Death Prediction Model

Machine Learning

Problem Statement: -

We are provided with a dataset which contains an enormous number of anonymized patient-related information who were affected from covid including pre-conditions compiled from different medical sources from rural and urban areas.

The raw dataset consists of 22 unique features and about 1M unique patients. In the Boolean features, 1 means "yes" and 2 means "No". Values as 97, 98 and 99 are missing data.

We will train a Machine Learning model from the features provided and try to predict whether a patient will die or not under the given features.

The model helps to predict the probability of death of a patient which can actually help a medical centre to judge the condition of an incoming patient.

This will be important to understand and predict the type of care a COVID patient should be given when admitted to a medical centre.

Features :-

****id **: unique id for patient**

1. sex
2. age: age of the patient.
3. classification final: Covid test findings. Values 1-3 mean that the patient was diagnosed with covid in different degrees. 4 or higher means that the patient is not a carrier of covid or that the test is inconclusive.
4. patient type: type of care the patient received in the unit. 1 for returned home and 2 for hospitalization.
5. pneumonia: whether the patient already have air sacs inflammation or not.
6. pregnancy: whether the patient is pregnant or not.
7. diabetes: whether the patient has diabetes or not.
8. copd: Indicates whether the patient has Chronic obstructive pulmonary disease or not.
9. asthma: whether the patient has asthma or not.
10. inmsupr: whether the patient is immunosuppressed or not.
11. hypertension: whether the patient has hypertension or not.
12. cardiovascular: whether the patient has heart or blood vessels related disease.
13. renal chronic: whether the patient has chronic renal disease or not.
14. other disease: whether the patient has other disease or not.
15. obesity: whether the patient is obese or not.
16. tobacco: whether the patient is a tobacco user.
17. usmr: Indicates whether the patient treated medical units of the first, second or third level.
18. medical unit: type of institution of the National Health System that provided the care.
19. intubed: whether the patient was connected to the ventilator.
20. icu: Indicates whether the patient had been admitted to an Intensive Care Unit.
21. date died: If the patient died, 1 for not died and 0 for died

Feature Engineering: -

The input features are studied, modified and properly cleaned before feeding into the model to ensure proper training of the model and improve the accuracy of the model.

1. Every binary classification is classified as 1 or 0.

2. Features like ICU and INTUBED have a large no. of missing values. So, these features are decided to be dropped.

3. Extremely unrelated features like sex and pregnancy with very low values of correlation are decided to be dropped to reduce computation cost and time. This will definitely not affect the accuracy of the model.

