CAB420 - Assignment 1A

Callum McNeilage - n10482652

Connor Smith - n9991051

Queensland University of Technology

Problem 1: Regression

How the Data is Handled

For this problem, we were given a .csv file named ‘communities.csv’ which contained socio-economic data from 1990 US census for various US communities, and the number of violent crimes per captia. The first task was to Split the data contained in ‘communities.csv’ into training, validation and testing sets. Initially, we removed the first five columns (state, county, community, communityname string, and fold) as per the task description. In addition we converted all non-values (“?”) in the dataset to NaN values in order to remove them from the dataset as well.

```
data_to_use = data.iloc[:,5:]
data_to_use[data_to_use == "?"] = np.nan # change all "?" to NaN
data_to_use = data_to_use.dropna()
```

In order to split the cleaned data set, we used a 70%, 15%, 15% split between training, testing and validation sets respectively. This was done sequentially, as per the data splitting in Week 2 Example 1 Regularised Regression.

```
num_samples = data_to_use.shape[0]
training_samples = int(num_samples*0.7)
validation_samples = int(num_samples*0.15)
X_train = X.iloc[0:training_samples, :]
Y_train = Y.iloc[0:training_samples]
X_val = X.iloc[training_samples:(training_samples + validation_samples), :]
Y_val = Y.iloc[training_samples:(training_samples + validation_samples)]
X_test = X.iloc[(training_samples + validation_samples):, :]
Y_test = Y.iloc[(training_samples + validation_samples):]
```

Details of Trained Models

The three models used for this problem were Linear Regression, LASSO Regression and Ridge Regression. Each of these approaches will be discussed in detail below.

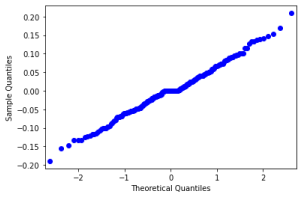
Linear Regression

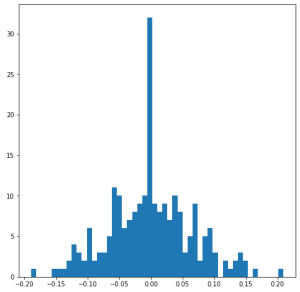
The Linear Regression model uses the Y_train and X_train data taken from the data split function and trains a model using OLS Regression.

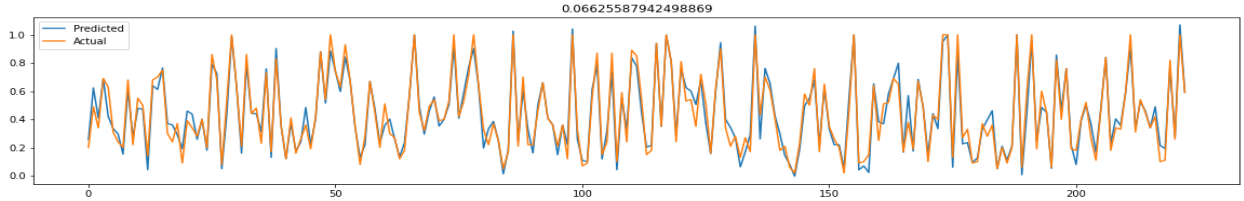
```
model = sm.OLS(Y_train, X_train)
trained_model = model.fit()
print(trained_model.summary())
```

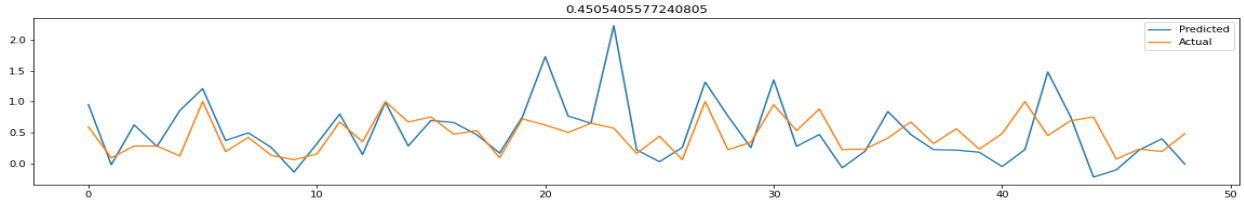
(See Appendix 1 for results of the model summary).

A qqplot and histogram are then used to check that the residuals of the trained model are normally distributed and we then plot the train and test values against the predicted values for each set (See Appendix 2 for code).



