**PROJECT REPORT**

ON

**DIAGNOSIS OF HEART DISEASES**

USING

**SUPPORT VECTOR MACHINE AND LOGISTIC REGRESSION ALGORITHM**

SUBMITTED BY

**ARSHIYA**

E&TC

IIEST, SHIBPUR

UNDER THE GUIDANCE OF

**MR. BIPUL SHAHI**

**ABSTRACT**

Machine Learning can play an essential role in predicting presence/absence of disorders, Heart diseases and more such ailments prevalent in human body by analyzing and processing huge amount of data. Such information, if predicted well in advance, can provide important insights to doctors who can then adapt their diagnosis and treatment per patient basis. The diagnosis of heart disease is usually based on signs, symptoms and physical examination of the patient. There are several factors that increase the risk of heart disease, such as body cholesterol level, blood pressure level etc. This paper makes use of heart disease dataset available in UCI machine learning repository. The proposed work predicts the chances of Heart Disease and classifies patient's risk level by implementing different data mining techniques such as Support Vector Machine and Linear Regression.

**INTRODUCTION**

Any irregularity to heart can cause distress in other parts of body. Any sort of disturbance to normal functioning of the heart can be classified as a Heart disease. There are several factors which contribute to certain afflictions caused in the Heart. Some factors being :

* Age: Your risk of heart disease increases as you get older. Men age 45 and older and women age 55 and older have a greater risk.
* Sex: Some risk factors may affect heart disease risk differently in women than in men. For example, estrogen provides women some protection against heart disease, but diabetes raises the risk of heart disease more in women than in men.
* Cholesterol: High levels of cholesterol can clog your arteries and raise your risk of coronary artery disease and heart attack. Lifestyle changes and medicines (if needed) can lower your cholesterol. Triglycerides are another type of fat in the blood. High levels of triglycerides may also raise the risk of coronary artery disease, especially in women.

The main challenge in today's healthcare is provision of best quality services and effective accurate diagnosis. Even if heart diseases are found as the prime source of death in the world in recent years, they are also the ones that can be controlled and managed effectively. The whole accuracy in management of a disease lies on the proper time of detection of that disease.

The proposed work makes an attempt to detect these heart diseases at early stage to avoid disastrous consequences by analyzing and processing data by algorithms deployed using Machine Learning. Machine Learning, in recent times has proved to be indispensable as it can accurately detect patterns by extracting useful information from huge set of data, otherwise a herculean task by manual methods.

**CONTEXT**

This database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland database is the only one that has been used by ML researchers to this date. The target field contains a binary representation of the detection of presence/ absence of heart disease in a particular person.

**CONTENT**

Attribute Information:

1. age
2. sex (1= Male, 0= Female)
3. chest pain type (4 values)
4. resting blood pressure
5. serum cholesterol in mg/dl
6. fasting blood sugar > 120 mg/dl
7. resting electrocardiographic results (values 0,1,2)
8. maximum heart rate achieved
9. exercise induced angina
10. oldpeak = ST depression induced by exercise relative to rest
11. the slope of the peak exercise ST segment (Ordinal) (Value 1: up sloping , Value 2: flat , Value 3: down sloping )
12. number of major vessels (0-3) colored by fluoroscopy
13. thal: 3 = normal; 6 = fixed defect; 7 = reversible defect

Target value gives a binary depiction of label i.e. diagnosis result of heart disease with 0 being the case of absence of Heart related afflictions and 1 being the presence of heart diseases.

**ANALYZING THE DATA**

import pandas as pd

import numpy as nd

#assigning the input data frame to df

df = pd.read\_csv('heart.csv')

#displaying the top 5 content of df

df.head()

|  | **age** | **sex** | **cp** | **trestbps** | **chol** | **fbs** | **restecg** | **thalach** | **exang** | **oldpeak** | **slope** | **ca** | **thal** | **target** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 63 | 1 | 3 | 145 | 233 | 1 | 0 | 150 | 0 | 2.3 | 0 | 0 | 1 | 1 |
| **1** | 37 | 1 | 2 | 130 | 250 | 0 | 1 | 187 | 0 | 3.5 | 0 | 0 | 2 | 1 |
| **2** | 41 | 0 | 1 | 130 | 204 | 0 | 0 | 172 | 0 | 1.4 | 2 | 0 | 2 | 1 |
| **3** | 56 | 1 | 1 | 120 | 236 | 0 | 1 | 178 | 0 | 0.8 | 2 | 0 | 2 | 1 |
| **4** | 57 | 0 | 0 | 120 | 354 | 0 | 1 | 163 | 1 | 0.6 | 2 | 0 | 2 | 1 |

#checking for number of null values each feature has

df.isnull().sum()

age 0

sex 0

cp 0

trestbps 0

chol 0

fbs 0

restecg 0

thalach 0

exang 0

oldpeak 0

slope 0

ca 0

thal 0

target 0

dtype: int64

There are no null values present in any column.

