EMORY UNIVERSITY

Machine Learning II

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

_____Qiyu Wang_____

(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.

```
[4] from pyspark.sql import SQLContext
    sqlContext = SQLContext(sc)

    df = sqlContext.read.format(source="com.databricks.spark.csv").options(header = 'true', inferSchema = 'true').load(path = '/content/drive/My Drive/train.csv')
```

```
[5] df.printSchema()
```

```
root
 |-- PassengerId: integer (nullable = true)
 |-- Survived: integer (nullable = true)
 |-- Pclass: integer (nullable = true)
 |-- Name: string (nullable = true)
 |-- Sex: string (nullable = true)
 |-- Age: double (nullable = true)
 |-- SibSp: integer (nullable = true)
 |-- Parch: integer (nullable = true)
 |-- Ticket: string (nullable = true)
 |-- Fare: double (nullable = true)
 |-- Cabin: string (nullable = true)
 |-- Embarked: string (nullable = true)
```

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.

Yes, all the schema matches as shown above and below, in terms of both data types and labels.

ii) Print the first 10 rows from the dataset.

```
df.show(10)
```

```
+-----------+--------+------+--------------------+------+----+-----+-----+----------------+-------+-----+--------+
|PassengerId|Survived|Pclass|                Name|   Sex| Age|SibSp|Parch|          Ticket|   Fare|Cabin|Embarked|
+-----------+--------+------+--------------------+------+----+-----+-----+----------------+-------+-----+--------+
|          1|       0|     3|Braund, Mr. Owen ...|  male|22.0|    1|    0|       A/5 21171|   7.25| null|       S|
|          2|       1|     1|Cumings, Mrs. Joh...|female|38.0|    1|    0|        PC 17599|71.2833|  C85|       C|
|          3|       1|     3|Heikkinen, Miss. ...|female|26.0|    0|    0|STON/O2. 3101282|  7.925| null|       S|
|          4|       1|     1|Futrelle, Mrs. Ja...|female|35.0|    1|    0|          113803|   53.1| C123|       S|
|          5|       0|     3|Allen, Mr. Willia...|  male|35.0|    0|    0|          373450|   8.05| null|       S|
|          6|       0|     3|    Moran, Mr. James|  male|null|    0|    0|          330877| 8.4583| null|       Q|
|          7|       0|     1|McCarthy, Mr. Tim...|  male|54.0|    0|    0|           17463|51.8625|  E46|       S|
|          8|       0|     3|Palsson, Master. ...|  male| 2.0|    3|    1|          349909| 21.075| null|       S|
|          9|       1|     3|Johnson, Mrs. Osc...|female|27.0|    0|    2|          347742|11.1333| null|       S|
|         10|       1|     2|Nasser, Mrs. Nich...|female|14.0|    1|    0|          237736|30.0708| null|       C|
+-----------+--------+------+--------------------+------+----+-----+-----+----------------+-------+-----+--------+
only showing top 10 rows
```

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.

```
+-------+-----------------+-------------------+------------------+------------------+------+------------------+
|summary|      PassengerId|           Survived|            Pclass|              Name|   Sex|               Age|
+-------+-----------------+-------------------+------------------+------------------+------+------------------+
|  count|              891|                891|               891|               891|   891|               714|
|   mean|            446.0| 0.3838383838383838| 2.308641975308642|              null|  null| 29.69911764705882|
| stddev|257.3538420152301|0.48659245426485753|0.8360712409770491|              null|  null|14.526497332334035|
|    min|                1|                  0|                 1|"Andersson, Mr. A...|female|              0.42|
|    max|              891|                  1|                 3|van Melkebeke, Mr...|  male|              80.0|
+-------+-----------------+-------------------+------------------+------------------+------+------------------+

+------------------+-------------------+------------------+-----------------+-----+--------+
|             SibSp|              Parch|            Ticket|             Fare|Cabin|Embarked|
+------------------+-------------------+------------------+-----------------+-----+--------+
|               891|                891|               891|              891|  204|     889|
|0.5230078563411896|0.38159371492704824|260318.54916792738| 32.2042079685746| null|    null|
|1.1027434322934315| 0.8060572211299488|471609.26868834975|49.69342859718089| null|    null|
|                 0|                  0|            110152|              0.0|  A10|       C|
|                 8|                  6|         WE/P 5735|         512.3292|    T|       S|
+------------------+-------------------+------------------+-----------------+-----+--------+
```

From the summary statistics table above, we notice that three columns have missing values, where Age has 177 missing values, Cabin has 687 missing values, and Embarked has 2 missing values.

