El título que queráis

Jorge Durán León, Jaime Enríquez Ballesteros, Marcos de las Heras Roncero

Escuela Politécnica Superior, Fundamentos de Aprendizaje Automático

Abstract. Resumen del trabajo realizado en 100 palabras.

1 Introducción

El proyecto que se ha realizado trata sobre el análisis de un conjunto de datos recogidos por 8 sensores de gas, un sensor de temperatura y un sensor de humedad. Estos sensores fueron expuestos a estímulos por la presencia de vino y plátanos. Además, se recogen datos de la respuesta a la no presencia de ninguno de ellos. El objetivo del proyecto es la clasificación de las respuestas de los sensores a los estímulos previamente dichos. Para ello primero se analizarán los datos del dataset mediante técnicas de preprocesamiento para ver que pueden ofrecer esos datos a la hora de clasificar las respuestas. Después, se elegirán los modelos de aprendizaje automático supervisado que mejor nos convengan para dicho dataset. Por último, se compararán los resultados de los distintos modelos y razonaremos si son resultados aceptables y qué modelos han dado mejores resultados.

2 Descripción del dataset

El dataset proporcionado de los datos de los sensores está compuesto por dos archivos: el archivo HT_Sensor_dataset.dat; que contiene el identificador de la inducción, instantes de tiempo para cada inducción y los datos de los sensores para cada inducción en esos instantes de tiempo, y el archivo HT_Sensor_metadata.dat; que contiene para cada inducción su identificador, el día en que fue realizada, el estímulo usado para la inducción, y el intervalo de tiempo de la inducción, que está dividido en tiempo inicial y duración de la misma.
En este último dataset hay 3 clases distintas para clasificar, que se corresponden con los estímulos que reciben los sensores: vino (wine), plátano (banana) y ningún estímulo (background). En total se realizan 100 inducciones, donde 36 se realizan con vino, 33 con plátano y 31 son background. De esas 100 inducciones se recogen 928991 datos en distintos instantes de tiempo. Al analizar los datos se puede comprobar que no hay datos para la inducción con el identificador 95, por lo que esta instancia es descartada del análisis. Además de los datos recogidos durante la inducción, se nos ofrecen los datos de los instantes previos y posteriores de la inducción. Esta división del tiempo se puede calcular fácilmente para cada inducción con la ayuda de los datos del tiempo del archivo de los metadatos (HT_Sensor_metadata.dat), por lo que podemos distinguir para cada experimento estos tres periodos diferenciados. Comprobamos que para los experimentos con identificadores 14 y 76 no tenemos datos posteriores a la inducción, por lo que también los descartamos, quedando 97 instancias válidas.

Los atributos del dataset son numéricos y reales, por lo que se pueden utilizar las medidas características de la distribución de cada atributo para analizarlo. Estas medidas pueden ser de centralización, como la media, o de dispersión, como la varianza. Con estas medidas se puede ver como se comporta cada sensor frente a los estímulos. En el siguiente apartado se verá un pequeño resumen sobre el análisis llevado a cabo sobre los datos y el proceso a seguir para obtener los atributos que serán utilizados para entrenar a los modelos en el apartado 4.

3 Análisis de los datos y elección de atributos

La mayoría de funciones utilizadas para el análisis de los datos y el preprocesamiento previo al entrenamiento de modelos se pueden encontrar en el fichero *Preprocess.py*. La única excepción es la función utilizada para la representación de instancias a lo largo del tiempo que encontramos en el fichero *Plot_Induction_Figure.py* que se facilita en [1]. Todas las operaciones de este apartado (así como del resto) pueden encontrarse en *proyecto.ipynb*. Se utilizan los módulos de Pandas [2], NumPy [3], Sci-kit Learn [4] y Matplotlib [5].
El primer paso para el análisis será representar los experimentos para cada clase y fijarse en qué atributos pueden marcar la diferencia a la hora de clasificar. La Figura 1 muestra al progresión de los sensores para 3 experimentos (id=0, id=1, id=69), uno por clase.

Se puede observar como los atributos de temperatura y humedad obtienen valores y varianzas muy similares independientemente de la clase. Al representar esta relación, se comprueba como estos atributos no nos van a ayudar a predecir la clase de cada experimento, al no estar correlacionados. Por otro lado, se observa como cuando la humedad es más alta, los sensores obtienen unos resultados más fluctuantes que con humedad baja. Se puede ver claramente cuando se representan los experimentos con id=0 e id=10 (se muestra en el código). Se estima que los atributos de temperatura y humedad pueden ser útiles si se utilizan junto a otros atributos para la clasificación, pero no si se utilizan solos.

