Predicting Average Price of Used Cars

DISSERTATION

Submitted in partial fulfillment
of the requirements for the course of

CSE 587 - Data Intensive Computing

By
Shri Gayathri Chandrasekar (ESDS - 50495167)
Sree Lekshmi Prasannan (ESDS - 50495144)
Jaskirat Singh (ESDS - 50495170)

Under the guidance of:

Dr. Shamsad Parvin
Department of Computer Science and Engineering

The State University of New York at Buffalo

<u>Phase 2</u>

Problem Statement

The market for second hand cars is enormous and thriving, which defines the automotive sector. Making accurate predictions about the average cost of used automobiles is becoming increasingly important as buyers look for trustworthy information to help them make informed selections.

In this project, we will use a data-driven approach to create a reliable predictive model that can calculate the average cost of used automobiles based on various traits and attributes. We will create a series of **regression models** that forecast used automobile prices while taking into account both auction bid prices and potential repair expenses, to help buyers make budget-conscious purchases.

Background

The used car industry is enormous and diverse with vehicles ranging in age, condition, and price. Due to hidden expenses, including potential repair costs that might not be immediately apparent during an auction or sale, it can be difficult for prospective buyers to determine a car's genuine value. A buyer's budget may be strained and regretful financial choices may be made as a result of overbidding on a vehicle that needs significant repairs. This difficulty is especially important at auctions, where the excitement and competitiveness can obscure a car's actual value. By accurately predicting the average price, inclusive of repair costs, buyers can navigate the second-hand car market with confidence, ensuring they obtain value for money and avoid financial pitfalls.

Data Acquisition

We took data from *data.world* website and it is available in CSV format -
[*https://data.world/data-society/used-cars-data*](https://data.world/data-society/used-cars-data)
The dataset has approx 370,000 rows and 20 columns of used cars scraped with Scrapy from Ebay-Kleinanzeigen.
<u>Some of the categorical values are in German, so we have to translate those into English.</u>

List of features in the dataset:

- *<u>dateCrawled</u>*: when this ad was first crawled
- *<u>name</u>:* `name` of the car
- *<u>seller</u>:* private or dealer

- *offerType:* `Gesuch` means `request` and `Angebot` means `listing`
- abtest: Tells whether the car is being modified or not
- vehicletype: Type of vehicle, SUV, coupe, etc
- yearOfRegistration: At which year the car was first registered
- gearbox: Car is automatic or manual
- powerPS: The horsepower of the car in PS
- model: Model of the car
- kilometer: How many kilometers the car has driven
- monthOfRegistration: At which month the car was first registered
- fuelType: Diesel, CNG, electric, etc
- Brand: Company of the car
- notRepairedDamage: If the car has damage and is not repaired yet
- dateCreated: The date for which the ad at eBay was created
- nrOfPictures: Number of pictures in the ad
- postalCode: Postal code of car location
- lastSeenOnline: When the crawler saw this ad last online

Feature Selection

List of features left after Data Cleaning -

- *name:* `name` of the car
- *seller:* private or dealer
- *offerType:* `Gesuch` means `request` and `Angebot` means `listing`
- abtest: Tells whether the car is being modified or not
- vehicletype: Type of vehicle, SUV, coupe, etc
- gearbox: Car is automatic or manual
- powerPS: The horsepower of the car in PS
- model: Model of the car
- kilometer: How many kilometers the car has driven
- fuelType: Diesel, CNG, electric, etc
- Brand: Company of the car
- notRepairedDamage: If the car has damage and is not repaired yet
- age_of_car: Extracted this feature from `yearOfRegistration` and `lastSeenOnline`

After feature selection and encoding categorical features using the one-hot encoder, the shape of the dataset is - **(183480, 309)**

Since the above dataset is too big for our local system to process and run algorithms and most importantly do hyperparameter tuning. Therefore, we have decided to reduce the dataset size.

To reduce the size we have chosen the `age_of_car` as a criterion - Currently, we have cars aged from 0 to 20 years old. To reduce the size we are only considering cars with ages between 0 and 10.

After applying the above filter the final shape of our dataset is: (57122, 286)

Machine Learning models will be trained on the dataset having the shape mentioned above.

Part 1 - Algorithms / Visualizations

List of Algorithms we will be implementing:

1. Decision Tree Regressor
2. Random Forest Regressor
3. XG Boost Regressor
4. KNN Regressor
5. Support Vector Regressor
6. Lasso / Ridge Regression
7. Light GBM Regressor
8. Gradient Boosting Regressor

Evaluation Metrics: This is a Regression problem where we are trying to predict the average price of a used car, therefore we have use various metrics like -

- **R^2 Score -** It represents the proportion of the variance in the dependent variable that is predictable from the independent variable.
- **MAE -** It gives you the average absolute difference between the predicted and actual values.
- **MSE -** It squares these differences before averaging
- **RMSE -** It's quite similar to Mean Squared Error (MSE), but the key difference is that RMSE takes the square root of the average of the squared differences between predicted and actual values.

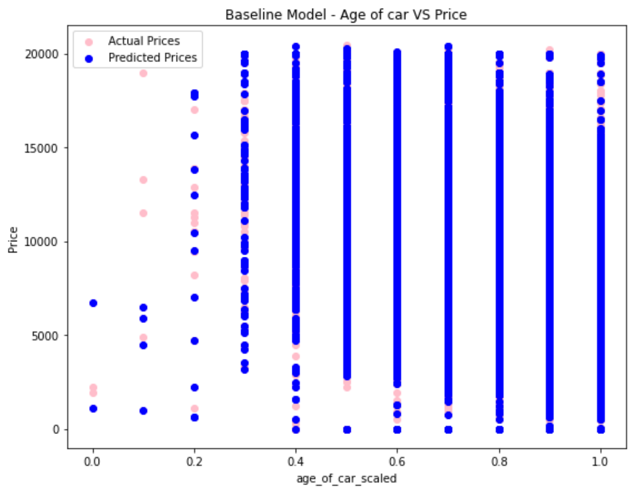
Decision Tree Regressor

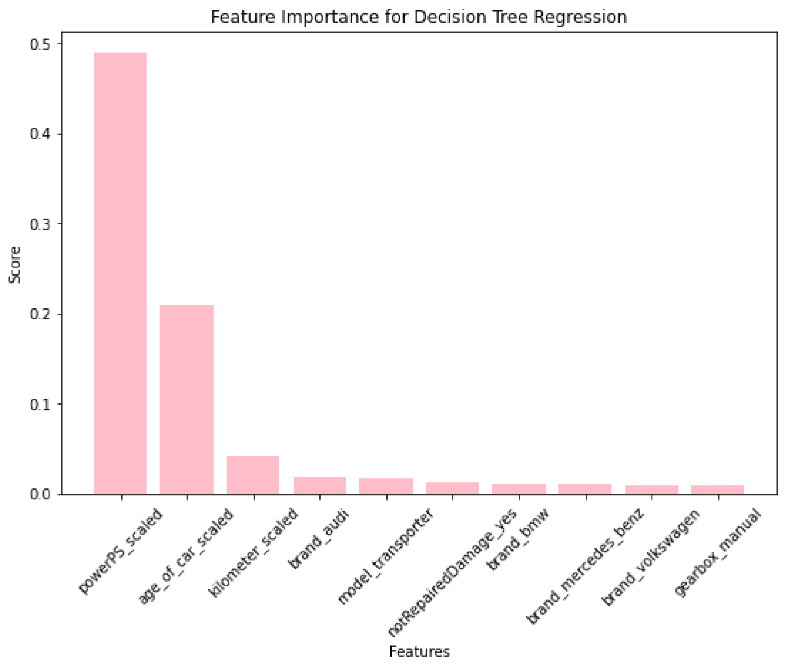
Short Description - The decision tree regressor creates a structure resembling a tree, with each node representing a decision based on a feature, and so on, until a prediction is made. This algorithm is flexible enough to manage interactions and non-linear relationships between features.

