Machine Learning - Report

Aryan Prakash Rao

I. Problem Framing (Abstract)

With the world struggling in the battle against the Covid-19 Virus, there has been one key development in curbing its outbreak. I am of course referring to the multiple vaccines created over the past 2 years. Biotech firms from multiple countries have developed several vaccines with varying success rates. These countries include USA (Pfizer, Moderna), China (Sinopharm, UK (Oxford-AstraZeneca) and Russia (Sputnik V). These vaccines are crucial in the battle against Covid-19. But, with the emergence of these vaccines, a crucial problem has emerged. I am referring to the shortage of vaccines around the world, there is no clear-cut solution to this issue. One notable country facing this issue is India.

One way to tackle this problem is prioritizing the vaccination of groups who need it the most i.e the people who are most likely to be hospitalized or die due to the Virus. Most countries have adopted some variation of this solution when deciding who is eligible in to receive a dose of the vaccine. Factors such as age, gender, chronic disease history etc. are considered when making such decisions. There is also an argument that vaccinating potential 'super-spreaders' should be preferred over the vulnerable (more likely to die). I will not be taking this argument into account, as there isn't enough valid data (within the dataset) to identify a 'super-spreader'.

So, my goal is to build a machine learning model(s) capable of deciding who is prioritized in a 'limited vaccine' scenario within a single country. The model will factor in features such as age group and gender. The target function for such a model will inherit features from the outcomes of previously recorded cases (registered within the dataset). This model will then be able to generalize to a new set of unseen values and make informed decisions.

II. Overview

To create the proposed model, we would first require some data. The data provided for this project is very messy and will have to be cleaned and binned. Only a few features (columns) of the dataset are required for the proposed model. The main attributes required are the outcomes of Covid-19 cases and the demographics of each victim. After required data is acquired and cleaned, the target function and output variable (Y) will be built using the cleaned data. After the 'ideal' output is created, the input variable will be prepared for the model. To do this, I will be using the 'Dummy Encoding' technique to portray each input feature as a binary value. It is also important to note that the 'outcome' feature will also be removed when feeding data to the machine learning model(s). This is because the models are intended to assign vaccine priority to a set of people who have not contracted the Virus. If the model took in 'outcome' as an input variable, the model would not be able to assign a priority to individuals who have not contracted Covid-19. This defeats the purpose of the model.

This problem is a multi-class classification problem. The output variable can have 4 potential values, which are 'Not Eligible', 'Low', 'Medium' and 'High'. 'Not Eligible' is for individuals below the age of 16 as vaccines like Pfizer and Sinopharm are only applicable to individuals over the age of 16. The other three values are self-explanatory. After both the input and output variables are computed, the data is then fed to the model(s).

This is the general workflow of the solution I am presenting. These processes will be discussed in more detail over the next few pages.

III. Data Preparation

This section will include all data selection, cleaning, binning, analysis, and transformation before feeding data to the various models. The data preparation workflow will consist of 4 separate processes. The first is 'Feature Selection' where we choose 'features' (columns of the dataset) relevant to the model. Next is 'Data Cleaning + Binning', this section includes all cleaning and grouping techniques applied to the 'raw' dataset; this part is crucial to the analysis section. The analysis section will include visualizations of key trends seen when observing the data. The final process is 'Data Transformation', this section includes techniques used to transform data into an input appropriate for each of the machine learning models. This section will also include details of the algorithm used to generate the 'ideal' Output variable for the target function.

A. Feature Selection

The dataset used [1] is stored within 'latestdata.csv' and contains 34 different columns. Some of these columns contain unique identifiers and some contain statistical information irrelevant to the problem. The first step to conceptualizing the problem is choosing features that are relevant to the problem. This process is done when reading the dataset file itself to ensure only required columns are read from the dataset. This can be done specifying the 'usecols' attribute from the pandas function read_csv.

All unique identifiers and location information were automatically ruled out. Although, country information could be useful in figuring out whether an individual is more likely to die from Covid-19. But this information is not necessary as I am dealing with vaccine shortages in a single country. This already rules out 18 columns of the dataset. Next, all date-based columns were ruled out as they aren't necessary for the models. This rules out 5 more columns. Out of the 11 remaining columns, 6 could potentially be used in determining vaccine priorities. These include 'age', 'sex', 'outcome', 'chronic_disease_binary', 'chronic_disease', and 'additional_information'.

