# Traffic Sign MATLAB Project Report

*Martin Tsvetanov - 37742493*

## Introduction

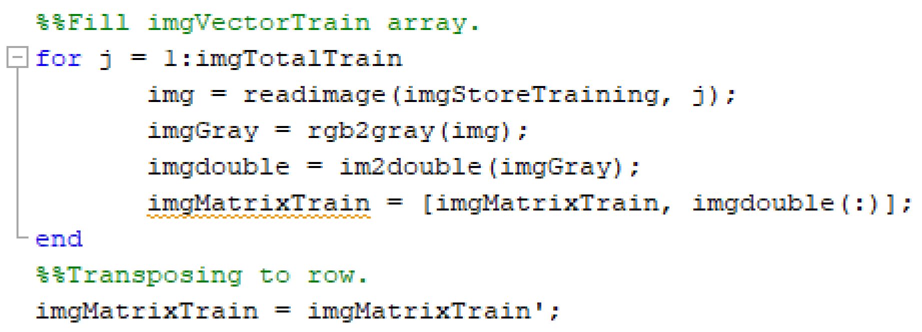
The goal of this project is to examine a dataset of road signs using machine learning algorithms and select the best classification model for most accurately determining the data in an image. The task is solved by using a supervised learning approach to discover the discriminative details (features).

## Datasets

To accomplish this task, I used two different private datasets. One is used to train the model and one is used for testing. Both datasets contain a collection of road signs with different speed limits, stored in subfolders that later become actors for label categorization. Those subfolders are named after the types of road sign images that they contain, which are “40”, “80”, “90”, and “100”. For testing the speed limits, some images were selected that make classification difficult because the details are obscured by light, angle, or even poor image quality. The training dataset contains a total of 278 images and the test dataset contains 69 images.

## Data Preparation

To use the images needed for training, I import each one into Matlab using the imageDatastore() function, which uses an ImageDatastore object to manage a collection of image files. In my code, they are stored in the imgStoreTraining and imgStoreTesting variables. Then, to prepare them for the classification algorithm, I convert them to grayscale, set their data type to double, and store them in a matrix using a for loop. Later, the matrix is transposed from image per row to image per column (see Figure 1. for more code details). This process is then repeated for the test images.

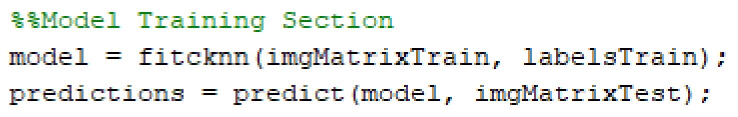


```
%%Fill imgVectorTrain array.
for j = 1:imgTotalTrain
        img = readimage(imgStoreTraining, j);
        imgGray = rgb2gray(img);
        imgdouble = im2double(imgGray);
        imgMatrixTrain = [imgMatrixTrain, imgdouble(:)];
end
%%Transposing to row.
imgMatrixTrain = imgMatrixTrain';
```

*Figure 1. - Image Matrix Preparation*

## Model Training and Evaluation Methods

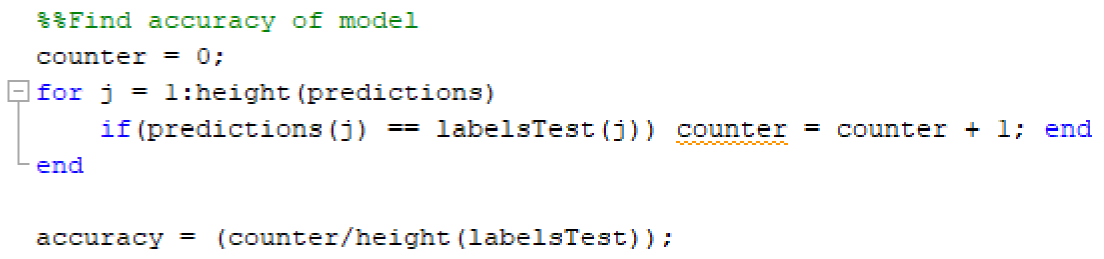
I have trained and compared three classification algorithms implemented in Statistics and Machine Learning Tools in Matlab. The classification algorithms used for the study are K-Nearest Neighbour, SVM for Multiclass, and Discriminant Analysis. The syntax for model training initiation requires a matrix, which holds the images, and a label vector, which the model uses to learn the correct guesses. When the model has completed its training, the predict() function is called and the predictions are stored in a variable called predictions. The predict() function takes two arguments, the trained model and the matrix of test images (see Figure 2. for further code details).



```
%%Model Training Section
model = fitcknn(imgMatrixTrain, labelsTrain);
predictions = predict(model, imgMatrixTest);
```

