REPORT
Biologically inspired artificial intelligence Heart Attack Disease Prediction

Lecturer: Dr inż. Grzegorz Baron

Date: 20.06.2023

Adrian Piróg
Jakub Łapaj

Short introduction presenting the project topic:

Heart disease is a significant health concern globally, and early detection plays a crucial role in improving patient outcomes. In this project, our goal is to develop a heart disease prediction model using machine learning algorithms and neural networks.

Analysis of the task:

1. Possie approaches to solve the problem with its pros/cons, detailed description of selected methodology:
   Our problem is based on analysing provided dataset that includes specific types and values. We have decided to use and compare machine learning models: Logistic Regression, Decision Tree, Support Vector Classifier and ANN(Artificial Neural Network). The pros of using models is the more efficient and faster learning on smaller datasets. Neural networks take more time and to work efficiently it should be provided a larger dataset.

2. Presentation of possible/available datasets and detailed description of the chosen ones:
   We base on .csv dataset which includes parameters:
   - Age
   - Sex
   - Chest Pain Type (cp)
   - Resting Blood Pressure (trtbps)
   - Serum Cholesterol (chol)
   - Fasting Blood Sugar (fbs)
   - Resting Electrocardiographic Results (restecg)
   - Maximum Heart Rate Achieved (thalachh)
   - Exercise Induced Angina (exang)
   - ST Depression Induced by Exercise Relative to Rest (oldpeak)
   - The Slope of The Peak Exercise ST Segment (slp)
   - Number of Major Vessels Colored by Flourosopy (caa)
   - Thallium Stress Test (thall)
   - Output (Diagnosis of Heart Disease)

	A	B	C	D	E	F	G	H
1	age,sex,cp,trtbps,chol,fbs,restecg,thalachh,exng,oldpeak,slp,caa,thall,output							
2	63,1,3,145,233,1,0,150,0,2.3,0,0,1,1							
3	37,1,2,130,250,0,1,187,0,3.5,0,0,2,1							
4	41,0,1,130,204,0,0,172,0,1.4,2,0,2,1							
5	56,1,1,120,236,0,1,178,0,0.8,2,0,2,1							
6	57,0,0,120,354,0,1,163,1,0.6,2,0,2,1							
7	57,1,0,140,192,0,1,148,0,0.4,1,0,1,1							
8	56,0,1,140,294,0,0,153,0,1.3,1,0,2,1							
9	44,1,1,120,263,0,1,173,0,0,2,0,3,1							
10	52,1,2,172,199,1,1,162,0,0.5,2,0,3,1							
11	57,1,2,150,168,0,1,174,0,1.6,2,0,2,1							
12	54,1,0,140,239,0,1,160,0,1.2,2,0,2,1							
13	48,0,2,130,275,0,1,139,0,0.2,2,0,2,1							
14	49,1,1,130,266,0,1,171,0,0.6,2,0,2,1							
15	64,1,3,110,211,0,0,144,1,1.8,1,0,2,1							
16	58,0,3,150,283,1,0,162,0,1,2,0,2,1							
17	50,0,2,120,219,0,1,158,0,1.6,1,0,2,1							
18	58,0,2,120,340,0,1,172,0,0,2,0,2,1							

Fig. 1 Dataset

For our project we have used a dataset available at:
https://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-prediction-dataset

3. Analysis of available tools/frameworks/libraries suitable for task solution; detailed presentation of selected tools/approach:

In our project we have decided to use models from the sklearn library. Specifying neural networks the choice aimed the Tensorflow library with Keras interface.

Logistic Regression - statistical model used for binary classification, where it predicts the probability of an instance belonging to a certain class based on input features. It utilizes a logistic function (sigmoid) to map the output to a probability value between 0 and 1.

Support Vector Classifier - is a machine learning model that is commonly used for classification tasks. It aims to find an optimal hyperplane in a high-dimensional space that best separates different classes of data points, maximizing the margin between them. Decision Trees - A decision tree is a model that recursively partitions the data into subsets based on feature values, making predictions at the leaf nodes. It uses a series of if-else conditions to traverse the tree and determine the final outcome.