From this set 'age', 'sex', 'chronic_disease_binary' and 'outcome' are selected. The feature 'chronic_disease' was not selected as there wasn't enough data recorded in this column. Out of 2 million cases only 200 or so had a registered chronic disease listed so 'chronic_disease_binary' is chosen instead. It has only 2 potential values, so no grouping needs to be done as well. The feature 'additional_information' was also left out, this is because it is almost impossible to group information from this field. It also has various types of information. Only a small amount of this information can be considered useful, like occupations of individuals. Information regarding occupation can be useful to identify essential workers who would be given a higher priority due to the nature of their job.

Coming back to the features chosen, I will now provide an explanation as to why each feature is relevant. The virus has varying effects on people depending on factors such as age and gender. Therefore ‘age’ and ‘sex’ are chosen. People with chronic diseases tend to have a higher mortality when contracting the virus. The ‘chronic_disease_binary’ field can convey this relation to the model. Finally, the ‘outcome’ field can be used to build the output variable based on actual documented cases. It is to be noted that the ‘outcome’ will not be an input variable for the model. It can be considered as a sort of reference point for the target function.

B. Data Cleaning + Binning

After the appropriate columns are selected, the cleaning process can begin. To start, all rows with a NaN(Not a number) element is removed from the data frame. This reduces the size of the data frame to around 33,000. There were originally more than 2,000,000 rows. This step concludes the bulk of the cleaning processes.

The outcome and age fields have 27 and 126 unique values, respectively. These values need to be grouped appropriately to be useful to the model. This is done using the map function offered by pandas. The outcomes were manually grouped to three separate values, namely, ‘Survived’, ‘Hospitalized’ and ‘Died’. The ‘Hospitalized’ group contains a frequency of over 20,000 as a lot of cases were listed under ‘Hospitalized’. The ‘Hospitalized’ cases could be grouped under ‘Survived’ but the outcome of these (‘Hospitalized’) cases is unclear. So, I consider ‘Hospitalized’ as a sort of middle ground between ‘Survived and ‘Died’. The ages were grouped into 5 separate age ranges. The ranges are ‘0-16’,’17-39’,’40-59’,’60-79’ and ‘80+’.

The chronic_disease_binary and sex attributes are left untouched during the binning process as they already, only contain 2 distinct values.

C. Data Analysis

Now that the raw data has been cleaned and grouped, we can finally analyze the data. There are a total of 20 input variable combinations within the dataset. (outcome is excluded). There are only 125 cases with chronic-disease - binary set to True. This is not representative of a real-life scenario, so I artificially increased this amount in the dataset to 10,010. This was done by duplicating the 125 actual datapoints. This method can be justified as there are only 20 input combinations. This also resulted in a more diverse set of age groups which is also closer to a real-life scenario. *Refer to Figs 1,2,3,4* to view demographic data for sex, chronic disease, age and outcome respectively.

