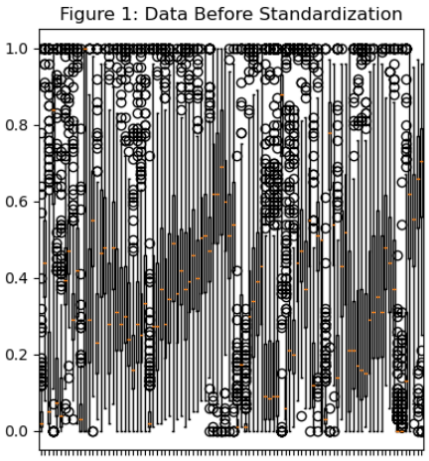
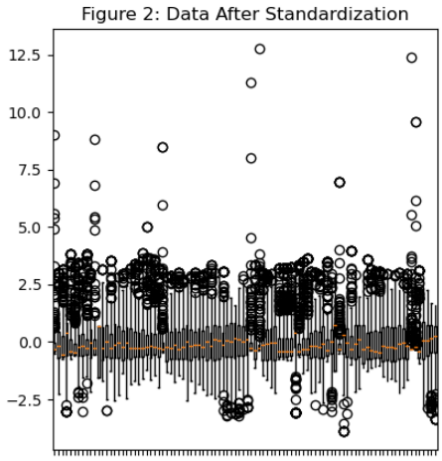
Problem 1. Regression

1.1 Discussion and justification of any pre-processing performed.

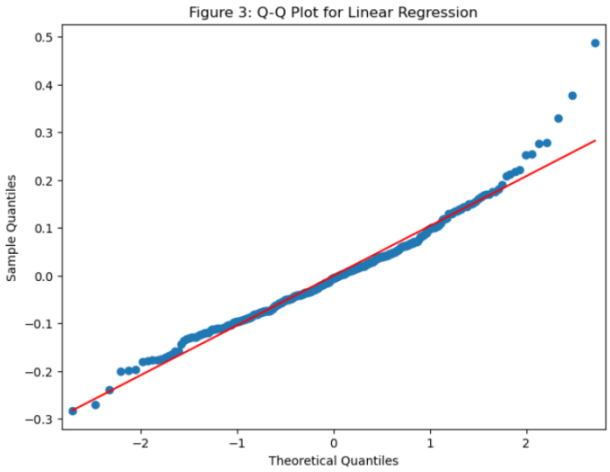
The dataset is already normalised, as shown in Figure 1, with a range between 0 and 1. Applying standardisation to the data would expand the range significantly to more than -2.5 and 12.5, as seen in Figure 2. Therefore, it was decided to use the provided normalisation and not apply standardisation. Standardisation is typically employed when measurements have disparate scales, but in this case, it would only increase the range and potentially cause overfitting.





1.2 Details of the three trained models.

Figure 3 displays a Q-Q plot representing the results of linear regression on the data, revealing significant outliers in the model. As the theoretical quantiles increase, more sample quantiles fall above the red line, indicating that some outlier data points do not conform to the normal distribution. The model is highly significant ($p < 0.001$) with an R-squared value of 0.759, explaining 75.9% of the variation. The adjusted R-squared value is 0.637. The training data has a very low RMSE of 0.1042 and the validation data has an RMSE of 0. 1549.



The Ridge model (Figure 4) was trained using a simple loop over a pre-defined range of alpha values, gradually narrowing down the value that controls the strength of regularisation. As the alpha values increase, the RMSE on the validation set also increases, indicating that the model becomes more biased and performs worse on unseen data. The optimal alpha value of 0.73 was selected based on the minimum RMSE obtained on the validation set. At this alpha value, the Ridge model achieved an RMSE of 0.1421, an adjusted R-squared of 0.5186 and an R-squared value of 0.6806.

Figure 4: Ridge Alpha VS RMSE

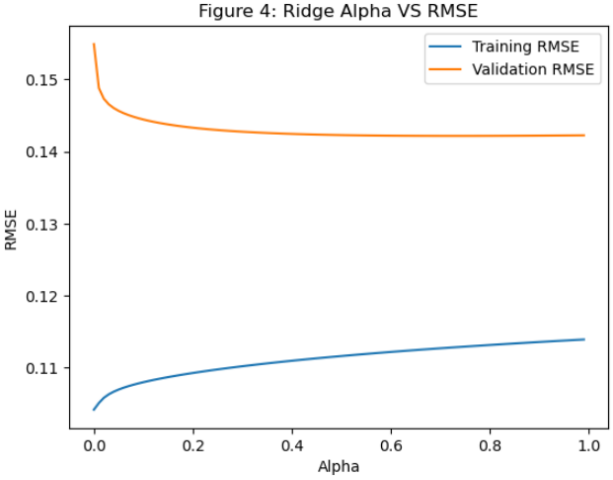
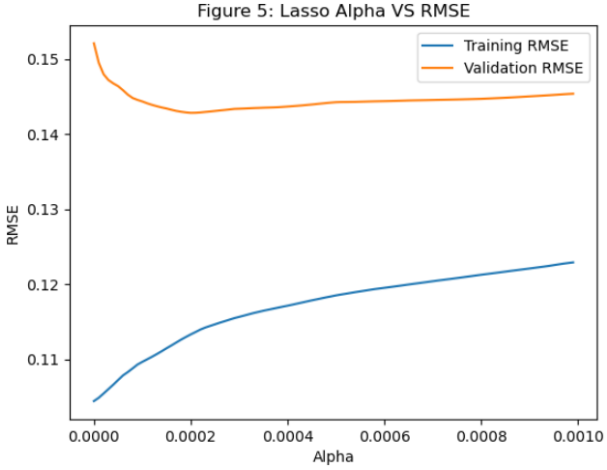


