+$$

2.3. Redes neuronales

A pesar de su nombre, las redes neuronales no tienen un concepto demasiado complicado detrás de ellas.

El nombre como se puede imaginar, viene de la idea de imitar el funcionamiento de las redes neuronales de los organismos vivos, las cuales consisten en un conjunto de neuronas conectadas entre sí y que trabajan en conjunto, sin que haya una tarea concreta para cada una. Con la experiencia, las neuronas van creando y reforzando ciertas conexiones para aprender algo que queda fijo en el tejido.

Este enfoque biológico se ha ido obteniendo y moviendo para encararlo en las matemáticas y la estadística, de tal forma que mediante ciertos parámetros se intenta buscar una forma de combinarlos y predecir un cierto resultado.

Por ejemplo, sabiendo los píxeles de una imagen habrá una forma de saber que número hay escrito, o conociendo la carga de servidores de un Centro de Procesamiento de Datos (CPD), su temperatura y demás, existirá una manera de saber cuánto van a consumir.

Si bien hay muchos algoritmos de inteligencia artificial en estos días, las redes neuronales pueden realizar lo que se ha denominado aprendizaje profundo, el cual se puede relacionar con el reconocimiento de voz, la visión por ordenador y el procesamiento del lenguaje natural.

Mientras la unidad básica del cerebro es la neurona, el elemento esencial de una red neuronal artificial es un perceptrón que realiza un procesamiento de señal simple conectándose posteriormente a una malla grande. A dicha red se le enseña a hacer una tarea haciendo que analice un conjunto de ejemplos de entrenamiento, los cuales se encuentran etiquetados previamente. A diferencia de otros algoritmos, las redes neuronales con su aprendizaje profundo no pueden programarse directamente para la tarea. Por el contrario, tienen el requisito, al igual que el cerebro en desarrollo, de que necesitan aprender la información.

Existen varias estrategias de aprendizaje:

- *Aprendizaje supervisado*: Se trata de la técnica más simple, ya que se parte de un conjunto de datos previamente etiquetados, con el que se entrena la red hasta que esta es capaz de obtener el resultado deseado.
- *Aprendizaje no supervisado*: Esta estrategia se usa en los casos en los que no hay un conjunto de datos etiquetado disponible para aprender. La red neuronal analiza el conjunto de datos, y luego una función de pérdida informa sobre qué lejos del objetivo se encuentra.
- *Aprendizaje reforzado*: En esta estrategia, la red neuronal se refuerza para obtener resultados positivos y se penaliza por un resultado negativo, lo que obliga a la red neuronal a aprender con el tiempo.

A parte de existir varias estrategias de aprendizaje, también existen múltiples tipos de redes neuronales, cómo, por ejemplo:

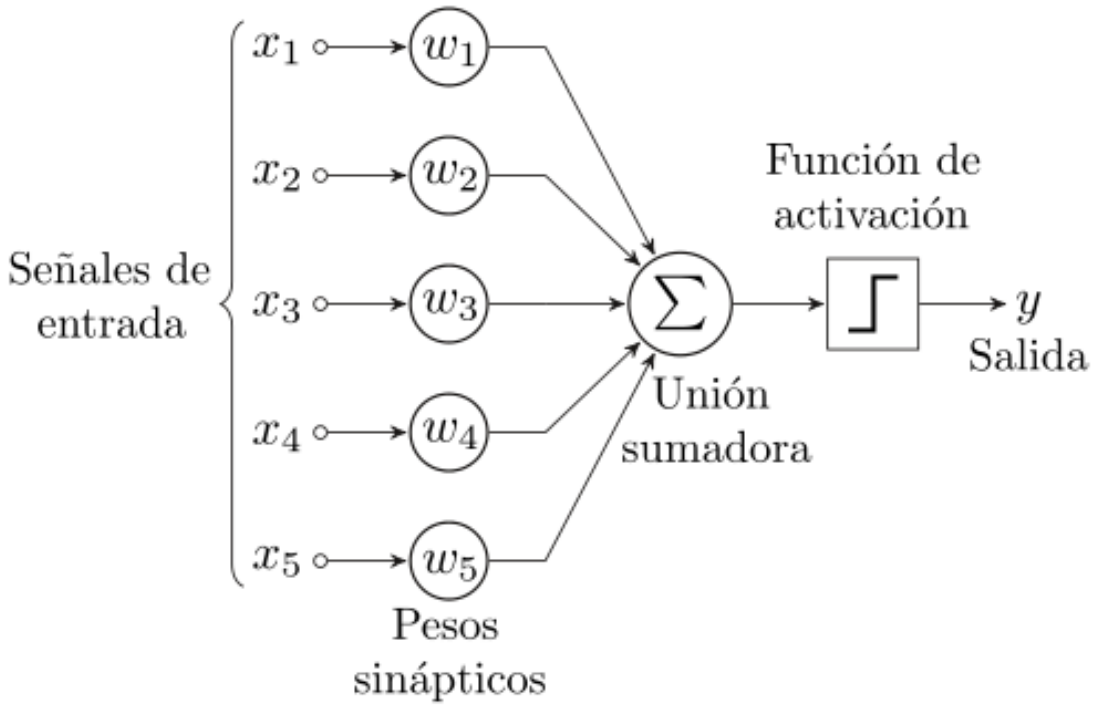
- *Multilayer Perceptrón (MLP)*
- *Long Short-Term Memory (LSTM)*
- *Redes convolucionales*
- *Redes Generativas (GANs)*

A pesar de la gran multitud de redes neuronales existentes, para la resolución de este problema se ha escogido el Multilayer Perceptrón, por lo que nos centraremos en este tipo.

2.3.1. Perceptrón multicapa (MLP):

Primero de todo, es necesario saber que el perceptrón simple es un algoritmo desarrollado por Frank Rosenblatt, el cual es capaz de clasificar elementos linealmente separables, generando un hiperplano que separe estos en n dimensiones. Esto quiere decir, que si dibujamos datos que previamente separamos en clases, digamos una clase C_1 y C_2, el algoritmo únicamente podrá clasificarlos si estos pueden ser separados por un plano.

La idea básica del perceptrón consiste un vector de entrada $x = [x_1, x_2, \ldots x_n]$, una unidad sumatoria, la función de evaluación y la salida de la misma, como se muestra en la siguiente figura:



Dónde su funcionamiento interno es tal y como sigue:

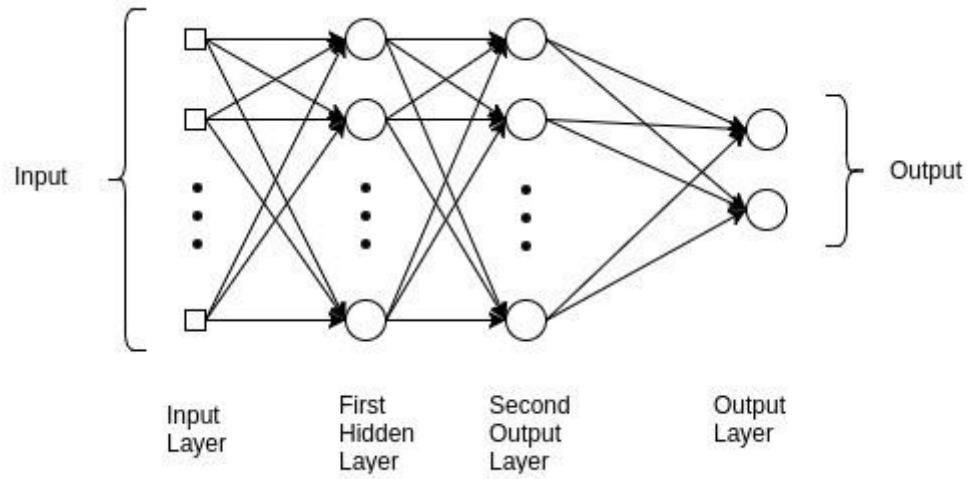
repeat **{**
error $= 0$;
for all $(n,\ 1 \leq n \leq N$ in a random way$)$ **{**
$g = c_n\ (\mathbf{w}^t \mathbf{x}_n + w_0)$;
if $(g < b)$ **{**
$\mathbf{w} = \mathbf{w} + \alpha\, c_n\, \mathbf{x}_n$; $w_0 = w_0 + \alpha\, c_n$;
error $++$;
}
}
} ***until*** (error $= 0$)

The final weights and threshold are: $\widehat{\mathbf{w}} = \sum_{n=1}^{N} \beta_n\ c_n\ \mathbf{x}_n$ and $\widehat{w_0} = \sum_{n=1}^{N} \beta_n\ c_n$

Como refleja la anterior imagen, el algoritmo se va iterando y actualizando los pesos en función de si el resultado obtenido en el clasificador es más pequeño que el margen previamente establecido.
Esta iteración recorre todas las muestras y se va repitiendo en caso de obtener algún error de clasificación en alguna de ellas.

Finalmente, una vez ejecutada una vuelta completa por todas las muestras sin obtener ningún error, se obtienen los pesos y el límite final de nuestro clasificador. De esta forma es como el propio algoritmo aprende a través de un conjunto de características.
Una vez entendido el funcionamiento del perceptrón simple, indagaremos un poco más en el tema y hablaremos del perceptrón multicapa o también llamado Red neuronal de alimentación directa (FFNN).

En el caso de trabajar con multicapa, nuestra red tendrá más de una capa lineal o de neuronas, generando la siguiente topología:



Tal y como podemos observar, la topología de cada capa sigue el mismo patrón que el perceptrón simple, pero en este caso vemos que disponemos de una capa de entrada, varias capas ocultas y una capa de salida.

En este caso el Multilayer perceptrón intenta aproximar la función $y = f(x)$, pero combinándola en las diferentes capas, quedando por ejemplo en un MLP de 3 capas de la siguiente forma, $f(x) = f(3) (f(2) (f(1) (x)))$.

Cada una de estas capas está compuesta de unidades que realizan una transformación afín de una suma lineal de entradas y cada capa se representa como $y = f(W \cdot xt + b)$. Donde f es la función de activación, W es el conjunto de parámetros o pesos en la capa, x es el vector de entrada, que también puede ser la salida de la capa anterior y b es el bias.

Las capas de un MLP se encuentran completamente conectadas porque cada unidad de una capa está conectada a todas las unidades de la capa anterior.

En una capa totalmente conectada, los parámetros de cada unidad son independientes del resto de las unidades, lo que significa que cada unidad posee un conjunto único de pesos.

En este problema, se utilizará el MLP como un sistema de regresión supervisado, donde cada vector de entrada estará etiquetado con su target, por lo que la salida de la capa de salida nos proporcionará una predicción para la entrada. Pero para poder medir el verdadero rendimiento de nuestro sistema, será necesario definir una función de pérdida, donde la pérdida será alta si la predicción se aleja mucho del valor real y de lo contrario será bajo.

Como función de pérdida se intentará minimizar el error cuadrático medio, sobre el cual se hablará más adelante en el momento de la definición de la red.

3. Conjunto de datos

3.1. Introducción a los datos

Se parte de un conjunto de datos con un tamaño de 173.883 líneas, las cuales corresponden a todas las transacciones realizadas en un cine de la ciudad de Lérida en el período comprendido entre 2011-2019.

Estas transacciones no se encuentran agrupadas por sesión y cada una de ellas puede hacer referencia a la compra o a la anulación de una o más entradas para una sesión en concreto.

Dichos datos han sido extraídos de una base de datos SQL mediante la aplicación Dynamics NAV gracias a un objeto llamado *Dataport*, el cual se encarga de recorrer tablas relacionadas y exportar a un fichero delimitado por comas o mejor dicho un fichero CSV, la información comprendida en dichas tablas.

Los datos iniciales tienen la siguiente estructura:

- *Código centro*: Identificador del cine, ya que en la base de datos hay datos transaccionales, para dos cines de Lérida y para otro de Valls.
- *Código sala*: Identificador de cada una de las salas de los cines.
- *Fecha sesión*: Consiste en la fecha en la que se emitió la película.
- *Hora sesión*: Consiste en la hora en la que se emitió la película.
- *ID Correcto*: Identificador de la película.
- *Cantidad*: Número de entradas que se han comprado o anulado en la transacción.
- *Importe IVA incl.*: Importe total con IVA incluido, de la transacción.
- *Descripción*: Descripción de la película a la que hace referencia la transacción.
- *Nombre de la película*: Nombre de la película a la que hace referencia la transacción.
- *Distribuidor*: Identificador del distribuidor que proporciona la película.
- *Nombre distribuidor*: Nombre del distribuidor que proporciona la película.
- *ID Original*: Identificador de la película, sin el prefijo añadido en la columna ID Correcto.

3.2. Agrupación de los datos

Tal y como se ha comentado, los datos no vienen agrupados, por lo que antes que nada es necesario agruparlos, para obtener el total de asistentes por cada sesión en concreto.

Esto se consigue agrupando las líneas mediante el script *group.py*, por la siguiente clave primaria:

'Código Sala – Fecha sesión – Hora sesión'

Una vez realizada la agrupación de las transacciones, se han encontrado líneas con un total de asistentes igual a 0, dado que algunas veces el cine se ha visto obligado a anular alguna de las sesiones, o también se han encontrado algunas líneas resultantes en negativo, las cuales hacen referencia a algún descuadre de la caja.

Para evitar distorsiones en el aprendizaje se han eliminado ambos tipos de transacciones resultantes de la agrupación.

Finalmente es necesario añadir también, que los días en los que hay Fiesta del Cine, la cual quiere decir que las entradas se encuentran en una cierta rebaja, generan mucha distorsión en el aprendizaje, ya que no siguen ninguna relación entre ellos.

Por lo tanto, también se ha decidido eliminar las entradas que hacen referencia a estos días y para lograr distinguirlos se ha utilizado un fichero llamado *events_category.json* del cual se hablará más adelante en la definición de los atributos de *Evento Normal* y *Evento Escolar*.

3.3. Definición de los atributos

En este apartado se presentarán los diferentes atributos que se han utilizado para el correcto aprendizaje de la red neuronal.

Añadir también que todo el conjunto de atributos que se verán a continuación, han sido fruto de una selección previa por parte del cliente.

3.3.1. Blockbuster

Inicialmente es necesario expresar uno de los requerimientos del cliente, ya que este ha definido la prioridad de abstraer lo máximo nuestro modelo de predicción del sector del cine, por lo que hemos decidido que únicamente incluiremos una columna que haga referencia a este sector.

