AutoTrader Vehicle Valuation Project

Jamie Atiyah

I. Data Processing

A. Data Understanding & Exploration

The provided data set contains a collection of 402005 anonymised adverts from AutoTrader. Each vehicle has 12 variables. Table XVII shows a sample of 5 vehicles before any data processing has been performed. Each vehicle has its own unique public reference which is set as the index of the data frame. Null values within the data are identified and displayed in table I. Figure 1 shows the distribution of mileage and the unique values of year. Mileage is shown to be a long-tailed distribution. In section II-C, the long-tailed distribution will be transformed. Erroneous values within the year column are highlighted in red in figure 1.

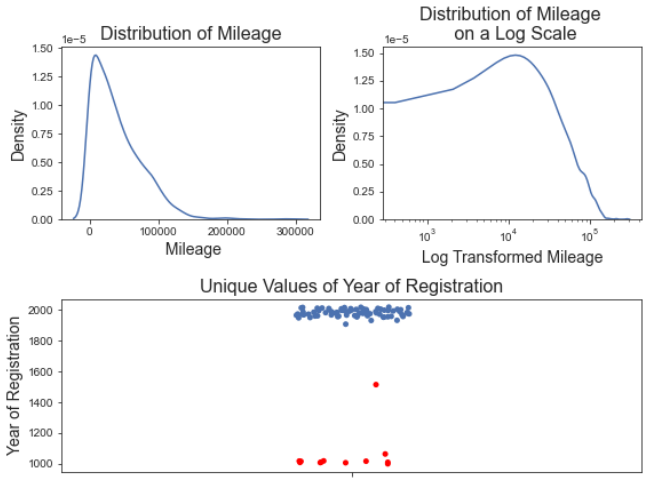

Figure 1: Distributions of Year and Mileage

Table I: Column Null Count and Description

Column	Non-Null Count (N)	Dtype	Description
mileage	401878	float64	Mileage driven
reg	370148	object	Registration code
colour	396627	object	Standard colour
make	402005	object	Standard make
model	402005	object	Standard model
condition	402005	object	Vehicle condition
year	368694	float64	Year of registration
price	402005	int64	Selling price
body	401168	object	Body type of the vehicle
car_van	402005	bool	Crossover car and van
fuel	401404	object	Fuel type of the vehicle

B. Detecting & Dealing with Outliers, Missing Values and Noise

Outliers, missing values and noise can cause issues in the machine learning process such as decreasing the normality of features and negatively impacting model performance. Sklearn models such as the RandomForest algorithm cannot support the presence of missing values. Therefore, processing outliers, missing values and noise is required. Outliers in the long tailed

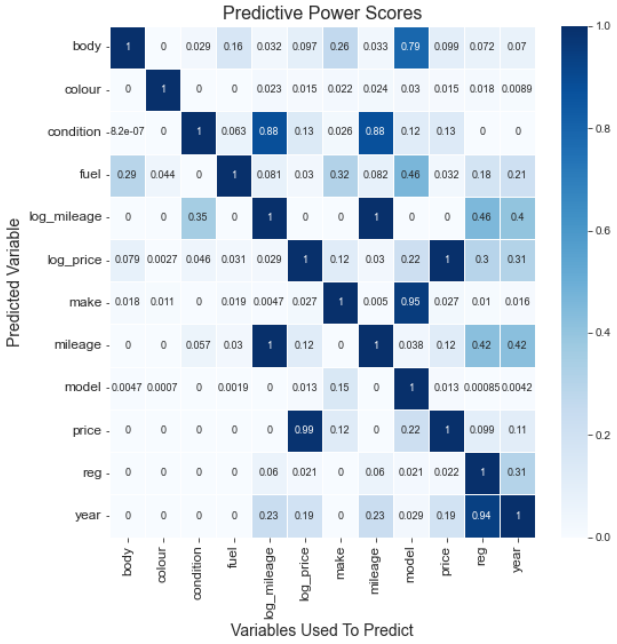

Figure 2: Predictive Power Matrix

mileage variable are identified and removed through the use of the interquartile range. Table I shows the number of null values in each of the columns in the data. Columns Mileage, reg, colour, year, body, and fuel all have null values. To identify the appropriate strategy to replace missing values, the predictive power of features was measured to identify any existing relationships between the variables. The ppscore package Wetschoreck and Krabel n.d. is used to get a correlation matrix. Higher values indicate better prediction. From figure 2, the following conclusions can be made:

- Reg code is a near perfect predictor of year.
- The model is a good predictor of the body and fuel.
- Year is the best predictor of mileage.
- Colour has little correlation with the other features.

The following strategies were adopted to process the data:

- Null year values for new condition vehicles were set to 2022.
- Erroneous values and other missing values within the year column were corrected using the the vehicles corresponding reg code as each reg code refers to a year AA n.d. By using the modulo 50 of the registration and adding 2000, the correct year of registration was found with any subsequent values exceeding 2022 being set to null.
- The remaining missing values in the year column were replaced using the median year.
- Null mileage values were replaced using the mean value of mileage relating for the vehicle's registration year.
- The mode colour was used to replace any null colour values.
- The mode body type for the vehicle's model replaced null body values. The remaining null body values were replaced with the mode body.

- The mode fuel type for the vehicle's model replaced null fuel values. The remaining null fuel values were replaced with the mode fuel.

II. Feature Engineering

Cardinality reduction II-A, feature creation II-D, feature transformation II-C and encoding II-B were used to enhance the feature vector.

