Heart Disease Prediction Using Robust Algorithm

Abstract

Heart disease is a major life threatening disease that can cause either death or a serious long term disability. However, there is lack of effective tools to discover hidden relationships and trends in e-health data. Medicaldiagnosis is a complicated task and plays a vital role in saving human lives so it needs to be executed accurately and efficiently. An appropriate and accurate computer based automated decision support system is required to reduce cost for achieving clinical tests. This paper provides an insight into machine learning techniques used in diagnosing various diseases. Various data mining classifiers have been discussed which has emerged in recent years for efficient and effective disease diagnosis.

However using data mining technique can reduce the number of test that are required. In orderto reduce from heart diseases there have to be a quick and efficient detection technique. Decision Tree is one of the effective data mining methods used. This research compares different algorithms of Decision Tree classification seeking better performance in heart disease diagnosis. The algorithms which are tested are SVM algorithm, K Nearest Neighbour algorithm and Random Forest algorithm .

Decision Tree is one of the effective data mining methods used. This datasets consists of 303 instances and 76 attributes. Subsequently, the classification algorithm that has optimal potential will be suggested for use in sizeable data. The goal of this study is to extract hidden patterns by applying data mining techniques, which are noteworthy to heart diseases and to predict the presence of heart disease in patients where this presence is valued from no presence to likely presence.

Keywords: Machine Learning, Data Mining, Heart Disease, Diagnosis, Classification

Introduction to Machine Learning

Machine learning involves computer to get trained using a given data set, and use this training to predict the properties of a given new data. For example, we can train computer by feeding it 1000 images of cats and 1000 more images which are not of a cat, and tell each time to computer whether a picture is cat or not. Then if we show the computer a new image, then from the above training, computer should be able to tell whether this new image is cat or not.

Process of training and prediction involves use of specialized algorithms. We feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data.

There are various machine learning algorithms like Decision trees, Naive Bayes, Random forest, Support vector machine, K-nearest neighbour, K-means clustering, etc.

Machine Learning is the art (and science) of enabling machines to learn things which are not explicitly programmed.

It involves as much mathematics as much it involves computer science. Most often, people (also read as "sometimes me too") are put off by the sheer amount of mathematical equations and concepts in machine learning papers or articles that we ditch the entire article without reading.

In this series, I will *(possibly with the help of my friends, which will be duly noted in the respective articles)* talk about machine learning and deep learning math-free.

Purists might argue, learning is incomplete without the math behind it. I AGREE. But this is not intended to be a complete reference to the machine learning concepts, this series intends to start a conversation, or encourage thought in this direction.

- **Machine learning is one of the applications of python.**

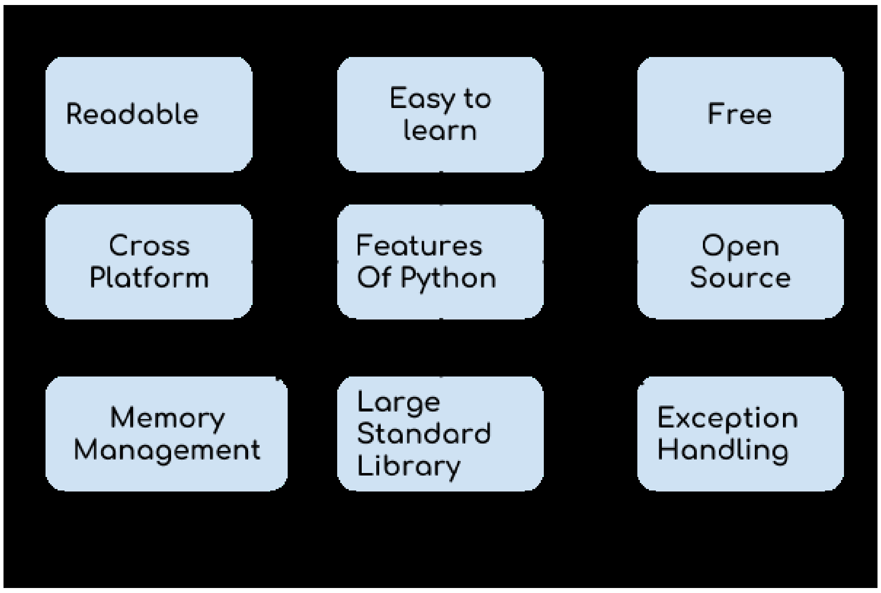
Python is widely used general-purpose, high-level programming language. It was initially designed by Guido van Rossum in 1991and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently. Python was designed for readability and has some similarities to the English language with influence from mathematics. Python uses new lines to complete a command as opposed to other programming language which often use semicolons or parentheses.

The most recent major version of Python is Python 3, which we shall be using in this tutorial. Python can be used on a server to create web applications. It can be used alongside software to create workflows. It can connect to database system and read and modify files.

Python can be used to handle big data and perform complex mathematics. It can be used for rapid prototyping, or for production ready software development. It works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc). It runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.

. Features of Python programming language



1. **Readable:** Python is a very readable language.

2. **Easy to Learn:** Learning python is easy as this is a expressive and high level programming language, which means it is easy to understand the language and thus easy to learn.

3. **Cross platform:** Python is available and can run on various operating systems such as Mac, Windows, Linux, Unix etc. This makes it a cross platform and portable language.

4. **Open Source:** Python is a open source programming language.

5. **Large standard library:** Python comes with a large standard library that has some handy codes and functions which we can use while writing code in Python.

6. **Free:** Python is free to download and use. This means you can download it for free and use it in your application. See: Open-Source Python License. Python is an example of a FLOSS (Free/Libre Open-Source Software), which means you can freely distribute copies of this software, read its source code and modify it.

7. **Supports exception handling:** If you are new, you may wonder what is an exception? Exception is an event that can occur during program exception and can disrupt the normal flow of program. Python supports exception handling which means we can write less error prone code and can test various scenarios that can cause an exception later on.

8. **Advanced features:** Supports generators and list comprehensions. We will cover these features later.

9. **Automatic memory management:** Python supports automatic memory management which means the memory is cleared and freed automatically. You do not have to bother clearing the memory.

- ## Applications of Python

1. **Web development** – Web framework like Django and Flask are based on Python. They help you write server-side code which helps you manage database, write backend programming logic, mapping URLs etc.

2**. Machine learning** – There are many machine learning applications written in Python. Machine learning is a way to write a logic so that a machine can learn and solve a particular problem on its own. For example, products recommendation in websites like Amazon, Flipkart, eBay etc. is a machine learning algorithm that recognizes user's interest. Face recognition and Voice recognition in your phone is another example of machine learning.

3**. Data Analysis** – Data analysis and data visualization in form of charts can also be developed using Python.4. **Scripting** – Scripting is writing small programs to automate simple tasks such as sending automated response emails etc. Such type of applications can also be written in Python programming language.

5**. Game development** – You can develop games using Python.

6. You can develop Embedded applications in Python.

7. **Desktop applications** – You can develop desktop application in Python using library like TKinter or QT.

Python is increasingly being used as a scientific language. Matrix and vector manipulations are extremely important for scientific computations. Both NumPy and Pandas have emerged to be essential libraries for any scientific computation, including machine learning, in python due to their intuitive syntax and high-performance matrix computation capabilities.

In this post, we will provide an overview of the common functionalities of NumPy and Pandas. We will realize the similarity of these libraries with existing toolboxes in R and MATLAB. This similarity and added flexibility have resulted in wide acceptance of python in the scientific community lately. Topic covered in the blog are:

- Overview of NumPy
- Overview of Pandas
- Using Matplotlib

This post is an excerpt from a live hands-on training conducted by CloudxLab on 25th Nov 2017. It was attended by more than 100 learners around the globe.

<u>NumPy:</u>

NumPy stands for 'Numerical Python' or 'Numeric Python'. It is an open-source module of Python which provides fast mathematical computation on arrays and matrices.

examples: A [2:5] will print items 2 to 4. Index in NumPy arrays starts from 0

A [2::2] will print items 2 to end skipping 2 items

A[:-1] will print the array in the reverse order

A [1:] will print from row 1 to end

Pandas:

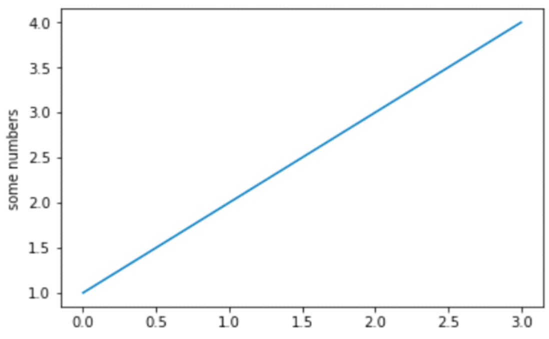
Similar to NumPy, Pandas is one of the most widely used python libraries in data science. It provides high-performance, easy to use structures and data analysis tools.

Matplotlib:

Matplotlib is a 2d plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments. Matplotlib can be used in Python scripts, Python and IPython shell, Jupyter Notebook, web application servers and GUI toolkits.

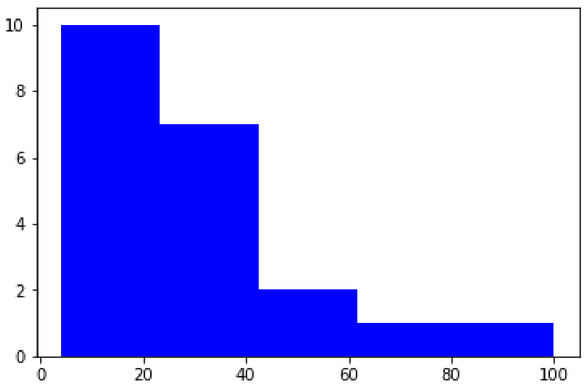
Example 1: Plotting a line graph

```
>>>import matplotlib.pyplot asplt
>>>plt.plot([1,2,3,4])
>>>plt.ylabel('some numbers')
>>>plt.show()
```



Example 2: Plotting a histogram

```
>>>import matplotlib.pyplot asplt
>>>x=[21,22,23,4,5,6,77,8,9,10,31,32,33
,34,35,36,37,18,49,50,100]
>>>num_bins=5
>>>plt.hist(x,num_bins,facecolor='blue')
>>>plt.show()
```



Objectives:

The Heart Disease Prediction application is an end user support and online consultation project. Here, we propose a web application that allows users to get instant guidance on their heart disease through an intelligent system online. The application is fed with various details and the heart disease associated with those details. The application allows user to share their heart related issues. It then processes user specific details to check for various illness that could be associated with it. Here we use some intelligent data mining techniques to guess the most accurate illness that could be associated with patient's details. Based on result, they can contact doctor accordingly for further treatment. The system allows user to view doctor's details too. The system can be used for free heart disease consulting online.
Heart disease is the leading cause of death in the world over the past 10 years (World Health Organization 2007). The European Public Health Alliance reported that heart attacks, strokes and other circulatory diseases account for 41% of all deaths (European Public Health Alliance 2010). Several different symptoms are associated with heart disease, which makes it difficult to diagnose it quicker and better.

