Outlier Detection using Supervised Machine Learning Algorithms

(A comparison of different algorithms used to detect outliers and the various tradeoffs)

Shrey Mishra, Neil Sharma, Rangan Basu

Abstract—Outlier detection finds numerous uses and applications in different industries such as fraud detection, structural defect analysis, detecting mislabeled data and incursion detection. An outlier is a pattern which is different to other remaining patterns in the data set. This paper examines the different algorithms which can be used to detect outliers in a dataset, the principles governing its viability and the trade-offs between various methods. The paper also reviews the recent developments in the field and the future scope of research.

Keywords: Linear Regression; Logistic Regression; Neural Networks; Support Vector Machines, Anomaly detection

I. INTRODUCTION

Outlier detecting systems have numerous real-world applications such as fraud detection in credit cards and calling cards, discovering computer incursion and criminal behaviors in loan application processing, detecting network system instability, etc. Major progress has been made in the area of outlier detection in the last decade and hence finds many researchers in academia interested in pursuing new developments in the field. In recent years, outlier detection methods have found widespread adoption in the medical industry to predict outliers in the medical records associated with a patient. This helps doctors and medical researchers to predict the chances of a particular disease and often times to preempt the occurrence of such ailments by suggesting corrective action for the same. Outlier detection systems find numerous applications in logistics, public safety, and transport system along with many other uses.

Outlier detection systems have also entered the realms of the banking industry, especially in tracking fraud and risky loan application processing. Tracking fraud and having an effective intrusion detection system is a much difficult and elusive goal for various system administrators and information security researchers. The different methods proposed in this paper are often used in silos or in conjunction with one another to get the desired outputs.

Outliers can be of two types: univariate (single variable) and multivariate (many variables). Univariate outliers can be found while looking at a diffusion of values in a singular feature space. Multivariate outliers can be found in an n-dimensional space of n-features.

While looking at distributions in an n-dimensional spaces can be very difficult and taxing to do manually, therefore we need to train a model to flag it for us. Outliers can be also classified into different segments, depending upon their environment: point outliers, contextual outliers, or collective outliers. Point outliers are respective data points that lies far away from the rest of the distribution. Contextual outliers can be regarded as noise in data, such as punctuation motifs when trying to analyze text data or noisy signal in the background at the time of performing speech recognition.

Over the years, various machine learning algorithms have been explored for the design of an Outlier detection system such as – linear regression, logistic regression, neural networks, and support vector machines with various kernels. Each of these algorithms has different success rates in detecting outliers in a dataset depending on the type of the data set and also on the nature of the outliers. Also discussed are the advantages and disadvantages of the various algorithms and the scope of integration of other techniques to improve the efficiency of current systems. The content listed below is organized as follows. An overview of machine learning algorithms is presented in Section II. In Section III, we discuss the different datasets and the parameters to judge the accuracy of results presented in Section IV and Section V proposes the future scope of research.

II. OUTLIER DETECTION ALGORITHMS AND TECHNIQUES OF OUTLIER ESTIMATION

As defined by Hawkins in 1980, an outlier is – "(an) observation which deviates so much from other observations as to arouse suspicion it was generated by a different mechanism." In other words, an outlier is an observation that diverges from an overall pattern on a sample. In the process of acquiring, gathering, processing and gaining insights from data, outliers may come from numerous sources and hide in multiple dimensions. Outliers which are not a product of an error are called novelties.

Identifying outliers is of prime importance in almost any quantitative disciplines like Physics, Economy, Finance, Machine Learning, and Cyber Security. In machine learning and in other quantitative domains the quality of data is equally important as the quality of the outlier detection algorithm or model.

Some of the most widely used outlier estimation techniques and algorithms namely regression, neural networks, support vector machines are discussed below in brief and how it can be implemented on outlier detection task. These techniques can be

used in silos or can be used in conjunction for a more holistic solution.

A. Linear Regression

In supervised learning, we are given a data set (X) and we already know what the correct output (y) should look like and also knowing that there exists a relationship between the input features and the output label.

Supervised learning problems are categorized into "regression" and "classification" problems. In a regression problem, we try to match results which take on a continuous range of values, implying that we are attempting to map out the input variables to some existing continuous function, however, for the implementation of regression algorithm on a classification task, we will assume a threshold i.e. 1.5 times the value predicted by the hypothesis to be set as a threshold value indicates anything above this threshold will be classified as an outlier point.

In a classification problem, we try to predict results for a discrete output. In precise words, we are now trying to map input variables into discrete categories or labels.

Our hypothesis function $h_\theta(x)$ has the general form for linear:

$y\hat{} = h_\theta(x) = \theta_0 * x_0 + \theta_1 * x_1 + \theta_2 * x_2 + ... + \theta_n * x_n$

for n number of dimensions in the data

[Note: that the outliers detected are subject to the change in the threshold predicted by the linear hypothesis, even though it might lead to more number of false positives and false negatives]

Example:

Input(x)	Output(y)
0	4
1	7
2	7
3	8

Suppose we have the following set of training data:
Now we can make a random guess about our hypothesis (h_θ) function: $\theta_0=2$ and $\theta_1=2$. The hypothesis function becomes $h_\theta(x)=2+2x$.

