**Name: Sanika Nandkishor Shirode**

**STATISTICAL LEARNING AND DATA MINING (PROJECT)**

**TOPIC :** **PREDICTING DEATH RISK RELATED TO HEART FAILURE USING STATISTICAL AND MACHINE LEARNING MODELS.**

**INTRODUCTION TO THE PROBLEM:**

Heart failure is a serious global health issue with high mortality rates. Early prediction of survival outcomes is crucial for effective patient care. In this project, we made a thorough analysis of dataset of 299 heart failure patients with 13 clinical features, including age, ejection fraction, serum creatinine, and comorbidities like diabetes and hypertension, etc.

The objective is to develop and evaluate machine learning models such as Logistic Regression, Decision Trees, SVM, and ensemble methods to classify patient survival (death event). Advanced techniques like bootstrap sampling and k-fold cross-validation are used for robust analysis.

The project aims to identify key predictors of heart failure outcomes, compare model performance, and provide insights to support better clinical decisions.

**DATA DESCRIPTION :**

The dataset consists of medical records from 299 heart failure patients, including 12 clinical features to predict patient survival (death event) and analyse the relationships between clinical features and outcomes :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Variable Name** | **Role** | **Type** | **Demographic** | **Description** | **Units** |
| age | Feature | Integer | Age | age of the patient | years |
| anaemia | Feature | Binary |  | decrease of red blood cells or hemoglobin |  |
| creatinine\_phosphokinase | Feature | Integer |  | level of the CPK enzyme in the blood | mcg/L |
| diabetes | Feature | Binary |  | if the patient has diabetes |  |
| ejection\_fraction | Feature | Integer |  | percentage of blood leaving the heart at each contraction | % |
| high\_blood\_pressure | Feature | Binary |  | if the patient has hypertension |  |
| platelets | Feature | Continuous |  | platelets in the blood | kiloplatelets/mL |
| serum\_creatinine | Feature | Continuous |  | level of serum creatinine in the blood | mg/dL |
| serum\_sodium | Feature | Integer |  | level of serum sodium in the blood | mEq/L |
| sex | Feature | Binary | Sex | woman or man |  |
| smoking | Feature | Binary |  | if the patient smokes or not |  |
| time | Feature | Integer |  | follow-up period | days |
| death\_event | Target | Binary |  | if the patient died during the follow-up period |  |

**EVALUATION METRICS:**

Since, we are in the setup of binary classification problem, where the target variable (DEATH\_EVENT) can take on only two values (e.g. 0-Non Death & 1-Death) evaluating the performance of a model is crucial. Several evaluation metrics are commonly used to assess a model's accuracy and reliability. These metrics provide insights into how well the model predicts the correct class and how well it balances false positives and false negatives.

This section will delve into the key evaluation metrics used in binary classification.

**Confusion Matrix:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predicted Class | True Class | | | |
|  | NON-DEATH(-) | DEATH(+) | Total |
| NON-DEATH(-) | TN | FN | TN+FN |
| DEATH(+) | FP | TP | FP+TP |
| Total | TN+FP | FN+TP | Grand Total |

Where:

**TP** (True Positives): Correctly predicted positive (Death) instances.

**TN** (True Negatives): Correctly predicted negative (Non-Death) instances.

**FP** (False Positives): Incorrectly predicted positive (Death) instances.

**FN** (False Negatives): Incorrectly predicted negative (Non-Death) instances.

The Various Evaluation metrics are listed as follows:

* **Accuracy** : Measures the proportion of correct predictions out of the total number of predictions.

=

* **Precision** : Measures the proportion of positive predictions that are actually positive.

=

* **Recall** : Measures the proportion of actual positive cases that are correctly identified.

=

* **F1-Score** : The harmonic mean of precision and recall.

=

* **ROC AUC Score** : Measures the ability of a model to distinguish between positive and negativeclasses.

These evaluation metrics were calculated on both the training and testing sets, which gives the better approximation of the model’s overall performance.

**MULTIPLE LOGISTIC REGRESSION:**

**MLR using Validation Set Approach:**

Here, We developed and evaluated a logistic regression model to predict death risk using patient data. The data was prepared for analysis following a validation set approach, splitting the data into training and testing sets. By utilising this approach, Split is made on the entire dataset with 70% data for the training and remaining 30% for the testing.

**Data Preprocessing:**

* **Feature Selection:** The first 12 columns from the data were used as features, including both numerical and categorical features.
* **Numerical Feature Scaling:** Numerical features (age, creatinine\_phosphokinase, ejection\_fraction, platelets, serum\_creatinine, serum\_sodium, time) were standardized using a StandardScaler().

**Model Training and Evaluation:**

* **Logistic Regression Model:** A logistic regression model was created and trained on the prepared training data.
* **Evaluation Metrics:** Several evaluation metrics were calculated on both the training and testing sets.

**Results**

The model achieved an accuracy of 88.04% on the training set and 78.89% on the testing set. Here's a detailed breakdown of the results:

**Confusion Matrix for Training Set:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predicted Class | True Class | | | |
|  | NON-DEATH(-) | DEATH(+) | Total |
| NON-DEATH(-) | 140 | 15 | 155 |
| DEATH(+) | 10 | 44 | 54 |
| Total | 150 | 59 | 209 |

**Confusion Matrix for Testing Set:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predicted Class | True Class | | | |
|  | NON-DEATH(-) | DEATH(+) | Total |
| NON-DEATH(-) | 50 | 16 | 66 |
| DEATH(+) | 3 | 21 | 24 |
| Total | 53 | 37 | 90 |

**Evaluation Metrics:**

|  |  |  |
| --- | --- | --- |
| Metric | Training Set | Testing Set |
| Accuracy | 0.8804 | 0.7889 |
| Precision | 0.8148 | 0.875 |
| Recall | 0.7458 | 0.5676 |
| F1-Score | 0.7788 | 0.6885 |
| ROC AUC Score | 0.9242 | 0.8343 |

1. Accuracy:

* Training Set: 88.04% indicates the model performs well on the training data.
* Testing Set: 78.89% shows a decrease, suggesting the model may slightly overfit the training data.

2. Precision:

* Precision on the testing set (87.5%) is higher than on the training set (81.48%), meaning the model performs well in correctly identifying patients who experienced a death event.

