AJUSTE DE MODELOS LINEALES MEMORIA P3 DE APRENDIZAJE AUTOMÁTICO

Ignacio Garach Vélez

4 de junio de 2022

Índice

1. Introducción

Este trabajo se centra en el ajuste y selección del mejor predictor lineal para dos conjuntos de datos, un problema de regresión y otro de clasificación. Para ello, se utilizará la librería Scikit-Learn. Esta librería contiene funciones de alto nivel para la labor que nos ocupa, tanto para visualización con la ayuda de Seaborn, codificación, preprocesado y entrenamiento y validaciónd de modelos. En cada función se explica porque es necesario su uso, se explica su funcionamiento y el significado y razones de peso para escoger sus parámetros. Se realizan al final estudios de sus curvas de aprendizaje y se estima el error fuera de la muestra y otros métodos de separación de datos que se podrían haber usado.

2. Problema de clasificación

Los datos que nos ocupan son relativos a una campaña telefónica de un banco portugués, en general se tiene un esquema sencillo de características $\mathcal{X}$ compuesto de variables numéricas como la edad o la duración de las llamadas, variables categóricas como el tipo de empleo o educación y variables binarias como la propiedad de una vivienda. En la sección de codificación se estudiarán más en profundidad. Por otro lado, se debe predecir $\mathcal{Y} = \{0, 1\}$ que corresponde a si el cliente contrató el producto que se le estaba ofertando.

Por tanto estamos en un problema de clasificación binaria a partir de 16 variables inicialmente. Esto es buscamos $f : \mathcal{X} \to \mathcal{Y}$ desconocida que para nuevas instancias de datos prediga correctamente si se contratará el producto.

2.1. Previsualización y codificación

Vamos a hacer algunas observaciones previas y un análisis de variables para aprender algo acerca del problema:

1. Se realiza una búsqueda de nulos o inválidos, resultando en la comprobación de la completitud del dataset.

2. Se observa que contamos con varias variables binarias codificadas con el string 'yes' o 'no', esto no será adecuado para el entrenamiento de modelos que como sabemos hacen cálculos con los valores de las variables, por tanto se codifican a 0 el no y a 1 el sí. En particular esto ocurre con las variables default, housing, loan y con la clase y.

3. Algo similar ocurre con las múltiples variables categóricas que se observan en el dataset, como pueden ser educación o empleo. Se había pensado en realizar una codificación por ordinales pero dado que la mayoría de las variables no están jerarquizadas y aparecen valores unknown (que no es razonable eliminar pues reduciriamos enormemente el dataset) se ha optado por codificar mediante la técnica One Hot Encoding. Dicha técnica codifica en valores binarios cada categoría posible de cada variables, esto es por ejemplo si existieran 3 empleos distintos se sustituiría la columna por 3 nuevas columnas que contienen en cada instancia 1 uno y 2 ceros dependiendo del empleo. En particular esto se realiza pàra las variables job, marital, education, contact y poutcome.

4. Finalmente se codifican los datos temporales día y mes mediante transformaciones seno y coseno. La idea residual de esto es darles un sentido periódico para el modelo. En general una sóla transformación dejaría el mismo valor para 2 datos distintos por la periodicidad de las funciones, por tanto se introducen ambas para la distinción unívoca. Por tanto se introducen 4 nuevas columnas que sustituirán a day y month por sus funciones trigonométricas previa normalización al intervalo de definición habitual del seno y del coseno. Esto tiene perfecto sentido para estos modelos lineales, sin embargo si en el futuro queremos entrenar modelos basados en árboles deberíamos replantearlo ya que no tienen capacidad para usar dichos pares de columnas de forma conjunta y separarían los datos perdiendo información.

2.2. Identificación de hipótesis

Vamos a utilizar modelos lineales, sabemos que para clasificación tenemos disponibles el perceptron, regresión logística y regresión lineal para clasificación, ya avanzamos que utilizaremos regularización Ridge para evitar el Overfitting. Luego por tanto el conjunto de hipótesis son todas las funciones lineales en los pesos:

$$H = \{h : \mathbf{R}^{d+1} \to \mathbf{R} : h(x) = w^T x \qquad w \in \mathbf{R}^{d+1}\}$$

a las que se le calcula la clase de clasificación ya sea en función del signo de $w_T X$ o mediante una función sigmoide en los distintos modelos. Hemos considerado la introducción de características de grado 2 pero aumentaba mucho la complejidad del modelo sin mejorar los resultados y disparaban los tiempos de ejecución luego lo hemos descartado.

