



Car Price Comparison and Prediction

Final Report

Word Count: 3370

Higher Diploma in Science in Data Analytics

Name: Thomas Quinn

Student Number: 10521715

Email: 10521715@mydbs.ie

Supervisor: Charles Nwankire

Course: Higher Diploma in Science in Data Analytics

Contents

Car Price Comparison and Prediction	1
Abstract.....	3
Acknowledgements.....	4
Chapter 1: Introduction	5
Student's Learning Objectives	5
Chapter 2: Background	6
Data File	6
Literature Review	6
Multiple Linear Regression	6
Linear regression	7
Chapter 3: Requirements Specification and Design	8
Technical Specification of the Project.....	8
Chapter 4: Implementation	9
Data Preparation process	9
Python data transformation	11
Code Walkthrough	11
Decision Tree Algorithm Vs Gradient Boosting Machines	14
Auto Model	15
Abbreviations	15
Chapter 5: Testing and Results	17
Auto Model Results.....	17
Attribute Correlation to Price	18
Analysis of price prediction error.....	19
Chapter 6: Conclusions and Future Work.....	21
References and Bibliography	22
Appendices.....	23
Data Preparation Code.....	23

Abstract

Using predictive modelling, descriptive analytics, and data analysis IT software, I wanted to conduct a study on the price of second-hand cars, noting what different factors had a more significant impact on the sale price. Using the functionality of Excel and Python code, I cleansed the dataset and had it prepared for machine learning. Using RapidMiner, several models were used to predict the value in the 'price' column for each record, to see which gave the most accurate results. I used Tableau to visualise the accuracy of the results. I then sought more data on car prices, in order to test out the algorithm obtained by the model. Using the correlation value of each column with price, I applied linear regression using the test dataset to see if a similarly good estimate of the prices could be given by the trained model on the new information.

Acknowledgements

Thanks to my personal supervisor, Charles Nwankire, for all the help and guidance with this project. He has always steered my focus in the right direction, suggesting simple solutions to issues that I had encountered and shifting the focus of this project from an academic perspective to a more business focused case study. He has been generous with his time and attention and we have had many conversations over the course of the last four months. His enthusiastic and positive outlook is much appreciated by me and I am very grateful for all his efforts.

Chapter 1: Introduction

My current car is old but reliable. It was suggested to me to sell my car and buy another. A second-hand car would be most affordable to me; however, I am unsure of what would be a fair price to pay for a certain car. Using predictive modelling, descriptive analytics, and data analysis IT software, I wanted to conduct a study on the price of second-hand cars, noting what different factors had a more significant impact on the sale price.

According to rte.ie, “... the number of used, or imported, cars licensed rose by 9.5% to 108,895 in 2019 compared with 99,456 in 2018.” “The CSO said that a total of 113,305 new private cars were licensed for the first-time last year, down 6.5% compared with 2018.”

Aim: to create a mathematical algorithm to estimate the price of a car, given its attributes.

Project Scope and Objectives

- Use linear regression to measure the ability to predict a car price based on the weighted sums of its attributes
- Examine the cross correlation, principal components to find which attributes provide us with the most information
- Use RapidMiner to trial the best methods for finding patterns in the dataset, with price as the target variable.
- Visualise our results on Tableau
- Apply algorithm to real world car price
- Use time series to estimate the prices of the cars in a year's time.
- Add more data from perhaps an Irish site, to this project, and test the prediction algorithm on it.

Student's Learning Objectives

Create a helpful tool using knowledge and skills gained from the Data Analytics modules.

Applying statistical models to the dataset and to gain an understanding of why a predictive model is effective or not.

Data cleansing methods such as one hot encoder, to convert data into a format that is easier to do predictive modelling with.

Improve my presentation aptitude by focusing in on how to explain the results in a universally understood way.

Chapter 2: Background

Data File

From Kaggle, I downloaded a csv file, **car_price**, from the forum, “*second-hand car price estimation*”. The data, which came originally from <https://www.carsales.com.au/>, contains cars registered from 1968 to 2018, a 50-year range. Prices range from 250 AUD (Australian Dollars) to 399,000 AUD. There are four car brands in the dataset – Honda, Mazda, Nissan, and Toyota.

Literature Review

Kaggle.com – Trying to predict used car value

Uploaded by Dan

<https://www.kaggle.com/ddmngml/trying-to-predict-used-car-value/notebook#Intro>

User Dan used Python to analyse a dataset of used car prices and car attributes. With car price as the target variable, she used a selection of Machine Learning algorithms and recorded her results. Her data was scraped with Scrapy from eBay Kleinanzeigen. She had the details of 370,000 used car that were advertised for sale.

Multiple Linear Regression

The general formula (link function) for linear regression is:

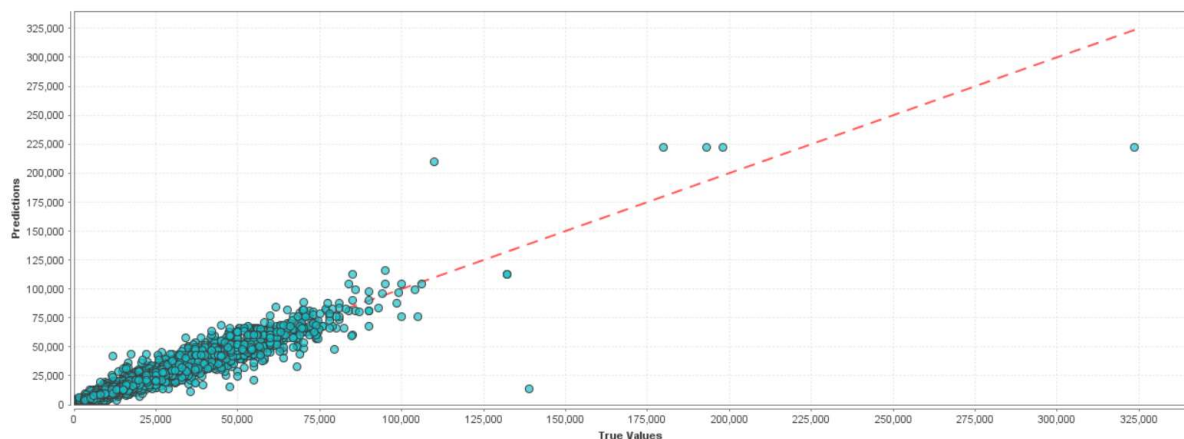
$$Y = \beta_0 + \sum X_i \beta_i + \epsilon_i$$

Where Y is the dependent variable, X_i 's are independent variables, β_i 's are weightings for each X_i , and ϵ_i represents noise in the dataset.

