The Role of Pre-Existing Conditions in COVID-19 Outcomes

Odalys Jordan

Nov 30, 2024

Contents

Abstract

The COVID-19 pandemic has had a profound impact on humanity, causing chaos and uncertainty as the virus spread rapidly across the globe, claiming countless lives. At the time this project began back in 2020, little was known about COVID-19, and there was limited publicly available data, especially from the United States. This project was inspired by the critical need to rely on data to guide decisions during times of crisis, providing insights that could save lives and support efforts like vaccine development.

This project demonstrates how data science can play a transformative role in addressing urgent challenges. Using a COVID-19 dataset with information on pre-existing health conditions, the project aimed to evaluate their impact on patient outcomes, identify the most common conditions, and assess their influence on severe illness and death. The findings reveal that individuals with pre-existing conditions are at significantly higher risk, with a death rate ten times higher than healthy individuals. While machine learning techniques were employed to predict ICU admissions, the model's performance was limited due to the dataset's characteristics. However, these results highlight the potential of data-driven approaches, suggesting that with further refinement and data optimization, predicting ICU requirements based on pre-existing conditions could become feasible, offering valuable insights to raise public awareness and assist healthcare professionals in making life-saving decisions.

Introduction

Motivation

In the midst of the 2020 COVID-19 pandemic, the world grappled with uncertainty and a dearth of reliable information. As the crisis unfolded, I was motivated to use data science to gain a deeper understanding of the virus and its impact. This project, initiated during those early days, reflects my personal journey of learning and applying data science techniques to address a pressing global issue.

The pandemic highlighted the critical role of data-driven insights in navigating health crises. By analyzing relevant datasets, we can uncover patterns, identify trends, and make informed decisions. This project aims to contribute to the broader understanding of COVID-19 and its potential implications, ultimately supporting healthcare professionals and policymakers in their efforts to combat the virus.

Relevance of the study

The results of this study hold significance for multiple audiences:

Individuals and Families: Identifying personal risk based on medical history and existing health conditions, enables individuals to take precautionary measures and protect themselves and those around them.

Healthcare Professionals: Insights into which patients are more likely to require ICU care can assist hospitals in resource planning and inform clinical decision-making.

Public Awareness Efforts: Clear communication of the most common risk factors can help raise public awareness among communities and encourage protective behaviors, particularly for vulnerable populations.

Project goal

The goal of this project is to evaluate the role of pre-existing conditions in determining COVID-19 outcomes and to identify the most prevalent conditions associated with severe illness and death.

Research questions

- Are individuals with pre-existing conditions more likely to die from COVID-19 compared to those without such conditions?
- Are COVID-19 patients with pre-existing conditions more vulnerable to severe illness requiring hospitalization and intensive care?
- What are the most common pre-existing conditions among COVID-19 patients?
- Can ICU admission requirements be predicted based on pre-existing health conditions in COVID-19 patients?

Acknowledgments

This project would not have been possible without the guidance and resources provided by Dr. Rafael Irizarry, whose expertise and dedication to education have been instrumental. Special thanks also go to the edX platform for making this professional program accessible to students around the world, fostering the development of critical skills needed to address real-world challenges.

About the dataset

Name and size
COVID-19 Patient pre-conditions (44.52 MB).

Source
Kaggle https://www.kaggle.com/
https://www.kaggle.com/tanmoyx/covid19-patient-precondition-dataset

Description
The data was obtained from the "COVID-19 related cases dataset", released by the Mexican government. The original dataset is public, and can be downloaded from the page of the General Direction of Epidemiology, belonging to the Health Secretary of the Mexican government. https://www.gob.mx/salud/documentos/datos-abiertos-152127

The data pertain to Mexican individuals. No other population is included. It contains a large number of anonymised patient information, including data about 12 medical conditions. The dataset has a total of 566,602 observations (rows) and 23 features (columns) to describe each observation. The dataset has a total of 566602 observations(rows) and 23 features(columns) to describe each observation.

Features description

Id: Patient id. Auto generated Random number.
Age: Patient age in years.
~~**Sex**: Patient Sex. Values: 1 - Male, 2 - Female, 99 - not specified~~
Sex: Patient Sex. Values: 1 - Female, 2 - Male, 99 - not specified *(correction made due to data inconsistency)*
Covid Res: Covid-19 Test Result. Values: 1 - Positive SARS-CoV-2, 2 - Negative SARS-CoV-2, 3 - Pending Result
Patient Type: Values: 1 - ambulatory, 2 - hospitalized, 99 - not specified
Date Died: Patient's death date. Values: date, other values: 9999-99-99
Intubed: Identifies if the patient required intubation. Values: 1 - Yes, 2 - No, other values: 97, 98, 99
ICU: Patient required ICU admission. Values: 1 - Yes, 2 - No, other values: 97, 98, 99
contact_other_covid: Identify if the patient had contact with any other case diagnosed with Covid-19. Values: 1 - Yes, 2 - No, other values: 97, 98, 99
Patient health conditions
Identifies if the patient was diagnosed with the Condition.

Values: 1 - Yes, 2 - No, other values: 97, 98, 99
List of conditions: pneumonia, pregnancy, diabetes, copd, asthma, immunosuppression, hypertension, cardiovascular, obesity, renal_chronic, tobacco, other_disease.

Other Values, utilized to represent missing data

Value - Meaning
97 - NOT APPLICABLE
98 - IGNORED
99 - NOT SPECIFIED

Assumptions and considerations

- "COVID-19 patient" = "confirmed case of COVID-19".
- Date Died Since this dataset doesn't provide any details on how to interpret the date of death with value of 9999-99-99, and considering the 99 value has the meaning of "not specified" in the rest of the features, it was assumed that, event though the value 9999-99-99 doesn't necessarily means the patient didn't died, the patient's death is not confirmed. Therefore, rows with Date Died equals 9999-99-99 won't be considered as confirmed deaths.
- Keep in mind that the data in this dataset pertain to individuals in Mexico.
- For this analysis, only confirmed cases of COVID-19 will be considered. Therefore, the dataset will be filter to drop those observations of patients that have not been confirmed to have the virus.

Key Steps from Data Cleaning to Model Evaluation

This project followed a structured data science process, encompassing data cleaning, exploratory data analysis, modeling, and evaluation. Each step was crucial in addressing the research questions and achieving the project's objectives.

- **Data Cleaning:** This phase focused on preparing the dataset for analysis. Once the data was loaded, irrelevant features were removed, missing values were handled, and categorical variables were transformed into meaningful labels. Boolean variables were standardized, and new features were derived to better capture the relationships between pre-existing conditions and COVID-19 outcomes. Special attention was given to removing data inconsistencies, ensuring the dataset was suitable for analysis and modeling.
- **Exploratory Data Analysis:** Following data cleaning, exploratory analysis was performed to understand the dataset's characteristics. This included summarizing key variables and investigating the distribution of pre-existing conditions among COVID-19 patients. Data visualization techniques, such as pie charts, bar charts, and histograms, were used to illustrate the relationships between pre-existing conditions, ICU admissions, and death rates. These visualizations highlighted important trends, such as the higher risk of severe illness and death among patients with pre-existing conditions.
- **Feature Engineering:** Additional features were created to enhance the dataset's predictive capabilities. For example, a feature indicating whether a patient had any pre-existing condition was added, and missing values for ICU admission requirements were removed to improve model quality.
- **Model Building:** The data modeling phase aimed to predict ICU admission requirements based on pre-existing conditions using a supervised machine learning approach. A decision tree model was initially applied due to its interpretability. Subsequently, a random forest model was implemented to capture more complex relationships. The Gradient Boosting Machines (GBM) model was also employed as a final attempt to enhance predictions.

- **Model Evaluation:** Finally, the models' performance was evaluated using metrics such as **accuracy, precision, recall**, and **F1-score**. Confusion matrices were used to assess how well the models classified ICU requirements.

This structured process ensured that every step was designed to extract meaningful insights and address the challenges posed by the dataset's limitations while contributing to the project's objectives.

Methods and Analysis

This section provides a detailed account of the steps followed to achieve the project's objective. The process encompasses the loading of necessary libraries, dataset creation, exploratory data analysis, data visualization, as well as model development and evaluation.

In order to analyze the data and answer the questions formulated for the project, different methods were used:

- Data visualization techniques. The bar and pie plots allowed to clearly visualize the data and get to conclusions easily. Using these methods, the findings can be communicated in an elegant and efficient way.
- Data structures. R vectors, factors and data frames, were the structures utilized to perform data analysis. These structures are very convenient for data manipulation.
- Machine learning algorithms. Classification models were applied to the dataset to attempt predicting patients ICU admission requirements based on underlying conditions. These algorithms are very appropriate since this is a binary classification problem, where the prediction is made between one of the two categories: patient required ICU or did not require ICU. A supervised machine learning approach was followed, because the target variable *icu* to be predicted by the model was known (provided).

– The popular Decision Tree algorithm will be used for building a first classification model. – Then, a Random Forests model was implemented to capture more complex relationships. – The Gradient Boosting Machines (GBM) model was also employed, in a final attempt to enhance predictions.

