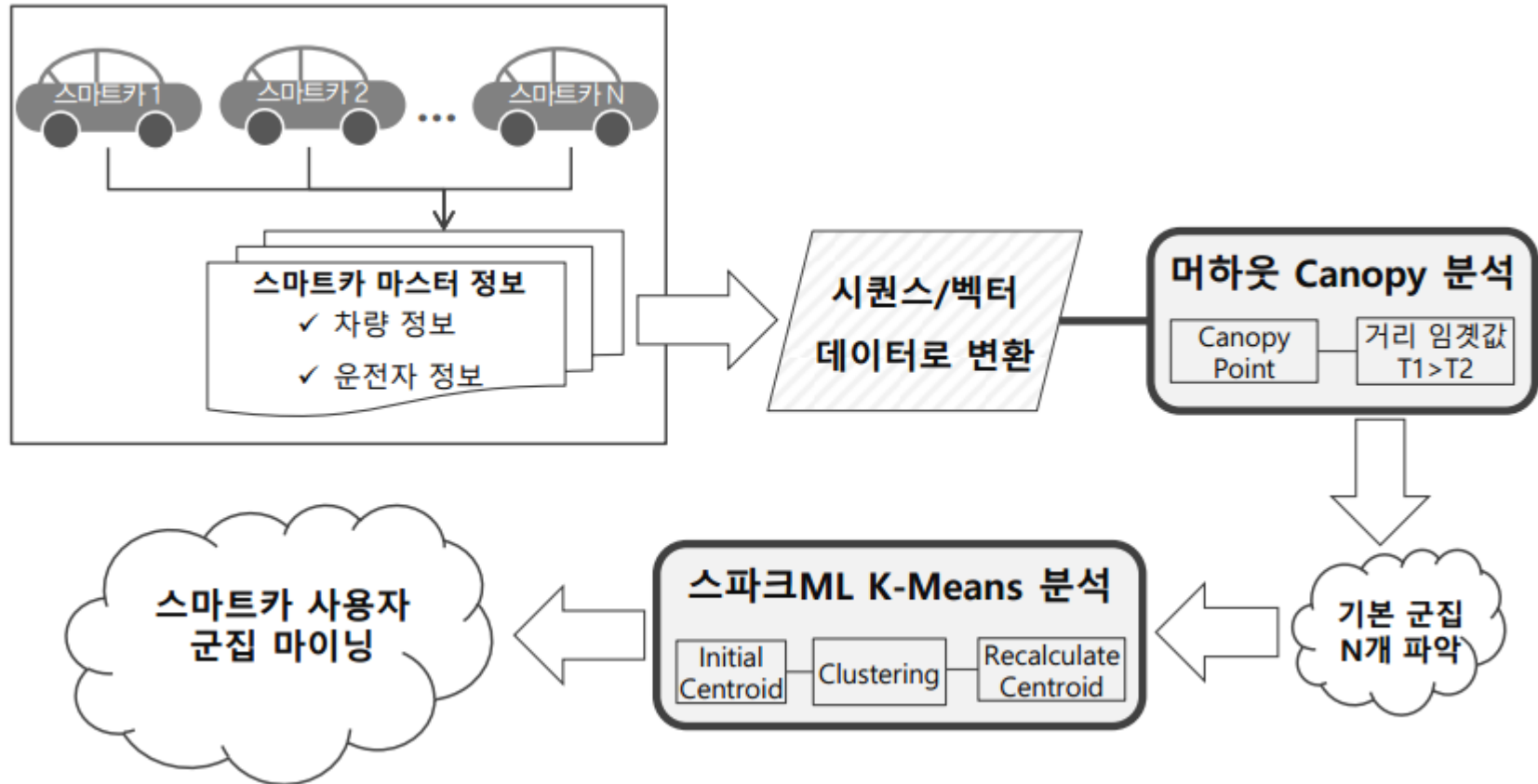


# 7.7 분석 파일럿 실행 5단계 - 머하웃&스파크 ML

## 스마트카 고객정보 군집 분석2



## ☆ 7.7 분석 파일럿 실행 5단계 - 머하웃&스파크 ML

### ☞ 스마트카 고객정보 군집 분석2

13. Canopy 군집분석에서 나온 클러스터 수(148)를 참고해서 다음에 진행할 K-Means 군집의 K값을 “148”보다 좀 더 큰 값인 “200”으로 확정한다. K-Means 분석은 제플린에서 스파크ML을 이용한다. 먼저 제플린이 종료됐다면 다음 명령어를 통해 제플린 서버를 실행하고, 크롬 브라우저를 통해 제플린 웹IDE에 접속한다.

```
$ zeppelin-daemon.sh restart
```

- 제플린 웹IDE URL: <http://server02.hadoop.com:8081/>

14. 제플린 상단 메뉴의 [Notebook] → [Create new note]를 선택하고 [Note Name]은 “SmartCar-Clustering”으로 입력하고, [Default Interpreter]는 “spark”를 선택한 후 [Create] 버튼을 클릭한다.

# 7.7 분석 파일럿 실행 5단계 - 머하웃&스파크 ML

## 스마트카 고객정보 군집 분석2

15. 첫 번째 입력창(Paragraph)에 사용할 라이브러리를 다음과 같이 입력하고 Shift+Enter 또는 우측 상단의 [Run] 버튼으로 실행한다. 관련 전체 소스코드는 C://예제소스/bigdata2nd-master/CH07/SparkML/SmartCar-Clustering.scala를 참고한다.

The image shows the Zeppelin Notebook interface. The top bar includes the Zeppelin logo, 'Notebook' and 'Job' tabs, a search bar, and a user dropdown set to 'anonymous'. The notebook title is 'SmartCar-Clustering'. Below the title is a toolbar with icons for play, zoom, view, edit, and download. The main area contains a code block with the following Scala imports:

```
import org.apache.spark.ml.feature.MinMaxScaler
import org.apache.spark.ml.feature.StringIndexer
import org.apache.spark.ml.clustering.KMeansModel
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.clustering.KMeans
import org.apache.spark.ml.evaluation.ClusteringEvaluator
```

On the right side of the code block, the status is 'FINISHED' and there are icons for play, zoom, view, and settings.

# 7.7 분석 파일럿 실행 5단계 - 머하웃&스파크 ML

## 스마트카 고객정보 군집 분석2

16. 하이브에서 생성해 둔 스마트카 마스터 데이터셋을 로드하고, 결과를 확인하기 위해 다음과 같이 코드를 입력한 후 실행 버튼을 클릭한다. 5개의 데이터가 조회된다.

```
val ds = spark.read.option("delimiter", " ")
                    .csv("/pilot-pjt/mahout/clustering/input/smartcar_master.txt")
ds.show(5)
```

FINISHED ▶ ⌘ 📖 ⚙

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| _c0|_c1|_c2|_c3|_c4|_c5|_c6|_c7|_c8|
+-----+-----+-----+-----+-----+-----+-----+-----+
|A0001|소형|NORMAL|F|여|30|미혼|프리랜서|서울|
|A0002|중형|NEW|A|남|50|미혼|주부|충남|
|A0003|중형|NEW|B|여|60|기혼|회사원|대전|
|A0004|중형|NORMAL|D|남|30|미혼|공무원|광주|
|A0005|소형|OLD|C|남|60|미혼|공무원|대구|
+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

# 7.7 분석 파일럿 실행 5단계 - 머하웃&스파크 ML

## 스마트카 고객정보 군집 분석2

17. 다음 코드를 실행해 스마트카 마스터 데이터셋의 컬럼명, 타입 등을 재구성한다.

```
val dsSmartCar_Master_1 = ds.selectExpr(  
    "cast(_c0 as string) car_number",  
    "cast(_c1 as string) car_capacity",  
    "cast(_c2 as string) car_year",  
    "cast(_c3 as string) car_model",  
    "cast(_c4 as string) sex",  
    "cast(_c5 as string) age",  
    "cast(_c6 as string) marriage",  
    "cast(_c7 as string) job",  
    "cast(_c8 as string) region"  
)
```

FINISHED ▶ ⌵ 📖 ⚙️

# 7.7 분석 파일럿 실행 5단계 - 머하웃&스파크 ML

## 스마트카 고객정보 군집 분석2

18. 다음 작업을 통해 문자형 카테고리 칼럼을 숫자형 칼럼으로 생성해 새로운 칼럼으로 추가한다.

FINISHED ▷

```
val dsSmartCar_Master_2 = new StringIndexer().setInputCol("car_capacity").setOutputCol("car_capacity_n")
    .fit(dsSmartCar_Master_1).transform(dsSmartCar_Master_1)
val dsSmartCar_Master_3 = new StringIndexer().setInputCol("car_year").setOutputCol("car_year_n")
    .fit(dsSmartCar_Master_2).transform(dsSmartCar_Master_2)
val dsSmartCar_Master_4 = new StringIndexer().setInputCol("car_model").setOutputCol("car_model_n")
    .fit(dsSmartCar_Master_3).transform(dsSmartCar_Master_3)
val dsSmartCar_Master_5 = new StringIndexer().setInputCol("sex").setOutputCol("sex_n")
    .fit(dsSmartCar_Master_4).transform(dsSmartCar_Master_4)
val dsSmartCar_Master_6 = new StringIndexer().setInputCol("age").setOutputCol("age_n")
    .fit(dsSmartCar_Master_5).transform(dsSmartCar_Master_5)
val dsSmartCar_Master_7 = new StringIndexer().setInputCol("marriage").setOutputCol("marriage_n")
    .fit(dsSmartCar_Master_6).transform(dsSmartCar_Master_6)
val dsSmartCar_Master_8 = new StringIndexer().setInputCol("job").setOutputCol("job_n")
    .fit(dsSmartCar_Master_7).transform(dsSmartCar_Master_7)
val dsSmartCar_Master_9 = new StringIndexer().setInputCol("region").setOutputCol("region_n")
    .fit(dsSmartCar_Master_8).transform(dsSmartCar_Master_8)
```

## 7.7 분석 파일럿 실행 5단계 - 머하웃&스파크 ML

### 스마트카 고객정보 군집 분석2

19. 앞선 머하웃의 Canopy 분석의 결과 중 스마트카 마스터 데이터에서 유효한 변수(차량용량, 차량연식, 차량모델, 운전자성별, 운전자결혼여부)만 선정해 클러스터링의 Features 변수로 사용한다.

```
val cols = Array("car_capacity_n", "car_year_n", "car_model_n", "sex_n", "marriage_n")  
  
val dsSmartCar_Master_10 = new VectorAssembler().setInputCols(cols).setOutputCol("features")  
                                .transform(dsSmartCar_Master_9)  
val dsSmartCar_Master_11 = new MinMaxScaler().setInputCol("features").setOutputCol("scaledFeatures")  
                                .fit(dsSmartCar_Master_10).transform(dsSmartCar_Master_10)
```

FINISHED

# 7.7 분석 파일럿 실행 5단계 - 머하웃&스파크 ML

## 스마트카 고객정보 군집 분석2

20. 기존 필드를 삭제하고 재구성된 스마트카 마스터 데이터셋을 확인해 본다. 이제 스파크ML을 이용해 K-Means 군집 분석을 하기 위한 데이터 전처리 작업이 끝났다.

FINISHED ▶ ✖ 📖 ⚙

```
val dsSmartCar_Master_12 = dsSmartCar_Master_11.drop("car_capacity").drop("car_year")
                                                    .drop("car_model").drop("sex")
                                                    .drop("age").drop("marriage")
                                                    .drop("job").drop("region")
                                                    .drop("features")
                                                    .withColumnRenamed("scaledfeatures", "features")

dsSmartCar_Master_12.show(5)
```

```
val Array(trainingData, testData) = dsSmartCar_Master_12.randomSplit(Array(0.7, 0.3))
```

car_number	car_capacity_n	car_year_n	car_model_n	sex_n	age_n	marriage_n	job_n	region_n	features
A0001	0.0	2.0	0.0	0.0	4.0	0.0	3.0	14.0	[0.0,1.0,0.0,0.0,...]
A0002	1.0	1.0	1.0	1.0	2.0	0.0	2.0	12.0	[0.5,0.5,0.142857...]
A0003	1.0	1.0	5.0	0.0	3.0	1.0	5.0	2.0	[0.5,0.5,0.714285...]
A0004	1.0	2.0	6.0	1.0	4.0	0.0	6.0	1.0	[0.5,1.0,0.857142...]
A0005	0.0	0.0	7.0	1.0	3.0	0.0	6.0	0.0	[0.0,0.0,1.0,1.0,...]

only showing top 5 rows



# 7.7 분석 파일럿 실행 5단계 - 머하웃&스파크 ML

## 스마트카 고객정보 군집 분석2

21. 다음 코드로 K-Means 군집 분석을 실행한다. 군집(클러스터)의 개수는 앞선 머하웃의 Canopy 분석에서 얻은 결과에 따라 200을 설정한다.

FINISHED ▶ ⌵ 📖 ⚙

```
val kmeans = new KMeans()  
  .setSeed(1L)  
  .setK(200)  
  .setFeaturesCol("features")  
  .setPredictionCol("prediction")  
val kmeansModel = kmeans.fit(dsSmartCar_Master_12)
```

kmeans: org.apache.spark.ml.clustering.KMeans = kmeans\_c19ec4c60291

kmeansModel: org.apache.spark.ml.clustering.KMeansModel = kmeans\_c19ec4c60291

Took 21 sec. Last updated by anonymous at April 16 2020, 4:11:47 PM. (outdated)

## 스마트카 고객정보 군집 분석2

22. K-Means 군집의 결과를 다음 코드를 실행해 확인해 본다. 출력된 결과 중 prediction 필드는 군집 번호로서 0 번~199번까지 총 200개의 군집을 나타내며, car\_number 필드는 각 군집번호에 포함된 차량번호를 나타낸다.

FINISHED    

```
val transKmeansModel = kmeansModel.transform(dsSmartCar_Master_12)
transKmeansModel.groupBy("prediction")
                  .agg(collect_set("car_number").as("car_number"))
                  .orderBy("prediction").show(200, false)
```

```

+-----+
+-----+
|prediction|car_number
|
+-----+
+-----+
|0          |[C0091, K0023, B0016, J0071, F0030, K0004, W0032, H0093, Y0009, G0080, S0009]
|
|1          |[H0008, R0078, E0033, B0008, M0028, M0003, D0061, T0026, X0055, L0069, I0001, J0015,
|
|2          |[K0010, M0033, Y0048, Z0066, T0045]
|
|3          |[N0001, Q0027, Z0042, J0096, D0076, I0003, W0077, H0081, M0039, V0035, J0010, G0010,
|
|4          |[D0083, Z0089, P0039, T0007, U0017, J0006, B0007, M0073, G0050, R0021, S0048, F0011,
|
|5          |[Y0099, F0035, V0080, U0066, N0009, W0063, L0009, T0040]

```

# 7.7 분석 파일럿 실행 5단계 - 머하웃&스파크 ML

## 스마트카 고객정보 군집 분석2

23. 군집 모델의 정확도를 평가하기 위해 다음 코드를 실행해 평균 실루엣(Silhouette) 스코어를 확인해 본다. 실루엣 스코어는 -1~1의 값을 가지며 각각 다음과 같은 의미를 가진다.

- -1에 가까운 값: 잘못된 군집에 포함된 개체가 많음
- 0에 가까운 값: 군집에 포함되지 않은 개체가 많음
- 1에 가까운 값: 군집에 포함된 개체가 많음

즉, 1에 가까운 값일수록 좋은 군집 모형일 가능성이 높다. 그림 7.104를 보면 실루엣 스코어가 0.85로, 비교적 좋은 군집 모델로 평가될 것으로 예상된다.

```
val evaluator = new ClusteringEvaluator()
val silhouette = evaluator.evaluate(transKmeansModel)

println(s"Silhouette Score = $silhouette")
```

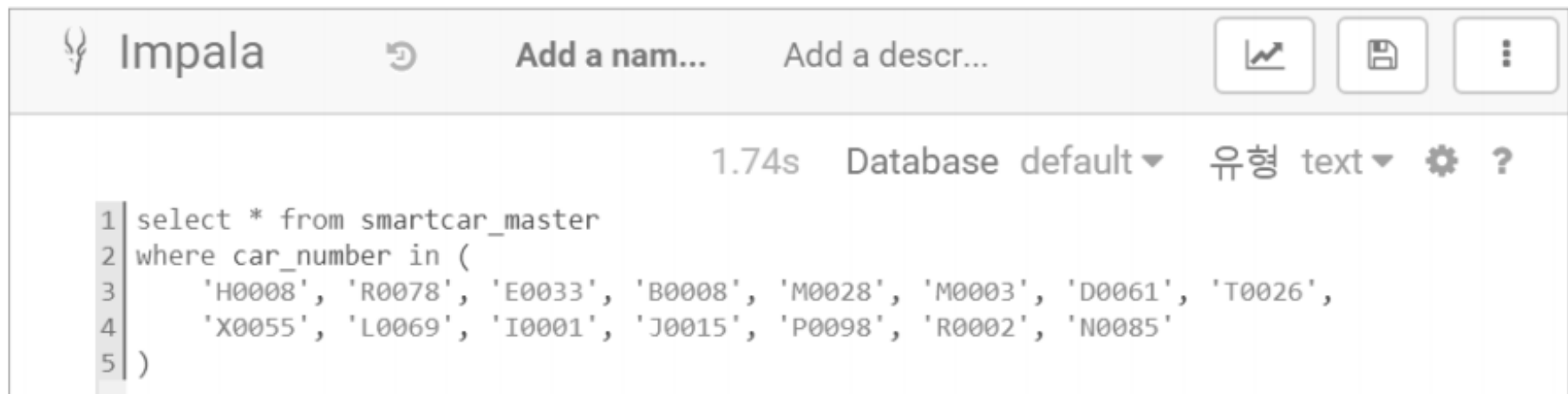
```
Silhouette Score = 0.8586774865237091
evaluator: org.apache.spark.ml.evaluation.ClusteringEvaluator = cluEval_fe036492c0b5
silhouette: Double = 0.8586774865237091
```

FINISHED ▶ 🔍 📖 ⚙️

# 7.7 분석 파일럿 실행 5단계 - 머하웃&스파크 ML

## 스마트카 고객정보 군집 분석2

24. 군집된 결과를 데이터 탐색 과정을 통해 좀 더 분석해 보자. 앞서 군집 결과인 그림 7.103을 보면 1번 군집에 15개의 차량 번호가 군집됐다. 15개의 차량 번호를 임팔라의 쿼리로 조회해서 어떤 특징이 있는지 확인해 본다. 휴를 실행해 상단의 쿼리콤보에서 [편집기] → [Impala]를 선택한 후 임팔라 편집기에서 다음 쿼리를 실행한다.



The image shows the Impala query editor interface. At the top, there's a header bar with the Impala logo, a refresh icon, and fields for 'Add a nam...' and 'Add a descr...'. To the right are icons for a chart, a save icon, and a menu icon. Below the header, the query execution time '1.74s' is displayed, followed by 'Database default' and '유형 text'. The main area contains a SQL query:

```
1 select * from smartcar_master
2 where car_number in (
3     'H0008', 'R0078', 'E0033', 'B0008', 'M0028', 'M0003', 'D0061', 'T0026',
4     'X0055', 'L0069', 'I0001', 'J0015', 'P0098', 'R0002', 'N0085'
5 )
```

# 7.7 분석 파일럿 실행 5단계 - 머하웃&스파크 ML

## 스마트카 고객정보 군집 분석2

	car_number	sex	age	marriage	region	job	car_capacity	car_year	car_model
1	B0008	남	61	기혼	경북	개인사업	1200	2002	F
2	D0061	남	48	기혼	전남	회사원	1000	2003	F
3	E0033	남	44	기혼	울산	공무원	1000	2004	F
4	H0008	남	41	기혼	경남	회사원	1200	2005	F
5	I0001	남	69	기혼	대구	주부	1500	2003	F
6	J0015	남	21	기혼	경북	학생	1000	2006	F
7	L0069	남	69	기혼	제주	프리랜서	1500	2003	F
8	M0003	남	17	기혼	서울	전문직	1500	2004	F
9	M0028	남	55	기혼	인천	프리랜서	1000	2001	F
10	N0085	남	53	기혼	대구	공무원	1000	2006	F
11	P0098	남	31	기혼	대구	주부	1500	2000	F
12	R0002	남	69	기혼	충남	공무원	1700	2000	F
13	R0078	남	23	기혼	충남	주부	1000	2003	F
14	T0026	남	51	기혼	부산	전문직	1500	2005	F
15	X0055	남	63	기혼	대전	학생	1500	2006	F

## ☆ 7.7 분석 파일럿 실행 5단계 - 머하웃&스파크 ML

### ☞ 스마트카 고객정보 군집 분석2

25. 조회 결과인 그림 7.106을 보면서 군집 1번의 특징을 파악해 보자.

- 고객 성향: 1번 군집은 50~60대의 기혼 남성들로, 스마트카 F모델 차량을 선호한다.
- 소득 추정: 1번 군집에 속한 고객군의 차량은 1500 CC 이하의 차량으로, 소득 수준이 낮을 것으로 추정된다.
- 분석 결과: 스마트카를 신규 구입하거나 재구매할 가능성이 있는 50~60대 기혼 남성 고객을 대상으로 저가형 스마트카 F모델을 타깃 마케팅한다.

## 7.7 분석 파일럿 실행 5단계 - 머하웃&스파크 ML

### 스마트카 고객정보 군집 분석2

# 실습