VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Fall Semester 2020-21

ARTIFICIAL INTELLIGENCE CSE 3013

SLOT: - F2

FACULTY: - SUBHASHINI.R

REVIEW 3 REPORT

SUBMITTED BY -

B Goutham	18BIS0112	B.Tech ECE spl IOT
K.Likhith	18BCE2380	B.Tech CSE

PROJECT TITLE

Predict admission to the ICU of confirmed COVID-19 cases

PROBLEM DEFINITION:-

The day when covid 19 was declared as pandemic by W.H.O, everyone was surged to their personal protection. Number of patients who are affected by this virus is increasing day by day. Medical facilities are not enough to treat everyone in hospitals. Manpower is also not in sufficient number to take care of the patients. Use of artificial intelligence is aroused in health care industry in drastic way.

The first question when a patient takes admission in the hospital is “Will this person has a need to get admitted in I.C.U?” To answer this question a machine-learning model is being trained on the patient’s data. This helps the hospitals to take care for the right person in less time.

Literature Review:

COVID-19 patients' clinical characteristics, discharge rate, and fatality rate of meta-analysis:-

The aim of this study was to analyze the clinical data, discharge rate, and fatality rate of COVID-19 patients for clinical help. The clinical data of COVID-19 patients from December 2019 to February 2020 were retrieved from four databases. We statistically analyzed the clinical symptoms and laboratory results of COVID-19
patients and explained the discharge rate and fatality rate with a single-arm meta- analysis. The available data of 1994 patients in 10 literatures were included in our study. The main clinical symptoms of COVID-19 patients were fever (88.5%), cough (68.6%), myalgia or fatigue (35.8%), expectoration (28.2%), and dyspnea (21.9%). Minor symptoms include headache or dizziness (12.1%), diarrhea (4.8%), nausea and vomiting (3.9%). The results of the laboratory showed that the lymphocytopenia (64.5%), increase of C-reactive protein (44.3%), increase of lactic dehydrogenase (28.3%), and leukocytopenia (29.4%) were more common.
The results of single-arm meta-analysis showed that the male took a larger percentage in the gender distribution of COVID-19 patients 60% (95% CI [0.54, 0.65]), the discharge rate of COVID-19 patients was 52% (95% CI [0.34,0.70]), and the fatality rate was 5% (95% CI [0.01,0.11]).

A comparative evaluation of the generalised predictive ability of eight machine learning algorithms across ten clinical metabolomics data sets for binary classification

Metabolomics is increasingly being used in the clinical setting for disease diagnosis, prognosis and risk prediction. Machine learning algorithms are particularly important in the construction of multivariate metabolite prediction. Historically, partial least squares (PLS) regression has been the gold standard for binary classification. Nonlinear machine learning methods such as random forests (RF), kernel support vector machines (SVM) and artificial neural networks (ANN) may be more suited to modelling possible nonlinear metabolite covariance, and thus provide better predictive models.

Machine Learning to Predict the Risk of Incident Heart Failure Hospitalization Among Patients With Diabetes: The WATCH-DM Risk Score

