DETECTING OUTLIER FROM HIGH DIMENSIONAL DATA USING MACHINE LEARNING

Linga Reddy Sruthi[*1],Manasa[*2], Thrinesh[*3], Vimal Thanigai[*4]

[*1] B. Tech Student, Department of CSE, Dayananda Sagar University ,Bangalore, India
[*2]B. Tech Student, Department of CSE, Dayananda Sagar University ,Bangalore, India
[*3] B. Tech Student, Department of CSE, Dayananda Sagar University ,Bangalore, India
[*4] B. Tech Student, Department of CSE, Dayananda Sagar University ,Bangalore, India

ABSTRACT

Detecting outliers is an important problem. Most of its applications typically possess high dimensional datasets. In high dimensional space, the data becomes sparse which implies that every object can be regarded as an outlier from the point of view of similarity. Furthermore, a fundamental issue is that the notion of which objects are outliers typically varies between users, problem domains or, even, datasets. In this paper, we present a novel robust solution which detects high dimensional outliers based on user examples and tolerates incorrect inputs. It studies the behaviour of projections of such a few examples, to discover further objects that are outstanding in the projection where many examples are outlying. Our experiments on both real and synthetic datasets demonstrate the ability of the proposed method to detect outliers corresponding to the user example

Keywords: SVM, KNN and Decision tree classifier

INTRODUCTION:

Detecting outliers is a critical task in applications such as fraud detection, financial analysis, health monitoring and so on. It is also a difficult problem especially when the data is in high dimensional spaces, which is often the case with its most applications. It has been shown that by examining the behaviour of the data in low dimensional projections, meaningful outliers are likely to be defined . It is also observed in that different objects may be detected as outliers with respect to different subsets of dimensions. To detect outliers, a fundamental issue is that, the notion of what is an outlier varies among users, problem domains and even datasets (problem instances). The issue intuitively leads to a result that different objects may be regarded as outliers, depending on different views from which we examine the datasets. Being experts in their problem domain, not in outlier detection, users often have a few example outliers in hand, which may "describe" their intentions and they want to find more objects that exhibit "outlier-ness" characteristics similar to those examples. Existing systems do not provide a direct way to incorporate such examples in the discovery process to find out the "hidden" outlier concept that users may have in mind. A main problem is the lack of knowledge of the outliers. Users may provide example outliers which are NOT "true" outliers. Or else, all of the provided examples can NOT be treated as outliers from the SAME views of projections. Therefore, an solution of outlier detection based on examples should be robust to allow incorrect inputs. In this paper, we propose a robust outlier detection method specially designed for high dimensional datasets that can discover the desired view of "outlier-ness", based on a small number of examples. The remainder of the paper is organized as follows: In the next section, we discuss related work on outlier detection. In section 3, we discuss the definition of "outlier-ness" in a high dimensional dataset. Section 4 presents the novel method to detect outliers based on examples and its design decisions in detail. The experiment evaluation on both synthetic and real datasets are reported in Section 5. Finally, section 6 concludes the paper

LITERATURE REVIEW:

[1] Charu C. Aggarwal and Philip S. Yu. Outlier Detection for High Dimensional Data. In Proc. SIGMOD Conf., 2001.

The outlier detection problem has important applications in the eld of fraud detection, network robustness analysis, and intrusion detection. Most such applications are high dimensional domains in which the data can contain hun- dreds of dimensions. Many recent algorithms use concepts of proximity in order to nd outliers based on their relationship to the rest of the data. However, in high dimensional space, the data is sparse and the notion of proximity fails to retain its meaningfulness. In fact, the sparsity of high dimensional data implies that every point is an almost equally good outlier from the perspective of proximity-based denitions. Consequently, for high dimensional data, the notion of nding meaningful outliers becomes substantially more complex and non-obvious. In this paper, we discuss new techniques for outlier detection which nd the outliers by studying the behaviour of pro junctions from the data set.

2. K. A. De Jong. Analysis of the Behavior of a Class of Genetic Adaptive Systems. Ph. D. Dissertation, University of Michigan, Ann Arbor, MI, 1975.

