TP6

Tomás Hüttebräucker
Aprendizaje Profundo y Redes neuronales artificiales
(6 de noviembre de 2020)

1. CLASIFICACIÓN DEL DATASET CARSEATS

Se desarrollaron varios métodos para poder clasificar el dataset Carseats. Para esto se cambiaron todas las variables del tipo *string* por numero enteros.

1.a. Separación de datos

Se dividieron los datos en training y testing, con el tamaño de testing 30 % del total de los datos.

1.b. Arbol Clasificador

Se buscó crear un árbol que permita clasificar a los datos de acuerdo a si su valor de *Sales* es mayor o menor a 8, para esto se creo una nueva variable llamada *High*, que sea 1 si *Sales* es mayor a 8 y 0 si es menor. Una vez definida esta variable, se removió la variable *Sales*, ya que había un correlación perfecta entre las mismas, y se definieron los datos y su clasificación por separado. Con los datos definidos, se entrenó el árbol sin limites de complejidad.

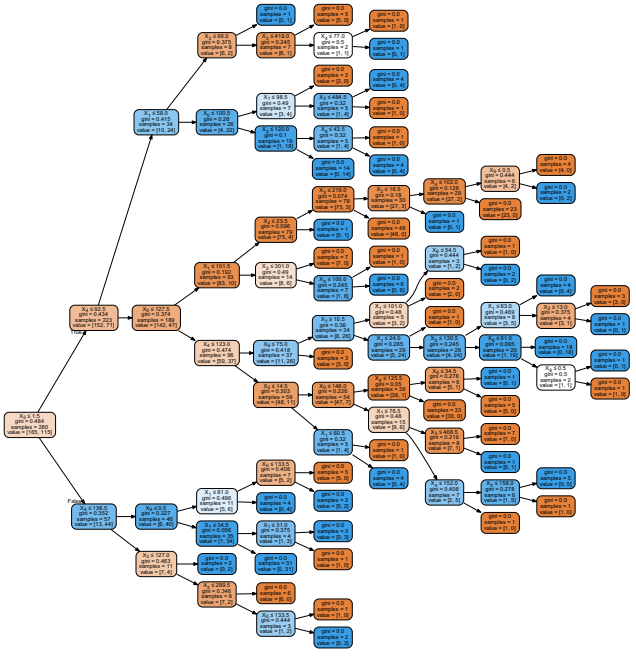


Figura 1: Árbol de clasificación generado.

En la figura 1 se muestra la estrucura del árbol obtenido, se ve que tiene una complejidad considerable, lo que indicaróa que es probable que haya overfitting. Viendo que la precisión sobre los datos de entrenamiento es 100 % y sobre los datos de testing es 66 %, se puede concluir que el árbol overfittea los datos.

