LNMIIT
The LNM Institute of
Information Technology

Introduction to Data Science

Maternal Health Risk Data Analysis And Classification

Aarsh Mehtani
21UCS001
Abhay Gupta
21UCS002
Kamal Manchanda
21DCS004
Anurag Garg
19UCS173

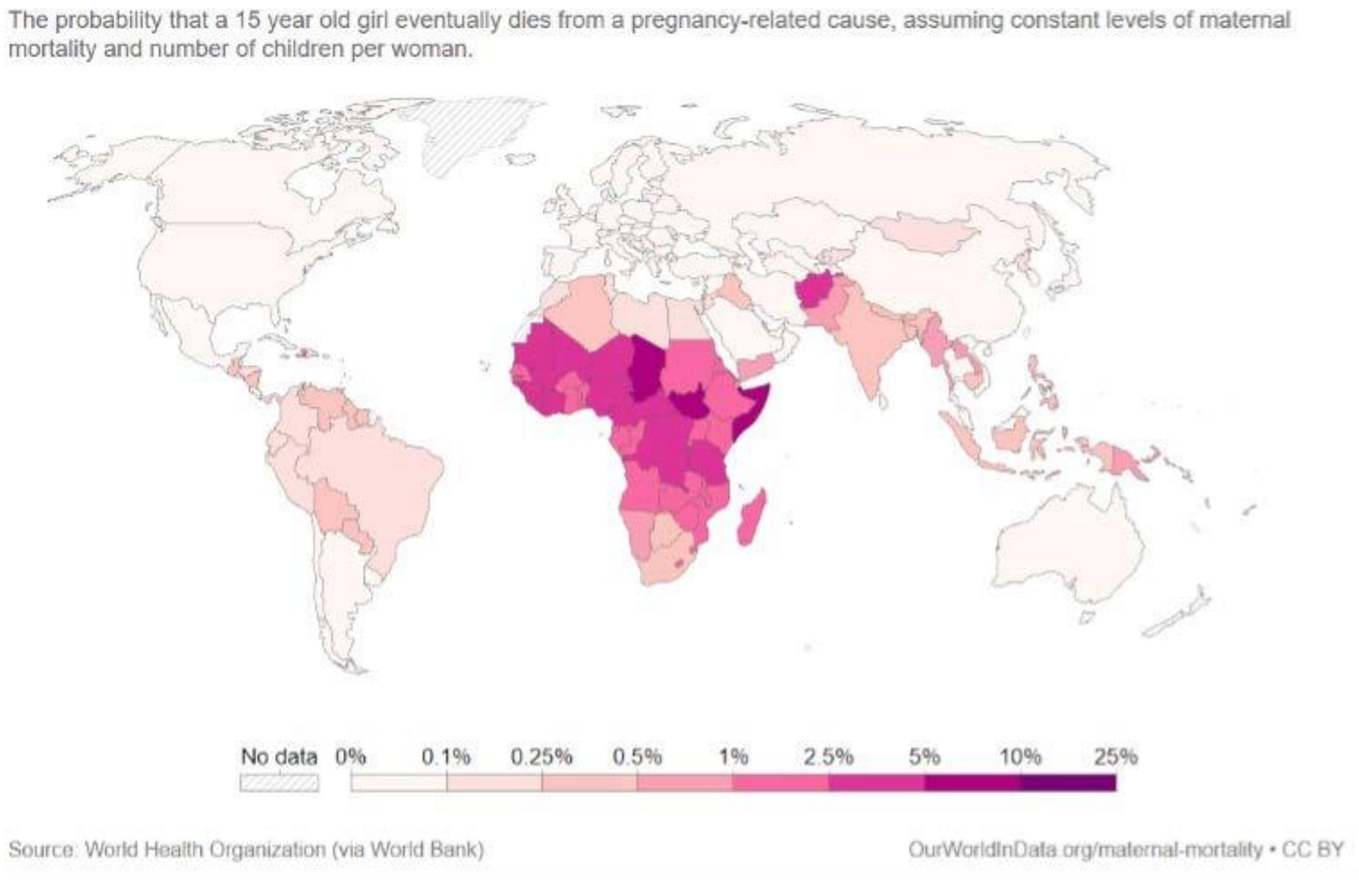
LNMIIT
The LNM Institute of
Information Technology

Contents

1. OBJECTIVE

Maternal mortality is unacceptably high. About 370 000 women died during and following pregnancy and childbirth in 2020. Most of these deaths (94%) occurred in low-resource settings, and most could have been prevented.

- Every day in 2020, approximately 910 women died from preventable causes related to pregnancy and childbirth.
- 94% of all maternal deaths occur in low and lower middle-income countries.
- Young adolescents (ages 10-14) face a higher risk of complications and death because of pregnancy than other women.
- Skilled care before, during and after childbirth can save the lives of women and new-borns.


The probability that a 15 year old girl eventually dies from a pregnancy-related cause, assuming constant levels of maternal mortality and number of children per woman.
No data
0%
0.1%
0.25%
0.5%
1%
2.5%
5%
10%
25%
Source: World Health Organization (via World Bank)
OurWorldInData.org/maternal-mortality • CC BY

Women die because of complications during and following pregnancy and childbirth. Most of these complications develop during pregnancy and most are preventable or treatable. Other complications

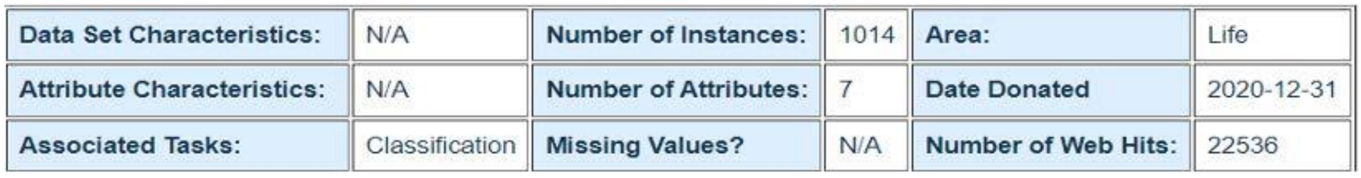
may exist before pregnancy but are worsened during pregnancy, especially if not managed as part of the woman’s care.
In our project, we use the Maternal Health Risk data set to understand which attributes that generally, lead to this Risk and diagnose which attribute impact more on it based on the training and testing set to save their lives by with proper care and detecting the Risks involved in early stages.

2. Introduction to Data Set

Abstract: Data has been collected from different hospitals, community clinics, maternal health cares from the rural areas through the IoT based risk monitoring system.
Data Set Information: Age, Systolic Blood Pressure as SystolicBP, Diastolic BP as DiastolicBP, Blood Sugar as BS, Body Temperature as BodyTemp, HeartRate and RiskLevel. All these are the responsible and significant risk factors for maternal mortality, that is one of the main concerns of SDG of UN.
Attribute Information:

- ❖ Age: Any ages in years when a women during pregnant.
- ❖ SystolicBP: Upper value of Blood Pressure in mmHg, another significant attribute during pregnancy.
- ❖ DiastolicBP: Lower value of Blood Pressure in mmHg, another significant attribute during pregnancy.
- ❖ BS: Blood glucose levels is in terms of a molar concentration, mmol/L.
- ❖ HeartRate: A normal resting heart rate in beats per minute.

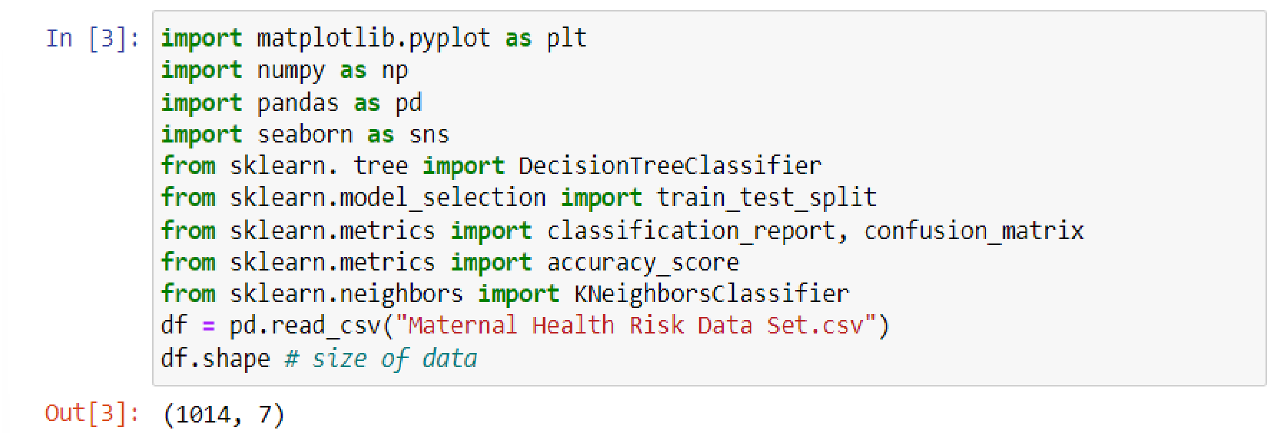
Risk Level: Predicted Risk Intensity Level during pregnancy considering the previous attribute.



Data Set Characteristics:	N/A	**Number of Instances:**	1014	**Area:**	Life
Attribute Characteristics:	N/A	**Number of Attributes:**	7	**Date Donated**	2020-12-31
Associated Tasks:	Classification	**Missing Values?**	N/A	**Number of Web Hits:**	22536

3. Importing Libraries
   ➢ Libraries for graphs and plots: matplotlib.pyplot, numpy, pandas. ➢ Libraries for ML classification: sklearn.

