“Vaticination - Heart Disease Prediction Using Machine Learning Technique”

Webhost: www.vaticination.ga:8888

A

Project Report

submitted in partial fulfillment of the

requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

by

	Name:	Kanwaljit Singh	Mayank Joshi	MD. Raise Azam
	SAP ID:	500044606	500045822	500045437
	Roll No:	R110215062	R110215073	R110215074
	Branch:	CCVT	CCVT	CCVT

Under the guidance of

Mr. Alind

Assistant Professor (SS),

School of Computer Science Engineering.

Department of Virtualization

School of Computer Science Engineering,

UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

Dehradun-248007

August-Dec’2018

CANDIDATE'S DECLARATION

I/We hereby certify that the project work entitled **"Vaticination - Heart Disease Prediction Using Machine Learning Technique"** in partial fulfilment of the requirements for the award of the Degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING with specialization in Cloud Computing and Virtualization Technology) and submitted to the Department of Computer Science & Engineering at Center for Information Technology, University of Petroleum & Energy Studies, Dehradun, is an authentic record of my/ our work carried out during a period from **August**, **2018** to **December**, **2018** under the supervision of **Mr. Alind, Assistant Professor(SS), SoCS**.

The matter presented in this project has not been submitted by me/ us for the award of any other degree of this or any other University.

Kanwaljit Singh (62)
Mayank Joshi (73)
MD. Raise Azam (74)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date: 13th December 2018

(**Mr. Alind)**
Project Guide

Dr. Amit Aggarwal
Head of the Department – CCVT (Cloud Computing and Virtualization Technology)
School of Computer Science Engineering,
University of Petroleum & Energy Studies
Dehradun – 248 007 (Uttarakhand)

ACKNOWLEDGEMENT

We wish to express our deep gratitude to our guide **Mr. Alind**, for all advice, encouragement and constant support he has given us throughout our project work. This work would not have been possible without his support and valuable suggestions.

We sincerely thank to our respected Program Head of the Department, **Dr. Amit Aggarwal**, for his great support in doing our project at **SoCS**.

We are also grateful to **Dr. Manish Prateek**, **Director** (SoCS), **UPES** for giving us the necessary facilities to carry out our project work successfully.

We would like to thank all our **friends** for their help and constructive criticism during our project work. Finally, we have no words to express our sincere gratitude to our **parents** who have shown us this world and for every support they have given us.

Name	**Kanwaljit Singh**	**Mayank Joshi**	**MD. Raise Azam**
Roll No.	**R-110215062**	**R-110215073**	**R-110215074**

Abstract:

Today, it is frequently seen that many people around us suffer from different diseases, some of them are curable but some of them are not. Some require quick attention but some can be taken in period of time. Heart diseases are the one which require proper attention and need quick treatment. There are systems that can identify the type of disease a person is suffering but still the systems are not precise and the level of correctness is also not met. So, there is a need of reliable, accurate and feasible system to diagnose such diseases in a time.

In our major we are focusing on this issue and are using different types of machine learning algorithm to train and test data-sets using labeled dataset of heart disease and try to find the classifier which predict that result with higher accuracy on out-of-sample data. At the end, we will also perform feature selection on our model to improve the level of accuracy. In client view, we are accepting the user inputs dynamically through suitable executable hosted on our cloud server according to the prerequisites required by the end-client system and catching them into remotely-empowered database and handling that client inputs utilizing trained data-sets for our final prediction and showing them on our centralized web-based interface.

Key words: vaticination – act of prediction, machine learning, out-of-sample data, accuracy, classifiers, heart disease, linear svm, decision tree, naïve bayes, jupyter-server, google-cloud.

TABLE OF CONTENTS

LIST OF FIGURES

S.No. FIGURES Page No

1. Introduction:

Heart disease has created a lot of serious concerned among researches; one of the major challenges in heart disease is correct detection and finding presence of it inside a human. Early techniques have not been so much efficient in finding it even medical professor are not so much efficient enough in predicating the heart disease [3]. There are various medical instruments available in the market for predicting heart disease there are two major problems in them, the first one is that they are very much expensive and second one is that they are not efficiently able to calculate the chance of heart disease in human. According to latest survey conducted by WHO, the medical professional able to correctly predicted only 67% of heart disease [2]so there is a vast scope of research in area of predicating heart disease in human. With advancement in computer science has brought vast opportunities in different areas, medical science is one of the fields where the instrument of computer science can be used. Medical science also used some of the major available tools in computer science; in last decade artificial intelligence has gained its moment because of advancement in computation power.

Machine Learning is one such tool which is widely utilized in different domains because it doesn't require different algorithm for different dataset. Reprogrammable capacities of machine learning bring a lot of strength and opens new doors of opportunities for area like medical science. In medical science heart disease is one of the major challenges; because a lot of parameters and technicality is involve for accurately predicating this disease. Machine learning could be a better choice for achieving high accuracy for predicating not only heart disease but also another diseases because this vary tool utilizes feature vector and its various data types under various condition for predicating the heart disease, algorithms such as Naive Bayes, Decision Tree, KNN, SVM, are used to predicate risk of heart diseases each algorithm has its speciality such as Naive Bayes used probability for predicating heart disease, whereas decision tree is used to provide classified report for the heart disease. All these techniques are using old patient record for getting predication about new patient. This predication system for heart disease helps doctors to predict heart disease in the early stage of disease resulting in saving millions of lives.

2. Problem Statement:

Clinical decisions are often made based on doctors' intuition and experience rather than on the knowledge rich data hidden in the database. This practice leads to unwanted biases, errors and excessive medical costs which affects the quality of service provided to patients.
There are many ways that a medical misdiagnosis can occur. Whether a doctor is at fault, or hospital staff, a misdiagnosis of a serious illness can have very extreme and harmful effects. The National Patient Safety Foundation cites that 42% of medical patients feel they have had experienced a medical error or missed diagnosis.

Thus, we proposed a system which would predict the heart disease objectively with the help of previously recorded healthcare data. Our system could potentially reduce medical errors, enhance patient safety and decrease malpractice from the doctors or healthcare department. We're comparing and analyzing the predicted result and their accuracy for heart disease using Supervised Machine Learning Techniques.

3. Literature Review:

Predication of heart disease based on machine learning algorithm is always curious case for researchers recently there is a wave of papers and research material on this area. Marjia Sultana, Afrin Haider and Mohammad ShorifUddin[4] have illustrated about how the datasets available for heart disease are generally a raw in nature which is highly redundant and inconsistent. There is a need of pre-processing of these data sets; in this phase high dimensional data set is reduced to low data set. They also show that extraction of crucial features from the data set because there is every kind of features. Selection of important features reduces work of training the algorithm and hence resulted in reduction in time complexity. Time is not only single parameter for comparison other parameters like accuracy also play vital role in proving effectiveness of algorithm similar.

An approach proposed in [4] have worked to improve the accuracy and found that performance of Bayes Net and SMO classifiers are much optimal than MLP, J48 and KStar. Performance is measured by running algorithms (Bayes Net and SMO) on data set collected from WEKA software and then compared using predictive accuracy, ROC curve, ROC value. Different methods have their own merits and demerits in work done by M.A. Jabbar, B.L Deekshatulu, Priti Chndra [5], an optimization of feature has been done to achieve higher classification efficiency in Decision Tree. It is an approach for early detection of heart disease by utilizing variety of feature. These kinds of approach can also be utilize for other sphere of research. Other than decision tree various other approach where adopt for achieving the goal of perfect detection of heart disease in human Yogeswaran Mohan et.al [6] have collected raw data form EEG device and used to train neural network for pattern classification. Here input output are depressive and non-depressive categories in the hidden layer scaled conjugate gradient algorithm is used for training to achieve efficient result. authors have got efficiency up to 95% with help of trained neural network watching the success of neural network researches working in the domain of SVM have used this technique to classify and achieve more better result in case where the feature vector are multi-dimensional and non-linear these methods defeated all other existing quantum contemporary techniques because it has capability to work under dataset of high dimensionality.

