Covid Fatality Rate in Toronto Report

Contents

Abstract:

In January 2020, SARS-CoV-2, the novel coronavirus that causes COVID-19, was declared a global pandemic. There are several platforms that are updated in real time to study the effects and spread. This analysis focuses on the open data source Toronto, which is derived from the provincial Case & Contact Management System and updated weekly (CCM). The goal is to create and verify a model that uses Apache Spark machine learning methods to accurately predict fatality rates in the Toronto area.

Introduction

A new form of corona virus was discovered in Wuhan, the capital of Hubei Province in China, in early December 2019. The new virus, severe acute respiratory syndrome coronavirus virus (SARS-CoV-2), has been given the term coronavirus illness 2019 by the World Health Organization (COVID-19) Toronto Public Health has been updating the data collection utilized to conduct the data analysis on a weekly basis since January 2020. The time is updated on a weekly basis.

Interpretation of Data

Since the first case was reported in January 2020, the data collection contains demographic, geographic, and severity information for all confirmed and probable cases reported to and managed by Toronto Public Health. The source data comes from the province's Case and Contact Management System (CCM). The total number of rows in this project is 170621, which will be reduced to 163315 when null values and probable situations are removed.

The columns in the original dataset of the data model are as follows:

ID	Unique row identifier for Open Data database
Assigned_ID	A unique ID assigned to cases by Toronto Public Health for the purposes of posting to Open Data, to allow for tracking of specific cases
Outbreak Associated	Outbreak associated cases are associated with outbreaks of COVID-19 in Toronto healthcare institutions and healthcare settings.

Age Group	Classification bases on the age
Neighborhood Name	List of 140 Neighborhood of Toronto
FSA	Forward sortation area
Sourceof Infection	Source cause of Infection of COVID-19
Classification	Segregation between confirmed or probable cases.
Episode Date	Earliest available date from: symptom onset laboratory specimen collection date, or reported date
Reported Date	Case recorded on a precise Date to Toronto Public Health.
Client Gender	Self-reported gender
Outcome	Cases Reported as : Fatal, Resolved and Active
Currently Hospitalized	Cases, which are currently admitted to hospital
Currently in ICU	Cases, which are currently admitted to the intensive care unit (ICU)
Currently Intubated	Cases which were intubated
Ever Hospitalized	Cases which were hospitalized
Ever in ICU	Cases which were admitted to the intensive care unit (ICU)

Ever Intubated	Cases which were intubated

Structure of Data

1	Age Group	Neighbourhood I	FSA	Source of Infecti	Classification	Episode Date	Reported Date	Client Gender	Outcome	Currently Hospitalized	ntly in ICU	Currently Intubat	Ever Hospitalize	E
2	50 to 59 Years	Willowdale East	M2N	Travel	CONFIRMED	2020-01-22	2020-01-23	FEMALE	RESOLVED	No	No	No	No	N
3	50 to 59 Years	Willowdale East	M2N	Travel	CONFIRMED	2020-01-21	2020-01-23	MALE	RESOLVED	No	No	No	Yes	N
4	20 to 29 Years	Parkwoods-Don:	M3A	Travel	CONFIRMED	2020-02-05	2020-02-21	FEMALE	RESOLVED	No	No	No	No	N
5	60 to 69 Years	Church-Yonge C	M4W	Travel	CONFIRMED	2020-02-16	2020-02-25	FEMALE	RESOLVED	No	No	No	No	N
6	60 to 69 Years	Church-Yonge C	M4W	Travel	CONFIRMED	2020-02-20	2020-02-26	MALE	RESOLVED	No	No	No	No	N
7	50 to 59 Years	Newtonbrook W	M2R	Travel	CONFIRMED	2020-02-24	2020-02-27	MALE	RESOLVED	No	No	No	No	N
8	80 to 89 Years	Milliken	M1V	Travel	CONFIRMED	2020-02-20	2020-02-28	MALE	RESOLVED	No	No	No	No	N
9	60 to 69 Years	Willowdale West	M2N	Travel	CONFIRMED	2020-02-21	2020-03-04	MALE	RESOLVED	No	No	No	Yes	N
10	50 to 59 Years	Willowdale East	M2N	Travel	CONFIRMED	2020-02-29	2020-02-29	MALE	RESOLVED	No	No	No	No	N
11	60 to 69 Years	Henry Farm	M2J	Travel	CONFIRMED	2020-02-26	2020-03-01	MALE	RESOLVED	No	No	No	No	N
12	70 to 79 Years	Don Valley Villag	M2J	Travel	CONFIRMED	2020-02-14	2020-03-01	FEMALE	RESOLVED	No	No	No	No	N
13	50 to 59 Years	Lawrence Park S	M4R	Travel	PROBABLE	2020-03-01	2020-03-02	MALE	RESOLVED	No	No	No	No	N
14	60 to 69 Years	Bridle Path-Sunr	M2L	Travel	CONFIRMED	2020-03-02	2020-03-03	MALE	RESOLVED	No	No	No	No	N
15	30 to 39 Years	Moss Park	M5A	Community	PROBABLE	2020-03-03	2020-03-04	MALE	RESOLVED	No	No	No	No	N
16	40 to 49 Years	Annex	M6G	Travel	CONFIRMED	2020-03-02	2020-03-05	MALE	RESOLVED	No	No	No	No	N
17	50 to 59 Years	Willowdale East	M2N	Travel	CONFIRMED	2020-03-03	2020-03-05	MALE	RESOLVED	No	No	No	No	N
18	40 to 49 Years	Leaside-Bennin	M4G	Travel	CONFIRMED	2020-03-04	2020-03-07	FEMALE	RESOLVED	No	No	No	No	N
19	40 to 49 Years	Moss Park	M5A	Outbreaks, Con	CONFIRMED	2020-04-14	2020-03-06	MALE	RESOLVED	No	No	No	Yes	N
20	60 to 69 Years	St.Andrew-Wind	M2P	Travel	CONFIRMED	2020-03-05	2020-03-07	MALE	RESOLVED	No	No	No	No	N
21	80 to 89 Years	Willowdale East	M2N	Travel	CONFIRMED	2020-03-03	2020-03-08	MALE	RESOLVED	No	No	No	No	N
22	70 to 79 Years	Willowdale East	M2N	Travel	CONFIRMED	2020-03-03	2020-03-08	FEMALE	RESOLVED	No	No	No	No	N
23	60 to 69 Years	Malvern	M1B	Travel	CONFIRMED	2020-03-04	2020-03-08	FEMALE	RESOLVED	No	No	No	Yes	Y
24	40 to 49 Years	High Park North	M6P	Travel	CONFIRMED	2020-03-02	2020-03-09	MALE	RESOLVED	No	No	No	No	N
25	30 to 39 Years	Waterfront Com	M5V	Travel	CONFIRMED	2020-03-03	2020-03-10	MALE	RESOLVED	No	No	No	No	N
26	20 to 29 Years	Leaside-Bennin	M4G	Close Contact	CONFIRMED	2020-03-09	2020-03-10	MALE	RESOLVED	No	No	No	No	N
27	20 to 29 Years			Travel	PROBABLE	2020-03-02	2020-03-10	MALE	RESOLVED	No	No	No	No	N
28	40 to 49 Years	Mimico (includes	M8Y	Travel	CONFIRMED	2020-03-07	2020-03-11	FEMALE	RESOLVED	No	No	No	No	N
29	40 to 49 Years	Danforth-East Y	M4J	Travel	CONFIRMED	2020-03-09	2020-03-11	MALE	RESOLVED	No	No	No	No	N

