Memoria Práctica 3

Aprendizaje Automático

Christian Vigil Zamora
3º - Grupo 2
Mayo, 2019

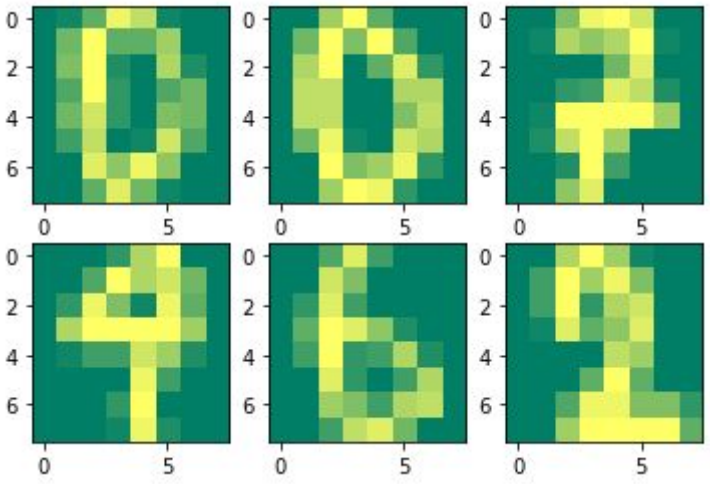
Índice

1. PROBLEMA DE CLASIFICACIÓN

1. Comprender el problema a resolver

Partimos de un dataset en el que se encuentra información con respecto a dígitos manuscritos numerados del 0 al 9. El dataset contiene 5620 instancias, las cuales ya vienen divididas en los conjuntos de entrenamiento y test, teniendo 3823 instancias de entrenamiento y sus respectivas etiquetas, y 1797 instancias de test con sus respectivas etiquetas. Por último comentar que el número de atributos es 64, los cuales toman valores enteros en el rango 0..16. Antes de hacer un visualizado de los datos, debemos tener en cuenta que lo que tenemos en cada fila del dataset representa una matriz 8x8, en la que cada atributo representa un elemento de ella. Una vez estudiado el dataset, llegamos a la conclusión de que nos encontramos ante un problema de Clasificación a resolver mediante Aprendizaje Supervisado.

En primer lugar, visualizamos el conjunto de datos de Entrenamiento y nos encontramos que lo representa cada fila es lo siguiente:

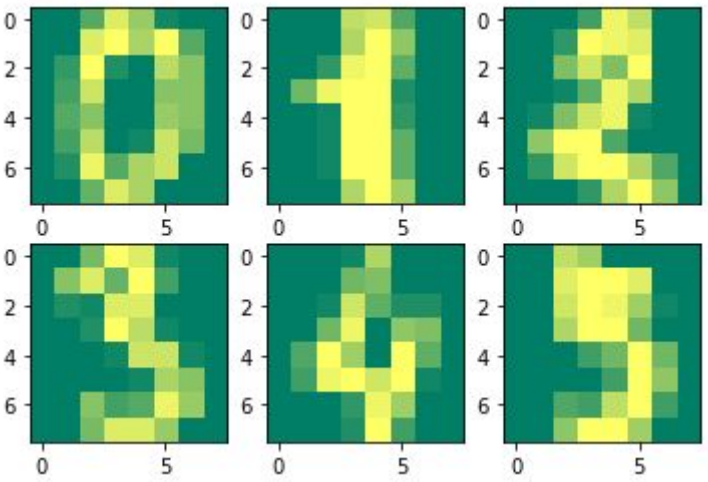


Es decir, corroboramos lo anteriormente comentado, cada instancia del conjunto de Entrenamiento se corresponde con un dígito manuscrito. En mi caso, he mostrado los 6 primeros dígitos del conjunto, también muestro las etiquetas de los 6 primeros datos para verificar que se corresponden:

```
-> Etiquetas correspondientes:

[0 0 7 4 6 2]
```

Seguidamente, repetimos el mismo proceso para el conjunto de datos de Test:



```
-> Etiquetas correspondientes:

[0 1 2 3 4 5]
```

Antes de pasar al Preprocesado de datos, debemos hacer una observación global a los conjuntos, por lo que he decidido comprobar cuál es la frecuencia de aparición de los dígitos en cada conjunto, es decir, se comprueba que tenemos unos conjuntos uniformes en los que el número de apariciones de cada dígito es parejo, para que el aprendizaje pueda ser llevado a cabo con normalidad.

Conjunto de Entrenamiento	**Conjunto de Test**
	

Como se puede observar, en ambos conjuntos, la frecuencia de aparición de los dígitos es muy pareja, con una representación del 10 % de cada dato, siendo el dígito 0, 2 y 8 los que presentan una menor aparición.

2. Preprocesado de los datos

Lo primero que se debe hacer en ambos conjuntos es eliminar aquellos datos que posean variabilidad muy baja. Para ello, recurro al método de Sklearn **'VarianceThreshold'** cuya función es eliminar aquellos datos con variabilidad menor a un umbral dado. En mi caso, he considerado como umbral 0.1, consiguiendo una reducción en ambos conjuntos de 64 atributos a 53. Como vemos, la reducción de dimensionalidad de los atributos no ha sido muy significativa, lo que nos da a entender que el dataset del que partíamos, presenta cierta calidad para el aprendizaje y no va a ser necesario llevar a cabo muchas técnicas de procesado.

