Ernesto Fernandez - Federico Fioritto - Javier Stabile
Grupo 6
MAA 2018

Informe Laboratorio 3 MAA

Ejercicio 7

Naive Bayes

Para el algoritmo de Naive Bayes se decidió extender según cuatro enfoques, los cuales combinándolos de diferentes formas generan dieciséis variantes del algoritmo.
Las variantes utilizadas son:

1. Tratamiento de valores desconocidos.
    a. Eliminación de la instancia con valor desconocido.
    b. Elección de atributo más común para valor desconocido.
2. Tratamiento de valores numéricos:
    a. Discretización de valores numéricos.
    b. No discretización de valores numéricos.
3. Distribución normal.
    a. Se utiliza distribución normal para el cálculo de la probabilidad condicional para los atributos numéricos.
    b. No se utiliza distribucion normal.
4. Cálculo de probabilidad con m-estimador.
    a. Se utiliza m-estimador.
    b. No se utiliza m-estimador.

1. a. Se asume que una instancia con valor desconocida es ruido y se elimina para no afectar el conjunto y las posteriores clasificaciones.
1. b. Se asume que una instancia con valor desconocida es probablemente igual a la mayoría corrigiendo así su valor, para conjuntos donde los atributos son mayoritariamente iguales es una buena decisión.

2. a. Para este ejercicio se discretiza la edad, pasando los años a décadas, de esta forma se logra disminuir la cantidad de valores distintos para “edad”, esta decisión implica que se consideren cercanos las instancias con “edad” en la misma década. Se observa que edades cercanas pero en diferentes décadas serán consideradas lejanas(29,30 -> 2,3), mientras que edades en los extremos de una década serán consideradas cercanas (20,29 -> 2,2).
2. b. En caso de no discretizar se mantienen las edades originales de las instancias, lo que puede derivar en obtener probabilidades muy bajas para algunos valores.

3. a. En el cálculo de probabilidad condicional en valores numéricos (“edad”) se asume una distribución normal, obteniendo mayor probabilidad las edades con valor cercano a la media de los conjuntos con distintas clasificaciones. Por ejemplo, Si el valor edad de una instancia a clasificar es cercano a la media de edades de los datos clasificados como positivos, existe más probabilidad de clasificar a la instancia como positiva también, análogo para el caso negativo.

La probabilidad condicional se calcula con la siguiente fórmula dentro del código:

$$p(x = v \mid C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}}\, e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}}$$

3. b. No se asume se asume una distribución normal para el valor “edad”, por lo tanto todas las probabilidades son calculada a partir de la frecuencia de aparición de los atributos perdiendo la información de la cercanía de edades.

4. a. Se utiliza un m-estimador para el caso en que no haya ejemplos para algún atributo, este caso puede provocar que al calcular la probabilidad de ese atributo, la misma de cero y anule todo el término clasificador. Utilizando un m-estimador, la probabilidad no es cero.
4. b. No se utiliza un m-estimador, esto puede generar que algunos términos clasificadores tengan cero probabilidad, en un atributo con muchos valores como por ejemplo uno numérico y más en uno continuo, sino se repite el mismo valor en el conjunto de entrenamiento, probablemente la clasificación sea al azar porque cualquier término clasificador será cero.

Implementación de Naive Bayes

La implementación de Naive Bayes se encuentra en el archivo **naive_bayes.py** y en él se encuentra la estructura necesaria para poder realizar clasificaciones y una función clasificadora que recibe una instancia y devuelve la clasificación más probable según Naive Bayes.

La estructura anteriormente mencionada almacena la frecuencia con la que se repiten los atributos dado una clasificación de instancia. Esto se representa como tres niveles de diccionarios como en el ejemplo siguiente:

values_frecuency[Atributo][Valor_Atributo][Clasificación]

Un ejemplo puede ser values_frecuency[Tiempo][Soleado][NO] = 10, lo que indica que existen 10 instancias clasificadas positivas en el conjunto de entrenamiento con Tiempo=Soleado.
También se genera una estructura que almacena la cantidad de instancias con cada clasificación:

target_values_frecuency[Clasificación]

