Description for regression data set:
http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime

Description for KNN & Decision Trees data set:
http://archive.ics.uci.edu/ml/datasets/Dry+Bean+Dataset

Regression

(1.1) Business Understanding:

This analysis is useful in highlighting which aspects of society might lead to higher rates of violent crime. Obviously, all communities would like to lower these rates. In this analysis, the R-squared value is not very high, due to the general unpredictability of human behaviour. As such, the business case here lies in finding the highest predictor variables of violent crime, rather then being able to predict individual incidences of violent crime themselves. For example, one of the strongest predictors of violent crime rates is whether children come from a household with two parents. If this is identified as an important factor in violent crime incidence, a city or community might be able to put in place supports for single parent families.

(1.2) Data Understanding and Preparation:

The original data set contained information on 128 variables for 1994 cities and towns in the United States of America. I dropped 17 variables, some of which contained information which would be of no use in my analysis (e.g. name of city), and some of which contained hundreds of missing values. One variable was missing data for just one row, so I decided to find the mean for the variable and substitute this value, rather than jettisoning the variable. All variables are numeric and were already normalised to values between 0 and 1 in the original set. The outcome variable was the rate of violent crime in a city/town. The input variables include a wide range of societal metrics including measures of income, educational achievement, household demographics and racial demographics for a given community.

(1.3) Modelling:

I initially ran a simple linear regression model using the lm function in R studio, modelling the relationship between the outcome variable, violent crimes per population, and all other variables:

[comm_model_all <- lm(ViolentCrimesPerPop ~ . , data = df1new)]

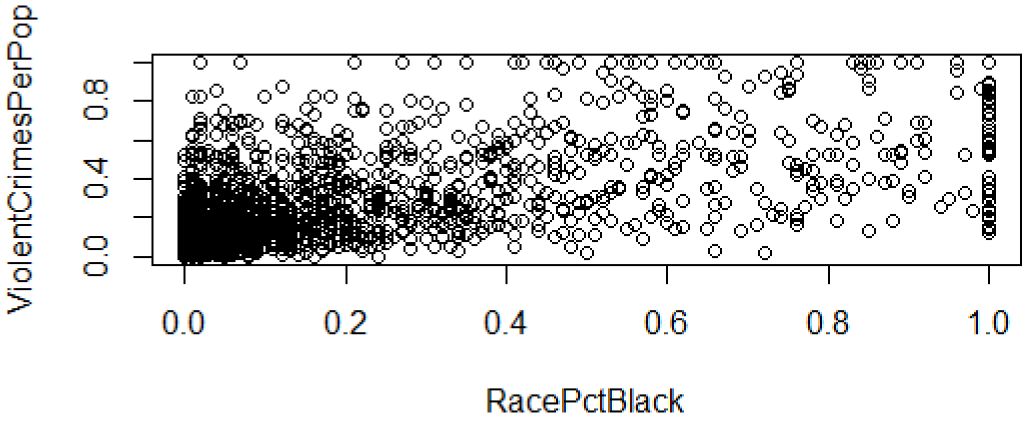
I then analysed a number of individual input variables, specifically those with low p-values. I also tried out polynomial regression for a number of variables whose graphs looked slightly curved. Percentage of people under the poverty level improved the most of the variables I tried;

[polyvar_pov = poly(df1new$PctPopUnderPov, degree = 8)]

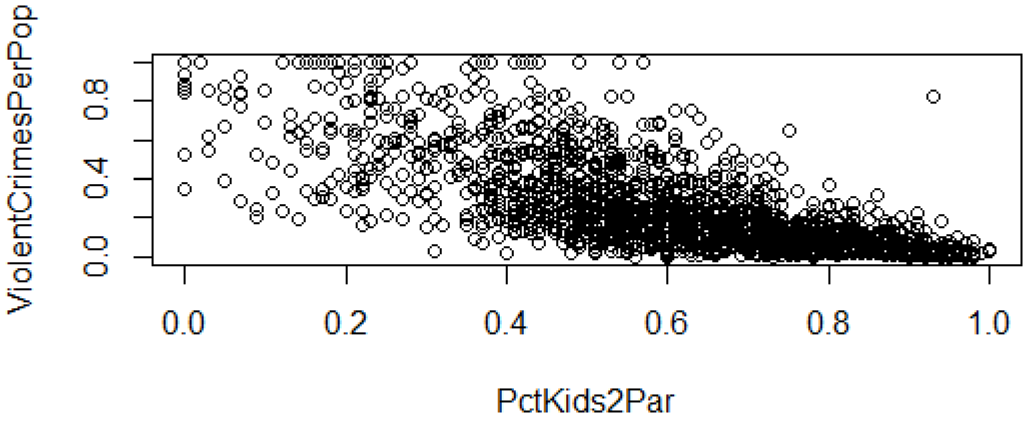
(1.4) Evaluation:

This was not an easy dataset to model, as is often the case when dealing with variables based on complex human behaviour and psychology. Many of those with low p-values from the first modelling of all the variables, such as PctWorkMom (percentage of moms of kids under 18 in labour force), which had a p-value of 0.000596, also had a low R-squared value when modelled individually (0.02267 for PctWorkMom). As such, while there was a statistically significant predictive relationship between them and the outcome variable, there was a lot of variability and thus high residuals.

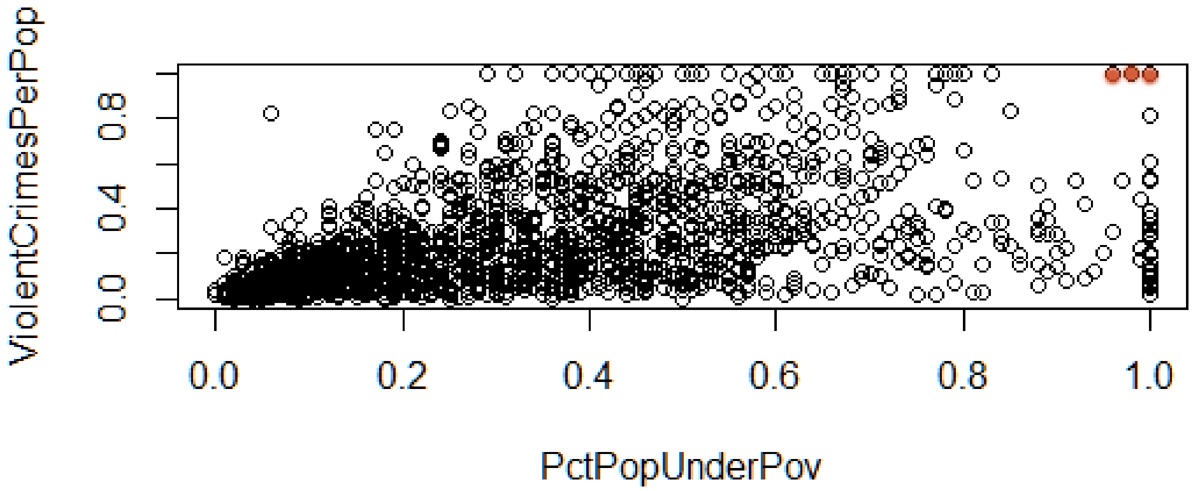
The graph below shows a variable with the lowest p-value in the initial model and a relatively high R-squared of 0.3985. One of the challenges in working with this dataset was that many variables had unbalanced proportionality along the x-axis. As we see in the below graph, there are relatively few communities with above thirty percent black residents, meaning that we should treat these results with caution.



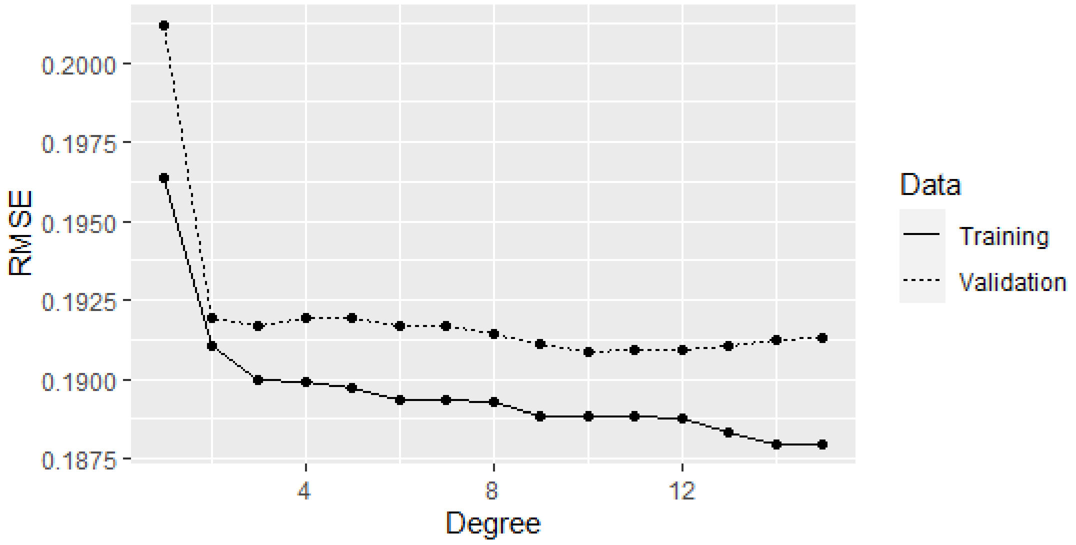
The model below, using percentage of kids in family housing with two parents as the independent variable shows a stronger correlation. It had a p-value in the initial model of 0.036, and an R-squared value of 0.5453 when modelled individually. This sort of correlation could be used to compel societies to put in place more supports for single parent families.

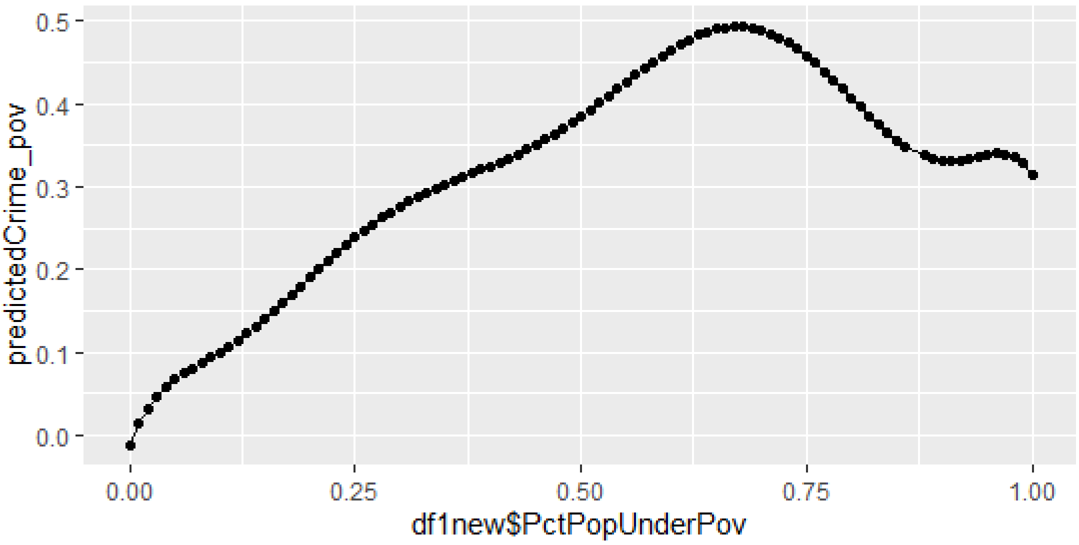


The variable for percentage of people under the poverty level for a community or city improved the most using polynomial regression of the ones I tried. It had a p-value of 0.004268 in the initial model. Its R-squared value went from 0.2724 under linear regression to 0.335 with polynomial regression of degree eight using the poly function.

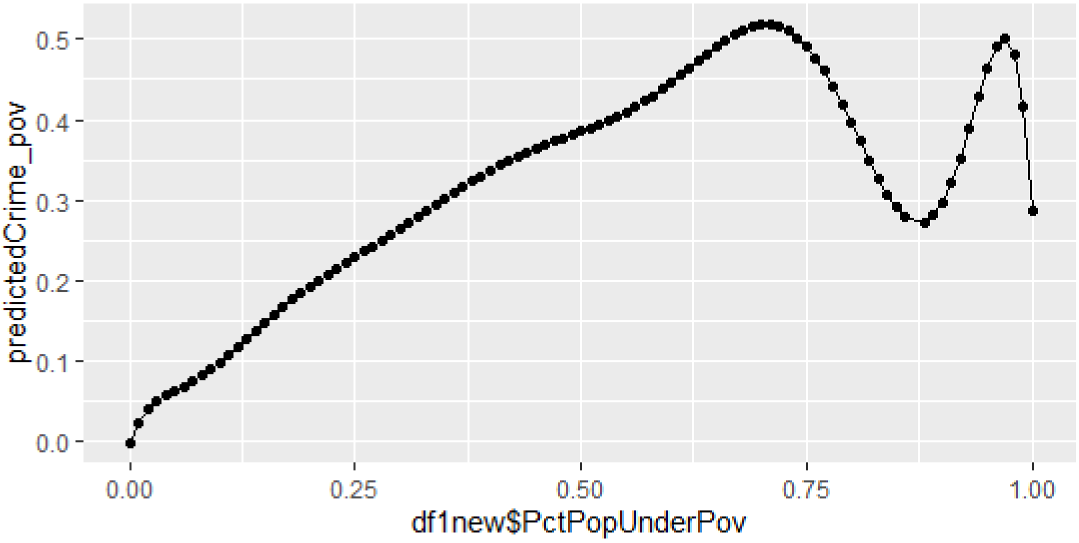


I performed cross validation for this polynomial model, calculating the root mean square of the errors for training and validation data. It showed that up to degree 10 or 12 the RMSE was decreasing for both data sets. However, examining the plots of this polynomial model tells a slightly different story.





