Sameeksha Gurrala – SEC01 (NUID 002922061)

Big Data System Engineering with Scala Spring 2023

Spark Assignmnet -2

-List of Tasks Implemented

Exploratory Data Analysis- Follow up on the previous spark assignment 1 and explain a few statistics. (20 pts)

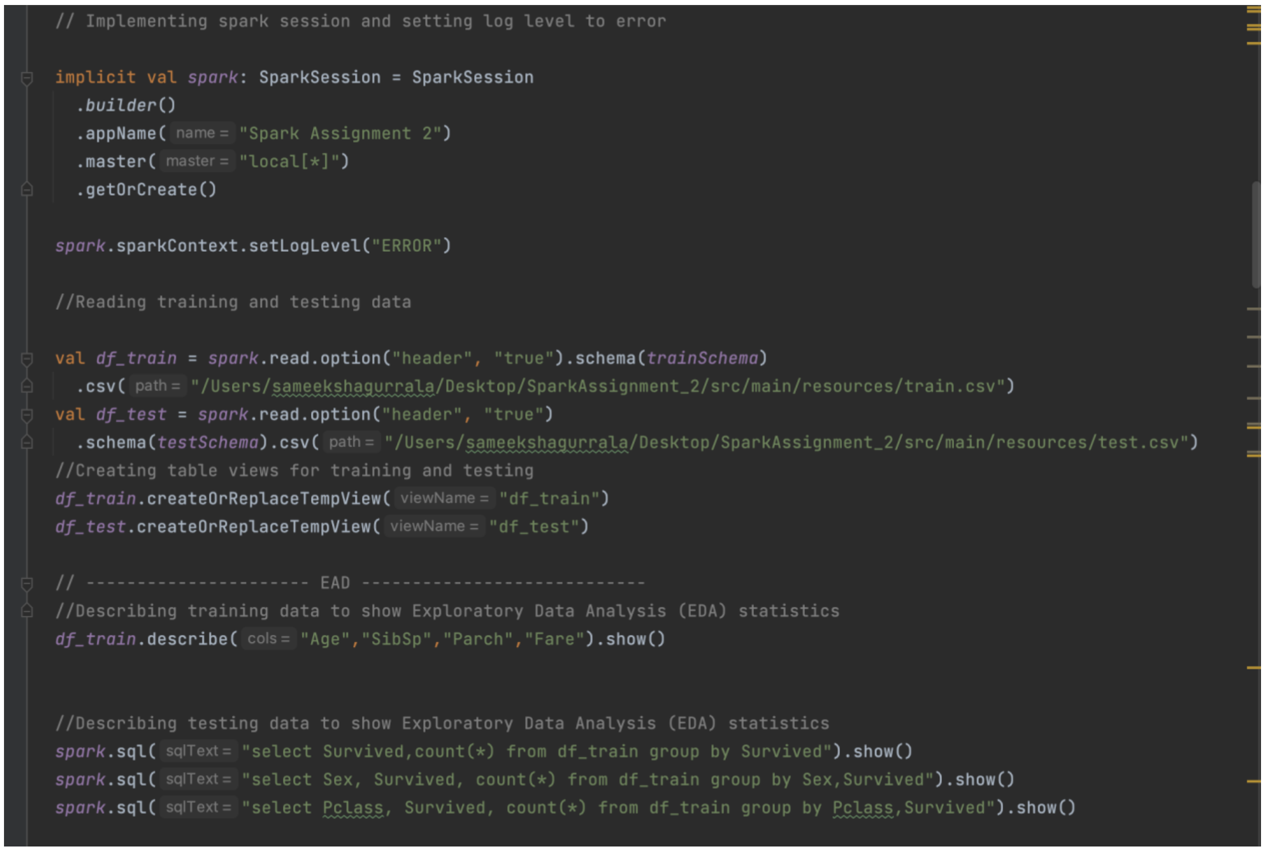
Feature Engineering - Create new attributes that may be derived from the existing attributes. This may include removing certain columns in the dataset. (30 pts)

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) (50 pts)

