Menú – Una Mente Maravillosa

Tarea Individual Módulo 4 Máster Inteligencia Artificial aplicada al Deporte

José Manuel Pérez Ricote

El objetivo de esta práctica es realizar un proyecto en el que ayudemos a José (el nutricionista) a aconsejar a Andrés de la mejor forma posible sobre su alimentación utilizando Inteligencia Artificial para ello. El proceso que vamos a diseñar se centra en tres pasos de la metodología CRISP-DM: Preparación de los datos, Modelado y Evaluación de los algoritmos. Para ello, hemos escogido el menú "Una Mente Maravillosa", el cual se centra más en el paso concreto del Modelado antes que en el paso concreto de la Preparación de los Datos. El menú se compone de cinco platos, los cuales conforman la estructura de este documento.

Plato I: Elegir un conjunto de datos

Hemos decidido escoger el conjunto de datos Food101. Contiene imágenes de 101 categorías de comida. Concretamente, 1.000 imágenes por categoría, de las cuales 750 son para entrenamiento y otras 250 para test. Por lo tanto, el conjunto de datos está formado por 101.000 imágenes. Las etiquetas de los datos de entrenamiento contienen algo de ruido. Sin embargo, las etiquetas de los datos de test están perfectamente asignadas.

Plato II: Elegir una variable a modelar

En este caso, la discusión se remonta a lo realizado en la actividad colaborativa. En principio, se quiso escoger la variable saber la cantidad que Andrés come en cada ingesta. Sin embargo, después de revisar la literatura existente, se llega a la conclusión de que realizar una estimación de calorías o de volumen a partir de imágenes de comida se aleja de utilizar únicamente inteligencia artificial y requiere de aplicación de algoritmos no englobados en este campo. Para ello, se había escogido también el conjunto de datos Yummly-28K, pues incluye información nutricional de calorías para hacer mejor el análisis.

Después de esta revisión narrativa, decidimos cambiar de objetivo que se va a plantear. Si hacemos uso del conjunto de datos Food101, tiene lógica alinearlo con el objetivo de saber las categorías de comida que ha ingerido Andrés. Como el conjunto de datos viene ya preparado y con las etiquetas ya organizadas en carpetas, nos ayuda a reducir la complejidad del análisis, a la vez de aportarnos información interesante con la que aconsejar a José y Andrés.

Plato III: Preparación de los datos

Para el paso de preparación de los datos, nos tocaría elegir dos tareas. Las tareas que hemos decidido elegir son ingeniería de variables y estandarización de variables. Además, se deja comentado en el código una librería muy interesante para realizar análisis exploratorio de datos (univariante, bivariante…), cuyo nombre es pandas-profiling. Ofrece la posibilidad de generar un informe en el que veamos un análisis de cada variable, las correlaciones entre distintas variables y mucha más información que se calcula automáticamente con una función (ProfileReport).

Se encuentra comentada por la imposibilidad de probarla con pycharm y utilizando un entorno virtual. Esta herramienta la he probado en diversas ocasiones (incluso a nivel productivo) para realizar un análisis exploratorio de datos sin necesidad del arduo trabajo que suele conllevar en notebooks de Jupyter. Sin embargo, para esta ocasión quería realizar un código dividido en clases que escalase mucho mejor en caso de querer mejorarlo, ampliarlo o incluso presentarlo en un repositorio.

Con respecto a la tarea de ingeniería de variables, presentamos una forma de seleccionar variables a partir del concepto de información mutua. La información mutua entre dos variables aleatorias es un valor no negativo que mide la dependencia entre variables. Si es igual a cero, ambas variables son independientes, y cuanto más alto sea el valor significa mayor dependencia. Esto se realiza con una función de la librería sklearn llamada mutual_info_classif, la cual estima la información mutua con respecto a una variable objetivo discreta.

