20220504

K-Means분석

- K-Means분석은 제플린에서 스파크 ML을 이용한다.
  - 제플린이 종료됐다면 명령어를 통해 제플린 서버를 실행

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

  - 제플린 웹IDE: http://server02.hadoop.com:8081

- 제플린에 “SmartCar_Clustering”으로새로운 노트 생성
  - 사용할 라이브러리 임포트

```
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
```

  - 데이터셋 조회

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

  - 데이터셋 재구성

```
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"
                      )
```

  - 문자형 칼럼을 연속형(숫자형) 칼럼으로 변환 및 생성

```
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)
```

- 클러스터링에서 사용할 피처 변수 선택

```
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)
```

- 스마트카 마스터 데이터셋 확인 및 학습/검증 데이터 생성

```
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))
```

- 스파크ML에서 K-Means 군집 분석 실행

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

- 스파크ML에서의 K-Means 군집 결과 확인

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

- 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
- 스파크ML에서의 K-Means 군집 모델 평가 – 실루엣 스코어

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

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

- 임팔라를 이용한 군집 정보 조회
  - 휴 UI [편집기] → [Impala]

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

분석 파일럿 실행 6단계 - 스쿱을 이용한 분석 결과 외부 제공

- 빅데이터의 HDFS에 저장된 데이터를 외부 시스템의 저장소(RDBMS)로 제공할 수 있어야 하는데, 이때 스쿱을 이용할 수 있다.

- 반대로 외부 시스템의 저장소(RDBMS)에 있는 데이터를 HDFS로 가져오기를 할 때도 스쿱을 이용할 수 있다.
- **스쿱의 내보내기 기능**
  - PostgreSQL의 cloudera-scm 계정의 패스워드 확인

```
cat /var/lib/cloudera-scm-server-db/data/generated_password.txt
```

- **PostgreSQL DB 작업을 진행**
  - PostgreSQL 서버 접속

```
psql -U cloudera-scm -p 7432 -h localhost -d postgres
```

  - 테이블 생성(CREATE TABLE이라고 나올시 정상적으로 만들어짐)

```
postgres=# create table smartcar_symptom_info (
car_number varchar,
speed_p_avg varchar,
speed_p_symptom varchar,
break_p_avg varchar,
break_p_symptom varchar,
steer_a_cnt varchar,
steer_p_symptom varchar,
biz_date varchar
);
```

  - PostgreSQL JDBC 드라이버를 스쿱의 라이브러리 경로에 복사

```
cp /opt/cloudera/parcels/CDH/jars/postgresql-*, jar /opt/cloudera/parcels/CDH/lib/sqoop/lib
```

  - 스쿱 내보내기 명령

```
sqoop export --connect jdbc:postgresql://192.168.56.101:7432/postgres --username
cloudera-scm --password acjZt03D1Q --table smartcar_symptom_info --export-dir /user/hive/
warehouse/managed_smartcar_symptom_info
```

  - 스쿱 내보내기 기능을 실행한 결과 확인

```
select * from smartcar_symptom_info;
```

파이썬을 이용한 웹 상태 정보 수집

- 파이썬에서는 urlib모듈을 사용하여 정보 수집
  - urlib.request 모듈은 웹 사이트에 있는 데이터에 접근할 수 있는 기능을 제공
  - 인증, 리다이렉트, 쿠키 처럼 인터넷을 이용한 다양한 요청과 처리를 지원함.
- 파일을 다운로드 할 때 , urlib.request 모듈에 있는 urlretrieve()함수를 사용, 직접 다운로드 가능함
  - 절대경로 입력시 현제 위치 확인하는 명령어

```
print(os.getcwd())
```

  - 예시 코드

```
# 라이브러리
import urllib.request

# 접속 경로 및 저장 이름
url = "http://uta.pw/shodou/img/28/214.png"
savename= "test_download.png"

# 다운로드
urllib.reqeust.urlretrieve(url,savename)
```

- 실습1

```
## 실습1
# NAVER의 로고 다운 받기

#라이브러리
import urllib.request

# 접속 경로 및 저장 이름
url = "http://s.pstatic.net/static/www/mobile/edit/20220503/mobile_20183742258.gif"
savename = "test_download.gif"

# 다운로드
urllib.request.urlretrieve(url,"download\\"+savename)
```

- 실습2

```
## 실습2
# Naver로고 다운받기... open을 이용한 다운로드
# 저장 파일명... naver_log_open_use.확장자

# 라이브러리 임포트
import urllib.request

# 경로 및 저장 이름
url = "http://s.pstatic.net/static/www/mobile/edit/20220503/mobile_20183742258.gif"
savename = "naver_logo_open_use.gif"

# urlopen을 이용하여 데이터를 저장(버퍼)
iread = urllib.request.urlopen(url).read()

# 저장된 내용을 open을 이용하여 저장(binary값으로 처리 : b)
wread = open("download\\"+savename, "wb")
wread.write(iread)
wread.close()
```

- 매개변수를 이용한 다운로드

```
### 기상청 RSS : http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp
## 매개변수 : stnId - 기상 정보를 알고 싶은 지역을 지정

## 전국 - 108       서울,경기 - 109     강원도 - 105    충청북도 - 131
## 충청남도 - 133   전라북도 - 146      전라남도 - 156  경상북도 - 143
## 경상남도 - 159   제주도 - 184

import urllib.request
import urllib.parse

API = "https://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp"

# 매개변수를 URL 인코딩
values = {
    "stnId" : "184"
}

params = urllib.parse.urlencode(values)
print(params)

url = API + "?" + params

# 다운로드
data = urllib.request.urlopen(url).read()
text = data.decode("utf-8")
```

```
print(text)
```

- 실습3

```
## 입력값(지역번호)을받아서 해당 지역의 날씨정보를 받아오세요...

import urllib.request
import urllib.parse

API = "https://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp"

# 매개변수를 URL 인코딩
num = input("지역번호를 입력하세요 : ")
values = {
    "stnId" : num
}

params = urllib.parse.urlencode(values)
print(params)

url = API + "?" + params

# 다운로드
data = urllib.request.urlopen(url).read()
text = data.decode("utf-8")

print(text)
```

- **selenium 사용하기**
  - 모듈 설치하기

```
pip install selenium
```

  - web드라이버 다운로드
    - http:/chromedriver.chromium.org/downloads 에 크롬에 맞는 정보에 해당하는 드라이버 다운
  - 제어

```
# 라이브러리
import selenium
from selenium import webdriver

# 패스 설정
path = "C:\\webdriver\\chromedriver"
# 웹드라이버를 이용한 크롬드라이버 실행
driver = webdriver.Chrome(path)

# 드라이버에서 url 정보를 얻어옴
driver.get("https://www.naver.com")

# class이름에 해당하는 정보를 담음
elements = driver.find_elements_by_class_name("link_txt")

# 반복문을 이용하여 해당 정보를 출력
for element in elements:
    print(element.text)
```