A continuación, sabiendo que los atributos toman valores entre 0 y 16, debemos normalizar nuestros datos a valores entre 0 y 1 para evitar que los atributos con mayor magnitud, tengan más importancia en la posterior ejecución de los algoritmos. Para ello, hago uso de **'MinMaxScaler'**.

Por último, se aplica una estandarización a los datos con el objetivo de que lleguen en un formato mucho más digerible para según qué algoritmos. He recurrido a **'StandardScaler'** con el que los datos pasan a pertenecer a una distribución con una desviación estándar igual 1 y cuya media de la distribución es 0.

En un principio, me planteé reducir la dimensionalidad más aún mediante PCA, el cuál lleva a cabo la reducción basándose en la Descomposición Singular de los valores. Tras realizar pruebas y documentarme más, llegué a la conclusión de que dado el dataset inicial, no es usar PCA pues el dataset no presenta cantidad de atributos muy extensa, ni tampoco el número de instancias es muy elevado, por lo que aplicar PCA no tendría mucho sentido en éste caso.

Llegados a éste punto, ambos conjuntos están procesados y listos para seguir trabajando con ellos.

3. Selección de clases de funciones a usar

Para éste problema la clase de funciones seleccionada a usar ha sido lineales. Dado que estamos tratando con modelos lineales y es la clase más básica, se debería empezar por ella. A la vista de los resultados obtenidos, no ha sido necesario aumentar la clase de funciones. Es cierto, que al experimentar con clases de funciones cuadráticas en éste problema, el nivel de precisión que obtenían los modelos mejoraba ligeramente, pero el gasto computacional era bastante mayor, por lo tanto, como con la clase de funciones lineales los resultados son más que aceptables y nos permiten de igual forma conocer qué modelo de los comparados es mejor, definitivamente esa ha sido la elegida.

4. Selección de los conjuntos de training, validación y test

Con respecto a la selección de los conjuntos de training y test, no es necesario realizar ninguna selección pues el propio dataset como hemos comprobado nos ofrece los datos ya particionados. Lo único que debemos tener en cuenta es que aplicamos el mismo preprocesado a ambos conjuntos, y que los algoritmos van a entrenarse con el conjunto de training para poder ser evaluados luego con el de test. Respecto al conjunto de validación si hay tareas que realizar. Yo he optado por utilizar la técnica de Validación Cruzada para 5 particiones, es decir, el conjunto de datos de entrenamiento va a ser dividido en 5 particiones, de las cuáles 4 de ellas se van a utilizar como conjunto de entrenamiento y la partición restante como conjunto de test. Éste proceso será repetido K veces, en mi caso 5, y por tanto en cada iteración, se combinan de forma distinta las particiones y se obtiene un valor medio obtenido durante esa iteración. En mi caso, el método empleado para elegir los hyperparámetros, del cual hablaré en un apartado posterior, incluye la opción de realizar Validación Cruzada por tanto tan sólo tendría que indicarle el número de particiones que quiero que se hagan. He de decir que dicho método, al tratarse nuestro dataset de atributos números enteros, el proceso que selecciona para la Validación Cruzada es Stratified K-Folds, cuya peculiaridad es que el porcentaje de representación de cada etiqueta en las particiones a generar se mantendrá con respecto del conjunto original.

5. Discutir la necesidad de regularización y en su caso la función usada para ello

Evidentemente, la regularización va a ser necesaria en éste problema. Podríamos decir que es indispensable a la hora de evaluar diferentes modelos, pues debemos evitar ante todo el Overfitting, y usando éste método tratamos de conseguir evitarlo de forma eficiente. Es decir, nuestro objetivo es reducir el número de grados de libertad del modelo para así dificultar el hecho de que pueda producirse Overfitting, para ello se restringe la complejidad de los pesos del modelo. Y, ¿cómo se realiza eso? Muy fácil, añadiendo a la función de pérdida un coste asociado al uso de pesos. Para obtener ese coste, he considerado 2 opciones:

- **L1 (Regularización Lasso)** : El coste es proporcional al valor absoluto de los coeficientes de peso.

- **L2 (Regularización Ridge)** : El coste es proporcional al cuadrado de los coeficientes de peso.

6. Definir los modelos lineales a usar y estimar sus hyperparámetros

Los modelos lineales que he decidido usar son Regresión Logística y Perceptrón. Para estimar sus hyperparámetros, he usado el método **'GridSearchCV'**, el cuál a partir de unos posibles valores tomados como parámetros y un estimador, elige cuáles de esos posibles valores son los que obtienen un mejor resultado para ese estimador. Yo he decidido que el propio método utilice la técnica de Validación Cruzada para 5 particiones. Por último decir que dicho método, devuelve los mejores parámetros así como la puntuación media obtenida en la Validación Cruzada. A continuación paso a comentar aspectos sobre ellos:

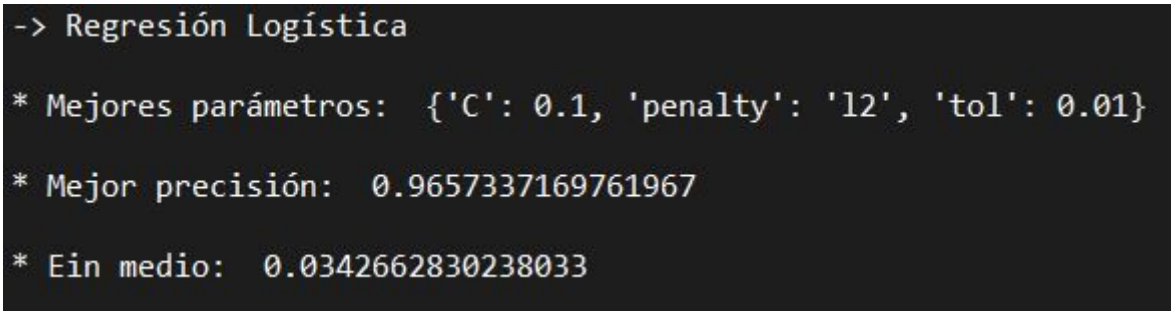
- **Regresión Logística**

En primer lugar, debo comentar que el algoritmo elegido para ser usado en la Regresión Logística ha sido **'newton-cg'.** El motivo ha sido porque nos encontramos ante un problema multi-clase, y dicho algoritmo es el que mejor funciona en éstos casos. Uno de los parámetros a estimar es la penalización, es decir, si usar **L1** o **L2**, comentadas en el apartado anterior, pero el hecho de haber elegido 'newton-cg' como algoritmo nos obliga a quedarnos exclusivamente con L2 pues no soporta otro tipo de penalización. Sí que vamos a poder estimar el valor del parámetro **'C'**, que es la inversa de la fuerza de Regularización y que contra menores valores tome, más dura será la Regularización. Por último, el último parámetro a estimar es **'tol'**, siendo la tolerancia para el criterio de parada.

Para saber sobre qué posibles valores estimar, he consultado el valor por defecto de ambos parámetros y he añadido valores cercanos a ese. La elección los posibles valores a estimar ha sido la siguiente:

[{**'penalty'**:['l2'], **'C'**:[0.0001, 0.001, 0.01, 0.1, 1, 10, 100], **'tol'**:[1e-2,1e-3,1e-4,1e-5]}]

Para éste modelo he obtenido:



- **Perceptrón**

Con el Perceptrón si tenemos la oportunidad de probar los distintos tipos de penalización, L1 y L2. También he considerado diferentes valores para el parámetro **'alpha'** que viene a ser una constante que multiplica el término de regularización, es decir, la penalización. Por último, al igual que en el modelo anterior, se va a estimar el valor óptimo de **'tol'**. Los posibles valores a estimar que he elegido son:

{**'penalty'**:['l1','l2'], **'alpha'**:[0.0001, 0.001, 0.01, 0.1, 1, 10, 100], **'tol'**:[1e-2,1e-3,1e-4,1e-5]}

Para éste modelo he obtenido:

```
-> Perceptron

* Mejores parámetros:  {'alpha': 0.0001, 'penalty': 'l1', 'tol': 0.001}

* Mejor precisión:  0.9492857281809265

* Ein medio:  0.050714271819073 51
```

7. Selección y ajuste del modelo final

El criterio para seleccionar cuál de los modelos evaluados va a ser elegido ha sido la métrica. Es decir, ambos algoritmos han sido ejecutados y evaluados probando diferentes parámetros, todo ello acompañado de una Validación Cruzada, por lo que tras la ejecución de ambos, se muestra la media obtenida en las ejecuciones de la Validación Cruzada. La métrica elegida para éste problema ha sido **'Accuracy'**, la que nos indica el porcentaje de acierto que ha obtenido la predicción de las etiquetas sobre el conjunto de etiquetas sin predecir. Es una métrica sencilla pero eficiente para éste problema. En el siguiente apartado valoraré su idoneidad.

Por lo tanto, si comparamos la precisión (resultado de 'Accuracy') obtenida con el algoritmo de Regresión Logística y con el Perceptrón, llegamos a la conclusión que el modelo que debemos seleccionar para realizar el ajuste final va a ser el obtenido con la Regresión Logística, pues es el que mayor precisión ha conseguido durante la Validación Cruzada.

Una vez seleccionado el mejor modelo, con sus respectivos parámetros, recurrimos al conjunto de Test para ajustarlo. Recuerdo que el conjunto de datos Test fue preprocesado de la misma forma que el conjunto de Entrenamiento, sino, sería imposible ajustar el modelo pues los datos con los que ha entrenado tendrían distinta dimensionalidad y distribución que los del conjunto de Test.

Llegados a aquí, realizamos una predicción de las etiquetas dado el conjunto de datos Test y obtenemos la precisión obtenida, es decir, la bondad de la predicción sobre las etiquetas del conjunto Test.

```
-> Modelo seleccionado: Regresión Logística

* Precisión obtenida sobre el conjunto Test 0.9488035614913745
```

8. Discutir la idoneidad de la métrica usada en el ajuste

La métrica 'Accuracy' para éste problema es idónea. Es una métrica muy simple e intuitiva que permite conocer la precisión que tienen los diferentes modelos a la hora de predecir las etiquetas. Como bien he indicado, es idónea para éste problema y ¿Por qué?. El motivo es muy sencillo, para que ésta métrica sea funcional y pueda servir para evaluar diferentes modelos, las clases del conjunto de datos sobre el que vamos a tratar deben estar equilibradas. Es por ello que al principio de la memoria, en la Visualización de los datos he incluído un gráfico de Sectores tanto para el conjunto de Entrenamiento como para el de Test y en ellos se puede observar que las clases de éste problema, están equilibradas, siendo cada una de ellas un 10% de la totalidad del conjunto y dato que tenemos 10 clases, queda más que confirmado el equilibrio.

