Maternity Health Risk and Mortality Rate Study using Machine Learning Models

Models Used for Training - Support Vector Classification, Decision Tree, Random Forest, GaussianNB

Understanding the dataset ¶

In [ ]:

In [9]:
```
import pandas as pd
import seaborn as sns
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import classification_report

import zipfile
import pandas as pd

zip_file_path = "C:/Users/manoj/Downloads/archive (4).zip"

with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    csv_filename = zip_ref.namelist()[0]
    zip_ref.extract(csv_filename, "./")

df = pd.read_csv(csv_filename)

print(df.head())

df.tail()
```

```
   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
```

Out[9]:

	Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate	RiskLevel
1009	22	120	60	15.0	98.0	80	high risk
1010	55	120	90	18.0	98.0	60	high risk
1011	35	85	60	19.0	98.0	86	high risk
1012	43	120	90	18.0	98.0	70	high risk
1013	32	120	65	6.0	101.0	76	mid risk

In [10]:
```
df.dtypes
```

Out[10]:
```
Age              int64
SystolicBP       int64
DiastolicBP      int64
BS             float64
BodyTemp       float64
HeartRate        int64
RiskLevel       object
dtype: object
```

In [11]:
```
df.info()
df.shape
```

```
<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
```

Out[11]:
```
(1014, 7)
```

In [12]:
```
#Missing Valuse
df.isnull().sum()
```

Out[12]:
```
Age            0
SystolicBP     0
DiastolicBP    0
BS             0
BodyTemp       0
HeartRate      0
RiskLevel      0
dtype: int64
```

In [13]:
```
#Duplicated value
data_dup = df.duplicated().any()
data_dup
```

Out[13]:
```
True
```

In [14]:
```
df = df.drop_duplicates()
data_dup = df.duplicated().any()
data_dup
```

Out[14]:
```
False
```

In [15]:
```
ca_val=[]
co_val=[]

for column in df.columns:
    if df[column].nunique() <=10:
        ca_val.append(column)
    else:
        co_val.append(column)
```

In [16]:
```
#Categorical Data
ca_val
```

Out[16]:
```
['BodyTemp', 'RiskLevel']
```

In [17]:
```
df['BodyTemp'].unique()
```

Out[17]:
```
array([ 98. , 100. , 102. , 101. , 103. ,  98.4,  99. ,  98.6])
```

In [18]:
```
df['RiskLevel'].unique()
```

Out[18]:
```
array(['high risk', 'low risk', 'mid risk'], dtype=object)
```

In [19]:
```
co_val
```

Out[19]:
```
['Age', 'SystolicBP', 'DiastolicBP', 'BS', 'HeartRate']
```

In [20]:
```
df['SystolicBP'].unique()
```

Out[20]:
```
array([130, 140,  90, 120,  85, 110,  70, 100,  75,  95,  76,  80, 115,
       135, 160, 129,  83,  99,  78], dtype=int64)
```

In [21]:
```
df['Age'].unique()
```

Out[21]:
```
array([25, 35, 29, 30, 23, 32, 42, 19, 20, 48, 15, 50, 10, 40, 21, 18, 16,
       22, 49, 28, 12, 60, 55, 45, 31, 17, 26, 54, 44, 33, 13, 34, 38, 39,
       63, 14, 37, 51, 62, 43, 65, 66, 56, 70, 27, 36, 59, 24, 41, 46],
      dtype=int64)
```

In [22]:
```
df['DiastolicBP'].unique()
```

Out[22]:
```
array([ 80,  90,  70,  85,  60,  89,  75, 100,  50,  65,  95,  49,  63,
        69,  76,  68], dtype=int64)
```

In [23]:
```
df['HeartRate'].unique()
```

Out[23]:
```
array([86, 70, 80, 76, 78, 77, 88, 90, 66, 82, 60, 75, 67, 65, 68,  7],
      dtype=int64)
```

In [24]:
```
RiskLevel = {'low risk':1,
        'mid risk':2,
        'high risk':3}

# applying using map
df['RiskLevel'] = df['RiskLevel'].map(RiskLevel).astype(float)
df
```

Out[24]:

	Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate	RiskLevel
0	25	130	80	15.0	98.0	86	3.0
1	35	140	90	13.0	98.0	70	3.0
2	29	90	70	8.0	100.0	80	3.0
3	30	140	85	7.0	98.0	70	3.0
4	35	120	60	6.1	98.0	76	1.0
...	...	...	...	...	...	...	...
673	12	100	50	6.4	98.0	70	2.0
674	15	100	60	6.0	98.0	80	1.0
703	15	100	49	7.6	98.0	77	1.0
704	12	100	50	6.0	98.0	70	2.0
705	21	100	50	6.8	98.0	60	1.0

452 rows × 7 columns

In [25]:
```
df['RiskLevel'].value_counts()
```

Out[25]:
```
1.0    234
3.0    112
2.0    106
Name: RiskLevel, dtype: int64
```

In [26]:
```
df.describe()
```

Out[26]:

	Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate	RiskLevel
count	452.000000	452.000000	452.000000	452.000000	452.000000	452.000000	452.000000
mean	29.194690	110.553097	75.418142	8.346173	98.692478	73.949115	1.730088
std	13.767379	17.872282	13.754578	2.829209	1.410897	8.156973	0.833169
min	10.000000	70.000000	49.000000	6.000000	98.000000	7.000000	1.000000
25%	19.000000	90.000000	65.000000	6.900000	98.000000	70.000000	1.000000
50%	25.000000	120.000000	80.000000	7.500000	98.000000	76.000000	1.000000
75%	35.000000	120.000000	86.000000	7.900000	98.000000	80.000000	2.000000
max	70.000000	160.000000	100.000000	19.000000	103.000000	90.000000	3.000000

Co-Relations

In [27]:
```
df[['RiskLevel','Age']].corr()
```

Out[27]:

	RiskLevel	Age
RiskLevel	1.00000	0.18301
Age	0.18301	1.00000

In [28]:
```
df[['RiskLevel','SystolicBP']].corr()
```

Out[28]:

	RiskLevel	SystolicBP
RiskLevel	1.000000	0.327365
SystolicBP	0.327365	1.000000

In [29]:
```
df[['RiskLevel','DiastolicBP']].corr()
```

Out[29]:

	RiskLevel	DiastolicBP
RiskLevel	1.000000	0.254239
DiastolicBP	0.254239	1.000000

In [30]:
```
df[['RiskLevel','BS']].corr()
```

Out[30]:

	RiskLevel	BS
RiskLevel	1.000000	0.548888
BS	0.548888	1.000000

In [31]:
```
df[['RiskLevel','HeartRate']].corr()
```

Out[31]:

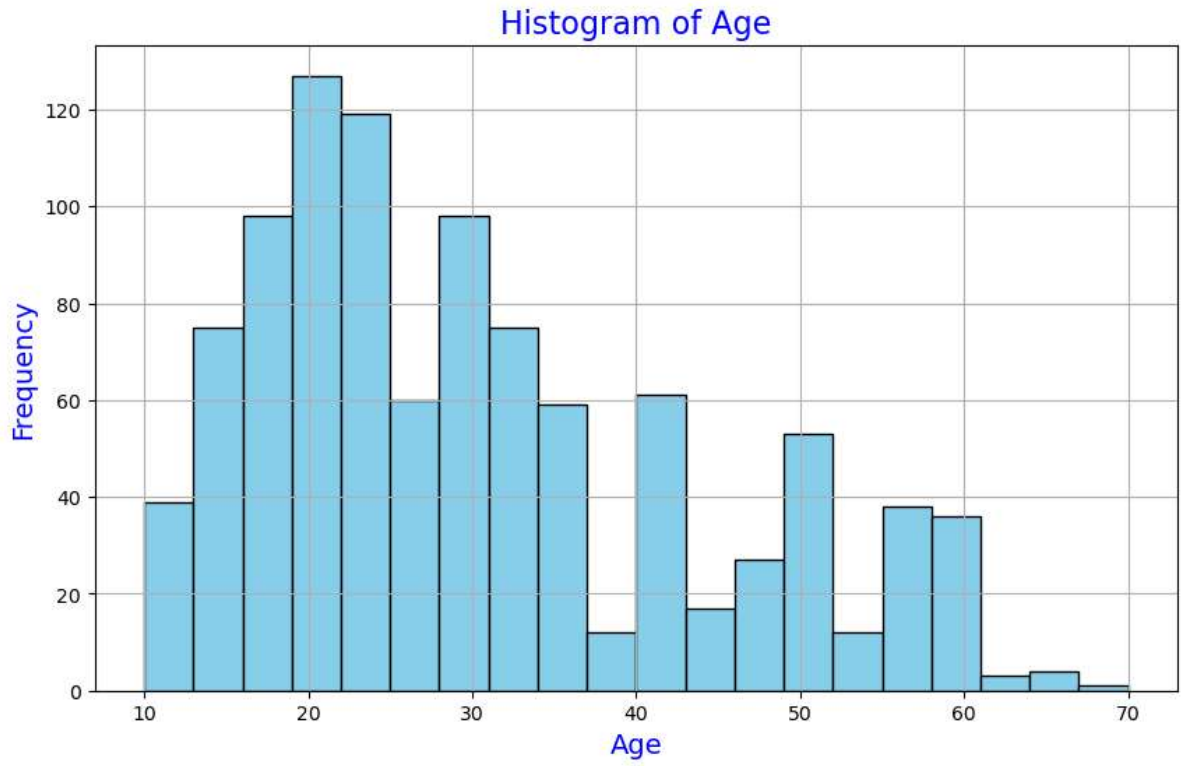
	RiskLevel	HeartRate
RiskLevel	1.000000	0.183289
HeartRate	0.183289	1.000000

In [3]:
```python
import pandas as pd
import zipfile
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

zip_file_path = "C:/Users/manoj/Downloads/archive (4).zip"

with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    csv_filename = zip_ref.namelist()[0]
    zip_ref.extract(csv_filename, "./")

df = pd.read_csv(csv_filename)

# Pairplot
sns.pairplot(df[['Age', 'SystolicBP', 'DiastolicBP', 'BS', 'HeartRate']], pale
plt.show()

# Scatter plot of SystolicBP vs DiastolicBP colored by RiskLevel
fig = px.scatter(df, x='SystolicBP', y='DiastolicBP', hover_data=['Age'], colo
fig.show()
```

```
C:\Users\manoj\anaconda3\Lib\site-packages\seaborn\axisgrid.py:1507: UserW
arning: Ignoring `palette` because no `hue` variable has been assigned.
  func(x=vector, **plot_kwargs)
C:\Users\manoj\anaconda3\Lib\site-packages\seaborn\axisgrid.py:1507: UserW
arning: Ignoring `palette` because no `hue` variable has been assigned.
  func(x=vector, **plot_kwargs)
C:\Users\manoj\anaconda3\Lib\site-packages\seaborn\axisgrid.py:1507: UserW
arning: Ignoring `palette` because no `hue` variable has been assigned.
  func(x=vector, **plot_kwargs)
C:\Users\manoj\anaconda3\Lib\site-packages\seaborn\axisgrid.py:1507: UserW
arning: Ignoring `palette` because no `hue` variable has been assigned.
  func(x=vector, **plot_kwargs)
C:\Users\manoj\anaconda3\Lib\site-packages\seaborn\axisgrid.py:1507: UserW
arning: Ignoring `palette` because no `hue` variable has been assigned.
  func(x=vector, **plot_kwargs)
C:\Users\manoj\anaconda3\Lib\site-packages\seaborn\axisgrid.py:1609: UserW
arning: Ignoring `palette` because no `hue` variable has been assigned.
  func(x=x, y=y, **kwargs)
C:\Users\manoj\anaconda3\Lib\site-packages\seaborn\axisgrid.py:1609: UserW
```