COVID19 cases

Limitations

As public health investigations into reported instances and continuous quality improvement measures continue, and new cases are reported, the data in this spreadsheet is subject to change. The data will be totally refreshed and rewritten on a weekly basis, with the data being extracted at 8:30 a.m. on Tuesdays and posted on Wednesdays. Please keep in mind that these figures may differ from those published elsewhere because data is collected at various periods and from various sources.

Methodology

First, a csv dataset downloaded from an open-source dataset is placed in Hadoop, and then spark is used to analyse it. Machine learning in GCP processed the data during the data understanding phase, then balanced the prediction model with random forest after building the final dataframe with the new cleaned dataset.

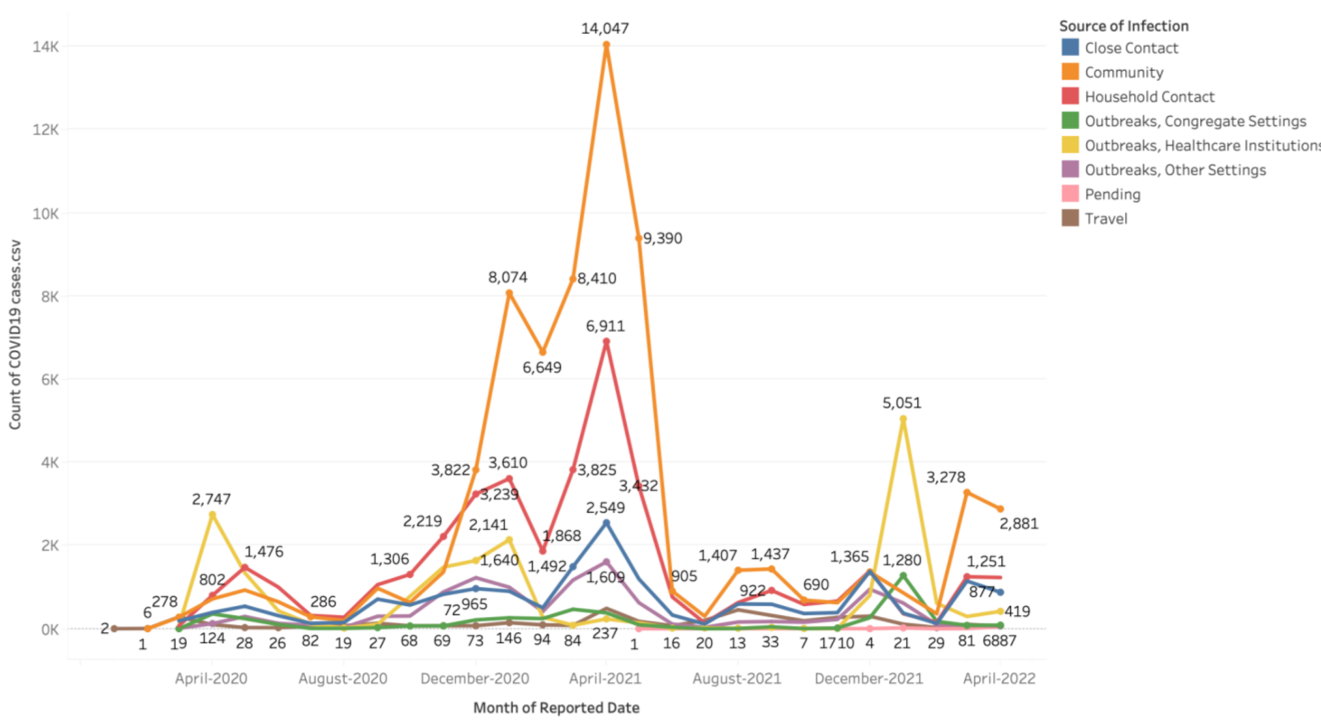
For categorised data, indexer and OneHotEncoder were used. The data is plotted and crucial useful insights are extracted from the data using the Tableau application.

Objectives

The objective of this data analysis is to produce a Spark machine learning prediction model that can accurately forecast fatality and fatality ratios. The following are the goals of this data analysis report: evaluate data and identify characteristics for the prediction model:

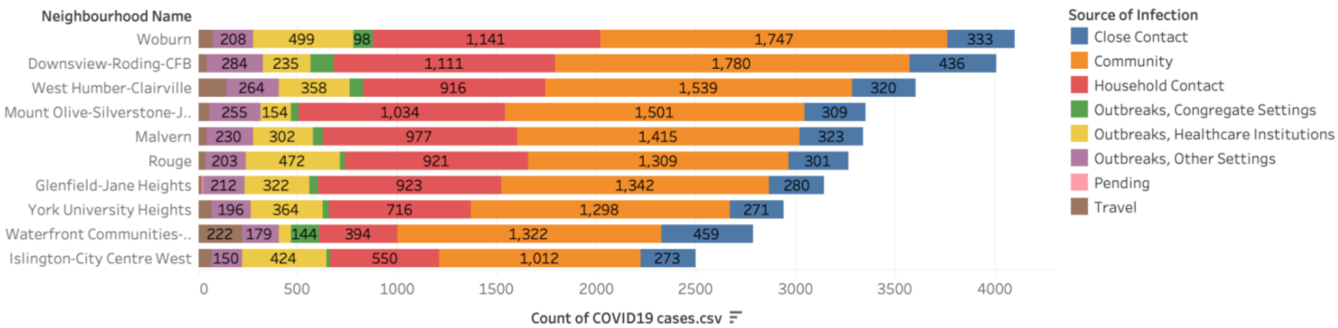
- The rise of cases based on the source of Infection.
- Which age group was the hardest hit?
- Influence of Top 10 Neighborhood on rise of covid cases.
- How well Covid-19 cases got treated.
- Medical History of different age groups which got infected.

