

# Covid-19 Prediction Using Machine Learning Algorithms

Machine Learning Project Report  
April 8th, 2022

## TABLE OF CONTENT

<b>S. No</b>	<b>TOPICS</b>	<b>Page No.</b>
1	INTRODUCTION	03
2	DATASET	03
3	EXPLORATORY DATA ANALYSIS	04
4	DATA PREPROCESSING	06
4	METHODOLOGY	08
5	MODEL EVALUATION AND OPTIMIZATION	08
6	FEATURE SELECTION	11
7	MODEL IMPLEMENTATION	12
8	CONCLUSION AND LIMITATIONS	16

# INTRODUCTION

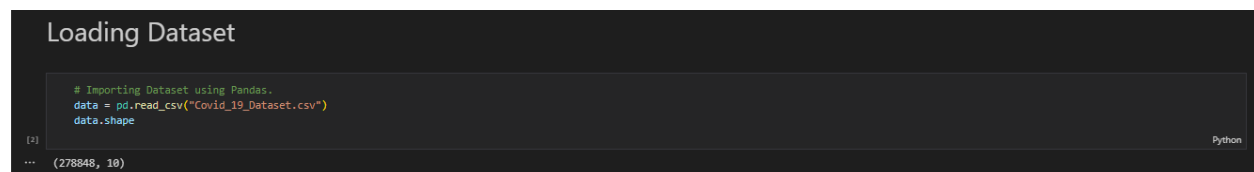
Millions of people have been impacted by the COVID-19 epidemic, which has significantly increased demand for testing and diagnosis. In this situation, machine learning has shown potential as a technique for estimating the probability that a patient will have COVID-19. Machine learning algorithms can identify patterns that are indicative of COVID-19 infection through analyzing a variety of factors including symptoms, demography, and medical history. The goal of this research is to create a machine learning model that can correctly predict a patient's COVID-19 status based on that person's specific characteristics. This model may help with early identification and resource allocation, ultimately assisting in preventing the spread of infection.

This project is a binary class classification problem where our target variable “corona result” is having only two classes which are negative and positive. For the accurate prediction of our target variable, we are using different classifiers including decision tree, random forest, naïve bayes classifier, logistic regression, boosting models etc. First, we are performing data preprocessing such as cleaning etc. and undertaking exploratory data analysis (EDA) to gain useful insights into our data.

Our goal is to build a robust and accurate prediction model that can help healthcare institutions identify patients who are at risk of having covid-19. So, for that reason, we are using different optimization techniques such as K fold cross validation to improve accuracy of the model and to validate our models and ensure they are not overfitting the data and to get best performing features we are selecting features with correlation matrix feature selection and selecting best possible features.

## DATASET: -

The dataset we are using in this project is from an online data repository. The dataset contains 2.7 million patient records and 10 features. We are using “Corona Result” as our target variable. It is a Binary class classification problem, where the value of class is Negative and Positive. The data is slightly imbalanced where negative values are more as compared to the positive cases and consists of 9 attributes other than target attribute.