-Code

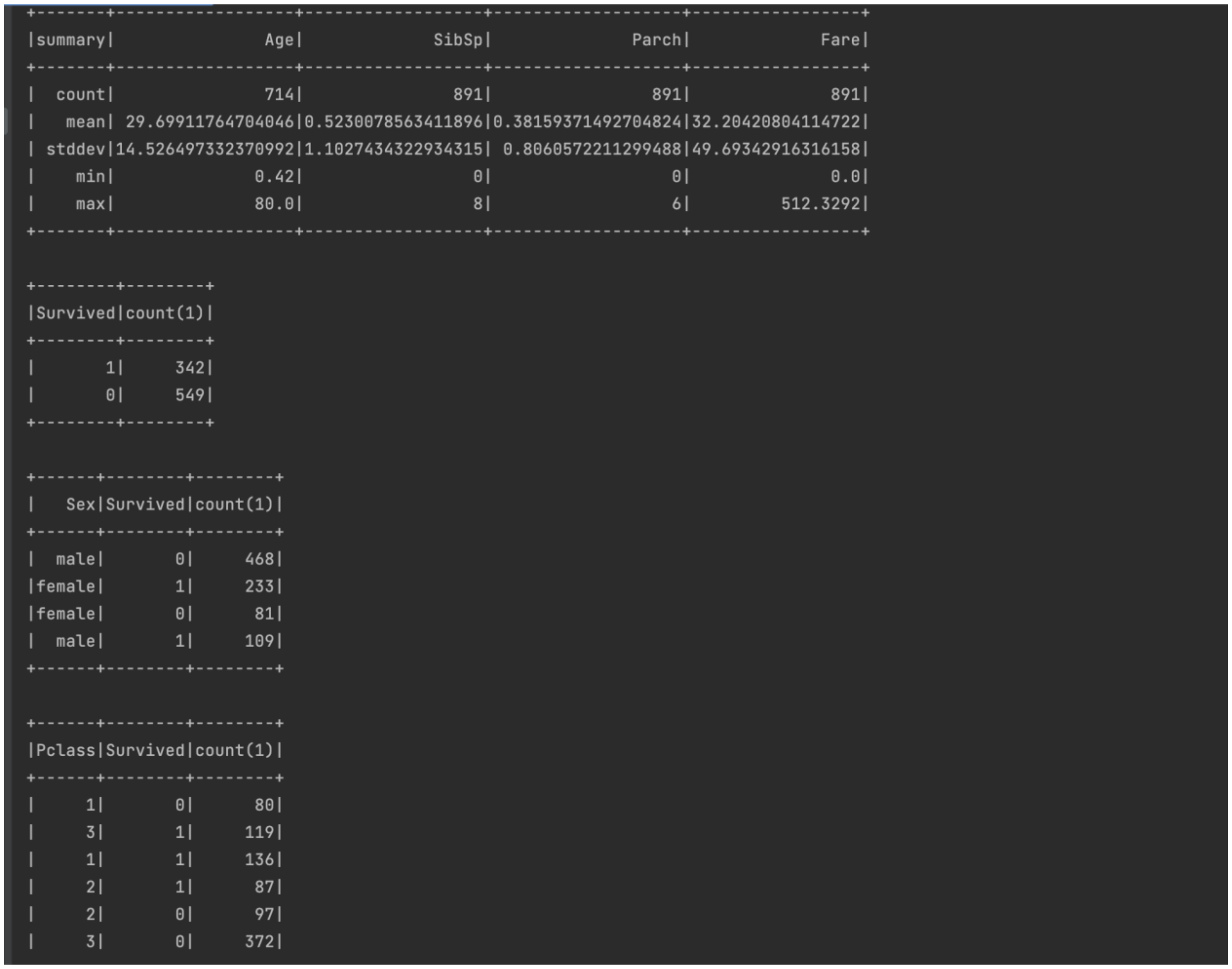
Exploratory Data Analysis : Refers to the critical process of performing initial investigations on data to discover pattern and spot outliers with help of statistics and graphical visualization.

Used statistical approach to process the data for EAD. Mean, std dev, max and min are presented for test and train dataset.



```
// Implementing spark session and setting log level to error

implicit val spark: SparkSession = SparkSession
  .builder()
  .appName( name = "Spark Assignment 2")
  .master( master = "local[*]")
  .getOrCreate()

spark.sparkContext.setLogLevel("ERROR")

//Reading training and testing data

val df_train = spark.read.option("header", "true").schema(trainSchema)
  .csv( path = "/Users/sameekshagurrala/Desktop/SparkAssignment_2/src/main/resources/train.csv")
val df_test = spark.read.option("header", "true")
  .schema(testSchema).csv( path = "/Users/sameekshagurrala/Desktop/SparkAssignment_2/src/main/resources/test.csv")
//Creating table views for training and testing
df_train.createOrReplaceTempView( viewName = "df_train")
df_test.createOrReplaceTempView( viewName = "df_test")

// ---------------------- EAD ---------------------------
//Describing training data to show Exploratory Data Analysis (EDA) statistics
df_train.describe( cols = "Age","SibSp","Parch","Fare").show()


//Describing testing data to show Exploratory Data Analysis (EDA) statistics
spark.sql( sqlText = "select Survived,count(*) from df_train group by Survived").show()
spark.sql( sqlText = "select Sex, Survived, count(*) from df_train group by Sex,Survived").show()
spark.sql( sqlText = "select Pclass, Survived, count(*) from df_train group by Pclass,Survived").show()
```

Output :

