Maestría en Explotación de Datos
y Descubrimiento del Conocimiento
Universidad de Buenos Aires

Facultad de Ciencias Exactas y Naturales
Facultad de Ingeniería

Junio 2022

Aprendizaje Automático

Trabajo Práctico N° 2

Grupo N°3

Altschuler, Florencia ; Krell, Federico ; Picasso, Juan Pablo

Resumen

El objetivo del presente trabajo práctico es lograr clasificar tipos de indumentaria a partir de imágenes y mediante diversos modelos de aprendizaje automático. A su vez, el objetivo didáctico de este trabajo es estudiar los modelos de *Boosting, Bagging, Naive Bayes*. Para lograrlo se buscó clasificar imágenes de indumentaria (ej. pantalón, remera, etc) en 10 clases, usando ensambles de modelos y comparando con otros modelos más sencillos (árboles de desición y *Naive Bayes*). Para esto, se utilizó el dataset de "*Fashion-MNIST*". Este contiene imágenes (28 x 28 píxeles) de indumentaria y el tipo de indumentaria de la que se trata (10 clases), consistente de un conjunto de entrenamiento de 60,000 datos y un conjunto de test de 10,000 datos balanceados. Se entrenaron diversos modelos como árbol de clasificación, bosques aleatorios (*Random Forest Classifier*, ensamble de tipo bagging), Gradient Boosting (ensamble de tipo boosting) y Bayes ingenuo (*Multinomial Naive Bayes*) tomando con métrica de interés la exactitud (*accuracy*). Se optimizaron los hiperparámetros `n estimators` y `max depth` de los modelos de bagging y boosting. El modelo de Random Forest mostró el mejor rendimiento (exactitud de 83.6%) al predecir con el conjunto de testeo. A fin de analizar la robustez de este modelo, se aplicó una rotación al set de testeo resultando en un rendimiento muy bajo (<6%).

Introducción

En el presente trabajo se analizaron imágenes de indumentaria con el objetivo de generar un algoritmo capaz de categorizar en tipo de indumentaria. Para esto, se utilizaron cuatro modelos de la librería de Sci-Kit Learn, dos de estos modelos se basan en métodos de ensambles como Boosting y Bootstrap aggregating (*Bagging*). Inicialmente, se presentan los datos utilizados y se describen las variables seleccionadas para las etapas posteriores. Luego, se desarrollan las técnicas y los criterios utilizados para la limpieza y análisis de los atributos de la base de datos. En esta sección también se describen los modelos de aprendizaje automático supervisado junto con los atributos y parámetros seleccionados. Por último, se desarrolla el refinamiento de hiper parámetros y selección del mejor algoritmo de clasificación para el problema abordado.

Datos

La base de datos utilizada fue Fashion-MNIST que contiene fotos de artículos de moda de la tienda de Zelando y se encuentra disponible como módulo de la API Keras de TensorFlow. Este conjunto de datos contiene 70000 imágenes cada una con su etiqueta correspondiente (la clase de indumentaria a la que pertenece[1]). De esta base, 60000 se emplearon como conjunto de desarrollo y 10000 como conjunto de testeo. Cada imagen está formada por 784 píxeles (un cuadrado de 28 píxeles de lado) y cada píxel contiene un valor entero de 0 a 255 correspondiente a una escala de grises. Tanto el dataset de entrenamiento como el de prueba están balanceados.

El conjunto de desarrollo se dividió, a su vez, en conjunto de entrenamiento y de validación, en una proporción 80% y 20%, respectivamente. Se transformó la configuración de matriz de cada imagen a una fila con 784 atributos correspondiendo cada atributo con un píxel de la imagen. Esta transformación fue necesaria debido a los tipos de modelos utilizados. Se evita utilizar un encoding ya que generaría 10 modelos distintos para cada prueba. El costo computacional es muy grande y además se perdería información final sobre cuáles son las categorías que más se confunden.

[1] En el "Anexo: Imágenes y mapeo" se detallan cada una de las etiquetas y, a modo de ejemplo, una respectiva imagen.

Por otro lado, para evitar que los modelos ordenen las categorías, se utilizó un mapeo aleatorio de letras para cada categoría. De esta forma se evitó que el modelo use los números de forma ordenada como información relevante.

Metodología

Todo el análisis se realizó en *Python* 3.7 y se utilizaron las librerías *Numpy, Pandas* y *Sklearn.* Algunos de los módulos de la librería Scikit-Learn que se emplearon fueron *model_selection, ensemble* y *metrics*.

