Introduction to Data Science CS842 - University of Regina

User Guide for Analysts

on

Mortality Prediction for COVID-19 and Organ Dysfunction patients

Group Name: Data Pirates

Srushti Vaghani - 200444489 - srushtivaghani@uregina.ca

Harsh Patel - 200445899 - hpp711@uregina.ca

Neel Patel - 200451256 - neelpatel@uregina.ca

April 14, 2021

CONTENTS

USER GUIDE FOR ANALYST

Follow all the steps mentioned in Self contained Setup Guide to download and install all dependencies. In this document detailed explanation of whole code will be explained.

Covid-preprocessing and model training.ipynb

Open the "FINAL PROJECT/code/Covid-preprocessing and model training.ipynb" notebook in the 8th step of the setup guide after following all 7 steps for data cleaning and model training of covid dataset.

Importing all the necessary libraries to run this code.

Libraries	Usage
Pandas	Used to load dataset
Numpy	Format data into arrays for the input of the model built
datetime	Used to count the days form the given date
Sklearn	Used to train various models. Also used metrics to evaluate and compare the models.
Matplotlib	Used for visualization of the data and to generate various graphs and diagrams.
Keras	Used to train deep learning keras model
Seaborn	Used to plot various graphs
Os	Used to get path of the desired file from the operating system
Pickle	Store the trained model in the pickle file.
Urllib	Retrieve the dataset from the downloadable link

```
import pandas as pd
import numpy as np
from datetime import datetime
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.svm import SVC
from sklearn.utils import resample,shuffle
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_predict, train_test_split, cross_val_score
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense
from sklearn import metrics
import seaborn as sb
import os
import pickle
import urllib.request
from sklearn.metrics import roc_curve
from sklearn.metrics import auc
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import MinMaxScaler

%matplotlib inline
```

Data cleaning and visualization:

Variable name	Details
data	Dataset covid.csv is loaded in this variable and can be manipulated without affecting the original csv file.
positive_patients	From the data variable, patients extracted whose covid_result is positive.
sb	library seaborn imported as sb
pd	Pandas imported as pd
corr_matrix	Correlation matrix generated by using pearson coefficient between each columns of data
plt	Matplotlib imported as plt
upper	Upper triangle of corr_matrix stored in upper
to_drop	List of columns to be dropped having correlation greater than 0.8

Functions:

calculateDays() : It calculates the difference between the given dates as an arguments.

```
[ ]  1 # make new column of number of days after patient died
     2 def calculateDays(start_date_str, end_date_str):
     3     start_date_obj = datetime.strptime(start_date_str, '%d-%m-%Y')
     4     end_date_obj = datetime.strptime(end_date_str, '%d-%m-%Y')
     5     days = end_date_obj - start_date_obj
     6     return days.days
```

Variable name	Details
start_date_obj	First argument is converted into datetime instance from string as a starting date.
end_date_obj	second argument is converted into datetime instance from string as a ending date.
days	Difference between start_date_obj and end_date_obj which is numbers of days

Logistic Regression :

Variable name	Details
target	Name of the target column
X	records of dataset with all input columns
Y	records of dataset with the label column
X_train	Used for training model a subset of X
X_test	Used for testing model a subset of X
Y_train	Used for training model a subset of Y
Y_test	Used for testing model a subset of Y
logistic_regression	Logistic regression model initialized from the sklearn library
Y_pred_logistic	Predicted output of model for the X_test inputs
Y_pred_logistic_prob	Predicted probabilities of output of model for the X_test inputs
confusion_matrix	Confusion matrix for actual outputs(Y_test) and predicted outputs(Y_pred_logistic)
fpr_logistic	False positive rate for logistic regression model
tpr_logistic	True positive rate for logistic regression model
thresholds_logistic	Threshold values for logistic regression model
auc_logistic	Generating a curve for fpr_logistic and tpr_logistic

