Aprendizaje Automático

Jorge Gangoso Klöck 49398653N
Práctica 3 - Entrega: 04/06/2021

Problema 1 - Superconductivity Data Set

En este apartado partimos de dos ficheros de datos. Ambos poseen información relevante sobre un conjunto de superconductores, siendo uno de ellos una descripción detallada de su fórmula química y el otro una aglomeración de características medidas sobre estos mismos compuestos.

El objetivo es predecir, partiendo de las características dadas, una de las características más relevantes de los superconductores: la temperatura crítica.
En el conjunto de datos se localiza en la última columna de cada instancia y la usaremos como etiqueta para los sets de entrenamiento y test.

De esta manera, podemos determinar X como el set de características empíricas sobre los superconductores, f como la función que relacione de la forma más acertada posible la temperatura crítica que tendrá un superconductor en base a sus otras características, e Y las temperaturas críticas asociadas a los superconductores.

Para el proceso de Regresión Lineal dispondremos de distintas herramientas para realizar el fit, las cuales evaluaremos y discutiremos más adelante.

No es difícil imaginarse que a pesar de contar con una cantidad sustancial de instancias en el DataSet (21.263 concretamente), realizar un ajuste de regresión lineal aplicando las 81 características puede desembocar en un overfitting excesivo y, en consecuencia, ser antiproducente a la hora de crear un modelo de predicción acertado fuera de la muestra.

Para evitar esta situación vamos a llevar a cabo un proceso de selección de las variables más relevantes para el modelo. Vamos a comenzar explicando las posibilidades que tenemos en este paso y cual se ha escogido, pero primero haré una breve descripción del Error que utilizaremos para valorar nuestras decisiones.

Métrica de Error

Como breve discusión sobre el error empleado para valorar los ajustes, partiremos de la base de que al estar tratando en un ámbito de números reales y puesto que nuestra tarea consiste en un ejercicio de predicción, los errores razonables a usar son:

-Error Absoluto Medio (EM): $\left[\left|\frac{\Sigma_{i=1}^{n} h_i(x) - y_i}{N}\right|\right]$. Calcula la distancia entre nuestra predicción y el valor real. Su medida es igual a la de la variable que se está calculando lo que lo vuelve una métrica interesante. Sin embargo, al valorar todos los errores por igual permite errores de mayor valor aunque sean en menor cantidad, y por el tipo de problema al que nos enfrentamos me parece más interesante tratar de dar valores más cercanos al real aunque haya más puntos desajustados en general.

-Error Cuadrático Medio (MSE): $\left[\frac{\Sigma_{i=1}^{n} (h_i(x) - y_i)^2}{N}\right]$. La distancia entre nuestra predicción y el valor real al cuadrado. Es muy semejante al EM pero está más relacionado con el método de mínimos cuadrados que empleamos durante los modelos. El principal problema de esta medida es que al no coincidir en métrica con la variable objetivo el resultado es puramente comparativo con otras medidas, y no se puede llevar al ámbito de estudio real.

-Raíz del Error Cuadrático Medio (RMSE): $\left[\sqrt{\frac{\Sigma_{i=1}^{n} (h_i(x) - y_i)^2}{N}}\right]$. El cálculo es el mismo que para el MSE pero al resultado se le aplica una raíz cuadrada. Esto permite al igual que con el EM que la medida del error se encuentre en las mismas unidades que la variable objetivo, pudiendo así llevar al ámbito práctico el error obtenido. En caso de que fuera a utilizarse nuestro modelo de regresión para cualquier tipo de fabricación por ejemplo, bastaría con utilizar nuestro RMSE para dar el intervalo de confianza en el que se dan nuestras medidas. Y puesto que se trata de error cuadrático, tenemos mayor certeza que con el EM de que las medidas estarán dentro de ese intervalo y que no habrá predicciones demasiado alejadas del valor real, ya que son penalizados enormemente por el cuadrado.
Por las razones presentadas anteriormente, esta es la medida que vamos a emplear para nuestro modelo.

Reducción de Dimensionalidad: eliminación de características poco relevantes

PCA (Principal Component Analysis)

Una de las herramientas para reducir la dimensionalidad del problema y mantener un porcentaje de varianza explicada conjunta elevado consiste en aplicar PCA.

Este proceso creará unas variables llamadas componentes principales que estarán compuestas por una combinación lineal de las características originales, siendo linealmente independientes entre sí, y, en consecuencia, asegurando un mejor ajuste.

Los modelos de regresión pueden aplicarse entonces directamente sobre estas componentes principales para lograr, como se comentó anteriormente, ajustes realmente certeros.
Sin embargo, el resultado de los ajustes realizados de esta manera son complejos de interpretar y no permiten recibir toda la información sobre la relación entre la hipótesis final y las características originales, y, dado que nuestro objetivo en la práctica es entender mejor el problema al que nos enfrentamos no vamos a optar por este proceso.

