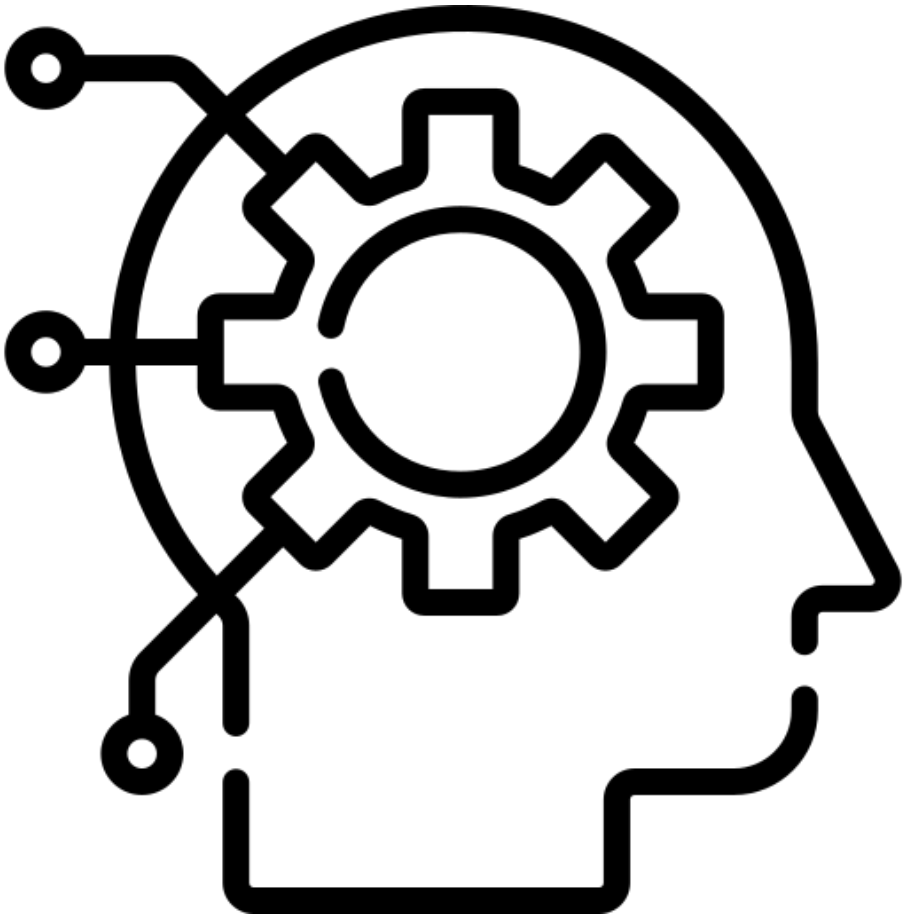
Queensland University of Technology

Machine Learning: Assessment 1A

CAB420



Don Kaluarachchi (N10496262)

Contents

List of Figures

List of Tables

1 Introduction

This report outlines the work done for the CAB420 Assessment 1A. The topics range from Regression to Classification to Deep Neural networks. This includes the word responses, diagrams and illustrations to explain the answers better and code snippets to show the process of work.

2 Regression

2.1 Pre-processing

In machine learning data undergoes a pre-processing step to improve the quality of data. Data pre-processing would consist of steps such as data cleaning , data transformation, data integration and Dimension Reduction (V7Labs, 2023).

Filtering and data cleaning to remove rows/columns are not to be performed for this dataset therefore no data is removed from this dataset. Standardisation is performed when the features of the input data set have large difference between thier ranges or are in different units. Performing standardisation so that each feature has a mean of 0 and a standard deviation of 1. By applying this to each dimension, we end up with all variables having a mean of 0 and standard deviation of 1, and thus our regularised regression can better consider all the terms equally. (Builtin, 2023).

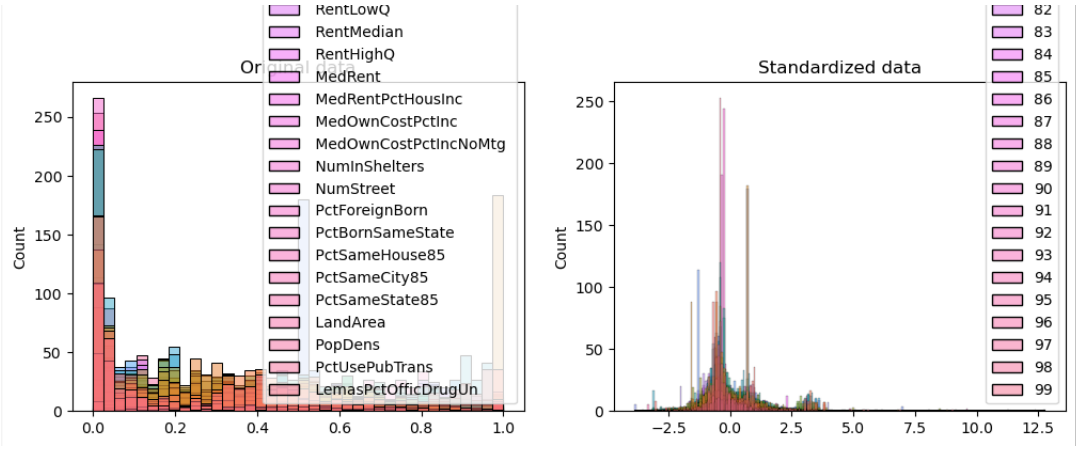


Figure 1: Original vs Standardised Data

2.2 Details of Models

The three models used for regression are,

1. Linear Regression Model :
   This is a statistical model that is used to describe the relationship between a dependent variable and one or more independent variables. relationship is given by the following equation,

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n + \epsilon \tag{1}$$

   where y is the dependant variable and x is the independant variable. Once this model is trained on data it can be used to make predictions on unknown data. This evaluation and performance is discussed in the following section.

2. LASSO Regression Model :
   LASSO stands for ”Least Absolute Shrinkage and Selection Operator”. It is a statistical formula for the regularisation of data models and feature selection. Regularization is implemented by adding a “penalty” term to the best fit derived from the trained data, to achieve a lesser variance with the tested data and also restricts the influence of predictor variables over the output variable by compressing their coefficients (Builtin, 2022).

   To choose the best value of lambda denoted by λ is calculated by iterating between values from 0 to 1 (on standardized data)and the best one is saved at each iteration. A lower value of RMSE indicates better performance of the model which allows to choose the correct value of lambda which controls the strength of the regularisation penalty.

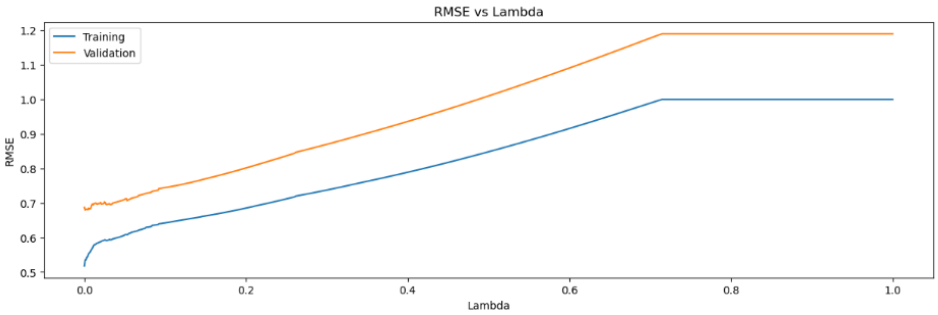


Figure 2: Lasso Regression Model: Lambda vs RMSE

3. Ridge Regression Model :
   The lambda value for this model is chosen using the same method as that of the Lasso regression model as the Ridge and Lasso model have similarities - especially during the implementation where the statsmodels uses the ”fit regularized” function that does both where the regularization term can be switched between 0 and 1.

