

ISYE 6740: Computational Data Analysis

Final Project

Traffic signs classification for vehicles

Authors

Madhumitha Ganesan
(gtID: 903624250)

Shanmukha Surya Teja Juttu
(gtID: 903567599)

Nagesh Kopparthi
(gtID: 903622019)

August 3, 2021

I. Introduction

Traffic sign recognition is a lucrative real-world problem which we have chosen to address due to its high practical relevance, especially in the context of its application in transportation. Traffic signs are designed to catch the eye of a human driver and be easily recognized. They usually follow a standard design, shape and color to aid accurate recognition. However, many real-world factors such as weather conditions, lighting in the environment and field of view might come into play; often leading to the drivers being misguided. There may also be instances when the human drivers may be unable to interpret the signs due to lack of knowledge. This can be mitigated using machine learning algorithms to accurately classify images of traffic signs, which could ultimately be used to prompt the drivers with clear instructions pertinent to the traffic signs. This capability can also be extrapolated to autonomous vehicles. To achieve level-5 autonomous, accurate classification would help the vehicles understand and follow traffic rules.

II. Problem Statement

The aim of this project is to read and classify traffic signs from multiple images of traffic signs captured from different ranges of lighting, distances and resolutions. We aim to improve get the best accuracy for specific signs like “warning” or “danger”, due to their nature of impact. We aim to interpret if specific features in an image like shape, color etc plays an important role in accurate classification of the groups. There are several different types of traffic signs like speed limits, no entry, traffic signals, turn left or right, children crossing, no passing of heavy vehicles, etc. These different classes of Traffic signs are grouped into 6 overarching categories and the classification modelling is performed to identifying the group of a traffic sign recognised by a car cam.

III. Methodology

Our methodology to building this traffic sign classification model can be broken down into five steps. This is a brief outline of the methodology. Each section will be elaborated in detail further in the report.

1. Exploratory data analysis
 - a. Understanding the distribution of traffic sign classes, sign shapes and image resolutions.
 - b. Performing Principal component analysis to identify the patterns for grouping
 - c. Grouping similar traffic signs into 6 subsets Speed, Prohibitory, De-restriction, Mandatory, Danger and Other based on the geometric shapes and structures as identified in the PCA
2. Data pre-processing
 - a. Scaling and resizing of images.
 - b. Down Sampling of data (depending on the modelling algorithm applied).
3. Training and evaluating 5 classification models:
 - a. Random Forest
 - b. Multi-Layer Perceptron Neural networks
 - c. Kernel SVM
 - d. KNN Classifier and
 - e. Linear Discriminant Analysis
4. Selecting the best performing model using validation dataset based on the evaluation metrics decided
5. Reporting the performance of selected model using a test set and interpreting the results by comparing against the human performance in identifying the traffic signs.

IV. Exploratory Data Analysis

1. **Data Source:** [GTSRB - German Traffic Sign Recognition Benchmark](#)

The dataset was created from approx. 10 hours of video that was recorded while driving on different road types in Germany in 2010. The traffic signs are extracted from the video. Data collection, annotation and image extraction was performed using the NISYS Advanced Development and Analysis Framework (ADAF)

Resulting is the traffic sign image data with varying resolutions, illumination conditions, colors and shapes of the signs. The dataset contains more than *50,000 images* of different traffic signs. Classified into *43 different classes*. The dataset is quite varying, some of the classes have many images while some classes have few images. The size of the dataset is around *300 MB*.

To ensure stochastic independence, the data is split into test and train considering

- (a) the overall class distribution is preserved for each individual set and
- (b) all images of one traffic sign instance are assigned to the same set

The test data set, we have used has *12,500 images*. The total number of training images are 39209.

Figure 1. is a snapshot of the 43 classes of images used for our model. As we can observe, there are multiple classes for speed limits, directions etc. We will further try to group this into subsets to reduce the number of classes.

