**INTERNATIONAL UNIVERSITY**

**VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY**

**-------------------------------------------------------**

**School of Computer Science and Engineering**

**--------------**

**PROJECT REPORT**

**HEART FAILURE**

**PREDICTION**

**ARTIFICIAL INTELLIGENCE (IT159IU)**

**Course by Br. Le Thanh Sach and Dr. Ly Tu Nga**

**TEAM MEMBERS: STUDENTS' ID:**

Tran Dinh An ITDSIU20113

Tran Ngoc Dang Khoi ITCSIU21197

Nguyen Hai Ngoc ITDSIU21057

**Ho Chi Minh City, May 2024**

**TABLE OF CONTENTS**

[**CONTRIBUTION TABLE** 2](#_Toc167140346)

[**CHAPTER 1: INTRODUCTION** 3](#_Toc167140347)

[**1.** **Methodology:** 3](#_Toc167140348)

[**2.** **About the dataset:** 4](#_Toc167140349)

[**3.** **Productive Tools** 4](#_Toc167140350)

[**CHAPTER 2: EDA & DATA PREPROCESSING** 5](#_Toc167140351)

[1. Exploratory Data Analysis (EDA): 5](#_Toc167140352)

[**1.1. Initial Inspection:** 5](#_Toc167140353)

[**1.2. Correlation matrix:** 6](#_Toc167140354)

[**1.3. Outliers:** 7](#_Toc167140355)

[**1.4. Conclusion:** 9](#_Toc167140356)

[2. Data Preprocessing: 9](#_Toc167140357)

[**2.1. Handling inapproriated datatype:** 9](#_Toc167140358)

[**2.2. Handling outliers by Robust Scaler:** 9](#_Toc167140359)

[**2.3. Splitting data:** 11](#_Toc167140360)

[**CHAPTER 3: MODELING AND EVALUATION** 12](#_Toc167140361)

[**1.** **Naive Bayesian** 12](#_Toc167140362)

[**2.** **Support Vector Machine** 15](#_Toc167140363)

[**3.** **Artificial Neural Networks** 17](#_Toc167140364)

[**CHAPTER 4: COMPARISION** 0](#_Toc167140365)

[**CHAPTER 5: CONCLUSION AND FUTURE WORKS** 3](#_Toc167140366)

[**1.** **Conclusions** 3](#_Toc167140367)

[**2.** **Future Works** 3](#_Toc167140368)

[**REFERENCES** 4](#_Toc167140369)

# **CONTRIBUTION TABLE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Full Name | Student ID | Task | Contribution |
| 1 | Trần Đình An | ITDSIU20113 |  | 33.3% |
| 2 | Trần Ngọc Đăng Khôi | ITCSIU21197 |  | 33.3% |
| 3 | Nguyễn Hải Ngọc | ITDSIU21057 |  | 33.3% |

# **CHAPTER 1: INTRODUCTION**

In the era of AI, the ability to effectively preprocess and analyze datasets has become increasingly vital. This project focuses on leveraging the dataset ‘Heart Failure Prediction’ from Kaggle to explore the capabilities of three distinct machine learning models: Naive Bayes, Support Vector Machine (SVM), and Artificial Neural Network (ANN). The goal is to evaluate and compare their performance in a structured manner.

By the end of this project, we aim to identify which model performs best under the given conditions and provide recommendations for similar classification tasks. This report details the methodology, findings, and conclusions derived from our analysis.

### **Methodology:**

The first phase of the project involves preprocessing and Exploratory Data Analysis (EDA) of the ‘Heart Failure Prediction’ dataset. Preprocessing is a critical step that ensures the data is clean, consistent, and suitable for analysis. Exploratory Data Analysis (EDA) is conducted to uncover initial insights and detect patterns. This phase includes handling data inconsistencies, scaling numerical features, and splitting the data into training and testing sets. Statistical analysis and data visualization techniques are also implemented to provide a foundation for selecting appropriate features, inform the choice of algorithms, and understand the underlying structure of the data.