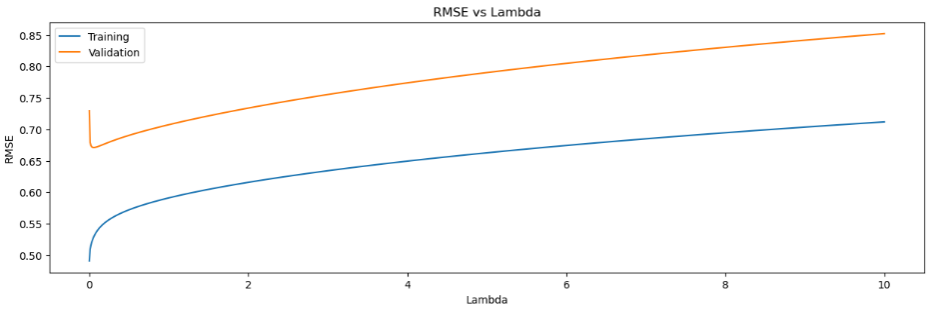


Figure 3: Ridge Regression Model: Lambda vs RMSE

2.3 Evaluation of Models

These models can be evaluated by checking their performance on the test set these include metrics such as R-squared, Mean Absolute Error, Mean Squared Error, Root Mean Squared Error, Q-Q Plots, Coefficient of determination.

The images given below are the results/info about each of the models when they have been run on the validation and testing data set which will be used to evaluate the performance of the models.

```
Mean squared error: 0.02
Variance score: 0.55
Linear Model Validation Data: RMSE = 0.15491210066258154
Linear Model Training Data: RMSE = 0.10419990101033945
                            OLS Regression Results
==============================================================================
Dep. Variable:      ViolentCrimesPerPop    R-squared:                       0.759
Model:                              OLS   Adj. R-squared:                  0.637
Method:                   Least Squares   F-statistic:                     6.207
Date:                  Sun, 16 Apr 2023   Prob (F-statistic):           7.72e-28
Time:                          08:34:29   Log-Likelihood:                251.07
No. Observations:                   298   AIC:                            -300.1
Df Residuals:                       197   BIC:                             73.27
Df Model:                           100
Covariance Type:              nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.2319      0.007     31.234      0.000       0.217       0.247
x1             0.0988      0.129      0.767      0.444      -0.155       0.353
x2            -0.0767      0.049     -1.571      0.118      -0.173       0.020
x3             0.0102      0.040      0.253      0.800      -0.069       0.090
x4            -0.0260      0.039     -0.667      0.506      -0.103       0.051
x5            -0.0185      0.020     -0.902      0.368      -0.059       0.022
x6            -0.0982      0.040     -2.464      0.015      -0.177      -0.020
...
x98     -0.039406
x99     -0.003160
x100     0.021252
```

Figure 4: Linear Regression Model Results

```
Best values on Validation Data set
Best R Squared = 0.7185647427447908
Best Adjusted = 0.5757042060670196
Best RMSE (val) = 0.142474394713848
Best RMSE (test) = 0.13154277144365992
Best coefficients on the normalised model
Best slope = [[ 0.00654498]
 [-0.08261657]
 [ 0.08734514]
 [-0.05381302]
 [ 0.00138797]
 [-0.17480713]
 [-0.06013663]
 [ 0.04399552]
 [ 0.06563322]
 [ 0.06039095]
 [ 0.00088043]
 [ 0.07462407]
 [ 0.05172256]
 [-0.05027768]
 [ 0.0145202 ]
 [-0.10611063]
 [ 0.06574661]
 [ 0.0472038 ]
 [-0.08570737]
...
 [-0.07059539]
 [-0.07982754]
 [-0.0185454 ]
 [ 0.10968608]]
```

Figure 5: Ridge Regression Model Results

```
Best values on Validation Data set
Best R Squared = 0.7134097799342415
Best Adjusted = 0.5679325108653286
Best RMSE (val) = 0.14422467514776394
Best RMSE (test) = 0.13433655408459
Best coefficients on the normalised model
Best slope = [[ 0.01686057]
 [-0.12863666]
 [ 0.15034114]
 [-0.03359002]
 [-0.00329562]
 [ 0.        ]
 [ 0.        ]
 [ 0.11409153]
 [ 0.        ]
 [ 0.11582227]
 [ 0.        ]
 [ 0.08619093]
 [ 0.06016083]
 [-0.08064985]
 [ 0.        ]
 [-0.09606443]
 [ 0.        ]
 [-0.00365155]
 [-0.1231907 ]
...
 [-0.08706783]
 [-0.09503593]
 [ 0.        ]
 [ 0.11207668]]
```

Figure 6: Lasso Regression Model Results

R squared

The R squared value is the proportion of the variance in the dependant variable to the independent variable. Where with a value of 0 indicating that the model captures nothing, and a value of 1 indicating that the model captures everything. The higher value of this indicates a better fit of the model.

Adjusted R squared is used as the measure of how well the model is fitted as it adjusts for the number of independent variables and avoids over fitting. The values are 0.637 (Linear), 0.576 (Ridge) , 0.567 (Lasso) where the linear model has been best fitted.

Mean Squared Error

The mean squared error is the average squared difference between the predicted values and the actual values given by the graph below.

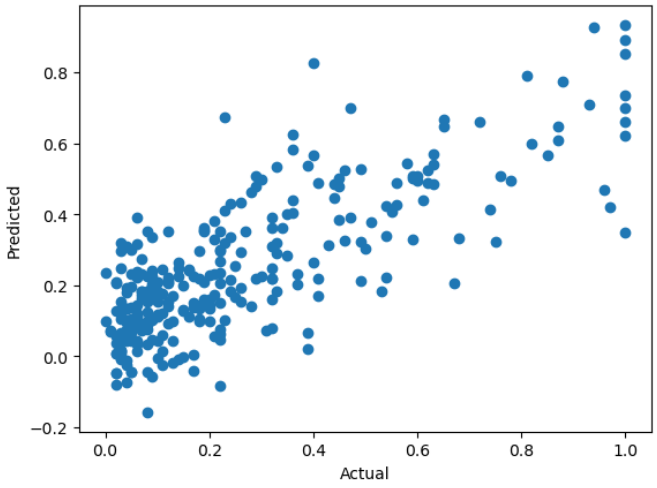


Figure 7: Actual vs Predicted values

All three models have a value of 0.02 which allows us to make the observation that the lambda value has been selected correctly.

Root Mean Squared Error

This is a measure of prediction error where a smaller value means it is better. The values are 0.155 (Linear), 0.132 (Ridge) , 0.134 (Lasso) where the linear model has been best fitted.

Q-Q Plot

A Q-Q plot, or quantile-quantile plot, is a graphical method for comparing a data distribution to a theoretical distribution. It is a probability plot that compares two probability distributions by plotting their quantiles against each other. It can also be used to identify outliers or other anomalies in the data.

