**Sameed Ahmed Khan**

**Tools for Artificial Neural Network and Machine Learning**

**Problem Statement 1:**

Explore, download and submit a report on five ANN tools. Perform experiments using WEKA tool with three classification algorithms on three different datasets. One of the classification algorithm/technique must be ANN (Multilayer Perceptron). Present all your results in a compact form (tables and graphs). Small, medium, and large-scale datasets should be used. The Experimenter interface (rather than the Explorer interface) of WEKA may be a more efficient way of performing experiments. Read about WEKA interfaces and implementation on the web tutorial to gain better understanding.

**WEKA:**

Slight Introduction:

**Waikato Environment for Knowledge Analysis –** WEKA is a free software licensed under GNU GPL.

It has two modes Explorer and Experimenter and allows a variety of tasks to be done.

Getting Started:

We can download WEKA from <http://www.cs.waikato.ac.nz/ml/weka/downloading.html>**. After the installation, we are good to go.**

**How It Works:**

**We can download and install a variety of tools including features selectors, classifiers, predictive analysis tools from the WEKA tool box.**

**Reading the Understanding the Data:**

**The data can be imported in arff format. The WEKA reads the columns of data which serves as attributes, and those as classes. On the basis of them, as soon as the data is read it shows it features like number of classes present, the distribution, and the attributes involved in determining that very feature.**

**In explorer mode, under visualize tab we have the visualizations of the data, showing clusters and much more.**

**Operations:**

**We can perform the operations like Classification, clustering and association. WEKA also enable to get access most of the parameter’s selection for the above operations.**

**Orange:**

Slight Introduction:

Orange- A software package for Data Mining, Visualizations and Analysis, is an open-source software package with components in C++ and wrappers in Python.

It has:

* Canvas – GUI for Analysis of Data
* Widgets – It offers operations like Reading and Visualizing Data, selecting features, training, comparing learning algorithms and other operations.

It is widely used by people belonging to domains of Bioinformatics and other related fields.

Getting Started:

We can download orange from [**https://orange.biolab.si**](https://orange.biolab.si/)**. After the installation, we set a directory for Orange to store the files. And then we are good to go. It has a very appealing and interactive front-end interface so we won’t find it any difficult to search for tools.**

**How It Works:**

**We can train model by importing the data file, cleaning the data, getting a classifier and building its model. Much of the process is like what we have done in WEKA.**

**Cleaning and Visualizing:**

**The data can be imported and the structure can be visualized using the widgets.**

**Using the Impute Widget we can clean our messed-up data.**

**For understanding the data, we have several plots like scatter plot, distribution plot, sieve diagram etc.**

**Rapid Miner:**

Slight Introduction:

**It is a tool for Data Science with an integrated environment for Data preparation, cleansing, visualizations, analysis. It also supports Machine Learning and Deep Learning and is mostly used for research and educational purposes.**

It comes with an interactive GUI which makes the task quite easy on the user end.

Getting Started:

We can start with 30-day trial of RapidMiner Studio Enterprise after which we will be automatically reverted to free version or we can apply for Educational License Program.

**Getting the Data Ready:**

We can import the data using the read excel operator and can be further on prepared by the ETL tool Turbo Prep which can save a lot of time required for Data Preparation.

We can play with data by appending and pivoting the data if necessary. We can also connect the databases that can be reused as templates and shared.

Modelling:

We can design several Machine Learning Models just like ion previous tools.

The Models can be created, applied, tested and validated on the same platform.

We can also optimize the model by parameter tuning or we can leave it to automatic optimizer.

**Keras:**

Slight Introduction:

**Keras is a Deep Learning and Neural Network Library in Python that allows designing and testing the models based on Neural Networks.**

Getting Started:

We can import Keras directly in python or through Tensorflow.Keras module.

**Getting the Data Ready:**

Keras has variety of methods to read the data. We can either import the Whole data and divide on some percentage as training and test data.

We can also get the Training, Testing data through separate parameters or we can flow it through some directory. The function will automatically identify the number of classes present in the directory referred.

Modelling:

We can design the ANN through model.sequence module. We can then go on defining the layers and its nodes. It is widely used for Deep Learning so we have multiple layers like Dropout, Dense, Conv2D, LSTM, RNN and multiple other layers which we can use to design the model. It is a vast library and includes lot of features to be used.