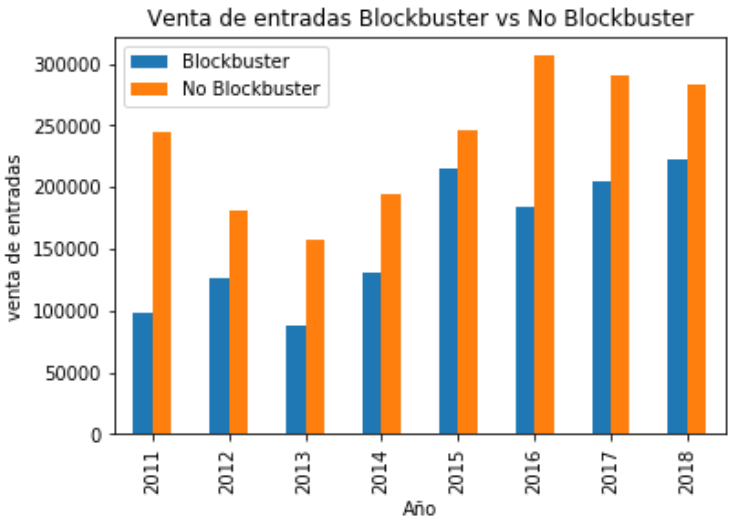
La columna se llamará *Blockbuster* y hará referencia a sí la película de la sesión pertenece a una película TOP en ventas. Según la información facilitada por el cliente, cada año se lanzan una media de 20 películas catalogadas como TOP, por lo que se escogerán 20 por año.

Para definir dicho ranking, se han recorrido todas las transacciones agrupadas, generando a su vez un contador de entradas vendidas por cada *ID Correcto* en un diccionario separado por años y de esta forma se han logrado etiquetar todos los registros.

Pero a pesar de que el cliente manifestó su profundo deseo de qué este fuera el único atributo relacionado con el mundo del cine y que era muy relevante para el aprendizaje, se ha querido comprobar que realmente es así.

Para ello, se han cogido las ventas de todas las sesiones de los diferentes años y se han agrupado por las ventas de sesiones *Blockbuster* y *No Blockbuster*, para poder comprobar la real magnitud del impacto de este tipo de películas.

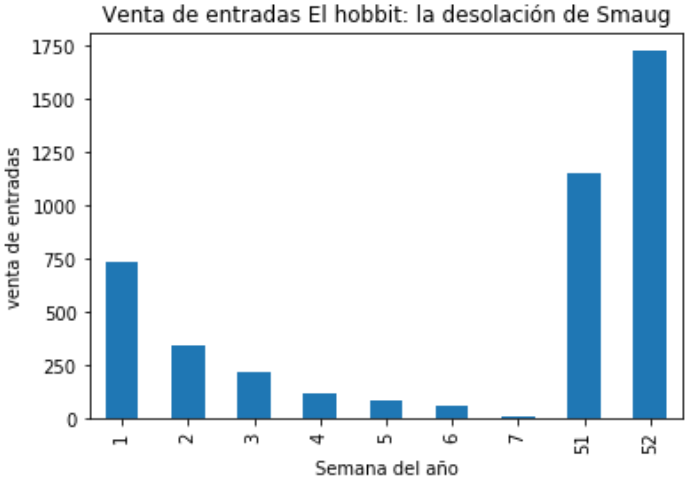
Una vez obtenidas las agrupaciones, se ha obtenido el siguiente gráfico:



*(**Nota**: No se ha sacado la venta de entradas para el año 2019, ya que como no ha terminado todavía la comparativa no sería realista.)*

Y efectivamente, podemos comprobar que la venta de las películas *Blockbuster* es más de la mitad de la venta de entradas del año, teniendo en cuenta que este tipo de películas únicamente son 20 por año.Pero dado un estudio previo, por parte del cliente, del impacto real de una película *Blockbuster* en la venta de entradas del cine, se ha detectado que cuando una de estas películas lleva más de dos semanas en taquilla su fuerte impacto disminuye considerablemente por lo que se puede dejar de considerar de este tipo.

A continuación, se muestra un gráfico donde se puede apreciar la disminución de impacto, pasadas las dos semanas:



Por lo tanto, también se ha tenido en cuenta este factor y en el momento de etiquetar cada sesión como *Blockbuster* o no, se ha comprobado que la película que se emite en dicha sesión no lleve más de dos semanas en taquilla y de ser así se ha etiquetado como una sesión normal.

3.3.2. Time-Slot

Como se ha comentado anteriormente, se trata de abstraer lo máximo la predicción del sector cine, por lo que el objetivo no será predecir la asistencia por sesión y sala concretas, sino que se tratará de predecir el número de asistentes en el cine por franjas horarias o time-slot, que será el nombre que se utilizará para esta agrupación.

El cliente ha comentado, que el horario habitual del cine y por lo tanto sobre el que quiere predecir, es de 15:00h a 22:00h (Lunes…Jueves) y de 15:00h a 00:00h (Viernes…Domingo), aunque en días especiales este incluya también sesiones matinales. Por lo que acordamos descartar también todas aquellas sesiones que no se encuentren en el horario habitual del cine.

En cuanto a la agrupación del time-slot, se han generado varios conjuntos de datos con diferentes time-slots, para posteriormente realizar entrenamientos en la red neuronal con diferentes particiones horarias de los datos.

Quedando las siguientes particiones:

- *Time-slots horario*: En este caso encontramos que cada día disponemos de 10 time-slots durante la semana y de 12 time-slots para el viernes, sábado y domingo. Quedándonos de esta forma un conjunto de datos con un total de 26447 líneas.

- *Time-slots de 2 horas*: En este caso encontramos que cada día disponemos de 5 time-slots durante la semana y de 6 para el viernes, sábado y domingo. Quedándonos de esta forma un conjunto de datos con un total de 13224 líneas.

- *Time-slots de 4 horas*: En este caso encontramos que cada día disponemos de 3 time-slots durante la semana, siendo más pequeño el último ya que únicamente contiene 2 horas, y de 3 time-slots para el viernes, sábado y domingo, del mismo tamaño. Quedándonos de esta forma un conjunto total de 6612 líneas.

Una vez decidido el tamaño del time-slot, se han agrupado las líneas según la siguiente clave primaria:

'Fecha sesión – Time-slot'

Donde cada línea se ha considerado dentro de un time-slot, según su columna *Hora de sesión*.

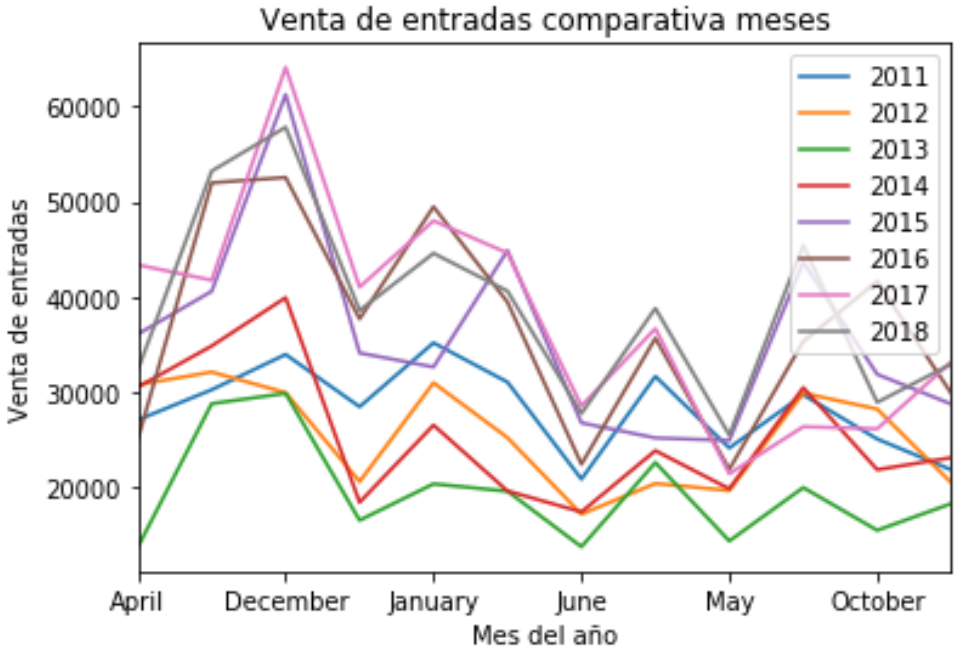
Una vez obtenidos los diferentes times-slots, ya no son necesarias parte de las columnas que se encuentran en el Dataset inicial ya que no son relevantes, por lo que es momento de eliminarlas:

'*Código centro*', '*Código sala*', '*Hora sesión*', '*ID Correcto*', '*Importe IVA incl*', '*Descripción*', '*Nombre de la película*', '*Distribuidor*', '*Nombre distribuidor*' y '*ID original*'.

3.3.3. Mes

También se ha decidido añadir como atributo, el mes al que corresponde el time-slot en cuestión ya que este da más información a la red de en que momento del año se encuentra la sesión.

Pero para salir de dudas se ha analizado la venta de entradas en los diferentes meses de los años y entre los diferentes años, y se ha comprobado que los meses guardan cierta relación entre ellos:

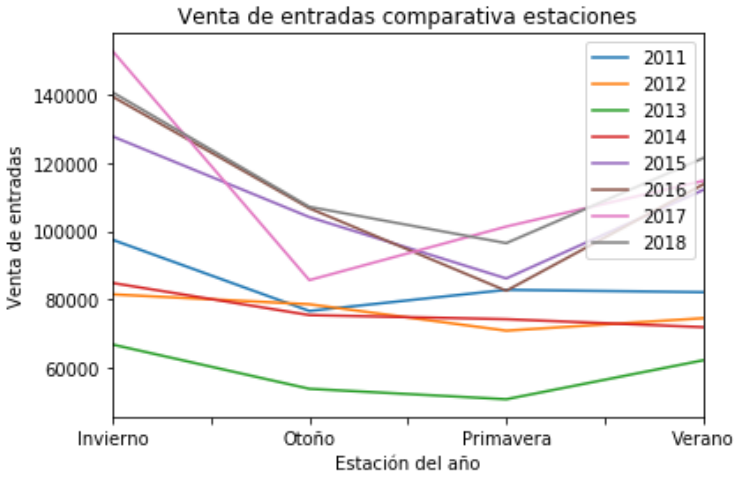


*(**Nota**: No se ha sacado la venta de entradas para el año 2019, ya que como no ha terminado todavía la comparativa no sería realista.)*

Tal y como podemos apreciar en la anterior figura, comprobamos que cada mes tiene cierta relación en cuanto a los picos de subida o bajada de venta de entradas, siendo estos picos mayores o inferiores en función del rendimiento anual al que corresponden.

3.3.4. Estación

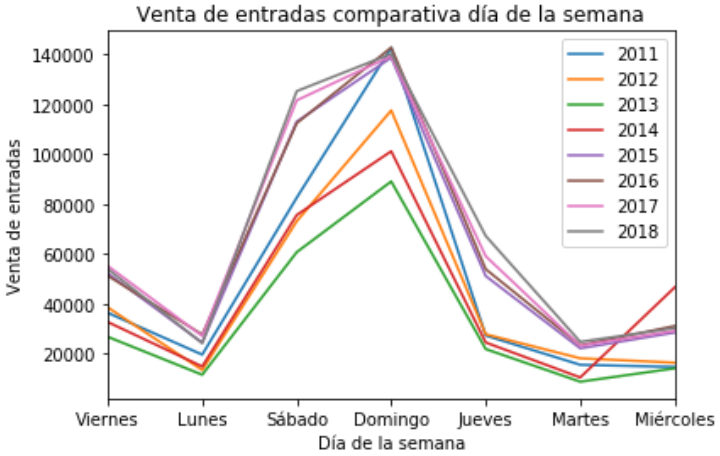
En cuanto el atributo estación, inicialmente no se había tenido en cuenta, pero al realizar el estudio de la venta de entradas por meses y poder comprobar que guardan cierta relación, pensé en también realizar la misma comprobación para las estaciones del año y por sorpresa mía, pude comprobar que estas también guardan cierta relación, aunque no tan fuerte como las semanas:



*(**Nota**: No se ha sacado la venta de entradas para el año 2019, ya que como no ha terminado todavía la comparativa no sería realista.)*

3.3.5. Día de la semana

En este caso, podríamos pensar que estamos hablando de un atributo del mismo estilo que el atributo mes o estación, pero mediante el siguiente gráfico se puede comprobar que la relación de la venta de entradas con el día de la semana, es sumamente más alta:



*(**Nota**: No se ha sacado la venta de entradas para el año 2019, ya que como no ha terminado todavía la comparativa no sería realista.)*

3.3.6. Tiempo

Para la obtención de este tipo de atributos, se ha hecho uso de una API *freemium* llamada *World Weather API*, la cual facilita mucho su extracción de datos ya que nos permite hacerlo mediante un script en *Python*.
Para ello se ha diseñado un script, el cual se encarga de hacer un conjunto de solicitudes que corresponden a cada uno de los meses de los diferentes años, con la siguiente estructura:

Parameter	Description	Required	Values
q	Location	Required	See note below.
extra	Include extra information	Optional	See note below.
date	The date to return the weather for.	Required	yyyy-MM-dd (Example: 2009-07-20 for 20 July 2009.)
enddate	If you wish to retrieve weather between two dates, use this parameter to specify the ending date.**Important:** the *enddate* parameter must have the same month and year as the *date* parameter.	Optional	yyyy-MM-dd (Example: 2009-07-22 for 22 July 2009.)
includelocation	Whether to return the nearest weather point for which the weather data is returned for a given postcode, zipcode and lat/lon values.	Optional	Valid values: •yes •no (default)
tp	Specifies the weather forecast time interval in hours. Options are: 1 hour, 3 hourly, 6 hourly, 12 hourly (day/night) or 24 hourly (day average).	Optional	Valid values: •1 •3 (default) •6 •12 •24
format	The output format to return. XML or JSON.	Optional	Valid values: •xml (default) •json
callback	The function name for JSON callback.	Optional	Example: callback=function_name
key	The API key.	Required	Provided when registering your application.

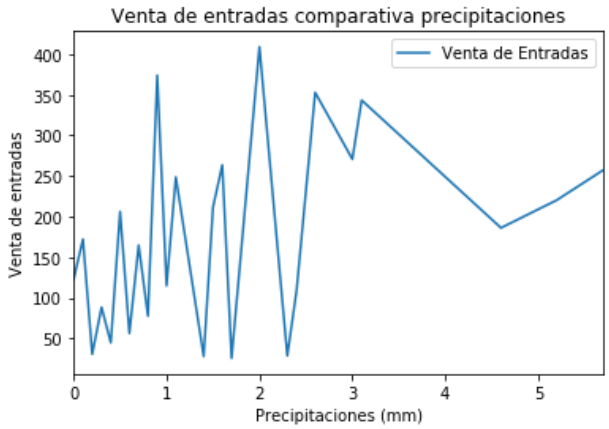
Y obteniendo como respuesta la siguiente respuesta:

Value	Description	Type
astronomy	Astronomical condition for the day.	astronomy element
date	Local forecast date	Date in yyyy-MM-dd. Example: 2013-05-31
maxtempC	Maximum temperature of the day in degree Celsius	Integer
maxtempF	Maximum temperature of the day in degree Fahrenheit	Integer
mintempC	Minimum temperature of the day in degree Celsius	Integer
mintempF	Minimum temperature of the day in degree Fahrenheit	Integer
uvIndex	UV Index	Integer
totalSnow_cm	Total snowfall amount in cm	Float
sunHour	Total sun hour	Float
hourly	Hourly weather conditions. Can have multiple.	hourly element

A través de la cual, se ha extraído el atributo *hourly* y se ha obtenido la temperatura y las precipitaciones por horas.

