Anirudha Joshi – SEC01 (NUID 002991365)

Big Data System Engineering with Scala
Spring 2023
Spark Assignment No. 2 (Titanic Prediction)

GitHub - https://github.com/anirudhajoshi2808/Anirudha_Joshi_CSYE7200

- Questions:

Url: https://www.kaggle.com/competitions/titanic/dataLinks to an external site.

For this assignment you will use the training and testing datasets.(train.csv & test.csv)

You are to load the dataset using Spark and perform the operations below:

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)

Please note: Do not include the test.csv while training the model. Also do not use the 'Survived' column during training. Doing these would defeat the purpose of the entire model.

- Code

localhost:8888/notebooks/Documents/NEU_Work/NEU_Spring2023_Work/Big_Data_Scala/Spark_Assignment2/SparkAssignment2.ipynb

jupyter SparkAssignment2 Last Checkpoint: 6 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Spark Assignment 2

Create Spark Session and Spark Context

As first step create a spark session & spark context

In [1]:
```
import org.apache.spark.sql.SparkSession

val spark = SparkSession
  .builder()
  .appName("Spark Assignment2")
  .getOrCreate()
```

```
Intitializing Scala interpreter ...

Spark Web UI available at http://10.110.104.100:4040
SparkContext available as 'sc' (version = 3.3.2, master = local[*], app id = local-1680109263066)
SparkSession available as 'spark'

23/03/29 13:01:06 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effec
t.
```

Out[1]:
```
import org.apache.spark.sql.SparkSession
spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@5f30e7a1
```

In [2]:
```
val sc = spark.sparkContext
```

Out[2]:
```
sc: org.apache.spark.SparkContext = org.apache.spark.SparkContext@46e24902
```

Data Ingestion

In [3]:
```
val testDf = spark.read.option("header", "true").csv("test.csv")
testDf.show()
```

```
+-----------+------+--------------------+------+---+-----+-----+----------------+-------+-----+--------+
|PassengerId|Pclass|                Name|   Sex|Age|SibSp|Parch|          Ticket|   Fare|Cabin|Embarked|
+-----------+------+--------------------+------+---+-----+-----+----------------+-------+-----+--------+
|        892|     3|    Kelly, Mr. James|  male|34.5|    0|    0|          330911| 7.8292| null|       Q|
|        893|     3|Wilkes, Mrs. Jame...|female| 47|    1|    0|          363272|      7| null|       S|
|        894|     2|Myles, Mr. Thomas...|  male| 62|    0|    0|          240276| 9.6875| null|       Q|
|        895|     3|    Wirz, Mr. Albert|  male| 27|    0|    0|          315154| 8.6625| null|       S|
|        896|     3|Hirvonen, Mrs. Al...|female| 22|    1|    1|         3101298|12.2875| null|       S|
|        897|     3|Svensson, Mr. Joh...|  male| 14|    0|    0|            7538|  9.225| null|       S|
|        898|     3|Connolly, Miss. Kate|female| 30|    0|    0|          330972| 7.6292| null|       Q|
|        899|     2|Caldwell, Mr. Alb...|  male| 26|    1|    1|          248738|     29| null|       S|
```

localhost:8888/notebooks/Documents/NEU_Work/NEU_Spring2023_Work/Big_Data_Scala/Spark_Assignment2/SparkAssignment2.ipynb

jupyter SparkAssignment2 Last Checkpoint: 6 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Out[3]:
```
testDf: org.apache.spark.sql.DataFrame = [PassengerId: string, Pclass: string ... 9 more fields]
```

