Regression Analysis of UCI Wine Quality Data Set

1. Data Analysis

We began by carrying out exploratory analysis on our wine data. The data-set contains 1,599 samples of red wine, with each sample receiving an expert's quality score (between 3 and 8). The input variables are Fixed Acidity, Volatile Acidity, Citric Acid, Residual Sugar, Chlorides, Free Sulfur Dioxide, Total Sulfur Dioxide, Density, pH, Sulphates and Alcohol. The output variable is Quality. Initially, we explored the distributions and summary statistics of the data. These are outlined in Figures 1.1 and 1.2. The shapes of some variables in Figure 1.1 (e.g. residual sugar or total sulfur dioxide) suggest that they are log-normally distributed, although for simplicity (without the time to carry out adequate mathematical analysis) we decided to treat all variables as normally distributed.

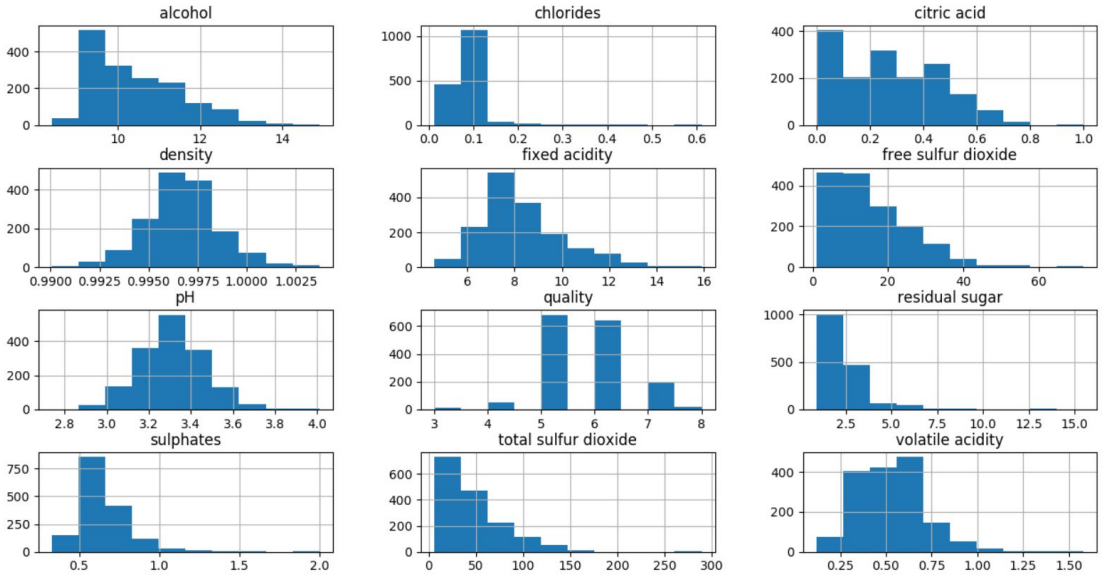


Figure 1.1 - Histograms of the data.

Summary statistics such as quartiles provide us means to exclude outliers (e.g. using Tukey's method, that states a data point is an outlier, if it lies outside the interval $[LQ - 1.5 \times IQR,\ UQ + 1.5 \times IQR]$). With time interests in mind, we decided to continue with all the data points. We continued by examining each input variable (covariate) and plotting scatter plots along with the best-fitting straight line through the points; some of our observations are shown in Fig 1.4. The line represents a linear regression of each predictor on the dependent variable. Sulphates and alcohol have a greater impact on quality than residual sugar and appeared to be the most influential.

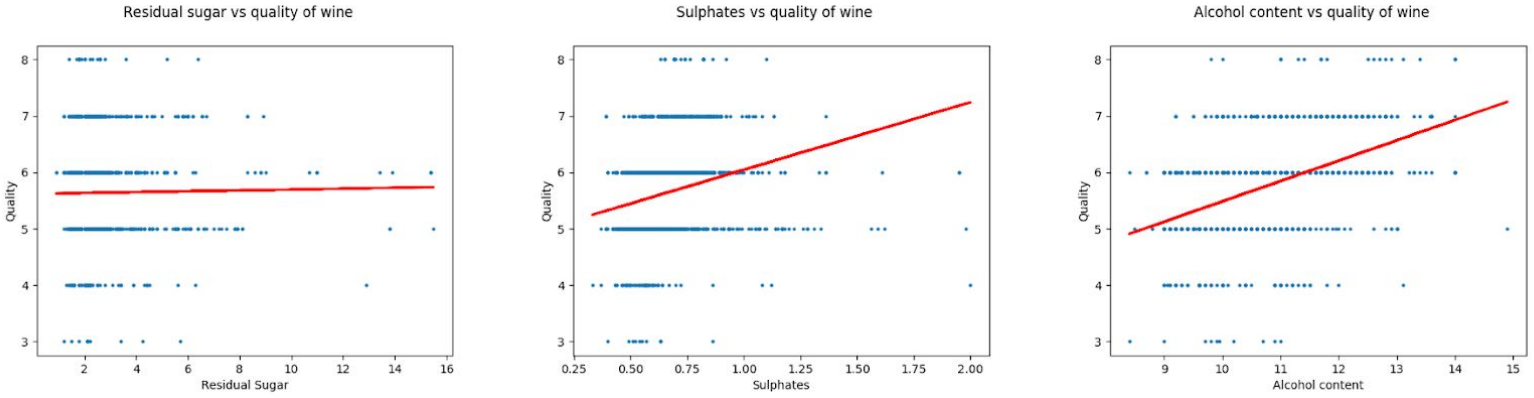


Figure 1.2 - Examples of regression lines.

