MAIB23 – Machine Learning

Prediction on Heart Failure dataset

SVM vs ANN

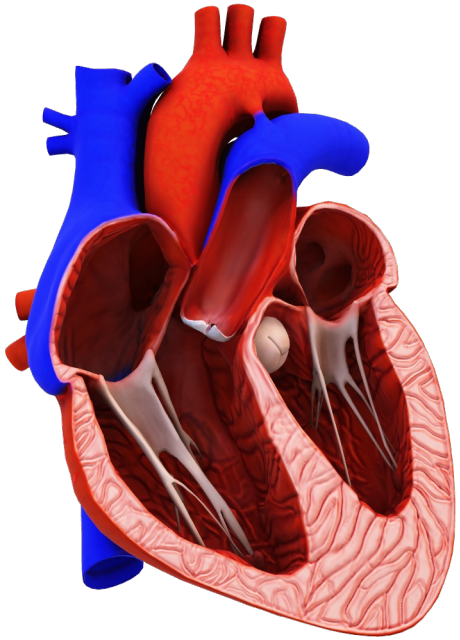
Farkhod Muradov Tran Khanh Ngoc Le

Content

1. **Introduction**
   An overview of background & objectives of project)
2. **Data exploration**
   Performance of exploring data analysis
3. **Implementation of Support Vector Machine (SVM)**
   Model Setup, Architecture & Evaluation
4. **Implementation of Artificial Neural Network (ANN)**
   Model Setup, Architecture & Evaluation
5. **Conclusion**
   Comparison between 2 models' performance and applications

1

Introduction



About this project

This project aims to use SVM and ANN models to predict heart failure in individuals, trained on a dataset with diverse heart health features. By evaluating both models, the goal is to find the best method for early heart failure detection. This effort will not only improve prediction accuracy but also offer a valuable tool for early medical intervention, leading to saved lives and decreased healthcare expenses.

2

Data Exploration

A glance of dataset

Made up by 13 features and 299 observations

No missing value

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0	4	1
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0	6	1
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1	7	1
3	50.0	1	111	0	20	0	210000.00	1.9	137	1	0	7	1
4	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	8	1

FEATURE TITLES

age: Age of the patient

anaemia: Haemoglobin level of patient (Boolean)

creatinine_phosphokinase: Level of the CPK enzyme in the blood (mcg/L)

diabetes: If the patient has diabetes (Boolean)

ejection_fraction: Percentage of blood leaving the heart at each contraction

high_blood_pressure: If the patient has hypertension (Boolean)

platelets: Platelet count of blood (kiloplatelets/mL)

serum_creatinine: Level of serum creatinine in the blood (mg/dL)

serum_sodium: Level of serum sodium in the blood (mEq/L)

sex: sex of the patient

smoking: If the patient smokes or not (Boolean)

time: Follow-up period (days)

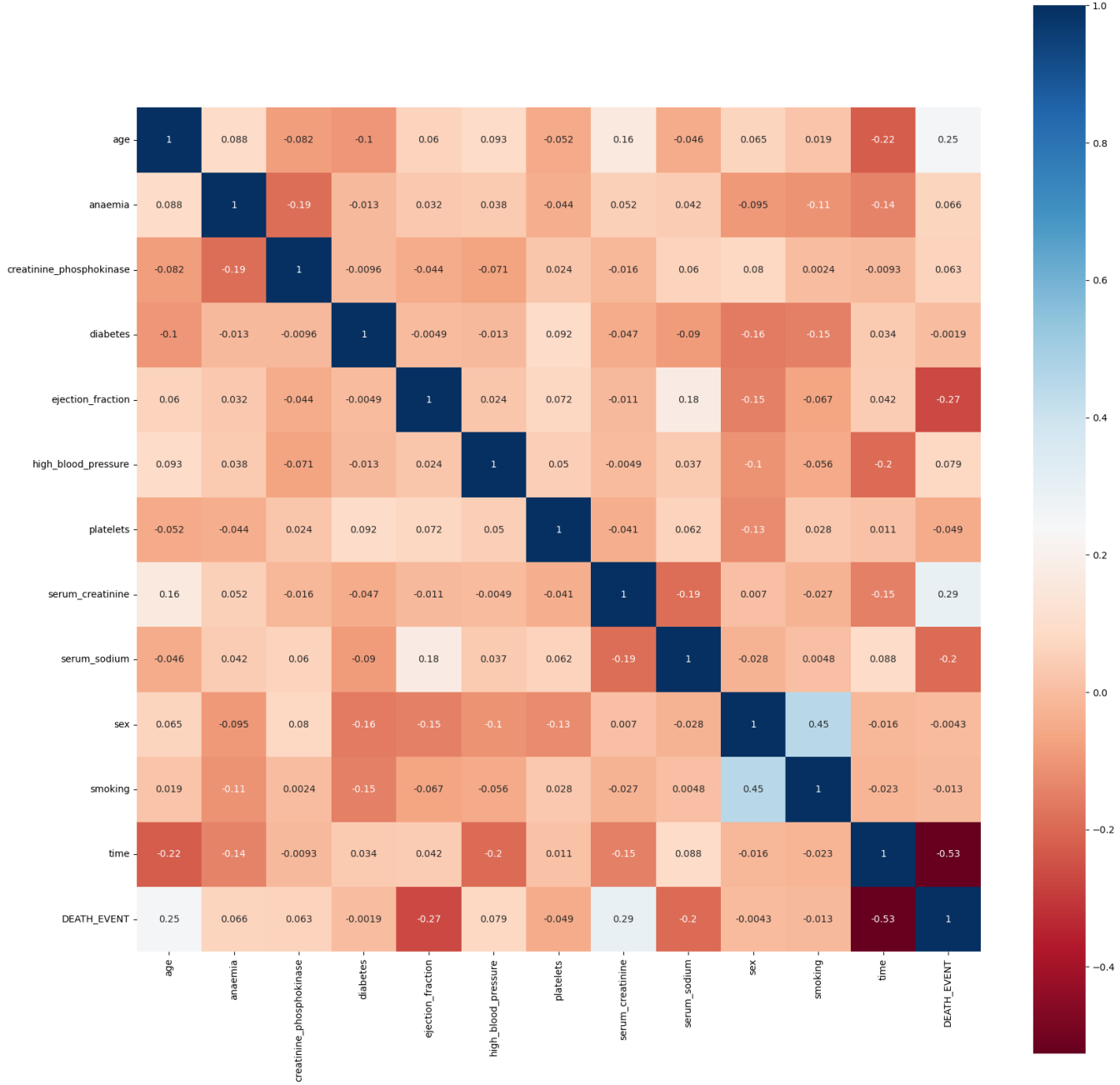
DEATH_EVENT: If the patient deceased during the follow-up period (Boolean)

[Boolean values: 0 = Negative (No); 1 = Positive (Yes)]