In [4]:
```
val trainDf = spark.read.option("header", "true").csv("train.csv")
trainDf.show()
```

```
+-----------+--------+------+--------------------+------+----+-----+-----+----------------+-------+-----+--------+
|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|
|         11|       1|     3|Sandstrom, Miss. ...|female|   4|    1|    1|         PP 9549|   16.7|   G6|       S|
|         12|       1|     1|Bonnell, Miss. El...|female|  58|    0|    0|          113783|  26.55| C103|       S|
|         13|       0|     3|Saundercock, Mr. ...|  male|  20|    0|    0|       A/5. 2151|   8.05| null|       S|
|         14|       0|     3|Andersson, Mr. An...|  male|  39|    1|    5|          347082| 31.275| null|       S|
|         15|       0|     3|Vestrom, Miss. Hu...|female|  14|    0|    0|          350406| 7.8542| null|       S|
|         16|       1|     2|Hewlett, Mrs. (Ma...|female|  55|    0|    0|          248706|     16| null|       S|
|         17|       0|     3|Rice, Master. Eugene|  male|   2|    4|    1|          382652| 29.125| null|       Q|
|         18|       1|     2|Williams, Mr. Cha...|  male|null|    0|    0|          244373|     13| null|       S|
|         19|       0|     3|Vander Planke, Mr...|female|  31|    1|    0|          345763|     18| null|       S|
|         20|       1|     3|Masselmani, Mrs. ...|female|null|    0|    0|            2649|  7.225| null|       C|
+-----------+--------+------+--------------------+------+----+-----+-----+----------------+-------+-----+--------+
only showing top 20 rows
```

Out[4]:
```
trainDf: org.apache.spark.sql.DataFrame = [PassengerId: string, Survived: string ... 10 more fields]
```

Data Cleaning and EDA

In [107]:
```
val trainDfSelected = trainDf.select("Survived", "Pclass", "Sex", "Age", "Fare", "Embarked")
```

Out[107]:
```
trainDfSelected: org.apache.spark.sql.DataFrame = [Survived: string, Pclass: string ... 4 more fields]
```

In [80]:
```
val train_df_clean = trainDfSelected.na.drop()
train_df_clean.describe().show()
```

```
+-------+-------------------+-------------------+------+------------------+-------------------+--------+
|summary|           Survived|             Pclass|   Sex|               Age|               Fare|Embarked|
+-------+-------------------+-------------------+------+------------------+-------------------+--------+
```

Fixing DataTypes of columns

In [81]:
```
import org.apache.spark.sql.functions.col

// Assuming df is your DataFrame and you want to change the data types of columns "age" and "salary"
val train_df_clean_final = train_df_clean
  .withColumn("Age", col("Age").cast("Double"))
  .withColumn("Fare", col("Fare").cast("Double"))
  .withColumn("Pclass", col("Pclass").cast("Integer"))
  .withColumn("Survived", col("Survived").cast("Integer"))
```

Out[81]:
```
import org.apache.spark.sql.functions.col
train_df_clean_final: org.apache.spark.sql.DataFrame = [Survived: int, Pclass: int ... 4 more fields]
```

In [82]:
```
train_df_clean_final.printSchema()
```

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

Blank values filled with mean so that prediction is more accurate

In [83]:
```
val ageMeantrain = train_df_clean_final.agg(avg("Age")).first()(0).asInstanceOf[Double]
val fareMeantrain = train_df_clean_final.agg(avg("Fare")).first()(0).asInstanceOf[Double]
val trainDFML = train_df_clean_final.na.fill(ageMeantrain, Seq("Age")).na.fill(fareMeantrain, Seq("Fare"))
```

Out[83]:
```
ageMeantrain: Double = 29.64209269662921
fareMeantrain: Double = 34.56725140449432
trainDFML: org.apache.spark.sql.DataFrame = [Survived: int, Pclass: int ... 4 more fields]
```

In [84]:
```
trainDFML.describe("Fare").show()
```

```
+-------+------------------+
|summary|              Fare|
+-------+------------------+
|  count|               712|
```

localhost:8888/notebooks/Documents/NEU_Work/NEU_Spring2023_Work/Big_Data_Scala/Spark_Assignment2/SparkAssignment2.ipynb

jupyter SparkAssignment2 Last Checkpoint: 6 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | spylon-kernel

In [85]:
```
val testDfSelected = testDf.select("PassengerId", "Pclass", "Sex", "Age", "Fare", "Embarked")
```

Out[85]:
```
testDfSelected: org.apache.spark.sql.DataFrame = [PassengerId: string, Pclass: string ... 4 more fields]
```