Figure 5: Lasso Alpha VS RMSE



The Lasso model (Figure 5) was also trained using a simple loop over a pre-defined range of alpha values. The RMSE on the validation set first decreases and then increases again as alpha increases, indicating that there may be an optimal value of alpha that balances the trade-off between overfitting and underfitting the model. The optimal alpha value for the Lasso model was found to be 0.0002, which achieved an RMSE of 0.1421, R-squared value of 0.6778 and Adjusted R squared of 0.5145. The Lasso path occurs because as the regularisation penalty becomes stronger, some of the coefficients in the model are shrunk to zero, leading to a sparser model.

1.3 An evaluation comparing the three models.

Model	Best Alpha	Train RMSE	Validation RMSE	R-Squared	Adjusted R-Squared
Linear Regression	N/A	0.1042	0.1549	0.759	0.637
Ridge	0.73	0.1129	0.1422	0.6807	0.5186
Lasso	0.0002	0.1133	0.1428	0.6778	0.5142

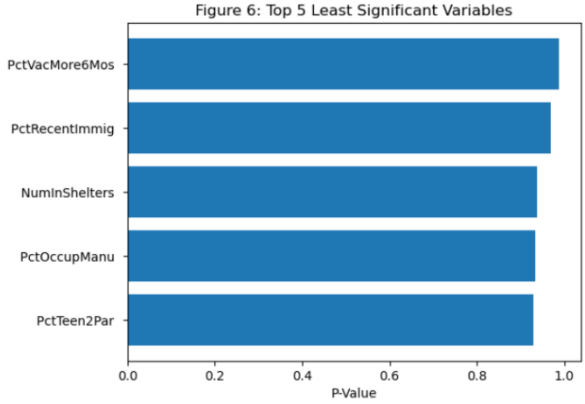
Table 1: Comparison of Regression Models on Crime Analysis Data

Table 1 shows the results of the three models: Linear Regression, Ridge, and Lasso. Based on the values, the following can be concluded:

- The linear regression model has an extremely low RMSE of 0.1042 for the training data and 0.1549 for the validation data. However, the model is likely to overfit and may not be able to generalise well on unseen data.
- The Ridge model may be more effective than linear regression in avoiding overfitting due to regularisation, but the high RMSE suggests that it may still not generalise well on unseen data.
- The Lasso model has the lowest R-squared and Adjusted R-squared, indicating that it may not capture all the significant predictors of the target variable and may perform worse in predicting new and unseen data.

Figure 6 shows the top five least significant variables that have little impact on 'ViolentCrimesPerPop' due to their weak linear relationship with the target variable. This helps identify irrelevant variables that can be removed to simplify the model and reduce the risk of overfitting.

Figure 6: Top 5 Least Significant Variables



The residual plot for the unregularised model in Figure 7 revealed the presence of significant outliers in the dataset, which violated the assumption of linearity. Moreover, the spread of the residuals was non-constant, indicating heteroscedasticity. Figure 8 illustrates that some of the outliers corresponded to a true count of 0. Since these values were of no use for prediction purposes, removing them may ameliorate the issues observed. As expected, the presence of serious outliers further undermined the validity of the model.

Figure 7: Residual plot for unregularised model

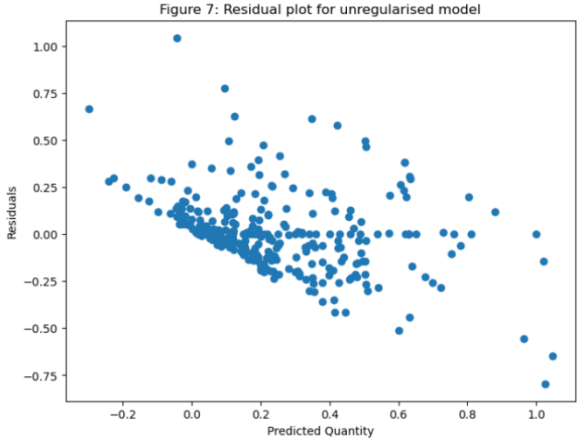
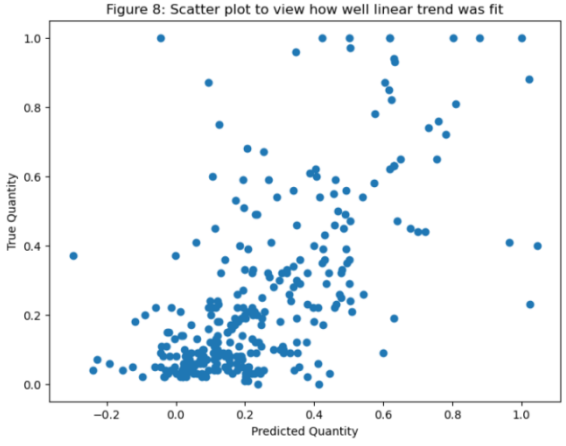