A. Cardinality

High cardinality categorical features can cause issues in the machine learning process due to the issues associated with one-hot encoding (the chosen method of categorical encoding used in section II-B). If a categorical feature has too many classes, then having to create one-hot encoded variables for each of the classes may expose the models to the curse of dimensionality. Furthermore, if particularly rare categories exist within the unseen data, then trained pipelines may struggle with dealing with encoding the unseen rare categories. The columns colour, make, model, fuel, and body have 22, 103, 1112, 8, and 15 unique values respectively. The make and model columns will be target encoded in section II-B. The number of classes to preserve for the colour, body, and fuel columns is calculated using the `num_classes` function 1 which trains a random forest regressor on a copy of the dataframe that consists of mileage, year and the variable in question for each number within a user defined region of preserved classes and calculate the mean absolute error (mae). Cardinality was reduced as follows:

- 1 class was preserved in the colour column with all other colours being set to 'Other' XVIII.
- 4 classes were preserved in the body column with all other body types beings set to 'Other' XIX.
- Fuel types that included the word 'Hybrid' were grouped into a 'Hybrid' class.
- Fuel types that weren't Petrol, Diesel or Hybrid were grouped into 'Other'.

The cardinality of the condition column was increased by setting vehicles older than 30 years (age feature created in section II-D) to 'Vintage'. Vehicles younger than 5 years old but older than 0 years were set to 'Near-New'.

B. Encoding

Target encoding was used to transform the high cardinality make and model columns. To avoid any potential leaking of the target into the feature vector for the test data, the target encoder was fitted on the training data and is then used to transform any provided data set. The remaining categorical features were one-hot encoded.

C. Feature Transformation

The long-tailed distribution of mileage required transformation to boost performance for the models. Both the logarithmic and square-root transformation techniques were both investigated as transformation methods. Two pre-processing pipes were used to train and test on a sample of 3000 vehicles, one with a log transformation of mileage and another with a square-root transformation of mileage. The 10-fold cross validated scores of each model showed that the log transformation yielded better results XX.

The remaining numeric features age, miles per year (mpy) (see section II-D , target encoded make and model were scaled by using standardisation 1 by using the respective mean (μ) and standard deviation (σ).

$$Z = \frac{(X - \mu)}{\sigma} \tag{1}$$

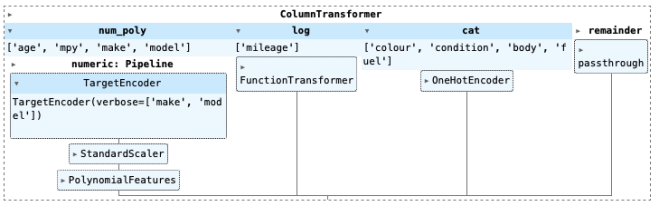

Figure 3: Pre-processor Pipeline

D. Feature Creation

The year of registration was engineered so that it represents the age of the vehicle by subtracting the year of registration from 2022. Mpy was calculated by dividing the mileage by the age (any vehicles that were age 0 had mpy set to mileage). Polynomial and interaction features were created to unveil complex relationships between the numeric features. The features used to create interaction and second-polynomial features were: age, mpy, target encoded make and model (see section II-B), and mileage.

The final pre-processor pipeline is displayed below in figure 3.

III. Feature Selection & Dimensionality Reduction

A. Feature Selection

The car and van column was dropped from the data set as 99.55% of the vehicles had a value of true, resulting in little information to be gained from the variable. As proven by the predictive power scores in figure 2 and domain knowledge, information stored within the reg code column directly correlates to the year of registration column, therefore, to avoid multicollinearity, the reg code column was also dropped.

After performing the feature engineering steps outlined in section II (figure 3), there are 33 features within the feature vector. Feature selection was required to reduce the amount of features in the feature vector with the aims of avoiding overfitting, multicollinearity, and removing uninformative features. The automatic feature selection tool `SelectKBest` from sklearn is used to select the k best features. The value of k was determined by fitting a random forest regressor on the pre-processor pipeline with the 5, 10, 15, 20 and 25 best features selected by the selectKBest function and finding the 5-fold cross validated maes on the train and test data sets. Figure 4 shows the mae on both the train and test data sets for varying values of k. A value of $k = 10$ is chosen as the mae for the test data stays relatively constant after a value of 10 but is much higher at a value of 5. The selected features are: age, make, model, mpy interaction with model, target encoded make squared, target encoded make interaction with target encoded model, target encoded model squared, log transformed mileage, one-hot encoded condition = used, one-hot encoded body = Hatchback.

B. Dimensionality Reduction

Whilst the unselected features could be disregarded from the feature vector entirely, it is hypothesised that there is important value within these features. Dimensionality reduction in the form of principal components analysis is employed to reduce the dimensionality of the unselected features. Figure 5 shows the

