**Machine learning algorithms:**

**Classification**

We applied 5 different supervised machine learning models to classify the data set of 5280 records with 62 attributes: 61 predictor and 1 response variables. To avoid over fitting, we used a 10-fold cross validation, i.e. the data set is partitioned into 10 folds and the accuracy in each is estimated. This precludes the possibility that the system learns from the noise and random fluctuations of data and consequently would not be able to generalize and handle new data. The Matlab™ Statistics and Machine Learning Toolboxes are used for data classification. The results of the classification using the 5 models are shown in Table 3. A brief description of each model is shown below. For a complete description of each model, the interested reader is referred to the machine learning documents found in the the Mathworks® website (Mathworks, 2017).

The **Boosted Trees Ensemble** model creates an ensemble of medium decision trees using the machine learning AdaBoost algorithm.

The **Complex Decision Trees** model uses decision trees to classify the input data set based on the selected response.

The **k-nearest neighbour (KNN)** classification model classifies a data object by a majority vote of its neighbours according to its k nearest neighbours.

The **Fine Gaussian Support Vector Machine (SVM)** classifies data by finding the best hyper-plane that separates one class of data from those of the other classes. Essentially, it uses the same concept as the linear SVM but with finely detailed distinctions between classes by using a kernel scale of where P is the number of predictors.

The **Linear** Discriminant model classifies data by estimating the parameters of a Gaussian distribution for each class. The new data is classified by finding the smallest misclassification cost.

**Evaluation of Results and Observations**

We compared the classification models on 10 distinct features: accuracy, training time, quantity of data correctly identified as abnormal (TP), quantity of data correctly identified as normal (TN), quantity of data incorrectly identified as abnormal (FP), quantity of data incorrectly identified as normal (FN), recall, precision, specificity, and informedness. The data used for calculating the last 4 features are extracted from the confusion matrices. A confusion matrix for the Boosted Tree Ensemble classifier is shown in Figure 6. The following formulae are used to calculate the last 4 features.

Since we are more interested in identifying abnormalities, i.e. attacks, rather than normalities, we assume an abnormal event as positive and a normal event as negative. Hence True Positive (TP) indicates the number of abnormal events that are correctly identified and True Negative (TN) are those events that are correctly identified as normal.

In this study, ***recall*** describes the completeness of identifying abnormal events, while ***precision*** describes the quality of classification. Simply stated, a ***high recall*** means most of the abnormal events have been identified; a ***high precision*** means abnormal events are correctly classified with very minimal misclassification. ***Specificity***, in this study,refers to the measure of identifying normalcy. Thus, a **high specificity** indicates a high rate of classifying normal events. Finally, Powers defines ***informedness*** as a measure of how informed a predictor is for a specific condition (Powers, 2011). Figure 7 depicts a performance comparison of the five machine learning classifiers.

**Using Machine learning and Statistics tool box:**

**Boosted Tree Ensemble classifier:**

Accuracy: 99.2%

Prediction speed: 17000 obs/sec

Training time: 7.6354 sec

**Model Type:**

Preset Boosted Trees

Ensemble method: AdaBoost

Learner type: Decision Tree

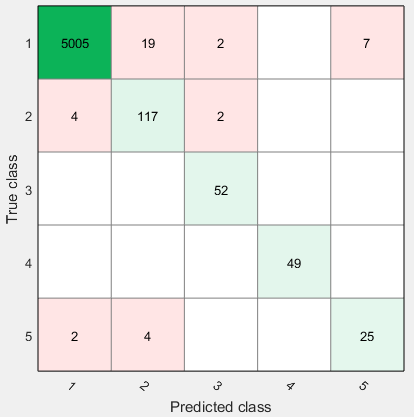
Max no of splits: 20

No of learners: 30

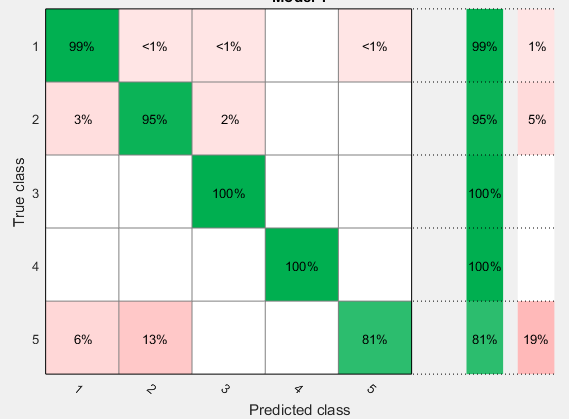
Learning rate: 0.1

Feature Selection: all features used in the model, before PCA.

