

Komal Suryan : NUID 002747707

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



**TASKS:**

Url: <https://www.kaggle.com/competitions/titanic/data>Links 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.

The classifier must have an accuracy of 70 % for 100 pts.

The submission must be on a new GitHub repository or data bricks public notebook along with the Assignment pdf. Please provide Screenshots of the results in the Assignment.pdf for submission.

Screenshots

```
import org.apache.spark.sql.SparkSession
```

```
val spark = SparkSession  
  .builder()  
  .appName("spark2")  
  .getOrCreate()
```

```
Initializing Scala interpreter ...  
Spark Web UI available at http://10.110.48.55:4040  
SparkContext available as 'sc' (version = 3.3.2, master = local[*], app id = local-1680803218993)  
SparkSession available as 'spark'  
23/04/06 13:47:01 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.  
import org.apache.spark.sql.SparkSession  
spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@5f30e7a1
```

```
val sc = spark.sparkContext
```

```
sc: org.apache.spark.SparkContext = org.apache.spark.SparkContext@46e24902
```

```
[ ] val test = spark.read.option("header", "true").csv("test.csv")  
val train = spark.read.option("header", "true").csv("train.csv")
```

```
test: org.apache.spark.sql.DataFrame = [PassengerId: string, Pclass: string ... 9 more fields]  
train: org.apache.spark.sql.DataFrame = [PassengerId: string, Survived: string ... 10 more fields]
```

## EDA

```
[ ] train.show(5)
```

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

## EDA

```
[ ] train.show(5)
```

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	Heikinen, 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

only showing top 5 rows

```
test.show(5)
```

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

only showing top 5 rows

```
[ ] import org.apache.spark.sql.functions.count
```

```
train.groupBy("Sex", "Survived")  
  .agg(count("Survived").alias("count"))
```



```
[ ] import org.apache.spark.sql.functions.count

train.groupBy("Sex", "Survived")
  .agg(count("Survived").alias("count"))
  .show()
```

Sex	Survived	count
female	0	81
female	1	233
male	1	109
male	0	468

```
import org.apache.spark.sql.functions.count
```

```
▶ train.groupBy("Pclass", "Survived")
  .agg(count("Survived").alias("count"))
  .show()
```

Pclass	Survived	count
1	1	136
2	0	97
3	1	119
3	0	372
2	1	87
1	0	80



```
[ ] train.groupBy("Pclass", "Sex")
  .agg(count("Sex").alias("count"))
  .show()
```

Pclass	Sex	count
2	female	76
2	male	108
3	male	347
1	male	122
1	female	94
3	female	144

## Feature Engineering

```
[ ] val test_new = test
  .withColumn("Age", col("Age").cast("Double"))
  .withColumn("Fare", col("Fare").cast("Double"))
  .withColumn("Pclass", col("Pclass").cast("Integer"))
  .withColumn("PassengerId", col("PassengerId").cast("Integer"))

test_new: org.apache.spark.sql.DataFrame = [PassengerId: int, Pclass: int ... 9 more fields]
```

```
[ ] val train_new = train
  .withColumn("Age", col("Age").cast("Double"))
  .withColumn("Fare", col("Fare").cast("Double"))
  .withColumn("Pclass", col("Pclass").cast("Integer"))
  .withColumn("Survived", col("Survived").cast("Integer"))
  .withColumn("PassengerId", col("PassengerId").cast("Integer"))
```

## • Prediction

```
[ ] val women = train.filter(col("Sex") === "female").select("Survived")
    val rate_women = women.agg(avg("Survived")).as[Double].head()
    println("% of women survived: " + rate_women)
```

```
% of women survived: 0.7420382165605095
women: org.apache.spark.sql.DataFrame = [Survived: string]
rate_women: Double = 0.7420382165605095
```

```
▶ val women = train.filter(col("Sex") === "male").select("Survived")
  val rate_men = women.agg(avg("Survived")).as[Double].head()
  println("% of men survived: " + rate_men)
```

```
▶ % of men survived: 0.18890814558058924
women: org.apache.spark.sql.DataFrame = [Survived: string]
rate_men: Double = 0.18890814558058924
```

```
[ ] import org.apache.spark.ml.Pipeline
    import org.apache.spark.ml.feature.{StringIndexer, OneHotEncoder, VectorAssembler}
    import org.apache.spark.ml.classification.LogisticRegression

    val sexIndexer = new StringIndexer()
      .setInputCol("Sex")
      .setOutputCol("SexIndex")

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

    val sexEncoder = new OneHotEncoder()
      .setInputCol("SexIndex")
```

```

import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.feature.{StringIndexer, OneHotEncoder, VectorAssembler}
import org.apache.spark.ml.classification.LogisticRegression

val sexIndexer = new StringIndexer()
  .setInputCol("Sex")
  .setOutputCol("SexIndex")

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

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

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

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

val lr = new LogisticRegression()
  .setLabelCol("Survived")
  .setFeaturesCol("features")
  .setMaxIter(10)

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

val pipelineModel = pipeline.fit(train_final)

val predictions = pipelineModel.transform(test_final)

