Covid-19 Survival Optimisation Modeling Analysis

(Word count: 2,145)

By

Aman upadhyay

Noida,India.

Table of contents

Introduction.....2

Problem Definition.....2

Objectives.....2

Data Source.....2

Implementation.....3

Early data analysis.....3

Modelling.....4

Conclusion.....9

Individual reflective reports.....9

References:.....20

Introduction

Coronavirus 2019 (COVID-19) is an infectious disease caused by coronavirus SARS-CoV2 (World Health Organization, 2020). Individuals infected with covid-19 showed signs of mild to moderate respiratory illness, dyspnoea, and ground-glass opacity demonstrated by imaging tests and fever, and in extreme cases, infected individuals died because of covid-19 (Huang et al., 2020).

Various researchers and virologist reported on the determining factors of the survival of patients with covid-19. Risk of death and severity of symptoms were reported higher in patients with comorbidities such as pulmonary, immune system pathology, diabetes, hypertension and smoking (Richardson et al., 2020). (Galvão et al., and Justino Fernández, 2020).

Problem Definition

Our motivation for choosing this project is to understand through analysis the impact of comorbidities and other features on the survival or death of a sample of covid-19 patients. Following our analysis, we intend to build a robust Machine Learning model that helps to predict a patient's survival.

Objectives

- Perform a descriptive analysis of covid-19 dataset using statistical techniques and plots, as well as displaying relationship between comorbidities and survival of a covid patient.
- Perform data cleaning by engineering features and dropping features not relevant to this project scope.
- Use Random Forest and Gaussian Naive Bayes models that predict if a patient will die or survive covid-19 given patient's current symptoms. Furthermore, perform cross validation and checks to see the performance of the models.

Data Source

Our data was obtained from https://www.kaggle.com/ in CSV format. Ethical considerations were discussed and as shown in *fig.1* data was publicly available for everyone for research purpose.

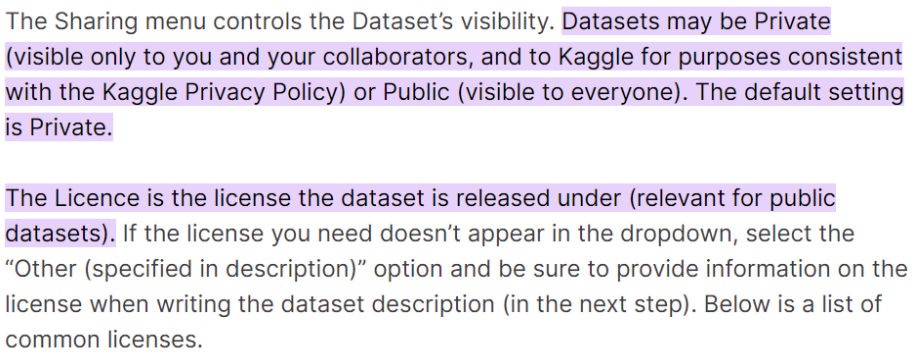


Figure 1: Licencing screenshot for dataset.

Implementation

In its raw form, data had 21 columns and 199999

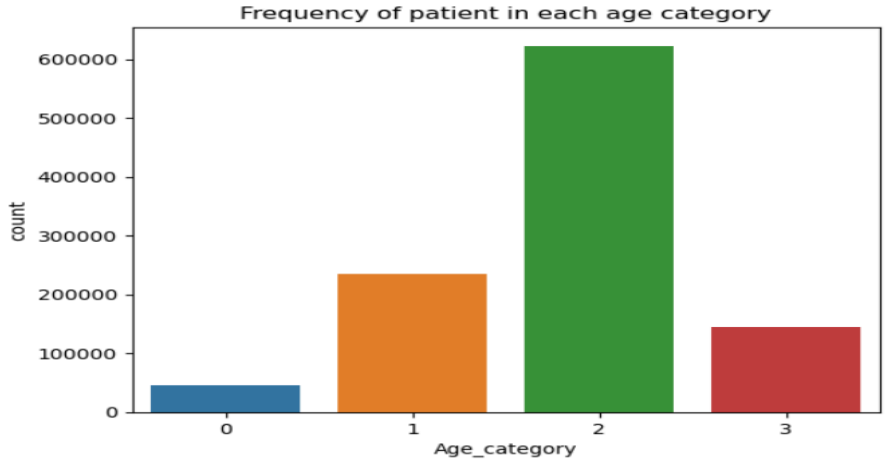
rows, and after cleaning, there were 74409

rows and 22 columns in our data.

Early data analysis

Data shows that most of the columns were encoded, other columns we manually encoded as guided by data dictionary in preparation for model building. [ICU & INTUBATED] columns were dropped because they had over 80% null-values and avoid altering data trueness through imputation.

Figure 2: Countplot for age category



Age column was grouped into 4 age categories, children age (0-15), adult (15-30), old (31-60) and senior (61-120) are category 0,1,2,3,4 respectively. Count plots *fig:2* show 2, have the highest count followed by 3, 1 and 0 in that order.