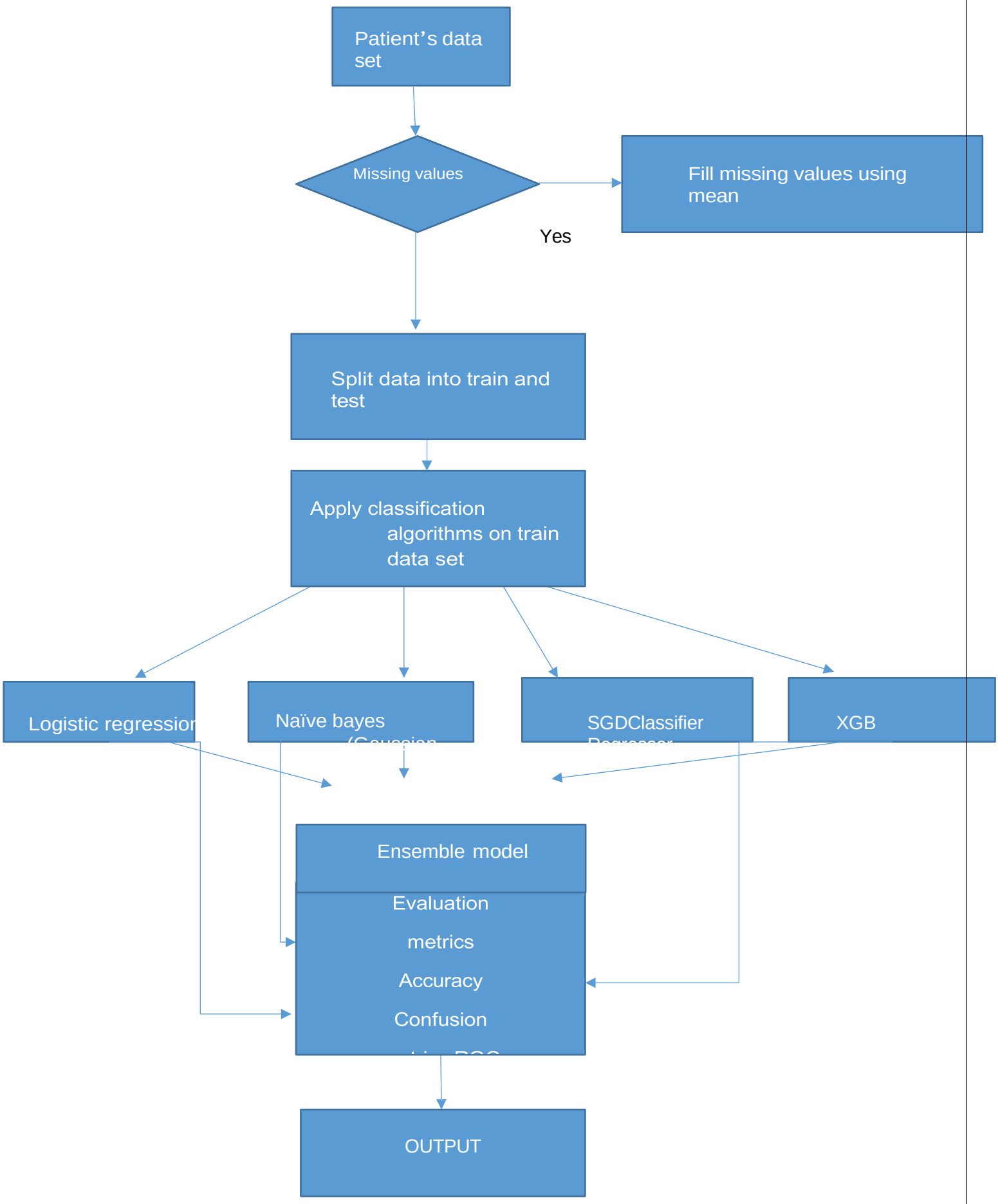
Using data from 8,756 patients free at baseline of HF, with <10% missing data, and enrolled in the Action to Control Cardiovascular Risk in Diabetes (ACCORD) trial, we used random survival forest (RSF) methods, a nonparametric decision tree machine learning approach, to identify predictors of incident HF. The RSF model was externally validated in a cohort of individuals with T2DM using the Antihypertensive and Lipid-Lowering Treatment to Prevent Heart Attack Trial (ALLHAT).

Prediction of epidemic trends in COVID-19 with logistic model and machine learning technics

COVID-19 has now had a huge impact in the world, and more than 8 million people in more than 100 countries are infected. To contain its spread, a number of countries published control measures. However, it's not known when the epidemic will end in global and various countries. Predicting the trend of COVID-19 is an extremely important challenge. We integrate the most updated COVID-19 epidemiological data before June 16, 2020 into the Logistic model to fit the cap of epidemic trend, and then feed the cap value into FbProphet model, a machine learning based time series prediction model to derive the epidemic curve and predict the trend of the epidemic. Three significant points are summarized from our modeling results for global, Brazil, Russia, India, Peru and Indonesia. Under mathematical estimation, the global outbreak will peak in late October, with an estimated 14.12 million people infected cumulatively.

Machine learning based early warning system enables accurate mortality risk prediction for COVID-19

Soaring cases of coronavirus disease (COVID-19) are pummeling the global health system. Overwhelmed health facilities have endeavored to mitigate the pandemic, but mortality of COVID-19 continues to increase. Here, we present a mortality risk prediction model for COVID-19 (MRPMC) that uses patients' clinical data on admission to stratify patients by mortality risk, which enables prediction of physiological deterioration and death up to 20 days in advance. This ensemble model is built using four machine learning methods including Logistic Regression, Support Vector Machine, Gradient Boosted Decision Tree, and Neural Network. We validate MRPMC in an internal validation cohort and two external validation cohorts, where it achieves an AUC of 0.9621 (95% CI: 0.9464-0.9778), 0.9760 (0.9613-0.9906), and 0.9246 (0.8763-0.9729), respectively. This model enables expeditious and accurate mortality risk stratification of patients with COVID-19, and potentially facilitates more responsive health systems that are conducive to high risk COVID-19 patients.


Patient's data set
Missing values
Fill missing values using mean
Yes
Split data into train and test
Apply classification algorithms on train data set
Logistic regressior
Naïve bayes
SGDClassifier
XGB
Ensemble model
Evaluation
metrics
Accuracy
Confusion
OUTPUT

STEPS TO BE FOLLOWED: -

Import necessary packages and load the data set.

Understand the data set with metadata.

Clean the null values in the data using

mean. Split the data set into train and

test data.

Import the model regression and rain model with given data.

Modulate data using k-fold techniques and train the models separately(each type of regression model is trained separately two times once with straight data and again with modified data.)

Perform evaluation metrics for each model (accuracy, confusion matrix and roc curve).

Create an ensemble model using the regression types and train the ensemble model.