CalculateMetricsLogisticRegression() : It splits the dataset which is given as an input into 2 parts, i.e., inputs and labels. Then it split the whole dataset into training and testing parts, i.e., X_train, X_test, Y_train and Y_test. Then model is trained by training inputs and training labels, and then after, prediction is done on testing inputs. After that comparision of actual testing labels and predicted labels is done by construction of confusion matrix and heatmap is plotted. Also there ROC curve is generated by using fpr(false positive rate) and tpr(true positive rate).

```
def CalculateMetricsLogisticRegression(dataset):
    target='death'
    X = dataset.drop([target,'days'],axis=1)
    Y = dataset[target]
    X,Y = shuffle(X,Y)
    X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.25,random_state=12)

    logistic_regression = LogisticRegression(max_iter=200)
    logistic_regression.fit(X_train,Y_train)
    Y_pred_logistic = logistic_regression.predict(X_test)
    Y_pred_logistic.reshape(len(Y_pred_logistic))

    Y_pred_logistic_prob=logistic_regression.decision_function(X_test)
    Y_pred_logistic_prob.reshape(len(Y_pred_logistic_prob))

    confusion_matrix = pd.crosstab(Y_test, Y_pred_logistic, rownames=['Actual Values'], colnames=['Predicted Values'])
    sb.heatmap(confusion_matrix, annot=True, fmt='d', cmap='YlGnBu')
    plt.show()

    print('Precision: ',metrics.precision_score(Y_test, Y_pred_logistic))
    print('Recall: ',metrics.recall_score(Y_test, Y_pred_logistic))
    print('F1-score: ',metrics.f1_score(Y_test, Y_pred_logistic))

    fpr_logistic, tpr_logistic, thresholds_logistic = roc_curve(Y_test, Y_pred_logistic_prob)
    auc_logistic = auc(fpr_logistic, tpr_logistic)

    plt.figure(1)
    plt.plot([0, 1], [0, 1], 'k--')
    plt.plot(fpr_logistic, tpr_logistic, label='Logistic Regression (area = {:.3f})'.format(auc_logistic))
    plt.xlabel('False positive rate')
    plt.ylabel('True positive rate')
    plt.title('ROC curve')
    plt.legend(loc='best')
    plt.show()
```

Samplinng :

Variable name	Details
patients_majority	Columns of patients whose death = 0
patients_minority	Columns of Patients whose death = 1
patients_majority_length	Total number of patients whose death = 0
patients_sample	Sampled dummy patients
positive_patients_oversampled	Dummy samples of the patients whose length is matched with majority patients
positive_patients_undersampled	Patients after removing some patients to match length with minority patients
positive_patients_mixedsampled	Patients after removing some patients from majority and adding some dummy patients to minority patients.

```
patients_majority = positive_patients[positive_patients.death==0]
patients_minority = positive_patients[positive_patients.death==1]
patients_majority_length = len(patients_majority)

patients_sample = resample(patients_minority,
                           replace=True,     # sample without replacement
                           n_samples=patients_majority_length,    # to match majority class
                           random_state=123)


positive_patients_oversampled = pd.concat([patients_majority, patients_sample])
positive_patients_oversampled.death.value_counts().plot(kind='bar')
```

Classification:

Variable name	Details
target	Name of the target column i.e. death
X	records of dataset with all input columns
Y	records of dataset with the label column
X_train	Used for training model a subset of X
X_test	Used for testing model a subset of X
Y_train	Used for training model a subset of Y
Y_test	Used for testing model a subset of Y
p_scores	Set to store precision of each model trained
r_scores	Set to store recall of each model trained
f1_scores	Set to store f1-score of each model trained
logistic_regression	Instance of logistic regression from skleran
Y_pred_logistic_prob	Predicted probabilities for X_test inputs for logistic regression
Y_pred_logistic	Predicted values for X_test inputs for logistic regression
confusion_matrix	Confusion matrix for actual vales(Y_test) and predicted outputs(Y_pred_logistic)
fpr_logistic	False positive rate for logistic regression model
tpr_logistic	True positive rate for logistic regression model
thresholds_logistic	Threshold values for logistic regression model
auc_logistic	Generating a ROC curve for fpr_logistic and tpr_logistic
decisionTree_classifier	Instance of Decision Tree Classifier from skleran
Y_pred_decision	Predicted values for X_test inputs for Decision Tree Classifier
Y_pred_decision_prob	Predicted probabilities for X_test inputs for Decision Tree Classifier
fpr_decision	False positive rate for Decision Tree Classifier
tpr_decision	True positive rate for Decision Tree Classifier
thresholds_decision	Threshold values for Decision Tree Classifier
auc_decision	Generating a ROC curve for fpr_decision and tpr_decision
random_forest_classifier	Instance of Random Forest Classifier from skleran
Y_pred_forest	Predicted values for X_test inputs for Random Forest Classifier
Y_pred_forest_prob	Predicted probabilities for X_test inputs for Random Forest Classifier
fpr_forest	False positive rate for Random Forest Classifier
tpr_forest	True positive rate for Random Forest Classifier
thresholds_forest	Threshold values for Random Forest Classifier
auc_forest	Generating a ROC curve for fpr_forest and tpr_forest