Working on heart disease patients databases can be compared to real-life application. Doctors knowledge to assign the weight to each attribute. More weight is assigned to the attribute having high impact on disease prediction. Therefore it appears reasonable to try utilizing the knowledge and experience of several specialists collected in databases towards assisting the Diagnosis process. It also provides healthcare professionals an extra source of knowledge for making decisions.

Methodology:

1.Logistic Regression: Logistic Regression was used in the biological sciences in early twentieth century. It was then used in many social science applications. Logistic Regression is used when the dependent variable(target) is categorical.

For example,

- To predict whether an email is spam (1) or (0)
- Whether the tumor is malignant (1) or not (0)

Consider a scenario where we need to classify whether an email is spam or not. If we use linear regression for this problem, there is a need for setting up a threshold based on which classification can be done. Say if the actual class is malignant, predicted continuous value 0.4 and the threshold value is 0.5, the data point will be classified as not malignant which can lead to serious consequence in real time.

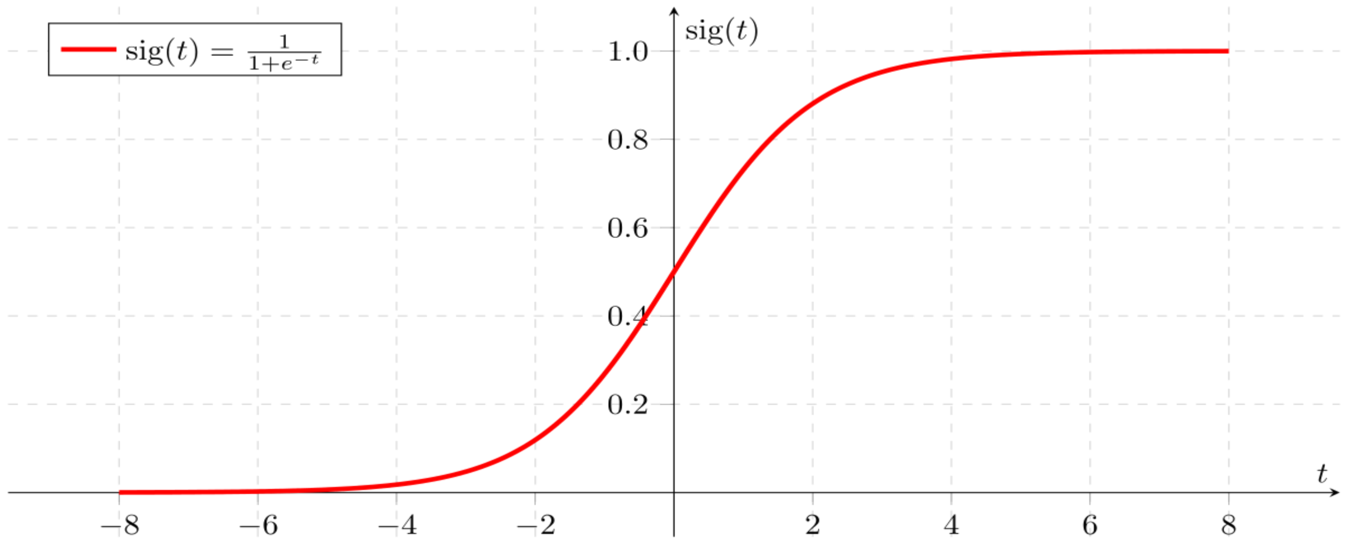
Model

Output = 0 or 1

Hypothesis => Z = WX + B

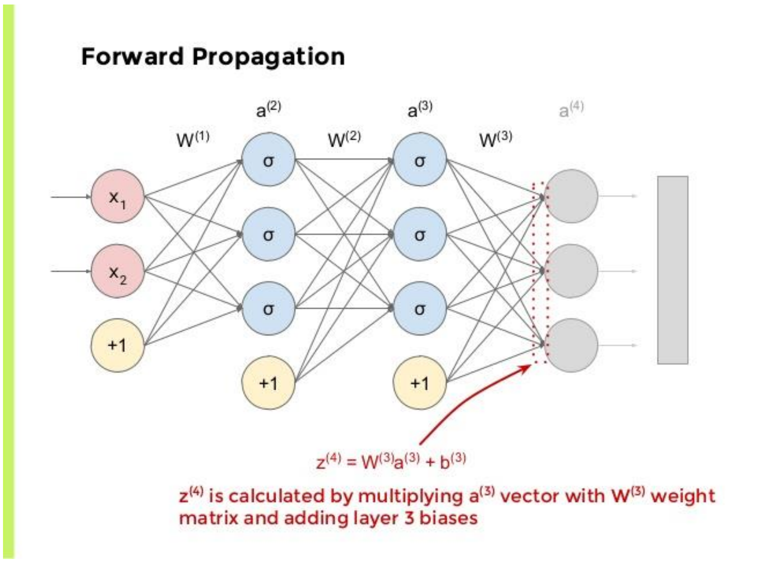
hΘ(x) = sigmoid (Z)

Sigmoid Function



If ‘Z’ goes to infinity, Y(predicted) will become 1 and if ‘Z’ goes to negative infinity, Y(predicted) will become 0.

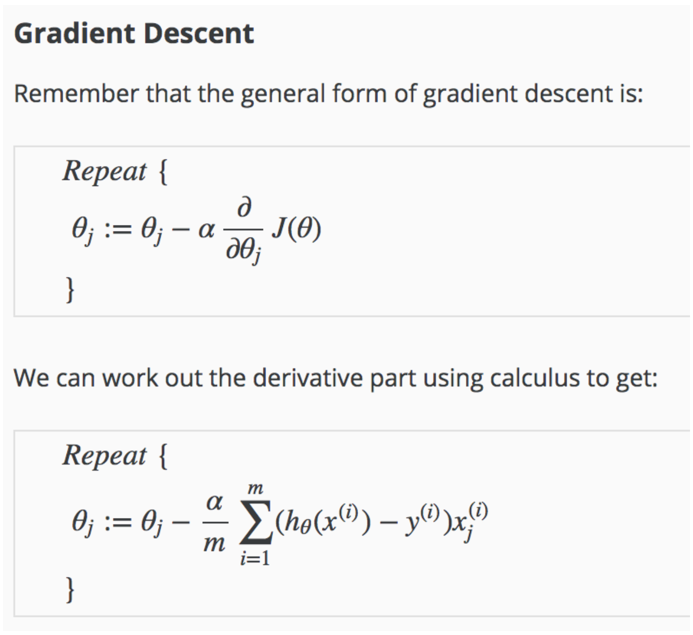
Forward and Backward Propagation


Forward Propagation
$a^{(2)}$
$a^{(3)}$
$a^{(4)}$
$W^{(1)}$
$W^{(2)}$
$W^{(3)}$
x_1
x_2
+1
σ
+1
$z^{(4)} = W^{(3)}a^{(3)} + b^{(3)}$
$z^{(4)}$ is calculated by multiplying $a^{(3)}$ vector with $W^{(3)}$ weight matrix and adding layer 3 biases

Cost function

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))]$$

Gradient descent



Gradient Descent

Remember that the general form of gradient descent is:

```
Repeat {
    $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$
}
```

We can work out the derivative part using calculus to get:

```
Repeat {
    $\theta_j := \theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$
}
```

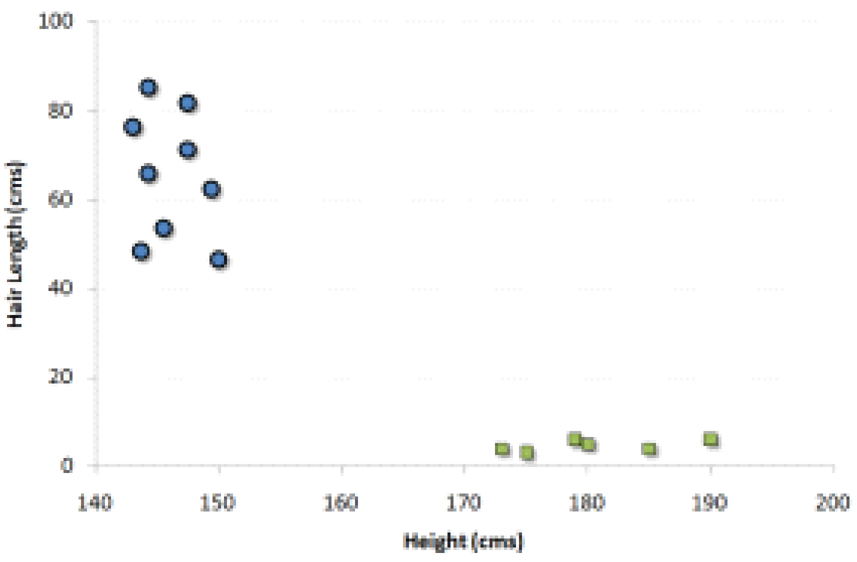
By the way in formulas.

- h0(x^i)= y_head
- y^i = y_train
- x^i = x_train

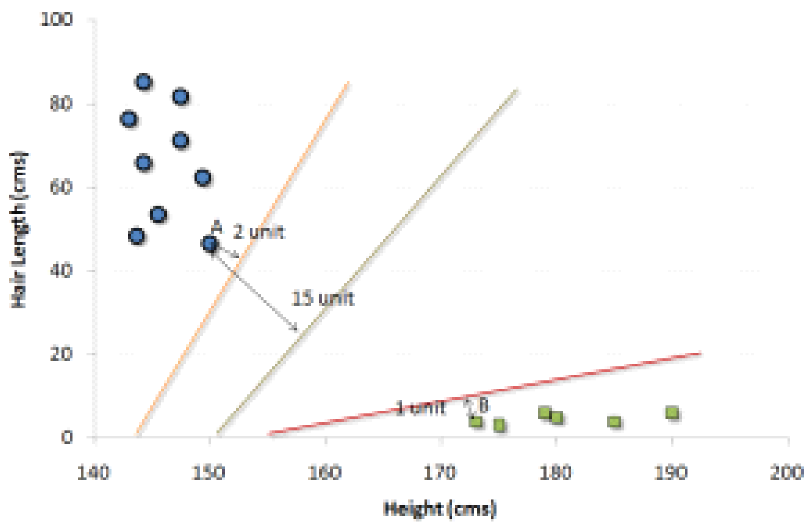
2.SVM (Support Vector Machine)

It is a classification method. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate.

