Problem Statement

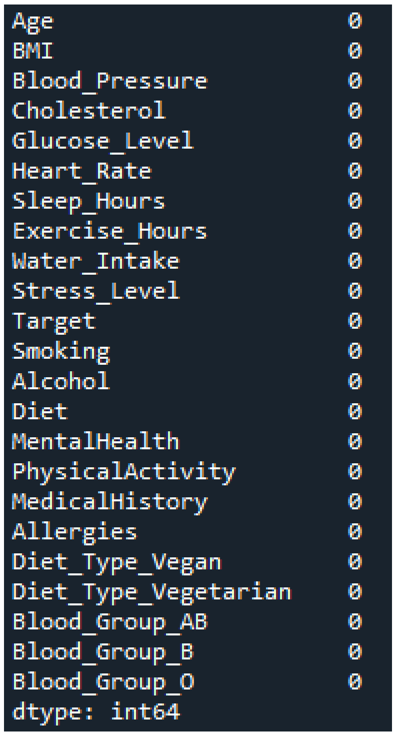
Anova Insurance, a global health insurance company, seeks to optimize its insurance policy premium pricing based on the health status of applicants. Understanding an applicant's health condition is crucial for two key decisions:
- Determining eligibility for health insurance coverage.
- Deciding on premium rates, particularly if the applicant's health indicates higher risks.

The objective is to Develop a predictive model that utilizes health data to classify individuals as 'healthy' or 'unhealthy'. This classification will assist in making informed decisions about insurance policy premium pricing.

Analysis

The analysis below contains the details which is almost the same as earlier using the KNN model or the Logistic regression model, however, the only difference is we're using a Decision Tree classifier

After reading the dataset, we print the missing values in any column - there are no missing values as seen earlier as well -

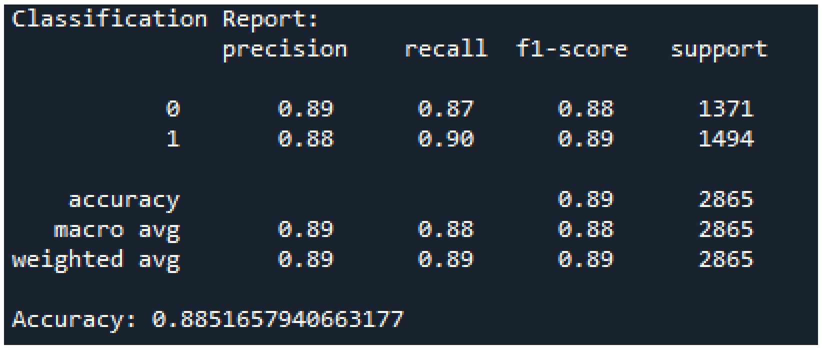


```
Age                     0
BMI                     0
Blood_Pressure          0
Cholesterol             0
Glucose_Level           0
Heart_Rate              0
Sleep_Hours             0
Exercise_Hours          0
Water_Intake            0
Stress_Level            0
Target                  0
Smoking                 0
Alcohol                 0
Diet                    0
MentalHealth            0
PhysicalActivity        0
MedicalHistory          0
Allergies               0
Diet_Type_Vegan         0
Diet_Type_Vegetarian    0
Blood_Group_AB          0
Blood_Group_B           0
Blood_Group_O           0
dtype: int64
```

We then convert the boolean data types in the diet and Blood group columns into integer form. After splitting the data into 70% train and 30% test data, a Decision Tree classifier model is chosen to fit the data.

We use a visualisation of the decision tree, however, the plot hasn't been added here due to too many features and the plot isn't easy to decipher.

