Harshil Shah – SEC01 (NUID 002780887)

Big Data System Engineering with Scala
Spring 2023
Spark Assignment 2

- GitHub Repo URL - https://github.com/harshilshahneu/CSYE7200-Harshil-Shah/tree/Spring2023/assignment-spark-2

- List of Tasks Implemented

Exploratory Data Analysis:-

- Show basic statistics for each column in train.csv
- Calculate the count of each categorical value in the Pclass column
- Calculate the count of each categorical value in the Embarked column
- Calculate Mean age of passengers by sex and class

Feature Engineering:-

- Create the columns - Title, Family Size, isAlone to get the the title, family size and whether the passenger was traveling alone

Prediction:-

Use the train.csv to train a Machine Learning model of your choice & test it on the test.csv. You are required to predict if the records in test.csv survived or not. Note(1 = Survived, 0 = Dead)

- Code & Results

1. **Exploratory Data Analysis**

```
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions.{avg, col, udf, when}
import org.apache.spark.ml.feature.{OneHotEncoder, StringIndexer, VectorAssembler}
import org.apache.spark.ml.classification.LogisticRegression
import org.apache.spark.ml.{Pipeline, PipelineModel}

object Titanic extends App {
  val spark = SparkSession
    .builder()
    .appName("Titanic Dataset Analysis")
    .master("local[*]")
    .getOrCreate()

  import spark.implicits._

  val testDf = spark.read.option("header", "true").csv("assignment-spark-2/src/main/resources/test.csv")
  val trainDf = spark.read.option("header", "true").csv("assignment-spark-2/src/main/resources/train.csv")

  /** EDA */
  trainDf.show(10) // Display the first 10 rows
  trainDf.printSchema() // Print the schema of the dataset
  trainDf.describe().show() // Compute basic statistics for each column
```

```
+-----------+--------+------+--------------------+------+----+-----+-----+----------------+-------+-----+--------+
|PassengerId|Survived|Pclass|                Name|   Sex| Age|SibSp|Parch|          Ticket|   Fare|Cabin|Embarked|
+-----------+--------+------+--------------------+------+----+-----+-----+----------------+-------+-----+--------+
|          1|       0|     3|Braund, Mr. Owen ...|  male|  22|    1|    0|       A/5 21171|   7.25| null|       S|
|          2|       1|     1|Cumings, Mrs. Joh...|female|  38|    1|    0|        PC 17599|71.2833|  C85|       C|
|          3|       1|     3|Heikkinen, Miss. ...|female|  26|    0|    0|STON/O2. 3101282|  7.925| null|       S|
|          4|       1|     1|Futrelle, Mrs. Ja...|female|  35|    1|    0|          113803|   53.1| C123|       S|
|          5|       0|     3|Allen, Mr. Willia...|  male|  35|    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|           17463|51.8625|  E46|       S|
|          8|       0|     3|Palsson, Master. ...|  male|   2|    3|    1|          349909| 21.075| null|       S|
|          9|       1|     3|Johnson, Mrs. Osc...|female|  27|    0|    2|          347742|11.1333| null|       S|
|         10|       1|     2|Nasser, Mrs. Nich...|female|  14|    1|    0|          237736|30.0708| null|       C|
+-----------+--------+------+--------------------+------+----+-----+-----+----------------+-------+-----+--------+
only showing top 10 rows

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

