Predicting Used Car Prices

Group 27

Written Report

Group Members	Kieran Flynn ➔ 16334663 ➔ kieran.flynn36@mail.dcu.ie Eoin O'Brien ➔ 15324971 ➔ eoin.obrien82@mail.dcu.ie
Programme	Computer Applications & Software Engineering Year 4 (CASE4)
Module	Data Warehousing & Data Mining (CA4010)
Date of Submission	18/12/2019

Declaration

Signed: Kieran Flynn, Eoin O'Brien   Date: 18/12/2019

1 Introduction

1.1 Idea

The aim of this project was to try and predict the price that users would sell a used car for on the German website Ebay-Kleinanzeigen. Using two different implementations of a Decision Tree based Python model, we will take a number of different entries with their own values for attributes detailing the car & use this to make an accurate assumption for how much a user of this website could potentially spend on certain vehicles. Using the latest version of Python alongside libraries such as Scikit-Learn, Matplot as well as Microsoft Excel, we performed the necessary calculations and created a number of noteworthy diagrams to showcase the distribution of the real and estimated data used in this project.

1.2 Dataset Description

The dataset used is a german based dataset that scraped details about the used cars that are sold on its source, Ebay-Kleinanzeigen. This dataset was found on the Kaggle. The dataset itself has roughly 370’000 entries of vehicles sold on the website and has 20 attributes for said entries. These can vary from details provided about the car by its seller, such as its price and mileage as well as its make & model, or details pulled about the offer itself such as when it was posted.

We chose this dataset for a number of reasons. Regardless of what idea we were going to pursue for this project, we needed to have a large number of data entries that we could use to train the data model. But unlike some other datasets we found in our research, the dataset was not constantly being updated. You could argue this makes the scope of the project less complex as its not live data, but we felt it was a safe option to use a dataset we would be able to look at anytime & know the range of its time frame for certain. This would then make it easier to segment the data for the project.

2 Data Preparation

2.1 Translation

The first problem we needed to tackle with our dataset was the fact that a lot of the attribute descriptions were in German. We initially explored using various python libraries (xlrd, googletrans[1]), which operated in conjunction with the Google Translate API, to perform and insert the translations. However, we also noticed that Microsoft Excel had a Translate function built in which we decided to use due to it being more straight forward with less chance of error. From there it was a case of translating a specific word, filtering by that German word and replacing all instances of it.

2.2 Redundant Columns

At this point we realised that there were a number of columns in our dataset that either provided no real use from an analysis point of view, or columns that were actually all of the same value.

2.2.1 Seller

The seller column details whether the listing is private or commercial. However, as there are only 3 instances of the commercial value we decided to remove this column as 3 out of 370,000+ entries is very insignificant.

2.2.2 Offer Type

The offerType column describes whether an entry is a listing or a petition. Similar to above, as there are only 12 instances of the petition value we decided to remove the column.

2.2.3 NrOfPictures

We observed that every entry in the nrOfPictures column had a value of 0. Due to this we removed it.

2.2.4 Postal Code

We decided that the postalCode column was not useful to us when predicting the car's price as we did not want to factor in location, we purely wanted to focus on the car itself.

2.2.5 Name

We decided to remove the name column as upon closer examination it was extremely hard to garner any useful insights from. This column is one that is filled in by the user when a car is listed and many entries had spelling errors, random non-alphanumeric characters and did not contain any information that was not already present in the make / model columns.

2.2.6 Date Crawled / Date Created / Last Seen

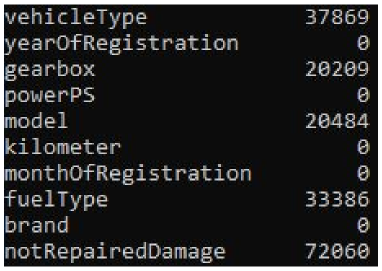
None of these columns provided any useful information to us when trying to predict the cars price as it was information about the listing itself. As such we removed these columns.

2.2.7 Abtest

The abtest column is a bit of a mystery as the two values are “control” and “test” and we’re not entirely sure what it actually represents. However, as there is a reasonably even split between the two values we are hesitant to remove this column as it could contain useful information. As such we decided not to remove the abtest column.

