Car Price Prediction using Regression Analysis and Machine Learning Techniques

Harshitha Poolakanda Somanna
Student ID: x22150366
Data Mining and Machine Learning 1
National College of Ireland, Dublin
x22150366@student.ncirl.ie

***Abstract* – This research utilizes two distinct machine learning modeling techniques to analyze three sizable datasets for predicting information related to the sales of used cars. The first technique, implemented using scikit-learn and XGBoost, is the Random Forest regressor. The second technique, implemented using the sklearn.ensemble and xgboost libraries, is Gradient boosting. To evaluate the performance of the models, each dataset is partitioned into separate training and testing records, and four different regression algorithms are applied. The resulting models are assessed based on their R-squared value. The findings of this study reveal that, across all three datasets, the Gradient XGBRegressor technique outperformed the other three algorithms in accurately predicting car prices. These results suggest that the Gradient XGBRegressor technique may be the most effective approach for predicting used car sales data.**

Keywords—RandomForest, XGBoost, Regression, R-Squared, accuracy, MSE, RMSE, MAE

I. Introduction

Machine learning is a subfield of artificial intelligence that involves the development of algorithms and statistical models which enable computer systems to improve their performance on specific tasks as they are exposed to more data. Its practical applications include image and speech recognition, fraud detection, recommendation systems, and autonomous vehicles. Machine learning models, trained on large datasets, can make predictions and decisions with high accuracy and efficiency, often outperforming human experts. This project employs various machine learning methods to address real-world challenges, highlighting the practical benefits of this rapidly evolving field.

In recent years, the rise in demand for used cars has been attributed to factors such as the shortage of new cars caused by the COVID-19 pandemic, inflation, changes in consumer preferences, and shifts in the economy. Accurately predicting used car prices has become crucial for both individuals and businesses in the market. Machine learning techniques can aid in predicting prices with greater accuracy, facilitating decision-making for both buyers and sellers. This research investigates the application of four different machine learning algorithms in analyzing three large datasets to predict used car sales data, highlighting the practical benefits of machine learning in solving real-world problems. The study creates models using the collected data, which are then compared and evaluated using various performance metrics to identify the optimal model.

A. Research question and approach summary.

RQ1 – *Do the different implementations of identically-named machine learning techniques perform exactly the same? If not, what are the outstanding implementations of the identically-named machine learning techniques for specific empirical designs, and evaluation measures? That is, if different implementations of the identically-named techniques perform differently, which implementation is better than the others in each dataset? This will identify the best-performing implementation of each.*

RQ2 – *How do the datasets employed in this work differ from each other? Specifically, how the three datasets employed are different from each other in terms of their characteristics which can impact the effectiveness of machine learning techniques?*

The following approach has been implemented to answer the research questions:

1. *Feature Engineering* : During the pre-processing stage for all three datasets, appropriate columns were encoded and irrelevant columns were removed. Missing values were imputed with their respective mean values or dropped when necessary.
2. *Data Cleaning:* Cars with a distance run greater than 100,000 and those older than 1970, considered vintage, were removed to ensure linearity of values for further processing. Outliers were removed using the interquartile range (IQR) method based on the size of the dataset.
3. *Transformation*: A transformation technique known as box cox transformation was applied to scale the target variable for all the datasets.
4. *Feature selection*:Feature selection was carried out by analyzing the correlation between features and the target variable, as well as identifying highly multi-correlated features using a correlation plot. The Variance Inflation Factor (VIF) was also computed to detect and eliminate any highly correlated features that may affect model performance. Only those features that did not significantly affect the model were removed.
5. *Model Building and Evaluation*: For all three datasets, random forest and gradient boosting algorithms were used to build the predictive models. The datasets were split into training and testing sets using an 70:30 ratio. The hyperparameters for the models were set for and the performance of the models was evaluated using metrics such as mean squared error (MSE), root mean squared error (RMSE), and R-squared. The models were also compared based on their

performance and the best-performing model was selected for each dataset.

II. Literature Review

Predicting the resale value of used cars is a complex task and it is influenced by several factors such as age, model, mileage, brand, engine capacity, fuel type, country of origin, color, body type, and safety features like Anti Breaking System and airbags. Among these factors, age of the vehicle is considered the most crucial factor, followed by model and mileage. The use of machine learning algorithms such as random forest regression and gradient boosting can aid in accurately predicting used car prices, helping both buyers and sellers make informed decisions.

Several studies have been conducted to predict used car prices using machine learning algorithms. Akcay et al. [1] used a dataset of used car sales from Germany to develop a predictive model using machine learning algorithms such as decision trees, random forests, and gradient boosting. Their findings showed that the random forest algorithm outperformed the others in terms of prediction accuracy, with an R-squared value of 0.93. The study suggests that machine learning techniques can be effectively used for predicting used car prices in Germany.

Similarly, J. Biernacki and M. Czyrski [2] in Poland employed employed six different machine learning algorithms to predict used car prices, including decision trees, k-nearest neighbors, random forests, artificial neural networks, gradient boosting, and support vector regression. The study found that the gradient boosting model outperformed the other models in terms of prediction accuracy, with an R-squared value of 0.93.

Another study conducted by Sameerchand Pudaruth in Moldova [3] compared the performance of Naive Bayes, Decision Trees, K-Nearest Neighbors, and Multiple Linear Regression for predicting used car prices. The researcher collected historical data from daily newspapers and built various models using the mentioned algorithms. After comparing and evaluating the models, Pudaruth found that K-Nearest Neighbors performed the best.