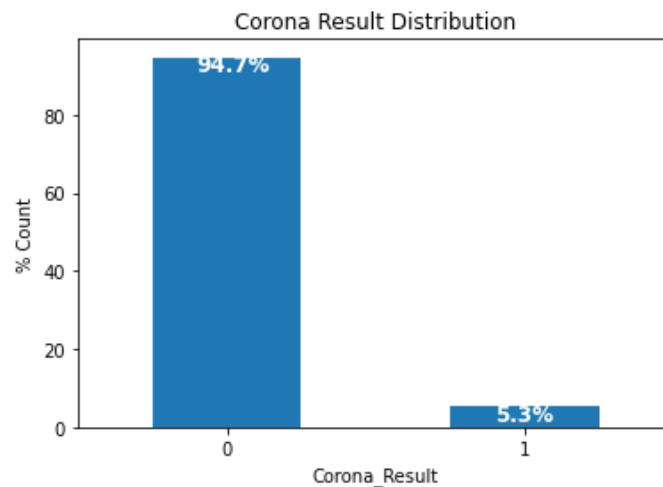
```
Loading Dataset

# Importing Dataset using Pandas.
data = pd.read_csv("Covid_19_Dataset.csv")
data.shape

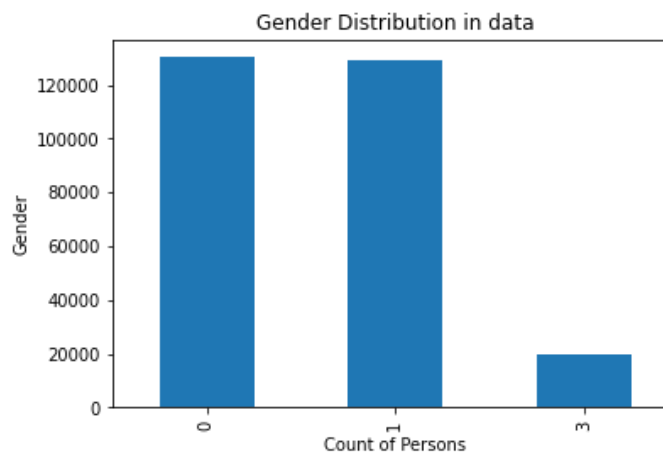
[1]
... (278848, 10) Python
```

## EXPLORATORY DATA ANALYSIS

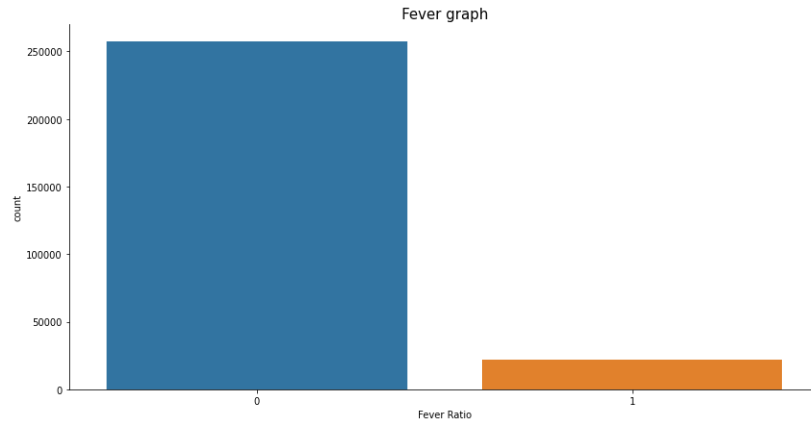
In this project, before performing data preprocessing, we are conducting EDA to get deep insights from our data. We are using different visualizations libraries such seaborn, matplotlib, and plotly etc. for this purpose.



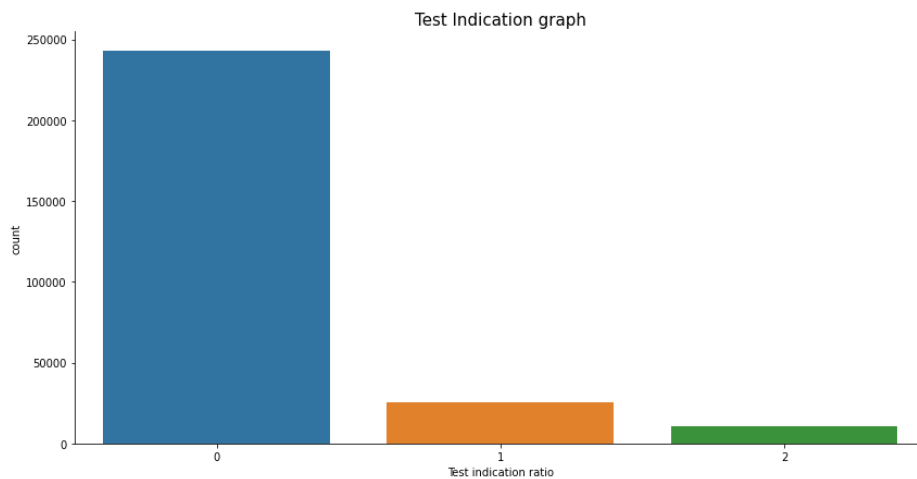
In this first plot, we have plotted a distribution of our target variable. We can see that distribution of class label is imbalanced in our dataset. There are 94.7% records of negative cases and only 5.3% records of positive cases. We have 2 classes so it's a binary class classification task.