2.3. Preprocesado

2.3.1. Separación en conjuntos de entrenamiento y test

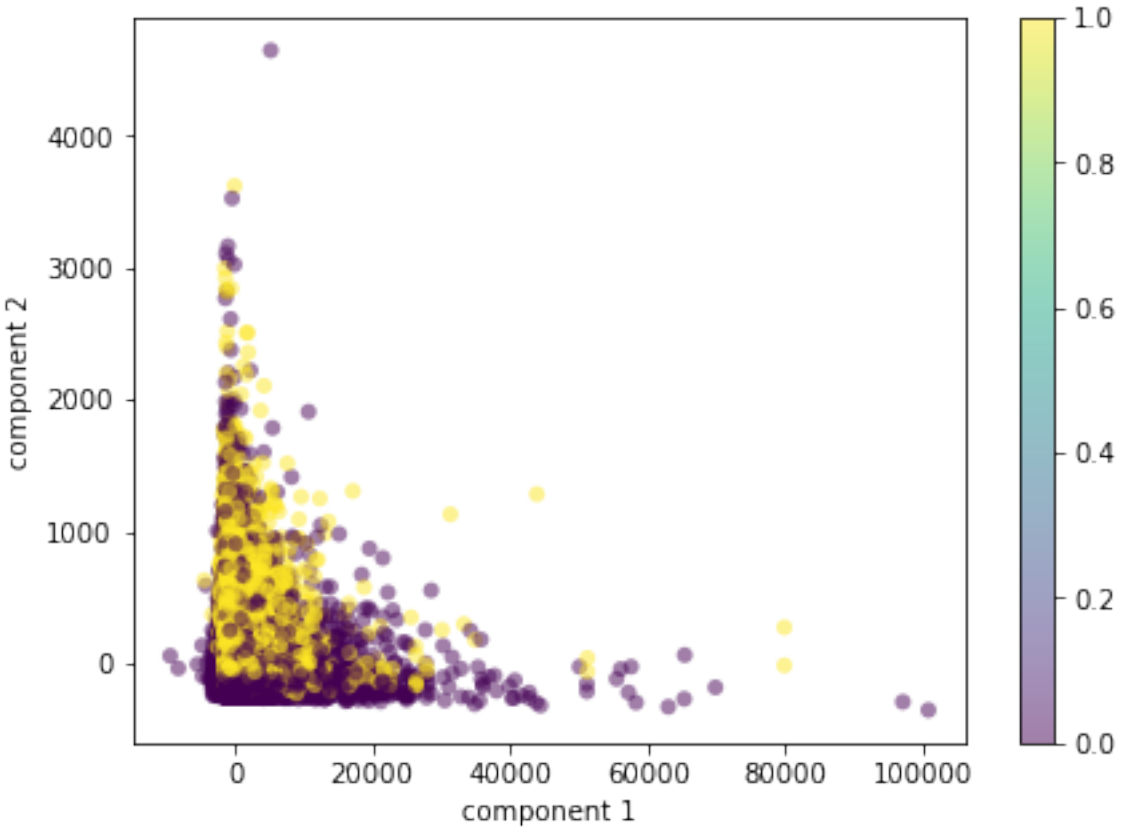
En primer lugar para no sesgarnos en nuestra exploración vamos a realizar una separación de los datos en conjuntos para entrenamiento y test, a posteriori para la estimación de parámetros se dividirá a su vez de nuevo entrenamiento.

Observamos que las clases estan totalmente desbalanceadas, por tanto para que el modelo realmente instancias de las 2 clases sera necesario utilizar una estratificación que mantenga datos de ambas clases tanto en entrenamiento como en test. Por otro lado contamos con bastantes datos luego se realizará una separación aleatoria estratificada con un 80 por ciento para entrenamiento y 20 por ciento para test, una regla del pulgar vista en teoría, tenemos bastantes datos y es un caso razonable. Comprobamos despues que en efecto hay una proporción de clases en cada conjunto.

Se ha utilizado la función traintestsplit de scikitlearn con los parámetros indicados arriba.

2.3.2. Ánalisis de componentes principales

El análisis de componentes principales es una técnica que permite simplificar de espacios de alta dimensionalidad, se trata de encontrar un pequeño número de variables subyacentes que consigan explicar aproximadamente lo mismo que todas las demás. Es una técnica que usaremos para hacernos una idea de la distribución de nuestros datos y de la posible presencia de datos anómalos. Utilizaremos la versión de ScikitLearn con reducción a 2 dimensiones de modo que podamos representarla y aprender algo. Mostramos a continuación dicho gráfico:



Se observa que el grueso de los datos estan distribuidos en una zona, tanto casos positivos como negativos, por lo tanto parece que obtener tasas muy altas de acierto en clasificación será complicado. Por otro lado, existen datos muy alejados que parecen anómalos, analizaremos en el siguiente apartado si puede tratarse de outliers y si debemos eliminarlos.

2.3.3. Ánalisis de varianza y outliers

Con la ayuda de la función describe de pandas obtenemos estadísticas generales de cada variable, no se observan variables que tengan una varianza extremadamente baja con respecto a las demás, por lo tanto no parece sensato eliminarlas pues estaríamos potencialmente perdiendo poder predictivo, además no tendremos un problema de tiempos previsiblemente ya que no tenemos excesiva dimensionalidad.

En cuanto a lo referido a datos anómalos, la visualización del apartado anterior nos ha hecho plantearnos el uso de algún método para eliminarlos aunque no con demasiada agresividad pùes parecían algunos de ellos casos aislados. Vamos a utilizar un método basado en árboles para ello conocido como IsolationForest.

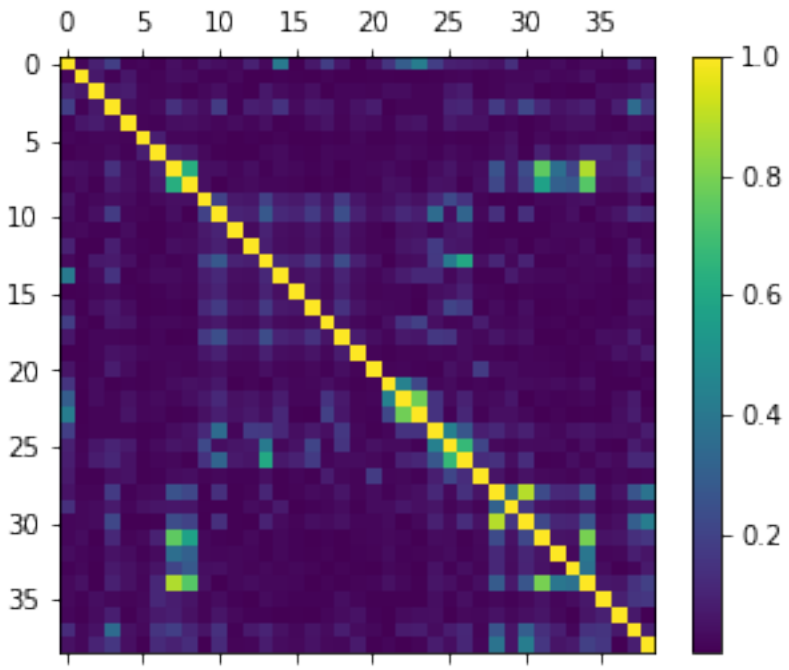
Este método esta inspirado en el algoritmo de RandomForest visto en teoría, se genera un nodo raíz con los datos de entrenamiento, se selecciona un atributo aleatorio y un valor dentro de su rango de valores, se separan 2 nodos separando según dicho valor y se repiten los pasos anteriores hasta llegar a los nodos hoja. Esto se varias veces para tener un bosque en lugar de un sólo árbol. El valor predicho para cada observación es el número de separaciones en promedio que se han necesitado para aislar dicha observación en el conjunto de árboles. Cuanto menor es el valor, mayor es la probabilidad de que se trate de una anomalía o outlier.