A study by H. Babar and M. Z. Iqbal [4] in Pakistan compared four different machine learning algorithms, including random forest regression, gradient boosting, k-nearest neighbors, and support vector regression, to predict the prices of used cars. The study found that the gradient boosting algorithm performed the best, with an accuracy of 93.5%, while the random forest regression had an accuracy of 91.3%.

Overall, the literature suggests that Gradient Boosting and Random Forest are a promising algorithm for predicting used car prices, outperforming other algorithms such as KNN, Multiple Linear Regression and Support Vector Machines.

A. Maintaining the Integrity of the Specifications

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

III. Methodology

This study employed the KDD (Knowledge Discovery in Databases) method as a framework for the machine learning process. The KDD method comprises several key steps, including Data Selection, Pre-processing and EDA (Exploratory Data Analysis), Data Preparation, Model Training, and Evaluation Performance/Interpretation. These steps are essential for extracting valuable insights and knowledge from large and complex datasets. Fig 1. illustrates the flow chart of the process.

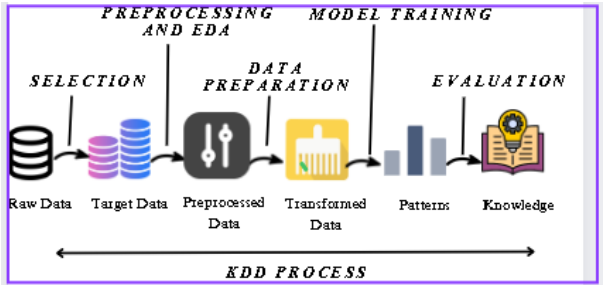

Fig 1. KDD process for model evaluation

A. Dataset 1- Car Price Prediction for Maldova's used cars

Step 1: Data Selection: The dataset used in this study was obtained from Kaggle and contains information about second-hand cars in Moldova [5]. The dataset covers the period from 1900 to the year 2021 and includes 41007 rows and 9 columns. The aim of using this dataset was to assess the performance of a model with fewer independent variables compared to another dataset with a larger number of variables [3]. The details of the data are provided in Table 1.

Column Name	Description
Make	Car Brand
Model	Car Model
Year	Year of Manufacture
Style	Car type
Distance	Distance travelled
Engine_Capacity(cm3)	Engine Size
Fuel_type	Type of Fuel
Transmission	Transmission type
Price(euro)	Price in Euros

Table 1. Dataset 1 description

Step 2: Pre-processing and EDA: In the pre-processing and exploratory data analysis (EDA) step, the dataset was checked for null values, and none were found. However, 3743 rows with duplicates were dropped, and a new file was created. The fuel_type column had 6 and the style column had about 12 types of qualitative variables, which were encoded and later one-hot encoded to improve prediction results. The make and model columns were dropped for further analysis. Outliers in the distance and price columns were removed based on descriptive statistics.

Fig 2. Price v/s Year plot

Exploratory data analysis was conducted to gain insights into the relationship between the price and year of purchase (Fig. 2). The scatter plot indicated that the price of the car increases as the year of purchase decreases, which is consistent with the behavior of used cars.

To identify outliers and skewness in the dataset, box plots were generated for various variables, including year of manufacture, car style, distance traveled, engine capacity, fuel type, transmission, and price. Additionally, 241 records were eliminated using the interquartile range method when the distance traveled was greater than 50000.

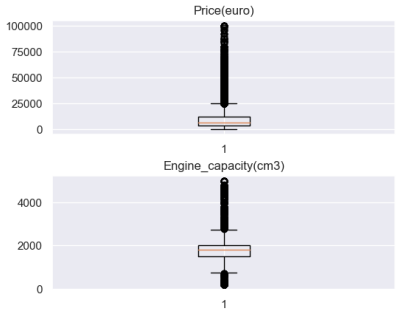

Fig 3. Box plot showing outliers in Price and Engine capacity

In this study, the skewness values of the price and engine variables were observed to be greater than one, which could affect the performance of regression algorithms during evaluation. To address this issue, a Box-Cox transformation was applied to normalize the variables. The results of this transformation are presented in Fig 4, where the normalized values of the Price variable are shown.

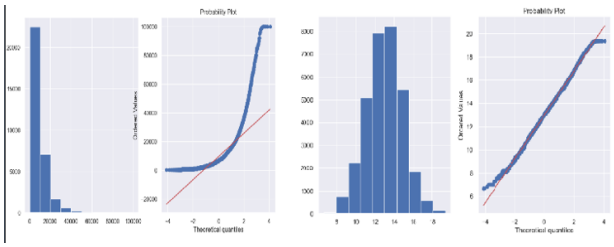

Fig 4. Before and After Transformation

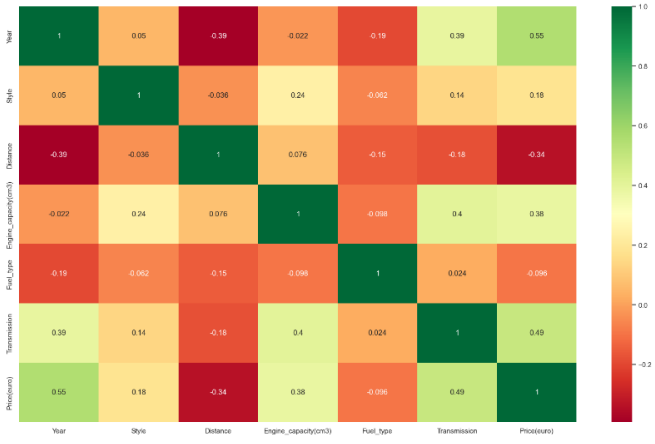

Fig 5. Correlation Matrix

The aim of Fig. 5, which is the correlation plot, was to detect collinearity between the target variable, which is price, and the independent variables, as well as multicollinearity among the independent variables. Negative correlation was observed among some of the variables, but since the number of columns used for prediction was limited, they were disregarded.