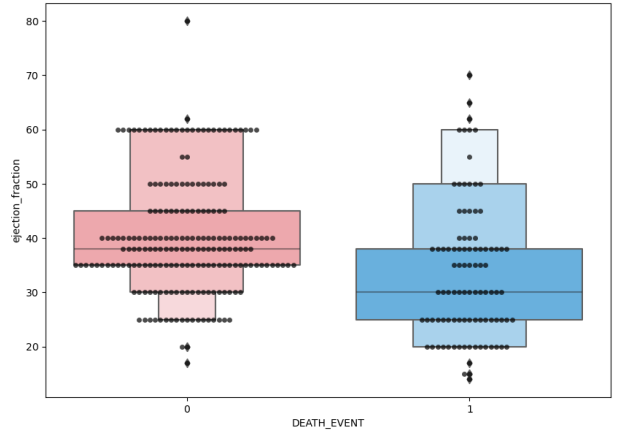
Univariate analysis

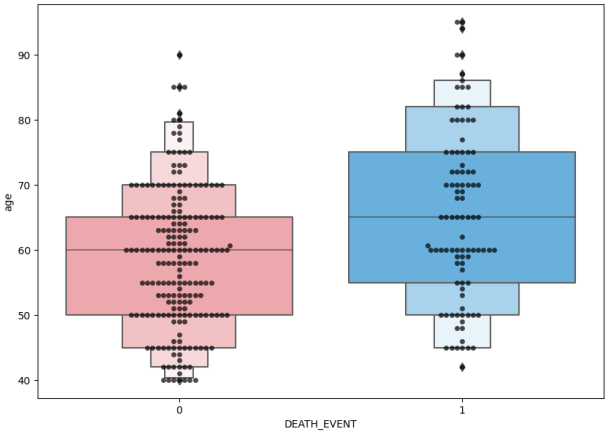
	count	mean	std	min	25%	50%	75%	max
age	299.0	60.833893	11.894809	40.0	51.0	60.0	70.0	95.0
anaemia	299.0	0.431438	0.496107	0.0	0.0	0.0	1.0	1.0
creatinine_phosphokinase	299.0	581.839465	970.287881	23.0	116.5	250.0	582.0	7861.0
diabetes	299.0	0.418060	0.494067	0.0	0.0	0.0	1.0	1.0
ejection_fraction	299.0	38.083612	11.834841	14.0	30.0	38.0	45.0	80.0
high_blood_pressure	299.0	0.351171	0.478136	0.0	0.0	0.0	1.0	1.0
platelets	299.0	263358.029264	97804.236869	25100.0	212500.0	262000.0	303500.0	850000.0
serum_creatinine	299.0	1.393880	1.034510	0.5	0.9	1.1	1.4	9.4
serum_sodium	299.0	136.625418	4.412477	113.0	134.0	137.0	140.0	148.0
sex	299.0	0.648829	0.478136	0.0	0.0	1.0	1.0	1.0
smoking	299.0	0.321070	0.467670	0.0	0.0	0.0	1.0	1.0
time	299.0	130.260870	77.614208	4.0	73.0	115.0	203.0	285.0
DEATH_EVENT	299.0	0.321070	0.467670	0.0	0.0	0.0	1.0	1.0

