COVID-19 Risk Prediction | ML Course Project

Meir Nizri - 312237563

Introduction

Coronavirus disease (COVID-19) is an infectious disease caused by a newly discovered coronavirus. Most people infected with COVID-19 virus will experience mild to moderate respiratory illness and recover without requiring special treatment. Older people, and those with underlying medical problems like cardiovascular disease, diabetes, chronic respiratory disease, and cancer are more likely to develop serious illness.

During the entire course of the pandemic, one of the main problems that healthcare providers have faced is the shortage of medical resources and a proper plan to efficiently distribute them. In these tough times, being able to predict what kind of resource an individual might require at the time of being tested positive or even before that will be of great help to the authorities as they would be able to procure and arrange for the resources necessary to save the life of that patient.

Using the model described in this report healthcare providers will be able to prioritize patients effectively and thus reduce mortality rates.

Project goals

The main goal of this project is to build a machine learning model that, given a Covid-19 patient's current symptom, status, and medical history, will predict whether the patient is in high risk or not. To this end, we will use several classification techniques in machine learning.

- K-Nearest Neighbours
- Support Vector Machine
- Decision Trees
- Multilayer Perceptron

In each technique we will test a variety of hyperparameters values to get the best model.

In addition, we would like to analyze how each feature, which will be detailed below, affects the chances of getting a severe illness from Covid-19, and consequently understand who the populations at increased risk are.

Dataset

The dataset for this project obtained from Kaggle (link). It was provided by the Mexican government (link). This dataset contains a huge number of anonymized patient-related information including pre-conditions. The raw dataset consists of 40 different features and 1,048,576 unique patients. Since the description of the data and features names was in Spanish, i had to first translate all the features names into English. Thereafter, the following actions were taken to make the data usable.

- All patients who haven't tested positive for COVID-19 were deleted.
- Features with unnecessary and irrelevant information have been deleted.
- For features which have many conclusive values all rows with inconclusive value were filtered.
- For features which have a very few conclusive values the entire feature deleted.
- All the data values modified to mainly ones and zeroes to get it converted into one hot vector.

After processing and cleaning, the dataset consists of 20 features (detailed below) and 388,878 unique patients. The entire data is divided into three groups: train (90%), validation (5%) and test (5%).

1. sex: female or male.
2. age: of the patient.
3. patient type: hospitalized or not hospitalized.
4. pneumonia: Indicates whether the patient already have air sacs inflammation or not.
5. pregnancy: Indicates whether the patient is pregnant or not.
6. diabetes: Indicates whether the patient has diabetes or not.
7. copd: Indicates whether the patient has Chronic obstructive pulmonary disease or not.
8. asthma: Indicates whether the patient has asthma or not.
9. inmsupr: Indicates whether the patient is immunosuppressed or not.
10. hypertension: Indicates whether the patient has hypertension or not.
11. cardiovascular: Indicates whether the patient has heart or blood vessels related disease.
12. renal chronic: Indicates whether the patient has chronic renal disease or not.
13. other disease: Indicates whether the patient has other disease or not.
14. obesity: Indicates whether the patient is obese or not.
15. tobacco: Indicates whether the patient is a tobacco user.
16. usmr: Indicates whether the patient treated medical units of the first, second or third level.
17. medical unit: type of institution of the National Health System that provided the care.
18. intubed: Indicates whether the patient was connected to the ventilator.
19. icu: Indicates whether the patient had been admitted to an Intensive Care Unit.
20. death: indicates whether the patient died or recovered.

The last three features serve as the label. That is, if a patient is intubed or treated in an intensive care unit or dies, he will be classified as at high risk (label 1).

Imbalanced Data

One of the significant challenges we will have to deal with is that the data is unbalanced. Only 15.4% (59,979) of the patients in the clean data are considered at risk while the rest of the patients are not at risk. As a result, we can not measure the quality of a model by its accuracy on the validation set, since it can always predict a patient is not at risk and his accuracy level will be 85%. Instead, we measure the quality of a model by F-measure, also called F_1-score, which measures the accuracy of binary classification. The F-measure is the harmonic mean of the recall and precision.

Moreover, to avoid the algorithms we will use to develop a model that always predicts the patient is not at risk, there are several common methods we can use to solve this problem.