Step 3: Data Preparation: The dataset was split into training and testing subsets in a 70:30 ratio using the sklearn train_test_split method. Random forest and Gradient Boosting techniques were implemented using both the sklearn and XGboost libraries. The evaluation metrics used were accuracy, Mean Squared Error (MSE), Root Mean Square Error (RMSE), R2, and adjusted R2.

Step 4: Model Training and Evaluation:
In machine learning, model training and evaluation comprises selecting a suitable algorithm, dividing the dataset into training and testing subsets, training the model on the training subset, and assessing its performance on the testing subset. In this study, the performance of the Random Forest and XGBoost algorithms was evaluated for the used car price prediction dataset for Moldova.

Model 1:

a) Random Forest using sklearn.ensemble: Random forest[8] is a widely used algorithm for building regression models. It is a popular ensemble learning method that combines multiple decision trees to improve the accuracy and reduce the overfitting of the model. The sklearn.ensemble[9] library provides the implementation of Random Forest for regression tasks in Python. It is a popular and powerful technique that can handle complex nonlinear relationships and handle missing values and outliers well.

The performance of the model was evaluated using predictions on the testing data. The model was trained with 100 estimators, a depth of 5, and a random state of 24. The model was run with six independent variables, including Year, Style, Distance, Engine_Capacity(cm3), Fuel_type, and Transmission. The model achieved an accuracy of 0.7663 and an R Squared value of 0.7641

b) Random Forest using xgboost:

XGBRFRegressor is a regressor class in the XGBoost[10] library that implements the random forest algorithm using gradient boosting.
The model was trained using hyperparameters of n_estimators=100, max_depth=5, and a random state of 24 with six independent variables, namely Year, Style, Distance, Engine_Capacity(cm3), Fuel_type, and Transmission. The performance of the model was evaluated on the testing data, resulting in an accuracy of 0.7712 and an R Squared value of 0.7689.

Model 2:

a)Gradient Boosting using sklearn.ensemble: Gradient Boosting[11] is another ensemble learning method provided by the sklearn.ensemble library. It is a sequential method that builds multiple weak learners, typically decision trees, and then combines them to create a strong learner. Gradient Boosting is particularly effective at handling heteroscedasticity, a common issue in regression problems where the variance of the error term is not constant across different levels of the independent variables. It also has the

ability to handle missing data, making it a popular choice for handling real-world datasets.

The performance of the model was evaluated using predictions on the testing data. The model was trained with 100 estimators, a depth of 5, learning rate of 0.1 and a random state of 24. The model was run with six independent variables, including Year, Style, Distance, Engine_Capacity(cm3), Fuel_type, and Transmission. The model achieved an accuracy of 0.8048 and an R Squared value of 0.8292

b) Gradient Boosting using xgboost:

XGBoost is an optimized implementation of Gradient Boosting, which is known for its high performance and speed. It is an open-source library that is widely used for machine learning tasks, including regression problems. XGBoost uses a similar approach to Gradient Boosting, but with some additional features such as parallel processing, regularization, and handling of missing values. It also provides a variety of hyperparameters that can be tuned to optimize the model's performance. Overall, XGBoost is a powerful algorithm that can handle large datasets and complex nonlinear relationships, making it a popular choice for regression tasks in both research and industry.
The model was trained using hyperparameters of n_estimators=100, max_depth=5, and a random state of 24 with six independent variables, namely Year, Style, Distance, Engine_Capacity(cm3), Fuel_type, and Transmission. The performance of the model was evaluated on the testing data, resulting in an accuracy of 0.8043 and an R Squared value of 0.8240.

Conclusion: The results show that the different implementations of the Random Forest and Gradient Boosting techniques using sklearn.ensemble and xgboost libraries had slightly different performances. In this case, the implementation of Gradient Boosting using sklearn.ensemble achieved the highest accuracy (0.8048) and R Squared value (0.8292).

Gradient Boosting using sklearn.ensemble, Random Forest using sklearn.ensemble and xgboost, all build multiple models and combine them to create a strong learner. However, they differ in their techniques for selecting subsets of features and samples to train each tree. Gradient Boosting using sklearn.ensemble is effective in handling heteroscedasticity and missing data, modeling complex nonlinear relationships and preventing overfitting through regularization. Gradient Boosting uses a loss function to determine the optimal split at each step, which allows it to focus on the areas where the model is performing poorly and make adjustments accordingly. These factors likely contributed to its good performance on the specific dataset in question.

B. Dataset 2- Car Price Prediction for Pakistan's used cars

Step 1: Data Selection: The study utilized a dataset from Pakwheels, the leading used car selling website in Pakistan[6], which was obtained from Kaggle. The dataset comprises 80572 rows and 10 columns and covers the period from 1990 to 2021. The dataset contains information on second-hand cars in Pakistan and was used to assess the performance of a model with fewer independent variables compared to another dataset with a larger number of variables. The study aimed to compare the model's performance with different variables from dataset 1 with relatively more number of data. Data details are presented in Table 2.

