Prediction of COVID-19 Patients – ICU/General Ward

Bhavesh Khatri
2018030
bhavesh18030@iiitd.ac.in

Mohd. Huzaifa
2018158
huzaifa18158@iiitd.ac.in

Navam Sundriyal
2018162
navam18162@iiitd.ac.in

Abstract

Keeping in mind the COVID-19 outbreak, and limited medical resources, it has become essential for the government to allocate the available resources in treating the right people, at the right time. It is observed that a patient's initial conditions and previous health records have a relation with being admitted to ICU and post health conditions. Furthermore, there are many cases where patients are not given ICU based on their current situation. But later, due to some previous comorbidities, their condition got worse and resulted in death. On the other hand, due to the increasing number of cases daily and limited health care resources, it's quite important that we utilise these resources in the best way possible. That is, we cannot afford to admit more patients to ICU without any reasonable logic. Thus we aim to develop a model that would recommend whether or not a patient should be admitted to ICU based on his initial conditions and previous health record. This way, patients who died after not being admitted to ICU would have better chances of survival and at the same time, unnecessary ICU allocations would be minimised.
Github Repository Link:
https://github.com/sphinx1618/ML-project

1. Introduction

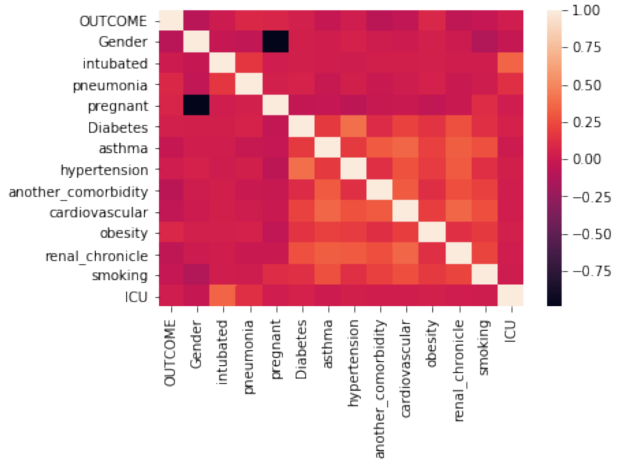
COVID-19 patients with a history of comorbidities like hypertension, obesity, chronic lung disease, diabetes, and cardiovascular disease have the worst prognosis and most often end up with deteriorating outcomes such as ARDS and pneumonia. Our main aim is to create a prediction model/ classifier which will take patients' data including age, gender, pregnancy status, previous health records, etc. as input and would predict whether or not the patient should be admitted to ICU(at present or in near future). Thus, we started by searching for some available data sets having records of patients' previous comorbidities, severity of treatment used and post diagnosis conditions. There were obviously some redundant and irrelevant features in the data set. Therefore, a thorough preprocessing is done along with dimensionality reduction. The number of patients admitted to ICU are obviously way less than the total number of patients. Therefore, suitable data balancing techniques are used. For evaluation also, a suitable metric like precision, recall and f1 score is used. Admitting the needy patients to ICU is of utmost importance, obviously more important than preventing the unnecessary ICU admissions. Therefore, we focused more on the True Positive rate compared to the False positive rate, but at the same time, we also have to maintain a balance between the two.

2. Literature Survey

Pool Testing for COVID-19: Suitable splitting procedure and pool for India[1]. The only way to control the coronavirus spread is to detect already infected persons and isolate them. With limited testing facilities, we need to minimise the number of tests done on a given pool without compromising on detection of the infected ones. The author aimed to generalise an equation for predicting the number of tests to be done in a given pool of data in order to detect every infected person in it. This would minimise the number of unnecessary tests done before. The percentage of tests required varies and depends on various factors like size of pooled sample, probability of an individual being affected in the population in a given area. Since the number of tests required to actually detect all the infected patients would be much less than individual testing, this will certainly help in increasing testing capacity and thus get a better hold on the situation.