In the car price prediction database, the price prediction is the dependent variable, the car attributes are the independent variables and the correlations assigned to each attribute by the RapidMiner auto model, are the betas. A beta weight will be equal the correlation coefficient when we only have one target variable.

Fig 4f: Line of best fit for scatter plot observations.

Decision Tree - Predictions Chart



Linear regression

Line of best fit. A straight line that is plotted so that it minimises the sum of squared distances to each datapoint. Linear regression is the most popular statistical method for modelling the relationship between a set of dependent and independent variables. Predictions on the dependent variables can be made this way.

Chapter 3: Requirements Specification and Design

The business of buying and selling used cars is global and ever-present. Cars are an expensive commodity. This makes it critical for buyers and sellers to know how to value a used car's worth. Add to this the fact that there are several factors that can indicate a car's expected reliability and performance and you realise that the estimate or prediction of a car's price can be an effective decision-making criterion.

From Kaggle, I downloaded a csv file, **car_price**, from the forum, "*second-hand car price estimation*". The data, which came originally from <https://www.carsales.com.au/>, contains cars registered from 1968 to 2018, a 50-year range. Prices range from 250 AUD (Australian Dollars) to 399,000 AUD. There are four car brands in the dataset – Honda, Mazda, Nissan, and Toyota.

There are 13 columns and 55,870 rows in this dataset.

There is an id column, a date column, two numerical columns and nine categorical columns.

Fig 3a: Dimensions in original dataset.

Type	No.	Col name
String	9	brand, model, title, discount, body, transmission, engine, state, seller
Decimal	1	Year
Integer	2	Price, Odometre
Other	1	id

Using the car data form the csv, I will use RapidMiner to perform predictive modelling, with the price column as the target variable. This will use several built-in predictive models and will score their respective relative error of predictions. The correlation coefficients from the most accurate models will be used as weights in the linear regression link function to calculate price predictions by hand.

Technical Specification of the Project

All the materials required to conduct this study are available on a PC with internet access and a few software packages installed. The software used by me included:

- Visual Studio 2019 with Python 3.7 (32 Bit).
- Microsoft Office 2016: Excel, using CSV and xls format.
- Tableau 2019.3
- RapidMiner Studio Version 9.5: Auto Model function with deep learning and model simulator extensions.

Chapter 4: Implementation

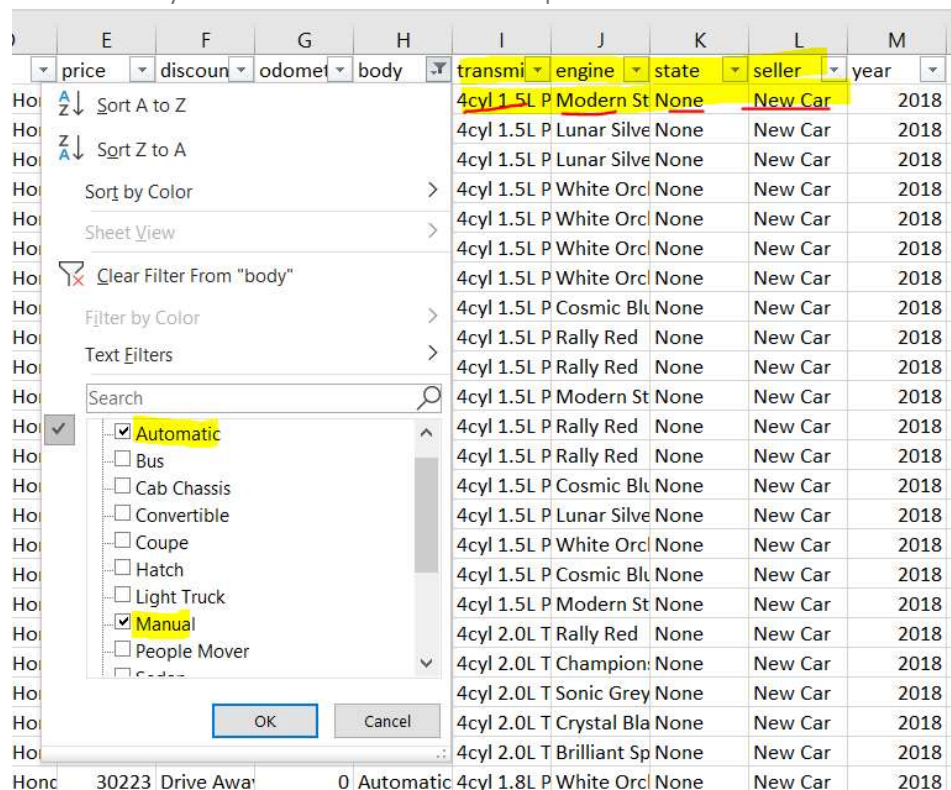
Data Preparation process

An immediate challenge I was confronted by was the condition of the dataset. Nine attributes are categorical, which makes predictive analysis much more challenging. Using OneHotEncoder, I created binary columns for each distinct piece of categorical data. It is recommended to use OneHotEncoder for columns with ten or less distinct categorical values. This became an issue as the 'engine' attribute column had 175 distinct values, so OneHotEncoder would not be practical. I used the Excel function, 'text to columns', to split this column, using white spaces as the delimiter. Four new columns were created, which I named 'Cylinder', 'Litre', 'Fuel' and 'Fuel 2'. For example, '4cyl 1.2L Turbo Petrol' was split into the four columns, in that order.

