# Introduction to Data Science

# (CS 842)

## Project Report

**By**

## Performance.AI

# Mortality Prediction Using Temporal COVID-19 Health Data


**Submitted by:**

**Samuel Anwo (200388496)**

**Akinmade David (200440384)**

**Instructor: Dr. Ali Manashty**

Department of Computer Science

University of Regina


April 15, 2021

## Table of Contents

## 1. Introduction

COVID-19 has had a key effect in the world and all of our lives. It is a disease caused by a new type of coronavirus named "SARS-Cov-2" which was first discovered in Wuhan, China, and was acknowledged by the World Health Organization on 31st December of 2019. It is a serious acute respiratory condition that could be fatal and consists of varying symptoms like dry cough, fever, weakness, headache, muscle aches, loss of taste and breath, etc. Some important steps to help limit the spread of the disease include utilizing facial masks to cover the nose, properly social distancing, and cleaning hands with sanitizers or soap and water, etc. [1]

This project work developed state-of-the-art machine learning models capable of estimating the chances of a patient dying from the virus and also estimated how long the process would take for each patient. Temporal COVID-19 patient data was sourced online from the government of Mexico's website and was used as the basis for this work, the best models on each data science problem were selected and productionized so that key decision makers in the medical field and their analysts would be armed with accurate information on how best to manage infection outbreaks that may strain their facilities in the future.

## 2. Related work

Chen, X et al (2020) in their paper tried to predict the trend of the number of individuals within 0-60 years old in china with tuberculosis in 2020 based on data available from 2004 -2016. It also examines and compares the performance of three models in accurately predicting the trend of tuberculosis across various age groups. The dataset used in the research paper was acquired from the Chinese health science data center. It consists of the amount of patients who have tuberculosis across all ages in China between 2004 and 2016. The dataset of ages was then subdivided into categories of 10 years. Gray prediction, auto regressive moving average (ARMA) and generalized regression neural network (GRNN) were the metric to assess the regression models and deep neural network. The DNN produced the highest $R^2$ metric of 0.71 in comparison to the linear regression and regression trees which had 0.64 and 0.68 respectively. Similarly, DNN had the least RMS error of 16,841 in comparison to 18,132 and 19,092 of the regression tree and linear regression respectively. The authors also used a scatter plot which highlighted the predicted total cost against true total cost. It was deduced that the prediction error of the DNN model increased as true total cost also increased and the model predicted more

accurately medical cost less than 50,000 dollars. The researchers also examined the DNN model which consisted of less features compared to previous works which in some situations consisted of less data, patient data restricted to specific age groups ( >65 years) and more features. Deep neural networks $R^2$ score of 0.71 outperforms all other algorithms used in previous literature.


*Fig 1 : Data Science Life cycle [3]*

## 3. Project Work Distribution and Tools

The table below shows how the individual aspects of this project were shared between the team mates, Samuel Anwo and David Akinmade who worked on it

| Section of Project work | Student |
|---|---|
| **Collection of dataset and Exploratory Data Analysis** | **Samuel** |
| **Data pre-processing and Feature selection** | **Samuel** |
| **Data transformation** | **David** |
| **Modelling (Classification)** | **David** |

| | |
|---|---|
| **Modelling (Regression)** | **Samuel** |
| **Model Evaluation and Results collation** | **David & Samuel** |
| **Model Productionization** | **David** |

*Table 1: Distribution of duties for the Proposed project*

The following tools were used for the execution of the project

- Programming Language - Python 3.8
- Development IDE - Jupyter Notebook, Google Colab, Spyder,
- Algorithm dev: Sci-kit learn, Keras, Tensorflow
- Result Visualization - Matplotlib, Seaborn
- Code/Model Productionization - Flask, Heroku

## 4. Methodology

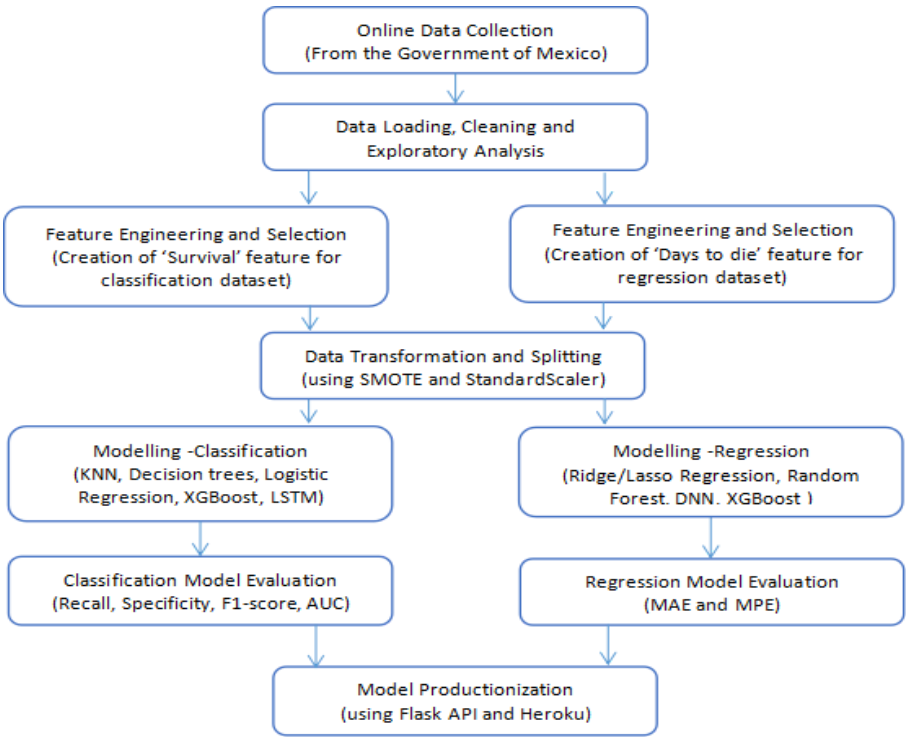



*Fig 2 : High level schematic of the proposed project*

## 4.1 Business Understanding & Relevance

There has been a surge in covid cases especially in developing countries due to the limited amount of vaccines to be administered. Similarly, medical practitioners lack the infrastructure which enables them to prioritise the vaccines based on the most vulnerable individuals. This project could potentially help thousands of publicly and privately owned hospitals in Mexico process millions of temporal COVID-19 data and come up with more accurate estimates of how fast they need to act to save the lives of patients more likely to die sooner than others who are less at risk. This will enable high level directors and practitioners in the medical profession make better decisions especially on how they choose to allocate the resources for the Vaccination process and deliver an optimum level of attention to high-risk patients.