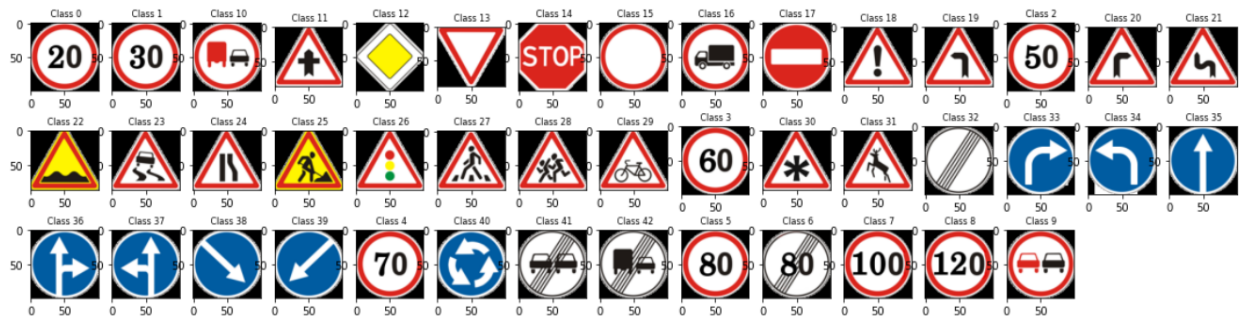


Figure 1. 43 Classes of Traffic Signs

2. Data Distribution

To understand the distribution of training and test images across each class, a histogram plot was used. Figure 2. shows the distribution of images across 43 classes. The distribution of data looks similar for test and train datasets. However, there is an imbalance in the number of images across classes. For example, Class 0 has around 150 training images, and Class 1 has over 2300 training images. This imbalance would cause the model to be skewed towards the class with greater number of images. We will apply sampling techniques to correct this.

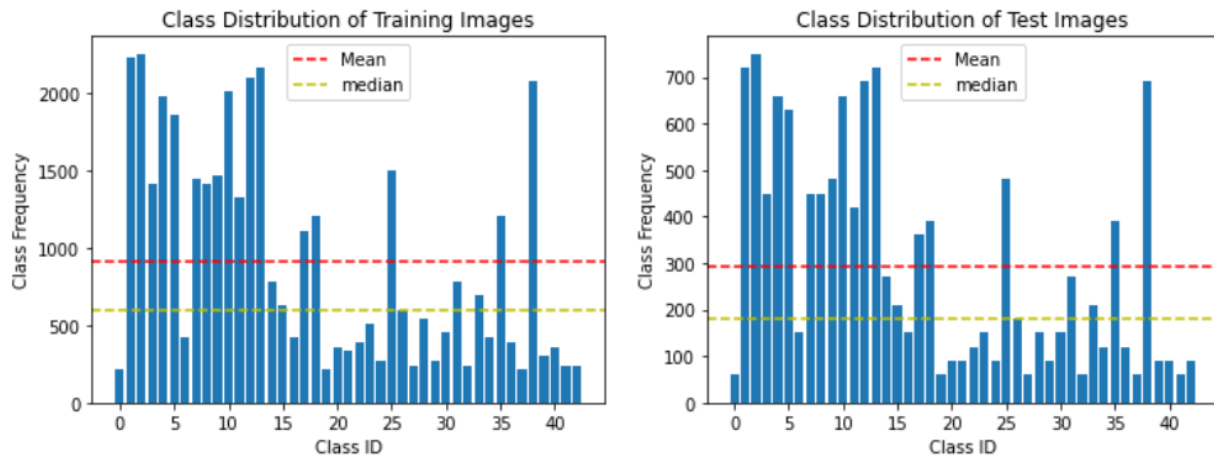


Figure 2. Distribution of Images across 43 classes

Image quality is another important for accurate classification. Figure 3. represents the range of pixels (resolution of images) in the training and testing data sets. Since image quality ranges from 1500 pixels to

15000 pixels, we will be resizing all images to a common resolution (30 * 30 * 3) for further analysis and modelling.