The core of the project is the deployment of three machine learning models: Naive Bayes, SVM, and ANN. Each model is trained on 2 sets of preprocessed data – one with all the features and one with only the topmost important features. These models are chosen for their diverse approaches to classification problems:

* Naive Bayes: A probabilistic classifier based on Bayes' theorem, particularly effective for high-dimensional data.
* Support Vector Machine (SVM): A powerful classifier that finds the optimal hyperplane for separating classes in the feature space.
* Artificial Neural Network (ANN): A flexible model that mimics the human brain's neural networks, capable of capturing complex patterns.

The final phase involves comparing the performance of these models using standard evaluation metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. This comparative analysis provides insights into the strengths and weaknesses of each model in relation to the specific dataset used.

### **About the dataset:**

Cardiovascular diseases (CVDs) are the number 1 cause of death globally, taking an estimated 17.9 million lives each year, which accounts for 31% of all deaths worldwide. Heart failure is a common event caused by CVDs, and this dataset contains 12 features that can predict mortality by heart failure.

Most cardiovascular diseases can be prevented by addressing behavioral risk factors such as tobacco use, unhealthy diet and obesity, physical inactivity, and harmful use of alcohol using population-wide strategies. People with cardiovascular disease or who are at high cardiovascular risk (due to the presence of one or more risk factors such as hypertension, diabetes, hyperlipidemia or already established disease) need early detection and management wherein a machine learning model can be of great help.

### **Productive Tools**

* Google Colab: a hosted Jupyter Notebook service that requires no setup to use and provides free access to computing resources, including GPUs and TPUs.
* Library used:
  + NumPy: For array operations and mathematical functions.
  + pandas: For data manipulation and analysis.
  + Matplotlib and Seaborn: For data visualization.
  + scikit-learn: For machine learning tasks.
  + Keras: For building and training neural network models.

# **CHAPTER 2: EDA & DATA PREPROCESSING**

This part details the exploratory data analysis (EDA) and preprocessing steps undertaken for a project on predicting heart failure events. The dataset used contains clinical records of patients with various features that may help predict heart failure outcomes.

## Exploratory Data Analysis (EDA):

Exploratory Data Analysis is the initial step aimed at understanding the dataset and uncovering patterns. Especially in this case, we explore the structure, distribution, and relationships within the data to figure out the risk of dataset.

### 1.1. Initial Inspection:

In general, the ‘Heart Failure Prediction’ dataset contains clinical records of patients, with various features that may contribute to predicting heart failure:

+ It includes 299 rows and 13 columns with no missing value by using `df.shape()` and `df.info()`

+ It includes 13 features with patient demographics, medical history and clinical measurement information.

+ Main datatype: Float and Integer, both numerical and binary representation included.

+ There is no duplicated row existing in the dataset. However, the ‘Float’ datatype is not appropriate with ‘age’ attribute.

A computer screen shot of a program

Description automatically generated

*Figure. Description of dataset*

### 1.2. Correlation matrix:

To check the correlations between the key attribute, `DEATH\_EVENT`, with the other attributes, we use correlation heatmap to check.

A graph of a heatmap

Description automatically generated

*Figure. Correlation Heatmap of “Heart Failure Prediction” dataset*

Here, we easily see that there are four attributes that has the typical correlation to the `DEATH\_EVENT` attributes:

* `time` attributes as the most significant negative correlation with -0.53.
* `ejection\_fraction ` as the negative correlation with -0.27.
* `serum\_creatinine` as the most significant positive correlation values with 0.29.
* `age ` as the positive correlation values with 0.29.

In order to make sense about the trend of data, we create `pair plot` graph.

A graph of a number of data

Description automatically generated with medium confidence

*Figure. ‘Pair Plot’ graph*

Overall, there is no significant trend between these four typical attributes. It may come from the unconsiderable correlation between them, which is testified by the correlation heatmap.

### 1.3. Outliers:

To check the outliers of the dataset, we implement `Box plot` to visualize all the numerical attributes of dataset. A box plot (or box-and-whisker plot) shows the distribution of quantitative data in a way that facilitates comparisons between variables.The box shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution.The box plot (a.k.a. box and whisker diagram) is a standardized way of displaying the distribution of data based on the seven numerical attributes:

+ age: age of the patient.