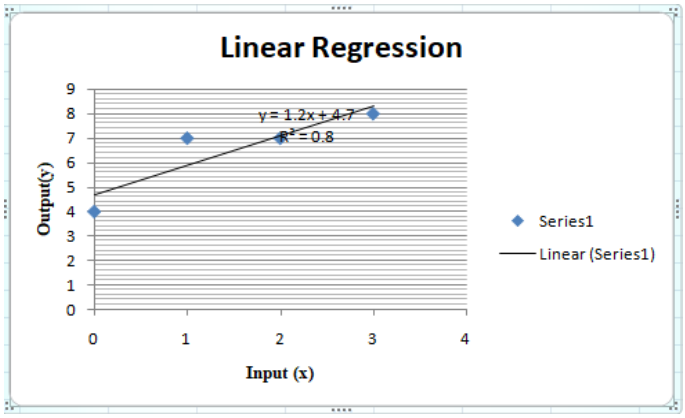

Figure1.1- Linear Regression hypothesis plot example with optimum θ parameters

So, for input of 1 to our hypothesis, y value corresponding to it will be 4. This is separated by 3. We will be trying out various values of θ_0 and θ_1 to try to find values which provide the best possible "fit" or the most expressive "straight line" through most of the data points mapped on the x-y plane as shown in figure 1.1

Cost Function (J): -

We can measure the accuracy of our hypothesis function by using a cost function. This takes an average (actually a least means square approach) of all the results of the hypothesis with inputs compared to the actual output.

$J(\theta_0, \theta_1, ... \Theta_n) = (1/2*m) (\sum_1^m (y\hat{} - yi)\hat{}2$, where

$y\hat{} = h_\theta(x) = \theta_0 * x_0 + \theta_1 * x_1 + \theta_2 * x_2 + ... + \theta_n * x_n$

(For n dimensions)

y_i= Actual label of the data
m= Total number of rows/samples

Our objective is to minimize this cost function in order to obtain the best fit line (hypothesis). We can minimize this function with an approach known as gradient descent as can be seen in Fig3.

Gradient Descent:

We will now see how well our hypothesis fits into the data by our cost function (J). Now we need to estimate the parameters in hypothesis function. This is where we will use gradient descent. To visualize the concept better, we will now put θ_0 on the x-axis and θ_1 on the y-axis, with the cost function on the vertical z-axis, all the points on our graph will be the outcome of the cost function using our hypothesis with those specific theta parameters we try to look for, making it a convex shaped function similar to Fig1.2

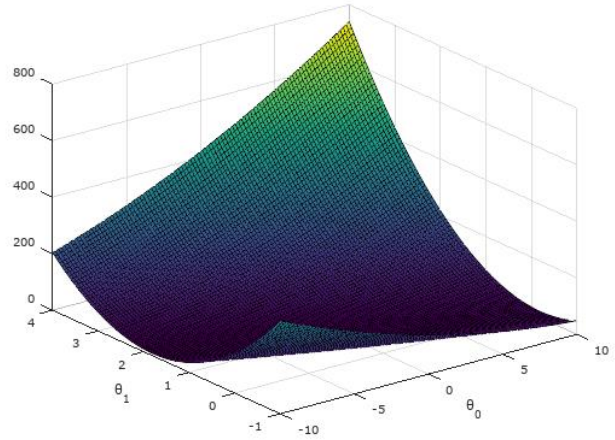

Figure1.2- The plot shows the varying Cost (J) is a convex function

When our cost function reaches the very bottom of the pits depicted in our graph, then we would have achieved our objective. When the value reaches the minimum, then we have optimized our function or in other words, the theta parameters depict the global minima.

One way of finding out the optimal theta parameters is by differentiating the cost equation. The derivative also indicates the slope of the tangential line at any chosen point on the cost function and this in-turn indicates the direction we need to move towards in order to obtain the minima. Learning rate or α is the parameter used to determine the size of each step taken in the direction of steepest descent. Figure 1.4 illustrates this example.

Setting α to too high a value might cause the algorithm to overshoot the point instead of converging at the global minima [2]. This process is repeated iteratively such that the convergence graph is set straight or narrowed out in the end, as is depicted in the figure below for vowels.

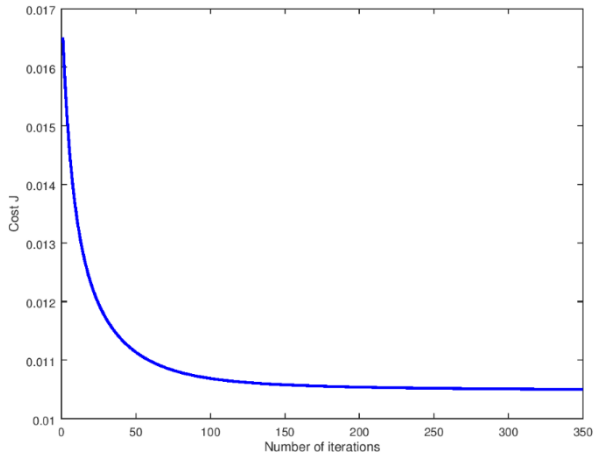

Fig1.3. Converging of Cost (J) after as iterations progress (dataset – vowels.mat)

For linear regression problems, applying gradient descent to minimize the cost function entails updating the cost iteratively while progressing steadily towards the global minimum. The weights (θ) are updated as:

$$\theta_j = \theta_j - \alpha \left(\frac{\partial}{\partial \theta j}\right) * J(\theta_0, \theta_1)$$

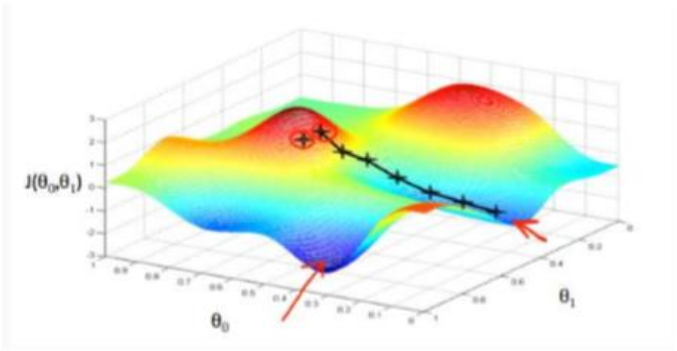

Fig1.4. Gradient descent

The weight (θ) can also be estimated by another common approach known as the Normal Equation method.

