Data Mining & ML

Group 4

November 2020

Contents

1 File management, data pre-processing, transformation and selection

1.1 File Management

To easily reuse repetitive code across all Tasks we have developed modules (See Figure A.1) inside the Scripts directory which contains code that performs various tasks. These functions help in keeping our notebooks neat and tidy, for example, getting file paths for our data files or inserting data values into a dataframe to be used later on.

To better understand the classified data we have implemented a confusion matrix as well as a function that displays all confusion matrices automatically for specified street signs, this provided quick analytics needed for Task 4 & 5.

1.2 Data pre-processing

1.3 Transformations and selection

Downsampling

We have used downsampling (downsampling.py) to get more information about exposing low level features. (Figure A.2) By downsampling images using downsample.py we are able to reduce the image size by averaging 4 pixels into 1. This way we are able to find out what effect each area of the image has on the classification. We have also downsampled the image to a 12x12 image (See A.3) By comparing the average greyscale to the row of the pixels, we can see where the darkest pixels are located (Lower greyscale value is darker, higher is lighter). In the heatmaps (Figure A.4) we can see that the darkest pixels were the most defining which distinguishes each type of street sign.

Balancing

See (Figure 2.1)

Binning

By implementing equal width binning we were able to convert the attributes from numerical to nominal type which enabled us to use Categorical Naive Bayes. Since the conditional probability distribution table is smaller, we learned that [Analysis/Conclusion Here]

2 Naive Bayesian Networks

To apply Naive Bayes classifier we have created a script called NaiveBayesCategorical.py / NaiveBayesGaussian.py which takes care of building NB models using the sklearn Python Library as well as splitting the data into training and testing which is required to appropriately find out how accurate a NB classifier is.

