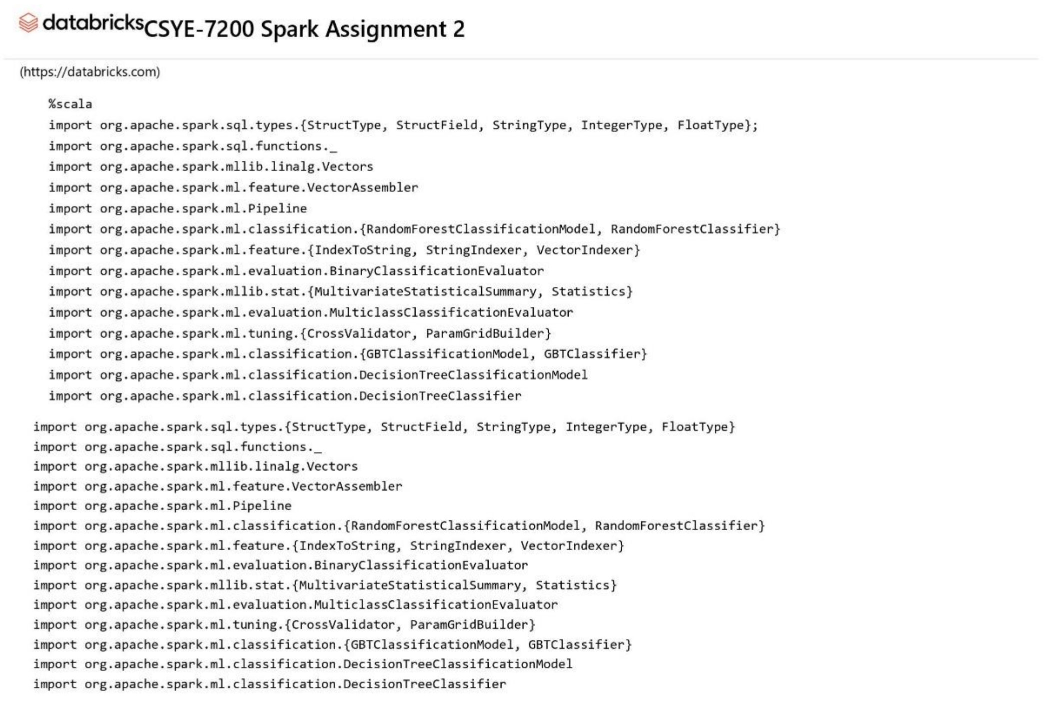
Tirth Snehal Patel – (NUID 002251943)

# Big Data System Engineering with Scala
# Fall 2023
# Spark Assignment No. 2

Below is the code implementation along with the output.


databricks CSYE-7200 Spark Assignment 2

(https://databricks.com)

```
%scala
import org.apache.spark.sql.types.{StructType, StructField, StringType, IntegerType, FloatType};
import org.apache.spark.sql.functions._
import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.classification.{RandomForestClassificationModel, RandomForestClassifier}
import org.apache.spark.ml.feature.{IndexToString, StringIndexer, VectorIndexer}
import org.apache.spark.ml.evaluation.BinaryClassificationEvaluator
import org.apache.spark.mllib.stat.{MultivariateStatisticalSummary, Statistics}
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.ml.tuning.{CrossValidator, ParamGridBuilder}
import org.apache.spark.ml.classification.{GBTClassificationModel, GBTClassifier}
import org.apache.spark.ml.classification.DecisionTreeClassificationModel
import org.apache.spark.ml.classification.DecisionTreeClassifier
```

```
import org.apache.spark.sql.types.{StructType, StructField, StringType, IntegerType, FloatType}
import org.apache.spark.sql.functions._
import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.classification.{RandomForestClassificationModel, RandomForestClassifier}
import org.apache.spark.ml.feature.{IndexToString, StringIndexer, VectorIndexer}
import org.apache.spark.ml.evaluation.BinaryClassificationEvaluator
import org.apache.spark.mllib.stat.{MultivariateStatisticalSummary, Statistics}
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.ml.tuning.{CrossValidator, ParamGridBuilder}
import org.apache.spark.ml.classification.{GBTClassificationModel, GBTClassifier}
import org.apache.spark.ml.classification.DecisionTreeClassificationModel
import org.apache.spark.ml.classification.DecisionTreeClassifier
```

```
// 1. Exploratory Data Analysis
// 1.1 Overview
// PassengerId is the unique id of the row and it doesn't have any effect on target
// Survived is the target variable we are trying to predict (0 or 1):
// 1 = Survived
// 0 = Not Survived
// Pclass (Passenger Class) is the socio-economic status of the passenger and it is a categorical ordinal feature which has 3
unique values (1, 2 or 3):
// 1 = Upper Class
// 2 = Middle Class
// 3 = Lower Class
// Name, Sex and Age are self-explanatory
// SibSp is the total number of the passengers' siblings and spouse
// Parch is the total number of the passengers' parents and children
// Ticket is the ticket number of the passenger
// Fare is the passenger fare
// Cabin is the cabin number of the passenger
// Embarked is port of embarkation and it is a categorical feature which has 3 unique values (C, Q or S):
// C = Cherbourg
// Q = Queenstown
// S = Southampton
// creating the schema for importing training and testing the datasets


val newTrainSchema = (new StructType)
.add("PassengerId", IntegerType)
.add("Survived", IntegerType)
.add("Pclass", IntegerType)
.add("Name", StringType)
.add("Sex", StringType)
.add("Age", FloatType)
.add("SibSp", IntegerType)
.add("Parch", IntegerType)