Figure 8: Scatter plot to view how well linear trend was fit

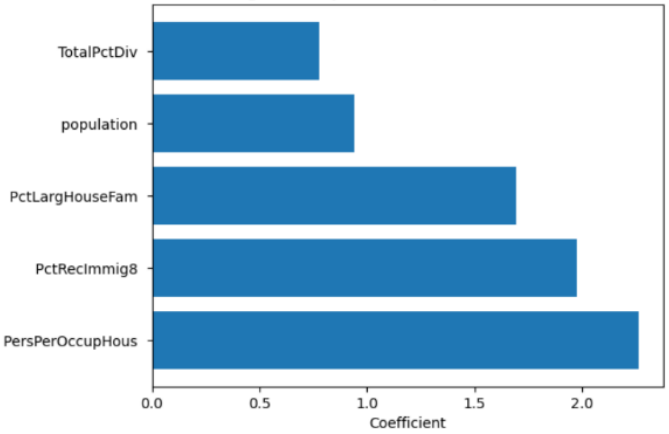


The Ridge and Lasso models can be better at avoiding overfitting than linear regression due to regularisation, but they may still not generalise well on new data. The Lasso model may be better at reducing irrelevant predictors, and therefore, it can be chosen for prediction. However, the accuracy and validity of the models can be influenced by socio-economic factors that vary across communities, which can impact the model's performance. It is essential to consider these underlying factors in addition to the model's accuracy and validity. The use of certain terms in the model can reflect socio-economic factors not captured, and the data may have multicollinearity, which can also impact the model's performance.

1.4 Ethical Concerns.

Predictive models for crime analysis raise ethical concerns that must be carefully considered, including potential biases in the data used to train the models, sensitive personal information use, and potential consequences for decision-making. The top 5 most important variables in Figure 9 suggest that population, immigrants, and divorced individuals are relevant to the number of violent crimes per capita. However, ethical principles of fairness, accountability, and sensitivity of variables must be considered to develop inclusive and transparent models that do not perpetuate discrimination or negative impacts on individuals or communities.

Figure 9: Top 5 Most Important Variables



Problem 2. Classification

2.1 Discussion and justification of any pre-processing performed.

In this project, standardisation was chosen as a pre-processing step for the data. Standardisation scales the features to a comparable range, which can help prevent certain features from dominating others during the training process. Algorithms such as CKNN, RF and SVM are particularly sensitive to the scale of the input features, and therefore, standardisation was chosen as a pre-processing step in this project. Figure 1 shows the distribution of the raw training data, which has a range of values between 200 and -100. However, after applying standardisation (Figure 2), the range of values has been narrowed down to between 8 and -8, indicating that the features are now in a more comparable range.

Figure 1: Raw Train Data

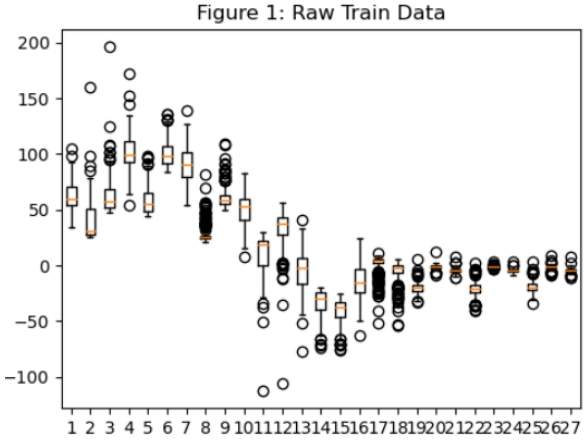
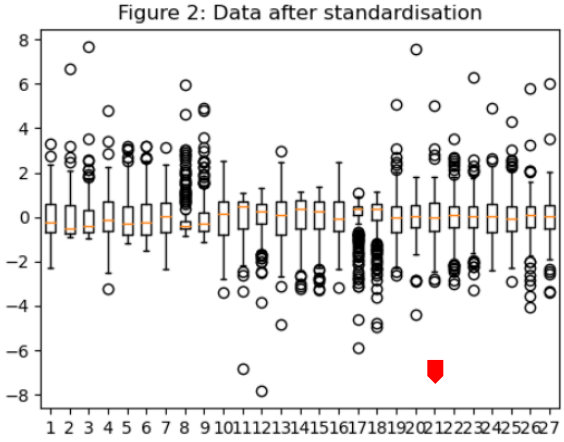


Figure 2: Data after standardisation



2.2 Details of the hyper-parameter selection method

2.2.1 K-Nearest Neighbours Classifier

The CKNN algorithm's hyperparameters were optimised using a grid search method, with a focus on three parameters: neighbours, metrics, and weights. Despite choosing the optimal number of neighbours, 198, the dataset was not suitable for the CKNN model, resulting in poor performance on the validation set (f1_score = 0.2805). A small value of K may lead to overfitting, while a larger K may lead to underfitting. The manhattan distance metric was chosen to handle outliers, which gives greater weight to nearby neighbours. However, these hyperparameters did not improve the model's performance. In summary, the CKNN model was not ideal for the dataset provided, and the chosen hyperparameters did not significantly impact the model's performance.