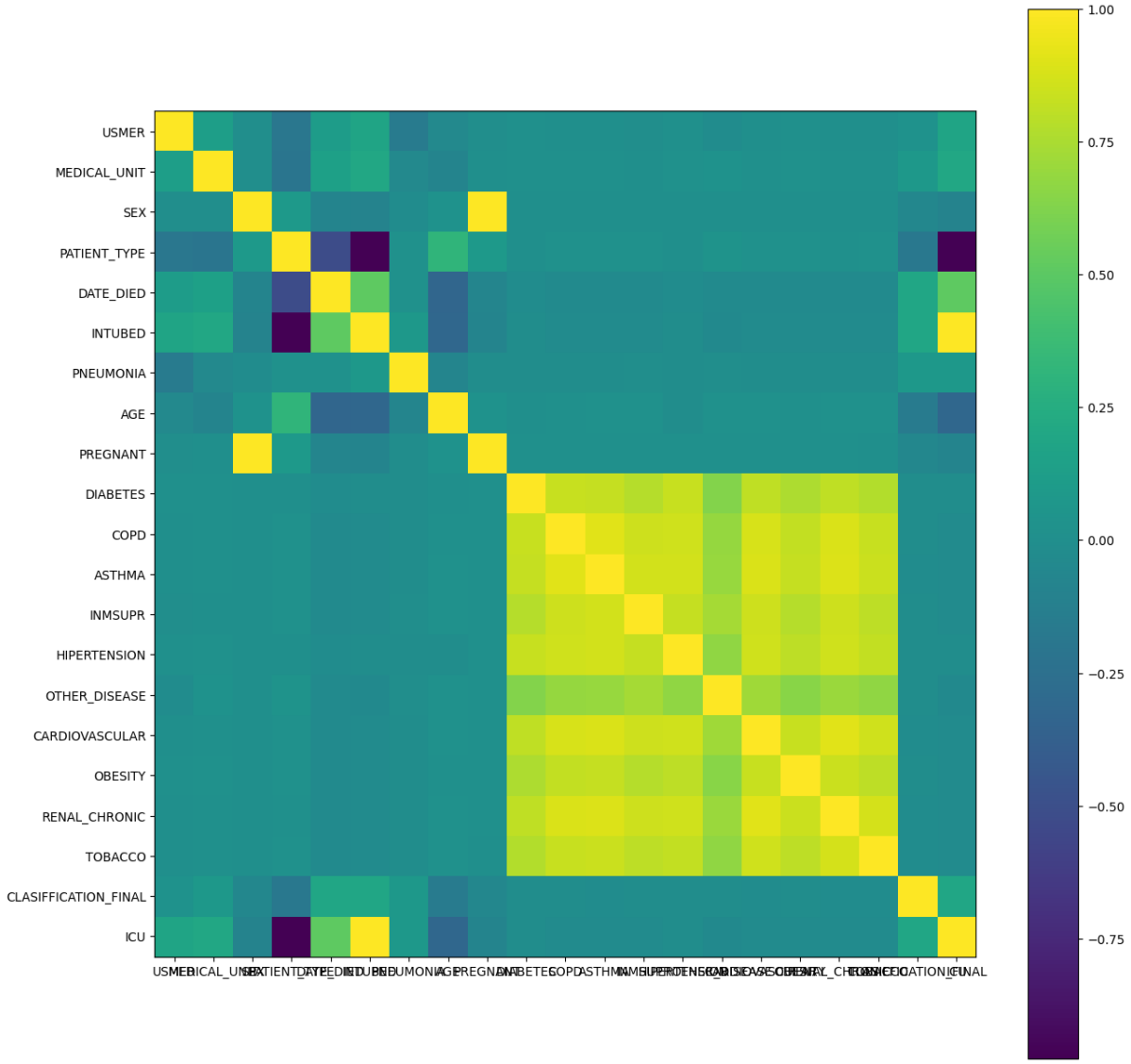
4. Highly related features with fairly high correlation among them are clubbed into single features.

5. A new feature named DISEASE is introduced for each patient which is the total no. of confirmed diseases a patient is affected with. This feature has quite high mutual correlation with DATE_DIED.

