Práctica 1: MLlib
Aprendizaje Automático a Gran Escala

Alejandro Rodríguez (alejandro.rodrigueze@udc.es), Pedro de la Fuente (p.de.la.fuente@udc.es) y Jorge Menéndez (j.menendezf@udc.es)

Universidade da Coruña

1 Introducción

Para este trabajo vamos a usar el dataset "Long-distance running dataset" [1], que contiene datos de distintas carreras de más de 30.000 atletas en los que se tratan la edad, sexo, distancias, etc.

Nuestro objetivo es que el modelo que creemos generalice los resultados de unos atletas y, dependiendo de ciertas características, prediga bien los resultados de atletas de similares características en nuevas actividades físicas.

2 Presentación del Dataset

Nuestro dataset contiene a 36.412 atletas y 13.326.793 filas de datos. Las variables que recoge son:

- `_c0`: Identificador de cada línea del dataset.
- `Datetime`: Fecha del evento.
- `Athlete`: Identificador del atleta.
- `Distance`: Distancia de la actividad en kilómetros.
- `Duration`: Duración de la actividad en minutos.
- `Gender`: Género del atleta.
- `Age_group`: Grupo de edad al que pertence el atleta.
- `Country`: País de origen del atleta.
- `Major`: Maratón(es) que corrió el atleta (lista).

Table 1. Cinco primeras filas del dataset original.

_c0	datetime	athlete	distance	duration	gender	age_group	country	major
0	2020-01-01	0	0.0	0.0	F	18 - 34	USA	CHI 2019
1	2020-01-01	1	5.72	31.63	M	35 - 54	GER	BER 2016
2	2020-01-01	2	0.0	0.0	M	35 - 54	UK	LON 2018, LON 2019
3	2020-01-01	3	0.0	0.0	M	18 - 34	UK	LON 2017
4	2020-01-01	4	8.07	38.61	M	35 - 54	USA	BOST 2017

3 Preprocesamiento de datos

3.1 Limpieza y filtrado

Empezamos haciendo un filtrado de filas. En él, eliminamos las filas en las que la duración o la distancia es 0, es decir, aquellas cuyo entrenamiento básicamente no se produjo. A posteriori, también eliminamos aquellas filas en las que hay algún valor nulo.

Como seguimos teniendo un número demasiado elevado de filas, decidimos hacer un filtro por el número de atletas. Tras probar con distintas cantidades, nos quedamos con 15.000 atletas, un número que garantiza suficientes datos.

3.2 Creación de una nueva variable: `season`

Como se trata de la realización de actividad física, hemos decidido contemplar también la estación en la que se produce cada evento. Esto se debe a que creemos que puede ser importante en nuestro análisis, ya que las condiciones de entrenamiento no son iguales en verano que en invierno. Dicha variable la creamos a partir del mes en el que se realiza el entrenamiento.

3.3 Selección de variables relevantes

Seleccionamos aquellas variables que nos resultan útiles para el análisis. Nos deshacemos de las que no aportan ninguna infromación como `major` o `datetime` (ya que ahora tenemos `season`).

3.4 Eliminación de valores atípicos

Eliminamos los valores atípicos del dataset basándonos en distancia y en duración, las dos variables claves. En el caso de distancia, suprimimos las filas con valores menores que 0.5 o mayores que 100. En el de duración, aquellas con menos de 5 minutos o más de 300.

De forma adicional, creamos una nueva variable (`pace_min_km`) que sería el ritmo medio (min/km). También nos fijaríamos en sus datos atípicos, desechando aquellos ritmos menores que 1 min/km o mayores que 9 min/km.

3.5 Tamaño final del dataset

Trabajaremos con un dataset de 1.806.322 filas, y 8 variables (`athlete`, `distance`, `duration`, `gender`, `age_group`, `country`, `season` y `pace_min_km`).

4 Preparación de datos para el entrenamiento

4.1 Codificación de variables categóricas

Dado que los modelos de `pyspark.ml` solo admiten variables numéricas, las columnas categóricas (`gender`, `age_group`, `season` y `country`) se transformaron mediante un proceso de codificación. Primero, se aplicó `StringIndexer` para convertir las categorías en índices numéricos, y posteriormente, `OneHotEncoder` para generar un vector binario por variable. Las columnas resultantes se integraron en una única variable de entrada llamada `features` utilizando `VectorAssembler`, pues todos los modelos de Spark ML esperan recibir una única variable en forma de vector.