In [86]:
```
val test_df_clean = testDfSelected.na.drop()
test_df_clean.describe().show()
```

```
+-------+-----------------+------------------+------+------------------+-----------------+--------+
|summary|      PassengerId|            Pclass|   Sex|               Age|             Fare|Embarked|
+-------+-----------------+------------------+------+------------------+-----------------+--------+
|  count|              331|               331|   331|               331|              331|     331|
|   mean|1100.232628398916|2.1419939577039275|  null|30.181268882175228|40.98208731117823|    null|
| stddev|122.91018015895622|0.8462507042885307|  null|14.104572594801617|61.22855822554924|    null|
|    min|             1001|                 1|female|              0.17|                0|       C|
|    max|              998|                 3|  male|                 9|             93.5|       S|
+-------+-----------------+------------------+------+------------------+-----------------+--------+
```

Out[86]:
```
test_df_clean: org.apache.spark.sql.DataFrame = [PassengerId: string, Pclass: string ... 4 more fields]
```

Fixing DataTypes of columns

In [87]:
```
import org.apache.spark.sql.functions.col

// Assuming df is your DataFrame and you want to change the data types of columns "age" and "salary"
val test_df_clean_final = test_df_clean
  .withColumn("Age", col("Age").cast("Double"))
  .withColumn("Fare", col("Fare").cast("Double"))
  .withColumn("Pclass", col("Pclass").cast("Integer"))
  .withColumn("PassengerId", col("PassengerId").cast("Integer"))
```

Out[87]:
```
import org.apache.spark.sql.functions.col
test_df_clean_final: org.apache.spark.sql.DataFrame = [PassengerId: int, Pclass: int ... 4 more fields]
```

In [88]:
```
test_df_clean_final.printSchema()
```

```
root
 |-- PassengerId: integer (nullable = true)
 |-- Pclass: integer (nullable = true)
 |-- Sex: string (nullable = true)
 |-- Age: double (nullable = true)
 |-- Fare: double (nullable = true)
 |-- Embarked: string (nullable = true)
```

Blank values filled with mean so that prediction is more accurate

In [89]:
```
val ageMean = test_df_clean_final.agg(avg("Age")).first()(0).asInstanceOf[Double]
val fareMean = test_df_clean_final.agg(avg("Fare")).first()(0).asInstanceOf[Double]
val testDFML = test_df_clean_final.na.fill(ageMean, Seq("Age")).na.fill(fareMean, Seq("Fare"))
```

Out[89]:
```
ageMean: Double = 30.181268882175228
fareMean: Double = 40.98208731117823
testDFML: org.apache.spark.sql.DataFrame = [PassengerId: int, Pclass: int ... 4 more fields]
```

In [90]:
```
testDFML.describe().show()
```

```
+-------+------------------+------------------+------+-----------------+-----------------+--------+
|summary|       PassengerId|            Pclass|   Sex|              Age|             Fare|Embarked|
+-------+------------------+------------------+------+-----------------+-----------------+--------+
|  count|               331|               331|   331|              331|              331|     331|
|   mean|1100.2326283987916|2.1419939577039275|  null|30.181268882175228|40.98208731117823|    null|
| stddev|122.91018015895622|0.8462507042885307|  null|14.104572594801617|61.22855822554924|    null|
|    min|               892|                 1|female|             0.17|              0.0|       C|
|    max|              1307|                 3|  male|             76.0|         512.3292|       S|
+-------+------------------+------------------+------+-----------------+-----------------+--------+
```

In [91]:
```
testDFML.describe("Fare").show()
```

```
+-------+-----------------+
|summary|             Fare|
+-------+-----------------+
|  count|              331|
|   mean|40.98208731117823|
| stddev|61.22855822554924|
|    min|              0.0|
|    max|         512.3292|
+-------+-----------------+
```

We can see the mean Fare overall was 40.98 out of 331 people

Let's see if there's a difference between Male and Female Fares

In [103]:
```
val testMaleDf = testDFML.filter(testDf("Sex") === "male")
```

Out[103]:
```
testMaleDf: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [PassengerId: int, Pclass: int ... 4 more field
```

localhost:8888/notebooks/Documents/NEU_Work/NEU_Spring2023_Work/Big_Data_Scala/Spark_Assignment2/SparkAssignment2.ipynb

YouTube | My Drive - Google... | NEU | Slack Channels N... | Data Engg | DSA | WhatsApp | DatabricksPlatform | GitHub - jwasham... | Data Engineering... | Showcase your da...

jupyter SparkAssignment2 Last Checkpoint: 6 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted | spylon-kernel

