Heart Attack Prediction Report

**Introduction**

The project is developed to predict whether a person with said indicators will have a heart attack or not, in this dataset we take note of large amounts of data with various health indicator fields which we will discuss in depth further in the report. Although I have used the dataset to predict heart attack, the same dataset contains other diseases data too, we can remodel the entire code to predict them too because of the correlation of many health indicators in a human body.

**Dataset Analysis**

The dataset chosen for this project is sourced from Kaggle and is called indicator of heart disease 2022. It contains 2022 annual CDC survey data of 400k+ adults in all over united states, It was given a usability score of 9.41 by Kaggle which was assessed on basis of reliability of source, completeness of data and update frequency.



Fig. 1 The Dataset Description.

The dataset contains 246,022 entries and 40 columns which represent all the health and demographic data of people being studied by CDC survey. The target variable or the variable we will be predicting in this study is the column “HadHeartAttack”.

The dataset contains both categorical features (e.g. Sex, GeneralHealth) and numerical features (e.g. BMI). The target variable was converted from categorical (Yes or no) to numerical for this project (1 for yes and 0 for no) using label encoding.

The health-related Indicators in the dataset include but not limited to:

* DifficultyWalking
* HadAngina
* HadStroke
* And etc.

The dataset was imbalanced because most samples were labelled “No” for the target variable so this was accounted for in preprocessing.

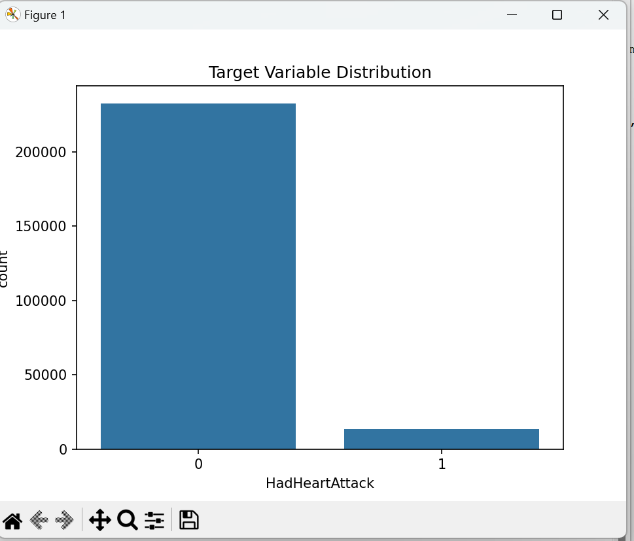
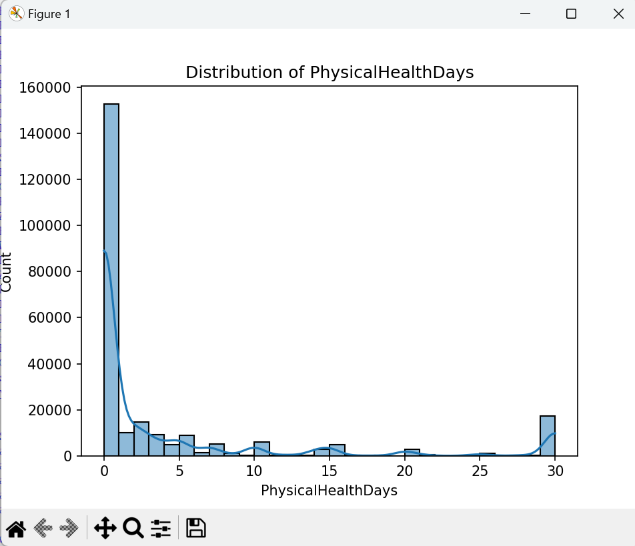
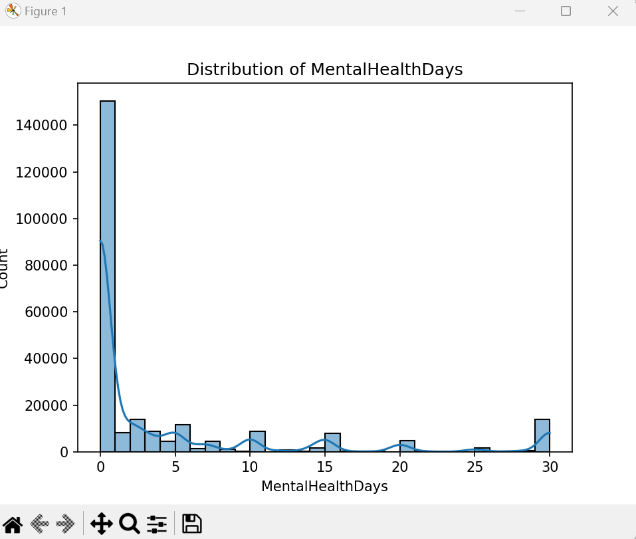
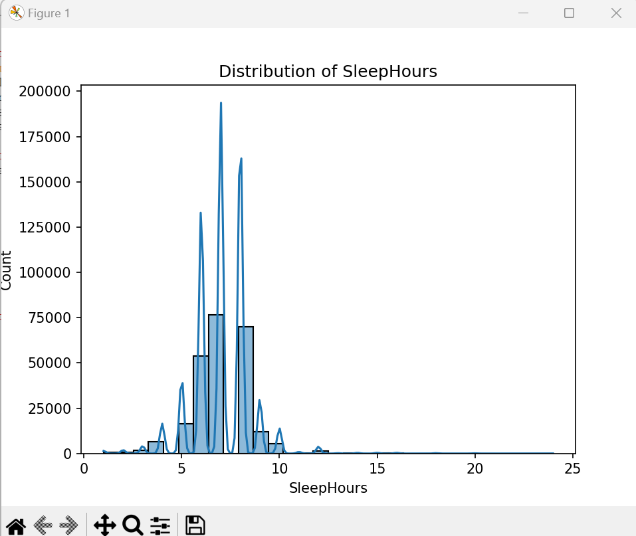