For example, if we only had two features like Height and Hair length of an individual, we'd first plot these two variables in two-dimensional space where each point has two co-ordinates (these co-ordinates are known as **Support Vectors**)



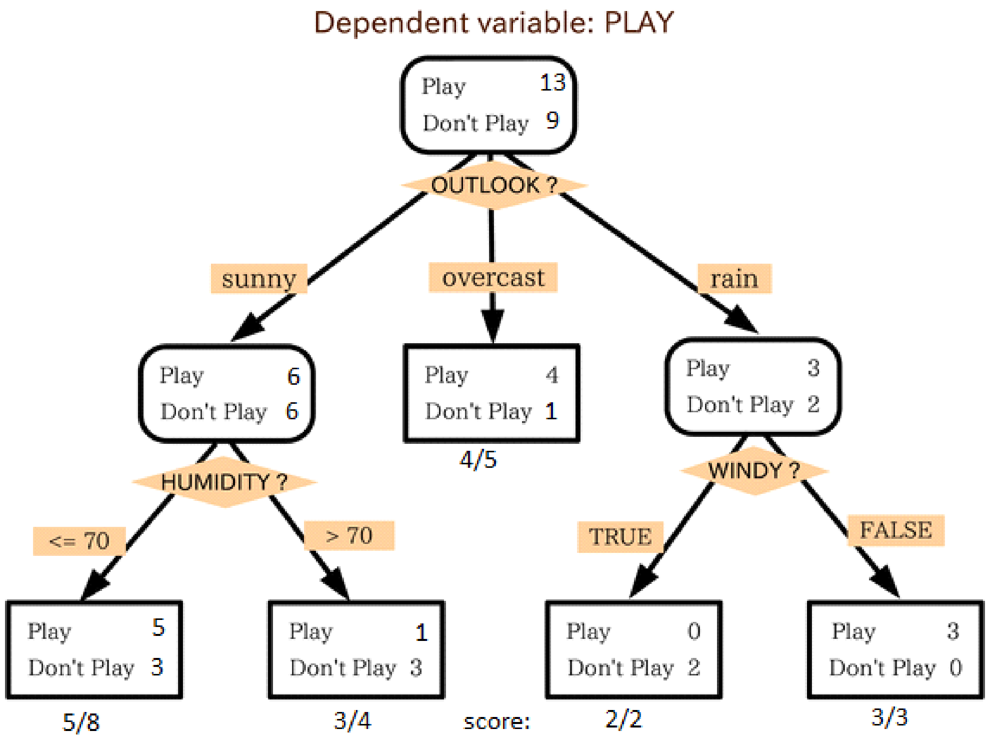
Now, we will find some *line* that splits the data between the two differently classified groups of data. This will be the line such that the distances from the closest point in each of the two groups will be farthest away.



For instance, orange frontier is closest to blue circles. And the closest blue circle is 2 units away from the frontier. Once we have these distances for all the frontiers, we simply choose the frontier with the maximum distance (from the closest support vector). Out of the three shown frontiers, we see the black frontier is farthest from nearest support vector (i.e. 15 units).

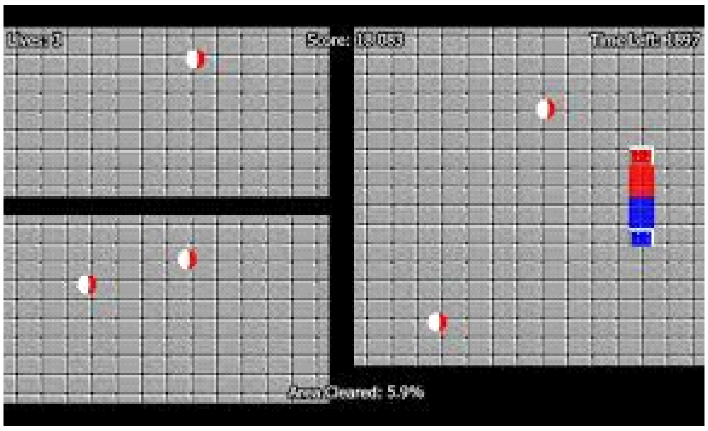
3.Decision Tree

This is one of my favorite algorithms and I use it quite frequently. It is a type of supervised learning algorithm that is mostly used for classification problems. Surprisingly, it works for both categorical and continuous dependent variables. In this algorithm, we split the population into two or more homogeneous sets. This is done based on most significant attributes/ independent variables to make as distinct groups as possible. For more details, you can read: Decision Tree Simplified.



In the image above, you can see that population is classified into four different groups based on multiple attributes to identify 'if they will play or not'. To split the population into different heterogeneous groups, it uses various techniques like Gini, Information Gain, Chi-square, entropy.

The best way to understand how decision tree works, is to play Jezzball – a classic game from Microsoft (image below). Essentially, you have a room with moving walls, and you need to create walls such that maximum area gets cleared off without the balls.



So, every time you split the room with a wall, you are trying to create 2 different populations within the same room. Decision trees work in very similar fashion by dividing a population in as different groups as possible.

4.KNN (K- Nearest Neighbors)

It can be used for both classification and regression problems. However, it is more widely used in classification problems in the industry. K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases by a majority vote of its k neighbors. The case being assigned to the class is most common amongst its K nearest neighbors measured by a distance function.

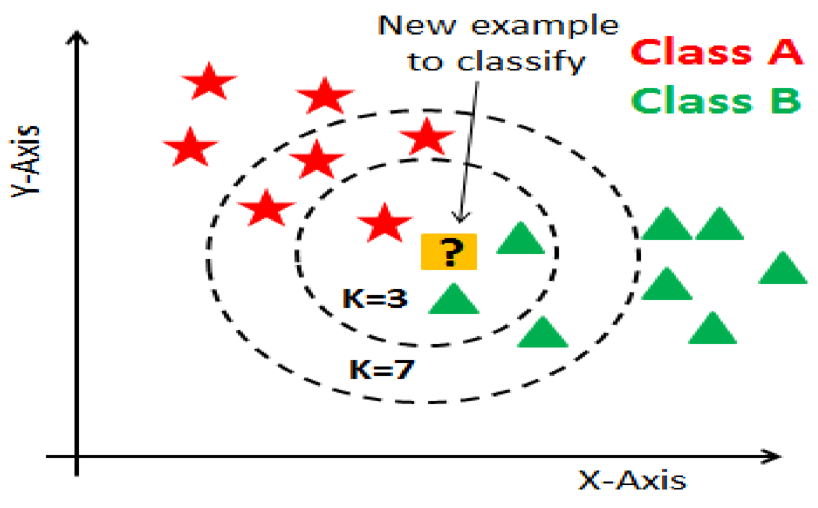
These distance functions can be Euclidean, Manhattan, Minkowski and Hamming distance. First three functions are used for continuous function and fourth one (Hamming) for categorical variables. If K = 1, then the case is simply assigned to the class of its nearest neighbor. At times, choosing K turns out to be a challenge while performing kNN modeling.

More: Introduction to k-nearest neighbors: Simplified.

KNN can easily be mapped to our real lives. If you want to learn about a person, of whom you have no information, you might like to find out about his close friends and the circles he moves in and gain access to his/her information!

Things to consider before selecting kNN:

KNN is computationally expensive.



Variables should be normalized else higher range variables can bias it.

Works on pre-processing stage more before going for kNN like outlier, noise removal.

5.Random Forest

Random Forest is a trademark term for an ensemble of decision trees. In Random Forest, we've collection of decision trees (so known as "Forest"). To classify a new object based on attributes, each tree gives a classification and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest).

Each tree is planted & grown as follows:

If the number of cases in the training set is N, then sample of N cases is taken at random but *with replacement*. This sample will be the training set for growing the tree.

If there are M input variables, a number m<<M is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.

Each tree is grown to the largest extent possible. There is no pruning.

Task Performed:

Dataset Description

We performed computer simulation on one dataset. Dataset is a Heart dataset. The dataset is available in UCI Machine Learning Repository [10]. Dataset contains 303 samples and 14 input features as well as 1 output feature. The features describe financial, personal, and social feature of loan applicants. The output feature is the decision class which has value 1 for Good credit and 2 for Bad credit. The dataset-1 contains 700 instances shown as Good credit while 300 instances as bad credit. The dataset contains features expressed on nominal, ordinal, or interval scales. A list of all those features is given in Table .

Feature No.	Feature Name
1	Age
2	Sex
3	Cp
4	Trestbps
5	Choi
6	Fbs
7	Restesg
8	Thalach
9	Exang
10	Oldpeak
11	Slop
12	Ca
13	Thal
14	Num

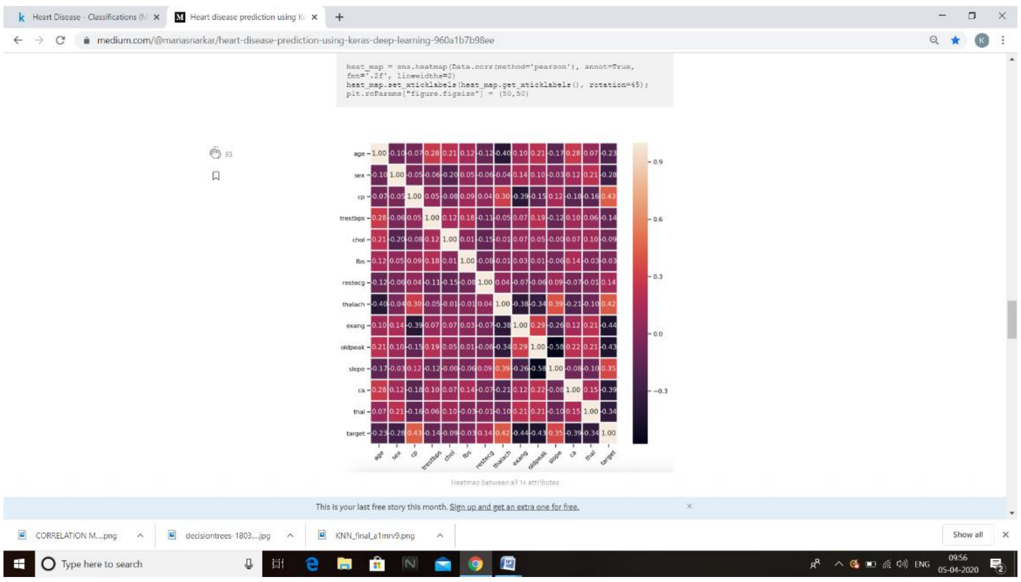
TABLE: SELECTED HEART DISEASE ATTRIBUTES

Name	**Type**	**Description**
Age	continuous	Age in years
Sex	discrete	0=female,1=male
Cp	discrete	Chest pain type: 1=typical angina, 2=a typical angina, 3=non anginal pain, 4=asymptom
trestbpc	continuous	Resting blood Pressure (in mm Hg)
Chol	continuous	Serum cholesterol In mg/dl
Fbs	discrete	Fasting blood Suger > 120 mg/dl: 1=true,0=false
Exang Continuous Maximum heart rate achieved	discrete	Exercise induced angina: 1=Yes, 0=No
Thalach	continuous	Maximum heart rate achieved
Oldpeak ST	continuous	Depression induced by exercise relative to rest
slope	discrete	The slope of the Peak excersise Segment : 1=up sloping, 2=flat, 3=down slopping
Ca	continuous	No.of measure Vessels colored by Fluoroscopy that Ranged between 0 & 3
Thal	discrete	3 = Normal,

