Informe Assignment 1

ICAI. Machine Learning.

Álvaro Rodríguez González, Pablo Sanz Caperote

Curso 2021-22. Última actualización: 2021-10-25

Índice

Análisis exploratorio de los datos.

Comenzamos cargando tanto los datos como las librerias que necesitaremos para el desarrollo de nuestros modelos. Tras ello hacemos una primera exploración rápida de estos a través de la tabla de R.

A continuación lo primero que hacemos es ver como está estructurado nuestro conjunto de datos y nos damos cuenta de que todas las variables son de tipo numérico (int o num). Esto nos supone un problema para trabajar con los modelos ya que necesitamos que nuestra variable de salida (en este caso será DIABETES) sea un factor. Por ello transformamos la variable DIABETES en factor.

Tras este cambio, lo que haremos será hacer un summary de la tabla de datos con la finalidad de ver si esta tiene algun valor nulo (NA's). La función nos devuelve que no existe ningún NA, por ello podemos pasar a buscar valores atípicos dentro de nuestras variables.

Para encontrar los outliers lo primero que haremos será una representación gráfica de todas las variables juntas (un ggpairs) con la finalidad de ver como se comportan. En este gráfico observamos que las variables GLUCOSE, BLOODPRESS, SKINTHICKNESS, BODYMASSINDEX e INSULIN tienen valores los cuales podriamos considerar atípicos. Para una mejor valoración haremos boxplots de las diferentes variables.

Tanto en la variable GLUCOSE como en la variable BODYMASSINDEX existen datos que toman el valor 0, lo cual es absolutamente imposible. Por ello, tendremos que decidir que hacer con ellos. A nuestro parecer existen tres posibles opciones, eliminar dichas observaciones (con la consecuente pérdida de información del resto de variables) o sustituirlas por alguna medida de tendencia central como puede ser la media o la mediana. Como no sabemos que opción nos dará un mejor resultado en nuestros modelos lo que haremos será hacer un modelo sencillo que enfrente a la variable salida con la variable a la que queremos evaluar que hacer con los valores atípicos (en nuestro caso dicho modelo será una regresión logística). Finalmente mirando los resultados de la tabla 1 decidimos que los valores iguales a 0 de la variable GLUCOSE los sustituiremos por la mediana mientras que en BODYMASSINDEX lo haremos por la media. Cabe destacar que puede que los valores de accuracy sean más altos que los de la media o mediana pero estamos primando tener más información del resto de variables que tener un poco más de accuracy sobre una variable.

En el resto de variables hemos hecho el mismo proceso y toda la información de los modelos está en la tabla 1. Si es cierto que hay que hacer especial hincapié en una de las variables, INSULIN. Esta tiene una gran cantidad de 0's que no aparecen como outliers (se debe a que la gran cantidad de 0's afecta a los valores de la media y cuantiles) en los boxplot pero que si que hacen la función de outliers. Debido a que el número de observaciones que tienen 0 en la variable INSULIN representa casi la mitad de la muestra es obvio que se descartará la opción de eliminar dichas observaciones, y por ende solo quedará la opción de sustituir las observaciones por la media o la mediana.

Variable	Eliminar Datos	Sust. Media	Sust. Mediana
GLUCOSE	0.7479623	0.7400434	0.7415142
BLOODPRESS	0.6533898	0.6536022	0.6466102
SKINTHICKNESS	0.6673865	0.632392	0.6347352
BODYMASSINDEX	0.6676503	0.6634551	0.6564784
INSULIN	-	0.6536224	0.6504231

Cuadro 1: Comparación accuracy opciones outliers

Una vez resuelto el problema de los outliers analizaremos las diferentes variables continuas del conjunto de datos. Usando otra vez el ggpairs nos damos cuenta de que las escalas de nuestras variables son muy diferentes, por ello tendremos que hacer una estandarización. Pero esta se implementará cuando nos pongamos a trabajar con los modelos, por tanto ahora mismo no nos tendremos que preocupar.

Antes de iniciar el desarrollo de los modelos debemos estudiar las posibles correlaciones que existen entre nuestras variables con el fin de poder conocer si existen variables que nos predigan lo mismo. Como se puede observar en la imagen inferior la correlación entre nuestras variables es muy baja como para contemplarse cualquier acción, por lo que podemos pasar al siguiente punto.

Por último antes de comenzar con los modelos miraremos si nuestra clase de salida esta balanceada o no. En caso de no estar deberemos contemplar si la balanceamos o no. Haciendo un table sobre la variable DIABETES observamos que nuestra clase no esta para nada balanceada 70-30. Por ello lo que haremos será mirar si en los modelos, estos son capaces de medirnos bien la salida de "Si".