Feature Extraction

El principal proceso involucrado en la extracción de features consiste en la aglomeración de variables relacionadas en variables únicas que recojan la información de las variables que lo forman.

La Feature Extraction precisa de un entendimiento elevado del significado de las variables que se están manejando, y, dado que la materia de estudio de este problema está bastante alejado de las competencias que poseo no va a ser tampoco utilizado en el proceso.

Feature Selection

Finalmente tenemos Feature Selection que es el procedimiento que vamos a seguir para reducir la dimensionalidad de nuestro problema. Consiste en utilizar diferentes medidas estadísticas para seleccionar solamente aquellas variables que determinan de forma notoria la variable objetivo T_C.

En primer lugar descartamos las características que sean prácticamente constantes para todos los datos del DataSet, es decir aquellos con una varianza muy baja.
Para ello, realizamos una normalización simple (Dividir los datos de cada característica entre la media de la misma) para tener un criterio de descarte justo.
Aplicamos la función VarianceThreshold(0.05) y un fit() para descartar aquellas variables que tengan un valor de varianza inferior a 0.05 (5% tras la normalización).

Un total de 7 características son descartadas mediante este proceso.

El siguiente paso es eliminar las características que no presenten correlación apenas con la variable T_C. (ϱ<0.1) y aquellas que estén fuertemente correlacionadas entre sí. (ϱ>0.9).
[Los valores decididos para determinar alta correlación están extraídos del siguiente documento: https://condor.depaul.edu/sjost/it223/documents/correlation.htm]

Para este paso calculamos la matriz de correlación y extraemos los que corresponden según nuestro criterio de reducción.

Matriz de Correlación respecto a T_c (Ordenado)

	index	critical_temp
0	gmean_fie	-0.026469
1	entropy_ThermalConductivity	0.079185
2	wtd_gmean_ElectronAffinity	-0.100189
3	mean_fie	0.102657
4	mean_atomic_radius	0.108276
5	mean_atomic_mass	-0.114270
6	std_Density	0.115413
7	wtd_mean_ElectronAffinity	0.117730
8	wtd_entropy_ThermalConductivity	-0.123615
9	range_FusionHeat	-0.142632
10	gmean_atomic_radius	-0.143195

Matriz de Correlación de todas las características

	number_of_elements	mean_atomic_mass	wtd_mean_atomic_mass	wtd_entropy_atomic_mass	range_atomic_mass	w
number_of_elements	1.000000	-0.138974	-0.351490	0.881760	0.681458	
mean_atomic_mass	-0.138974	1.000000	0.814063	-0.097340	0.127643	
wtd_mean_atomic_mass	-0.351490	0.814063	1.000000	-0.414340	-0.139991	
wtd_entropy_atomic_mass	0.881760	-0.097340	-0.414340	1.000000	0.621205	
range_atomic_mass	0.681458	0.127643	-0.139991	0.621205	1.000000	
wtd_range_atomic_mass	-0.324145	0.451365	0.723701	-0.547678	-0.107388	
mean_fie	0.163159	-0.288492	-0.207524	0.122362	0.210697	
wtd_mean_fie	0.480013	-0.224164	-0.524747	0.528654	0.436037	
wtd_entropy_fie	0.718881	-0.164104	-0.128894	0.696533	0.542463	
range_fie	0.779021	-0.255697	-0.452497	0.743460	0.646083	
wtd_range_fie	0.324371	-0.075695	-0.417773	0.322205	0.270831	
mean_atomic_radius	-0.001767	0.499734	0.285230	0.084614	-0.001304	
wtd_mean_atomic_radius	-0.420393	0.379815	0.662289	-0.470652	-0.359897	
wtd_range_atomic_radius	-0.374127	0.150775	0.374978	-0.575374	-0.348838	
mean_Density	-0.417713	0.758233	0.751670	-0.397558	-0.159889	
wtd_mean_Density	-0.505527	0.608885	0.844712	-0.550259	-0.298346	
gmean_Density	-0.627773	0.597011	0.713379	-0.595685	-0.480905	

Reducción de Dimensionalidad: selección empírica de las características a modelar.

Una vez finalizado el paso anterior y dependiendo de los valores de ϱ escogidos nos encontraremos con una matriz de m características (39 en nuestro ejemplo).

Sin embargo, no tenemos garantía de que utilizar las m características nos aporte el mejor resultado posible. Por ello, procedemos a utilizar tres modelos de regresión lineal con una cantidad de características que va incrementándose desde 1 hasta m.