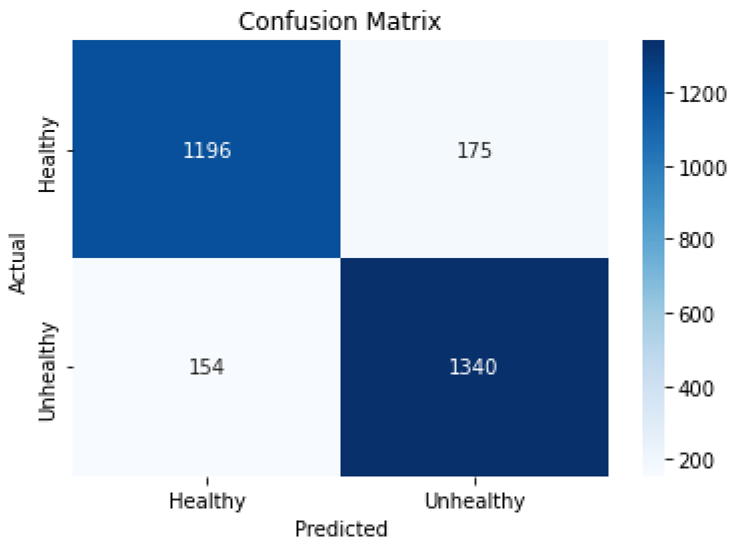
Using the entire dataset, and after having fitted the model, we look at the reports for evaluation of the model including accuracy scores, F1 scores and also the Confusion matrix.



```
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.87      0.88      1371
           1       0.88      0.90      0.89      1494

    accuracy                           0.89      2865
   macro avg       0.89      0.88      0.88      2865
weighted avg       0.89      0.89      0.89      2865

Accuracy: 0.8851657940663177
```

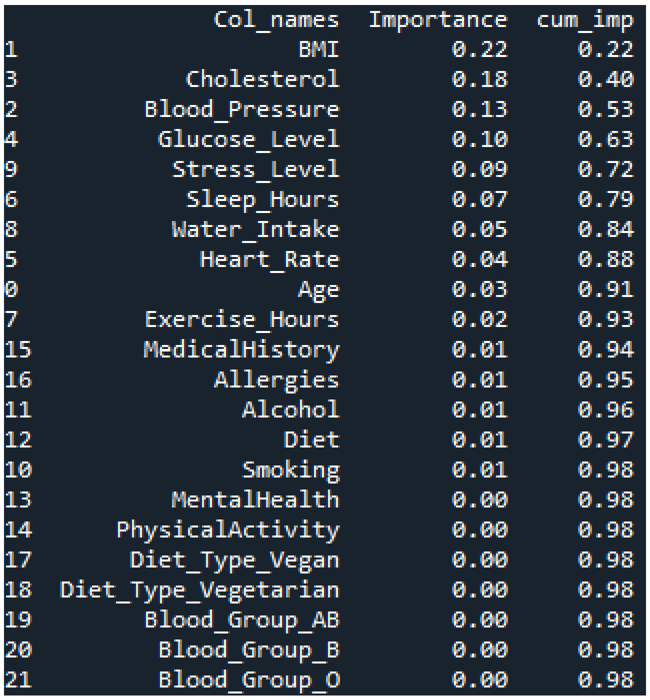
We can see a F1 score of 89% and similarly an accuracy of 88.5%, which is much higher than what we had seen earlier using a KNN model.

Similarly, the confusion matrix also shows much better results than earlier seen for the KNN model.



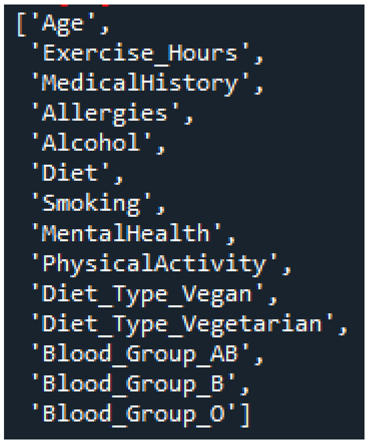
Feature Selection

After looking at the entire dataset, we look at the more important features using feature selection by importance, and use cumulative importance upto 90% to drop the less important features. As can be seen below, the list of features by their importance and cumulative importance-