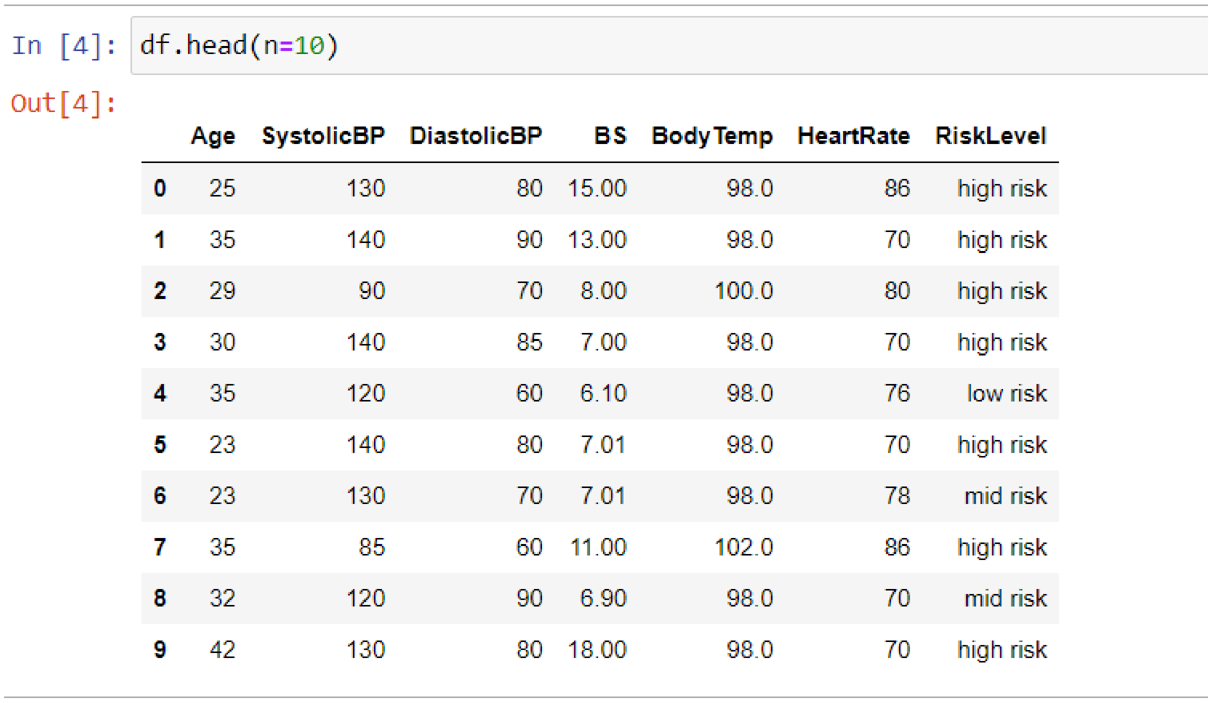
Data set contains 1014 rows and 7 columns:



```
In [3]: import matplotlib.pyplot as plt
        import numpy as np
        import pandas as pd
        import seaborn as sns
        from sklearn. tree import DecisionTreeClassifier
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import classification_report, confusion_matrix
        from sklearn.metrics import accuracy_score
        from sklearn.neighbors import KNeighborsClassifier
        df = pd.read_csv("Maternal Health Risk Data Set.csv")
        df.shape # size of data

Out[3]: (1014, 7)
```

4. **Data Overview:**

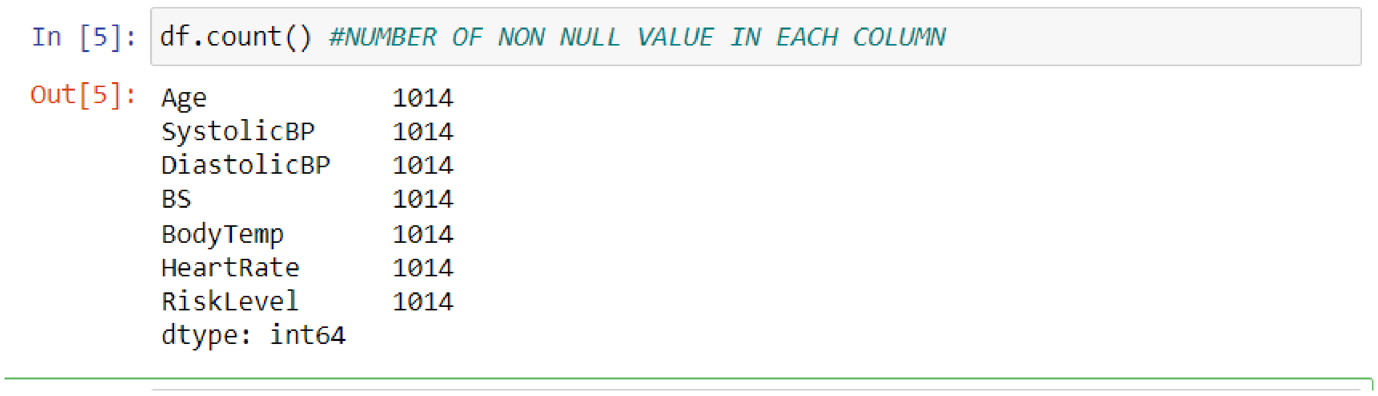
The first 10 rows of data set:



```
In [4]: df.head(n=10)

Out[4]:
```

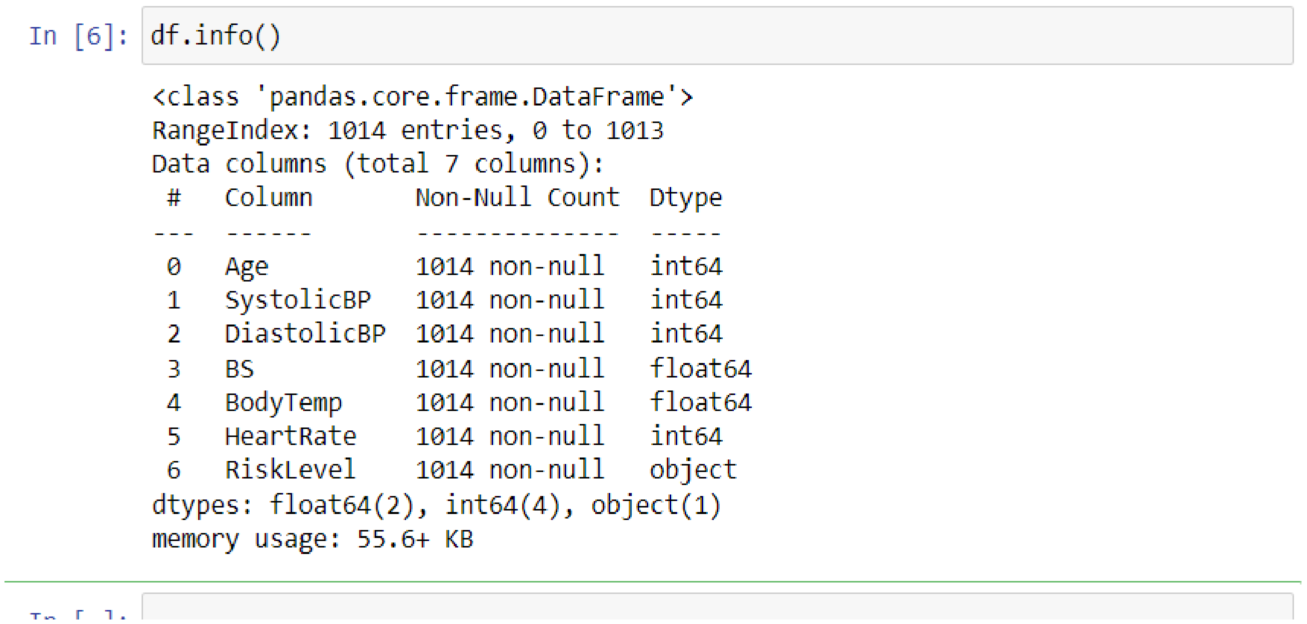
	Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate	RiskLevel
0	25	130	80	15.00	98.0	86	high risk
1	35	140	90	13.00	98.0	70	high risk
2	29	90	70	8.00	100.0	80	high risk
3	30	140	85	7.00	98.0	70	high risk
4	35	120	60	6.10	98.0	76	low risk
5	23	140	80	7.01	98.0	70	high risk
6	23	130	70	7.01	98.0	78	mid risk
7	35	85	60	11.00	102.0	86	high risk
8	32	120	90	6.90	98.0	70	mid risk
9	42	130	80	18.00	98.0	70	high risk

Below code display number of unique or non null values in each column:



```
In [5]: df.count() #NUMBER OF NON NULL VALUE IN EACH COLUMN

Out[5]: Age            1014
        SystolicBP     1014
        DiastolicBP    1014
        BS             1014
        BodyTemp       1014
        HeartRate      1014
        RiskLevel      1014
        dtype: int64
```

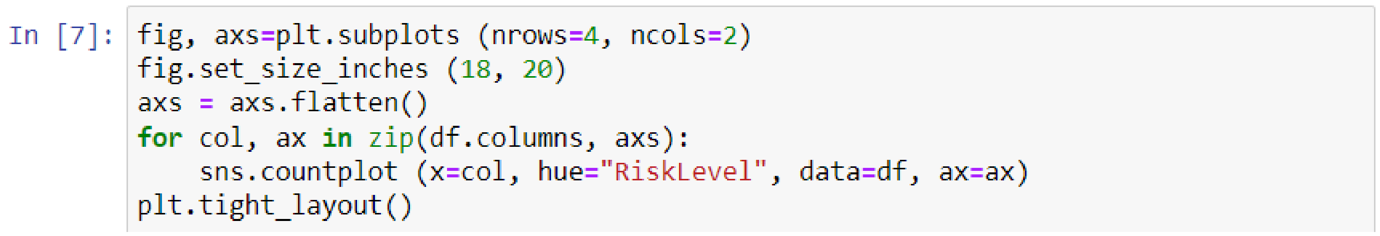
The info() method prints information about the DataFrame.
The information contains the number of columns, column labels, column data types, memory usage, range index, and the number of cells in each column (non-null values).