La gráfica resultante de este estudio nos demostrará cuál es la cantidad mínima de características que ofrece una mejora sustancial a nuestra función de pérdida (Error Cuadrático Medio o RMSE) y también cual maximiza el coeficiente de determinación (R^2)

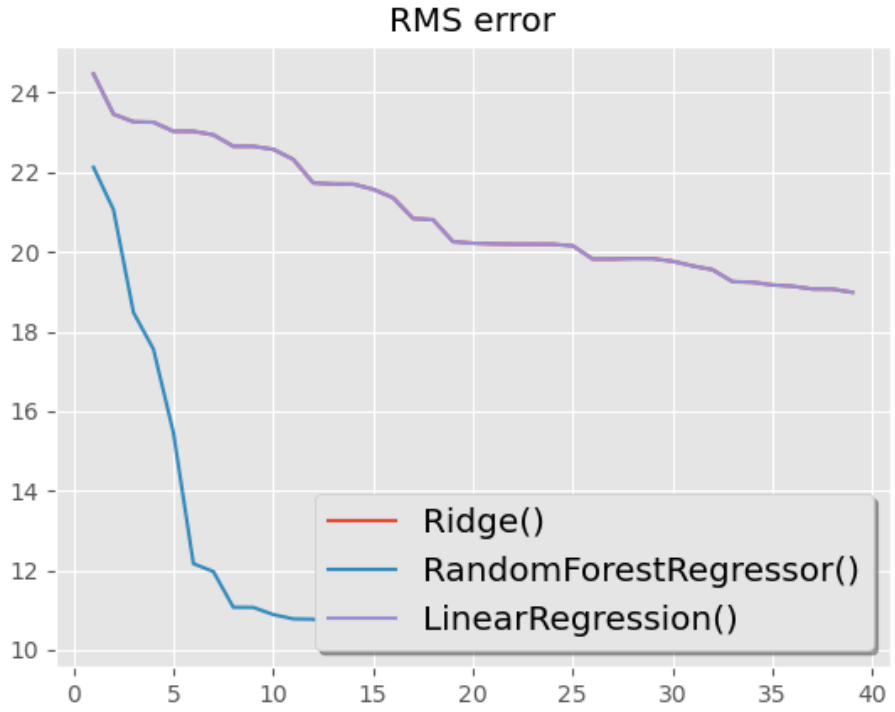
Los modelos empleados son:

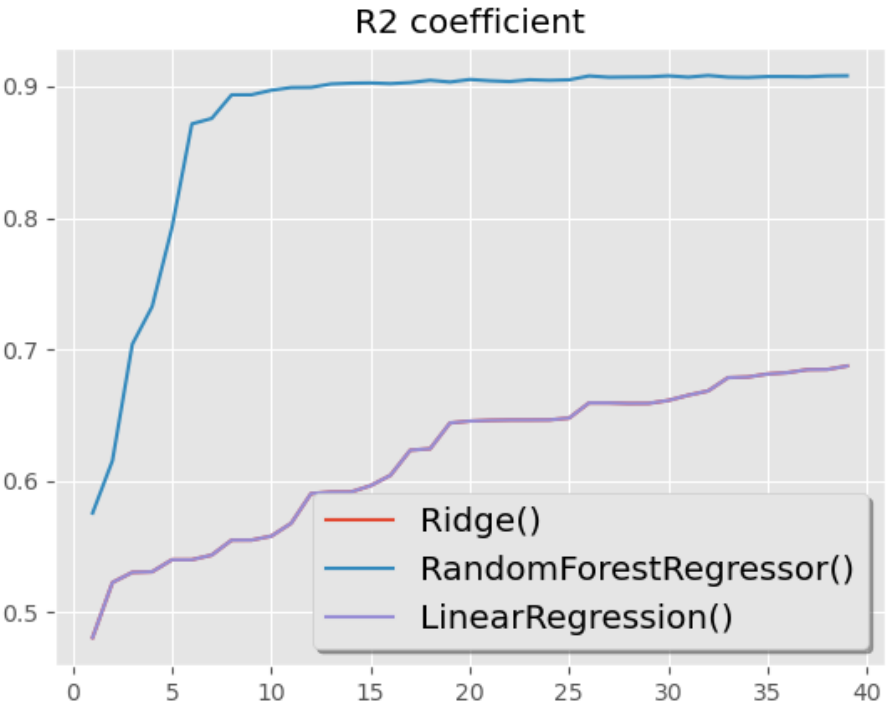
-Linear Regression: Función de sklearn que implementa una versión sencilla del ajuste por mínimos cuadrados.

-Ridge: Función de sklearn que implementa mínimos cuadrados pero aplicando una regularización que penaliza los pesos con valores lejanos a 0.

-RandomForestRegressor: Función de sklearn que implementa una regresión mediante árboles de decisión.

