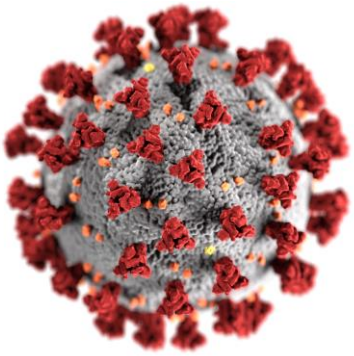


TORONTO COVID CASES FATALITY PREDICTOR

Using a Random Forest Classifier with Spark

Naveen Selvaraju
Data Collection and Curation

1 INTRODUCTION

Toronto Public Health is responding to an ongoing COVID-19 outbreak, in the context of an evolving global pandemic. This report has the findings of a classification algorithm that has been trained on the covid cases dataset to predict the possible fatalities of this ongoing pandemic. The model takes into consideration of the patient's age group, gender, outbreak associated, source of infection and past medical history to predict the probability of fatality for an active patient. The dataset is obtained from Toronto open data catalog.

2 DATASET

This data set contains demographic, geographic, and severity information for all confirmed and probable cases reported to and managed by Toronto Public Health since the first case was reported in January 2020. This includes cases that are sporadic (occurring in the community) and outbreak associated. The data are extracted from the provincial Case & Contact Management System. The data will be completely refreshed and overwritten on a weekly basis.

Outbreak Associated	Age Group	Neighbourhood Name	FSA	Source of Infection	Classification	Episode Date	Reported Date
Sporadic	19 and younger	West Hill	M1E	Close Contact	CONFIRMED	2021-03-27	2021-04-01
Sporadic	19 and younger	West Hill	M1E	Close Contact	CONFIRMED	2021-03-26	2021-04-01
Sporadic	40 to 49 Years	West Hill	M1E	Close Contact	CONFIRMED	2021-03-27	2021-04-01

Figure 1: Dataset Preview

3 DATASET FEATURES

Column	Description
_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 (e.g. long-term care homes,

	retirement homes, hospitals, etc.) and other Toronto congregate settings (such as homeless shelters).
Age Group	Age at time of illness. Age groups (in years): ≤19, 20-29, 30-39, 40-49, 50-59, 60-69, 70-79, 80-89, 90+, unknown (blank)
Neighbourhood Name	Toronto is divided into 140 geographically distinct neighborhoods that were established to help government and community agencies with local planning by providing socio-economic data for a meaningful geographic area. Please note that client postal code information is mapped to the most up-to-date census tract (CT) and neighbourhood information available from the city. As a result, neighbourhood information is not available for those with missing postal code or when postal code could not be mapped/linked to a CT.
FSA	Forward sortation area (i.e. first three characters of postal code) based on the case's primary home address. FSA values are generated from client postal codes. One FSA can span multiple neighbourhoods.
Source of Infection	The most likely way that cases acquired their COVID-19 infection is determined by examining several data fields including: • A public health investigator's assessment of the most likely source of infection. • Being associated with a confirmed COVID-19 outbreak • Reported risk factors such as contact with a known case or travel If the public health investigator's assessment is absent, then the other data fields are used to infer source of acquisition using the following hierarchy: Cases with episode dates before April 1 2020: • Travel > Outbreak (settings described below) > Household Contact > Close Contact > Community > No information Cases with episode dates on or after April 1 2020: • Outbreak (settings described below) > Household Contact > Close Contact > Travel > Community > No information. Descriptions: • **Household contact**: Case who acquired infection from a household contact with a confirmed or probable COVID-19 case (e.g. family member, roommate). • **Close contact with a case**: Case who acquired infection from a close contact with a confirmed or probable COVID-19 case (e.g. co-worker). • **Outbreaks**: Cases linked to known confirmed COVID-19 outbreaks. These could include the index case who may have acquired the infection elsewhere. Outbreaks settings include: • *Outbreaks, Congregate Settings*: confirmed outbreaks in Toronto in shelters, correctional facilities, group homes, or other congregate settings such as hostels or rooming houses.

