## Name - Pushkar Ashok Narkehde

## Roll - 2203528 | MITU20BTCSD018

## Class - TY CSE IS3 batch B

## Assignment 3

**Aim -**

- Download the any dataset from UCI or Data.org or from any other data repositories and Implement Single and multilayer perceptron on a dataset.

**Objective -**

1. To learn about classification and regression
2. To learn MLP and backpropagation
3. To demonstrate and analyse the results

**Theory -**

**Regression -**

```
1. Regression is a type of problem in supervised machine learning where a
machine tries to predict continuous values from the set of input data
2. In Regression we have given with a data which helps to machine
learning algorithm to find hidden pattern from data and learn from that
pattern in term of mathematical formulation or model parameter.
3. for Example, if we have given with the employee data such as their
work experience, designation and many more and we have to comes up with
what salary to be given to that employee or freshers who willings to join
the company, in this case machine learning model predicting a continuous
single value which is stated as salary in as per our example.
```

**Regression metrices -**

```
1. When we perform regression algorithm then we need some mathematical
functions or metrices which measure the loss, accuracy so that model can
optimise and generalise itself.
2. For regression, we majorly focuses on Mean Squared Error which measure
the loss or error between the actual output and the predicted model
outpur.
3. when the data is imabalanced, we focuses on Adjusted Mean squared
error which tries to reduce the noise of the data and then calculate the
loss.
```

**Classification -**

```
1. Classification is a supervised machine learning type in which a
machine learning algorithm tries to classify the data in either binary or
```

multinomial groups.
2. As classification is supervised learning algorithm hence there is a labbeld data or target data where a algorithm tries to learn and reduce the loss to make correct classification in unseen data.
3. fro Example, if we have given with covid 19 symptons and tell our model to classifiy if the person is suffering from covid +ve or not in that case our model perform binary classification as it is only tries to classifiy in either yes or no values, in this case the model predict the probability of covid +ve as well as covid -ve class and whichever probability is high is stated as result.
4. in classification problem we have a threshold values which applies on the predicted probability to which helps to change output of model by setting the threshold in between 0.1-0.9 in case we found less loss other than default threshold.

**Classification metrices -**

1. When we perform classification algorithm then we need some mathematical functions or metrices which measure the positive predicted rate, accuracy, as well as loss so that model can optimise and generalise itself.
2. For classification, we majorly focuses on Precision, recall and in case of imabalnced data we focus on F1 score which is best metrics for imabalnced data.
3. also we have ROC-AUC curve which tells how much the model confident and accuracte to classify the data.

**MLPClassifier (Multilayer Perceprton) Basic Intution -**

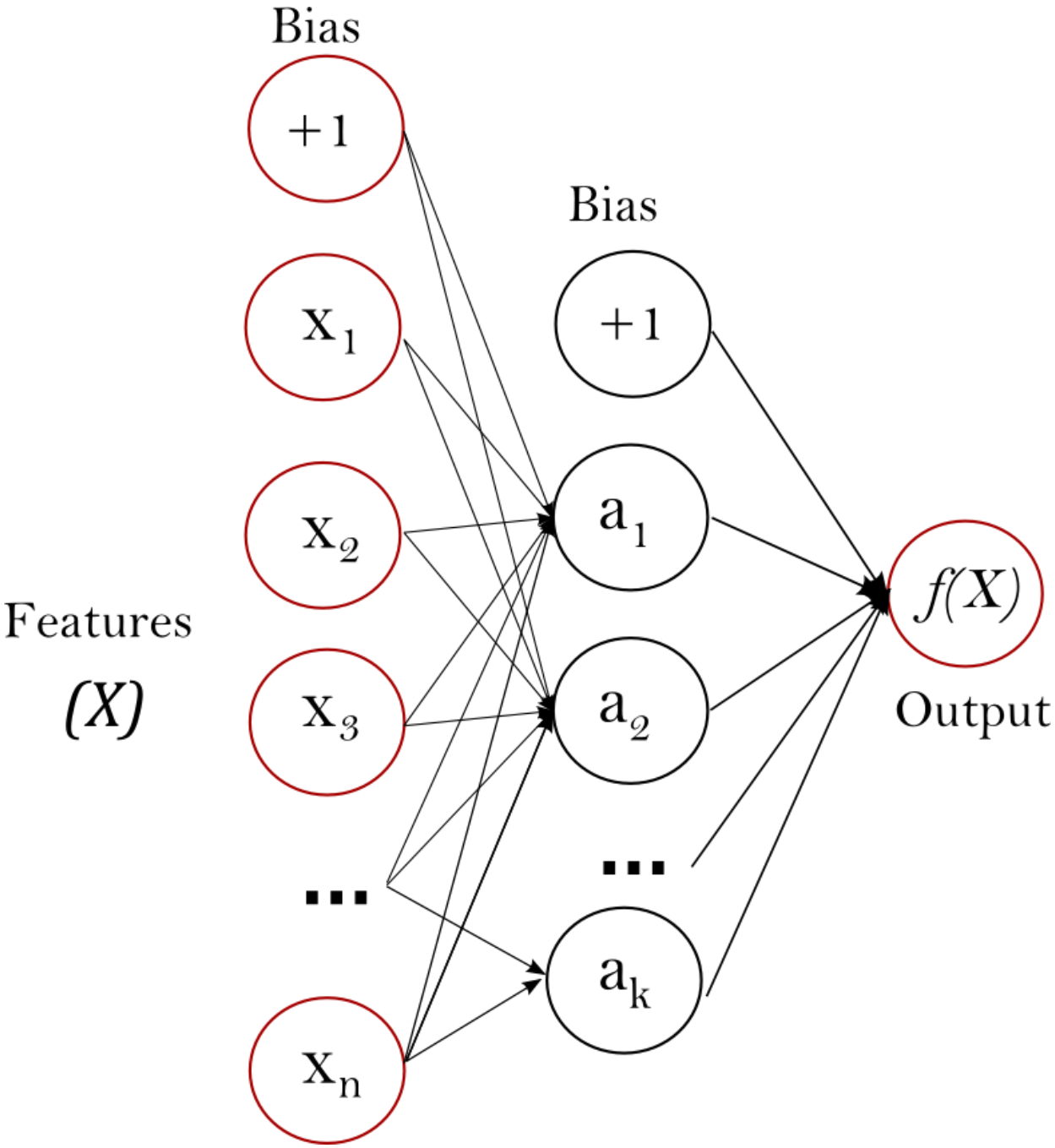
1. MLP is a multiple layes perceptron networks which tries to classify the linear data also it is capabale of predicting the continuous value in case of regression problem.
2. A perceptron is a network like the human brain which consists of neurons, layers, activation function, let's talk about these units in details.
    1. Neurons - It is somewhat like a biological neuron in a brain which accept the input and pass forwards, but actaully in MLP a neurons is somewhat which takes input from user and gives output to user. The primary purpose of neuron is to pass the data to next neurons which the helps of connection and when the data is passed from one neuron to other some operation are perfromed so that the next layers will get the best value rather the origina value.
    2. Layers - Layers playes important role in MLP, layers are nothing but the collection of neurons which helps to go deeper so that the model can extract the important characteristcs from the input data. There are basically 3 layers: Input layer, Hidden Layer, Output Layer.
        2.1 Input Layer - this layer is first layer in MLP and it's work is to accept the input from the user. The size of input layer is same as the size of data i.e dimesnions of data for ex. if a data consist of 120 features then the size of input layer is 120 neurons.
        2.2 Hidden layers - This is a hyperparameter, i.e a parameter