Si observamos las gráficas obtenidas, podemos ver que la cantidad de parámetros que minimiza nuestro error de forma considerable con la mínima complejidad posible (Navaja de Ockham) es de 8 y mediante el modelo de RandomForest. Podemos observar como el método de mínimos cuadrados y el Ridge presentan un rendimiento tan parecido que apenas hay diferencia.
[Nota: estas gráficas han sido generadas usando RandomForestRegressor() con los parámetros base. En el código aportado se modifican los parámetros para reducir el tiempo de cómputo y no ralentizar la corrección, ya que el resultado es muy semejante]





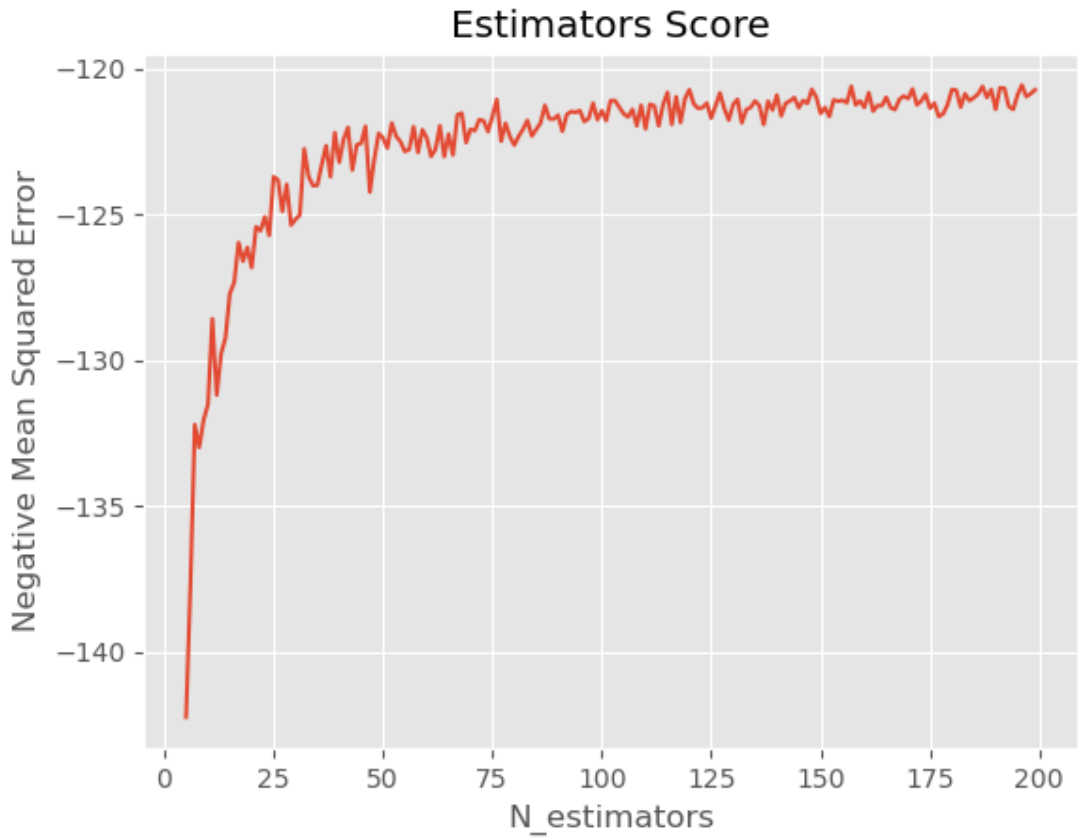
Sabiendo esto podemos comenzar a ajustar nuestro modelo de RandomForest con las 8 primeras variables de nuestra matriz de correlación con la T_C.

Ajuste de RandomForest

RandomForestRegressor() cuenta con una multitud de parámetros que se pueden ajustar. Los que más nos interesan de entre ellos son aquellos que nos permiten alterar la cantidad de árboles que generamos (n_estimators).
Esta variable nos permitirá acelerar el proceso de ajuste a cambio de un posible descenso en la precisión del resultado. Sin embargo será útil modificarlo para determinar la cantidad óptima de categorías a utilizar para el ajuste, y más adelante utilizaremos validación cruzada para obtener el valor que maximice la bondad del ajuste.

N_estimators

La cantidad adecuada de estimators (árboles del bosque) la comprobaremos usando cross-validation con una cantidad de Folds=5 que es un valor empíricamente demostrado que otorga buenos resultados. En cada iteración se utilizan ⅘ del set para entrenar y ⅕ para validar, esto se repite 5 veces.
La gráfica resultante del proceso es la siguiente, en la que representamos la cantidad de estimators frente al E_{CV}, y, de ella, obtenemos la cantidad idónea de estimadores para nuestro problema.



