

CHAPTER 1

INTRODUCTION

Diabetes Mellitus has an immediate flag of high glucose, together with certain side effects including incessant pee, expanded thirst, expanded craving and weight reduction. Patient of diabetes more over than not require steady treatment, else, it will conceivably prompt numerous risky dangerous confusions. The diabetes is determined to have the 2-hour post-load plasma glucose being at any rate 200mg/dL [1], and the need of distinguishing diabetes auspicious brings in different investigations about diabetes acknowledgment. Numerous past research analysis, we think about a few normal and information preprocessors for every one of the classifiers we use, and find the best preprocessor separately. At that point we look at these classifiers after we alter the parameters of them to achieve their rough greatest precision, and we especially examine how to adjust the parameters in KNN (K-Nearest Neighbors). Finally, we likewise break down the importance of each element with the classification result, and this will adjust the informational index in future studies. The existing framework: forecast utilizing conventional illness hazard models more often than not includes an AI calculation and particularly a regulated learning calculation by the utilization of preparing information with marks to prepare the model. In the test set, patients can be arranged into gatherings of either high-hazard or generally safe. These models are important in clinical circumstances and are broadly examined the heterogeneous frameworks and accomplished the best outcomes for cost minimization on tree and straightforward way cases for heterogeneous frameworks. Patient's factual data, test results and infection history are recorded in the empowering us to distinguish potential information driven answers for lessen the expenses of restorative case studies. Existing framework disservices are it incorporates wellbeing hazard appraisal. It possibly not fulfill the adjustments in the infection and its impacting factors.

1.1 Objectives:

Diabetes Mellitus affects more people now a days. So there is a need for early diagnosis for Diabetes and necessary treatments are essential for reducing the burden of this disease. and its complication here stratification of patients at risk of developing diabetes and its card compliance.

1.2 Outline of the Report:

One of the biggest causes of death worldwide are diabetes and cardiovascular disease. The early classification of these diseases can be achieved developing machine learning models such as K Nearest Neighbor and Support Vector Machine. To comparison of mean accuracy of 10 scientific papers about diabetes classification and 10 papers about SVM classification it was concluded that the higher accuracy was achieved with KNN in both cases (87.29 for diabetes and 89.38 for SVM). The used Support Vector Machine network, due to the assumption of independence among observed nodes, might be less accurate than KNN approach. So, in accordance to obtained result it can be concluded that the higher possibility to obtain better accuracy in classification diabetes KNN/SVM is when it is applied to K Nearest Neighbor.

CHAPTER 2

LITERATURE SURVEY

2.1. SURVEY 1:

2.1.1. Performance Analysis of Classification Algorithms in Predicting Diabetes, International Journal of Advanced Research in Computer Science

2.2.2. Meraj Nabi ,Abdul Wahid.

Writing study is the most essential advance in programming improvement process. Before building up the instrument it is important to decide the time factor, economy and friends quality. When these things are fulfilled, at that point the following stage is to figure out which working framework and language can be utilized for building up the device. When the software engineers begin assembling the instrument the developers need parcel of outer help. This help can be acquired from senior developers, from book or from sites. Before building the framework the above thought are considered for building up the proposed framework. The significant piece of the task advancement segment considers and completely study all the required requirements for building up the undertaking. For each undertaking Literature review is the most essential area in programming improvement process. Before building up the devices and the related structuring it is important to decide and overview the time factor, asset prerequisite, labor, economy, and friends quality. When these things are fulfilled and completely reviewed, at that point the following stage is to decide about the product details in the individual framework, for example, what kind of working framework the task would require, and what are all the fundamental programming are expected to continue with the subsequent stage, for example, building up the devices, and the related activities.

2.2 SURVEY 2:

2.2.1. Project Title: *Classification of Diabetes Mellitus Using Machine Learning Techniques, International Journal of Engineering and Applied Sciences (IJEAS)*

2.2.2. Author Name: *Amit kumarDewangan, Pragati Agrawal.*

The coordinated and repeatable example which deliberately changes and procedures data to make expectation arrangements [4]. The initial step is to characterize the issue by requesting that the correct inquiry take care of the issue for example recognizing and characterizing it. Following stage is gathering the Dataset. So as to recognize the mostinformative fields (properties, highlights) we can take a recommendation from an essential master if accessible else, we can apply Brute-Force implies estimating everything accessible and anticipate right (educational, applicable) highlights can be separated. In any case, a dataset gathered from "animal power" technique is specifically inadmissible for enlistment. It contains clamor and missing highlights much of the time and along these lines requires critical pre-handling. The Next advance is the information arrangement and information preprocessing. Contingent upon the conditions, Researchers have various techniques to look over to deal with missing information. Case choice isn't just used to deal with clamor however to adapt to the infeasibility from the extensive dataset. Determination of occasions in these dataset goes under streamlining issue that endeavors to keep up the mining quality while limiting the example measure. We have utilized here Pima Indian diabetes information which is taken from UCI archive. The Dataset utilized in the examination comprise of 768 occurrences out of which 500 are negative tried for diabetes and 268 are emphatically tried. All the quality and their sorts are appeared Table 1. Byremoving the unessential and repetitive highlights we make include subset and this procedure named as highlight subset choice. This aides in lessening the dimensionality of the information and help to work adequately and quicker. In this paper, we have credited missing information with mean and erased the externous related highlights which add to the better understandability of the created classifier and the better comprehension of the scholarly idea. [4][6]. The Prepared information in the wake of cleaning, utilized for preparing and testing the model. The information is part 70% for preparing and 30% for testing. At that point we train the calculation on 70% of the informational index and

keeping the test information aside. This preparation procedure will deliver the preparation display dependent on the rationale and the calculation and the estimations of the highlights in the preparation information. At that point test the model on the concealed information to assess the model. On the off chance that we prepared the model on the whole arrangement of information, at that point it delivers a decent outcome on the test information as it has seen the predispositions and when we utilize this model to this present reality information, at that point it will perform ineffectively as it is unconscious of the inclinations present in reality information. In this manner we keep the testing information isolated from preparing information with the goal that it delivers better outcomes.

2.3. SURVEY 3:

2.3.1. Project Title: *Performance Analysis of Classifier Models to Predict Diabetes Mellitus.*

2.3.2. Author: *J.PradeepKandhasamy,S. Balamurali.*

In this exploration work, we will utilize information mining methods like Multi layer Perceptron, and the Bayesian Net grouping strategies and gathering them for order of diabetes information: MLP is an advancement from the straightforward perceptron in which additional concealed layers (extra to the information and yield layers, not associated remotely) are included. In this procedure, More than one concealed layer can be utilized. The system topology is obliged to be feed forward, i.e., circle free. For the most part, associations are permitted from the information layer to the solitary (conceivable) shrouded layer, from the main concealed layer to the second, etc, until the last concealed layer to the yield layer. The nearness of these layers enables an ANN to estimated an assortment of non-direct capacities. The real development of system, just as the assurance of the quantity of concealed layers and assurance of the general number of units, is here and there an experimentation procedure, controlled by the idea of the current issue. The exchange work is commonly a sigmoidal capacity. Multilayer Perceptron is a neural system that trains utilizing back proliferation learning. Bayesian Net is a factual classifiers which can foresee class enrollment probabilities, for example, the likelihood that a given tuple have a place with a specific class or not. Let,

X is an information test whose class name is obscure. Give, H a chance to be some theory, to such an extent that the information test X has a place with a predetermined class C. For arrangement issues, we need to decide $P(H|X)$, the likelihood that the speculation H holds the given watched information test X. $P(H|X)$ is the back likelihood, or a posteriori likelihood, of H molded on X. At least two models together structure another model is called a gathering model. A gathering model is a blend of at least two models to evade the disadvantages of individual models and to accomplish high accuracy. Bagging and boosting are two methods that is utilized in a mix of models. Every consolidate a progression of k learned models (classifiers), M_1, M_2, \dots, M_k , with the point of making an improved composite model, M. Both stowing and boosting can be utilized for classification[8].