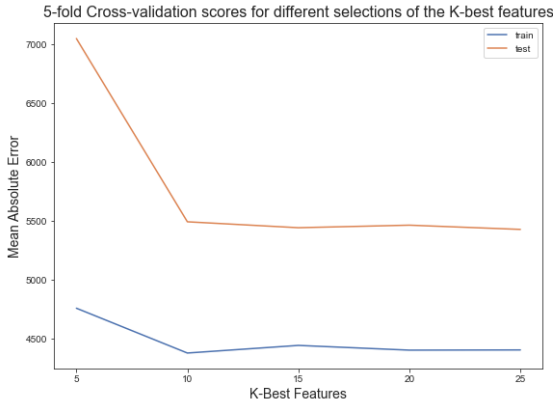

Figure 4: MAE for Different Values of k

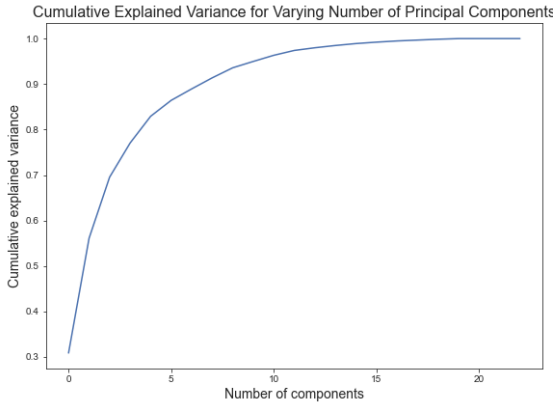

Figure 5: Cumulative Explained Variance for Varying Number of Principal Components

cumulative explained variance for varying number of principal components. With the aim of selecting an appropriate number of principal components that maximises the explained variance so as to retain as much information from the fitted variables, whilst keeping the number of components low, a value of 3 components is selected as the increase in explained variance slows down after 3 components. Assessing the influence of features on the principal components will add some interpretability to the components, which will be important when assessing feature importance in a later section. Figure 6 shows the influence of the unselected features in the 3 components. Second-degree polynomial of age has great influence in the first component, whilst second-degree polynomial of mpy has great influence in the second component. Interaction features age make, age model and mpy make have influence on the third component. One-hot encoded variables have little influence across all principal components.

To assess if adding the 3 principal components that are fitted with the unselected features adds any value to the feature vector, machine learning models random forest regressor, gradient boosting regressor and histogram gradient boosting regressor were fitted on a feature vector with the components and also without the components. Figure 7 shows the mae for the three models on the training and test data for the different feature vectors. The plot shows an increase in performance across all models when the principal components are included in the feature vector. Therefore, the 3 principal components of the unselected features are added to the feature vector. Whilst it could be argued that assessing whether the entire feature vector should be reduced into principal components could provide potential gain in terms of model performance, the model may also lose vital interpretability and explainability.

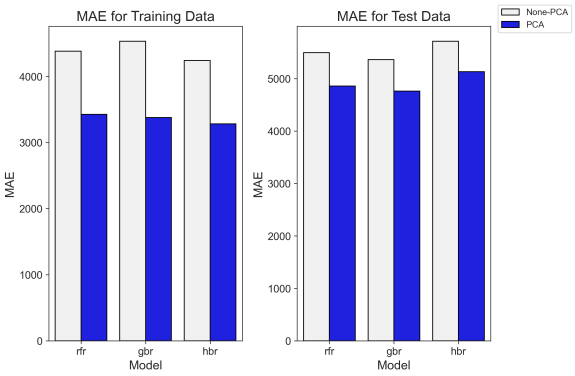

Figure 7: MAE for Models Fitted With & Without 3 Principal Components

Creating additional features for the feature vector by using unsupervised learning was attempted with the aim of capturing complex relationships between the features. However, no gain in performance was found and the clustering was ineffective in separating the data points into distinct clusters as the silhouette scores had a maximum value of 0.5 (see figure 14).

IV. Model Building

A. Train & Validation & Test Folds

20% of the data was assigned as test data to obtain the final performance of the selected model. For the pro-typing and experimental model building stage, a sample of 3000 vehicles was used to save on computational time. The train and validation folds were split in a 67%, 33% split. The final model is trained on the entire 80% of the data.

B. Dummy Regressor

Several models were built in this project. The types of these models include linear models, random forest, and boosted trees. In addition, ensembling methods voter and stacker techniques are tried and tested. Sklearn models are utilised for consistency. To obtain a baseline performance for predicting price, a dummy regressor is fitted which predicts the mean value of price for every vehicle. Table II shows the performance of the fitted dummy regressor.

Table II: Dummy Regressor Performance

Model	Data Set	MAE	RMSE
Dummy Regressor	Training	10602.34	17273.65
Dummy Regressor	Test	10772.93	19754.71

C. Linear Models

The sklearn linear models are displayed in table III.

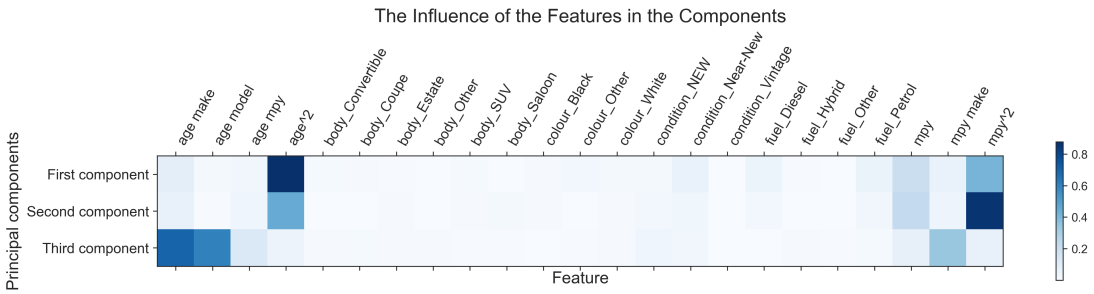

Figure 6: Influence of Unselected Features in the Components

Table IV: Linear Model Performances

Model	Data Set	MAE	RMSE
Linear Regression	Train	4659.56	7811.34
Linear Regression	Test	5911.14	11905.79
Ridge	Train	4657.05	7808.59
Ridge	Test	5839.84	11837.71
Lasso	Train	4642.90	7811.62
Lasso	Test	5767.86	11838.75