After going through majority of state of art technique we have pointed out certain loop holes existed in them. Some of them are discussed below

- There is wide need for more robust algorithm which can minimized the noise in the dataset because medical dataset may consists of various types of redundancy and noise in them.
- Recently with advancement in deep learning there could be chance to enhance efficiency and accuracy for detection heart disease.

4. Objectives Achieved:

1) We found training accuracy for different classifiers.
2) We found testing accuracy for our classifiers as well.
3) We did cross validation to find the best value of 'K' in cross validation for different classifier.
4) We have performed feature selection over the given data set.
5) We have developed a client-side executable file that will sign up user, provide login and input symptoms via remote SQL server.
6) We have trained our model over the google cloud and predicting its output via dynamic user inputs.
7) We are performing all our computation on Jupiter server which is set up on cloud and it's ipv4 address we have mapped to our domain.
8) We have developed and deployed the GUI using lamp stack for showing each user profile with its symptoms and ease for admin management.

5. Methodology:

The model we took for our software project depends on our whole analysis i.e. Evolutionary Prototype Software Development Process.

Step1: We have to select a labeled Dataset that would help in prediction of heart diseases.

Step2: We will be using few supervised learning models, that are Support Vector Machine, Decision Tree, Logistic Regression, Naïve Bayes, and KNN and train them using the Data set.

Step3: We will follow three different approach for training and testing our models.

Step4: In first approach, we will find Training Accuracy. In this we will be using the whole data set to train as well as test the model and after testing will find the accuracy of all the models.

Step5: In second approach, we will find Testing Accuracy. In this we will be using some part of the Data set for training the models and the other left part of the dataset for the testing purpose. Similar to step 4, we need to find accuracy of all the models.

Step6: In third approach, we will perform K-Fold cross Validation. In this we will be using different fractions of data set to train and test our models and find the accuracy of all the models.

Step7: Now we will compare the accuracy among all the models in three approaches and find best among all. Then we will select that model or classifier which will give better estimate on Out-of-sample data.

Step8: Finally using this model we will try to perform feature selection and will try to improve the accuracy of our system.

5.1. Naïve Bayes: -

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes' Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

A naive Bayes classifier uses probability theory to classify data. Naive Bayes classifier algorithms make use of Bayes' theorem. The key insight of Bayes' theorem is that the probability of an event can be adjusted as new data is introduced.

What makes a naive Bayes classifier naive is its assumption that all attributes of a data point under consideration are independent of each other. A classifier sorting fruits into apples and oranges would know that apples are red, round and are a certain size, but would not assume all these things at once. Oranges are round too, after all.

A naive Bayes classifier is not a single algorithm, but a family of machine learning algorithms that make uses of statistical independence. These algorithms are relatively easy to write and run more efficiently than more complex Bayes algorithms.

The most popular application is spam filters. A spam filter looks at email messages for certain key words and puts them in a spam folder if they match.

Despite the name, the more data it gets, the more accurate a naive Bayes classifier becomes, such as from a user flagging email messages in an inbox for spam

5.2. Decision Tree: -

A decision tree is a graphical representation of specific decision situations that are used when complex branching occurs in a structured decision process. A decision tree is a predictive model based on a branching series of Boolean tests that use specific facts to make more generalized conclusions.

The main components of a decision tree involve decision points represented by nodes, actions and specific choices from a decision point. Each rule within a decision tree is represented by tracing a series of paths from root to node to the next node and so on until an action is reached.

Decision trees are a popular and powerful tool used for classification and prediction pu rposes. Decision trees provide a convenient alternative for viewing and managing large sets of business rules, allowing them be translated in a way that allows humans to understand them and apply the rules constraints in a database so that records falling into a specific category are sure to be retrieved.

Decision trees generally consist of the following four steps:

1. Structuring the problem as a tree by creating end nodes of the branches, which are associated with a specific path or scenario along the tree
2. Assigning subject probabilities to each represented event on the tree
3. Assigning payoffs for consequences. This could be a specific dollar amount or utility value that is associated with a particular scenario.
4. Identifying and selecting the appropriate course(s) of action based on analyses

5.3. Logistic Regression

Logistic regression is a kind of statistical analysis that is used to predict the outcome of a dependent variable based on prior observations. For example, an algorithm could determine the winner of a presidential election based on past election results and economic data. Logistic regression algorithms are popular in machine learning.

Logistic regression is a technique in statistical analysis that attempts to predict a data value based on prior observations. A logistic regression algorithm looks at the relationship between a dependent variable and one or more dependent variables.

Logistic regression has a number of applications in machine learning. A logistic regression algorithm might attempt to predict which candidate would win in an election by averaging all the polling results. A more sophisticated algorithm might also incorporate economic data and past elections in its model. Another algorithm might try to identify which users of a website would click on certain ads. It is also commonly used in database preparation to classify data for extract, transform and load (ETL) operations.

5.4. KNN

A k-nearest-neighbor algorithm, often abbreviated k-nn, is an approach to data classification that estimates how likely a data point is to be a member of one group or the other depending on what group the data points nearest to it are in.

The k-nearest-neighbor is an example of a "lazy learner" algorithm, meaning that it does not build a model using the training set until a query of the data set is performed.

A k-nearest-neighbor is a data classification algorithm that attempts to determine what group a data point is in by looking at the data points around it.

An algorithm, looking at one point on a grid, trying to determine if a point is in group A or B, looks at the states of the points that are near it. The range is arbitrarily determined, but the point is to take a sample of the data. If the majority of the points are in group A, then it is likely that the data point in question will be A rather than B, and vice versa.

The k-nearest-neighbor is an example of a "lazy learner" algorithm because it does not generate a model of the data set beforehand. The only calculations it makes are when it is asked to poll the data point's neighbors. This makes k-nn very easy to implement for data mining.