Hemos usado la función asociada de la librería con la recomendación de tamaño del bosque (200) del autor para nuestro tamaño de datos, igualmente se probó con distintos tamaños y resultado similar a partir del umbral de 100. Se añadió una probabilidad de existencia de outliers de un 5 por ciento por lo observado en visualización y se le pasa en el parámetro indicado. Trás

esto eliminamos algo menos de un 5 por ciento de los datos que el algoritmo ha considerado como outliers

2.3.4. Estudio de correlación y reducción de dimensionalidad

Con la ayuda de las herramientas de numpy visualizamos la matriz de coeficientes de correlación de Pearson entre las variables que indican la influencia de cada una en las otras, es importante para eliminar redundancia que no sean altas, si lo son esto indicaría que necesitamos eliminar variables.



Podemos observar que en pequeños nexos en torno a la parte baja de la diagonal hay zonas de alta correlación esto puede ser debido a la codificación utilizada para las variables categóricas, en efecto estará correlada la pertenencia a una categoría con la no pertenencia a otra, pero esto no es preocupante pues la hemos introducido nosotros. Las zonas alejadas de la diagonal pueden ser más problemáticas pues indican alta correlación entre variables supuestamente algo menos dependientes, esto nos lleva a reducir dimensionalidad mediantes el método de análisis de componentes principales explicado anteriormente de modo que con menos variables consigamos un grado de explicación similar y evitando correlaciones que puedan ser dañinas para los algoritmos de aprendizaje y puedan provocar overfitting.

2.3.5. Estandarización

En la descripción que proporcionaba pandas observamos que había cambios de escala muy importantes en las diferentes variables, esto es dañino pues los algoritmos por construcción pueden dar mas peso a variables con valores más grandes. Además es necesario aplicarla para poder utilizar el algoritmo de reducción de la dimensionalidad de forma satisfactoria.

La estandarización es el reescalado de los datos para que se adapten a una distribución normal de media 0 y varianza 1. Usaremos la función StandardScaler de scikitlearn. Es importante que para evitar fenómenos de Data Snooping o Data Leakage no se utilicen los datos de test, pues sesgarían el entrenamiento. Por tanto el proceso es escalar con los datos de entrenamientos y despues aplicar dicha escala adaptada a entrenamiento también a los datos de test.

2.4. Métricas de error

Como es un problema de clasificación binaria es natural utilizar como métrica el accuracy o exactitud definida como tasa de acierto en clasificación:

$$Accuracy(h) = \frac{1}{N}\sum_{i=1}^{N}[[h(x_n) == y_n]]$$

Por tanto el error se define como 1 - Accuracy.
Como se tiene una distribución de clases desbalanceada, es importante darse cuenta que la accuracy puede no ser la mejor opción para mostrar la calidad del modelo, ya que un predictor absurdo que siempre prediga la clase más común va a tener buena accuracy. Por tanto vamos a definir la métrica F1 a partir de la precisión y el recall de forma que se penalicen elecciones como la del predictor absurdo.

Sabemos que la precisión y el recall se definen como:

$$Precision = \frac{\#\ of\ True\ Positives}{\#\ of\ True\ Positives + \#\ of\ False\ Positives}$$

$$Recall = \frac{\#\ of\ True\ Positives}{\#\ of\ True\ Positives + \#\ of\ False\ Negatives}$$

Es importante señalar que en caso de que numerador y denominador sean 0 la métrica por lógica será 0, esto es lo que implementa Scikit-learn y de hecho a nosotros nos ocurrirá en algunos casos.
La precisión nos informa acerca de la tasa de positivos predecidos correctamente entre todos los positivos predecidos. El recall nos informa acerca de la tasa de positivos encontrados entre todos los que realmente había.
El F1 Score se define como la media armónica de ambas tasas, por lo que será alta cuando ambas la sean, media cuando alguna flaquee y baja cuando ambas lo hagan.

$$F1\ score = 2 * \frac{Precision\ * Recall}{Precision + Recall}$$

2.5. Regularización

Es importante utilizar técnicas de regularización para reducir el overfitting y limitar la complejidad del modelo lo que hará que tengamos mayores posibilidades de generalización según el enfoque PAC que estamos usando. Contemplaremos las dos posibilidades habituales L1 y L2, Lasso y Ridge, podría utilizarse alguna combinación de las mismas pero finalmente se ha descartado por su complicada interpretación:

- Regularización L1 Lasso: Añadimos una penalización a la función de pérdida que es lineal con respecto a los pesos, es una suma de los valores absolutos de ellos.

$$Loss_{L1}(w) = Loss(w) + \lambda \sum_i |w_i|$$

- Regularización L2 Ridge: Añadimos una penalización a la función de pérdida que es cuadrática con respecto a la norma de los pesos:

$$Loss_{L2}(w) = Loss(w) + \lambda ||w||_2^2$$

El λ es un parámetro del modelo que deberá ser ajustado en validación y que controla la agresividad de la regularización, si es pequeño no reducimos bien el overfitting pero si en muy grande podríamos acabar en el extremo contrario del underfitting y no ajustar correctamente los datos de entrenamiento.

Hemos tomado la determinación de utilizar el modelo de Ridge pues penaliza en mayor medida valores altos y funciona mejor cuando se consideran importantes todas las variables. Por otro lado, en el perceptrón como esperamos malos resultados le damos la posibilidad en validación de usar también L1 a ver si mejora.