.add("Ticket", StringType)
.add("Fare", FloatType)
.add("Cabin", StringType)
.add("Embarked", StringType)

val newTestSchema = (new StructType)
.add("PassengerId", IntegerType)
.add("Pclass", IntegerType)
.add("Name", StringType)
.add("Sex", StringType)
.add("Age", FloatType)
.add("SibSp", IntegerType)
.add("Parch", IntegerType)
.add("Ticket", StringType)
.add("Fare", FloatType)
.add("Cabin", StringType)
.add("Embarked", StringType)

val trainSchema = StructType(newTrainSchema)
val testSchema = StructType(newTestSchema)
val csvFormat = "com.databricks.spark.csv"
val df_train = sqlContext.read.format(csvFormat).option("header","true").schema(trainSchema).load("/FileStore/tables/train-
4.csv")
val df_test = sqlContext.read.format(csvFormat).option("header","true").schema(testSchema).load("/FileStore/tables/test-5.csv")

//Creating table views for training and testing
df_train.createOrReplaceTempView("df_train")
df_test.createOrReplaceTempView("df_test")
```

```
newTrainSchema: org.apache.spark.sql.types.StructType = StructType(StructField(PassengerId,IntegerType,true),StructField(Survive
d,IntegerType,true),StructField(Pclass,IntegerType,true),StructField(Name,StringType,true),StructField(Sex,StringType,true),Stru
ctField(Age,FloatType,true),StructField(SibSp,IntegerType,true),StructField(Parch,IntegerType,true),StructField(Ticket,StringTyp
e,true),StructField(Fare,FloatType,true),StructField(Cabin,StringType,true),StructField(Embarked,StringType,true))
```

```
newTestSchema: org.apache.spark.sql.types.StructType = StructType(StructField(PassengerId,IntegerType,true),StructField(Pclass,I
ntegerType,true),StructField(Name,StringType,true),StructField(Sex,StringType,true),StructField(Age,FloatType,true),StructField
(SibSp,IntegerType,true),StructField(Parch,IntegerType,true),StructField(Ticket,StringType,true),StructField(Fare,FloatType,tru
e),StructField(Cabin,StringType,true),StructField(Embarked,StringType,true))
trainSchema: org.apache.spark.sql.types.StructType = StructType(StructField(PassengerId,IntegerType,true),StructField(Survived,I
ntegerType,true),StructField(Pclass,IntegerType,true),StructField(Name,StringType,true),StructField(Sex,StringType,true),StructF
ield(Age,FloatType,true),StructField(SibSp,IntegerType,true),StructField(Parch,IntegerType,true),StructField(Ticket,StringType,t
rue),StructField(Fare,FloatType,true),StructField(Cabin,StringType,true),StructField(Embarked,StringType,true))
testSchema: org.apache.spark.sql.types.StructType = StructType(StructField(PassengerId,IntegerType,true),StructField(Pclass,Inte
gerType,true),StructField(Name,StringType,true),StructField(Sex,StringType,true),StructField(Age,FloatType,true),StructField(Sib
Sp,IntegerType,true),StructField(Parch,IntegerType,true),StructField(Ticket,StringType,true),StructField(Fare,FloatType,true),St
ructField(Cabin,StringType,true),StructField(Embarked,StringType,true))
csvFormat: String = com.databricks.spark.csv
df_train: org.apache.spark.sql.DataFrame = [PassengerId: int, Survived: int ... 10 more fields]
df_test: org.apache.spark.sql.DataFrame = [PassengerId: int, Pclass: int ... 9 more fields]
```

```
//compute numcolumn summary statistics.
