University of Reading

Department of Computer Science

Predicting the Price of Cars using Machine Learning

Faris Hanna

Supervisor: Carmen Lam

A report submitted in partial fulfilment of the requirements of the University of Reading for the degree of
Bachelor of Science in Computer Science

September 29, 2021

Declaration

I, Faris Hanna, of the Department of Computer Science, University of Reading, confirm that this is my own work and figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person's work, except where the works of others have been explicitly acknowledged, quoted, and referenced. I understand that if failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

I give consent to a copy of my report being shared with future students as an exemplar.

I give consent for my work to be made available more widely to members of UoR and public with interest in teaching, learning and research.

Faris Hanna

September 29, 2021

Abstract

With factors such as depreciation, how can it be possible to formulate an appropriate valuation price for a car? There are factors, including the year and company, that contribute to a car's price. What tool can be developed that can help in predicting the price of a car based on multiple factors? This study will aim at audiences such as car insurance and rental companies that would benefit from the aid of the algorithm whether it is for determining prices or premium costs of cars. It can also aid individuals who are interested in determining their car's valuations to aid in selling or renting. This study will use data science and machine learning techniques to implement a model that can predict a car's price accurately based on a set of inputs. This model will be integrated with a UI form to allow non-technical users to interact with this tool. Techniques in this implementation involve data preprocessing, model implementation, and optimization. The best algorithm found for this problem is the Catboost algorithm, which has achieved impressive results of 97% accuracy. With high and impressive results obtained, a solution using data science and machine learning can be implemented for this problem. This shows that using machine learning is a great tool for many problems within the predictive analytics field.

Keywords: Data Science, Machine Learning, Car Price Predictor, Regression, Catboost

GitHub: https://github.com/farishanna/FYP-Project

Word Count: 12454

Contents

List of Figures

List of Tables

1. Introduction

1.1 Background

There are two main topics to discuss to have a greater background of this study. Cars are an exciting area, as it continues to develop and evolve through the years. It would be an interesting study to implement data science methods to make predictions of car prices, considering factors such as depreciation.

1.1.1 Background of Cars and the Production of Cars

A car is a motor vehicle with an engine that is powered by fuel of mainly either diesel or petrol. The use of cars is primarily for transportation and has been around since they were invented in 1886 (Car - Wikipedia, 2022) and have been developing and evolving ever since. People use cars to be able to travel long distances without having to walk, which is good for cases such as comfortability when traveling and for disabled and elderly people to travel. It also helps move supplies from place to place and is useful for time-sensitive situations, productivity, and emergencies. Complications in cars can be that they can be dangerous as many car accidents happen, so it would need suitable regulation. The production of cars has highly increased as their demand has been high for most of the general population. In Figure 1-1 (Happy motoring: Global automobile production 1900 to 2016, 2022), a graph is shown to present the increase in production of

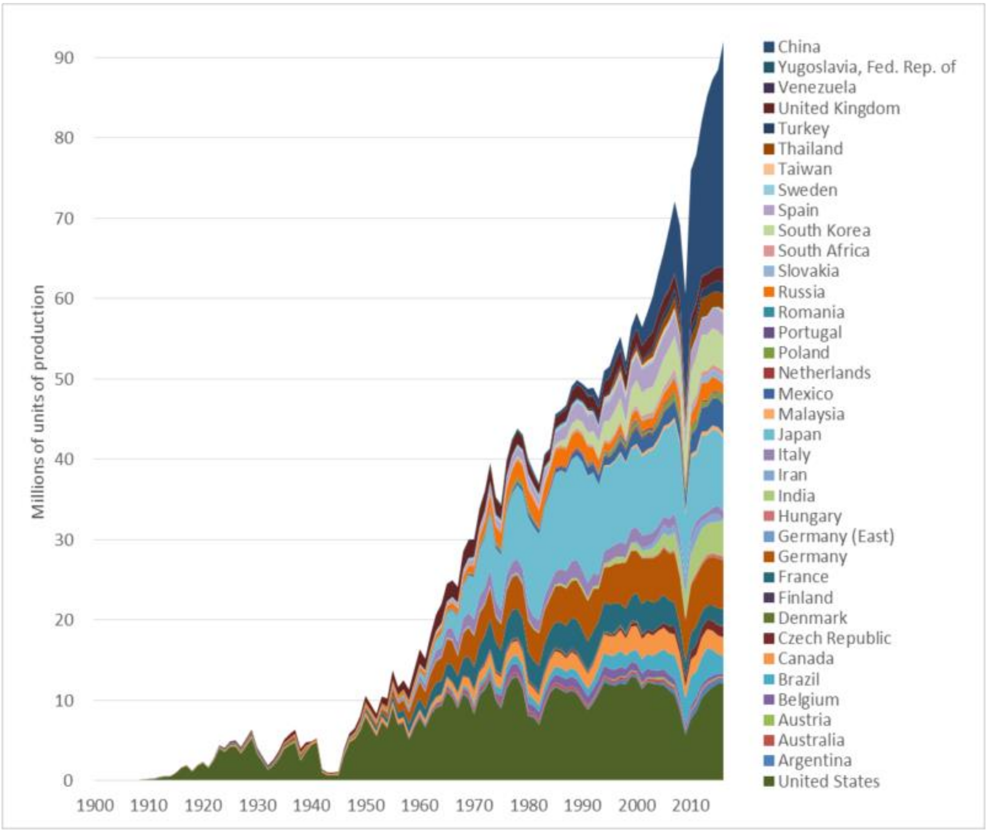


Figure 1-1: Production of cars by country from 1900-2016

cars from the year 1900 to 2016. Now in 2021, cars are very common around the world and have been normalized for decades.

As demand for cars is high, the prices of cars can be high and not very affordable for many people. Required supplies to produce cars can also be expensive. Considering these factors creates a competitive market. With the idea of depreciation, used cars can be cheaper than new cars, making them more easily accessible to certain people. Depreciation is the reduction in the value of an asset over time, due to wear and tear (DEPRECIATION | Meaning & Definition for UK English | Lexico.com, 2022), so as a car gets older, it would lose value because the material of the car would rot, and the engine and design can also not be as efficient and clean as a newer and unused car. Also, the model of the car can be outdated, missing any new and modern features. Based on these factors, used cars can still maintain their value due to the high prices of new cars.

1.1.2 Background of Data Science

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms, and systems to extract knowledge and insights from noisy data, and apply knowledge and actionable insights from data across a broad range of application domains (Data science - Wikipedia, 2022). Data science is useful for organizations or even individuals trying to make decisions such as predictions, forecasting, etc., with high accuracy. With the combination of data, statistics, and machine learning, accurate predictions and decisions can be produced to solve many problems.

1.2 Problem Statement

The main market population for used cars is applied in car rental companies, car insurance companies, and individuals. When determining the price of a used car, factors that lead up to its price can vary from what company, model, year, mileage, etc. the car is. There can be many factors that valuate the car, and these details can be complicated for some people to understand.

1.2.1 Problems for Individuals

Many individuals own cars, and many others would, over time, sell their car to upgrade to a newer or more advanced car, or buy a car of their preference. If an individual would like to sell their car, it can be confusing to choose what price their car should be sold for, especially if they do not understand much about what determines a car's value. It would be helpful to have an accurate estimate of the car's value before attempting to sell it.

On the other hand, an individual would possibly like to save some money and buy a used car. When deciding which car to choose, it would be best to understand the value of the car to ensure that the car bought would be of a suitable price. No customer would like to buy a less advanced car for the same price as a more advanced car.

1.2.2 Problems for Used Car Rental Companies

When car rental companies rent their cars to their customers, their prices should be reasonable and competitive when compared to other companies. These companies can have a service where individuals or organizations can use their services to find clients to rent their cars to, for a commission. Considering the commission, competition, and the use of used cars, it would be helpful to have a tool to help predict good rent and commission prices for them.

1.2.3 Problems for Car Insurance Companies

Car insurance companies mainly make money by setting premium prices for customers. Their service includes covering any costs of damage to a specified car, in return for a premium price. What determines the price of the premiums is the value of the car. This is because more valuable cars are more likely to have more expensive car parts and would, in total, have higher costs and fees. It would be helpful for car insurance companies to have a tool to check how valuable a car is, to better aid them in deciding the price of their premiums.

1.3 Aims and Objectives

1.3.1 Aims

The aims and desired outcomes for this project would be to construct a machine learning model that will aid in providing an accurate prediction of the value or right price of a used car based on the main factors that determine the price.

It should be simple for a user to enter the details of their car, and then present the price to the user. The algorithms and technologies used should be helpful to individuals that would like to buy or sell a car, or to the car insurance and rental companies that would be assisted in deciding the prices whether to sell or rent a car, based on the price of the premiums.

To individuals, there can be various cases where this program will aid them, such as if someone has been offered a car to sell, the individual can just input the details the program will require, and the algorithm can help give the individual the idea of the range of the price. This would help in avoiding buying overpriced cars. And this can also work even if an individual would like to sell their car. They can just input the details, understand the price of the car, and decide the price to sell it for. This can benefit them by avoiding the confusion of what price to sell it for, and therefore can help possibly minimize losses.

Used car rental/seller companies can benefit from this tool when determining the price to give to a car they would like to buy or sell, or whether it can be to minimize risk, so if there is a scenario where they have a car offer to buy, it can help when making pricing strategies to generate the highest number of profits.

Car insurance companies can also value this algorithm when placing prices for their premiums. Because when customers want to use insurance for their car, the insurance company would want to know the value of the car. Because depending on how expensive the car parts would cost, the premium the customer would pay would be higher (Cuvva, 2022).

This algorithm would benefit these companies by helping analyze the risks and estimations of possible costs, resulting in suitable premium prices.

1.3.2 Objectives

The objectives in achieving the aims of this project would require a dataset of cars with information such as the car company, model, price, year of production, mileage, fuel type, and more. Within the dataset, the common features that decide and influence car prices would be explored and analyzed. A machine learning algorithm would be implemented on the dataset and would return the predicted price of a car. The machine learning algorithm should aim to have accurate results, with at least 95% accuracy. Finally, once all implementations are complete, a user-interface form will be presented to the user where they can enter the details of a car and be provided with the price. The user interface should provide a simple experience where non-technical users can also obtain predictions.

1.4 Solution Approach

1.4.1 Data Collection

A dataset of used cars with their details, such as prices, model, year, fuel type, etc. must be obtained. The attributes of the cars must be collected into a single pandas data frame. To ensure depreciation is considered, mileage and year of make must be included.

1.4.2 Data Cleansing and Exploratory Data Analysis (EDA)

The data cleaning process involves steps such as clearing any missing values, replacing or removing data that is incorrect and handling any outliers. This will ensure that data is true in its nature and avoid any bias. EDA will help in understanding the nature, statistics, and any patterns of the data so that this information would help in deciding which attributes can be implemented into the machine learning model. For example, maybe the year and mileage of the car have more importance than what type of transmission or fuel the car uses, and therefore, focus should be more on the year and mileage. This will need some data analysis and visualization to confirm assumptions.

1.4.3 Data Pre-processing

Once the data is clean and the nature is understood, the data at this stage will have to be transformed into values that can be understood by the algorithm, such as label encoding, and splitting the data into training and testing sets for the algorithms to learn and get tested.

1.4.4 Implementation of the Machine Learning Algorithms

Once the data is processed, the machine learning models will be implemented. This method will compare three different machine learning algorithms that consist of linear regression, decision tree, and Catboost regression. The best-performing model will be selected and optimized with the best features and parameters.

1.4.5 Testing and Evaluation

With the help of the testing data, the final model will be tested and evaluated to assess its performance and significance. This will discuss the best model, features, and parameters used, and how accurate and reliable the predictions of the algorithm can be.

1.4.6 User Interface Form

This step will develop a user interface form where the user can input details of a car and be returned a prediction of the price. This should be a simple and user-friendly form that non-technical users can use with ease.

1.5 Summary of contributions and achievements

The result is a UI form where the program will request input of car details, which will then return the predicted price. The best machine learning algorithm for this implementation is the Catboost model, with optimizations resulting in 97% accuracy. This implies that the algorithm can be trustworthy and consistent with the help of cross-validation analysis.

1.6 Organization of the Report

The report of this study will contain and discuss the following chapters:

- The Literature review section will go more in-depth into the regression algorithms and will discuss research and analysis made on existing work of similar studies related to this area. Furthermore, a critique of the review that will compare the studies to this study will be included.
- There will be a methodology section that will discuss the study's problem and tasks specifications with the algorithms and tools used. Highly detailed steps of

implementation and the experiments used for testing will be demonstrated and examined.

- Then, there will be a section for results and testing that will present the findings, such as the best algorithm used, and its evaluation using metrics and the experiments discussed in the Methodology chapter.
- Then, there will be a discussion and analysis section which will discuss and analyze the results and reaffirm the significance of the study. Limitations discovered and learned throughout the study will also be provided.
- The Conclusion section will conclude the report with the findings and final thoughts, with a discussion on future work. The Reflection section will summarize a personal view of the experience of the study.

2. Literature Review

2.1 Review of the state-of-the-art

This section will give a background to readers for further understanding of predictive analytics and modeling, machine learning, and the regression algorithms that are used to help predict values.

2.1.1 Predictive Analytics

"Predictive analytics is the use of data, statistical algorithms, and machine learning techniques to identify the likelihood of future outcomes based on historical data. The goal is to go beyond knowing what has happened to be providing the best assessment of what will happen in the future." (Predictive Analytics: What it is and why it matters, 2022). This explanation appears to relate to a solution to the given problem of predicting the price of used cars. The price is the future outcome that this review is trying to predict, and the historical data would be the data of cars with any relevant information used to help get a more accurate prediction of the price. It would not be wise to just simply get, for example, the average price of the car as some factors and circumstances would adjust the price. Predictive analytics are being widely used in the workforce, especially in businesses requiring assistance in decision-making to attract more customers (Predictive Analytics Definition, 2022).

2.1.2 Machine Learning in Predictive Analytics

Predictive analytics and machine learning are often used interchangeably, as machine learning is a major tool for making predictions. In machine learning, there are two major models used to make predictions, which are classification and regression models (Predictive modeling, analytics, and machine learning, 2022). Classification models are used to predict classes. Banks can benefit from classification algorithms to determine the credit scores of customers. This would help when deciding whether a customer is eligible for a loan. On the other hand, regression models are used for numerical predictions. Regression models can be used by sports companies to determine the transfer fee of an athlete (Linear Regression: Real-life example, 2022). This would help sports clubs in helping make decisions in determining the fees or determining the club's value/assets based on the regressions model's predictions.

2.1.3 Regression Models

There are many types of regression models. The main models this section will be focused on are:

- Linear Regression
- Logistic Regression
- Ridge, Lasso, and Elastic Net Regression
- Decision Tree Regression
- Gradient Boosting Regression

Linear Regression: The linear regression model is the most used algorithm. It consists of predicting a dependent variable based on an independent variable(s). For example, determining the weight of a person based on their height. Usually, this would be an uptrend regression since often, taller people generally have more weight. Figure 2-1 contains a graph of linear regression. Consider the variable X to be the weight of the person, and the variable Y to be the height. In this case, to predict the height, the X value would have to be known (weight). Using the red line on the graph will give the predicted value of Y based on the corresponding value of X. This value is considered the predicted value based on this graph.

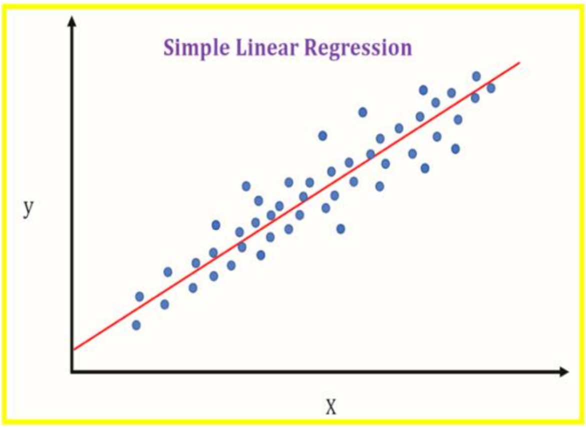


Figure 2-1: Linear Regression chart example

Logistic Regression: Logistic regression is mainly used to solve classification problems while linear regression is focused on regression problems. Logistic models are mainly used to determine discrete values, such as Yes/No or True/False, while Linear regression models are used to determine continuous values, such as age, price, and height (Dwivedi, 2022). Figure 2-2 represents an example of a logistic graph, with the blue samples representing true, and the red samples representing false.