3. Recall (Sensitivity):

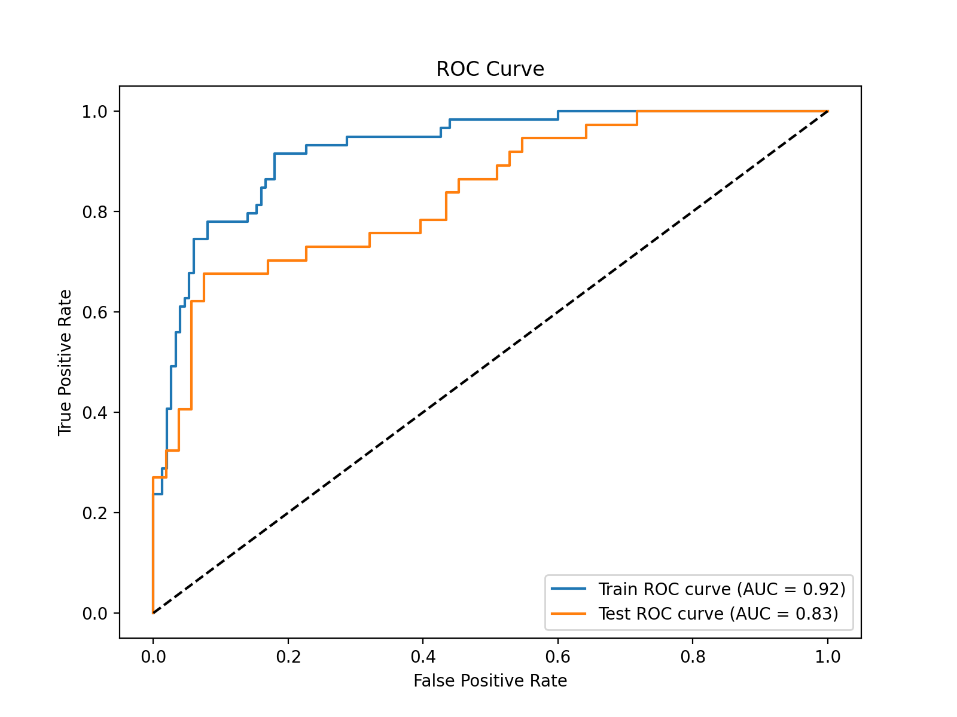
* Training Set: 74.58% indicates the model captures a good portion of actual death events.
* Testing Set: 56.76% is lower, suggesting the model misses some actual death events when tested on unseen data.

4. F1-Score:

* Training Set: 77.88% combines precision and recall, showing a balanced performance.
* Testing Set: 68.85% indicates reduced overall performance, particularly due to the drop in recall.

5. ROC AUC Score:

* Training Set: 92.42% reflects excellent discriminatory power on training data.
* Testing Set: 83.43% shows good performance but highlights a slight decline in generalization.

****

**Interpretation**

The logistic regression model achieved moderate performance. The model's accuracy was higher on the training set compared to the testing set, indicating some overfitting. The ROC AUC scores suggest an acceptable ability to discriminate between patients with and without heart failure death risk.

**Further analysis of MULTIPLE LOGISTIC REGRESSION**:

To assess the significance of individual features in predicting heart failure death risk using a logistic regression model, we employed the following procedure:

1. **Model Training:**
   * A logistic regression model was trained on the prepared dataset.
   * The statsmodels() library was used to fit the model and obtain coefficient estimates and p-values.
2. **Feature Significance Analysis:**
   * The p-values associated with each feature's coefficient were examined.
   * A p-value less than 0.05 is typically considered statistically significant, indicating that the feature has a significant impact on the outcome (DEATH\_EVENT).

**Results and Interpretation:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Coefficient** | **p-value** |
| age | 0.753444 | 0.0029 |
| anaemia | -0.388538 | 0.4366 |
| creatinine\_phosphokinase | 0.078308 | 0.7105 |
| diabetes | 0.154849 | 0.7418 |
| ejection\_fraction | -0.981694 | 0.0002 |
| high\_blood\_pressure | -0.273453 | 0.566 |
| platelets | -0.239042 | 0.3217 |
| serum\_creatinine | 0.897255 | 0.0001 |
| serum\_sodium | -0.296369 | 0.2115 |
| sex | -1.202001 | 0.0276 |
| smoking | 0.487026 | 0.4042 |
| time | -1.902261 | 0.0001 |

Based on the analysis, the following features were found to be statistically significant at the 5% significance level:

* **Age:** Older age is associated with a higher risk of death due to heart failure.
* **Ejection Fraction:** Lower ejection fraction indicates reduced heart pumping efficiency and is associated with higher risk.
* **Serum Creatinine:** Higher serum creatinine levels suggest impaired kidney function and are associated with increased risk.
* **Sex:** Female sex may be associated with a lower risk of heart failure.
* **Time:** Longer follow-up time might be associated with a higher risk of heart failure event.

Features with p-values greater than 0.05 were considered not statistically significant and may not contribute significantly to the model's predictive power. These features include:

* Anaemia
* Creatinine Phosphokinase
* Diabetes
* High Blood Pressure
* Platelets
* Serum Sodium
* Smoking

**Conclusion:**

The logistic regression model identified several significant features that contribute to heart failure prediction. Age, ejection fraction, serum creatinine, sex, and time were found to be statistically significant predictors. These findings can be used to identify high-risk individuals and inform clinical decision-making.

**LOGISTIC REGRESSION After Eliminating Insignificant Regressors**

Here, we have developed and evaluated a logistic regression model to predict heart failure death risk, focusing on features identified as statistically significant in a previous analysis.

**Data Preprocessing:**

* **Feature Selection:** Based on previous analysis, only features with significant p-values were selected: age, ejection\_fraction, serum\_creatinine, and sex.
* **Numerical Feature Scaling:** Numerical features (age, ejection\_fraction, serum\_creatinine) were standardized using a StandardScaler.

**Model Training and Evaluation:**

* **Logistic Regression Model:** A logistic regression model was created and trained on the prepared training data.
* **Evaluation Metrics:** Several evaluation metrics were calculated on both the training and testing sets, given as follows:

**Confusion Matrix for Training Set:**

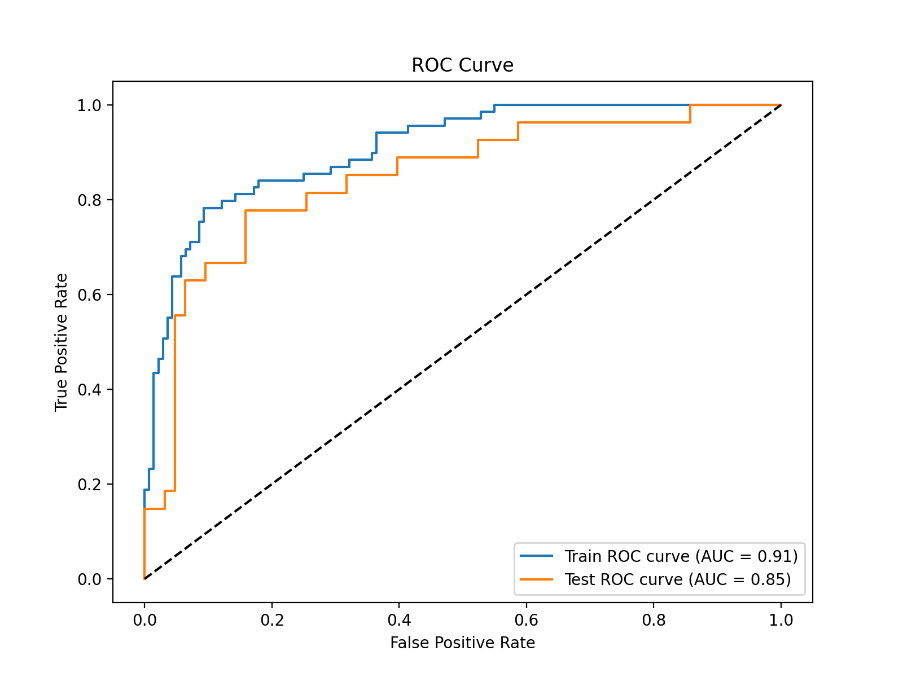
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predicted Class | True Class | | | |
|  | NON-DEATH(-) | DEATH(+) | Total |
| NON-DEATH(-) | 128 | 19 | 147 |
| DEATH(+) | 12 | 50 | 62 |
| Total | 140 | 69 | 209 |