```
+-------+-----------------+-------------------+------------------+-------------------+------+-----------------+------------------+------------------+------------
 -----+-----------------+-----+--------+
|summary|      PassengerId|           Survived|            Pclass|               Name|   Sex|              Age|             SibSp|             Parch|
 Ticket|             Fare|Cabin|Embarked|
+-------+-----------------+-------------------+------------------+-------------------+------+-----------------+------------------+------------------+------------
 -----+-----------------+-----+--------+
|  count|              891|                891|               891|                891|   891|              714|               891|               891|
   891|              891|  204|     889|
|   mean|            446.0| 0.3838383838383838| 2.308641975308642|               null|  null| 29.69911764705882|0.5230078563411896|0.3815937149270482|260318
.54916792738| 32.2042079685746| null|    null|
| stddev|257.3538420152301|0.48659245426485753|0.8360712409770491|               null|  null|14.526497332334035|1.1027434322934315| 0.8060572211299488|471609
.26868834975|49.69342859718089| null|    null|
|    min|                1|                  0|                 1|"Andersson, Mr. A...|female|             0.42|                 0|                 0|
 110152|                0|  A10|       C|
|    max|               99|                  1|                 3|van Melkebeke, Mr...|  male|                9|                 8|                 6|        WE/P
  5735|            93.5|    T|       S|
+-------+-----------------+-------------------+------------------+-------------------+------+-----------------+------------------+------------------+------------
 -----+-----------------+-----+--------+
```

```
// Count of each categorical value in the Pclass column
trainDf.groupBy("Pclass").count().show()

// Count of each categorical value in the Embarked column
trainDf.groupBy("Embarked").count().show()

// Mean age of passengers by sex and class
trainDf.groupBy("Sex", "Pclass")
  .agg(avg("Age").alias("mean_age"))
  .show()
```

```
+------+-----+
|Pclass|count|
+------+-----+
|     3|  491|
|     1|  216|
|     2|  184|
+------+-----+
```

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

```
+------+------+------------------+
|   Sex|Pclass|          mean_age|
+------+------+------------------+
|female|     3|             21.75|
|  male|     2| 30.74070707070707|
|female|     1| 34.61176470588235|
|  male|     1| 41.28138613861386|
|  male|     3|26.507588932806325|
|female|     2|28.722972972972972|
+------+------+------------------+
```

2. Feature Engineering

```
/** Feature Engineering */
// Extract the titles from the Name column and create a new column called Title
val titleRegex = """([\w]+)\. """.r
val getTitle = udf((name: String) => titleRegex.findFirstMatchIn(name).map(_.group(1)).getOrElse(""))
val trainDFWithTitle = trainDf.withColumn("Title", getTitle($"Name"))
val testDFWithTitle = trainDf.withColumn("Title", getTitle($"Name"))

// Create a new column called FamilySize by adding the SibSp and Parch columns
val trainDFWithFamilySize = trainDFWithTitle.withColumn("FamilySize", $"SibSp" + $"Parch" + 1)
val testDFWithFamilySize = testDFWithTitle.withColumn("FamilySize", $"SibSp" + $"Parch" + 1)

// Create a new column called IsAlone to indicate whether a passenger was traveling alone or with family
val trainDFWithIsAlone = trainDFWithFamilySize.withColumn("IsAlone", when($"FamilySize" === 1, 1).otherwise(0))
val testDFWithIsAlone = testDFWithFamilySize.withColumn("IsAlone", when($"FamilySize" === 1, 1).otherwise(0))