2.3 Null Values

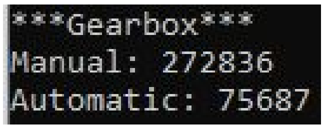
There were a number of options that we considered when it came to dealing with NaN / Null values in our dataset. The method used for each attribute differed from column to column. We first examined our data to see out how many missing values there were for each attribute, which can be seen to the right.



```
vehicleType              37869
yearOfRegistration           0
gearbox                  20209
powerPS                      0
model                    20484
kilometer                    0
monthOfRegistration          0
fuelType                 33386
brand                        0
notRepairedDamage        72060
```

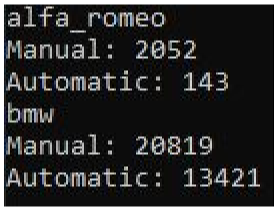
2.3.1 Gearbox

As seen above there were **20,209** null Gearbox values. We decided to examine the split between Manual / Automatic.



```
***Gearbox***
Manual: 272836
Automatic: 75687
```

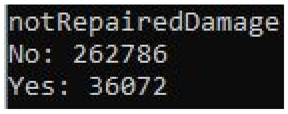
Because there were a significant number of “Automatic” values we decided to sort the values by brand and gearbox together to try and garner some extra insight. From there we were able to insert the highest frequency gearbox option for each brand of car.



```
alfa_romeo
Manual: 2052
Automatic: 143
bmw
Manual: 20819
Automatic: 13421
```

2.3.2 Not Repaired Damage

This column had about a 7:1 split between “No” and “Yes” and as such we filled in all NaN values with “No”.



```
notRepairedDamage
No: 262786
Yes: 36072
```

2.3.3 Fuel Type

Fuel type NaN’s were filled in with “petrol” as it was the most common entry.

2.3.4 Vehicle Type

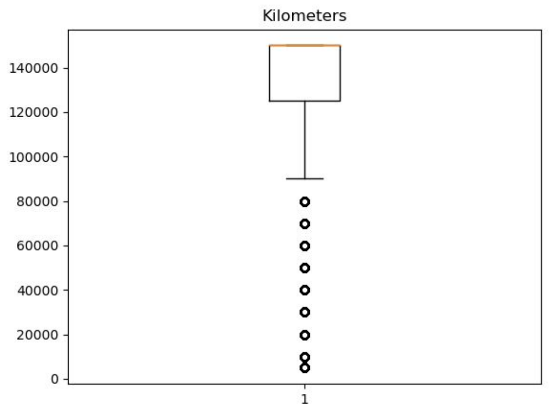
Vehicle Type was done in a similar fashion to Gearbox. It was grouped in combination with Fuel Type and filled according to the highest value in each category of fuel.

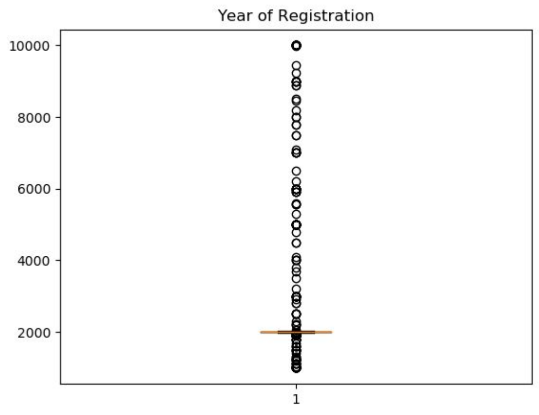
2.3.5 Model

When looking at the Model column we observed that there were a very high number of unique values, as well as there being no stand out value. Because of this we simply filled in NaN values with “Not Declared” so as not to influence the prediction algorithms.

2.4 Outlier Reduction / Data Limits

At this point in the assignment we have removed most of the problematic data. We noticed however there were a number of outliers in each of the attributes that would need to be removed before continuing on to the prediction phase. We created boxplots for each numeric attribute, two examples of which can be seen below for the “Year Of Registration” and “Kilometer” columns.