		6 = Fixed Defect, 7=ReversibleDefect
class	discrete	Diagnosis classes: 0=No presence, 1=least likely to have heart disease, 2=>1,3=>2,4=more likely have heart disease

- **Snapshot of Correlation Matrix**

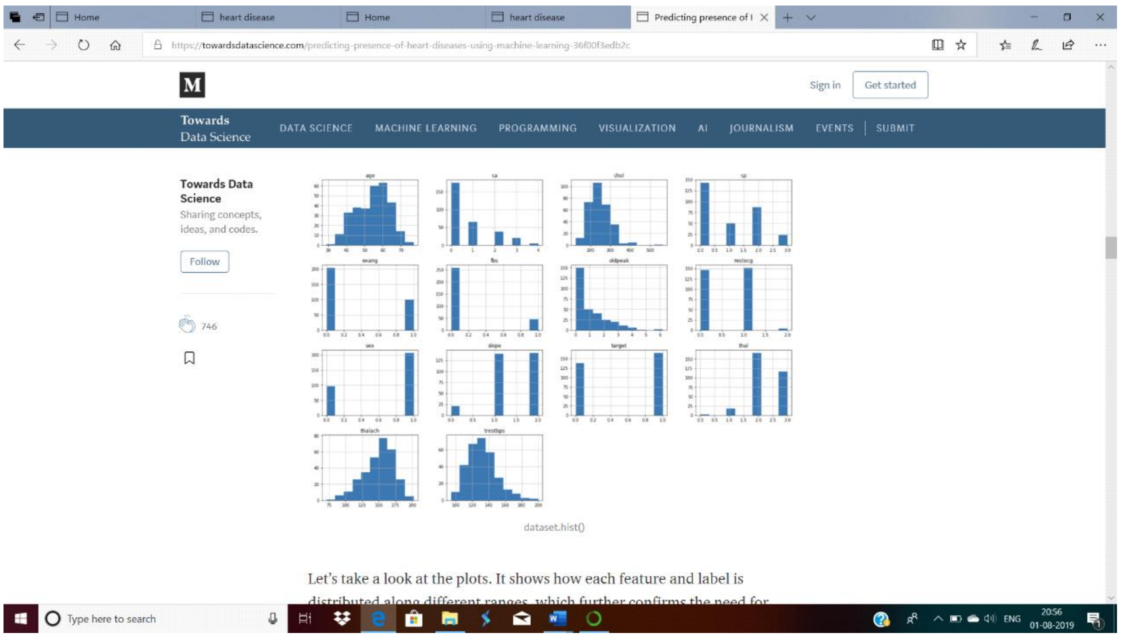
The correlation coefficient is also known as the Pearson product-moment correlation coefficient, or Pearson's correlation coefficient. A correlation matrix is a table showing correlation coefficients between sets of variables. Each random variable (X_i) in the table is correlated with each of the other values in the table (X_j). This allows you to see which pairs have the highest correlation.


```
heat_map = sns.heatmap(Data.corr(method='pearson'), annot=True,
fmt='.2f', linewidths=2)
heat_map.set_xticklabels(heat_map.get_xticklabels(), rotation=45);
plt.rcParams["figure.figsize"] = (50,50)
```
Heart Disease - Classifications
Heart disease prediction using K
medium.com/@manasnarkar/heart-disease-prediction-using-keras-deep-learning-960a1b7b98ee
age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
target
0.9
0.6
0.3
0.0
-0.3
Heatmap between all 14 attributes
This is your last free story this month. Sign up and get an extra one for free.
CORRELATION M....png
decisiontrees-1803....jpg
KNN_final_a1mrv9.png
Show all
Type here to search
ENG
09:56
05-04-2020

As you can observe there is no strong correlation between any of the 14 attributes.

- **Snapshot of dataset. History**

The best part about this type of plot is that it just takes a single command to draw the plots and it provides so much information in return. Just use dataset.hist().Let's take a look at the plots. It shows how each feature and label is distributed along different ranges, which further confirms the need for scaling. Next, wherever you see discrete bars, it basically means that each of these is actually a categorical variable. We will need to handle these categorical variables before applying Machine Learning. Our target labels have two classes, 0 for no disease and 1 for disease.



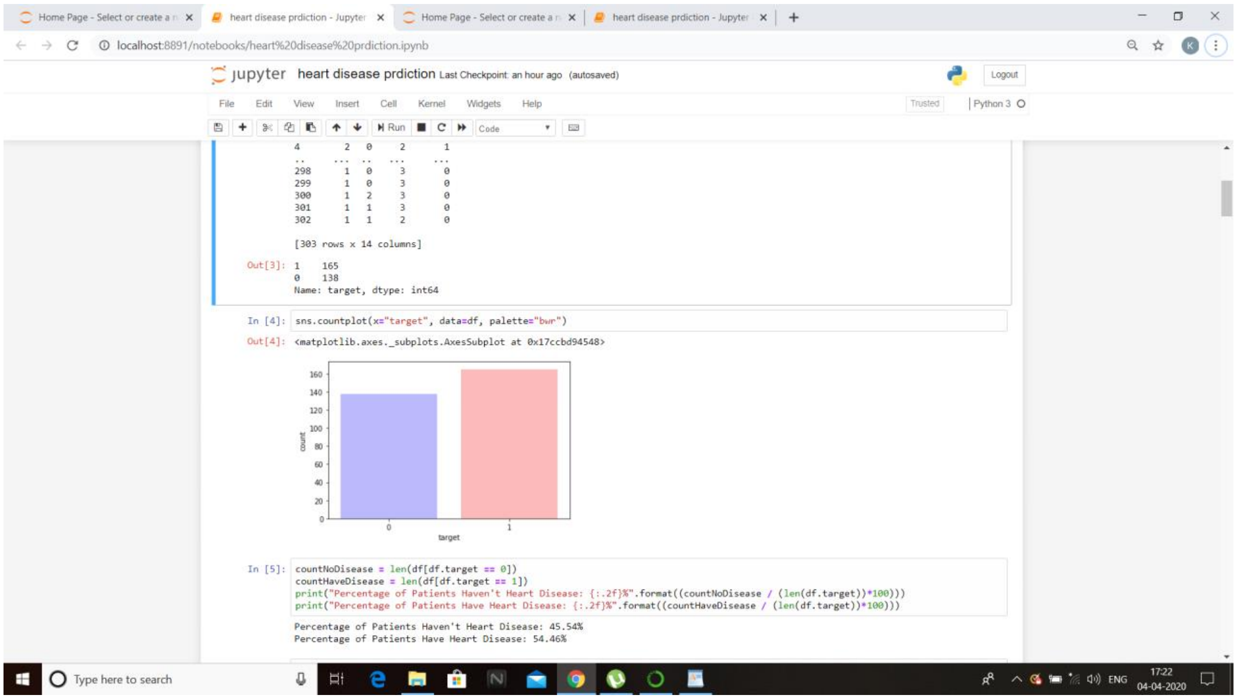
dataset.hist()

Let's take a look at the plots. It shows how each feature and label is distributed along different ranges, which further confirms the need for

- **Snapshot of Target Classes**

It's really essential that the dataset we are working on should be approximately balanced. An extremely imbalanced dataset can render the whole model training useless and thus, will be of no use. Let's understand it with an example.

Let's say we have a dataset of 100 people with 99 non-patients and 1 patient. Without even training and learning anything, the model can always say that any new person would be a non-patient and have an accuracy of 99%. However, as we are more interested in identifying the 1 person who is a patient, we need balanced datasets so that our model actually learns.


Home Page - Select or create a n
heart disease prdiction - Jupyter
localhost:8891/notebooks/heart%20disease%20prdiction.ipynb
jupyter heart disease prdiction Last Checkpoint: an hour ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code
count
target

```
4         2   0     2       1
..      ...  ..   ...     ...
298       1   0     3       0
299       1   0     3       0
300       1   2     3       0
301       1   1     3       0
302       1   1     2       0

[303 rows x 14 columns]

Out[3]: 1    165
        0    138
        Name: target, dtype: int64

In [4]: sns.countplot(x="target", data=df, palette="bwr")

Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x17ccbd94548>

In [5]: countNoDisease = len(df[df.target == 0])
        countHaveDisease = len(df[df.target == 1])
        print("Percentage of Patients Haven't Heart Disease: {:.2f}%".format((countNoDisease / (len(df.target))*100)))
        print("Percentage of Patients Have Heart Disease: {:.2f}%".format((countHaveDisease / (len(df.target))*100)))

        Percentage of Patients Haven't Heart Disease: 45.54%
        Percentage of Patients Have Heart Disease: 54.46%
```

From the plot, we can see that the classes are almost balanced and we are good to proceed with data processing

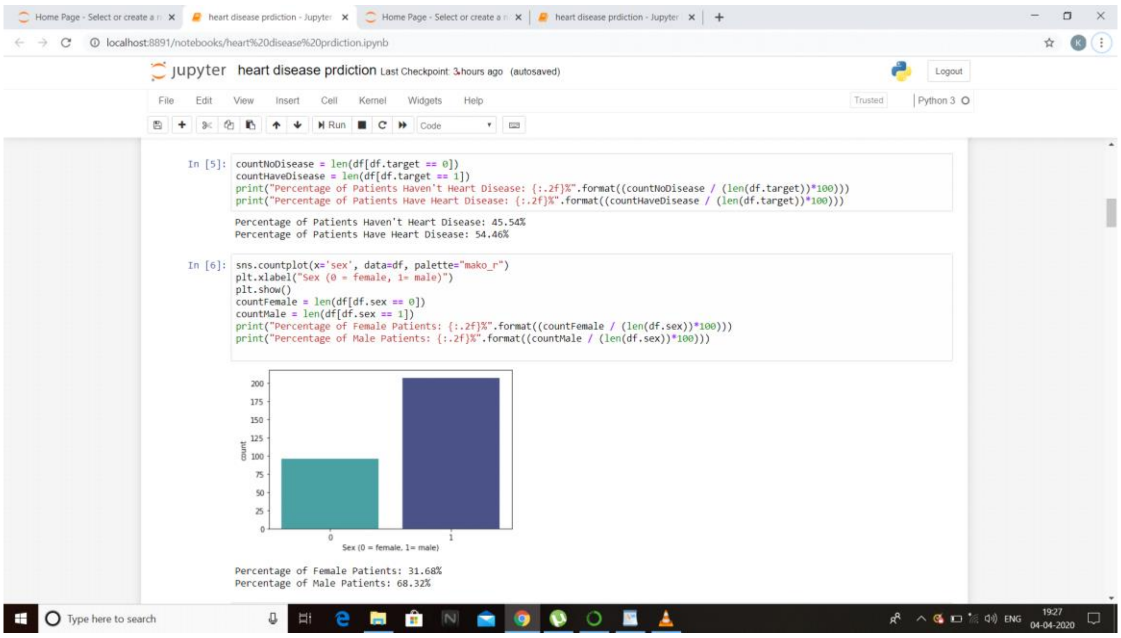
- **Snapshot of Persons having Heart disease (based on the 'Sex' ,'Age','Heart Rate' and 'Fasting blood sugar')**