+ creatinine\_phosphokinase: Level of the CPK enzyme in the blood (mcg/L).

+ ejection\_fraction: Percentage of blood leaving the heart at each contraction.

+ platelets: Platelets in the blood (kiloplatelets/mL)

+ serum\_creatinine: Level of the serum creatinine in blood (mg/dL)

+ serum\_sodium: Level of serum sodium in the blood (mEq/L)

+ time: Follow-up period (days)

A screenshot of a computer screen

Description automatically generated

*Figure. ‘Box Plot’graph of numerical attributes in dataset.*

Overall, almost outliers exist in numerical clinical measurement attributes, except for `time` and `age` attribute. It could come from the unknown clinical measurement, which possibly impacts the final result.

### 1.4. Conclusion:

In conclusion, we approach the properties of the dataset with clear description:

+ There are 299 non-null values in all the attributes, thus no missing value.

+ No duplicated rows.

+ There are many outliers existing in the dataset, which are mainly consisted in clinical attributes with unknown reasons.

+ Datatype is also either `Float` or `Integer`, all works well except age, whose datatype `Float` is not suitable for that attribute.

## Data Preprocessing:

### 2.1. Handling inapproriated datatype:

To handle the inpproriated datatype mainly happening in `age` attribute, we use `.astype()` method provided by Pandas to convert the `age` attribute’s datatype into the `int64` (Integer). This ensures consistency and allows for proper numerical analysis.

|  |  |
| --- | --- |
| A screen shot of a computer program  Description automatically generated  *Figure. Convert ‘float64’ of ‘age’ attribute into ‘int64’.* | A screenshot of a computer  Description automatically generated  *Figure. Result of conversion.* |

### 2.2. Handling outliers by Robust Scaler:

Feature Scaling is one of the most important steps of Data Preprocessing. Due to the presence of outliers with unknown reasoning in the clinical attributes of the dataset, we opted to use the “Robust Scaler” to scale the dataset's values.

The Robust Scaler is a scaling technique that is particularly effective in handling outliers. Unlike standard scaling methods that use the mean and standard deviation, the Robust Scaler uses the median and the interquartile range (IQR) for scaling. This makes it more resilient to outliers, as these statistics are less influenced by extreme values. It works by:

A group of black text

Description automatically generated with medium confidence

*Figure. Robust Scaler formula.*

+ Median Centering: Each feature is centered around the median, ensuring that the scaling is not skewed by outliers.

+ Interquartile Range Scaling: Features are scaled according to the IQR, which is the range between the 25th and 75th percentiles. This method reduces the impact of outliers by focusing on the central 50% of the data.

By using the Robust Scaler, we ensure that the scaled dataset maintains robustness against outliers, leading to more reliable and stable models.

**A computer screen shot of a program code

Description automatically generated**

*Figure. Implementation of ‘Robust Scaler’.*

**A screenshot of a computer

Description automatically generated**

*Figure. Result.*

### 2.3. Splitting data:

To prepare our dataset for modeling, we first separate the features from the target variable (`DEATH\_EVENT`) and then split the data into training, validation, and testing sets. This ensures that our model can be trained and validated effectively before being tested on unseen data.

A screen shot of a computer code

Description automatically generated

*Figure. Splitting data for preparation.*

To do that:

+ We split dataset into training and testing sets, with 80% of the data used for training and 20% reserved for testing. The `random\_state=42` ensures the reproducible split.

+ Furtherly, the training data is split into training and validation sets, with 75% of the data used for training and 25% for validation. This allows us to tune our model and validate its performance during the training process.

By structuring the data splits in this manner, we ensure that we have a robust training set, a validation set to tune and evaluate our model during development, and a testing set to asess the model’s performance on unseen data.

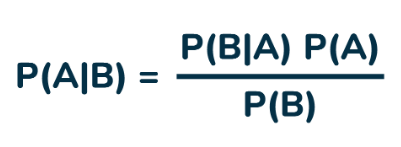
# **CHAPTER 3: MODELING AND EVALUATION**

This chapter outlines our approach to modeling mortality prediction in heart failure cases and evaluating model performance. We present details of the design and architecture of Naive Bayes, Support Vector Machine, and Artificial Neural Network models. We then discuss evaluation metrics such as accuracy, precision, recall, F1-score, and ROC-AUC, alongside visual aids like confusion matrices and ROC curves.

