UNIVERSIDAD DE GRANADA

ETSIIT
Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Práctica 3: Ajuste de Modelos Lineales

Aprendizaje Automático

Mayo-Junio, 2022

Autor:
Lugli, Valentino Glauco · YB0819879

Índice

0. Notas sobre la implementación

- La práctica fue realizada completamente en Python haciendo uso del IDE "Spyder", este IDE permite tener "celdas" de código que se pueden ejecutar independientemente del código que tienen antes o después, es decir, es como una celda de código de un Colab Notebook: no es necesario ejecutar todo el programa entero, se puede ir paso a paso.
- Al igual que un Colab Notebook, se han de ejecutar las celdas en orden la primera vez que se carga el fichero para tener las funciones y variables de celdas anteriores en memoria. Una vez realizado esto se pueden ejecutar las celdas en cualquier orden.
- Las celdas se delimitan con un comentario de la forma "`#%%`" y pueden ejecutarse haciendo `Ctrl+Enter` en una celda resaltada o bien haciendo clic en el icono en la Figura 1 que está a la derecha del icono de "Play" que ejecuta el código entero secuencialmente.
- Las celdas están organizadas de manera que la primera celda abarca todas las funciones implementadas, y luego existe una celda por cada ejercicio, de esta manera solamente es necesario ejecutar la celda inicial, denominada la celda 0 y la celda del ejercicio que se desee ejecutar, el cual se encuentra apropiadamente identificada en el código.
- También se muestra donde comienzan y terminan las funciones de cada ejercicio para facilitar la legibilidad de la práctica.
- Se tiene una función auxiliar `stop()` que puede añadirse en cualquier lugar que se piense sea necesario para detener la ejecución.
- La entrega posee dos ficheros: `p3_1.py` donde se encuentra el problema de clasificación y `p3_2.py` que contiene el de regresión.

Figura 1: Icono para ejecutar la celda seleccionada

1. Problema de Clasificación

1.1. Análisis del problema

1.1.1. Descripción del problema

El problema que se tiene trata sobre campañas de marketing de una institución bancaria portuguesa, estas campañas fueron realizadas llamando a los clientes del banco y preguntándoles si les interesaba hacer un depósito a plazo fijo en el mismo, la base de datos del problema posee un total de 45211 muestras, no tiene datos faltantes.

1.1.2. Identificando X, Y y f

Las características de entrada, aquellas que forman la matriz X del problema de aprendizaje se compone de datos del cliente además de datos sobre la campaña y contactos previos con el cliente:

1. Edad del cliente, `age`.
2. Trabajo que posee, `job`.
3. Estado civil, `marital`.
4. Educación, `education`.
5. Si ha tenido impagos, `default`.
6. Saldo medio anual en Euros, `balance`.
7. Si tiene hipotecas, `housing`.
8. Si tiene préstamos personales, `loan`.
9. Medio de contacto, móvil o fijo, `contact`.
10. Día del mes del último contacto, `day`.
11. Mes del ultimo contacto, `month`.
12. Duración de la última llamada, `duration`.
13. Número de contactos previos en esta campaña, `campaign`.
14. Días que pasaron desde el último contacto de una campaña previa, `pdays`.
15. Número de contactos previos a esta campaña para el cliente, `previous`.
16. Resultado de la última campaña bancaria, `poutcome`.

El vector Y de salidas o etiquetas para este problema es bien un "si" o "no" como respuesta a esa oferta ofrecida en la campaña de marketing, desde el punto de vista del aprendizaje automático, esto es claramente un problema de clasificación binaria, pues se desea saber dadas diferentes características de entrada, si la salida es positiva o negativa.

Se puede pensar que la función teórica y desconocida $f : X \to Y$, la que genera o mapea todos los X con sus etiquetas correctas Y sería la situación personal, mental y financiera en la que se encuentra cada cliente cuando está en curso la campaña de marketing, pues su disposición a querer participar en la campaña, es decir, dejar un dinero congelado de su cuenta por un tiempo fijo dependerá — al menos intuitivamente — mucho de si ese cliente puede mantenerse bien sin ese dinero por un tiempo, si el cliente sabe y conoce lo que le están ofreciendo y que al cliente le interese en general la oferta y crea que pueda beneficiarse de la misma.

Del punto de vista del autor se piensa que las características X elegidas para el problema permiten aproximar f al menos intuitivamente, porque los factores como la estabilidad económica del cliente

puede verse reflejado tanto en la edad, el trabajo, educación y si posee impagos, hipotecas, préstamos e incluso si posee un teléfono móvil o fijo, intuitivamente uno podría imaginar que alguien ya de mediana edad, con un trabajo estable que no ha tenido impagos, tiene hipotecas y préstamos, se encuentra con el capital para aceptar la oferta, también el conocimiento de lo que se está ofertando puede verse reflejado en la educación y el trabajo que realiza, alguien que solo haya cursado educación primaria o bien realice un trabajo muy desconectado de temas de economía capaz desconozca o no entienda del todo que le ofertan y por lo tanto rechace la oferta; por último los datos sobre las campañas previas pueden dar un poco de luz sobre la personalidad del cliente, podría pensarse que un cliente que ya ha sido contactado antes y haya aceptado anteriores ofertas sea alguien que sienta que le está sacando provecho a las campañas de marketing y probablemente le interese esta oferta también, mientras que alguien que siempre les ha colgado y nunca haya aceptado se encuentre más cohibido a aceptar esta oferta.

1.1.3. Visualizaciones del conjunto de entrenamiento

Una vez se han separado los datos, descrito en detalle en la Sección 1.3, se procede a analizar los atributos anteriormente mencionados de la **muestra de entrenamiento** para entender el problema y tener una idea de como están distribuidos los datos.

Notar que para la realización de este experimento se ha obviado el uso del atributo `duration` de los datos originales, esto se ha realizado pues los autores de la base de datos indican que esta columna depende mucho de la etiqueta, y de hecho, fue rellenada luego de saber la etiqueta; por lo tanto siendo realistas este dato no podría saberse a priori para realizar una predicción de si un cliente aceptaría o no la oferta.

De las variables numéricas, se obtuvo su media, desviación típica, valores mínimos, máximos y los percentiles 25, 50 y 75 de los datos. Se puede observar esta información en la Tabla 1.

De las categóricas, se obtuvo las etiquetas únicas, cual es más común y su proporción con el total de datos, se puede observar en la Tabla 2.

Por último, el análisis de las etiquetas de salida, como ya se sabe, son solamente "sí" y "no", pero la proporción es de 11.70 % y 88.30 %.