Load Necessary Libraries

To implement the project, a variety of libraries were loaded to support tasks such as data manipulation, visualization, and modeling. The **tidyverse** package was used for data manipulation, while **ggplot2** facilitated data visualization. Libraries such as **caret** and **Metrics** were employed to support model building and evaluation. Other libraries like **data.table** and **randomForest** were used for efficient data handling and modeling.

```
if (!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if (!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if (!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if (!require(data.table)) install.packages("recipes", repos = "http://cran.us.r-project.org")
if (!require(data.table)) install.packages("themis", repos = "http://cran.us.r-project.org")

library(dplyr)
library(tidyverse)
library(data.table)
library(ggplot2)
library(caret)
library(rpart)
library(randomForest)
```

Data loading and dataset creation

First, the *covid19 patient pre-conditions* dataset was downloaded from the source: https://www.kaggle.com/api/v1/datasets/download/tanmoyx/covid19-patient-precondition-dataset, and its content was extracted

into a new folder named *"covid19-patients-preconditions-dataset* in the project's root directory. For convenience, the dataset files are provided along with the project files.

The actual patient data in this dataset is in a csv file named **covid.csv**. The data in this file will be loaded into a data frame for the analysis. In the dataset's directory there is also a file with the data dictionary and another one with the catalogs that describe the data in the dataset.
- **Description.xlsx**
- **Catalogs.xlsx**

The data frame created to hold the data was named **"covid19_preconditions"**.

Data cleaning

In this phase, In order to prepare the data for analysis, the following operations were performed on the dataset:

- Remove duplicate rows.
- Transform missing encoded data into NA for better manipulation.
- Fix Sex column definition.
- Discard irrelevant data.
- Rename columns. The patient type and icu columns to better reflect its content.
- Decode the Sex feature for easier recognition.
- Transform the data to Boolean for easier manipulation.
- Add necessary columns for the analysis. The dataset was modified to include a new boolean columns: has_preconditions, confirmed_deceased, and confirmed_icu.

Initial data exploration

First, the dataset was explored to see if there were data quality issues (noisy, missing data, inconsistencies, etc.). This initial data exploration allowed to identify the actions to take in order to clean and/or transform the raw data for analysis.

Dataset Overview

```
# Data Summary and Structure

str(covid19_preconditions) # Data Shape compactly displayed
summary(covid19_preconditions)  # Summary statistics
```

The dataset has a total of **566602 observations (rows)** and **23 features (columns)** that describe each observation.

First rows in the dataset

```
head(covid19_preconditions) # first few rows of the data frame
```

Detect data inconsistencies

The dplyr package was utilized to filter the data based on specific conditions.

- Total men in the dataset: 279490
- Total women in the dataset: 287112
- There are 4063 observations of men with pregnancy = 1 (Yes)

Analyze pregnancy values in women
The data is filtered to keep only rows with sex equals 2 (female).
Count the number of women with unknown pregnancy values (i.e., pregnancy is NA).

- Total women with unknown pregnancy: 287112 .
- Total women with pregnancy equals NOT APPLICABLE: 287112 .

Conclusion: All the records from women in the dataset have pregnancy equals NOT APPLICABLE.

Detect observations with patients not hospitalized (ambulatory) and admitted to ICU

- There are no observations of ambulatory patients admitted to an ICU.

Conclusion: There are no inconsistencies in the data related to patient type and hospitalizations.

Detect observations of patients that tested negative for COVID-19 and were hospitalized

The data is filtered by patient type equals 2 (hospitalized) and covid-19 test result equals 2 (Negative)

- There are observations corresponding to non-COVID-19 patients that were hospitalized.

The dataset contains data from patients that were hospitalized for reasons other than a COVID-19 diagnosis. This is not a data inconsistency, but it is important to take it into account in the data analysis, in case these observations are not relevant and need to be excluded.

Problems encountered in the dataset

Missing values encoded
Per the results presented during the data exploration, it seems like the dataset doesn't have any explicit missing value. However, by looking into the data dictionary and catalogs files, it can be determined that the missing data has been encoded with specific values:
97 - NOT APPLICABLE , 98 - IGNORED, 99 - NOT SPECIFIED
This means other operations like replace are needed in order to better identify unwanted missing data, and remove it as needed.

The "Sex" column definition was swapped
The documentation for this dataset states that sex = 1 is male, and sex = 2 is female. The data, on the other hand, indicates that: - No women are pregnant. All pregnancy values for women in this dataset are unknown (represent missing data) - There are pregnant men. - All women pregnancy values are NA.

All these facts indicate that there is an error in the data. It looks like either the "pregnancy" or the "sex" feature was flipped, causing these inconsistencies. After downloading the original descriptors for the dataset from the Mexican government web page, it was confirmed that sex should be switched: sex = 1 is female, and sex = 2 is male. This was a mistake that probably occurred at kaggle, while translating the dataset description from Spanish to English.

Action taken: The definition for the sex column was corrected in the "Features description" section of this report.

Data preparation

Handle missing values and duplicates

The dataset used values 97, 98, and 99 to represent missing data. These values were replaced with *NA*. The duplicated observations were removed from the dataset.

Discard irrelevant data

Since the purpose of this project is to analyze the pre-existing conditions in confirmed cases of COVID-19, the information that is no directly related to those features will be discarded from the dataset.

The observations that are relevant for the analysis are those of confirmed Covid-19 cases. Thus, all the other observations can be drooped.

```
# Filter the observations where covid_res is equals 1 (Positive. Confirmed cases) and then remove the c
covid19_confirmed_patients <- covid19_preconditions |>
  filter(covid_res == 1) |> # keep only confirmed Covid-19 cases
  select(-covid_res)  # Remove covid_res column corresponding to "Covid-19 Test Result" as it is no lon
```

After these operations, the dataset now has a total of **220657 observations (rows) and 22 features (columns)**.

Table 1: Dataset Dimensions

Metric	Value
Rows	220657
Columns	22

Feature Engineering

Apply transformations and create new features using mutate and logical conditions with dplyr.

Select relevant columns and discard irrelevant data

Define which features are relevant for the analysis.

```
# Create vectors of relevant and irrelevant features
irrelevant_features <- c("id", "entry_date", "date_symptoms", "contact_other_covid", "intubed")

relevant_features <- c("age", "sex", "patient_type", "icu", "date_died",
                       "pneumonia", "pregnancy", "diabetes", "copd", "asthma", "inmsupr",
                       "hypertension", "cardiovascular", "obesity", "renal_chronic",
                       "tobacco", "other_disease")
```

Exclude the columns that aren't directly related to the analysis (irrelevant_features), and keep only those in relevant_features.

The dataset now has a total of 17 features (columns).

Rename columns

Rename the *patient type* and *icu* columns to better reflect its content.

```
# Rename the patient_type column to inpatient and the icu column to required_icu
covid19_confirmed_patients <- covid19_confirmed_patients |>
  rename(inpatient = patient_type, required_icu = icu)
```

Decode and transform columns to boolean

Convert the *sex, inpatient, and required_icu* columns to make them more interpretable, and transform the *conditions_features* to logical (boolean) values for easier manipulation during the analysis. Verify the data to make sure it makes sense now:

- Total women: 99858
- Total women with unknown pregnancy: 684
- Total women with pregnancy = NOT APPLICABLE: 0

Add new columns to the data frame for further analysis

- **has_preconditions (boolean):** indicates if the patient has pre-existing conditions. This is a representation of whether or not the patient has risk factors.
- **confirmed_deceased (boolean):** indicates whether the patient's death is confirmed with a valid date, or it's unknown.
- **confirmed_icu (boolean):** indicates if the patient was admitted to an ICU or not.

Resulting dataset

Table 2: Dataset Dimensions

Metric	Value
Rows	220657
Columns	20

Data analisis and visualization

This section covers exploring and visualizing the most relevant aspects of this dataset (e.g., health conditions and distributions).
The calculations needed to quantify the variables needed for the analisis, will be performed on the dataset. This will be complemented with data visualization techniques to aid gaining insights on the data.These techniques will address the first three research questions formulated for this project, by comparing death rates, hospitalization rates, and the prevalence of underlying conditions.

These are the research questions, related to deaths and hospitalizations rates:
- Are individuals with underlying health conditions more likely to die from COVID-19 compared to healthy people?
- Are COVID-19 patients with underlying health conditions more vulnerable to becoming severely ill with the virus, requiring hospitalization and intensive care?
- What are the most common underlying conditions in COVID-19 patients?