### **Naive Bayesian**

Naive Bayes algorithm is a probabilistic classifier based on Bayesian’s Theorem, with the “naive” assumption of independence between every pair of features. Despite its simplicity, it often performs well in various applications.

Bayesian’s Theorem provides a way to update the probability estimate for a hypothesis as more evidence or information becomes available. It is mathematically expressed as:



Where:

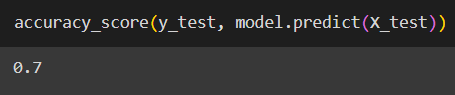
* P(A|B) is the posterior probability of the hypothesis A given the evidence of B
* P(B|A) is the likelihood of the evidence given that A is true.
* P(A) is the prior probability of the hypothesis before seeing the evidence.
* P(B) is the total probability of the evidence.

Naive Bayes assumption simplifies the calculation by assuming the features are conditionally independent given the class. Thus, for a set of features x = (x1, x2, ..., xn), the probability P(x|C) can be written as:

**P(x|C) = P(x1|C) . P(x2|C) . ... . P(xn|C)**

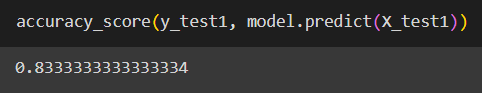
In Python, Naive Bayes model includes a parameter call *var\_smoothing*. This parameter is used to solve the Zero – Probability problem when computing the probability of each part. In this dataset, *var\_smoothing* is assigned to be default. It will be optimized in the future by the hyperparameter tuning process.

In the project, the model is trained twice. First, it is trained with all features of the dataset, which produces an accuracy of 70%.



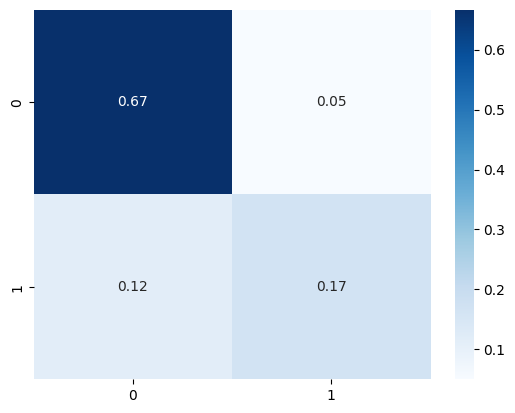
*Fig accuracy score of NB full attributes*

Then, the model is trained with 4 attributes that have the highest correlation to the target variable: *age, ejection\_fraction, serum\_creatinine,* and *time*. Obviously, the accuracy score increased to around 83%.



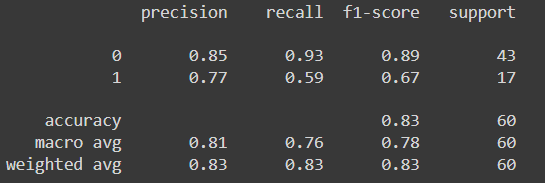
*Fig accuracy score of NB 4-attributes*

After training the model, the testing dataset is used to display the classification report, confusion matrix, and ROC AUC score.

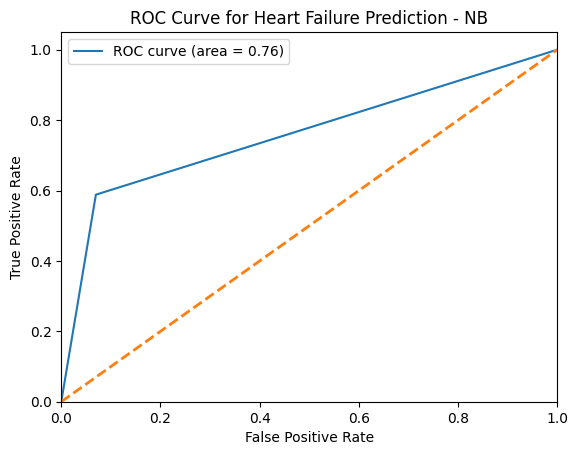


*Fig confusion matrix of NB*

By the confusion matrix, there is 67% of data that is true positive, 18% true negative, 5% false positive, and 12% false negative.