Rise of Covid-19 cases based on Source of Infection



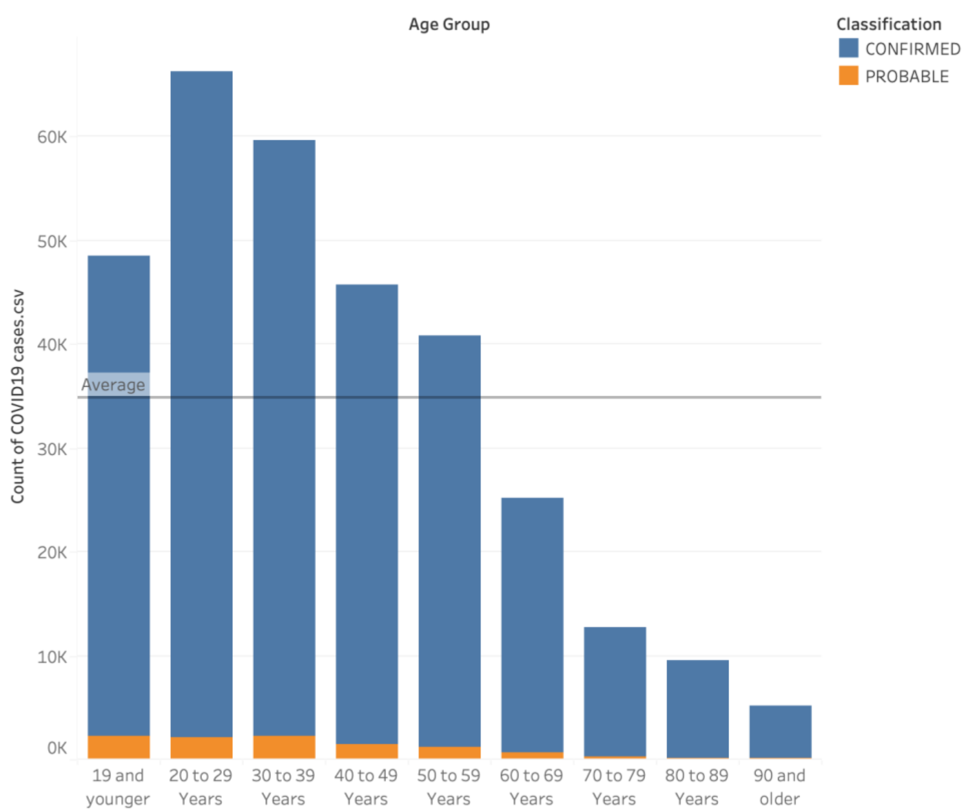
Above Analysis shows which Infection source has the highest impact of rise in covid 19 cases over the time. Here we can see the largest impact was from Community, which is almost 14,047 cases.

Influence of Top 10 Neighborhood on rise of covid cases.



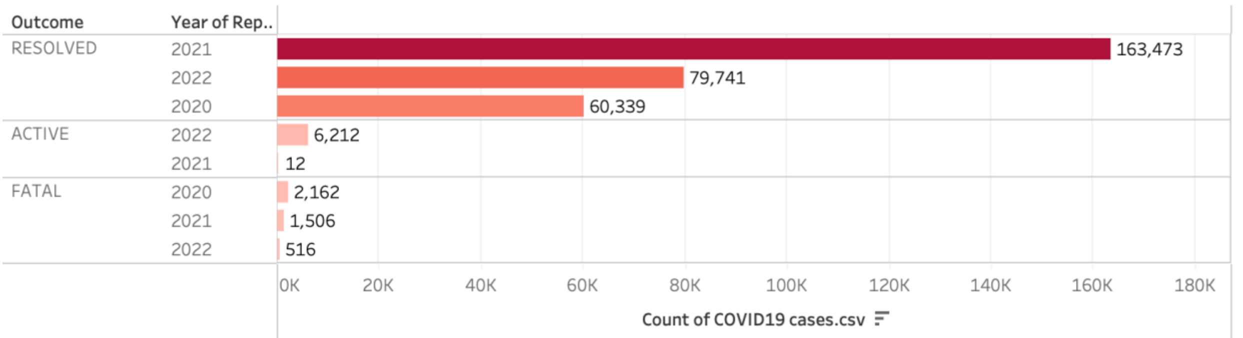
Now the above analysis further explains the root cause of spread based on top 10 neighborhoods which fuel the overall spread and hot spots which were supposed to be mobilized or locked down first. If we could observe the influence of community and household contracts was the highest in the neighborhood.

Which age group was the hardest hit?