**Confusion Matrix for Testing Set:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predicted Class | True Class | | | |
|  | NON-DEATH(-) | DEATH(+) | Total |
| NON-DEATH(-) | 57 | 9 | 66 |
| DEATH(+) | 6 | 18 | 24 |
| Total | 63 | 27 | 90 |

**Evaluation Metrics:**

|  |  |  |
| --- | --- | --- |
| Metric | Training Set | Testing Set |
| Accuracy | 0.8517 | 0.8333 |
| Precision | 0.8065 | 0.75 |
| Recall | 0.7246 | 0.6667 |
| F1-Score | 0.7634 | 0.7059 |
| ROC AUC Score | 0.9079 | 0.8466 |



**Interpretation:**

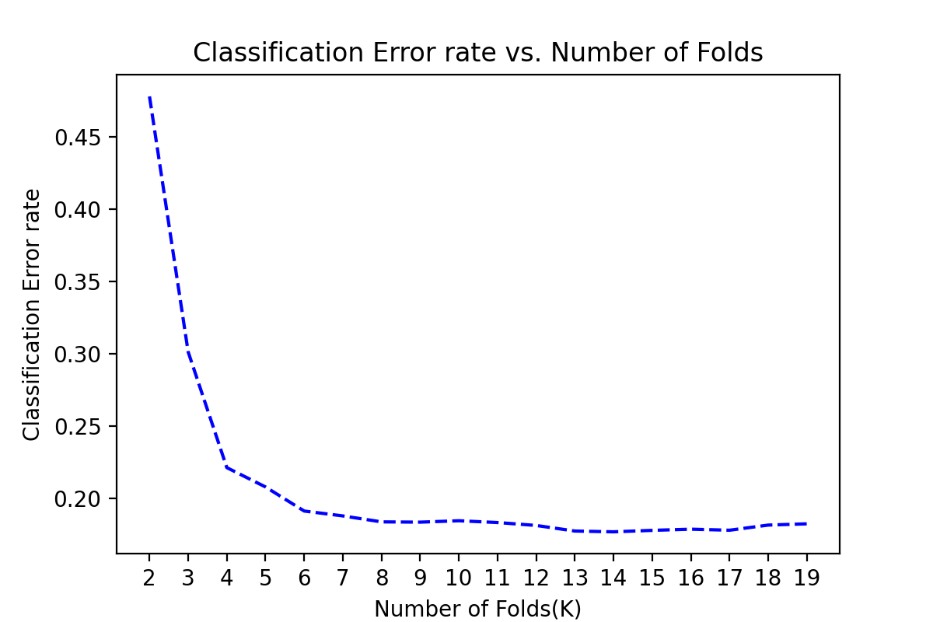
The logistic regression model with a reduced feature set achieved moderate performance in predicting heart failure death risk. The model's accuracy was similar on both the training and testing sets, suggesting reduced overfitting compared to the previous model with all features. The ROC AUC scores indicate an acceptable ability to discriminate between patients with and without heart failure. However, the performance metrics on the testing set are slightly lower than those on the training set, suggesting some potential for improvement in generalizability.

**MULTIPLE LOGISTIC REGRESSION (With k-fold cross validation):**

In this section, we evaluate the performance of the logistic regression model using K-Fold Cross-Validation (KFold-CV). KFold-CV is a technique for assessing model generalizability by repeatedly splitting the data into training and testing sets.

**Methodology**

1. **K-Fold Loop:** We iterates through K values from 2 to 19.
2. **Inner Loop (Cross-Validation):**
   * For each K value, a logistic regression model is created.
   * The data is split into K folds using stratified KFold-CV.
   * In each fold:
     + The model is trained on the remaining K-1 folds (training set).
     + The model is evaluated on the held-out fold (testing set).
   * The accuracy score is calculated for each fold.
3. **Error Rate Calculation:** The average accuracy scores across the folds are calculated for each K value. The error rate is computed as .
4. **Plotting the Results:** A line graph is created to visualize the classification error rate versus the number of folds (K).



**Results**

The KFold-CV results indicate that the model's performance generally improves with an increasing number of folds, up to a certain point. Here are some key observations:

* The error rate decreases as the number of folds increases from 2 to 12. This suggests that the model benefits from more data for training in each fold, leading to potentially better generalization.
* The error rate stabilizes or fluctuates slightly after 10 folds. This suggests that using a higher number of folds might not significantly improve performance and could increase computational cost.

**Overall, the KFold-CV analysis suggests that the logistic regression model achieves moderate performance with acceptable generalizability. Thus, we choose the value of K=10 for the further analysis.**

**MULTIPLE LOGISTIC REGRESSION with Leave-One-Out Cross-Validation (LOOCV)**

In addition to K-Fold Cross-Validation, we employed Leave-One-Out Cross-Validation (LOOCV) for a more rigorous evaluation of the logistic regression model's performance.

**LOOCV** is a specific type of cross-validation where each data point is used as a validation set once, and the remaining data points are used as the training set. This approach is computationally intensive, especially for large datasets, but it provides a robust estimate of the model's generalization performance.

**Results**

The LOOCV accuracy of the logistic regression model is 0.8294, corresponding to an error rate of 0.1706. This result indicates that the model performs reasonably well in predicting heart failure risk, with an error rate of approximately 17%.

**Comparing K-Fold CV and LOOCV**

While K-Fold CV is generally more computationally efficient, LOOCV provides a more accurate estimate of the model's generalization performance, especially for smaller datasets like ours. In this case, the performance of the model is consistent across both methods, suggesting that the model's generalization ability is robust.

**Conclusion**

The combination of K-Fold CV and LOOCV provides strong evidence for the logistic regression model's ability to accurately predict heart failure risk based on the selected features.