*Fig classification report*



*Fig ROC Curve*

We can easily observe that the precision, recall, and F1-score of class 0 is higher than the result in class 1. The recall of class 1 is high (0.93), which means that the number of correct samples that the model predicted over the total samples that is labeled as class 0, is 93%. Recall of class 1 is much smaller than recall of class 0, and the precision (77% that the sample predicted correctly by the model over the total of samples being predicted). That is the reason for the large range of difference between F1-score (stands for the combination of precision and the recall) of class 0 and class 1. The total accuracy states that the model correctly predicts the class labels for 83% of the samples. Finally, the ROC AUC score of 76% represents that the model has a high ability to separate between the two classes, and we can conclude that Naive Bayes did a better job of classifying the positive class in this dataset.

### **Support Vector Machine**

Support Vector Machine (SVM) is a supervised learning algorithm used for both classification and regression tasks. SVM is particularly well – suited for classification of complex datasets with a clear margin of separation.

Some key concepts of SVM include:

* **Hyperplane:** In an n-dimensional space, a hyperplane is an (n-1)-dimensional subspace that divides the space into two half-spaces. In SVM, the hyperplane is used to separate different classes.
* **Support Vectors:** These are the data points that are closest to the hyperplane. The position of the support vectors is crucial because they define the hyperplane's orientation and position.
* **Margin:** This is the distance between the hyperplane and the nearest data point from either class. SVM aims to maximize this margin, which helps in achieving better generalization.
* **Kernel Trick**: When data is not linearly separable in the original feature space, SVM uses a kernel function to transform the data into a higher-dimensional space where a hyperplane can be used to separate the classes. Common kernels include: Linear Kernel, Polynomial Kernel, RBF Kernel, and Sigmoid Kernel.

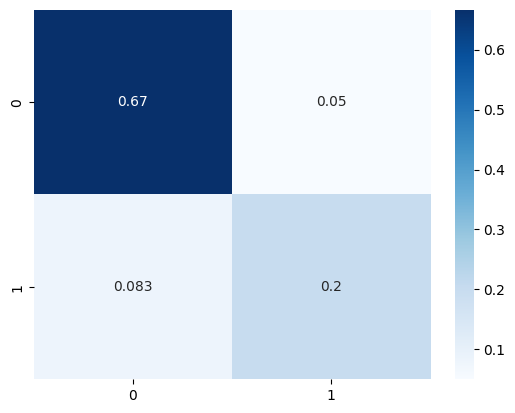
In Python, SVM model is defined as SVC. Parameters of SVC are:

* **C:** Regularization parameter. The strength of the regularization is inversely proportional to **C.** This parameter must be strictly positive.
* **kernel:** Specifies the kernel type to be used in the algorithm. It can be “linear”, “poly”, “rbf”, “sigmoid”, or callable.
* **degree:** Degree of the polynomial kernel function.
* **gamma:** kernel coefficient for “rbf”, “poly”, and “sigmoid”.
* **coef0:** Independent term in kernel function. It is only significant in “poly” and “sigmoid”.

The above information is the fundamental concept of SVM. However, the model is assigned to be default in this current code. In the future, we will optimize those parameters in the process called hyperparameter-tuning.

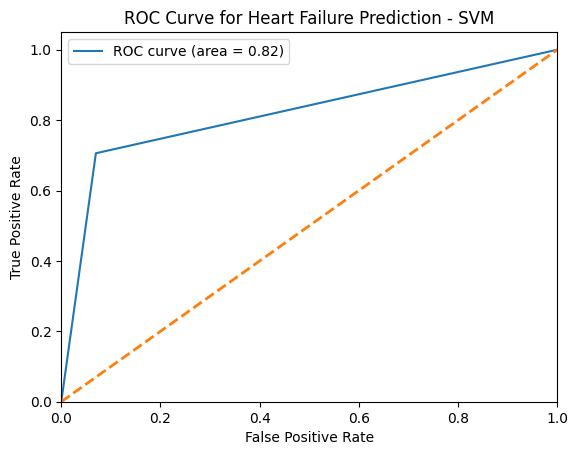
In the project, the model is trained twice. First, it is trained with all features of the dataset, which produces an accuracy of 66.7%, which means the model correctly predicts the class labels for 66.7% of the samples.