Conduct evaluation metrics for the ensemble model.

MODULES USED

Pandas :- pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

Numpy:- Provides a huge boost in the mathematical fields

with matrices and arrays

Matplotlib:- Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python

Scikit learn(sklearn):- Scikit-learn is a free software machine learning library for the Python programming language.

Stratified k fold

Stratified K-Folds cross-validator. Provides train/test indices to split data in train/test sets. This cross-validation object is a variation of KFold that returns stratified folds. The folds are made by preserving the percentage of samples for each class.

Stratification is the process of rearranging the data as to ensure each fold is a good representative of the whole. For example in a binary classification problem where each class comprises 50% of the data, it is best to arrange the data such that in every fold, each class comprises around half the instances.

Code and detailed module explanation:-

MODULE 1

Load the data set:-

The patients' dataset is available in kaggle.

```
from sklearn.impute   import SimpleImputer
import pandas as pd
df=pd.read_excel("C:\\Users\\admin\\Desktop\\datasets\\605991_1272346_bundle_archive\\Kaggle_Sirio_Libanes_ICU_Prediction.xlsx")
```

MODULE 2:-

Understand the data:-

The following code snippet will show

```
df.columns
```

```
Index(['PATIENT_VISIT_IDENTIFIER', 'AGE_ABOVE65', 'AGE_PERCENTIL', 'GENDER',
       'DISEASE GROUPING 1', 'DISEASE GROUPING 2', 'DISEASE GROUPING 3',
       'DISEASE GROUPING 4', 'DISEASE GROUPING 5', 'DISEASE GROUPING 6',
       ...
       'TEMPERATURE_DIFF', 'OXYGEN_SATURATION_DIFF',
       'BLOODPRESSURE_DIASTOLIC_DIFF_REL', 'BLOODPRESSURE_SISTOLIC_DIFF_REL',
       'HEART_RATE_DIFF_REL', 'RESPIRATORY_RATE_DIFF_REL',
       'TEMPERATURE_DIFF_REL', 'OXYGEN_SATURATION_DIFF_REL', 'WINDOW', 'ICU'],
      dtype='object', length=231)
```

The following code will tell us the null value count of each column in the dataset.

```
df.isnull().sum()
```

```
PATIENT_VISIT_IDENTIFIER        0
AGE_ABOVE65                     0
AGE_PERCENTIL                   0
GENDER                          0
DISEASE GROUPING 1              5
                             ...
RESPIRATORY_RATE_DIFF_REL     748
TEMPERATURE_DIFF_REL          694
OXYGEN_SATURATION_DIFF_REL    686
WINDOW                          0
ICU                             0
Length: 231, dtype: int64
```

Module 3:-

Clear the null values in the data set. Cleaning of the dataset is very important for any dataset which helps the model to train a better way.

First change the column values in string type to numerical type.

```
# Data Preparation
df['AGE_PERCENTIL'] = df['AGE_PERCENTIL'].str.replace('Above ','').str.extract(r'(.+?)th')
df['WINDOW'] = df['WINDOW'].str.replace('ABOVE_12','12-more').str.extract(r'(.+?)-')
```

Fill the null values with the mean values of the column. This can be done by using imputer method in sklearn module.

```
 # Missingness as features
df['row_missingness'] = df.isnull().sum(axis=1)
```

```
# Mean imputation
mean_impute  = SimpleImputer(strategy='mean')
imputed_data = mean_impute.fit_transform(df)
imputed_data = pd.DataFrame(imputed_data, columns = df.columns)
```

Now check the null values in the data set. In the following picture we can observe that there are no null values and each column is of datatype integer.

```
imputed_data.isnull().sum()
```

```
PATIENT_VISIT_IDENTIFIER      0
AGE_ABOVE65                   0
AGE_PERCENTIL                 0
GENDER                        0
DISEASE GROUPING 1            0
                             ..
TEMPERATURE_DIFF_REL          0
OXYGEN_SATURATION_DIFF_REL    0
WINDOW                        0
ICU                           0
row_missingness               0
Length: 232, dtype: int64
```

MODULE 3

Divide the dataset into train and testing parts.

```
target=["ICU"]
un=["row_missingness"]
numericals=list(set(imputed_data.columns.values)-set(target)-set(un))
len(numericals)
```

```
230
```

```
new_df=imputed_data[numericals]
new_df.shape
tar=imputed_data[target]
tar.shape
from sklearn.model_selection import train_test_split
x,x_test,y,y_test=train_test_split(new_df,tar,test_size=0.3)
x.shape
```

The data is being divided in 7:3 ratio for train and test data using train test split method in sklearn module.

MODULE 4

Train the binary classification models. We are going to train by using 4 algorithms.

4.a.1 Logistic regression:-

The following figure will show that we create a model based on logistic regression and we train the model with training data.

```
import warnings
warnings.filterwarnings('ignore')
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(x,y)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Now we can evaluate the model using metrics. Accuracy of the model is reported as 84.4% and the confusion matrix.