Figure 1.3 shows a selection of contours plotted, displaying the covariance between certain inputs.

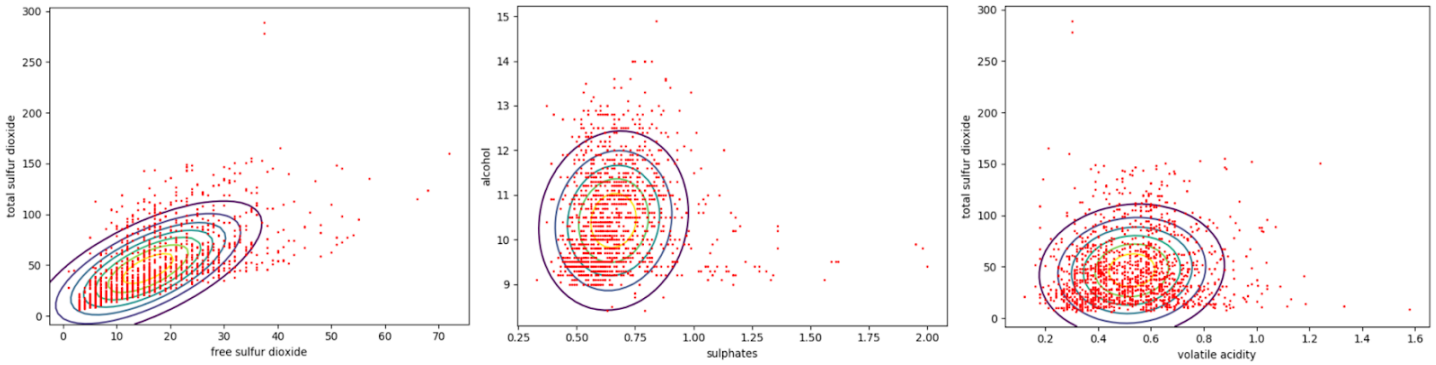


Figure 1.3 - Contour plots showing covariance between some input variables.

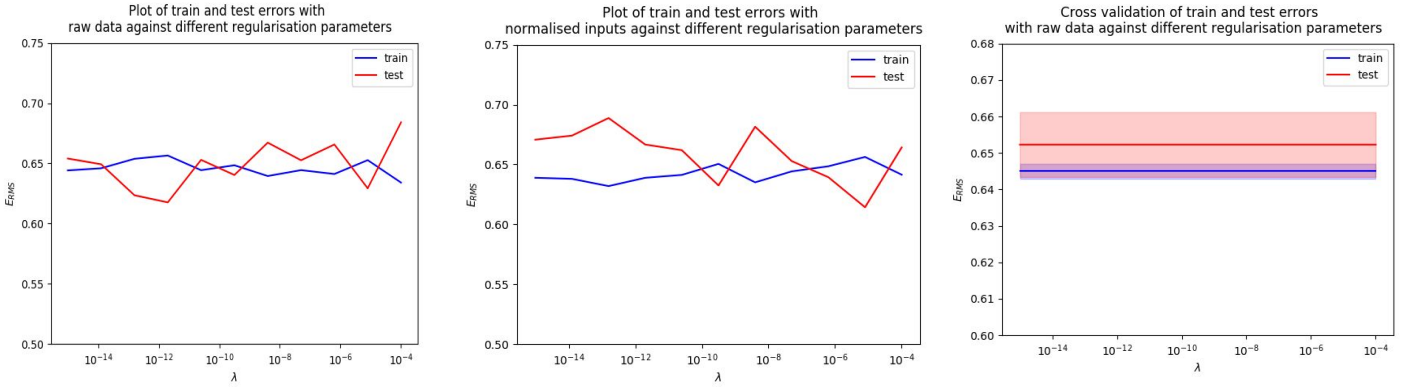
From these, we observe that total sulfur dioxide and free sulfur dioxide levels have some dependence. Thus, in a more advanced modelling situation, we might be able to reduce the dimensionality of the inputs. Other contours, such as total sulfur dioxide and volatile acidity, are relatively axis-aligned - indicating independence - so it would be important to keep an essence of these variables, should they both be influential on the wine quality.