  The count of number of heart disease patients to count of not having the disease is as replicated through the percentages as given with

  Percentage of Patients Haven't Heart Disease: 45.54%

  Percentage of Patients Have Heart Disease: 54.46%

  And now as such these counts are related to the sex of the person as represented by a bar graph



Home Page - Select or create a n
heart disease prdiction - Jupyter
localhost:8891/notebooks/heart%20disease%20prdiction.ipynb
jupyter
heart disease prdiction Last Checkpoint: 3 hours ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code
count
Sex (0 = female, 1= male)
Type here to search
ENG
19:27
04-04-2020

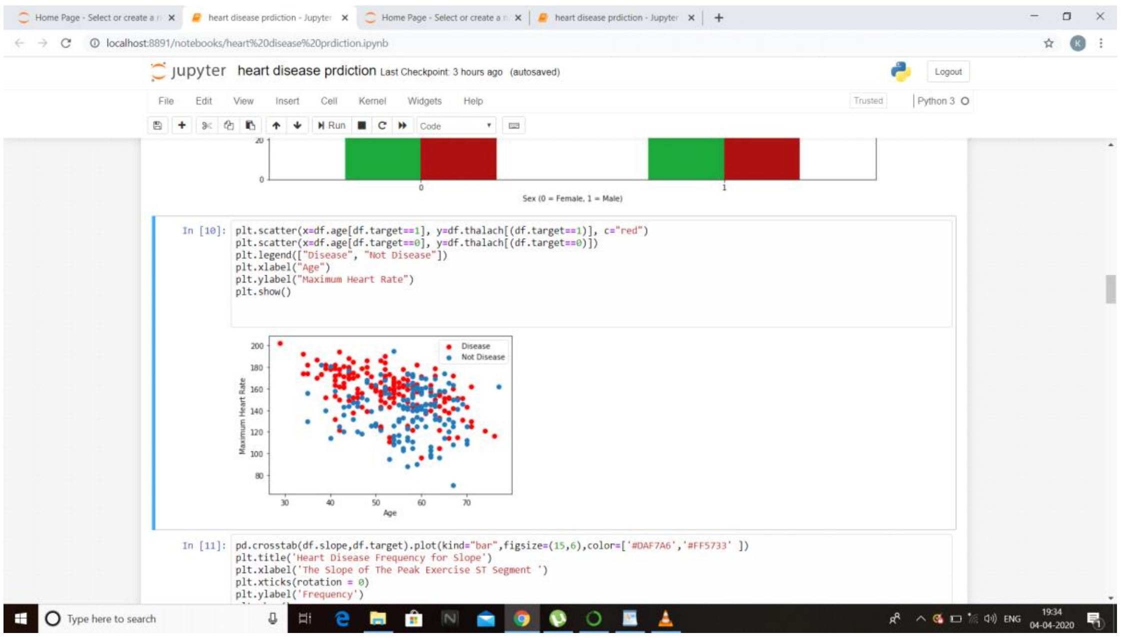
```
In [5]: countNoDisease = len(df[df.target == 0])
        countHaveDisease = len(df[df.target == 1])
        print("Percentage of Patients Haven't Heart Disease: {:.2f}%".format((countNoDisease / (len(df.target))*100)))
        print("Percentage of Patients Have Heart Disease: {:.2f}%".format((countHaveDisease / (len(df.target))*100)))

        Percentage of Patients Haven't Heart Disease: 45.54%
        Percentage of Patients Have Heart Disease: 54.46%

In [6]: sns.countplot(x='sex', data=df, palette="mako_r")
        plt.xlabel("Sex (0 = female, 1= male)")
        plt.show()
        countFemale = len(df[df.sex == 0])
        countMale = len(df[df.sex == 1])
        print("Percentage of Female Patients: {:.2f}%".format((countFemale / (len(df.sex))*100)))
        print("Percentage of Male Patients: {:.2f}%".format((countMale / (len(df.sex))*100)))

        Percentage of Female Patients: 31.68%
        Percentage of Male Patients: 68.32%
```

The target forms of the datasets forms a major role of class classification with 1 or 0.

So as such no comaparing the systems of the datasets i.e; the comparison of the 'maximum heart rate 'to that of a age of a particular person stands alone to be peculiar in the graph shown.

As according to the graph we can see that the majority of the people with heart rate above a threshold value are at a age range of 40-52. And declared to be a heart disease patient,conversly to the peole at the age range of

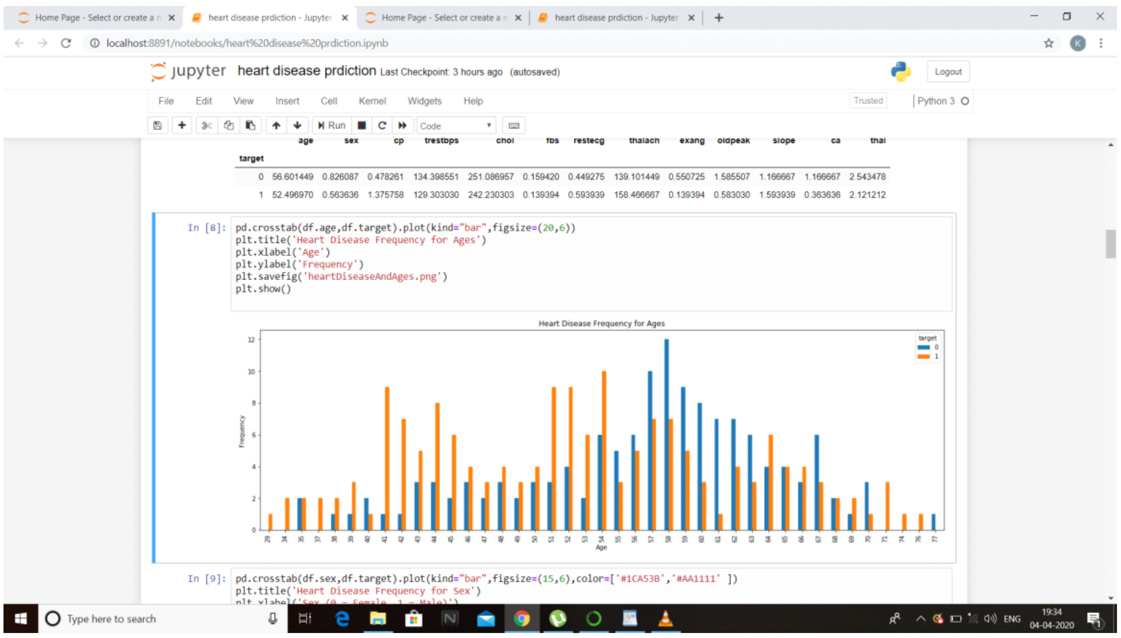
50-67 are having lower heart rate and termed to be a non diseased patients



```
In [10]: plt.scatter(x=df.age[df.target==1], y=df.thalach[(df.target==1)], c="red")
         plt.scatter(x=df.age[df.target==0], y=df.thalach[(df.target==0)])
         plt.legend(["Disease", "Not Disease"])
         plt.xlabel("Age")
         plt.ylabel("Maximum Heart Rate")
         plt.show()
```

```
In [11]: pd.crosstab(df.slope,df.target).plot(kind="bar",figsize=(15,6),color=['#DAF7A6','#FF5733' ])
         plt.title('Heart Disease Frequency for Slope')
         plt.xlabel('The Slope of The Peak Exercise ST Segment ')
         plt.xticks(rotation = 0)
         plt.ylabel('Frequency')
```

The problem still remains unknown as to how the data's gets treated with, so on here with a outlier as a 'target' We take a step in creating it to be a factor of classification ,which intends for the removal of most common possibilities,as a result we see the frequency of heart disease with respect to ages as it merges to a probabilistic value .

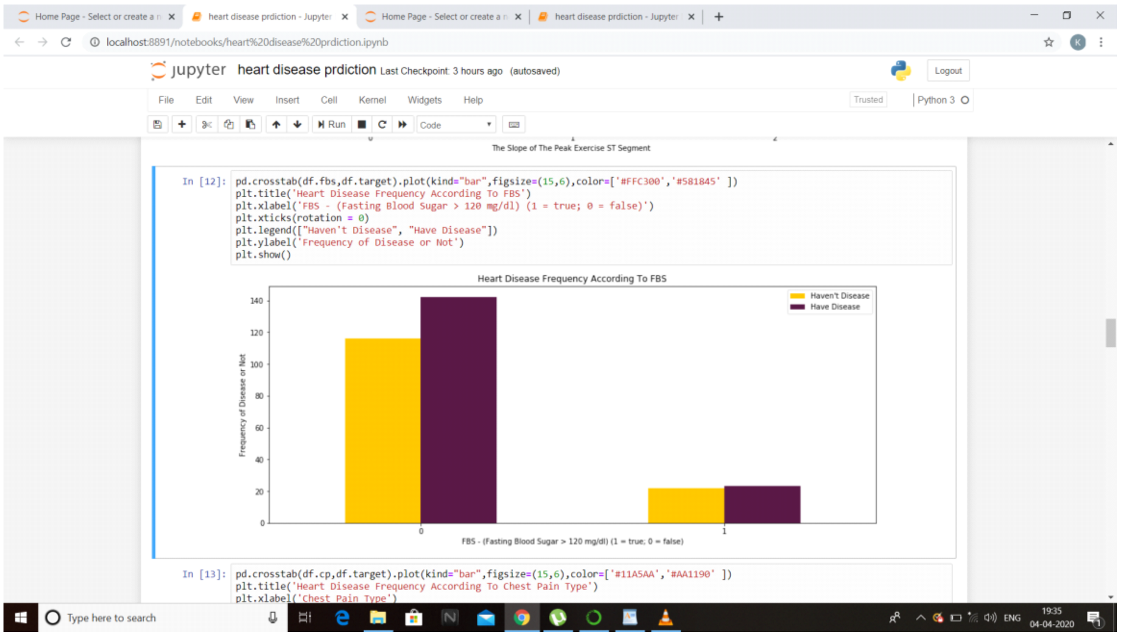


target	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	56.601449	0.826087	0.478261	134.398551	251.086957	0.159420	0.449275	139.101449	0.550725	1.585507	1.166667	1.166667	2.543478
1	52.496970	0.563636	1.375758	129.303030	242.230303	0.139394	0.593939	158.466667	0.139394	0.583030	1.593939	0.363636	2.121212

```
In [8]: pd.crosstab(df.age,df.target).plot(kind="bar",figsize=(20,6))
        plt.title('Heart Disease Frequency for Ages')
        plt.xlabel('Age')
        plt.ylabel('Frequency')
        plt.savefig('heartDiseaseAndAges.png')
        plt.show()
```

```
In [9]: pd.crosstab(df.sex,df.target).plot(kind="bar",figsize=(15,6),color=['#1CA53B','#AA1111' ])
        plt.title('Heart Disease Frequency for Sex')
```

