Weka en la generación de modelos de aprendizaje automático

Nicolas Jiménez, Oscar Forero
Laboratorio 2
Universidad de los Andes, Bogotá, Colombia
en.jimenez@uniandes.edu.co
of.forero41@uniandes.edu.co
Fecha de presentación: octubre 28 de 2020

1 Introducción

El segundo laboratorio tiene como objetivo aprender a usar técnicas de aprendizaje automático y usar aplicaciones para la clasificación de set de datos. Para esto, es importante el manejo de la herramienta Weka, entender sus funciones, la estructura de los datos y la visualización de resultados como insumos para el análisis de la información.

2 Entendimiento de la herramienta y de los datos

2.1 Contenido archivo ARFF

Los archivos ARFF tienen dos secciones principales: un encabezado donde se escriben los comentarios e información resaltada, el nombre de la relación y los atributos de la relación (consta de un nombre y un tipo de datos). Y los datos que son el contenido de los atributos, separados por coma.

3 Entrenamiento de un clasificador

3.1 Entrenamiento con Weka

Se utilizó el algoritmo de Naive Bayes para entrenar el clasificador dividiendo los datos entregados en 60% entrenamiento y 40% test, este ejercicio se realizó con los datasets de “subjects” y “bodies” sin lematizar.

Para el dataset de “subjects” los resultados nos muestran que, de un total de 26 instancias de test, el clasificador realizó correctamente la clasificación de 19 instancias, siendo estas el 73% del total. Además, vemos diferentes métricas como el error absoluto medio el cual da 0,2514. También, tenemos la matriz de confusión la cual nos muestra que los errores están repartidos casi equitativamente entre los falsos positivos (3 casos) y los falsos negativos (4 casos).

Para el dataset de “bodies” los resultados nos muestran que, de un total de 26 instancias de test, el clasificador realizó correctamente la clasificación de 18 instancias, siendo estas el 69,23% del total. Además, vemos diferentes métricas como el error absoluto medio el cual da 0,3077. También, tenemos la matriz de confusión la cual nos muestra que los errores son por falsos negativos (8 casos). Es decir, el clasificador falló en detectar 8 correos que anunciaban conferencias y los clasifico como “todo lo demás”.

Se reentrenó el mismo modelo y esta vez se validó utilizando el método de Cross Validation con Folds igual a 10.

Para el dataset de “subjects” los resultados nos muestran que, de un total de 64 instancias, el clasificador realizó correctamente la clasificación de 57 instancias, siendo estas el 89% del total. Además, vemos diferentes métricas como el error absoluto medio el cual da 0,1764. También, tenemos la matriz de confusión la cual nos muestra que los errores están repartidos casi equitativamente entre los falsos positivos (4 casos) y los falsos negativos (3 casos).

Para el dataset de “bodies” los resultados nos muestran que, de un total de 64 instancias, el clasificador realizó correctamente la clasificación de 48 instancias, siendo estas el 75% del total. Además, vemos diferentes métricas como el error absoluto medio el cual da 0,2498. También, tenemos la matriz de confusión la cual nos muestra que los errores son en su mayoría por falsos negativos (14 casos). Es decir, el clasificador falló en detectar 14 correos que anunciaban conferencias y los clasifico como “todo lo demás”. Y tuvo 2 errores por falsos positivos.

Para comparar la calidad del clasificador podemos utilizar diferentes medidas, en este caso vamos a ver el porcentaje de instancias clasificadas correctamente y las métricas de error absoluto medio y de error cuadrático medio. Teniendo en cuenta estos valores en los clasificadores de los datasets de “subjects” y “bodies” podemos concluir que la calidad del clasificador es mejor cuando se utiliza el método de Cross Validation. Esto es evidente ya que el porcentaje de instancias clasificadas correctamente es mayor al utilizar Cross Validation. Además, las métricas de errores, que al ser menores indican mayor calidad, son menores en los casos que se utilizó Cross Validation. (Ver tabla 1 para detalles de los valores).