For all three models the plot the data does follow the straight line however there are some points at the tail which deviate from the stright line indicating there might be outliers in the data.

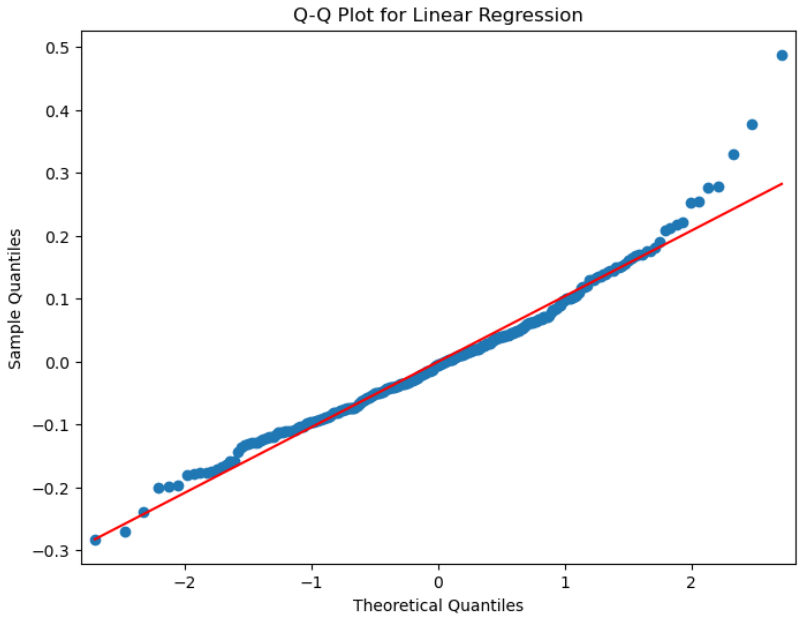


Figure 8: Q-Q Plot Linear Regression

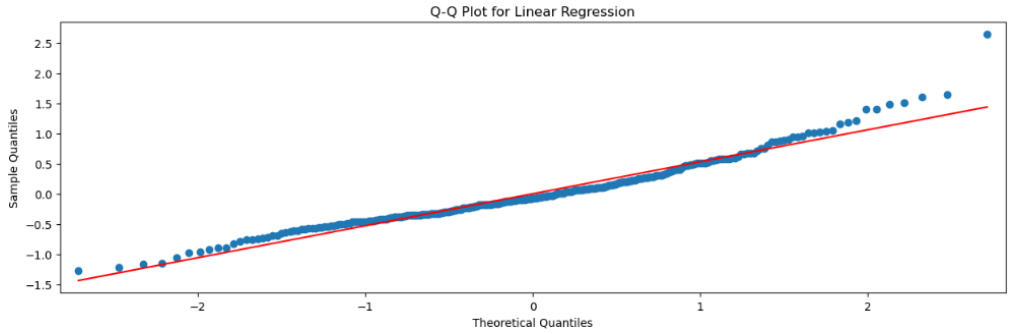


Figure 9: Q-Q Plot Ridge Regression

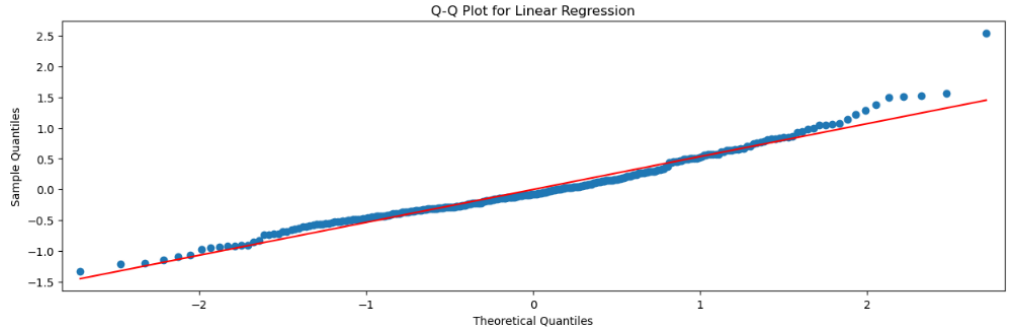


Figure 10: Q-Q Plot Lasso Regression

Test Perfomance

Given below are the test performance graphs for both Ridge and Lasso models. The plots align for the training and testing set indicating good performance.

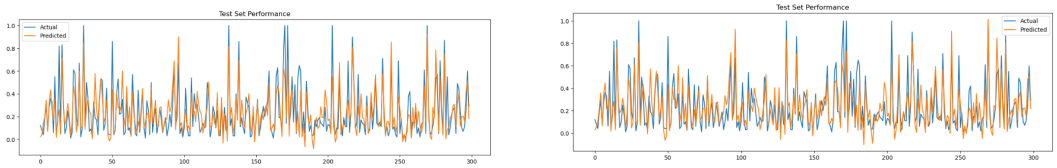


Figure 11: Test Performance plot for Ridge and Lasso models

2.4 Ethical Concerns

When dealing with sensitive data the training data used, the variables compared and models used need to be evaluated as there can be potential biases and misinterpretations of the data if done incorrectly.

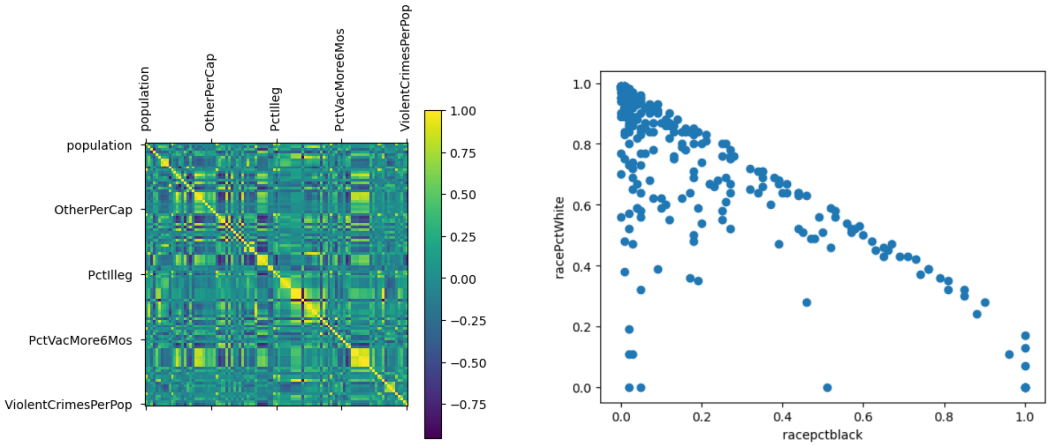


Figure 12: Correlation plot of all data and Correlation plot of different races

The presented graphs above show the correlation between all of the data. The yellow-green areas indicate high levels of correlation between the columns of data. The graph on the right shows one of the biases that come into play when dealing with sensitive information. For example this chose plot chose that there is a negative correlation between the population of one race in the presence of the other. This is a highly sensitive premise to make and one that could have adverse effects if wrong conclusions are drawn.

Researchers, Machine Learning Scientists should therefore be wary. They should assess the existing data set that they have and evaluate if these already consist of biases and prejudices. They should consider which parameters are going to be compared. What predictions are made and do these exacerbate existing social inequalities.

