Práctica evaluable de Spark

Grupo

Alumno: Adolfo González Blázquez
Email: adolfo.gonzalez@elcorteingles.es
Teléfono: +34 609 964 414

Alumno: José María Álvarez Fernández
Email: josemaria.alvarezfernandez@elcorteingles.es
Teléfono: +34 682 780 953

Alumno: César Colado Rodríguez
email: cesar.colado@elcorteingles.es
Telefono: +34 661 415 555

Enunciado

Desarrollo de una solución basada en Spark con ML (regresión/clustering) y Streaming. La parte de Streaming se usará para enviar los datos necesarios para hacer la predicción con el modelo entrenado mediante Spark ML (MLlib o ML Pipelines). Se deberá entregar:

Hemos optado por la opción 2 de la práctica final de Spark. El entregable consta de las siguientes partes:

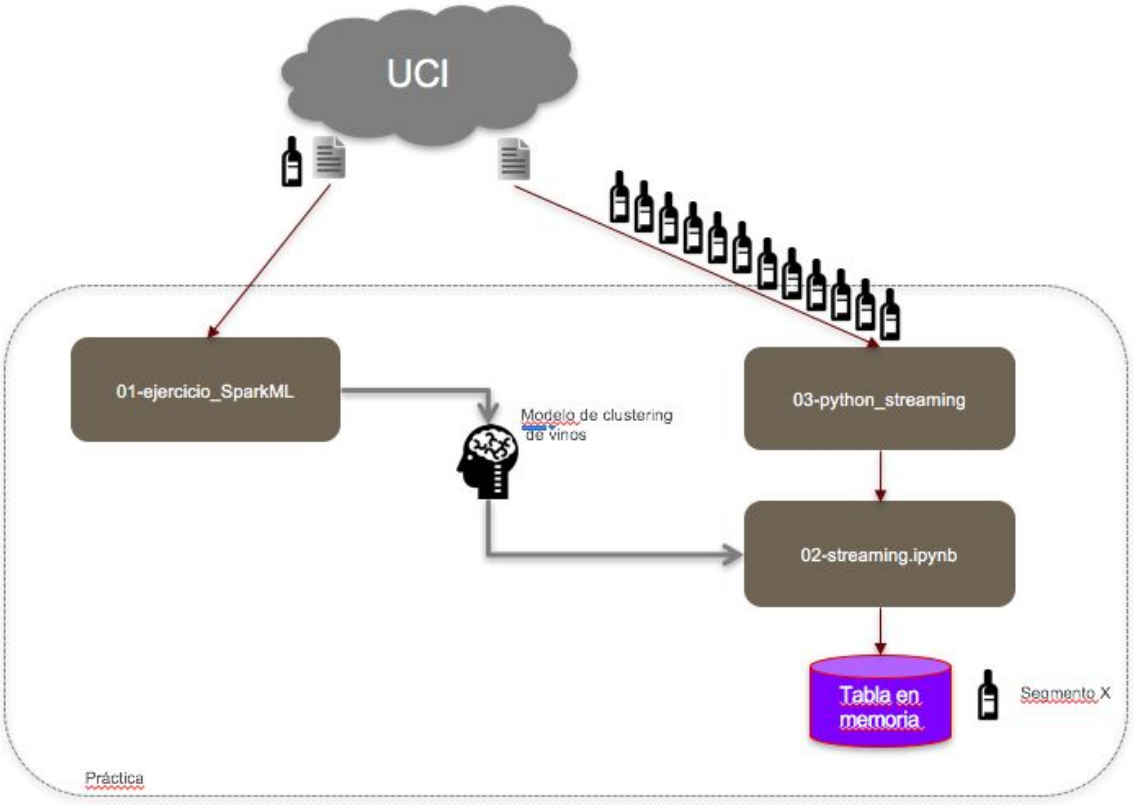
1) Notebook con la solución desarrollada para el entrenamiento del modelo en Spark:
   01-ejercicio_SparkML.ipynb

