Abstract

Coronaviruses are a large group of viruses known to cause the common cold and even more serious illnesses, such as Middle East Respiratory syndrome (MERS) and Severe acute Respiratory syndrome (SARS).mNovel Coronavirus (CoVID-19) is a kind of novel coronavirus that has not been found in humans before 2019.To date, the total number of confirmed cases worldwide has exceeded 18 million. Seven hundred thousand people die of the disease, and the number is rising. How to control COVID-19 and reduce mortality has become a top priority for countries around the world. We now have data sets from Toronto on COVID-19 cases, and we want to analyze these data sets through the KDD process to develop models that actually mitigate the impact of COVID-19 globally and protect specific populations.

Key words: COVID-19 cases, Toronto, KDD process

目录

1.1 Identify the objectives of the business

1.1.1 Background

As a known large group of viruses, coronavirus can cause the common cold and even more serious diseases, such as the Middle East respiratory syndrome (MERS) and severe acute respiratory syndrome (SARS). The novel coronavirus (COVID-19) is a new coronavirus which has not been found in humans before 2019.

The World Health Organization (who) defines coronavirus disease as a pandemic. So far, the total number of confirmed cases in the world has exceeded 18 million. 700000 people have died of the disease, and the number is still on the rise. It not only has a profound impact on human health, but also has an indelible impact on the global economic and social environment. How to curb the novel coronavirus pneumonia and reduce mortality has become the primary problem to be solved.

1.1.2 Business objectives

（1）Analyze the impact of age, gender, source of infection, outbreak associated, hospitalization status. Identify key factors and characteristics that contribute to survive in patients with COVID-19.

（2）Control these characteristics and factors that lead to death in patients with COVID-19, and guide the country, institution or family to locate groups that are more likely to survive from COVID-19.

（3）Protect those at risk for COVID-19 by pre-positioning and taking preventive measures.

（4）Reduce national or regional mortality from COVID-19 through prevention.

1.1.3 Business success criteria

(1) Significant increasing in survival rate of novel coronavirus pneumonia. Obtain characteristics of COVID-19 cases, including sex, age, source of infection, hospital status, etc. Implementing targeted prevention, and successfully improve COVID-19 survival rate.

(2) The project can be completed on time without exceeding the budget.

1.2 Assess the situation

1.2.1 Assumption

(1) Survival rate for COVID-19 patients was associated with age. Generally speaking, young children are at the greatest risk of infection. For example, approximately 57% of malaria occurs in children under 5 years of age. However, in the face of the new coronavirus, the elderly are the most at risk. This may be because older people have potential health problems, especially cardiovascular diseases and respiratory diseases. The elderly are more likely to have these health problems than the young, which may be one of the important reasons why the elderly are not likely to survive the risk of COVID-19.

(2) The gendered impact on health outcomes. In many cases, since most of the world's health workers are women, women seem to be more likely to be diagnosed with covid-19. At the same time, compared with women, the male mortality rate in each country has maintained a higher growth trend. This may be due to the fact that men have a higher smoking rate and are more likely to suffer from cardiopulmonary diseases.

(3) Whether or not a person is treated, the level of treatment and the patient's ability to recover from the disease all affect whether or not he will die of a disease.

1.2.2 Constraints

(1) The source of data is single. The data of COVID-19 cases for this study are from Toronto Public Health in January 2020. The loss of some data will still cause certain errors in the overall statistics of outcome data.

(2) In order to protect personal privacy, some more detailed data cannot be obtained, and the lack of critical information may skew the overall objectivity of the results.

1.3 Determine data mining objectives

1.3.1 Data mining goals

(1) Get a set of decision rules that determine the survival rate of COVID-19.

(2) Use decision rules to predict or classify the COVID- 19 survival rate of a person or group of people in the future

1.3.2 Data mining success criteria

(1) Model quality

More than 85% accuracy; Faster response speed; The output is easy to understand

(2) Engineering dimension

Flexible model; easy to use; tight layout, embeddable and extensible.

(3) Logistical constraints

Simple calculation; less development time.

1.4 Produce a project plan

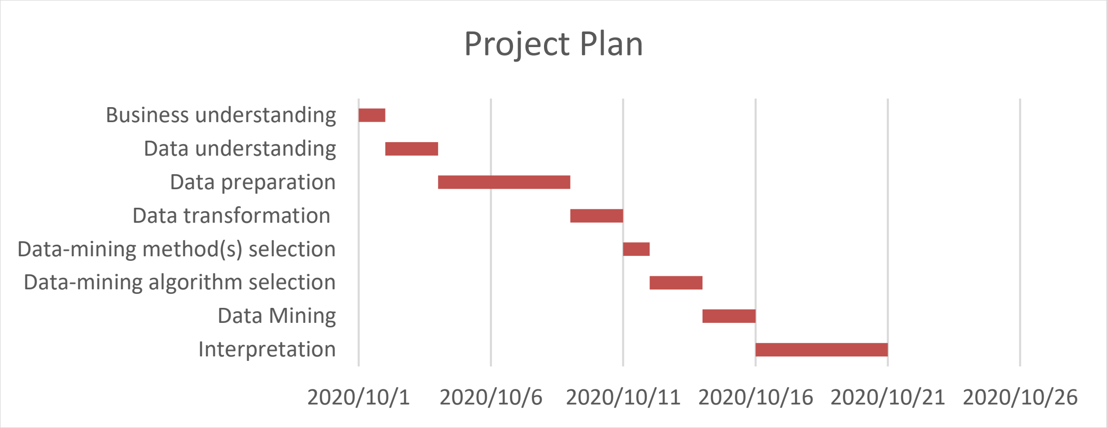
1.4.1 Project Plan Overview

The whole project will take about a month, which is divided into eight steps. Data preparation and interpraction accounted for the largest proportion. This is because data preparation is the basis of the following steps. Detailed data preparation can help us speed up the data transformation, the selection of methods and algorithms, and data mining. The interpraction step requires us to evaluate and review the previous seven steps, which will affect the accuracy and relevance of the modeling.

Note : Modeling is a phase in which multiple iterations usually occur.

The overview plan for the study is as shown in the table below.

Table 1.1 project plan overview



1.4.2 Resources

Because the data set is simple and the project needs less resources, I only need my time and myself as the resources for data analysis.

1.4.3 Risks

There will be some risks in some stages of the project, which will affect the progress, quality and results of the project.

Table 1.2 project plan risk

Task	Days	Start Date	Risk
Business understanding	1	2020/10/1	Economic change
Data understanding	2	2020/10/2	Data problems, technology problems
Data preparation	5	2020/10/4	Data problems, technology problems
Data transformation	2	2020/10/9	Data problems, technology problems
Data-mining method(s) selection	1	2020/10/11	Technology problems, inability to find adequate model
Data-mining algorithm selection	2	2020/10/12	Technology problems, inability to find adequate model

Data Mining	2	2020/10/14	Technology problems, inability to find adequate model
Interpretation	5	2020/10/16	Economic change, inability to implement results

2.1 Collect initial data

2.1.1 Source: Kaggle website

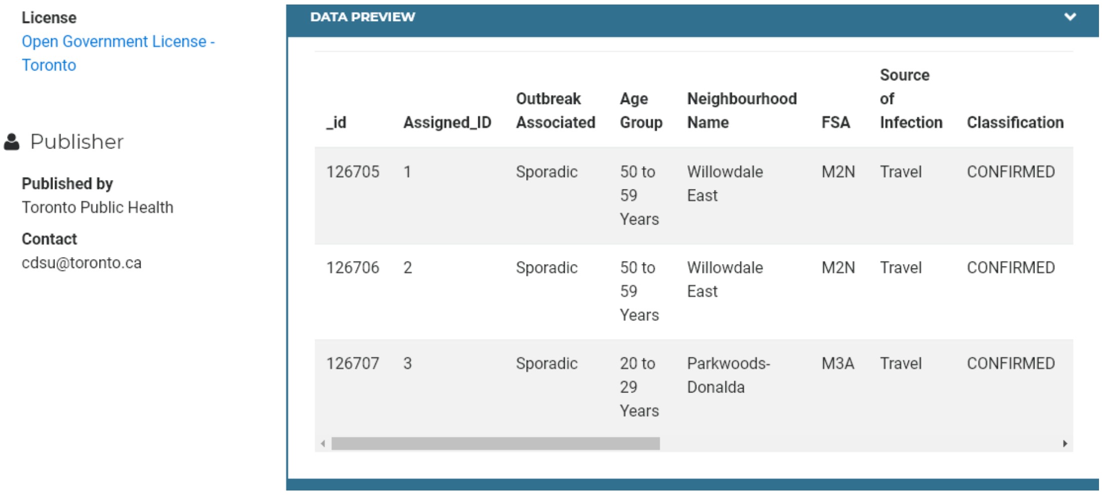
https://www.kaggle.com/divyansh22/toronto-covid19-cases

2.1.2 Original source:

Toronto Public Health

Collection methodology: The data was collected published by Toronto Public Health under Open Government License - Toronto

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



License
Open Government License - Toronto

Publisher

Published by
Toronto Public Health

Contact
cdsu@toronto.ca

DATA PREVIEW

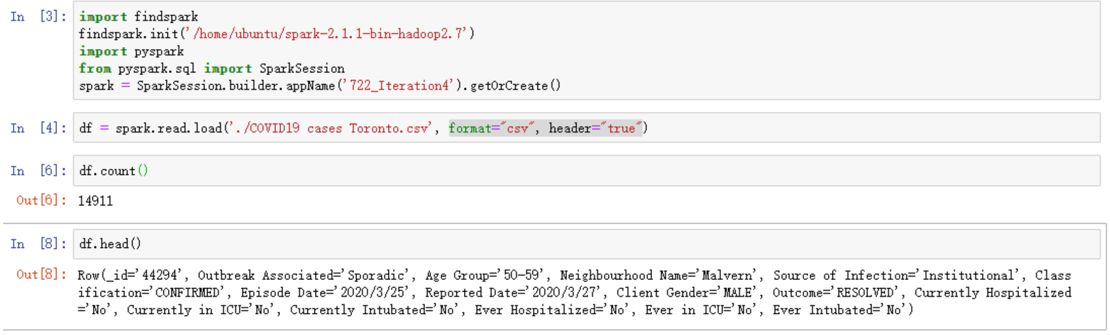
_id	Assigned_ID	Outbreak Associated	Age Group	Neighbourhood Name	FSA	Source of Infection	Classification
126705	1	Sporadic	50 to 59 Years	Willowdale East	M2N	Travel	CONFIRMED
126706	2	Sporadic	50 to 59 Years	Willowdale East	M2N	Travel	CONFIRMED
126707	3	Sporadic	20 to 29 Years	Parkwoods-Donalda	M3A	Travel	CONFIRMED

figure 1. About COVID-19 Cases in Toronto

2.2 Describe the Data

2.2.1 Data quantity

Amount of data: 14911 records and 16 attributes, the data used in this analysis is to record the COVID- 19 survival rate of different populations.



```
In  [3]: import findspark
         findspark.init('/home/ubuntu/spark-2.1.1-bin-hadoop2.7')
         import pyspark
         from pyspark.sql import SparkSession
         spark = SparkSession.builder.appName('722_Iteration4').getOrCreate()

In  [4]: df = spark.read.load('./COVID19 cases Toronto.csv', format="csv", header="true")

In  [6]: df.count()

Out[6]: 14911

In  [8]: df.head()

Out[8]: Row(_id='44294', Outbreak Associated='Sporadic', Age Group='50-59', Neighbourhood Name='Malvern', Source of Infection='Institutional', Class
        ification='CONFIRMED', Episode Date='2020/3/25', Reported Date='2020/3/27', Client Gender='MALE', Outcome='RESOLVED', Currently Hospitalized
        ='No', Currently in ICU='No', Currently Intubated='No', Ever Hospitalized='No', Ever in ICU='No', Ever Intubated='No')
```

2.2.2 Data Quality

(1) Relevant Attributes: _id, Outbreak Associated, Age Group, Neighbourhood Name, Client Gender, Classification, Source of Infection, Episode Date, Reported Date, Outcome, Currently Hospitalized, Currently in ICU, Currently Intubated, Ever Hospitalized, Ever in ICU, Ever Intubated

Column	Description
_id	Unique row identifier for Open Data database

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 groups (in years): ≤19, 20-29, 30-39, 40-49, 50-59, 60-69, 70-79, 80-89, 90+, unknown (blank)
Neighbourhood Name	To help government and community agencies with local planning, Toronto is divided into 140 geographically diverse communities. This is about the specific location of the case
Source of Infection	The most likely routes of covid-19 infection may include: travel, close contact with a case, institutional setting, healthcare setting, community, unknown / missing, Pending（Information on source of infection pending）, N/A

	（Outbreak-associated cases）
Classification	The provincial case definition is applied to classify the cases as confirmed or possible cases according to the standards.
Episode Date	The date of onset is a derivative variable, which refers to the onset of symptoms (the first day of covid-19 symptoms), the date of collection of laboratory specimens or the date of reporting.
Reported Date	The date on which the case was reported to Toronto Public Health.
Client Gender	People are classified according to the specified physiological sex.
Outcome	Fatal: Cases with a fatal outcome reported. Resolved: Cases with no reported deaths, and those reported as

	"recovered" or reported more than 14 days after the onset of symptoms, and the case is not currently hospitalized. Active: All other cases
Currently Hospitalized	Cases that are currently admitted to hospital
Currently in ICU	Cases that are currently admitted to the intensive care unit (ICU)
Currently Intubated	Cases that were intubated related to their COVID-19 infection
Ever Hospitalized	Cases that were hospitalized related to their COVID-19 infection
Ever in ICU	Cases that were admitted to the intensive care unit (ICU) related to their COVID-19 infection.
Ever Intubated	Cases that were intubated related to their COVID-19 infection

(3) Data type

1. Basic data of the people:

_id: numeric

Age Group: categorical (string)

Client Gender: categorical (string)

Neighbourhood Name: categorical (string)

2. Basic data on virus infection

Outbreak Associated: categorical (string)

Classification: categorical (string)

Source of Infection: categorical (string)

Episode Date: timestamp（string）

Reported Date: timestamp （string）

3. Basic data of severity information

Outcome: flag(string)

Currently Hospitalized: flag (string)

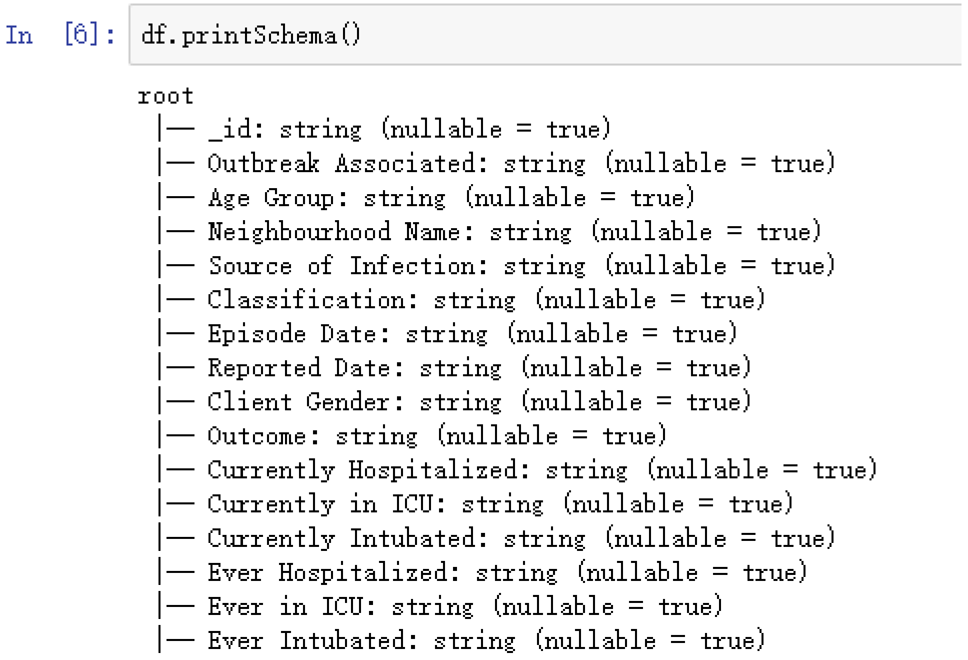
Currently in ICU: flag (string)

Currently Intubated: flag (string)

Ever Hospitalized: flag (string)

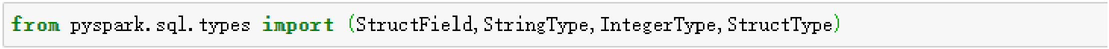
Ever in ICU: flag (string)

Ever Intubated: flag (string)

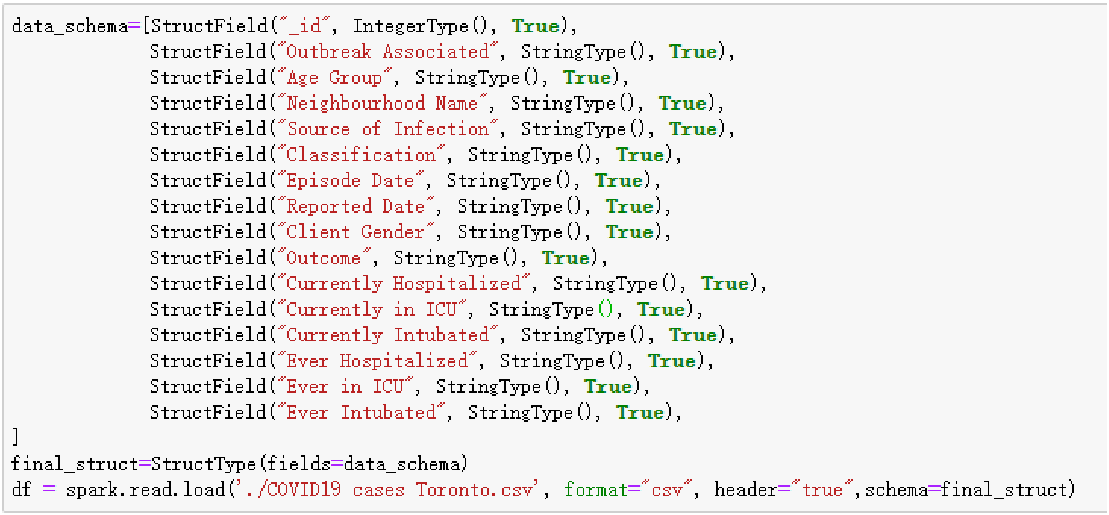


```
In  [6]: df.printSchema()

         root
          |— _id: string (nullable = true)
          |— Outbreak Associated: string (nullable = true)
          |— Age Group: string (nullable = true)
          |— Neighbourhood Name: string (nullable = true)
          |— Source of Infection: string (nullable = true)
          |— Classification: string (nullable = true)
          |— Episode Date: string (nullable = true)
          |— Reported Date: string (nullable = true)
          |— Client Gender: string (nullable = true)
          |— Outcome: string (nullable = true)
          |— Currently Hospitalized: string (nullable = true)
          |— Currently in ICU: string (nullable = true)
          |— Currently Intubated: string (nullable = true)
          |— Ever Hospitalized: string (nullable = true)
          |— Ever in ICU: string (nullable = true)
          |— Ever Intubated: string (nullable = true)
```