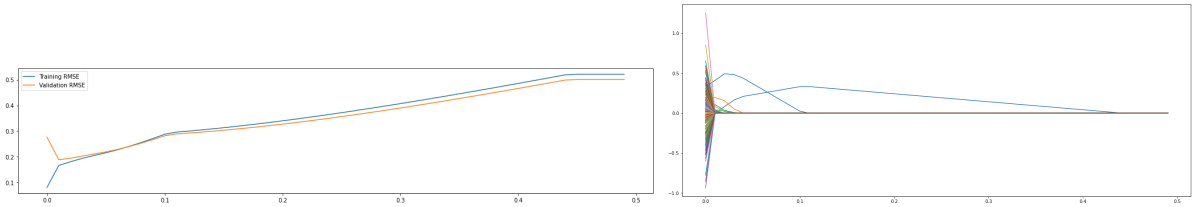




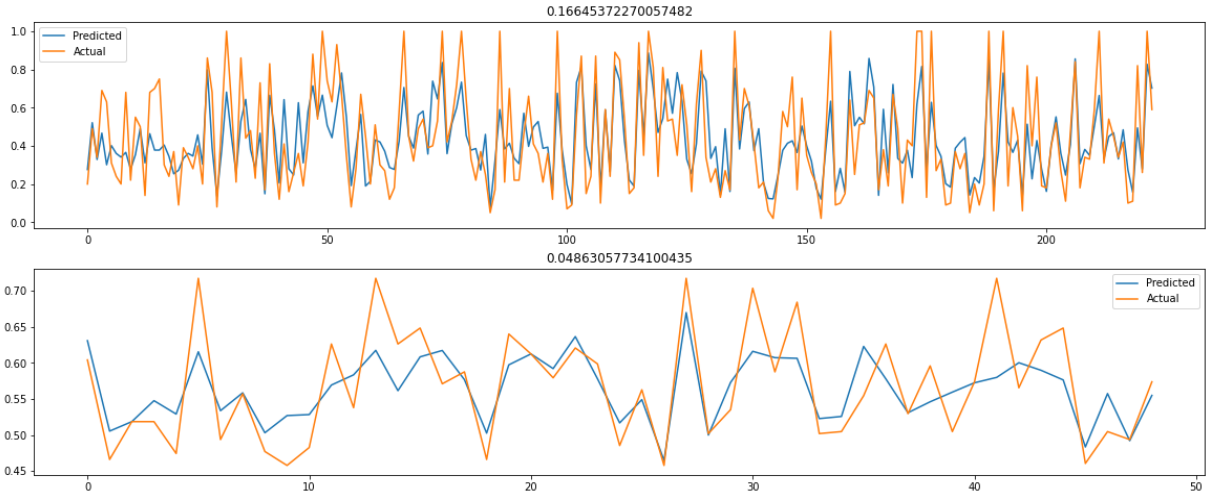


LASSO Regression

LASSO Regression was also tested on the dataset as per the task requirements. This required that we find a suitable lambda (λ) value. We achieved this by testing the RMSE value of lambda values between 0.0 and 0.5 with a spacing of 0.1 (See appendix 3 for code) and then calculating the best lambda value using `best_lambda = lambdas[np.argmin(rmse_validation)]` to find the best lambda value to be 0.01.

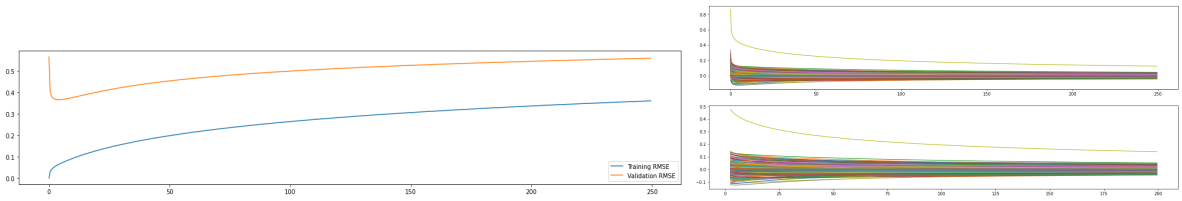


We then trained the model using a LASSO Regression with the lambda value of 0.01 on X_train and Y_train and a plot was constructed using the actual and predicted values similar to the plot constructed from Linear Regression (See Appendix 4 for code).