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

<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

In [ ]:
```

5. Data Visualization:

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

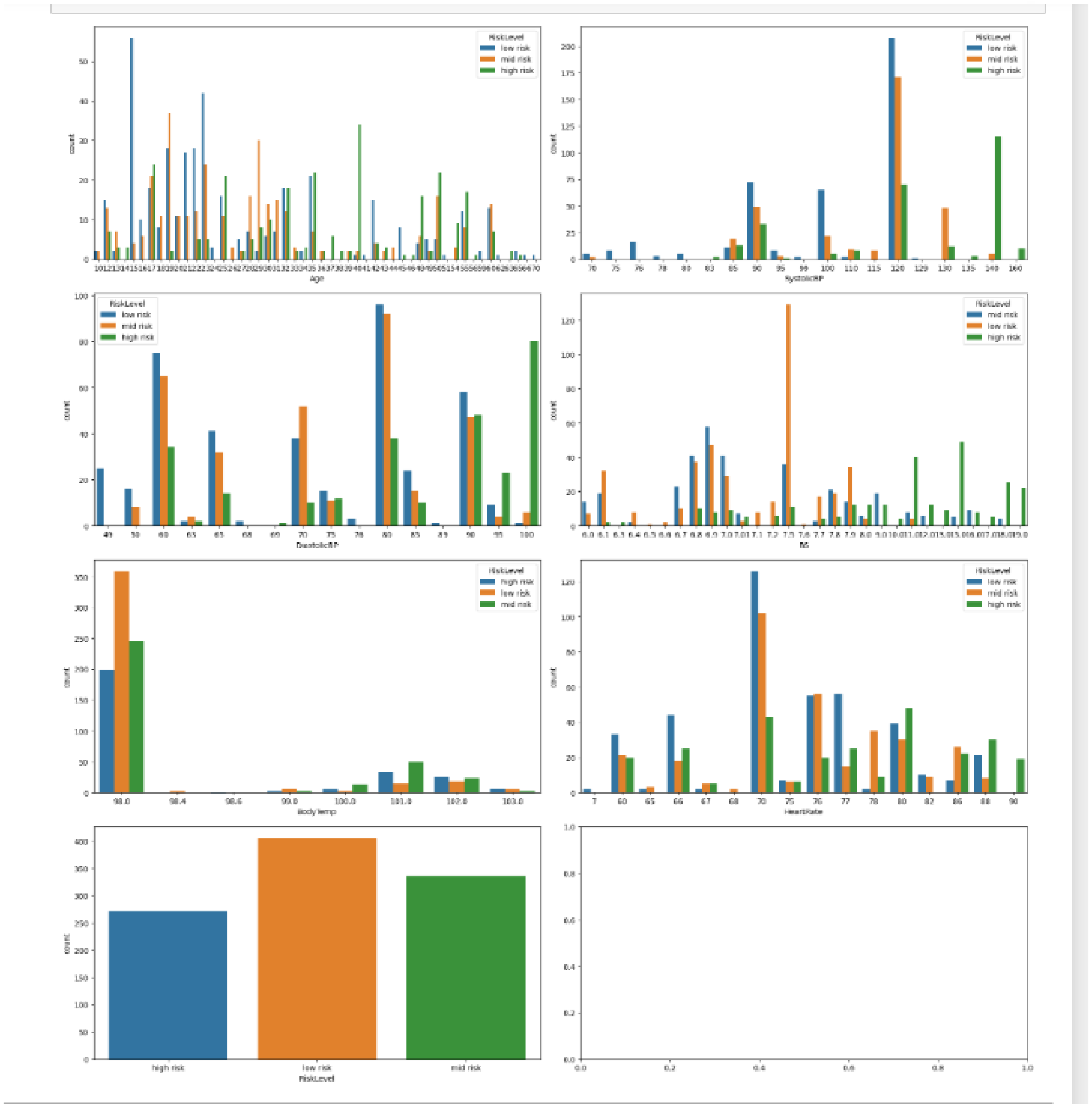


```
In [7]: fig, axs=plt.subplots (nrows=4, ncols=2)
        fig.set_size_inches (18, 20)
        axs = axs.flatten()
        for col, ax in zip(df.columns, axs):
            sns.countplot (x=col, hue="RiskLevel", data=df, ax=ax)
        plt.tight_layout()
```

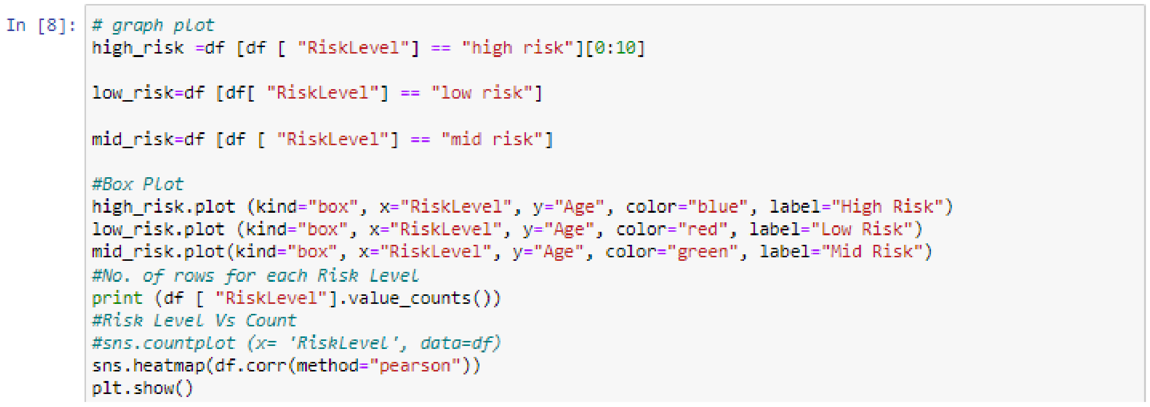
Description:

1. In first graph by using bar plot to visualize the risk level for certain ages.
2. In second graph by using bar plot to visualize the risk level for certain SystolicBP levels.
3. In third graph by using bar plot to visualize the risk level for certain DiastolicBP levels.
4. In fourth graph by using bar plot to visualize the risk level for certain BS values.

5. In fifth graph by using bar plot to visualize the risk level for certain body temperature.
6. In sixth graph by using bar plot to visualize the risk level for certain heart rate.
7. In seventh graph by using bar plot to visualize count of each risk level.

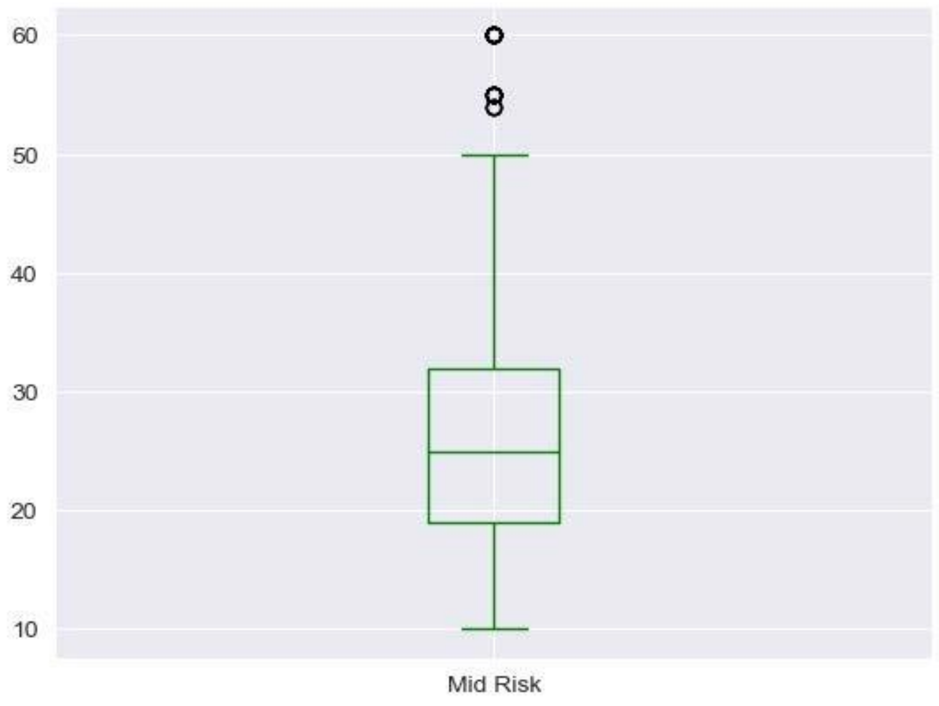

RiskLevel
low risk
mid risk
high risk
count
Age
SystolicBP
DiastolicBP
BS
BodyTemp
HeartRate
RiskLevel
high risk
low risk
mid risk

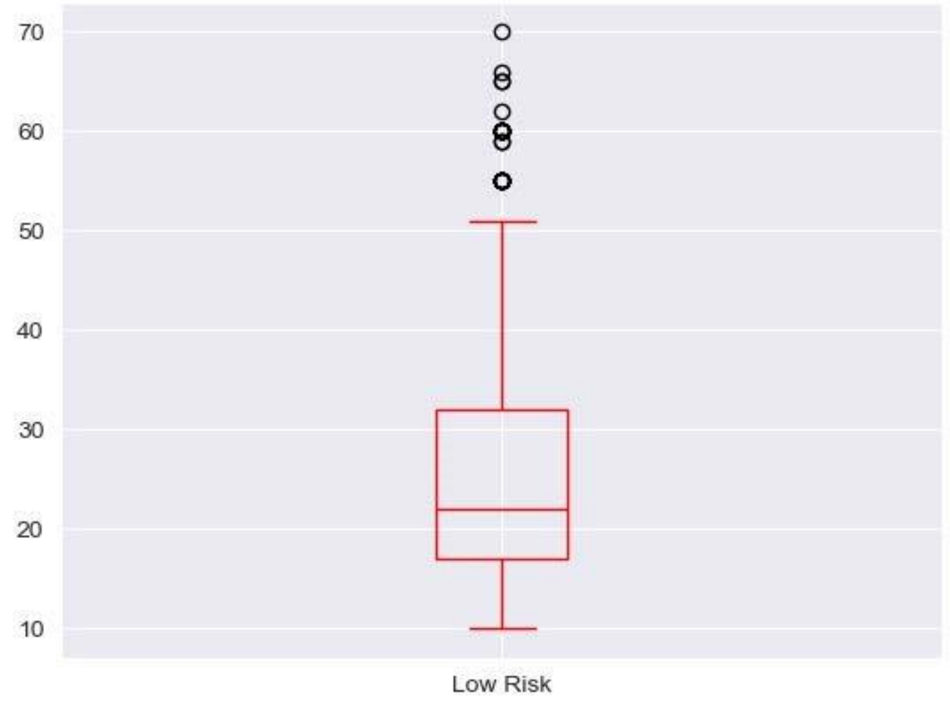
6. Data Analysis:

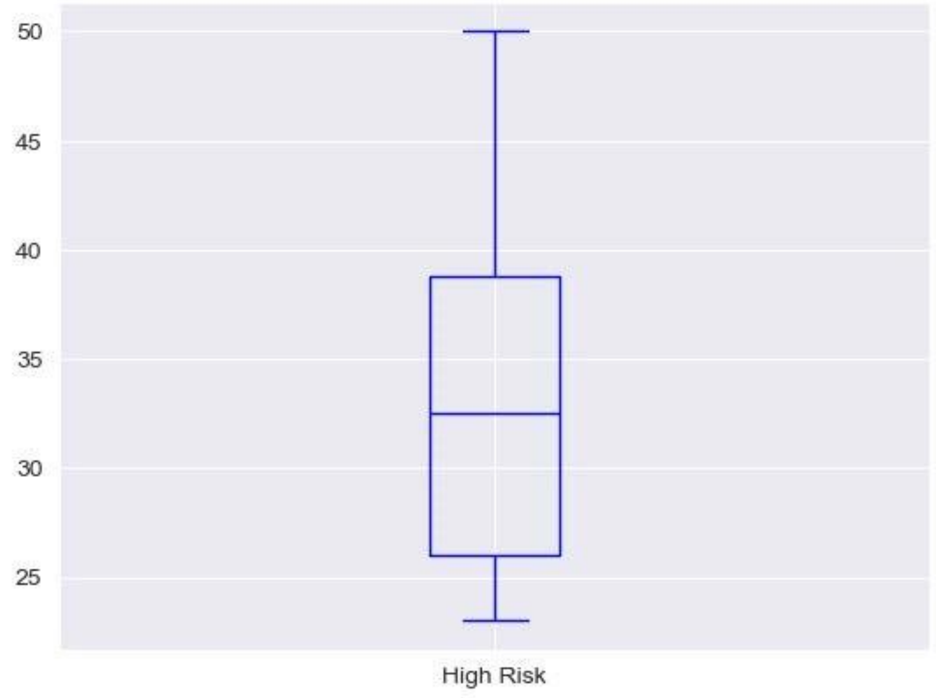


```
In [8]: # graph plot
        high_risk =df [df [ "RiskLevel"] == "high risk"][0:10]

        low_risk=df [df[ "RiskLevel"] == "low risk"]

        mid_risk=df [df [ "RiskLevel"] == "mid risk"]

        #Box Plot
        high_risk.plot (kind="box", x="RiskLevel", y="Age", color="blue", label="High Risk")
        low_risk.plot (kind="box", x="RiskLevel", y="Age", color="red", label="Low Risk")
        mid_risk.plot(kind="box", x="RiskLevel", y="Age", color="green", label="Mid Risk")
        #No. of rows for each Risk Level
        print (df [ "RiskLevel"].value_counts())
        #Risk Level Vs Count
        #sns.countplot (x= 'RiskLevel', data=df)
        sns.heatmap(df.corr(method="pearson"))
        plt.show()
```

6.1 Box Plot (Risk Level Vs Age) :





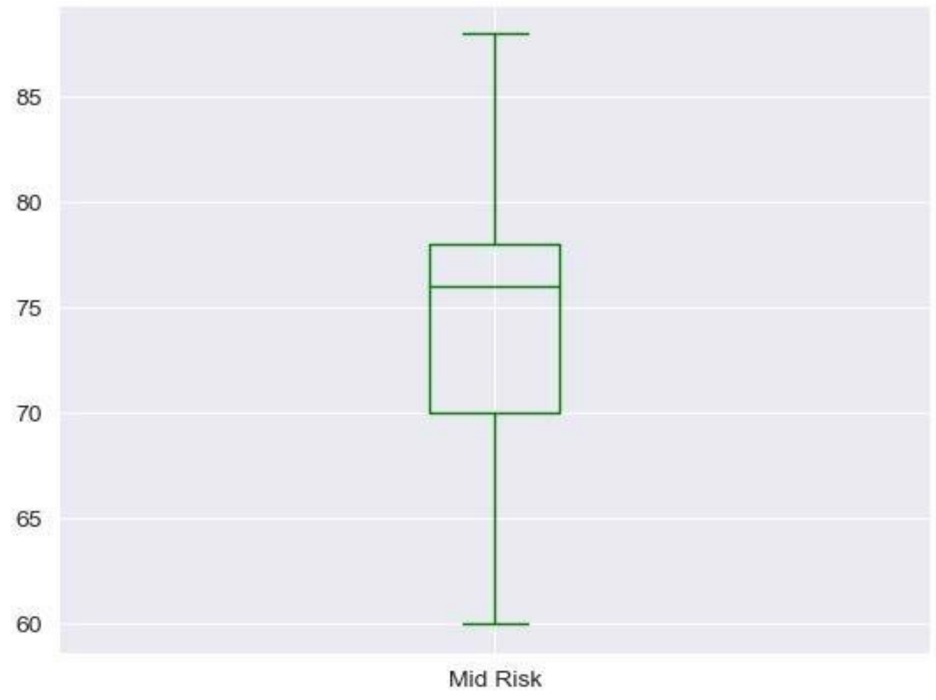


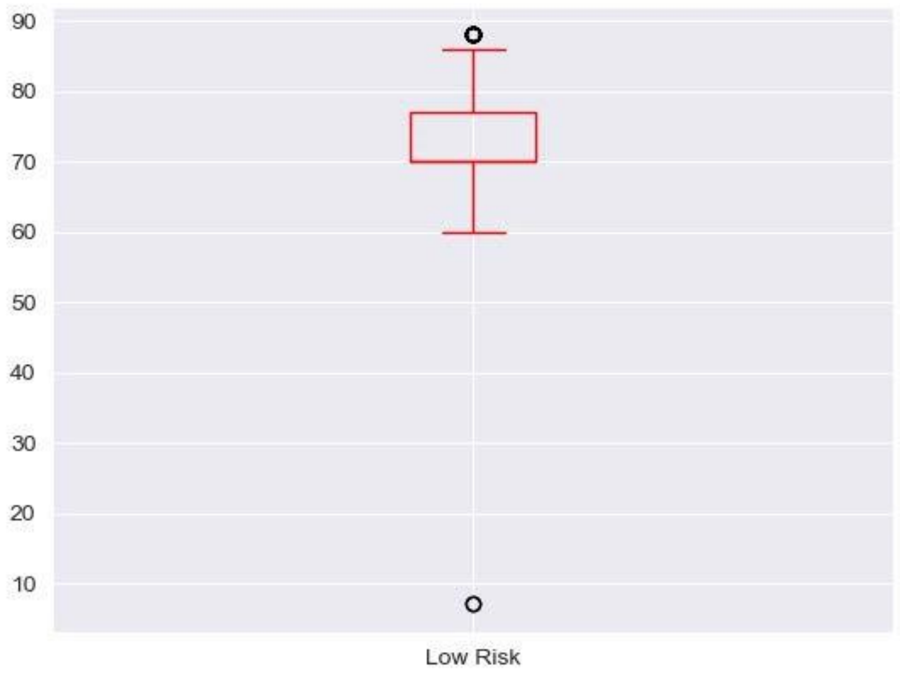
Inference:

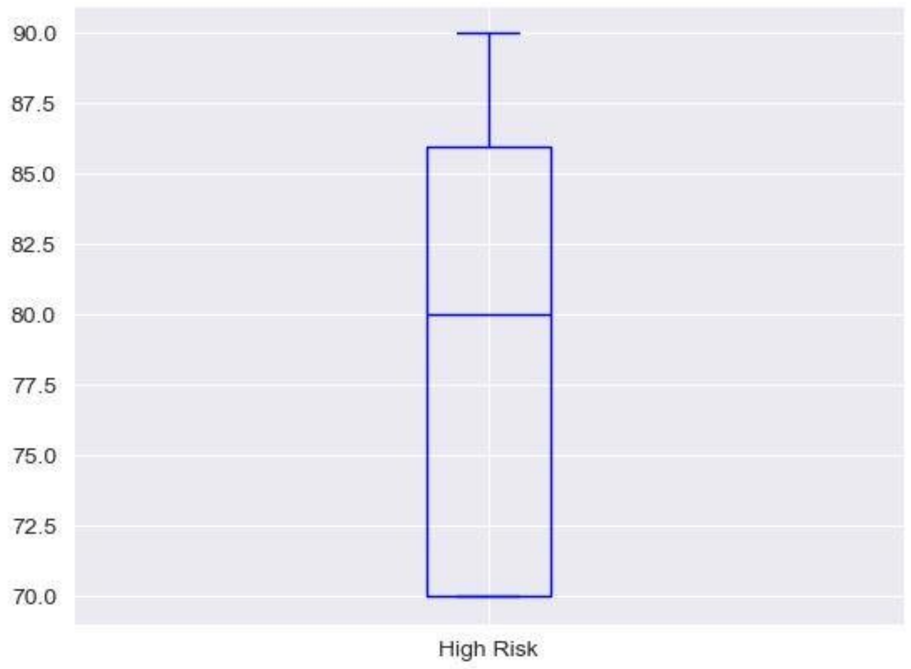
1. A box plot visualization allows you to examine the distribution of data. One box plot appears for each attribute element.

2. Each box plot displays the minimum, first quartile, median, third quartile, and maximum values.
3. From above box plot we conclude that in young age people (20-25 years) the maternal risk is low, in middle age people (25-30 years) the maternal risk is medium, in aged people (> 30 years) risk is high.

6.2 Box Plot (Risk Level Vs Heart rate) :




90
80
70
60
50
40
30
20
10
Low Risk

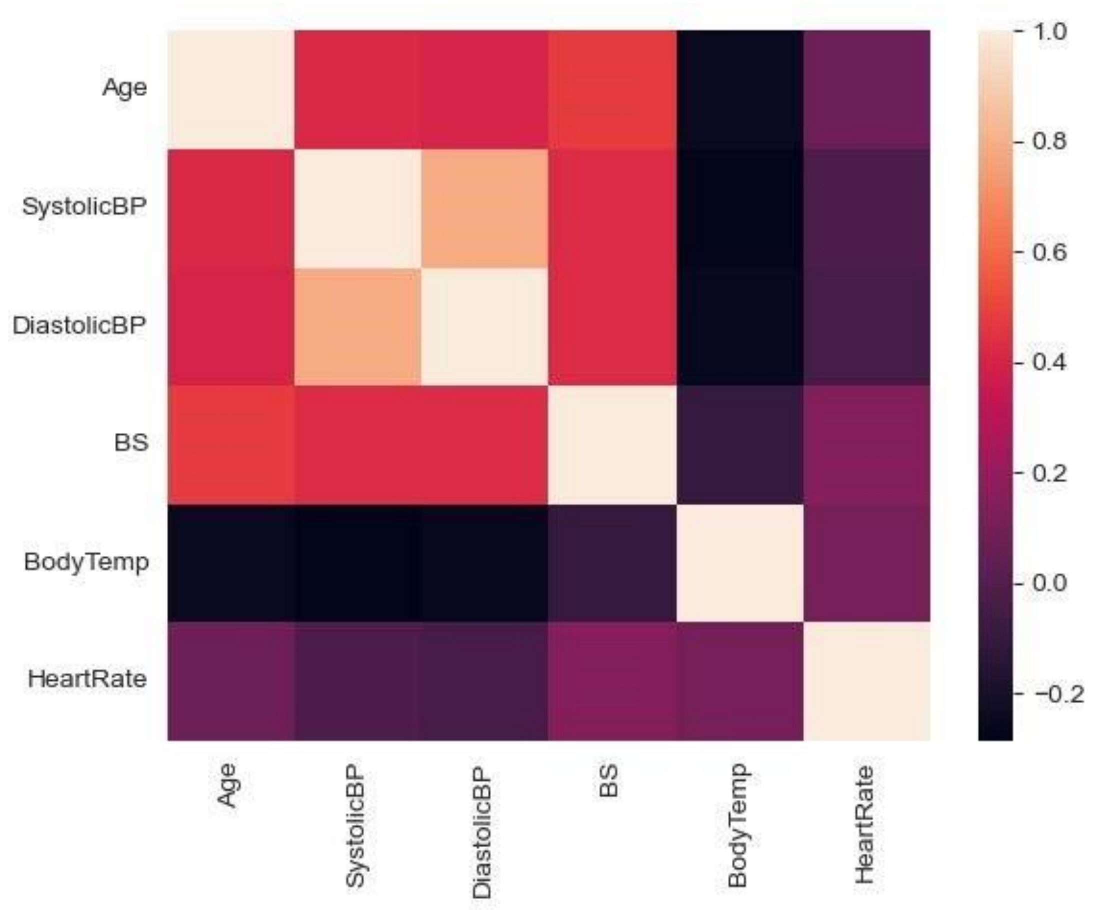

90.0
87.5
85.0
82.5
80.0
77.5
75.0
72.5
70.0
High Risk

Inference:

1. From above box plot we conclude that for low heart rate the risk is medium, at normal heart rate the risk is low and at high heart rate the risk is high.

6.2 Correlation matrix :

- After inferencing the dataset from our graphs, we move towards the mathematical values of correlation between the attributes.
- For that, we plot the Correlation Matrix, using DataFrame.corr() in the seaborn.heatmap() function.



Inference:

1. There is high correlation between:
    a. SystolicBP and age
    b. DiastolicBP and SystolicBP, age
    c. BS and SystolicBP,DiastolicBP, age
    d. Body Temp and BS
    e. HeartRate and Age, BS, BodyTemp

7. Data Preprocessing

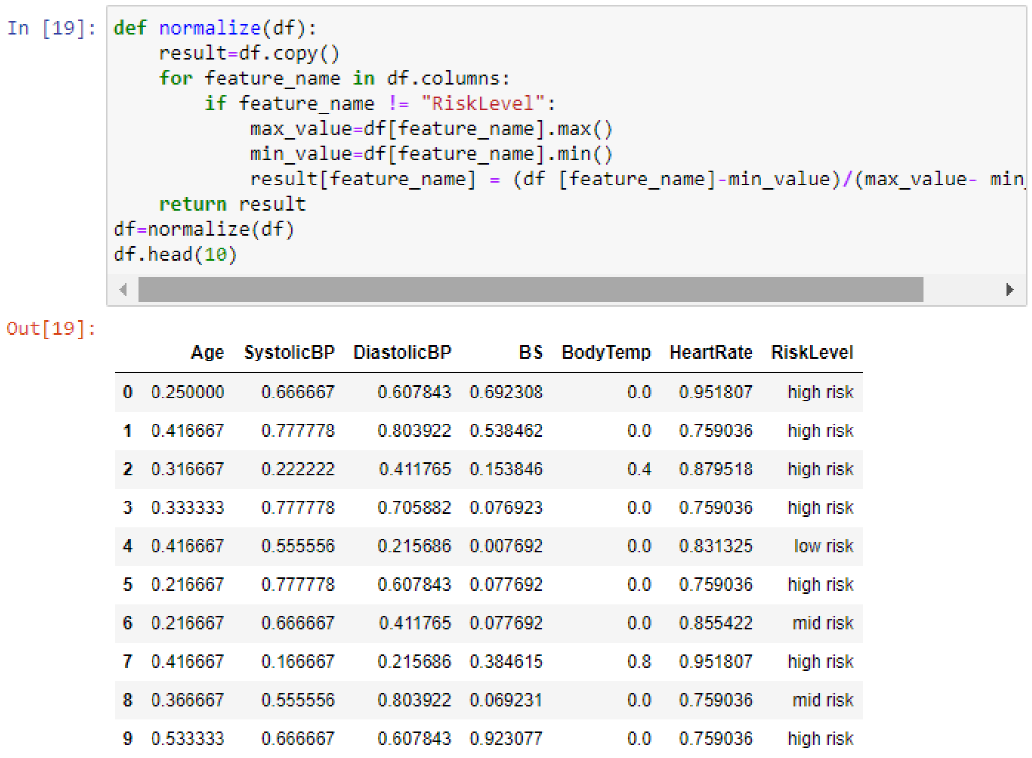
All attributes are non-null in this dataset. Hence, we can proceed with the processing of data.

7.1 Binarization

In our dataset, all attributes are numeric hence no attributes need to binarize. Thus, we can proceed further.

7.2 Normalization using Min-Max scaler function

Since all attributes in our dataset have varying ranges, we need to normalize the at tributes so that each attribute contributes equally in the prediction of class labels.



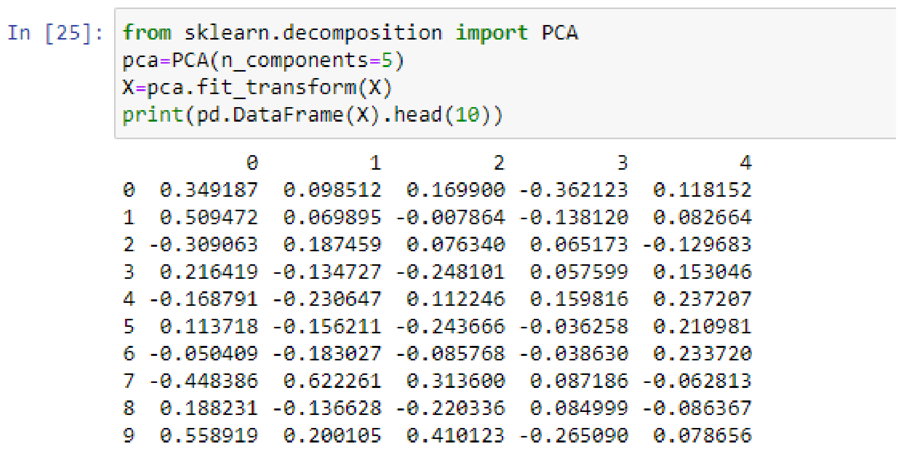
```
In [19]: def normalize(df):
             result=df.copy()
             for feature_name in df.columns:
                 if feature_name != "RiskLevel":
                     max_value=df[feature_name].max()
                     min_value=df[feature_name].min()
                     result[feature_name] = (df [feature_name]-min_value)/(max_value- min
             return result
         df=normalize(df)
         df.head(10)
```

Out[19]:

	Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate	RiskLevel
0	0.250000	0.666667	0.607843	0.692308	0.0	0.951807	high risk
1	0.416667	0.777778	0.803922	0.538462	0.0	0.759036	high risk
2	0.316667	0.222222	0.411765	0.153846	0.4	0.879518	high risk
3	0.333333	0.777778	0.705882	0.076923	0.0	0.759036	high risk
4	0.416667	0.555556	0.215686	0.007692	0.0	0.831325	low risk
5	0.216667	0.777778	0.607843	0.077692	0.0	0.759036	high risk
6	0.216667	0.666667	0.411765	0.077692	0.0	0.855422	mid risk
7	0.416667	0.166667	0.215686	0.384615	0.8	0.951807	high risk
8	0.366667	0.555556	0.803922	0.069231	0.0	0.759036	mid risk
9	0.533333	0.666667	0.607843	0.923077	0.0	0.759036	high risk

7.3 Filtering the Unimportant Variables (Principal Component Analysis)

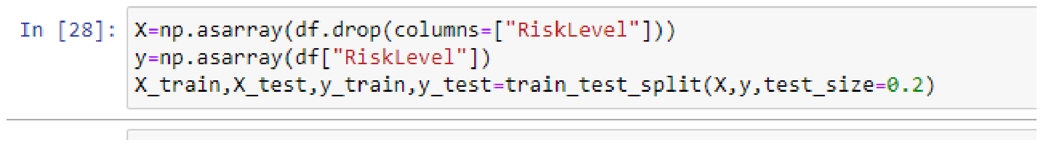
- Principal component analysis (PCA) is a technique for reducing the dimensionality of such datasets, increasing interpretability but at the same time minimizing information loss. It does so by creating new uncorrelated variables that successively maximize variance.
- Since we have 6 dimension in the the data, we use PCA in order to do dimensionality reduction as follow and reduce it to 5 dimensions.



```
In [25]: from sklearn.decomposition import PCA
         pca=PCA(n_components=5)
         X=pca.fit_transform(X)
         print(pd.DataFrame(X).head(10))

          0         1         2         3         4
0  0.349187  0.098512  0.169900 -0.362123  0.118152
1  0.509472  0.069895 -0.007864 -0.138120  0.082664
2 -0.309063  0.187459  0.076340  0.065173 -0.129683
3  0.216419 -0.134727 -0.248101  0.057599  0.153046
4 -0.168791 -0.230647  0.112246  0.159816  0.237207
5  0.113718 -0.156211 -0.243666 -0.036258  0.210981
6 -0.050409 -0.183027 -0.085768 -0.038630  0.233720
7 -0.448386  0.622261  0.313600  0.087186 -0.062813
8  0.188231 -0.136628 -0.220336  0.084999 -0.086367
9  0.558919  0.200105  0.410123 -0.265090  0.078656
```

7.4 Splitting the data into training and test data:

- We need some data to train and test the model
- Sklearn provides the function that splits the data into training and test using some algorithm



```
In [28]: X=np.asarray(df.drop(columns=["RiskLevel"]))
         y=np.asarray(df["RiskLevel"])
         X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
```

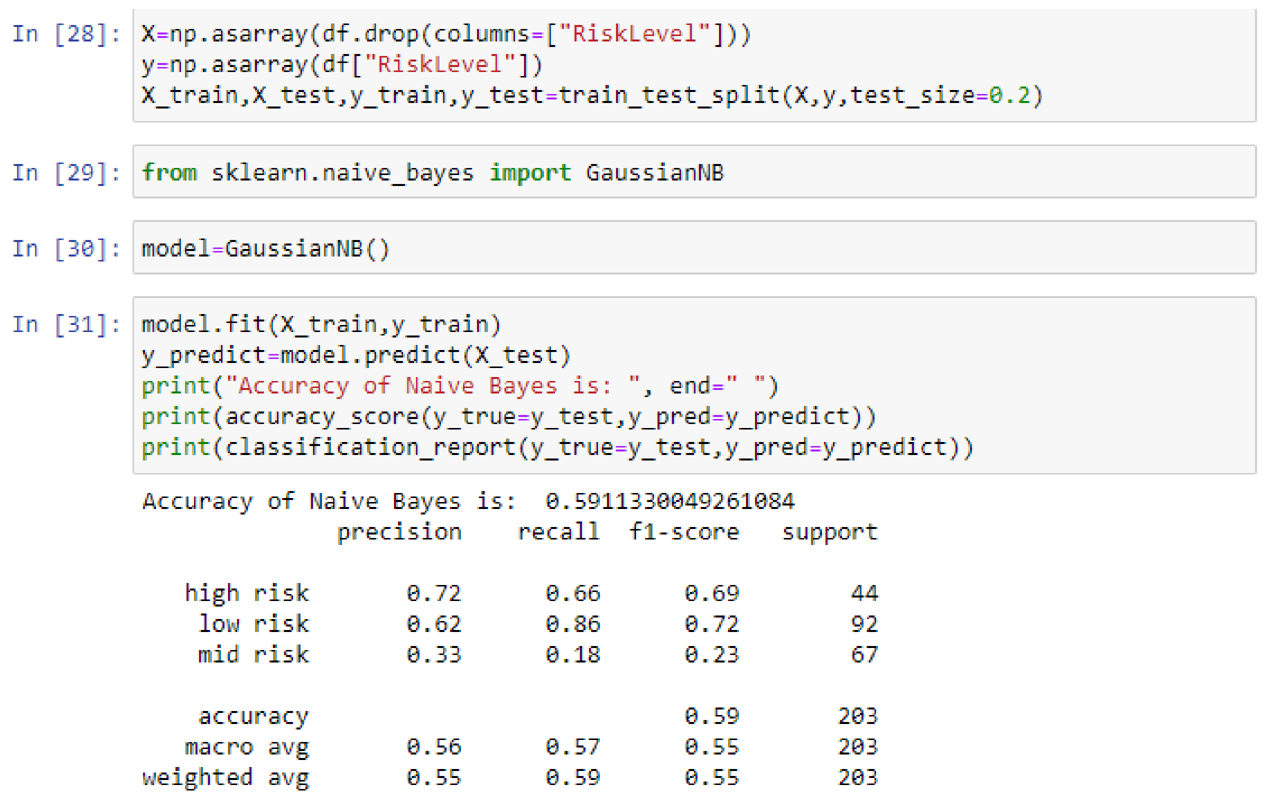
- Now before applying any classification algorithms, we need to do the following tasks.
- Split the dataset into X(input) and Y(output) as 2 separate variables.
- Next, we split our X and Y variables into training data and testing data.
- We use the train_test_split() function of sklearn.model_selection library to split the dataset.

  (80% training and 20% testing)

8. Classification and Prediction:

8.1 Applying Naive Bayes Classifier Algorithm:

- Naive Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- It is mainly used in text classification that includes a high-dimensional training dataset.
- Naive Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

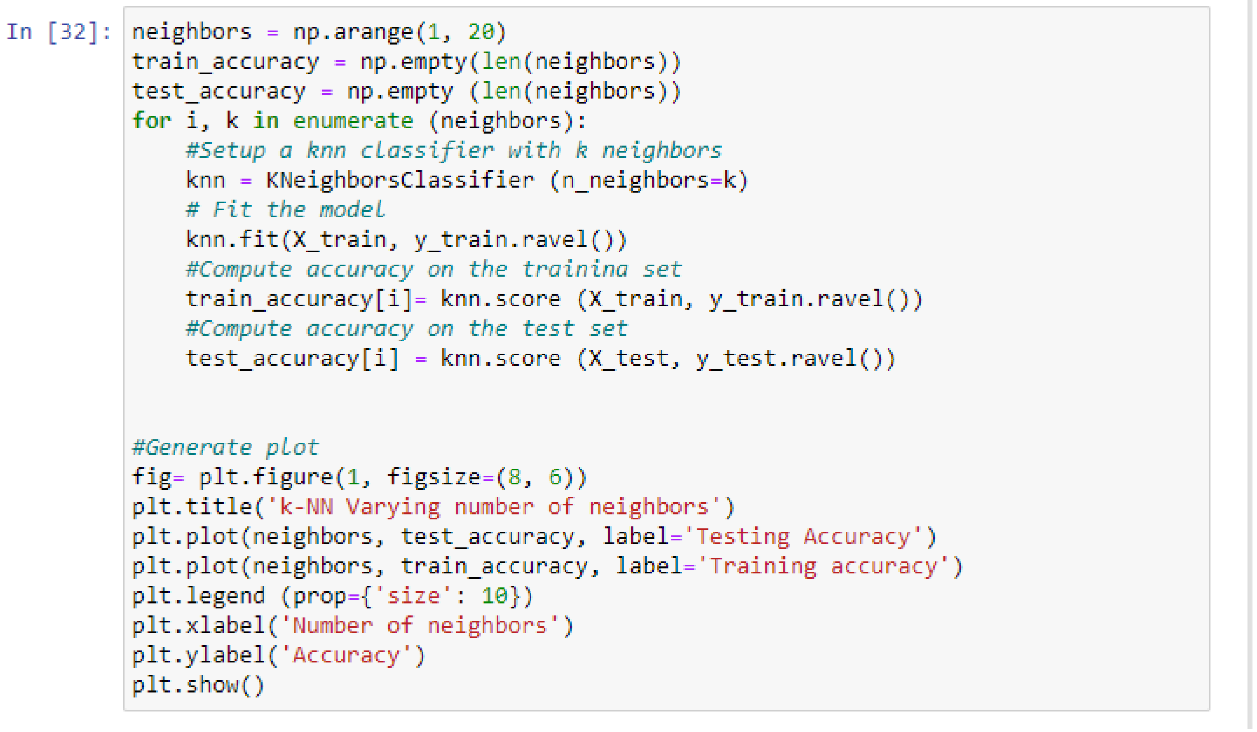


```
In [28]: X=np.asarray(df.drop(columns=["RiskLevel"]))
         y=np.asarray(df["RiskLevel"])
         X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)

