COVID-19 Capstone Project

Mohamed Elsayed

November 2nd, 2021

```
#################################################################
# WARNING: Due to the size of this Dataset & the use of randomForest
# model, running this code might take some time to finish processing
#################################################################
```

I. Introduction

Alberta has been one of the hardest hit places around the world with the fourth wave of the pandemic. Lack of restrictions on social events, relatively low vaccination rates & the Delta variant being highly infectious drove the daily case numbers to unprecedented levels. Having higher daily case counts than Ontario which has three times the population of Alberta prompted national news headlines like "Western provinces driving Canada's fourth wave" & led to the government of Alberta declaring a state of public health emergency. Armed forces were sent to help the health sector & patients were being flown out of province for treatment as ICU capacity was reaching critical levels.

As a resident of Alberta & being affected by this situation, I wanted to create an algorithm that could predict deaths due to COVID-19 with **Accuracy exceeding 90%**. I have downloaded the case data between March 6, 2020 and October 12, 2021 in csv format from the Alberta government website to my GitHub repository & will be automatically downloaded to R when the code runs.

The data includes more than 297,000 entries & 7 variables like Gender, Age group, Zone, Status & Date. In this project I will focus on predicting the status (Died) based on the Age group, Gender, Health Zone & Case Type data available. I will be using **logistic regression, decision tree and random forest** models to try to reach my goal in model Accuracy. My main focus will be on the **"Sensitivity" and "Balanced Accuracy"** outputs of each model as I aim to correctly predict deaths. Since this will be a classification machine learning algorithm, there will be some data preparation done before fitting my models which will be explained in the sections below.

II. Methods

This is the longest section in this report in which I will outline the general methodology I used in this project. This section will be divided into two main parts:

1. Data Exploration through Visualization & the insights gained from it.
2. Data preparation & Building the prediction models

The methods used in this project aim at developing a classification machine learning algorithm that predicts deaths due to COVID-19 based on various parameters. As mentioned above, Balanced Accuracy will be the main metric I will use to evaluate model performance although an equally important metric in the model evaluation process will be **Sensitivity**.

The dataset will be split in 70-30 ratio as follows:

1. `train` set will hold 70% of the data for model development.
2. `test` set will hold 30% of the data for model evaluation.

1. Data Exploration & Visualization

Below I will import the data into Rstudio then rename the columns to remove spaces & simplify names then remove unknown values. Also I've chosen to eliminate the case status "Active" from the dataset as this is naturally updated to either "Recovered" or "Died". Then will get a glimpse of the dataset.

```
# Download from GitHub
covid_alberta <- read_csv("https://raw.githubusercontent.com/mofouadelsayed/Harvard.Capstone/main/covid

# Rename Columns
colnames(covid_alberta)<- c("Number", "Date", "Zone", "Gender", "Age_group", "Status",
                            "Type")

# Exclude Unknown entries
covid_alberta<- covid_alberta %>% filter(!Age_group== "Unknown") %>%
  filter(!Gender== "Unknown") %>% filter(!Status== "Active")

as_tibble(head(covid_alberta))
```