3.3.7. Precipitaciones

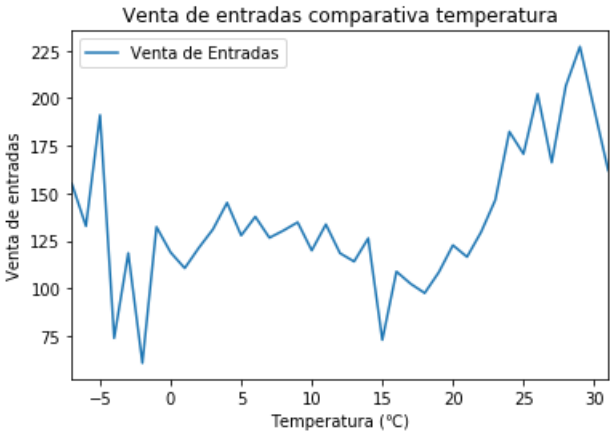
Dicho atributo corresponde a la cantidad de lluvia que ha caído en milímetros, durante el time-slot en cuestión. Podríamos pensar que este atributo no es realmente relevante para el problema a solucionar, pero la siguiente gráfica nos demuestra lo contrario:



En el cual se puede comprobar que la tendencia de los clientes del cine, consiste en consumir más cine en los momentos lluviosos.

3.3.8. Temperatura

Dicho atributo corresponde a la temperatura en grados centígrados, durante el time-slot en cuestión. De igual forma que el atributo anterior, nos proporciona cierto conocimiento en cuanto a la venta de entradas, ya que como podemos ver a continuación:



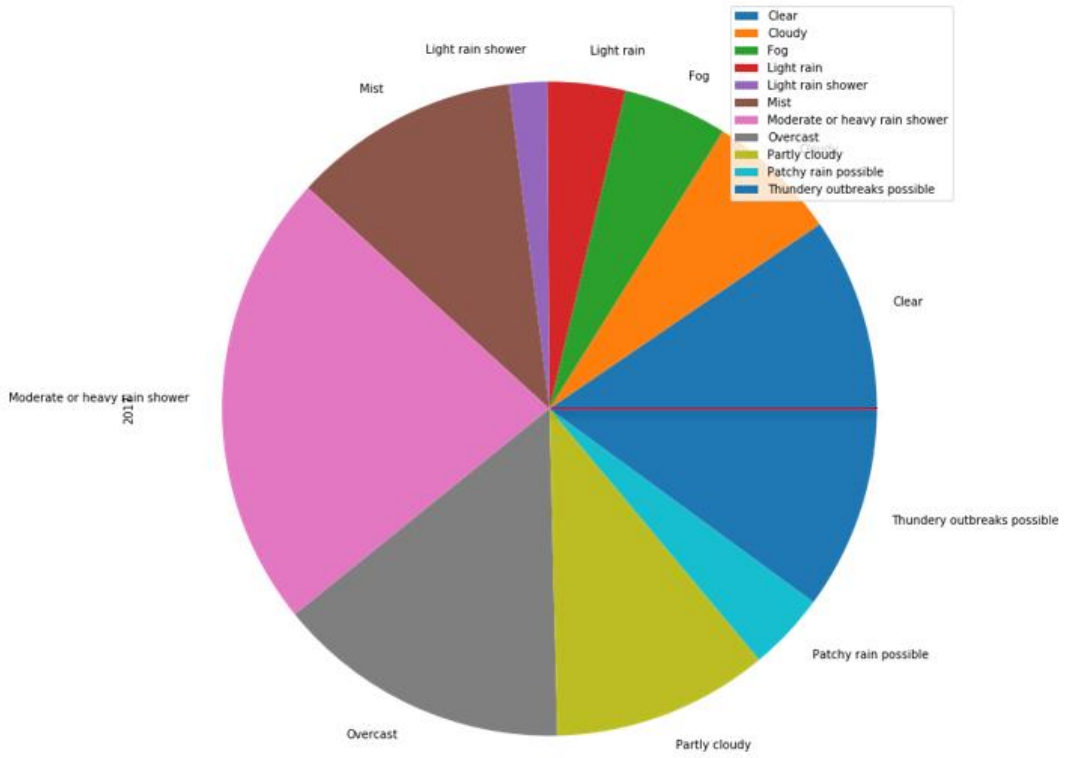
La gente se decide a ir más al cine en temperaturas extremas de calor o frío.

3.3.9. Descripción tiempo

Dicho atributo hace referencia a una palabra descriptiva del tiempo, la cual proporciona la propia API, donde el conjunto de descripciones es limitado y es el siguiente:

- *Clear*
- *Cloudy*
- *Fog*
- *Light rain*
- *Light rain shower*
- *Mist*
- *Moderate or heavy rain shower*
- *Overcast*
- *Partly cloudy*
- *Patchy rain posible*
- *Thundery outbreaks possible*

Para comprobar si los estudios hechos hasta el momento en cuanto al tiempo, se adecuan con este, se ha sacado una gráfica del año en el que se han producido más descripciones de tiempo diferentes:



Y efectivamente podemos comprobar, que tal y como habíamos asumido anteriormente, los días lluviosos son los preferidos por los habitantes de la ciudad de Lérida para escoger ir al cine.

3.3.10. Festivo nacional

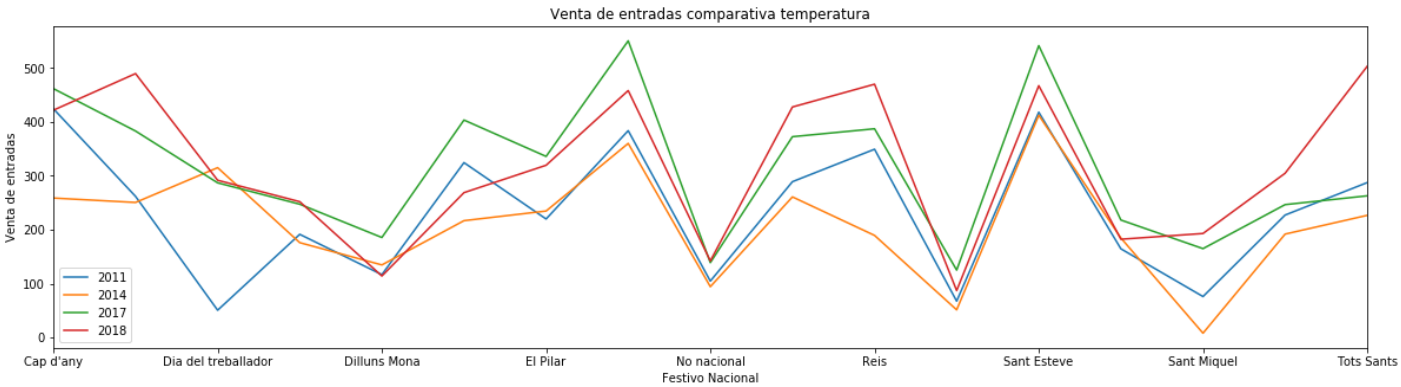
Este atributo hace referencia a esos días que son festivos en el país de España y por tanto nacionales.
Para la obtención de estas fechas, se ha generado manualmente un fichero holidays.*json*, con la siguiente estructura:

{
"Nombre festivo":
[Conjunto de fechas para el festivo],
"Nombre festivo":
.
.
.
}

Dentro de los festivos nacionales se han contemplado los siguientes:

- *Fin de año*
- *Reyes*
- *Viernes Santo*
- *Lunes de pascua*
- *Día del trabajador*
- *San Juan*
- *Santa María*
- *Diada de Cataluña*
- *El Pilar*
- *Todos Santos*
- *Constitución Española*
- *Purísima de la Concepción*
- *Navidad*
- *San Esteve*
- *San Anastasi (Festivo en Lérida, considerado como nacional)*
- *San Miquel (Festivo en Lérida, considerado como nacional)*

Para esta columna, también se ha realizado un estudio para comprobar cual, es el impacto real de las fiestas nacionales en el sector del cine.
Obteniendo como resultado la siguiente gráfica:



En la cual podemos comprobar que cada uno de los festivos nacionales, más o menos tiene el mismo comportamiento indiferentemente del año y que tiene un impacto fuerte en cuanto a la venta de entradas, ya que se vende más entradas de media en los días de festivos nacionales que en los días normales.

Se han descartado varios años para la representación de la gráfica, ya que esos años el cine no abrió en uno de los festivos nacionales.

3.3.11. Evento normal

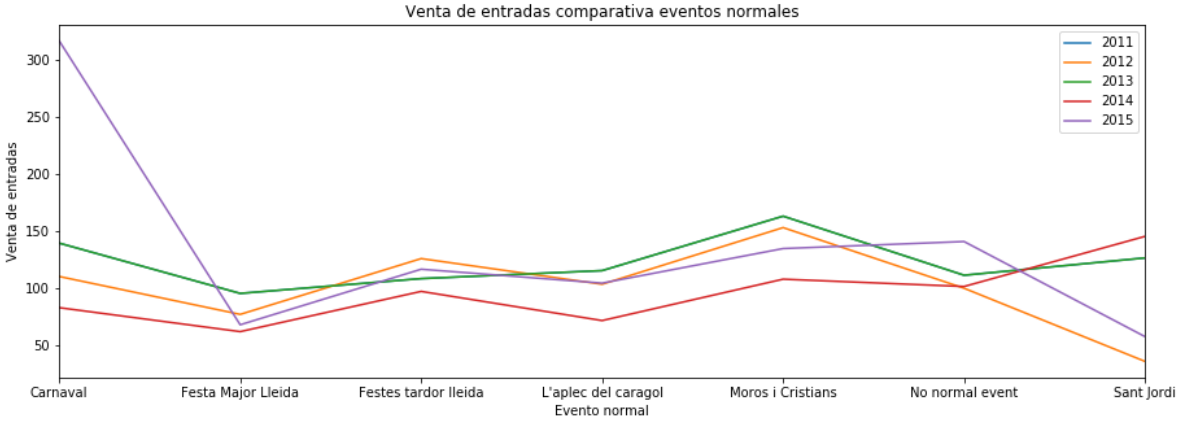
Cuando se habla de evento normal, hace referencia a todos esos días o conjunto de días, que suponen un día no normal en la ciudad de Lérida. Al decir no normal, se hace referencia a días que reflejan algún tipo de festividad para la ciudad, aunque eso no quiera decir días no laborables para los ciudadanos. Para la obtención de estas fechas, se ha generado manualmente un fichero *events_category.json*, con la siguiente estructura:

```
{
        "Normal":{
                "Nombre evento":
                        [ Conjunto de fechas para el festivo ],
                }
        "Otros tipos de eventos":{
        }
}
```

Dentro de los eventos normales se han contemplado los siguientes:

- *Carnaval*
- *Moros y Cristianos*
- *Fiestas de otoño en Lérida*
- *Fiesta mayor de Lérida*
- *El aplec del caracol.*
- *San José.*

Al igual que en todos los atributos anteriores, también se ha realizado un estudio para comprobar si el impacto de dichos eventos es relevante en cuanto al cine, para posteriormente escoger el atributo como válido.



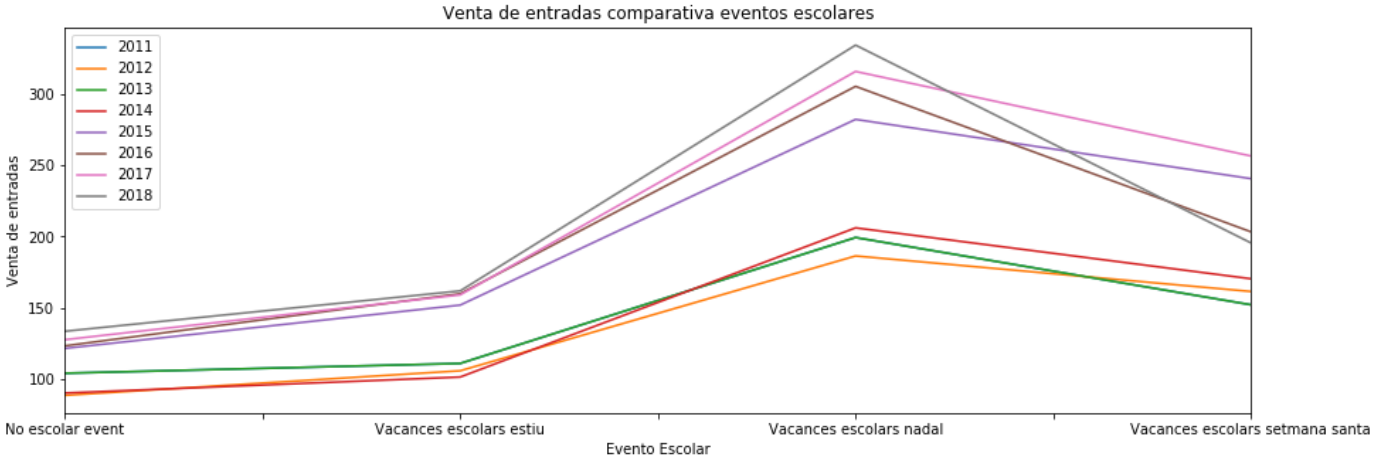
Como se puede ver en el gráfico anterior, cada evento normal tiene cierta relación en cuanto a la venta de entradas con sus años anteriores, ya que los picos de subida o bajada están relacionados.
Haciendo referencia al impacto en la venta de entradas, este no es muy agresivo ya que la media de venta es muy similar en cuanto es un evento normal o un día normal.
Se han descartado varios años para la representación de la gráfica, ya que esos años el cine no abrió en uno de los eventos normales.