df_train.describe("Age", "SibSp", "Parch", "Fare").show()
```

```
+-------+------------------+------------------+-------------------+-----------------+
|summary|               Age|             SibSp|              Parch|             Fare|
+-------+------------------+------------------+-------------------+-----------------+
|  count|               714|               891|                891|              891|
|   mean| 29.69911764704046|0.5230078563411896|0.38159371492704824|32.20420804114722|
| stddev|14.526497332370992|1.1027434322934315| 0.8060572211299488|49.69342916316158|
|    min|              0.42|                 0|                  0|              0.0|
|    max|              80.0|                 8|                  6|         512.3292|
+-------+------------------+------------------+-------------------+-----------------+
```

```
df_train.show()
```

```
+-----------+--------+------+--------------------+------+----+-----+-----+----------------+-------+-----+--------+
|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|
|         11|       1|     3|Sandstrom, Miss. ...|female| 4.0|    1|    1|         PP 9549|   16.7|   G6|       S|
|         12|       1|     1|Bonnell, Miss. El...|female|58.0|    0|    0|          113783|  26.55| C103|       S|
|         13|       0|     3|Saundercock, Mr. ...|  male|20.0|    0|    0|       A/5. 2151|   8.05| null|       S|
|         14|       0|     3|Andersson, Mr. An...|  male|39.0|    1|    5|          347082| 31.275| null|       S|
|         15|       0|     3|Vestrom, Miss. Hu...|female|14.0|    0|    0|          350406| 7.8542| null|       S|
|         16|       1|     2|Hewlett, Mrs. (Ma...|female|55.0|    0|    0|          248706|   16.0| null|       S|
|         17|       0|     3|Rice, Master. Eugene|  male| 2.0|    4|    1|          382652| 29.125| null|       Q|
|         18|       1|     2|Williams, Mr. Cha...|  male|null|    0|    0|          244373|   13.0| null|       S|
```

```
//Summary stats for cateogrical columns
sqlContext.sql("select Survived, count(*) from df_train group by Survived").show()
```

```
+--------+--------+
|Survived|count(1)|
+--------+--------+
|       1|     342|
|       0|     549|
+--------+--------+
sqlContext.sql("select Pclass, Survived, count(*) from df_train group by Pclass, Survived").show()
```

```
+------+--------+--------+
|Pclass|Survived|count(1)|
+------+--------+--------+
|     1|       0|      80|
|     3|       1|     119|
|     1|       1|     136|
|     2|       1|      87|
|     2|       0|      97|
|     3|       0|     372|
+------+--------+--------+
```

```
sqlContext.sql("select Sex, Survived, count(*) from df_train group by Sex,Survived").show()
```

```
+------+--------+--------+
|   Sex|Survived|count(1)|
+------+--------+--------+
|  male|       0|     468|
|female|       1|     233|
|female|       0|      81|
|  male|       1|     109|
+------+--------+--------+
```

```
//Calculating avg Age and Fare to fill null values for training
val AvgAge = df_train.select("Age")
  .agg(avg("Age"))
  .collect() match {
    case Array(Row(avg: Double)) => avg
    case _ => 0
  }
```

```
AvgAge: Double = 29.69911764704046
```

```
//Calculating the average fare for filling gaps in dataset train
val AvgFare = df_train.select("Fare")
  .agg(avg("Fare"))
  .collect() match {
    case Array(Row(avg: Double)) => avg
    case _ => 0
  }
```

```
AvgFare: Double = 32.20420804114722
```

```
//Calculate avg Age and Fare to fill null values for test data
val AvgAge_test = df_test.select("Age")
  .agg(avg("Age"))
  .collect() match {
  case Array(Row(avg: Double)) => avg
  case _ => 0
}
```

```
AvgAge_test: Double = 30.272590361490668
```

```
//Calculate average fare for filling gaps in dataset test
val AvgFare_test = df_test.select("Fare")
  .agg(avg("Fare"))
  .collect() match {
  case Array(Row(avg: Double)) => avg
  case _ => 0
}
```

```
AvgFare_test: Double = 35.62718864996656
```

```
// for training
val embarked: (String => String) = {
  case "" => "S"
  case null => "S"
  case a => a
}
val embarkedUDF = udf(embarked)
```

```
embarked: String => String = $Lambda$8107/1598529161@41d4b905
embarkedUDF: org.apache.spark.sql.expressions.UserDefinedFunction = SparkUserDefinedFunction($Lambda$8107/1598529161@41d4b905,String
Type,List(Some(class[value[0]: string])),Some(class[value[0]: string]),None,true,true)
```

```
//for test
val embarked_test: (String => String) = {
  case "" => "S"
  case null => "S"
  case a => a
}
val embarkedUDF_test = udf(embarked_test)
```

```
embarked_test: String => String = $Lambda$8119/1680966287@7f24d46
embarkedUDF_test: org.apache.spark.sql.expressions.UserDefinedFunction = SparkUserDefinedFunction($Lambda$8119/1680966287@7f24d46,St
ringType,List(Some(class[value[0]: string])),Some(class[value[0]: string]),None,true,true)
```

```
//Filling null values with avg values for training dataset
val imputeddf = df_train.na.fill(Map("Fare" -> AvgFare, "Age" -> AvgAge))
val imputeddf2 = imputeddf.withColumn("Embarked", embarkedUDF(imputeddf.col("Embarked")))

//splitting training data into training and validation
val Array(trainingData, validationData) = imputeddf2.randomSplit(Array(0.7, 0.3))

//Filling null values with avg values for test dataset
val imputeddf_test = df_test.na.fill(Map("Fare" -> AvgFare_test, "Age" -> AvgAge_test))
val imputeddf2_test = imputeddf_test.withColumn("Embarked", embarkedUDF_test(imputeddf_test.col("Embarked")))
```

```
imputeddf: org.apache.spark.sql.DataFrame = [PassengerId: int, Survived: int ... 10 more fields]
imputeddf2: org.apache.spark.sql.DataFrame = [PassengerId: int, Survived: int ... 10 more fields]
trainingData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [PassengerId: int, Survived: int ... 10 more fields]
validationData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [PassengerId: int, Survived: int ... 10 more fields]
imputeddf_test: org.apache.spark.sql.DataFrame = [PassengerId: int, Pclass: int ... 9 more fields]
imputeddf2_test: org.apache.spark.sql.DataFrame = [PassengerId: int, Pclass: int ... 9 more fields]
```

```
// Feature Engineering - Create new attributes that may be derived from the existing attributes. This may include removing
certain columns in the dataset.