1.c. Árbol Regresor

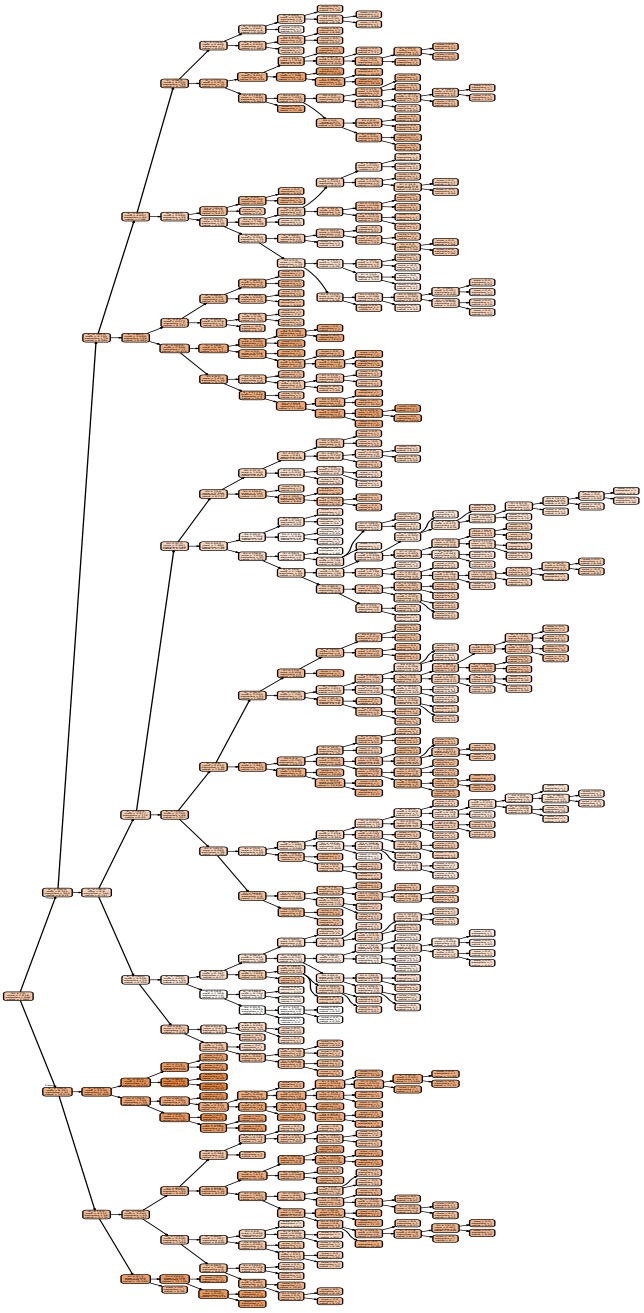


Figura 2: Árbol de regresión generado.

Ahora se busco crear un árbol regresor que estime la variable *Sales* de los datos. En este caso se dejó fuera la

variable *High*, por la correlación de las mismas. Se entrenó el árbol, sin restricción de complejidad.

Para verificar la precisión del modelo, se usa el parámetro de *score* S del mismo, el cual consiste en:

$$S = 1 - \frac{\sum(y_{true} - y_{pred})^2}{\sum(y_{true} - y_{mean})^2}$$

donde y_{true} son los valores correctos, y_{pred} son los valores predichos y y_{mean} es la media de los valores correctos. De esta forma, si la predicción es perfecta $S = 1$ y $S\epsilon[-\infty, 1]$, a mayor score, mejor es el regresor.

Como se muestra en la figura 2, la estructura del árbol es extremadamente compleja, contando con 279 hojas, que, siendo los datos de entrenamiento 280 en total, claramente indica que hay overfitting. Observando que el valor de score para los datos de entrenamiento es de 1 y para los datos de testing es 0.43, se puede concluir que el modelo presenta un overfitting importante.

1.d. Comparación

Ambos modelos presentan overfitting considerable.

1.e. Cross Validation + Pruning

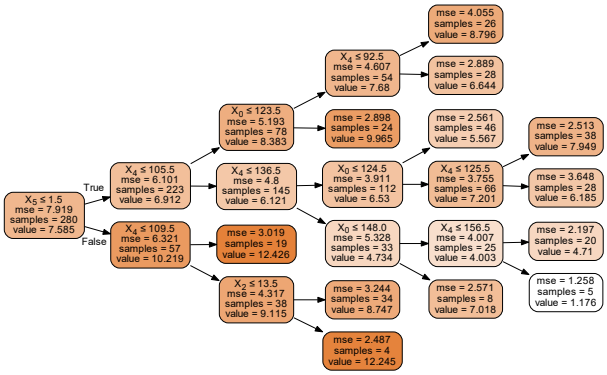


Figura 3: Árbol de regresión generado.

Se procedió a realizar la regresión, entrenando con el método de *CrossValidation+Pruning*. Para esto se realizó primero una búsqueda de los parámetros de máxima profundidad y de el valor de α de *Pruning* para un arbol regresor. La busqueda se realizo con la función `GridSearchCV` que evalúa el desempeño de varios estimadores con los parámetros indicados usando *CrossValidation*. Se barrió la máxima profundidad entre 1 y 20 y el valor de α entre 0 y 2 con un paso de 0.02. Los parámetros óptimos resultaron ser α=0.06 y `max_depth=7`. En la figura 3 se muestra la estructura del árbol de regresión entrenado con los parámetros óptimos usando.

Se ve que la estructura del árbol es mucho mas simple que las anteriores. Esto es debido al *Pruning* y al `max_depth`. Es esperable que no haya tanto overfitting esta vez. Sobre los datos de entrenamiento se obtuvo un score de S_{train}=0.63, bastante peor que el arbol sin *Pruning* y con complejidad ilimitada, sin embargo, sobre los datos de test se obtuvo un score de S_{test}=0.48, notablemente mejor.