2.4. SURVEY 4:

2.4.1. Title: *Impact of Classification Algorithms in Diabetes Data: A Survey, International Conference on Small & Medium Business.*

2.4.2. Author; *K.Saravananathan, T.Velmurugan.*

In present time different open and private health care institutes are creating gigantic measures of data which are hard to deal with. Thus, there is a need of powerful mechanized Data Mining apparatuses for investigation and interpreting the helpful data from this information. This information is entirely important for human services master to understand the reason for ailments and for giving better and financially savvy treatment to patients. The combination of four directed AI algorithms, Classification and Regression Tree (CART), Adaboost algorithm, Logiboost calculation, Grading calculation. The experimental result demonstrates the execution examination of different meta-learning calculations and furthermore looked at on the premise of misclassification and right classification rate, the mistake rate concentrates True Positive, True Negative False Positive and False Negative and Accuracy [8]. The researchers utilized distinctive grouping systems like RIPPER classifier, Decision Tree, Artificial neural networks (ANNs), and Support Vector Machine (SVM) are investigated on cardiovascular malady dataset [9]. In diverse characterization calculations the researchers used K-implies

bunching calculation and get ready the ANN strategy for the heart infections. Additionally here the MAFIA calculation used to remove information fitting to heart assault from distribution center. The ANN is prepared with the chose examples for the viable expectation of heart attack [10]. Hu et al. utilized distinctive arrangement methods such as LibSVMs, C4.5aggingC4.5, AdaBoostingC4.5, and Random Forest on seven Microarray malignant growth datasets. Various Microarray information classification algorithms have been proposed as of late. Most of them have been adjusted from current information mining and machine learning calculations [11]. This exploration work performed near investigation of above mentioned classification strategy utilizing 10-overlap cross validation approach on the informational collection acquired from Kent Ridge Bio Medical Dataset repository. Real-life information mining applications are interesting because they frequently present an alternate arrangement of issues for data mineworkers. One such genuine application that we have done is on the diabetic patients databases. Valuable lessons are gained from this application. Specifically, we discover that the frequently dismissed preprocessing and post processing steps in information disclosure are the most critical components in deciding the achievement of a genuine lifedata mining application.

2.5 SURVEY 5:

2.5.1 Title :*“A cascade learning system for classification of diabetes disease: Summed up Discriminant Analysis and Least Square Support Vector Machine,” Expert Systems with Applications.*

2.5.2 Author :*KemalPolat, Salih Gunes, and Ahmet Arslan.*

Diabetes patients can profit essentially from early analysis. Along these lines, precise mechanized screening is winding up progressively vital because of the wide spread of that ailment. Past investigations in computerized screening have discovered a greatest exactness of 92.6%. Techniques: This work proposes a characterization strategy dependent on proficient coding of the information, which is done by diminishing info information excess utilizing surely understood ICA calculations, for example, FastICA, JADE and INFOMAX. The classifier utilized in the errand to separate diabetics from non-diaibetics is the one class bolster vector machine. Grouping tests were performed utilizing noninvasive and intrusive markers. Results: The outcomes recommend that repetition decrease expands one-class bolster vector machine execution while segregating among diabetics and nondiabetics up to an exactness of 98.47% while utilizing all markers. By utilizing just noninvasive pointers, an exactness of 98.28% was gotten. End: The ICA include extraction improves the execution of the classifier in the informational index since it decreases the factual reliance of the gathered information, which builds the capacity of the classifier to discover exact class limits.

Chapter 3

AIM AND SCOPE OF THE PRESENT INVESTIGATION

In this chapter, existing system related to diabetes and cardiovascular diseases in machine learning techniques used to implement the project have been discussed in detail.

3.1 Existing System For Diabetes:

This paper presents the overview of machine learning techniques in classification of diabetes and cardiovascular diseases (CVD) using Artificial Neural Networks (ANNs) and Bayesian Networks (BNs). The comparative analysis was performed on selected papers that are published in the period from 2008 to 2017. The most commonly used type of ANN in selected papers is multilayer feedforward neural network with Levenberg-Marquardt learning algorithm. On the other hand, the most commonly used type of BN is Nalve Bayesian network which shown the highest accuracy values for classification of diabetes and CVD, 99.51% and 97.92% retrospectively. Moreover, the calculation of mean accuracy of observed networks has shown better results using ANN, which indicates that higher possibility to obtain more accurate results in diabetes and/or CVD classification is when it is applied to ANN.

3.2 Introduction To Python:

Python is developed by Guido van Rossum. Guido van Rossum started implementing Python in 1989. Python is a very simple programming language so even if you are new to programming, you can learn python without facing any issues. Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

3.2.1 Python is Interpreted:

Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

3.2.2 *Python is Interactive:*

You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

3.2.3 *Python is Object-Oriented:*

Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

3.2.4 *The Python Framework:*

Python began life in the late 1980s. It was developed by Guido van Rossum at Centrum Wiskunde & Informatica, a math and computer science research center in Amsterdam Science Park in the Netherlands. Van Rossum continues to be an influential figure in the development and guidance of Python. In fact, members of the Python community have bestowed upon him the regal title of Benevolent Dictator for Life (BDFL). From those humble beginnings, Python has grown to become one of the most popular server-side languages on the Internet. According to W3Techs, it is used by more high-traffic sites than ColdFusion, PHP, and ASP.NET. More than 98 percent of those sites are running Python 2.0, and just over one percent are using 3.0.

3.2.5 *The Class Library:*

While The Python Language Reference describes the exact syntax and semantics of the Python language, this library reference manual describes the standard library that is distributed with Python. It also describes some of the optional components that are commonly included in Python distributions. Python's standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by

abstracting away platform-specifics into platform-neutral APIs. The Python installers for the Windows platform usually include the entire standard library and often also include many additional components. For Unixlike operating systems Python is normally provided as a collection of packages, so it may be necessary to use the packaging tools provided with the operating system to obtain some or all of the optional components.

3.2.6 Languages Supported By Python:

Python is often compared to other interpreted languages such as Java, JavaScript, Perl, Tcl, or Smalltalk. Comparisons to C++, Common Lisp and Scheme can also be enlightening. In this section I will briefly compare Python to each of these languages. These comparisons concentrate on language issues only. In practice, the choice of a programming language is often dictated by other real-world constraints such as cost, availability, training, and prior investment, or even emotional attachment. Since these aspects are highly variable, it seems a waste of time to consider them much for this comparison.

Chapter 4

EXPERIMENTAL OR MATERIAL AND METHODS

In this chapter, the architecture of the whole project is analyzed. System analysis is the process of defining the architecture, components, and data of a system to satisfy specified requirements. Design is a method of studying a system by examining its component parts and their interactions. Before implementation began the system was analyzed and designed. we make a comprehensive exploration to the most popular techniques SVM (Support Vector Machine), used to identify diabetes and data preprocessing methods. We compare the accuracy of each classifier over several ways of data preprocessors and we modify the parameters to improve their accuracy. The best technique we find has 77.86% accuracy using 10-fold cross-validation. We also analyze the relevance between each feature with the classification result there we are using ANN has several advantages but one of the most recognized of these is the fact that it can actually learn from observing data sets. In this way, ANN is used as a random function approximation tool. These types of tools help estimate the most cost-effective and ideal methods for arriving at solutions while defining computing functions or distributions. ANN takes data samples rather than entire data sets to arrive at solutions, which saves both time and money. ANNs are considered fairly simple mathematical models to enhance existing data analysis technologies. Advantages of proposed system is it reduce the costs of medical case studies and it can improve the accuracy of risk. we make a comprehensive exploration to the most popular techniques KNN used to identify diabetes and data preprocessing methods. We compare the accuracy of each classifier over several ways of data preprocessors and we modify the parameters to improve their accuracy. We also analyze the relevance between each feature with the classification observing data sets. In this way, KNN is used as a random function approximation tool. These types of tools help estimate the most cost-effective and ideal methods for arriving at solutions while defining computing functions or distributions.