Por último se genera una estructura que contiene la media y la varianza de los valores del atributo “edad”, para poder calcular la probabilidad condicional con distribución normal:

normalize_age_probabilities[“Media”][Clasificación]
normalize_age_probabilities[“Varianza”][Clasificación]

Todas las estructuras generadas se cargan con un solo procesamiento del conjunto de entrenamiento. La función clasificadora usa esos datos cargados

Ernesto Fernandez - Federico Fioritto - Javier Stabile
Grupo 6
MAA 2018

KNN

Se implementa el algoritmo k-nearest-neighbor el cual clasifica una instancia basándose en los k ejemplos más cercanos. La distancia entre 2 instancias se define como la sumatoria de las distancias de cada uno de los atributos, para atributos numéricos la distancia se calcula utilizando la distancia euclidiana, y para los no numéricos la distancia se define como d=1 si los valores del atributo de ambas instancias difieren, y d=0 cuando son iguales, esto se definió así porque con el simple hecho de mirar los datos, no logramos encontrar algo que indique un patrón entre dos valores de un atributo que nos indique si una instancia es más o menos cercana a otra, sumado a que la mayoría de los atributos son binarios.

Las variantes utilizadas en el ejercicio para el algoritmo son:

1. Tratamiento de valores desconocidos:
    a. Eliminación de la instancia con valor desconocido
    b. Elección de atributo más común para valor desconocido
2. Tratamiento de valores numéricos:
    a. Estandarización
    b. Min-max
3. Tres valores diferentes para k (1, 3 y 7)
4. Clasificar la instancia asignando pesos a los k ejemplos más cercanos

1. a. Se asume que una instancia con valor desconocida es ruido y se elimina para no afectar el conjunto y las posteriores clasificaciones.
1. b. Se asume que una instancia con valor desconocida es probablemente igual a la mayoría corrigiendo así su valor, para conjuntos donde los atributos son mayoritariamente iguales es una buena decisión.

2. K-nn maneja bien los atributos numéricos, pero por cómo definimos las distancias entre dos instancias, si no los llevamos a otra escala, estos afectan mucho a la función distancia, en comparación con los no numéricos.
Para atacar este problema, se utilizan dos técnicas posibles de escalamiento:

a. Estandarización: los valores son reescalados para que tengan las propiedades de una distribución normal estándar ($\mu=0$ y $\sigma=1$), donde μ es la media y σ la varianza
b. Min-max: se re escalan los valores a un rango entre 0 y 1

En cualquiera de ambos casos, el re escalamiento se ejecuta en tiempo de entrenamiento, y al clasificar una nueva instancia ésta se re escala utilizando los mismos parámetros (μ y σ o min y max) que se calcularon previamente, es decir no se modifican.

3. Variar k indica cuantos ejemplos más cercanos se toman en cuenta para clasificar a la instancia desconocida, entonces por ejemplo si se tiene k = 1, el resultado de la instancia desconocida se clasifica igual al resultado del ejemplo más cercano a ésta.

4. Una mejora clara para knn es considerar que los ejemplos más cercanos a la instancia desconocida, son más relevantes a la hora de clasificar, y por ejemplo una buena manera de hacerlo (y como está implementado) es calcular el peso de cada instancia cercana como el inverso del cuadrado de la distancia entre las dos instancias:

$$w_i \equiv \frac{1}{d(x_q, x_i)^2}$$

Implementación de K Nearest Neighbor

La implementación de K Nearest Neighbor se encuentra en el archivo **KNN.py** y en él se encuentran las funciones necesarias para clasificar una instancia. Cada vez que se clasifica una instancia se recorre todo el conjunto de entrenamiento y se ejecuta la función distancia para cada ejemplo para encontrar los k ejemplos más cercanos a la instancia desconocida, por todas estas iteraciones, el algoritmo knn demora un tiempo razonable en ejecutar.