Por otro lado, los sensores obtienen estimulaciones muy diferentes entre ellos según la clase. La fluctuación de los sensores durante la estimulación del experimento (entre las dos líneas azules) es muy pronunciada para el vino, seguido del plátano y muy poco notable cuando no se introduce ningún elemento en el sensor. En la Figura 1 se muestra muy claramente. Por lo tanto, la varianza de cada sensor puede ser un posible atributo para nuestro modelo.

Además, como se muestra en el notebook, la media y mediana de los sensores son muy similares independientemente de la clase que utilicemos. No serán considerados como atributos útiles para el modelo final. Lo mismo sucede con los valores máximos y mínimos de cada sensor. Aunque los valores varían mucho dependiendo de cada ejemplo, no existe un patrón que se repita según la clase. En el caso de utilizar alguna de estas cuantías como atributo de un modelo, se

obtienen resultados muy malos. Modificando mínimamente el código se puede comprobar este suceso.

Volviendo a la variabilidad de los sensores, la distancia entre el máximo y mínimo valor de cada sensor (que será referido como amplitud) también podría ser útil a la hora de clasificar. Como con la varianza, se obtienen unos valores mayores para los casos en los que el objeto es vino en comparación con un plátano o cuando no se inserta nada. Se puede considerar por lo tanto, que los atributos que se tomarán para entrenar a nuestro modelo se basarán en la variabilidad de cada uno de los sensores, al ser las cuantías que más correlación observada obtienen con la correspondiente clase. Tras calcular la varianza y amplitud de cada sensor, se obtienen 16 atributos para cada ejemplo. A estos atributos se les añaden las medias de la humedad y temperatura, que como se explica anteriormente, pueden ayudar en el proceso de clasificiación, aunque por sí solos no sean útiles. En total se tendrán 18 atributos de entrada por ejemplo del dataset.

Todos los atributos serán normalizados previo al entrenamiento del modelo. Este paso mejora la predicción de los modelos al no utilizar distintas escalas para cada atributo . Para este proyecto se utiliza el método de Sci-kit Learn

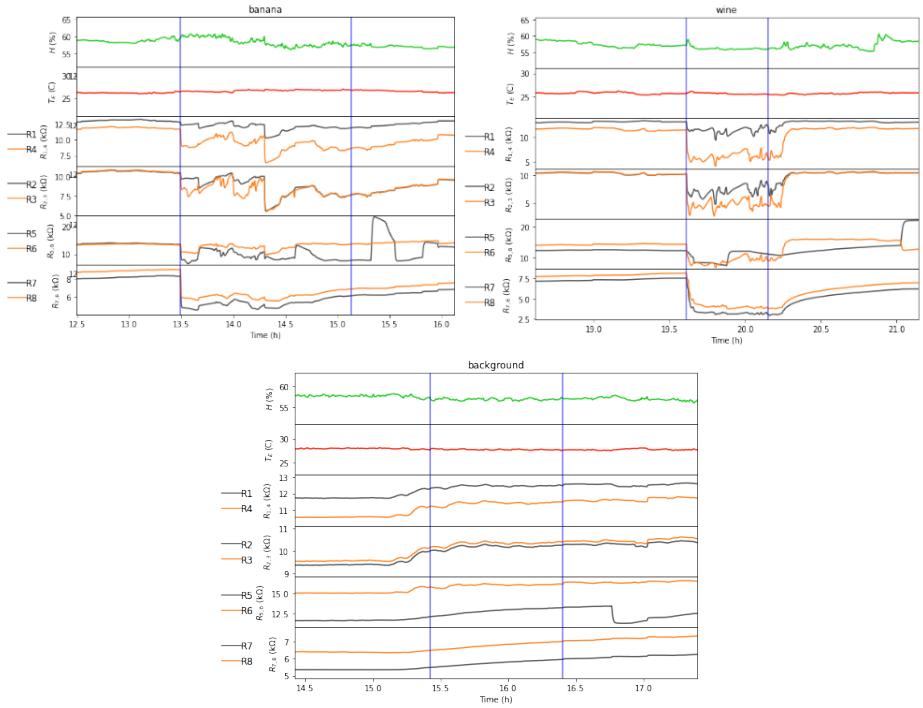

Fig. 1: Comparación entre clases

Modelo	Accuracy	F1-Score
RandomForestClassifier	0.76	0.75
ExtraTreesClassifier	0.74	0.74
XGBClassifier	0.74	0.73
LogisticRegression	0.74	0.72
LinearSVC	0.73	0.71

Table 1: Age and weight of people.