which can be change by a programmer. In MLP Hidden layers playes important roles as it tries to extract as more as information it can. There is no limit for using the np of hidden layer and neurons in hidden leyers.

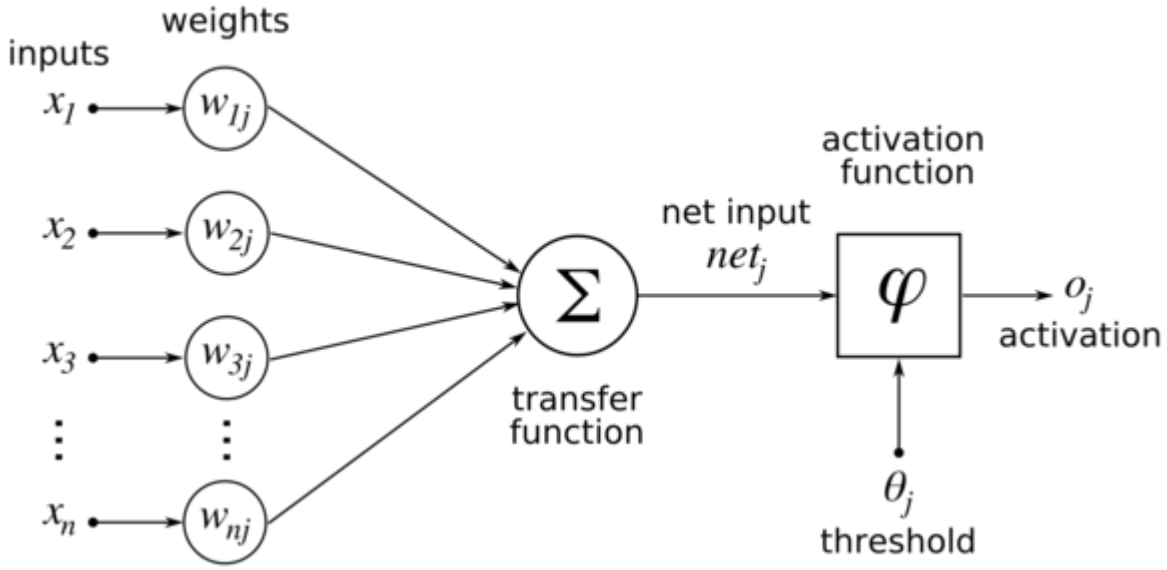
2.3 Activation function - It is a mathematical function which actually fire the neuorns, i.e it tell the network which neurnon is to fire as an output. It plays the major role in predicting or classifing the data. Usually we have many AF and the first one is step function which makes 0 if the value is -ve and 1 if value is greter than 1, which is not usually good, hence to avid this we comes up with differnt AF like sigmoid, tanh, Relu, LeakyRely, Softmax, which are used for differnt purposes as per their need.