#checking the data type of each column

df.dtypes

age int64

sex int64

cp int64

trestbps int64

chol int64

fbs int64

restecg int64

thalach int64

exang int64

oldpeak float64

slope int64

ca int64

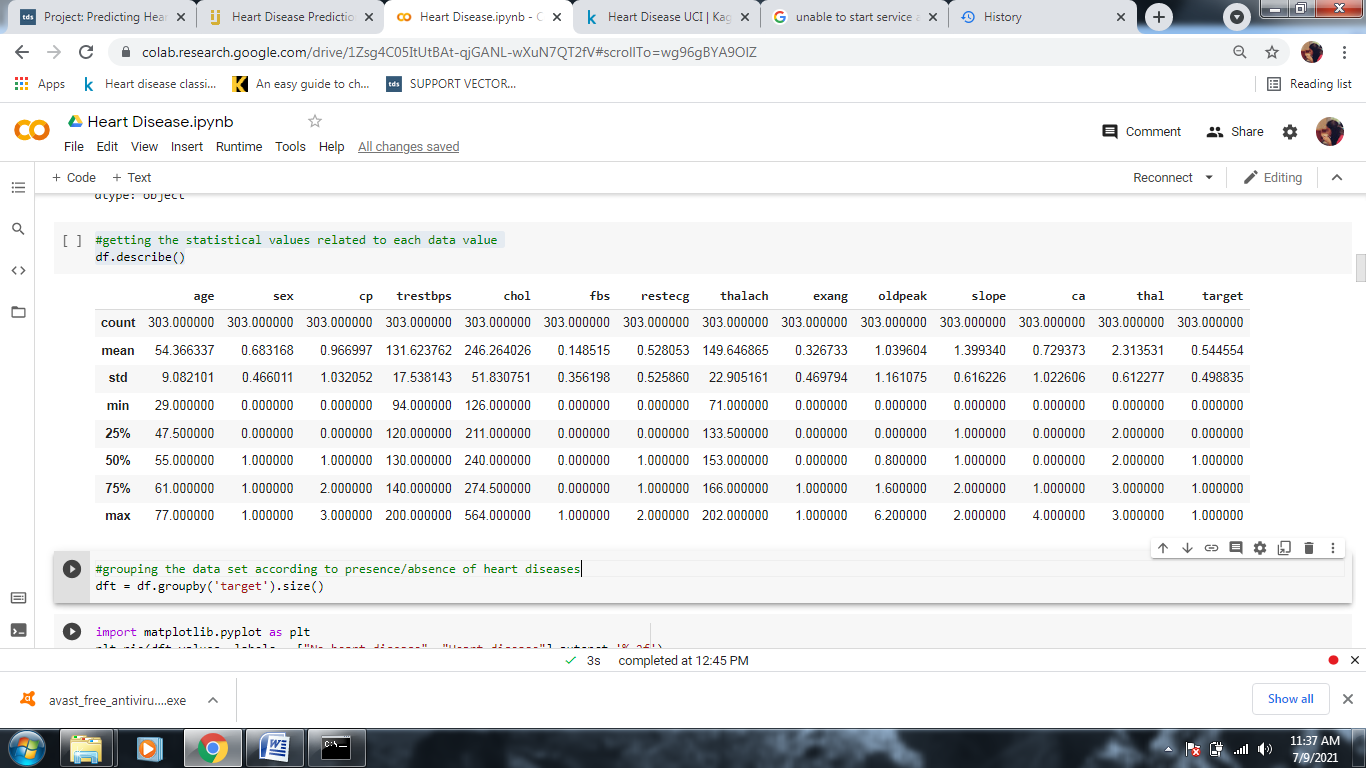
thal int64

target int64

dtype: object

#getting the statistical values related to each data value

df.describe()



