Práctica 6: Método basado en Árboles de Decisión

Evelyn G. Coronel
Redes Neuronales y Aprendizaje Profundo para Visión Artificial
Instituto Balseiro

(6 de noviembre de 2020)

EJERCICIO 1

Item A

Este ejercicio, se utilizan arboles de clasificación y regresión para obtener la cantidad de unidades de asientos para bebés se compran, en función de las características del producto y del lugar donde se venden. Las variables a tener en cuenta del conjunto de datos son:

1. Sales: Cantidad de asientos vendidos en una tienda.
2. CompPrice: Precio del asiento según la competencia.
3. Income: Ingreso de dinero medio en miles de USD del lugar donde se localiza la tienda.
4. Advertising: Inversión en propaganda de la tienda en miles de USD.
5. Population: Población local en miles.
6. Price: Precio de venta por cada asiento.at each site
7. ShelveLoc: Posición del asiento en vitrina, se clasifica las posición como Bad, Good and Medium (Malo, Bueno y Medio).
8. Age: Edad promedio de la población local.
9. Education: Nivel de educación
10. Urban: *Yes* si la tienda está ubicada en un lugar urbano, *No* caso contrario.
11. US: *Yes* si la tienda está ubicada en Estados Unidos, *No* caso contrario.

Además de estas variables, se agregó una última variable llamada *High* es *Yes* si las ventas superan las 8 unidades, *No* caso contrario.

Para realizar los ajustes, se transformaron los datos descriptos *Yes* y *No* como 1 y 0 respectivamente, y *Good*, *Medium*, *Bad* con 3, 2 y 1. Además se separó el conjunto de datos en un 80 % de entrenamiento y un 20 % de validación. Las variables *Sales* y *High* se separan del resto de los atributos para hacer los ajustes.

Item B

Usando un árbol de decisión de clasificación, se utilizó la columna *High* como salida esperada y el resto de los atributos como entrada. El árbol está instanciado de tal forma que el ajuste se realice hasta que las hojas sean puras, es decir, que el coeficiente de gini del nodo sea 0. Con esto se obtuvo un árbol con las características que se presentan en la Tabla I. Los valores de precisión son los scores para los datos de validación o entrenamiento según corresponda.

Profundidad	11
Hojas	50
Error de entrenamiento	0 de 320
Precisión de entrenamiento	1
Error de validación	6 de 80
Precisión de validación	0.675

Tabla I: Características del árbol de decisión de clasificación posterior al ajuste del item B

Las primeras cuatro ramas del árbol se muestran en la Fig.1. En este caso se llega hasta un profundidad de 11 porque no estamos limitando el valor de gini de las hojas, la clasificación se realiza hasta que los datos de entrenamiento tengan una clasificación fiel, por eso se observa el overfitting del árbol. El primer nodo usa la posición en la vitrina para clasificar los datos, ya que este parámetro divide mejor los datos, dicho de otra forma, tiene el coeficiente de gini mayor de todos los nodos.

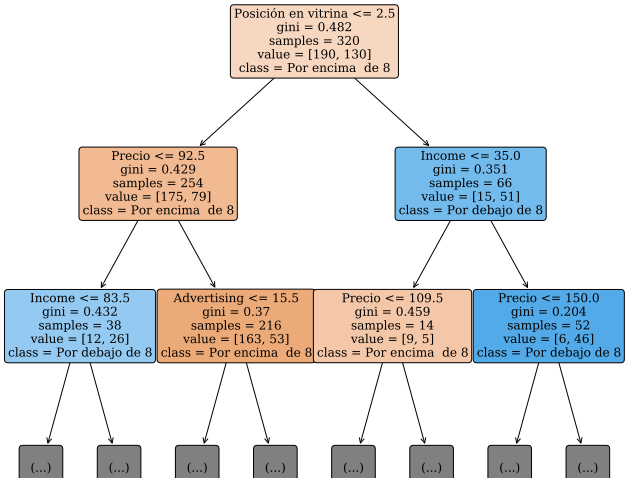

Fig. 1: Las primeras cuatro ramas de la estructura del árbol de clasificación para el item B.

Item C

El árbol instanciado en este item es el árbol de regresión, los parámetros por defecto también hacen que el árbol regresione hasta tener hojas con gini nulo. Ahora los datos de salida esperado son los datos de la columna de *Sales*.

A partir de este item en adelante, el valor de error es el error cuadrático medio entre la salida predicha por el árbol y la salida esperada. Como se ven en la Tabla II se muestran las características del árbol luego de realizar el ajuste. Nuevamente se tiene un overfitting con el conjunto de entrenamiento.