2.6. Cross validation y elección del modelo

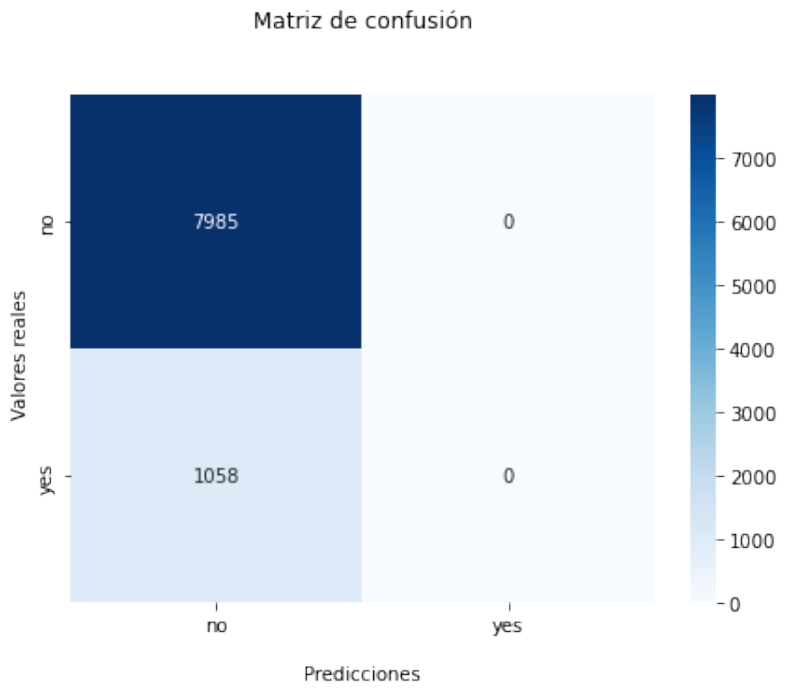
Vamos a seleccionar ahora los mejores modelos y observar que resultados proporcionan. Se usará la técnica de K-fold Cross Validation en la cual se divide el conjunto de entrenamiento en K subconjuntos y se entrena con K-1 de ellos y testea con el faltante, esto se hace en las K formas posibles. Nosotros usaremos K=5, un estándar habitual en ciencia de datos. Se calcula la media de los errores y se escoge el modelo de menor error. También usamos esto para estimar el mejor modo de regularización y los hiperparámetros. Tenemos gran comodidad usando la función GridSearch a la que le pasamos un diccionario con todos los posibles parámetros y realiza la validación brindandonos el modelo con mejores resultados en validación, esto es realiza todo el proceso descrito en una sóla línea de código.

Vamos a usar como modelos el perceptrón, la regresión logística y la regresión lineal para clasificación, observamos los resultados proporcionados por los mismos con los mejores parámetros:

2.6.1. Perceptrón

Los parámetros con mejor resultado han sido utilizar regularización L1, con $\lambda = 0.1$ y ha arrojado las métricas siguientes en el conjunto de Test y la siguiente matriz de confusión:

$$Accuracy : 0.883 \qquad F1Score : 0.000$$

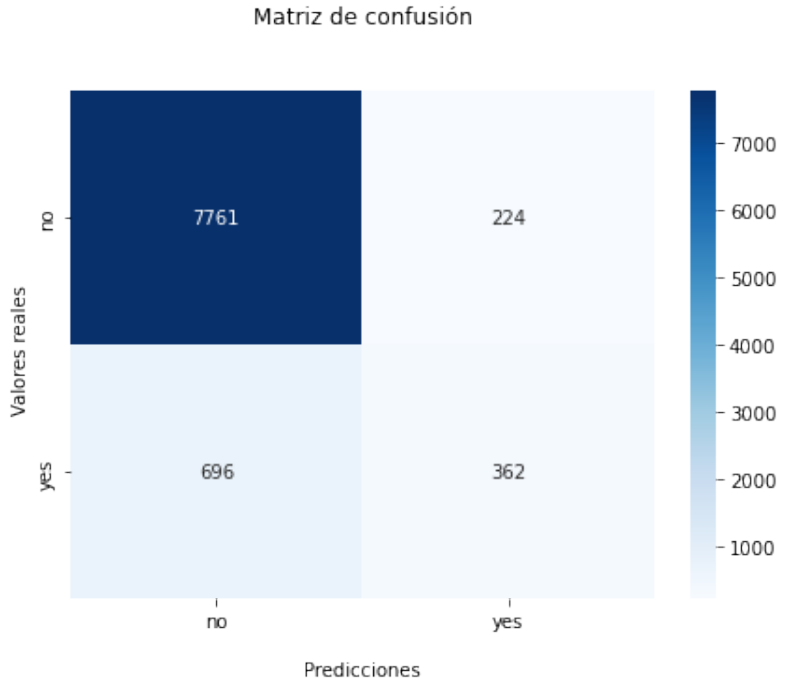


Esto nos muestra que el perceptron como era de esperar es muy mal modelo ya que no tiene que converger si los datos no son linealmente separables, coincide que la mejor forma que consigue de aumentar el accuracy es clasificar del mismo modo que un DummieClassifier que siempre clasifica la clase más común. Pero esto evidentemente es poco informativo. El F1 Score nos lo refrenda, es nulo, ya que todas las predicciones son No.

2.6.2. Regresión logística

Los parámetros con mejor resultado han sido utilizar regularización L2, con $C = 1$ y ha arrojado las métricas siguientes en el conjunto de Test y la siguiente matriz de confusión:

$$Accuracy : 0.899 \qquad F1Score : 0.44$$

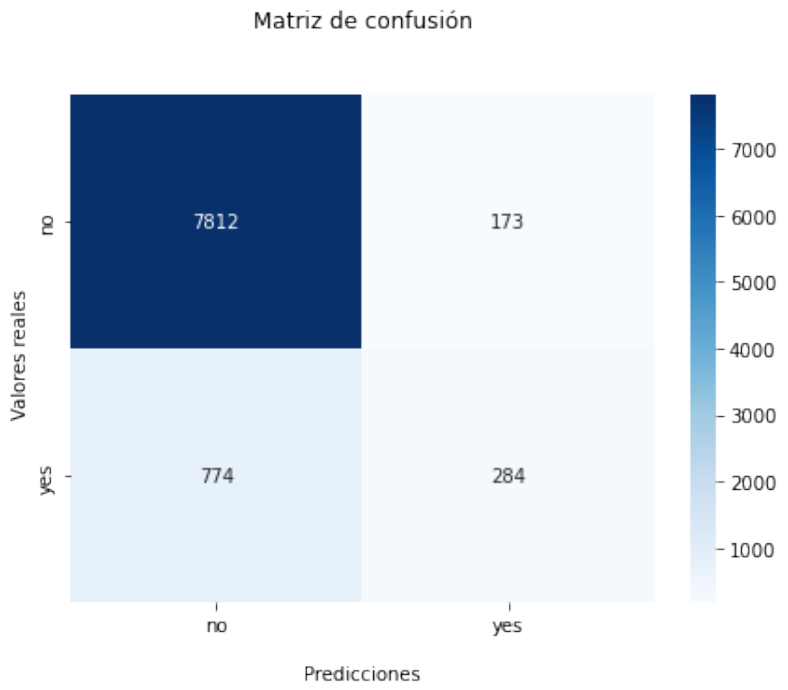


Observamos un comportamiento mejor en comparación con el perceptrón pues acierta también algunas clases positivas pero no son muy satisfactorios los resultados, parece que estamos ante un dataset complicado. El F1 score se acerca al 50 por ciento indicando un valor aceptable para la labor de predicción, este va a ser el mejor modelo que encontremos.

2.6.3. Regresión lineal para clasificación

Vamos a utilizar el objeto RidgeClassifier por simplificar el trabajo, ya tiene implementada la regularización, la elección del parámetro pr GridSearch ha sido $\alpha = 0.1$ y ha arrojado las métricas siguientes en el conjunto de Test y la siguiente matriz de confusión:

$$Accuracy : 0.895 \qquad F1Score : 0.375$$

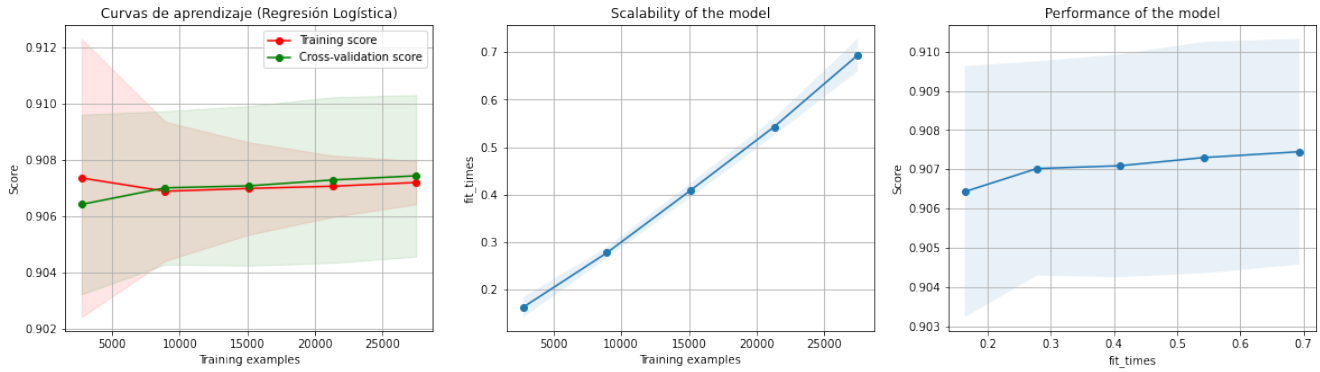


Observamos un comportamiento mejor en comparación con el perceptrón pero no se acerca el rendimiento promedio de la regresión logística. El F1 score es menor que en regresión logística indicando unos valores de precisión y recall menores y por tanto peores para la labor de predicción.

En definitiva el mejor modelo que hemos encontrado ha sido la regresión logística con regularización de Ridge.

2.7. Curvas de aprendizaje

Hacemos ahora un análisis en mayor profundidad del modelo con mejor resultado, usaremos como herramienta las curvas de aprendizaje. Esta curva tiene los resultados de ir entrenando el modelo con porciones cada vez mayores del conjunto de entrenamiento y reservando un conjunto de validación para ir evaluandolo y ver como aumenta el rendimiento.



En la primera curva se observa que el accuracy en entrenamiento va disminuyendo conforme aumentan los datos y como en validación avanza de forma inversa, es un comportamiento esperado y observamos que en términos de accuracy el modelo satura muy pronto y con pocos datos ambos accuracy son cercanos. Las otras gráficas muestran la escalabilidad del modelo en cuanto a tiempo respecto al número de ejemplos y el accuracy respecto al tiempo de entrenamiento, denotando que el accuracy no aumenta demasiado con el tiempo. Esto es debido a como hemos comentado anteriormente el desbalanceo provoca altos accuracys, se podría hacer un análisis con otras métricas para ver si realmente compensa tener muchos datos.

2.8. Estimación del error y conclusión

Es esperable que el error de Cross Validation sea un buen estimador del error fuera de la muestra. Por otro lado la mejor medida del error vendría a partir del error en test debido a que no se han usado en ningún momento y por la forma en que se han separado es probable que sean representativos de la población. Podemos estimarlo mediante la cota de Hoeffding que conocemos en test.

Fijada una tolerancia de 0.05 se tendría al 95 por ciento de confianza que el error de clasificación está acotado por:

$$E_{out} \leq E_{test} + \sqrt{\frac{1}{2N} log \frac{2}{\delta}} = 0.116$$

Aunque la cota de error de $E_{CV} = 0.092$ es mejor, es menos fiable así que disponemos la de arriba. Luego si debiera realizar este ajuste para una empresa hubiera realizado la separación en training-test anteriormente explicada, la cross-validation en 5-fold como hemos comentado y el mejor modelo sería regresión logística con regularización Ridge y podemos asegurar al 95 por ciento de confianza un error menor al 11 por ciento.

