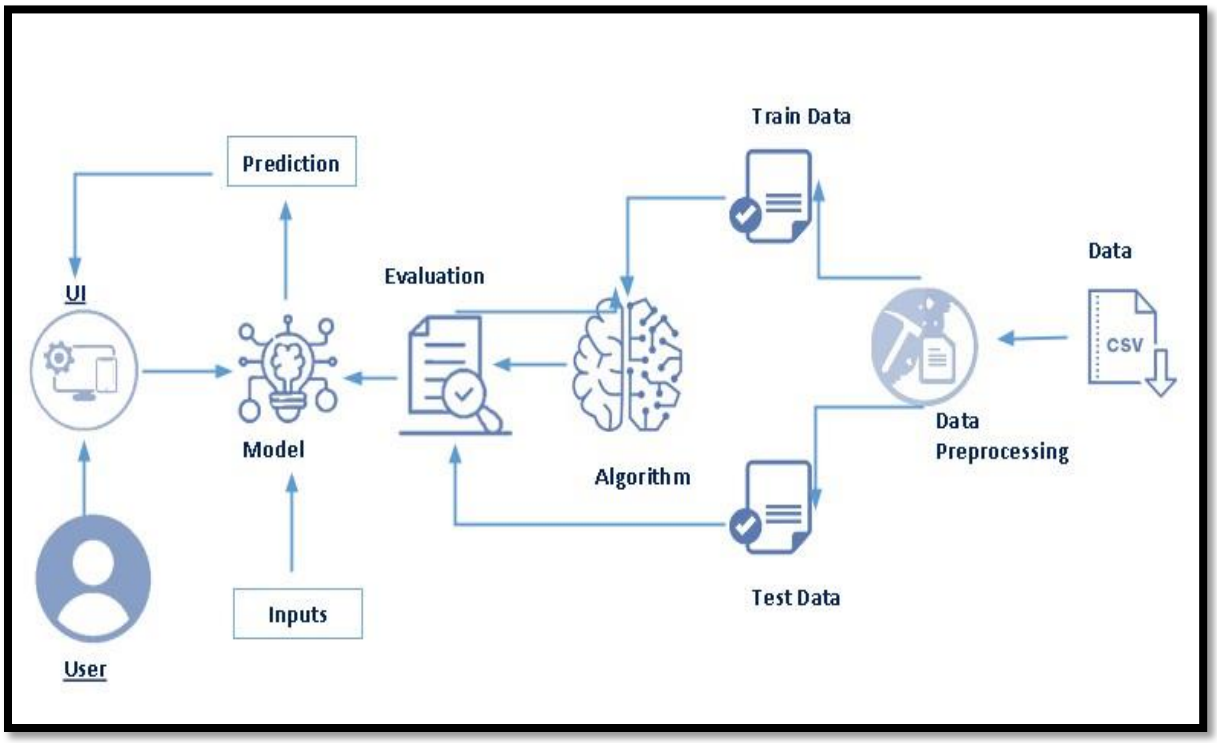
Perinatal health risk predictors using artificial intelligence

Project Description:

Advances in public health and medical care have enabled better pregnancy and birth outcomes. The rates of perinatal health indicators such as maternal mortality and morbidity; fetal, neonatal, and infant mortality; low birthweight; and preterm birth have reduced over time. However, they are still a public health concern, and considerable disparities exist within and between countries. For perinatal researchers who are engaged in unraveling the tangled web of causation for maternal and child health outcomes and for clinicians involved in the care of pregnant women and infants, artificial intelligence offers novel approaches to prediction modeling, diagnosis, early detection, and monitoring in perinatal health. Machine learning, a commonly used artificial intelligence method, has been used to predict preterm birth, birthweight, preeclampsia, mortality, hypertensive disorders, and postpartum depression. Real-time electronic health recording and predictive modeling using artificial intelligence have found early success in fetal monitoring and monitoring of women with gestational diabetes especially in low-resource settings. Artificial intelligence–based methodologies have the potential to improve prenatal diagnosis of birth defects and outcomes in assisted reproductive technology too.

Technical Architecture:



Pre requisites:

To complete this project, you must require following software's, concepts and packages

- **Anaconda navigator and spider:**
  - Refer the link below to download anaconda navigator
  - Link : https://youtu.be/1ra4zH2G4o0
- **Python packages:**
  - Open anaconda prompt as administrator
  - Type"pipinstallnumpy"andclick enter.
  - Type"pipinstallpandas"andclickenter.
  - Type"pipinstall matplotlib"andclickenter.
  - Type"pipinstallpickle-mixin"andclickenter.
  - Type"pipinstallseaborn"andclickenter.
  - Type"pipinstallFlask"andclickenter.

Prior Knowledge:

You must have prior knowledge of following topics to complete this project.

- **ML Concepts**
  - Supervised learning: https://www.javatpoint.com/supervised-machine-learning
  - Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning
  - Classification
  - Decision tree: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

    - Random forest: https://www.javatpoint.com/machine-learning-random-forest-algorithm
    - KNN: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
    - Xgboost: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/
    - Evaluation metrics: https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/
- **Flask Basics** : https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Objectives:

By the end of this project you will:

- Know fundamental concepts and techniques used for machine learning.
- Gain a broad understanding about data.
- Have knowledge on pre-processing the data/transformation techniques on outlier and some visualization concepts.

Project Flow:

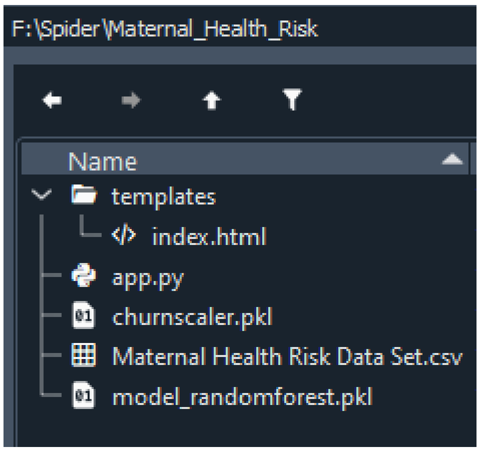
- User interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

- Data collection
    - Collect the dataset or create the dataset
- Visualizing and analyzing data
    - Univariate analysis
    - Bivariate analysis
    - Multivariate analysis
    - Descriptive analysis
- Data pre-processing
    - Checking for null values
    - Handling outlier
    - Handling categorical data
    - Splitting data into train and test
- Model building
    - Import the model building libraries
    - Initializing the model
    - Training and testing the model
    - Evaluating performance of model
    - Save the model
- Application Building
    - Create an HTML file
    - Build python code

Project Structure:

Create the Project folder which contains files as shown below


F:\Spider\Maternal_Health_Risk

Name
- templates
  - index.html
- app.py
- churnscaler.pkl
- Maternal Health Risk Data Set.csv
- model_randomforest.pkl

- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- Model.pkl is our saved model. Further we will use this model for flask integration.
- Training folder contains model training files and training_ibm folder contains IBM deployment files.

Milestone 1: Data Collection

ML depends heavily on data, It is most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

Activity 1: Download the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used Maternal Health Risk Data Set.csvdata. This data is downloaded from UC Irvine Machine Learning Repository. Please refer the link given below to download the dataset.

Link:https://archive.ics.uci.edu/ml/datasets/Maternal+Health+Risk+Data+Set

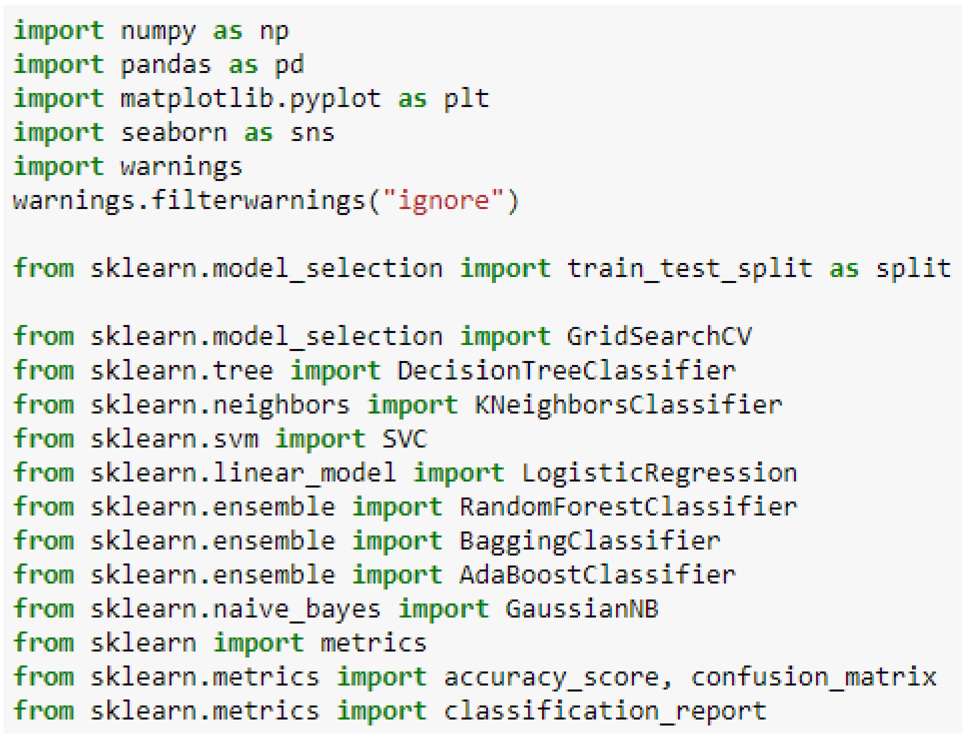
Milestone 2: Visualizing and analysing the data

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analysing techniques.