Normal Equation Method:

In the normal equation method, optimum weight (θ) is estimated without having to perform any optimizations on the cost function. The optimum value is given by the equation:

$$\Theta = (X^{T*}X)^{-1} X^T y$$

In practice, estimating optimal weights by the Normal equation method almost always performs poorly in comparison to the gradient descent method, as is re-enforced by the results section of our paper. Generally, gradient descent is preferred to Normal equation method as time complexity of calculating the inverse in the normal equation method is to the order of O (n^3). Gradient descent on the other hand has the time complexity of O (n^2). However, the tradeoff is the absence of learning rate and iterations required in the gradient descent method [3].

B. Logistic Regression

Logistic regression is applied to classify a discrete range of values. In a binary classification problem, the target values are either of two values. In all our datasets, y assumes one of two values '0' denoting an inlier and '1' denoting an outlier. Hence, y∈{0,1}. Zero is sometimes referred to as the negative group or class and similarly 'One' is referred to as the positive group or class [4].

If x (i) is known to us, the corresponding y (i) would be our labels in case of the training example.

We can infer that $h_\theta(x)$ will not take values larger than 1 or lesser than 0 if $y \in \{0, 1\}$. Hence, we modify $h_\theta(x)$ such that it now satisfies the below range.

$0 \leq h_\theta(x) \leq 1$

This is done by inserting $\theta^T x$ in the Logistic Function which is commonly referred to as the sigmoid function g(z).

Where,

$$h_\theta(x) = g(\theta^T x)$$
$$z = \theta^T x$$
$$g(z) = 1/(1+e^{-z})$$

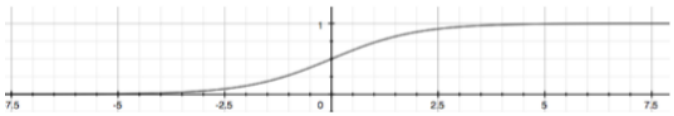

Fig2.1. Sigmoid Function

We use a different cost function from the one used in linear regression because the Logistic Function causes the output to be wavy. Such a wave-like nature implies multiple local optimum values thus causing our optimization algorithm to get stuck in the local minima. In other words, such a function will not be a convex function [5].

Thus, the cost function (J) in case of logistic regression looks likes:

$$\text{Cost}(h_\theta(x), y) = -y * \log(h_\theta(x)) + (1-y) * \log(1-h_\theta(x))$$

Inputting (y=0 or y=1) in the respective parts causes that part to be nullified. The generic cost equation is represented as:

$$J(\theta) = (-1/m) * \sum_{i=1}^{m} [y^{(i)} \log(h_\theta(x^{(i)})) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)})) + \lambda/2m \sum_{j=1}^{m} Q_j^2]$$

The regularization parameter, λ, depicted in the above equation makes the hypothesis generic and less prone to over-fitting. [6]

To classify our dataset, we now apply gradient descent algorithm and minimize the cost function.

C. Artificial Neural Networks

Neural networks basically consist of three layers-

1. Input layer– As the name suggests, this layer is used to feed our network with the required inputs in (i.e. dimensions and features) our data. We feed the network with every row sample present in our data set on every epoch.

2. Hidden layers– **All** layers present in between the input layer and output layer are classified into this category. Generally, multiple hidden layers help in improving the accuracy of the final prediction. For our model, we employ 2 hidden layers and each such layer consists of half the number of nodes present in the input layer.

A generic way of deciding the number of nodes present in the hidden layer can be given by the formulae below:

$$N_h = N_s / (\alpha * (N_i + N_o))$$

N_i = Number of input neurons
N_o = Number of output neurons
N_s = Number of samples in training data set
α = An arbitrary scaling factor usually 2-10

Considering the low sample size of some of the training data sets and in order gain a greater efficiency (Hit rate), we will stick with the first approach where the number of nodes in the hidden layers are always equal to half the value of nodes in the network's input layer and all our results are based on the same approach.

3. Output layer- The output layer contains the number of nodes we want to classify the various outliers of our data set in. In our case, we have two nodes with output {0, 1} for inlier point and {1, 0} for an outlier.

Particular impetus must be given to the fact of initializing the random weights with a value near to but not equal to zero, the network's next layer computes to be the weighted sum of the previous layer and is followed by the activation function. For our purpose, we have employed the Sigmoid activation used previously in logistic regression see figure 3.1. [7]

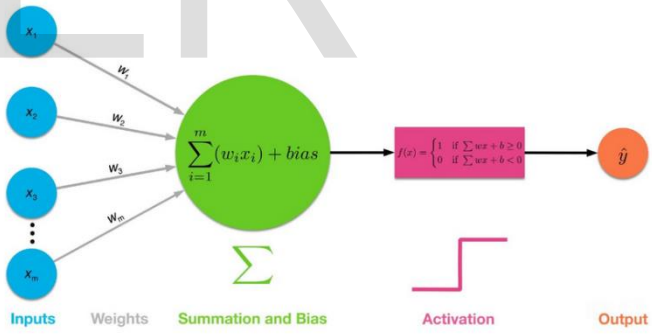

Fig3.1. Forward Activation

A bias term equal to 1 is also added before feeding the output to the next layer.