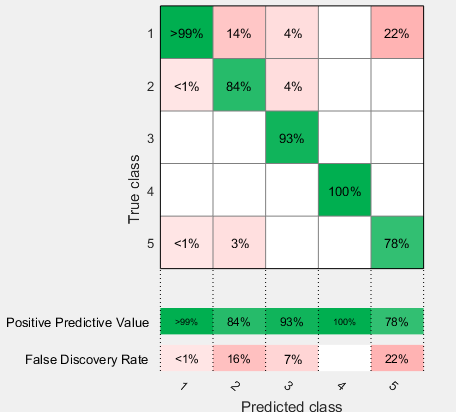
**Confusion Matrix plot for number of observations:**



**Confusion matrix plot for True Positive Rates and False Negative rates:**

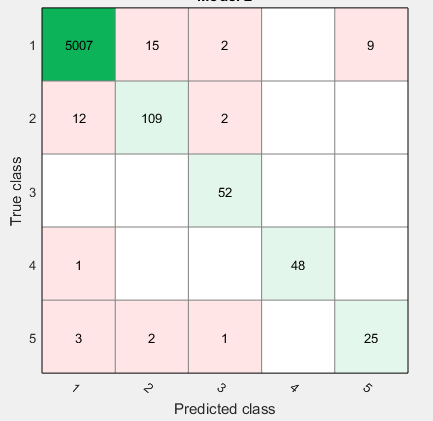


**Confusion Matrix plot for positive predictive values and false discovery rates:**



**Complex Tree:**

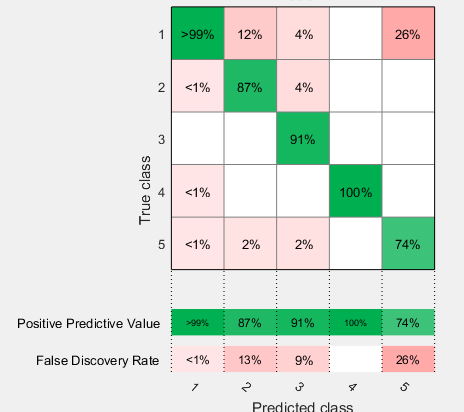
Confusion Matrix for no of observations:



Confusion matrix for true positive rates and false positive rates:

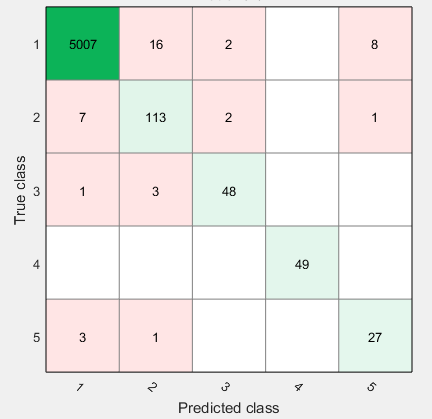


Confusion Matrix plot for positive predictive values and false discovery rates:

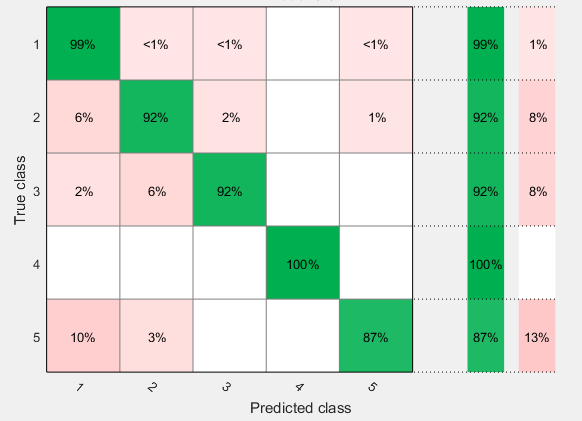


Weighted KNN:

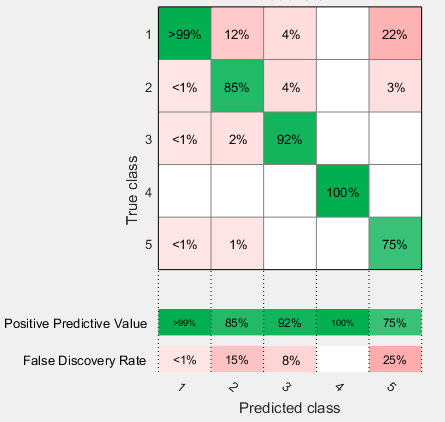
Confusion Matrix for no of observations:



Confusion matrix for true positive rates and false positive rates:

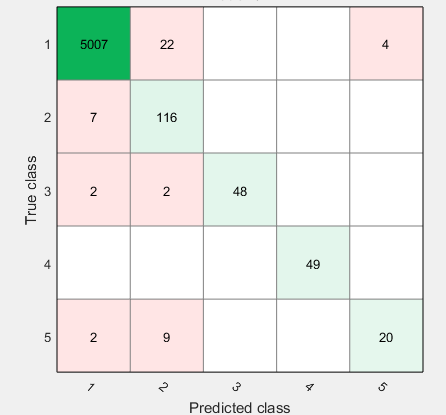


Confusion Matrix plot for positive predictive values and false discovery rates:

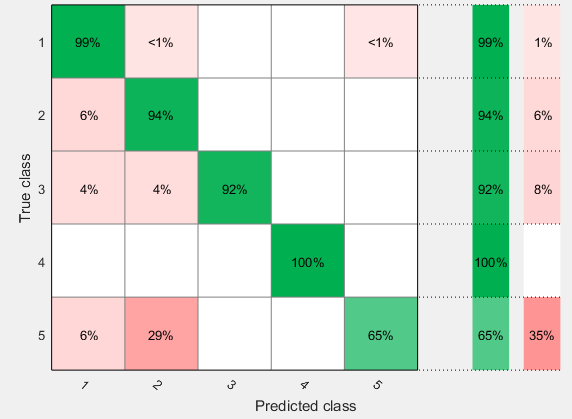


Quadratic SVM:

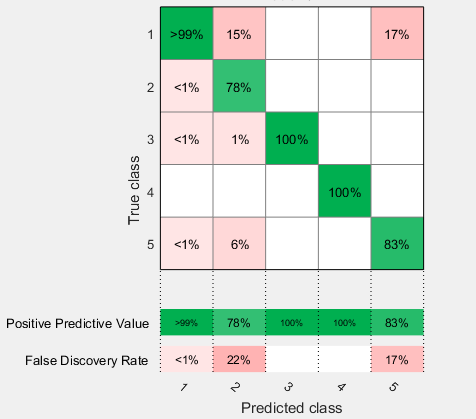
Confusion Matrix for no of observations:



Confusion matrix for true positive rates and false positive rates:

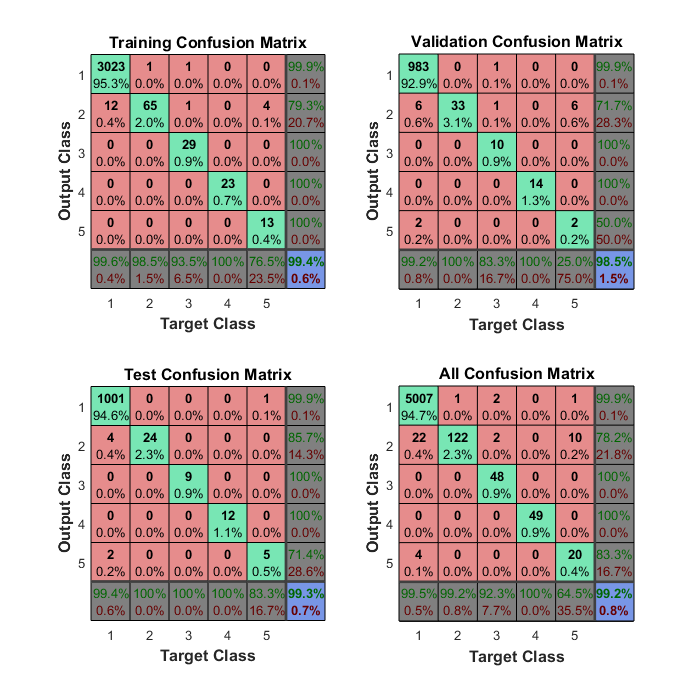


Confusion Matrix plot for positive predictive values and false discovery rates:



**Neural Network Tool Box:**

Neural Network Scaled Conjugate Backpropagation Algorithm:



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Classifier Model** | **Accuracy (%)** | **Training Time (sec)** | **TP** | **TN** | **FP** | **FN** | **Recall (%)** | **Precision (%)** | **Specificity (%)** | **Informedness (%)** |
| ***Boosted Trees Ensemble*** | 99.20 | 7.64 | 243 | 5005 | 6 | 28 | 89.67 | 97.59 | 99.88 | 89.55 |
| ***Complex Decision Trees*** | 99.1 | 1.33 | 234 | 5007 | 16 | 26 | 90.00 | 93.60 | 99.68 | 89.68 |
| ***Weighted KNN*** | 99.2 | 2.18 | 237 | 5007 | 11 | 26 | 90.11 | 95.56 | 99.78 | 89.89 |
| ***Quadratic Support Vector Machine (SVM)*** | 99.1 | 6.33 | 233 | 5007 | 11 | 26 | 89.96 | 95.49 | 99.78 | 89.74 |
| ***Conjugate gradient backpropagation*** | 99.2 | - | 239 | 5007 | 26 | 4 | 98.35 | 90.19 | 99.48 | 97.83 |

Literature Review:

## 2.1 Machine Learning:

The MathWorks Team (MathWorksTeam, 2016) described two techniques of machine learning: Supervised and Unsupervised. While supervised learning uses known input and output data to generate a predictive model by using classification and regression techniques, the unsupervised learning type draws inferences by discovering underlying patterns in the input data by using clustering technique.

2.2 Machine Learning model notation: (Andrew)

= the column vector of all the feature inputs of the ith training example.

= value of feature j in the ith training example.

= target output that we are trying to predict.

= training example

m = number of training examples

n = number of features

X = space of input features

Y = space of output values

h (hypothesis) = for given training set, to learn a function h: X->Y so that h(X) is a good predictor for corresponding value Y.

J (cost function) = measure of the accuracy of proposed hypothesis (the difference between predicted and actual value).

= model parameter to define h.

α = learning rate and constant lies between 0 and 1.

= regularization parameter, controls the cost function from overfitting () or underfitting ()

***Regression*:** When the target variable that we are trying to predict is continuous, the learning problem known as a regression problem.

***Classification*:** When the target variable can take only a few number of discrete values, called it as a classification problem.

2.3 Gradient descent algorithm:(Andrew) The cost function indicates how well the data fits our defined hypothesis. To minimize the cost function error, estimating the parameters necessary. Gradient descent algorithms help to estimate and minimize cost function. Also, known as batch gradient descent algorithm.

***Algorithm:***

repeat until converges:

Where j = represents the feature index number, 0 to n.

= learning rate. If too small, convergence slow; if too large, cost function not decrease or may not converge.

By setting our input values roughly in the same range can run gradient descent algorithm efficiently. The reason is will descend quickly on small ranges and slowly on large ranges, and so will oscillate inefficiently down to optimum when the feature variable values are uneven. Feature scaling and mean normalization techniques might helpful.

***Feature scaling:*** It involves dividing the input values by the range (i.e. the maximum value minus the minimum value) of the input variable, resulting in a new range of just 1. The ideal ranges are -1 ≤ ≤ 1 or -0.5 ≤ ≤ 0.5

***Mean normalization:*** It involves subtracting the average value for an input variable from the values of that input variable resulting in a new average value for the input variable of just zero.

By implementing these two techniques, can adjust the input variable with the following formula:

= average of all values of feature i.

= range of values (max-min) or standard deviation.

2.4 Linear regression algorithm**:** (Andrew) It tries to fit the data in the region of a straight line. It is effective if the training data has non-polynomial features. The disadvantage is computationally expensive for polynomial features.

2.4.1 For one feature variable***:***

Hypothesis:

Cost function:

Gradient descent:

Repeat until convergence: {

}

2.4.2 For multiple features***:***

=

Repeat until convergence: {

}

2.5 Logistic regression algorithm**:** (Andrew) It is used to classify examples into negative and positive classes based on decision boundary. For example, if ≥ 0.5, then places into positive class otherwise, places into negative class. It can be applicable to single class or multiple classes classifier. The function of this algorithm is nonlinear.

Hypothesis or logistic function or sigmoid function:

***Cost function:***

***Gradient descent:*** repeat: {

}

2.6 Neural networks**:** (Andrew) It is another model to train data in a nonlinear fashion. An effective algorithm for classifying data into multiple classes. The feature sometimes called as bias unit and it uses the same logistic function which is used in logistic regression, sometimes called it as the activation function. The parameters also called as weights. The following is the simple representation of this model.