La segunda tarea que se presenta realiza una estandarización de las variables, dividida en tres tipos de transformación de los valores de las variables: normalización, estandarización y escalado. Todo ello se vuelve a realizar con funciones de la librería sklearn: MinMaxScaler, StandardScaler y RobustScaler. La primera realiza una normalización de los datos en un rango que se puede preestablecer y que por defecto es entre 0 y 1. La segunda realiza una sustracción de la media y un escalado con respecto a la varianza unitaria. La tercera utiliza estadísticos que son robustos a valores atípicos. El paso de preparación contiene un parámetro en su función para indicar cual de los tres escaladores se quieren utilizar para realizar la prueba que se vaya a diseñar.

Más allá de las tareas en sí, definimos otra función para codificar las clases de salida con valores numéricos discretos para ambos conjuntos de entrenamiento y test.

Plato IV: Modelado

En el paso del modelado de los datos, tenemos que escoger tres algoritmos diferentes. Hemos decidido escoger tres algoritmos con los que queríamos profundizar su conocimiento para tener una perspectiva mejor del ecosistema de algoritmos regresores y clasificadores. Son los siguientes algoritmos: clasificador Random Forest, Support Vector Machine y Naive Bayes.

El algoritmo clasificador de Random Forest utiliza la función RandomForestClassifier de la librería sklearn para modelar nuestros datos. En primer lugar, ejecutamos dicha librería con un clasificador por defecto para que nos sirva como entrada a la función GridSearchCV de sklearn. Ésta realiza un análisis de sensibilidad de los hiperparámetros de un Random Forest para obtener el modelo óptimo en rendimiento para nuestro propósito. Una vez obtenido el modelo con los mejores resultados, lo guardamos para la evaluación del mismo.

El algoritmo de clasificación de Support Vector Machine que hemos escogido está en la función SVC de sklearn. Realizamos el mismo proceso que para el modelo de Random Forest: ejecutar un modelo por defecto que nos sirva como entrada para la función de análisis de sensibilidad de los hiperparámetros del algoritmo. Posteriormente, ejecutar dicha función para obtener el modelo más óptimo y, finalmente, devolver el mejor modelo para realizar la evaluación de datos.

Por último, el modelo Naive Bayes funciona de una forma mucho más sencilla, al no contar con hiperparámetros que optimizar. En este caso, ejecutamos un modelo por defecto de su variable gaussiana, aplicando la función GaussianNB de la librería sklearn.

Plato V: Evaluación del modelado

El último paso de nuestro proceso es la evaluación de los algoritmos previamente configurados. En él, calculamos tres métricas y los mostramos por pantalla a través de los logs. Estas métricas son: valor f1, matriz de confusión y área por debajo de la curva ROC.

En primer lugar, la métrica F1 es una métrica que balancea los resultados de la matriz de confusión, calculando la media harmónica de la precisión y la exhaustividad. Su fórmula matemática es la siguiente:

$$F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{\mathrm{tp}}{\mathrm{tp} + \frac{1}{2}(\mathrm{fp} + \mathrm{fn})}.$$

Añadimos un parámetro para elegir entre tres tipos de opciones del parámetro average de la función f1_score: micro, macro y weighted. Micro calcula la métrica f1 globales, macro calcula la métrica F1 para cada etiqueta y a partir de ahí calcula la media, y weighted calcula la métrica F1 para cada etiqueta y a partir de ahí calcula la media ponderada.

En segundo lugar, la matriz de confusión es una herramienta que permite visualizar el rendimiento de un algoritmo de clasificación (englobado en aprendizaje supervisado). En su definición más sencilla, en la cual sólo hay dos clases y se denomina matriz de confusión binaria, definimos los siguientes estados:

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Por último, el área por debajo de la curva ROC (AUC-ROC) es una medida del rendimiento para problemas de clasificación que nos indica la calidad del modelo con valores entre 0 y 1 (siendo 1 un modelo con una exactitud del 100% en sus predicciones. Compara la ratio de falsos positivos (eje X) y la ratio de verdaderos positivos (eje Y).