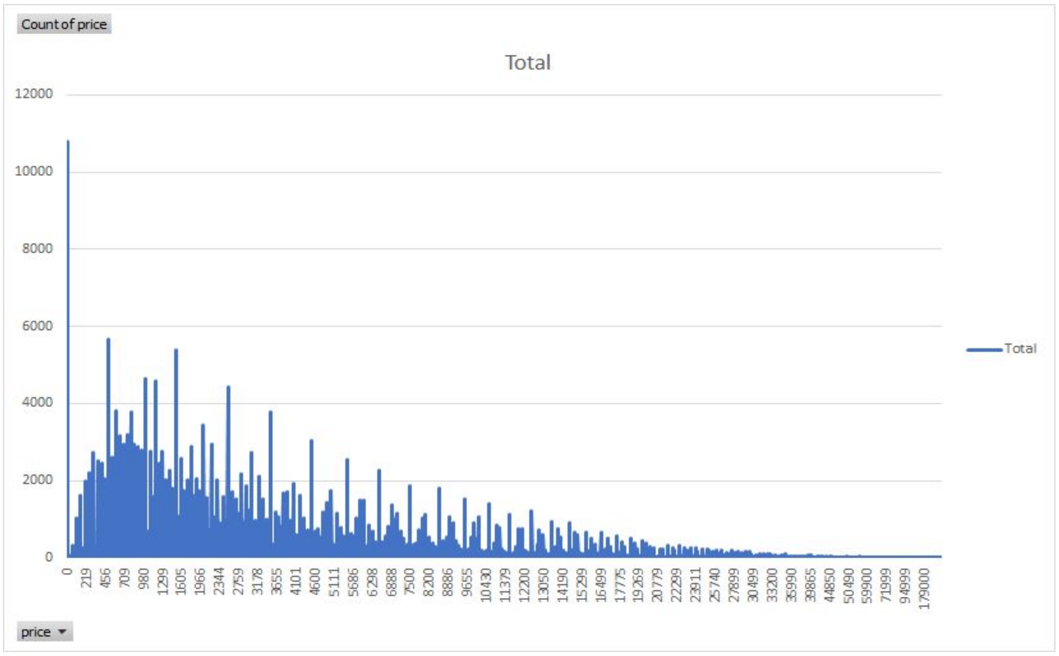
As you can see in the “Kilometers” boxplot there are quite a number of outliers on the lower end of the scale. However, we deemed these to be valid entries despite being outliers

2.4.2 Year Of Registration

We also noted that the Year of Registration attribute had a significant number of outliers. There were also a number of values past the current year (which is user error) so we removed this first. We also decided to limit the values to values remaining values to the years 1945 to 2016. We hoped that this would prevent this attribute from dominating the price too much.

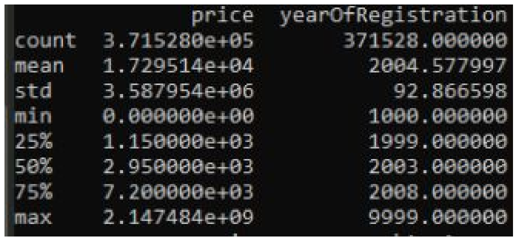
2.4.3 Price

It was obvious from our workshop that this was an attribute with quite an odd distribution, as can be seen below.

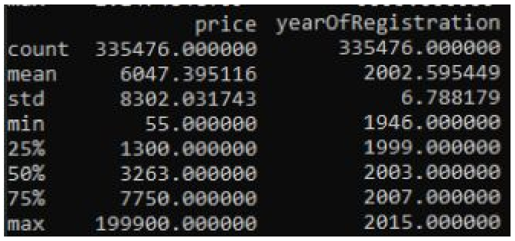


As you can see in the diagram there were over 10,000 instances of “0” as a price which is likely user error. On the other end of the scale were a number of massive values for this attribute that would also need to be removed. As such we settled on a €50 to €200,000 price range to strike a balance between removing problematic data and not removing useful information.

In the below images you can see a sharp drop in all of the price's main metrics, as well as the invalid years being removed from the yearOfRegistration column.