Data visualizations:

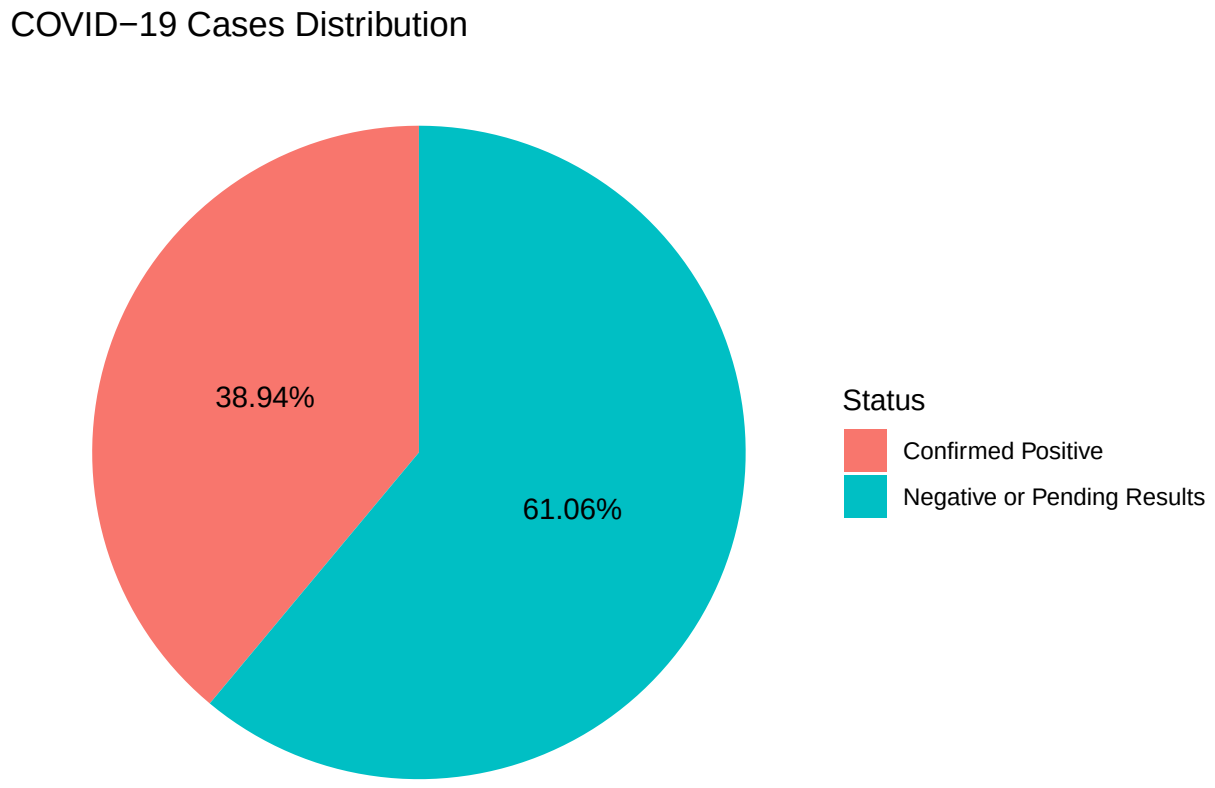
Pie Charts for overall case distribution and health condition distribution.
Bar Charts for hospitalization and death rates, categorized by underlying conditions.
Bar Chart showing the most common pre-existing health conditions.
Bar Chart of Common Underlying Conditions

Analyze COVID-19 patients case distribution. Pie Chart.

- Total observations in the dataset: 566602

- Total observations of confirmed cases of Covid-19 in the dataset: 220657
- Total women confirmed cases of Covid-19: 99858- Total men confirmed cases of Covid-19: 120799- Total observations of confirmed COVID-19 cases in the dataset: 220657 (38.94 %)

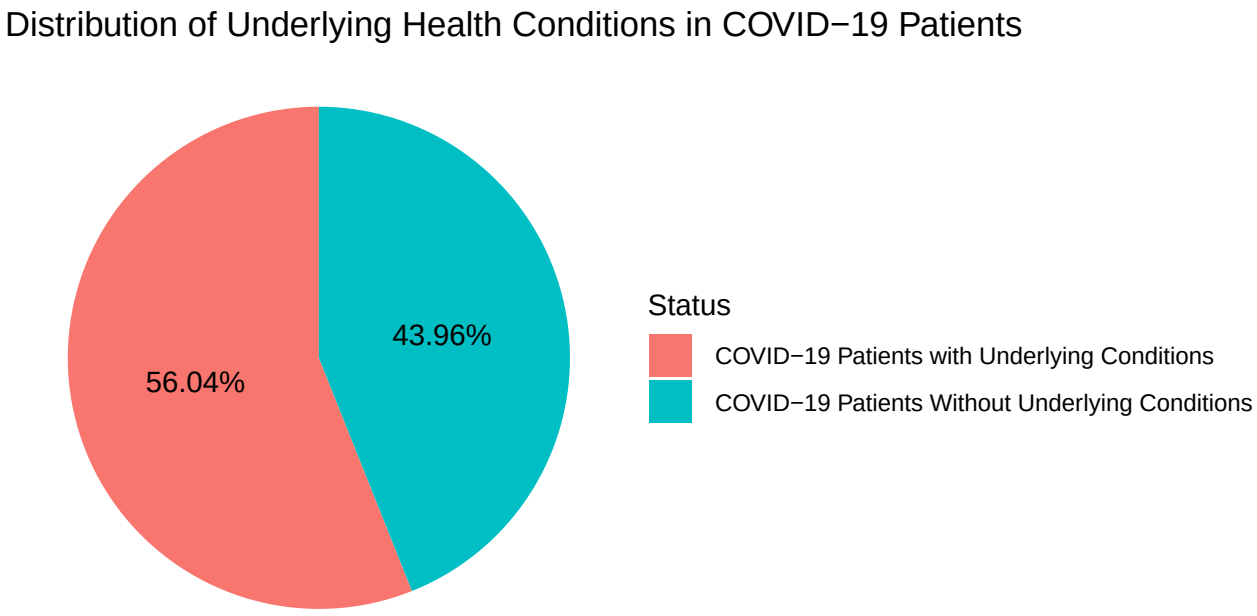
COVID-19 Cases Distribution Pie Chart



Analyze patients with and without Health Conditions. Pie Chart.

Total observations of confirmed COVID-19 cases with health conditions: 123665 (56.04 %) Total observations of confirmed COVID-19 cases without health conditions: 96992 (43.96 %)

COVID-19 Cases Health Conditions Distribution Pie Chart

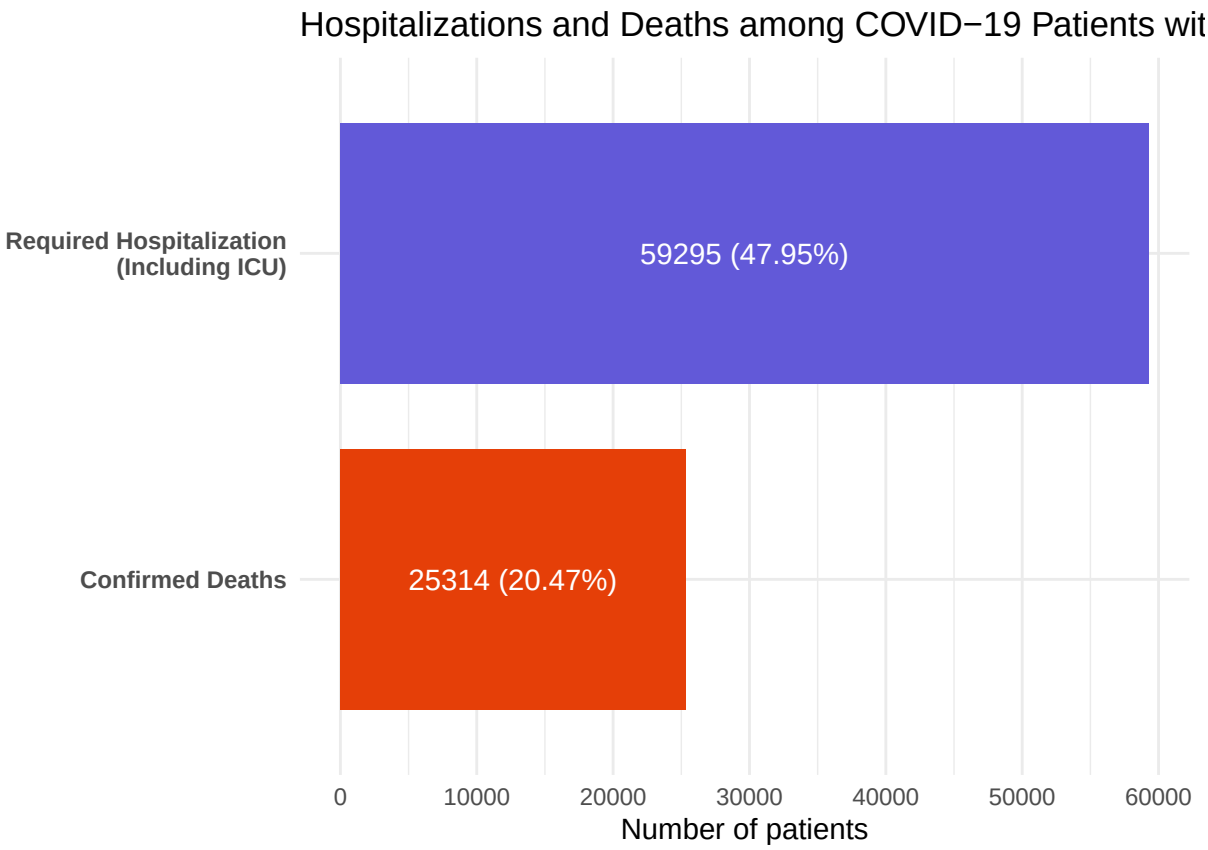


Analyze patients hospitalization and death rates

To answer the questions about death and hospitalization rates among those with and without underlying conditions, bar charts were created to compare these rates.