trainDFWithIsAlone.show()
testDFWithIsAlone.show()
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title	FamilySize	IsAlone
1	0	3	Braund, Mr. Owen ...	male	22	1	0	A/5 21171	7.25	null	S	Mr	2.0	0
2	1	1	Cumings, Mrs. Joh...	female	38	1	0	PC 17599	71.2833	C85	C	Mrs	2.0	0
3	1	3	Heikkinen, Miss. ...	female	26	0	0	STON/O2. 3101282	7.925	null	S	Miss	1.0	1
4	1	1	Futrelle, Mrs. Ja...	female	35	1	0	113803	53.1	C123	S	Mrs	2.0	0
5	0	3	Allen, Mr. Willia...	male	35	0	0	373450	8.05	null	S	Mr	1.0	1
6	0	3	Moran, Mr. James	male	null	0	0	330877	8.4583	null	Q	Mr	1.0	1
7	0	1	McCarthy, Mr. Tim...	male	54	0	0	17463	51.8625	E46	S	Mr	1.0	1
8	0	3	Palsson, Master. ...	male	2	3	1	349909	21.075	null	S	Master	5.0	0
9	1	3	Johnson, Mrs. Osc...	female	27	0	2	347742	11.1333	null	S	Mrs	3.0	0
10	1	2	Nasser, Mrs. Nich...	female	14	1	0	237736	30.0708	null	C	Mrs	2.0	0
11	1	3	Sandstrom, Miss. ...	female	4	1	1	PP 9549	16.7	G6	S	Miss	3.0	0
12	1	1	Bonnell, Miss. El...	female	58	0	0	113783	26.55	C103	S	Miss	1.0	1
13	0	3	Saundercock, Mr. ...	male	20	0	0	A/5. 2151	8.05	null	S	Mr	1.0	1
14	0	3	Andersson, Mr. An...	male	39	1	5	347082	31.275	null	S	Mr	7.0	0
15	0	3	Vestrom, Miss. Hu...	female	14	0	0	350406	7.8542	null	S	Miss	1.0	1
16	1	2	Hewlett, Mrs. (Ma...	female	55	0	0	248706	16	null	S	Mrs	1.0	1
17	0	3	Rice, Master. Eugene	male	2	4	1	382652	29.125	null	Q	Master	6.0	0
18	1	2	Williams, Mr. Cha...	male	null	0	0	244373	13	null	S	Mr	1.0	1
19	0	3	Vander Planke, Mr...	female	31	1	0	345763	18	null	S	Mrs	2.0	0
20	1	3	Masselmani, Mrs. ...	female	null	0	0	2649	7.225	null	C	Mrs	1.0	1

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title	FamilySize	IsAlone
1	0	3	Braund, Mr. Owen ...	male	22	1	0	A/5 21171	7.25	null	S	Mr	2.0	0
2	1	1	Cumings, Mrs. Joh...	female	38	1	0	PC 17599	71.2833	C85	C	Mrs	2.0	0
3	1	3	Heikkinen, Miss. ...	female	26	0	0	STON/O2. 3101282	7.925	null	S	Miss	1.0	1
4	1	1	Futrelle, Mrs. Ja...	female	35	1	0	113803	53.1	C123	S	Mrs	2.0	0
5	0	3	Allen, Mr. Willia...	male	35	0	0	373450	8.05	null	S	Mr	1.0	1
6	0	3	Moran, Mr. James	male	null	0	0	330877	8.4583	null	Q	Mr	1.0	1
7	0	1	McCarthy, Mr. Tim...	male	54	0	0	17463	51.8625	E46	S	Mr	1.0	1
8	0	3	Palsson, Master. ...	male	2	3	1	349909	21.075	null	S	Master	5.0	0
9	1	3	Johnson, Mrs. Osc...	female	27	0	2	347742	11.1333	null	S	Mrs	3.0	0
10	1	2	Nasser, Mrs. Nich...	female	14	1	0	237736	30.0708	null	C	Mrs	2.0	0
11	1	3	Sandstrom, Miss. ...	female	4	1	1	PP 9549	16.7	G6	S	Miss	3.0	0
12	1	1	Bonnell, Miss. El...	female	58	0	0	113783	26.55	C103	S	Miss	1.0	1
13	0	3	Saundercock, Mr. ...	male	20	0	0	A/5. 2151	8.05	null	S	Mr	1.0	1
14	0	3	Andersson, Mr. An...	male	39	1	5	347082	31.275	null	S	Mr	7.0	0
15	0	3	Vestrom, Miss. Hu...	female	14	0	0	350406	7.8542	null	S	Miss	1.0	1
16	1	2	Hewlett, Mrs. (Ma...	female	55	0	0	248706	16	null	S	Mrs	1.0	1
17	0	3	Rice, Master. Eugene	male	2	4	1	382652	29.125	null	Q	Master	6.0	0
18	1	2	Williams, Mr. Cha...	male	null	0	0	244373	13	null	S	Mr	1.0	1
19	0	3	Vander Planke, Mr...	female	31	1	0	345763	18	null	S	Mrs	2.0	0
20	1	3	Masselmani, Mrs. ...	female	null	0	0	2649	7.225	null	C	Mrs	1.0	1