**Classification with DECISION TREE**

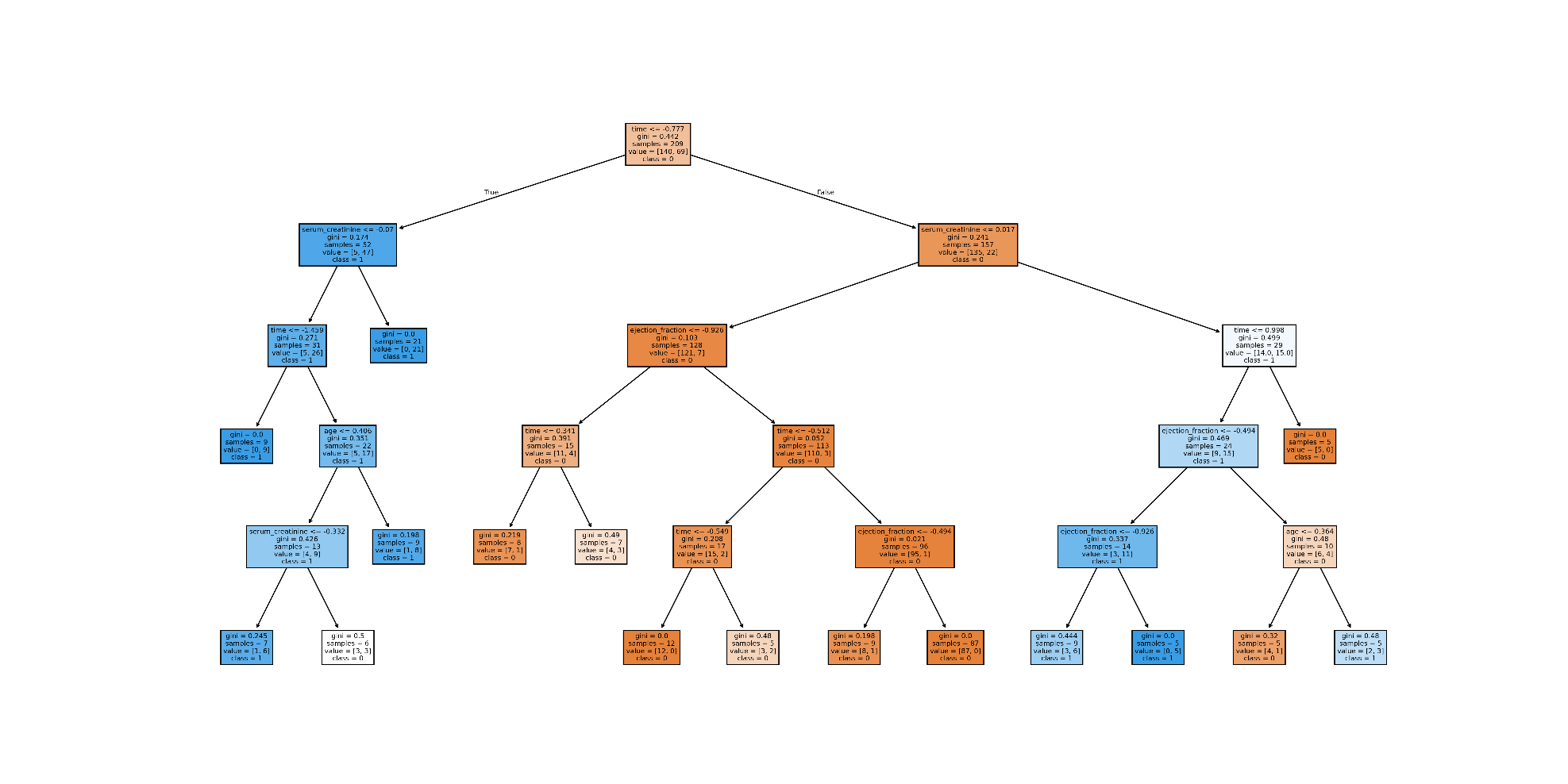
In this section, we explore the performance of a Decision Tree Classifier model (unpruned) for predicting heart failure death risk. An unpruned decision tree, as the name suggests, is a decision tree that has not been subjected to any pruning techniques. This means that the tree continues to grow until each leaf node contains instances of a single class or until a stopping criterion, such as a minimum number of samples per leaf (In our case, it is 5), is reached. Once a decision tree model is trained, it can be used to make predictions on new, unseen data. Here's how the prediction process works:

1. **Start at the Root Node:** Begin at the root node of the tree.
2. **Evaluate Feature:** Check the value of the feature specified at the current node.
3. **Follow the Branch:** Based on the feature value, follow the appropriate branch (left or right) to the next node.
4. **Repeat Steps 2 and 3:** Continue this process until you reach a leaf node.
5. **Make the Prediction:** The class label associated with the leaf node is the predicted class for the input instance.

**Model Training and Evaluation**

* A Decision Tree Classifier was created with a minimum samples per leaf of 5.
* The model was trained on the **training data (70% of the dataset)**.
* Confusion matrices were generated for both the training and testing sets.
* The model’s performance was evaluated on both the training and testing sets.

**Decision Tree - Unpruned**



**Confusion Matrix for Training Set:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predicted Class | True Class | | | |
|  | NON-DEATH(-) | DEATH(+) | Total |
| NON-DEATH(-) | 143 | 10 | 153 |
| DEATH(+) | 7 | 49 | 56 |
| Total | 150 | 59 | 209 |

**Confusion Matrix for Testing Set:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predicted Class | True Class | | | |
|  | NON-DEATH(-) | DEATH(+) | Total |
| NON-DEATH(-) | 47 | 10 | 57 |
| DEATH(+) | 6 | 27 | 33 |
| Total | 53 | 37 | 90 |

**Evaluation Metrics:**

|  |  |  |
| --- | --- | --- |
| Metric | Training Set | Testing Set |
| Accuracy | 0.9187 | 0.8222 |
| Precision | 0.875 | 0.8182 |
| Recall | 0.8305 | 0.7297 |
| F1-Score | 0.8522 | 0.7714 |

**Results**

The Decision Tree Classifier achieved a high accuracy score of 91.87% on the training set but a lower accuracy of 82.22% on the testing set. This suggests potential overfitting, where the model performs well on the training data but may not generalize well to unseen data. This suggests that the model might be overly complex and capturing noise in the training data that doesn't generalize well.