2.1 Equal Width Binning

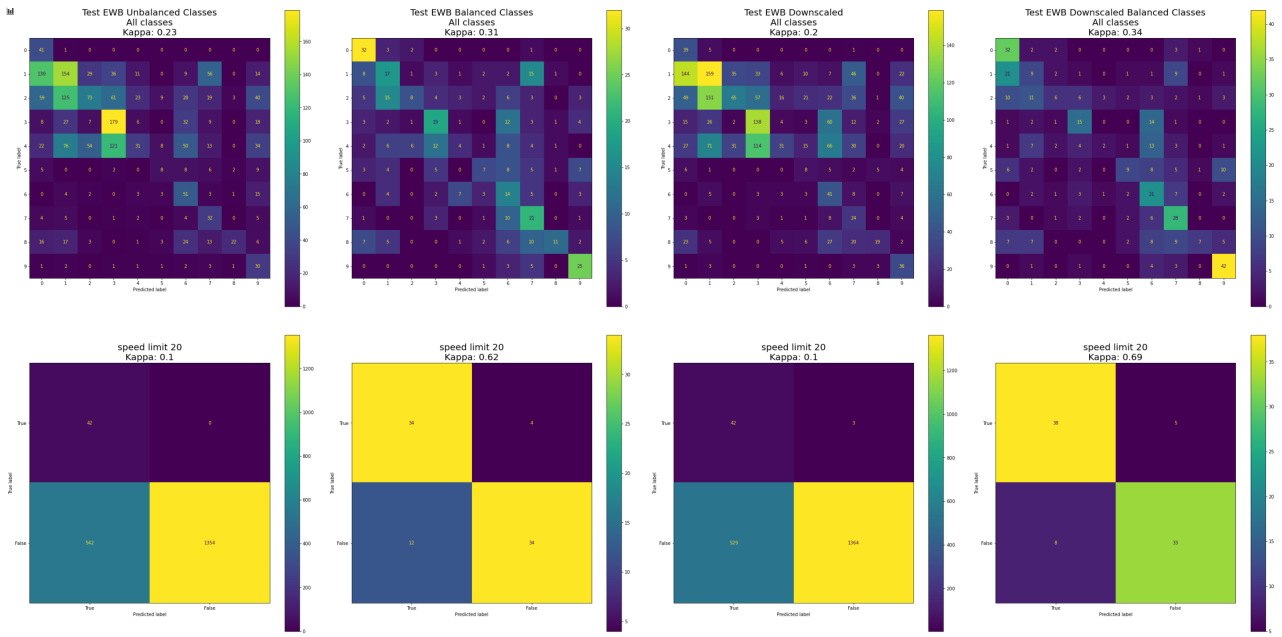


IMAGE NEEDS RESIZING TO READ TEXT

Observations

1. Using no preprocessing technique there is a severe negative effect on the quality of predictions.
2. Just downscaling the images without other preprocessing provides no real benefit.
3. Naive Bayes has very poor accuracy when attempting to classify all classes, a simple binary classification performs far better.

(Figure A.5)

Observation: Having unbalanced class distribution negatively affects Naive Bayes across the board.

3 Complex Bayes nets

Describe & analyse the problem. Show all experiments complete with graphs and tables. Discuss produced software quality & discuss interesting properties of the data and algorithms

3.1 Building Bayes Networks

Bayes networks represent probablistic directed acyclic graphs that define the relationshps between conditional dependencies and random variables. A naive Bayesian network can be represented in a Bayes network where the node representing the probability distribution of the class is the only parent of all other nodes and no other edges exist in the network. By adding additional edges (so long as the graph remains acyclic) we can represent causal relations between random variables.

We decided to approach this task using both Weka and Python, with the intention of verifying our results against the other. Attempting to build the network both ways gave us solid insights into the problems that would have to be solved to produce a Bayes Network. We decided to use the pgmpy library to build our Bayes Networks in python, this immediately presented us with 2 computational complexity problems:

1. Building all the conditional probability factors.
2. Learning the optimal edges.

We handled the first problem by discretizing the greyscale values using equal width and frequency binning (see section 1.3) When using Weka to compute Bayesian networks we observed that Weka would perform extremely aggressive binning of the greyscale values often discretizing down to only 2 bins. This had a profound effect on the speed of learning the parameters and edges.

