A black background with white text

Description automatically generated with low confidence

MSc Data Science Project

7PAM2002-0509-2024

Department of Physics, Astronomy and Mathematics

**Data Science FINAL PROJECT REPORT**

Project Title:

Soft and Hard Hybrid Modelling for Insurance Cost Prediction

Student Name and SRN:

Ramya Palla and 22063652

Supervisor: Dr. Hyungrok Kim

Date Submitted:

Word Count: 5461

GitHub Link: <https://github.com/pallaramya/Soft-and-Hard-Hybrid-Modelling-for-Insurance-Cost-Prediction>

# Declaration Statement

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in Data Science at the University of Hertfordshire.

I have read the detailed guidance to students on academic integrity, misconduct and plagiarism information at [Assessment Offences and Academic Misconduct](https://ask.herts.ac.uk/assessment-offences-and-academic-misconduct) and understand the University process of dealing with suspected cases of academic misconduct and the possible penalties, which could include failing the project or course.

I certify that the work submitted is my own and that any material derived or quoted from published or unpublished work of other persons has been duly acknowledged. (Ref. UPR AS/C/6.1, section 7 and UPR AS/C/5, section 3.6)

I did not use human participants in my MSc Project.

I hereby give permission for the report to be made available on module websites provided the source is acknowledged.

Student Name printed: Ramya Palla

Student Name signature: Ramya Palla

Student SRN number:22063652

UNIVERSITY OF HERTFORDSHIRE

SCHOOL OF PHYSICS, ENGINEERING AND COMPUTER SCIENCE

Acknowledgement

As I come close to finishing my post-graduate studies, I would like to emphasize that it has been a wonderful learning experience, and I want to express my gratitude to all the people who have supported me along the way.

I'd like to start by expressing my gratitude to Almighty God for never ceasing to inspire me with His endless blessings and for giving me the confidence and valour to move forward with assurance and self-belief.

I would like to convey my appreciation and gratitude to Dr. Hyungrok Kim, who served as my supervisor, for his constant advice and assistance in this project. I am appreciative of his constant support and his patience towards my inquisitiveness.

I would also like to express my gratitude towards all my professors at the University of Hertfordshire who helped me gain knowledge and understanding of the subjects and helped me throughout my course.

I would also like to thank my parents, and my friends for their unwavering encouragement and support, without which this would not have been possible.

# Abstract

Insurance cost prediction plays a vital role in healthcare and finance by enabling accurate estimation of medical expenses for fair premium setting and risk management. However, challenges such as nonlinear relationships, diverse data types, and outliers complicate model development and generalization. This research addresses these issues by comparing hybrid ensemble machine learning models—soft voting (probabilistic averaging) and hard voting (majority voting)—using a structured insurance dataset with demographic, lifestyle, and regional features. After preprocessing and exploratory analysis, regression models including Gradient Boosting, Decision Tree, and Random Forest were trained individually and in hybrid combinations. Gradient Boosting achieved the best standalone performance with an R² of 88.8% on validation and 82.9% on testing. Classification models also showed strong results, with hybrid ensembles reaching up to 92% accuracy in validation and 88% in testing. Soft voting ensembles slightly outperformed hard voting in both regression and classification tasks, demonstrating superior generalization. This study highlights the effectiveness of hybrid ensemble methods in improving insurance cost prediction under complex data conditions.

Table of Contents

[Declaration Statement ii](#_Toc206712620)

[Abstract iv](#_Toc206712621)

[1 Introduction 1](#_Toc206712622)

[1.1 Project Overview 1](#_Toc206712623)

[1.2 Research Context and Motivation 1](#_Toc206712624)

[1.3 Aim 1](#_Toc206712625)

[1.4 Research Question 2](#_Toc206712626)

[1.5 Research Objectives 2](#_Toc206712627)

[1.6 Significance of the Study 2](#_Toc206712628)

[1.7 Feasibility 2](#_Toc206712629)

[1.8 Report Outline 3](#_Toc206712630)

[2 Literature Comparison 4](#_Toc206712631)

[3 Methodology 6](#_Toc206712632)

[3.1 Research Design 6](#_Toc206712633)

[3.2 Dataset Description 7](#_Toc206712634)

[3.3 Data Preprocessing and Feature Engineering 8](#_Toc206712635)

[3.4 Data Splitting 8](#_Toc206712636)

[3.5 Model Selection and Justification 8](#_Toc206712637)

[3.6 Ensemble Learning Approach 9](#_Toc206712638)

[3.7 Hyperparameter Tuning and Validation 9](#_Toc206712639)

[3.8 Performance Evaluation 9](#_Toc206712640)

[3.9 Ethical Considerations 9](#_Toc206712641)

[3.10 Summary 10](#_Toc206712642)

[4 Experiments 11](#_Toc206712643)

[4.1 Data Preparation 11](#_Toc206712644)

[4.2 Model Implementation and Evaluation 17](#_Toc206712645)

[5 Results Analysis 19](#_Toc206712646)

[5.1.1 Regression Models 19](#_Toc206712647)

[5.1.2 Classification Models 20](#_Toc206712648)

[5.2 Key Findings of the Research 24](#_Toc206712649)

[5.3 Answer for Research Question 24](#_Toc206712650)

[5.4 Comparative Analysis of Current Research with Previous Studies 25](#_Toc206712651)

[6 Conclusion 26](#_Toc206712652)

[6.1 Future Work 26](#_Toc206712653)

[References 27](#_Toc206712654)

[Appendix 28](#_Toc206712655)

[Code 28](#_Toc206712656)

[Learning Curves 40](#_Toc206712657)

[Graphs 42](#_Toc206712658)

[Confusion Matrices with chosen Parameters and Evaluation Metrics 43](#_Toc206712659)

[ROC Curves 47](#_Toc206712660)

[Task List and Project Timeline 51](#_Toc206712661)

[Gantt Chart 51](#_Toc206712662)

# Introduction

## Project Overview

In recent years, insurance companies have increasingly adopted data-driven approaches to price insurance policies more accurately and efficiently. Predicting insurance charges is a critical task in the domain of health insurance analytics (Albalawi et al, 2023). The growing availability of structured customer data, including age, BMI, smoking status, and region, has made it feasible to improve ML models that can detect insurance costs with substantial accuracy. Accurate cost prediction benefits both insurers and customers: insurers can better assess risk and set fair premiums, while customers receive personalized and equitable policy pricing (Smith and Owen, 2024).

The purpose of this project is to explore and compare two types of hybrid ML models soft and hard hybrid approaches for the task of insurance cost prediction. Hybrid models leverage the strength of multiple base learners to improve generalization performance. The project aims to evaluate which type of hybrid combination strategy soft (probability-based) or hard (voting-based)offers better predictive accuracy, robustness, and practical applicability in insurance cost forecasting.

## Research Context and Motivation

The application of ensemble learning in regression and classification problems has gained popularity due to its ability to reduce variance and bias, thus improving model performance. Specifically, ensemble methods which average predictions made by a variety of base learners can frequently produce better results than any single model, using complementary sources of strength. Nevertheless, compared to ensemble methods like bagging and boosting, which are quite popular, comparing soft and hard hybrid strategies specifically on insurance data that are moderate in their complexity and have non-linear relationships have received less attention.

As prediction errors in practical insurance scenarios might result in financial penalty in case of underestimation and loss of prospective clients in case of overestimation, this study aims at exploring which of the hybrid strategies offers superior generalization and reliability.

## Aim

The aim of this project is to develop and compare soft and hard hybrid machine learning models to predict insurance costs, in order to identify which hybrid approach offers better generalization and predictive performance on structured insurance data.

## Research Question

1. Which hybrid approach—soft combination (Probability based) or hard combination  voting based)—yields better generalization and performance in insurance prediction?

## Research Objectives

* To conduct exploratory data analysis (EDA) on the insurance dataset to uncover patterns, relationships, and key features influencing insurance charges.
* To build and evaluate individual base models like random forests, decision trees, and gradient Boosting for both regression and classification tasks for predicting insurance costs, establishing benchmarks for hybrid model performance.
* To develop soft hybrid models by integrating the probabilistic outputs of base learners using techniques such as soft voting for classification and averaging for regression.
* To construct hard hybrid models by aggregating predictions from base learners using voting-based methods such as hard voting for classification and averaging for regression scheme.
* To systematically compare the generalization and predictive performance of soft versus hard hybrid models using evaluation metrics such as R², MAE, RMSE for regression and accuracy, precision, recall, F1-score for classification.
* To evaluate the robustness, scalability, and practical applicability of each hybridization approach in real-world insurance analytics settings.

## Significance of the Study

Proper prediction of insurance cost poses great importance to both the insurers and the customers. Superior predictive models have the potential to produce equitable pricing, lesser losses, and enhanced allocation of resources in the insurance firms. Moreover, the investigation of hybrid model methods leads to the improved machine learning applications in financial services, which shows how ensembles learning can be practically used to gain better results and resilience. The results of this study might be used in the further research and implementation of hybrid models in practical insurance analytics.

## Feasibility

The present research is possible due to the presence of a well-organized public insurance dataset with the appropriate features including age, BMI, smoking status, and region. The data is readily available and stored in a credible GitHub repository, no additional data are to be gathered, and there are no personally identifying details, which removes any ethical limitations. Furthermore, the project utilizes popular open-source libraries Scikit-learn by ML algorithms, Pandas and NumPy by data manipulation, and Matplotlib and Seaborn by visualization. Such libraries together with available computational resources will guarantee the trouble-free implementation and evaluation of base and hybrid ensemble models within the project timeframe.

## Report Outline

The structure of the report is the following: Chapter 2 will include an extensive literature review and background, where related works will be analyzed, and this study will be placed in the context of another research. Chapter 3 describes the research methodology, including dataset description, justification of chosen methods, algorithms and techniques used, and evaluation metrics. Chapter 4 presents the experimental results obtained from the models. Chapter 5 provides an in-depth analysis and discussion of the findings, linking them back to the research objectives. Finally, Chapter 6 concludes the report by summarizing key results, drawing conclusions, discussing practical applications, and suggesting directions for future work.

# Literature Comparison

Recent studies are increasing the role of ML and hybrid models to predict insurance costs with the objective of having an accurate and at the same time scalable solution for practical applications. In this section, those efforts are compared to the  project to build and compare soft (probability based) and hard (voting based) modelling approaches applied to structured insurance data to maximize predictive performance with respect to the generalization.

Panda et al, (2022) in their work, derive the health insurance premiums by using Regression models like Polynomial Regression, Lasso Regression, Ridge Regression, Multiple Linear Regression and Simple Linear Regression. For example, their dataset included the customer attributes and by utilizing Polynomial Regression, the optimal results were achieved with RMSE of 5100.53, R² of 0.80 and accuracy of 80.97%. They brought out scalability issues with large datasets and offered cloud based solutions. Contrary to their concentration on separate, single models, this project combines base learners into soft and hard hybrid frameworks for the aim of generalization.

In Patra et al, (2024) used KNN, Lasso, Ridge and XGBoost for prediction of medical costs from dataset which had age, BMI and geographical features among others. It is seen that XGBoost achieved high (R²: 86.81, RMSE: 4450.4) while KNN performed poorly (R²: 55.21, RMSE: 4431.1). On the basis of small sample sizes, they proposed to add lifestyle factors and metaheuristic algorithms. Their use of advanced ML with this project fits, but concentrates their work on comparing soft and hard hybrid approaches for robustness and applicability. AbdElminaam et al, (2024) evaluate five ML algorithms based on MAPE and R² to test five ML algorithms on four datasets. MAPE Of 3.5% with 10-fold cross validation was achieved by Decision Trees, whereas Neural Networks struggled to (R²: -15.206). In data accuracy and ethical deployment, they pointed problems. While this project carries the same importance of robust evaluation metrics (MAE, RMSE, R²) to eliminate any poor fitting models, it puts a larger focus on hybrid models for better generalization.

Hassan et al, (2021) have evaluated ML Algorithms including XGBoost, Stochastic Gradient Boosting and RF, by using a Kaggle dataset. SGB performed the highest with 86% accuracy and RMSE of 0.340 among these considered algorithms. Thus, they suggested the use of the metaheuristic algorithms for future improvements. This work continues their work by combining the base learner into soft (weighted averaging) and hard (majority voting) hybrid models and evaluate the effectiveness of the resulting models. Albalawi et al, (2023) used ML and Gradient Boosted Tree on PySpark on a Kaggle data set where the later had an R² of 0.9067. Linear models were faced with the challenges of nonlinear data. Similarly, this project uses ensemble techniques but hybridizes base learners and tests scalability of these hybridized learners to an insurance environment.

Wilson et al, (2024) analyzed the comparison of GLM, GBM, ANN and a hybrid GLM–ANN model for motor insurance loss cost detection. The best performed model is the hybrid one (MAE: 2011.907). In this project, their hybrid approach is extended to compare soft and hard combination and its performance on structured insurance data. One type of hybrid models called a Linear Regression-Gaussian DBN hybrid model (LR – GDBN) was developed by Bhargavi and Arumugam, (2024) with low prediction error (0.4926). Heterogeneous datasets were not scalable. This project also looks at hybrid models and evaluates their probabilistic and voting based approaches for more utilization. For instance, Shyamala Devi et al, (2021) used Polynomial and Random Forest regression and reported R² scores of 88% and 86% respectively. ANOVA found the feature ‘region’ to have insignificant contribution to the dependent variable. In this project, things run similarly and here the focus is on hybrid ensembles with an emphasis on their robustness.