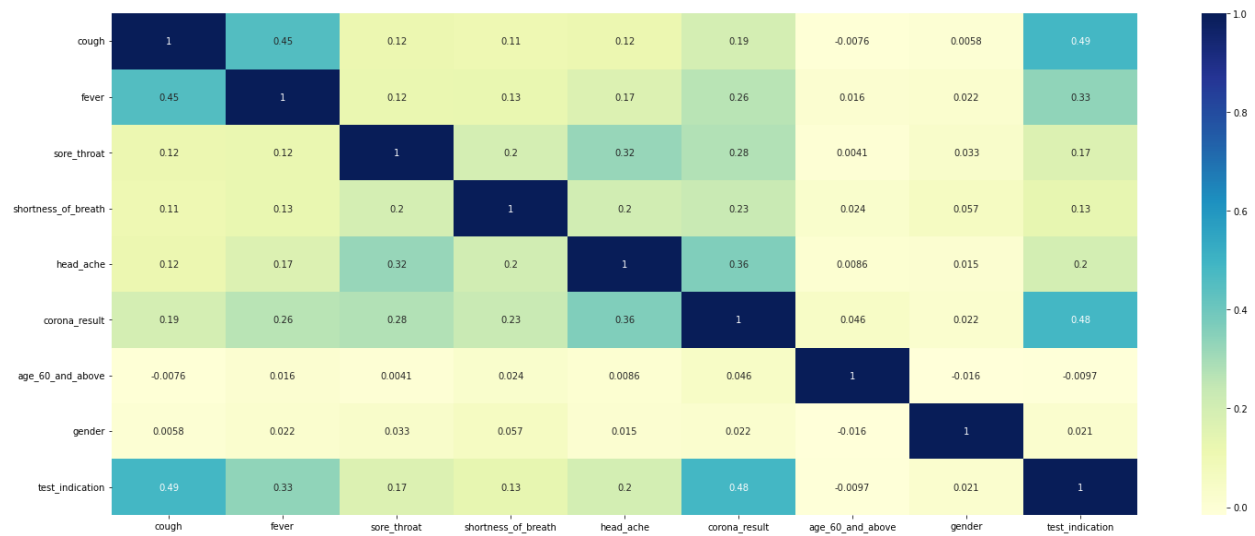
In the second plot, we are plotting distribution of gender in the dataset (Here 1 = male, 2 = female, 3 =others ). We can observe clearly from the bar chart that female are equal as compare to the males in the data. So, our dataset is balanced in terms of gender.



Some patients are having fever in our dataset. These patients are at high risk of covid-19. But most of the people do not have any fever.



We can observe from the bar chart above, most of the people are having source of virus from unconfirmed source, and very few of them have virus source from abroad and some are having virus from confirmed contact with other patient. (0: others, 1: Abroad, 2: confirm contact)



Here in the correlation plot we can observe that headache, test indication, soar throat, and fever are having strong positive correlation with the target variable. Means if the patient is having these indications, then it is more likely that he is having covid-19. Multicollinearity is also present in the data as we can see the blue block in matrix.

## DATA PREPROCESSING

Before using the data in machine learning algorithms, preparing the data is a crucial step. Data preprocessing is a data mining technique used to turn raw data into a format that is both practical and effective.

For our dataset, we are also performing data preprocessing. The steps we performed for preprocessing of data are below:

### 1- Data Cleaning

Data cleaning is the process of eliminating or changing data that is inaccurate, incomplete, irrelevant, redundant, or incorrectly formatted to prepare it for analysis. When it comes to data analysis, this data is usually not necessary or beneficial because it can slow down the process or produce false results. Faulty data will badly impact our result, so we must remove it.