Scoring systems for predicting mortality for severe patients with COVID-19[2]. The authors aimed to analyse the risk factors for mortality of severe patients with COVID-19. Via this study, they want to establish a scoring system that would give a prediction of in-hospital deaths, given the details about the hospital and clinical conditions of the patients. For this, they

Figure 1. Correlation Heat Map



collected patients' clinical data from different hospitals in Wuhan and analysed them. After doing the analysis, they created a model using Lasso Regression and multivariate analysis. As a result they found out various interesting things like the mortality rate in males and females depending upon their age, strength of antibiotics received based on the severity of the patient's condition, comparison of treatment and outcomes in the training cohort.

3. Dataset

The dataset contains medical information of 2.63 Lakhs Mexican people and is available at Kaggle [3]. The attributes includes the timestamps :- date when record last updated, when symptoms were first observed, death, lag in days between a reported case and test result dates. It contains the information about the comorbidities (if any). It includes: hypertension, diabetes, pneumonia, asthma, obesity and few more. It also contains more details about the patient origin, state(within the country), so that we can also analyse the effect of workload on the patient's death. Now coming to outcome attributes; it contains the test results (RT-PCR) of patients , whether they were moved to ICU or not, or if the patient died or not. Common Attributes values and what they stand for 1 : YES ; 2 : NO ; 97 : NA; 98 : Ignored; 99 : Not specified The correlation between various attributes of data is shown in the form of a heat map in figure 1.

4. Methodology

Our dataset was slightly imbalanced, so accuracies won't reflect the correct measure for our predictions. As we are predicting the ICU admissions, that means there should be least scope for false negatives. But, false positives also matter as we want to manage resources as well. Therefore, there is a trade off between precision and recall.
Metric used : F1-score and Roc-Auc score.

4.1. Baseline Models

We trained multiple machine learning models, including Naive bayes classifier, Logistic Regression, Random Forest classifier, Support Vector Machine and XGBoost classifier. These were the baseline models. Before training different models, since our data was imbalanced, we tried oversampling and under-sampling using SMOTE, which synthesises new minority instances to balance our data.

4.2. Dimensionality reduction and visualisation

Dimensionality reduction affected the performance of our model significantly. We had 14 features previously, which itself is quite feasible in terms of computation. Therefore, we decided not to go forward with the principle components and considered all the previous features instead.

4.3. Training models after feature selection

We did feature selection, using the correlation matrix. Considering the top-10 features that were having higher correlation to the target, and then training the same models and comparing the scores.

4.4. Clustering based idea for grouping and labelling the groups

There were many data points that were pointing to different labels, showing the obvious human behavior. To solve that problem we tried to experiment with the dataset using the basic idea of unsupervised learning. As in clusters we don't look at labels and then group the data points. Similarly, here we grouped the patients with the same features and then labelled them according to the majority label of that particular group. Again we trained our models and compared with the previously trained models.

4.5. Feature selection after Clustering

We did feature selection, using the correlation matrix. Considering the top-10 features that were having higher correlation to the target, and then training the same models and comparing the scores.

4.6. Optimisations

1. Stacking
It involves two levels, one level includes multiple baselines models and the second level has one meta learner. We trained multiple models including Naive bayes, Svm, XGBoost and random forest classifiers. And after getting predictions from all the models, we feed those predictions to meta-learner (logistic regression in our case), which in turn gives us final predictions. The basic difference is that instead of majority voting, we train a model in this layer to give final predictions.

2. Bagging
Here we used Bootstrapping as our bagging technique. We created 100 bootstraps samples and then trained 100 models on that. Then we took the majority voting for our final predictions. In Bagging, we are actually creating parallel weak learners and then taking summation over all models, this reduces the variance compared to the individual variance of the weak model.

3. Boosting
We used ADA-Boost as our boosting technique. In Bagging, we are actually creating parallel weak learners and then make final predictions, as variance reduces compared to different models.In Boosting, we actually learn sequentially, we train a model and then learn from the misclassification, we train our subsequent models by giving more weights to the records that were misclassified.