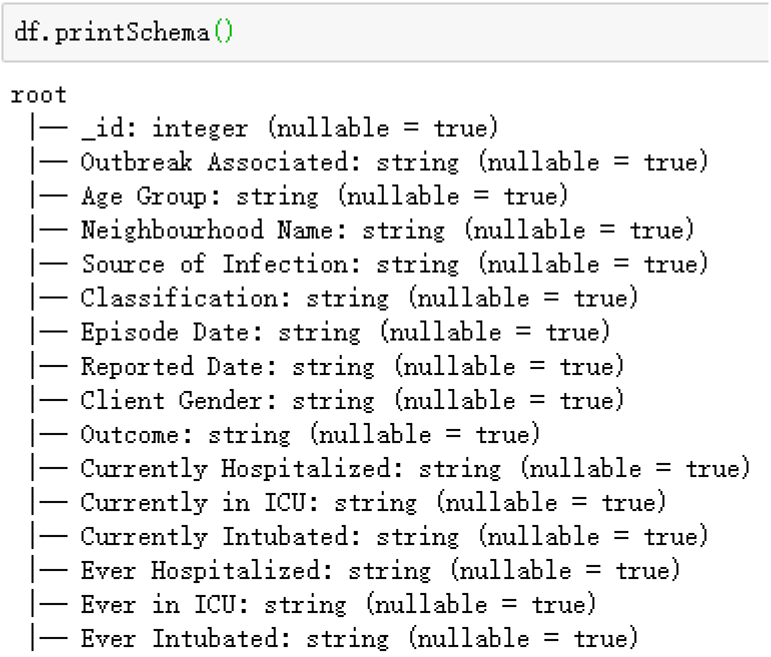
Some types of data seem to be incorrect, such as_ id， so I redefined the value type.



```
from pyspark.sql.types import (StructField,StringType,IntegerType,StructType)
```



```
data_schema=[StructField("_id", IntegerType(), True),
             StructField("Outbreak Associated", StringType(), True),
             StructField("Age Group", StringType(), True),
             StructField("Neighbourhood Name", StringType(), True),
             StructField("Source of Infection", StringType(), True),
             StructField("Classification", StringType(), True),
             StructField("Episode Date", StringType(), True),
             StructField("Reported Date", StringType(), True),
             StructField("Client Gender", StringType(), True),
             StructField("Outcome", StringType(), True),
             StructField("Currently Hospitalized", StringType(), True),
             StructField("Currently in ICU", StringType(), True),
             StructField("Currently Intubated", StringType(), True),
             StructField("Ever Hospitalized", StringType(), True),
             StructField("Ever in ICU", StringType(), True),
             StructField("Ever Intubated", StringType(), True),
]
final_struct=StructType(fields=data_schema)
df = spark.read.load('./COVID19 cases Toronto.csv', format="csv", header="true",schema=final_struct)
```



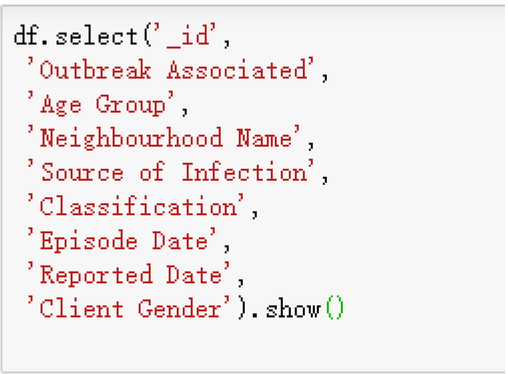
```
df.printSchema()
```

```
root
 |— _id: integer (nullable = true)
 |— Outbreak Associated: string (nullable = true)
 |— Age Group: string (nullable = true)
 |— Neighbourhood Name: string (nullable = true)
 |— Source of Infection: string (nullable = true)
 |— Classification: string (nullable = true)
 |— Episode Date: string (nullable = true)
 |— Reported Date: string (nullable = true)
 |— Client Gender: string (nullable = true)
 |— Outcome: string (nullable = true)
 |— Currently Hospitalized: string (nullable = true)
 |— Currently in ICU: string (nullable = true)
 |— Currently Intubated: string (nullable = true)
 |— Ever Hospitalized: string (nullable = true)
 |— Ever in ICU: string (nullable = true)
 |— Ever Intubated: string (nullable = true)
```

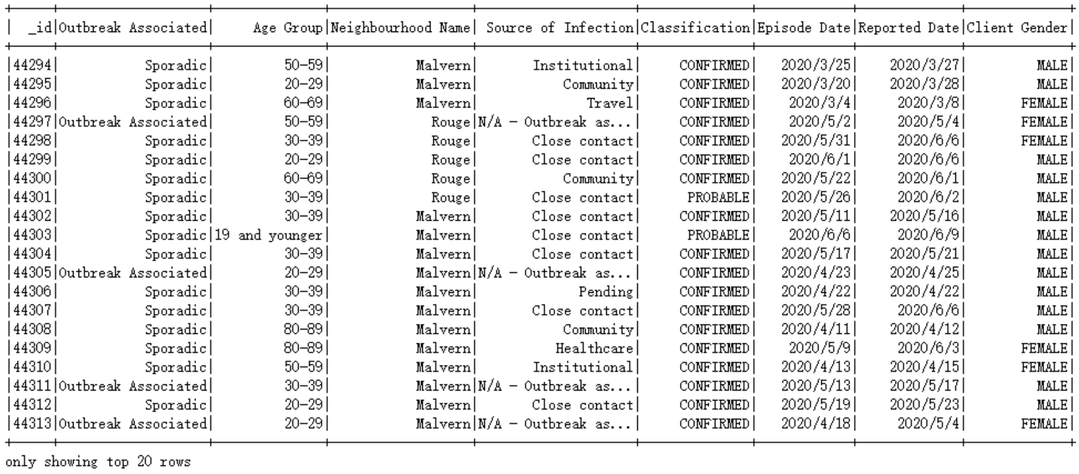
2.3 Explore data

2.3.1 Data Overview

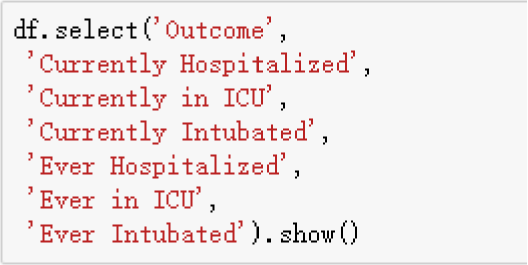
This data set contains demographic, geographic, and severity information for all confirmed and probable cases reported to and managed by Toronto Public Health since the first case was reported in January 2020.



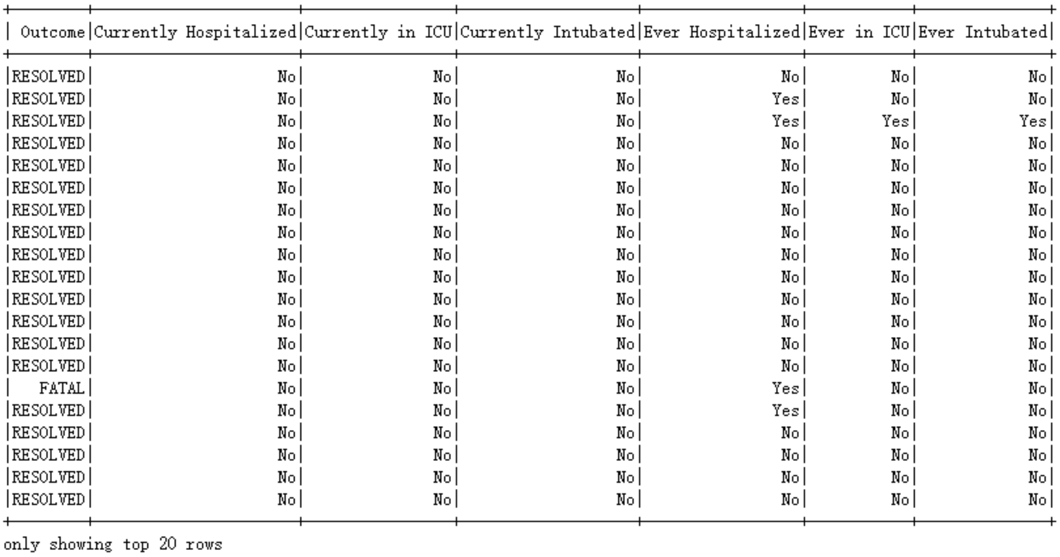
```
df.select(’_id’,
 ’Outbreak Associated’,
 ’Age Group’,
 ’Neighbourhood Name’,
 ’Source of Infection’,
 ’Classification’,
 ’Episode Date’,
 ’Reported Date’,
 ’Client Gender’).show()
```



```
+-----+-------------------+--------------+------------------+--------------------+--------------+------------+-------------+-------------+
|  _id|Outbreak Associated|     Age Group|Neighbourhood Name| Source of Infection|Classification|Episode Date|Reported Date|Client Gender|
+-----+-------------------+--------------+------------------+--------------------+--------------+------------+-------------+-------------+
|44294|           Sporadic|         50-59|           Malvern|       Institutional|     CONFIRMED|   2020/3/25|    2020/3/27|         MALE|
|44295|           Sporadic|         20-29|           Malvern|           Community|     CONFIRMED|   2020/3/20|    2020/3/28|         MALE|
|44296|           Sporadic|         60-69|           Malvern|              Travel|     CONFIRMED|    2020/3/4|     2020/3/8|       FEMALE|
|44297|Outbreak Associated|         50-59|             Rouge|N/A - Outbreak as...|     CONFIRMED|    2020/5/2|     2020/5/4|       FEMALE|
|44298|           Sporadic|         30-39|             Rouge|       Close contact|     CONFIRMED|   2020/5/31|     2020/6/6|       FEMALE|
|44299|           Sporadic|         20-29|             Rouge|       Close contact|     CONFIRMED|    2020/6/1|     2020/6/6|         MALE|
|44300|           Sporadic|         60-69|             Rouge|           Community|     CONFIRMED|   2020/5/22|     2020/6/1|         MALE|
|44301|           Sporadic|         30-39|             Rouge|       Close contact|      PROBABLE|   2020/5/26|     2020/6/2|         MALE|
|44302|           Sporadic|         30-39|           Malvern|       Close contact|     CONFIRMED|   2020/5/11|    2020/5/16|         MALE|
|44303|           Sporadic|19 and younger|           Malvern|       Close contact|      PROBABLE|    2020/6/6|     2020/6/9|         MALE|
|44304|           Sporadic|         30-39|           Malvern|       Close contact|     CONFIRMED|   2020/5/17|    2020/5/21|         MALE|
|44305|Outbreak Associated|         20-29|           Malvern|N/A - Outbreak as...|     CONFIRMED|   2020/4/23|    2020/4/25|         MALE|
|44306|           Sporadic|         30-39|           Malvern|             Pending|     CONFIRMED|   2020/4/22|    2020/4/22|         MALE|
|44307|           Sporadic|         30-39|           Malvern|       Close contact|     CONFIRMED|   2020/5/28|     2020/6/6|         MALE|
|44308|           Sporadic|         80-89|           Malvern|           Community|     CONFIRMED|   2020/4/11|    2020/4/12|         MALE|
|44309|           Sporadic|         80-89|           Malvern|          Healthcare|     CONFIRMED|    2020/5/9|     2020/6/3|       FEMALE|
|44310|           Sporadic|         50-59|           Malvern|       Institutional|     CONFIRMED|   2020/4/13|    2020/4/15|       FEMALE|
|44311|Outbreak Associated|         30-39|           Malvern|N/A - Outbreak as...|     CONFIRMED|   2020/5/13|    2020/5/17|         MALE|
|44312|           Sporadic|         20-29|           Malvern|       Close contact|     CONFIRMED|   2020/5/19|    2020/5/23|         MALE|
|44313|Outbreak Associated|         20-29|           Malvern|N/A - Outbreak as...|     CONFIRMED|   2020/4/18|     2020/5/4|       FEMALE|
+-----+-------------------+--------------+------------------+--------------------+--------------+------------+-------------+-------------+
only showing top 20 rows
```



```
df.select(’Outcome’,
 ’Currently Hospitalized’,
 ’Currently in ICU’,
 ’Currently Intubated’,
 ’Ever Hospitalized’,
 ’Ever in ICU’,
 ’Ever Intubated’).show()
```



```
+--------+----------------------+----------------+-------------------+-----------------+-----------+--------------+
| Outcome|Currently Hospitalized|Currently in ICU|Currently Intubated|Ever Hospitalized|Ever in ICU|Ever Intubated|
+--------+----------------------+----------------+-------------------+-----------------+-----------+--------------+
|RESOLVED|                    No|              No|                 No|               No|         No|            No|
|RESOLVED|                    No|              No|                 No|              Yes|         No|            No|
|RESOLVED|                    No|              No|                 No|              Yes|        Yes|           Yes|
|RESOLVED|                    No|              No|                 No|               No|         No|            No|
|RESOLVED|                    No|              No|                 No|               No|         No|            No|
|RESOLVED|                    No|              No|                 No|               No|         No|            No|
|RESOLVED|                    No|              No|                 No|               No|         No|            No|
|RESOLVED|                    No|              No|                 No|               No|         No|            No|
|RESOLVED|                    No|              No|                 No|               No|         No|            No|
|RESOLVED|                    No|              No|                 No|               No|         No|            No|
|RESOLVED|                    No|              No|                 No|               No|         No|            No|
|RESOLVED|                    No|              No|                 No|               No|         No|            No|
|RESOLVED|                    No|              No|                 No|               No|         No|            No|
|RESOLVED|                    No|              No|                 No|               No|         No|            No|
|   FATAL|                    No|              No|                 No|              Yes|         No|            No|
|RESOLVED|                    No|              No|                 No|              Yes|         No|            No|
|RESOLVED|                    No|              No|                 No|               No|         No|            No|
|RESOLVED|                    No|              No|                 No|               No|         No|            No|
|RESOLVED|                    No|              No|                 No|               No|         No|            No|
|RESOLVED|                    No|              No|                 No|               No|         No|            No|
+--------+----------------------+----------------+-------------------+-----------------+-----------+--------------+
only showing top 20 rows
```

data set of COVID-19 cases

2.3.2 Data Hypotheses

A total of 16 attributes can help us explore factors that can influence the outcome of recovery in COVID-19 cases. It is assumed that gender, age, source of infection and medical degree are all factors that will influence the condition of the case, and geographical location and source of infection are useless factors for further exploration.

2.3.3 Data analyses

This project uses jupyter notebook to analyze the basic data and conduct preliminary statistical analysis of the data set.

(1) Target analyses

In order to measure the recovery of a case, the outcome attribute can be used as the target value. The outcome property contains three values, FATAL, ACTIVE and RESOLVED. It can be seen from the figure that the number of recovered cases is far more than that of death.

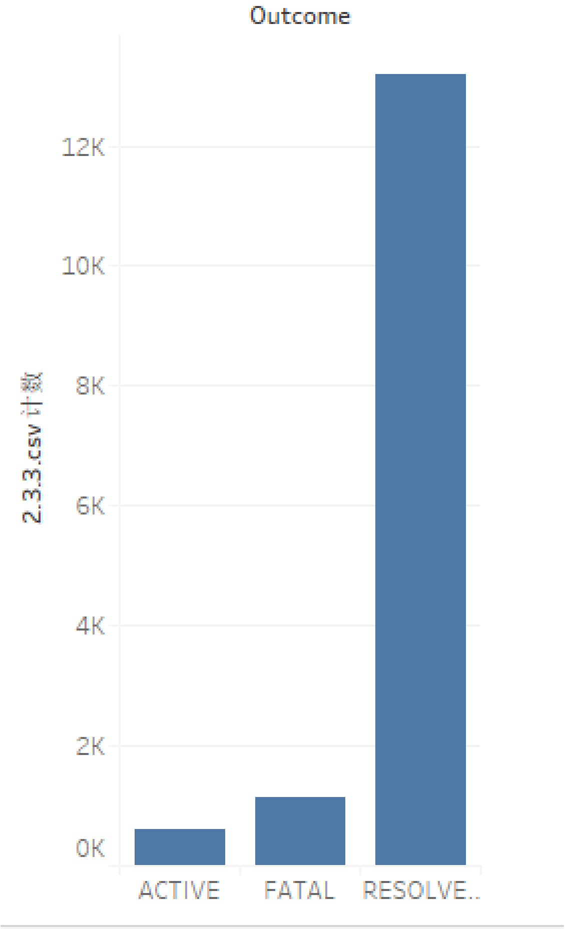


figure 6. Target value distribution

(2) Taking age as an example, it can be seen from the figure that among the confirmed or possibly confirmed cases, there are more cases of rehabilitation between 20 and 60 years old, almost all of which are more than 1500, while the cases of other age groups are less.

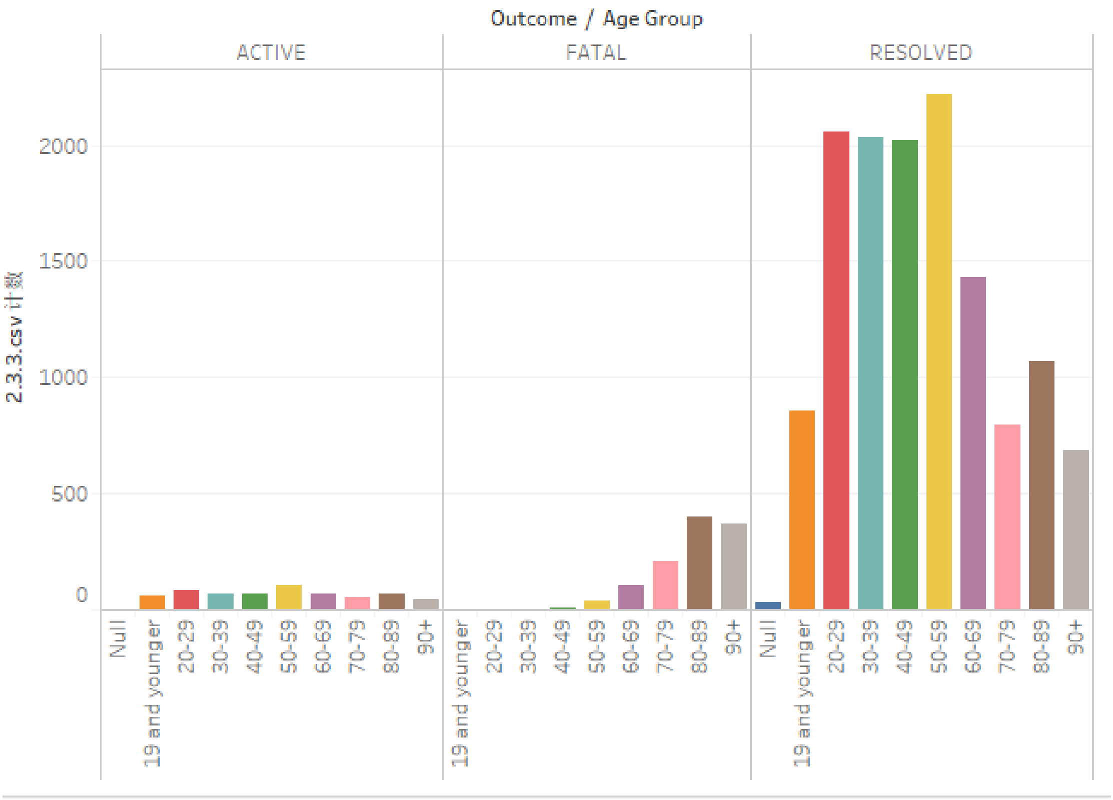


figure 7 confirmed or suspected cases over different age group

（3）It can be seen from the figure that women have more rehabilitation cases than men, which shows that gender may be one of the factors affecting the level of rehabilitation. It is also possible that there are more cases in women than men.

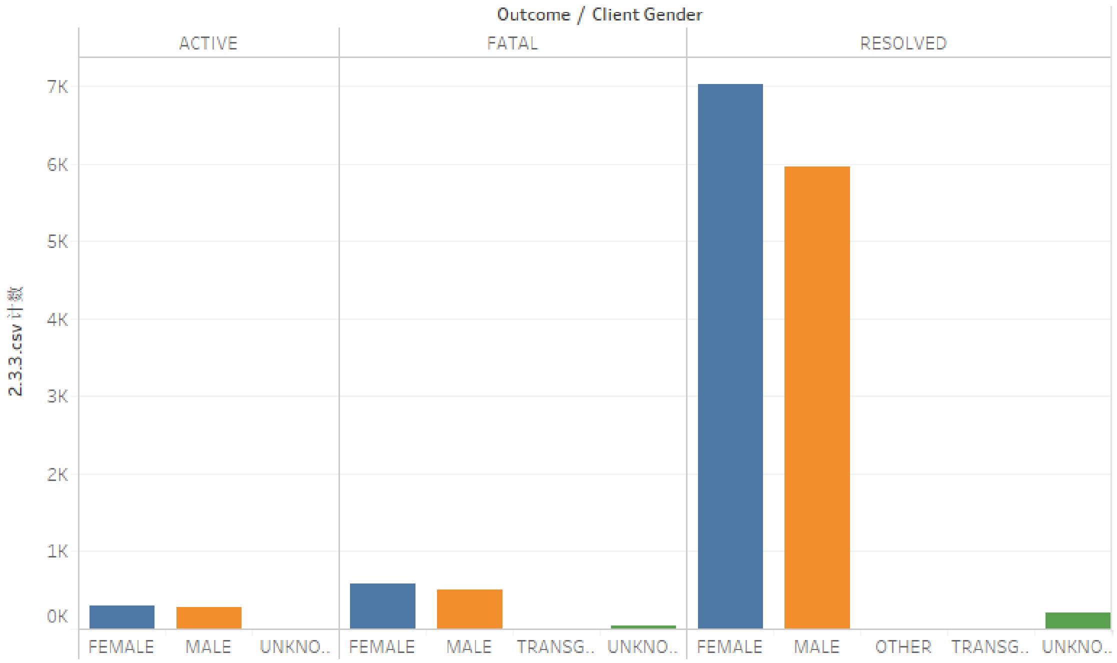


figure 8 confirmed or suspected cases over different gender group

（3）Before I did novel coronavirus pneumonia, I think that the medical conditions of the cases will affect their rehabilitation level of the new crown pneumonia cases. However, through visual analysis, it seems that most of the cases without treatment or hospitalization still have a high recovery rate. Currently, novel coronavirus pneumonia related treatments are rare, and only a few cases have been hospitalized.