Profundidad	18
Hojas	320
Error de entrenamiento	0.0
Precisión de entrenamiento	1
Error de validación	0.073
Precisión de validación	0.247

Tabla II: Características del árbol de regresión posterior al ajuste en el item C

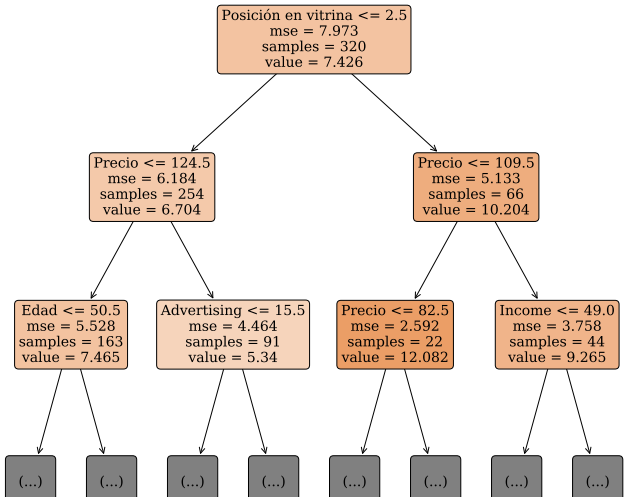

Fig. 2: Las primeras cuatro ramas de la estructura del árbol de clasificación para el item C.

A pesar del overfitting del árbol, el árbol aprende la correlación de los atributos con la cantidad de asientos. Esto se observa en la Fig.3, dado que los ejes x e y son la cantidad de asientos reales y predichos, mientras más cerca estén los puntos a la línea, mejor es la predicción del árbol.

Item E

En este ejercicio se utilizó la función `GridSearchCV`, que toma vectores con valores para los parámetros de profundidad del árbol `max_depth` y el coeficiente de *Cost-Complexity Pruning*. Este último impone un valor mínimo para el cual el nodo se considera una hoja o no, disminuyendo así el overfitting del árbol.

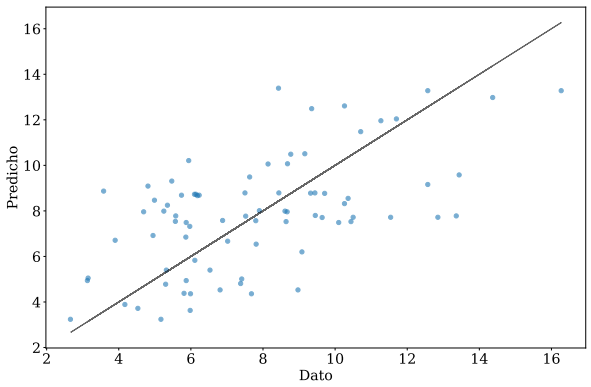

Fig. 3: Comparación entre el dato y la predicción del árbol de regresión para else item C.

Mediante la función `GridSearchCV`, se calculan los parámetros que maximizan el *score* mediante k-fold *cross validation*(CV). La función tiene un valor predeterminado de $k = 5$ que fue el utilizado.

Los parámetros óptimos obtenidos son: Profundidad igual a 5 y coeficiente de *Cost-Complexity Pruning* ó α_{ccp} igual a 0.2, donde el mejor score obtenido fue de 0.36. Posterior al entrenamiento con estos parámetros, el árbol tiene las características listadas en la Tabla III. Ahora el error de validación para este item es menor que el item B, donde de 0.07 pasa a 0.06, pero la precisión sobre los datos de validación es mayor en el caso anterior. La Fig.4 muestra la comparación entre precio predicho y real, en el mismo se que los precios predichos sigue la tendencia de los reales, a pesar de la baja precisión.

Profundidad	5
Hojas	10
Error de entrenamiento	0.009
Precisión de entrenamiento	0.60
Error de validación	0.06
Precisión de validación	0.35

Tabla III: Características del árbol de regresión posterior al ajuste en el item E

Además de la función `GridSearchCV`, se utilizó el método `cost_complexity_pruning_path` propia de la clase `DecisionTreeRegressor`, que devuelve los valores posibles de α_{cpp}. Con los mismos se calcularon los errores y profundidades a las que llega los datos de entrenamiento, y el error de los datos de validación. En la Fig.5 se muestra como varian los errores y la profundidad con α_{cpp} y se observa un mínimo en el error de validación para un valor de $\alpha_{cpp} = 0.25$, donde la red tiene una profundidad de 5 niveles.

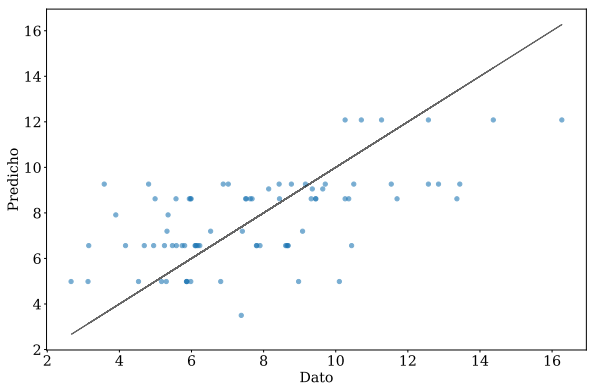

Fig. 4: Comparación entre el dato y la predicción del árbol de regresión para el item E.