	○ *Outbreaks, Healthcare Institutions*: confirmed outbreaks in Toronto in long-term care homes, retirement homes, hospitals, chronic care hospitals, or other institutional settings. ○ *Outbreaks, Other Settings*: confirmed outbreaks in Toronto in workplaces, schools, day cares, or outbreaks outside of Toronto. We do not validate outbreaks that occur in other health units, as such these cases may not be linked to confirmed outbreaks. • **Travel**: Case that travelled outside of Ontario in the 14 days prior to their symptom onset or test date, whichever is the earliest. • **Community**: Cases who did not travel outside of Ontario, did not identify being a close contact with a COVID-19 case, and were not part of a known confirmed COVID-19 outbreak. • **No information**: Cases with no information on the source of infection
Classification	The application of the provincial case definition to categorize the cases as confirmed or probable, according to standard criteria. Please refer to the Ontario Ministry of Health website for Ontario's
Episode Date	The episode date is a derived variable that best estimates when the disease was acquired and refers to the earliest available date from: symptom onset (the first day that COVID-19 symptoms occurred), laboratory specimen collection date, or reported date.
Reported Date	The date on which the case was reported to Toronto Public Health.
Client Gender	Self-reported gender. Gender is a system that operates in a social context and generally classifies people based on their assigned biological sex.
Outcome	**Fatal:** Cases with a fatal outcome reported. **Resolved:** Cases not reported as deceased, and who are either: • Reported as 'recovered' OR • Where the report date is more than 14 days from symptom onset AND the case is not currently hospitalized. This number is underreported due to a lag in data entry. **Active:** All other 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).
Currently Intubated	Cases that were intubated related to their COVID-19 infection (includes cases that are currently intubated and those that have been discharged or deceased).
Ever Hospitalized	Cases that were hospitalized related to their COVID-19 infection (includes cases that are currently hospitalized and those that have been discharged or are deceased).
Ever in ICU	Cases that were admitted to the intensive care unit (ICU) related to their COVID-19 infection (includes cases that are currently in ICU and those that have been discharged or are deceased).
Ever Intubated	Cases that were intubated related to their COVID-19 infection (includes cases that are currently intubated and those that have been discharged or deceased)

4 EXPLORATORY DATA ANALYSIS

The first step of training a classification algorithm is exploratory data analysis, where the structure and composition of the dataset is analyzed, in this section, we will go over the exploratory data analysis steps taken on the dataset,

```
df.head()
```

	_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
0	771963	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
1	771964	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
2	771965	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
3	771966	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
4	771967	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