Further issues had arisen. I needed to treat 'Turbo' and 'Diesel' as one attribute, despite the white space between them in the initial column. Same scenario for Turbo and Petrol, and S and Petrol. These values were taken out of the 'Fuel 2' column and were combined with the data in the fuel column. There were also values that did not seem to describe the characteristics of a car engine, rather they seemed to describe colours. After changing the 'Fuel 2' column, only the 'colours' were left. I renamed this column 'Colour'.

Another issue was that the 'transmission' column, which was to the left of the 'engine' column, contained data values that should have been in the engine column. It appeared that on certain rows, a few data points were erroneously placed in the column to the left of where the data belonged. The offending columns were body, transmission, and engine.

Fig 5a: Our data issue. Only manual or automatic should be options for transmission.



	E	F	G	H	I	J	K	L	M
	price	discount	odomet	body	transmi	engine	state	seller	year
Ho					4cyl 1.5L P	Modern St	None	New Car	2018
Ho					4cyl 1.5L P	Lunar Silve	None	New Car	2018
Ho					4cyl 1.5L P	Lunar Silve	None	New Car	2018
Ho					4cyl 1.5L P	White Orcl	None	New Car	2018
Ho					4cyl 1.5L P	White Orcl	None	New Car	2018
Ho					4cyl 1.5L P	White Orcl	None	New Car	2018
Ho					4cyl 1.5L P	White Orcl	None	New Car	2018
Ho					4cyl 1.5L P	Cosmic Bl	None	New Car	2018
Ho					4cyl 1.5L P	Rally Red	None	New Car	2018
Ho					4cyl 1.5L P	Rally Red	None	New Car	2018
Ho					4cyl 1.5L P	Modern St	None	New Car	2018
Ho					4cyl 1.5L P	Rally Red	None	New Car	2018
Ho					4cyl 1.5L P	Rally Red	None	New Car	2018
Ho					4cyl 1.5L P	Cosmic Bl	None	New Car	2018
Ho					4cyl 1.5L P	Lunar Silve	None	New Car	2018
Ho					4cyl 1.5L P	White Orcl	None	New Car	2018
Ho					4cyl 1.5L P	Cosmic Bl	None	New Car	2018
Ho					4cyl 1.5L P	Modern St	None	New Car	2018
Ho					4cyl 2.0L T	Rally Red	None	New Car	2018
Ho					4cyl 2.0L T	Champion	None	New Car	2018
Ho					4cyl 2.0L T	Sonic Grey	None	New Car	2018
Ho					4cyl 2.0L T	Crystal Bla	None	New Car	2018
Ho					4cyl 2.0L T	Brilliant Sp	None	New Car	2018
Honc	30223	Drive Awa		0 Automatic	4cyl 1.8L P	White Orcl	None	New Car	2018

On further investigation, in every instance where this had occurred, the value in the odometer column (to the left of body, the first of the three offending columns) was zero. State had value 'none' and seller had value 'new car'. The engine column contained values that seemed to describe colours. Assuming these were errors, I moved the data points to the right to align them in their proper column.

I decided not to use OneHotEncoder on the cylinder and litres columns, as these columns contained numbers with letters representing the measure (Cyl and L). Instead I used the LEFT() command (in excel) to drop the letters off the values, thus leaving the columns only with numeric data. I added the measure name to the column labels, 'Cylinder (no. of)' and 'Litres (size)'.

Fig 5b: Still data quality issues in the litres column. For example, 1.2 litres was sometimes represented as '1200' litres. Also note the value '13b'.