2.2.2 Random Forest

The Random Forest Classifier has two main hyperparameters: N (n_estimators) and Depth (max_depth) Class (class_weight). The N parameter specifies the number of trees in the forest, while Depth represents the maximum depth of the tree and Class can be used to assign different weights to classes. The nested loop searched over various values of these hyperparameters and identified the combination of hyperparameters that resulted in the highest f1_score on the validation set. In this case, the best-performing hyperparameters were N=28, Depth=1, and Class=None, which achieved an f1_score of 0.456. A higher number of N can increase the randomness and diversity in the forest, reducing the risk of overfitting but also increasing computational costs. Additionally, a higher number of Depth values can lead to better performance but increases the risk of overfitting, while a smaller number of values can lead to underfitting.

2.2.3 Ensemble of Support Vector Machines

The SVM model was trained using a grid search to determine the best hyperparameters, including C, gamma, kernel, ens_type, and class_weights. However, the dataset provided was not optimal for SVM, as evidenced by the always-selected highest value of C (1000000000) indicating a potential for overfitting. To improve performance and better balance the trade-off between training and validation error, additional

hyperparameters were explored. The final model parameters were C=1000000000, gamma=0.0001, kernel=linear, balanced, and OneVsOne, resulting in the highest f1_score on the validation set (0.606). A smaller gamma value increases regularisation strength, simplifying the decision boundary and reducing overfitting, while larger gamma values lead to more complex boundaries and increased overfitting. The C parameter controls the trade-off between low validation and low testing errors.

2.3 An evaluation and comparison of the final three models

Model	Execution Time
CKNN	0.2758 seconds
RF	3.1412 seconds
SVM	6.063 seconds

Table 1: Model execution time

Model	Precision	Recall	F1-Score
CKNN	0.15	0.39	0.21
RF	0.53	0.52	0.50
SVM	0.67	0.63	0.62

Table 2: Model Evaluation Results

In Table 1, the CKNN model had the shortest execution time followed by the RF and SVM models, with the latter having the longest execution time due to the nested loop grid search. However, the evaluation of the models in Table 2 suggests that the SVM model outperforms the other two models in terms of precision, recall, and F1-score. The RF model appeared to overfit the training data, as indicated by its higher performance on the training set than the testing set.

Figures 3 (CKNN), 4 (RF) and 5 (SVM) reveal that all models did not correctly predict any instances of this class h. The CKNN model had poor performance across all classes, with precision, recall, and F1-score of 0 for classes 'd', 'h', and 'o'. The RF model performed relatively well on the 'o' class but had low precision for the 'd' class and was unable to predict any instances of the 'h' class. The SVM model had high precision for the 's' class but had low precision for the 'o' class and was unable to predict any instances of the 'h' class.