Artificial bee colony (ABC) algorithm is one of the popular swarm intelligence algorithms. ABC has been developed by being inspired foraging and waggle dance behaviors of real bee colonies in 2005. Since its invention in 2005, many ABC models have been proposed in order to solve different optimization problems. In all the models proposed, there are only one scout bee and a constant limit value used as control

parameters for the bee population. In this study, the performance of ABC algorithm on the numeric optimization problems was analyzed by using different number of scout bees and limit values. Experimental results show that the results obtained by using more than one scout bee and different limit values, are better than the results of basic ABC. Therefore, the control parameters of the basic ABC should be tuned according to given class of optimization problems. In this paper, we propose reasonable value ranges of control parameters for the basic ABC in order to obtain better results on the numeric optimization problems.

[3] D. E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, Reading, MA, 1989.

With the development of Computerized Business Application, the amount of data is increasing exponentially. Cloud computing provides high performance computing resources and mass storage resources for massive data processing. In distributed cloud computing systems, data intensive computing can lead to data scheduling between data centers. Reasonable data placement can reduce data scheduling between the data centers effectively, and improve the data acquisition efficiency of users. In this paper, the mathematical model of data scheduling between data centers is built. By means of the global optimization ability of the genetic algorithm, generational evolution produces better approximate solution, and gets the best approximation of the data placement at last. The experimental results show that genetic algorithm can effectively work out the approximate optimal data placement, and minimize data scheduling between data centers.

[4] A. K. Jain, M. N. Murty, and P. J. Flynn. Data Clustering: a Review. ACM Comp. Surveys, 31(3):264- 323,1999

Clustering is the unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters). The clustering problem has been addressed in many contexts and by researchers in many disciplines; this reflects its broad appeal and usefulness as one of the steps in exploratory data analysis. However, clustering is a difficult problem combinatorially, and differences in assumptions and contexts in different communities has made the transfer of useful generic concepts and methodologies slow to occur. This paper presents an overview of pattern clustering methods from a statistical pattern recognition perspective, with a goal of providing useful advice and references to fundamental concepts accessible to the broad community of clustering practitioners. We present a taxonomy of clustering techniques, and identify cross-cutting themes and recent advances. We also describe some important applications of clustering algorithms such as image segmentation, object recognition, and information retrieval.

[5] C. Zhu, H. Kitagawa, S. Papadimitriou, and C. Faloutsos. OBE: Outlier by Example. In Proc. PAKDD. pp. 222-234, 2004

Outlier detection in large datasets is an important problem. There are several recent approaches that employ very reasonable definitions of an outlier. However, a fundamental issue is that the notion of which objects are outliers typically varies between users or, even, datasets. In this paper, we present a novel solution to this problem, by bringing users into the loop. Our OBE (Outlier By Example) system is, to the best of our knowledge, the first that allows users to give some examples of what they consider as outliers. Then, it can directly incorporate a small number of such examples to successfully discover the hidden concept and spot further objects that exhibit the same "outlier-ness" as the examples. We describe the key design decisions and algorithms in building such a system and demonstrate on both real and synthetic datasets that OBE can indeed discover outliers that match the users' intentions.

[6] T. Johnson, I. Kwok, and R. T. Ng. Fast Computation of 2-Dimensional Depth Contours. In Proc. KDD,pp. 224-228, 1998.

One person's noise is another person's signal." For many applications, including the detection of credit card frauds and the monitoring of criminal activities in electronic commerce, an important knowledge discovery problem is the detection of exceptional/outlying events. In computational statistics, a depth-based approach detects outlying data points in a 2-D dataset by, based on some definition of depth, organizing the data points in layers, with the expectation that shallow layers are more likely to contain outlying points than are the deep layers. One robust notion of depth, called depth contours, was introduced by Tukey. ISODEPTH, developed by Ruts and Rousseeuw, is an algorithm that computes 2-D depth contours. In this paper, we give a fast algorithm, FDC, which computes the first k 2-D depth contours by restricting the computation to a small selected subset of data points, instead of examining all data points. Consequently, FDC scales up much better than ISODEPTH. Also, while ISODEPTH relies on the non-existence of collinear points, FDC is robust against collinear points. Keywords: depth contours, computational statistics, convex hulls

EXISTING METHOD:

The increasing growth of machine learning, computer techniques divided into traditional methods and machine learning methods. This section describes the related works of classification of detecting outlier from high dimensional data Using Machine Learning Model Detection and how machine learning methods are better than traditional methods. The existing method in this project have a certain flow is used for model development Decision Tree is used algorithms in existing system. But it requires large memory and result is not accurate.

