Data Collection and Curation

Assignment – 2

Report

Machine Learning with Spark

Group 25 (One-member group)

By

Irfan Abdul Rahman

200480839

REPORT ON THE COVID-19 CASES OF THE CITY OF TORONTO

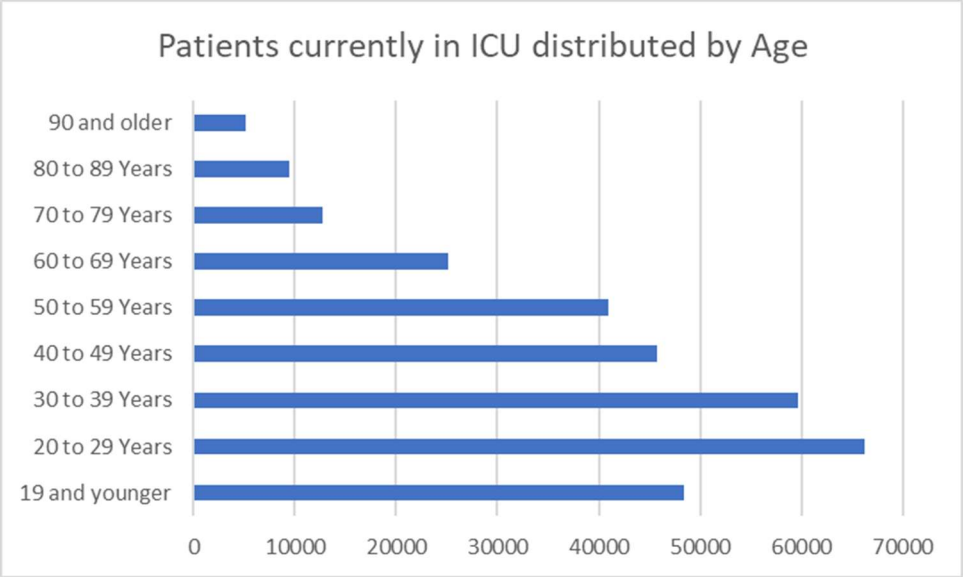
Introduction

The COVID-19 virus which can also be termed as coronavirus was a massive outbreak in the whole world. The people of Canada were also among the people of many different countries around the world who got affected losing their jobs and lives.

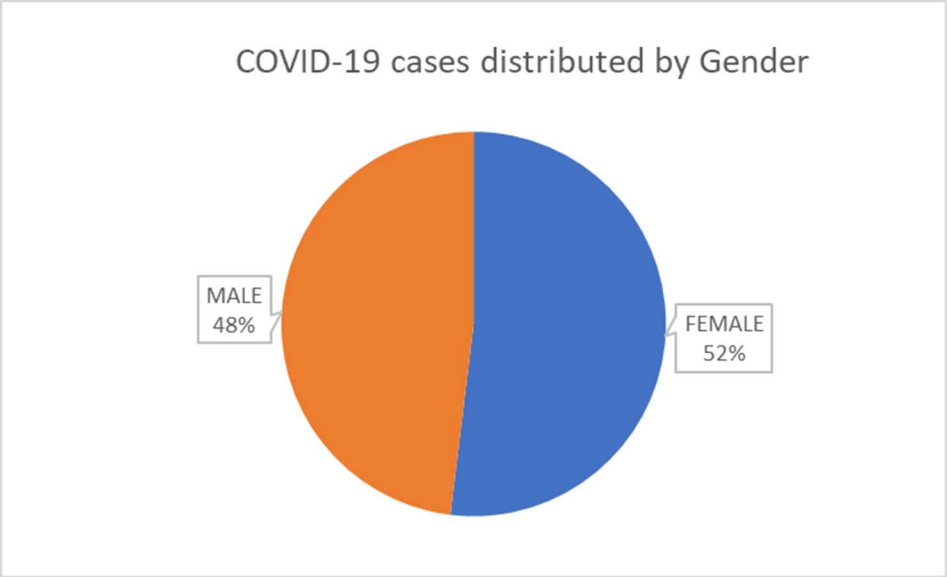
In this report, we have focused on the effects of the COVID -19 pandemic in the most populous city of Canada – Toronto. The city of Toronto recorded close to 300,000 cases in which thousands lost their lives.

With the COVID-19 information of patients of Toronto available in as open-source data, several exploratory analysis were made and also predictions were done to see if a patient is tend to get resolved or lose their life with the help machine learning algorithms.

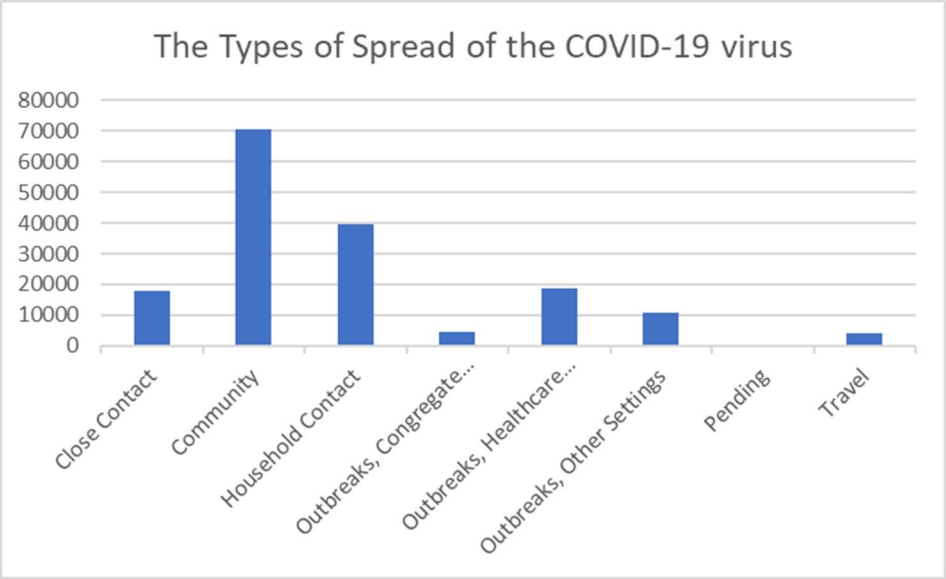
Exploratory Data Analysis



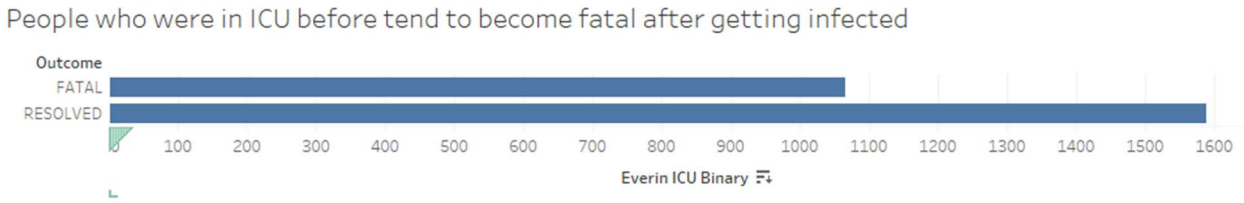
As we can infer from the above chart, the age group of people that suffered the most were in the category of 20 to 29 years.



As far as gender distribution is concerned the covid cases for both male and female were evenly distributed.



As expected, community spread recorded the highest count comparatively when analyzing at the source of infection.



The above chart clearly tells the story that people who have been in an ICU before are very vulnerable to becoming fatal.

Explanation of the code

```
irfanabdulrahman58@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/ivory-alcove-348014/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=en_US&projectNumber=391313542188&useAdminProxy=true&troubleshoot4005Enabled=true&tr...
Connected, host fingerprint: ssh-rsa 0 00:FA:7B:17:33:E3:0B:23:4D:CC:A3:67:B2:1E
:CA:7F:6A:E4:28:3E:A7:66:E8:6D:C0:EC:F0:31:6F:52:EC:0E
Linux bigdata-m 5.10.0-0.bpo.12-amd64 #1 SMP Debian 5.10.103-1~bpo10+1 (2022-03-
08) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Apr 23 20:43:42 2022 from 35.235.245.129
irfanabdulrahman58@bigdata-m:~$ hadoop fs -mkdir /BigData
mkdir: `/BigData': File exists
irfanabdulrahman58@bigdata-m:~$ hadoop fs -copyFromLocal COVID19cases11.csv /BigData/.
copyFromLocal: `/BigData/COVID19cases11.csv': File exists
irfanabdulrahman58@bigdata-m:~$ spark-shell --master yarn

Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/04/24 02:07:49 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
22/04/24 02:07:49 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
22/04/24 02:07:49 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
22/04/24 02:07:49 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
Spark context Web UI available at http://bigdata-m.us-central1-c.c.ivory-alcove-348014.internal:37651
Spark context available as 'sc' (master = yarn, app id = application_1650765991032_0001).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.1.2
      /_/

Using Scala version 2.12.14 (OpenJDK 64-Bit Server VM, Java 1.8.0_322)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

The latest version of Spark which is Spark – 3.0 was used in this analysis.

```
irfanabdulrahman58@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/ivory-alcove-348014/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=en_US&projectNumber=391313542188&useAdminProxy=true&troubleshoot4005Enabled=true&tr...
scala> :paste
// Entering paste mode (ctrl-D to finish)

import org.apache.spark.sql.functions._
import org.apache.spark.ml.feature.{VectorAssembler}
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.regression.{LinearRegression}
import org.apache.spark.ml.tuning.{CrossValidator, CrossValidatorModel, ParamGridBuilder}
import org.apache.spark.ml.evaluation.{RegressionEvaluator}
import org.apache.spark.ml.param.ParamMap
import org.apache.spark.sql.types.{DoubleType}
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}



// Exiting paste mode, now interpreting.

import org.apache.spark.sql.functions._
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.regression.LinearRegression
import org.apache.spark.ml.tuning.{CrossValidator, CrossValidatorModel, ParamGridBuilder}
import org.apache.spark.ml.evaluation.RegressionEvaluator
import org.apache.spark.ml.param.ParamMap
import org.apache.spark.sql.types.DoubleType
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}...

scala> :paste
// Entering paste mode (ctrl-D to finish)
```

All the necessary libraries were imported that are required to run the machine learning algorithm.

```
irfanabdulrahman58@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/ivory-alcove-348014/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=en_US&projectNumber=391313542188&useAdminProxy=true&troubleshoot4005Enabled=true&tr...
 .load("hdfs://10.128.0.2:8020/BigData/COVID19cases11.csv")


// Exiting paste mode, now interpreting.

covidresults: org.apache.spark.sql.DataFrame = [_id: string, Assigned_ID: string ... 18 more fields]

scala> :paste
// Entering paste mode (ctrl-D to finish)

val firstsort = covidresults.select(col("_id"), col("Outcome"), col("Age Group"), col("EverHospitalizedBinary").cast(IntegerType),
 col("EverinICUBinary").cast(IntegerType), col("Ever Intubated"))


// Exiting paste mode, now interpreting.

firstsort: org.apache.spark.sql.DataFrame = [_id: string, Outcome: string ... 4 more fields]

scala> :paste
// Entering paste mode (ctrl-D to finish)


val secondsort = firstsort.select(col("_id"), col("Age Group"), col("Outcome"), col("EverHospitalizedBinary").cast(IntegerType),
 col("EverinICUBinary").cast(IntegerType), col("Ever Intubated")). filter(col("Outcome") !== "ACTIVE")



// Exiting paste mode, now interpreting.

warning: one deprecation (since 2.0.0); for details, enable `:setting -deprecation' or `:replay -deprecation'
secondsort: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [_id: string, Age Group: string ... 4 more fields]

scala> :paste
// Entering paste mode (ctrl-D to finish)

val Array(trainingData, testData) = secondsort.randomSplit(Array(0.8, 0.2), 800)

// Exiting paste mode, now interpreting.

trainingData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [_id: string, Age Group: string ... 4 more fields]
testData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [_id: string, Age Group: string ... 4 more fields]
```

The dataset was split into train and test data in a proportion of 80 and 20 respectively.

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val indexer0 = new StringIndexer()
  .setInputCol("Outcome")
  .setOutputCol("Outcome_indexed")

// Exiting paste mode, now interpreting.

indexer0: org.apache.spark.ml.feature.StringIndexer = strIdx_0ad94555088c

scala> :paste
// Entering paste mode (ctrl-D to finish)

val assembler = new VectorAssembler()
 .setInputCols(Array("EverHospitalizedBinary", "EverinICUBinary"))
 .setOutputCol("assembled-features")

// Exiting paste mode, now interpreting.

assembler: org.apache.spark.ml.feature.VectorAssembler = VectorAssembler: uid=vecAssembler_4fd471c2226f, handleInvalid=error, numInputCols=2

scala> :paste
// Entering paste mode (ctrl-D to finish)

val rf = new RandomForestClassifier()
 .setFeaturesCol("assembled-features")
 .setLabelCol("Outcome_indexed")
 .setSeed(1234)


// Exiting paste mode, now interpreting.

rf: org.apache.spark.ml.classification.RandomForestClassifier = rfc_45ccc133523b

scala> :paste
// Entering paste mode (ctrl-D to finish)
```

The “outcome” variable was the label (dependent variable) and two variables namely “EverHospitalized” and “EverinICUBinary were the features.

```
val pipeline = new Pipeline()
  .setStages(Array(indexer0, assembler, rf))


// Exiting paste mode, now interpreting.

pipeline: org.apache.spark.ml.Pipeline = pipeline_0336d62485ae

scala> :paste
// Entering paste mode (ctrl-D to finish)

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

// Exiting paste mode, now interpreting.

evaluator: org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator = MulticlassClassificationEvaluator: uid=mcEval_c836a325d0d6, metricName=accuracy, metricLabel=0.0,
 beta=1.0, eps=1.0E-15

scala> :paste
// Entering paste mode (ctrl-D to finish)

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

// Exiting paste mode, now interpreting.
```

The pipeline was set up in an order such that the indexer run fast, followed by the assembler and the random forest classifier (Random Forest was the classification algorithm used since this was a classification problem).

```
paramGrid: Array[org.apache.spark.ml.param.ParamMap] =
Array({
        rfc_45ccc133523b-impurity: entropy,
        rfc_45ccc133523b-maxDepth: 3
}, {
        rfc_45ccc133523b-impurity: gini,
        rfc_45ccc133523b-maxDepth: 3
}, {
        rfc_45ccc133523b-impurity: entropy,
        rfc_45ccc133523b-maxDepth: 5
}, {
        rfc_45ccc133523b-impurity: gini,
        rfc_45ccc133523b-maxDepth: 5
})

scala> :paste
// Entering paste mode (ctrl-D to finish)

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

// Exiting paste mode, now interpreting.

cross_validator: org.apache.spark.ml.tuning.CrossValidator = cv_4029cfd8cdd4

scala> :paste
// Entering paste mode (ctrl-D to finish)

val cvModel = cross_validator.fit(trainingData)

// Exiting paste mode, now interpreting.
```

Cross Validation was performed for the division of the training datasets.

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val cvModel = cross_validator.fit(trainingData)

// Exiting paste mode, now interpreting.

cvModel: org.apache.spark.ml.tuning.CrossValidatorModel = CrossValidatorModel: uid=cv_4029cfd8cdd4, bestModel=pipeline_0336d62485ae, numFolds=3

scala> :paste
// Entering paste mode (ctrl-D to finish)

val predictions = cvModel.transform(testData)

// Exiting paste mode, now interpreting.

predictions: org.apache.spark.sql.DataFrame = [_id: string, Age Group: string ... 9 more fields]

scala> :paste
// Entering paste mode (ctrl-D to finish)

val accuracy = evaluator.evaluate(predictions)

// Exiting paste mode, now interpreting.

accuracy: Double = 0.9856332032635686
```

```
irfanabdulrahman58@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/ivory-alcove-348014/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=en_US&projectNumber=391313542188&useAdminProxy=true&troubleshoot4005Enabled=t
  .setEvaluator(evaluator)
  .setEstimatorParamMaps(paramGrid)
  .setNumFolds(3)

// Exiting paste mode, now interpreting.

cross_validator: org.apache.spark.ml.tuning.CrossValidator = cv_4029cfd8cdd4

scala> :paste
// Entering paste mode (ctrl-D to finish)

val cvModel = cross_validator.fit(trainingData)

// Exiting paste mode, now interpreting.

cvModel: org.apache.spark.ml.tuning.CrossValidatorModel = CrossValidatorModel: uid=cv_4029cfd8cdd4, bestModel=pipeline_0336d62485ae, numFolds=3

scala> :paste
// Entering paste mode (ctrl-D to finish)

val predictions = cvModel.transform(testData)

// Exiting paste mode, now interpreting.

predictions: org.apache.spark.sql.DataFrame = [_id: string, Age Group: string ... 9 more fields]

scala> :paste
// Entering paste mode (ctrl-D to finish)

val accuracy = evaluator.evaluate(predictions)

// Exiting paste mode, now interpreting.

accuracy: Double = 0.9856332032635686

scala> :paste
// Entering paste mode (ctrl-D to finish)

println("accuracy on test data = " + accuracy)

// Exiting paste mode, now interpreting.

accuracy on test data = 0.9856332032635686

scala>
```

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

println("accuracy on test data = " + accuracy)

// Exiting paste mode, now interpreting.

accuracy on test data = 0.9856332032635686
```

Conclusion –

The model was evaluated and an accuracy of 98% was achieved for the model created.