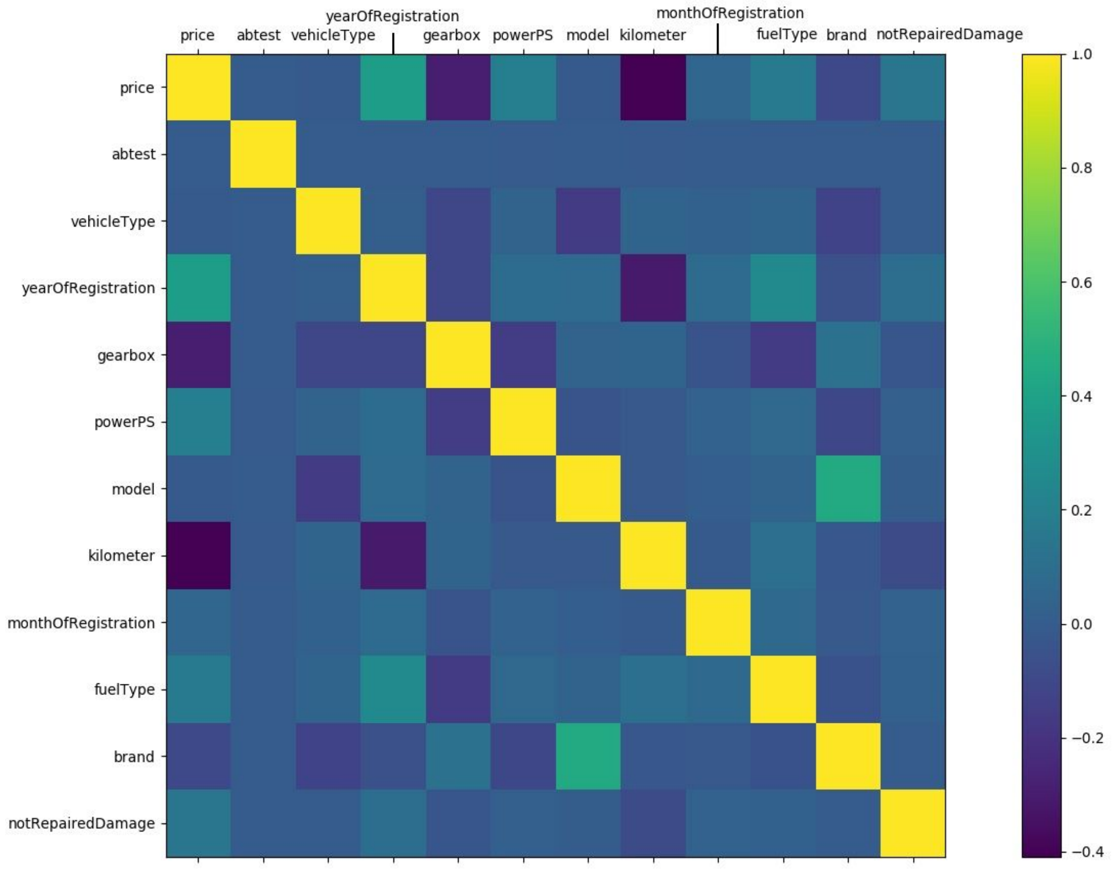
```
              price  yearOfRegistration
count  3.715280e+05       371528.000000
mean   1.729514e+04         2004.577997
std    3.587954e+06           92.866598
min    0.000000e+00         1000.000000
25%    1.150000e+03         1999.000000
50%    2.950000e+03         2003.000000
75%    7.200000e+03         2008.000000
max    2.147484e+09         9999.000000
```



```
               price  yearOfRegistration
count  335476.000000       335476.000000
mean     6047.395116         2002.595449
std      8302.031743            6.788179
min        55.000000         1946.000000
25%      1300.000000         1999.000000
50%      3263.000000         2003.000000
75%      7750.000000         2007.000000
max    199900.000000         2015.000000
```