2) Los datos usados para el entrenamiento/test (validación y precisión del modelo) están disponibles en la siguiente URL:
   http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv

3) Un documento descriptivo de la solución desarrollada:
   04-PracticaFinalDeSpark-Memoria.pdf

4) Un script/notebook Python (ver el ejemplo de Streaming hecho en clase) que se encargue de generar los datos para la predicción:
   03-python_streaming.py y 02-streaming.ipynb

Propuesta



Generación del modelo

Hemos elegido un dataset sobre segmentación de vinos de UCI.

http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/
El objetivo será generar un modelo de segmentación, que nos permita agrupar los vinos que nos van llegando a un congreso de vinos. La cantidad de vinos que llegan de cada uno de los fabricantes, nos podría obligar a ir sometiendo cada vino al modelo diseñado y solicitar al productor la exposición de sus vinos en las categorías que se le asignen.

Para hacer esto, modelaremos la llegada de la información de cada vino mediante un "streaming" que hemos realizado en un script en python.

Esta aproximación a la segmentación de vinos nos permitiría por ejemplo hacer lo mismo con datos de cliente, segmentando clientes según entran en los centros comerciales y aplicándoles promociones diferenciadas según el segmento al que pertenecerían.

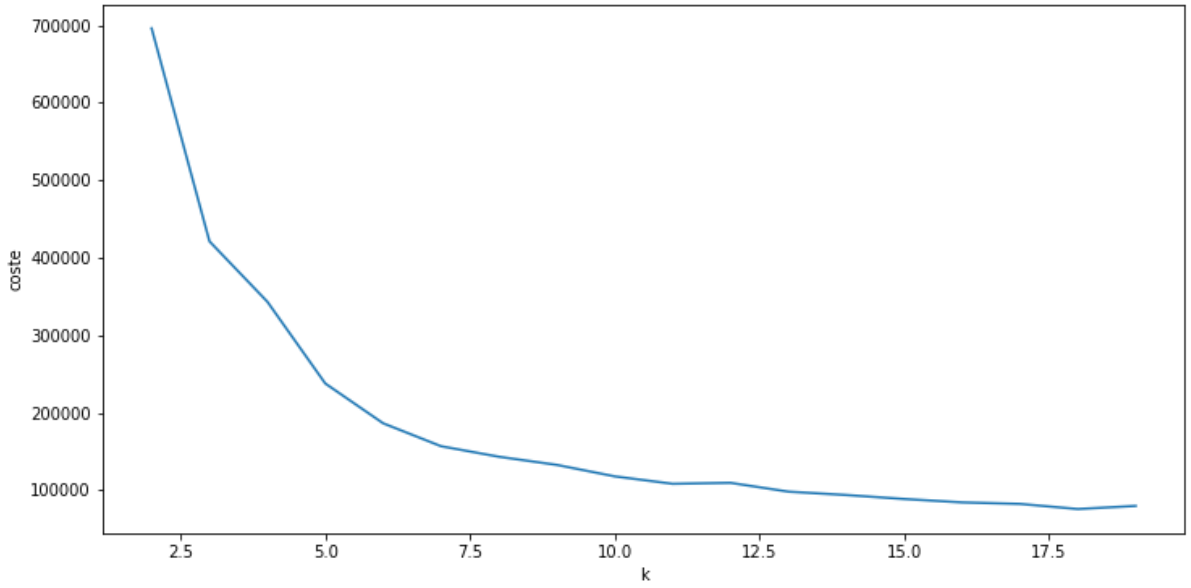
En el notebook ("01-ejercicio_SparkML.ipynb") generamos el modelo que almacenamos en formato "parquet" en el directorio "wine_kmeans_model" para su posterior aplicación en la clusterización de vinos cuyas características se evalúan en "streaming".

Como se puede observar, entendemos que estamos calculando la calidad del modelo valorando la distancia a los centroides de cada cluster. Como vemos cuantos más cluster generamos, reducimos las distancias y por tanto parece que se mejora el modelo.