They must use appropriate data sources that do not contain biases and more distributed data and state the limitations of the data used and be aware of the dangers present in the models trained.

3 Classification

3.1 Pre-processing

Similar to the previous pre-processing done in section 2 for regression, pre-processing is done during classification techniques as well. As previously stated and shown by the box plots shown below this makes sure that the feature data has a mean of 0 and a standard deviation of 1. By applying this to each dimension, we end up with all variables having a mean of 0 and standard deviation of 1.

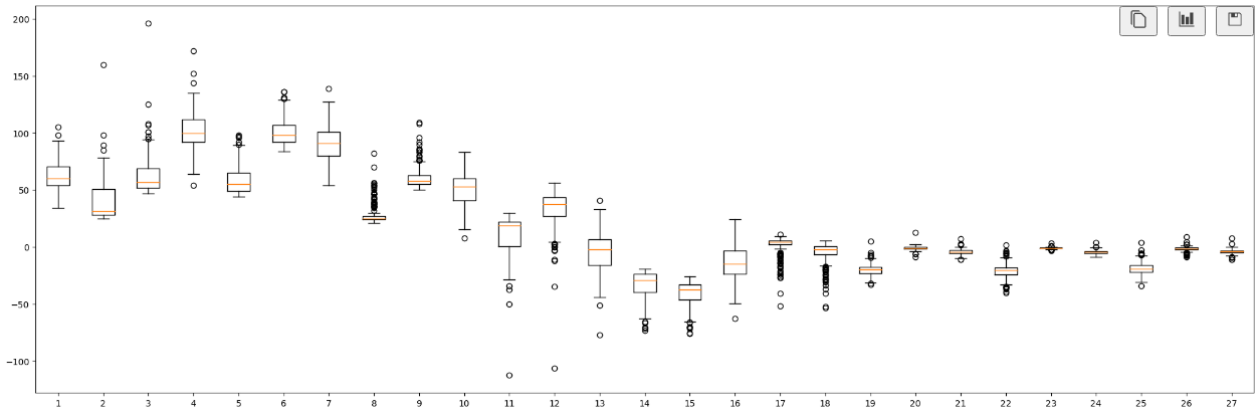


Figure 13: Pre Standardization

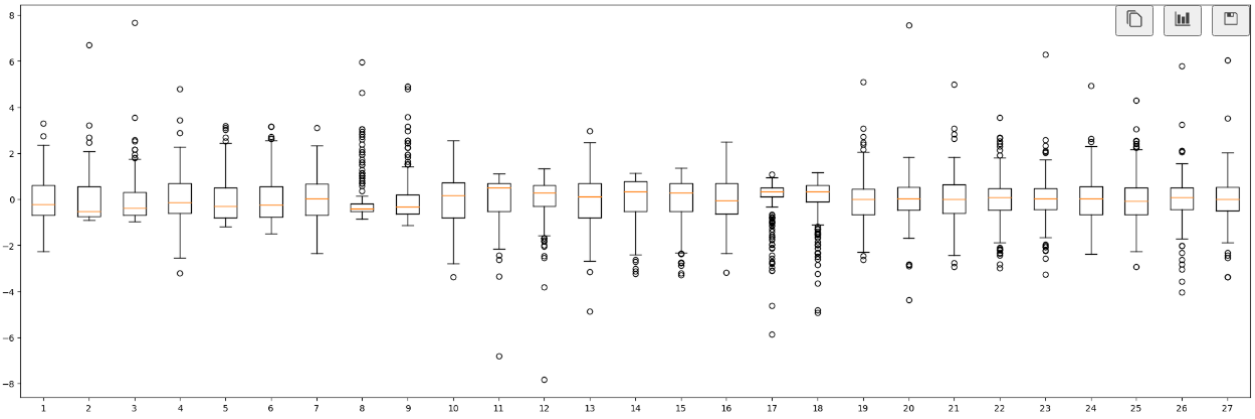


Figure 14: Post Standardization

3.2 Details, Evaluation and Comparison of the models

This section discusses both the details and the comparison for the models. For the process of classification the following multi class classifiers are chosen: Continuous K-Nearest Neighbor (CKNN), Random Forest Classifier and Support Vector Machines (SVM).

For choosing the hyper parameters for each of these models a grid search is done iterating over the validation dataset to find the best combination of parameters for each of the models.

Continuous K-Nearest Neighbor (CKNN)
A CKNN classifier can be trained using the 'KNeighborsClassifier' class in the sklearn package. The hyper parameters such as n-neighbors, weights and metric can be set for this classifier.

The n-neighbours changes the number of neighbours that will be considered in that instance i.e a numerical value is to be chosen. Generally a higher value of this will lead to poor performance but better decision and when this is low there is a possibility of being over fitted but a more complex decision. The weights parameter has the option of being 'uniform' which means all the neighbours are of equal weight or being set to 'distance' where the closer neighbours will be given a higher weight. The metric parameter, by default is set to 'minkowski' and can be switched to use 'Manhattan' or 'Euclidean' distance. This metric has a substantial influence on the performance of the model. Therefore, it necessitates additional investigation beyond the standard cross-validation grid search performed in the baseline evaluation.

To consider the best hyperparameters a grid search a cross-validation grid search is done with all the parameters on the validation dataset. The figure below shows the resulting values from the grid search that are the best hyper parameters. A euclidean distance with nearest neighbours as 1 and the weights to be uniform.
While the results of this grid search provides great performance as indicated by the confusion matrix the argument can be made that further investigation can be done and the choice for which hyperparameters to be left open to the user based on the application. The reason for this being the nearest neighbours should not be one as this can lead to over fitting, the model being highly sensitive to noise and leading to high variance. The metric finalized upon can also be criticized as the distance used has significant impact on the performance of the model.

```
Best hyperparameters: {'metric': 'euclidean', 'n_neighbors': 1, 'weights': 'uniform'}
              precision    recall  f1-score   support

           d       0.82      0.92      0.87        53
           h       0.62      0.87      0.72        15
           o       0.96      0.71      0.81        31
           s       0.88      0.81      0.84        62

    accuracy                           0.83       161
   macro avg       0.82      0.83      0.81       161
weighted avg       0.85      0.83      0.83       161
```

Figure 15: Best Hyper parameters for CKNN

The diagram below shows the confusion matrix for the validation data. The diagonal remains mostly yellow-light green which is a great indicator for the performance of the model and the off diagonal has a low score (purple) which indicates less incorrect predictions.

