

# Appendix

Thomas Fishwick

## 1 Glossary

- Bagging – using sampling with replacement from the data to create artificial training data (bootstrap samples)
- Bayesian Optimisation – a function used to run through an array of different hyperparameters for a machine learning model
- BFGS - (Broyden-Fletcher-Goldfarb-Shanno quasi-Newton algorithm) a second order derivative used to help optimise linear regression.[1] (paraphrased)
- Decision Tree Regression – splits the data into distinct regions with the region’s mean being the response to a prediction [2] (paraphrased)
- Features - The independent variables used to predict something in Regression
- Hyperparameter – a parameter of the machine learning model (so named as machine learning models are used to find the parameters of a model)
- Labels - the target/y value in Regression
- Lasso Regression – Least Absolute Shrinkage and Selection Operator. Tries to eliminate as many features as possible when performing the regression to have as sparse a model as possible [4]
- Linear Regression: an algorithm which attempts to fit an equation of the form  $Y = wX + c$  to the data so that its error to the true value is minimised as far as possible
- Normalised: the data is shifted so that its mean is zero and rescaled to move its standard deviation to 1. We are excluding the zero/one categorical columns from this, as it doesn’t make sense to include them.
- Random Forest Regression – a collection of decision tree regressors (hence the forest) that behaves as a single model as the average prediction is returned from the collection of decision tree results. Each individual tree is trained using “*bagging and random features*” [3]. A random subset of features is given to each tree.
- Ridge Regression – can create a parsimonious model (small and simple model) where the number of features is greater than the number of rows. “*Adds just enough bias to make the estimates reasonably reliable*” [5] (paraphrased and directly quoted)

## 2 Intermediate results

Type	Hyperparameters	RMSE	MAE	Accuracy (1-NMSE)
Linear Regression	Default (columns: model, year, mileage, fuel type, MPG, engine size)	4782.14	3602.55	-0.04
Linear Regression	Default (columns: model, year, mileage, fuel type, MPG, engine size), Normalised	2427.54	1717.89	0.73

Linear Regression	Lambda', 0.0002972, 'Learner', 'least-squares', 'Regularization', 'ridge', 'Solver', 'bfgs' (columns: model, year, mileage, fuel type, MPG, engine size), Normalised	1828.18	1346.7	0.85
Linear Regression	Lambda', 0.0002972, 'Learner', 'least-squares', 'Regularization', 'ridge', 'Solver', 'bfgs' (columns: model, year, mileage, fuel type, MPG, engine size), Normalised, with validation data	1837.15	1350.96	0.85
Linear Regression	Lambda', 0.000010015, 'Learner', 'least-squares', 'Regularization', 'ridge', 'Solver', 'bfgs' (columns: model, year, mileage, fuel type, MPG, engine size), Normalised, validation	1859.48	1358.33	0.84
Linear Regression	Lambda', 0.000010015, 'Learner', 'least-squares', 'Regularization', 'ridge', 'Solver', 'bfgs' (columns: model, year, mileage, fuel type, MPG, engine size), Normalised	1826.26	1347.21	0.85
Linear Regression	Lambda', 0.000010015, 'Learner', 'least-squares', 'Regularization', 'ridge', 'Solver', 'bfgs', 'Kfold', 5 (columns: model, year, mileage, fuel type, MPG, engine size), Normalised (1st model)	1818.02	1344.83	0.85
Linear Regression	Lambda', 0.000010015, 'Learner', 'least-squares', 'Regularization', 'ridge', 'Solver', 'bfgs', 'Kfold', 5 (columns: model, year, mileage, fuel type, MPG, engine size), Normalised (average of models)	1826.27	1347.26	0.85
Random Forest	Default (columns: model, year, mileage, fuel type, MPG, engine size)	1279.66	889.33	0.93
Random Forest (technically boosted forest)	MinLeafSize, 31, Method, LSBoost, NumLearningCycles, 193, LearnRate 0.38119. (columns: model, year, mileage, fuel type, MPG, engine size).	1185.69	837.56	0.94
Random Forest	MinLeafSize, 31, Method, Bag, NumLearningCycles, 193, (columns: model, year, mileage, fuel type, MPG, engine size).	1398.75	968.24	0.91