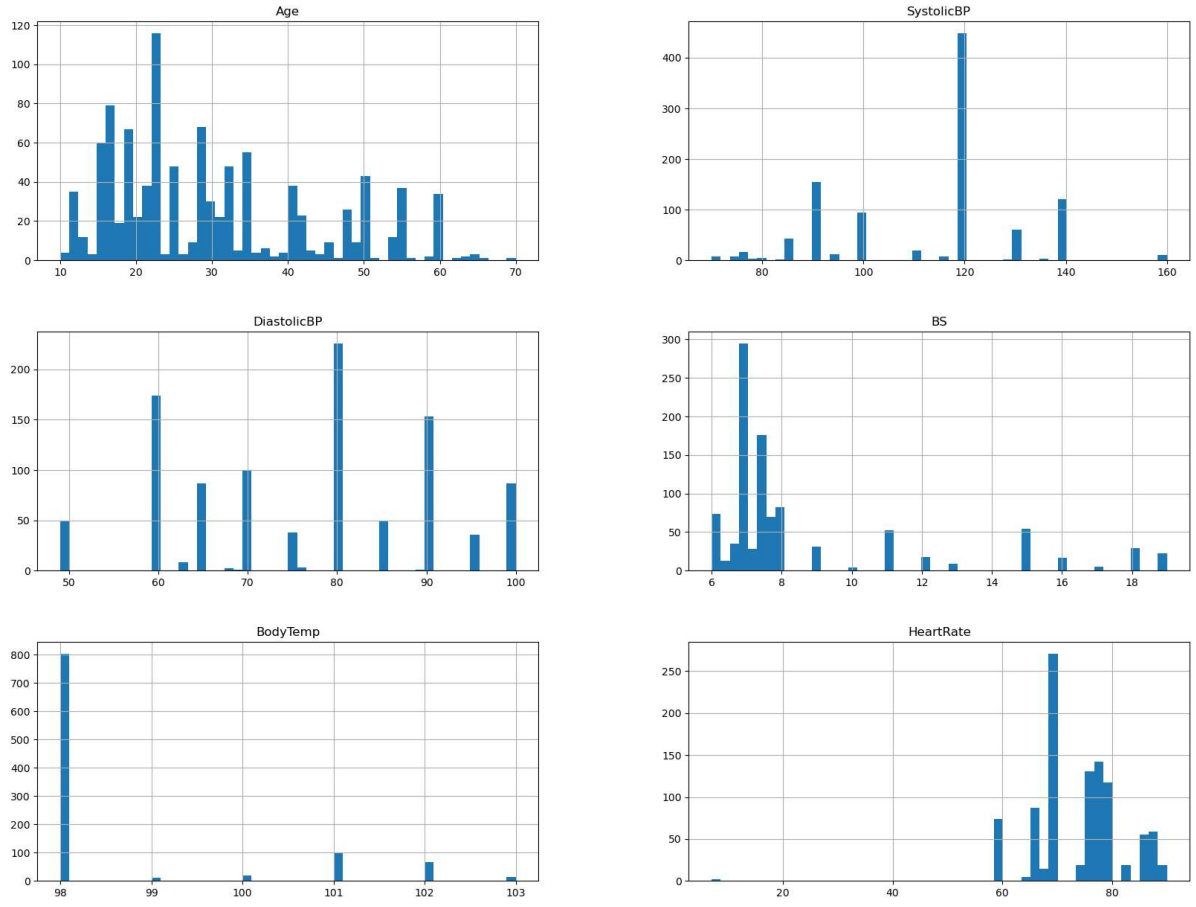
Frequency v/s Age Histogram

In [33]:
```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(10, 6))

ax.hist(df['Age'], bins=20, color='skyblue', edgecolor='black')

ax.set_title('Histogram of Age', color='blue', fontsize=16)
ax.set_xlabel('Age', color='blue', fontsize=14)
ax.set_ylabel('Frequency', color='blue', fontsize=14)

ax.grid(True)

plt.show()
```



Histograms for different factors

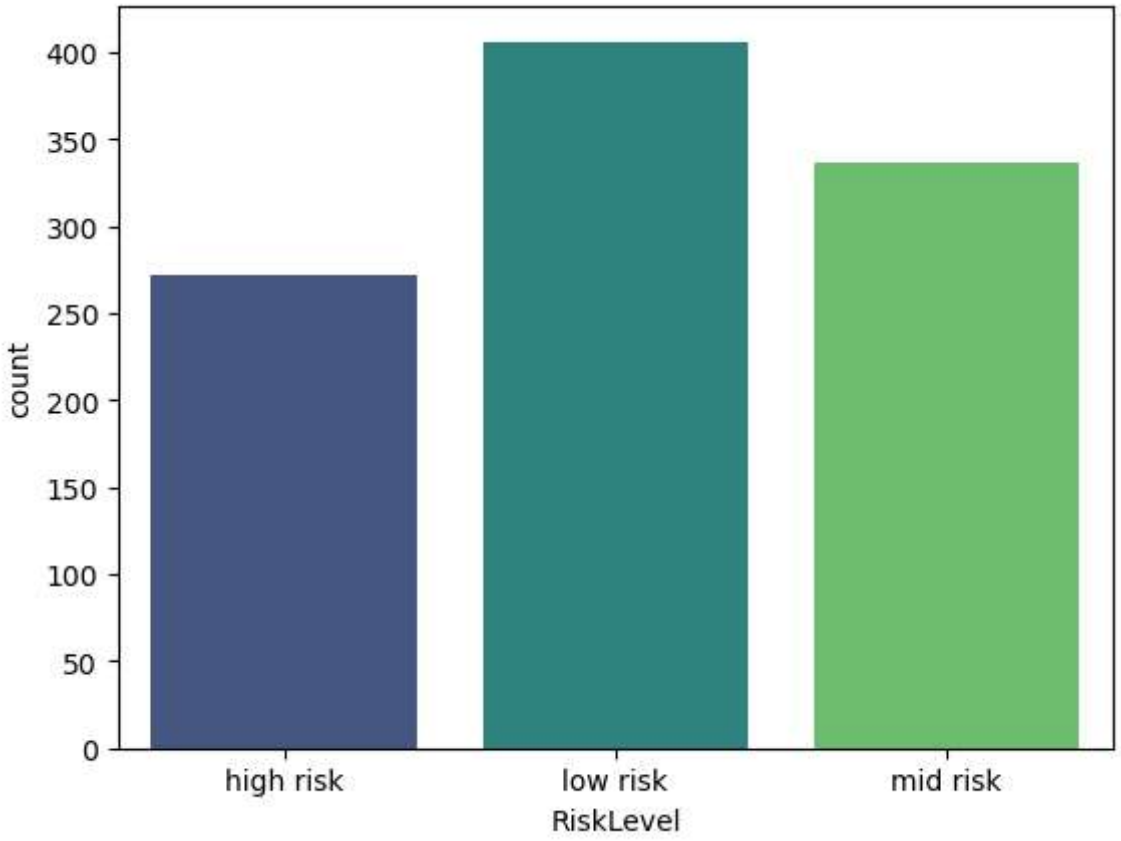
In [34]:
```
import matplotlib.pyplot as plt
df.hist(bins=50, figsize=(20, 15))
plt.show()
```



Risk Level value count

In [35]:
```
df['RiskLevel'].value_counts()

df['RiskLevel'].value_counts() * 100 / len(df)


sns.countplot(x='RiskLevel', data=df, palette='viridis')
```

Out[35]: <Axes: xlabel='RiskLevel', ylabel='count'>



In [36]:
```
X = df.drop('RiskLevel',axis=1)
y = df['RiskLevel']
```

In [37]:
```
X
```

Out[37]:

	Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate
0	25	130	80	15.0	98.0	86
1	35	140	90	13.0	98.0	70
2	29	90	70	8.0	100.0	80
3	30	140	85	7.0	98.0	70
4	35	120	60	6.1	98.0	76
...	...	...	...	...	...	...
1009	22	120	60	15.0	98.0	80
1010	55	120	90	18.0	98.0	60
1011	35	85	60	19.0	98.0	86
1012	43	120	90	18.0	98.0	70
1013	32	120	65	6.0	101.0	76

1014 rows × 6 columns

In [38]:
```
y
```

Out[38]:
```
0       high risk
1       high risk
2       high risk
3       high risk
4        low risk
          ...
1009    high risk
1010    high risk
1011    high risk
1012    high risk
1013     mid risk
Name: RiskLevel, Length: 1014, dtype: object
```