**Classification with PRUNED DECISION TREE**

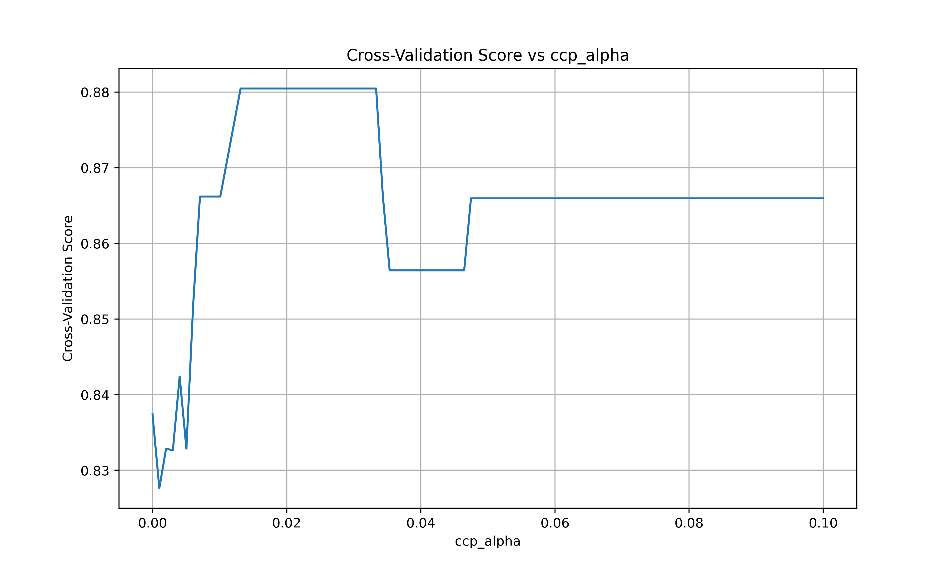
This section addresses the overfitting issue observed in the unpruned decision tree by incorporating cost complexity pruning (CCP).

**Cost Complexity Pruning (CCP)**

CCP is a technique for pruning decision trees by introducing a cost complexity parameter (alpha). The goal is to find the optimal alpha value that balances the model's fit to the training data with its generalization ability.

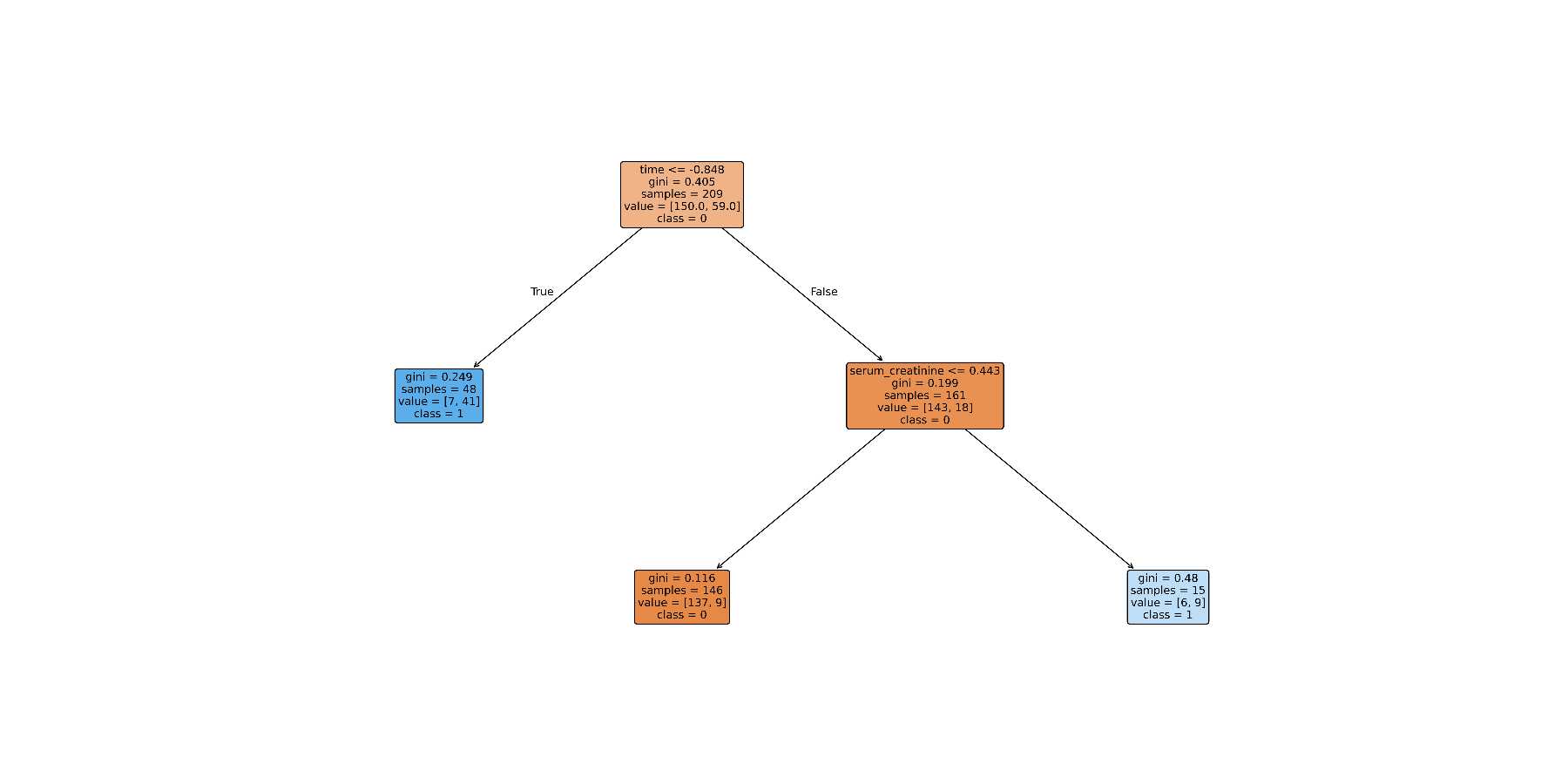
**Finding the Optimal Alpha**

1. **Define Alpha Range:** A range of alpha values (ccp\_alphas) is defined to explore.
2. **Cross-Validation:** For each alpha value, a decision tree is created and evaluated using cross-validation (here, 10-fold CV). The average cross-validation score is recorded.
3. **Optimal Alpha Selection:** The alpha value corresponding to the highest average CV score is considered the optimal alpha for pruning the tree.



**Results**

From the above analysis, we identified an optimal alpha value of 0.01313 based on the cross-validation scores. The plot of alpha vs. CV score given above visualizes this selection process.



**Pruned Tree Performance**

A decision tree model was created using the optimal alpha value and evaluated on both the training and testing sets. Here's a summary of the results:

**Confusion Matrix for Training Set:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predicted Class | True Class | | | |
|  | NON-DEATH(-) | DEATH(+) | Total |
| NON-DEATH(-) | 137 | 9 | 146 |
| DEATH(+) | 13 | 50 | 63 |
| Total | 150 | 59 | 209 |

**Confusion Matrix for Testing Set:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predicted Class | True Class | | | |
|  | NON-DEATH(-) | DEATH(+) | Total |
| NON-DEATH(-) | 45 | 15 | 60 |
| DEATH(+) | 8 | 22 | 30 |
| Total | 53 | 37 | 90 |

**Evaluation Metrics:**

|  |  |  |
| --- | --- | --- |
| Metric | Training Set | Testing Set |
| Accuracy | 0.8947 | 0.7444 |
| Precision | 0.7937 | 0.7333 |
| Recall | 0.8475 | 0.5946 |
| F1-Score | 0.8197 | 0.6567 |

The pruned tree achieved a slightly lower accuracy on the training set (89.47%) compared to the unpruned tree (91.87%) and same on the testing set (74.44% vs. 82.22%). The pruned tree exhibits better generalizability on the testing set while maintaining reasonable performance on the training set.