```
from sklearn.metrics import accuracy_score,confusion_matrix,roc_auc_score,roc_curve
accuracy_score(y_test,pred_val)
```

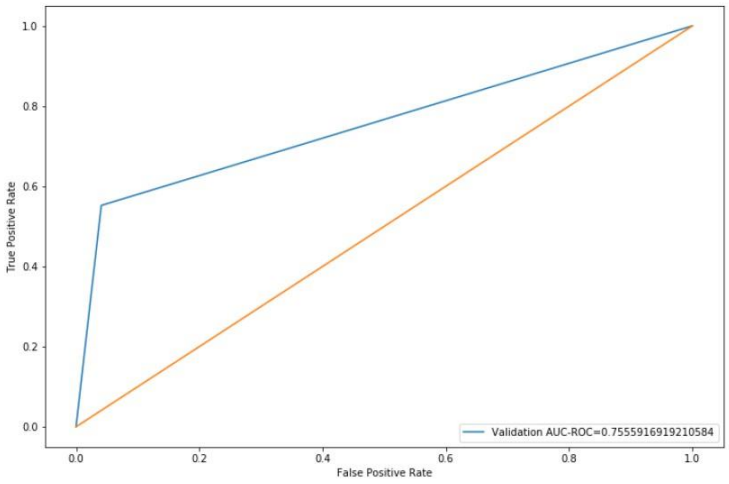
```
0.8442906574394463
```

```
confusion_matrix(y_test,pred_val)
```

```
array([[398,  17],
       [ 73,  90]], dtype=int64)
```

Visualizing the roc curve of the model using matplotlib library. For visualizing the roc curve two parameters are required. Model predictions of test data part and original test values.

```
import numpy as np
import matplotlib.pyplot as plt
fpr,tpr,_=roc_curve(y_test,pred_val)
auc = roc_auc_score(y_test, pred_val)
plt.figure(figsize=(12,8))
plt.plot(fpr,tpr,label="Validation AUC-ROC="+str(auc))
x = np.linspace(0, 1, 1000)
plt.plot(x, x, linestyle='-')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.show()
```



4.a.2 logistic regression with stratified k fold

```
from sklearn.model_selection import StratifiedKFold
accuracy=[]
skf=StratifiedKFold(n_splits=10,random_state=None)
skf.get_n_splits(new_df,tar)
```

```
10
```

```
for train_index,test_index in skf.split(new_df,tar):
    x1_train,x1_test=new_df.iloc[train_index],new_df.iloc[test_index]
    y1_train,y1_test=tar.iloc[train_index],tar.iloc[test_index]
    model23.fit(x1_train,y1_train)
    prediction=model23.predict(x1_test)
    score=accuracy_score(prediction,y1_test)
    accuracy.append(score)
```

The accuracy of each fold is shown in the following figure.

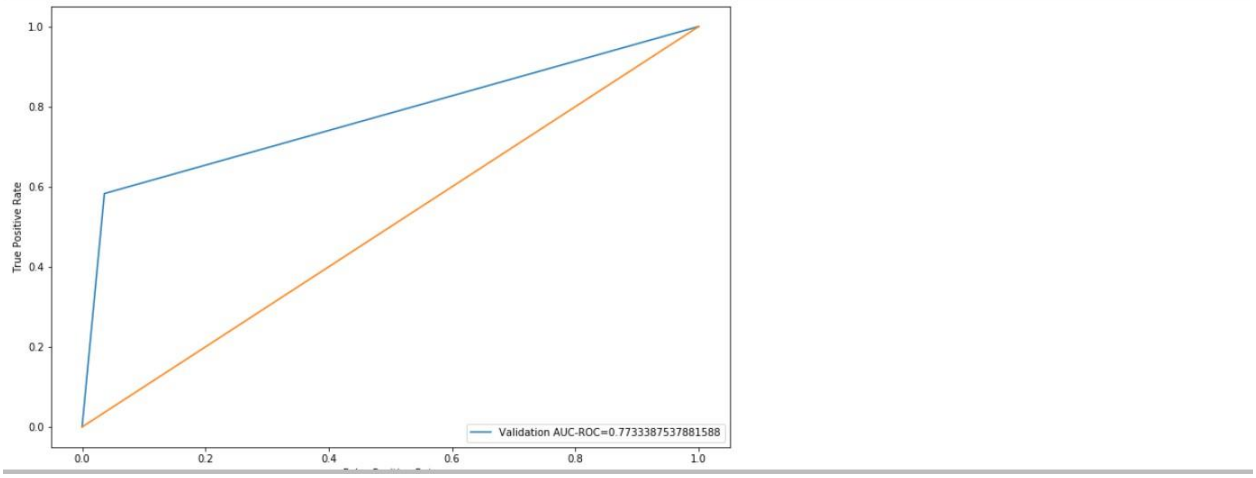
```
print(accuracy)
```

```
[0.8911917098445595, 0.8704663212435233, 0.844559585492228, 0.8601036269430051, 0.8290155440414507, 0.8385416666666666, 0.8333333333333334, 0.8802083333333334, 0.859375, 0.8489583333333334]
```

The final accuracy of the model can be taken as mean of the array.

Evaluation metrics for this model is.