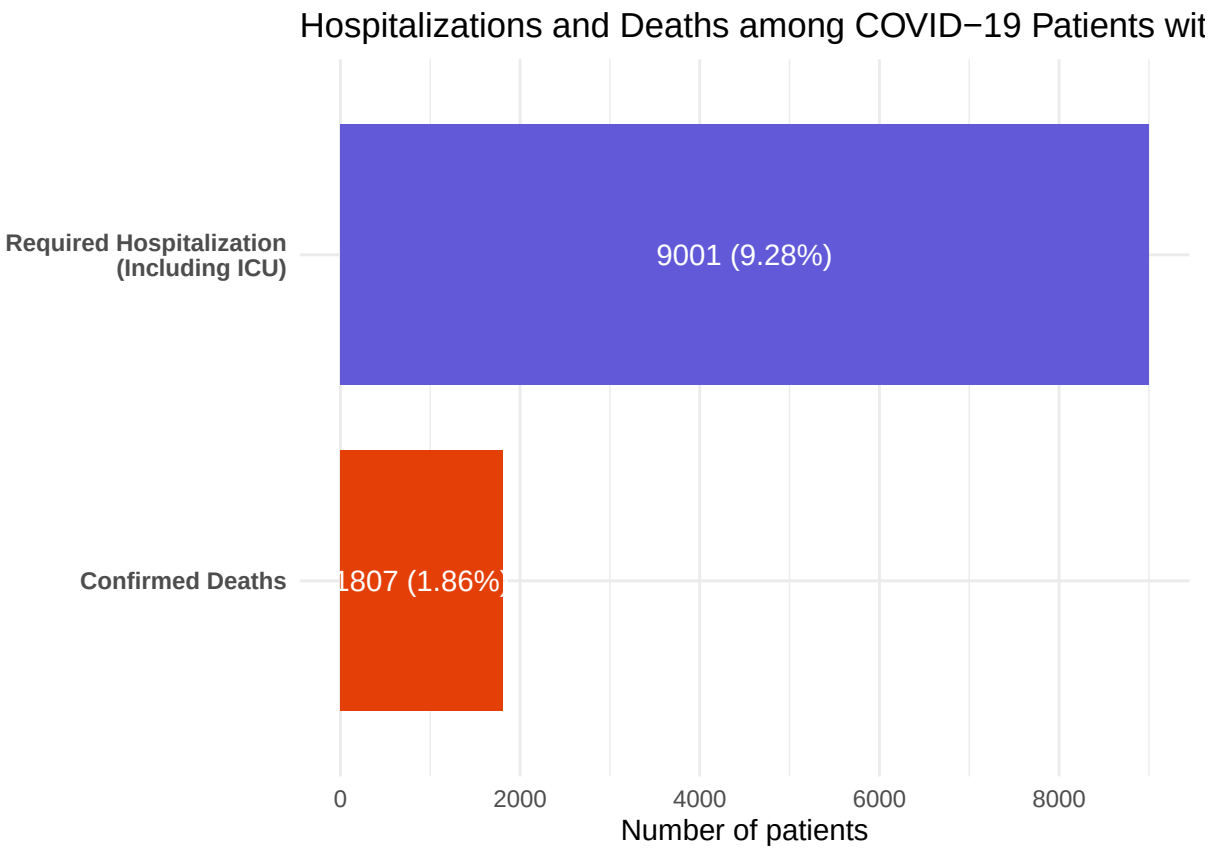
A: Hospitalizations and deaths in COVID-19 patients with underlying health conditions

- Total observations of confirmed COVID-19 cases with health conditions that required hospitalization (Including ICU): 59295 (47.95 %)
- Total observations of confirmed COVID-19 cases with health conditions that died: 25314 (20.47 %)



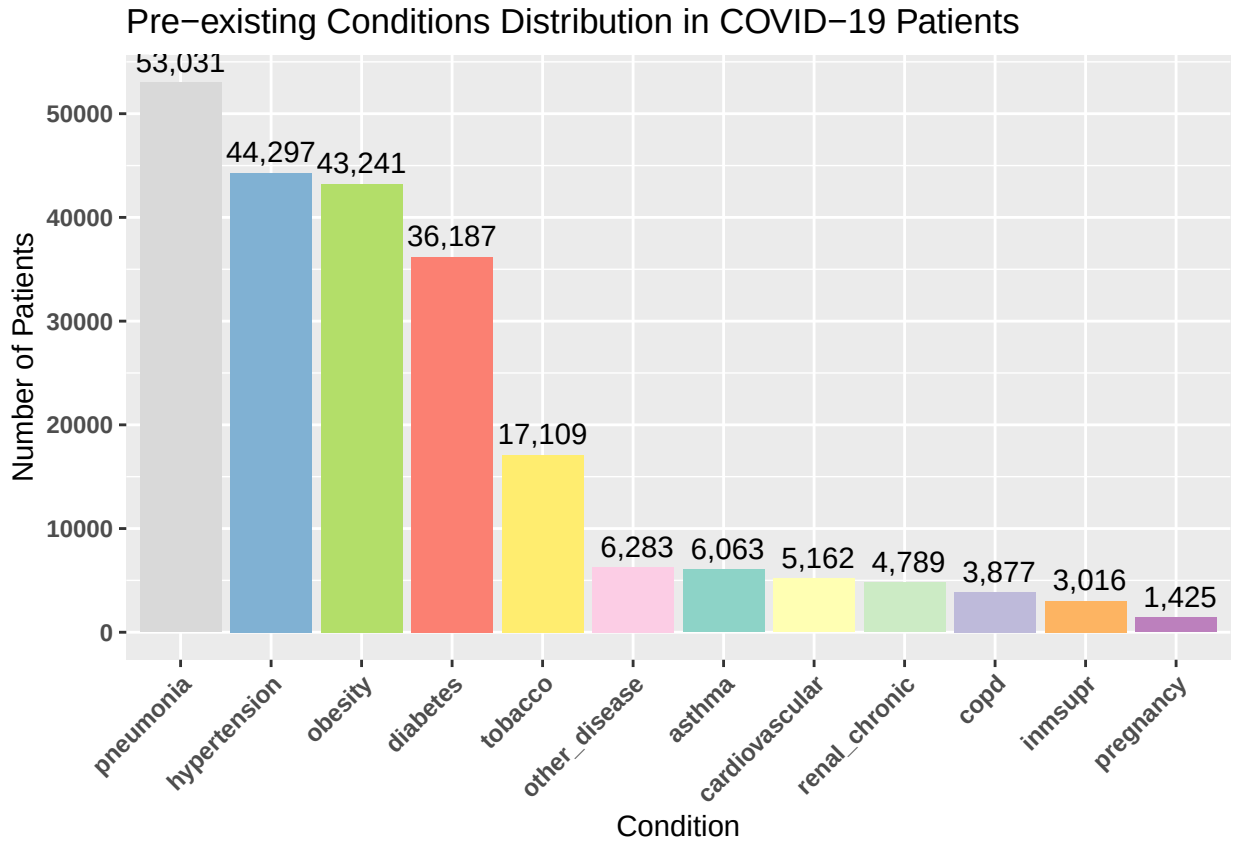
B: Hospitalizations and Deaths among COVID-19 patients without conditions

- Total observations of confirmed COVID-19 cases with health conditions that required hospitalization (Including ICU): 9001 (9.28 %)
- Total observations of confirmed COVID-19 cases with health conditions that died: 1807 (1.86 %)



Discovery of the most common underlying conditions

This final chart identifies the most common health conditions among confirmed COVID-19 patients, answering the third question.



Insights gained:

Per the graphic above the most common underlying conditions, reported by more than 30,000 COVID-19 patients in Mexico, were pneumonia, hypertension, obesity and diabetes. The next most common is tobacco, reported by 17,109 patients. Less than 10,000 patients reported other pre-existing conditions.

The U.S. Centers for Disease Control and Prevention (CDC) June 2020 COVID-19 report, states that the most common underlying conditions reported in COVID-19 patients in the United States were heart disease, diabetes and chronic lung disease.

Although it is logical that the most common conditions vary by country, it can be seen that there are some such as diabetes and heart disease, which are a common denominator between countries.

Data Modeling

In order to check the factibility of predicting if COVID-19 patients are going to require admission to an intensive care unit (ICU), based on his underlying medical conditions, the Classification machine learning technique was applied to the dataset.

The input data was the underlying medical conditions features in this dataset. Each of this features specifies if the patient reported to have that medical conditions or not.

The target variable in this modeling problem is "confirmed_icu". This feature's value represents the ICU requirement for a patient, which is what we want to be able to predict. This feature was provided as part of the "COVID-19 patient conditions" dataset.

Steps in the modeling phase:

1 Data cleaning and preparation
- Select relevant classification features.
- Remove rows with missing values in the target variable "confirmed_icu".

2 Train-Test Split
- The data will be divided into 70% training and 30% testing subsets.

3 Model building
- Build a model for binary classification on ICU admission.
- Evaluate model performance on a test set.

Models created:
- Decision Tree Model
- Random Forests Model
- Gradient Boosting Machines (GBM) Model

4 Model Evaluation
- Implement "accuracy" calculation and evaluate model performance.
- Select other metrics to evaluate the model performance.