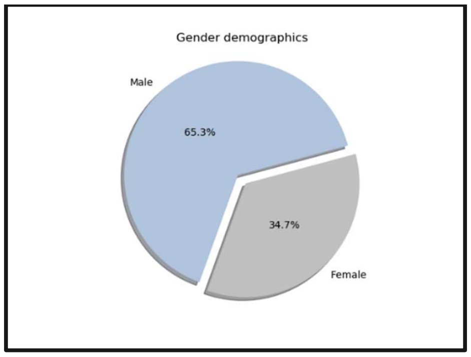

Fig 1

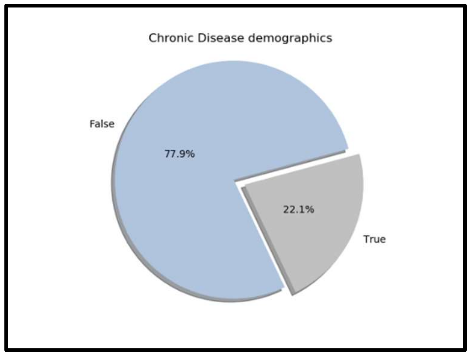

Fig 2

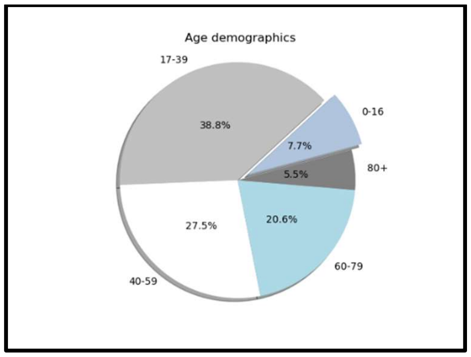

Fig 3

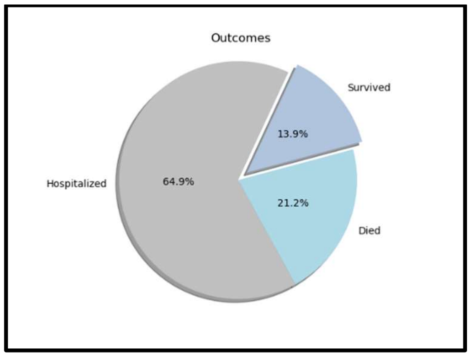

Fig 4

On observing the data, you may think the gender is imbalanced and that this may cause a bias. I have left this mismatch in as I don’t mind the models being slightly biased in this aspect. In general, men have a higher fatality rate when it comes to covid-19(2.8%, women have a 1.7% fatality rate) [2]. The age and chronic disease stats are pretty similar to real-world stats and are thus balanced (except for age 0-16, but they aren’t eligible anyway).

From this we can conclude that the dataset is mostly balanced. Now, I shall plot each demographic against outcome frequency as a percentage. *Refer to Figs 5,6,7.* From these figures I will be able to make conclusions which will help in designing the output variable and the target function itself. It is to be noted that the death rates in Fig .5 and 7 are inflated due to the artificial increase in the number of chronic cases. But roughly the same conclusions can be made even if the chronic cases were not artificially increased. Increasing death rate will not really affect the machine learning models directly as the ‘outcome’ is not used as an input variable.

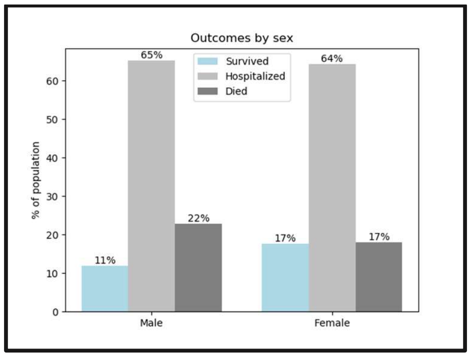

Fig 5

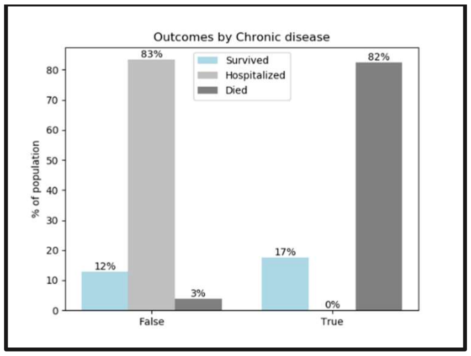

Fig 6

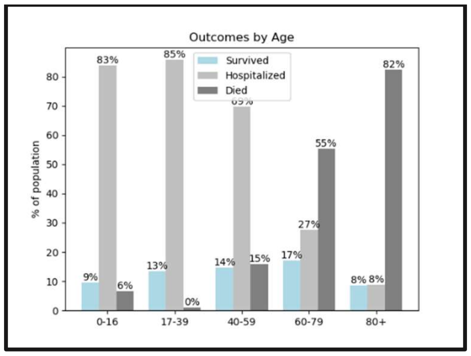

Fig 7

From *Fig 5,* we can ascertain that our statement that Covid-19 is more deadly for males is indeed correct. In *Fig 6* we can clearly see that individuals with chronic diseases are more likely to die from Covid-19. Finally, in *Fig-7* we can observe that the Covid-19 fatality rate tends to increase with age. This is also supported by figures in [2]. These conclusions will be factored in when building the output variable.

D. *Data Transformation*

All data within the dataset is mapped to an integer value. The age field is converted to a set of 5 separate binary columns denoting each age group using the 'Dummy Encoding' technique. This is done through the pandas' function 'get_dummies'. After this, the 'outcomes' column is removed from the dataset.

After the input variables are set up, the output variable can be designed. The output variable will have 4 values namely, 'Not Eligible':0,'Low':1,'Medium':2,'High:3. 'Not Eligible' will contain all members of the 0-16 age group. 'Low' will mostly contain members from the 17-39 and 40-59 age groups. 'Medium' will mostly contain members from the 40-59 and 60-79 age groups. 'High' will contain members from the 80+ age group, members with chronic diseases and all members whose outcome was registered 'Died'. These decisions were made based on the conclusions from the Data Analysis section.