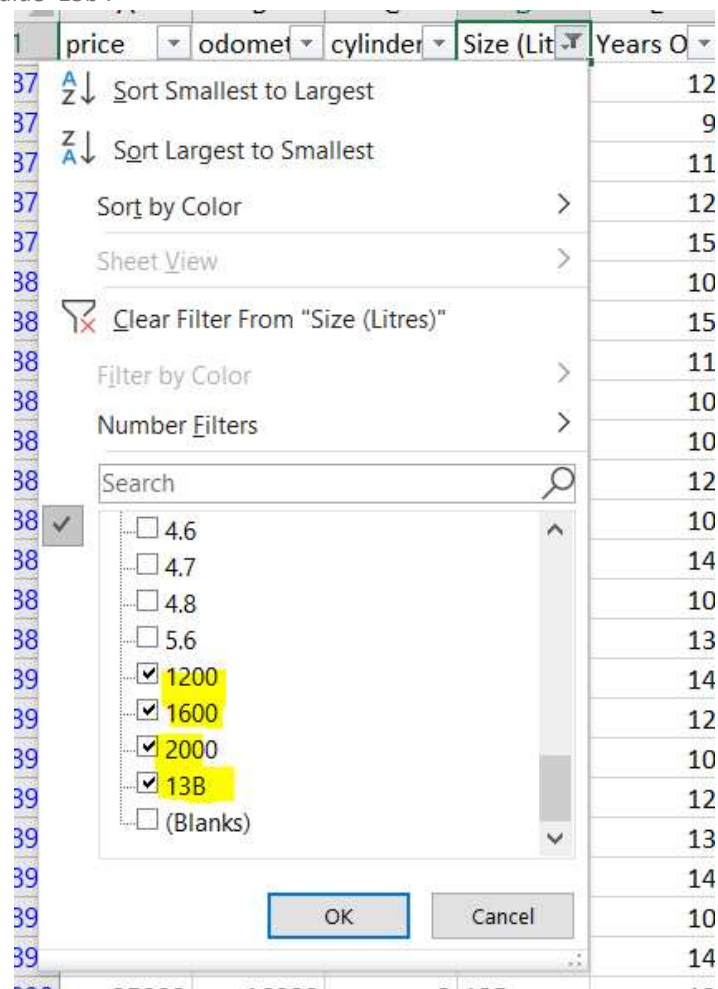
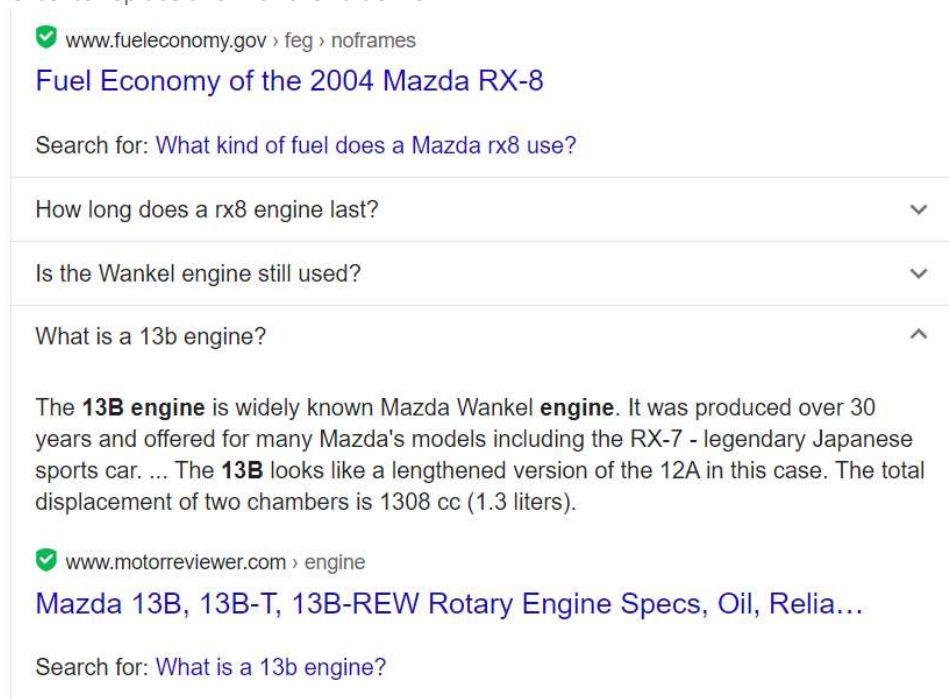


Fig 5c: 13b always corresponds to Mazda cars in our dataset. It is a 1.3 litre engine, so I used the find and replace function in excel to replace this with the value 1.3.



Python data transformation

Using pandas, I read in the dataset and converted it to a dataframe and drop the rows where there was no value in the 'price' column. I also dropped columns that did not provide significant entropy. As a prelude to OneHotEncoder, the code output the number of distinct values for every column. The dataframe contained 51 columns after all conversions and removals were complete. The code read the dataframe into a new csv file, which was our cleansed dataset, ready for modelling.

Code Walkthrough

```
1 import pandas as pd
2 df = pd.read_csv("carpricefixedbetter2.csv")
3 df.dropna(subset=["price"]) #drop rows with null values in price column
4 df=df.drop(['id','title','model','Colour'],1) #
5 print(df)
6 col=list(df.columns)
```

I imported the PANDAS module and loaded the csv into a dataframe

```
8 for i in range(1,len(df.columns)):
9     j=df[col[i]].unique()
10    print(col[i],"contains ",len(j)," distinct values\n")
```

This for loop prints the column name and its number of distinct values.

```
13 #Convert calendar year to years old.
14 df["Years Old"]=2018-df["year"]
15 #
16 from sklearn.preprocessing import OneHotEncoder
```

```
17 enc = OneHotEncoder(handle_unknown='ignore')
```

I subtracted the value 2018 from each cell in the 'year' column, so that instead of showing the year of registration, it gives the number of years since the car was registered.
Highlighted in yellow is the import of the OneHotEncoder module.

```
20 enc_df = pd.DataFrame(enc.fit_transform(df[['brand']]).toarray())
21 df = df.join(enc_df)
22 print(df.columns)
23
24 categories=df['brand'].unique()
25 for i in range(0,len(categories)):
26 df.rename(columns={i:categories[i]}, inplace=True)
```

This is the process of OneHotEncoder choosing each distinct categorical datum per column and creating a binary column. In this column, rows where the category was present have a value of 1 and for rows that do not contain the category, a value of '0'. The binary column is then joined onto the end of the dataframe, 'df'. Now every distinct category has a corresponding binary column. These columns are just named '0','1','2',...,'n' where n= no. of distinct categorical values in column. We rename these columns as the category these represent.

Fig 4d: Before and after the binary columns corresponding to 'Brand' were renamed.

```
year contains 47 distinct values

Index(['brand', 'price', 'discount', 'odometres', 'body', 'transmission',
      'cylinders (No. of)', 'Size (Litres)', 'Fuel', 'state', 'seller', 'year',
      'Years Old', 0, 1, 2, 3],
      dtype='object')
Index(['brand', 'price', 'discount', 'odometres', 'body', 'transmission',
      'cylinders (No. of)', 'Size (Litres)', 'Fuel', 'state', 'seller',
      'year', 'Years Old', 'honda', 'mazda', 'nissan', 'toyota',
      'Excl. Govt. Charges', 'None', 'Drive Away', 'Get Price*'],
      dtype='object')
```

Decision tree is a supervised learning algorithm. It is a set of rules for partitioning the data set based on the values of different predictors. Decision tree aims to split on the variable that produces the purest child nodes (the ones with the maximum number of same target labels). The variable is selected based on calculation of information gain for all variables.

Gradient boosting machines combine decision trees, but unlike the random forest algorithm, it starts the combining process at the beginning, instead of at the end. Gradient boosting builds one tree at a time. *“If you carefully tune parameters, gradient boosting can result in better performance than random forests. However, gradient boosting may not be a good choice if you have a lot of noise, as it can result in overfitting.”*

Fig 4e: Snapshot of a tiny portion of the decision tree.