Figure 2: The dataset

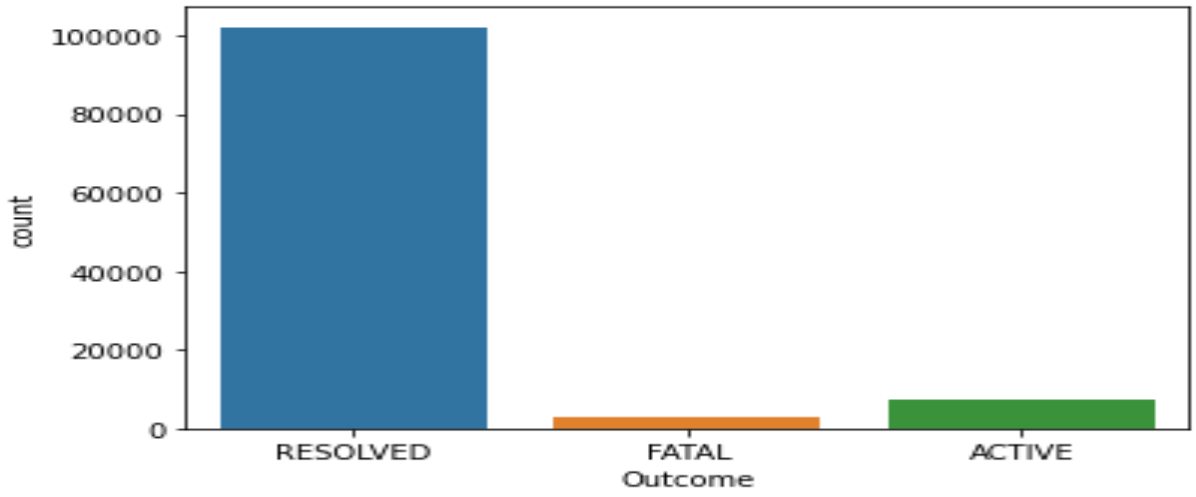


Figure 3: Outcome column distribution

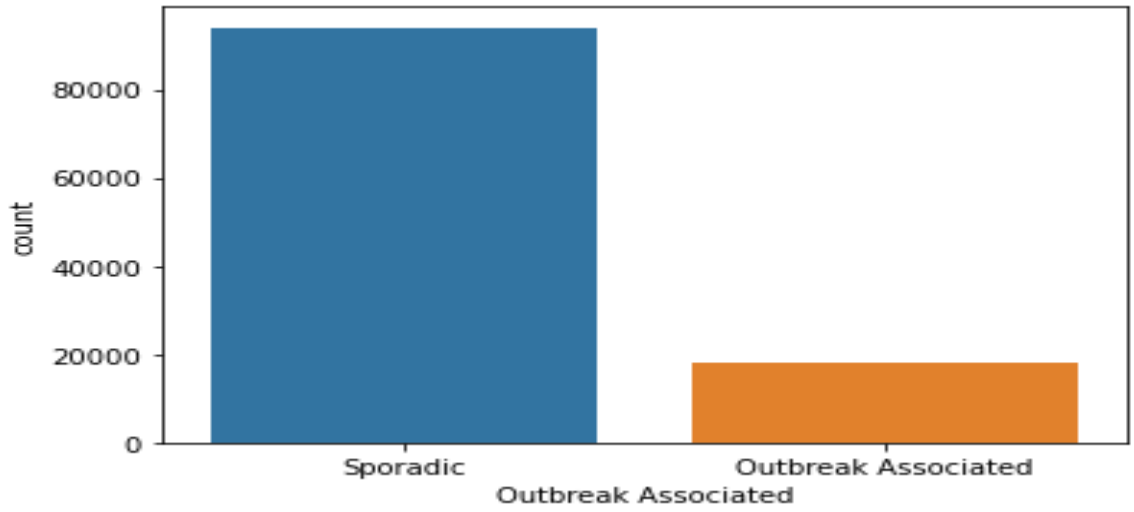


Figure 4: Outbreak Associated distribution

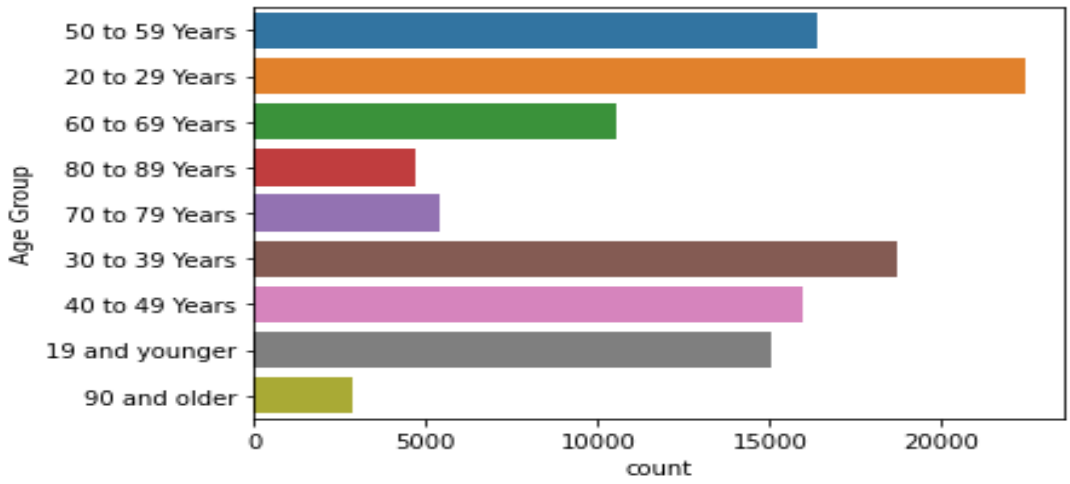


Figure 5: Age Group distribution

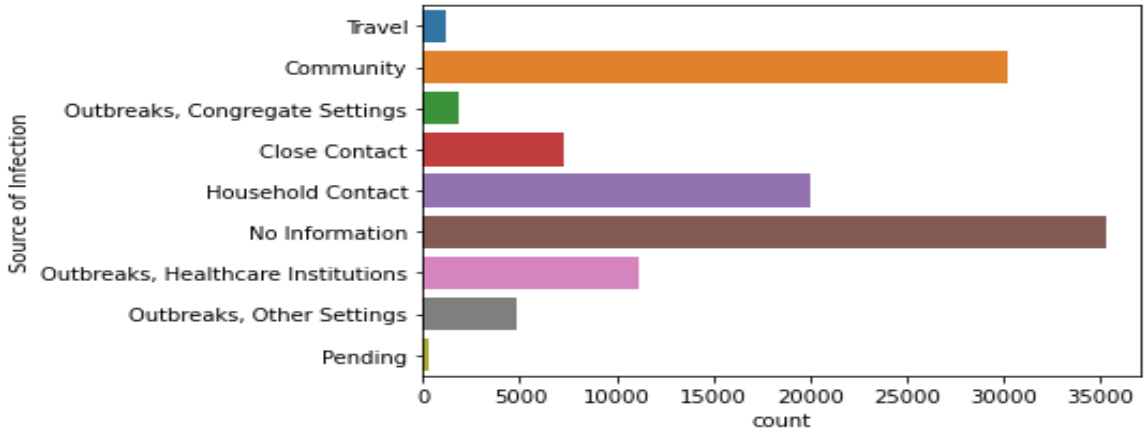