6. Missing values are replaced with “No Disease”.

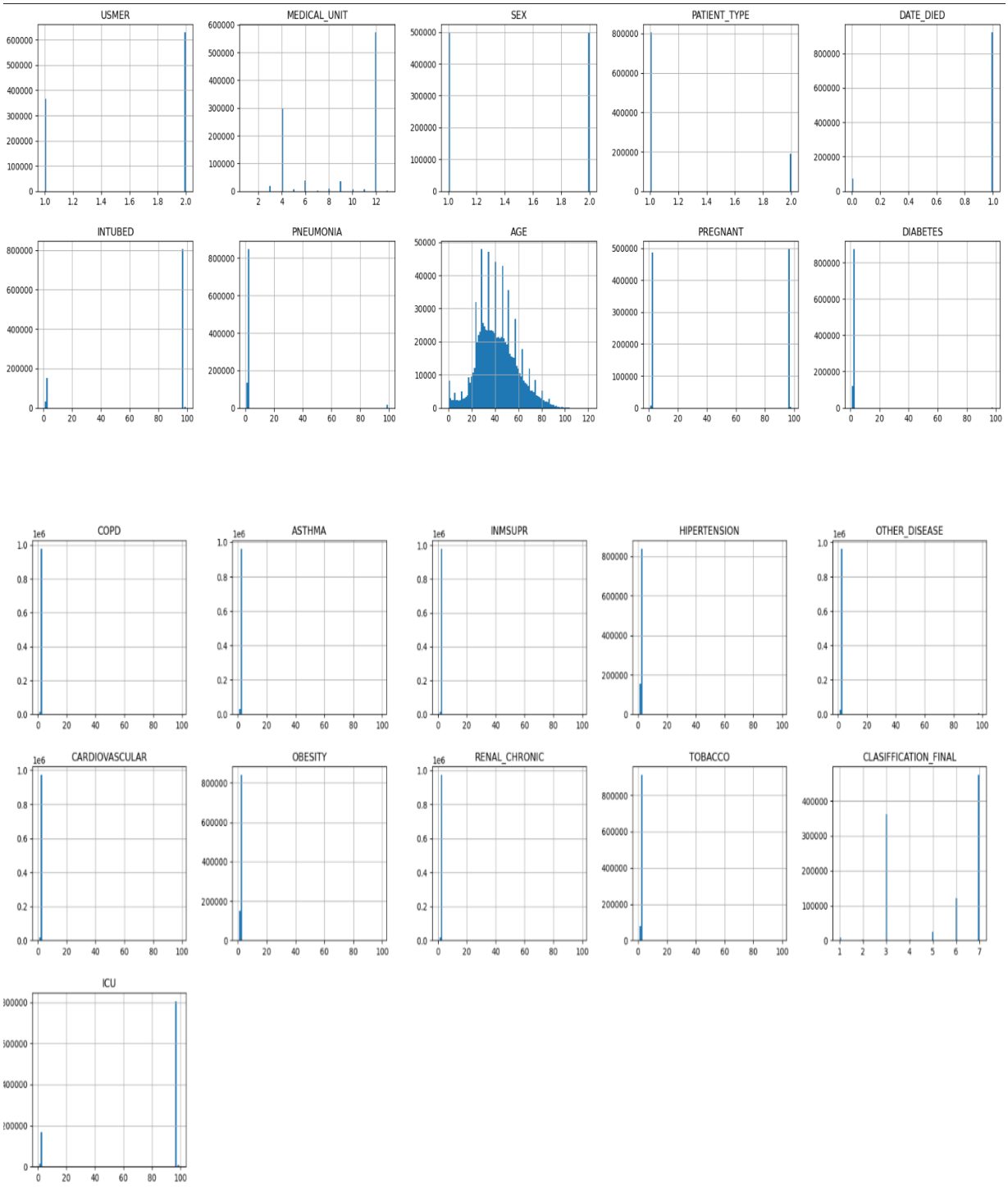
Data Visualisations: -

The following is the heatmap which shows the correlation b/w every pair of features from which we can actually derive some very important conclusions.



The bright square shows the high correlation between the diseases. So, the entire set of diseases can be clubbed with a smaller set of features to ensure lower computation time and cost.

The following is the histogram of all the features which help to visualise the distribution of the values of every feature in the training dataset.



The following is a row of the heatmap where we have extracted all the features and found its correlation with the model. Features with a correlation of > 0.1 are significant in training the model.

	DATE_DIED
USMER	0.112698
MEDICAL_UNIT	0.149542
SEX	0.081310
PATIENT_TYPE	0.516095
DATE_DIED	1.000000
INTUBED	0.495259
PNEUMONIA	0.469393
AGE	0.319741
PREGNANT	0.078820
DIABETES	0.215950
COPD	0.089438
ASTHMA	0.018338
INMSUPR	0.049493
HIPERTENSION	0.203300
OTHER_DISEASE	0.062877
CARDIOVASCULAR	0.075868
OBESITY	0.056280
RENAL_CHRONIC	0.119246
TOBACCO	0.005332
CLASIFFICATION_FINAL	0.188858
ICU	0.498906
DISEASE	0.226227

We will train the model using the below 3 models: -

1. Logistic Regression
2. Decision tree
3. Random forest

Then, we will use a stacking classifier using the above 3 models and finally predict our classification output.

1. Logistic Regression Performance

The logistic regression model performs well when DATE_DIED = 1 but its f1-score is a bit low.

```
from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression(max_iter=10000)
logmodel.fit(X_train,y_train)
predictions = logmodel.predict(X_test)
print(classification_report(y_test,predictions))
```

✓ 15.1s

	precision	recall	f1-score	support
0	0.60	0.42	0.49	14739
1	0.95	0.98	0.97	184491
accuracy			0.94	199230
macro avg	0.78	0.70	0.73	199230
weighted avg	0.93	0.94	0.93	199230

2. Decision Tree Performance

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=15)
rfc.fit(X_train, y_train)
predictions4 = rfc.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,predictions4))
print(confusion_matrix(y_test,predictions4))
```

✓ 9.1s