only showing top 20 rows

3. Prediction

```
/** Prediction */
// Select the required columns for training and testing
val selectedTrain = trainDf.select("Survived", "Pclass", "Sex", "Age", "Fare", "Embarked")
val selectedTest = testDf.select("PassengerId", "Pclass", "Sex", "Age", "Fare", "Embarked")

// Clean the data
val cleanTrain = selectedTrain.na.drop()
val cleanTest = selectedTest.na.drop()

// Cast the data to required types
val finalTrain = cleanTrain
  .withColumn("Age", col("Age").cast("Double"))
  .withColumn("Fare", col("Fare").cast("Double"))
  .withColumn("Pclass", col("Pclass").cast("Integer"))
  .withColumn("Survived", col("Survived").cast("Integer"))

val finalTest = cleanTest
  .withColumn("Age", col("Age").cast("Double"))
  .withColumn("Fare", col("Fare").cast("Double"))
  .withColumn("Pclass", col("Pclass").cast("Integer"))
  .withColumn("PassengerId", col("PassengerId").cast("Integer"))

// Fill missing values with the mean
val ageMeanTrain = finalTrain.agg(avg("Age")).first()(0).asInstanceOf[Double]
val fareMeanTrain = finalTrain.agg(avg("Fare")).first()(0).asInstanceOf[Double]
val trainFilled = finalTrain.na.fill(ageMeanTrain, Seq("Age")).na.fill(fareMeanTrain, Seq("Fare"))

val ageMeanTest = finalTest.agg(avg("Age")).first()(0).asInstanceOf[Double]
val fareMeanTest = finalTest.agg(avg("Fare")).first()(0).asInstanceOf[Double]
val testFilled = finalTest.na.fill(ageMeanTest, Seq("Age")).na.fill(fareMeanTest, Seq("Fare"))
```

```
// Preprocessing: StringIndexer and OneHotEncoder
val sexIndexer = new StringIndexer()
  .setInputCol("Sex")
  .setOutputCol("SexIndex")

val sexEncoder = new OneHotEncoder()
  .setInputCol("SexIndex")
  .setOutputCol("SexVec")

val embarkedIndexer = new StringIndexer()
  .setInputCol("Embarked")
  .setOutputCol("EmbarkedIndex")

// OneHotEncoder
val embarkedEncoder = new OneHotEncoder()
  .setInputCol("EmbarkedIndex")
  .setOutputCol("EmbarkedVec")

val assembler = new VectorAssembler()
  .setInputCols(Array("Pclass", "SexVec", "Age", "Fare", "EmbarkedVec"))
  .setOutputCol("features")

val logisticRegModel = new LogisticRegression()
  .setFeaturesCol("features")
  .setLabelCol("Survived")

val pipeline = new Pipeline()
  .setStages(Array(sexIndexer, embarkedIndexer, sexEncoder, embarkedEncoder,
    assembler, logisticRegModel))
```

```
val modelFit: PipelineModel = pipeline.fit(trainFilled)
val results = modelFit.transform(testFilled)
val predictionResults = results.select("PassengerId", "prediction")
predictionResults.show()
predictionResults.write
  .format("csv")
  .option("header", "true")
  .option("delimiter", ",")
  .mode("overwrite")
  .save("/Users/harshilshah/Desktop/Scala assignments/Scala new assignment repo/CSYE7200-Harshil-Shah/assignment-spark-2/src/main/scala/results.csv")
```

```
+-----------+----------+
|PassengerId|prediction|
+-----------+----------+
|        892|       0.0|
|        893|       0.0|
|        894|       0.0|
|        895|       0.0|
|        896|       1.0|
|        897|       0.0|
|        898|       0.0|
|        899|       0.0|
|        900|       1.0|
|        901|       0.0|
|        903|       0.0|
|        904|       1.0|
|        905|       0.0|
|        906|       1.0|
|        907|       1.0|
|        908|       0.0|
|        909|       0.0|
|        910|       1.0|
|        911|       1.0|
```