```

```

[ ] 23/04/06 14:48:22 WARN InstanceBuilder$JavaBLAS: Failed to load implementation from:dev.ludovic.netlib.blas.VectorBLAS
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.feature.{StringIndexer, OneHotEncoder, VectorAssembler}
import org.apache.spark.ml.classification.LogisticRegression
sexIndexer: org.apache.spark.ml.feature.StringIndexer = strIdx_11e1809eeda4
embarkedIndexer: org.apache.spark.ml.feature.StringIndexer = strIdx_17316ab1fba1
sexEncoder: org.apache.spark.ml.feature.OneHotEncoder = oneHotEncoder_34b6c30464cd
embarkedEncoder: org.apache.spark.ml.feature.OneHotEncoder = oneHotEncoder_38baf25c479b
assembler: org.apache.spark.ml.feature.VectorAssembler = VectorAssembler: uid=vecAssembler_8e210e4ce38e, handleInvalid=error, numInputCols=3
lr: org.apache.spark.ml.classification.LogisticRegression = logreg_a35e622db1da
pipeline: org.apache.spark.ml.Pipeline = pipeline_7c06efe6a676
pipel...

```

```

predictions.show()

```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	SexIndex	EmbarkedIndex	SexVec
	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	null	Q	0.0	2.0	(1,[0],[1.0])
	893	3	Wilkes, Mrs. Jame...	female	47.0	1	0	363272	7.0	null	S	1.0	0.0	(1,[],[1])
	894	2	Myles, Mr. Thomas...	male	62.0	0	0	240276	9.6875	null	Q	0.0	2.0	(1,[0],[1.0])
	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	null	S	0.0	0.0	(1,[0],[1.0])
	896	3	Hirvonen, Mrs. Al...	female	22.0	1	1	3101298	12.2875	null	S	1.0	0.0	(1,[],[1])
	897	3	Svensson, Mr. Joh...	male	14.0	0	0	7538	9.225	null	S	0.0	0.0	(1,[0],[1.0])
	898	3	Connolly, Miss. Kate	female	30.0	0	0	330972	7.6292	null	Q	1.0	2.0	(1,[],[1])
	899	2	Caldwell, Mr. Alb...	male	26.0	1	1	248738	29.0	null	S	0.0	0.0	(1,[0],[1.0])
	900	3	Abraham, Mrs. Jos...	female	18.0	0	0	2657	7.2292	null	C	1.0	1.0	(1,[],[1])
	901	3	Davies, Mr. John ...	male	21.0	2	0	A/4 48871	24.15	null	S	0.0	0.0	(1,[0],[1.0])
	902	3	Ilieff, Mr. Yllo	male	30.272590361445783	0	0	349220	7.8958	null	S	0.0	0.0	(1,[0],[1.0])
	903	1	Jones, Mr. Charle...	male	46.0	0	0	694	26.0	null	S	0.0	0.0	(1,[0],[1.0])
	904	1	Snyder, Mrs. John...	female	23.0	1	0	21228	82.2667	B45	S	1.0	0.0	(1,[1],[1])
	905	2	Howard, Mr. Benjamin	male	63.0	1	0	24065	26.0	null	S	0.0	0	(1,[],[1])
	906	1	Chaffee, Mrs. Her...	female	47.0	1	0	W.E.P. 5734	61.175	E31	S	1.0	0	(1,[],[1])
	907	2	del Carlo, Mrs. S...	female	24.0	1	0	SC/PARIS 2167	27.7208	null	C	1.0	1	(1,[],[1])
	908	2	Keane, Mr. Daniel	male	35.0	0	0	233734	12.35	null	Q	0.0	2	(1,[],[1])
	909	3	Assaf, Mr. Gerios	male	21.0	0	0	2692	7.225	null	C	0.0	1	(1,[],[1])
	910	3	Ilmakangas, Miss....	female	27.0	1	0	STON/O2. 3101270	7.925	null	S	1.0	0.0	(1,[],[1])
	911	3	Assaf Khalil, Mr...	female	45.0	0	0	2696	7.225	null	C	1.0	1.0	(1,[],[1])