Spark ML

21.08.2021

Ronak Kumar
8740799

Introduction

Apache Spark is an analytics engine designed for big-data processing. It has polyglot support for programming languages including Scala, Java, Python and R.

The purpose of this assignment is to analyze the use of simulators that can determine the number of active cases likely to happen at a hospital. The task is to write a SparkML algorithm in order to predict one of the categories (RESOLVED) in this case, mentioned in the outcome variable.

Data Cleaning

The dataset roughly involves 1.7 lakh rows and contains 18 columns which are mentioned below :-

- _id
- Assigned_ID
- Outbreak Associated
- Age Group
- Neighbourhood Name
- FSA
- Source of Infection
- Classification
- Episode Date
- Reported Date
- Client Gender
- Outcome
- Currently Hospitalized
- Currently in ICU
- Currently Intubated
- Ever Hospitalized
- Ever in ICU
- Ever Intubated

For analysis purposes, most of the columns weren't relevant and so out of these 18 columns, about five variables were selected as input variables which include Classification, Age Group, Ever Intubated, Ever in ICU, Ever Hospitalized and the output variable Outcome.

The input variable “Classification” include two values “CONFIRMED” and “PROBABLE” and gives us a confirmed reason or probable output that the person might have COVID symptoms

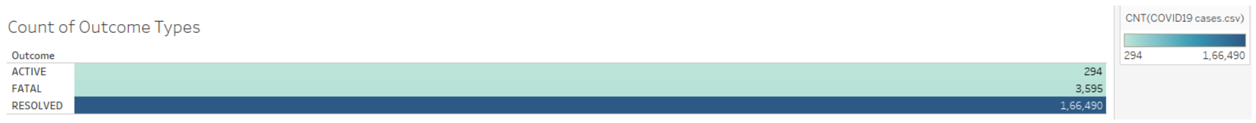
The input variable “Age Group” was chosen since most of the people having COVID-19 are old with ages more than 50 years.

The input variables “Ever in ICU”, “Ever Hospitalized”, “Ever Intubated” were chosen representing YES or NO to gather information about people who have been admitted in the past.

Since, the Outcome variable involves three values, FATAL, RESOLVED and ACTIVE which are categorical in nature, they were converted to 1’s and 0’s using the withColumn method and any NULL values were removed from the dataset.

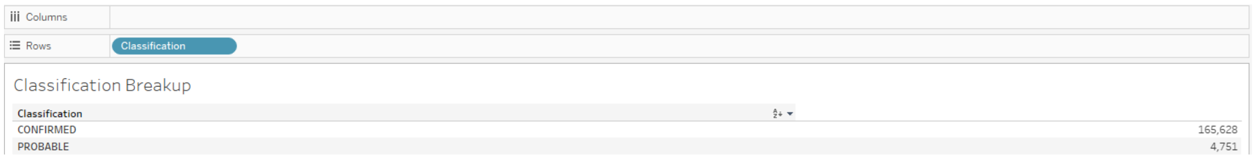
Using Tableau, some of the visualizations were made. Some of them are :-

Count of Outcome Types


Count of Outcome Types

Outcome	
ACTIVE	294
FATAL	3,595
RESOLVED	1,66,490

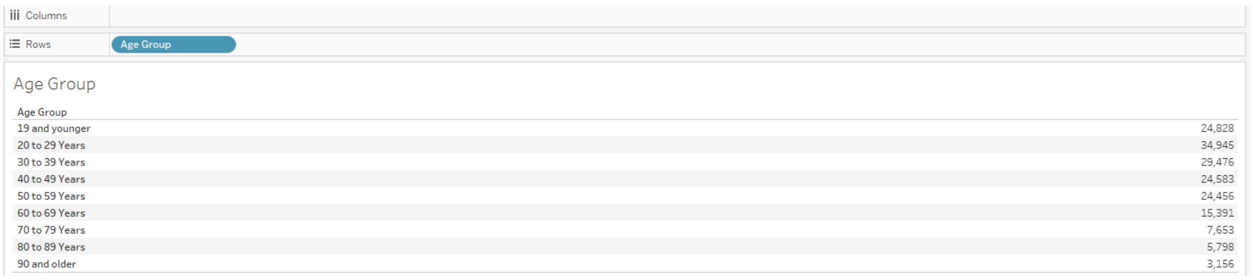
Classification Breakup



Classification Breakup

Classification	
CONFIRMED	165,628
PROBABLE	4,751

<u>Age Group</u>



Age Group

Age Group	
19 and younger	24,828
20 to 29 Years	34,945
30 to 39 Years	29,476
40 to 49 Years	24,583
50 to 59 Years	24,456
60 to 69 Years	15,391
70 to 79 Years	7,653
80 to 89 Years	5,798
90 and older	3,156

