**IMPLEMENTATION**

**MODULES:**

* Data Collection
* Dataset
* Data Preparation
* Model Selection
* Analyze and Prediction
* Accuracy on test set
* Saving the Trained Model

**MODULES DESCSRIPTION:**

**Data Collection:**

This is the first real step towards the real development of a machine learning model, collecting data. This is a critical step that will cascade in how good the model will be, the more and better data that we get; the better our model will perform.

There are several techniques to collect the data, like web scraping, manual interventions and etc.

The dataset used in this Cleveland Heart Disease dataset taken from the following link:

https://archive.ics.uci.edu/ml/datasets/chronic\_kidney\_disease

**Dataset:**

The dataset consists of 401 individual data. There are 25 columns in the dataset, which are described below.

1. age - age
2. bp - blood pressure
3. sg - specific gravity
4. al - albumin
5. su - sugar
6. rbc - red blood cells
7. pc - pus cell
8. pcc - pus cell clumps
9. ba - bacteria
10. bgr - blood glucose random
11. bu - blood urea
12. sc - serum creatinine
13. sod - sodium
14. pot - potassium
15. hemo - hemoglobin
16. pcv - packed cell volume
17. wc - white blood cell count
18. rc - red blood cell count
19. htn - hypertension
20. dm - diabetes mellitus
21. cad - coronary artery disease
22. appet - appetite
23. pe - pedal edema
24. ane - anemia
25. class - class

**Data Preparation:**

Wrangle data and prepare it for training. Clean that which may require it (remove duplicates, correct errors, deal with missing values, normalization, data type conversions, etc.)

Randomize data, which erases the effects of the particular order in which we collected and/or otherwise prepared our data

Visualize data to help detect relevant relationships between variables or class imbalances (bias alert!), or perform other exploratory analysis

Split into training and evaluation sets

**Model Selection:**

We used support vector machine algorithm, We got a accuracy of 0.9375 on test set so we implemented this algorithm.

Support Vector Machines (SVM) are learning systems that use a hypothesis space of linear functions in a high dimensional feature space, trained with a learning algorithm from optimization theory that implements a learning bias derived from statistical learning theory

Goal of the SVM is to find the optimal hyperplane that divides the two classes. There can be different planes that can divide the two classes, but the main focus is on to finding out such plane that we can achieve maximum margin between the classes. It means pick the hyperplane so that the distance from the hyperplane to the nearest data point is maximized

## How does it work?

Above, we got accustomed to the process of segregating the two classes with a hyper-plane. Now the burning question is “How can we identify the right hyper-plane?”. Don’t worry, it’s not as hard as you think!

Let’s understand:

**Identify the right hyper-plane (Scenario-1):**Here, we have three hyper-planes (A, B, and C). Now, identify the right hyper-plane to classify stars and circles.  


You need to remember a thumb rule to identify the right hyper-plane: “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.

**Identify the right hyper-plane (Scenario-2):**Here, we have three hyper-planes (A, B, and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?



Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**. Let’s look at the below snapshot:[[](https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_4.png)](https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_4.png)

Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.

**Identify the right hyper-plane (Scenario-3):**Hint:Use the rules as discussed in previous section to identify the right hyper-plane

**[](https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_5.png)**

Some of you may have selected the hyper-plane **B**as it has higher margin compared to **A.**But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is **A.**

**Can we classify two classes (Scenario-4)?:**Below, I am unable to segregate the two classes using a straight line, as one of the stars lies in the territory of other(circle) class as an outlier.  **[](https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_61.png)**

As I have already mentioned, one star at other end is like an outlier for star class. The SVM algorithm has a feature to ignore outliers and find the hyper-plane that has the maximum margin. Hence, we can say, SVM classification is robust to outliers.  
**[](https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_71.png)**

**Find the hyper-plane to segregate to classes (Scenario-5):**In the scenario below, we can’t have linear hyper-plane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyper-plane.

**[](https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_8.png)**

SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature z=x^2+y^2. Now, let’s plot the data points on axis x and z:  
[[](https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_9.png)](https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_9.png)In above plot, points to consider are:

All values for z would be positive always because z is the squared sum of both x and y

In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.

In the SVM classifier, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, the SVM  algorithm has a technique called the **kernal trick**. The SVM kernel is a function that takes low dimensional input space and transforms it to a higher dimensional space i.e. it converts not separable problem to separable problem. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then finds out the process to separate the data based on the labels or outputs you’ve defined.

When we look at the hyper-plane in original input space it looks like a circle:  
[](https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_10.png)

**Analyze and Prediction:**

In the actual dataset, we chose only 18 features :

1. age - age
2. bp - blood pressure
3. al - albumin
4. su - sugar
5. rbc - red blood cells
6. pc - pus cell
7. pcc - pus cell clumps
8. ba - bacteria
9. bgr - blood glucose random
10. bu - blood urea
11. sc - serum creatinine
12. pot - potassium
13. wc - white blood cell count
14. htn - hypertension
15. dm - diabetes mellitus
16. cad - coronary artery disease
17. pe - pedal edema
18. ane - anemia
19. class – class(Label)

**Accuracy on test set:**

We got a accuracy of 0.9375% on test set.

**Saving the Trained Model:**

Once you’re confident enough to take your trained and tested model into the production-ready environment, the first step is to save it into a .h5 or .pkl file using a library like pickle.

Make sure you have pickle installed in your environment.

Next, let’s import the module and dump the model into .pkl file.