4.2 Pipeline

Se construye un `Pipeline` para automatizar todas las etapas del flujo de trabajo.

4.3 División de los datos en entrenamiento y test

Después de preparar los vectores asignamos aleatoriamente el 80% de los datos para entrenamiento y el 20% restante para test. Lo hacemos seleccionando atletas distintos, de manera que aquellos atletas que se utilicen para el entrenamiento no se encuentren en test.

4.4 Definición de la variable objetivo

Renombramos la columna de la variable objetivo (`duration`) como `label` para evitar posibles errores de nombración, ya que `pyspark.ml` coge por defecto ese nombre.

5 Modelado

5.1 Creación y entrenamiento de modelos

Para abordar el problema de predicción de rendimiento de los atletas decidimos hacer una regresión, donde la variable objetivo es la duración (`duration`) del entrenamiento. Para esto, entrenamos varios modelos disponibles de la librería MLlib:

- **Regresión Lineal:** Modelo base utilizado como referencia.
- **Árbol de Decisión:** Permite capturar relaciones no lineales entre las variables.
- **Random Forest Regressor:** Conjunto de árboles. Mejor la estabilidad y precisión.

Configuramos un CrossValidator que nos divide el conjunto de entrenamiento en 3 "folds" para usar 2 de ellos como entrenamiento y el otro como validación. Realizamos la búsqueda de hiperparámetros para encontrar cuáles eran los valores que nos iban a permitir crear un mejor modelo. Luego, entrenamos los modelos con los mejores hiperparámetros y vemos cual es el que nos proporciona mejores resultados.

Los mejores hiperparámetros para cada modelo fueron los siguientes:

Regresión Lineal	Valores
regParam	0.001
elasticNetParam	1.0
maxIter	50

Árbol de Decisión	Valores
maxDepth	15
minInstancesPerNode	5

Random Forest	Valores
numTrees	10
maxDepth	10
featureSubsetStrategy	auto

El modelo de regresión lineal es fácilmente interpretable y evalúa si un modelo lineal explica correctamente la relación entre variables de nuestro modelo. Un árbol de decisión puede modelar relaciones no lineales manteniendo la interpretabilidad. Por último, un modelo de Random Forest, que es una combinación de árboles que ayuda a mejorar la generalización de un único árbol, es decir, reduce el sobreajuste al mismo tiempo que captura relaciones más complejas.

En nuestro caso, se ve claramente el efecto de la linealidad en el dataset, ya que el modelo base de regresión lineal es el que proporciona mejores resultados. Esto sucede porque la relación entre nuestras variables es, por lógica, lineal, ya que cuanto más distancia se recorre, más tiempo se tarda en acabar el entrenamiento.

5.2 Evaluación de modelos

Los modelos se evaluaron usando la métrica `RMSE` (Root Mean Squared Error) y el coeficiente de determinación($\texttt{R}^2$), las adecuadas para problemas de regresión como el nuestro. En concreto, el RMSE lo utilizamos porque mide la desviación de la predicción en promedio y además, mantiene las unidades originales de medida. Por su parte, el R^2 mide la proporción de la variabilidad total de la variable dependiente que puede explicar el modelo.

En general, el R^2 mide la capacidad explicativa del modelo y complementa al RMSE, ya que mide qué proporción de la variabilidad total explica el modelo. Además, es independiente a la escala de medida.

5.3 Función para añadir ritmos

Creamos una función para mostrar unos resultados más completos. Esta realiza el cálculo del ritmo medio resultante, tanto sobre la duración predicha como sobre la duración real.

5.4 Resultados de predicciones

Table 2. Resultados proporcionados por el modelo.

athlete	distance	label	prediction	pace_real_min_km	pace_pred_min_km
1580	11.54	63.00	59.30	5.45	5.13
5518	13.20	57.45	68.56	4.35	5.19
9852	10.18	46.16	48.40	4.53	4.75
10623	20.60	107.00	118.49	5.19	5.75
11141	8.45	48.25	50.30	5.71	5.93

References

1. Kaggle: Long-distance running dataset, https://www.kaggle.com/datasets/mexwell/long-distance-running-dataset/data, el archivo dataset adecuado es el "run_ww_2020_d.csv".