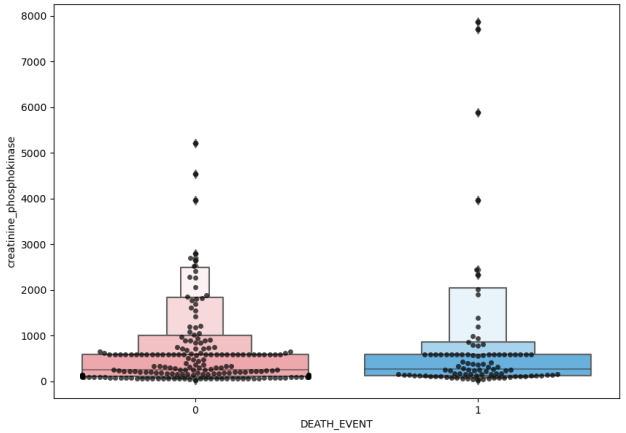
Bivariate analysis

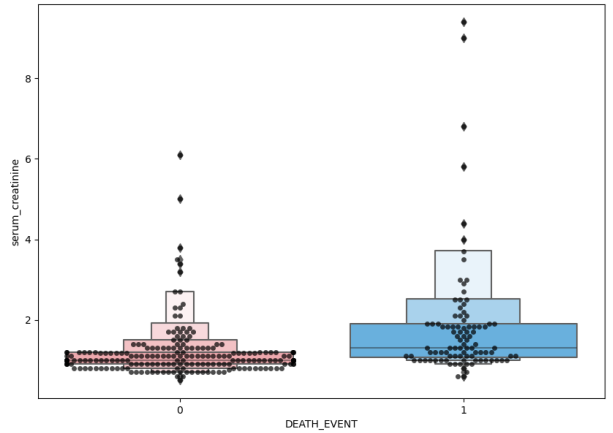


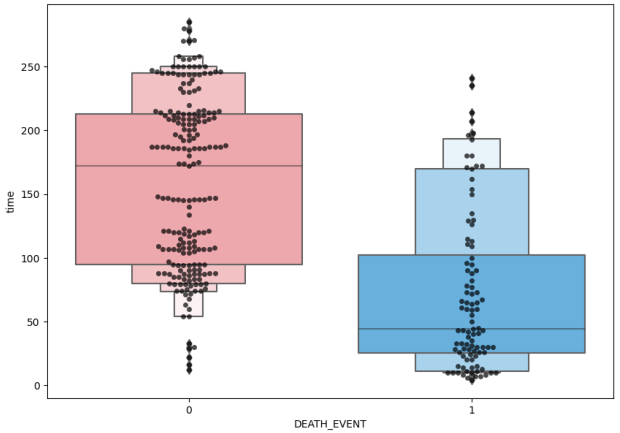
Outlier

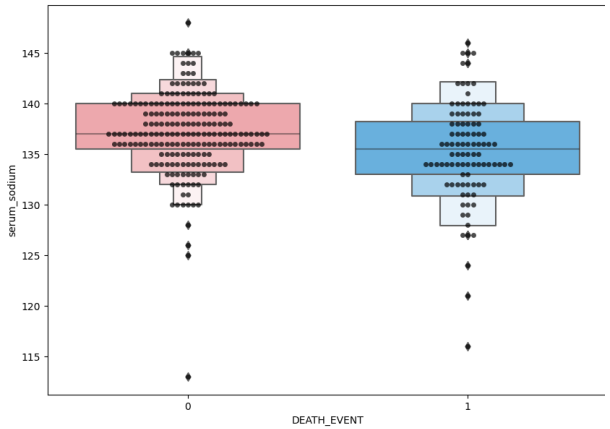

ejection_fraction
DEATH_EVENT

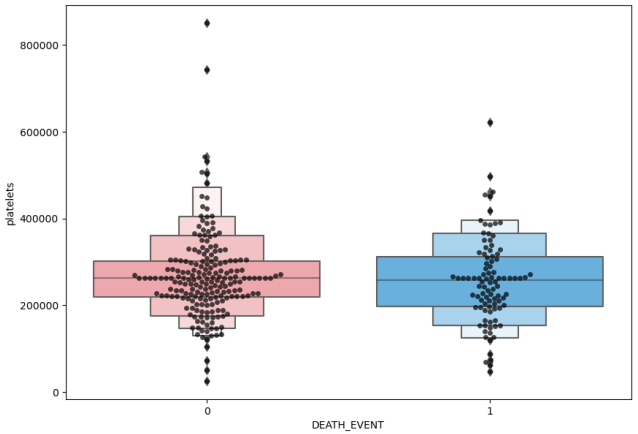

age
DEATH_EVENT


creatinine_phosphokinase
DEATH_EVENT

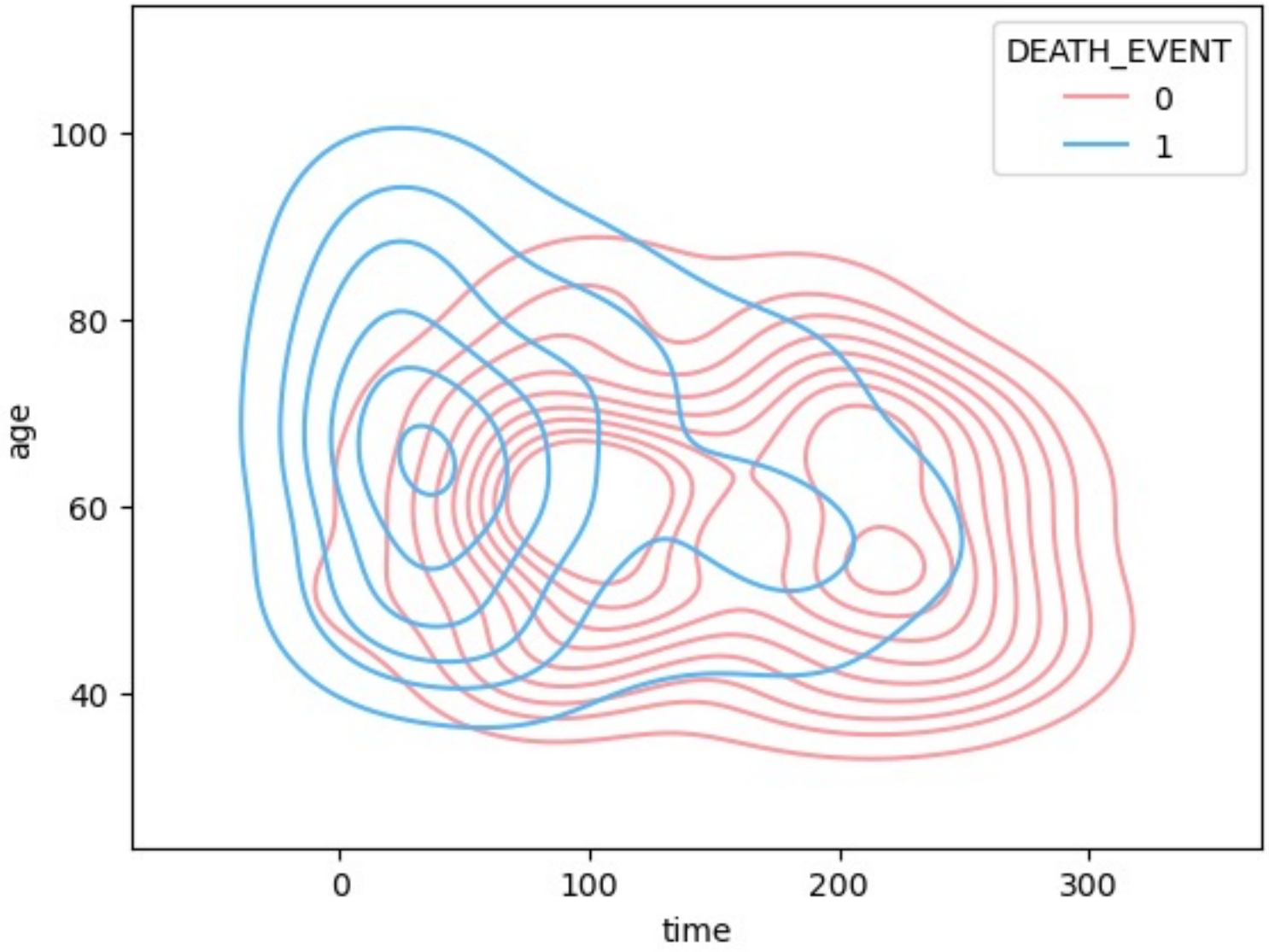

serum_creatinine
DEATH_EVENT


time
DEATH_EVENT

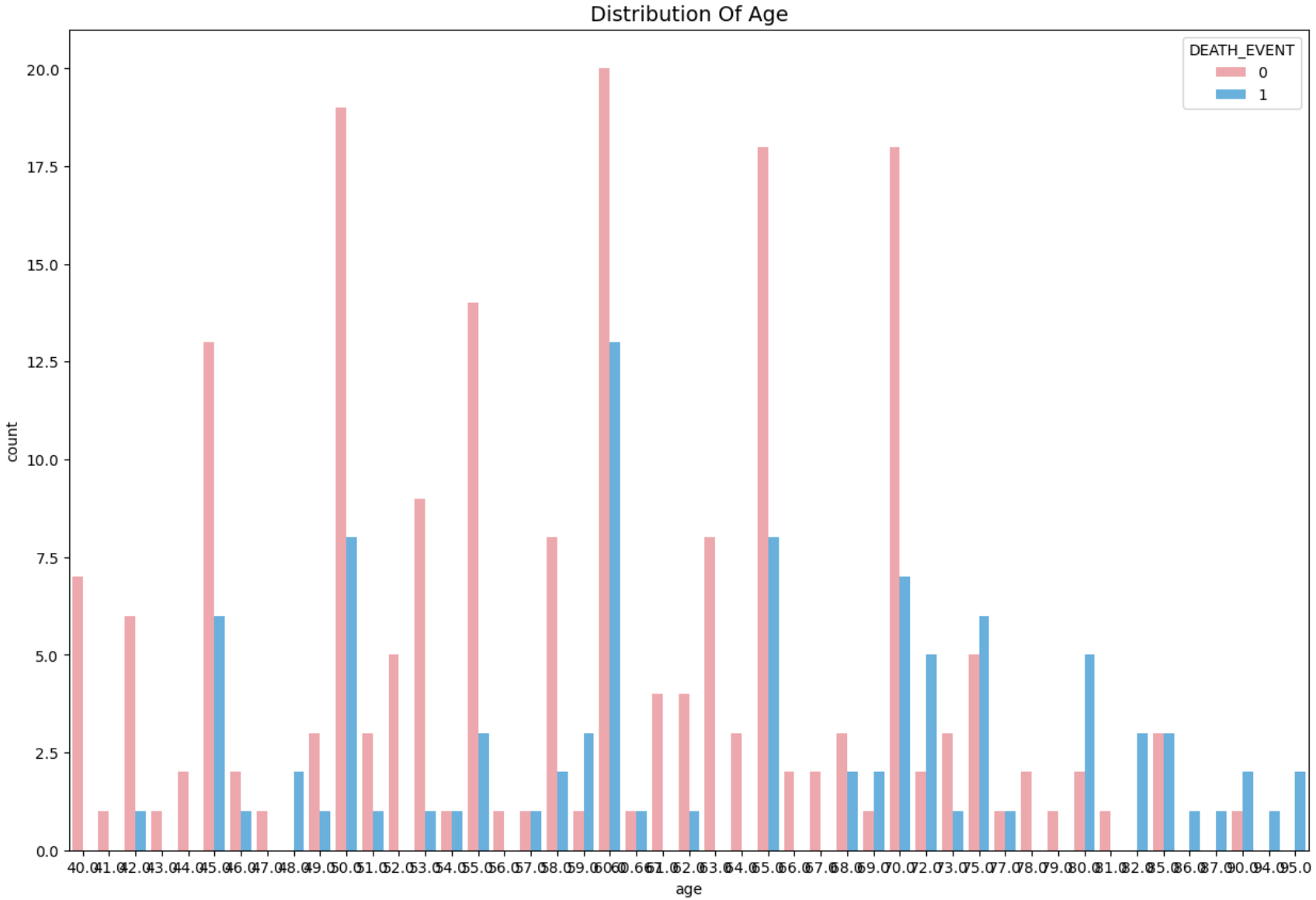

serum_sodium
DEATH_EVENT


platelets
DEATH_EVENT

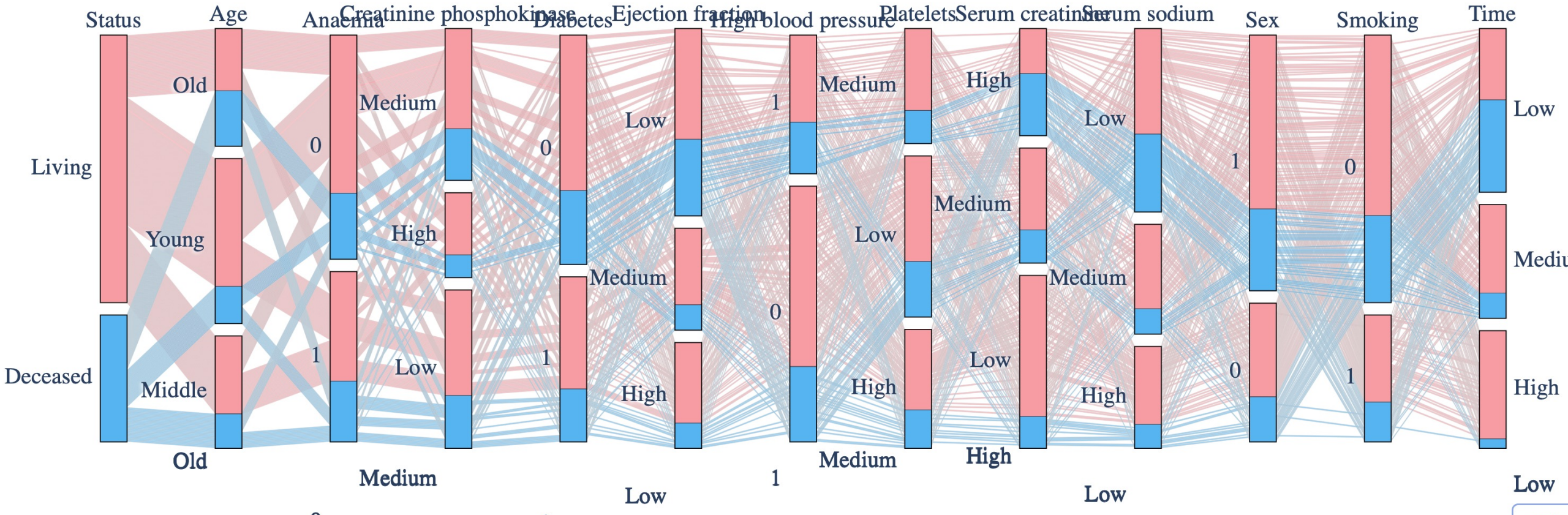
Age vs Checking frequency



Age vs Fatality



In general,...


Status
Age
Anaemia
Creatinine phosphokinase
Diabetes
Ejection fraction
High blood pressure
Platelets
Serum creatinine
Serum sodium
Sex
Smoking
Time
Living