Disadvantages:

- Accuracy low
- Requires more time
- Difficult to handle

PROPOSED SYSTEM:

Many machine learning algorithms are available for detecting outlier from high dimensional data Using Machine Learning algorithm are SVM, KNN and Ethereum and we used proposed Ensemble Voting method and compute best method . In this stage we have first implement Random Forest Classifier algorithm on these datasets and the implement algorithm individual

then we are implement Voting Ensemble algorithm for combine these results and an compute the final accuracy.

Advantages:

- Requires less time
- Good Accuracy
- Easy to handle

BLOCK DIAGRAM:

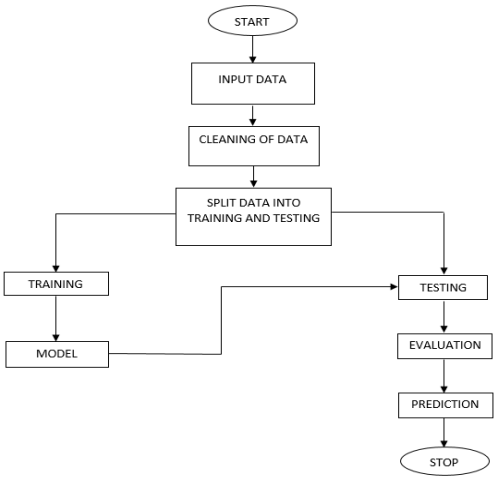
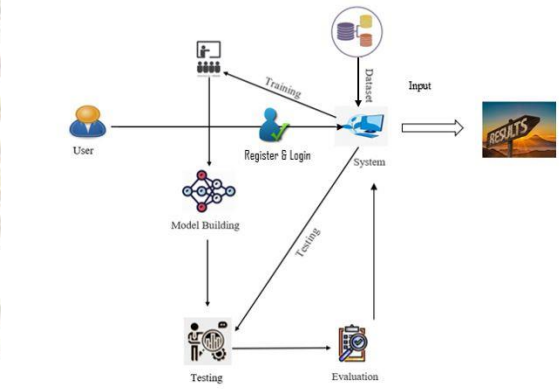


Fig. Block diagram of proposed method

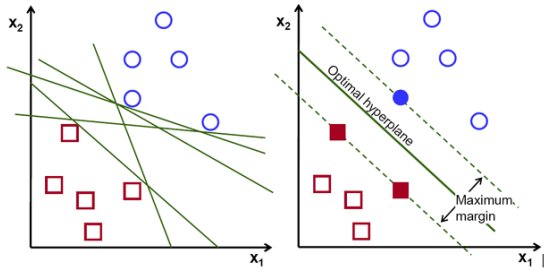
ARCHITECTURE:



ALGORITHMS

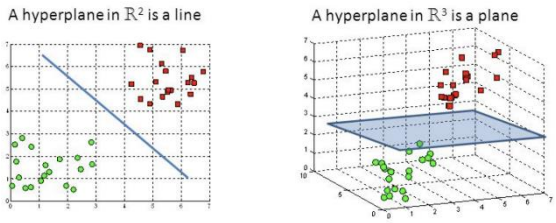
SUPPORT VECTOR MACHINES:

The objective of the support vector machine algorithm is to find a hyper plane in an N-dimensional space (N — the number of features) that distinctly classifies the data points.



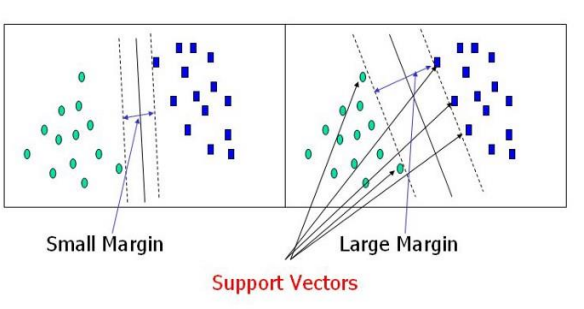
Possible hyper planes :

To separate the two classes of data points, there are many possible Hyper planes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e. the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.



Hyper planes in 2D and 3D feature space

Hyper planes are decision boundaries that help classify the data points. Data points falling on either side of the hyper plane can be attributed to different classes. Also, the dimension of the hyper plane depends upon the number of features. If the number of input features is 2, then the hyper plane is just a line. If the number of input features is 3, then the hyper plane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.



Support Vectors

Support vectors are data points that are closer to the hyper plane and influence the position and orientation of the hyper plane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyper plane. These are the points that help us build our SVM.

$$min_w \lambda \parallel w \parallel^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle)_+$$

Large Margin Intuition

In logistic regression, we take the output of the linear function and squash the value within the range of [0,1] using the sigmoid function. If the squashed value is greater than a threshold value (0.5) we assign it a label 1, else we assign it a label 0. In SVM, we take the output of the linear function and if that output is greater than 1, we identify it with one class and if the output is -1, we identify is with another class. Since the threshold values are changed to 1 and -1 in SVM, we obtain this reinforcement range of values ([-1, 1]) which acts as margin.

Cost Function and Gradient Updates

In the SVM algorithm, we are looking to maximize the margin between the data points and the hyper plane. The loss function that helps maximize the margin is hinge loss.

Hinge loss function (function on left can be represented as a function on the right)

$$c(x, y, f(x)) = (1 - y * f(x))_+ \quad \text{if } y * f(x) \geq 1$$

The cost is 0 if the predicted value and the actual value are of the same sign. If they are not, we then calculate the loss value. We also add a regularization parameter the cost function. The objective of the regularization parameter is to balance the margin maximization and loss. After adding the regularization parameter, the cost functions looks as below.

Loss function for SVM

Now that we have the loss function, we take partial derivatives with respect to the weights to find the gradients. Using the gradients, we can update our weights.

Gradients

When there is no misclassification, i.e. our model correctly predicts the class of our data point, we only have to update the gradient from the regularization parameter.

$$w = w - \alpha \cdot (2\lambda w)$$

Gradient Update — No misclassification

When there is a misclassification, i.e. our model make a mistake on the prediction of the class of our data point, we include the loss along with the regularization parameter to perform gradient update.

$$w = w + \alpha \cdot (y_i \cdot x_i - 2\lambda w)$$

K-Nearest Neighbor

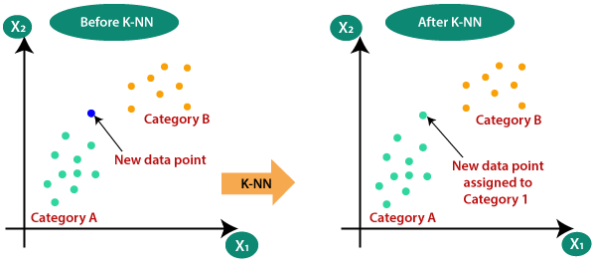
- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

$$\frac{\delta}{\delta w_k} \lambda \| w \|^2 = 2\lambda w_k$$

$$\frac{\delta}{\delta w_k}(1 - y_i\langle x_i, w\rangle)_+ = \begin{cases} 0, & \text{if } y_i\langle x_i, w\rangle \geq 1 \\ -y_i x_{ik}, & \text{else} \end{cases}$$

Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



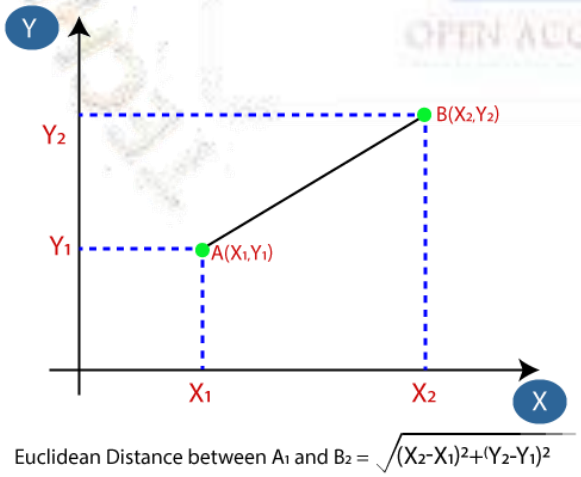
How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