Run Markdown

We can see the mean Fare overall was 40.98 out of 331 people

Let's see if there's a difference between Male and Female Fares

In [103]:
```
val testMaleDf = testDFML.filter(testDf("Sex") === "male")
```

Out[103]:
```
testMaleDf: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [PassengerId: int, Pclass: int ... 4 more field
s]
```

In [104]:
```
val testFemaleDf = testDFML.filter(testDf("Sex") === "female")
```

Out[104]:
```
testFemaleDf: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [PassengerId: int, Pclass: int ... 4 more fiel
ds]
```

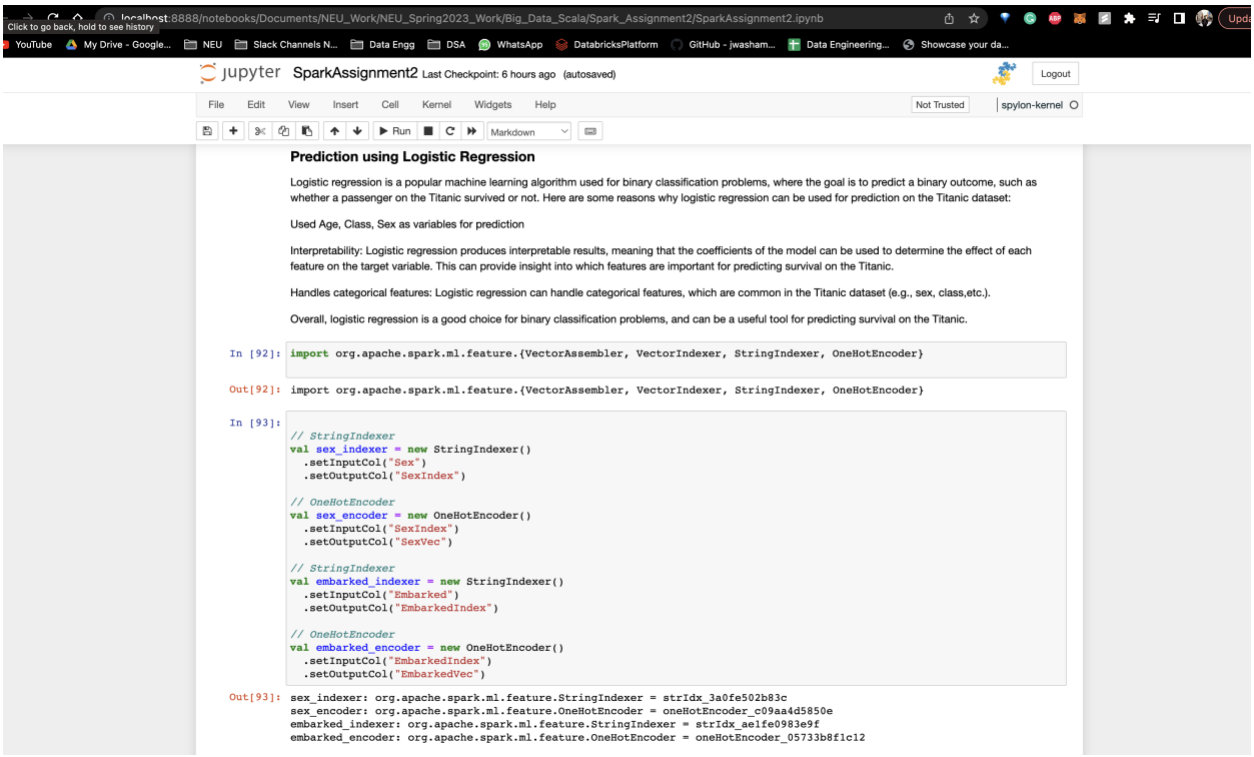
In [105]:
```
testMaleDf.describe("Fare").show()
```