**Classification with BAGGING Ensemble**

This section explores Bagging, an ensemble method for improving the performance and stability of decision tree models.

**Bagging Classifier**

Bagging (Bootstrap Aggregating) is an ensemble learning technique that trains multiple decision trees on different subsets of the training data with replacement. By combining the predictions from these individual trees, Bagging aims to reduce the variance of the model and improve its generalization ability.

**Model Creation and Evaluation**

* A base decision tree classifier with a minimum samples per leaf of 5 (min\_samples\_leaf=5) was defined.
* A Bagging classifier was created using the base decision tree as the base estimator and setting the number of estimators (n\_estimators) to 100. This means 100 decision trees will be trained on bootstrapped samples of the training data.
* The Bagging classifier was trained on the training data (70% of the dataset).
* The model's performance was evaluated on both the training and testing sets.

**Confusion Matrix for Training Set:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predicted Class | True Class | | | |
|  | NON-DEATH(-) | DEATH(+) | Total |
| NON-DEATH(-) | 132 | 9 | 141 |
| DEATH(+) | 8 | 60 | 68 |
| Total | 140 | 69 | 209 |

**Confusion Matrix for Testing Set:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predicted Class | True Class | | | |
|  | NON-DEATH(-) | DEATH(+) | Total |
| NON-DEATH(-) | 54 | 12 | 66 |
| DEATH(+) | 9 | 15 | 24 |
| Total | 63 | 27 | 90 |

**Evaluation Metrics:**

|  |  |  |
| --- | --- | --- |
| Metric | Training Set | Testing Set |
| Accuracy | 0.9091 | 0.7556 |
| Precision | 0.8226 | 0.7586 |
| Recall | 0.8644 | 0.5946 |
| F1-Score | 0.843 | 0.6667 |

**Results**

The Bagging classifier achieved a high training accuracy of 90.91%, indicating good performance on the data it was trained on. However, the testing accuracy was lower at 75.56%, suggesting potential for improvement in generalizability.

**Classification with RANDOM FOREST**

This section explores Random Forest, a powerful ensemble method that builds upon the success of Bagging.

**Random Forest Classifier**

Random Forest is an ensemble learning technique that combines multiple decision trees trained on different subsets of the data with replacement (similar to Bagging). However, Random Forest introduces an additional layer of randomness during tree creation by randomly selecting a subset of features at each split point. This helps to decorrelate the trees and further reduce variance, leading to a more robust model.

**Model Creation and Evaluation**

* A Random Forest classifier was created with 100 decision trees (n\_estimators=100), a random state of 35 (random\_state=35) for reproducibility, and a minimum samples per leaf of 5 (min\_samples\_leaf=5).
* The Random Forest classifier was trained on the training data (**70% of the dataset**).
* The model's performance was evaluated on both the training and testing sets.

**Confusion Matrix for Training Set:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predicted Class | True Class | | | |
|  | NON-DEATH(-) | DEATH(+) | Total |
| NON-DEATH(-) | 146 | 8 | 154 |
| DEATH(+) | 4 | 51 | 55 |
| Total | 150 | 59 | 209 |

**Confusion Matrix for Testing Set:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predicted Class | True Class | | | |
|  | NON-DEATH(-) | DEATH(+) | Total |
| NON-DEATH(-) | 48 | 17 | 65 |
| DEATH(+) | 5 | 20 | 25 |
| Total | 53 | 37 | 90 |

**Evaluation Metrics:**

|  |  |  |
| --- | --- | --- |
| Metric | Training Set | Testing Set |
| Accuracy | 0.9426 | 0.7556 |
| Precision | 0.9273 | 0.8 |
| Recall | 0.8644 | 0.5405 |
| F1-Score | 0.8947 | 0.6452 |

**Results & Interpretation**

The Random Forest classifier achieved an even higher training accuracy (94.26%) compared to the Bagging model, indicating strong performance on the training data. However, the testing accuracy remained at 75.56%, similar to the previous models, suggesting that all three models (decision tree, bagging, random forest) might be suffering from limitations in the data or require further hyperparameter tuning.

**Hyperparameter Tuning: RANDOM FOREST**

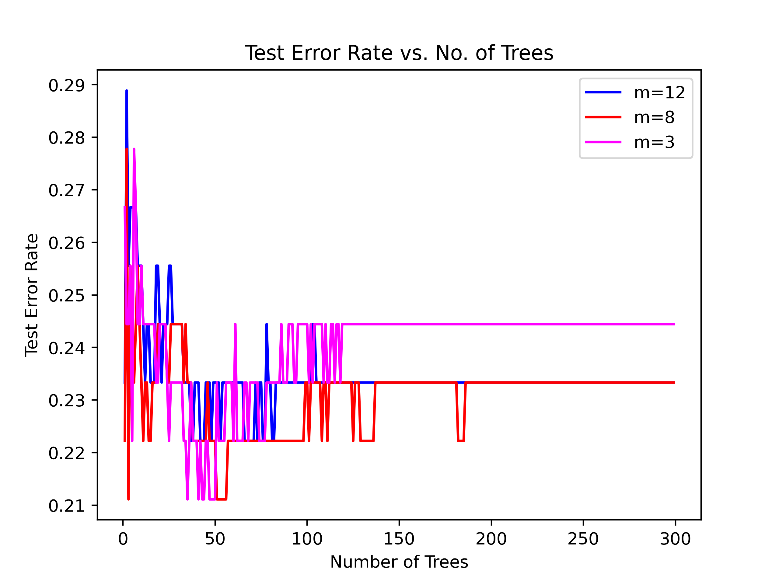
Random Forest is a powerful ensemble learning method that leverages multiple decision trees for improved performance and robustness. A key hyperparameter in Random Forest is the number of features out of the total 12 features that should be allowed to consider to perform a particular split in the ensemble (*m*). Here, we explored the impact of the number of trees on the model's performance using K-Fold Cross-Validation (CV) with a fixed k of 10 (as chosen in the previous section).

**Experiment Design**

We trained Random Forest models with varying numbers of trees (n\_estimators) ranging from 1 to 300. Three configurations were evaluated:

* **m=12:** All 12 features are allowed to be considered for a particular split.
* **m=8:** Only 8 out of 12 features are allowed to be considered for a particular split.
* **m=3:** Only 3 out of 12 features are allowed to be considered for a particular split.

For each configuration, a Random Forest model was fit on the training data, and its performance was evaluated on the unseen testing data. The error rate (1 - accuracy) was calculated to compare the models.



**Results**

The graph (refer to the image above) illustrates the Test Error Rate for each configuration as the number of trees (n\_estimators) increases. As expected, the error rate generally decreases as the number of trees increases, indicating improved model performance with more trees in the ensemble. However, the rate of improvement diminishes after a certain point, suggesting potential diminishing returns from adding more trees.

