Final Year Project Report

**Full Unit – Interim Report**

**Using Machine Learning Algorithms to Predict the Price of Pre-owned Cars**

Rizwan Bagdadi

A report submitted in part fulfilment of the degree of

**BSc (Hons) in Computer Science**

**Supervisor:** Li Zhang



Department of Computer Science

Royal Holloway, University of London

November 28, 2022

**Declaration**

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count:

Student Name: Rizwan Bagdadi

Date of Submission:

Signature: Rizwan Bagdadi

**Contents**

[**Abstract** 3](#_Toc121365707)

[**Project Specification** 4](#_Toc121365708)

[**1.** **Introduction** 5](#_Toc121365709)

[**2.** **Methodology** 5](#_Toc121365710)

[**2.1.** **Data Wrangling** 6](#_Toc121365711)

[**2.2.** **Normalisation** 7](#_Toc121365712)

[**2.3.** **Data Visualisation** 7](#_Toc121365713)

[**3.** **Implementation** 8](#_Toc121365714)

[**3.1.** **K-Nearest Neighbours** 9](#_Toc121365715)

[**3.2.** **Decision Tree** 12](#_Toc121365716)

[**4.** **Planning and time scale** 14](#_Toc121365717)

[**5.** **My diary** 14](#_Toc121365718)

[Bibliography 14](#_Toc121365719)

# **Abstract**

In this paper, I am investigating the application of supervised Machine Learning (ML) algorithms to predict the price of pre-owned cars in the UK. The goal of this project is to use ML algorithms to find the most suitable algorithm able to carry out the most accurate predictions, as well as providing the best performance and overall efficiency, by comparing their benchmark data. Two techniques, K-nearest neighbour and decision trees, have been used to make these predictions, using a dataset containing the features and prices of over 100,000 pre-owned cars, to train the algorithms.

Throughout this paper, I have first discussed why it is I have chosen to research this area, discussing the problems, the client and data. I have then extensively described and explained how each algorithm is effectively used, discussing the drawbacks and benefits of each.

# **Project Specification**

Enter here………

1. **Introduction**

Predicting the price of a pre-owned car is an important problem that not only benefits the buyer, but also the seller. Individuals or dealerships often state their buying price at very unreasonable prices - in comparison to the actual worth of the vehicle; taking advantage of those who are unsure of the approximate value of the car. When it comes to human-to-human interactions in quoting a fair and accurate price of a pre-owned car, there are a few problems that must be highlighted. This being a lack of knowledge, lack of awareness, lack of communication or lack of resources. Another problem being discrimination; this includes age, gender, race and religion. There two perspectives in selling a used car, and both could be affected in such situation. From a seller's perspective, there a possibility that they are unsure of the worth of the vehicle they are selling, leaving them vulnerable, this commonly occurs with elderlies trying to sell their cars, discriminating against their age. From the buyer's perspective, they could be quoted an unfair price due to the seller having malicious intent or prejudice against certain groups.

During the process of selling or buying a car, there are several distinct features that must be considered and examined to determine a price that is both accurate and reliable. These factors consist of brand, model, mileage, engine size, condition, registration year, transmission type, fuel type and fuel economy [h]. To consider this number of features that are needed to be taken into consideration will result in a time-consuming process and therefore will be inefficient for the estimation to be calculated manually.

A system that uses a dataset of hundreds of thousands of pre-owned cars selling prices, to predict the selling price of a used vehicle, would benefit both sellers and buyers. I have decided to use machine learning to create this system, that allows for fair, reliable and accurate results, eliminating all bias factors by removing the user from controlling the system. The system uses supervised algorithms, and will need to be trained by a reliable data set. ML is useful to use when it comes to making accurate predictions, as it has the ability to learn from massive amounts of data and can continue to learn. Advanced ML can also predict stock market trends - which would be extremely useful when it comes to stating the selling prices of cars no matter what the state of the market is.

The goal of this project is to use ML algorithms to find the most suitable algorithm able to carry out the most accurate predictions, as well as providing the best performance and overall efficiency, by comparing their benchmark data. With this project I will be applying at least three different supervised machine learning algorithms; consisting of logistic regression, k-nearest neighbour, and decision trees. To analyse these algorithms, I will perform a regression analysis and cross-validation of each algorithm used, in order to find which provides the best performance. To train the algorithms, I am going to use a dataset, last updated in 2020, that has the selling price and features of over 100,000 pre-owned cars [3].