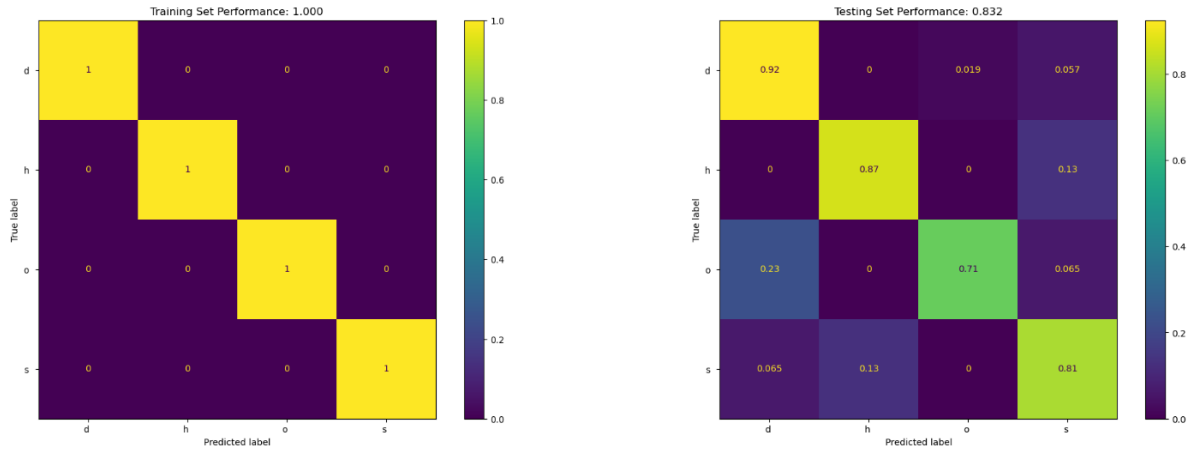


Figure 16: Confusion Matrix for chosen Hyper parameters

As discussed above a further investigation is done into the performance between the two possible choices for the distance metric. For Manhattan distance a non-one value for nearest neighbours seems to be 10 as indicated by the graph For the euclidean distance a value of 5 is more appropriate for nearest neighbours.

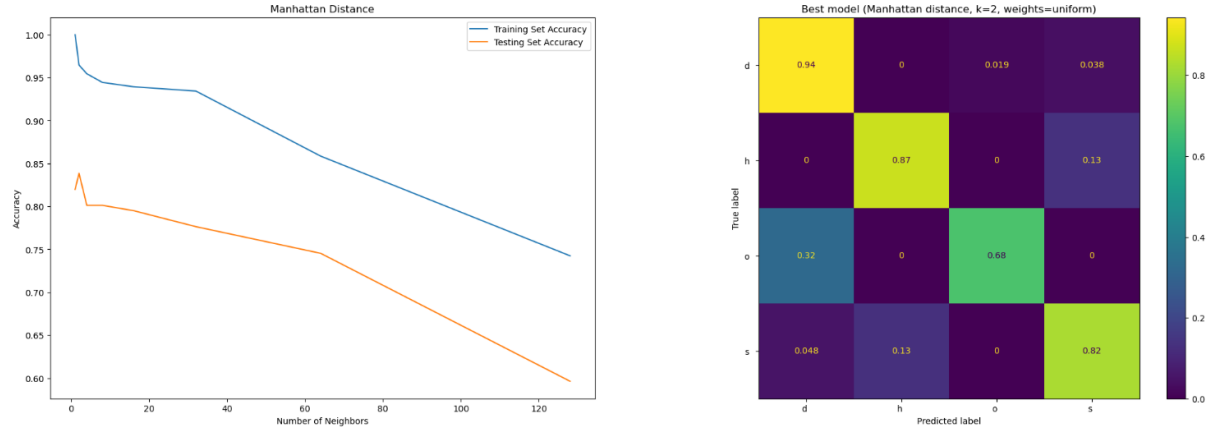


Figure 17: Best Manhattan for CKNN

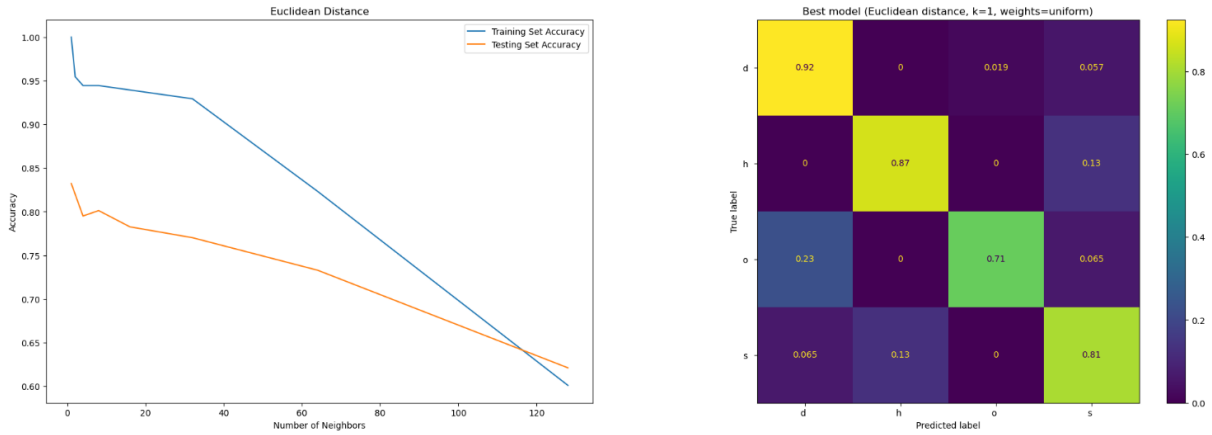


Figure 18: Best Euclidean for CKNN

Random Forest Classifier

A random forest classifier can be trained using the 'RandomForestClassifier'class in the sklearn package. The hyperparameters such as n-estimators, max-depth, random-state and class weights can be set for the classifier.

To start the investigating these hyper parameters can be adjusted by starting off with a simple model then making the trees deeper and adding class weights and then finding a balance between deeper trees and balanced weights.

A cross-validation is done across all these parameters by doing a grid search on the validation dataset to see which will give the best performance. Changing the n-estimators changes the number of decision trees in the forest - performance can be increased by increasing this number but can lead to increased computational cost with the downside of over fitting. The max-depth sets the depth for the decision tree - similar to the n-estimators increasing this will give a better performance but could lead to overfiting. The random-state is usually set to zero - this is more to set a specific state for the algorithm to ensure to get specific results each time. class weight can be adjusted to be balanced or to 'balanced-subsample' which will adjust the weights based on the classes to handle certain datasets.

```
Best hyperparameters:  {'n_estimators': 20, 'max_depth': 3, 'random_state': 5,
'class_weight': 'balanced'}
```

Figure 19: Best hyperparamters for random forest classifier

With the chosen hyperparameters the computation complexity isn't too heavy as both the n estimator and the max depth are of moderate size. The iteration size for the number of estimators were in range from 1 to 200 and the max depth from 3 to 9. The resulting cross validation grid search chose values in the lower end for both of these parameters which still resulted in a good confusion

	precision	recall	f1-score	support
d	0.80	0.77	0.79	53
h	0.68	0.87	0.76	15
o	0.92	0.77	0.84	31
s	0.82	0.85	0.83	62
accuracy			0.81	161
macro avg	0.81	0.82	0.81	161
weighted avg	0.82	0.81	0.81	161