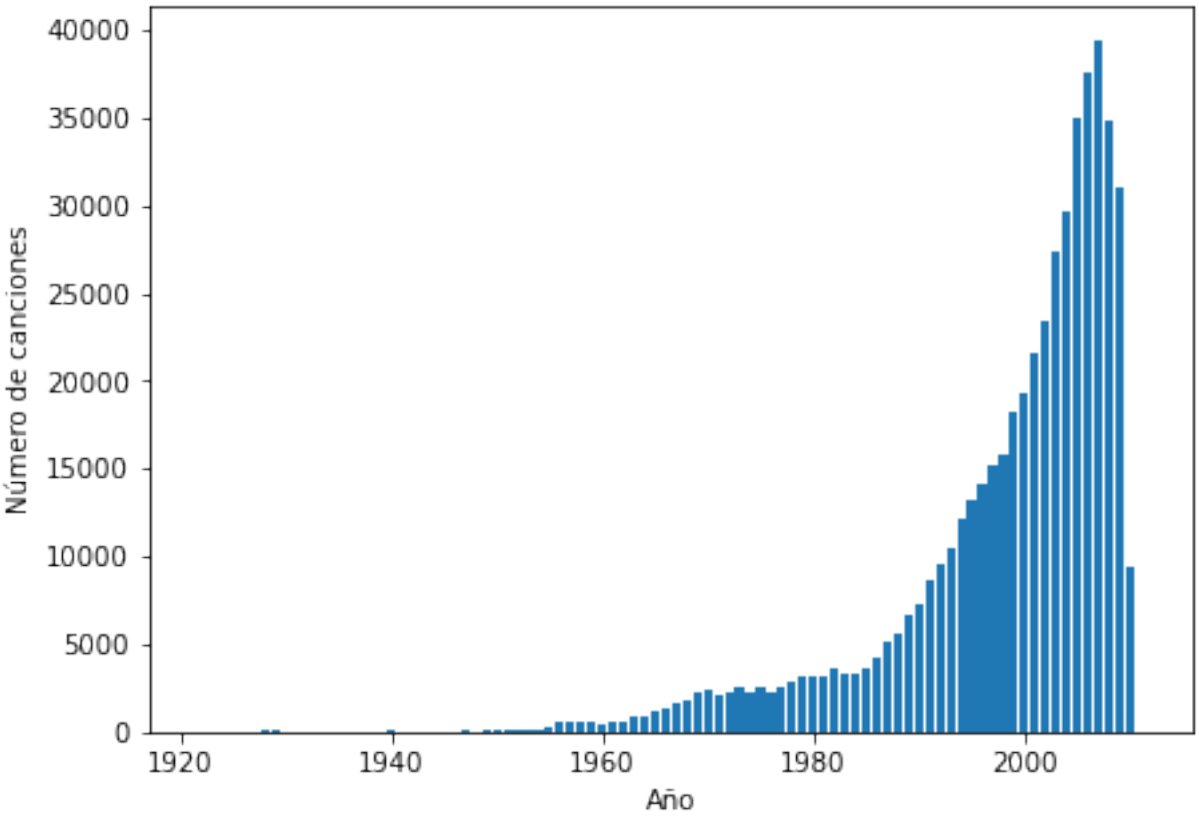
3. Problema de regresión

Los datos que nos ocupan son relativos a canciones mayormente de género western de los años 1922-2011 con gran cantidad en los años 2000, luego $\mathcal{X}$ está compuesto de variables numéricas referentes a medias de timbre y covarianzas de timbre de las canciones, luego no podemos extraer mucha información a priori son datos sin significado para nosotros. No necesitan por tanto ninguna codificación especial pues son valores reales. Por otro lado, se debe predecir el año de lanzamiento de la canción luego $\mathcal{Y} = [1922, 2011]$ es un intervalo real.

Por tanto estamos en un problema de regresión a partir de 90 variables inicialmente. Esto es buscamos $f : \mathbb{R}^{90} \to [1922, 2011]$ desconocida que para nuevas instancias de datos de canciones prediga su año de lanzamiento.

3.1. Visualización

Vamos a hacer algunas observaciones previas y un análisis de variables para aprender algo acerca del problema, en primer lugar observemos la distribución de datos con respecto al objetivo en un histograma:



Se observa que el grueso de los datos se encuentra en torno de los años 2000 aunque hay una cantidad considerable a partir de los años 70. Podríamos por tanto considerar un Dummy-Regressor basado en la media de los años de lanzamiento.

3.2. Identificación de hipótesis

Vamos a utilizar modelos lineales, sabemos que para regresión tenemos disponible regresión lineal, ya avanzamos que utilizaremos regularización Ridge y Lasso para evitar el Overfitting. Luego por tanto el conjunto de hipótesis son todas las funciones lineales en los pesos:

$$H = \{h : \mathbf{R}^{91} \to \mathbf{R} : h(x) = w^T x \qquad w \in \mathbf{R}^{91}\}$$

Hemos considerado la introducción de características de grado 2 pero aumentaba mucho la complejidad del modelo sin mejorar los resultados y disparaban los tiempos de ejecución luego lo hemos descartado por inviable. Ciertamente el dataset es inmenso, he tenido problemas serios con uso de CPU y RAM al tratar de hacerlo.

3.3. Preprocesado

3.3.1. Separación en conjuntos de entrenamiento y test

En primer lugar para no sesgarnos en nuestra exploración vamos a realizar una separación de los datos en conjuntos para entrenamiento y test, a posteriori para la estimación de parámetros se dividirá a su vez de nuevo entrenamiento.

Observamos que en este problema de regresión en el que tenemos una cantidad tremendamente elevada de datos no tiene sentido considerar mantener una consonancia con las instancias de la distribución que parece tender a la normal. Es esperable que un reparto aleatorio de al modelo suficientes ejemplos para aprender de todo el rango de valores.

Por otro lado contamos con muchos datos luego no hay problema en sacrificar buena parte de ellos para usarlos como test, luego usaremos un 80 por ciento aleatorio para entrenamiento y 20 por ciento para test, una regla del dedo gordo"vista en teoría.