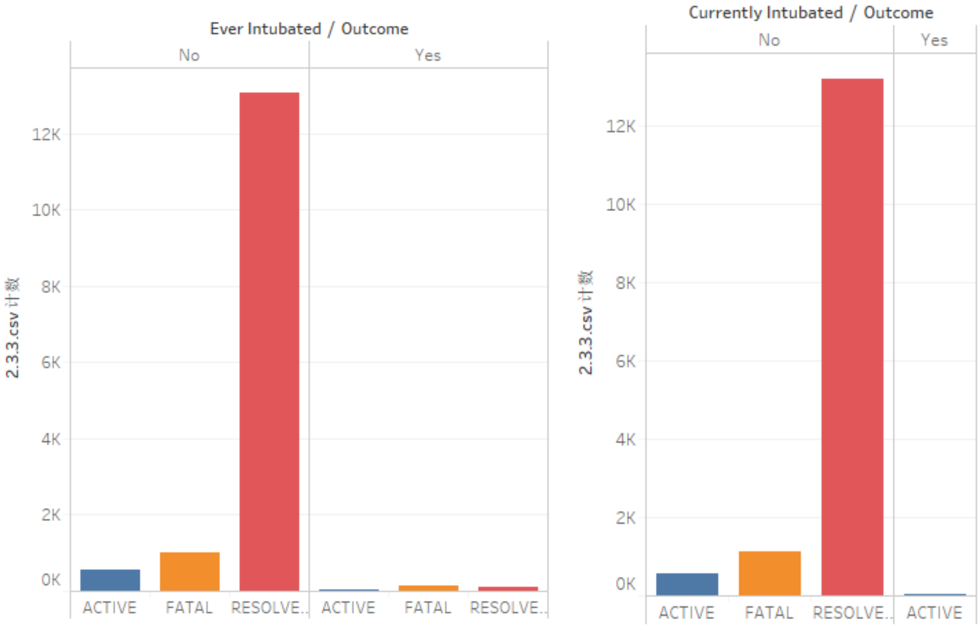


figure 9 Ever Intubated and currently Intubated

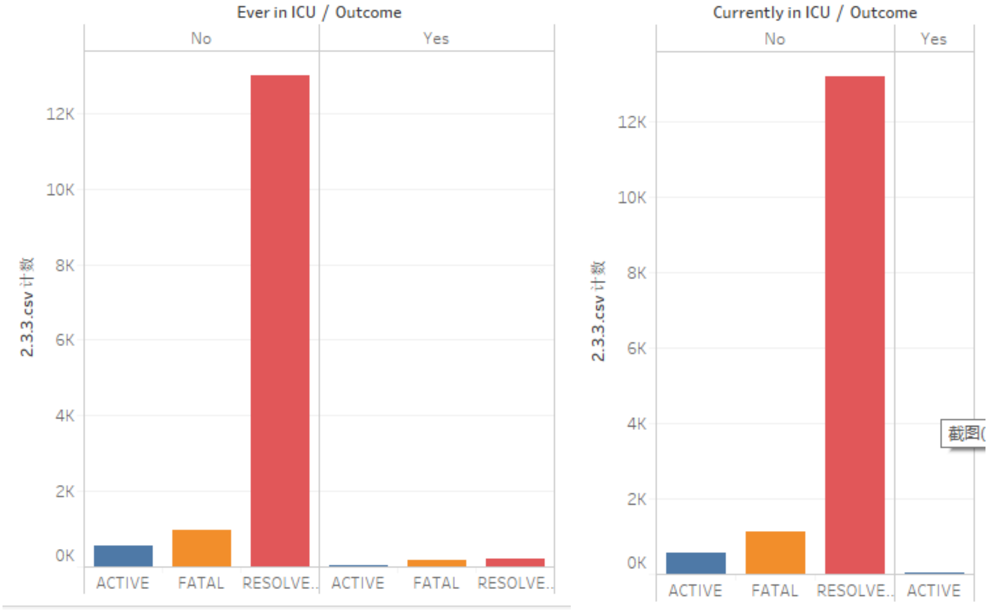


figure 10 Ever in ICU and currently in ICU

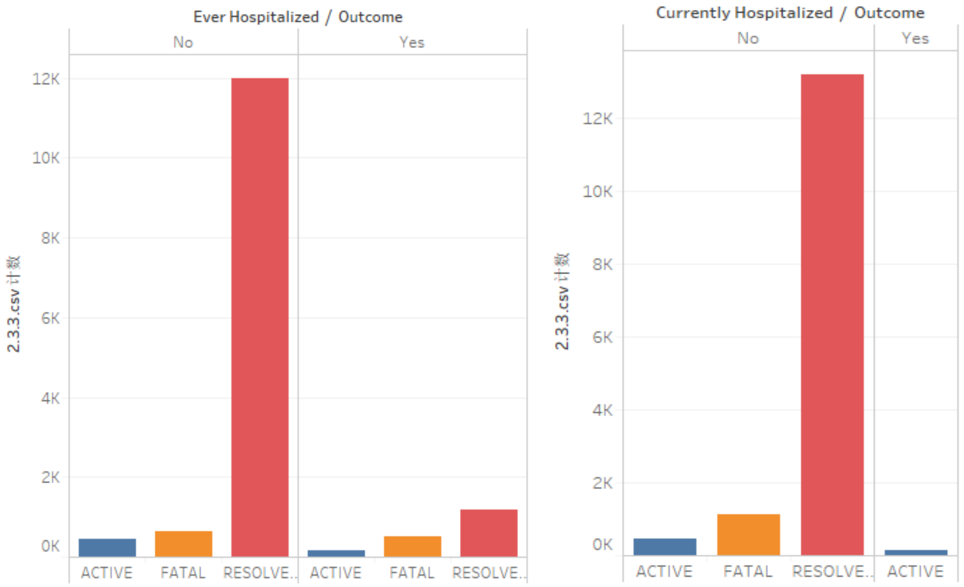


figure 11 Ever Hospitalized and Hospitalized

（4）The source of infection can be roughly divided into two categories: outbreak related cases and sporadic cases. The number of patients recovered from sporadic infection was more than that of outbreak related cases. No matter from Figure 11 or Figure 12, it can be seen that the number of death cases related to outbreak was more than that of sporadic infection. This suggests that the source of infection may be an important factor in the death of the case.

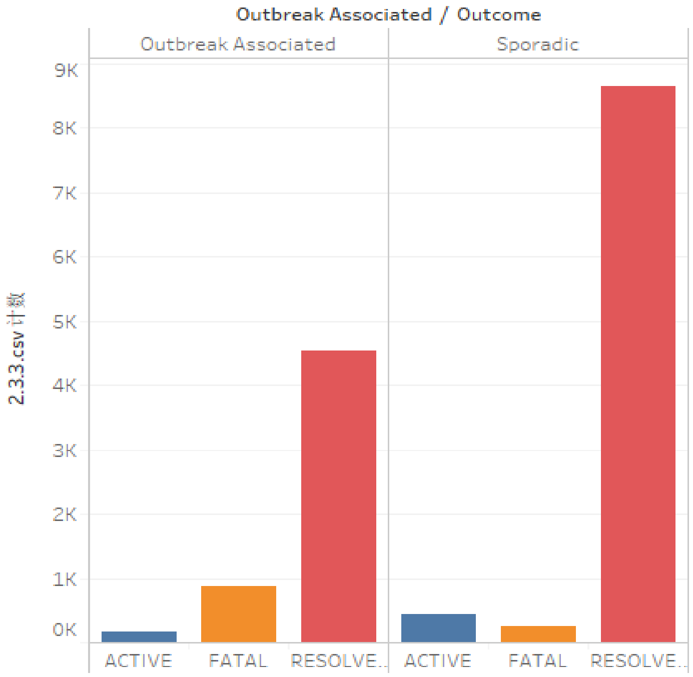


figure 11. Sporadic and Outbreak Associated cases

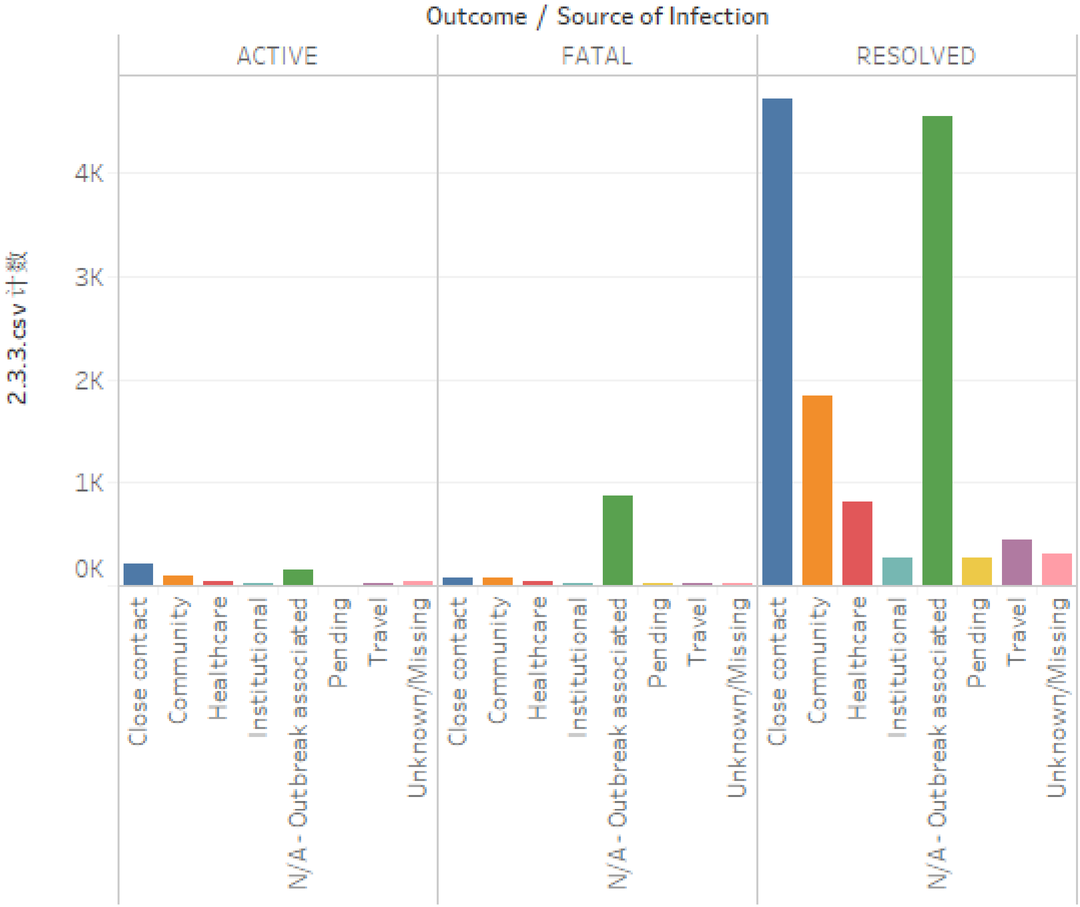
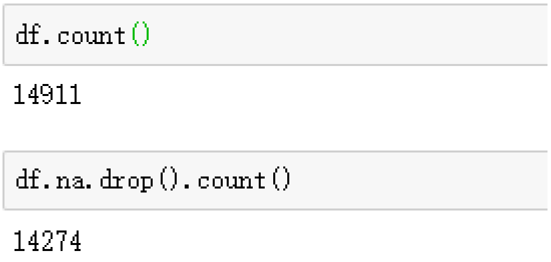


figure 12. Source of Infection

2.4 Verify the data quality

2.4.1 Data missing

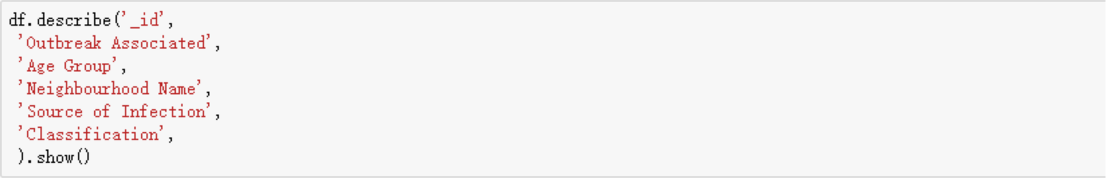
First of all, all magnetic field measurements are correct. As can be seen from Figure 14, the data is relatively complete, only the age group and Neighbourhood Name has a small number of null value. This may be because it is difficult to collect complete information during the COVID-19 outbreak. There are some values in the dataset that require further processing.



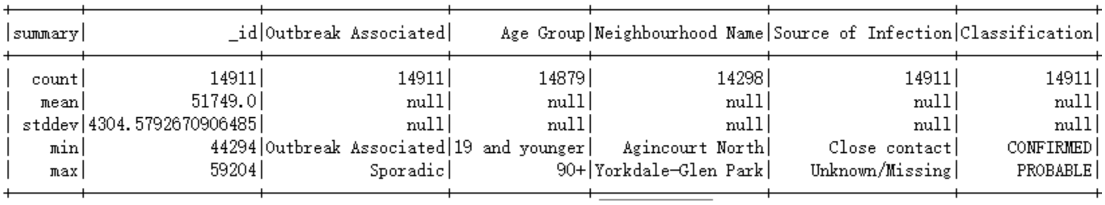
```
df.count()

14911

df.na.drop().count()

14274
```

2.4.2 Data errors and metric errors

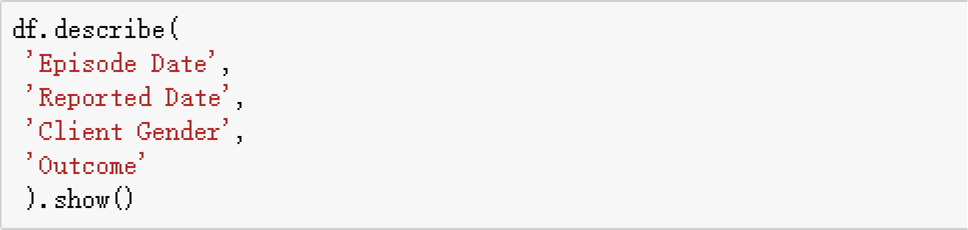
Since most of the data is a string type and cannot be calculated, there are basically no extreme values and outliers in the data. Only _id the only data of type int, _id can detect outliers and extreme values. However, due to _id itself is just a case number, and will not have any impact on the recovery or death of cases, so it has no practical significance. In addition to these defects and limitations about the dataset, no other data errors were found.



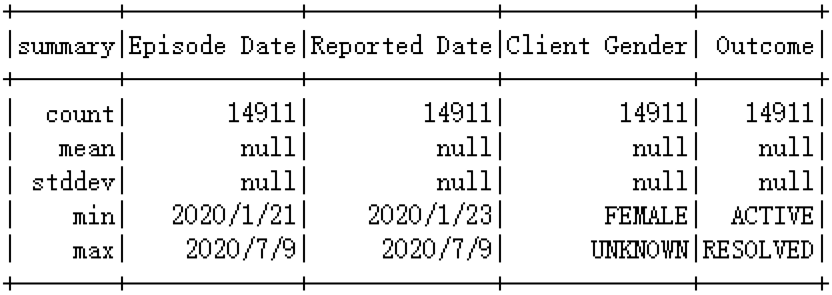
```
df.describe('_id’,
 'Outbreak Associated’,
 'Age Group’,
 'Neighbourhood Name’,
 'Source of Infection’,
 'Classification’,
 ).show()
```



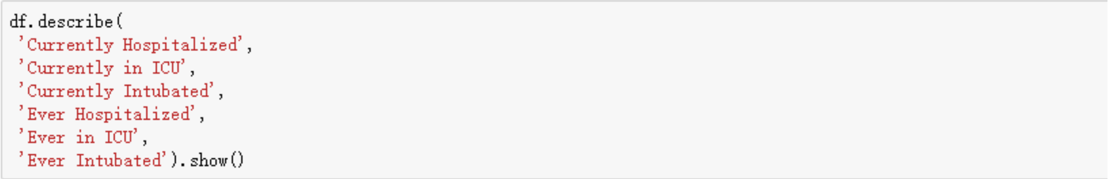
```
+-------+------------------+-------------------+--------------+------------------+-------------------+--------------+
|summary|               _id|Outbreak Associated|     Age Group|Neighbourhood Name|Source of Infection|Classification|
+-------+------------------+-------------------+--------------+------------------+-------------------+--------------+
|  count|             14911|              14911|         14879|             14298|              14911|         14911|
|   mean|           51749.0|               null|          null|              null|               null|          null|
| stddev|4304.5792670906485|               null|          null|              null|               null|          null|
|    min|             44294|Outbreak Associated|19 and younger|   Agincourt North|      Close contact|     CONFIRMED|
|    max|             59204|           Sporadic|           90+|Yorkdale-Glen Park|    Unknown/Missing|      PROBABLE|
+-------+------------------+-------------------+--------------+------------------+-------------------+--------------+
```



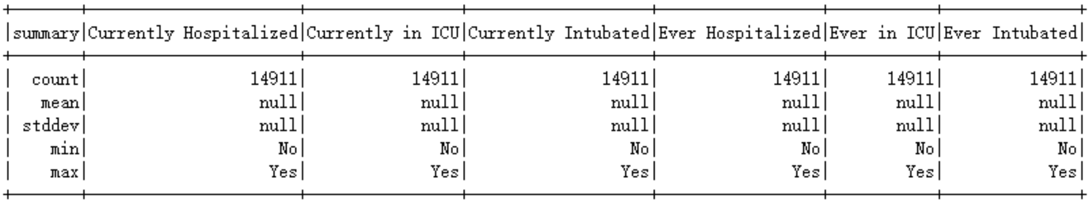
```
df.describe(
 'Episode Date’,
 'Reported Date’,
 'Client Gender’,
 'Outcome’
 ).show()
```



```
+-------+------------+-------------+-------------+--------+
|summary|Episode Date|Reported Date|Client Gender| Outcome|
+-------+------------+-------------+-------------+--------+
|  count|       14911|        14911|        14911|   14911|
|   mean|        null|         null|         null|    null|
| stddev|        null|         null|         null|    null|
|    min|   2020/1/21|    2020/1/23|       FEMALE|  ACTIVE|
|    max|    2020/7/9|     2020/7/9|      UNKNOWN|RESOLVED|
+-------+------------+-------------+-------------+--------+
```



```
df.describe(
 'Currently Hospitalized’,
 'Currently in ICU’,
 'Currently Intubated’,
 'Ever Hospitalized’,
 'Ever in ICU’,
 'Ever Intubated’).show()
```

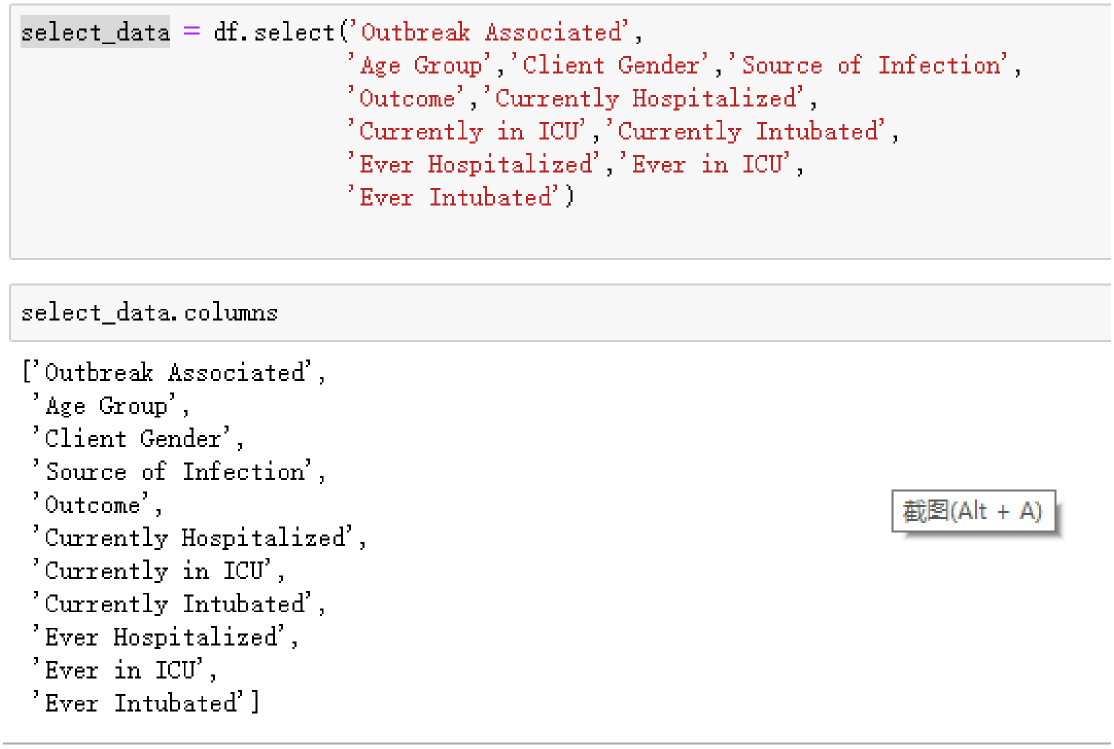


```
+-------+----------------------+----------------+-------------------+-----------------+-----------+--------------+
|summary|Currently Hospitalized|Currently in ICU|Currently Intubated|Ever Hospitalized|Ever in ICU|Ever Intubated|
+-------+----------------------+----------------+-------------------+-----------------+-----------+--------------+
|  count|                 14911|           14911|              14911|            14911|      14911|         14911|
|   mean|                  null|            null|               null|             null|       null|          null|
| stddev|                  null|            null|               null|             null|       null|          null|
|    min|                    No|              No|                 No|               No|         No|            No|
|    max|                   Yes|             Yes|                Yes|              Yes|        Yes|           Yes|
+-------+----------------------+----------------+-------------------+-----------------+-----------+--------------+
```

3.1 Select the Data

By observing complete data and the distribution of missing data, I decided to select ten attributes in the dataset.