Splitting the Dataset

In [39]:
```
#Splitting The Dataset

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,
                                               random_state=42)
y_test
```

Out[39]:
```
752      mid risk
519     high risk
210     high risk
611      low risk
914      low risk
         ...
1003    high risk
227     high risk
868      mid risk
584     high risk
808      mid risk
Name: RiskLevel, Length: 305, dtype: object
```

Testing

In [40]:
```
print(f'Training Shape x:',X_train.shape)
print(f'Testing Shape x:',X_test.shape)
print('_____________________________')
print(f'Training Shape y:',X.shape)
print(f'Testing Shape y:',y.shape)
```

```
Training Shape x: (709, 6)
Testing Shape x: (305, 6)

_____________________________
Training Shape y: (1014, 6)
Testing Shape y: (1014,)
```

Standard Scaler

In [41]:
```
#StandardScaler

ss = StandardScaler()

X_train = ss.fit_transform(X_train)

X_test= ss.transform(X_test)
```

Support Vector classification

In [42]:
```
#Applying SVC (Support Vector Classification)
from sklearn.svm import SVC

#Create svm
svm = SVC(kernel='rbf', random_state=0, gamma=.10, C=1.0)
svm.fit(X_train, y_train)
print("Train accuracy:",svm.score(X_train,y_train))
print("Test accuracy:",svm.score(X_test,y_test))

y_pred = svm.predict(X_test)
print(y_pred)
cm = confusion_matrix(y_test, y_pred)
print(f'CM:',cm)
print(f'Accuracy:',accuracy_score(y_test, y_pred)* 100 ,'%')
print(classification_report(y_test, svm.predict(X_test)))
```

```
Train accuracy: 0.7108603667136812
Test accuracy: 0.6885245901639344
['mid risk' 'high risk' 'high risk' 'low risk' 'mid risk' 'mid risk'
 'low risk' 'mid risk' 'mid risk' 'high risk' 'high risk' 'low risk'
 'low risk' 'high risk' 'mid risk' 'low risk' 'high risk' 'low risk'
 'low risk' 'mid risk' 'low risk' 'mid risk' 'low risk' 'high risk'
 'low risk' 'mid risk' 'high risk' 'low risk' 'low risk' 'low risk'
 'high risk' 'high risk' 'low risk' 'low risk' 'low risk' 'high risk'
 'high risk' 'low risk' 'low risk' 'high risk' 'high risk' 'mid risk'
 'high risk' 'high risk' 'low risk' 'mid risk' 'high risk' 'mid risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'low risk' 'low risk'
 'low risk' 'mid risk' 'high risk' 'low risk' 'low risk' 'low risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'low risk' 'high risk'
 'high risk' 'low risk' 'low risk' 'high risk' 'high risk' 'high risk'
 'high risk' 'mid risk' 'mid risk' 'high risk' 'low risk' 'high risk'
 'high risk' 'low risk' 'low risk' 'low risk' 'low risk' 'low risk'
 'low risk' 'low risk' 'low risk' 'mid risk' 'high risk' 'low risk'
 'mid risk' 'mid risk' 'low risk' 'high risk' 'low risk' 'low risk'
 'low risk' 'low risk' 'high risk' 'low risk' 'low risk' 'mid risk'
 'mid risk' 'low risk' 'low risk' 'low risk' 'low risk' 'mid risk'
 'low risk' 'mid risk' 'mid risk' 'mid risk' 'low risk' 'high risk'
 'high risk' 'low risk' 'low risk' 'high risk' 'low risk' 'high risk'
 'mid risk' 'low risk' 'low risk' 'low risk' 'low risk' 'high risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'low risk' 'low risk'
 'low risk' 'mid risk' 'low risk' 'low risk' 'high risk' 'low risk'
 'low risk' 'mid risk' 'high risk' 'mid risk' 'high risk' 'high risk'
 'low risk' 'low risk' 'high risk' 'low risk' 'low risk' 'low risk'
 'low risk' 'low risk' 'low risk' 'mid risk' 'high risk' 'mid risk'
 'low risk' 'low risk' 'low risk' 'high risk' 'low risk' 'low risk'
 'low risk' 'mid risk' 'low risk' 'mid risk' 'high risk' 'low risk'
 'low risk' 'mid risk' 'high risk' 'low risk' 'low risk' 'low risk'
 'low risk' 'high risk' 'mid risk' 'low risk' 'low risk' 'low risk'
 'high risk' 'low risk' 'low risk' 'high risk' 'high risk' 'low risk'
 'high risk' 'low risk' 'low risk' 'mid risk' 'mid risk' 'low risk'
 'mid risk' 'low risk' 'low risk' 'low risk' 'low risk' 'high risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'high risk' 'mid risk'
 'mid risk' 'low risk' 'mid risk' 'high risk' 'low risk' 'mid risk'
 'high risk' 'low risk' 'low risk' 'low risk' 'low risk' 'high risk'
 'low risk' 'high risk' 'low risk' 'low risk' 'high risk' 'high risk'
 'mid risk' 'mid risk' 'low risk' 'mid risk' 'low risk' 'low risk'
 'high risk' 'low risk' 'low risk' 'low risk' 'mid risk' 'low risk'
 'mid risk' 'mid risk' 'low risk' 'low risk' 'mid risk' 'mid risk'
 'high risk' 'low risk' 'low risk' 'low risk' 'mid risk' 'high risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'high risk' 'high risk'
 'mid risk' 'low risk' 'mid risk' 'low risk' 'low risk' 'high risk'
 'high risk' 'high risk' 'low risk' 'low risk' 'low risk' 'low risk'
 'mid risk' 'high risk' 'low risk' 'low risk' 'high risk' 'low risk'
 'mid risk' 'high risk' 'low risk' 'low risk' 'high risk' 'mid risk'
 'low risk' 'low risk' 'high risk' 'mid risk' 'low risk' 'high risk'
 'low risk' 'high risk' 'high risk' 'low risk' 'high risk' 'low risk'
 'high risk' 'mid risk' 'mid risk' 'low risk' 'low risk' 'low risk'
 'low risk' 'high risk' 'high risk' 'low risk' 'mid risk' 'low risk'
 'high risk' 'mid risk' 'mid risk' 'high risk' 'low risk']
CM: [[ 63   3  10]
 [  5 104   8]
 [ 11  58  43]]
Accuracy: 68.85245901639344 %
```

	precision	recall	f1-score	support
high risk	0.80	0.83	0.81	76
low risk	0.63	0.89	0.74	117
mid risk	0.70	0.38	0.50	112
accuracy			0.69	305
macro avg	0.71	0.70	0.68	305
weighted avg	0.70	0.69	0.67	305