El compromiso lo tenemos en k=11, como hemos comentado anteriormente, que es donde encontramos la maximización del beneficio en añadir más clusters.

k=11. Within Set Sum of Squared Errors = 99494.80305208372
k=6. Within Set Sum of Squared Errors = 193423.65775839088
k=15. Within Set Sum of Squared Errors = 73612.36286295216

Si bien es cierto que aunque seguimos mejorando conforme aumentamos el número de clusters a generar, hemos elegido 11, tal y como se comentó anteriormente porque es donde la ganancia es mayor como se puede ver en la siguiente gráfica.



En Spark 2.3 aparece disponible un "clustering evaluator" que no podemos utilizar, dado que este ejercicio lo estamos resolviendo con Spark 2.2.1

Streaming y aplicación del modelo

Para la carga el entrenamiento se usó un dataset de vinos tintos. Ahora para la carga en streaming se usará el de vinos blancos, y se intentará predecir con los clusters que se sacaron para los vinos tintos. Si lo que dicen los sumillers es cierto, los tintos y los blancos son vinos muy diferentes, pero hemos querido probar esto como una manera más de intentar hacer esto divertido. Evidentemente no esperamos grandes resultados, pero si ejemplificar que hemos entendido la materia.

La alternativa sería hacerlo con los mismo datos de entrenamiento o un subconjunto del dataset que hemos utilizado como generador del modelo. No obstante, como no somos capaces de hacer una evaluación objetiva, no sabríamos si el cluster asignado al ejemplar evaluado es bueno o malo.

La evaluación del modelo sobre datos en streaming se ha hecho utilizando el API de Structured Streaming, en vez del Spark Streaming convencional, dado que resultaba realmente sencillo el tratamiento directo de un DataFrame sobre el modelo, en vez de un DStream.

Para intentar cargar los datos con un schema ya conocido, se ha utilizado la posibilidad de cargar ficheros CSV en stream, y se ha simulado la carga de datos continua a través de la generación de ficheros cada segundo.

Se ha creado un programa en python (python_streaming.py) que carga el fichero desde Internet (http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv), y lo va introduciendo en ficheros en un directorio, cada fila en un fichero, y crea uno cada segundo.

Una vez aplicado el modelo en streaming, se guarda la clasificación del vino en una tabla en memoria que se va rellenando y en el notebook se muestra cómo se va rellenando durante 20 segundos. Viéndose a continuación cómo se va rellenando. En la primera vista, todavía no había llegado a evaluarse el vino 9 y el vino 10. En la segunda muestra de la tabla, ya había llegado en streaming la información necesaria para clasificar los primeros diez vinos. Si siguiéramos viendo la tabla, veríamos el resto de vinos clasificados.

```
+-----+----------+
|   id|prediction|
+-----+----------+
|wine4|         5|
|wine3|        10|
|wine2|         5|
|wine1|         5|
|wine5|         5|
|wine6|        10|
|wine7|         5|
|wine8|         5|
+-----+----------+

+------+----------+
|    id|prediction|
+------+----------+
| wine4|         5|
| wine3|        10|
| wine2|         5|
| wine1|         5|
| wine5|         5|
| wine6|        10|
| wine7|         5|
| wine8|         5|
| wine9|         5|
|wine10|         5|
+------+----------+
```

El script de python 3 para insertar los ficheros es como se adjunta a continuación:

```
# coding: utf-8

import socket, time, csv, urllib, io, os, logging
print("Started...")
import urllib.request

url =
"http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-
white.csv"
data = urllib.request.urlopen(url)
try:
    os.makedirs("./data")
except:
    logging.info("Directory exists, continuing")

remote_file = csv.reader(io.TextIOWrapper(data), delimiter=";")
i = 0
for row in remote_file:
  if i == 0:
     i+=1
     continue
  print(row)
  f = open("./data/data_"+str(i)+".csv",'w')
  row.append("wine" + str(i))
  f.write(";".join(row)) #Give your csv text here.
  f.close()
  time.sleep(1.0)
  i+=1
```