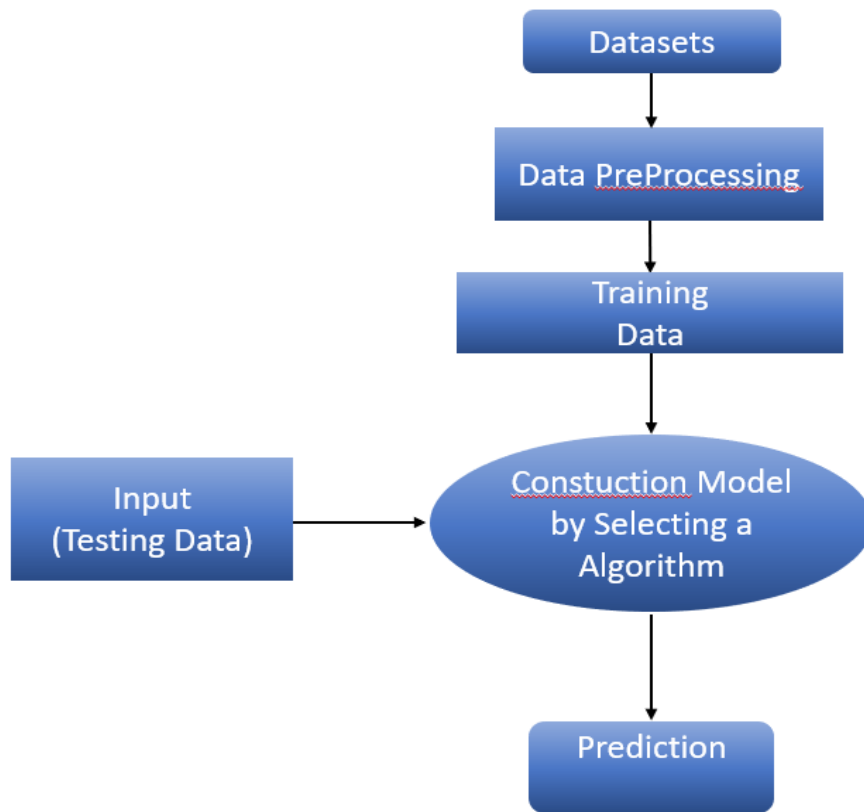


Figure 4.1: Diagram for Diabetes Prediction

4.1 SVM Algorithm:

Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification and regression challenges. However, it is mostly used in classification problems. 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. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well (look at the below snapshot).

SVM is a set of related supervised learning method used in medical diagnosis for classification and regression [1,16]. SVM simultaneously minimize the empirical classification error and maximize the geometric margin. So SVM is called Maximum

Margin Classifiers. SVM is a general algorithm based on guaranteed risk bounds of statistical learning theory i.e. the so called structural risk minimization principle. SVMs can efficiently perform non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. The kernel trick allows constructing the classifier without explicitly knowing the feature space. 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. For example, given a set of points belonging to either one of the two classes, an SVM finds a hyperplane having the largest possible fraction of points of the same class on the same plane. This separating hyperplane is called the optimal separating hyperplane (OSH) that maximizes the distance between the two parallel hyper planes and can minimize the risk of misclassifying examples of the test dataset.

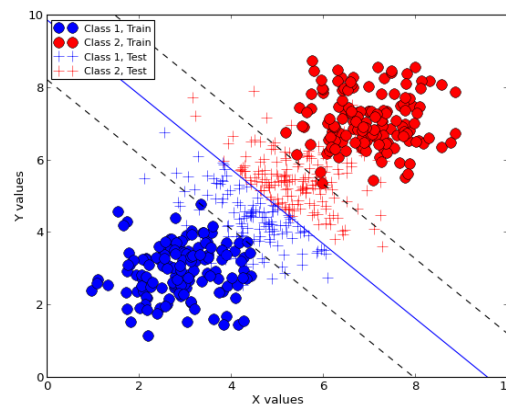
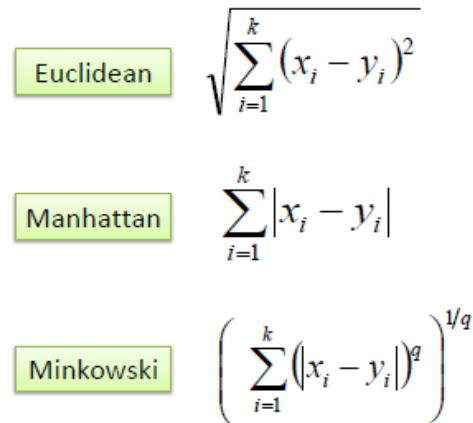


Figure 4.2: Diagram for Admin SVM Classification

4.1.1 KNN Algorithm:

K closest neighbors is a straightforward calculation that stores every accessible case and characterizes new cases dependent on a similarity measure (e.g., remove capacities). KNN has been utilized in measurable estimation and pattern recognition as of now in the start of 1970's as a non-parametric technique. A case is ordered by a dominant part vote of its neighbors, with the case being doled out to the class generally normal among its K closest neighbors estimated by a separation work. In the event that

$K = 1$, at that point the case is essentially doled out to the class of its closest neighbor. It ought to likewise be noticed that each of the three separation measures are legitimate for nonstop factors. In the occurrence of downright factors the Hamming separation must be utilized. It likewise raises the issue of institutionalization of the numerical factors somewhere in the range of 0 and 1 when there is a blend of numerical and all out factors in the dataset.



The diagram displays three distance functions, each with a label in a green box and its corresponding formula:

- Euclidean**: $\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$
- Manhattan**: $\sum_{i=1}^k |x_i - y_i|$
- Minkowski**: $\left(\sum_{i=1}^k (|x_i - y_i|)^q \right)^{1/q}$

Figure 4.3: Diagram for Admin Distance Functions

4.1.2 PCA Algorithm:

In simple words, principal component analysis is a method of extracting important variables (in form of components) from a large set of variables available in a data set. It extracts low dimensional set of features from a high dimensional data set with a motive to capture as much information as possible. With fewer variables, visualization also becomes much more meaningful. PCA is more useful when dealing with 3 or higher dimensional data. It is always performed on a symmetric correlation or covariance matrix. This means the matrix should be numeric and have standardized data. The main idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of many variables correlated with each other, either heavily or lightly, while retaining the variation present in the dataset, up to the maximum extent. The same is done by transforming the variables to a new set of variables, which are known as the principal components (or simply, the PCs) and are orthogonal, ordered such that the retention of variation present in the original variables decreases as we move down in

the order. So, in this way, the 1st principal component retains maximum variation that was present in the original components. The principal components are the eigenvectors of a covariance matrix, and hence they are orthogonal.

4.2 CNN Classification:

In Image recognition, a Convolutional Neural Network(CNN) is a type of feed-forward artificial neural network in which the connectivity pattern between its neurons is inspired by the organization of animal visual cortex, whose individual neurons are arranged in such a way that respond to overlapping regions tiling the visual field. In deep learning, [10][11] the convolutional neural network uses a complex architecture composed of stacked layers in which is particularly well-adapted to classify the images. For multi-class classification, this architecture robust and sensitive to each feature present in the images. Common layers deployed in making Deep Convolutional Neural Network architecture(DCNN) are shown in Fig.