1.f. Bagging

Se utilizó la técnica de ensamble *Bagging* para entrenar un clasificador basado en el mejor clasificador obtenido en el inciso anterior. Con 10 estimadores en total los scores del ensamble fueron S_{train}=0.74 y S_{test}=0.63. Una considerable mejoría en comparación con el inciso anterior. Al ser un ensamble, no hay forma de graficar la estructura del *Bagging*.

En la tabla 1 se muestra la importancia de los datos a la hora de clasificar. Las caracteristicas que no aparecen en la tabla no tienen importancia para el modelo.

Característica	**Importancia**
Price	0.41
ShelveLoc	0.24
CompPrice	0.11
Age	0.08
Advertising	0.08
Population	0.03
US	0.015
Income	0.013
Education	0.012

Tabla 1: Importancia de características al clasificar con Bagging.

1.g. Random Forest

Se usó la técnica de *Random Forest* para entrenar un ensamble de estimadores y hacer la regresión. Se varió la profundidad máxima de los estimadores y el número máximo de las caracteristicas a usar para ele entrenamiento.

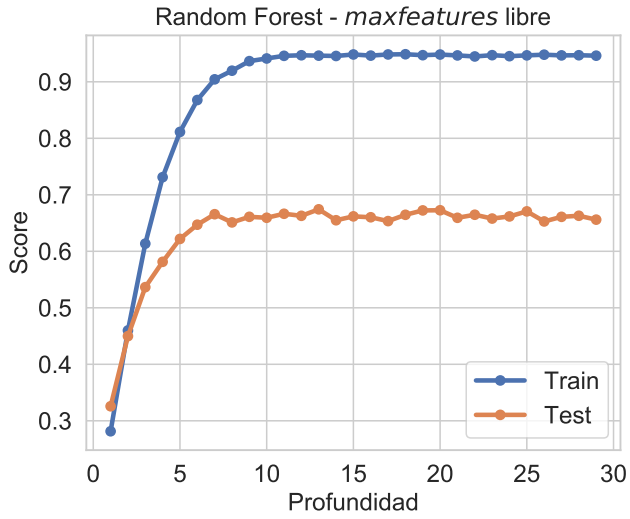
En la figura 4 se muestra el score sobre los datos de test y train en función de las variables `max_depth` y `max_features` del regresor. Se ve que a partir de una profundida de aproximadamente 5, el modelo no cambia su comportamiento y a medida que mas carácteristicas se incluyen, mejora su generalización.

Característica	**Importancia**
Price	0.34
ShelveLoc	0.21
CompPrice	0.12
Age	0.10
Advertising	0.08
Income	0.05
Population	0.04
Education	0.03
US	0.01
Urban	0.005

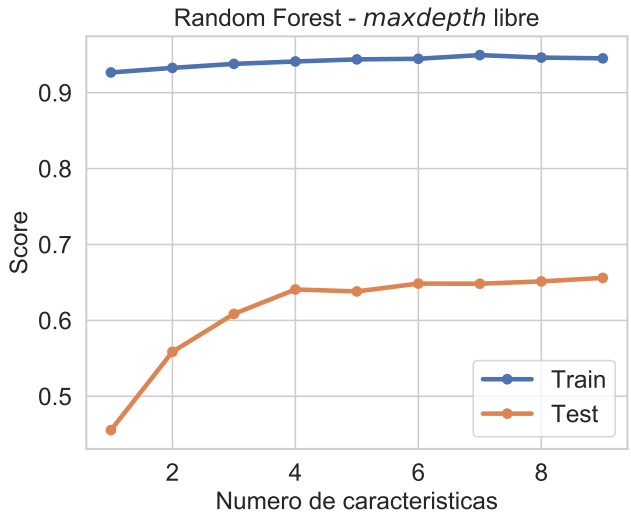
Tabla 2: Importancia de características al clasificar con Random Forest.

En la tabla 2 se muestran las características ordenadas por importancia para el clasificador *Random Forest*. Se ve que en general son similares a las importancias obtenidas

con el clasificador *Bagging*, cambiando un poco los valores en general y el orden en las últimas.