Haciendo una regresión logística obtenemos que sensitividad tanto en entrenamiento como en test es muy baja, 0.6 y 0.35 respectivamente, por lo que nos vemos obligados a balancear los datos. Para ello usaremos la libreria ROSE a traves de la función ovun.sample. A su vez también definiremos los métodos de control y la partición de datos (80-20) que usaremos para nuestros modelos.

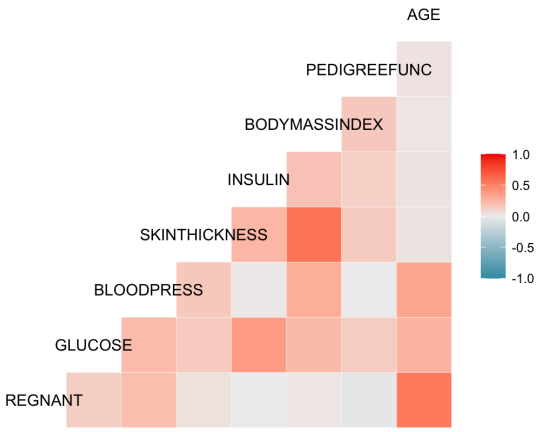

Figura 1: Matriz correlación

Regresión Logística

Distinguiremos entre dos modelos diferentes, el primero donde trabajaremos sobre todas las variables y veremos cuales son las más importantes y luego el optimizado que solo tomará las variables clave.

Caso general

Nuestro primer modelo de regresión logística nos da un accuracy de 0.746 y un valor de kappa de 0.4925. Pero lo realmente interesante es que el modelo nos da también las variables más importantes, si ejecutamos la función summary obtenemos que las variables PREGNANT con un p-valor de 6.32e-05, GLUCOSE con p-valor menor que 2e-16 , BODYMASSINDEX con p-valor de 8.80e-08 y PEDIGREEFUNC con p-valor de 0.00785 son las más importantes de nuestro modelo.

Una vez obtenidas cuales son las variable más importante podemos pasar a optimizar nuestro modelo.

Optimizada

En este modelo de regresión logística tendremos como inputs a las cuatro variables obtenidas en el modelo anterior y como variable de salida tendremos a DIABETES. Además estableceremos como parametro de control una validación cruzada con 10 folds.

Tras entrenar el modelo con una partición de 80-20 y 10 folds, obtenemos un accuracy de 0.76 con un valor de kappa de 0.52. Esto nos indica que trabajar con variables las cuales el modelo no considera importantes puede lastrarnos haciendo que fallemos más de lo debido. La validación cruzada nos da los siguientes valores de accuracy:

Nº Fold	1	2	3	4	5	6	7	8	9	10
Accuracy	0.7875	0.7750	0.8125	0.5875	0.7625	0.7500	0.8250	0.6875	0.8000	0.7500

Cuadro 2: Accuracy Oversampling

Estos son altos pero existen algunos casos donde existe gran diferencia con el resto de muestras, como es el caso del fold 4 que esta por debajo del 0.6.

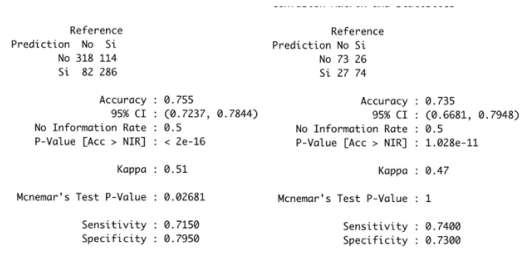

Figura 2: Matrices de confusión

Tras haber entrenado al modelo con el conjunto de datos de entrenamiento vamos a evaluarlo con nuestro conjunto de test. Para comprobar que todo va bien usaremos las matrices de confusión del modelo tanto para el conjunto de entrenamiento como para test. Como se puede comprobar en la imagen inferior nuestro modelo tiene un accuracy muy parecido tanto en training como en test por lo que podemos dejar de lado el problema de sobreentrenamiento.

También es interesante el estudio de nuestros modelos a partir de sus curvas ROC. El objetivo es que se acerquen lo máximo posible a la esquina superior izquierda, para poder obtener un valor muy alto de área bajo la curva, ya que esto indica que habrá un alto ratio de Positivos Verdaderos (TP)/Falsos Positivos (FP).