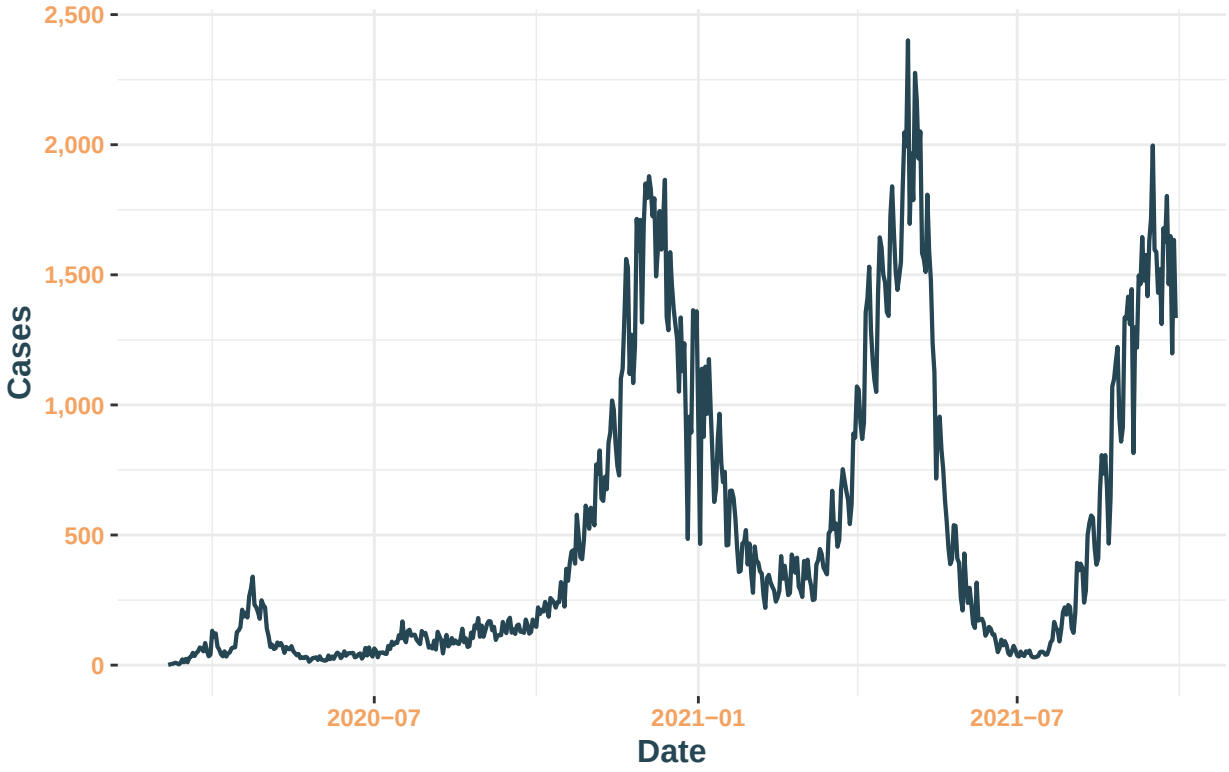
```
## # A tibble: 6 x 7
##   Number Date       Zone          Gender Age_group   Status    Type
##    <dbl> <date>     <chr>         <chr>  <chr>       <chr>     <chr>
## 1      1 2020-11-13 Calgary Zone  Female 1-4 years   Recovered Confirmed
## 2      2 2021-04-21 Edmonton Zone Male   30-39 years Recovered Confirmed
## 3      3 2021-05-17 North Zone    Male   10-19 years Recovered Confirmed
## 4      4 2020-12-13 Edmonton Zone Male   5-9 years   Recovered Confirmed
## 5      5 2021-01-05 Central Zone  Male   50-59 years Recovered Confirmed
## 6      6 2021-05-11 Edmonton Zone Male   60-69 years Recovered Confirmed
```

Now that the data is imported I will start exploring the dataset to start building the approach with which I will develop the prediction models. I will start by checking the dimensions of the data & what's the type of each variable then will try to gather meaningful information for various visualizations as pointed below.