Select attribute: Age Group, Client Gender, Source of Infection, Outcome, Currently Hospitalized, Currently in ICU, Currently Intubated, Ever Hospitalized, Ever in ICU and Ever Intubated. Because these attributes may have different degrees of influence and response in analyzing COVID- 19 survival rate of different populations.



```
select_data = df.select('Outbreak Associated',
                        'Age Group','Client Gender','Source of Infection',
                        'Outcome','Currently Hospitalized',
                        'Currently in ICU','Currently Intubated',
                        'Ever Hospitalized','Ever in ICU',
                        'Ever Intubated')
```

```
select_data.columns
```

```
['Outbreak Associated',
 'Age Group',
 'Client Gender',
 'Source of Infection',
 'Outcome',
 'Currently Hospitalized',
 'Currently in ICU',
 'Currently Intubated',
 'Ever Hospitalized',
 'Ever in ICU',
 'Ever Intubated']
```

(1) Source of Infection

The condition of confirmed cases may be related to different sources of infection, so different sources of infection may increase or decrease the survival rate of COVID-19.

(2) Client Gender

Gender may lead to different COVID- 19 survival rate because of men and women differences in personality and physical conditions.

(3) Age Group

People in different age groups have large differences in their physical, mental, or economic conditions, so this attribute should be retained, and its effect on COVID- 19 survival rate.

(4) Outcome

Data on the status of all confirmed and probable cases at present, which is the direct data of COVID- 19 survival rate.

(5) Currently Hospitalized and Ever Hospitalized

Whether or not a patient is admitted to hospital represents to some extent, the level of treatment for a confirmed case, which may affect the patient's level of recovery.

(6) Currently in ICU and Ever in ICU

The ICU represents the level of treatment for severe cases and affects the survival rate of COVID-19 cases.

(7) Currently Intubated and Ever Intubated

Intubated as treatment may play a decisive role in the rehabilitation of COVID-19 cases and increase the survival rate of COVID-19.

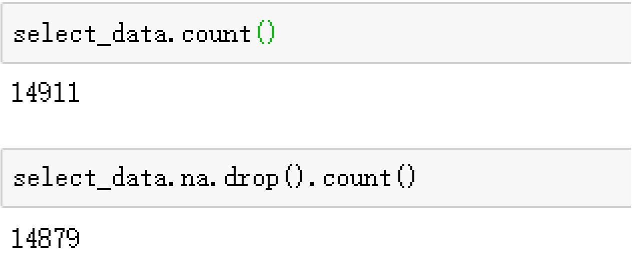
Table 3.1. Useless data and reason

Useless Data	reason

_id	no research value
Classification	It had nothing to do with survival after covid-19.
Episode Date	Means the time of coVID-19 diagnosis is independent of survival rate.
Reported Date	Means the time of coVID-19 diagnosis is independent of survival rate.

3.2 Clean the Data

After selecting the data, only the age group attribute has missing data. As the number of missing values is very small, only 32, which has little impact on the sample population, so I choose to fill the missing data with the previous data. After filling, the missing quantity is zero.

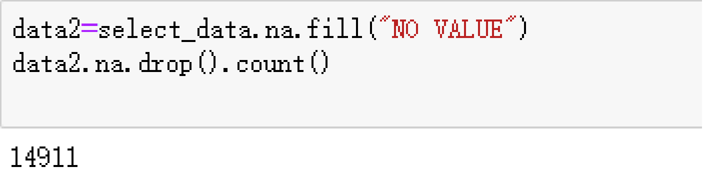


```
select_data.count()
```

```
14911
```

```
select_data.na.drop().count()
```

```
14879
```

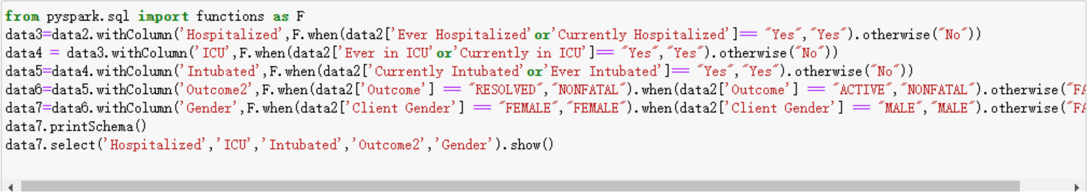


```
data2=select_data.na.fill(″NO VALUE″)
data2.na.drop().count()
```

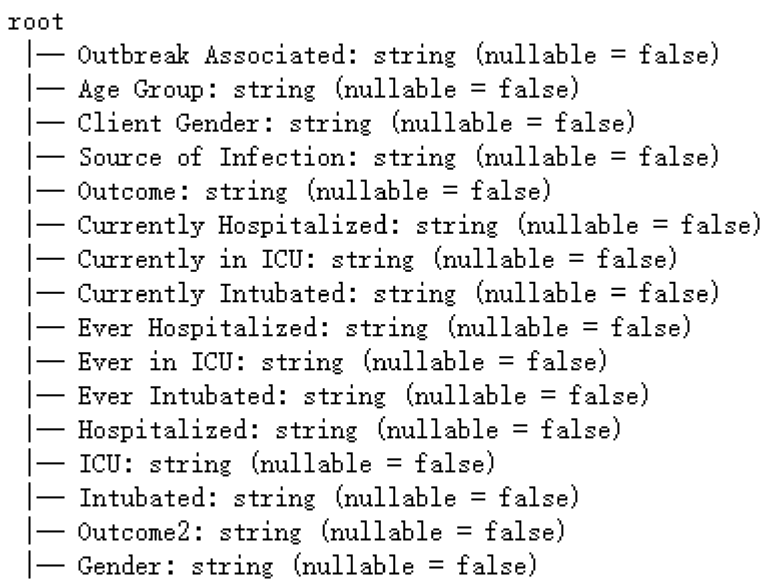
```
14911
```

3.3 Construct the data

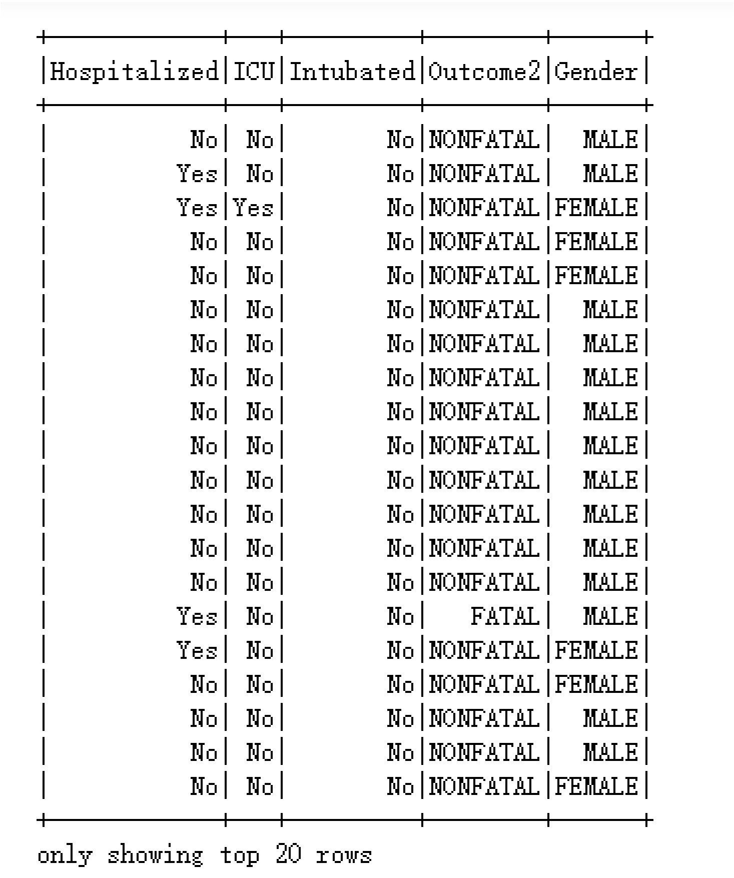
Create five new attributes Outcome2, ICU, Intubated ,Hospitalized and Gender. Both the former hospitalization and the current hospitalization belong to the hospitalization treatment, which can be combined into one field. Similarly, Intubated experiences can also compose a field. There are three values in Outcome, ACTIVE, RESOLVED and FATAL. ACTIVE and RESOLVED both represent patients still alive and can be combined into a single value. In the "client gender" attribute, "unknown", ″ transgender″ and "other" values can be classified into one group because of their small number



```
from pyspark.sql import functions as F
data3=data2.withColumn(’Hospitalized’,F.when(data2[’Ever Hospitalized’or’Currently Hospitalized’]== ″Yes″,″Yes″).otherwise(″No″))
data4 = data3.withColumn(’ICU’,F.when(data2[’Ever in ICU’or’Currently in ICU’]== ″Yes″,″Yes″).otherwise(″No″))
data5=data4.withColumn(’Intubated’,F.when(data2[’Currently Intubated’or’Ever Intubated’]== ″Yes″,″Yes″).otherwise(″No″))
data6=data5.withColumn(’Outcome2’,F.when(data2[’Outcome’] == ″RESOLVED″,″NONFATAL″).when(data2[’Outcome’] == ″ACTIVE″,″NONFATAL″).otherwise(″FA
data7=data6.withColumn(’Gender’,F.when(data2[’Client Gender’] == ″FEMALE″,″FEMALE″).when(data2[’Client Gender’] == ″MALE″,″MALE″).otherwise(″FA
data7.printSchema()
data7.select(’Hospitalized’,’ICU’,’Intubated’,’Outcome2’,’Gender’).show()
```



```
root
 |-- Outbreak Associated: string (nullable = false)
 |-- Age Group: string (nullable = false)
 |-- Client Gender: string (nullable = false)
 |-- Source of Infection: string (nullable = false)
 |-- Outcome: string (nullable = false)
 |-- Currently Hospitalized: string (nullable = false)
 |-- Currently in ICU: string (nullable = false)
 |-- Currently Intubated: string (nullable = false)
 |-- Ever Hospitalized: string (nullable = false)
 |-- Ever in ICU: string (nullable = false)
 |-- Ever Intubated: string (nullable = false)
 |-- Hospitalized: string (nullable = false)
 |-- ICU: string (nullable = false)
 |-- Intubated: string (nullable = false)
 |-- Outcome2: string (nullable = false)
 |-- Gender: string (nullable = false)
```



```
+------------+---+---------+--------+------+
|Hospitalized|ICU|Intubated|Outcome2|Gender|
+------------+---+---------+--------+------+
|          No| No|       No|NONFATAL|  MALE|
|         Yes| No|       No|NONFATAL|  MALE|
|         Yes|Yes|       No|NONFATAL|FEMALE|
|          No| No|       No|NONFATAL|FEMALE|
|          No| No|       No|NONFATAL|FEMALE|
|          No| No|       No|NONFATAL|  MALE|
|          No| No|       No|NONFATAL|  MALE|
|          No| No|       No|NONFATAL|  MALE|
|          No| No|       No|NONFATAL|  MALE|
|          No| No|       No|NONFATAL|  MALE|
|          No| No|       No|NONFATAL|  MALE|
|          No| No|       No|NONFATAL|  MALE|
|          No| No|       No|NONFATAL|  MALE|
|          No| No|       No|NONFATAL|  MALE|
|         Yes| No|       No|   FATAL|  MALE|
|         Yes| No|       No|NONFATAL|FEMALE|
|          No| No|       No|NONFATAL|FEMALE|
|          No| No|       No|NONFATAL|  MALE|
|          No| No|       No|NONFATAL|  MALE|
|          No| No|       No|NONFATAL|FEMALE|
+------------+---+---------+--------+------+
only showing top 20 rows
```

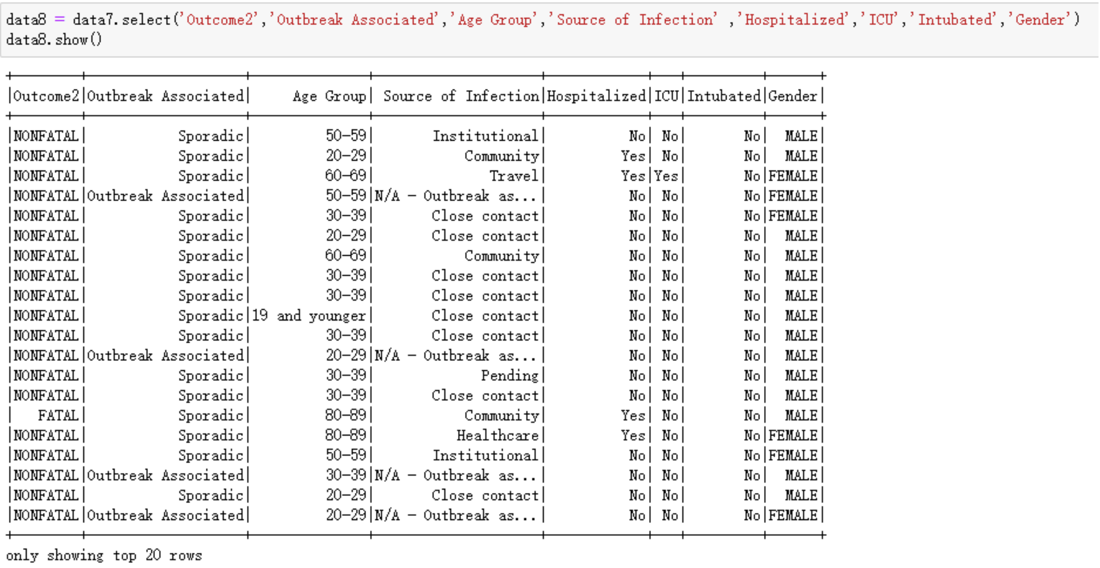
3.4 Integrate various data sources

There is no need to integrate data because the project has only one data source, since the current data source already includes medical factors (hospitalization, source of infection) and personal factors (gender, age). There is already a comprehensive set of factors affecting cure rates, so we can use current data sources for data mining.

3.5 Format the data as required

3.5.1 Check again for useless fields

The Hospitalized attribute is generated based on the currently hospitalized and the 'ever hospitalized' attributes. The Intubated attribute is generated based on the currently intubated and the ever Intubated. ICU, currently in ICU and, 'ever in ICU' are similar and repetitive attributes. To avoid duplication, some attributes can be removed. Therefore, when selecting attributes as features, we should avoid selecting attributes with the same meaning.

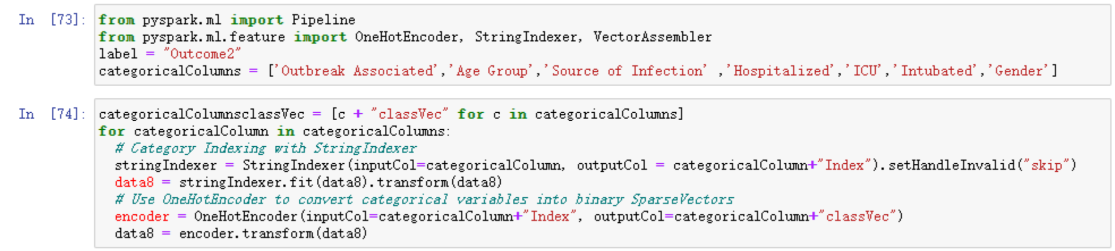


```
data8 = data7.select('Outcome2','Outbreak Associated','Age Group','Source of Infection' ,'Hospitalized','ICU','Intubated','Gender')
data8.show()
```

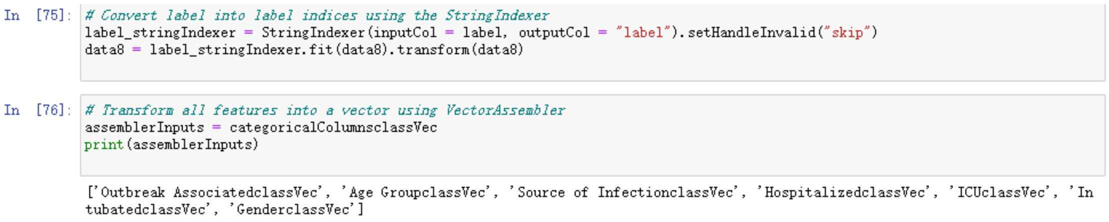
```
+--------+-------------------+--------------+--------------------+------------+---+---------+------+
|Outcome2|Outbreak Associated|     Age Group| Source of Infection|Hospitalized|ICU|Intubated|Gender|
+--------+-------------------+--------------+--------------------+------------+---+---------+------+
|NONFATAL|           Sporadic|         50-59|       Institutional|          No| No|       No|  MALE|
|NONFATAL|           Sporadic|         20-29|           Community|         Yes| No|       No|  MALE|
|NONFATAL|           Sporadic|         60-69|              Travel|         Yes|Yes|       No|FEMALE|
|NONFATAL|Outbreak Associated|         50-59|N/A - Outbreak as...|          No| No|       No|FEMALE|
|NONFATAL|           Sporadic|         30-39|       Close contact|          No| No|       No|FEMALE|
|NONFATAL|           Sporadic|         20-29|       Close contact|          No| No|       No|  MALE|
|NONFATAL|           Sporadic|         60-69|           Community|          No| No|       No|  MALE|
|NONFATAL|           Sporadic|         30-39|       Close contact|          No| No|       No|  MALE|
|NONFATAL|           Sporadic|         30-39|       Close contact|          No| No|       No|  MALE|
|NONFATAL|           Sporadic|19 and younger|       Close contact|          No| No|       No|  MALE|
|NONFATAL|           Sporadic|         30-39|       Close contact|          No| No|       No|  MALE|
|NONFATAL|Outbreak Associated|         20-29|N/A - Outbreak as...|          No| No|       No|  MALE|
|NONFATAL|           Sporadic|         30-39|             Pending|          No| No|       No|  MALE|
|NONFATAL|           Sporadic|         30-39|       Close contact|          No| No|       No|  MALE|
|   FATAL|           Sporadic|         80-89|           Community|         Yes| No|       No|  MALE|
|NONFATAL|           Sporadic|         80-89|          Healthcare|         Yes| No|       No|FEMALE|
|NONFATAL|           Sporadic|         50-59|       Institutional|          No| No|       No|FEMALE|
|NONFATAL|Outbreak Associated|         30-39|N/A - Outbreak as...|          No| No|       No|  MALE|
|NONFATAL|           Sporadic|         20-29|       Close contact|          No| No|       No|  MALE|
|NONFATAL|Outbreak Associated|         20-29|N/A - Outbreak as...|          No| No|       No|FEMALE|
+--------+-------------------+--------------+--------------------+------------+---+---------+------+
only showing top 20 rows
```