Usually, the function model is represented in the form node layers. The input feature nodes treated as input layer and the layer at which hypothesis function outputs called it as output layers. The intermediate layers between the input layer and output layer termed as hidden layers and its node called it as activation units.

***Notation:***

= “activation” of unit i in layer j.

= matrix of weights controlling function mapping from layer j to layer j+1.

L = total number of layers in the network

Sl = number of units (excluding bias unit) in layer l

K = number of output units/classes

Note: if network has units in layer j and units in j+1 layer, then will be of dimension

Activation function:

2.6.1 Forward propagation**:**

where temporary matrix and = x (input feature matrix)

**Cost function:**

(the i in triple sum does not refer to training example i)

2.6.2 Backpropagation**:** It’s a neural network terminology and the functionality is similar to gradient checking, i.e. to minimize the cost function.

***Algorithm:***

Given training set { , , ……… }

* Set for all (l,i,j)
* for (training example) i = 1 to m {
  + set
  + perform forward propagation to compute
  + using , compute
  + compute using

}

* if j=0

2.6.3 The best approach to training neural network**:**

1. First, pick a network architecture; choose the layout of the neural network by determining no. of input and output units, no. of hidden units per layer, and total no. of layers to include in the network.
2. After picking a neural network, randomly initialize the weights.
3. Implement forward propagation to get hypothesis for each feature
4. Implement the cost function
5. Implement back propagation to compute partial derivatives or minimize the cost function.
6. Use gradient checking to confirm that back propagation working correctly, then disable gradient checking.
7. Use gradient descent or built-in optimization function to minimize the cost function with the weights in theta.

2.7 Support Vector Machine (SVM)**:** (Andrew) A robust algorithm, tries to separate data or classifies data with a large margin. Mainly uses with various kernels. Kernels draws new features from x based on landmarks (l(i)) with similarity function (similarity (x, ) = where ).

***Hypothesis***:

= 1 if

= 0 otherwise

***Cost function***:

If y=1 (wants )

If y=0 (wants )

where C is regularized parameter (reciprocal of and act as decision boundary.

2.7.1 SVM with kernels (Gaussian Kernel)**:**

Choose landmarks, , ,……., from given training examples, , , ………, then compute new features(..,) using similarity function. With these new features, predict y=1 if and compute cost function

2.8 K-means clustering algorithm**:** (Andrew) This algorithm falls under unsupervised learning algorithm where output labels or variables are unknown. It groups the given data in the form of clusters. Market segmentation, social network analysis, organize computing clusters, and astronomical data analysis are some applications of this algorithm.

***Notation:***

= index of cluster (1,2,3, …, K) to which example is currently assigned.

= cluster centroid k

= cluster centroid of cluster to which example has been assigned.

***Algorithm:***

Input: K (no. of clusters), Training set {}

Randomly initialize K cluster centroids .

Repeat {

for i =1 to m

= index (from 1 to K) of cluster centroid closest to

for k = 1 to K

= average (mean) of points assigned to cluster k

}

***Optimization objective:***

2.9 Anomaly detection algorithm**:** (Andrew) It is used for density estimation or finding an anomalous sample from given data set by modeling P(x) from data. It helps to raise a flag when abnormal behavior occurs. Fraud detection, testing manufacturing equipment’s, monitoring computers in a data center are the major applications of this algorithm. It uses the concept of Gaussian distribution.

***Algorithm:***

* Choose features that might be indicative of anomalous examples.
* Fit parameters
* Given new example, compute p(x):
  + =
* Raise anomaly if p(x) <

Using F1 score metric can evaluate the algorithm accuracy on cross-validation and test set data.

2.9.1 Anomaly detection vs. Supervised learning:

|  |  |
| --- | --- |
| **Anomaly detection** | **Supervised learning** |
| Appropriate if few number of anomalous examples available or large number of non-anomalous examples available.  Effective if a high number of anomaly types present; if future anomalies look nothing and have a small number of anomalous examples, modeling with non-anomalous examples is appropriate.  Ex: Fraud detection, manufacturing, monitoring machines in a data center, etc. | Appropriate if a large number of anomalous and non-anomalous examples exist.  Requires enough positive examples to get a sense of what positive examples like, future examples likely to be similar to the training set.  Ex: Email spam classification, weather prediction, cancer classification, etc. |