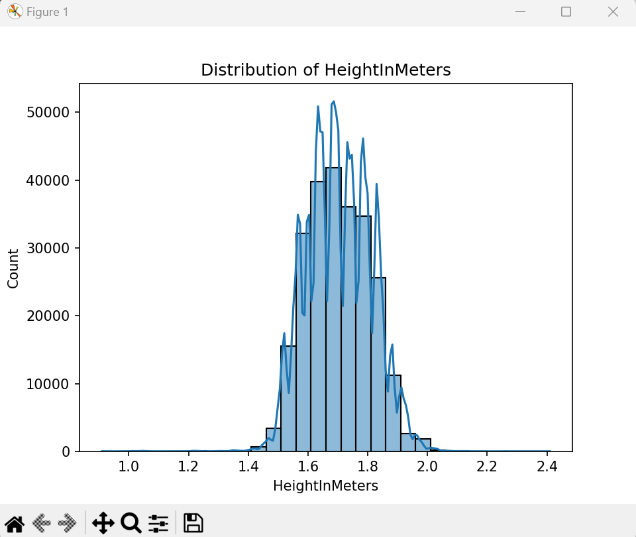
Fig. 2 The distribution of target var.

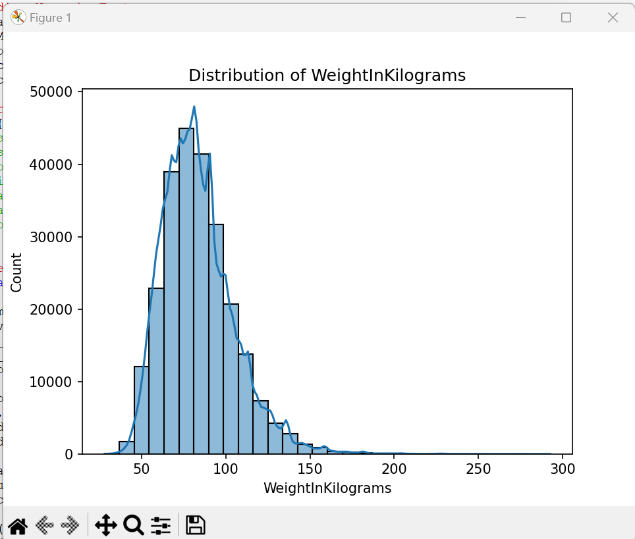
The distribution of other numeric features was also studied and taken into account while preprocessing. The features like physical health days, mental health days, sleep hours, height, weight and bmi were studied.