3.5.2 Format the data to fit decision tree model

The categories should be converted to a number so that they can be utilised effectively.



```
In [73]: from pyspark.ml import Pipeline
         from pyspark.ml.feature import OneHotEncoder, StringIndexer, VectorAssembler
         label = "Outcome2"
         categoricalColumns = ['Outbreak Associated','Age Group','Source of Infection' ,'Hospitalized','ICU','Intubated','Gender']

In [74]: categoricalColumnsclassVec = [c + "classVec" for c in categoricalColumns]
         for categoricalColumn in categoricalColumns:
           # Category Indexing with StringIndexer
           stringIndexer = StringIndexer(inputCol=categoricalColumn, outputCol = categoricalColumn+"Index").setHandleInvalid("skip")
           data8 = stringIndexer.fit(data8).transform(data8)
           # Use OneHotEncoder to convert categorical variables into binary SparseVectors
           encoder = OneHotEncoder(inputCol=categoricalColumn+"Index", outputCol=categoricalColumn+"classVec")
           data8 = encoder.transform(data8)
```

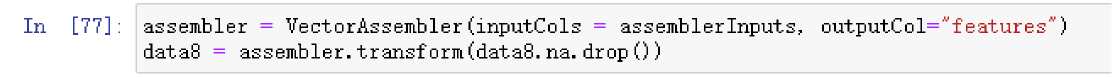


```
In [75]: # Convert label into label indices using the StringIndexer
         label_stringIndexer = StringIndexer(inputCol = label, outputCol = "label").setHandleInvalid("skip")
         data8 = label_stringIndexer.fit(data8).transform(data8)

In [76]: # Transform all features into a vector using VectorAssembler
         assemblerInputs = categoricalColumnsclassVec
         print(assemblerInputs)

         ['Outbreak AssociatedclassVec', 'Age GroupclassVec', 'Source of InfectionclassVec', 'HospitalizedclassVec', 'ICUclassVec', 'In
         tubatedclassVec', 'GenderclassVec']
```

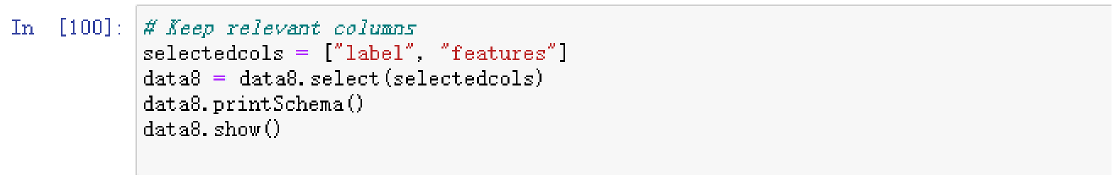
4.1 Reduce the Data

4.1.1 Feature selection

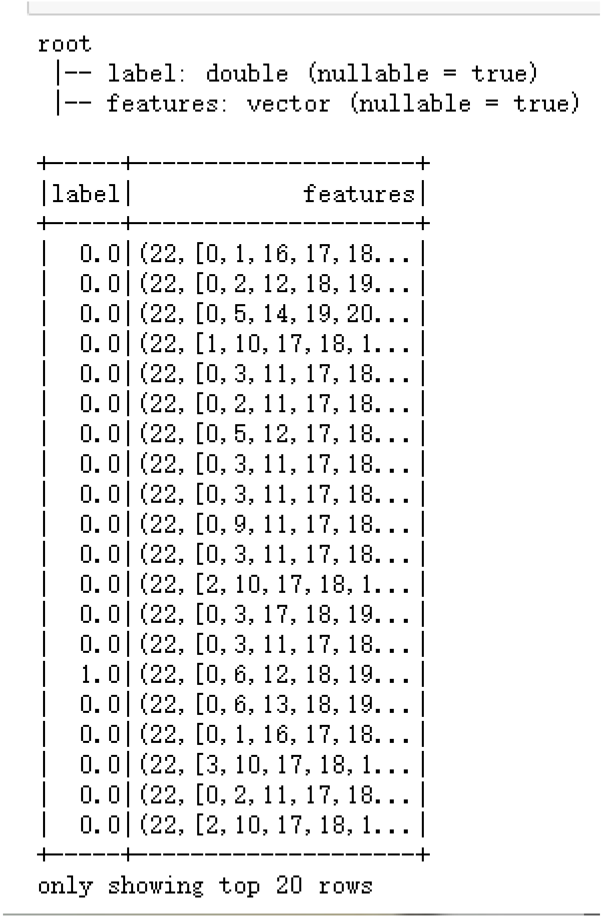
(1) Select features related to the predictor variable Outcome2. Use this to reduce data vertically



```
In [77]: assembler = VectorAssembler(inputCols = assemblerInputs, outputCol="features")
         data8 = assembler.transform(data8.na.drop())
```

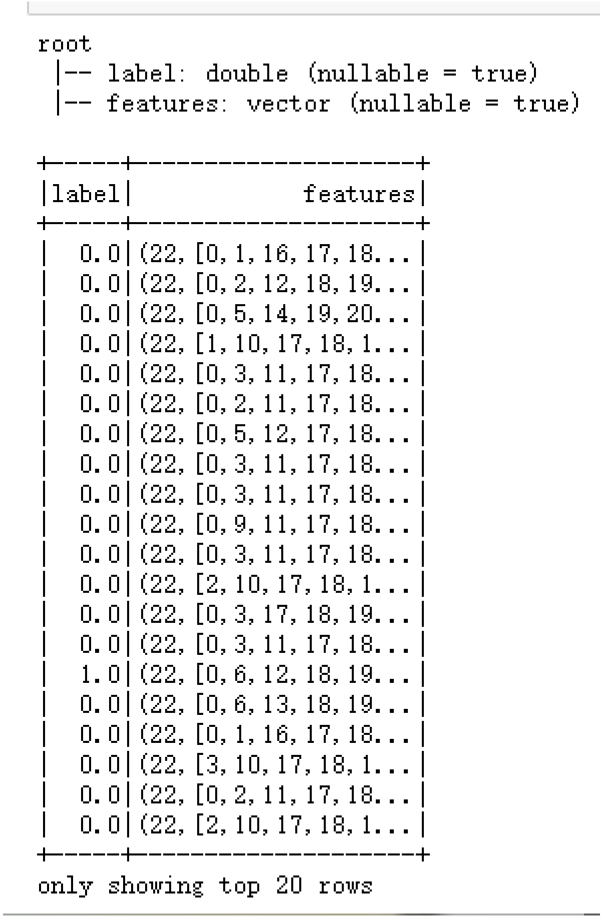


```
In [100]: # Keep relevant columns
          selectedcols = ["label", "features"]
          data8 = data8.select(selectedcols)
          data8.printSchema()
          data8.show()
```



```
root
 |-- label: double (nullable = true)
 |-- features: vector (nullable = true)

+-----+--------------------+
|label|            features|
+-----+--------------------+
|  0.0|(22,[0,1,16,17,18...|
|  0.0|(22,[0,2,12,18,19...|
|  0.0|(22,[0,5,14,19,20...|
|  0.0|(22,[1,10,17,18,1...|
|  0.0|(22,[0,3,11,17,18...|
|  0.0|(22,[0,2,11,17,18...|
|  0.0|(22,[0,5,12,17,18...|
|  0.0|(22,[0,3,11,17,18...|
|  0.0|(22,[0,3,11,17,18...|
|  0.0|(22,[0,9,11,17,18...|
|  0.0|(22,[0,3,11,17,18...|
|  0.0|(22,[2,10,17,18,1...|
|  0.0|(22,[0,3,17,18,19...|
|  0.0|(22,[0,3,11,17,18...|
|  1.0|(22,[0,6,12,18,19...|
|  0.0|(22,[0,6,13,18,19...|
|  0.0|(22,[0,1,16,17,18...|
|  0.0|(22,[3,10,17,18,1...|
|  0.0|(22,[0,2,11,17,18...|
|  0.0|(22,[2,10,17,18,1...|
+-----+--------------------+
only showing top 20 rows
```

（2）ChiSqSelector stands for chi square feature selection. It is suitable for label data with category characteristics. According to independent chi square test, chisqselector selects the features that the category label mainly depends on. It can help us select the most predictive features.

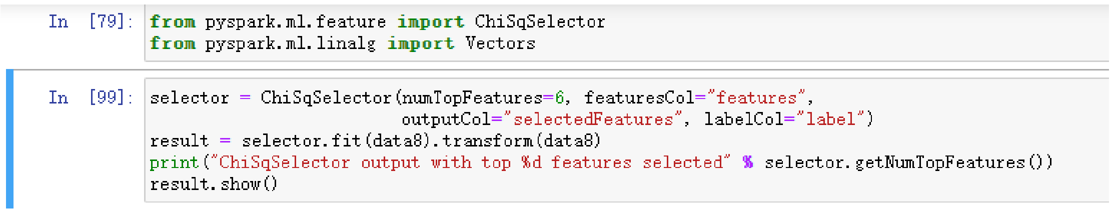


```
root
 |-- label: double (nullable = true)
 |-- features: vector (nullable = true)

+-----+--------------------+
|label|            features|
+-----+--------------------+
|  0.0|(22,[0,1,16,17,18...|
|  0.0|(22,[0,2,12,18,19...|
|  0.0|(22,[0,5,14,19,20...|
|  0.0|(22,[1,10,17,18,1...|
|  0.0|(22,[0,3,11,17,18...|
|  0.0|(22,[0,2,11,17,18...|
|  0.0|(22,[0,5,12,17,18...|
|  0.0|(22,[0,3,11,17,18...|
|  0.0|(22,[0,3,11,17,18...|
|  0.0|(22,[0,9,11,17,18...|
|  0.0|(22,[0,3,11,17,18...|
|  0.0|(22,[2,10,17,18,1...|
|  0.0|(22,[0,3,17,18,19...|
|  0.0|(22,[0,3,11,17,18...|
|  1.0|(22,[0,6,12,18,19...|
|  0.0|(22,[0,6,13,18,19...|
|  0.0|(22,[0,1,16,17,18...|
|  0.0|(22,[3,10,17,18,1...|
|  0.0|(22,[0,2,11,17,18...|
|  0.0|(22,[2,10,17,18,1...|
+-----+--------------------+
only showing top 20 rows
```

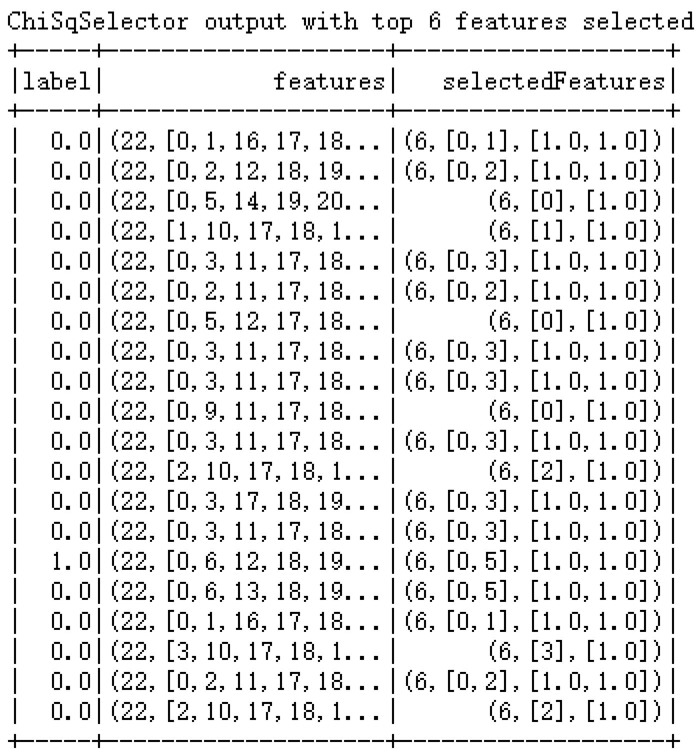
4.1.2 Reduce unimportant attribute

numTopFeatures: the top (Num) features with the most predictive ability were selected by chi square test;

We set numtopfeatures to 6 and select six important features from the seven features. The result is shown below,



```
In [79]: from pyspark.ml.feature import ChiSqSelector
         from pyspark.ml.linalg import Vectors

In [99]: selector = ChiSqSelector(numTopFeatures=6, featuresCol="features",
                                  outputCol="selectedFeatures", labelCol="label")
         result = selector.fit(data8).transform(data8)
         print("ChiSqSelector output with top %d features selected" % selector.getNumTopFeatures())
         result.show()
```

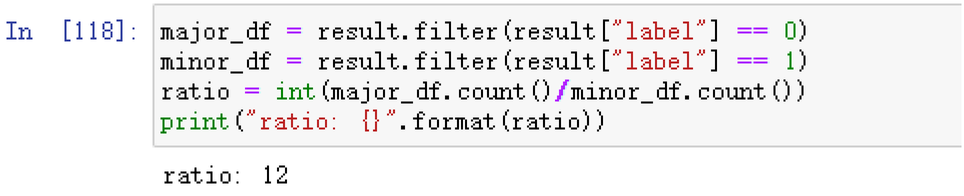


```
ChiSqSelector output with top 6 features selected
+-----+--------------------+--------------------+
|label|            features|    selectedFeatures|
+-----+--------------------+--------------------+
|  0.0|(22,[0,1,16,17,18...|(6,[0,1],[1.0,1.0])|
|  0.0|(22,[0,2,12,18,19...|(6,[0,2],[1.0,1.0])|
|  0.0|(22,[0,5,14,19,20...|      (6,[0],[1.0])|
|  0.0|(22,[1,10,17,18,1...|      (6,[1],[1.0])|
|  0.0|(22,[0,3,11,17,18...|(6,[0,3],[1.0,1.0])|
|  0.0|(22,[0,2,11,17,18...|(6,[0,2],[1.0,1.0])|
|  0.0|(22,[0,5,12,17,18...|      (6,[0],[1.0])|
|  0.0|(22,[0,3,11,17,18...|(6,[0,3],[1.0,1.0])|
|  0.0|(22,[0,3,11,17,18...|(6,[0,3],[1.0,1.0])|
|  0.0|(22,[0,9,11,17,18...|      (6,[0],[1.0])|
|  0.0|(22,[0,3,11,17,18...|(6,[0,3],[1.0,1.0])|
|  0.0|(22,[2,10,17,18,1...|      (6,[2],[1.0])|
|  0.0|(22,[0,3,17,18,19...|(6,[0,3],[1.0,1.0])|
|  0.0|(22,[0,3,11,17,18...|(6,[0,3],[1.0,1.0])|
|  1.0|(22,[0,6,12,18,19...|(6,[0,5],[1.0,1.0])|
|  0.0|(22,[0,6,13,18,19...|(6,[0,5],[1.0,1.0])|
|  0.0|(22,[0,1,16,17,18...|(6,[0,1],[1.0,1.0])|
|  0.0|(22,[3,10,17,18,1...|      (6,[3],[1.0])|
|  0.0|(22,[0,2,11,17,18...|(6,[0,2],[1.0,1.0])|
|  0.0|(22,[2,10,17,18,1...|      (6,[2],[1.0])|
+-----+--------------------+--------------------+
```

4.2 Project the Data

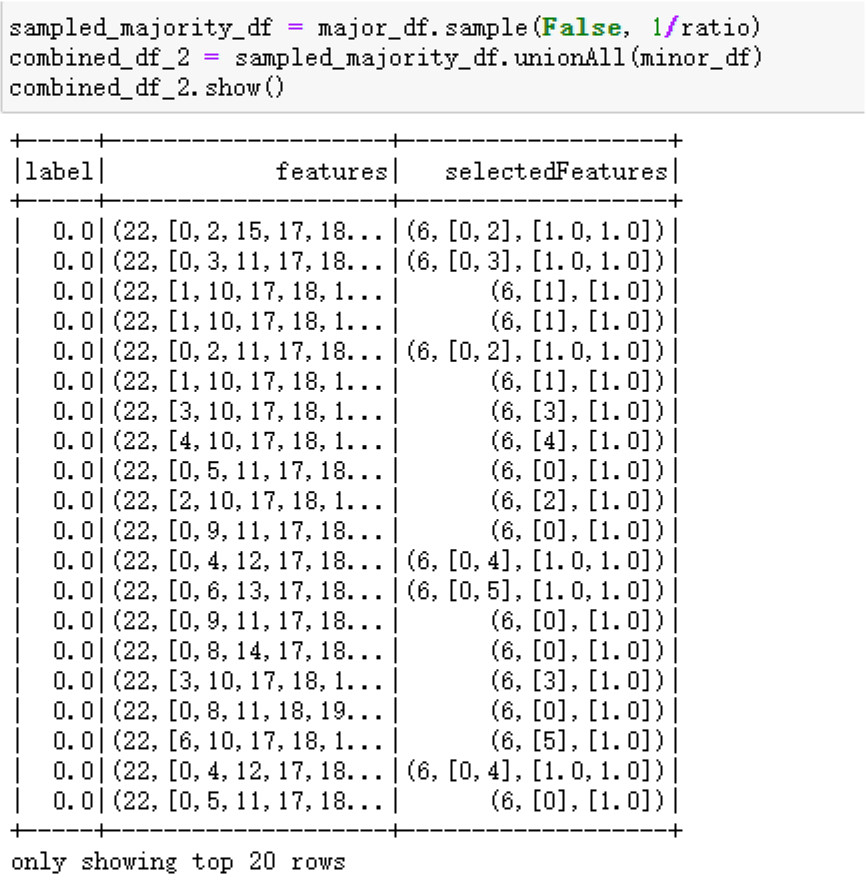
4.2.1 Balance the Data

(1) Select the target attribute and display its distribution. We want to predict the 'Outcome2', so it should be our ‘label’ rather than part of 'data'. The results are shown in the following figure. In the ‘label’’ attribute, The value of label "0" is 12 times of the number of "1" label.



