# Problem 1: Regression

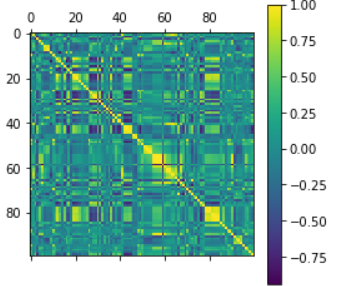
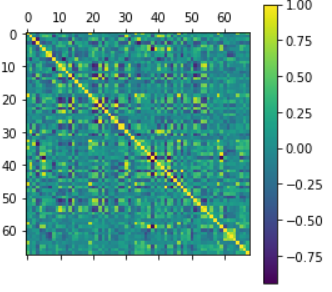
Discussion of data trimming, cleaning, removal and splitting

To get our data ready for analyzing, we first surveyed the data in excel and printed it in python to get an understanding of the type and contents of the data. We first realised we had to remove the leading and trailing whitespaces from the column headers. We next saw that there were 1675 missing data points shown by the ‘?’ sign and only 319 rows with full data given. We wanted to keep as much data as possible, so we set a limit on the number of ‘?’ in a column to 50 and would remove individual rows instead of the whole column. We found that of the columns that contained missing data, all bar one column contained the same high number of ‘?’ and removed these columns. We then saw that the ‘OtherPerCap’ column only contained one ‘?’ and went ahead and removed the one row containing ‘?’.

We split the data into train, testing and validation 60%, 20% and 20% respectively. We did this using the inbuilt train\_test\_split function twice, which randomly chooses data for each category.



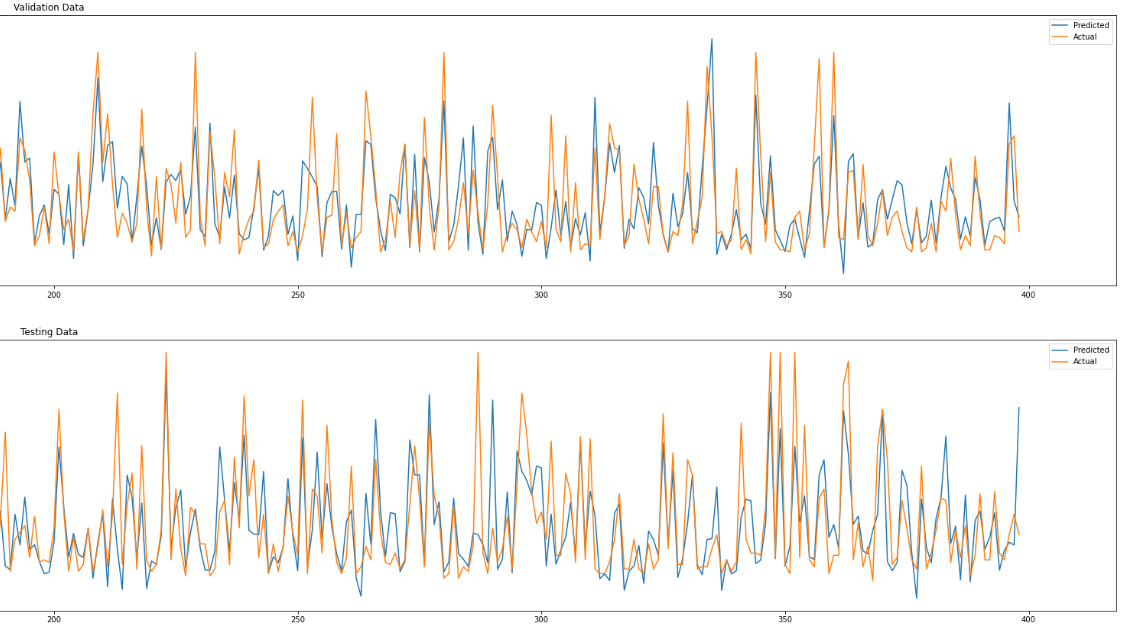
Training a linear regression model

After training the origin dataset without cleansing the variables, it resulted in a reasonable RMSE(validation) of 0.14117668570708608. Adjusted R-squared was 0.691 and F-Stats is 27.67. There were a few large p-value results amongst the variables. Before removing variables with large p-values bluntly, we considered the correlation between variables which indicates the proximity of the linear relationship. This would display the coefficients which are trending in the same way. The higher the correlation, the stronger the relationship.

The next step is to remove some variables that have a high correlation value. The threshold selected was 0.9 as this indicates a strong relationship without the need to remove many variables from the columns. The second correlation chart looks much cleaner than the first one. We then trained the new model and yielded RMSE(validation): 0.13800293289151652, Adj. R-squared: 0.681, and F-Stats: 37.88. The model becomes slightly better, but still isn’t quite optimal.

The next is to consider removing variables with large p-value. The approach taken is backward elimination. backward elimination is a process to have all independent variables in the equation and remove them one at a time provided their contribution to the regression equation is insignificant. The result after selection is RMSE(validation): 0.1368746237213597, Adj. R-squared: 0.680, and F-statistic: 98.39. This illustrates the fit is better and the relationship between predicting variables and the predictors is reliable.

Regularisation - Lasso and Ridge Regression

Before creating Lasso and Ridge regression, we decided to standardise the data first as it scales dimensions equally. Then we split the data in training, validation, and test. Firstly, a linear regression function was fitted. We then calculated the RMSE for validation and test dataset with results of 0.6079140113937643 and 0.6431149701511697 respectively. Looking at the performance for validation data and test data, they are both moderately fine but not optimal.

In order to get a better fitting and avoid overfitting at the same time, we will use some regularisation techniques to reduce the error by adding a penalty term in the error function. After importing Lasso, and Ridge from sklearn.linear\_model, a lasso regression function was performed on the original whole trimmed dataset with all the variables. In this case, we selected three lambda values as three different models, 0.01, 0.1, and 0.5 to test. The result of the Linear model was added to observe the performance comparison between Lasso models and Linear model. The lowest RMSE is from lambda value 0.01, second lowest is the linear regression model. Third is from lambda value 0.1. This gives an idea of where the best lambda lies within. We then tested the lambda value of 0.005 which gave a RMSE that is in between the linear model and 0.01. These results will slowly indicate where the best lambda value sits. We manually tested Ridge regression in the same fashion. The reason for doing this is to allow us to understand how gradient descent works and how learning rate is affecting the best lambda value selection.

The next inline is to automate this manual process and create an algorithm to help retrieving the best lambda value. Once the algorithm is completed, we tested different lambda ranges like what we did in the manual processes to get an idea of where the result is sitting. We used a large range with a small number of steps in the beginning to find the middle ground then we tested the first half and second half to see where the lowest RMSE is heading and repeat until we find an appropriate range. We would then increase the range by 50% and perform a large number of steps. The result from Ridge regression is the best lambda value of 0.10914182836567315, with an Adjusted R squared of 0.669018736427156 and a RMSE (validation) of 0.13726578349467927. The result from Lasso regression is best lambda value of 0.002301150575287644, with an Adjusted R squared of 0.6717290736912699 and a RMSE (val) of 0.1376842488636504. The original linear regression model has a RMSE (val) of 0.14117668570708608. Adjusted R-squared was 0.691.

In this case, the best performance on validation dataset is Ridge regression with the lowest RMSE. The RMSE for test dataset for Linear, Lasso, and Ridge are 0.1474005938347445, 0.14553004429760505, and 0.14439106197569165. Therefore, overall in the comparison, the choice of model is Ridge regression as it outperformed the rest two in the scoring.