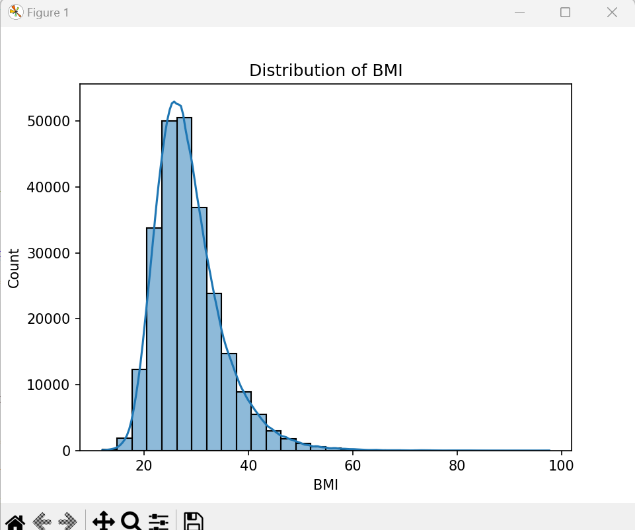


Fig. 3 Collection of feature graphs.

The correlation of the features with the target variable was studied to establish the relevancy of each feature of the dataset being used for model use and prediction. The largest correlating factor for the target variable was Angina diagnosis which gave us a value of 0.44 and the lowest correlating factor with the target variable was Sleep hours with only 0.003 correlation. Still I didn’t remove this feature because of the medical relevancy and sometimes in complex relations the correlation scaling can be inaccurate.

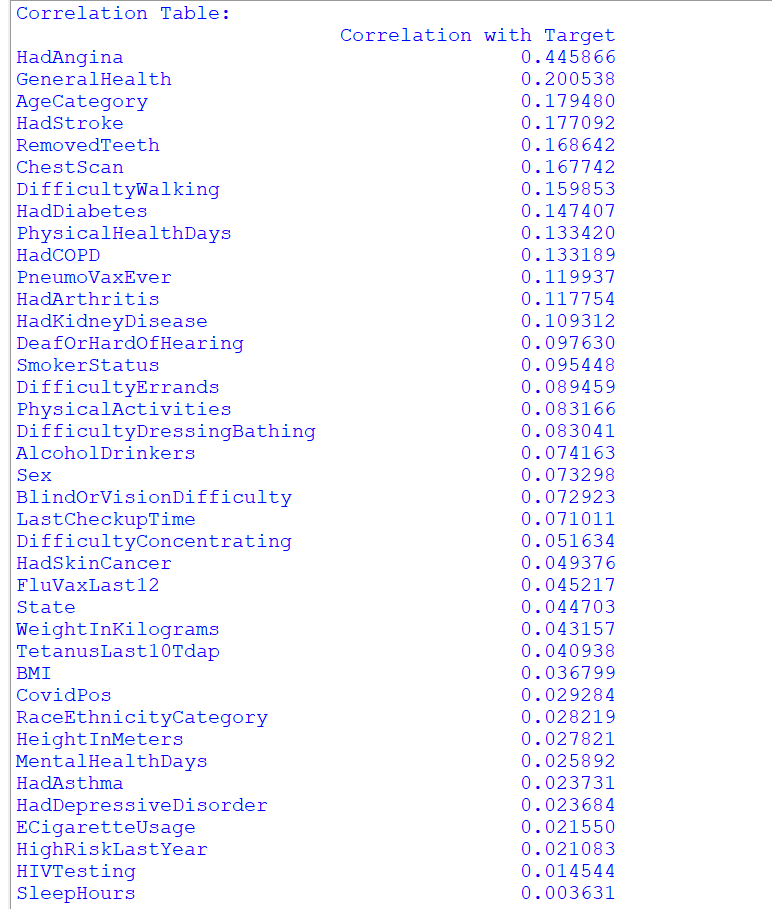


Fig. 4 Correlation table for dataset.