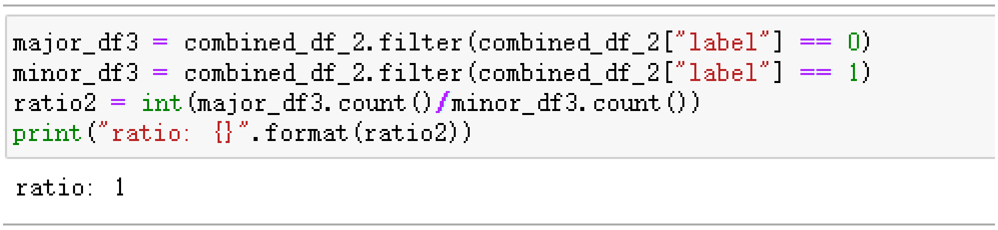
```
In  [118]: major_df = result.filter(result["label"] == 0)
           minor_df = result.filter(result["label"] == 1)
           ratio = int(major_df.count()/minor_df.count())
           print("ratio: {}".format(ratio))

           ratio: 12
```

（2）Because the original data set is large and the major class of the minority class is quite different, I decided to use undersampling for balance data.



```
sampled_majority_df = major_df.sample(False, 1/ratio)
combined_df_2 = sampled_majority_df.unionAll(minor_df)
combined_df_2.show()
```

```
+-----+--------------------+-------------------+
|label|            features|   selectedFeatures|
+-----+--------------------+-------------------+
|  0.0|(22,[0,2,15,17,18...|(6,[0,2],[1.0,1.0])|
|  0.0|(22,[0,3,11,17,18...|(6,[0,3],[1.0,1.0])|
|  0.0|(22,[1,10,17,18,1...|      (6,[1],[1.0])|
|  0.0|(22,[1,10,17,18,1...|      (6,[1],[1.0])|
|  0.0|(22,[0,2,11,17,18...|(6,[0,2],[1.0,1.0])|
|  0.0|(22,[1,10,17,18,1...|      (6,[1],[1.0])|
|  0.0|(22,[3,10,17,18,1...|      (6,[3],[1.0])|
|  0.0|(22,[4,10,17,18,1...|      (6,[4],[1.0])|
|  0.0|(22,[0,5,11,17,18...|      (6,[0],[1.0])|
|  0.0|(22,[2,10,17,18,1...|      (6,[2],[1.0])|
|  0.0|(22,[0,9,11,17,18...|      (6,[0],[1.0])|
|  0.0|(22,[0,4,12,17,18...|(6,[0,4],[1.0,1.0])|
|  0.0|(22,[0,6,13,17,18...|(6,[0,5],[1.0,1.0])|
|  0.0|(22,[0,9,11,17,18...|      (6,[0],[1.0])|
|  0.0|(22,[0,8,14,17,18...|      (6,[0],[1.0])|
|  0.0|(22,[3,10,17,18,1...|      (6,[3],[1.0])|
|  0.0|(22,[0,8,11,18,19...|      (6,[0],[1.0])|
|  0.0|(22,[6,10,17,18,1...|      (6,[5],[1.0])|
|  0.0|(22,[0,4,12,17,18...|(6,[0,4],[1.0,1.0])|
|  0.0|(22,[0,5,11,17,18...|      (6,[0],[1.0])|
+-----+--------------------+-------------------+
only showing top 20 rows
```

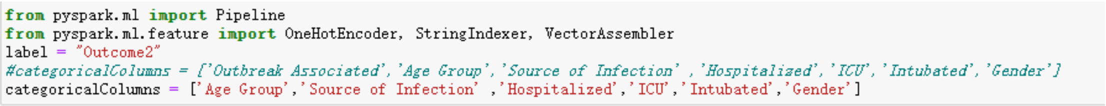


```
major_df3 = combined_df_2.filter(combined_df_2["label"] == 0)
minor_df3 = combined_df_2.filter(combined_df_2["label"] == 1)
ratio2 = int(major_df3.count()/minor_df3.count())
print("ratio: {}".format(ratio2))
```

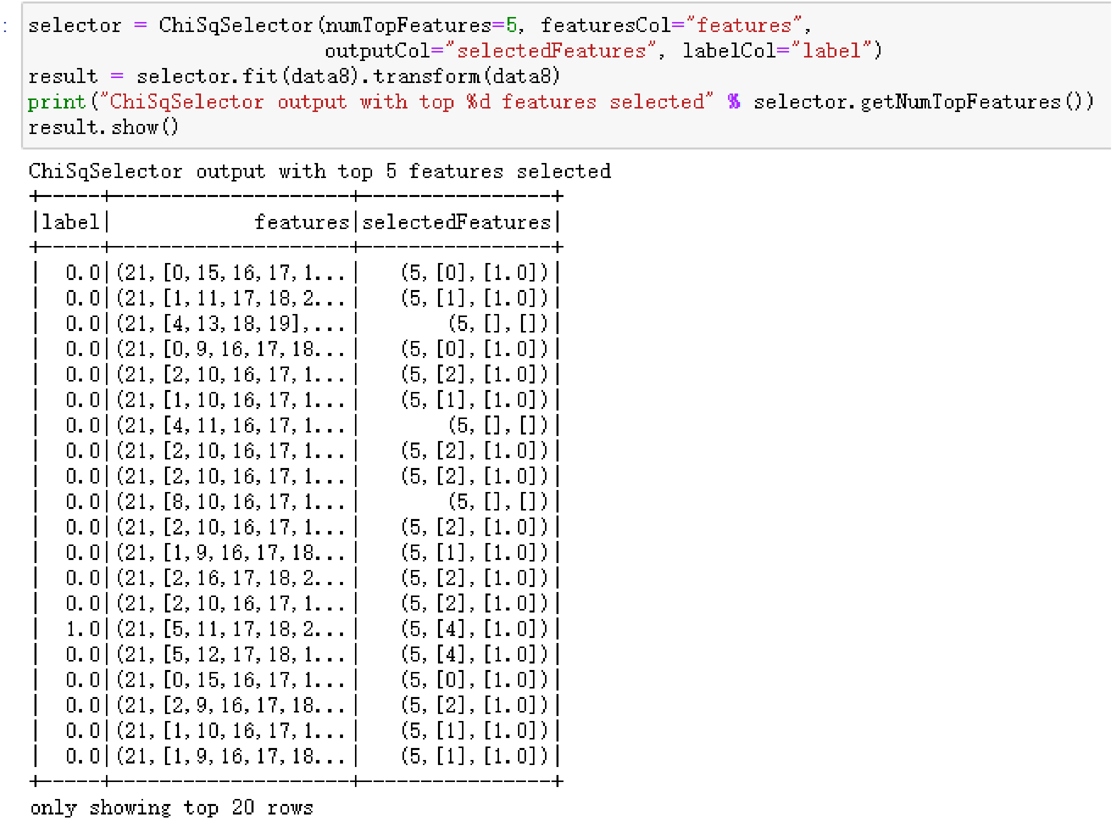
```
ratio: 1
```

4.2.2 Distribution of target attribute

So far, most of the steps have been completed. After reviewing the data again, it is found that the 'source of effect' attribute includes' outbreak associated '. Therefore, delete 'outbreak associated' from features and modify the value of NumTopfeatures in select feature to 5. There are only five important attributes left.



```
from pyspark.ml import Pipeline
from pyspark.ml.feature import OneHotEncoder, StringIndexer, VectorAssembler
label = "Outcome2"
#categoricalColumns = ['Outbreak Associated','Age Group','Source of Infection' ,'Hospitalized','ICU','Intubated','Gender']
categoricalColumns = ['Age Group','Source of Infection' ,'Hospitalized','ICU','Intubated','Gender']
```



```
selector = ChiSqSelector(numTopFeatures=5, featuresCol="features",
                         outputCol="selectedFeatures", labelCol="label")
result = selector.fit(data8).transform(data8)
print("ChiSqSelector output with top %d features selected" % selector.getNumTopFeatures())
result.show()
```

```
ChiSqSelector output with top 5 features selected
+-----+--------------------+----------------+
|label|            features|selectedFeatures|
+-----+--------------------+----------------+
|  0.0|(21,[0,15,16,17,1...|   (5,[0],[1.0])|
|  0.0|(21,[1,11,17,18,2...|   (5,[1],[1.0])|
|  0.0|(21,[4,13,18,19],...|       (5,[],[])|
|  0.0|(21,[0,9,16,17,18...|   (5,[0],[1.0])|
|  0.0|(21,[2,10,16,17,1...|   (5,[2],[1.0])|
|  0.0|(21,[1,10,16,17,1...|   (5,[1],[1.0])|
|  0.0|(21,[4,11,16,17,1...|       (5,[],[])|
|  0.0|(21,[2,10,16,17,1...|   (5,[2],[1.0])|
|  0.0|(21,[2,10,16,17,1...|   (5,[2],[1.0])|
|  0.0|(21,[8,10,16,17,1...|       (5,[],[])|
|  0.0|(21,[2,10,16,17,1...|   (5,[2],[1.0])|
|  0.0|(21,[1,9,16,17,18...|   (5,[1],[1.0])|
|  0.0|(21,[2,16,17,18,2...|   (5,[2],[1.0])|
|  0.0|(21,[2,10,16,17,1...|   (5,[2],[1.0])|
|  1.0|(21,[5,11,17,18,2...|   (5,[4],[1.0])|
|  0.0|(21,[5,12,17,18,1...|   (5,[4],[1.0])|
|  0.0|(21,[0,15,16,17,1...|   (5,[0],[1.0])|
|  0.0|(21,[2,9,16,17,18...|   (5,[2],[1.0])|
|  0.0|(21,[1,10,16,17,1...|   (5,[1],[1.0])|
|  0.0|(21,[1,9,16,17,18...|   (5,[1],[1.0])|
+-----+--------------------+----------------+
only showing top 20 rows
```

5.1 Match and discuss the objectives of data mining to data mining methods

5.1.1 Supervised and Unsupervised Learning

(1) How to select learning Method

Whether the data set use labeled data

(2) Supervised learning

The supervisory model uses the values of one or more input fields to predict the values of one or more output or target fields. Some examples are:

decision trees (C &R trees, CHAID, QUEST and C5.0 algorithms), regression (linear, logical, generalized linear and Cox regression algorithms), neural networks, support vector machines, and Bayesian networks.

Oversight models help organizations predict known outcomes, such as whether customers will buy or leave, or whether transactions conform to known fraud patterns. Model technology includes machine learning, rule induction, subgroup identification, statistical method and multi model generation.

（3） Unsupervised learning

Unsupervised learning groups data into unlabeled data sets according to potential hidden features. Because there is no label, the results cannot be evaluated (this is the key difference between supervised learning algorithms). By grouping the data by unsupervised learning, you can learn about some raw data that may not be visible in other situations. In high-dimensional data sets or large data sets, this problem is more obvious (Jones,2017).

5.1.2 Classification, Association and Segmentation

(1) Classification: Find the common characteristics of a group of data objects in the database and divide them into different categories according to the classification mode. The purpose is to map the data items in the database to a given category through the classification model (Data for Data Mining, n.d.).

(2) Association: The association model finds a pattern in the data in which one or more entities (such as events, purchases, or properties) are associated with one or more other entities. The model builds the set of rules that define these relationships. Here, the fields in the data can be either input or target. You can find these associations manually, but the association rules algorithm finds them faster and explores more complex patterns. (IBM Knowledge Center, n.d.).

(3) Segmentation: The segmentation model divides the data into segments or clusters with similar input field patterns. Because they are only interested in input fields, the segmentation model has no concept of output fields or target fields. Examples of segmentation models include Kohonen network, K-means clustering, two-step clustering and anomaly detection (IBM Knowledge Center, n.d.) Subdivision models (also known as "clustering models") are useful when specific results are unknown (for example. The clustering model focuses on identifying groups of similar records and labeling them according to the group they belong to. This is done without prior knowledge of the group and its characteristics, and it distinguishes the clustering model from other modeling techniques because the model does not have predefined output or target fields for prediction (IBM Knowledge Center, n.d.).

5.2 Select the appropriate data-mining method (s) based on discussion

5.2.1 Choose supervised learning

Because there are both input variables and output variables (Outcome), so choose to use supervised learning methods. Because the prediction target is continuous, the regression method is appropriate.

5.2.2 Choose classification

Classification can be used for forecasting. The purpose of classification is to automatically derive the general description of given data from historical data records, so as to predict future data. Different from regression, the output of classification is discrete, while the output of regression is continuous.

Because the target attribute Outcome is categorical value, so we choose the classification model (IBM Knowledge Center, n.d.).

6.1 Conduct exploratory analysis and discuss

6.1.1 Algorithm discussion

In pyspark, there are many algorithms for classification, such as logistic regression. But considering the data format, as well as the characteristics of the data, I choose the following algorithms for research.

In my project, I use VectorAssembler combining raw features and features generated by different feature transformers into a single feature vector. This step helps me train ML models like logistic regression and decision trees.

(1) logistic regression

In spark.ml, binary results can be predicted by logistic regression, and multiple results can be predicted by multivariate logistic regression (Apache Spark, n.d.).

(2) decision trees.

Decision tree is widely used because it is easy to interpret, process classification features, expand to multi class classification settings, do not need feature scaling, and can capture nonlinear and feature interaction. Tree ensemble algorithms, such as random forest and boosting, perform best in classification and regression tasks (Apache Spark, n.d.).

(3) Random forest are ensembles of decision trees. Random forests combine many decision trees to reduce the risk of over fitting. The spark.ml The implementation supports random forests for binary and multiclass classification and regression, using both continuous and categorical features (Apache Spark, n.d.).

(4) Gradient boosted trees (GBTs) are ensembles decision trees. GBTs iteratively trains the decision tree to minimize the loss function. The spark.ml The implementation supports binary classification and regression of GBTs, and uses continuous and classification features(Apache Spark, n.d.).

6.2 Select data-mining algorithms based on discussion

6.2.1 Algorithm requirements:

(1) Objective: Find out the relationship between the target attribute and the predictors, and use it to predict the probability of the target variable occurring.

(2) Label attribute type: double

(3) transform: VectorAssembler

(4) Result requirements: prefer model results which are easy to demonstrate

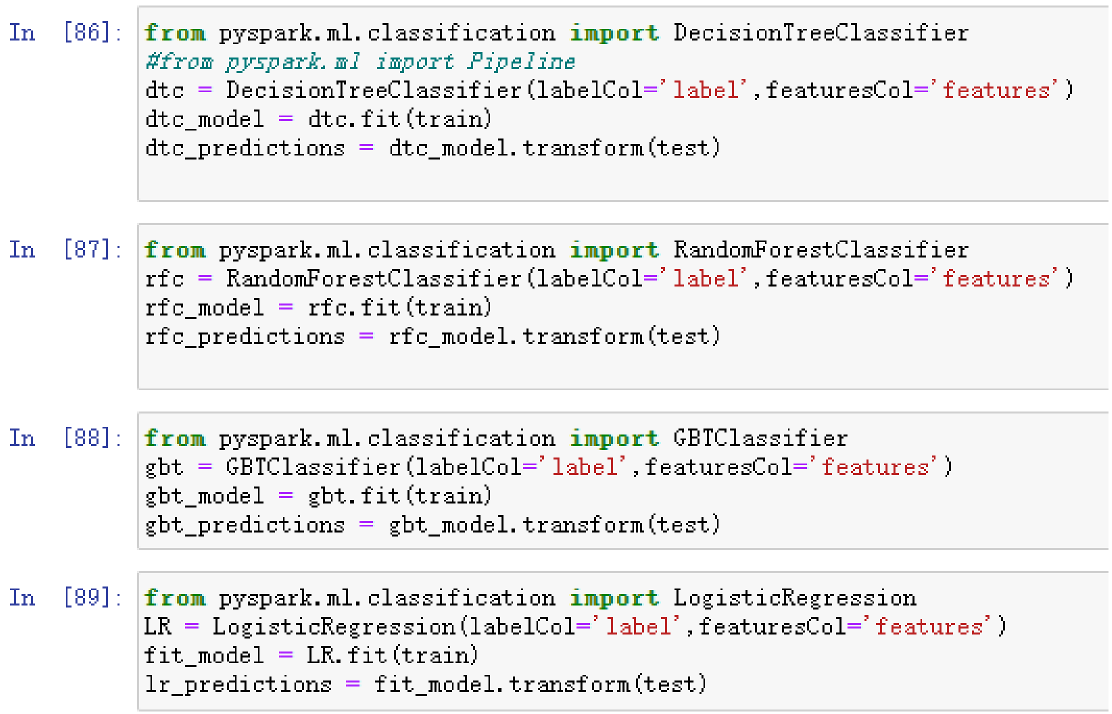
Since random forest and gradient enhancement tree are based on decision tree algorithm, both decision tree and logistic regression algorithm can meet the type requirements of binary outcome and feature. Now I can't decide which algorithm is best, so I decide to use both algorithms and compare the model results.

6.3 Select appropriate model(s) and choose relevant parameter(s)

Now let's use the algorithm for predicting new cases of coronavirus pneumonia in pyspark and create our prediction model: we have a set of data for new cases of coronavirus pneumonia, which will be our feature set.

First, import the relevant classifiers, which are decisiontree classifier, gbtclassifier， random forest classifier and LogisticRegression.

"Label", that is, "outcome2", will be the target of our prediction. We need to train the models first, and then compare the quality of models with test sets.



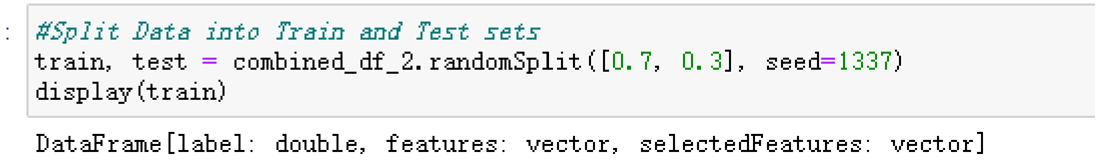
```
In [86]: from pyspark.ml.classification import DecisionTreeClassifier
         #from pyspark.ml import Pipeline
         dtc = DecisionTreeClassifier(labelCol='label',featuresCol='features')
         dtc_model = dtc.fit(train)
         dtc_predictions = dtc_model.transform(test)

In [87]: from pyspark.ml.classification import RandomForestClassifier
         rfc = RandomForestClassifier(labelCol='label',featuresCol='features')
         rfc_model = rfc.fit(train)
         rfc_predictions = rfc_model.transform(test)

In [88]: from pyspark.ml.classification import GBTClassifier
         gbt = GBTClassifier(labelCol='label',featuresCol='features')
         gbt_model = gbt.fit(train)
         gbt_predictions = gbt_model.transform(test)

In [89]: from pyspark.ml.classification import LogisticRegression
         LR = LogisticRegression(labelCol='label',featuresCol='features')
         fit_model = LR.fit(train)
         lr_predictions = fit_model.transform(test)
```

7.1 Logical test designs

7.1.1 create logical test design

(1) When an algorithm use a limited data set to search for the best parameters for a particular model, it may model not only the general pattern in the data, but also any noise specific to that data set, resulting in poor performance of the model on the test data, that is, overfitting (Usama, Gregory & Padhraic,1996). So we chose cross-validation to solve this problem

(2) Due to the small number of data samples and the need for sufficient data to test results, we set 70% training set and 30% testing set. Use randomSplit to divide the data into 70% training data and 30% test data



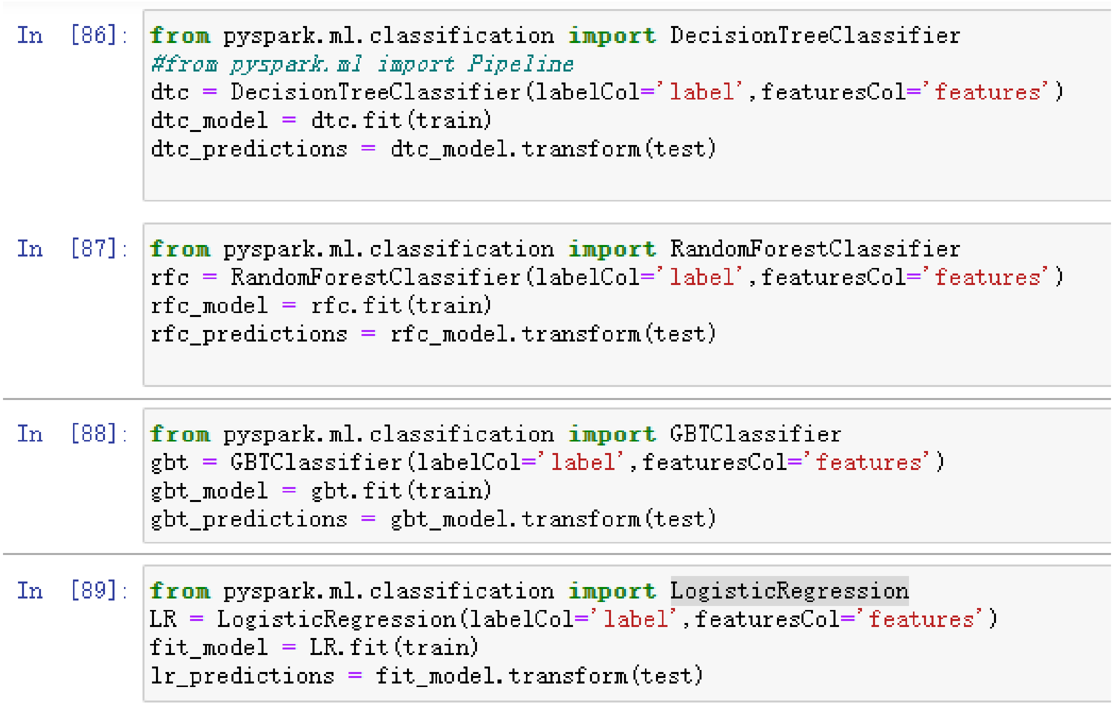
```
#Split Data into Train and Test sets
train, test = combined_df_2.randomSplit([0.7, 0.3], seed=1337)
display(train)
```

```
DataFrame[label: double, features: vector, selectedFeatures: vector]
```

7.2 Data mining must be conducted (the model must run).

Run models

The model is tested successfully, and the comparison results are obtained.