2. Regression Models

In this section we investigate the performance of a selection of regression models on our data set. To ensure consistency and enhance the interpretability of our results a constant seed value of 30 was set for all models.

2.1.1 Simple Linear Regression

Simple Linear Regression is a very simple approach for supervised learning. It is used to make a prediction for the quantitative response, Y, from the the predictor variable, X. In this model, the target, Y, was the wine quality and the data inputs, X, were all other features of wine in the data file. For this model, the data was split into train and test parts; test fractions between 0.20 and 0.25 were used. A linear model was then fit to the training part using the least squares approach and subsequently evaluated on both training and test data. Maximum likelihood weights were used, which gave the best linear fit between the processed inputs and the targets. A graph of the train and test errors was plotted, by using raw data against different regularisation parameters, as shown in Figure 2.1.1(b). Standardisation was then used to take into account the different scales of the raw data and allow a faster computation. This was done to investigate if it had any effect on the test and train errors and a graph of this observation is shown in figure Figure 2.1.1(b).



(a) (b) (c)

Figure 2.1.1 - Test and train errors obtained from linear regression using a) raw data, b) standardised data, c) cross-validation.

The train and test errors obtained by using the raw data were 0.651 and 0.632 respectively. Similar to this, the train and test errors obtained by using the normalised inputs were 0.637 and 0.678 respectively. The difference between the mean value of these errors was minute, moreover it was observed that normalisation

had no effect on the results of linear regression. Furthermore, it was observed that the change in the regularisation parameter showed minor fluctuations in the errors values, however the magnitude of these fluctuations was miniscule.

Cross-validation was then performed using the K-fold method. The number of folds used to perform this function was 5, i.e. the data was divided into 5 subsets. The method was repeated 5 times, such that each time, one of the 5 subsets was used as the test / validation set and the other 4 subsets were combined to form a training set. The error estimation was averaged over all 5 trials to get the total effectiveness of our model. The cross-validation plot in Figure 2.1.1(c) shows the variations of the train and test errors with their error bars, against different regularisation parameters. The mean train error obtained across the different measurements was 0.645 and the mean test error was 0.652 – a minor difference from the results obtained previously thereby warranting the integrity of our model. It can also be observed that the changes in the regularisation parameters had no effect on the linear regression model.

2.1.2 Simple Linear Regression with Gradient Descent

To further explore simple linear regression, the gradient descent method was considered to perform linear regression on our data set; gradient descent is an algorithm that is used to minimize a function, in this case, the cost function. The aim was to find a value of θ which renders the lowest error and enables the cost function to reach a local minimum. Each iteration in this method aims to find a new value of θ that yields a slightly lower error than the previous iteration. A learning rate is also used to control how large of a step we take downhill during each iteration. In this model, a small learning rate of 0.01 was used for a slow descent thereby increasing the accuracy of this approach. This model used the normalised inputs for optimisation and 1000 iterations were run to find the local minimum. A graph of the error function against the iterations was obtained and is depicted in Figure 2.1.2.

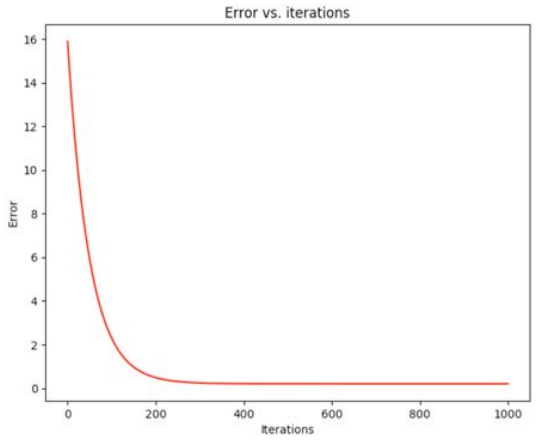


Figure 2.1.2 - Plot of the error function against the number of iterations by performing linear regression using gradient descent

It can be seen in Figure 2.1.2 that after a certain point the error function reaches the local minimum. The value of theta that allowed the error function to reach this point was then used to create a predict function. The input variables from the data set were then used to compute the quality and the results obtained showed that this method rendered a test error of 0.646.

The simple linear regression model is an easy and intuitive method to use and understand and is a convenient way of finding relationships between different variables. This method, however, assumes that the covariates and response variables exhibit a linear relationship, which is not the case in many situations and this may oversimplify the problem. While gradient descent did not necessarily improve the performance of our model, it is beneficial to use this method in cases of multidimensional data to reduce the run-time of the model.

2.2 Radial Basis Functions

We started by evaluating a simple linear approximation, in order to have a basis for comparison with our model using the radial basis functions network. The test error we got was about 0.6909. We then standardised our input variables (to even out the scale differences) and proceeded to an evaluation for various regularisation parameters. As an initial estimate for the centres, we sampled 10% of the input points randomly and chose an initial width analogous to the standard deviation. We constructed a feature mapping function $\boldsymbol{\phi}(\mathbf{x})$ and applied it to our inputs to get the design matrix. We specified a range of regularisation parameters to examine and train

our model. As a side note, we also split our data to training and testing subsets for our analysis (for example by setting aside 20% of inputs to serve as test data).