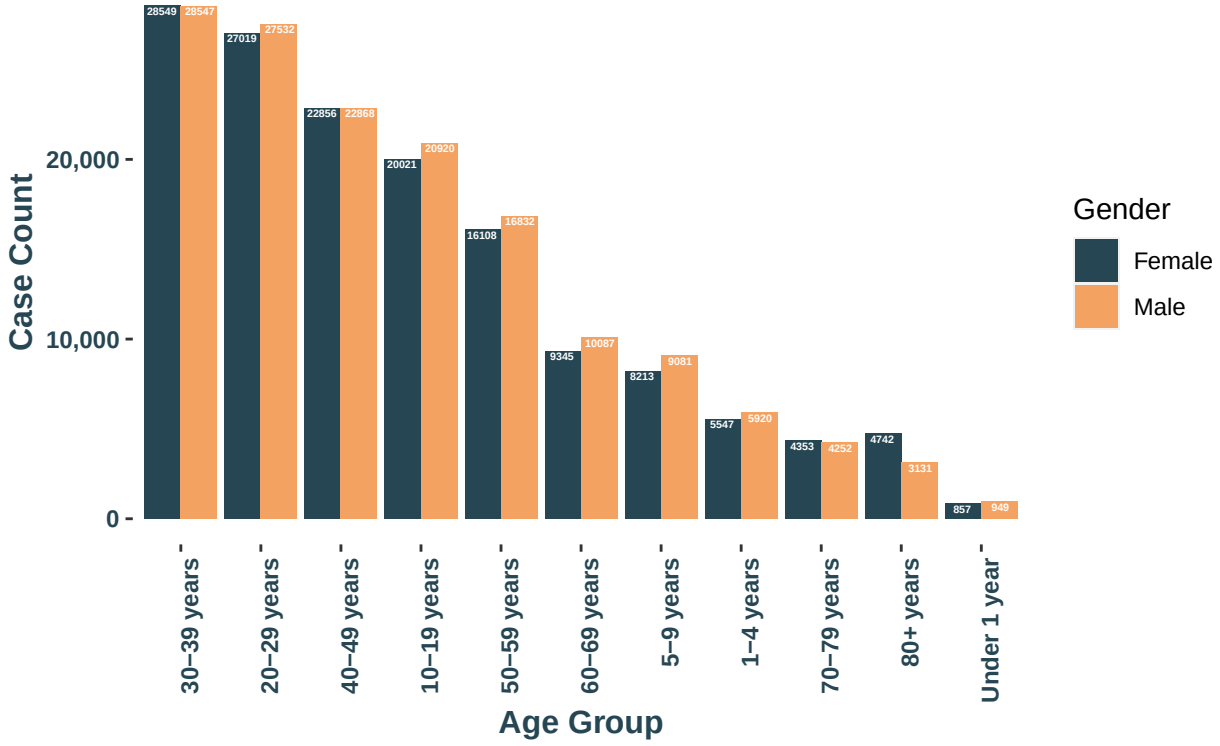
```
# Dimensions & variable classes of the dataset
glimpse(covid_alberta)
```

```
## Rows: 297,729
## Columns: 7
## $ Number    <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1~
## $ Date      <date> 2020-11-13, 2021-04-21, 2021-05-17, 2020-12-13, 2021-01-05,~
## $ Zone      <chr> "Calgary Zone", "Edmonton Zone", "North Zone", "Edmonton Zon~
## $ Gender    <chr> "Female", "Male", "Male", "Male", "Male", "Male", "Female", ~
## $ Age_group <chr> "1-4 years", "30-39 years", "10-19 years", "5-9 years", "50-~
## $ Status    <chr> "Recovered", "Recovered", "Recovered", "Recovered", "Recover~
## $ Type      <chr> "Confirmed", "Confirmed", "Confirmed", "Confirmed", "Confirm~
```