#### a) Treating Null Values

So now after importing our dataset, we now check for null values using python panda's library and find that various features in our dataset are containing null values.

```
# Checking null values in all features.
data.isnull().sum()

[19]
... test_date      0
     cough        0
     fever        0
     sore_throat   0
     shortness_of_breath 0
     head_ache     0
     corona_result  0
     age_60_and_above 0
     gender        0
     test_indication 0
     dtype: int64
```

We can see that in our dataset we do not have any null values.

## b) Label Encoding

Most of the columns in our dataset are in integer format, but two features country and status are in categorical object format. To apply machine learning models on the data we first need to convert these features into the numerical format. For that we will perform label encoding on the data.

```
data['corona_result'].unique()

[23]
... array(['negative', 'positive', 'other'], dtype=object)

data['corona_result'] = data['corona_result'].map({'negative': 0, 'positive': 1, 'other': 2})
data['corona_result'] = pd.to_numeric(data['corona_result'])

[24]
```

## TRAIN TEST SPLIT

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model. Here we are splitting our dataset into train dataset and test dataset with the ratio of 70-30% where 70% is our train dataset and 30% is our test dataset.

```
from sklearn.model_selection import train_test_split
trainX, testX, trainy, testy = train_test_split(X, y, test_size=0.3, random_state=2)
print(trainX.shape)
print(trainy.shape)
print(testX.shape)
print(testy.shape)

[25]
... (195193, 8)
     (195193, 1)
     (83655, 8)
     (83655, 1)
```

## METHODOLOGY

In this project, we use various machine learning algorithms to predict the target variable corona result and compare the outcomes of each approach to see which model fits and performs best on the training data set. To boost the model's performance, we are also using various model enhancement strategies. These methodologies include voting classifiers, K-fold cross validation and feature selection for best features. Our only goal is to increase the model's performance with highest accuracy.

We first implement machine learning models on the full data with all the features and then apply feature selection techniques to select top performing features and to reduce dimensionality of the data.

### Model Evaluation: -

We are evaluating the performance of our model's using accuracy, AUC graph, precision score, sensitivity, specificity, miss rate, precision, and recall values. Before we move forward to model implementation, it is necessary to know the following terms.

#### 1) Precision: -

The proportion of True Positives to All Positives is known as precision. That would be the percentage of cases we correctly identify as having an illness out of all the cases having it for our issue statement.

#### 2) Recall: -

The recall serves as a gauge of how well our model detects True Positives. Recall therefore indicates the proportion of cases with disease that we correctly identified.

#### 3) Sensitivity: -

The sensitivity of a machine learning model indicates how effectively it can recognize successful examples. Other names for it include the true positive rate (TPR) or recall. Sensitivity is used to evaluate model performance since it allows us to ascertain how many examples the model was able to correctly identify.



#### 4) Specificity: -

Specificity is the ratio of original negatives that are correctly predicted class by the models.

#### 5) Miss Rate: -

The false negative rate – also called the miss rate – is the probability that a true positive will be missed by the test. It's calculated as  $FN / (FN + TP)$ , where FN is the number of false negatives and TP is the number of true positives (FN+TP being the total number of positives).

#### 6) False Positive Rate: -

The false positive rate is calculated as  $FP / (FP + TN)$ , where FP is the number of false positives and TN is the number of true negatives (FP+TN being the total number of negatives). It's the probability that a false alarm will be raised: that a positive result will be given when the true value is negative.

#### 7) Confusion Matrix: -

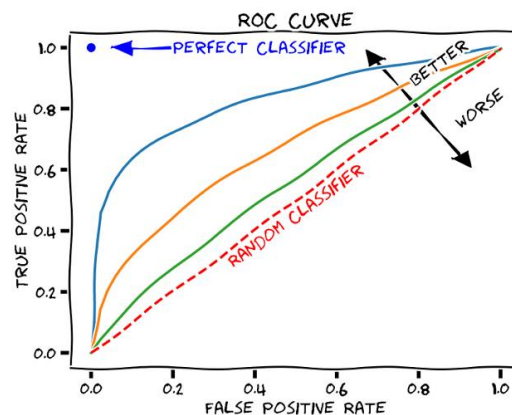
A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix.