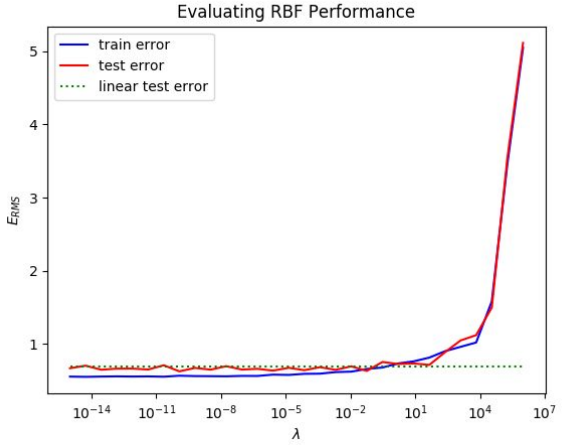


Figure 2.2.1 - Initial examination of RBF performance across a range of λ.

We observed that our model does not really perform much better than a simple linear one. Our next step was model selection, by searching for the optimal parameter values. We constructed matrices with ranges of scales and λs as indexes, containing the train and test errors for each pair of parameters, respectively. This enabled us to determine the pair that produces the minimum test error, our best model choice.

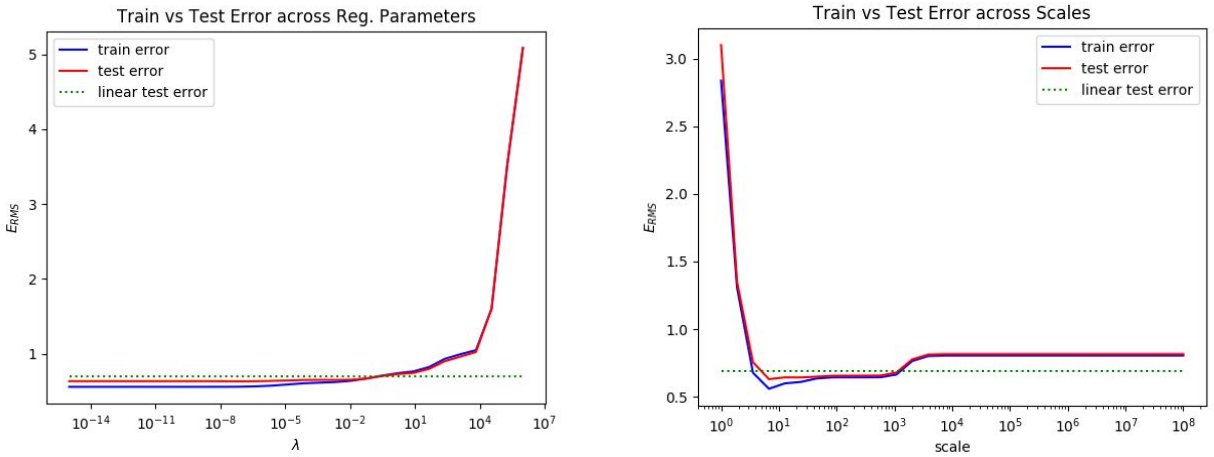


Figure 2.2.2 - Searching for the best parameters.

As an addition, we also evaluated the effect of choosing a different percentage of inputs to use as centres, meaning a different number of centres. We found from Figure 2.2.3 that our best parameters give us a very small reduction in RMSE and centre number (as a proportion of inputs) does not appear to have a significant effect.

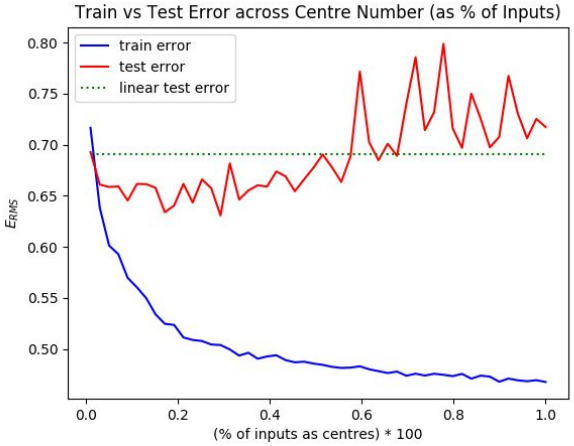


Figure 2.2.3 - Examining the effect of centre numbers.