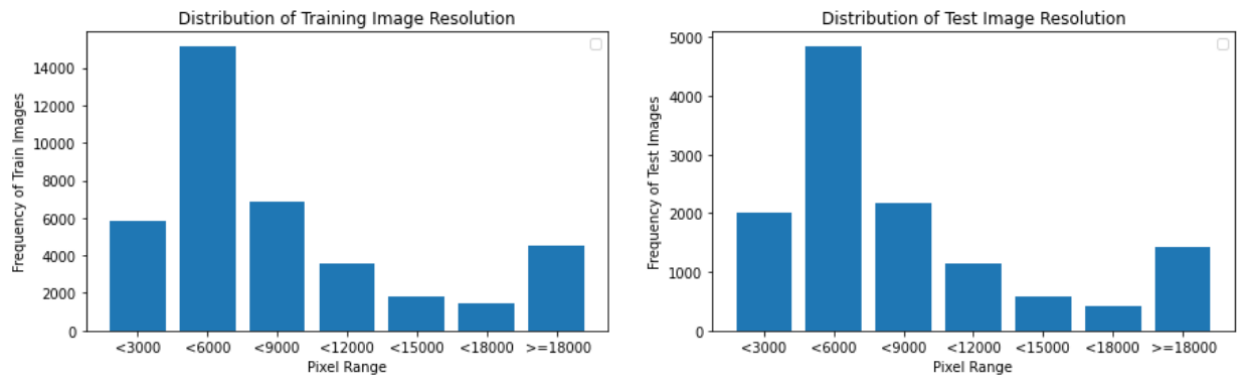


Figure 3. Distribution of Pixels (Not cumulative)

3. Image Analysis using Principal Components

To understand important features in images of each class, a principal component analysis was performed to visualize the top 5 principal components across all training images for each class. For example, 210 images of class 0 were reduced to 5 images represent its principal components. Figure 4. displays results from a few classes. For speed limit 60, the numeric is captured only in the 5th principal component. This is because of many low-resolution images for this class. Similarly, for signs for Road intersection and Left curve, triangle shape appears dominant. For simplicity and better interpretation, this analysis proved useful to group classes into subsets for classification. The specific subsets will be explained in the next section.

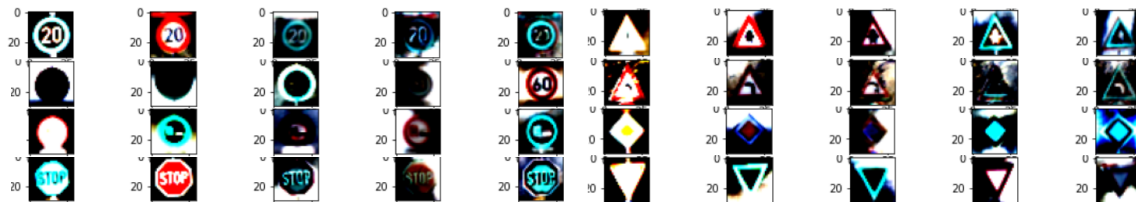


Figure 4. Principal components of Speed limit 20, 60, No Entry, Stop, Road Intersection Left Curve, Main Road, Give Way (read from top left downwards)

IV. Data Pre-processing

This section details the pre-processing of input data before modelling. Pixels in range (0-255) we scaled down to (0-1) to aid in computation. Due to different resolutions of input images, all images were resized to a resolution of (30 * 30 * 3). Figure 5. shows the pixel distribution highlighting the common resolution chosen for modelling. To study the effect of color in classification, images were also resized to (30 * 30) Gray-scale.