```
fpr,tpr,_=roc_curve(y_test,pred23)
auc = roc_auc_score(y_test, pred23)
plt.figure(figsize=(12,8))
plt.plot(fpr,tpr,label="Validation AUC-ROC="+str(auc))
x = np.linspace(0, 1, 1000)
plt.plot(x, x, linestyle='-')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.show()
```




```
confusion_matrix(y_test,pred23)
```

```
array([[400,  15],
       [ 68,  95]], dtype=int64)
```

4.b.1 Gaussian NB

The following figure will show that we create a model based on Gaussian NB and we train the model with training data.

```
from sklearn.naive_bayes import GaussianNB
nb=GaussianNB()
nb.fit(x,y)
```

Now we can evaluate the model using metrics. Accuracy of the model is reported as 77.4% and the confusion matrix.

```
accuracy_score(y_test,y_pred1)
```

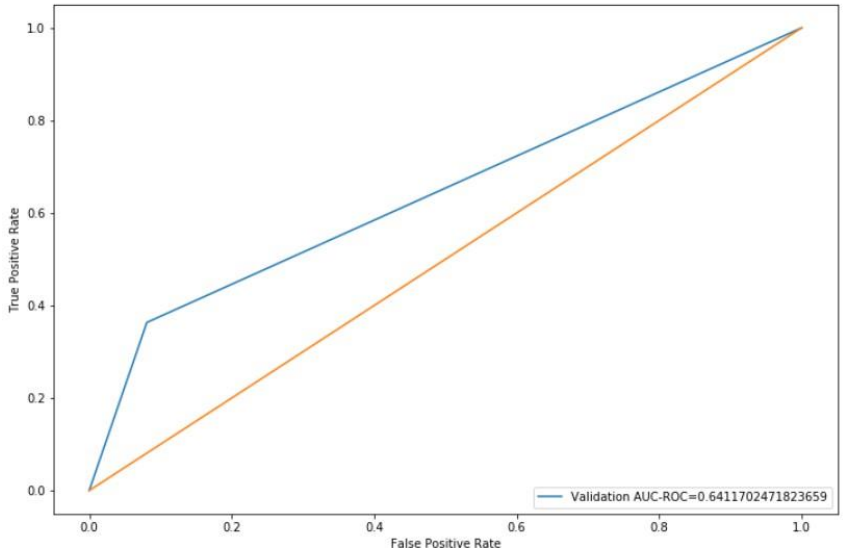
```
0.7740259740259741
```

```
confusion_matrix(y_test,y_pred1)
```

```
array([[523,  46],
       [128,  73]], dtype=int64)
```

Roc curve:-

```
fpr,tpr,_=roc_curve(y_test,y_pred1)
auc = roc_auc_score(y_test, y_pred1)
plt.figure(figsize=(12,8))
plt.plot(fpr,tpr,label="Validation AUC-ROC="+str(auc))
x = np.linspace(0, 1, 1000)
plt.plot(x, x, linestyle='-')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.show()
```



4.b.2 Gaussian NB with stratified k-fold

```
from sklearn.model_selection import StratifiedKFold
acc=[]
skf=StratifiedKFold(n_splits=10,random_state=None)
skf.get_n_splits(new_df,tar)
```

```
10
```

```
from sklearn.naive_bayes import GaussianNB
nb2=GaussianNB()
for train_index,test_index in skf.split(new_df,tar):
    x1_train,x1_test=new_df.iloc[train_index],new_df.iloc[test_index]
    y1_train,y1_test=tar.iloc[train_index],tar.iloc[test_index]
    nb2.fit(x1_train,y1_train)
    prediction=nb2.predict(x1_test)
    score=accuracy_score(prediction,y1_test)
    acc.append(score)
```

Accuracy of this model is

```
print(acc)
```

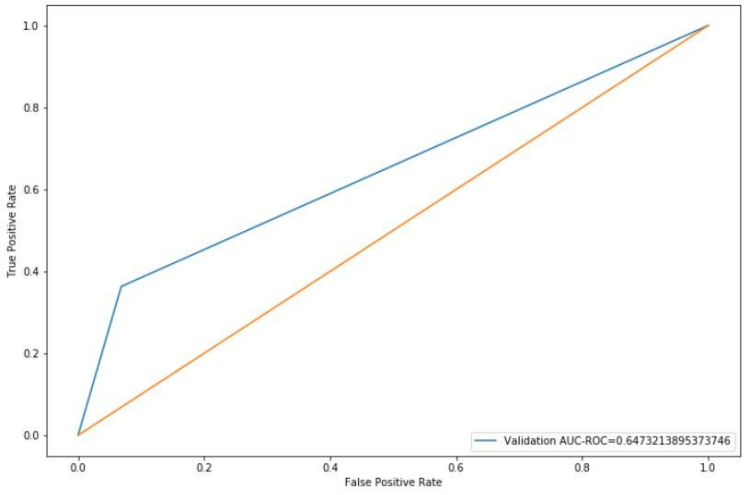
```
[0.7823834196891192, 0.7616580310880829, 0.7461139896373057, 0.7875647668393783, 0.772020725388601, 0.8020833333333334, 0.78125, 0.7916666666666666, 0.8072916666666666, 0.7916666666666666]
```