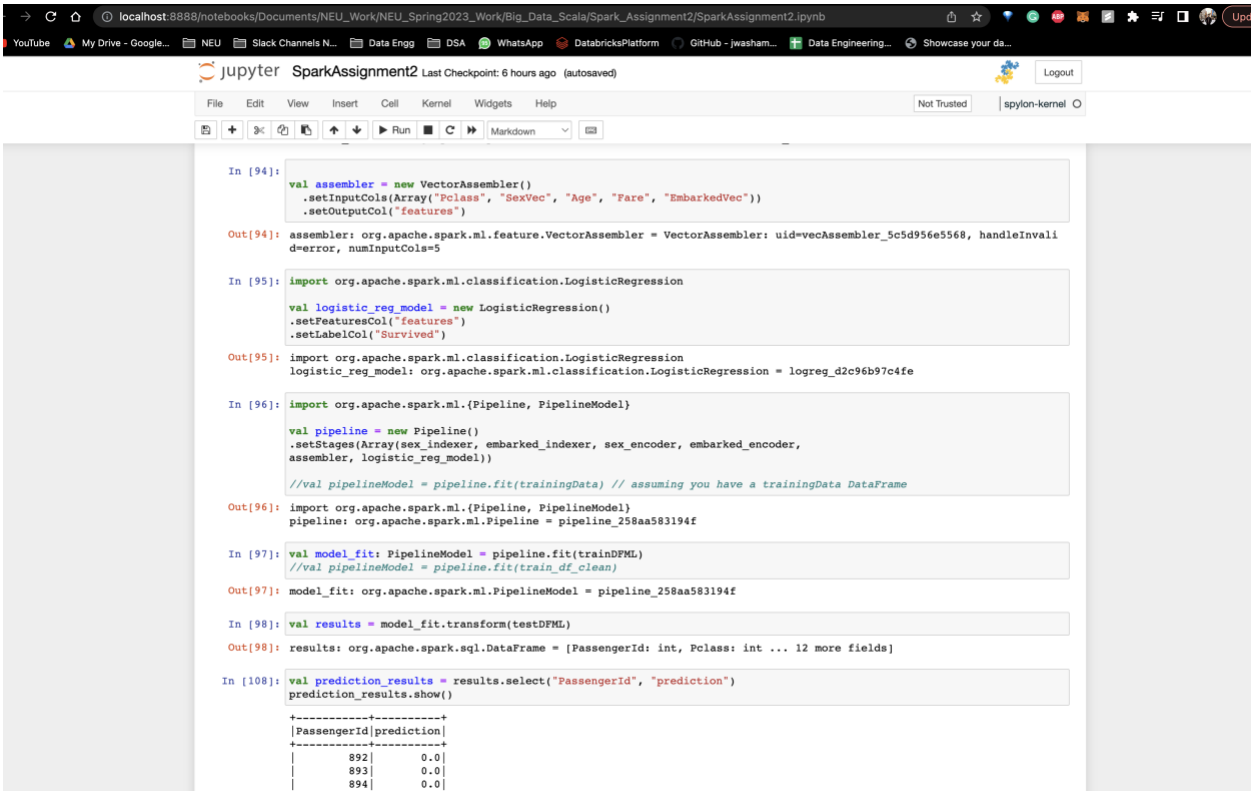
```
+-------+-----------------+
|summary|             Fare|
+-------+-----------------+
|  count|              204|
|   mean|31.688071078431378|
| stddev|45.511592302124235|
|    min|              0.0|
|    max|          262.375|
+-------+-----------------+
```