		True Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

## 8) ROC – AUC Curve: -

A measurement tool for binary classification challenges is the Receiver Operator Characteristic (ROC) curve. In essence, it separates the "signal" from the "noise" by plotting the TPR against the FPR at different threshold values. The capacity of a classifier to differentiate between classes is measured by the Area Under the Curve (AUC), which is used as a summary of the ROC curve.

The model performs better at differentiating between the positive and negative classes the higher the AUC. The classifier can accurately discriminate between all Positive and Negative class points when  $AUC = 1$ . The classifier would be predicting all Negatives as Positives and all Positives as Negatives, however, if the AUC had been 0.



## K fold Cross Validation

A machine learning model's performance is assessed using the K-fold cross-validation technique, which divides the dataset into K subsets (or folds) of about similar size. Each fold is utilized as the test set once and the remaining K-1 folds are used as the training set as the model is trained and evaluated K times.

Because it offers a more precise assessment of model performance than a single train-test split, K-fold cross-validation is a frequently used technique for model evaluation. K-fold cross-validation lowers the chance of overfitting and offers a more accurate prediction of how the model will perform on brand-new, untested data by employing numerous train-test splits.

In our project, we are using K fold cross validation with 10 folds. It tells about how our model will perform on the actual test data.

```
# K - FOLD Cross Validation.

kfold = KFold(n_splits=5, random_state=100, shuffle=True)

DT = DecisionTreeClassifier(max_depth=200)
results_kfold = cross_val_score(DT, trainX, trainy.values.ravel(), cv=10)
print(results_kfold)
print("Accuracy of DECISION TREE Kfold with k=10: %.2f%%" % (results_kfold.mean()*100.0))

pred = cross_val_predict(DT, trainX, trainy.values.ravel(), cv=5)
conf_mat = confusion_matrix(trainy, pred)
print(conf_mat)
disp = ConfusionMatrixDisplay.from_predictions(trainy, pred)
```

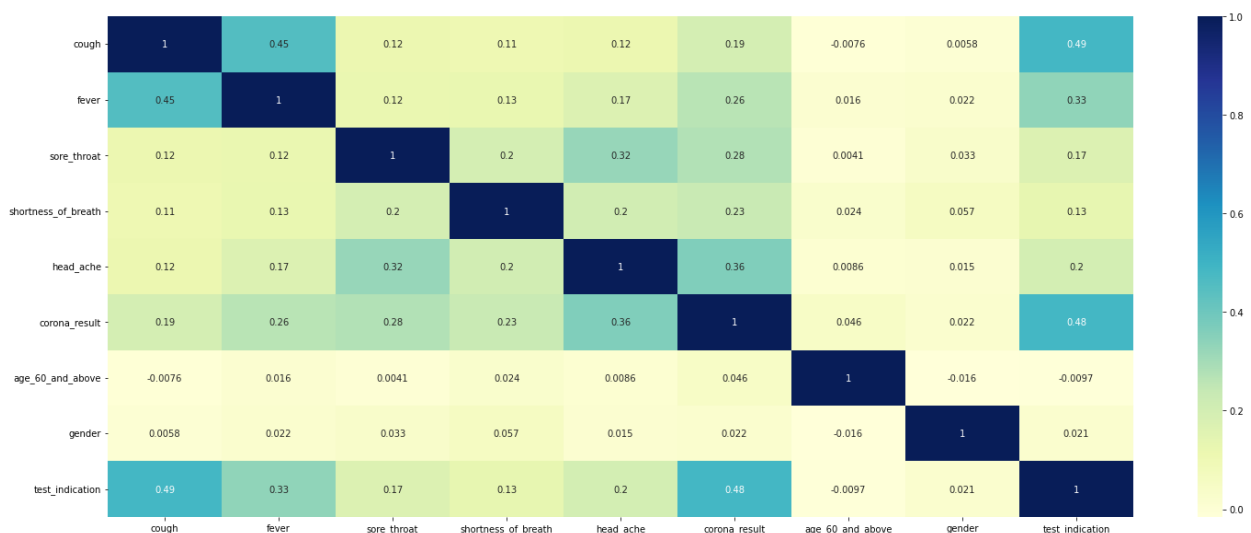
Python

## Feature Selection Using Correlation matrix.

Feature selection using a correlation matrix involves identifying pairs of features in a dataset that are highly correlated with each other and selecting a subset of features based on their correlation with the target variable.

