Aprendizaje Automático: Trabajo práctico 1

M. Sc. Saúl Calderón Ramírez
Instituto Tecnológico de Costa Rica,
Escuela de Ingeniería en Computación
PAttern Recongition and MAchine Learning Group (PARMA-Group)

15 de marzo de 2022

Fecha de entrega: Domingo 24 de Abril.
Entrega: Un archivo .zip con el código fuente LaTeX o Lyx, el pdf, y un notebook Jupyter, debidamente documentado, con una función definida por ejercicio. A través del TEC-digital.
Modo de trabajo: Grupos de 3 personas.

Resumen

En el presente trabajo práctico se introducir los arboles de decision por medio de su implementación para resolver un problema práctico.

1. Implementación de la clasificación multi-clase con árboles de decisión

1. El conjunto de datos disponible en `https://www.kaggle.com/argonalyst/sao-paulo-real-estate-sale-rent-april-2019` corresponde a registros de propiedades en venta en la ciudad de Sao Paulo, Brasil, en Abril del 2019. El objetivo del modelo a construir es estimar el valor de la propiedad. **Utilizar la version recortada del dataset, disponible en la carpeta del curso**. La Figura 1 despliega una muestra del conjunto de datos.

   a) Para el modelo a implementar, se plantea considerar los atributos numéricos de *Rooms, Size, Toilets y Parking*. Implemente usando Pandas y/o Pytorch, la lectura del dataset usando solamente tales atributos. ***No use ciclos***.

   b) Como preprocesamiento del conjunto de datos, es necesario discretizar la variable a estimar y (precio, en reales brasileños), en 4 categorias:

      1) $900000 < y$, categoría 4

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Price	**Condo**	**Size**	**Rooms**	**Toilets**	**Suites**	**Parking**	**Elevator**	**Furnished**	**Swimming Pool**	**New**	**District**	**Negotiation Type**	**Property Type**	**Latitude**	**Longitude**
10000000	0	343	4	7	4	5	0	0	0	0	Iguatemi/SÃ£o Paulo	sale	apartment	-23.5855	-46.6817
9979947	0	343	4	6	4	5	1	0	1	0	Iguatemi/SÃ£o Paulo	sale	apartment	-23.5855	-46.6817
8500000	7200	420	4	6	4	4	1	0	1	0	Jardim Paulista/SÃ£o Paulo	sale	apartment	-23.564	-46.6609
8039200	0	278	4	7	4	4	1	1	1	0	Vila Olimpia/SÃ£o Paulo	sale	apartment	-23.5965	-46.6806
8000000	0	278	4	5	3	5	1	0	1	0	Vila Olimpia/SÃ£o Paulo	sale	apartment	-23.5965	-46.6806
8000000	0	269	4	5	4	4	1	0	1	0	Itaim Bibi/SÃ£o Paulo	sale	apartment	-23.5883	-46.6808
7765616	0	279	4	6	4	4	1	0	1	0	Iguatemi/SÃ£o Paulo	sale	apartment	-23.5855	-46.6817
7559420	0	377	3	4	3	4	1	0	1	0	Alto de Pinheiros/SÃ£o Paulo	sale	apartment	-23.5495	-46.7174
7559420	0	377	3	4	3	4	0	0	1	0	Alto de Pinheiros/SÃ£o Paulo	sale	apartment	-23.5495	-46.7174
7521000	2500	377	6	7	6	4	0	0	1	0	Alto de Pinheiros/SÃ£o Paulo	sale	apartment	-23.5495	-46.7174
7500000	0	275	4	5	4	4	1	0	1	0	Iguatemi/SÃ£o Paulo	sale	apartment	-23.5965	-46.6806
7500000	7428	415	4	5	4	5	0	0	1	0	Jardim Paulista/SÃ£o Paulo	sale	apartment	-23.5686	-46.6553
7200000	8920	387	4	5	4	4	0	0	0	0	Alto de Pinheiros/SÃ£o Paulo	sale	apartment	-23.5572	-46.6928
7080000	0	275	4	5	4	5	0	0	1	0	Iguatemi/SÃ£o Paulo	sale	apartment	-23.5845	-46.6823
7000000	4500	400	4	5	4	5	0	1	1	0	Alto de Pinheiros/SÃ£o Paulo	sale	apartment	-23.5605	-46.7009
6950000	5000	302	4	5	4	5	0	0	1	0	Moema/SÃ£o Paulo	sale	apartment	-23.5984	-46.6664
6900000	7000	321	3	4	3	5	0	0	1	0	Itaim Bibi/SÃ£o Paulo	sale	apartment	-23.5945	-46.6804
6800000	0	286	4	5	4	4	1	0	1	0	Alto de Pinheiros/SÃ£o Paulo	sale	apartment	-23.5592	-46.6994
6750000	0	275	4	6	4	4	1	0	1	0	Iguatemi/SÃ£o Paulo	sale	apartment	-23.5855	-46.6817