Table V: Ridge Grid Search Results

Alpha	Mean Test Score	Std Test Score	Mean Train Score
10.0	-7932.87	1270.80	-7777.72
100.0	-7937.10	1318.91	-7813.48
1.0	-7937.12	1267.52	-7776.93
0.1	-7937.66	1267.24	-7776.92
0.01	-7937.71	1267.22	-7776.92

Table III: Sklearn Linear Models

Model	Description
Linear Regression	A linear model fitted to minimise the residual sum of squares
Ridge	Minimise the least squares. Regularisation is given by the l2-norm
Lasso	Trained with the L1 prior as regulariser

A Grid Search is utilised to find the optimal parameters of the Ridge and Lasso models. 7 values (0.001, 0.01, 0.1, 1, 10, 100, 1000) of the regularisation hypterparameter α were included in the parameter grid for both models. The small parameter grid, whilst being relatively simple in comparison to parameter grids use for other models trained in a later section, are computationally inexpensive. The optimal value of the hyperparmater was determined by obtaining the lowest 5-fold cross validated score. Table V shows the performance of the ridge model for the varying combinations of parameters. A value of 10 for $\alpha = 10$ is shown to be optimal. Table IV shows the 10-fold cross validated mae and rmse for the models. The linear regression model and the hyper-parameterised ridge and lasso models are all shown to slightly over-fit the data as the respective performance metrics are better on the training data compared to the test data for each model. The best performing linear model is shown to be the ridge model.

D. Random Forest Regressor

A random forest regressor was trained on the data set. In contrast to the simple parameter grid of the linear models that were discussed in section IV-C, there are many hyper-parameters to be fine tuned in the random forest regressor. The chosen hyper-parameters to be optimised are shown in table VI. If a grid search was used on this parameter grid, there would be a total of 608 different configurations. Therefore, to save on computational time, a randomised grid search is used which takes a random sample of the parameter space and fits models for the different random configurations. The number of iterations was set to 50, resulting in fitting 5 folds for each of the 50 parameter values, totalling 250 fits. From looking at the 5-fold cross validated scores of the randomised grid search, the following conclusions were made:

- Oob score can be set to false.
- 500 estimators should be used.
- Only 1 sample per leaf is required.
- The max depth can be set to 80.

A grid search was performed on the remaining hyper-parameters: max features, min samples split, and bootstrap. Setting bootstrap to true and the minimum samples to split a node as 2 is shown as being optimal (table VIII). There is no value for max features that seems optimal therefore the default value of 'sqrt' will be used. The performance of the hyper-parameterised is shown in table IX. The random forest model is a clear improvement on the dummy regressor models and performs well.

Table VII: Random Forest Parameter Grid

Parameter	Values	Description
n estimators	100, 150, 200 250, 300, 350, 400, 450, 500	The number of trees in the forest
max features	sqrt, log2	The number of features to consider when looking for the best split
min samples split	2, 6, 10, 14	The minimum number of samples to split a node
bootstrap	True, False	Boolean for deciding if bootstrap samples are used
oob score	True, False	Boolean for deciding if out-of-bag samples are used to estimate generalisation score
max depth	80, 90, 100, 110	Maximum depth of the trees in the forest
min samples leaf	1, 2, 6, 10, 14	Minimum number of samples at a leaf node

Table VIII: Random Forest Grid Search

Bootstrap	Max Features	Min Samples Split	Mean Test Score	Rank Test Score
True	sqrt	2	-6302.045275	1
True	log2	2	-6302.045275	1
False	sqrt	2	-6343.367790	3

Table IX: Random Forest Performance

Model	Data Set	Mae	RMSE
Random Forest	Train	3230.42	6103.40
Random Forest	Test	4237.85	8997.87

E. Boosted Trees

1) Gradient Boosting Regressor: Many different libraries offer boosted tree models that could be used such as XGBoost, Light-

Table VI: Random Forest Randomised Grid Search

Oob Score	N Estimators	Min Samples Split	Min Samples Leaf	Max Features	Max Depth	Bootstrap	Mean Test Score	Rank Test Score
False	500	2	1	log2	80	True	-6302.05	1
False	500	6	1	sqrt	80	False	6362.63	2
True	500	6	1	sqrt	110	True	-6415.49	3

GBM, and CatBoost. However, for consistency, the boosted tree models used in this project come from the sklearn library. The boosted tree models employed are the gradient boosting regressor and the histogram-based gradient boosting regressor. There are many hyper-parameters available in both boosting models. The parameter grid used for the gradient boosting model is displayed in table XI. As previously done for the random forest model, 50-iterations of 5-fold randomised grid search fits are used to save on computational expense resulting in an overall total of 250 fits. Table X shows the results of the randomised grid search. The following conclusions were made:

- 400 - 450 estimators should be used.
- The minimum samples at a leaf node should be used.
- Only 1 sample per leaf is required.
- A learning rate of between 0.05 and 0.1 should be used.

A grid search of 5 folds for all remaining 60 candidates was used to find the optimal values for n estimators, max depth, min samples split and the criterion. Table XII indicates the following:

- The friedman mse criterion should be used.
- A max depth of 110 is optimal.
- 5 samples are required to split a node.
- 450 estimators is optimal.

Table XI: Gradient Boosting Regressor Parameter Grid

Parameter	Values	Description
N Estimators	100, 150, 200, 250, 300, 350, 400, 450	The number of boosting stages
max depth	None, 10, 0, 50, 70, 90, 110	The maximum depth of trees
learning rate	0, 0.05, 0.1, 0.2, 0.4, 0.5, 1	Varies the contribution of each tree
min samples leaf	1, 2, 3, 4, 5	The minimum samples at a leaf node
min samples split	1, 2, 3, 4, 5	The minimum samples to split the leaf node
criterion	friedman_mse, squared_error	The measurement of the quality of the leaf node split

Table XII: Gradient Boosting Regressor Grid Search

Criterion	Max Depth	Min Samples Split	N Estimators	Mean Test Score	Rank Test Score
friedman mse	110	5	450	-6533.15	1
friedman mse	30	3	400	-6551.90	2
squared error	110	3	400	-6584.15	3

2) Histogram-Based Gradient Boosting Regressor: A histogram-based gradient boosting regressor is also fitted. The optimal value of 4 hyper-parameters were found using 50 iterations of randomised grid-search. The best parameters were found to be:

- Maximum of 1000 iterations.
- Maximum depth of 50.
- Maximum number of 250 bins
- Learning rate of 0.05.

The performance of the boosted tree models are displayed in table XIII. Both models exceed the performance of the linear models. However, the performance is less than that of the random forest model.

Table XIII: Boosted Tree Model Performance

Model	Data Set	MAE	RMSE
Gradient Boosting Regressor	Train	3397.24	6401.82
Gradient Boosting Regressor	Test	4916.50	11322.22
Histogram-Based Gradient Boosting Regressor	Train	3208.67	6186.59
Histogram-Based Gradient Boosting Regressor	Test	4887.89	10348.15

F. Model Ensembling

To further increase performance, model ensembling techniques such as voting and stacking. From the performances shown in sections IV-C, IV-D, and IV-E, the chosen models to ensemble are the boosted tree models and the random forest model.

1) Voter: The first ensemble method is the voting regressor. The voting regressor looks to average the predictions of each of the models that are being ensembled.

2) Stacker: The stacker ensemble method is also assessed. The stacking regressor uses the outputs of models to be used as input in a final regressor. Due to the random forest regressor being the best performing model, the random forest regressor is set as the final regressor whilst the boosting tree models are set as the estimators which provide the inputs to the random forest model.

The performances of both the voter and the stacked ensemble models are displayed in table XIV. The voter ensemble method is shown to provide the best results across both data sets and performance metrics.

V. Model Analysis

Analysing model performance can help provide better explainability and interpretability to stakeholders. In the event of significant errors that lead from the use of black box machine learning models, model explainability is highly important to identify the cause of such error. Identifying important variables can be useful information to stakeholders.

A. Shap Values: Global Explanations

The importance of the features was assessed by plotting the Shap values. The global Shap values can identify the relationship

Table X: Gradient Boosting Regressor Randomised Grid Search

N Estimators	Min Samples Split	Min Samples Leaf	Max Depth	Learning Rate	Criterion	Mean Test Score	Mean Train Score	Rank Test Score
450	5	5	110	0.1	friedman_mse	-6533.15	-200.62	1
400	3	5	30	0.05	friedman_mse	-6551.90	-349.91	2
400	3	5	110	0.1	squared_error	-6584.15	-203.28	3

Table XIV: Ensemble Model Performances

Ensemble Method	Data Set	MAE	RMSE
Voter	Train	3112.70	5904.03
Voter	Test	4398.04	9803.18
Stacked	Train	3735.15	7428.84
Stacked	Train	5543.96	13435.36

with the features within the feature vector with the target variable. The Shap explainer was fitted with the predicted values from the random forest model with the fine-tuned hyper-parameters discussed in section IV-D. Figure 9 displays the Shap summary plot. The features are ordered in terms of importance according to the Shap values with features higher up being of more importance. The target encoded model feature is shown to be of the highest importance, with the numeric age and log transformed mileage being the 2nd and 3rd most important features according to Shap. The 3 principal components that were created to reduce the dimensionality of the unselected features by the automated feature selection are shown to be of low importance. Interestingly, the 3rd principal component is of higher importance than the other two principal components. Referring back to figure 6, this further shows the target encoded model and numeric age and log-transformed mileage to have a strong effect on the target, as the third component is heavily influenced by interactions which include age miles, and target encoded make and model.

B. Shap Values: Local Explanations

More localised plots derived from the shap values can offer interesting perspectives as to why certain vehicles are being predicted to be a certain price. Figure 8 shows the effect of each feature on the predicted value for one observation in the test data. Log transformed mileage has the highest value but only causes a small change in this instance. Target encoded make and model change the predicted price in opposite directions but with similar magnitude. Most of the values in the features for this observation cause a decrease in prediction in price.

C. Partial-Dependency Plots

Partial-dependency plots can be highly useful in highlighting relationships between individual features and the target. Figure 10 shows the partial dependence plots for the scaled age and log transformed mileage. Between the values of -1.5 and 1, there is the biggest change in the target for the age feature, with the target value decreasing as the age increases. For the log-transformed mileage, there is less of a sudden decrease in the smaller values of mileage, however there is still a sizeable decrease with very high values of mileage. An assessment of the feature interactions with the partial dependence shows high value cars are being predicted with both lower values of age and mileage. One clear issue with figure 10 is the scales used for the features, making it less clear as the original scale of the variables has been removed.

D. Permutation Importance

Feature importance can be assessed by using permutation importance which shuffles features and assess the effect on performance. The voter ensemble is trained and weights of each feature can be calculated from the permutations. From table XV, target encoded model is shown to be highly important whilst the one-hot encoded classes body hatchback and condition used are shown to have little importance.

E. Real vs Prediction

Figure 11 shows the distribution of vehicle price against the voter ensemble predictions. The model performs consistently when predicting the vehicles that are priced between 0 and 30,000.

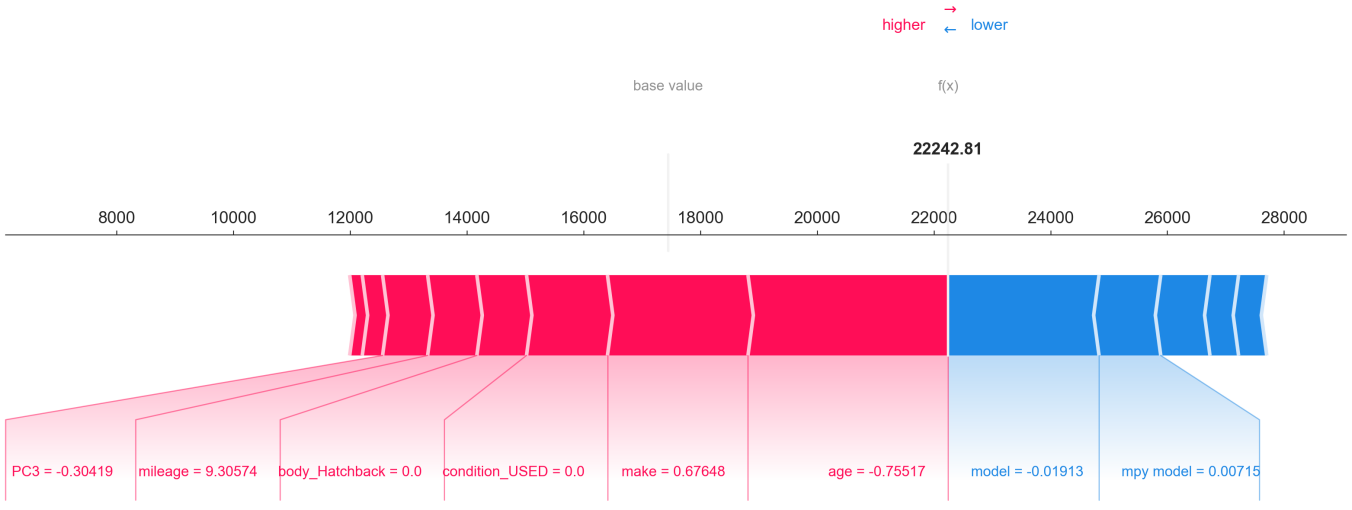

Figure 8: Localised Shap Values

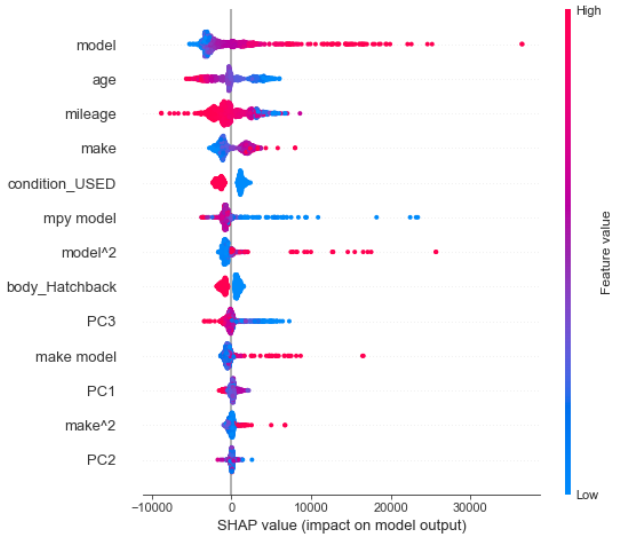

Figure 9: Shap Summary Plot

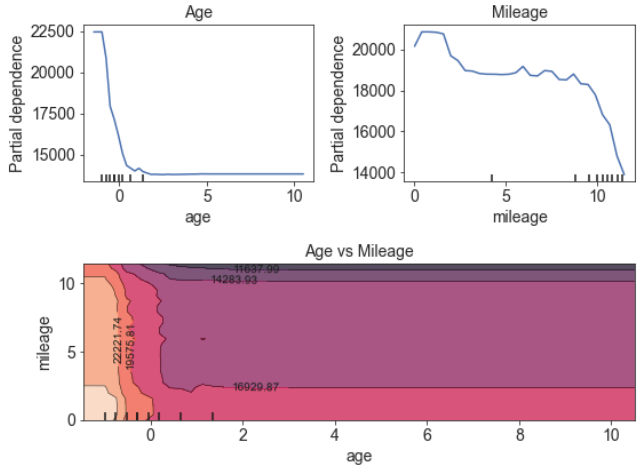

Figure 10: Partial Dependence Plot

Table XV: Permutation Importance

Weight	Feature
0.5588 ± 0.0333	Model
0.0951 ± 0.0150	Mpy Model
0.0908 ± 0.0038	Age
0.0765 ± 0.0070	Mileage
0.0471 ± 0.0031	PC3
0.0297 ± 0.0017	Make
0.0251 ± 0.0062	PC1
0.0239 ± 0.0004	$Model^2$
0.0238 ± 0.0004	Make Model
0.0124 ± 0.0033	PC2
0.0069 ± 0.0003	$Make^2$
0.0059 ± 0.0004	Body Hatchback
0.0027 ± 0.0003	Condition Used

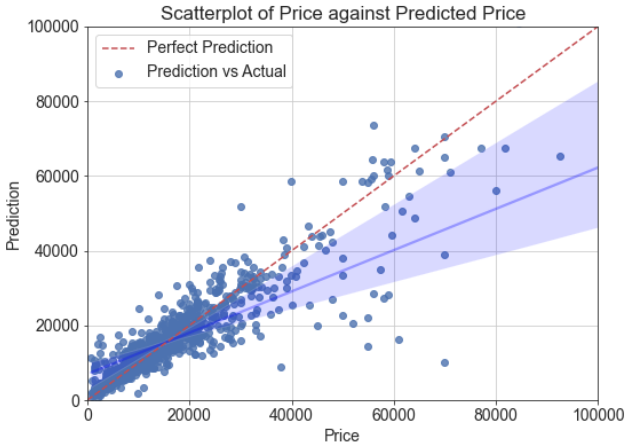

Figure 11: Price against Predicted Price

However, for higher value vehicles, the predictions become more variable and deviate more from the actual value. This may be due to the low sample size of high priced vehicles in comparison to the abundance of vehicles that have low prices.

F. Residual Analysis

Analysing the residuals can provide interesting information as to why certain vehicles are being predicted poorly. The absolute values of the residuals are calculated by taking the absolute value of the difference between the predicted value by the voter ensemble model and the actual value. To get a sense of why particular vehicles are being poorly predicted, vehicles that have an absolute value of a residual that is greater or equal to 9000 are classed as high residuals. Figure 12 shows the distribution of mileage and age. The values of age and mileage are transformed so that they are on their original scale for easier explainability by using the inverse transform feature of the transformation pipelines. The plot suggests that vehicles of low ages are being poorly predicted. The plot suggests that vehicles with low mileage are being predicted poorly.

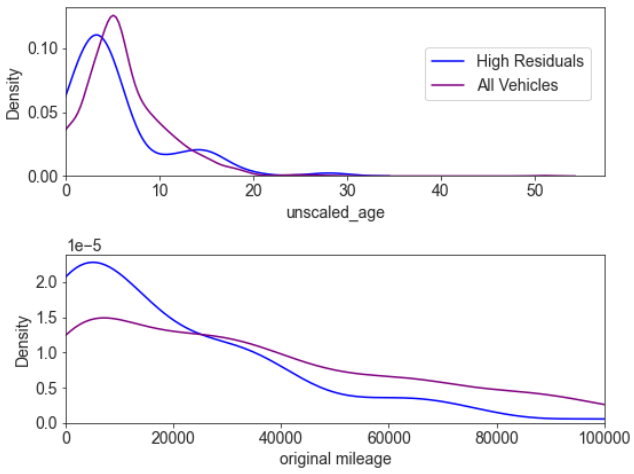

Figure 12: Residual Analysis

Table XVI: Model Performance on the entire data.

Ensemble Method	Data Set	MAE
Voter	Train	2932.47
Voter	Test	3184.38

Figure 13 shows the mean residuals for the 5 most common makes. Whilst the mean residuals for all 5 makes are below the 9000 threshold to determine them as being high residuals, there is a substantial difference in the average residuals between the groups.

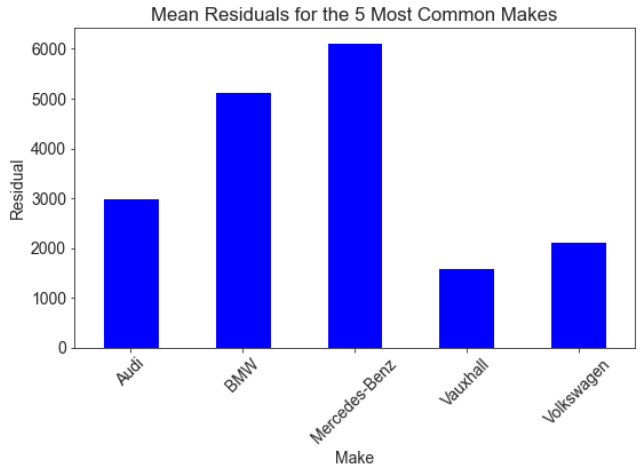

Figure 13: Residuals for Common Makes

VI. Applying the Model to the Entire Data set

To obtain the finalised performance of the voter ensembled model outlined in section section IV-F1 was applied. However, to save on computation, default parameters for the random forest, gradient boosting regressor, and histogram-based gradient boosting regressor were used to prevent the ensemble model from being too complex. The simpler ensemble model was trained on 80% of the data and was tested using test data which accounts for 20 % of the data. Table XVI shows the 10-fold cross validated mae of the voting ensemble for the train and test data. The model performs well and generalises well to the test data.

VII. Conclusion

Throughout this project, a machine learning workflow was applied with the aim of creating a regression model for predicting the selling price of a vehicle. The best performing model was an ensembled voter model which took the average prediction from three models that had been fine-tuned with optimal hyper-parameters: the random forest, gradient boosting regressor, and a histogram-based gradient boosting regressor. There was significant value in predictive performance from target encoding the make and in particular the model of a vehicle. Not all variables were of significant use, such as the crossover car and van variable. The log-transformed mileage and age were shown to have importance in predicting selling price. In particular, a relationship was identified that shows how the features interact with each other to help accurately predict price.