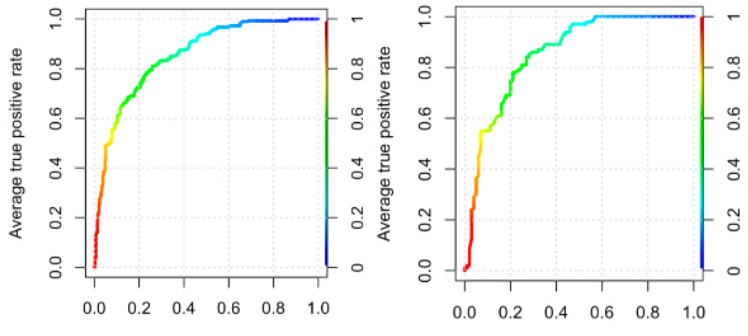

Figura 3: Curvas ROC

KNN

El modelo KNN se basa en comparar cada uno de los datos con un número k de vecinos más cercanos. Para este modelo se necesita el hiperparámetro k, que indica el número de vecinos cercanos que se utilizan en la comparación.

Para obtener el k más óptimo se realiza un barrido con diferentes k hasta obtener los resultados más óptimos. En la figura de la derecha se puede observar el barrido del parámetro k.

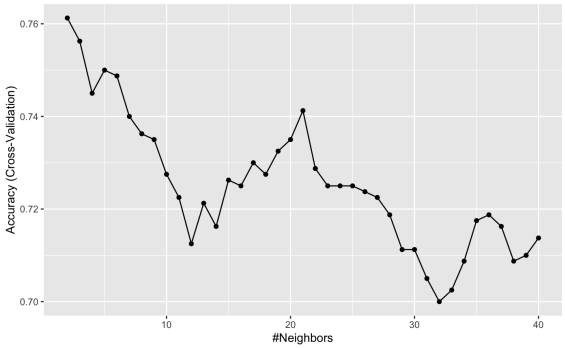

Figura 4: Valores de diferentes k

Se puede observar que el valor de accurancy más alto se obtiene cuando el k=2. También podemos observar que el valor de k más alto que hemos probado es 40, porque se ve claramente que el accuracy tiene una línea descendente cuanto más alto es el k. Que el valor seccionado sea $k = 2$ puede ocasionar un sobre entrenamiento y por ello vemos también el valor Kappa que en este caso es Kappa = 0,5225.

Evaluando los resultados más de cerca, podemos calcular la matriz de confusión para los datos de test. Esta nos dará información importante como el accuracy de nuestro modelo en el test. Este nos da un accuracy de 0.715 lo cual no esta mal ya que su diferencia con el accuracy del modelo es relativamente pequeña.

	No	Sí
No	62	19
Sí	38	81

Cuadro 3: Matriz confusión kNN

Además de esta matriz, vamos a utilizar el indicador de la curva ROC. A continuación vemos la gráfica de la curva ROC para el conjunto de test. El área bajo la curva tiene un valor de 0.81145 lo que es aceptable.

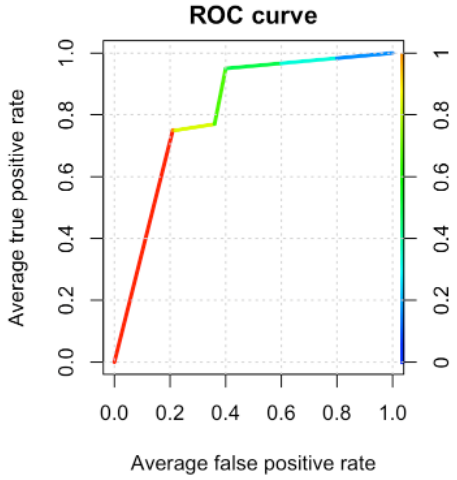

Figura 5: ROC KNN Test

Árboles de decisión

El modelo de árboles de decisión sigue un consiste en realizar divisiones recursivas. Para ello se analiza la mejor variable para dividir el conjunto de datos y se van obteniendo diferentes ramificaciones que permiten clasificar cada uno de los datos. El número de divisiones que se ejecutan son los diferentes nodos del árbol. En estos modelos el hiperparámetro que se utiliza es uno llamado cp que sirve para penalizar a los árboles que tienen una altura mayor. Si este parámetro tuviera el valor 0, el árbol tendría un gran número de nodos y muy probablemente el modelo estaría sobre entrenado. Por ello, realizaremos un barrido de este parámetro cp yendo desde 0 hasta 0.1 en pasos de 0.001. Además, entrenaremos 3 modelos diferentes que podemos ver a continuación.