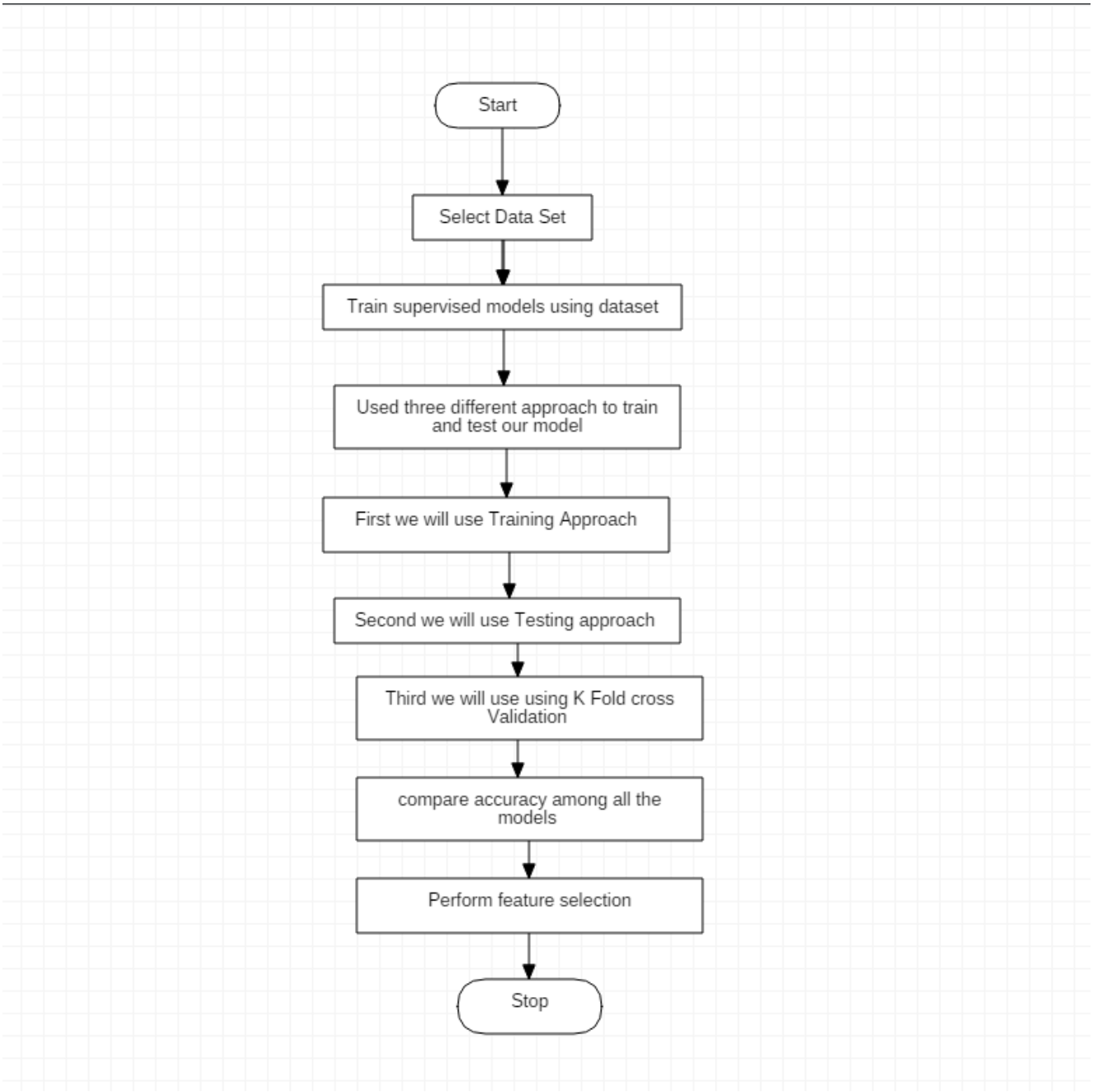
5.5. SVM

Support Vector Machines is another popular classification technique. A support vector machine constructs a hyperplane or set of hyperplanes in a high-dimensional space such that the separation is maximum. This is the reason the SVM is also called the maximum margin classifier. The hyperplane identifies certain examples close to the plane which are called as support vectors. SVM is a classifier derived from statistical learning theory by Vapnik and Chervonenkis.

It is supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. SVMs are learning systems that use a hypothesis space of linear functions in a high dimensional feature space — Kernel function. It is trained with a learning algorithm from optimization theory called Lagrange.

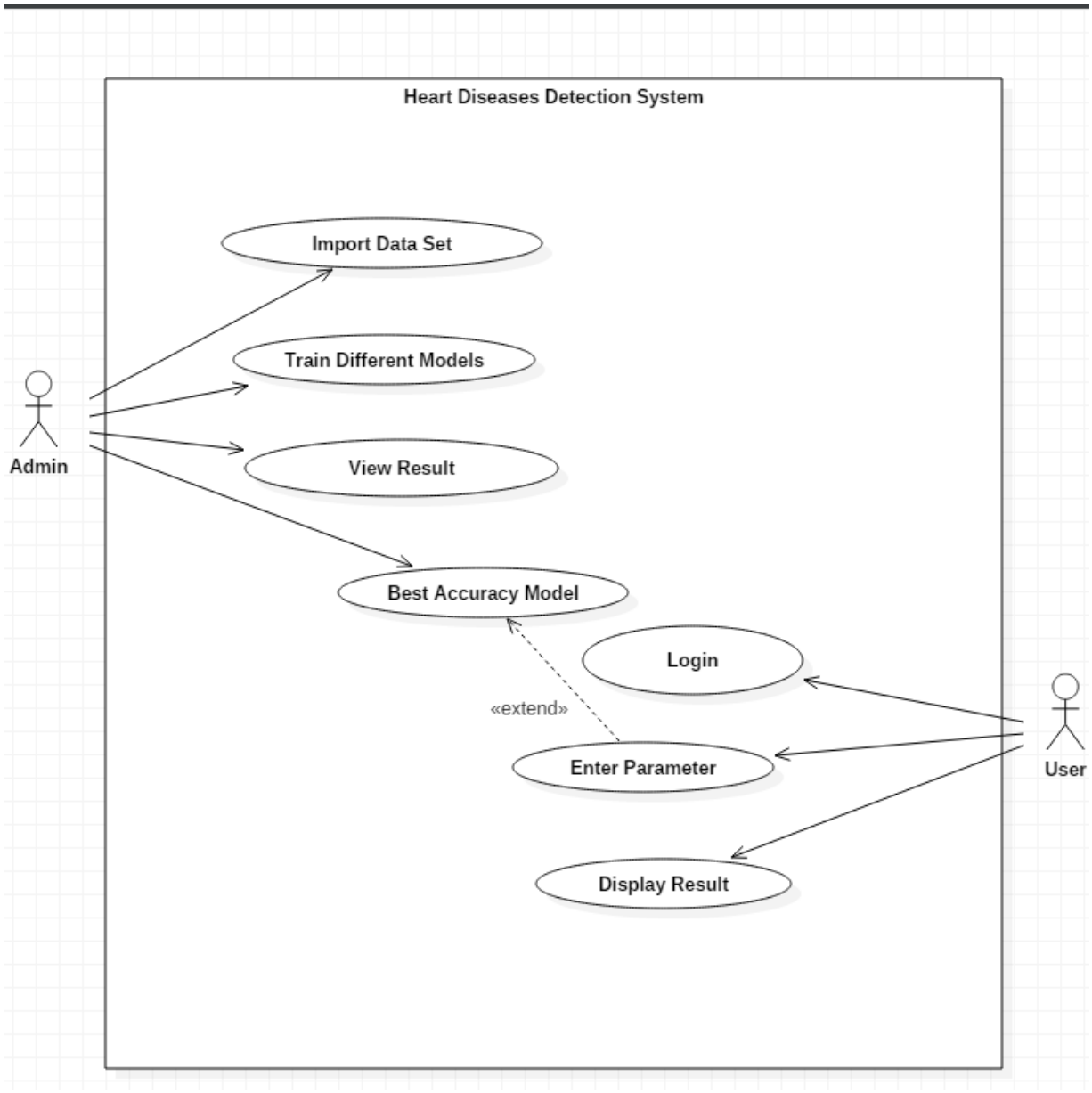
The basic SVM takes a set of input data and predicts, for each given input, which of two possible classes forms the output, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall. SVMs are helpful in text and hypertext categorization. Classification of images and hand-written characters can also be recognized using SVM.