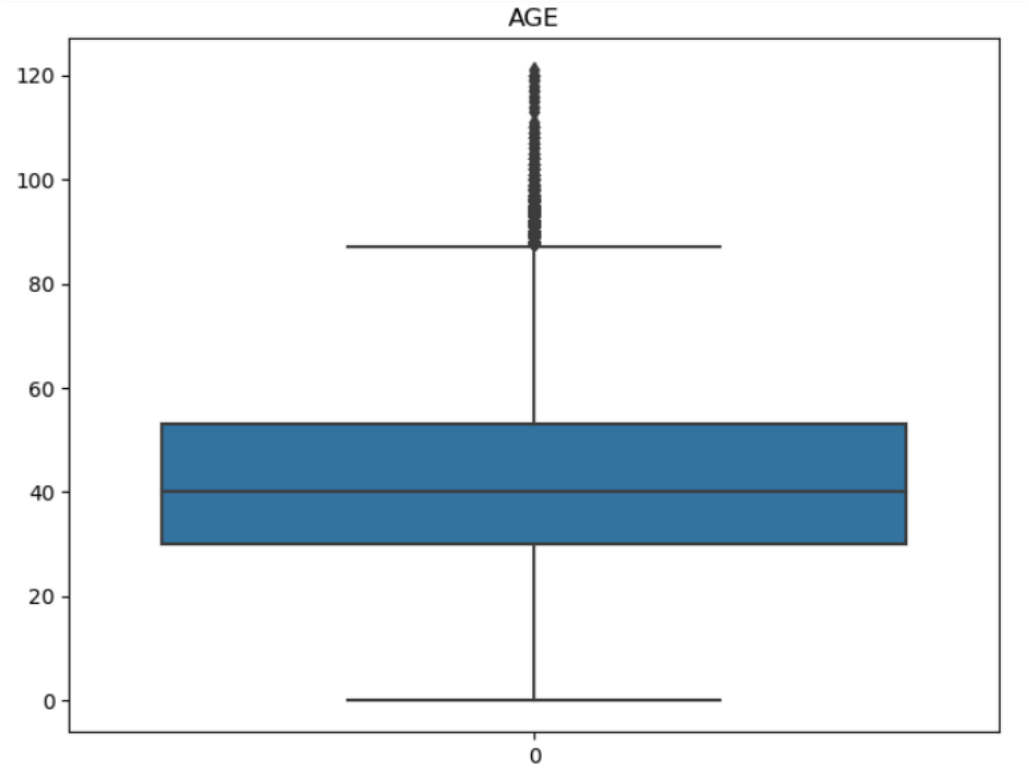


Figure 3: Boxplot for age

Boxplot*(fig 3)* shows that the majority of the patients are aged between 0 and approximately 95 years old, with some outliers above 100. Domain knowledge suggests, covid-19 has severe effects on old patients compared to younger patients, hence outliers are significant. 0 years old could be babies that are less than a year old. *Fig:4* is a correlation plot of death and survival patients in each age category. Overall, more people survived covid-19 in our sample data (*fig 5)*.

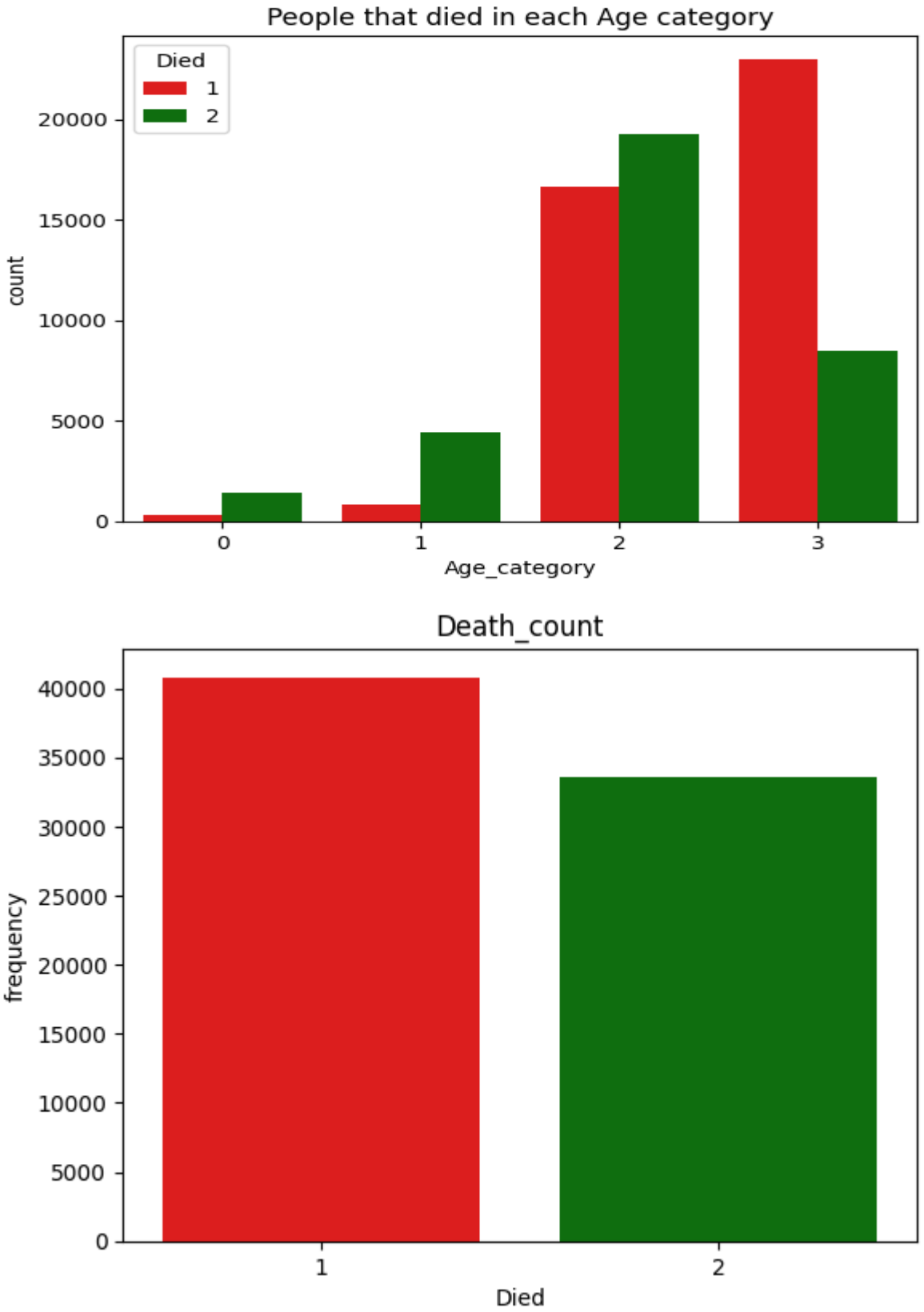


Figure 5: Proportion of died to survived.

Modelling

Why Gaussian Naive Bayes and Random Forest (GNB and RF)?