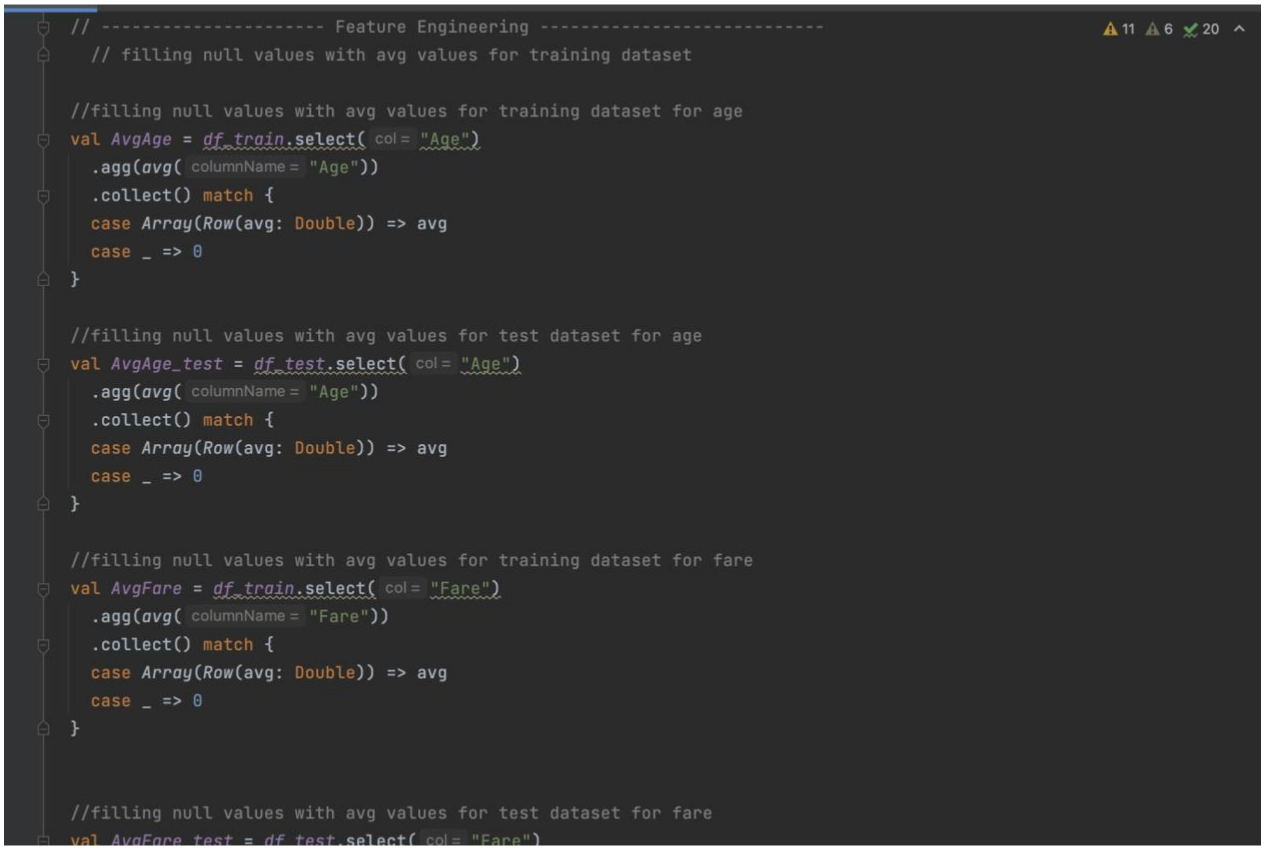


```
+-------+------------------+------------------+-------------------+-----------------+
|summary|               Age|             SibSp|              Parch|             Fare|
+-------+------------------+------------------+-------------------+-----------------+
|  count|               714|               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|
+-------+------------------+------------------+-------------------+-----------------+

+--------+--------+