6. Flow Chart:



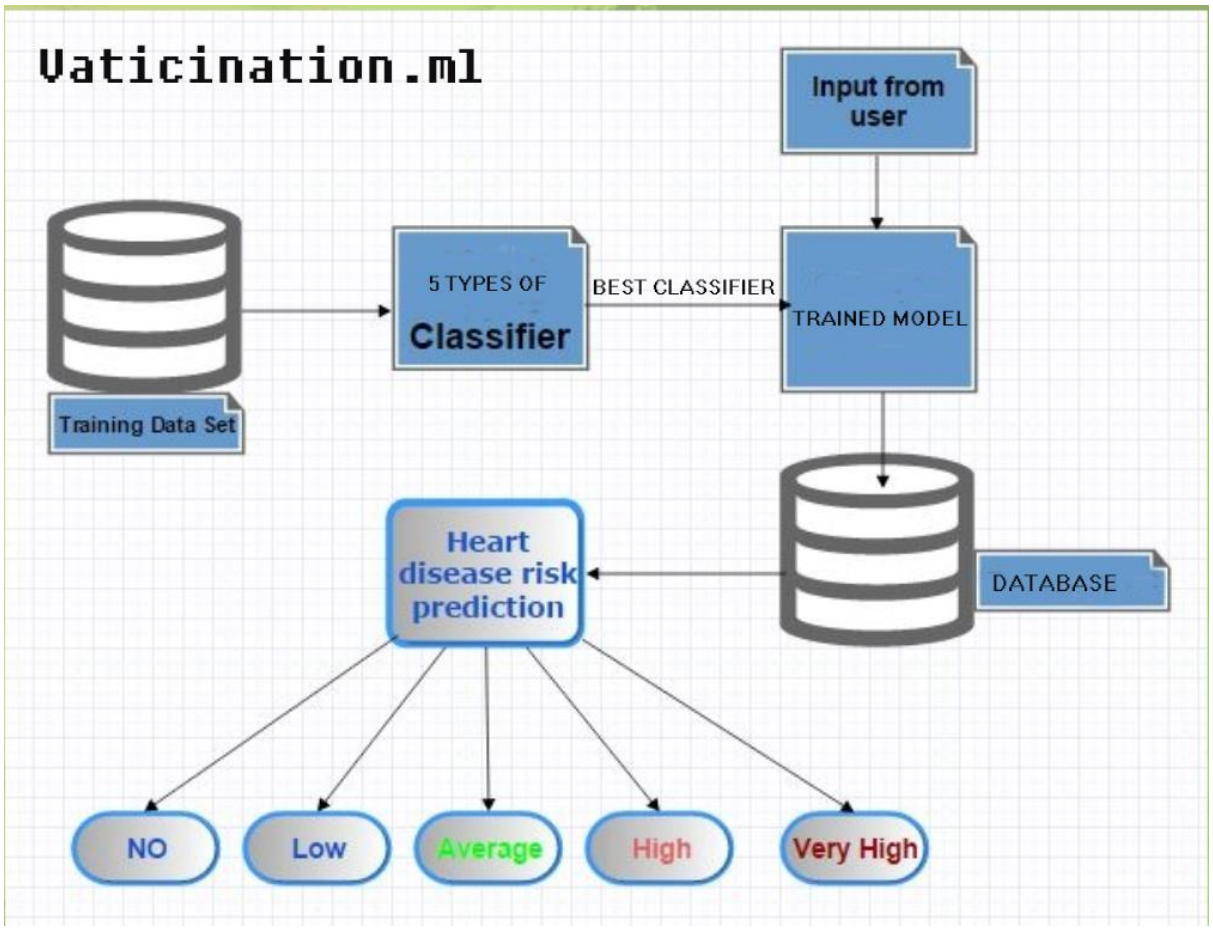
(Figure 6 – Mechanism of Selecting best Model Using Various Classifier)

7. Use Case Diagram:



(Figure 7 – Use Case Diagram of the System)

8. Architecture:



(Figure 8 – Architecture of the system)

9. Source Code (Snippet):

```
# coding: utf-8

# In[1]:


import pandas as pd
import numpy as np
columns = ["age", "sex", "cpp", "restbp", "chol", "fbs", "restecg","thalach", "exang", "oldpeak", "slope", "ca", "thal", "num"]
df= pd.read_csv("D:\\StudyStuff\\Major\\processed_cleveland_data1.csv", header=None,names=columns)
df.head()


# In[2]:


y=df['num']


# In[3]:


X=df[ ["age", "sex", "cpp", "restbp", "chol", "fbs", "restecg","thalach", "exang", "oldpeak", "slope", "ca", "thal"]]


# In[4]:


from sklearn.linear_model import LogisticRegression

# instantiate the model (using the default parameters)
logreg = LogisticRegression()

# fit the model with data
logreg.fit(X, y)

# predict the response values for the observations in X
logreg.predict(X)


# In[5]:


y_pred = logreg.predict(X)
# compute classification accuracy for the logistic regression model
#Training Accuracy for logistic Regression
from sklearn import metrics
print(metrics.accuracy_score(y, y_pred))


# In[6]:


from sklearn.neighbors import KNeighborsClassifier
# try K=1 through K=25 and record training accuracy
k_range = list(range(1, 26))
scores = []
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X,y)
    y_pred = knn.predict(X)
    scores.append(metrics.accuracy_score(y, y_pred))
    # import Matplotlib (scientific plotting library)
```

```
import matplotlib.pyplot as plt

# allow plots to appear within the notebook
get_ipython().run_line_magic('matplotlib', 'inline')

# plot the relationship between K and testing accuracy
plt.plot(k_range, scores)
plt.xlabel('Value of K for KNN')
plt.ylabel('Training Accuracy')


# In[7]:


# Select k=1, for training Accuracy
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X,y)
y_pred = knn.predict(X)
print(metrics.accuracy_score(y, y_pred))


# In[8]:


#Training Accuracy on Naive Bayes
from sklearn.naive_bayes import GaussianNB
nb= GaussianNB()
nb.fit(X,y)
y_pred = nb.predict(X)
print(metrics.accuracy_score(y, y_pred))


# In[9]:


#Training Accuracy on Decision Trees
        from sklearn import tree
        dt=tree.DecisionTreeClassifier()
        dt.fit(X,y)
        y_pred = dt.predict(X)
        print(metrics.accuracy_score(y, y_pred))


        # In[10]:


        #Training Accuracy on SVM
        from sklearn.svm import SVC
        svm=SVC()
        svm.fit(X,y)
        y_pred = svm.predict(X)
        print(metrics.accuracy_score(y, y_pred))


        # In[11]:


        #Now we will check Testing Accuracy as training accuracy overfits the data

        # STEP 1: split X and y into training and testing sets
        from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ran
    dom_state=4)

        # STEP 2: train the model on the training set
        # First we will use Logistic Regression
        logreg = LogisticRegression()
        logreg.fit(X_train, y_train)
```