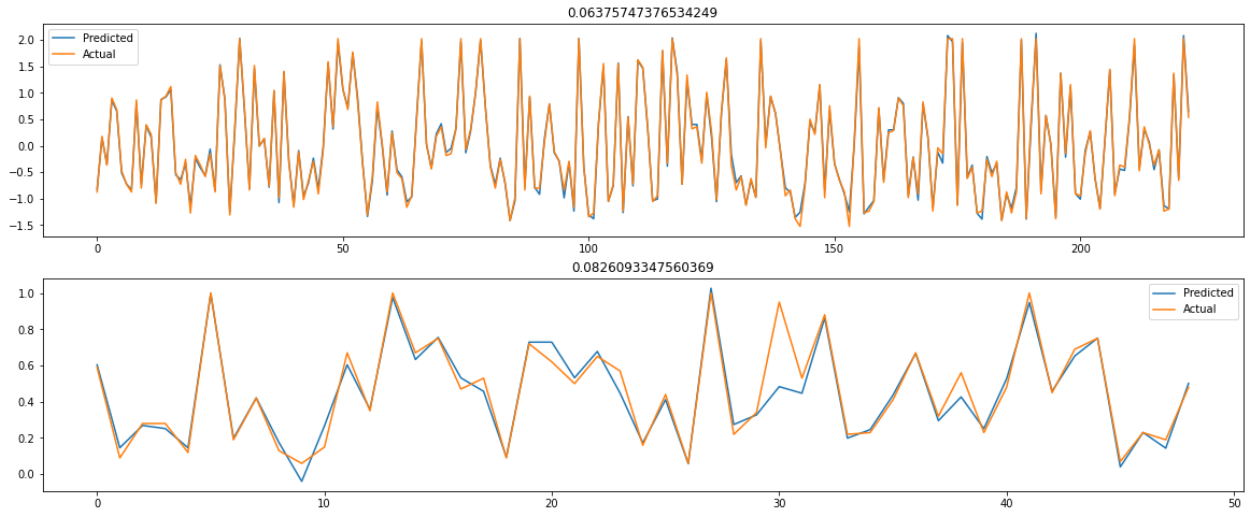


Ridge Regression

The Ridge Regression model had a similar process to the LASSO model. For this model we tested lambda values between 0 and 250 with a spacing of 0.5 and discovered a best lambda value of 4.0 (See appendix 5 for code).



This best lambda value was then used to train a Ridge Regression model using X_train and Y_train and a plot constructed using the actual and predicted values similar to the previous models (See Appendix 6 for code).

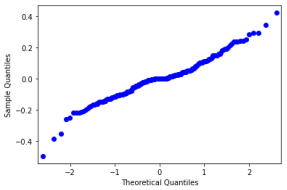


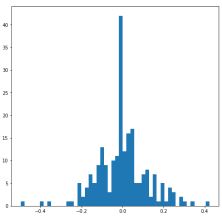
Evaluation of Trained Models

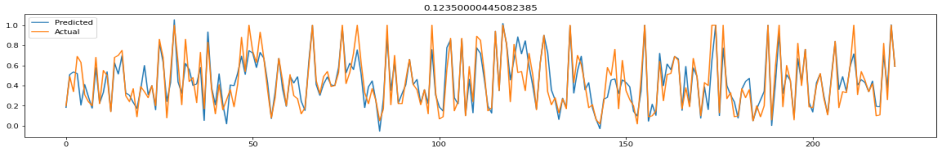
In the process of completing the models, it was observed that there was a serious amount of overfitting occurring in the training data with some models having almost no distinguishable difference between the actual and predicted values of the training set. As such, it was decided to try to reduce the amount of data considered by the model. Through research (Jan Chaiken et al., 1994), we discovered that

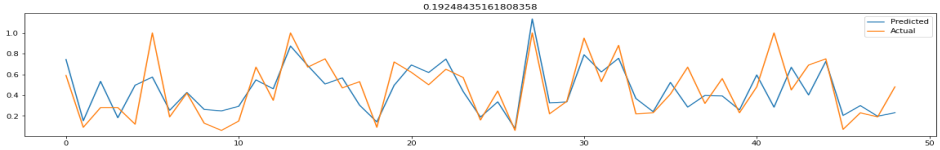
> "Participation in violence is associated with broad social categories such as sex, race, age, and socioeconomic status, but among persons who have participated in violent behavior, other factors have a stronger association with their crime commission rates and persistence. Conversely, correlates of persistence and rates of committing violent crimes—among those who engage in violent behavior—may not be correlated with participation in violent behavior. For example, many people use drugs and many have drinking problems, but most of them are not violent. However, among people who have committed violence, drug users and those with drinking problems are more likely to repeat violence, and among those who have committed a violent act and use drugs, those who frequently use heroin or other opiates are more likely than others to commit violent acts at high rates"

As such, we removed any irrelevant data such that the number of columns were reduced from 186 to 29 (See Appendix 7 for code). This had the effect, in the Linear Regression Model, of greatly improving the fit of the test data without impacting the residuals.