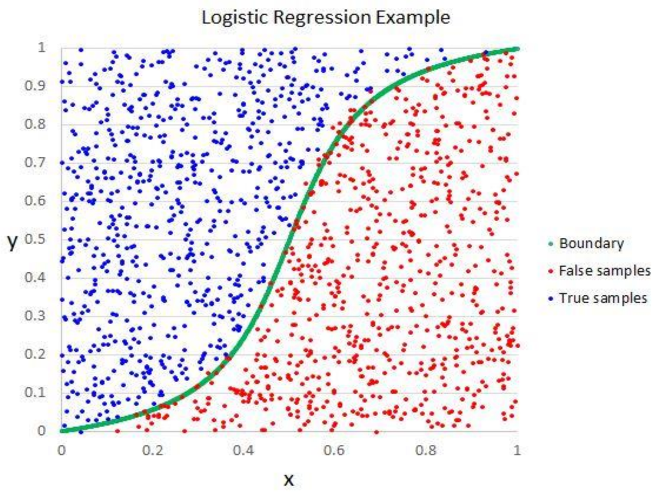


Figure 2-2: Logistic Regression example

Ridge, Lasso, and ElasticNet Regression: The algorithms shown in figure 2-3 will be explained together as they are all similar. These methods are regularized forms of linear regression. They all penalize the coefficients of linear regression. Ridge regression penalizes the sum of squared coefficients, while lasso regression penalizes the sum of absolute values of the coefficients. ElasticNet is simply the combination of both ridge and lasso models. (Oleszak, 2019)

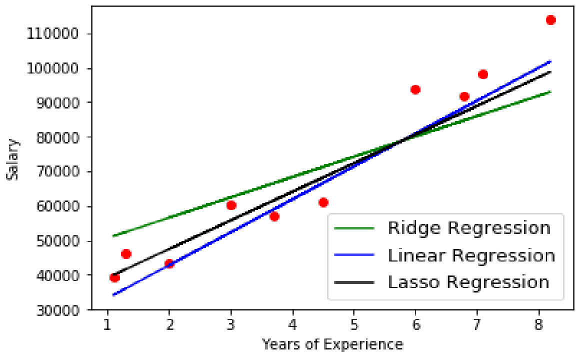


Figure 2-3: Graph representing Linear, Lasso, and Ridge Regression

Polynomial Regression: Polynomial regression models are used when the data on the graph is shown to have a polynomial equation, as shown in figure 2-4. Linear regression is used when there is a straight line, while polynomial regression is used when the best fit line is curved.

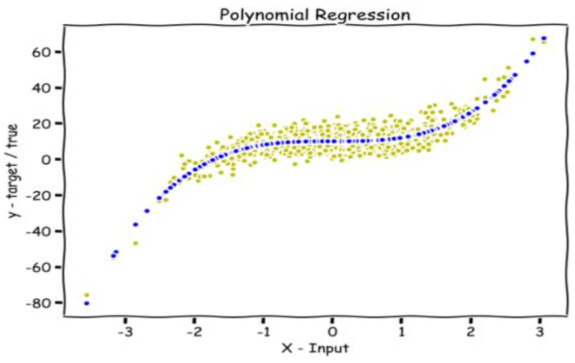


Figure 2-4: Polynomial Regression Example

Decision Tree & Gradient Boosting (Ensemble) Regressors: “Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.” (Decision Tree Tutorials & Notes | Machine Learning | HackerEarth, 2022). “Gradient Boosting Regressors build trees one at a time, where each new tree helps to correct errors made by the previously trained tree.” (Abolfazl Ravanshad, 2018). Like decision trees, gradient boosting regressors are known to be a more enhanced version. What differentiates these methods from the previously described algorithms is that they can be used for classification and regression problems and are mainly tree structures. There are different enhanced versions of Gradient Boost Regressors which have grown popular. The most common algorithms are the XGBoost and CatBoost Regressors.

2.2 Research and Analysis of existing and similar work

This section is dedicated to the research of similar applications using machine learning for predicting car prices, and the regression algorithms used within the domain of predictive analytics.
Similar applications that are going to be discussed are:

- Predicting Prices of Houses
- Predicting Prices in Sales Forecasting
- Predicting Prices of Laptops

The Actual Cash Value, a method that does not use data science, and existing research on predicting car prices with machine learning are also going to be researched and analyzed.

2.2.1 Prediction of Prices of Houses

As widely known, houses have a price based on their value which depends on multiple factors. The factors that influence house prices are the following (6 Factors That Influence a Home's Value, n.d.):

- Historical Sale Prices
- Neighborhood
- The Market
- Size and Appeal
- Age and Condition
- Nearby Features

These factors tend to influence the price of a home. Whether a homeowner is intending to sell or rent their house, it would be beneficial to understand the value of it to be able to manage assets. There can be a case where a homeowner has an expensive home but is not able or is struggling to support their expenses or debts. Understanding the value of their home can help support their money/asset management, so it would be best in this case to sell their home to move to a new and affordable one where expenses would not be an issue. Another reason to know the value of a home is it would help to know if it could be a good idea to add home improvements for the reason of increasing its value. If the home is priced already highly, it might not be the best idea to add home improvements for the sake of increasing the home's value (Wohl, n.d.). With these explanations of why it is best to understand a house's price, it

is clearly like the main problem, is understanding a car's value. Factors such as the age, condition, size, location, number of bedrooms, number of bathrooms, number of floors, etc. can likely correlate with the price of a home. Using Data Science and Machine Learning can assist in giving a prediction of the price of a home. Firstly, if there is a dataset containing prices of homes with its features, it can be analyzed to understand the features that would influence the price, and a machine learning model can be implemented to give predictions. Since price is a continuous variable, regression algorithms will be used.
Existing studies researched:

Prediction of House Price Using XGBoost Regression Algorithm (Et. al., 2021)

In this study, an ensemble algorithm, specifically the XGBoost regressor is implemented to predict house prices. The significance of this writer's project is similar to the reasonings of this project, which is mainly because the constraints that determine the price of a house can be complex for a human to predict, and it would be much more efficient to implement an algorithm to tackle this problem and give an accurate prediction. This can benefit bankers when giving mortgages, and individuals when a homeowner is confused and unaware of the value. The data pre-processing used by the writer is divided into four stages that include data cleaning, data editing, data reduction, and data wrangling. After pre-processing the data, the XGBoost algorithm was implemented on the writer's chosen dataset. Their results were tested and modified to compare different training, testing, and validation data, and the best model was chosen.

This study predicts the prices of houses using XGBoost Regressor. The significance of this study is to benefit individuals and companies. Banks can use this when calculating mortgage rates, and individuals can use it when trying to sell their house or buy a new house. Using data science can provide a huge benefit. The pre-processing techniques are the basic steps, such as cleaning, encoding, and visualization. The exploratory data analysis was not shown in the study, and the data described was not detailed enough to understand the process and nature of the data. Only the XGBoost regressor algorithm was used. This could have been extended to using multiple ensemble algorithms, including the Catboost algorithm which is helpful for categorical data as required by the researcher. The training, testing, and validation data were split into different sizes to see the best model. The metric used was the MSE value. This could have been extended to multiple metrics for better evaluation. No form of optimization has been done, which could have made the model even more accurate.

Overall, the study was significant, but the methodology could have been enhanced to result in a higher quality predictor.

Machine Learning based Predicting House Prices using Regression Techniques (J. Manasa, 2020)

In this study, the dataset used by the researcher is of houses in Bengaluru, India. It has factors such as area type, availability, size of bedrooms, society, size in sq ft, location, price, etc. EDA analysis was performed to find which factors had the most significance when determining the price, but it has not been discussed clearly which features determine the price. The categorical data is then encoded, scaled, and then the data was split between the training and testing data. The machine learning models implemented were linear regression models, including the basic linear, lasso, ridge, support vector machine, and XGBoost. The metrics used on this were the r-squared, root mean squared error, and root means logarithmic error. The most successful algorithm was the XGBoost, being the most stable, having the fastest execution speed, and having the least variance.

Compared to the previous study, this study created a machine learning model to predict house prices in Bengaluru, India. India in general is highly populated, and the house price would differ compared to most studies made in western countries. The dataset used by the researcher was clear, containing the area type, size, location, etc. This was clearer than the previous studies' description of their dataset. Clear exploratory data analysis was performed, considering all factors about the price. This study did not use validation sets in the training and testing stages, while the previous study did which ensured a more reliable model. Linear, Lasso, Ridge, SVR, and XGBoost algorithms were implemented, which is a wider range compared to the previous study. The metrics used were the R-squared, RMSE, and RMLSE scores, which is a good range. The model did not have any optimizations, which could have led to a more precise model. Overall, the study was significant considering Bengaluru house's valuations depended differently than the western country houses. The implementation techniques were acceptable, but a few adjustments could have led to greater results.

2.2.2 Prediction of Prices in Sales Forecasting

This section is broader than the other applications that have and will be discussed. Sales forecasting is the study in businesses that based on some factors, to estimate the possible revenue that a business can earn. This is done by predicting the number of products or services a sales unit will sell in the future (Anaplan, 2018). Sales forecasting benefits companies with managing risks, decision making, and managing preparation by understanding the resources needed in the future.

The possible factors that affect sales forecasting can be (TEAM, 2020):

- Economic Conditions
- Business Conditions
- Internal Conditions
- Sociological Conditions
- Psychological Conditions
- Competitive Conditions

Sales forecasting overall is a common practice among businesses, and it can be both complicated in predicting the future revenue, and businesses must be fast and efficient when applying sales forecasting to stay up with their market and competition in case of any sudden changes in the conditions stated above. An algorithm when implemented correctly to meet a business's requirements can help much more profoundly, especially in cases where time is an issue.

Existing studies researched:

E-commerce Sales Forecast Based on Ensemble Learning (Zhan, Li, Jiang, Sha, & Guo, 2020)

In this study, the factors that influence the sales of products are investigated to develop machine learning models to predict an important strategy by e-commerce companies, called the procurement strategy which is used to help reduce overall costs. The data used in this study contained several hundred e-bay accounts that had sales of hundreds of thousands of products. The machine learning algorithms implemented on the data were the gradient boosting decision tree, CatBoost, XGBoost, and LightGBM. The metrics used by this model were the mean absolute error, root, normal mean squared error, and the r squared score. The model was tested with various hyperparameters to return possibly higher scores. The most successful algorithm applied was the CatBoost algorithm, containing an r squared score of 0.6 after applying hyperparameters.

This study is attempting to predict sales of products in the e-commerce sector, as it is a highly growing sector that will benefit businesses, especially in decision making. The study takes into consideration mainly the dates of products, which checks the impact of sales during holiday seasons. The data used contains sales of more than 200 million USD and contains e-bay accounts. Not much detail is given about the data, other than their date and that 31 categories are involved. Exploratory data analysis shows the sales of products based on year and countries, whereby year sales are increasing, and country sales come mainly from North America and Europe, which is good enough to understand the model's needs. The machine learning algorithms are of ensemble methods, which include Catboost, GBDT, LightGBM, and XGBoost. The metrics used were the MAE, RMSE, MSE, and R-Squared, which is great considering a wide range of metrics. Optimization was done to configure models, the best model was the Catboost model and the highest r2 score obtained was around 0.6, which is acceptable. Overall, this study used a proper strategy. The dataset used could have been more detailed providing information such as the age groups the sales were made by, and more, but this can be justified in case of the privacy policies.

Forecasting new product demand using machine learning (P S Smirnov and V A Sudakov 2021 J. Phys.: Conf. Ser. 1925 012033)

In this study, machine learning models are implemented to predict how much demand a new product will gain based on its features and characteristics. The main goal is to not make this model restricted to limited products. This can help businesses in increasing potential profits and reducing costs due to a better understanding of the possible demand for the product can rise. The data obtained was from a Russian e-commerce company called Ozon. The dataset used contained the categories of the products, such as sport, furniture, books, etc., with it its date, price and sales, and the average sales of the data. Exploratory Data Analysis was performed and showed that products between June and October had the least sales, and the average sales have been increasing year by year. The data was then split between training, testing, and validation sets, and the Gradient Tree Boosting algorithm was applied to the data. The model was then optimized using the random search optimizer, and the metric used was the RMSE scores. The best model obtained had an RMSE score of 4.00129. And the most important feature in the model was the day of the year of the product.

Compared to the previous study, this calculates the demand for products. The data is clear as it is using the date, price, identity (category), and aggregates. This is a good and clearer choice of data. The average sales were compared by month and year. The month is necessary to analyze more than the year, which the previous study did not do. This study used training, testing, and evaluation whilst the last study only used training and testing, which is a safer choice to ensure the model is not biased. The model used was only the Gradient Tree Boosting for regression. Other models could have been used and compared with each other's to get the best model. For metrics, only the RMSE was used. Other metrics could have been used to give better assurance of the results, mainly the r2 score, as it is not so clear how efficient the model is. The model has been optimized, which is good. Overall, this was a good study that has helped benefit the rising sector of E-businesses.

2.2.3 Prediction of Prices of Laptops

Laptops are commonly used nowadays in the 21st century. It has helped individuals and organizations in productivity, communications, entertainment, and much more sectors. It has become a new norm to own a laptop in the house. But people can find owning a laptop pricy, especially when they would need it for simple reasons, such as messaging and emailing people. In these cases, it would be better in finding a used laptop to help with these simple tasks. Or it can be simply that people would prefer a specific kind of laptop, but the current make of that laptop can be pricy, so they can buy a used version of that kind.
The factors that can influence a price of a laptop can be (Skrebiec, 2013):

- Screen size
- Processor speed
- Hard drive capacity
- Hard drive speed
- Weight
- Battery life
- RAM speed
- RAM size
- Laptop thickness
- Average consumer ratings

With all these factors, it can be confusing to individuals not specialized in computing, such as the elderly, whether they would like to buy or sell a used laptop. The factors listed above can sound confusing to some people, and when it would come to determining the price, it can be an issue to get a fair price that it’s worth. It would be much more efficient and simpler if an algorithm can give a prediction to give a right price of a laptop based on the factors.
Existing studies researched:

Predicting Laptop Prices Using ML (Guozhen, Analytics Vidha, 2021)
In this study, machine learning was implemented as a technique to give a prediction of the price of the model. The following steps are taken:

- Obtaining Laptop Prices dataset
- Basic data exploration
- Feature Engineering
- Explanatory Data Analysis (EDA)
- Data Preprocessing
- Modeling
- Building web interface
- Deployment of web application

Firstly, the data of laptops was obtained that contained the laptops’ company, product, type, screen resolution, CPU, RAM, memory, GPU, operating system, weight, and the target variable which is the price. The writer has done some feature engineering to separate relevant and irrelevant data, such as touch screens, and memory types and the GPU was removed as it contained a lack of consistency. After, an EDA analysis was performed to check which features influenced the price the most. The RAM, size, and CPU contributed the most to the price. After this, the data was ready to be pre-processed, with categorical data being both either label encoded or done by one-hot encoding. The data was then split between training and testing data, with the testing data being 1/3 of the data. Then the model has been implemented using Random Forest Regressor, Decision Tree Regressor, Linear Regression, and XGBoost Regressor. The metrics used to test the data were the r2 score and the mean absolute error, and the XGBoost model was used as it is the most accurate. Finally, a web app was deployed containing the algorithm to make it user-friendly to try and predict the price of a laptop.

This study was significant, and the methodology has been implemented properly. The feature engineering was done necessarily by separating good features, such as touch screen, memory, and HDD. The EDA analysis was interestingly done which showed the significant features and how they are affected by price. The encoding decision was good as both label and one-hot encoding were done where necessary. The models and metrics chosen for evaluation were good and the XGBoost Regressor was chosen with an r2 score of 0.75. The web app deployed by the user was very nice and was the only study in this review where a web app was deployed, which makes it simple for a non-technical user to use. The only problem with this solution is that it would be better to make the model even more accurate by using hyperparameters for optimizations of the model, as a 75% accuracy might not be enough for a set of users. But this study was mainly a tutorial for machine learning in general, and, understandably, it should not be a highly detailed study as it is mainly for beginners in the area.

2.2.4 Predicting Car Prices (using ACV & Machine Learning)

Actual Cash Value (ACV)

Actual Cash Value is a method used by car insurance companies to calculate the price of a car minus the depreciation cost. This method can be used for properties such as houses and cars (Kagan, n.d.). The Actual Cash Value is calculated in the equation (Actual Cash Value Calculator, n.d.):

$$ACV = \frac{R \times (E - C)}{E}$$

ACV = Actual Cash Value
R = Replacement Cost or Purchase Price of the Item
E = Expected Life of the Item
C = Current Life of the Item

The life of the items represents the way that depreciation is calculated. Using this equation, the insurance companies would estimate the right price of the item in case the item has been destroyed or unusable.

Insurance companies use the ACV equation to give prices. But insurance companies might not be the best to trust when giving their customer's a true value. According to (Brozic, 2021), has said "If you disagree with the insurer's valuation, you may be able to negotiate a higher payout. However, you will need to have the evidence to back it up." This can cause problems for customers who lack skills in negotiating, and it can be a bother to keep looking for different insurance companies. There can be cases where insurance companies would give a highly overvalued estimate to mislead customers into thinking that the premiums paid are much more costly than they should be, especially for customers that are not aware of cars and their fixing costs and what they can be truly owed. A machine learning model that can give customers a quick and accurate estimate can help people negotiate with evidence into what the customer should pay for their premiums, and give customers an ease of mind that they are paying the right price for their insurance and will therefore help in their making their decision in choosing the right company.