3.3.12. Evento escolar

Dicho atributo hace referencia a ese conjunto de días en que en las escuelas es festivo, es decir que son vacaciones escolares.
Para la obtención de dichas fechas, se ha utilizado el mismo fichero que en el caso de los eventos normales, pero se han insertado en el apartado de "*Escolares*" en lugar de "*Normal*".
En este tipo de vacaciones distinguimos las siguientes 3:

- *Vacaciones escolares de verano.*
- *Vacaciones escolares de navidad.*
- *Vacaciones escolares de semana santa.*

De igual forma que en todos los otros atributos, se ha realizado un estudio para comprobar el
impacto de este tipo de días en el cine y para ello se ha sacado la siguiente gráfica:



Inicialmente, se puede comprobar que el comportamiento de cada uno de los eventos escolares sigue la misma relación en los diferentes años en cuanto a los picos de subida y de bajada de venta. También podemos comprobar que el impacto es fuerte, ya que se han vendido más entradas de media en cualquiera de los días de este tipo que en los días normales y que sobre todo el evento más fuerte es durante las vacaciones escolares de navidad.

3.3.13. Semana del año

Este atributo hace referencia al número de semana del año, teniendo en cuenta que en el año tenemos un total de 52 semanas en lo común, se encontrará en el siguiente rango *[1…52]*.
Inicialmente podríamos pensar que se trata de un atributo que proporciona muy poca información para el posterior aprendizaje, pero para asumir dicha asunción es necesario previamente realizar un estudio de su comportamiento.
Para ello se ha sacado la siguiente gráfica:

En la cual podemos comprobar que la relación no es sumamente grande, ya que también influye mucho el tipo de películas que se han emitido durante esas semanas en sus diferentes años, pero vemos que muchos picos sí que siguen una relación, al igual que tal y como hemos comprobado anteriormente en los eventos escolares, cuando se venden más entradas es en las últimas y primeras semanas del año, pertenecientes a navidades.
Por lo que finalmente me he decidido por mantener este atributo.

3.4. Tratamiento de los atributos

En este apartado se describirá el tratado previo completo que se ha realizado de los datos, para que estos se adapten correctamente a la red neuronal y de esta forma que sea más fácil aprender de ellos. Pero para ello, inicialmente es necesario entender un poco qué tipo de atributos existen y que posibilidades de tratamiento tienen.

Cuando hablamos del concepto '*Data mining*', generalmente discutimos sobre el descubrimiento de conocimiento a partir de datos.

El '*data mining*' incluye conocer los datos y encontrar la relación entre ellos mismos. Pero para conocer los datos es necesario discutir sobre estos objetos de datos, atributos de datos y tipos de atributos de datos.

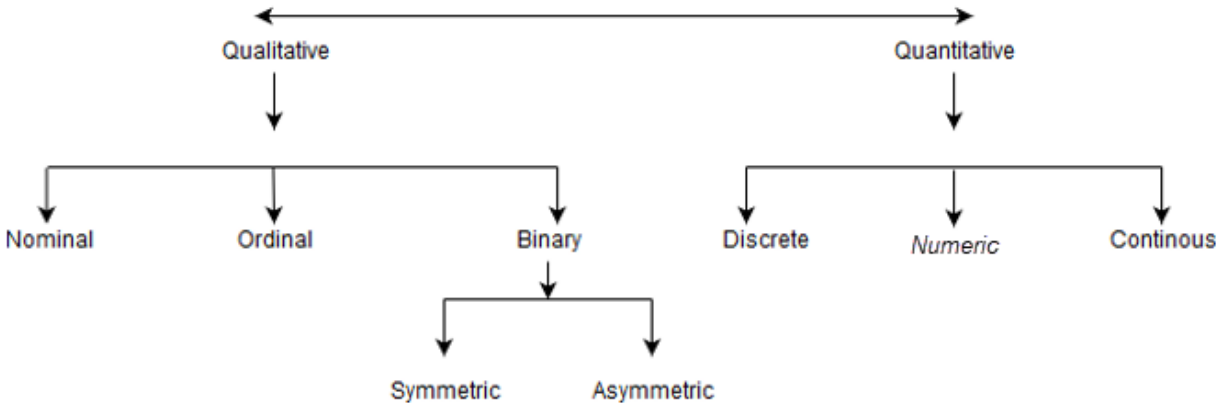
Los objetos de datos son la parte esencial de una base de datos. Un objeto de datos representa la entidad y es como un grupo de atributos de una entidad. Por ejemplo, un objeto de datos de ventas puede representar clientes, ventas o compras. Cuando un objeto de datos aparece en una base de datos, se denomina tuplas de datos.

Se puede ver como un campo de datos que representa características o características de un objeto de datos. Para un objeto de cliente, los atributos pueden ser ID. de cliente, dirección, etc… Podemos decir que un conjunto de atributos utilizados para describir un objeto dado se conoce como vector de atributos o vector de características.

3.4.1. Tipos

Estos atributos pueden ser de diferentes tipos, tal y como se ha hablado en el apartado del estado del arte:

- Cualitativo: Nominal, Ordinal o Binario.
- Cuantitativo: Discreto o continuo.



3.4.1.1. Atributos Cualitativos

- Atributos nominales: Los valores de un atributo nominal son nombres de cosas, algún tipo de símbolos. Los valores de los atributos nominales representan alguna categoría o estado y es por eso que el atributo nominal también se conoce como atributo categórico y no existe un orden (rango, posición) entre los valores del atributo nominal.
- A continuación, se muestra un ejemplo:

Atributo	Valores
Colores	Rojo, Lila, Negro
Datos categóricos	Pintor, Mecánico, Cocinero.

- **Atributos binarios**: Los datos binarios tienen solo 2 valores / estados. Por ejemplo, sí o no, afectado o no afectado, verdadero o falso. Y estos pueden ser de dos formas:

    - **Simétricos**: Ambos valores son igualmente importantes, por ejemplo, en el caso del género.

Atributo	Valores
Género	Masculino, Femenino

    - **Asimétricos**: Ambos valores no son igualmente importantes, por ejemplo, resultado final.

Atributo	Valores
Gripe	Sí, No
Resultado	Ganador, Perdedor

3.4.1.2. Atributos Cuantitativos

- **Atributos numéricos**: Un atributo numérico es cuantitativo porque es una cantidad medible, representada en valores enteros o reales.
  Los atributos numéricos son de 2 tipos:

    - **Intervalo**: Un atributo a escala de intervalo tiene valores, cuyas diferencias son interpretables, pero los atributos numéricos no tienen el punto de referencia correcto o el cual podemos llamar punto cero. Los datos se pueden sumar y restar a escala de intervalo, pero no se pueden multiplicar o dividir.
      Por ejemplo, consideramos un ejemplo de temperatura en grados centígrados. Si la temperatura de un día es el doble que la del otro día, no podemos decir que un día sea dos veces más caluroso que otro día.

    - **Relación**: Un atributo a escala proporcional es un atributo numérico con un punto cero fijo. Si una medición tiene una escala proporcional, podemos decir que un valor es un múltiple (o relación) de otro valor. Los valores están ordenados, y también podemos calcular la diferencia entre los valores, y se puede dar la media, la mediana, la moda, el rango de cuantiles y el resumen de cinco números.
      Por ejemplo, un tipo de datos relación podrían ser las cantidades monetarias, ya que sí que podemos que un producto que vale el doble que otro, es el doble de caro.

- **Atributos discretos**: Los datos discretos tienen valores finitos, pueden ser numéricos y también en forma categórica. Tienen un número finito o infinito contable de valores. Como podrían ser una profesión o los códigos postales.

Atributo	Valroes
Géneros musicales	Pop, Rock, Electrónica.
Código Postal	25005…99990

- **Atributos continuos**: Los datos continuos tienen un número infinito de estados y son de tipo flotante. Como podrían ser el peso o la altura.

Atributo	Valores
Altura	6.8, 9.7, etc…
Peso	45.5, 76.5, etc…

3.4.2. Normalización

La normalización es una técnica que a menudo se aplica como parte de la preparación de datos para el aprendizaje automático.

El objetivo de la normalización es cambiar los valores de las columnas o atributos numéricos en el conjunto de datos a una escala común, sin distorsionar las diferencias en los rangos de valores.

Cabe recalcar que, para el aprendizaje automático, cada conjunto de datos no requiere normalización, ya que solo lo requieren las características que tienen diferentes rangos. Cuando hablamos de normalización de características, los dos métodos más comunes son dos:

- ***Estandarización***: El resultado de la estandarización (o normalización Z-score) es que las características serán reescaladas para que tengan las propiedades de una distribución normal estándar con μ=0 y σ=1, dónde μ es la media (promedio) de los valores de la característica y σ es la desviación estándar de la media. Por lo que los valores normalizados, se calculan mediante la siguiente fórmula:

$$z = \frac{x - \mu}{\sigma}$$

Donde x es el valor de la característica a normalizar y z es el valor normalizado. Estandarizar las características para que se centren en 0 con una desviación estándar de 1, no solo es importante si estamos comparando mediciones que tienen unidades diferentes, sino que también es un requisito general para muchos algoritmos de aprendizaje automático.
Intuitivamente, podemos pensar en el descenso de gradiente como un ejemplo destacado (un algoritmo de optimización utilizado a menudo en regresión

logística, SVM, perceptrones, redes neuronales, etc…) con características en escalas diferentes, en el que ciertos pesos pueden actualizarse más rápido que otros, ya que los valores de las características x_j desempeñan un papel en las actualizaciones de los pesos:

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j} = \eta \sum_i (t^{(i)} - o^{(i)}) x_j^{(i)},$$

Por lo tanto, tenemos $W_j = wj+\Delta wj$, donde η es el learning rate, t la etiqueta de la clase objetivo y o la salida real.

- ***Min-Max***: Un enfoque alternativo para la normalización (o estandarización) es la llamada escala Min-Max.

  En este enfoque, los datos se escalan en un rango fijo que generalmente va de 0 a 1.
  El costo de tener este rango limitado, en contraste con la estandarización, es que terminaremos con desviaciones estándar más pequeñas, que pueden suprimir el efecto de los valores atípicos.
  Una escala Min-Max normalmente se realiza mediante la siguiente ecuación:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

  Donde X es el valor a normalizar, X_{min} es el mínimo valor de esa característica y X_{max} es el máximo valor de esa característica.

  Para resolver el problema en cuestión, se ha utilizado la normalización Min-Max, por lo que a continuación se presentarán los atributos que han sufrido dicha normalización.

3.4.2.1. Blockbuster

Dicho atributo, forma parte de los atributos cuantitativos y dentro de estos entraría en el grupo de los atributos discretos, ya que consta de valores finitos y es número entero.

Cuando decimos que consta de valores finitos, es porque este atributo puede coger los valores desde 0 a 7, dónde 7 es el máximo número de sesiones simultáneas que se pueden dar en un time-slot.

Aplicando la fórmula anterior comentada, nos quedaría una normalización como la siguiente:

X - sin normalizar	X - normalizado
0	0.0
1	0.143
2	0.286
3	0.429
4	0.571
5	0.714
6	0.857
7	1.0

3.4.2.2. Temperatura

En este caso, la característica pertenece a los atributos cuantitativos y dentro de estos se encuentra en el grupo de los atributos continuos, ya que puede tener un número infinito de estados y es de tipo flotante.

Se dice que puede coger un número infinito de estados, ya que, al tratarse de un número flotante, podemos encontrar temperaturas como 30.4, 20.54, etc… aunque no es muy común en temperaturas trabajar con un elevado número de decimales.

En este caso tenemos como valor máximo el 31 y como valor mínimo el -7, por lo qué aplicando la fórmula comentada anteriormente obtenemos una normalización como la siguiente:

X – sin normalizar	X - normalizado
-7	0.0
0	0.184
5	0.315
…	…
15	0.578
20	0.710
31	1.0

3.4.2.3. Precipitación

De igual forma que el atributo de temperatura, esta característica pertenece a los atributos cuantitativos y dentro de estos también está en el grupo de los atributos continuos, ya que puede tener un número infinito de estados y es de tipo flotante.
En este caso, en dicho atributo si que es más común encontrarlo con un número más elevado de decimales.

Los valores máximos y mínimos son 5.7 y 0.0 respectivamente, por lo qué aplicando la fórmula descrita anteriormente, obtenemos una normalización como la siguiente:

X – sin normalizar	X - normalizado
0.0	0.0
1.3	0.228
2.7	0.473
…	…
3.5	0.614
4.8	0,842
5.7	1.0

3.4.3. Codificación

La codificación de igual forma que la normalización es una técnica que se usa muy a menudo como parte de la preparación de los datos para el posterior aprendizaje, pero en este caso se aplica a las variables categóricas.

Los datos categóricos son comunes en muchos problemas de Data Science y Machine Learning, pero generalmente son más difíciles de manejar que los datos numéricos. En particular, muchos algoritmos de aprendizaje automático requieren que su entrada sea numérica y, por lo tanto, las características categóricas deben transformarse en características numéricas antes de poder usar cualquiera de estos algoritmos.

Por lo tanto, es necesario hacer uso de varias técnicas para lograr transformar las etiquetas en valores numéricos.

A continuación, se hará una breve introducción a las técnicas utilizadas para abordar el problema:

- **Codificación one-hot**: Dicha codificación es el enfoque clásico para tratar con datos nominales y tal vez ordinales.
  El codificador one-hot crea una columna para cada valor para comparar con todos los demás valores. Para cada nueva columna, una fila obtiene un 1 si la fila contenía el valor de esa columna y un 0 si no lo tenía.
  Por ejemplo, si queremos codificar la columna fruta y esta columna contiene los valores plátano, manzana y cereza, por ese orden, quedaría de la siguiente forma:

Fruta_Plátano	Fruta_Manzana	Fruta_Cereza
1	0	0
0	1	0
0	0	1

- **Codificación binaria**: Este codificador se puede considerar como un híbrido de codificadores de hash y one-hot. En este caso, se asigna un valor entero a cada una de las etiquetas diferentes de la característica y una vez asignados todos los valores, estos se codifican en binario.
  Este tipo de codificador puede funcionar muy bien en características ordinales de mayor dimensionalidad, ya que no crea tantas columnas como el codificador one-hot.

  Siguiendo con el ejemplo anterior, en este caso la codificación quedaría de la siguiente forma:

Fruta_1	Fruta_2
0	1
1	0
1	1

3.4.3.1. Mes

Dicho atributo forma parte de los atributos cualitativos y dentro de este grupo se encuentra en el grupo de los atributos nominales, por eso se ha utilizado el codificador one-hot para su codificación.

Los valores posibles son 12 y hacen referencia al nombre de cada uno de los meses del año.