Note: There is n number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1: Importing the libraries

Import the necessary libraries as shown in the image.

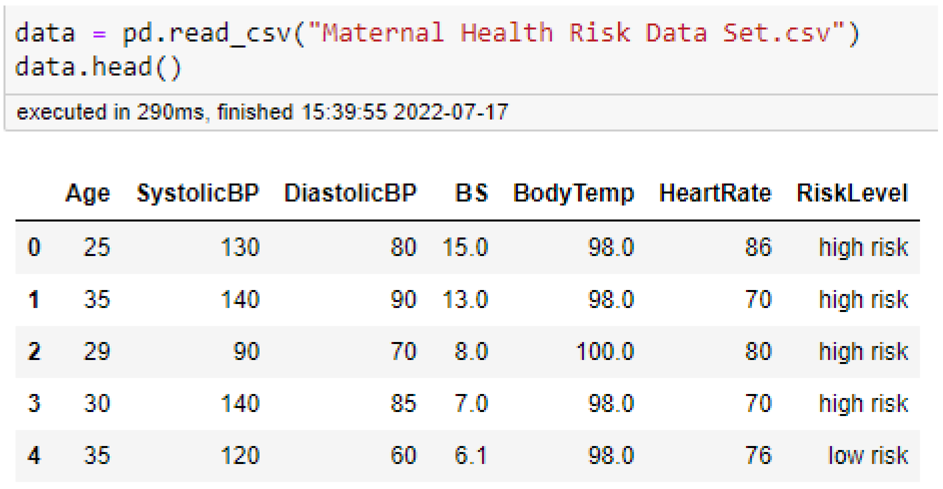


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")

from sklearn.model_selection import train_test_split as split

from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import classification_report
```

Activity 2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of csv file.



```
data = pd.read_csv("Maternal Health Risk Data Set.csv")
data.head()
executed in 290ms, finished 15:39:55 2022-07-17
```

	Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate	RiskLevel
0	25	130	80	15.0	98.0	86	high risk
1	35	140	90	13.0	98.0	70	high risk
2	29	90	70	8.0	100.0	80	high risk
3	30	140	85	7.0	98.0	70	high risk
4	35	120	60	6.1	98.0	76	low risk

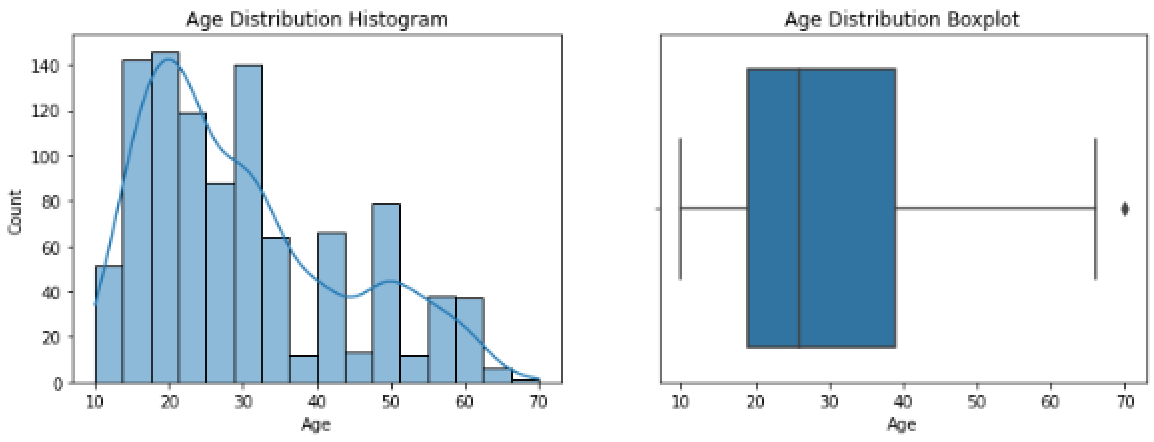
Activity 3: Univariate analysis

In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as distplot and countplot.

- Seaborn package provides a wonderful function histplot.and boxplotWith the help of histplot, and boxplot we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.

```
def num_plot(data, col):
    fig, ax = plt.subplots(1, 2, figsize=(12, 4))

    sns.histplot(data=data, x=col, kde=True, ax=ax[0])
    sns.boxplot(data=data, x=col, ax=ax[1])
    ax[0].set_title(f"{col} Distribution Histogram")
    ax[1].set_title(f"{col} Distribution Boxplot")

    plt.show()
```
executed in 11ms, finished 15:39:59 2022-07-17

```
# Age
num_plot(data, "Age")
```
executed in 1.04s, finished 15:40:00 2022-07-17

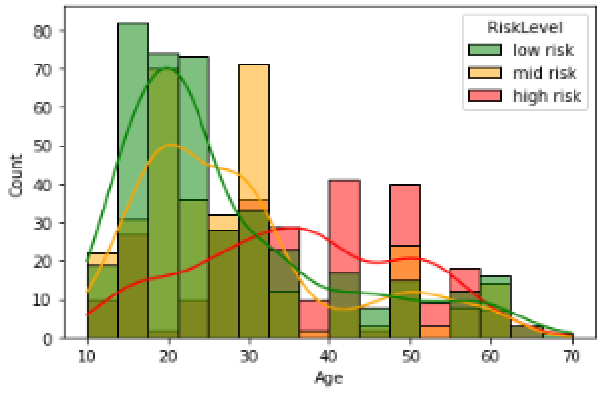


Activity 4: Bivariate analysis

To find the relation between two features we use bivariate analysis. Here we are visualizing the relationship between drug & BP, drug & sex and drug & cholesterol.

- Histplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.
- From the below plot you can understand that as Age increases the patients Risk level increases.