## 4.2 Data Understanding

The data utilised for this project obtained from Kaggle was provided by the Government of Mexico [10]. It consists of details pertaining to COVID-19 virus across hospitals in Mexico. The initial dataset which has a size of 45 MB consists of 566602 instances and 23 features. The COVID-19 Dataset was uploaded to Google colab as a dataframe using pandas which is shown in the figure below:

| | id | sex | patient_type | entry_date | date_symptoms | date_died | intubed | pneumonia | age | pregnancy | diabetes | copd | asthma | inmsupr | hypertension | other_disease | cardiovascular | obesity | renal_chronic | tobacco | contact_other_covid | covid_res | icu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 16169f | 2 | 1 | 04-05-2020 | 02-05-2020 | 9999-99-99 | 97 | 2 | 27 | 97 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 97 |
| 1 | 1009bf | 2 | 1 | 19-03-2020 | 17-03-2020 | 9999-99-99 | 97 | 2 | 24 | 97 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 99 | 1 | 97 |
| 2 | 167386 | 1 | 2 | 06-04-2020 | 01-04-2020 | 9999-99-99 | 2 | 2 | 54 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 99 | 1 | 2 |
| 3 | 0b5948 | 2 | 2 | 17-04-2020 | 10-04-2020 | 9999-99-99 | 2 | 1 | 30 | 97 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 99 | 1 | 2 |
| 4 | 0d01b5 | 1 | 2 | 13-04-2020 | 13-04-2020 | 22-04-2020 | 2 | 2 | 60 | 2 | 1 | 2 | 2 | 2 | 1 | 2 | 1 | 2 | 2 | 2 | 99 | 1 | 2 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 566597 | 01ff60 | 2 | 1 | 13-05-2020 | 03-05-2020 | 9999-99-99 | 97 | 2 | 58 | 97 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 97 |
| 566598 | 047cd1 | 1 | 1 | 07-04-2020 | 06-04-2020 | 9999-99-99 | 97 | 2 | 48 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 97 |
| 566599 | 1beb81 | 1 | 2 | 14-05-2020 | 01-05-2020 | 9999-99-99 | 2 | 1 | 49 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 99 | 3 | 2 |
| 566600 | 16fb02 | 1 | 1 | 31-05-2020 | 29-05-2020 | 9999-99-99 | 97 | 1 | 43 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 97 |
| 566601 | 0021c9 | 2 | 1 | 16-05-2020 | 16-05-2020 | 9999-99-99 | 97 | 1 | 65 | 97 | 1 | 2 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | 2 | 2 | 3 | 97 |

566602 rows × 23 columns

*Fig 3: Raw dataset acquired*

| Feature name | Feature description |
|---|---|
| id | Unique set of characters which identifies a patient |

| | |
|---|---|
| sex | Identifies the gender of the patient (Female - 0, Male - 1) |
| patient_type | Outpatient - 1, inpatient -2. inpatient is admitted to the hospital to stay overnight while an outpatient does not require hospitalization |
| entry_date | The patient's date of entry into the hospital |
| date_symptoms | date of the patients first COVID-19 symptoms |
| date_died | date of death of the patient |
| intubed | Identifies if the patient required intubation (yes - 1, no - 2) |
| pneumonia | Identifies if the patient was diagnosed with pneumonia (yes - 1, no - 2) |
| age | the age of the patient |
| pregnancy | Identifies if the patient was pregnant<br>(yes - 1, no - 2) |
| diabetes | Establish if a patient is diabetic ( yes - 1, no - 2) |
| copd | Establish if a patient has chronic obstructive pulmonary disease?<br>(yes -1, no - 2) |
| asthma | Establish if the patient is asthmatic (yes - 1, no - 2) |
| insumpr | Establish if the patient was immunocompromised (yes - 1, no - 2) |
| hypertension | Identifies if the patient is hypertensive (yes -1, no - 2) |
| other_disease | Identifies if the patient has any other disease different from the specified diseases<br>(yes - 1, no - 2) |
| cardiovascular | Specifies if the patient has any cardiovascular disease<br>(yes - 1, no - 2) |
| obesity | Specifies if the patient is obese (yes -1, no - 2) |
| renal_chronic | Specifies if a patient has chronic kidney disease (yes -1, no - 2 |
| tobacco | Identifies if the patients has a history of smoking tobacco<br>( yes - 1, no - 2) |
| contact_other_covid | Establish if the patient has come in contact with an individual who has been diagnosed with COVID-19<br>( yes - 1, no - 2 ) |
| covid_res | Identifies the patient's COVID -19 test results (positive - 1, negative - 2) |
| icu | Establish if the patient has been admitted into the intensive care unit (ICU) ( |

| | |
|---|---|
| | yes - 1, no - 2) |

*Table 2: Description of features within the dataset*

**Exploratory Data Analysis**

The following diagrams describe the various insights that were obtained during the exploratory univariate and bivariate analysis carried out on the features available in the raw dataset:

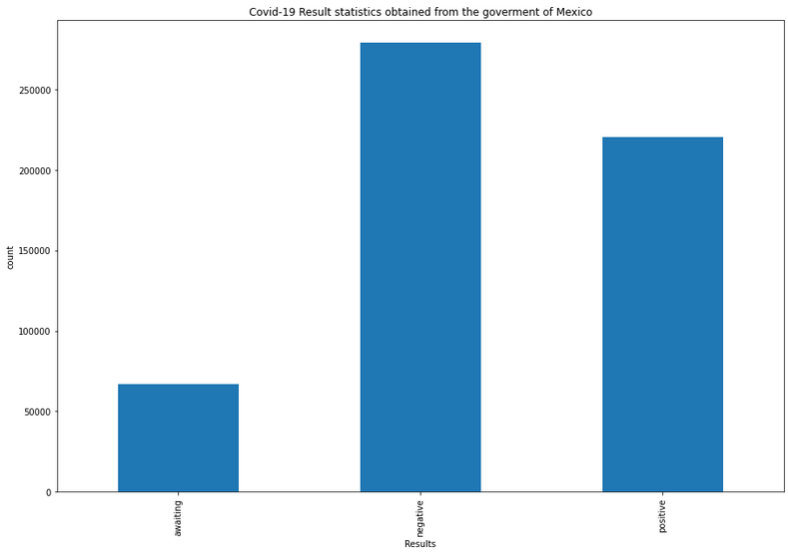



*Fig 4:Covid-19 results statistics obtained from the government of Mexico*

Presence of 'Awaiting' results: For the feature that contains the COVID results for each patient "Covid_res", it was discovered that over 60,000 data-points were still awaiting their results. As this project was only concerned with patients that were already tested as Positive or Negative for COVID, patients with awaiting results were therefore removed.

Correlation between Age and Survival: Bivariate analysis between the age distribution and survival revealed that people aged over 40 years of age were more likely to die from the virus than other members of the age demographic.

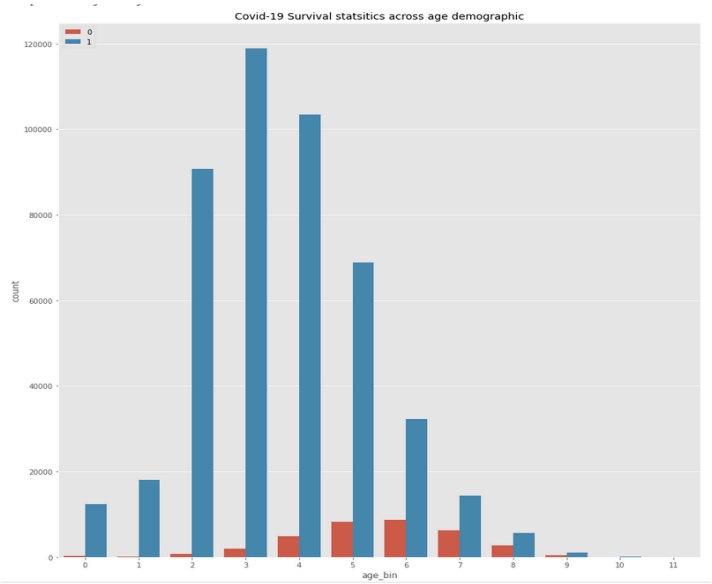



*Fig 5: Covid -19 survival statistics across age demographic*

Correlation between Patient type and Survival: It was also discovered that where the patient was treated was also correlated with the survival of the patient. Patients that were treated at the hospital were more likely to die than patients who received their treatment at home. This could be due to the fact that patients that had to be hospitalised were those who were in the worst shape of all infected patients.