A continuación, se muestra una tabla que representa la codificación final de dicho atributo:

	Mes_Enero	Mes_Febrero	Mes_Marzo	Mes_Abril	Mes_Mayo	Mes_Junio	Mes_Julio	Mes_Agosto	Mes_Septiembre	Mes_Octubre	Mes_Noviembre	Mes_Diciembre
Enero	1	0	0	0	0	0	0	0	0	0	0	0
Febrero	0	1	0	0	0	0	0	0	0	0	0	0
Marzo	0	0	1	0	0	0	0	0	0	0	0	0
Abril	0	0	0	1	0	0	0	0	0	0	0	0
Mayo	0	0	0	0	1	0	0	0	0	0	0	0
Junio	0	0	0	0	0	1	0	0	0	0	0	0
Julio	0	0	0	0	0	0	1	0	0	0	0	0
Agosto	0	0	0	0	0	0	0	1	0	0	0	0
Septiembre	0	0	0	0	0	0	0	0	1	0	0	0
Octubre	0	0	0	0	0	0	0	0	0	1	0	0
Noviembre	0	0	0	0	0	0	0	0	0	0	1	0
Diciembre	0	0	0	0	0	0	0	0	0	0	0	1

Transformando la característica mes en 12 características.

3.4.3.2. Time-slot

Al igual que el atributo anterior, el time-slot es cualitativo y nominal, por lo que también se ha utilizado el codificador one-hot.

En este caso, tal y como se ha comentado en el apartado de la definición de los atributos, se han realizado 3 particiones de las franjas horarias en el cine, pero a continuación se hablará para la partición horaria.
Dicha partición contiene un total de 10 etiquetas diferentes y mediante la codificación quedan definidas de la siguiente forma:

	timeslot_15	timeslot_16	timeslot_17	timeslot_18	timeslot_19	timeslot_20	timeslot_21	timeslot_22	timeslot_23	timeslot_24
15	1	0	0	0	0	0	0	0	0	0
16	0	1	0	0	0	0	0	0	0	0
17	0	0	1	0	0	0	0	0	0	0
18	0	0	0	1	0	0	0	0	0	0
19	0	0	0	0	1	0	0	0	0	0
20	0	0	0	0	0	1	0	0	0	0
21	0	0	0	0	0	0	1	0	0	0
22	0	0	0	0	0	0	0	1	0	0
23	0	0	0	0	0	0	0	0	1	0
24	0	0	0	0	0	0	0	0	0	1

Transformando la característica time-slot en 10 características.

3.4.3.3. Estación

Al igual que los atributos anteriores, el time-slot es cualitativo y nominal, por lo que también se ha utilizado el codificador one-hot.

En este caso el atributo contiene 4 valores que pertenecen a las estaciones del año, y una vez codificadas quedan de la siguiente forma representadas:

	Estacion_Verano	Estacion_Otoño	Estacion_Invierno	Estacion_Primavera
Verano	1	0	0	0
Otoño	0	1	0	0
Invierno	0	0	1	0
Primavera	0	0	0	1

Transformando la característica estación en 4 características.

3.4.3.4. Día de la semana

Este atributo también pertenece al grupo de los anteriores, por lo que se ha utilizado el codificador one-hot.

En este caso contiene 7 valores que pertenecen a los días de la semana, y una vez codificados quedan de la siguiente forma representados:

	Diadelasemana_Lunes	Diadelasemana_Martes	Diadelasemana_Miercoles	Diadelasemana_Jueves	Diadelasemana_Viernes	Diadelasemana_Sábado	Diadelasemana_Domingo
Lunes	1	0	0	0	0	0	0
Martes	0	1	0	0	0	0	0
Miercoles	0	0	1	0	0	0	0
Jueves	0	0	0	1	0	0	0
Viernes	0	0	0	0	1	0	0
Sábado	0	0	0	0	0	1	0
Domingo	0	0	0	0	0	0	1

Transformando la característica día de la semana en 7 características.

3.4.3.5. Festivo nacional

Dicho atributo sigue el patrón de los anteriores, por lo que también se ha utilizado el codificador one-hot para codificarlo.

En este caso tal y como se ha comentado en su definición tiene 17 valores, contando la etiqueta no festivo nacional, y una vez codificados quedan de la siguiente forma representados:

	Fnacional_1	Fnacional_2	Fnacional_3	Fnacional_4	Fnacional_5	...	Fnacional_14	Fnacional_15	Fnacional_16	Fnacional_17
Fin de año	1	0	0	0	0	...	0	0	0	0
Reyes	0	1	0	0	0	...	0	0	0	0
V. Santo	0	0	1	0	0	...	0	0	0	0
Pascua	0	0	0	1	0	...	0	0	0	0
El Pilar	0	0	0	0	1	...	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...
Naviidad	0	0	0	0	0	...	1	0	0	0
San Esteve	0	0	0	0	0	...	0	1	0	0
Todos Santos	0	0	0	0	0	...	0	0	1	0
San Juán	0	0	0	0	0	...	0	0	0	1

Transformando la característica festivo nacional en 17 características.

3.4.3.6. Evento escolar

Este atributo también sigue el mismo patrón de los anteriores, por lo que también se ha utilizado el codificador one-hot para codificarlo.

En este caso tal y como se ha comentado en su definición tiene 4 valores, contando la etiqueta no evento escolar, y una vez codificados quedan de la siguiente forma:

	Eescolar_1	Eescolar_2	Eescolar_3	Eescolar_4
No escolar	1	0	0	0
Vacaciones verano	0	1	0	0
Vacaciones navidad	0	0	1	0
Vacaciones semana santa	0	0	0	1

Transformando la característica evento escolar en 4 características.

3.4.3.7. Evento normal

Al igual que en los casos anteriores, este atributo se agrupa como atributo cualitativo nominal, por lo que se ha usado el codificador one-hot para codificarlo.

Por otro lado, tal y como se ha comentado en la definición del atributo, este contiene 7 valores, contando la etiqueta no evento normal. Quedando de la siguiente forma una vez codificado:

	EventoNormal_1	EventoNormal_2	EventoNormal_3	EventoNormal_4	EventoNormal_5	EventoNormal_6	EventoNormal_7
Carnaval	1	0	0	0	0	0	0
FM Lérida	0	1	0	0	0	0	0
Moros y C.	0	0	1	0	0	0	0
Aplec Caracol	0	0	0	1	0	0	0
Fiestas Otoño	0	0	0	0	1	0	0
San José	0	0	0	0	0	1	0
No evento normal	0	0	0	0	0	0	1

Transformando la característica evento normal en 7 características.

3.4.3.8. Descripción tiempo

Aunque este atributo pertenece al mismo grupo que los anteriores, pero como hemos podido comprobar, el número de características se ha ido engrandando bastante, por lo que en este caso se ha utilizado el codificador binario para reducir la dimensionalidad de la característica.

Tal y como se ha definido en su sección, dicho atributo tiene 11 valores, y una vez codificados han quedado de la siguiente forma:

	Tiempo_1	Tiempo_2	Tiempo_3	Tiempo_4	Tiempo_5	Tiempo_6
Clear	0	0	0	0	0	1
Cloudy	0	0	0	0	1	0
Fog	0	0	1	0	1	1
Light rain	0	0	0	1	0	0
Light rain show	0	0	0	1	0	1
Mist	0	0	0	1	1	0
Moderate	0	0	0	1	1	1
Overcast	0	0	1	0	0	0
Partly cloudy	0	0	1	0	0	1
Patchy rain posible	0	0	1	0	1	0
Thundery outbreak	0	0	1	0	1	1

Transformando la característica descripción tiempo en 6 características.

3.4.3.9. Semana del año

De igual forma que el atributo anterior, este se engloba en el grupo de los cualitativos y nominales, pero debido a su gran cantidad de valores se ha decidido codificar mediante el codificador binario, para no agrandar más la dimensionalidad del problema.

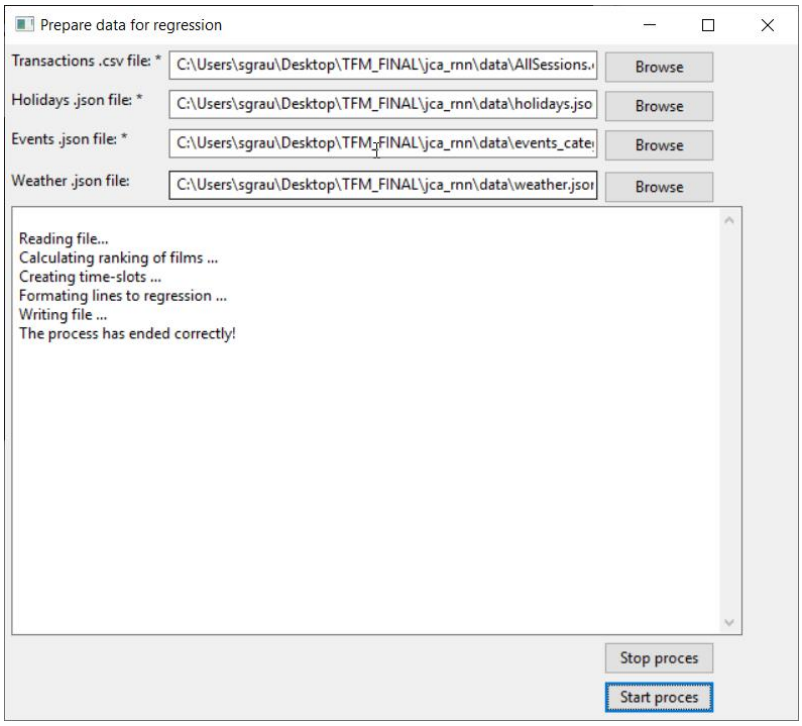
Tal y como se ha definido anteriormente, dicho atributo tiene 52 valores, y una vez codificados han quedado de la siguiente forma:

	Tiempo_1	Tiempo_2	Tiempo_3	Tiempo_4	Tiempo_5	Tiempo_6	Tiempo_7
1	0	0	0	0	0	0	1
2	0	0	0	0	0	1	0
3	0	0	0	1	0	1	1
4	0	0	0	0	1	0	0
5	0	0	0	0	1	0	1
6	0	0	0	0	1	1	0
7	0	0	0	0	1	1	1
8	0	0	0	1	0	0	0
9	0	0	0	1	0	0	1
10	0	0	0	1	0	1	0
11	0	0	0	1	0	1	1
12	0	0	0	1	1	0	0
13	0	0	0	1	1	0	1
14	0	0	0	1	1	1	0
15	0	0	0	1	1	1	1
16	0	0	1	0	0	0	0
17	0	0	1	0	0	0	1
...	...	...	...	...	...	...	...
52	0	1	1	0	1	0	0

Transformando la característica semana del año en 7 características.

3.5. Procesamiento de los datos

Para lograr un correcto procesamiento de los datos y hacer que este no sea tedioso, se ha implementado un programa en Python de entorno de escritorio, con la función de realizar la creación de todos los atributos comentados en las secciones anteriores.
Es necesario recalcar que previamente a la ejecución de dicho programa, es necesario ejecutar el script *group.py* tal y como se ha comentado en la sección de *Agrupación de los datos*, ya que los datos no nos vienen agrupados.
Una vez ejecutado dicho script, es el momento de lanzar la aplicación de procesamiento la cual se llama *interface*, obteniendo el siguiente formulario de entrada de ficheros:



- ***Transactions .csv file**:* Este campo es obligatorio y hace referencia al fichero en format .csv que contiene todas las transacciones del cine agrupadas por sesión.

- ***Holidays*** .json file: Este campo es obligatorio y hace referencia al fichero en formato .json que contiene todos los festivos nacionales con sus fechas mapeadas.

- ***Events .json file***: Este campo es obligatorio y hace referencia al fichero en formato .json que contiene todos los eventos normales, eventos escolares y días de la fiesta del cine con sus fechas mapeadas.

- ***Weather .json file***: Este campo no es obligatorio, por lo que si no se introduce no se calcularan los atributos: *Descripción del tiempo, precipitaciones y temperatura*. Hace referencia a un fichero en formato .json que contiene toda la información necesaria del tiempo mapeada en sus correspondientes fechas.

Una vez realizada la ejecución del programa, este nos devolverá un fichero llamado *AllSessions_parsed.csv* el cual contendrá nuestro dataset con todos los atributos preparados para ser normalizados.

A continuación, es necesario ejecutar el script *normalize.py*, el cual es el encargado de normalizar todos los atributos siguiendo las técnicas comentadas en su debida sección, devolviendo como resultado un fichero llamado *AllSessions_normalized.csv*, preparado para empezar a entrenar nuestra red.

Se ha decidido dividir los procesos en dos, ya que de esta forma mediante el fichero *AllSessions_parsed.csv* es más fácil poder realizar comprobaciones en los datos que si se devuelven normalizados desde un buen principio.

Quedando de esta forma, en el caso de la división de los datos en franjas de una hora, un total de 26447 líneas de una dimensión igual a 77 o 78 si contamos el atributo *Venta de Entradas* (Target).

4. Implementación del Perceptrón multicapa

Tal y como se ha comentado en la introducción a dicho modelo, para abordar el problema se utilizará el Perceptrón multicapa.

Para la implementación de este, se ha hecho uso de las librerías que nos facilita *Keras*, junto con *sklearn.*

A continuación, se hará una descripción de todos los pasos que se han ido siguiendo juntamente con sus pruebas, para lograr llegar a minimizar el error lo máximo posible.

Pero hablando del error, es necesario previamente comentar que tipos de funciones de pérdida tenemos en la regresión y cuál de ellas se ha escogido.

4.1. Función de pérdida

En el ámbito de la regresión, existen varios tipos de funciones de pérdida, pero en este caso únicamente se definirán los 3 siguientes:

- Pérdida del error cuadrático medio: Dónde el error cuadrático medio se calcula como el promedio de las diferencias cuadráticas entre los valores predichos y reales. El resultado siempre es positivo, ya que se le aplica el valor absoluto. Con esta función se intenta que los errores más grandes sean mayormente penalizados que los errores de menor rango.

- Pérdida del error absoluto medio: En dicha función se calcula el promedio de la diferencia absoluta entre los valores reales y los predichos.

- Pérdida del error logarítmico cuadrático medio: En este caso primero se calcula el logaritmo natural de cada uno de los valores pronosticados y posteriormente se calcula el error cuadrático medio. Con dicha función logramos no penalizar con tanta fuerza como se hace en la función del error cuadrático medio.

Una vez entendidos los tipos anteriores, se ha escogido como función de pérdida, la del error cuadrático medio, ya que en nuestras predicciones nos interesa no cometer errores de rango muy elevado, por lo que es necesario penalizar estos errores para disminuir dichos rangos.

(Nota: Aunque se minimice el error cuadrático medio, en todas las gráficas que se presentarán a continuación se representa el error absoluto medio, ya que los valores a interpretar son mucho más entendibles.)

4.2. Punto de partida

Como punto de partida, se ha escogido implementar un MLP muy básico con la siguiente topología:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 512)               39936
_________________________________________________________________
dense_2 (Dense)              (None, 512)               262656
_________________________________________________________________
activation_1 (Activation)    (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               262656
_________________________________________________________________
activation_2 (Activation)    (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 512)               262656
_________________________________________________________________
activation_3 (Activation)    (None, 512)               0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 513
=================================================================
Total params: 828,417
Trainable params: 828,417
Non-trainable params: 0
```

Como se puede comprobar, se parte de una red constituida por 5 capas:

- 1 capa de entrada de 512 neuronas.
- 3 capas ocultas con una función de activación RELU y 512 neuronas.
- 1 capa de salida con una única neurona y con una función de activación lineal, ya que se trata de una regresión lineal.

Como optimizador se ha utilizado el *Adam*, ya que se han realizado varias pruebas con varios otros, pero las mejores sensaciones se han obtenido con este.

Y por último también se le ha añadido un factor de learning rate, tal que así:

```
if  125 < epoch:
    return 0.01
if  250 < epoch:
    return 0.001
if  375 < epoch:
    return 0.0001
else:
    return .1
```

4.3. Asignar dimensión

En una de las reuniones realizadas con el cliente, este nos hizo saber qué en el factor del cine, la historia pasada es muy importante en cuanto a la predicción de entradas vendidas para un día. Pero también afirmó que no estaba seguro cuanto tiempo atrás se tiene que tener en cuenta para la predicción de un día. Por lo tanto, se han realizado varias pruebas, para comprobar cuantos días atrás se tienen que tener en cuenta para una predicción. Para ello, inicialmente se ha ordenado el dataset por fecha de menor a mayor y seguidamente se ha concatenado a cada fila referente a una predicción, las filas correspondientes a 1, 2 o 3 semanas.

Por ejemplo, si disponemos de 4 filas que hacen referencia a un día cada una, ordenadas cronológicamente, siendo la Fila1 la primera en ocurrir y la 4 la última:

- Fila1: [0,2,1,5]
- Fila2: [9,4,7,4]
- Fila3: [2,6,2,1]
- Fila4: [7,8,4,6]

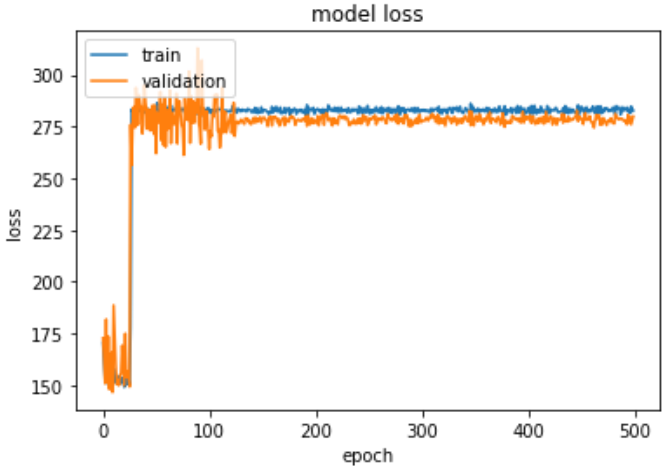
Y queremos tener en cuenta la historia pasada de 3 días, el resultado de la fila 4 sería el siguiente:

- Filaa4: [0,2,1,5,9,4,7,4,2,6,2,1,7,8,4,6]

(Es necesario añadir que las pruebas se han iniciado con una partición horaria de franjas de 4 horas, con 500 epochs, con u 80% del dataset para entrenamiento, un 10% para validación y el restante para test)

4.3.1. Diaria

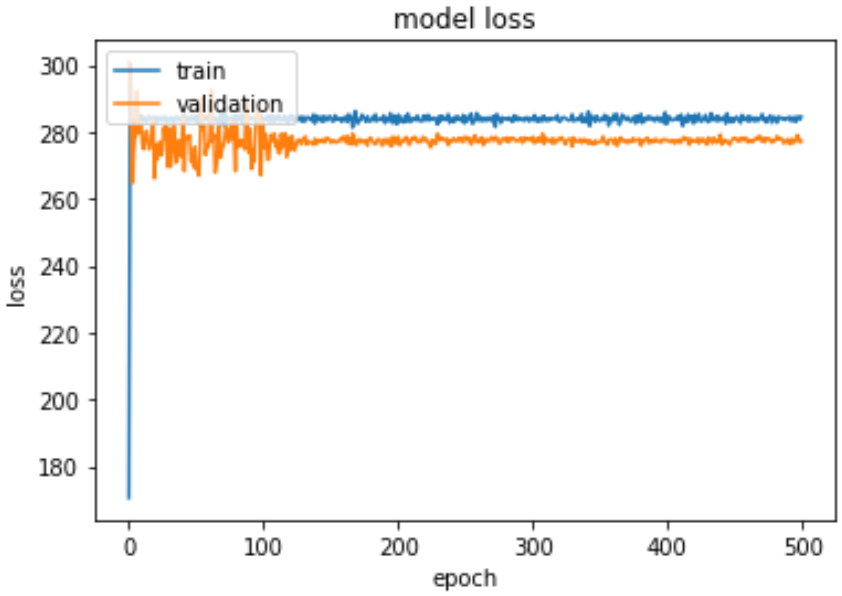
Tras entrenar la red neuronal se ha obtenido la siguiente gráfica:



Obteniendo en test un error medio absoluto igual a 279.9631.

4.3.2. Una semana

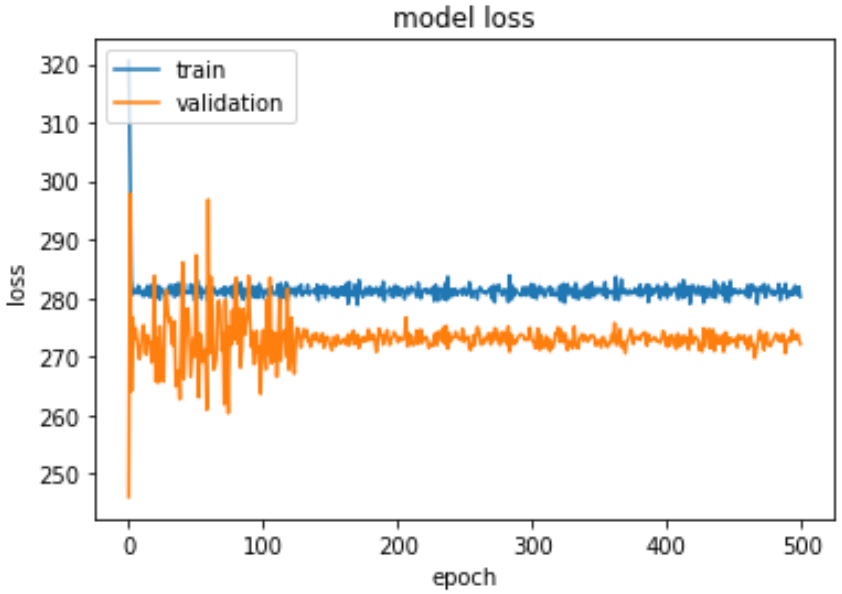
Tras entrenar la red neuronal se ha obtenido la siguiente gráfica:



Obteniendo en test un error medio absoluto igual a 277.2937.

4.3.3. Dos semanas

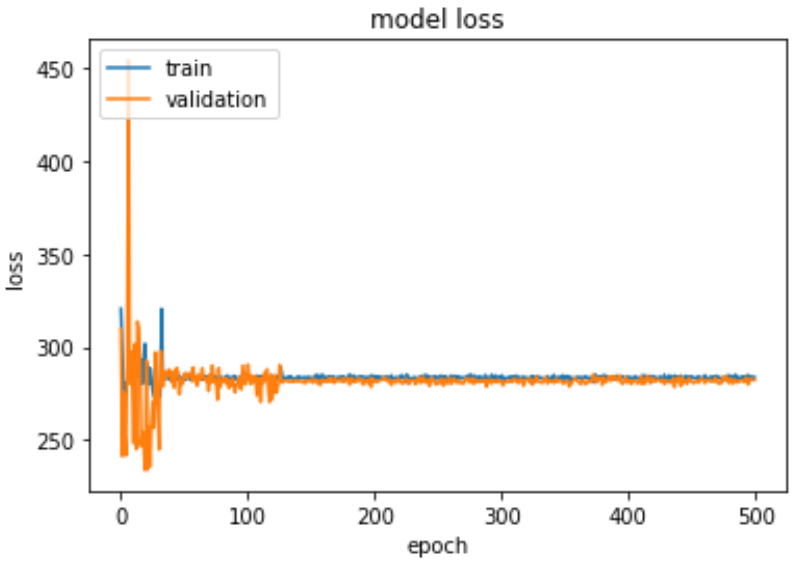
Tras entrenar la red neuronal se ha obtenido la siguiente gráfica:



Obteniendo en test un error medio absoluto igual a 272.1307.

4.3.4. Tres semanas

Tras entrenar la red neuronal se ha obtenido la siguiente gráfica:



Obteniendo en test un error medio absoluto igual a 281.1057.

4.3.5. Resultados

Una vez realizadas las anteriores pruebas, obtenemos la siguiente tabla de resultados:

Semanas	Pérdida en test
0	279.9631
1	277.2937
2	**272.1307**
3	282.1057

Por lo tanto, se ha decidido por tener en cuenta dos semanas de historia pasada. Esto aumenta significativamente la dimensión del problema, pero vamos a esperar al próximo apartado para obtener la dimensión final.

4.4. Asignar time-slot

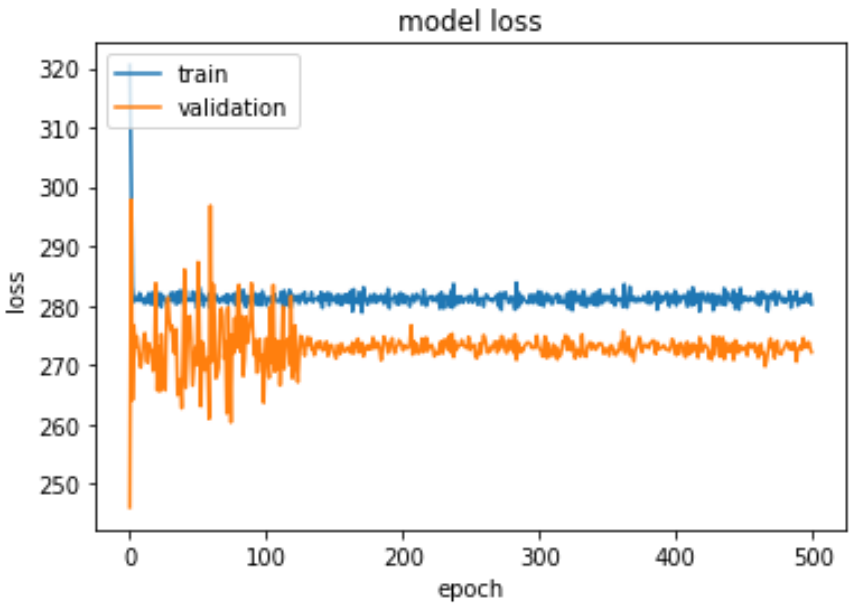
De igual forma que con el apartado anterior, el cliente nos comentó que le interesaría predecir la venta de entradas por franjas horarias, pero que no sabía exactamente como dividirlas ya que los datos quitan un poco de flexibilidad a dicha división, por lo que se ha decidido realizar varias pruebas para comprobar cuál es la partición correcta.

4.4.1. 4 horas

Por lo tanto, dispondremos de las siguientes franjas horarias:

- *15:00h -18:00h*
- *19:00h - 22:00h*
- *23:00h - 00:00h*

Tras entrenar la red neuronal se ha obtenido la siguiente gráfica:



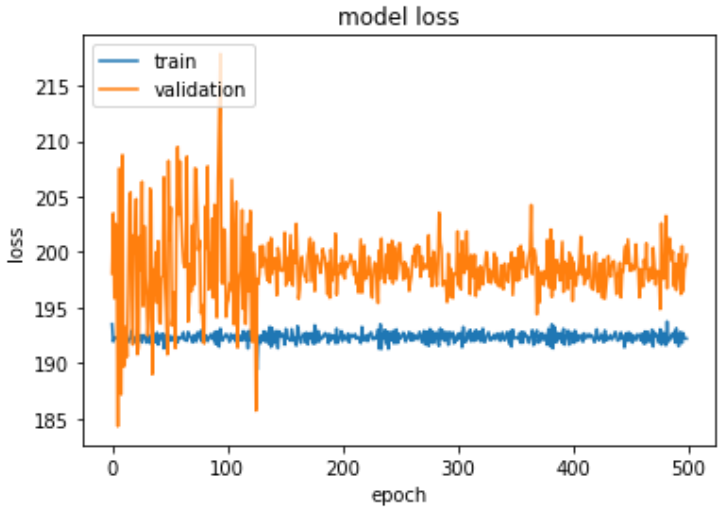
Obteniendo en test un error medio absoluto igual a 272.1307.

4.4.2. 2 horas

Por lo tanto, dispondremos de las siguientes franjas horarias:

- *15:00h – 16:00h*
- *17:00H – 18:00h*
- *19:00h – 20:00h*
- *21:00h – 22:00h*
- *23:00h – 00:00h*

Tras entrenar la red neuronal se ha obtenido la siguiente gráfica:



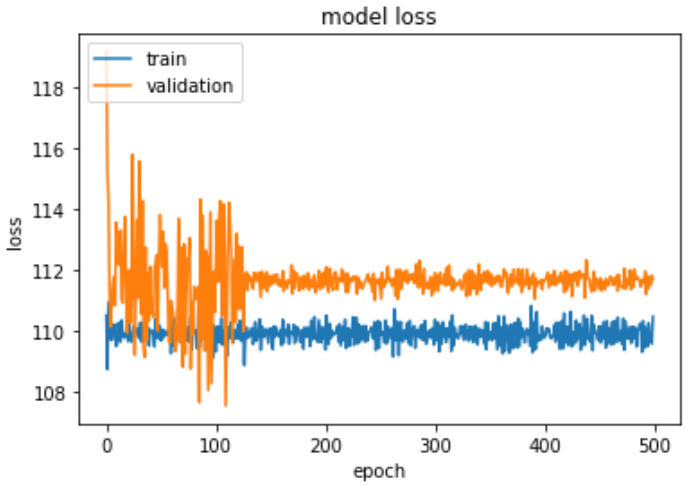
Obteniendo en test un error medio absoluto igual a 199.7475.

4.4.3. 1 hora

En este caso dispondremos de un time-slot para cada hora:

- *15:00h - 16:00h*
- *17:00h - 18:00h*
- *19:00h - 20:00h*
- *21:00h - 22:00h*
- *23:00h - 00:00h*

Tras entrenar la red neuronal se ha obtenido la siguiente gráfica:



Obteniendo en test un error medio absoluto igual a 111.7032.

4.4.4. Resultados

Una vez realizadas las anteriores pruebas, obtenemos la siguiente tabla de resultados:

Franja horaria	Pérdida en test
4 horas	272.1307
2 horas	199.7475
1 hora	**111.7032**

Por lo tanto, se ha decidido por dividir las franjas horarias en horas, obteniendo de esta forma finalmente un máximo de 10 time-slots.

Entonces, ahora ya podemos calcular la dimensionalidad de nuestro problema, ya que si sabemos que de lunes a jueves el cine únicamente abre de 15:00h a 22:00h (8 time-slots) y de viernes a domingo abre de 15:00h a 00:00h (10 time-slots), sabiendo que cada fila hace referencia a un time-slot diferente, tenemos que:

- 1 semana es igual a 62 time-slots, por lo tanto, dos semanas son iguales a 124 time-slots.

- Tal y como hemos comentado anteriormente cada fila tiene una dimensionalidad inicial de 77 características, por lo tanto, si concatenamos cada fila con sus dos semanas de historia pasada, obtenemos que cada entrada será de una dimensión igual a 77 * 124 = 9.548.

- Y al agrupar el problema en horas, el conjunto de datos acaba teniendo un total de 26447 líneas.

4.5. Partición conjunto de datos

Por lo tanto, si partimos de un conjunto de datos de 26447 líneas y de una dimensión de 9548 (Para realizar las particiones se ha utilizado la función *train_test_split* de *sklearn)*:

- **Conjunto de test**: 10% del total, 2644 muestras de dimensión 9548, con sus respectivas etiquetas en otro vector de 2644 muestras de dimensión 1.

- **Conjunto de entrenamiento**: 90% del total – test, 21422 muestras de dimensión 9548, con sus respectivas etiquetas en otro vector de 21422 muestras de dimensión 1.

- **Conjunto de entrenamiento**: 10% del total – test, 2380 muestras de dimensión 9548, con sus respectivas etiquetas en otro vector de 2380 muestras de dimensión 1.

4.6. Regularización

El sobre aprendizaje es un fenómeno que ocurre cuando un modelo de aprendizaje automático o estadístico se adapta a un conjunto de datos en particular y no puede generalizarse a otros conjuntos de datos, es decir se adapta al conjunto de entrenamiento, pero no se generaliza al conjunto de test.

Entonces entra en juego la regularización, la cual consiste en un proceso de introducción de información adicional para evitar el sobre aprendizaje, penalizando la función de pérdida, que en nuestro caso es la del error cuadrático.

Para ello tenemos 2 tipos de regularizaciones:

- Regularización l1 o lasso regression: Según nuestra función de pérdida, al agregarle el término de regularización l1 se vería de la siguiente forma:

$$L_1 = (wx + b - y)^2 + \lambda|w|$$

  Donde el parámetro de regularización $\lambda > 0$ se ajusta manualmente. Teniendo en cuenta que |w| es diferenciable en todas partes excepto cuando w = 0, como se muestra a continuación:

$$\frac{d|w|}{dw} = \begin{cases} 1 & w > 0 \\ -1 & w < 0 \end{cases}$$

  Dicha regularización es utilizada para penalizar la función de pérdida referente al error medio absoluto.

- Regularización l2 o ridge regresión: Es similar a la anterior, pero en este caso se encuentra elevada al cuadrado tal y como se muestra a continuación:

$$L_2 = (wx + b - y)^2 + \lambda w^2$$

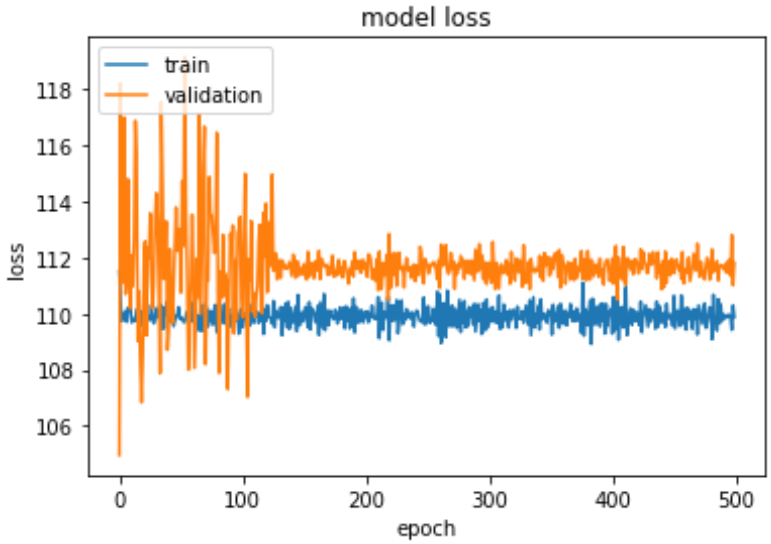
  Donde de igual forma que anteriormente el parámetro de regularización $\lambda > 0$ se ajusta manualmente.

  Dicha regularización es utilizada para penalizar la función de pérdida referente al error cuadrático medio.

Una vez introducido el concepto de la regularización y sus tipos, se han realizado dos pruebas para la elección correcta de regularización.

4.6.1. L1 (Lasso regression)

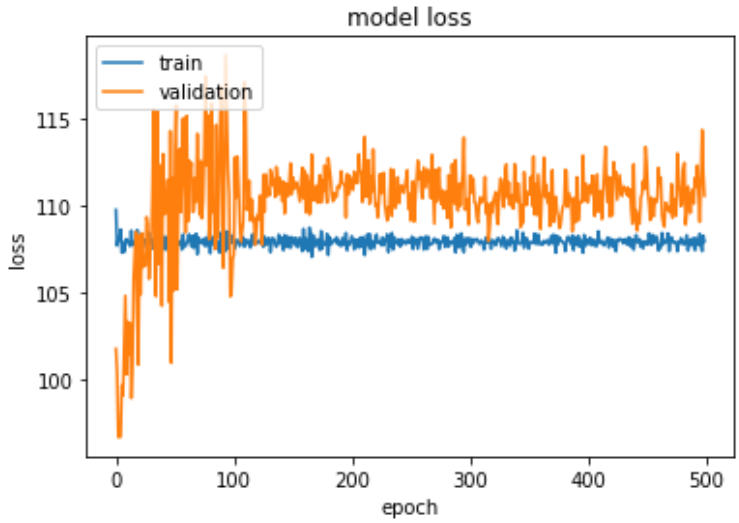
Aplicando como parámetro $\lambda = 0.0001$, se ha aplicado la regularización en las capas ocultas y se ha obtenido el siguiente gráfico:



Obteniendo en test un error medio absoluto igual a 111.6243.

4.6.2. L2 (Ridge regression)

Aplicando como parámetro $\lambda = 0.0001$, se ha aplicado la regularización en las capas ocultas y se ha obtenido el siguiente gráfico:



Obteniendo en test un error medio absoluto igual a 109.8823.

4.6.3. Resultados

Una vez realizadas las anteriores pruebas, obtenemos la siguiente tabla de resultados:

Regularización	Pérdida en test
Sin regularización	111.7032
L1	111.6243
L2	**109.8823**

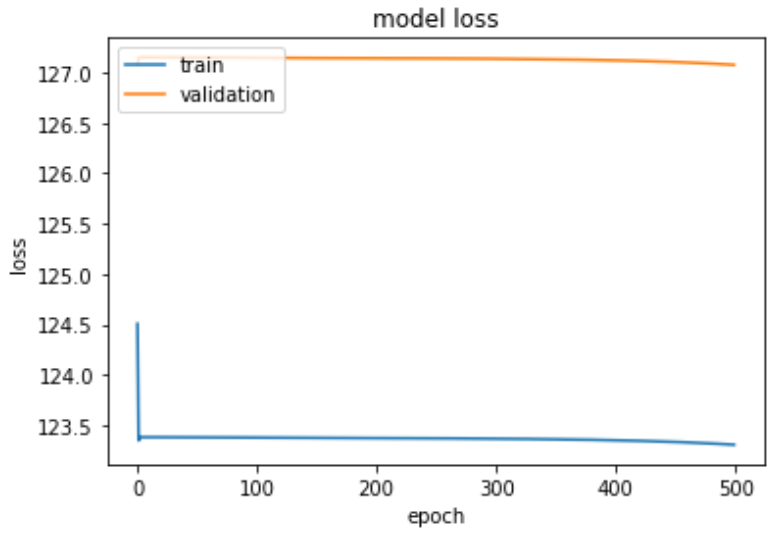
Por lo tanto, aunque la mejora no haya sido muy grande, la regularización nos proporciona una pérdida menor de la que habíamos obtenido hasta el momento, por lo que se añade esta nueva configuración al modelo actual.

4.7. Topología

Actualmente tenemos una red compuesta por 1 capa de entrada (512 neuronas), 3 capas ocultas (512 neuronas) y una capa de salida (1 neurona), pero no estamos seguros de que esta sea la tipología que mejor se adapte al problema, por lo que a continuación se realizarán un par de pruebas con diferentes topologías.

4.7.1. Pirámide

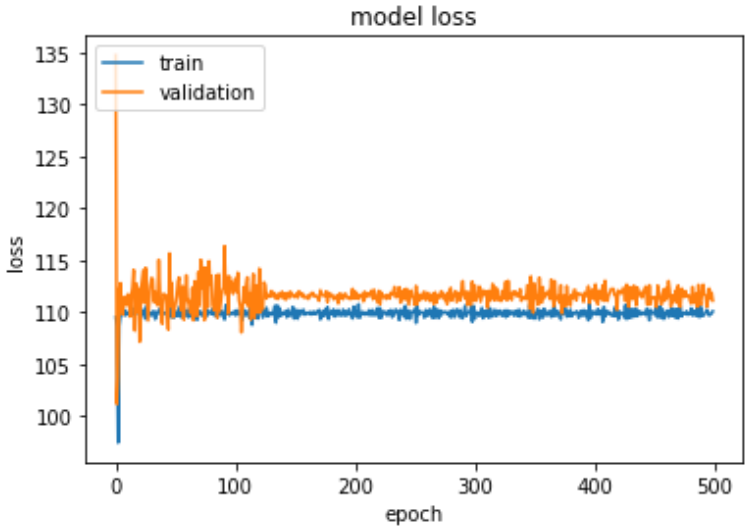
Para esta prueba se ha utilizado una capa de entrada (1024 neuronas), 3 capas ocultas (512, 256 y 64 neuronas respectivamente) y una capa de salida (1 neurona), obteniendo el siguiente gráfico final:



Obteniendo en test un error medio absoluto de 127.0737.

4.7.2. Totalmente conectada

Para esta prueba se ha utilizado una capa de entrada (1024 neuronas), 3 capas ocultas (1024 neuronas) y una capa de salida (1 neurona), obteniendo el siguiente gráfico final:



4.7.3. Resultados

Una vez realizadas las anteriores pruebas, obtenemos la siguiente tabla de resultados:

Topologia	Pérdida en test
512-512-512-512-1	**109.8823**
1024-512-256-61-1	127.0737
1024-1024-1024-1024-1	111.1163

Como podemos comprobar ninguna de las pruebas realizadas, mejora el resultado obtenido hasta el momento, por lo que, aunque probablemente si realizáramos un conjunto más elevado de pruebas encontraríamos alguna topología que se adaptará mejor, nos quedaremos con la utilizada hasta el momento.

4.8. Normalización por lotes (Batch Normalization)

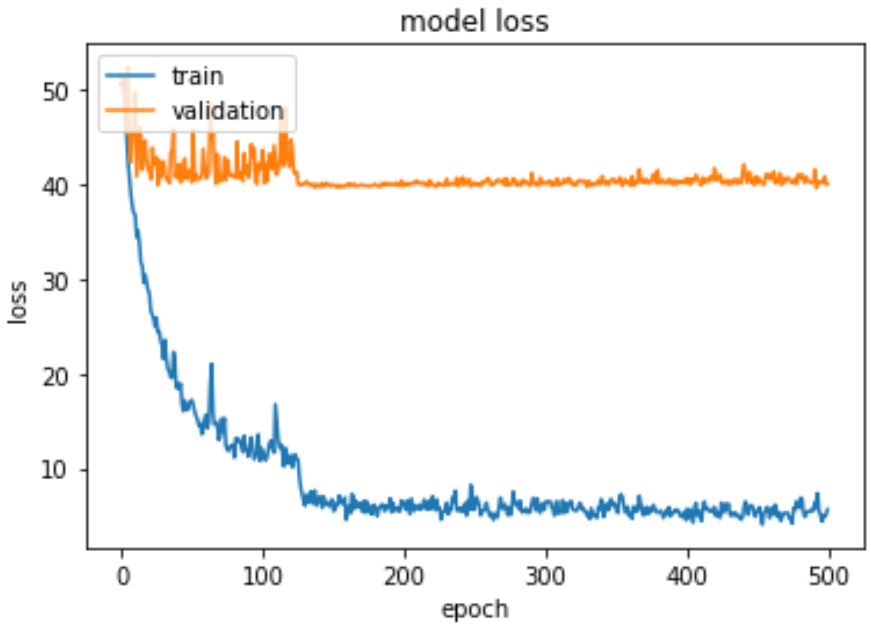
La normalización por lotes tiene como finalidad reducir la covarianza de los valores de cada capa oculta. Es decir, actúa como la normalización que hemos realizado previamente a los datos, pero interiormente en las capas ocultas, por lo que si esto nos ha funcionado para los valores de entrada, aparentemente nos tendría que ayudar a reducir bastante nuestra pérdida.

Gracias a la normalización por lotes,

- Podemos usar tasas de aprendizaje más altas porque asegura que no existe activaciones realmente altas o bajas, agilizando bastante el proceso de entrenamiento.

- Reduce el sobre aprendizaje, ya que también tiene un ligero efecto de regularización, agregando algo de ruido a las activaciones de cada capa oculta.

Entonces se ha aplicado una normalización por lotes a cada una de las capas ocultas justo antes de la función de activación y se ha obtenido el siguiente gráfico:



Obteniendo en test un error absoluto medio de 39.9786.

4.8.1. Resultados

Una vez realizada la anterior prueba, obtenemos la siguiente tabla de resultados:

Normalización por lotes	Pérdida en test
No	109.8823
Sí	**39.9786**

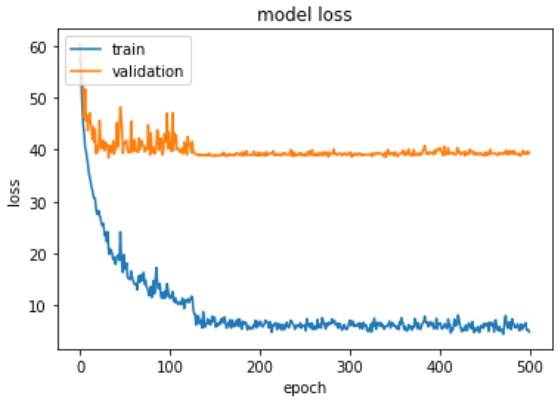
Por lo que podemos comprobar, gracias a la normalización por lotes obtenemos una notable mejora en nuestra pérdida, por lo que se adopta esta nueva configuración para nuestra red.

4.9. Ruido gaussiano (Gaussian Noise)

El ruido gaussiano, también tiene como finalidad reducir el sobre aprendizaje y de esta forma lograr que nuestro modelo consiga generalizar en mayor cantidad. El hecho de añadir ruido durante un entrenamiento en una red neuronal tiene un efecto regularizador, ya que tiene un impacto similar en la función de pérdida como la adición de un término de penalización, y a su vez aumenta la robustez del modelo.
El agregar ruido expande el tamaño del conjunto de datos de entrenamiento, ya que cada vez que una muestra de entrenamiento se expone al modelo, se agrega ruido aleatorio a las variables de entrada haciéndolas diferentes cada vez que se exponen al modelo.
Por lo tanto, la red es menos capaz de memorizar muestras de entrenamiento porque estas están cambiando todo el tiempo, lo que resulta en pesos de red más pequeños y una red más robusta que tiene un menor error de generalización.

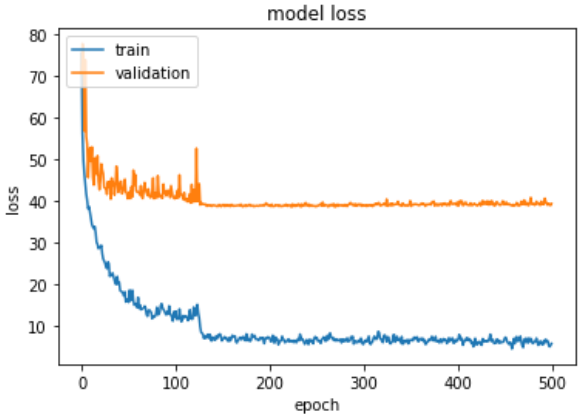
4.9.1. Capas ocultas

- **Aplicar 0.2 de ruido**: Se ha aplicado un 0.2 de ruido gaussiano en cada una de las capas ocultas justo antes de su activación y se ha sacado la siguiente gráfica:



Obteniendo en test un error absoluto medio de 39.4212.

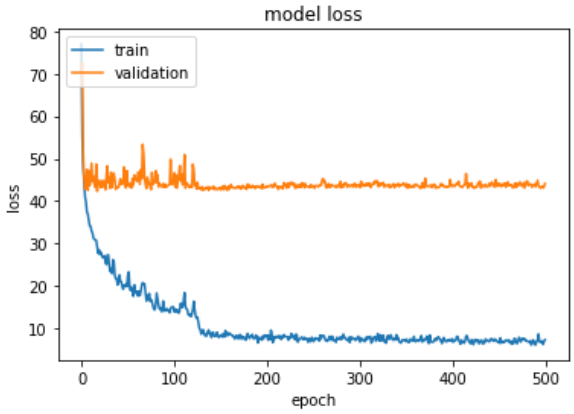
- Aplicar 0.4 de ruido: Se ha aplicado un 0.4 de ruido gaussiano en cada una de las capas ocultas justo antes de su activación y se ha sacado la siguiente gráfica:



Obteniendo en test un error absoluto medio de 38.3178.

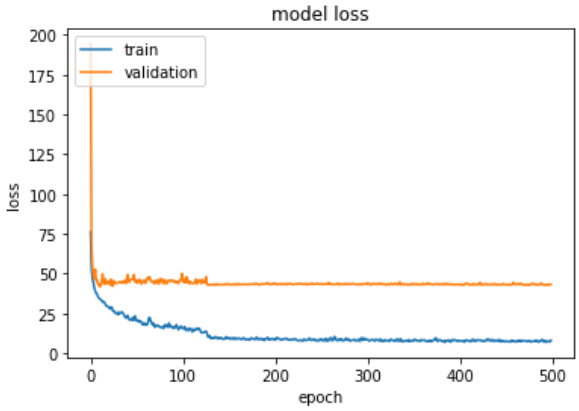
4.9.2. Capa de entrada y capas ocultas

- Aplicar 0.01 y 0.2 de ruido: Se ha aplicado un 0.01 de ruido gaussiano en la capa de entrada y 0.2 de ruido gaussiano en cada una de las capas ocultas justo antes de su activación. Obteniendo el siguiente gráfico:



Obteniendo en test un error absoluto medio de 44.1447.

- Aplicar 0.01 y 0.4 de ruido: Se ha aplicado un 0.01 de ruido gaussiano en la capa de entrada y 0.4 de ruido gaussiano en cada una de las capas ocultas justo antes de su activación. Obteniendo el siguiente gráfico:



Obteniendo en test un error medio absoluto de 43.2065.

4.9.2.1. Learning Rate

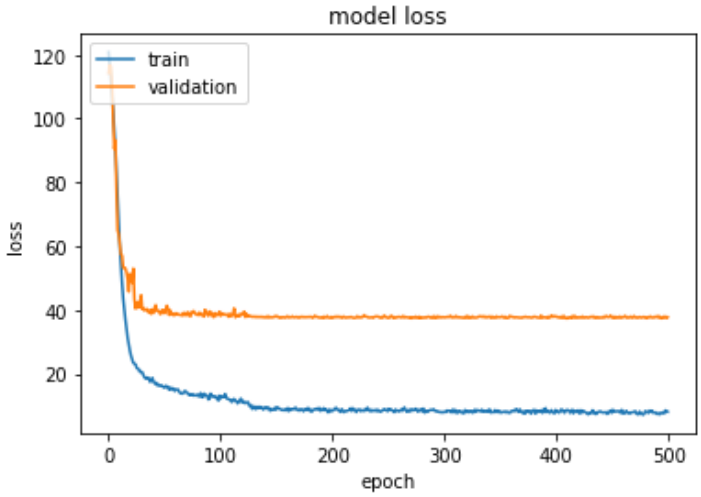
La tasa de aprendizaje o learning rate es un parámetro que controla cuánto estamos ajustando los pesos de nuestra red con respecto al gradiente de pérdida. Cuanto más bajo es el valor, más lento se pasa a lo largo de la pendiente descendente.
Además, la tasa de aprendizaje afecta a la rapidez con la que nuestro modelo puede converger a un mínimo local.

Por lo tanto, podemos decir que a cuanto más grande es el valor más saltos dará nuestra pérdida y a cuanto más pequeño, esta será mas estable.

Dado que, con la tasa de aprendizaje utilizada hasta el momento, al añadirle ruido gaussiano en la capa de entrada, nuestra red ha empeorado, vamos a realizar un par de pruebas disminuyendo la tasa de aprendizaje, como se muestra a continuación:

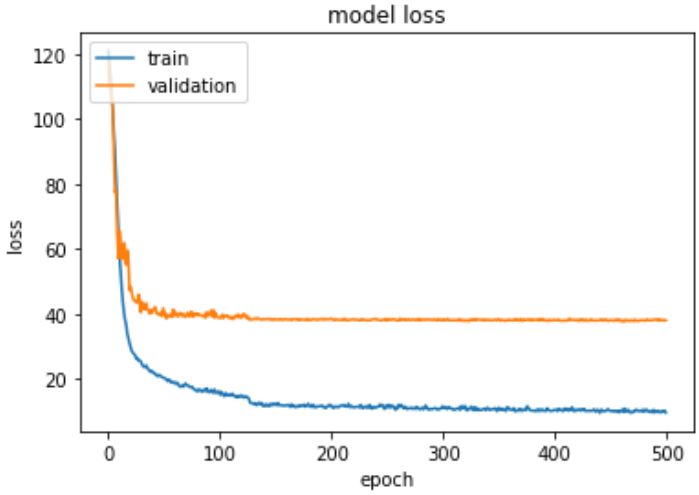
```
def scheduler(epoch):
    if  125 < epoch:
        return 0.0001
    if  250 < epoch:
        return 0.00001
    if  375 < epoch:
        return 0.000001
    else:
        return .001
```

- Aplicar 0.01 y 0.2 de ruido: Se ha aplicado un 0.01 de ruido gaussiano en la capa de entrada y 0.2 de ruido gaussiano en cada una de las capas ocultas justo antes de su activación. Obteniendo el siguiente gráfico:



Obteniendo en test un error absoluto medio de 37.7394.

- Aplicar 0.01 y 0.4 de ruido: Se ha aplicado un 0.01 de ruido gaussiano en la capa de entrada y 0.4 de ruido gaussiano en cada una de las capas ocultas justo antes de su activación. Obteniendo el siguiente gráfico:



Obteniendo en test un error absoluto medio de 37.0577.

4.9.3. Resultados

Una vez realizadas las anteriores pruebas, obtenemos la siguiente tabla de resultados:

Capa Entrada	Capa Oculta	Learning rate	Pérdida en test
0.0	0.0	0.1 – 0.01-0.001-0.0001	39.9786
0.0	0.2	0.1 – 0.01-0.001-0.0001	39.4212
0.0	0.4	0.1 – 0.01-0.001-0.0001	38.3178
0.01	0.2	0.1 – 0.01-0.001-0.0001	44.1447
0.01	0.4	0.1 – 0.01-0.001-0.0001	43.2065
0.01	0.2	0.001 – 0.0001-0.00001-0.000001	37.7394
0.01	**0.4**	**0.001 – 0.0001-0.00001-0.000001**	**37.0577**

Por lo que podemos comprobar que, aunque no mucho, gracias al ruido aplicado y la disminución de la tasa de aprendizaje, se ha disminuido la pérdida en test.

4.10. Modelo final

Una vez realizadas todas las pruebas y parametrizaciones anteriores, hemos llegado a un modelo que nos ha dado el mejor resultado obtenido en test de *37.0577*, por lo qué, este será el modelo escogido para empezar a funcionar en el entorno del cliente.

Dicho modelo final, ha quedado de la siguiente forma:

Función de pérdida	*Error cuadrático medio*
Optimizador	*Adam*
Dimensión entrada	*9548*
Regularización	*L2:* λ = 0.0001
Topología	*512 – 512 – 512 – 512 - 1*
Normalización por lotes (Batch Norm.)	*Sí*
Ruido Gaussiano (Gaussian Noise)	*0.01 (Capa entrada) – 0.4 (Capas ocultas)*
Learning rate	*0.001 – 0.0001-0.00001-0.000001*

5. Conclusiones

El hecho de realizar predicciones sobre pasajeros, asistentes, compradores, etc.., de los cuales se tienen datos históricos, puede parecer aparentemente una tarea bastante sencilla en la cual el verdadero peso recae sobre la implementación de la red neuronal.

Realmente yo ya sabía que esto no era así, pero tampoco me llegaba a imaginar que prácticamente todo el trabajo se encontrara en el tratamiento de datos y más cuando estos no llegan de la manera más idónea.

A su vez, aunque no he presentado pruebas en el trabajo de entrenamientos con los datos sin normalizar y codificar, he aprendido el real impacto de aprendizaje que proporciona dicha normalización y codificación en la red, ya que los resultados sin tratar los datos son abrumadoramente peores.

En cuanto a la implementación de la red neuronal, me ha quedado más que demostrado que la parametrización depende muchísimo de los datos utilizados, ya que hasta el momento por ejemplo el Dropout siempre me había funcionado para mejorar la precisión de la red en mis pasados proyectos, pero en este caso la empeoraba y mucho, motivo por el cual no he puesto ningún apartado de Dropout en el trabajo.

Por último, añadir que inicialmente yo quería abordar el problema con un LSTM (Long Short Term Memory) ya que la predicción viene muy fuertemente ligada con la historia pasada tal y como se ha comentado. Pero mi tutor me comentó que no lo hiciera así, que probara hacerlo con un multilayer perceptrón.

Inicialmente yo estaba muy escéptico a que dicho modelo logrará facilitarme unos buenos resultados para mi problema, porque como ya he dicho yo lo encasillaba a un problema de series temporales, pero por muy asombro mío dicho modelo ha respondido gratamente al problema y ha logrado proporcionarme unos muy buenos resultados.

6. Bibliografía

- D, F. D. F. (2017, 20 octubre). Batch normalization in Neural Networks [Publicación en un blog]. Recuperado 5 septiembre, 2019, de https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c

- Urvashi, U. J. Jaitley. (2019, 9 abril). Why Data Normalization is necessary for Machine Learning models [Publicación en un blog]. Recuperado 3 septiembre, 2019, de https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029

- Brownie, J. B. Jason. (2019, 16 agosto). SmarterHow to Choose Loss Functions When Training Deep Learning Neural Networks Ways to Encode Categorical Data for Machine Learning [Publicación en un blog]. Recuperado 4 septiembre, 2019, de https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

- Brownlee, J. B. Jason. (2019, 6 agosto). Train Neural Networks With Noise to Reduce Overfitting [Publicación en un blog]. Recuperado 6 septiembre, 2019, de https://machinelearningmastery.com/train-neural-networks-with-noise-to-reduce-overfitting/

- Hale, J. H. Jeff. (2018, 11 septiembre). Smarter Ways to Encode Categorical Data for Machine Learning [Publicación en un blog]. Recuperado 3 septiembre, 2019, de https://towardsdatascience.com/smarter-ways-to-encode-categorical-data-for-machine-learning-part-1-of-3-6dca2f71b159

- DeMuro, J. M. Jonas. (2018, 11 agosto). What is a neural network? [Publicación en un blog]. Recuperado 3 septiembre, 2019, de https://www.techradar.com/news/what-is-a-neural-network

- Jordi, J. O. S. Ollé Sanchez. (2019, 7 marzo). Qué es y cómo interpretar una regresión logística [Publicación en un blog]. Recuperado 2 septiembre, 2019, de https://conceptosclaros.com/que-es-regresion-logistica/

- Karim, R. K. Raimi. (2018, 26 diciembre). Intuitions on L1 and L2 Regularisation [Publicación en un blog]. Recuperado 4 septiembre, 2019, de https://towardsdatascience.com/intuitions-on-l1-and-l2-regularisation-235f2db4c261

- Gandhi, R. G. Rohith. (2018, 7 junio). Support Vector Machine — Introduction to Machine Learning Algorithms [Publicación en un blog]. Recuperado 2 septiembre, 2019, de https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

- Ray, S. R. Sunil. (2015, 10 agosto). 7 Regression Techniques you should know! [Publicación en un blog]. Recuperado 2 septiembre, 2019, de https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/60

- Mohit, M. Y. Yadav. (2019, 23 junio). Understanding Data Attribute Types | Qualitative and Quantitative [Publicación en un blog]. Recuperado 3 septiembre, 2019, de https://www.geeksforgeeks.org/understanding-data-attribute-types-qualitative-and-quantitative/