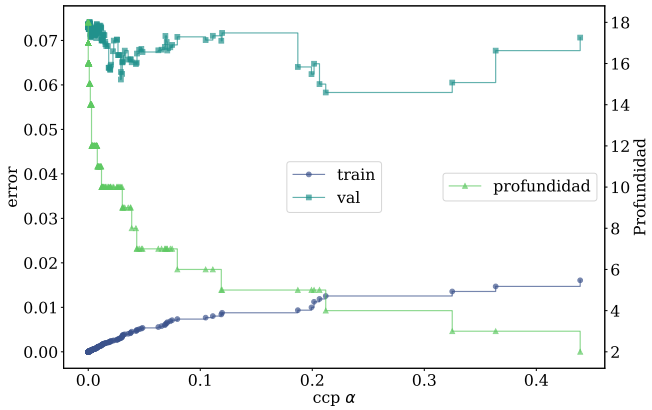

Fig. 5: Error y Profundidad de los datos de entrenamiento y error de validación en función del valor de α_{cpp}

Item F

En este item se compararon los resultados del árbol de regresión y un conjunto de árboles de decisión, implementados mediante Bagging. El mismo instancia distintos árboles para varias particiones del conjunto de datos, luego ajusta los parámetros de cada árbol y la función `predict` devuelve la media de las salida de todos los árboles. En este trabajo se optó por usar 25 árboles distintos un Bagging Forest.

Considerando que en el item anterior se obtuvo mediante `GridSearchCV` que el valor óptimo de $\alpha_{cpp} = 0.2$, se instanciaron árboles con $\alpha_{cpp} = 0.0$ y $\alpha_{cpp} = 0.2$ para comparar resultados en las Tablas IV y V. En ambas tablas se puede observar que el Bagging tiene una precisión mejor que un sólo árbol ya que el overfitting disminuyó. Otro aspecto es el error de validación, donde un sólo árbol tiene un error mayor que utilizando el emsemble.

Atributo	Simple	Bagging
CompPrice	0.1045	0.1148
Income	0.0980	0.0706
Advertising	0.0413	0.0730
Population	0.0429	0.0340
Price	0.2960	0.2946
ShelveLoc	0.2739	0.2932
Age	0.0986	0.0840
Education	0.0195	0.0279
Urban	0.0151	0.0053
US	0.0097	0.0022
Error de entrenamiento	0.0	0.0015
Precisión de entrenamiento	1	0.95
Error de validación	0.07	0.03
Precisión de validación	0.32	0.63

Tabla IV: Características de los árboles de regresión posterior al ajuste en el item F con $\alpha_{cpp} = 0$

Atributo	Simple	Bagging
CompPrice	0.0864	0.0634
Income	0.	0.0240
Advertising	0.0432	0.0771
Price	0.3588	0.3269
ShelveLoc	0.4198	0.4408
Age	0.0919	0.0678
Error de entrenamiento	0.0099	0.0077
Precisión de entrenamiento	0.60	0.70
Error de validación	0.06	0.045
Precisión de validación	0.36	0.53

Tabla V: Características de los árboles de regresión posterior al ajuste en el item F con $\alpha_{cpp} = 0.2$

Las importancias ó importancia de gini de cada atributo es número que indica cuan relevantes son esos atributos en la regresión, las mismas son un atributo para la clase del árbol utilizado en `sklearn`. Para obtener las importancias de los atributos en el Bagging, se tomó la media para cada atributo a partir de todos los árboles del emsemble. Con $\alpha_{cpp} = 0.0$, el parámetro con mayor importancia para un árbol simple en todas las ejecuciones del código resultó ser el precio del asiento, en cambio para el Bagging el parámetro más importante varía entre ejecución pudiendo ser la posición en la vitrina o el precio del asiento. Con $\alpha_{cpp} = 0.2$, como se observa en la Tabla V, hay atributos cuyo peso es nulo, esto se debe a que no son utilizados en la regresión. Además en este caso el atributos con mayor importancia resultó ser la posición en la vitrina del negocio.

Comparando las precisiones sobre la validación en el Bagging con $\alpha_{cpp} = 0.0$ y $\alpha_{cpp} = 0.2$, se observa en las tablas que el primero llega a una mejor precisión que la segunda opción. Por lo que en el Bagging Forest, se tiene una mejor generalización con $\alpha_{cpp} = 0.0$.