Figure 6: Source of Infection

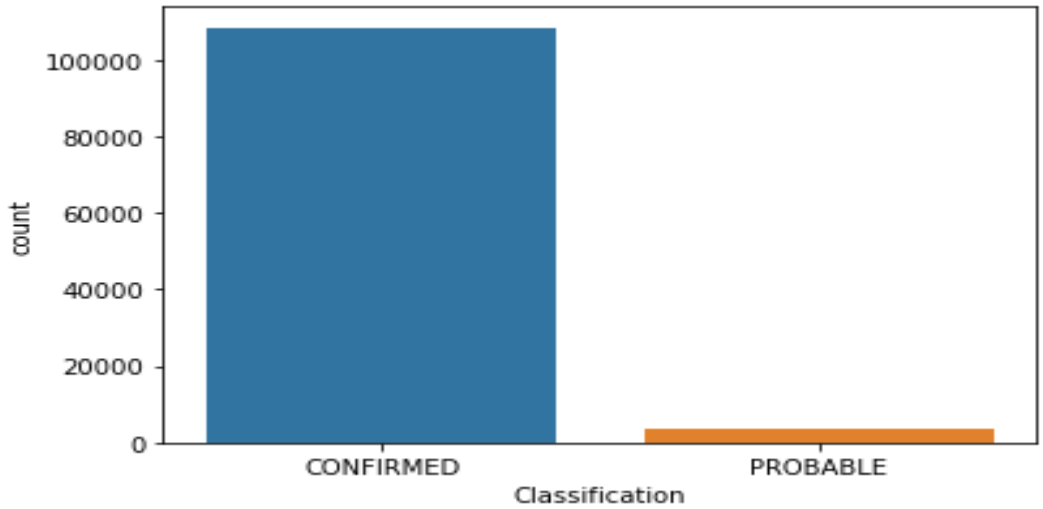


Figure 7: Classification

5 Training Random Forest Classifier

Next, the dataset is loaded into a Hadoop cluster in GCP data proc and the data is cleaned and engineered to extract new features and trained with a Random Forest classifier,