Deceased
Old
Young
Middle
0
1
Medium
High
Low
Low
Medium
High
Medium
Low
High
Low
Medium
High
Low
Medium
High
1
0
1
0
1
0
1
0
Low
Medium
High

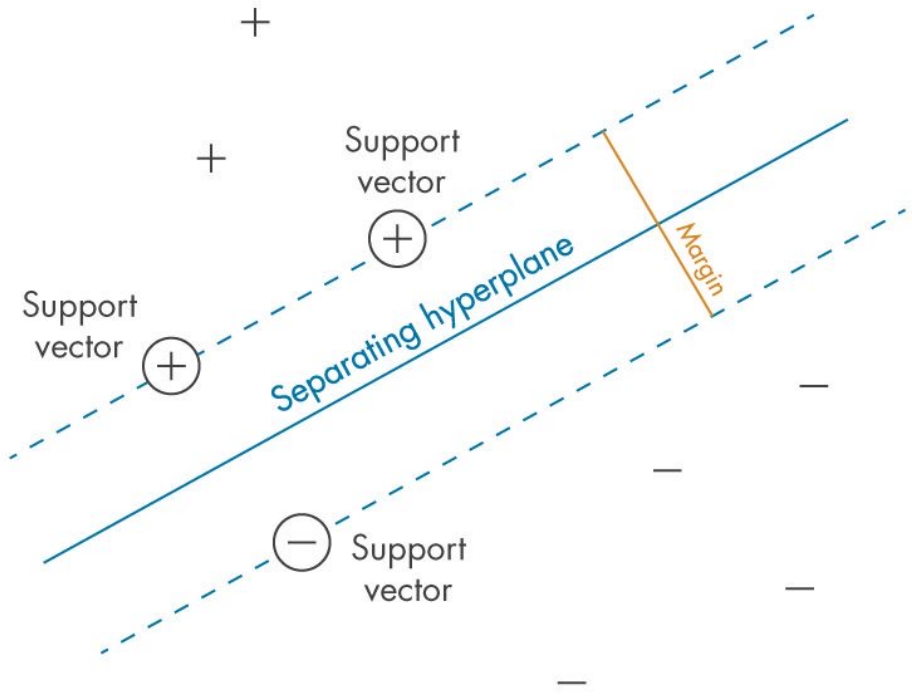
3

Implementation of Support Vector Machine

Overview

Method

Support Vector Machines (SVM)



Support Vectors are data points nearest to the decision boundary and are pivotal in defining the optimal location of the decision surface.

Hyperplane is the decision boundary that separates different classes.

Margin represents the distance between the decision boundary and the closest data points. SVM aims to maximize this distance.

Kernel Trick allows SVM to handle non-linear data by transforming it into a higher-dimensional space. Common kernels include polynomial, RBF, and sigmoid.