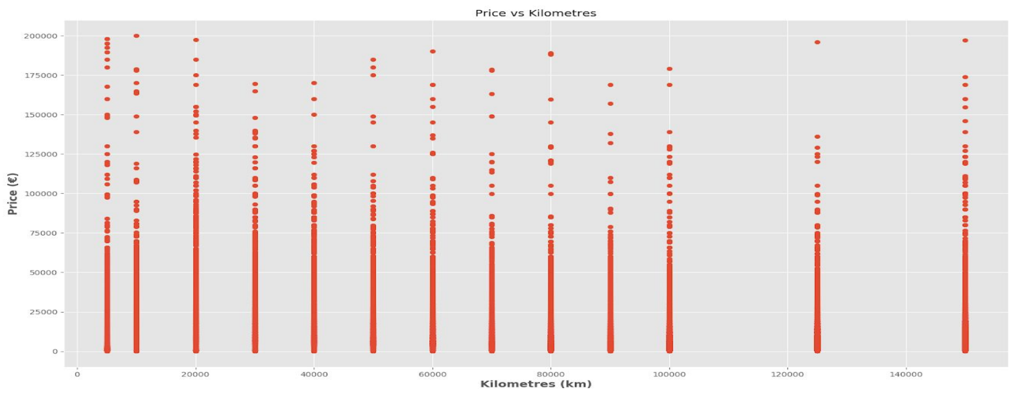
Before removing outliers *After removing outliers*

2.5 Data Stats & Graphs

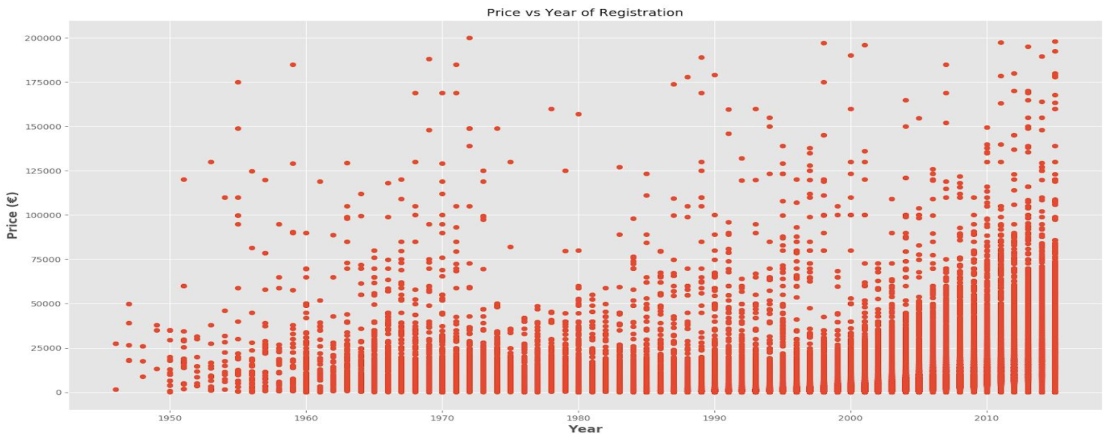


The above correlation diagram showcases the relationship that can be found between each of the attributes utilised in the data set. While it is hard to pinpoint a relationship between a lot of attributes, it is quite significant to see the strong relationship between the price and certain attributes such as year of registration and fuel type. That is on top of obvious pre existing relationships such as the vehicles brand and model.

2.6 Correlation with Price



From this diagram, we did not notice anything new or significant in terms of the skew of how the attributes are correlated but we were able to pinpoint some key ideas. Aside from a few outliers, vehicles that have garnered a high amount of mileage do not command a high price in the dataset. Majority of high mileage cars tend to go for a lower price.

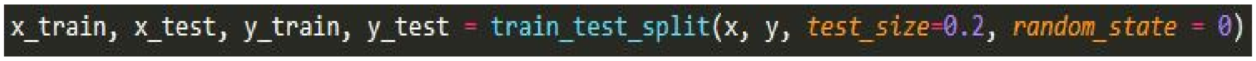


From this diagram, we can identify a negatively skewed distribution between the year of registration and the price given to a certain vehicle. Outside of a few outliers, prices of vehicles tend to rise the younger they are. Especially when they were registered in the past two decades. Meanwhile, less money is tended to be charged for vehicles. The scarcity of vehicles registered between 1945 and 1960 is not lost on us as it was a period of rebuilding for Germany following their loss in World War II.