Vemos cómo a partir de 50 ya no mejora apenas, y tenemos un pico en 75. Podemos usar el valor de 50 para los cálculos posteriores pesados y el de 75 para el cálculo de la hipótesis final.

Criterion

El criterio de valoración. Como se comenta en el apartado de métrica de Error, utilizaremos MSE (default) a falta de la posibilidad de usar RMSE ya que nos interesa obtener resultados dentro de un intervalo de confianza pequeño con una buena estimación de probabilidad.

Max_features

Cantidad orientativa (no impone una restricción estricta) de las características a tener en cuenta para elegir el mejor split. Las opciones principales son “auto” que mantiene la cantidad total de características, “log” que tiene en cuenta $\log_2$(n_características) y “sqrt” que tiene en cuenta sqrt(n_características).
También estudiado el caso con CV(K_Fold = 5) tenemos que el mejor parámetro es sqrt presentando una mejora de aproximadamente un 3% en el MSE con respecto a las otras dos opciones.

Auto	Log_2	Sqrt
-132.61994827929718	-131.9984167679482	-129.4912122838111

Otros

Los demás parámetros no queremos ajustarlos ya que sirven para limitar los recursos utilizados por el modelo. Puesto que en esta etapa lo que nos interesa es utilizar los parámetros que nos permitan el ajuste lo más acertado posible, sin importar tanto el coste computacional, se mantienen en su valor defecto que es el de máximo rendimiento.

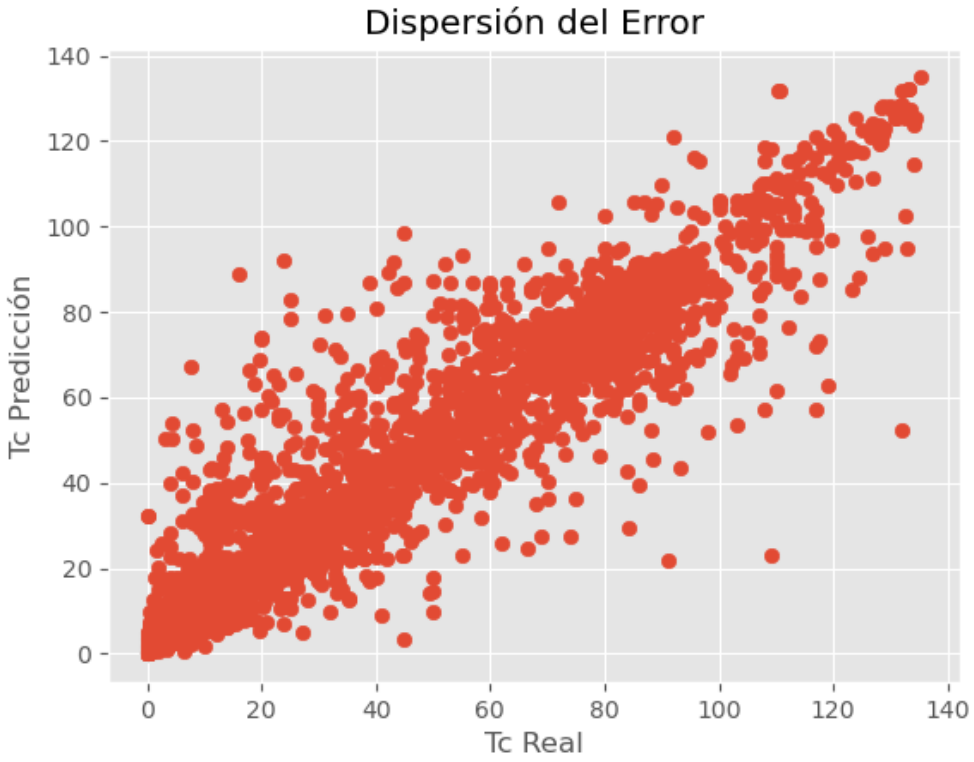
Entrenamiento y E_{OUT}

Decididos los parámetros podemos proceder a entrenar nuestro modelo con toda la partición X_{train} que generamos inicialmente (70% de los datos totales, 80/20 está demostrado empíricamente que aporta resultados sustanciales [Pareto Principle] pero debido a la alta intensidad computacional que supone el RandomForestResgressor() y a la cantidad sustancial de ejemplos en el DataSet he decidido rebajarlo a 70/30)