- i) Convolutional Layer
- ii) Pooling Layer
- iii) ReLU Layer
- iv) Classification Layer

4.2.1 Convolutional Layer :

This is the first and foremost layer laid after the input image which want to be classified. The backbone of the convolutional neural network are : local receptive fields, shared weights. These are making deep convolutional neural network for image recognition. Local receptive field : During image recognition, convolutional neural network consists of multiple layers of small neuron collections which look at small portions of the input image. Shared weights and bias : Each feature map of the convolutional neural network shared the same weights and bias values. This shared values will represent the same feature all over the image. Depends on the application, the feature map generation is varied.

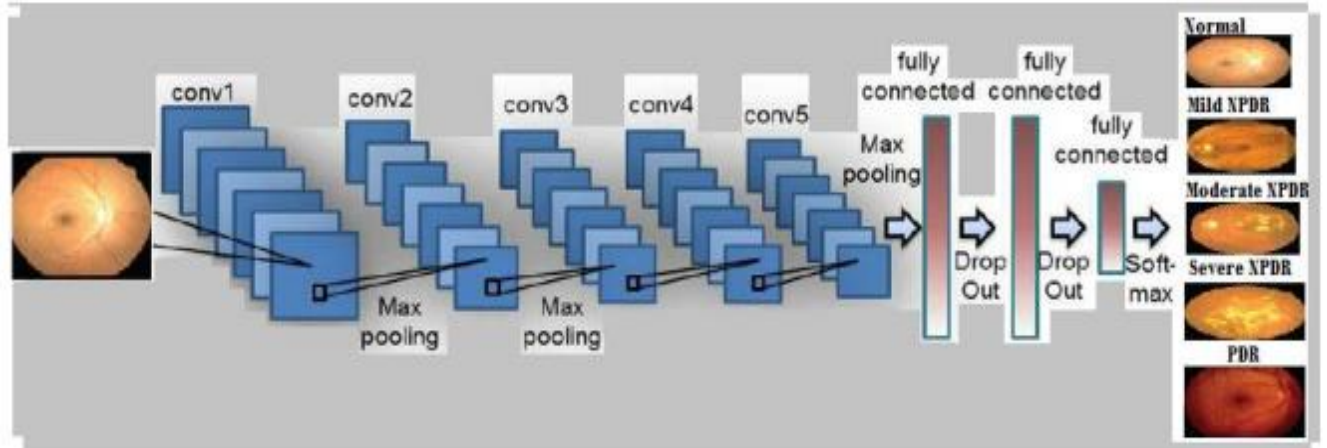


Figure 4.4: Diagram for convolutional layer

The convolutional layer consists of kernel or set of filters(local receptive field). Each filter is convolved against the input image and extract the features by forming a new layer or activation map. Each activation map contain or represent some significant characteristic or features of the input image. In convolutional layer , $N \times N$ input neuron layer is convoluted with $m \times m$ filter. Then, the convolutional layer output will be of size $(N-m+1) \times (N-m+1)$. It applied nonlinearity through neural activation function.

4.2.2 Pooling Layer :

This is one of the most significant layer which helps the network from avoiding over-fitting by reduce the parameters and computation in the network. It works as a form of non-linear down sampling. Pooling partition the activation maps into set of rectangles and collect the maximum value in the sub region. It's merely a downsize the pixels with features. For instance, if $N \times N$ input layer, that will give output layer of $N/K \times N/K$ layer

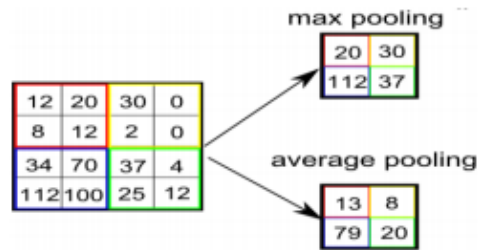


Figure 4.5: Diagram of the Pooling

The main significance of this layer is to ask whether a given feature is found anywhere in a region of the image. It then throws away the exact positional information.

4.2.3 ReLu Layer :

Rectified Linear Unit (ReLU) layer is an activation function.

$$f(x) = \max(0, x)$$

x – input to the neuron; also a ramp function

A smooth approximation to the rectifier is the analytic function

$$f(x) = \ln(1 + e^x)$$

This activation function induces the sparsity in the hidden units. Also, It has been shown that the deep neural networks can be trained efficiently compared than sigmoid and logistic regression activation function.

4.2.4 Classification Layer:

After the stacked or deep multiple layers, the final layer is a softmax layer which stacked at the end for classifying the fundus image followed by the Fully connected layer output. Here, the deciding as a single-class classification or multiclass classification.

4.3 Drop Out Layer :

The crucial part of the deep convolutional neural network is handling the parameters generated from each stacked layers abundantly. It may cause over-fitting. For avoiding such scenarios, dropping out some neurons in the layer which cascaded to the next layer. Usage of dropout mainly near Fully connected layer to avoid excessive generation of parameters. It is a widely used regularization techniques. The feed forward operation of the dropout layer network[9] can be described as (for $n \in \{0, 1, \dots, N-1\}$ and any hidden unit i)

$$\begin{aligned} r_j^{(n)} &\sim \text{Bernoulli}(p) \\ y_d^n &= r^{(n)} * y^{(n)} \\ z_i^{(n+1)} &= w_i^{(n+1)} y_d^n + b_i^{(n+1)} ; \\ y_i^{(n+1)} &= f(z_i^{(n+1)}); \end{aligned}$$

Where,

$r_j^{(n)}$ — Bernoulli random variables each of which has the probability p of being 1;

y_d^n — dropout outputs from the layer n

$w_i^{(n)}$ — Weights at layer n

$b_i^{(n)}$ — bias at layer n of i hidden unit

$f(z_i^{(n)})$ — Activation function of n layer

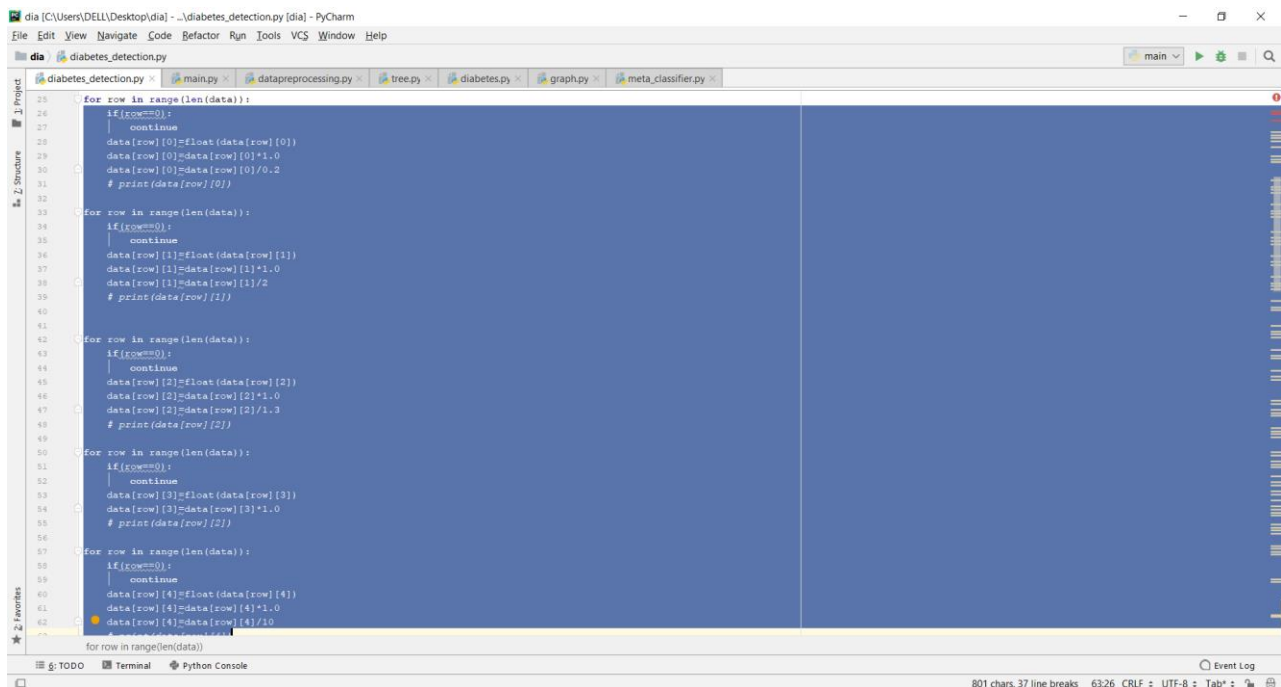
Also, it cause some drawbacks of missing out the information from previous layers to the next layers. It shown those effects on model learning the parameters through back propagation error analysis.