Caso general

Este modelo se entrenará con todas las variables del conjunto de datos inicial. En la parte derecha se muestra el gráfico del accuracy variando el cp:

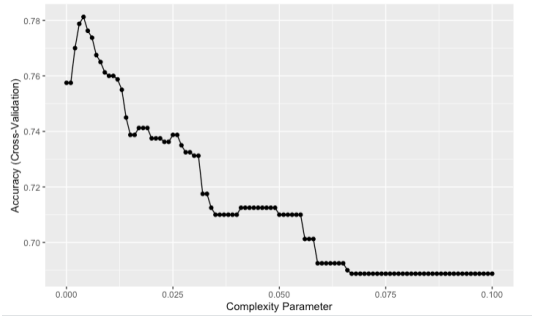

Figura 6: Accuracy y cp

Se puede ver que el valor más alto es $cp = 0{,}004$. Evaluando los resultados más de cerca, tenemos la matriz de confusión:

	No	Sí
No	74	20
Sí	26	80

Cuadro 4: Matriz confusión DecisionTree

Optimizada

Pero además de los datos representados anteriormente, este modelo nos proporciona una tabla que indica qué variables son importantes.

Podemos ver que las variables importantes son: Glucose, Age, BodyMassIndex, Pedigreefunc, SkinThinckness y Pregnant. Aunque algunas de ellas tienen más importancia que otras, vamos a coger estas 6 para crear un nuevo modelo. Y analicemos la Accurancy en función del hoperparámetro cp:

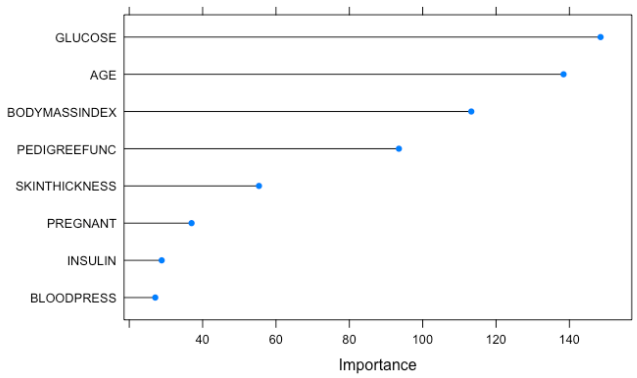

Figura 7: Importancia DT

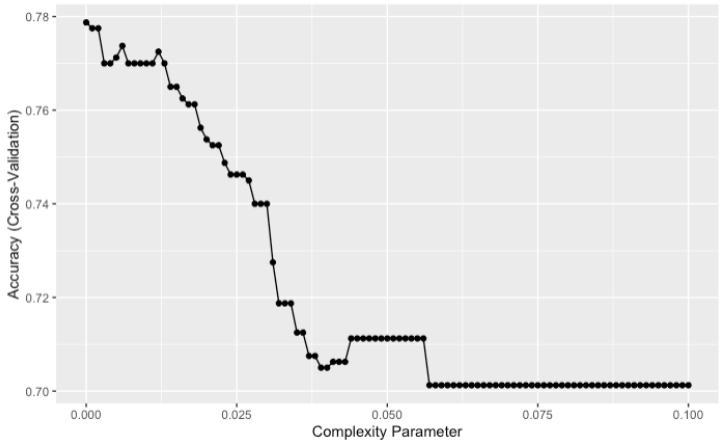

Figura 8: Accurancy y cp

Se puede ver que el valor más alto es $cp = 0$. Evaluando los resultados más de cerca, tenemos la matriz de confusión:

	No	Sí
No	69	22
Sí	31	78

Cuadro 5: Matriz confusión DecisionTree2

Además de esta matriz, vamos a utilizar el indicador de la curva ROC. Y al igual que los Accurancy eran bastante similares, ocurre lo mismo con el indicador de la curva ROC. Porque para el modelo anterior el área bajo la curva tiene un valor de 0.8215. Y para este nuevo modelo tenemos un valor de del área bajo la curva de 0.8201, que veremos en la siguiente figura. Por todo esto podemos decir que los modelos son bastante similares.