De forma más visual, se pueden observar como están distribuidas con más detalle las variables categóricas y binarias en las Figuras 2 y 3.

	age	*balance*	*day*	*campaign*	*pdays*	*previous*
Media	40.99	1348.94	15.77	2.75	40.40	0.58
Desviación	10.63	2949.15	8.32	3.07	100.26	1.92
Valor mínimo	18.00	-8019.00	1.00	1.00	-1.00	0.00
Percentil 25	33.00	72.00	8.00	1.00	-1.00	0.00
Percentil 50	39.00	444.00	16.00	2.00	-1.00	0.00
Percentil 75	48.00	1413.00	21.00	3.00	-1.00	0.00
Valor máximo	95.00	81204.00	31.00	63.00	871.00	51.00

Tabla 1: Estadísticas de las variables continuas de la muestra de entrenamiento

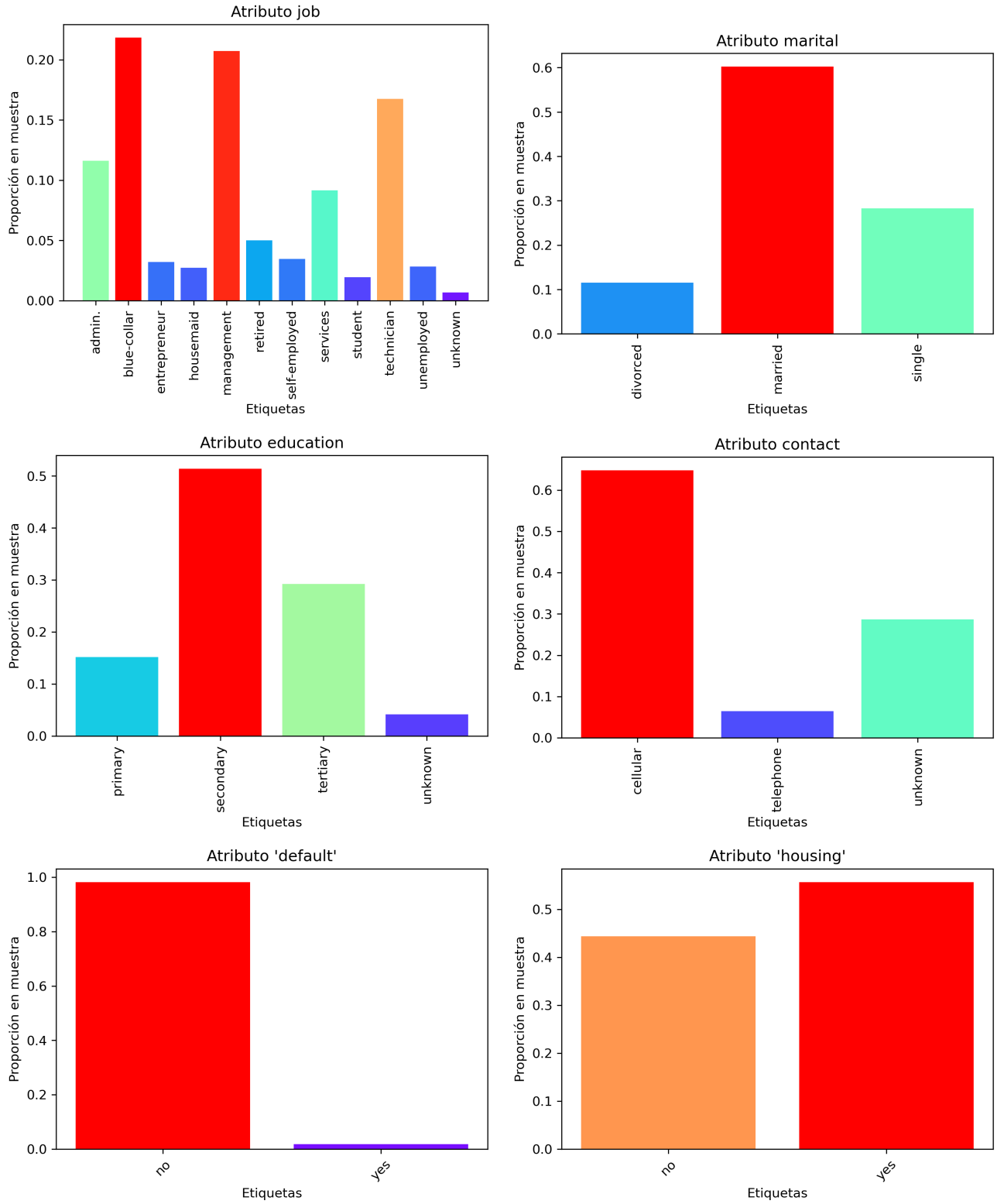

Figura 2: Distribución de los datos de muestra en variables categóricas y binarias.