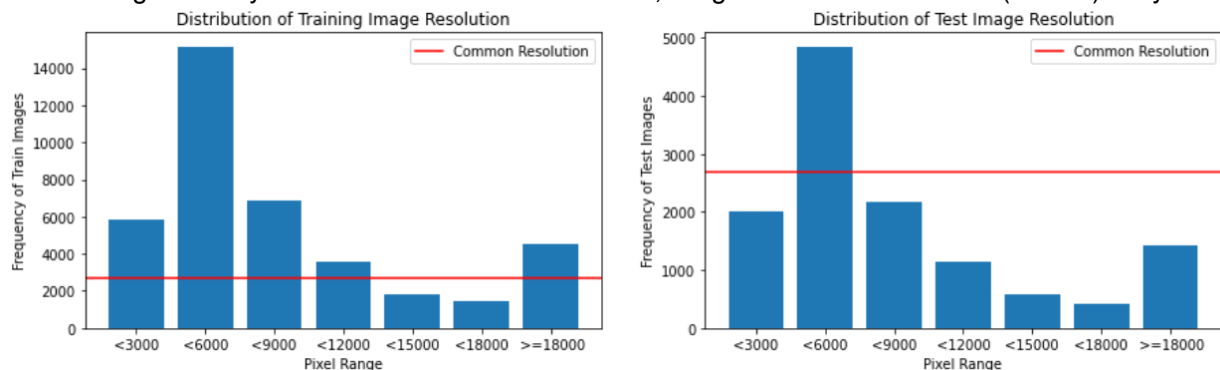


Figure 5. Pixel distribution including common resolution

4. Grouping of classes

Following principal component analysis, images from 43 classes were grouped into 6 classes. Table 1. shows the final grouped categories.

Group / Label	Group Description	Original Classes
1	Speed	[0,1,2,3,4,5,7,8]
2	Prohibitory	[9,10,15,16]
3	De-restriction	[6,32,41,42]
4	Mandatory	[33,34,35,36,37,38,39,40]
5	Danger	[11,18,19,20,21,22,23,24,25,26,27,28,29,30,31]
6	Other	[12,14,13,17]

Table 1. Grouping of classes







Group	Class of traffic signs
1 - Speed	
2 - Prohibitory	
3 - De-restriction	
4 - Mandatory	
5 - Danger	
6 - Other	

Figure 6. shows the distribution of images across the 6 groups.

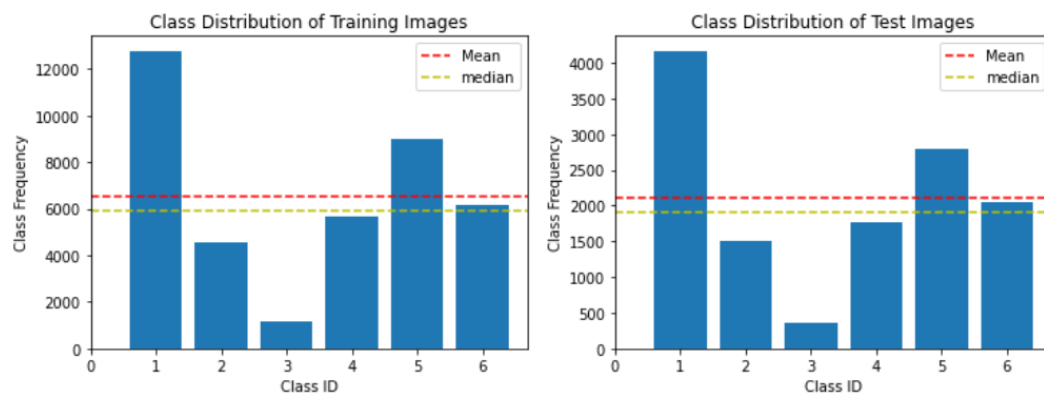


Figure 6. Distribution of Images after grouping.