Parte a)

Aplicaciones de los algoritmos ID3, NB y KNN al ejemplo visto en el teórico.
El conjunto de datos que se utiliza para entrenamiento es el siguiente:

#Ej	Tiempo	Temperatura	Humedad	Viento	Juega
1	Soleado	Caluroso	Alta	Suave	No
2	Soleado	Caluroso	Alta	Fuerte	No
3	Nuboso	Caluroso	Alta	Suave	Sí
4	Lluvioso	Templado	Alta	Suave	Sí
5	Lluvioso	Frío	Normal	Suave	Sí
6	Lluvioso	Frío	Normal	Fuerte	No
7	Nuboso	Frío	Normal	Fuerte	Sí
8	Soleado	Templado	Alta	Suave	No
9	Soleado	Frío	Normal	Suave	Sí
10	Lluvioso	Templado	Normal	Suave	Sí
11	Soleado	Templado	Normal	Fuerte	Sí
12	Nuboso	Templado	Alta	Fuerte	Sí
13	Nuboso	Caluroso	Normal	Suave	Sí
14	Lluvioso	Templado	Atla	Fuerte	No

Los tres algoritmos se aplican en sus versiones sin extensiones.
La instancia a clasificar es la siguiente

<soleado, frío, alta, fuerte>

Naive Bayes

Se clasifica la instancia dada con el algoritmo de Naive Bayes, dando un resultado idéntico al teórico, clasificando como NO

ID3

El algoritmo ID3 con ejemplo del teórico genera el árbol de la figura 1:

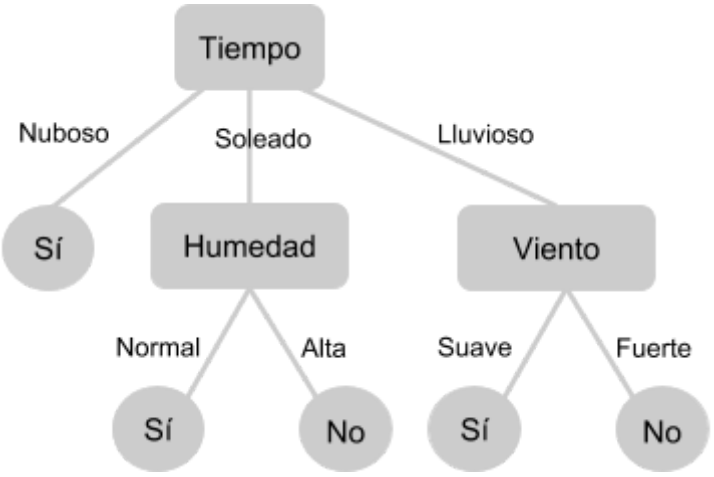


Figura 1

La validación de la instancia del ejemplo con el árbol anterior da como resultado “NO”, por lo tanto, se obtiene el mismo resultado que con Naive Bayes.

KNN

Para el caso del algoritmo KNN se utilizan tres diferentes k (1, 3, 7).
Para el caso de k = 1 y k = 3 el resultado de clasificar con KNN es idéntico al de los algoritmos anteriores dando “NO” como resultado.
Para el caso de k = 7 a diferencia de los resultados anteriores, la instancia es clasificada como “YES”. La diferencia de esta última clasificación puede ser producto de las pocas instancias del conjunto inicial, lo que genera que sus vecinos más cercanos, en este caso siete, no necesariamente sean cercanos, pudiendo tener clasificaciones lejanas.

La ejecución de los algoritmos anteriores se encuentran en el archivo main.py adjunto a la entrega.

Ernesto Fernandez - Federico Fioritto - Javier Stabile
Grupo 6
MAA 2018

Parte b)

Al inicio de este documento, en la introducción de ambos algoritmos se explican las distintas estrategias utilizadas para manejar tanto atributos numéricos como valores faltantes.

Parte c)

Naive Bayes

Para Naive Bayes se tienen 16 posibles variantes del algoritmo (2 posibilidades para valores desconocidos, 2 para valores numéricos, 2 para el uso de distribución normal en “age” y para el uso de m-estimador o no).