**MLP Working -**

1. As i stated earlier the MLP has multiple layers where ther input is propagated to next layer, now earlier i said it never send the original ip to next layer each time, it uses some mathematical formulation so that the model can find the complex relationship between the data and reduce the loss.
2. When a single neurons progpagate it output to next layer's neurons the output is always get multilied by some weight metrics and a bias is added. now the question comes in mind why we do this an the reason is if there is high error we always try to reduce and in case of reducing the error we change the model paramter, the same intution works here in case we got more error then as a result we can't change the input to chane or reduce the error hence we used weight metrics which got updated in every time.
3. (input * weight + Bias) this calculation happens every time when a neuron pass data to next neuron and a bias is added, now if we observe this carefully we get to know that this is the equation of line i.e (y = mx + c) and that's right.
4. once the output of neuron is passed to next neuron the same operation is performed again but before that we use AF to change the output of the current neurons to reduce the error or loss.
5. after all iteration the error is generated by calculating the predicted values i.e probabilit and the actual data
( error = sqrt( origian - prediction)^2 ). if the error is hight then we updates the weights in such a way that for next iteration the error is less as compared to previous.




**MLP Networks with bias shown and input to understant how data passed.**

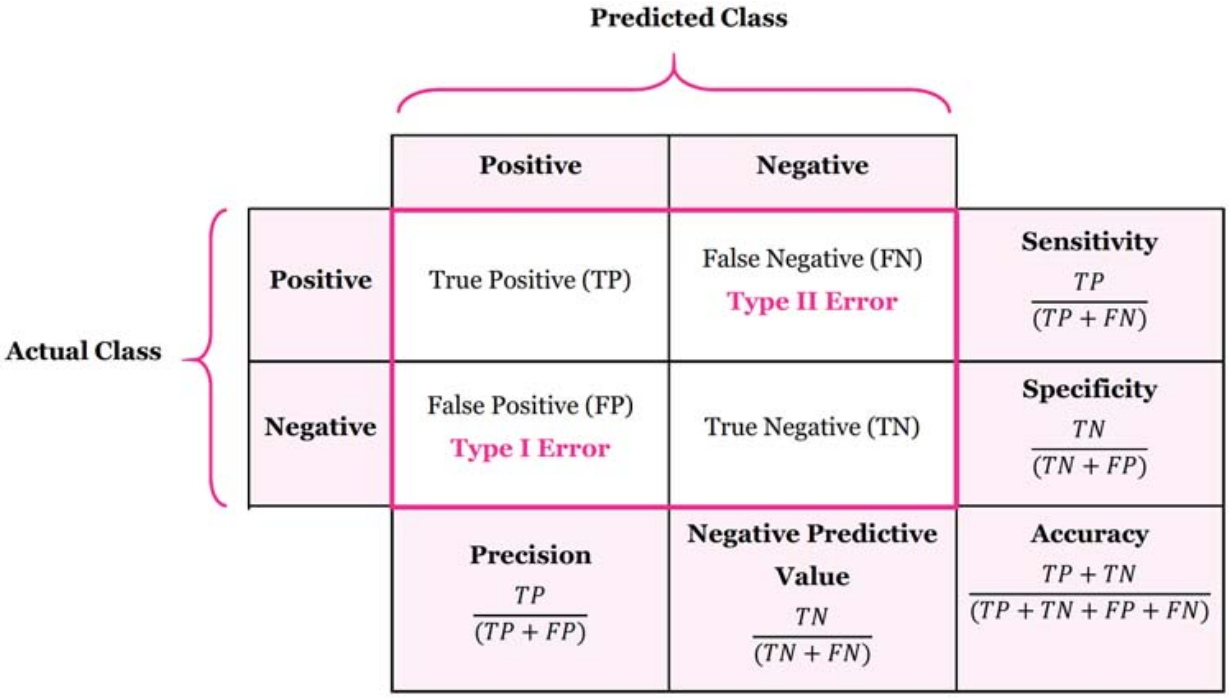



**MLP Networks with weights shown and input to understant how weights are initilized and how AF is used.**

**Metrices -**

**Confusion matrix -**

1. This matrix shows that how much data is classified as positive and negative as well as it also indicates that how that positive and negatve classification done w.r.t positive and negative classe, i.e. True positive, False Positive, True Negative, False NEgative
2. True Positive - It is noting but how much positive we are predeicted that are acutually positive. i.e actaul also +ve and predited also +ve
3. False Positive - It shows that how much positive we are predicted that are actually false i.e. actaul is -ve but predicted as +ve
4. True Negative - It sows that how much negative we are predicted that are actaully negaive i.e actual is -ve and predicted is also -ve.
5. False Negative - It shows that how much negative we are predicted that are actaually positive i.e actual is +ve but predicted as -ve.
6. By using confusion matrix we can comes up with precision, recall, accuracy, f1-score etc.




Accuracy -

1. It is nothing but how much correct we are predicte from the total data i.e TP+TN / TP+TN+FP+FN
2. it is a measuer which shows that how much confident our algorihtm on the data in prediction

Precision -

1. Precison is nothing but a positive predicted rate i.e TP / TP+FP
2. for example assume that we have True positive as 4 and TP+FP as 4+1 then our Precision is 4/4+1 i.e. 4/5
3. in any casey your TPR should be as possible as more.

Recall -

1. Recall score is measure of TP / TP+FN i.e ratio of TP and TP+FN which should be high as possible as it can.
2. Recall is also called as sebsitivity which is calculated on total datapoints.