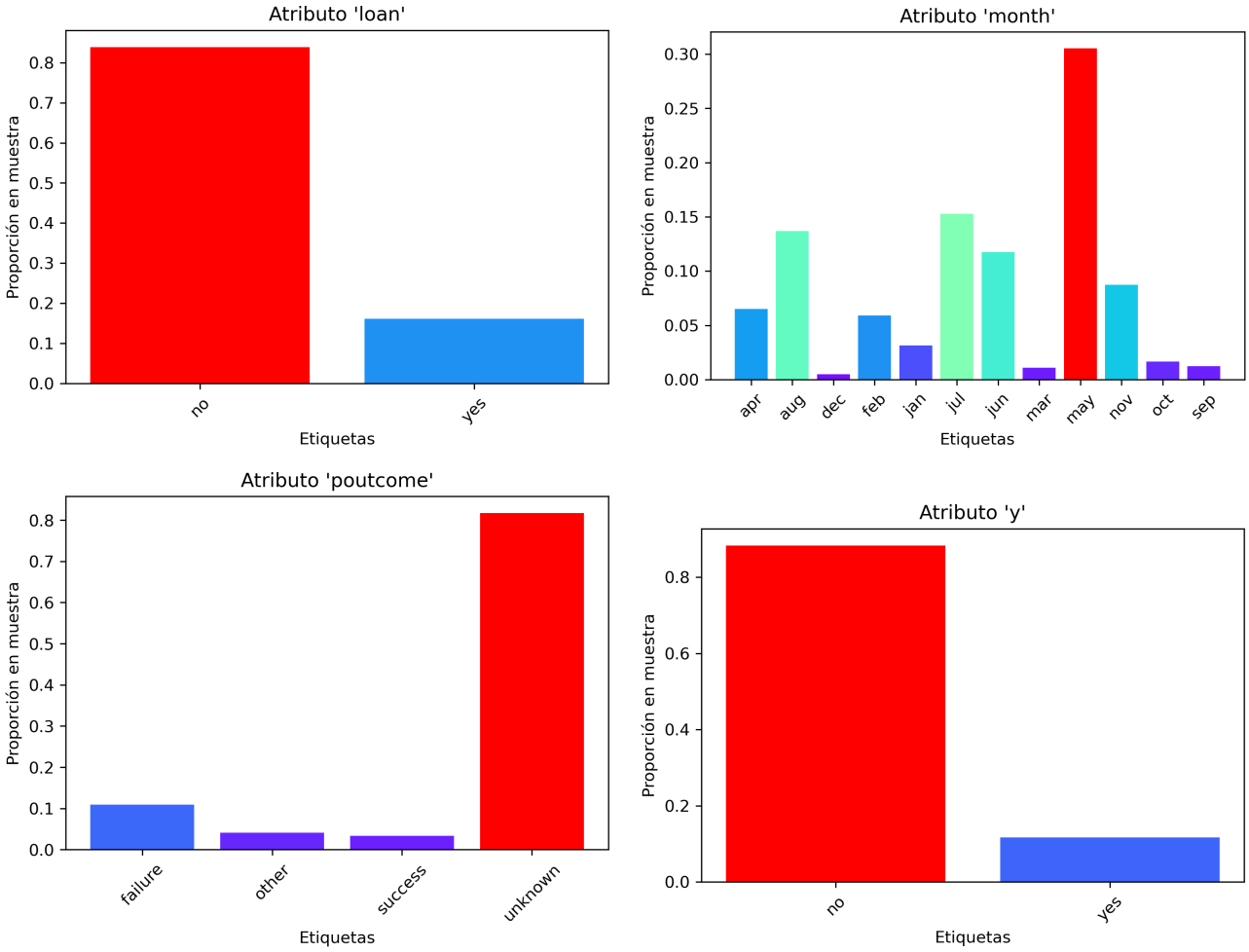

Figura 3: Distribución de los datos de muestra en variables categóricas y binarias, continuado.

	job	*marital*	*education*	*default*	*housing*	*loan*	*contact*	*month*	*poutcome*
Etiquetas únicas	12	3	4	2	2	2	3	12	4
Etiqueta más común	*blue-collar*	*married*	*secondary*	*no*	*yes*	*no*	*cellular*	*may*	*unknown*
% de la muestra	21.86	60.22	51.41	98.20	55.65	83.86	64.83	30.54	81.67

Tabla 2: Estadísticas de las variables categóricas de la muestra de entrenamiento.

Viendo todo esto se puede tener un poco más de entendimiento del problema a tratar, en primer lugar ya se observa que, —al menos la muestra de entrenamiento— posee un gran desbalanceo de las etiquetas, con un 80 % siendo "no". Ciertamente tiene sentido que una gran mayoría de los que reciben una llamada sin previo aviso de publicidad de su banco negarán la oferta por ser una molestia. Esto permitió definir una métrica de error adecuada al problema, esto se discute en la Sección 1.5.

De igual forma se puede observar que las variables continuas poseen escalas diferentes, y también poseen valores mínimos y máximos bastante diversos, por ejemplo `age` y `balance`, por otro lado se puede ver también que otros datos están más centrados y agrupados, mientras que otros bastante sesgados, por ejemplo `pdays` y `previous` que la mayoría de sus percentiles poseen un mismo valor, aunque luego poseen valores máximos mucho más grandes, y una desviación típica bastante alta, o sea que los valores están muy dispersos.

Lo mismo ocurre con las variables categóricas, se puede obtener una intuición con la Tabla 2, pero es más claro al ver las Figuras 2 y 3. En la gran mayoría de las variables se observa una categoría que se encuentra mucho más representada que el resto, esto también da a entender que estos datos poseen bastante sesgo y desbalanceo al igual que la etiqueta de salida, no parece que siguieran alguna clase de distribución, nuevamente esto ha sido de utilidad para las métricas de error y que parámetros definir a la hora de realizar la validación cruzada entre diferentes modelos lineales.

1.2. Conjunto de hipótesis

Debido a que se están utilizando solamente modelos lineales, se tiene a disposición PLA-Pocket y regresión logística.

Bien se sabe que PLA-Pocket trabaja con valores deterministas mientras que regresión logística trabaja con probabilidades, viendo que se está tratando con un problema del mundo real, donde se sabe que existe ruido inherente en los datos aunque no falte ninguno, y la función f descrita depende de lo que piensa una cliente en un determinado momento — queriendo decir que pueden haber características muy similares que tengan una etiqueta diferente —, es mucho más favorable trabajar con probabilidades y por lo tanto se seleccionó regresión logística.

1.3. Generación de conjuntos de training y test

Los datos no se encuentran separados por defecto, para generar los conjuntos de training y test se hizo uso de la función `train_test_split(X, Y, test_size, stratify)` con los parámetros `X` indicando el conjunto de características del problema entero, `Y` indicando el conjunto de las etiquetas del problema entero, con `test_size` se indicó la proporción de muestras para el conjunto de pruebas, en este caso se eligió que sea un 30 % de las muestras por ser uno de los valores

estándar en aprendizaje automático.

Se utilizó el parámetro `stratify=Y` para mantener en todo momento la misma proporción de etiquetas de los datos en los conjuntos de entrenamiento y de test, ya que se conoce que al menos en las muestras de entrenamiento los datos se encuentran desbalanceados. No se tocaron los demás parámetros de la función `train_test_split()` porque no se pensaron que fueran necesarios o bien porque ya el valor por defecto es el que se desea, por ejemplo, el parámetro `shuffle` que es para barajear las muestras ya se encuentra activado por defecto, el tamaño del conjunto de training se dejó por defecto que en este caso es la proporción opuesta del conjunto de test, es decir, un 70 % de los datos.

Se realizó de esta manera para luego aplicar la técnica de Validación Cruzada de k pliegues / k-fold Cross Validation para seleccionar el mejor modelo de regresión logística dado los datos de entrenamiento, que serán a su vez subdivididos en datos de entrenamiento por pliegue y de validación.

1.4. Preprocesado de datos

Antes de realizar algún ajuste, se realizaron distintas manipulaciones a los datos para evitar añadir sesgos por la forma en que se encuentran codificados originalmente o bien facilitar el ajuste al escalarlos a un mismo rango de valores

Para las variables categóricas que no son binarias, es decir, las variables `job`, `marital`, `education` y `contact` se realizó el típico One-Hot Encoding, para esto se utilizó la función de Pandas `get_dummies()` que está específicamente implementada para esto, así se evita imponer alguna clase de orden a las variables, por ejemplo, se podría pensar en codificar `education` de 0 a 4, pero esto podría implicar que tener una educación universitaria tiene más peso que tener solamente educación secundaria y no necesariamente tiene que ser así.