Se ha utilizado la función traintestsplit de scikitlearn con los parámetros indicados arriba y en el código.

3.3.2. Estandarización

En la descripción y resumen de las variables que proporcionaba pandas observamos que había cambios de escala muy importantes en las diferentes variables, esto es dañino pues los algoritmos por construcción pueden dar mas peso a variables con valores más grandes. Además es necesario aplicarla para poder utilizar el algoritmo de reducción de la dimensionalidad de forma satisfactoria.

Procedemos como en clasificación usando la función StandardScaler de scikitlearn. Recordemos que para evitar fenómenos de Data Snooping o Data Leakage no se utilicen los datos de test, pues sesgarían el entrenamiento. Podremos ahora analizar en detalle las varianzas y correlaciones.

3.3.3. Normalización del Target

Este ha sido en mi opinión el mayor logro de mejora conseguido en la práctica, he observado que los valores del año eran demasiado grandes y espaciados y producían problemas de escala en los algoritmos empeorando bastante los resultados. He optado por utilizar una normalización hacia el intervalo $[0, 1]$ con la función MinMaxScaler() que ha conseguido que aumente el resultado del mejor modelo desde un R2 de 0.13 hasta casi el doble 0.23. Es importante reseñar que al finalizar la ejecución se ha utilizado la transformada inversa para devolver correctamente el año de lanzamiento.

3.3.4. Ánalisis de varianza y outliers

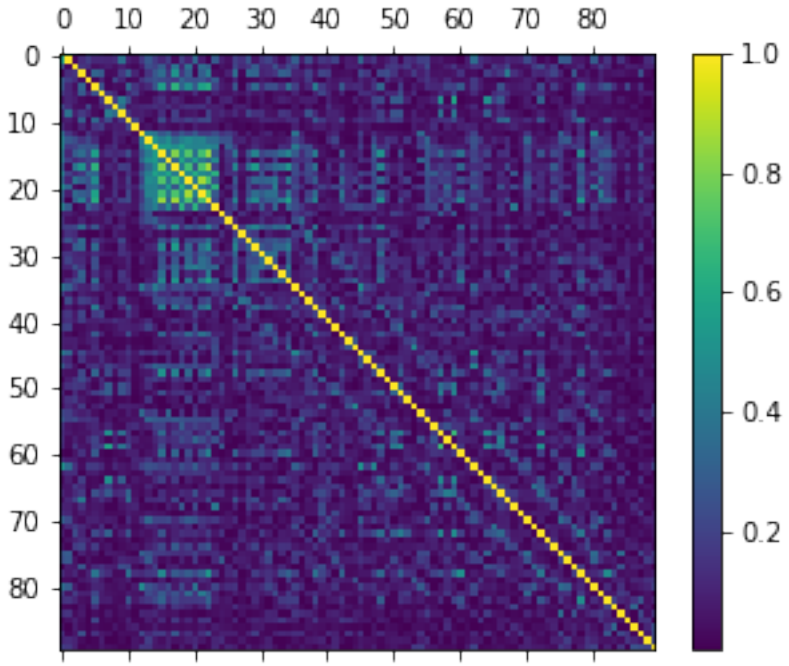
Con la ayuda de la función describe de pandas obtenemos estadísticas generales de cada variable, no se observan variables que tengan una varianza extremadamente baja con respecto

a las demás, sin embargo a posteriori realizaremos un análisis del umbral de varianza por si podemos reducir la dimensionalidad.

En cuanto a lo referido a datos anómalos, no tenemos indicios de los mismos como en el problema de clasificación aunque probablemente los haya. Sin embargo no sabemos como abordar de forma sensata este problema en tan alta dimensionalidad, sería posible indagar en algoritmos para localizarlos pero no lo consideramos el objetivo de esta práctica.

3.3.5. Estudio de correlación y reducción de dimensionalidad

Con la ayuda de las herramientas de numpy visualizamos la matriz de coeficientes de correlación de Pearson entre las variables que indican la influencia de cada una en las otras, es importante para eliminar redundancia que no sean altas, si lo son esto indicaría que necesitamos eliminar variables.



Se puede observar que de forma generalizada aparecen zonas con correlación intermedia y en la parte superior de la diagonal hay zonas de alta correlación. Esto puede ser problemático o positivo según se mire pues si utilizamos la técnica de análisis de componentes principales para reducir la dimensionalidad ahorraremos caractéristicas redundantes y disminuiremos el tiempo de entrenamiento. Además evitaremos correlaciones que puedan ser dañinas para los algoritmos de aprendizaje y puedan provocar overfitting. Finalmente no se ha hecho pues los resultados empeoraban y la penalización en tiempos no era grande.

3.4. Métricas de error

En este problema de regresión vamos a utilizar 2 métricas de error, el error cuadrático medio y el coeficiente de determinación R^2:

- MSE : El error cuadrático mide la media de las distancias al cuadrado entre la predicción y el valor real. Tiene un problema, no está acotada luego sólo se puede usar para comparaciones de un mismo conjunto, se centrará su uso como función de pérdida en los algoritmos de aprendizaje con los datos de entrenamiento.

$$MSE(h) = \frac{1}{N}\sum_{i=1}^{N}(h(x_i) - y_i)^2$$

- R^2 : El coeficiente de determinación proveniente del ámbito estadístico indica la bondad del ajuste y está acotado lo cual tiene muchas ventajas y es un gran indicador del desempeño del modelo.

$$R^2(h) = 1 - \frac{\sum_{i=1}^{N}(y_i - h(x_i))^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}, \quad \text{con } \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i.$$

3.5. Regularización

Es importante utilizar técnicas de regularización para reducir el overfitting y limitar la complejidad del modelo. Contemplaremos las dos posibilidades habituales L1 y L2, Lasso y Ridge, podría utilizarse alguna combinación de las mismas pero finalmente se ha descartado por su complicada interpretación. Hemos utilizado un rango de valores amplio recomendado para el GridSearch y adelantamos que el mejor resultado se ha obtenido con Lasso L1 para un valor del hiperparámetro $\alpha = 0.0001$.