Summary and Plots:

After we have designed the model we can view its structure and features using model.summary () method.

After training the data on the designed model we can plot the training accuracy, testing accuracy and other evaluations history dictionary returned on model.fit () call.

**MATLAB:**

Slight Introduction:

**MATLAB is a** multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages.

Getting Started:

We need to buy the MATLAB from <https://www.mathworks.com/products/matlab.html> or get the student’s version.

Modelling:

We can design a Multi-Layer Perceptron using the Neural Network toolbox in MATLAB. With the Deep Network Designer app, we can design, analyze, and train networks graphically. We can also keep track of training parameters and analyze results. We can visualize layer activations and graphically monitor training progress.

**Problem Statement 2:**

Write a report on classification of Road/Traffic Sign Detection using the links below: You can use more material from the web. Provide details of the datasets, comparative analysis of algorithms, implementation details, results and analysis.

**Algorithms:**

Multi-Layer Perceptron (MLP):

A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN). The term MLP is utilized questionably, now and again freely to allude to any feedforward ANN, now and again carefully to allude to systems made out of various layers of perceptron. Multilayer perceptron are some of the time informally alluded to as "vanilla" neural network, particularly when they have a solitary covered up layer.

MLP comprises of at any rate three layers of nodes: an input layer, a hidden layer and an output layer. Aside from the input nodes, every node is a neuron that utilizes a nonlinear actuation function. MLP uses a regulated learning strategy called backpropagation for training. Its various layers and non-direct enactment recognize MLP from a linear perceptron. It can recognize information that isn't linearly separable.

AdaBoost:

It works by selecting a training subset randomly and iteratively training the AdaBoost machine learning model by selecting the training set based on the accurate prediction of the last training. Then, it assigns the higher weight to wrong classified observations so that in the next iteration these observations will get the high probability for classification. Moreover, it assigns the weight to the trained classifier in each iteration according to the accuracy of the classifier. The more accurate classifier will get high weight.

This process iterates until the complete training data fits without any error or until reached to the specified maximum number of estimators. The algorithm having the highest votes then classifies.

Logistic Regression:

Multinomial logistic regression is also a classification algorithm same like the logistic regression for binary classification. Whereas in logistic regression for binary classification the classification task is to predict the target class which is of binary type. Like **Yes/NO, 0/1, Male/Female**.

When it comes to multinomial logistic regression. The idea is to use the logistic regression techniques to predict the target class (**more than** 2 target classes).

The underline technique will be same like the [logistic regression for binary classification](https://dataaspirant.com/2017/03/02/how-logistic-regression-model-works/) until calculating the probabilities for each target. Once the probabilities were calculated. We need to transfer them into **one hot encoding** and uses the cross-entropy methods in the training process for calculating the properly optimized weights.

**Experimental Setup:**

* We downloaded WEKA from <http://www.cs.waikato.ac.nz/ml/weka/downloading.html> and installed using the preferred method.
* We installed the Classification Tools required for the assignment from the Tools section of WEKA. The tools we used are MLP, AdaBoost and Logistic Regression and have been defined in above section.
* From the WEKA Explorer section, we loaded the datasets and performed classifications from the classify tabs.
* Throughout the steps, we have kept the training size to be 2/3rd of Dataset with 10 folds.

**Datasets Used:**

Keeping in mind the limited processing capabilities of my machine, I decided to keep the length short for the datasets however, a notable size difference is present between the datasets.

We used the following datasets for the given size:

**Small Scale Dataset:**

Iris dataset has been used as the small-scale dataset. The features of the dataset are as follows:

* Samples: The dataset consists of 150 total samples and 50 samples of each flower type.
* Outcomes: There are 3 outcomes: Iris Virginica, Iris Setosa, Iris Versicolor
* Features: The features are: Petal Length, Petal Length, Sepal Width, Sepal Length

**Medium Scale Dataset:**

Ionosphere dataset has been used as the medium-scale dataset. The features of the dataset are as follows:

* Samples: The dataset consists of 351 total samples.
* Outcomes: There are 2 outcomes: Good and Bad
* Features: We have 34 features in this dataset