(a) Variando la profundidad de los árboles.



(b) Variando el numero de características máximas.

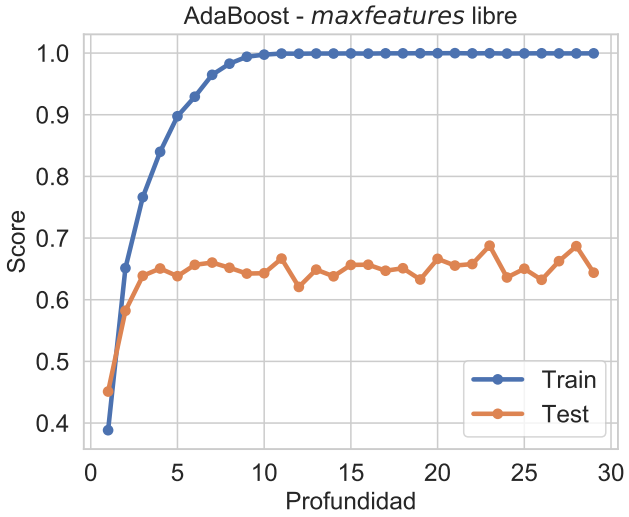
Figura 4: Score sobre train y testo variando `max_depth` y `max_features` del regresor *Random Forest*.

1.h. Ada Boost

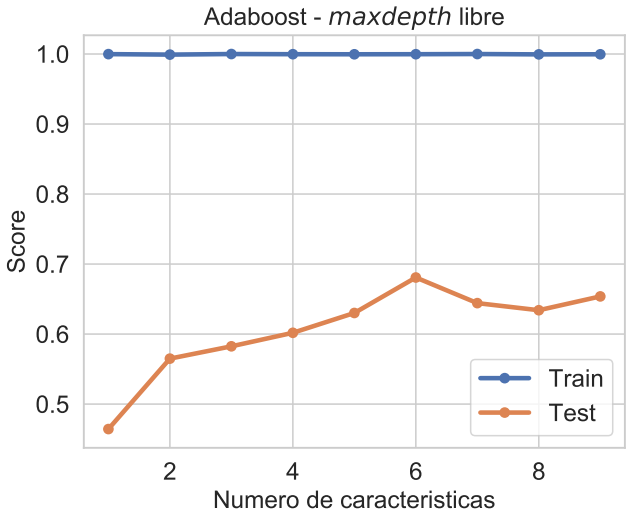
Se utilizó la técnica de AdaBoost para entrenar un regresor. Para esto se uso como base un regresor del tipo árbol de decisión. Nuevamente se varió la profundidad máxima de los estimadores y el número máximo de las características a usar para el entrenamiento.

Característica	Importancia
Price	0.37
ShelveLoc	0.16
CompPrice	0.12
Age	0.10
Advertising	0.08
Income	0.06
Population	0.04
Education	0.03
US	0.01
Urban	0.005

Tabla 3: Importancia de características al clasificar con Random Forest.



(a) Variando la profundidad de los árboles.



(b) Variando el numero de características máximas.

Figura 5: Score sobre train y testo variando `max_depth` y `max_features` del regresor AdaBoost.

En la figura 5 se muestra el score sobre los datos de test y train en función de las variables `max_depth` y `max_features` del regresor. Comparandolo con el *Random Forest*, se ve que consiguen resultados similares para los datos de test, pero el AdaBoost mejora significativamente

el score para train, esto indica que hay mas overfitting.

En la tabla 1 se muestran la importancia de las características según el regresor. El orden de las mismas es el mismo que en el modelo *Random Forest* y hay una ligera variación en los valores de las características mas relevantes.

CONCLUSIÓN Y COMPARACIÓN

En la tabla 4 se muestran los scores sobre los diferentes modelos. Se ve que usar un Árbol simple es claramente la peor forma de realizar la regresión. EN principio parece mejor usar Ada Boost que Random Forest y Random Forest que Baggins. Sin embargo hay una dependencia entre como son los datos y cual es el método mas adecuado.

Método	**Score Test**
Árbol simple	0.42
CV + Pruning	0.48
Bagging	0.63
Random Forest	0.67
Ada Boost	0.69

Tabla 4: Calores de score sobre los datos de test para los diferentes estimadores.