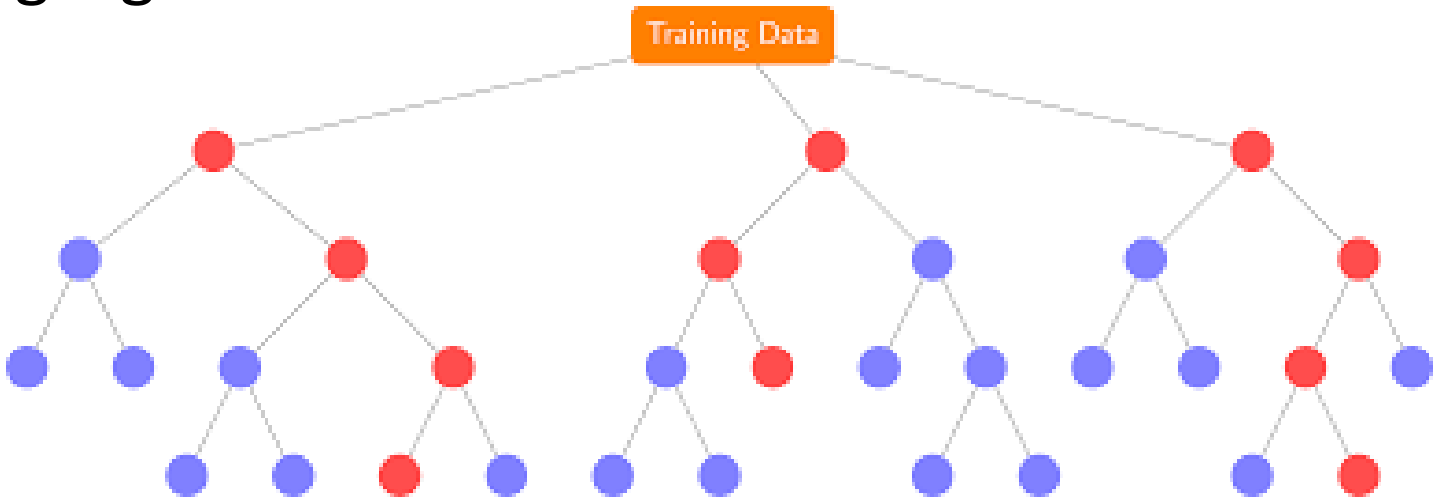
Problem Statement

The COVID-19 outbreak may increase, and so the hospitals may anytime be fully occupied. The sudden increase in the patient population may cause a plethora of problems to hospital management. The problem statement here is to predict the number of patients who may be admitted to the hospital in the future. Moreover, this helps hospital management to be ready for admitting any number of patients with ease.

Plan of Action

In order to predict future COVID-19 cases, machine learning algorithms should be used. Also, the Random Forest algorithm will be the best match for predicting future patients getting into hospitals. As the dataset is a bit big, the SPARK language is used for implementing this machine learning algorithm.



About the Dataset

The dataset has Toronto's COVID-19 outbreak data, and it has around 170000 rows and 18 columns, and they are as follow, Currently Intubated, Source of Infection, Ever Hospitalized, Ever in ICU, Age Group, Client Gender, Ever Intubated, Currently Hospitalized, Currently in ICU, Id, Assigned Id, Neighborhood Name, FSA, Classification, Reported Date, Episode Date, Classification and Outbreak Associated.

REFERENCE:

https://open.toronto.ca/dataset/covid-19-cases-in-toronto/

Data cleaning

The data cleaning part was done with Microsoft Excel, and the string values are replaced with integer values i.e. 1 and 0 for implementing the Random Forest algorithm for predicting the number of patients who may be admitted to the hospital in the future in Toronto.

VLOOKUP TABLES and IF STATEMENTS are used for replacing strings to 1’s and 0’s.