Based on the top 10 rows and schema, we notice that ‘Pclass’, 'Survived', 'Age', 'SibSp', 'Parch' are discrete variables, and ‘Fare’ is continuous. The rest are categorical variables.

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.

The PassengerId and Name presented as unique identifier in this dataset, where each instance has only one occurrence.

```
df.groupBy('Sex').count().orderBy("count", ascending = False).show(10)
```

```
+------+-----+
|   Sex|count|
+------+-----+
|  male|  577|
|female|  314|
+------+-----+
```

For Sex, male has more count of 577, and female has 314.

```
df.groupBy('Ticket').count().orderBy("count", ascending = False).show(10)
```

```
+------------+-----+
|      Ticket|count|
+------------+-----+
|      347082|    7|
|        1601|    7|
|    CA. 2343|    7|
|     3101295|    6|
|      347088|    6|
|     CA 2144|    6|
|S.O.C. 14879|    5|
|      382652|    5|
|        4133|    4|
|  W./C. 6608|    4|
+------------+-----+
only showing top 10 rows
```

```
[47] df.groupBy('Ticket').count().groupBy('count').count().orderBy("count", ascending = False).show(10)
```

```
+-----+-----+
|count|count|
+-----+-----+
|    7|    3|
|    6|    3|
|    5|    2|
|    4|   11|
|    3|   21|
|    2|   94|
|    1|  547|
+-----+-----+
```

For tickets, there seems are tickets bought together, but the chances are low.

```
df.groupBy('Cabin').count().orderBy("count", ascending = False).show(10)
```

```
+-----------+-----+
|      Cabin|count|
+-----------+-----+
|       null|  687|
|         G6|    4|
|C23 C25 C27|    4|
|    B96 B98|    4|
|       E101|    3|
|    C22 C26|    3|
|          D|    3|
|         F2|    3|
|        F33|    3|
|        E44|    2|
+-----------+-----+
only showing top 10 rows
```

```
[48] df.groupBy('Cabin').count().groupBy('count').count().orderBy("count", ascending = False).show(10)
```

```
+-----+-----+
|count|count|
+-----+-----+
|  687|    1|
|    4|    3|
|    3|    5|
|    2|   38|
|    1|  101|
+-----+-----+
```

For Cabin, most information is missing, and the chances see people in the same cabin is low.

```
df.groupBy('Embarked').count().orderBy("count", ascending = False).show(10)
```

```
+--------+-----+
|Embarked|count|
+--------+-----+
|       S|  644|
|       C|  168|
|       Q|   77|
|    null|    2|
+--------+-----+
```

For Embarked, most frequent class is S with 644 times, followed by C with 168 and Q with 77, and it contains two missing value.

v) Based on the above, which columns would you keep as features and which would you drop? Justify your answer.

The PassengerId and Name have absolutely no value, since they uniquely identify each instance. Ticket is not so helpful too, since most people bought it individually. Also it provide similar information as Parch, since the people who bought tickets together are likely to be family. Cabin has too many missing values and labels are not clean, it might not be helpful for prediction. So we should drop it.

```
[65] df = df.drop("PassengerId","Name","Ticket","Cabin")
```

The rest will be kept,

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.

```
[66] df = df.withColumn("SibSp",df["SibSp"].cast("double")).withColumn("Pclass",df["Pclass"].cast("double"))\
     .withColumn("Parch",df["Parch"].cast("double")).withColumn("Survived",df["Survived"].cast("double"))
```

```
[67] df.printSchema()
```

```
root
 |-- Survived: double (nullable = true)
 |-- Pclass: double (nullable = true)
 |-- Sex: string (nullable = true)
 |-- Age: double (nullable = true)
 |-- SibSp: double (nullable = true)
 |-- Parch: double (nullable = true)
 |-- Fare: double (nullable = true)
 |-- Embarked: string (nullable = true)
```

We transform the columns, "Pclass", "Slibsp" and "Survided" to float point numbers.