Now, we finally have both X (Input variables) and y (Output variable) data formatted and ready to serve as an input to the model.

IV. Machine Learning Models Used

Now that the data is prepared, we may try inputting the data to the predictive models. As mentioned earlier, the problem is a multi-class classification problem. So, we will need models capable of solving a multi-class classification problem. The three models I have chosen for this problem are 'Naïve Bayes', 'Logistic Regression' and 'Decision Tree Classifier'. Each of these models can solve multi-class classification problems and are available through scikit-learn.

For each model, I will use cross validation to ensure there is no overfitting. The technique I am using for this is called 'Stratified k-fold Cross Validation'. k-fold Cross validation is a re-sampling procedure where the dataset is split into k groups. Among the k groups one group is held out as the test set and the rest are used for training. Model accuracy is calculated for each fold as the test set (k iterations) [3]. K-fold cross validation can be done with random sampling, but I am using stratified sampling. Stratified sampling ensures that the demographics of each of the k groups are representative of the whole dataset [4]. If there is 60:40 split between male and female in the dataset, stratified sampling will ensure that each of the k-groups will also contain the same 60-40 gender split.

The metrics used for comparison will be the precision score, recall score and the accuracy. Runtimes will also be compared. The accuracy I am looking for is in the range of 92.5% to 94.5%. Anything above 95% would be overfitting. This is because of how the output variable (y) was set up. The application of real-world data in defining the target function gives few members of lower risk groups positions in the high-risk category. This is because a small fraction of these age groups has also perished from the virus. Around 3-4% of all datapoints fall under this category. Hence, a 95% + accuracy would be a clear indicator for overfitting.

A confusion matrix will be plotted for each model. The values in the confusion will be normalized to a 0-100 scale. An average of the row-mean and column-mean will be used in the normalization process.

A. *Naïve Bayes*

Naïve Bayes is a probabilistic algorithm which applies the Bayes theorem to predict an output value. The algorithm creates frequency tables of each input attribute and then finds the likelihood probability of triggering a certain output based on the input combination [5]. The main reason I have chosen this algorithm is that it has a fast runtime due to its simplistic nature.

I have used 'Gaussian Naïve Bayes Classifier' from the scikit-learn module. There is no algorithm-specific hyperparameter for this model. The highest accuracy obtained was 82.1% with k (number of folds) set to 350. Lowering the value of k decreases accuracy of model. Increasing k beyond 350 also decreases accuracy. *Refer to Fig 8* for the confusion matrix obtained from the model with 82.1% accuracy.

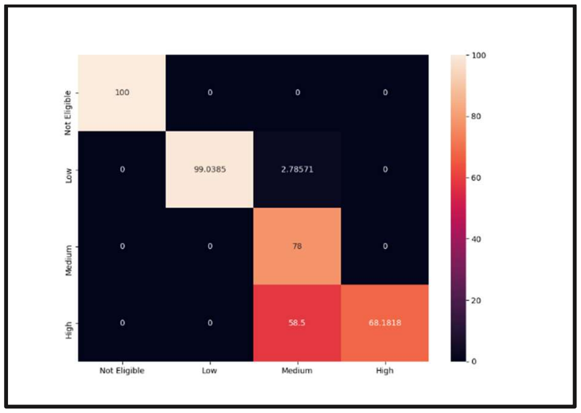

Fig 8

As you can see, the model struggles to predict 'High' and 'Medium' outputs. The accuracy obtained for the 'Low' output is very good, but this is not that important. It is most important to predict the 'High' output accurately. The algorithm seems to have put too many members from the '60-79' age group in the medium category.

The main issue with this algorithm is assumes every feature is independent of each other. This is not the case for features such as age and chronic disease. Older people are more likely to have chronic diseases i.e they are interdependent. Another issue is that the dataset is too small for the algorithm. Naïve Bayes works better on larger datasets and 42000 datapoints is a bit too small.

B. Logistic Regression

Logistic Regression is a popular technique used for solving classification problems. The algorithm is probabilistic in nature. It observes trends found in input and output data and makes predictions. I have chosen this algorithm because it is easy to implement and that it makes no assumptions based on distributions of classes in feature space [6].

