Revolutionizing Used Car Pricing: A Game-Changing Tool for Informed Decisions in the Automotive Market

**PG-DBDA March 2023**

**Project Guide:**

**Parikshit Chaudhari**

**Submitted by:**

* **Mahesh Talmale**
* **Mayur Lingayat**
* **Mahadev Jadhav**
* **Mitali Girwalkar**
* **Niket Agasgekar**
* **Rajas Gondhali**

# INTRODUCTION

In the ever-evolving landscape of the automotive industry, one aspect has remained persistently challenging - used car pricing. We are proud to introduce a groundbreaking initiative that is poised to completely transform how we approach this crucial facet of the market - "Revolutionizing Used Car Pricing."

Gone are the days of uncertainty and ambiguity surrounding the true value of a used car. Our project represents a paradigm shift, a leap into an era where transparency, accuracy, and data-driven insights reign supreme. The aim is simple yet powerful: to empower individuals on both sides of the transaction - buyers and sellers - with the knowledge and confidence they need to make informed decisions.

At its core, "Revolutionizing Used Car Pricing" harnesses the potential of cutting-edge technology and advanced data analytics. By analyzing a diverse array of factors, from the vehicle's specifications and condition to market trends and historical data, our project generates comprehensive and precise valuations in real-time. This revolutionary tool transcends the limitations of guesswork, enabling users to make decisions grounded in solid information.

However, this project isn't just about numbers and algorithms. It's about empowering individuals with the ability to navigate the complex used car market with assurance. Buyers can explore options armed with accurate pricing insights, ensuring that they receive the best value for their investment. Sellers, on the other hand, can confidently set prices that reflect the true worth of their vehicles, thus fostering trust and satisfaction.

"Revolutionizing Used Car Pricing" isn't just a project; it's a movement towards a fairer, more transparent automotive market. It's an invitation to all stakeholders - from industry enthusiasts to everyday drivers - to join us in this transformative journey. Together, we're reshaping the way we buy and sell used cars, ushering in an era where clarity prevails, transactions thrive, and the automotive experience is elevated for all.

Welcome to the future of used car pricing. Welcome to a new standard of automotive transactions. Welcome to "Revolutionizing Used Car Pricing."

Welcome to the "Driving the Future of Vehicle Pricing: A Machine Learning Approach" project. In this project, our goal is to develop a machine learning model that can accurately predict the price of a vehicle based on various factors such as make, model, year, mileage, and condition. The automobile industry is one of the largest and most competitive industries in the world, with millions of vehicles being sold each year. The price of a vehicle can have a significant impact on a consumer's purchasing decision, and it is crucial for both buyers and sellers to have an understanding of the market value of a vehicle. Through the use of machine learning algorithms and data analysis, we aim to provide a reliable and robust model that can assist in determining the fair market value of a vehicle. Join us on this exciting journey as we delve into the world of vehicle price prediction.

Models used in this project:

1. Linear Regression - Base Model
2. Random Forest - Bagging Model
3. LightGBM - Boosting Model
4. XGBoost - Extreme Gradient Boosting Model

# PROBLEM STATEMENT

The vehicle price prediction project aims to address a crucial business problem in the automotive industry: accurately determining the fair market value of a vehicle. The automobile market is highly competitive and the price of a vehicle is a key factor in a consumer's purchasing decision. For both buyers and sellers, having an accurate understanding of a vehicle's market value is essential in making informed decisions.

However, determining the fair market value of a vehicle can be a complex and challenging task, as it depends on a variety of factors such as make, model, year, mileage, condition, and location. As a result, there is a growing need for a reliable and objective tool that can assist in determining the market value of a vehicle.

The vehicle price prediction project aims to provide such a tool by utilizing advanced machine learning algorithms and data analysis techniques to accurately predict the price of a vehicle based on relevant factors. This will not only help consumers make more informed purchasing decisions but also provide valuable insights for dealers, manufacturers, and other stakeholders in the automotive industry.

our task is to build a predictive model that can accurately predict the price of a used car based on its features. This model will help potential buyers and sellers to estimate the appropriate market value of a used car, allowing them to make more informed decisions during transactions.

# LITERATURE SURVEY

### Introduction

This section surveys existing literature related to used car pricing prediction and machine learning applications. It delves into various machine learning algorithms commonly used for regression tasks, outlining their strengths and weaknesses in the context of pricing prediction. The review also discusses the role of data sources, feature selection, and model evaluation metrics in enhancing prediction accuracy.

In the dynamic realm of automotive commerce, the process of pricing used cars has long been an intricate challenge, prompting a multitude of researchers and experts to delve into the subject. This literature survey delves into the diverse landscape of research, analyses, and innovations that have collectively contributed to the vision of "Revolutionizing Used Car Pricing."

## EXISTING METHODS

### Used Car Prediction Using Machine Learning

The research work done by Ashish Chandak, Information Technology, Shri Ramdeobaba College of Engineering, Rashtra Sant Tukadoji Maharaj Nagpur University, Nagpur, India.

In this chapter, we discuss various applications and methods which inspired us to build our project. We did a background survey regarding the basic ideas of our project and used those ideas for the collection of information like the technological stack, algorithms, and shortcomings of our project which led us to build a better project.

In this chapter, we discuss various algorithms and the required dataset that were implemented to build this module. A dataset containing more than 3 lakh rows will be used for training the model. Attributes such as kilometers traveled, year of registration, fuel type and fiscal power determine the worth of an automobile.

# LIBRARIES USED

### 4 Pandas Library: Data Manipulation and Analysis in Python

Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data.

