**Project -2**

Paresh Patil

**Contents**

* Abstract
* Introduction
* What kind of Data?
* Methodology
* Objectives
* Terminal Screen results
* Code
* References

**Abstract**

Coronaviruses are a type of virus that infects people. There are numerous different types, and some of them are disease-causing. SARS-CoV-2, a coronavirus discovered in 2019, has produced a pandemic of respiratory sickness known as COVID-19. This report analyse the dataset given by the city of Toronto on COVID19 cases to find some relativity between features. The main objective is to create a SparkML machine learning algorithm to predict one of the categories using the provided dataset.

**Introduction**

In the event of a pandemic, data and surveillance are critical. The information gathered can help us understand how the pandemic is progressing, estimate its severity and risk, and design the necessary interventions to stop it from spreading further.

In the midst of a global pandemic, Toronto Public Health is responding to an ongoing COVID-19 outbreak. Since the first case was reported in January 2020, this data collection provides demographic, geographic, and severity information for all confirmed and probable cases reported to and managed by Toronto Public Health.

**What kind of data?**

Table

Description automatically generated with medium confidence

* 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:- Age of the patient’s child to adult
* Neighbourhood Name:- Toronto is separated into 140 unique neighbourhoods that were created to assist government and community agencies with local planning by providing socioeconomic statistics for a specific geographic area.
* FSA:- Client postal codes are used to produce FSA values. A single FSA can cover numerous areas.
* Source of Infection:- Several data fields are examined to establish the most likely route that cases acquired their COVID-19 infection, including travel, household, community
* Classification:- The use of the provincial case definition to classify instances as confirmed or probable based on established criteria.
* Episode Date:- The episode date relates to the earliest possible date from: symptom onset, laboratory specimen collection date, or reported date, and is a derived variable that best estimates when the disease was acquired.
* Reported Date:- The Date on which the case was reported to Toronto public health
* Client Gender:- Gender as stated by the individual. Gender is a social classification system that divides people into two groups depending on their biological sex.
* Outcome: Fatal- critical, Resolved and, active cases.
* Currently Hospitalized:- Cases that are currently admitted to hospital (i.e., no discharge date reported).
* Currently in ICU:- Cases that are currently admitted to the intensive care unit (ICU) (i.e. no discharge date reported).

**Methodology**

Downloading this data set from the given URL and looking for NULL and Na values. Once the data is clean observed the numeric and categorical data. Check whether the data set is balanced or not. Once all the prerequisites are completed. It is good to apply the algorithm(Random forest). Using indexer and oneHotEncoder for categorized data. Furthermore, visualization is performed in tableau for better understanding of data.

**Objectives**

* What is the fatality ratio?

A picture containing timeline

Description automatically generated

* How many active cases, resolved cases, and fatal cases for different gender?

Graphical user interface, text

Description automatically generated

* How many people have ever been hospitalized based on different gender?

Graphical user interface, application, Teams

Description automatically generated

* How many cases in every year with different sources of infection?

Chart, bar chart

Description automatically generated

* What is the fatal ratio according to age group?

A picture containing table

Description automatically generated

* Which age group is more hospitalized?

Chart, bar chart

Description automatically generated

Code screenshots:

1. Reading the dataset from hdfs and removing the null values and display 10 records.

Text

Description automatically generated

1. Showing schema

Text

Description automatically generated

1. Checking the dataset is balanced or not

Text

Description automatically generated

1. Balancing the dataset

Text

Description automatically generated

1. Implementing the indexing

Text

Description automatically generated

A picture containing table

Description automatically generated

1. OneHotEncoder

Text

Description automatically generated

1. Random forest algorithm

Text

Description automatically generated

1. Prediction and accuracy

Text

Description automatically generated

**Code**

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.evaluation.{RegressionEvaluator}

import org.apache.spark.ml.param.ParamMap

import org.apache.spark.sql.types.{DoubleType}

import org.apache.spark.ml.feature.StringIndexer

import org.apache.spark.sql.types.DataType

import org.apache.spark.sql.types.NumericType

import org.apache.spark.sql.types.IntegerType

import org.apache.spark.ml.feature.OneHotEncoder

import org.apache.spark.ml.classification.RandomForestClassifier

import org.apache.spark.ml.classification.Classifier

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

import org.apache.spark.ml.evaluation.RegressionEvaluator

import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator

import org.apache.spark.ml.Estimator

val df1=spark.read.format("csv").option("header","true").load("hdfs://10.128.0.5:8020/BigData/covid19.csv")

val cleandf1 = df1.na.drop()

cleandf1.show(10)

cleandf1.schema(10)

cleandf1.printSchema()

val dataset=cleandf1.select(col("Outcome"),

col("Age Group"),col("Ever Hospitalized"),col("Ever in ICU")

,col("Client Gender"),col("Source of Infection")).filter(cleandf1("Classification")==="CONFIRMED")

dataset.count()

dataset.groupBY("Outcome","Age Group","Client Gender","Source of Infection").count().show()

dataset.groupBy("Outcome").count().show()

dataset.groupBy("Age Group").count().show()

dataset.groupBy("Client Gender").count().show()

dataset.groupBy("Source of Infection").count().show()

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

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

val ratiodf=fataldf.count().toDouble/dataset.count()

val resolvedsampledf = resolveddf.sample(false,ratiodf)

val balanceddf = fataldf.unionAll(resolvedsampledf)

balanceddf.groupBy("Outcome").count().show()

val inputCols = Array("Age Group","Client Gender","Source of Infection")

val outputCols=Array("Age\_index","Gender\_index","SOI\_index")

val indexer = new StringIndexer().setInputCols(inputCols).setOutputCols(outputCols)

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

val df\_indexed=indexer.fit(balanceddf).transform(balanceddf)

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("Gender\_index").cast(IntegerType),col("SOI\_index").cast(IntegerType))

rankdf.show(10)

val encoder=new OneHotEncoder().setInputCols(Array("Age\_index","Gender\_index","SOI\_index"))

.setOutputCols(Array("Age\_vector","Gender\_vector","SOI\_vector"))

val df\_encoder=encoder.fit(rankdf).transform(rankdf)

df\_encoder.show(10)

#RandomForestClassifier

val Array(train, test) = df\_encoder.randomSplit(Array(0.7,0.3),720)

train.count()

test.count()

val assembler=new VectorAssembler().setInputCols(Array("Age\_vector","Gender\_vector","SOI\_vector",

"Age\_index","Gender\_index","SOI\_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 model1=pipeline.fit(train)

val predict = model1.transform(test)

val evaluator=new MulticlassClassificationEvaluator()

.setLabelCol("Outcome\_index")

.setPredictionCol("prediction")

.setMetricName("accuracy")

val accuracy = evaluator.evaluate(predict)

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

**References**

* *Open data dataset*. City of Toronto Open Data Portal. (n.d.). Retrieved April 22, 2022, from [https://open.toronto.ca/dataset/covid-19-cases-in-toronto/](https://open.toronto.ca/dataset/covid-19-cases-in-toronto/%20)
* Sample Assignment Retrieved April 22, 2022, from [https://conestoga.desire2learn.com/d2l/le/content/535415/Home?itemIdentifier=D2L.LE.Content.ContentObject.ModuleCO-12000774](https://conestoga.desire2learn.com/d2l/le/content/535415/Home?itemIdentifier=D2L.LE.Content.ContentObject.ModuleCO-12000774%20)
* Tutorial points <https://www.tutorialspoint.com/pyspark/index.htm>