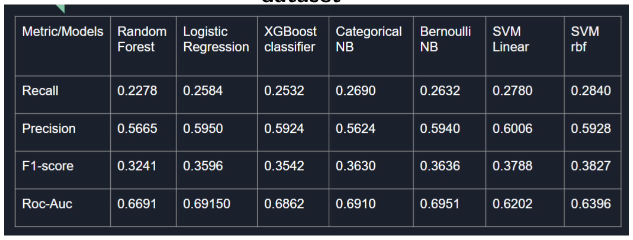
4.7. Stacking and Bagging

Generated 50 bootstrap sample. For each bootstrap sample, we trained multiple models including Naive bayes, Svm, XGBoost and random forest classifiers. Each of them gave their predictions. These predictions were fed to a logistic regressor, which in turn gave us the final predictions. We got 50 such predictions. Mean of these 50 predictions is the final predicted output of this Bagging and stacking model.

4.8. XGB Regressor

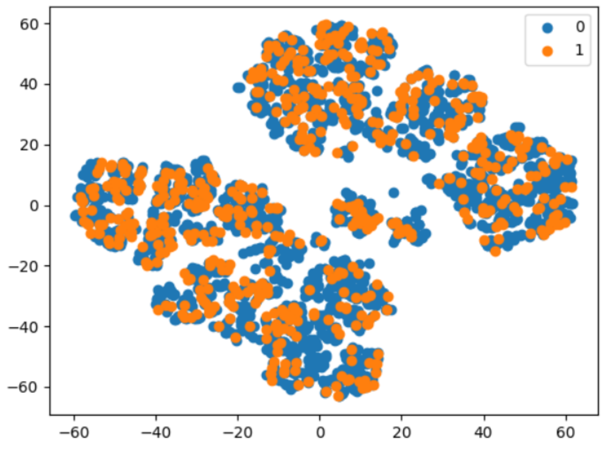
XGBoost regressor will optimise after each iteration for regression task, and at the last step we tuned different thresholds and reported the best accuracy. In XGboost classifier, it only optimises for the classification purpose, so there is slight difference, that's why it performed slightly better.

Figure 2. Scores for Different Models for original dataset



Metric/Models	Random Forest	Logistic Regression	XGBoost classifier	Categorical NB	Bernoulli NB	SVM Linear	SVM rbf
Recall	0.2278	0.2584	0.2532	0.2690	0.2632	0.2780	0.2840
Precision	0.5665	0.5950	0.5924	0.5624	0.5940	0.6006	0.5928
F1-score	0.3241	0.3596	0.3542	0.3630	0.3636	0.3788	0.3827
Roc-Auc	0.6691	0.69150	0.6862	0.6910	0.6951	0.6202	0.6396

Figure 3. t-SNE SVD



5. Results and Analysis

5.1. Baseline Models

Among different models, SVM and Naive Bayes performed better. As patients with similar comorbidities forms a group and are close to each other, so data is separable and SVM performs better. Naive Bayes performed better, as the features were mostly independent and even by intuition this problem seems to require conditional probability based solution. Shown in Fig 2.

5.2. Dimensionality reduction and visualisation

SVD Plot shown in figure 3.

PCA Plot shown in figure 4.
Dimensionality reduction affected the performance of our model significantly. We had 14 features previously, which itself is quite feasible in terms of computation. Therefore, we decided not to go forward with the principle components and considered all the previous features instead.