```
np.array(acc).mean()
```

```
0.7823699265975821
```

Confusion matrix and roc curve of the model

```
fpr,tpr,_=roc_curve(y_test,pred34)
auc = roc_auc_score(y_test, pred34)
plt.figure(figsize=(12,8))
plt.plot(fpr,tpr,label="Validation AUC-ROC="+str(auc))
x = np.linspace(0, 1, 1000)
plt.plot(x, x, linestyle='-')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.show()
```




```
confusion_matrix(y_test,pred34)
```

```
array([[530,  39],
       [128,  73]], dtype=int64)
```

4.c.1 SGD classifier

The following figure will show that we create a model based on SGD Classifier and we train the model with training data.

```
from sklearn.linear_model import SGDClassifier
sgd=SGDClassifier(loss='modified_huber',shuffle=True,random_state=101)
sgd.fit(x,y)
```

Now we can evaluate the model using metrics. Accuracy of the model is reported as 78.8% and the confusion matrix.

```
accuracy_score(y_test,pred)
```

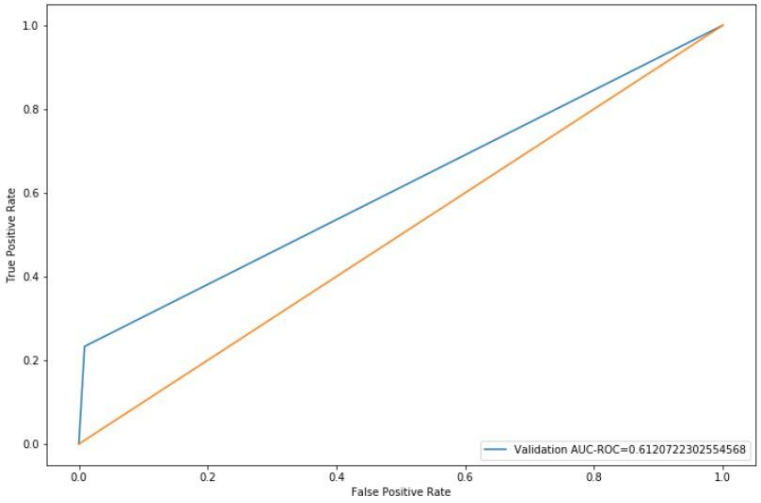
```
0.7883116883116883
```

```
confusion_matrix(y_test,pred)
```

```
array([[559,   5],
       [158,  48]], dtype=int64)
```

Roc curve:-

```
fpr,tpr,_=roc_curve(y_test,pred)
auc = roc_auc_score(y_test, pred)
plt.figure(figsize=(12,8))
plt.plot(fpr,tpr,label="Validation AUC-ROC="+str(auc))
x = np.linspace(0, 1, 1000)
plt.plot(x, x, linestyle='-')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.show()
```



4.c.2 SGD classifier with stratified k-fold

```
from sklearn.model_selection import StratifiedKFold
acc2=[]
skf=StratifiedKFold(n_splits=10,random_state=None)
skf.get_n_splits(new_df,tar)
```

```
10
```

```
from sklearn.linear_model import SGDClassifier
sgd2=SGDClassifier(loss='modified_huber',shuffle=True,random_state=101)
for train_index,test_index in skf.split(new_df,tar):
    x1_train,x1_test=new_df.iloc[train_index],new_df.iloc[test_index]
    y1_train,y1_test=tar.iloc[train_index],tar.iloc[test_index]
    sgd2.fit(x1_train,y1_train)
    prediction=nb2.predict(x1_test)
    score=accuracy_score(prediction,y1_test)
    acc2.append(score)
```

Evaluation metrics of this model