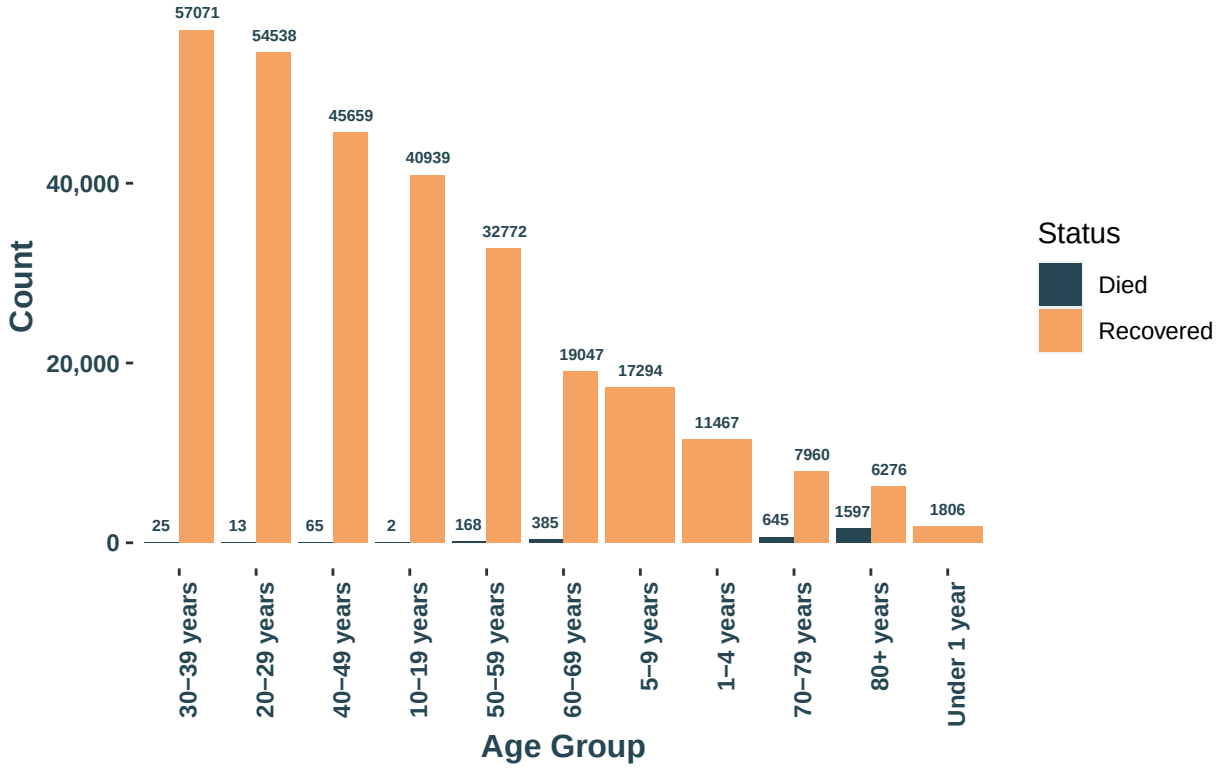
Daily COVID Cases in Alberta



Case Counts by Age Group & Gender

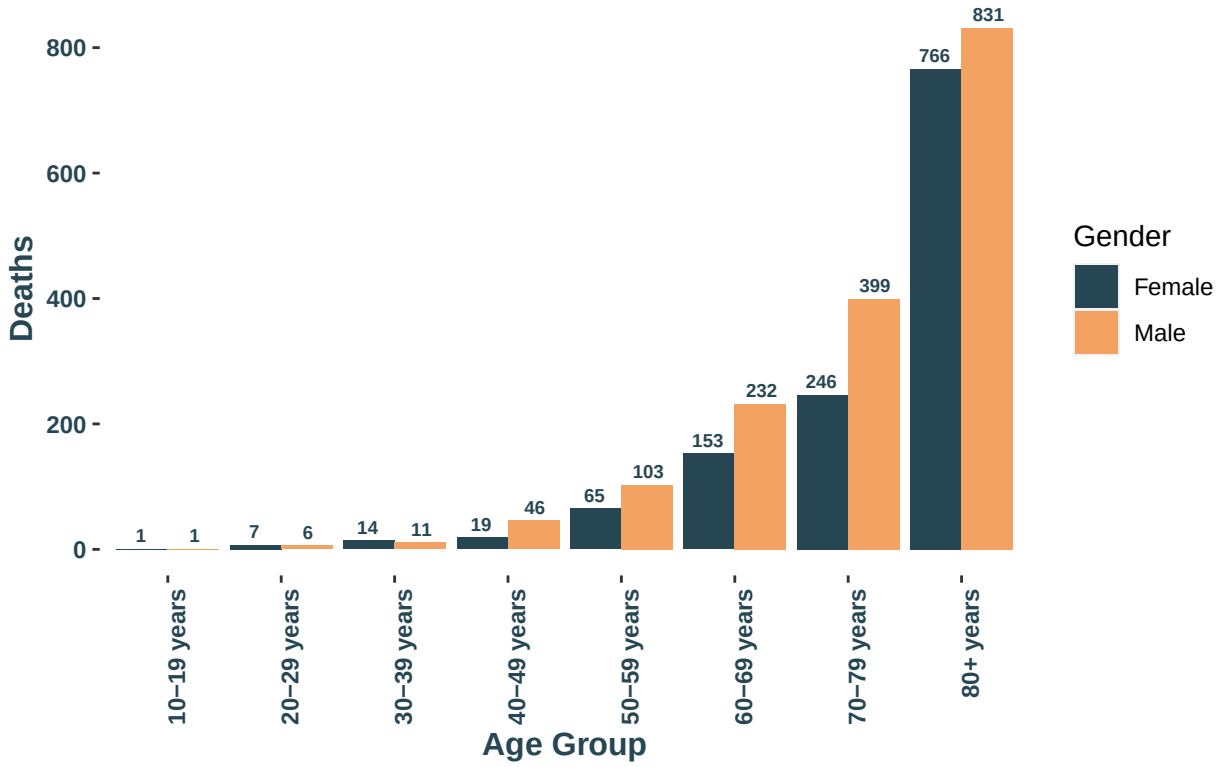


Recovery Status by Age Group

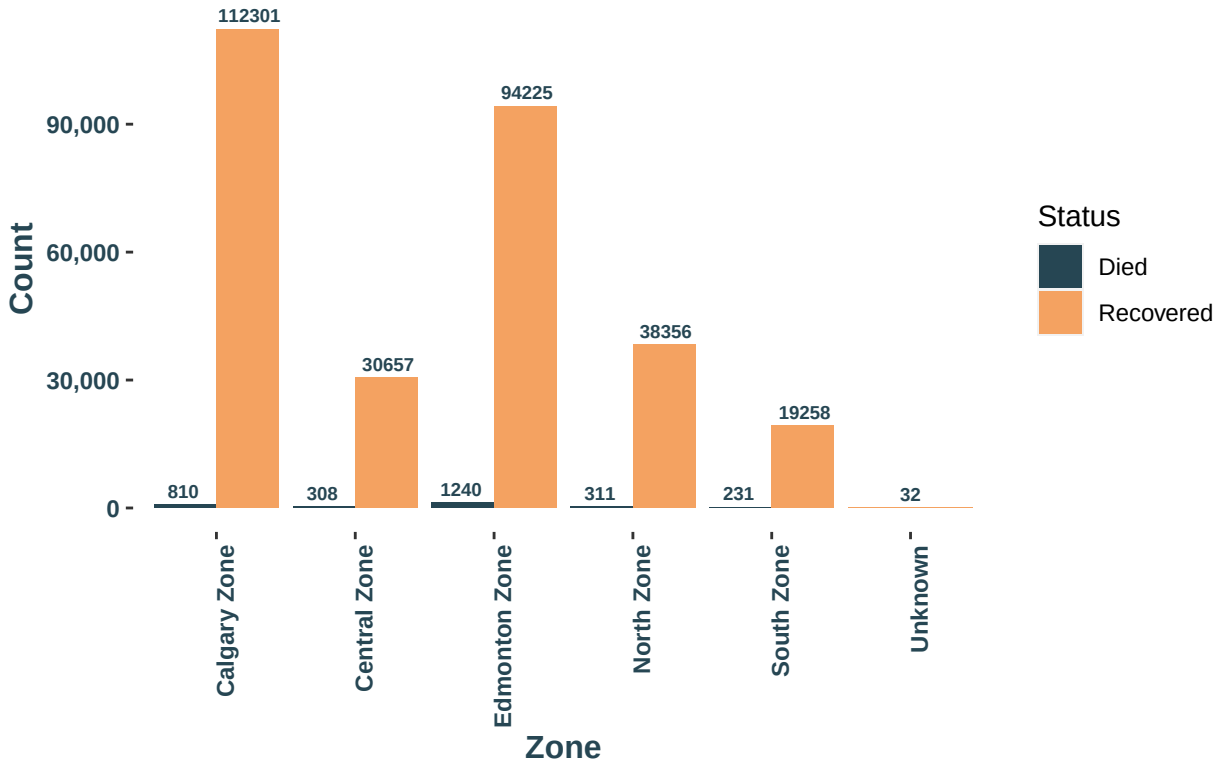


Recovery Status by Gender

COVID Deaths by Age Group & Gender



Zone Cases Vs. Deaths



```r
# Percentage of total deaths from COVID
mean(covid_alberta$Status== "Died")*100
```

```
## [1] 0.9740402
```

After exploring the structure of the dataset & the plots above, there are a few insights observed that are going to be useful in preparing the data for modeling and in knowing what are the expectations from the variables.

1. All variables are in "character" format which will need to be converted into "factors" to fit into the classification models. Also the number & date variables need to be removed.
2. Death rate is significantly small **(<0.1% of the data)** which implies that the dataset is highly imbalanced. This will constitute a challenge for machine learning algorithms.
3. The first four plots show no significance of any variable affecting death rates but It's very visible on the fifth plot that the majority of deaths happened in age groups above 60 years old.
4. The last plot reveals that although Calgary has the highest recorded number of cases, it's actually Edmonton that recorded the highest number of deaths.

2. Preparing the data & Builing the prediction models

Since the majority of data variables are in character format, everything will need to be converted into factors. But when it comes to the health zones, that needed to be converted in a different way because zones are not related to each other, not like age for example. So I have used `dummyVars` function in the "caret" package to expand the health zones into separate columns. If a COVID case is recorded in a particular zone, that zone will have a value of "1" with all other zones on the same row will show "0". So in summary, it's changing the health zone into a binary variable.

Then the `train` & `test` sets of the data will be created to start building the models

```r
# Converting the Type, Status & Gender variables to factors to be able to
# feed them into the models.
covid_alberta$Type<-  factor(covid_alberta$Type,
                levels = c('Confirmed', 'Probable'),
                labels = c(1, 0))

covid_alberta$Status<-  factor(covid_alberta$Status,
                  levels = c('Recovered','Died'),
                  labels = c(0, 1))

covid_alberta$Gender<- factor(covid_alberta$Gender,
                  levels = c('Male', 'Female'),
                  labels = c(1, 0))

covid_alberta$Age_group<- factor(covid_alberta$Age_group,
                     levels = c('Under 1 year','1-4 years','5-9 years',
                                '10-19 years','20-29 years',
                                '30-39 years','40-49 years','50-59 years','60-69 years',
                                '70-79 years','80+ years'),
                     labels = c(0,1,2,3,4,5,6,7,8,9,10))
```

```
# Converting Zones by dummyVars
covid_alberta$Zone<- factor(covid_alberta$Zone, exclude = NULL)
covid_alberta$Zone<- addNA(covid_alberta$Zone)
dummy<- caret::dummyVars("~Zone", data= covid_alberta)
df<- data.frame(predict(dummy, newdata = covid_alberta))
covid_alberta<- cbind(covid_alberta, df)

# Removing the Zone Column as replaced by the dummyVars outpput & Remove Date & Number
covid_alberta<- covid_alberta %>% select(c(-Zone, -Date, -Number))
as_tibble(head(covid_alberta))
```

```
## # A tibble: 6 x 11
##   Gender Age_group Status Type  Zone.Calgary.Zone Zone.Central.Zone
##   <fct>  <fct>     <fct>  <fct>             <dbl>              <dbl>
## 1 0      1         0      1                     1                  0
## 2 1      5         0      1                     0                  0
## 3 1      3         0      1                     0                  0
## 4 1      2         0      1                     0                  0
## 5 1      7         0      1                     0                  1
## 6 1      8         0      1                     0                  0
## # ... with 5 more variables: Zone.Edmonton.Zone <dbl>, Zone.North.Zone <dbl>,
## #   Zone.South.Zone <dbl>, Zone.Unknown <dbl>, Zone.NA <dbl>
```

```
# Creating the train & test sets
set.seed(123, sample.kind = "Rounding")
ind<- createDataPartition(covid_alberta$Status, times= 1, p=0.7, list= FALSE)
train<- covid_alberta[ind,]
test<- covid_alberta[-ind,]
```

Below I will start building the prediction models, the logistic regression model with the data as it is will be my baseline model for predicting deaths.

- **1: Unbalanced data glm model (Baseline model)**

```
unbal_glm_fit<- train %>% glm(Status ~ ., data=., family = "binomial")
unbal_glm_pred<- ifelse(predict(unbal_glm_fit, newdata = test, type = "response")
                        >0.5, 1, 0)

Accuracy_results <- tibble(Method = "Unbalanced glm",
                           Balanced_Accuracy = confusionMatrix(as.factor(unbal_glm_pred),
                           test$Status, positive = "1")$byClass["Balanced Accuracy"])
```

```r
confusionMatrix(as.factor(unbal_glm_pred), test$Status, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 88448   870
##          1     0     0
##
##                Accuracy : 0.9903
##                  95% CI : (0.9896, 0.9909)
##     No Information Rate : 0.9903
##     P-Value [Acc > NIR] : 0.509
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.99026
##              Prevalence : 0.00974
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : 1
##
```

As Seen above, the Confusion matrix resulted in a misleading high Accuracy because more than 99% of the case status is recovered "0". It's very visible as Sensitivity is 0% & Balanced Accuracy is 50%. This will remain my Baseline model & I will try to fix the data imbalance in the dataset by "Over-Sampling" the "Died" entries & "Under-Sampling" the "Recovered" so that the algorithm is trained on data that is not biased. Then I will apply the best performing method to the "Decision Tree" & "Random Forest" models.

```r
# Unbalanced data showing status "1" (Died) is Less than 1% of the total data
table(train$Status)
```

```
##
##      0      1
## 206381   2030
```

I will Balance the Data using over & under sampling then will apply both methods to the logistic regression model to see which performs better.

```
# Over-Sampling
set.seed(123, sample.kind = "Rounding")
over<- ovun.sample(Status ~ ., data=train, method = "over", N= 412762)$data
table(over$Status)
```

```
##
##      0      1
## 206381 206381
```

```
# Under-Sampling
set.seed(123, sample.kind = "Rounding")
under<- ovun.sample(Status~ ., data= train, method = "under", N= 4060)$data
table(under$Status)
```

```
##
##    0    1
## 2030 2030
```

- **2: Balanced glm model using Over-Sampling**

```
bal_glm_fit_over<- over %>% glm(Status ~ ., data=., family = "binomial")
bal_glm_pred_over<- ifelse(predict(bal_glm_fit_over, newdata = test, type = "response")
                           >0.5, 1, 0)

Accuracy_results <- bind_rows(Accuracy_results,
                              tibble(Method = "Balanced Over-sampling glm",
                       Balanced_Accuracy = confusionMatrix(as.factor(bal_glm_pred_over),
                       test$Status, positive = "1")$byClass["Balanced Accuracy"]))

confusionMatrix(as.factor(bal_glm_pred_over), test$Status, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 78259    78
##          1 10189   792
##
##                Accuracy : 0.8851
##                  95% CI : (0.8829, 0.8871)
##     No Information Rate : 0.9903
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1177
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.910345
##             Specificity : 0.884802
```

```
##          Pos Pred Value : 0.072125
##          Neg Pred Value : 0.999004
##              Prevalence : 0.009740
##          Detection Rate : 0.008867
##    Detection Prevalence : 0.122943
##       Balanced Accuracy : 0.897574
##
##        'Positive' Class : 1
##
```

- **3: Balanced glm model using Under-Sampling**

```
bal_glm_fit_under<- under %>% glm(Status ~ ., data=., family = "binomial")
bal_glm_pred_under<- ifelse(predict(bal_glm_fit_under, newdata = test, type = "response")
                            >0.5, 1, 0)

Accuracy_results <- bind_rows(Accuracy_results,
                              tibble(Method = "Balanced Under-Sampling glm",
                      Balanced_Accuracy = confusionMatrix(as.factor(bal_glm_pred_under),
                      test$Status, positive = "1")$byClass["Balanced Accuracy"]))

confusionMatrix(as.factor(bal_glm_pred_under), test$Status, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 78269    79
##          1 10179   791
##
##                Accuracy : 0.8852
##                  95% CI : (0.883, 0.8872)
##     No Information Rate : 0.9903
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1177
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.909195
##             Specificity : 0.884915
##          Pos Pred Value : 0.072106
##          Neg Pred Value : 0.998992
##              Prevalence : 0.009740
##          Detection Rate : 0.008856
##    Detection Prevalence : 0.122820
##       Balanced Accuracy : 0.897055
##
##        'Positive' Class : 1
##
```

Judging by the results below & the Sensitivity in the confusion matrix for both models, it appears that the results from the "Over-Sampling" method performs slightly better so it will be the method which I will apply to the two remaining models.

```
Accuracy_results %>% knitr::kable()
```