F1-score -

1. F1-Score is measure in case of data imabalnce and it is best fit between Precision and Recall.
2. F1-Score is 2*precision + recall / Precision * recall.
3. F1 score shouuld be as possible as high as it is combination of precision and recall

# Code

- Problem Statement - Prediction of Covid disease with the helps of medical attributes.
- Dataset - Covid dataset from Kaggle.
- Dataset Description -

1. test_date ==> Date for covid 19 test
2. cught ==> is the patience having cough or not (binary variable yes=1, no=0)
3. fever ==> is the patience having fever or not (binary variable yes=1, no=0)
4. sore_throat ==> is the patience having sore_throat or not (binary variable yes=1, no=0)
5. shortness_of_breadth ==> is the patience having shortness_of_breadth or not (binary variable yes=1, no=0)
6. head_ache ==> is the patience having head_ache or not (binary variable yes=1, no=0)
7. corona_result ==> is the patience having corona_result or not (categorical variable -ve for covid,+ve for covid, other=> no confirmation with covid or not) *self encoded* **(1= +ve, 0= -ve)**
8. gender ==> is the patience having gender or not (categorical variable male,female) *self encoded* **(1= male, 0=female)**
9. test_indication ==> is the patience having gender or not (categorical variable contact_with_confirm=> covid due to contact of covid 19 person, abord=> covid in other country, other=>no idea about covid infection)

In [23]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")

from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier, MLPRegressor
from sklearn.ensemble import BaggingClassifier,ExtraTreesClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix,roc_auc_score,classification_report,precis
```

In [2]:
```
df = pd.read_csv('corona_tested_individuals_ver_0083.english.csv')
df.shape
```

Out[2]: (2742596, 10)

In [3]:
```
df.head()
```

Out[3]:

| | test_date | cough | fever | sore_throat | shortness_of_breath | head_ache | corona_result | age_60_and_abov |
|---|---|---|---|---|---|---|---|---|
| **0** | 2020-11-12 | 0 | 0 | 0 | 0 | 0 | negative | N |

| | test_date | cough | fever | sore_throat | shortness_of_breath | head_ache | corona_result | age_60_and_abov |
|---|---|---|---|---|---|---|---|---|
| **1** | 2020-11-12 | 0 | 1 | 0 | 0 | 0 | negative | N |
| **2** | 2020-11-12 | 0 | 0 | 0 | 0 | 0 | negative | Ye |
| **3** | 2020-11-12 | 0 | 0 | 0 | 0 | 0 | negative | N |
| **4** | 2020-11-12 | 0 | 1 | 0 | 0 | 0 | negative | N |

- Here we do not need the test_date features as it is not valid characteristics to contributes towards covid 19 prediction.
- also test_indication feature indicates that how the person affected by covid.
- the feature age_60_and_above will only work if the person age is more than 60 which is also not sufficient to contribute in covid19, still we use ha feature.
- also the data is already binary encoded for some features lets first encode the feature understand the data.

In [4]:
```
df.drop(['test_date', 'test_indication'],axis=1,inplace=True)
df.shape
```

Out[4]:
```
(2742596, 8)
```

In [5]:
```
pd.set_option('display.float_format', lambda x: '%.3f' % x)
```

In [6]:
```
df.describe()
```

Out[6]:

| | cough | fever | sore_throat | shortness_of_breath | head_ache |
|---|---|---|---|---|---|
| **count** | 2742596.000 | 2742596.000 | 2742596.000 | 2742596.000 | 2742596.000 |
| **mean** | 0.041 | 0.035 | 0.011 | 0.004 | 0.022 |
| **std** | 0.197 | 0.185 | 0.104 | 0.063 | 0.146 |
| **min** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| **25%** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| **50%** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| **75%** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| **max** | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |

- as our data is encoded with binary values hence we can't infer more information from describe statistics.

- Now, we are checking for null values in the data so that we can either impute them or drop them.

In [7]:
```
df.isna().sum()
```

Out[7]:
```
cough                       0
fever                       0
sore_throat                 0
shortness_of_breath         0
head_ache                   0
corona_result               0
age_60_and_above       547644
gender                  92886
dtype: int64
```

In [8]:
```
null_age = df.age_60_and_above.isnull().sum()
null_gender = df.gender.isnull().sum()
total_null = null_age+null_gender
print('Total data in dataset : ' , df.shape[0])
print('Total null values in dataset : ' , total_null)

print('Total data remains if we remove all null values = ', df.shape[0]-total_null)
print('{} % of null data is present in the dataset.'.format(total_null * 100 / df.shape
```