**Large Scale Dataset:**

Soybean dataset has been used as the large-scale dataset. The features of the dataset are as follows:

* Samples: The dataset consists of 681 total samples.
* Outcomes: There are 19 outcomes: diaporthe-stem-canker, charcoal-rot, rhizoctonia-root-rot, phytophthora-rot, brown-stem-rot, powdery-mildew, downy-mildew, brown-spot, bacterial-blight, bacterial-pustule, purple-seed-stain, anthracnose, phyllosticta-leaf-spot, alternarialeaf-spot, frog-eye-leaf-spot, diaporthe-pod-&-stem-blight, cyst-nematode, 2-4-d-injury, herbicide-injury
* Features: We have 36 features in this dataset: date, plant-stand, precip, temp, hail, crop-hist, area-damaged, severity, seed-tmt, germination, plant-growth, leaves, leafspots-halo, leafspots-marg, leafspot-size, leaf-shread, leaf-malf, leaf-mild, stem, lodging, stem-cankers, canker-lesion, fruiting-bodies, external-decay, mycelium, int-discolor, sclerotia, fruit-pods, fruit-spots, seed, mold-growth, seed-discolor, seed-size, shriveling, roots

**Analysis of Results:**

**Iris Dataset:**

AdaBoost:

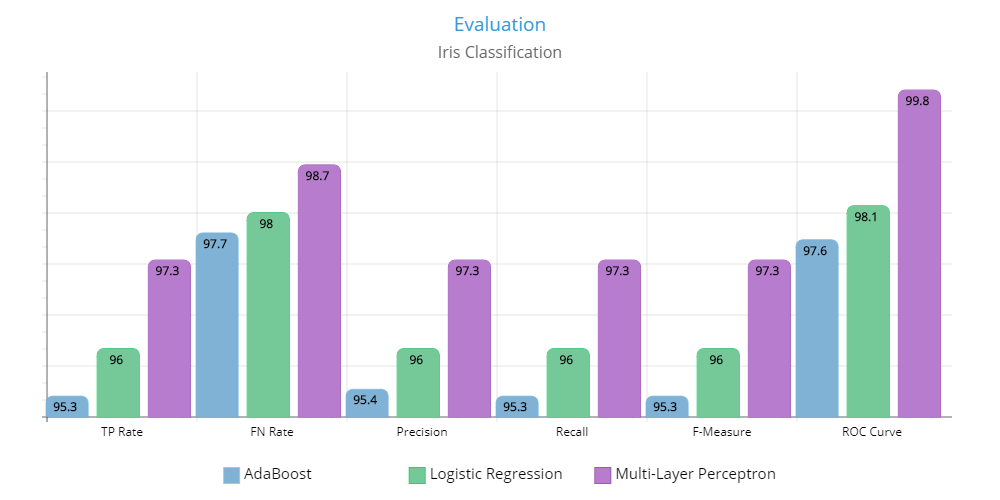
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Curve |
| Iris-setosa | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| Iris-versicolor | 0.900 | 0.020 | 0.957 | 0.900 | 0.928 | 0.963 |
| Iris-virginica | 0.960 | 0.050 | 0.906 | 0.960 | 0.932 | 0.965 |