3.2 Algorithms & Data

3.3 Experimental Results

4 Clustering

Appendices

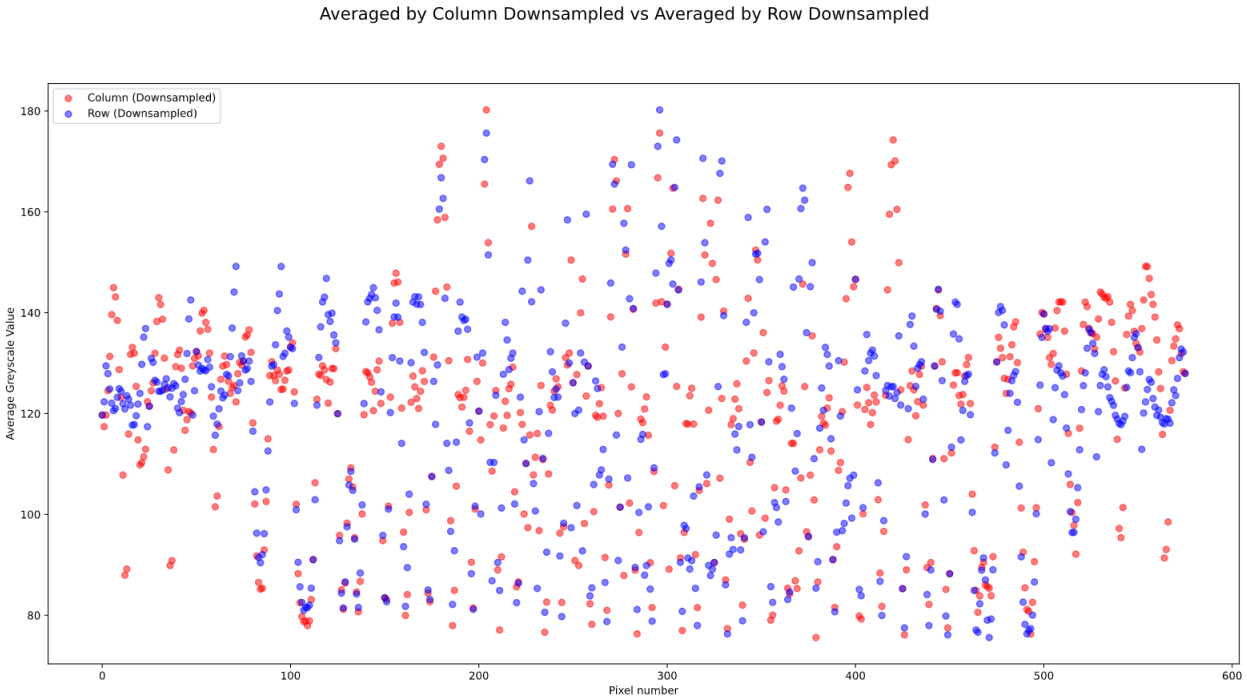
A Appendix A

A.1 Module Table

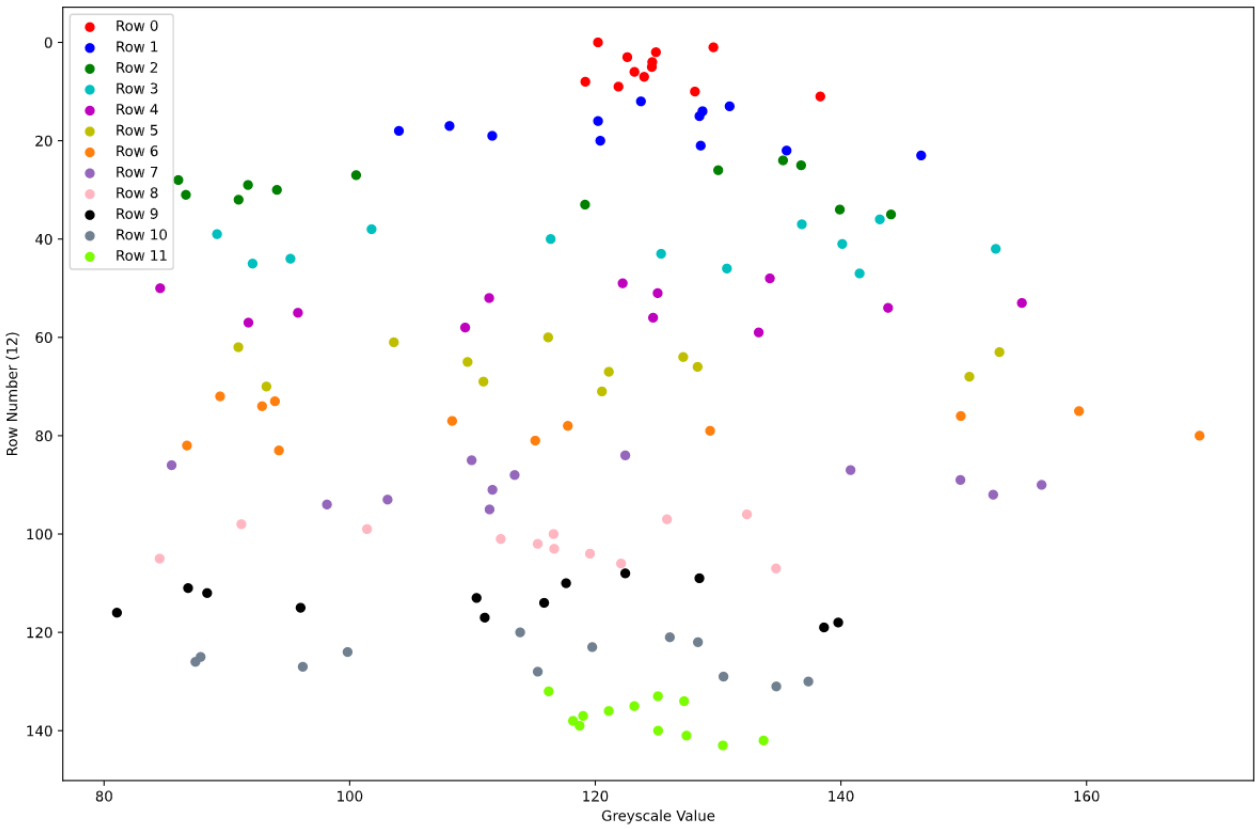
The following are located in the Scripts folder