Even when bootstrapping this whole parameter search as many times as 50 using a loop, there was not much improvement. To conclude, we evaluated the impact of using cross-validation, if any. We used the optimal values we had already discovered and created from 5 to 10 folds. Figure 2.2.4 displays the results:

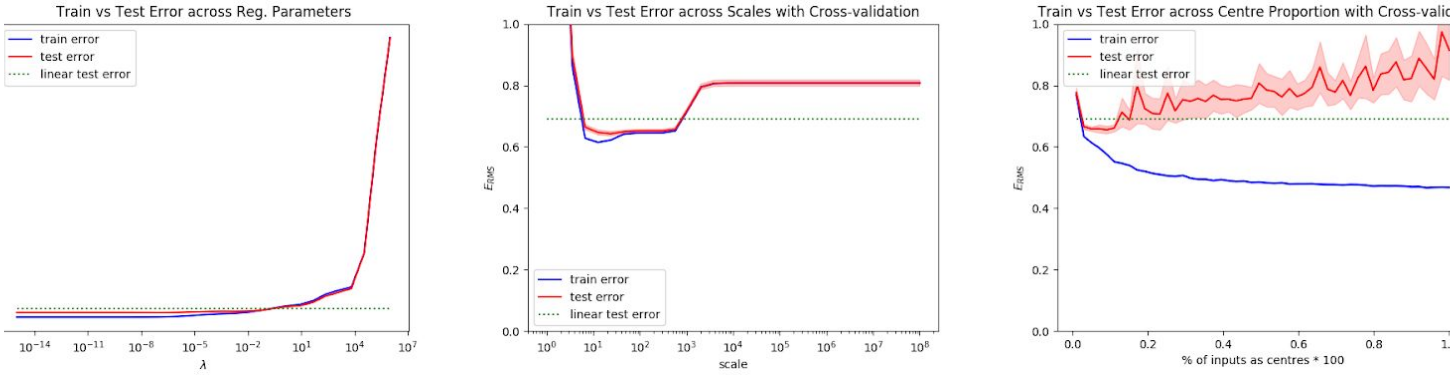


Figure 2.2.4 - Examining the effect of cross-validation on performance.

As the number (proportion) of centres increases, we notice in the last graph the effect of overfitting, since the training error reduces, but the test error increases. Increasing the number of folds all the way up to 10 did not make a difference in our cross-validation. In conclusion, using a radial basis functions network did not improve the performance of our model significantly over a simple linear regression. The order of improvement in error rate was around 1% compared to a simple linear model.

2.3 A Bayesian Treatment of Radial Basis Functions

After concluding the Radial Basis Functions analysis, we decided to explore the application of Bayesian methods to those networks, thus steering away from frequencies to beliefs (Liu and Wasserman, 2014). After standardising our inputs and splitting them to train / test parts, we set the centres to be sampled from our inputs according to the optimal proportion discovered above. We followed a similar procedure for our Gaussian basis function scale (width s), deeming these to be our best initial estimates and with the aim of controlling overfitting, as described by Bishop (2006). We consider the widths to be fixed for simplicity, as illustrated by Barber and Schottky (1998, p. 2). Following that, we calculate the feature mapping function $\Phi(X)$ and plot the basis functions for illustration purposes, using a standardised sample data matrix and the resulting design matrix.

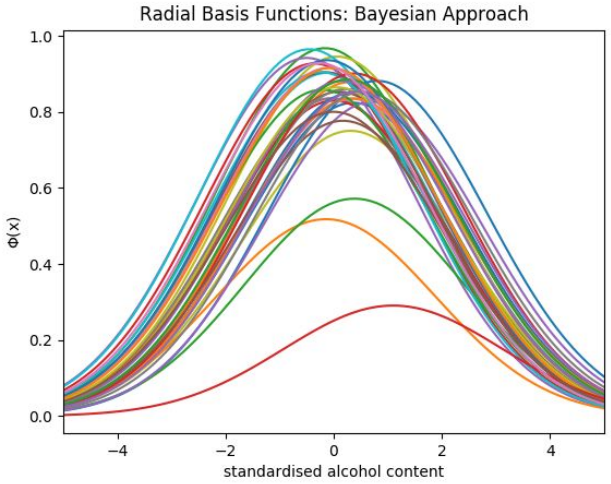


Figure 2.3.1 - Radial Basis Functions Using Design Matrix.

We proceeded to estimate our α and β, which define our initial curve shape. Our β represents the noise precision of our data (spread of the curve) and was estimated as the reciprocal of the variance of targets. The higher our initial precision β, the less overlap we assume between distributions, so it is worth exploring the effect of different values. As a side note, we also tried estimating our beta using the variance of the residuals $t_n - y(x_n)$ from a simple linear regression, but that did not appear to make any observable difference. As far as

our α is concerned, we decided to set it to the mode / mean of the targets because it represents our prior assumption about the top of our Gaussian curve. We defined a prior μ_0 and Σ_0 and calculated the posterior distribution (μ_N, Σ_N) over weights, fitting our design matrix to the targets. We then plotted the mean prediction (MAP estimate), with one of our input variables on the x axis for two-dimensional visualisation purposes. Finally, we took a number of samples from the posterior, and plotted the predictive distributions, along with lower and upper bounds. Here is the outcome of our analysis:

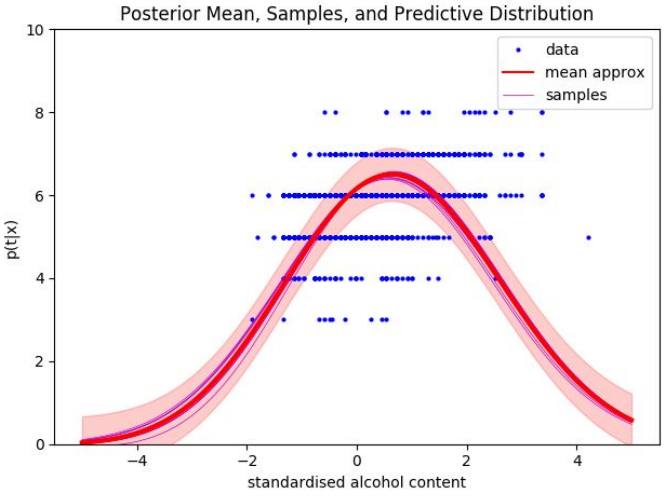


Figure 2.3.2 - Plotting central prediction, posterior samples, and predictive distribution.

The root mean squared test error we get from the central prediction of our Bayesian inference is slightly higher than the aforementioned methods (around 0.72-0.73), but it is an inadequate validation metric, due to its fundamental disregard for uncertainty. It does not consider or take advantage of the probabilistic nature of Bayesian applications. We also attempted briefly to compare the negative joint log probability to that of a simple linear regression, which appeared to indicate that our Bayesian approach is very slightly better. Furthermore, a similar procedure to our non-Bayesian parameter search could be implemented for model selection, examining ranges of hyperparameter constants to discover those that minimise our validation score of choice. Due to time and space constraints, this has to be reserved for the Future Work section, even though our educated guess is that the final model would not depend on prior assumptions, given enough data. Finally, we have concluded that cross-validation is not necessary for α and β, in the interest of a lack of time and based on Barber and Schottky's (1998, p. 5) conclusions regarding the retrieval of a-posteriori probabilities.

2.4 kNN

K-nearest neighbour regression is another simple approach to predict result Y based on multiple attributes X. It takes the average of the target value Y of K nearest neighbours as the prediction. For our project we used Euclidean Distance to calculate the distance between target and neighbours. The data was split into train set and test set with a train:test ratio of 4:1 as performed in the other models. A graph of test mean square error was then plotted, as shown below, to evaluate the performance of the prediction.

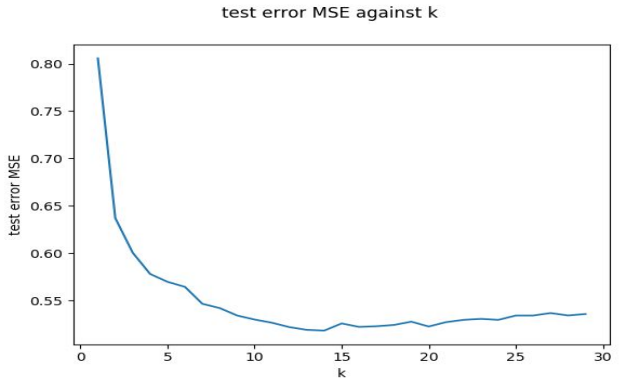


Figure 2.4.1 - Test error (MSE) against K (number of neighbours).

Based on the graph, we can get our minimum test error at k=14, where the mean square error is equal to 0.518. Notice that as the number of neighbours increases, the error rate decreases first and then increases slightly after it hits the minimum point. This is because at the beginning there are not many neighbours and some factor might affect the target stronger than others. At the end, as the number of neighbours increases, extra noises might be introduced to the model and cause the error to raise.

Cross-validation was also performed for K-nearest neighbour regression model. The sample was divided into K parts and each part was tested against other data. For our model, we used 5 folds, which means the data was split into 5 parts and tests were carried out 5 times.

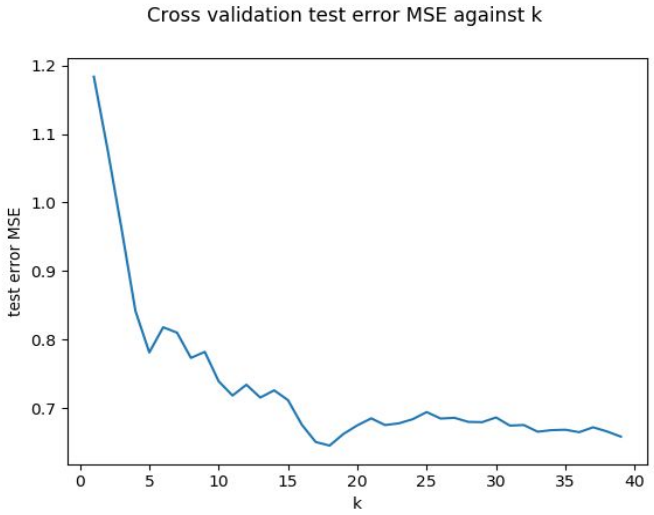


Figure 2.4.2 - Cross-validation for kNN.