Dataset	Algoritmo	Método de validación	Total instancias	Instancias clasificadas correctamente	% Instancias clasificadas correctamente	Mean absolute error	Root mean squared error
Subjects	Naive Bayes	Split 60%	26	19	73,08%	0,2514	0,3972
		Cross Validation Folds = 10	64	57	89,06%	0,1764	0,3151
Bodies	Naive Bayes	Split 60%	26	18	69,23%	0,3077	0,5547
		Cross Validation Folds = 10	64	48	75%	0,2498	0,4995

Tabla 1: Detalle del algoritmo Naive Bayes con comparación de métodos de validación en instancias clasificadas correctamente y metricas de errores.

Al entrenar el modelo con los datasets lematizados y el método de Cross Validation y compararlos con los resultados de los datasets sin lematizar encontramos que no es posible determinar cual tiene mejores resultados. En el caso del dataset “subjects” vemos que la lematización mejora la cantidad de instancias clasificadas correctamente en dos. Sin embargo, las métricas de errores no mejoran. De hecho, el error cuadrático medio es peor por poco para el dataset lematizado. En el caso del dataset “bodies” solo se mejora la cantidad de instancias clasificadas en una. Las métricas de errores si mejoran por poco en comparación a las métricas del dataset sin lematizar. Se propone realizar este ejercicio con un dataset que contenga más datos para poder identificar si mejoran considerablemente las instancias clasificadas correctamente y las métricas. (Ver tabla 2 para detalles de los valores).

Dataset	Algoritmo	Método de validación	Total instancias	Instancias clasificadas correctamente	% Instancias clasificadas correctamente	Mean absolute error	Root mean squared error
Subjects	Naive Bayes	Split 60%	26	19	73,08%	0,2514	0,3972
		Cross Validation Folds = 10	64	57	89,06%	0,1764	0,3151
Subjects Stemmed	Naive Bayes	Cross Validation Folds = 10	64	55	85,94%	0,1745	0,3167
Bodies	Naive Bayes	Split 60%	26	18	69,23%	0,3077	0,5547
		Cross Validation Folds = 10	64	48	75%	0,2498	0,4995
Bodies Stemmed	Naive Bayes	Cross Validation Folds = 10	64	49	76,56%	0,2374	0,4847

Tabla 2: Detalle del algoritmo Naive Bayes con comparación de datasets lematizados y no lematizados y con métodos de validación en instancias clasificadas correctamente y metricas de errores.

Se entrenó un modelo con el algoritmo de k – Nearest para los datasets “subjects” y “bodies” sin lematizar y lematizados. Al comparar los resultados en cuanto a calidad vemos que el algoritmo k – Nearest tiene peores resultados que el de Naive Bayes con el método de validación Cross Validation en los ambos datasets lematizados y sin lematizar. Esto se evidencia al ver el porcentaje de instancias clasificadas correctamente las cuales son menores al utilizar el algoritmo de k – Nearest. Además, el algoritmo k – Nearest presenta metricas mayores, es decir peores, frente al algoritimo de Naive Bayes. (Ver tabla 3 para detalles de los valores).

Dataset	Algoritmo	Método de validación	Total instancias	Instancias clasificadas correctamente	% Instancias clasificadas correctamente	Mean absolute error	Root mean squared error	Tiempo para construir el modelo (Segundos)
Subjects	Naive Bayes	Split 60%	26	19	73,08%	0,2514	0,3972	0,01
		Cross Validation Folds = 10	64	57	89,06%	0,1764	0,3151	0,01
	k - Nearest	Cross Validation Folds = 10	64	51	79,69%	0,254	0,4487	0
Subjects Stemmed	Naive Bayes	Cross Validation Folds = 10	64	55	85,94%	0,1745	0,3167	0
	k - Nearest	Cross Validation Folds = 10	64	53	82,81%	0,1965	0,4023	0
Bodies	Naive Bayes	Split 60%	26	18	69,23%	0,3077	0,5547	0,12
		Cross Validation Folds = 10	64	48	75%	0,2498	0,4995	0,01
	k - Nearest	Cross Validation Folds = 10	64	38	59,37%	0,4093	0,627	0
Bodies Stemmed	Naive Bayes	Cross Validation Folds = 10	64	49	76,56%	0,2374	0,4847	0,02
	k - Nearest	Cross Validation Folds = 10	64	41	64,06%	0,3642	0,59	0

Tabla 3: Detalle de algoritmo Naive Bayes y k - Nearest con comparación de datasets lematizados y no lematizados, con métodos de validación en instancias clasificadas correctamente y metricas de errores y tiempo para construir el modelo en segundos.