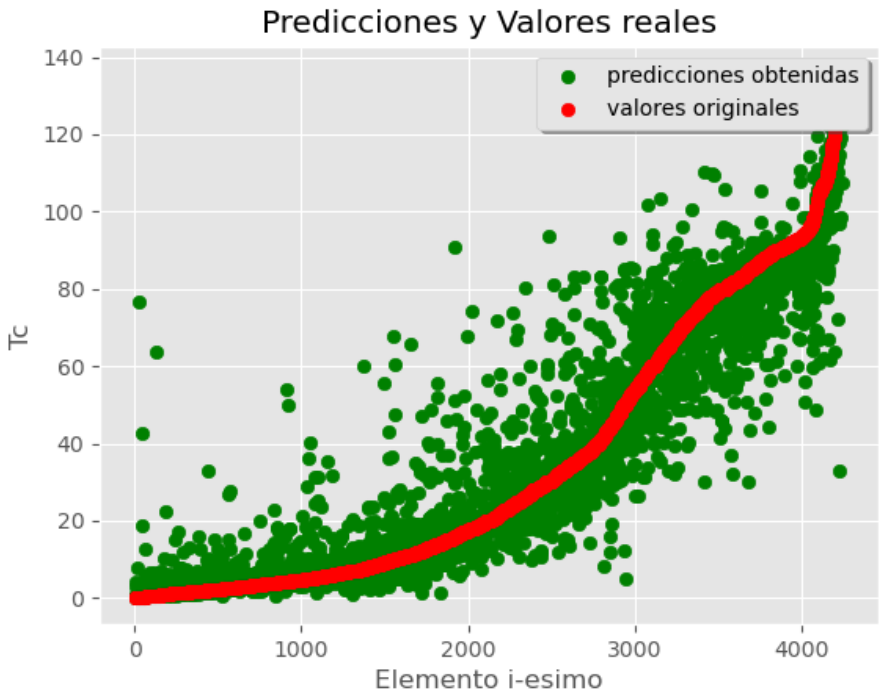
Puesto que el modelo empleado es RandomForest no vamos a obtener un vector de pesos asociados a cada característica. Por ello, procedemos directamente a calcular nuestro E_{OUT} empleando los datos de X_{test}.

Los resultados son bastante satisfactorios y pueden variar ligeramente entre ejecuciones debido al factor aleatorio en el modelo:

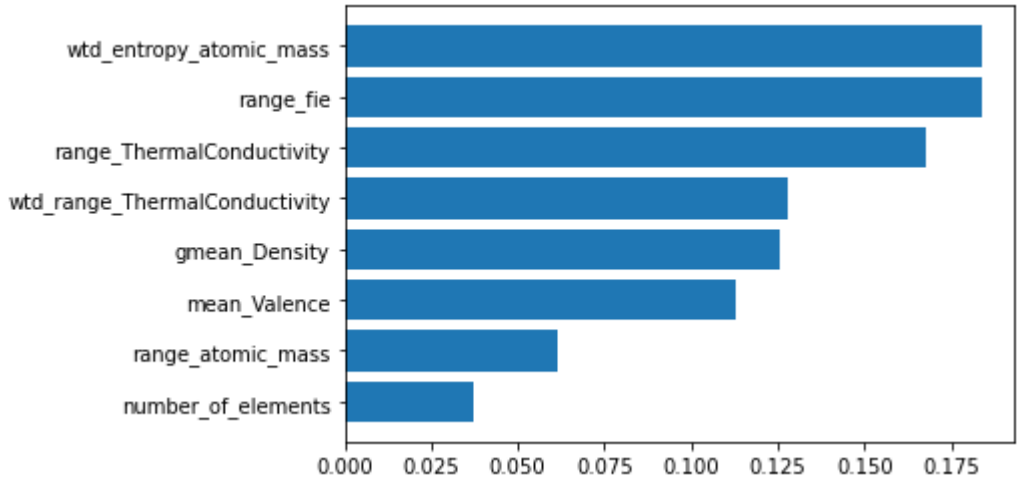
E_{OUT}: $\pm\ 10K$



Nuestro valor de E_{OUT} con esta dispersión de error es indicativo de un modelo preciso que realizará predicciones en un intervalo de confianza muy cercano a lo que serán los valores reales.



Podemos ver cómo las predicciones siguen la pista de los valores reales de T_C con un error asumiblemente pequeño.



Aquí podemos apreciar la importancia que da nuestro modelo a cada una de las variables que lo conforman de mayor a menor.