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.

```
df = df.withColumn("Age_M",df.Age.isNull())
df.show(10)
```

```
+--------+------+------+----+-----+-----+-------+--------+-----+
|Survived|Pclass|   Sex| Age|SibSp|Parch|   Fare|Embarked|Age_M|
+--------+------+------+----+-----+-----+-------+--------+-----+
|     0.0|   3.0|  male|22.0|  1.0|  0.0|   7.25|       S|false|
|     1.0|   1.0|female|38.0|  1.0|  0.0|71.2833|       C|false|
|     1.0|   3.0|female|26.0|  0.0|  0.0|  7.925|       S|false|
|     1.0|   1.0|female|35.0|  1.0|  0.0|   53.1|       S|false|
|     0.0|   3.0|  male|35.0|  0.0|  0.0|   8.05|       S|false|
|     0.0|   3.0|  male|null|  0.0|  0.0| 8.4583|       Q| true|
|     0.0|   1.0|  male|54.0|  0.0|  0.0|51.8625|       S|false|
|     0.0|   3.0|  male| 2.0|  3.0|  1.0| 21.075|       S|false|
|     1.0|   3.0|female|27.0|  0.0|  2.0|11.1333|       S|false|
|     1.0|   2.0|female|14.0|  1.0|  0.0|30.0708|       C|false|
+--------+------+------+----+-----+-----+-------+--------+-----+
only showing top 10 rows
```

I created a column named Age_M, indicating if an age value is missing.

```
mean_age=df.select(mean(df.Age)).collect()[0][0]
print(mean_age)
```

```
29.699117647058763
```

```
[97] df = df.na.fill(mean_age,subset=['Age'])
```

```
[98] df.show(10)
```

```
+--------+------+------+-----------------+-----+-----+-------+--------+-----+
|Survived|Pclass|   Sex|              Age|SibSp|Parch|   Fare|Embarked|Age_M|
+--------+------+------+-----------------+-----+-----+-------+--------+-----+
|     0.0|   3.0|  male|             22.0|  1.0|  0.0|   7.25|       S|false|
|     1.0|   1.0|female|             38.0|  1.0|  0.0|71.2833|       C|false|
|     1.0|   3.0|female|             26.0|  0.0|  0.0|  7.925|       S|false|
|     1.0|   1.0|female|             35.0|  1.0|  0.0|   53.1|       S|false|
|     0.0|   3.0|  male|             35.0|  0.0|  0.0|   8.05|       S|false|
|     0.0|   3.0|  male|29.69911764705882|  0.0|  0.0| 8.4583|       Q| true|
|     0.0|   1.0|  male|             54.0|  0.0|  0.0|51.8625|       S|false|
|     0.0|   3.0|  male|              2.0|  3.0|  1.0| 21.075|       S|false|
|     1.0|   3.0|female|             27.0|  0.0|  2.0|11.1333|       S|false|
|     1.0|   2.0|female|             14.0|  1.0|  0.0|30.0708|       C|false|
+--------+------+------+-----------------+-----+-----+-------+--------+-----+
only showing top 10 rows
```

```
[152] df = df.withColumn("Age_M",df["Age_M"].cast("double"))
```

Here I filled in missing values with the average of the age column, and convert into float point.

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

```
[109] df = df.na.drop()
```

Here I dropped NA's to remove the two instance where Embarks has 2 missing values.

```
df.show(10)
```

```
+--------+------+------+------------------+-----+-----+-------+--------+-----+
|Survived|Pclass|   Sex|               Age|SibSp|Parch|   Fare|Embarked|Age_M|
+--------+------+------+------------------+-----+-----+-------+--------+-----+
|     0.0|   3.0|  male|              22.0|  1.0|  0.0|   7.25|       S|false|
|     1.0|   1.0|female|              38.0|  1.0|  0.0|71.2833|       C|false|
|     1.0|   3.0|female|              26.0|  0.0|  0.0|  7.925|       S|false|
|     1.0|   1.0|female|              35.0|  1.0|  0.0|   53.1|       S|false|
|     0.0|   3.0|  male|              35.0|  0.0|  0.0|   8.05|       S|false|
|     0.0|   3.0|  male|29.69911764705882|  0.0|  0.0| 8.4583|       Q| true|
|     0.0|   1.0|  male|              54.0|  0.0|  0.0|51.8625|       S|false|
|     0.0|   3.0|  male|               2.0|  3.0|  1.0| 21.075|       S|false|
|     1.0|   3.0|female|              27.0|  0.0|  2.0|11.1333|       S|false|
|     1.0|   2.0|female|              14.0|  1.0|  0.0|30.0708|       C|false|
+--------+------+------+------------------+-----+-----+-------+--------+-----+
only showing top 10 rows
```

```
df.describe().show()
```

```
+-------+-------------------+-------------------+------+------+-----------------+-------------------+-------------------+-----------------+--------+-----+
|summary|           Survived|             Pclass|   Sex|   Age|            SibSp|              Parch|               Fare|Embarked|Age_M|
+-------+-------------------+-------------------+------+------+-----------------+-------------------+-------------------+-----------------+--------+-----+
|  count|                889|                889|   889|   889|              889|                889|                889|              889|     889|  889|
|   mean|0.38245219347581555|2.3115860517435323|  null|  null|29.653446370674192|0.5241844769403825|0.38245219347581555|32.09668087739029|    null| null|
| stddev|0.48625968831477334|0.8346997785705753|  null|  null|12.968366309252314| 1.103704875596923| 0.8067607445174785|49.69750431670795|    null| null|
|    min|                0.0|                1.0|female|  0.42|              0.0|                0.0|                0.0|              0.0|       C|    C|
|    max|                1.0|                3.0|  male|  80.0|              8.0|                6.0|           512.3292|              S|       S|    S|
+-------+-------------------+-------------------+------+------+-----------------+-------------------+-------------------+-----------------+--------+-----+
```