Data cleaning and preparation

Select the relevant features for this classification problem.
Remove rows with missing values in the *confirmed_icu* feature.

```
# Define classification features and filter relevant data
classification_features <- c("confirmed_icu", "has_preconditions", "pneumonia", "pregnancy",
                             "diabetes", "copd", "asthma", "inmsupr", "hypertension",
                             "cardiovascular", "obesity", "renal_chronic", "tobacco", "other_disease")

# Select classification data per the selected features
classification_data <- covid19_confirmed_patients |>
  select(all_of(classification_features))
```

Total Observations before cleaning: 220657

Check for missing values in the resulting data frame.

```
# Check the features that have missing values
any_na <- any(is.na(classification_data))

cat("Are there any missing values remaining in the data frame?", any_na,"\n")
```

Are there any missing values remaining in the data frame? TRUE

Remove Missing Values from the data set

The goal is to drop rows with any missing values in the *classification_data* dataset and check the before-and-after row counts to ensure proper cleaning.

Total of observations from patients that were admitted to an ICU: 5822 Are there any missing values remaining in the data frame now? FALSE[1] FALSE TRUE Are there any missing values in the target variable *confirmed_icu*? FALSE

Print Row Counts

Review the number of observations before and after cleaning to understand the impact of removing missing values.

- The classification data frame has 14 features
- Total observations before cleaning: 220657
- Total observations after cleaning: 25087 After cleaning, there are a total of 1864 observations from patients that were admitted to an ICU

After these operations, the dataset now has a total of 25087 observations (rows) and 14 features (columns) .

Table 3: Dataset Dimensions

Metric	Value
Rows	25087
Columns	14

Inspect the quality of the data

The characteristics of the data affects the model's selection. Common data issues are that the features (independent variables) might have constant or irrelevant values, or the target variable is poorly balanced.

Check Data Balance

FALSE TRUE 23223 1864

Table 4: Target Variable Class Distribution

Class	Count
FALSE	23223
TRUE	1864

Insights gained:

The target variable *confirmed_icu* is heavily imbalanced, with the majority of cases being FALSE (FALSE: 23223, TRUE: 1864). This class imbalance may hinder model performance. To fix this, balancing techniques like oversampling or undersampling can be applied to the dataset.

Feature Engineering

Perform further feature selection based on the quality of the data, and the correlation with the target variable *confirmed_icu*.

Inspect Features to check for low variance or irrelevant features in the data.

```r
summary(classification_data)
```

Evaluation of the features and recommendations for discarding irrelevant or low-variance features.

Features considered for keeping:

has_preconditions: High variance (FALSE: 3192, TRUE: 21895), likely informative. Retain.
pneumonia: Reasonable variance (FALSE: 8915, TRUE: 16172), likely informative. Retain.
diabetes: Moderate variance (FALSE: 16482, TRUE: 8605), likely informative. Retain.
hypertension: Reasonable variance (FALSE: 15167, TRUE: 9920), likely informative. Retain.
obesity: Reasonable variance (FALSE: 18332, TRUE: 6755), likely informative. Retain.

Features to potentially discard:

All the features below have very low variance and low predictive value, and therefore will contribute very little to distinguishing between TRUE and FALSE for confirmed_icu, as their distributions are highly skewed toward FALSE.

pregnancy (FALSE: 24661, TRUE: 426): likely irrelevant feature unless we are specifically analyzing pregnancy-related outcomes, which is not the case.
copd (FALSE: 23997, TRUE: 1090)
asthma (FALSE: 24251, TRUE: 836)
inmsupr (FALSE: 24297, TRUE: 790)
cardiovascular (FALSE: 23953, TRUE: 1134)
renal_chronic (FALSE: 23808, TRUE: 1279)
tobacco (FALSE: 24066, TRUE: 1021)
other_disease (FALSE: 23735, TRUE: 1352)

Determine feature importance using correlation

The correlation values indicate the strength of the linear relationship between each feature and the target variable *confirmed_icu*.
To verify feature importance using correlation with the target variable confirmed_icu, the point-biserial correlation between the target (binary) and each feature (binary or logical) was calculated.

In the cor() function that calculates correlation, the parameter *use* specifies how missing values are handled when calculating the correlation. In this case, to make sure the correlation is calculated on valid (non-missing) data points, the value selected for the *use* parameter was "complete.obs". This setting tells the function to compute the correlation using only the rows where the subject variables have non-missing values (i.e., no NAs in the paired observations). Any rows with NA in either variable are excluded from the calculation.

```r
# Convert logical variables to numeric
classification_data_numeric <- classification_data |>
  mutate(across(everything(), as.numeric))

# Calculate correlations with the target variable
correlations <- sapply(classification_data_numeric[, -1],
                       function(x)
                         cor(x, classification_data_numeric$confirmed_icu, use = "complete.obs")
                       )

# Sort correlations by absolute value
sorted_correlations <- sort(correlations, decreasing = TRUE)

# Print sorted correlations
print(sorted_correlations)
```

```
    pneumonia has_preconditions           obesity          diabetes
 1.258794e-01      6.484421e-02      3.463966e-02      1.845362e-02
      inmsupr    cardiovascular      hypertension         pregnancy
 1.244714e-02      1.078554e-02      9.925610e-03      7.467330e-03
other_disease              copd     renal_chronic            asthma
 1.712696e-03      7.540760e-04     -2.178701e-05     -9.822972e-05
      tobacco
-1.527811e-02
```

Insights gained:

The correlation values indicate the strength of the linear relationship between each feature and confirmed_icu:

- Values close to 1 or -1: Strong correlation (high importance).
- Values close to 0: Weak correlation (low importance, potentially irrelevant).

Interpreting the Correlation Values

Positive Values: A positive correlation indicates that as the feature increases, the likelihood of requiring ICU also increases.
Negative Values: A negative correlation suggests the opposite relationship.
Absolute Magnitude: The larger the absolute value, the stronger the linear relationship. Typically:
- > 0.1: Strong enough to retain as a meaningful predictor.
- 0.05 - 0.1: Moderately important; consider retaining based on context.
- < 0.05: Weak or negligible correlation; consider discarding unless domain knowledge suggests otherwise.

Feature Ranking

Highly Relevant Features:

pneumonia (0.125): Strongest correlation. Retain. **has_preconditions (0.065):** Moderately correlated. Retain.

Moderately Relevant Features:

obesity (0.035): Weak correlation, but retain as it might interact with other conditions. diabetes (0.018): Weak but potentially important in a medical context. Retain.

Weakly Relevant Features (considered for discarding):

inmsupr (0.012): Minimal correlation; discard unless domain knowledge suggests relevance. **cardiovascular (0.011):** Similar to inmsupr, likely discard. **hypertension (0.010):** Weak, but retain due to its prevalence in ICU patients.

Least Relevant Features (proposed for discarding):

pregnancy (0.007): Negligible correlation, likely discard. **other_disease (0.001):** Very weak, discard. **copd (0.0007):** Very weak, discard. **renal_chronic (-0.00002):** No meaningful relationship, discard. **asthma (-0.0001):** No meaningful relationship, discard. **tobacco (-0.015):** Weak negative correlation, discard.

Based on the results above, these are the relevant and informative features, that will be retained to be used to build the models:

- pneumonia

- has_preconditions
- obesity
- diabetes
- hypertension

```
# Update the dataset to remove the less relevant features.
classification_data <- classification_data |>
  select(confirmed_icu, pneumonia, has_preconditions, obesity, diabetes, hypertension)
```

Conclusion:

It was confirmed that the data is imbalanced. In this scenario, the decision tree model more likely won't work because it will struggle splitting the data to improve the classification. Therefore, other models that handle imbalanced will be utilized. The Random Forests model is a well known machine learning approach that automatically balances decision trees across multiple subsets. This model aims to improve prediction performance by averaging multiple decision trees, created as a forest constructed with randomness.

Now, with clean and relevant data in the data frame, the next step is to build the classification models.

Train-Test data split

Split the data into training and testing sets. The data will be divided into 70% training and 30% testing subsets.

```
# Split data into training and test sets (70% train, 30% test)
set.seed(324) # To ensure reproducibility
train_index <- createDataPartition(classification_data$confirmed_icu, p = 0.7, list = FALSE)
train_data <- classification_data[train_index, ]
test_data <- classification_data[-train_index, ]

# Check the class distribution before oversampling
table(train_data$confirmed_icu)
```

FALSE TRUE 16257 1305

```
# Convert the table to a data frame for better readability
class_distribution_df <- as.data.frame(table(train_data$confirmed_icu))

kable(class_distribution_df, col.names = c("Class", "Count"), caption = "Target Variable Class Distribu
```

Table 5: Target Variable Class Distribution

Class	Count
FALSE	16257
TRUE	1305

Balance the training data

It was confirmed before that the target Variable *confirmed_icu* is highly imbalanced, with FALSE: 23223(majority class) and TRUE: 1864(minority class). To avoid poor model performance, and improve its ability to learn from the minority class the training data will be balanced using oversampling. The ROSE package will be used for this task.