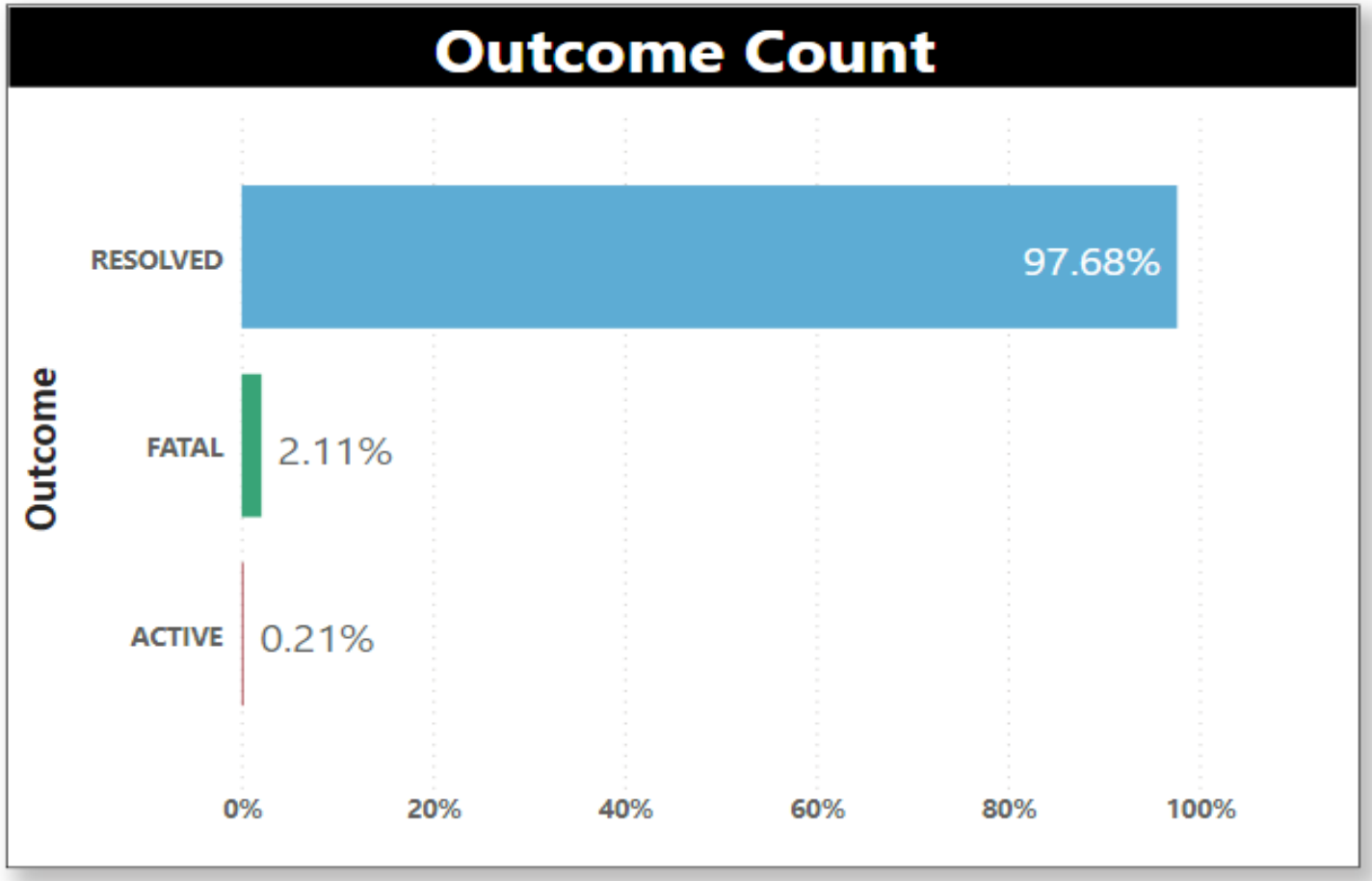
EXAMPLE:

1. =IF(M2=“YES”,1,0)
2. =VLOOKUP(D2,$S$2:$T$10,2,FALSE)

The first 20 rows and all columns of the dataset

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	_id	Assigned_	Outbreak	Age Group	Neighbour	FSA	Source of	Classificat	Episode Dat	Reported Da	Client Gen	Outcome	Currently	Currently	Currently	Ever Hosp	Ever in ICU	Ever Intub
2	3400296	1	Sporadic	5	Willowdal	M2N	9	CONFIRMI	22-01-2020	23-01-2020	0	RESOLVED	0	0	0	0	0	0
3	3400297	2	Sporadic	5	Willowdal	M2N	9	CONFIRMI	21-01-2020	23-01-2020	1	RESOLVED	0	0	0	1	0	0
4	3400298	3	Sporadic	2	Parkwood	M3A	9	CONFIRMI	05-02-2020	21-02-2020	0	RESOLVED	0	0	0	0	0	0
5	3400299	4	Sporadic	6	Church-Yo	M4W	9	CONFIRMI	16-02-2020	25-02-2020	0	RESOLVED	0	0	0	0	0	0
6	3400300	5	Sporadic	6	Church-Yo	M4W	9	CONFIRMI	20-02-2020	26-02-2020	1	RESOLVED	0	0	0	0	0	0
7	3400301	6	Sporadic	5	Newtonbr	M2R	9	CONFIRMI	24-02-2020	27-02-2020	1	RESOLVED	0	0	0	0	0	0
8	3400302	7	Sporadic	8	Milliken	M1V	9	CONFIRMI	20-02-2020	28-02-2020	1	RESOLVED	0	0	0	0	0	0
9	3400303	8	Sporadic	6	Willowdal	M2N	9	CONFIRMI	21-02-2020	04-03-2020	1	RESOLVED	0	0	0	1	0	0
10	3400304	9	Sporadic	5	Willowdal	M2N	9	CONFIRMI	29-02-2020	29-02-2020	1	RESOLVED	0	0	0	0	0	0
11	3400305	10	Sporadic	6	Henry Far	M2J	9	CONFIRMI	26-02-2020	01-03-2020	1	RESOLVED	0	0	0	0	0	0
12	3400306	11	Sporadic	7	Don Valley	M2J	9	CONFIRMI	14-02-2020	01-03-2020	0	RESOLVED	0	0	0	0	0	0
13	3400307	12	Sporadic	5	Lawrence	M4R	9	PROBABLE	01-03-2020	02-03-2020	1	RESOLVED	0	0	0	0	0	0
14	3400308	13	Sporadic	6	Bridle Path	M2L	9	CONFIRMI	02-03-2020	03-03-2020	1	RESOLVED	0	0	0	0	0	0
15	3400309	14	Sporadic	3	Moss Park	M5A	2	PROBABLE	03-03-2020	04-03-2020	1	RESOLVED	0	0	0	0	0	0
16	3400310	15	Sporadic	4	Annex	M6G	9	CONFIRMI	02-03-2020	05-03-2020	1	RESOLVED	0	0	0	0	0	0
17	3400311	16	Sporadic	5	Willowdal	M2N	9	CONFIRMI	03-03-2020	05-03-2020	1	RESOLVED	0	0	0	0	0	0
18	3400312	18	Sporadic	4	Leaside-Be	M4G	9	CONFIRMI	04-03-2020	07-03-2020	0	RESOLVED	0	0	0	0	0	0
19	3400313	19	Outbreak	4	Moss Park	M5A	5	CONFIRMI	14-04-2020	06-03-2020	1	RESOLVED	0	0	0	1	0	0
20	3400314	20	Sporadic	6	St.Andrew	M2P	9	CONFIRMI	05-03-2020	07-03-2020	1	RESOLVED	0	0	0	0	0	0