Queda reflejado así, que si nuestros datos no tuvieran las clases equilibradas, no sería una métrica correcta, pues imaginemos que nuestro dataset está formado por 10 gatos y 90 perros, si siempre predijimos un pero, la precisión sería de 0.9, por lo que con total seguridad nuestro modelo va a predecir la clase perro siempre, sin importar cual sea la entrada.

9. Estimación del error Eout del modelo lo más ajustada posible

La estimación del error Eout del modelo ha sido:

```
-> Eout obtenido 0.051196438508625554
```

Como vemos, el error fuera de la muestra obtenido por el modelo elegido es bastante razonable, es decir, es un error que podemos asumir en problema de éste calibre. Si lo comparamos con el error Ein obtenido por el modelo sobre el conjunto de Entrenamiento (Ein = 0.034) vemos que el modelo se comporta de forma similar con datos que no conoce. Por tanto, nos indica a primera vista que el problema de evitar 'Overfitting' ha sido superado.

10. Discutir y justificar la calidad del modelo encontrado y las razones por las que considera que dicho modelo es un buen ajuste que representa adecuadamente los datos muestrales

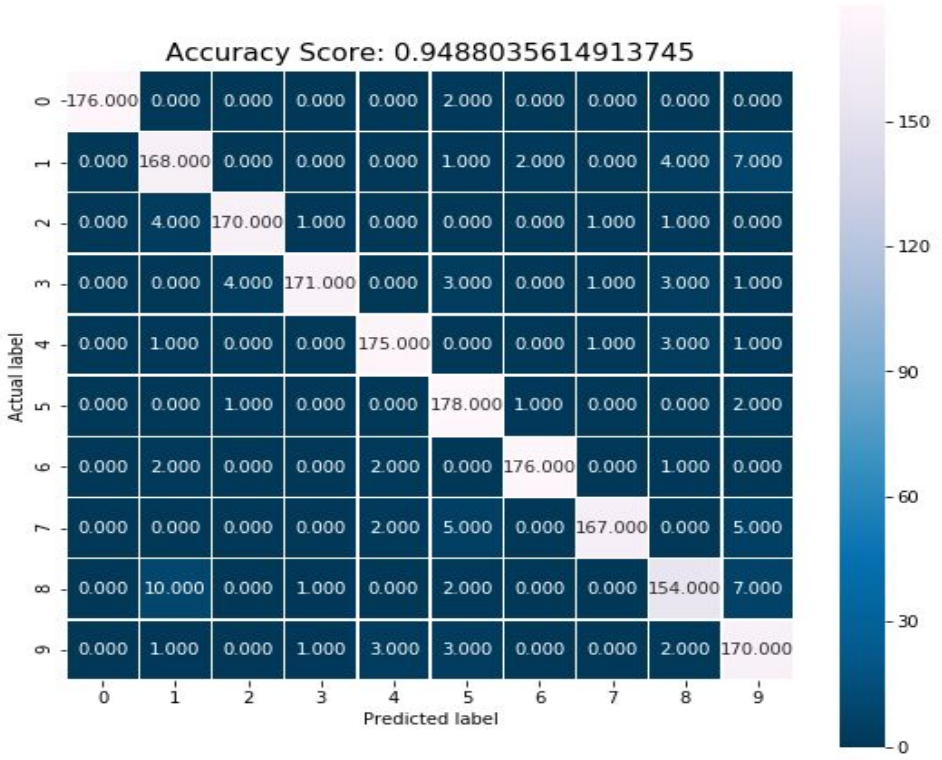
Considero que el modelo encontrado es un modelo de calidad pues a la vista de los resultados de precisión y error Eout, el modelo se comporta de forma similar tanto con datos que conoce como con datos que no. Los valores obtenidos son más que razonables, sí que es cierto que podrían haberse mejorado haciendo un preprocesado de datos más exhaustivo, pero ha sido evitado a fin de evitar coste computacional. A continuación muestro la cantidad de datos de cada dígito en el Conjunto de Test:

Tabla de Distribución:

DISTRIBUCIÓN DE CLASES EN EL CONJUNTO DE TEST	
Número de 0's	178
Número de 1's	182
Número de 2's	177
Número de 3's	183
Número de 4's	181
Número de 5's	182
Número de 6's	181
Número de 7's	179
Número de 8's	174
Número de 9's	180

El motivo por el que muestro ésta tabla es porque para razonar que es un buen ajuste y observar sus pros y sus contras, recurro a la Matriz de Confusión, técnica la cual nos permite observar con facilidad dónde ha fallado más nuestro modelo y nos permite comparar de forma muy intuitiva con los datos mostrados en la tabla.