Module Name	**Description**
helperfn.py	Provides functions to load and transform with all datasets required.
downsample.py	Provides functions to downsample images
pixelFinder.py	Provides functions to find the most important pixels within a dataset of a chosen street sign.
bayseNet.py	Provides functions used for getting a score for a model by testing all test data against labels.
confusionMatrix.py	Provides functions for building and displaying confusion matrices as well as methods for calculating kappa values.
plotScripts.py	Provides functions for plotting data into graphs
wekaConversion.py	Provides functions to convert preprocessed data to be consumable by Weka

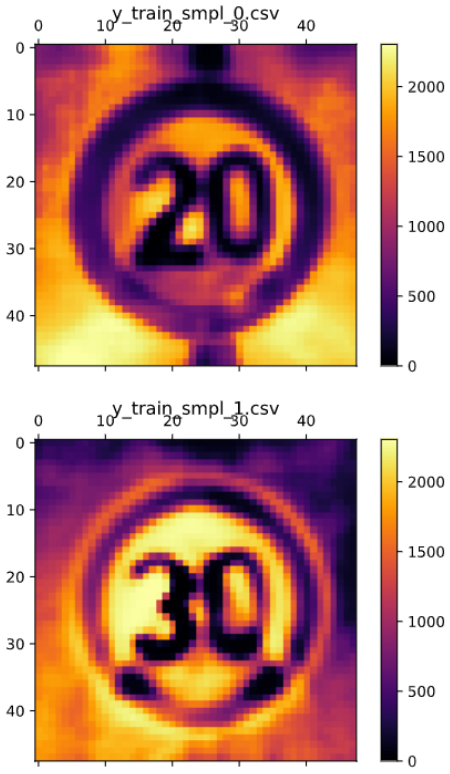
A.2 Averaged by Column Downsampled vs Average by Row Downsampled