StandardScaler que normaliza los atributos restando su correspondiente media y dividiendo el resultado por la desviación estándar. Tras esta transformación se puede comenzar a entrenar a los modelos. En el siguiente apartado se describe el proceso seguido.

4 Modelos propuestos

A la hora de definir los distintos conjuntos de datos de entrenamiento y prueba, se deberá prestar atención a mantener la misma proporción de cada clase en cada conjunto. De este modo no habrá infrajuste sobre una clase que tenga poca presencia en el conjunto de entrenamiento. Para ello se hace uso del método de Sci-kit Learn *StratifiedShuffleSplit* que mezclará el conjunto de datos entero y dividirlo entre conjunto de entrenamiento y prueba. Se utilizará un conjunto de prueba del 10% de tamaño del conjunto total.

Utilizando el módulo de Python *LazyPredict* se podrá hacer una rápida evaluación sobre qué modelos obtendrán buenos resultados. Uno de los modelos del módulo puede entrenar muchos algoritmos de clasificación de Sci-kit Learn[4], xgboost [8] o lightgbm [9] a partir de unos parámetros de entrada y calcular el error de predicción sobre un conjunto de prueba calculando "F-1 Score" o "Accuracy" para cada modelo. Debido a que este conjunto de datos es tan pequeño y es probable que los modelos se sobreajusten sobre los datos de entrenamiento, causando que las predicciones sean muy erróneas, el proceso de entrenamiento y validación de *LazyPredict* se repetirá un total de 30 veces. De este modo se dividirá el conjunto de datos cada una de las veces de manera distinta, se almacenarán los resultados y se calculará la media para obtener así el mejor modelo sobre las 30 iteraciones.

Sobre los 28 modelos, los mejores cinco que se obtienen (y su correspondientes métricas) se muestran el la Tabla 1.

Los resultados podrán variar según la ejecución pero entre los cinco mejores, siempre se repiten los mostrados en la Tabla 1.

A partir de estos resultados, se deciden utilizar los cinco modelos: textit-LogisticRegression, KNeighborsClassifier, RandomForestClassifier,LinearSVC y

XGBClassifier. Utilizando la última división del conjunto de datos de las 30 iteraciones, se realiza una validación cruzada para cada modelo con nuestros datos de entrenamiento con el número de folds siendo igual a 10. De este modo, se realizarán más validaciones y los modelos podrán ser entrenados sobre la mayoría de datos del conjunto de entrenamiento. A partir de las valdaciones cruzadas podremos obtener una matriz de confusión. Esto es posible gracias al método de Sci-kit Learn textitcross_val_predict que almacena todas las predicciones sobre cada fold de la validación cruzada. Los resultados se muestran en la Figura 2.

Se decide probar a utilizar un modelo de redes neuronales y comprobar su eficiencia sobre nuestro conjunto de datos. Se implementa mediante el módulo de Python, Keras [7]. Se utilizará una red neuronal de una capa oculta con la arquitectura que muestra la Figura 3, con un input de 18 atributos, una capa oculta de 10 perceptrones y un output de 3 probabilidades. Las dos primeras capas con una función de activación ReLu y la última Softmax, para calcular las probabilidades de cada ejemplo. La Figura 3 también muestra los malos resultados de validación del modelo, con un 50% de accuracy y un error que baja muy ligeramente a medida que aumentan las épocas.

Los malos resultados de la red neuronal motivan a utilizar uno de los modelos anteriores de Sci-kit Learn. Observando las matrices de confusión, se eligen los modelos de Regresión Logística (LogisticRegression) y de "Extreme Gradient Boosting"[8] (XDBClassifier), al ser los que mejor exactitud ("accuracy") ob-