classification_model	Neural Network Classification model made by using keras Sequential Class
history	Save the model's progress during training such as metrics and loss function
Y_pred_keras	Predicted values for X_test inputs for Neural Network
Y_pred_keras_prob	Predicted probabilities for X_test inputs for Neural Network
fpr_keras	False positive rate for Neural Network
tpr_keras	True positive rate for Neural Network
thresholds_keras	Threshold values for Neural Network
auc_keras	Generating a ROC curve for fpr_keras and tpr_keras
model_names	List of names of all models
p_scores_list	List of precision scores of all models
r_scores_list	List of recall scores of all models
f1_scores_list	List of f-1 scores of all models
scores	Dataframe containing all the scores of all models to visualize it in table format

	Models	Precision	recall	f1 score
0	Logistic Regression	0.83	0.88	0.86
1	Decision Tree	0.84	0.94	0.89
2	Random Forest	0.84	0.94	0.89
3	Neural Network	0.82	0.93	0.87

Regression:

Variable name	Details
regression_dataset	Dataset extracted from positive_patients by only getting patients which died
target	Name of the target column i.e. days
X	records of dataset with all input columns
Y	records of dataset with the label column
X_train	Used for training model a subset of X
X_test	Used for testing model a subset of X
Y_train	Used for training model a subset of Y
Y_test	Used for testing model a subset of Y
scaler	MinMax scaler to normalize all the numerical columns
mae_scores	Set to store mean absolute error of each model trained
mse_scores	Set to store mean squared error of each model trained
r2_scores	Set to store r2-score of each model trained
linear_regression	Instance of linear regression from skleran
Y_pred	Predicted values for X_test inputs for regression
p1	Maximum value from Y_pred or Y_test used for plotting the graph

P2	Minimum value from Y_pred or Y_test used for plotting the graph
regr	Instance of Support Vector Regressor from skleran
clf	Instance of Decision Tree Regressor from skleran
reg	Instance of Gradient Boosting Regressor from skleran
rf_regr	Instance of Random Forest Regressor from skleran
model	Neural Network Regressor model made by using keras Sequential Class
history	Save the model's progress during training such as metrics and loss function
model_names	List of names of all models
mae_scores_list	List of mean absolute error scores of all models
mse_scores_list	List of mean squared error scores of all models
r2_scores_list	List of r-1 scores of all models
scores	Dataframe containing all the scores of all models to visualize it in table format

sofa-preprocessing.ipynb

Open the "FINAL PROJECT/code/sofa-preprocessing.ipynb" notebook in the 8th step of the setup guide after following all 7 steps for data cleaning and model training of covid dataset.

Importing all the necessary libraries to run this code.

Libraries	Usage
Pandas	Used to load dataset
Numpy	Format data into arrays for the input of the model built
missingno	Library used to plot missing values heatmap of entire dataset
Matplotlib	Used for visualization of the data and to generate various graphs and diagrams.
Seaborn	Used to plot various graphs
os	Used to get path of the desired file from the operating system
Urllib	Retrieve the dataset from the downloadable link

```
1 import pandas as pd
2 import missingno as msno
3 import os
4 import urllib.request
5 import matplotlib.pyplot as plt
6 import seaborn as sb
7 import numpy as np
```

Data cleaning and visualization:

Variable name	Details
data	Dataset covid.csv is loaded in this variable and can be manipulated without affecting the original csv file.
patients	Dataset after removing all unnamed columns
msno	Instance of missgno library to plot all the missing values in the dataset
sb	library seaborn imported as sb
pd	Pandas imported as pd
rename_columns	Set of all columns with old and new names to change it to more understandable names
corr_matrix	Correlation matrix generated by using pearson coefficient between each columns of data
plt	Matplotlib imported as plt
upper	Upper triangle of corr_matrix stored in upper
to_drop	List of columns to be dropped having correlation greater than 0.8

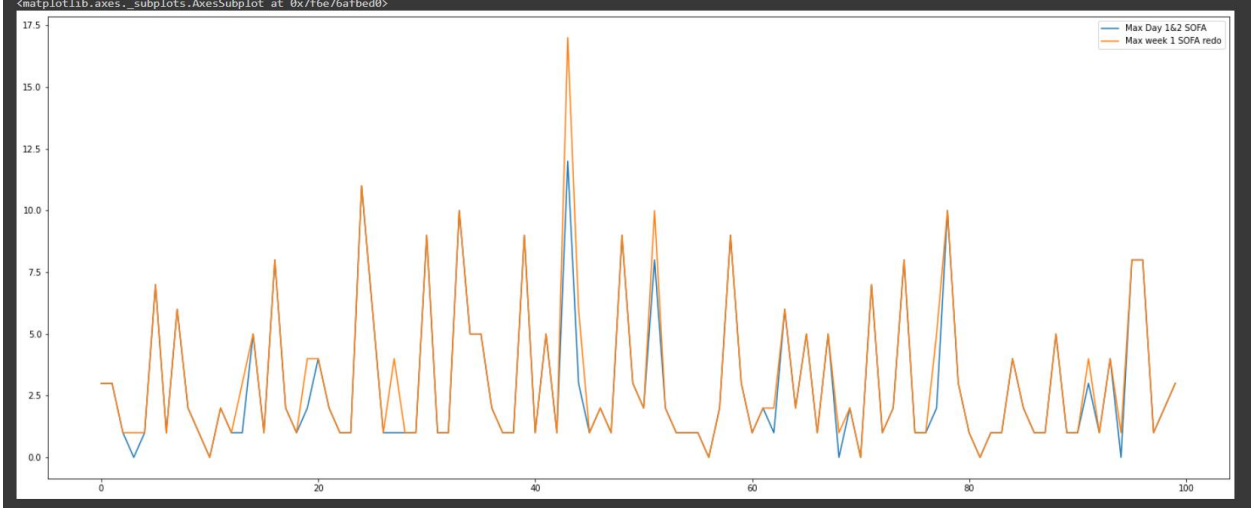
Label for regression 'Number of days' is calculated by subtracting Inpatient days prior to ICU from total hospital length of stay as the days can be only counted from the admission of patient in ICU. So y removing prior days from ICU we can get number of days in which patient may die.

```
#calculating number of days the patients was alive
patients['days'] = patients['Hospital_Length_of_Stay'] - patients['Inpatient days prior to ICU']
```