Data Analysis



The dataset has around 98% as a resolved outcome, around 2% as a fatal outcome, and less than 1% as an active outcome. The Random Forest algorithm will mostly predict the outcome as resolved because of its higher percentage in the dataset, and so balancing is done on the dataset using SPARK.

Spark Code for Balancing and Implementing Random Forest Algorithm

```
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: Sun Aug  8 15:04:01 2021 from 35.235.244.34
keerthyraaj@spark-m:~$ ls
COVID-19.csv
keerthyraaj@spark-m:~$ hadoop fs -mkdir /BigData
mkdir: `/BigData': File exists
keerthyraaj@spark-m:~$ hadoop fs -copyFromLocal COVID-19.csv /BigData/.
keerthyraaj@spark-m:~$ spark-shell --master yarn

Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://spark-m.us-central1-a.c.zeta-serenity-317413.internal:44805
Spark context available as 'sc' (master = yarn, app id = application_1628429294921_0006).
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_292)
Type in expressions to have them evaluated.
Type :help for more information.
```

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

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}
val results = spark.read
 .format("csv")
 .option("header", "true")
 .load("hdfs://10.128.0.7:8020/BigData/COVID-19.csv")
val fatal = results.filter(col("Outcome") === "FATAL")
val resolved = results.filter(col("Outcome") === "RESOLVED")
val active = results.filter(col("Outcome") === "ACTIVE")
val resolved_active = resolved.unionAll(active)
val sampleRatio = fatal.count().toDouble / results.count().toDouble
val resolved_active_Sample = resolved_active.sample(false, sampleRatio)
fatal.unionAll(resolved_active_Sample)
val underSampleData = fatal.select(col("Outcome"),col("Source of Infection").cast(DoubleType),
 col("Currently Intubated").cast(DoubleType),
 col("Ever Hospitalized").cast(DoubleType),col("Ever in ICU").cast(DoubleType),
 col("Age Group").cast(DoubleType),col("Client Gender").cast(DoubleType))
val Array(trainingData, testData) = underSampleData.randomSplit(Array(0.8, 0.2), 754)
val indexer = new StringIndexer()
  .setInputCol("Outcome")
  .setOutputCol("Outcome_inx")
val assembler = new VectorAssembler()
 .setInputCols(Array("Currently Intubated","Source of Infection","Ever Hospitalized","Ever in ICU","Age Group","Client Gender"))
 .setOutputCol("assembled-features")
```

```
val rf = new RandomForestClassifier()
 .setFeaturesCol("assembled-features")
 .setLabelCol("Outcome_inx")
 .setSeed(1234)
val pipeline = new Pipeline()
.setStages(Array(indexer,assembler,rf))
val evaluator = new MulticlassClassificationEvaluator()
  .setLabelCol("Outcome_inx")
  .setPredictionCol("prediction")
  .setMetricName("accuracy")
 val paramGrid = new ParamGridBuilder()
  .addGrid(rf.maxDepth, Array(3, 5))
  .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)
val accuracy = evaluator.evaluate(predictions)
println("accuracy on test data = " + accuracy)

// Exiting paste mode, now interpreting.

accuracy on test data = 1.0
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}
results: org.apache.spark.sql.DataFrame = [_id: string, Assigned_ID: string ... 16 more fields]
fatal: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [_id: string, Assigned_ID: str...

scala>
```

REFERENCE

http://blog.madhukaraphatak.com/class-imbalance-part-3/