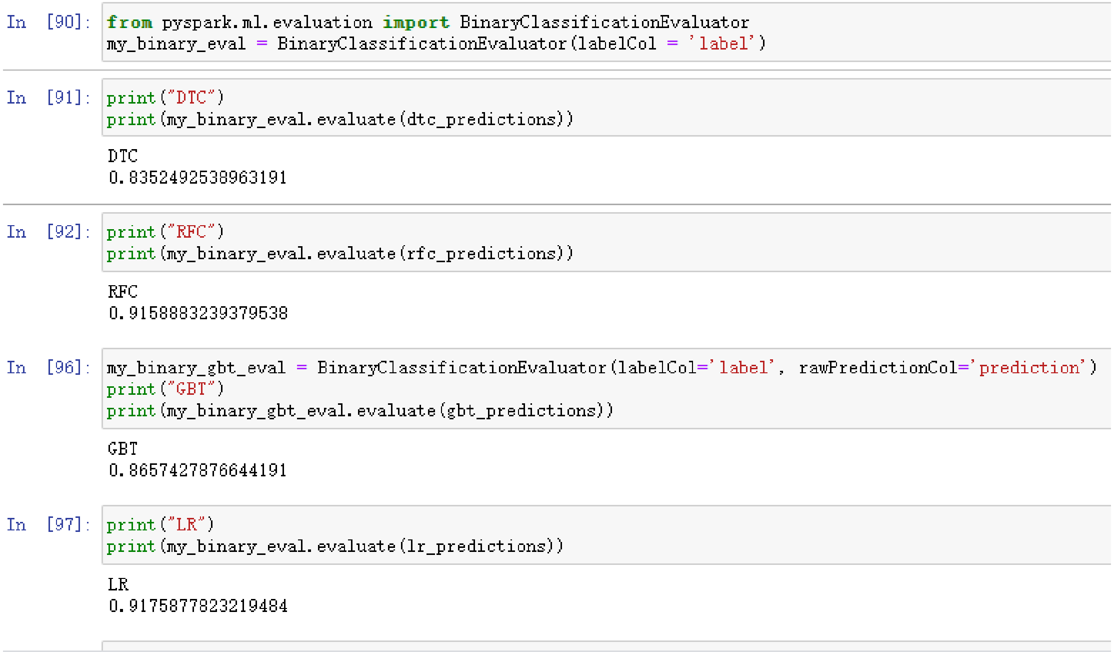


```
In [86]: from pyspark.ml.classification import DecisionTreeClassifier
         #from pyspark.ml import Pipeline
         dtc = DecisionTreeClassifier(labelCol='label',featuresCol='features')
         dtc_model = dtc.fit(train)
         dtc_predictions = dtc_model.transform(test)
```

```
In [87]: from pyspark.ml.classification import RandomForestClassifier
         rfc = RandomForestClassifier(labelCol='label',featuresCol='features')
         rfc_model = rfc.fit(train)
         rfc_predictions = rfc_model.transform(test)
```

```
In [88]: from pyspark.ml.classification import GBTClassifier
         gbt = GBTClassifier(labelCol='label',featuresCol='features')
         gbt_model = gbt.fit(train)
         gbt_predictions = gbt_model.transform(test)
```

```
In [89]: from pyspark.ml.classification import LogisticRegression
         LR = LogisticRegression(labelCol='label',featuresCol='features')
         fit_model = LR.fit(train)
         lr_predictions = fit_model.transform(test)
```

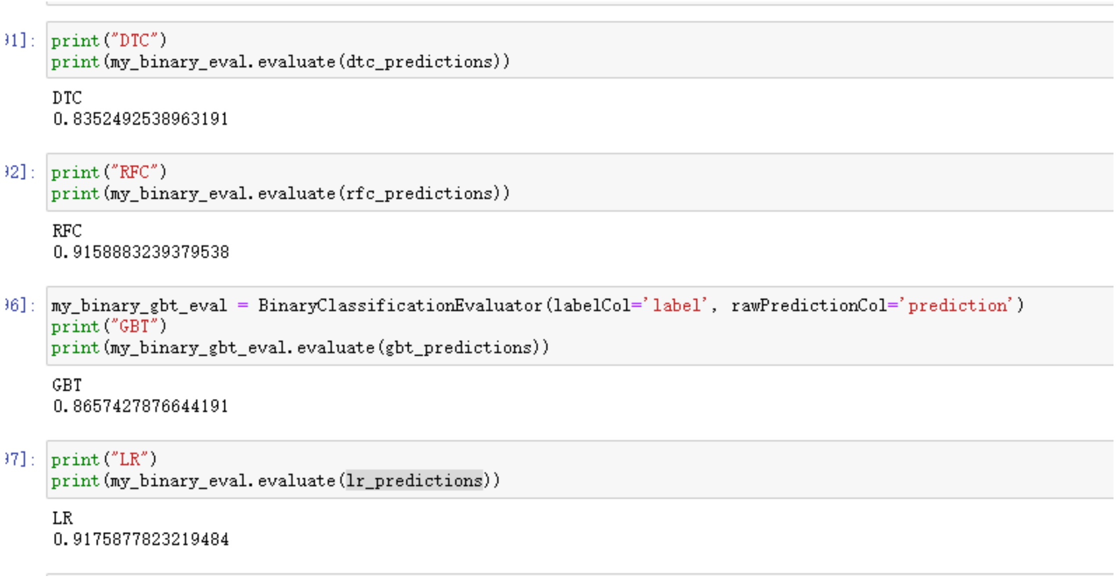


```
In [90]: from pyspark.ml.evaluation import BinaryClassificationEvaluator
         my_binary_eval = BinaryClassificationEvaluator(labelCol = 'label')

In [91]: print("DTC")
         print(my_binary_eval.evaluate(dtc_predictions))

         DTC
         0.8352492538963191

In [92]: print("RFC")
         print(my_binary_eval.evaluate(rfc_predictions))

         RFC
         0.9158883239379538

In [96]: my_binary_gbt_eval = BinaryClassificationEvaluator(labelCol='label', rawPredictionCol='prediction')
         print("GBT")
         print(my_binary_gbt_eval.evaluate(gbt_predictions))

         GBT
         0.8657427876644191

In [97]: print("LR")
         print(my_binary_eval.evaluate(lr_predictions))

         LR
         0.9175877823219484
```

7.3 Search for patterns and document the model's output.

There are many patterns to choose from. In order to make the model as simple and understandable as possible, I only chose three methods, A single decision tree, a random forest and a gradient boosted tree classifier all of which meet the requirements of the data set, cooperate so that I have many discoveries.

As can be seen from the figure below, the accuracy of this model is relatively high. For the prediction of the results is more accurate, parameter setting is also relatively simple, easy to understand. This is a very useful model for predicting the survival and mortality of new crown pneumonia cases.

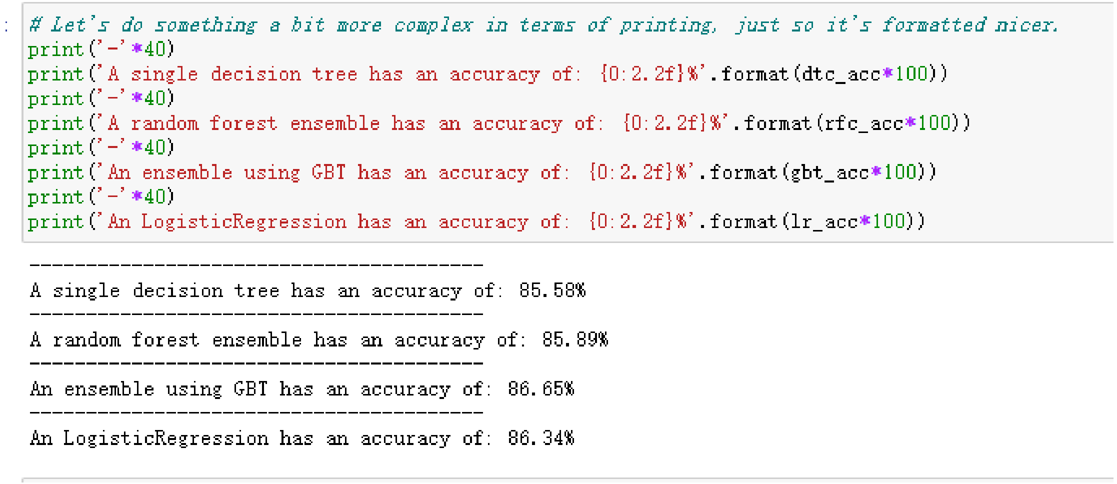


```
print(”DTC”)
print(my_binary_eval.evaluate(dtc_predictions))

DTC
0.8352492538963191
```

```
print(”RFC”)
print(my_binary_eval.evaluate(rfc_predictions))

RFC
0.9158883239379538
```

```
my_binary_gbt_eval = BinaryClassificationEvaluator(labelCol=’label’, rawPredictionCol=’prediction’)
print(”GBT”)
print(my_binary_gbt_eval.evaluate(gbt_predictions))

GBT
0.8657427876644191
```

```
print(”LR”)
print(my_binary_eval.evaluate(lr_predictions))

LR
0.9175877823219484
```



```
# Let’s do something a bit more complex in terms of printing, just so it’s formatted nicer.
print(’-’*40)
print(’A single decision tree has an accuracy of: {0:2.2f}%’.format(dtc_acc*100))
print(’-’*40)
print(’A random forest ensemble has an accuracy of: {0:2.2f}%’.format(rfc_acc*100))
print(’-’*40)
print(’An ensemble using GBT has an accuracy of: {0:2.2f}%’.format(gbt_acc*100))
print(’-’*40)
print(’An LogisticRegression has an accuracy of: {0:2.2f}%’.format(lr_acc*100))
```

```
----------------------------------------
A single decision tree has an accuracy of: 85.58%
----------------------------------------
A random forest ensemble has an accuracy of: 85.89%
----------------------------------------
An ensemble using GBT has an accuracy of: 86.65%
----------------------------------------
An LogisticRegression has an accuracy of: 86.34%
```

8.1 Study and discuss the mined patterns.

Carry out an in-depth discussion about the data, results, models and patterns.

(1) data and result

The data range is only from the Toronto area, and the geographical scope is relatively small. In addition, due to the different cultural habits, living habits and protection measures for COVID-19, the infected population may be different. The outcome of the forecast may change depending on the location. The selection of data increases the spatial limitation of the prediction.

In addition, this data set mainly includes data from January 2020. In the months when the epidemic is very severe, data statistics may be affected by the epidemic, resulting in information loss or incorrect data input, which may increase the error.

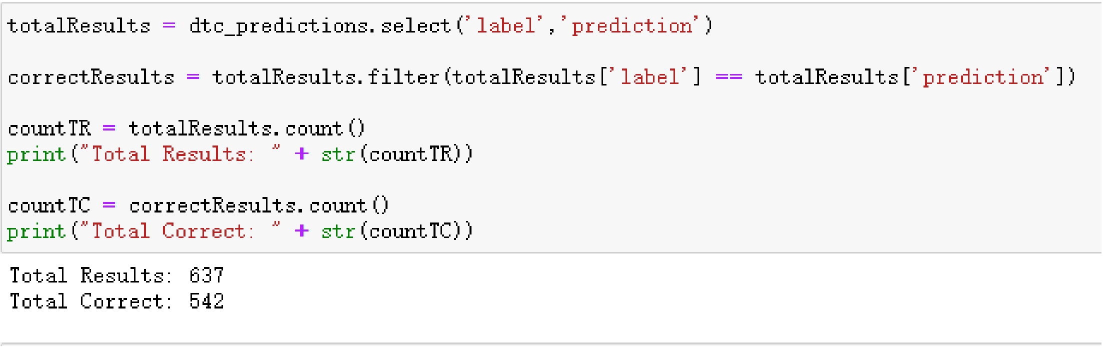
(2) models and patterns

Object data dominate my data set, and the target value is a value in a flag format. Identify the factors that influence the mortality or survival of COVID-19 by distinguishing between the high mortality and the low mortality groups. Predict which populations are vulnerable, and help people increase their survival rates. So I chose the decisiontree classifier, gbtclassifier, random forest classifier and LogisticRegression. These three methods help us to identify which types of characteristics people are more likely to die, with an accuracy of more than 85%, which shows the performance of the models.

What models what types of novel coronavirus pneumonia or those who have characteristics are more likely to be infected with new crown pneumonia. For example, novel coronavirus pneumonia is a very useful predictor of age and whether it is a new crown pneumonia. This helps us to attach special importance to the protection of the elderly. Novel coronavirus pneumonia can be eliminated by novel coronavirus pneumonia. After research, gender factors have little effect on whether they have a new crown, which eases the panic of women. What is the first mock exam for the diagnosis of suspected cases? Whether the treatment is taken or when treatment is taken. If the suspected case is predicted to be sick, he can take corresponding treatment and isolation as soon as possible, so as to prevent the disease from becoming more serious or even infecting more people during the waiting period for diagnosis.

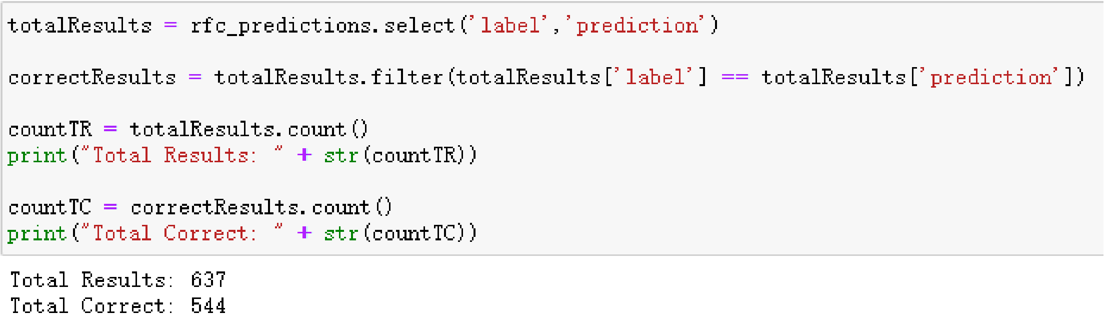
8.2 - Visualize the data, results, models and patterns in a clear and effective manner.

(1) results, models and pattern



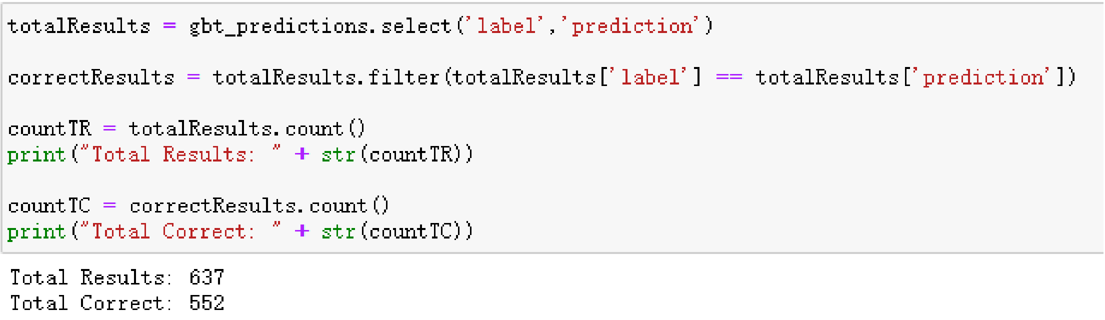
```
totalResults = dtc_predictions.select('label','prediction')

correctResults = totalResults.filter(totalResults['label'] == totalResults['prediction'])

countTR = totalResults.count()
print("Total Results: " + str(countTR))

countTC = correctResults.count()
print("Total Correct: " + str(countTC))
```

```
Total Results: 637
Total Correct: 542
```



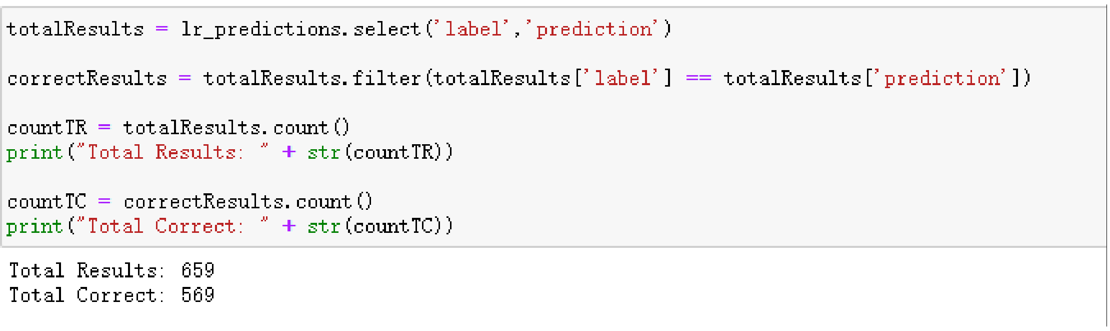
```
totalResults = rfc_predictions.select('label','prediction')

correctResults = totalResults.filter(totalResults['label'] == totalResults['prediction'])

countTR = totalResults.count()
print("Total Results: " + str(countTR))

countTC = correctResults.count()
print("Total Correct: " + str(countTC))
```

```
Total Results: 637
Total Correct: 544
```



```
totalResults = gbt_predictions.select('label','prediction')

correctResults = totalResults.filter(totalResults['label'] == totalResults['prediction'])

countTR = totalResults.count()
print("Total Results: " + str(countTR))

countTC = correctResults.count()
print("Total Correct: " + str(countTC))
```

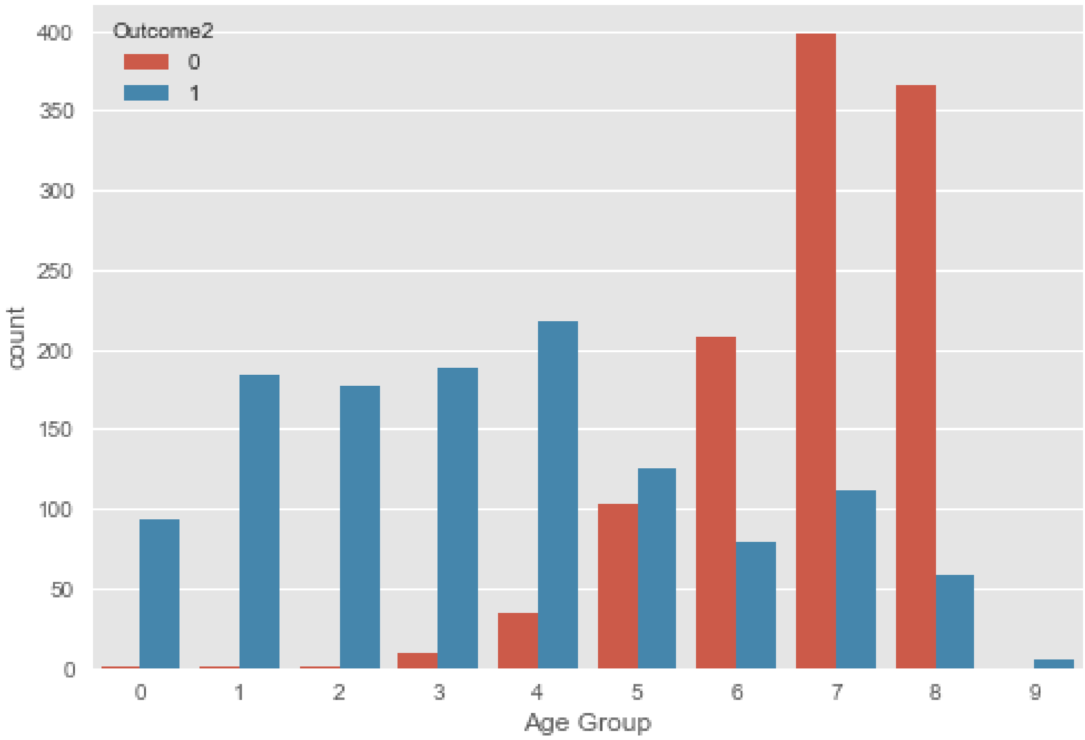
```
Total Results: 637
Total Correct: 552
```