A weight optimizing algorithm together with back-propagation is employed to tune the random weights. To calculate errors in our prediction, back-propagation compares the output against the random weights to the desired labels for the specific data row.

The error in the last layer is denoted by $\delta^{(l)}$. We calculate the error in each hidden layer upon moving backward towards the

first hidden layer for $\delta^{(L-1)}$, $\delta^{(L-2)}$,…… $\delta^{(2)}$, using a generic formula defined as

$\delta^{(l)} = ((\theta^{(l)})^T\delta^{(L-1)}).*a^{(l)}.*(1-a^{(l)})$
Here $a^{(l)}$ define the vector of outputs for l, th layer. [7]

The generic cost function (J) for neural networks, is defined as:

$J(\theta)=(-1/m)*[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)}*\log(h_\theta(x^{(i)}))_k+(1-y^{(i)}_k)*\log(1-h_\theta(x^{(i)}))_k]+(\lambda/2m)*\sum_{l=1}^{L-1}\sum_{i=1}^{sl}\sum_{j=1}^{sl+1}(\theta_{ji}^{(l)})^2$

Also, $\Delta_{i,j}^{(l)} = \Delta_{i,j}^{(l)} + a_j^{(l)}\delta_i^{(l+1)}$

In the subsequent step, we calculate partial derivatives of the cost function with respect to current weights denoted as $D_{ij}^{(l)}$ lth layer.

$D_{ij}^{(l)} = \frac{\partial}{\partial\theta ij} * J(\theta)$

The $D_{ij}^{(l)}$ for every layer can also be defined as:

$D_{i,j}^{(l)} = (\frac{1}{m})\ (\Delta_{i,j}^{(l)}+ \lambda\theta_{i,j}^{(l)})$,if j is not zero. (with regularization)

$D_{i,j}^{(l)} = (\frac{1}{m})\Delta_{i,j}^{(l)}$ if j=0 (without regularization)

We will now update the weights assigned with the hidden layers by subtracting the partial derivative term (D_{ij}) and then feeding weights and cost to an advanced optimization algorithm as previously explained in the gradient descent approach.

With every epoch, optimization algorithm will take steps in the direction of the global minima calculating the optimal weight parameters required to classify an anomaly.

Once the weights are trained on all the samples, the result is a fine-tuned set of parameters which can be further be tested on the unseen remaining (30%) of the test data as demonstrated in the results section.

D. Support Vector Machines

SVM is a perceptron-like neural network and is basically used for binary classification of patterns that can be linearly separated by a decision boundary. A perceptron-solution, however, is different because a number of possible hyper-planes can be made between the two classes. The points which are at the minimal distance from the optimal hyper-plane are called as "support vectors." Classifiers that make use of this attribute are hence called as support vector machines. [9]

The elementary idea of Support Vector Machines is to generate a hyper-plane which can maximize the separation between the margins belonging to the zero (normal) and the one (anomaly) classes. [10] One of the favourable feature of the SVM is that it is a rough enactment of the Structural Risk Minimization principle, which according to Wikipedia is, based on analytical learning theory rather than the Empirical Risk Minimization method, in which the classification function is borrowed by minimizing the Mean Square Error over the training data set. One of the major suppositions of SVM is that all samples are individually and equitably, at the same time, distributed in the training set. [9]

Several alterations and improvements are made since the introduction of the original idea: hard-margin SVMs for differentiable cases, soft-margin SVMs that are used for cases that are non-separable and robust SVMs that are used to exhibit great generalization characteristics while handling data which is noisy and mislabeled. [11]

If we consider our training samples to be (x1, y1), · · · , (xl, yl), yi∈ {0, 1}, i = 1, ..., l where {(xi , yi)} i = 1, ..., l are feature vectors and yi∈ {0, 1} , i = 1, ..., l are the corresponding labels, the zero class represents normal behavior and one class represents incongruous behavior. Therefore, the classification problem can now be mannered as a constrained optimization problem. [1] It would be ethical if, in a given a set used for training, we somehow managed to find a boundary, also known as a decision boundary, that allow us to predict all the training examples that are correct and confident, meaning far from the decision boundary.

The cost function (J) for Support Vector Machine is detailed below:

$\text{Min } (C\sum_{i=0}^{m}[y^{(i)}\text{cost}_1(\theta^Tx^{(i)})+(1-y^{(i)})\text{cost}_0(\theta^Tx^{(i)})]+(1/2)\sum_{i=q}^{n}\theta_j^2)$

In the above equation, 'C' is a positive value that is used for controlling the penalty for training examples that are not classified correctly. A large C instructs the SVM to try and classify all the examples correctly. C plays a role identical to 1/λ, where λ is the regularization parameter that we were using previously for logistic regression.

The effect of C with different values is demonstrated in the figures illustrated with numbers Fig4.1 – Fig4.3 on Vowels dataset.

