Lab 11 - CKCS 113 Intro to Machine Learning

Solution Lab 11

Following references are used for this module

- Dataset: https://archive.ics.uci.edu/ml/datasets/adult
- https://docs.databricks.com/applications/machine-learning/mllib/binary-classification-mllib-pipelines.html

Installing spark library and setting the Java envoirement

```
!pip install pyspark
!pip install -U -q PyDrive
!apt install openjdk-8-jdk-headless -qq
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.6/dist-packages (2.4.5)
Requirement already satisfied: py4j==0.10.7 in /usr/local/lib/python3.6/dist-packages (from pyspark) (0.10.7)
openjdk-8-jdk-headless is already the newest version (8u242-b08-0ubuntu3~18.04).
0 upgraded, 0 newly installed, 0 to remove and 25 not upgraded.
```

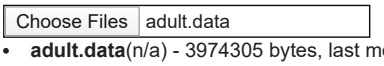
Import the necessary libraries and setting the spark session

```
from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import IndexToString, StringIndexer, VectorIndexer
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("LogisticRegressionExample").getOrCreate()
```

Upload the BostonHousing.csv file

```
from google.colab import files
uploaded = files.upload()
```



- **adult.data**(n/a) - 3974305 bytes, last modified: 3/31/2020 - 100% done

```
Saving adult.data to adult (1).data
```

Read the adult.data file into a dataset and name the columns

```
dataset = spark.read.format("csv").option("header","false").load("adult.data").toDF('age',
  'workclass',  'fnlwgt',  'education',  'education_num',  'marital_status',  'occupation',
  'relationship',  'race',  'sex',  'capital_gain',  'capital_loss',  'hours_per_week',  'native_country',  'income')
```

Print the schema of the dataset using describe and printschema functions

```
dataset.printSchema()
```

```
root
 |-- age: string (nullable = true)
 |-- workclass: string (nullable = true)
 |-- fnlwgt: string (nullable = true)
 |-- education: string (nullable = true)
 |-- education_num: string (nullable = true)
 |-- marital_status: string (nullable = true)
 |-- occupation: string (nullable = true)
 |-- relationship: string (nullable = true)
 |-- race: string (nullable = true)
 |-- sex: string (nullable = true)
 |-- capital_gain: string (nullable = true)
 |-- capital_loss: string (nullable = true)
 |-- hours_per_week: string (nullable = true)
 |-- native_country: string (nullable = true)
 |-- income: string (nullable = true)
```

Print first 5 records of the dataset

```
dataset.head(5)
```

```
[Row(age='39', workclass=' State-gov', fnlwgt=' 77516', education=' Bachelors', education_num=' 13', marital_status=' Never-ma
 Row(age='50', workclass=' Self-emp-not-inc', fnlwgt=' 83311', education=' Bachelors', education_num=' 13', marital_status=' M
 Row(age='38', workclass=' Private', fnlwgt=' 215646', education=' HS-grad', education_num=' 9', marital_status=' Divorced', o
 Row(age='53', workclass=' Private', fnlwgt=' 234721', education=' 11th', education_num=' 7', marital_status=' Married-civ-spo
 Row(age='28', workclass=' Private', fnlwgt=' 338409', education=' Bachelors', education_num=' 13', marital_status=' Married-o
```

```
dataset.describe()
```

```
DataFrame[summary: string, age: string, workclass: string, fnlwgt: string, education: string, education_num: string, marital_s
```

Changing Data Types

```
dataset = dataset.withColumn("age", dataset["age"].cast("Float"))
dataset = dataset.withColumn("fnlwgt", dataset["fnlwgt"].cast("Float"))
dataset = dataset.withColumn("education_num", dataset["education_num"].cast("Float"))
dataset = dataset.withColumn("capital_gain", dataset["capital_gain"].cast("Float"))
dataset = dataset.withColumn("capital_loss", dataset["capital_loss"].cast("Float"))
dataset = dataset.withColumn("hours_per_week", dataset["hours_per_week"].cast("Float"))
dataset.printSchema()
```

```
root
 |-- age: float (nullable = true)
 |-- workclass: string (nullable = true)
 |-- fnlwgt: float (nullable = true)
 |-- education: string (nullable = true)
 |-- education_num: float (nullable = true)
 |-- marital_status: string (nullable = true)
 |-- occupation: string (nullable = true)
 |-- relationship: string (nullable = true)
 |-- race: string (nullable = true)
 |-- sex: string (nullable = true)
 |-- capital_gain: float (nullable = true)
 |-- capital_loss: float (nullable = true)
 |-- hours_per_week: float (nullable = true)
 |-- native_country: string (nullable = true)
 |-- income: string (nullable = true)
```

```
dataset.head(5)
```

```
[Row(age=39.0, workclass=' State-gov', fnlwgt=77516.0, education=' Bachelors', education_num=13.0, marital_status=' Never-marr
 Row(age=50.0, workclass=' Self-emp-not-inc', fnlwgt=83311.0, education=' Bachelors', education_num=13.0, marital_status=' Mar
 Row(age=38.0, workclass=' Private', fnlwgt=215646.0, education=' HS-grad', education_num=9.0, marital_status=' Divorced', occ
 Row(age=53.0, workclass=' Private', fnlwgt=234721.0, education=' 11th', education_num=7.0, marital_status=' Married-civ-spous
 Row(age=28.0, workclass=' Private', fnlwgt=338409.0, education=' Bachelors', education_num=13.0, marital_status=' Married-civ
```

```
dataset1 = dataset[['age', 'fnlwgt', 'education_num', 'capital_gain', 'capital_loss', 'hours_per_week','income']]
```

```
dataset1.head(5)
```

```
[Row(age=39.0, fnlwgt=77516.0, education_num=13.0, capital_gain=2174.0, capital_loss=0.0, hours_per_week=40.0, income=' <=50K
 Row(age=50.0, fnlwgt=83311.0, education_num=13.0, capital_gain=0.0, capital_loss=0.0, hours_per_week=13.0, income=' <=50K'),
 Row(age=38.0, fnlwgt=215646.0, education_num=9.0, capital_gain=0.0, capital_loss=0.0, hours_per_week=40.0, income=' <=50K'),
 Row(age=53.0, fnlwgt=234721.0, education_num=7.0, capital_gain=0.0, capital_loss=0.0, hours_per_week=40.0, income=' <=50K'),
 Row(age=28.0, fnlwgt=338409.0, education_num=13.0, capital_gain=0.0, capital_loss=0.0, hours_per_week=40.0, income=' <=50K')]
```

```
from pyspark.ml.feature import StringIndexer
indexer = StringIndexer(inputCol="income", outputCol="label")
dataset1 = indexer.fit(dataset1).transform(dataset1)
```

```
dataset1.head(5)
```

```
[Row(age=39.0, fnlwgt=77516.0, education_num=13.0, capital_gain=2174.0, capital_loss=0.0, hours_per_week=40.0, income=' <=50K
 Row(age=50.0, fnlwgt=83311.0, education_num=13.0, capital_gain=0.0, capital_loss=0.0, hours_per_week=13.0, income=' <=50K', l
 Row(age=38.0, fnlwgt=215646.0, education_num=9.0, capital_gain=0.0, capital_loss=0.0, hours_per_week=40.0, income=' <=50K', l
 Row(age=53.0, fnlwgt=234721.0, education_num=7.0, capital_gain=0.0, capital_loss=0.0, hours_per_week=40.0, income=' <=50K', l
 Row(age=28.0, fnlwgt=338409.0, education_num=13.0, capital_gain=0.0, capital_loss=0.0, hours_per_week=40.0, income=' <=50K',
```

Make the predictions by considering the numeric column only

```
from pyspark.ml.feature import VectorAssembler
```

```
assembler = VectorAssembler(inputCols=['age', 'fnlwgt', 'education_num', 'capital_gain', 'capital_loss', 'hours_per_week'], output(
```

```
output = assembler.transform(dataset1)
```

```
finalized_data = output.select('featuresVector','label')
finalized_data.show()
```

```
+--------------------+-----+
|      featuresVector|label|
+--------------------+-----+
|[39.0,77516.0,13....|  0.0|
|[50.0,83311.0,13....|  0.0|
|[38.0,215646.0,9....|  0.0|
|[53.0,234721.0,7....|  0.0|
|[28.0,338409.0,13...|  0.0|
|[37.0,284582.0,14...|  0.0|
|[49.0,160187.0,5....|  0.0|
|[52.0,209642.0,9....|  1.0|
|[31.0,45781.0,14....|  1.0|
|[42.0,159449.0,13...|  1.0|
|[37.0,280464.0,10...|  1.0|
|[30.0,141297.0,13...|  1.0|
|[23.0,122272.0,13...|  0.0|
|[32.0,205019.0,12...|  0.0|
|[40.0,121772.0,11...|  1.0|
|[34.0,245487.0,4....|  0.0|
|[25.0,176756.0,9....|  0.0|
|[32.0,186824.0,9....|  0.0|
|[38.0,28887.0,7.0...|  0.0|
|[43.0,292175.0,14...|  1.0|
+--------------------+-----+
only showing top 20 rows
```

```
lr = LogisticRegression(labelCol="label", featuresCol="featuresVector", maxIter=10)
```

```
train_data,test_data = finalized_data.randomSplit([0.70,0.30])
```

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

```
result = model.evaluate(test_data)
```

```
result.accuracy
```

```
result.accuracy
```

```
0.7979922147101004
```

```
predictions = model.transform(test_data.select('featuresVector'))
predictions.show()
```

```
+--------------------+--------------------+--------------------+----------+
|      featuresVector|       rawPrediction|         probability|prediction|
+--------------------+--------------------+--------------------+----------+
|[17.0,19752.0,7.0...|[3.02213570909736...|[0.95356418533907...|       0.0|
|[17.0,24090.0,9.0...|[2.38959747619218...|[0.91603061174725...|       0.0|
|[17.0,25051.0,6.0...|[3.43379594515935...|[0.96874420940362...|       0.0|
|[17.0,28031.0,5.0...|[3.63845403939451...|[0.97438064816155...|       0.0|
|[17.0,32763.0,6.0...|[3.46072194535894...|[0.96954928829136...|       0.0|
|[17.0,34019.0,6.0...|[3.34706070729597...|[0.96600845295609...|       0.0|
|[17.0,36218.0,7.0...|[3.14513009003590...|[0.95871640436456...|       0.0|
|[17.0,39815.0,6.0...|[3.23579042220981...|[0.96215914110057...|       0.0|
|[17.0,41979.0,6.0...|[2.89396197485463...|[0.94754714738736...|       0.0|
|[17.0,47199.0,7.0...|[3.05945498176334...|[0.95518897432250...|       0.0|
|[17.0,47407.0,7.0...|[3.37966808306390...|[0.96706303458203...|       0.0|
|[17.0,47425.0,7.0...|[3.26535486266936...|[0.96322096514559...|       0.0|
|[17.0,47771.0,7.0...|[3.15121438072778...|[0.95895654512499...|       0.0|
|[17.0,52486.0,7.0...|[3.33661381589899...|[0.96566374252561...|       0.0|
|[17.0,52967.0,6.0...|[3.67714307264619...|[0.97532892029221...|       0.0|
|[17.0,54257.0,7.0...|[3.15463017830655...|[0.95909077655959...|       0.0|
|[17.0,57324.0,6.0...|[3.13068869047550...|[0.95814102283219...|       0.0|
|[17.0,63734.0,6.0...|[3.36270986317834...|[0.96651857996160...|       0.0|
|[17.0,67808.0,6.0...|[2.90756460138284...|[0.94821911782751...|       0.0|
|[17.0,75333.0,6.0...|[3.27736021942994...|[0.96364391381173...|       0.0|
+--------------------+--------------------+--------------------+----------+
only showing top 20 rows
```