Figura 1: Muestra del conjunto de datos a utilizar.

2) $580000 < y < 900000$, categoría 3
3) $400000 < y < 580000$, categoría 2
4) $y < 400000$, categoría 1

Utilice pytorch para realizar la convención correspondiente. ***No use ciclos***. Guarde las etiquetas como la ultima columna de la variable *dataset_torch* a retornar en la funcion *read_dataset*

2. Para resolver el problema de clasificar las propiedades según los rangos de precio antes definidos. Para ello utilizará el código provisto en el *notebook* de *Jupyter*.

   a) El código provisto define las clases *CART* y Node_*CART*, las cuales permiten construir un CART binario. Cada nodo del arbol tiene atributos como el *feature*, el umbral, y el coeficiente de *gini* de la partición definida en tal nodo. Además define el atributo *dominant_class* para el nodo, el cual es el resultado de calcular la clase con mayor cantidad de apariciones en la partición que define al nodo. Finalmente el código incluye la funcionalidad para generar un archivo *xml* (el cual se puede abrir en cualquier navegador web), para representar fácilmente el árbol.

   b) **(10 puntos)** Implemente el método *calculate_gini(data_partition_torch, num_classes = 4)*, el cual calcule el coeficiente de gini para el conjunto de datos recibido en un tensor de *pytorch*. Para ello utilice la definición inidicada en el material del curso:

$$E_{\text{gini},\rho}\left(\tau_d\right) = 1 - \sum_{k=1}^{K} a_k^2.$$

   a) **(20 puntos)** Implemente el método *select_best_feature_and_thresh(data_torch, list_features_selected = [], num_classes = 4)*, de la clase *Node_CART*. Este método recibe como parámetros el conjunto de datos en un tensor

tipo torch a analizar, además de la lista de *features* seleccionados hasta ahora (para ignorarlos en futuras pruebas), y la cantidad de clases a discriminar. El método debe probar de forma extensiva todos los posibles features y sus correspondientes umbrales en los datos recibidos, hasta dar con el menor coeficiente ponderado de gini (o la minima entropía, dependiendo de la función de error a utilizar) (ignorando los features en *list_features_selected = []*). **Utilice indexación lógica para evitar al máximo el uso de estructuras de repetición tipo *for***. Solamente puede usar estructuras de repetición para iterar por los *features* y posibles umbrales dentro del conjunto de datos. Recuerde que para evaluar una posible partición, es necesario calcular el coeficiente de gini ponderado sugerido para decidir el feature y umbral óptimos es:

$$\overline{E}_{\text{gini}}\left(\tau_d, d\right) = \frac{n_i}{n} E_{\text{gini}}\left(D_i\right) + \frac{n_d}{n} E_{\text{gini}}\left(D_d\right).$$

b) **(5 puntos)** Implemente la función *test_CART* la cual evalúe un CART previamente entrenado para un conjunto de datos D representado en un tensor. Calcule la tasa de aciertos (*accuracy*), definida como:

$$a = \frac{c}{n}$$

donde c corresponde a las estimaciones correctas, para tal conjunto de datos y retornela.