|Survived|count(1)|
+--------+--------+
|       1|     342|
|       0|     549|
+--------+--------+

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

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

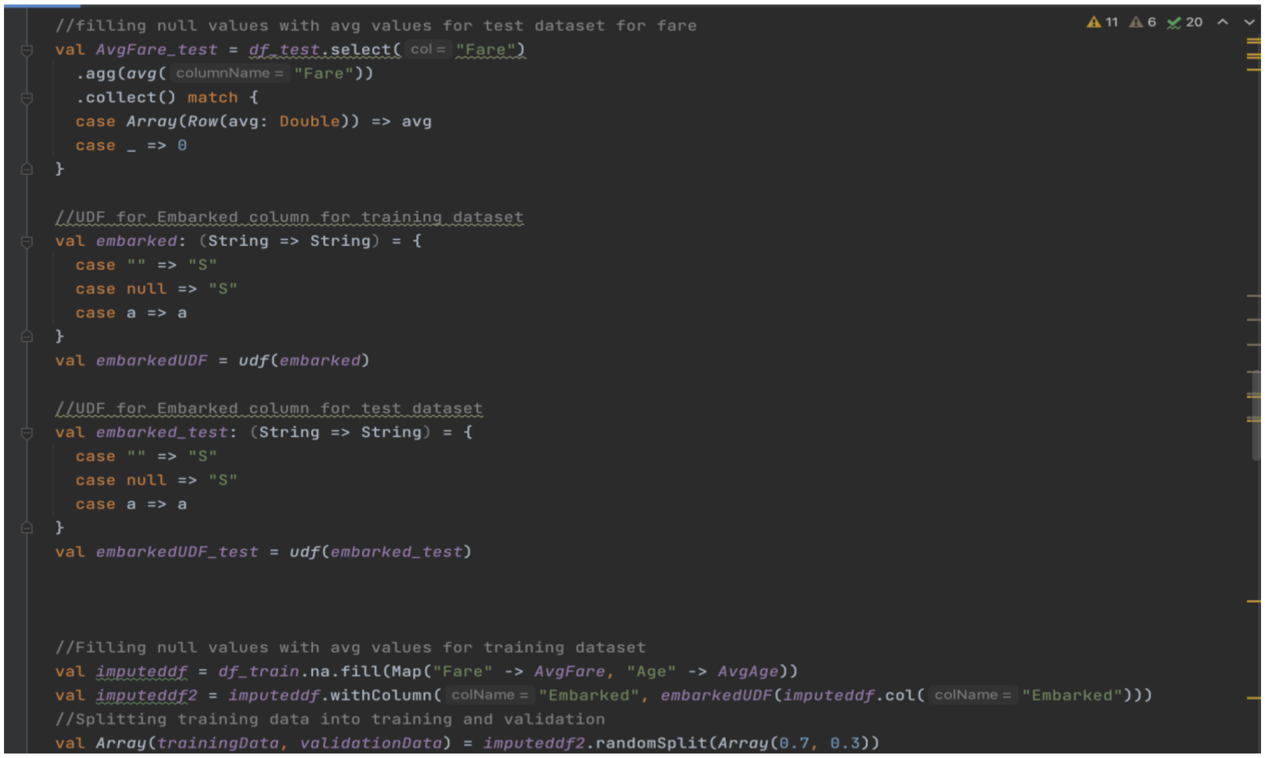
Code: Feature engineering

Created new attributes that may be derived from the existing attributes.

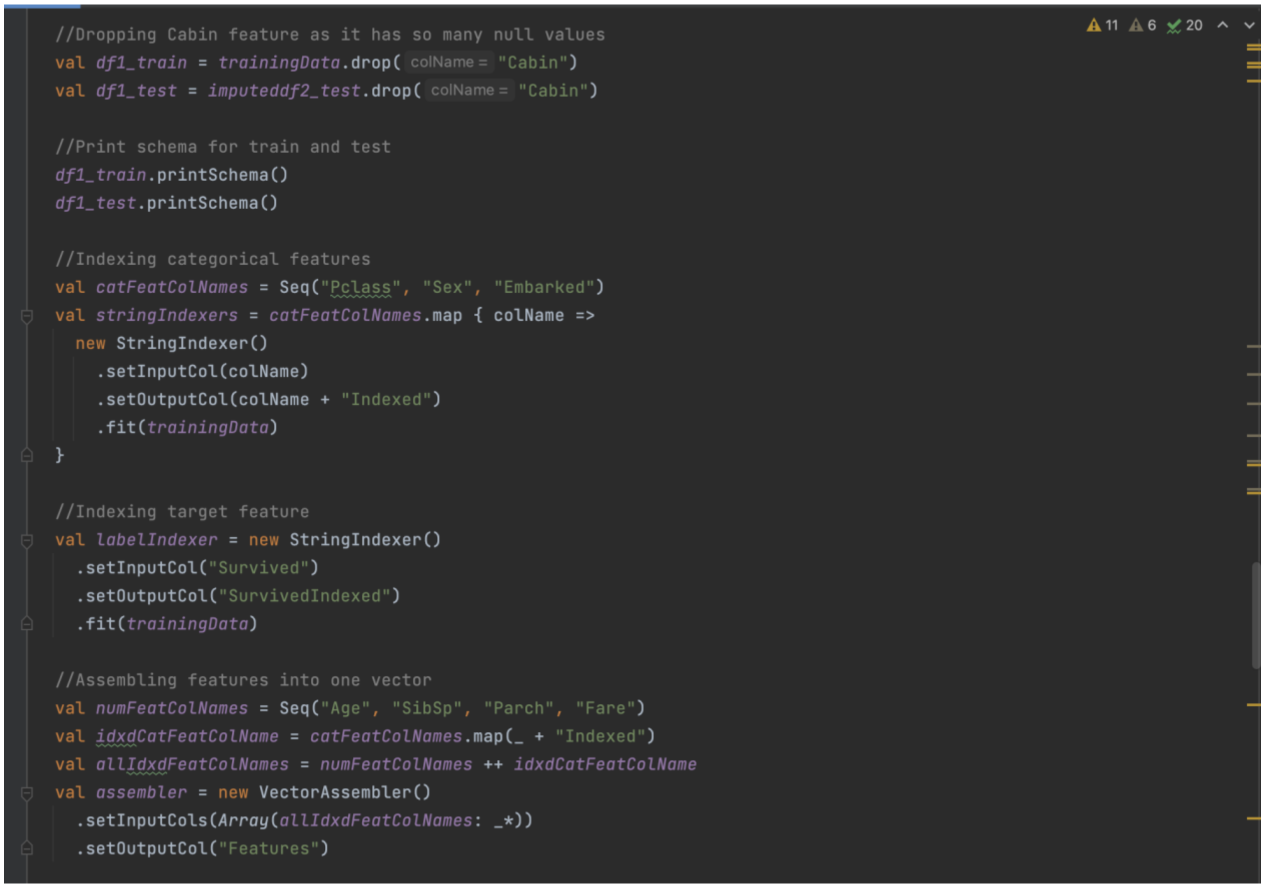
This included in **removing cabin column** in the dataset since it contains lot of null values.