Artificial Neural Networks (ANN) - computational models inspired by the structure and functioning of biological neural networks in the brain. They consist of interconnected nodes, or neurons, organized in layers, which process and transmit information. By adjusting the connection weights, ANN can learn and make predictions, making them a powerful tool for various machine learning tasks such as classification, regression, and pattern recognition.

We worked with Jupyter Notebook provided by Anaconda Navigator to compile the project.

Internal and external specification of the software solution:

1. Classes/objects/methods/scripts/functions etc. – depending on the approach to the project solution + data structures (a, b):

 # Load the dataset
df = pd.read_csv("kaggle/input/heart.csv") df.head()

Then we mapped the values to respective categories, created a copy of the dataset for the further processing. Categorical columns are transformed into dummy variables, numerical columns are scaled using StandardScaler, and the correlation matrix is calculated and visualized using a heatmap.

 # Split data into features (X) and
target (y) X = data.drop('output',
axis=1) y = data['output']

 # Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

The performance of each model have been
compared
 # Evaluate and compare the performance of each
model for model in models:

```
train_acc, test_acc, y_pred = evaluate_model(model, X_train, y_train, X_test,
y_test)  train_results.append(train_acc)  test_results.append(test_acc)  cm
= confusion_matrix(y_test, y_pred)
plot_confusion_matrix(cm, classes=['No Disease', 'Disease'],
title=model.__class__.__name__)
plt.show()

# Evaluate and print the performance of Logistic Regression CV
train_acc_lr, test_acc_lr, y_pred_lr = evaluate_model(lr, X_train, y_train, X_test,
y_test) print("Logistic Regression CV:") print("Train Accuracy:", train_acc_lr)
print("Test Accuracy:", test_acc_lr)

# Evaluate and print the performance of SVM with Grid Search CV
train_acc_svm, test_acc_svm, y_pred_svm = evaluate_model(svm, X_train, y_train,
X_test, y_test)
print("SVM (Support Vector Machine) with Grid Search CV:")
print("Train Accuracy:", train_acc_svm)
print("Test Accuracy:", test_acc_svm)

# Evaluate and print the performance of Decision Tree with Randomized Search CV
train_acc_dtc, test_acc_dtc, y_pred_dtc = evaluate_model(dtc, X_train, y_train, X_test,
y_test) print("Decision Tree with Randomized Search CV:")
print("Train Accuracy:", train_acc_dtc)
print("Test Accuracy:", test_acc_dtc)
```

After dealing with the models we have focused on building a sequential neural network model with Tensorflow and evaluating it's performance:

```
model = keras.Sequential(
  [
    keras.layers.Dense(
      256, activation="relu", input_shape=[23]
    ),
    keras.layers.Dense(515, activation="relu"),
keras.layers.Dropout(0.3),
    keras.layers.Dense(50, activation="relu"),
keras.layers.Dropout(0.3),
    keras.layers.Dense(1, activation="sigmoid"),
  ]
)
```

```
model.summary()

model.compile(optimizer="Adam", loss="binary_crossentropy",
metrics=["binary_accuracy"])

early_stopping = keras.callbacks.EarlyStopping(
   patience=20, min_delta=0.001, restore_best_weights=True
)
history =
model.fit(
X_train,
y_train,
validation_data=(
X_test, y_test),
batch_size=15,
epochs=50,
callbacks=[early_
stopping],
   verbose=1,
)

model.evaluate(X_test, y_test)
```

We computed the loss value and specified metrics of the trained model on the test data.

2. User interface / GUI/console

While working with jupyter notebook we were using the input console which returns the models, plotted diagrams etc. Right after we write intended code like in the example below:

```
for model in models:
    train_acc, test_acc, y_pred = evaluate_model(model, X_train, y_train, X_test, y_test)
    train_results.append(train_acc)
    test_results.append(test_acc)
    cm = confusion_matrix(y_test, y_pred)
    plot_confusion_matrix(cm, classes=['No Disease', 'Disease'], title=model.__class__.__name__)
    plt.show()
```

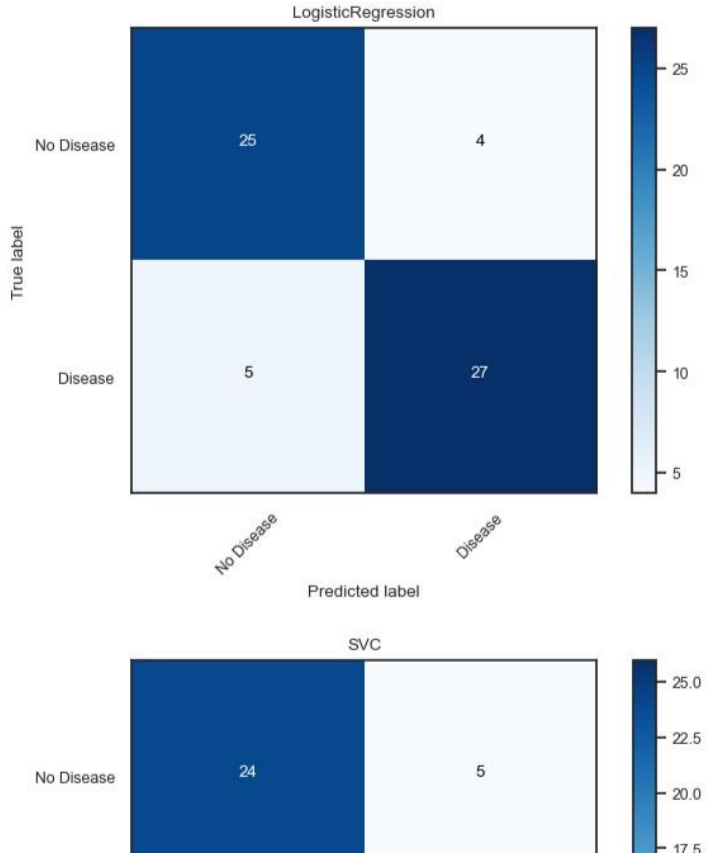

Fig. 2 Confusion matrix example

Experiments:

While we experimented with our trained program we were comparing different ways to create and train machine learning models. As we can see below the returned information was slightly differing between single plots. After training three distinct machine learning models, we can conclude that the data is reliable, but with slight variations depending on the chosen model.

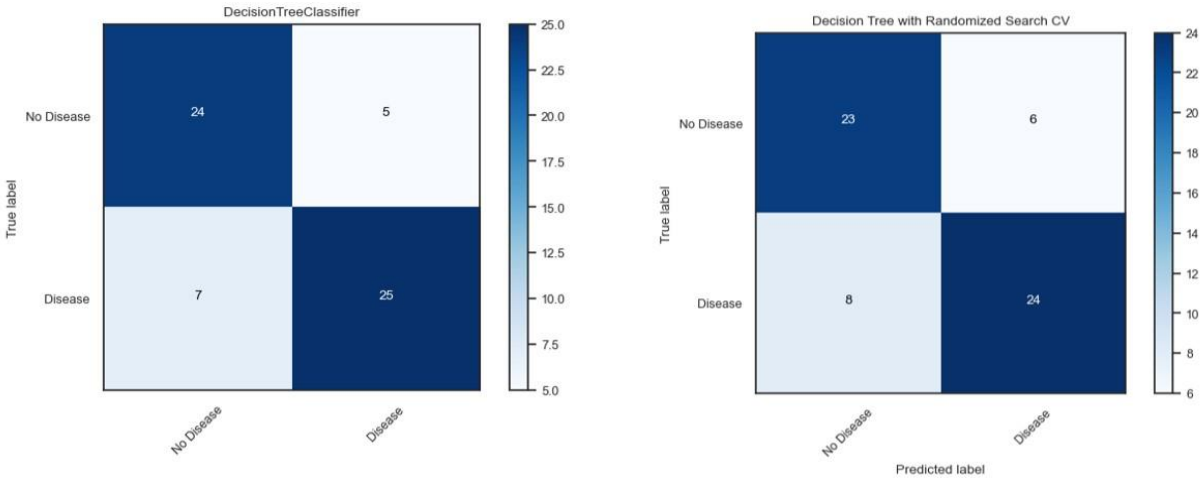

Fig. 3 Decision Trees confusion matrix

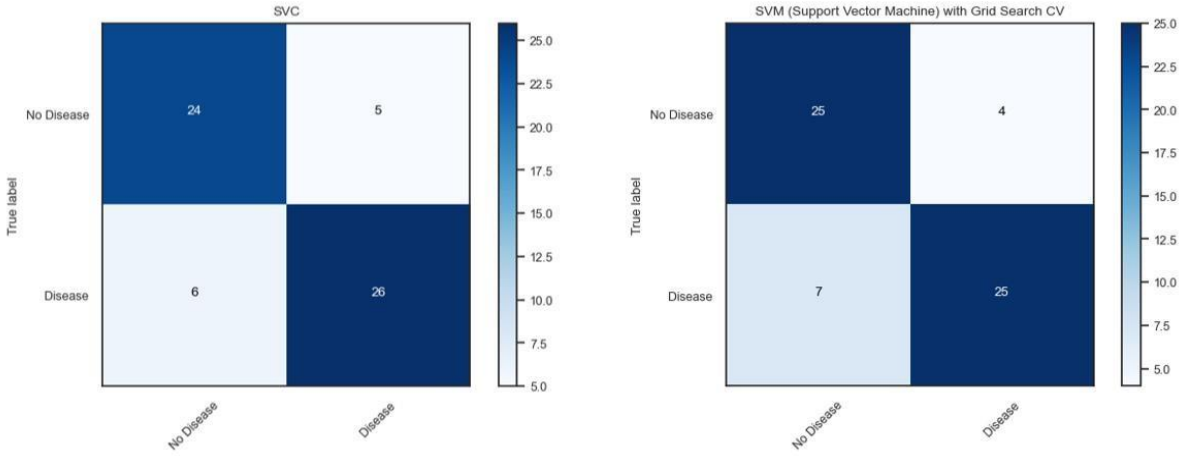

Fig. 4 SVC confusion matrix

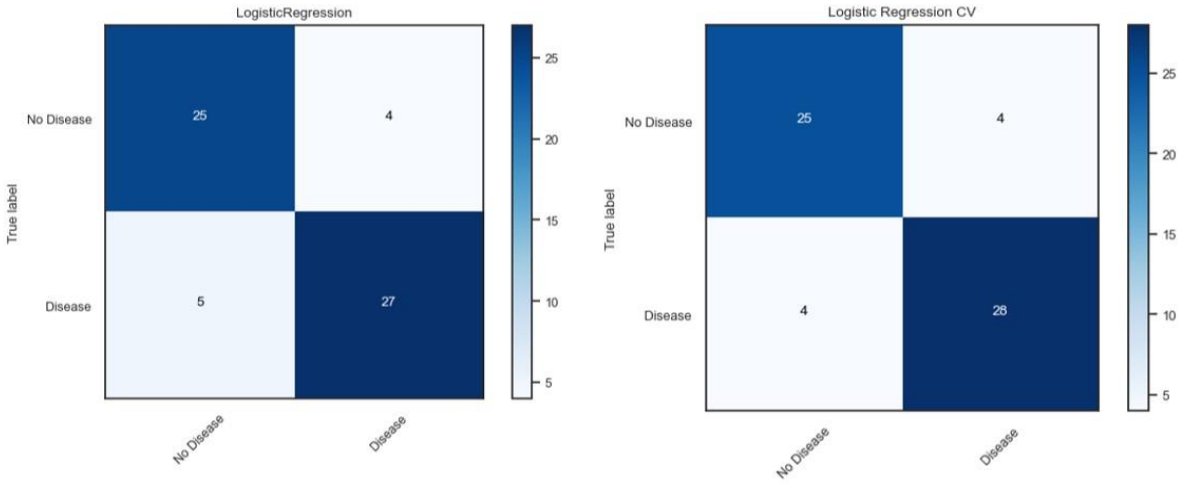

Fig. 5 Logistic Regression confusion matrix

We have plotted the same type of confusion matrix for the trained ANN:

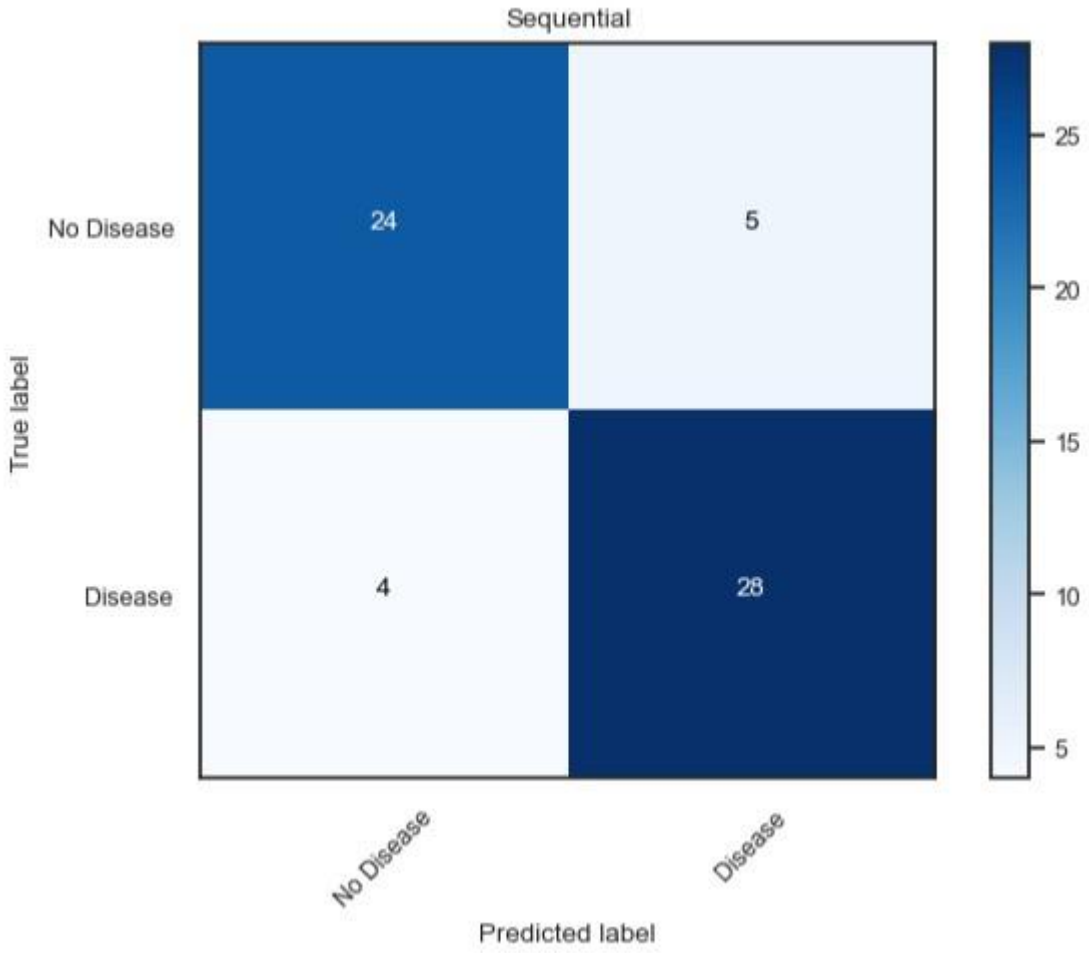

Fig. 6 ANN confusion matrix

We have also compared the accuracy of each model and neural network:

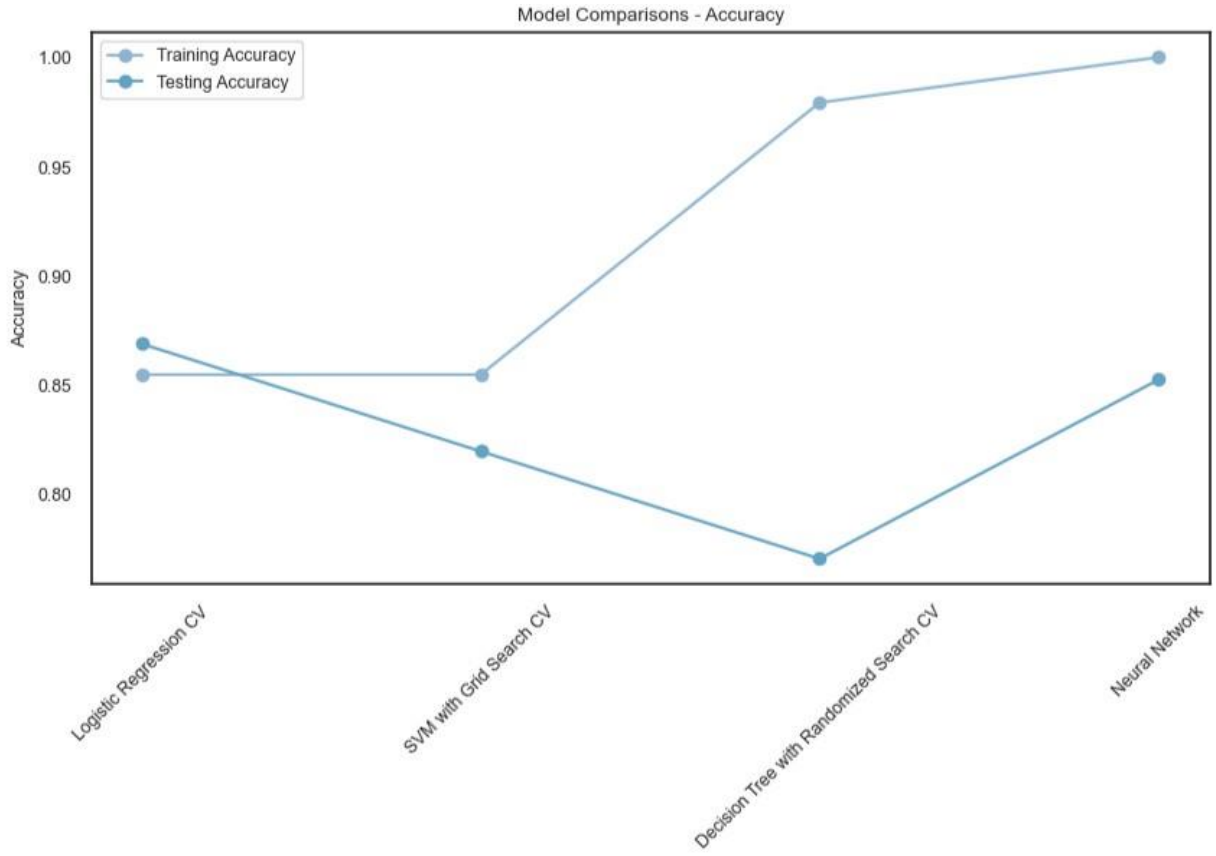

Fig. 7 All models accuracy compared

Conclusions:

While the models showed variations in training and testing accuracies, the Linear Regression model performed the best in terms of accuracy during the testing phase. On the other hand, the Decision Tree model achieved the highest accuracy during the training phase. Interestingly, the neural network quickly reached its maximum performance within just 20 epochs during the training phase. After repeating the training three times, the sequential neural network achieved over 99% accuracy in the learning phase. However, achieving nearly 100% accuracy on the training data does not necessarily mean that the model will perform well on test data. In fact, it often indicates overfitting, where the model becomes too specialized to the training data and fails to generalize well to new examples.

Based on these findings, we can conclude that neural networks perform exceptionally well whendealing with large datasets. They demonstrate a remarkable ability to quickly learn and encompass all potential outcomes but it can also be observed that neural networks have a

tendency to overfit when trained on small datasets, meaning they may become too specialized to the training data.

References – list of sources used during the work on the project:

- www.kaggle.com
- https://towardsdatascience.com/all-machine-learning-models-explained-in-6-minutes9fe30ff6776a
- https://en.wikipedia.org/wiki/Machine_learning
- https://www.javatpoint.com/logistic-regression-in-machine-learning
- https://pythonprogramming.net/linear-svc-example-scikit-learn-svm-python/
- https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

Link to the shared cloud/Google Drive/OneDrive/Git/etc. where all the files are placed:

https://github.com/AdrianAEI/BIAI-Heart-Attack-Prediction