Then, the model is trained with 4 attributes that have the highest correlation to the target variable: *age, ejection\_fraction, serum\_creatinine,* and *time*. Obviously, the accuracy score increased to around 86.7%, which means the model correctly predicts the class labels for 86.7% of the samples after applying features selection.

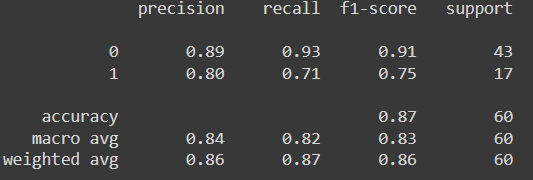


*Fig confusion matrix*

By the confusion matrix, there is 67% of data that is true positive, 20% true negative, 5% false positive, and 8.3% false negative.



*Fig ROC Curve*



*Fig classification report*

We can easily observe that the precision, recall, and F1-score of class 0 is slightly higher than the result in class 1. The recall of class 1 is high (0.93), which means that the number of correctly samples that the model predicted over the total samples that is labeled as class 0, is 93%. Recall of class 1 is much smaller than recall of class 0, and the precision (80% that the sample predicted correctly by the model over the total of samples being predicted). That is the reason for the large range of difference between F1-score (stands for the combination of precision and the recall) of class 0 and class 1. The total accuracy states that the model correctly predicts the class labels for 87% of the samples. Finally, the ROC AUC score of 82% represents that the model has a high ability to separate between the two classes, and we can conclude that Support Vector Machine did a better job of classifying the positive class in this dataset.

### **Artificial Neural Networks**

An Artificial Neural Network (ANN) is a computational model inspired by the biological neural networks of the human brain. It consists of interconnected nodes, or neurons, organized into layers. ANN is a type of deep learning algorithm capable of learning complex patterns and relationships from data.

Structure of an ANN:

* Input Layer: Neurons in this layer receive input data features.
* Hidden Layers: Intermediate layers between the input and output layers. These layers perform transformations on the input data through weighted connections and activation functions.
* Output Layer: Neurons in this layer provide the final output, often representing predictions or classifications.

A screenshot of a computer program

Description automatically generated*Fig ANN’s architecture and implementation of the model with all attributes.*

A neural network model is defined with an inpur layer, two hidden layers, each using ReLU activation, and one output layer using sigmoid activation for binary classification. A Sequential model, which is a linear stack of layers, is created first. For the input layer, a Dense layer is used. Dense layers are fully connected layers, meaning that each neuron in the layer is connected to every neuron in the preceding layer. We specified the number of neurons in this layer to be 16, the weights of the neurons are initialized uniformly, the activation function used in the neurons to be ReLU, and the number of input features to the model to be 12. The hidden layers are created using two another Dense layers with 8 neurons and ReLU activation. After each hidden layers, a dropout layer is added with fraction of 25% and 50%, respectively. Dropout is a regularization technique used to prevent overfitting in neural networks. It randomly drops a fraction of the connections between neurons during training to reduce interdependence between neurons. Finally, an output Dense layer with a single neuron and sigmoid activation is added. Sigmoid activation is commonly used in binary classification problems as it squashes the output between 0 and 1, representing the probability of the input belonging to the positive class.

After defining its architecture, the model is complied with Adam optimizer, binary cross-entropy loss function and specified the metric to evaluate the performance of the model during training and testing to accuracy. Adam is a popular optimization algorithm that combines the advantages of two other extensions of stochastic gradient descent: Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). Binary cross-entropy is the loss function used for binary classification problems. It measures the performance of a classification model whose output is a probability value between 0 and 1.