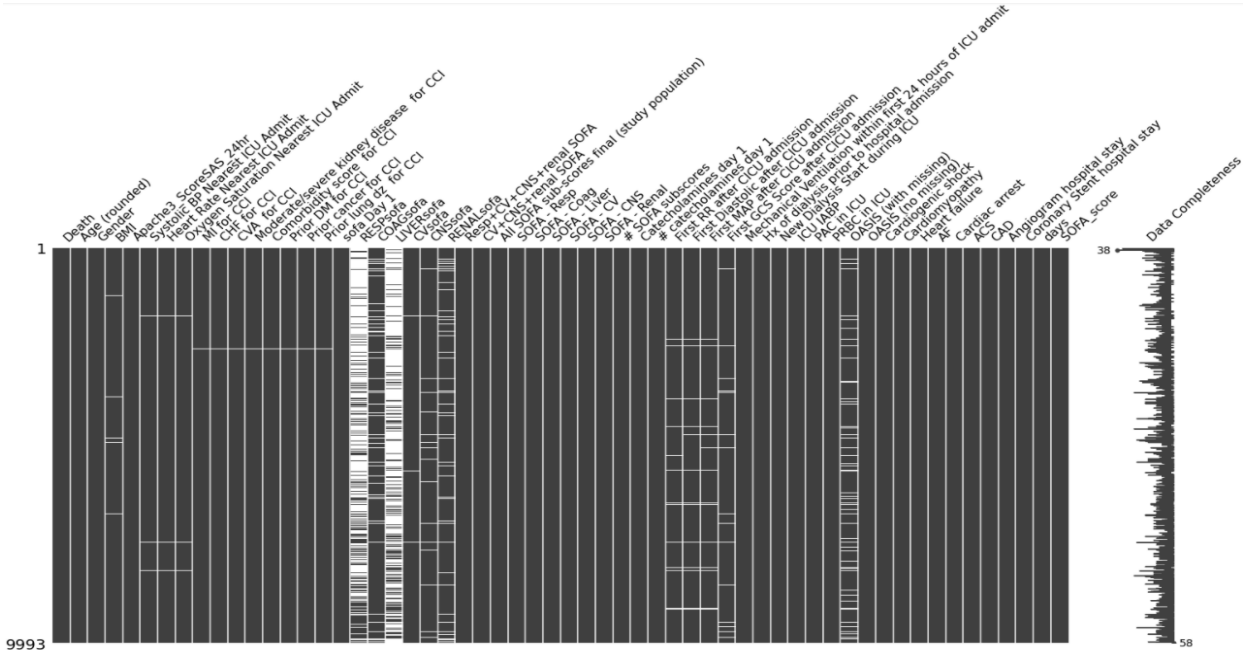
SOFA score of the patients indicates the severity of the patient. Higher SOFA score indicates the high level of severity. We consider the max sofa score reached in a week instead of two max score.

```
patients[['Max Day 1&2 SOFA','Max week 1 SOFA redo']].head(100).plot(kind='line',figsize=(25,10))
```

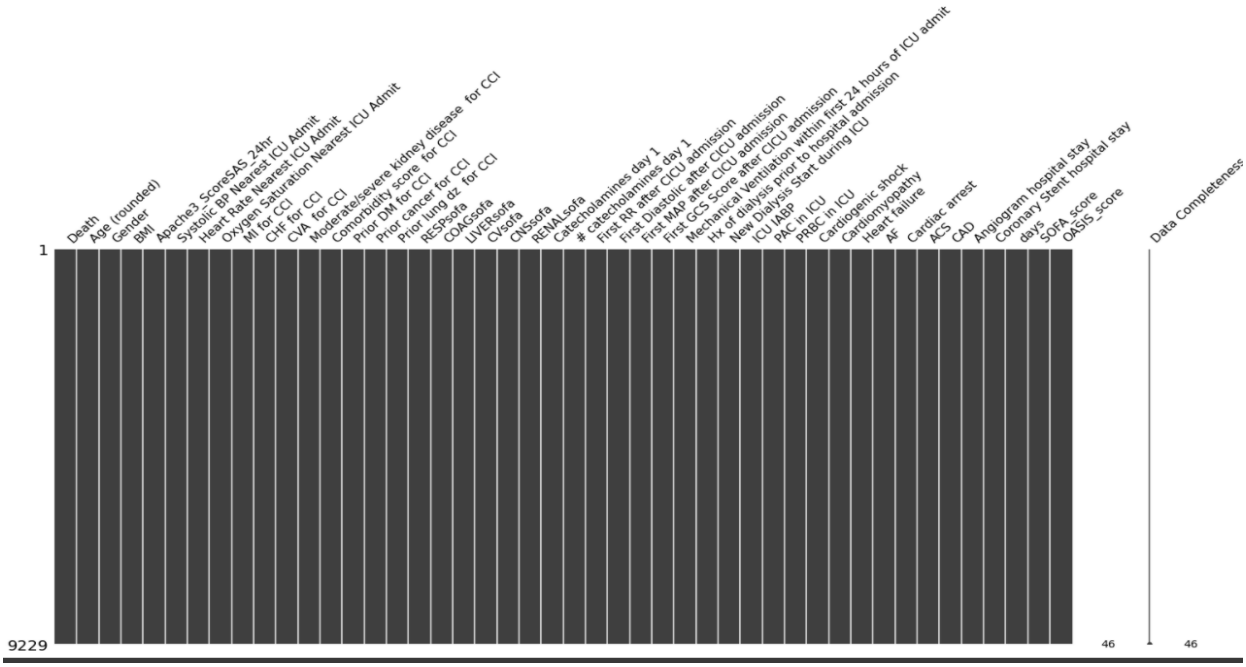
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6e76afbed0>
```



Plotting the missing values in the whole dataset by using missingno library. The maximun number of missing values were in the colums SOFAscore and LIVERsofa. These can be seen in below picture.



To calculate these missing values, we performed operations on the other columns and extracted the missing values from it. This operation can be seen in the code.x After cleaning the missing values we got around 9229 recored reduced from 9993 records.



After cleaning, csv file is exported as cleaned_sofa.csv