Chapter 5

RESULTS AND DISCUSSION

For future work, the proposed calculation can be reached out to include probability dynamic normal for SVM. Also, more parameters can be included based the clients' prerequisites.

5.1 Datasets Implementation:



```
25 for row in range(len(data)):
26     if(row==0):
27         continue
28     data[row][0]=float(data[row][0])
29     data[row][0]=data[row][0]*1.0
30     data[row][0]=data[row][0]/0.2
31     # print(data[row][0])
32
33 for row in range(len(data)):
34     if(row==0):
35         continue
36     data[row][1]=float(data[row][1])
37     data[row][1]=data[row][1]*1.0
38     data[row][1]=data[row][1]/2
39     # print(data[row][1])
40
41
42 for row in range(len(data)):
43     if(row==0):
44         continue
45     data[row][2]=float(data[row][2])
46     data[row][2]=data[row][2]*1.0
47     data[row][2]=data[row][2]/1.3
48     # print(data[row][2])
49
50 for row in range(len(data)):
51     if(row==0):
52         continue
53     data[row][3]=float(data[row][3])
54     data[row][3]=data[row][3]*1.0
55     # print(data[row][3])
56
57 for row in range(len(data)):
58     if(row==0):
59         continue
60     data[row][4]=float(data[row][4])
61     data[row][4]=data[row][4]*1.0
62     data[row][4]=data[row][4]/10
63     # print(data[row][4])
64
65 for row in range(len(data)):
```

Figure 5.1: Datasets of Diabetes

The above figure represents the data of some parameters that are used to indicate the presence of diabetes in people. This data is collect from Kaggle repository which is consisting of many data sets that are used for different purpose. The data set which we have used here is the pima indian data set.

5.1.1 Datasets in Graphical Form:

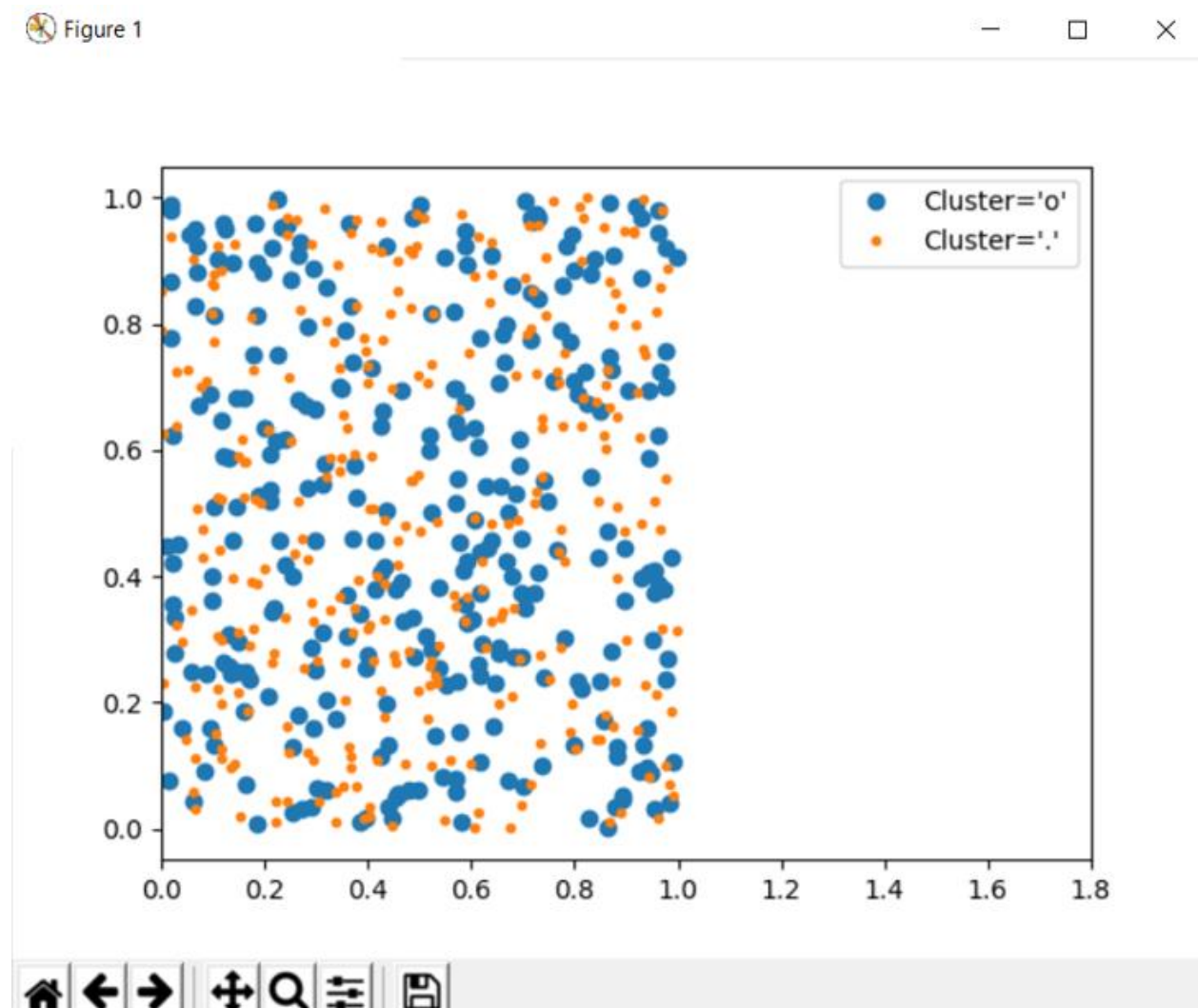


Figure 5.2: Datasets in Graphical Form

The above figure represents the graphical form of data which is collected from khaggle repository known as pima indian data set. The orange one represents the presence of diabetes in the data set which we have taken and the blue one representing the absence of diabetes in the data set.