Item G e Item H

En este item seguimos abordando el problema de regresionar la cantidad de asientos vendidos con otros datos, utilizando distintos emsembles de árboles de regresión. En las Tabla VI se comparan las importancias, errores y precisiones para los emsembles Random Forest y AdaBoost. Para realizar los ajustes se tomaron 25 árboles en cada emsemble, sin condiciones en la profundidad y con el valor de α_{cpp} nulo.

Atributo	Random Forest	AdaBoost
CompPrice	0.1062	0.1180
Income	0.0594	0.0704
Advertising	0.0818	0.0793
Population	0.0406	0.0340
Price	0.2778	0.2929
ShelveLoc	0.2952	0.2902
Age	0.1000	0.0773
Education	0.0280	0.0274
Urban	0.0068	0.0063
US	0.0041	0.0044
Error de entrenamiento	0.0013	0.0012
Precisión de entrenamiento	0.946	0.953
Error de validación	0.0369	0.0334
Precisión de validación	0.621	0.656

Tabla VI: Comparación entre los emsembles de regresión Random Forest y AdaBoost posterior al ajuste en los items G y H con $\alpha_{cpp} = 0.0$

Comparando los errores de validación de los árboles de regresión anteriores con los emsembles de este inciso, el Random Forest y el AdaBoost tienen los menores errores en este trabajo, donde el último emsemble es el menor valor obtenido. También el AdaBoost tiene la precisión de validación más alta entre los árboles y emsembles utilizados.

Las importancias de los atributos del Random Forest, comparado con el Bagging, en cada ejecución la posición en la vitrina era la más importante, mientras que en el Bagging dependía de la ejecución si la posición o el precio era más importante. Esto varía con los datos tomados en cada fold del k-fold en el Bagging.

En la ejecución de los emsembles de este inciso, se puede definir cual es la profundidad de los árboles del emsembles o cual es la cantidad de atributos considerados para hacer la regresión. En las Figs. 6 y 7 se muestran los errores de validación en función de los parámetros mencionados anteriormente.

Para la Fig. 6, se deja libre el parámetro de profundidad de la red. La cantidad de atributo a considerar puede variar entre 1 y 10, dado que la máxima cantidad de atributos disponibles son 10. Los errores varían entre ejecuciones por lo que en la figura mencionada se presenta la media de 10 ejecuciones distintas. Se observa que hay un mínimo en la curva cuando se toman 5 atributos, y que para el AdaBoost se obtiene el menor error en este valor.

Para la Fig.7 se varía el parámetro de profundidad mientras que el número de atributos a considerar son 10, que es el predeterminado. Lo mismo que para la figura anterior, se tomó el promedio de 10 ejecuciones. Se observa que para una profundidad a partir de 5 es error permanece constante.

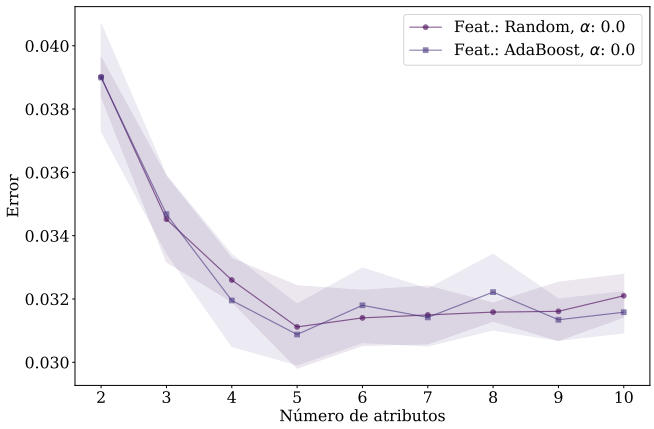

Fig. 6: Error medio de 10 ejecuciones del programa en función del número de atributos para Random Forest y AdaBoost.

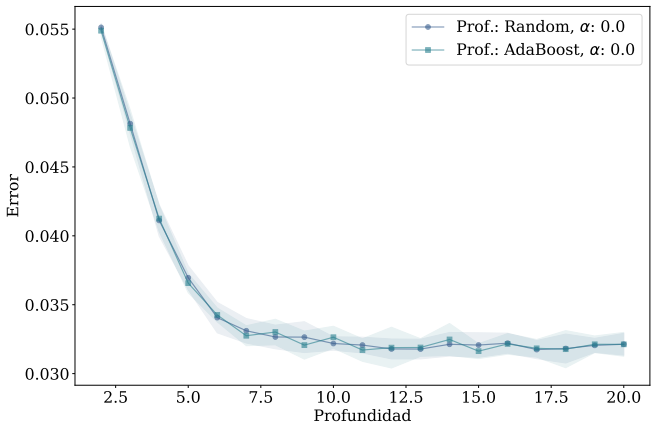

Fig. 7: Error medio de 10 ejecuciones del programa en función de la profundidad de los emsembles Random Forest y AdaBoost.