```
Total data in dataset :  2742596
Total null values in dataset :  640530
Total data remains if we remove all null values =  2102066
23.354879829183737 % of null data is present in the dataset.
```

- Here we can seee that some of our features having huge amount of null values and as we have huge amount of data we can drop that null values so that we are left with original data rather than any miss imputation.
- also our feature **age_60_and_above** have more null values hence dropping that column is always an option for us because we have seen befor the type of data in thet particular feature.
- fill null vlaues may lead to improper prediction and it will impact more.
- so as we have too much data it is ok if we drop that null values. except the null values are more than 5%.

In [9]:
```
df.drop(['age_60_and_above'],axis=1,inplace=True)
```

In [10]:
```
df.dropna(inplace=True)
```

In [11]:
```
df.shape
```

Out[11]:
```
(2649710, 7)
```

- Now we are left with the useful data.
- still as we have huge data but it is vertically huge hence we have to check for the data duplication also.
- if there is too much duplicate data we have to remove that data.
- second we have to focus on target class distribution.

In [12]:
```
df.corona_result.value_counts()
```

Out[12]:
```
negative    2390508
positive     219681
other         39521
Name: corona_result, dtype: int64
```

- here we have 3 classes where the class **other** is not useful for us. lets remove the data having result other.

In [13]:
```
df.drop(df[df['corona_result']=='other'].index,inplace=True)
df.corona_result.value_counts()
```

Out[13]:
```
negative    2390508
positive     219681
Name: corona_result, dtype: int64
```

- Now we are left with proper data of target column.
- we just need to encode some data and we can go further for inferenceing the knowledge.

In [14]:
```
binary = {'positive': 1, 'negative': 0}
df['corona_result'] = df['corona_result'].map(binary)
```

In [15]:
```
binary_gender = {'male': 1, 'female': 0}
df['gender'] = df['gender'].map(binary_gender)
```

In [16]:
```
df.head()
```

Out[16]:

| | cough | fever | sore_throat | shortness_of_breath | head_ache | corona_result | gender |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| **1** | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| **2** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| **4** | 0 | 1 | 0 | 0 | 0 | 0 | 1 |

- now, we are have all encoded data, but by observing the target column i get to know that there

may be data imbalanced and data duplication.

- let's try to find the imabalance ratio.

In [17]:
```
covid_pos_rate = np.sum(df.corona_result) / len(df.corona_result) *100
print('Covid +ve rate - ',covid_pos_rate)
```

```
Covid +ve rate -  8.416287096451637
```

- oops, we have imbalance data in our hand.
- let's first check for duplicate data.

### Issue -

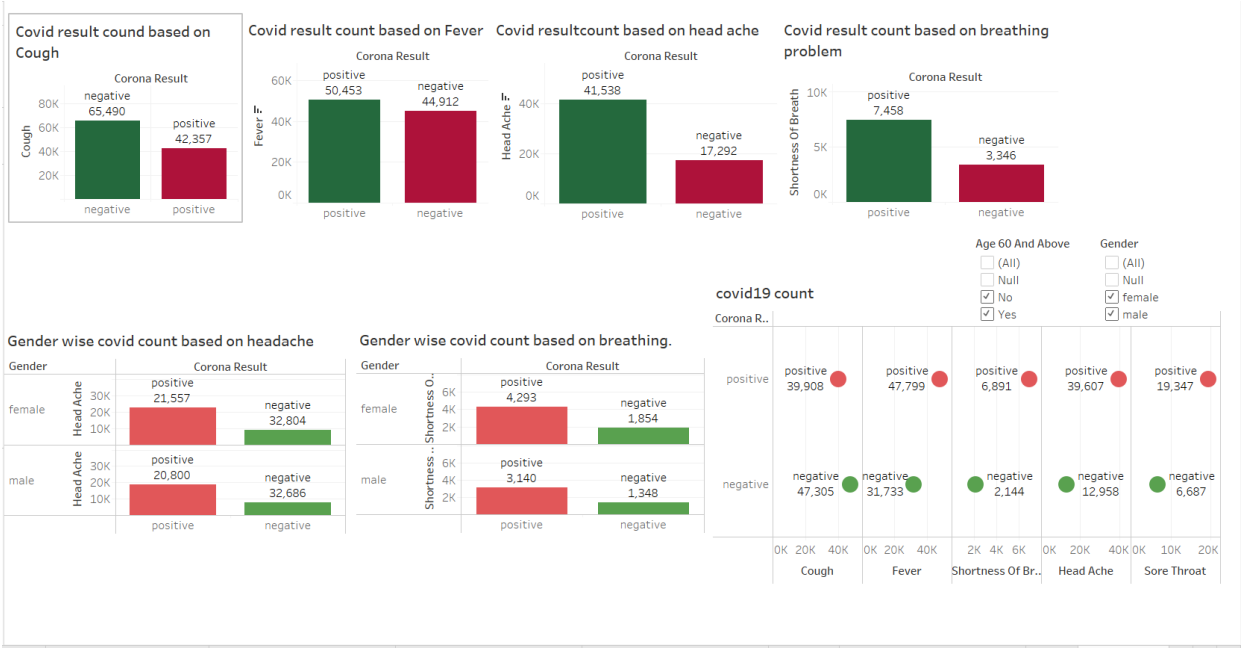
1. The dataset in our hand is Imbalanced dataset.
2. The given dataset contains only 9.70 % of covid +ve data which is not sufficient for building accurate model.
3. if we use these data then our model is highly accurate for covid -ve but not for covid +ve which is wrost condition ever.
4. hence to maintain sensitivity, specipicity, and precision we need a balance dataset.
5. to balance a dataset we are going to use either undersampling of covid -ve or oversampling of covid +ve.
6. if the data is duplicate then we have to drop dupplicated data.

In [18]:
```
print('DataFrame shape is : ', df.shape[0])
print('Total duplicate rows : ', df.duplicated().sum())
```