As a classification factor know we come across the "fasting blood sugar" which forms a basis of the info of system with blood sugar having a level of >120mg/dl are intended to have a major chances of heart disease condition with a target value conferenced to be 1 for such case, and conversly without a heart disease .



```
In [12]: pd.crosstab(df.fbs,df.target).plot(kind="bar",figsize=(15,6),color=['#FFC300','#581845' ])
         plt.title('Heart Disease Frequency According To FBS')
         plt.xlabel('FBS - (Fasting Blood Sugar > 120 mg/dl) (1 = true; 0 = false)')
         plt.xticks(rotation = 0)
         plt.legend(["Haven't Disease", "Have Disease"])
         plt.ylabel('Frequency of Disease or Not')
         plt.show()
```

The Slope of The Peak Exercise ST Segment
Heart Disease Frequency According To FBS
Haven't Disease
Have Disease
Frequency of Disease or Not
FBS - (Fasting Blood Sugar > 120 mg/dl) (1 = true; 0 = false)

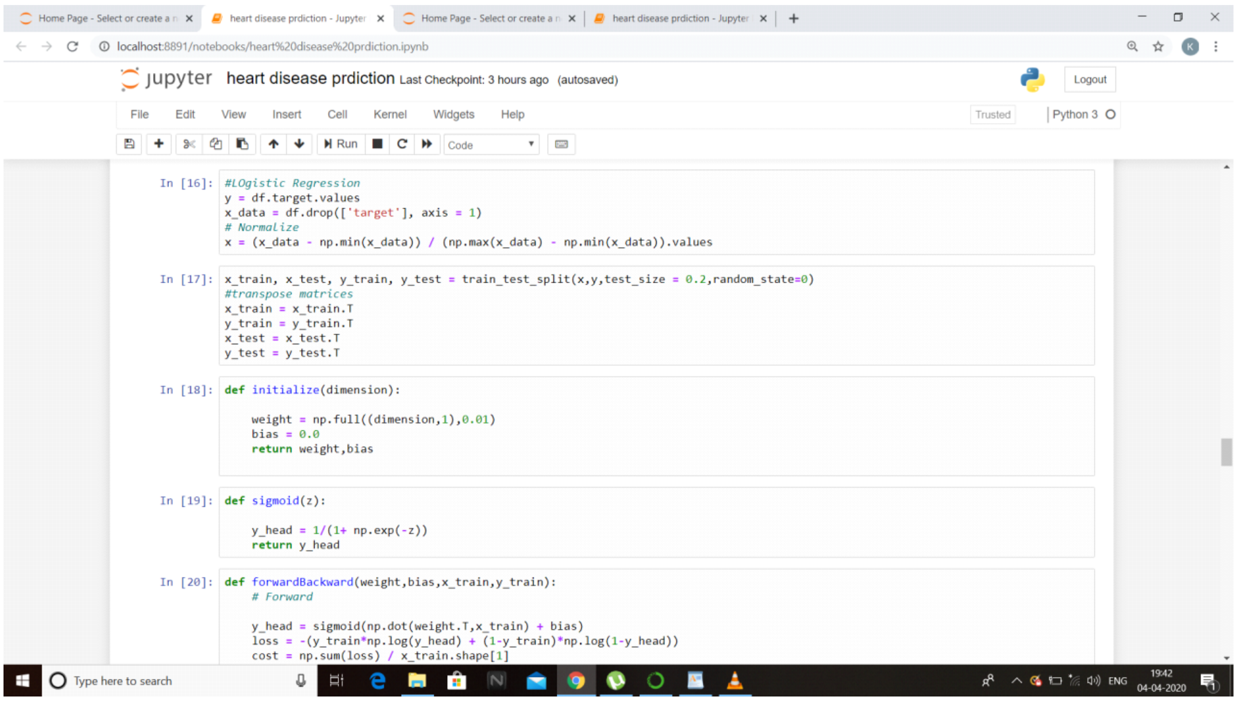
```
In [13]: pd.crosstab(df.cp,df.target).plot(kind="bar",figsize=(15,6),color=['#11A5AA','#AA1190' ])
         plt.title('Heart Disease Frequency According To Chest Pain Type')
         plt.xlabel('Chest Pain Type')
```

- **Snapshot of Logistic Regression classification with Sklearn**

  To predict which class a data belongs, a threshold can be set. Based upon this threshold, the obtained estimated probability is classified into classes.

  Say, if predicted_value ≥ 0.5, then classify it to have heart disease or not.

  Decision boundary can be linear or non-linear. Polynomial order can be increased to get complex decision boundary.



```
In [16]: #LOgistic Regression
         y = df.target.values
         x_data = df.drop(['target'], axis = 1)
         # Normalize
         x = (x_data - np.min(x_data)) / (np.max(x_data) - np.min(x_data)).values

In [17]: x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2,random_state=0)
         #transpose matrices
         x_train = x_train.T
         y_train = y_train.T
         x_test = x_test.T
         y_test = y_test.T

In [18]: def initialize(dimension):

             weight = np.full((dimension,1),0.01)
             bias = 0.0
             return weight,bias

In [19]: def sigmoid(z):

             y_head = 1/(1+ np.exp(-z))
             return y_head

In [20]: def forwardBackward(weight,bias,x_train,y_train):
             # Forward

             y_head = sigmoid(np.dot(weight.T,x_train) + bias)
             loss = -(y_train*np.log(y_head) + (1-y_train)*np.log(1-y_head))
             cost = np.sum(loss) / x_train.shape[1]
```

We will split our data. 80% of our data will be train data and 20% of it will be test data.

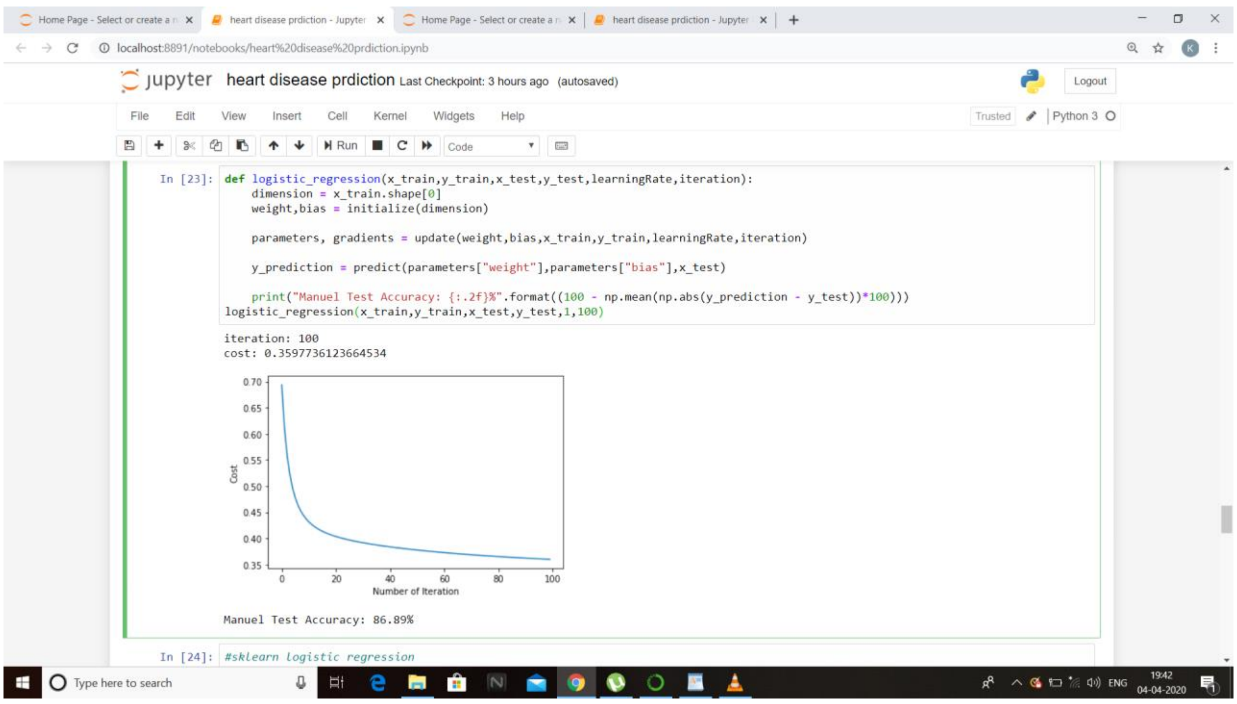
Once the features and labeled are separated, we're ready to split the data into train and test sets. This will separate 25%(default value) of the data into a subset for testing part and the remaining 75% will be used for our training subset.

```
#transpose matrices
x_train = x_train.T
y_train = y_train.T
x_test = x_test.T
y_test = y_test.T
```