Poly – degree = 8



Poly – degree = 10

There are very few communities in the dataset with roughly 75 % – 95 % of the population under the poverty line, although there are quite a few at 100 %. Crucially, there are three instances with a value of 1 for violent crimes which are skewing the graphs above (as highlighted with the red dots in the top right hand corner of graph on page 3). In particular, the graph of the degree 10 polynomial term kicks back up violently to capture these three instances, before arcing back down. Rather unexpectedly we see that communities with 100 % poverty rate or near it are actually rather less violent than we might have thought. Again, given the challenging nature of this dataset, while polynomial regression has improved the R-squared value, it is perhaps not very suitable in this instance, and the curve produced appears rather random in nature and does not give us any further information about the relationship between the two variables, bar the interesting drop seen in violence levels in communities with near 100 % poverty levels.

Decision Trees

(2.1 & 3.1) Business Understanding:

The business case here could be twofold. The primary aim of the analysis is to be able to determine which variety of bean a given bean is, after we have measured various aspects of its shape. This determination could then be used to separate beans at the processing scenario, if it was impractical to do so at the production stage.

Secondly, the better the predictive value for a given bean, the more uniform that variety of bean is, which is a trait often valued by food producers and sellers. A food producer may wish to regularly test the uniformity of each variety and keep the most uniform variety for sale in its raw state, while using more irregular varieties for say, flour production or in pre-prepared meals.

(2.2 & 3.2) Data Understanding and Preparation:

13611 high quality images of different beans, made up of seven varieties, were analysed to record the 16 features which are our independent variables, with the variety being our outcome or dependent variable. All 16 of the input variables are numeric. They include the area, perimeter and roundedness of a bean. Since some variables consist of very high values compared to others, it was necessary to normalise the data for the kNN analysis. For both techniques to was also necessary to convert the outcome variable “Class” to a factor.

(3.3) Modelling:

The 13611 rows in the data frame were first shuffled and then split roughly 85:15 into training and test data. The training data was then used as the basis for our model, using the c5.0 function, imported from the c50 package;

[bean_model <- C5.0(as.factor(Class) ~ .,data = dfbean_train)]

There were 132 nodes in the decision tree, with 616 errors (5.4% error rate). Given the large number of nodes in the tree, I have decided not to include the plot of it here as it is rather messy looking. This model was then used to predict the class of bean for all instances in the test data, using the predict function;

[bean_predictions <- predict(bean_model, dfbean_test)]

(3.4) Evaluation:

By creating a table of the predicted values versus the actual values of bean class for the test data, it is easy to calculate the percentage accuracy of the model;

```
[ctable <- table(bean_predictions, dfbean_test$Class)]
[sum(diag(ctable)) / sum(ctable)]
```

The accuracy rate was 91.663% which is obviously quite a good result. I then performed a boosting operation when creating the model, setting trials=10 in the c5.0 function. There was a reasonable increase to 92.184%.

(4.3) Modelling:

Just as for decision trees, the bean data set was shuffled and split into training and test data. All values were also normalised to between 0 and 1. The model was built using the knn function, using 107 nearest neighbours;

```
[preds = knn(train = dfbean_n_train, test = dfbean_n_test, cl = dfbean_n_train_labels, k = 107)]
```

I repeated the procedure while changing the value of k to 20 and then 1. I also used z-scaling to standardise the variables using the scale function (dropping the 17th column);

```
[dfbean_z <- as.data.frame(scale(dfbean_rand[-17]))]
```

(4.4) Evaluation:

As with decision trees it was possible to calculate the percentage accuracy using the diagonal of a crosstable. The result was 91.615% when k was 107. With k set to 20 the accuracy went up to 92.1% and was still high at 89.9% when k was set to just 1. Using z-scaling the results were 91.5% with k=107, rising to 92.38% with k=20, with 90% accuracy at k=1. As such, z-scaling performed slightly worse for building a model, but slightly better at k=20, illustrating the variability of using kNN for modelling purposes. Regardless, the prediction rate was high, with accuracy of prediction roughly equal between kNN and decision tree models.