c) **(5 puntos)** Implemente la función *partition_validation(dataset_torch, max_CART_depth, num_splits)* la cual genere *num_splits* particiones de forma aleatoria del conjunto de datos recibido en *dataset_torch* (matriz de datos donde la ultima fila son las etiquetas). Para ello utilice la funición *ShuffleSplit* del paquete *sklearn.model_selection*

2. Evaluación del CART

1. **(20 puntos)** Evalúe el CART implementado con el conjunto de datos completo provisto usandolo como conjunto de datos de entrenamiento y prueba. Reporte la tasa de aciertos obtenida e incluya el código de la evaluación. Pruebe con una profundidad máxima de 2 y 3 nodos, siempre con minimo 2 observaciones por hoja.

2. **(10 puntos)** Para una profundidad máxima de 2 y 3 nodos: evalúe el CART implementado usando 10 particiones aleatorias del conjunto de datos, con un 70 % del conjunto de datos como conjunto de datos de entrenamiento, y el restante 30 % como conjunto de datos de prueba. Reporte una tabla con la tasa de aciertos de cada una de las 10 corridas, y el promedio y desviación estándar para las 10 corridas.

3. Implementación de el Boque Aleatorio (Random Forest)

1. Implemente el modelo de *Random Forests,* en la clase del mismo nombre. Tal clase tiene como atributos la cantidad de árboles que conforman el bosque, y la profundidad máxima de todos ellos, según el código provisto.

   a) **(10 puntos)** Implemente la función *train_random_forest* la cual realice K particiones aleatorias disjuntas, para entrenar con cada una de ellas un CART, según se implementó anteriormente. Para tal implementación use la función *KFold* del paquete *sklearn.model_selection.*

   b) **(5 puntos)** Implemente la función *evaluate_random_forest(input_torch),* la cual para una observación *input_torch,* calcule la estimación del random forest, usando el esquema de votación máxima comentado en clase.

   c) **(5 puntos)** Implemente la función *test_random_forest(testset_torch)*, la cual evalúe la tasa de aciertos *(accuracy)* del *random forest* para todo un conjunto de datos de test, representado en la matriz *testset_torch* (la cual tiene una columna por feature, y la última columna corresponde a la etiqueta de tal observación en esa fila).

      1) Pruebe y reporte la tasa de aciertos del *random forest* con una profundidad de 3 nodos y 5 CARTs. Comente los resultados respecto a los resultados anteriores.

4. Evaluación del Random Forest

1. **(10 puntos)** Para una profundidad máxima de 3 nodos con 3 y 5 CARTs: evalúe el *random forest* implementado usando 10 particiones aleatorias del conjunto de datos, con un 70 % del conjunto de datos como conjunto de datos de entrenamiento, y el restante 30 % como conjunto de datos de prueba. Reporte una tabla con la tasa de aciertos de cada una de las 10 corridas, y el promedio y desviación estándar para las 10 corridas. Comente los resultados respecto a los resultados anteriores.

5. CART para regresion

1. **(20 puntos extra)** Implemente una modificacion del CART para realizar regresion.

   a) Investigue que cambios es necesario hacer (funcion de error, valor estimado por nodo, algoritmo de optimizacion, etc.) para hacer que

el modelo estime un valor continuo. Documente tales modificaciones, citando las referencias correspondientes, en no mas de 3 parrafos.

b) Implemente la variante del CART para regresion segun lo investigado.

c) Para una profundidad máxima de 2 y 3 nodos: evalúe el CART de regresion implementado usando 10 particiones aleatorias del conjunto de datos, con un 70 % del conjunto de datos como conjunto de datos de entrenamiento, y el restante 30 % como conjunto de datos de prueba. Reporte una tabla con la tasa de aciertos de cada una de las 10 corridas, y el promedio y desviación estándar para las 10 corridas.

d) Compare los resultados del CART para clasificacion con el CART para regresion y comente.