Figure 20: Performance from best hyperparameters

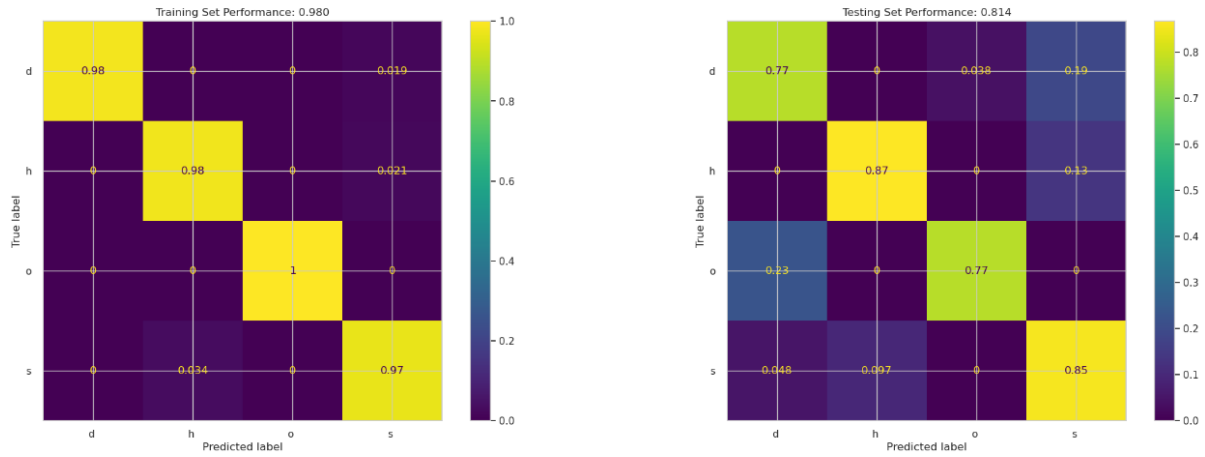


Figure 21: Confusion matrix for best hyperparamters

matrix on the testing set with mostly having a diagonal of yellow-green while the off diagonal is heavily towards purple.
Had these two parameters been extremely large the computation cost would be high and likely result in over fitting and the model would be sensitive which would cause a higher level of incorrect prediction making the off-diagonal a higher value.

Support Vector Machines (SVM)
A SVM classifier can be trained using the 'SVC' class in the sklearn package. There are two types of SVM: one vs one and one vs all classifiers and for each of these the hyper parameters can be adjusted to see which combination would yield the best result. As with the other classifiers a cross validation technique with a grid search is used to determine the best parameters.

```
One vs One SVM
              precision    recall  f1-score   support

           d       0.86      0.81      0.83        53
           h       0.62      0.87      0.72        15
           o       0.89      0.81      0.85        31
           s       0.84      0.84      0.84        62

    accuracy                           0.83       161
   macro avg       0.80      0.83      0.81       161
weighted avg       0.84      0.83      0.83       161

One vs All SVM
              precision    recall  f1-score   support

           d       0.88      0.81      0.84        53
           h       0.57      0.87      0.68        15
           o       0.92      0.77      0.84        31
           s       0.84      0.85      0.85        62

    accuracy                           0.83       161
   macro avg       0.80      0.83      0.80       161
weighted avg       0.84      0.83      0.83       161

Best hyperparameters: {'C': 10, 'gamma': 0.01, 'kernel': 'rbf'}
...
    accuracy                           0.86       161
   macro avg       0.83      0.87      0.84       161
weighted avg       0.88      0.86      0.86       161
```

Figure 22: Best hyperparameters for SVM

With the chosen hyper parameters the SVM is trained for both one v one and one vs all. The confusion matrices for these are given below.
As seen by the plots both models have a similar performance on the test set. The difference on a one vs one and a one vs all is that the former is used for small to medium sized datasets while the later works better for larger data sets where it trains for each cass agaisnt all the other classes.

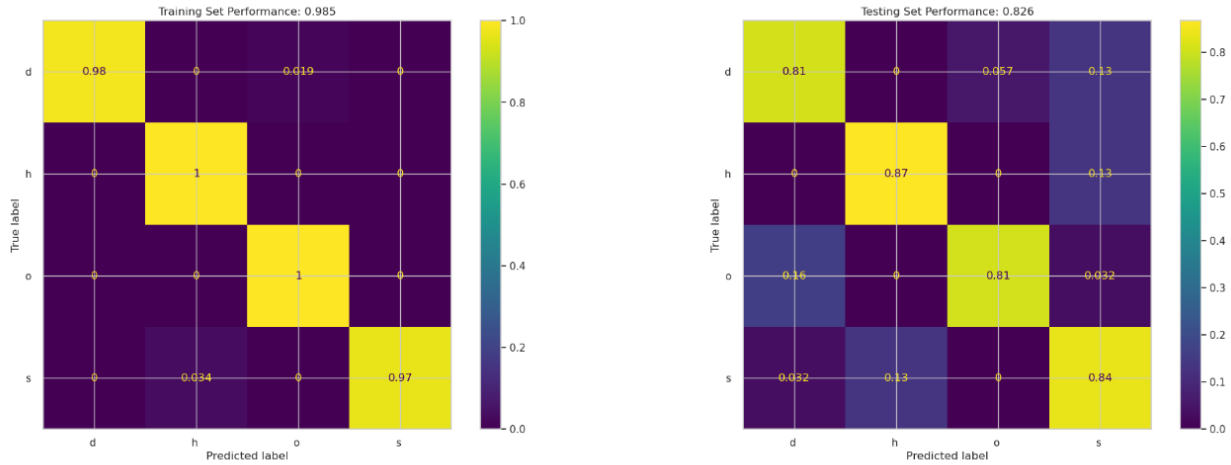


Figure 23: Confusion matrix for one v one SVM

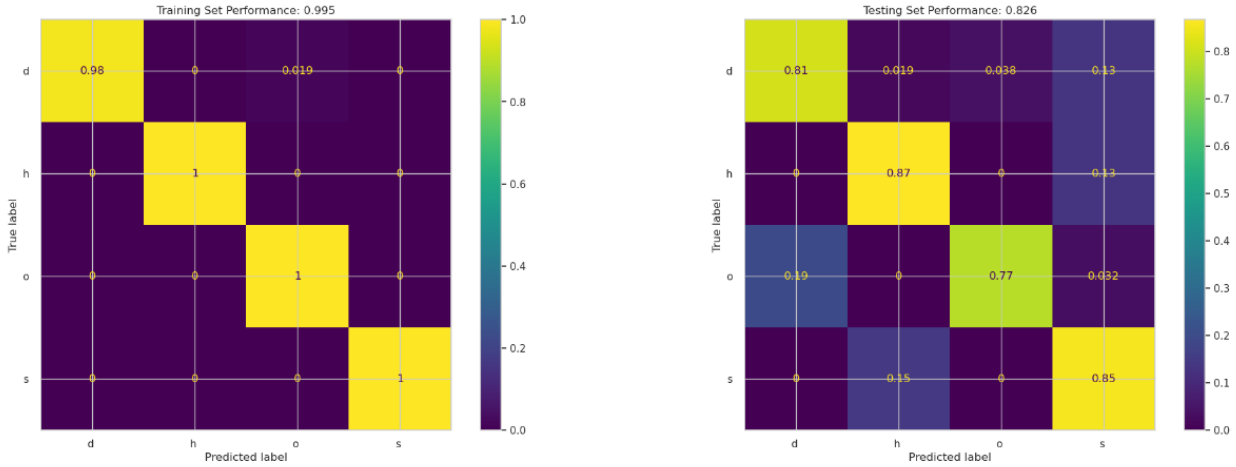


Figure 24: Confusion matrix for one v all SVM

4 Training and Adapting Deep Networks

4.1 Discussion of neural network design

For the design of the neural network there is a choice of using convolution layers, activation layer and max pooling layers. For this neural network the model definition has been given below. it involves convolution layers which perform the convolution operation on the input image using multiple filters which learn specific features of the input image. Followed by an activation layer which allows the network to learn more complex representations of the input image. Finally max pooling layers to perform down sampling of the features obtained from the previous convolution layers. These are stacked in the order of multiple convolution and activation followed by max pooling. These layers are then repeated.