```
              precision    recall  f1-score   support

           0       0.57      0.45      0.50     14739
           1       0.96      0.97      0.96    184491

    accuracy                           0.93    199230
   macro avg       0.76      0.71      0.73    199230
weighted avg       0.93      0.93      0.93    199230

[[  6659   8080]
 [  5083 179408]]
```

3. Random Forest Classifier

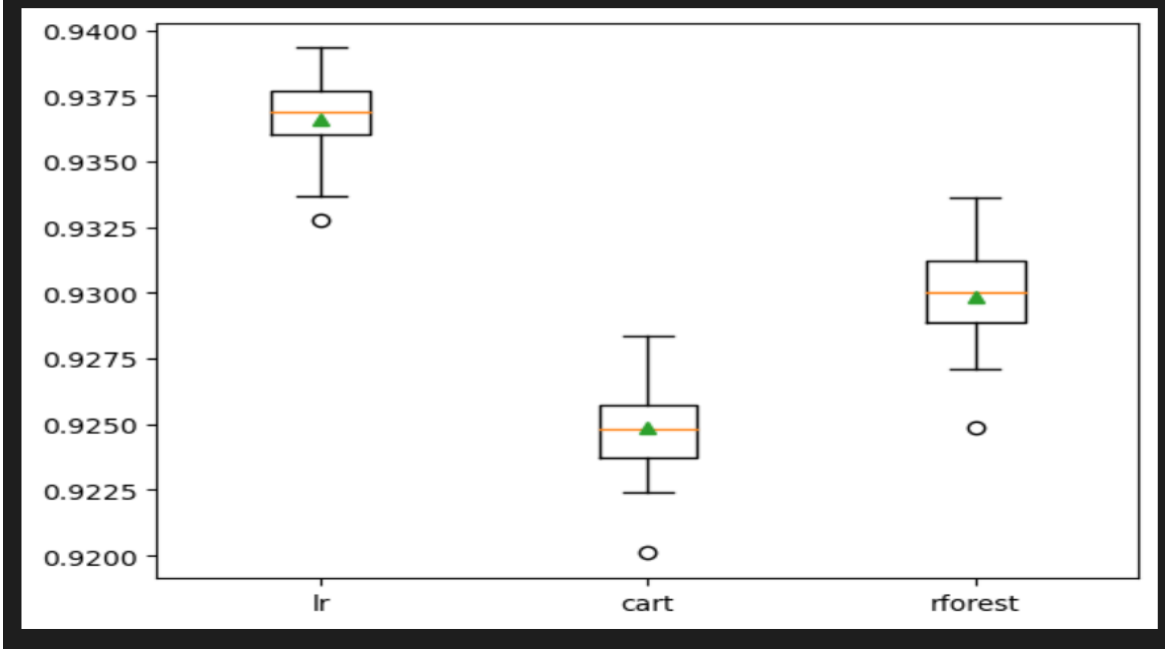
```
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier()
dtree.fit(X_train,y_train)
predictions = dtree.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,predictions))
print(confusion_matrix(y_test,predictions))
```

✓ 1.8s

```
              precision    recall  f1-score   support

           0       0.53      0.48      0.51     14739
           1       0.96      0.97      0.96    184491

    accuracy                           0.93    199230
   macro avg       0.75      0.73      0.74    199230
weighted avg       0.93      0.93      0.93    199230

[[  7145   7594]
 [  6212 178279]]
```

Performance Comparison: -

```
>lr 0.937 (0.002)
>cart 0.925 (0.002)
>rforest 0.930 (0.002)
```



Stacking Classifier Performance

	precision	recall	f1-score	support
0	0.62	0.41	0.50	14739
1	0.95	0.98	0.97	184491
accuracy			0.94	199230
macro avg	0.79	0.70	0.73	199230
weighted avg	0.93	0.94	0.93	199230

Conclusion: -

The model has an overall accuracy of 94% on new test cases. The model performs very well on cases where the person didn't actually die. This is indicated by the high value of f1-score when the output is actually 1.

On the other hand, the model's performance is quite poor when the person actually died. Very few numbers of dead case reports in the dataset (case of undersampling) are accountable for the fact.

*** Thank You ***