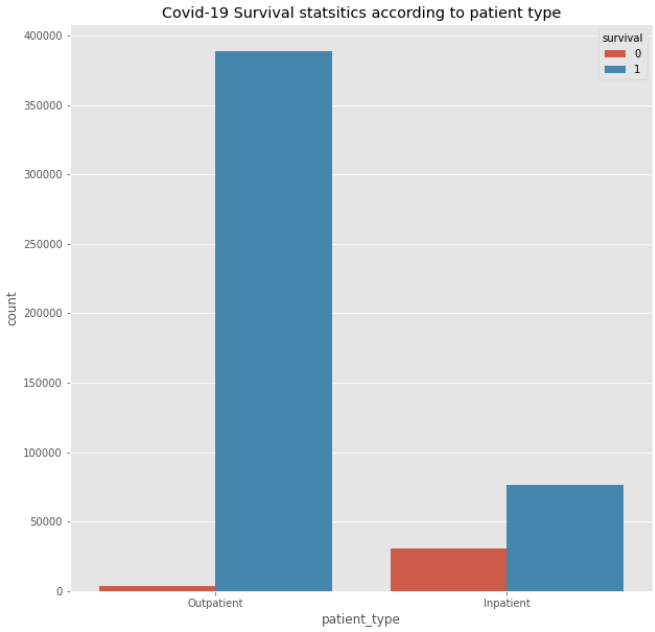



*Fig 6: Covid-19 survival statistics according to patient type*

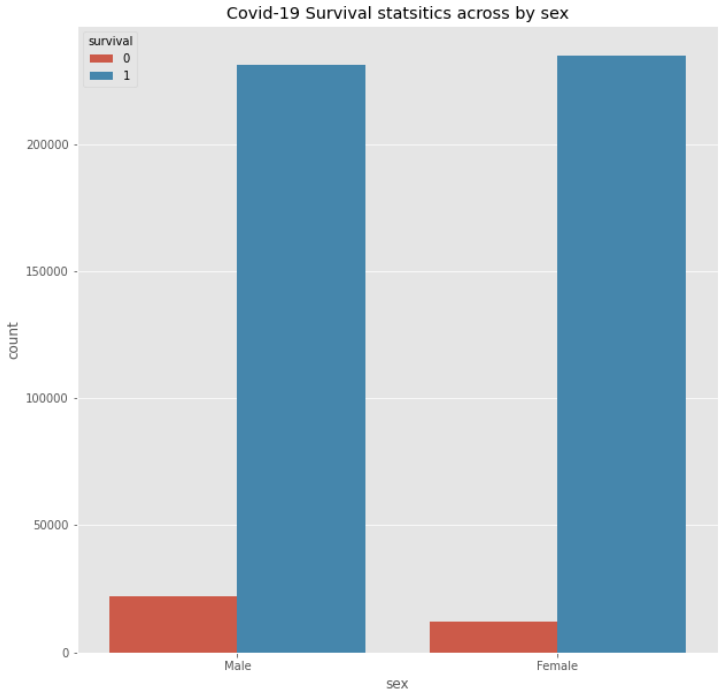



*Fig 7: Covid-19 survival statistics based on sex*

Correlation between Sex and Survival: Also was found that on the average, more men died from the virus than women.

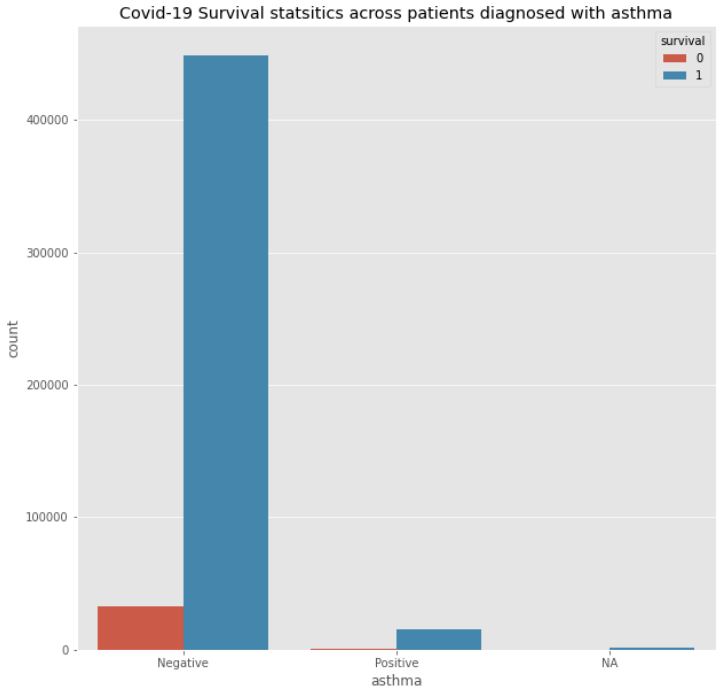



*Fig 8: Covid-19 survival statistics across patients diagnosed with asthma*

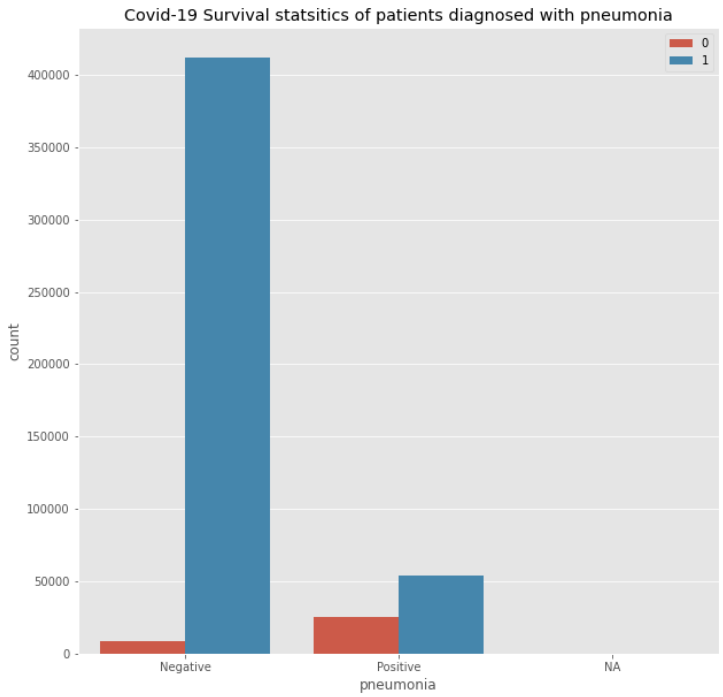



*Fig 9: Covid -19 survival statistics of patents diagnosed with pneumonia*

Correlation between Asthma, Pneumonia and Survival: Patients with respiratory illnesses like Pneumonia and Asthma were found to be more likely to be at risk of dying from the virus than others. This comes as no surprise as the virus itself affects the respiratory functions of the body and it is no surprise that patients with a medical history of respiratory illnesses such as these were more at risk.