We chose the died column as our target to build a model that predicts if a covid-19 patient survived covid-19 or not given independent variables. We chose these models because both are used for classification tasks. RF builds multiple decision trees and merges their predictions; therefore, it is more accurate and robust to outliers and noisy data. GNB is a probabilistic supervised learning model, that uses conditional probability to classify target.

Results:

Random Forest:

```
[ ]  # Splitting our target feature

     X = Covid_Data  # Features independent variable
     y = Covid_data['Died']  # Label of dependent variable (target variable)
```

```
     # Split dataset into training set and test set
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20,  random_state = 11) # 80% training and 20% test

     X.shape, y.shape, X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((48365, 20), (48365,), (38692, 20), (9673, 20), (38692,), (9673,))
```

Figure 6: Random Forest fitting

RF was fitted on at (10,20,30,40) % testing size on n_estimators (50,100,150) and max_depth of 3. **Max_depth** represents depth of each individual decision tree and **n_estimators** represent the number of decision trees that will be built in the random forest, both are hyperparameters that helps us build a more robust model.

As shown in *Fig 7*, when **n_estimators**= 150 and **Max_depth** =3 **precision** of RF was (63,62,62,64) at testing size of (10,20,30,40) % respectively and **accuracy** of 89 across 4 testing trail.

```
Accuracy: 0.7378269409697095
Precision: 0.7190416492547709
Accuracy: 0.74
Precision: 0.72
```

Figure 7: Evaluation of n_estimator of 150 and max_depth of 3

Fig 8 shows when **n_estimators**= 100 and **Max_depth** =3 **precision** of RF was (63,62,62,64) at testing size of (10,20,30,40) % respectively and **accuracy** of 89 across 4 testing trail, obviously there was no improvement to the precision and accuracy.

n_estimators = 100

1. Test_size = 40%

Accuracy: 0.8916889528903518 Precision: 0.6204328726025929 Accuracy: 0.89 Precision: 0.62 2. Test_size = 30% Accuracy: 0.892323253102928 Precision: 0.624457622215794 Accuracy: 0.89 Precision: 0.62 3. Test_size = 20% Accuracy: 0.893257560172804 Precision: 0.6302702702702703 Accuracy: 0.89 Precision: 0.63 4. Test_size = 10% Accuracy: 0.8933861345402181 Precision: 0.6363438520130577 Accuracy: 0.89 Precision: 0.64

Figure 8: Evaluation of n_estimator of 100 and max_depth of 3

Fig 9 shows when **n_estimators**= 50 and **Max_depth** =3 **precision** of RF was (73,71,69,73) and **accuracy** of (87,88,88,87) at testing size of (10,20,30,40) % respectively. Our model predicted more accurately and precisely when reduced the n_estimators to 50.

RF Confusion matrix:

Fig 10 shows model predicted 2017 false positives (FP) than true positives, the FP are patients that did not die from covid-19 and model predicted them dead. However, model predicted more true-negative TN than false negative (FN) meaning model predicted to greater extent correctly patient that survived.

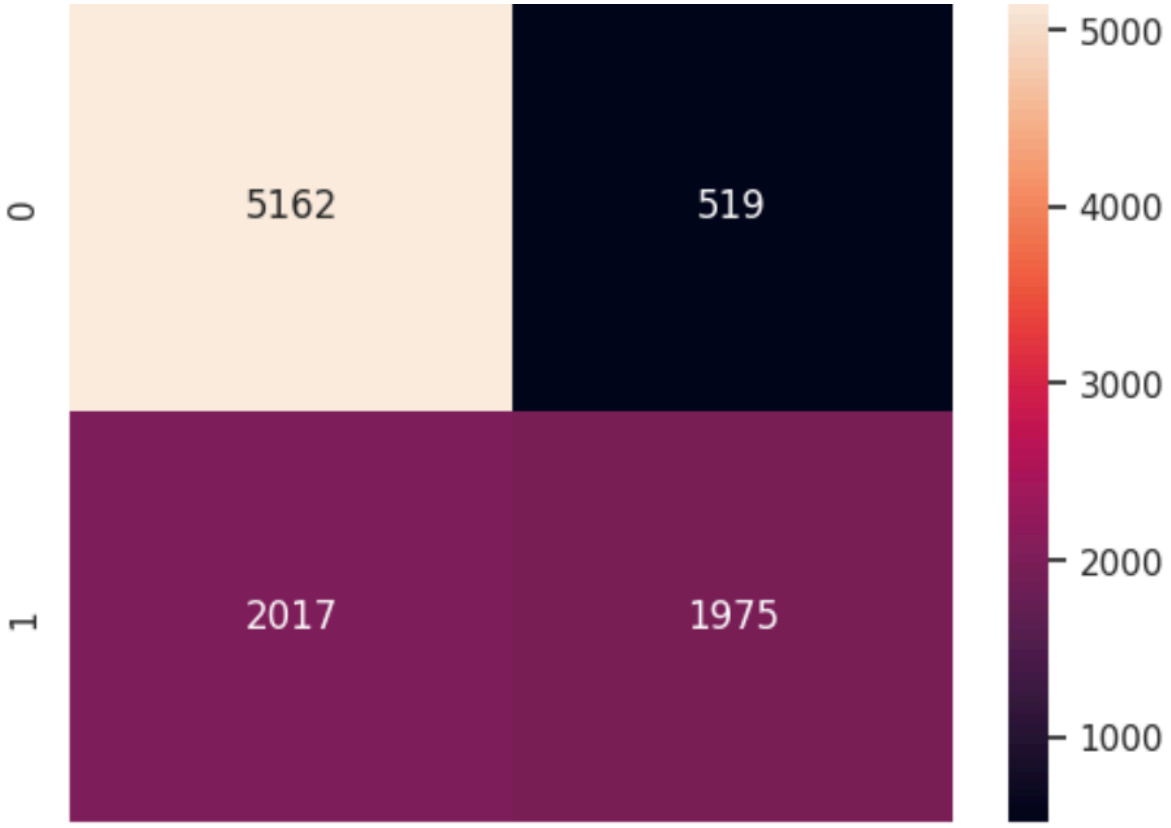


Figure 9: RF confusion matrix before balancing

Gaussian Naive Bayes:

```
# Training
classi = GaussianNB()
classi.fit(X_train, y_train)
```

```
GaussianNB()
```

In a Jupyter environment, please rerun this cell to sh
On GitHub, the HTML representation is unable to re

```
# Predicting the test results
y_pred = classi.predict(X_test)

# Check the actual and predicted value
y_compare = np.vstack((y_test, y_pred)).T
# Displaying 15 values
y_compare[:15,:]
```

Figure 10: GNB implementation

GNB was fitted on at (10,20,30,40) % testing size. As shown in *Fig 10*, when at testing size of (10,20,30,40) % **precision** of GNB was (47,46,46,47) respectively and **accuracy** of 75 across 4 testing trails.

GNB Confusion matrix:

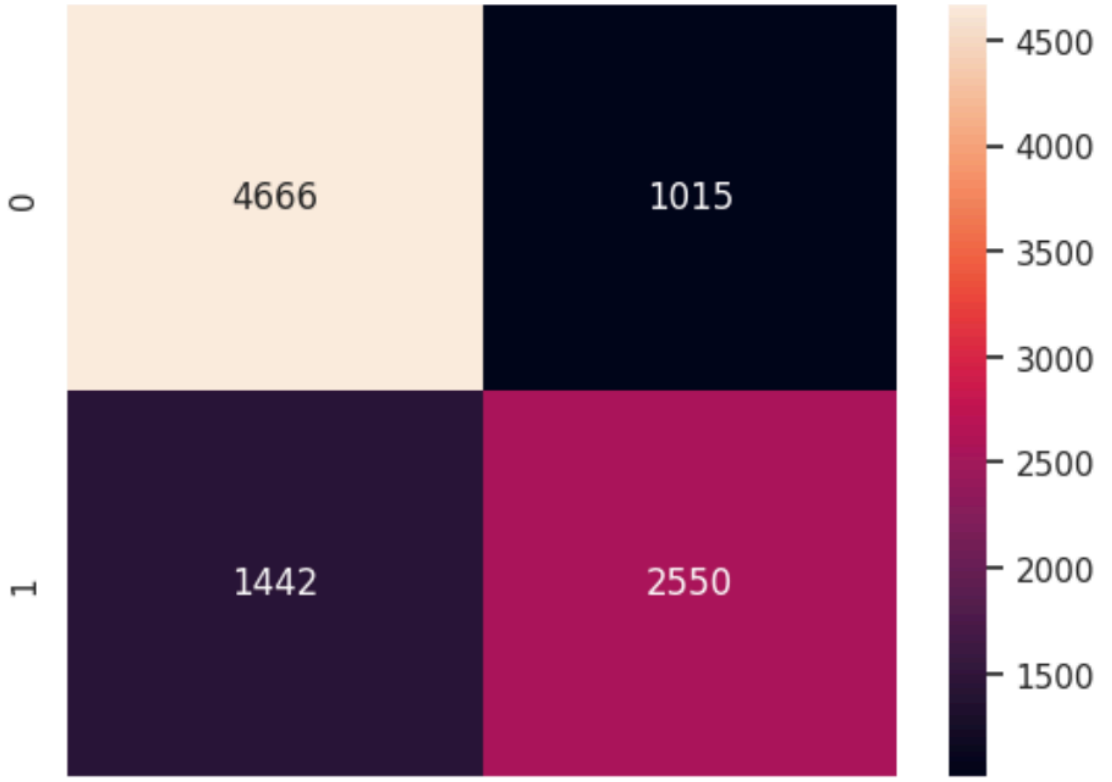


Figure 11: GNB confusion matrix before balancing

Fig 11 shows model predicted 1442 false positives (FP) and 4666 true positives, the FP are patients that did not die from covid, and model predicted them dead. However, model predicted 2550 true negatives (TN) than false negative (FN) meaning model predicted to greater extent correctly patient that survived.

We believe this was due to imbalance in our data, there were more patients that survived than died hence the model was bias toward patients that survived. To improve our model performance, we tried scaling, adjusting our target encoding and PCA but none of these approaches affected our model performance. We did minmax scaling because our data was skewed, and PCA to reduce dimensionality of our data, hence improving our model performance. Finally, we balanced our data because there were more patients that survived than died, hence our model was bias.

Conclusion

Haven carefully evaluated our model results, we can observe that a RF model performed better for this sample data. This model is consistently predicting if a person died or survived Covid-19 with an Accuracy of 76% - 78% with a proportion of actual positives correctly identified (Recall) of 81% (survived) and 95% (died).

The results were obtained using:

1. Metrics from Sklearn to evaluate the Accuracy and Precision.

2. Classification report from Sklearn.metrics to evaluate Precision, Recall, F1-score and Accuracy.

3. Cross validation score from Sklearn.model_selection to evaluate the Accuracy dividing the data into multiple subsets to obtain a reliable metric and compare it with our previous ones.

Reflective reports

As a pair we decided to work on covid-19 case. During the deliberation, I looked for the dataset we used for our project and after we both reviewed the data, we agreed to use it.

Data Preparation (DP):

Data preparation also known as data preprocessing in a crucial step to a reliable data analysis and modelling. This process involves early data analysis (EDA), data cleaning, transformation, handling imbalanced data, feature engineering, handling imbalanced data, visualisation, and feature selection. DP takes over 80% of modelling process because if we put ill-prepared data into our machine learning model, we would get bad results, hence having through DP is imperative to getting a reliable result from our model. During our project, I prepared our data for analysis.

EDA:

EDA helps us to understand the dataset by summarizing its characteristics and often visualising them with plots. During this step I was able to familiarise with the dataset, what columns we have in our data, and how we can use them for our analysis. Furthermore, with EDA I was able to visualise distribution patterns in numerical variable for better understanding *fig 12.*

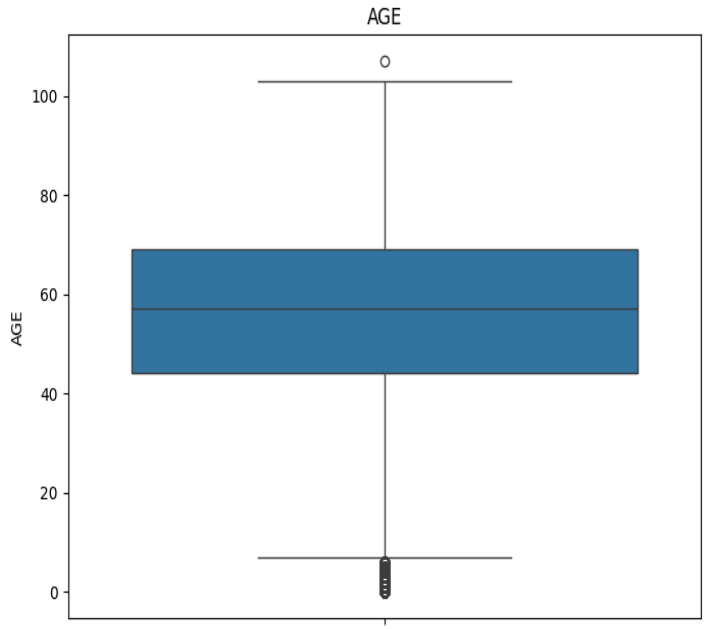


Figure 12: Boxplot for age

Data Cleaning (DC):

DC is the process of making our data clean for modelling, a task I handled during our project. DC involves but not limited to handling missing values, engineering features, transforming data. Initially 'isnull()' assessment showed no null-values, further exploration revealed missing-values coded as integers (95-98). To handle this, I converted them to NAN in other to pinpoint their locations and quantity. [ICU, INTUBATED] columns had over 80% NAN, to prevent introducing bias through imputation method I dropped them and used 'dropna()' I dropped other rows with NAN values.

Feature engineering:

The Date_died variable contained patient that survived and died in '999-999-99' and 'dd/mm/yy' format respectively, In other to extract our target variable from this column, I wrote a function that iterates through the variable and return 1 if patient died and 2 if patient survived as shown in *fig.18* I then assigned this call function to a new column called 'died', hence extracting our target variable.

```
Name: DATE_DIED, Length: 401, dtype: int64
```

```
19]:  # Died= df['DATE_DIED'].value_counts()
      def get_die(value):
          if (value=='9999-99-99'):
              return 2
          else:
              return 1
```

```
20]:  df['Died']= df['DATE_DIED'].apply(get_die)
      df.head()
```

Figure 18: Function to create target variable.

Manipulating data to improve model performance:

Our Initial machine learning setup had poor precision. To Optimise model performance, my approach involves dropping classification finals, scaling independent variables *fig.19,* and balancing the dataset *fig.20.*

MinMax scaller is good for skwed data because it scales the observation to be between 0 and 1

```
.06]: from sklearn.preprocessing import MinMaxScaler
      scaler = MinMaxScaler()
      covid_scaled = scaler.fit_transform(df3)
```

Figure 19: Scaling the independent variables.

I scaled both random forest and Gaussian Naive Bayes models, averaging accuracies at 0.89% and precisions of 0.55. Despite adjusting the target value and eliminating noise, improvements in

GNB 4

```
[136]: # SMOTE (to balance the data)

       X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state = 11)
       smt = SMOTE()

       X_train, y_train = smt.fit_resample(X_train, y_train)
       X_test, y_test = smt.fit_resample(X_test, y_test)
```

Figure 20: Balancing x and y variables.

accuracy and precision were limited. Utilizing Principal Component Analysis (PCA) *fig.21* aims to reduce dimensionality, enhancing model performance by curbing noise in dataset. Additionally, scaling the dependent variable ensures uniformity, a pivotal feature engineering step for improved model accuracy.

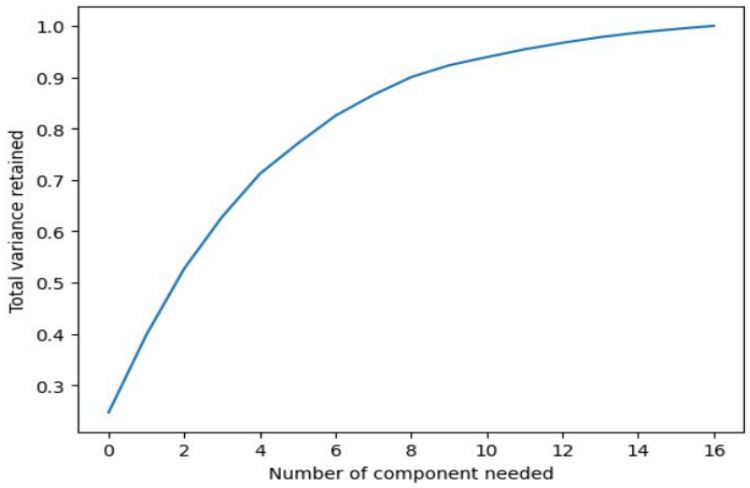


Figure 21: PCA plotting.

Conclusion:

Data preprocessing is an important process in any data analysis and modelling process. “Garbage in Garbage out” meaning the quality of result is dependent on the quality of data

passed into machine learning algorithm. My main contribution during our project was to prepare our data. After preparation during modelling process, I also contributed to manipulating our data, to optimise model performance. After completion of our coding, I drafted our report in collaboration with my group mate.

Leopoldo Rojo Romero

Contributions

Following the start of the Exploratory Data Analysis (EDA), we realized that we have got missing values in our data, thus I performed the following code to find the percentage of missing values in each feature.