*Figure 2. - Model Training and Testing*

To evaluate the accuracy of a model, I use an evaluation method consisting of a for loop with a simple if condition that logs the correct estimates in a counter. Then, accuracy is calculated by dividing the number of correct predictions by the total number of labels (see Figure 3).



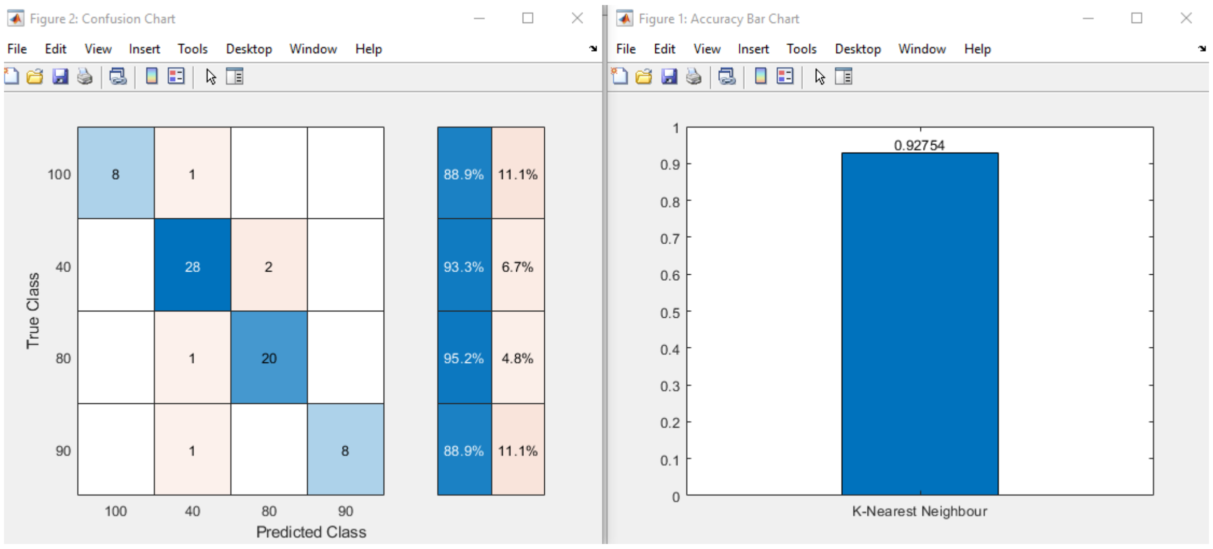
```
%%Find accuracy of model
counter = 0;
for j = 1:height(predictions)
    if(predictions(j) == labelsTest(j)) counter = counter + 1; end
end

accuracy = (counter/height(labelsTest));
```

*Figure 3. - Accuracy Evaluation*

I also created a bar graph and a confusion plot to visually display the accuracy and correct predictions of the model.