Se utilizaron las siguientes técnicas de aprendizaje automático: árbol de clasificación, bosques aleatorios (*Random Forest Classifier*, ensamble de tipo bagging), Gradient Boosting (ensamble de tipo boosting) y Bayes ingenuo (*Multinomial Naive Bayes*). Inicialmente, se evaluó la performance de cada modelo con el conjunto de validación y entrenandolo con sus correspondientes parámetros por default. La métrica seleccionada para medir la performance fue "accuracy" (exactitud). Dicha métrica nos permite saber si el modelo pudo efectivamente clasificar la imágenes de forma adecuada, no dándole peso a los falsos positivos o falsos negativos.

A fin de optimizar los modelos de Random Forest y de Gradient Boosting, se realizó una búsqueda de hiper-parámetros variando el número de estimadores (n_estimators) y la profundidad máxima (max_depth). Para la selección de la mejor combinación de hiper-parámetros, además de la performance, se tuvo en cuenta el tiempo de ajuste. Para el mejor modelo de cada uno de los cuatro modelos considerados se procedió a evaluar su performance respecto al conjunto de datos de testeo.

Para el modelo con mejor performance en la predicción de los datos de testeo, se analizó la matriz de confusión a fin de entender cuáles fueron las categorías que más confundió el modelo pero el mayor peso para evaluar la efectividad en este caso se le dio a la correcta clasificación de cada imagen en sí misma.

Finalmente, con la intención de analizar la estabilidad del mejor modelo seleccionado previamente, se evaluó su desempeño con imágenes alteradas en su orientación.

Resultados y Conclusiones.

Los resultados pueden dividirse en cuatro partes. Prueba inicial de modelos, optimización, prueba final y robustez.

La prueba inicial puede verse en la tabla 1 del anexo. En dicha prueba podemos ver una leve mejora, probablemente circunstancial, del modelo de random forest luego del cambio por letras aleatorias a los números de las categorías. Además se obtiene una primera dimensión de las capacidades de los nuevos modelos utilizados siendo inicialmente random forest el más adecuado (accuracy de 88.1%). Cabe aclarar que se incluyó en esta prueba inicial un modelo con el algoritmo XGBoost Classifier, pero como no mostró una performance (accuracy de 86.2%) superior al modelo de Random Forest se decidió descartarlo de los análisis posteriores.

La segunda prueba fue de optimización para bagging y boosting. En las figuras 1,2,3 y 4 del anexo puede verse que en todos los casos hay puntos de inflexión a medida que se sube el parámetro. En el caso de boosting puede verse que los tiempos son un orden de magnitud mayores teniendo tiempos de un minuto para random forest y tiempos de casi 10 para los parámetros más grandes explorados en boosting.

Se observa una diferencia más pronunciada entre entrenamiento y validación para el random Forest. Específicamente con la variación de N Estimators Por otro lado por una cuestión de tiempo computacional no se llega a ver la diferencia en gradient Boosting pero se observa que de seguir la tendencia sería a separarse.

Se selecciona para random forest a 50 estimadores y una profundidad de 9. Para boosting 5 estimadores y profundidad de 5 y los valores por defecto de Naive Bayes y árbol de decisión.

Por último se explora el conjunto de testing con todos los modelos (Tabla 2 Anexo Resultados). Resultando el modelo de random forest, el cual puede verse en la figura 5 resulta el de mejor accuracy.

En este modelo se evalúa la matriz de confusión, en la cual se observa que las variedades 6 (camisa) y 4 (Campera) son las que más problemas tienen. Siendo la 6 la que menos logra diferenciarse particularmente de las 0 (Remera), 2 (Buzo) y 4. La indumentaria que mejor clasifica es el 8 (Cartera).

Luego de terminar el análisis se realiza un análisis de robustez. El cual implicó la rotación del set de prueba y predecir el mismo con el modelo optimizado para el test de entrenamiento y validación.

En dicha prueba se encontró que la accuracy fue 0.054. A las imágenes de entrenamiento se les agregaron las mismas imágenes rotadas en 90° y se volvió a entrenar y predecir con el conjunto de test original sumado al rotado y se obtuvo un accuracy de 0.798. Queda así construido un modelo más robusto, para futuras pruebas se podría agregar otras rotaciones para mejorar el modelo.