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2,random_state=0)
```

Once the model is defined, we can work to fit our data. We're going to use the fit method on the model to train the data. Note that the fit method takes two parameters here: variables x_train and y_train. We created these variables we created in some previous code using the train_test_split function.

Once model training is complete, its time to predict the data using the model. For this, we're going to use the predict method on the model and pass the x_test values for predicting. We're storing the predicted values in the y_pred variable.



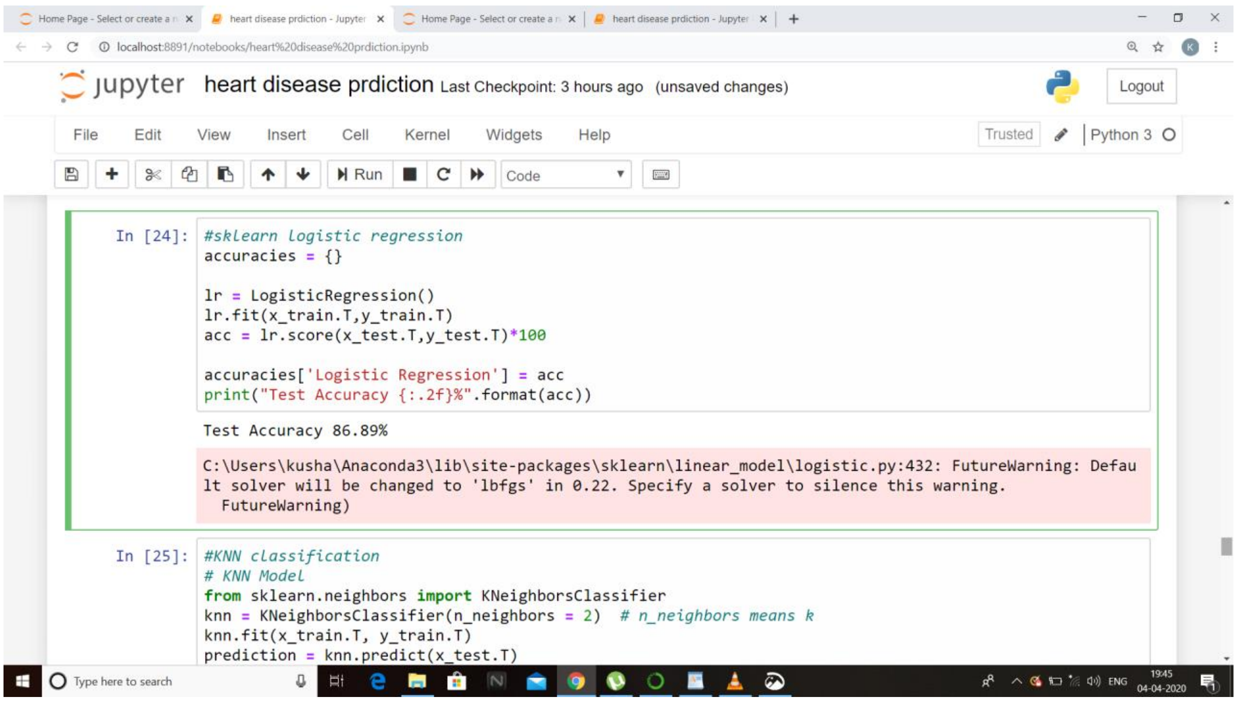
```
In [23]: def logistic_regression(x_train,y_train,x_test,y_test,learningRate,iteration):
             dimension = x_train.shape[0]
             weight,bias = initialize(dimension)

             parameters, gradients = update(weight,bias,x_train,y_train,learningRate,iteration)

             y_prediction = predict(parameters["weight"],parameters["bias"],x_test)

             print("Manuel Test Accuracy: {:.2f}%".format((100 - np.mean(np.abs(y_prediction - y_test))*100)))
         logistic_regression(x_train,y_train,x_test,y_test,1,100)

iteration: 100
cost: 0.3597736123664534
```

heart disease prdiction Last Checkpoint: 3 hours ago (autosaved)
localhost:8891/notebooks/heart%20disease%20prdiction.ipynb
Cost
Number of Iteration

```
Manuel Test Accuracy: 86.89%

In [24]: #sklearn logistic regression
```

Linear regression uses mean squared error as its cost function. If this is used for logistic regression, then it will be a non-convex function of parameters (theta). Gradient descent will converge into global minimum only if the

function is convex.


jupyter heart disease prdiction Last Checkpoint: 3 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [24]: #sklearn logistic regression
         accuracies = {}

         lr = LogisticRegression()
         lr.fit(x_train.T,y_train.T)
         acc = lr.score(x_test.T,y_test.T)*100

         accuracies['Logistic Regression'] = acc
         print("Test Accuracy {:.2f}%".format(acc))

         Test Accuracy 86.89%

         C:\Users\kusha\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Defau
         lt solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
           FutureWarning)

In [25]: #KNN classification
         # KNN Model
         from sklearn.neighbors import KNeighborsClassifier
         knn = KNeighborsClassifier(n_neighbors = 2)  # n_neighbors means k
         knn.fit(x_train.T, y_train.T)
         prediction = knn.predict(x_test.T)
```

As on implementing it with a sklearn a accuracy of 86.89% is estimated

- **Snapshot of KNN Classifier scores**

The KNN Algorithm

1. Load the data

2. Initialize K to your chosen number of neighbors(from 1-20)

3. For each example in the data

3.1 Calculate the distance between the query example and the current example from the data.

3.2 Add the distance and the index of the example to an ordered collection

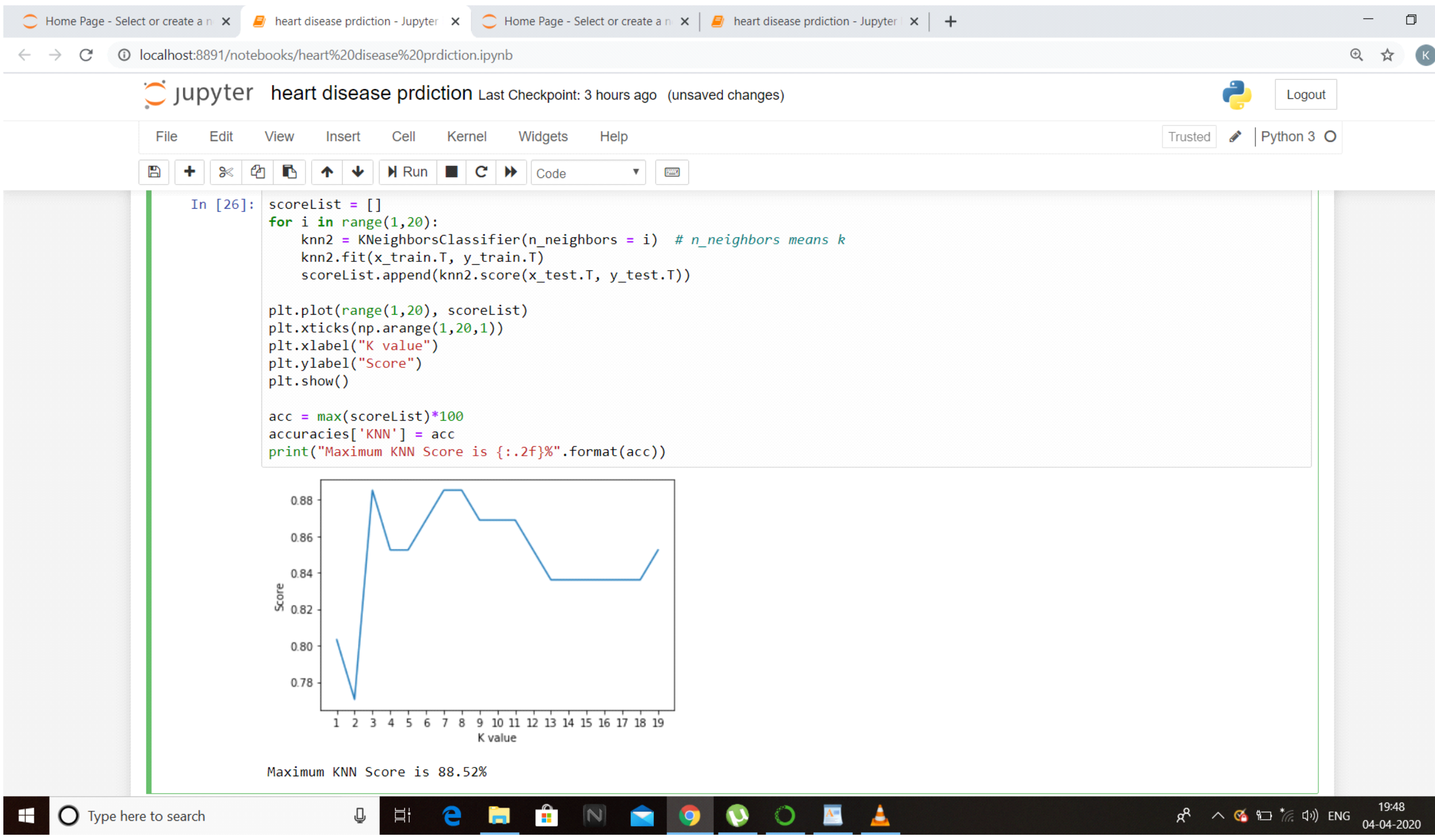
4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances

5. Pick the first K entries from the sorted collection

6. Get the labels of the selected K entries(update the score list)

The updated score list is now compared or cross verified with each k - value and the resulting output as to how

the value differs across the score list is examined.and the accuracy of the classifier system is found to be 88.52%


Home Page - Select or create a n
heart disease prdiction - Jupyter
localhost:8891/notebooks/heart%20disease%20prdiction.ipynb
jupyter heart disease prdiction Last Checkpoint: 3 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Score
K value
Type here to search
ENG
19:48
04-04-2020

```
In [26]: scoreList = []
         for i in range(1,20):
             knn2 = KNeighborsClassifier(n_neighbors = i)  # n_neighbors means k
             knn2.fit(x_train.T, y_train.T)
             scoreList.append(knn2.score(x_test.T, y_test.T))

         plt.plot(range(1,20), scoreList)
         plt.xticks(np.arange(1,20,1))
         plt.xlabel("K value")
         plt.ylabel("Score")
         plt.show()

         acc = max(scoreList)*100
         accuracies['KNN'] = acc
         print("Maximum KNN Score is {:.2f}%".format(acc))
```

```
Maximum KNN Score is 88.52%
```

- **Snapshot of 'SVM' , 'Naive Bayes' , 'Decision Tree' and 'Random Forest' Classifier scores**

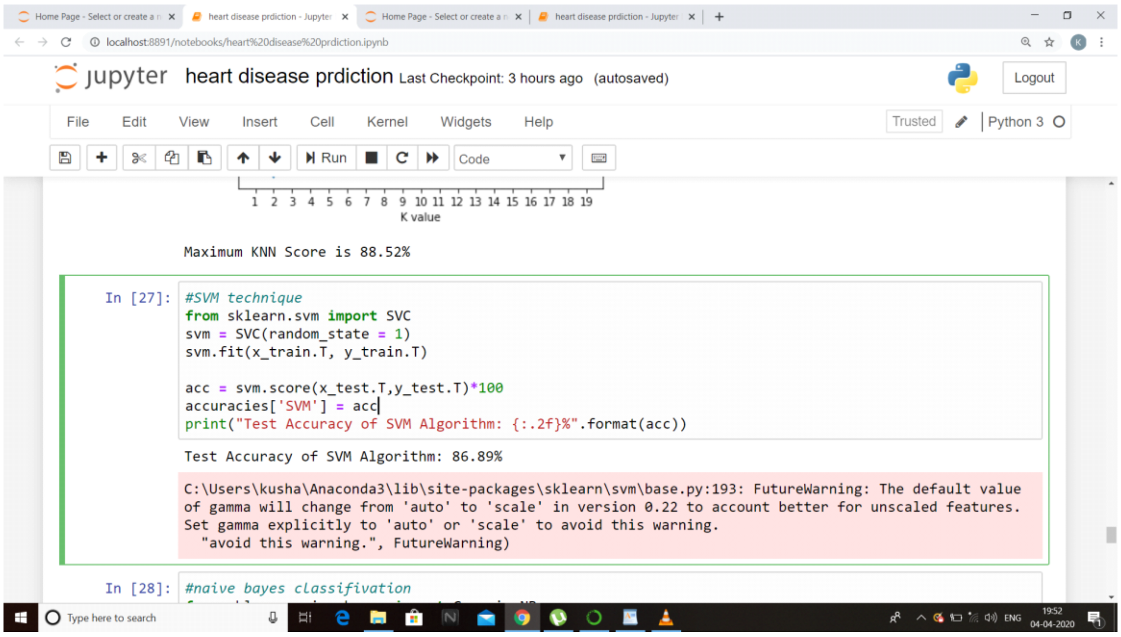
SVC, **NuSVC** and **LinearSVC** are classes capable of performing multi-class classification on a dataset.