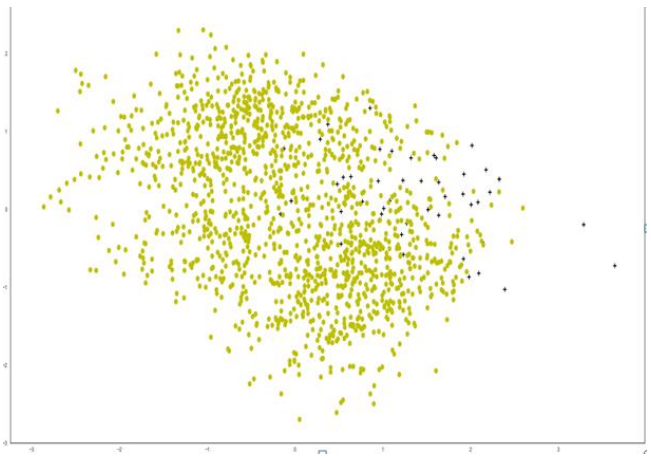

Fig4.1 Vowels Dataset which can be separated by a linear boundary

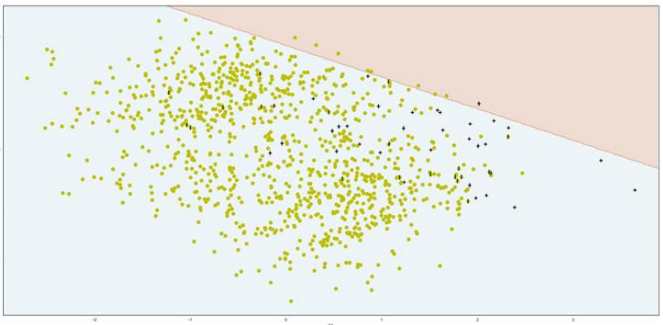

Fig4.2 (a) SVM decision boundary with C=1

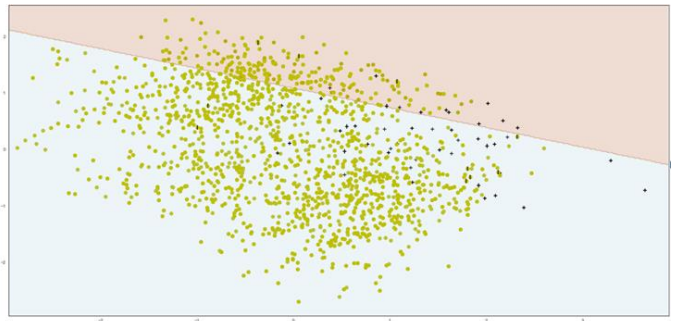

Fig4.2 (b) SVM decision boundary with C=150

If the data-set is not linearly separable or we want to implement a nonlinear boundary curve, then we use the Gaussian kernel to find out the decision boundaries.

Using Gaussian kernels

The Gaussian kernel can be understood as an analogous function which is used for measuring the "distance" between a combination of examples, (x(i), x(j)). Furthermore, the Gaussian kernel is parameterized by a bandwidth parameter, σ, which is used to determine the rapidness of the decline of the similarity metric to 0 as the examples are moved further apart from each other.
The Gaussian Kernel function is defined as:

$K_{gaussian}(x^{(i)},x^{(j)}) = \exp\left(- \|x^{(i)} - x^{(j)}\|^2/2\sigma^2\right)$
$= \exp\left(- \sum_{k=1}^{n}(x^{(i)}{}_k x^{(j)}{}_k)^2/2\sigma^2\right)$

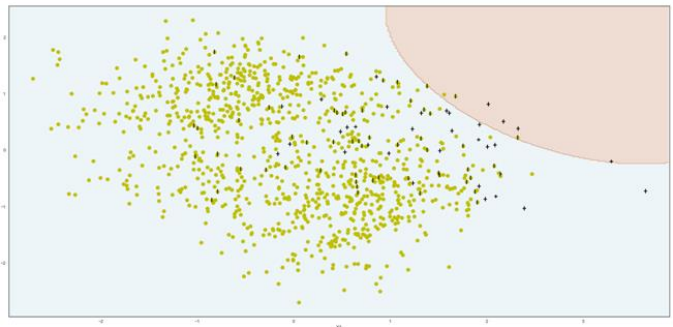

Fig4.3 (a) SVM decision boundary with C=1 after applying Gaussian Kernel

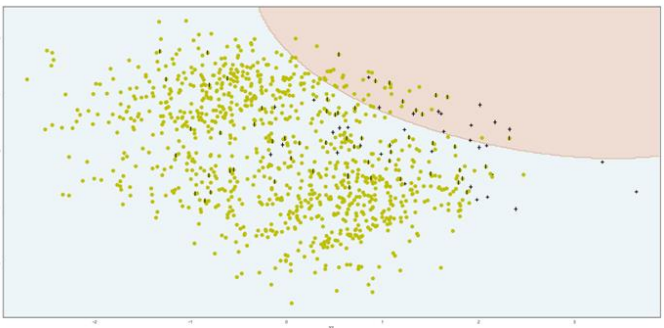

Fig 4.3 (b) SVM decision boundary with C=150 after applying Gaussian Kernel

The implementation of SVM with kernels such as Gaussian or polynomial could result in learning a better nonlinear complex boundary curve and in turn yielding higher accuracy but often comes at the cost of extra computational power required.

III. DATA SETS AND OUTLIER DETECTION EFFICIENCY METRICS

The application of the algorithms detailed above to a chosen set of datasets and their results are discussed below. The different data sets on which the machine learning algorithms are applied are discussed in brief. For all data sets being tested, for y-label = (1 – for Outliers, 0 – for Inliers) here is a small description about the first 5 datasets tested:

1. Vowels.mat – It is a multivariate time series data, where nine male speakers proclaimed two Japanese vowels /ae/ successively. It is a classification dataset to separate the speakers. For outlier detection, every row in the data used for training is considered as a separated data point. [C. C. Aggarwal and S. Sathe, "Theoretical foundations and algorithms for outlier ensembles." ACM SIGKDD Explorations Newsletter, vol. 17, no. 1, pp. 24–47, 2015]