5.1 Importing Libraries

All libraries required to loading, cleaning and training and testing the dataset are imported into jupyter lab context.

```
import org.apache.spark.sql.functions._
import org.apache.spark.sql.DataFrame
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.BinaryClassificationEvaluator
import org.apache.spark.ml.param.ParamMap
import org.apache.spark.sql.types.{IntegerType, DoubleType}
```

```
Intitializing Scala interpreter ...
Spark Web UI available at http://dcc-cluster-m:8088/proxy/application_1618078927905_0005
SparkContext available as 'sc' (version = 3.1.1, master = yarn, app id = application_1618078927905_0005)
SparkSession available as 'spark'
import org.apache.spark.sql.functions._
import org.apache.spark.sql.DataFrame
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.BinaryClassificationEvaluator
import org.apache.spark.ml.param.ParamMap
import org.apache.spark.sql.types.{IntegerType, DoubleType}
```

5.2 Load Dataset

Load the dataset from the Hadoop cluster into the spark context as Scala data frame,

```
val data = spark.read
 .format("csv")
 .option("header", "true")
 .load("hdfs://10.128.0.8:8020/BigData/COVID19_cases.csv")
```

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

5.3 Split Active and Past Cases

The data frame is divided into two as active and past cases where the past cases are used to train and test the model and predict the probability of fatality in the active cases,

```
def getActiveCases(dataset: DataFrame): DataFrame = {
    dataset.filter($"Outcome" === "ACTIVE")
}
def getPastCases(dataset: DataFrame): DataFrame = {
    dataset.filter($"Outcome" !== "ACTIVE")
}
```

```
getActiveCases: (dataset: org.apache.spark.sql.DataFrame)org.apache.spark.sql.DataFrame
getPastCases: (dataset: org.apache.spark.sql.DataFrame)org.apache.spark.sql.DataFrame
```

```
val active_df = getActiveCases(data)
val data_df = getPastCases(data)
```

```
active_df: org.apache.spark.sql.DataFrame = [_id: string, Assigned_ID: string ... 16 more fields]
data_df: org.apache.spark.sql.DataFrame = [_id: string, Assigned_ID: string ... 16 more fields]
```

```
data_df.select($"Outcome").distinct.show()
```

```
+--------+
| Outcome|
+--------+
|   FATAL|
|RESOLVED|
+--------+
```

5.4 CLEANING THE DATASET

Next, the dataset is cleaned using the following steps,

1. The null values in the data frame are removed.
2. The integer and date fields are casted from string values.
3. A new feature “Days Since Episode” is created from fields “Reported Date” and “Episode Date”.
4. Select the feature that will be used to train the model.

```
def cleanDataset(dataset: DataFrame): DataFrame = {
    //Remove Null values
    val data_clean = dataset.na.drop(Seq("Age Group"))

    //Cast dataset
    val df_cast = data_clean.withColumn("Episode Date", to_date($"Episode Date", "yyyy-MM-dd"))
                .withColumn("Reported Date", to_date($"Reported Date", "yyyy-MM-dd"))
                .withColumn("_id",  $"_id".cast("int"))
                .withColumn("Assigned_ID",  $"Assigned_ID".cast("int"))

    //Generate Days Since Episode column from Reported Date and Episode Date
    val df_clean = df_cast.withColumn("Days Since Episode", datediff($"Reported Date", $"Episode Date"))
                .where($"Days Since Episode" >= 0)

    //Select features to train the model
    val df_select = df_clean.select("Outbreak Associated", "Age Group", "Source of Infection", "Classification", "Client Gender", "Outcome", "Days Since Episode",
                    "Currently Hospitalized", "Currently in ICU", "Currently Intubated", "Ever Hospitalized", "Ever in ICU", "Ever Intubated")
    df_select
}
```

```
cleanDataset: (dataset: org.apache.spark.sql.DataFrame)org.apache.spark.sql.DataFrame
```