//Dropping Cabin feature as it has so many null values
// val df1_train = trainingData.drop("Cabin")

// val df1_test = imputeddf2_test.drop("Cabin")



import org.apache.spark.ml.feature.Bucketizer
import org.apache.spark.sql.functions._
import org.apache.spark.sql.types._



// Define the bucketizer splits based on the quantiles
val splits = Array(Double.NegativeInfinity, 7.55, 7.854, 8.05, 10.5, 14.4542, 21.6792, 27.0, 39.6875, 77.9583, 512.329,
Double.PositiveInfinity)

// Create a Bucketizer transformer
val bucketizer = new Bucketizer()
  .setInputCol("Fare")  // Specify the input column
  .setOutputCol("FareBucket") // Specify the output column
  .setSplits(splits)

// Apply the bucketizer transformation to the DataFrame
val df_binned = bucketizer.transform(df_train)

// Now df_binned contains the 'FareBucket' column, which represents the binned 'Fare' values
```

```
import org.apache.spark.ml.feature.Bucketizer
import org.apache.spark.sql.functions._
import org.apache.spark.sql.types._
splits: Array[Double] = Array(-Infinity, 7.55, 7.854, 8.05, 10.5, 14.4542, 21.6792, 27.0, 39.6875, 77.9583, 512.329, Infinity)
bucketizer: org.apache.spark.ml.feature.Bucketizer = Bucketizer: uid=bucketizer_41eda3470974
df_binned: org.apache.spark.sql.DataFrame = [PassengerId: int, Survived: int ... 11 more fields]
```

```
df_binned.show()
```

```
+-----------+--------+------+--------------------+------+----+-----+-----+----------------+-------+-----+--------+----------+
|PassengerId|Survived|Pclass|                Name|   Sex| Age|SibSp|Parch|          Ticket|   Fare|Cabin|Embarked|FareBucket|
+-----------+--------+------+--------------------+------+----+-----+-----+----------------+-------+-----+--------+----------+
|          1|       0|     3|Braund, Mr. Owen ...|  male|22.0|    1|    0|       A/5 21171|   7.25| null|       S|       0.0|
|          2|       1|     1|Cumings, Mrs. Joh...|female|38.0|    1|    0|        PC 17599|71.2833|  C85|       C|       8.0|
|          3|       1|     3|Heikkinen, Miss. ...|female|26.0|    0|    0|STON/O2. 3101282|  7.925| null|       S|       2.0|
|          4|       1|     1|Futrelle, Mrs. Ja...|female|35.0|    1|    0|          113803|   53.1| C123|       S|       8.0|
|          5|       0|     3|Allen, Mr. Willia...|  male|35.0|    0|    0|          373450|   8.05| null|       S|       3.0|
|          6|       0|     3|    Moran, Mr. James|  male|null|    0|    0|          330877| 8.4583| null|       Q|       3.0|
|          7|       0|     1|McCarthy, Mr. Tim...|  male|54.0|    0|    0|           17463|51.8625|  E46|       S|       8.0|
|          8|       0|     3|Palsson, Master. ...|  male| 2.0|    3|    1|          349909| 21.075| null|       S|       5.0|
|          9|       1|     3|Johnson, Mrs. Osc...|female|27.0|    0|    2|          347742|11.1333| null|       S|       4.0|
|         10|       1|     2|Nasser, Mrs. Nich...|female|14.0|    1|    0|          237736|30.0708| null|       C|       7.0|
|         11|       1|     3|Sandstrom, Miss. ...|female| 4.0|    1|    1|         PP 9549|   16.7|   G6|       S|       5.0|
|         12|       1|     1|Bonnell, Miss. El...|female|58.0|    0|    0|          113783|  26.55| C103|       S|       6.0|
|         13|       0|     3|Saundercock, Mr. ...|  male|20.0|    0|    0|       A/5. 2151|   8.05| null|       S|       3.0|
|         14|       0|     3|Andersson, Mr. An...|  male|39.0|    1|    5|          347082| 31.275| null|       S|       7.0|
|         15|       0|     3|Vestrom, Miss. Hu...|female|14.0|    0|    0|          350406| 7.8542| null|       S|       2.0|
|         16|       1|     2|Hewlett, Mrs. (Ma...|female|55.0|    0|    0|          248706|   16.0| null|       S|       5.0|
|         17|       0|     3|Rice, Master. Eugene|  male| 2.0|    4|    1|          382652| 29.125| null|       Q|       7.0|
|         18|       1|     2|Williams, Mr. Cha...|  male|null|    0|    0|          244373|   13.0| null|       S|       4.0|
import org.apache.spark.ml.feature.Bucketizer
import org.apache.spark.sql.functions._
import org.apache.spark.sql.types._



// Define the bucketizer splits for quantiles
val splits = Array(Double.NegativeInfinity, 19.0, 22.0, 25.0, 28.0, 31.0, 35.0, 39.0, 45.0, 55.0, 65.0, Double.PositiveInfinity)

// Create a Bucketizer transformer
val bucketizer = new Bucketizer()
  .setInputCol("Age")         // Specify the input column
  .setOutputCol("AgeBucket")  // Specify the output column
  .setSplits(splits)

// Apply the bucketizer transformation to the DataFrame
val df_binned = bucketizer.transform(df_tr)

// Now df_binned contains the 'AgeBucket' column, which represents the binned 'Age' values
```

```
import org.apache.spark.ml.feature.Bucketizer
import org.apache.spark.sql.functions._
import org.apache.spark.sql.types._
splits: Array[Double] = Array(-Infinity, 19.0, 22.0, 25.0, 28.0, 31.0, 35.0, 39.0, 45.0, 55.0, 65.0, Infinity)
bucketizer: org.apache.spark.ml.feature.Bucketizer = Bucketizer: uid=bucketizer_55340fd47ead