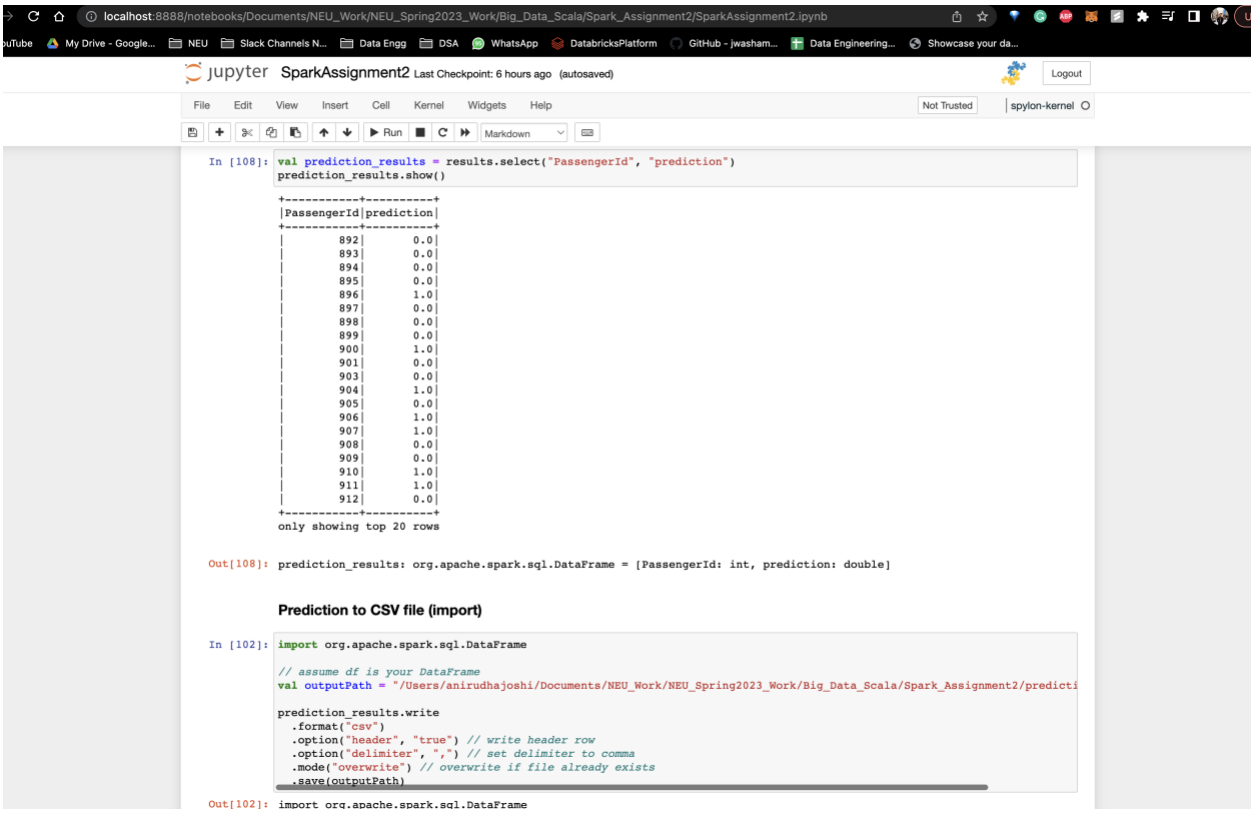
In [106]:
```
testFemaleDf.describe("Fare").show()
```

```
+-------+-----------------+
|summary|             Fare|
+-------+-----------------+
|  count|              127|
|   mean|55.911058267716 52|
| stddev|78.21154608341406|
|    min|             6.95|
|    max|         512.3292|
+-------+-----------------+
```

As we can see there are 204 males and the Mean Fare for "Male" is 31.68 and there are 152 females and the mean for "Female" is 55.91. We can say that Females paid more than males on an average


localhost:8888/notebooks/Documents/NEU_Work/NEU_Spring2023_Work/Big_Data_Scala/Spark_Assignment2/SparkAssignment2.ipynb
jupyter SparkAssignment2 Last Checkpoint: 6 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
spylon-kernel
Run
Markdown
Prediction using Logistic Regression
Logistic regression is a popular machine learning algorithm used for binary classification problems, where the goal is to predict a binary outcome, such as whether a passenger on the Titanic survived or not. Here are some reasons why logistic regression can be used for prediction on the Titanic dataset:
Used Age, Class, Sex as variables for prediction
Interpretability: Logistic regression produces interpretable results, meaning that the coefficients of the model can be used to determine the effect of each feature on the target variable. This can provide insight into which features are important for predicting survival on the Titanic.
Handles categorical features: Logistic regression can handle categorical features, which are common in the Titanic dataset (e.g., sex, class,etc.).
Overall, logistic regression is a good choice for binary classification problems, and can be a useful tool for predicting survival on the Titanic.
In [92]:
import org.apache.spark.ml.feature.{VectorAssembler, VectorIndexer, StringIndexer, OneHotEncoder}
Out[92]:
import org.apache.spark.ml.feature.{VectorAssembler, VectorIndexer, StringIndexer, OneHotEncoder}
In [93]:
// StringIndexer
val sex_indexer = new StringIndexer()
.setInputCol("Sex")
.setOutputCol("SexIndex")
// OneHotEncoder
val sex_encoder = new OneHotEncoder()
.setInputCol("SexIndex")
.setOutputCol("SexVec")
// StringIndexer
val embarked_indexer = new StringIndexer()
.setInputCol("Embarked")
.setOutputCol("EmbarkedIndex")
// OneHotEncoder
val embarked_encoder = new OneHotEncoder()
.setInputCol("EmbarkedIndex")
.setOutputCol("EmbarkedVec")
Out[93]:
sex_indexer: org.apache.spark.ml.feature.StringIndexer = strIdx_3a0fe502b83c
sex_encoder: org.apache.spark.ml.feature.OneHotEncoder = oneHotEncoder_c09aa4d5850e
embarked_indexer: org.apache.spark.ml.feature.StringIndexer = strIdx_ae1fe0983e9f
embarked_encoder: org.apache.spark.ml.feature.OneHotEncoder = oneHotEncoder_05733b8f1c12