De la misma forma que para la métrica F1, también añadimos un parámetro para decidir entre dos de las opciones del parámetro average de la función roc_auc_score (macro y weighted). También, para la clasificación multi clase, la función de roc_auc_score necesita especificar el parámetro multi_class a one over the rest (OVR) o one-vs-one (OVO). Para la opción macro, utilizamos la estrategia OVO y para las otras opciones utilizamos la estrategia OVR. OVR trata a cada clase contra todas las demás, mientras que OVO trata a cada clase contra las demás una a una.

Resultados

Por último, añadimos una serie de conclusiones obtenidas durante todo el proceso. Para hacer eso, antes especificamos los resultados obtenidos en la evaluación. Los mejores resultados con las clases ['apple_pie', 'pizza', 'omelette'] son:

1. Para el algoritmo Random Forest, los mejores resultados son los siguientes:
   - Hiperparámetros óptimos
     - Criterio: entropy
     - Máxima profundidad = 12
     - Número de estimadores = 500
     - Número máximo de características: auto
   - Métrica F1: f1 = 0.656
   - Matriz de confusión: conf_matrix = [[141, 41, 68], [41, 163, 46], [27, 35, 188]]
   - Área bajo la curva ROC: auc_roc = 0.742
2. Para el algoritmo SVM, los mejores resultados son los siguientes:
   - Hiperparámetros óptimos
     - C = 10
     - gamma = 0.001
   - Métrica F1: f1 = 0.606
   - Matriz de confusión: conf_matrix = [[162, 44, 44], [81, 127, 42], [48, 36, 166]]
   - Área bajo la curva ROC: auc_roc = 0.705
3. Para el algoritmo Naive Bayes, los resultados son los siguientes:
   - Métrica F1: f1 = 0.332
   - Matriz de confusión: conf_matrix = [[0, 0, 250], [0, 0, 250], [1, 0, 249]]
   - Área bajo la curva ROC: auc_roc = 0.499

Los mejores resultados con las clases ['baby_back_ribs', 'caprese_salad', 'carrot_cake'] son:

1. Para el algoritmo Random Forest, los mejores resultados son los siguientes:
   - Hiperparámetros óptimos
     - Criterio: entropy
     - Máxima profundidad = 14
     - Número de estimadores = 1000
     - Número máximo de características: auto
   - Métrica F1: f1 = 0.751
   - Matriz de confusión: conf_matrix = [[180, 44, 26], [22, 211, 17], [57, 21, 172]]
   - Área bajo la curva ROC: auc_roc = 0.813
2. Para el algoritmo SVM, los mejores resultados son los siguientes:
   - Hiperparámetros óptimos
     - C = 5
     - gamma = 0.001
   - Métrica F1: f1 = 0.75

- Matriz de confusión: conf_matrix = [[189, 17, 44], [33, 185, 32], [49, 12, 189]]
- Área bajo la curva ROC: auc_roc = 0.813

3. Para el algoritmo Naive Bayes, los resultados son los siguientes:
   - Métrica F1: f1 = 0.337
   - Matriz de confusión: conf_matrix = [[3, 0, 247], [0, 0, 250], [0, 0, 250]]
   - Área bajo la curva ROC: auc_roc = 0.503

Los mejores resultados con las clases [‘donuts’, ‘fried_rice’, 'baby_back_ribs', 'caprese_salad', 'carrot_cake', 'apple_pie', 'pizza', 'omelette', ‘hamburger’, ‘lasagna’] son los siguientes:

1. Para el algoritmo Random Forest, los mejores resultados son los siguientes:
   - Hiperparámetros óptimos
     - Criterio: entropy
     - Máxima profundidad = 14
     - Número de estimadores = 1000
     - Número máximo de características: auto
   - Métrica F1: f1 = 0.3724
   - Matriz de confusión: conf_matrix = [[30, 22, 19, 53, 24, 14, 23, 15, 18, 32], … [5, 10, 51, 11, 12, 40, 57, 7, 44, 13], [3, 20, 22, 5, 12, 21, 18, 20, 14, 115]]
   - Área bajo la curva ROC: auc_roc = 0.651
2. Para el algoritmo SVM, los mejores resultados son los siguientes:
   - Hiperparámetros óptimos
     - C = 10
     - gamma = 0.001
   - Métrica F1: f1 = 0.3736
   - Matriz de confusión: conf_matrix = [[74, 15, 7, 34, 21, 23, 11, 19, 20, 26], … [33, 10, 24, 18, 13, 23, 32, 16, 65, 16], [17, 9, 13, 14, 10, 25, 11, 20, 13, 118]]
   - Área bajo la curva ROC: auc_roc = 0.652
3. Para el algoritmo Naive Bayes, los resultados son los siguientes:
   - Métrica F1: f1 = 0.102
   - Matriz de confusión: conf_matrix = [[0, 0, 0, 0, 250, 0, 0, 0, 0, 0], … [0, 0, 0, 0, 250, 0, 0, 0, 0, 0], [0, 3, 0, 0, 247, 0, 0, 0, 0, 0]]
   - Área bajo la curva ROC: auc_roc = 0.501

Como se puede observar, hemos probado diferentes configuraciones de clases para ver qué resultados se obtienen. Sin embargo, no todas ellas aparecen en nuestro desglose de resultados. La lista completa de combinaciones probadas es la siguiente:

- ['baby_back_ribs', 'caprese_salad', 'carrot_cake'] -> Mejores resultados
- ['apple_pie', 'pizza', 'omelette']
- [‘donuts’, ‘fried_rice’, 'baby_back_ribs', 'caprese_salad', 'carrot_cake']
- [‘donuts’, ‘fried_rice’, 'baby_back_ribs', 'caprese_salad', 'carrot_cake', 'apple_pie', 'pizza', 'omelette', ‘hamburger’, ‘lasagna’]

Análisis del Modelado: interpretación de los resultados

Una vez expuestos los resultados para los distintos algoritmos, pasamos al resumen de las interpretaciones más importantes que se extraen del análisis realizado.

En cuanto a algoritmos, queda demostrado que los algoritmos de SVM y Random Forest consiguen aprender al menos en parte cierta información extraída a partir de las imágenes. Sin embargo, el modelo de Naive Bayes no resulta eficaz, pues devuelve unos resultados en los cuales directamente se clasifican todas las muestras como si fueran de una misma clase. Por lo tanto, se puede concluir que el modelo de Naive Bayes no es adecuado para la tarea que realizamos, o al menos, en su variante gaussiana.

El mejor de los rendimientos de ambos modelos se sitúa en torno a un 75% de exactitud en la clasificación de las clases y un valor de AUC-ROC un poco superior al 80%. En cuanto a la exactitud, esto significa que el 75% de las muestras se clasifican en la clase que les corresponde. Como bien hemos explicado previamente, superar ligeramente el 80% en la curva AUC-ROC significa que tenemos 4 veces más verdaderos positivos que falsos positivos a lo largo de todos los umbrales de decisión posibles. A la hora de comparar entre ambos modelos, los resultados del clasificador Random Forest resultan ser un poco mejores en área bajo la curva ROC y en métrica F1. También queremos destacar que dependiendo de la similitud entre las clases que se escojan para evaluar nuestro modelo, los resultados son mejores o peores. Una tarta de manzana, una pizza y una tortilla se parecen mucho más entre sí que unas costillas, una ensalada y una tarta. Por lo tanto, siempre obtendremos mejores resultados para las segundas tres clases que para las tres primeras.

Además, podemos concluir que el número de clases es inversamente proporcional al resultado de nuestro modelo, pues al tener que decidir entre más clases, la probabilidad de acertar la correcta disminuye considerablemente. Para las pruebas en las que sólo decidimos entre tres clases los resultados siempre serán notablemente mejores que si decidimos entre 20 clases.

Quedan abiertas muchas posibilidades que se podrían implementar para mejorar los resultados, como la segmentación de las imágenes, la utilización de redes neuronales más eficaces para este tipo de problemas de clasificación de imágenes (redes convolucionales, redes más complejas como la Inception V3…), aplicación de técnicas que ayuden a reconocer los ingredientes, los alimentos o las cantidades de las porciones de comida.