**Identification of gaps**

A variety of improvements has been made in insurance cost prediction, but there is still a wide range of gaps. The majority of the studies such as Panda et al, (2022) and Patra et al, (2024), examine either individual ML models or hybrid approaches which combine statistical and neural methods. However, these works do not systematically compare soft hybrid models (e.g., probabilistic outputs weighted averaging) against hard hybrid models (e.g., majority voting). A second limitation is that this limits insights into what hybridization strategy is capable of providing better generalization and performance for structured insurance data.

Moreover, studies such as Shyamala Devi et al, (2021), address feature selection, but studying the effect of ANOVA or RFE on hybrid model performance is not explored. As Patra et al, (2024) mentioned, although lifestyle or socio-economics predictors can improve model robustness such integration is often ignored. A major limitation for Pandemic et al, (2022) and Bhargavi and Arumugam (2024) is scalability and applicable in the real world for hybrid models that are deployed in insurance settings. This project filled in these gaps by developing and comparing soft and hard hybrid models with respect to metrics like MAE, RMSE and R². In addition, it performs exploratory data analysis to uncover important features to the modelling process and evaluates the influence of feature selection on modelling effectiveness to help better understand strategies for hybridization of insurance cost prediction models.

# Methodology

This chapter gives a description of the methodology used in predicting insurance cost based on the machine learning techniques. A publicly available, semi-conveyed dataset on insurance is utilized in the study providing characteristics like age, BMI, amount of children, sex, smoking, and region. Regression and classification solutions are both examined to offer a detailed view of cost estimation and the ensemble learning algorithms can be used to increase precision and robustness. The preprocessing of the data, feature engineering, picking of the models, tuning of the hyperparameter, and the performance assessment are included in the methodology and able to deliver reliable and interpretable results.

## Research Design

The workflow of the project starts with the preprocessing of the data, specifically, the removal of different duplicates, exploration of data using histograms and pie charts, and one-hot encoding of categorical variables. Data is divided into 602020 train-validation-test. This is followed by individual algorithm implementation, training Gradient Boosting, Decision Tree and Random Forest following GridSearchCV hyperparameter optimization.

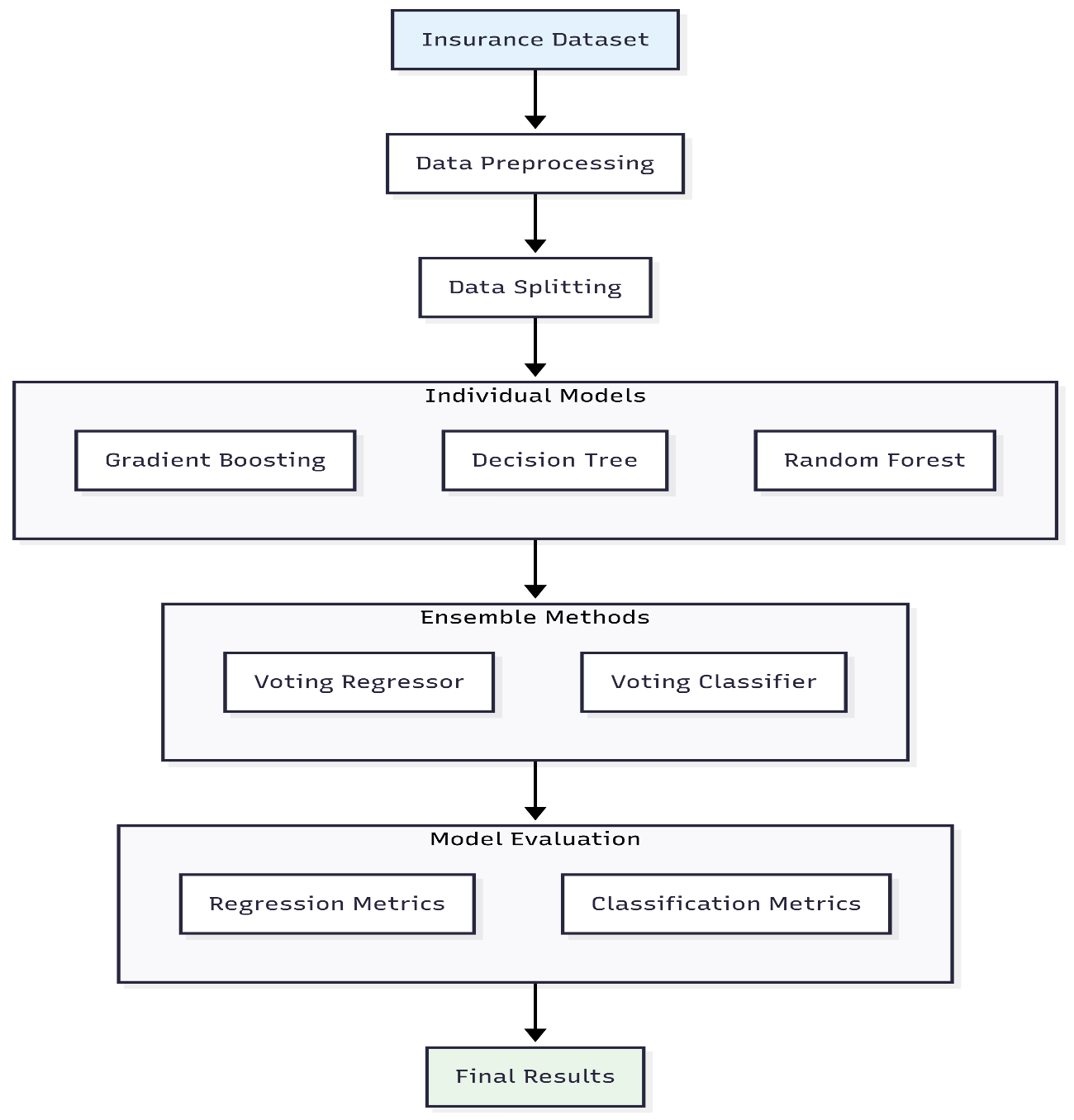


Figure 1 Project Design

Ensemble implementation ensemble implementation is the combination of algorithms with Voting Regressor employed with continuous predictions and Hard/Soft Voting Classifiers used in categorical predictions. Evaluation phase computes overall measures (R², MAE, MSE, RMSE in case of regression; precision, recall, F1-score in case of classification) comparing individual and ensemble performance between the two formulations of the problems.

## Dataset Description

The dataset used in this research is freely available in the Machine Learning with R dataset repository (Lantz, 2013) on GitHub. It has 7 variables that include age, sex, BMI, number of children, smoking, region, and charges (target variable) with a total of1,338 samples. The specified dataset is very relevant to the regression and the cost of insurance prediction because it includes not only the demographic characteristics but also the features concerning the health conditions as the factors affecting the insurance costs that are known. Categorical variables (Sex, Smoker and Region) were coded to the corresponding numerical types before the modeling phase. Exploratory investigation found significant correlation amongst the features like BMI and the smoking status with the insurance charges. Data was well partitioned into training data (80%) and testing data (20%) to verify whether the performance of the model would be accurate on new data. The small size and the abundance of features qualify the dataset as an excellent proposal to train hybrid machine learning models in a controlled, but realistic environment.

## Data Preprocessing and Feature Engineering

An essential first step in this research is to provide data quality assurance. There were no missing values found and duplicates were eliminated to uphold the integrity of data. Key numerical variables like age, bmi, and children where analyzed using exploratory data analysis and histogram to know their distribution. To determine the proportions of such categorical variables as sex, smoker, and region, pie charts were used to visualize them. The categorical variables used in machine learning required one-hot encoding transformation (so they could not form artificial ordinalities) by creating dummy variables such as sex male, smoker yes, and region northeast.

To classify the data in the classification task, the continuous charges variable was grouped into three groups like Low, Medium, and High based on quartiles (25 th and 75 th percentiles). That this new categorical target, charges category, reduces the issue to a multi-class classification, having models predicting which category of the risk applies, rather than a specific charge.

## Data Splitting

The data was divided into the training (60%), validation (20%) and testing (20%) sets through fixed random states to guarantee reproducibility. The training set was utilized to train the models whereas the validation set which did not help in training but helped to perform a hyperparameter tuning and model selection that was advantageous because it led to unbiased evaluation. To approximate real world predictive performance, final model assessment was carried out in the test set. This splitting approach avoids data leakage and overfitting which is important to generating generalizable machine learning models.

## Model Selection and Justification

In this research, three important machine learning models are used, namely Decision Tree, Random Forest, and Gradient Boosting, which will be used both on regression and classification. The interpretability of Decision Trees was selected because it would provide simple, easy-to-understand decision rules such as, “smoker yes and bmi > 30, then charges would be high,” which would help businesses be insightful and allow regulatory clarity. The use of Random Forest was chosen due to its high stability, as a bootstrap aggregation is used to lower the overfitting rate, and random feature selection is capable of dealing with outliers and heterogeneity of the data like disparity in age and the number of children (Gorgipour, 2022). Gradient Boosting was chosen on the basis of sequential learning, which is error-corrective, and captures complex non-linear relationships among features, such as age, smoker or non-smoker, and charges. Collectively, the above models compete because they are traded off to balance clarity, stability and accuracy. A combination of them in ensemble structures can make the bias-variance tradeoff superior and achieve an overall better solution in performance.

## Ensemble Learning Approach

In the regression models of continuous charges, the model performance was evaluated based on the R²Score, the explained variance, together with the Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) of the prediction errors which penalized large estimates. Under classification, the quality of prediction by charges category (Low, Medium, High) was evaluated by accuracy, precision, recall and F1-score in classes. The correct and incorrect predictions were graphically represented in the confusion matrices that reflected frequent errors. All these measures offer a comprehensive insight into how a model performs both in regression and classification.

## Hyperparameter Tuning and Validation

The Random Forest, Gradient Boosting, and Decision Tree models had their key hyperparameters optimized using GridSearchCV with 2-fold cross-validation using hyperparameters such as: n\_estimators, max\_depth and criterion. In this tuning model, the complexity did not over fit by picking the best parameters depending on the performance of the validation set, and this was determined as R² in regression and F 1 score in classification.

## Performance Evaluation

The performance of the regression models was measured in terms of variance explained and errors made in predictions using R², MAE, and RMSE. Classification models have been benchmarked using accuracy, precision, recall, F1-score, and confusion matrices that gave a clear picture of the performance of the models depending on Low, Medium, and High categories of insurance charges.

## Ethical Considerations

In predictive modeling research, data privacy has been attended to by employing publicly available, anonymized datasets, and ethics have been observed. Since some features like sex and region can create a bias, the use of the ensemble methods can suppress personal model bias. It is required to use these models to assist human judgment in the underwriting process of an insurance company instead of defining this process as either automated or transparent, avoiding unfairness.

## Summary

Overall, the approach combines the comprehensive data selection and processing with the built in, advanced levels of supervised learning such as Decision Trees, Random Forests and Gradient Boosting. The research classifies the recurring insurance payments into categories thereby conducting category usage of classification in addition to regression whereby ensemble methods such as voting are adopted to improve predictions. Strict validation and evaluation measurements make the model reliable, and ethical aspects guarantee acceptable use of prediction analytics in insurance. Such an organized practice builds good background towards proper and interpretable cost prediction in insurance.

# Experiments

This chapter outlines the experimental workflow for predicting medical insurance costs using both regression and classification approaches. It covers data preparation, exploratory data analysis with visualizations, model implementation using Gradient Boosting, Decision Tree, Random Forest, and ensemble methods, along with performance evaluation using appropriate metrics.

## Data Preparation

This experiment aims to predict medical insurance costs using the insurance.csv dataset, which contains 1,338 records and 7 attributes. The features include demographic variables (age, sex, region), health indicators (bmi, children, smoker), and the target variable charges. No missing values were present, but one duplicate entry was removed.

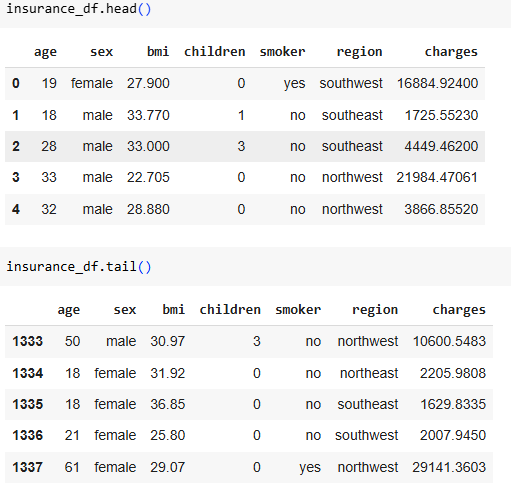


Figure 2 Dataset Overview with head and tail of the Dataset

Exploratory Data Analysis (EDA) revealed that age shows a relatively uniform distribution across the age range with considerable variation, clearly deviating from normality. BMI displays a more bell-shaped pattern that approximates normal distribution with some deviations. Charges are heavily right-skewed due to the presence of high-cost outliers, with most individuals clustering at lower cost values and a long tail extending toward higher charges. Most individuals have zero to two children, non-smokers dominate the dataset, and regional distribution is balanced across the four regions.