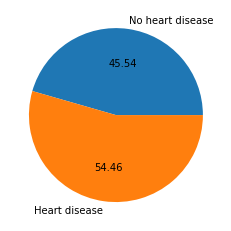
#grouping the data set according to presence/absence of heart diseases

dft = df.groupby('target').size()

import matplotlib.pyplot as plt

plt.pie(dft.values, labels = ["No heart disease", "Heart disease"],autopct='%.2f')

plt.show()

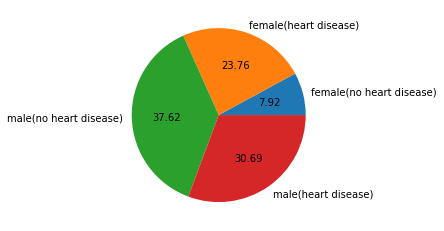


As we can see, the percentage of people having heart disease is more than those without any ailment in the given data set

dfv = df.groupby(['sex','target']).size()

plt.pie(dfv.values,labels = ['female(no heart disease)','female(heart disease)','male(no heart disease)','male(heart disease)'],autopct='%.2f')

plt.show()



Some risk factors may affect heart disease risk differently in women than in men. For example, estrogen provides women some protection against heart disease, but diabetes raises the risk of heart disease more in women than in men.

Obtaining Correlation Matrix

Correlation is a term that is a measure of the strength of a linear relationship between two quantitative variables. Positive correlation is a relationship between two variables in which both variables move in the same direction. This is when one variable increases while the other increases and vice versa. Whilst negative correlation is a relationship where one variable increases as the other decreases, and vice versa.

#correlation matrix

corr = df.corr()

plt.subplots(figsize=(15,10))

import seaborn as sns

sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns, annot=True, cmap=sns.diverging\_palette(220, 20, as\_cmap=True))

sns.heatmap(corr, xticklabels=corr.columns,

            yticklabels=corr.columns,

            annot=True,

            cmap=sns.diverging\_palette(220, 20, as\_cmap=True))



We can see there is a positive correlation between chest pain (cp) & target (our predictor). This makes sense since, the greater amount of chest pain results in a greater chance of having heart disease. In addition, we see a negative correlation between exercise induced angina (exang) & our predictor. This makes sense because when you exercise, your heart requires more blood, but narrowed arteries slow down blood flow.

Relation of age with presence of heart disease

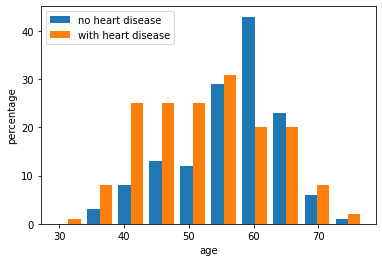
plt.hist([df[df.target==0].age, df[df.target==1].age],label = ["no heart disease","with heart disease"])

plt.xlabel("age")

plt.ylabel("percentage")

plt.legend()

plt.show()



According to the give data set, people in the age group of 40-60 are more prone to suffer from cardio-vascular diseases.

Relation of blood pressure with heart disease

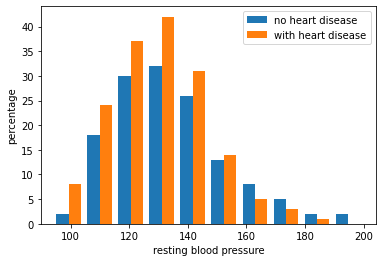
plt.hist([df[df.target==0].trestbps, df[df.target==1].trestbps],label = ["no heart disease","with heart disease"])

plt.xlabel("resting blood pressure")

plt.ylabel("percentage")

plt.legend()

plt.show()



High blood pressure can damage your arteries by making them less elastic, which decreases the flow of blood and oxygen to your heart and leads to heart disease. In addition, decreased blood flow to the heart can cause: Chest pain, also called angina.

Relation of cholesterol with heart diseases.

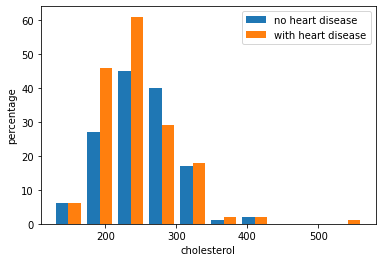
plt.hist([df[df.target==0].chol, df[df.target==1].chol],label = ["no heart disease","with heart disease"])

plt.xlabel("cholesterol")

plt.ylabel("percentage")

plt.legend()

plt.show()



The main risk associated with high cholesterol is coronary heart disease (CHD). Your blood cholesterol level has a lot to do with your chances of getting heart disease. If your cholesterol is too high, it builds up on the walls of your arteries. Over time, this buildup is known as atherosclerosis.