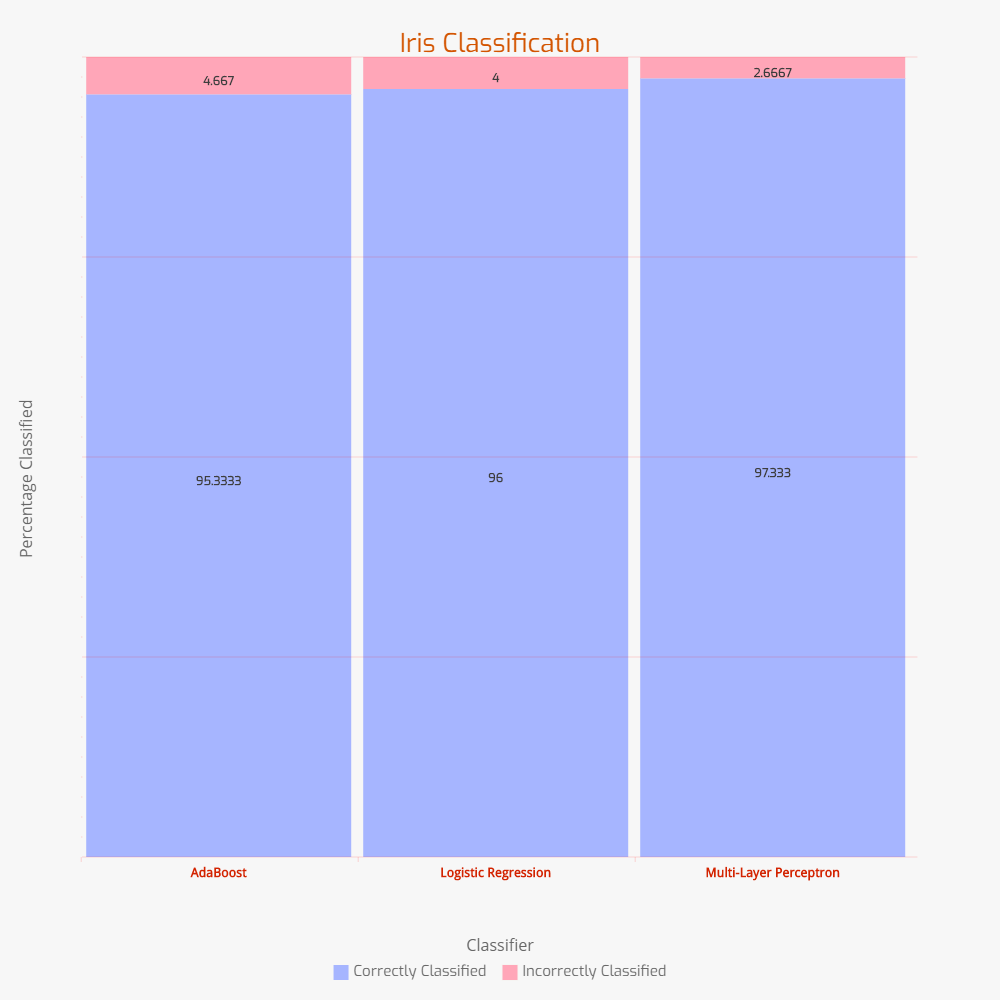
Logistic Regression:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Curve |
| Iris-setosa | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| Iris-versicolor | 0.920 | 0.020 | 0.958 | 0.920 | 0.939 | 0.972 |
| Iris-virginica | 0.960 | 0.040 | 0.923 | 0.960 | 0.941 | 0.972 |

Multi-Layer Perceptron:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Curve |
| Iris-setosa | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| Iris-versicolor | 0.960 | 0.020 | 0.960 | 0.960 | 0.960 | 0.996 |
| Iris-virginica | 0.960 | 0.020 | 0.960 | 0.960 | 0.960 | 0.996 |





**Ionosphere Dataset:**

AdaBoost:

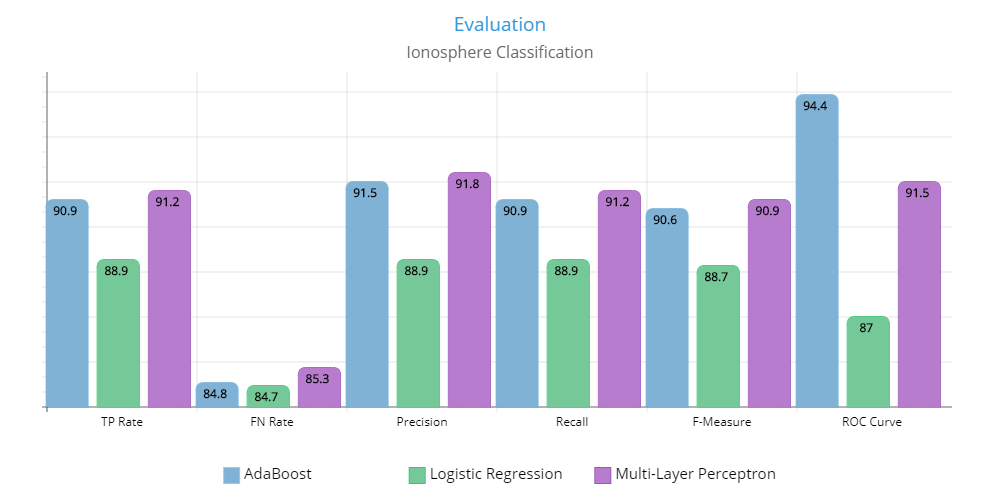
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Curve |
| Bad | 0.770 | 0.013 | 0.970 | 0.770 | 0.858 | 0.944 |
| Good | 0.987 | 0.230 | 0.884 | 0.987 | 0.933 | 0.944 |