From the above visualizations, most of the people are from age 20 to 29 years which is about 35000. Most of the cases are CONFIRMED and very few people are present in the ICU. The dataset had most of the cases in RESOLVED outcome which was about 1.66 lakh.

Since, all of the chosen input columns were of String DataType, therefore, StringIndexer was applied to assign numerical value to the String columns.

Data Analysis

Based upon the initial data cleaning, the following columns were chosen as the input which include :-

- Classification
- Age Group
- Ever Intubated
- Ever in ICU
- Ever Hospitalized

After StringIndexing, these columns were fed as parameters to the VectorAssembler method which combines them into an output named "assembled-features" for our machine learning model.

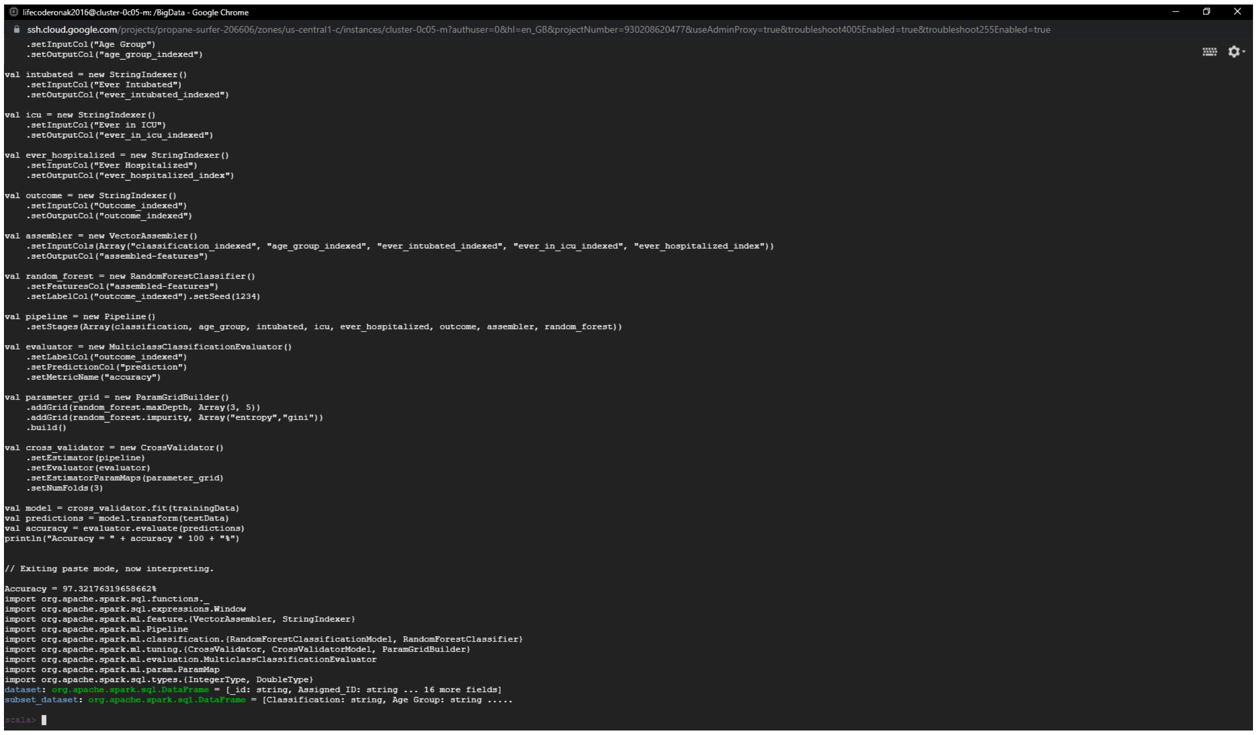
Following this, RandomForestClassifier class was instantiated which takes the "assembled-features" column as features column and the output variable as "outcome_indexed". A random seed value of 1234 was added so that the same results are reproducible. RandomForestClassifer was chosen as it avoids overfitting when compared to standard Decision Trees.

Furthermore, a pipeline was generated with stages that include all of the StringIndexer feature columns mentioned above with the outcome variable, random_forest and our assembler.

Finally, the MulticlassClassificationEvaluator method was instantiated with metrics as Accuracy along with the Parameter Grid to filter out the best accuracy with entropy value as "gini". Also, the model had cross validation to mitigate overfitting with numFolds being set to three.

The model was fitted with the training dataset containing 80% of the observations and predictions were generated on the remaining 20% of the dataset.

After running the code, the Accuracy came around 97.3% for the RESOLVED outcome, which is great for the ensemble learning method Random Forest being used. Here's the screenshot for the same


lifecoderonak2016@cluster-0c05-m: /BigData - Google Chrome
ssh.cloud.google.com/projects/propane-surfer-206606/zones/us-central1-c/instances/cluster-0c05-m?authuser=0&hl=en_GB&projectNumber=930208620477&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true

```
    .setInputCol("Age Group")
    .setOutputCol("age_group_indexed")

val intubated = new StringIndexer()
    .setInputCol("Ever Intubated")
    .setOutputCol("ever_intubated_indexed")

val icu = new StringIndexer()
    .setInputCol("Ever in ICU")
    .setOutputCol("ever_in_icu_indexed")

val ever_hospitalized = new StringIndexer()
    .setInputCol("Ever Hospitalized")
    .setOutputCol("ever_hospitalized_index")

val outcome = new StringIndexer()
    .setInputCol("Outcome_indexed")
    .setOutputCol("outcome_indexed")

val assembler = new VectorAssembler()
    .setInputCols(Array("classification_indexed", "age_group_indexed", "ever_intubated_indexed", "ever_in_icu_indexed", "ever_hospitalized_index"))
    .setOutputCol("assembled-features")

val random_forest = new RandomForestClassifier()
    .setFeaturesCol("assembled-features")
    .setLabelCol("outcome_indexed").setSeed(1234)

val pipeline = new Pipeline()
    .setStages(Array(classification, age_group, intubated, icu, ever_hospitalized, outcome, assembler, random_forest))

val evaluator = new MulticlassClassificationEvaluator()
    .setLabelCol("outcome_indexed")
    .setPredictionCol("prediction")
    .setMetricName("accuracy")

val parameter_grid = new ParamGridBuilder()
    .addGrid(random_forest.maxDepth, Array(3, 5))
    .addGrid(random_forest.impurity, Array("entropy","gini"))
    .build()

val cross_validator = new CrossValidator()
    .setEstimator(pipeline)
    .setEvaluator(evaluator)
    .setEstimatorParamMaps(parameter_grid)
    .setNumFolds(3)

val model = cross_validator.fit(trainingData)
val predictions = model.transform(testData)
val accuracy = evaluator.evaluate(predictions)
println("Accuracy = " + accuracy * 100 + "%")


// Exiting paste mode, now interpreting.

Accuracy = 97.32176319658662%
import org.apache.spark.sql.functions._
import org.apache.spark.sql.expressions.Window
import org.apache.spark.ml.feature.{VectorAssembler, StringIndexer}
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.classification.{RandomForestClassificationModel, RandomForestClassifier}
import org.apache.spark.ml.tuning.{CrossValidator, CrossValidatorModel, ParamGridBuilder}
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.ml.param.ParamMap
import org.apache.spark.sql.types.{IntegerType, DoubleType}
dataset: org.apache.spark.sql.DataFrame = [_id: string, Assigned_ID: string ... 16 more fields]
subset_dataset: org.apache.spark.sql.DataFrame = [Classification: string, Age Group: string .....

scala> 
```

Suggestions

- Random Forest is a great classifier with a very high accuracy of 97.3%. This can further be improved using parameter tuning
- Majority of the dataset involves RESOLVED in Outcome column which was the category predicted

Conclusion

To recapitulate, the dataset had 1.7 lakh rows and contained 18 columns which were filtered down to about 5 columns as input columns. These were converted into numerical values using StringIndexer. The model was generated using these columns and choosing the output variable being “Outcome” which was fitted on 80% of the original dataset. Finally, it was run on the remaining 20% of the testing data to generate predictions which resulted in an accuracy of around 97.3%.