Entonces para la validación cruzada se ejecutan los casos:

1) Usa most-common, no discretiza, usa m-estimate y no usa distribución normal.
2) Descarta ejemplos, no discretiza, usa m-estimate y no usa distribución normal.
3) Usa most-common, discretiza, usa m-estimate y no usa distribución normal.
4) Descarta ejemplos, discretiza, usa m-estimate y no usa distribución normal.
5) Usa most-common, no discretiza, usa m-estimate y usa distribución normal.
6) Descarta ejemplos, no discretiza, usa m-estimate y usa distribución normal.
7) Usa most-common, discretiza, usa m-estimate y usa distribución normal.
8) Descarta ejemplos, discretiza, usa m-estimate y usa distribución normal.
9) Usa most-common, no discretiza, no usa m-estimate y no usa distribución normal.
10) Descarta ejemplos, no discretiza, no usa m-estimate y no usa distribución normal.
11) Usa most-common, discretiza, no usa m-estimate y no usa distribución normal.
12) Descarta ejemplos, discretiza, no usa m-estimate y no usa distribución normal.
13) Usa most-common, no discretiza, no usa m-estimate y usa distribución normal.
14) Descarta ejemplos, no discretiza, no usa m-estimate y usa distribución normal.
15) Usa most-common, discretiza, no usa m-estimate y usa distribución normal.
16) Descarta ejemplos, discretiza, no usa m-estimate y usa distribución normal.

La variante que retorne menor cantidad de errores, es la que se utilizará para la ejecución de la parte C) 2. del ejercicio (validación hold-out) y para comparar con ID3 y con la variante que retorne menor cantidad de errores del algoritmo KNN.

Ernesto Fernandez - Federico Fioritto - Javier Stabile
Grupo 6
MAA 2018

La *tabla 1-Naive Bayes* indica la tasa de errores para cada variante del algoritmo en cada iteración de validación cruzada y además el porcentaje de errores del promedio de cada iteración.

Tabla 1-Naive Bayes

#	Iter 1	Iter 2	Iter 3	Iter 4	Iter 5	Iter 6	Iter 7	Iter 8	Iter 9	Iter10	%E
1	0.000	0.070	0.035	0.018	0.036	0.071	0.018	0.107	0.036	0.071	4.624%
2	0.000	0.060	0.040	0.000	0.061	0.061	0.020	0.123	0.020	0.082	4.674%
3	0.018	0.105	0.053	0.036	0.036	0.071	0.018	0.071	0.018	0.035	4.612%
4	0.020	0.100	0.060	0.040	0.041	0.061	0.020	0.102	0.000	0.041	4.853%
5	0.035	0.105	0.035	0.036	0.036	0.071	0.018	0.071	0.018	0.036	4.612%
6	0.020	0.100	0.060	0.020	0.041	0.061	0.020	0.061	0.000	0.041	4.244%
7	0.018	0.105	0.052	0.036	0.036	0.071	0.018	0.071	0.018	0.036	4.611%
8	0.020	0.100	0.060	0.040	0.041	0.061	0.020	0.041	0.000	0.041	4.241%
9	0.018	0.140	0.053	0.054	0.089	0.071	0.107	0.071	0.071	0.071	7.283%
10	0.020	0.120	0.040	0.040	0.082	0.102	0.061	0.123	0.061	0.082	7.302%
11	0.035	0.123	0.053	0.054	0.054	0.071	0.054	0.071	0.054	0.054	6.212%
12	0.040	0.120	0.060	0.060	0.061	0.082	0.041	0.102	0.041	0.061	6.678%
13	0.053	0.123	0.035	0.054	0.054	0.071	0.054	0.071	0.036	0.054	6.034%
14	0.040	0.120	0.060	0.040	0.061	0.082	0.041	0.061	0.020	0.061	5.865%
15	0.035	0.123	0.053	0.054	0.054	0.071	0.054	0.071	0.036	0.054	6.033%
16	0.040	0.120	0.060	0.060	0.061	0.082	0.041	0.061	0.020	0.061	6.065%

Como se puede apreciar en la tabla, la variante de Naive Bayes que obtuvo mayor desempeño fue la número 8, esta consistía en descartar los ejemplos que tuvieran valores desconocidos, discretizar el valor numérico “edad” o “age”, usar un m-estimador y asumir que el atributo “age” presenta una distribución normal.

Hipótesis que explique la variante ganadora:

- Los valores desconocidos aportan ruido al conjunto, eliminandolos se mejora la clasificación.
- Atributos numéricos con muchos valores empeoran la clasificación, discretizando se logra disminuir la cantidad de valores posibles.

- Existen atributos poco frecuentes que anulan el término clasificador, utilizando el m-clasificador se soluciona esto.
- La edad se modela bien con distribución normal (las instancias de igual clasificación tiene edades parecidas)..