```
// --------------------- Feature Engineering ---------------------------
  // filling null values with avg values for training dataset

//filling null values with avg values for training dataset for age
val AvgAge = df_train.select( col = "Age")
  .agg(avg( columnName = "Age"))
  .collect() match {
  case Array(Row(avg: Double)) => avg
  case _ => 0
}

//filling null values with avg values for test dataset for age
val AvgAge_test = df_test.select( col = "Age")
  .agg(avg( columnName = "Age"))
  .collect() match {
  case Array(Row(avg: Double)) => avg
  case _ => 0
}

//filling null values with avg values for training dataset for fare
val AvgFare = df_train.select( col = "Fare")
  .agg(avg( columnName = "Fare"))
  .collect() match {
  case Array(Row(avg: Double)) => avg
  case _ => 0
}


//filling null values with avg values for test dataset for fare
val AvgFare_test = df_test.select( col = "Fare")
```



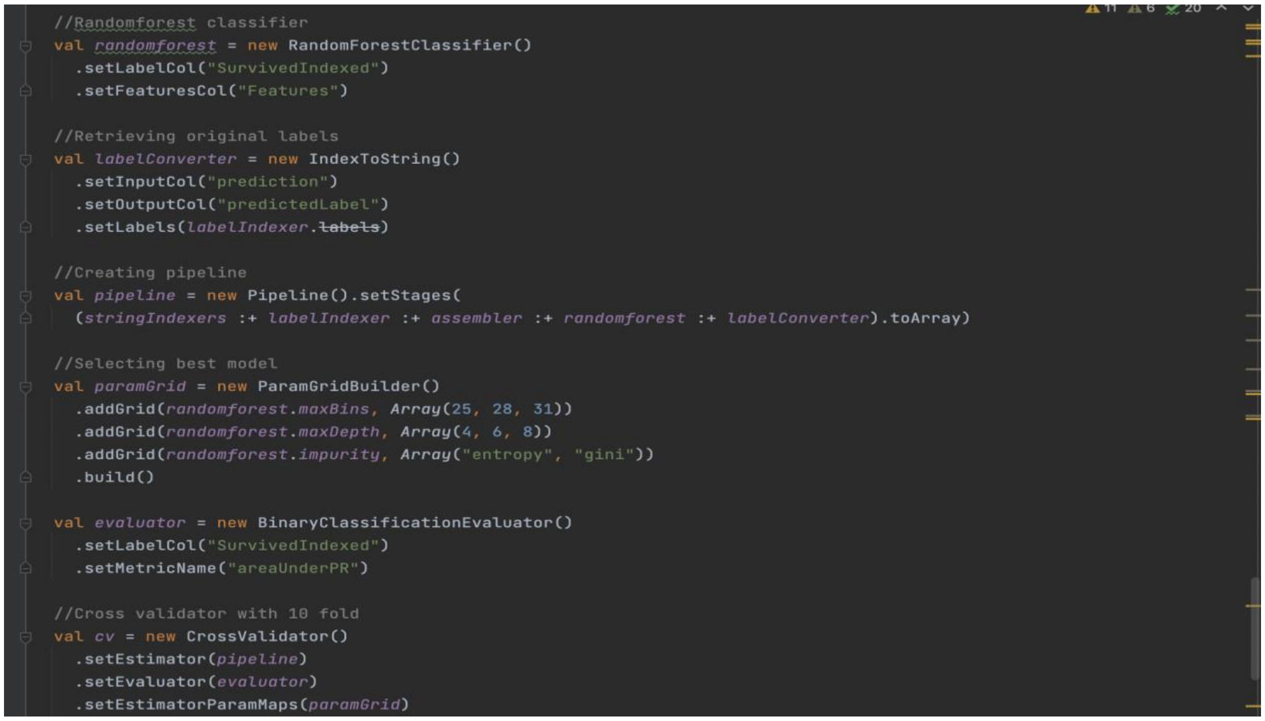
```
//filling null values with avg values for test dataset for fare
val AvgFare_test = df_test.select( col = "Fare")
  .agg(avg( columnName = "Fare"))
  .collect() match {
  case Array(Row(avg: Double)) => avg
  case _ => 0
}