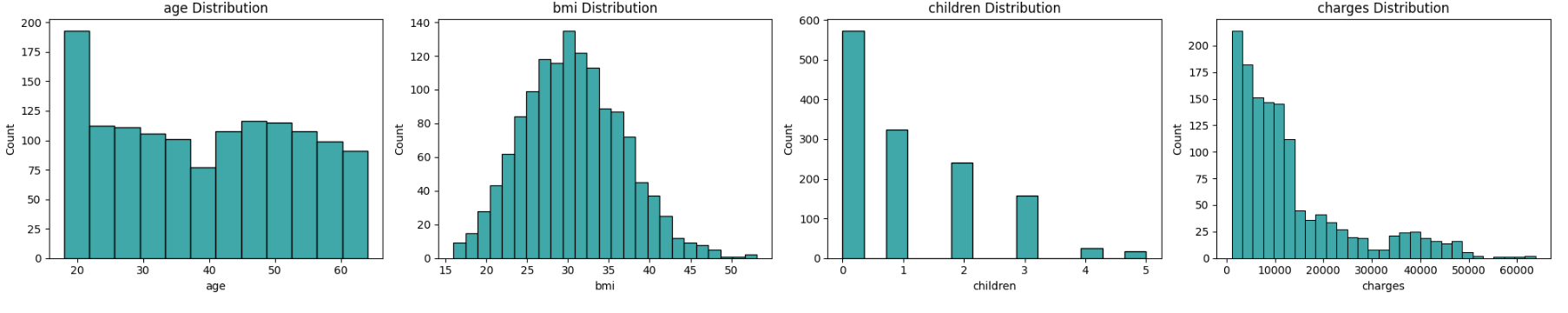


Figure 3 Visualization of Numeric Columns

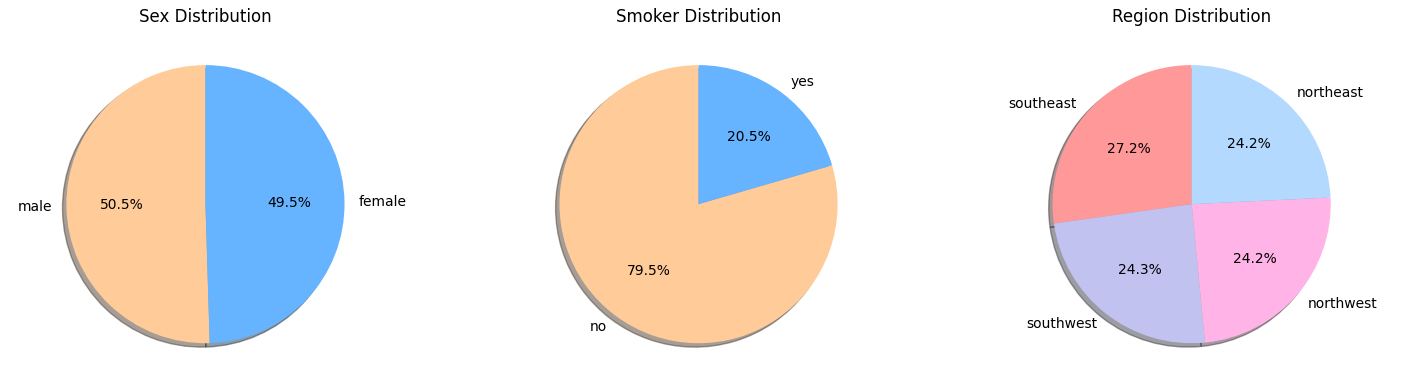


Figure 4 Visualization of Object Columns

To visually examine these patterns, histograms were plotted for the target variable (charges) using a kernel density estimate with a dark cyan hue, and for BMI using a tan color. A pie chart of the ‘region’ feature displayed a balanced distribution across four categories. Another pie chart of the ‘smoker’ feature revealed the dominance of non-smokers.

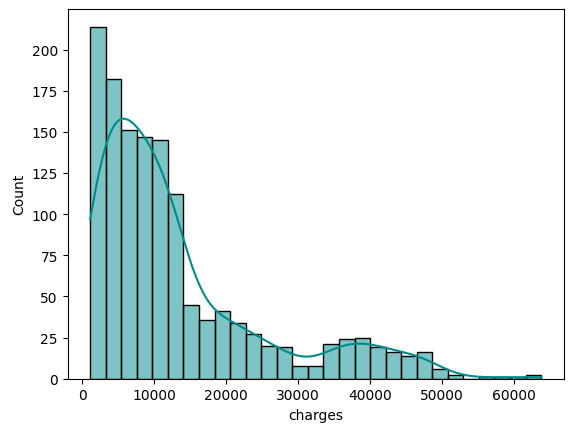


Figure 5 Histogram – Target

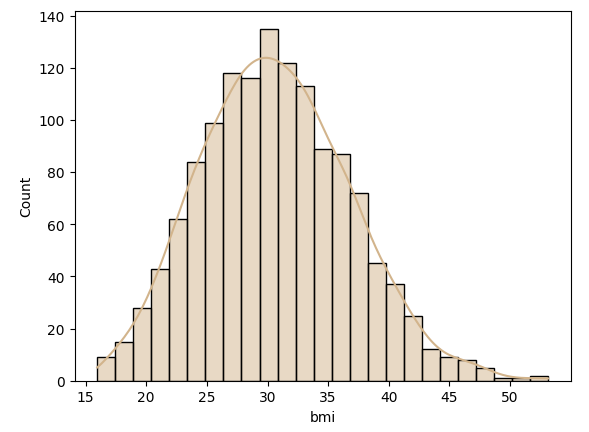


Figure 6 Histogram – BMI

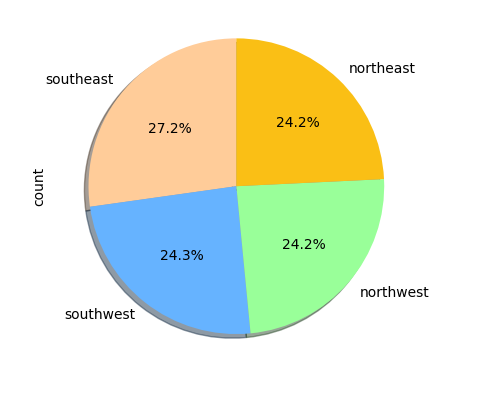


Figure 7 Pie Chart - Regions

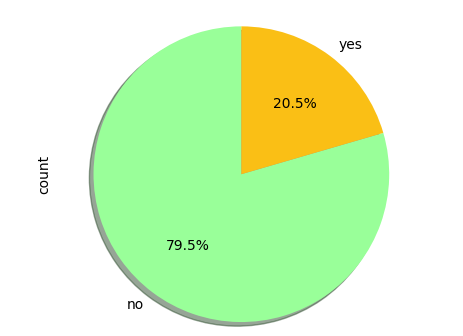


Figure 8 Pie Chart – Smoker

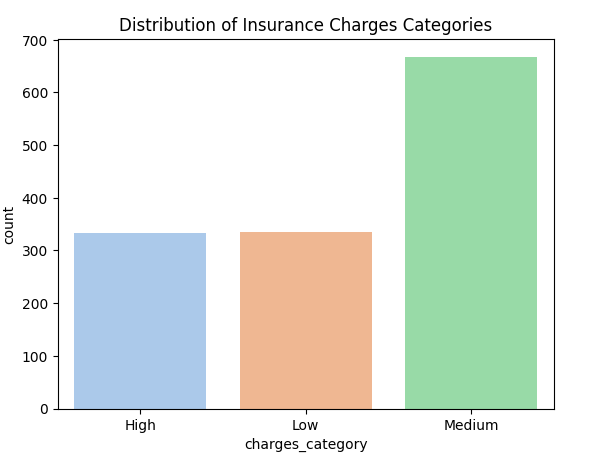


Figure 9 Distribution of Insurance Charges Categories

Categorical features (sex, smoker, region) were converted into binary indicators via one-hot encoding, producing a final dataset of 1,337 rows and 12 columns. The predictor columns in the training set are : age, bmi, children, sex female, sex male, smoker no, smoker yes, region northeast, region northwest, region southeast, region southwest. The target column is charges.

Since the target variable charges is continuous, it was quantized into three discrete categories (low, medium, high). This transformation enabled the application of classification models, including soft-voting and hard-voting ensembles, which require categorical labels rather than continuous values.

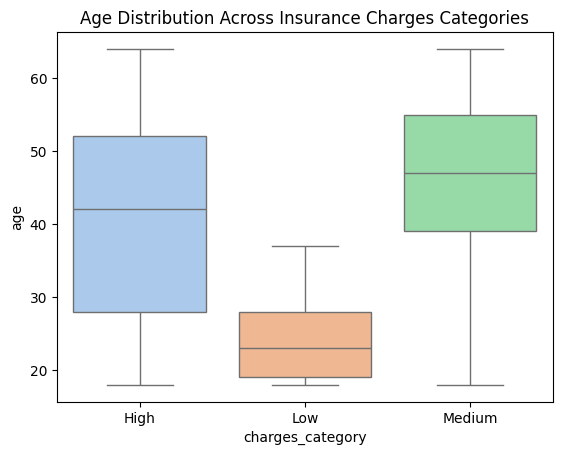


Figure Box Plot – Age Distribution Across Charges Categories

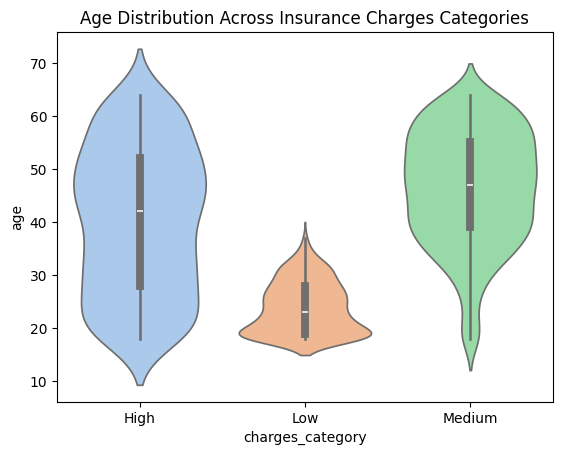


Figure Violin Plot – Age Distribution Across Charges Categories

The boxplot and violin plot shows that a positive relationship between age and insurance charges. The High charges category shows older individuals with wider age distribution and higher median age, while the Low charges category is dominated by younger individuals with narrower age ranges. The Medium category demonstrates balanced age distribution between the extremes. The violin plot confirms that younger individuals concentrate in Low charges, while older individuals predominantly fall into High charges categories.