Evaluation Metrics: Some of the metrics for baseline model -

```
R2 score - Decision Tree Regression (Baseline): 0.72380
Mean squared error - Decision Tree Regression (Baseline): 5282202.79548
Mean absolute error - Decision Tree Regression (Baseline): 1512.30791
Root mean squared error - Decision Tree Regression (Baseline): 2298.30433
```

Visualization: Feature Importance for baseline model:





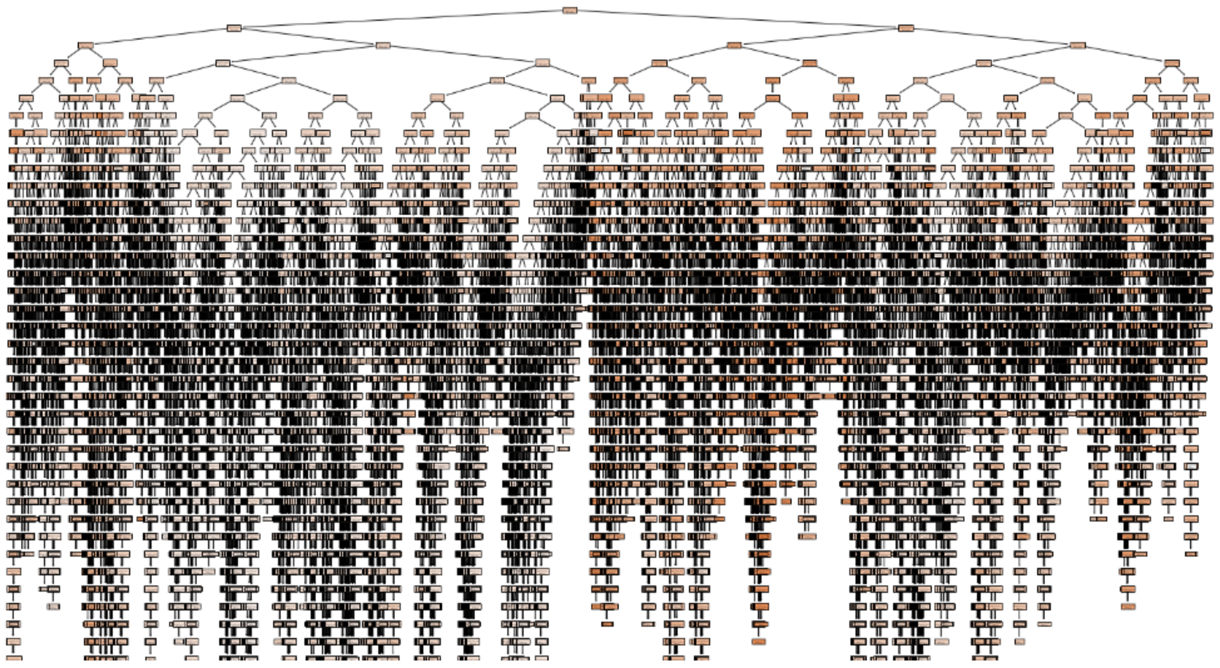
Random Forest Regressor

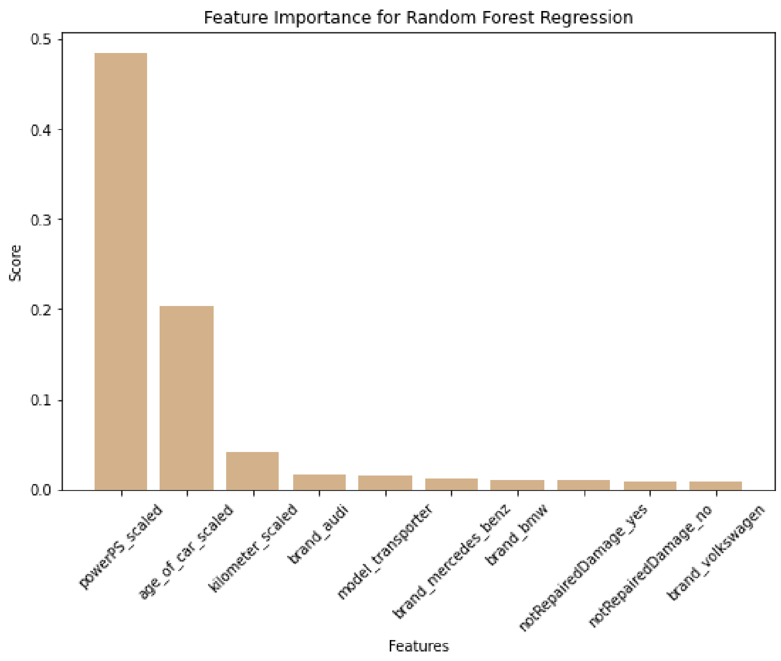
Short Description - Using the strength of multiple decision trees, the Random Forest Regressor is an ensemble learning algorithm that improves the accuracy of regression task predictions. It enhances overall performance and generalization by assembling a collection of different decision trees rather than depending on a single one.

Evaluation Metrics: Some of the metrics for baseline model -

```
R2 score - Random Forest Regression (Baseline): 0.78830
Mean squared error - Random Forest Regression (Baseline): 4048670.12597
Mean absolute error - Random Forest Regression (Baseline): 1360.23983
Root mean squared error - Random Forest Regression (Baseline): 2012.13074
```

Visualization: The trees are not very readable in the screenshot but we can zoom in the code.





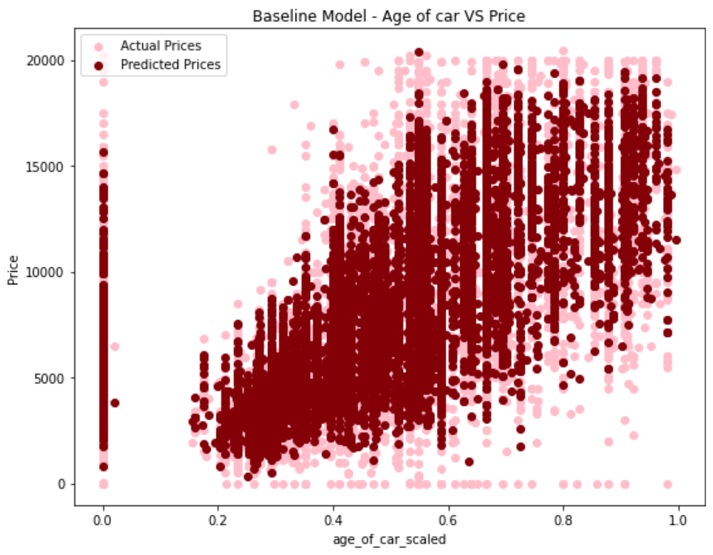
<u>XG Boost Regressor</u>

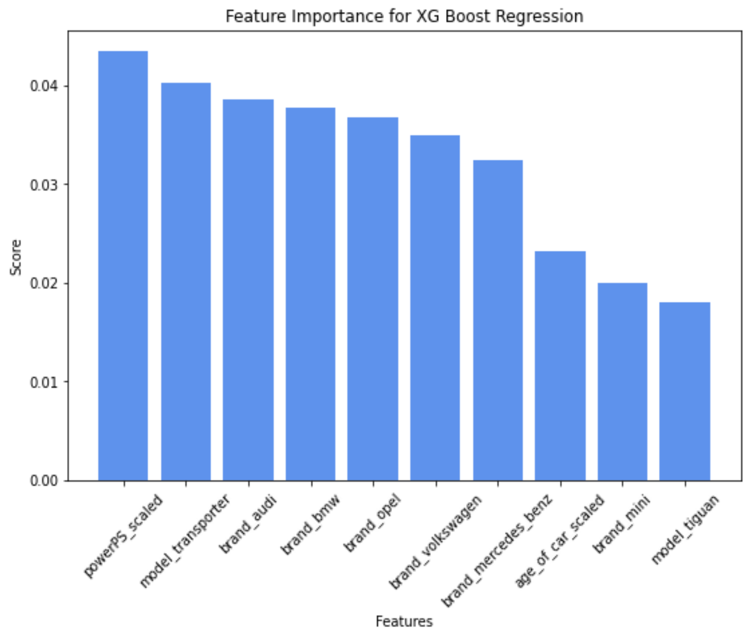
***Short Description* -** Extreme Gradient Boosting, or XGBoost, is a well-liked and incredibly effective machine learning algorithm for regression applications. It belongs to the gradient-boosting family of algorithms, which sequentially combines weak learners (usually decision trees) to create a strong predictive model.