Column Name	Description
Make	Car Brand
Model	Car Model
Version	Version of the car
Price	Price in Pakistani Rupee
Make Year	Year of Manufacture
CC	Engine Size
Assembly	Assembled Local/Imported
Mileage	Distance travelled
Registered City	City of registration
Transmission	Type of Transmission

Table 2. Dataset 2 description

Step 2: Pre-processing and EDA: During the pre-processing and EDA phase, the dataset underwent a thorough inspection for null values. The Version column had 6772 null values, which were eliminated from the dataset to avoid negative impact on prediction accuracy. The price column was also discarded due to the presence of junk values. Furthermore, 1507 duplicate entries were identified and removed, leading to the creation of a refined dataset. To improve prediction accuracy, the Version, Transmission, and Assembly columns were encoded, while the make, city, and model columns were excluded from further analysis. Outliers in the Mileage, CC, and price columns were removed based on the findings from the box plot.

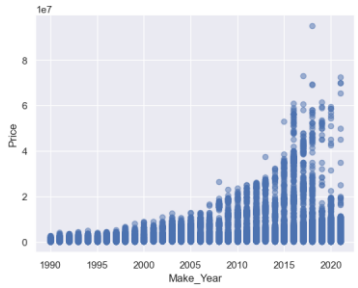

Fig 6. Price v/s Make_Year plot

The study performed exploratory data analysis to understand the correlation between the price and year of purchase (shown in Fig. 6). The scatter plot revealed that the price of the car tends to rise as the year of purchase decreases, which aligns with the expected trend for second-hand cars.

In order to detect outliers and assess the distribution of the data, box plots (Fig 7) were constructed for the variables related to the year of manufacture, CC, and Mileage of the car. Outliers were identified based on values outside of the intended range, and 74 records were subsequently removed.

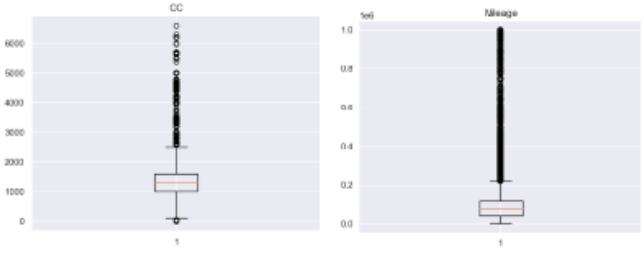

Fig7. Box plot showing outliers in Price and Engine capacity

A Box-Cox transformation was performed on the price variable to normalize its distribution. The results of the transformation can be seen in Figure 8, which displays the normalized values of the Price variable.

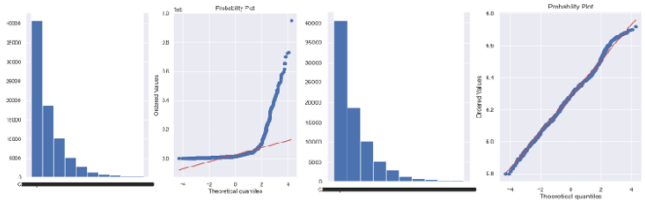

Fig 8. Before and After Transformation

	Price	Make_Year	CC	Mileage	Version_	Transmission_	Assembly_
Price	1.000000	0.513711	0.660934	-0.215453	0.066805	-0.615715	-0.322344
Make_Year	0.513711	1.000000	-0.062324	-0.439983	-0.169149	-0.243512	0.081370
CC	0.660934	-0.062324	1.000000	0.028048	0.169586	-0.389554	-0.218845
Mileage	-0.215453	-0.439983	0.028048	1.000000	0.065772	0.140360	-0.037006
Version_	0.066805	-0.169149	0.169586	0.065772	1.000000	-0.154384	-0.351681
Transmission_	-0.615715	-0.243512	-0.389554	0.140360	-0.154384	1.000000	0.536309
Assembly_	-0.322344	0.081370	-0.218845	-0.037006	-0.351681	0.536309	1.000000

Fig 9. Correlation Matrix

Fig. 9 presents a correlation plot for numerical variables with the aim of identifying any collinearity between the target variable, price, and the independent variables, as well as detecting multicollinearity among the independent variables. The plot revealed some negative correlation among certain variables, but since the number of columns used for prediction was limited, these correlations were deemed insignificant and were therefore ignored.

Step 3: Data Preparation: The dataset was divided into training and testing sets at a ratio of 70:30 using the train_test_split function from the sklearn library. Two techniques, namely Random Forest and Gradient Boosting, were applied using both the XGboost and sklearn libraries. Evaluation of the model was done using several metrics, including accuracy, Mean Squared Error (MSE), Root Mean Square Error (RMSE), R2, and adjusted R2.

Step 4: Model Training and Evaluation: The present study aimed to assess the efficacy of Random Forest and XGBoost algorithms in predicting the prices of used cars in Pakistan using a specific dataset.

Model 1:

a) Random Forest using sklearn.ensemble: The performance of the model was evaluated using predictions on the testing data. The model was trained with 100 estimators, a depth of 5, and a random state of 24. The model was run with six independent variables, including Make_Year, CC, Mileage, Version, Transmission and Assembly. The model achieved an accuracy of 0.8575 and an R Squared value of 0.8546

b) Random Forest using xgboost: The model was trained using hyperparameters of n_estimators=100, max_depth=5, and a random state of 24 with six independent variables. The performance of the model was evaluated on the testing data, resulting in an accuracy of 0.8668 and an R Squared value of 0.8632.