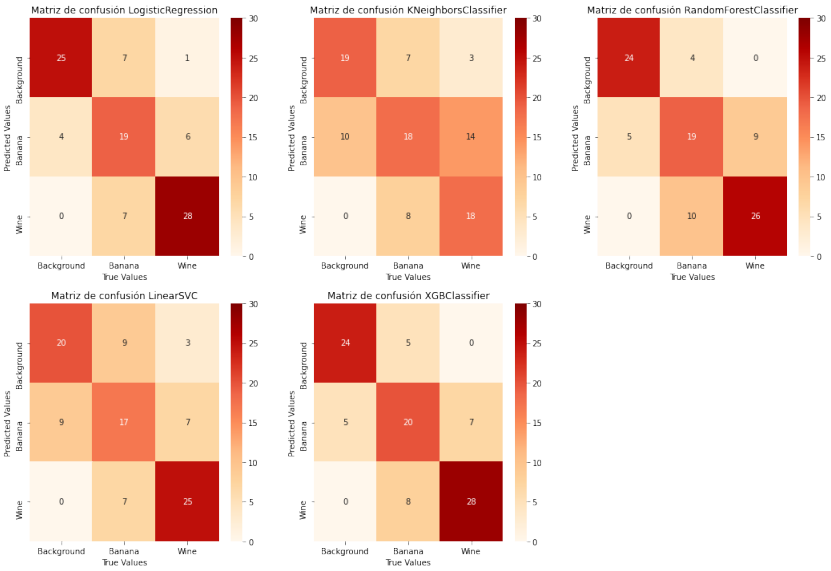

Fig. 2: Matrices de confusión sobre distintos modelos

tienen. Estos dos modelos serán refinados mediante el módulo de Sci-kit Learn, *GridSearchCV* para obtener los valores de sus parámetros que maximicen el acierto en las predicciones.
Con los modelos obtenidos tras el refinamiento de parámetros se realiza un entrenamiento y una predicción sobre los conjuntos de entrenamiento y de prueba, correspondientemente. Para el modelo de "Extreme Gradient Boosting" se obtiene un acierto del 80% mientras que para el de regresión logística un acierto del 70%. Como se menciona anteriormente, estos valores pueden variar según se ejecute otra vez, pero los resultados no deberían de cambiar más del 5%.

5 Discusión de resultados

6 Conclusión y futuro trabajo

7 Memoria

La longitud máxima serán *8 caras* sin incluir referencias. Se valorará la capacidad de síntesis por lo que superar las 8 páginas tendrá penalización. La memoria se deben tratar, de forma orientativa, los siguientes aspectos:

- Introducción [1pt]: breve introducción al problema a analizar, descripción del dataset y objetivos.
- Análisis exploratorio de los datos [1pt]: Descripción estadística de los datos: Número de clases, distribución de las clases, otras estadísticas y análisis.
- Descripción de los distintos atributos propuestos y cómo se obtienen [2pt] Modelos utilizados, descripción del protocolo experimental, estimación de

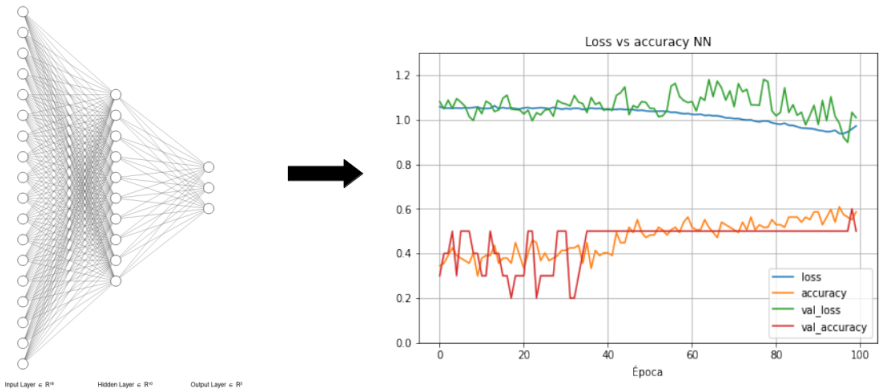

Fig. 3: Matrices de confusión sobre distintos modelos

parámetros, etc [2pt]: En esta sección se debe especificar toda la información necesaria para que otra persona, sin acceso a vuestro código, pueda reproducir los experimentos que habéis hecho. Debe quedar claro en la descripción que no se usan los datos de test para entrenar los modelos.

- Resultados obtenidos en forma tabular y/o usando gráficas [1pt]. Se debe describir que muestra cada tabla o gráfica.
- Discusión de los resultados obtenidos y conclusiones [2pt] Esta sección es la más importante del documento ya que es dónde se pone en valor el trabajo realizado. Debéis responder a preguntas tipo ¿Qué atributos y métodos han dado mejores resultados? ¿Por qué creéis que es así? ¿Son resultados aceptables? ¿Qué modelos recomendaríais bajo qué condiciones? Tal vez un modelo funcione mejor cuando se entrena con pocos datos o funcione mejor para clasificar una de las clases y peor para otras, etc.
- Además se deben utilizar al menos dos de las técnicas descritas a los largo del curso por vuestros compañeros [1pt]

Se valorará la correcta redacción del documento.

8 Presentación

Debéis entregar un ppt o pdf con el resumen del trabajo. Debe ser una presentación para presentar en 12 minutos. El tiempo de presentación será estricto y se parará a los que se pasen de tiempo.

9 Citas

Es una buena práctica referenciar los trabajos en los que se ha basado vuestro análisis. Por ejemplo, si los experimentos se han realizado utilizado scikit-learn habría que citarlo [**?**].