We used the same error measure for our cross validation as well. As we can observe from Figure 2.4.2, there is a similar trend. However, the error value is slightly bigger than that of the actual test error and the fluctuation. This might be due to two reasons: Firstly, the way we selected our data, which was randomly selected, might cause variance for the error with the same k value, and the final error displayed in the cross validation is the mean value of these errors. Secondly, raw data was used in the model - generalisation might be helpful to increase the accuracy. It is also worth noticing that the optimal k value for normal test error evaluation and CV test error falls in the same range, as optimal k for normal error test is 14 and optimal k for cross validation is 18.

2.5 Logistic Regression

The wine qualities were split into two discrete categories – “Good” and “Bad”. Wine was deemed to be “Good”, if its quality rating was at least 6. A rating of 5 or lower was re-coded as a “Bad” score. These two categories formed the 1 and 0 classes in our logistic regression analysis. To control overfitting, we used an L2-regularisation technique. This method involves squaring the error term – thus severely punishing larger internal parameters.

During our initial investigation, the input variables “Alcohol” and “Sulphates” were found to have much higher odds ratios than the other variables, as shown in Figure 2.5.1. This means that a change in these variables would have a greater impact on the odds of admitting wine as “Good” or “Bad”. Following this observation, we analysed logistic regression using all the input dimensions, as well as using just Alcohol and Sulphates as input variables.

```
==================================
Odds ratios:
fixed acidity             1.099908
volatile acidity          0.036153
citric acid               0.278362
residual sugar            1.038830
chlorides                 0.018561
free sulfur dioxide       1.022837
total sulfur dioxide      0.983629
density                   0.000827
pH                        0.551776
sulphates                15.428373
alcohol                   2.475627
```

Figure 2.5.1 - The odds ratios of the input variables.

A higher score indicates a larger impact on the acceptance odds of the wine quality.

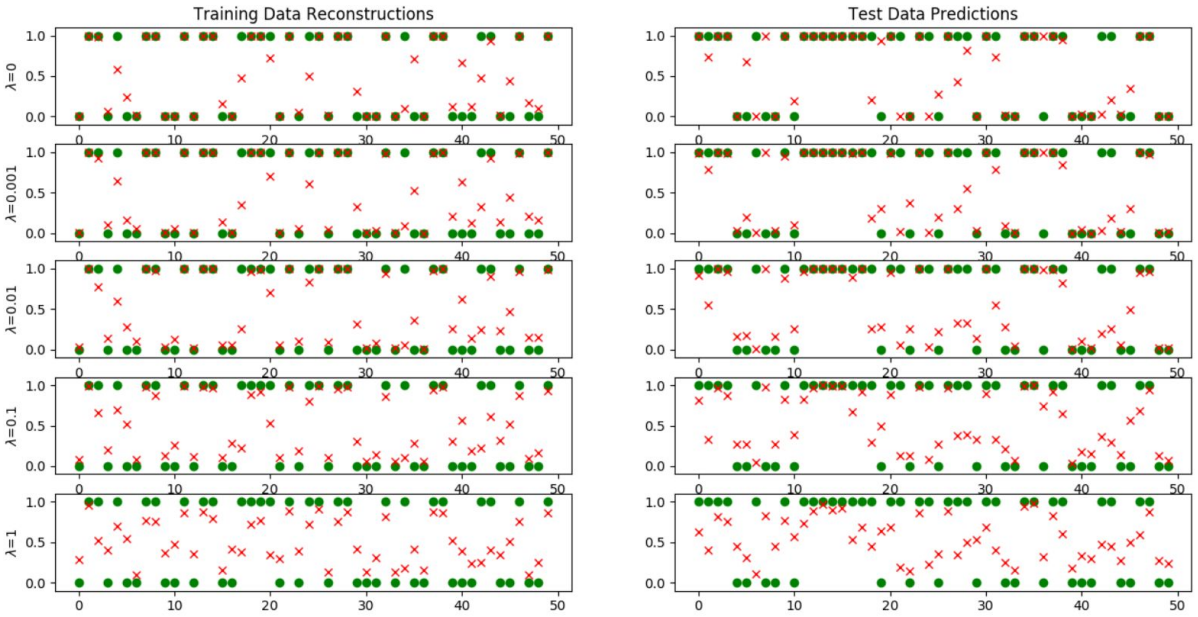


Figure 2.5.2 - Results of Logistic Regression with L2 regularisation.

Figure 2.5.2 shows a subset of the results of the logistic regression analysis, taking all dimensions as inputs. In the figure, the left-hand-side represents a reconstruction of the training data, while the right-hand-side represents the test data. The x-axes represent different data points and a zoomed-in view on a subset of the total data has been used for the sake of visual clarity. Data points are scored as 1 or 0 on the y-axes, dependent on their quality score. The red crosses show the certainty of our model that the wine quality was "Good". For example, a red cross that has a y-value of 0.7 indicates that we are 70% certain that this wine quality was in the "Good" category. A y-value of 0.1 indicates that we are just 10% certain that the wine quality is "Good", i.e. we suspect that this wine quality is, in fact, "Bad".