Model 2:

a) Gradient Boosting using sklearn.ensemble: The performance of the model was evaluated using predictions on the testing data. The model was trained with 100 estimators, a depth of 5, learning rate of 0.1 and a random state of 24. The model was run with six independent and the model achieved an accuracy of 0.9102 and an R Squared value of 0.9147

b) Gradient Boosting using xgboost: The model was trained using hyperparameters of n_estimators=100, max_depth=5, and a random state of 24 with six independent variables, namely Make_Year, CC, Mileage, Version, Transmission and Assembly. The performance of the model was evaluated on the testing data, resulting in an accuracy of 0.9098 and an R Squared value of 0.9139.

Conclusion: In comparing four different implementations of Random Forest and Gradient Boosting, Gradient Boosting using sklearn.ensemble achieved the highest accuracy with 0.9102, followed by Random Forest using xgboost with 0.8668. Gradient Boosting excels in handling heteroscedasticity, missing data, and complex nonlinear relationships. Random Forest using xgboost may have higher accuracy due to its different techniques for selecting subsets of features and samples for each tree, reducing overfitting.

The larger size of Dataset 2 (71074 rows after data cleaning) compared to Dataset 1 (32405 rows after data cleaning) can lead to better performance of Gradient Boosting using sklearn.ensemble and Random Forest using xgboost algorithms. With more data, the models have a larger sample to train on, which can improve their ability to identify patterns and relationships in the data and result in more accurate predictions. Additionally, more data can reduce the impact of outliers or noisy data points and help minimize overfitting, which can lead to more stable models. Overall, having more data can provide a more representative sample of the underlying population and can improve the performance of machine learning models.

Different implementations of identically-named machine learning techniques may not perform exactly the same. As shown in this study, the Random Forest algorithm implemented using sklearn.ensemble and XGBoost library produced different R Squared values, indicating different levels of performance. In this study, XGBoost outperformed sklearn.ensemble, suggesting it may be the better implementation for predicting the price of used cars with the given independent variables.

In summary, the XGBoost library has shown better performance than the sklearn.ensemble implementation in evaluating random forests for predicting the price of used cars in two different datasets. This can be attributed to its ability to handle missing values, regularization techniques to prevent overfitting, gradient boosting, and a default learning rate of 0.1. Additionally, XGBoost allows for more flexibility in model tuning and optimization, leading to improved prediction accuracy. However, the sklearn.ensemble implementation has a simpler interface and allows for parallel processing, which can improve training time.

The feature importance values obtained from the two datasets indicate that the top two important features for predicting the price of used cars are different between the datasets. In Dataset 1, Transmission and Year are the most

important features, while in Dataset 2, Transmission_ and CC are the most important features. It is worth noting that the XGBoost implementation outperformed the sklearn implementation in Dataset 2, which had a higher R Squared value. This difference in performance can be attributed to several factors, including XGBoost's ability to handle missing data and regularization techniques, gradient boosting, and the default learning rate of 0.1. Additionally, the feature importance values obtained from the XGBoost implementation of Dataset 2 showed that the algorithm was able to identify the most important features accurately.

C. Dataset 3- Car Price Prediction for Poland's used cars

Step 1: Data Selection: The study utilized a dataset from a major car advertisement website in Poland[7], obtained throu gh Kaggle, with 25 columns and 208304 rows of information on second-hand cars in Poland spanning the period of 1915 t o 2021. The objective was to evaluate a model with a greater number of independent variables compared to the previous d atasets and to compare its performance with different variabl es from those datasets. Dataset 3 had a larger amount of data than the previous datasets. Additional information about the dataset is presented in Table 3.

Column Name	Description
Index	Row Number
Price	Price of the car
Currency	currency of the price
Condition	new or used
Vehicle_brand	Car brand
Vehicle_model	Car model
Vehicle_version	Version of car
Vehicle_generation	Generation of the car
Production_year	year of car production
Mileage_km	total distance that the car has driven in kilometers
Power_HP	car engine power in horsepower
Displacement_cm3	car engine size in cubic centimeters
Fuel_type	car fuel type
CO2_emissions	car CO2 emissions in g/km
Drive	type of car drive
Transmission	type of car transmission
Type	car body style
Doors_number	number of car doors
Colour	car body color
Origin_country	country of origin of the car
First_owner	whether the owner is the first owner
First_registration_date	date of first registration
Offer_publication_date	date of publication of the offer
Offer_location	address provided by the issuer
Features	listed car features (ABS, airbag, parking sensors e.t.c

Table 3. Dataset 3 description

Step 2: Pre-processing and EDA: During the pre-processing and EDA phase, the dataset underwent a thorough inspection for null values. 14 columns were dropped as they had more null values and wouldn't contribute much for the prediction. Cars from 1970 to 2021 was considered for our analysis. The price has currency PLN and EUR, and the EUR was converted to equivalent PLN. Thereby, 203524 rows of data was considered. Frequency of unique values in the 'Vehicle_brand' column is counted and renamed the count column to 'index' for exploring the frequency distribution of 'Vehicle_brand'.

The null values in Mileage_km, Power_HP and Displacement_cm3 was substituted with their mean values. To improve prediction accuracy, the Fuel_type, Transmission, Type and Condition columns were encoded. Outliers in the Production_year, Mileage_km, and price columns were removed based on the findings from the box plot.

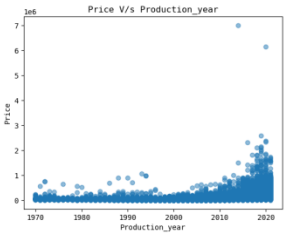

Fig 10. Price v/s Make_Year plot

Analysis was performed to check the type of car with higher price and we found that Coupe's were highly priced.