Matriz de Confusión para el modelo:



Lo primero en lo que nos vamos a fijar es en la diagonal de la matriz, pues en ella se encuentran el número de datos que han sido clasificados correctamente. Como vemos, la diferencia de datos clasificados correctamente con respecto de los datos representados en la tabla anterior no varía en más de 10-12 datos por clase en general. Por lo tanto, podemos decir que el modelo es un buen ajuste que representa adecuadamente los datos muestrales. Como curiosidad, observamos en la matriz que el dígito 8 es el que más ha costado de predecir correctamente, pues ha sido predecido como dígito 1 en 10 ocasiones, como dígito 9 en 7 ocasiones y como dígito 3 en 1 ocasión. El hecho de que el dígito 8 fuese uno de los que menos representación tenía en el conjunto de datos de Entrenamiento ha podido ser uno de los factores de ello. Tras el dígito 8, los datos que con mayor frecuencia han sido mal clasificados han sido el 1 y el 7.

2. PROBLEMA DE REGRESIÓN

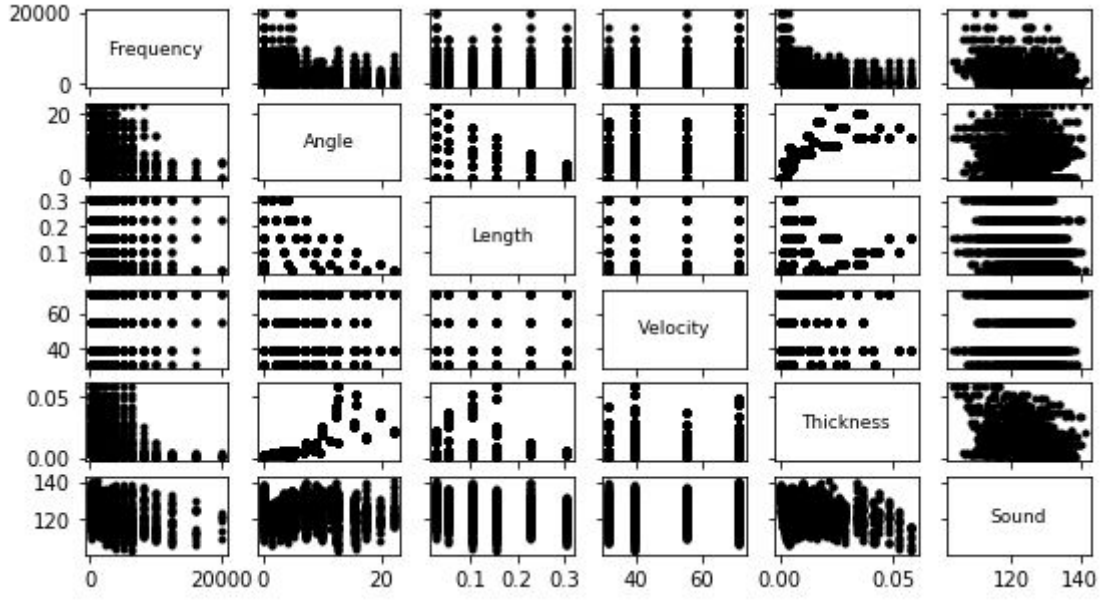
1. Comprender el problema a resolver

En ésta ocasión nos encontramos ante un dataset proporcionado por la NASA, el cuál comprende perfiles aerodinámicos NACA 0012 de diferentes tamaños a varias velocidades de túnel de viento y ángulos de ataque. La duración del perfil aerodinámico y la posición del observador fueron las mismas en todos los experimentos. Las entradas que tiene nuestro problema, es decir las características son 5:

1. Frecuencia, en hercios.
2. Ángulo de ataque, en grados.
3. Longitud de la cuerda, en metros.
4. Velocidad de flujo libre, en metros por segundo.
5. Espesor de desplazamiento lateral de aspiración, en metros.

Y la etiqueta representa el Nivel de presión sonora escalada, en decibelios.

Visualización de los datos:



Por lo tanto, estaríamos ante un dataset de 6 características formado por 1503 instancias, lo que a simple vista nos hacer ver que no tenemos un dataset muy amplio. También es necesario destacar que nos encontramos ante un dataset de datos contínuos, además de evidenciar que nos encontramos ante un problema

de Regresión como bien indica la etiqueta , ya que no sería posible clasificar éste dataset pues no hay clases.

2. Preprocesado de los datos

A diferencia del problema anterior, no nos encontramos ante un dataset ya particionado en conjunto de Entrenamiento y Test, sino que tenemos el dataset original, por lo que vamos a preprocesarlo y posteriormente, dividiremos en conjuntos. Lo primero que se debe hacer es eliminar aquellos datos que posean variabilidad muy baja. Para ello, recurro al método de Sklearn **'VarianceThreshold'** cuya función es eliminar aquellos datos con variabilidad menor a un umbral dado.
Tras realizar varias pruebas con diferentes valores de umbral, llegamos a la conclusión de que es absurdo utilizar éste método pues los datos no presentan variabilidad apenas, cosa que era de esperar en cierta medida, dada la corta extensión del dataset.

A continuación, debemos normalizar los datos. La normalización que mejores resultados me ha brindado ha sido la Transformación de Yeo-Johnson, la cuál consigue que los datos tengan una forma más Gaussiana y cuyo parámetro óptimo para estabilizar la varianza y minimizar la asimetría se estima con la máxima probabilidad.