A correlation matrix is a table that shows the correlation coefficients between all pairs of variables in a dataset. The correlation coefficient measures the strength of the linear relationship between two variables, ranging from -1 (perfect negative correlation) to 1 (perfect positive correlation).

We first determine the correlation matrix for each feature in the dataset before doing feature selection using it. Afterwards, we search for feature pairings that are highly associated with one another (e.g., correlation coefficient greater than a threshold value such as 0.8). We choose the feature that is most strongly positively correlated with the target variable out of each pair of highly correlated characteristics and remove the other one. By doing so, we can simplify the dataset's dimensions and get rid of extraneous data.



## Model Implementation: -

First, we are implementing different machine learning classifier on our dataset first without any feature selection and then after doing selection with methods of correlation matrix feature selection. We have 10 attributes at first, and our data is 70-30% divided into train and test sets.

### 1- Decision Tree

The first model we implemented is the decision tree classifier. A decision tree is a supervised machine learning model that is used for classification and regression tasks. It is a tree-like model that consists of nodes and branches, where each node represents a feature or attribute, and each branch represents a decision rule or condition based on that feature or attribute.

In a classification task, the goal of a decision tree is to divide the dataset into smaller and more homogeneous groups, where each group corresponds to a particular class or category. The decision tree gives us accuracy of 96.7% on our full feature dataset, but its accuracy decreases to 96% after reducing dimension and using only top performing features.

### 2- Random Forest Classifier

The second model we are implementing after decision tree is random forest classifier. A Random Forest Classifier is a supervised machine learning algorithm that uses an ensemble of decision trees to perform classification tasks. The random forest algorithm is an extension of the decision tree algorithm, which overcomes some of the limitations of decision trees such as overfitting and instability.

On our dataset, random forest is performing well, giving accuracy of 96.75% accuracy on full feature dataset and 96.66% accuracy after feature selection.

### 3- Gradient Boosting Classifier

One of the famous boosting algorithms, which combines different weak learners into strong by minimizing the loss function. It is an ensemble technique. The goal of gradient boosting classifier is to iteratively add new models to the ensemble that correct the errors of the previous models.

In our project Gradient boosting is performing well on both full feature and selected feature dataset. It is getting the highest accuracy of 96.7% after feature selection.

#### 4- XGBoost Classifier

A well-known machine learning technique that is a member of the gradient boosting family is called XGBoost (Extreme Gradient Boosting) Classifier. It is a development of the conventional gradient boosting approach, which minimizes a loss function to combine several weak models into a strong one.

High performance and accuracy are hallmarks of XGBoost, particularly when applied to structured data issues like feature engineering and tabular data. In comparison to other gradient boosting methods, XGBoost key benefit is its scalability, which enables it to handle huge datasets with millions of rows and columns.

In our project Xtreme Gradient boosting is performing better than all the other models, maybe because of the reason that XGBoost can capture non-linear relationships between the features and the target variable. It is getting the highest accuracy of 96.76% before feature selection.

#### 5- Naïve Bayes Classifier

Naive Bayes is a simple yet effective machine learning algorithm for classification tasks. It is based on the Bayes theorem and assumes that the features are conditionally independent given the class variable. This assumption is known as the naive assumption, which makes the algorithm computationally efficient and easy to implement.

For our dataset, naïve bayes is not performing very well and its accuracy was below expectations.

#### 6- Logistic Regression

An analysis of relationships between a binary dependent variable and one or more independent variables is done statistically using logistic regression.

In our project, LR is working fine, and its accuracy was 96% with full feature dataset.