Evaluation Metrics: Some of the metrics for baseline model -

```
R2 score - XG Boost Regression (Baseline): 0.79332
Mean squared error - XG Boost Regression (Baseline): 3952612.81208
Mean absolute error - XG Boost Regression (Baseline): 1373.65604
Root mean squared error - XG Boost Regression (Baseline): 1988.11791
```

Visualization:





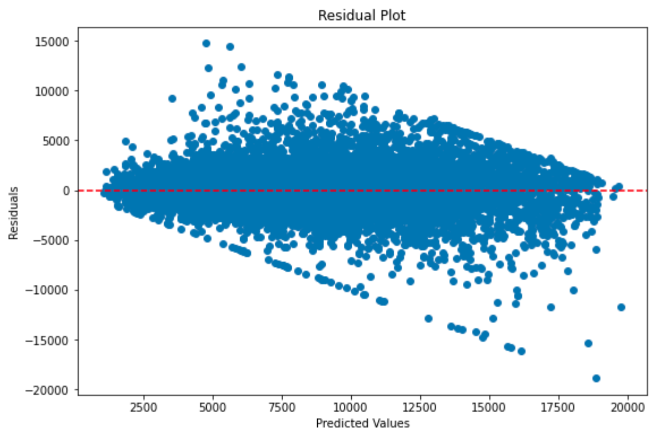
KNN Regressor

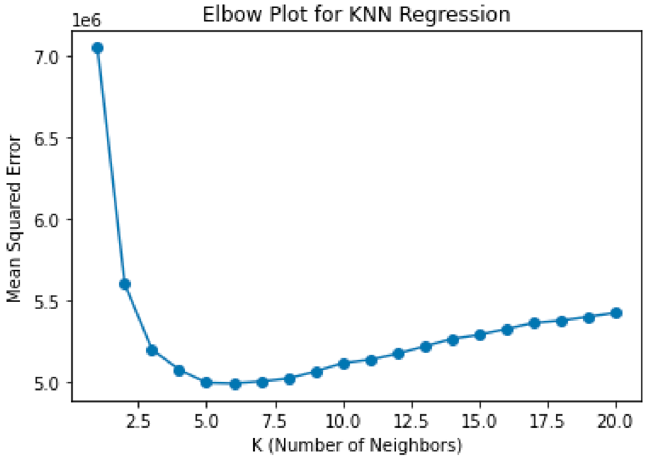
***Short Description* -** For regression tasks, the K-Nearest Neighbours (KNN) Regressor is a straightforward but powerful machine learning algorithm. KNN is a non-parametric algorithm, which means it doesn't make any strong assumptions about the underlying data distribution like parametric models do.

Evaluation Metrics: Some of the metrics for baseline model -

```
R2 score - Nearest-Neighbour Regression (Baseline): 0.73898
Mean squared error - Nearest-Neighbour Regression (Baseline): 4991855.80661
Mean absolute error - Nearest-Neighbour Regression (Baseline): 1548.47373
Root mean squared error - Nearest-Neighbour Regression (Baseline): 2234.24614
```

Visualization:





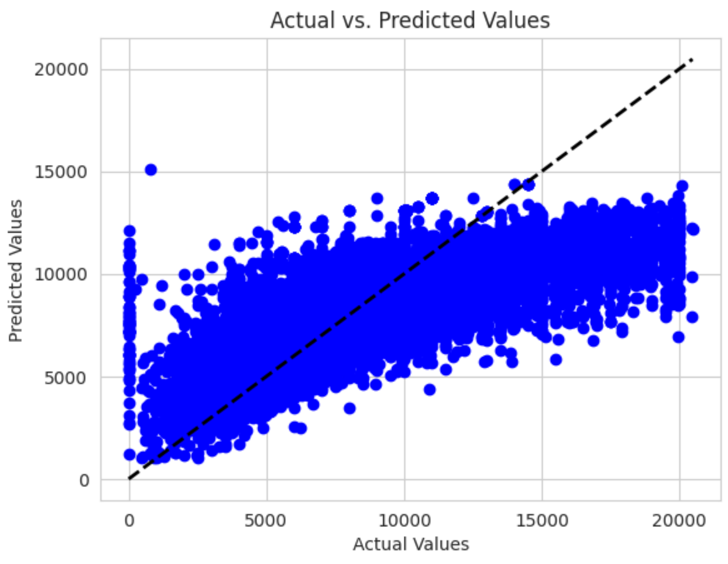
<u>Support Vector Regressor</u>

***Short Description* -** The Support Vector Regressor (SVR) is a machine learning algorithm used for regression tasks. Support vector machines, or SVMs, were initially created for classification purposes, and this is an extension of them. However, SVR has been modified to forecast continuous results.

Evaluation Metrics: Some of the metrics for baseline model -

```
Mean Squared Error: 8949816.524953583
Mean Absolute Error: 2231.4043291929356
R2 Score:0.5336725578793953
```

Visualization:



Lasso / Ridge Regressor

***Short Description* -** Regularisation techniques like Ridge Regression are used with linear regression models to improve generalization and avoid overfitting. The objective function of the linear regression includes an L2 regularisation term, which is added by Ridge Regression, also referred to as Tikhonov regularisation. The squared sum of the coefficients multiplied by a regularisation parameter (alpha) is the regularisation term.

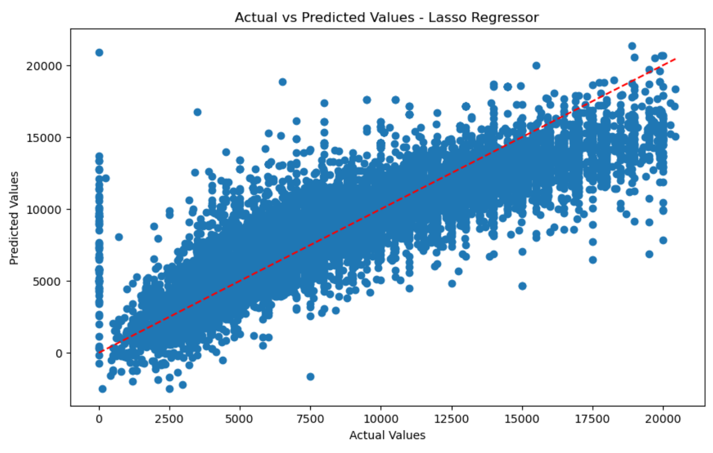
Evaluation Metrics: Some of the metrics for baseline Lasso model -

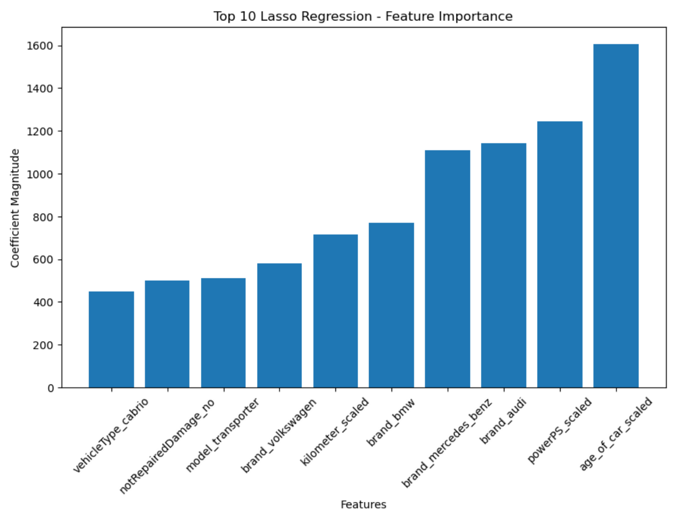
```
Mean Absolute Error: 1615.293239183182
Mean Squared Error: 4945786.816950195
RMSE for: 2223.912502089548
R-squared: 0.7452871063689195
```