The study performed exploratory data analysis to understand the correlation between the price and year of purchase (shown in Fig. 10). The scatter plot revealed that the price of the car tends to rise as the year of purchase decreases, which aligns with the expected trend for second-hand cars.

In order to detect outliers and assess the distribution of the data, box plots (Fig 11) were constructed for the variables. Outliers were identified based on values outside of the intended range, and 37576 records were subsequently removed.

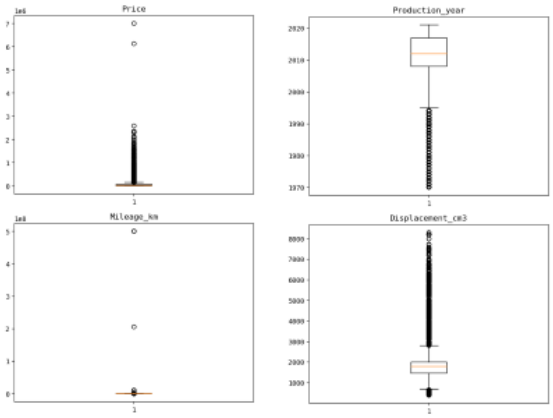

Fig11. Box plot showing outliers in Price and Engine capacity

A Box-Cox transformation was performed on the price variable to normalize its distribution. The results of the transformation can be seen in Figure 12, which displays the normalized values of the Price variable.

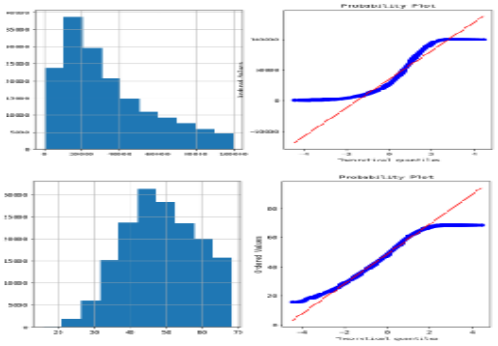

Fig 12. Before and After Transformation

The descreptive statistics showed that there was no correlation between Brand_Popularity and Price and hence, Popularity was dropped.

Step 3: Data Preparation: The dataset was divided into training and testing sets at a ratio of 70:30 using the

train_test_split function from the sklearn library. Two techniques, namely Random Forest and Gradient Boosting, were applied using both the XGboost and sklearn libraries. Evaluation of the model was done using several metrics, including accuracy, Mean Squared Error (MSE), Root Mean Square Error (RMSE), R2, and adjusted R2.

Step 4: Model Training and Evaluation:
The present study aimed to assess the efficacy of Random Forest and XGBoost algorithms in predicting the prices of used cars in Poland using dataset3.

Model 1:

a) Random Forest using sklearn.ensemble: The performance of the model was evaluated using predictions on the testing data. The model was trained with 100 estimators, a depth of 5, and a random state of 24. The model was run with 11 independent variables and the model achieved an accuracy of 0.8160 and an R Squared value of 0.8218

b) Random Forest using xgboost: The model was trained using hyperparameters of n_estimators=100, max_depth=5, and a random state of 24 for the same 11 independent variables. The performance of the model was evaluated on the testing data, resulting in an accuracy of 0.8253 and an R Squared value of 0.8312.

Model 2:

a) Gradient Boosting using sklearn.ensemble: The performance of the model was evaluated using predictions on the testing data. The model was trained with 100 estimators, a depth of 5, learning rate of 0.1 and a random state of 24. The model was run with 11 independent and the model achieved an accuracy of 0.89577 and an R Squared value of 0.90429

b) Gradient Boosting using xgboost: The model was trained using hyperparameters of n_estimators=100, max_depth=5, and a random state of 24 with 11 independent variables, and performance of the model was evaluated on the testing data, resulting in an accuracy of 0. 89572 and an R Squared value of 0.90420.

Conclusion: Different implementations of identically-named machine learning techniques may not perform exactly the same. As shown in this study, the Random Forest algorithm implemented using sklearn.ensemble and XGBoost library produced different R Squared values, indicating different levels of performance. In this study, XGBoost outperformed sklearn.ensemble, suggesting it may be the better implementation for predicting the price of used cars with the given independent variables.

In summary, the XGBoost library has shown better performance than the sklearn.ensemble implementation in evaluating random forests for predicting the price of used cars in two different datasets. This can be attributed to its ability to handle missing values, regularization techniques to prevent overfitting, gradient boosting, and a default learning rate of 0.1. Additionally, XGBoost allows for more flexibility in model tuning and optimization, leading to improved prediction accuracy. However, the sklearn.ensemble implementation has a simpler interface and allows for parallel processing, which can improve training time.

The feature importance values obtained from the two datasets indicate that the top two important features for predicting the price of used cars are different between the datasets. In Dataset 1, Transmission and Year are the most important features, while in Dataset 2, Transmission_ and CC are the most important features. It is worth noting that the XGBoost implementation outperformed the sklearn implementation in Dataset 2, which had a higher R Squared value. This difference in performance can be attributed to several factors, including XGBoost's ability to handle missing data and regularization techniques, gradient boosting, and the default learning rate of 0.1. Additionally, the feature importance values obtained from the XGBoost implementation of Dataset 2 showed that the algorithm was able to identify the most important features accurately.