df_binned: org.apache.spark.sql.DataFrame = [PassengerId: int, Survived: int ... 11 more fields]
```

```
df_binned.show()
```

```
+-----------+--------+------+--------------------+------+----+-----+-----+----------------+-------+-----+--------+---------+
|PassengerId|Survived|Pclass|                Name|   Sex| Age|SibSp|Parch|          Ticket|   Fare|Cabin|Embarked|AgeBucket|
+-----------+--------+------+--------------------+------+----+-----+-----+----------------+-------+-----+--------+---------+
|          1|       0|     3|Braund, Mr. Owen ...|  male|22.0|    1|    0|       A/5 21171|   7.25| null|       S|      2.0|
|          2|       1|     1|Cumings, Mrs. Joh...|female|38.0|    1|    0|        PC 17599|71.2833|  C85|       C|      6.0|
|          3|       1|     3|Heikkinen, Miss. ...|female|26.0|    0|    0|STON/O2. 3101282|  7.925| null|       S|      3.0|
|          4|       1|     1|Futrelle, Mrs. Ja...|female|35.0|    1|    0|          113803|   53.1| C123|       S|      6.0|
|          5|       0|     3|Allen, Mr. Willia...|  male|35.0|    0|    0|          373450|   8.05| null|       S|      6.0|
|          6|       0|     3|    Moran, Mr. James|  male|null|    0|    0|          330877| 8.4583| null|       Q|     null|
|          7|       0|     1|McCarthy, Mr. Tim...|  male|54.0|    0|    0|           17463|51.8625|  E46|       S|      8.0|
|          8|       0|     3|Palsson, Master. ...|  male| 2.0|    3|    1|          349909| 21.075| null|       S|      0.0|
|          9|       1|     3|Johnson, Mrs. Osc...|female|27.0|    0|    2|          347742|11.1333| null|       S|      3.0|
|         10|       1|     2|Nasser, Mrs. Nich...|female|14.0|    1|    0|          237736|30.0708| null|       C|      0.0|
|         11|       1|     3|Sandstrom, Miss. ...|female| 4.0|    1|    1|         PP 9549|   16.7|   G6|       S|      0.0|
|         12|       1|     1|Bonnell, Miss. El...|female|58.0|    0|    0|          113783|  26.55| C103|       S|      9.0|
|         13|       0|     3|Saundercock, Mr. ...|  male|20.0|    0|    0|       A/5. 2151|   8.05| null|       S|      1.0|
|         14|       0|     3|Andersson, Mr. An...|  male|39.0|    1|    5|          347082| 31.275| null|       S|      7.0|
|         15|       0|     3|Vestrom, Miss. Hu...|female|14.0|    0|    0|          350406| 7.8542| null|       S|      0.0|
|         16|       1|     2|Hewlett, Mrs. (Ma...|female|55.0|    0|    0|          248706|   16.0| null|       S|      9.0|
|         17|       0|     3|Rice, Master. Eugene|  male| 2.0|    4|    1|          382652| 29.125| null|       Q|      0.0|
|         18|       1|     2|Williams, Mr. Cha...|  male|null|    0|    0|          244373|   13.0| null|       S|     null|
```

```
import org.apache.spark.sql.functions._
import org.apache.spark.sql.DataFrame



// Adding a new column 'Family_Size' to df_all
val df_with_family_size: DataFrame = df_binned.withColumn("Family_Size", col("SibSp") + col("Parch") + lit(1))

// Now df_with_family_size contains the 'Family_Size' column
```

```
import org.apache.spark.sql.functions._
import org.apache.spark.sql.DataFrame
df_with_family_size: org.apache.spark.sql.DataFrame = [PassengerId: int, Survived: int ... 12 more fields]
```

```
df_with_family_size.show()
```

```
+-----------+--------+------+--------------------+------+----+-----+-----+----------------+-------+-----+--------+---------+---
--------+
|PassengerId|Survived|Pclass|                Name|   Sex| Age|SibSp|Parch|          Ticket|   Fare|Cabin|Embarked|AgeBucket|Fam
ily_Size|
+-----------+--------+------+--------------------+------+----+-----+-----+----------------+-------+-----+--------+---------+---
--------+
|          1|       0|     3|Braund, Mr. Owen ...|  male|22.0|    1|    0|       A/5 21171|   7.25| null|       S|      2.0|
2|
|          2|       1|     1|Cumings, Mrs. Joh...|female|38.0|    1|    0|        PC 17599|71.2833|  C85|       C|      6.0|
2|
|          3|       1|     3|Heikkinen, Miss. ...|female|26.0|    0|    0|STON/O2. 3101282|  7.925| null|       S|      3.0|
1|
|          4|       1|     1|Futrelle, Mrs. Ja...|female|35.0|    1|    0|          113803|   53.1| C123|       S|      6.0|
2|
|          5|       0|     3|Allen, Mr. Willia...|  male|35.0|    0|    0|          373450|   8.05| null|       S|      6.0|
1|
|          6|       0|     3|    Moran, Mr. James|  male|null|    0|    0|          330877| 8.4583| null|       Q|     null|
1|
|          7|       0|     1|McCarthy, Mr. Tim...|  male|54.0|    0|    0|           17463|51.8625|  E46|       S|      8.0|
1|
|          8|       0|     3|Palsson, Master. ...|  male| 2.0|    3|    1|          349909| 21.075| null|       S|      0.0|
```

```
import org.apache.spark.sql.expressions.Window
import org.apache.spark.sql.functions._
import org.apache.spark.sql.DataFrame



// Define a Window specification to partition by 'Ticket' column
val windowSpec = Window.partitionBy("Ticket")

// Add a new column 'Ticket_Frequency' to df_all