Decision Tree

In [43]:
```
from sklearn import tree

#Create tree object
decision_tree = tree.DecisionTreeClassifier(criterion='gini')

#Train DT based on scaled training set
decision_tree.fit(X_train, y_train)

print("Train accuracy:",decision_tree.score(X_train,y_train))
print("Test accuracy:",decision_tree.score(X_test,y_test))

y_pred = decision_tree.predict(X_test)
print(y_pred)
cm = confusion_matrix(y_test, y_pred)
print(f'CM:',cm)
print(f'Accuracy:',accuracy_score(y_test, y_pred)* 100 ,'%')
print(classification_report(y_test, decision_tree.predict(X_test)))
```

```
Train accuracy: 0.9464033850493653
Test accuracy: 0.8
['mid risk' 'high risk' 'high risk' 'low risk' 'low risk' 'mid risk'
 'mid risk' 'mid risk' 'mid risk' 'low risk' 'high risk' 'mid risk'
 'mid risk' 'high risk' 'mid risk' 'mid risk' 'high risk' 'low risk'
 'low risk' 'mid risk' 'mid risk' 'mid risk' 'high risk' 'high risk'
 'low risk' 'mid risk' 'high risk' 'low risk' 'low risk' 'low risk'
 'mid risk' 'high risk' 'low risk' 'mid risk' 'low risk' 'high risk'
 'high risk' 'low risk' 'low risk' 'high risk' 'high risk' 'low risk'
 'high risk' 'high risk' 'low risk' 'mid risk' 'high risk' 'mid risk'
 'low risk' 'low risk' 'mid risk' 'low risk' 'low risk' 'mid risk'
 'mid risk' 'mid risk' 'high risk' 'low risk' 'low risk' 'mid risk'
 'low risk' 'mid risk' 'mid risk' 'low risk' 'low risk' 'high risk'
 'mid risk' 'high risk' 'mid risk' 'high risk' 'high risk' 'high risk'
 'mid risk' 'mid risk' 'mid risk' 'high risk' 'low risk' 'mid risk'
 'high risk' 'low risk' 'low risk' 'mid risk' 'low risk' 'low risk'
 'mid risk' 'low risk' 'low risk' 'mid risk' 'high risk' 'mid risk'
 'mid risk' 'high risk' 'low risk' 'high risk' 'mid risk' 'low risk'
 'mid risk' 'low risk' 'high risk' 'mid risk' 'low risk' 'high risk'
 'low risk' 'low risk' 'mid risk' 'mid risk' 'low risk' 'mid risk'
 'low risk' 'mid risk' 'mid risk' 'mid risk' 'low risk' 'high risk'
 'mid risk' 'low risk' 'low risk' 'high risk' 'mid risk' 'high risk'
 'mid risk' 'low risk' 'low risk' 'mid risk' 'high risk' 'high risk'
 'mid risk' 'mid risk' 'mid risk' 'mid risk' 'low risk' 'low risk'
 'mid risk' 'mid risk' 'low risk' 'low risk' 'high risk' 'mid risk'
 'low risk' 'mid risk' 'high risk' 'mid risk' 'high risk' 'high risk'
 'low risk' 'low risk' 'high risk' 'low risk' 'low risk' 'mid risk'
 'low risk' 'mid risk' 'low risk' 'mid risk' 'high risk' 'mid risk'
 'low risk' 'mid risk' 'low risk' 'high risk' 'low risk' 'low risk'
 'low risk' 'mid risk' 'low risk' 'mid risk' 'mid risk' 'mid risk'
 'low risk' 'low risk' 'high risk' 'low risk' 'mid risk' 'low risk'
 'mid risk' 'high risk' 'mid risk' 'mid risk' 'low risk' 'low risk'
 'high risk' 'mid risk' 'low risk' 'high risk' 'high risk' 'low risk'
 'high risk' 'low risk' 'mid risk' 'low risk' 'mid risk' 'mid risk'
 'mid risk' 'mid risk' 'low risk' 'mid risk' 'mid risk' 'high risk'
 'low risk' 'mid risk' 'mid risk' 'mid risk' 'high risk' 'high risk'
 'mid risk' 'mid risk' 'mid risk' 'high risk' 'mid risk' 'mid risk'
 'high risk' 'mid risk' 'low risk' 'low risk' 'mid risk' 'high risk'
 'mid risk' 'high risk' 'low risk' 'low risk' 'high risk' 'high risk'
 'mid risk' 'mid risk' 'mid risk' 'mid risk' 'mid risk' 'low risk'
 'high risk' 'mid risk' 'low risk' 'mid risk' 'mid risk' 'low risk'
 'mid risk' 'mid risk' 'low risk' 'low risk' 'mid risk' 'high risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'mid risk' 'high risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'high risk' 'high risk'
 'high risk' 'mid risk' 'high risk' 'low risk' 'low risk' 'high risk'
 'high risk' 'high risk' 'mid risk' 'mid risk' 'mid risk' 'mid risk'
 'mid risk' 'high risk' 'mid risk' 'mid risk' 'high risk' 'low risk'
 'high risk' 'high risk' 'low risk' 'mid risk' 'high risk' 'high risk'
 'low risk' 'low risk' 'high risk' 'low risk' 'mid risk' 'low risk'
 'low risk' 'high risk' 'high risk' 'mid risk' 'high risk' 'low risk'
 'mid risk' 'mid risk' 'high risk' 'mid risk' 'low risk' 'low risk'
 'low risk' 'high risk' 'high risk' 'low risk' 'mid risk' 'low risk'
 'high risk' 'high risk' 'mid risk' 'high risk' 'mid risk']
CM: [[69  1  6]
 [ 4 88 25]
 [ 9 16 87]]
Accuracy: 80.0 %
```

	precision	recall	f1-score	support
high risk	0.84	0.91	0.87	76
low risk	0.84	0.75	0.79	117
mid risk	0.74	0.78	0.76	112
accuracy			0.80	305
macro avg	0.81	0.81	0.81	305
weighted avg	0.80	0.80	0.80	305