[illegible]

Auto Model

The initial dataset and cleansed dataset were tested on RapidMiner. Part of the auto model process recommends the optimal columns to model on, based on the id-ness and correlation values. It recommended 23 of the attributes to model on. I added the 'Honda' column for completeness so that all car brands in the dataset were included in the model.

Fig 5a: The 24 variables modelled on.

Automatic	Cab Chassis	Coupe	Cylinders (No. Of)	Excl. Govt. Charges	Get Price*
Honda	In Stock	Light Truck	Manual	Mazda	New Car
Nissan	None	Not_Known	Odometres	Qld	Size (Litres)
State_None	Toyota	Turbo Petrol	Ute	Vic	Years Old

RapidMiner creates a 40% hold-out set from your input data to evaluate the model. Predictions will be created for those to calculate how well the models work. The remaining 60% of the data was used to train the model.

Abbreviations

Ute = Utility Vehicle, or pickup truck.

Vic = State of Victoria, in Australia

Chapter 5: Testing and Results

Auto Model Results

Decision Tree and Gradient Boosted Trees gave best results, for both the original and prepared datasets. There was a significant improvement on the prediction accuracy when the cleaned data was used. (See fig 7b and 7c, below).

Fig 5a: Before- Gradient Boosted Trees gave a relative error score of 19.3% and Decision Tree gave 20%.

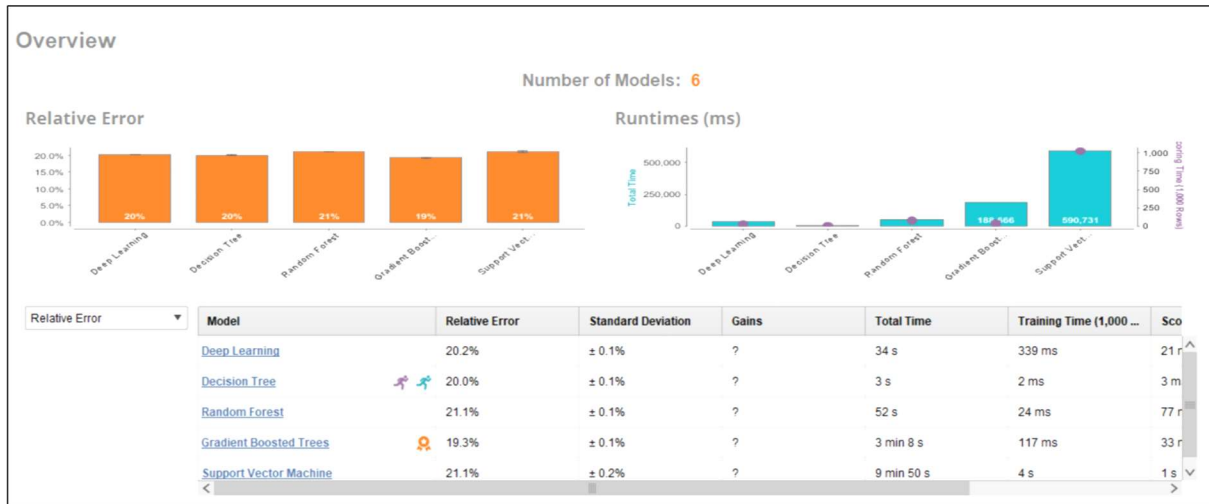


Fig 5b: After - Gradient Boosted Trees gave a relative error score of 11.7% and Decision Tree gave 13%.