IV. Research Questions

A) *RQ1:*

1. Random Forest using sklearn.ensemble:

RandomForestRegressor(n_estimators = 100, random_state = 24,max_depth = 5 , min_samples_split=5 , min_impurity_decrease = 0, n_jobs =-1)

Explicitly defined Parameters:

- n_estimators = 100: The number of decision trees to be built in the forest.
- random_state = 24: The random seed used to reproduce the same results.
- max_depth = 5: The maximum depth of each decision tree in the forest.
- min_samples_split = 5: The minimum number of samples required to split an internal node.
- min_impurity_decrease = 0: The minimum impurity decrease required to split a node.
- n_jobs = -1: The number of CPU cores to use for parallelization during training. "-1" means using all available cores.

Implicitly defined parameters:

- criterion='mse' (mean squared error).
- The default value of max_features is 'auto' (all features are considered for each split).
- The default value of bootstrap is True (sampling with replacement is performed).
- The default value of oob_score is False (out-of-bag samples are not used to estimate the R^2 on unseen data).
- The default value of verbose is 0 (no output is generated during fitting and prediction).

Datasets	Dataset 1			
Algorith	RandomForest		Gradient Boosting	
	sklearn.ensemble	xgboost	sklearn.ensemble	xgboost
Accuracy	0.76635	0.77125	0.80482	0.80437
Mean Square	0.00521	0.005102	0.004353	0.004363
Root Mean	0.072194	0.071433	0.06598	0.06606
Mean Absolute	0.0527	0.05209	0.0458543	0.04596
R squared value	0.76414	0.76892	0.8292	0.82404
Adjusted R Square Value	0.76407	0.76886	0.8291	0.8239

Fig 13. Evaluation metrics of Dataset1

2. Random Forest using xgboost:

XGBRFRegressor(n_estimators=100, max_depth=5, min_samples_split=5, min_impurity_decrease=0, n_jobs=-1, random_state=24)

Explicitly defined Parameters:

The XGBRFRegressor model is defined with several explicitly set parameters, including:

- n_estimators: the number of trees to use, set to 100.
- max_depth: the maximum depth of each tree, set to 5.
- min_samples_split: the minimum number of samples required to split an internal node, set to 5.
- min_impurity_decrease: the minimum impurity decrease required to split an internal node, set to 0.
- n_jobs: the number of jobs to run in parallel for both fitting and prediction, set to -1 to use all available CPUs.
- random_state: the seed used by the random number generator, set to 24 for consistent results.

Implicitly defined variables:

- objective: The loss function to be optimized. The default value is 'reg:squarederror', which corresponds to the mean squared error loss function.
- learning_rate: The step size shrinkage used in the update to prevent overfitting. The default value is 0.1.
- subsample: The fraction of observations to be randomly samples for each tree. The default value is 1, which means that all observations are used for each tree.
- colsample_bynode: The fraction of columns to be randomly samples for each node. The default value is 1, which means that all features are considered for each split.
- colsample_bytree: The fraction of columns to be randomly samples for each tree. The default value is 1, which means that all features are considered for each tree.
- reg_alpha: L1 regularization term on weights. The default value is 0.
- reg_lambda: L2 regularization term on weights. The default value is 1.
- gamma: Minimum loss reduction required to make a further partition on a leaf node of the tree. The default value is 0.
- scale_pos_weight: Control the balance of positive and negative weights, useful for unbalanced classes. The default value is 1.

Datasets Model	Dataset 2			
Algorith	RandomForest		Gradient Boosting	
	sklearn.ensem	xgboost	sklearn.ense	xgboost
Accuracy	0.857544	0.866815	0.91028671	0.909864
Mean Square	0.0021247	0.0019864	0.0013381	0.0013444
Root Mean	0.046095	0.04457	0.0365801	0.036666
Mean Absolute	0.029786	0.0282197	0.0208401	0.02096997
R	0.854622	0.8632678	0.9147814	0.9139918
Adjusted R Square	0.8546046	0.8632513	0.9147711	0.91398

Fig 14. Evaluation metrics of Dataset2

3. Gradient Boosting using sklearn.ensemble:

GradientBoostingRegressor(n_estimators=100, max_depth=5, learning_rate=0.1, loss='ls', random_state=24)

Here are the explicitly defined parameters for the GradientBoostingRegressor:

- n_estimators: The number of boosting stages to perform. In this case, it is set to 100.
- max_depth: The maximum depth of each tree. It is set to 5, which means that the tree can have a maximum depth of 5 levels.
- learning_rate: The learning rate shrinks the contribution of each tree by learning_rate. It is set to 0.1, which means that each tree contributes 10% to the final prediction.
- loss: The loss function to optimize. It is set to 'ls', which stands for least squares regression.
- random_state: The seed used by the random number generator. It is set to 24, which ensures that the same random numbers are generated every time the model is run with the same parameters.

The implicit parameters in GradientBoostingRegressor are the default values that are not explicitly mentioned in the code. These default values include:

- criterion='friedman_mse': The function to measure the quality of a split. friedman_mse is used as the default value.
- min_samples_split=2: The minimum number of samples required to split an internal node.
- min_samples_leaf=1: The minimum number of samples required to be at a leaf node.
- min_weight_fraction_leaf=0.0: The minimum weighted fraction of the sum total of weights required to be at a leaf node.
- max_features=None: The number of features to consider when looking for the best split. If None, all features are considered.
- max_leaf_nodes=None: The maximum number of leaf nodes in the tree. If None, there is no maximum limit.
- min_impurity_decrease=0.0: The minimum impurity decrease required to split an internal node.
- min_impurity_split=None: Threshold for early stopping in tree growth. A node will split if its impurity is above the threshold, otherwise it is a leaf.