Percentage of missing values / Column

```
Covid_data.isnull().mean()*100 # Porcentage of missing data
```

	0
USMER	0.000000
MEDICAL_UNIT	0.000000
SEX	0.000000
PATIENT_TYPE	0.000000
INTUBED	0.000000
PNEUMONIA	2.377401
PREGNANT	0.000000
DIABETES	0.000000
COPD	0.000000
ASTHMA	0.000000
INMSUPR	0.000000
HIPERTENSION	0.000000
OTHER_DISEASE	0.000000
CARDIOVASCULAR	0.000000
OBESITY	0.000000
RENAL_CHRONIC	0.000000
TOBACCO	0.000000
CLASIFFICATION_FINAL	35.001142
ICU	0.000000
Died	0.000000
AGE_Cat	0.000000
colors	0.000000

Figure 13: Missing values in each feature (%).

After we notice that two of our features had more than 80% of missing values, we decided to remove them due to the did not hold enough data for our analysis objectives.

```
[ ] # Split dataset into training set and test set
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20,  random_state = 11) # 80% training and 20% test

    X.shape, y.shape, X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((48365, 20), (48365,), (38692, 20), (9673, 20), (38692,), (9673,))
```

```
[ ] # Execute Random Forest

    ramF = RandomForestClassifier(n_estimators = 50, max_depth = 3)

    #Train the model using the training sets y_pred=clf.predict(X_test)
    ramF.fit(X_train, y_train)

    y_pred = ramF.predict(X_test)
```

Model Evaluation

```
[ ] # Model Accuracy, how often is the classifier correct?
    print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
    print("Precision:",metrics.precision_score(y_test, y_pred))

    # Rounded upto 2 decimal places
    print( "Accuracy: {:.2f}".format(metrics.accuracy_score(y_test, y_pred)))
    print( "Precision: {:.2f}".format(metrics.precision_score(y_test, y_pred)))
```

```
Accuracy: 0.7378269409697095
Precision: 0.7190416492547709
Accuracy: 0.74
Precision: 0.72
```

Figure 14: Random Forest (RF) model.

```python
from sklearn.metrics import classification_report, confusion_matrix
y_predict = ramF.predict(X_test)
cm = confusion_matrix(y_test, y_predict)
sns.heatmap(cm, annot=True, fmt='g')
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           1       0.72      0.91      0.80      5681
           2       0.79      0.49      0.61      3992

    accuracy                           0.74      9673
   macro avg       0.76      0.70      0.71      9673
weighted avg       0.75      0.74      0.72      9673
```

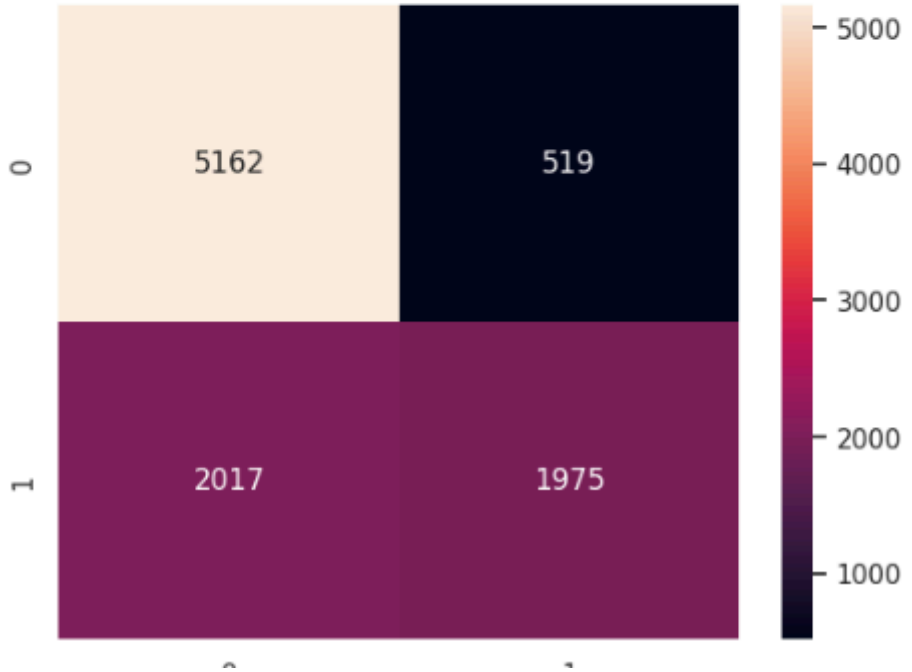


Figure 15: Classification report of the RF model.

Gaussian Naive Bayes

```
[ ] from sklearn.naive_bayes import GaussianNB
```

```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20,  random_state = 11) # 80% training and 20% test

X.shape, y.shape, X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((48365, 20), (48365,), (38692, 20), (9673, 20), (38692,), (9673,))
```

```
[ ] # Training
classi = GaussianNB()
classi.fit(X_train, y_train)
```

```
GaussianNB
GaussianNB()
```

```
# Predicting the test results
y_pred = classi.predict(X_test)

# Check the actual and predicted value
y_compare = np.vstack((y_test, y_pred)).T
# Displaying 15 values
y_compare[:15,:]
```

```
array([[2, 1],
```

Figure 16: Gaussian Naive Bayes (GNB) model.

Model Evaluation

```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision:",metrics.precision_score(y_test, y_pred))

# Rounded upto 2 decimal places
print( "Accuracy: {:.2f}".format(metrics.accuracy_score(y_test, y_pred)))
print( "Precision: {:.2f}".format(metrics.precision_score(y_test, y_pred)))
```

```
Accuracy: 0.7459940039284607
Precision: 0.7639161755075311
Accuracy: 0.75
Precision: 0.76
```

1.Test_size = 40%

Accuracy: 0.7459940039284607 Precision: 0.7639161755075311 Accuracy: 0.75 Precision: 0.76.

2.Test_size = 30%

Accuracy: 0.7359940039284607 Precision: 0.7539161755075311 Accuracy: 0.74 Precision: 0.75

3.Test_size = 20%

Accuracy: 0.7239161755075311 Precision: 0.7439161755075311 Accuracy: 0.74 Precision: 73.