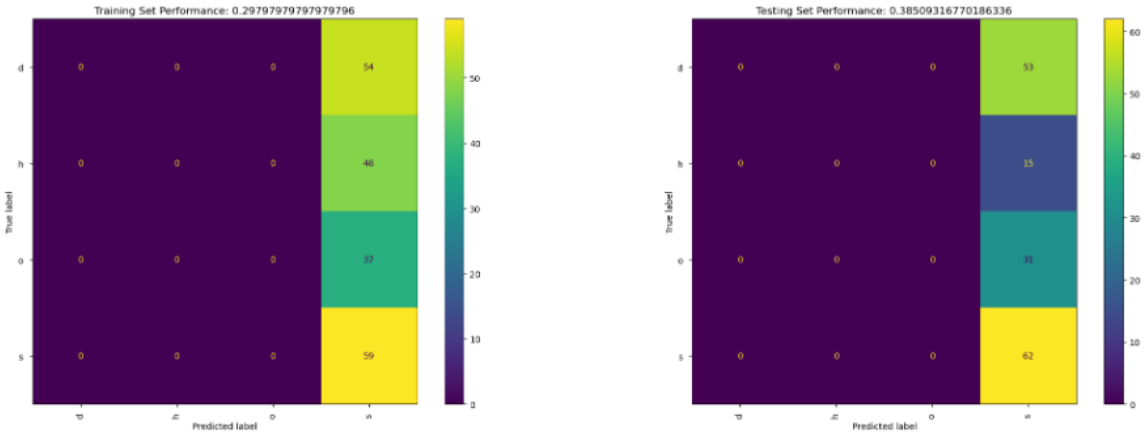


Figure 3: CKNN Confusion matrix

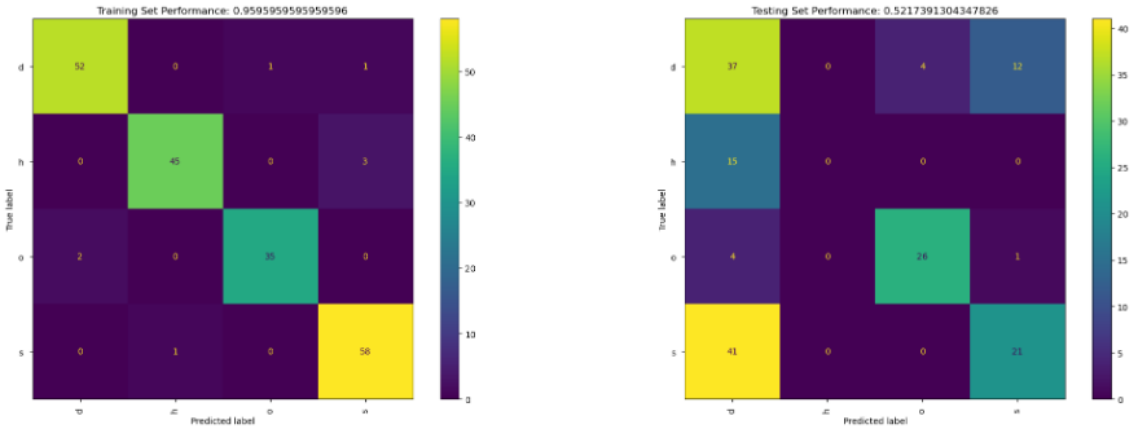


Figure 4: Random Forest Confusion matrix

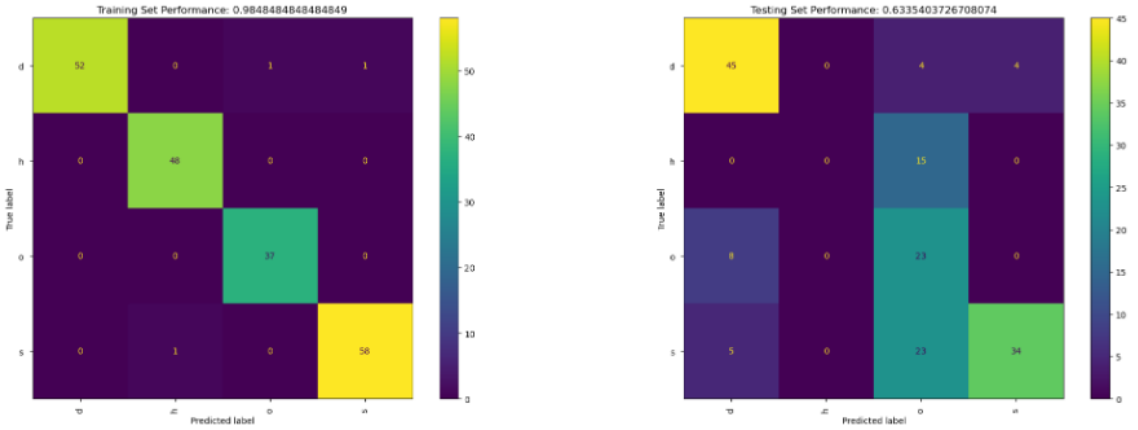


Figure 5: SVM Confusion matrix

It is important to note that the class distribution in the testing set was different from that of the training set, which may have contributed to the lower performance on the testing set. The imbalance in the distribution of classes (Figure 6), particularly for the minority classes 'h' and 'o', may have biased the predictions and led to lower accuracy, precision, and recall values for these classes.