val df_with_ticket_frequency: DataFrame = df_with_family_size.withColumn("Ticket_Frequency", count("*").over(windowSpec))

// Now df_with_ticket_frequency contains the 'Ticket_Frequency' column
```

```
import org.apache.spark.sql.expressions.Window
import org.apache.spark.sql.functions._
import org.apache.spark.sql.DataFrame
windowSpec: org.apache.spark.sql.expressions.WindowSpec = org.apache.spark.sql.expressions.WindowSpec@3e8679a5
df_with_ticket_frequency: org.apache.spark.sql.DataFrame = [PassengerId: int, Survived: int ... 13 more fields]
```

```
df_with_ticket_frequency.show()
```

```
+-----------+--------+------+--------------------+------+----+-----+-----+------+-------+-----+--------+---------+-----------+-
---------------+
|PassengerId|Survived|Pclass|                Name|   Sex| Age|SibSp|Parch|Ticket|   Fare|Cabin|Embarked|AgeBucket|Family_Size|T
icket_Frequency|
+-----------+--------+------+--------------------+------+----+-----+-----+------+-------+-----+--------+---------+-----------+-
---------------+
|        258|       1|     1|Cherry, Miss. Gladys|female|30.0|    0|    0|110152|   86.5|  B77|       S|      4.0|          1|
3|
|        505|       1|     1|Maioni, Miss. Rob...|female|16.0|    0|    0|110152|   86.5|  B79|       S|      0.0|          1|
3|
|        760|       1|     1|Rothes, the Count...|female|33.0|    0|    0|110152|   86.5|  B77|       S|      5.0|          1|
3|
|        263|       0|     1|   Taussig, Mr. Emil|  male|52.0|    1|    1|110413|  79.65|  E67|       S|      8.0|          3|
3|
|        559|       1|     1|Taussig, Mrs. Emi...|female|39.0|    1|    1|110413|  79.65|  E67|       S|      7.0|          3|
3|
|        586|       1|     1| Taussig, Miss. Ruth|female|18.0|    0|    2|110413|  79.65|  E68|       S|      0.0|          3|
3|
|        111|       0|     1|Porter, Mr. Walte...|  male|47.0|    0|    0|110465|   52.0| C110|       S|      8.0|          1|
2|
```

```scala
import org.apache.spark.sql.{DataFrame, SparkSession}
import org.apache.spark.sql.functions._



// Initialize a SparkSession
val spark = SparkSession.builder()
  .appName("DataFrame Transformation")
  .getOrCreate()

// Add a new column 'Title' by splitting the 'Name' column
val df_with_title: DataFrame = df_with_ticket_frequency.withColumn("Title", split(col("Name"), ", ")(1))
  .withColumn("Title", split(col("Title"), "\\.")(0))

// Add a new column 'Is_Married' and set it to 1 for 'Title' equal to 'Mrs'
val df_with_is_married: DataFrame = df_with_title.withColumn("Is_Married", when(col("Title") === "Mrs", 1).otherwise(0))

// Now df_with_is_married contains the 'Title' and 'Is_Married' columns
```

```
import org.apache.spark.sql.{DataFrame, SparkSession}
import org.apache.spark.sql.functions._
spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@3d025628
df_with_title: org.apache.spark.sql.DataFrame = [PassengerId: int, Survived: int ... 14 more fields]
df_with_is_married: org.apache.spark.sql.DataFrame = [PassengerId: int, Survived: int ... 15 more fields]
```

```scala
df_with_is_married.show()
```

```
+-----------+--------+------+--------------------+------+----+-----+-----+------+-------+-----+--------+---------+-----------+-
---------------+------------+----------+
|PassengerId|Survived|Pclass|                Name|   Sex| Age|SibSp|Parch|Ticket|   Fare|Cabin|Embarked|AgeBucket|Family_Size|T
icket_Frequency|       Title|Is_Married|
+-----------+--------+------+--------------------+------+----+-----+-----+------+-------+-----+--------+---------+-----------+-
---------------+------------+----------+
|        258|       1|     1|Cherry, Miss. Gladys|female|30.0|    0|    0|110152|   86.5|  B77|       S|      4.0|          1|
3|        Miss|         0|
|        505|       1|     1|Maioni, Miss. Rob...|female|16.0|    0|    0|110152|   86.5|  B79|       S|      0.0|          1|
3|        Miss|         0|
|        760|       1|     1|Rothes, the Count...|female|33.0|    0|    0|110152|   86.5|  B77|       S|      5.0|          1|
3|the Countess|         0|
|        263|       0|     1|   Taussig, Mr. Emil|  male|52.0|    1|    1|110413|  79.65|  E67|       S|      8.0|          3|
3|          Mr|         0|
|        559|       1|     1|Taussig, Mrs. Emi...|female|39.0|    1|    1|110413|  79.65|  E67|       S|      7.0|          3|
3|         Mrs|         1|
|        586|       1|     1| Taussig, Miss. Ruth|female|18.0|    0|    2|110413|  79.65|  E68|       S|      0.0|          3|
3|        Miss|         0|
|        111|       0|     1|Porter, Mr. Walte...|  male|47.0|    0|    0|110465|   52.0| C110|       S|      8.0|          1|
2|          Mr|         0|
|        476|       0|     1|Clifford, Mr. Geo...|  male|null|    0|    0|110465|   52.0|  A14|       S|     null|          1|
```

```scala
import org.apache.spark.sql.functions._
import org.apache.spark.sql.{DataFrame, SparkSession}
import java.util.regex.Pattern
import java.util.regex.Matcher

// Assuming you have a DataFrame df_all

// Initialize a SparkSession