Here's the new top 10 rows of the dataframe and the recalculated statistics.

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.

```
[113] from pyspark.ml.linalg import Vectors
      from pyspark.ml import Pipeline
      from pyspark.sql import Row
      from pyspark.ml.feature import StringIndexer, VectorAssembler
      from pyspark.ml.classification import DecisionTreeClassifier, LogisticRegression
      from pyspark.mllib.feature import StandardScaler, OneHotEncoder
      from pyspark.mllib.regression import LinearRegressionWithSGD, LabeledPoint
      from pyspark.mllib.stat import Statistics
      from pyspark.sql.functions import *
      from pyspark.sql import *
```

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

```
[122] sex_indexer = StringIndexer(inputCol="Sex",outputCol="Sex_ix")
      embarked_indexer = StringIndexer(inputCol="Embarked",outputCol="Embarked_ix")
```

```
[123] sex_encoder = OneHotEncoder(inputCol="Sex_ix",outputCol="Sex_oh")
      embarked_encoder = OneHotEncoder(inputCol="Sex_ix",outputCol="Embarked_oh")
```

```
[364] ss = StandardScaler(withMean=True, withStd = True, inputCol = "features", outputCol = "scaled")
```

I defined two indexers and two encoders, and standardized the features here,

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.

```
[125] x_cols = ["Pclass", "Sex_oh", "Age", "SibSp", "Parch", "Fare", "Embarked_oh"]

      va = VectorAssembler(inputCols = x_cols, outputCol = 'features')
```

Set up a VectorAssembler with two one-hot variables and the Age_M column, totally to be 8 columns.

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

```
[129] lr = LogisticRegression(maxIter=10,featuresCol = "features", labelCol = "Survived")
```

Set up a logistic regression model with the features created.

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

```
[365] steps = [sex_indexer, embarked_indexer, sex_encoder, embarked_encoder, va, ss, lr]
```

```
[366] pl = Pipeline(stages=steps)
```

Set up a pipeline with the transformer and regressor.

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.

```
[161] from pyspark.ml.tuning import TrainValidationSplit

      train, test = df.randomSplit([0.7, 0.3], seed=666)
```

Here I split the original datasets into 70% training and 30% testing.

ii) Verify the size of each dataset after the split.

```
[166] train.count()

      631
```

```
[167] test.count()

      258
```

Train has 631 rows and test has 258 rows, which is appropriately a 70/30 split.

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.

```
[165] plmodel = pl.fit(train)
```

Here I fitted the model.

ii) Use the fitted model for prediction on the test set.

```
[171] predictions = plmodel.transform(test)
```

Here I made the predictions.

iii) Report the logistic regression coefficients.

```
[389] import pandas as pd

      pd.DataFrame([coefficients], columns = x_cols)
```

	Pclass	Sex_oh	Age	Age_M	SibSp	Parch	Fare	Embarked_oh	Embarked_oh2	intercept
0	-0.896848	-1.328635	-0.413685	-0.080899	-0.438067	-0.116797	0.026506	-0.233985	-0.101337	-0.623776

iv) Interpret the obtained coefficients.

Based on the coefficients, we see that Sex has the biggest impact of -1.33, followed by Pclass at -0.89. Then Age and Sibsp played significant roles too at -0.4. The rest have coefficients from -0.1 to -0.2, not as significant as the top three features. All features have negative effects on survival. For categorical, the better the Pclass the higher chance of survival.