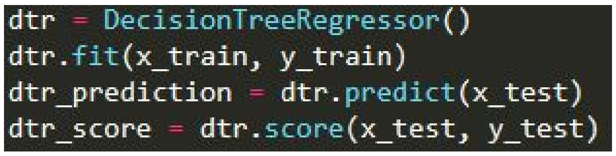
3 Algorithm Description

In the development of our algorithms for this project, we decided for the basis of it would be based around the use of a method that we had come across in this module in discussing classifiers. That being Decision Trees. This decision came after a period of brainstorming of methods we came across in this module and away from it. We initially considered using a Poisson Distribution model that we have previous experience with from our work on other projects. As well as this, we dabbled in using the built in Nayes Bayes methods included with the Scikit-learn module that we integrated with Python for the purpose of calculating important figures. While there are other languages for performing machine learning such as R, Python was the one we felt most comfortable using as its the language we have had the most experience with in DCU. The reason we ultimately did not proceed with Nayes Bayes or Poisson beyond just experimenting is that it they did not suit what we were trying to compute. Poisson and BernoulliNB are more useful for more binary based data, such as predicting which team will win a football match. While MultinominalNB is more focused on discrete data and how frequently something occurs and Gaussian is more concerned with normalised, continuous data. The shortcomings of these methods for our data set became apparent when we made brief attempts at trying to use them to predict prices as the accuracy was substantially lower than the targeted 60% and above. Knowing our data & the market it is based on, the prices of used cars was always going to be positively skewed towards cheaper prices.

However, by using a Decision Tree based algorithm, we were able to find a method where we could implement the standard 80:20 split between training and testing that would also be suited to the data we had at our disposal. The Decision Tree implementation conducts training by using a tree-like structure to produce data predictions. It utilises recursive partitioning by using different labels to iteratively navigate through different nodes from the root node to its children on the left and right side.

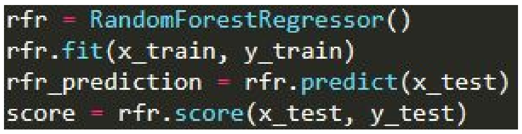


```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state = 0)
```



```
dtr = DecisionTreeRegressor()
dtr.fit(x_train, y_train)
dtr_prediction = dtr.predict(x_test)
dtr_score = dtr.score(x_test, y_test)
```

Another, more advanced method of Decision Tree generation we implemented was a Random Forest based model. Random Forest is a decision tree implementation based around randomised samples and subsets. It utilises bootstrapping to train a collection of trees on different samples and creates by finding an average from the prediction of the entire collection. As a result, our algorithm will not be overly concerned with the positive skew in our dataset.



```
rfr = RandomForestRegressor()
rfr.fit(x_train, y_train)
rfr_prediction = rfr.predict(x_test)
score = rfr.score(x_test, y_test)
```

4 Results

4.1 Random Forest Regression

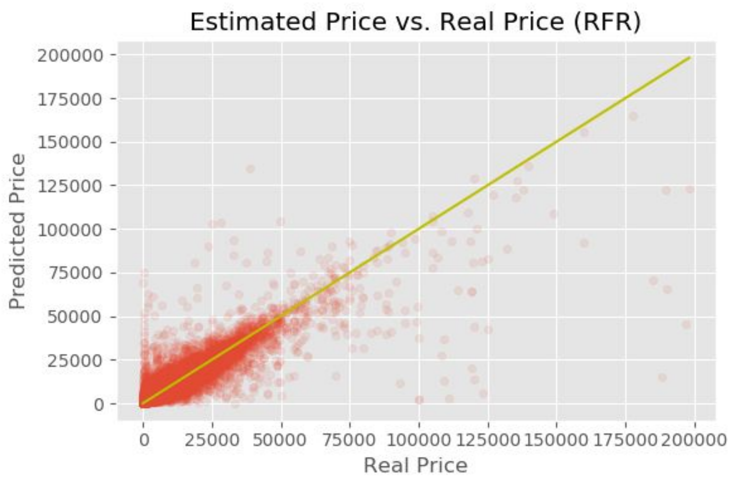
Overall, we were quite pleased with the results we got for our predictions with the Random Forest algorithm. Each initial test we did was greater than the 60% targeted accuracy and after adjustments made to the attribute limits, we were able to have a model that could average **a prediction accuracy of 83.5%**.