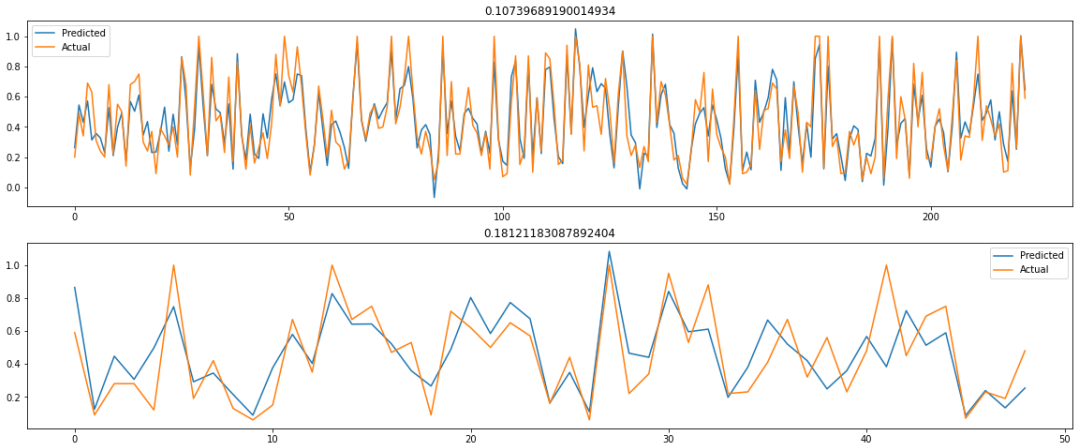




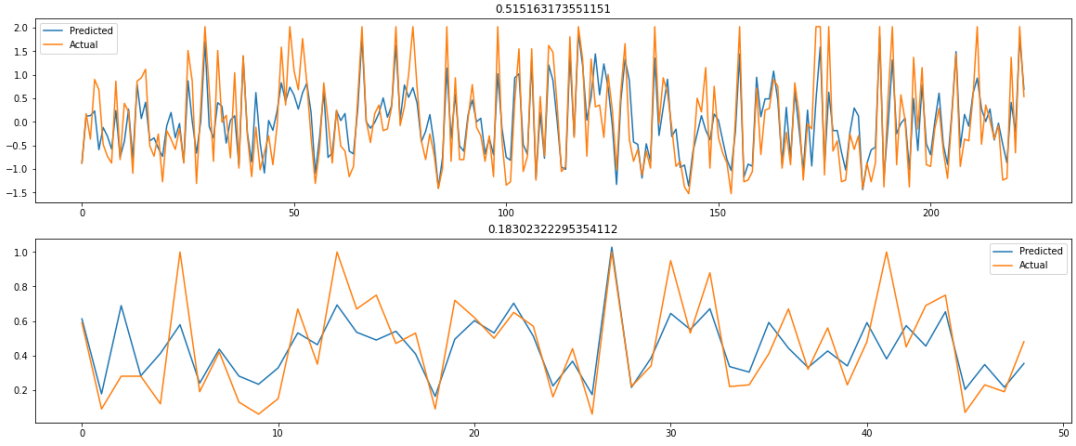


However, this reduced dataset did not provide any improvement in the results obtained by the LASSO and Ridge models, with the LASSO model maintaining a similar RMSE value and the Ridge model achieving a considerably worse result in the test set and both maintaining a large amount of overfitting in the training set.

LASSO Model (Reduced Dataset):



Ridge Model (Reduced Dataset):



Problem 2: Classification

How the data is split

The data was received already split into training and testing sets. The testing set was further split using Sklearn's train_test_split method into 2 sets, testing and validation, for the hyperparameter selection. The split was 70/30, with 70% of the data going to validation. This is due to 70% allowing a sufficiently large set to train the models effectively during hyperparameter optimization, whilst leaving enough for testing the grid search used to optimise the hyperparameters.

Hyper-parameter Selection

All 3 model types had their hyperparameters selected using a grid search. For the SVMs, a grid consisting of a range of 'C' values, linear, rbf, and polynomial kernels, and whether balanced weighting is to be used or not. The rbf grid also altered gamma values and the polynomial grid had a range of degrees being tested. The grid search returned with the optimised values: `{'C': 1, 'class_weight': 'balanced', 'kernel': 'linear'}`

NuSVMs were also optimised using a similar grid search. The same grid was used replacing 'C' with values of 'nu'. The grid search returned the optimised values of:

`{'class_weight': 'balanced', 'gamma': 0.075, 'kernel': 'rbf', 'nu': 0.25}`

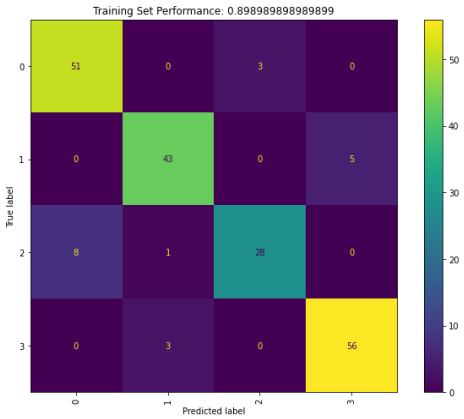
For K nearest neighbour classifier, the values being optimised were the number of neighbours 'n', ranging from 1 to 21. The weights were also varied, either uniform or distance. The grid returned the optimised values of `{'n_neighbors': 9, 'weights': 'distance'}`