Llegados a éste punto, nos planteamos la opción de reducir la dimensionalidad de los datos usando PCA. Pero si tenemos en cuenta que nuestro dataset apenas tiene 1500 instancias y sobre todo, sólo 5 atributos, pierde mucho sentido aplicar PCA puesto que nos va a reducir como mucho 1 o 2 atributos, y dado que no tenemos una gran cantidad de datos, puede ser no beneficioso. Además, PCA es un método usado cuando el número de instancias y de atributos es muy superior al de nuestro conjunto, por lo tanto, descartamos hacer uso de ello.

De ésta forma, ambos conjuntos están procesados y listos para seguir trabajando con ellos.

3. Selección de clases de funciones a usar

Sería razonable pensar que deberíamos comenzar por una clase de funciones lo más simple posible, de hecho es recomendable siempre empezar a probar con una clase que no sea muy compleja. En nuestro problema, salta a la vista que va a ser difícil producir un buen modelo usando la clase de funciones lineales dada la observación que hemos realizado anteriormente de los datos. A la mínima prueba que realizamos ajustando cualquier modelo nos damos cuenta de que los datos no están preparados para ser ajustados por modelos lineales, por lo tanto recurro a incrementar la clase de funciones a usar. ¿Cómo?. Con el método **PolynomialFeatures**, el cuál consiste en expandir los atributos calculando sus valores polinómicos. Es decir, si tenemos un conjunto de datos de la forma:

$$X = [x_0, x_1, x_2, x_3, \ldots, x_N]$$

El resultado de aplicar PolynomialFeatures con grado 2, sería:

$$X_{polinómica} = [x_0, x_1, x_1^2, x_2, x_2^2, \ldots, x_N, x_N^2]$$

Para nuestro caso concreto, el grado 3 ha sido el que mejores resultados ha obtenido y por tanto el elegido. Pero todo no puede ser tan bonito, ya que el hecho de incrementar la clase de funciones de ésta forma, aumenta la probabilidad de que pueda producirse 'Overfitting', por lo que habrá que asegurarse de comprobarlo posteriormente. De ésta forma, afirmamos que vamos a trabajar con clase de función polinómica, más en concreto, con clase de función cúbica.

4. Selección de los conjuntos de training, validación y test

Puesto que en ésta ocasión no tenemos los datos divididos en conjuntos, debemos realizarlo. Porello, voy a crear un conjunto de datos de Entrenamiento, cuyo tamaño va a ser el 80 % de los datos originales y un conjunto de Test, con tamaño del 20% con respecto de los datos originales. El motivo por el que hago ésta selección es debido a que el conjunto de Entrenamiento lo voy a usar tanto para entrenar los modelos como para validarlos, por medio de la Validación Cruzada, por tanto no me es necesario seleccionar un conjunto de Validación, ya que la propia Validación Cruzada lo hace por sí misma. Por tanto, el conjunto de Test creado, servirá para ajustar el modelo final y comprobar cuánto de bueno es el modelo que hemos elegido.

5. Discutir la necesidad de regularización y en su caso la función usada para ello

Al igual que en problema anterior, la regularización es imprescindible puesto que debemos evitar ante todo el Overfitting. Es decir, nuestro objetivo es reducir el número de grados de libertad del modelo para así dificultar el hecho de que pueda producirse Overfitting, para ello se restringe la complejidad de los pesos del modelo. El hecho de hacer hincapié en la Regularización es porque de nada nos serviría elegir un modelo que ajusta muy bien los datos del conjunto de Entrenamiento, pero que su rendimiento con datos que no conoce es mucho peor. Además, como hemos expandido los atributos a polinomios cúbicos, tenemos más riesgo se sufrir Overfitting, por lo tanto, son más que de sobra los

motivos para usar regularización. Vamos a considerar las mismas técnicas de regularización que para el ejercicio anterior:

- **L1 (Regularización Lasso)** : El coste es proporcional al valor absoluto de los coeficientes de peso.

- **L2 (Regularización Ridge)** : El coste es proporcional al cuadrado de los coeficientes de peso.

6. Definir los modelos lineales a usar y estimar sus hyperparámetros

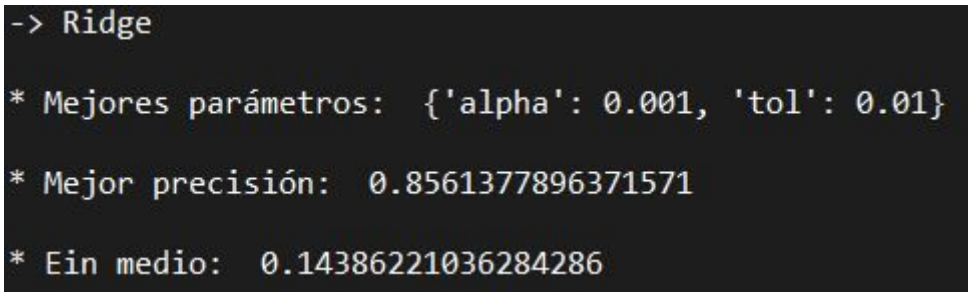
Los modelos lineales que he decidido usar son Ridge, Lasso y Regresión Lineal. Para estimar sus hyperparámetros, he usado el método **'GridSearchCV'**, con Validación Cruzada en 5 particiones, es decir, misma descripción que para el problema anterior. Modelos evaluados:

- **Ridge**

En éste modelo, el primer parámetro que he considerado ha sido **'alpha'**, el cuál indica la dureza de la Regularización. Debo matizar que en éste modelo, sólo es posible llevar a cabo la Regularización L2. Y por último, he considerado el parámetro **'tol'**, para ajustar la precisión de la solución. Por tanto, para saber sobre qué posibles valores estimar, he consultado el valor por defecto de ambos parámetros y he añadido valores cercanos a ese. La elección los posibles valores a estimar ha sido la siguiente:

[{**'alpha'**:[0.001, 0.01, 0.1, 1, 10, 100], **'tol'**:[1e-2,1e-3,1e-4,1e-5]}]

Para éste modelo he obtenido:



Como vemos, la dureza de la Regularización para la que mejor resultado ha obtenido no es muy alta, por lo tanto sabemos que la regularización que ha llevado a cabo no ha sido muy dura.

- **Lasso**

En éste modelo, pasa justo lo contrario que en el anterior. La única Regularización permitida, es L1. La elección de parámetros ha sido la misma, **'alpha'**, el valor de la constante que multiplica el término de Regularización L1 y **'tol'**, en éste modelo, la tolerancia para la optimización. Los posibles valores a estimar que he elegido son:

[{**'alpha'**:[0.001, 0.01, 0.1, 1, 10, 100], **'tol'**:[1e-2,1e-3,1e-4,1e-5]}]

Para éste modelo he obtenido:

```
-> Lasso

* Mejores parámetros:  {'alpha': 0.001, 'tol': 1e-05}

* Mejor precisión:  0.8539807259146603

* Ein medio:  0.14601927408533966
```

De nuevo, para éste modelo el valor de alpha sigue sin ser duro.

- **Regresión Lineal**

He decidido hacer uso también del modelo de Regresión Lineal básico, en el que el único parámetro cuyos valores puedo evaluar es **'normalize'**, con el que se evaluará si llevar a cabo o no la normalización. Posibles valores a estimar:

[{**'normalize'**:['True','False','optional']}]

Para éste modelo he obtenido:

```
-> Linear Regression

* Mejores parámetros:  {'normalize': 'True'}

* Mejor precisión:  0.8561291088318225

* Ein medio:  0.14387089116817753
```

7. Selección y ajuste del modelo final

El criterio para seleccionar cuál de los modelos evaluados va a ser elegido ha sido la métrica. Es decir, todos los algoritmos han sido ejecutados y evaluados probando diferentes parámetros, todo ello acompañado de una Validación Cruzada, por lo que tras la ejecución de ambos, se muestra la media obtenida en las ejecuciones de la Validación Cruzada. La métrica elegida para éste problema ha sido **'r2'**, también conocida como Coeficiente de Determinación, la cual mide la proporción de **Y** (etiquetas) que es explicada por el modelo. Dicha métrica, contra más se acerque el valor a 1 que obtiene, mejor será el modelo, por lo tanto ese será nuestro criterio. En el siguiente apartado daré más detalles de ésta métrica.

Por lo tanto, al realizar la comparación entre los 3 modelos validados anteriormente, nos damos cuenta de que el modelo con mayor Coeficiente de Determinación durante la Validación Cruzada ha sido el Ridge, aunque es cierto que la diferencia con respecto a los demás no es muy significativa. De ésta forma, concluímos y seleccionamos nuestro mejor modelo.

Una vez seleccionado el mejor modelo, con sus respectivos parámetros, recurrimos al conjunto de Test para ajustarlo, que de hecho sabemos ya que está bien preprocesado, pues la división en conjuntos se realizó después del preprocesado de datos. Recuerdo también que los modelos no han sido validados con el conjunto de Test, sino mediante Validación Cruzada sobre el conjunto de Entrenamiento, ya que sino el aprendizaje sería nulo.

Llegados a éste punto, realizamos una predicción de las etiquetas dado el conjunto de datos Test y obtenemos el valor del Coeficiente de Determinación:

```
-> Modelo Seleccionado: Ridge

* Coeficiente de Determinación obtenido sobre el conjunto Test:  0.8401097243054624
```

8. Discutir la idoneidad de la métrica usada en el ajuste

Primero voy a comentar por qué he descartado ciertas métricas. La primera descartada ha sido MSE (Error Medio Cuadrático) , si bien es la más simple, no nos permite evaluar de forma correcta los modelos puesto que con que hagamos una predicción bastante mala, el hecho de ser cuadrático hace que el error sea aún peor y por tanto nos haría pensar que el modelo es más malo de lo que pueda serlo en realidad. Otra métrica descartada ha sido MAE (Error Absoluto Medio), que calcula la media de la diferencia absoluta entre los puntos de datos reales y la

predicción. Es cierto que MAE es una métrica mucho mejor interpretable que MSE, pero ha sido descartada por la métrica **Coeficiente de Determinación** ó r^2.