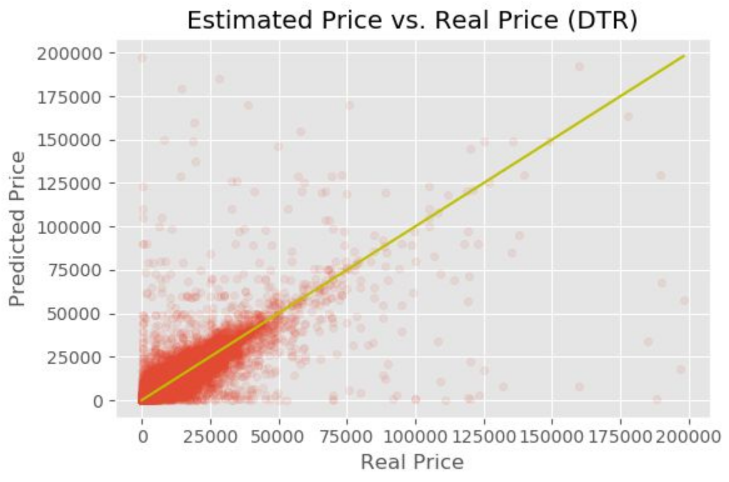
From experimenting with the attribute limits, we were able to get a clearer picture of how certain attributes were able to affect the accuracy of our prediction. For example, our age range for the car goes from the post World War II period of 1945 until 2016. When the age barrier is extended beyond that, you can actually increase the accuracy beyond that. Conversely, reducing the age range can lower the accuracy below 80%. The price limits also have a significant effect on the algorithms accuracy. Reducing the price as low as €50’000 can actually increase the accuracy of our model and bring it closer to 90% accuracy. That being said, that would have the model increasingly biased towards cheaper vehicles and would ignore the obvious existence of premium sports cars & range vehicles.

4.2 Decision Tree Regression

Meanwhile, our Decision Tree based algorithm also had a fair degree of success with trying to predict the price of used cars in our data set. While the accuracy of the model varied in value more than the Random Forest model, **it still managed to average just below 70% accuracy**. This is still greater than the targeted 60% that has been cited as greater odds than tossing a coin. The reason the Random Forest model is capable of a more accurate prediction than that of the single decision tree model is that the RFR model has a number of trees to generate its prediction from. The Decision Tree model is a singular node based representation of the entire dataset, while the Random Forest model is a collection of randomised decision trees that generates an average among the forest to make its predictions.

The diagrams on this page reflect how the two models predict the price. Either side of a line of best fit, the diagrams showcase a plot of data entries inside the price range as specified previously. You can see that from these two images that at the cheaper prices there is a stronger correlation between the real life prices for the vehicles and those estimated by our models. As said previously, this is particularly prominent in the range below €50’000. Hence why there was a stronger accuracy measure for when we tested the data set with those limits. That being said, it is also easy to identify outliers that are not even remotely close to the line of best fit once you go beyond that €50’000 cluster. There are a consistent number of values in the real data collection that don’t match up with the estimated values. Though you can also pinpoint a number of values scattered away from the best fit line on the estimation side in the Decision Tree model. But thankfully there is not a significant cluster of values scattered away from the margin on either side of the field. Hence showcasing that both models have made a commendable attempt at predicting the car prices.





That being said, our models are not perfect solutions and the Mean Absolute Error values for both models showcases this. The MAE for the Decision Tree model and Random Forest model averaged around €1’600 and €1’300 respectively. Given that range in both models is from €50 to €200’000, it won’t be of major significance to vehicles that already cost thousands of euro, especially in the upper market. It is, however, still a monetary risk if consumers were to use this as a basis to make a purchase.

In conclusion, after an initial period of anxiety as to whether a prediction for this dataset would be even possible, we are very satisfied with how our models were able to perform. It gave us further knowledge of resources we have dabbled with in the past few years here in DCU by introducing us to advanced machine learning technologies that are widely used in the industry and, arguably, serve as one of the most interesting aspects of computer programming. By experimenting with the python implementations of the classifiers we have encountered in this module, we have garnered know how that cannot be matched by a pen and paper in an exam situation by applying the formula to real life information to solve practical, real life problems. Hopefully this will serve as a stepping stone into potentially working with data mining techniques outside of DCU in our future careers.

5 References

1. https://pypi.org/project/googletrans/
2. https://becominghuman.ai/understand-regression-performance-metrics-bdb0e7fcc1b3
3. https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76
4. https://scikit-learn.org/stable/
5. http://kenzotakahashi.github.io/k-nearest-neighbor-from-scratch-in-python.html