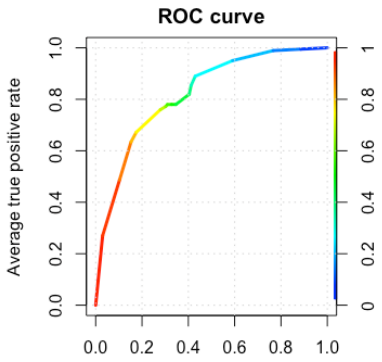

Figura 9: ROC DC Test 2

SVM

Aquí hemos utilizado dos modelos diferentes dentro de la misma familia. El SVM lineal y el SVM Radial. No obstante, para ambos modelos se utiliza un hiperparámetro C. Para calcular cual es el valor más óptimo mostraremos un barrido de la variación del Accurancy con dicho parámetro. Analizaremos cada uno de los 2 casos:

SVM Lineal

Veamos el barrido del parámetro C con este modelo:

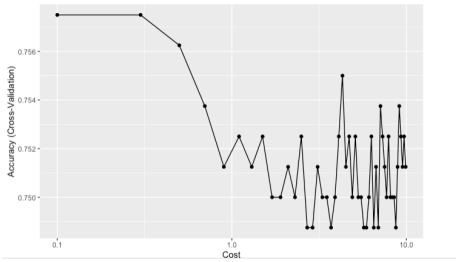

Figura 10: SVM Lineal parámetro

El parámetro C comentado anteriormente refleja el grado de flexibilidad que permite la separación de los puntos del modelo. A mayor C, mayor es el número de observaciones que pueden estar en el lado incorrecto de la separación. Mirando en el gráfico se puede ver que los puntos $C = 0{,}1$ y $C = 0{,}3$ generan los mismos valores de Accurancy, entorno a 0.78. Sin embargo, nos quedaremos con el primero de ellos, $C = 0{,}1$. Calculando los resultados en función del conjunto de test obtenemos un $Accurancy = 0{,}73$. Y evaluando los resultados más de cerca, tenemos la matriz de confusión:

	No	Sí
No	74	28
Sí	26	72

Cuadro 6: Matriz confusión SVM Lineal

SVM Radial

Analicemos ahora la versión mejorada del modelo anterior. En este modelo no solo tenemos que modificar el parámetro C como ocurría en el SVM Lineal, sino que también tenemos que ver el parámetro Sigma. Para obtener una correcta configuración hemos establecido un rango de ambos parámetros y hemos realizado otro barrido. Lo vemos en la Figura 13.

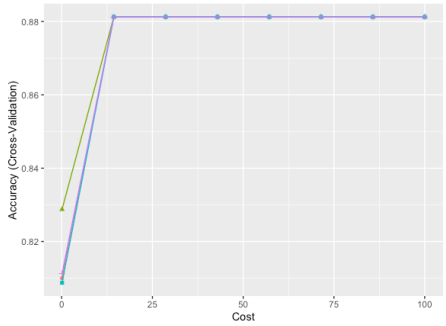

Figura 11: Params SVM Radial

Se puede ver que los diferentes valores de sigma convergen cuando C es grande. Por este motivo vamos a coger la línea que siempre está por encima y que representa el valor $sigma = 450$ y $C = 14{,}5$ Esta configuración genera una $Accurancy = 085$ para el conjunto de test. Evaluando los resultados más de cerca, tenemos la matriz de confusión:

	No	Sí
No	100	30
Sí	0	0

Cuadro 7: Matriz confusión SVM Radial

Además de esta matriz, vamos a utilizar el indicador de la curva ROC. A continuación vemos la gráfica de la curva ROC para el conjunto de test.

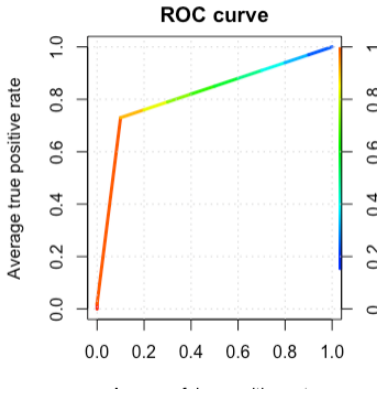

Figura 12: ROC SVM Radial

El área bajo la curva tiene un valor de 0.85 lo que bastante bueno. De hecho, es un poco mejor que en el modelo SVM Lineal, cuyo área bajo la curva es 0.819. Por todo esto, este nuevo modelo mejora al anterior.

Redes Neuronales

Los modelos de redes neuronales se basan en los hiperparámetros que señalan el número de nodos en la capa oculta (para nosotros es por defecto una sola), y el parámetro lambda del weight decay. Nuestra idea en este modelo será similar a la de la regresión logística, primero evaluaremos el modelo al completo para conocer las variables importantes y posteriormente haremos un modelo con las variables importantes unicamente.