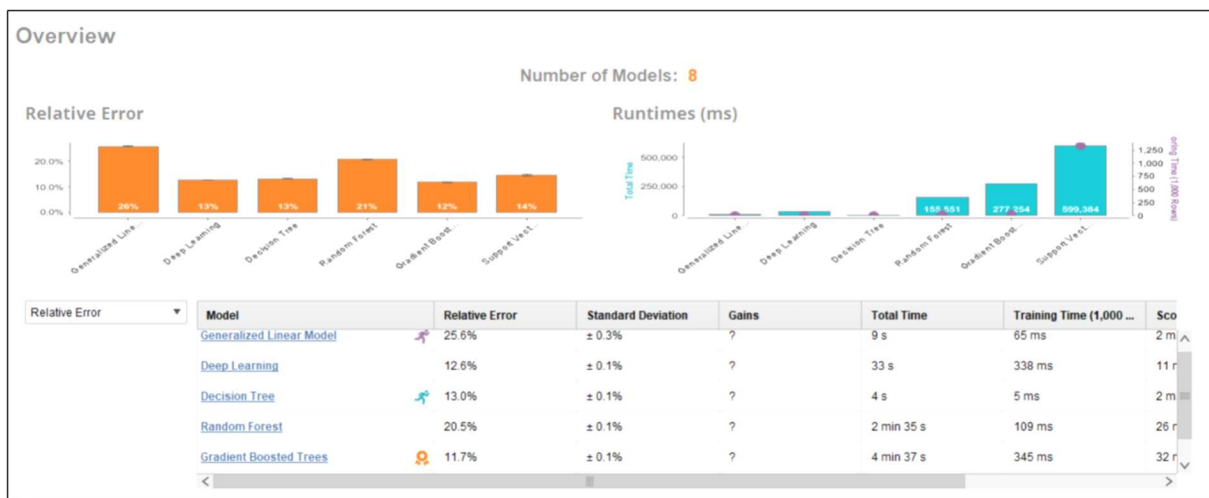
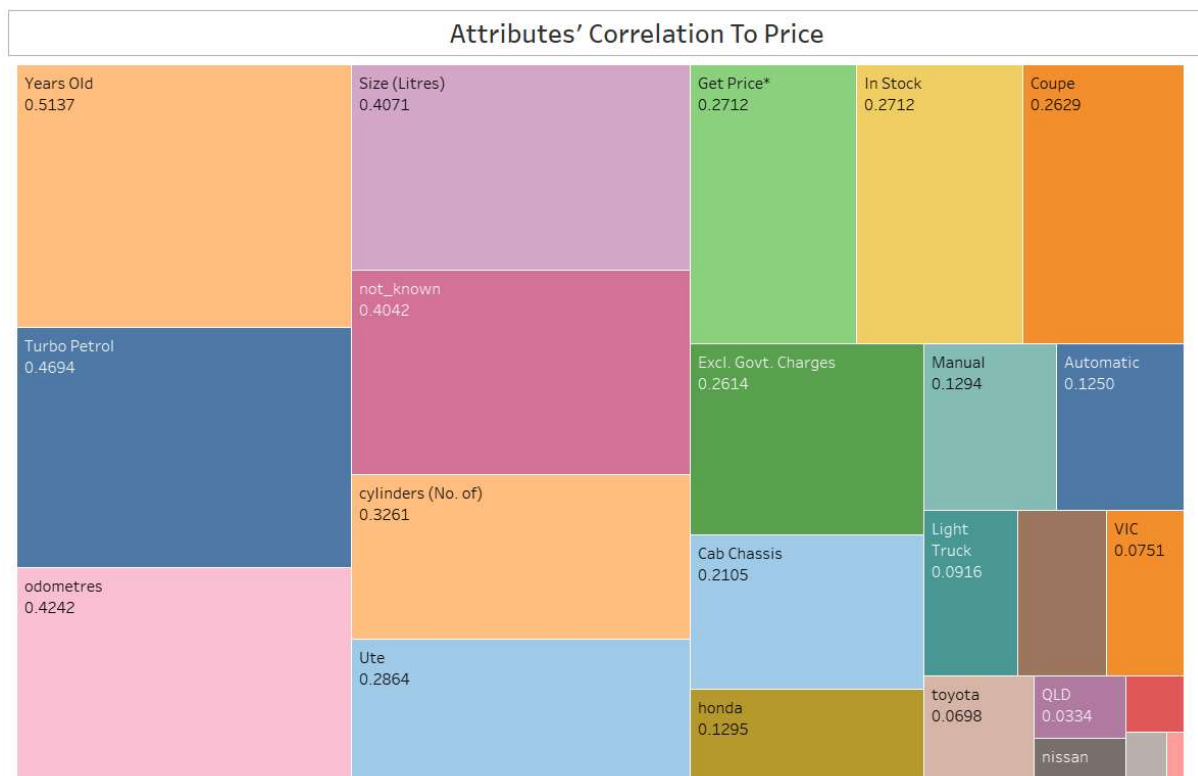


Fig 5d: Dataset with price predictions

Decision Tree - Predictions							
Row No.	price	prediction(p...	odometres	cylinders (N...	Size (Litres)	Years Old	honda
1	13990	16299.909	54830	4	2.400	8	1
2	5900	6841.625	180000	4	2.400	11	1
3	18998	17043.181	102612	4	2.400	5	1
4	12772	13813.387	102966	4	2.400	6	1
5	43555	19556.500	26130	4	2	3	1
6	36990	21096.667	36887	6	3.500	2	1
7	29588	24216.750	16183	4	2.400	2	1
8	6999	9017	121200	6	3	12	1
9	11990	10945	122310	6	3.500	10	1
10	39888	45265	12003	6	3.500	1	1
11	39490	34184.333	2010	4	2.400	2	1
12	24090	24448	10431	4	2.400	1	1

Attribute Correlation to Price



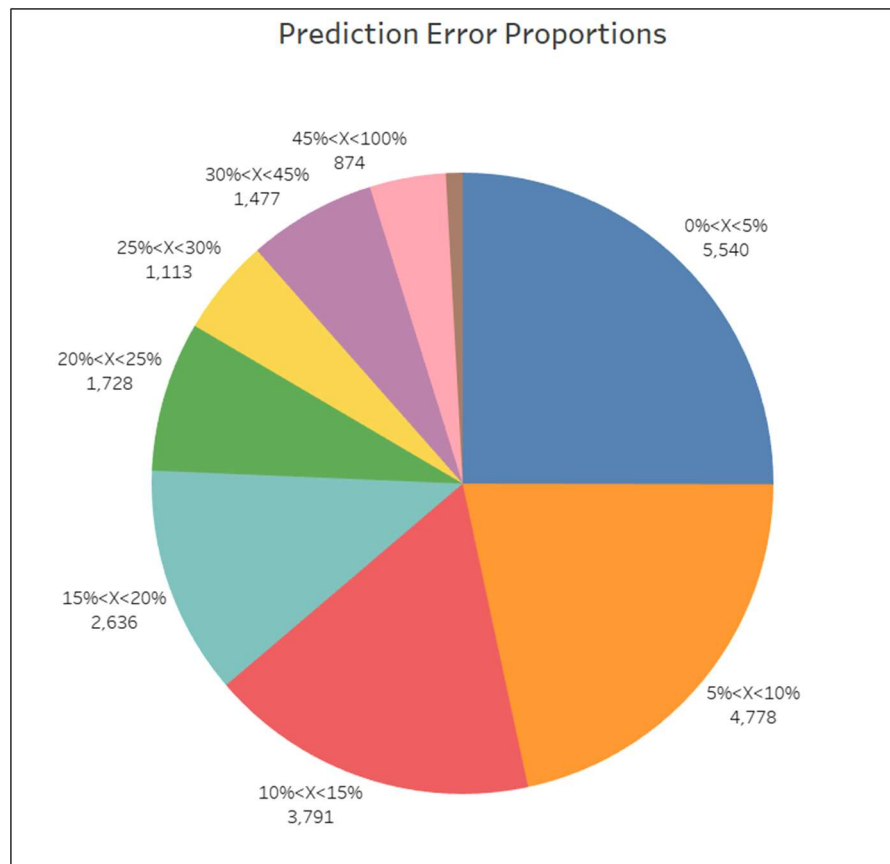
Analysis of price prediction error

I exported the price predictions onto a csv. I created a calculated column for the percentage error in the price predictions, and then ordered the data by lowest % error. Using the COUNTIFS function, I counted the number of rows whose % error fell within a certain interval.