```
DataFrame shape is :  2610189
Total duplicate rows :  2610061
```

- here we can observe that we have around 99.99% of duplicated data which is not good for training the model.
- also, we have imabalnced data which may lead to one side classification.
- let's take some sample from both class and make new datafram from them to work with models.

## data analysis on Tableau dashboard (due to huge data)




In [19]:
```
df.columns
```

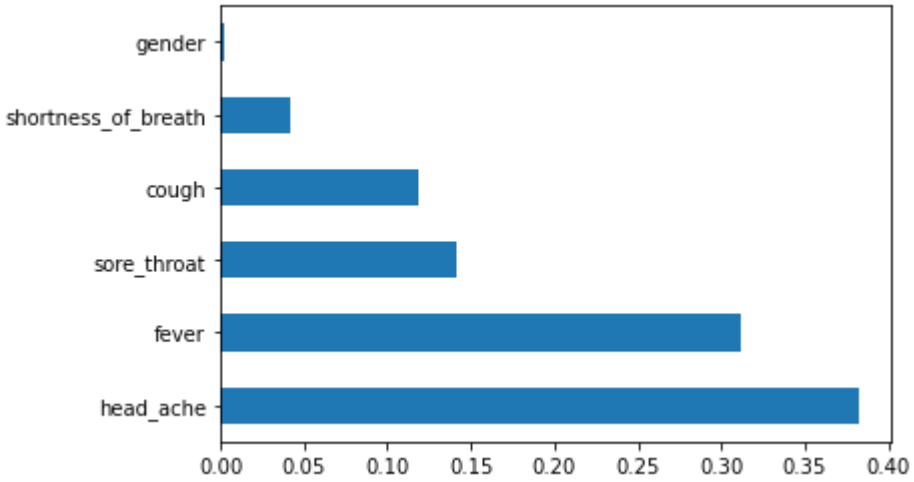
Out[19]:
```
Index(['cough', 'fever', 'sore_throat', 'shortness_of_breath', 'head_ache',
       'corona_result', 'gender'],
      dtype='object')
```

- Let's first check for fearure importance to work with.

In [20]:
```
X = df.drop('corona_result', axis=1 )
y = df['corona_result']
```

In [25]:
```
Xt = X
yt = y
model = ExtraTreesClassifier()
model.fit(Xt,yt)
print(model.feature_importances_)
feat_importances = pd.Series(model.feature_importances_, index=Xt.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.show()
```

```
[0.11900226 0.31227104 0.14140028 0.04245954 0.38244299 0.00242389]
```




- in the above plot we can clearly observe that the gender feature is not contributing much w.r.t other features.
- hence dropping that feature or building model with that feature will not affect largly on accuracy, but still we have to deal with duplicate data

In [26]:
```
df.drop(['gender'],axis=1,inplace=True)
```

- Lets seperate the data for covid 19 positive and negative samples

In [27]:
```
df.columns
```

Out[27]:
```
Index(['cough', 'fever', 'sore_throat', 'shortness_of_breath', 'head_ache',
       'corona_result'],
      dtype='object')
```

In [28]:
```
covid_zero = df[(df.cough==0) & (df.fever==0) & (df.sore_throat==0)& (df.shortness_of_b
           & (df.head_ache==0) & (df.corona_result==0)]
covid_one = df[(df.cough==1) & (df.fever==1) & (df.sore_throat==1)& (df.shortness_of_br
           & (df.head_ache==1) & (df.corona_result==1)]
```

In [29]:
```
df.drop(df[(df.cough==0) & (df.fever==0) & (df.sore_throat==0)& (df.shortness_of_breath
           & (df.head_ache==0) & (df.corona_result==0)].index,inplace=True)
df.drop(df[(df.cough==1) & (df.fever==1) & (df.sore_throat==1)& (df.shortness_of_breath
           & (df.head_ache==1) & (df.corona_result==1)].index,inplace=True)
```

In [30]:
```
df.shape
```

Out[30]:
```
(315557, 6)
```

In [31]:
```
covid_zero.corona_result.value_counts()
```

Out[31]:
```
0    2294115
```

```
Name: corona_result, dtype: int64
```

In [32]:
```
covid_one.corona_result.value_counts()
```

Out[32]:
```
1    517
Name: corona_result, dtype: int64
```

In [33]:
```
df.corona_result.value_counts()
```

Out[33]:
```
1    219164
0     96393
Name: corona_result, dtype: int64
```

- in above cell we can observe the amount of dulicate data we have.
- as we are dealing with the health care problem we have to use some technique which can solve the data duplication problme.
- bcause of data duplication, data imbalance occurs

In [35]:
```
df.shape
```

Out[35]:
```
(315557, 6)
```

- Now we have 3 seperate dataframe.
- one with all the data with yes values.
- another with all the dta with no values.
- third one with mixed values i.e yes and no.

- let's take some sample from covid_zero because that contains more duplicated sample

In [36]:
```
covid_zero_sampled = covid_zero.sample(frac= .0003)
covid_zero_sampled.shape
```

Out[36]:
```
(688, 6)
```

In [37]:
```
df2 = pd.concat([df,covid_zero_sampled,covid_one],axis=0)
df2.shape
```

Out[37]:
```
(316762, 6)
```

In [38]:
```
df2.corona_result.value_counts()
```

Out[38]:
```
1    219681
0     97081
```

```
Name: corona_result, dtype: int64
```

In [39]:
```
covid_pos_rate = np.sum(df.corona_result) / len(df.corona_result) *100
print('Covid +ve rate - ',covid_pos_rate)
```

```
Covid +ve rate -  69.4530623627427
```

- now we have some amount of balanced data lets train the model on the data and see what happen's if we have duplicate data also.

In [40]:
```
X = df2.drop('corona_result', axis=1 )
y = df2['corona_result']
X_train,X_test,y_train,y_test = train_test_split(X,y,train_size=0.7,random_state=42)
X_train.shape , X_test.shape, y_train.shape, y_test.shape
```

Out[40]: ((221733, 5), (95029, 5), (221733,), (95029,))

- Let's write a function to automated the metrices.

In [45]:
```
import numpy as np
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import cohen_kappa_score, roc_auc_score
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import log_loss

def classification_metric(y_test,y_pred,y_prob,label,n=1,verbose=False):