Evaluation Metrics: Some of the metrics for baseline Ridge model -

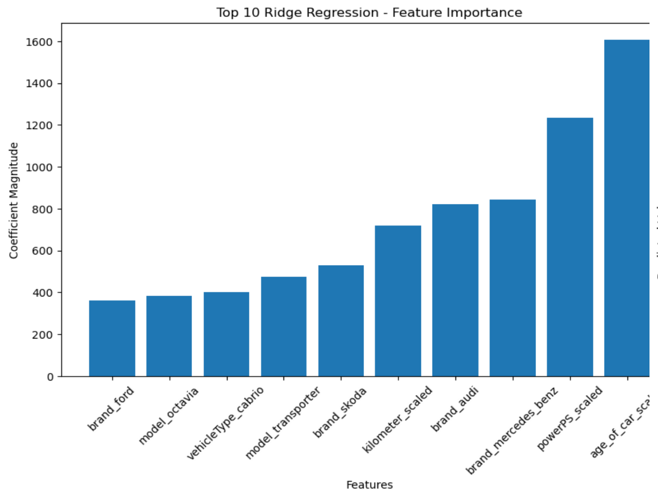
```
Mean Absolute Error: 1616.8172407508478
Mean Squared Error: 4951214.74980568
RMSE: 2225.13252409956
R-squared: 0.7450075624792967
```

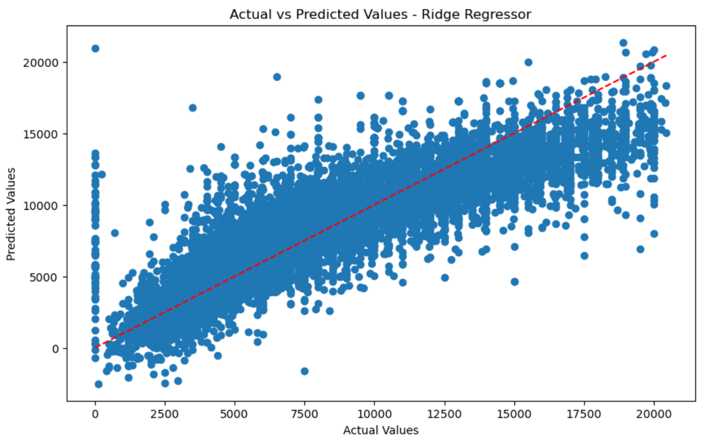
Visualization
Lasso Regressor -





Ridge Regressor -





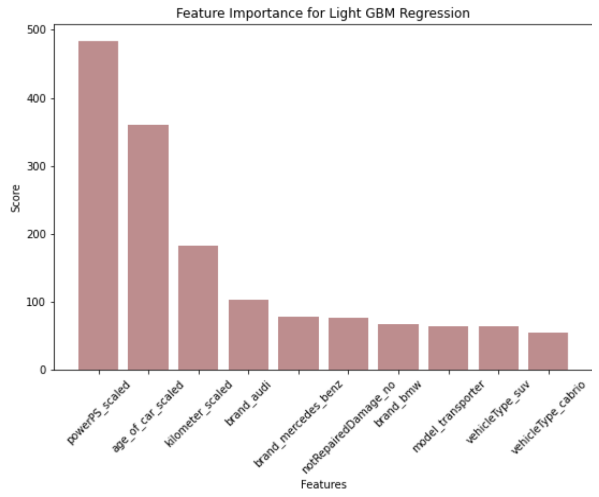
<u>Light GBM Regressor</u>

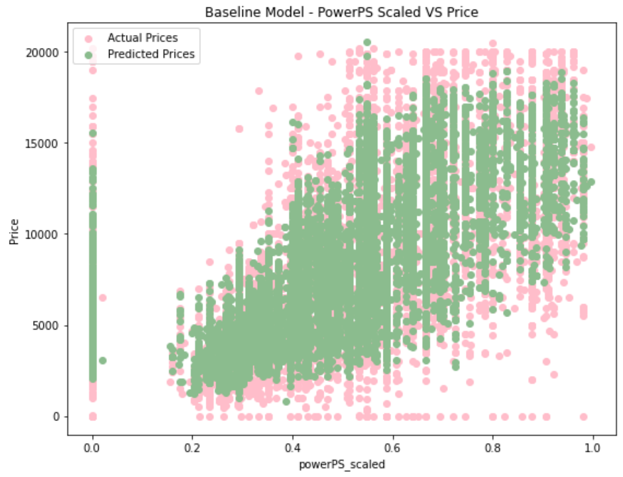
Short Description - The gradient boosting framework known as LightGBM, or Light Gradient Boosting Machine, prioritizes speed and efficiency. It works especially well with high-dimensional and large datasets.
Algorithms like XGBoost and AdaBoost, LightGBM are built on the gradient boosting framework. It increases prediction performance by building an ensemble of weak learners (usually decision trees) one after the other.

Evaluation Metrics: Some of the metrics for baseline model -

```
R2 score – Light GBM Regression (Baseline): 0.78662
Mean squared error – Light GBM Regression (Baseline): 4080735.83947
Mean absolute error – Light GBM Regression (Baseline): 1402.38410
Root mean squared error – Light GBM Regression (Baseline): 2020.08313
```

Visualization:

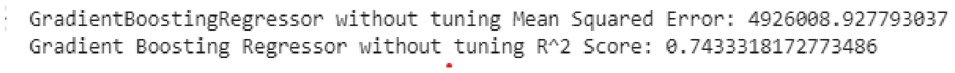




Gradient Boosting Regressor

***Short Description* -** Gradient Boosting Regressor is a machine learning ensemble algorithm that is used for regression tasks. It adds decision trees sequentially to correct errors, optimizing using gradient descent. It is well-known for its high predictive accuracy and robustness, and it is widely used in a variety of domains for predicting continuous numeric target variables

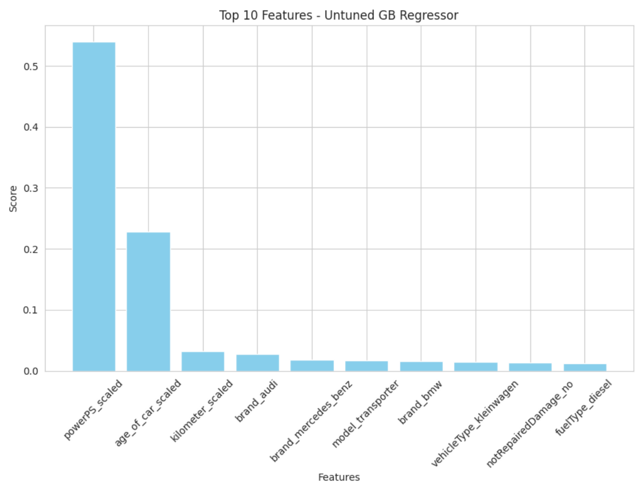
Evaluation Metrics: Some of the metrics for baseline model -

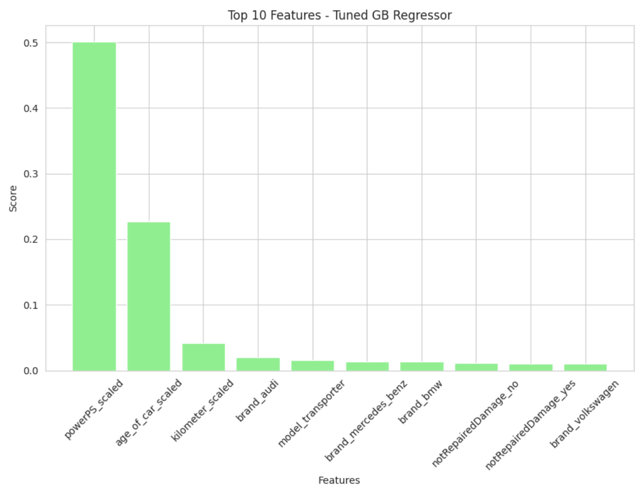


```
GradientBoostingRegressor without tuning Mean Squared Error: 4926008.927793037
Gradient Boosting Regressor without tuning R^2 Score: 0.7433318172773486
```