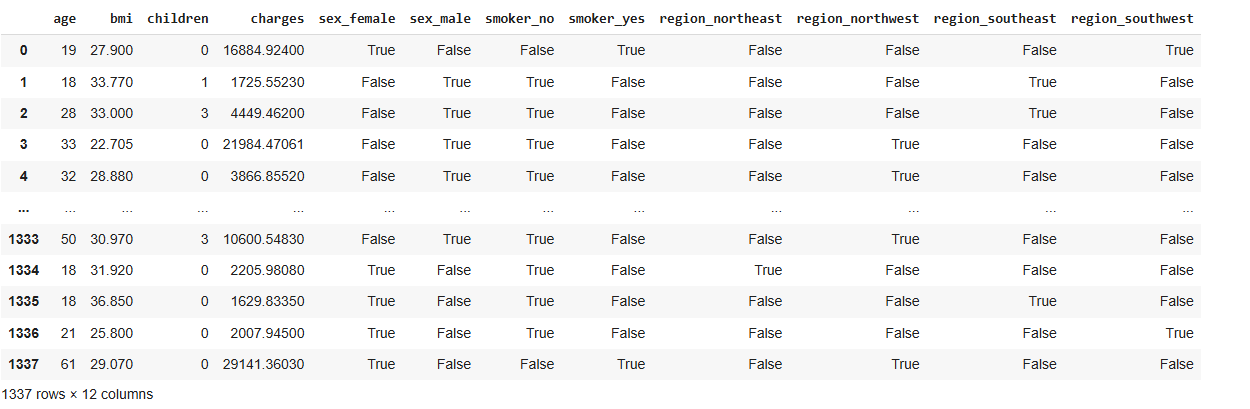
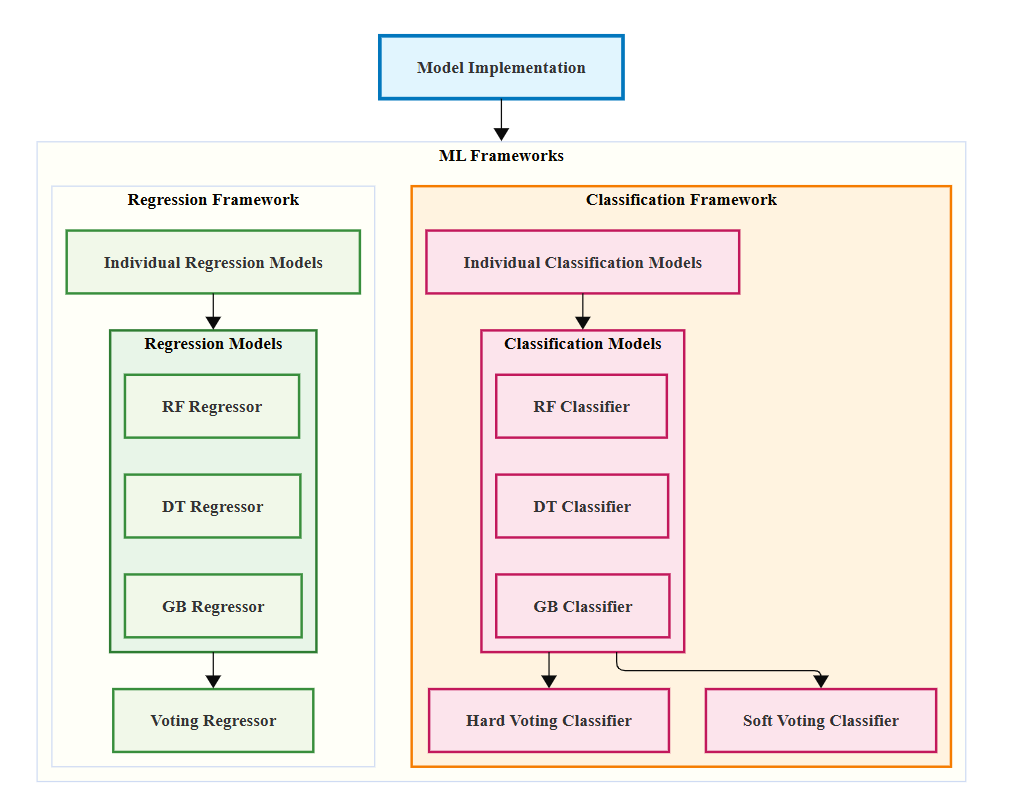


Figure 12 Dataset - After Pre-processing

## Model Implementation and Evaluation

The model implementation process involved developing both regression and classification frameworks using Gradient Boosting (GB), Decision Tree (DT), Random Forest (RF), and ensemble voting techniques. For regression tasks, individual models (GB, DT, RF) were trained to predict continuous target values, and their predictions were aggregated in a Voting Regressor to improve stability and generalization. For classification, the same base models were applied to predict categorical outcomes, complemented by Hard Voting and Soft Voting classifiers to leverage the strengths of multiple learners.



All models were implemented using Python’s Scikit-learn library, with hyperparameters tuned through grid search on the validation set to optimize performance. The dataset, containing **11 columns,** was split into 60% training (802 samples), 20% validation (267 samples), and 20% testing (268 samples) using a fixed random state for reproducibility. Models were evaluated using appropriate metrics—R², MAE, MSE, and RMSE for regressors, and accuracy, precision, recall, and F1-score for classifiers—to ensure a comprehensive assessment of predictive accuracy and generalization capability.

# ****Results Analysis****

### ****Regression Models****

The Gradient Boosting (GB) regressor achieved the **highest validation R² score** (88.80%) and maintained strong generalization on the test set (82.89%). It also recorded the **lowest RMSE** in both validation (4077.00) and testing (4812.98), indicating more accurate cost predictions with fewer large errors.

Table 1 Result Analysis - Regressor Models

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Regressor Algorithms** | | **R2 Score** | **MAE** | **MSE** | **RMSE** |
| **Validation** | **GB** | **88.8002** | **1595.7870** | **16621968.9360** | **4077.0048** |
| **DT** | 87.0180 | 2731.3241 | 19266975.4031 | 4389.4162 |
| **RF** | 87.5765 | 2641.3234 | 18438116.2630 | 4293.9627 |
| **Voting** | 88.2665 | 2293.5412 | 17414033.8278 | 4173.0126 |
| **Testing** | **GB** | 82.8951 | 1948.6318 | 23164742.5428 | 4812.9764 |
| **DT** | 81.2652 | 3091.2508 | 25371979.4335 | 5037.0606 |
| **RF** | 81.9876 | 3019.0790 | 24393666.1170 | 4938.9944 |
| **Voting** | 82.4662 | 2667.6787 | 23745538.5642 | 4872.9394 |



Figure Chart of Results Analysis - Regression Algorithms

The Voting Regressor performed comparably to GB, with a slightly lower validation R² (88.27%) but better generalization than Random Forest (RF) and Decision Tree (DT). Random Forest showed stable but slightly lower accuracy than GB, while Decision Tree had the **lowest R²** across both validation (87.02) and testing (81.27%), reflecting its limited predictive stability compared to ensemble methods.

Overall, **Gradient Boosting is the most effective regression model**, balancing high predictive power with low error rates.

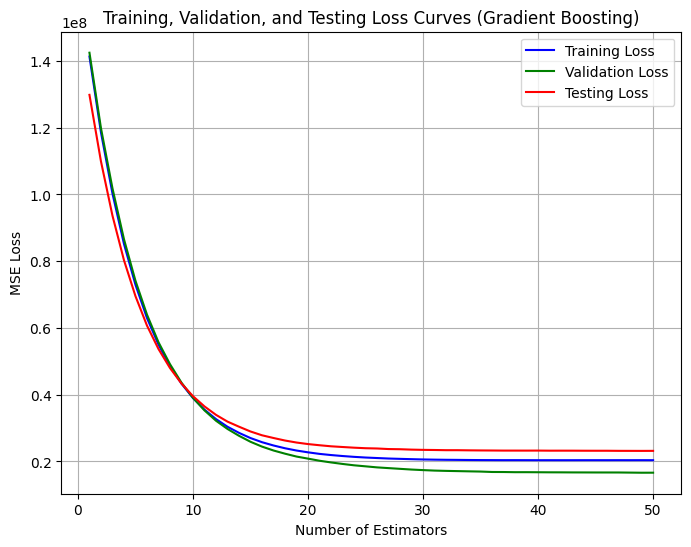


Figure Learning Curve – GB Regressor

The GB learning curve demonstrates **optimal convergence behavior** with MSE loss dropping rapidly from ~1.4 to ~0.3 within the first 10-15 estimators, then stabilizing around 0.2-0.25. The close alignment between training, validation, and testing curves indicates **excellent generalization with no overfitting,** suggesting optimal performance around 15-20 estimators with diminishing returns thereafter. (Learning curves for other algorithms are provided in the Appendix.)

### ****Classification Models****

#### ****Data Balancing for Classification****

The dataset exhibited class imbalance with Medium charges (668), Low charges (335), and High charges (334). **SMOTE (Synthetic Minority Oversampling Technique)** was applied to balance the classes, generating synthetic samples to equalize all categories at 668 instances each.

Table Class Distribution Before and After SMOTE Balancing

|  |  |  |
| --- | --- | --- |
| Charges Category | Before SMOTE | After SMOTE |
| Medium | 668 | 668 |
| Low | 335 | 668 |
| High | 334 | 668 |
| ****Total**** | ****1,337**** | ****2,004**** |

The balanced dataset was then split into training (60%), validation (20%), and testing (20%) sets, resulting in 1,202 training samples, 401 validation samples, and 401 test samples.

SMOTE balancing was applied exclusively for classification algorithms to ensure equal representation across all charge categories and prevent bias toward the majority class. Regression algorithms used the original unbalanced dataset since they predict continuous values rather than discrete classes, making class distribution balancing irrelevant for their performance.

In classification tasks, all models delivered high and consistent performance, with validation accuracies ranging between 0.91 and 0.92. Gradient Boosting and Hard Voting achieved the highest validation accuracy (0.92) and retained strong performance on the test set (0.88).

Table 3 Results Analysis - Classification Models

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Classification Algorithms** | | **Accuracy** | **Precision** | **Recall** | **F1- score** |
| **Validation** | **GB** | 0.92 | 0.93 | 0.92 | 0.92 |
| **DT** | 0.91 | 0.92 | 0.91 | 0.91 |
| **RF** | 0.91 | 0.91 | 0.91 | 0.91 |
| **Hard Voting** | **0.92** | **0.92** | **0.92** | **0.92** |
| **Soft Voting** | 0.91 | 0.92 | 0.91 | 0.91 |
| **Testing** | **GB** | 0.88 | 0.89 | 0.88 | 0.88 |
| **DT** | 0.88 | 0.89 | 0.88 | 0.87 |
| **RF** | 0.86 | 0.87 | 0.86 | 0.86 |
| **Hard Voting** | 0.88 | 0.89 | 0.88 | 0.88 |
| **Soft Voting** | 0.88 | 0.89 | 0.88 | 0.88 |



Figure Chart of Results Analysis - Classification Models

Precision, recall, and F1-scores were nearly identical across models, indicating balanced classification capability and no significant bias toward either class. Random Forest performed slightly lower (0.86 test accuracy) than the other ensemble models.

To further assess model performance, confusion matrices were evaluated for all classifiers. They confirm that the models classified each charges category with minimal misclassification.

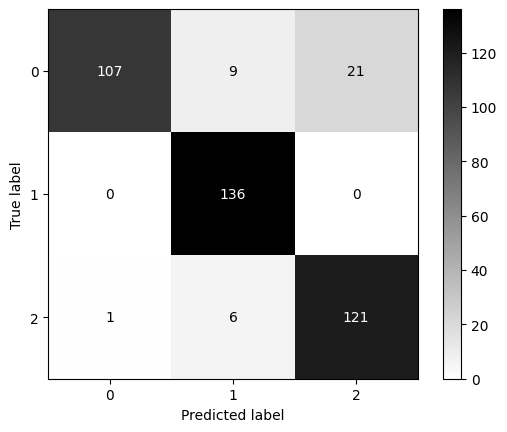


Figure 16 Confusion Matrix - GB Validation

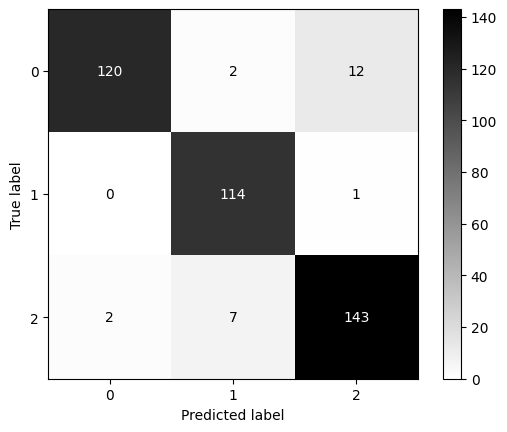


Figure Confusion Matrix - GB Testing

Confusion matrices for other models (DT, RF, Hard Voting, Soft Voting) are provided in the **Appendix.**

Receiver Operating Characteristic (ROC) curves were plotted for all classification models to evaluate their discriminative ability. The area under the curve (AUC) indicates how well each model separates the three charge categories. For example,  **Gradient Boosting Classifier (GBC) ROC Curve is shown below.**

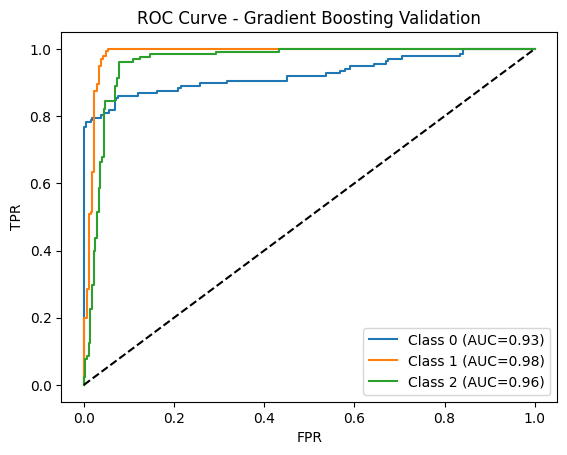


Figure ROC Curve - GB Validation Set

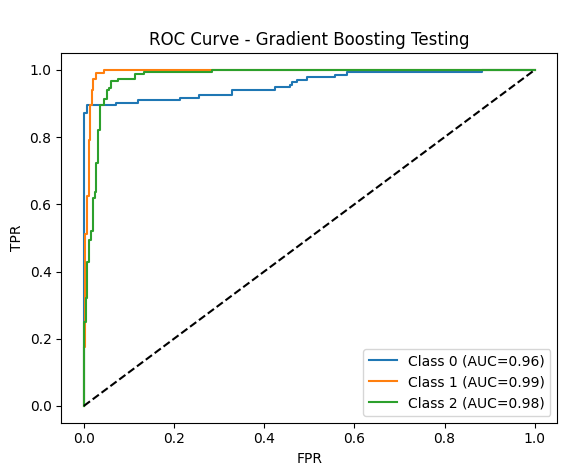


Figure ROC Curve - GB Testing Set

ROC curves for the remaining models (DT, RF, Hard Voting, Soft Voting) are provided in the **Appendix**.