2. Lympho.mat - It is a multi-class dataset having four classes, but two of them are quite small (2 and 4 data records). Therefore, those two small classes are merged and considered as outliers compared to other two large classes (81 and 61 data records). [Lazarevic and V. Kumar, "Feature bagging for outlier detection." in ACM SIGKDD, 2005, pp. 157–166]

3. Letter.mat – Capital letters of the English alphabet represented in 16 dimensions. To get data suitable for outlier

detection, we subsample data from 3 letters to form the normal class and randomly concatenate pairs of them so that their dimensionality doubles. To form the outlier class, we randomly select few instances of letters that are not in the normal class and concatenate them with instances from the normal class. [https://www.inf.ethz.ch/personal/mcbrian/pdfs/odd.pdf]

4. Glass.mat – This dataset contains attributes regarding several glass types (multi-class). Here, class 6 is a clear minority class; as such points of class 6 are marked as outliers, while all other points are inliers. [Keller, E. Muller, K. Bohm."HiCS: High-contrast subspaces for density-based outlier ranking." ICDE, 2012]

5. Vertebral.mat - The data set is characterized by six biomechanical attributes derived from the shape and orientation of the pelvis and lumbar spine (in this order): pelvic incidence, pelvic tilt, lumbar lordosis angle, sacral slope, pelvic radius and grade of spondylolisthesis. The following convention is used for the class labels: Normal (NO) and Abnormal (AB). Here, "AB" is the majority class having 210 instances which are used as inliers and "NO" is rescaled from 100 to 30 instances as an outlier class. [Saket Sathe and Charu C. Aggarwal. LODES: Local Density meets Spectral Outlier Detection. SIAM Conference on Data Mining, 2016]

We look at some of the performance measure metrics in brief.

A. *Hit Rate*

Hit rate is the percentage of successful predictions to the total number of attempts. A higher hit rate implies better performance of our algorithm in detecting outliers.

B. *Miss Rate*

Miss rate is the percentage of unsuccessful predictions to the total number of attempts. Higher miss rate implies poor performance of our algorithm in detecting outliers.

C. *False Positives*

False positives are test results which falsely characterize that a particular state or a specific attribute is present. For Example, if our algorithm classifies an inlier point as an outlier, then such a result is a false positive.

D. *False Negatives*

False negatives are results where our algorithm misclassifies the outlier to be an inlier.

IV. RESULTS OF OUTLIER DETECTION ALGORITHMS

In this section, we showcase an overview of the results after applying the different outlier detection algorithms to our datasets. We then discuss the challenges we faced and particular parameters if any and justify our choice for the same.

A. *Linear Regression*

The Linear regression algorithm is applied to the five datasets, and the efficiency metrics are calculated on the results. Minimization of the cost function is done using the gradient descent and later compared with the normal equation method. As demonstrated in the results below, applying gradient descent to estimate the optimal θ parameter always outperforms the normal equation method.

The Gradient descent approach always proves over normal equations as explained in the table below:

Datasets	Hit Rate	False Negatives	False Positives
Vowels	91.140	0	129
Lympho	89.864	0	15
Letter	89.810	0	163
Glass	88.780	0	24
Vertebral	90.010	0	24

Table1. Linear (Gradient Descent) Results

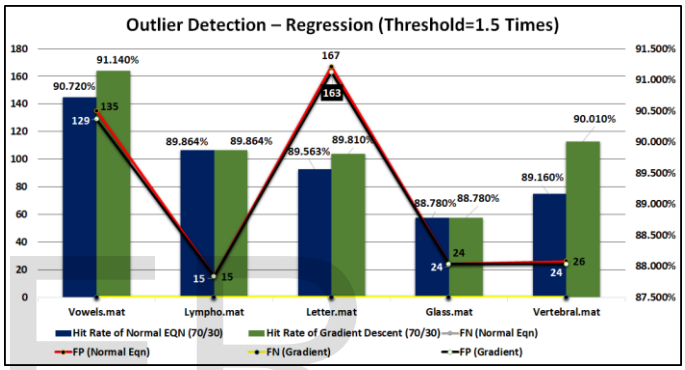

Fig5. (a-1) Linear regression based on a threshold

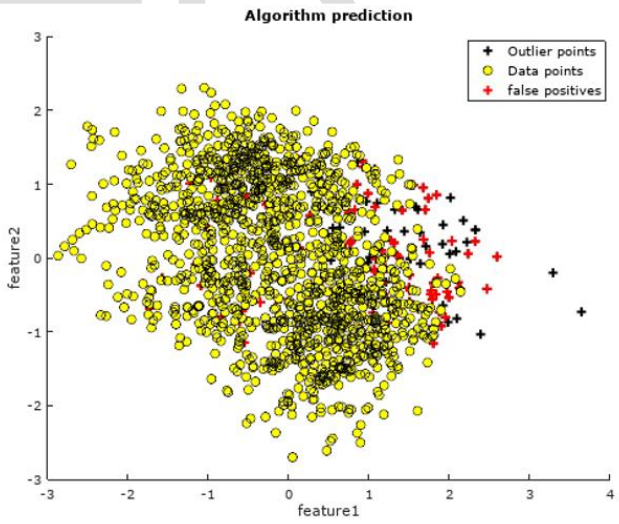

Fig5. (a-2) Linear regression depicting false positives

B. *Logistic Regression*

The logistic regression algorithms are applied on the data sets after having regularized (a technique that applies to objective functions in ill-posed optimization problems) the parameters as well.

Two approaches have been utilized to test the accuracy of our prediction when the lambda parameter is set to 0. The two approaches are detailed in brief.