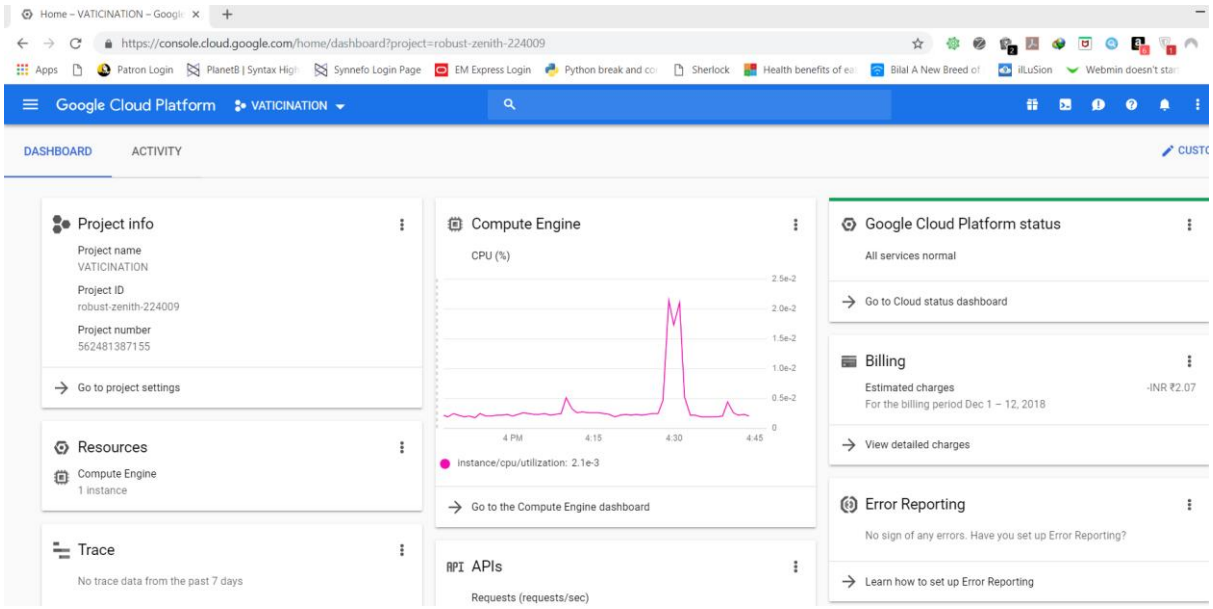
```
130.
131.        # STEP 3: make predictions on the testing set
132.        y_pred = logreg.predict(X_test)
133.
134.        # compare actual response values (y_test) with predicted response values (y_
    pred)
135.        print(metrics.accuracy_score(y_test, y_pred))
136.
137.
138.        # In[12]:
139.
140.
141.        from sklearn.neighbors import KNeighborsClassifier
142.        # try K=1 through K=30 and record training accuracy
143.        k_range = list(range(1, 31))
144.        scores = []
145.        for k in k_range:
146.            knn = KNeighborsClassifier(n_neighbors=k)
147.            knn.fit(X_train,y_train)
148.            y_pred = knn.predict(X_test)
149.            scores.append(metrics.accuracy_score(y_test, y_pred))
150.            # import Matplotlib (scientific plotting library)
151.        import matplotlib.pyplot as plt
152.
153.        # allow plots to appear within the notebook
154.        get_ipython().run_line_magic('matplotlib', 'inline')
155.
156.        # plot the relationship between K and testing accuracy
157.        plt.plot(k_range, scores)
158.        plt.xlabel('Value of K for KNN')
159.        plt.ylabel('Testing Accuracy')
160.
161.
162.
163.        # In[13]:
164.
165.
166.        # Select k=15, for testing Accuracy using the plot above
167.        knn = KNeighborsClassifier(n_neighbors=15)
168.        knn.fit(X_train,y_train)
169.        y_pred = knn.predict(X_test)
170.        print(metrics.accuracy_score(y_test, y_pred))
171.
172.
173.        # In[14]:
174.
175.
176.        #Testing Accuracy on Naive Bayes
177.        from sklearn.naive_bayes import GaussianNB
178.        nb= GaussianNB()
179.        nb.fit(X_train,y_train)
180.        y_pred = nb.predict(X_test)
181.        print(metrics.accuracy_score(y_test, y_pred))
182.
183.
184.        # In[15]:
185.
186.
187.        #Testing Accuracy on Decision Trees
188.        from sklearn import tree
189.        dt=tree.DecisionTreeClassifier()
190.        dt.fit(X_train,y_train)
191.        y_pred = dt.predict(X_test)
192.        print(metrics.accuracy_score(y_test, y_pred))
193.
194.
195.        # In[16]:
196.
197.
```

```
        #Testing Accuracy on SVM
        from sklearn.svm import SVC
        svm=SVC()
        svm.fit(X_train,y_train)
        y_pred = svm.predict(X_test)
        print(metrics.accuracy_score(y_test, y_pred))


        # In[17]:


        # k-fold cross-validation to improve testing accuracy

        from sklearn.model_selection import cross_val_score


        cross_range = list(range(2, 31))
        scores = []
        for c in cross_range:
            scores.append(cross_val_score(logreg, X_train, y_train, cv=c, scoring='a
    ccuracy').mean())


        import matplotlib.pyplot as plt

        # allow plots to appear within the notebook
        get_ipython().run_line_magic('matplotlib', 'inline')

        # plot the relationship between K and testing accuracy
        plt.plot(cross_range, scores)
        plt.xlabel('Value of cv for Cross Validation in Logistic Regression')
        plt.ylabel('Testing Accuracy')


        # In[18]:


        #Best value of k-Fold cross validation is at cv=29 for logistic Regression
        cross_val_score(logreg, X_train, y_train, cv=29, scoring='accuracy').mean()


        # In[19]:


        cross_range = list(range(2, 31))
        scores = []
        for c in cross_range:
            scores.append(cross_val_score(knn, X_train, y_train, cv=c, scoring='accu
    racy').mean())
        import matplotlib.pyplot as plt

        # allow plots to appear within the notebook
        get_ipython().run_line_magic('matplotlib', 'inline')

        # plot the relationship between K and testing accuracy
        plt.plot(cross_range, scores)
        plt.xlabel('Value of cv for Cross Validation in KNN')
        plt.ylabel('Testing Accuracy')


        # In[20]:


        #Best value of k-Fold cross validation is at cv=22 for KNN
        cross_val_score(knn, X_train, y_train, cv=22, scoring='accuracy').mean()
```