Future work concerning the use of machine learning methodology may want to explore methods of improving the available features. In particular, whilst the cardinality of the vehicle condition variable was increased by adding classes near-new and vintage, there may be better methods to obtain more accurate descriptions of a vehicle. One solution could involve the use of deep-learning models such as the VGG-16 (Simonyan and Zisserman 2014) and ResNet50 (He et al. 2016), for image classification from a sellers uploaded image of the vehicle to recognise if a vehicle has been scratched, damaged, or tainted. An alternative method to obtain more accurate results could be to employ more advanced hardware such as supercomputers like Archer2 ARCHER2 UK National Supercomputing Service n.d. to further fine-tune the hyper parameters of the models and to ensemble this fine-tuned models. As always, the trade-off with computational expense must be acknowledged.

References

AA, The (n.d.). *How to read number plates*. https://www.theaa.com/car-buying/number-plates. Accessed: 2022-04-28.

ARCHER2 UK National Supercomputing Service (n.d.). https://www.archer2.ac.uk.

He, Kaiming et al. (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Simonyan, Karen and Andrew Zisserman (2014). "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556*.

Wetschoreck, Florian and Tobias Krabel (n.d.). *Ppscore*. https://pypi.org/project/ppscore/. Accessed: 2022-04-28.

Table XVII: AutoTrader Data Set

Public Reference	Mileage	Reg	Colour	Make	Model	Condition	Year	Price	Body	Car_Van	Fuel
202007211517817	35385.0	17	Blue	Renault	Captur	USED	2017.0	8480	SUV	False	Petrol
202010104838190	10.0	NaN	Brown	Honda	CR-V	NEW	NaN	25990	SUV	False	Petrol
202010144993195	7312.0	69	Blue	BMW	5 Series	USED	2020.0	29450	Estate	False	Diesel Hybrid
202009294347575	44846.0	64	White	MINI	Hatch	USED	2014.0	7985	Hatchback	False	Diesel
202007151279199	116000.0	10	Black	Audi	A3	USED	2010.0	3995	Hatchback	False	Diesel

Appendix A
Tables

Table XVIII: mae for Varying Number of Preserved Classes in the Colour Variable.

Number of Classes Reduced	mae $_{2dp}$
1	11914.46
2	11987.52
3	12235.76
4	12235.76
5	12235.76
6	12235.76
7	12235.76
8	12235.76
9	12235.76
10	12235.76

Table XIX: mae for Varying Number of Preserved Classes in the Body Variable.

Number of Classes Reduced	mae $_{2dp}$
1	10768.13
2	10643.10
3	10771.64
4	10217.71
5	10818.63
6	10495.72
7	10395.03
8	10669.95

Table XX: Root Mean Squared Error for Different Transformations

Model	Root Mean Squared Error
Log Transformation	14959.72
Square-Root Transformation	15187.76

Appendix B
Graphs

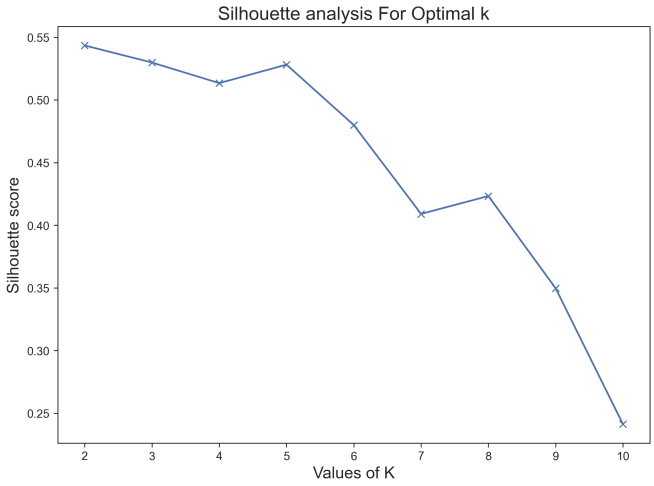

Figure 14: Silhouette Analysis for Optimal K

Appendix C
Code

```
def num_classes(variable, df, minimum, maximum):

    loss_scores = {}

    for num_classes in range(minimum, maximum+1):

        # Create a copy of the data frame

        df_copy = df.copy()
        df_copy.loc[~df_copy[variable].isin(df_copy[variable].value_counts().head(num_classes).index.tolist()),variable] = 'Other';

        # one-hot encode the categorical features

        cat_features = df_copy.select_dtypes(include = 'object').columns
        df_copy = pd.get_dummies(df_copy, columns=cat_features)

        # Use train_test_split to split the data into train and test folds with a random_state for reproducability

        X = df_copy.drop(columns='price')
        y = df_copy['price']

        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

        # Fit a random forest regressor with 100 estimators on the training data

        rf = RandomForestRegressor(n_estimators=100, random_state=42)
```

```
        rf.fit(X_train, y_train)

        # Calculate the mae on the test data set
        y_pred = rf.predict(X_test)
        mae = mean_absolute_error(y_test, y_pred)

        # Add to the loss score dict
        loss_scores[num_classes] = mae

    return loss_scores
```

Listing 1: Number of Classes to preserve for Cardinality Reduction