## Selected Model and Criteria

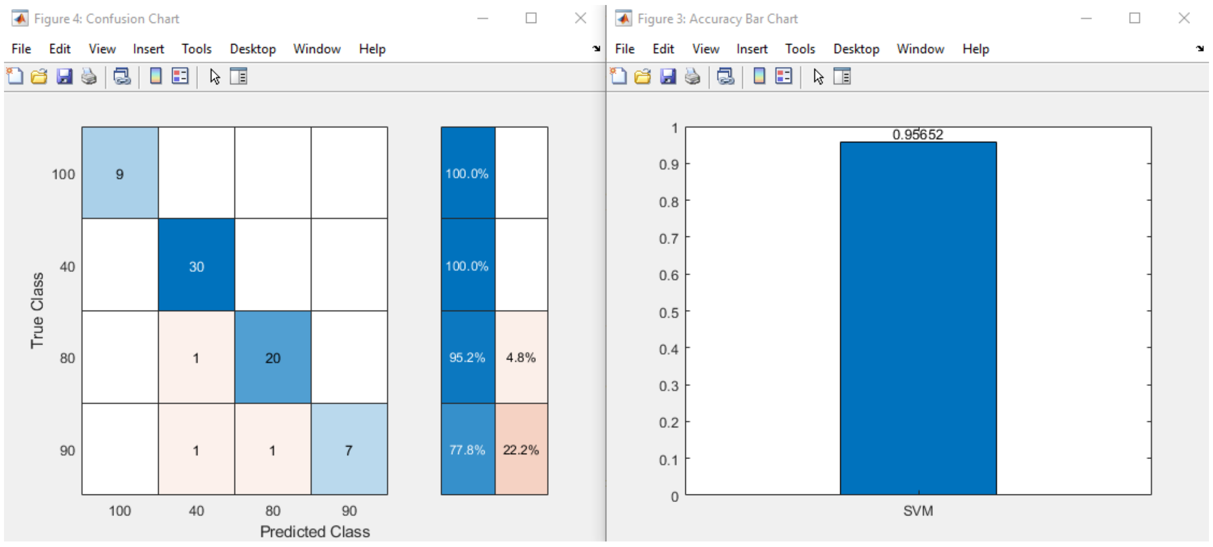
The first classification algorithm tested is K-Nearest Neighbour (KNN). The KNN's results are satisfactory, as it's accuracy is 92.7%, making 4 mistakes in total. You can see where the false predictions appear by looking at Figure 4.



| True \ Predicted | 100 | 40 | 80 | 90 | | |
|---|---|---|---|---|---|---|
| 100 | 8 | 1 | | | 88.9% | 11.1% |
| 40 | | 28 | 2 | | 93.3% | 6.7% |
| 80 | | 1 | 20 | | 95.2% | 4.8% |
| 90 | | 1 | | 8 | 88.9% | 11.1% |

Figure 4. - KNN Confusion Chart and Accuracy Bar Graph

The second algorithm tested is the SVM for Multiclass (SVM), which again provides us with a satisfactory result, with an accuracy of 95.6%, making 3 mistakes in total. See Figure 5 for further details on accuracy and false predictions.



| True \ Predicted | 100 | 40 | 80 | 90 | | |
|---|---|---|---|---|---|---|
| 100 | 9 | | | | 100.0% | |
| 40 | | 30 | | | 100.0% | |
| 80 | | 1 | 20 | | 95.2% | 4.8% |
| 90 | | 1 | 1 | 7 | 77.8% | 22.2% |

Figure 5. - SVM Confusion Chart and Accuracy Bar Graph

The third classification algorithm studied is Discriminant Analysis. This model reaches an excellent accuracy of 97.1% by making only 2 mistakes. See Figure 6 for more details.

