BÁO CÁO LAB 03

CÀI ĐẶT CÁC THUẬT TOÁN MÁY HỌC BẰNG SPARKML API

GVLT: CÔ, NGUYỄN NGỌC THẢO

GVHD: THẦY, LÊ NGỌC THÀNH

THÔNG TIN NHÓM

Nhóm 1:

- Võ Nhật Vinh – 1612815
- Hồng Thanh Hoài – 1612855
- Huỳnh Minh Huấn – 1612858

Cài đặt chi tiết

Nhóm sử dụng spark trên windows, code trên môi trường jupyter lab.

1. Cài đặt thuật toán decision tree

- Khởi tạo spark session.

```
import findspark
findspark.init()
```

```
import pyspark # only run after findspark.init()
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
```

- Đọc tập dữ liệu và in ra mô tả của dữ liệu.

```
data = spark.read.csv("Absenteeism_at_work.csv", inferSchema=True, header = True, sep=",")
```

```
data.printSchema()
```

```
root
 |-- ID: integer (nullable = true)
 |-- Reason for absence: integer (nullable = true)
 |-- Month of absence: integer (nullable = true)
 |-- Day of the week: integer (nullable = true)
 |-- Seasons: integer (nullable = true)
 |-- Transportation expense: integer (nullable = true)
 |-- Distance from Residence to Work: integer (nullable = true)
 |-- Service time: integer (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Work load Average/day : double (nullable = true)
 |-- Hit target: integer (nullable = true)
```

- Thêm vào các cột với các tên rút gọn từ các cột gốc.

```
data = data.withColumn("MOA", data["Month of absence"]). \
            withColumn("label", data['Height']). \
            withColumn("ROA", data["Reason for absence"]). \
            withColumn("distance", data["Distance from Residence to Work"]). \
            withColumn("BMI", data["Body mass index"])
```

- Tạo các vector assembler, labelIndexer và featureIndexer.

```
from pyspark.ml.feature import VectorAssembler, StringIndexer, VectorIndexer
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml import Pipeline
```

```
assem = VectorAssembler(inputCols=['label', 'distance'], outputCol='features')
data = assem.transform(data)
```

```
labelIndexer = StringIndexer(inputCol='label', outputCol='indexedLabel')
```

```
featureIndexer = VectorIndexer(inputCol="features", outputCol="indexedFeatures", maxCategories=4)
```

- Tách tập dữ liệu thành 2 tập train và test.

```
(trainingData, testData) = data.randomSplit([0.7, 0.3])
```

- Tạo classifier với labelCol là “indexedLabel” và featuresCol là “indexedFeatures”. Tạo pipeline cho để build model.

```
dt_clf = DecisionTreeClassifier(labelCol="indexedLabel", featuresCol="indexedFeatures")
```

```
pipeline = Pipeline(stages=[labelIndexer, featureIndexer, dt_clf])
```

- Build model bằng phương thức fit trên tập train và dung model dự đoán trên tập test bằng phương thức transform.

```
model = pipeline.fit(trainingData)
```

```
# Make predictions.
predictions = model.transform(testData)
```

- Đánh giá kết quả dự đoán

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# Select (prediction, true label) and compute test error
evaluator = MulticlassClassificationEvaluator(labelCol="indexedLabel",\
                                              predictionCol="prediction",\
                                              metricName="accuracy")
```

```
accuracy = evaluator.evaluate(predictions)
```

- Kết quả

```
Decision Tree - Test Accuracy = 0.964286
Decision Tree - Test Error = 0.0357143
```

- Sử dụng thư viện sklearn để đó các độ đo khác như precision, recall

```
precision = precision_score(y_true, y_pred, average='micro')

recall = recall_score(y_true, y_pred, average='micro')
```

```
The precision score for Decision Tree Model is: 0.0297297
The recall score for Decision Tree Model is: 0.0297297297
```

2. Cài đặt thuật toán Naïve Bayes

- Đọc tập dữ liệu và in ra mô tả của dữ liệu.

```
data = spark.read.csv("Absenteeism_at_work.csv", inferSchema=True, header = True, sep=",")
```

```
data.printSchema()
```

```
root
 |-- ID: integer (nullable = true)
 |-- Reason for absence: integer (nullable = true)
 |-- Month of absence: integer (nullable = true)
 |-- Day of the week: integer (nullable = true)
 |-- Seasons: integer (nullable = true)
 |-- Transportation expense: integer (nullable = true)
 |-- Distance from Residence to Work: integer (nullable = true)
 |-- Service time: integer (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Work load Average/day : double (nullable = true)
 |-- Hit target: integer (nullable = true)
```

- Thêm vào các cột với các tên rút gọn từ các cột gốc.

```
data = data.withColumn("MOA", data["Month of absence"]). \
            withColumn("label", data['Height']). \
            withColumn("ROA", data["Reason for absence"]). \
            withColumn("distance", data["Distance from Residence to Work"]). \
            withColumn("BMI", data["Body mass index"])
```

- Tạo các vector assembler, labelIndexer và featureIndexer.

```
from pyspark.ml.feature import VectorAssembler
assem = VectorAssembler(inputCols=["label", "MOA"], outputCol='features')
```

```
data = assem.transform(data)
```

```
from pyspark.ml.feature import StringIndexer
labelIndexer = StringIndexer(inputCol="label", outputCol="indexedLabel")
```

```
from pyspark.ml.feature import VectorIndexer
featureIndexer = VectorIndexer(inputCol="features", outputCol="indexedFeatures", maxCategories=4)
```

- Chia tập dữ liệu thành 2 tập train và test.

```
# Split the data into training and test sets (30% held out for testing)
(trainingData, testData) = data.randomSplit([0.7, 0.3],seed=1)
```

- Tạo classifier cho RandomForest với labelCol là “indexedLabel” và featuresCol là “indexedFeatures”. Và pipeline để build model.

```
from pyspark.ml.classification import NaiveBayes
nb = NaiveBayes(labelCol="indexedLabel", featuresCol="indexedFeatures")
```

```
from pyspark.ml import Pipeline
# Chain indexers and forest in a Pipeline
pipeline = Pipeline(stages=[labelIndexer, featureIndexer, nb])#, labelConverter])
```

- Train model bằng phương thức “fit” và predict model bằng phương thức ‘transform”.

```
# train the model
model = pipeline.fit(trainingData)
```

```
# select example rows to display.
predictions = model.transform(testData)
```

- Đánh giá kết quả prediction.

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
# compute accuracy on the test set
evaluator = MulticlassClassificationEvaluator(labelCol="indexedLabel",\
                                              predictionCol="prediction",\
                                              metricName="accuracy")
```

```
accuracy = evaluator.evaluate(predictions)
```

- Kết quả:

```
Naive Bayes - Test Accuracy = 0.272321
Naive Bayes - Test Error = 0.727679
```

3. cài đặt thuật toán random forest

- Đọc dữ liệu và in Schema mô tả.

```
data = spark.read.csv("Absenteeism_at_work.csv", inferSchema=True, header = True, sep=",")
```

```
data.printSchema()
```

- Thêm vào các cột với các tên rút gọn từ các cột gốc.

```
data = data.withColumn("MOA", data["Month of absence"]). \
            withColumn("label", data['Height']). \
            withColumn("ROA", data["Reason for absence"]). \
            withColumn("distance", data["Distance from Residence to Work"]). \
            withColumn("BMI", data["Body mass index"])
```

- Tạo các vector assembler, labelIndexer và featureIndexer.

```
from pyspark.ml.feature import VectorAssembler
assem = VectorAssembler(inputCols=["label", "MOA"], outputCol='features')
```

```
data = assem.transform(data)
```

```
from pyspark.ml.feature import StringIndexer
labelIndexer = StringIndexer(inputCol="label", outputCol="indexedLabel")
```

```
from pyspark.ml.feature import VectorIndexer
featureIndexer = VectorIndexer(inputCol="features", outputCol="indexedFeatures", maxCategories=4)
```

- Chia tập dữ liệu thành 2 tập train và test.

```
(trainingData, testData) = data.randomSplit([0.7, 0.3], seed=1)
```

- Tạo classifier cho RandomForest với labelCol là “indexedLabel” và featuresCol là “indexedFeatures”. Và pipeline để build model.

```
from pyspark.ml.classification import RandomForestClassifier
# Train a RandomForest model.
rf = RandomForestClassifier(labelCol="indexedLabel", featuresCol="indexedFeatures", numTrees=10)
```

```
from pyspark.ml import Pipeline
pipeline = Pipeline(stages=[labelIndexer, featureIndexer, rf])
```

- Train model bằng phương thức “fit” và predict model bằng phương thức ‘transform”.

```
# Train model.  This also runs the indexers.
model = pipeline.fit(trainingData)
```

```
# Make predictions.
predictions = model.transform(testData)
```

- Đánh giá kết quả predictions

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
# Select (prediction, true label) and compute test error
evaluator = MulticlassClassificationEvaluator(labelCol="indexedLabel",\
                                              predictionCol="prediction",\
                                              metricName="accuracy")
```

```
accuracy = evaluator.evaluate(predictions)
```

- Kết quả:

```
Random Forest - Test Accuracy = 0.977679
Random Forest - Test Error = 0.0223214
```

4. Chạy với những tập dữ liệu khác

a. Thực hiện giải thuật Decision tree trên tập dữ liệu “Heathcare stroke” (nguồn Kaggle – có đính kèm trong thư mục nộp):

- Khởi tạo SparkSession và đọc dữ liệu. In ra Schema dữ liệu

Healthcare stroke dataset

```
[9]: spark = SparkSession.builder.appName('stroke').getOrCreate()
     train = spark.read.csv('train_2v.csv', inferSchema=True,header=True)
```

```
[10]: train.printSchema()
```

```
root
 |-- id: integer (nullable = true)
 |-- gender: string (nullable = true)
 |-- age: double (nullable = true)
 |-- hypertension: integer (nullable = true)
 |-- heart_disease: integer (nullable = true)
 |-- ever_married: string (nullable = true)
 |-- work_type: string (nullable = true)
 |-- Residence_type: string (nullable = true)
 |-- avg_glucose_level: double (nullable = true)
 |-- bmi: double (nullable = true)
 |-- smoking_status: string (nullable = true)
 |-- stroke: integer (nullable = true)
```

(dữ liệu có các cột gender, ever_married, work_type, Residence_type, smoking_status cần được chuẩn hóa về dạng số để thực hiện thuật toán. Ngoài ra, một số cột trong dữ liệu quan trọng nhưng vẫn có thể chứa giá trị null)

- Xem xét trên dữ liệu label cần phân lớp. Ta thấy rằng dữ liệu bị mất cân bằng (imbalanced data).

```
[12]: train.groupby('stroke').count().show()
```

```
+------+-----+
|stroke|count|
+------+-----+
|     1|  783|
|     0|42617|
+------+-----+
```

- Thực hiện filter các giá trị null trên 2 cột smoking_status ('No Info') và bmi (mean).

```
[22]: train_f = train.na.fill('No Info', subset=['smoking_status'])
# fill in miss values with mean
from pyspark.sql.functions import mean
mean = train_f.select(mean(train_f['bmi'])).collect()
mean_bmi = mean[0][0]
train_f = train_f.na.fill(mean_bmi,['bmi'])
```

- Xây dựng các bộ indexer (StringIndexer) và encoder (OneHotEncoder) đối với các cột có kiểu dữ liệu string (gender, ever_married, work_type, …) đã đề cập ở trên.

```
[23]: from pyspark.ml.feature import VectorAssembler,OneHotEncoder,StringIndexer
```

```
[24]: genderIndexer = StringIndexer(inputCol='gender', outputCol='indexedGender')
gender_encoder = OneHotEncoder(inputCol='indexedGender', outputCol='genderVec')
```

```
[25]: ever_marriedIndexer = StringIndexer(inputCol='ever_married', outputCol='indexedEver_married')
ever_married_encoder = OneHotEncoder(inputCol='indexedEver_married', outputCol='ever_marriedVec')
```

```
[26]: workTypeIndexer = StringIndexer(inputCol='work_type', outputCol='indexedWork_type')
workType_encoder = OneHotEncoder(inputCol='indexedWork_type', outputCol='workTypeVec')
```

```
[27]: ResidenceTypeIndexer = StringIndexer(inputCol='Residence_type', outputCol='indexedResidence_type')
ResidenceType_encoder = OneHotEncoder(inputCol='indexedResidence_type', outputCol='residenceTypeVec')
```

```
[28]: smoking_statusIdexer = StringIndexer(inputCol='smoking_status', outputCol='indexedSmoking_status')
smokingStatus_encoder = OneHotEncoder(inputCol='indexedSmoking_status', outputCol='smokingStatusVec')
```

- Tạo assembler vector (VectorAssembler) outputCol là 'features'

```
[29]: assembler = VectorAssembler(inputCols=['genderVec','age','hypertension','heart_disease', 'ever_marriedVec','worl
                                             ,'residenceTypeVec', 'avg_glucose_level', 'bmi', 'smokingStatusVec'],outp
```

- Khởi tạo classifier Decision tree với featuresCol là 'features' và labelCol là 'stroke'. Xây dựng Pipeline.

```
[30]: from pyspark.ml.classification import DecisionTreeClassifier
      dt_clf = DecisionTreeClassifier(featuresCol='features', labelCol='stroke')

[31]: from pyspark.ml import Pipeline
      pipeline = Pipeline(stages=[genderIndexer, ever_marriedIndexer, workTypeIndexer, ResidenceTypeIndexer,smoking_s
                                  gender_encoder, ever_married_encoder, workType_encoder, ResidenceType_encoder, smok
                                  assembler, dt_clf])
```

- Tách tập dữ liệu thành 2 tập train (train_data) và validation (val_data). Tạo model với tập train. Đánh giá trên tập validation.

```
[32]: train_data,val_data = train_f.randomSplit([0.8,0.2])

[33]: model = pipeline.fit(train_data)

[34]: validations = model.transform(val_data)
```

- Đánh giá kết quả model

```
[35]: from pyspark.ml.evaluation import MulticlassClassificationEvaluator
      # Select (prediction, true label) and compute test error
      acc_evaluator = MulticlassClassificationEvaluator(labelCol="stroke", predictionCol="prediction", metricName
      dtc_acc = acc_evaluator.evaluate(validations)
      print('A Decision Tree algorithm had an accuracy of: {0:2.2f}%'.format(dtc_acc*100))

      A Decision Tree algorithm had an accuracy of: 98.30%
```

b. Thực hiện giải thuật Random forest trên tập dữ liệu zoo

- Đọc tập dữ liệu và in schema mô tả dữ liệu.

```
zoo_data = spark.read.csv('zoo.csv', inferSchema=True,header=True)
```

```
data = zoo_data.drop('animal_name')
```

```
features.printSchema()
```

```
root
 |-- hair: integer (nullable = true)
 |-- feathers: integer (nullable = true)
 |-- eggs: integer (nullable = true)
 |-- milk: integer (nullable = true)
 |-- airborne: integer (nullable = true)
 |-- aquatic: integer (nullable = true)
 |-- predator: integer (nullable = true)
 |-- toothed: integer (nullable = true)
 |-- backbone: integer (nullable = true)
 |-- breathes: integer (nullable = true)
 |-- venomous: integer (nullable = true)
 |-- fins: integer (nullable = true)
 |-- legs: integer (nullable = true)
 |-- tail: integer (nullable = true)
 |-- domestic: integer (nullable = true)
 |-- catsize: integer (nullable = true)
 |-- class_type: integer (nullable = true)
```

- Tạo vector assembler "features"

```
assembler = VectorAssembler(inputCols=['hair', 'feathers', 'eggs', 'milk',\
                                       'airborne', 'aquatic', 'predator', 'toothed',\
                                       'backbone', 'breathes', 'venomous', 'fins',\
                                       'legs', 'tail', 'domestic', 'catsize'],outputCol='features')
```

- Tạo classifer RandomForest featuresCol=’features’ và labelCol=’class_type’. Tạo pipeline để train model. Train model bằng phương thức “fit” và lưu validation bằng phương thức “transform”.

```
from pyspark.ml.classification import RandomForestClassifier
rf_clf = RandomForestClassifier(featuresCol='features', labelCol='class_type')
```

```
pipeline = Pipeline(stages=[assembler, rf_clf])
```

```
model = pipeline.fit(train_data)
```

```
validations = model.transform(val_data)
```

- Kết quả

```
acc_evaluator = MulticlassClassificationEvaluator(labelCol="class_type", predictionCol="prediction", metricName="acc
dtc_acc = acc_evaluator.evaluate(validations)
print('A Decision Tree algorithm had an accuracy of: {0:2.2f}%'.format(dtc_acc*100))
```

```
A Decision Tree algorithm had an accuracy of: 100.00%
```

c. Diabetes

- Đọc dữ liệu và in ra schema dữ liệu

dataset: pima-indians-diabetes

link: https://www.kaggle.com/uciml/pima-indians-diabetes-database#diabetes.csv

```
data_diabetes = spark.read.csv('diabetes.csv', inferSchema=True,header=True)
```

```
data_diabetes.printSchema()
```

```
root
 |-- Pregnancies: integer (nullable = true)
 |-- Glucose: integer (nullable = true)
 |-- BloodPressure: integer (nullable = true)
 |-- SkinThickness: integer (nullable = true)
 |-- Insulin: integer (nullable = true)
 |-- BMI: double (nullable = true)
 |-- DiabetesPedigreeFunction: double (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Outcome: integer (nullable = true)
```

- Show dữ liệu để có cái nhìn tổng quát

```
data_diabetes.show()
```

```
+-----------+-------+-------------+-------------+-------+----+------------------------+---+-------+
|Pregnancies|Glucose|BloodPressure|SkinThickness|Insulin| BMI|DiabetesPedigreeFunction|Age|Outcome|
+-----------+-------+-------------+-------------+-------+----+------------------------+---+-------+
|          6|    148|           72|           35|      0|33.6|                   0.627| 50|      1|
|          1|     85|           66|           29|      0|26.6|                   0.351| 31|      0|
|          8|    183|           64|            0|      0|23.3|                   0.672| 32|      1|
|          1|     89|           66|           23|     94|28.1|                   0.167| 21|      0|
|          0|    137|           40|           35|    168|43.1|                   2.288| 33|      1|
|          5|    116|           74|            0|      0|25.6|                   0.201| 30|      0|
|          3|     78|           50|           32|     88|31.0|                   0.248| 26|      1|
|         10|    115|            0|            0|      0|35.3|                   0.134| 29|      0|
|          2|    197|           70|           45|    543|30.5|                   0.158| 53|      1|
|          8|    125|           96|            0|      0| 0.0|                   0.232| 54|      1|
```

- Thực hiện fill các giá trị ‘0’ tại các cột BloodPressure, ‘SkinThickness’, ‘Insulin’, ‘BMI’ bằng giá trị mean của mỗi cột.

```
mean = data_diabetes.agg({'BloodPressure': 'mean', 'SkinThickness': 'mean', 'Insulin': 'mean', 'BMI': 'mean'}).
data_diabetes_cp = data_diabetes.replace(0, round(mean[0], 0), subset='BloodPressure')
data_diabetes_cp = data_diabetes_cp.replace(0, round(mean[1], 0), subset= 'SkinThickness')
data_diabetes_cp = data_diabetes_cp.replace(0, round(mean[2], 0), subset='Insulin')
data_diabetes_cp = data_diabetes_cp.replace(0, round(mean[3], 0), subset='BMI')
```

```
mean[0], mean[1], mean[2], mean[3]
```

```
(69.10546875, 20.536458333333332, 79.79947916666667, 31.992578124999977)
```

- Bảng sau khi điền các dữ liệu thiếu.

```
+-----------+-------+-------------+-------------+-------+----+------------------------+---+-------+
|Pregnancies|Glucose|BloodPressure|SkinThickness|Insulin| BMI|DiabetesPedigreeFunction|Age|Outcome|
+-----------+-------+-------------+-------------+-------+----+------------------------+---+-------+
|          6|    148|           72|           35|     80|33.6|                   0.627| 50|      1|
|          1|     85|           66|           29|     80|26.6|                   0.351| 31|      0|
|          8|    183|           64|           21|     80|23.3|                   0.672| 32|      1|
|          1|     89|           66|           23|     94|28.1|                   0.167| 21|      0|
|          0|    137|           40|           35|    168|43.1|                   2.288| 33|      1|
|          5|    116|           74|           21|     80|25.6|                   0.201| 30|      0|
|          3|     78|           50|           32|     88|31.0|                   0.248| 26|      1|
|         10|    115|           69|           21|     80|35.3|                   0.134| 29|      0|
|          2|    197|           70|           45|    543|30.5|                   0.158| 53|      1|
|          8|    125|           96|           21|     80|32.0|                   0.232| 54|      1|
```

- Tạo vector features assemble các thuộc tính. Tạo classifier của decision tree.

```
assembler = VectorAssembler(inputCols=['Pregnancies', 'Glucose', 'BloodPressure',\
                                       'SkinThickness','Insulin', 'BMI',\
                                       'DiabetesPedigreeFunction', 'Age'],outputCol='features')
```

```
from pyspark.ml.classification import DecisionTreeClassifier, RandomForestClassifier
dt_clf = RandomForestClassifier(featuresCol='features', labelCol='Outcome', numTrees=500)
```

- Tạo pipeline để xây dựng model. Chia tập dữ liệu thanhf 2 phần train, test.

```
pipeline = Pipeline(stages=[assembler, rf_clf])
```

```
train_data, test_data = data_diabetes_cp.randomSplit([0.8, 0.2], 10)
```

- Train model với tập train và output test trên tập test.

```
model = pipeline.fit(train_data)
```

```
test = model.transform(test_data)
```

- Kết quả: độ chính xác gần 79.00%

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
acc_evaluator = MulticlassClassificationEvaluator(labelCol="Outcome", predictionCol="prediction", metricName="
dtc_acc = acc_evaluator.evaluate(test)
print('A Decision Tree algorithm had an accuracy of: {0:2.2f}%'.format(dtc_acc*100))
```

```
A Decision Tree algorithm had an accuracy of: 77.98%
```

- Show 20 dòng trong dataFrame test

```
+----------+-------+
|prediction|Outcome|
+----------+-------+
|       0.0|      0|
|       0.0|      0|
|       0.0|      0|
|       1.0|      1|
|       1.0|      1|
|       0.0|      0|
|       1.0|      1|
|       0.0|      0|
|       0.0|      0|
|       0.0|      0|
|       0.0|      0|
|       1.0|      1|
|       1.0|      1|
|       0.0|      0|
|       1.0|      1|
|       0.0|      0|
|       0.0|      0|
|       0.0|      0|
|       0.0|      1|
|       0.0|      1|
+----------+-------+
only showing top 20 rows
```

d. breast-cancer-wisconsin

- Đọc dataset và in ra schema dữ liệu

```
data_bcw = spark.read.csv('breast-cancer-wisconsin.csv', inferSchema=True,header=True)
```

```
data_bcw.printSchema()
```

```
root
 |-- id: integer (nullable = true)
 |-- clump_thickness: integer (nullable = true)
 |-- size_uniformity: integer (nullable = true)
 |-- shape_uniformity: integer (nullable = true)
 |-- marginal_adhesion: integer (nullable = true)
 |-- epithelial_size: integer (nullable = true)
 |-- bare_nucleoli: string (nullable = true)
 |-- bland_chromatin: integer (nullable = true)
 |-- normal_nucleoli: integer (nullable = true)
 |-- mitoses: integer (nullable = true)
 |-- class: integer (nullable = true)
```

- In ra dữ liệu loại bỏ cột ‘id’ vì không cần thiết. label là ‘class’, ứng với 2:

11. Class: (2 for benign, 4 for malignant)

```
data_bcw.drop('id').show(10)
```

```
+--------------+--------------+---------------+----------------+--------------+-------------+---------------+---------------+-------+-----+
|clump_thickness|size_uniformity|shape_uniformity|marginal_adhesion|epithelial_size|bare_nucleoli|bland_chromatin|normal_nucleoli|mitoses|class|
+--------------+--------------+---------------+----------------+--------------+-------------+---------------+---------------+-------+-----+
|             5|             1|              1|               1|             2|            1|              3|              1|      1|    2|
|             5|             4|              4|               5|             7|           10|              3|              2|      1|    2|
|             3|             1|              1|               1|             2|            2|              3|              1|      1|    2|
|             6|             8|              8|               1|             3|            4|              3|              7|      1|    2|
|             4|             1|              1|               3|             2|            1|              3|              1|      1|    2|
|             8|            10|             10|               8|             7|           10|              9|              7|      1|    4|
|             1|             1|              1|               1|             2|           10|              3|              1|      1|    2|
|             2|             1|              2|               1|             2|            1|              3|              1|      1|    2|
|             2|             1|              1|               1|             2|            1|              1|              1|      5|    2|
|             4|             2|              1|               1|             2|            1|              2|              1|      1|    2|
+--------------+--------------+---------------+----------------+--------------+-------------+---------------+---------------+-------+-----+
only showing top 10 rows
```

- Tiền xử lý trên 2 cột bare_nucleoli (tạo index sang cột index_bare_nucleoli) và class (tạo cột ‘label’).

```
data_indexed = StringIndexer(inputCol='bare_nucleoli', outputCol='indexed_bare_nucleoli').fit(data_bcw_drop).transform(
```

```
data_indexed = StringIndexer(inputCol='class', outputCol='label').fit(data_indexed).transform(data_indexed)
```

```
data_indexed[['class', 'label']].show()
```

```
+-----+-----+
|class|label|
+-----+-----+
|    2|  0.0|
|    2|  0.0|
|    2|  0.0|
|    2|  0.0|
|    2|  0.0|
|    4|  1.0|
|    2|  0.0|
|    2|  0.0|
|    2|  0.0|
|    2|  0.0|
|    2|  0.0|
|    2|  0.0|
|    4|  1.0|
```

- Khởi tạo pipeline để train model.

```
train_data, test_data = data_indexed.randomSplit([0.8, 0.2], 10)
```

```
assembler = VectorAssembler(inputCols=['clump_thickness', 'size_uniformity', 'shape_uniformity',\
                                       'marginal_adhesion','epithelial_size', 'indexed_bare_nucleoli',\
                                       'bland_chromatin', 'normal_nucleoli', 'mitoses'],outputCol='features')
```

```
from pyspark.ml.classification import DecisionTreeClassifier, NaiveBayes
#dt = DecisionTreeClassifier(labelCol="label", featuresCol="features")
#nb = NaiveBayes(labelCol="label", featuresCol="features")
rf = RandomForestClassifier(numTrees=20)
```

```
pipeline = Pipeline(stages=[assembler, rf])
```

- Train và output test.

```
# train the model
model = pipeline.fit(train_data)
```

```
test = model.transform(test_data)
```

- Kết quả với thuật toán Random forest

```
rf_acc = evaluator.evaluate(test)
print('Algorithm had an accuracy of: {0:2.2f}%'.format(rf_acc*100))
```

```
Algorithm had an accuracy of: 97.89%
```

- Kết quả với thuâtj toán Decision tree

```
dt_acc = evaluator.evaluate(test)
print('A Decision Tree algorithm had an accuracy of: {0:2.2f}%'.format(dt_acc*100))
```

```
A Decision Tree algorithm had an accuracy of: 94.37%
```

- Kết quả với thuật toán Naïve Bayes

```
nb_acc = evaluator.evaluate(test)
print('A Decision Tree algorithm had an accuracy of: {0:2.2f}%'.format(nb_acc*100))
```

```
A Decision Tree algorithm had an accuracy of: 89.44%
```

- Chọn Random forest: kết quả cho độ chính xác 97.89%

Tham khảo

- Link hướng dẫn: https://github.com/Ruthvicp/CS5590_BigDataProgramming/wiki/Lab-Assignment-4----Spark-MLlib-classification-algorithms,-word-count-on-twitter-streaming
- Thư viện spark ml: https://spark.apache.org/docs/latest/api/python/pyspark.ml.html
- Decisiontree trên tập dữ liệu Heathcare stroke: https://towardsdatascience.com/healthcare-dataset-with-spark-6bf48019892b