KNN

Para KNN se tienen 24 posibles variantes del algoritmo (2 posibilidades para valores desconocidos, 2 para valores numéricos, 3 variantes de k y usar pesos o no).

Entonces para la validación cruzada se ejecutan los casos:

1) k = 1, usa most-common para atributos faltantes, usa pesos y estandarización.
2) k = 1, usa most-common para atributos faltantes, usa pesos y min-max.
3) k = 1, usa most-common para atributos faltantes, NO usa pesos y estandarización.
4) k = 1, usa most-common para atributos faltantes, NO usa pesos y min-max.
5) k = 1, descarta ejemplos con atributos faltantes, usa pesos y estandarización.
6) k = 1, descarta ejemplos con atributos faltantes, usa pesos y min-max.
7) k = 1, descarta ejemplos con atributos faltantes, NO usa pesos y estandarización.
8) k = 1, descarta ejemplos con atributos faltantes, NO usa pesos y min-max.
9) k = 3, usa most-common para atributos faltantes, usa pesos y estandarización.
10) k = 3, usa most-common para atributos faltantes, usa pesos y min-max.
11) k = 3, usa most-common para atributos faltantes, NO usa pesos y estandarización.
12) k = 3, usa most-common para atributos faltantes, NO usa pesos y min-max.
13) k = 3, descarta ejemplos con atributos faltantes, usa pesos y estandarización.
14) k = 3, descarta ejemplos con atributos faltantes, usa pesos y min-max.
15) k = 3, descarta ejemplos con atributos faltantes, NO usa pesos y estandarización.
16) k = 3, descarta ejemplos con atributos faltantes, NO usa pesos y min-max.
17) k = 7, usa most-common para atributos faltantes, usa pesos y estandarización.
18) k = 7, usa most-common para atributos faltantes, usa pesos y min-max.
19) k = 7, usa most-common para atributos faltantes, NO usa pesos y estandarización.
20) k = 7, usa most-common para atributos faltantes, NO usa pesos y min-max.
21) k = 7, descarta ejemplos con atributos faltantes, usa pesos y estandarización.
22) k = 7, descarta ejemplos con atributos faltantes, usa pesos y min-max.
23) k = 7, descarta ejemplos con atributos faltantes, NO usa pesos y estandarización.
24) k = 7, descarta ejemplos con atributos faltantes, NO usa pesos y min-max.

La variante que retorne menor cantidad de errores, es la que se utilizará para la ejecución de la parte C) 2. del ejercicio (validación hold-out) y para comparar con ID3 con la variante que retorne menor cantidad de errores del algoritmo Naive Bayes.

Ernesto Fernandez - Federico Fioritto - Javier Stabile
Grupo 6
MAA 2018

La *tabla 1-KNN* indica la tasa de errores para cada variante del algoritmo en cada iteración de validación cruzada y además el porcentaje de errores del promedio de cada iteración.