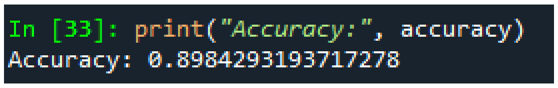
```
              Col_names  Importance  cum_imp
1                   BMI        0.22     0.22
3           Cholesterol        0.18     0.40
2        Blood_Pressure        0.13     0.53
4         Glucose_Level        0.10     0.63
9          Stress_Level        0.09     0.72
6           Sleep_Hours        0.07     0.79
8          Water_Intake        0.05     0.84
5            Heart_Rate        0.04     0.88
0                   Age        0.03     0.91
7        Exercise_Hours        0.02     0.93
15       MedicalHistory        0.01     0.94
16            Allergies        0.01     0.95
11              Alcohol        0.01     0.96
12                 Diet        0.01     0.97
10              Smoking        0.01     0.98
13         MentalHealth        0.00     0.98
14     PhysicalActivity        0.00     0.98
17      Diet_Type_Vegan        0.00     0.98
18  Diet_Type_Vegetarian       0.00     0.98
19        Blood_Group_AB       0.00     0.98
20         Blood_Group_B       0.00     0.98
21         Blood_Group_O       0.00     0.98
```

So we drop the following features -



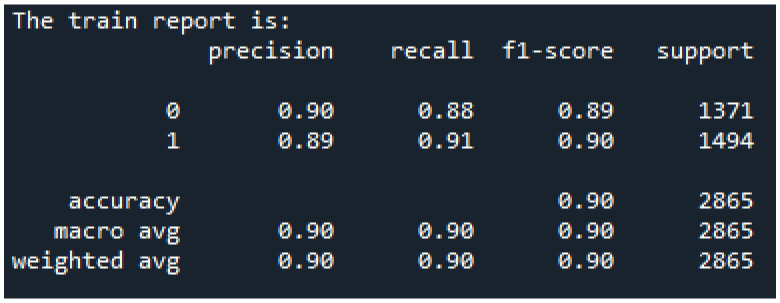
```
['Age',
 'Exercise_Hours',
 'MedicalHistory',
 'Allergies',
 'Alcohol',
 'Diet',
 'Smoking',
 'MentalHealth',
 'PhysicalActivity',
 'Diet_Type_Vegan',
 'Diet_Type_Vegetarian',
 'Blood_Group_AB',
 'Blood_Group_B',
 'Blood_Group_O']
```

We refit the model and again run the classification reports and accuracy of the model -



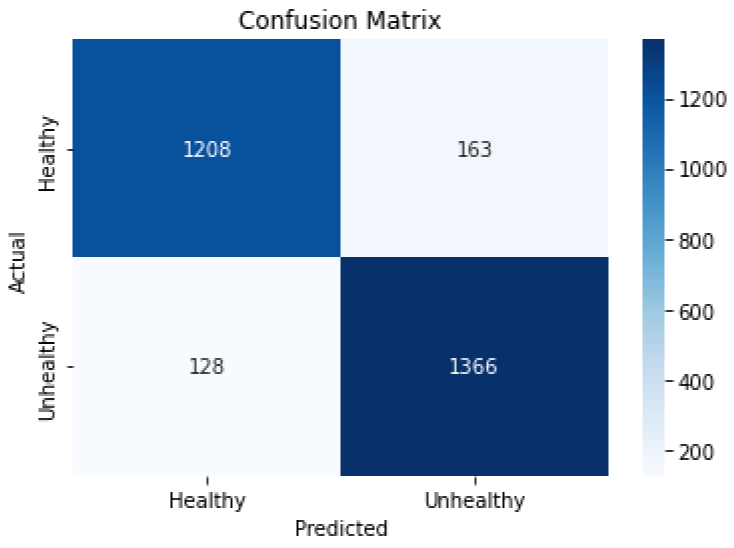
```
In [33]: print("Accuracy:", accuracy)
Accuracy: 0.8984293193717278
```

We can notice there is a slight improvement in accuracy and the F1 score after selecting the important features.



```
The train report is:
              precision    recall  f1-score   support

           0       0.90      0.88      0.89      1371
           1       0.89      0.91      0.90      1494

    accuracy                           0.90      2865
   macro avg       0.90      0.90      0.90      2865
weighted avg       0.90      0.90      0.90      2865
```

Similarly, an improvement in the confusion matrix is also observed -

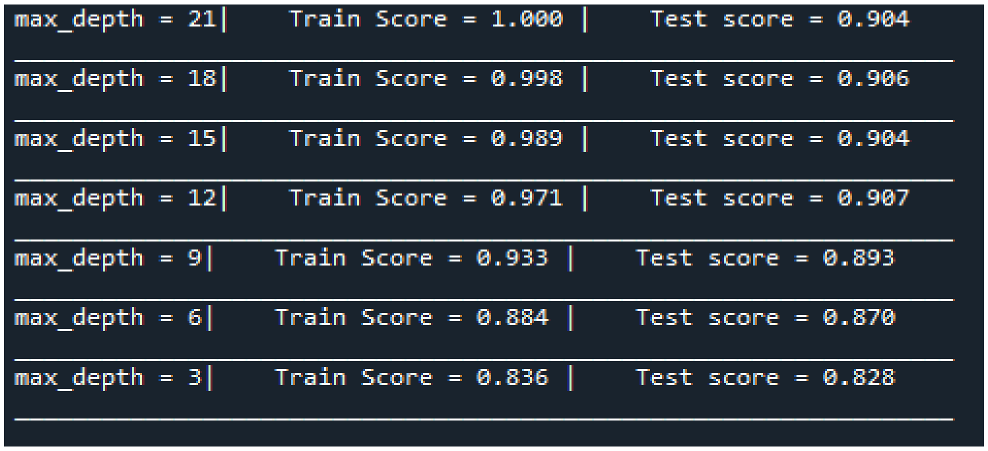


<u>Hyperparameter Tuning for Model Optimisation</u>

We look at hyperparameter tuning by using both max depth and also minimum sample leaf.

Max depth:

Using a list of different max depths, we look at how the train and test scores change -

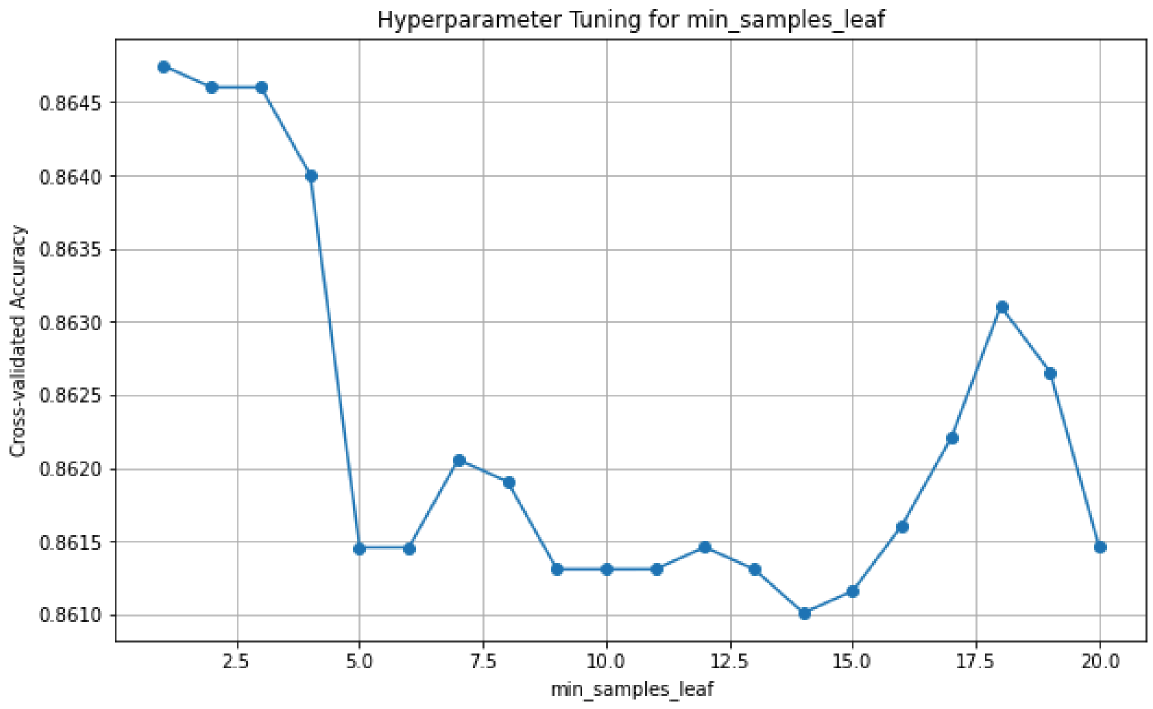


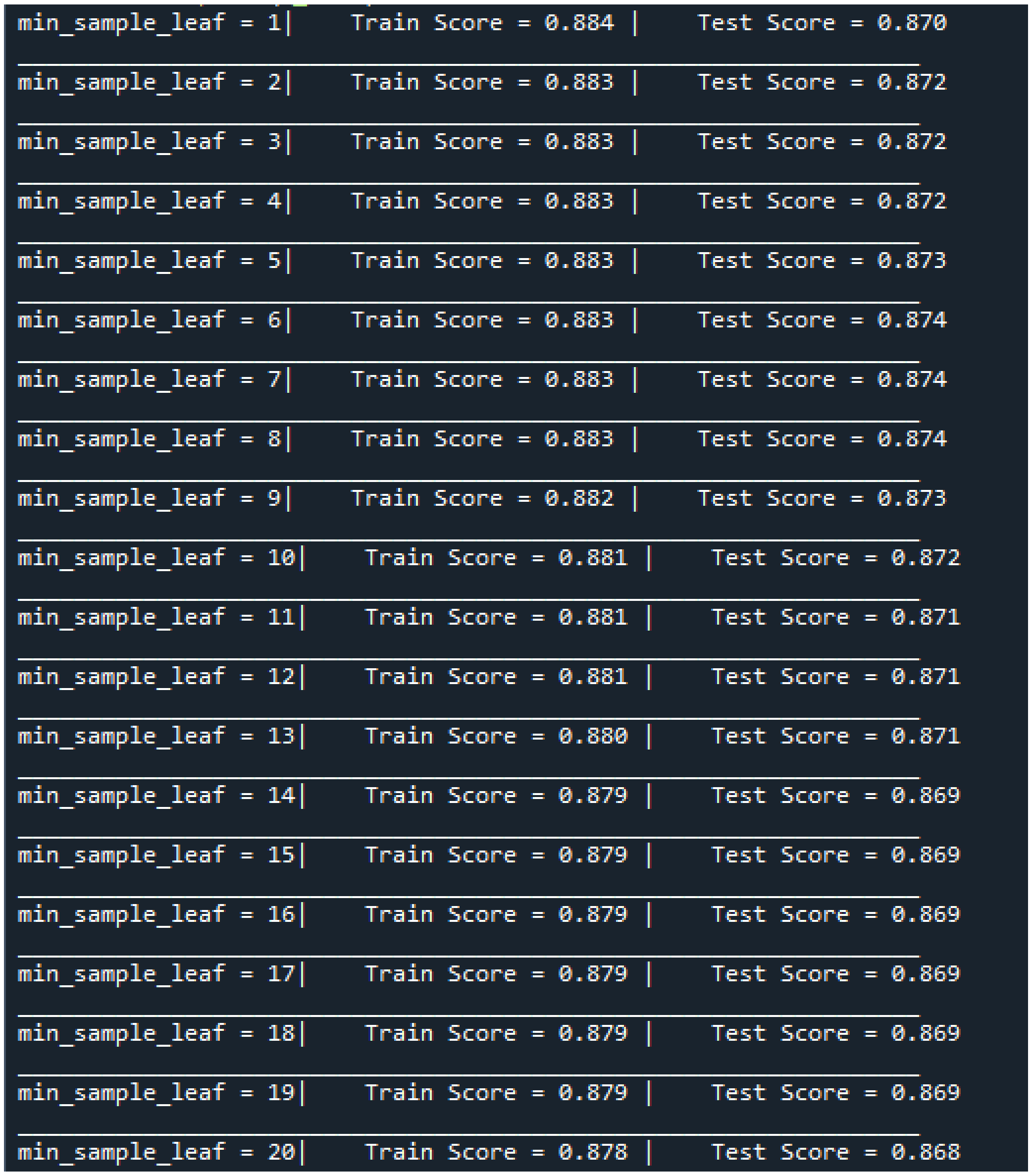
```
max_depth = 21|     Train Score = 1.000 |     Test score = 0.904
__________________________________________________________________
max_depth = 18|     Train Score = 0.998 |     Test score = 0.906
__________________________________________________________________
max_depth = 15|     Train Score = 0.989 |     Test score = 0.904
__________________________________________________________________
max_depth = 12|     Train Score = 0.971 |     Test score = 0.907
__________________________________________________________________
max_depth = 9|     Train Score = 0.933 |     Test score = 0.893
__________________________________________________________________
max_depth = 6|     Train Score = 0.884 |     Test score = 0.870
__________________________________________________________________
max_depth = 3|     Train Score = 0.836 |     Test score = 0.828
__________________________________________________________________
```

We can see from the above that for max_depth = 6, the difference between the train score and test score is the lowest.

Minimum Sample leaf:

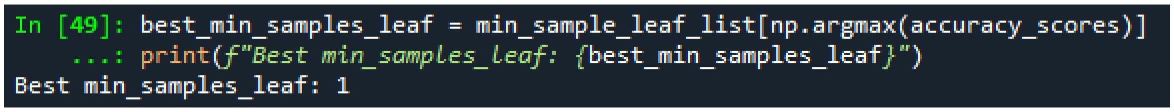
We loop through different values of minimum sample leaf from 1 to 20, and similarly check the train and test scores and also calculate the accuracy scores using cross validation. The train and test scores for different values of minimum sample leaf are shown in the table below and also a plot of different levels of accuracy achieved as we loop through different minimum sample leaf values -





```
min_sample_leaf = 1|    Train Score = 0.884 |    Test Score = 0.870
_________________________________________________________________
min_sample_leaf = 2|    Train Score = 0.883 |    Test Score = 0.872
_________________________________________________________________
min_sample_leaf = 3|    Train Score = 0.883 |    Test Score = 0.872
_________________________________________________________________
min_sample_leaf = 4|    Train Score = 0.883 |    Test Score = 0.872
_________________________________________________________________
min_sample_leaf = 5|    Train Score = 0.883 |    Test Score = 0.873
_________________________________________________________________
min_sample_leaf = 6|    Train Score = 0.883 |    Test Score = 0.874
_________________________________________________________________
min_sample_leaf = 7|    Train Score = 0.883 |    Test Score = 0.874
_________________________________________________________________
min_sample_leaf = 8|    Train Score = 0.883 |    Test Score = 0.874
_________________________________________________________________
min_sample_leaf = 9|    Train Score = 0.882 |    Test Score = 0.873
_________________________________________________________________
min_sample_leaf = 10|    Train Score = 0.881 |    Test Score = 0.872
_________________________________________________________________
min_sample_leaf = 11|    Train Score = 0.881 |    Test Score = 0.871
_________________________________________________________________
min_sample_leaf = 12|    Train Score = 0.881 |    Test Score = 0.871
_________________________________________________________________
min_sample_leaf = 13|    Train Score = 0.880 |    Test Score = 0.871
_________________________________________________________________
min_sample_leaf = 14|    Train Score = 0.879 |    Test Score = 0.869
_________________________________________________________________
min_sample_leaf = 15|    Train Score = 0.879 |    Test Score = 0.869
_________________________________________________________________
min_sample_leaf = 16|    Train Score = 0.879 |    Test Score = 0.869
_________________________________________________________________
min_sample_leaf = 17|    Train Score = 0.879 |    Test Score = 0.869
_________________________________________________________________
min_sample_leaf = 18|    Train Score = 0.879 |    Test Score = 0.869
_________________________________________________________________
min_sample_leaf = 19|    Train Score = 0.879 |    Test Score = 0.869
_________________________________________________________________
min_sample_leaf = 20|    Train Score = 0.878 |    Test Score = 0.868
```

We can almost no difference between the train and test scores for different values of minimum sample leaf, whereas the best accuracy is seen at min_sample_leaf = 1, wherein the accuracy drops to its lowest at around min_sample_leaf = 14 while again rising as we move from 14 to 18 as min_samples_leaf.



```
In [49]: best_min_samples_leaf = min_sample_leaf_list[np.argmax(accuracy_scores)]
    ...: print(f"Best min_samples_leaf: {best_min_samples_leaf}")
Best min_samples_leaf: 1
```

Overall, we see a high accuracy in classification using a Decision tree classifier than as seen for a KNN model or using Logistic regression.