La variable `poutcome` tuvo un tratamiento diferente por la descripción que se tiene de la misma y lo que significa para el problema: La variable puede tener los valores `sucess`, `failure`, `unknown` y `other`. En vez de utilizar One-Hot que estaría añadiendo 4 columnas adicionales, esta variable puede codificarse casi de forma binaria, con `sucess` mapeado a 1, `failure` mapeado a -1 pues es lo opuesto y `unknown` y `other` a 0 pues en ambos casos es ambigua la respuesta, no es ni buena ni mala para el problema, al menos así se ha interpretado.

Las variables binarias — `default`, `housing`, `loan` y la etiqueta `y` — fueron codificadas de `yes/no` a $1, -1$.

Las variables cíclicas — `day`, `month` — se codificaron como senos y cosenos; se realiza de esta manera dado que la codificación original se presta a que sea mal interpretada por el modelo, ya que por ejemplo Diciembre y Enero están a un mes de distancia, pero en la codificación numérica estarían a 11 unidades de distancia lo cual no refleja la realidad, igual que los días 30 o 31 de un mes con el día 1 del siguiente mes.

En primer lugar se transformó la variable `month` de tener el nombre del mes a tener el número del mes — con la función implementada `monthToInt(data, col, monthDict)` que toma el Dataframe, la columna y un diccionario — así también se evitó hacer un One-Hot Encoding que añadiría 12 columnas, luego de esto se aplicó la transformación siguiente a los datos, tanto a `day` como a

`month`, se utilizó `valor_maximo=31` y `12` respectivamente.

$$\texttt{dato\_sin} = \sin\left(\frac{2 \times \pi \times \texttt{dato}}{\texttt{valor\_maximo}}\right) \tag{1}$$

$$\texttt{dato\_cos} = \cos\left(\frac{2 \times \pi \times \texttt{dato}}{\texttt{valor\_maximo}}\right) \tag{2}$$

De esta forma, se está mapeando el dato a una parte del seno y del coseno, los cuales son periódicos y de esta manera se mantiene la naturaleza cíclica de la variable de forma que sea "entendible" por el modelo [1].

Para la realización de toda esta sección en primer lugar se definieron varias listas, conteniendo las variables categóricas, binarias, con mapeado particular y cíclicas; las últimas dos se componen de diccionarios para indicar de que forma se tiene que mapear de manera más específica la variable.

Esto luego se pasa junto con el DataFrame de Pandas a la función definida `processData(data, categorical, binary, cyclical, custom)` para que se apliquen los cambios descritos anteriormente.

Luego, para las variables numéricas, visto que poseen valores en diferentes rangos se realizó un escalado de las mismas usando el objeto de Scikit `MinMaxScaler(feature_range)` dónde `feature_range` es una tupla del valor mínimo y máximo al que se desea escalar las variables.

Con el objeto, se utiliza su método `fit(columnas)` para ajustar los parámetros a las `columnas`. Se utilizaron los datos de entrenamiento para este ajuste, luego se utilizó el método `transform()` para escalar tanto los datos de entrenamiento como de test al ajuste realizado.

Esto se encapsuló dentro de la función `normalizeData(train, test, normCols, featRange=((-1, 1)))` que toma los datos de entrenamiento, de test, las columnas a escalar y el rango que se desea, por defecto escalar entre -1 y 1.

Para este problema se escalaron las variables `age`, `balance`, `pdays`, `campaign` y `previous`.

1.5. Métrica de error

Siendo un problema de clasificación, la primera métrica que uno pensaría en utilizar es la del *accuracy*, es decir, la proporción de aciertos respecto al total de muestras; el problema con esta métrica es que en problemas desbalanceados puede dar valores demasiado optimistas, pues si se tiene un 80 % de etiquetas negativas, un clasificador tenderá a siempre predecir la clase dominante y se obtendrá una *accuracy* alta sin que realmente se haya aprendido.

Como se pudo observar en la Sección 1.1.3 la base de datos posee un desbalanceo de las etiquetas fuerte, en este caso, se pudo observar en la literatura de Aprendizaje Automático [2] y de las diapositivas de teoría que en estos casos una métrica más recomendada es Valor-F1 / *F1-Score* pues toma en cuenta las métricas de *Precision* y *Recall* que reportan las muestras que están mal clasificadas, por lo tanto, representa mejor la calidad del ajuste cuando los datos están desbalanceados, justo lo que sucede en este problema.

Se utilizaron de igual forma ambas métricas para dar un contraste de los valores reportados pero vale la pena adelantar que se utilizó el Valor-F1 para la selección del mejor modelo, adicionalmente

se incluye la función de pérdida de entropía cruzada para poder dar los valores de Error de las diferentes muestras.

1.6. Parámetros y regularización

Para realizar el ajuste en concreto se utilizó la clase `SGDClassifier()`, pues la misma permite utilizar Regresión Logística para ajustar el modelo.

La manera en que se seleccionaron los parámetros ha sido utilizando la clase `GridSearchCV()` que permite definir en un diccionario los distintos parámetros que se desean explorar junto con los valores que pueden tomar, la clase toma esos valores y realiza una validación cruzada de cada modelo y va "recorriendo" ese espacio de modelos para obtener los parámetros que mejor ajustan los datos de entrenamiento dentro de la validación cruzada.

Los valores que se consideraron para generar los diferentes modelos de validación cruzada fueron:

- `class_weight`, que indica si se debe asignar un peso a cada etiqueta inversamente proporcional a la frecuencia de aparición de las mismas.
- `alpha`, controla el término de regularización y para computar la tasa de aprendizaje si está puesta como `optimal`; se consideraron los valores 0,0001, 0,001 y 0,01. Se eligieron estos valores siguiendo recomendaciones generales [3].
- `eta0`, la tasa de aprendizaje inicial, se consideraron los valores 0,01, 0,1 y 0,3, nuevamente se eligieron al ser valores por defecto según la literatura consultada [3].
- `learning_rate`, el tipo de tasa de aprendizaje, se determinó que los que funcionan mejor para Regresión Logistica son `adaptive` y `optimal`, la `adaptive` indica que la tasa se mantiene constante mientras el error se reduzca, cuando se estanca se divide entre 5, `optimal` utiliza una expresión heurística para calcularla.

Aparte de esto, se ajustó el número de épocas con `max_iters` — el nombre no ayuda, pero es lo que indica la documentación — puesto a 10000 épocas; el resto de parámetros bien no aplican al utilizar la pérdida de entropía cruzada, `log_loss` o por defecto se encuentra fijado como se desea, por ejemplo, que se realice el barajeo de los datos.

Siendo la versión estocástica de la implementación del descenso de gradiente, se está utilizando un `batch_size = 1` implícito; de igual forma se sabe por la teoria y prácticas anteriores que para regresión logística, mientras más pequeño el `batch_size`, generalmente mejor funciona el modelo.