70-30 Train-Test Approach: We split our dataset into 2 parts. The first part is called training set (contains randomly shuffled 70% of the data with at least 70% of the outliers) and the second part is called testing data (contains randomly shuffled 30% of the data with at least 30% of the outliers)

Cross-validation Approach: We split our data into 5 parts (making sure that every part has at a minimum (20%) of the outlier data to train on) and then we compare each part with all the remaining parts. This approach makes sure that each section contributes to both training and testing.

Logistic regression with cross-validation:

Datasets	Hit Rate	False Negatives	False Positives
Vowels	98.351	1	0
Lympho	97.950	0	0
Letter	91.938	24	2
Glass	94.419	1	0
Vertebral	90.000	5	5

Table 2. Logistic regression (Cross-validation, λ=0) Results

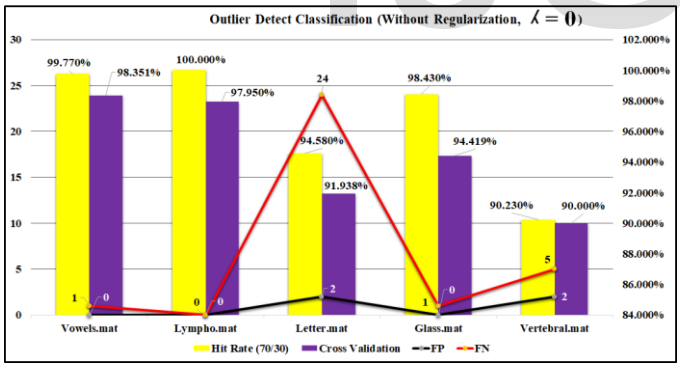

Fig 5. (b) Logistic regression with no regularization parameter

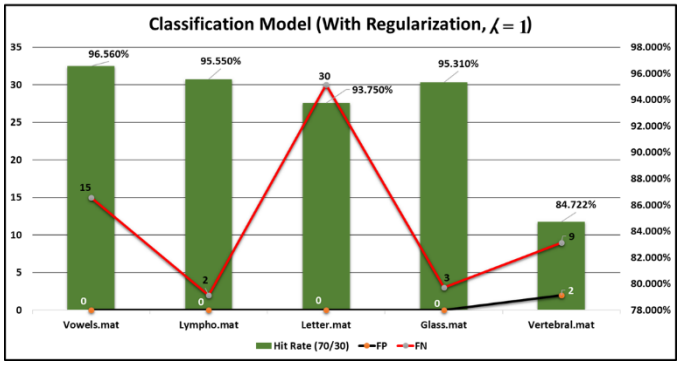

Fig5. (c) Logistic regression with regularization parameter

C. *Neural Networks*

For neural networks, we see that the results when computed with optimal regularization parameters (λ=0.5 for anomaly detection) and a fixed number of nodes in each of the 2 hidden layers. Low accuracies in some cases are due to the severely limited size of the dataset along with the architectural design of ANN used. The results of applying the neural network to the five datasets are detailed below:

Datasets	Hit Rate	False Negatives	False Positives
Vowels	99.77	1	0
Lympho	97.78	1	0
Letter	94.79	22	3
Glass	98.46	2	0
Vertebral	88.11	10	0

Table.3. ANN Accuracy Results

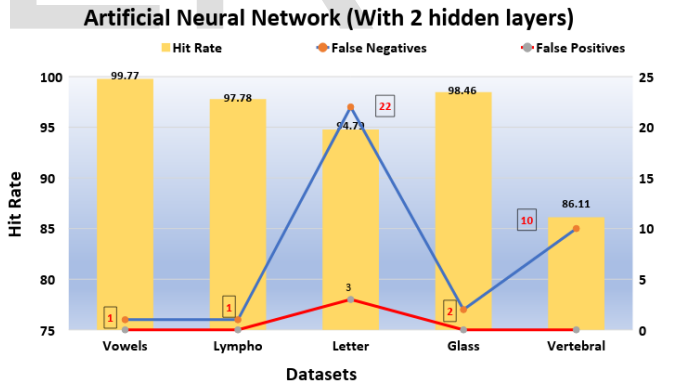

Fig5. (d) ANN with 2 hidden layers

D. *Support Vector Machines*

The Support Vector Machines outperformed all other supervised learning algorithms when it comes to the Hit-rate of our model. As it can be seen in the results below, in our case, applying Linear Kernel and then drawing the decision boundary was able to classify more number of outliers and inliers correctly. The results are detailed below.

Datasets	Hit Rate	False Negatives	False Positives
Vowels	100	0	0
Lympho	100	0	0
Letter	93.94	27	0
Glass	98.46	0	1
Vertebral	84.72	6	5

Table4. SVM Accuracy Results

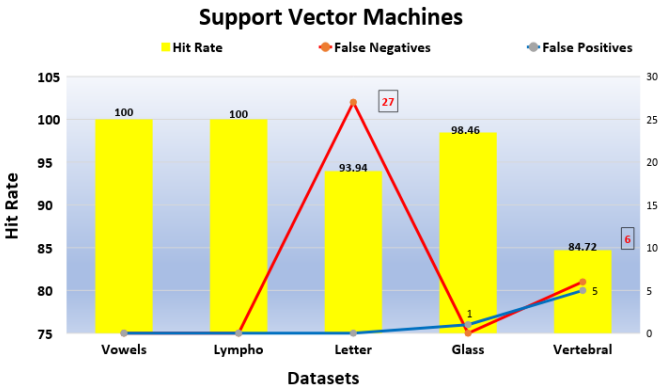

Fig5. (e) SVM Results with linear kernel

Datasets	Hit Rate	False Negatives	False Positives
Vowels	100	0	0
Lympho	97.7	1	0
Letter	93.94	27	0
Glass	98.46	0	1
Vertebral	86.11	10	0