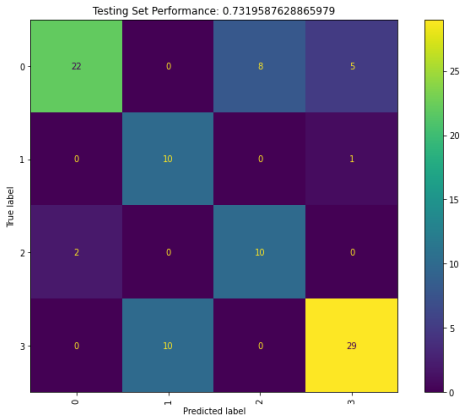
For the random forest classifier, the values for 'n_estimators', 'max_depth', and 'class_weight' were considered. With n_estimators ranging from 1 to 250, max_depth from 1 to 21 and class_weight either balanced, balanced_subsample' or 'None'. The grid search returned the values of:

`{'class_weight': 'balanced', 'max_depth': 12, 'n_estimators': 50}`

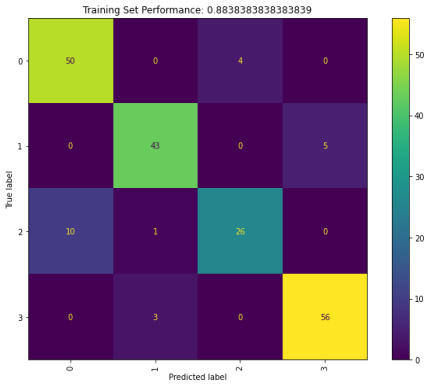
Evaluation of trained models

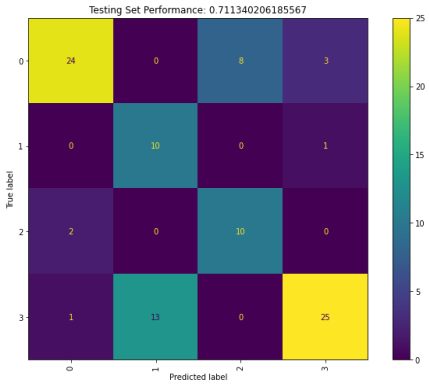
For SVMs, both the regular SVC method and NuSVC method were used. Running the default SVC on the training data resulted in the following model:





Running it again with the optimised values results in:

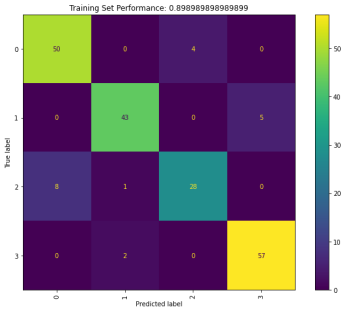


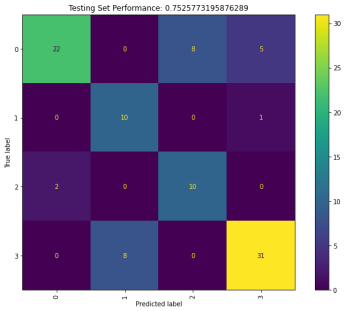


This resulted in a 1% decrease in training accuracy and a 2% decrease in testing accuracy. This decrease is likely due to using the validation set to optimise the values.

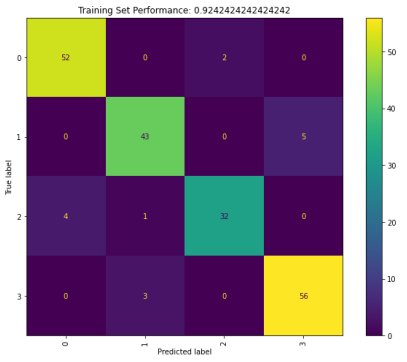
For NuSVC, a result of a 3% accuracy increase in training and 1% decrease in testing was observed.

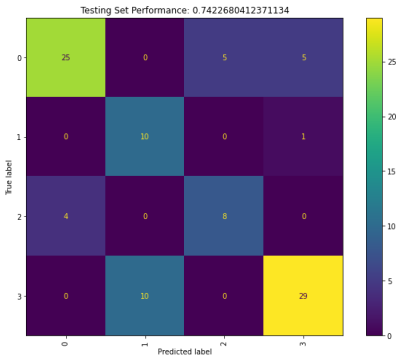
Training with the default values:





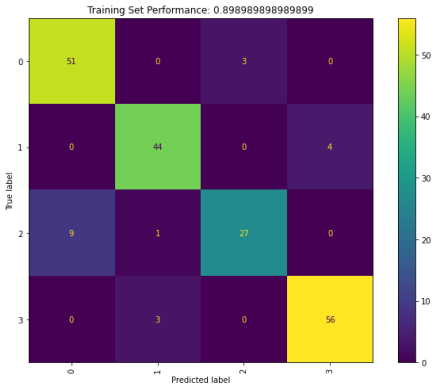
Training with the optimised values:

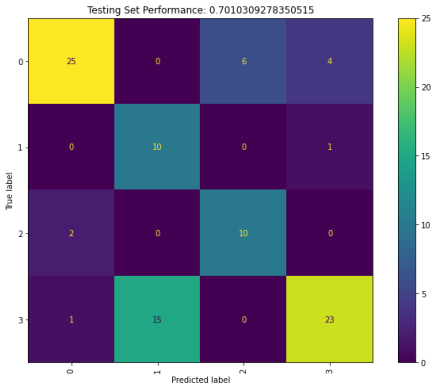




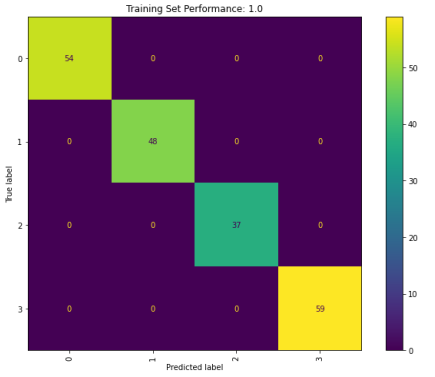
For K nearest neighbour classifier, a 10% increase in training and a 4% increase in testing was observed.

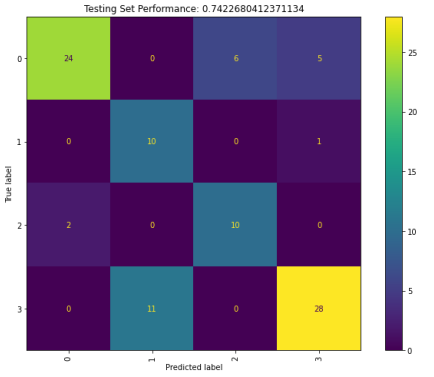
Training with the default values:





Training with the optimised values:

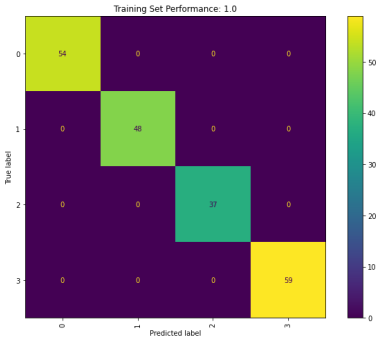


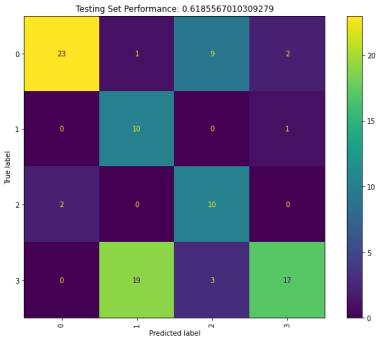


Overall the models are performing similarly to the SVMs. The optimised values look to be overfitting though, with the training performance at 100% whereas the testing is at 74%.

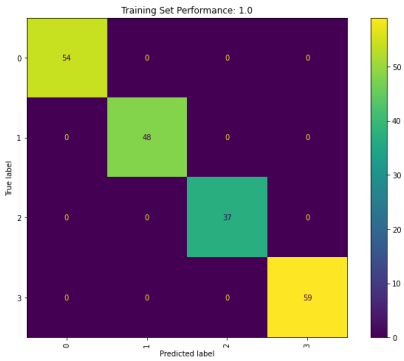
For random forest classifier, the same performance in training and 1% increase in testing was observed.

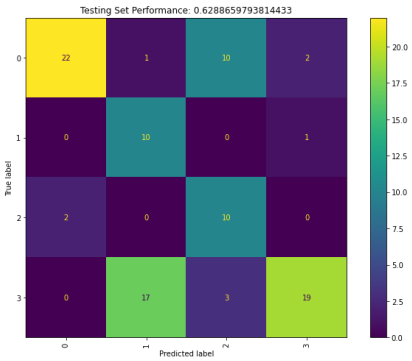
Training with the default values:





Training with the optimised values:





Our models appear to have overfit explaining the great training performance and the poor testing performance.

Overall, most models performed reasonably well, with the best being the default NuSVM. It has 89% accuracy on training and 75% accuracy on testing. However, a few of the models seemed to have started to overfit or are performing worse after the optimisation of values. There are a few reasons as to why this is the case. The validation set was used to optimise the values to be used

with the training set. The values may have been the best with the validation set but not always the case with the training set. Further training and hyper parameter fine tuning could produce better accuracies. Manual review of the hyperparameters or hyperparameter selection using the training set could also lead to better results, but may lead to overfitting. Manual sanitisation of the data may have helped to prevent overfitting, however, the vague column names make this difficult without better understanding of the dataset.