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

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



//Filling null values with avg values for training dataset
val imputeddf = df_train.na.fill(Map("Fare" -> AvgFare, "Age" -> AvgAge))
val imputeddf2 = imputeddf.withColumn( colName = "Embarked", embarkedUDF(imputeddf.col( colName = "Embarked")))
//Splitting training data into training and validation
val Array(trainingData, validationData) = imputeddf2.randomSplit(Array(0.7, 0.3))
```



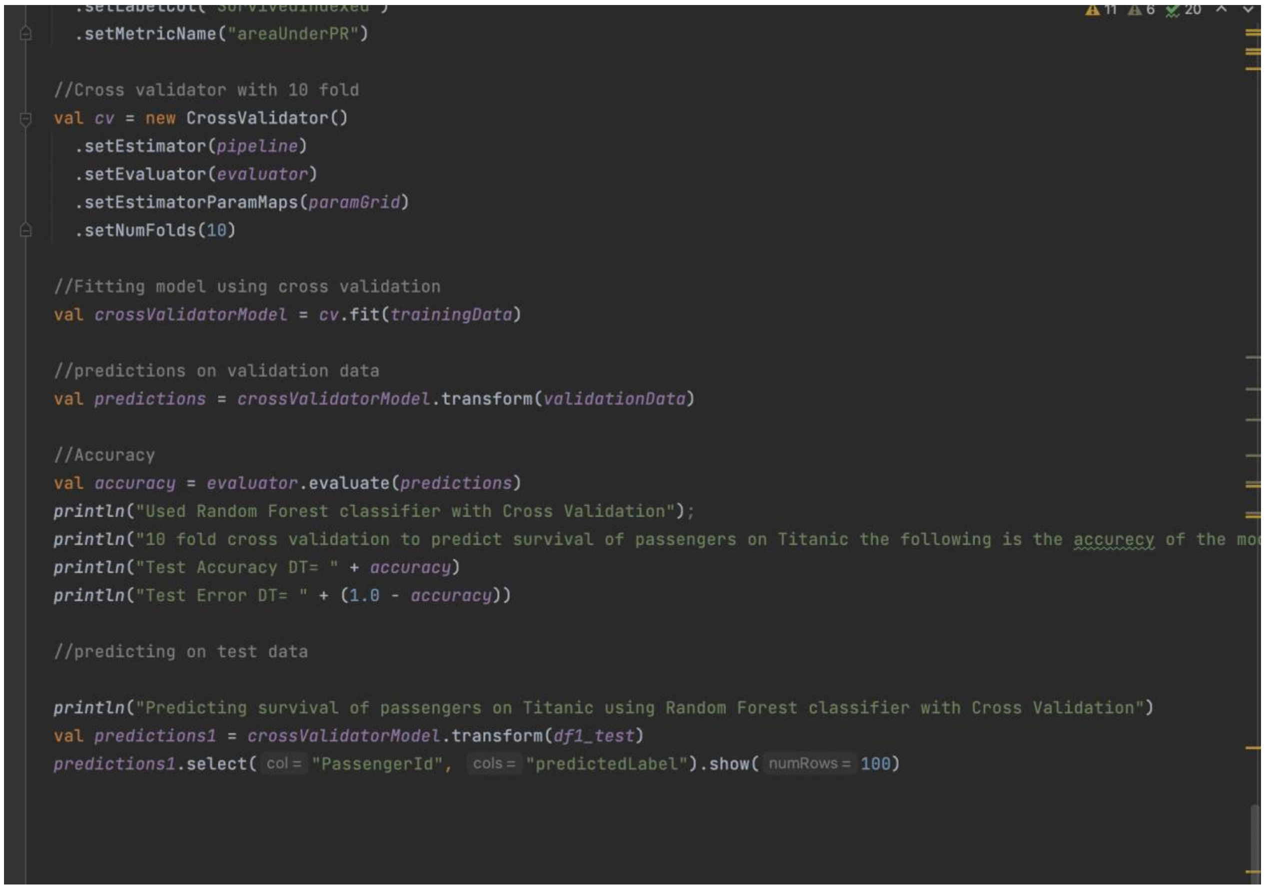
```
//Dropping Cabin feature as it has so many null values
val df1_train = trainingData.drop( colName = "Cabin")
val df1_test = imputeddf2_test.drop( colName = "Cabin")

//Print schema for train and test
df1_train.printSchema()
df1_test.printSchema()