## 4.3 Data Preparation

Data preparation is a very important and valuable step in data mining. Depending on the steps taken, the effectiveness of a machine learning model can widely vary. The following steps were taken in the data preparation phase:

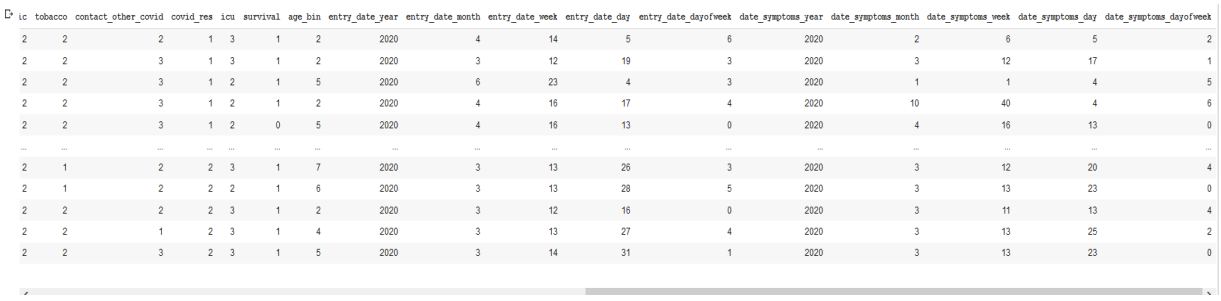
Feature encoding: The features for the date a patient visited the hospital and the date they started showing symptoms of COVID were converted to date time format. The two features were then

encoded by cutting the datetime variable to create five new variables: year, month, week, day, and day of the week

```
[267] #convert to datetime format
      df_class['entry_date'] = pd.to_datetime(df_class['entry_date'], infer_datetime_format=True)
      df_class['date_symptoms'] = pd.to_datetime(df_class['date_symptoms'], infer_datetime_format=True)
```

```
#Encoding the entry_date
df_class['entry_date_year'] = df_class['entry_date'].dt.year
df_class['entry_date_month'] = df_class['entry_date'].dt.month
df_class['entry_date_week'] = df_class['entry_date'].dt.week
df_class['entry_date_day'] = df_class['entry_date'].dt.day
df_class['entry_date_dayofweek'] = df_class['entry_date'].dt.dayofweek
```

*Fig 10: Date Encoding of 'Entry date' feature*



| ic | tobacco | contact_other_covid | covid_res | icu | survival | age_bin | entry_date_year | entry_date_month | entry_date_week | entry_date_day | entry_date_dayofweek | date_symptoms_year | date_symptoms_month | date_symptoms_week | date_symptoms_day | date_symptoms_dayofweek |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 2 | 1 | 3 | 1 | 2 | 2020 | 4 | 14 | 5 | 6 | 2020 | 2 | 6 | 5 | 2 |
| 2 | 2 | 3 | 1 | 3 | 1 | 2 | 2020 | 3 | 12 | 19 | 3 | 2020 | 3 | 12 | 17 | 1 |
| 2 | 2 | 3 | 1 | 2 | 1 | 5 | 2020 | 6 | 23 | 4 | 3 | 2020 | 1 | 1 | 4 | 5 |
| 2 | 2 | 3 | 1 | 2 | 1 | 2 | 2020 | 4 | 16 | 17 | 4 | 2020 | 10 | 40 | 4 | 6 |
| 2 | 2 | 3 | 1 | 2 | 0 | 5 | 2020 | 4 | 16 | 13 | 0 | 2020 | 4 | 16 | 13 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 2 | 1 | 2 | 2 | 3 | 1 | 7 | 2020 | 3 | 13 | 26 | 3 | 2020 | 3 | 12 | 20 | 4 |
| 2 | 1 | 2 | 2 | 2 | 1 | 6 | 2020 | 3 | 13 | 28 | 5 | 2020 | 3 | 13 | 23 | 0 |
| 2 | 2 | 2 | 2 | 3 | 1 | 2 | 2020 | 3 | 12 | 16 | 0 | 2020 | 3 | 11 | 13 | 4 |
| 2 | 2 | 1 | 2 | 3 | 1 | 4 | 2020 | 3 | 13 | 27 | 4 | 2020 | 3 | 13 | 25 | 2 |
| 2 | 2 | 3 | 2 | 3 | 1 | 5 | 2020 | 3 | 14 | 31 | 1 | 2020 | 3 | 13 | 23 | 0 |

*Fig 11: Output of the dataset after encoding the Entry date*

Feature Engineering: The feature named 'Days to Die' was created by subtracting the entry date from the date a patient died in the hospital. This feature was chosen as the target variable for the classification dataset

```
df['Days_to_die'] = (df['dd'] - df['ed']).dt.days
df.head(5)
df.dtypes
```

*Fig 12: Creating the 'Days to Die' feature*

Meanwhile, for the Classification dataset, the target feature named 'Survival' was created by selecting data points where that data died feature was not null.

```
# creating the classfication target feature
df['survival'] = df.apply(lambda x: 0 if pd.notnull(x['dd']) else 1, axis= 1)
df['survival'].value_counts()
```

*Fig 13:Creating the 'Survival' feature*

Data bifurcation: The Initial dataset was subdivided into two, each representing the classification and regression datasets. Data points where patients had ‘awaiting’ COVID-19 results highlighted in figure 5 were dropped. The regression dataset was created by choosing data points where the survival feature was equal to 0 (0 here represents the patient died) while the classification dataset was obtained by selecting data points where the data died feature was not null.

Feature Binning: Next, the ‘Age’ feature was divided into 12 bins using the binning technique to create the’age_bin’ variable. Binning which is used to transform numerical variables into categorical features, helped group the records in the age feature into categories that consists of a range of values (between 0 and 120) which enhances the performance of the machine learning models

```
[ ]  # df.loc[df['age'] == 120]
     df_class["age_bin"] = pd.cut(df_class['age'], bins=12)
     df_class["age_bin"].value_counts()

     (30.0, 40.0]      120831
     (40.0, 50.0]      108229
     (20.0, 30.0]       91388
     (50.0, 60.0]       77040
     (60.0, 70.0]       40945
     (70.0, 80.0]       20613
     (10.0, 20.0]       18230
     (-0.12, 10.0]      12689
     (80.0, 90.0]        8305
     (90.0, 100.0]       1350
     (100.0, 110.0]        56
     (110.0, 120.0]        16
     Name: age_bin, dtype: int64
```

*Fig 14: Age Binning*

Label encoding: Label encoder was imported from the scikit learn preprocessing library to transform the ‘age_bin’ feature into a numeric form suitable for the machine learning models.

```
[24] #LabelEncoding age_bin
     from sklearn.preprocessing import LabelEncoder
     labelencoder = LabelEncoder()

     #refer to value counts above for label assignment
     df_class['age_bin'] = labelencoder.fit_transform(df_class['age_bin'])
     df_class["age_bin"].value_counts()

     3     120831
     4     108229
     2      91388
     5      77040
     6      40945
     7      20613
     1      18230
     0      12689
     8       8305
     9       1350
     10        56
     11        16
     Name: age_bin, dtype: int64
```

*Fig 15: Label Encoding*

Outlier Removal: Z-score function was applied to identify outliers in both data sets. Z score also referred to as the standard metric of an observation is utilized for detecting the amount of standard deviations a data point is from its mean. The Z-score was obtained from the scipy.stats library on sci kit learn. Data points which had absolute scores greater than 3 were eliminated.

```
from scipy.stats import zscore
z_scores = zscore(df_class)

abs_z_scores = np.abs(z_scores)
filtered_entries = (abs_z_scores < 3).all(axis=1)
new_df = df_class[filtered_entries]

new_df.head()
```

*Fig 16: Removing data points that fall outside 3 standard deviation*

Removal of Duplicate and Null Entries: Records which were empty or had duplicate values were eliminated from both datasets.