**IDENTIFYING AND REMOVING OUTLIERS**

Outliers are data points that are far from other data points. In other words, they’re unusual values in a dataset. Outliers are problematic for many statistical analyses because they can cause tests to either miss significant findings or distort real results.

Using Z-scores to Detect Outliers

Z-scores can quantify the unusualness of an observation when your data follow the normal distribution. Z-scores are the number of standard deviations above and below the mean that each value falls. For example, a Z-score of 2 indicates that an observation is two standard deviations above the average while a Z-score of -2 signifies it is two standard deviations below the mean. A Z-score of zero represents a value that equals the mean.

To calculate the Z-score for an observation, take the raw measurement, subtract the mean, and divide by the standard deviation. Mathematically, the formula for that process is the following:

z-score equation

The further away an observation’s Z-score is from zero, the more unusual it is. A standard cut-off value for finding outliers are Z-scores of +/-3 or further from zero. The probability distribution below displays the distribution of Z-scores in a standard normal distribution. Z-scores beyond +/- 3 are so extreme you can barely see the shading under the curve.

from scipy import stats

import numpy as np

threshold =3

z = np.abs(stats.zscore(df))

df.shape

(303, 14)

#Remoing all outliers uding Z score of 3

df = df[(z < 3).all(axis=1)]

df.shape

(287, 14)

**USING MACHINE LEARNING**

**Train-Test Split**

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model. It can be used for classification or regression problems and can be used for any supervised learning algorithm. The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

* Train Dataset: Used to fit the machine learning model.
* Test Dataset: Used to evaluate the fit machine learning model.

x = df.drop('target',axis=1)

y = df['target']

from sklearn.model\_selection import train\_test\_split

xtrain, xtest, ytrain, ytest = train\_test\_split(x, y, test\_size=0.25, random\_state=9)

**Parameters of train-test-split**

Test Size

The procedure has one main configuration parameter, which is the size of the train and test sets. This is most commonly expressed as a percentage between 0 and 1 for either the train or test datasets. Test size of 0.25 is taken here which means that 75% data is used as train set.

Random State

This is to check and validate the data when running the code multiple times. Setting random\_state a fixed value will guarantee that same sequence of random numbers are generated each time you run the code. And unless there is some other randomness present in the process, the results produced will be same as always. This helps in verifying the output.

**DEPLOYING SVM**

Support vector machines so called as SVM is a supervised learning algorithm which can be used for classification and regression problems. It can easily handle multiple continuous and categorical variables. SVM constructs a hyperplane in multidimensional space to separate different classes. SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error. The core idea of SVM is to find a maximum marginal hyperplane(MMH) that best divides the dataset into classes.

**SVM Kernels**

The SVM algorithm is implemented in practice using a kernel. A kernel transforms an input data space into the required form. SVM uses a technique called the kernel trick. Here, the kernel takes a low-dimensional input space and transforms it into a higher dimensional space. Linear Kernel A linear kernel can be used as normal dot product any two given observations. The product between two vectors is the sum of the multiplication of each pair of input values.

from sklearn import svm

smodel = svm.SVC(C=2,kernel='linear',gamma= 0.5)

**Parameters of SVC**

C

Inverse of the strength of regularization. Behavior: As the value of ‘C’ increases the model gets over-fits. As the value of ‘c’ decreases the model under-fits.

Gamma

A lower value of Gamma will loosely fit the training dataset, whereas a higher value of gamma will exactly fit the training dataset, which causes over-fitting.

smodel.fit(xtrain,ytrain)

SVC(C=2, break\_ties=False, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3, gamma=0.5, kernel='linear',

max\_iter=-1, probability=False, random\_state=None, shrinking=True,

tol=0.001, verbose=False)

**CONFUSION MATRIX**

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model.



from sklearn.metrics import confusion\_matrix

confusion\_matrix(ytrain,smodel.predict(xtrain))

array([[ 75, 19],

[ 8, 113]])

confusion\_matrix(ytest,smodel.predict(xtest))

array([[27, 8],

[ 2, 35]])

from sklearn import metrics

print("Accuracy:",metrics.accuracy\_score(ytest,smodel.predict(xtest)))

print("Precision:",metrics.precision\_score(ytest,smodel.predict(xtest)))

print("Recall:",metrics.recall\_score(ytest,smodel.predict(xtest)))