1. **Methodology**

Data was downloaded from Kaggle [ ], uploaded by user Aditya, and was last updated in 2020. The data was collected in 2020, from live listings. When looking for an appropriate dataset, I made sure that all the data was well organised, containing information of price, mileage, road tax, miles per gallon (mpg) and engine size, and ensured there were no duplicate listings that would affect my results. A sample of the data used is shown below in Table 1.

Text

Description automatically generated

**Table 1.** *Sample Data Used of Audi Vehicles*

Before being able to use any of the data, I had to make it suitable for my models. I used Data preprocessing which includes the steps needed to make the raw data easily parsed by the algorithms and system [e]. Most real-world data tend to have missing values, noise, inconsistency, or unusable formats that cannot directly be used in machine learning algorithms. As seen in Table 1, there are a few problems that preprocessing can take care of. For example, there are string values for certain features and the integer variables vary from 1.0 to 35000+ for different features. This process increases the accuracy and efficiency of the models.

## **Data Wrangling**

For the first step of preprocessing, I performed the data wrangling process, which manipulates raw data into a usable and workable format, making it easier to process the data and give a reliable output. The first step I took was to look at the shape and type of variables included. The dataset is split into car brands, each containing thousands of samples. To reduce the sample size, I have chosen to only use one car brand, Audi, but hope to adapt my code to work with other brands in the future. Unfortunately, the data was still too large for my laptop to process, containing a shape of (10668, 9). To adapt for this I had then reduced the sample size to 1000, using the Pandas head() function.

Text

Description automatically generated with low confidence

**Table 2.** *Data Types Used*

Table 2 shows the data types found in the data set, using the function dtypes from the Pandas library. The variables contain mostly of integers, with some float and object types. As transmission, fuel type and the car model are all string values, that led to a problem where I could not use them in the algorithms. Considering the huge factor that these features have in the outcome of the selling price, I had used Scikit-Learns LabelEncoder() feature, which gives the strings a values between 0 and the number of values – 1.

## **Normalisation**

Before using the dataset, I had to consider the spread of data. In Table 1, you can see that the data varies from a wide range. The integer variables vary from 1.0, for the engine size, to 35000+, for the mileage, having a huge difference in their ranges. To fix this, I used the process of normalisation, which transforms the variables in columns to contain the same scale. Not every dataset needs to be normalised for machine learning, it is only needed when the characteristics have different ranges [f].

Some machine learning algorithms benefit from normalisation, especially K-Nearest Neighbours and SVM. This is due to using them using distance from data points to determine their similarities [g]. Figure 1 below shows the effects of scaling on a set of data points, when comparing the Euclidean distance. We can see scaling has brought both features closer to each other, making them more comparable than before.

A picture containing text

Description automatically generated

**Figure 1.** *Example of before and after of scaling [g]*

On the other hand, tree-based algorithms aren’t affected much by the scale of the features, as the decision process isn’t influenced by other features [g]. There are many different methods of normalisation, but for my project I chose to use Min-Max scaling. Min-Max scaling works by making the minimum of each feature equal to 0 and its maximum should equal to 1, transforming the range into 0-1 [f]. For this project, I have only normalised the data for K-nearest neighbour algorithm.

When normalising the data, I had the option of normalising the data before or after splitting the data, but I had chosen to normalise the data after splitting the data set, as doing it before splitting can cause details of the test set to be leaked, as the mean and standard deviation that is used in the normalisation process will be used on the whole dataset, including the test data [i].

## **Data Visualisation**

After having cleaned the data, we can now visualise the data much more clearly, seeing the relations and correlations between each aspect of the cars. Figure 1 shows the relations between each feature, using the Seaborn libraries Heatmap feature. Looking at the heatmap, we can see that price has a negative correlation with Mileage and mpg and has a positive correlation with how new the car is, relative to the selling year.

*Chart, bar chart

Description automatically generated*

**Figure 1.** *Relations Heatmap*

When buying a used car, people consider the mileage highly, as high mileage usually means the engine is less reliable due to the amount of use and stress the engine has been through. In Figure 2, we see a much more in-depth view which shows that the mileage has a significant impact on the selling price, and that the cars with higher mileage tend to be older. We can also see that most cars sold were sold with 5 years of it being owned – with the mode of the years from the data set being 2019.

*Chart, scatter chart

Description automatically generated*

**Figure 2.** *Scatter Plot of Relation Between Mileage, Price, and Year*

1. **Implementation**

The dataset I have used has been split into training (75%) and testing (25%) subsets. Using a large training set allows for my outcomes to be more reliable. For my system I have decided to create and analyse two algorithms, K-Nearest Neighbours and Decision Trees. Both models have different approaches and will most definitely output different results. The first algorithm I had tested was K-Nearest Neighbours.

* 1. **K-Nearest Neighbours**

K-nearest neighbour (KNN) is a non-parametric method of supervised learning which uses proximity to make predictions about the grouping of an individual data point [a]. It is usually used for classification but can also be used for regression cases.



**Figure 3.** *Visual of KNN [ ]*

KNN works by taking the closest k data points (k can be whatever you choose) to the new point. We do this by calculating the Euclidian distance to each training data point and choosing the k closest ones. There are other known methods for calculating the distance, of which the most common ones are Euclidian, Manhattan (for continuous data), and Hamming distance [b]. The Euclidian distance is calculated by square rooting the sum of the squared differences between the new point and training points, as shown in Figure 4. When the k closest points have been calculated, we then work out the average of their labels to determine the prediction, in this case the label is the price.

Chart, line chart

Description automatically generated

**Figure 4.** *Euclidian distance [c]*

To analyse the accuracy of the algorithm, I had worked out the RMSE value of each K value, from 0 to 100. In doing so, I was able to see that 4 nearest neighbours were the best value of K for my data set, giving a value of 3448.40 (2 dp). RMSE is the root mean square error, which shows how far the predictions falls from the actual values, by calculating the average deviation between the actual price and predicting price, using the Euclidean distance between them[j]. The lower the RMSE value, the better the model fits the dataset. In figure 5, we see that the best values of K for this data set are between 3 and 6, with the best being 4 neighbours. After k = 6, there is a huge linear increase in the value of RMSE, showing that the larger the value of K, the less accurate the results.

Chart, line chart

Description automatically generated

**Figure 5.** *RMSE value for values of K*

Below is an extract from my code, that is slightly modified to be more readable.

*# K-Nearest Neighbour method*

def KNN(train, testRow, yTrain, yTest, K):

distance = list() #Stores distance of each point

for i in range(len(train)-1):

dist = eucDistance(train[i], testRow)

distance.append((train[i], dist, yTrain [i]))

distance = sort(distance) *# Gets sent to sort method – sorts in ascending order*

kNeighbours = list() #list to store K amount of neighbour results

for i in range(K):

kNeighbours.append((testRow, distance[i][1], distance[i][2]))

return kNeighbours

*#Predictes the price of the given car*

def predict(train, test, yTrain, num\_neighbors):

predictions = list()

for i in range(len(test)):

neighbour = KNN(train, test[i], yTrain, num\_neighbors)

labels = [] #Stores yTrain for each test variable

for i in range (len(neighbour)):

t = labels.append(neighbour[i][2]) #Appends yTrain

predictions.append(mean(labels))

return predictions

*# Evaluating 100 nearest neighbours*

def eva(train, test, yTrain, yTest):

rmseValues = []

for k in range(1, 101):

y\_pred = predict(train, test, yTrain, k)

error = rmse(yTest, y\_pred)

rmseValues.append(error)

print('RMSE value for k =' , k , 'is:', error)

return

In the code above we can see how the evaluation process begin by creating a list of the RMSE values, inside the “eva()” method, then proceeds to loop through 100 values of K, in creating predictions. The prediction method takes all the training values and only the features of the test set, with this it creates a loop that iterates over each row (car) in the test set and finds the K nearest neighbours to that row using the KNN method.

The KNN method works by iterate rating over all the training set rows and finding their Euclidean distance to the test row. This is a costly process as it must work out the distance for 750 points to just one of the 250 rows of test data. Meaning the Euclidean distance must be worked out for 187,500 times for only one value of K. This is a huge factor when considering using KNN, as big datasets will be very time consuming. If I hadn’t reduced the Audi dataset to 1,000 vehicles, the Euclidean distance would have had to be calculated an astonishing 18,750,000 times for one value of K.

Once the KNN method has calculated the Euclidean distance to each training row, it then sorts the list of distances and takes the K nearest points to the test row and returns these values, along with their labels, to the prediction method. Finally, the prediction method takes the labels of the K nearest neighbours and calculates their mean, returning the prediction of that test row or user input.

For the evaluation process, the prediction method stores all given predictions in a list, of which it is returned to the evaluation after the completion of all predictions. With the given predictions, the evaluation method then calculates the RMSE value for each value of K and proceeds to plot these values, as shown in figure 5, enabling a visualisation of the accuracy of my algorithm.

Text, letter

Description automatically generated

**Figure 6.** *Demonstration of user input for KNN algorithm*

After testing my algorithm against a test set and finding the optimal number of neighbours for my dataset, I created a temporary user interface, shown in figure 6, that allows users to provide details of their own vehicles, currently limited to Audis only, and receive a reasonably accurate prediction. Figure 6 shows an interaction where the details for a 2016 Audi RS6, with 49,050 miles, and an actual worth of £44,985 was inputted into my algorithm using 4 neighbours, outputting a prediction of £44,717. This result being -£268 from the actual worth and with a calculation time of 0.094 (3 dp) seconds demonstrates promising results from my KNN algorithm. In order to calculate the prediction time, I used the time library to measure the time it took for the prediction method to run and return a prediction.

* 1. **Decision Tree**

A decision tree, like KNN, is a non-parametric form of supervised learning which uses labelled input and output datasets to train models. This technique is mainly used to solve classification problems but are also used in regression problems to predicts outputs from unseen data. Essentially, a decision tree is a tree where “each node shows a feature (attribute), each link (branch) shows a decision (rule) and each leaf shows an outcome (categorical or continues value)” [d]. Decision trees follow a human-like thinking, resulting in much more natural thought process, comparable to how humans think. Figure 7 illustrates the basic layout of a decision tree, showing sub-trees that fine-tune the predictions from the previous node, starting from the root node, and ending with leaf nodes.

Diagram

Description automatically generated

**Figure 7.** *Decision Tree Illustration [k]*

When building a decision tree using the training set, instead of looking ahead and choosing a split that will produce a more desirable tree at a later stage, the best split is made at each step, following a greedy approach. With classification, a method called entropy is used to measure information gain on a new examples class [L]. Entropy is the randomness within the information being processed [M]. However, regression relies on the assumption that we will produce an output given the input variables rather than attempt to predict a class label. I did this by examining whether a split would result in a decrease in the variance of the data. A split is considered useful if the collective weighted variance of its children has a smaller value that the parent’s original variance. I then continued to make splits on the dataset until the current node reached a predetermined minimum number of splits required to split an internal node and a maximum depth allowed. The method of splitting uses recursive binary splitting, meaning it follows a top-down, greedy approach to grows the tree with the training set.

Diagram

Description automatically generated with medium confidence

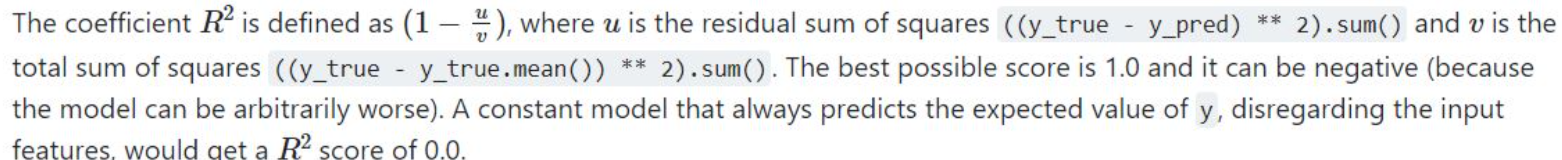
**Figure 8.** *Perspectives of recursive binary splitting [N]*

Figure 8 shows perspectives of the recursive binary splitting process, with the top right showing the splitting on a two-dimensional plane, which is then translated into the tree in the bottom left, and then into a 3D perspective of that [N]. This figure can help vaguely visualise the depth of recursive binary splitting that happens on many planes with the dataset of cars I am using.

Diagram

Description automatically generated

Info gain and entropy <https://towardsdatascience.com/entropy-and-information-gain-in-decision-trees-c7db67a3a293>



1. **Planning and time scale**
2. **My diary**

|  |  |
| --- | --- |
| Date | Tasks |
| 27/10/2022 | - Begun to research into the different types of Machine Learning Algorithms  - looked in K-Nearest Neighbour, how it works and how to implement it |
| 28/10/2022 | - Installed scikit-learn  - Added dataset to repository |
| 01/11/2022 | - Started coding to Jupyter Notebook - so that all data can be visualised/compiled in one place  - Researched the seaborn library to potentially visualise data on informative statistical graphic  - Also Researched Pandas library - to potentially analyse datasets  - split the data set using scikit-Learn  - Created a method to work out the Euclidean distance between two points  - Created a method to work out the KNN |
| 03/11/2022 | - Trying to convert .csv file into a .txt file - because reading the data set file by row has become difficult  - Created a sort method to sort out the distances in ascending order  - Used numpy's genfromtxt instead, but dataset is too large and takes too long to process |
| 04/11/2022 | - Cleaned the data: getting rid of model, transmission and fuel type.  - Broke down dataset into first 200 entries, because dataset is too large to process  - Created method for 1NN to test  - Numpy's genfromtxt doesn't like the engineSize and tax variables - so I have left them out for the time being |
| 05/11/2022 | - Created a prediction method to predict the price of the cars  - Results from prediction method are returning in an unwanted format  - Using Pandas library to read from library - for efficiency and better readings from the dataset  - Created a seaborn heatmap to visualise dataset  - Looked into Seaborn's distplots graph |
| 17/11/2022 | - Began writing the evaluation method for KNN - is currently suited for classification, not regression  - Began work writing interim report. |
| 18/11/2022 | - Began writing the methodology in the interim report |
| 21/11/2022 | - Begun writing description of KNN algorithm to Interim report |
| 23/11/2022 | - Begun coding the decision tree algorithm: imported & split data, method using scikit-learn, entropy method  - Begun writing description of decision algorithm to Interim report  - Attempted to visualise the decision tree |
| 25/11/2022 | - Normalised data for KNN  - Added preprocessing and normalization to Interim report  - Removed unnecessary .txt files |
| 26/11/2022 | - Fixed issued with reading normalised data  - Added code examples to interim report |
| 29/11/2022 | - Completed the evaluation method for KNN, using RMSE |
| 30/11/2022 | - Changed how I reduced data set to increase accuracy by 0.3 at peak  - Used Pandas feature "get\_dummies()" to generate dummy variables for model, transmission and fuel type  - Created a user input to get prediction of their own car |
| 01/12/2022 | - Removed "get\_dummies()" and replaced with "LabelEncoder()"  - Completed user prediction input - fully working |
| 02/12/2022 | - Fixed an issue with prediction, where predictions were completely wrong - fixed by normalising user input |
| 05/12/2022 | - Worked on decision tree algorithm - understanding what each method does  - Calculated how many times Euclidean distance is calculated and how long a single prediction takes  - Described my KNN algorithm code in the interim report |
| 06/12/2022 | - Created Decision tree class  - Initialised values  - Created fit method to put features and label together  - Created start of build tree method  - Created method to find the best split |
| 07/12/2022 | - Created a method to find split tree into branches - extension of best split method  - Using template entropy method to calculate entropy of a node - temporary to test later methods  - Created method to calculate information gain  - Completed best split and tree build methods  - Created method to predict the price of a car - currently not working fully |
| 08/12/2022 |  |

Bibliography

[a] <https://www.ibm.com/uk-en/topics/knn#:~:text=The%20k%2Dnearest%20neighbors%20algorithm%2C%20also%20known%20as%20KNN%20or,of%20an%20individual%20data%20point>.

[b] <https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/>

[c] <https://www.cuemath.com/euclidean-distance-formula/>

[d] <https://www.researchgate.net/profile/Purvi-Prajapati/publication/330138092_Study_and_Analysis_of_Decision_Tree_Based_Classification_Algorithms/links/5d2c4a91458515c11c3166b3/Study-and-Analysis-of-Decision-Tree-Based-Classification-Algorithms.pdf>

[e] <https://www.v7labs.com/blog/data-preprocessing-guide#h1>

[f] <https://deepchecks.com/glossary/normalization-in-machine-learning/>

[g] <https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/>

[h] <https://www.moneycrashers.com/factors-affect-used-cars-resale-value/>

[i] <https://towardsdatascience.com/preventing-data-leakage-in-your-machine-learning-model-9ae54b3cd1fb>

[j] <https://www.statology.org/rmse-vs-r-squared/>

[k] <https://anderfernandez.com/en/blog/code-decision-tree-python-from-scratch/>

[L] <https://www.jeremyjordan.me/decision-trees/>

[m] <https://www.askpython.com/python/examples/decision-trees>

[n] article <https://hastie.su.domains/ISLR2/ISLRv2_website.pdf>

[o]

[p]