//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")
```

Prediction – Used the train.csv to train by using Random Forest classifier with cross validation & tested it on the test.csv. Predicted if the records in test.csv survived or not. Note(1 = Survived, 0 = Dead)

Accuracy of 83% is obtained on the model using Random Forest algorithm. 10-fold cross validation to predict the survival of passengers on titanic dataset.

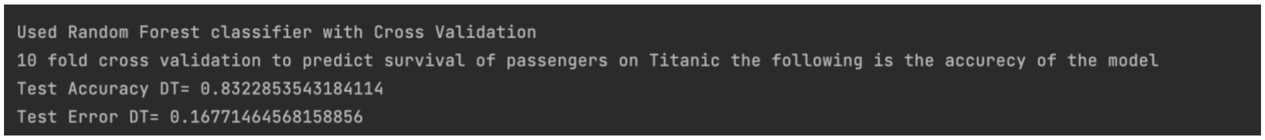


```
//Randomforest classifier
val randomforest = new RandomForestClassifier()
  .setLabelCol("SurvivedIndexed")
  .setFeaturesCol("Features")

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

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

//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)
```

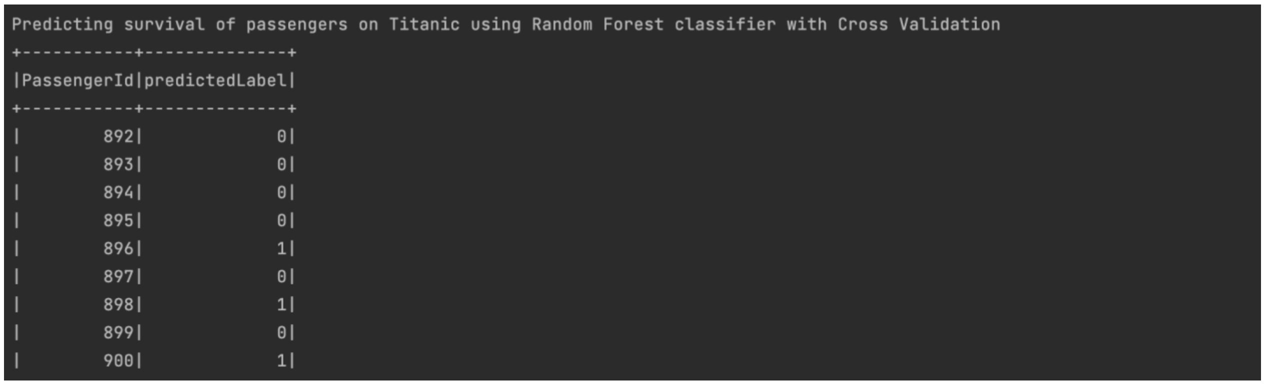


```
  .setLabelCol("SurvivedIndexed")
  .setMetricName("areaUnderPR")

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

//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("Used Random Forest classifier with Cross Validation");
println("10 fold cross validation to predict survival of passengers on Titanic the following is the accurecy of the mo
println("Test Accuracy DT= " + accuracy)
println("Test Error DT= " + (1.0 - accuracy))