References

Jan Chaiken, Marcia Chaiken, & William Rhodes. (1994). Predicting Violent Behavior and Classifying Violent Offenders. In National Research Council, *Understanding and Preventing Violence* (Vol. 4, pp. 217–295). The National Academies Press. https://doi.org/10.17226/4422

Appendix

Appendix 1

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      ViolentCrimesPerPop   R-squared:                       0.942
Model:                              OLS   Adj. R-squared:                  0.720
Method:                   Least Squares   F-statistic:                     4.237
Date:                  Tue, 13 Apr 2021   Prob (F-statistic):           7.72e-08
Time:                          11:37:31   Log-Likelihood:                288.85
No. Observations:                   223   AIC:                            -223.7
Df Residuals:                        46   BIC:                             379.4
Df Model:                           176
Covariance Type:              nonrobust
------------------------------------------------------------------------------
```

Appendix 2

```
f = sm.qqplot(trained_model.resid)
fig = plt.figure(figsize=[8, 8])
ax = fig.add_subplot(1, 1, 1)
ax.hist(trained_model.resid, 50)

Y_train_pred = trained_model.predict(X_train)
Y_test_pred = trained_model.predict(X_test)
rmse_train = np.sqrt(np.mean((Y_train_pred - Y_train)**2))
rmse_test = np.sqrt(np.mean((Y_test_pred - Y_test)**2))

fig = plt.figure(figsize=[20, 8])
ax = fig.add_subplot(2, 1, 1)
ax.plot(np.arange(len(Y_train_pred)), Y_train_pred, label='Predicted')
ax.plot(np.arange(len(Y_train_pred)), Y_train, label='Actual')
ax.set_title(rmse_train)
ax.legend()
ax = fig.add_subplot(2, 1, 2)
ax.plot(np.arange(len(Y_test_pred)), Y_test_pred, label='Predicted')
ax.plot(np.arange(len(Y_test_pred)), Y_test, label='Actual')
ax.set_title(rmse_test)
ax.legend();
```

Appendix 3

```
lambdas = numpy.arange(0.0, 0.5, 0.01)
rmse_train = []
rmse_validation = []
coeffs = []
for l in lambdas:
    trained_model_poly_lasso = Lasso(fit_intercept=False, alpha=l).fit(X_train_poly, Y_train)
    coeffs.append(trained_model_poly_lasso.coef_)
    rmse_train.append(numpy.sqrt(numpy.mean((trained_model_poly_lasso.predict(X_train_poly) -
Y_train)**2)))
    rmse_validation.append(numpy.sqrt(numpy.mean((trained_model_poly_lasso.predict(X_val_poly) -
Y_val)**2)))

fig = plt.figure(figsize=[20, 4])
ax = fig.add_subplot(1, 1, 1)
ax.plot(lambdas, rmse_train, label='Training RMSE')
ax.plot(lambdas, rmse_validation, label='Validation RMSE')
ax.legend();
```

Appendix 4

```
trained_model_lasso = Lasso(fit_intercept=False, alpha=best_lambda).fit(X_train, Y_train)

Y_train_pred = trained_model_lasso.predict(X_train)
Y_test_pred = trained_model_lasso.predict(X_test)
rmse_train = np.sqrt(np.mean((Y_train_pred - Y_train)**2))
rmse_test = np.sqrt(np.mean(((Y_test_pred*Y_sigma + Y_mu) - (Y_test*Y_sigma + Y_mu))**2))

fig = plt.figure(figsize=[20, 8])
ax = fig.add_subplot(2, 1, 1)
ax.plot(np.arange(len(Y_train_pred)), Y_train_pred, label='Predicted')
ax.plot(np.arange(len(Y_train_pred)), Y_train, label='Actual')
ax.set_title(rmse_train)
ax.legend()
ax = fig.add_subplot(2, 1, 2)
ax.plot(np.arange(len(Y_test_pred)), Y_test_pred*Y_sigma + Y_mu, label='Predicted')
ax.plot(np.arange(len(Y_test_pred)), Y_test*Y_sigma + Y_mu, label='Actual')
ax.set_title(rmse_test)
ax.legend();
```

Appendix 5

```
lambdas = np.arange(0, 250, 0.5)
rmse_train = []
rmse_validation = []
coeffs = []
for l in lambdas:
    trained_model_ridge = Ridge(fit_intercept=False, alpha=l).fit(X_train, Y_train)
    coeffs.append(trained_model_ridge.coef_)
    rmse_train.append(np.sqrt(np.mean((trained_model_ridge.predict(X_train) - Y_train)**2)))
    rmse_validation.append(np.sqrt(np.mean((trained_model_ridge.predict(X_val) - Y_val)**2)))

fig = plt.figure(figsize=[20, 4])
ax = fig.add_subplot(1, 1, 1)
ax.plot(lambdas, rmse_train, label='Training RMSE')
ax.plot(lambdas, rmse_validation, label='Validation RMSE')
ax.legend();

fig = plt.figure(figsize=[20, 8])
ax = fig.add_subplot(2, 1, 1)
ax.plot(lambdas, coeffs);
coeffs = np.array(coeffs)
ax = fig.add_subplot(2, 1, 2)
ax.plot(lambdas[5:400], coeffs[5:400,:]);
```

Appendix 6