    # confusion matrix
    cm = confusion_matrix(y_test,y_pred)
    row_sum = cm.sum(axis=0)
    cm = np.append(cm,row_sum.reshape(1,-1),axis=0)
    col_sum = cm.saum(axis=1)
    cm = np.append(cm,col_sum.reshape(-1,1),axis=1)

    labels = label+['Total']
    plt.figure(figsize=(10,6))
    sns.heatmap(cm,annot=True,cmap='summer',fmt='0.2f',xticklabels=labels,
                yticklabels=labels,linewidths=3,cbar=None,)

    plt.xlabel('Predicted Values')
    plt.ylabel('Actual Values')
    plt.title('Confusion Matrix')
    plt.show()

    print('*'*30+'Classifcation Report'+'*'*30+'\n\n')
    cr = classification_report(y_test,y_pred)
    print(cr)

    print('\n'+'*'*36+'Kappa Score'+'*'*36+'\n\n')
    # Kappa score
```

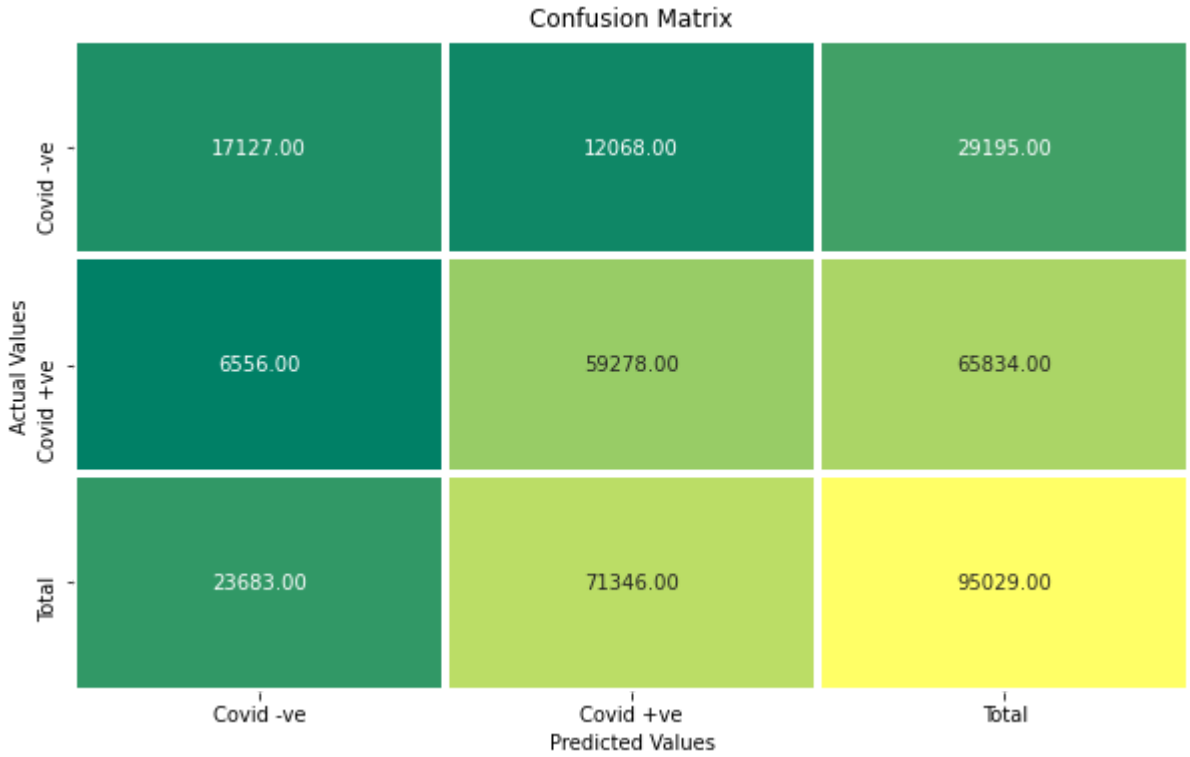
```
    kappa = cohen_kappa_score(y_test,y_pred) # Kappa Score
    print('Kappa Score =',kappa)

    print('\n'+'*'*30+'Area Under Curve Score'+'*'*30+'\n\n')
    # Kappa score
    roc_a = roc_auc_score(y_test,y_pred) # Kappa Score
    print('AUC Score =',roc_a)

    # ROC
    plt.figure(figsize=(8,5))
    fpr,tpr, thresh = roc_curve(y_test,y_prob)
    plt.plot(fpr,tpr,'r')
    print('Number of probabilities to build ROC =',len(fpr))
    if verbose == True:
        for i in range(len(thresh)):
            if i%n == 0:
                plt.text(fpr[i],tpr[i],'%0.2f'%thresh[i])
                plt.plot(fpr[i],tpr[i],'v')