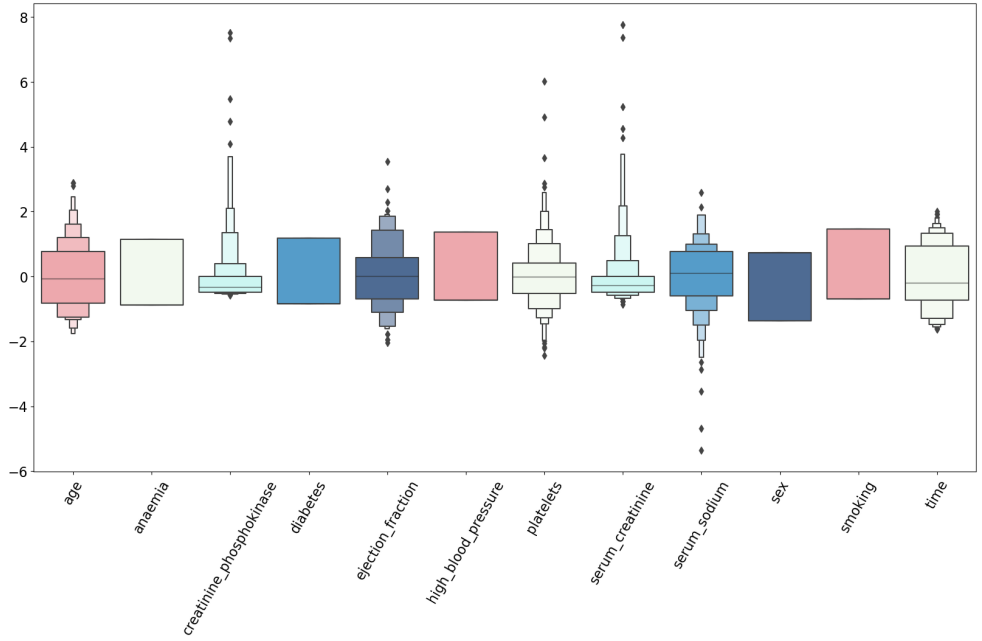
Method

1 – Setting up training & testing data.

```
x=dataset.drop(["DEATH_EVENT"],axis=1) #Independent
y=dataset["DEATH_EVENT"] #Dependent
```

2 – Configuring a standard scaler for the attributes and examining it subsequently.

```
s_scaler = preprocessing.StandardScaler()
x_scaled= s_scaler.fit_transform(x)
```



3 – Splitting variables into training and test set with ratio 70:30 with a seed for the random number generator.

4 – Apply SVM model for classification tasks.

Method

1 – Setting up training & testing data.

2 – Configuring a standard scaler for the attributes and examining it subsequently.

3 – Splitting variables into training and test set with ratio 70:30 with a seed for the random number generator.

```
x_train, x_test, y_train,y_test = train_test_split(x_scaled,y,
                                                   test_size=0.30,
                                                   random_state=25)
```

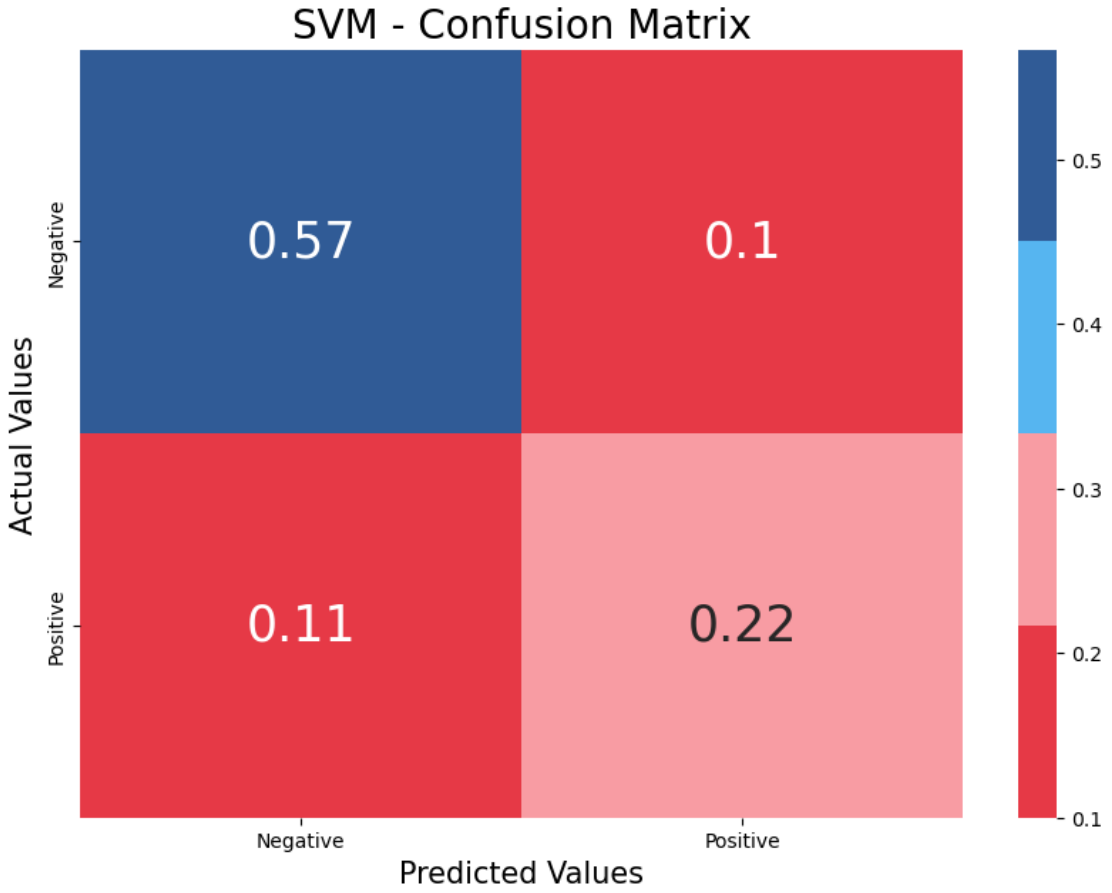
4 – Apply SVM model for classification tasks.

```
# Initialization
model_SVM=svm.SVC()

# Training the Model
model_SVM.fit (x_train, y_train)

# Performing prediction
y_pred = model_SVM.predict(x_test)
```

Method

Evaluation

1 – Confusion matrix



2 – Other metrics basing on confusion matrix

3 – ROC curve

Method

Evaluation

1 – Confusion matrix

2 – Other metrics basing on confusion matrix

 # Sensitivity, recall: The proportion of actual positive cases that were correctly identified.

$$TPR = \frac{TP}{TP+FN} = \frac{0.22}{0.22+0.11} = 0.667$$

 # Specificity: The proportion of actual negative cases that were correctly identified.

$$TNR = \frac{TN}{TN+FP} = \frac{0.57}{0.57+0.1} = 0.85$$

 # Precision: The proportion of the predicted positive & negative cases that were correct.

$$Precision = \frac{TP}{TP+FP} = \frac{0.22}{0.22+0.1} = 0.6875$$

$$NPV = \frac{TN}{TN+FN} \quad \frac{0.57}{0.57+0.11} = 0.8382$$

 # Accuracy: The overall proportion of correct predictions.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{0.22+0.57}{0.22+0.57+0.1+0.11} = 0.79$$

3 – ROC curve

Method

Evaluation

1 – Confusion matrix

2 – Other metrics basing on confusion matrix

```
(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
Living case - 0	0.84	0.85	0.84	60
Death case - 1	0.69	0.67	0.68	30
accuracy			0.79	90
macro avg	0.76	0.76	0.76	90
weighted avg	0.79	0.79	0.79	90

HIGH PERFORMANCE

3 – ROC curve

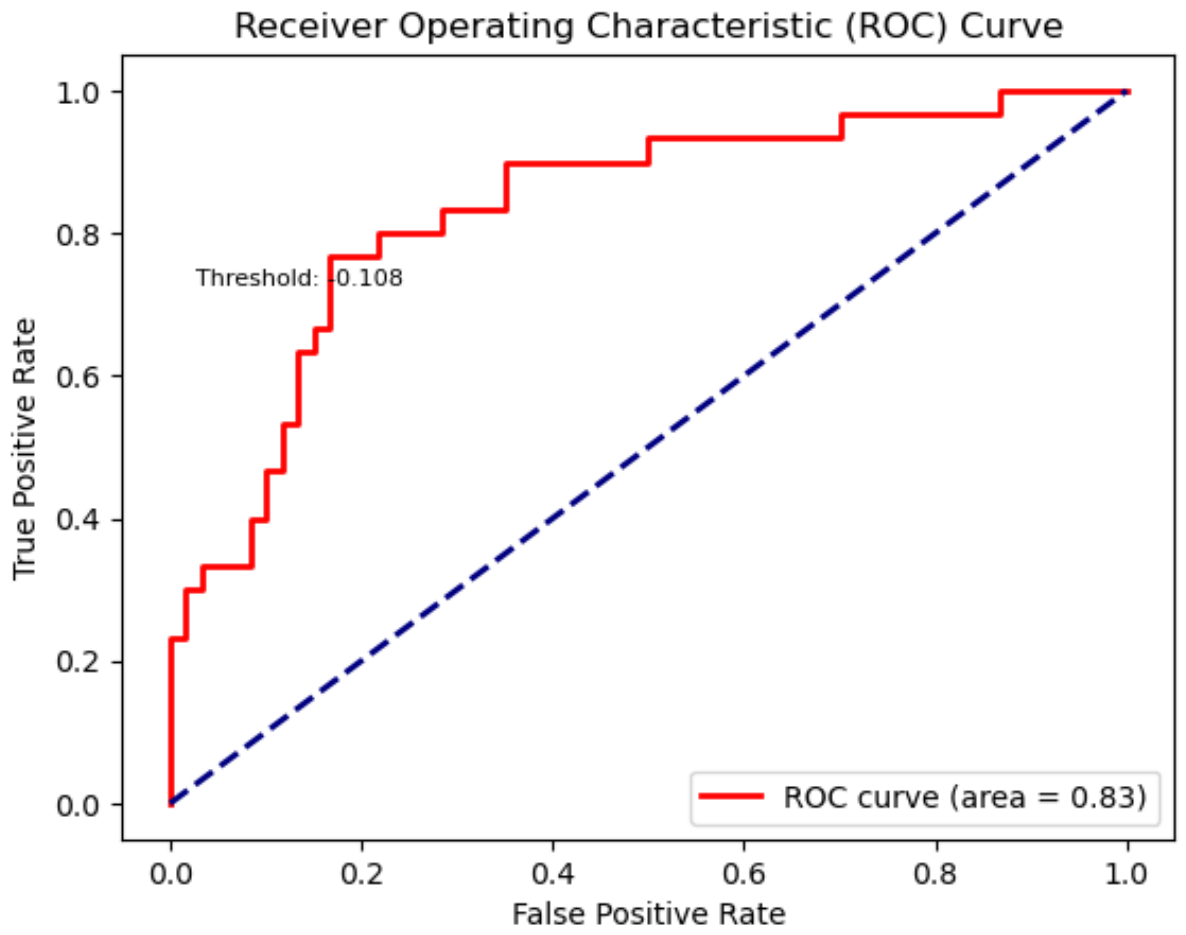
Method

Evaluation

1 – Confusion matrix

2 – Other metrics basing on confusion matrix

3 – ROC curve



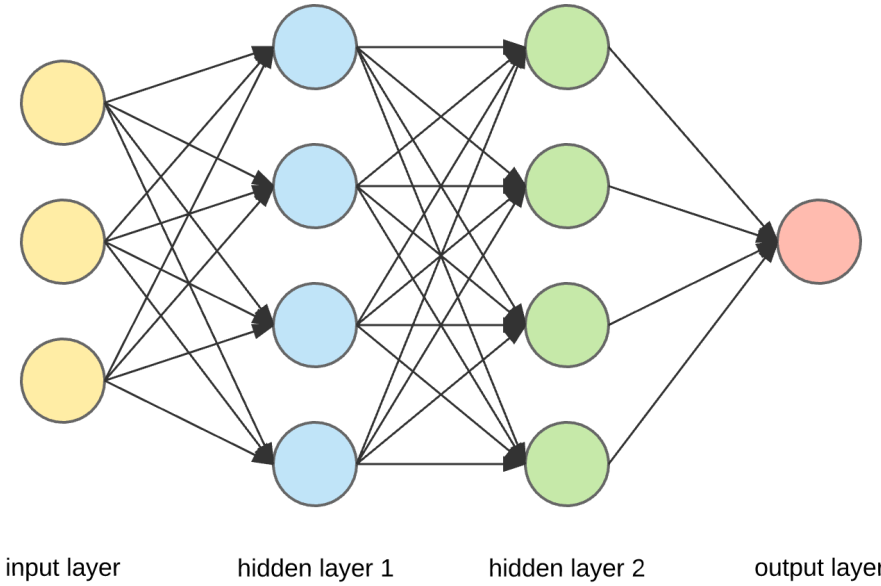
4

Implementation of Artificial Neural Network

Artificial Neural Network (ANN)

Overview

Method



Neuron: At its core, an ANN consists of artificial "neurons" or "nodes". Each neuron receives one or more inputs, processes it (usually applying a non-linear transformation), and produces an output.

Layers: Neurons are organized into layers:

- **Input Layer:** Represents feature inputs.
- **Hidden Layers:** Intermediate layers between input and output, where the actual processing happens. An ANN can have multiple hidden layers, leading to the term "deep" in deep learning when there are many layers.
- **Output Layer:** Produces the result for given inputs.

Method

1 – Setting up training & testing data.

Same as SVM model

2 – Configuring a standard scaler for the attributes and examining it subsequently.

Same as SVM model

3 – Splitting variables into training and test set with ratio 70:30 with a seed for the random number generator.

Same as SVM model

4 – Apply ANN model for classification tasks.

```
# Building the model
```

```
# Setting stopping resolution

stopping_res = callbacks.EarlyStopping(
    min_delta=0.001,
    # minimium amount of change to count as an improvement
    patience=30,
    restore_best_weights=True)
```

The **EarlyStopping** callback halts neural network training if the monitored metric doesn't improve by **at least 0.001 for 30 consecutive epochs**.

```
# Training the Model

# Performing prediction
```

Overview

Method

Evaluation

1 – Setting up training & testing data.

2 – Configuring a standard scaler for the attributes and examining it subsequently.

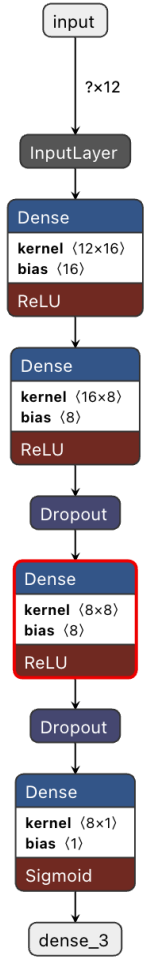
3 – Splitting variables into training and test set with ratio 70:30 with a seed for the random number generator.

4 – Apply ANN model for classification tasks.

Building the model

A simple feedforward neural network with three dense layers, two dropout layers to mitigate overfitting, and an output layer for binary classification.

```
model_ANN = Sequential()

model_ANN.add(Dense(units = 16, kernel_initializer = 'uniform',
                    activation = 'relu', input_dim = 12))
model_ANN.add(Dense(units = 8, kernel_initializer = 'uniform',
                    activation = 'relu'))
model_ANN.add(Dropout(0.25))
model_ANN.add(Dense(units = 8, kernel_initializer = 'uniform',
                    activation = 'relu'))
model_ANN.add(Dropout(0.5))
model_ANN.add(Dense(units = 1, kernel_initializer = 'uniform',
                    activation = 'sigmoid'))
```



Training the Model

Performing prediction

Overview

Method

Evaluation

1 – Setting up training & testing data.

2 – Configuring a standard scaler for the attributes and examining it subsequently.

3 – Splitting variables into training and test set with ratio 70:30 with a seed for the random number generator.

4 – Apply ANN model for classification tasks.

Building the model

Training the Model

The model will learn over **90 iterations** by batches of **30 samples** at a time of the entire dataset but will potentially stop early if the validation performance doesn't improve.

```
history_ANN = model_ANN.fit(x_train, y_train, batch_size = 30, epochs = 90,
                            callbacks=[stopping_res], validation_split=0.3)
```

```
Epoch 18/90
5/5 [==============================] - 0s 5ms/step - loss: 0.6663 - accuracy: 0.6301 - val_lo
ss: 0.6408 - val_accuracy: 0.8095
```

Performing prediction

```
y_pred = model_ANN.predict(x_test)
y_pred = (y_pred > 0.5)
```

Predicts class probabilities using the trained neural network, then classifies each test sample based on a 0.5 threshold.

Method

Evaluation

1 – Loss and Accuracy from Training and Validation

Loss

Accuracy

2 – Confusion matrix

3 – Other metrics basing on confusion matrix

4 – ROC curve

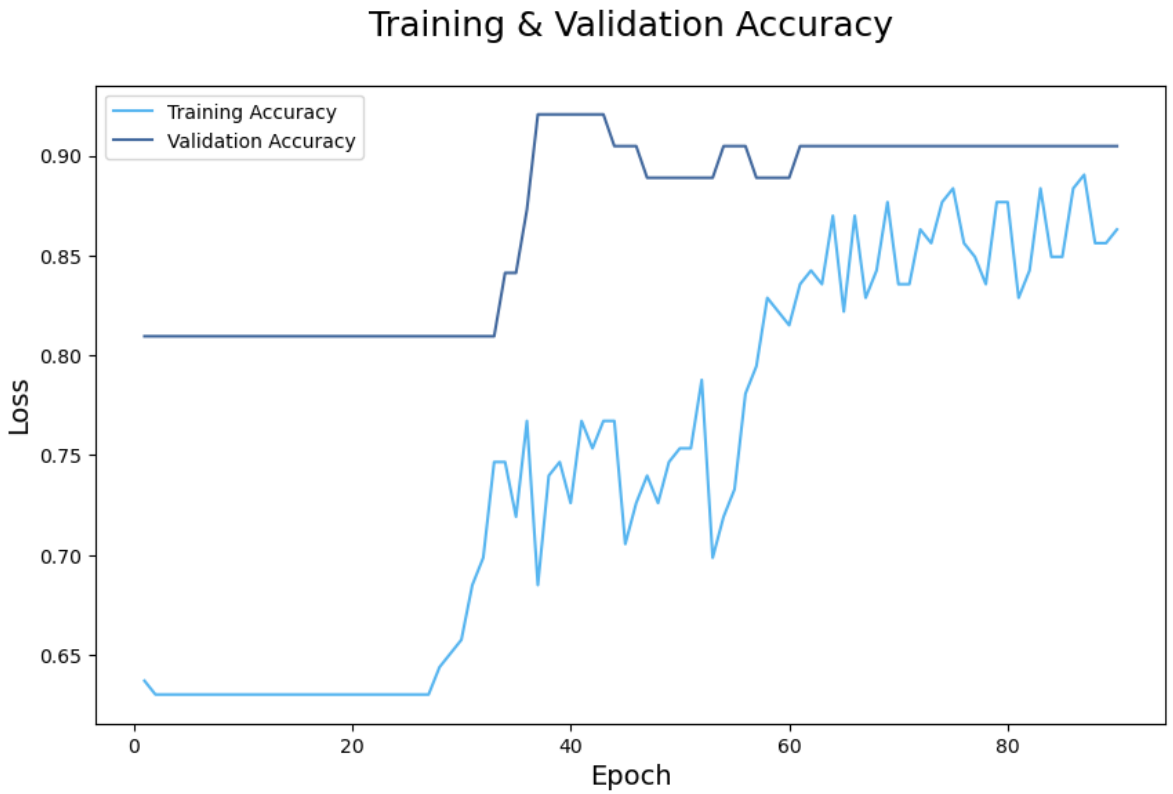
Method

Evaluation

1 – Loss and Accuracy from Training and Validation

Loss

Accuracy



2 – Confusion matrix

3 – Other metrics basing on confusion matrix

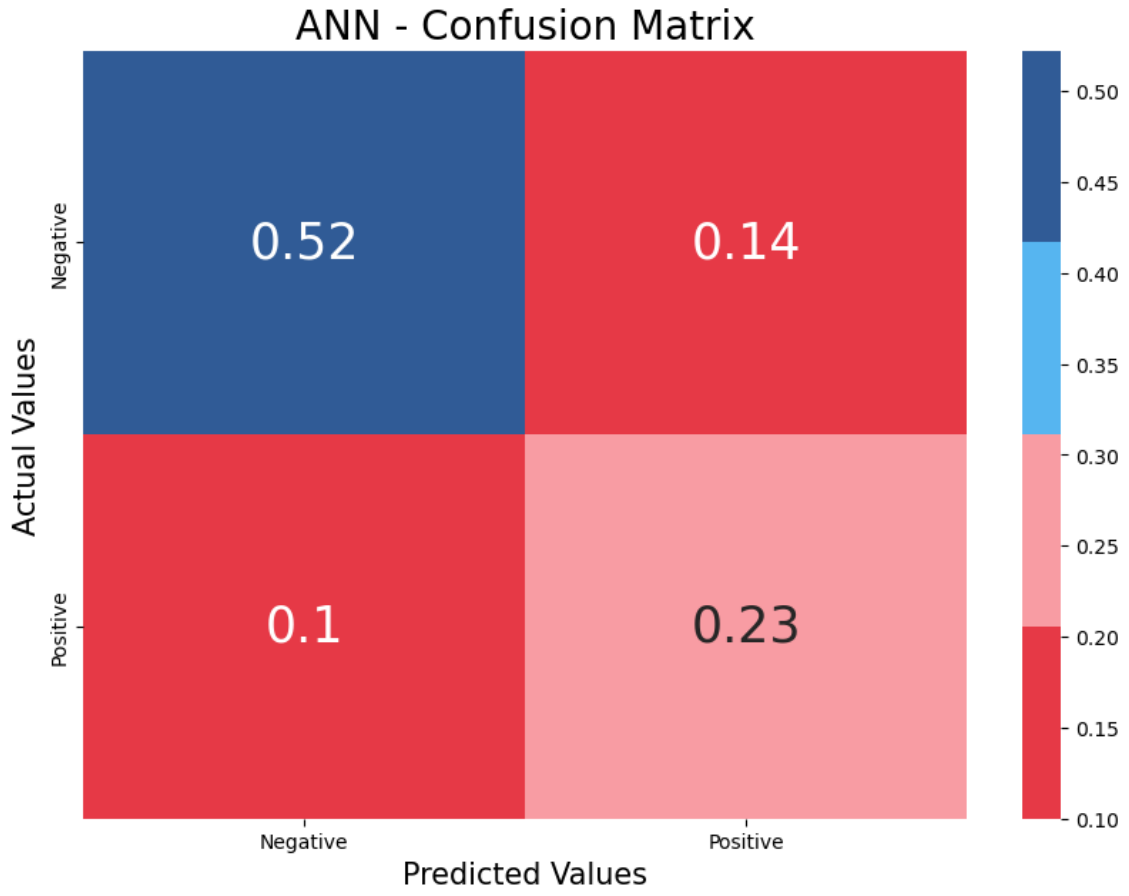
4 – ROC curve

Method

Evaluation

1 – Loss and Accuracy from Training and Validation

2 – Confusion matrix



3 – Other metrics basing on confusion matrix

4 – ROC curve

Method

Evaluation

1 – Loss and Accuracy from Training and Validation

2 – Confusion matrix

3 – Other metrics basing on confusion matrix

 # Sensitivity, recall: The proportion of actual positive cases that were correctly identified.

$$Sensitivity = \frac{TP}{TP+FN} = \frac{0.24}{0.24+0.10} = 0.706$$

 # Specificity: The proportion of actual negative cases that were correctly identified.

$$Specificity = \frac{TN}{TN+FP} = \frac{0.52}{0.52+0.14} = 0.788$$

 # Precision: The proportion of the predicted positive & negative cases that were correct.

$$Precision = \frac{TP}{TP+FP} = \frac{0.24}{0.24+0.14} = 0.632 \quad NPV = \frac{0.52}{0.52+0.10} = \frac{0.52}{0.62} = 0.839$$

 # Accuracy: The overall proportion of correct predictions.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{0.24+0.52}{0.24+0.52+0.14+0.10} = 0.76$$

3 – ROC curve

Method

Evaluation

1 – Loss and Accuracy from Training and Validation

2 – Confusion matrix

3 – Other metrics basing on confusion matrix

```
(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
Living case - 0	0.84	0.78	0.81	60
Death case - 1	0.62	0.70	0.66	30
accuracy			0.76	90
macro avg	0.73	0.74	0.73	90
weighted avg	0.77	0.76	0.76	90

HIGH PERFORMANCE

3 – ROC curve

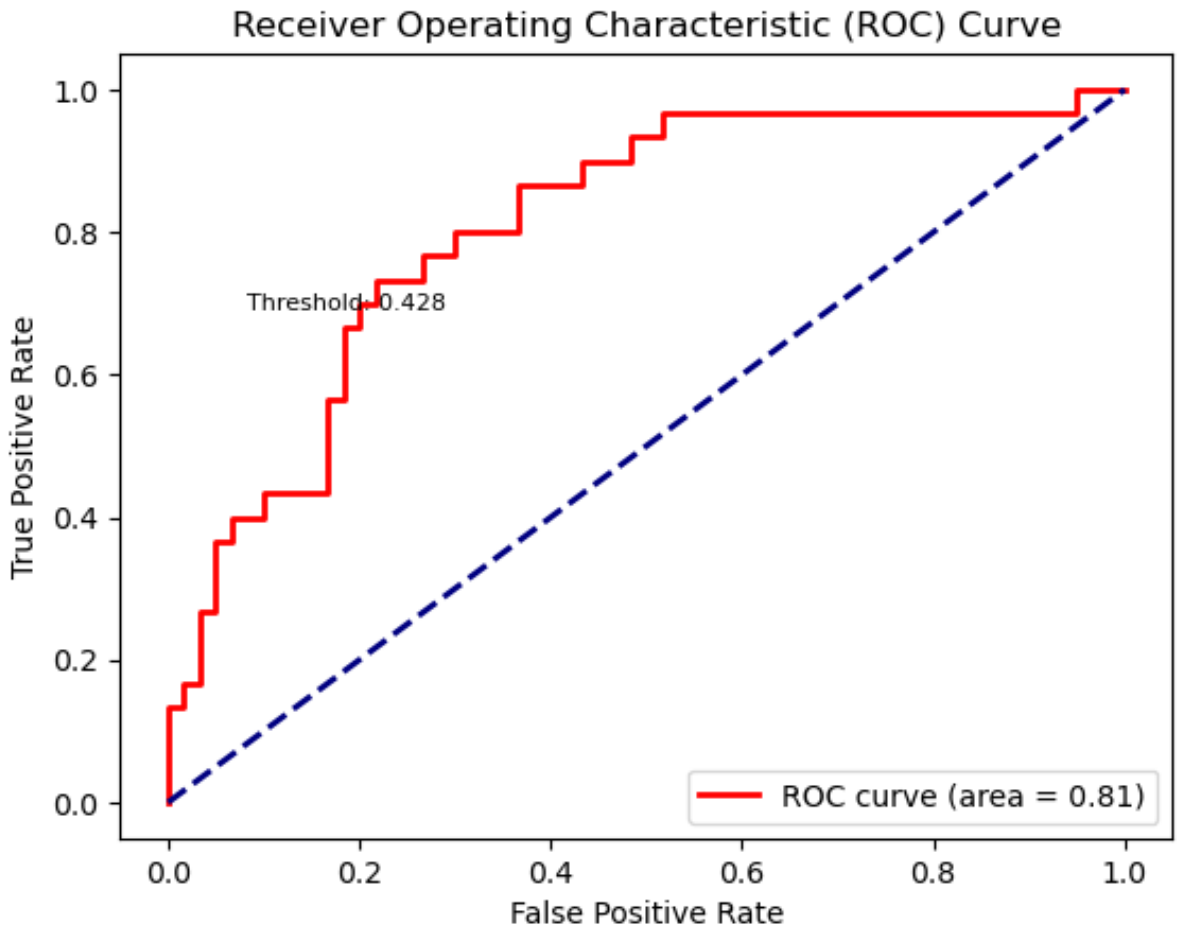
Method

Evaluation

1 – Confusion matrix

2 – Other metrics basing on confusion matrix

3 – ROC curve



5

Conclusion

Model Performance	SVM Class 0 – 1	ANN Class 0 – 1
Precision	0.84 – 0.69	0.84 – 0.62
Recall	0.85 – 0.67	0.78 – 0.70
F1-score	0.84 – 0.68	0.81 – 0.66
Accuracy	0.79	0.76
ROC curve AUC	0.83	0.81

While both models exhibit competitive performance metrics, the SVM slightly edges out the ANN in this specific dataset, based on accuracy, precision, and the ROC AUC for this dataset

Model Performance	SVM Class 0 – 1	ANN Class 0 – 1
Precision	0.84 – 0.69	0.84 – 0.62
Recall	0.85 – 0.67	0.78 – 0.70
F1-score	0.84 – 0.68	0.81 – 0.66
Accuracy	0.79	0.76
ROC curve AUC	0.83	0.81

While both models exhibit competitive performance metrics, the SVM slightly edges out the ANN in this specific dataset, based on accuracy, precision, and the ROC AUC for this dataset

Future consideration

Model selection depends not just on performance metrics but also on hyperparameter choices, as tweaking them can affect results. While metrics are vital, other factors, including the model's interpretability, computational demands, training duration, and specific application needs, play a role in the decision.

THANK YOU!

MAIB23 – Machine Learning

Farkhod Muradov

Tran Khanh Ngoc Le