V. Modeling

We developed 5 different supervised classification models using the input images for training, subject to the following steps as needed:

- Under Sampling: Since distribution of images had imbalances, we performed random under sampling of data to balance distribution across classes. Models were executed on the original dataset without sampling also, to understand impacts and results.
- Scaling/ Re-sizing: Images were scaled to re-sized to a common resolution.
- Cross-Validation: We performed K-fold to get the optimal hyperparameters based on a weighted average of model accuracy, precision, and recall. Specific steps for hyperparameter tuning are detailed below for each model.

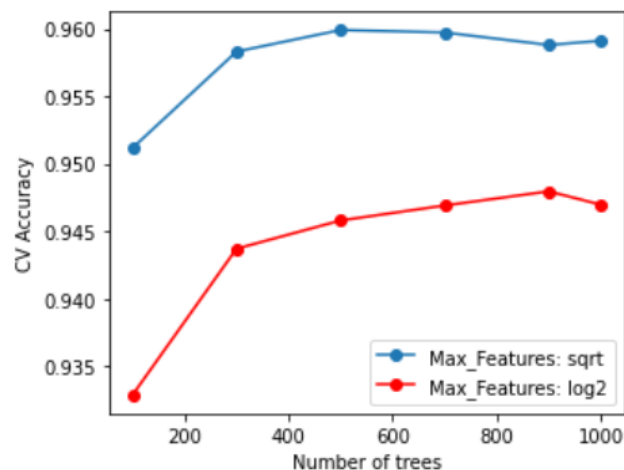
Our goal was to improve recall and evaluate classification of classes that take a higher priority (danger, prohibitory).

(1) Random Forest

A Random Forest model utilizes bagging to aggregate decision trees into a more generalizable predictive model. Data was scaled and resized to a common resolution. The number of trees to use was tuned using cross-validation. The number of trees is a trade-off between accuracy and computation time. Generally, the more trees in a random forest, the more accurate the model and the longer it takes. To be computationally efficient, we under sampled input data, and used 500 trees for building the model. Figure 7. shows the cross-validation accuracy score which stabilizes after 500 trees with square root of original features.

Validation accuracy: 0.96

Test accuracy: 0.93

*Figure 7. Cross Validation plot for Optimal Number of trees*

	precision	recall	f1-score	support
1	0.86	0.99	0.92	4170
2	0.99	0.83	0.90	1500
3	1.00	0.61	0.76	360
4	0.97	0.84	0.90	1770
5	0.96	0.98	0.97	2790
6	0.95	0.92	0.93	2040
accuracy			0.93	12630
macro avg	0.95	0.86	0.90	12630
weighted avg	0.93	0.93	0.92	12630

Figure 8. Performance on test set – Random Forest

[4127	3	0	9	20	11]
[214	1241	0	6	35	4]
[119	0	221	1	0	19]
[194	0	0	1484	21	71]
[51	0	0	3	2732	4]
[79	13	0	34	28	1886]]

Figure 9. Classification matrix of test set predictions – Random Forest**(2) Linear Discriminant Analysis (LDA)**

We had compared different solvers like singular value decomposition, eigenvalue decomposition and least squares solution to arrive at the best performing model during cross-validation. We have finally used singular value decomposition solver with 5 components in our LDA solver. The following classification accuracy results were achieved.

Validation accuracy: 0.83**Test accuracy: 0.80**

	precision	recall	f1-score	support
1	0.75	0.95	0.84	4170
2	0.83	0.73	0.78	1500
3	0.68	0.52	0.59	360
4	0.68	0.68	0.68	1770
5	0.97	0.79	0.87	2790
6	0.84	0.72	0.77	2040
accuracy			0.80	12630
macro avg	0.79	0.73	0.75	12630
weighted avg	0.81	0.80	0.80	12630

Figure 10. Performance on test set – LDA

[3967	34	42	104	5	18]
[293	1094	0	85	8	20]
[139	18	186	0	0	17]
[313	69	14	1202	36	136]
[364	37	15	92	2199	83]
[191	62	18	286	23	1460]]

Figure 11. Classification matrix of test set predictions – LDA

(3) K-Nearest Neighbors (KNN)

KNN uses a distance-based measure to classify a data point based on the nearest neighbors. K-fold cross validation was used to identify optimal K. KNN performed on entire data set to determine the best K that minimizes the Classification error.