Package: ROSE
Type: Package
Title: Random Over-Sampling Examples
Version: 0.0-4
Date: 2021-06-14
Author: Nicola Lunardon, Giovanna Menardi, Nicola Torelli
Description: Functions to deal with binary classification
problems in the presence of imbalanced classes. Synthetic balanced samples are
generated according to ROSE (Menardi and Torelli, 2013).
Functions that implement more traditional remedies to the class imbalance
are also provided, as well as different metrics to evaluate a learner accuracy.
These are estimated by holdout, bootstrap or cross-validation methods.
License: GPL-2
Packaged: 2021-06-14 07:29:48 UTC; nicola
NeedsCompilation: no
Repository: CRAN
Date/Publication: 2021-06-14 08:10:09 UTC
Built: R 4.2.3; ; 2024-04-24 00:40:09 UTC; windows

```
# TRUE and FALSE are logical values in R, and logical values cannot be used directly as class labels in
# Renaming them to "Yes" and "No" ensures that they are treated as valid factor levels that can be used

# Convert the target variable to a factor with valid levels.
train_data$confirmed_icu <- factor(train_data$confirmed_icu,
                                   levels = c("FALSE", "TRUE"),
                                   labels = c("No", "Yes"))

# Convert the target variable in test_data to match the levels in train_data
test_data$confirmed_icu <- factor(test_data$confirmed_icu,
                                  levels = c("FALSE", "TRUE"),
                                  labels = c("No", "Yes"))

library(ROSE)

# Oversample the minority class in training data
balanced_train_data <- ovun.sample(confirmed_icu ~ ., data = train_data, method = "over", N = 2 * max(t

# Check the class distribution after oversampling
table(balanced_train_data$confirmed_icu)

# Convert the table to a data frame for better readability
class_distribution_df <- as.data.frame(table(balanced_train_data$confirmed_icu))

kable(class_distribution_df, col.names = c("Class", "Count"), caption = "Target Variable Class Distribu


# Check for missing values in the training data
sum(is.na(balanced_train_data))
summary(balanced_train_data)
```

There are no missing values in the training data.

Model building

Data Classification using Decision Trees.

Model #1: Decision Trees

```
# Use the rpart package for decision tree creation and testing.

library(rpart)

# Build the decision tree model
icu_tree <- rpart(confirmed_icu ~ ., data = balanced_train_data, method = "class",
                  control = rpart.control(minsplit = 2, cp = 0.001, maxdepth = 30))

# Check if the tree splits
 print(icu_tree)
```

n= 32514

node), split, n, loss, yval, (yprob) * denotes terminal node

1) root 32514 16257 No (0.5000000 0.5000000)

2) pneumonia< 0.5 8345 2317 No (0.7223487 0.2776513) *
3) pneumonia>=0.5 24169 10229 Yes (0.4232281 0.5767719) *

Model Overview

n = 32514: This represents the total number of observations in the dataset.
Nodes: The tree is split into three nodes, including the root and two terminal nodes (denoted by *).
yval: This indicates the predicted class at each node (No or Yes).
(yprob): This provides the probabilities of each class (No, Yes) at the node. For example, (0.5000000 0.5000000) means 50% probability for both No and Yes.

Insights

Feature Importance: The pneumonia feature was the first and only split in the decision tree. This indicates it is the most important predictor for distinguishing between patients requiring ICU care (Yes) and those who do not (No).

Class Distribution: Patients without pneumonia (pneumonia < 0.5) are more likely to not require ICU care, with a 72.2% chance of being classified as No.
Patients with pneumonia (pneumonia >= 0.5) are more likely to require ICU care, with a 57.7% chance of being classified as Yes.

Tree Simplicity: The decision tree is very shallow, with only one split on the pneumonia feature. This simplicity might suggest either a strong predictive power of the pneumonia feature, or insufficient complexity in the model to capture nuances in the data.

Once the model's performance is evaluated using accuracy, precision, recall, and F1-score, it will be confirmed if it trully has predictive power.

Model #2: Random Forests

The Random Forests algorithm improve predictive accuracy by combining the outputs of multiple decision trees. These trees are built using a process that introduces randomness in both the selection of data subsets and features, creating a "forest" of diverse trees. The final prediction is obtained by aggregating their outputs, typically through majority voting for classification or averaging for regression, which helps reduce overfitting and improve generalization.

Train the Random Forest model on the balanced data using the randomForest() function. This function implements Breiman's random forest algorithm (based on Breiman and Cutler's original Fortran code) for classification and regression.

```
# Penalize misclassification of the minority class by setting class weights:
# Assign class weights inversely proportional to their frequency
class_weights <- c(
  "No" = 1 / table(balanced_train_data$confirmed_icu)["No"],
  "Yes"  = 1 / table(balanced_train_data$confirmed_icu)["Yes"]
)

# The names in the class_weights should match the levels in the target variable confirmed_icu
names(class_weights) <- levels(balanced_train_data$confirmed_icu)
levels(balanced_train_data$confirmed_icu)
print(class_weights)

# Train Random Forest model with class weights
set.seed(324)
icu_rf <- randomForest(confirmed_icu ~ ., data = balanced_train_data, ntree = 100, mtry = 3, classwt = c
```

Model evaluation

Accuracy

The accuracy of the models will be calculated initially using the "accuracy" metric, defined as the proportion of correctly classified observations over the total number of observations. This measurement is computed as:

Accuracy = "Number of Correct Predictions" / "Total Number of Observations"

This metric is straightforward to calculate and understand, and it provides an overall measure of how well the model is performing. Here's why accuracy was chosen:

Why Accuracy?

Binary Classification Task:

This is a binary classification problem where the target variable is confirmed_icu, has two possible outcomes (TRUE for requiring ICU and FALSE for not requiring ICU). For these kind of problems, accuracy is a commonly used metric to evaluate model performance.

Balanced Dataset Assumption:

If the dataset has a relatively balanced number of cases for both classes (e.g., patients requiring ICU vs. not requiring ICU), accuracy is a good starting point to evaluate the model.

Baseline Understanding:

Accuracy provides a quick, high-level understanding of the model's performance, making it useful for initial comparisons between models (e.g., decision tree vs. random forest).

To evaluate the models, predictions are made on the test set, and then the accuracy metric is calculated.

Model #1: Decision Tree. Accuracy: 41.24 %

Model #2: Random Forest. Accuracy: 41.24 %

Preliminary Results

- The Decision Tree model provides an interpretable model for understanding key predictors of ICU admission.
- The Random Forest model didn't seemed to improve the predictive performance compared to the Decision Tree model.
- Potential Limitations of the Accuracy metric.
- Accuracy alone might not be the best metric:
  - Class Imbalance: If one class (e.g., FALSE) significantly outweighs the other (e.g., TRUE), accuracy might be misleading because the model could predict the majority class most of the time and still appear to perform well.
  - Importance of False Positives/Negatives: If the cost of false positives (predicting ICU when not needed) or false negatives (missing ICU patients) is high, additional metrics like precision, recall, or F1-score would be more informative.

Additional Metrics

Accuracy is a good starting point to evaluate the models if the dataset has a relatively balanced number of the classes subject to analysis: patients requiring ICU vs. not requiring ICU. Since this dataset is unbalanced, other metrics are needed to analyze the model's performance more thoroughly.

The following metrics will be computed for both the Decision Tree and Random Forests models: confusion matrix, precision, recall, and F1-score. These metrics provide deeper insight into the model's handling of each class, especially if the dataset is imbalanced and the cost of errors is critical. This is the case of medical applications where accurate prediction of critical outcomes is essential.

- Confusion Matrix: analyze the confusion matrix to see the distribution of true positives, false positives, true negatives, and false negatives.
- Other metrics: calculate precision, recall, and F1-score to provide a more nuanced evaluation of the models.

Confusion Matrix

Build a confusion matrix focused on predicting ICU admissions (Yes) as the positive class, which is the main objective of the analysis.

The confusion matrix summarizes the predictions as follows:

True Positive (TP):
The model correctly predicts Yes (patients who require ICU care are identified as needing ICU).

False Positive (FP):
The model incorrectly predicts Yes (patients who do not require ICU care are identified as needing ICU).

True Negative (TN):
The model correctly predicts No (patients who do not require ICU care are identified as not needing ICU).

False Negative (FN):
The model incorrectly predicts No (patients who require ICU care are missed and classified as not needing ICU).

These values are the basis for calculating other metrics.