```
print(acc2)
saved_model6=pickle.dumps(sgd2)
```

```
[0.7823834196891192, 0.7616580310880829, 0.7461139896373057, 0.8031088082901554, 0.7823834196891192, 0.8072916666666666, 0.796875, 0.796875, 0.8125, 0.7916666666666666]
```

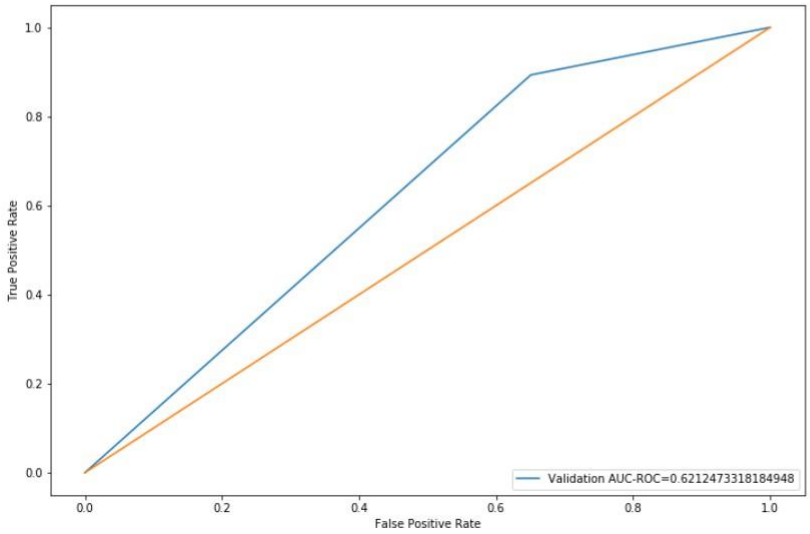
```
np.array(acc2).mean()
```

```
0.7880856001727117
```

```
confusion_matrix(y_test,pred45)
```

```
array([[197, 367],
       [ 22, 184]], dtype=int64)
```

```
fpr,tpr,_=roc_curve(y_test,pred45)
auc = roc_auc_score(y_test, pred45)
plt.figure(figsize=(12,8))
plt.plot(fpr,tpr,label="Validation AUC-ROC="+str(auc))
x = np.linspace(0, 1, 1000)
plt.plot(x, x, linestyle='-')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.show()
```



4.d.1 XGB regressor

The following figure will show that we create a model based on XGB regressor and we train the model with training data.

```
from xgboost import XGBClassifier
model7=XGBClassifier()
model7.fit(x,y)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

Now we can evaluate the model using metrics. Accuracy of the model is reported as 88.9% and the confusion matrix.

```
accuracy_score(pred7,y_test)
```

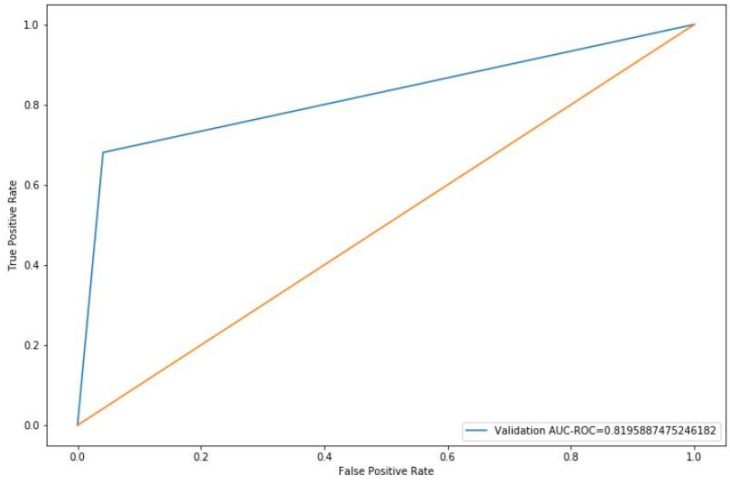
```
0.8896103896103896
```

```
confusion_matrix(y_test,pred7)
```

```
array([[555,  24],
       [ 61, 130]], dtype=int64)
```

Roc curve:-

```
fpr,tpr,_=roc_curve(y_test,pred7)
auc = roc_auc_score(y_test, pred7)
plt.figure(figsize=(12,8))
plt.plot(fpr,tpr,label="Validation AUC-ROC="+str(auc))
x = np.linspace(0, 1, 1000)
plt.plot(x, x, linestyle='-')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.show()
```



4.d.2XGB regressor with stratified k fold

```
from sklearn.model_selection import StratifiedKFold
acc3=[]
skf=StratifiedKFold(n_splits=10,random_state=None)
skf.get_n_splits(new_df,tar)
```

```
10
```

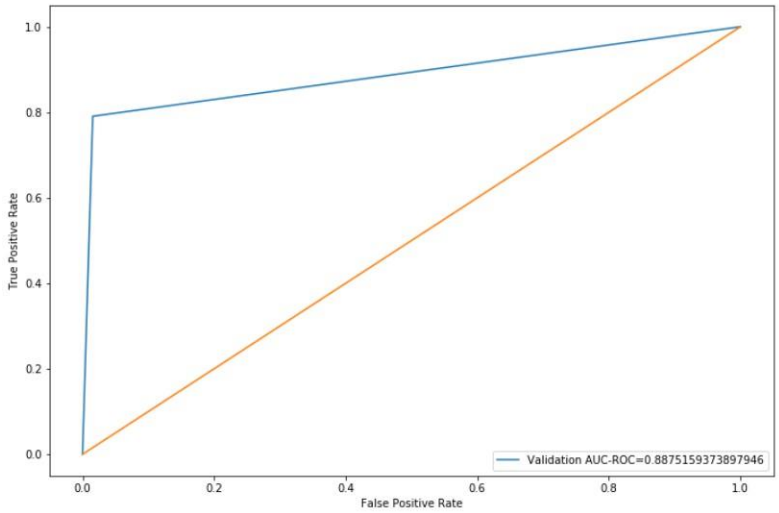
```
from xgboost import XGBClassifier
model8=XGBClassifier()
for train_index,test_index in skf.split(new_df,tar):
    x1_train,x1_test=new_df.iloc[train_index],new_df.iloc[test_index]
    y1_train,y1_test=tar.iloc[train_index],tar.iloc[test_index]
    model8.fit(x1_train,y1_train)
    prediction=model8.predict(x1_test)
    score=accuracy_score(prediction,y1_test)
    acc3.append(score)
```