4.Test_size = 10%

Accuracy: 0.7339161755075311 Precision: 0.7339161755075311Accuracy: 0.85 Precision: 0.47

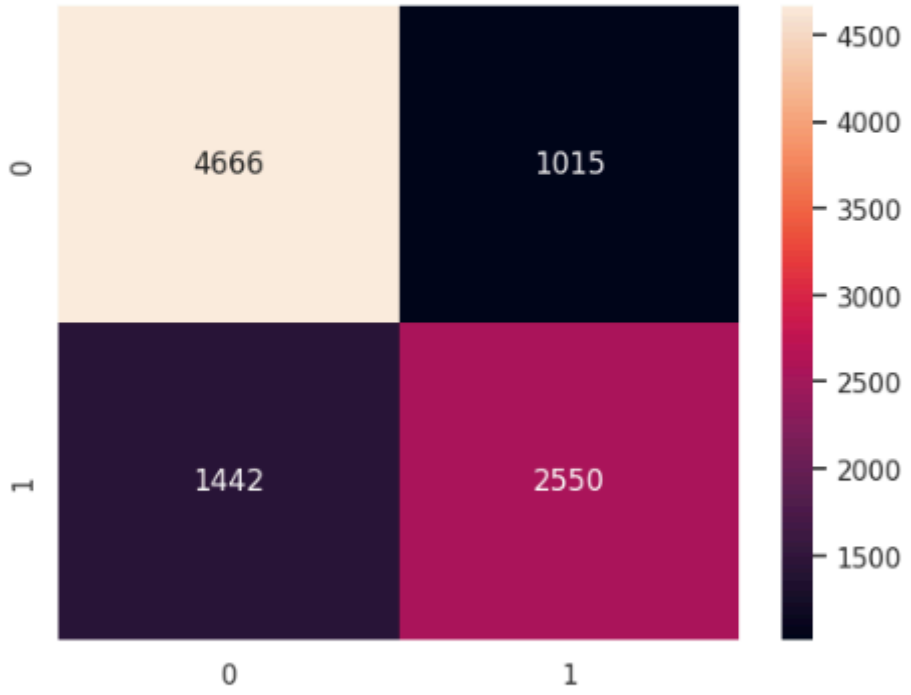
Figure 17: Results of GNB model.

```
y_predict = classi.predict(X_test)
cm = confusion_matrix(y_test, y_predict)
sns.heatmap(cm, annot=True, fmt='g')
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           1       0.76      0.82      0.79      5681
           2       0.72      0.64      0.67      3992

    accuracy                           0.75      9673
   macro avg       0.74      0.73      0.73      9673
weighted avg       0.74      0.75      0.74      9673
```



1.Test_size = 40% Accuracy = 0.75, F1 score = 0.79 and 0.67
2.Test_size = 30% Accuracy = 0.75, F1 score = 0.79 and 0.67
3.Test_size = 20% Accuracy = 0.75, F1 score = 0.79 and 0.67
4.Test_size = 10% Accuracy = 0.75, F1 score = 0.79 and0.67

Figure 18: Classification report of GNB mod

After testing our models, we noticed that we required to perform further data preparation to get better results, therefore we proceed to implement the following:

- Remove an additional feature that was not adding any value.
- Scale our data since it was slightly skewed.
- Reduce dimensionality using PCA.
- Balance our data due that our machine learning models were bias to patients that survived.

Random Forest 4

```
[ ]  # SMOTE (to balance the data)

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state = 11)
     smt = SMOTE()

     X_train, y_train = smt.fit_resample(X_train, y_train)
     X_test, y_test = smt.fit_resample(X_test, y_test)
```

```
[ ]  X_train.shape, X_test.shape,y_train.shape,y_test.shape
```

```
((44884, 16), (11362, 16), (44884,), (11362,))
```

```
rfc= RandomForestClassifier(n_estimators= 50, max_depth=3)

#Train the model using the training sets y_pred=clf.predict(X_test)
rfc.fit(X_train, y_train)

y_pred = rfc.predict(X_test)
```

Model Evaluation

```
[ ]  # Model Accuracy, how often is the classifier correct?
     print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
     print("Precision:",metrics.precision_score(y_test, y_pred))

     # Rounded upto 2 decimal places
     print( "Accuracy: {:.2f}".format(metrics.accuracy_score(y_test, y_pred)))
     print( "Precision: {:.2f}".format(metrics.precision_score(y_test, y_pred)))
```

```
Accuracy: 0.7569970075690899
Precision: 0.7411628675256029
Accuracy: 0.76
Precision: 0.74
```

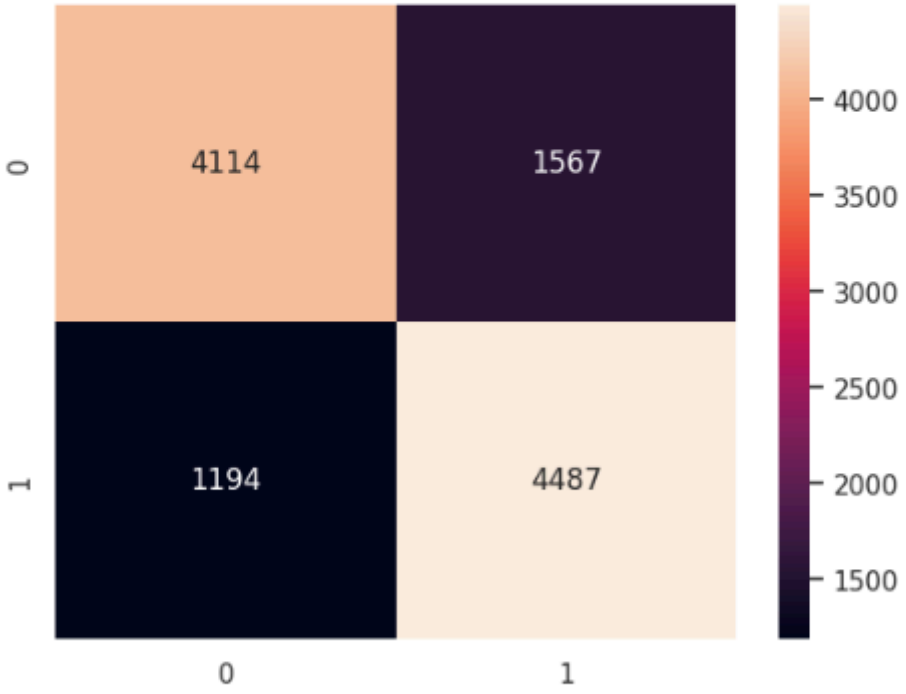
Figure 19: RF model after further data preparation.

```
y_predict = rfc.predict(X_test)
cm = confusion_matrix(y_test, y_predict)
sns.heatmap(cm, annot=True, fmt='g')
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.78      0.72      0.75      5681
           1       0.74      0.79      0.76      5681

    accuracy                           0.76     11362
   macro avg       0.76      0.76      0.76     11362
weighted avg       0.76      0.76      0.76     11362
```




```
# Calculate the score using cross validation method
cross_val_score(rfc, X, y, scoring = 'accuracy', cv = 8).mean()
```

```
0.7581089952219091
```