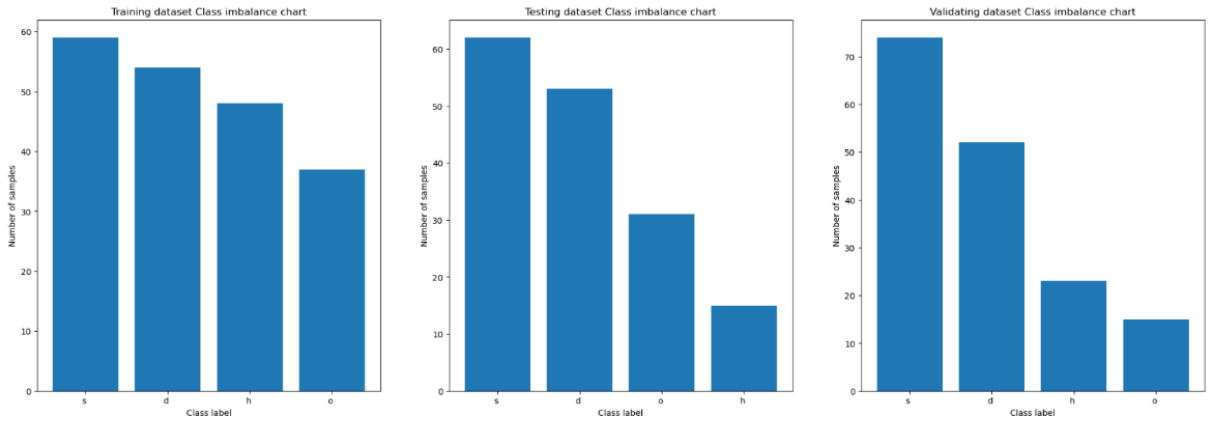


Figure 6: Class imbalance chart

In conclusion, while the SVM model had the highest overall performance, all models struggled to predict instances of the minority class 'h'. Both the SVM and CKNN models consistently selected the highest value of their respective hyperparameters, C and the number of neighbours, during our experiments. This suggests that further data collection and pre-processing may be necessary to improve the models' performance. The difference in performance between the models may be due to their respective algorithms and hyperparameter settings, as well as the characteristics of the data.

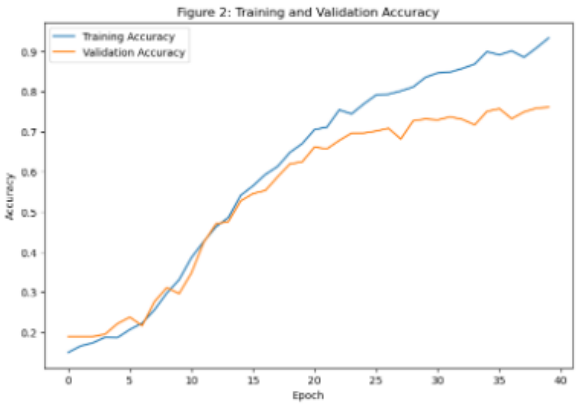
Problem 3. Training and Adapting Deep Networks

1.0 Discussion of neural network design

The SVHN dataset with only 1000 training samples, designing a very deep architecture could result in overfitting. Therefore, a modified AlexNet architecture with three convolutional blocks and increasing filter sizes (16, 32, 64) was chosen. The model also includes a flatten layer, two dense layers (256 and 10 neurons), and a dropout layer, totalling 298,666 trainable parameters. The DCNN model was trained on the given dataset for 40 epochs with a batch size of 128. The model was trained using the Adam optimiser with a learning rate of 0.001. The ReLU activation function was used in all layers except for the output layer, which used a softmax activation function. As shown in Table 1, the model achieved a training time of 19.37 seconds and inference times of 0.14 seconds on the training set and 1.05 seconds on the testing set. These results examined that the selected architecture and hyperparameters were appropriate for the task at hand and that the model was trained efficiently on the given dataset without significant computational constraints.

Type of Time Measurement	Time Taken (in seconds)
DCNN Training Time	19.366131
DCNN Inference Time (training set)	0.145057
DCNN Inference Time (testing set)	1.050770

Table 1 DCNN time measurements without augmentation



However, overfitting was observed in the validation metrics, as shown in Figures 1 and 2, where the validation accuracy plateaued and the validation loss increased after approximately 21 epochs. Overfitting was also evident in Figure 3, with the training F1 score of 0.9699 and testing F1 score of 0.7799. Although increasing the number of epochs could have allowed the model to converge, the chosen batch size and epochs appear sufficient to train the model efficiently on the given dataset. The designed DCNN was not constrained by limited computational resources as the QUT Jupyter Lab provided 8GB of memory.