3.6. Cross validation y elección del modelo

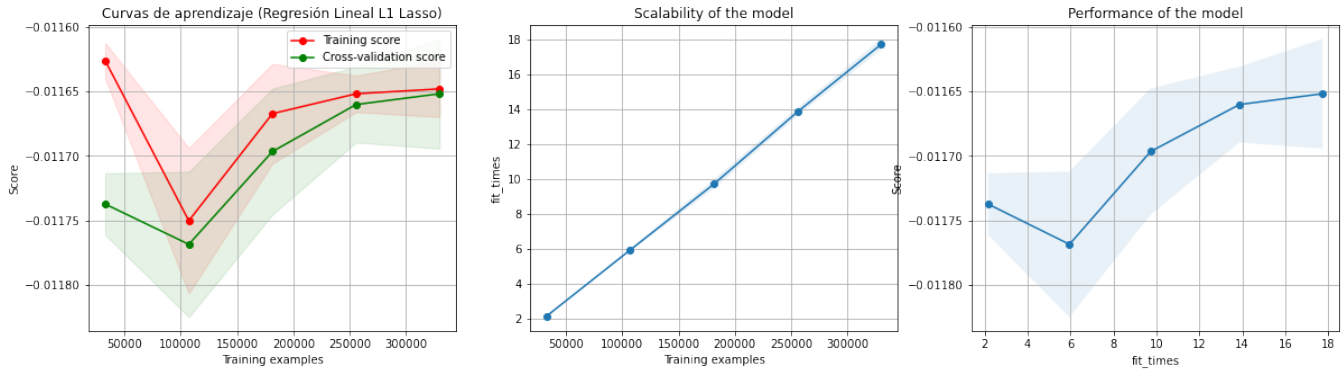
Vamos a seleccionar ahora los mejores modelos y observar que resultados proporcionan. Se usará la técnica de K-fold Cross Validation en la cual se divide el conjunto de entrenamiento en K subconjuntos y se entrena con K-1 de ellos y testea con el faltante, esto se hace en las K formas posibles. Nosotros usaremos K=5, un estándar habitual en ciencia de datos. Se calcula la media de los errores y se escoge el modelo de menor error. También usamos esto para estimar el mejor modo de regularización y los hiperparámetros.

- En este caso disponemos como opciones ambos modos de regularización con un espacio paramétrico equidistribuido dado por el intervalo $[10^{-4}, 10^4]$.
- En este caso debido al gran tamaño del dataset, utilizamos procesamiento paralelo con el parámetro njobs usando todos los núcleos.
- Como la función GridSearch trata de maximizar la métrica que se le pase, y queremos usar el error cuadrático medio necesitamos indicarle como parámetro su opuesto.
- La gran cantidad de datos hace que fijemos el número máximo de iteraciones en 1000.

Finalizamos indicando que el mejor resultado se ha obtenido con Lasso L1 para un valor del hiperparámetro $\alpha = 0.0001$.

3.7. Curvas de aprendizaje

Hacemos ahora un análisis en mayor profundidad del modelo con mejor resultado, usaremos como herramienta las curvas de aprendizaje. Esta curva tiene los resultados de ir entrenando el modelo con porciones cada vez mayores del conjunto de entrenamiento y reservando un conjunto de validación para ir evaluándolo y ver como aumenta el rendimiento.



La interpretación es muy similar a la que se dió en el problema de clasificación se tiene el comportamiento esperado con la cross validation con la excepción de un pico negativo en torno al tamaño 100000 no tenemos interpretación para él luego lo más probable es que sea debido al valor de la semilla aleatoria . Las otras gráficas muestran la escalabilidad del modelo en cuanto a tiempo respecto al número de ejemplos y el accuracy respecto al tiempo de entrenamiento.

3.8. Estimación del error y conclusión

Es esperable que el error de Cross Validation sea un buen estimador del error fuera de la muestra. Por otro lado la mejor medida del error vendría a partir del error en test debido a que no se han usado en ningún momento y por la forma en que se han separado es probable que sean representativos de la población.

Calculando el coeficiente de determinación en test se obtiene 0.235, es bastante malo pero estamos en un dataset complejo luego vamos a considerarlo aceptable, de hecho veremos que no es tan malo como parece si le damos un margen a la predicción.

Vamos a definir una pseudomedida de acierto del modelo, consideramos un margen de años en torno al valor real y consideramos acierto la pertenencia de la predicción al intervalo generado, finalmente dividiremos el número de aciertos entre el tamaño de test obteniendo una tasa. Esto tiene sentido pues el problema que nos ocupa era relativo a la evolución de las canciones con los años, es de esperar que las tendencias se mantengan durante años incluso décadas.

Mostramos que evidentemente si aumentamos el margen aumenta nuestra tasa, pero observamos que con márgenes de una década empiezan a obtenerse tasas de acierto aceptables que con las variables dadas para la predicción puede considerarse un logro.

- Margen de 5 años $\Rightarrow$ 0.26
- Margen de 1 década $\Rightarrow$ 0.49
- Margen de 15 años $\Rightarrow$ 0.69
- Margen de 2 décadas $\Rightarrow$ 0.8

4. Bibliografía

Fuentes de consulta on-line, en general documentación de las librerías:

1.scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

2. scikit-learn.org/stable/modules/generated/sklearn.modelselection.GridSearchCV.html

3. scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

4. towardsdatascience.com/cyclical-features-encoding-its-about-time-ce23581845ca

5. scikit-learn.org/stable/auto_examples/applications/plot_cyclical_feature_engineering.html

6. cienciadedatos.net/documentos/py22-deteccion-anomalias-isolation-forest-python

7. towardsdatascience.com/the-f1-score-bec2bbc38aa6f1 score

8. stackabuse.com/calculating-pearson-correlation-coefficient-in-python-with-numpy/

9. stackvidhya.com/plot-confusion-matrix-in-python-and-why/

10. jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html

11. scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html

12. machinelearningmastery.com/how-to-transform-target-for-regression-with-scikit-learn/