Application of Machine Learning Techniques to Predict the Price of Pre-Owned Cars in Bangladesh (Amik, F.R.; Lanard, A.; Ismat, A.; Momen, S., 2021)

In this study, four machine learning algorithms are implemented, compared, and deployed to predict the prices of cars. The study aims to deploy a web application, making it accessible to a populated and small country such as Bangladesh. The dataset used contains car prices of cars in Bangladesh with information such as car name, model, year, transmission, kilometers run, etc. The data was pre-processed to make it suitable for the algorithms to learn from. Furthermore, EDA analysis is performed, and missing values and outliers have been removed or handled. Highly correlated features have been removed by using the Pearson coefficient, the suggestion is that highly correlated features will represent the same value and are not needed. The clean data is then split and scaled before being trained. The algorithms implemented are Linear regression, Lasso regression, Decision Tree, Random Forest, and XGBoost. The metrics used for model evaluation are the r2, RMSE, and MAE scores. All the models have been optimized, and the best model used is the XGBoost model, scoring an r2 score of 91.32%. The final model was then deployed and integrated with an HTML page, where the user can input values and have a predicted price returned to them.

Overall, this study is significant, as it has provided users in Bangladesh with a user-friendly system to interact with the machine learning model. Steps such as removing outliers, and feature selection have made the model more accurate, reliable, and efficient. All the algorithms implemented have been hyper-parameter tuned. This is good, however, the best performing model before optimizing can just be selected, as it is unnecessary, and can be time-consuming. In this case, only the XGBoost (r2 = 91.32%) and Random Forest (r2 = 90.14%) can be tuned, and the best-performing model can be selected as the final model.

2.3 Critique of the Review

The main findings conclude that it can be complex to calculate these prices considering factors such as age and depreciation, whether it comes to houses, sales, laptops, etc., and it would be much more efficient and stress-free to use an algorithm to predict a right price. Most of the studies used multiple algorithms to return the best model, multiple metrics to confirm a model's accuracy, and some studies used optimization techniques to return an enhanced output. Only the laptop and car price study contained a form for simplicity of using the algorithm, which is helpful for non-technical individuals. Considering this, the car price predictor model that is going to be implemented will consider the positives of the studies, which will mean testing multiple algorithms with multiple metrics, optimizing the best performing algorithm, and then implementing an interactive form to add simplicity for users that will use this model, because it can be assumed that most will come from a non-technological background. The study will also look more into the Catboost algorithm, as it is assumed that it can be a good model for this problem, as there are categorical data such as the company name, model name, etc.

2.4 Summary

To conclude, this chapter has given this study a more in-depth background of predictive analysis and the applications involved, including machine learning. The next chapter will make use of this knowledge and apply it to aid the implementation of the car price predictor.

3. Methodology

3.1 Problems (tasks) descriptions and specifications

The problem statement is to build a machine learning model that can predict the price of a car. This is a complex problem as there are many factors to consider when valuing a car. The following tasks will be done to have a fully working machine learning model:

- Collect data that have the relevant features to be used in the model.
- Perform data cleansing and exploratory data analysis of the data to extract insights into the data, and to prepare it for pre-processing.
- Perform data pre-processing on the clean data to prepare to be fed to the model.
- Implement, compare, and evaluate three different machine learning algorithms
- Create a UI form that takes users' inputs and displays the predicted price.

3.2 Algorithms/Tools/Technologies/etc. descriptions

The machine learning algorithms that are going to be implemented are the Linear Regression, Decision Tree, and Catboost regression algorithms. This gives a nice variety of the different regression algorithms, one consisting of multiple linear regression, a tree, and gradient descent.

The programming language that will be used for this implementation is Python. It is a highly significant language for data scientists. According to the Stack Overflow Developer Survey in 2021, Python is the 3rd most popular programming language. And when it comes to machine learning, according to GeeksforGeeks, Python is the 1st most popular language. With the help of its libraries, Python is best for its simplicity and ease of use when creating machine learning models.

The Pandas library can help in data manipulation and extracting data insights from datasets.

The Matplotlib and Seaborn libraries help in creating intriguing data visualization of data, which helps in decision making.

The Scit-Kit Learn library AKA sklearn helps in creating machine learning models very easily, it consists of many machine learning algorithms that make it easy to implement.

The Jupyter Notebook by Anaconda will be used, which allows markdown and python scripts to be implemented together. This can make it much more friendly to use as multiple outputs can be shown within a notebook, which will help in the EDA analysis.

3.3 Implementations

3.3.1 Data Collection

To implement the machine learning model, the first step of the process would be to collect the data relevant for modeling. The data should contain independent variables that determine the value of the car. In previous chapters, it was discussed that depreciation is a factor that determines a car's price; therefore, the age of the car should be provided. The mileage of a car should be provided as well since it determines how much the car has been used, which would be a price factor. Other than that, there are luxurious cars, and there are budget-friendly cars. Considering this, the make and model of the car should also be provided. Other than the make, model, year, and mileage of the car, there should be some additional data that can be determined by this. Collecting every car make in the world to add to the data would not be realistic, therefore a limited number of companies will be used. The dataset that satisfies these requirements has been obtained from Kaggle.com. The data contains 100,000 used cars in the UK. The dataset provides the necessary attributes required to build this model. Nine car makes are also provided, which is suitable.
The final data has the following columns as shown in table 3-1:

Company	Model	Year	Transmission	Mileage	Fuel Type	Tax	MPG	Engine Size	Price

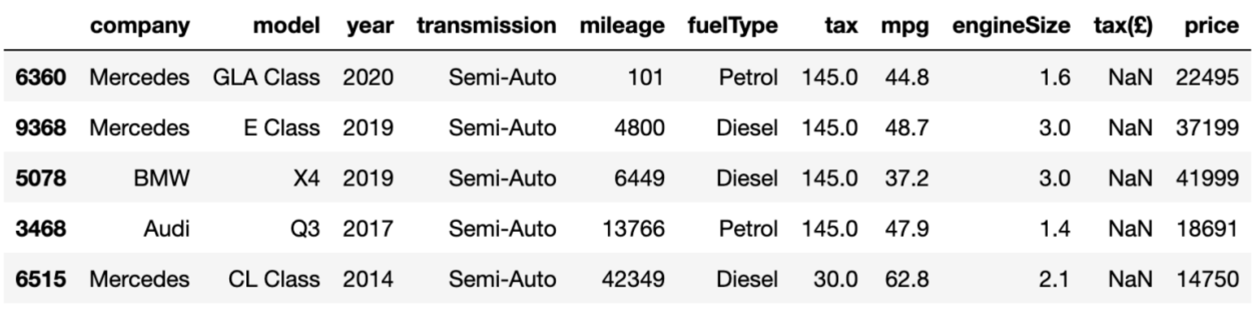
Table 3-1: Columns of the dataset

3.3.2 Data Cleaning and Exploratory Data Analysis

This process will contain several steps which are:

- Merging files.
- Ensuring there are no whitespaces.
- Ensuring data types are correct.
- Ensuring data values are 'normal'.
- Detecting and removing outliers.
- Discovering patterns and confirming assumptions.
- Removing unnecessary columns.

The dataset obtained contains separate CSV files for each car company. Each file will be made into a pandas dataframe and then merged into one pandas dataframe. Before that, each file should have a new column called ‘company’ containing the name of the company to be able to differentiate between them.
After merging, the dataframe will be displayed as in Figure 3-1.



	company	model	year	transmission	mileage	fuelType	tax	mpg	engineSize	tax(£)	price
6360	Mercedes	GLA Class	2020	Semi-Auto	101	Petrol	145.0	44.8	1.6	NaN	22495
9368	Mercedes	E Class	2019	Semi-Auto	4800	Diesel	145.0	48.7	3.0	NaN	37199
5078	BMW	X4	2019	Semi-Auto	6449	Diesel	145.0	37.2	3.0	NaN	41999
3468	Audi	Q3	2017	Semi-Auto	13766	Petrol	145.0	47.9	1.4	NaN	18691
6515	Mercedes	CL Class	2014	Semi-Auto	42349	Diesel	30.0	62.8	2.1	NaN	14750

Figure 3-1: Sample of the dataframe

Looking at Figure 3-1, there are two tax columns. A heatmap of the data’s missing values using the seaborn library has been produced to see if there is any relation between the two columns.

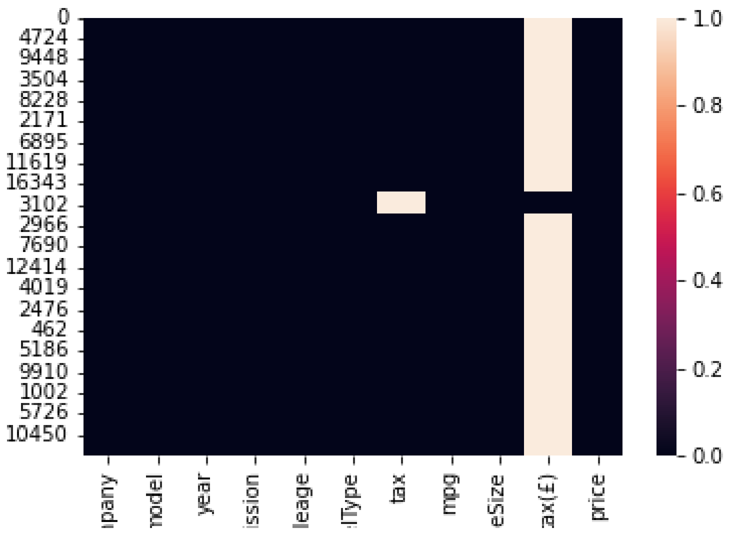


Figure 3-2: Heatmap showing missing values

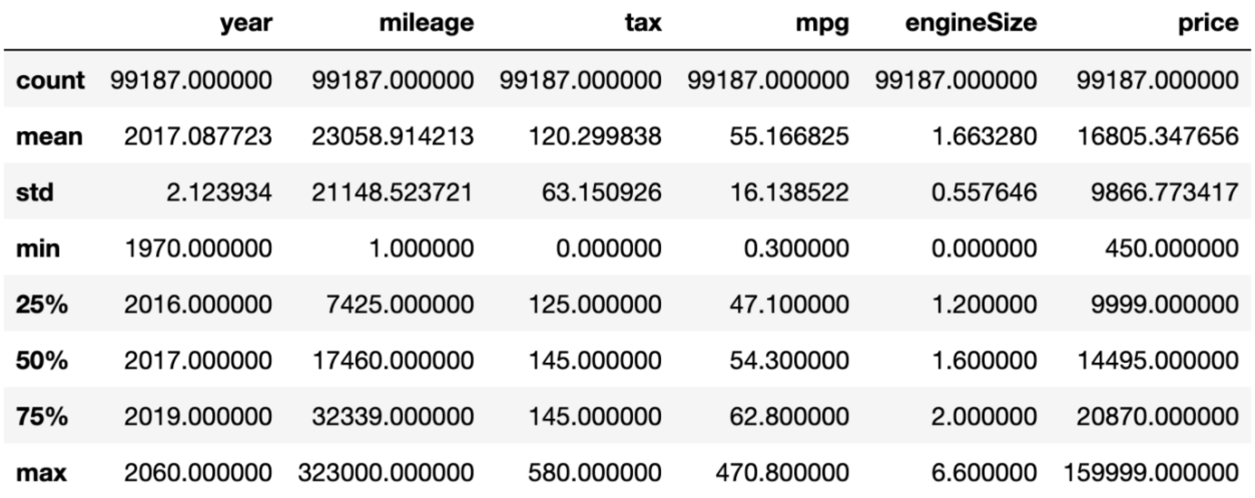
Clearly, the missing data in the normal tax column is in the tax(£) column as shown in Figure 3-2, therefore the missing values will be moved to the tax column before removing the tax(£) column. This will be done by executing the following code:

```
df['tax'] = df['tax'].fillna(df['tax(£)'])
df = df.drop(columns=['tax(£)'])
```

In some columns, there are extra whitespaces or spaces for some values. To remove this, the following code will be run:

```
for i in df.columns:
    if df[i].dtypes == 'O': # For columns of object type
        df[i] = df[i].str.strip()
```

To get a better insight into the data, the pandas .describe() function will show the statistics of the quantitative values.



	year	mileage	tax	mpg	engineSize	price
count	99187.000000	99187.000000	99187.000000	99187.000000	99187.000000	99187.000000
mean	2017.087723	23058.914213	120.299838	55.166825	1.663280	16805.347656
std	2.123934	21148.523721	63.150926	16.138522	0.557646	9866.773417
min	1970.000000	1.000000	0.000000	0.300000	0.000000	450.000000
25%	2016.000000	7425.000000	125.000000	47.100000	1.200000	9999.000000
50%	2017.000000	17460.000000	145.000000	54.300000	1.600000	14495.000000
75%	2019.000000	32339.000000	145.000000	62.800000	2.000000	20870.000000
max	2060.000000	323000.000000	580.000000	470.800000	6.600000	159999.000000

Figure 3-3: Summary of the statistics of the quantitative values

Notes to consider from the statistics in Figure 3-3:

- In the year column, 2060 is the max value, which is incorrect.
- The engine size column has 0 as a minimum value. This is assumed to define electric cars.
- The minimum price of 450 is not realistic, and so is the value of 0.3 miles per gallon.
- The minimum values with the 25% percentile and the maximum values with their 75% appear to both have a large gap. This assumes there are outliers in the data.

To ensure the years are correct and real, this code will be run:

```
df = df.loc[df['year'] <= 2020]
```

To ensure engine size values of zero are electric, a check will be made by checking what the cars' fuel types are. By using the following code:

```
df.loc[df['engineSize'] == 0]
```

After running the statement above, the output will return that there are only 273 rows representing electric cars. It would be best to remove all these rows including the electric cars as there would not be much data to support them. The following code will remove these rows:

```
df = df.loc[df['engineSize'] != 0]
```

Therefore, it would also be best to remove any electric cars in the data, as by far they only have 4 cars left.

```
df = df.loc[df['fuelType'] != 'Electric']
```

Before going any further, the data types should be correct. Using the pandas .info() function, the data types can be verified. After checking this, the data types are truly correct and therefore can move forward.

The next step is to detect and remove any outliers to prevent any model bias with the help of boxplots from seaborn. Firstly, each company's outliers will be shown by using the following code:

```
sns.boxplot(data=df, x='company', y='price')
```

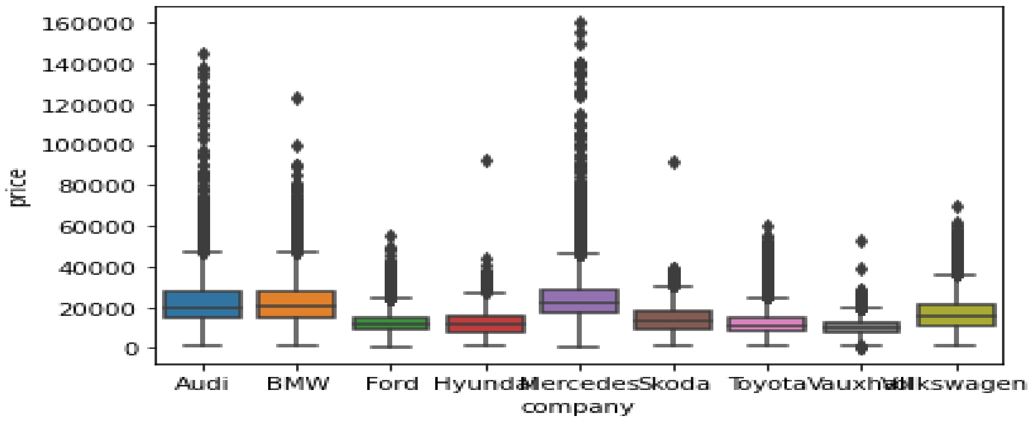


Figure 3-4: Boxplots of prices of companies

Figure 3-4 shows that the companies containing outliers that need to be removed are BMW, Hyundai, Skoda, and Vauxhall cars. There are 5 total outliers.