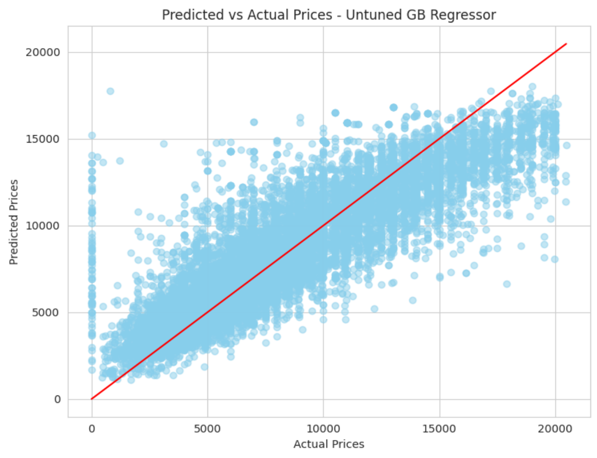
***Visualization*:**

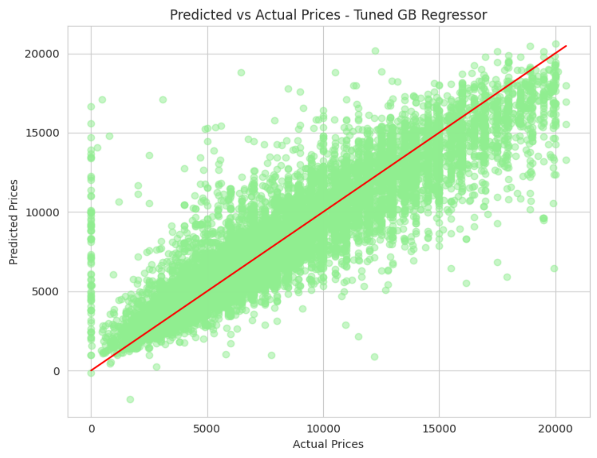
Feature Importance





Predicted Vs Actual Values

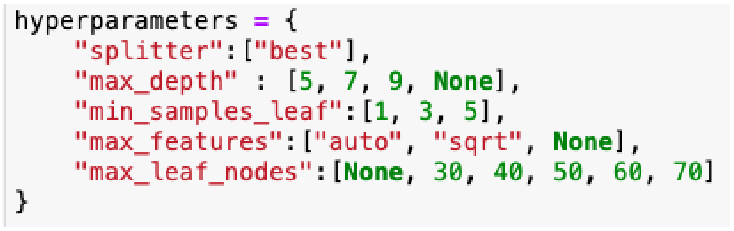




Part 2 - Explanation and Analysis

1. **<u>Decision Tree Regressor</u> -**
   Why did we choose this algorithm for this problem?
   Decision Tree Regressor algorithm has several advantages that make it well-suited for predicting used car prices:
   a. Decision trees are highly interpretable, which makes it simple to comprehend why the model produces particular predictions. This can be helpful for users who want to understand why a particular car is priced the way it is.
   b. Both the auction bid prices and the possible cost of repairs are intricate processes influenced by numerous variables. These kinds of intricate relationships are ideal for modeling with decision trees.
   c. The used automobile market may be extremely competitive, and prices can differ greatly based on the particular vehicle and how well it is maintained. Even for cars that are not well-represented in the training data, decision trees can pick up on these subtleties and produce reliable forecasts.

   How did we tune/train the model?
   We used Grid Search function to perform hyperparamater tuning. Below is the parameters grid we used to tune the Decision Tree Regressor



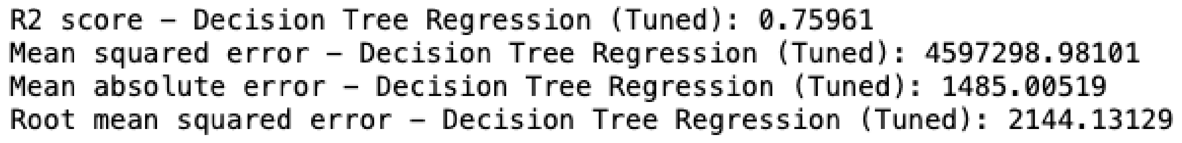
```
hyperparameters = {
    "splitter":["best"],
    "max_depth" : [5, 7, 9, None],
    "min_samples_leaf":[1, 3, 5],
    "max_features":["auto", "sqrt", None],
    "max_leaf_nodes":[None, 30, 40, 50, 60, 70]
}
```

   Best hyperparameters:

```
{
'max_depth': None,
   'max_features': None,
   'max_leaf_nodes': None,
   'min_samples_leaf': 5,
'splitter': 'best'
}
```

Effectiveness of the algorithm when applied to used cars dataset
Discussion of relevant metrics -

After tuning the hyperparameter:



```
R2 score - Decision Tree Regression (Tuned): 0.75961
Mean squared error - Decision Tree Regression (Tuned): 4597298.98101
Mean absolute error - Decision Tree Regression (Tuned): 1485.00519
Root mean squared error - Decision Tree Regression (Tuned): 2144.13129
```

Any intelligence we were able to gain from the application of the algorithm?
Decision Tree Regressor revealed key features

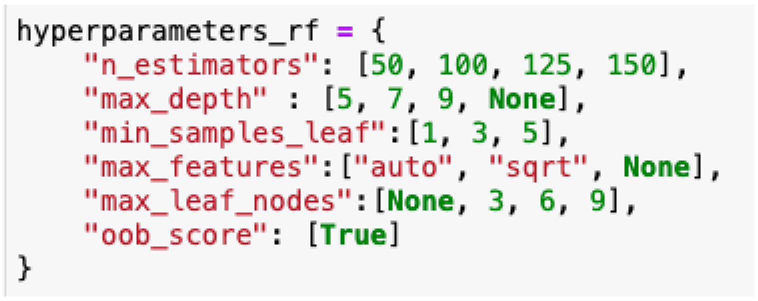
a. Driving car prices,
b. Showcasing non-linear relationships and subgroup variations,
c. Interpretability was a strength,
d. Visualizing the decision-making process.

2. Random Forest Regressor -

Why did we choose this algorithm for this problem?
Some specific reasons why Random Forest Regressors may be particularly well-suited for predicting used car prices:

a. Compared to Decision Trees, random forests are less likely to overfit the training set. This is because the noise in the data is reduced by Random Forests, which average the forecasts of several Decision Trees.
b. Random forests can be scaled to train on very large datasets. We have certainly very large dataset with a large number of columns which makes it a good choice to solve this problem.
c. On complicated datasets, Random Forests' capacity to discover complex relationships between the target variable and the characteristics is more than Decision Trees.
d. When it comes to data noise and outliers, Random Forests are comparatively resilient. This is crucial since used automobile data is frequently noisy.