localhost:8888/notebooks/Documents/NEU_Work/NEU_Spring2023_Work/Big_Data_Scala/Spark_Assignment2/SparkAssignment2.ipynb
jupyter SparkAssignment2 Last Checkpoint: 6 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
spylon-kernel
Run
Markdown
In [94]:
val assembler = new VectorAssembler()
.setInputCols(Array("Pclass", "SexVec", "Age", "Fare", "EmbarkedVec"))
.setOutputCol("features")
Out[94]:
assembler: org.apache.spark.ml.feature.VectorAssembler = VectorAssembler: uid=vecAssembler_5c5d956e5568, handleInvalid=error, numInputCols=5
In [95]:
import org.apache.spark.ml.classification.LogisticRegression
val logistic_reg_model = new LogisticRegression()
.setFeaturesCol("features")
.setLabelCol("Survived")
Out[95]:
import org.apache.spark.ml.classification.LogisticRegression
logistic_reg_model: org.apache.spark.ml.classification.LogisticRegression = logreg_d2c96b97c4fe
In [96]:
import org.apache.spark.ml.{Pipeline, PipelineModel}
val pipeline = new Pipeline()
.setStages(Array(sex_indexer, embarked_indexer, sex_encoder, embarked_encoder,
assembler, logistic_reg_model))
//val pipelineModel = pipeline.fit(trainingData) // assuming you have a trainingData DataFrame
Out[96]:
import org.apache.spark.ml.{Pipeline, PipelineModel}
pipeline: org.apache.spark.ml.Pipeline = pipeline_258aa583194f
In [97]:
val model_fit: PipelineModel = pipeline.fit(trainDFML)
//val pipelineModel = pipeline.fit(train_df_clean)
Out[97]:
model_fit: org.apache.spark.ml.PipelineModel = pipeline_258aa583194f
In [98]:
val results = model_fit.transform(testDFML)
Out[98]:
results: org.apache.spark.sql.DataFrame = [PassengerId: int, Pclass: int ... 12 more fields]
In [108]:
val prediction_results = results.select("PassengerId", "prediction")
prediction_results.show()
+-----------+----------+
|PassengerId|prediction|
+-----------+----------+
892	0.0
893	0.0
894	0.0


localhost:8888/notebooks/Documents/NEU_Work/NEU_Spring2023_Work/Big_Data_Scala/Spark_Assignment2/SparkAssignment2.ipynb
jupyter SparkAssignment2 Last Checkpoint: 6 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
spylon-kernel
Run
Markdown
In [108]:
val prediction_results = results.select("PassengerId", "prediction")
prediction_results.show()
+-----------+----------+
|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
912	0.0
+-----------+----------+
only showing top 20 rows
Out[108]:
prediction_results: org.apache.spark.sql.DataFrame = [PassengerId: int, prediction: double]
Prediction to CSV file (import)
In [102]:
import org.apache.spark.sql.DataFrame
// assume df is your DataFrame
val outputPath = "/Users/anirudhajoshi/Documents/NEU_Work/NEU_Spring2023_Work/Big_Data_Scala/Spark_Assignment2/predicti
prediction_results.write
.format("csv")
.option("header", "true") // write header row
.option("delimiter", ",") // set delimiter to comma
.mode("overwrite") // overwrite if file already exists
.save(outputPath)
Out[102]:
import org.apache.spark.sql.DataFrame