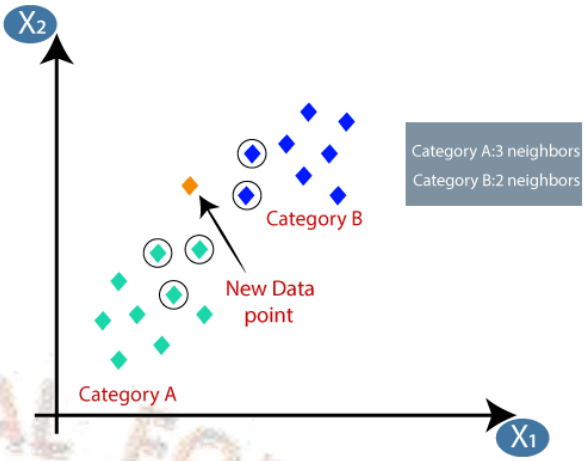
- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:

- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

Advantages of KNN Algorithm:

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.
- **Decision Tree**
- A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. Though a commonly used tool in data mining for deriving a strategy to reach a particular goal.

- A decision tree is drawn upside down with its root at the top. In the image on the left, the bold text in black represents a condition/internal node, based on which the tree splits into branches/ edges. The end of the branch that doesn't split anymore is the decision/leaf, in this case, whether the passenger died or survived, represented as red and green text respectively.
- Although, a real dataset will have a lot more features and this will just be a branch in a much bigger tree, but you can't ignore the simplicity of this algorithm. The feature importance is clear and relations can be viewed easily. This methodology is more commonly known as learning decision tree from data and above tree is called Classification tree as the target is to classify passenger as survived or died. Regression trees are represented in the same manner, just they predict continuous values like price of a house. In general, Decision Tree algorithms are referred to as CART or Classification and Regression Trees.
- So, what is actually going on in the background? Growing a tree involves deciding on which features to choose and what conditions to use for splitting, along with knowing when to stop. As a tree generally grows arbitrarily, you will need to trim it down for it to look beautiful. Let's start with a common technique used for splitting

OUTPUT SCREEN SHOTS WITH DESCRIPTION

Home Page:

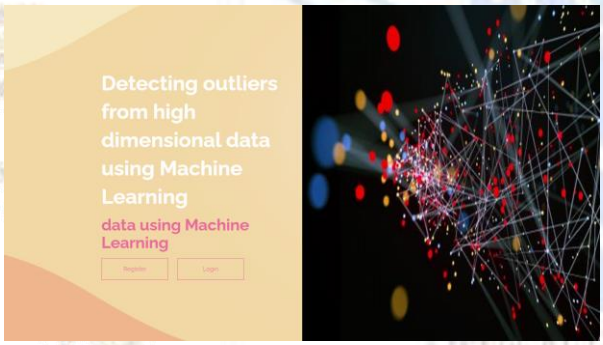


Fig1: Home Page

User Registration page: User can register with required details.

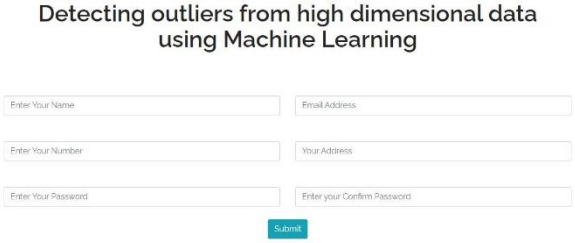


Fig2: Registration Page**.**

User Login: User can login with valid credentials.