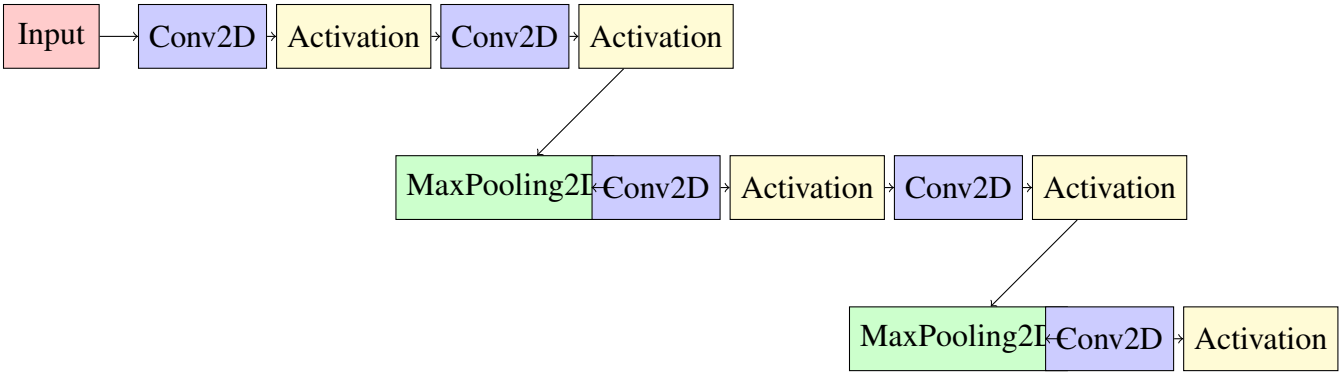


Figure 25: Convolution Neural Network Architecture.

```
Model: "CNN"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 img (InputLayer)            [(None, 32, 32, 3)]       0

 conv2d_72 (Conv2D)          (None, 32, 32, 8)         224

 activation_54 (Activation)  (None, 32, 32, 8)         0

 conv2d_73 (Conv2D)          (None, 32, 32, 8)         584

 activation_55 (Activation)  (None, 32, 32, 8)         0

 max_pooling2d_48 (MaxPoolin  (None, 16, 16, 8)        0
 g2D)

 conv2d_74 (Conv2D)          (None, 16, 16, 16)        1168

 activation_56 (Activation)  (None, 16, 16, 16)        0

 conv2d_75 (Conv2D)          (None, 16, 16, 16)        2320

 activation_57 (Activation)  (None, 16, 16, 16)        0

...
Total params: 285,970
Trainable params: 285,970
Non-trainable params: 0
_________________________________________________________________
```

Figure 26: CNN architecture

The image above shows the architecture used for the CNN. The figure below shows the neural network during the training stage. The image depicted below is indicative of bad performance as the loss keeps increasing and the accuracy keeps decreasing.

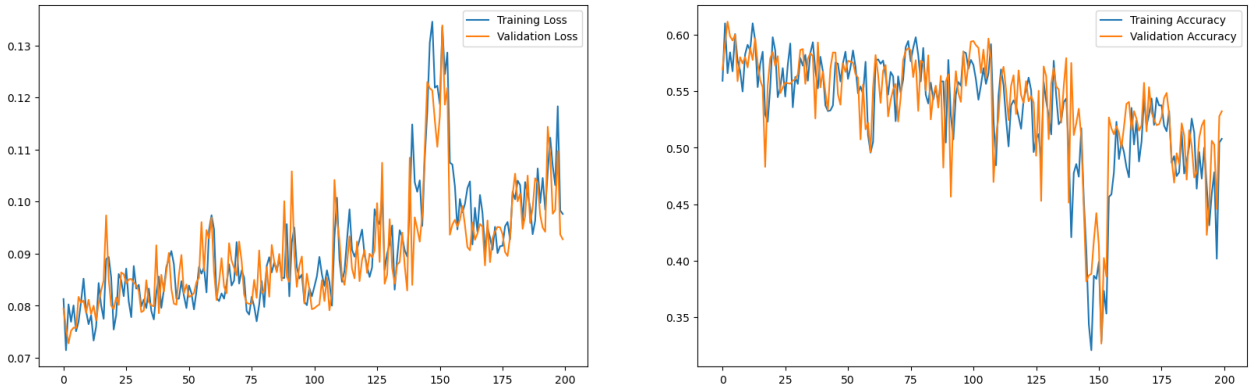


Figure 27: Failed Neural Network during training

4.2 Discussion of data augmentations

The DCNNs are trained with data augmentation and one without. Data augmentation a technique performed on the images to increase the size of the data set so that there is more training data. Conventionally adding data augmentation improves the performance of the model during testing.

Common ways of augmenting data is to flip the images, perform rotations, zooming, cropping and shifting the images. For this neural network a rotation is done for 5 degrees, a horizontal shift by 5 percent, vertical shift by 5 percent, zoom by 0.1 and horizontal flip.

4.3 Comparison between the two DCNNs and SVM

The three models being compare are a SVN model, DCNN models without data augmentation and one with data augmentation. They are judged on their performance and the time taken for training/inference.

In terms of time taken, as seen by the figure for the SVN it is clearly much faster for training and testing time coming under 20 seconds while the DCNN with augmentation too under 500 seconds and the DCNN with augmentation took under 1500 seconds.

”Time to train and evaluate without augmentation: 1521.031299”
”Time to train and evaluate with augmentation: 418.968524”

Based on the f1-score across the models the best to worst are as follows: DCNN with data augmentation, DCNN without data augmentation and the SVM.
This performance aligns with the assumptions about neural networks as a model that has been given data augmentation has a larger dataset and would therefore perform better.