Para la regularización, se utiliza la regularización L1 o Lasso pues se piensa que dados los datos que se tienen, existen variables con tantos valores a un atributo en particular que es fácil pensar que no todas las variables aporten al problema, por lo tanto, al usar L1 el clasificador podrá poner cerca o a 0 aquellas que sean necesarias y poder concentrarse en el resto de variables que aporten al ajuste.

Una vez los valores se definen en un diccionario, se crea un objeto `GridSearchCV(estimator, param_grid, scoring, refit, cv, n_jobs)` con `estimator` siendo `SGDClassifier(loss='log', penalty='l1', max_iter=10000)`, `param_grid` siendo el diccionario conteniendo la información previamwnte descrita, `scoring` contiene la métrica que mejorar, en este caso y como se mencionó

antes, se está utilizando la métrica Valor-F1 pues los datos están desbalanceados, con 10 pliegues de validación cruzada y `n_jobs=-1` para utilizar todos los núcleos de la CPU para reducir el tiempo de cálculo.

Una vez se tiene el objeto definido, se realiza la validación cruzada con `fit(train_x, train_y)`.

1.7. Mejor hipótesis para el problema

Una vez finalizada la ejecución del método `fit()` de `GridSearchCV`, se obtuvo como resultado del mejor modelo una F1 promedio de 0.3356, y un error de entropía cruzada para el *cross validation*, E_{CV}, de 11.0427 que fueron obtenidos con los siguientes parámetros: `'alpha': 0.0001, 'class_weight': 'balanced', 'eta0': 0.1, 'learning_rate': 'optimal'`.

Obtenidos estos valores, se procedió a realizar el ajuste del clasificador con todos los datos de entrenamiento, esto dio como resultado un E_{train} de 11.0428, un valor F1 de 0.3311 y una *accuracy* de 68.03 %.

El ajuste con los datos de test obtuvo un E_{out} de 11.2093, un F1 de 0.3275 y una *accuracy* de 67.55 %.

Se puede observar la Matriz de Confusión en la Tabla 3 y las métricas relacionadas en la Tabla 4.

		Actual	
		Positivo	Negativo
Predicción	Positivo	1072	515
	Negativo	3887	8090

Tabla 3: Matriz de Confusión del problema

	Precision	*Recall*	*F1-score*	Soporte
-1	0.94	0.68	0.79	11977
1	0.22	0.68	0.33	1587
Accuracy			0.68	13564
Macro Avg	0.58	0.68	0.56	13564
Weighted avg	0.86	0.68	0.73	13564

Tabla 4: Distintas métricas de clasificación

Como puede observarse, en general el modelo no logra ajustarse bien a los datos, en primer lugar, si bien la *Accuracy* es relativamente alta, casi llegando a 70 %, puede verse por ejemplo que lo que concierne a la clase positiva los valores son mucho menores que en la clase negativa, indicando que el modelo tiende a predecir la clase negativa más que la positiva. Lo que podría esperarse al tener tal desbalanceo de los datos y se intentaron varios métodos para mitigarlos, se concluye que es necesario emplear técnicas más avanzadas para poder mejorar el modelo.

1.8. Curvas de aprendizaje

Una vez realizada la validación cruzada, el ajuste a los datos de entrenamiento y la obtención de los valores de test, se generaron dos curvas de aprendizaje del modelo por medio de la función

`learning_curve()` de Scikit-Learn, haciendo uso de un *10-fold cross validation* para cada tamaño de datos.

Se pueden apreciar las curvas en la Figura 4, que fueron realizadas respecto al Error de Entropía Cruzada y a la métrica F1.

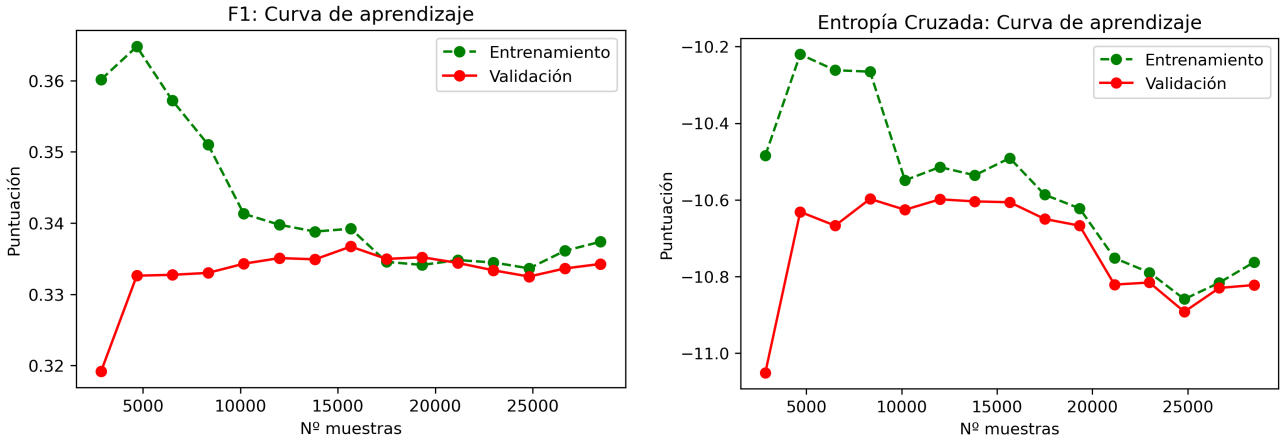

Figura 4: Curvas de Aprendizaje centrándose en la métrica F1 y en el Error de Validación Cruzada

Si bien la forma que describen tiene cierto sentido con lo visto en teoría, por ejemplo en la curva de aprendizaje de la medida F1, donde se obtienen valores altos en el subconjunto de entrenamiento en cada pliegue y mucho más bajos en el conjunto de validación y eventualmente ambos se acercan a un valor intermedio al aumentar la cantidad de datos; esto sucede en menor manera en la curva del error de entropía cruzada.

Se puede concluir de las curvas lo mismo que se pudo concluir del análisis del E_{out} de la Sección anterior, el modelo no se está ajustando muy bien a los datos, se obtienen valores de error bastante altos y la métrica F1 bastante baja; nuevamente esto debido al fuerte desbalanceo de los datos que se tiene.

Aunque se intentaron varias técnicas básicas para intentar mejorar el ajuste, lo cierto es que se piensa que utilizar un modelo lineal es limitante, aunque puede verse que las métricas evolucionan, indicando que hay aprendizaje, lo cierto es que el error al que se aproximan sigue siendo muy alto, esto es indicativo de *underfitting*, lo opuesto de *overfitting*; se necesita de un modelo más complejo para ajustar estos datos de forma satisfactoria.