# Voting Classifier (Ensemble Model)

Voting classifier is basically an ensemble technique which unites different base classifiers and based on the result of these classifiers it computes new result. It is a technique that may be used to improve model performance, ideally achieving better performance than any single model used in the ensemble. We are using a soft voting technique here which computes probability based on probabilities and weights associated with different classifiers. In voting classifier, we are using our best models.

We are using 4 models which are random forest models, Naïve Bayes, gradient boosting model and XGBOOST which were giving us best results. So, after combining all our best performing models we create a voting classifier model, which gives us the best result better than all other base models.


```
Voting Classifier (Ensemble Models) - Best Performing Model

r0 = GradientBoostingClassifier(learning_rate = 0.01,max_depth=5,n_estimators=400)
r1 = RandomForestClassifier(max_depth=10,n_estimators = 1200)
r2 = XGBClassifier(learning_rate = 0.85, max_depth = 2)
r3 = GaussianNB()

# In soft voting, every individual classifier provides a probability value that a specific data point belongs to a particular target class
voting = VotingClassifier([('gb1', r0), ('gb2', r1), ('gb3', r2), ('gb4', r3)], voting='soft')
voting.fit(trainX,trainy.values.ravel())
y_pred_voting = voting.predict(testX)
```

# STREAMLIT WEB APPLICATION

After implementing different machine learning models, we finally got our best performing model which is a voting classifier so now we are saving our model with the help of pickle and deploying our model on a web app created on streamlet. This web app will take different inputs which are features of our data set and predict whether the patient is having COVID-19 or not based on the provided information.

 **COVID 19 Predictor**

**Covid 19 Detection App**

Cough = No : 0, Yes : 1

0 - +

Fever = No : 0, Yes : 1

0 - +

Sour Throat = No : 0, Yes : 1

0 - +

ShortNess of Breath = No : 0, Yes : 1

0 - +

head\_ache = No : 0, Yes : 1

0 - +

Age 60 Above = No : 0, Yes : 1

0 - +

Gender = Female : 0, Male : 1

0 - +

Test Indication

0 - +

Predict

Covid is not Detected : Negative

## CONCLUSION

We have implemented Naive Bayes as well but the performance of Naive bayes was below average as compared to the other model maybe this is because of imbalances ness of data or its feature independence assumption.

One benefit of Naive Bayes is that it's not much time complex like other tree models. Naive Bayes is known for its fast training and prediction times compared to other machine learning algorithms. The linear time complexity of Naive Bayes makes it suitable for large datasets with high-dimensional feature spaces. However, the accuracy of the model may be affected if the independence assumption does not hold or if there are strong correlations between features.

After that our decision tree model is performing well on the specific low depths but after increasing its depth the model performance decreases. When a decision tree is trained on a dataset, it may learn the specific patterns and noise in the data rather than the underlying relationships. This can result in a tree that is too complex and overfits the data. When the depth of the tree is increased beyond a certain point, the overfitting becomes more severe, and the accuracy decreases. To reduce the effect of overfitting, random forest model is introduced which is a bagging model and do sampling with replacement to reduce overfitting effect.

Time complexity of decision tree and other tree base models depends upon the number of records and feature. If the dataset has too many features and the number of depth is also high then computation power of the model will also be high.

After performing a deep analysis on the dataset, we have successfully built a voting classifier model that has proven to be the best in predicting the outcome of our target variable. Through rigorous testing and evaluation, we have determined that the voting classifier model provides the most accurate and reliable predictions, surpassing all other models that we have explored.

Our best model (Voting classifier) is getting highest accuracy.

## LIMITATION

Although the techniques used in this project helped to improve the performance of the models, there are some limitations that should be considered.

First, the quality of the results depends on the quality of the data. If the data is imbalanced, or biased, the performance of the models may be limited.



For this project only six algorithms were used, there might be other supervised algorithms such as KNN, SVM etc. that can also be used to check if they are performing better or low. Other than that, hyper parameter tuning of models can also be used to increase efficiency.

For future work, we can also perform feature scaling on the data and then perform machine learning and maybe performance of the models increase after that.

Another limitation is, instead of splitting the dataset, a different dataset could be used to evaluate the model performance.