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.

```
[390] predictions.show(5,truncate=False)

+--------+------+----+-----------------+-----+-----+------+--------+-----+------+-----------+-------------+-------------+--------------------------------------------------+---------
|Survived|Pclass|Sex |Age              |SibSp|Parch|Fare  |Embarked|Age_M|Sex_ix|Embarked_ix|Sex_oh       |Embarked_oh  |features                                          |scaled
+--------+------+----+-----------------+-----+-----+------+--------+-----+------+-----------+-------------+-------------+--------------------------------------------------+---------
|0.0     |1.0   |male|19.0             |3.0  |2.0  |263.0 |S       |0.0  |0.0   |0.0        |(1,[0],[1.0])|(2,[0],[1.0])|[1.0,1.0,19.0,0.0,3.0,2.0,263.0,1.0,0.0]          |[-1.5119
|0.0     |1.0   |male|29.69911764705882|0.0  |0.0  |0.0   |S       |1.0  |0.0   |0.0        |(1,[0],[1.0])|(2,[0],[1.0])|[1.0,1.0,29.69911764705882,1.0,0.0,0.0,0.0,1.0,0.0]   |[-1.5119
|0.0     |1.0   |male|29.69911764705882|0.0  |0.0  |25.925|S       |1.0  |0.0   |0.0        |(1,[0],[1.0])|(2,[0],[1.0])|[1.0,1.0,29.69911764705882,1.0,0.0,0.0,25.925,1.0,0.0]|[-1.5119
|0.0     |1.0   |male|29.69911764705882|0.0  |0.0  |26.55 |S       |1.0  |0.0   |0.0        |(1,[0],[1.0])|(2,[0],[1.0])|[1.0,1.0,29.69911764705882,1.0,0.0,0.0,26.55,1.0,0.0] |[-1.5119
|0.0     |1.0   |male|33.0             |0.0  |0.0  |5.0   |S       |0.0  |0.0   |0.0        |(1,[0],[1.0])|(2,[0],[1.0])|[1.0,1.0,33.0,0.0,0.0,0.0,5.0,1.0,0.0]            |[-1.5119
+--------+------+----+-----------------+-----+-----+------+--------+-----+------+-----------+-------------+-------------+--------------------------------------------------+---------
only showing top 5 rows
```

```
predictions.show(5,truncate=False)

------------------------------------------------------------------------------+-----------------------------------------+-----------------------------------------+----------+
                                                                              |rawPrediction                            |probability                              |prediction|
------------------------------------------------------------------------------+-----------------------------------------+-----------------------------------------+----------+
3747153,1.9698331045964685,4.798751861331281,0.6361625783893372,-0.5015822824361749]   |[1.014297309232067,-1.014297309232067]    |[0.7338602969914926,0.26613970300850737]|0.0       |
5433434,-0.4697443026662824,-0.6723090858318324,0.6361625783893372,-0.5015822824361749] |[0.2722829804527223,-0.2722829804527223]  |[0.5676532872600958,0.43234671273990416]|0.0       |
5433434,-0.4697443026662824,-0.13300393353067752,0.6361625783893372,-0.5015822824361749]|[0.2579883037003868,-0.2579883037003868]  |[0.5641417081507305,0.4358582918492695] |0.0       |
5433434,-0.4697443026662824,-0.12000236283874996,0.6361625783893372,-0.5015822824361749]|[0.2576436875781029,-0.2576436875781029]  |[0.5640569700533508,0.43594302994664924]|0.0       |
5433434,-0.4697443026662824,-0.568296520296412,0.6361625783893372,-0.5015822824361749]  |[0.17460539210439746,-0.17460539210439746]|[0.5435407850441925,0.45645921495580755]|0.0       |
------------------------------------------------------------------------------+-----------------------------------------+-----------------------------------------+----------+
```

ii) Report the AUC for this model.

```
[391] print("AUC outsample : %s" % plmodel.stages[-1].summary.areaUnderROC)

AUC outsample : 0.855969551282051
```

```
[392] from sklearn.metrics import roc_auc_score

cols = ['probability', 'Survived']
evalue = predictions.select(cols)

evalue = evalue.select("*").toPandas()
evalue['probability'] = evalue['probability'].apply(lambda x: pd.Series(x.toArray()))[1]

print("AUC outsample : %s" % roc_auc_score(evalue["Survived"], evalue["probability"]))

AUC outsample : 0.8546106223525578
```