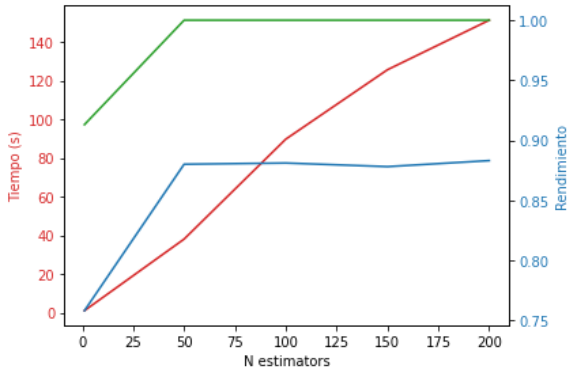
Bibliografía

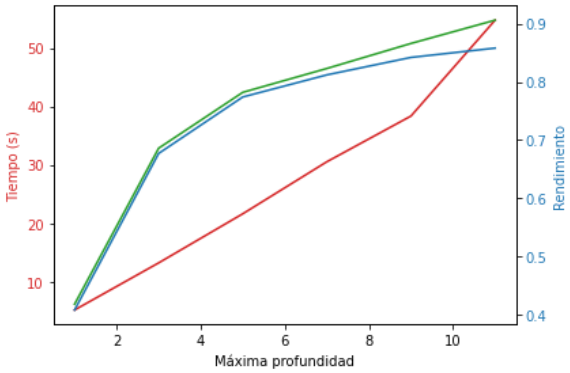
- Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems. Aurélien Géron, O'Reilly, 2da edición, 2019.
- Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. Link: https://scikit-learn.org/
- Material de clase.
- GitHub - zalandoresearch/fashion-mnist: A MNIST-like fashion product database. Benchmark

ANEXO:Resultados

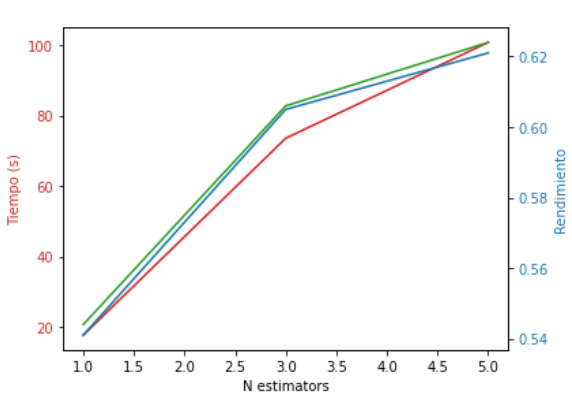
Modelo	Exactitud promedio
Gradient Boosting	0.621
Random forest	0.881 (después de mapear) 0.882 (antes de mapear)
Naive Bayes	0.664
Decision tree	0.795

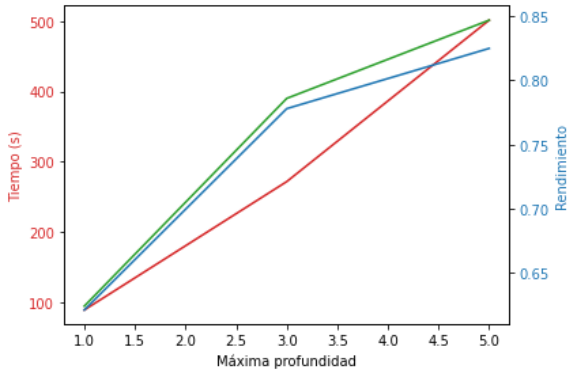
Tabla 1. Modelos utilizados con la respectiva exactitud calculada con el conjunto de validación.





Figuras 1 y 2. Optimización de hiper-parámetros: número de estimadores (izquierda) y máxima profundidad (derecha) para el modelo Random Forest. La línea roja representa el tiempo que tardan en correr los distintos modelos y en línea azul se muestra el rendimiento para el conjunto de validación y en verde para el conjunto de entrenamiento.





Figuras 3 y 4. Optimización de hiper-parámetros: número de estimadores (izquierda) y máxima profundidad (derecha) para el modelo Gradient Boosting Machine. La línea roja representa el tiempo que tardan en correr los distintos modelos y en línea azul se muestra el rendimiento para el conjunto de validación y en verde para el conjunto de entrenamiento.

Modelo	Exactitud promedio
Gradient Boosting	0.818
Random forest	0.835
Naive Bayes	0.655
Decision tree	0.789

Tabla 2. Modelos utilizados con la respectiva exactitud calculada con el conjunto de testeo.