The outliers shown in Figure 3.4 will be removed by running the following code:

```
company_outliers ={ # Manually listing the outliers
    'BMW' : 100000,
    'Hyundai' : 60000,
    'Skoda' : 60000,
    'Vauxhall' : 30000,
}

for company in list(company_outliers):
    value = company_outliers[company]
    df = df.loc[~((df['price'] > value) & (df['company'] == company))]
```

The rest of the outliers in nominal values are the year, mileage, and mpg. The following code will display a boxplot of the column (only the code of year will be shown for demonstration):

```
sns.boxplot(x=df['year'])
```

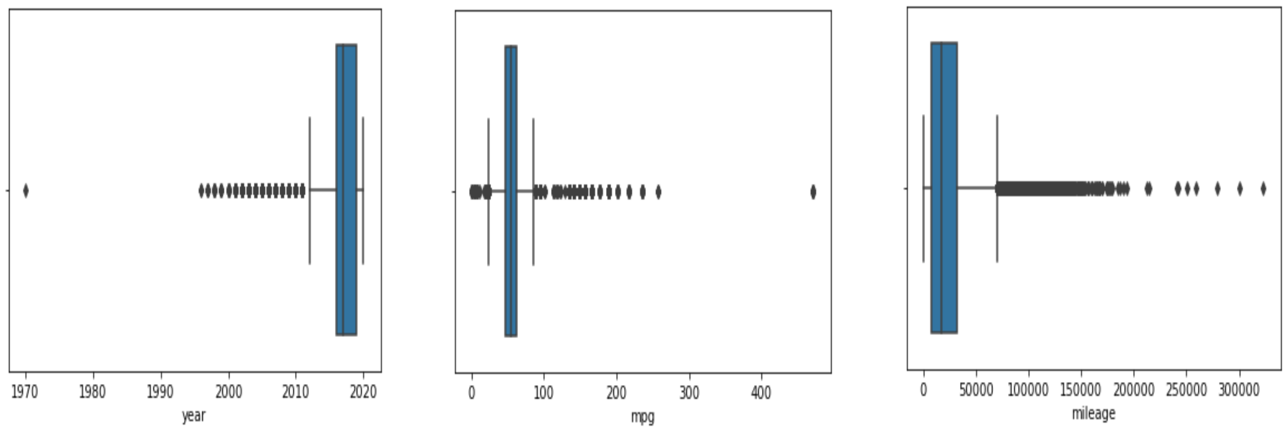


Figure 3-5: Boxplot of year, mpg, and mileage

As shown in Figure 3-5, outliers are available in the year 1970, with mpg of over 400, and mileages of over 200,000. 17 total outliers have been removed by running the following code (year as an example):

```
df = df.loc[df['year'] > 1970]
```

The remaining potential outliers can also be in the categorical columns which include transmission and fuel type. Pie charts of the counts of the categories will be represented, and categories that have a low count will be removed.

The following code will display the pie chart (transmission as example):

```
# Creates pivot table of the mean of prices of each transmission type
table = pd.pivot_table(df, values='price', index=['transmission'],
aggfunc='count')
# Creating pie chart
plt.pie(table.price, labels=table.index)
# Adding title
plt.title('Counts of cars by the Transmission Type')
# Showing graph
plt.show()
```

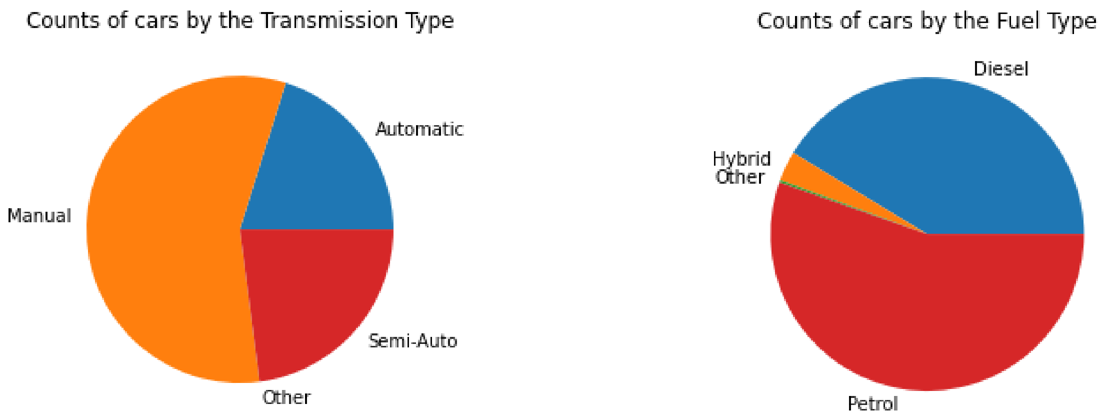
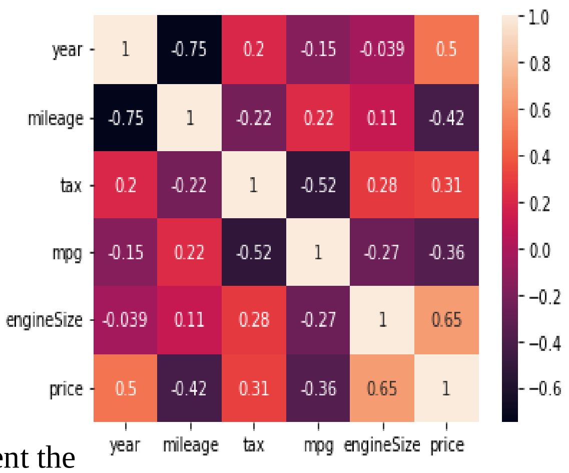


Figure 3-6: Pie charts of counts of Transmission and Fuel Type

In Figure 3-6, the ‘Other’ value is shown in both columns and has little data to back it, therefore it is best if this is removed. There are 254 outliers located. The following code will remove values of Other (transmission as an example):

```
df = df.loc[df['transmission'] != 'Other']
```

A total of 276 total outliers have been removed, making the data less prone to error. Before moving forward to preprocessing, it would be best to view a correlation of the whole data to confirm any assumptions, such as if the higher the year, the higher price, or the higher the mileage, the lower the price. This will help when analyzing the models to ensure the results are true in their nature. In Figure 3-7, a heatmap has been plotted to represent the correlations between the attributes of the data.



	year	mileage	tax	mpg	engineSize	price
year	1	-0.75	0.2	-0.15	-0.039	0.5
mileage	-0.75	1	-0.22	0.22	0.11	-0.42
tax	0.2	-0.22	1	-0.52	0.28	0.31
mpg	-0.15	0.22	-0.52	1	-0.27	-0.36
engineSize	-0.039	0.11	0.28	-0.27	1	0.65
price	0.5	-0.42	0.31	-0.36	0.65	1

Figure 3-7: Heatmap of numerical data

3.3.3 Data Pre-Processing

The next step will be to implement pre-processing of the data so that it can be understood by the machine learning models. These steps will contain the following processes:

- Splitting the independent variables with the target variables
- Encoding categorical columns
- Splitting the training and testing sets by using cross-validation

The first step will be to split the independent and target variables into separate variables so that the target can be specified for modeling. This will be done simply as follows:

```
# Splitting target variable (price) and independent variables
X = df_le.drop(columns=['price'])
Y = df_le['price']
```

Encoding the categorical columns into numerical values will be needed since machine learning models can only process numbers. The values of a categorical column will be separated by numbers that will replace text. With the help of sklearn's preprocessing module, the LabelEncoder will be imported to encode the categorical columns into numbers. The columns that will be encoded are, company, model, fuel type, and transmission. The following function will encode the categorical columns of the data (only the company shown for demonstration):

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
def encode(df_le):
    # Only company shown as example
    df_le['company'] = le.fit_transform(df_le['company'])
    return df_le
```

The final step for preprocessing is splitting the data between training and testing sets to evaluate the model for accuracy and other metrics. To ensure results are valid and will not suffer from overfitting, cross-validation will be used to give scores of the model. The technique that is going to be used is the KFold from sklearn's model selection module. This will split the data into 10 (k) folds and returns k results. The data will be split by running the following code:

```
# K fold splitting training and testing data
kf = KFold(n_splits=10, shuffle=True)
for train, test in kf.split(X):
    X_train, X_test, y_train, y_test = X[train], X[test], Y[train], Y[test]
```

The data is now ready to be implemented into the model. However, to justify not scaling the data, this will not be needed unless the linear regression or decision tree outperforms the Catboost algorithm. This is because Catboost is a powerful algorithm that is likely to perform well, and if used, scaling will only be an impractical step as the algorithm does not require it. *(Boost your model's performance with these fantastic libraries, 2022)*

3.3.4 Machine Learning models

Now the training sets are ready to be implemented into the machine learning models. The models will be fitted with the training data, and then tested and cross-validated with the testing data. The three machine learning algorithms that will be implemented and compared are the linear regression, decision tree regressor, and catboost regressor models. A function (*evaluate_model()*) will be made that takes a machine learning model, X (independent) and Y (target) variables. The X variables get split into training and testing sets, fit the model with the training data, predict the test data, and will get evaluated with the r2 score, and the results are plotted in a scatter plot with a line that represents the accuracy by correlation. Cross-validation scores will be included using the RMSE metric (since it can be easier to decide better parameters for model tuning later) and a boxplot of the results will be plotted. If parameter tuning will be used, the best parameters also get displayed. The full function can be shown in the Model Implementation section in the Appendix; however, the basic idea of the function will be implemented as follows:

```
# Function that takes a model, features, and a target variable. This will
print the metrics (r2 & rmse) of the model scored
def evaluate_model(model, x, y):
    model.fit(X_train, y_train) # Fit the train data
    y_pred = model.predict(X_test) # Predict the test data
    r2 = str(round(r2_score(y_test, y_pred)*100,2))+'%' # Model metrics
    # Cross Validation
    cvrmse = -1*cross_val_score(model,x, y,cv=kf,
                                 scoring='neg_root_mean_squared_error')
    cvscore = round(np.median(cvrmse),2)
    # Displaying scores
    print(f"R2 SCORE: {r2}\t",
          f"CV (RMSE) Score: {cvscore}\t")
```

3.3.5 Model Tuning

The next step will be to tune the model. The two procedures will be:

1. Feature selection
2. Hyperparameter tuning

Before selecting features, the features' importance of the model will be obtained, sorted in a table with their features, and plotted as a horizontal bar chart (Figure 4-1). This will be done with the help of the get_feature_importance() function, which will give values of the importance of the features. From this plot, if there are features that go very low (e.g. <1%), then that column will get removed. Algorithms for feature selection will not be used as some models, e.g., the Catboost model, cannot be manipulated.

```
# Feature importance of model
feature_imp = pd.DataFrame(model.get_feature_importance())
feature_imp.index = df.drop(columns=['price']).columns
feature_imp.columns = ['feature importance']
feature_imp=feature_imp.sort_values(by='feature importance',
ascending=False)
# Plot
plt.subplots(figsize=(10,10))
sns.barplot(x=feature_imp['feature importance'], y=feature_imp.index)
plt.title('Feature Importance')
plt.show()
```

The final technical step of the model implementation will be hyperparameter tuning. This step will find the best combination of parameters for the model (David, 2022). The method that will be used for finding the best combinations is by using the GridSearchCV algorithm. This algorithm will take a model with a dictionary of selected parameters and try every combination and the parameters with the best score will be used as the final algorithm. The parameters that will be used in this step are the:

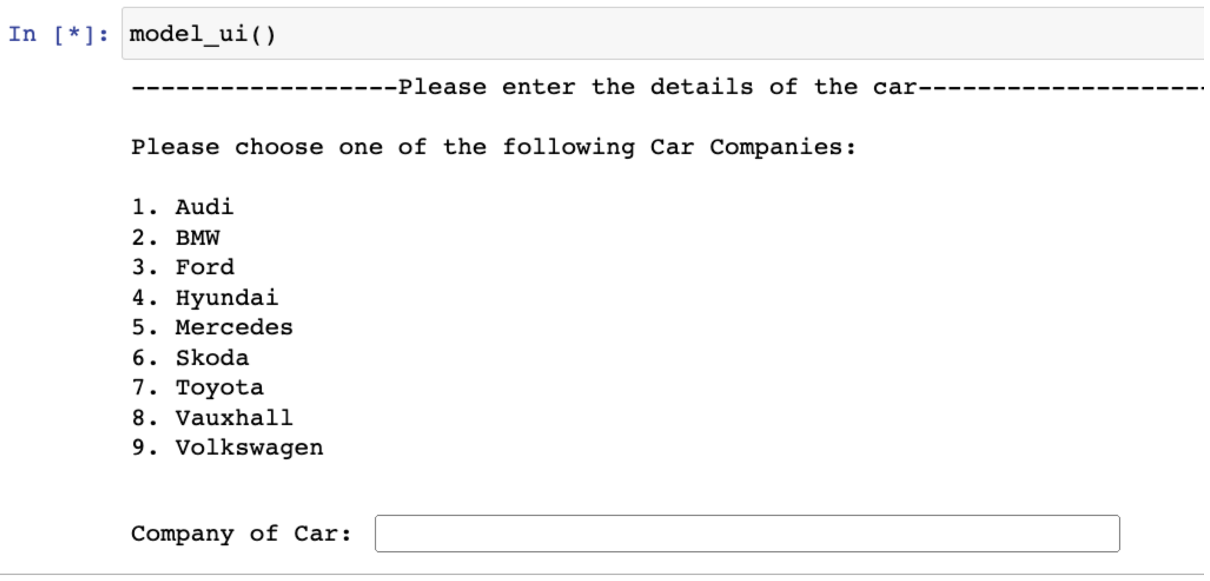
- Learning rate
- Depth
- Iterations

It is decided to only use three parameters to save time and computing power. The learning rate controls how much to change the model in response to the estimated error each time the model weights are updated (Brownlee, 2019), while the depth is the size of the tree, and the iterations indicate the number of times the algorithm's parameters are updated (Gaillard, F., Murphy, A. Iteration (machine learning)). The code on the next page demonstrates how hyperparameter tuning is performed:

```
# Gridsearch to get best parameters
parameters = {
'learning_rate': [0.33, 1],
'depth': [6, 8],
'iterations': [100, 500, 1250]
}

# Optimizing model using GridSearchCV
grid = GridSearchCV(model, parameters, cv=kf)
print('Parameter tuning results')
evaluate_model(grid, X, Y, grid=True) # Print best parameters as well
```

3.3.6 UI Form with Model

The model now is ready to create accurate predictions, however, there is no way to interact with the model with inputs in a user-friendly manner. To do this, inputs will have to be taken from a user, this will be done with the help of Python's built-in input() and print() functions. A function called model_ui() will be made that prints to the user to input their car features, and from these inputs, the predicted price will be displayed. Figure 3-8 shows the first feature to input (company). Each input will then be taken as a variable, then merged with the original dataset to retrieve the inputs with the encoded values for its categorical columns. The price for this row will then be predicted by the final model, and the user will be represented with the price. The code for the function and examples of outputs can be found in the Appendix section.



```
In [*]: model_ui()

        ------------------Please enter the details of the car--------------------

        Please choose one of the following Car Companies:

        1. Audi
        2. BMW
        3. Ford
        4. Hyundai
        5. Mercedes
        6. Skoda
        7. Toyota
        8. Vauxhall
        9. Volkswagen

        Company of Car:
```

Figure 3-8: UI Form

3.4 Experiments design and setup

To finalize this section, the next page(s) will contain the experiments and tests performed on the model to ensure the algorithm and form are working efficiently.

- The first experiment is to select the best algorithm to use before tuning, with the help of the evaluate_model() function made previously. The results of the three algorithms will then be compared and evaluated.

- The second experiment is going to be selecting the best features to use for the final model, this will simply test to see if a model that does not use features with very low importance can obtain a high or at least the same accuracy as the original model. If the model required all features to remain more accurate, then all features will be selected. This will be tested with the following code:

```
x_fs = x.drop(columns=['ColumnName']).values
evaluate_model(model, x_fs, Y, return_cvscore=True)
```

- The third experiment is selecting the best parameters to use with the GridSearchCV algorithm. Table 3-2 will be the list of parameters to be compared:

Parameter	Values
learning_rate	0.33, 1
depth	6, 8
iterations	100, 500, 1250

Table 3-2: Parameters and values to be tested

The values will be compared, and the algorithm with the new parameters will use the evaluate_model() function to compare the results with the original parameters.

- The fourth and last experiment will be to test the model using the form to see if the prices are appropriate. This will be helped with the model_ui() function created. Table 3-3 (on the next page) will show the tests that are going to be conducted:

	Test Description			
Test Case	Feature	Category 1	Category 2	Expected Outcome
1	Company	Mercedes	Skoda	Case 1 Price > Case 2 Price
2	Mileage	100	25,000	Case 1 Price > Case 2 Price
3	Year	2020	2010	Case 1 Price > Case 2 Price

Table 3-3: Experiment test cases

The other features aside from the test cases will have the same value, considering only one feature is impacted. The description of the test cases is as follows:

- Test Case 1: It is known that Mercedes cars are pricy while Skoda cars are not, therefore the category with the pricier brand must have a higher price.
- Test Case 2: If the car has higher mileage, it is more likely to be less valuable, therefore the category with the lower mileage should have a reasonably higher price for the test to be successful.
- Test Case 3: The newer the car make, the more valuable it should be, therefore the category with the newer car must have a higher price for the test to be successful.

3.5 Summary

To conclude, this chapter has given detailed steps of the solution in implementing machine learning algorithms to predict the price of cars. This algorithm has been optimized, and the setup of the experiments has been made to evaluate the model to obtain the results, which will be presented in the next chapter.

4. Results and Testing

4.1 Experiment 1 - Comparison of Algorithms

Of the three machine learning algorithms, Catboost is the best and most accurate algorithm, however, it is also the algorithm with the longest training time. The decision tree is slightly less accurate than the Catboost, but it is much faster. The linear regression algorithm is the least accurate, with predicted prices reaching negative values. Table 4-1 is a summary of the performances of the algorithms:

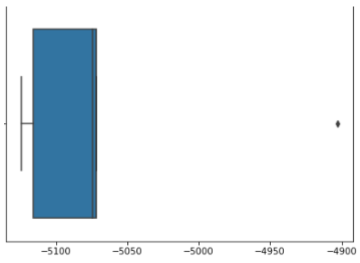
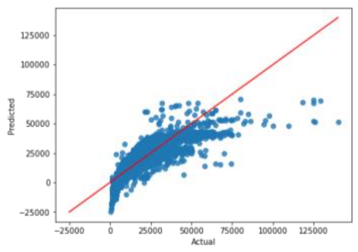
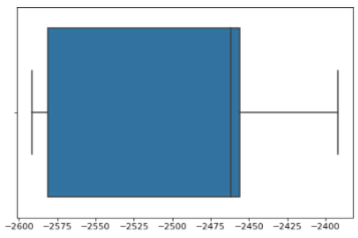
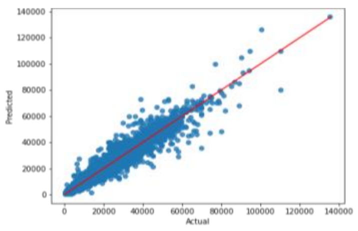
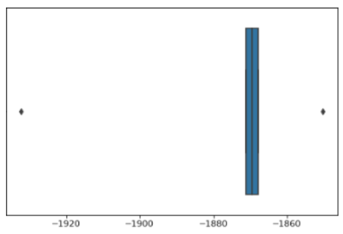
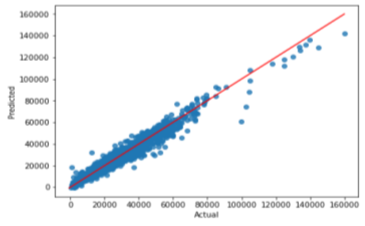
	Metric				
Regressor	**R2**	**RMSE**	**Boxplot of CV Results**	**Actual vs Predicted**	**Fit Time**
Linear	73.84%	5055.05			0.02s
Decision Tree	93.79%	2526.11			0.34s
Cat Boost	96.62%	1882.96			6.32s

Table 4-1: Comparison of Algorithms

4.2 Experiment 2 - Feature Selection

In Figure 4-1, the feature importance of the Catboost model is shown. Engine size is the most important feature, contributing to 35% of the prediction. The fuel types and tax are of very little significance, only contributing less than 2% combined.

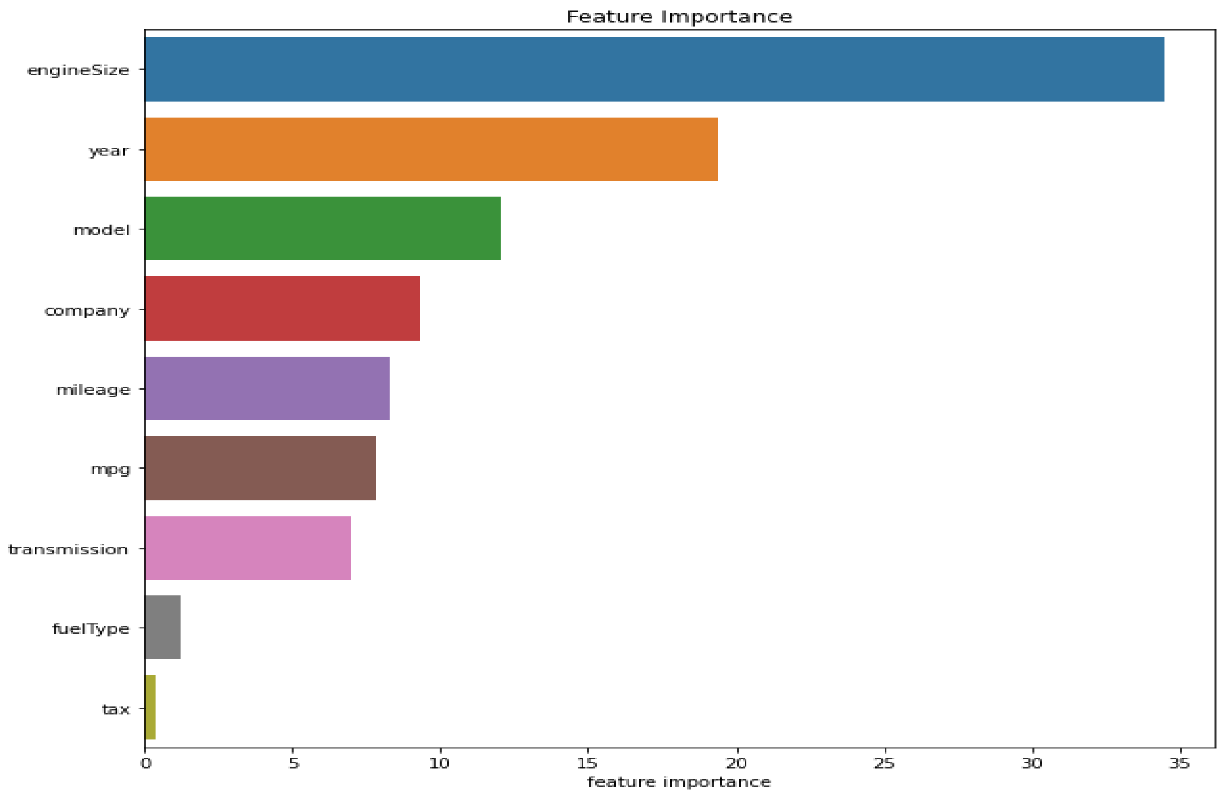


Figure 4-1: Bar chart of feature importance

By running the code provided that checks the RMSE CV score without the specified features, table 4-2 will display a summary of the results (Catboost algorithm):

Feature (removed)	**Score of Model (RMSE CV)**
Tax	1848.02
Tax & Fuel Type	1874.99
Fuel Type	1902.93

Table 4-2: Score of models by removing certain feature(s)

The original model scored an RMSE score of 1882.96, while the highest score when selecting features (tax) is 1848.02. Therefore, only the Tax column will be removed.

4.3 Experiment 3 - Hyper-parameter Tuning

Table 4-3 represents the scores obtained by using the new parameters. The RMSE CV mean shows that the error rate has decreased by 8.27% from the original model. This shows that using hyperparameter tuning would help predict slightly more accurate prices. The R2 score shows the final model is 97.18% accurate. Table 4-4 shows the parameters with the best combinations using the Catboost model. In Figure 4-2, the scatter graph visually shows that the results are more correlated. The boxplot shows that the RMSE score is usually between around -1600 to -1750, however an outlier score of -1950 appeared once.

Metric	**Score**	**% Increase from original model**
R2	97.24%	0.64%
RMSE CV Mean	1678.97	-10.35%

Table 4-3: Scores of optimized model

Parameter	**Value**
Depth	8
Iterations	1250
Learning rate	0.33

Table 4-4: Best model parameters

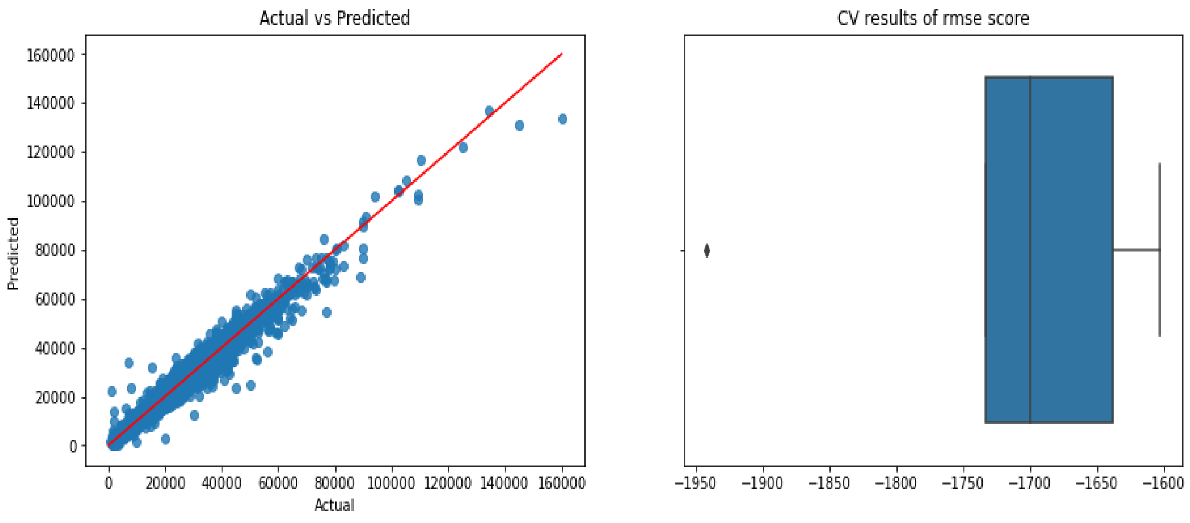


Figure 4-2: Plots of Actual vs Predicted and CV Results of RMSE score

4.4 Experiment 4 – Test Cases

The full cases can be found in the Appendix.

4.4.1 Test Case 1 – Mercedes and Skoda Price

For this test case, the Mercedes C Class, and the Skoda Octavia's predicted prices are going to be compared. For this test case to be successful, the Mercedes car must be higher than the Skoda car. The test has been successful with the Mercedes C Class price being £18557, while the Skoda Octavia price is £12199. Both inputs used the same features.

4.4.2 Test Case 2 – Mileage of 100 and 25,000

This case is going to be like case 1. In this case, the Audi A1 car price will be predicted with a mileage of 100m and 25,000m. To be successful, the price of the car with a mileage of 100 must the higher price. The test has been successful with the price with a mileage of 100 is £21109, while the latter is only £11861.

4.4.3 Test Case 3 – Year of 2020 and 2010

This case will use the same process as case 2. To be successful, the price of the 2020 car must be higher than the 2010 car price. The test has been successful with the 2020 car for £23157, while the latter for only £11810.

4.5 Summary

The results show that the best model is the catboost algorithm, which does not select the tax feature, and the best parameters for it are depth = 8, iterations = 1250, and learning rate = 0.33. The R2 score of the final model is 97.24%. These results will be analyzed and discussed in the next chapter.

5. Discussion and Analysis

5.1 Analysis of Results

The catboost model has been the most accurate model of the three. In the Literature Review section, most models of the studies have scored between 60-80%, while this study has a final model accuracy of 97%, which shows that in this study, predictions are much more reliable and accurate than in previous studies. The catboost algorithm has highly contributed to achieving these results. This algorithm has great capabilities in handling categorical data, due to its structure as being based on a gradient boosted decision trees (How training is performed, n.d.).

By looking at the table in the results of Experiment 1, the boxplot of the cross-validation results using the catboost model shows that even its higher outlier of its RMSE value is lower than the lower bound of the results using the decision tree model. However, it can be argued that using the decision tree model is the better model, as it is much faster than the catboost model by almost 20x. If it was the case that the algorithm must predict many values at a fast pace e.g., real-time predicting of large data, then using a decision tree would be the better algorithm. However, this is not the case as the algorithm would only have to make one prediction at every run, which should not be an issue, and therefore using catboost would be the superior algorithm.

But considering the speed of the Catboost algorithm, it has led to the suggestion that the categorical data of the original data must be encoded using label encoding instead of one-hot encoding, this is because if the one-hot encoding is used, this would add many more columns, resulting in a much slower algorithm, although it can be argued that using one-hot encoding could lead to more accurate results, also that there would be no need to encode the data from the start as catboost already encodes these values. However, the extra time that would need to be taken, especially considering the number of car models, would not be a reasonable approach for a little more accuracy.

5.2 Significance of the findings

The significance of these findings has shown that with this highly efficient model, accurate predictions of car prices can be obtained. This includes factors that involve depreciation, a car’s company, model, and more factors. This can benefit individuals that would like a valuation of their car, or to find estimations of the best car that can satisfy their price range. Car rental and insurance companies can use this algorithm to help aid in pricing a car or estimating premium costs. This study is also significant as the algorithm can be used by non-technical users with the help of an interactive form, as it can be assumed that a majority will not have a technical background. This tool can also help prove a car's valuation, for example, a car seller posting an ad of their car can attach their car's valuation that is predicted by the final model of this study.

5.3 Limitations

The following limitations of this study have been preserved as follows:

- It has been mentioned that the Catboost algorithm is very slow compared to the other algorithms. Considering the size of the data, if the catboost algorithm is to be optimized further with different ranges of parameters, high amounts of computing power and time will be consumed and will not be a feasible solution for individuals attempting to enhance the algorithm. If time and/or power are an issue, then it would be best to use the Decision Tree model and enhance the model in the same steps.
- Because of the limited data available from online sources, only 9 car companies can be predicted with this model. This can be an issue, especially for rental/insurance companies as it is likely that many more car companies would need valuations. This issue can be resolved by rough searching through available and ready datasets, or by requesting data from relevant car companies to expand the number of companies the model can. However, this can be possibly difficult to obtain due to legal or privacy issues.
- The dataset used does not consider a car's physical condition, as there can be salvaged/scratched parts or internal damage. It also does not consider cars that have upgrades/features to them that can increase their value. These factors can significantly impact a car's valuation. Therefore, the algorithm implemented in this study is limited to only good and normally conditioned cars. This can be solved by finding a new similar dataset that categorizes the data with a scaled list e.g., 1-Excellent, 2-Good, 3-Poor.

5.4 Summary

The discussion concludes that the catboost algorithm performs excellently despite its speed. It also satisfies the needs of individuals and car insurance/rental companies and can be enhanced even further to match specific requirements, such as adding more relevant features or enriching the UI form to be a dynamic web page.

6. Conclusions and Future Work

6.1 Conclusions

To conclude, this study has achieved the aims and objectives that include creating a program to predict the price of cars, and the simplicity of inputting car details to obtain predictions from the algorithm. The results have been highly accurate with the help of the Catboost algorithm and reliable thanks to cross-validation. This study has proven to be a significant study for companies including insurance and car rentals, and individuals. The limitations have been listed and can be avoided with additional steps or procedures if needed.

6.2 Future Work

This study can be built up further to achieve the following work:

- Can be implemented as a tool within a car-related company's database that would predict the price of their cars available to give reasonable and unbiased prices.
- The form implemented has been simple, however, the program can be implemented on a website with a better UX form, that can include drop-down lists for the car companies, images for website design, etc. If this website is online, it can allow individuals to enter details of their car and valuate their car from home.
- Can be built further to tackle the limitations and to provide more accurate and reliable results with a variety of inputs.

7. Reflection

During the summer of 2021, I decided to do an internship in a bank in the Data Mining department. This has led me to have a greater interest in the Data Science and Machine Learning field. Initially, it sounded complicated, however, it seemed exciting to learn. When I learned that the main language used is Python, it was even more exciting as I have always kept reading about how simple the Python language is compared to others. I haven’t struggled to adapt to the language and was immediately determined in enhancing my knowledge in the area.

During the Autumn Term, I have received great support from my supervisor, Dr. Carmen Lam, as she has provided guidance and clearance in understanding the requirements. I have also taken the opportunity to learn from other modules, such as Python for Data Science (CS3PP19) and AI (CS3AI18).

By the end of the Winter Break, I have completed the implementation, and literature review section. This has resulted in greater self-discipline, as I was determined to make use of the break.

During the Spring Break, I have kept revising, and coding, and have found mistakes or enhancements along the way, which has led me to learning much more in-depth knowledge of the area.

Overall, this study has given me greater skills, such as time-management, problem-solving, researching, report-writing, and project management, which is for sure to aid my future career goals.

References

1) *En.wikipedia.org. 2022. Car - Wikipedia. [online] Available at: <https://en.wikipedia.org/wiki/Car>*
2) *Darrin Qualman. 2022. Happy motoring: Global automobile production 1900 to 2016. [online] Available at: <https://www.darrinqualman.com/global-automobile-production/>*
3) *Lexico Dictionaries | English. 2022. DEPRECIATION | Meaning & Definition for UK English | Lexico.com. [online] Available at: <https://www.lexico.com/definition/depreciation>*
4) *En.wikipedia.org. 2022. Data science - Wikipedia. [online] Available at: <https://en.wikipedia.org/wiki/Data_science>*
5) *Cuvva.com. 2022. [online] Available at: <https://www.cuvva.com/how-insurance-works/vehicle-value-and-price>*
6) *Sas.com. 2022. Predictive Analytics: What it is and why it matters. [online] Available at: <https://www.sas.com/en_us/insights/analytics/predictive-analytics.html#:~:text=Predictive%20analytics%20is%20the%20use,will%20happen%20in%20the%20future.>*
7) *Investopedia. 2022. Predictive Analytics Definition. [online] Available at: <https://www.investopedia.com/terms/p/predictive-analytics.asp>*
8) *Sas.com. 2022. Predictive modeling, analytics, and machine learning. [online] Available at: <https://www.sas.com/en_gb/insights/articles/analytics/a-guide-to-predictive-analytics-and-machine-learning.html>*
9) *Medium. 2022. Linear Regression: Real-life example. [online] Available at: <https://medium.com/@kumarvaishnav17/linear-regression-real-life-example-3e23cd5e47ab>*
10) *Dwivedi, R., 2022. How Does Linear And Logistic Regression Work In Machine Learning? | Analytics Steps. [online] Analyticssteps.com. Available at: <https://www.analyticssteps.com/blogs/how-does-linear-and-logistic-regression-work-machine-learning>*
11) *Oleszak, M., 2019. Regularization Tutorial: Ridge, Lasso and Elastic Net. [online] Datacamp.com. Available at: <https://www.datacamp.com/community/tutorials/tutorial-ridge-lasso-elastic-net>*

12) HackerEarth. 2022. Decision Tree Tutorials & Notes | Machine Learning | HackerEarth. [online] Available at: <https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/ml-decision-tree/tutorial/>
13) Abolfazl Ravanshad. (2018, April 27). Gradient Boosting vs Random Forest. Medium; Medium. https://medium.com/@aravanshad/gradient-boosting-versus-random-forest-cfa3fa8f0d80
14) 6 factors that influence a home's value. (n.d.). Inman. https://www.inman.com/2017/08/07/6-factors-that-influence-a-homes-value/
15) Wohl, K. (n.d.). The benefits of knowing your home's real value [Review of The benefits of knowing your home's real value]. EdmondLifeAndLeisure.com. https://edmondlifeandleisure.com/the-benefits-of-knowing-your-homes-real-value-p13545-84.htm
16) Et. al., J. A. (2021). Prediction of House Price Using XGBoost Regression Algorithm. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(2). https://doi.org/10.17762/turcomat.v12i2.1870
17) J. Manasa, R. Gupta and N. S. Narahari, "Machine Learning based Predicting House Prices using Regression Techniques," 2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), 2020, pp. 624-630, doi: 10.1109/ICIMIA48430.2020.9074952.
18) Anaplan. (2018, October 8). Sales Forecasting Guide (Fundamentals, Process, Tips & More). Anaplan. https://www.anaplan.com/blog/sales-forecasting-guide/
19) TEAM, M. T. (2020, April 4). Factors Affecting Sales Forecasting/Marketing Management. MBA TUTS. https://www.mbatuts.com/factors-affecting-sales-forecasting/
20) C. Zhan, J. Li, W. Jiang, W. Sha and Y. Guo, "E-commerce Sales Forecast Based on Ensemble Learning," 2020 IEEE International Symposium on Product Compliance Engineering-Asia (ISPCE-CN), 2020, pp. 1-5, doi: 10.1109/ISPCE-CN51288.2020.9321858.
21) Smirnov, P. S., & Sudakov, V. A. (2021). Forecasting new product demand using machine learning. In Journal of Physics: Conference Series (Vol. 1925). IOP Publishing Ltd.
22) Skrebiec, C. (2013, December 9). Prezi.com. https://prezi.com/l4v1veelkvuo/factors-that-affect-prices-of-laptops/

23) *Guozhen, A. (2021, June 30). Predicting Laptop Prices Using ML | By Andy Guozhen | Analytics Vidhya | Medium. Medium. https://medium.com/analytics-vidhya/predicting-laptop-prices-using-ml-e60a0315b45a*

24) *Kagan, J. (n.d.). Actual Cash Value. Investopedia. https://www.investopedia.com/terms/a/actual-cash-value.asp*

25) *Actual Cash Value Calculator. (n.d.). Miniwebtool.com. Retrieved from https://miniwebtool.com/actual-cash-value-calculator/*

26) *Brozic, J. (2021, November 9). Actual Cash Value: How it Works for Car Insurance [Review of Actual Cash Value: How it Works for Car Insurance]. Kelly Blue Book. https://www.kbb.com/car-advice/insurance/actual-cash-value/*

27) *Amik, F.R.; Lanard, A.; Ismat, A.; Momen, S. Application of Machine Learning Techniques to Predict the Price of Pre-Owned Cars in Bangladesh. Information 2021, 12, 514. https://doi.org/10.3390/ info12120514*

28) *Stack Overflow Developer Survey 2021. (n.d.). Stack Overflow. https://insights.stackoverflow.com/survey/2021#most-popular-technologies-language*

29) *Top 5 Programming Languages and their Libraries for Machine Learning in 2020. (2020, June 26). GeeksforGeeks. https://www.geeksforgeeks.org/top-5-programming-languages-and-their-libraries-for-machine-learning-in-2020/*

30) *Medium. 2022. Boost your model's performance with these fantastic libraries. [online] Available at: <https://towardsdatascience.com/boost-your-models-performance-with-these-fantastic-libraries-8dc10579b7ff#:~:text=We%20need%20to%20apply%20feature%20scaling%20only%20for%20the%20Decision,not%20for%20XGBoost%20and%20CatBoost.&text=Import%20the%20DecisionTreeClassifier%20class%20and%20then%20create%20an%20instance%20of%20the%20class.>*

31) *David, D., 2022. Hyperparameter Optimization Techniques to Improve Your Machine Learning Model's Performance. [online] freeCodeCamp.org. Available at: <https://www.freecodecamp.org/news/hyperparameter-optimization-techniques-machine-learning/>*

32) *Brownlee, J., 2019. Understand the Impact of Learning Rate on Neural Network Performance. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/>*

33) *Catboost.ai. n.d., How training is performed. [online] Available at: <https://catboost.ai/en/docs/concepts/algorithm-main-stages>*

Appendix – Jupyter Notebook

Car Price Predictor

Importing Libraries

In [1]:

```
# For data manipulation
import pandas as pd
import numpy as np
import time

# For plotting
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

# For encoding categorical data
from sklearn.preprocessing import LabelEncoder
# Cross validation and hyperparameter tuning
from sklearn.model_selection import cross_val_score, KFold, GridSearchCV
# Machine Learning models
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from catboost import CatBoostRegressor
# Metrics for evaluation
from sklearn.metrics import mean_squared_error, r2_score
```

Importing and Cleaning Data

In [2]:

```
# Gathering the Data
audi = pd.read_csv('audi.csv')
bmw = pd.read_csv('bmw.csv')
ford = pd.read_csv('ford.csv')
hyundi = pd.read_csv('hyundi.csv')
merc = pd.read_csv('merc.csv')
skoda = pd.read_csv('skoda.csv')
toyota = pd.read_csv('toyota.csv')
vauxhall = pd.read_csv('vauxhall.csv')
vw = pd.read_csv('vw.csv')

# Adding the company columns to the data frames
audi['company'] = 'Audi'
bmw['company'] = 'BMW'
ford['company'] = 'Ford'
hyundi['company'] = 'Hyundai'
merc['company'] = 'Mercedes'
skoda['company'] = 'Skoda'
toyota['company'] = 'Toyota'
vauxhall['company'] = 'Vauxhall'
vw['company'] = 'Volkswagen'

# Append all the data into one dataframe
df = audi.append([bmw, ford, hyundi, merc, skoda, toyota, vauxhall, vw],
sort=False)
```

```
# Move the company column to the left
df = df[ ['company'] + [ col for col in df.columns if col != 'company' ] ]
# Move the price column to the right
df = df[ [ col for col in df.columns if col != 'price' ] + ['price'] ]

# Removing empty spaces
for i in df.columns:
    if df[i].dtypes == 'O':
        df[i] = df[i].str.strip()

df.head()
```

Out[2]:

	company	model	year	transmission	mileage	fuelType	tax	mpg	engineSize	tax(£)	price
0	Audi	A1	2017	Manual	15735	Petrol	150.0	55.4	1.4	NaN	12500
1	Audi	A6	2016	Automatic	36203	Diesel	20.0	64.2	2.0	NaN	16500
2	Audi	A1	2016	Manual	29946	Petrol	30.0	55.4	1.4	NaN	11000
3	Audi	A4	2017	Automatic	25952	Diesel	145.0	67.3	2.0	NaN	16800
4	Audi	A3	2019	Manual	1998	Petrol	145.0	49.6	1.0	NaN	17300

In [3]:

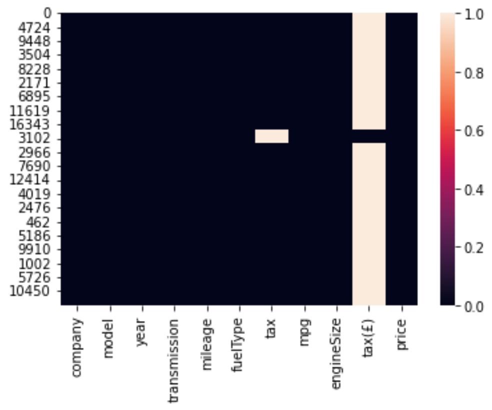
```
# Checking for any null values
df.isnull().sum()
```

Out[3]:

```
company            0
model              0
year               0
transmission       0
mileage            0
fuelType           0
tax             4860
mpg                0
engineSize         0
tax(£)         94327
price              0
dtype: int64
```

In [4]:

```
# Heatmap of null values
sns.heatmap(df.isnull())
plt.show()
```



In [5]:

```
# Filling null values of tax column with tax(£) column
df['tax'] = df['tax'].fillna(df['tax(£)'])
# Dropping not needed column
df = df.drop(columns=['tax(£)'])
```

In [6]:

```
# Getting data statistics
df.describe()
```

Out[6]:

	year	mileage	tax	mpg	engineSize	price
count	99187.000000	99187.000000	99187.000000	99187.000000	99187.000000	99187.000000
mean	2017.087723	23058.914213	120.299838	55.166825	1.663280	16805.347656
std	2.123934	21148.523721	63.150926	16.138522	0.557646	9866.773417
min	1970.000000	1.000000	0.000000	0.300000	0.000000	450.000000
25%	2016.000000	7425.000000	125.000000	47.100000	1.200000	9999.000000
50%	2017.000000	17460.000000	145.000000	54.300000	1.600000	14495.000000
75%	2019.000000	32339.000000	145.000000	62.800000	2.000000	20870.000000
max	2060.000000	323000.000000	580.000000	470.800000	6.600000	159999.000000

In [7]:

```
# Removing 2060 year row
df = df.loc[df['year'] <= 2020]
```

In [8]:

```
# Ensuring engine size of zero are only electric
df.loc[df['engineSize'] == 0]['fuelType'].describe()
```

Out[8]:

```
count         273
unique          5
```

```
top       Petrol
freq         163
Name: fuelType, dtype: object
```

In [9]:

```
# It is clear that the majority of cars with an engine size of zero are not
electric, therefore will be removed
df = df.loc[df['engineSize'] != 0]
```

In [10]:

```
# Checking for electric cars
df.loc[df['fuelType'] == 'Electric']
```

Out[10]:

	company	model	year	transmission	mileage	fuelType	tax	mpg	engineSize	price
8835	BMW	i3	2015	Automatic	29465	Electric	0.0	470.8	1.0	17400
6385	Ford	Mondeo	2016	Automatic	9396	Electric	0.0	67.3	2.0	15975
11959	Ford	Mondeo	2016	Automatic	24531	Electric	0.0	67.3	2.0	15500
13317	Vauxhall	Ampera	2015	Automatic	34461	Electric	0.0	235.4	1.4	12999

In [11]:

```
# It would be best to remove the electrical cars
df = df.loc[df['fuelType'] != 'Electric']
```

In [12]:

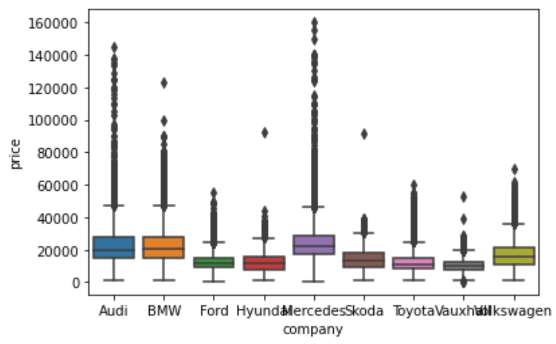
```
# Ensuring the data types are correct
df.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 98909 entries, 0 to 15156
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   company       98909 non-null  object 
 1   model         98909 non-null  object 
 2   year          98909 non-null  int64  
 3   transmission  98909 non-null  object 
 4   mileage       98909 non-null  int64  
 5   fuelType      98909 non-null  object 
 6   tax           98909 non-null  float64
 7   mpg           98909 non-null  float64
 8   engineSize    98909 non-null  float64
 9   price         98909 non-null  int64  
dtypes: float64(3), int64(3), object(4)
memory usage: 8.3+ MB
```

Exploratory Data Analysis (EDA)

Company and Price

In [13]:

```
sns.boxplot(data=df, x='company', y='price')
plt.show()
```



In [14]:

```
# Length before removing outliers
len(df)
```

Out[14]:

```
98909
```

In [15]:

```
company_outliers ={ # Manually listing the outliers
    'BMW' : 100000,
    'Hyundai' : 60000,
    'Skoda' : 60000,
    'Vauxhall' : 30000,
}
# Removing outliers based on company_outliers
for company in list(company_outliers):
    value = company_outliers[company]
    df = df.loc[~((df['price'] > value) & (df['company'] == company))]
# Length after removing outliers
len(df)
```

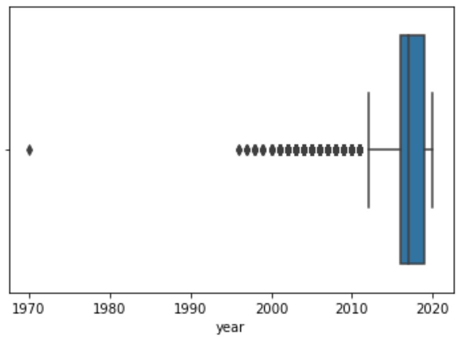
Out[15]:

```
98904
```

Year and Price

In [16]:

```
sns.boxplot(x=df['year'])
plt.show()
```



In [17]:

```
df = df.loc[df['year'] > 1970]
len(df)
```

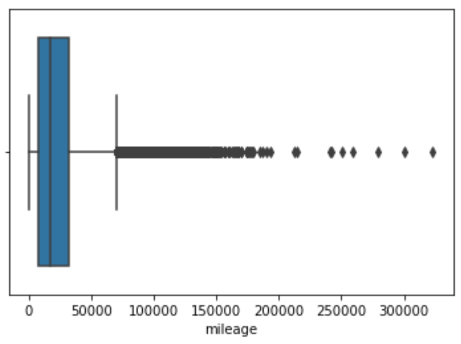
Out[17]:

```
98903
```

Mileage and Price

In [18]:

```
sns.boxplot(x=df['mileage'])
plt.show()
```



In [19]:

```
df = df.loc[df['mileage'] < 200000]
len(df)
```

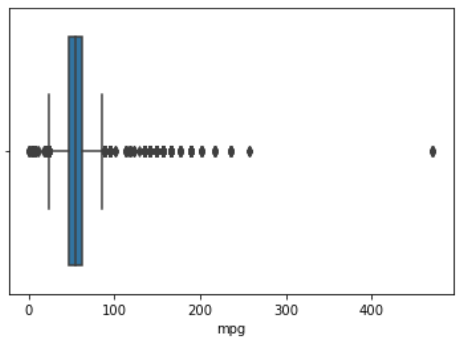
Out[19]:

```
98894
```

Miles per Gallon and Price

In [20]:

```
sns.boxplot(x=df['mpg'])
plt.show()
```



In [21]:

```
df = df.loc[df['mpg'] < 300]
len(df)
```

Out[21]:

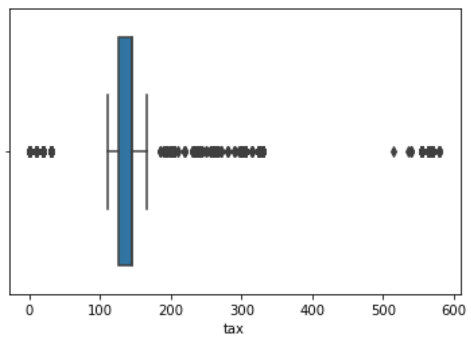
```
98887
```

The chart shows that there is an increase in price the less the miles per gallon is.

Tax and Price

In [22]:

```
sns.boxplot(x=df['tax'])
plt.show()
```

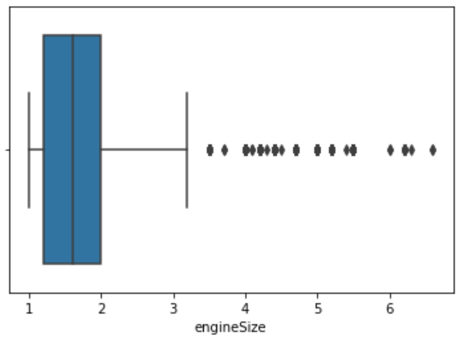


There seems to be no correlation between the price and tax. Therefore, we can remove this column.

Engine Size and Price

In [23]:

```
sns.boxplot(df['engineSize'])
plt.show()
/Users/farishanna/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py
:36: FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



There is not much correlation between the engine size and the price. Only that usually more expensive cars can have a large engine size.

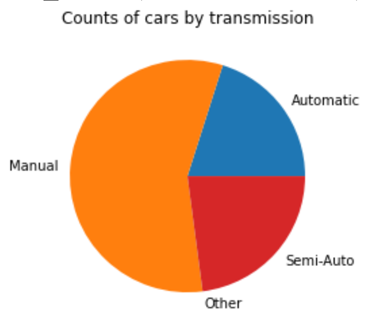
Transmission and Price

In [24]:

```
def cat_clean(column):
    # Creates pivot table of the mean of prices of each transmission type
    table = pd.pivot_table(df, values='price', index=[column], aggfunc='count')
    # Creating pie chart
    plt.pie(table.price, labels=table.index)
    # Adding title
    plt.title('Counts of cars by ' + column)
    # Showing graph
    plt.show()
```

In [25]:

```
cat_clean('transmission')
```



In [26]:

```
df = df.loc[df['transmission'] != 'Other']
len(df)
```

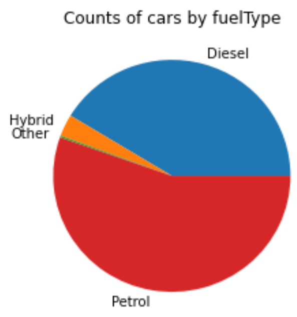
Out[26]:

```
98879
```

Fuel Type and Price

In [27]:

```
cat_clean('fuelType')
```



In [28]:

```
df = df.loc[df['fuelType'] != 'Other']
len(df)
```

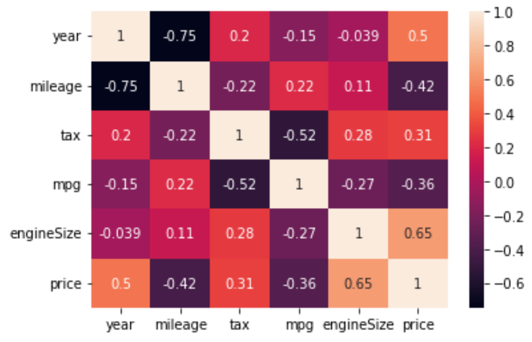
Out[28]:

```
98640
```

Further analysis

In [29]:

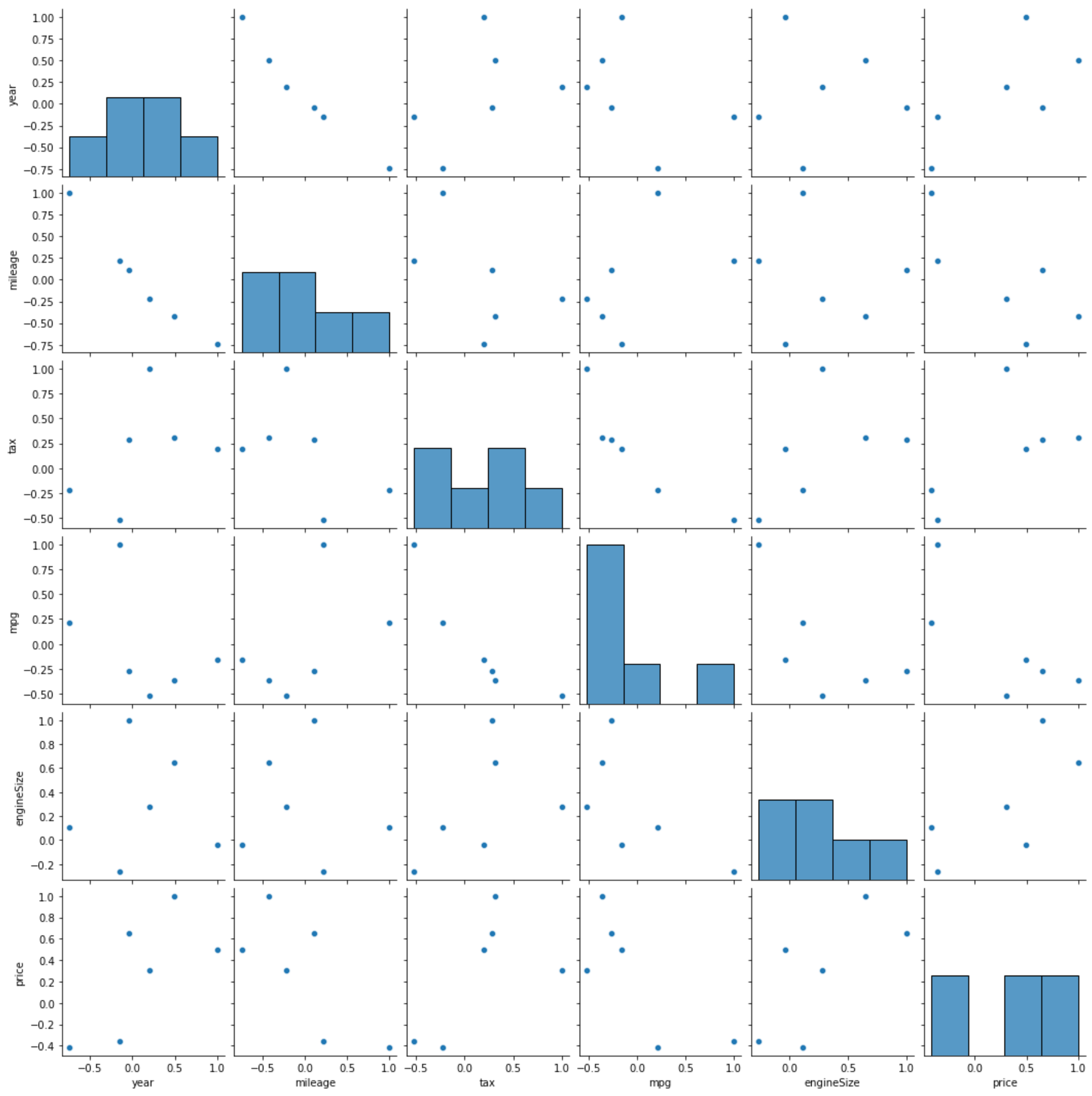
```
# Checking for any correlations in the dataframe using a heatmap
sns.heatmap(df.corr(), annot=True)
plt.show()
```



	year	mileage	tax	mpg	engineSize	price
year	1	-0.75	0.2	-0.15	-0.039	0.5
mileage	-0.75	1	-0.22	0.22	0.11	-0.42
tax	0.2	-0.22	1	-0.52	0.28	0.31
mpg	-0.15	0.22	-0.52	1	-0.27	-0.36
engineSize	-0.039	0.11	0.28	-0.27	1	0.65
price	0.5	-0.42	0.31	-0.36	0.65	1

In [30]:

```
sns.pairplot(df.corr())
plt.show()
```



Data Pre-Processing

In [31]:

```
# # Splitting target variable (price) and independent variables
X = df.drop(columns=['price'])
Y = df['price'].values
```

In [32]:

```
# Encoding the categorical data into integers
le = LabelEncoder()
def encode(df_le):
    df_le['company'] = le.fit_transform(df_le['company'])
    df_le['model'] = le.fit_transform(df_le['model'])
    df_le['transmission'] = le.fit_transform(df_le['transmission'])
    df_le['fuelType'] = le.fit_transform(df_le['fuelType'])
    return df_le

X = encode(X)
X.head()
```

Out[32]:

	company	model	year	transmission	mileage	fuelType	tax	mpg	engineSize
0	0	12	2017	1	15735	2	150.0	55.4	1.4
1	0	17	2016	0	36203	0	20.0	64.2	2.0
2	0	12	2016	1	29946	2	30.0	55.4	1.4
3	0	15	2017	0	25952	0	145.0	67.3	2.0
4	0	14	2019	1	1998	2	145.0	49.6	1.0

In [33]:

```
# K fold splitting training and testing data
kf = KFold(n_splits=5, shuffle=True)
# Turning features to array
X=X.values
```

Model Implementation

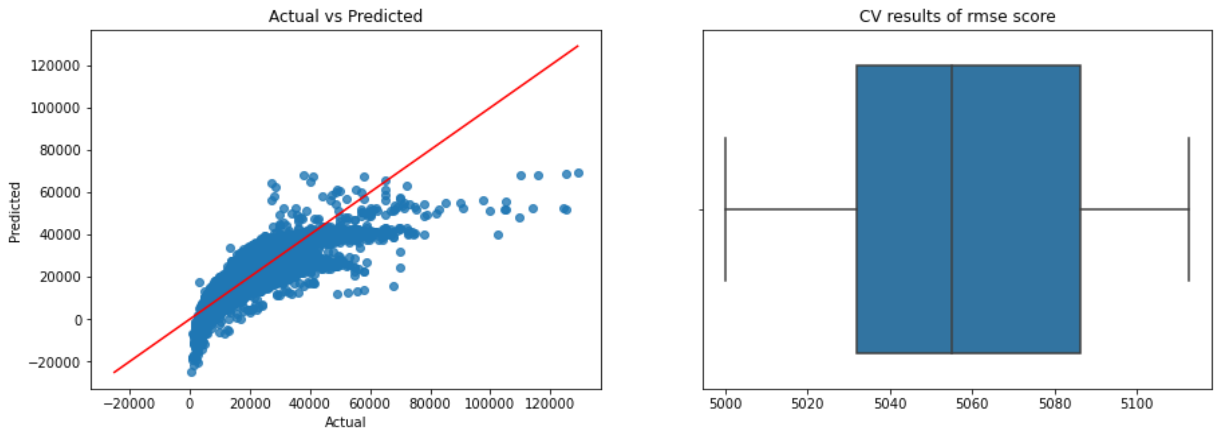
In [34]:

```
# Function that will score models' scores in the dictionary, and will plot the
models' accuracy
def evaluate_model(model, x, y, return_r2=False, return_cvscore=False,
grid=False):

    # K fold splitting training and testing data
    for train, test in kf.split(x):
        X_train, X_test, y_train, y_test = x[train], x[test], y[train], y[test]

    # Timing and fitting the training data
    start_time = time.time()
    # Fit the model using the training data
    model.fit(X_train, y_train)
    time_taken = time.time() - start_time
    # Predict the test data
    y_pred = model.predict(X_test)

    # Model metrics
    r2 = str(round(r2_score(y_test, y_pred)*100,2))+'%'
    # Cross Validation
    cvrmse = -1*cross_val_score(model,x, y,cv=kf,
                         scoring='neg_root_mean_squared_error')
    cvscore = round(np.median(cvrmse),2)
    # Return rmse when specified
    if(return_cvscore == True):
        return cvscore
    # Displaying scores
    print(f"R2 SCORE: {r2}\t",
          f"CV (RMSE) Score: {cvscore}\t",
          f"\tTime taken in seconds: {round((time_taken),2)}\n")
    # Print best params if gridsearch is used
    if(grid==True):
        print('Best Parameters: ', model.best_params_, '\n')

    # Plotting regression and boxplot
    fig, ax = plt.subplots(1,2, figsize=(15,5))
    # Plotting chart
    results = pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})
    lm=sns.regplot(x='Actual',y='Predicted',data=results, fit_reg=False, ax=ax[0])
    # Plotting line
    line = np.arange(results.min().min(), results.max().max())
    ax[0].plot(line, line, color='red')
    ax[0].set_title('Actual vs Predicted')
    # Cross validation results in boxplot (rmse)
    sns.boxplot(x=cvrmse, ax=ax[1])
    ax[1].set_title('CV results of rmse score')
    plt.show()
```

Linear Regression

In [35]:

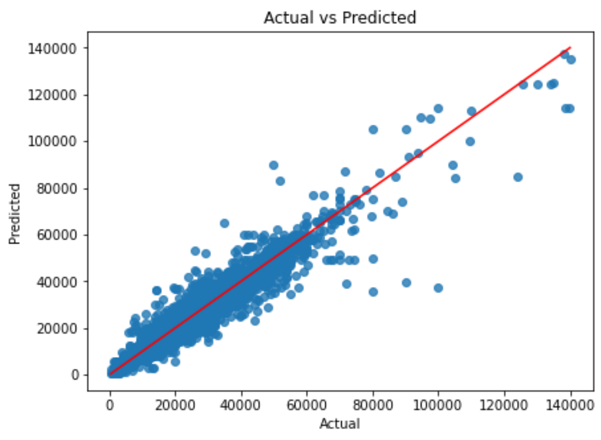
```
model = LinearRegression()
print('Linear Regression')
evaluate_model(model, X, Y)
Linear Regression
R2 SCORE: 73.84%     CV (RMSE) Score: 5055.05        Time taken in seconds: 0.02
```

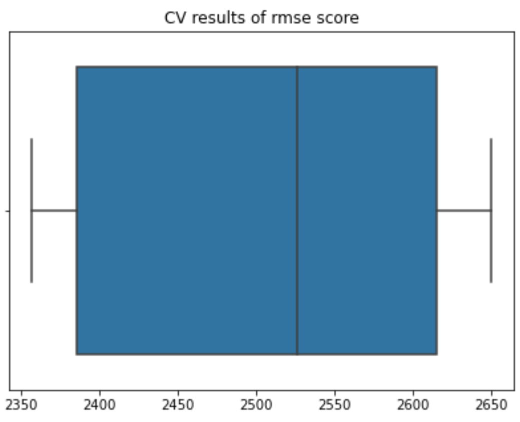


Decision Tree Regression

In [36]:

```
model = DecisionTreeRegressor()
print('Decision Tree Regression')
evaluate_model(model, X, Y)
Decision Tree Regression
R2 SCORE: 93.79%     CV (RMSE) Score: 2526.11        Time taken in seconds: 0.34
```

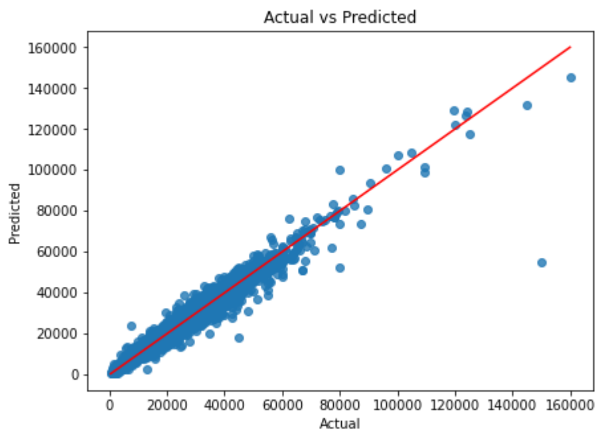


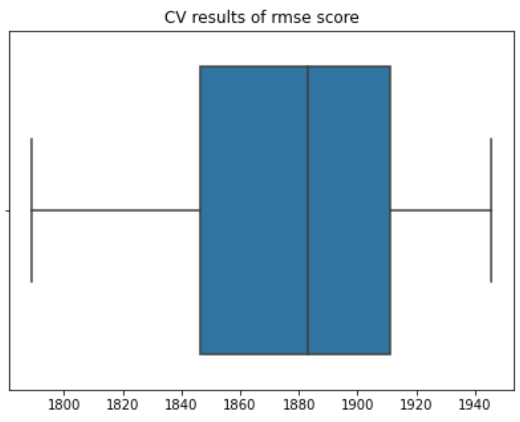


CatBoost Regressor

In [37]:

```
model = CatBoostRegressor(silent=True)
print('CatBoost Regression')
evaluate_model(model, X, Y)
CatBoost Regression
R2 SCORE: 96.22%     CV (RMSE) Score: 1882.96        Time taken in seconds: 6.32
```



