ML Assignment

1) Understanding the Dataset:

Based on the analysis shown below we were able to find out that the independent variables are: Age, SystolicBP, DiastolicBP, BS, BodyTemp and HeartRate
and dependent or target variable is: Type

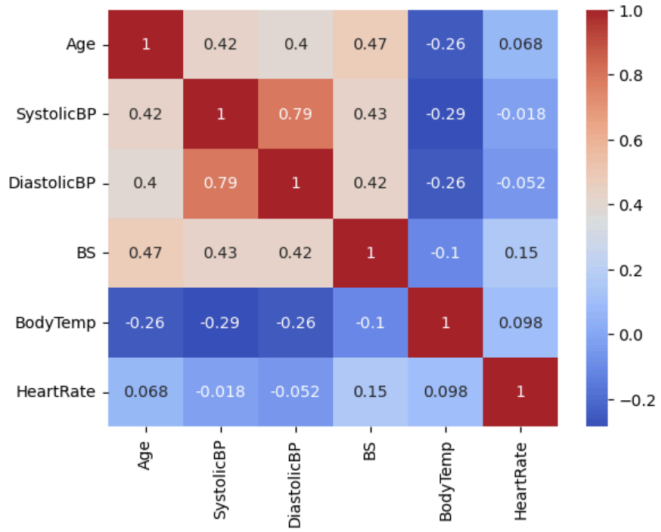
```
# getting summary of the dataset
X.describe()
```

	Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate
count	1014.000000	1014.000000	1014.000000	1014.000000	1014.000000	1014.000000
mean	29.871795	113.198225	76.460552	8.725986	98.665089	74.301775
std	13.474386	18.403913	13.885796	3.293532	1.371384	8.088702
min	10.000000	70.000000	49.000000	6.000000	98.000000	7.000000
25%	19.000000	100.000000	65.000000	6.900000	98.000000	70.000000
50%	26.000000	120.000000	80.000000	7.500000	98.000000	76.000000
75%	39.000000	120.000000	90.000000	8.000000	98.000000	80.000000
max	70.000000	160.000000	100.000000	19.000000	103.000000	90.000000

During exploration it was found that there is a high correlation between SystolicBP and DiastolicBP, which was around : 0.79 as shown in the image below:

```
[42]: # Checking for any correlations among the features
      sns.heatmap(X.corr(), annot = True, cmap='coolwarm')
```

```
[42]: <Axes: >
```



Ideally when we have high correlation between 2 columns (features) it is better to remove one of the columns to reduce the complexity of the model. In our case if we removed either of the features (SystolicBP or DiastolicBP), the accuracy of the model falls a lot. This phenomenon can be explained by the interaction effect. There are times when interaction between the features provide valuable information that gets lost once if we remove either of the columns. Hence all given above columns are considered as a part of an independent variable and 'Type' is our dependent variable.

This is a classification problem statement as in our dataset we have 3 classes/ labels mentioned as **High Risk, Mid Risk and Low Risk.**

2) Data Exploration and Preprocessing:

While performing data exploration and checking for the missing values and data imbalances based on the different classes we found out that there were no missing values, so no need do any data missing preprocessing and our classes were almost equally distributed and hence no imbalances found, this can seen as below:

```
#Checking for any na or missing values in any of the columns
df.isna().sum()
```

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

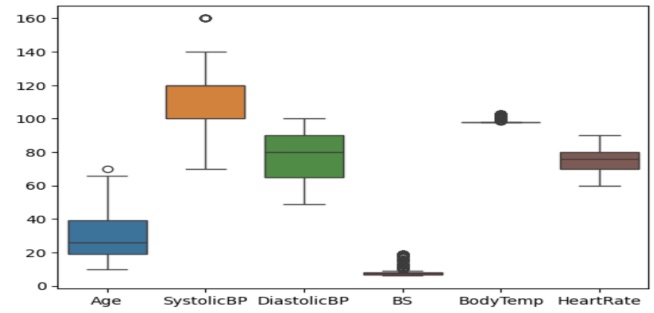
```
#checking for the proportion (percentage)
df['Type'].value_counts()/len(df)
```

```
Type
low risk     0.400394
mid risk     0.331361
high risk    0.268245
Name: count, dtype: float64
```

Although during our exploration for the outliers we saw there were quite a lot of outliers in their individual columns, and the total no. of rows having at least one column as outlier came out to be 392 rows which is almost 0.38 percent of the overall dataset. Removing a big chunk like this would lead to a loss of information, therefore we would need to have some domain knowledge in order to remove some . Although there were 2 rows which had Heart Rate which was even below 10 (lowest recorded so far is in the range of 30-40 among athletes, domain knowledge which I got from my own research over the internet), hence these 2 outliers were removed. Final output of boxplot came out to be as shown below:

```
sns.boxplot(X)
```

```
<Axes: >
```



We have performed feature scaling on the datasets using the StandardScaler function, but since we are using Decision Tree and Random Forest algorithms, it not actually necessary and had shown the results in notebook that training and test scores in case of simple dataset and dataset with feature scaling done it are similar.
Example for Decision Tree:

```
# best classifier
best_tree = grid_search.best_estimator_

print("Model Training Score: ",best_tree.score(X_train_std, y_train))
print("Model Testing Score: ", best_tree.score(X_test_std, y_test))
```

```
Model Training Score:  0.9283065512978986
Model Testing Score:  0.8374384236453202
```

```
print("Model Training Score: ",best_tree_test.score(X_train, y_train))
print("Model Testing Score: ", best_tree_test.score(X_test, y_test))
```

```
Fitting 5 folds for each of 72 candidates, totalling 360 fits
Best parameter from decision tree:  {'criterion': 'entropy', 'max_depth': None, 'min_sample
f': 1, 'min_samples_split': 2}
Model Training Score:  0.9283065512978986
Model Testing Score:  0.8325123152709359
```

Reason for this is because, scaling is effective only for those algorithms which are distance based, for tree based algorithms it's not necessary. Even if we do scaling like normalisation or standard scaling etc. we would get the same result.

3) Algorithm Selection and Application:

In my case as stated up for the classification problem I have used 2 algorithms namely Decision Tree and Random forest.

Decision Tree: A decision tree is a model that splits the data into subsets based on feature values to make predictions. It forms a tree like structure where the internal nodes represents the decision made and leaves are the outcomes (labels)

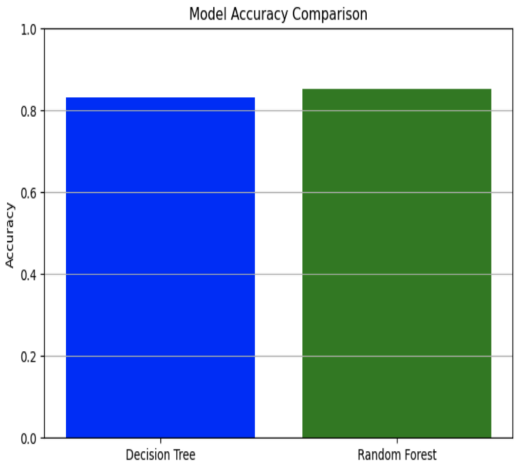
Random Forest: A Random Forest is an ensemble of decision trees (collections of decision trees). It creates multiple decision trees randomly and selects subsets of features and data points to reduce overfitting and improve accuracy.

We have applied hyperparameter tuning in both cases of Decision Trees and Random forest such as :
Decision tree -> max_depth, min_samples_split, min_samples_leaf, criterion (gini or entropy)
Random Forest -> n_estimators (no. of trees in forest) , max_depth, min_samples_split, min_samples_leaf
Max_depth - deciding how deep a tree should grow (setting a limit)
Min_samples_split - minimum number of samples required to be split an internal node
Min_samples_leaf - minimum samples required to be at the leaf node.

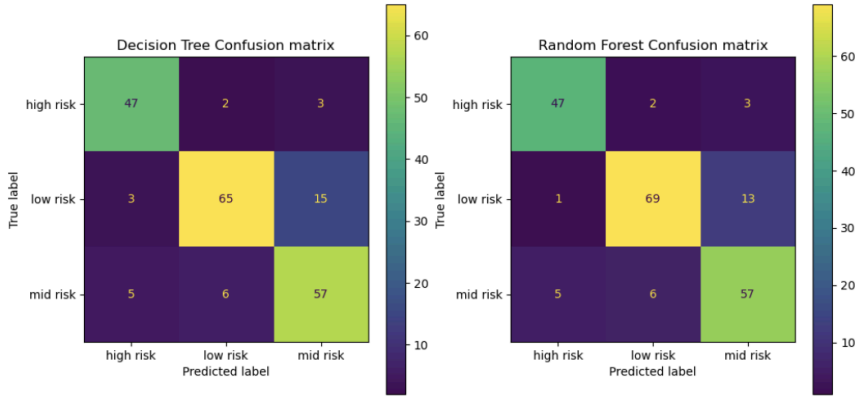
Based on the Hyperparameter tuning using GridSearchCV we found out that our score for both the model increased training ~93% and test ~83%, which was quite low when we just took random values for hyperparameter (training ~72% and test ~ 64 %) which can be seen in python notebook.

4) Model Evaluation and Comparative Analysis:

During evaluations we checked that accuracy performance for Decision Tree model and Random Forest models are quite similar, although Random Forest model performing little better as compared to the later. We can see this based on the graph shown below:



The Confusion matrix for both can be shown as below:



We see the performance of both the models for classification is quite similar to the performance of both models because the dataset is quite small in our case (~1000) and secondly Random Forest is an ensemble of Decision Trees. So if data is simple or lacks complexity then Random Forest won't show much of improvement over decision tree prediction.

5) Ethical Considerations:

We see that in the model, there are high chances that the model classifies labels like High Risk, Low Risk and Mid Risk classes correctly, but at the same time some of the classes are misclassified. Such kind of misclassification can lead to big issues. For example if a person who is completely healthy comes out to be a high risk, based on the model prediction, which could lead to unnecessary worry, stress and additional medical treatments which could later on lead to financial costs and could lead to emotional strain to mother, even though everything is fine. And conversely if a high risk patient is misclassified as healthy then both the mother and child are at risk. The patient won't be able to receive urgent care and could lead to delayed medical intervention which could be dangerous for both mother and baby.

Secondly, unlike Decision Tree, which are quite easily explainable, Random Forest acts more of like a black box model, complex and harder to interpret. It is necessary for the healthcare professionals to understand why a particular output was given, if it's not explained or interpreted out properly, then there will be a reduction in trust of model performance, no matter how accurate it may be. At the end of the day why a person is labelled as high/ low risk should be easily explainable to both the doctor and patient.

The Ethical way of using such a predictive model should be, it shouldn't completely replace clinical judgement by the health professionals. This is because if we solely depend on the models, then who is accountable for the judgement provided by the model, is it healthcare provider, developer or the institution who deployed it?

References:

https://scikit-learn.org/1.5/modules/tree.html
https://scikit-learn.org/dev/modules/generated/sklearn.ensemble.RandomForestClassifier.html