How did we tune/train the model?
We used the Grid Search function to perform hyperparameter tuning. Below is the parameters grid we used to tune the Random Forest Regressor



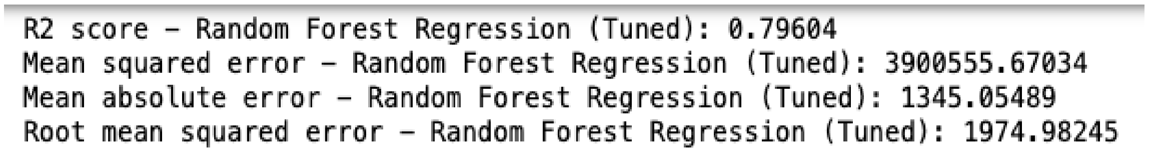
```
hyperparameters_rf = {
    "n_estimators": [50, 100, 125, 150],
    "max_depth" : [5, 7, 9, None],
    "min_samples_leaf":[1, 3, 5],
    "max_features":["auto", "sqrt", None],
    "max_leaf_nodes":[None, 3, 6, 9],
    "oob_score": [True]
}
```

Best hyperparameters:

{
'max_depth': None,
'max_features': 'sqrt',
'max_leaf_nodes': None,
'min_samples_leaf': 1,
'n_estimators': 150,
'oob_score': True
}

Effectiveness of the algorithm when applied to used cars dataset
Discussion of relevant metrics

After tuning the hyperparameters:



```
R2 score - Random Forest Regression (Tuned): 0.79604
Mean squared error - Random Forest Regression (Tuned): 3900555.67034
Mean absolute error - Random Forest Regression (Tuned): 1345.05489
Root mean squared error - Random Forest Regression (Tuned): 1974.98245
```

Any intelligence we were able to gain from the application of the algorithm?
Random Forest Regressor uncovered nuanced relationships in car price prediction -

a. Leveraging ensemble learning for improved accuracy,
b. Feature importance analysis highlighted crucial factors influencing prices,
c. Model interpretability was maintained, offering insights into decision-making,
d. Vigilance against overfitting

3. <u>XG Boost Regressor</u> -

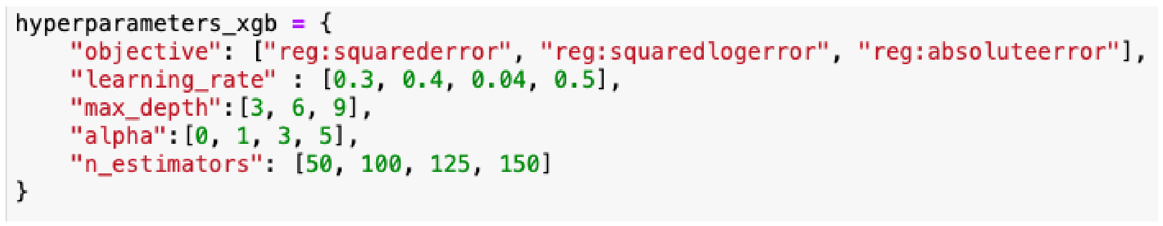
Why did we choose this algorithm for this problem?

a. XGBoost Regressors are a good choice for predicting used car prices because they are accurate, scalable, regularized, and interpretable.

b. Particularly on complicated datasets, XGBoost frequently outperforms decision trees and random forests in terms of accuracy. This is because XGBoost trains the model using a gradient-boosting approach, which lowers prediction variance and bias.

How did we tune/train the model?

We used Grid Search function to perform hyperparamater tuning. Below is the parameters grid we used to tune the XG Boost Regressor



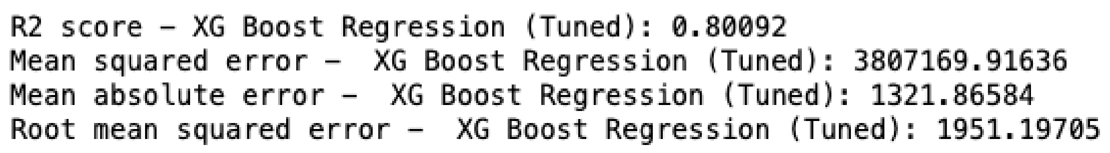
```
hyperparameters_xgb = {
    "objective": ["reg:squarederror", "reg:squaredlogerror", "reg:absoluteerror"],
    "learning_rate" : [0.3, 0.4, 0.04, 0.5],
    "max_depth":[3, 6, 9],
    "alpha":[0, 1, 3, 5],
    "n_estimators": [50, 100, 125, 150]
}
```

Best hyperparameters:

{
 'alpha': 0,
 'learning_rate': 0.3,
'max_depth': 9,
'n_estimators': 150,
'objective': 'reg:squarederror'
}

Effectiveness of the algorithm when applied to used cars dataset
Discussion of relevant metrics

After tuning the hyperparameters:



```
R2 score - XG Boost Regression (Tuned): 0.80092
Mean squared error -  XG Boost Regression (Tuned): 3807169.91636
Mean absolute error -  XG Boost Regression (Tuned): 1321.86584
Root mean squared error -  XG Boost Regression (Tuned): 1951.19705
```

Any intelligence we were able to gain from the application of the algorithm?

XGBoost Regressor showcased superior predictive performance in modeling used car prices-

a. Capturing intricate patterns and boosting overall accuracy,
b. Feature importance analysis unveiled critical factors steering price variations,
c. Despite its complexity, XGBoost maintained interpretability through feature insights.

d. Hyperparameter tuning and regularization techniques played a pivotal role in optimizing the model's efficiency.

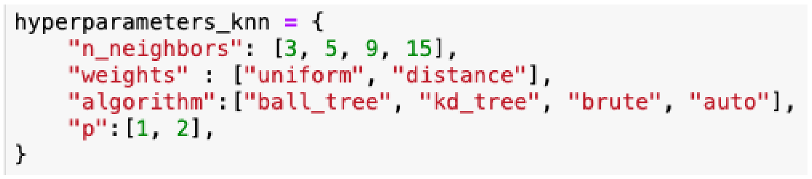
4. <u>K Nearest Neighbour Regressor</u> -

Why did we choose this algorithm for this problem?

a. The used car market is constantly changing, and new information is available all the time. Since KNN is a non-parametric method, it makes no assumptions on the data's distribution. Because of this, it works well for simulating dynamic markets, such as the used car market.
b. However, it is important to note that KNN can also be inaccurate on some datasets, especially when the number of features is large or the relationships between the features and the target variable are complex.
c. For this use case, it did not give good results and it was expected.

How did we tune/train the model?

We used Grid Search function to perform hyperparamater tuning. Below is the parameters grid we used to tune the KNN Regressor

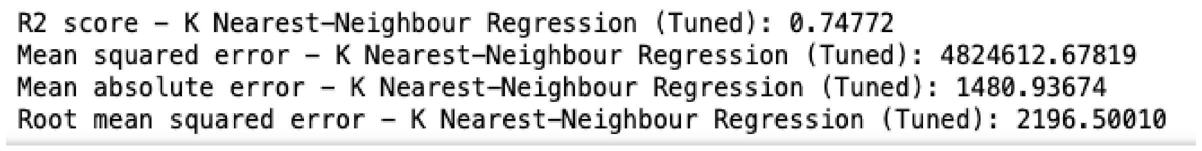


```
hyperparameters_knn = {
    "n_neighbors": [3, 5, 9, 15],
    "weights" : ["uniform", "distance"],
    "algorithm":["ball_tree", "kd_tree", "brute", "auto"],
    "p":[1, 2],
}
```

Best hyperparameters:
{
 'algorithm': 'brute',
'n_neighbors': 15,
'p': 1,
'weights': 'distance'
}

Effectiveness of the algorithm when applied to used cars dataset. Discussion of relevant metrics

After tuning the hyperparameters:



```
R2 score - K Nearest-Neighbour Regression (Tuned): 0.74772
Mean squared error - K Nearest-Neighbour Regression (Tuned): 4824612.67819
Mean absolute error - K Nearest-Neighbour Regression (Tuned): 1480.93674
Root mean squared error - K Nearest-Neighbour Regression (Tuned): 2196.50010
```

Any intelligence we were able to gain from the application of the algorithm?

KNN regression showed average result in predicting used car prices -

a. Emphasizing the impact of proximity-based neighbors.
b. The model's sensitivity to feature scales highlighted the importance of normalization for optimal performance.
c. Exploring alternative models like decision trees may complement KNN's findings for a more comprehensive understanding.