Given the minimal variation, **Gradient Boosting and Hard Voting** can be considered the top-performing classifiers due to their stable and slightly superior accuracy across both validation and test sets.

## ****Key Findings of the Research****

The research clearly demonstrated that hybrid ensemble models significantly enhance prediction accuracy for insurance cost estimation. Both soft voting (probability-based averaging) and hard voting (majority-based voting) ensemble approaches consistently outperformed individual models such as Gradient Boosting, Decision Tree, and Random Forest. For example, the Voting Regressor achieved a validation R² score of 88.27%, slightly surpassing standalone models like Gradient Boosting (88.80%), Decision Tree (87.02%), and Random Forest (87.58%). On the testing dataset, the Voting Regressor maintained superior generalization with an R² score of 82.47%, indicating that combining multiple base learners improves robustness and predictive power across unseen data.

Among the individual regressors examined, Gradient Boosting emerged as the strongest standalone model. It achieved the highest R² scores of 88.80% on the validation set and 82.90% on the testing set, reflecting its ability to capture complex nonlinear relationships within the insurance dataset. Additionally, Gradient Boosting recorded the lowest error metrics, with a validation RMSE of 4077.00 and a testing RMSE of 4812.98, highlighting its precision in cost prediction. This confirms Gradient Boosting as a highly effective and reliable algorithm for insurance cost modeling when used independently.

## Answer for Research Question

1. Which hybrid approach—soft combination (Probability based) or hard combination  voting based)—yields better generalization and performance in insurance prediction?

The research investigates whether the soft hybrid approach, which combines base learner probabilities through weighted averaging, or the hard hybrid approach, which aggregates predictions via majority voting, provides better generalization and predictive accuracy in insurance cost prediction. Results showed that soft voting models achieved an R² of 88.27%, MAE of 2293.54, and RMSE of 4173.01 on validation data, slightly outperforming hard voting models with an R² of 88.26%, MAE of 2293.54, and RMSE of 4173.01. On testing data, soft voting also performed marginally better, indicating improved handling of uncertainty and robustness on structured insurance data.

## ****Comparative Analysis of Current Research with Previous Studies****

This project enhances insurance cost prediction by comparing soft (probability-based) and hard (voting-based) hybrid ensemble models on structured insurance data. Previous studies mainly focused on individual machine learning models or hybrid methods combining statistical and neural techniques, but rarely contrasted soft versus hard hybrid strategies on the same dataset.

For example, Panda et al. (2022) applied various regression models, facing scalability issues and without using ensemble hybrids. Patra et al. (2024) evaluated models like KNN and XGBoost, with XGBoost achieving strong results (R² ~86.8%), yet did not explore ensemble voting. Similarly, Hassan et al. (2021) and Albalawi et al. (2023) focused on individual or ensemble tree methods without systematically comparing hybrid voting approaches.

The research has employed Gradient Boosting, Decision Tree, Random Forest learners in a soft and hard voting situation, and has been successful in reaching competitive R squared values (validation ~88.8%, testing ~82.9%) and performance in classification (~0.91 to 0.92). It also focuses more on feature analysis and model interpretability that has been ignored in the previous work.

This project gives a strong, reproducible pipeline that offers optimal hyperparameter tuning. Comparison of hybrid ensemble algorithm introduces useful ideas in enhancing the generalization and accuracy of predicting the whole existence of insurance costs and shifting the direction of the industry towards scalable, interpretable solutions.

# Conclusion

This research was based on developing and comparing soft and hard hybrid ensemble machine learning models to forecast the insurance costs relying upon structured insurance data. These results demonstrate that the hybrid models are more effective than single base learners (Gradient Boosting, Decision Tree, and Random Forest), which confirms how the combination of several models could enhance the generalization and robustness. Gradient Boosting among individual models was the best performer with R2 scores as high as 88.8 on the validation and 82.9 during testing dataset. Competitively, soft voting (probabilistic weighted averaging) and hard voting (majority voting) hybrid methods turned out to be competitive, and soft voting obtained a minor advantage based on predictive accuracy and measures of error. This implies that use of probabilistic information present in base learners aids in improved modelling of uncertainties, which amounts to improved stability of predictions. It was also shown that exploratory data analysis and proper preprocessing, i.e., one-hot encoding is essential in readying the insurance dataset to be successfully used in modelling.

## Future Work

To be extended in future, the research would be enhanced with inclusion of other characteristics like lifestyle, socio-economic, and medical history factors, which would have a consequence on the insurance expenses, as they were not added in the study. Further improvement of prediction accuracy could be achieved by the exploration of better hybrid models that combine neural networks with traditional machine learning techniques. Moreover, it is possible that the use of metaheuristic algorithms during hyperparameter tuning can make the model perform better than in case of grid search. Finally, application of these hybrid models to bigger and more diverse data such as cross country insurance data would confirm in real life and how they could be ultimately generalized in other diverse insurance contexts.

# References

AbdElminaam, D.S., Farouk, M., Shaker, N., Elrashidy, O. and Elazab, R., (2024). An Efficient Framework for Predict Medical Insurance Costs Using Machine Learning. *Journal of Computing and Communication*, *3*(2), pp.55-64.

Albalawi, S., Alshahrani, L., Albalawi, N., Alharbi, R. and Alhakamy, A.A., 2023. Prediction of healthcare insurance costs. *Computers and Informatics*, *3*(1), pp.9-18.

Bhargavi, R. and Arumugam, S., (2024). Predictive Analytics in Healthcare: A Hybrid Model for Efficient Medical Cost Estimation for Personalized Insurance Pricing.

Gorgipour, A., (2022), Feature selection using random forests and bootstrap sampling on cancer drug screening data. Bachelor’s thesis, Heinrich Heine University Düsseldorf, Germany. (Available at: <https://www.cs.hhu.de/fileadmin/redaktion/Fakultaeten/Mathematisch-Naturwissenschaftliche_Fakultaet/Informatik/Algorithmische_Bioinformatik/Bachelor-_Masterarbeiten/2685083_ba_ifo_AbschlArbeit_klau_marscht_angor106_20220912_1325.pdf>) [Accessed 17th July 2025]

Lantz, B., (2013). Machine Learning with R. Birmingham: Packt Publishing Ltd.

Panda, S., Purkayastha, B., Das, D., Chakraborty, M. and Biswas, S.K., (2022), May. Health insurance cost prediction using regression models. In *2022 International conference on machine learning, big data, cloud and parallel computing (COM-IT-CON)* (Vol. 1, pp. 168-173). IEEE.

Patra, G.K., Kuraku, C., Konkimalla, S., Boddapati, V.N., Sarisa, M. and Reddy, M.S., (2024). An Analysis and Prediction of Health Insurance Costs Using Machine Learning-Based Regressor Techniques. *Journal of Data Analysis and Information Processing*, *12*(4), pp.581-596.

Shyamala Devi, M., P. Swathi, M. Purushotham Reddy, V. Deepak Varma, A. Praveen Kumar Reddy, Saranya Vivekanandan, and Priyanka Moorthy., (2021). "Linear and ensembling regression based health cost insurance prediction using machine learning." In *Smart Computing Techniques and Applications: Proceedings of the Fourth International Conference on Smart Computing and Informatics, Volume 2*, pp. 495-503. Springer Singapore, 2021.

Smith, J. and Owen, A., (2024), ‘Simulating market conditions for insurance premium optimization’, Unpublished research manuscript, June 2024. (Available at: <https://www.researchgate.net/profile/Antony-Owen/publication/389068498_Simulating_Market_Conditions_for_Insurance_Premium_Optimization/links/67b3e1324c479b26c9e54632/Simulating-Market-Conditions-for-Insurance-Premium-Optimization.pdf> ) [Accessed 14th July 2025]

ul Hassan, C.A., Iqbal, J., Hussain, S., AlSalman, H., Mosleh, M.A. and Sajid Ullah, S., (2021). A computational intelligence approach for predicting medical insurance cost. *Mathematical Problems in Engineering*, *2021*(1), p.1162553.

Wilson, A.A., Nehme, A., Dhyani, A. and Mahbub, K., (2024). A Comparison of Generalised Linear Modelling with Machine Learning Approaches for Predicting Loss Cost in Motor Insurance. *Risks*, *12*(4), p.62.

# Appendix

## Code

**Regression - Insurance Cost Prediction**

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import train\_test\_split

import math

from sklearn.metrics import r2\_score

from sklearn.metrics import mean\_absolute\_error

from sklearn.metrics import mean\_squared\_error

from sklearn.model\_selection import GridSearchCV

import warnings

warnings.filterwarnings('ignore')

insurance\_df = pd.read\_csv('insurance.csv')

insurance\_df.head()

insurance\_df.tail()

print("Insurance Cost Data Shape:",insurance\_df.shape)

print("\nInformation of Insurance Cost:\n")

insurance\_df.info()

print("\nStatistical Analysis of Insurance Cost:\n")

insurance\_df.describe()

insurance\_df['charges'].value\_counts()

### Dropping Null and Duplicates

print("NaN values present in Insurance Cost: ",insurance\_df.isnull().values.sum())

print("Duplicates present in Insurance Cost: ",insurance\_df.duplicated().sum())

insurance\_df = insurance\_df.drop\_duplicates(keep='first')

print("Final Duplicates Count: ",insurance\_df.duplicated().sum())

### Visualization of Numeric Columns

# Selecting all the numerical columns (int and float)

numeric\_cols = insurance\_df.select\_dtypes(include=['int64', 'float64']).columns

fig, axes = plt.subplots(1, len(numeric\_cols), figsize=(5 \* len(numeric\_cols), 4))

for j, col in enumerate(numeric\_cols):

sns.histplot(insurance\_df[col], ax=axes[j], color='darkcyan')

axes[j].set\_title(f'{col} Distribution')

plt.tight\_layout()

plt.show()

### Visualization of Object Columns

# Selecting all the object columns

object\_columns = insurance\_df.select\_dtypes(include='object').columns

fig, axes = plt.subplots(1, len(object\_columns), figsize=(5 \* len(object\_columns), 4))

for j, col in enumerate(object\_columns):

if j == len(object\_columns) - 1:

pie\_colors = ['#ff9999', '#c2c2f0', '#ffb3e6', '#b3d9ff']

else:

pie\_colors = ['#ffcc99', '#66b3ff', '#99ff99', '#fabf15']

insurance\_df[col].value\_counts().plot.pie(

ax=axes[j],

autopct='%1.1f%%',

startangle=90,

shadow=True,

colors=pie\_colors

)

axes[j].set\_title(f'{col.capitalize()} Distribution')

axes[j].set\_ylabel('') # Hide y-axis label

plt.tight\_layout()

plt.show()

# One-hot encoding for 'sex', 'smoker', and 'region' columns

insurance\_df = pd.get\_dummies(insurance\_df, columns=['sex', 'smoker', 'region'], drop\_first=False)

insurance\_df

insurance\_df.to\_csv('insurance\_cost\_prediction.csv', index=False)

insurance\_df.info()

Splitting

insuranceX = insurance\_df.drop('charges',axis=1)

insuranceY = insurance\_df['charges']

ins\_Xtr, ins\_Xts, ins\_Ytr, ins\_Yts = train\_test\_split(insuranceX, insuranceY, test\_size=0.4, random\_state=20)

ins\_Xva, ins\_Xts, ins\_Yva, ins\_Yts = train\_test\_split(ins\_Xts, ins\_Yts, test\_size=0.5, random\_state=20)

ins\_Xtr

ins\_Xts

ins\_Xva

### Gradient Boosting Regressor

# train, validation and testing of Gradient Boosting Regressor

from sklearn.ensemble import GradientBoostingRegressor

params\_regressors = { 'loss':['squared\_error', 'absolute\_error', 'huber', 'quantile'],

'n\_estimators': [100, 30, 50, 80],

'criterion': ['friedman\_mse', 'squared\_error']

}

ins\_model = GradientBoostingRegressor()

ins\_model = GridSearchCV(ins\_model, params\_regressors, cv=2)

ins\_model.fit(ins\_Xtr, ins\_Ytr)

print("Chosen Parameters by GB Regressor:\n")

print(ins\_model.best\_params\_)

ins\_model = ins\_model.best\_estimator\_

ins\_model.fit(ins\_Xtr, ins\_Ytr)

ins\_Ypred = ins\_model.predict(ins\_Xva)

print("\n==Validation of Gradient Boosting==\n")

print("R2 Score :", r2\_score(ins\_Yva, ins\_Ypred)\*100)

print("MAE :", mean\_absolute\_error(ins\_Yva, ins\_Ypred))

print("MSE :", mean\_squared\_error(ins\_Yva, ins\_Ypred))

print("RMSE :", math.sqrt(mean\_squared\_error(ins\_Yva, ins\_Ypred)))

ins\_Ypred = ins\_model.predict(ins\_Xts)

print("\n==Testing of Gradient Boosting==\n")

print("R2 Score :", r2\_score(ins\_Yts, ins\_Ypred)\*100)

print("MAE :", mean\_absolute\_error(ins\_Yts, ins\_Ypred))

print("MSE :", mean\_squared\_error(ins\_Yts, ins\_Ypred))

print("RMSE :", math.sqrt(mean\_squared\_error(ins\_Yts, ins\_Ypred)))

