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 is for the same dataset as seen earlier where we had done the classification using the KNN model, the Logistic regression model, Decision Tree classifier. And also using Bagging and Random Forest classifier. This time we will try different forms of Boosting - namely Gradient Boosting, XGBoost and LightGBM.

Gradient Boosting: It sequentially builds trees to correct errors made by previous trees.

XGBoost: This is called Extreme Gradient Boosting. It stands out for advanced handling of sparse data & its built-in validation. It is an efficient and scalable implementation of Gradient Boosting which also enhances model generalisation & prevents overfitting through the integration of Lasso and Ridge regularization techniques. It is thus expected to achieve high accuracy due to advanced regularization and tree boosting optimizations.

Light GBM: This is optimised for minimal memory use and swift computation. It employs a histogram-based splitting for quicker calculations. It is typically faster and potentially more accurate for large or complex datasets.

The dataset contains 9549 rows and 20 columns (original data without preprocessing), the no. of columns becomes 23 post preprocessing because of encoding, the 23 columns includes both numerical and categorical variables. The different variables in the dataset have been explained in the data dictionary, while the Target variable is a binary outcome variable, with '1' indicating 'Unhealthy' and '0' indicating 'Healthy'.

The boolean data types are converted to Int datatype before splitting the dataset into train and test. There are no missing values in any of the columns.

```
In [4]: df_hc.dtypes
Out[4]:
Age                       float64
BMI                       float64
Blood_Pressure            float64
Cholesterol               float64
Glucose_Level             float64
Heart_Rate                float64
Sleep_Hours               float64
Exercise_Hours            float64
Water_Intake              float64
Stress_Level              float64
Target                      int64
Smoking                     int64
Alcohol                     int64
Diet                        int64
MentalHealth                int64
PhysicalActivity            int64
MedicalHistory              int64
Allergies                   int64
Diet_Type__Vegan             bool
Diet_Type__Vegetarian        bool
Blood_Group_AB               bool
Blood_Group_B                bool
Blood_Group_O                bool
dtype: object
```

```
In [8]: print(missing_values)
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
```

After splitting the data into 80% train and 20% test data, and using scaling, I have initialised the GradientBoosting, XGBoost and LightGBM classifiers and used GridSearchCV to find the optimal parameters for n_estimators, learning_rate, and tree-specific parameters like max_depth, min_samples_leaf and min_samples_split.

Each model was tuned with 5-fold cross-validation for evaluation.

The performance summary after hyperparameter tuning is shown below:

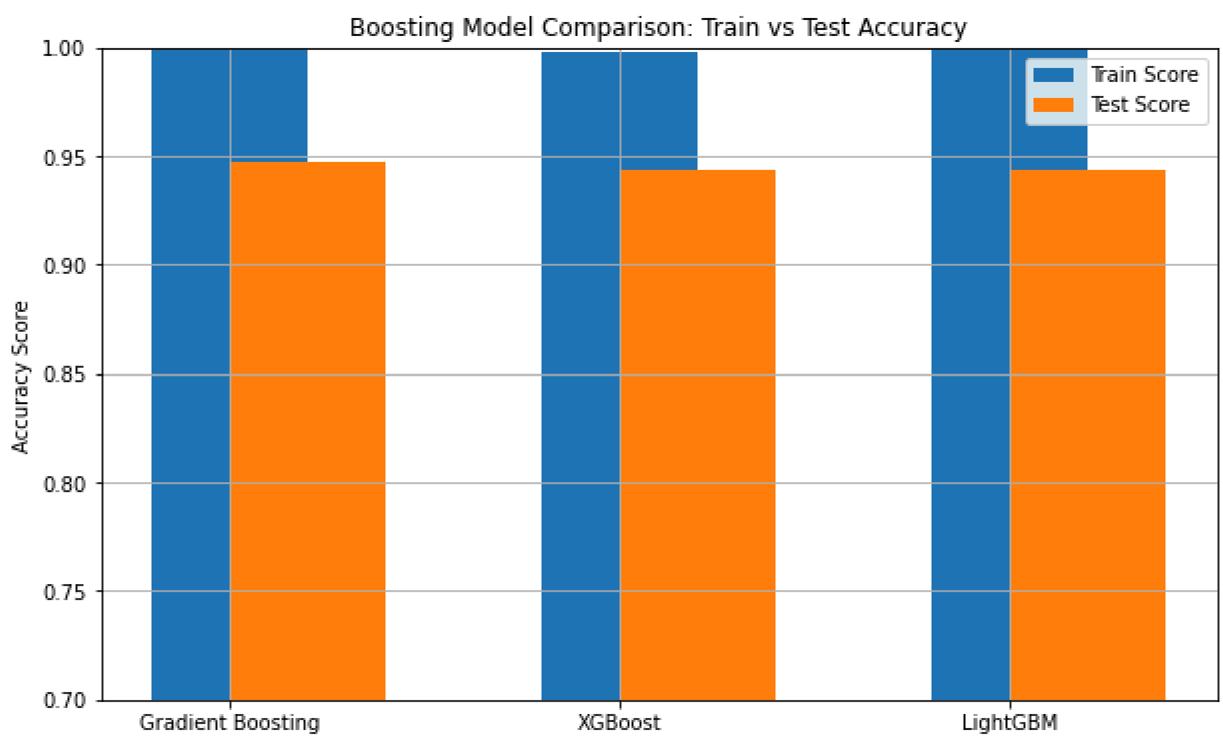
Model	Best params	Train Score	Test score
Gradient Boosting	{'n_estimators': 150, 'Learning_rate':0.2, 'min_samples_leaf': 2, 'min_samples_split': 5, 'max_depth': 7}	1	0.9471
XGBoost	{'n_estimators': 100, 'Learning_rate':0.2, 'min_child_weight': 1, 'subsample': 0.8, 'max_depth': 7}	0.9979	0.9440
Light GBM	{'n_estimators': 150, 'Learning_rate':0.2, 'num_leaves': 63, 'subsample': 0.8, 'max_depth': 7}	1	0.9440

Gradient Boosting achieves the highest test accuracy among the three models. The test accuracy of 94.71% suggests a strong performance with good generalization despite the high training accuracy.

XGBoost performs well with 94.4% test accuracy. It slightly underperforms compared to Gradient Boosting, though still very strong. The smaller number of estimators (n_estimators=100) could be a reason for this slight drop in test accuracy compared to Gradient Boosting, which used 150 trees.

LightGBM's performance is very similar to XGBoost, with the same test accuracy of 94.4%. While LightGBM's tree structure (with num_leaves=63) and larger number of estimators are tuned for faster computation, the test accuracy is still slightly behind Gradient Boosting.

I have compared the train and test accuracy after hyperparameter tuning across all three models in a bar chart as shown below -



Conclusion

Gradient Boosting model yields the highest test accuracy (94.71%) and proves to be the best for this dataset. Both XGBoost and LightGBM models have very similar performance, with 94.4% test accuracy. They are slightly lower in performance compared to Gradient Boosting but still provide very good results. XGBoost offers better performance when tuned with fewer trees (100), thus suggesting perhaps that the extra 50 estimators in Gradient Boosting might not be improving performance significantly. LightGBM is also highly competitive, although it doesn't surpass Gradient Boosting in terms of test accuracy.