Casapao, Jinky S.
Niosco, Kyle Anthony F.
Santiago, Maria Eloisa O.
III – BSCS STDS

December 22, 2021
Prof. Roselyn C. Maaño
Data Science Fundamentals

Classification Algorithm on Statlog Heart Disease Prediction using Decision Tree Machine Learning

Introduction

In predicting categorical values, the use of a Classification algorithm is necessary. The Classification algorithm is a supervised learning approach that uses training data to identify the category of new observations. A program in Classification learns from a given dataset or observations and then classifies additional observations into one of many classes or groupings (Classification Algorithm in Machine Learning - Javatpoint, 2021). In this paper, the main objectives include providing relevant information from the raw data and predicting decision whether a person acquired heart disease or not. Thus, a decision tree algorithm will be used. In this supervised learning technique, a model is created that helps anticipate the value of a target variable depending on input values (Akpan & Starkey, 2021). The information in this study will help determine the presence or absence of heart disease in a person. In addition, this could also assist in providing prescriptions and possible treatments to cure the particular heart disease.

Problem Statement

This paper seeks to predict decisions about the presence or absence of Statlog heart disease in a person. This also covers the factors that will impact analyzing and determining the prediction. In line with this, the following are the objectives of this paper:

- To prepare the stat log heart disease data set to fit in the model
- To generate a model that could formulate decisions about the presence or absence of heart disease
- To evaluate the results of the model generated
- To plot the evaluated results of the model

Methodology

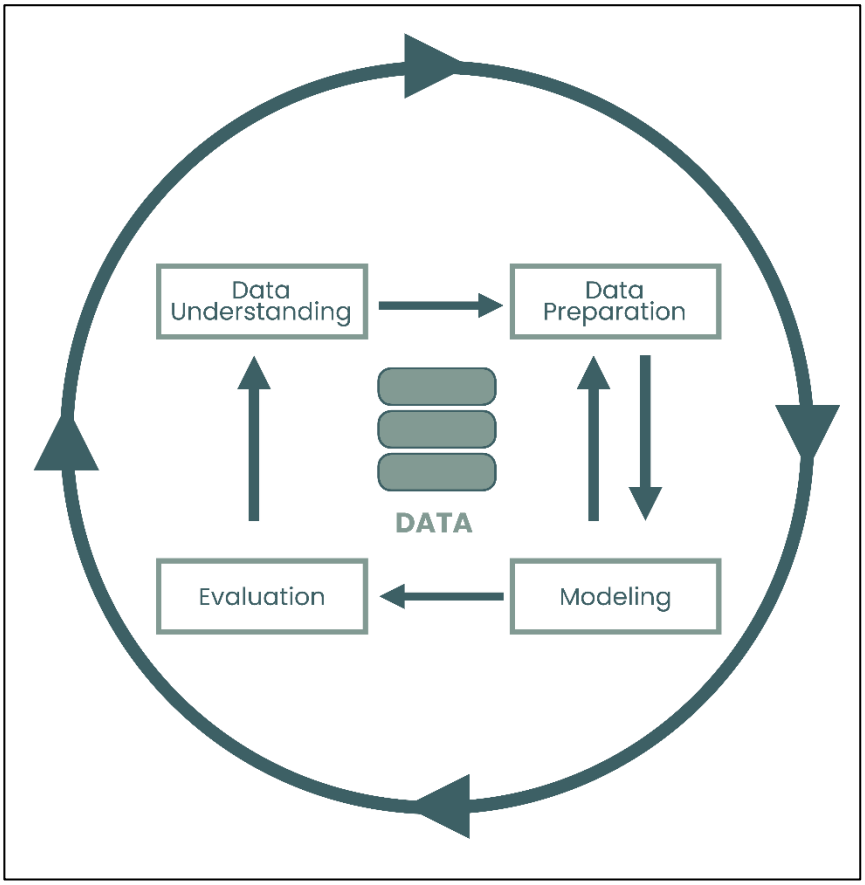

Figure 1. The Modified CRISP-DM Model

Figure 1 illustrates the flow of the phases that this model undergoes. In this paper, the Cross-Industry Standard Process for Data Mining or CRISP-DM was utilized to describe the phases in this data science life cycle. However, this process model has been modified into four

phases only: Data Preparation, Modeling, Evaluation, and Data Understanding. This model will be relevant in achieving the objectives through data analysis, evaluation, and results prediction.

Data Understanding

This is the phase where the data necessary for the study is provided. The data set was collected from the UCI Machine Learning Repository. This repository contains datasets, domain theories, and data generators that the machine learning community uses for the empirical analysis of machine learning algorithms (UCI Machine Learning Repository: About, 2021). This data will be helpful in the analysis and the prediction and evaluation of the results.

Data

Data File Name	Description	Columns	Rows	Size	Data Set Characteristics	Attribute Characteristics
heart.csv	Heart Disease Database	13 columns	270 rows	16,461 bytes	Multivariate	Categorical, Real

Associated Tasks	Number of Instances	Number of Attributes	Missing Values?	Area	Date Donated	Number of Web Hits
Classification	270	13	No	Life	N/A	264282

Attribute Information

- age – (Real)
- sex – (Binary)
- chest pain type (4 values) – (Nominal)
- resting blood pressure – (Real)
- serum cholesterol in mg/dl – (Real)

- fasting blood sugar > 120 mg/dl (Binary)
- resting electrocardiographic results (values 0,1,2) – (Nominal)
- maximum heart rate achieved – (Real)
- exercise-induced angina – (Binary)
- oldpeak = ST depression induced by exercise relative to rest – (Real)
- the slope of the peak exercise ST segment – (Ordered)
- number of major vessels (0-3) colored by fluoroscopy – (Real)
- thal: 3 = normal; 6 = fixed defect; 7 = reversible defect – (Nominal)
- predicted value: Absence (1) or presence (2) of heart disease – (Binary)

Data Preparation

This phase is where the dataset is modified to prepare for the algorithm and modeling. The dataset is scanned for missing values and formatted properly in data cleaning. There are no missing values, but the data points are stored in one column, so they must be separated to be able to determine the attributes and target values. Moreover, the python programming language was utilized through Google Colab notebook.

Loading the libraries

For Data Preparation

```
import pandas as pd
```

For Modelling

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

For Evaluation

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

For Graph

```
from sklearn.tree import export_graphviz
from six import StringIO
from IPython.display import Image
import pydotplus
```

Using the read.csv() function, the dataset was read from the downloaded file in Google Colab and displayed the internal structure of the variable "heart_data."

```
def importdata():
    heart_data = pd.read_csv('heart.csv', sep = ',', header = None, names = ['data'])

    print ("Dataset: ", heart_data)
```

```
Dataset:                                                    data
0    70.0 1.0 4.0 130.0 322.0 0.0 2.0 109.0 0.0 2.4...
1    67.0 0.0 3.0 115.0 564.0 0.0 2.0 160.0 0.0 1.6...
2    57.0 1.0 2.0 124.0 261.0 0.0 0.0 141.0 0.0 0.3...
3    64.0 1.0 4.0 128.0 263.0 0.0 0.0 105.0 1.0 0.2...
4    74.0 0.0 2.0 120.0 269.0 0.0 2.0 121.0 1.0 0.2...
..                                                    ...
265  52.0 1.0 3.0 172.0 199.0 1.0 0.0 162.0 0.0 0.5...
266  44.0 1.0 2.0 120.0 263.0 0.0 0.0 173.0 0.0 0.0...
267  56.0 0.0 2.0 140.0 294.0 0.0 2.0 153.0 0.0 1.3...
268  57.0 1.0 4.0 140.0 192.0 0.0 0.0 148.0 0.0 0.4...
269  67.0 1.0 4.0 160.0 286.0 0.0 2.0 108.0 1.0 1.5...
```

Figure 2. Structure of the data frame 'heart_data'

The list() function is used to separate the data points, and lambda is applied to split the points separated by a single space (' ').

```
heart_data = list(heart_data['data'].apply(lambda x:x.split(" ") ))
```

Using the DataFrame() function, the list is converted into a data frame with column names indicated for each column. The length and shape of the dataset are displayed as well as the dataset.

```
col_names = ['age', 'sex', 'chest pain type', 'resting blood pressure', 'serum cholesterol',
             'fasting blood sugar', 'resting electrocardiographic results', 'maximum heart rate achieved',
             'exercise induced angina', 'oldpeak', 'slope of the peak exercise', 'number of major vessels', 'thal', 'label']

heart_dataframe = pd.DataFrame(heart_data, columns = col_names)

print ("\nDataset Length: ", len(heart_dataframe))
print ("Dataset Shape: ", heart_dataframe.shape)

print ("Dataset: ", heart_dataframe)

return heart_dataframe
```

```
Dataset Length:  270
Dataset Shape:  (270, 14)
Dataset:        age  sex chest pain type  ... number of major vessels thal label
0    70.0  1.0              4.0  ...                       3.0  3.0     2
1    67.0  0.0              3.0  ...                       0.0  7.0     1
2    57.0  1.0              2.0  ...                       0.0  7.0     2
3    64.0  1.0              4.0  ...                       1.0  7.0     1
4    74.0  0.0              2.0  ...                       1.0  3.0     1
..    ...  ...              ...  ...                       ...  ...   ...
265  52.0  1.0              3.0  ...                       0.0  7.0     1
266  44.0  1.0              2.0  ...                       0.0  7.0     1
267  56.0  0.0              2.0  ...                       0.0  3.0     1
268  57.0  1.0              4.0  ...                       0.0  6.0     1
269  67.0  1.0              4.0  ...                       3.0  3.0     2

[270 rows x 14 columns]
```

Figure 3. Length, Shape, and Internal structure of the 'heart_dataframe'

Setting the attributes' values (X) and the target values (Y) from the heart_dataframe.

```
def splitdataset(heart_dataframe):
    X = heart_dataframe.values[:, 0:13]
    Y = heart_dataframe.values[:, 13]
```

Using the train_test_split() function, the dataset is split into the training data and test data. The test data size will be 0.3 (30%) and while for training data is 0.7 (70%).

```
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_state = 100)

    return X, Y, X_train, X_test, y_train, y_test
```

Modeling

Creating the model using the function DecisionTreeClassifier() with criterion as 'entropy,' max_depth of 4 (number of decision nodes), and min_samples_leaf of 5 (minimum number of leaf nodes).

```
def train_using_entropy(X_train, y_train):

    clf_entropy = DecisionTreeClassifier(criterion = "entropy", random_state = 100, max_depth = 4, min_samples_leaf = 5)
```

To determine the best attributes to be used in the decision tree model, the fit() function is used.

```
    clf_entropy.fit(X_train, y_train)

    return clf_entropy
```

Predicting the target values of the test data using the predict() function of the entropy model. Then, displaying the predicted values and test data length.

```python
def prediction(X_test, clf_object):

    y_pred = clf_object.predict(X_test)
    print("Predicted values:")
    print(y_pred)
    print("\nTest data: ", len(y_pred))

    return y_pred
```

```
Results Using Entropy:
Predicted values:
['1' '1' '2' '2' '1' '2' '1' '1' '2' '1' '2' '1' '2' '2' '1' '2' '1' '1'
 '1' '2' '1' '1' '2' '2' '2' '1' '1' '1' '1' '1' '1' '2' '1' '2' '1' '1'
 '1' '1' '1' '1' '2' '1' '1' '1' '1' '1' '1' '1' '1' '1' '1' '2' '1' '2' '1'
 '2' '1' '2' '1' '1' '1' '1' '1' '1' '2' '1' '2' '1' '1' '2' '1' '1' '1'
 '2' '1' '1' '2' '2' '1' '2' '1' '2']

Test data:  81
```

Figure 4: Predicted values of test data and its size (30% of 270)

Evaluation

Evaluating the model using the functions confusion_matrix(), accuracy_score(), and classification_report() with the actual target values and predicted target values of test data.

```
def cal_accuracy(y_test, y_pred):

    print("\nConfusion Matrix: \n", confusion_matrix(y_test, y_pred))

    print ("\nAccuracy : ", accuracy_score(y_test,y_pred)*100)

    print("\nReport : ", classification_report(y_test, y_pred))
```

```
Confusion Matrix:
 [[41  4]
 [13 23]]

Accuracy :  79.01234567901234

Report :               precision    recall  f1-score   support

           1       0.76      0.91      0.83        45
           2       0.85      0.64      0.73        36

    accuracy                           0.79        81
   macro avg       0.81      0.77      0.78        81
weighted avg       0.80      0.79      0.78        81
```

Figure 5. Confusion Matrix, Accuracy, and Classification Report of the Model

Manual computation of precision, recall, f1-score, and accuracy based on confusion matrix.

		Predicted Values		
		1	2	
Actual Values	1	**41 (TP)**	**4 (FN)**	45
	2	**13 (FP)**	**23 (TN)**	36
		54	27	**81**

Class 1

Precision = TP/ Predicted Yes = 41/54 = 0.759 = **0.76** Recall = TP/ Actual Yes = 41/45 = **0.91**

F1-score = $2 \times (\frac{precision \times recall}{precision + recall}) = 2 \times (\frac{0.76 \times 0.91}{0.76 + 0.91}) = 2 \times 0.414131 = 0.828 =$ **0.83**

Class 2

Precision = TN/ Predicted No = 23/27 = **0.85** Recall = TN/ Actual No = 23/36 = 0.638 = **0.64**

F1-score = $2 \times (\frac{precision \times recall}{precision + recall}) = 2 \times (\frac{0.85 \times 0.64}{0.85 + 0.64}) = 2 \times 0.365100 =$ **0.73**

Accuracy = (TP + TN)/ Total = (41 +23)/ 81 = **0.79**

Graph of Decision Tree

The StringIO() module is used to output the dot file and the export_graphviz() function to convert the decision tree classifier into a dot file. The feature (attributes) columns and the class names (target class) are set.

```
def decisionTreeGraph(clf_entropy):
    feature_cols = ['age', 'sex', 'chest pain type', 'resting blood pressure', 'serum cholesterol',
                'fasting blood sugar', 'resting electrocardiographic results', 'maximum heart rate achieved',
                'exercise induced angina', 'oldpeak', 'slope of the peak exercise', 'number of major vessels', 'thal']

    dot_data = StringIO()
    export_graphviz(clf_entropy, out_file=dot_data,
                    filled=True, rounded=True,
                    special_characters=True, feature_names = feature_cols,class_names = clf_entropy.classes_)
```

Using pydotplus to convert the dot file to png, write_png() to write the name of the png file, and Image()module to create the png file into a displayable form.

```
    graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
    graph.write_png('heart_data.png')
    Image(graph.create_png())
```

The program's main function is to call all the defined functions above.

```
def main():

    data = importdata()
    X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
    clf_entropy = train_using_entropy(X_train, y_train)

    print("\nResults Using Entropy:")

    y_pred_entropy = prediction(X_test, clf_entropy)
    cal_accuracy(y_test, y_pred_entropy)

    decisionTreeGraph(clf_entropy)


main()
```

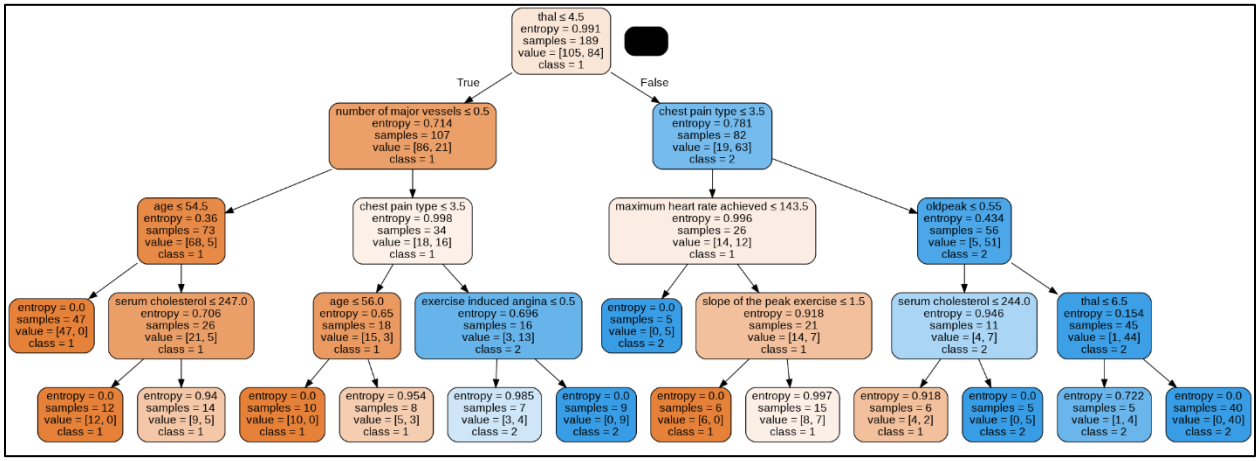

Figure 6. Graph of the Decision Tree

Results (Predictions)

Decision Rules for Class 1 (Absence of Heart Disease)

- If thal ≤ 4.5 and number of major vessels ≤ 0.5 and age ≤ 54.5, then class = 1
- If thal ≤ 4.5 and number of major vessels ≤ 0.5 and age !≤ (>) 54.5 and serum cholesterol ≤ 247.0, then class = 1
- If thal ≤ 4.5 and number of major vessels ≤ 0.5 and age !≤ (>) 54.5 and serum cholesterol !≤ 247.0 then class = 1
- If thal ≤ 4.5 and number of major vessels !≤ 0.5 and chest pain type ≤ 3.5 and age ≤ 56.0, then class = 1
- If thal ≤ 4.5 and number of major vessels !≤ 0.5 and chest pain type ≤ 3.5 and age !≤ 56.0, then class = 1
- If that !≤ 4.5 and chest pain type ≤ 3.5 and maximum heart rate achieved !≤ 143.5 and slope of the peak exercise ≤ 1.5, then class = 1
- If that !≤ 4.5 and chest pain type ≤ 3.5 and maximum heart rate achieved !≤ 143.5 and slope of the peak exercise !≤ 1.5 then class = 1
- If that !≤ 4.5 and chest pain type !≤ 3.5 and old peak ≤ 0.55 and serum cholesterol ≤ 244.0, then class = 1

Decision Rules for Class 2 (Presence of Heart Disease)

- If thal ≤ 4.5 and number of major vessels !≤ 0.5 and chest pain type !≤ 3.5 and exercise-induced angina ≤ 0.5 then class = 2
- If thal ≤ 4.5 and number of major vessels !≤ 0.5 and chest pain type !≤ 3.5 and exercise-induced angina !≤ 0.5 then class = 2

- If that !≤ 4.5 and chest pain type ≤ 3.5 and maximum heart rate achieved ≤ 143.5, then class = 2
- If that !≤ 4.5 and chest pain type !≤ 3.5 and old peak ≤ 0.55 and serum cholesterol !≤ 244.0 then class = 2
- If that !≤ 4.5 and chest pain type !≤ 3.5 and old peak !≤ 0.55 and thal ≤ 6.5 then class = 2
- If that !≤ 4.5 and chest pain type !≤ 3.5 and old peak !≤ 0.55 and thal !≤ 6.5 then class = 2

Conclusion

In Figure 4, it was depicted that among the test data set with size 81 (30% of 270), the predicted values for class 1 (absence of heart disease) accumulated 54 or 66.67%, while the predicted values for class 2 (presence of heart disease) obtained 27 or 33.33%. Meanwhile, based on Figure 5, it can be concluded that the generated model was able to produce good results as it garnered an accuracy rate of 79% and an error rate of 21%. Moreover, Figure 6 revealed that the best attribute used in the model is *thal* which has the highest information gain.

References

Classification Algorithm in Machine Learning - Javatpoint. (2021). Retrieved from JavaTPoint: https://www.javatpoint.com/classification-algorithm-in-machine-learning

Navlani, A. (2018). *Python Decision Tree Classification with Scikit-Learn DecisionTreeClassifier*. Retrieved from Data Camp Community: https://www.datacamp.com/community/tutorials/decision-tree-classification-python?utm_source=adwords_ppc&utm_medium=cpc&utm_campaignid=14989519638&utm_adgroupid=127836677279&utm_device=c&utm_keyword=&utm_matchtype=&utm_network=g&utm_adpostion=&utm_creative

Python | Decision tree implementation - GeeksforGeeks. (2017, November). Retrieved from GeeksforGeeks. https://www.geeksforgeeks.org/decision-tree-implementation-python/?ref=lbp

Rizzo, P. (2014). *Sensing solutions for assessing and monitoring railroad tracks*. Retrieved from ScienceDirect: https://www.sciencedirect.com/topics/engineering/classification-algorithm

UCI Machine Learning Repository: Statlog (Heart) Data Set. (2021). Retrieved from UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/Statlog+%28Heart%29

Uduak Idio Akpan, A. S. (2021). *Review of classification algorithms with changing inter-class distances*. Retrieved from ScienceDirect: https://www.sciencedirect.com/science/article/pii/S2666827021000128?dgcid=raven_sd_aip_email