Data Transformation: Finally, the features of the regression and classification datasets were standardized through the sklearn.preprocessing.standardscaler library in sci kit learn where their mean and standard deviation were represented by 0 and 1 respectively.

```
#scale the data
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)
X
```

```
array([[-1.37358607,  0.31917326,  0.13322966, ..., -0.55207765,
        -1.44957411, -0.32147755],
       [ 0.72802136,  0.31917326,  0.13322966, ..., -0.19238397,
         0.04679156, -0.88209769],
       [-1.37358607, -3.1330945 ,  2.13414333, ..., -0.31228186,
         0.54558011,  0.23914259],
       ...,
       [ 0.72802136,  0.31917326,  0.13322966, ...,  1.36628863,
        -0.451997  , -1.44271784],
       [ 0.72802136,  0.31917326,  0.13322966, ...,  1.60608441,
        -0.95078556,  1.36038288],
       [-1.37358607,  0.31917326, -1.86768401, ..., -1.63115868,
         1.04436867, -2.56395812]])
```

*Fig 17: Standardization of the dataset through standard scalar*

## 4.4 Modelling

In this phase, the aim is to choose machine learning techniques suitable for solving the regression (days to die) and classification (determining if a patient dies or survives COVID-19 disease) problems. This is done through planning and selecting features which enable the model to have its best performance, building each model and carrying out assessment of the chosen models.

### Model Planning

In order to understand and quantify the contribution each feature gives to the overall prediction for the regression and classification problems, a Cross Validated Recursive Feature Elimination model (RFECV) from python feature_selection module is fitted to the entire dataset using a Stratified K-fold in order to check which number of features gives the model the highest level of performance. All the input features from the regression and classification dataset are selected excluding their target variables ‘days to die’ and ‘survival’ respectively. Cross validation is a method for evaluating machine learning algorithms to determine how well it will generalize on an independent dataset.  k-fold cross validation was implemented for the project work. Cross validation is essentially a resampling method where the value of K refers to the numbers of sections the dataset will be divided into