Evaluation metrics

```
print(np.array(acc3).mean())
```

```
0.8768701424870466
```

```
fpr,tpr,_=roc_curve(y_test,pred56)
auc = roc_auc_score(y_test, pred56)
plt.figure(figsize=(12,8))
plt.plot(fpr,tpr,label="Validation AUC-ROC="+str(auc))
x = np.linspace(0, 1, 1000)
plt.plot(x, x, linestyle='-')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.show()
```




```
confusion_matrix(y_test,pred56)
```

```
array([[570,   9],
       [ 40, 151]], dtype=int64)
```

4.e

Ensemble model

We can ensemble all models into one stack model and we can train the model by using the dataset

Create and run ensemble model

```
from sklearn.ensemble import StackingClassifier
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import cross_val_score
def get_stacking():
    # define the base models
    level0 = list()
    level0.append(('lr', LogisticRegression()))
    level0.append(('xgb', XGBClassifier()))
    level0.append(('bayes', GaussianNB()))
    # define meta learner model
    level1 = LogisticRegression()
    # define the stacking ensemble
    model = StackingClassifier(estimators=level0, final_estimator=level1, cv=5)
    return model
```

```
def get_models():
    models = dict()
    models['lr'] = LogisticRegression()
    models['xgb'] = XGBClassifier()
    models['bayes'] = GaussianNB()
    models['stacking'] = get_stacking()
    return models
```

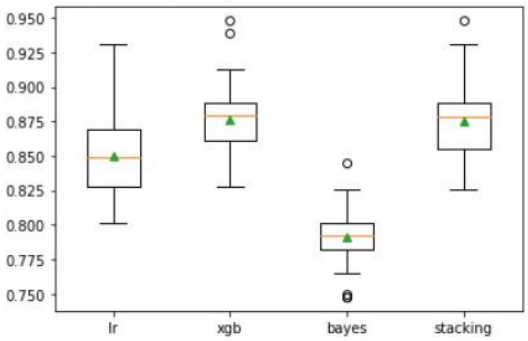
```
def evaluate_model(model, X, y):
    cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
    scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1, error_score='raise')
    return scores
```

```
from sklearn.model_selection import train_test_split
x,x_test,y,y_test=train_test_split(new_df,tar,test_size=0.4)
models = get_models()
# evaluate the models and store results
results, names = list(), list()
for name, model in models.items():
    scores = evaluate_model(model, x, y)
    results.append(scores)
    names.append(name)
    print('>%s %.3f (%.3f)' % (name, np.mean(scores), np.std(scores)))
```

We can see the compare performances of individual model and stacked model by box plat and bar plot

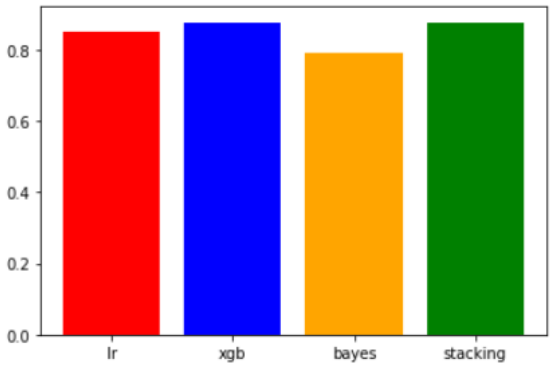
```
# plot model performance for comparison
plt.boxplot(results, labels=names, showmeans=True)
plt.show()
```

```
>lr 0.850 (0.026)
>xgb 0.876 (0.027)
>bayes 0.792 (0.022)
>stacking 0.875 (0.027)
```



```
b=[]
for i in range(4):
    b.append(results[i].mean())
b
plt.bar(names,b,color=["red","blue","orange","green"])
```

```
<BarContainer object of 4 artists>
```



References:-

https://www.kaggle.com/S%C3%ADrio-Libanes/covid19/tasks?taskId=1195

https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.ht ml

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.ht ml

https://towardsdatascience.com/logistic-regression-detailed-overview- 46c4da4303bc

https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

https://www.kaggle.com/gayathrydasika/xgb-regressor-basic

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html

https://www.kdnuggets.com/2017/02/stacking-models-imropved-predictions.html