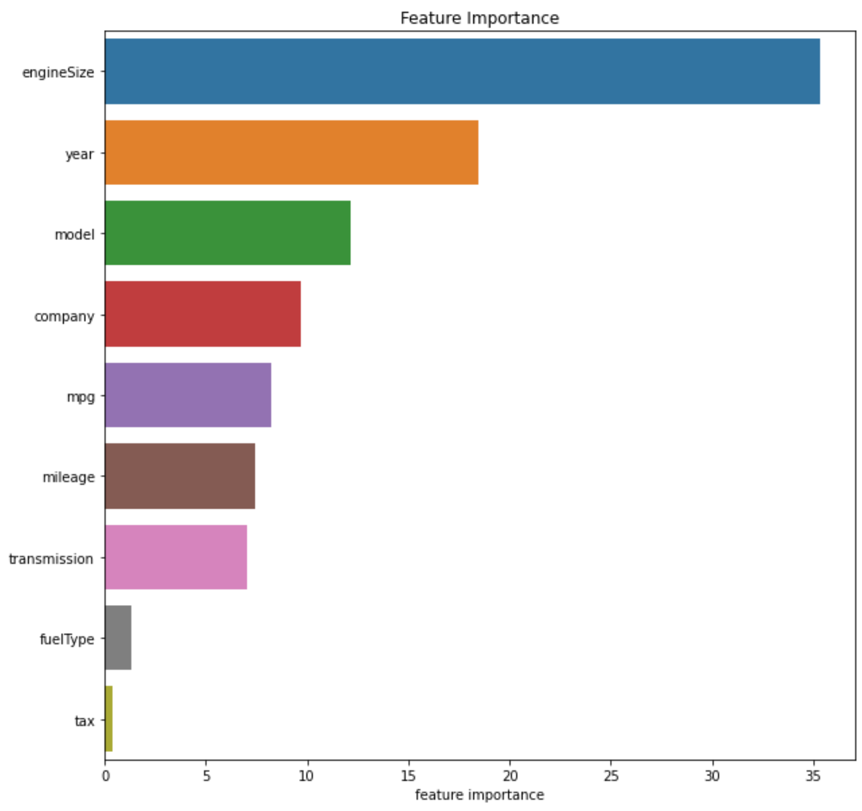


Model Tuning

Feature Selection

In [38]:

```
# Feature importance of model
feature_imp = pd.DataFrame(model.get_feature_importance())
feature_imp.index = df.drop(columns=['price']).columns
feature_imp.columns = ['feature importance']
feature_imp=feature_imp.sort_values(by='feature importance', ascending=False)
# Plot
plt.subplots(figsize=(10,10))
sns.barplot(x=feature_imp['feature importance'], y=feature_imp.index)
plt.title('Feature Importance')
plt.show()
```



In [42]:

```
x=encode(df.drop(columns=['price']))
# RMSE CV Score median without fueltype
x_fs = x.drop(columns=['fuelType']).values
ft_score = evaluate_model(model, x_fs, Y, return_cvscore=True)
print('Without fueltype:', ft_score)
# Without both fuel type and tax
x_fs = x.drop(columns=['fuelType', 'tax']).values
both_score = evaluate_model(model, x_fs, Y, return_cvscore=True)
print('Without tax & ft:', both_score)
# RMSE CV Score median without tax
x_fs = x.drop(columns=['tax']).values
tax_score = evaluate_model(model, x_fs, Y, return_cvscore=True)
print('Without tax: ', tax_score)
```

```
Without fueltype: 1902.93
Without tax & ft: 1874.99
Without tax:  1848.02
```

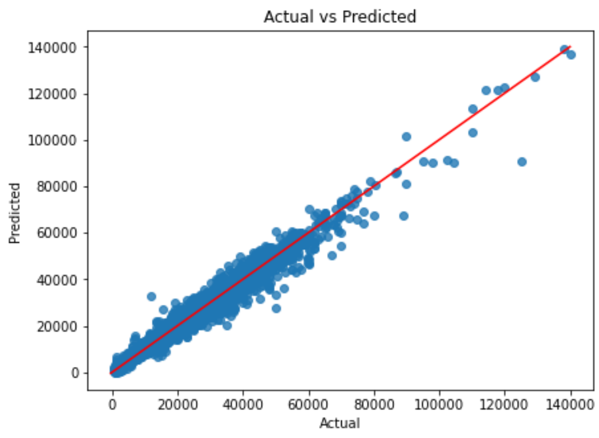
Only tax to be removed

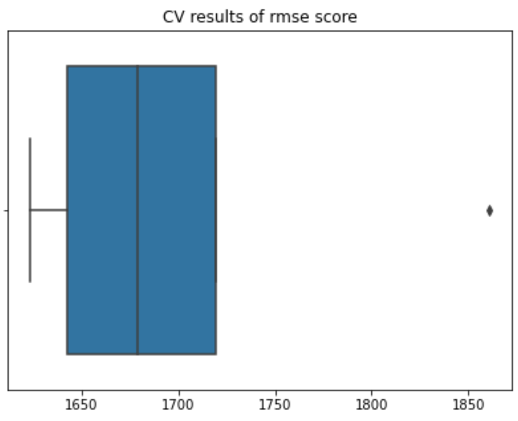
GridSearchCV

In [43]:

```
# Gridsearch to get best parameters
parameters = {
'learning_rate': [0.33,1],
'depth': [6,8],
'iterations': [250,750,1250]
} # Best params: 0.33, 8, 1250

# Optimizing pipeline using GridSearchCV
grid = GridSearchCV(model, parameters, cv=kf)
print('Parameter tuning results')
evaluate_model(grid, x_fs, Y, grid=True)
```

```
Parameter tuning results
R2 SCORE: 97.24%     CV (RMSE) Score: 1678.97       Time taken in seconds: 58.96

Best Parameters:  {'depth': 8, 'iterations': 1250, 'learning_rate': 0.33}
```





UI Form with Model

In [44]:

```
def model_ui():
    # Extracting user inputs, and changing data types when appropriate
    print("------------------Please enter the details of the car------------------
---------\n")
    print("Please choose one of the following Car Companies:\n")
    i=0
    for company in df.company.unique():
        i=i+1
        print(str(i) + '. ' + str(company)) # Displays all the car companies

    company = input("\nCompany of Car: ")
    dfi = df.loc[df['company'] == company] # Get data of specific company

    print('---------------------------------------------------------------------
-----------')
    print('\nPlease choose one of the following Car Models:\n')
    i=0
    for model in dfi.model.unique(): # Models of the specified company
        i=i+1
        print(str(i) + '. ' + model) # Display all models of company

    model = input("\nModel of Car: ") # Takes the model
    print('---------------------------------------------------------------------
-----------')
    year = int(input("\nYear of Car: ")) # Takes the year

    print('---------------------------------------------------------------------
-----------')
    print('\nPlease choose one of the following Transmissions:\n')
    i=0
    for transmission in dfi.transmission.unique(): # Transmissions of the
specified company
        i=i+1
        print(str(i) + '. ' + transmission) # Displays transmissions
    transmission = input("\nTransmission of Car: ")
    print('-----------------------------------------------------------------------
-----------')
    mileage = int(input("\nMileage driven by car: ")) # Takes mileage

    print('-----------------------------------------------------------------------
-----------')
    print('\nPlease choose one of the following Fuel Types:\n')
    i=0
    for fueltype in dfi.fuelType.unique(): # Fuel types of specified company
        i=i+1
        print(str(i) + '. ' + fueltype) # Displays fueltypes
    fuelType = input("\nFuel Type of Car: ")

    print('-----------------------------------------------------------------------
-----------')
    mpg = float(input("\nMiles per Gallon of Car: ")) # Takes car mpg

    print('-----------------------------------------------------------------------
-----------')
    engineSize = float(input("\nEngine Size of Car in Litres: ")) # Takes Engine
Size

    print('-----------------------------------------------------------------------
-----------')
    print('\nPredicting Price...') # For UI purposes

    # Turning inputs into a pandas df, to merge with the original df
    df_ui = pd.DataFrame({'company':[company],
                          'model':[model],
                          'year':[year],
                          'transmission':[transmission],
                          'mileage':[mileage],
                          'fuelType':[fuelType],
                          'tax':[0],
                          'mpg':[mpg],
                          'engineSize':[engineSize]})

    # Splitting target variable and independent variables
    X = df.drop(columns=['price'])
    Y = df['price']
    # Concatenate independent variables (X) with input
    X = pd.concat([df_ui,X], ignore_index=True)
    # Encoding data
    X = encode(X)
    # Removing tax
    X=X.drop(columns=['tax'])
    # Extracting input row with encoding
    X_pred = X.iloc[[0]]
    # Predict user input
    output = grid.predict(X_pred)
    # Display prediction
    print('\nThe predicted price of the car is worth:',
"\033[1m"+'£'+str(int(output[0]))+"\033[0m")
```

Test Cases

Test Case 1: Company

Mercedes

In [45]:

```
model_ui()
------------------Please enter the details of the car---------------------------

Please choose one of the following Car Companies:

1. Audi
2. BMW
3. Ford
4. Hyundai
5. Mercedes
6. Skoda
7. Toyota
8. Vauxhall
9. Volkswagen

Company of Car: Mercedes
----------------------------------------------------------------------------------

Please choose one of the following Car Models:

1. SLK
2. S Class
3. SL CLASS
4. G Class
5. GLE Class
6. GLA Class
7. A Class
8. B Class
9. GLC Class
10. C Class
11. E Class
12. GL Class
13. CLS Class
14. CLC Class
15. CLA Class
16. V Class
17. M Class
18. CL Class
19. GLS Class
20. GLB Class
21. X-CLASS
22. 180
23. CLK
24. R Class
25. 220
26. 200

Model of Car: C Class
----------------------------------------------------------------------------------

Year of Car: 2015
----------------------------------------------------------------------------------

Please choose one of the following Transmissions:
```

```
1. Automatic
2. Manual
3. Semi-Auto

Transmission of Car: Automatic
---------------------------------------------------------------------------------

Mileage driven by car: 1000
---------------------------------------------------------------------------------

Please choose one of the following Fuel Types:

1. Petrol
2. Hybrid
3. Diesel

Fuel Type of Car: Petrol
---------------------------------------------------------------------------------

Miles per Gallon of Car: 55.4
---------------------------------------------------------------------------------

Engine Size of Car in Litres: 1.8
---------------------------------------------------------------------------------

Predicting Price...

The predicted price of the car is worth: £18557
```

Skoda

In [46]:

```
model_ui()
------------------Please enter the details of the car---------------------------

Please choose one of the following Car Companies:

1. Audi
2. BMW
3. Ford
4. Hyundai
5. Mercedes
6. Skoda
7. Toyota
8. Vauxhall
9. Volkswagen

Company of Car: Skoda
---------------------------------------------------------------------------------

Please choose one of the following Car Models:

1. Octavia
2. Citigo
3. Yeti Outdoor
4. Superb
5. Kodiaq
```

```
6. Rapid
7. Karoq
8. Fabia
9. Yeti
10. Scala
11. Roomster
12. Kamiq

Model of Car: Octavia
----------------------------------------------------------------------------------

Year of Car: 2015
----------------------------------------------------------------------------------

Please choose one of the following Transmissions:

1. Manual
2. Automatic
3. Semi-Auto

Transmission of Car: Automatic
----------------------------------------------------------------------------------

Mileage driven by car: 1000
----------------------------------------------------------------------------------

Please choose one of the following Fuel Types:

1. Petrol
2. Diesel
3. Hybrid

Fuel Type of Car: Petrol
----------------------------------------------------------------------------------

Miles per Gallon of Car: 55.4
----------------------------------------------------------------------------------

Engine Size of Car in Litres: 1.8
----------------------------------------------------------------------------------

Predicting Price...

The predicted price of the car is worth: £12199
```
Test successful

Test Case 2: Mileage

100

In [48]:

```
model_ui()
------------------Please enter the details of the car---------------------------

Please choose one of the following Car Companies:

1. Audi
2. BMW
```

```
3. Ford
4. Hyundai
5. Mercedes
6. Skoda
7. Toyota
8. Vauxhall
9. Volkswagen

Company of Car: Audi
----------------------------------------------------------------------------------

Please choose one of the following Car Models:

1. A1
2. A6
3. A4
4. A3
5. Q3
6. Q5
7. A5
8. S4
9. Q2
10. A7
11. TT
12. Q7
13. RS6
14. RS3
15. A8
16. Q8
17. RS4
18. RS5
19. R8
20. SQ5
21. S8
22. SQ7
23. S3
24. S5
25. A2
26. RS7

Model of Car: A1
----------------------------------------------------------------------------------

Year of Car: 2015
----------------------------------------------------------------------------------

Please choose one of the following Transmissions:

1. Manual
2. Automatic
3. Semi-Auto

Transmission of Car: Automatic
----------------------------------------------------------------------------------

Mileage driven by car: 100
----------------------------------------------------------------------------------
```

```
Please choose one of the following Fuel Types:

1. Petrol
2. Diesel
3. Hybrid

Fuel Type of Car: Petrol
-----------------------------------------------------------------------------------

Miles per Gallon of Car: 55.4
-----------------------------------------------------------------------------------

Engine Size of Car in Litres: 1.8
-----------------------------------------------------------------------------------

Predicting Price...

The predicted price of the car is worth: £21109
```
25,000

In [49]:

```
model_ui()
------------------Please enter the details of the car---------------------------

Please choose one of the following Car Companies:

1. Audi
2. BMW
3. Ford
4. Hyundai
5. Mercedes
6. Skoda
7. Toyota
8. Vauxhall
9. Volkswagen

Company of Car: Audi
-----------------------------------------------------------------------------------

Please choose one of the following Car Models:

1. A1
2. A6
3. A4
4. A3
5. Q3
6. Q5
7. A5
8. S4
9. Q2
10. A7
11. TT
12. Q7
13. RS6
14. RS3
15. A8
```

```
16. Q8
17. RS4
18. RS5
19. R8
20. SQ5
21. S8
22. SQ7
23. S3
24. S5
25. A2
26. RS7

Model of Car: A1
--------------------------------------------------------------------------------

Year of Car: 2015
--------------------------------------------------------------------------------

Please choose one of the following Transmissions:

1. Manual
2. Automatic
3. Semi-Auto

Transmission of Car: Automatic
--------------------------------------------------------------------------------

Mileage driven by car: 25000
--------------------------------------------------------------------------------

Please choose one of the following Fuel Types:

1. Petrol
2. Diesel
3. Hybrid

Fuel Type of Car: Petrol
--------------------------------------------------------------------------------

Miles per Gallon of Car: 55.4
--------------------------------------------------------------------------------

Engine Size of Car in Litres: 1.8
--------------------------------------------------------------------------------

Predicting Price...

The predicted price of the car is worth: £11861
```

Test successful

Test Case 3: Year

2020

In [50]:

```
model_ui()
------------------Please enter the details of the car---------------------------
```

```
Please choose one of the following Car Companies:

1. Audi
2. BMW
3. Ford
4. Hyundai
5. Mercedes
6. Skoda
7. Toyota
8. Vauxhall
9. Volkswagen

Company of Car: Audi
--------------------------------------------------------------------------------

Please choose one of the following Car Models:

1. A1
2. A6
3. A4
4. A3
5. Q3
6. Q5
7. A5
8. S4
9. Q2
10. A7
11. TT
12. Q7
13. RS6
14. RS3
15. A8
16. Q8
17. RS4
18. RS5
19. R8
20. SQ5
21. S8
22. SQ7
23. S3
24. S5
25. A2
26. RS7

Model of Car: A1
--------------------------------------------------------------------------------

Year of Car: 2020
--------------------------------------------------------------------------------

Please choose one of the following Transmissions:

1. Manual
2. Automatic
3. Semi-Auto

Transmission of Car: Automatic
```

```
---------------------------------------------------------------------------------

Mileage driven by car: 1000
---------------------------------------------------------------------------------

Please choose one of the following Fuel Types:

1. Petrol
2. Diesel
3. Hybrid

Fuel Type of Car: Petrol
---------------------------------------------------------------------------------

Miles per Gallon of Car: 55.4
---------------------------------------------------------------------------------

Engine Size of Car in Litres: 1.8
---------------------------------------------------------------------------------

Predicting Price...

The predicted price of the car is worth: £23157
```

2010

In [51]:

```
model_ui()
------------------Please enter the details of the car---------------------------

Please choose one of the following Car Companies:

1. Audi
2. BMW
3. Ford
4. Hyundai
5. Mercedes
6. Skoda
7. Toyota
8. Vauxhall
9. Volkswagen

Company of Car: Audi
---------------------------------------------------------------------------------

Please choose one of the following Car Models:

1. A1
2. A6
3. A4
4. A3
5. Q3
6. Q5
7. A5
8. S4
9. Q2
10. A7
11. TT
```

```
12. Q7
13. RS6
14. RS3
15. A8
16. Q8
17. RS4
18. RS5
19. R8
20. SQ5
21. S8
22. SQ7
23. S3
24. S5
25. A2
26. RS7

Model of Car: A1
---------------------------------------------------------------------------------

Year of Car: 2010
---------------------------------------------------------------------------------

Please choose one of the following Transmissions:

1. Manual
2. Automatic
3. Semi-Auto

Transmission of Car: Automatic
---------------------------------------------------------------------------------

Mileage driven by car: 1000
---------------------------------------------------------------------------------

Please choose one of the following Fuel Types:

1. Petrol
2. Diesel
3. Hybrid

Fuel Type of Car: Petrol
---------------------------------------------------------------------------------

Miles per Gallon of Car: 55.4
---------------------------------------------------------------------------------

Engine Size of Car in Litres: 1.8
---------------------------------------------------------------------------------

Predicting Price...

The predicted price of the car is worth: £11810
```
Test successful