Documentación de Prueba Técnica 5

Prueba

Construye un sistema de recomendación usando PySpark. Para la lógica del sistema puedes emplear métodos de factorización de matrices o algún otro algoritmo que consideres pertinente. Altamente recomendable usar datos abiertos gubernamentales. En tu script debes indicar claramente la fuente de datos que estás utilizando.

Solución

El Dataset seleccionado fue https://datos.gob.mx/busca/dataset/total-de-exposiciones-temporales-del-inba-en-el-ano-actual para entrenar el modelo para recomendar museos.

Código

Se importan las clases necesarias de PySpark.

```
from pyspark.sql import SparkSession
from pyspark.ml.recommendation import ALS
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.feature import StringIndexer
```

Se crea una sesión de Spark.

```
spark = SparkSession.builder.appName("RecomendacionExposiciones").getOrCreate()
```

Se especifica la ruta del conjunto de datos de exposiciones.

```
dataset_path = "exposiciones.csv"
```

Se lee el conjunto de datos CSV y se infiere el esquema automáticamente.

```
data = spark.read.csv(dataset_path, header=True, inferSchema=True)
```

Se convierten las columnas de fecha a tipo date.

```
data = data.withColumn("Fecha inicio", data["Fecha inicio"].cast("date"))
data = data.withColumn("Fecha fin", data["Fecha fin"].cast("date"))
```

Se especifican las columnas de exposiciones para convertirlas al tipo double.

```
exposiciones_cols = ['Exposiciones permanentes', 'Exposiciones temporales', 'Exposiciones itinerantes nacionales',
                     'Exposiciones internacionales en México', 'Exposiciones nacionales en el extranjero',
                     'Total de exposiciones', 'Total de asistentes']
for col in exposiciones_cols:
    data = data.withColumn(col, data[col].cast("double"))
```

Se aplica la indexación de cadenas a la columna 'Museo' para convertirla en un índice numérico.

```
indexer = StringIndexer(inputCols=['Museo'], outputCols=['MuseoIndex'])
data_indexed = indexer.fit(data).transform(data)
```

Se divide el conjunto de datos en conjuntos de entrenamiento y prueba en una proporción de 80:20.

```
(training, test) = data_indexed.randomSplit([0.8, 0.2])
```

Se configura y entrena el modelo ALS (Alternating Least Squares) utilizando la columna 'MuseoIndex' como usuario, 'MuseoIndex' como elemento, y 'Total de asistentes' como valor de clasificación.

```
als = ALS(maxIter=5, regParam=0.01, userCol="MuseoIndex", itemCol="MuseoIndex", ratingCol="Total de asistentes",
          coldStartStrategy="drop")
model = als.fit(training)
```

Se generan predicciones en el conjunto de prueba utilizando el modelo entrenado.

predictions = model.transform(test)

Se evalúa el modelo calculando el error cuadrático medio (RMSE) entre las etiquetas reales y las predicciones.

```
evaluator = RegressionEvaluator(metricName="rmse", labelCol="Total de asistentes", predictionCol="prediction")

rmse = evaluator.evaluate(predictions)

print("Root Mean Squared Error (RMSE) = " + str(rmse))
```

Se generan recomendaciones para todos los usuarios utilizando el modelo entrenado.

```
user_recs = model.recommendForAllUsers(5)
```