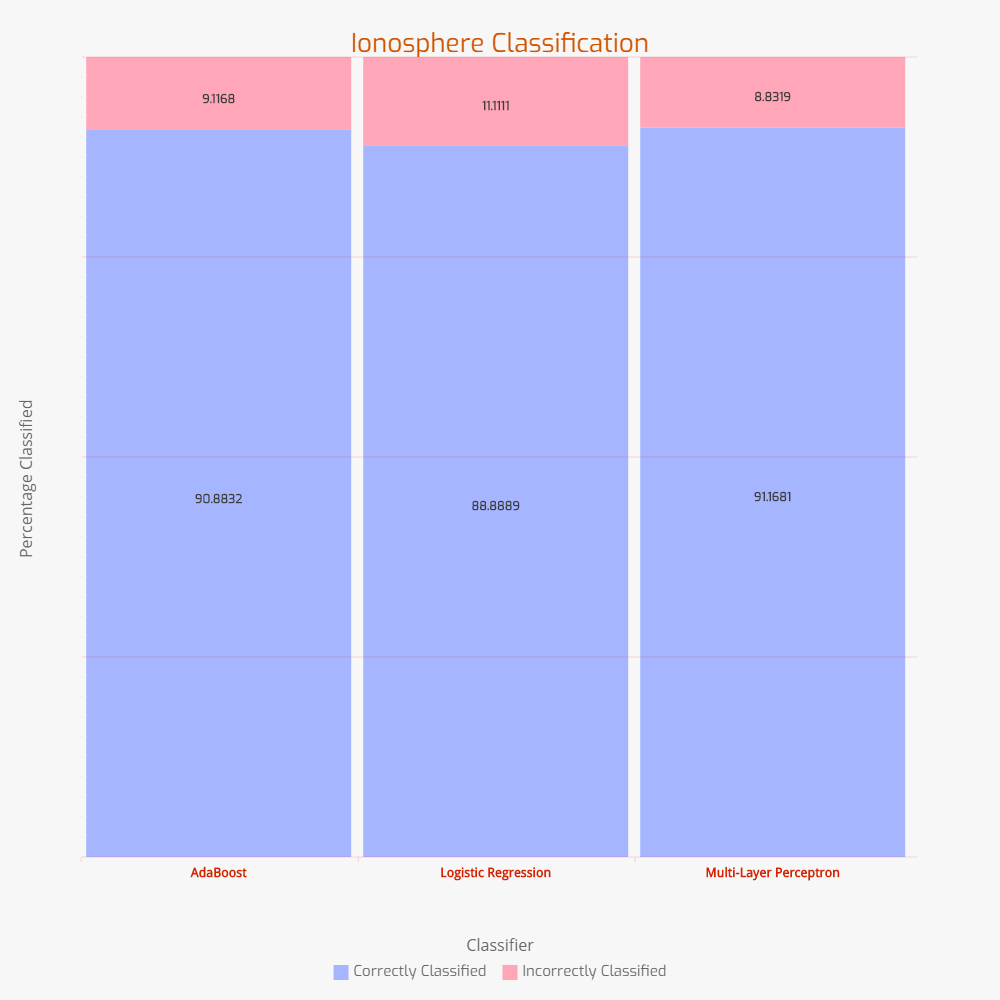
Logistic Regression:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Curve |
| Bad | 0.794 | 0.058 | 0.885 | 0.794 | 0.837 | 0.896 |
| Good | 0.942 | 0.206 | 0.891 | 0.942 | 0.916 | 0.870 |