**Dataset Preprocessing**

To make the dataset ready for running machine learning models, several preprocessing steps were applied:

1. **Encoding of Categorical Features**:
   * Label encoding was used to convert string values in categorical columns into numerical representations like in the target variable.
2. **Feature Scaling**:
   * Numerical features were scaled using **MinMaxScaler** to normalize values between 0 and 1. This helps models like logistic regression and support vector machines converge faster and improves performance.
3. **Correlation Analysis**:
   * Spearman and Pearson correlation coefficients were computed for numerical features to evaluate their relationship with the target variable.
   * Categorical features were analyzed using **Cramér’s V**, which measures association strength for categorical variables.

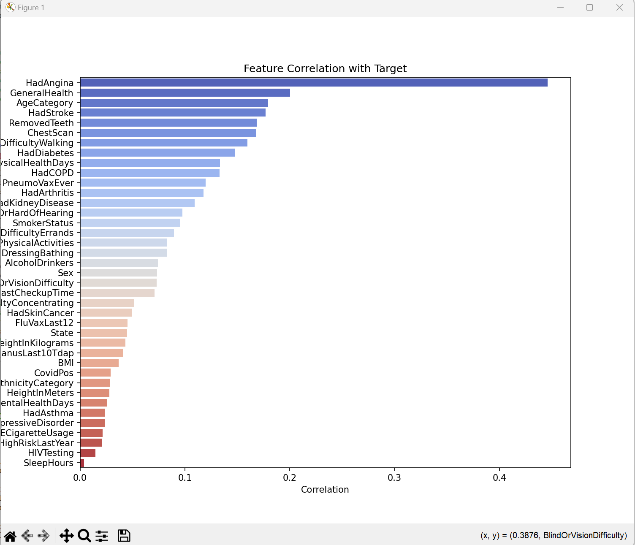


Fig. 5 Correlation Graph for dataset.

1. **Train-Test Split**:
   * The dataset was split into training (70%) and testing (30%) subsets, ensuring the evaluation metrics are unbiased.
2. **Imbalance Handling**:
   * While no resampling was done in this experiment, metrics like **ROC-AUC** and **precision-recall** were used to account for the class imbalance.

**Machine Learning Algorithms**

The Algorithms used by me is as follows:

1. **Logistic Regression**: A simple and interpretable linear model that predicts probabilities using a logistic function. In this model I have set the penalty to 12 to reduce overfitting and increased maximum iterations to ensure proper convergence.

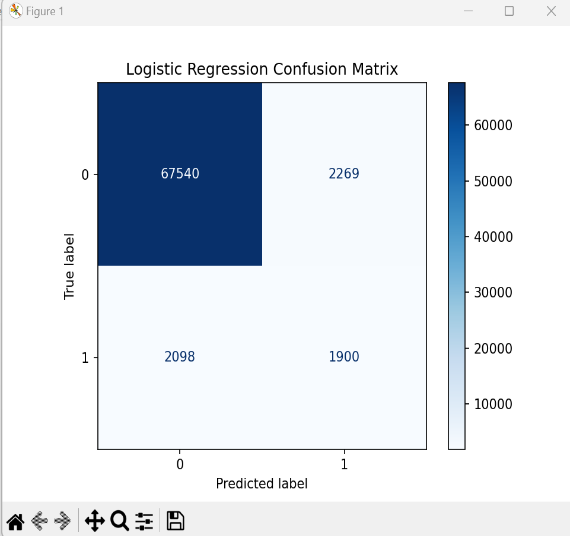


Fig 6. Confusion matrix of LR.

1. **Decision Tree Classifier**: A non-linear model based on tree structures, capable of capturing complex interactions. In this model I have limited the depth to 10 and minimum sample required for a node split to 50 for ensuring that overfitting is avoided and small unstable splits which result in poor generalization are avoided.