An EarlyStopping instance from Keras is created to stop training when a monitored metric has stopped improving. We specifies the minimum change in the monitored metric must be greater than 0.001 to qualify as an improvement. The number of epochs with no improvement after which training will be stopped is set at 20. And once training stops, the model weights from the epoch with the best value of the monitored metric will be restored. Finally, we train the model with a batch size of 25, run for up to 500 epochs, and include the ‘early\_stop’ instance we created earlier.

A graph of training and validation

Description automatically generatedA graph of a graph

Description automatically generated

*Fig Training, Validation loss & accuracy of the model with all attributes.*

During the training of the model, both training and validation losses showed a consistent decrease over the epochs, indicating that the model was learning and improving its performance. The training loss started at 0.6926 and gradually decreased, reaching lower values as training progressed. Similarly, the validation loss also showed a steady decline, starting from 0.6904 and improving significantly. The validation accuracy began high at 81.67% and remained consistently high throughout the training process, peaking at 85.00%. This suggests that the model was not only learning effectively on the training data but also generalizing well to the unseen validation data. The use of early stopping prevented overfitting, as evidenced by the stabilization of loss and accuracy metrics.

We then create another model with the same architecture of the previous model. The only difference from the first model is the ‘input\_dim’ parameter of the input layer to be 4, since this time we will only train the model using 4 most important attributes found in the ‘Preprocessing & EDA’ part.

A graph of training and validation

Description automatically generatedA graph with red lines

Description automatically generated

*Fig Training, Validation loss & accuracy of the model with 4 most important attributes*.

The new training history indicates that the model trained using only the four most important attributes shows a noticeable improvement in validation accuracy and loss compared to the previous model trained with all attributes. Initially, the validation accuracy remained at 81.67% for several epochs, similar to the earlier model, but it soon improved, peaking at 86.67% by epoch 27, and then stabilizing around 85% for the remainder of the epochs. The validation loss also consistently decreased, reaching a much lower value compared to the earlier model, demonstrating that the model trained on the selected important attributes was more effective and efficient. This suggests that focusing on the most relevant features can enhance model performance, reduce overfitting, and simplify the training process.

A blue squares with white text

Description automatically generated A graph of a curve

Description automatically generated

A screenshot of a computer screen

Description automatically generated

*Fig Confusion matrix, ROC Curve and Classification Report of the ANN model with 4 attributes.*

The classification report, ROC AUC score, and confusion matrix provide a comprehensive evaluation of the model's performance using the four most important attributes. The overall accuracy is 73%, indicating that the model correctly predicts the class labels for 73% of the samples. Precision, recall, and F1-score are higher for class 0 (precision: 0.74, recall: 0.83, F1-score: 0.78) than for class 1 (precision: 0.71, recall: 0.60, F1-score: 0.65), suggesting that the model performs better at identifying the negative class. The ROC AUC score of 0.71 reflects a fair level of separability between the classes. The confusion matrix shows that 48% of the predictions are true negatives, 25% are true positives, with 10% and 17% being false positives and false negatives, respectively. These results imply that while the model has a reasonable performance, there is room for improvement, particularly in reducing false negatives and enhancing the detection of the positive class

# **CHAPTER 4: COMPARISION**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

*Fig Classification Reports of each model*

Comparing the performance of Naive Bayes, Support Vector Machine (SVM), and Artificial Neural Network (ANN) models reveals distinct strengths and weaknesses for each approach. The Naive Bayes model shows strong performance with an overall accuracy of 83%. It has a high precision (0.85) and recall (0.93) for class 0, which translates to an F1-score of 0.89, indicating that it is very effective at identifying negative instances. However, the model's performance for class 1 is weaker, with a precision of 0.77 and a recall of 0.59, resulting in a lower F1-score of 0.67. The ROC AUC score of 0.76 suggests a moderate ability to distinguish between the classes.

The SVM model outperforms both Naive Bayes and ANN in several metrics. It achieves the highest overall accuracy at 87% and excels in both precision (0.89) and recall (0.93) for class 0, leading to an F1-score of 0.91. For class 1, SVM demonstrates better performance than Naive Bayes with a precision of 0.80 and a recall of 0.71, yielding an F1-score of 0.75. This indicates that the SVM model not only effectively identifies negative instances but also performs relatively well with positive instances. The ROC AUC score of 0.82 reflects its superior discriminative ability among the models compared.

The ANN model, while achieving a reasonable accuracy of 73%, lags behind the other two models. It has lower precision (0.74) and recall (0.83) for class 0 compared to Naive Bayes and SVM, with an F1-score of 0.78. For class 1, its precision is 0.71 and recall is 0.60, resulting in an F1-score of 0.65. The ANN model's ROC AUC score of 0.71 is the lowest among the three, indicating that it is less effective at distinguishing between the classes.

A screenshot of a graph

Description automatically generated

A graph with numbers and squares

Description automatically generated with medium confidence

*Fig Confusion Matrices of each model*

The confusion matrices of the Naive Bayes, Support Vector Machine (SVM), and Artificial Neural Network (ANN) models provide insights into their classification performance. These matrices show how each model predicts the two classes (0 and 1) and allow us to compare their effectiveness in identifying true positives and true negatives.

* Naive Bayes: The confusion matrix for the Naive Bayes model shows that it correctly classified 67% of the instances as true negatives and 17% as true positives. However, it misclassified 12% of the actual positives as negatives and 5% of the actual negatives as positives. This indicates a solid performance in identifying true negatives but a slightly higher rate of false negatives, suggesting the model is somewhat conservative in predicting the positive class.
* SVM: The confusion matrix for the SVM model indicates a slightly better performance in certain aspects. It correctly classified 67% of the instances as true negatives and 20% as true positives, showing an improvement in correctly identifying positives compared to Naive Bayes. The false negative rate is reduced to 8.3%, and the false positive rate remains at 5%. This matrix demonstrates that SVM has a balanced approach, maintaining a low rate of misclassifications while slightly improving the detection of true positives.
* ANN: The confusion matrix for the ANN model reveals different strengths and weaknesses. It correctly classified 48% of the instances as true negatives and 25% as true positives. However, the false negative rate is 17%, and the false positive rate is 10%. This indicates that the ANN model has a higher rate of misclassifications compared to Naive Bayes and SVM, particularly in predicting true negatives. The model has a more significant challenge in accurately identifying the negative class, which affects its overall performance.

In conclusion, the comparison of Naive Bayes, Support Vector Machine (SVM), and Artificial Neural Network (ANN) models highlights the importance of selecting the right algorithm for a specific task. While Naive Bayes demonstrates strong performance in identifying negative instances, it struggles with classifying positive instances accurately. SVM emerges as the top performer with high precision, recall, and accuracy for both classes, making it a robust choice for this classification task. On the other hand, ANN, despite its reasonable accuracy, falls short in comparison to the other models, particularly in distinguishing between the classes and minimizing misclassifications. Therefore, when considering classification tasks like this one, the SVM model stands out as the most reliable and effective option.

# **CHAPTER 5: CONCLUSION AND FUTURE WORKS**

### **Conclusions**

In conclusion, this project explored the performance of three machine learning models—Naive Bayes, Support Vector Machine (SVM), and Artificial Neural Network (ANN)—for a binary classification task. Each model exhibited distinct strengths and weaknesses in classifying instances of the dataset. Naive Bayes showed strong performance in identifying negative instances but struggled with positive instances, while SVM emerged as the top performer with high precision, recall, and accuracy for both classes. However, ANN lagged behind in terms of classification accuracy and discriminative ability compared to the other models.

### **Future Works**

Future work for this project could involve several avenues of exploration. Firstly, fine-tuning the hyperparameters of each model could potentially enhance their performance further. Additionally, exploring ensemble methods, such as combining the predictions of multiple models, might lead to improved classification accuracy and robustness. Furthermore, collecting more data or utilizing data augmentation techniques could help address any issues related to data scarcity or imbalance. Lastly, investigating advanced deep learning architectures or techniques, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), could provide insights into whether more complex models can yield better performance for this classification task. Overall, further experimentation and refinement of the models could lead to even more accurate and reliable classification results.

# **REFERENCES**

[1]: Davide C., Giuseppe J. (2020, 03 February): *Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone.* BMC Medical Informatics and Decision Making. <https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-020-1023-5>