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 Where other published and unpublished source materials have been used, these have been acknowledged.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](#_Toc120392169)

[**Project Specification** 4](#_Toc120392170)

[**1.** **Introduction** 5](#_Toc120392171)

[**2.** **Methodology** 6](#_Toc120392172)

[**2.1.** **Data Wrangling** 6](#_Toc120392173)

[**2.2.** **Normalization** 7](#_Toc120392174)

[**2.3.** **Data Visualisation** 7](#_Toc120392175)

[**3.** **Implementation** 8](#_Toc120392176)

[**3.1.** **K-Nearest Neighbour** 8](#_Toc120392177)

[**3.2.** **Decision Tree** 9](#_Toc120392178)

[Bibliography 11](#_Toc120392179)

# **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 two main problems that must be highlighted; the first being human error. Figure 1 shows a list designed by Gordon Dupont called the “dirty dozen” which highlights 12 of the most common human errors from which several are applicable when it comes to selling pre-owned cars, such as stress, lack of knowledge, lack of awareness, lack of communication and lack of resources. The second problem being discrimination; this includes race, gender, age, and religion. From the seller's perspective, they may not know the worth of the vehicle they are selling, leaving them vulnerable, this commonly occurs with elderlies trying to sell their cars, therefore the seller is discriminating against their age. From the buyer's perspective, they could be quoted an unfair price due to the seller having prejudice against certain groups.

Text

Description automatically generated

There are several distinct features that must be considered and examined to determine a prediction that is both accurate and reliable. Some of these factors consist of mileage, engine size, condition, registration year, transmission type, fuel type and fuel economy [2]. The 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.

Having 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. However, this system should not be user controlled, due to unfair and/or unreliable results. Hence the reason I believe Machine Learning (ML) should be implemented; to eliminate any bias and provide a fair, accurate selling price.  However, the system will use 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. **AUDI ONLY**

Text

Description automatically generated

**Table 1.** *Sample Data Used*

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. Yet still the data was too large for my laptop to process, containing a shape of (10668, 9). I had then reduced the sample size to 200, using the head() function. I am looking to solve this issue in the future, using an virtual machine.

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 I could not use the object types, as they are strings, I had removed them from my sample, using the drop() function. However, transmission, fuel type and the car model, are huge factors when considering the selling price and I would like to implement these in the prediction phase in the future. I will implement these key features by replacing them with integer e.g. petrol = 0, diesel = 1.

## **Normalization**

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, we use the process of normalization, 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 normalization, 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 normalization, 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 normalized the data for K-nearest neighbour algorithm.

Many machine learning algorithms require normalization. However, it is important to normalize AFTER splitting data. If you normalize before splitting, the mean and standard deviation used to normalize the data will be based on the full dataset and not the training subset — therefore leaking information about the test or validation sets into the train set. <https://towardsdatascience.com/preventing-data-leakage-in-your-machine-learning-model-9ae54b3cd1fb>

## **Data Visualisation #THIS IS CORRELATION**

After having cleaned the data, we can now visualise the data much more clearly, seeing the relations between each aspect. Figure 1 shows the relations, 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 pay serious attention to the mileage, as high mileage usually means the engine is less reliable. 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 used has been split into training (90%) and testing (10%) subsets. I have used two algorithms, K-Nearest Neighbours and Decision Trees.

* 1. **K-Nearest Neighbour**

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 be used for regression.



**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]*

An extract from my code (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

*# Euclidean Distance between two points*

def eucDistance(variable1, variable2):

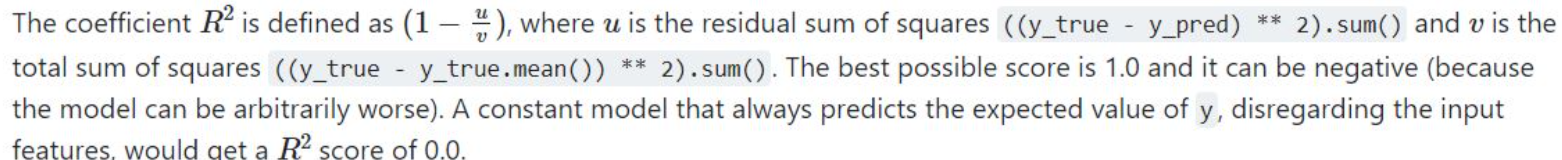
distance = 0

for i in range(len(variable2)):

distance += (variable1[i] - variable1[i])\*\*2

return np.sqrt(distance)

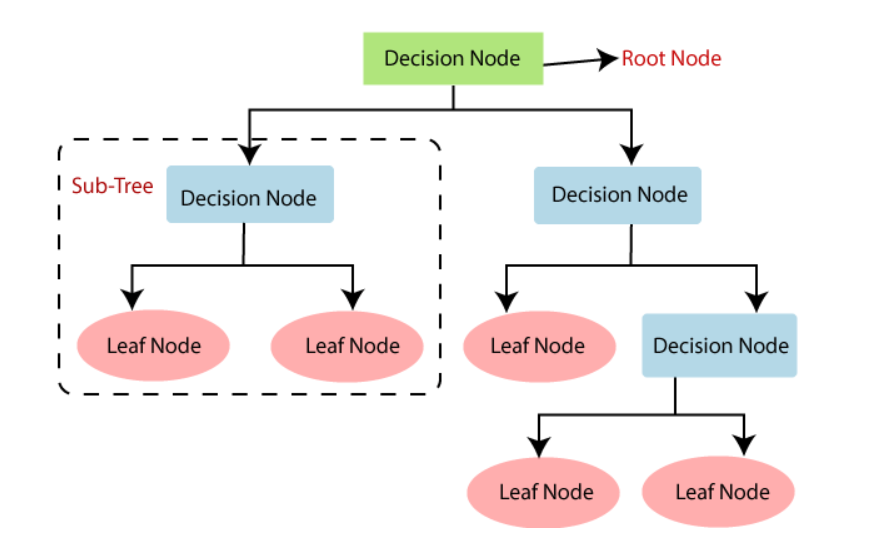
EVALUATION: Uses RMSE to calculate the average deviation between the predicted car price made by the model and the actual car price <https://www.statology.org/rmse-vs-r-squared/>



* 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. 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 an

decision trees represent classification knowledge more naturally to the way of human thinking and are more robust in tolerating imprecise, conflict, and missing information



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

Diagram

Description automatically generated

USEFUL INFO:

* new cost function — Root Mean Square Error (RMSE). It is the standard deviation of how far from the regression line data points are. In other words, it tells you how concentrated the data is around the line of best fit.

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]

[i]

[j]