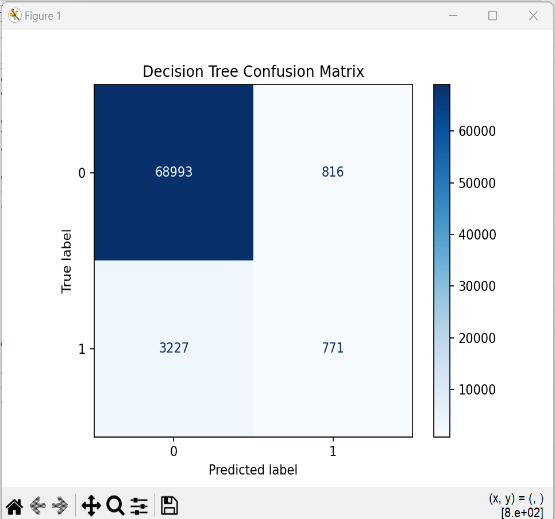


Fig. 7 Confusion matrix of DT.

1. **Random Forest Classifier**: A model using multiple decision trees for robust predictions. I set the number of trees to 100 and limited their depth to 15 while using balanced class weights so that the stability and accuracy is maintained and balanced weight class improves the minority class recall.

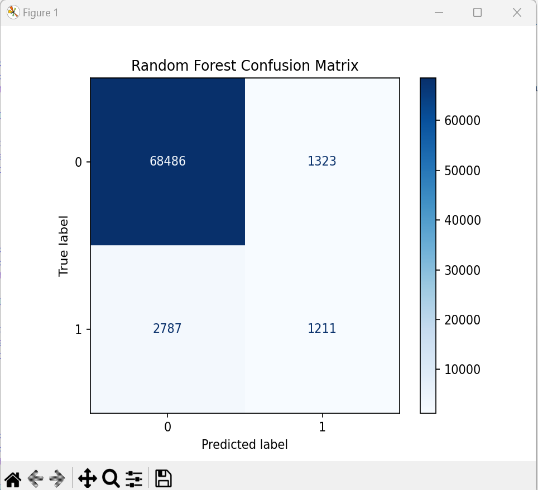


Fig. 8 Confusion matrix of RF.

1. **Gradient Boosting Classifier**: Model that builds trees sequentially to minimize errors. I set the number of boosting stages to 150 and reduced the learning rate to 0.05 while limiting the maximum depth to 5 for achieving results which are not overfitting and influenced by noise.

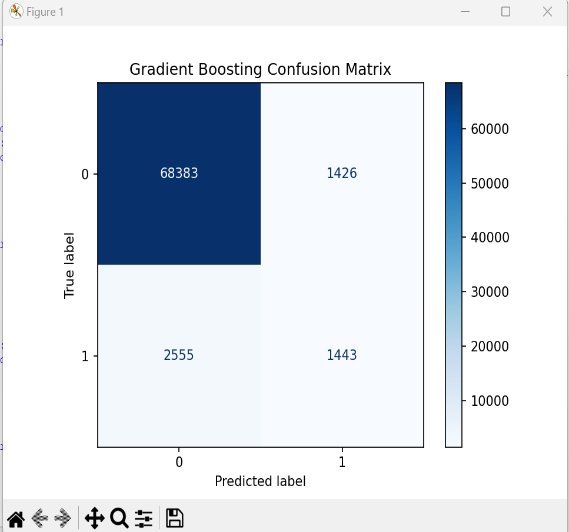


Fig. 9 Confusion matrix of GB.

1. **Neural Network (MLP)**: A deep learning approach using layers of neurons for predictions. There were two hidden layers put in the neural network with 64 and 32 neurons, dropout was set to 0.3 to avoid overfitting. The optimizer chosen was Adam and learning rate was 0.001. I also set the batch size to 256, epoch to 50 and maximum iterations to 300 to reduce computational cost and making execution faster without compromising accuracy.

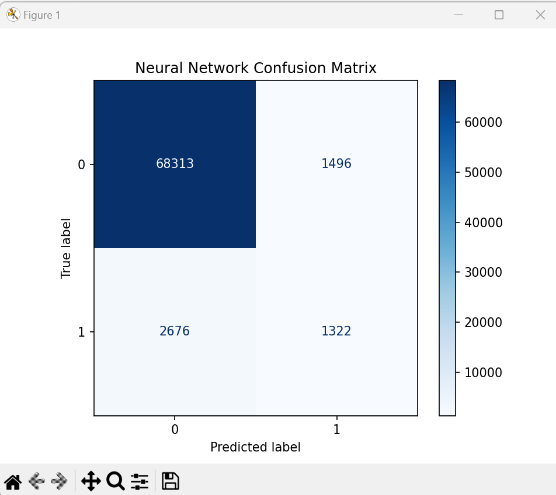


Fig. 10 Confusion matrix NN.