Random Forest

In [44]:
```
#Applying RandomForest
from sklearn.ensemble import RandomForestClassifier

#Create Random Forest object
random_forest = RandomForestClassifier()

#Train model
random_forest.fit(X_train, y_train)

print("Train accuracy:",random_forest.score(X_train,y_train))
print("Test accuracy:",random_forest.score(X_test,y_test))

y_pred = random_forest.predict(X_test)
print(y_pred)
cm = confusion_matrix(y_test, y_pred)
print(f'CM:',cm)
print(f'Accuracy:',accuracy_score(y_test, y_pred)* 100 ,'%')
print(classification_report(y_test, random_forest.predict(X_test)))
```

```
Train accuracy: 0.9464033850493653
Test accuracy: 0.8032786885245902
['mid risk' 'high risk' 'high risk' 'low risk' 'low risk' 'mid risk'
 'mid risk' 'mid risk' 'mid risk' 'low risk' 'high risk' 'mid risk'
 'mid risk' 'high risk' 'mid risk' 'mid risk' 'high risk' 'low risk'
 'low risk' 'mid risk' 'mid risk' 'mid risk' 'high risk' 'high risk'
 'low risk' 'mid risk' 'high risk' 'mid risk' 'low risk' 'low risk'
 'mid risk' 'high risk' 'low risk' 'mid risk' 'high risk' 'high risk'
 'high risk' 'mid risk' 'low risk' 'high risk' 'high risk' 'low risk'
 'high risk' 'high risk' 'low risk' 'mid risk' 'high risk' 'mid risk'
 'low risk' 'low risk' 'mid risk' 'low risk' 'low risk' 'mid risk'
 'mid risk' 'mid risk' 'high risk' 'low risk' 'low risk' 'mid risk'
 'low risk' 'mid risk' 'mid risk' 'mid risk' 'low risk' 'high risk'
 'mid risk' 'high risk' 'mid risk' 'high risk' 'high risk' 'high risk'
 'mid risk' 'mid risk' 'mid risk' 'high risk' 'low risk' 'mid risk'
 'high risk' 'low risk' 'low risk' 'mid risk' 'low risk' 'low risk'
 'mid risk' 'low risk' 'low risk' 'mid risk' 'high risk' 'mid risk'
 'mid risk' 'high risk' 'low risk' 'high risk' 'mid risk' 'low risk'
 'mid risk' 'low risk' 'high risk' 'mid risk' 'low risk' 'high risk'
 'low risk' 'low risk' 'mid risk' 'mid risk' 'low risk' 'mid risk'
 'low risk' 'mid risk' 'mid risk' 'mid risk' 'low risk' 'high risk'
 'mid risk' 'low risk' 'low risk' 'high risk' 'low risk' 'high risk'
 'mid risk' 'low risk' 'low risk' 'mid risk' 'high risk' 'high risk'
 'mid risk' 'mid risk' 'mid risk' 'mid risk' 'low risk' 'low risk'
 'mid risk' 'mid risk' 'low risk' 'low risk' 'high risk' 'mid risk'
 'low risk' 'mid risk' 'high risk' 'mid risk' 'high risk' 'high risk'
 'low risk' 'low risk' 'high risk' 'low risk' 'low risk' 'mid risk'
 'low risk' 'mid risk' 'low risk' 'mid risk' 'high risk' 'mid risk'
 'low risk' 'mid risk' 'low risk' 'high risk' 'low risk' 'low risk'
 'low risk' 'mid risk' 'low risk' 'mid risk' 'mid risk' 'low risk'
 'low risk' 'low risk' 'high risk' 'low risk' 'mid risk' 'low risk'
 'mid risk' 'high risk' 'mid risk' 'low risk' 'low risk' 'low risk'
 'high risk' 'mid risk' 'low risk' 'high risk' 'high risk' 'low risk'
 'high risk' 'low risk' 'low risk' 'low risk' 'mid risk' 'mid risk'
 'mid risk' 'mid risk' 'low risk' 'low risk' 'mid risk' 'high risk'
 'low risk' 'mid risk' 'mid risk' 'mid risk' 'high risk' 'high risk'
 'mid risk' 'mid risk' 'mid risk' 'high risk' 'mid risk' 'mid risk'
 'high risk' 'mid risk' 'low risk' 'low risk' 'mid risk' 'high risk'
 'mid risk' 'high risk' 'low risk' 'low risk' 'high risk' 'high risk'
 'mid risk' 'mid risk' 'mid risk' 'mid risk' 'mid risk' 'low risk'
 'high risk' 'mid risk' 'low risk' 'mid risk' 'mid risk' 'low risk'
 'mid risk' 'mid risk' 'low risk' 'low risk' 'mid risk' 'high risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'mid risk' 'high risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'high risk' 'high risk'
 'high risk' 'mid risk' 'high risk' 'low risk' 'low risk' 'high risk'
 'high risk' 'high risk' 'mid risk' 'mid risk' 'mid risk' 'mid risk'
 'mid risk' 'high risk' 'mid risk' 'mid risk' 'high risk' 'mid risk'
 'high risk' 'high risk' 'low risk' 'mid risk' 'high risk' 'high risk'
 'low risk' 'low risk' 'high risk' 'low risk' 'low risk' 'low risk'
 'low risk' 'high risk' 'high risk' 'low risk' 'high risk' 'low risk'
 'mid risk' 'mid risk' 'high risk' 'mid risk' 'low risk' 'low risk'
 'low risk' 'high risk' 'high risk' 'low risk' 'mid risk' 'low risk'
 'high risk' 'high risk' 'mid risk' 'high risk' 'mid risk']
CM: [[69  2  5]
 [ 5 89 23]
 [ 9 16 87]]
Accuracy: 80.32786885245902 %
```

	precision	recall	f1-score	support
high risk	0.83	0.91	0.87	76
low risk	0.83	0.76	0.79	117
mid risk	0.76	0.78	0.77	112
accuracy			0.80	305
macro avg	0.81	0.82	0.81	305
weighted avg	0.80	0.80	0.80	305