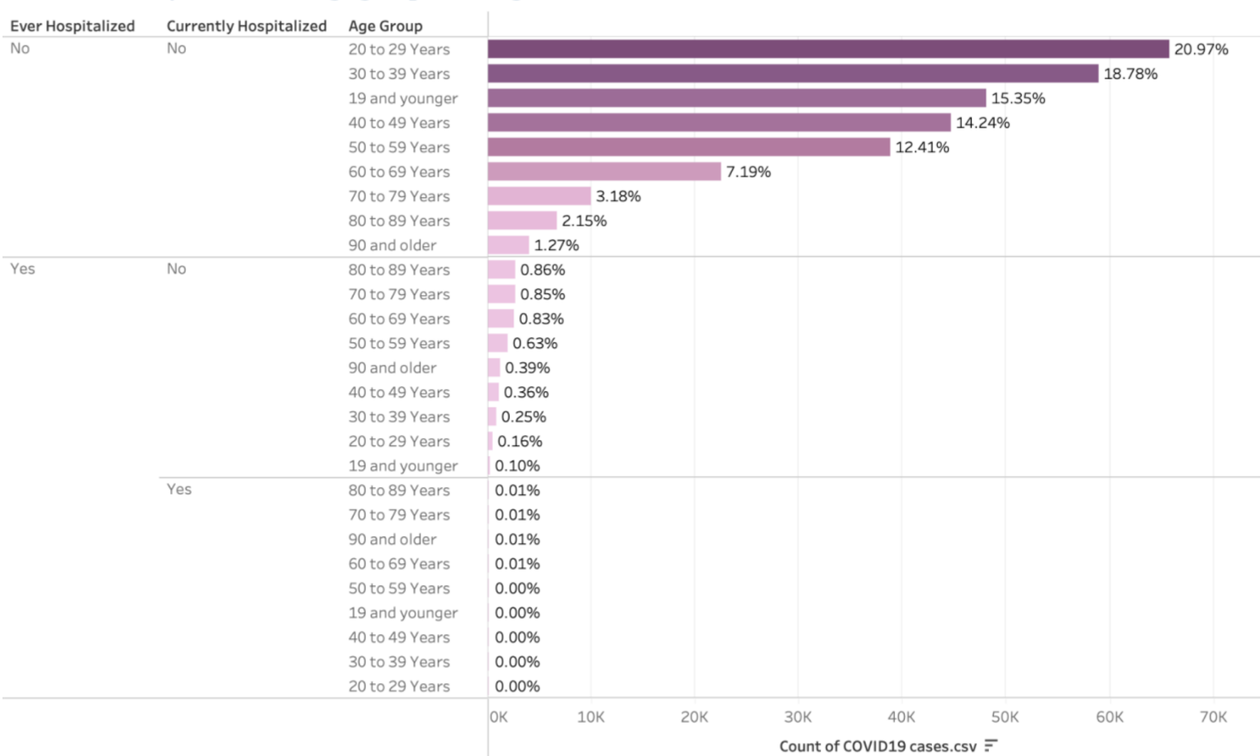
This analysis signifies which age group got the highest hit when it comes to the number of covid cases. Clearly the most impacted age group was 20 to 29 years. Followed by 30-39 years and then 19 or younger.

Insight on Outcome of Covid -19 Cases



Since the outburst of covid-19 cases, across the year of reported cases, 2021 was the year where it got controlled effectively. The fatal and active cases are comparatively less.

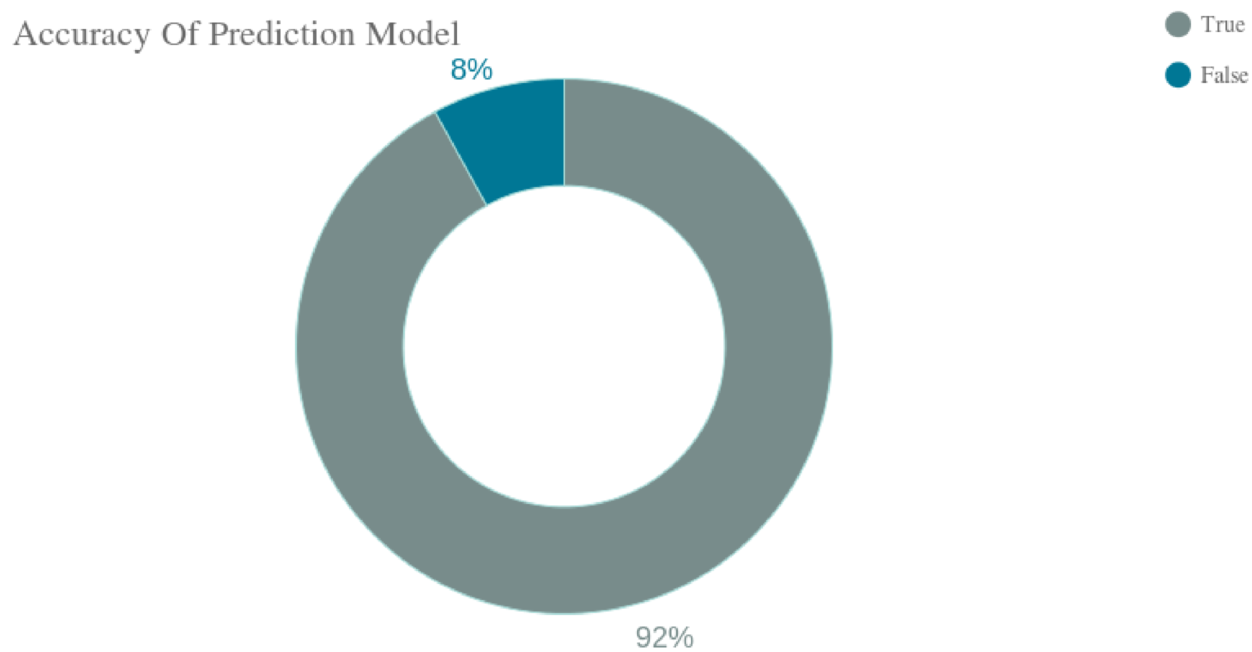
Medical History of different age groups which got infected.



Above analysis shows the medical history of different age groups who ever been hospitalized and if they are still in the hospital. If we can observe the percentage of people who got hospitalized whether they are currently in hospital or not is shockingly way less than the people who were never hospitalized and not currently in the hospital. Approximately 21% of people who has never been to hospital are not currently in the hospital which means the recovery rate is excellent.

Accuracy Of Prediction Model

The graph shows the results of the prediction model built in Spark giving 92.6% accuracy.



Conclusion and Recommendation

We were able to obtain around 92 percent accuracy using Random Forest to predict people with COVID-19. Random forest is a powerful and easy-to-use machine learning technique that provides outstanding results. The primary limitation of random forest is that a large number of trees might cause the process to slow down, rendering it useless for real-time predictions. These algorithms are generally quick to train but slow to forecast once they've been taught.