```
sns.histplot(data=data, x="Age", hue="RiskLevel", kde=True, hue_order=risk_order, palette=p_colors)
plt.show()
```
executed in 646ms, finished 15:40:23 2022-07-17



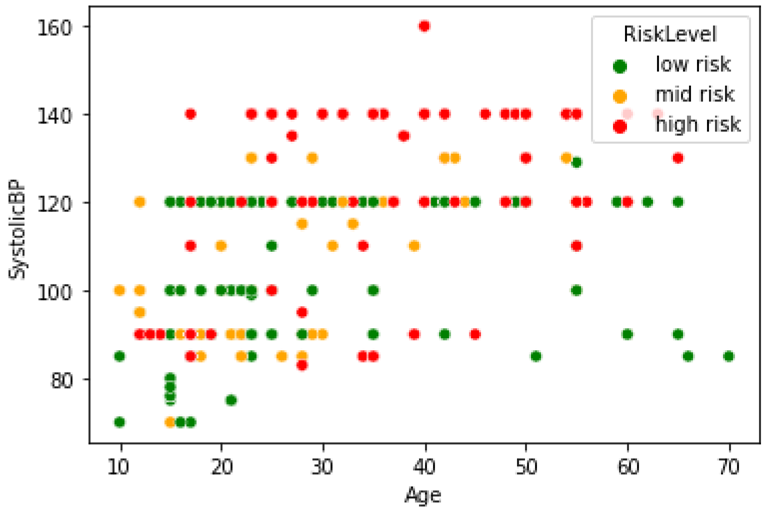
With the help of age feature we are creating an age interval and finding the relation between RiskLevel feature and age interval feature. Function histplot is used to find the relationship.

From the above image we get a clear understanding, RiskLevelincreases only for patients above age 30 years.

Activity 5: Multivariate analysis

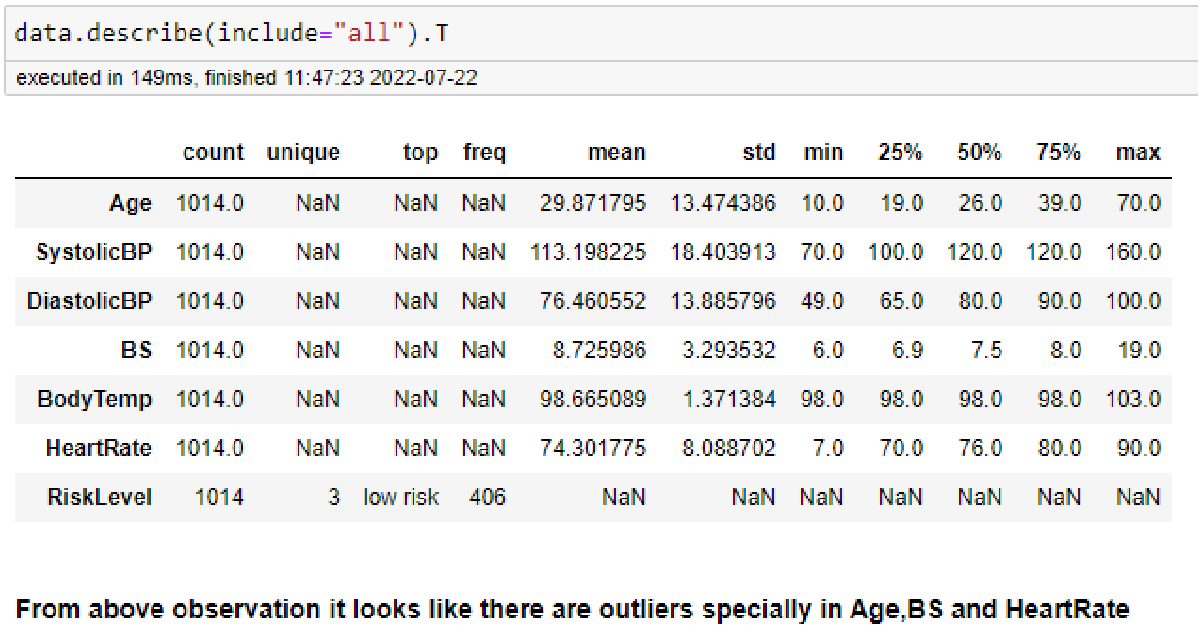
In simple words, multivariate analysis is to find the relation between multiple features. Here we have usedscatterplotfrom seaborn package.From the below image, we came to a conclusion Pregnant women with high SystolicBP seems to have a high health risk, regardless of their age.

-



Activity 6: Descriptive analysis

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.



```
data.describe(include="all").T
executed in 149ms, finished 11:47:23 2022-07-22
```

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Age	1014.0	NaN	NaN	NaN	29.871795	13.474386	10.0	19.0	26.0	39.0	70.0
SystolicBP	1014.0	NaN	NaN	NaN	113.198225	18.403913	70.0	100.0	120.0	120.0	160.0
DiastolicBP	1014.0	NaN	NaN	NaN	76.460552	13.885796	49.0	65.0	80.0	90.0	100.0
BS	1014.0	NaN	NaN	NaN	8.725986	3.293532	6.0	6.9	7.5	8.0	19.0
BodyTemp	1014.0	NaN	NaN	NaN	98.665089	1.371384	98.0	98.0	98.0	98.0	103.0
HeartRate	1014.0	NaN	NaN	NaN	74.301775	8.088702	7.0	70.0	76.0	80.0	90.0
RiskLevel	1014	3	low risk	406	NaN	NaN	NaN	NaN	NaN	NaN	NaN

From above observation it looks like there are outliers specially in Age,BS and HeartRate

Milestone 3: Data Pre-processing

As we have understood how the data is lets pre-process the collected data.

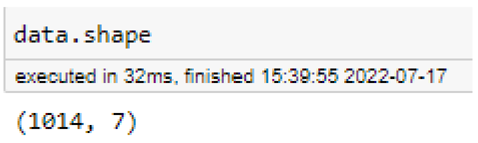
The download data set is not suitable for training the machine learning model as it might have so much of randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling categorical data
- Handling outliers
- Scaling Techniques
- Splitting dataset into training and test set

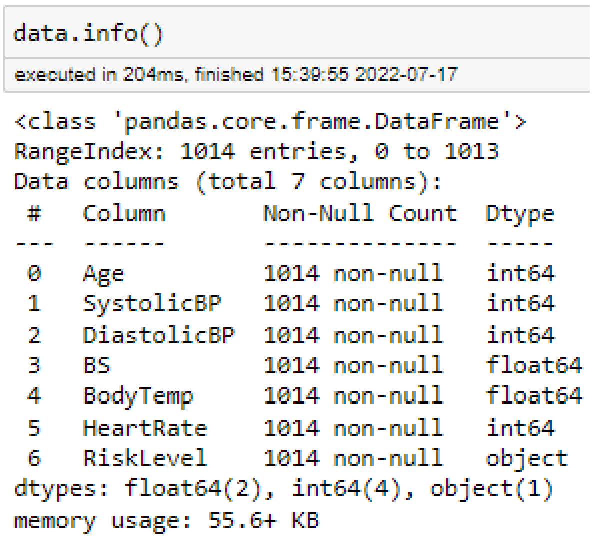
Note: These are the general steps of pre-processing the data before using it for machinelearning. Depending on the condition of your dataset, you may or may not have togo through all these steps.

Activity 1: Checking for null values

- Let’s find the shape of our dataset first, To find the shape of our data, df.shape method is used. To find the data type, df.info() function is used.



```
data.shape
executed in 32ms, finished 15:39:55 2022-07-17
(1014, 7)
```

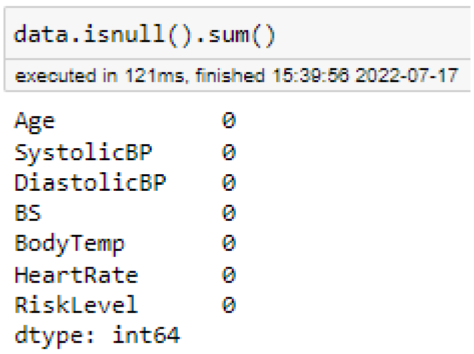


```
data.info()
```

executed in 204ms, finished 15:39:55 2022-07-17

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1014 entries, 0 to 1013
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Age          1014 non-null   int64  
 1   SystolicBP   1014 non-null   int64  
 2   DiastolicBP  1014 non-null   int64  
 3   BS           1014 non-null   float64
 4   BodyTemp     1014 non-null   float64
 5   HeartRate    1014 non-null   int64  
 6   RiskLevel    1014 non-null   object 
dtypes: float64(2), int64(4), object(1)
memory usage: 55.6+ KB
```

- For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function to it. From the below image we found that there are no null values present in our dataset. So we can skip handling of missing values step.



```
data.isnull().sum()
```

executed in 121ms, finished 15:39:56 2022-07-17

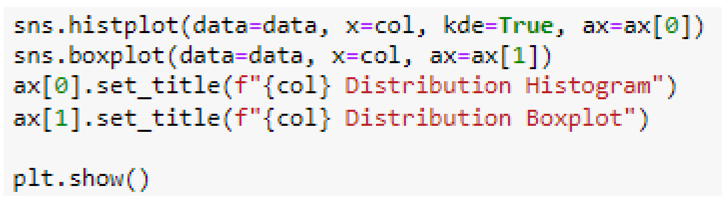
```
Age            0
SystolicBP     0
DiastolicBP    0
BS             0
BodyTemp       0
HeartRate      0
RiskLevel      0
dtype: int64
```

Let’s look for any outliers in the dataset

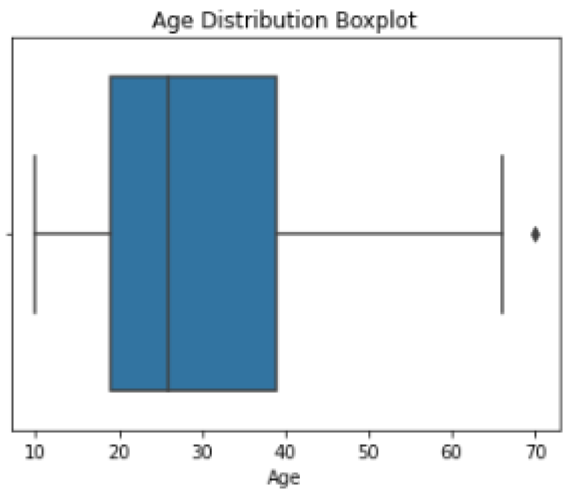
Activity 2: Handling outliers

With the help of boxplot, outliers are visualized. And here we are going to find upper bound and lower bound of all feature with some mathematical formula.