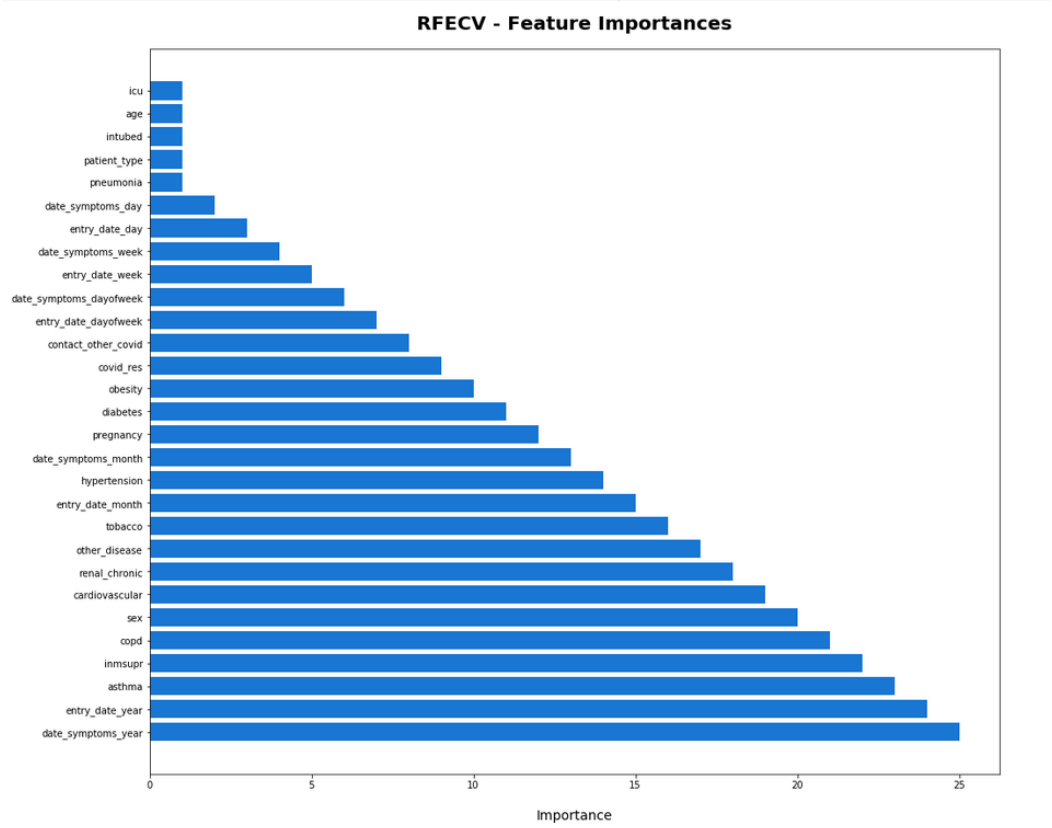



*Fig 18: RFECV -Feature Importances (Classification data)*

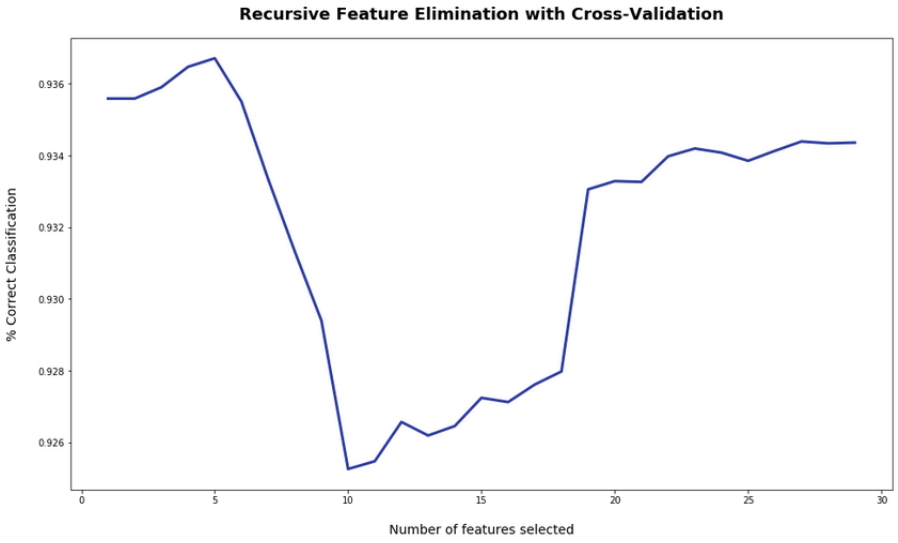



*Fig 19: Number of selected features against the level of accuracy of the model*

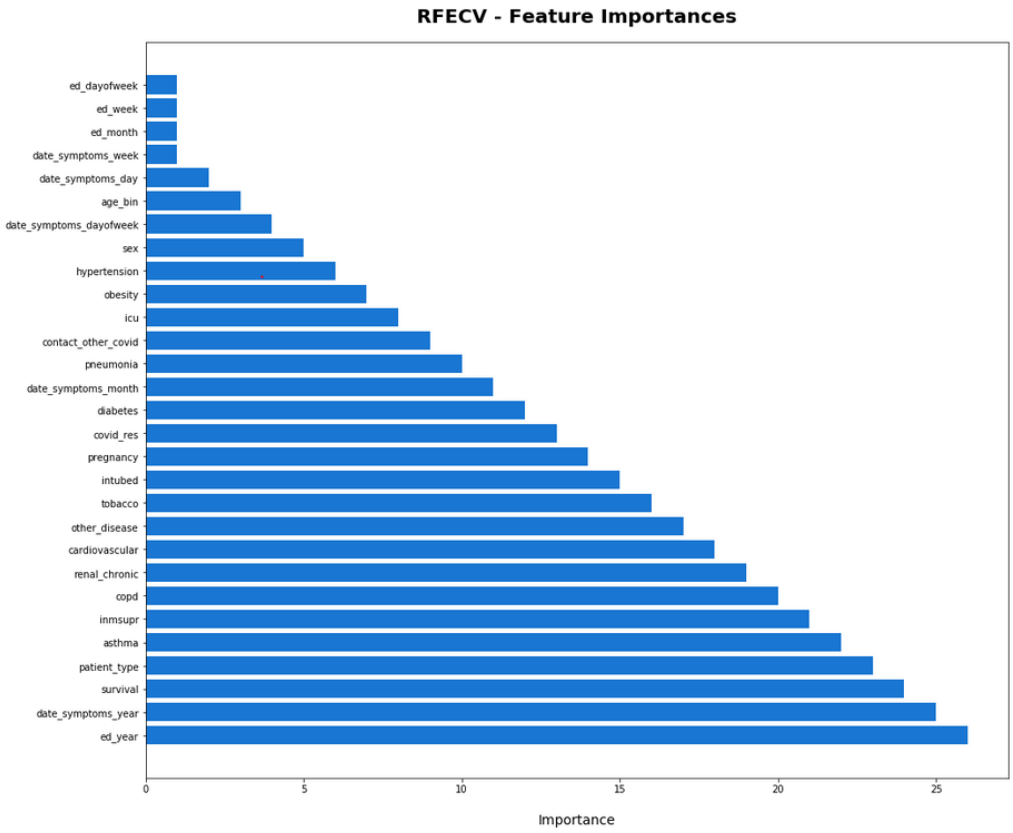



*Fig 20: RFECV -Feature Importances (Regression data)*

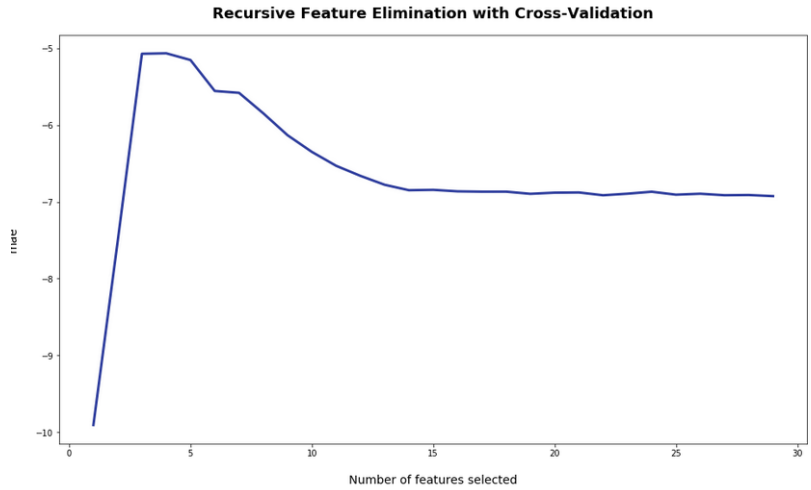



*Fig 21: Number of selected features against the mean absolute error*

For this project, seven (7) different algorithms were used and compared to find the best fit for the regression problem. They include the following :

- Linear regression
- Random Forest regressor
- Ridge regressor
- Decision tree Regressor
- XGBoost regressor
- Recurrent neural network (RNN)
- Long short-term memory (LSTM) networks

Similarly, six (6) algorithms were selected to find the best fit for the classification problem. they include:

- Logistic regression
- KNN
- Decision tree
- XGBoost
- CatBoost
- Long short-term memory (LSTM) networks

XGBoost refers to an enhanced distributed gradient boosting library developed to provide high flexibility and efficiency [5]. It executes machine learning algorithms 'under the gradient boosting library'. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that resolves many machine learning problems. In this project work it was used to resolve the regression and classification problems. Similarly, CatBoost is a machine learning algorithm which utilizes gradient boosting on decision trees. Mamprin, M et al. (2020) in their research work utilised CATboost and XGBoost to predict the one-year chance of death of a patient after undergoing a TAVI [5]. Random forest also referred to as a random decision tree is an ensemble machine learning technique which comprises multiple decision trees. It is a supervised learning algorithm.

A long short-term memory (LSTM) is a type of recurrent neural network which solves the issue of vanishing gradients in plain RNNs by adding records, input and output gates. LSTM was utilized to solve regression and classification problems in this project work. Harrison, E. et al aimed at forecasting the near term mortality of patients hospitalized for cirrhosis at the ICU

department of the University of Virginia using RNN-LSTM and logistic regression model with the aim of incorporating it with electronic health record systems in real time (2018) [6].

**Model Building**

Data splitting**:** The data was divided into 80% train data and 20% by utilising the Train-Test-Split function. The stratify parameter was also set on the target feature to make sure that the number of data points from each class in the target feature was proportionally split into the train and test sets. 80% of the data was used for training used so as to maximize the prediction accuracy of the models especially on the data points that fall in the negative target feature class in the classification dataset.

Data balancing**:** To fix the issue of data imbalance in the classification dataset, the negative class data points in the training set (i.e where Survival = 0) are over-sampled using the SMOTE function from the Imbalanced-Learn python library to match the number of positive class data points. This raises the total number of data points from X to Y, where the new dataset has Z data samples with Survival class of A and also B data samples with Survival class of B, therefore balancing the total number of positive and negative target class labels.

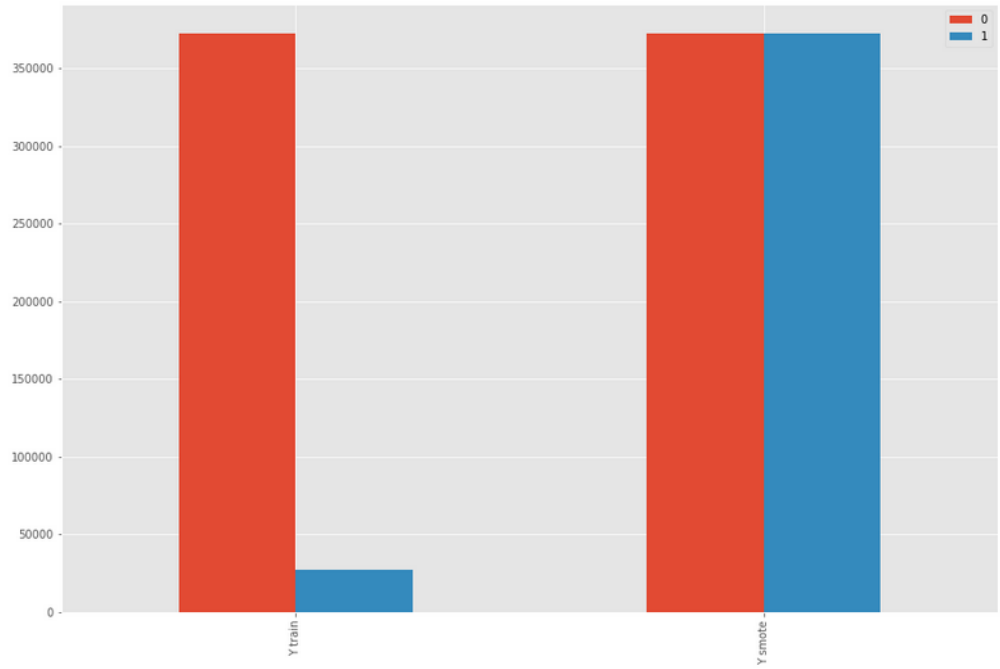



*Fig 22: Synthetic Minority Oversampling Technique (SMOTE) for the classification dataset*