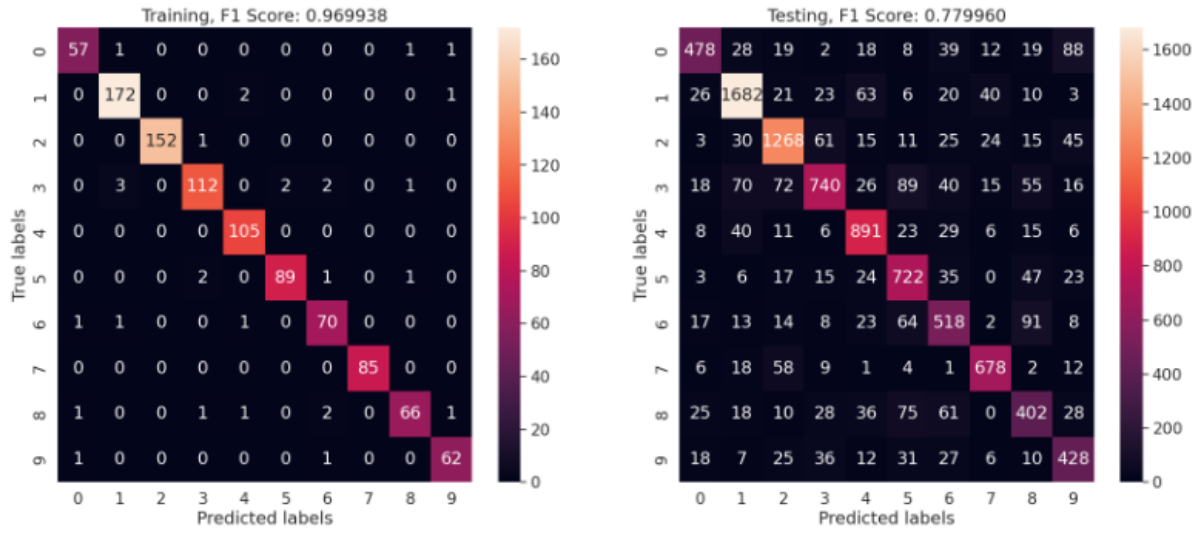


Figure 3: With data augmentation confusion matrix

2.0 Discussion of the data augmentations used

Since the images in this project only contain numbers, rotating them by up to 15 degrees can provide additional variability without significantly distorting the images. Randomly shifting the images horizontally and vertically by up to 5% of their respective dimensions, randomly flipping the images horizontally (since single-digit numbers are symmetrical), and randomly zooming the images by up to 5% can also add variability. Each image can be augmented in 21 different ways (5 degrees rotation, 2 horizontal flips, and 3 different types of shifts/zooms), resulting in a dataset of 21,000 images after augmentation. As shown in Figure 4, the image before and after augmentation can be compared to see the effect of the augmentations.

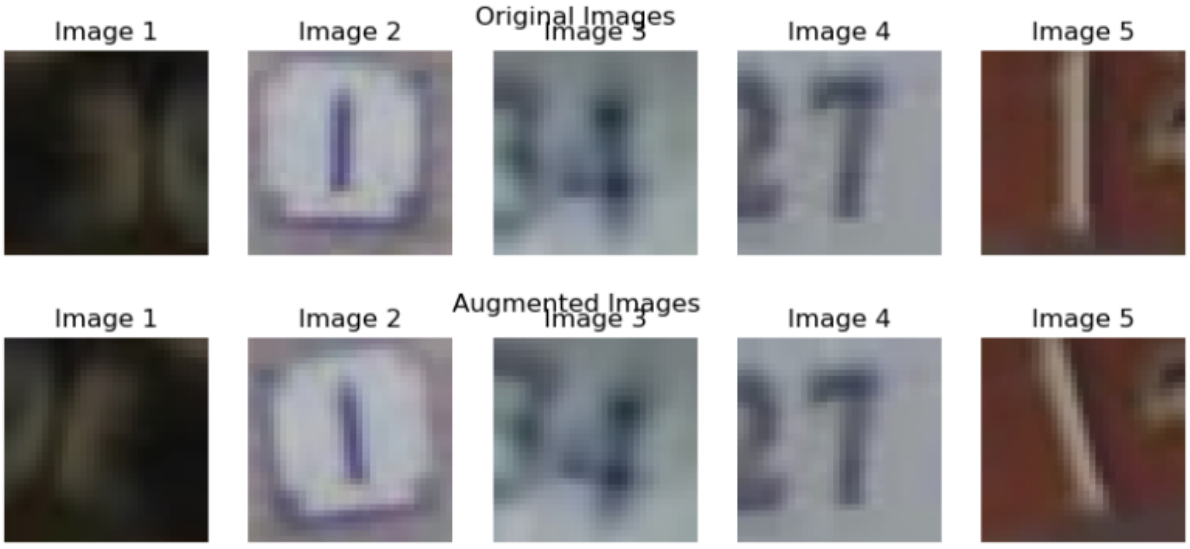


Figure 4: Before and after augmentation

3.0 Comparison between the two DCNNs (with and without augmentation) and the SVM

The model was trained for 250 epochs using a batch size of 512, and to prevent testing with the parameters learned from training, it was reinitialised. The results presented in Figures 5 and 6 show that while the training accuracy increased over time, the validation accuracy remained stagnant or fluctuated. Moreover, in epochs 40+, the loss increased by almost 0.25, while the accuracy dropped by 0.1. This observation could indicate overfitting of the model to the training data, leading to poor generalisation ability. However, the quick

recovery of accuracy and loss within a few epochs suggests that the model may be able to correct itself and learn to generalise better.

Figure 7 shows that the F1 score for the training set is 0.943, while it is only 0.7404 for the testing set, indicating overfitting. The misclassification of true labels '5' to '3' and '4' to '1' further highlights this observation, suggesting that the model is memorising specific examples from the training data instead of generalising well.