In [29]: from sklearn.naive_bayes import GaussianNB

In [30]: model=GaussianNB()

In [31]: model.fit(X_train,y_train)
         y_predict=model.predict(X_test)
         print("Accuracy of Naive Bayes is: ", end=" ")
         print(accuracy_score(y_true=y_test,y_pred=y_predict))
         print(classification_report(y_true=y_test,y_pred=y_predict))
```

```
Accuracy of Naive Bayes is:  0.5911330049261084
              precision    recall  f1-score   support

   high risk       0.72      0.66      0.69        44
    low risk       0.62      0.86      0.72        92
    mid risk       0.33      0.18      0.23        67

    accuracy                           0.59       203
   macro avg       0.56      0.57      0.55       203
weighted avg       0.55      0.59      0.55       203
```

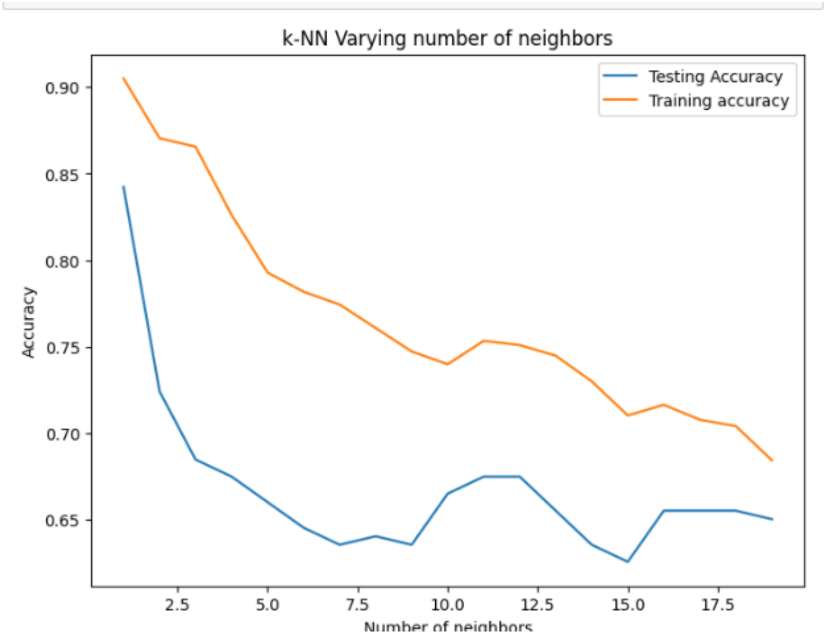
8.2 Applying KNN Algorithm:

- K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.



```
In [32]: neighbors = np.arange(1, 20)
         train_accuracy = np.empty(len(neighbors))
         test_accuracy = np.empty (len(neighbors))
         for i, k in enumerate (neighbors):
             #Setup a knn classifier with k neighbors
             knn = KNeighborsClassifier (n_neighbors=k)
             # Fit the model
             knn.fit(X_train, y_train.ravel())
             #Compute accuracy on the training set
             train_accuracy[i]= knn.score (X_train, y_train.ravel())
             #Compute accuracy on the test set
             test_accuracy[i] = knn.score (X_test, y_test.ravel())


         #Generate plot
         fig= plt.figure(1, figsize=(8, 6))
         plt.title('k-NN Varying number of neighbors')
         plt.plot(neighbors, test_accuracy, label='Testing Accuracy')
         plt.plot(neighbors, train_accuracy, label='Training accuracy')
         plt.legend (prop={'size': 10})
         plt.xlabel('Number of neighbors')
         plt.ylabel('Accuracy')
         plt.show()
```

The below graph gives us brief idea about how accuracy is decrease with increase in number of neighbors in KNN Algorithm.

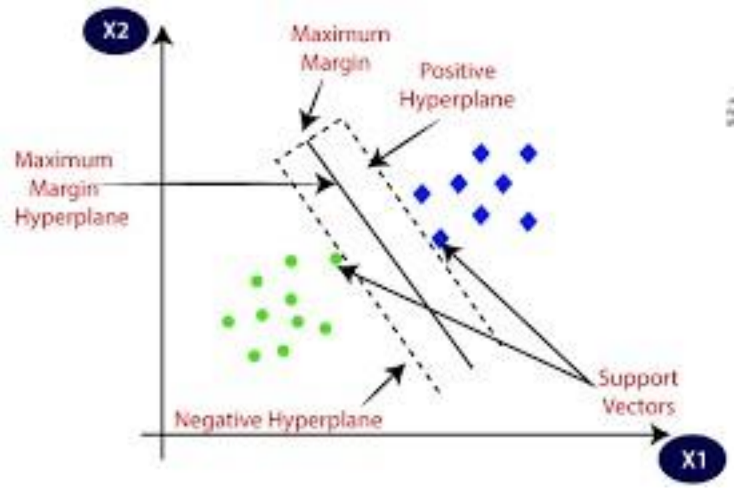


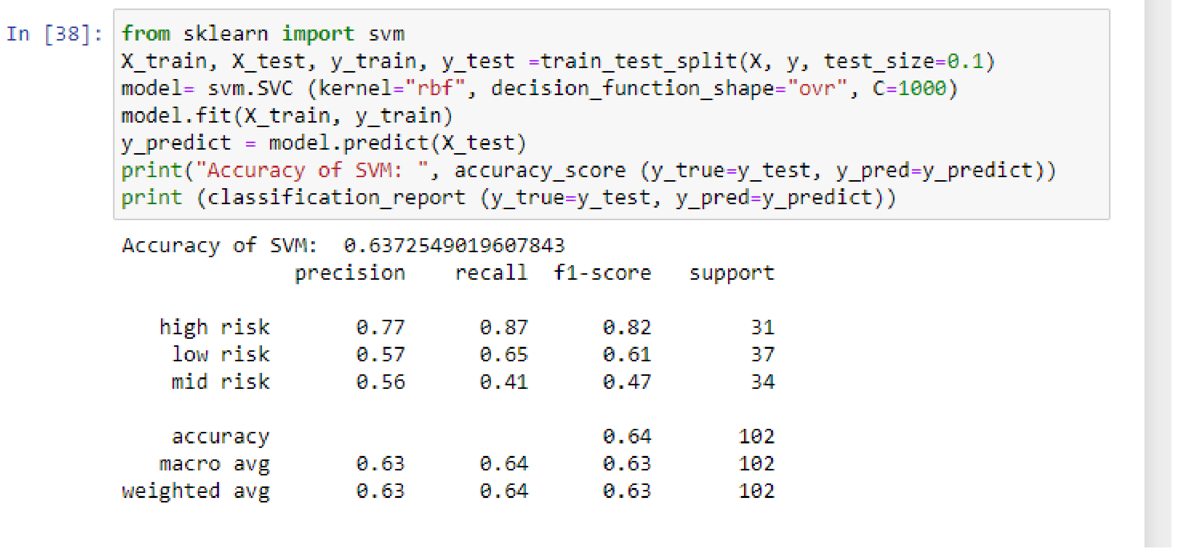
8.3 Applying Support vector machine (SVM) Algorithm:

- Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

- The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

- SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



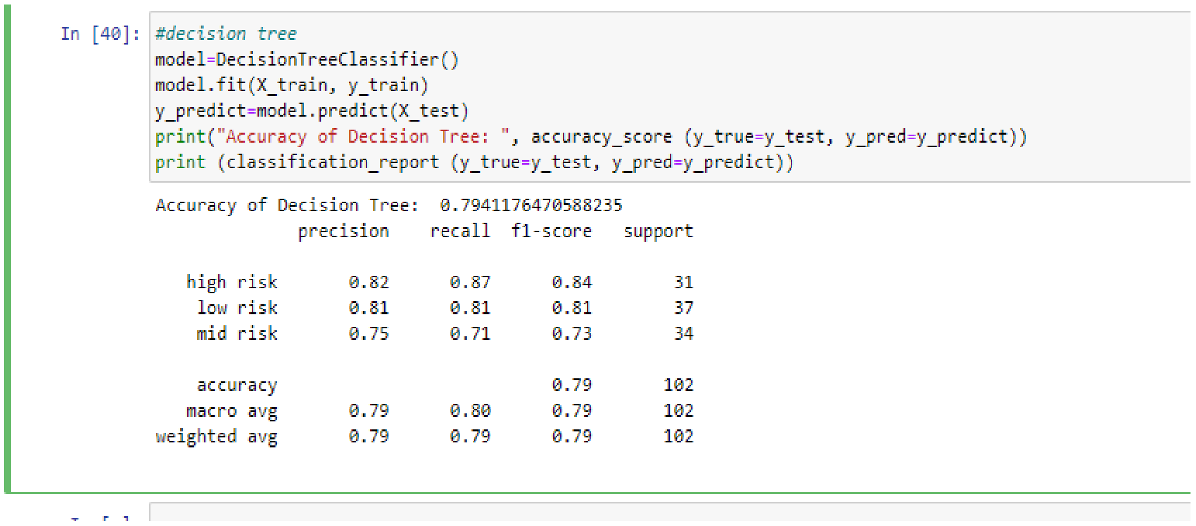


```
In [38]: from sklearn import svm
         X_train, X_test, y_train, y_test =train_test_split(X, y, test_size=0.1)
         model= svm.SVC (kernel="rbf", decision_function_shape="ovr", C=1000)
         model.fit(X_train, y_train)
         y_predict = model.predict(X_test)
         print("Accuracy of SVM: ", accuracy_score (y_true=y_test, y_pred=y_predict))
         print (classification_report (y_true=y_test, y_pred=y_predict))
```