Caso general

En este modelo nuestro objetivo es obtener las variables clave. Por ello entrenamos nuestro modelo con todas las variables y nos sale un accuracy de 0.85 con un valor kappa de 0.7 Las variables más importantes de nuestro modelo son BODYMASSINDEX, AGE, SKINTHICKNESS, GLUCOSE y BLOODPRESS. Esto se puede observar en el análisis de sensibilidad de la imagen inferior.

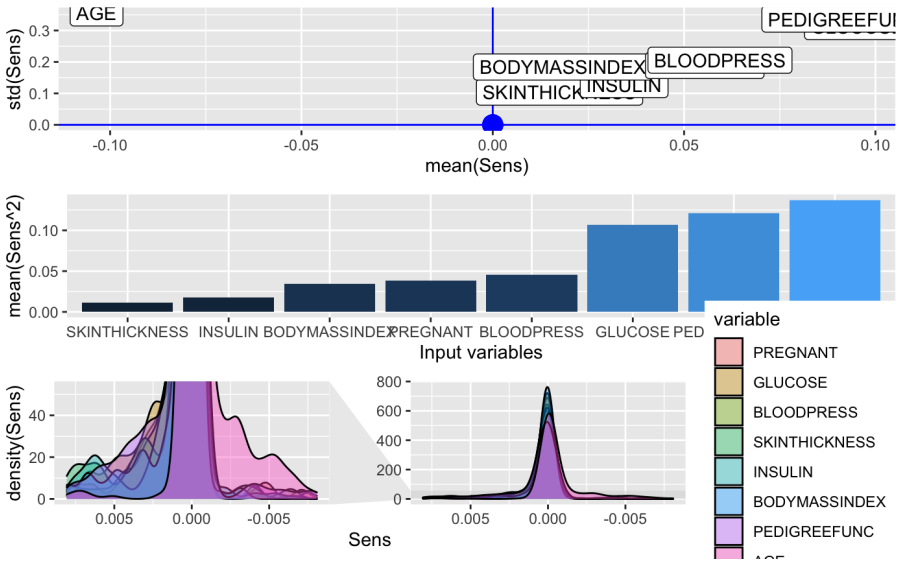

Figura 13: Análisis de sensibilidad

Una vez obtenidas cuales son las variable más importante podemos pasar a optimizar nuestro modelo.

Optimizada

En este modelo de redes neuronales tendremos como inputs a las cuatro variables obtenidas en el modelo anterior y como variable de salida tendremos a la variables DIABETES. Además estableceremos como parametro de control una validación cruzada con 10 folds.

A su vez para realizar un modelo óptimo estableceremos un rango de valores tanto para el número de

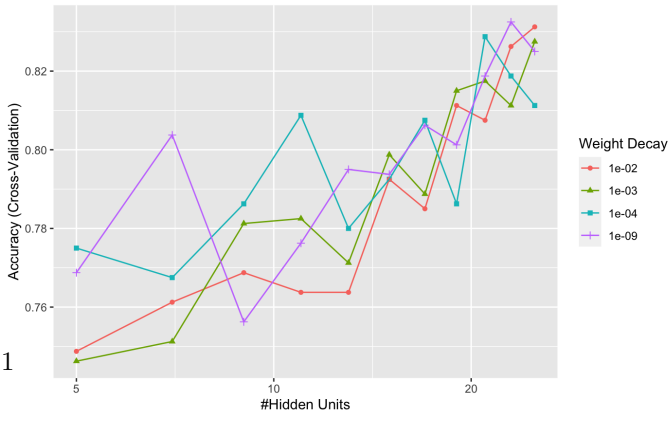

Figura 14: Elección de parámetros

nodos como para el valor de decay. Esto permitirá elegir aquellos parámetros que maximicen el accuracy de nuestro modelo. El como varia el accuracy del modelo según los parametros se puede ver en la imagen de la derecha.

El modelo optimizado nos da un accuracy de 0.8325, lo cual esta bastante próximo al valor del modelo anterior. Esto significa que nuestro modelo funciona de forma correcta. A su vez tras haberlo configurado con 250 iteraciones nos da como valores óptimos de número de nodos y decay, 23 y 1e-09 respectivamente.

Como en los modelos anteriores para ver si nuestro modelo no cae en la trampa del sobreentrenamiento calculemos las matrices de confusión, las cuales devolverán el accuracy de cada conjunto para poder compararlos.