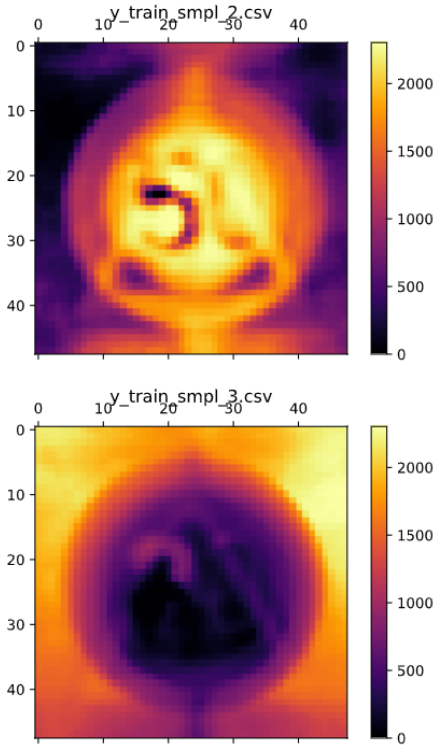
A.3 12x12 Downsampled image



A.4 Heatmaps

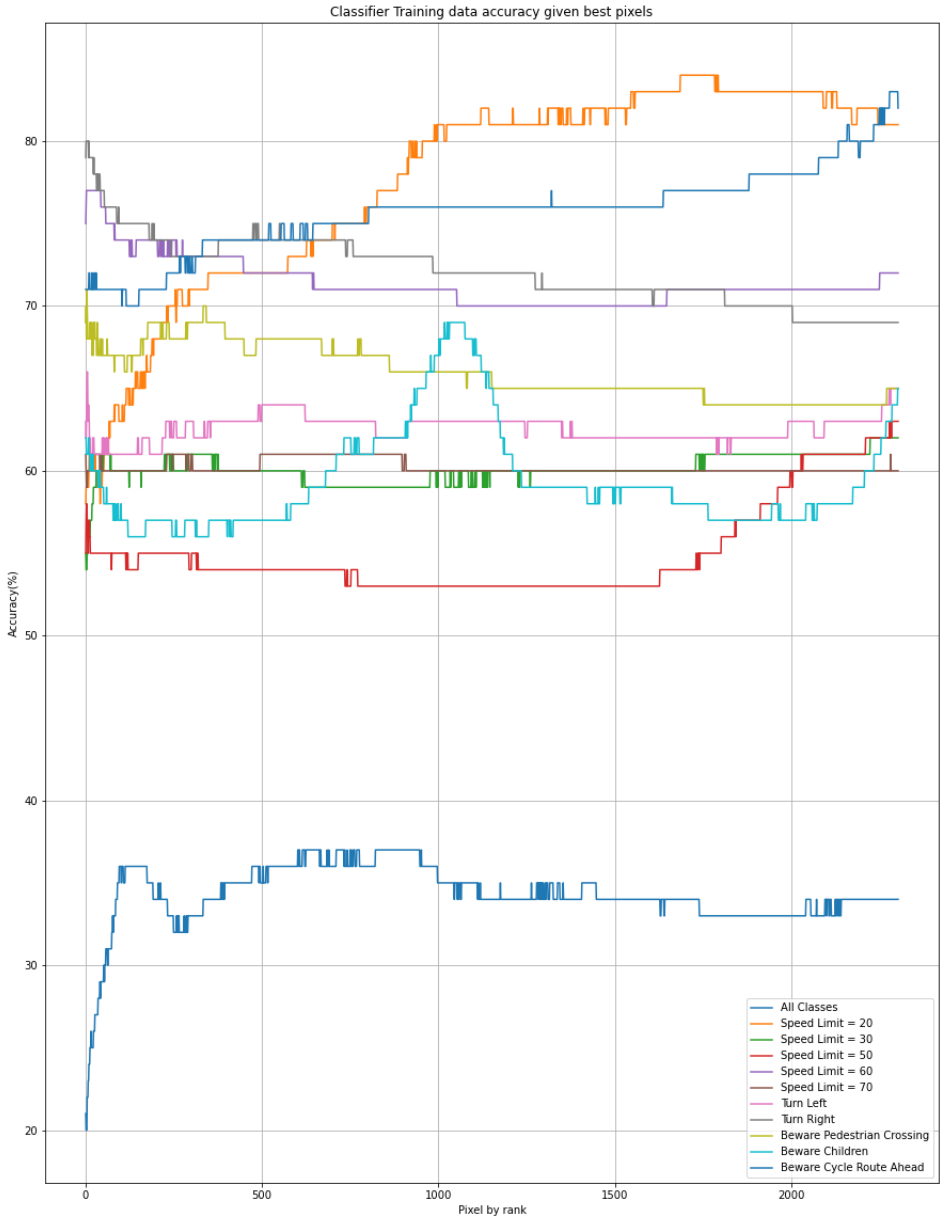


(a) 20mph & 60mph



(b) 50mph & 60mph

A.5 Training data accuracy



A.6 Training confusion matrices for Naive Bayes