GaussianNB

In [45]:
```
#Applying GaussianNB
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(X_train , y_train)

print("Train accuracy:",nb.score(X_train,y_train))
print("Test accuracy:",nb.score(X_test,y_test))

y_pred = nb.predict(X_test)
print(y_pred)
cm = confusion_matrix(y_test, y_pred)
print(f'CM:',cm)
print(f'Accuracy:',accuracy_score(y_test, y_pred)* 100 ,'%')
print(classification_report(y_test, nb.predict(X_test)))
```

```
Train accuracy: 0.61212976022567
Test accuracy: 0.5639344262295082
['low risk' 'high risk' 'high risk' 'low risk' 'mid risk' 'mid risk'
 'low risk' 'mid risk' 'low risk' 'low risk' 'high risk' 'low risk'
 'low risk' 'low risk' 'mid risk' 'low risk' 'low risk' 'low risk'
 'low risk' 'low risk' 'low risk' 'mid risk' 'low risk' 'high risk'
 'low risk' 'low risk' 'high risk' 'low risk' 'low risk' 'low risk'
 'mid risk' 'high risk' 'low risk' 'low risk' 'low risk' 'high risk'
 'high risk' 'low risk' 'low risk' 'high risk' 'high risk' 'mid risk'
 'high risk' 'low risk' 'low risk' 'mid risk' 'high risk' 'low risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'low risk' 'low risk'
 'low risk' 'mid risk' 'low risk' 'low risk' 'low risk' 'low risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'low risk' 'high risk'
 'high risk' 'low risk' 'low risk' 'mid risk' 'low risk' 'high risk'
 'high risk' 'low risk' 'low risk' 'high risk' 'low risk' 'high risk'
 'high risk' 'low risk' 'low risk' 'low risk' 'low risk' 'low risk'
 'low risk' 'low risk' 'low risk' 'mid risk' 'high risk' 'low risk'
 'mid risk' 'mid risk' 'low risk' 'low risk' 'low risk' 'low risk'
 'mid risk' 'low risk' 'high risk' 'low risk' 'low risk' 'mid risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'low risk' 'mid risk'
 'low risk' 'low risk' 'mid risk' 'low risk' 'low risk' 'high risk'
 'high risk' 'mid risk' 'low risk' 'high risk' 'low risk' 'high risk'
 'mid risk' 'low risk' 'low risk' 'low risk' 'low risk' 'high risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'low risk' 'low risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'high risk' 'low risk'
 'low risk' 'low risk' 'high risk' 'mid risk' 'low risk' 'high risk'
 'low risk' 'low risk' 'high risk' 'low risk' 'low risk' 'low risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'mid risk' 'low risk'
 'low risk' 'low risk' 'low risk' 'high risk' 'low risk' 'low risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'high risk' 'low risk'
 'low risk' 'low risk' 'high risk' 'low risk' 'low risk' 'low risk'
 'low risk' 'high risk' 'mid risk' 'low risk' 'low risk' 'low risk'
 'mid risk' 'low risk' 'low risk' 'high risk' 'high risk' 'low risk'
 'high risk' 'low risk' 'low risk' 'low risk' 'low risk' 'low risk'
 'mid risk' 'low risk' 'low risk' 'low risk' 'low risk' 'high risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'low risk' 'mid risk'
 'low risk' 'low risk' 'low risk' 'high risk' 'low risk' 'low risk'
 'high risk' 'low risk' 'low risk' 'low risk' 'low risk' 'mid risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'high risk' 'high risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'low risk' 'low risk'
 'high risk' 'low risk' 'low risk' 'low risk' 'mid risk' 'low risk'
 'high risk' 'low risk' 'low risk' 'low risk' 'mid risk' 'low risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'mid risk' 'high risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'low risk' 'high risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'low risk' 'high risk'
 'high risk' 'high risk' 'low risk' 'low risk' 'low risk' 'low risk'
 'low risk' 'high risk' 'low risk' 'low risk' 'mid risk' 'low risk'
 'low risk' 'mid risk' 'low risk' 'low risk' 'high risk' 'low risk'
 'low risk' 'low risk' 'high risk' 'low risk' 'low risk' 'low risk'
 'low risk' 'high risk' 'mid risk' 'low risk' 'high risk' 'low risk'
 'high risk' 'low risk' 'mid risk' 'low risk' 'low risk' 'low risk'
 'low risk' 'high risk' 'high risk' 'low risk' 'low risk' 'low risk'
 'high risk' 'low risk' 'low risk' 'high risk' 'low risk']
CM: [[ 46  19  11]
 [  3 109   5]
 [ 10  85  17]]
Accuracy: 56.393442622950815 %
```

```
              precision    recall  f1-score   support

   high risk       0.78      0.61      0.68        76
    low risk       0.51      0.93      0.66       117
    mid risk       0.52      0.15      0.23       112

    accuracy                           0.56       305
   macro avg       0.60      0.56      0.53       305
weighted avg       0.58      0.56      0.51       305
```

Overall Comparison:

Best Performer: Random Forest has the highest test accuracy among the models, making it the best performer in this scenario.

In [49]:
```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

random_forest = RandomForestClassifier()

grid_search = GridSearchCV(random_forest, param_grid, cv=5, scoring='accuracy'

grid_search.fit(X_train, y_train)

best_params = grid_search.best_params_
print("Best Parameters:", best_params)

best_random_forest = RandomForestClassifier(**best_params)
best_random_forest.fit(X_train, y_train)

print("Train accuracy:", best_random_forest.score(X_train, y_train))
print("Test accuracy:", best_random_forest.score(X_test, y_test))
```

```
Best Parameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_spli
t': 2, 'n_estimators': 100}
Train accuracy: 0.9464033850493653
Test accuracy: 0.8
```

MODEL SELECTED - Random Forest

Hyperparameter Tuning

In [11]:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, classification_r
from sklearn.model_selection import GridSearchCV

X = df.drop('RiskLevel', axis=1)
y = df['RiskLevel']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rando

print(f'Training Shape x:', X_train.shape)
print(f'Testing Shape x:', X_test.shape)
print('_____')
print(f'Training Shape y:', X.shape)
print(f'Testing Shape y:', y.shape)

ss = StandardScaler()
X_train = ss.fit_transform(X_train)
X_test = ss.transform(X_test)

# Applying RandomForest
random_forest = RandomForestClassifier()
random_forest.fit(X_train, y_train)

print("Train accuracy:", random_forest.score(X_train, y_train))
print("Test accuracy:", random_forest.score(X_test, y_test))

y_pred = random_forest.predict(X_test)
print(y_pred)
cm = confusion_matrix(y_test, y_pred)
print(f'CM:', cm)
print(f'Accuracy:', accuracy_score(y_test, y_pred) * 100, '%')
print(classification_report(y_test, random_forest.predict(X_test)))

# Hyperparameter tuning with GridSearchCV
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

rf_classifier = RandomForestClassifier()
grid_search = GridSearchCV(estimator=rf_classifier, param_grid=param_grid, cv=
grid_search.fit(X_train, y_train)

print("Hyperparameters:")
print(grid_search.best_params_)

best_rf_classifier = grid_search.best_estimator_
best_rf_classifier.fit(X_train, y_train)

print("Train accuracy:", best_rf_classifier.score(X_train, y_train))
print("Test accuracy:", best_rf_classifier.score(X_test, y_test))

y_pred = best_rf_classifier.predict(X_test)
```

```
print("CM:", confusion_matrix(y_test, y_pred))
print("Accuracy:", accuracy_score(y_test, y_pred) * 100, "%")
print(classification_report(y_test, y_pred))
```

```
Training Shape x: (709, 6)
Testing Shape x: (305, 6)