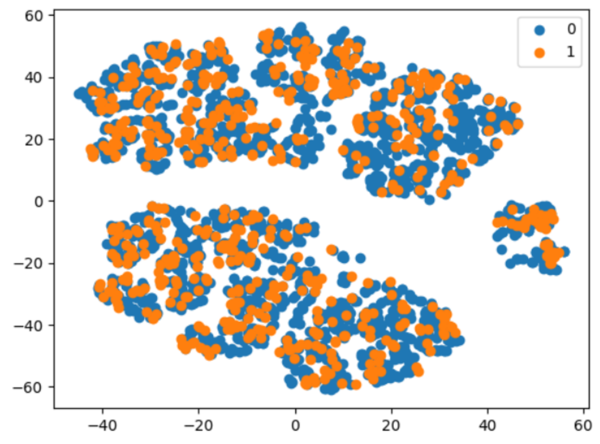


Figure 4. t-SNE PCA

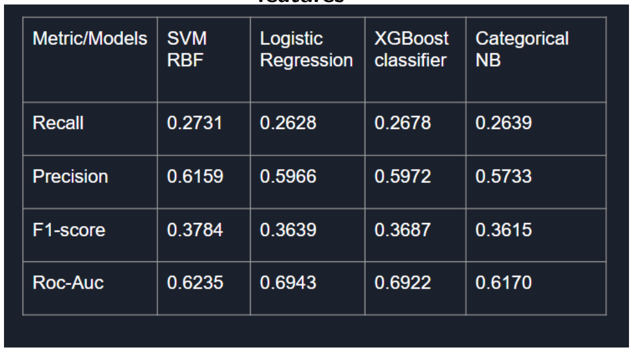


Figure 5. Scores for Different Models for best 10 features

Metric/Models	SVM RBF	Logistic Regression	XGBoost classifier	Categorical NB
Recall	0.2731	0.2628	0.2678	0.2639
Precision	0.6159	0.5966	0.5972	0.5733
F1-score	0.3784	0.3639	0.3687	0.3615
Roc-Auc	0.6235	0.6943	0.6922	0.6170

5.3. Training models after feature selection

After selecting top-k features, here we checked for k=10. The f1-scores didn't changed much, even reduced for some models. The reason being most of our features are independent, can be seen in the EDA. And each attributes have more or less similar contribution towards the target attributes, so we have to include all the features. Shown in Fig 5.

5.4. Clustering based idea for grouping and labelling the groups

The F1-score nearly doubled, precision remained nearly same but decent change in recall value was observed. Increase in recall means, we were reducing the False Negatives and increasing True Positives. Reason for drastic change, as same data points were pointing to different labels, so it was making data impure. After modifying the data, impurity was reduced, which even solved the class imbalanced problem. Shown in Fig 6.

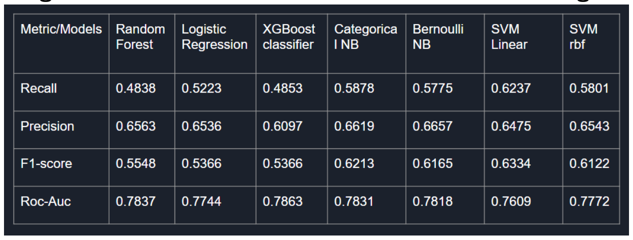


Figure 6. Scores for Different Models for clustering

Metric/Models	Random Forest	Logistic Regression	XGBoost classifier	Categorica l NB	Bernoulli NB	SVM Linear	SVM rbf
Recall	0.4838	0.5223	0.4853	0.5878	0.5775	0.6237	0.5801
Precision	0.6563	0.6536	0.6097	0.6619	0.6657	0.6475	0.6543
F1-score	0.5548	0.5366	0.5366	0.6213	0.6165	0.6334	0.6122
Roc-Auc	0.7837	0.7744	0.7863	0.7831	0.7818	0.7609	0.7772

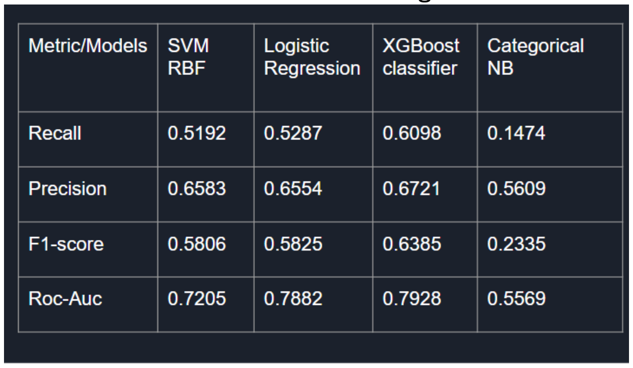


Figure 7. Scores for Different Models For best 10 features after clustering

Metric/Models	SVM RBF	Logistic Regression	XGBoost classifier	Categorical NB
Recall	0.5192	0.5287	0.6098	0.1474
Precision	0.6583	0.6554	0.6721	0.5609
F1-score	0.5806	0.5825	0.6385	0.2335
Roc-Auc	0.7205	0.7882	0.7928	0.5569