It is interesting to see that, in the elbow curve, as the k increases, the error rate increases, we chose the optimal k=3 as chosen by cross validation

Validation accuracy = 93%

Test accuracy = 69%

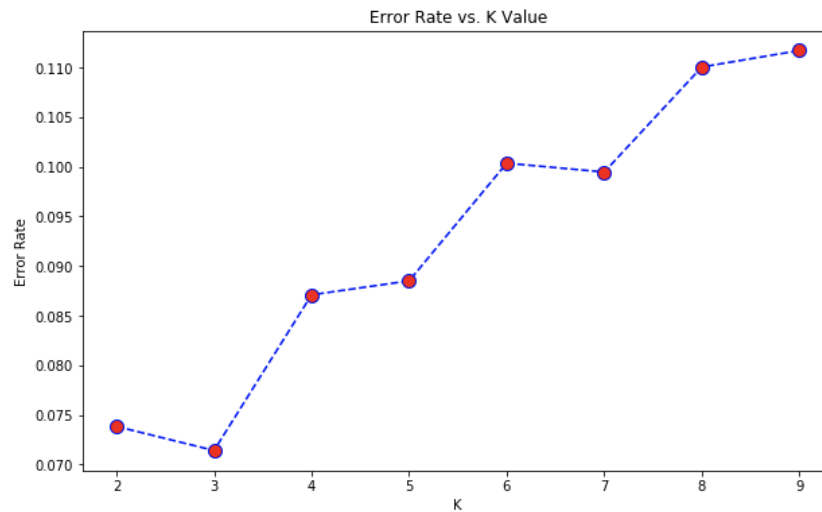


Figure 12. Performance on test set – KNN

	precision	recall	f1-score	support
1	0.64	0.85	0.73	4170
2	0.59	0.54	0.56	1500
3	0.77	0.68	0.72	360
4	0.61	0.43	0.50	1770
5	0.83	0.77	0.80	2790
6	0.77	0.60	0.67	2040
accuracy			0.69	12630
macro avg	0.70	0.64	0.66	12630
weighted avg	0.70	0.69	0.68	12630

[3547	258	25	152	87	101]
[539	803	8	35	74	41]
[88	4	245	1	3	19]
[510	159	8	757	185	151]
[426	51	9	107	2135	62]
[415	89	22	192	97	1225]]

Figure 13. Classification matrix of test set predictions – KNN

(4) Neural networks

We used the following parameters to build the final Neural Network classification model that yielded good classification results. The output layer with 43 nodes worked better than the one with one 6 nodes, we believe this is because the neural net model was able to better differentiate between the original 43 classes than the 6 sub-groups.

Activation function: RELU

Solver: Stochastic Gradient Descent

Hidden layer size: 43 nodes and 15 layers

Validation accuracy: 0.93

Test accuracy: 0.93

	precision	recall	f1-score	support
1	0.93	0.97	0.95	4170
2	0.93	0.90	0.92	1500
3	0.95	0.87	0.91	360
4	0.93	0.87	0.90	1770
5	0.96	0.96	0.96	2790
6	0.91	0.91	0.91	2040
accuracy			0.93	12630
macro avg	0.93	0.91	0.92	12630
weighted avg	0.93	0.93	0.93	12630

Figure 14. Performance on test set – Neural Network (Multi Layer Perceptron)

[4047	18	9	35	17	44]
[86	1350	3	20	28	13]
[30	16	313	0	0	1]
[74	30	1	1542	25	98]
[63	7	2	10	2685	23]
[56	28	3	55	43	1855]]

Figure 15. Classification matrix of test set predictions – Neural Network (Multi Layer Perceptron)

(5) Support Vector Machine (SVM)- Radial kernel

From EDA we noticed that the data is not linearly separable. To add flexibility to the model, we used a radial basis function (kernel) in building the SVM classification model.