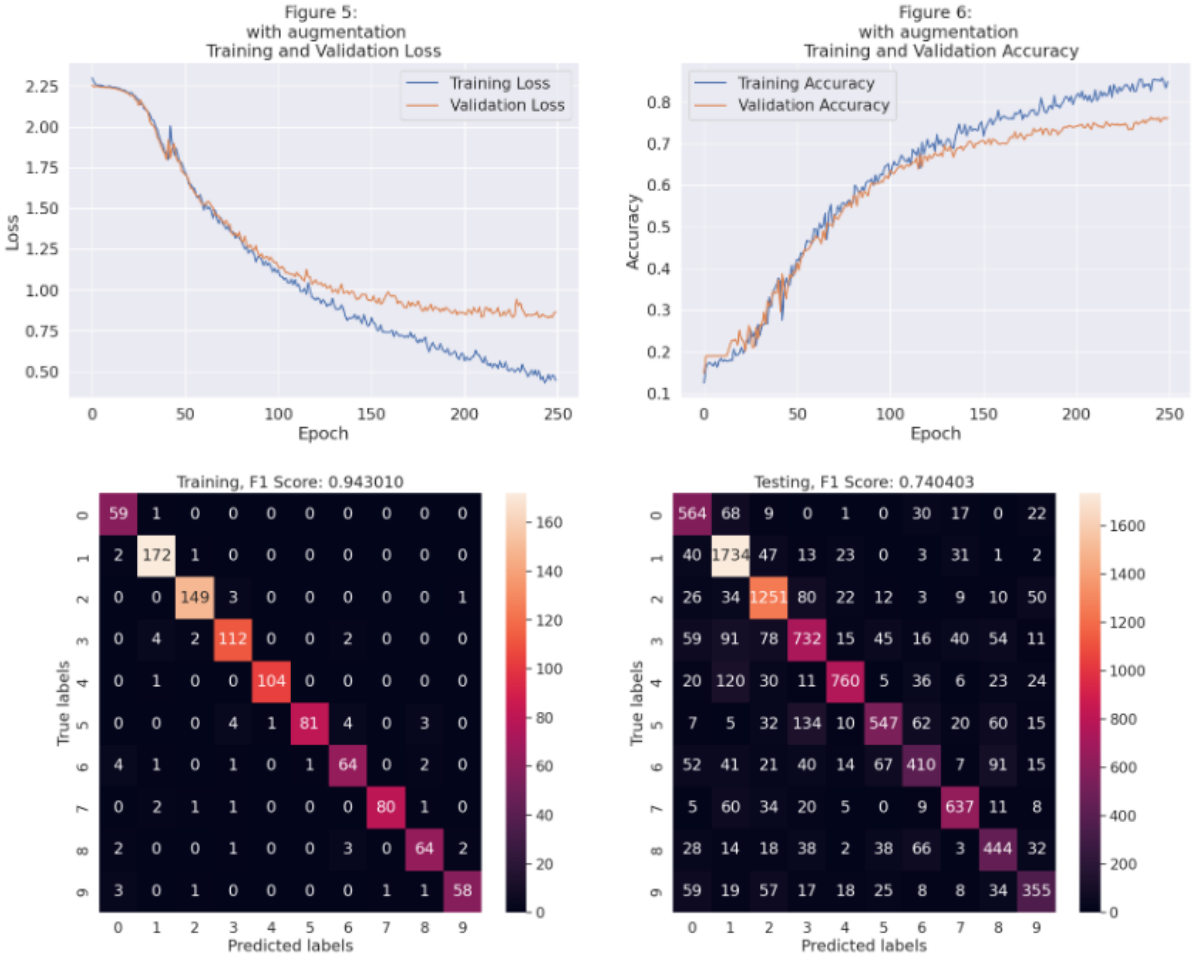
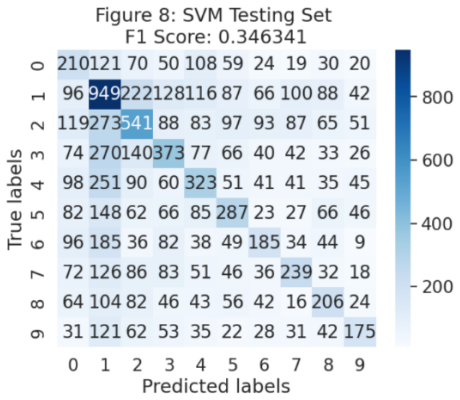


Figure 7: With data augmentation confusion matrix

In addition, when considering real-time applications, inference time is an important factor. The SVM (Figure 8) had the lowest training time, but its high misclassification of labels 0, 1, and 2 resulted in lower F1 scores than the DCNN models. Furthermore, it had a much higher inference time compared to both DCNN models. The DCNNs with and without augmentation had similar inference times, and they were much faster than the SVM.



Description	Without augmentation	With augmentation	SVM
Training Time	19.366131	292.060317	3.451244
Inference Time (training)	0.145057	0.212568	1.477223
Inference Time (testing)	1.050770	1.046296	14.825623
Training F1 Score	0.96999	0.94301	0.869683
Testing F1 Score	0.77996	0.740403	0.346341

Table 2: Performance and Time Metrics for DCNN Models and SVM

Table 2 presents the performance and time metrics for the DCNN models and the SVM. The DCNN without augmentation achieved the highest training and testing F1 scores of 0.97 and 0.78, respectively. However, it took significantly longer to train than the DCNN with augmentation and the SVM. The DCNN with augmentation achieved a training F1 score of 0.93 and a testing F1 score of 0.74, while training for significantly longer than the DCNN without augmentation, but still much less than the SVM.

In terms of inference time, the DCNNs with and without augmentation had similar times, taking only slightly over one second for both training and testing. The SVM had the highest inference time, taking over 14 seconds for testing.

Based on these results, it can be concluded that the DCNN with augmentation is better suited for real-time applications due to its better generalisation and faster inference time, despite having slightly lower F1 scores compared to the DCNN without augmentation.