5.5. Feature selection after Clustering

After selecting top-k features, here we checked for k=10. The f1-scores didn't changed much, even reduced for some models. The reason being most of our features are independent, can be seen in the EDA. And each attributes have more or less similar contribution towards the target attributes, so we have to include all the features. Shown in Fig 7.

5.6. Optimisations

1. **Stacking** It improved the f1-score by 1-2 percent.

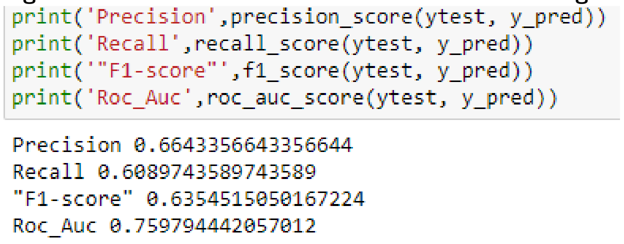
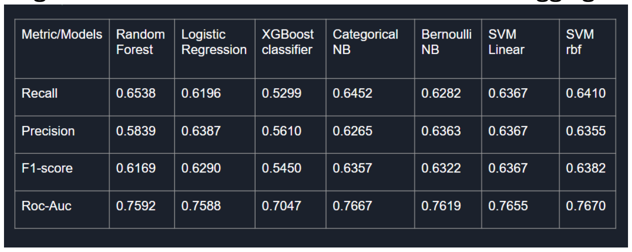


Figure 8. Scores for Different Models After Stacking

```
print('Precision',precision_score(ytest, y_pred))
print('Recall',recall_score(ytest, y_pred))
print('"F1-score"',f1_score(ytest, y_pred))
print('Roc_Auc',roc_auc_score(ytest, y_pred))
```

```
Precision 0.6643356643356644
Recall 0.6089743589743589
"F1-score" 0.6354515050167224
Roc_Auc 0.759794442057012
```

Figure 9. Scores for Different Models After Bagging



Metric/Models	Random Forest	Logistic Regression	XGBoost classifier	Categorical NB	Bernoulli NB	SVM Linear	SVM rbf
Recall	0.6538	0.6196	0.5299	0.6452	0.6282	0.6367	0.6410
Precision	0.5839	0.6387	0.5610	0.6265	0.6363	0.6367	0.6355
F1-score	0.6169	0.6290	0.5450	0.6357	0.6322	0.6367	0.6382
Roc-Auc	0.7592	0.7588	0.7047	0.7667	0.7619	0.7655	0.7670

Reason being, instead of using one model, we are considering predictions from multiple different models and then predicting. It basically provides us with the advantages of all the models. Shown in Fig 8.

2. Bagging Most of the models before bagging were giving f1-score around 58, but after bagging it is around 63. Reason being the main principle of bagging, we train different weak learners and then take voting over all models, in order to reduce the variance of final model, thus avoiding overfitting. Shown in Fig 9.

3. Boosting
F1-Score :- 0.6376
Recall :- 0.5940
Precision :- 0.688
Reason for Ada-Boost to perform better :- It performs better than an individual model, as after each iteration we learn from the wrongly classified records and try to classify it correctly in the further trees we create. So it helped in the increasing the f1-score. It's results were more or less comparable to the Bagging. Shown in Fig 10.