Validation accuracy: 0.82

Test accuracy: 0.81

	precision	recall	f1-score	support
1	0.90	0.87	0.88	4170
2	0.60	0.85	0.71	1500
3	0.91	0.84	0.87	360
4	0.65	0.62	0.63	1770
5	0.89	0.85	0.87	2790
6	0.83	0.74	0.78	2040
accuracy			0.81	12630
macro avg	0.80	0.79	0.79	12630
weighted avg	0.82	0.81	0.81	12630

Figure 16. Performance on test set – rbf kernelized SVM

[3618	266	12	167	45	62]
[82	1275	2	110	19	12]
[27	12	301	0	0	20]
[82	318	9	1093	93	175]
[134	174	5	70	2367	40]
[93	70	1	235	126	1515]]

Figure 17. Classification matrix of test set predictions – RBF Kernelized SVM

Evaluation and Results

To measure how each model performs and be able to determine the one with the best performance, we used the test images provided with the data. Since we are more interested in predicting sign groups correctly, we used recall as our main measure of performance. The results of the 5 models are summarized below.

Model	Train time (in sec)	Overall Accuracy	Weighted Avg. Recall	Recall range
Random Forest	500	93%	93%	61-99%
Linear Discriminant Analysis (LDA)	6	80%	80%	52-95%
K Nearest Neighbors	6	70%	69%	43-85%
Neural Networks (MLP)	53	93%	93%	87-97%
Support Vector Machine (RBF)	114	80%	81%	62-87%

	Speed	Prohibitory	Derestriction	Mandatory	Danger	Other
Human	97.63%	99.93%	98.89%	99.72%	98.67%	100%
Random F.	99%	83%	61%	84%	98%	92%
LDA	95%	73%	52%	68%	79%	72%
KNN	85%	54%	68%	43%	77%	60%
Neural N.	97%	90%	87%	87%	96%	91%
SVM	87%	85%	84%	62%	85%	74%

We have also compared below the performance of all the classification algorithms against human results obtained from the research paper. For the groups with less data, the classification struggle to get good accuracy. Only Neural nets and SVM could deliver good classification rates for class 3 which has the lowest training samples.

Observation and recommendations:

- We were able to observe that Class 3 has bad recall across all models owing to presence of lesser training data – this can be overcome by having a more extensive and balanced training set across all classes
- We could also observe that LDA performs better than KNN in terms of both training time and recall
- The above results clearly signify that Neural Networks is the best classifier for the given data outperforming KNN, SVM and LDA in terms of recall and Random forest in terms of training time with similar recall
- Although none of the models could outperform the average human accuracy, we are certain that with the presence of a more extensive data set with better image quality and resolution and better resources to handle computational complexity, Neural Networks would perform a comparable job to the average human and the classification algorithms could be used as ancillary prompting tools in vehicles

Packages used (Python):

```
#Models
from sklearn.model_selection import RepeatedStratifiedKFold, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.svm import SVC, NuSVC
from sklearn.neural_network import MLPClassifier
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import auc
```

Contributions:

Madhumitha Ganesan: Random Forest and Linear Discriminant Analysis

Shanmukha Surya Teja Juttu: Neural Networks and KNN

Nagesh Kopparthi: Exploratory Data Analysis and SVM

Group: Upon collating results from our respective analyses and models, we discussed our model evaluation metrics and compiled a list of observations and recommendations

References:

- Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton - ImageNet Classification with Deep Convolutional Neural Networks
URL: <https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>
- J. Stallkamp, M. Schlipsing, J. Salmen and C. Igel Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition
URL: <https://www.sciencedirect.com/science/article/abs/pii/S0893608012000457>
- Data source : GTSRB - German Traffic Sign Recognition Benchmark
URLs: <https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign>
<https://benchmark.ini.rub.de/>