Este valor de $E_{OUT}(g^-)$ es una cota superior a lo que será el $E_{OUT}(g)$ del modelo entrenado con todos los datos.
El valor de $E_{OUT}(g)$ no podemos calcularlo por su propia definición al utilizar todo el DataSet para entrenarse, pero podemos hacer una aproximación.
Calculamos el valor de $E_{IN}(g)$ = 5.305
El valor esperado de E_{IN} se calcula como sigue:

$$\left[E_{IN}\right] = \sigma^2\left(1 - \frac{d+1}{N}\right) = 5.305$$

$$\sigma^2 = \frac{5.305}{\left(1-\frac{d+1}{N}\right)} = \frac{5.305}{\left(1-\frac{8+1}{21263}\right)} = 5.307$$

Teniendo sigma cuadrado podemos obtener el valor esperado de E_{OUT} con su propia ecuación.

$$\left[E_{OUT}\right] = \sigma^2\left(1 + \frac{d+1}{N}\right) = 5.307\left(1 + \frac{8+1}{21263}\right) = 5.309$$

Y ya tenemos el valor esperado de nuestro E_{OUT}(g). La variación es muy pequeña debido a la inmensa cantidad de elementos que conforman el DataSet. Este error por supuesto es sólo una estimación optimista, ya que puede empeorar debido a cualquier error por ruido o overfitting que hayamos cometido, pero al menos tenemos una estimación de lo bueno que es nuestro modelo.

Aclaraciones finales y conclusiones

Voy a afrontar diversos puntos en este apartado:

Primero, algunas de las gráficas que se muestran en esta memoria han sido generadas mediante jupyter-notebook, por lo que no se mostrarán durante la ejecución del código en spyder, sin embargo las funciones para mostrarlas se mantienen para revisión del profesorado.

Segundo, debido al funcionamiento de RandomForest no se ha realizado una regularización de las estudiadas en teoría como la de castigar parámetros alejados del 0. La forma de regularización se muestra en el parámetro max_features que gobierna la cantidad de características (“features”) seleccionadas al azar (ahí aparece el factor aleatorio del experimento) que se emplean en generar cada árbol. Esa es nuestra mejor manera de evitar over-fitting con este modelo.

Tercero, si bien habría sido interesante utilizar el modelo Ridge() para probar los métodos de regularización y transformación no-lineal de parámetros estudiados en clase, la diferencia inicial de ajuste es demasiado grande como para ignorarla y puesto que se especifica que tomemos las decisiones que más vayan a minimizar nuestro Error es por lo que se ha tomado finalmente la decisión de emplear RandomForest().

Problema 2 - Dataset for Sensorless Drive Diagnosis

Para este problema disponemos de un DataSet que no especifica en qué consisten sus características. Únicamente disponemos de 58508 vectores 48 características en formato de valores reales (X), sus correspondientes etiquetas [1,11] que indican la categoría a la que pertenece dicho elemento (y) y debemos encontrar un modelo que permita predecir la categoría en la que entrará un elemento cualquiera (f).

Muchos de los pasos a seguir son iguales a aquellos utilizados para el problema 1 de regresión por lo que no entraremos en demasiada profundidad.

Comencemos hablando una vez más de la métrica de error a utilizar.

Métrica de Error

Para el error vamos a emplear la puntuación de precisión (accuracy score), que es una medida realmente sencilla que nos proporciona una fracción [0,1] del número de clasificaciones bien realizadas frente al número de clasificaciones totales realizadas.

En este problema no tiene sentido considerar de manera diferente errores de categorías más distantes numéricamente, ya que no tenemos ninguna información sobre si las categorías más cercanas son también más semejantes. Por tanto consideraremos un error como un único error, independientemente de si la predicción es 2 y el valor real es 1, o si la predicción de ese mismo valor es 11.

Reducción de Dimensionalidad: eliminación de características poco relevantes

El proceso que hemos llevado a cabo es idéntico al realizado para la regresión lineal, tras este proceso acabamos con únicamente 5 características relevantes para nuestro problema. Puesto que no disponemos de nada de información acerca de qué representan no he visto necesario incluir exactamente qué características son, aunque no sería difícil extraer esa información del código.

Elección del modelo

Se han tenido en cuenta los siguientes modelos de clasificación, LinearSVC(), SVC() y NuSVC() ya que las support vector machines son el modelo que mejor he entendido de los estudiados en teoría. Para seleccionar el más adecuado se ha realizado una prueba de rendimiento con los parámetros base sobre nuestro problema. El resultado se muestra en la siguiente tabla:

LinearSVC()	SVC()	NuSVC()
0.5658861733037087	0.13219962399589813	0.8531874893180653

Puesto que NuSVC() es el modelo que ha presentado una mayor precisión, he decidido focalizar en él y probar con diversos parámetros para intentar ajustarlo lo mejor posible.

Ajuste de NuSVC

Kernel

El parámetro fundamental, determina el algoritmo que se utilizará para el ajuste, las opciones son: Linear, Poly, Rbf o Sigmoidal. Dentro de la clase polinómica se tiene acceso a modificar el grado del polinomio que se utiliza, lo cual se menciona en el siguiente apartado.

Degree

El grado del polinomio en caso de utilizar un kernel polinomial. Este parámetro por razones evidentes se obvia para los demás kernels.