5. <u>Support Vector Regressor</u> -

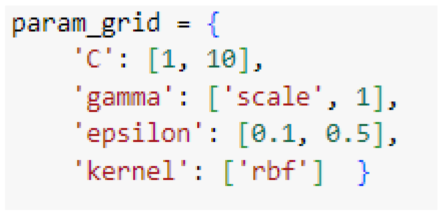
Why did we choose this algorithm for this problem?

a. Even in high-dimensional datasets, SVR can discover complex relationships between the features and the target variable. This is significant again because a variety of factors, including the car's make and model, year of manufacture, mileage, condition, and the state of the market, affect the used car market.
b. Although SVR can be computationally expensive to train on large datasets, it gives good performance if we tune hyperparameters well.

How did we tune/train the model?

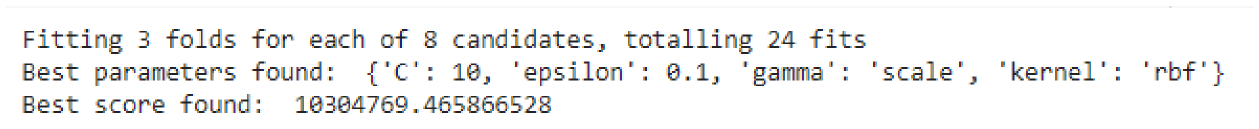
We tried the SVR model's performance through hyperparameter tuning using scikit-learn's GridSearchCV method. Key hyperparameters such as 'C,' 'gamma,' 'epsilon,' and 'kernel' were explored systematically to minimize mean squared error (MSE), using 'neg_mean_squared_error' as the scoring metric.

Hyperparameter grid:



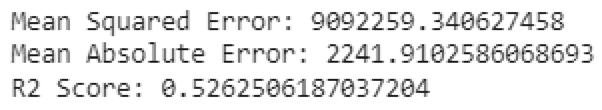
```
param_grid = {
    'C': [1, 10],
    'gamma': ['scale', 1],
    'epsilon': [0.1, 0.5],
    'kernel': ['rbf']  }
```

Best Parameters after the search



```
Fitting 3 folds for each of 8 candidates, totalling 24 fits
Best parameters found:  {'C': 10, 'epsilon': 0.1, 'gamma': 'scale', 'kernel': 'rbf'}
Best score found:  10304769.465866528
```

After tuning



```
Mean Squared Error: 9092259.340627458
Mean Absolute Error: 2241.9102586068693
R2 Score: 0.5262506187037204
```

Effectiveness of the algorithm when applied to used cars dataset

Any intelligence we were able to gain from the application of the algorithm?

We gained valuable insights into how different features affect used car prices by using the SVR algorithm. The optimized SVR model enabled us to understand which factors are important in determining used car pricing, providing valuable intelligence for pricing strategies or market analysis.

6. **Lasso Regressor -**

Why did we choose this algorithm for this problem?

a. Lasso regression, with its L1 regularization, excels in feature selection, promoting sparsity for simpler models with interpretability.
b. By shrinking coefficients towards zero, it handles multicollinearity and prevents overfitting in high-dimensional datasets.
c. The resulting sparse models are suitable for applications prioritizing simplicity.
d. Lasso ensures predictive accuracy while highlighting influential features through variable selection.

How did we tune/train the model?

a. LassoCV (Lasso Cross-Validation) is used for tuning the hyperparameter, specifically the regularization parameter (alpha), in a Lasso regression model.
b. The purpose of tuning is to find the optimal alpha that minimizes prediction errors and prevents overfitting.
c. Cross-validation is then performed using the specified range of alphas and identifies the optimal alpha that minimizes the mean squared error or another specified loss function

Effectiveness of the algorithm when applied to used cars dataset

- ***Discussion of relevant metrics***

  → **MSE:**

  Before Tuning: 4945786.82

  After Tuning: 4945857.82

  The MSE increased slightly after tuning, reflecting a small rise in the average squared prediction errors. Similar to MAE, the change is minimal.

  → **RMSE**:

Before Tuning: 2223.91
After Tuning: 2223.93
The RMSE increased by a very small amount after tuning. RMSE is a measure of the average magnitude of the errors, and in this case, the change is negligible.

→ **R-squared (R2-Score)**:
Before Tuning: 0.7453
After Tuning: 0.7453
The R-squared value remained virtually unchanged after tuning. R-squared represents the proportion of the variance in the dependent variable that is predictable from the independent variables, and in this case, the model's explanatory power did not improve significantly with tuning.

Any intelligence we were able to gain from the application of the algorithm?

a. The fact that the model's performance was relatively stable before and after tuning indicates that the initial Lasso model, without hyperparameter adjustments, was competent in capturing the patterns in the data.
b. The small changes in metrics post-tuning suggest that the chosen hyperparameters, or the default ones, were already in a reasonable range.
c. It's possible that the dataset and the problem at hand did not benefit significantly from further hyperparameter tuning in this particular case.
d. The consistency in metrics before and after tuning implies that the model is robust and that the choice of the regularization parameter did not drastically alter its predictive capabilities.

7. <u>Ridge Regressor</u> -

Why did we choose this algorithm for this problem?

- Ridge regression is a relatively simple algorithm to understand and implement. This makes it a good choice for beginners and for problems where interpretability is important but, in this case, the relationship between features and target value is not that simple.
- Also, Ridge regression models can be difficult to interpret, especially when the number of features is large.
- Therefore, it did not perform as well as other algorithms, such as Random Forests or XGBoost.

How did we tune/train the model?

- The model was tuned using RidgeCV, which stands for Ridge Cross-Validation. RidgeCV is a Ridge regression model with built-in cross-validation to find the optimal regularization parameter (alpha).
- It performs cross-validation across a range of alpha values and selects the one that minimizes the mean squared error or another specified criterion.

Effectiveness of the algorithm when applied to used cars dataset
Discussion of relevant metrics

→ **MAE**

Before Tuning: 1616.82
After Tuning: 1616.34

The MAE shows a slight improvement after tuning. The decrease indicates a small reduction in the average absolute error between the predicted and actual values on the test set

→ **MSE**

Before Tuning: 4951214.75
After Tuning: 4950982.14

The MSE also exhibits a minor reduction after tuning. This suggests a small improvement in the overall squared differences between the predicted and actual values on the test set.

→ **RMSE**

Before Tuning: 2225.13
After Tuning: 2225.08

The RMSE shows a negligible decrease after tuning. This indicates a slight improvement in the standard deviation of the errors, but the change is minimal.

→ **R2-Score**

Before Tuning: 0.7450
After Tuning: 0.7450

The R-squared value remains virtually unchanged after tuning. This indicates that the proportion of the variance in the dependent variable explained by the model remains consistent.

In summary, the tuning of the Ridge regression model has led to very minimal improvements in MAE, MSE, and RMSE, suggesting a slightly better fit to the test data. The R-squared value, representing the model's overall performance, remains largely unaffected by the tuning process. The changes in evaluation metrics are subtle, indicating that the default model and the tuned model perform similarly on the given data.

Any intelligence we were able to gain from the application of the algorithm?

→ The application of Ridge regression and its tuning provided insights into model interpretability and the optimal regularization strength (alpha).
→ The selected alpha value during tuning (92.37 in this case) represents the strength of the regularization applied.
→ The optimal alpha strikes a balance between fitting the training data well and avoiding overfitting.
→ The marginal improvements in evaluation metrics (MAE, MSE, RMSE) suggest a balanced trade-off between fitting the training data well and preventing overfitting.
→ The stability of R-squared indicates consistent explanation of variance.

Overall, the tuned model demonstrates slight enhancements in generalization to unseen data without significantly altering the initial model's performance.

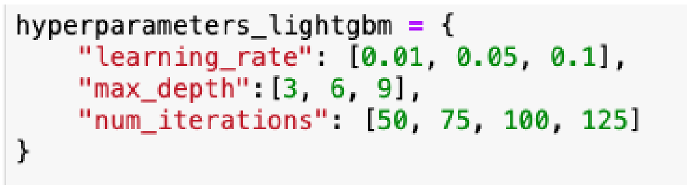
8. <u>Light GBM Regressor</u> -

Why did we choose this algorithm for this problem?