The *caret::confusionMatrix* function is used to calculate confusion matrices for the Decision Tree and Random Forest models.

```
# Confusion Matrix for Decision Tree
 dt_conf_matrix <- confusionMatrix(data = as.factor(icu_predictions),
                                   reference = as.factor(test_data$confirmed_icu),
                                   positive = "Yes") # set interpretation and metrics to focus
                                                     # on predicting Yes (ICU requirements)

# Confusion Matrix for Random Forest
rf_conf_matrix <- confusionMatrix(data = as.factor(rf_predictions),
                                   reference = as.factor(test_data$confirmed_icu),
                                    positive = "Yes") # set interpretation and metrics to focus
                                                      # on predicting Yes (ICU requirements)
```

Decision Tree Confusion Matrix:

Confusion Matrix and Statistics

```
      Reference
```

Prediction No Yes No 2621 77 Yes 4345 482

```
           Accuracy : 0.4124
             95% CI : (0.4012, 0.4236)
No Information Rate : 0.9257
P-Value [Acc > NIR] : 1

              Kappa : 0.0529
```

Mcnemar's Test P-Value : <2e-16

```
        Sensitivity : 0.86225
        Specificity : 0.37626
     Pos Pred Value : 0.09985
     Neg Pred Value : 0.97146
         Prevalence : 0.07429
     Detection Rate : 0.06405
```

Detection Prevalence : 0.64146
Balanced Accuracy : 0.61926

```
   'Positive' Class : Yes
```

Random Forest Confusion Matrix:

Confusion Matrix and Statistics

```
      Reference
```

Prediction No Yes No 2621 77 Yes 4345 482

```
           Accuracy : 0.4124
             95% CI : (0.4012, 0.4236)
No Information Rate : 0.9257
P-Value [Acc > NIR] : 1

              Kappa : 0.0529
```

Mcnemar's Test P-Value : <2e-16

```
        Sensitivity : 0.86225
        Specificity : 0.37626
     Pos Pred Value : 0.09985
     Neg Pred Value : 0.97146
         Prevalence : 0.07429
     Detection Rate : 0.06405
```

Detection Prevalence : 0.64146
Balanced Accuracy : 0.61926

```
   'Positive' Class : Yes
```

Confusion Matrix interpretation

The Confusion Matrices for both the decision tree and random forest models show the same numbers. This indicates that both models fail to handle the class imbalance effectively, performing poorly in predicting the minority class (Yes), which is critical for this analysis.

Key insights:

- The model is highly sensitive (captures most ICU cases) but low in precision (misclassifies many patients as needing ICU care when they do not).
- The low specificity and poor precision indicate the model struggles with distinguishing between ICU and non-ICU cases, leading to many false positives.
- While it identifies most patients who need ICU (high sensitivity), the trade-off is a lack of reliability in its positive predictions.

Accuracy (41.24%):

Overall, the model correctly classifies 41.24% of the cases. This is a low accuracy and reflects poor performance, especially given the imbalanced nature of the dataset.

Sensitivity (86.23%):

The model correctly identifies 86.23% of actual ICU cases (Yes). This is high and indicates that the model performs well in catching most true positives (patients who need ICU care).

Specificity (37.63%):

The model correctly identifies only 37.63% of actual non-ICU cases (No). This low value means the model generates a high number of false positives, incorrectly predicting ICU needs for patients who do not require it.

Positive Predictive Value (PPV/Precision, 9.99%):

When the model predicts Yes (ICU needed), it is correct only 9.99% of the time. This low precision suggests that the model frequently predicts ICU care incorrectly, overwhelming actual positives with false positives.

Negative Predictive Value (NPV, 97.15%):

When the model predicts No (ICU not needed), it is correct 97.15% of the time. This indicates high confidence in its negative predictions, minimizing false negatives. Balanced Accuracy (61.93%):

This metric averages sensitivity and specificity. It provides a fairer evaluation of model performance on imbalanced data. While better than random guessing (50%), it still highlights room for improvement.

Kappa (0.05):

This value reflects agreement between predicted and actual classifications beyond chance. A score of 0.05 indicates very poor agreement.

Mcnemar's Test (P-Value < 2e-16):

This result indicates a statistically significant difference in the proportions of false positives and false negatives, highlighting imbalance in prediction errors.

Other Metrics (Precision, Recall, F1-Score)

Calculations:

Precision: Proportion of predicted positives that are actual positives. *Precision = TP / (TP + FP)*

Recall (Sensitivity): Proportion of actual positives that are correctly identified. *Recall = TP / (TP + FN)*

F1-Score: Harmonic mean of precision and recall. *F1-Score = 2 x ((Precision × Recall) / (Precision + Recall))*

Decision Tree Metrics: - Precision: 0.1 - Recall: 0.86 - F1-Score: 0.18

Random Forest Metrics: - Precision: 0.1 - Recall: 0.86 - F1-Score: 0.18

Metrics interpretation

Precision: Measures how often the model's positive predictions (requiring ICU) are correct. A high precision minimizes false positives.
Recall: Measures how well the model identifies actual positives (requiring ICU). A high recall minimizes false negatives.
F1-Score: Combines precision and recall into a single metric, useful when there's an uneven class distribution or when both false positives and false negatives are costly.

In summary:

Both models (Decision Tree and Random Forest) are good at identifying ICU patients (high recall), but they struggle with precision. The low precision means that when they predict ICU care, they are often wrong, leading to many false positives. The low F1-Score suggests that the models are not well-balanced and might need further tuning or other techniques (e.g., resampling, adjusting thresholds) to improve precision and overall performance.

Precision (0.1):
Both models have a precision of 0.1, meaning that only 10% of the predictions where the models predicted Yes (ICU needed) are actually correct. This is very low, indicating a high number of false positives, where the model incorrectly predicts that patients require ICU care when they do not.

Recall (0.86):
Both models have a recall of 0.86, meaning that 86% of the actual Yes cases (patients who truly need ICU care) are correctly identified. This is a strong recall, showing that the models are effective at identifying most ICU cases.

F1-Score (0.18):
The F1-Score is a balance between precision and recall. With an F1-Score of 0.18, this indicates that while the models are good at identifying ICU patients (high recall), the large number of false positives (low precision) significantly reduces their overall effectiveness. The low F1-Score reflects this imbalance.

Random Forests model tuning

Tune Hyperparameters using the caret package to help improve the model's performance.

Summary of Tuning Process

Grid Search: perform a grid search over mtry, ntree, and nodesize using 5-fold cross-validation.
Best Hyperparameters: Identified the optimal combination of hyperparameters.
Final Model: Used the optimal hyperparameters to train a final Random Forest model.
Evaluation: Evaluated model performance on the test set using accuracy, precision, recall, F1-score, and confusion matrix.

The caret package allows tuning using cross-validation. The following hyperparameters were tuned:

ntree: The number of trees in the forest.
mtry: The number of variables randomly sampled as candidates at each split.
nodesize: The minimum size of terminal nodes (leaf nodes).

Make predictions on the test data, and calculate the metrics for the tuned model.

Tuned Random Forest Confusion Matrix: Confusion Matrix and Statistics

```
      Reference
```

Prediction No Yes No 2621 77 Yes 4345 482

```
          Accuracy : 0.4124
            95% CI : (0.4012, 0.4236)
No Information Rate : 0.9257
P-Value [Acc > NIR] : 1

             Kappa : 0.0529
```

Mcnemar's Test P-Value : <2e-16

```
        Sensitivity : 0.86225
        Specificity : 0.37626
     Pos Pred Value : 0.09985
     Neg Pred Value : 0.97146
         Prevalence : 0.07429
     Detection Rate : 0.06405
```

Detection Prevalence : 0.64146
Balanced Accuracy : 0.61926

```
'Positive' Class : Yes
```

Tuned Random Forest Metrics: - Precision: 0.1 - Recall: 0.86 - F1-Score: 0.18

Metrics interpretation

The tuned random forest model is good at identifying ICU cases (high recall) but poor at making accurate ICU predictions (low precision). This means the model catches most true positives but generates too many false positives, limiting its practical use in predicting ICU needs.

Accuracy (0.4124): The model correctly predicts the outcome for only 41.24% of cases. This low accuracy indicates poor overall performance.

Balanced Accuracy (0.6193): This average of sensitivity (recall for Yes) and specificity indicates moderate performance, better than random guessing but far from ideal.

Specificity (0.3763): Only 37.63% of non-ICU cases (No) are correctly identified, indicating frequent false positives where non-ICU patients are misclassified as ICU needs.

Precision (0.1): Out of all the cases where the model predicted Yes (ICU needed), only 10% were correct. This suggests many false positives, where the model incorrectly predicts ICU needs.

Recall (0.86): The model correctly identifies 86% of actual ICU cases (Yes). This high recall shows that the model effectively captures most true positives, minimizing missed ICU cases.

F1-Score (0.18): The F1-Score combines precision and recall, reflecting a balance between the two. The low score (18%) highlights that while recall is strong, the model struggles with precision, resulting in poor overall effectiveness.

Other Models consireded

Models that are well-suited for handling imbalanced datasets include XGBoost, LightGBM, and Gradient Boosting Machines (GBM).

In this project, the GBM model will be used to predict ICU requirements to evaluate whether it can deliver better results compared to the two models previously built.

Model #3: Gradient Boosting Machines (GBM)

Simplicity: GBM is simpler to implement compared to other models like XGBoost and LightGBM. It uses a standard boosting framework and is readily available in R through the caret package. The XGBoost and LightGBM models require additional libraries and more complex parameter tuning.