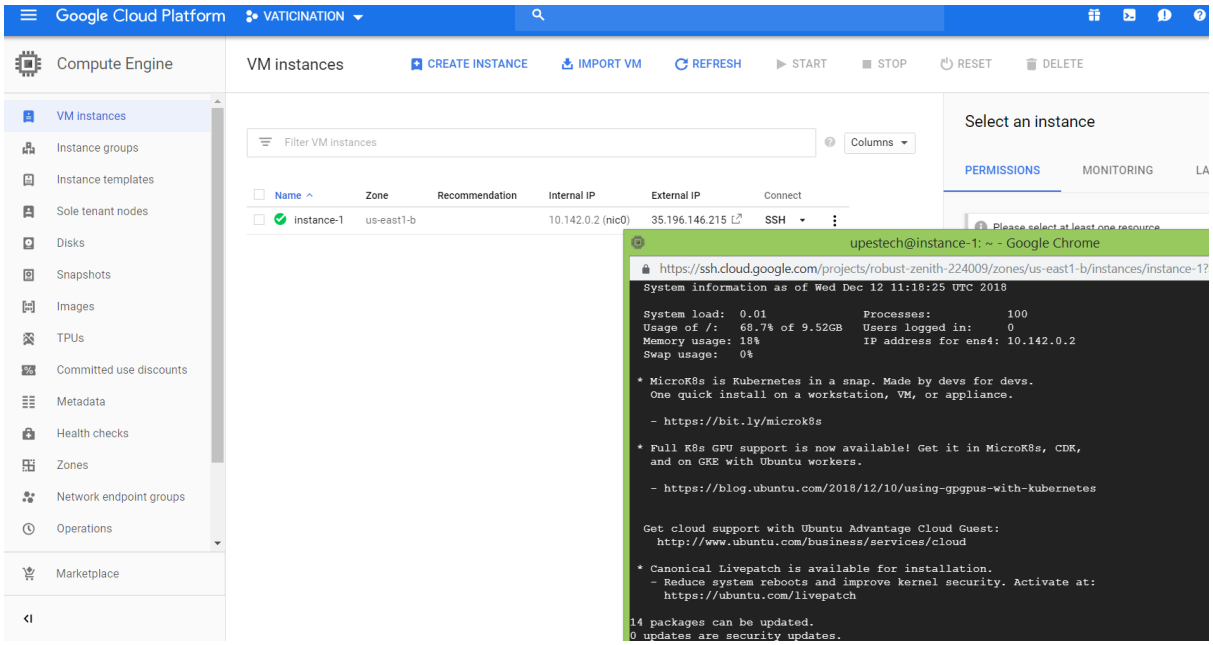
```
264.
265.
266.        # In[21]:
267.
268.
269.        cross_range = list(range(2, 31))
270.        scores = []
271.        for c in cross_range:
272.            scores.append(cross_val_score(nb, X_train, y_train, cv=c, scoring='accur
    acy').mean())
273.        import matplotlib.pyplot as plt
274.
275.        # allow plots to appear within the notebook
276.        get_ipython().run_line_magic('matplotlib', 'inline')
277.
278.        # plot the relationship between K and testing accuracy
279.        plt.plot(cross_range, scores)
280.        plt.xlabel('Value of cv for Cross Validation in Naive Bayes')
281.        plt.ylabel('Testing Accuracy')
282.
283.
284.
285.        # In[22]:
286.
287.
288.        #Best value of k-Fold cross validation is at cv=20 for Naive Bayes
289.        cross_val_score(nb, X_train, y_train, cv=20, scoring='accuracy').mean()
290.
291.
292.        # In[23]:
293.
294.
295.        cross_range = list(range(2, 31))
296.        scores = []
297.        for c in cross_range:
298.            scores.append(cross_val_score(dt, X_train, y_train, cv=c, scoring='accur
    acy').mean())
299.        import matplotlib.pyplot as plt
300.
301.        # allow plots to appear within the notebook
302.        get_ipython().run_line_magic('matplotlib', 'inline')
303.
304.        # plot the relationship between K and testing accuracy
305.        plt.plot(cross_range, scores)
306.        plt.xlabel('Value of cv for Cross Validation in Decision Trees')
307.        plt.ylabel('Testing Accuracy')
308.
309.
310.
311.        # In[24]:
312.
313.
314.        #Best value of k-Fold cross validation is at cv=15 for Decision Trees
315.        cross_val_score(dt, X_train, y_train, cv=15, scoring='accuracy').mean()
316.
317.
318.        # In[25]:
319.
320.
321.        cross_range = list(range(2, 31))
322.        scores = []
323.        for c in cross_range:
324.            scores.append(cross_val_score(svm, X_train, y_train, cv=c, scoring='accu
    racy').mean())
325.        import matplotlib.pyplot as plt
326.
327.        # allow plots to appear within the notebook
328.        get_ipython().run_line_magic('matplotlib', 'inline')
329.
```

```
330.        # plot the relationship between K and testing accuracy
331.        plt.plot(cross_range, scores)
332.        plt.xlabel('Value of cv for Cross Validation in SVM')
333.        plt.ylabel('Testing Accuracy')
334.
335.
336.
337.        # In[26]:
338.
339.
340.        #High value of cv in cross validation will lead to high variance and thus Ov
    erfitting problem
341.        #Best value of k-Fold cross validation is at cv=118 for SVM
342.        cross_val_score(svm, X_train, y_train, cv=20, scoring='accuracy').mean()
343.
344.
345.        # In[27]:
346.
347.
348.        #Finally we are selecting Naive Bayes as it is having higher accuracy after
    taking into consideration the problem of Over-fitting
349.        from sklearn.naive_bayes import GaussianNB
350.        nb= GaussianNB()
351.        nb.fit(X_train,y_train)
352.        #Taking user define input to test the model
353.        X_new= pd.read_csv("http://vaticination.ga/dynamic_data.php")
354.        y_pred = nb.predict(X_new)
355.        print(' '.join(map(str,y_pred)))
356.
357.        Refer Full Source Code: - http://vaticination.ga/source_code.py
```

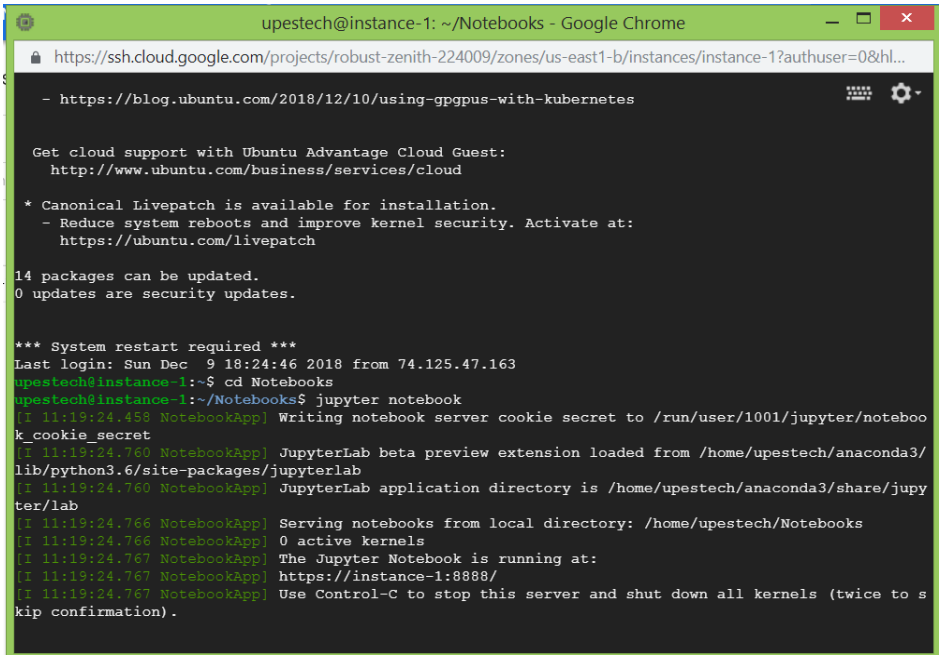
10. Program Output:



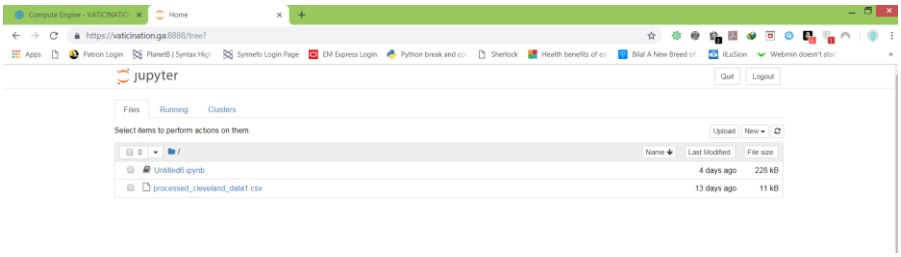
(Figure 10.1 – Google Cloud Instance Performance Metrics for Jupyter Server)



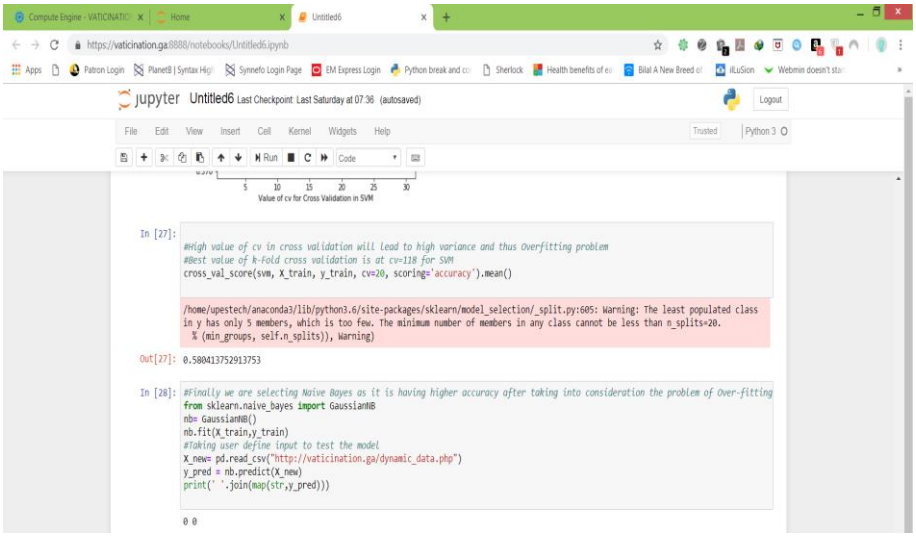
(Figure 10.2 – Google Cloud Instance's SSH for Jupyter Server)