Tabla 2-KNN

#	Iter 1	Iter 2	Iter 3	Iter 4	Iter 5	Iter 6	Iter 7	Iter 8	Iter 9	Iter10	%E
1	0,035	0,105	0,053	0,071	0,036	0,071	0,107	0,125	0,054	0,018	0,068
2	0,053	0,105	0,035	0,071	0,036	0,071	0,089	0,143	0,054	0,000	0,066
3	0,035	0,105	0,053	0,071	0,036	0,071	0,107	0,125	0,054	0,018	0,068
4	0,053	0,105	0,035	0,071	0,036	0,071	0,089	0,143	0,054	0,000	0,066
5	0,040	0,100	0,060	0,060	0,041	0,122	0,122	0,143	0,061	0,041	0,079
6	0,060	0,100	0,040	0,060	0,041	0,102	0,102	0,163	0,061	0,020	0,075
7	0,040	0,100	0,060	0,060	0,041	0,122	0,122	0,143	0,061	0,041	0,079
8	0,060	0,100	0,040	0,060	0,041	0,102	0,102	0,163	0,061	0,020	0,075
9	0,053	0,123	0,018	0,036	0,054	0,071	0,054	0,089	0,036	0,036	0,057
10	0,018	0,088	0,018	0,036	0,054	0,054	0,054	0,089	0,018	0,036	0,046
11	0,053	0,123	0,018	0,036	0,054	0,071	0,054	0,089	0,036	0,036	0,057
12	0,018	0,088	0,018	0,036	0,054	0,054	0,054	0,089	0,018	0,036	0,046
13	0,06	0,140	0,020	0,020	0,041	0,102	0,082	0,082	0,020	0,041	0,061
14	0,020	0,120	0,020	0,020	0,041	0,082	0,061	0,102	0,020	0,041	0,053
15	0,060	0,140	0,020	0,020	0,041	0,102	0,082	0,082	0,020	0,041	0,061
16	0,020	0,120	0,020	0,020	0,041	0,082	0,061	0,102	0,000	0,041	0,051
17	0,035	0,123	0,035	0,036	0,036	0,071	0,054	0,054	0,000	0,018	0,046
18	0,035	0,088	0,035	0,036	0,036	0,089	0,054	0,054	0,000	0,018	0,044
19	0,035	0,123	0,035	0,036	0,036	0,071	0,054	0,054	0,000	0,018	0,046
20	0,035	0,088	0,035	0,036	0,036	0,089	0,054	0,054	0,000	0,018	0,044
21	0,040	0,120	0,040	0,020	0,041	0,082	0,061	0,061	0,020	0,000	0,049
22	0,020	0,100	0,040	0,020	0,020	0,082	0,061	0,061	0,000	0,000	0,040
23	0,040	0,120	0,040	0,020	0,041	0,082	0,061	0,061	0,020	0,000	0,049
24	0,020	0,100	0,040	0,020	0,020	0,082	0,061	0,061	0,000	0,000	0,040

Como se puede apreciar en la tabla, la variante de KNN que obtuvo mayor desempeño fue la número 22, esta consiste en tomar k = 7, descarta ejemplos con atributos faltantes, usar pesos y min-max.

Hipótesis que explique la variante ganadora:

- Al igual que sucede en Naive Bayes, los atributos desconocidos introducen ruido al conjunto, es buena idea su eliminación.
- Según se observa empíricamente un modelo k = 7 modela mejor la realidad.
- Usar pesos permite dar más jerarquía a los atributos que tienen mayor importancia en la determinación de la clasificación.

ID3

Se realizó una validación cruzada sobre el mismo conjunto de entrenamiento que los algoritmos anteriores con un árbol generado por el algoritmo ID3 de la tarea anterior. Los resultados se muestran en la siguiente tabla.