### Decision Tree Regression

# train, validation and testing of Decision Tree Regression

from sklearn.tree import DecisionTreeRegressor

params\_regressors = { 'criterion':['squared\_error','friedman\_mse','absolute\_error','poisson'],

'splitter': ['best', 'random'],

'max\_depth': [1, 3, 5, 8]

}

ins\_model = DecisionTreeRegressor()

ins\_model = GridSearchCV(ins\_model, params\_regressors, cv=2)

ins\_model.fit(ins\_Xtr, ins\_Ytr)

print("Chosen Parameters by DT Regressor:\n")

print(ins\_model.best\_params\_)

ins\_model = ins\_model.best\_estimator\_

ins\_model.fit(ins\_Xtr, ins\_Ytr)

ins\_Ypred = ins\_model.predict(ins\_Xva)

print("\n==Validation of Decision Tree Regressor==\n")

print("R2 Score :", r2\_score(ins\_Yva, ins\_Ypred)\*100)

print("MAE :", mean\_absolute\_error(ins\_Yva, ins\_Ypred))

print("MSE :", mean\_squared\_error(ins\_Yva, ins\_Ypred))

print("RMSE :", math.sqrt(mean\_squared\_error(ins\_Yva, ins\_Ypred)))

ins\_Ypred = ins\_model.predict(ins\_Xts)

print("\n==Testing of Decision Tree Regressor==\n")

print("R2 Score :", r2\_score(ins\_Yts, ins\_Ypred)\*100)

print("MAE :", mean\_absolute\_error(ins\_Yts, ins\_Ypred))

print("MSE :", mean\_squared\_error(ins\_Yts, ins\_Ypred))

print("RMSE :", math.sqrt(mean\_squared\_error(ins\_Yts, ins\_Ypred)))

!apt-get install graphviz

from sklearn.tree import export\_graphviz

import graphviz

from IPython.display import display

dot\_data = export\_graphviz(

ins\_model,

out\_file=None,

feature\_names=ins\_Xtr.columns,

filled=True,

rounded=True,

special\_characters=True

)

graph = graphviz.Source(dot\_data)

graph.render("DTC", format="pdf", cleanup=True);

display(graph)

### Random Forest Regression

# train, validation and testing of Random Forest Regression

from sklearn.ensemble import RandomForestRegressor

params\_regressors = { 'criterion':['squared\_error','friedman\_mse','absolute\_error','poisson'],

'n\_estimators': [20, 30, 100],

'max\_depth': [1, 3, 5, 8]

}

ins\_model = RandomForestRegressor()

ins\_model = GridSearchCV(ins\_model, params\_regressors, cv=2)

ins\_model.fit(ins\_Xtr, ins\_Ytr)

print("Chosen Parameters by RF Regressor:\n")

print(ins\_model.best\_params\_)

ins\_model = ins\_model.best\_estimator\_

ins\_model.fit(ins\_Xtr, ins\_Ytr)

ins\_Ypred = ins\_model.predict(ins\_Xva)

print("\n==Validation of Random Forest Regressor==\n")

print("R2 Score :", r2\_score(ins\_Yva, ins\_Ypred)\*100)

print("MAE :", mean\_absolute\_error(ins\_Yva, ins\_Ypred))

print("MSE :", mean\_squared\_error(ins\_Yva, ins\_Ypred))

print("RMSE :", math.sqrt(mean\_squared\_error(ins\_Yva, ins\_Ypred)))

ins\_Ypred = ins\_model.predict(ins\_Xts)

print("\n==Testing of Random Forest Regressor==\n")

print("R2 Score :", r2\_score(ins\_Yts, ins\_Ypred)\*100)

print("MAE :", mean\_absolute\_error(ins\_Yts, ins\_Ypred))

print("MSE :", mean\_squared\_error(ins\_Yts, ins\_Ypred))

print("RMSE :", math.sqrt(mean\_squared\_error(ins\_Yts, ins\_Ypred)))

# Visualizing the 1st tree from the Random Forest

estimator = ins\_model.estimators\_[0]

dot\_data = export\_graphviz(

estimator,

out\_file=None,

feature\_names=ins\_Xtr.columns,

filled=True,

rounded=True,

special\_characters=True

)

graph = graphviz.Source(dot\_data)

graph.render("RFC", format="pdf", cleanup=True);

display(graph)

### Combining Tree Models using Voting Regressor

# train, validation and testing of Hard Voting

from sklearn.ensemble import VotingRegressor as insurance\_VoteRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import GradientBoostingRegressor

ins\_model\_gbc = GradientBoostingRegressor(criterion= 'friedman\_mse', loss= 'huber', n\_estimators= 50)

ins\_model\_dtc = DecisionTreeRegressor(criterion= 'poisson', max\_depth= 3, splitter= 'best')

ins\_model\_rfc = RandomForestRegressor(criterion= 'poisson', max\_depth= 3, n\_estimators= 30)

ins\_model = insurance\_VoteRegressor(estimators=[('GBC', ins\_model\_gbc), ('DTC', ins\_model\_dtc), ('RFC', ins\_model\_rfc)])

ins\_model.fit(ins\_Xtr, ins\_Ytr)

ins\_Ypred = ins\_model.predict(ins\_Xva)

print("\n==Validation of GBC, DTC and RFC using Voting Regressor==\n")

print("R2 Score :", r2\_score(ins\_Yva, ins\_Ypred)\*100)

print("MAE :", mean\_absolute\_error(ins\_Yva, ins\_Ypred))

print("MSE :", mean\_squared\_error(ins\_Yva, ins\_Ypred))

print("RMSE :", math.sqrt(mean\_squared\_error(ins\_Yva, ins\_Ypred)))

ins\_Ypred = ins\_model.predict(ins\_Xts)

print("\n==Testing of GBC, DTC and RFC using Voting Regressor==\n")

print("R2 Score :", r2\_score(ins\_Yts, ins\_Ypred)\*100)

print("MAE :", mean\_absolute\_error(ins\_Yts, ins\_Ypred))

print("MSE :", mean\_squared\_error(ins\_Yts, ins\_Ypred))

print("RMSE :", math.sqrt(mean\_squared\_error(ins\_Yts, ins\_Ypred)))

# Get the DTC model from Voting Regressor

dtc\_model = ins\_model.named\_estimators\_['DTC']

dot\_data = export\_graphviz(

dtc\_model,

out\_file=None,

feature\_names=ins\_Xtr.columns,

filled=True,

rounded=True,

special\_characters=True

)

graph = graphviz.Source(dot\_data)

graph.render("DTC-Voting Regressor", format="pdf", cleanup=True);

display(graph)

# Get the RFC model from Voting Regressor

rfc\_model = ins\_model.named\_estimators\_['RFC']

# Visualize the first tree from the Random Forest

estimator = rfc\_model.estimators\_[0]

dot\_data = export\_graphviz(

estimator,

out\_file=None,

feature\_names=ins\_Xtr.columns,

filled=True,

rounded=True,

special\_characters=True

)

graph = graphviz.Source(dot\_data)

graph.render("RFC-Voting Regressor", format="pdf", cleanup=True);

display(graph)

**Classification - Using Soft/Hard Voting Classifiers**

### \*\*Note:\*\* The terms "soft" and "hard" are commonly used in classification tasks, not regression. So convert the target attribute into categorical and implement the hard and soft voting classifiers.

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import train\_test\_split

import math

from sklearn.metrics import r2\_score

from sklearn.metrics import mean\_absolute\_error

from sklearn.metrics import mean\_squared\_error

from sklearn.model\_selection import GridSearchCV

import warnings

warnings.filterwarnings('ignore')

insurance\_df = pd.read\_csv('insurance.csv')

insurance\_df.head()

insurance\_df.tail()

print("Insurance Cost Data Shape:",insurance\_df.shape)

print("\nInformation of Insurance Cost:\n")

insurance\_df.info()

print("\nStatistical Analysis of Insurance Cost:\n")

insurance\_df.describe()

insurance\_df['charges'].value\_counts()

### Dropping Null and Duplicates

print("NaN values present in Insurance Cost: ",insurance\_df.isnull().values.sum())

print("Duplicates present in Insurance Cost: ",insurance\_df.duplicated().sum())

insurance\_df = insurance\_df.drop\_duplicates(keep='first')

print("Final Duplicates Count: ",insurance\_df.duplicated().sum())

### Plots for Insurance Cost Prediction

Histogram - Target

sns.histplot(insurance\_df['charges'], kde=True, color='darkcyan')

Histogram - Bmi

sns.histplot(insurance\_df['bmi'], kde=True, color='tan')

Pie Chart - Region

insurance\_df['region'].value\_counts().plot.pie(autopct='%1.1f%%',startangle=90,shadow=True, colors=['#ffcc99', '#66b3ff', '#99ff99', '#fabf15'])

Pie Chart - Smoker

insurance\_df['smoker'].value\_counts().plot.pie(autopct='%1.1f%%',startangle=90,shadow=True, colors=['#99ff99', '#fabf15'])

insurance\_Q1 = insurance\_df['charges'].quantile(0.25)

insurance\_Q2 = insurance\_df['charges'].median() # or quantile(0.5)

insurance\_Q3 = insurance\_df['charges'].quantile(0.75)

print(f"Charges - 25% (Q1): {insurance\_Q1}, \nCharges - 50% (Q2): {insurance\_Q2}, \nCharges - 75% (Q3): {insurance\_Q3}")

def categorize\_charge(charge):

if charge <= insurance\_Q1:

return 'Low'

elif charge <= insurance\_Q3:

return 'Medium'

else:

return 'High'

insurance\_df['charges\_category'] = insurance\_df['charges'].apply(categorize\_charge)

insurance\_df

insurance\_df['charges\_category'].value\_counts()

sns.countplot(data=insurance\_df, x='charges\_category', palette='pastel')

plt.title('Distribution of Insurance Charges Categories')

plt.show()

del insurance\_df['charges']

# One-hot encoding for 'sex', 'smoker', and 'region' columns

insurance\_df = pd.get\_dummies(insurance\_df, columns=['sex', 'smoker', 'region'], drop\_first=False)

insurance\_df

Splitting

insuranceX = insurance\_df.drop('charges\_category',axis=1)

insuranceY = insurance\_df['charges\_category']

ins\_Xtr, ins\_Xts, ins\_Ytr, ins\_Yts = train\_test\_split(insuranceX, insuranceY, test\_size=0.4, random\_state=20)

ins\_Xva, ins\_Xts, ins\_Yva, ins\_Yts = train\_test\_split(ins\_Xts, ins\_Yts, test\_size=0.5, random\_state=20)

ins\_Xtr

ins\_Xts

ins\_Xva

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import ConfusionMatrixDisplay

### Gradient Boosting Classifier

# train, validation and testing of Gradient Boosting Classifier

from sklearn.ensemble import GradientBoostingClassifier

ins\_params\_classif = { 'loss':['exponential', 'log\_loss'],

'n\_estimators': [100, 30, 50, 80],

'criterion': ['friedman\_mse', 'squared\_error']

}

ins\_model\_gbc = GradientBoostingClassifier()

ins\_model\_gbc = GridSearchCV(ins\_model\_gbc, ins\_params\_classif, cv=2)

ins\_model\_gbc.fit(ins\_Xtr, ins\_Ytr)

print("Chosen Parameters by GB Classifier:\n")

print(ins\_model\_gbc.best\_params\_)

ins\_model\_gbc = ins\_model\_gbc.best\_estimator\_

ins\_model\_gbc.fit(ins\_Xtr, ins\_Ytr)

ins\_Ypred = ins\_model\_gbc.predict(ins\_Xva)

print("\n==Validation of Gradient Boosting==\n")

print(classification\_report(ins\_Yva, ins\_Ypred))

confus\_matrix = confusion\_matrix(ins\_Yva, ins\_Ypred)

ConfusionMatrixDisplay(confusion\_matrix = confus\_matrix, display\_labels = [0,1,2]).plot(cmap='binary')

plt.show()

ins\_Ypred = ins\_model\_gbc.predict(ins\_Xts)

print("\n==Testing of Gradient Boosting==\n")

print(classification\_report(ins\_Yts, ins\_Ypred))

confus\_matrix = confusion\_matrix(ins\_Yts, ins\_Ypred)

ConfusionMatrixDisplay(confusion\_matrix = confus\_matrix, display\_labels = [0,1,2]).plot(cmap='binary')

plt.show()

### Decision Tree Classifier

# train, validation and testing of Decision Tree Classifier

from sklearn.tree import DecisionTreeClassifier

ins\_params\_classif = { 'min\_samples\_split':[1, 2, 5, 6],

'max\_depth': [10, 3, 5, 8],

'criterion': ['gini', 'entropy', 'log\_loss']

