Spark 머신러닝

스파크 기반 머신러닝

- 분산 처리 환경을 기반으로 대규모 데이터셋에서도 비교적 빠른 속도로 머신러닝 알고리즘 훈련 및 적용

- 머신러닝 작업의 대부분을 한 곳에서 수행할 수 있는 통합 플랫폼 제공
  - 데이터를 즉시 모델 훈련과 평가에 활용하고 완성된 모델을 운영 환경에 즉시 적용
  - 모든 과정을 단일 API로 구현 가능

- 라이브러리
  - Mllib → RDD 기반 머신러닝 라이브러리
  - ML → DataFrame 기반 머신러닝 라이브러리

주요 객체

- Vectors
  - Double 형식의 값을 포함하는 컬렉션으로 구현
  - 포함된 데이터는 순서대로 0부터 시작하는 인덱스 부여
  - DenseVector, SparseVector로 구분

```
# dense vector

v1 = np.array([0.1, 0.0, 0.2, 0.3])
v2 = Vectors.dense([0.1, 0.0, 0.2, 0.3])

# sparse vector

v3 = Vectors.sparse(4, [(0, 0.1), (2, 0.2), (3, 0.3)])
v4 = Vectors.sparse(4, [0, 2, 3], [0.1, 0.2, 0.3])
```

```
[0.1,0.0,0.2,0.3]
(4,[0,2,3],[0.1,0.2,0.3])
(4,[0,2,3],[0.1,0.2,0.3])
```

주요 객체

- LabeledPoint
  - 지도 학습 데이터를 표현하기 위한 벡터
  - 특성 값을 담고 있는 벡터와 타겟 정보로 구성

```
v1 = np.array([0.1, 0.0, 0.2, 0.3])

v6 = LabeledPoint(1.0, v1)

print("label:%s, features:%s" % (v6.label, v6.features))
```

```
label:1.0, features:[0.1,0.0,0.2,0.3]
```

주요 객체

- Tokenizer
  - 공백 문자를 기준으로 입력 문자열을 개별 단어의 배열로 변환하고
  - 이 배열을 값으로 하는 새로운 컬럼을 생성하는 트랜스포머
  - 공백이 아닌 정규식으로 분할할 경우 RegexTokenizer 클래스 사용

```
data = [(0, "Tokenization is the process"), (1, "Refer to the Tokenizer")]
inputDF = spark.createDataFrame(data).toDF("id", "input")

tokenizer = Tokenizer(inputCol="input", outputCol="output")
outputDF = tokenizer.transform(inputDF)

outputDF.show(5, False)
```

```
+---+--------------------------+--------------------------------+
|id |input                     |output                          |
+---+--------------------------+--------------------------------+
|0  |Tokenization is the process|[tokenization, is, the, process]|
|1  |Refer to the Tokenizer    |[refer, to, the, tokenizer]     |
+---+--------------------------+--------------------------------+
```

주요 객체

- StringIndexer
  - 문자열 컬럼에 대응하는 숫자형 컬럼 생성

```
df1 = spark.createDataFrame([
    (0, "red"), (1, "blue"), (2, "green"), (3, "yellow")]).toDF("id", "color")

strignIndexer = StringIndexer(inputCol="color", outputCol="colorIndex").fit(df1)

df2 = strignIndexer.transform(df1)
df2.show()
```

```
+---+------+----------+
| id| color|colorIndex|
+---+------+----------+
|  0|   red|       3.0|
|  1|  blue|       0.0|
|  2| green|       2.0|
|  3|yellow|       1.0|
+---+------+----------+
```

주요 객체

- IndexToString
  - StringIndexer의 인코딩 결과를 원래 문자열로 복원하는 트랜스포머

```
df1 = spark.createDataFrame([
    (0, "red"), (1, "blue"), (2, "green"), (3, "yellow")]).toDF("id", "color")

strignIndexer = StringIndexer(inputCol="color", outputCol="colorIndex").fit(df1)

df2 = strignIndexer.transform(df1)
# df2.show()

indexToString = IndexToString(inputCol="colorIndex", outputCol="originalColor")

df3 = indexToString.transform(df2)
df3.show()
```

```
+---+------+----------+-------------+
| id| color|colorIndex|originalColor|
+---+------+----------+-------------+
|  0|   red|       3.0|          red|
|  1|  blue|       0.0|         blue|
|  2| green|       2.0|        green|
|  3|yellow|       1.0|       yellow|
+---+------+----------+-------------+
```

주요 알고리즘 구현

- 회귀 알고리즘 구현 도구
  - 설명변수를 통해 연속형 변수를 예측하는 모델

도구	설명
LinearRegression	연속형 변수를 예측하는 선형 회귀 모델 구현 학습(fit)의 결과는 LinearRegressionModel 객체
GeneralizedLinearRegression	잔차 분포가 정규 분포를 따르지 않는 경우의 선형 회귀 모델 구현
DecisionTreeRegressor	결정 트리 기반 회귀 알고리즘 구현
RandomForestRegressor	다수의 결정 트리를 결합한 트리의 앙상블 알고리즘 구현
GBTRegressor	Stochastic Gradient Boosting 기반의 결정 트리 앙상블 알고리즘 구현

주요 알고리즘 구현

- 분류 알고리즘 구현 도구
  - 설명 변수를 통해 범주형 변수를 예측하는 모델

도구	설명
LogisticRegression	확률 기반 선형 분류 알고리즘 구현
DecisionTreeClassifier	결정 트리 기반의 분류 알고리즘 구현
RandomForestClassifier	트리 앙상블 기반의 분류 알고리즘 구현
GBTClassifier	Stochastic Gradient Boosting 트리 앙상블 분류 알고리즘 구현
MultilayerPerceptronClassifier	다층 신경망 기반 분류 알고리즘 구현
OneVsRest	이진 분류 도구를 기반으로 다중 분류 알고리즘 구현
NaiveBayes	조건부 확률에 관한 베이즈 이론 기반의 분류 알고리즘 구현 Multinomial Naive Bayes, Bernoulli Naive Bayes 모델 선택 가능

주요 알고리즘 구현

- 클러스트링
  - 데이터의 유사도를 기반으로 데이터의 그룹을 분류하는 모델
  - 레이블 없이 학습하는 비지도 학습 알고리즘

도구	설명
KMeans	확률 기반 선형 분류 알고리즘 구현
GaussianMixture	전체 데이터셋을 다수의 가우시안 분포 합으로 분류

- 협업 필터링
  - 사용자 선호의 유사도를 기반으로 사용자의 관심사를 예측
  - 상품 추천 등의 추천 시스템에 많이 사용

도구	설명
ALS	사용자와 상품 사이의 평점 정보로 구성된 거대 희소 행렬에서 비어 있는 값을 찾는 알고리즘 중 가장 많이 사용되는 Alternating Least Squares 알고리즘 구현