Hyper-parameter Tuning: To ensure that all the models used were operating at their maximum capacity on the datasets, we used GridSearchCV to tune the Hyper-parameters for each of the models before finally fitting them on the datasets to make predictions

```
# using the logistic regression model to establish a baseline performance
lr_model = LogisticRegression()
lr_model.fit(X_smote, y_smote)

tuned_parameters = {'C': [0.1, 0.5, 1, 5, 10, 50, 100]}
grid = GridSearchCV(LogisticRegression(solver='liblinear'), tuned_parameters, cv=3, scoring="accuracy")
grid.fit(X_smote, y_smote)

# printing out its performance
y_pred_lr = lr_model.predict(X_test)
print (accuracy_score(y_test, y_pred_lr))
print(metrics.classification_report(y_test, y_pred_lr))
```

*Fig 23:Tuning parameters with gridsearch*

### 4.5 Evaluation and Results

The models trained were evaluated based on our predefined metrics in order to judge which models performed best for both Regression and Classification problems.

For the Classification problem, the following metrics were chosen:

1. Accuracy: total number of correct predictions / total number of predictions

2. Precision: true positive/ (true positive + false positive)

3. Recall: true positive/ (true positive + false negative)

4. Support: This refers to the number of data samples used from each class

5. F1-score: This refers to the ratio of the product of Precision and Recall to their sum multiplied by two. This metric was very important as it allows us to evaluate how well both the Precision and Recall values are balanced by each model. It can be expressed mathematically as:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

*Fig 24: Formula for F1 score [7]*

For the Regression problem, the Mean Absolute Error (MAE) was chosen as the preferred metric for evaluation.

$$MAE = \frac{\sum_{i=1}^{n} |y_{pred,i} - y_i|}{n}$$

*Fig 25: Formula for F1 score [8]*

## Results

| Algorithm | Accuracy (AUC) | Precision | | Recall | | F1 Score | |
|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 0 | 1 | 0 | 1 |
| Logistic Regression | 0.88 | 0.35 | 0.99 | 0.90 | 0.87 | 0.50 | 0.93 |
| KNN | 0.89 | 0.35 | 0.98 | 0.76 | 0.90 | 0.48 | 0.94 |
| Decision Tree | 0.92 | 0.44 | 0.97 | 0.63 | 0.94 | 0.52 | 0.96 |
| **XGBoost** | **0.94** | **0.53** | **0.97** | **0.59** | **0.96** | **0.56** | **0.97** |
| CATBoost | 0.91 | 0.41 | 0.98 | 0.81 | 0.92 | 0.55 | 0.95 |
| LSTM | 0.89 | 0.38 | 0.99 | 0.86 | 0.90 | 0.53 | 0.94 |

*Table 3: Evaluation of Classification metrics*

The best performing Classification model was XGBoost. It had an accuracy of 0.94 and its F1 score was 0.56 and 0.97 on the negative and positive classes respectively. Coming in close second was Decision Tree Classifier with an accuracy of 0.96 and its F1 score was 0.52 and 0.96 on the negative and positive classes respectively. The results were very acceptable given that they both outperformed the baseline model, Logistic Regression which had an accuracy of 0.88 with an F1 score of 0.50 and 0.93 on the negative and positive classes respectively.

| Algorithm | Mean Absolute Error |
|---|---|
| Linear Regression | 5.55 |
| Ridge Regressor | 5.55 |
| Random Forest Regressor | 5.12 |
| **XGBoost Regressor** | **5.09** |
| Decision Tree Regressor | 6.13 |

| | |
|---|---|
| RNN | 6.52 |
| LSTM | 5.92 |

*Table 4:Evaluation of Regression metric*

For the Regression task, the best performing model once again was the XGBoost Regressor which had a MAE score of 5.09 days. It's performance was followed closely by the Random Forest Regressor which had an MAE of 5.12 days. These models were also very satisfactory as the mean number of days it takes for diagnosed patients to succumb to the virus is 24 days. However, it was noticed that the neural network models did not perform as well as the other models, this could be due to the fact that much less data was available for the neural networks to work with thus preventing the model from reaching its full potential which traditionally requires large amounts of data.

## 4.6 Deployment/Productionization

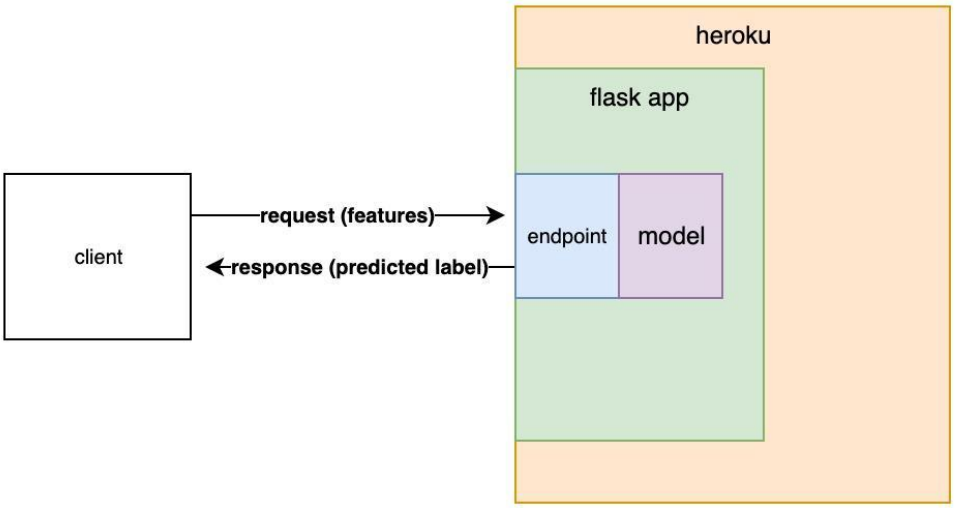



*Fig 26: Schematics of the productionized model [9]*

The final model obtained at the end of the evaluation process was put into production using Python's Flask framework [include reference]. Flask framework is easy to use, has an extensive history of documentation and possesses RESTful request dispatching which is a function that was very handy for the purpose of this project. The following steps describe how this was accomplished:

1. Model Serializing: The top-performing model after the evaluation process was serialized into a character stream using pickling. The pickled model was then exported to the same path as the rest of the project's application in a file called model.pkl.
2. User Interface design: The front end was made using an HTML script that provides a form for the user to input the relevant information about the patient. 16 fields (corresponding to the 16 features required by our model) are provided on the form. The form was then styled using CSS to add more aesthetics to the buttons and the background of the form.
3. API building: Next was the building of the API which receives health information about the infected patient via the user interface, and then computes the value for the estimated number of days the patient has to live based on the final model. To allow the model work with the incoming data from the GUI, it was de-serialized back into a python object. The

   predicted values for each patient are shown by issuing a POST request to /results which receives them in JSON format, then uses the final model to make a prediction and return the value (also in a JSON format) to the API endpoint.