Appendix

Understanding and Previewing Data

- Load data from HDFS and cleaning it

```
valdf=spark.read.format("csv").option("header","true").load("hdfs://10.128.0.20:8020/BigData/covid/COVID19.csv")v
al cleanDF=df.na.drop()
```

```
import org.apache.spark.ml.tuning.{CrossValidator, CrossValidatorModel, ParamGridBuilder}

scala> import org.apache.spark.ml.feature.{StringIndexer, OneHotEncoder}
import org.apache.spark.ml.feature.{StringIndexer, OneHotEncoder}

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

val
df=spark.read
.format("csv")
.option("header",
"true")
.load("hdfs://10.128.0.20:8020/bigdata/COVID19")
val cleanDF=df.na.drop()


// Exiting paste mode, now interpreting.

df: org.apache.spark.sql.DataFrame = [_id: string, Assigned_ID: string ... 16 more fields]
cleanDF: org.apache.spark.sql.DataFrame = [_id: string, Assigned_ID: string ... 16 more fields]

scala> cleanDF.head(10)
res1: Array[org.apache.spark.sql.Row] = Array([1,1,Sporadic,50 to 59 Years,Willowdale East,M2N,Travel,CONFIRMED,2020-01-22,2020-01-23,FEMALE,RESOLVED,No,No,No,No,No,No], [2,2,Sporadic,
50 to 59 Years,Willowdale East,M2N,Travel,CONFIRMED,2020-01-21,2020-01-23,MALE,RESOLVED,No,No,No,Yes,No,No], [3,3,Sporadic,20 to 29 Years,Parkwoods-Donalda,M3A,Travel,CONFIRMED,2020-02
-05,2020-02-21,FEMALE,RESOLVED,No,No,No,No,No,No], [4,4,Sporadic,60 to 69 Years,Church-Yonge Corridor,M4W,Travel,CONFIRMED,2020-02-16,2020-02-25,FEMALE,RESOLVED,No,No,No,No,No,No], [5,
5,Sporadic,60 to 69 Years,Church-Yonge Corridor,M4W,Travel,CONFIRMED,2020-02-20,2020-02-26,MALE,RESOLVED,No,No,No,No,No,No], [6,6,Sporadic,50 to 59 Years,Newtonbrook West,M2R,Travel,CO
NFIRMED,2020-02-24,2020-02-27,MALE,...

scala> cleanDF.schema(10)
res2: org.apache.spark.sql.types.StructField = StructField(Client Gender,StringType,true)

scala> cleanDF.show(10)
+---+-----------+------------------+---------------+--------------------+---+-------------------+--------------+------------+-------------+-------------+--------+--------------------
+--------------+------------------+-----------------+-----------+-------------+
|_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|
+---+-----------+------------------+---------------+--------------------+---+-------------------+--------------+------------+-------------+-------------+--------+--------------------
+--------------+------------------+-----------------+-----------+-------------+
|  1|          1|          Sporadic|50 to 59 Years|     Willowdale East|M2N|             Travel|     CONFIRMED|  2020-01-22|   2020-01-23|       FEMALE|RESOLVED|                    No
|             No|                No|               No|         No|           No|
|  2|          2|          Sporadic|50 to 59 Years|     Willowdale East|M2N|             Travel|     CONFIRMED|  2020-01-21|   2020-01-23|         MALE|RESOLVED|                    No
|             No|                No|              Yes|         No|           No|
|  3|          3|          Sporadic|20 to 29 Years|   Parkwoods-Donalda|M3A|             Travel|     CONFIRMED|  2020-02-05|   2020-02-21|       FEMALE|RESOLVED|                    No
|             No|                No|               No|         No|           No|
|  4|          4|          Sporadic|60 to 69 Years|Church-Yonge Corr...|M4W|             Travel|     CONFIRMED|  2020-02-16|   2020-02-25|       FEMALE|RESOLVED|                    No
|             No|                No|               No|         No|           No|
|  5|          5|          Sporadic|60 to 69 Years|Church-Yonge Corr...|M4W|             Travel|     CONFIRMED|  2020-02-20|   2020-02-26|         MALE|RESOLVED|                    No
|             No|                No|               No|         No|           No|
|  6|          6|          Sporadic|50 to 59 Years|    Newtonbrook West|M2R|             Travel|     CONFIRMED|  2020-02-24|   2020-02-27|         MALE|RESOLVED|                    No
|             No|                No|               No|         No|           No|
|  7|          7|          Sporadic|80 to 89 Years|            Milliken|M1V|             Travel|     CONFIRMED|  2020-02-20|   2020-02-28|         MALE|RESOLVED|                    No
|             No|                No|               No|         No|           No|
|  8|          8|          Sporadic|60 to 69 Years|     Willowdale West|M2N|             Travel|     CONFIRMED|  2020-02-21|   2020-03-04|         MALE|RESOLVED|                    No
```

• Load new dataset with selected fields and checking the data

val dataset=cleanDF.select(col("Outcome"),

col("Age Group"),

col("Ever Hospitalized"),

col("Ever in ICU"),

col("Ever Intubated"),

col("Client Gender"))

.filter(cleanDF("Classification")==="CONFIRMED")

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

val dataset=cleanDF.select(col("Outcome"),
col("Age Group"),
col("Ever Hospitalized"),
col("Ever in ICU"),
col("Ever Intubated"),
col("Client Gender"))
.filter(cleanDF("Classification")==="CONFIRMED")

// Exiting paste mode, now interpreting.

dataset: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Outcome: string, Age Group: string ... 4 more fields]