```
best_lambda = lambdas[np.argmin(rmse_validation)]
print(best_lambda)
trained_model_ridge = Ridge(fit_intercept=False, alpha=best_lambda).fit(X_train, Y_train)

Y_train_pred = trained_model_ridge.predict(X_train)
Y_test_pred = trained_model_ridge.predict(X_test)
rmse_train = np.sqrt(np.mean((Y_train_pred - Y_train)**2))
rmse_test = np.sqrt(np.mean(((Y_test_pred*Y_sigma + Y_mu) - (Y_test*Y_sigma + Y_mu))**2))

fig = plt.figure(figsize=[20, 8])
ax = fig.add_subplot(2, 1, 1)
ax.plot(np.arange(len(Y_train_pred)), Y_train_pred, label='Predicted')
ax.plot(np.arange(len(Y_train_pred)), Y_train, label='Actual')
ax.set_title(rmse_train)
ax.legend()
ax = fig.add_subplot(2, 1, 2)
ax.plot(np.arange(len(Y_test_pred)), Y_test_pred*Y_sigma + Y_mu, label='Predicted')
ax.plot(np.arange(len(Y_test_pred)), Y_test*Y_sigma + Y_mu, label='Actual')
ax.set_title(rmse_test)
ax.legend();
```

Appendix 7

```
data_smaller = data_to_use.drop(columns=[' householdsize ', ' numbUrban ', ' pctUrban ', ' whitePerCap
', ' blackPerCap ', ' indianPerCap ', ' AsianPerCap ', ' OtherPerCap ', ' HispPerCap ', '
PctLess9thGrade ', ' PctNotHSGrad ', ' PctBSorMore ', ' MalePctDivorce ', ' MalePctNevMarr ', '
FemalePctDiv ', ' TotalPctDiv ', ' PersPerFam ', ' PctFam2Par ', ' PctKids2Par ', ' PctYoungKids2Par ',
' PctTeen2Par ', ' PctWorkMomYoungKids ', ' PctWorkMom ', ' NumIlleg ', ' PctIlleg ', ' NumImmig ', '
PctImmigRecent ', ' PctImmigRec5 ', ' PctImmigRec8 ', ' PctImmigRec10 ', ' PctRecentImmig ', '
PctRecImmig5 ', ' PctRecImmig8 ', ' PctRecImmig10 ', ' PctSpeakEnglOnly ', ' PctNotSpeakEnglWell ', '
PctLargHouseFam ', ' PctLargHouseOccup ', ' PersPerOccupHous ', ' PersPerOwnOccHous ', '
PersPerRentOccHous ', ' PctPersOwnOccup ', ' PctPersDenseHous ', ' PctHousLess3BR ', ' MedNumBR ', '
HousVacant ', ' PctHousOccup ', ' PctHousOwnOcc ', ' PctVacantBoarded ', ' PctVacMore6Mos ', '
MedYrHousBuilt ', ' PctHousNoPhone ', ' PctWOFullPlumb ', ' OwnOccLowQuart ', ' OwnOccMedVal ', '
OwnOccHiQuart ', ' RentLowQ ', ' RentMedian ', ' RentHighQ ', ' MedRent ', ' MedRentPctHousInc ', '
MedOwnCostPctInc ', ' MedOwnCostPctIncNoMtg ', ' PctForeignBorn ', ' PctBornSameState ', '
PctSameHouse85 ', ' PctSameCity85 ', ' PctSameState85 ', ' LemasSwornFT ', ' LemasSwFTPerPop ', '
LemasSwFTFieldOps ', ' LemasSwFTFieldPerPop ', ' LemasTotalReq ', ' LemasTotReqPerPop ', '
PolicReqPerOffic ', ' PolicPerPop ', ' RacialMatchCommPol ', ' PctPolicWhite ', ' PctPolicBlack ', '
PctPolicHisp ', ' PctPolicAsian ', ' PctPolicMinor ', ' OfficAssgnDrugUnits ', ' NumKindsDrugsSeiz ', '
PolicAveOTWorked ', ' LandArea ', ' PopDens ', ' PctUsePubTrans ', ' PolicCars ', ' PolicOperBudg ', '
LemasPctPolicOnPatr ', ' LemasGangUnitDeploy ', ' LemasPctOfficDrugUn ', ' PolicBudgPerPop '])
```