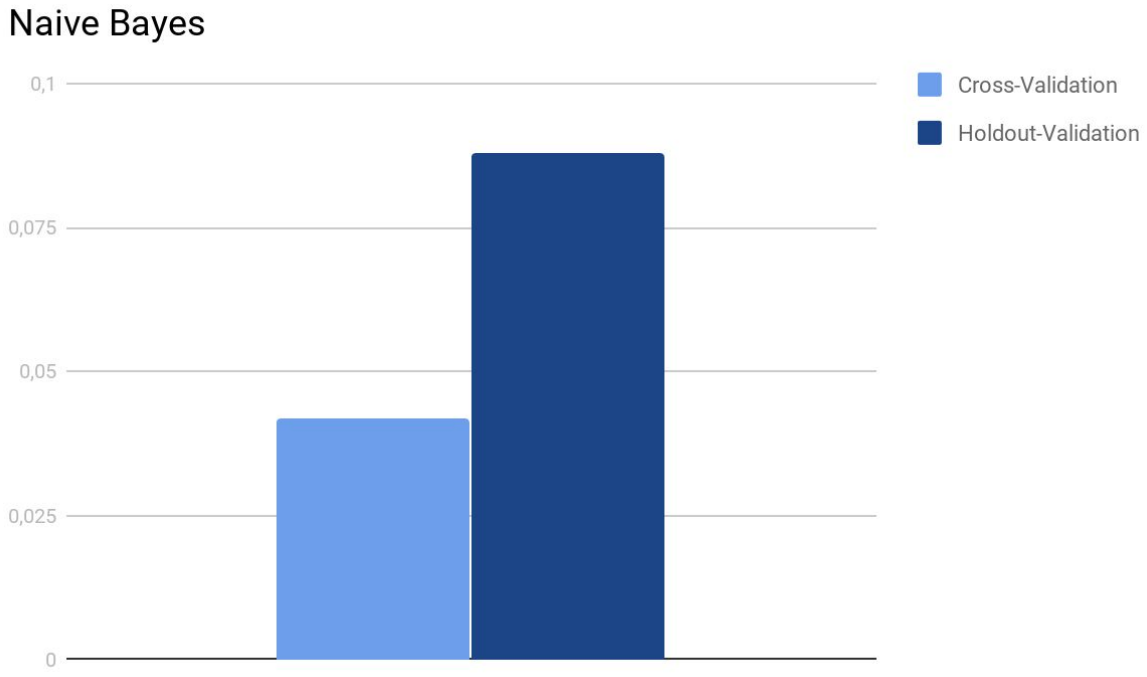
Tabla 3 - ID3

#	Iter 1	Iter 2	Iter 3	Iter 4	Iter 5	Iter 6	Iter 7	Iter 8	Iter 9	Iter10	%E
1	0,245	0,184	0,082	0,306	0,306	0,143	0,122	0,184	0,143	0,184	0,190

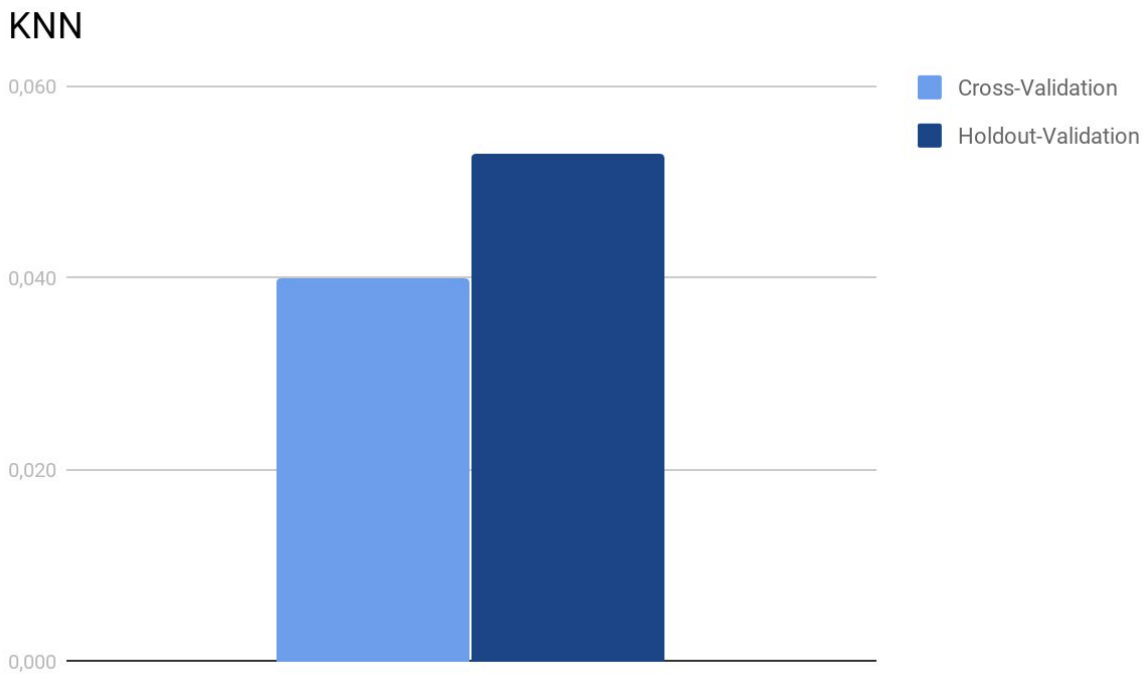
Ernesto Fernandez - Federico Fioritto - Javier Stabile
Grupo 6
MAA 2018