Multi-Layer Perceptron:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Curve |
| Bad | 0.778 | 0.013 | 0.970 | 0.778 | 0.863 | 0.915 |
| Good | 0.987 | 0.222 | 0.888 | 0.987 | 0.935 | 0.915 |





**Soybean Dataset:**

Logit Boost:

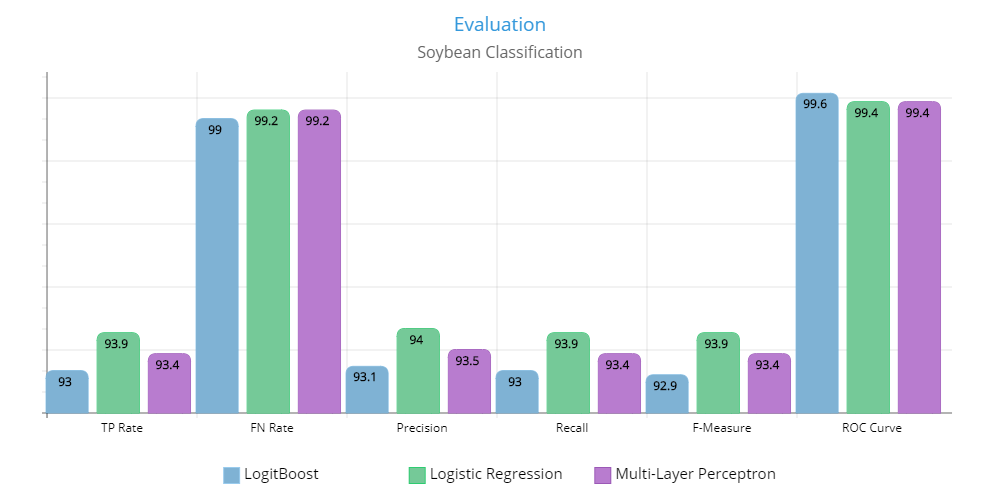
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Curve |
| diaporthe-stem-canker | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| charcoal-rot | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| rhizoctonia-root-rot | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| phytophthora-rot | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| brown-stem-rot | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| powdery-mildew | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| downy-mildew | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| brown-spot | 0.924 | 0.014 | 0.914 | 0.924 | 0.919 | 0.996 |
| bacterial-blight | 1.000 | 0.005 | 0.870 | 1.000 | 0.930 | 0.999 |
| bacterial-pustule | 0.85 | 0.000 | 1.000 | 0.85 | 0.919 | 1.00 |
| purple-seed-stain | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| anthracnose | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| phyllosticta-leaf-spot | 0.85 | 0.005 | 0.85 | 0.85 | 0.85 | 0.996 |
| alternarialeaf-spot | 0.89 | 0.037 | 0.786 | 0.89 | 0.835 | 0.990 |
| frog-eye-leaf-spot | 0.725 | 0.020 | 0.846 | 0.725 | 0.781 | 0.981 |
| diaporthe-pod-&-stem-blight | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| cyst-nematode | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| 2-4-d-injury | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| herbicide-injury | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |

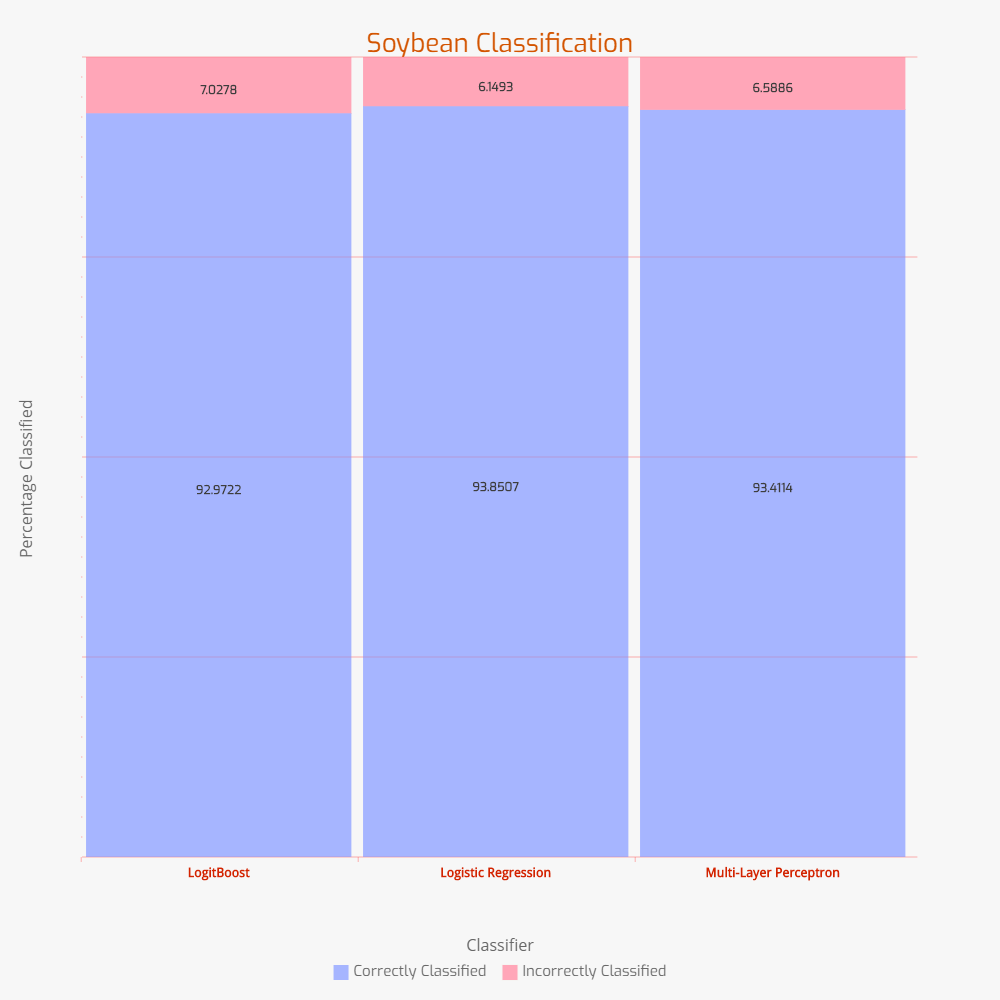
Logistic Regression:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Curve |
| diaporthe-stem-canker | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| charcoal-rot | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| rhizoctonia-root-rot | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| phytophthora-rot | 0.989 | 0.002 | 0.989 | 0.989 | 0.989 | 1.000 |
| brown-stem-rot | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| powdery-mildew | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| downy-mildew | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| brown-spot | 0.935 | 0.007 | 0.956 | 0.935 | 0.945 | 0.995 |
| bacterial-blight | 0.950 | 0.003 | 0.905 | 0.950 | 0.927 | 0.999 |
| bacterial-pustule | 0.900 | 0.000 | 1.000 | 0.900 | 0.947 | 0.999 |
| purple-seed-stain | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| anthracnose | 0.977 | 0.002 | 0.977 | 0.977 | 0.977 | 1.000 |
| phyllosticta-leaf-spot | 0.900 | 0.008 | 0.783 | 0.900 | 0.837 | 0.989 |
| alternarialeaf-spot | 0.868 | 0.027 | 0.832 | 0.868 | 0.849 | 0.986 |
| frog-eye-leaf-spot | 0.813 | 0.020 | 0.860 | 0.813 | 0.836 | 0.978 |
| diaporthe-pod-&-stem-blight | 1.000 | 0.001 | 0.938 | 1.000 | 0.968 | 0.999 |
| cyst-nematode | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| 2-4-d-injury | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| herbicide-injury | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |

Multi-Layer Perceptron:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Curve |
| diaporthe-stem-canker | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| charcoal-rot | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| rhizoctonia-root-rot | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| phytophthora-rot | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| brown-stem-rot | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| powdery-mildew | 1.000 | 0.000 | 1.000 | 1.000 | 0.976 | 1.000 |
| downy-mildew | 1.000 | 0.002 | 0.952 | 1.000 | 0.917 | 1.000 |
| brown-spot | 0.902 | 0.010 | 0.933 | 0.902 | 0.930 | 0.992 |
| bacterial-blight | 1.000 | 0.005 | 0.870 | 1.000 | 0.919 | 1.000 |
| bacterial-pustule | 0.850 | 0.000 | 1.000 | 0.850 | 1.000 | 0.999 |
| purple-seed-stain | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| anthracnose | 1.000 | 0.000 | 1.000 | 1.000 | 0.818 | 1.000 |
| phyllosticta-leaf-spot | 0.900 | 0.009 | 0.750 | 0.900 | 0.840 | 0.997 |
| alternarialeaf-spot | 0.835 | 0.024 | 0.844 | 0.835 | 0.829 | 0.987 |
| frog-eye-leaf-spot | 0.824 | 0.025 | 0.833 | 0.824 | 1.000 | 0.977 |
| diaporthe-pod-&-stem-blight | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| cyst-nematode | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| 2-4-d-injury | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| herbicide-injury | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 |





\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Problem 2 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Datasets:**

We have several datasets available for Traffic Sign Detection some of which include:

* German TSR Benchmark (GTSRB)
* KUL Belgium Traffic Signs Data set (KUL Data set)
* Swedish Traffic Signs Data set (STS Data set)
* RUG Traffic Sign Image Database (RUG Data set)

After thorough research, we found the German TSR Benchmark Dataset to be more applicable in our case.

**About German TSR Benchmark Dataset:**

Features:

* The Dataset has over 40 classes with unique real-world signs
* There are more than 50,000 instances
* Semi-automatic annotation has made it reliable ground-truth data

Structure:

The structure of the data that forms the training data is as follows:

* Each class has a directory
* Training images are grouped by tracks each of which contains 30 images of one physical sign

About Each Image:

* Image contains one object belonging to one class
* The image has portable pixmap format (PPM) and contains 1677,7216
* A border of size comprising of 10% or row and column length has been kept in each image

Annotation:

The csv contains the following attributes separated by comma.

* Filename: The name of file containing the image
* Width: The width of the image
* Height: The height of the image
* X1: X coordinate of Top Left corner of the bounding box
* Y1: Y coordinate of Top Left corner of the bounding box
* X2: X coordinate of Bottom Right corner of the bounding box
* Y2: Y coordinate of Bottom Right corner of the bounding box

**Comparative Analysis of Algorithms:**

Challenges:

Traffic Sign Detection and Recognition is a pure application of Autonomous Vehicles and we do face several challenges in its implementation. On the basis of these challenges, we would be evaluating our algorithms. Given the different color information and shapes due to multiple factors, we have following challenges:

* Gradient Moment:

The images captured from running vehicles are blurred and can have noise.

* Real Time Processing:

Since, the vehicles capture the data on real time they need results in real time as well. A little delay can be way too risky.

* Ambiguous Sign:

The sign may have some natural damage or cover that can lead to false classification.

* Imbalanced Lightening or Shadows:

The natural conditions such as rain, bright sunlight, different time of the day can lead to ambiguity in detection.

* Misleading Angle:

Since, the object (vehicle) is moving, the difference in angle may lead to detection of Sign board but false sign detection.

Analysis:

* Real Time Processing:

Support Vector Machines (SVM), MultiLayer Perceptron (MLP) and Decision Trees were compared for real time processing and Decision Trees were found to be fastest as well as most accurate and 770 times faster than SVM and 9.6 times faster than MLP.

* Imbalanced Lightening or Shadows:

RBG to YCbCr color space transformation was used to tackle this issue and Neural Network accuracy was observed to improve.

* Misleading Angle and Ambiguous Sign:

Since there is no dependence on error surface in the case of Genetic Algorithm it can solve multi-dimensional and multi directional issues.

Some other methods were also adopted and seemed effective like:

Hough based SVM is invariant to most of the problems like variable lightening, viewing angle etc. Gamma Compression and Image normalization were applied top counter further issues.

**Analysis and Results:**

With the discussion above we can evaluate the pros and cons to the different methods as:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Imbalanced Lightening | Real-time | Ambiguous Sign | Misleading Angle | Gradient Moment |
| Genetic Algorithm + Probabilistic NN | Y | Y | N | N | N |
| YCbCr + Image Normalization+NN | Y | Y | N | N | N |
| Hough based  SVM | Y | Y | Y | Y | N |