Table5. SVM Accuracy with Gaussian Results

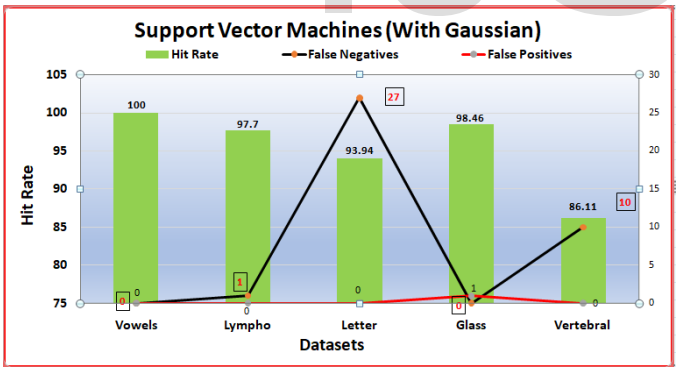

Fig5. (f) SVM Result with Gaussian kernel

The linear approach had previously flagged many false positives which are ideally not suitable for anomaly detection approach and therefore we will now test our algorithms on more diverse datasets listed below:

V. CONCLUSION AND FUTURE SCOPE

In the comparison of all the approaches used for testing on all of the anomaly detection datasets, the SVM's, in general, has offered the highest accuracy for most of the low dimensional data. In most cases, its time complexity is far too low when compared with the Artificial Neural Networks.

The choice of detection algorithm heavily depends on the dataset itself as seen when dealing with digits or speech data sets or in general the datasets with a large number of columns having many samples, the ANN approach has outperformed the SVM's prediction.

The logistic regression approach fails to compete with SVM's and ANN's, as seen for most of the datasets and hence would not be a suitable choice for the task.

With improvements in the field, there are many other hybrid models involving the convolutional neural networks and the state of the art capsule networks could possibly yield better results on the task of anomaly detection.

Apart from all the supervised learning approaches a lot of unsupervised learning algorithms such as the k means clustering and Self-organizing maps can be tested and can prove to be more efficient.

REFERENCES

Datasets tested	Rows(m)	Column (n)	SVM (Without Gaussian)	SVM (With Gaussian)	ANN	Logistic Regression	Anomaly %
Vowels	1456	12	100	100	99.77	98.351	3.4
Lympho	148	18	100	97.77	97.78	97.95	4.1
Letter	1600	32	93.94	93.94	94.79	91.938	6.25
Glass	214	9	98.46	98.46	98.46	94.419	9
Vertebral	240	6	84.72	86.11	88.11	90	12.5
Mammography	11183	6	98.36	98.59	98.68	98.42	2.32
Smtp	95156	3	99.98	99.98	99.96	99.98	0.03
Annthyroid	7200	6	94.4	94.44	92.12	94.7	7.42
Thyroid	3772	6	99.38	99.02	97.73	99.04	2.5
Pima	768	8	78.78	63.2	71.86	78.16	35
Wbc	278	30	99.12	99.12	95.61	95.47	5.6
Ionosphere	351	33	88.67	89.62	85.84	83.71	36
Optdigits	5216	64	99.36	98.4	99.87	N/A	3
Satellite	6435	36	N/A	68.41	88.67	87.956	32
Satimage	5803	36	99.54	98.85	96.56	99.58	1.2
Speech	3686	400	97.1	98.28	99.88	96.98	1.65
Breast	683	9	96.09	96.58	96.09	95.18	35
Arrhythmia	452	274	85.29	83.82	91.18	84.66	15

[1] Priya Stephen ; Suresh Jaganathan,“ Linear regression for pattern recognition” International Conference on Green Computing Communication and Electrical Engineering, 16 October 2014.

[2] Kavitha S; Varuna S; Ramya R,“ A comparative analysis on linear regression and support vector regression” Online International Conference on Green Engineering and Technologies, 04 May 2017.

[3] Hai Wang ; Fei Hao “An efficient linear regression classifier” IEEE International Conference on Signal Processing, Computing and Control, 25 June 2012.

[4] Sanizah Ahmad, Norazan Mohamed Ramli, Habshah Midi “Outlier detection in logistic regression and its application in medical data analysis” IEEE Colloquium on Humanities, Science and Engineering, May 27, 2013.

[5] Abdul Nurunnabi, Geoff West “Outlier Detection in Logistic Regression: A Quest for Reliable Knowledge from Predictive Modeling and Classification” IEEE 12th International Conference on Data Mining Workshops, Dec 10, 2012.

[6] Cuixia Gao, Zhitang Li, Lin Chen “Host Anomalies Detection Using Logistic Regression Modeling” First International Workshop on Education Technology and Computer Science, 26 May 2009

[7] Andrew Ng CS229: Backpropagation, Bishop auditorium, Stanford University, USA.

[8] Shuxiang Xu, Ling Chen “A Novel Approach for Determining the Optimal Number of Hidden Layer Neurons for FNN's and Its Applications n DataMining” 5th International Conference on Information Technology and Applications (ICITA 2008).

[9] Wenjie Hu, Yihua Liao, V. Rao Vemuri “Robust Anomaly Detection Using Support Vector Machines" CiteSeer, In Proceedings of the International Conference on Machine Learning, June 2003.

[10] V. N. Vapnik. Statistical Learning Theory. John Wiley & Sons, Inc. New.

[11] Andrew Ng CS229: *Support Vector Machines*, Bishop Auditorium, Stanford University, Stanford, USA.