Fig 5c

AG2											
=COUNTIF(AC:AC,"<=0.05")											
	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI
1	price	prediction(price)	abs error	percent error of actual price					count in each 5% block	X= percentage error	% of total population
2	44000.0	44000.0	0	0.00%					5540	0%<X<5%	49.775%
3	15990.0	15990.0	0	0.00%					4778	5%<X<10%	42.929%
4	44000.0	44000.0	0	0.00%					3791	10%<X<15%	34.061%
5	18888.0	18888.0	0	0.00%					2636	15%<X<20%	23.684%
6	56888.0	56888.0	0	0.00%					1728	20%<X<25%	15.526%
7	6990.0	6990.0	0	0.00%					1113	25%<X<30%	10.000%
8	7990.0	7990.0	0	0.00%					734	30%<X<35%	6.595%
9	28990.0	28990.0	0	0.00%					435	35%<X<40%	3.908%
10	28990.0	28990.0	0	0.00%					308	40%<X<45%	2.767%
11	28990.0	28990.0	0	0.00%					224	45%<X<50%	2.013%
12	28990.0	28990.0	0	0.00%					147	50%<X<55%	1.321%
13	28990.0	28990.0	0	0.00%					149	55%<X<60%	1.339%
14	28990.0	28990.0	0	0.00%					70	60%<X<65%	0.629%
15	28990.0	28990.0	0	0.00%					80	65%<X<70%	0.719%
16	28990.0	28990.0	0	0.00%					42	70%<X<75%	0.377%
17	28990.0	28990.0	0	0.00%					44	75%<X<80%	0.395%
18	28990.0	28990.0	0	0.00%					51	80%<X<85%	0.458%
19	28990.0	28990.0	0	0.00%					19	85%<X<90%	0.171%
20	28990.0	28990.0	0	0.00%					20	90%<X<95%	0.180%
21	28990.0	28990.0	0	0.00%					28	95%<X<100%	0.252%
22	28990.0	28990.0	0	0.00%					193	100%<X<1000%	1.734%

Fig 5d – Pie Chart showing proportions of records falling within various percentage error intervals

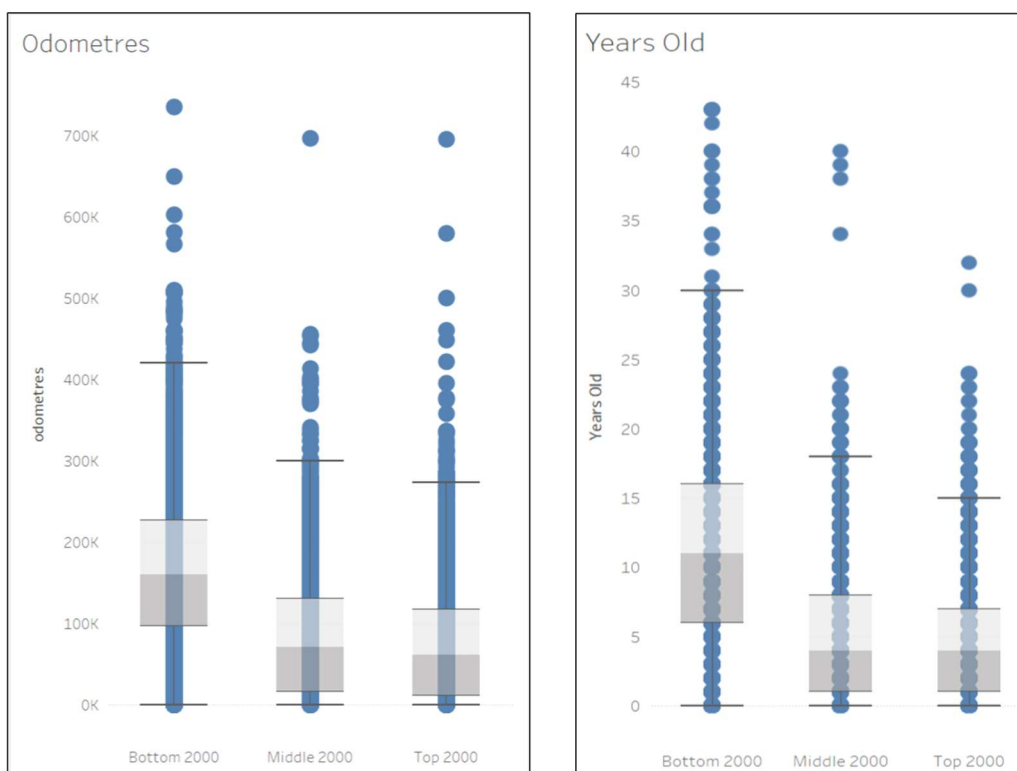


The results were impressive. Almost half the test data was within 5% of the actual price amount and 92.5% of the test data was within 10% of the actual price.

The overall average error of predictions was 16.11% and the median was 10.85%.

I examined the output dataset and extracted the 2000 most accurate predictions, the 2000 least accurate predictions, and the middle 2000 predictions. I calculated the average value for each attribute in these three categories and calculated the differences between the top, middle and bottom averages. For the 'Odometres' and 'Years Old' dimensions, there were significant differences in the average value of the best predictions and worst predictions.

Fig 5f: Box plots showing the distribution of values for 'Odometres' and 'Years Old' for top, middle and bottom 2000 most accurate price predictions.



Chapter 6: Conclusions and Future Work

In my findings, I found that the age of car and mileage are two significant factors in determining a car's sale value. The decision tree and gradient boosted vectors methods gave the most accurate price predictions.