}

ins\_model\_dtc = DecisionTreeClassifier()

ins\_model\_dtc = GridSearchCV(ins\_model\_dtc, ins\_params\_classif, cv=2)

ins\_model\_dtc.fit(ins\_Xtr, ins\_Ytr)

print("Chosen Parameters by DT Classifier:\n")

print(ins\_model\_dtc.best\_params\_)

ins\_model\_dtc = ins\_model\_dtc.best\_estimator\_

ins\_model\_dtc.fit(ins\_Xtr, ins\_Ytr)

ins\_Ypred = ins\_model\_dtc.predict(ins\_Xva)

print("\n==Validation of Decision Tree==\n")

print(classification\_report(ins\_Yva, ins\_Ypred))

confus\_matrix = confusion\_matrix(ins\_Yva, ins\_Ypred)

ConfusionMatrixDisplay(confusion\_matrix = confus\_matrix, display\_labels = [0,1,2]).plot(cmap='binary')

plt.show()

ins\_Ypred = ins\_model\_dtc.predict(ins\_Xts)

print("\n==Testing of Decision Tree==\n")

print(classification\_report(ins\_Yts, ins\_Ypred))

confus\_matrix = confusion\_matrix(ins\_Yts, ins\_Ypred)

ConfusionMatrixDisplay(confusion\_matrix = confus\_matrix, display\_labels = [0,1,2]).plot(cmap='binary')

plt.show()

### Random Forest Classifier

# train, validation and testing of Random Forest Classifier

from sklearn.ensemble import RandomForestClassifier

ins\_params\_classif = { 'max\_depth':[10, 3, 5, 8],

'n\_estimators': [100, 30, 50, 80],

'criterion': ['gini', 'entropy', 'log\_loss']}

ins\_model\_rfc = RandomForestClassifier()

ins\_model\_rfc = GridSearchCV(ins\_model\_rfc, ins\_params\_classif, cv=2)

ins\_model\_rfc.fit(ins\_Xtr, ins\_Ytr)

print("Chosen Parameters by RF Classifier:\n")

print(ins\_model\_rfc.best\_params\_)

ins\_model\_rfc = ins\_model\_rfc.best\_estimator\_

ins\_model\_rfc.fit(ins\_Xtr, ins\_Ytr)

ins\_Ypred = ins\_model\_rfc.predict(ins\_Xva)

print("\n==Validation of Random Forest==\n")

print(classification\_report(ins\_Yva, ins\_Ypred))

confus\_matrix = confusion\_matrix(ins\_Yva, ins\_Ypred)

ConfusionMatrixDisplay(confusion\_matrix = confus\_matrix, display\_labels = [0,1,2]).plot(cmap='binary')

plt.show()

ins\_Ypred = ins\_model\_rfc.predict(ins\_Xts)

print("\n==Testing of Random Forest==\n")

print(classification\_report(ins\_Yts, ins\_Ypred))

confus\_matrix = confusion\_matrix(ins\_Yts, ins\_Ypred)

ConfusionMatrixDisplay(confusion\_matrix = confus\_matrix, display\_labels = [0,1,2]).plot(cmap='binary')

plt.show()

### Using 'hard' Voting Classifier Combinations

# train, validation and testing of Hard Voting

from sklearn.ensemble import VotingClassifier

ins\_model\_gbc = GradientBoostingClassifier(criterion= 'squared\_error', loss= 'log\_loss', n\_estimators= 30)

ins\_model\_dtc = DecisionTreeClassifier(criterion= 'log\_loss', max\_depth= 5, min\_samples\_split= 2)

ins\_model\_rfc = RandomForestClassifier(criterion= 'log\_loss', max\_depth= 5, n\_estimators= 50)

ins\_model = VotingClassifier(estimators=[('GBC', ins\_model\_gbc), ('DTC', ins\_model\_dtc), ('RFC', ins\_model\_rfc)], voting='hard')

ins\_model.fit(ins\_Xtr, ins\_Ytr)

ins\_Ypred = ins\_model.predict(ins\_Xva)

print("\n==Validation of GBC, DTC and RFC using Hard Voting==\n")

print(classification\_report(ins\_Yva, ins\_Ypred))

confus\_matrix = confusion\_matrix(ins\_Yva, ins\_Ypred)

ConfusionMatrixDisplay(confusion\_matrix = confus\_matrix, display\_labels = [0,1,2]).plot(cmap='binary')

plt.show()

ins\_Ypred = ins\_model.predict(ins\_Xts)

print("\n==Testing of GBC, DTC and RFC using Hard Voting==\n")

print(classification\_report(ins\_Yts, ins\_Ypred))

confus\_matrix = confusion\_matrix(ins\_Yts, ins\_Ypred)

ConfusionMatrixDisplay(confusion\_matrix = confus\_matrix, display\_labels = [0,1,2]).plot(cmap='binary')

plt.show()

### Using 'soft' Voting Classifier Combinations

# train, validation and testing of Soft Voting

from sklearn.ensemble import VotingClassifier

ins\_model\_gbc = GradientBoostingClassifier(criterion= 'squared\_error', loss= 'log\_loss', n\_estimators= 30)

ins\_model\_dtc = DecisionTreeClassifier(criterion= 'log\_loss', max\_depth= 5, min\_samples\_split= 2)

ins\_model\_rfc = RandomForestClassifier(criterion= 'log\_loss', max\_depth= 5, n\_estimators= 50)

ins\_model = VotingClassifier(estimators=[('GBC', ins\_model\_gbc), ('DTC', ins\_model\_dtc), ('RFC', ins\_model\_rfc)], voting='soft')

ins\_model.fit(ins\_Xtr, ins\_Ytr)

ins\_Ypred = ins\_model.predict(ins\_Xva)

print("\n==Validation of GBC, DTC and RFC using Soft Voting==\n")

print(classification\_report(ins\_Yva, ins\_Ypred))

confus\_matrix = confusion\_matrix(ins\_Yva, ins\_Ypred)

ConfusionMatrixDisplay(confusion\_matrix = confus\_matrix, display\_labels = [0,1,2]).plot(cmap='binary')

plt.show()

ins\_Ypred = ins\_model.predict(ins\_Xts)

print("\n==Testing of GBC, DTC and RFC using Soft Voting==\n")

print(classification\_report(ins\_Yts, ins\_Ypred))

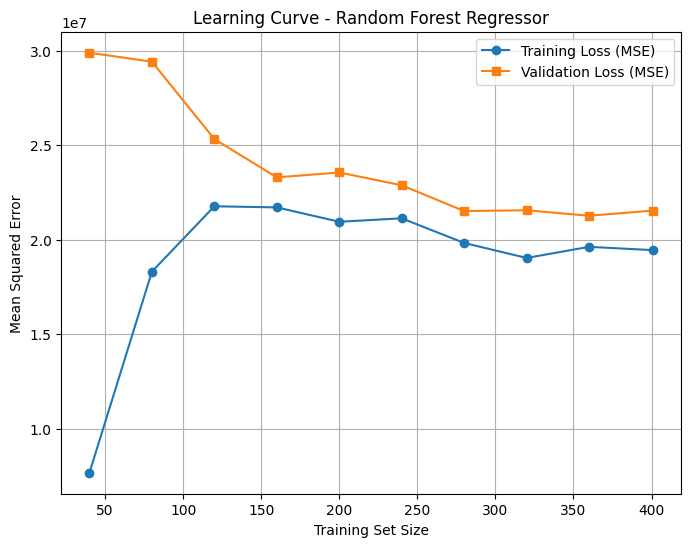
confus\_matrix = confusion\_matrix(ins\_Yts, ins\_Ypred)

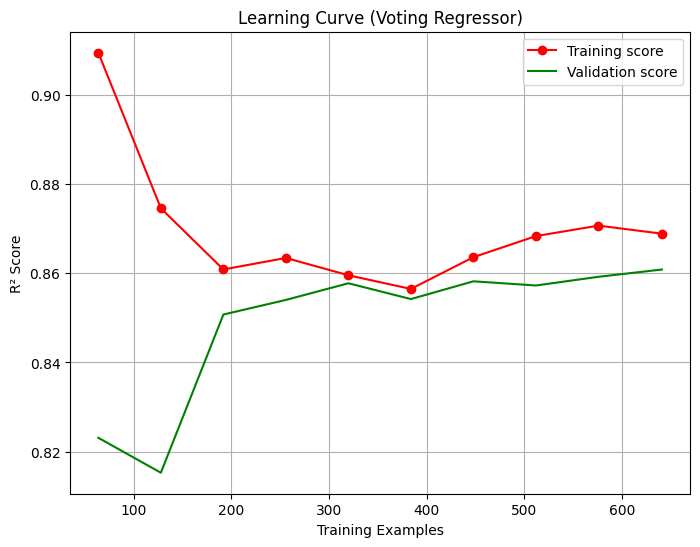
ConfusionMatrixDisplay(confusion\_matrix = confus\_matrix, display\_labels = [0,1,2]).plot(cmap='binary')

plt.show()