5.5 Feature Engineering

The categorical features in the data frame are converted into numerical variables using the StringIndexer class,

```
def featureEngg(dataset: DataFrame): DataFrame = {
    val indexer = new StringIndexer()
  .setInputCols(Array("Outbreak Associated", "Age Group", "Source of Infection", "Classification", "Client Gender", "Outcome",
                  "Currently Hospitalized", "Currently in ICU", "Currently Intubated", "Ever Hospitalized", "Ever in ICU", "Ever Intubated"))
  .setOutputCols(Array("outbreakAssociatedIdx", "ageGroupIdx", "sourceOfInfectionIdx", "classificationIdx", "clientGenderIdx",
                   "outcomeIdx", "currentlyHospitalizedIdx", "currentlyInICUIdx", "currentlyIntubatedIdx", "everHospitalizedIdx", "everInICUIdx", "everIntubatedIdx"))

    val df = indexer.fit(dataset).transform(dataset)

    val select_df = df.select("outbreakAssociatedIdx", "ageGroupIdx", "sourceOfInfectionIdx", "classificationIdx", "clientGenderIdx", "Days Since Episode",
                   "outcomeIdx", "currentlyHospitalizedIdx", "currentlyInICUIdx", "currentlyIntubatedIdx", "everHospitalizedIdx", "everInICUIdx", "everIntubatedIdx")
    select_df
}
```

```
featureEngg: (dataset: org.apache.spark.sql.DataFrame)org.apache.spark.sql.DataFrame
```

5.6 Training a Random Forest classifier

```
def trainWithRandomForest(dataset: DataFrame): CrossValidatorModel = {

    //Split dataset into Train and Test datasets
    val Array(train_df, test_df) = dataset.randomSplit(Array(0.8, 0.2))

    //VectorAssembler to merge multiple features into single feature
    val assembler = new VectorAssembler()
     .setInputCols(Array("outbreakAssociatedIdx", "ageGroupIdx", "sourceOfInfectionIdx", "classificationIdx", "clientGenderIdx", "Days Since Episode",
                    "currentlyHospitalizedIdx", "currentlyInICUIdx", "currentlyIntubatedIdx", "everHospitalizedIdx", "everInICUIdx", "everIntubatedIdx"))
     .setOutputCol("assembled-features")

    //Train a classification model using RandomForestClassifier
    val rf = new RandomForestClassifier()
     .setFeaturesCol("assembled-features")
     .setLabelCol("outcomeIdx")

    //pipeline for assembler and classifier
    val pipeline = new Pipeline()
      .setStages(Array(assembler, rf))

    //BinaryClassificationEvaluator to find the accuracy of the trained model
    val evaluator = new BinaryClassificationEvaluator()
      .setLabelCol("outcomeIdx")

    //ParamGridBuilder to build different possible combination of parameters
    val paramGrid = new ParamGridBuilder()
      .addGrid(rf.maxDepth, Array(3, 5))
      .addGrid(rf.impurity, Array("entropy","gini")).build()
```

```
    //K-Fold cross validation
    val cross_validator = new CrossValidator()
      .setEstimator(pipeline)
      .setEvaluator(evaluator)
      .setEstimatorParamMaps(paramGrid)
      .setNumFolds(3)

    //Fit training data to the classification model
    val cvModel = cross_validator.fit(train_df)

    //Find predictions with test dataset
    val predictions = cvModel.transform(test_df)

    //Find the accuracy of predicated values
    val accuracy = evaluator.evaluate(predictions)

    println("accuracy = " + accuracy)

    cvModel
}
```

```
val clean_df = cleanDataset(data_df)
val df_fengg = featureEngg(clean_df)
val trainedModel = trainWithRandomForest(df_fengg)
```

```
accuracy = 0.9704688660060319
clean_df: org.apache.spark.sql.DataFrame = [Outbreak Associated: string, Age Group: string ... 11 more fields]
df_fengg: org.apache.spark.sql.DataFrame = [outbreakAssociatedIdx: double, ageGroupIdx: double ... 11 more fields]
trainedModel: org.apache.spark.ml.tuning.CrossValidatorModel = CrossValidatorModel: uid=cv_a86367a9479e, bestModel=pipeline_5956aa600784, numFolds=3
```