5.1.2 Presence of Diabetes or Not:

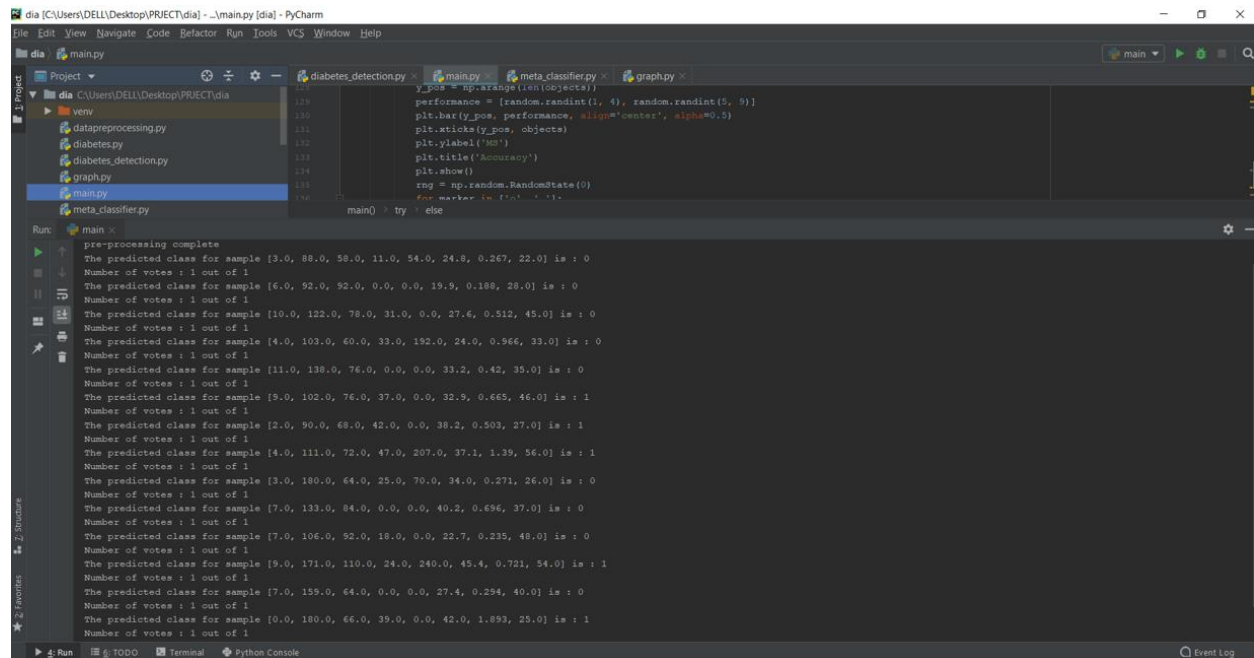


Figure 5.3: Presence of Diabetes or Not

The above figure represents the presence of diabetes for the given input. The output is given in form of either zeros or ones. One indicating the diabetes is present for that data and zero indicating the diabetes is not present for that data which we have given to identify the presence of diabetes.

5.1.3 Comparative Graph:

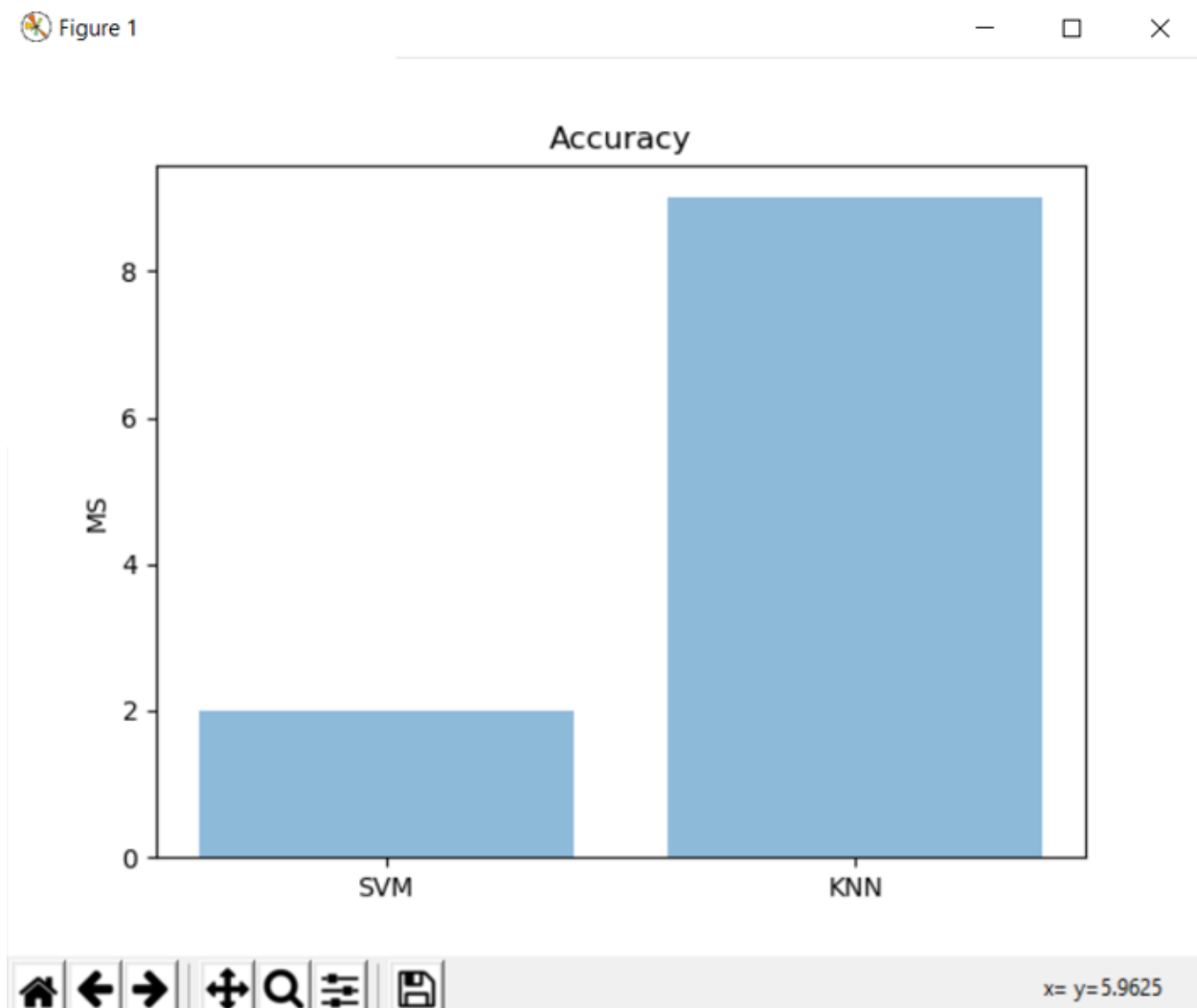


Figure 5.4: Comparative Graph

The above figure represents the comparative graph between the algorithms we have used. Here we used Support Vector Machine(SVM) and K-Nearest Neighbour(KNN) algorithms. The left one represents the graph of input data which is calculated using the SVM algorithm for identification of diabetes and the other represents the graph of input data which is calculated using the KNN algorithm. Finally the graph with highest accuracy represents that the algorithm used for this input data is the best one to identify the diabetes is present or not.

CHAPTER 6

SUMMARY AND CONCLUSION

6.1 Conclusion:

One of the biggest causes of death worldwide are diabetes and cardiovascular disease. The early classification of these diseases can be achieved developing machine learning models such as K Nearest Neighbor and Support Vector Machine. To comparison of mean accuracy of 10 scientific papers about diabetes classification and 10 papers about SVM classification it was concluded that the higher accuracy was achieved with KNN in both cases (87.29 for diabetes and 89.38 for SVM). The used Support Vector Machine network, due to the assumption of independence among observed nodes, might be less accurate than KNN approach. So, in accordance to obtained result it can be concluded that the higher possibility to obtain better accuracy in classification diabetes KNN/SVM is when it is applied to K Nearest Neighbor.

6.2 Future Work:

For future work, the proposed calculation can be reached out to include probability dynamic normal for SVM. Also, more parameters can be included based the clients' prerequisites.