I have used the Logistic Regression model provided by scikit-learn. The only hyperparameter for this model is the maximum number of iterations (max_iter). This value was set to 200 to prevent a Convergence Warning. If this value is decreased, the warning will be issued during runtime. An accuracy of 94.2% was obtained with k (number of folds) set to 75. *Refer to Fig 9* for the confusion matrix of this model.

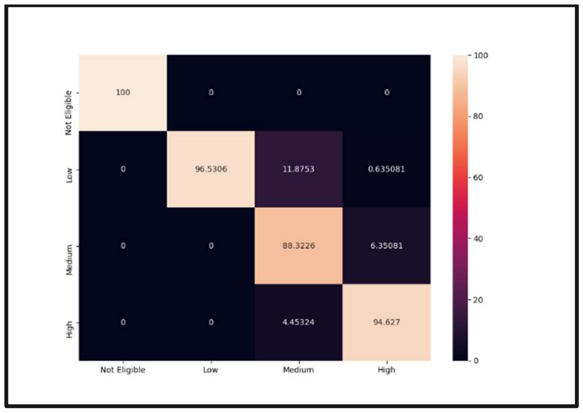

Fig 9

This is almost an ideal result. The 'High' category has 94.6% which is very good. The value of k was tuned to 75 to get an accuracy of around 93-95%. For k values below 50 the accuracy is usually around 92-92.5%. For k values above 100 the accuracy will go over 95% which is overfitting. So, the logistic regression algorithm has managed to achieve the target accuracy.

The main issue with the logistic regression algorithm is that takes too much time to run. For 75 folds it takes around 4-5 minutes to run. If a larger dataset with more attributes was used this would be an issue.

C. Decision Tree Classification

'Decision Tree' is an algorithm which divides data into several subsets to form a tree-like structure. All partitions in the tree are binary [7]. I have picked this algorithm due to its simplicity. Due to this, the algorithm performs tasks almost instantaneously.

I have used the Decision Tree Classifier provided by scikit-learn. The hyperparameter, max tree depth was set to 3 to improve run time (increasing this does not affect accuracy). An accuracy of 94.2% was obtained with k (number of folds) set to 75. *Refer to Fig 9* for the confusion matrix of this model.

The Decision Tree model always obtains the exact same result as the Logistic Regression model (same dataset sample is used). This is mainly due to how the Output Variable was mapped and similarities between the two models. The key difference between the models is the runtime. The Decision Tree model is computed almost instantaneously whereas the Logistic Regression model takes almost 5 minutes.

V. Comparison And Conclusion

To perform a proper comparison between the three models, each of the models are applied to the same random state of the dataset using the Stratified K-fold cross validation technique with k =75. Here are the results: (The random state used for this comparison is different from the one used in the previous section; hence, the result will vary)

Metric	**Model Comparison**		
	Naïve Bayes	**Logistic Regression**	**Decision Tree**
Accuracy (%)	78.01%	94.76%	94.76%
Precision(%)	87.7%	95.18%	95.18%
Recall(%)	80.21%	95.34%	95.34%
Runtime(s)	1.84s	198.25s	1.41s

Naïve Bayes has a good runtime with bad accuracy. Logistic Regression has the best accuracy with a bad runtime. Decision Tree is best in runtime and accuracy. Clearly, we can conclude that the Decision Tree classification algorithm is the best suited model for this problem. If a larger dataset with more attributes was available, Decision Tree would still outclass its peers. We can also see that the results obtained from the Logistic Regression and Decision Tree algorithms are identical if the same random state is used to evaluate both models. This is mainly due to how I have mapped the target function. The only issue is that there are too few input variables used, important attributes like height, weight etc. are not involved due to lack of data. All in all, I believe I have managed to aptly solve the problem I proposed. I have found a model which managed to achieve my target accuracy in a computationally efficient way and that was my goal.

References

[1] https://github.com/beoutbreakprepared/nCoV2019/tree/master/latest_data

[2] https://www.worldometers.info/coronavirus/coronavirus-age-sex-demographics/I. S

[3] https://machinelearningmastery.com/k-fold-cross-validation/

[4] https://www.geeksforgeeks.org/stratified-k-fold-cross-validation/

[5] https://www.analyticsvidhya.com/blog/2021/01/gaussian-naive-bayes-with-hyperpameter-tuning/

[6] https://www.geeksforgeeks.org/advantages-and-disadvantages-of-logistic-regression/

[7] https://webfocusinfocenter.informationbuilders.com/wfappent/TLs/TL_rstat/source/DecisionTree47.htm

.