## Learning Curves

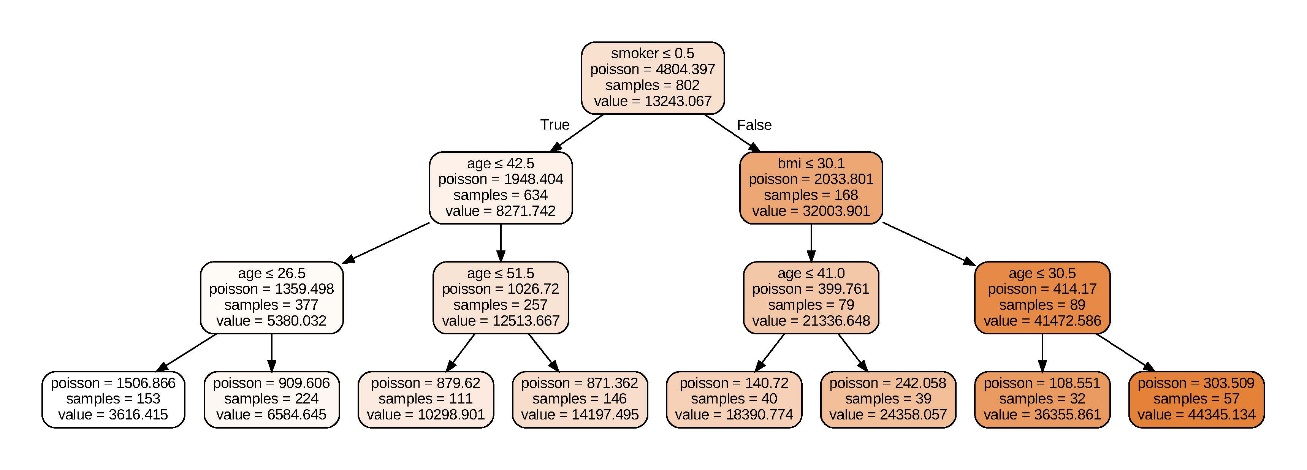




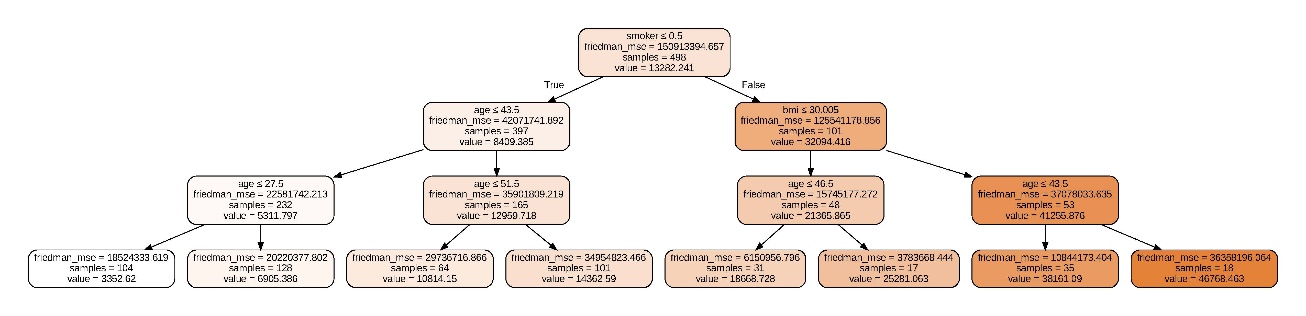


## Graphs

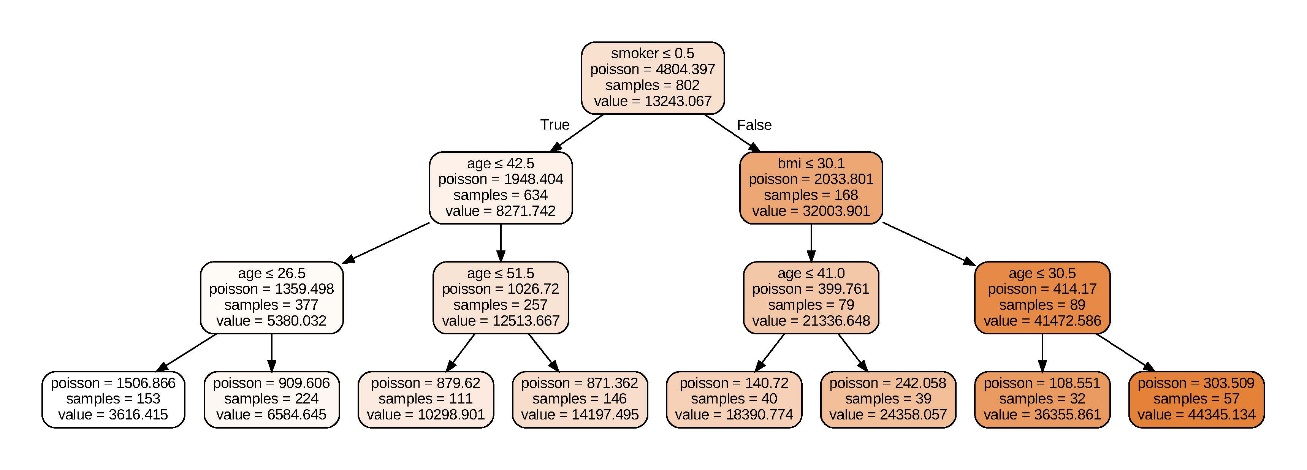
**DTC**



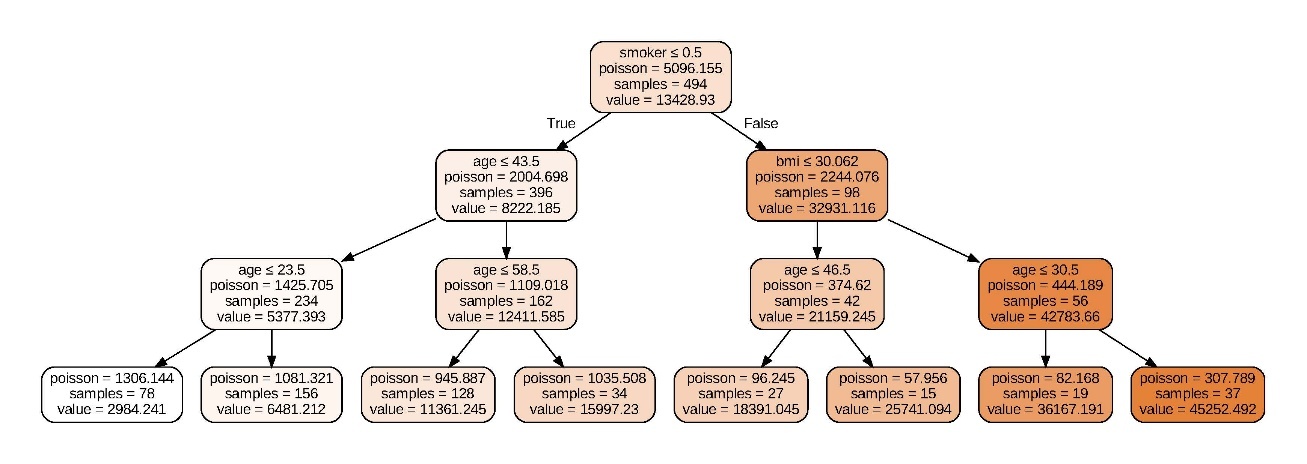
**RFC**



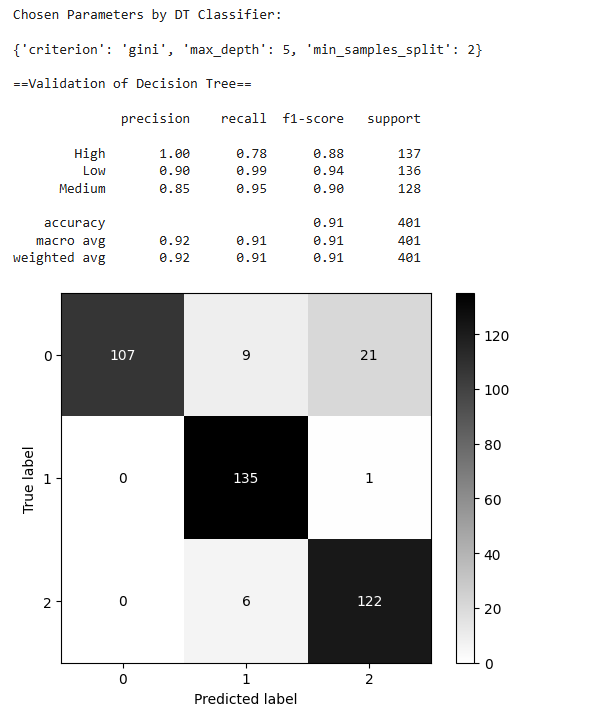
**DTC – Voting Regressor**

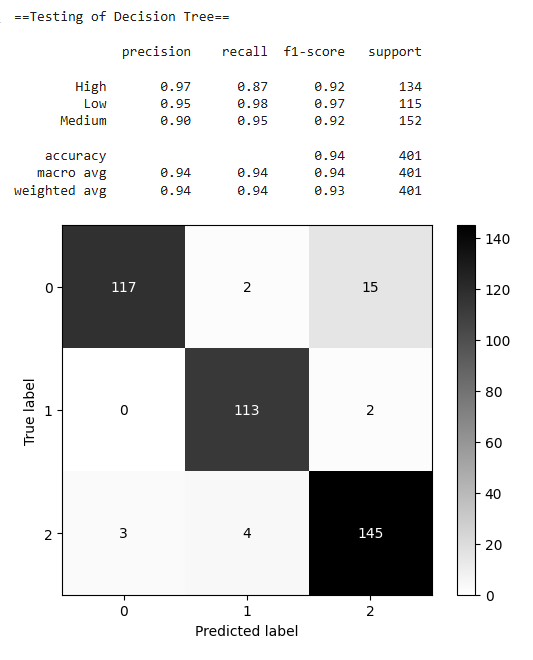
****

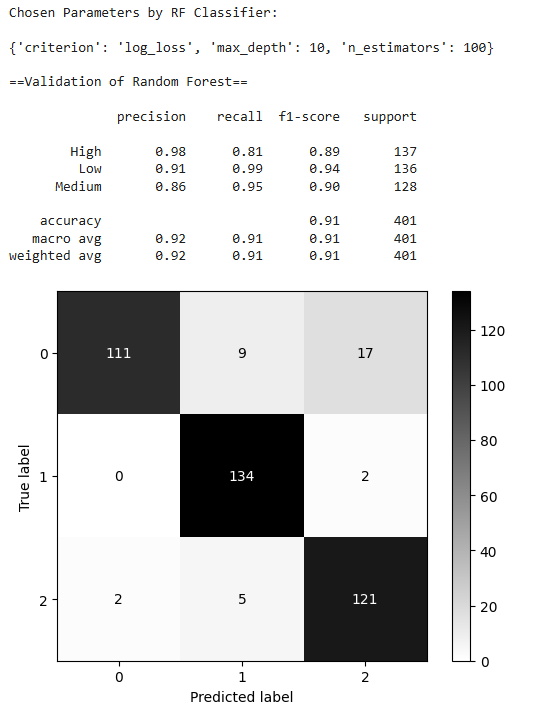
**RFC – Voting Regressor**

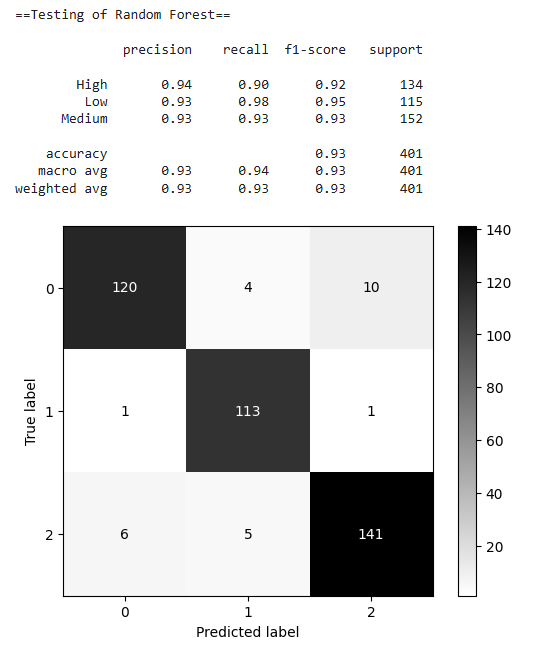
****

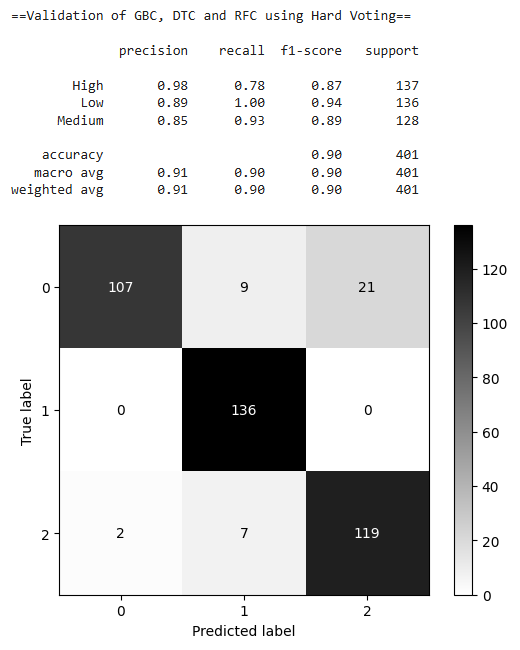
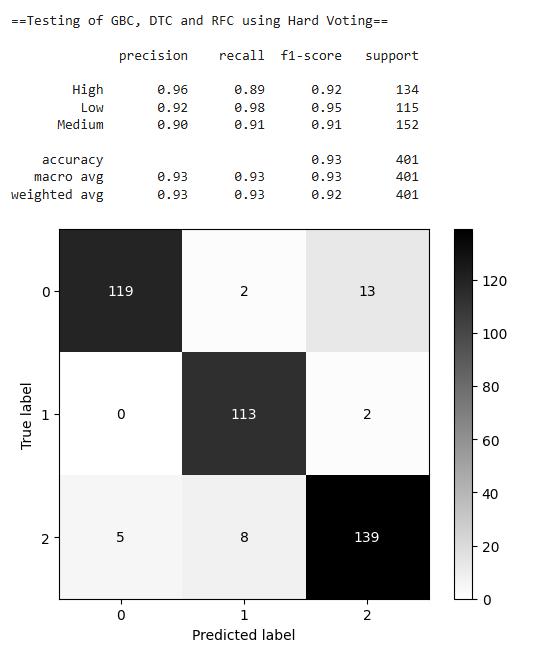
## Confusion Matrices with chosen Parameters and Evaluation Metrics

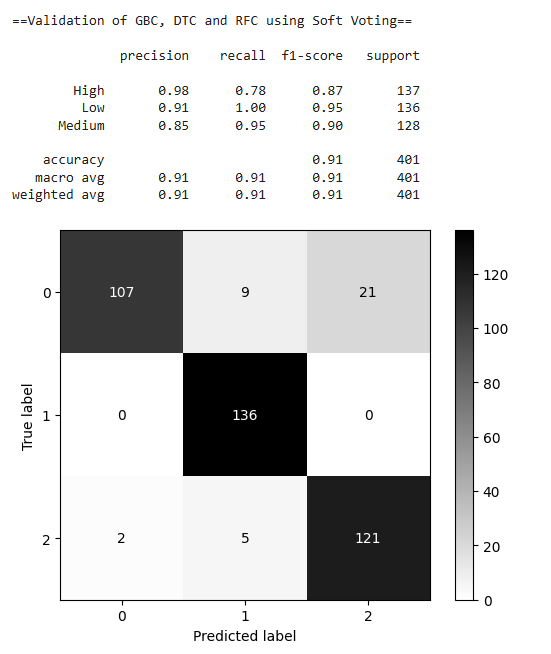
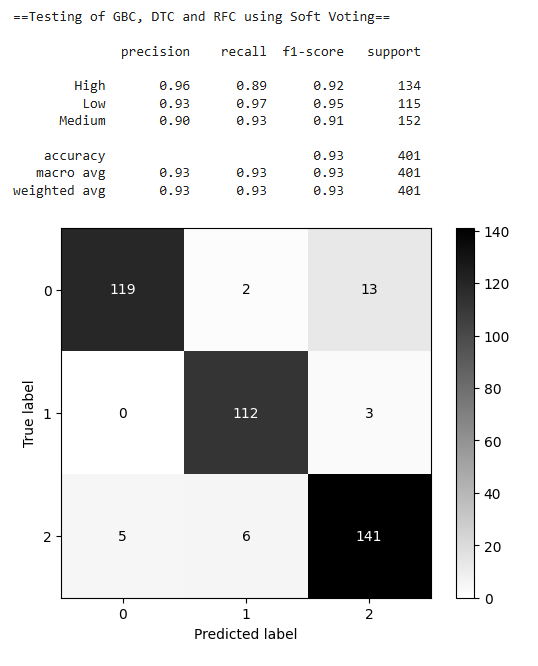








## ROC Curves