_____
Training Shape y: (1014, 6)
Testing Shape y: (1014,)
Train accuracy: 0.9464033850493653
Test accuracy: 0.8098360655737705
['mid risk' 'high risk' 'high risk' 'low risk' 'low risk' 'mid risk'
 'mid risk' 'mid risk' 'mid risk' 'low risk' 'high risk' 'mid risk'
 'mid risk' 'high risk' 'mid risk' 'mid risk' 'high risk' 'low risk'
 'low risk' 'mid risk' 'mid risk' 'mid risk' 'high risk' 'high risk'
 'low risk' 'mid risk' 'high risk' 'mid risk' 'low risk' 'low risk'
 'mid risk' 'high risk' 'low risk' 'mid risk' 'high risk' 'high risk'
 'high risk' 'mid risk' 'low risk' 'high risk' 'high risk' 'low risk'
 'high risk' 'high risk' 'low risk' 'mid risk' 'high risk' 'mid risk'
 'low risk' 'low risk' 'mid risk' 'low risk' 'mid risk' 'mid risk'
 'mid risk' 'mid risk' 'high risk' 'low risk' 'low risk' 'mid risk'
 'low risk' 'mid risk' 'mid risk' 'mid risk' 'low risk' 'high risk'
 'mid risk' 'high risk' 'mid risk' 'high risk' 'high risk' 'high risk'
 'mid risk' 'mid risk' 'mid risk' 'high risk' 'low risk' 'mid risk'
 'high risk' 'low risk' 'low risk' 'mid risk' 'low risk' 'low risk'
 'mid risk' 'low risk' 'low risk' 'mid risk' 'high risk' 'mid risk'
 'mid risk' 'high risk' 'low risk' 'high risk' 'mid risk' 'low risk'
 'mid risk' 'mid risk' 'high risk' 'mid risk' 'low risk' 'high risk'
 'low risk' 'low risk' 'mid risk' 'mid risk' 'low risk' 'mid risk'
 'low risk' 'mid risk' 'mid risk' 'mid risk' 'low risk' 'high risk'
 'mid risk' 'low risk' 'low risk' 'high risk' 'low risk' 'high risk'
 'mid risk' 'low risk' 'low risk' 'mid risk' 'high risk' 'high risk'
 'mid risk' 'mid risk' 'mid risk' 'mid risk' 'low risk' 'low risk'
 'mid risk' 'mid risk' 'low risk' 'low risk' 'high risk' 'mid risk'
 'mid risk' 'mid risk' 'high risk' 'mid risk' 'high risk' 'high risk'
 'low risk' 'low risk' 'high risk' 'low risk' 'low risk' 'mid risk'
 'low risk' 'mid risk' 'low risk' 'mid risk' 'high risk' 'mid risk'
 'low risk' 'mid risk' 'low risk' 'high risk' 'low risk' 'low risk'
 'low risk' 'mid risk' 'low risk' 'mid risk' 'mid risk' 'low risk'
 'low risk' 'low risk' 'high risk' 'low risk' 'mid risk' 'low risk'
 'mid risk' 'high risk' 'mid risk' 'low risk' 'low risk' 'low risk'
 'high risk' 'mid risk' 'low risk' 'high risk' 'high risk' 'low risk'
 'high risk' 'mid risk' 'low risk' 'low risk' 'mid risk' 'mid risk'
 'mid risk' 'mid risk' 'mid risk' 'low risk' 'mid risk' 'high risk'
 'low risk' 'mid risk' 'mid risk' 'mid risk' 'high risk' 'high risk'
 'mid risk' 'mid risk' 'mid risk' 'high risk' 'mid risk' 'mid risk'
 'high risk' 'mid risk' 'low risk' 'low risk' 'mid risk' 'high risk'
 'mid risk' 'high risk' 'low risk' 'low risk' 'high risk' 'high risk'
 'mid risk' 'mid risk' 'mid risk' 'mid risk' 'mid risk' 'low risk'
 'high risk' 'mid risk' 'low risk' 'mid risk' 'mid risk' 'low risk'
 'mid risk' 'mid risk' 'low risk' 'low risk' 'mid risk' 'high risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'mid risk' 'high risk'
 'low risk' 'low risk' 'low risk' 'low risk' 'high risk' 'high risk'
 'high risk' 'mid risk' 'high risk' 'low risk' 'low risk' 'high risk'
 'high risk' 'high risk' 'mid risk' 'mid risk' 'mid risk' 'mid risk'
 'mid risk' 'high risk' 'mid risk' 'mid risk' 'high risk' 'low risk'
 'high risk' 'high risk' 'low risk' 'mid risk' 'high risk' 'high risk'
 'low risk' 'low risk' 'high risk' 'low risk' 'low risk' 'low risk'
 'low risk' 'high risk' 'high risk' 'low risk' 'high risk' 'low risk'
 'mid risk' 'mid risk' 'high risk' 'mid risk' 'low risk' 'low risk'
 'low risk' 'high risk' 'high risk' 'low risk' 'mid risk' 'low risk'
```

```
 'high risk' 'high risk' 'mid risk' 'high risk' 'mid risk']
CM: [[69  2  5]
 [ 5 88 24]
 [ 9 13 90]]
Accuracy: 80.98360655737706 %
              precision    recall  f1-score   support

   high risk       0.83      0.91      0.87        76
    low risk       0.85      0.75      0.80       117
    mid risk       0.76      0.80      0.78       112

    accuracy                           0.81       305
   macro avg       0.81      0.82      0.82       305
weighted avg       0.81      0.81      0.81       305

Hyperparameters:
{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimat
ors': 200}
Train accuracy: 0.9464033850493653
Test accuracy: 0.8
CM: [[69  2  5]
 [ 5 88 24]
 [ 9 16 87]]
Accuracy: 80.0 %
              precision    recall  f1-score   support

   high risk       0.83      0.91      0.87        76
    low risk       0.83      0.75      0.79       117
    mid risk       0.75      0.78      0.76       112

    accuracy                           0.80       305
   macro avg       0.80      0.81      0.81       305
weighted avg       0.80      0.80      0.80       305
```

In [10]:

```python
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier

new_input_data = pd.DataFrame({
    'Age': [25],
    'SystolicBP': [130],
    'DiastolicBP': [80],
    'BS': [15.0],
    'BodyTemp': [98.0],
    'HeartRate': [86]
})

new_input_data_standardized = ss.transform(new_input_data)

predictions = random_forest.predict(new_input_data_standardized)
print("Predictions:", predictions)
```

```
Predictions: ['high risk']
```

THE PREDICTION CAN BE CONSIDERED CORRECT AS:

SystolicBP: [130] A normal blood pressure is less than 120 systolic over less than 80 diastolic. A systolic blood pressure in the 120s is considered elevated, even with a normal diastolic reading. Hypertension is diagnosed with a systolic reading of 130 or higher, or a diastolic reading of 80 or higher.

DiastolicBP: [80] The normal diastolic blood pressure usually ranges between 60 to 80 mmHg. If your diastolic blood pressure reading is 80-89, you need to pay special attention because you already have pre-hypertension. Otherwise, your condition may worsen and you may develop malignant hypertension in which your diastolic blood pressure is above 140

BS: [15.0] A result of 15% suggests that 15% of the hemoglobin in your red blood cells is saturated with sugar. Results of 15% or higher on an A1c test indicate diabetes. Elevated levels of A1c indicating prediabetes are usually between 5.7–6.4%, while estimated average glucose levels are between 4–5.6%.

In [ ]: