PROJECT REPORT

(Data Mining)



**MCS FINAL 2021**

GROUP MEMBERS:

1. **ANUM ZEHRA (P19101008)**
2. **IQRA MEHDI (P19101025)**
3. **SANA ASHFAQ (P19101063)**
4. **SHAHID IQBAL (P19101064)**

SUPERVISOR:

**Dr. TAHSEEN A. JILANI**

**Heart Disease Prediction Project Report**

Our project is to predict whether patients have heart disease or not by given a number of features from patients. The inspiration of our project is to spare human assets in medical centers and move forward exactness of conclusion. In our project we utilize diverse strategies to detect heart disease such as Logistic Regression, SVM, Naïve Bayes, Random Forest and Artificial neural network. And among all these calculations Random Forest gives us the finest precision of 91.8%.

1. **Introduction**

In day to day life many factors that influence a human heart. Many problems are happening at a fast pace and new heart diseases are rapidly being recognized. In today’s world of stress Heart, being an essential organ in a human body which pumps blood through the body for the blood circulation is fundamental and its health is to be conserved for a sound living. The health of a human heart is based on the encounters in a person’s life and is completely dependent on proficient and personal behaviours of a person. There may also be a few genetic factors through which a sort of heart illness is passed down from eras. Concurring to the World Health Organization, every year more than 12 million deaths are happening around the world due to the different sorts of heart diseases which is additionally known by the term cardiovascular disease. The term Heart disease includes numerous diseases that are diverse and particularly affect the heart and the arteries of a human being. Even youthful matured individuals around their 20-30 a long time of life expectancy are getting influenced by heart diseases. The increment within the possibility of heart disease among young may be due to the bad eating habits, lack of rest, anxious nature, depression, discouragement and various other factors such as obesity, poor diet, family history, high blood pressure, high blood cholesterol, idle behaviour, smoking and hypertension.

The diagnosis of the heart diseases could be a exceptionally important and is itself the most complicated task in the medical field. All the mentioned components are taken into consideration when analysing and understanding the patients by the specialist through manual check-ups at regular intervals of time. The symptoms of heart disease significantly depend upon which of the distress felt by an person. A few side effects are not usually identified by the common people. However, common symptoms include chest pain, breathlessness, and heart palpitations. The chest pain common to many types of heart disease is known as angina, or angina pectoris, and happens when a portion of the heart does not get sufficient oxygen. Angina may be activated by stressful events or physical effort and normally lasts under 10 minutes. Heart attacks can also happen as a result of different types of heart disease.

Data Mining is an important decision-making process information from past collections for future analysis or forecast. Information may be anonymous and may not be identified without using a datamine. The section says a single data mining process where the future result or predictions can be made based on historical data i.e. available. Digging for medical data has created a possible solution

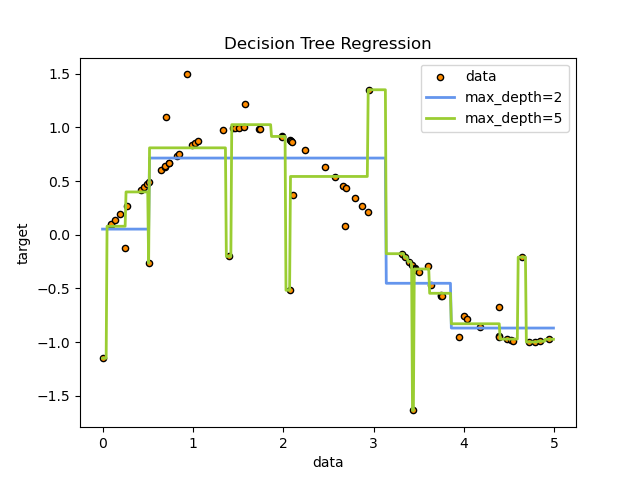
combine classification techniques and deliver by computer database training that leads continuously to hidden tests patterns in medical data sets used for prediction of the patient's future status. So, using medical data to dig it is able to provide information about patient history and is capable provided clinical support through analysis. Clinical analysis in patients, these patterns are very important. In English, medical data mining uses classification algorithms that is an important part of diagnosing the possibility of a heart attack before it happened. Separation algorithms can be trained and tested to make decisive predictions a person's condition of heart attack.

# DECISION TREES

**INTRODUCTION:**

**Decision Trees (DTs)** are a non-parametric supervised learning method used for [classification](https://scikit-learn.org/stable/modules/tree.html#tree-classification) and [regression](https://scikit-learn.org/stable/modules/tree.html#tree-regression). The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

For instance, in the example below, decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules. The deeper the tree, the more complex the decision rules and the fitter the model.



**Advantage:**

Some advantages of decision trees are:

* Simple to understand and to interpret. Trees can be visualised.
* Requires little data preparation. Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values.
* The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
* Able to handle both numerical and categorical data. However scikit-learn implementation does not support categorical variables for now. Other techniques are usually specialised in analysing datasets that have only one type of variable. See [algorithms](https://scikit-learn.org/stable/modules/tree.html#tree-algorithms) for more information.
* Able to handle multi-output problems.
* Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret.
* Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
* Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

**Disadvantages**

* Decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting. Mechanisms such as pruning, setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
* Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.
* Predictions of decision trees are neither smooth nor continuous, but piecewise constant approximations as seen in the above figure. Therefore, they are not good at extrapolation.
* The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees in an ensemble learner, where the features and samples are randomly sampled with replacement.
* There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems.
* Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.

**Classification:**

[**DecisionTreeClassifier**](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier) is a class capable of performing multi-class classification on a dataset.

As with other classifiers, [**DecisionTreeClassifier**](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier) takes as input two arrays: an array X, sparse or dense, of shape (n\_samples, n\_features) holding the training samples, and an array Y of integer values, shape (n\_samples,), holding the class labels for the training samples:

>>>

**>>> from** **sklearn** **import** tree

**>>>** X = [[0, 0], [1, 1]]

**>>>** Y = [0, 1]

**>>>** clf = tree.DecisionTreeClassifier()

**>>>** clf = clf.fit(X, Y)

After being fitted, the model can then be used to predict the class of samples:

>>>

**>>>** clf.predict([[2., 2.]])

array([1])

In case that there are multiple classes with the same and highest probability, the classifier will predict the class with the lowest index amongst those classes.

As an alternative to outputting a specific class, the probability of each class can be predicted, which is the fraction of training samples of the class in a leaf:

>>>

**>>>** clf.predict\_proba([[2., 2.]])

array([[0., 1.]])

[**DecisionTreeClassifier**](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier) is capable of both binary (where the labels are [-1, 1]) classification and multiclass (where the labels are [0, …, K-1]) classification.

Using the Iris dataset, we can construct a tree as follows:

>>>

**>>> from** **sklearn.datasets** **import** load\_iris

**>>> from** **sklearn** **import** tree

**>>>** iris = load\_iris()

**>>>** X, y = iris.data, iris.target

**>>>** clf = tree.DecisionTreeClassifier()

**>>>** clf = clf.fit(X, y)

Below is an example graphviz export of the above tree trained on the entire iris dataset; the results are saved in an output file iris.pdf:

>>>

**>>> import** **graphviz**

**>>>** dot\_data = tree.export\_graphviz(clf, out\_file=**None**)

**>>>** graph = graphviz.Source(dot\_data)

**>>>** graph.render("iris")

The [**export\_graphviz**](https://scikit-learn.org/stable/modules/generated/sklearn.tree.export_graphviz.html#sklearn.tree.export_graphviz) exporter also supports a variety of aesthetic options, including coloring nodes by their class (or value for regression) and using explicit variable and class names if desired. Jupyter notebooks also render these plots inline automatically:

>>>

**>>>** dot\_data = tree.export\_graphviz(clf, out\_file=**None**,

**...**  feature\_names=iris.feature\_names,

**...**  class\_names=iris.target\_names,

**...**  filled=**True**, rounded=**True**,

**...**  special\_characters=**True**)

**>>>** graph = graphviz.Source(dot\_data)

**>>>** graph

**>>> import** **graphviz**

**>>>** dot\_data = tree.export\_graphviz(clf, out\_file=**None**)

**>>>** graph = graphviz.Source(dot\_data)

**>>>** graph.render("iris")

The [**export\_graphviz**](https://scikit-learn.org/stable/modules/generated/sklearn.tree.export_graphviz.html#sklearn.tree.export_graphviz) exporter also supports a variety of aesthetic options, including coloring nodes by their class (or value for regression) and using explicit variable and class names if desired. Jupyter notebooks also render these plots inline automatically:

>>>

**>>>** dot\_data = tree.export\_graphviz(clf, out\_file=**None**,

**...**  feature\_names=iris.feature\_names,

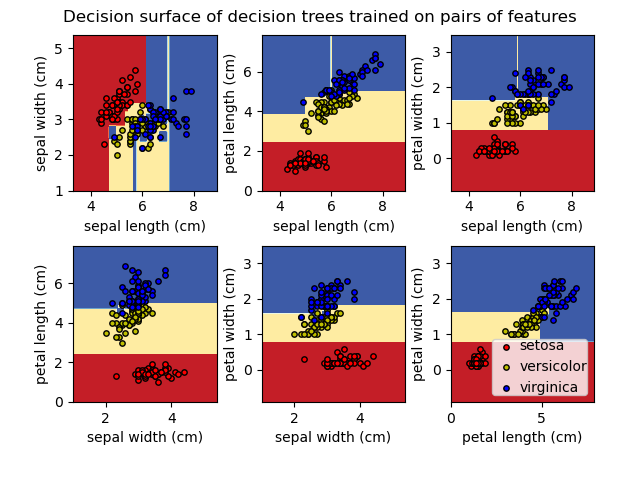
**...**  class\_names=iris.target\_names,

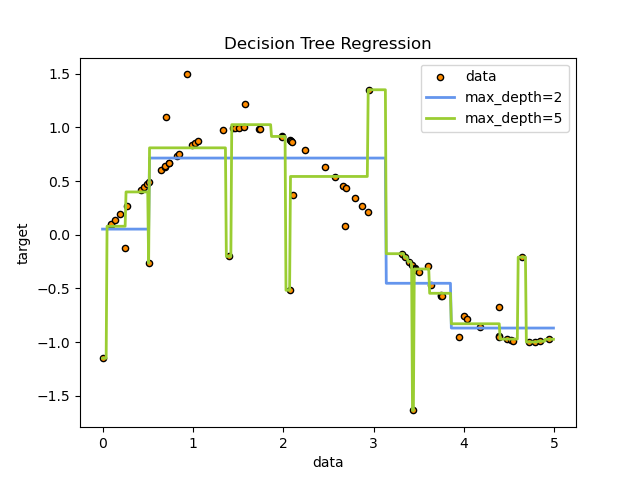
**...**  filled=**True**, rounded=**True**,

**...**  special\_characters=**True**)

**>>>** graph = graphviz.Source(dot\_data)

**>>>** graph





**Multi-output problems**

A multi-output problem is a supervised learning problem with several outputs to predict, that is when Y is a 2d array of shape (n\_samples, n\_outputs).

When there is no correlation between the outputs, a very simple way to solve this kind of problem is to build n independent models, i.e. one for each output, and then to use those models to independently predict each one of the n outputs. However, because it is likely that the output values related to the same input are themselves correlated, an often better way is to build a single model capable of predicting simultaneously all n outputs. First, it requires lower training time since only a single estimator is built. Second, the generalization accuracy of the resulting estimator may often be increased.

With regard to decision trees, this strategy can readily be used to support multi-output problems. This requires the following changes:

* Store n output values in leaves, instead of 1;
* Use splitting criteria that compute the average reduction across all n outputs.

This module offers support for multi-output problems by implementing this strategy in both [**DecisionTreeClassifier**](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier) and [**DecisionTreeRegressor**](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html#sklearn.tree.DecisionTreeRegressor). If a decision tree is fit on an output array Y of shape (n\_samples, n\_outputs) then the resulting estimator will:

* Output n\_output values upon predict;
* Output a list of n\_output arrays of class probabilities upon predict\_proba.

The use of multi-output trees for regression is demonstrated in [Multi-output Decision Tree Regression](https://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression_multioutput.html#sphx-glr-auto-examples-tree-plot-tree-regression-multioutput-py). In this example, the input X is a single real value and the outputs Y are the sine and cosine of X.

**COMPLEXITY:**

n general, the run time cost to construct a balanced binary tree is O(nsamplesnfeatureslog⁡(nsamples)) and query time O(log⁡(nsamples)). Although the tree construction algorithm attempts to generate balanced trees, they will not always be balanced. Assuming that the subtrees remain approximately balanced, the cost at each node consists of searching through O(nfeatures) to find the feature that offers the largest reduction in entropy. This has a cost of O(nfeaturesnsampleslog⁡(nsamples)) at each node, leading to a total cost over the entire trees (by summing the cost at each node) of O(nfeaturesnsamples2log⁡(nsamples)).

**TIPS ON PRACTICAL USE:**

* Decision trees tend to overfit on data with a large number of features. Getting the right ratio of samples to number of features is important, since a tree with few samples in high dimensional space is very likely to overfit.
* Consider performing dimensionality reduction ([PCA](https://scikit-learn.org/stable/modules/decomposition.html#pca), [ICA](https://scikit-learn.org/stable/modules/decomposition.html#ica), or [Feature selection](https://scikit-learn.org/stable/modules/feature_selection.html#feature-selection)) beforehand to give your tree a better chance of finding features that are discriminative.
* [Understanding the decision tree structure](https://scikit-learn.org/stable/auto_examples/tree/plot_unveil_tree_structure.html#sphx-glr-auto-examples-tree-plot-unveil-tree-structure-py) will help in gaining more insights about how the decision tree makes predictions, which is important for understanding the important features in the data.
* Visualise your tree as you are training by using the export function. Use max\_depth=3 as an initial tree depth to get a feel for how the tree is fitting to your data, and then increase the depth.
* Remember that the number of samples required to populate the tree doubles for each additional level the tree grows to. Use max\_depth to control the size of the tree to prevent overfitting.
* Use min\_samples\_split or min\_samples\_leaf to ensure that multiple samples inform every decision in the tree, by controlling which splits will be considered. A very small number will usually mean the tree will overfit, whereas a large number will prevent the tree from learning the data. Try min\_samples\_leaf=5 as an initial value. If the sample size varies greatly, a float number can be used as percentage in these two parameters. While min\_samples\_split can create arbitrarily small leaves, min\_samples\_leaf guarantees that each leaf has a minimum size, avoiding low-variance, over-fit leaf nodes in regression problems. For classification with few classes, min\_samples\_leaf=1 is often the best choice.

Note that min\_samples\_split considers samples directly and independent of sample\_weight, if provided (e.g. a node with m weighted samples is still treated as having exactly m samples). Consider min\_weight\_fraction\_leaf or min\_impurity\_decrease if accounting for sample weights is required at splits.

* Balance your dataset before training to prevent the tree from being biased toward the classes that are dominant. Class balancing can be done by sampling an equal number of samples from each class, or preferably by normalizing the sum of the sample weights (sample\_weight) for each class to the same value. Also note that weight-based pre-pruning criteria, such as min\_weight\_fraction\_leaf, will then be less biased toward dominant classes than criteria that are not aware of the sample weights, like min\_samples\_leaf.
* If the samples are weighted, it will be easier to optimize the tree structure using weight-based pre-pruning criterion such as min\_weight\_fraction\_leaf, which ensure that leaf nodes contain at least a fraction of the overall sum of the sample weights.
* All decision trees use np.float32 arrays internally. If training data is not in this format, a copy of the dataset will be made.
* If the input matrix X is very sparse, it is recommended to convert to sparse csc\_matrix before calling fit and sparse csr\_matrix before calling predict. Training time can be orders of magnitude faster for a sparse matrix input compared to a dense matrix when features have zero values in most of the samples.

**Tree algorithms: ID3, C4.5, C5.0 and CART**

What are all the various decision tree algorithms and how do they differ from each other? Which one is implemented in scikit-learn?

[ID3](https://en.wikipedia.org/wiki/ID3_algorithm) (Iterative Dichotomiser 3) was developed in 1986 by Ross Quinlan. The algorithm creates a multiway tree, finding for each node (i.e. in a greedy manner) the categorical feature that will yield the largest information gain for categorical targets. Trees are grown to their maximum size and then a pruning step is usually applied to improve the ability of the tree to generalise to unseen data.

C4.5 is the successor to ID3 and removed the restriction that features must be categorical by dynamically defining a discrete attribute (based on numerical variables) that partitions the continuous attribute value into a discrete set of intervals. C4.5 converts the trained trees (i.e. the output of the ID3 algorithm) into sets of if-then rules. These accuracy of each rule is then evaluated to determine the order in which they should be applied. Pruning is done by removing a rule’s precondition if the accuracy of the rule improves without it.

C5.0 is Quinlan’s latest version release under a proprietary license. It uses less memory and builds smaller rulesets than C4.5 while being more accurate.

[CART](https://en.wikipedia.org/wiki/Predictive_analytics#Classification_and_regression_trees_.28CART.29) (Classification and Regression Trees) is very similar to C4.5, but it differs in that it supports numerical target variables (regression) and does not compute rule sets. CART constructs binary trees using the feature and threshold that yield the largest information gain at each node.

scikit-learn uses an optimised version of the CART algorithm; however, scikit-learn implementation does not support categorical variables for now.

**MATHEMATICAL FORMULATION:**

Given training vectors

xi∈Rn, i=1,…, l

and a label vector

y∈Rl, a decision tree recursively partitions the feature space such that the samples with the same labels or similar target values are grouped together.

Let the data at node m be represented by Qm with Nm samples. For each candidate split

θ=(j,tm)

consisting of a feature j and threshold tm, partition the data into

Qmleft(θ)

 and

Qmright(θ)

 subsets

Qmleft(θ)={(x,y)|xj<=tm}Qmright(θ)=Qm∖Qmleft(θ)

The quality of a candidate split of node m is then computed using an impurity function or loss function H(), the choice of which depends on the task being solved (classification or regression)

G(Qm,θ)=NmleftNmH(Qmleft(θ))+NmrightNmH(Qmright(θ))

Select the parameters that minimises the impurity

θ∗=argminθ⁡G(Qm,θ)

Recurse for subsets Qmleft(θ∗) and Qmright(θ∗) until the maximum allowable depth is reached, Nm<minsamples or Nm=1.

**REGRESSION CRITERIA:**

If the target is a continuous value, then for node m, common criteria to minimize as for determining locations for future splits are Mean Squared Error (MSE or L2 error), Poisson deviance as well as Mean Absolute Error (MAE or L1 error). MSE and Poisson deviance both set the predicted value of terminal nodes to the learned mean value y¯m of the node whereas the MAE sets the predicted value of terminal nodes to the median median(y)m.

Mean Squared Error:

Half Poisson deviance:

Setting criterion="poisson" might be a good choice if your target is a count or a frequency (count per some unit). In any case, y>=0 is a necessary condition to use this criterion. Note that it fits much slower than the MSE criterion.

Mean Absolute Error:

**MINIMAL COST-COMPLEXITY PRUNING:**

Minimal cost-complexity pruning is an algorithm used to prune a tree to avoid over-fitting, described in Chapter 3 of [[BRE]](https://scikit-learn.org/stable/modules/tree.html#bre). This algorithm is parameterized by α≥0 known as the complexity parameter. The complexity parameter is used to define the cost-complexity measure, Rα(T) of a given tree T:

Rα(T)=R(T)+α|T~|

where |T~| is the number of terminal nodes in T and R(T) is traditionally defined as the total misclassification rate of the terminal nodes. Alternatively, scikit-learn uses the total sample weighted impurity of the terminal nodes for R(T). As shown above, the impurity of a node depends on the criterion. Minimal cost-complexity pruning finds the subtree of T that minimizes Rα(T).

The cost complexity measure of a single node is Rα(t)=R(t)+α. The branch, Tt, is defined to be a tree where node t is its root. In general, the impurity of a node is greater than the sum of impurities of its terminal nodes, R(Tt)<R(t). However, the cost complexity measure of a node, t, and its branch, Tt, can be equal depending on α. We define the effective α of a node to be the value where they are equal, Rα(Tt)=Rα(t) or αeff(t)=R(t)−R(Tt)|T|−1. A non-terminal node with the smallest value of αeff is the weakest link and will be pruned. This process stops when the pruned tree’s minimal αeff is greater than the ccp\_alpha parameter.

**ILLUSTRARTIONS:**

NOMREG HeartDisease (BASE=LAST ORDER=ASCENDING) WITH Age Sex ChestPainType BP Cholesterol

FBSOver120 EKGResults MaxHR ExcerciseAngina STDepression SlopeOfST NoOfVesselsFluro Thallium

/CRITERIA CIN(95) DELTA(0) MXITER(100) MXSTEP(5) CHKSEP(20) LCONVERGE(0) PCONVERGE(0.000001)

SINGULAR(0.00000001)

/MODEL

/STEPWISE=PIN(.05) POUT(0.1) MINEFFECT(0) RULE(SINGLE) ENTRYMETHOD(LR) REMOVALMETHOD(LR)

/INTERCEPT=INCLUDE

/PRINT=FIT PARAMETER SUMMARY LRT CPS STEP MFI.

**Nominal Regression**

|  |  |  |
| --- | --- | --- |
| **Notes** | | |
| Output Created | | 30-DEC-2021 01:21:41 |
| Comments | |  |
| Input | Data | C:\Users\Dilawar Asad\Desktop\HeartDisease.sav |
| Active Dataset | DataSet1 |
| Filter | <none> |
| Weight | <none> |
| Split File | <none> |
| N of Rows in Working Data File | 270 |
| Missing Value Handling | Definition of Missing | User-defined missing values are treated as missing. |
| Cases Used | Statistics are based on all cases with valid data for all variables in the model. |
| Syntax | | NOMREG HeartDisease (BASE=LAST ORDER=ASCENDING) WITH Age Sex ChestPainType BP Cholesterol  FBSOver120 EKGResults MaxHR ExcerciseAngina STDepression SlopeOfST NoOfVesselsFluro Thallium  /CRITERIA CIN(95) DELTA(0) MXITER(100) MXSTEP(5) CHKSEP(20) LCONVERGE(0) PCONVERGE(0.000001)  SINGULAR(0.00000001)  /MODEL  /STEPWISE=PIN(.05) POUT(0.1) MINEFFECT(0) RULE(SINGLE) ENTRYMETHOD(LR) REMOVALMETHOD(LR)  /INTERCEPT=INCLUDE  /PRINT=FIT PARAMETER SUMMARY LRT CPS STEP MFI. |
| Resources | Processor Time | 00:00:00.08 |
| Elapsed Time | 00:00:00.08 |

|  |
| --- |
| **Warnings** |
| There are 270 (50.0%) cells (i.e., dependent variable levels by subpopulations) with zero frequencies. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Case Processing Summary** | | | |
|  | | N | Marginal Percentage |
| HeartDisease | 0 | 150 | 55.6% |
| 1 | 120 | 44.4% |
| Valid | | 270 | 100.0% |
| Missing | | 0 |  |
| Total | | 270 |  |
| Subpopulation | | 270a |  |

|  |
| --- |
| a. The dependent variable has only one value observed in 270 (100.0%) subpopulations. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model Fitting Information** | | | | |
| Model | Model Fitting Criteria | Likelihood Ratio Tests | | |
| -2 Log Likelihood | Chi-Square | df | Sig. |
| Intercept Only | 370.959 |  |  |  |
| Final | 179.598 | 191.361 | 13 | .000 |

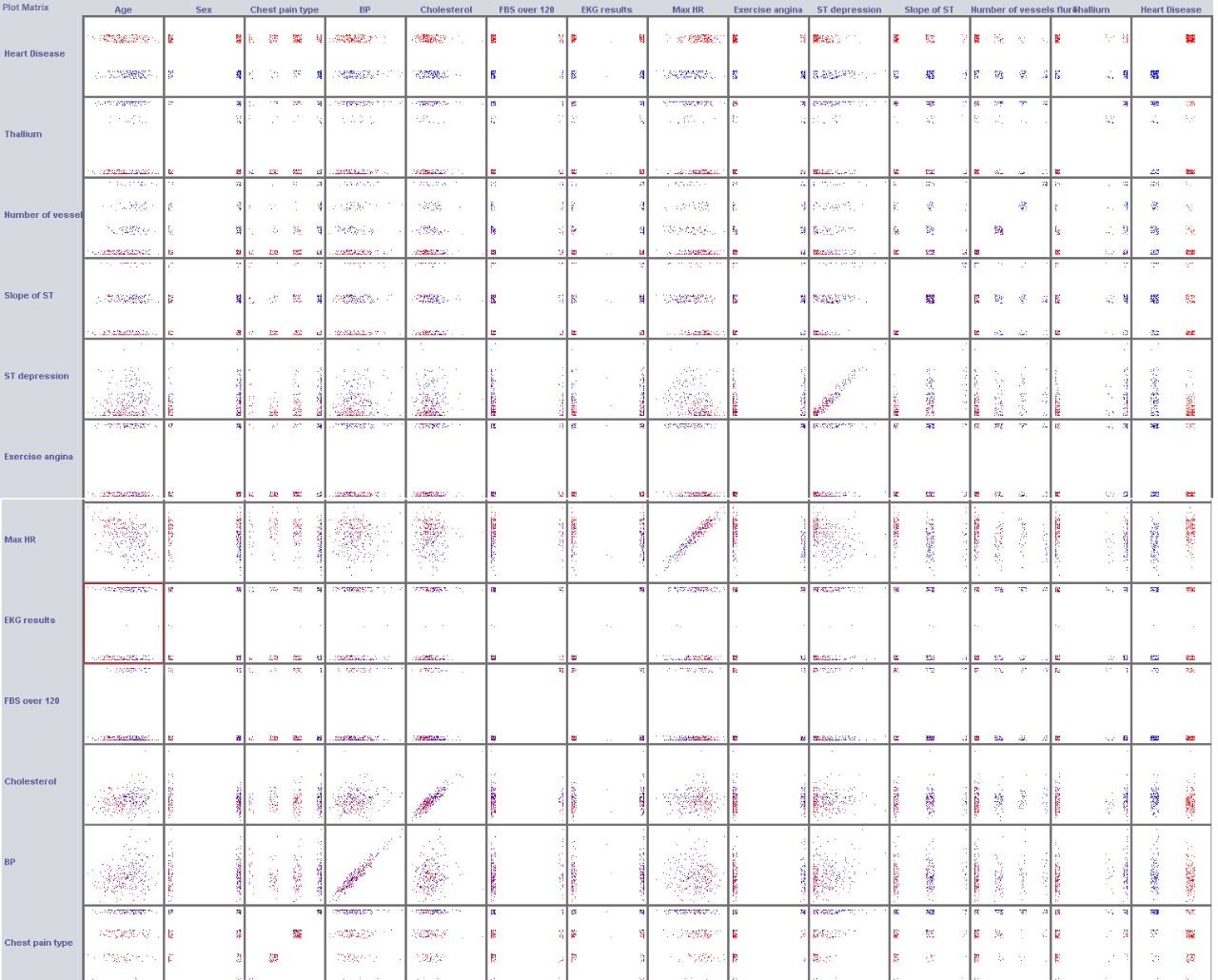
|  |  |  |  |
| --- | --- | --- | --- |
| **Goodness-of-Fit** | | | |
|  | Chi-Square | df | Sig. |
| Pearson | 232.117 | 256 | .856 |
| Deviance | 179.598 | 256 | 1.000 |

|  |  |
| --- | --- |
| **Pseudo R-Square** | |
| Cox and Snell | .508 |
| Nagelkerke | .680 |
| McFadden | .516 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Likelihood Ratio Tests** | | | | |
| Effect | Model Fitting Criteria | Likelihood Ratio Tests | | |
| -2 Log Likelihood of Reduced Model | Chi-Square | df | Sig. |
| Intercept | 187.628 | 8.031 | 1 | .005 |
| Age | 180.063 | .465 | 1 | .495 |
| Sex | 188.408 | 8.810 | 1 | .003 |
| ChestPainType | 191.342 | 11.744 | 1 | .001 |
| BP | 184.646 | 5.048 | 1 | .025 |
| Cholesterol | 182.847 | 3.249 | 1 | .071 |
| FBSOver120 | 181.571 | 1.973 | 1 | .160 |
| EKGResults | 181.956 | 2.358 | 1 | .125 |
| MaxHR | 183.687 | 4.089 | 1 | .043 |
| ExcerciseAngina | 183.262 | 3.664 | 1 | .056 |
| STDepression | 181.947 | 2.349 | 1 | .125 |
| SlopeOfST | 180.862 | 1.264 | 1 | .261 |
| NoOfVesselsFluro | 202.822 | 23.224 | 1 | .000 |
| Thallium | 190.248 | 10.650 | 1 | .001 |

|  |
| --- |
| The chi-square statistic is the difference in -2 log-likelihoods between the final model and a reduced model. The reduced model is formed by omitting an effect from the final model. The null hypothesis is that all parameters of that effect are 0. |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Parameter Estimates** | | | | | | | | | |
| HeartDiseasea | | B | Std. Error | Wald | df | Sig. | Exp(B) |  |  |
|  |  |
| 0 | Intercept | 8.446 | 3.088 | 7.481 | 1 | .006 |  |  |  |
| Age | .017 | .026 | .462 | 1 | .497 | 1.018 |  |  |
| Sex | -1.542 | .541 | 8.132 | 1 | .004 | .214 |  |  |
| ChestPainType | -.701 | .215 | 10.600 | 1 | .001 | .496 |  |  |
| BP | -.025 | .011 | 4.850 | 1 | .028 | .975 |  |  |
| Cholesterol | -.007 | .004 | 3.142 | 1 | .076 | .993 |  |  |
| FBSOver120 | .795 | .575 | 1.913 | 1 | .167 | 2.214 |  |  |
| EKGResults | -.302 | .198 | 2.325 | 1 | .127 | .740 |  |  |
| MaxHR | .021 | .011 | 3.957 | 1 | .047 | 1.021 |  |  |
| ExcerciseAngina | -.829 | .431 | 3.701 | 1 | .054 | .436 |  |  |
| STDepression | -.344 | .227 | 2.291 | 1 | .130 | .709 |  |  |
| SlopeOfST | -.442 | .391 | 1.279 | 1 | .258 | .643 |  |  |
| NoOfVesselsFluro | -1.165 | .269 | 18.726 | 1 | .000 | .312 |  |  |
| Thallium | -.341 | .106 | 10.359 | 1 | .001 | .711 |  |  |



**Examples:**

[Multi-output Decision Tree Regression](https://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression_multioutput.html#sphx-glr-auto-examples-tree-plot-tree-regression-multioutput-py)

[Face completion with a multi-output estimators](https://scikit-learn.org/stable/auto_examples/miscellaneous/plot_multioutput_face_completion.html#sphx-glr-auto-examples-miscellaneous-plot-multioutput-face-completion-py)

**References:**

[kaggle.com/ronitf/heart-disease-uci/version/1](file:///G:\backup\4th%20Sem\DataMining\kaggle.com\ronitf\heart-disease-uci\version\1)

<https://nthu-datalab.github.io/ml/labs/03_Decision-Trees_Random-Forest/03_Decision-Tree_Random-Forest.html>

M. Dumont et al, [Fast multi-class image annotation with random subwindows and multiple output randomized trees](http://www.montefiore.ulg.ac.be/services/stochastic/pubs/2009/DMWG09/dumont-visapp09-shortpaper.pdf), International Conference on Computer Vision Theory and Applications 2009