3.2 Resultados clasificación de un texto

Utilizando el proyecto JAVA que permite la clasificación de un texto utilizando un modelo Weka, se obtienen los siguientes resultados:

Texto original

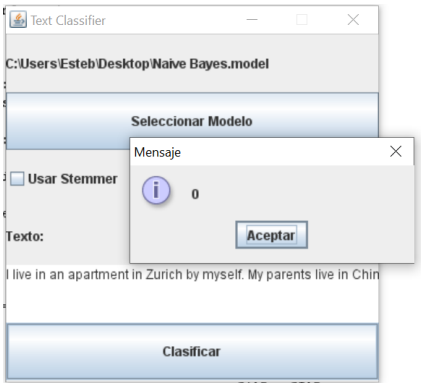
> “I live in an apartment in Zurich by myself. My parents live in China, where I was born and brought up. My sister is a high school student. And now she studies at a high school in Ohio as an exchange student. I also want to study abroad in my near future. I am busy every day as I have to study, practice tennis, and do all the housework.”

Estos son los resultados entrenando modelos mediante los siguientes archivos .arff (set de datos), modo de test y algoritmo clasificador:

Archivo ARFF: dbworld_bodies.arff

Modo de Test: cross-validation

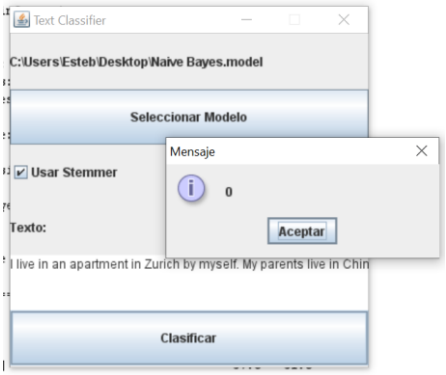
Modelo Clasificador: Naive Bayes



Archivo ARFF: dbworld_bodies_stemmed.arff

Modo de Test: cross-validation

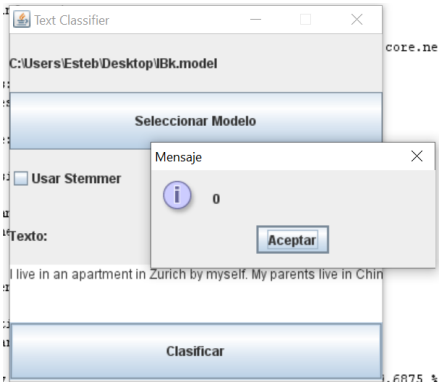
Modelo Clasificador: Naive Bayes



Archivo ARFF: dbworld_subjects.arff

Modo de Test: cross-validation

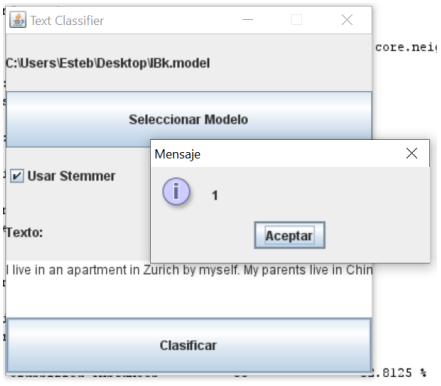
Modelo Clasificador: IBk



Archivo ARFF: dbworld_subjects_stemmed.arff

Modo de Test: cross-validation

Modelo Clasificador: IBk



3.3 Conclusiones

En conclusión, la generación de modelos de aprendizaje automático con Weka es un proceso que puede ser ejecutado mediante varias modalidades, unas más escalables que otras, y con diferentes algoritmos, que pueden ser óptimos unos más que otros. Por lo tanto, la elección de un algoritmo y modo de prueba dependen del contexto, las características del set de datos y los resultados esperados que se requieran analizar.

4 Bibliografía

1. **Michele Filannino, PhD.** DBWorld e-mails Data Set. [En línea] http://archive.ics.uci.edu/ml/datasets/DBWorld+e-mails#.

2. **Porras, Christian Fernando Ariza.** *Introducción a Weka.* s.l. : Universidad de los Andes.

3. **Morate, Diego García.** *MANUAL DE WEKA.*

4. **Aler, Ricardo.** *Tutorial Weka 3.6.0.* 2009.