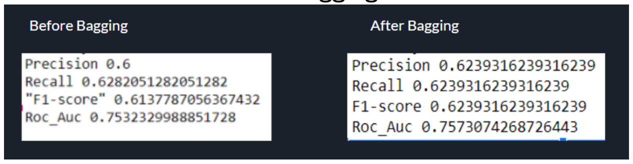
5.7. Stacking and Bagging

We got the following results
F1-Score :- 0.5327
Recall :- 0.4871
Precision :- 0.5876
ROC score :- 0.6939
Reason for bad performance : Because of multiple ML models with 50 bagging samples, the model became too complex and led to overfitting. Therefore, we decided not to go forward with these kind of combinations.

5.8. XGB Regressor

For threshold around 0.36, the precision and recall values were merging.And then using this threshold we converted our continuous values into the 0's and 1's. Result Scores are shown in Fig 11. Fig 12 shows score

Figure 10. Scores For XGB Regressor Before and After Bagging



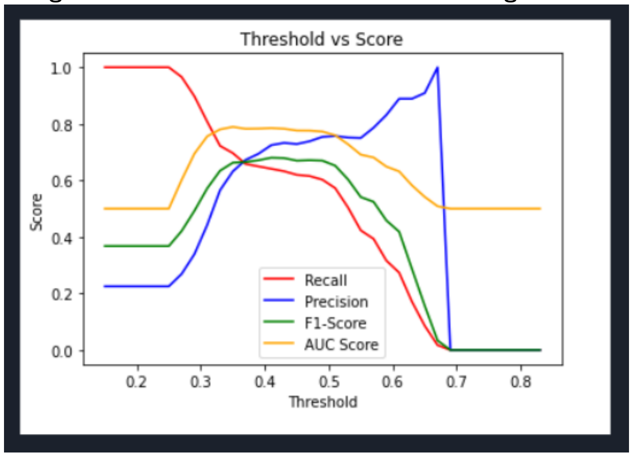
Before Bagging

```
Precision 0.6
Recall 0.6282051282051282
"F1-score" 0.6137787056367432
Roc_Auc 0.7532329988851728
```

After Bagging

```
Precision 0.6239316239316239
Recall 0.6239316239316239
F1-score 0.6239316239316239
Roc_Auc 0.7573074268726443
```

Figure 11. Score vs Threshold for XGB Regressor



vs different thresholds.

6. Conclusion

This machine learning model can prove to be very beneficial in early detection of the ICU patients on the basis of the comorbidities.It can help in the segregation of the patients early-on after which the doctors can use the live biological data for further ICU admissions. The main aim for early separation is for better ustilisations of the resources and only provide hospitalisation/ICU admissions to the needful patients and others can be kept in normal wards. Even the model can be used to give extra care to those patients that were predicted to go to the ICU in future, so that if possible we can contain them to normal wards only with little extra care, for better resource management.

7. Learning

1. The data that we got was imbalanced, so learnt about various data balancing techniques like SMOTE, Random Oversampling/Undersampling.
2. During data exploration and preprocessing, we learnt about the co-relation between all the features and analyzed them for feature selection.
3. Testing out different ML models, ensemble and boosting techniques and how they performed.
4. We also tested out regression models for classification

problem, using thresholding technique.
5. Most importantly, we learnt how different factors like pre-processing, model selection, parameter tuning, etc. collectively contribute to the efficient working of a mode.

8. Contribution

Bhavesh: Preprocessing, feature selection, model training, Stacking
Navam: Boosting, Preprocessing, feature selection, model training
Huzaifa: Bagging, Boosting, Visualization, model training

9. References

[1]B. Rai, A. Shukla, G. Choudhary and A. Singh, ”Pool Testing for COVID-19: Suitable Splitting Procedure and Pool Size for India”, Disaster Medicine and Public Health Preparedness, pp. 1-17, 2020.
[2]Y. Shang, T. Liu, Y. Wei, J. Li, L. Shao, M. Liu, Y. Zhang, Z. Zhao, H. Xu, Z. Peng, F. Zhou and X. Wang, ”Scoring Systems for Predicting Mortality for Severe Patients with COVID-19”, SSRN Electronic Journal, 2020.
[3] Dataset Available:
https://www.kaggle.com/lalish99/covid-19-in-mexico.