Accuracy: 0.8611111111111112

Precision: 0.813953488372093

Recall: 0.9459459459459459

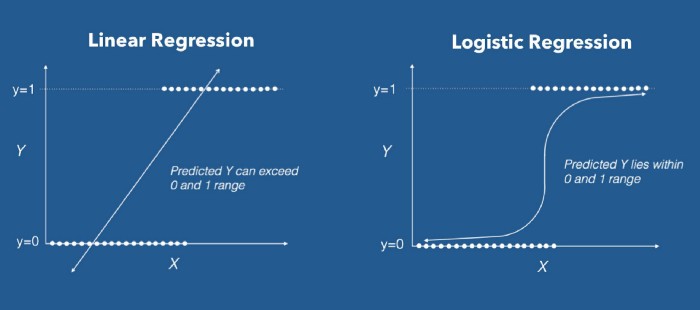
**Accuracy rate = 86.11%**

**Precision = 81.39%**

**Recall Rate - 94.59%**

**LOGISTIC REGRESSION MODEL**

Logistic Regression is a Machine Learning algorithm which is used for the classification problems, it is a predictive analysis algorithm and based on the concept of probability.



from sklearn.linear\_model import LogisticRegression

lmodel = LogisticRegression()

lmodel.fit(xtrain,ytrain)

from sklearn.metrics import mean\_absolute\_error

print(mean\_absolute\_error(ytrain,lmodel.predict(xtrain)))

print(mean\_absolute\_error(ytest,lmodel.predict(xtest)))

0.13023255813953488

0.1111111111111111

print(lmodel.score(xtrain,ytrain))

print(lmodel.score(xtest,ytest))

0.8697674418604651

0.8888888888888888

confusion\_matrix(ytrain,lmodel.predict(xtrain))

array([[ 74, 20],

[ 8, 113]])

confusion\_matrix(ytest,lmodel.predict(xtest))

array([[28, 7],

[ 1, 36]])

tp\_test,fn\_test,fp\_test,tn\_test = confusion\_matrix(ytest,lmodel.predict(xtest), labels=[1,0]).ravel()

tp\_test,tn\_test,fp\_test,fn\_test

(36, 28, 7, 1)

precision\_rate = tp\_test / (tp\_test + fp\_test)

recall\_rate = tp\_test / (tp\_test + fn\_test)

print("The precision rate is: ", precision\_rate)

print("The recall rate is: ", recall\_rate)

The precision rate is: 0.8372093023255814

The recall rate is: 0.972972972972973

**Accuracy rate = 88.88%**

**Precision = 83.72%**

**Recall Rate - 97.29%**

As we can see, Logistic Regression model gives better results with higher accuracy rate, precision value and recall rate.

**GIVING USER-DEFINED PARAMETERS TO CHECK THE TEST RESULTS**

input = np.zeros(len(xtrain.columns))

age = 55

sex = 1

cp = 0

trestbps = 140

chol = 250

fbs = 0

restecg = 0

thalach = 150

exang = 1

oldpeak = 1

slope = 1

ca = 0

thal = 2

input[0] = age

input[1] = sex

input[2] = cp

input[3] = trestbps

input[4] = chol

input[5] = fbs

input[6] = restecg

input[7] = thalach

input[8] = exang

input[9] = oldpeak

input[10] = slope

input[11] = ca

input[12] = thal

print(lmodel.predict([input])[0])

0

A result of 0 value signifies or points out to a prediction that the patient with the given parameters is at low risk of contracting cardiovascular heart diseases.

**CONCLUSION**

We can thus conclude that a given data set was used to train a practical model by splitting matrices into random train and test subsets and Logistical Regression algorithm deployed using supervised form of machine learning. This model can successfully diagnose the risk of heart disease in a patient using parameters such as age, sex, cholesterol etc and can classify patients with heart disease with a high accuracy rate of 88.88%.

# ****Acknowledgement****

1. Hungarian Institute of Cardiology. Budapest: Andras Janosi, M.D.
2. University Hospital, Zurich, Switzerland: William Steinbrunn, M.D.
3. University Hospital, Basel, Switzerland: Matthias Pfisterer, M.D.
4. V.A. Medical Center, Long Beach and Cleveland Clinic Foundation: Robert Detrano, M.D., Ph.D.

**BIBLIOGRAPHY**

1. colab.research.google.com

2. towardsdatascience.com

3. www.datacamp.com

4. stackoverflow.com

5. scikit-learn.org