1. **K-Nearest Neighbors (KNN)**: A distance-based algorithm. The number of neighbours I used was 5 which is neither too less nor too many like 20. The features were scaled using min max scaler. I did this to capture local patterns in the data and improving accuracy.

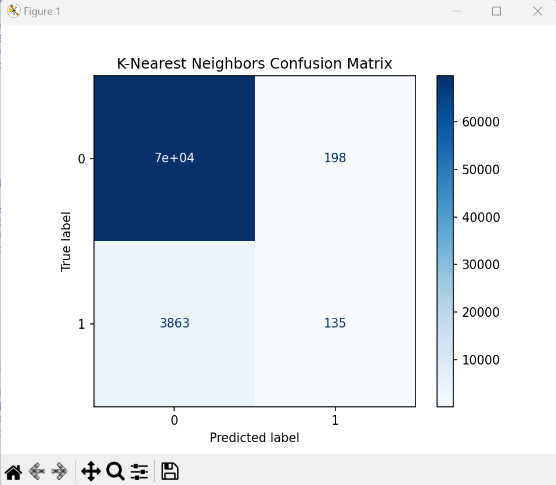


Fig. 11 Confusion matrix KNN.

1. **Support Vector Machine (SVM)**: A classification model that finds a hyperplane to separate classes, using a kernel trick for non-linear separations. The kernel I used was RBF, regularization parameter or c was set to 1.0 and maximum iterations were limited to 50 to avoid large computational time and overfitting.

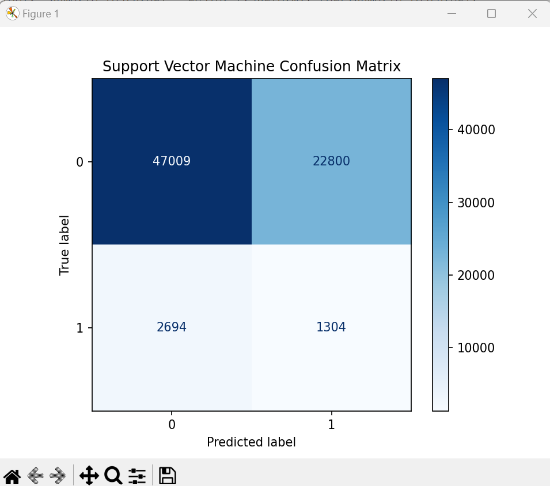


Fig. 12 Confusion matrix SVM.

**Evaluation Metrics**

The parameters I used to evaluate the dataset keeping in mind all the imbalances and models:

* **Accuracy**: Calculates the percentage of correct predictions. Symbolizes the overall model performance. Almost all the models averaged around 94%+ accuracy.
* **ROC-AUC**: Evaluates the model's ability to distinguish between classes. A high AUC indicates a good balance between sensitivity and specificity. Almost all the models got 85%+ AUC.
* **Classification Report**:
* **Precision**: Portion of correctly predicted positive instances to all predicted positives.
* **Recall**: Portion of correctly predicted positives to all actual positives.
* **F1-Score**: Harmonic mean of precision and recall, balancing the false positives and negatives.
* **Confusion Matrix**: Provides data of true positives, true negatives, false positives, and false negatives.