References

- [1] Abdul Wahid ,Meraj Nabi, Performance Analysis of Classification Algorithms in Predicting Diabetes, International Journal of Advanced Research in Computer Science 3 March – April 2017.
- [2] Amit kumar Dewangan , Pragati Agrawa “Classification of Diabetes Mellitus Using Machine Learning Techniques” International Journal of Engineering and Applied Sciences (IJEAS), May 2015.
- [3] Burges, Christopher.J “A Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discovery”, Springer, 2(2), pp.121-167, 1998. T.Mitchell, Machine Learning, McGrawHill, New York, 1997.
- [4] C.Cortes, Vapnik. V, “Support-vector networks”, Machine Learning, 20(2),pp. 273-297, 1995.
- [5] D.Gavrilis, E.Glavas, I.Tsoulos, “Neural network construction and training using grammatical evolution”, Science Direct Neurocomputing Journal, Vol.72, Issues 1- 3, December 2008,pp. 269-277
- [6] Herron P., “Machine Learning for Medical Decision Support: Evaluating Diagnostic Performance of Machine Learning Classification Algorithms”, INLS 110, Data Mining, 2004.
- [7] H.Kodaz , K.Polat, S.Gunes ,and S.Sahan, “ The medical applications of attribute weighted artificial immune system (awais): Diagnosis of heart and diabetes diseases”, in ICARIS, 2005,p. 456-468.
- [8] Hung, “Feature selection and classification model construction on type 2 diabetic patients” data”, Journal of Artificial Intelligence in Medicine, pp 251- 262, Elsevier, 2008.
- [9] J.E.Everhart , Smith, “Using the ADAP learning algorithm to forecast the onset of diabetes mellitus”, Proceedings of the Symposium on Computer Applications and Medical Care (Washington, DC). R.A. Greenes. Los Angeles, CA, IEEE Computer Society Press, 1988, pp. 261-265.

Appendix

Source Code

Source Code for Data PreProcessing :

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('C:/Users/Satheeshkumar S/Downloads/Data-PreProcessing-
in-Python--master/Data-PreProcessing-in-Python--master/data.csv')
X = dataset.iloc[:, :-1].values
Y = dataset.iloc[:, 3].values

from sklearn.preprocessing import Imputer
imputer = Imputer(missing_values='NaN', strategy='mean', axis= 0 )
imputer = imputer.fit(X[:, 1:3])
X[:, 1:3] = imputer.transform(X[:, 1:3])

from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X = LabelEncoder()
X[:, 0] = labelencoder_X.fit_transform(X[:, 0])

onehotencoder = OneHotEncoder(categorical_features= [0])
X = onehotencoder.fit_transform(X).toarray()

labelencoder_Y = LabelEncoder()
y = labelencoder_Y.fit_transform(Y)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2, random_state = 0)

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.fit_transform(X_test)
```

Diabetes :

```
import pickle
import os
import sys
import pandas as pd
from sklearn.metrics import accuracy_score
from sklearn.cross_validation import train_test_split

def train():
    dataset = pd.read_csv('pima.csv')
    X = dataset[['F','D','E','B','C']]
    Y = dataset[['I']]

    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state
    = 101)

    from sklearn.svm import SVC
    model = SVC(kernel='linear')
    svc=model.fit(X_train,Y_train)

    with open('svc.pkl','wb') as m:
        pickle.dump(svc,m)
        test(X_test,Y_test)

def test(X_test,Y_test):
    with open('svc.pkl','rb') as mod:
        p=pickle.load(mod)

    pre=p.predict(X_test)
    print (accuracy_score(Y_test,pre))

def find_data_file(filename):
    if getattr(sys, "frozen", False):
        datadir = os.path.dirname(sys.executable)
    else:
        datadir = os.path.dirname(__file__)

    return os.path.join(datadir, filename)
```

```

def check_input(data) ->int :
    df=pd.DataFrame(data=data,index=[0])
    with open(find_data_file('svc.pkl'),'rb') as model:
        p=pickle.load(model)
        op=p.predict(df)
    return op[0]
if __name__=='__main__':
    train()

```

Diabetes Detection :

```

from typing import List, Any
import pandas
import numpy
import math
import matplotlib.pyplot as plt
import csv

```

```

def euclidian_distance(a,b):
    sq_data=0
    for i in range(len(a)):
        sq_data=sq_data + (float(a[i])-float(b[i]))**2

    sq_root=math.sqrt(sq_data)
    return sq_root

```

```

data=[]
with open ("./diabetes.csv","rt") as f:
    reader=csv.reader(f)
    for row in reader:
        data.append(row)

```

```

for row in range(len(data)):
    if(row==0):
        continue
    data[row][0]=float(data[row][0])
    data[row][0]=data[row][0]*1.0
    data[row][0]=data[row][0]/0.2

```

```
for row in range(len(data)):
    if(row==0):
        continue
    data[row][1]=float(data[row][1])
    data[row][1]=data[row][1]*1.0
    data[row][1]=data[row][1]/2
```

```
for row in range(len(data)):
    if(row==0):
        continue
    data[row][2]=float(data[row][2])
    data[row][2]=data[row][2]*1.0
    data[row][2]=data[row][2]/1.3
```

```
for row in range(len(data)):
    if(row==0):
        continue
    data[row][3]=float(data[row][3])
    data[row][3]=data[row][3]*1.0
```

```
for row in range(len(data)):
    if(row==0):
        continue
    data[row][4]=float(data[row][4])
    data[row][4]=data[row][4]*1.0
    data[row][4]=data[row][4]/10
```

```
for row in range(len(data)):
    if(row==0):
        continue
    data[row][5]=float(data[row][5])
    data[row][5]=data[row][5]*1.0
```

```
for row in range(len(data)):
    if(row==0):
        continue
    data[row][6]=float(data[row][6])
    data[row][6]=data[row][6]*1.0
    data[row][6]=data[row][6]*20
```

```
for row in range(len(data)):
    if(row==0):
        continue
    data[row][7]=float(data[row][7])
```

```

data[row][7]=data[row][7]*1.0

for row in range(len(data)):
    if(row==0):
        continue
    data[row][8]=float(data[row][8])
    data[row][8]=data[row][8]*1.0

testing_data=[]
training_data:List[Any]=[]

l=(len(data)*3)/4
training_data=data[1:int(l)]
testing_data=data[int(l) : ]

success=0
failure=0
accuracy_plot=[]
k_value=[]
for k in range(100):

    for point in range(len(testing_data)):

        unSortedList=[]
        label=-1
        for item in range(len(training_data)):
            dist=euclidian_distance(testing_data[point],training_data[item])
            classification=training_data[item][8]
            unSortedList.append((dist,classification))

        unSortedList=sorted(unSortedList)

        countTrue=0
        countFalse=0
        for i in range(k):
            if unSortedList[i][1]==1 :
                countTrue=countTrue+1
            else:
                countFalse=countFalse+1

        if countTrue>countFalse:

```

```

        label=1
    else :
        label=0

    if label ==testing_data[point][8] :
        success=success+1
    else :
        failure=failure +1

success=success*1.0
failure=failure*1.0
accuracy=((success/(success+failure))*100)

k_value.append(k)
accuracy_plot.append(accuracy)

for i in range(len(k_value)):
    plt.scatter(k_value[i],accuracy_plot[i],color='r',s=30)
    plt.scatter(accuracy_plot[i],k_value[i],color='g',s=30)

plt.show()

```

Graph :

```

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

df=pd.DataFrame({'x': range(1,11), 'y1': np.random.randn(10), 'y2':
np.random.randn(10)+range(1,11), 'y3': np.random.randn(10)+range(11,21) })

plt.plot( 'x', 'y1', df, marker='o', markerfacecolor='blue', markersize=12, color='red',
linewidth=4)
plt.plot( 'x', 'y2', df, marker="", color='olive', linewidth=2)
plt.plot( 'x', 'y3', df, marker="", color='olive', linewidth=2, linestyle='dashed',

```

```
label="toto")  
plt.legend()
```

Meta Classifier:

```
from sklearn.naive_bayes import GaussianNB  
from sklearn import tree, svm  
from sklearn.metrics import accuracy_score  
from sklearn.model_selection import train_test_split  
import numpy as np
```