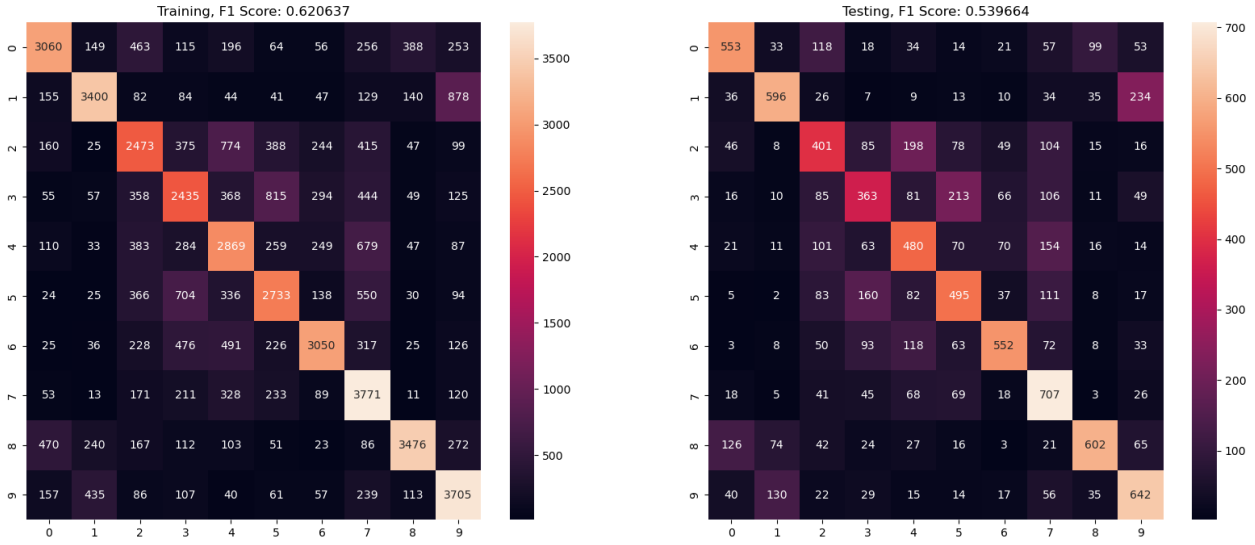


Figure 28: Confusion Matrix for DCNN without augmentation

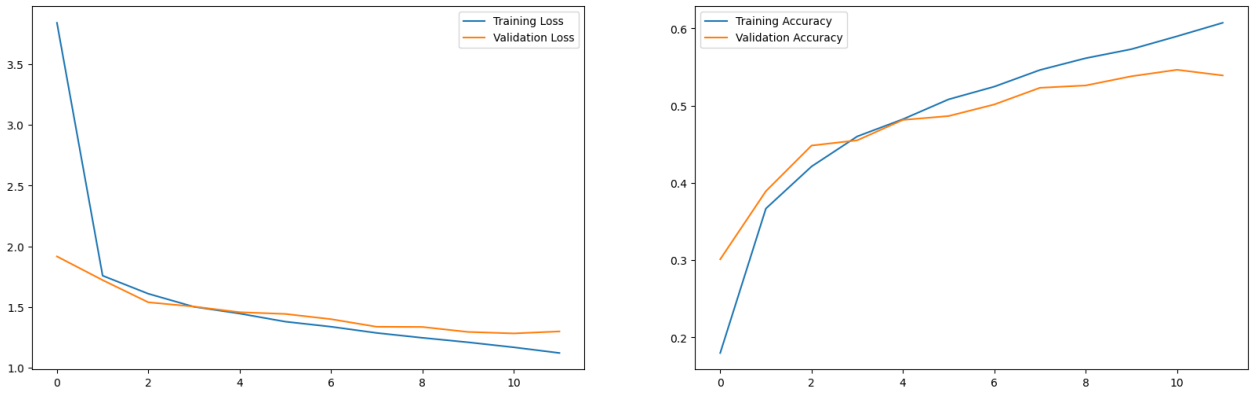


Figure 29: Accuracy and Loss Plots without augmentation

Given above is the confusion matrix and plots for the DCNN without augmentation. The diagonal of the testing and training stay in a relatively good prediction range as they are a creamish-white color and the off diagonal being purple giving a low number of incorrect predictions.
The epochs are kept to a low value to avoid over fitting while the batch size is increased to ensure convergence. The plots show that the model converges and after doing so continue down in the loss graph and in the accuracy graph the line moves upwards showing the increasing accuracy of the model. All of these factors show the model performing well.

Figure 30: Confusion Matrix for DCNN with augmentation

Figure 31: Accuracy and Loss Plots with augmentation

Given above is the confusion matrix and plots for the DCNN with augmentation. The diagonal of the testing and training stay in a relatively good prediction range as they are a creamish-white color and the off diagonal being purple giving a low number of incorrect predictions.
The epochs are have been increased to a large value along with steps for each epoch, the batch size was also increased to test the limitations and capabilities of the model. The plots show that the model converges and afterwards moves downwards as an overall direction. The accuracy graph shows the increasing accuracy of the model. All of these factors show the model performing well.

```
Training Time: 1.455758
Inference Time (training set): 0.636705
Inference Time (testing set): 5.653699
              precision    recall  f1-score   support

           0       0.22      0.30      0.25       711
           1       0.37      0.50      0.43      1894
           2       0.39      0.36      0.37      1497
           3       0.36      0.33      0.34      1141
           4       0.34      0.31      0.32      1035
           5       0.35      0.32      0.34       892
           6       0.32      0.24      0.28       758
           7       0.38      0.30      0.34       789
           8       0.32      0.30      0.31       683
           9       0.38      0.29      0.33       600

    accuracy                           0.35     10000
   macro avg       0.34      0.33      0.33     10000
weighted avg       0.35      0.35      0.35     10000

Evaluation Time: 19.395764
```

Figure 32: Information about the SVM

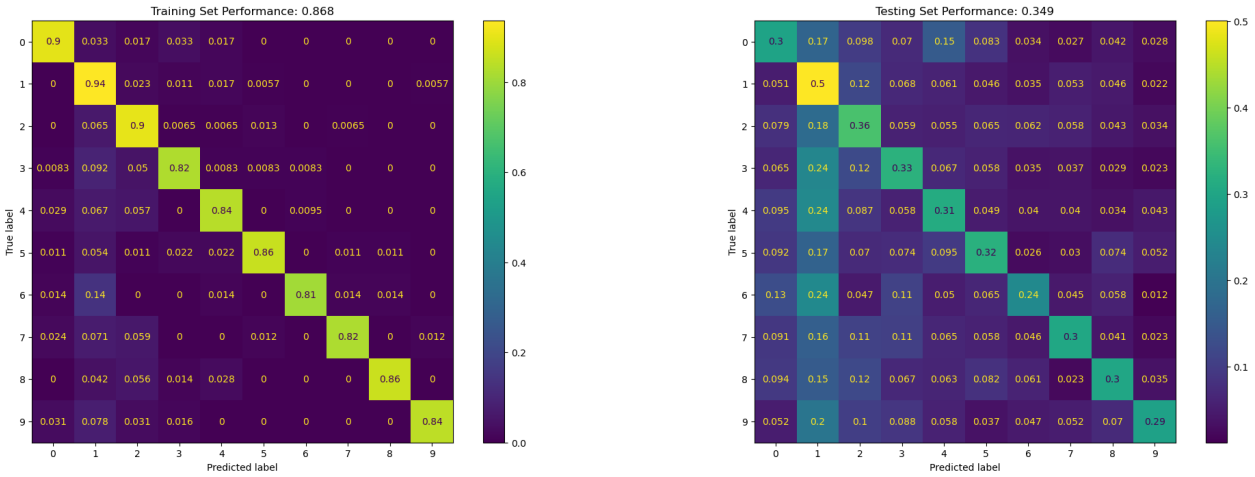


Figure 33: Confusion matrix for SVM

Given above is the info for the SVM. Given that this didn't undergo a complex training and convolution layers it is understand that it has a f1-score less than the other models. The diagonal and the cross diagonal still have good performance as they are relatively yellow and a darker shade of green on the test set.

References

Builtin (2022). Understanding of lasso regression. great learning blog. https://www.mygreatlearning.com/blog/understanding-of-lasso-regression/. Accessed: April 6, 2023.

Builtin (2023). When and why to standardize your data. built in. https://builtin.com/data-science/when-and-why-standardize-your-data. Accessed: April 6, 2023.

V7Labs (2023). Data preprocessing guide. in v7labs. https://www.v7labs.com/blog/data-preprocessing-guide#h1. Accessed: April 6, 2023.

Appendix A: Jupyter Notebook

Notebooks not attached as report is substantially longer than the required number of pages. The code if needed can be found at https://github.com/DonMMK/Machine-Learning-CAB420