scala> dataset.show(10)
+--------+--------------+-----------------+-----------+--------------+--------
-----+
| Outcome|     Age Group|Ever Hospitalized|Ever in ICU|Ever Intubated|Client G
ender|
+--------+--------------+-----------------+-----------+--------------+--------
-----+
|RESOLVED|50 to 59 Years|               No|         No|            No|       F
EMALE|
|RESOLVED|50 to 59 Years|              Yes|         No|            No|
 MALE|
|RESOLVED|20 to 29 Years|               No|         No|            No|       F
EMALE|
|RESOLVED|60 to 69 Years|               No|         No|            No|       F
EMALE|
|RESOLVED|60 to 69 Years|               No|         No|            No|
 MALE|
|RESOLVED|50 to 59 Years|               No|         No|            No|
 MALE|
|RESOLVED|80 to 89 Years|               No|         No|            No|
 MALE|
|RESOLVED|60 to 69 Years|              Yes|         No|            No|
 MALE|
|RESOLVED|50 to 59 Years|               No|         No|            No|
 MALE|
|RESOLVED|60 to 69 Years|               No|         No|            No|
 MALE|
+--------+--------------+-----------------+-----------+--------------+--------
-----+
only showing top 10 rows
```

Balanced Dataset

val fatalityDf = dataset.filter(dataset("Outcome") === "FATAL")

val nonfatalityDf = dataset.filter(dataset("Outcome")=== "RESOLVED")

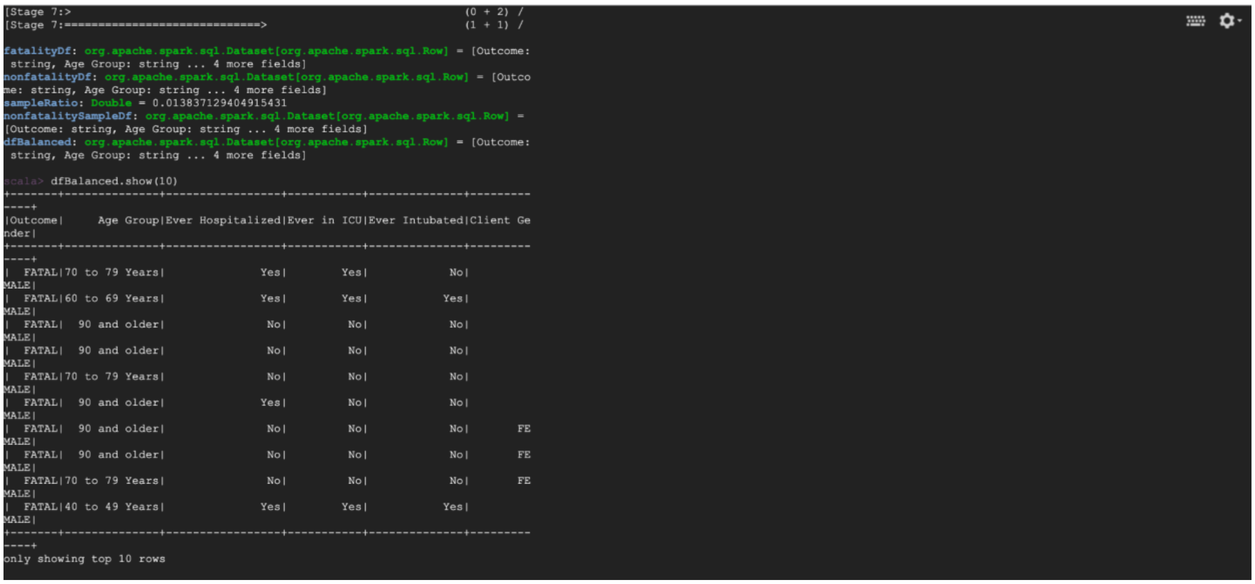
val sampleRatio = fatalityDf.count().toDouble/dataset.count().toDouble

val nonfatalitySampleDf = nonfatalityDf.sample(false, sampleRatio)

val dfBalanced = fatalityDf.unionAll(nonfatalitySampleDf)

To Show the data

dfBalanced.show(10)



```
[Stage 7:>                                                          (0 + 2) /
[Stage 7:=============================>                             (1 + 1) /

fatalityDf: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Outcome:
 string, Age Group: string ... 4 more fields]
nonfatalityDf: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Outco
me: string, Age Group: string ... 4 more fields]
sampleRatio: Double = 0.013837129404915431
nonfatalitySampleDf: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] =
[Outcome: string, Age Group: string ... 4 more fields]
dfBalanced: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Outcome:
 string, Age Group: string ... 4 more fields]

scala> dfBalanced.show(10)
+-------+-------------+-----------------+-----------+-------------+---------
----+
|Outcome|     Age Group|Ever Hospitalized|Ever in ICU|Ever Intubated|Client Ge
nder|
+-------+-------------+-----------------+-----------+-------------+---------
----+
|  FATAL|70 to 79 Years|              Yes|        Yes|           No|
MALE|
|  FATAL|60 to 69 Years|              Yes|        Yes|          Yes|
MALE|
|  FATAL|  90 and older|               No|         No|           No|
MALE|
|  FATAL|  90 and older|               No|         No|           No|
MALE|
|  FATAL|70 to 79 Years|               No|         No|           No|
MALE|
|  FATAL|  90 and older|              Yes|         No|           No|
MALE|
|  FATAL|  90 and older|               No|         No|           No|       FE
MALE|
|  FATAL|  90 and older|               No|         No|           No|       FE
MALE|
|  FATAL|70 to 79 Years|               No|         No|           No|       FE
MALE|
|  FATAL|40 to 49 Years|              Yes|        Yes|          Yes|
MALE|
+-------+-------------+-----------------+-----------+-------------+---------
----+
only showing top 10 rows
```

Indexing

val inputColumns = Array("Age Group","Ever Hospitalized", "Ever in ICU","Ever Intubated","Client Gender")

val outputColumns =Array("Age_index","Hospitalized_index","ICU_index","Intubated_index","Gender_index")

val indexer = new StringIndexer()

indexer.setInputCols(inputColumns)

indexer.setOutputCols(outputColumns)

val stringIndexer = new StringIndexer().setInputCol("Outcome").setOutputCol("Outcome_index")

val DF_indexed = indexer.fit(dfBalanced).transform(dfBalanced)

val DF_indexed2 = stringIndexer.fit(DF_indexed).transform(DF_indexed)

val rankDf = DF_indexed2.select(col("Outcome_index").cast(IntegerType),

col("Age_index").cast(IntegerType),

col("Hospitalized_index").cast(IntegerType),

col("ICU_index").cast(IntegerType),

col("Intubated_index").cast(IntegerType),

col("Gender_index").cast(IntegerType))

rankDf.show(10)

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

val fatalityDf = dataset.filter(dataset("Outcome") === "FATAL")
val nonfatalityDf = dataset.filter(dataset("Outcome")=== "RESOLVED")
val sampleRatio = fatalityDf.count().toDouble/dataset.count().toDouble
val nonfatalitySampleDf = nonfatalityDf.sample(false, sampleRatio)
val dfBalanced = fatalityDf.unionAll(nonfatalitySampleDf)
```

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

val stringIndexer = new StringIndexer().setInputCol("Outcome").setOutputCol("Out
come_index")
val DF_indexed = indexer.fit(dfBalanced).transform(dfBalanced)
val DF_indexed2 = stringIndexer.fit(DF_indexed).transform(DF_indexed)
val rankDf = DF_indexed2.select(col("Outcome_index").cast(IntegerType),
col("Age_index").cast(IntegerType),
col("Hospitalized_index").cast(IntegerType),
col("ICU_index").cast(IntegerType),
col("Intubated_index").cast(IntegerType),
col("Gender_index").cast(IntegerType))