```
export_dataset(patients,'cleaned_sofa.csv')
```

Model-traing-sofa.ipynb

Open the "FINAL PROJECT/code/model-training-sofa.ipynb" notebook in the 8th step of the setup guide after following all 7 steps for data cleaning and model training of covid dataset.

Importing all the necessary libraries to run this code.

Libraries	Usage
Pandas	Used to load dataset
missingno	Format data into arrays for the input of the model built
datetime	Used to count the days form the given date
Sklearn	Used to train various models. Also used metrics to evaluate and compare the models.
Matplotlib	Used for visualization of the data and to generate various graphs and diagrams.
Keras	Used to train deep learning keras model
Seaborn	Used to plot various graphs
Os	Used to get path of the desired file from the operating system
Pickle	Store the trained model in the pickle file.
Urllib	Retrieve the dataset from the downloadable link

```
import pandas as pd
import missingno as msno
import os
import urllib.request
import matplotlib.pyplot as plt
import seaborn as sb
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_predict, train_test_split, cross_val_score
from sklearn import metrics
from sklearn.utils import resample
from sklearn.utils import shuffle
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import plot_roc_curve
from keras.models import Sequential
from keras.layers import Dense
from matplotlib import cm
from sklearn.metrics import roc_curve
from sklearn.metrics import auc
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
import pickle
```

After this cleaned_sofa.csv is imported in pandas

```
#read dataset into pandas
data = pd.read_csv('cleaned_sofa.csv')
data
```

Data cleaning and visualization:

Variable name	Details
data	Dataset covid.csv is loaded in this variable and can be manipulated without affecting the original csv file.
Binary_data	Columns having binary data is stored in this variable
numeric_data	Columns having numeric data is stored in this variable
sb	library seaborn imported as sb
pd	Pandas imported as pd
feature_names	List of features having numerical data
corr_matrix	Correlation matrix generated by using pearson coefficient between each columns of data
plt	Matplotlib imported as plt
scatter	Scatter plot for all numerical features to compare with each other and how much it affects the label 'Death'.

Logistic Regression

Variable name	Details
target	Name of the target column
X	records of dataset with all input columns
Y	records of dataset with the label column
X_train	Used for training model a subset of X
X_test	Used for testing model a subset of X
Y_train	Used for training model a subset of Y
Y_test	Used for testing model a subset of Y
logistic_regression	Logistic regression model initialized from the sklearn library
Y_pred_logistic	Predicted output of model for the X_test inputs
Y_pred_logistic_prob	Predicted probabilities of output of model for the X_test inputs
confusion_matrix	Confusion matrix for actual outputs(Y_test) and predicted outputs(Y_pred_logistic)
fpr_logistic	False positive rate for logistic regression model
tpr_logistic	True positive rate for logistic regression model
thresholds_logistic	Threshold values for logistic regression model
auc_logistic	Generating a curve for fpr_logistic and tpr_logistic

CalculateMetricsLogisticRegression() : It splits the dataset which is given as an input into 2 parts, i.e., inputs and labels. Then it split the whole dataset into training and testing parts, i.e., X_train, X_test, Y_train and Y_test. Then model is trained by training inputs and training labels, and then after, prediction is done on testing inputs. After that comparision of actual testing labels and predicted labels is done by

construction of confusion matrix and heatmap is plotted. Also there ROC curve is generated by using fpr(false positive rate) and tpr(true positive rate).

Sampling:

Variable name	Details
patients_majority	Columns of patients whose death = 0
patients_minority	Columns of Patients whose death = 1
patients_majority_length	Total number of patients whose death = 0
patients_sample	Sampled dummy patients
data_oversampled	Dummy samples of the patients whose length is matched with majority patients
data _undersampled	Patients after removing some patients to match length with minority patients
data _mixedsampled	Patients after removing some patients from majority and adding some dummy patients to minority patients.