Validación cruzada vs validación Holdout

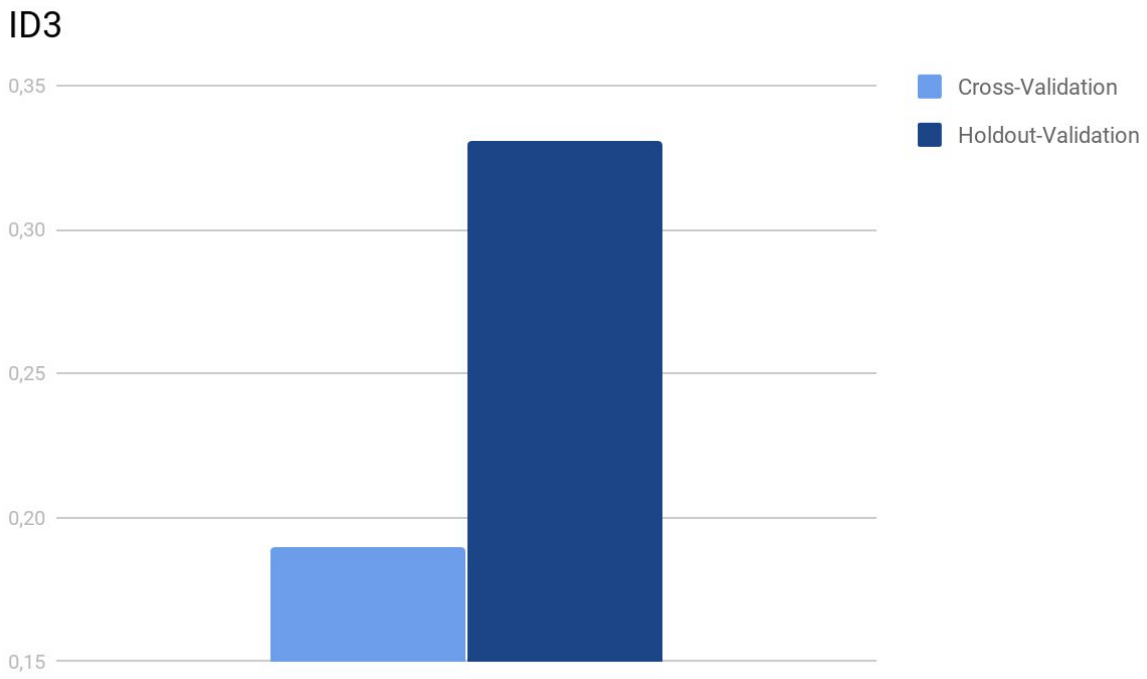
A continuación se comparan los dos métodos de validación utilizados en el laboratorio (Cross-Validation y Holdout-Validation) para una variante de cada clase de algoritmo (Árbol de decisión, Bayes sencillo y KNN). En particular, se seleccionaron las variantes que mejor resultado obtuvieron en la parte anterior en Naive Bayes y KNN.



Vemos que en Naive Bayes, la validación de tipo Holdout sobre el 20% del conjunto original de datos y entrenando con el 80% restante muestra que se cometieron errores en el 8,77% de los casos, mientras que la validación cruzada lo hizo en 4,2% de los ejemplos.



Por su parte, en el algoritmo KNN se encontró mayor similitud en la tasa de errores entre ambos métodos de validación: 5,26% en Holdout y 4,0% en validación cruzada.



Finalmente, las pruebas con el algoritmo ID3 de la tarea anterior arrojaron un error promedio con validación cruzada de 19% y con Holdout de 33,6%.

Se aprecia que para cada tipo de clasificador, el método de validación cruzada no fue una buena aproximación de la validación simple. Sólo en el clasificador KNN ambos resultados son similares.

Por último, se muestra en la siguiente tabla un resumen del desempeño de los distintos algoritmos sobre el conjunto de validación, resultando como ganador el KNN seguido por Naive Bayes. Por otro lado, el Naive Bayes es más eficiente que KNN dado que posee un menor tiempo de ejecución, esto es debido a que el KNN para validar cada instancia necesita recorrer todo el conjunto de entrenamiento, mientras que Naive Bayes lo procesa una única vez almacenando la frecuencia de ocurrencia de los atributos.
La validación solo requiere una cuenta aritmética para Naive Bayes.

Algoritmo	Cantidad elementos	Cantidad de errores	% de errores
ID3	114	38	33,6%
Naive Bayes	114	10	8.77%
KNN	114	6	5.26%