Flexibility with caret: GBM integrates seamlessly with the caret package, which simplifies hyperparameter tuning and cross-validation. The implementation in caret allows for straightforward handling of class imbalance using techniques like SMOTE or class weights.

```r
# Train the GBM model:

gbm_train_control <- trainControl(
  method = "cv",                # Cross-validation
  number = 5,                   # 5-fold cross-validation
  classProbs = TRUE,            # Enable class probabilities
  sampling = "smote",           # Use SMOTE for balancing classes
  summaryFunction = twoClassSummary,  # Evaluate using metrics like ROC
  verboseIter = TRUE            # Print progress during training
```

```
)

gbm_grid <- expand.grid(
  n.trees = c(50, 100, 150),          # Number of trees
  interaction.depth = c(1, 3, 5),     # Depth of each tree
  shrinkage = c(0.01, 0.1, 0.2),      # Learning rate
  n.minobsinnode = c(10, 20)          # Minimum samples in terminal nodes
)

set.seed(324)
tuned_gbm <- train(
  confirmed_icu ~ .,
  data = balanced_train_data,
  method = "gbm",                    # Specify GBM as the model
  trControl = gbm_train_control,     # Use the defined train control
  tuneGrid = gbm_grid,               # Use the tuning grid
  metric = "ROC",                    # Optimize based on ROC
  verbose = FALSE                    # Suppress verbose output from GBM
)
```

Make predictions on the test data, and calculate the metrics.

GBM Model Confusion Matrix: Confusion Matrix and Statistics

```
      Reference
```

Prediction No Yes No 2621 77 Yes 4345 482

```
           Accuracy : 0.4124
             95% CI : (0.4012, 0.4236)
No Information Rate : 0.9257
P-Value [Acc > NIR] : 1

              Kappa : 0.0529
```

Mcnemar's Test P-Value : <2e-16

```
        Sensitivity : 0.86225
        Specificity : 0.37626
     Pos Pred Value : 0.09985
     Neg Pred Value : 0.97146
         Prevalence : 0.07429
     Detection Rate : 0.06405
```

Detection Prevalence : 0.64146
Balanced Accuracy : 0.61926

```
   'Positive' Class : Yes
```

GBM Model Metrics: Precision: 0.1 Recall: 0.86 F1-Score: 0.18

Metrics Interpretation:

Accuracy: 41.24%. The accuracy is quite low, which suggests the model is not doing well overall. Given the class imbalance (with No being the majority class), this metric is not enough to assess the model's performance.

No Information Rate (NIR): 92.57% Interpretation: The NIR represents the accuracy you would get by always predicting the majority class (i.e., always predicting No). This is the baseline accuracy.

Since the NIR is 92.57%, the model's accuracy of 41.24% is far below the baseline, meaning the model performs significantly worse than simply predicting the majority class.

Kappa: 0.0529. Kappa measures the agreement between the predicted and actual values, adjusted for chance. A value of 0.0529 suggests very poor agreement, meaning the model's predictions are not much better than random guessing.

Sensitivity(recall): 37.63% Sensitivity measures how well the model identifies positive cases (Yes in this case). A sensitivity of 37.63% means the model misses more than 60% of the actual ICU cases (Yes), which is quite poor.

This indicates the model misses many cases that will more likely require ICU, which is a problem in this medical context where failing to identify patients requiring ICU could have severe consequences.

Specificity: 86.23% Specificity measures how well the model identifies negative cases (No). While specificity is relatively high, it reflects that the model is good at predicting the majority class (No), but that's not enough for this problem since identifying Yes is more critical.

Precision: 97.15% Precision measures how many of the Yes predictions are actually correct. The precision of 97.15% means that when the model predicts Yes, it's almost always correct. However, this does not mean much since the model fails to predict enough Yes cases in the first place.

F1-Score: 0.54 This metric is a harmonic mean of precision and recall. A F1-score of 0.54 indicates that the model's balance between precision and recall is not great. The model is biased toward precision (correctly predicting Yes when it predicts), but it misses a significant proportion of Yes cases.

Balanced Accuracy: 61.93% This metric is a balanced accuracy takes into account both sensitivity and specificity, giving an average of how well the model performs for both classes. Given the class imbalance, a balanced accuracy of 61.93% is still not good enough, especially considering that the goal is to improve the model's ability to detect Yes cases.

Results

Findings

- 38% of the patients in the dataset were confirmed cases of COVID-19.
- 61% of the patients tested negative for the virus, or were awaiting results when the data was collected.

These facts considerably reduced the sample size, since only the confirmed cases of COVID-19 are relevant to this analysis.

From the total of 220,657 confirmed COVID-19 cases in the dataset:

- 56.04% reported at least one underlying health condition.
- 43.96% reported no underlying health conditions.

Are COVID-19 patients with underlying health conditions more vulnerable to becoming severely ill with the virus, requiring hospitalization and intensive care?

The bar plots show that about 48% of COVID-19 patients with underlying health conditions required hospitalization, including intensive care, compared with a 9.3% of those who did not reported any pre-existing health conditions.

Are individuals with underlying health conditions more likely to die from COVID-19 compared to healthy people?

The graphics show that deaths were 10 times higher among patients with underlying conditions (20.5%), compared with those without reported conditions (1.9%).

What are the most common underlying conditions in COVID-19 patients?

The bar plot showed that the most common underlying conditions, reported by more than 30,000 COVID-19 patients in Mexico, were pneumonia, hypertension, obesity and diabetes.

The next most common was tobacco, reported by 17,109 patients.

Less than 10,000 patients reported other medical conditions like asthma and renal chronic disease.

Can ICU admission requirements be predicted based on underlying health conditions in Covid-19 patients?

Machine Learning Classification Models techniquea were applied to the dataset to predict ICU admissions.

Model Performance:

The low sensitivity (37.63%) indicates that the models are not identifying enough ICU patients (Yes).

The high specificity (86.23%) indicates that the models are correctly identifying No cases, but this is not useful in this context because the focus should be on detecting the minority class (Yes).

Class Imbalance:

The "No" Information Rate (NIR) of 92.57% shows that the dataset is imbalanced, with the majority class (No) dominating.
The model essentially predicts No for most cases, which leads to the poor performance on the minority class (Yes).

Model Tuning:

Hyperparameter tuning did not improve the model's performance, likely because the class imbalance is too severe and the model's ability to detect *Yes* cases remains weak.

Conclusions

- The analysis of this dataset undoubtedly shows that individuals with underlying health conditions are at higher risk for more severe COVID-19.
- Underlying health conditions clearly raise the risk of severe outcomes from COVID-19. From the analysis performed on this dataset, it was found that:
- About 48% of COVID-19 patients with underlying health conditions required hospitalization, including intensive care, compared with a 9.3% of those who did not reported any preexisting conditions.
- Deaths were 10 times as high among patients with underlying conditions (20.5%), compared with those without reported conditions (1.9%).
- The most common underlying conditions, reported in more than 30,000 COVID-19 patients in the dataset, are pneumonia, hypertension, obesity and diabetes. The next most common is tobacco, reported by 17,109 patients. Less than 10,000 patients reported other medical conditions.

These findings in Mexico, are consistent with the results of similar studies conducted in countries such as the United States and China. Thus confirming the negative incidence of the presence of pre-existing health conditions in the evolution of COVID-19 patients internationally.

As far as the ability to predict ICU admission requirements in COVID-19 patients, based on the information on their underlying health conditions, the machine learning classification models applied to the dataset didn't perform very well.

- Models built: decision tree model, random forests model, and Gradient Boosting Machines (GBM) model. Despite hyperparameter tuning and class balancing techniques, including oversampling, none of the models achieved satisfactory results, likely due to the dataset's inherent limitations. Results (xxx% accuracy).
- The models' performance was evaluated using metrics such as accuracy, precision, recall, and F1-score. Confusion matrices were used to assess how well the models classified ICU requirements. Despite limited success in achieving high predictive accuracy, the evaluation highlighted the potential for future optimization and refinement to improve performance.

In general, as a result of the analysis performed, it was shown that people with underlying health conditions are definitely at increased risk if they contract the virus, for which there is no vaccine yet and only limited drug treatment. If these people have symptoms of COVID-19, including fever, cough, or shortness of breath, they should immediately contact their health care provider.

Limitations

The results of this analysis are relative to people who lives in Mexico. Therefore the findings may not be 100% applicable to other countries. It was found that some medical conditions are common among nations, but this can not be considered a trend without further investigation to support it.

Only the 38% of the patients in the dataset were confirmed COVID-19 cases. This reduced the sample considerably. It would have been nice to have the 11% that was awaiting results.

Next Steps:

Address Class Imbalance:
Adjust class weights to make the models penalize misclassifications of the minority class more heavily.

Explore Additional Features:
Investigate whether additional or alternative features can improve specificity and overall predictive power.

Try Other Models:
Consider using other models that handle class imbalance better, like XGBoost or LightGBM, which can be more effective in imbalanced settings.

References

Introduction to Data Science.
Statistics and Prediction Algorithms Through Case Studies. By Rafael A. Irizarry
https://rafalab.dfci.harvard.edu/dsbook-part-1
https://rafalab.dfci.harvard.edu/dsbook-part-2

R Markdown from RStudio
https://rmarkdown.rstudio.com/

Pandoc's Markdown
https://rmarkdown.rstudio.com/authoring_pandoc_markdown.html

RStudio Markdown Cheatsheet
https://github.com/rstudio/cheatsheets/raw/main/rmarkdown-2.0.pdf

R Markdown: The Definitive Guide
Yihui Xie, J. J. Allaire, Garrett Grolemund (2023-12-30)
https://bookdown.org/yihui/rmarkdown/

Kaggle website
https://www.kaggle.com/

CDC website
https://www.cdc.gov/coronavirus/2019-ncov/

Gradient Boosting Algorithm: A Complete Guide for Beginners
https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/

A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning
mastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/