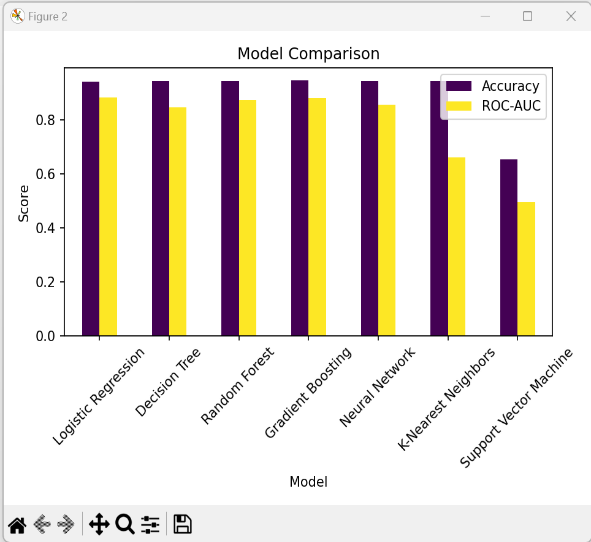
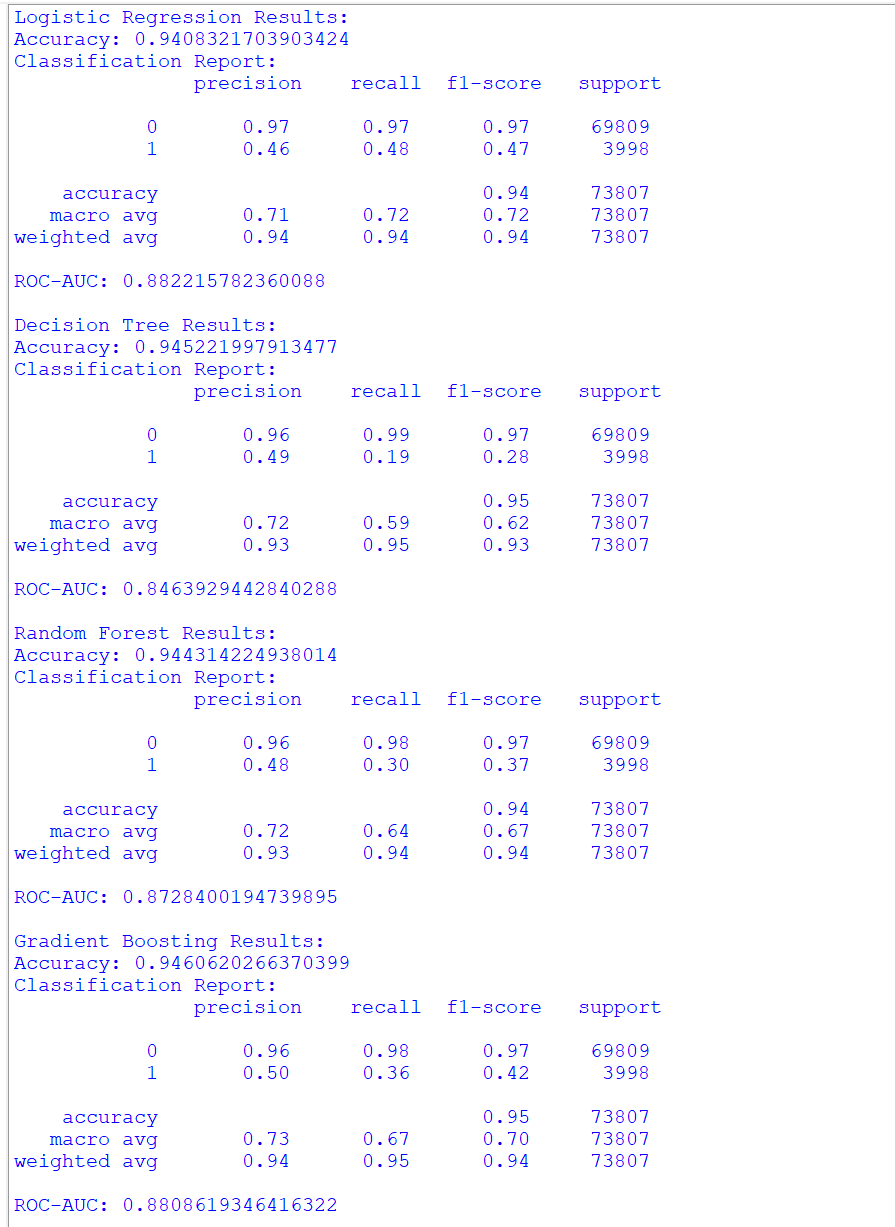