5.7 Save The Trained Model

```
trainedModel.write.overwrite()
  .save("/models/covid-model")
```

5.8 Load The Trained Model

```
val cvModelLoaded = CrossValidatorModel
  .load("/models/covid-model")
```

```
cvModelLoaded: org.apache.spark.ml.tuning.CrossValidatorModel = CrossValidatorModel: uid=cv_a86367a9479e, bestModel=pipeline_5956aa600784, numFolds=3
```

5.9 The Active cases data frame is Cleaned and Engineered

```
val active_df_clean = cleanDataset(active_df)
val active_fengg_df = featureEngg(active_df_clean)
```

```
active_df_clean: org.apache.spark.sql.DataFrame = [Outbreak Associated: string, Age Group: string ... 11 more fields]
active_fengg_df: org.apache.spark.sql.DataFrame = [outbreakAssociatedIdx: double, ageGroupIdx: double ... 11 more fields]
```

5.10 The Fatalities In Active Cases are Predicted using The Trained Model

```
val active_df_prediction = cvModelLoaded.transform(active_fengg_df)
```

```
active_df_prediction: org.apache.spark.sql.DataFrame = [outbreakAssociatedIdx: double, ageGroupIdx: double ... 15 more fields]
```

```
active_df_prediction.select($"prediction", $"probability").show()
```

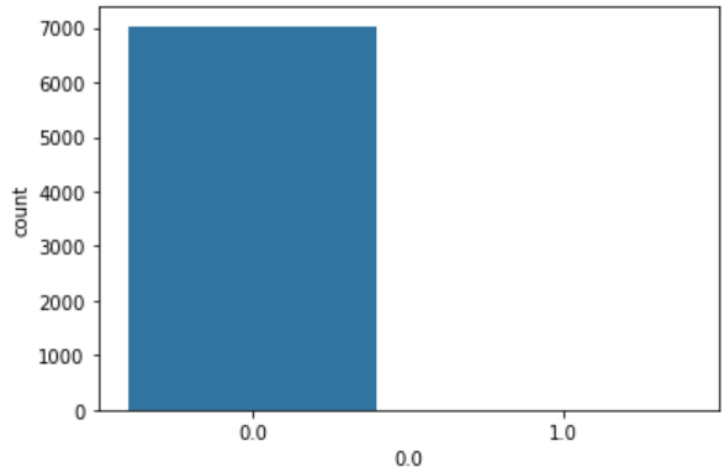
```
+----------+--------------------+
|prediction|         probability|
+----------+--------------------+
|       0.0|[0.86337569193441...|
|       0.0|[0.85870555623947...|
|       0.0|[0.93678559737393...|
|       0.0|[0.93678559737393...|
|       0.0|[0.93678559737393...|
|       0.0|[0.94119980625711...|
|       0.0|[0.86337569193441...|
|       0.0|[0.90504146547444...|
|       0.0|[0.93607833256672...|
|       0.0|[0.93756016906094...|
|       1.0|[0.39136161601867...|
|       0.0|[0.93678559737393...|
|       0.0|[0.93607833256672...|
|       0.0|[0.90874840955042...|
|       1.0|[0.47694540844891...|
|       1.0|[0.42152218100542...|
|       0.0|[0.71434456097648...|
|       0.0|[0.83355446168685...|
|       0.0|[0.69404949430676...|
|       0.0|[0.83241859623954...|
+----------+--------------------+
only showing top 20 rows
```

```
active_df_prediction.select($"prediction")
.write.format("csv").save("hdfs://10.128.0.8:8020/BigData/active_predict.csv")
```

```
predict_df = pd.read_csv('part-00001-bf23655d-ed2b-42f6-b207-37985b2b8c5c-c000.csv')
```

```
sns.countplot(data=predict_df, x="0.0")
```

```
<AxesSubplot:xlabel='0.0', ylabel='count'>
```




```
0.0    7031
1.0       3
Name: predictions, dtype: int64
```

6 CONCLUSION

The trained model has 97% accuracy, when tested with a list of active cases the model returned three fatalities out of 7034 active cases.