Documentación

En dicha práctica debíamos implementar los algoritmos de aprendizaje supervisado y no supervisado descritos en la asignatura (Arboles de Decisión, Aprendizaje Bayesiano y K-Means Clustering).

Árboles de decisión

Empezando por los Arboles de decisión disponíamos de un fichero "treepredict.py" con una serie de funciones ya implementadas.

Los árboles de decisión son uno de los más simples métodos de aprendizaje máquina. Son un método totalmente transparente de la clasificación de observaciones.

Cuando tienes un árbol de decisión, Es muy fácil ver cómo toma todas sus decisiones. Simplemente siga el camino hacia abajo del árbol que responde a cada pregunta correctamente y eventualmente llegará a una respuesta. El rastreo desde el nodo donde terminó da una justificación para la clasificación final.

El primer paso es crear una representación de un árbol.

La clase decisiónnode, → Representa cada nodo en el árbol.

UniqueCounts → Dicha funcion encuentra todos los diferentes resultados posibles y los devuelve como un diccionario de cuántas veces cada uno aparece. Esto es utilizado por las otras funciones para calcular cómo se mezcla un conjunto. Hay algunas métricas diferentes para medir esto, y dos se considerarán aquí: Gini impureza y entropía.

Giniimpurity →Esta función calcula la probabilidad de cada resultado posible dividiendo el número de veces que el resultado ocurre por el número total de filas en el conjunto. A continuación, suma los productos de todas estas probabilidades.

La función de entropía (Entropy()) calcula la frecuencia de cada elemento (el número de veces que aparece dividido por el número total de filas)

BuildTree → Esta función se llama primero con la lista de filas. Se pasa por todas las columnas (excepto la última, que tiene el resultado en ella), encuentra todos los valores posibles para esa columna y divide el conjunto de datos en dos nuevos subconjuntos. Calcula la entropía media ponderada para cada par de nuevos subconjuntos multiplicando la entropía de cada conjunto por la fracción de los ítems que terminaron en cada conjunto y recuerda qué par tiene la entropía más baja. Si el mejor par de subconjuntos no tiene una entropía media ponderada más baja que el conjunto actual, esa rama termina y los conteos de los posibles resultados se almacenan. De lo contrario, buildtree se llama en cada conjunto y se agregan al árbol. Los resultados de las llamadas

en cada subconjunto se adjuntan a las ramas Verdadero y Falso de los nodos, eventualmente construyendo un árbol completo. Iterativamente lo realizamos por medio de un Stack.

Printtree → Toma un árbol devuelto por el árbol de la estructura y lo atraviesa, y sabe que ha llegado al final de una rama cuando alcanza el nodo con resultados. Hasta que llega a ese punto, imprime los criterios para las ramas Verdadero y Falso y llama printtree en cada uno de ellos, cada vez que aumenta la cadena de sangría.

Classify → Esta función atraviesa el árbol de la misma manera que printtree. Después de cada llamada, comprueba si ha llegado al final de esta rama buscando resultados. De lo contrario, evalúa la observación para ver si la columna coincide con el valor. Si lo hace, llama a clasificar de nuevo en la rama True; Si no, llama a clasificar en la rama Falsa.

Prune → Cuando esta función se llama en el nodo raíz, recorrerá todo el camino hasta el árbol a los nodos que sólo tienen nodos hoja como niños. Creará una lista combinada de resultados de ambas hojas y probará la entropía. Si el cambio de entropía es menor que el parámetro mingain, las hojas se borrarán y todos sus resultados se moverán a su nodo padre. El nodo combinado se convierte entonces en un posible candidato para supresión y fusión con otro nodo.

MdClassify → Es una simple modificación de classificar. La única diferencia es al final donde, si faltan los datos importantes, se calculan los resultados para cada rama y luego se combinan con sus respectivas ponderaciones.

Test_performace → Dados dos sets, con uno creamos el Árbol de decisión i con el otro comprobamos que es correcto, una vez realizado esto calculamos el porcentaje.

Aprendizaje Bayesiano

De Aprendizaje Bayesiano disponemos de un fichero “docclass.py” con una serie de funciones ya implementadas en las transparencias de la teoría.

El aprendizaje bayesiano es un método de aprendizaje supervisado, que sus objetivos son:

Clasificar los documentos de acuerdo a su contenido, filtrar el spam, ya que el correo no deseado no sólo afecta al correo electrónico, sino que también afecta a las webs que se han vuelto más interactivas (Solicitud de comentarios de los usuarios y la creación de contenido). Dichas webs/aplicaciones necesitan una estrategia para eliminar el spam.

La idea es aplicar un algoritmo de aprendizaje capaz de clasificar documentos en categorías, uno de los cuales es el spam

En dicho fichero se nos pide implementar la función “classify” que nos permita clasificar un elemento. Dicha función devuelve la Categoría o un valor predeterminado cuando la categoría no puede ser determinada.

K-Means Clustering

De K-Means Clustering disponemos de un fichero “clusters.py” con una serie de funciones ya implementadas en las transparencias de la teoria.

El agrupamiento jerárquico y el agrupamiento en K-means son técnicas de aprendizaje sin supervisión, lo que significa que no requieren ejemplos de datos de entrenamiento porque no intentan hacer predicciones.

Mientras que el agrupamiento jerárquico crea un árbol de los elementos, el K-means Clustering realmente separa los datos en grupos distintos. También requiere que decida cuántos grupos desea antes de que comience a ejecutarse el algoritmo.

Resultados

Disponiamos de la web “http://archive.ics.uci.edu/ml/datasets.html” para encontrar diversas bases de datos con multitud de conjuntos de datos para realizar distintas pruebas.

En mi caso he utilizado dos data base de dicha web.

Car Evaluation Data Set

{'acc': 384, 'unacc': 1209, 'good': 69, 'vgood': 65}

Balance Scale Data Set

{1: 124, 2: 125, 3: 125, 4: 125, 5: 125}