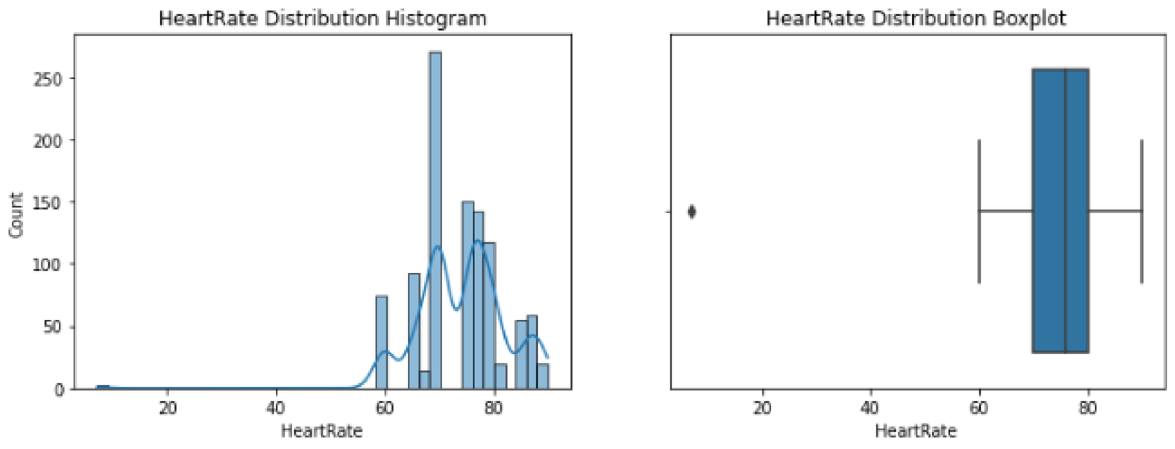
- From the below diagram, we could visualize that age feature has outliers. Boxplot from seaborn library is used here.



```
sns.histplot(data=data, x=col, kde=True, ax=ax[0])
sns.boxplot(data=data, x=col, ax=ax[1])
ax[0].set_title(f"{col} Distribution Histogram")
ax[1].set_title(f"{col} Distribution Boxplot")

plt.show()
```



- After looking for outliers in all features we see that most of the data is practically possible other than HeartRate. The row containing the outlier is removed.

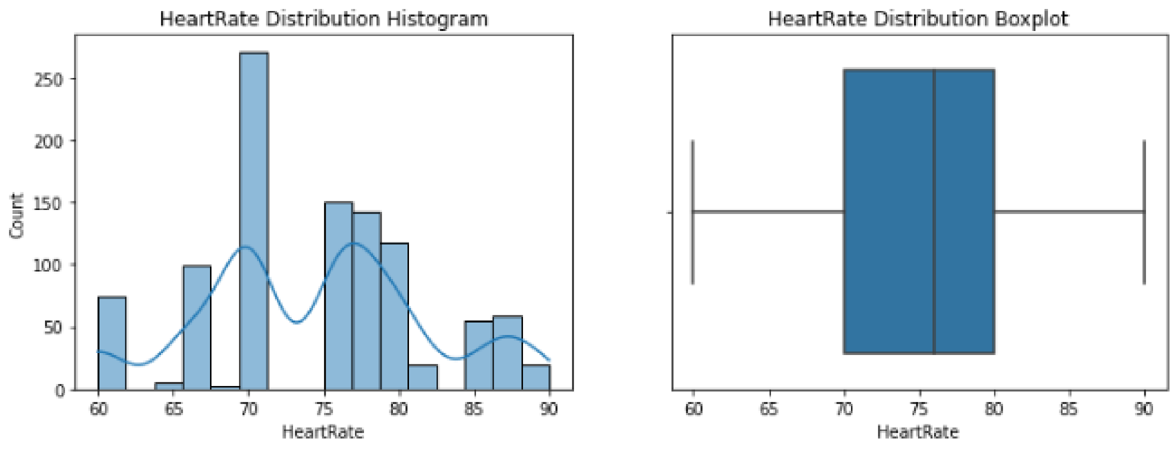


```
data= data.drop(data.index[data.HeartRate == 7])
```

```
executed in 1.46s, finished 15:40:05 2022-07-17
```

```
# HeartRate

num_plot(data, "HeartRate")
```

```
executed in 1.60s, finished 15:40:06 2022-07-17
```



Activity 3: Handling Categorical Values

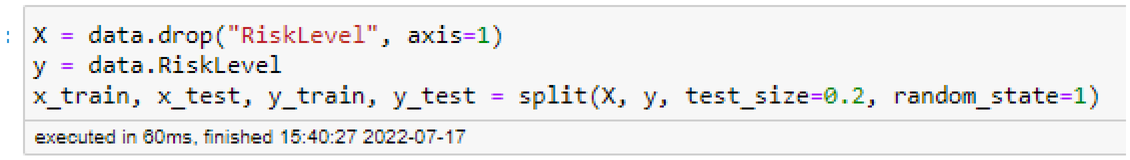
Since in our data the only categorical feature is dependent feature and all independent features are numerical handling he categorical values is not necessary in this model.

Activity 4: Splitting data into train and test

Now let's split the Dataset into train and test sets

Changes: first split the dataset into x and y and then split the data set

Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.



```
X = data.drop("RiskLevel", axis=1)
y = data.RiskLevel
x_train, x_test, y_train, y_test = split(X, y, test_size=0.2, random_state=1)
executed in 60ms, finished 15:40:27 2022-07-17
```

Milestone 4: Model Building

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. for this project we are applying eight classification algorithms. The best model is saved based on its performance.