```
#oversampling the data
patients_majority = data[data.Death==0]
patients_minority = data[data.Death==1]
patients_majority_length = len(patients_majority)

patients_sample = resample(patients_minority,
                                replace=True,     # sample without replacement
                                n_samples=patients_majority_length,    # to match majority class
                                random_state=123)


data_oversampled = pd.concat([patients_majority, patients_sample])
data_oversampled.Death.value_counts().plot(kind='bar')
```

Classification:

Variable name	Details
target	Name of the target column i.e. death
X	records of dataset with all input columns
Y	records of dataset with the label column
X_train	Used for training model a subset of X
X_test	Used for testing model a subset of X
Y_train	Used for training model a subset of Y
Y_test	Used for testing model a subset of Y
p_scores	Set to store precision of each model trained
r_scores	Set to store recall of each model trained
f1_scores	Set to store f1-score of each model trained
logistic_regression	Instance of logistic regression from skleran
Y_pred_logistic_prob	Predicted probabilities for X_test inputs for logistic regression
Y_pred_logistic	Predicted values for X_test inputs for logistic regression
confusion_matrix	Confusion matrix for actual vales(Y_test) and predicted outputs(Y_pred_logistic)
fpr_logistic	False positive rate for logistic regression model
tpr_logistic	True positive rate for logistic regression model

thresholds_logistic	Threshold values for logistic regression model
auc_logistic	Generating a ROC curve for fpr_logistic and tpr_logistic
decisionTree_classifier	Instance of Decision Tree Classifier from skleran
Y_pred_decision	Predicted values for X_test inputs for Decision Tree Classifier
Y_pred_decision_prob	Predicted probabilities for X_test inputs for Decision Tree Classifier
fpr_decision	False positive rate for Decision Tree Classifier
tpr_decision	True positive rate for Decision Tree Classifier
thresholds_decision	Threshold values for Decision Tree Classifier
auc_decision	Generating a ROC curve for fpr_decision and tpr_decision
random_forest_classifier	Instance of Random Forest Classifier from skleran
Y_pred_forest	Predicted values for X_test inputs for Random Forest Classifier
Y_pred_forest_prob	Predicted probabilities for X_test inputs for Random Forest Classifier
fpr_forest	False positive rate for Random Forest Classifier
tpr_forest	True positive rate for Random Forest Classifier
thresholds_forest	Threshold values for Random Forest Classifier
auc_forest	Generating a ROC curve for fpr_forest and tpr_forest
classification_model	Neural Network Classification model made by using keras Sequential Class
history	Save the model's progress during training such as metrics and loss function
Y_pred_keras	Predicted values for X_test inputs for Neural Network
Y_pred_keras_prob	Predicted probabilities for X_test inputs for Neural Network
fpr_keras	False positive rate for Neural Network
tpr_keras	True positive rate for Neural Network
thresholds_keras	Threshold values for Neural Network
auc_keras	Generating a ROC curve for fpr_keras and tpr_keras
model_names	List of names of all models
p_scores_list	List of precision scores of all models
r_scores_list	List of recall scores of all models
f1_scores_list	List of f-1 scores of all models
scores	Dataframe containing all the scores of all models to visualize it in table format

	Models	Precision	recall	f1 score
0	Logistic Regression	0.83	0.83	0.83
1	Decision Tree	0.93	1.0	0.96
2	Random Forest	0.97	1.0	0.99
3	Neural Network	0.89	0.99	0.93

Regression:

Variable name	Details
regression_data	Dataset extracted from positive_patients by only getting patients which died
scaler	
target	Name of the target column i.e. days
X	records of dataset with all input columns
Y	records of dataset with the label column
X_train	Used for training model a subset of X
X_test	Used for testing model a subset of X
Y_train	Used for training model a subset of Y
Y_test	Used for testing model a subset of Y
mae_scores	Set to store mean absolute error of each model trained
mse_scores	Set to store mean squared error of each model trained
r2_scores	Set to store r2-score of each model trained
linear_regression	Instance of linear regression from skleran
Y_pred	Predicted values for X_test inputs for regression
p1	Maximum value from Y_pred or Y_test used for plotting the graph
P2	Minimum value from Y_pred or Y_test used for plotting the graph
regr	Instance of Support Vector Regressor from skleran
clf	Instance of Decision Tree Regressor from skleran
reg	Instance of Gradient Boosting Regressor from skleran
rf_regr	Instance of Random Forest Regressor from skleran
model	Neural Network Regressor model made by using keras Sequential Class
history	Save the model's progress during training such as metrics and loss function
model_names	List of names of all models
mae_scores_list	List of mean absolute error scores of all models
mse_scores_list	List of mean squared error scores of all models
r2_scores_list	List of r-1 scores of all models
scores	Dataframe containing all the scores of all models to visualize it in table format

Flask Application:

To deploy and run the flask application, user must complete the steps mentioned in setup guide.

By going to the flask application folder, you can find 2 folders and 4 files which are required to run the project.

Model folder contains all the trained models for both the dataset and also it contains the pickle file for normalization.

Templates folder contains three html and css template files for the user Interface for flask application.

File name	Usage
covid_classification.pkl	Classification model for covid prediction
covid_regression.hdf5	Keras model for regression for covid prediction
covid_scaler.pkl	Normalization scaler for covid data
sofa_classification.pkl	Classification model for sofa prediction
sofa_regression.hdf5	Keras model for regression for sofa prediction
sofa_scaler.pkl	Normalization scaler for sofa data
Covid.html	Html page for covid prediction which contains form to get values from the user
Homepage.html	Html template for the home page of the flask application
sofaScore.html	Html page for sofa prediction which contains form to get values from the user

Backend of the flask application is in the main.py file.

Main.py file manages all the templates, models and takes input from the user.

Libraries	Usage
Flask	Necessary library to work with flask framework
Numpy	Format data into arrays for the input of the model built
Sklearn	Used to get Min-Max Normalizer to normalize the input data
Keras	Used for extracting the keras model which is required for the prediction of the Regression.
tensorflow	Used for prediction for the regression
Pickle	Retrieve the trained model from the pickle file.