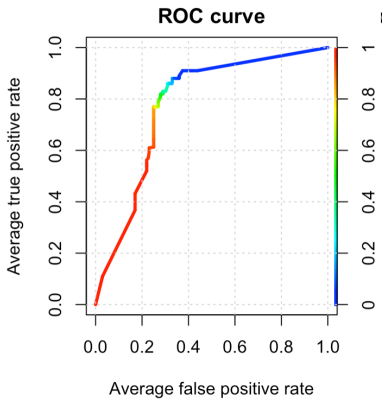

Figura 15: Curvas ROC

En el caso del conjunto de entrenamiento este presenta un accuracy de 0.9675 el cual es realmente alto y puede indicar que nos encontramos ante algún problema de overfitting.

El accuracy del conjunto de test es de 0.765 lo cual es significativamente más pequeño que el de entrenamiento. Dicha diferencia no puede ser nada bueno y muy probablemente lo que ocurra es que nuestro modelo se ha sobreentrenado y por ende no va a generalizar bien.

Por último, para poder comprobar el rendimiento del modelo, se representan las curvas ROC, calculándolas áreas bajo las curvas. En la imagen inferior podemos observar como para el entrenamiento la curva casi toca la esquina superior izquierda, mientras que en el test esta dista mucho de la anterior.

Podemos afirmar que nuestro modelo ha caido en el overfitting y no se podrá usar para ninguna generalización.

Random Forest

Como queriamos ir un poco más alla en el proyecto y la parte de modelos hemos decidido realizar un modelo que aún no hemos visto y que en si es una generalización de los arboles de decisión. El modelo en cuestión son los random forest que consisten en una gran muestra de arboles de decisión todos ellos entrenados con una muestra ligeramente diferente. A su vez también estableceremos una lista de valores de donde puede salir el hiperparametro de dicho modelo (el hiperparametro consiste en el número de variables muestreadas aleatoriamente como candidatas en cada división). Otro parametro que le meteremos es que el número de arboles sea 500.

Tras entrenarlo con nuestro conjunto de entrenamiento el modelo nos devuelve un accuracy de 0.87875, lo cual es un valor bastante alto. A su vez el valor del número de variables muestreadas aleatoriamente óptimo es de 2.

```
          Reference
Prediction  No  Si
        No 400   0
        Si   0 400

               Accuracy : 1
                 95% CI : (0.9954, 1)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0
            Specificity : 1.0
```

```
          Reference
Prediction No Si
        No 80  6
        Si 20 94

               Accuracy : 0.87
                 95% CI : (0.8153, 0.9133
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.74

 Mcnemar's Test P-Value : 0.01079

            Sensitivity : 0.9400
            Specificity : 0.8000
```

Figura 16: Matriz de confusión

Para saber si dicho modelo es un modelo correcto calcularemos las matrices de confusión. La matriz de confusión de los datos de entrenamiento no deja duda alguna, el modelo lo ha hecho a la perfección ya que tenemos un accuracy de 1. Ahora debemos mirar el valor del accuracy para los datos de test, donde si este tuviese una gran diferencia con el de los datos de entrenamiento evidenciaria que nos encontramos antes un caso de overfitting. El accuracy para los datos de test es de 0.87, lo cual dista un poco del accuracy de los datos de entrenamiento pero es un valor razonable para negar la existencia de overfitting.

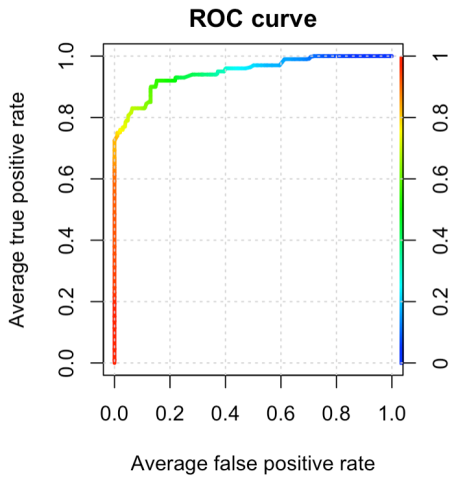

Figura 17: Curva ROC Random Forest

Cabe destacar que nuestro modelo falla principalmente en la detección de los negativos, ya que tiene una especificidad de 0.8 frente a una sensitividad de 0.94.

Por último miraremos la curva ROC de los datos de test, ya que la de los datos de entrenamiento es obvio como va a ser al tener un accuracy de 1. En la imagen de la izquierda podemos observar como la curva ROC se acerca en gran cantidad a la esquina superior izquierda, lo que nos dice que nuestro modelo se ajusta bien y permite generalización. A su vez tenemos que el area bajo la curva (AUC) es de 0.94775, es decir, nuestro modelo generalizará bastante bien antes nuevos datos que no hayan sido evaluados.