Dicha métrica es la elegida y la usada como criterio, ¿el motivo?, su intuición y su facilidad para ser interpretada. El Coeficiente de Determinación podría entenderse como una versión estandarizada del MSE, que representa la varianza de los valores de Y, explicada por el modelo. Su desglose podría verse como:

$$\text{Coeficiente de Determinación} \rightarrow \quad R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$$

$$\text{Suma de Cuadrados Total} \rightarrow \quad SST = \sum (y - \bar{y})^2$$

$$\text{Regresión de Suma de Cuadrados} \rightarrow \quad SSR = \sum (y' - \bar{y'})^2$$

$$\text{Error de Suma de Cuadrados} \rightarrow \quad SSE = \sum (y - y')^2$$

La forma de comprender lo que nos quiere decir la métrica es que conforme más se acerque el valor devuelto a 1, el modelo será mejor, mientras que conforme se vaya acercando a 0 (o a valores negativos, que puede tomarlos), será peor. Además, otra de sus ventajas es que es independiente de la escala de Y.

9. Estimación del error Eout del modelo lo más ajustada posible

La estimación del error Eout del modelo ha sido:

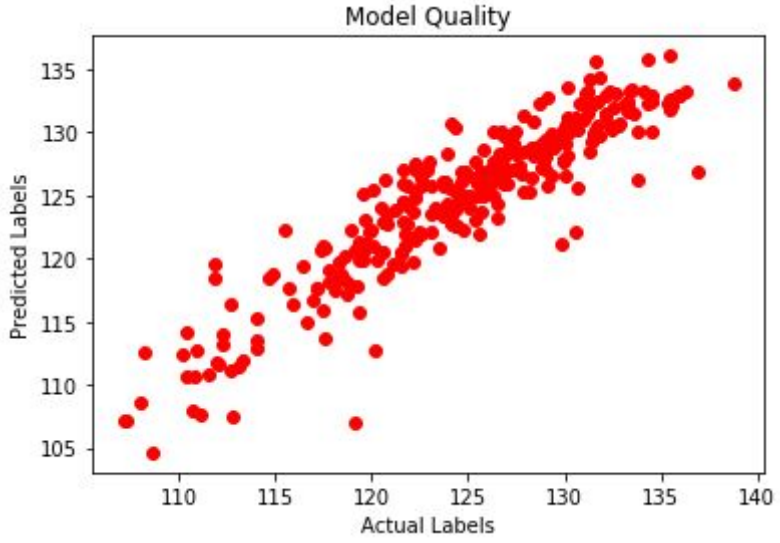
```
-> Eout obtenido:  0.15989027569453762
```

Como vemos, el valor de Eout es aceptable. Evidentemente hay modelos que podrían reducir ese error considerablemente, pero contando sólo con los modelos que hemos validado, es un buen valor de error y más si lo comparamos con el valor de Ein obtenido en el conjunto de Entrenamiento (Ein = 0.1438). Ambos valores son muy próximos los que nos hace ver que el modelo ajuste prácticamente de igual forma tanto a los datos de Entrenamiento como a los datos que no conoce, en éste caso, los datos de Test. Además, la proximidad de ambos errores nos indica ciertos aspectos positivos que serán comentados en el apartado de a continuación.

10. Discutir y justificar la calidad del modelo encontrado y las razones por las que considera que dicho modelo es un buen ajuste que representa adecuadamente los datos muestrales

Llegados a éste punto, considero que el modelo encontrado es un buen ajuste que representa adecuadamente los datos muestrales. En primer lugar, la clase de funciones elegida ha sido un acierto, pues ha mejorado considerablemente los resultados de la predicción. En segundo lugar, el principal riesgo que corríamos al incrementar hacia clases polinómicas era la aparición de 'Overfitting', y ya podemos afirmar que hemos conseguido evitarlo. Como hemos visto, el modelo encontrado realiza un ajuste del conjunto de Test muy similar al conjunto de Entrenamiento, los que nos indica que no se ha producido 'Overfitting' durante el aprendizaje, pues si hubiera ocurrido, la calidad del ajuste del modelo sobre el conjunto Test hubiera descendido considerablemente, y tal y como hemos visto en apartados anteriores, no ha sido así.

La mejor forma de justificar la calidad de dicho modelo sobre los datos muestrales es mediante una representación, es por ello que adjunto la siguiente:



En la gráfica, han sido representadas las etiquetas conocidas del conjunto de Test en el eje X y las etiquetas predichas por el modelo en el eje Y. La forma de interpretar dicha gráfica es la siguiente: Si nuestro modelo ha ajustado con cierta calidad y es representativo de los datos muestrales, debería apreciarse una recta en la que ambas etiquetas coincidieran en la mayoría de los puntos, de forma que se viera una recta sólida y sin mucha dispersión. En caso contrario, los puntos correspondientes a las etiquetas tanto del conjunto de Test como predichas no coincidirían y la apreciación de una recta sólida sería mucho más complicada. En nuestro caso, a la vista de la gráfica, podemos apreciar que gran cantidad de las etiquetas coinciden, además de observarse a la perfección el hiperplano, cerrando así la conclusión de que el modelo encontrado presenta calidad y representa adecuadamente a los datos muestrales.

3. REFERENCIAS BIBLIOGRÁFICAS

- https://scikit-learn.org/stable/modules/model_evaluation.html#scoring-parameter
- https://scikit-learn.org/stable/
- https://iartificial.net/regresion-polinomica-en-python-con-scikit-learn/
- https://medium.com/
- https://towardsdatascience.com/
- http://sitiobigdata.com/index.php/2019/01/19/machine-learning-metrica-clasificacion-parte-3/