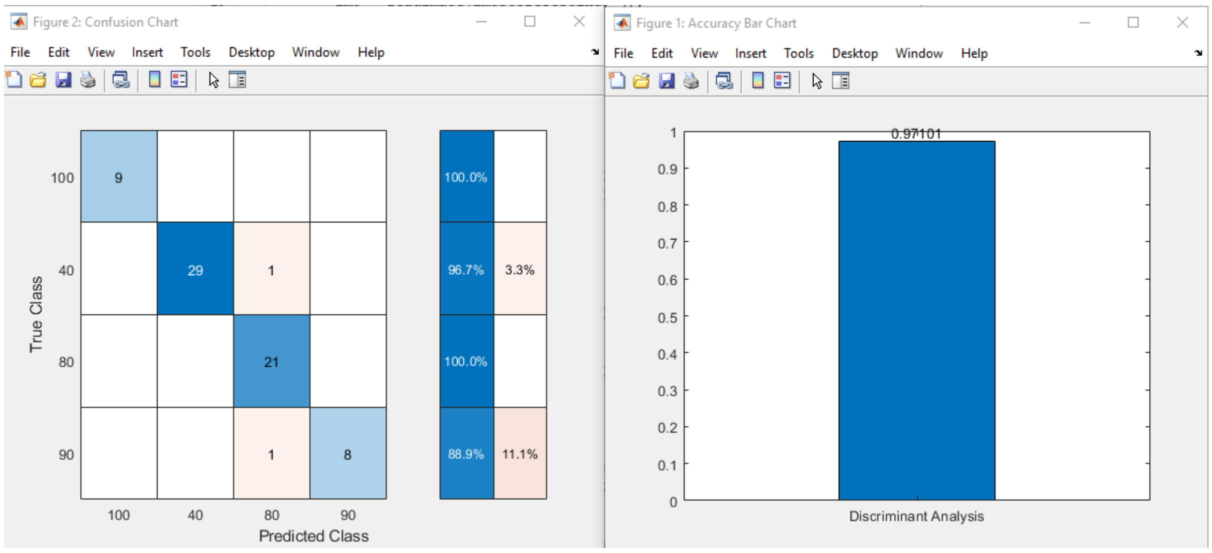


*Figure 6. - Discriminant Analysis Confusion Chart and Accuracy Bar Graph*

In summary, the most optimal algorithm for the completion of this task is the Discriminant Analysis, as it is the most accurate one.

## Error Analysis and Observations

Observing the errors, I have found that false predictions happen on images where the signs are obscured due to low image quality or side factors like light. I am satisfied with how the script is optimized, as the execution time is approximately 4 seconds.

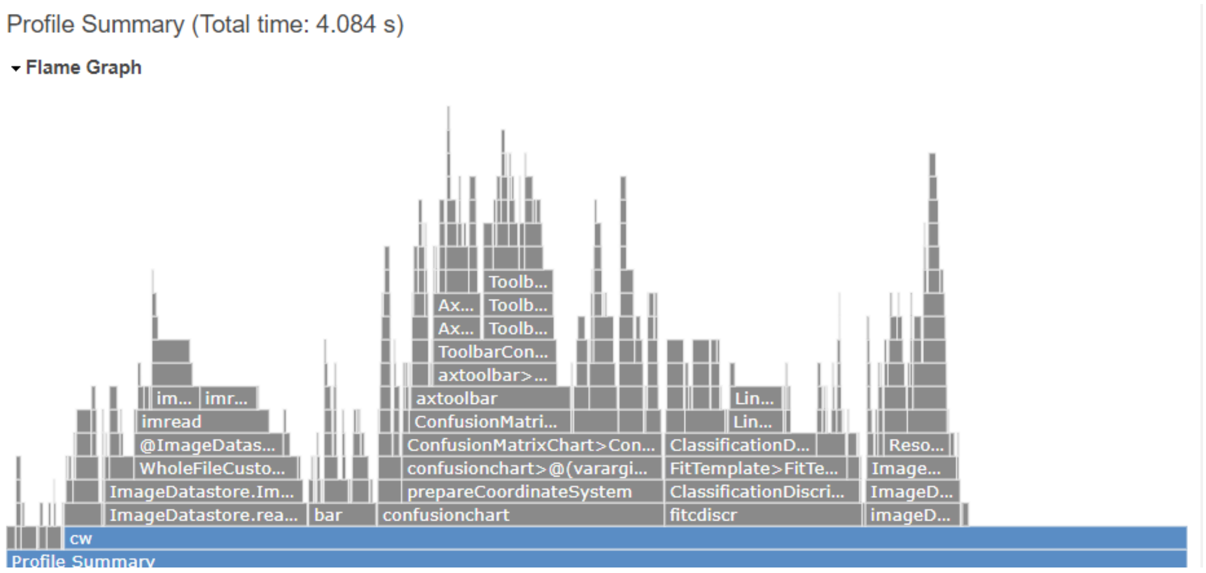


*Figure 7. - Profile Summary*