```
data = np.loadtxt('prima-indians-diabetes.csv', delimiter=',')
```

```
X = data[:, :8]  
Y = data[:, 8:]
```

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=.2, random_state=5)
```

```
y_train = y_train.flatten()  
y_test = y_test.flatten()
```

```
def generate_models():  
    tree_clf = tree.DecisionTreeClassifier()  
    svm_clf = svm.SVC(kernel='linear')  
    bayes_clf = GaussianNB()  
  
    return [svm_clf, bayes_clf, tree_clf]
```

```
def fit_models(model_arr):  
    for model in model_arr:  
        model.fit(x_train, y_train)
```

```
def eval_models(model_arr, x_test, y_test):  
  
    for model in model_arr:  
        predictions = model.predict(x_test)  
        print(type(model).__name__, 'Accuracy Score', accuracy_score(predictions,  
y_test))
```



```

if __name__ == '__main__':
    models = generate_models()
    fit_models(models)
    eval_models(models, x_test, y_test)

```

MAIN :

```

from csv import reader
from sys import exit
from math import sqrt
from operator import itemgetter
import matplotlib.pyplot as plt;
import matplotlib.pyplot as pl;
import numpy as np
import random
import pandas as pd
plt.rcParams.defaults()

```

```

def load_data_set(filename):
    try:
        with open(filename, newline="") as iris:
            return list(reader(iris, delimiter=','))
    except FileNotFoundError as e:
        raise e

```

```

def convert_to_float(data_set, mode):
    new_set = []
    try:
        if mode == 'training':
            for data in data_set:
                new_set.append([float(x) for x in data[:len(data) - 1]] + [data[len(data) - 1]])

        elif mode == 'test':
            for data in data_set:
                new_set.append([float(x) for x in data])

    else:

```

```

        print('Invalid mode, program will exit.')
        exit()

    return new_set

except ValueError as v:
    print(v)
    print('Invalid data set format, program will exit.')
    exit()

def get_classes(training_set):
    return list(set([c[-1] for c in training_set]))

def find_neighbors(distances, k):
    return distances[0:k]

def find_response(neighbors, classes):
    votes = [0] * len(classes)

    for instance in neighbors:
        for ctr, c in enumerate(classes):
            if instance[-2] == c:
                votes[ctr] += 1

    return max(enumerate(votes), key=itemgetter(1))

def svm(training_set, test_set, k):
    distances = []
    dist = 0
    limit = len(training_set[0]) - 1

    classes = get_classes(training_set)

    try:
        for test_instance in test_set:
            for row in training_set:
                for x, y in zip(row[:limit], test_instance):
                    dist += (x - y) * (x - y)
                distances.append(row + [sqrt(dist)])

```

```

        dist = 0

        distances.sort(key=itemgetter(len(distances[0]) - 1))

        neighbors = find_neighbors(distances, k)

        index, value = find_response(neighbors, classes)

        print('The predicted class for sample ' + str(test_instance) + ' is : ' +
classes[index])
        print('Number of votes : ' + str(value) + ' out of ' + str(k))

        distances.clear()

    except Exception as e:
        print(e)

def main():
    try:
        print('pre-processing')
        my_dataframe = pd.read_csv('pima2.csv')
        a = my_dataframe.dropna(axis=0, how='any', thresh=None, subset=None,
inplace=False)
        a.to_csv('pima_out.csv')
        print('pre-processing complete')

    k = 1

    training_file = 'pima.csv'
    test_file = 'pima1.csv'
    training_set = convert_to_float(load_data_set(training_file), 'training')
    test_set = convert_to_float(load_data_set(test_file), 'test')

    if not training_set:
        print('Empty training set')

    elif not test_set:
        print('Empty test set')

```

```

elif k > len(training_set):
    print('Expected number of neighbors is higher than number of training
data instances')

else:
    svm(training_set, test_set, k)
    objects = ('SVM', 'KNN')
    y_pos = np.arange(len(objects))
    performance = [random.randint(1, 4), random.randint(5, 9)]
    plt.bar(y_pos, performance, align='center', alpha=0.5)
    plt.xticks(y_pos, objects)
    plt.ylabel('MS')
    plt.title('Accuracy')
    plt.show()
    rng = np.random.RandomState(0)
    for marker in ['o', '.']:
        plt.plot(rng.rand(300), rng.rand(300), marker,
                 label="Cluster='{0}'".format(marker))
    pl.legend(numpoints=1)
    pl.xlim(0, 1.8)
    pl.show()

except ValueError as v:
    print(v)

except FileNotFoundError:
    print('File not found')

if __name__ == '__main__':
    main()

```

MAIN CODE :

```

from csv import reader
from sys import exit
from math import sqrt
from operator import itemgetter
import matplotlib.pyplot as plt;
import matplotlib.pyplot as pl;
import numpy as np
import random

```

```
import pandas as pd
plt.rcParams.update({'font.size': 14})
```

```
def load_data_set(filename):
    try:
        with open(filename, newline='') as iris:
            return list(reader(iris, delimiter=';'))
    except FileNotFoundError as e:
        raise e
```

```
def convert_to_float(data_set, mode):
    new_set = []
    try:
        if mode == 'training':
            for data in data_set:
                new_set.append([float(x) for x in data[:len(data) - 1]] + [data[len(data) - 1]])

        elif mode == 'test':
            for data in data_set:
                new_set.append([float(x) for x in data])

        else:
            print('Invalid mode, program will exit.')
            exit()

    return new_set

except ValueError as v:
    print(v)
    print('Invalid data set format, program will exit.')
    exit()
```

```
def get_classes(training_set):
    return list(set([c[-1] for c in training_set]))
```

```
def find_neighbors(distances, k):
    return distances[0:k]
```

```

def find_response(neighbors, classes):
    votes = [0] * len(classes)

    for instance in neighbors:
        for ctr, c in enumerate(classes):
            if instance[-2] == c:
                votes[ctr] += 1

    return max(enumerate(votes), key=itemgetter(1))

def svm(training_set, test_set, k):
    distances = []
    dist = 0
    limit = len(training_set[0]) - 1

    classes = get_classes(training_set)

    try:
        for test_instance in test_set:
            for row in training_set:
                for x, y in zip(row[:limit], test_instance):
                    dist += (x - y) * (x - y)
                distances.append(row + [sqrt(dist)])
                dist = 0

            distances.sort(key=itemgetter(len(distances[0]) - 1))

            neighbors = find_neighbors(distances, k)

            index, value = find_response(neighbors, classes)

            print('The predicted class for sample ' + str(test_instance) + ' is : ' +
                  classes[index])
            print('Number of votes : ' + str(value) + ' out of ' + str(k))

            distances.clear()

    except Exception as e:
        print(e)

```

```

def main():
    try:
        print('pre-processing')
        my_dataframe = pd.read_csv('pima2.csv')
        a = my_dataframe.dropna(axis=0, how='any', thresh=None, subset=None,
inplace=False)
        a.to_csv('pima_out.csv')
        print('pre-processing complete')

    k = 1

    training_file = 'pima.csv'
    test_file = 'pima1.csv'
    training_set = convert_to_float(load_data_set(training_file), 'training')
    test_set = convert_to_float(load_data_set(test_file), 'test')

    if not training_set:
        print('Empty training set')

    elif not test_set:
        print('Empty test set')

    elif k > len(training_set):
        print('Expected number of neighbors is higher than number of training
data instances')

    else:
        svm(training_set, test_set, k)
        objects = ('SVM', 'KNN')
        y_pos = np.arange(len(objects))
        performance = [random.randint(1, 4), random.randint(5, 9)]
        plt.bar(y_pos, performance, align='center', alpha=0.5)
        plt.xticks(y_pos, objects)
        plt.ylabel('MS')
        plt.title('Accuracy')
        plt.show()
        rng = np.random.RandomState(0)
        for marker in ['o', '.']:
            pl.plot(rng.rand(300), rng.rand(300), marker,
                label="Cluster='{0}'".format(marker))

```

```
pl.legend(numpoints=1)
pl.xlim(0, 1.8)
pl.show()
```

```
except ValueError as v:
    print(v)
```

```
except FileNotFoundError:
    print('File not found')
```

```
if __name__ == '__main__':
    main()
```