Figure 20: Classification report of the RF model.

Subsequently, we got to a machine learning model that was consistently predicting if a Covid-19 patient survived or died with and accuracy between 86% - 88% with a proportion of actual positives correctly identified of 81% for patients who survived and 95% for patients who did not survive.

Team dynamics

For this assignment and as a team, we agreed to work remotely using Whatsapp as our communication tool. Throughout these weeks, we shared ideas, information and we have been in constant communication regarding our assignment, whilst completely understanding each other's availability from our everyday lives, thus, we have balanced the workload of the project, nevertheless, we were involved in every phase.

Furthermore, we concurred to finish our team task first since we both understand the importance of our time, we believe that carrying out our assignment with this methodology allowed us to have a better time management.

Learning journey

From the beginning of this assignment, we used our classes notes for guidance which helped us in the process of each phase of the project; we constantly realized that we required to implement new techniques, for instance, hyperparameter optimization to our machine learning models.

We know that the knowledge gained during this assignment will be of great support throughout the two semesters of our Higher Diploma.

Conclusions

Overall, as a team we think that our topic of Covid-19 led us to revise and reinforce everything that we have learned at CCT College, owing to the fact that we applied multiple techniques while performing our analysis to accomplish the objectives of this assignment, while been in fact, an interesting and meaningful topic from our point of view due to the impact that had in the world.

In my personal experience, working with my teammate is helping me to continue developing my soft skills such as, time management, problem solving, listening, critical thinking, communication, and teamwork; this assignment definitely took us beyond what we have learned in classes and pushed us into new topics in regard to Python coding.

References:

Galvão, M.H.R., Roncalli, A.G., Galvão, M.H.R. and Roncalli, A.G. (2020). Fatores associados a maior risco de ocorrência de óbito por COVID-19: análise de sobrevivência com base em casos confirmados. Revista Brasileira de Epidemiologia, [online] 23. doi:https://doi.org/10.1590/1980-549720200106.

Huang, C., Wang, Y., Li, X., Ren, L., Zhao, J., Hu, Y., Zhang, L., Fan, G., Xu, J., Gu, X., Cheng, Z., Yu, T., Xia, J., Wei, Y., Wu, W., Xie, X., Yin, W., Li, H., Liu, M. and Xiao, Y. (2020). Clinical features of patients infected with 2019 novel coronavirus in Wuhan, China. The Lancet, [online] 395(10223), pp.497–506. doi:https://doi.org/10.1016/s0140-6736(20)30183-5.

Jamal, M., Shah, M., Almarzooqi, S.H., Aber, H., Khawaja, S., El Abed, R., Alkhatib, Z. and Samaranayake, L.P. (2020). Overview of transnational recommendations for COVID-19 transmission control in dental care settings. Oral Diseases. doi:https://doi.org/10.1111/odi.13431.

Justino Fernández (2020). Catálogo de las exposiciones de arte en 1951. Anales del Instituto de Investigaciones Estéticas. doi:https://doi.org/10.22201/iie.18703062e.1952.sup1.2454.

kaggle.com. (n.d.). Covid-19 Survival Prediction (EDA and MODEL). [online] Available at: https://www.kaggle.com/code/anishwarbehera/covid-19-survival-prediction-eda-and-model?cellIds=24&kernelSessionId=135588147 [Accessed 15 Nov. 2023].

Lauer, S.A., Grantz, K.H., Bi, Q., Jones, F.K., Zheng, Q., Meredith, H.R., Azman, A.S., Reich, N.G. and Lessler, J. (2020). The Incubation Period of Coronavirus Disease 2019 (COVID-19) From Publicly Reported Confirmed Cases: Estimation and Application. Annals of Internal Medicine, [online] 172(9), pp.577–582. doi:https://doi.org/10.7326/M20-0504.

Richardson, S., Hirsch, J.S., Narasimhan, M., Crawford, J.M., McGinn, T., Davidson, K.W., Barnaby, D.P., Becker, L.B., Chelico, J.D., Cohen, S.L., Cookingham, J., Coppa, K., Diefenbach, M.A., Dominello, A.J., Duer-Hefele, J., Falzon, L., Gitlin, J., Hajizadeh, N., Harvin, T.G. and Hirschwerk, D.A. (2020). Presenting Characteristics, Comorbidities, and Outcomes Among 5700 Patients Hospitalized With COVID-19 in the New York City Area. JAMA, 323(20). doi:https://doi.org/10.1001/jama.2020.6775.

World Health Organization (2020). Coronavirus disease (COVID-19). [online] World Health Organization. Available at: https://www.who.int/emergencies/diseases/novel-coronavirus-2019/question-and-answers-hub/q-a-detail/coronavirus-disease-covid-19 [Accessed 21 Nov. 2023].

World Health Organization (2021). The true death toll of COVID-19: estimating global excess mortality. [online] World Health Organization. Available at: https://www.who.int/data/stories/the-true-death-toll-of-covid-19-estimating-global-excess-mortality.