Random Forest	MinLeafSize, 1, Method, Bag, Num-LearningCycles, 499, (columns: model, year, mileage, fuel type, MPG, engine size).	1175.02	838.02	0.94
Random Forest	MinLeafSize, 1, Method, Bag, Num-LearningCycles, 450, (columns: model, year, mileage, fuel type, MPG, engine size).	1175.03	838.63	0.94
Random Forest	MinLeafSize, 1, Method, Bag, Num-LearningCycles, 50, (columns: model, year, mileage, fuel type, MPG, engine size).	1178.42	839.35	0.94
Random Forest	MinLeafSize, 1, Method, Bag, Num-LearningCycles, 500, (columns: model, year, mileage, fuel type, MPG, engine size).	1174.68	838.28	0.94
Random Forest	MinLeafSize, 1, Method, Bag, Num-LearningCycles, 499, (columns: model, year, mileage, fuel type, MPG, engine size).	1177.4	839.2	0.94
Random Forest	MinLeafSize, 1, Method, Bag, Num-LearningCycles, 499, (columns: model, year, mileage, fuel type, MPG, engine size). Kfold (average of models)	1182.52	843.05	0.94
Random Forest	MinLeafSize, 1, Method, Bag, Num-LearningCycles, 50, Kfold, 5, CrossVal, On, (columns: model, year, mileage, fuel type, MPG, engine size). Kfold (average of models)	1185.95	842.7	0.94
Random Forest	MinLeafSize, 1, Method, Bag, Num-LearningCycles, 100, (columns: model, year, mileage, fuel type, MPG, engine size).	1172.39	836.57	0.94

### 3 Implementation Details

For the implementation of the top models, we chose to use a single script named main to hold the majority of the code. This was to enable the easy running of all sections of code at once. This was also so that it was straightforward to run all of the code from top to bottom and reset all variables so that previous runs couldn't pollute the next run. By setting the data transformations to run each time the code ran we could ensure that our dataset remained pure and allowed for experimentation (such as how changing the hybrid's MPG from 202 to different numbers affected the models (it made the linear regression better and random forest worse)).

We used the `rng` function to set the random number seed so that all of the results were reproducible (this has the risk of the models being trapped in local minimums rather than finding the best parameters). Using the seed of 42 rather than 52 gave an RMSE of 1799.16 for Linear Regression (versus 1818.02) and 1120.16 for Random Forest (versus 1172.39). So, there is a much bigger effect on Random Forest, but not much difference for Linear Regression (either way Random Forest performed much better than Linear Regression).

For both models we excluded the tax and transmission type columns. For tax it was because it is derived from other columns and for transmission type it is because it had little effect on the price column.

For linear regression we normalised the training data and then normalised the test data using the same parameters. This dramatically increased the accuracy of the model (the RMSE went from £4782 down to £2428).

Using the script `OptimizeLinearRegression` we got the generalised hyperparameters for linear regression, then we used Bayesian optimisation to get a better lambda figure.

For random forest we used the function `fitrensemble` to train a collection of decision trees (we used a template tree so that we could adjust the minimum leaf size). We then used the script `OptimiseRandomForest` to find the best hyperparameters for the model. Then within the same script we used Bayesian optimisation to get the optimum number of trees.

To fairly analyse both models (so both were looked at the same way) we used the function `analyseRegression`, this worked out the MAE, RMSE and 1 - NMSE, then ran off the residual plots and other graphs.

To allow for some experimentation the code in main was sectioned so that the models could be run without needing to rerun the statistics and preparation sections. Long-running sections, such as Feature Importance, experiments with K Fold for Linear Regression and Random Forest and optimising the two models, were placed into their own code files and run once the main script had run. So that we could get summary statistics on the data we created the function `GetSummaryStats`, this was to recreate Python's `Pandas describe` function in MATLAB.

## References

- [1] Jason Brownlee. A gentle introduction to the BFGS optimization algorithm. URL <https://machinelearningmastery.com/bfgs-optimization-in-python/>.
- [2] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. An introduction to statistical learning with applications in r. URL <https://ebookcentral.proquest.com/lib/city/reader.action?docID=6686746>.
- [3] Schapire Robert. Random forests. URL <https://link.springer.com/content/pdf/10.1023/A:1010933404324.pdf>.
- [4] Stephanie. Lasso regression: Simple definition, . URL <https://www.statisticshowto.com/lasso-regression/>.
- [5] Stephanie. Ridge regression: Simple definition, . URL <https://www.statisticshowto.com/ridge-regression/>.