Lower values of λ (the regularisation parameter) mean that the model had less control for overfitting. In these cases (the graphs at the top of the figures), the model was nearly always very certain about the wine quality. However, it was sometimes very wrong as well – there are several cases where, for example, the model predicted the wine to be "Good" with near 100% certainty, when it was actually "Bad". With higher values of λ, we see a reduced certainty about the wine quality, but also a lower mistake-rate. Certainty values within a certain range could be ignored, for example if the certainty is between 35% and 65%. This shows that the value of λ must also not be too high (as we see in when λ = 1) – so that we can, at least, glean some useful results from the regression. The model created using just Alcohol level and Sulphates as inputs is near-identical to this one. This illustrates the lack of effect of the other variables on the logistic quality score – as hypothesised by the odds ratios.

The Logistic Regression model used has benefits in its simplicity and controllability – managing the overfitting does not require a great degree of model complication and the results are clear to interpret. In the case of the wine quality data, a score of 5 is sufficiently close to a score of 6 for it not to be a problem to misallocate these scores in other models. Here, however, the difference between “Good” and “Bad” is seen as huge and so the model could be ignoring scores that would otherwise be quite close.

3. Discussion and Conclusion

A simple linear regression produces a test error in the range of 0.63 to 0.69. The simple linear regression approach tends to predict values of quality around 5-6 for most inputs, which is reasonable given the distribution of our output variable (quality) and puts this error value in context. Applying a radial basis functions network did not give us much in terms of performance improvement. Possible reasons for that include the fact that the data is categorical in nature, making them more suited for classification techniques rather than regression. The Bayesian approach gave us an indication of the underlying probabilities. Logistic regression appears to have a good belief of the probability of wine being “Good” for certain values of λ; however, one can question its appropriateness given the inherent contrast between its binary nature and the non-binary nature of wine quality (i.e. bad could be very close to good or very far away). KNN produce a lower test error of 0.52 than other models, it give a rather accurate estimate as most of the predicted values are same or similar to the actual results. A k value around 15 will be optimal for this model, according to the evaluation.

4. Future Work

Future work could involve investigating methods to reduce the dimensionality of the input data. 11 co-variates make the analysis cumbersome and - perhaps more importantly - many of them are correlated, introducing thus collinearity questions. Potential methods that could be explored to reduce the number of features would be Principal Component Analysis or Principal Axis Factoring. For instance, dimensions such as fixed acidity and volatile acidity, or free sulfur dioxide and total sulfur dioxide could probably be reduced to common factors, while still preserving the necessary degree of variation and not losing too much in terms of information.

Furthermore, the distribution of some of our input variables appears to follow a log-normal distribution, rather than a Gaussian one (as already discussed in our initial exploratory analysis). This definitely merits some additional exploration in terms of preparing our variables and the underlying distributions we assume in our analyses. One final note is the presence of outliers in our data set, which appear to distort the result, for example when it comes to logistic regression. Removing these outliers could improve the performance and interpretability of our findings.

Finally, more informed approaches should be explored, for example with regards to basis function or parameter selection and modelling using Gamma distributions. When everything is taken into consideration, however, it is our belief that the data ultimately calls for a classification approach, but even then it might not be possible to get much lower error rates based on the provided input variables. Our personal guess is that the wine ‘quality’ target is too subjective to be predicted with high accuracy.

References

Cortez, P., Cerdeira, A., Almeida, F., Matos, T. and Reis, J., 2009. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, 47(4), pp.547-553.

Wittenauer, J.,2015. Machine Learning Exercises in Python, Part 2. [Blog] Available at: http://nbviewer.jupyter.org/github/jdwittenauer/ipython-notebooks/blob/master/notebooks/ml/ML-Exercise1.ipynb.

Christopher, M.B., 2006. Pattern Recognition And Machine Learning. Springer-Verlag New York.

Barber, D. and Schottky, B., 1998. Radial basis functions: a Bayesian treatment. In Advances in Neural Information Processing Systems (pp. 402-408).

Liu, H. and Wasserman, L., 2014, Statistical Machine Learning. Chapter 12: Bayesian Inference. Available at: http://www.stat.cmu.edu/~larry/=sml/Bayes.pdf.

Guestrin, C. and Fox, E., 2018. Overfitting & Regularization in Logistic Regression. [video] Available at: https://www.coursera.org/learn/ml-classification/lecture/DBTNt/l2-regularized-logistic-regression.

Tarlow, D., 2009. Python logistic regression (with L2 regularization). [Blog] This Number Crunching Life: Randomness In The World With A Smattering Of Other Randomness. Available at: http://blog.smellthedata.com/2009/06/python-logistic-regression-with-l2.html.