Method	Balanced_Accuracy
Unbalanced glm	0.5000000
Balanced Over-sampling glm	0.8975736
Balanced Under-Sampling glm	0.8970554

- **4: Decision Tree Balanced using Over-Sampling**

```
bal_ctree_over<- over %>% ctree(Status ~ ., data=.)
bal_ctree_over_pred<- predict(bal_ctree_over, test)

Accuracy_results <- bind_rows(Accuracy_results,
                              tibble(Method = "Balanced Over-Sampling Ctree",
                              Balanced_Accuracy = confusionMatrix(bal_ctree_over_pred,
                              test$Status, positive = "1")$byClass["Balanced Accuracy"]))

confusionMatrix(bal_ctree_over_pred, test$Status, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 77349    62
##          1 11099   808
##
##                Accuracy : 0.875
##                  95% CI : (0.8729, 0.8772)
##     No Information Rate : 0.9903
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1103
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.928736
##             Specificity : 0.874514
##          Pos Pred Value : 0.067859
##          Neg Pred Value : 0.999199
##              Prevalence : 0.009740
##          Detection Rate : 0.009046
##    Detection Prevalence : 0.133310
##       Balanced Accuracy : 0.901625
##
##        'Positive' Class : 1
##
```

- **5: Random Forest using Over-Sampling**

```
rf<- over %>% ranger(Status ~ ., data=., mtry = 8, num.trees = 200)
```

```
## Growing trees.. Progress: 44%. Estimated remaining time: 39 seconds.
## Growing trees.. Progress: 88%. Estimated remaining time: 8 seconds.
```

```
rf_pred<- predict(rf, test)

Accuracy_results <- bind_rows(Accuracy_results,
                              tibble(Method = "Balanced Over-Sampling Random Forest",
                              Balanced_Accuracy = confusionMatrix(rf_pred$predictions,
                              test$Status, positive = "1")$byClass["Balanced Accuracy"]))

confusionMatrix(rf_pred$predictions, test$Status, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 77423    64
##          1 11025   806
##
##                Accuracy : 0.8758
##                  95% CI : (0.8737, 0.878)
##     No Information Rate : 0.9903
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1108
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.926437
##             Specificity : 0.875350
##          Pos Pred Value : 0.068126
##          Neg Pred Value : 0.999174
##              Prevalence : 0.009740
##          Detection Rate : 0.009024
##    Detection Prevalence : 0.132459
##       Balanced Accuracy : 0.900894
##
##        'Positive' Class : 1
##
```

III. Results

All three models applied to the "Over-Sampling" data showed significant improvement over the baseline model as seen in the summary table below. Although the high imbalance in the data & the absence of numerical values in the dataset impacted model performance, all three model results were close to each other with the "Decision Tree" model slightly out-performing the linear regression & the random forest algorithms when it comes to Sensitivity. **Please Note:** I have used the `ranger` function to run the random forest algorithm as it's designed to work much faster on larger datasets, I have also limited the tuning parameters of that model so that running the code doesn't crash the Rstudio session.

```
Accuracy_results %>% arrange(desc(Balanced_Accuracy)) %>% knitr::kable()
```

Method	Balanced_Accuracy
Balanced Over-Sampling Ctree	0.9016247
Balanced Over-Sampling Random Forest	0.9008936
Balanced Over-sampling glm	0.8975736
Balanced Under-Sampling glm	0.8970554
Unbalanced glm	0.5000000

IV. Conclusion

Exploring the data up to the date of download reveals that death from COVID-19 is around the same figures reported by media & health organizations. It also reveals that up to that date, the majority of deaths from COVID-19 happen in ages 60 Years & above. The unavailability of the vaccination status & the strength of the social restrictions applied at the time when each case was recorded affected the accuracy of all models. The size of the data also impacted the final results of this project which could have been improved with applying techniques like "Cross-Validation" & "Random-Searching" for the random forest model parameter tuning.

Future work & further development would include adding vaccination status, the strength of social restrictions at the time of recording each case & tuning the random forest parameters would be very helpful in achieving higher prediction accuracy.