Comparación de modelos

Para comparar los modelos usaremos una tabla donde compararemos los diferentes accuracies obtenidos asi como también veremos los hiperparametros seleccionados. A su vez también usaremos una gráfica para comparar de formas más visual los diferentes modelos. Veamos como es la tabla:

Modelo	Hiperparámetro	Accuracy CrosVal	Train Accuracy	Test Accuracy
RegLog	-	0.746	0.75	0.74
RegLog Óptima	-	0.76	0.755	0.735
KNN	k = 2	0.762	0.763	0.715
Arbol Dec	cp = 0.004	0.78125	0.82	0.775
Arbol Dec Óptimo	cp = 0.004	0.782	0.79	0.77
SVM Lineal	C= 0.1	0.757	0.79	0.73
SVM Radial	σ = 450 y C= 14.53	0.88	0.965	0.85
RNeuro	n = 10 y decay = 1e-04	0.85	0.8962	0.825
RNeuro Óptimo	n = 23 y decay = 1e-09	0.8325	0.9675	0.765
Random Forest	mtry = 2	0.87875	1	0.87

Veamos la gráfica que nos muestra los diferentes accuracies de los modelos evaluados:

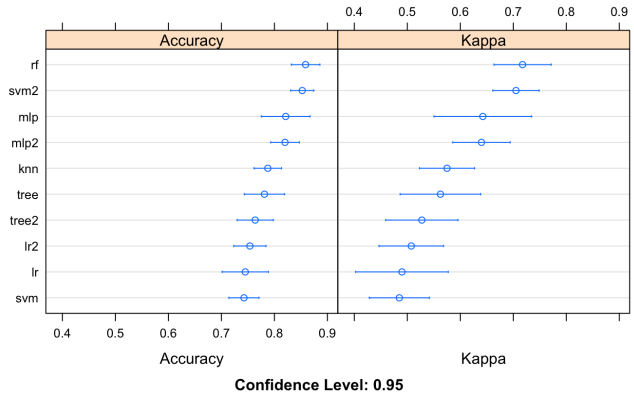

Figura 18: Comparación modelos

Viendo ambos elementos para comparar podemos afirmar que los modelos que mejor predicen para este tipo de datos son el modelo de random forest y el SVM radial.

Conclusiones

Haciendo enfoque en los resultados obtenidos, se puede ver que son relativamente buenos ya que en la gran mayoria de ellos la diferencia entre el accuracy de entrenamiento y de test no es relevante. Sin embargo existen excepciones como el caso de redes neuronales donde existe una diferencia relevante entre train y test. Si tuviesemos que escoger un modelo para trabajar con este tipo de datos estariamos entre el modelo de random forest o el de support vector machine con un método radial, ya que son los que mejor accuracy tienen y no tienen una gran diferencia entre entrenamiento y test.

Parte Extra

Por último, hemos creado un nuevo modelo modificando previamente el conjunto de datos. La modificación realizada consiste en tomar como variable categórica la variable *Pregnant* en lugar de tomarla como variable continua. De esta forma, hemos creado los siguientes grupos: 0 hijos, 1, 2, 3, 4 y 5 o más. Para comparar los resultados únicamente hemos creado un modelo de Redes Neuronales para compararlo con los 2 que ya teníamos creados y evaluar si de esta forma obtenemos mejores resultados.

Aplicamos el mismo proceso que en los modelos anteriores realizando un barrido de los hiperparámetros similar. En la siguiente figura vemos esto:

La configuración óptima para este modelo son $neuronas = 12$ y $decay = 1e-09$ De esta forma obtenemos un $Accurancy = 0{,}83$ para el conjunto de entrenamiento. Y evaluando el conjunto de Test, tenemos un resultado de $Accurancy = 0{,}79$. Si lo comparamos con los 2 modelos de Redes Neuronales anteriores vemos que se comparta bastante peor que ellos.

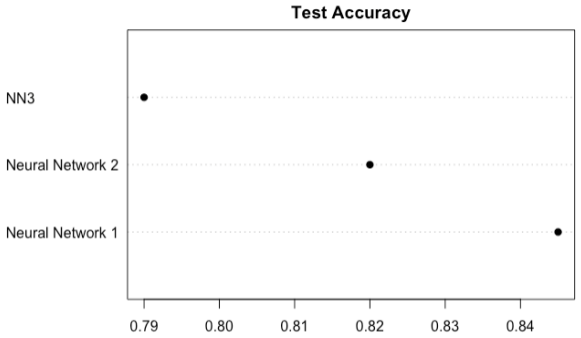

Figura 19: Comparativa NN