1: Decision tree model

2: KNN model

3: SVC model

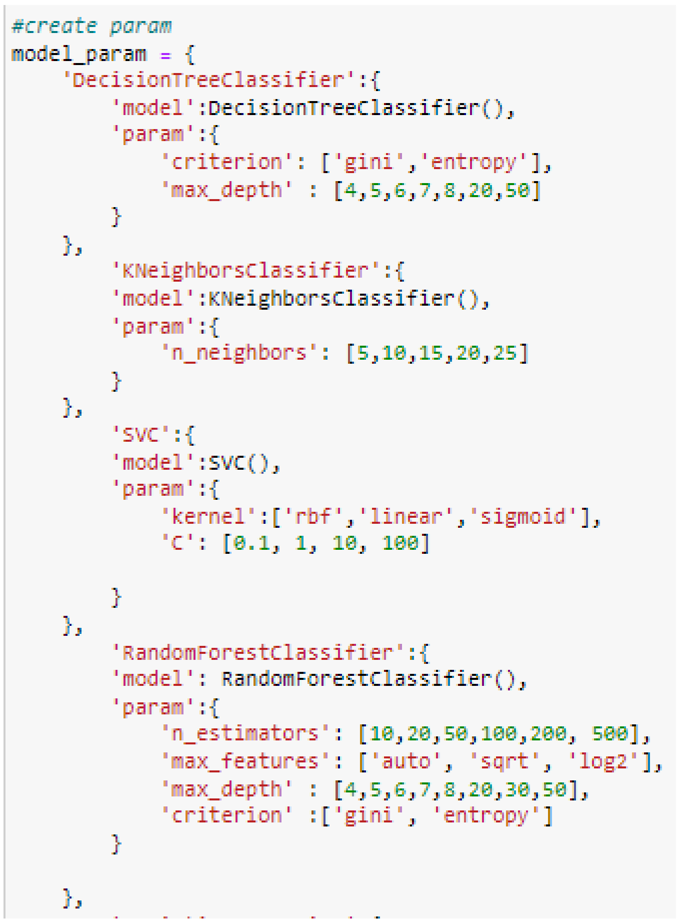
4: Random forest model

5: LogisticRegression

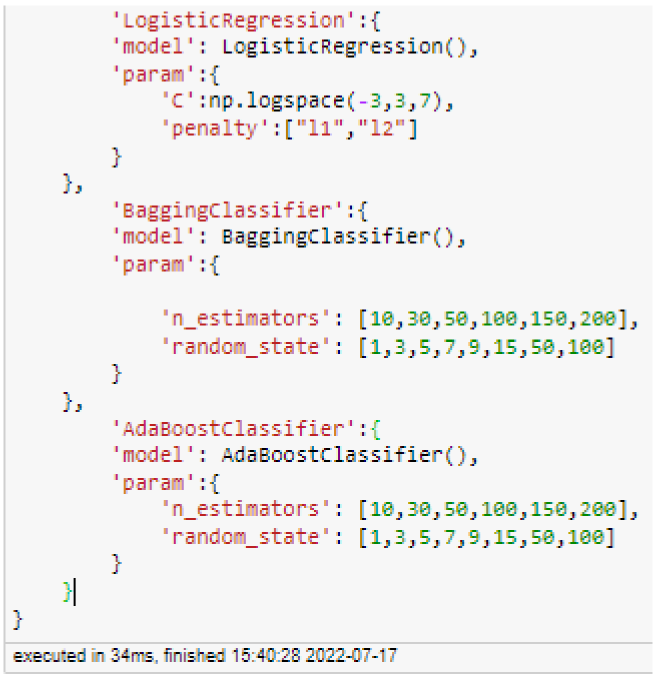
6: BaggingClassifier

7: AdaBoostClassifier

8: Naive bayes

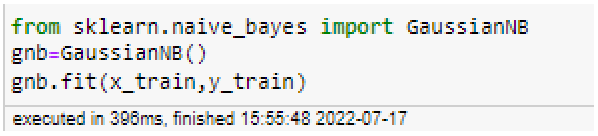


```
#create param
model_param = {
    'DecisionTreeClassifier':{
        'model':DecisionTreeClassifier(),
        'param':{
            'criterion': ['gini','entropy'],
            'max_depth' : [4,5,6,7,8,20,50]
        }
    },
        'KNeighborsClassifier':{
        'model':KNeighborsClassifier(),
        'param':{
            'n_neighbors': [5,10,15,20,25]
        }
    },
        'SVC':{
        'model':SVC(),
        'param':{
            'kernel':['rbf','linear','sigmoid'],
            'C': [0.1, 1, 10, 100]

        }
    },
        'RandomForestClassifier':{
        'model': RandomForestClassifier(),
        'param':{
            'n_estimators': [10,20,50,100,200, 500],
            'max_features': ['auto', 'sqrt', 'log2'],
            'max_depth' : [4,5,6,7,8,20,30,50],
            'criterion' :['gini', 'entropy']
        }

    },
```



```
        'LogisticRegression':{
        'model': LogisticRegression(),
        'param':{
            'C':np.logspace(-3,3,7),
            'penalty':["l1","l2"]
        }
    },
        'BaggingClassifier':{
        'model': BaggingClassifier(),
        'param':{

            'n_estimators': [10,30,50,100,150,200],
            'random_state': [1,3,5,7,9,15,50,100]
        }
    },
        'AdaBoostClassifier':{
        'model': AdaBoostClassifier(),
        'param':{
            'n_estimators': [10,30,50,100,150,200],
            'random_state': [1,3,5,7,9,15,50,100]
        }
    }
}
```

executed in 34ms, finished 15:40:28 2022-07-17



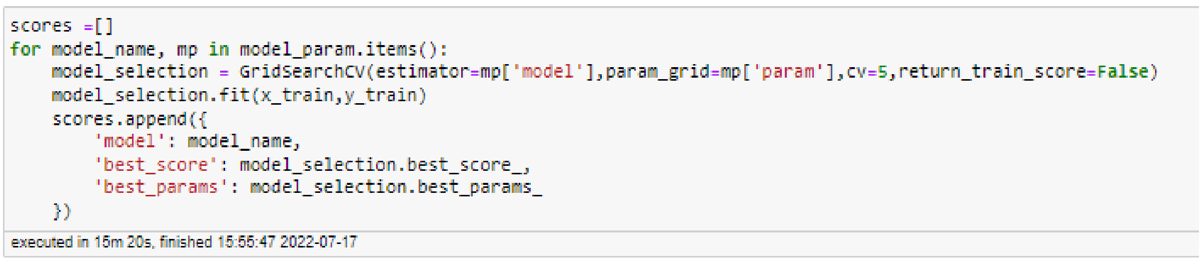
```
from sklearn.naive_bayes import GaussianNB
gnb=GaussianNB()
gnb.fit(x_train,y_train)
```

executed in 396ms, finished 15:55:48 2022-07-17

Now let's see the performance of all the models and save the best model

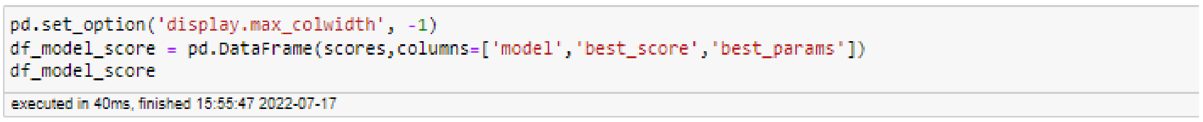
Compare the model

For comparing the above eight modelsGridSearchCV function is defined.



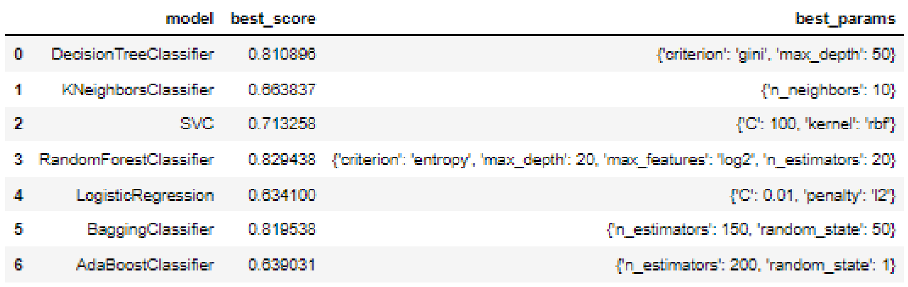
```
scores =[]
for model_name, mp in model_param.items():
    model_selection = GridSearchCV(estimator=mp['model'],param_grid=mp['param'],cv=5,return_train_score=False)
    model_selection.fit(x_train,y_train)
    scores.append({
        'model': model_name,
        'best_score': model_selection.best_score_,
        'best_params': model_selection.best_params_
    })
```

```
executed in 15m 20s, finished 15:55:47 2022-07-17
```



```
pd.set_option('display.max_colwidth', -1)
df_model_score = pd.DataFrame(scores,columns=['model','best_score','best_params'])
df_model_score
```

```
executed in 40ms, finished 15:55:47 2022-07-17
```



	model	best_score	best_params
0	DecisionTreeClassifier	0.810896	{'criterion': 'gini', 'max_depth': 50}
1	KNeighborsClassifier	0.663837	{'n_neighbors': 10}
2	SVC	0.713258	{'C': 100, 'kernel': 'rbf'}
3	RandomForestClassifier	0.829438	{'criterion': 'entropy', 'max_depth': 20, 'max_features': 'log2', 'n_estimators': 20}
4	LogisticRegression	0.634100	{'C': 0.01, 'penalty': 'l2'}
5	BaggingClassifier	0.819538	{'n_estimators': 150, 'random_state': 50}
6	AdaBoostClassifier	0.639031	{'n_estimators': 200, 'random_state': 1}

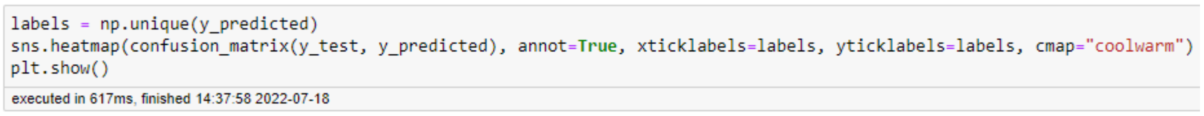
After calling the function, the results of models are displayed as output. From the eight model random forest and is performing well. From the below image, We can see the accuracy of the model. Models have 82.9% accuracy. Even confusion matrix also have same results. Training time of decision tree is faster than random forest. In such case we have to select decision tree model (time saving & cost wise profitable). But, here random forest is selected and evaluated with cross validation. Additionally, we can tune the model with hyper parameter tuning techniques.

Evaluating performance of the model and saving the model

From sklearn.metrics: confusion_matrix andclassification_report is used to evaluate the score of the model. On the parameters, we have given y_test,y_predictedfor confusion_matrix and y_test, y_predicted, target_names=target_names for classification_report.

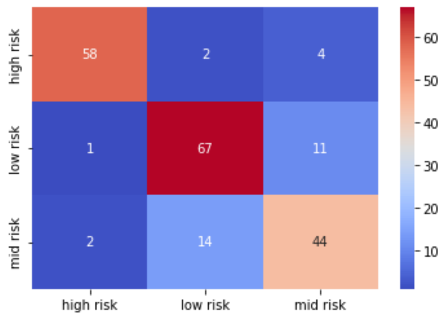
Our model is performing well. So, we are saving the model by pickle.dump().

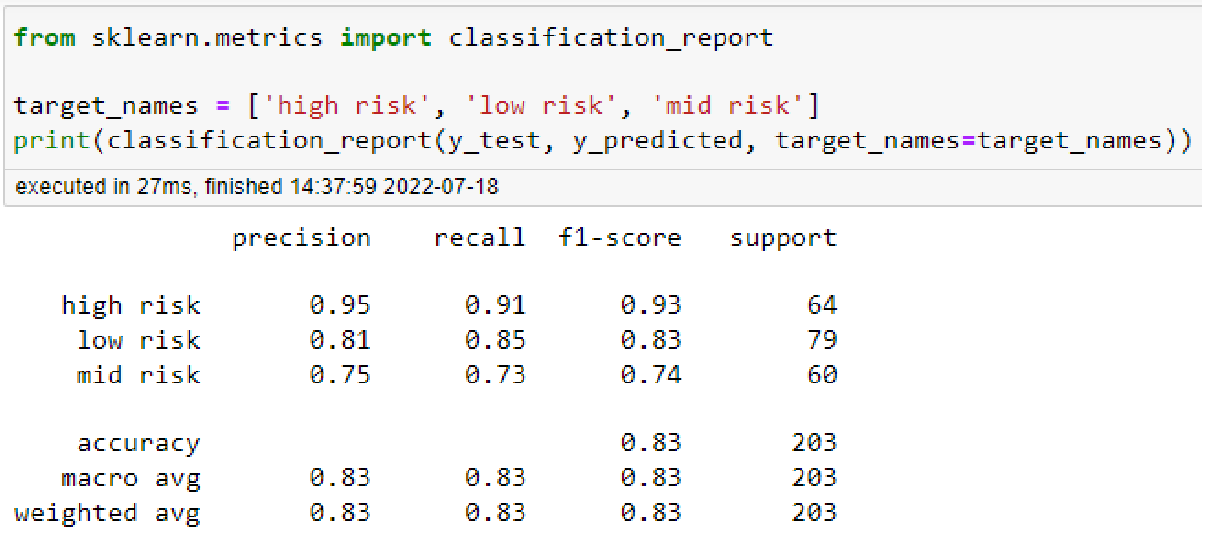
.



```
labels = np.unique(y_predicted)
sns.heatmap(confusion_matrix(y_test, y_predicted), annot=True, xticklabels=labels, yticklabels=labels, cmap="coolwarm")
plt.show()
```

executed in 617ms, finished 14:37:58 2022-07-18

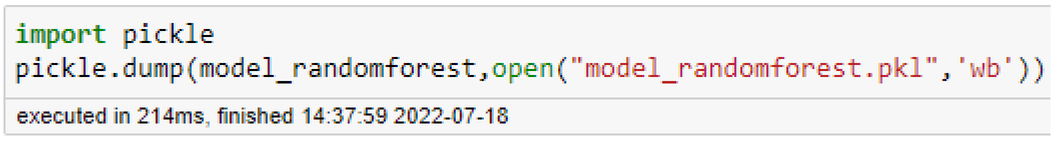




```
from sklearn.metrics import classification_report

target_names = ['high risk', 'low risk', 'mid risk']
print(classification_report(y_test, y_predicted, target_names=target_names))
```

executed in 27ms, finished 14:37:59 2022-07-18

```
              precision    recall  f1-score   support

   high risk       0.95      0.91      0.93        64
    low risk       0.81      0.85      0.83        79
    mid risk       0.75      0.73      0.74        60

    accuracy                           0.83       203
   macro avg       0.83      0.83      0.83       203
weighted avg       0.83      0.83      0.83       203
```



```
import pickle
pickle.dump(model_randomforest,open("model_randomforest.pkl",'wb'))
```

executed in 214ms, finished 14:37:59 2022-07-18

Milestone 5: Application Building

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building serverside script

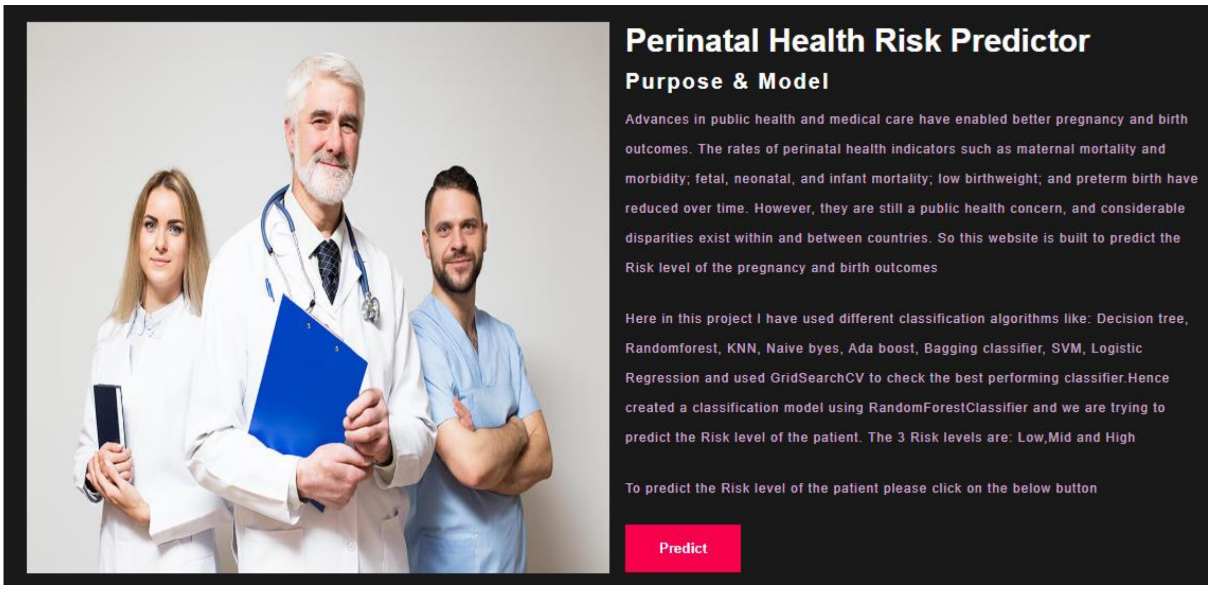
Activity1: Building Html Pages:

For this project create three HTML files namely

- home.html
- predict.html
- submit.html

and save them in templates folder.

Let’s see how our about.html page looks like:



Perinatal Health Risk Predictor

Purpose & Model

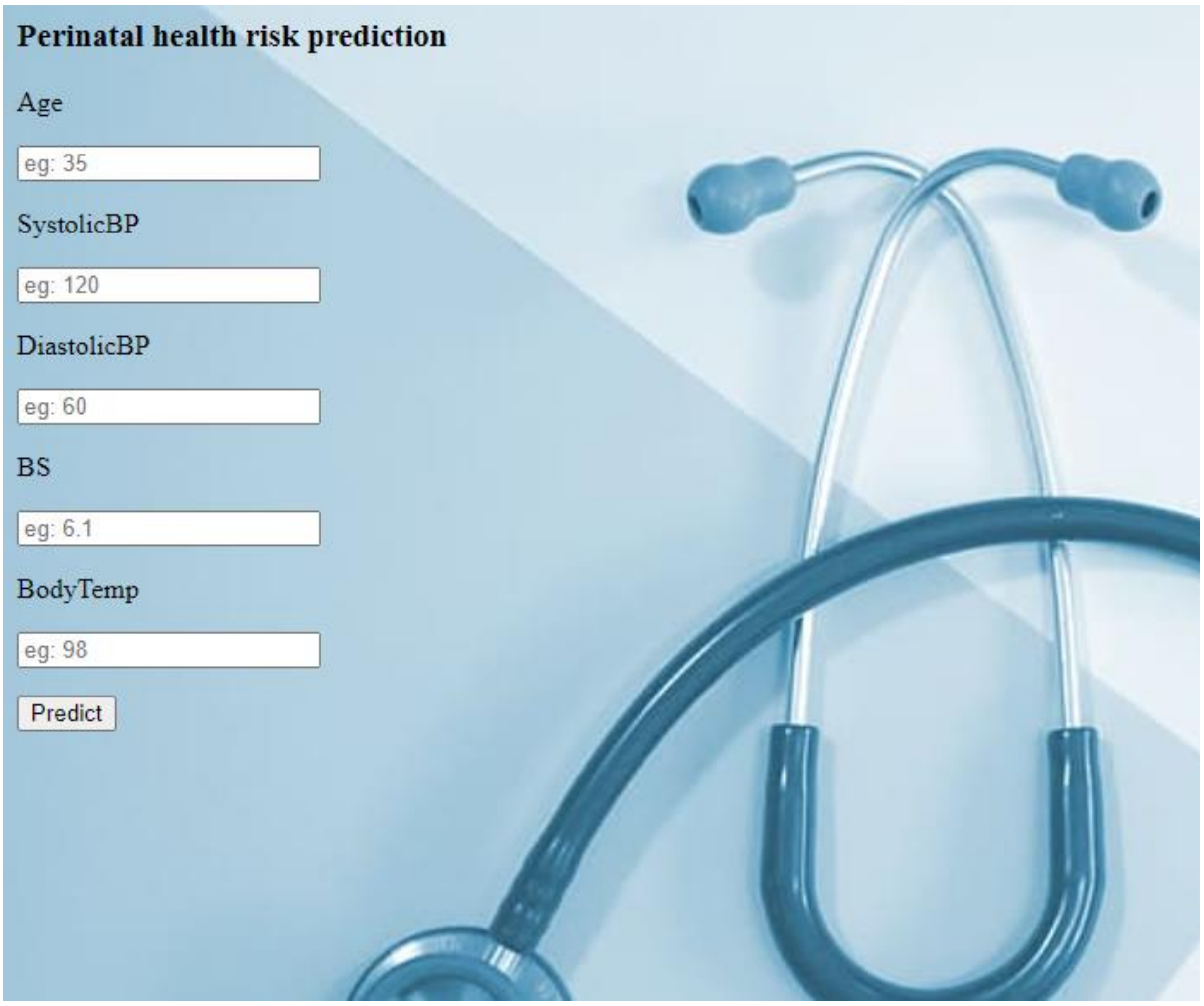
Advances in public health and medical care have enabled better pregnancy and birth outcomes. The rates of perinatal health indicators such as maternal mortality and morbidity; fetal, neonatal, and infant mortality; low birthweight; and preterm birth have reduced over time. However, they are still a public health concern, and considerable disparities exist within and between countries. So this website is built to predict the Risk level of the pregnancy and birth outcomes

Here in this project I have used different classification algorithms like: Decision tree, Randomforest, KNN, Naive byes, Ada boost, Bagging classifier, SVM, Logistic Regression and used GridSearchCV to check the best performing classifier.Hence created a classification model using RandomForestClassifier and we are trying to predict the Risk level of the patient. The 3 Risk levels are: Low,Mid and High

To predict the Risk level of the patient please click on the below button

Now when you click on predict button you will get redirected to predict.html

Lets look how our index.html file looks like:



Perinatal health risk prediction

Age

eg: 35

SystolicBP

eg: 120

DiastolicBP

eg: 60

BS

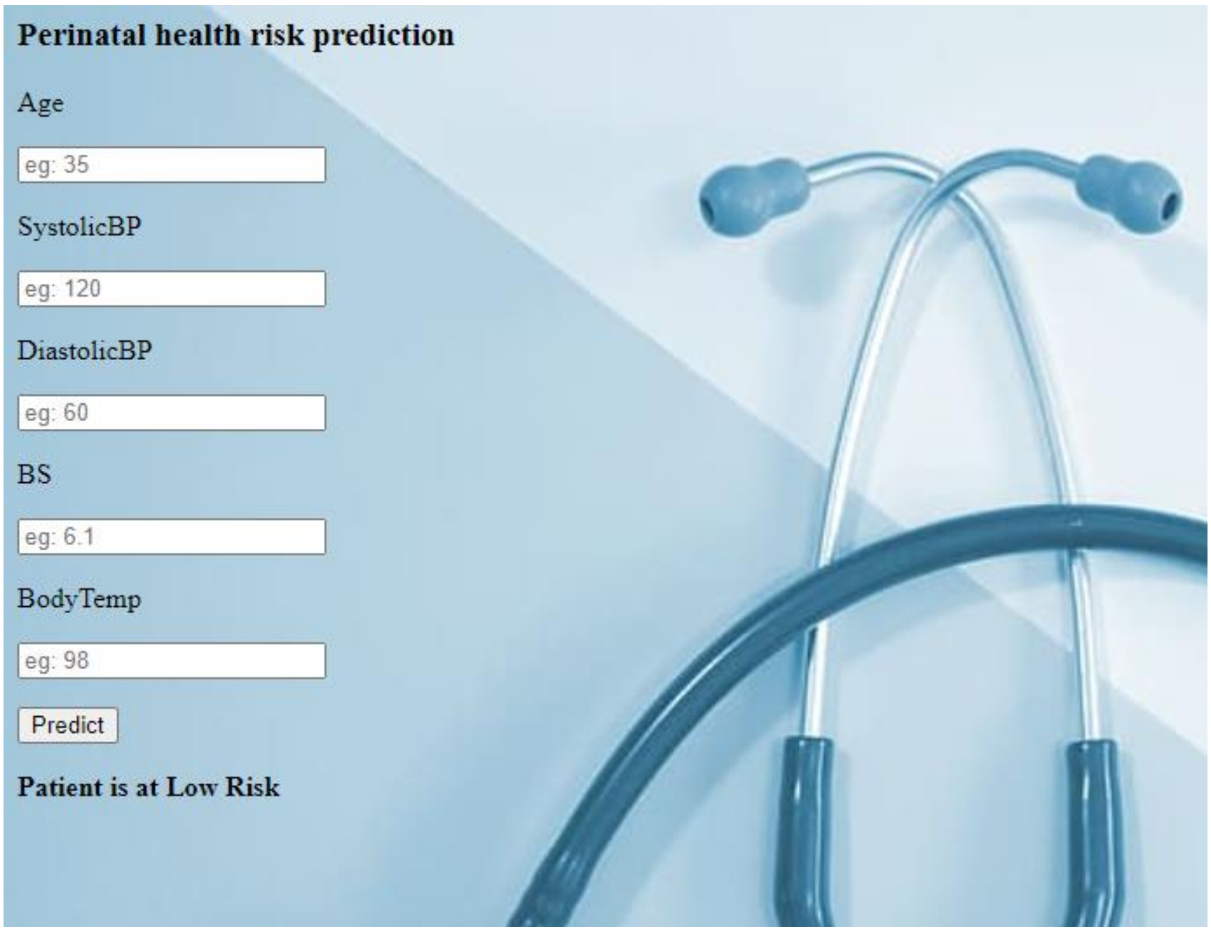
eg: 6.1

BodyTemp

eg: 98

Now when you click on Predict button from left bottom corner you will get redirected to /y_predict

Lets look how our submit.html file looks like:


Perinatal health risk prediction

Age

eg: 35

SystolicBP

eg: 120

DiastolicBP

eg: 60

BS

eg: 6.1

BodyTemp

eg: 98

Predict

Patient is at Low Risk

Activity 2: Build Python code:

Import the libraries

```python
from flask import Flask, render_template,request
import pickle
```

Load the saved model. Importing flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument.

```python
model=pickle.load(open("model_randomforest.pkl","rb"))
app=Flask(__name__)
```

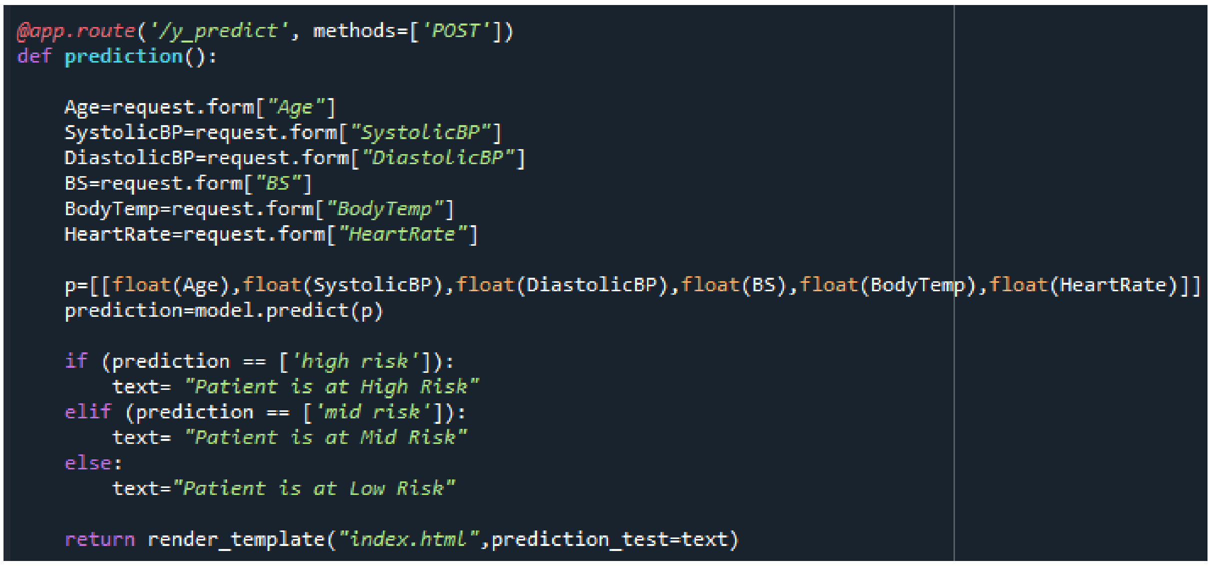
Render HTML page:

```python
@app.route('/')
def loadpage():
    return render_template('index.html')
```

Here we will be using declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with home.html function. Hence, when the home page of the web server is opened in browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

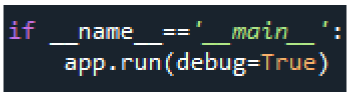
Retrieves the value from UI:



```python
@app.route('/y_predict', methods=['POST'])
def prediction():

    Age=request.form["Age"]
    SystolicBP=request.form["SystolicBP"]
    DiastolicBP=request.form["DiastolicBP"]
    BS=request.form["BS"]
    BodyTemp=request.form["BodyTemp"]
    HeartRate=request.form["HeartRate"]

    p=[[float(Age),float(SystolicBP),float(DiastolicBP),float(BS),float(BodyTemp),float(HeartRate)]]
    prediction=model.predict(p)

    if (prediction == ['high risk']):
        text= "Patient is at High Risk"
    elif (prediction == ['mid risk']):
        text= "Patient is at Mid Risk"
    else:
        text="Patient is at Low Risk"

    return render_template("index.html",prediction_test=text)
```

Here we are routing our app to y_predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will rendered to the text that we have mentioned in the submit.html page earlier.

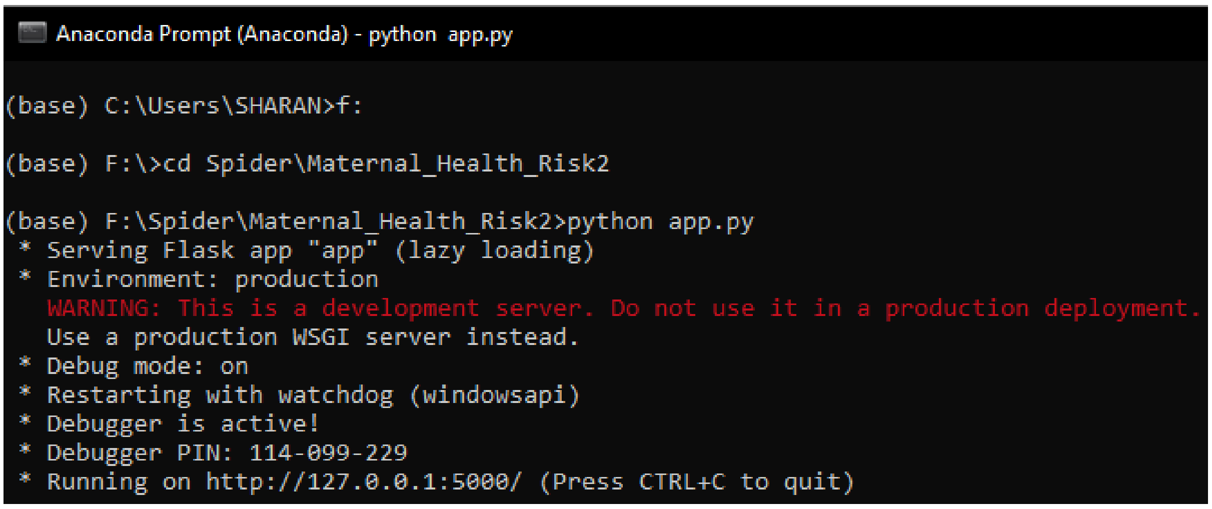
Main Function:



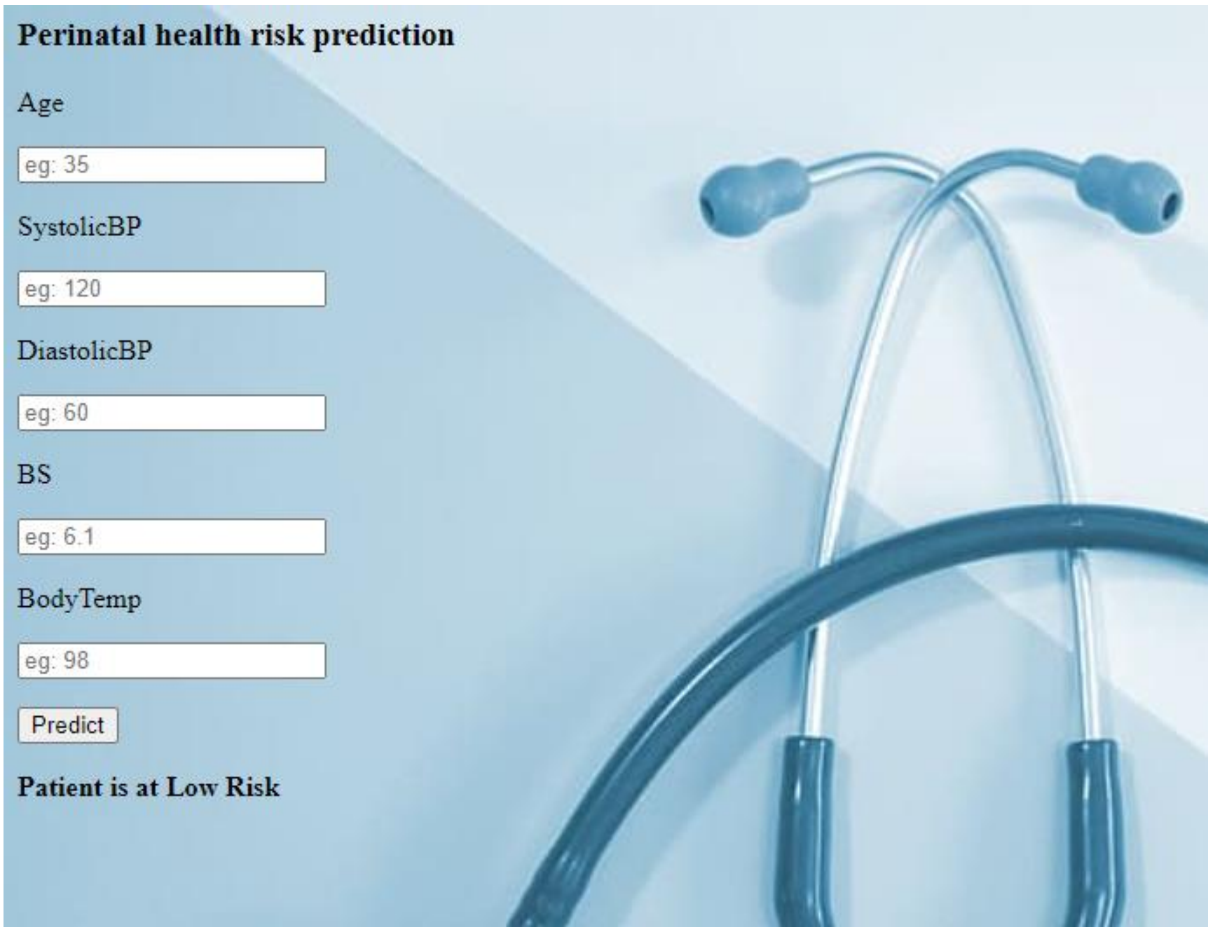
```python
if __name__=='__main__':
    app.run(debug=True)
```

Activity 3: Run the application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type "python app.py" command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top right corner, enter the inputs, click on the submit button, and see the result/prediction on the web.



```
Anaconda Prompt (Anaconda) - python  app.py

(base) C:\Users\SHARAN>f:

(base) F:\>cd Spider\Maternal_Health_Risk2

(base) F:\Spider\Maternal_Health_Risk2>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 114-099-229
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```



Perinatal health risk prediction

Age

eg: 35

SystolicBP

eg: 120

DiastolicBP

eg: 60

BS

eg: 6.1

BodyTemp

eg: 98

Predict

Patient is at Low Risk