- **Under-sampling** - randomly delete examples from the majority class until the data is balanced.
- **Over-sampling** - randomly duplicate examples from the minority class until the data is balanced.
- **Loss function intervention** - Multiply in the loss function the percentage of error on the minority class by constant C which is the ratio between the number of objects in the data and the number of objects belonging to the minority class.

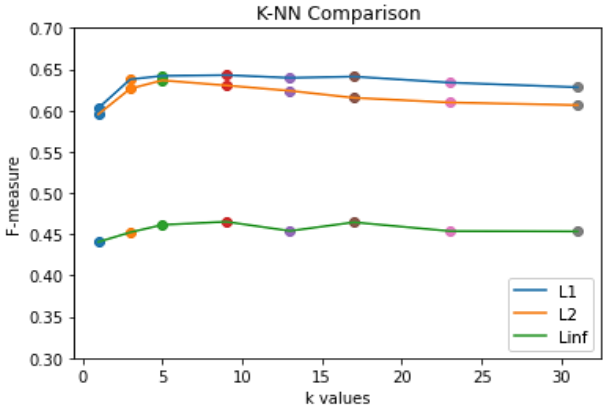
I will use these methods in each of the machine learning techniques and select the model with the best result.

K-Nearest Neighbors

The K-Nearest Neighbours is non-linear classifier in which an object is classified to the class most common among its k nearest neighbours in the train set. The hyperparameters for the KNN algorithm are the value of k and the distance function that calculates the distance between two objects.

I implemented the KNN algorithm as part of Assignment 4 in the ML course and used this implementation here as well. The prediction process in KNN is very slow so to speed it up I decided to use a distance matrix which is a matrix of shape [num validation, num train] where cell [i, j] hold the distance between the i-th validation point and the j-th training point. The main difficulty was how to calculate the distance matrix by each distance function. Fortunately, I found a function in the "scipy.spatial" library that calculates a distance matrix and gets a p value for the distance function as a parameter. Another difficulty was how to implement the algorithm as a generic class that could be run on any dataset. After many attempts I did it successfully.

The k values I chose to check are k=[1, 3, 5, 9, 13, 17, 23, 31]. Each of this k values are tested on three distance functions: Manhattan distance (L_1), Euclidean distance (L_2) and Fréchet distance (L_∞). Since KNN is very slow, the training set can not be larger than a few thousand. Therefore, the only method we can use to balance the data is undersampling. The results of all these different models on the validation set are shown in the following figure. The best model uses the L_1 distance function with k = 9. This model yielded F-measure = 0.64.



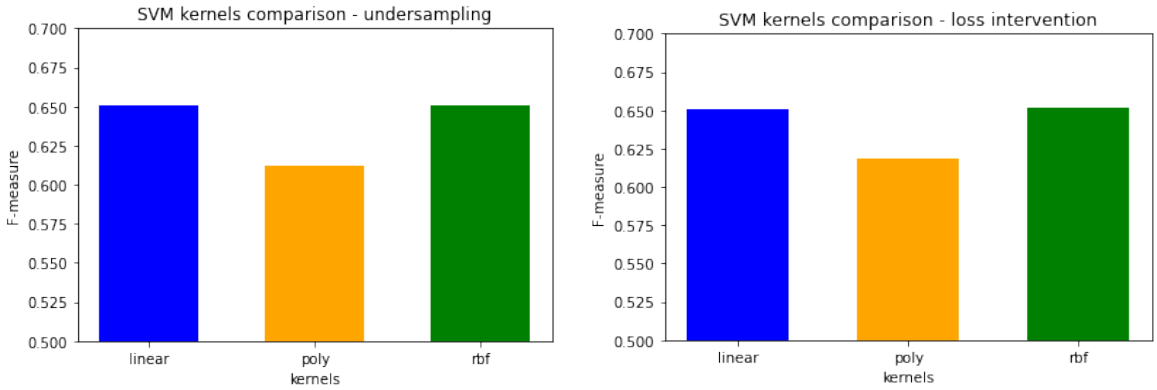
L_∞ yielded the worst results. The reason for this is that the distance in L_∞ is calculated according to only one feature while our data includes many features that are all worth considering simultaneously. In addition, as the value k increases the F-measure slightly decreases. This can be explained by the fact that the data includes many points from both classes that are close to each other, so the more neighbors we consider the more likely we are to find neighbors who are not from the patient class. As expected, the KNN model results are not good enough, as our data includes many patients with the same symptoms but at a different level of risk.