**Key Points**

* The number of trees (n\_estimators) is a crucial parameter in Random Forest models.
* Increasing the number of trees generally improves performance, but with diminishing returns.
* This analysis suggests that a moderate number of trees (between 50 and 150) might be sufficient for this dataset based on the observed trends.

**Classification with SUPPORT VECTOR MACHINE (SVM)**

This section explores Support Vector Machines (SVMs), a powerful machine learning algorithm for classification tasks.

**Support Vector Machines (SVM)**

SVMs aim to find a hyperplane in the feature space that best separates the data points of different classes with the maximum margin. This margin is defined by the distance between the hyperplane and the closest data points of each class, called support vectors. SVMs offer strong performance in various classification problems and can handle high-dimensional data effectively.

**Model Creation and Evaluation**

* An SVM classifier with a sigmoid kernel was created (SVC(kernel='sigmoid', C=10)). The sigmoid kernel is a popular choice for non-linear problems especially when the response is binary. The hyperparameters *C* control the trade-off between model complexity and overfitting.
* The SVM model was trained on the training data (**70% of the dataset**).
* The model's performance was evaluated using following metrics:

**Training Accuracy: 0.7894**

**Test Accuracy: 0.7777**

**10-Fold CV Accuracy: 0.5988**

**10-Fold Error rate : 0.4011**

**Results**

The SVM model achieved a low training accuracy (78.94%) and the same testing accuracy (77.77%). This suggests that the model might be capturing the underlying structure of the data effectively. The 10-Fold CV accuracy (59.88%) is unable to provide a more reliable estimate of the model's generalizability.

**Interpretation**

The current SVM model configuration underperforms on this classification task. The low accuracy suggests potential learning problems with the model's ability to learn the relationships between features and the target variable.

**Classification with LINEAR DISCRIMINANT ANALYSIS (LDA)**

This section explores Linear Discriminant Analysis (LDA), a classification technique that assumes a linear relationship between features and class labels.

**Linear Discriminant Analysis (LDA)**

LDA projects data points onto a lower-dimensional space while maximizing the separation between classes. Unlike SVMs, which find a hyperplane for separation, LDA assumes that the data classes are linearly separable. It identifies a linear discriminant function that best separates the data points of different classes. LDA performs well when the class distributions are Gaussian and the classes are well-separated in the feature space.

**LDA Assumptions**

Linear Discriminant Analysis (LDA) assumes that the data is normally distributed within each class and that the covariance matrices of the different classes are equal. While these assumptions might not be strictly met in our dataset, LDA can still perform well under certain conditions.

Despite these assumptions, LDA can still perform well if:

* The classes are well-separated, even if the distributions are not perfectly normal.
* The number of features is significantly smaller than the number of samples.
* The model is regularized to prevent overfitting, especially when the number of features is large.

**Model Creation and Evaluation**

* An LDA model was created (LinearDiscriminantAnalysis()).
* The LDA model was trained on the training data (**70% of the Dataset**).
* The model's performance was evaluated.

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | Training Set | Testing Set | 10-Fold CV |
| Accuracy | 0.8804 | 0.8111 | 0.809 |
| Precision | 0.8036 | 0.8846 | - |
| Recall | 0.7627 | 0.6216 | - |
| F1-Score | 0.7826 | 0.7302 | - |

**Results**

The LDA model achieved a reasonable training accuracy (88.04%) and a good testing accuracy (81.11%). This suggests that the model captures the underlying structure of the data to some extent. The 10-Fold CV accuracy (80.90%) reinforces the performance observed on the training and testing sets. The 10-Fold CV Error Rate (19.10%) indicates potential for improvement, but overall, LDA seems to be a viable contender for this classification task.

In the case of our dataset, while the assumptions might not be perfectly met, the model's performance suggests that LDA is still able to capture the underlying patterns in the data and make accurate predictions. This could be due to factors like the relatively simple structure of the problem, the presence of informative features, or the robustness of the LDA algorithm to deviations from its assumptions.

**Classification with QUADRATIC DISCRIMINANT ANALYSIS (QDA)**

This section explores Quadratic Discriminant Analysis (QDA), a classification technique that relaxes the linearity assumption of LDA.

**Quadratic Discriminant Analysis (QDA)**

QDA builds upon LDA by allowing for non-linear relationships between features and the target variable. Unlike LDA, which assumes a linear decision boundary between classes, QDA can learn more complex decision boundaries using quadratic functions. This flexibility comes at the cost of increased model complexity and potential for overfitting.

**QDA Assumptions**

Quadratic Discriminant Analysis (QDA) assumes that the data within each class is normally distributed and that the covariance matrices of different classes are different. While these assumptions might not be strictly met in our data, QDA can still perform well under certain conditions.

Despite these assumptions, QDA can still perform well if:

* The classes are well-separated, even if the distributions are not perfectly normal.
* The number of features is significantly smaller than the number of samples.
* The model is regularized to prevent overfitting, especially when the number of features is large.

In the case of our dataset, while the assumptions might not be perfectly met, the QDA model's performance suggests that it can still capture the underlying patterns in the data and make reasonable predictions. However, the model's tendency to overfit is evident in the lower testing accuracy compared to the training accuracy.

**Model Creation and Evaluation**

* A QDA model was created (QuadraticDiscriminantAnalysis()).
* The QDA model was trained on the training data (**70% of the dataset**).
* The model's performance was evaluated.

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | Training Set | Testing Set | 10-Fold CV |
| Accuracy | 0.8517 | 0.7 | 0.7524 |
| Precision | 0.78 | 0.75 | - |
| Recall | 0.661 | 0.4054 | - |
| F1-Score | 0.7156 | 0.5263 | - |

**Results**

The QDA model achieved a reasonable training accuracy (85.17%), but the testing accuracy was lower (70.00%). This suggests that the model might be overfitting the training data and not generalizing well to unseen data.

The 10-Fold CV accuracy (75.24%) confirms the limitations observed in the testing set and indicates potential overfitting. The 10-Fold CV Error Rate (24.76%) suggests room for improvement. The QDA model shows a higher training accuracy compared to LDA but suffers from overfitting, leading to a lower testing accuracy. While QDA can handle non-linear relationships, it might be more prone to overfitting with complex datasets or limited data.

**Classification with NAIVE BAYES CLASSIFIER**

This section explores Naive Bayes, a probabilistic classification technique based on Bayes' theorem.

**Naive Bayes Classifier**

Naive Bayes assumes independence between features, meaning that the presence of one feature does not influence the presence of another feature given the class label. This assumption simplifies the model and allows for efficient learning. Naive Bayes works well for data with features that are relatively independent and follow a Gaussian distribution. Categorical features are incorporated in the model without explicit preprocessing.

**To check independence of Categorical-Categorical Features (Chi-Square Test):**

Tests revealed that some categorical variables like anaemia, diabetes, and smoking are not independent (p-values < 0.05). The violation of independence assumptions indicates relationships between certain predictors, which Naive Bayes does not explicitly account for.