```
Accuracy of SVM:  0.6372549019607843
              precision    recall  f1-score   support

   high risk       0.77      0.87      0.82        31
    low risk       0.57      0.65      0.61        37
    mid risk       0.56      0.41      0.47        34

    accuracy                           0.64       102
   macro avg       0.63      0.64      0.63       102
weighted avg       0.63      0.64      0.63       102
```

8.4 Applying Decision Tree Algorithm:

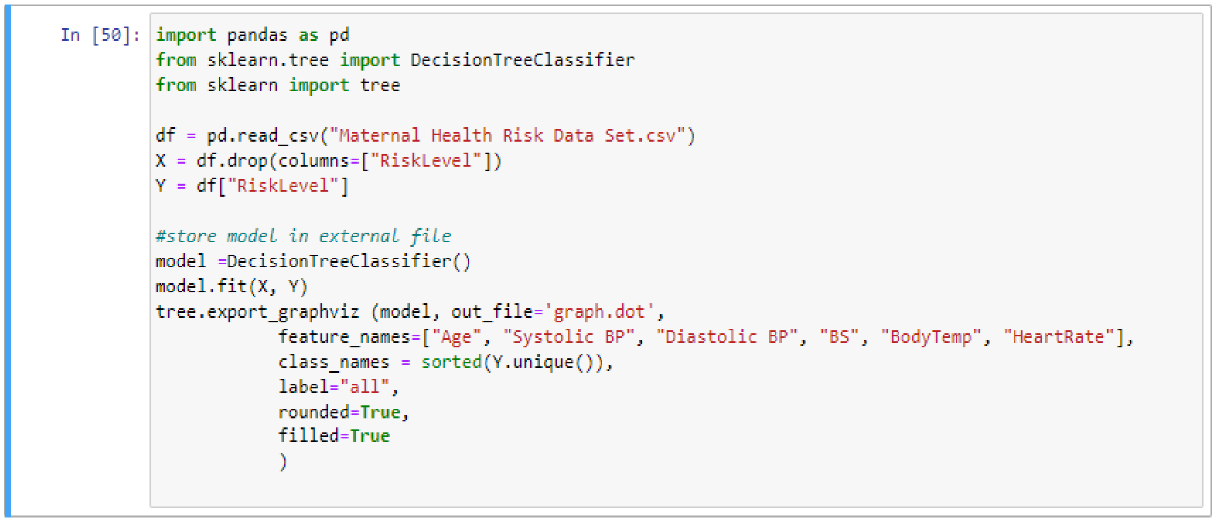
- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.



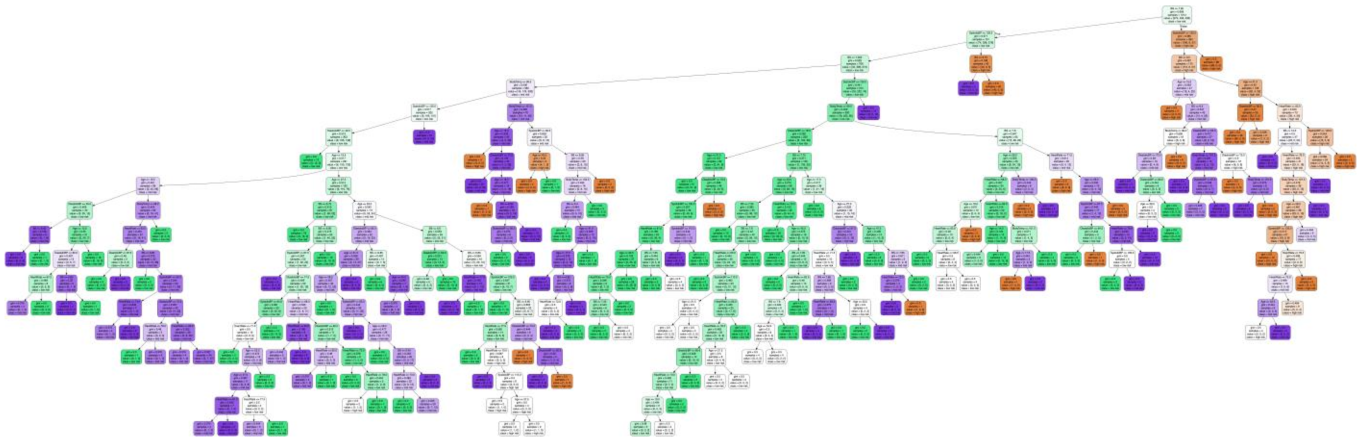
```
In [40]: #decision tree
         model=DecisionTreeClassifier()
         model.fit(X_train, y_train)
         y_predict=model.predict(X_test)
         print("Accuracy of Decision Tree: ", accuracy_score (y_true=y_test, y_pred=y_predict))
         print (classification_report (y_true=y_test, y_pred=y_predict))
```

```
Accuracy of Decision Tree:  0.7941176470588235
              precision    recall  f1-score   support

   high risk       0.82      0.87      0.84        31
    low risk       0.81      0.81      0.81        37
    mid risk       0.75      0.71      0.73        34

    accuracy                           0.79       102
   macro avg       0.79      0.80      0.79       102
weighted avg       0.79      0.79      0.79       102
```

The Decision tree looks like:



```
In [50]: import pandas as pd
         from sklearn.tree import DecisionTreeClassifier
         from sklearn import tree

         df = pd.read_csv("Maternal Health Risk Data Set.csv")
         X = df.drop(columns=["RiskLevel"])
         Y = df["RiskLevel"]

         #store model in external file
         model =DecisionTreeClassifier()
         model.fit(X, Y)
         tree.export_graphviz (model, out_file='graph.dot',
                    feature_names=["Age", "Systolic BP", "Diastolic BP", "BS", "BodyTemp", "HeartRate"],
                    class_names = sorted(Y.unique()),
                    label="all",
                    rounded=True,
                    filled=True
                    )
```



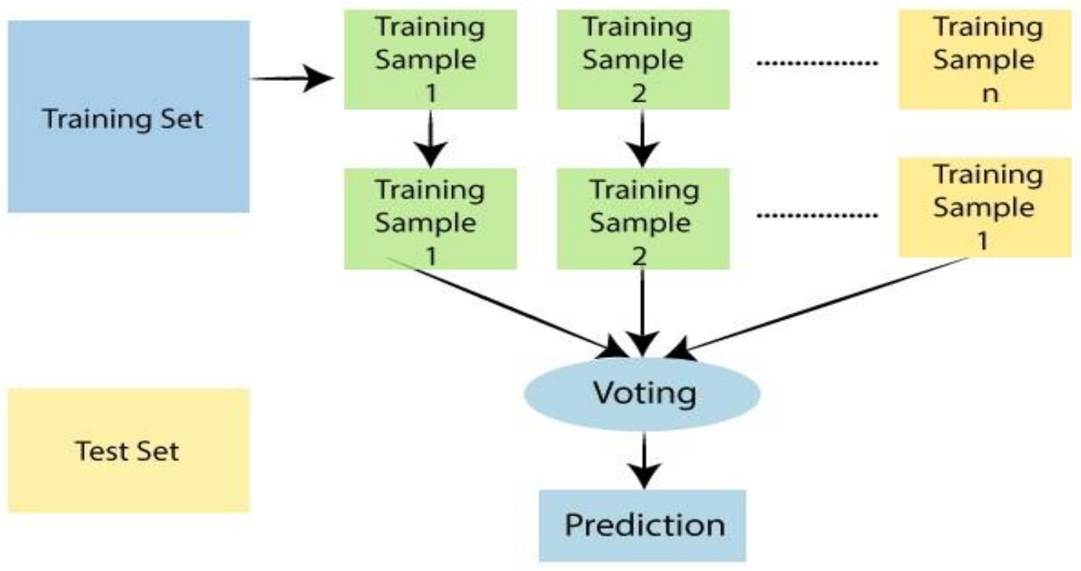
8.5 Applying Random Forest:

- Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

- As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

- The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

- The below diagram explains the working of the Random Forest algorithm:



In [52]:
```
#random forest after PCA
from sklearn.ensemble import RandomForestClassifier
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1)
model = RandomForestClassifier()
model.fit(X_train, y_train)
y_predict= model.predict(X_test)
print (accuracy_score (y_true = y_test, y_pred= y_predict))
print (classification_report (y_true = y_test, y_pred=y_predict))
confusion_matrix (y_true = y_test, y_pred = y_predict)
```

```
0.8333333333333334
              precision    recall  f1-score   support

   high risk       0.83      0.87      0.85        23
    low risk       0.90      0.83      0.86        46
    mid risk       0.75      0.82      0.78        33

    accuracy                           0.83       102
   macro avg       0.83      0.84      0.83       102
weighted avg       0.84      0.83      0.83       102
```

Out[52]:
```
array([[20,  1,  2],
       [ 1, 38,  7],
       [ 3,  3, 27]], dtype=int64)
```

9. Conclusion:

Model	**Accuracy**
Naïve Bayes	0.5911
KNN(k=5)	0.80
SVM	0.64
Decision Tree	0.794
Random Forest	0.83

- By carefully doing data visualization and preprocessing we divided training and testing data set and applying supervised machine learning classification model such as Naive Bayes's, KNN, SVM, Decision Tree and Random Forest.
- We conclude that Random Forest and Decision Tree model give better accuracy than other model to classify the test data.
- So we can use Random forest model to predict the risk level of women's maternal mortality.
- Women with Higher SBP and DBP are generally are at high risk of maternal mortality.
- Younger women with lower SBP, DBP and Blood Sugar (< 7 mmol/L) are at low risk of maternal mortality.

References:

- UCI Machine Learning Repository, "Maternal Health Risk" https://archive.ics.uci.edu/ml/datasets/Maternal+Health+Risk+Data+Set# (accessed Nov 1, 2023).

- Trends in maternal mortality: 2000 to 2017: estimates by WHO, UNICEF, UNFPA, World Bank Group and the United Nations Population Division. Geneva: World Health Organization; 2019.

- WHO: https://www.who.int/news-room/fact-sheets/detail/maternal-mortality