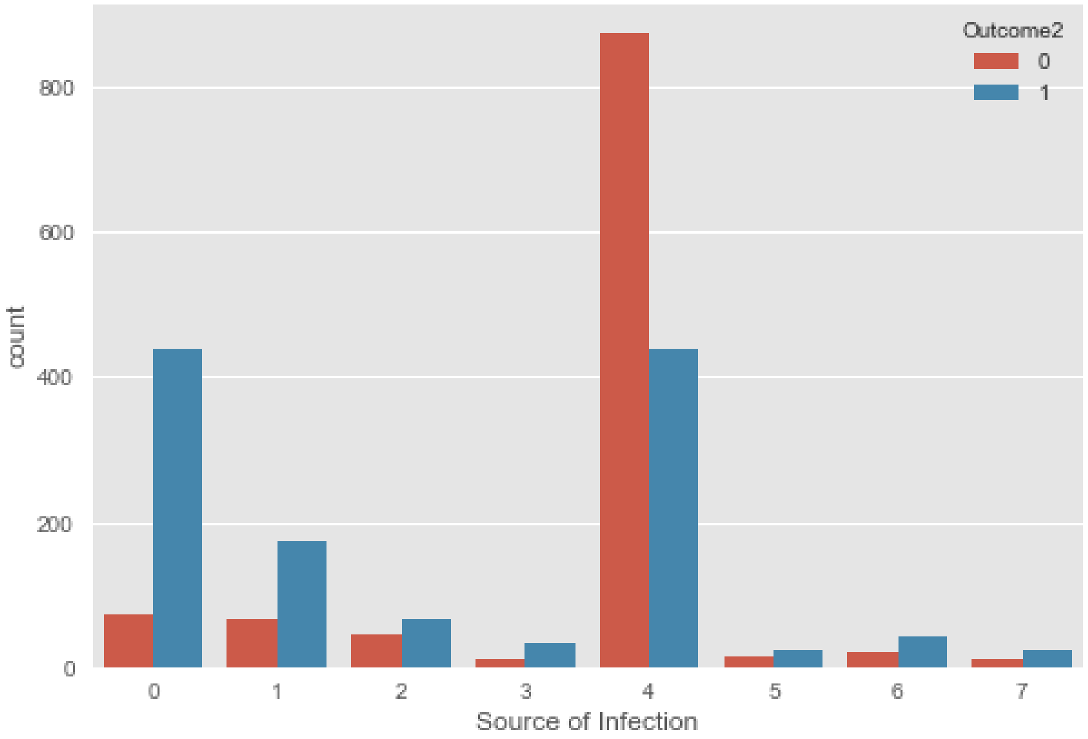


```
totalResults = lr_predictions.select('label','prediction')

correctResults = totalResults.filter(totalResults['label'] == totalResults['prediction'])

countTR = totalResults.count()
print("Total Results: " + str(countTR))

countTC = correctResults.count()
print("Total Correct: " + str(countTC))
```

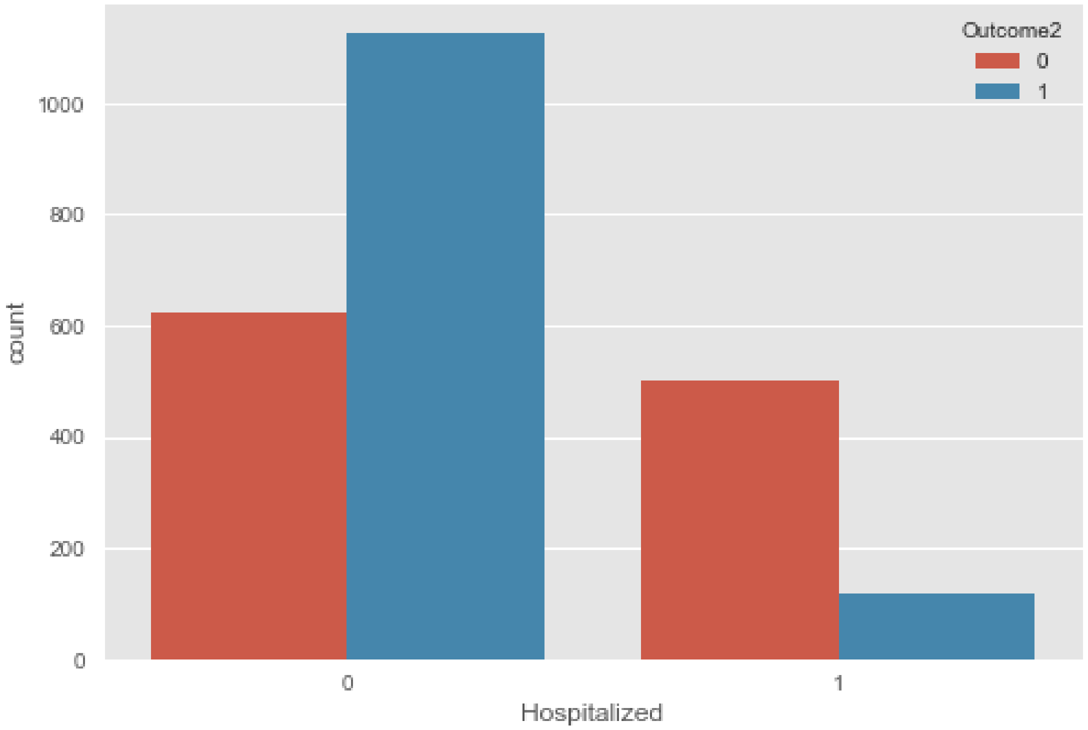
```
Total Results: 659
Total Correct: 569
```

(2) data

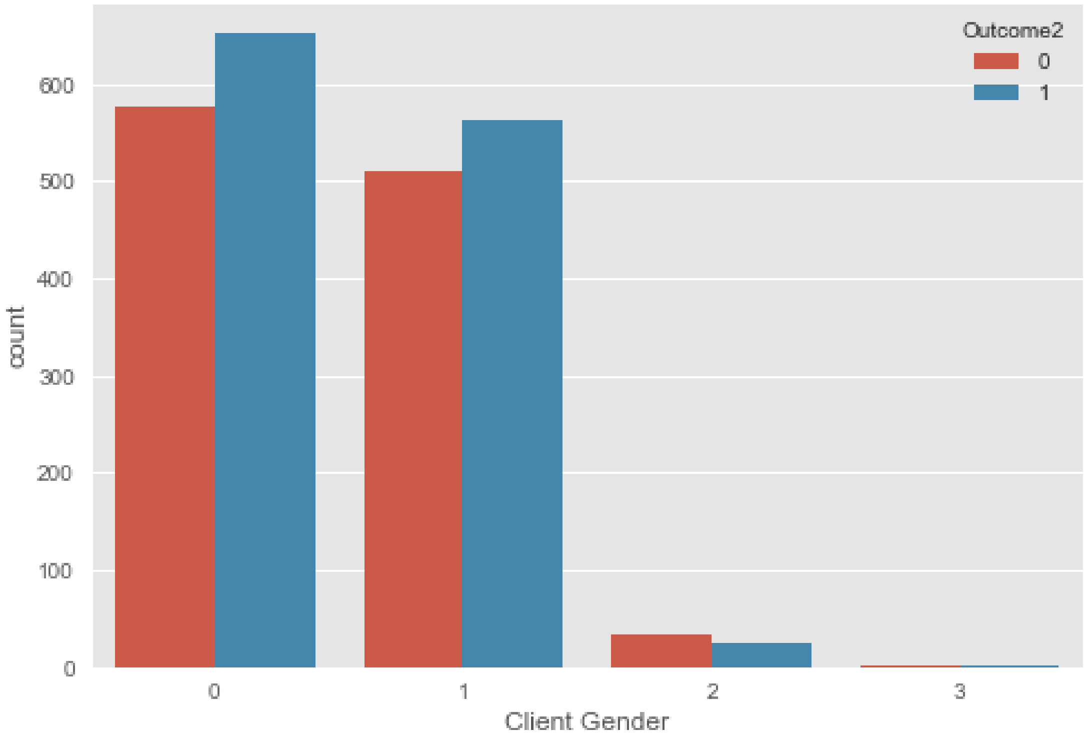




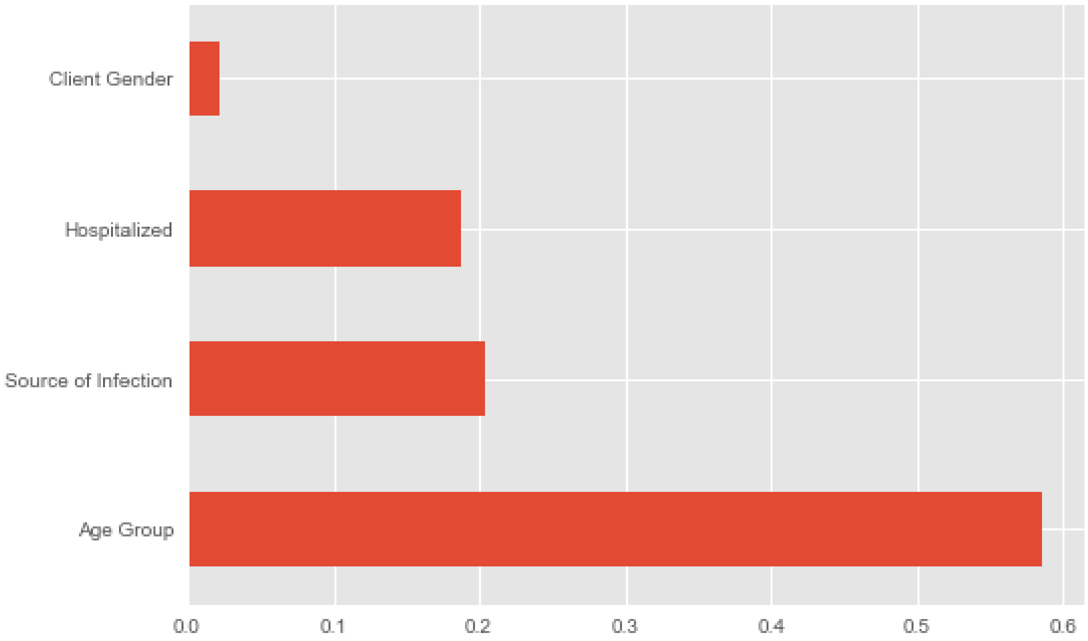
Outcome2
0
1
count
800
600
400
200
0
0
1
2
3
4
5
6
7
Source of Infection



Outcome2
0
1
count
1000
800
600
400
200
0
0
1
Hospitalized



(2). Attribute importance ranking



8.3 Interpret the results, models and patterns showing a clear understanding of the results.

Through the above steps, I have a clear understanding of the project's results, models, and patterns.

(1) Novel coronavirus pneumonia is the most important factor affecting the survival rate of patients with new crown pneumonia, and the number of deaths over 50 years old has even exceeded the survival rate from the results of 8.2. This suggests that we must pay attention to the key protection of the middle-aged and the elderly, which is very important.

Among the factors of infection source, N / a - outbreak associated had the greatest influence. This novel coronavirus pneumonia case is mainly from concentrated infection, or that the cases of concentrated infection are more likely to die. That's what we need to control the concentration of people to control the death toll.

（2）In the construction of the model and understanding of the model, I encountered many difficulties, especially in the application of pyspark, often can not achieve the effect I want. Therefore, I chose for relatively easy to operate models, Decision Tree Classifier, GBT Classifier, Random Forest Classifier and LogisticRegression. Through the evaluation results of the four models, we can find that the precision and recall of the four models are relatively high. This gives us more confidence in using this model to predict the results.

（3） Achieve business goals

According to the bar chart of predictor importance, the characteristics that affect suicide rate are Age Group, Source of Infection, Hospitalized.

The rule set can be used to guide the state, institutions or families to focus on protecting groups of older persons. These groups with lower survival rates are regularly checked and equipped with more effective protective measures.

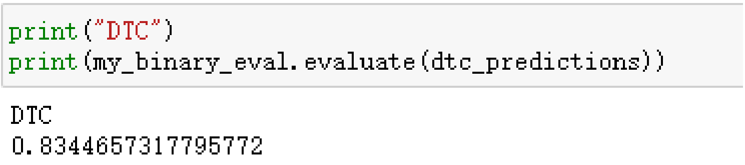
8.4 - Assess and evaluate the results, models and patterns using the appropriate methods/processes.

8.4.1 assess the results, models and patterns

(1) Test Accuracy

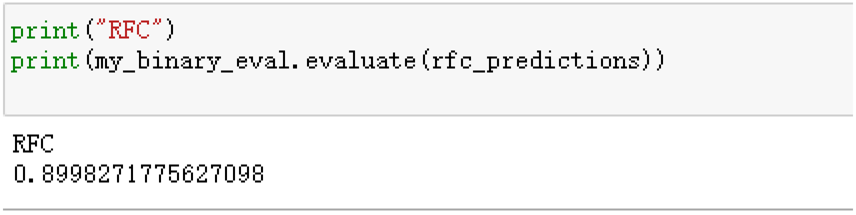
When evaluate the model, and the results are shown below. All of the model have a high accuracy rate. The accuracy rate of single decision tree model is 83.4%. The accuracy rate of LogisticRegression. model is about 92%. The accuracy rate of gradient boosted tree classifier model is 86.3%.

The accuracy rate of Random Forest model is 86.3%.



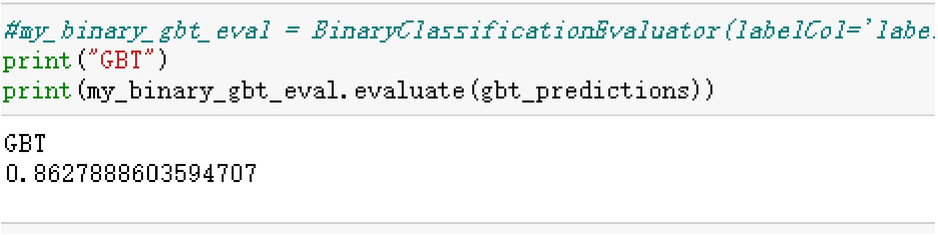
```
print("DTC")
print(my_binary_eval.evaluate(dtc_predictions))
```

```
DTC
0.8344657317795772
```



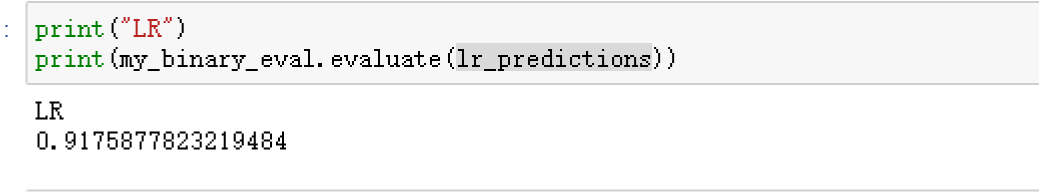
```
print("RFC")
print(my_binary_eval.evaluate(rfc_predictions))
```

```
RFC
0.8998271775627098
```



```
#my_binary_gbt_eval = BinaryClassificationEvaluator(labelCol='labe
print("GBT")
print(my_binary_gbt_eval.evaluate(gbt_predictions))
```

```
GBT
0.8627888603594707
```

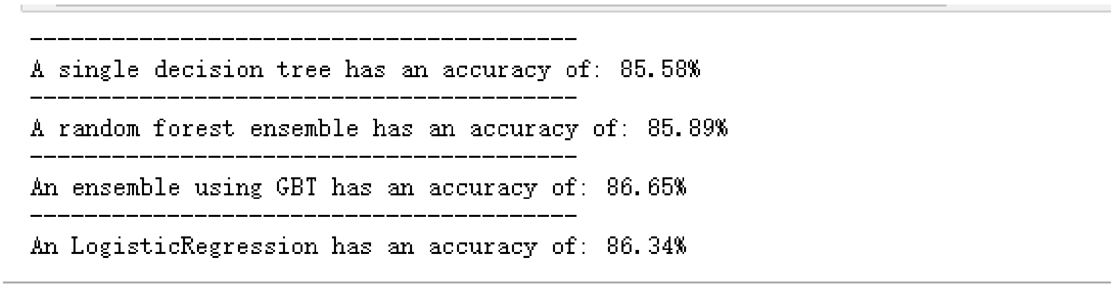


```
print("LR")
print(my_binary_eval.evaluate(lr_predictions))
```

```
LR
0.9175877823219484
```

(2) multi-class classification evaluator

We can use the multi-class classification evaluator to test precision, accuracy and recall directly. The accuracy of these three methods are relatively high, which is about 86%, and the accuracy of gradient boosted tree is the highest, reaching 87%.



```
----------------------------------------
A single decision tree has an accuracy of: 85.58%
----------------------------------------
A random forest ensemble has an accuracy of: 85.89%
----------------------------------------
An ensemble using GBT has an accuracy of: 86.65%
----------------------------------------
An LogisticRegression has an accuracy of: 86.34%
```

8.4.2 evaluation the results, models and patterns

(1) Business goals

People at high risk of COVID-19 can be protected by pre-locating and taking preventive measures.

(2) Data mining goals

Decision rules can be used to predict or classify a group of people with high COVID-19 survival rates, achieving the required model accuracy, but the data quality is poor and may affect the use of the model. The result of the model is logical. The results are easy to understand and easy to deploy

8.5 Iterate prior steps (1 – 7) as required

8.5.1 Business understanding

Covid-19 is one of the most difficult diseases in the world, and there are still tens of thousands of living cases every day in countries that are at risk of death every day. However, the vaccine development process still needs a lot of time. How to reduce the increase of cases in the meantime, who should we care about most. To find out who is more deserving of additional protection and special care, we used a dataset of COVID-19 cases in the Toronto area as a data source to analyze which factors contribute to the survival of infected persons and which factors are associated with case fatality.

8.5.2 data understanding

Data collection, preliminary cognition and processing of data, and certain cognition of data type, data size and processing method. There are assumptions and analysis.

8.5.3 Data preparation

Select data, clean data, merge data, deal with missing value, and format data. Prepare the data required for subsequent modelling.

8.5.4 Data transformation

To further simplify the data set and remove the unimportant and disturbing data, balance the data.

8.5.5 Data-mining method selection

The output variable and the input variable are both String values. In consideration of the target values and methods to predict, I choose supervisory learning and classification.

8.5.6 Data-mining algorithms selection

Pyspark provides many reliable algorithms. After investigation, it was found that these algorithms have their own advantages and disadvantages, so I finally chose single decision tree model , random forest and gradient boosted tree classifier model and .

8.5.7 Data-mining

Through the construction of the model, let me have a feeling of suddenly enlightened. The model clearly shows the importance of the attributes and their relationship to the target value, shaping the prediction.

8.5.8 Interpretation

This step can also involve visualizing the extracted patterns and models, and we evaluate and evaluate the models, results, and their reliability.

Reference

Apache Spark. (n.d). Classification and regression - Spark 3.0.1 Documentation. Retrieved from https://spark.apache.org/docs/latest/ml-classification-regression.html

Data for Data Mining. (n.d.). *Principles of Data Mining*, 11–21. doi:10.1007/978-1-84628-766-4_1.

IBM Knowledge Center (n.d.) Overview of modeling nodes**.** Retrieved from https://www.ibm.com/support/knowledgecenter/en/SS3RA7_18.1.0/modeler_mainhelp_client_ddita/clementine/modeling_nodes.html

Jones, M.(2017). Unsupervised learning for data classification. Retrieved from https://developer.ibm.com/articles/ccunsupervised-learning-data-classification/

Pal, S. (2018). Scikit-learn Tutorial: Machine Learning in Python. Retrieved from https://www.dataquest.io/blog/sci-kit-learn-tutorial/

Pedregosa *et al.,* （2011）. Scikit-learn: Machine Learning in Python. *JMLR 12,* pp. 2825-2830. Retrieved from https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Usama, F., Gregory, P.-S., & Padhraic, S. (1996) Knowledge discovery and data mining: toward a unifying framework. In *Proceeding of The Second Int. Conference on Knowledge Discovery and Data Mining*, pages 82-88, 1996.

"I acknowledge that the submitted work is my own original work in accordance with the University of Auckland guidelines and policies on academic integrity and copyright.
(See: https://www.auckland.ac.nz/en/students/forms-policies-and-guidelines/student-policies-and-guidelines/academic-integrity-copyright.htmlLinks to an external site.).

I also acknowledge that I have appropriate permission to use the data that I have utilized in this project. (For example, if the data belongs to an organization and the data has not been published in the public domain then the data must be approved by the rights holder.) This includes permission to upload the data file to Canvas. The University of Auckland bears no responsibility for the student's misuse of data."