Support Vector Machine

SVM is a classifier that separates data points using a linear hyperplane with the largest amount of margin between the two classes in the train set. SVM can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping the inputs into high-dimensional feature spaces.

I used the "sklearn.svm" library to run the SVM algorithm on the data. I chose to compare between all the kernels that this library allows, which are: linear, polynomial, and RBF (Radial Basis Function) kernels. Since SVM is very slow on large training set we can not use oversampling to balance the data. Therefore, I ran all the models once with undersampling the data and again with loss intervention built-in "sklearn.svm". The results

of all these different models on the validation set are shown in the following figure. The best model uses the RBF kernel with undersampling the data. This model yielded F-measure = 0.65.

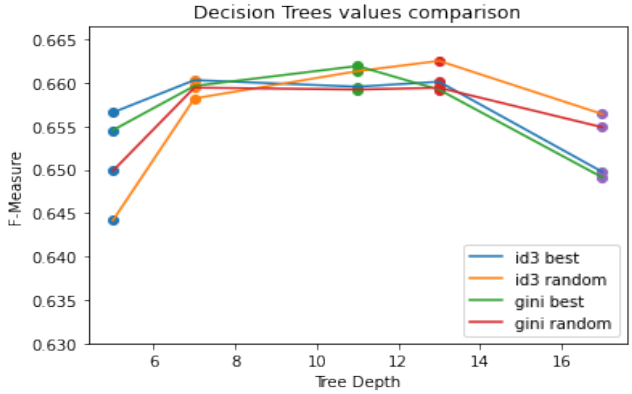


Surprisingly, there is no significant difference between the two data balancing methods nor between a linear classification and a non-linear classification. A possible reason for this is since most of the data is represented as zero's and one's linear planes are sufficient to separate the two classes.

Decision Trees

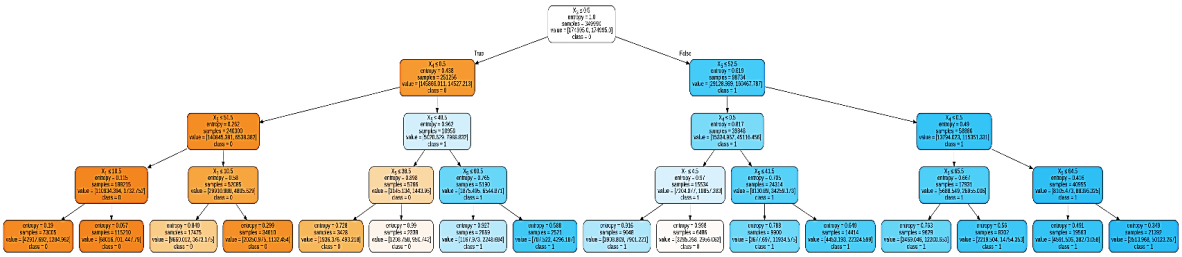
A decision tree is a flowchart-like structure in which each internal node represents a "test" on a feature, each branch represents the outcome of the test, and each leaf node represents a class label. The paths from root to leaf represent classification rules. The decision tree algorithm uses heuristic functions for selecting the splitting criterion that partition data into the best possible manner.

I used the "sklearn.tree" library to run the decision tree algorithm on the data. I chose to compare between the two heuristic functions that this library allows, which are: ID3 and Gini index. For each of these heuristics you can choose the best split or the best random split. In addition, I chose to compare several values of the maximum tree depth that we allow the algorithm to build. The values tested are [5, 7, 11, 13, 17] . The maximum possible depth is 17 as this is the number of features we have in the clean dataset. The results of all these different models on the validation set are shown in the following figure. The best model uses the random id3 heuristic functions with maximum depth of 13 and loss intervention to balance the data.

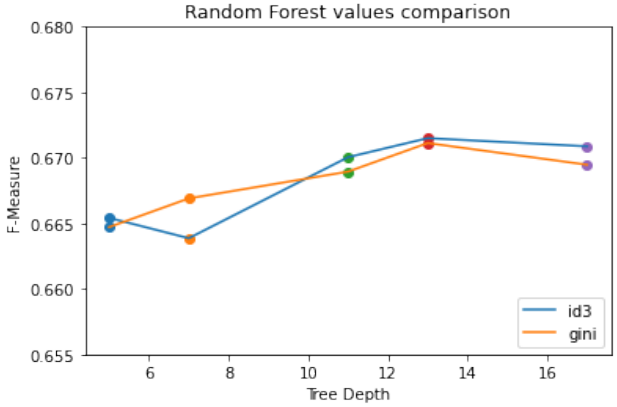


Up to a certain depth the results improve but from this depth the results start to drop. This is because the tree needs to be deep enough to be able to separate all the objects from the two classes. Although from this depth onwards there is a decrease because the VC dimension in decision trees also depends on the depth of the tree. According to the generalization bound a large VC dimension reduces the quality of the model.