// Exiting paste mode, now interpreting.

stringIndexer: org.apache.spark.ml.feature.StringIndexer = strIdx_c0ac25b45e63
DF_indexed: org.apache.spark.sql.DataFrame = [Outcome: string, Age Group: string
 ... 9 more fields]
DF_indexed2: org.apache.spark.sql.DataFrame = [Outcome: string, Age Group: strin
g ... 10 more fields]
rankDf: org.apache.spark.sql.DataFrame = [Outcome_index: int, Age_index: int ...
 4 more fields]

scala> rankDf.show(10)
+-------------+---------+------------------+---------+---------------+----------
--+
|Outcome_index|Age_index|Hospitalized_index|ICU_index|Intubated_index|Gender_ind
ex|
+-------------+---------+------------------+---------+---------------+----------
--+
|            0|        2|                 1|        1|              0|
 0|
|            0|        4|                 1|        1|              1|
 0|
|            0|        1|                 0|        0|              0|
 0|
|            0|        1|                 0|        0|              0|
 0|
|            0|        2|                 0|        0|              0|
 0|
|            0|        1|                 1|        0|              0|
 0|
|            0|        1|                 0|        0|              0|
 1|
|            0|        1|                 0|        0|              0|
 1|
|            0|        2|                 0|        0|              0|
 1|
|            0|        7|                 1|        1|              1|
 0|
+-------------+---------+------------------+---------+---------------+----------
--+
only showing top 10 rows
```

OneHotEncoder

val encoder = new OneHotEncoder()

.setInputCols(Array("Age_index","Hospitalized_index","ICU_index","Intubated_index","Gender_index"))

.setOutputCols(Array("Age_vector","Hospitalized_vector","ICU_vector","Intubated_vector","Gender_vector"))

val DF_Encoder= encoder.fit(rankDf).transform(rankDf)

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

val encoder = new OneHotEncoder()
.setInputCols(Array("Age_index","Hospitalized_index","ICU_index","Intubated_inde
x","Gender_index"))
.setOutputCols(Array("Age_vector","Hospitalized_vector","ICU_vector","Intubated_
vector","Gender_vector"))

// Exiting paste mode, now interpreting.

encoder: org.apache.spark.ml.feature.OneHotEncoder = oneHotEncoder_aa021dca7d55

scala> val DF_Encoder= encoder.fit(rankDf).transform(rankDf)
DF_Encoder: org.apache.spark.sql.DataFrame = [Outcome_index: int, Age_index: int
 ... 9 more fields]

scala> DF_Encoder.show(10)
+-------------+---------+------------------+---------+---------------+----------
--+-------------+----------------+-------------+-------------+------------------
-+
|Outcome_index|Age_index|Hospitalized_index|ICU_index|Intubated_index|Gender_ind
ex|Gender_vector|Intubated_vector|   Age_vector|   ICU_vector|Hospitalized_vecto
r|
```

DF_Encoder.show(10)

```
scala> DF_Encoder.show(10)
+-------------+---------+------------------+---------+---------------+------------+-------------+----------------+-------------+-------------+-------------------+
|Outcome_index|Age_index|Hospitalized_index|ICU_index|Intubated_index|Gender_index|Gender_vector|Intubated_vector|   Age_vector|   ICU_vector|Hospitalized_vector|
+-------------+---------+------------------+---------+---------------+------------+-------------+----------------+-------------+-------------+-------------------+
|            0|        2|                 1|        1|              0|           0|(2,[0],[1.0])|   (1,[0],[1.0])|(8,[2],[1.0])|    (1,[],[])|          (1,[],[])|
|            0|        4|                 1|        1|              1|           0|(2,[0],[1.0])|       (1,[],[])|(8,[4],[1.0])|    (1,[],[])|          (1,[],[])|
|            0|        1|                 0|        0|              0|           0|(2,[0],[1.0])|   (1,[0],[1.0])|(8,[1],[1.0])|(1,[0],[1.0])|      (1,[0],[1.0])|
|            0|        1|                 0|        0|              0|           0|(2,[0],[1.0])|   (1,[0],[1.0])|(8,[1],[1.0])|(1,[0],[1.0])|      (1,[0],[1.0])|
|            0|        2|                 0|        0|              0|           0|(2,[0],[1.0])|   (1,[0],[1.0])|(8,[2],[1.0])|(1,[0],[1.0])|      (1,[0],[1.0])|
|            0|        1|                 1|        0|              0|           0|(2,[0],[1.0])|   (1,[0],[1.0])|(8,[1],[1.0])|(1,[0],[1.0])|          (1,[],[])|
|            0|        1|                 0|        0|              0|           1|(2,[1],[1.0])|   (1,[0],[1.0])|(8,[1],[1.0])|(1,[0],[1.0])|      (1,[0],[1.0])|
|            0|        1|                 0|        0|              0|           1|(2,[1],[1.0])|   (1,[0],[1.0])|(8,[1],[1.0])|(1,[0],[1.0])|      (1,[0],[1.0])|
|            0|        2|                 0|        0|              0|           1|(2,[1],[1.0])|   (1,[0],[1.0])|(8,[2],[1.0])|(1,[0],[1.0])|      (1,[0],[1.0])|
|            0|        7|                 1|        1|              1|           0|(2,[0],[1.0])|       (1,[],[])|(8,[7],[1.0])|    (1,[],[])|          (1,[],[])|
+-------------+---------+------------------+---------+---------------+------------+-------------+----------------+-------------+-------------+-------------------+
only showing top 10 rows
```

Random Forest Machine Learning

val Array(trainingData,testData) = DF_Encoder.randomSplit(Array(0.8,0.2),650)

val assembler = new VectorAssembler()
.setInputCols(Array("Age_vector","Hospitalized_vector","ICU_vector","Intubated_vector","Gender_vector",
"Age_index","Hospitalized_index","ICU_index","Intubated_index","Gender_index"))
.setOutputCol("assembled-features")

val rf = new RandomForestClassifier()
.setFeaturesCol("assembled-features")
.setLabelCol("Outcome_index")
.setSeed(1234)
val pipeline = new Pipeline()
.setStages(Array(assembler,rf))

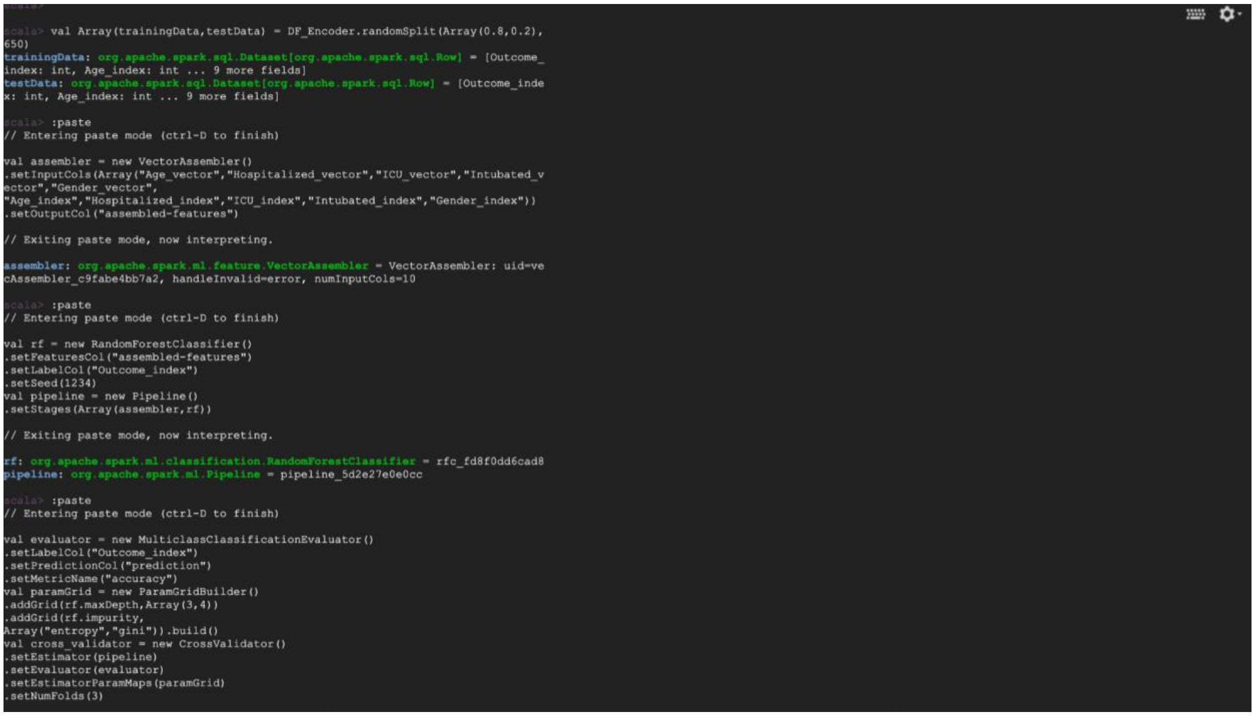
val evaluator = new MulticlassClassificationEvaluator()
.setLabelCol("Outcome_index")
.setPredictionCol("prediction")
.setMetricName("accuracy")
val paramGrid = new ParamGridBuilder()
.addGrid(rf.maxDepth,Array(3,4))
.addGrid(rf.impurity,
Array("entropy","gini")).build()
val cross_validator = new CrossValidator()
.setEstimator(pipeline)
.setEvaluator(evaluator)

.setEstimatorParamMaps(paramGrid)

.setNumFolds(3)

val cvModel=cross_validator.fit(trainingData)

val predictions = cvModel.transform(testData)



```
scala> val Array(trainingData,testData) = DF_Encoder.randomSplit(Array(0.8,0.2),
650)
trainingData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Outcome_
index: int, Age_index: int ... 9 more fields]
testData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Outcome_inde
x: int, Age_index: int ... 9 more fields]

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

val assembler = new VectorAssembler()
.setInputCols(Array("Age_vector","Hospitalized_vector","ICU_vector","Intubated_v
ector","Gender_vector",
"Age_index","Hospitalized_index","ICU_index","Intubated_index","Gender_index"))
.setOutputCol("assembled-features")

// Exiting paste mode, now interpreting.

assembler: org.apache.spark.ml.feature.VectorAssembler = VectorAssembler: uid=ve
cAssembler_c9fabe4bb7a2, handleInvalid=error, numInputCols=10

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

val rf = new RandomForestClassifier()
.setFeaturesCol("assembled-features")
.setLabelCol("Outcome_index")
.setSeed(1234)
val pipeline = new Pipeline()
.setStages(Array(assembler,rf))

// Exiting paste mode, now interpreting.

rf: org.apache.spark.ml.classification.RandomForestClassifier = rfc_fd8f0dd6cad8
pipeline: org.apache.spark.ml.Pipeline = pipeline_5d2e27e0e0cc

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

val evaluator = new MulticlassClassificationEvaluator()
.setLabelCol("Outcome_index")
.setPredictionCol("prediction")
.setMetricName("accuracy")
val paramGrid = new ParamGridBuilder()
.addGrid(rf.maxDepth,Array(3,4))
.addGrid(rf.impurity,
Array("entropy","gini")).build()
val cross_validator = new CrossValidator()
.setEstimator(pipeline)
.setEvaluator(evaluator)
.setEstimatorParamMaps(paramGrid)
.setNumFolds(3)
```

```
scala>

scala> val Array(trainingData,testData) = DF_Encoder.randomSplit(Array(0.8,0.2),
650)
trainingData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Outcome_
index: int, Age_index: int ... 9 more fields]
```

Final Output

val accuracy = evaluator.evaluate(predictions)

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

```
scala> val accuracy = evaluator.evaluate(predictions)
accuracy: Double = 0.9282178217821783

scala> println("accuracy on test data="+accuracy)
accuracy on test data=0.9282178217821783

scala>
```

Reference

Amini, S. (2022). *Data Analytics with Spark and Machine Learning*. Sample Assignment. Retrieved 2022, from https://www.dropbox.com/sh/ehhoe2k3ggbrfyp/AADb5DkX1SRV1ra3LqT01ZCLa?dl=0&preview=Assignment+%232+Spark+Machine+Learning+-+Sample+Report.pdf

Open data dataset. City of Toronto Open Data Portal. (n.d.). Retrieved April 23, 2022, from https://open.toronto.ca/dataset/covid-19-cases-in-toronto/