Fig3: Login Page**.**

Model Training:

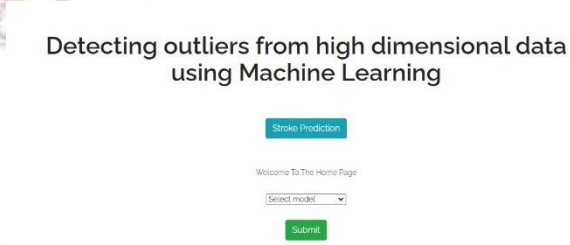


Fig4: Model Page**.**

Prediction:

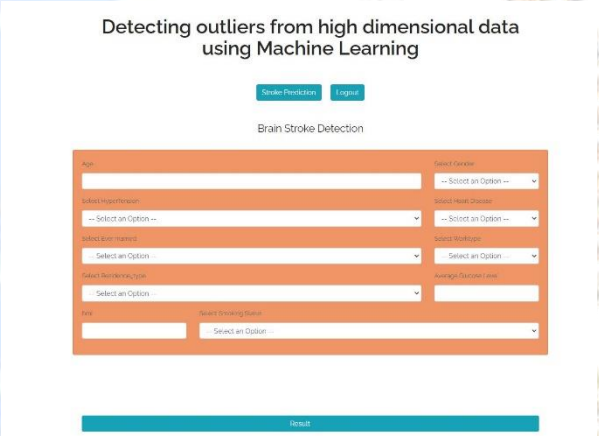
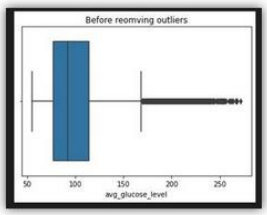
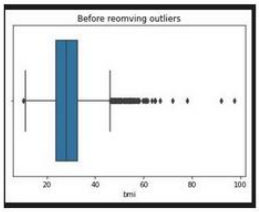
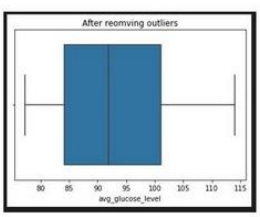


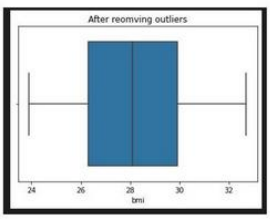
Fig5: Prediction Page**.**

Before : Before, Outliers are observations that are significantly different from the majority of the data and can distort statistical analyses .









After: After detecting the outlier, it have high dimensional data. the outliers may be important and informative for the analysis.

CONCLUSION:

In this project, we have developed a user friendly application called Detecting outlier from high dimensional data using Machine Learning Model techniques such as SVM, KNN and Decision tree. We used the best techniques we found and its show the weather it is STROKE are not STROKE.

REFFERENCES

[1] Charu C. Aggarwal and Philip S. Yu. Outlier Detectionfor High Dimensional Data. In Proc. SIGMOD Conf.,2001.

[2] K. A. De Jong. Analysis of the Behavior of a Class ofGenetic Adaptive Systems. Ph. D. Dissertation, University of Michigan, Ann Arbor, MI, 1975.

[3] D. E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, Reading, MA, 1989.

[4] A. K. Jain, M. N. Murty, and P. J. Flynn. Data Clustering: a Review. ACM Comp. Surveys, 31(3):264-

323,1999.

[5] E. M. Knorr and R. T. Ng. Algorithms for MiningDistance-Based Outliers in Large Datasets. In Proc.VLDB, pp. 392-403, 1998.

[6] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft.When is Nearest Neighbors Meaningful?. In Proc. of theInt. Conf. Database Theories, pp. 217-235, 1999.

[7] http://www.ics.uci.edu/˜ mlearn/MLRepository.html

[8] M. M. Breunig, H. P. Kriegel, R. T. Ng, and J. Sander.LOF: Identifying Density-Based Local Outliers. In Proc.

SIGMOD Conf., pp. 93-104, 2000.

[9] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C.Faloutsos. LOCI: Fast Outlier Detection Using the LocalCorrelation Integral. In Proc. ICDE, pp. 315-326, 2003.

[10] T. Johnson, I. Kwok, and R. T. Ng. Fast Computation of 2-Dimensional Depth Contours. In Proc. KDD,pp. 224-228, 1998.

[11] V. Barnett and T. Lewis. Outliers in Statistical Data. John Wiley and Sons, 1994.

[12] C. Zhu, H. Kitagawa, S. Papadimitriou, and C. Faloutsos. OBE: Outlier by Example. In Proc. PAKDD. pp. 222-234, 2004