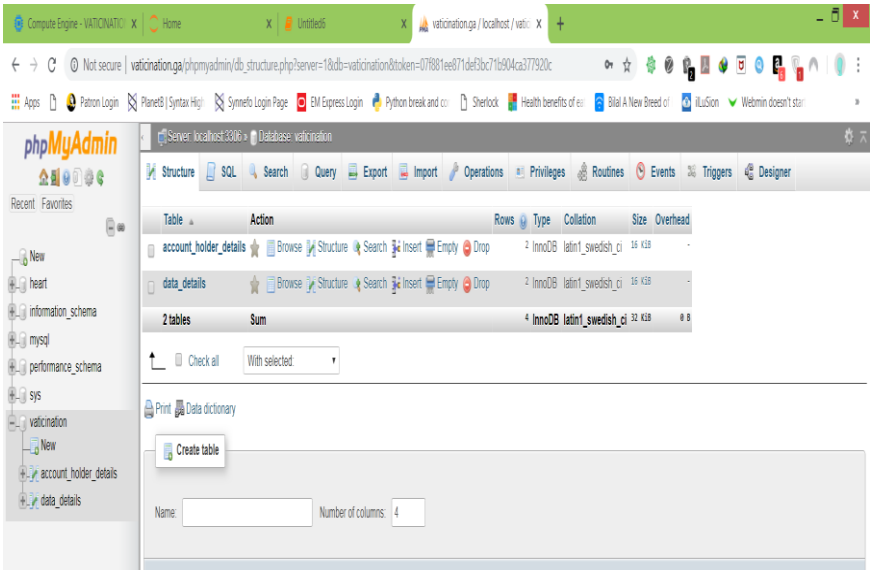
(Figure 10.3 – Jupyter Server Initialization via SSH)



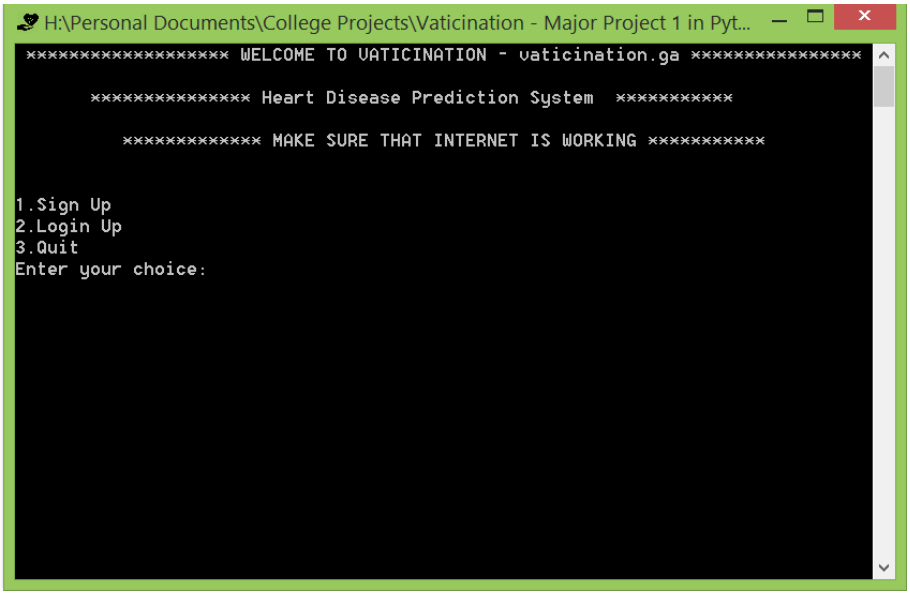
(Figure 10.4 – Jupyter Server with Classifier and Data-set files)



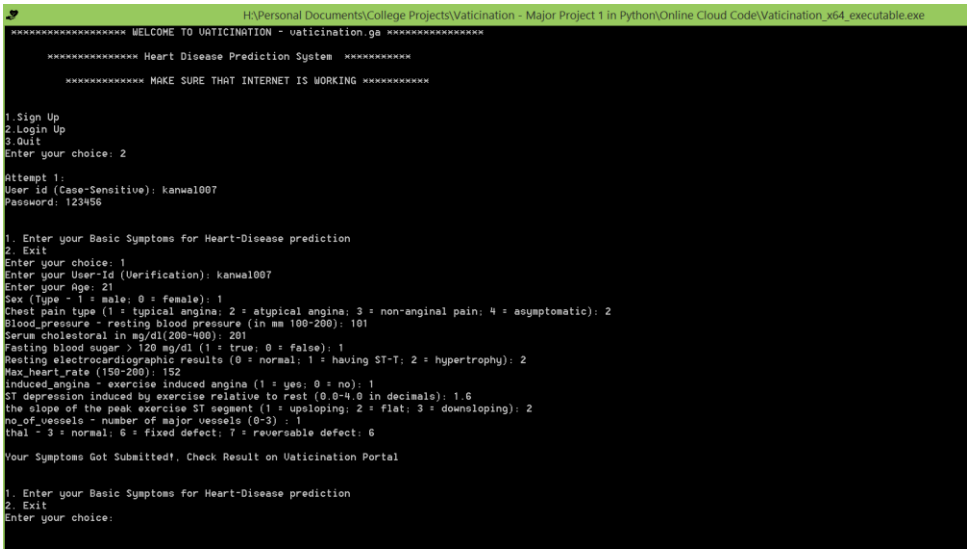
(Figure 10.5 – Jupyter Notebook file of Various Classifiers resulting in best classifier)



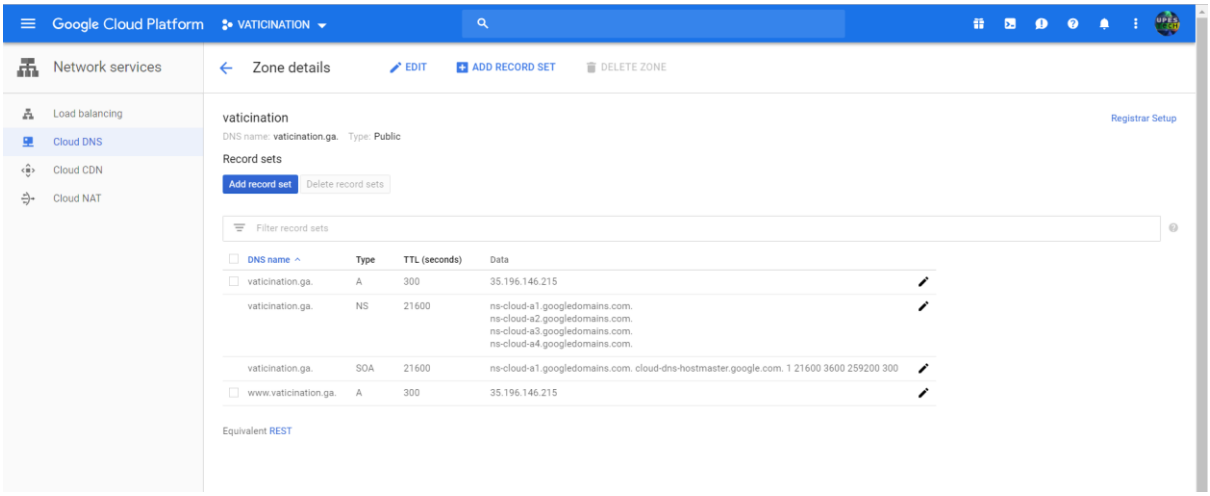
(Figure 10.6 – Remote SQL Server receiving various inputs from end-user)