- For estimating the average cost of a used car, LightGBM is a good option, particularly if the dataset is large and contains a lot of columns. This algorithm is quick, accurate, scalable, and comparatively resistant to overfitting.
- However, it did not give the best performance out of all the algorithms used because LightGBM has several hyperparameters that need to be tuned to achieve optimal performance. This is a time-consuming process and we could find better values of hyperparameters than the default ones but not the best ones.

How did we tune/train the model?

We used the Grid Search function to perform hyperparameter tuning. Below is the parameters grid we used to tune the Light GBM Regressor -



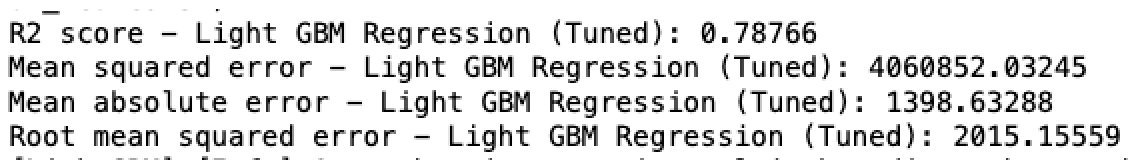
```
hyperparameters_lightgbm = {
    "learning_rate": [0.01, 0.05, 0.1],
    "max_depth":[3, 6, 9],
    "num_iterations": [50, 75, 100, 125]
}
```

Best hyperparameters:

{
    'learning_rate': 0.1,
    'max_depth': 9,
     'num_iterations': 125
}

Effectiveness of the algorithm when applied to used cars dataset
Discussion of relevant metrics

After tuning the hyperparameters:



```
R2 score - Light GBM Regression (Tuned): 0.78766
Mean squared error - Light GBM Regression (Tuned): 4060852.03245
Mean absolute error - Light GBM Regression (Tuned): 1398.63288
Root mean squared error - Light GBM Regression (Tuned): 2015.15559
```

Any intelligence we were able to gain from the application of the algorithm?

a. Light GBM exhibited robust performance in forecasting used car prices, leveraging gradient boosting for enhanced accuracy.
b. Feature importance analysis uncovered pivotal factors influencing pricing dynamics.
c. The algorithm's efficiency in handling large datasets and inherent regularization contributed to model interpretability.

9. <u>Gradient Boosting Regressor</u> -

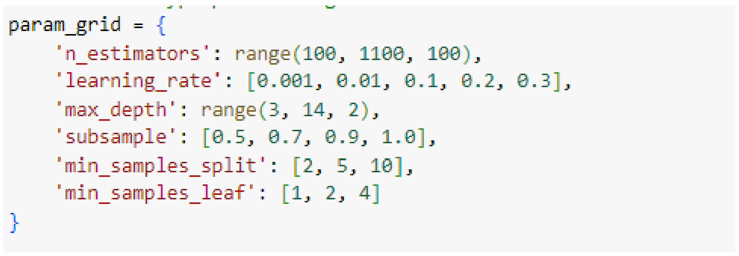
Why did we choose this algorithm for this problem?

- The GradientRegressor was selected due to its great performance in managing regression tasks that require continuous numerical predictions, such as predicting the cost of used cars.
- One of the model's strongest points is that it can reduce prediction errors, which is a key feature for datasets that have a variety of numerical features and continuous target variables. In this instance, GradientRegressor is especially adequate because it has scaled attributes like powerPS_scaled, kilometer_scaled, and age_of_car_scaled.

How did we tune/train the model?

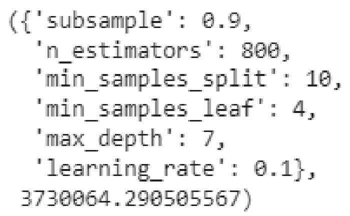
Using the Randomised Search technique, we utilized a hyperparameter grid to methodically examine a range of values for important parameters such as n_estimators, learning_rate, max_depth, etc. This method aids in identifying the best combination to raise model performance.

Effectiveness of the algorithm when applied to used cars dataset
Parameter grid used for RandomizedSearchCV tuning



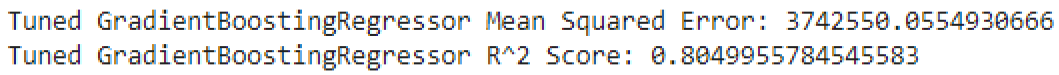
```
param_grid = {
    'n_estimators': range(100, 1100, 100),
    'learning_rate': [0.001, 0.01, 0.1, 0.2, 0.3],
    'max_depth': range(3, 14, 2),
    'subsample': [0.5, 0.7, 0.9, 1.0],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

Best parameter got from the search



```
({'subsample': 0.9,
  'n_estimators': 800,
  'min_samples_split': 10,
  'min_samples_leaf': 4,
  'max_depth': 7,
  'learning_rate': 0.1},
 3730064.290505567)
```

MSE and R2 score After tuning



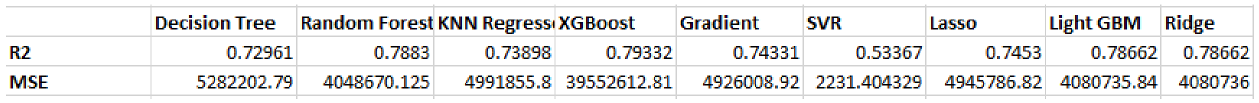
```
Tuned GradientBoostingRegressor Mean Squared Error: 3742550.0554930666
Tuned GradientBoostingRegressor R^2 Score: 0.8049955784545583
```

Any intelligence we were able to gain from the application of the algorithm?
GBR can provide insights into which features (like mileage, make, model, year, etc.) most significantly impact used car prices. The model likely offers valuable predictions about how auction bid prices and repair costs influence the final selling price.

Conclusion:

Performance of different models based on accuracy:

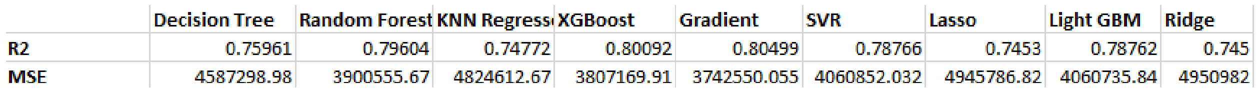
Baseline models:



	Decision Tree	Random Forest	KNN Regress	XGBoost	Gradient	SVR	Lasso	Light GBM	Ridge
R2	0.72961	0.7883	0.73898	0.79332	0.74331	0.53367	0.7453	0.78662	0.78662
MSE	5282202.79	4048670.125	4991855.8	39552612.81	4926008.92	2231.404329	4945786.82	4080735.84	4080736

Out of all the models -Gradient Boosting and XGBoost gave the best R square value and less error rates.

Tuned models:



	Decision Tree	Random Forest	KNN Regress	XGBoost	Gradient	SVR	Lasso	Light GBM	Ridge
R2	0.75961	0.79604	0.74772	0.80092	0.80499	0.78766	0.7453	0.78762	0.745
MSE	4587298.98	3900555.67	4824612.67	3807169.91	3742550.055	4060852.032	4945786.82	4060735.84	4950982

After tuning Gradient Boosting and XG Boost outperformed all other algorithms.

References:

- Gradient Boosting regression — scikit-learn 1.3.2 documentation
- Complete Machine Learning Guide to Parameter Tuning in Gradient Boosting (GBM) in Python
- Tuning of Adaboost with Computational Complexity | by Srijani Chaudhury | Medium
- 3 Methods to Tune Hyperparameters in Decision Trees - Inside Learning Machines
- sklearn.tree.DecisionTreeRegressor — scikit-learn 1.3.2 documentation
- How to tune a Decision Tree?. Hyperparameter tuning | by Mukesh Mithrakumar | Towards Data Science
- Tuning of Adaboost with Computational Complexity | by Srijani Chaudhury | Medium
- Comparison of kernel ridge regression and SVR
- Visualize a Decision Tree in 4 Ways with Scikit-Learn and Python | MLJAR
- Visualizing categorical data — seaborn 0.13.0 documentation
- Tuning of hyperparameters of SVR [closed]