```
import flask
import pickle
import pandas as pd
import numpy as np
import sys
import joblib
import tensorflow as tf
from keras.models import load_model
from sklearn.preprocessing import MinMaxScaler
"
```

All the models are imported in the application from pickle and .hdf5 files and stored in the variables.

Variables	Details
covid_c_model	Classification model for covid prediction
covid_scaler	Min-Max Normalizer for covid prediction
sofa_c_model	Classification model for sofa prediction
sofa_scaler	Min-Max Normalizer for sofa prediction
covid_r_model	Classification model for sofa prediction
sofa_r_model	Regression model for sofa prediction

After importing models, inputs are taken from the user and stored in appropriate variables, and then after those inputs are converted to normalized data by using Min-Max normalizers which are also imported from the pickle files and that normalized inputs are stored in 'normalized_data' variable, which will be used as an input for the model.

Variables	Details
input_variables	Input from the user stored as the dataframe by using pandas
death	Prediction from the classification model based on input_variables
normalized_data	Normalized data after using scaler to normalize the input data
days	Prediction from the regression model
d	Number of days from the 'days' variable
h	Number of hours from the 'days' variable
message	Variable to display appropriate message based on prediction done.

If the output of the classification variable is 1, i.e., if the patient is going to die, then only it will perform the regression prediction, as number of days cannot counted for the patients who are safe.