val spark = SparkSession.builder()
  .appName("DataFrame Transformation")
  .getOrCreate()

// Define a user-defined function (UDF) to extract surnames
val extractSurname = udf((name: String) => {
  val nameNoBracket = if (name.contains("(")) name.split("\\(")(0) else name
  val family = nameNoBracket.split(",")(0).replaceAll("[^a-zA-Z]", "").trim
  family
})

// Add a new column 'Family' by applying the UDF to the 'Name' column
val df_with_family: DataFrame = df_with_is_married.withColumn("Family", extractSurname(col("Name")))

// Now df_train and df_test are the training and testing DataFrames, and df_with_family contains the 'Family' column
```

```
import org.apache.spark.sql.functions._
import org.apache.spark.sql.{DataFrame, SparkSession}
import java.util.regex.Pattern
import java.util.regex.Matcher
spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@3d025628
extractSurname: org.apache.spark.sql.expressions.UserDefinedFunction = SparkUserDefinedFunction($Lambda$8827/125107648@b431115,Strin
gType,List(Some(class[value[0]: string])),Some(class[value[0]: string]),None,true,true)
df_with_family: org.apache.spark.sql.DataFrame = [PassengerId: int, Survived: int ... 16 more fields]
df_train: org.apache.spark.sql.DataFrame = [PassengerId: int, Survived: int ... 16 more fields]
df_test: org.apache.spark.sql.DataFrame = [PassengerId: int, Survived: int ... 16 more fields]
```

```scala
df_with_family.show()
```

```
+-----------+--------+------+--------------------+------+----+-----+-----+------+-------+-----+--------+---------+-----------+-
---------------+-----------+----------+--------------------+
|PassengerId|Survived|Pclass|                Name|   Sex| Age|SibSp|Parch|Ticket|   Fare|Cabin|Embarked|AgeBucket|Family_Size|T
icket_Frequency|      Title|Is_Married|              Family|
+-----------+--------+------+--------------------+------+----+-----+-----+------+-------+-----+--------+---------+-----------+-
---------------+-----------+----------+--------------------+
|        258|       1|     1|Cherry, Miss. Gladys|female|30.0|    0|    0|110152|   86.5|  B77|       S|      4.0|          1|
3|       Miss|         0|              Cherry|
|        505|       1|     1|Maioni, Miss. Rob...|female|16.0|    0|    0|110152|   86.5|  B79|       S|      0.0|          1|
3|       Miss|         0|              Maioni|
|        760|       1|     1|Rothes, the Count...|female|33.0|    0|    0|110152|   86.5|  B77|       S|      5.0|          1|
3|the Countess|         0|              Rothes|
|        263|       0|     1|   Taussig, Mr. Emil|  male|52.0|    1|    1|110413|  79.65|  E67|       S|      8.0|          3|
3|         Mr|         0|             Taussig|
|        559|       1|     1|Taussig, Mrs. Emi...|female|39.0|    1|    1|110413|  79.65|  E67|       S|      7.0|          3|
3|        Mrs|         1|             Taussig|
|        586|       1|     1| Taussig, Miss. Ruth|female|18.0|    0|    2|110413|  79.65|  E68|       S|      0.0|          3|
3|       Miss|         0|             Taussig|
|        111|       0|     1|Porter, Mr. Walte...|  male|47.0|    0|    0|110465|   52.0| C110|       S|      8.0|          1|
2|         Mr|         0|              Porter|
|        476|       0|     1|Clifford, Mr. Geo...|  male|null|    0|    0|110465|   52.0|  A14|       S|     null|          1|
//Indexing categorical features
val catFeatColNames = Seq("Pclass", "Sex", "Embarked")
val stringIndexers = catFeatColNames.map { colName =>
  new StringIndexer()
    .setInputCol(colName)
    .setOutputCol(colName + "Indexed")
    .fit(trainingData)
}

//Indexing target feature
val labelIndexer = new StringIndexer()
.setInputCol("Survived")
.setOutputCol("SurvivedIndexed")
.fit(trainingData)

//Assembling features into one vector
val numFeatColNames = Seq("Age", "SibSp", "Parch", "Fare")
val idxdCatFeatColName = catFeatColNames.map(_ + "Indexed")
val allIdxdFeatColNames = numFeatColNames ++ idxdCatFeatColName
val assembler = new VectorAssembler()
  .setInputCols(Array(allIdxdFeatColNames: _*))
  .setOutputCol("Features")
```

```
catFeatColNames: Seq[String] = List(Pclass, Sex, Embarked)
stringIndexers: Seq[org.apache.spark.ml.feature.StringIndexerModel] = List(StringIndexerModel: uid=strIdx_52c20ddf3e58, handleInvali
d=error, StringIndexerModel: uid=strIdx_510c7847544f, handleInvalid=error, StringIndexerModel: uid=strIdx_16154abc0509, handleInvali
d=error)
labelIndexer: org.apache.spark.ml.feature.StringIndexerModel = StringIndexerModel: uid=strIdx_d082391f2c11, handleInvalid=error
numFeatColNames: Seq[String] = List(Age, SibSp, Parch, Fare)
idxdCatFeatColName: Seq[String] = List(PclassIndexed, SexIndexed, EmbarkedIndexed)
allIdxdFeatColNames: Seq[String] = List(Age, SibSp, Parch, Fare, PclassIndexed, SexIndexed, EmbarkedIndexed)
assembler: org.apache.spark.ml.feature.VectorAssembler = VectorAssembler: uid=vecAssembler_fe05be893203, handleInvalid=error, numInp
utCols=7
```

```
//Training the model from the pool and determining the best one among them.