    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characterstic')
    plt.legend(['AUC = {}'.format(roc_a)])
    plt.plot([0,1],[0,1],'b--',linewidth=2.0)
    plt.grid()
    plt.show()









class threshold():

    def __init__(self):
        self.th = 0.5

    def predict_threshold(self,y):
        if y >= self.th:
            return 1
        else:
            return 0
```

In [48]:

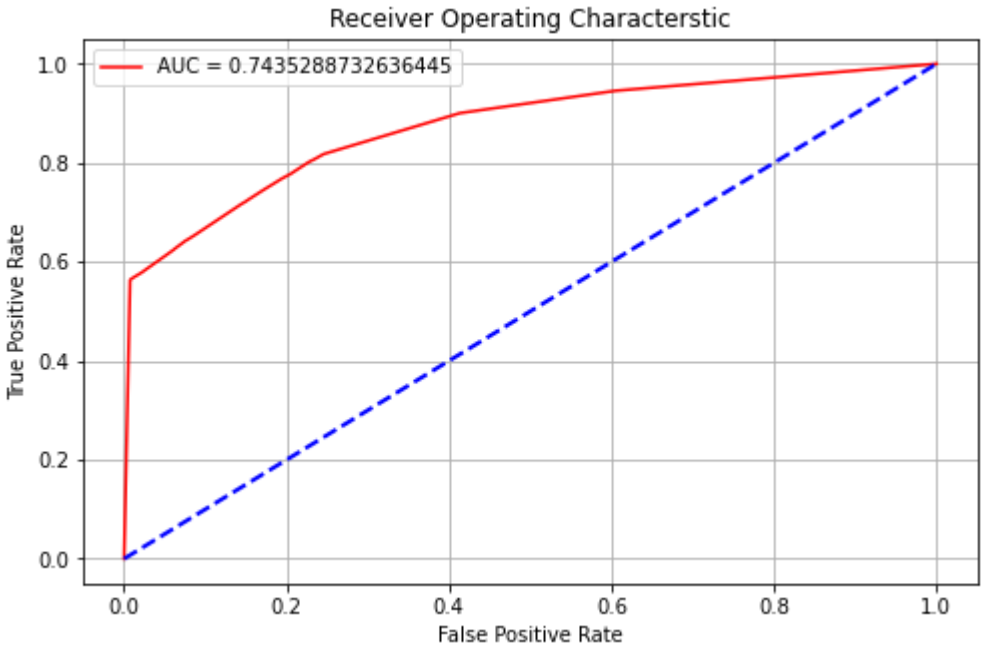
```
mlp = MLPClassifier()
mlp.fit(X_train,y_train)
y_pred_mlp=mlp.predict(X_test)
y_pred_prob_mlp = mlp.predict_proba(X_test)[:,1]
print('MLP Classifier with default parameter ')
print('Trainig Score: ', mlp.score(X_train,y_train))
print('Testing Accuracy Score: ', metrics.accuracy_score(y_test,y_pred_mlp))
```

```
MLP Classifier with default parameter
Trainig Score:  0.8039624232748396
Testing Accuracy Score:  0.8040177209062497
```

In [44]:
```
classification_metric(y_test,y_pred_mlp,y_pred_prob_mlp,label=['Covid -ve','Covid +ve'])
```




```
*******************************Classifcation Report*******************************

              precision    recall  f1-score   support

           0       0.72      0.59      0.65     29195
           1       0.83      0.90      0.86     65834

    accuracy                           0.80     95029
   macro avg       0.78      0.74      0.76     95029
weighted avg       0.80      0.80      0.80     95029


***********************************Kappa Score***********************************


Kappa Score = 0.514065112754125

*****************************Area Under Curve Score*****************************


AUC Score = 0.7435288732636445
Number of probabilities to build ROC = 36
```




- here we have trained MLP classifier with all default parameter and we got good accuracy.
- also the prediction is not one sided hence it mean that we have balanced our data correctly.

- Let's Fine Tune the model by adding some hidden layers and chaning the optimizers and loss.

In [49]:
```
mlp = MLPClassifier(hidden_layer_sizes=(200,200),solver='sgd')
mlp.fit(X_train,y_train)
y_pred_mlp=mlp.predict(X_test)
y_pred_prob_mlp = mlp.predict_proba(X_test)[:,1]
print('MLP CLassifier with fine tunned parameter ')
print('Trainig Score: ', mlp.score(X_train,y_train))
print('Testing Accuracy Score: ', metrics.accuracy_score(y_test,y_pred_mlp))
```

```
MLP CLassifier with fine tunned parameter
Trainig Score:  0.8039624232748396
Testing Accuracy Score:  0.8040177209062497
```

In [50]:
```
mlp = MLPClassifier(hidden_layer_sizes=(200,200),activation='tanh',solver='sgd',max_ite
mlp.fit(X_train,y_train)
y_pred_mlp=mlp.predict(X_test)
y_pred_prob_mlp = mlp.predict_proba(X_test)[:,1]
print('MLP CLassifier with fine tunned parameter ')
print('Trainig Score: ', mlp.score(X_train,y_train))
print('Testing Accuracy Score: ', metrics.accuracy_score(y_test,y_pred_mlp))
```

```
MLP CLassifier with fine tunned parameter
Trainig Score:  0.8039624232748396
Testing Accuracy Score:  0.8040177209062497
```

## Conclusion -

- As we have trainedd 3 different models with different parameters but have't gow any change in accuracy neither decreasing nor increasing.
- as we know that we have imabalnced and duplicated data that's why this is hapening.
- our model with deafult parameter is my suggesion on this data.

## Issue -

- As the data is duplicated and imabalnced hence model is learning the data directly rather than learning other important characteristics.
- we need more unique data which is not imabalnced nor duplicated for perfection in accuracy.
- also we need more dimensional data.

In [ ]: