EMORY UNIVERSITY

Machine Learning II

Homework #2 **Due: turned in by Monday 02/10/2020 before class**

Adil Ashish Kumar

(put your name above)

Total grade: ________ out of ___100___ points

Please answer the following questions and submit your assignment as a single PDF file by uploading it to the HW2 drop-box on the course website.

If you use others' work as part of your answer, please properly cite your source.

Part I. Hands on (100 points)

1. Analyze the Titanic Dataset using pyspark.ml (100 points)

This is a popular dataset for machine learning. A description of the dataset can be found at https://www.kaggle.com/c/titanic/data (our dataset corresponds to the train.csv file from Kaggle). You should train a logistic regression model using this dataset and the pyspark.ml package. The goal is to predict for each passenger whether he/she survive the Titanic tragedy as well as to use the pipeline and feature functionality of pyspark.ml.

Step 1 (8 points): The goal of this step is to read the data from the local folder. You should infer the schema (e.g., columns) from the csv file. Tip: explore the spark_csv package from databricks.

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('ml2hw2).getOrCreate()
       from pyspark.sql import Row
data = spark.read.csv('s3://ml2hw2/train.csv',header=True,inferSchema=True)
       data.show()
```

Step 2 (15 points): The goal of this step is to familiarize yourself with the dataset. This step is useful in detecting data problems, informing the data engineering steps, and informing the feature selection processes. You should:

i) Print the dataset and verify that the schema contains all the variables.
ii) Print the first 10 rows from the dataset.
iii) Obtain summary statistics for all variables in the dataframe. Pay attention to whether there are missing data as well as whether the field appears to be continuous or discrete.
iv) For each of the string columns (except name and ticket), print the count of the 10 most frequent values ordered by descending order of frequency.
v) Based on the above, which columns would you keep as features and which would you drop? Justify your answer.

```
       data.printSchema()
       data.show(10)
       data.describe().show()
data.createOrReplaceTempView('data')
freqct_sex = spark.sql("SELECT Sex, count(*) FROM data group by Sex order by count(*) desc limit 10 ")
freqct_sex.show()
freqct_cabin = spark.sql("SELECT Cabin, count(*) FROM data group by Cabin order by count(*) desc limit 10 ")
freqct_cabin.show()
freqct_em = spark.sql("SELECT Embarked, count(*) FROM data group by Embarked order by count(*) desc limit 10 ")
       freqct_em.show()
```

I would keep Sex, Embarked, SibSp, Pclass, Parch, age and Fare since there is sufficient data while Cabin column has too many null values

Step 3 (12 points): The goal of this step is to engineer the necessary features for the machine learning model. You should:

i) Select all feature columns you plan to use in addition to the target variable (i.e., 'Survived') and covert all numerical columns into double data type. Tip: you can use the .cast() from pyspark.sql.functions.

ii) Replace the missing values in the Age column with the mean value. Create also a new variable (e.g., 'AgeNA') indicating whether the value of age was missing or not.

iii) Print the revised dataframe and recalculate the summary statistics.

```
from pyspark.sql.types import DoubleType
dblCol=['Pclass','Sibsp','Parch']
data = data.withColumn('Pclass', data['Pclass'].cast(DoubleType()))
data = data.withColumn('Sibsp', data['Sibsp'].cast(DoubleType()))
data = data.withColumn('Parch', data['Parch'].cast(DoubleType()))
data = data.select('Sex','Embarked','Pclass','SibSp','Parch','Age','Fare','Survived')
           data.show()

mean_age = data.agg({"Age": "avg"})
mean_age.show()
data = data.fillna({'Age':29.7})
       data.show()

from pyspark.sql.functions import udf
from pyspark.sql.types import *
def vtc(value):
if value == 29.7: return True
else: return False
udfvtc = udf(vtc, StringType())
       data_with_cat = data.withColumn("AgeNA", udfvtc("Age"))

data_with_cat.show()
       data_with_cat.describe().show()
```

Step 4 (9 points): The goal of this step is to encode all string and categorical variables in order to use them in the pipeline afterwards. You should:

i) Import all necessary pyspark functions.

ii) Create indexers and encoders for categorical string variables. Call them [field]_indexer and [field]_encoder, respectively. For instance, gender_indexer and gender_encoder.

```
from pyspark.mllib.linalg import Vectors
from pyspark.ml import Pipeline
from pyspark.sql import Row
from pyspark.ml.feature import StringIndexer,VectorAssembler
from pyspark.ml.feature import OneHotEncoder
       from pyspark.ml.classification import LogisticRegression

catcols = ["Sex", "Embarked", "AgeNA"]
stages = [] # stages in Pipeline
for catcol in catcols:
# Indexing Category with StringIndexer
stringIndexer = StringIndexer(inputCol=catcol, outputCol=catcol+"Index")
# Using OneHotEncoder to convert categorical variables into binary SparseVectors
encoder = OneHotEncoder(inputCol=catcol+"Index", outputCol=catcol+"encoder")
# Add stages. These are not run here, but will run all at once later on.
       stages += [stringIndexer, encoder]

labelstringIdx = StringIndexer(inputCol = "Survived", outputCol = "label")
       stages += [labelstringIdx]
```

Step 5 (7 points): The goal of this step is to assemble all feature columns into a feature vector in order to be used in the pipeline. Tip: you can use the VectorAssembler to do this.

```
numCols = ["Pclass", "SibSp", "Parch", "Age", "Fare"]
assemblerInputs = map(lambda c: c + "encoder", catcols) + numCols
assembler = VectorAssembler(inputCols=assemblerInputs, outputCol="features")
        stages += [assembler]
```

Step 6 (7 points): The goal of this step is to create the logistic regression model to be used in the pipeline.

```
lrmodel = LogisticRegression(labelCol="label", featuresCol="features", maxIter=10)
        stages += [lrmodel]
```

Step 7 (6 points): The goal of this step is to assemble the pipeline.

```
ppl = Pipeline(stages=stages)
allData=ppl.fit(data_with_cat).transform(data_with_cat)
        allData.cache()
```

Step 8 (12 points): The goal of this step is to prepare the training and test datasets. You should:

i) Use a 70-30 random split for the training and test sets, respectively.
ii) Verify the size of each dataset after the split.

```
trainData, testData = allData.randomSplit([0.7, 0.3], seed = 100)
trainData.show()
testData.describe().show()
```

Step 9 (12 points): The goal of this step is to fit the model and then use it on the test set to generate predictions. You should:

i) Fit the model using the predefined pipeline on the training set.
ii) Use the fitted model for prediction on the test set.
iii) Report the logistic regression coefficients.
iv) Interpret the obtained coefficients.

```
lrfit = lrmodel.fit(trainData)

transformed = lrfit.transform(testData)
print("Coefficients are : " + str(lrfit.coefficients))
print("Intercept is: " + str(lrfit.intercept))
```

Step 10 (12 points): The goal of this step is to evaluate the model performance. You should:

i) Print the first 5 rows of the results.
ii) Report the AUC for this model.

```
from pyspark.mllib.evaluation import BinaryClassificationMetrics as metric
result = transformed.select(['probability','label'])

results.show(5)

result_collect = results.collect()
result_list = [(float(i[0][0]),1.0-float(i[1])) for i in result_collect]
scoreAndLabels = sc.paralleize(result_list)
metrics = metric(scoreAndLabels)
print("ROC score is ",metrics.areaUnderROC)
```