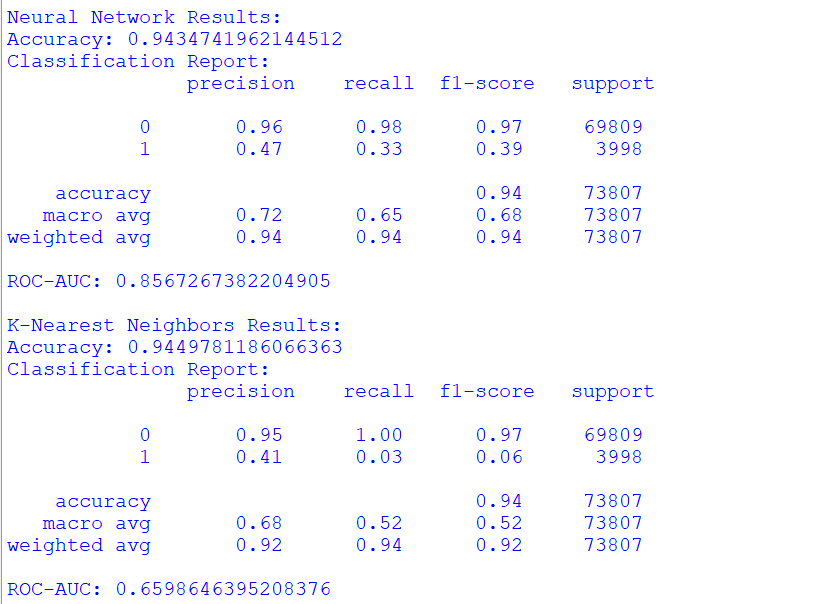
Fig. 13 Accuracy and AUC of all.

**RESULTS**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| | **Model** | **Accuracy** | **ROC-AUC** | **Precision (1)** | **Recall (1)** | **F1-Score (1)** | | --- | --- | --- | --- | --- | --- | |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Logistic Regression | 94.08% | 0.88 | 0.46 | 0.48 | 0.47 | |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Decision Tree | 94.52% | 0.85 | 0.49 | 0.19 | 0.28 | |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Random Forest | 94.43% | 0.87 | 0.48 | 0.30 | 0.37 | |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Gradient Boosting | 94.60% | 0.88 | 0.50 | 0.36 | 0.42 | |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Neural Network | 94.34% | 0.85 | 0.47 | 0.33 | 0.39 | |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | K-Nearest Neighbour | 94.50% | 0.66 | 0.41 | 0.03 | 0.06 | |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Support Vector Machine | 65.45% | 0.49 | 0.05 | 0.33 | 0.09 | |

The models were optimized to balance accuracy and generalization. Logistic Regression performed well overall (Accuracy: **94.08%, ROC-AUC: 0.88**) due to regularization. Decision Tree and Random Forest, with depth caps, provided good results (Accuracy: **94.52% and 94.43%**, ROC-AUC: 0.85 and 0.87). Gradient Boosting achieved the best recall for minority classes with great learning parameters (Accuracy: **94.61%, ROC-AUC: 0.88**). Neural Networks efficiently handled non-linear relationships while not taking too much computational time (Accuracy: **94.35%, ROC-AUC: 0.86**). KNN and SVM, while less accurate for the imbalanced dataset, demonstrated the importance of scaling and kernel selection (Accuracy: **94.49% and 65.45%**, ROC-AUC: 0.66 and 0.50). Overall, these methods and boosting provided the best balance of all precision, recall and ROC-AUC.





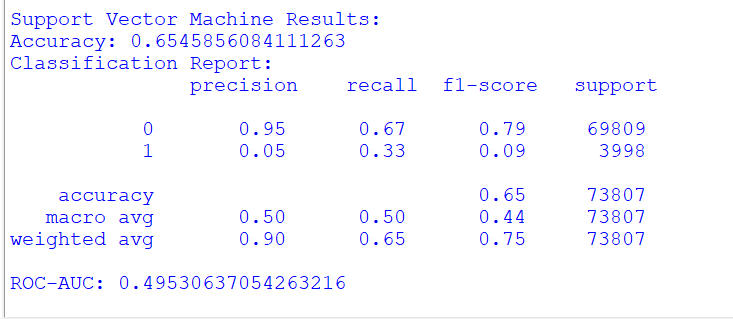


Fig 14. Terminal Display of results.

**References**

1. <https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease>

**Made By:**

Kartekay Kaushik

KXK230116

MS COMPUTER ENGINEERING