**To check the independence of Continuous-Continuous Features (Pearson Correlation Coefficient):**

Certain continuous features (e.g., age and time) show moderate to high correlation. Correlated features may introduce redundancy but are handled reasonably well by Gaussian Naive Bayes.

**To check the independence of Categorical-Continuous Features (ANOVA F-test):**

Dependencies exist between categorical variables (e.g., sex and smoking) and continuous features (e.g., ejection\_fraction). The presence of statistical dependencies further challenges the Naive Bayes assumption of independence.



**Model Creation and Evaluation**

* A Naive Bayes model using a Gaussian distribution (GaussianNB()) was created.
* The Naive Bayes model was trained on the training data (**70% of the dataset**).
* The model's performance was evaluated.

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | Training Set | Testing Set | 10-Fold CV |
| Accuracy | 0.8278 | 0.7667 | 0.7856 |
| Precision | 0.7556 | 0.8636 | - |
| Recall | 0.5763 | 0.5135 | - |
| F1-Score | 0.6538 | 0.6441 | - |

**Results**

The Naive Bayes model achieved a reasonable training accuracy (82.78%) but a slightly lower testing accuracy (76.67%). This suggests that the model captures some of the underlying relationships in the data but might benefit from further improvements.

The 10-Fold CV accuracy (78.56%) aligns with the observations from the training and testing sets. The 10-Fold CV Error Rate (21.44%) indicates potential for improvement.

**Interpretation**

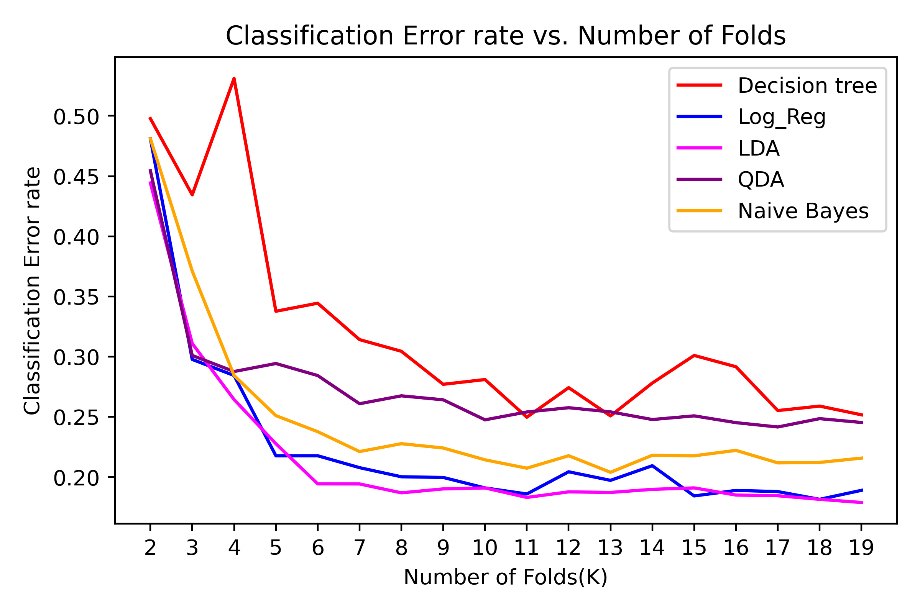
The Naive Bayes model shows reasonable performance on both the training and testing sets. While the model might benefit from addressing the assumption of conditional independence between features, it seems to capture some relevant information for classification. The lower recall compared to precision suggests room for improvement in identifying true positive cases.

**Reason behind K=10 for K-Fold Cross-Validation**

K-Fold Cross-Validation (CV) is a technique for evaluating the performance of machine learning models. It involves splitting the data into k folds, training the model on k-1 folds, and evaluating it on the remaining fold. This process is repeated k times, ensuring each data point is used for both training and testing. The average performance across all folds provides a more robust estimate of the model's generalizability compared to a single train-test split.

**Choosing the Right K**

The number of folds (k) is a crucial parameter in K-Fold CV. While a higher k value generally reduces the variance of the performance estimate, it also increases the computational cost. Here, we explored various classification algorithms (Decision Tree, Logistic Regression, LDA, QDA, and Naive Bayes) and compared their performance using K-Fold CV with different values of k (from 2 to 19).



**Error Rate vs. Number of Folds**

The graph above depicts the Classification Error Rate () for each model across different k values. As expected, the error rate tends to fluctuate but generally stabilizes as the number of folds increases. This suggests that a higher k value might provide a more reliable estimate of the model's performance, but there might be a point of diminishing returns after a certain value of k.

**In this case, based on the graph, a k value of 10 seems like a reasonable choice for all the models.** It offers a balance between reducing variance and computational efficiency.

**OVERALL CONCLUSION OF THE STUDY**

In this study, we compared the performance of several classification algorithms for predicting heart failure death risk using a dataset with 13 clinical features. The results indicate that ensemble methods, particularly **Bagging and Random Forest**, showed promising performance with high accuracy on both the training and test sets. Notably, Random Forest achieved an accuracy of 0.9426 on the training set and 0.7556 on the test set, suggesting good generalization ability.

|  |  |  |
| --- | --- | --- |
| Classification Method | Accuracy | |
| Training Set | Testing Set |
| LDA | 0.8804 | 0.8111 |
| LDA(10-Fold) | - | 0.809 |
| QDA | 0.8517 | 0.7 |
| QDA(10-Fold) | - | 0.7524 |
| Naive Bayes | 0.8278 | 0.7667 |
| Naive Bayes(10-Fold) | - | 0.7856 |
| Logistic Reg. | 0.8517 | 0.8333 |
| Logistic Reg.(10-Fold) | - | 0.8156 |
| Logistic Reg.(LOOCV) | - | 0.8294 |
| Decision Tree(Unpruned) | 0.9187 | 0.8222 |
| Decision Tree(Pruned) | 0.8947 | 0.7444 |
| Bagging | 0.9091 | 0.7556 |
| Random Forest | 0.9426 | 0.7556 |
| SVM | 0.7894 | 0.7777 |
| SVM(10-Fold) | - | 0.5988 |

**Key Findings are:-**

**Decision Trees:** While decision trees can be effective for classification tasks, they are prone to overfitting, especially when not pruned properly.

**Bagging and Random Forest:** These ensemble methods significantly improved the model's performance compared to single decision trees. They reduced overfitting and enhanced generalization.

**Logistic Regression:** Despite its simplicity, logistic regression achieved reasonable performance, demonstrating its effectiveness in binary classification tasks.

**SVM:** SVM with the sigmoid kernel showed mixed results. It struggled to capture complex patterns in the data and suffered from potential overfitting.

**LDA and QDA:** These linear and quadratic discriminant analysis models performed well, even after their assumptions are not followed. However, they might be limited by the linearity assumption in some cases.

**Naive Bayes:** While Naive Bayes assumes feature independence, it still achieved reasonable performance, indicating that the features might be relatively independent or the model is robust to violations of this assumption.