1.9. Modelo para una empresa

Si se tratase de un modelo para una empresa, es decir, que la empresa ha proporcionado los datos sin distinción entre lo que sería el conjunto de entrenamiento y el conjunto de test, para este problema podría suponerse que lo que se ha estado utilizando es una parte de una muestra mayor — por ejemplo, que la empresa tenga reservado su propio conjunto de test privado —, en todo caso, se piensa que lo mejor sería presentar el modelo actual, aquel que obtuvo los mejores resultados en validación cruzada con una proporción reducida de los datos y luego entrenado y probado con el conjunto de test generado, se presentaría el E_{out} pues se piensa de esta manera se está dando una idea más realista y capaz más pesimista de la calidad del modelo, puesto que si dio los mejores

valores luego de los 10 pliegues de validación cruzada, tiene sentido pensar que el modelo es el que logra generalizar mejor todos los datos presentados.

El único problema, claro está es que si el tamaño de la base de datos es relativamente pequeña, entonces se pueden tener cotas de error alto, aún así en este caso la base de datos posee un tamaño significativo y por lo tanto se utilizó validación cruzada.

De todas formas, se deberían probar más técnicas para poder sobrellevar el desbalanceo de datos, ya sea aplicando alguna técnica para generar o duplicar los datos de una clase para balancear la proporción o bien utilizar modelos más complejos como SVMs o Redes neuronales, que por motivos de tiempo estas técnicas no pudieron ser implementadas en esta práctica.

2. Problema de Regresión

2.1. Análisis del problema

2.1.1. Descripción del problema

El problema que se tiene trata sobre la predicción del año, comprendido entre 1922 y 2011, en el que se publicó una canción ciertas características acústicas de dicha canción, la base de datos consiste en 515345 muestras diferentes, no tiene datos faltantes.

2.1.2. Identificando X, Y y f

Las características de entrada, aquellas que forman la matriz X del problema de aprendizaje se compone de 90 características acústicas: 12 de las mismas tratan sobre el timbre acústico promedio de la canción para las 12 notas musicales, las 78 características restantes son las covarianzas de cada una de las notas musicales con las otras, notar que si bien técnicamente serían $12 \times 12 = 144$ covarianzas en total, puesto que las covarianzas de una nota i con otra nota j es la misma que la nota j con la nota i, se da que $12 \cdot \frac{13}{2} = 78$ diferentes covarianzas entre los timbres de las notas.

El vector Y de salidas o etiquetas para este problema comprende valores continuos del año en que se publicó una canción, desde el punto de vista del aprendizaje automático esto es un claro problema de regresión lineal, pues debe de estimarse un valor numérico y no una clase; aunque también sería posible transformarlo a un problema de clasificación, no se piensa sea lo más lógico.

Se puede pensar que la función teórica y desconocida $f : X \to Y$, la que mapea los X a las Y se trata en este caso del estilo de música popular cuando se produjo la canción, además de las idiosincrasias del artista que la produjo y también el género de la canción, pues estos patrones también influyen en la composición musical de la misma, de todas formas se puede pensar que aún con estos factores, para un año determinado ya sea por la tecnología existente o las nociones de la música, todas las canciones de un año poseerán patrones similares.

Del punto de vista del autor, tiene sentido las características X del problema, pues se está analizando directamente la acústica de la canción y como se relaciona el timbre de una nota promedio con el resto de las notas, de forma se puede pensar que surgirán patrones identificables para los distintos años.

2.1.3. Visualizaciones del conjunto de entrenamiento

Una vez separados los datos, descrito en la Sección 2.3, se procede a analizar las variables que componen el conjunto de datos para tener una idea más clara del problema, se visualizaron únicamente los datos que se encuentran en el **conjunto de entrenamiento**.

En este problema solo se tienen variables numéricas, a las cuales a unas cuantas se le obtuvo su media, desviación típica, valores mínimos, máximos y percentiles 25, 50 y 75, puesto que analizar 90 variables diferentes es intratable, solamente se desea obtener una intuición de cómo están dispuestos los datos. Se puede observar esta información en la Tabla 5

	1	2	3	4	...	87	88	89	90
Media	43.39	1.35	8.68	1.18	...	-26.11	4.46	20.02	1.33
Desviación	6.06	51.52	35.24	16.32	...	173.90	13.35	184.26	22.03
Valor Mínimo	2.64	-337.09	-291.02	-154.18	...	-3819.93	-217.19	-4969.06	-318.22
Percentil 25	39.96	-26.03	-11.43	-8.49	...	-101.26	-2.57	-59.42	-8.82
Percentil 50	44.26	8.53	10.52	-0.65	...	-21.10	3.12	7.78	0.07
Percentil 75	47.83	36.16	29.75	8.80	...	52.61	9.98	86.35	9.72
Valor Máximo	61.49	384.07	322.85	228.41	...	2833.61	229.69	7393.40	677.90

Tabla 5: Una muestra de los tipos de valores que poseen las variables en la base de datos del problema

Puede observarse que los valores de las variables tienen rangos diferentes y varían muchísimo entre sí, al igual que el problema de clasificación esto se tomará en cuenta para las siguientes secciones con el objetivo de facilitar el proceso de entrenamiento, sobre los valores de las etiquetas, se puede observar un histograma con la frecuencia de aparición de los años en la Figura 5.

Puede observarse que nuevamente los datos poseen un desbalanceo hacia la década de los 2000s, con muy muy pocos datos entre los años de 1920 y 1960; esto da a conocer que este problema también posee un desbalanceo significativo de las etiquetas y por lo tanto se tendrá que aplicar ciertas métricas para contrarrestarlo dentro de lo posibe.

2.2. Conjunto de hipótesis

Puesto que se está realizando el ajuste con modelos lineales, la opción obvia es utilizar regresión lineal, puesto que es método que ha sido estudiado para esta clase de problemas, existiendo también la Pseudoinversa pero como bien se sabe para el tamaño que posee este problema, utilizarla es inviable; por lo tanto se decidió regresión lineal.

También sería posible convertir este problema en un problema de clasificación, pero tener un vector de salida de casi 100 etiquetas no parece ser lo más ideal, como se ha comentado previamente.

Naturalmente dentro del conjunto de hipótesis de regresión lineal existen mucha variación, se explorará que parámetros e hiperparámetros se considerarán de este modelo para ajustar el modelo.

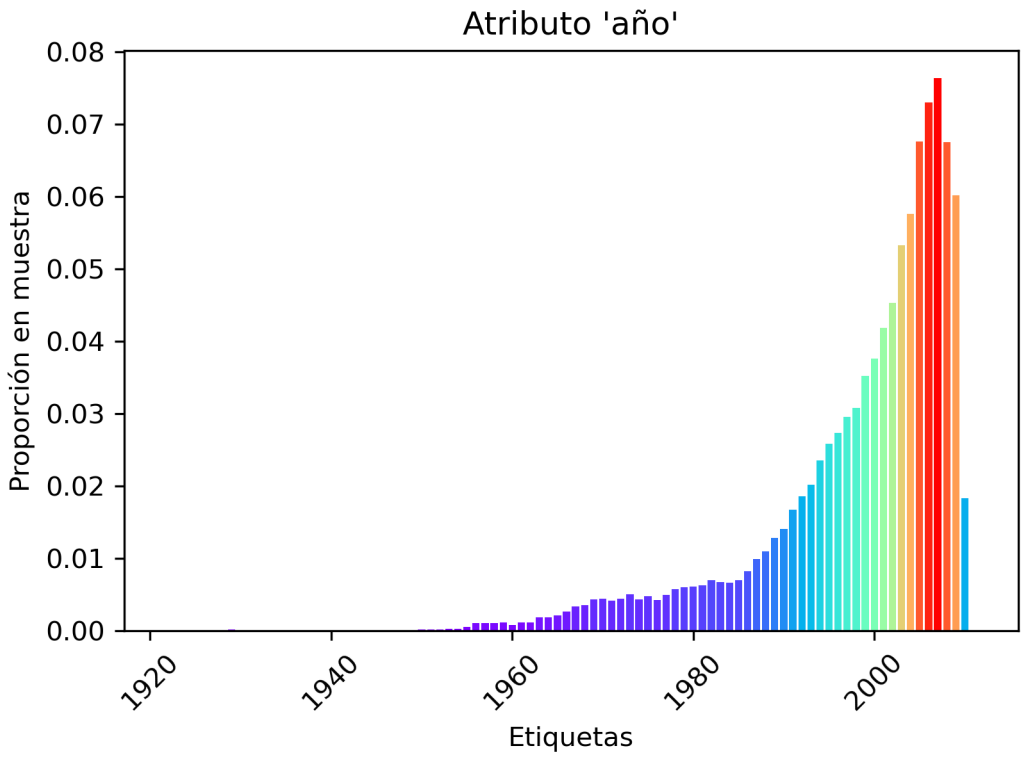

Figura 5: Frecuencias de aparición de años en las etiquetas de los datos de entrenamiento

2.3. Generación de conjuntos de training y test

En la página de la base de datos se tienen dispuestos unos conjuntos de entrenamiento y test definidos por los autores, los cuales fueron ignorados: se cargó en su totalidad los datos al programa y de igual forma que con el problema de clasificación, se utilizó la función `train_test_split()` con los mismos parámetros exceptuando `stratify` porque existen años en donde solamente se tiene una muestra y por lo tanto, no es posible dividir proporcionalmente los datos.

Se mantiene la división 70-30 de los datos, al igual que el otro problema, se utilizó este valor por ser los valores estándares de división de datos, y el resto de parámetros de la función ya se encuentran bien por defecto, por ejemplo, el parámetro `shuffle=True` para barajear dichos datos.

2.4. Preprocesado de datos

Antes de realizar los ajustes, dado que se nitó al explorar los datos de entrenamiento que los mismos poseen diferentes escalas y valores, en primer lugar lo que se realizó fue el escalado de los mismos, se utilizó el `MinMaxScaler()`, reutilizando la función mencionada previamente `normalizeData()` para escalar todos los valores a un rango entre -1 y 1.

Hecho esto, otro problema que se pudo observar es que se tratan de demasiadas variables, 90 variables por 515345 muestras en total; lo cual se convierte en un número intratable a la hora de entrenar un 70 % de esos datos.

Por lo tanto, se ha empleado también la reducción de dimensionalidad por medio de PCA; para realizar esto se llama a la función `PCA(n_components)` de Scikit-Learn para generar un objeto con

porcentaje `n_components` de varianza, en este caso se utilizó para mantener un 90 % de la varianza, una vez creado el objeto se llama al método `fit(data)` con los datos de entrenamiento y luego se transforman tanto los datos de entrenamiento como de test con el método `transform(data)` del objeto, ahora se ha reducido las variables de 90 a 55, un valor mucho menor; de esta forma se podrá entrenar más rápido manteniendo todavía la información de covarianza de datos, además que con rangos de valores similares que también ayudan al algoritmo a encontrar mínimos en el espacio de búsqueda.

2.5. Métrica de error

Siendo un problema de regresión, la métrica que uno utilizaría de error es la del Error Cuadrático Medio o MSE que es la que se ha impartido en teoría, aún así, realizando investigaciones en la literatura se ha encontrado que una métrica más intuitiva e igual de útil es la de utilizar el coeficiente de determinación R^2, que básicamente igual que MSE, indica que tan bien se está realizando un ajuste con el beneficio de que los valores se encuentran siempre entre 0 y 1, siendo el ajuste mejor si los valores obtenidos por el coeficiente se acercan a 1.

Se utiliza igualmente MSE como métrica secundaria para dar un contraste de los valores, además así se puede dar el error en cada conjunto de datos, aunque la selección del mejor modelo fue realizado con los valores que se obtenían de R^2.

2.6. Parámetros y regularización

Para realizar el ajuste como tal se utilizó la clase `SGDRegressor()` pues la misma permite utilizar regresión lineal con el algoritmo de gradiente descendente estocástica, recordando que la implementación de Scikit-Learn aplica un `batch_size=1` implícito.

La manera en que se seleccionaron los parámetros para el mejor modelo fue por medio de la clase `GridSearchCV` que, como se ha mencionado previamente, permite definir un diccionario con los distintos parámetros que se desean explorar junto con los valores que pueden tomar.

Por cada configuración distinta se realiza la validación cruzada y se van almacenados los que posean la mejor puntuación de la métrica que se ha definido.

Los valores que se consideraron para generar los distintos modelos fueron:

- `alpha`, que controla el término de regularización con valores 0.001 y 0.01.
- `eta0`, la tasa de aprendizaje inicial, se consideraron los valores 0.01 y 0.1.
- `learning_rate`, el tipo de tasa de aprendizaje, al igual que para clasificación se consideraron `adaptive` y `optimal`.

Si bien se hubiera querido probar con más configuraciones, la cantidad tan grande de datos hace que este proceso de búsqueda en el espacio de modelos sea significativamente más lenta que para el problema de clasificación y por lo tanto se redujo el número de opciones para que la ejecución no se tardara más de 5 minutos.

Aparte de esto se dejaron fijadas el número de épocas on `max_iters` a 10000; la métrica de error del objeto se fija a `squared_error` para indicar regresión lineal.

Para la regularización, se ha fijado a L2 o Ridge, pues habiendo observado la muestra de datos y también el significado de las variables de los datos que poseen del problema, se piensa que es mejor que todas las variables puedan aportar algo a la regresión en vez de hacer que algunas se vayan a cero, pues con tanta variabilidad que puede haber en la música no se puede descartar que existan muchas características sutiles que afecten la predicción.

Una vez se definieron los valores en el diccionario, se creó como tal el objeto `GridSearchCV()` donde se le pasa como parámetro el `SGDRegressor()` con unos parámetros ya fijados, se pasa también el diccionario de los parámetros con los que se va a probar la validación cruzada, utilizando como métrica para puntuar el modelo el coeficiente de correlación R^2 con 3 plieges por validación cruzada para reducir el tiempo de cómputo, al igual que en el problema de clasificación, se utiliza `n_jobs=-1` para utilizar todos los hilos disponibles en la CPU para acelerar el cálculo.

Una vez definido, se realiza la validación cruzada y selección del mejor modelo con el método de `GridSearchCV`, `fit(red_train_x, train_y)`.

2.7. Mejor hipótesis para el problema

Una vez finalizada la ejecución del método `fit()` se obtuvo un coeficiente de determinación R^2 de 0.2214, un MSE $E_{CV} = 92{,}8717$ con los parámetros `'alpha': 0.001, 'eta0': 0.1, 'learning_rate': 'adaptive'`.

Obtenidos estos valores se procedió a realizar el ajuste de todos los datos de entrenamiento, esto dio como resultado un E_{train} de 92.8724 con un coeficiente R^2 de 0.2217 y para E_{out} se tiene 93.3340 y un R^2 de 0.2212

Se puede observar que en general el modelo no logra un ajuste bueno a los datos, esto se observa tanto en el coeficiente R^2 pues se encuentra bastante alejado del mejor valor 1, y también porque el valor obtenido de los errores es bastante alejado del 0, que sería el ajuste perfecto.

2.8. Curvas de aprendizaje

Una vez realizada la validación cruzada, el ajuste de datos de entrenamiento y de test, se generaron dos curvas de aprendizaje nuevamente con la función `learning_curve()` utilizando un 3 Fold Cross Validation por motivos de tiempo.

Se pueden observar las curvas en la Figura 6, que se realizaron respecto al valor del coeficiente R^2 y el MSE.

De los gráficos puede verse que en general las curvas siguen la tendencia esperada, con el conjunto de Entrenamiento teniendo un error más alto, mientras que el de Validación tiene uno menor, y mientras se van añadiendo más datos, las curvas convergen a un valor en particular; en ese sentido se puede ver que están aprendiendo, pero al mismo no es suficiente, pues los modelos se estancan en errores muy grandes o bien en un coeficiente R^2, al menos con esto se puede ver que no está sucediendo *overfitting*, en todo caso, está ocurriendo lo opuesto: *underfitting*. Por lo tanto, se requeriría utilizar técnicas o modelos más complejos para poder mejorar aún más el ajuste de datos.

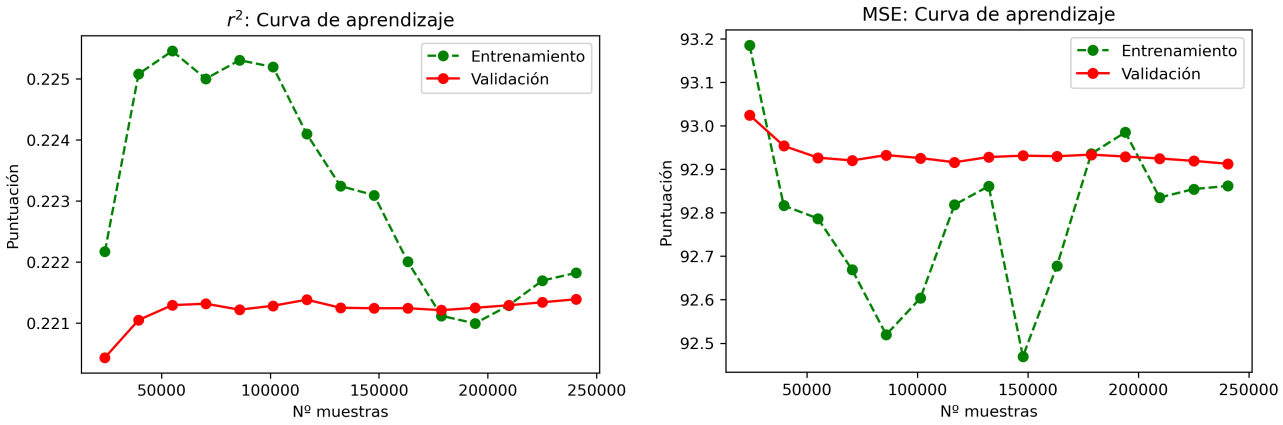

Figura 6: Curvas de Aprendizaje centrándose en el coeficiente de determinación R^2 y en el Error Cuadrado Medio (MSE).

2.9. Modelo para una empresa

En una situación hipotética donde una empresa es la que nos da acceso a estos datos sin distinción entre entrenamiento y test, dependiendo de la cantidad de datos se podría realizar lo que se ha hecho en estos experimentos, primero particionarlos como se realizó y luego realizar la validación cruzada, de la validación se encontraría el modelo que generaliza mejor el modelo y con los parámetros del mismo se entrenaría en todo el conjunto de entrenamiento, evaluando el E_{out} en la partición de Test.

Naturalmente si la base de datos es exageradamente grande — más incluso que la de este problema — realizar la validación cruzada puede resultar demasiado demandante y requerir demasiado tiempo, por lo que o bien se podrían utilizar menos pliegues o bien utilizar solamente Hold-Out, que es lo que se realiza al dividir en primer lugar el conjunto de Entrenamiento y Test.

Utilizar tanto Hold-Out como Validación Cruzada implican dividir más y más los datos provistos, aunque si se tienen suficientes datos no resulta ser un problema, aunque si hay que notar que si es cierto que lo ideal sería poder utilizar todos los datos disponibles para el ajuste, pues con más datos se tiene una vista más general de la población, lamentablemente, realizar esto puede dar como resultado una métrica poco realista de la población puesto que no se tienen otros datos que nunca se han visto para compararlo, luego no se puede saber si el modelo está sobrentrenando o si realmente está generalizando bien.

Referencias

[1] Avanwyk, *Encoding cyclical features for Deep Learning*, abr. de 2018. dirección: `https://www.kaggle.com/code/avanwyk/encoding-cyclical-features-for-deep-learning/notebook`.

[2] G. Lahera, *Unbalanced datasets & what to do*, jul. de 2019. dirección: `https://medium.com/strands-tech-corner/unbalanced-datasets-what-to-do-144e0552d9cd`.

[3] J. Korstanje, *The F1 score*, ago. de 2021. dirección: `https://towardsdatascience.com/the-f1-score-bec2bbc38aa6`.