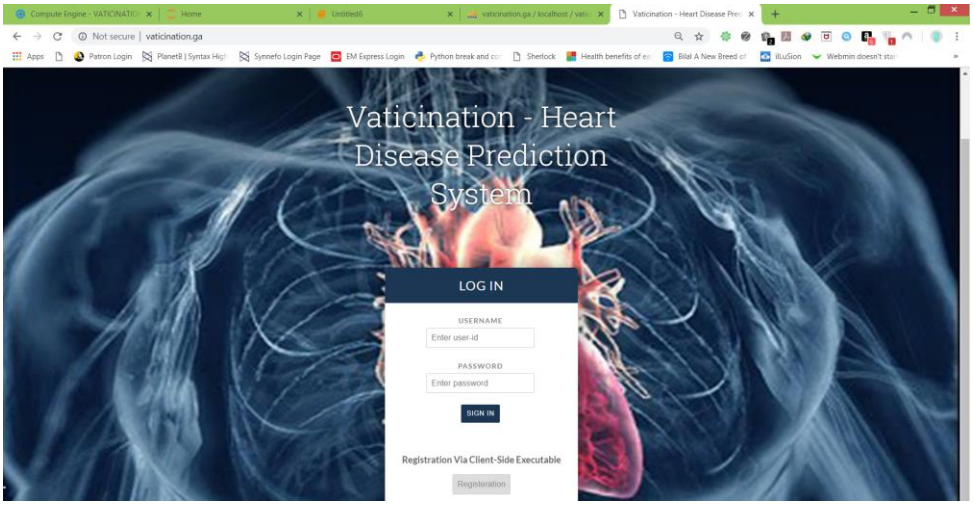
(Figure 10.7 – Client-Side executable comprises of user-registration and login)



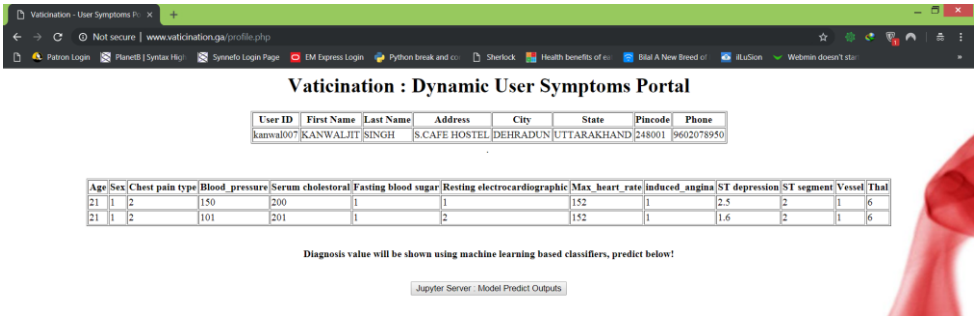
(Figure 10.8 – Client-Side executable taking various symptoms from end-user)



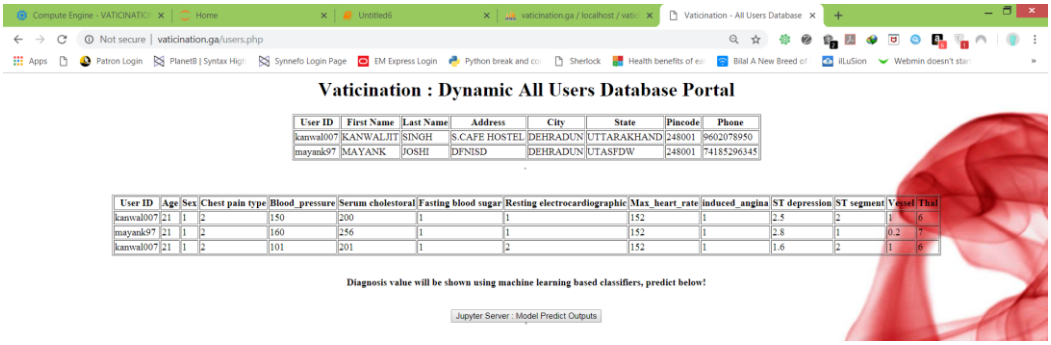
(Figure 10.9 – Google Cloud DNS for mapping vatication.ga to instance IPV4 address)



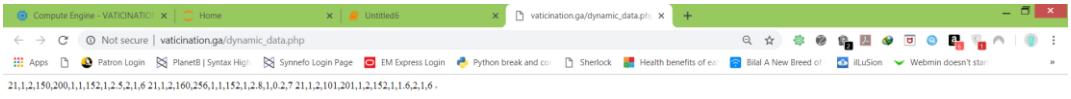
(Figure 10.11 – GUI providing portal for client profile management)



(Figure 10.12 – GUI retrieving the specific client details on login-up)



(Figure 10.13 – Admin GUI displaying all users in a system)



(Figure 10.14 – Plain text URL input exported for trained model)

```
In [28]: #Finally we are selecting Naive Bayes as it is having higher accuracy after taking into consideration the problem of Over-fitting
from sklearn.naive_bayes import GaussianNB
nb= GaussianNB()
nb.fit(X_train,y_train)
#Taking user define input to test the model
X_new= pd.read_csv("http://vaticination.ga/dynamic_data.php")
y_pred = nb.predict(X_new)
print(' '.join(map(str,y_pred)))

         0 0
```

(Figure 10.15 – Trained model output using dynamic inputs in Naïve Bayes)

9. Schedule: (PERT CHART)

Start	80 days – 11 Weeks (Entire Team)			
Analysis & Design 17/08/18 – 31/08/18				
14 Days Entire team (Requirement Analysis and Designing the solution)	Planning Prototype 31/08/18 – 14/09/18 14 Days Entire team (Planning a Prototype with defining all constraints for adapt future design)	Coding 14/09/18 – 26/10/18 42 Days Entire team (Algorithm Implementation)	Testing 26/10/18 – 01/11/18 7 Days Entire team (Debugging)	Documentation 01/11/18 – 04/11/18 3 Days Entire team (Project Completion)

10. References:

[1] V. Kirubha and S. M. Priya, “Survey on Data Mining Algorithms in Disease Prediction,” vol. 38, no. 3, pp. 124 – 128, 2016

[2] M. A. Jabbar, P. Chandra, and B. L. Deekshatulu, “Prediction of risk score for hea rt disease using associative classification and hybrid feature subset selection,” Int. Conf. Intell. Syst. Des. Appl. ISDA, pp. 628 – 634, 2012.

[3] M. Sultana, A. Haider, and M. S. Uddin, “Analysis of data mining techniques for heart disease prediction,” 2016 3rd Int. Conf. Electr. Eng. Inf. Commun. Technol. iCEEiCT 2016 , 2017.

[4] M. Akhil, B. L. Deekshatulu, and P. Chandra, “Classification of Heart Disease Using K - Nearest Neighbor and Genetic Algorithm,” Procedia Technol., vol. 10, pp. 85 – 94, 2013.

[5] P. Sharma and A. P. R. Bhartiya, “Implementation of Decision Tree Algorithm to Analysis the Performance,” Int. J. Adv. Res. Comput. Commun. Eng., vol. 1, no. 10, pp. 861– 864, 2012.