The most accurate price predictions are for cars that, on average, are much newer. Indeed, the overall accuracy of predictions was skewed by a small sample of outliers whose price predictions were up to 800% different to the actual price.

Had I more time to spend in the project, I would have attempted to predict the depreciation in a car's value, using time series, using the price average difference between cars of each age in the dataset.

This project has shown that there is consistency to car pricing and that a reasonable estimate of a car's sale value can be achieved, by applying the appropriate weights to the car's attributes.

References and Bibliography

1. Dataset in csv file sourced from:
Kaggle.com – *Second-hand car price prediction*

Uploaded by: [XIAO JIN](#)

<https://www.kaggle.com/bahamutedean/secondhand-car-price-estimation>
2. Statistic for used car sales in Ireland:
Used car sales reach highest ever level in 2019 - CSO

RTE News

<https://www.rte.ie/news/business/2020/0113/1106090-cso-car-sales/#:~:text=But%20the%20number%20of%20used,since%202007%2C%20the%20CSO%20added.>
3. Similar project
Kaggle.com – *Trying to predict used car value*

Uploaded by Dan

<https://www.kaggle.com/ddmngml/trying-to-predict-used-car-value/notebook#Intro>
4. Very helpful website for information on data analytics.
<https://towardsdatascience.com/>
5. Decision Tree vs Random Forest vs Gradient Boosting Machines: Explained Simply
<https://www.datasciencecentral.com/profiles/blogs/decision-tree-vs-random-forest-vs-boosted-trees-explained#:~:text=Like%20random%20forests%2C%20gradient%20boosting,one%20tree%20at%20a%20time.>
Posted by [Stephanie Glen](#) on July 28, 2019.

Appendices

Data Preparation Code

```
1 import pandas as pd
2 df = pd.read_csv("carpricefixedbetter2.csv")
3 df.dropna(subset=["price"]) #drop rows with null values in price column
4 df=df.drop(['id','title','model','Colour'],1) #
5 print(df)
6 col=list(df.columns)
7
8 for i in range(1,len(df.columns)):
9     j=df[col[i]].unique()
10    print(col[i], "contains ",len(j), " distinct values\n")
11
12 #
13 #Convert calendar year to years old.
14 df["Years Old"]=2018-df["year"]
15 #
16 from sklearn.preprocessing import OneHotEncoder
17 enc = OneHotEncoder(handle_unknown='ignore')
18
19
20 enc_df = pd.DataFrame(enc.fit_transform(df[['brand']]).toarray())
21 df = df.join(enc_df)
22 print(df.columns)
23
24 categories=df['brand'].unique()
25 for i in range(0,len(categories)):
26     df.rename(columns={i:categories[i]}, inplace=True)
27
28 enc_df = pd.DataFrame(enc.fit_transform(df[['discount']]).toarray())
29 df = df.join(enc_df)
30 print(df.columns)
31 categories=df['discount'].unique()
32 for i in range(0,len(categories)):
33     df.rename(columns={i:categories[i]}, inplace=True)
34
35 df["body"].fillna("other_Body_Type",inplace=True) #input contains NaN.
36 #OneHotEncoder cannot work with NULL values.
37 #So I replaced them.
38
39 enc_df = pd.DataFrame(enc.fit_transform(df[['body']]).toarray())
40
41 df = df.join(enc_df)
42 print(df.columns)
43
44 categories=df['body'].unique()
45 for i in range(0,len(categories)):
46     df.rename(columns={i:categories[i]}, inplace=True)
47
48 enc_df = pd.DataFrame(enc.fit_transform(df[['seller']]).toarray())
49
50 df = df.join(enc_df)
51 print(df.columns)
52
53 categories=df['seller'].unique()
```

```

54 for i in range(0,len(categories)):
55     df.rename(columns={i:categories[i]}, inplace=True)
56
57
58
59 df["transmission"].fillna("Transmission_Other",inplace=True)
60 enc_df = pd.DataFrame(enc.fit_transform(df[["transmission"]]).toarray())
61
62 df = df.join(enc_df)
63 print(df.columns)
64
65
66 categories=df["transmission"].unique()
67 print(len(categories))
68 for i in range(0,len(categories)):
69     df.rename(columns={i:categories[i]}, inplace=True)
70
71
72 #Fuel
73 df["Fuel"].fillna("Fuel_type_not_known",inplace=True)#input contains NaN.
OneHotEncoder cannot work with NULL values. So I replaced them.
74 enc_df = pd.DataFrame(enc.fit_transform(df[["Fuel"]]).toarray())
75
76 df = df.join(enc_df)
77 print(df.columns)
78
79
80 categories=df["Fuel"].unique()
81 print(len(categories))
82 for i in range(0,len(categories)):
83     df.rename(columns={i:categories[i]}, inplace=True)
84
85
86 #States-OneHotEncoder
87 enc_df = pd.DataFrame(enc.fit_transform(df[["state"]]).toarray())
88
89 df = df.join(enc_df)
90 print(df.columns)
91
92
93 categories=df["state"].unique()
94 print(len(categories))
95 for i in range(0,len(categories)):
96     df.rename(columns={i:categories[i]}, inplace=True)
97
98
99 #Drop now redundant columns
100 print("DROPPING the following columns as they are now redundant: \nyear, brand,
discount, body, seller, transmission, state, Fuel.\n\n\n\n\n\n\n")
101 df=df.drop(['brand', 'discount', 'body', 'seller', 'transmission', 'state', 'year',
'Fuel'],1)
102
103
104
105 #Write dataframe df to csv
106 df.to_csv('car_price_dataCleansed.csv',index=False)
107

```



```
108
109
110 df2 = pd.read_csv("car_price_dataCleansed.csv")
111 print(df2)
112 print("\n\n\n\n\n\n\n\n\n\n")
113
114
```