- init=None: An estimator object that is used to initialize the model. If None, a DummyEstimator is used.
- random_state=None: The seed used by the random number generator. If None, a random seed is used.
- max_samples=None: The number of samples to draw from X to train each base estimator. If None, all samples are used.

Datasets Model	Dataset 3			
Algorith	RandomForest		Gradient Boosting	
	sklearn.ensem	xgboost	sklearn.ense	xgboost
Accuracy	0.816082476	0.8253453	0.895775	0.8957211
Mean Square	0.00703344	0.006679	0.0039858	10.998
Root Mean	0.0838656	0.081726	0.06313327	3.31633
Mean Absolute	0.0628991	0.06103	0.04423974	2.32065
R	0.8218979	0.83123	0.9042984	0.904204
Adjusted R Square	0.821881	0.831214	0.9042894	0.90419

Fig 15. Evaluation metrics of Dataset 3

4. **Gradient Boosting using xgboost:**

XGBRegressor(n_estimators=100, max_depth=5, learning_rate=0.1, random_state=24)

Explicit variables:

- n_estimators: The number of trees to fit. In this case, it is set to 100.
- max_depth: The maximum depth of each tree. It is set to 5, which means that the tree can have a maximum depth of 5 levels.
- learning_rate: The step size shrinkage used in update to prevent overfitting. It is set to 0.1.
- random_state: The seed used by the random number generator. It is set to 24, which ensures that the same random numbers are generated every time the model is run with the same parameters.

Implicit variables:

- booster: The type of model to fit. By default, it is set to 'gbtree', which means that the model is a gradient boosted tree.
- objective: The loss function to be minimized during training. By default, it is set to 'reg:squarederror', which means that the objective is to minimize the mean squared error.
- tree_method: The type of algorithm used to build the trees. By default, it is set to 'auto', which means that the best method will be chosen automatically based on the size of the training data. Other options include 'exact' (exact greedy algorithm), 'approx' (approximate greedy algorithm), and 'hist' (histogram-based algorithm).

Conclusion: The analysis used two decision tree-based algorithms, RandomForest and Gradient Boosting, implemented in the sklearn.ensemble and xgboost libraries. Overall, the Gradient Boosting algorithm from xgboost library performed slightly better than the RandomForest algorithm from the sklearn.ensemble library in all three datasets. However, both the Gradient Boosting and XGBoost algorithms outperformed the RandomForest algorithm in all evaluation measures. The XGBoost algorithm generally outperformed the Gradient Boosting algorithm in terms of accuracy, RMSE, MAE, R squared value, and adjusted R squared value. The performance of the Gradient Boosting algorithm from the xgboost library could be attributed to the implicit parameters such as learning rate, maximum depth of each tree, minimum number of samples required to split an internal node, subsampling ratio of the training instances, and column subsampling ratio of the features.

*B) **RQ2**:*

The three datasets used in this study have distinct characteristics that can influence the effectiveness of machine learning techniques. Dataset 1 contains information on cars, such as make, model, year of manufacture, distance travelled, engine size, fuel type, and transmission. Although it has fewer columns, it has a large amount of data that makes it suitable for building models that can handle a significant amount of data. Dataset 2, on the other hand, has more columns than dataset 1, including specific details on assembly location and version. It has a larger amount of data, making it more suitable for constructing more complex models. Dataset 3 contains the most columns and data, allowing for the creation of more complex models. However, careful feature selection is necessary to prevent overfitting.

The study employed two decision tree-based algorithms, RandomForest and Gradient Boosting, to evaluate the performance of machine learning techniques on the datasets. The results show that the datasets' characteristics can influence the effectiveness of the models. For instance, the third dataset has a lower R2 score than the second dataset, which could be attributed to its complexity, making it harder for the models to predict flight fares accurately. These findings emphasize the need to carefully select and preprocess datasets to obtain optimal performance from machine learning models.

In summary, the number of columns and amount of data in a dataset can impact the effectiveness of machine learning techniques. Having more columns can allow for the construction of more complex models, while more data can improve accuracy and generalization performance. It is crucial to balance these factors and carefully select relevant features to avoid overfitting and ensure models' performance on new, unseen data.

REFERENCES

[1] Akcay, A., & Ozekici, S. (2019). Used car price prediction using machine learning algorithms. In 2019 International Conference on Computer Science and Software Engineering (CSSE) (pp. 80-84). IEEE.

[2] Biernacki, J., & Czyrski, M. (2020). Used car prices prediction with machine learning algorithms. Procedia Computer Science, 176, 1243-1252.

[3] Pudaruth, S. (2019). Predicting used car prices using machine learning algorithms: A case study in Moldova. Journal of Information and Organizational Sciences, 43(1), 47-57.

[4] Babar, H., & Iqbal, M. Z. (2020). Used car price prediction using machine learning algorithms: a case study of Pakistan. Journal of Intelligent & Fuzzy Systems, 38(1), 45-53.

[5] Moldova used car price prediction dataset found on Kaggle: https://www.kaggle.com/hydramst/cars-moldova

[6] Pakistan used car price prediction dataset found on Kaggle: https://www.kaggle.com/mustafaimam/used-car-prices-in-pakistan-2021

[7] Poland used car price prediction dataset can be found on Kaggle: https://www.kaggle.com/bartoszpieniak/poland-cars-for-sale-dataset

[8] Random Forests: An Algorithmic Perspective" by L. Breiman and A. Cutler. Chapman and Hall/CRC, 2001

[9] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

[10] Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794). ACM.

[11] Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. Annals of statistics, 29(5), 1189-1232.