By plotting the tree of the best model, we can conclude that the most important features that affect a patient's risk from covid-19 are his age and whether he suffers from pneumonia which mean air sacs inflammation.



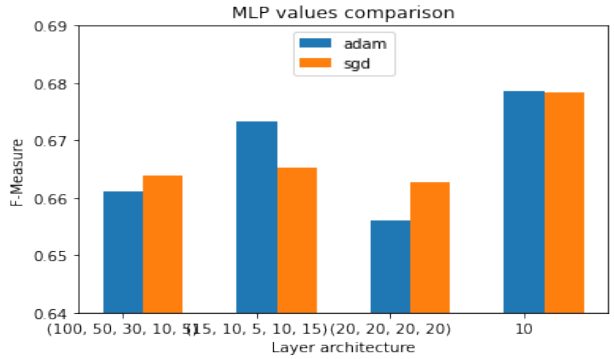
I also tried a model of a random forest using the "sklearn.ensemble" library. I tested this algorithm with the heuristics id3 and Gini index for the same depths mentioned above in decision trees. The results of all the models are shown in the following figure. The best model in decision trees is also the best model in random forest and as expected even improves it.



Multilayer Perceptron

An MLP model consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training.

I used the "sklearn.neural_network" library to run the MLP algorithm on the data. I chose to compare several layers architectures on the two optimizers Adam and SGD. The layers architectures tested are [(100,50,30,10,5), (15,10,5,10,15), (20,20,20,20), (10)]. Since the MLP algorithm works very slowly on large data, we cannot balance the data by using oversampling, so I used undersampling. The results of all these different models on the validation set are shown in the following figure. The best model uses Adam optimizer and only one hidden layer with 10 neurons.

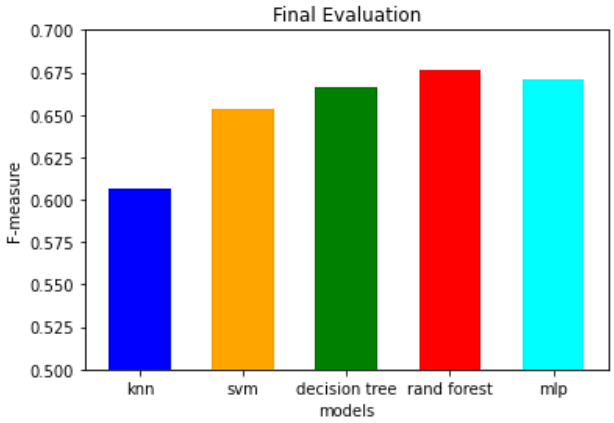


Interestingly, the simplest neural network produced, with a big difference, the best results. This outcome shows that there are no deep connections between the various features in the dataset. That is, a combination of several features, which represent in the data different diseases, do not necessarily affect a patient's being at risk

from Covid-19. In addition, the MLP model provides good results. This outcome reinforces our conclusion from the results of the SVM models, that a planar classifier is good enough on our data.

Final Evaluation

Running the best models from each technique on the test set returns the following results.



From the results shown in the picture it can be concluded that the model with the best result is a random forest. This conclusion is not surprising as studies show that in general a random forest is the best model on a dataset that does not contain photos or videos. More details on the results of the Random Forest model are shown in the following image.

	precision	recall	f1-score	support
0	0.98	0.85	0.91	16389
1	0.53	0.93	0.68	3055
accuracy			0.86	19444
macro avg	0.76	0.89	0.79	19444
weighted avg	0.91	0.86	0.87	19444

Conclusion: the best model manages to classify a high percentage of 93% critical patients successfully while classifying only 15% of the mild ones incorrectly. The overall accuracy is 86%. The difficulty in predicting optimally may be due to the uncertain nature of the covid-19 virus as well as the quality of the dataset.

Link to project Colab notebook:
https://colab.research.google.com/drive/10IxZhzfjOLg3Qv-F2JKAAlDxgG4Knv2n#scrollTo=XnHKMDedt0tB