//predicting on test data

println("Predicting survival of passengers on Titanic using Random Forest classifier with Cross Validation")
val predictions1 = crossValidatorModel.transform(df1_test)
predictions1.select( col = "PassengerId",  cols = "predictedLabel").show( numRows = 100)
```

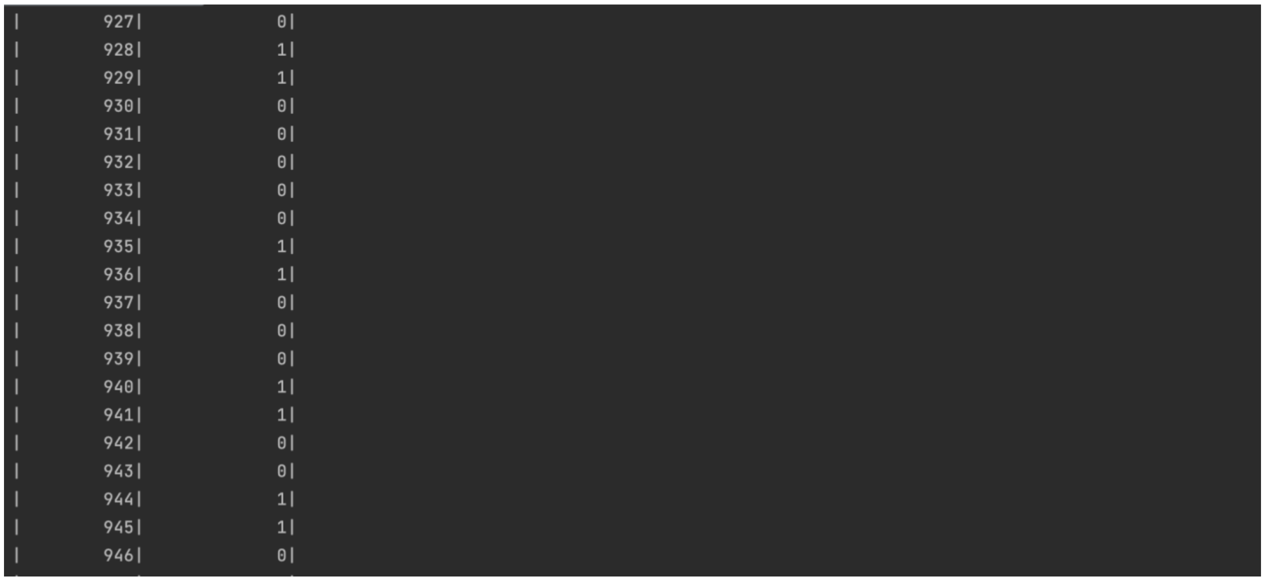
OUTPUT:



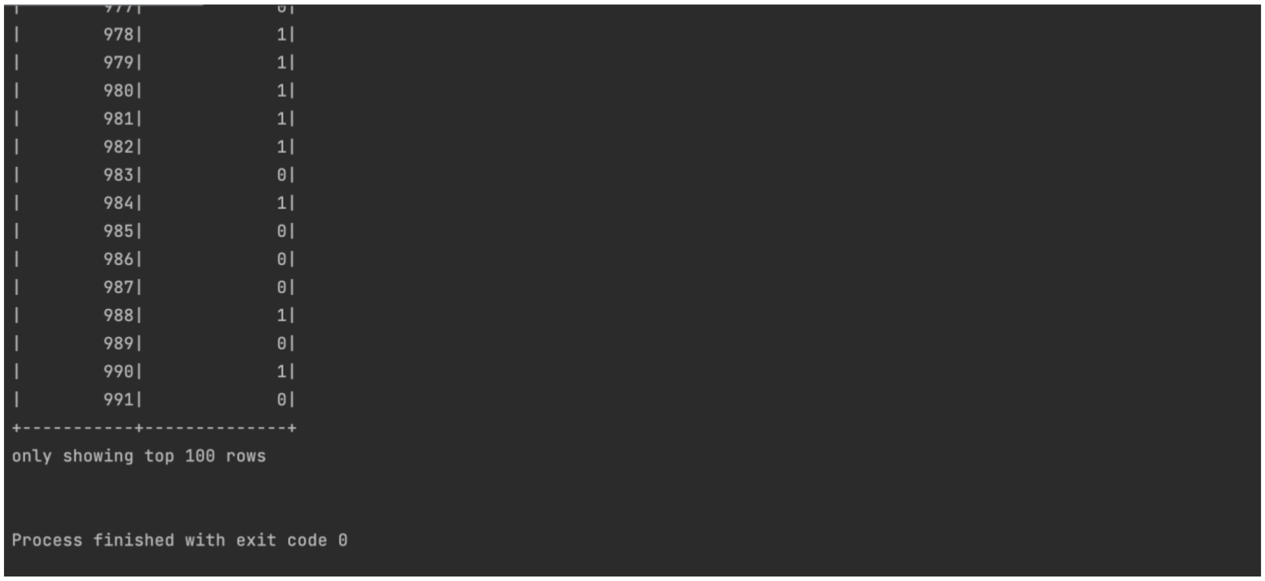
```
Used Random Forest classifier with Cross Validation
10 fold cross validation to predict survival of passengers on Titanic the following is the accurecy of the model
Test Accuracy DT= 0.8322853543184114
Test Error DT= 0.16771464568158856
```



```
Predicting survival of passengers on Titanic using Random Forest classifier with Cross Validation
+-----------+-------------+
|PassengerId|predictedLabel|
+-----------+-------------+
|        892|            0|
|        893|            0|
|        894|            0|
|        895|            0|
|        896|            1|
|        897|            0|
|        898|            1|
|        899|            0|
|        900|            1|
```



```
|        927|            0|
|        928|            1|
|        929|            1|
|        930|            0|
|        931|            0|
|        932|            0|
|        933|            0|
|        934|            0|
|        935|            1|
|        936|            1|
|        937|            0|
|        938|            0|
|        939|            0|
|        940|            1|
|        941|            1|
|        942|            0|
|        943|            0|
|        944|            1|
|        945|            1|
|        946|            0|
```



```
|        978|            1|
|        979|            1|
|        980|            1|
|        981|            1|
|        982|            1|
|        983|            0|
|        984|            1|
|        985|            0|
|        986|            0|
|        987|            0|
|        988|            1|
|        989|            0|
|        990|            1|
|        991|            0|
+-----------+-------------+
only showing top 100 rows


Process finished with exit code 0
```

Link : https://github.com/Sameeksha-11/Sparkassignment-2