4. Requests Module calling: Using the requests module, APIs from the FLASK application are then called in order to have the predicted number of days for each patient be displayed at the top of the page.

This application can be run on local servers or hosted on web platforms such as Heroku, and details on how it can be installed are included in the setup file attached with this project.

```
(flask_env) C:\Users\akinmade\FlaskAPI>python wsgi.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
[2, 1, 10, 2, 2, 2, 99, 2, 6, 24, 11, 3, 2020, 6, 25, 0]
```

*Fig 27: Example of a query being issued on the backend to the RESTapi*

```
(base) C:\Users\akinmade>curl -X GET http://127.0.0.1:5000/predict
{"response": 3.9072354497354507}
```

*Fig 28: Backend response to the query*

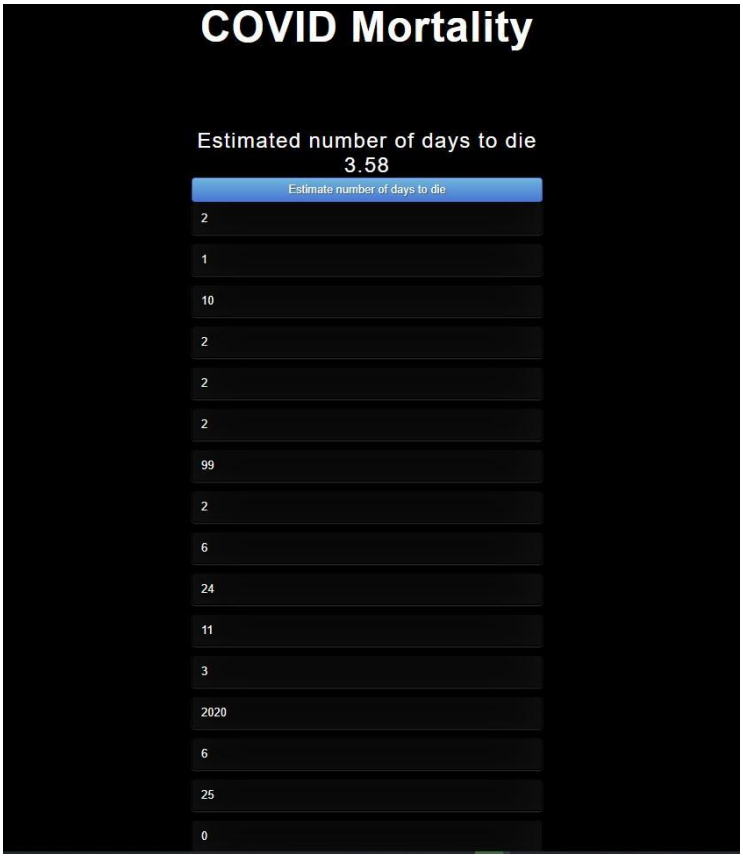



*Fig 29: User interface*

## 5. Extensions and Future Work

The scope of this work covered using COVID data from Mexico only. To capture even more global COVID trends, more data from other nations would be very helpful towards the performance of the models. Also, the neural networks used surprisingly did not live up to the expectations of the performance expected. Alternate layer depths and hyper-parameters can be explored to obtain better results. For the design of the application, even though a simple and elegant interface was used, a bit more functionality can be added so as to include prediction capabilities for Classification problems also. Another technique that would have been worth exploring would be the use of ensemble model stacking which is a way of combining the best of two or more predictions into a single model. This could serve to bring the best of the top performing models like XGBoost and Random Forests to improve the overall performance of the final model.

The project can also be furthered by clustering the results from future predictions of the model such that for a given precinct, state or province the involved government authorities would be well informed on how best to prioritize activities such as vaccination processes.

## 6. Conclusion

In conclusion, we have been able to take the COVID data from the government of Mexico and used it to generate two separate datasets which were used for a regression and classification problem respectively. Separate baseline and state-of-the-art models were executed separately for each problem and the results were evaluated based on the pre-selected metrics chosen. For the regression problem, XGB Regressor emerged as the best performing model with an MAE score of 5.09 days while for the classification problem, XGBoost Classifier emerged as the best performing model with an overall accuracy of 0.94.

It therefore stands to reason that for the particular approach taken in this project, Boosting ensembles seemed to outperform the rest of the models used, although it might just be that this kind of data is well suited to them. Based on the literature regarding Neural networks that we collected, they were purported to perform best on medical data such as this however we do not experience the same results in this project. Overall the results provided on both datasets were very satisfactory especially on the regression data which produced results that we believe would fare well under real world use cases.

## 7. References


1. “Data Preprocessing in Machine Learning: 7 Easy Steps To Follow,” upGrad blog, 25-Sep-2020. [Online]. Available: https://www.upgrad.com/blog/data-preprocessing-in-machine-learning/. [Accessed: 12-Oct-2020]
2. Chen, X., Sa, J., Li, M., & Zhou, Y. (2020). Combined prediction model of tuberculosis based on generalized regression neural network. 2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA). doi:10.1109/icaica50127.2020.9182578
3. https://commons.wikimedia.org/wiki/File:CRISP-DM_Process_Diagram.png
4. Z. Malki, E.-S. Atlam, A. E. Hassanien, G. Dagnew, M. A. Elhosseini, and I. Gad, “Association between weather data and COVID-19 pandemic predicting mortality rate: Machine learning approaches,” Chaos, Solitons & Fractals, 17-Jul-2020
5. Mamprin, M., Zinger, S., de With, P. H. N., Zelis, J. M., and Tonino, P. A. L. (2020, September).Gradient boosting on decision trees for mortality prediction in transcatheter aortic valve implantation.In Proceedings of the 2020 10th International Conference on Biomedical Engineering and Technology(pp. 325-329)
6. Harrison, E., Chang, M., Hao, Y., Flower, A. (2018). Using machine learning to predict near-termmortality in cirrhosis patients hospitalized at the University of Virginia health system. 2018 Systemsand Information Engineering Design Symposium (SIEDS). doi:10.1109/sieds.2018.8374719
7. Mukherjee, T. (2020, July 22). COVID-19 patient Pre-condition dataset. Retrieved April 15, 2021, from https://www.kaggle.com/tanmoyx/covid19-patient-precondition-dataset?select=covid.csv
8. Aggelen, A. (2018, June 06). Linear regression and gradient descent. Retrieved April 15, 2021, from https://blog.arinti.be/linear-regression-and-gradient-descent-89938783cb59
9. 7.I., C. (2019, December 09). Productionize a machine learning model with Flask and Heroku. Retrieved February 15, 2021, from https://towardsdatascience.com/productionize-a-machine-learning-model-with-flask-and-heroku-8201260503d2
10. https://www.kaggle.com/tanmoyx/covid19-patient-precondition-dataset?select=covid.csv