//Randomforest classifier
val randomforest = new RandomForestClassifier()
  .setLabelCol("SurvivedIndexed")
  .setFeaturesCol("Features")

//Retrieving original labels
val labelConverter = new IndexToString()
  .setInputCol("prediction")
  .setOutputCol("predictedLabel")
  .setLabels(labelIndexer.labelsArray.flatten)

//Creating pipeline
val pipeline = new Pipeline().setStages(
  (stringIndexers :+ labelIndexer :+ assembler :+ randomforest :+ labelConverter).toArray)
```

```
randomforest: org.apache.spark.ml.classification.RandomForestClassifier = rfc_d39929b5363e
labelConverter: org.apache.spark.ml.feature.IndexToString = idxToStr_deec7f48ee96
pipeline: org.apache.spark.ml.Pipeline = pipeline_f7c61e980795
```

```
  //Selecting best model
  val paramGrid = new ParamGridBuilder()
    .addGrid(randomforest.maxBins, Array(25, 28, 31))
    .addGrid(randomforest.maxDepth, Array(4, 6, 8))
    .addGrid(randomforest.impurity, Array("entropy", "gini"))
    .build()

  val evaluator = new BinaryClassificationEvaluator()
    .setLabelCol("SurvivedIndexed")
    .setMetricName("areaUnderPR")

  //Cross validator with 10 fold
  val cv = new CrossValidator()
    .setEstimator(pipeline)
    .setEvaluator(evaluator)
    .setEstimatorParamMaps(paramGrid)
    .setNumFolds(10)
```

```
paramGrid: Array[org.apache.spark.ml.param.ParamMap] =
Array({
	rfc_d39929b5363e-impurity: entropy,
	rfc_d39929b5363e-maxBins: 25,
	rfc_d39929b5363e-maxDepth: 4
}, {
	rfc_d39929b5363e-impurity: gini,
	rfc_d39929b5363e-maxBins: 25,
	rfc_d39929b5363e-maxDepth: 4
}, {
	rfc_d39929b5363e-impurity: entropy,
	rfc_d39929b5363e-maxBins: 25,
	rfc_d39929b5363e-maxDepth: 6
}, {
	rfc_d39929b5363e-impurity: gini,
	rfc_d39929b5363e-maxBins: 25,
	rfc_d39929b5363e-maxDepth: 6
}, {
	rfc_d39929b5363e-impurity: entropy,
	rfc_d39929b5363e-maxBins: 25,
	rfc_d39929b5363e-maxDepth: 8
```

```
  //Fitting model using cross validation
  val crossValidatorModel = cv.fit(trainingData)

  //predictions on validation data
  val predictions = crossValidatorModel.transform(validationData)

  //Accuracy
  val accuracy = evaluator.evaluate(predictions)
  println("Test Error DT= " + (1.0 - accuracy))
```

```
Test Error DT= 0.19512230928774577
crossValidatorModel: org.apache.spark.ml.tuning.CrossValidatorModel = CrossValidatorModel: uid=cv_ce476272474c, bestModel=pipeline_f
7c61e980795, numFolds=10
predictions: org.apache.spark.sql.DataFrame = [PassengerId: int, Survived: int ... 19 more fields]
accuracy: Double = 0.8048776907122542

Test Error = 0.20172748564799914
gbt: org.apache.spark.ml.classification.GBTClassifier = gbtc_145212851005
pipeline: org.apache.spark.ml.Pipeline = pipeline_200b998877e7
model: org.apache.spark.ml.PipelineModel = pipeline_200b998877e7
predictions: org.apache.spark.sql.DataFrame = [PassengerId: int, Survived: int ... 19 more fields]
evaluator: org.apache.spark.ml.evaluation.BinaryClassificationEvaluator = BinaryClassificationEvaluator: uid=binEval_913ee5288d5a, m
etricName=areaUnderPR, numBins=1000
accuracy: Double = 0.7982725143520009
```