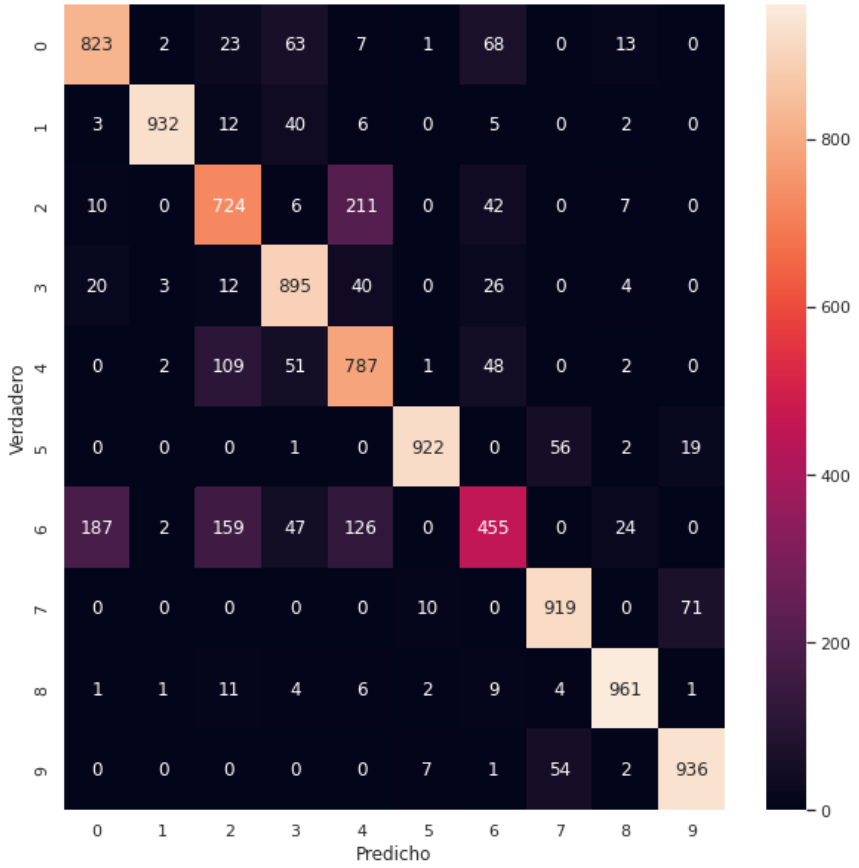


Figura 5. Matriz de confusión del modelo Random Forest para la predicción del conjunto de testeo.

Anexo: Imágenes y mapeo utilizado.

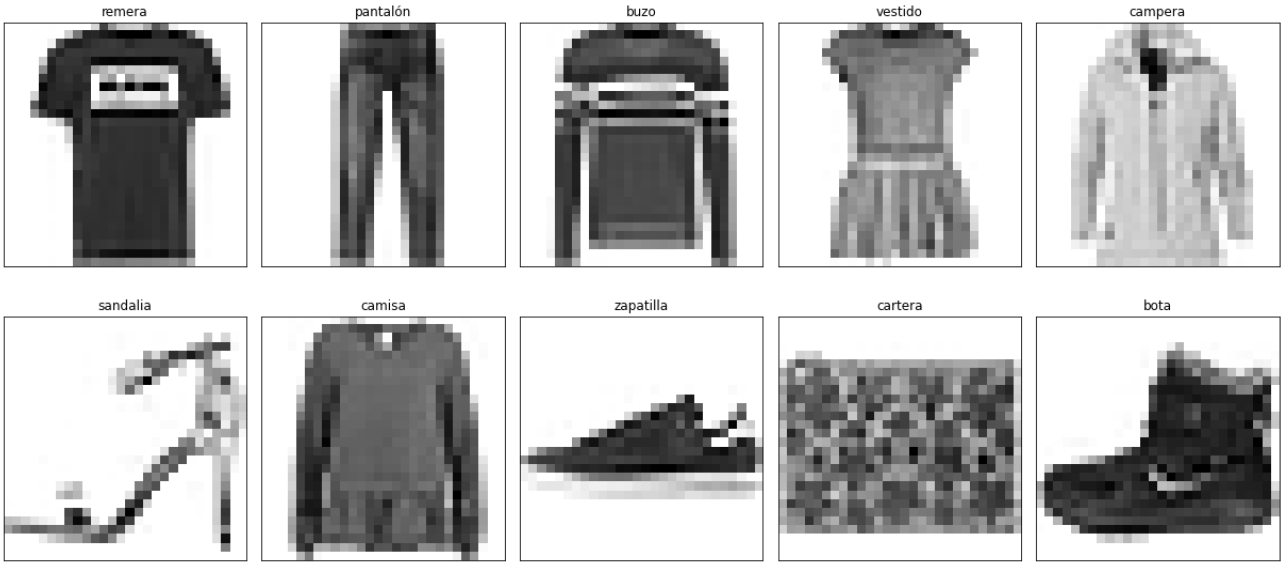


Tabla 1. Ejemplos de las clases con sus etiquetas. El mapeo utilizado fue: map={ 0:'t', 1:'e', 2:'v', 3:'j', 4:'p', 5:'d', 6:'w', 7:'h', 8:'i', 9:'q'}