SVC and **NuSVC** are similar methods, but accept slightly different sets of parameters and have different mathematical formulations (see section Mathematical formulation). On the other hand, **LinearSVC** is another implementation of Support Vector Classification for the case of a linear kernel. Note that **LinearSVC** does not accept keyword kernel, as this is assumed to be linear. It also lacks some of the members of **SVC** and **NuSVC**, like support_.

As other classifiers, **SVC**, **NuSVC** and **LinearSVC** take as input two arrays: an array X of size [n_samples, n_features] holding the training samples, and an array y of class labels (strings or integers), size [n_samples]:

random_stateint, *RandomState instance or None, optional (default=None)*
The seed of the pseudo random number generator used when shuffling the data for probability estimates. If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random.

The accurate state of the system is as defined by the accuracy of 86.89%

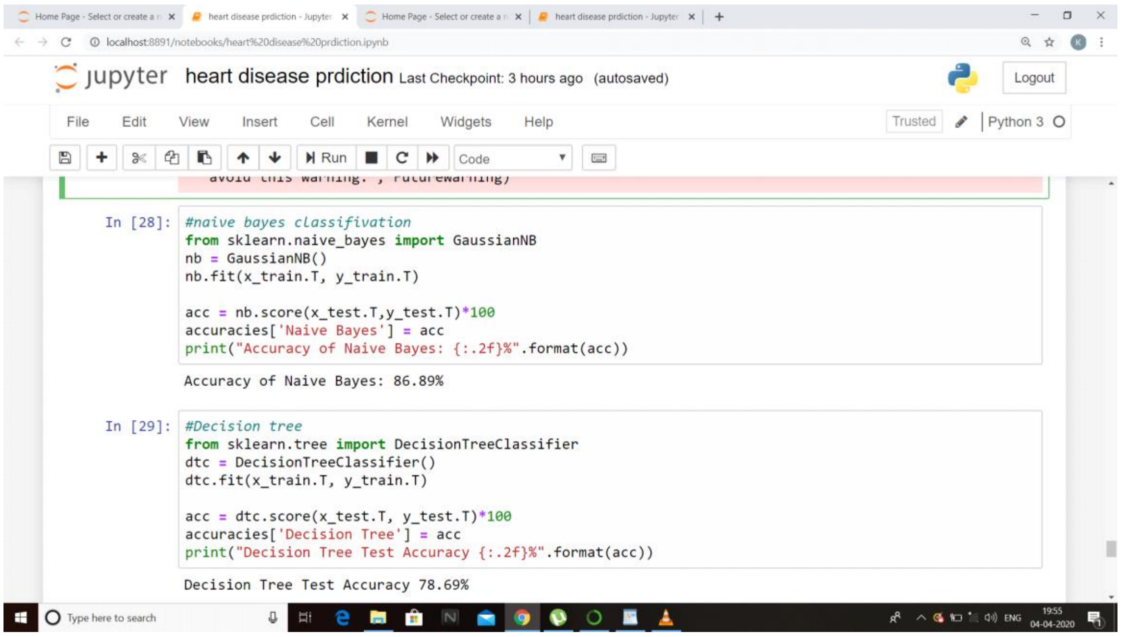

jupyter heart disease prdiction Last Checkpoint: 3 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
K value

```
Maximum KNN Score is 88.52%

In [27]: #SVM technique
         from sklearn.svm import SVC
         svm = SVC(random_state = 1)
         svm.fit(x_train.T, y_train.T)

         acc = svm.score(x_test.T,y_test.T)*100
         accuracies['SVM'] = acc
         print("Test Accuracy of SVM Algorithm: {:.2f}%".format(acc))

         Test Accuracy of SVM Algorithm: 86.89%

         C:\Users\kusha\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default value
         of gamma will change from 'auto' to 'scale' in version 0.22 to account better for unscaled features.
         Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
           "avoid this warning.", FutureWarning)

In [28]: #naive bayes classifivation
```

Generate a model using naive bayes classifier in the following steps:

- Create naive bayes classifier
- Fit the dataset on classifier
- Perform prediction

After model generation, check the accuracy using actual and predicted values.an itsis fund to be 86.89%.


jupyter heart disease prdiction Last Checkpoint: 3 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

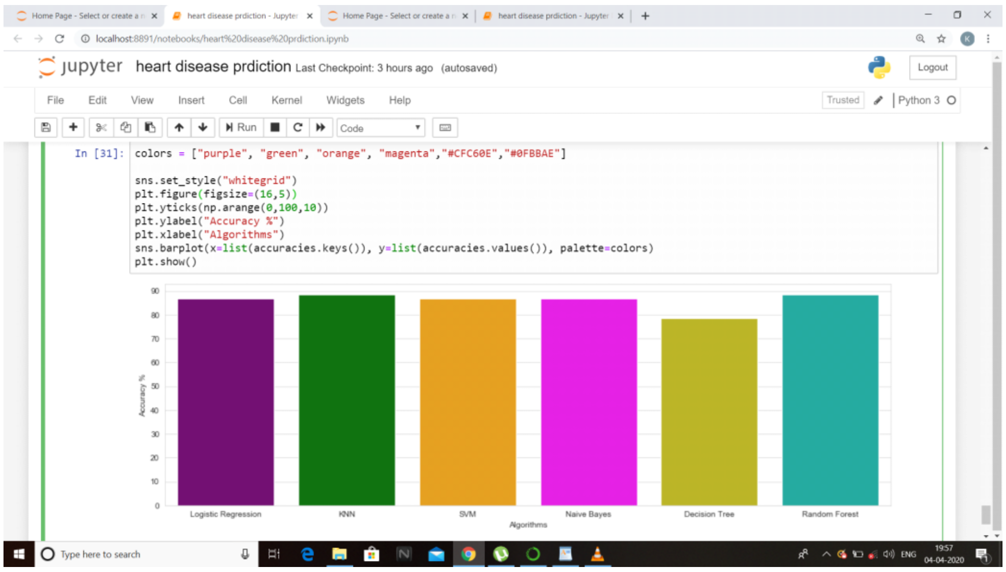
```
In [28]: #naive bayes classifivation
         from sklearn.naive_bayes import GaussianNB
         nb = GaussianNB()
         nb.fit(x_train.T, y_train.T)

         acc = nb.score(x_test.T,y_test.T)*100
         accuracies['Naive Bayes'] = acc
         print("Accuracy of Naive Bayes: {:.2f}%".format(acc))

         Accuracy of Naive Bayes: 86.89%

In [29]: #Decision tree
         from sklearn.tree import DecisionTreeClassifier
         dtc = DecisionTreeClassifier()
         dtc.fit(x_train.T, y_train.T)

         acc = dtc.score(x_test.T, y_test.T)*100
         accuracies['Decision Tree'] = acc
         print("Decision Tree Test Accuracy {:.2f}%".format(acc))

         Decision Tree Test Accuracy 78.69%
```

DecisionTreeClassifier is a class capable of performing multi-class classification on a dataset.As with other classifiers, **DecisionTreeClassifier** takes as input two arrays: an array X, sparse or dense, of

size [n_samples, n_features] holding the training samples, and an array Y of integer values,
size [n_samples],

holding the class labels for the training samples:

- **Snapshot of typical comparisons of efficiency of each algorithm**


heart disease prdiction Last Checkpoint: 3 hours ago (autosaved)
localhost:8891/notebooks/heart%20disease%20prdiction.ipynb
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
90
80
70
60
50
40
30
20
10
0
Accuracy %
Logistic Regression
KNN
SVM
Naive Bayes
Decision Tree
Random Forest
Algorithms
Type here to search
ENG
19:57
04-04-2020

```
In [31]: colors = ["purple", "green", "orange", "magenta","#CFC60E","#0FBBAE"]

         sns.set_style("whitegrid")
         plt.figure(figsize=(16,5))
         plt.yticks(np.arange(0,100,10))
         plt.ylabel("Accuracy %")
         plt.xlabel("Algorithms")
         sns.barplot(x=list(accuracies.keys()), y=list(accuracies.values()), palette=colors)
         plt.show()
```

Conclusion:

The project involved analysis of the heart disease patient dataset with proper data processing. Then, 4 models were trained and tested with maximum scores as follows:

K Neighbours Classifier: 87%

Support Vector Classifier: 83%

Decision Tree Classifier: 79%

Random Forest Classifier: 84%

K Neighbours Classifier scored the best score of 87% with 8 neighbours.

Reference:

[1]C. S. Dangare and S. S. Apte, “Improved study of heart disease prediction system using data mining classification techniques,” International Journal of Computer Applications, vol. 47, no. 10, pp. 44–48, 2012.

[2] S. Palaniappan and R. Awang, “Intelligent heart disease prediction systemusing data mining techniques,” pp. 108–115, 2008.

[3] Y. E. Shao, C.-D. Hou, and C.-C. Chiu, “Hybrid intelligent modelling schemes for heart disease classification,” Applied Soft Computing,vol. 14, pp. 47–52, 2014.

[4] M. Shouman, T. Turner, and R. Stocker, “Using data mining techniquesin heart disease diagnosis and treatment,” pp. 173–177, 2012.;3

[5] P. V. Ankur Makwana, “Identify the patients at high risk of re-admissionin hospital in the next year,” International Journal of Science andResearch, vol. 4, pp. 2431–2434, 2015.

[6] J. Nahar, T. Imam, K. S. Tickle, and Y.-P. P. Chen, “Computationalintelligence for heart disease diagnosis: A medical Knowledge driven approach,” Expert Systems with Applications, vol. 40, no. 1, pp. 96–104,2013.

[7] Y. Xing, J. Wang, Z. Zhao, and Y. Gao, “Combination data miningmethods with new medical data to predicting outcome of coronary heartdisease,” pp. 868–872, 2007.

[8] Combination data mining methods with new medical data to predictingoutcome of coronary heart disease,” in Convergence InformationTechnology, 2007. International Conference on. IEEE, 2007, pp. 868–872.

[9] Y. E. Shao, C.-D. Hou, and C.-C. Chiu, “Hybrid intelligent modelling, schemes for heart disease classification,” Applied Soft Computing,vol. 14, pp. 47–52, 2014.

+github link: https://github.com/kushalsair/intern-pro/blob/master/heart%20disease%20prdiction.ipynb