Otros

Los demás parámetros están preajustados para otorgar máxima precisión a cambio de coste computacional. Una vez más puesto que lo que nos interesa es máxima precisión los mantenemos por defecto.

Poly 1	Poly 2	Poly 3	Poly 4	Poly 5
0.866	0.760	0.784	0.680	0.712

Linear	Rbf	Sigmoid
0.851	0.874	0.772

El resultado de probar con los distintos parámetros mencionados es el que se muestra en la tabla superior. Finalmente escogemos por tener la mayor puntuación el modelo de radial basis function, que coincide con el parámetro por defecto de la función de sklearn.

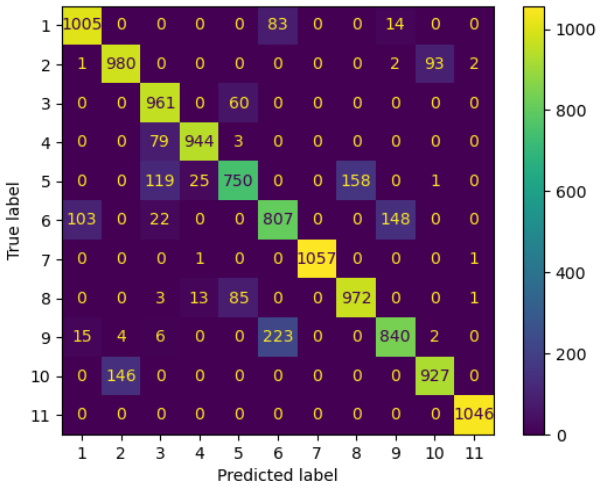
Entrenamiento y E_{OUT}

En este caso utilizamos la partición train-test en proporción 80/20 por los motivos que comentamos en la sección homónima del problema 1.
Entrenamos nuestro modelo con el conjunto de training y obtenemos el siguiente Eout.

E_{OUT} = 0.879251410015382

Quizás llamarlo E_{OUT} no sea lo más correcto ya que como mencionamos en la métrica de error el valor de esta medida oscila entre [0,1] siendo 1 el mejor valor posible pero por mantener la nomenclatura tradicional lo dejaremos así, sólo hay que tener en cuenta que en este caso es un valor a maximizar, no a minimizar. Podríamos declarar el Error como (1-accuracy_score) y así tendríamos un valor a minimizar pero no parece demasiado relevante en nuestro caso.

La mejor forma de visualizar este tipo de errores es mediante una matriz de confusión, así que, vamos a observar la cantidad de predicciones acertadas y erróneas en forma matricial.



Es bastante esperable el resultado, la mayor parte de las predicciones se encuentran en la diagonal (Predicciones correctas) y hay algunas distribuidas parece que de forma bastante heterogénea, llegando a formarse cúmulos de error significativos en ciertas categorías y ausencia total en otras.
Podemos considerar que o bien esas categorías poseen características comunes que las hacen difíciles de diferenciar, o bien nuestro modelo no posee la complejidad suficiente para aislar correctamente los casos.

Ahora vamos a hacer una estimación de nuestro E_{OUT} de la hipótesis final entrenada con todo el DataSet de la misma manera que lo hicimos anteriormente, sólamente recordar que el valor que tenemos ahora es inversamente proporcional a un error.

$$\left[E_{IN}\right] = \sigma^{2}\left(1 + \frac{d+1}{N}\right) = 0.875$$

$$\sigma^2 = \frac{0.875}{(1+\frac{d+1}{N})} = \frac{0.875}{(1+\frac{5+1}{58508})} = 0.875$$

Teniendo sigma cuadrado podemos obtener el valor esperado de E_{OUT} con su propia ecuación.

$$\left[E_{OUT}\right] = \sigma^2(1 - \frac{d+1}{N}) = 0.875(1 - \frac{5+1}{58508}) = 0.875$$

Y ya tenemos el valor esperado de nuestro E_{OUT}(g). En este caso tenemos aún más datos que en el caso de la regresión y la dimensionalidad de nuestro modelo es menor. Por lo tanto la variación es tan pequeña que no llega a ser apenas tangible.

Aclaraciones finales y conclusiones

Para este segundo problema tengo pocas aclaraciones que mencionar. El problema ha sido resuelto de una manera relativamente brusca y con poca profundidad matemática, soy consciente. Hay muchos puntos en los que se debería profundizar más y la cantidad de datos mostrados no son suficientes.
Pido disculpas por ello pero he hecho el mejor trabajo posible con el tiempo de que disponía.

Como conclusión destacar que los problemas de clasificación parecen relativamente más sencillos de afrontar que los de regresión y se dispone de multitud de funciones muy potentes para resolverlos.

Bibliografía y Enlaces de interés

https://scikit-learn.org/
https://towardsdatascience.com/
https://pandas.pydata.org/