* **Data Loading and Saving:** Pandas provides functions to read data from various file formats, including CSV, Excel, SQL databases, and JSON. It also supports writing data to these formats.
* **Data Cleaning and Transformation:** Pandas offers a wide range of methods for cleaning and transforming data, such as handling missing values, removing duplicates, changing data types, and applying functions across rows or columns.
* **Indexing and Selection:** Pandas allows for intuitive indexing and slicing of data using labels or positions. This makes it simple to extract specific rows, columns, or elements from a Data Frame.
* **Grouping and Aggregation:** Pandas facilitates grouping data based on one or more columns and then performing various aggregation functions like sum, mean, count, etc., on the grouped data.
* **Visualization:** While Pandas itself doesn't provide visualization tools, it seamlessly integrates with libraries like Matplotlib and Seaborn to create informative plots and visualizations.
* **Performance:** Pandas is built to be fast and efficient, especially when handling large datasets. However, for even greater performance, techniques like vectorized operations and the .apply() method should be utilized.
* Pandas has a large and active community of users, which means there's a wealth of resources available for learning and troubleshooting. It also integrates well with other data analysis and visualization libraries like NumPy, Matplotlib, and Seaborn.
* **Conclusion:**

Pandas is an essential tool for data manipulation and analysis in Python. Its Data Frame and Series structures, along with its wide range of functions, make it a go-to choice for handling various data tasks, from cleaning and preprocessing to exploration and transformation. Whether you're a beginner or an experienced data scientist, Pandas is a must-know library for efficient data handling.

### 4.1 NumPy Library: Fundamental Package for Scientific Computing in Python

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is open-source software. It contains various features including these important ones:

* **Multi-dimensional Arrays:** NumPy introduces the ND array (n-dimensional array) as its primary data structure. It's capable of representing arrays of various dimensions (1D, 2D, 3D, etc.) and supports a wide range of data types.
* **Efficient Element-Wise Operations:** NumPy allows you to perform element-wise mathematical operations on arrays, which are much faster than equivalent operations on regular Python lists. These operations are executed in compiled C code, optimizing performance.
* **Indexing and Slicing:** Similar to Python lists, NumPy arrays support indexing and slicing to access elements or sub-arrays. This makes it easy to extract and manipulate data.
* **Broadcasting:** NumPy enables broadcasting, a powerful feature that allows arrays with different shapes to be used in arithmetic operations. Broadcasting automatically applies the operation element-wise to match the dimensions.
* **Array Creation and Manipulation:** NumPy offers functions to create arrays with specific shapes and values. It allows reshaping, stacking, splitting, and transposing arrays, providing flexibility in array manipulation.
* **Linear Algebra Operations:** NumPy includes functions for linear algebra operations, such as matrix multiplication, eigenvalue decomposition, singular value decomposition, and more.
* **Random Number Generation:** NumPy provides tools for generating random numbers from various probability distributions, which is essential for simulations and statistical analysis.
* **Mathematical and Statistical Functions:** NumPy supports a wide range of mathematical and statistical functions, including trigonometry, logarithms, mean, median, variance, and more.
* **Memory Efficiency:** NumPy's memory layout is efficient, allowing for better utilization of memory and optimized storage of arrays.
* **Performance Considerations:** NumPy's array operations are implemented in C, making them considerably faster than equivalent operations performed using Python loops. This performance improvement is crucial for handling large datasets and complex computations efficiently.
* **Conclusion:**

NumPy is a cornerstone of the Python data science toolkit. Its efficient array operations, mathematical functions, and integration with other libraries make it an essential tool for numerical computations, data manipulation, and scientific research. Whether you're performing simple arithmetic or complex simulations, NumPy's capabilities are indispensable for any data scientist or researcher working in Python.

### 4.2 Matplotlib: Data Visualization in Python

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible. . It contains various features including these important ones:

* **Publication-Quality Plots:** Matplotlib allows you to create high-quality plots suitable for publications, presentations, and reports. Its output can be saved in various formats, including PNG, PDF, SVG, and more.
* **Support for Various Plot Types:** Matplotlib supports a wide range of plot types, including line plots, scatter plots, bar plots, histograms, pie charts, heatmaps, contour plots, and more.
* **Customization and Styling:** Matplotlib provides extensive control over plot styling, including colors, line styles, markers, fonts, labels, and annotations. This allows you to tailor visualizations to match your preferences or adhere to specific design guidelines.
* **Subplots and Layouts:** You can create multi-panel plots using subplots, allowing you to display multiple plots within a single figure. Matplotlib provides flexibility in arranging and aligning subplots.
* **Text and Annotation:** Matplotlib supports adding text, labels, and annotations to your plots. This is useful for providing context, explanations, or additional information within visualizations.
* **Interactive Plotting:** While Matplotlib primarily produces static plots, it can be combined with libraries like Jupyter Notebook and Python to create interactive plots that allow for exploration and data interaction.
* **Integration with Pandas:** Matplotlib integrates well with Pandas Data Frames, making it easy to visualize data directly from Pandas structures.
* **3D Plotting:** Matplotlib supports three-dimensional plotting, which is useful for visualizing data in a three-dimensional space, such as 3D scatter plots and surface plots.
* **Animations:** Matplotlib can be used to create animations by sequentially updating the data or plot elements over time.
* **Community and Ecosystem:** Matplotlib has a large and active community, providing abundant resources, tutorials, and examples for learning and using the library effectively. Additionally, Matplotlib serves as the foundation for other visualization libraries like Seaborn and Plotly, which build on top of its functionalities.
* **Conclusion:**

Matplotlib is a versatile and powerful library that enables you to create a wide range of visualizations to communicate data insights effectively. Whether you're a data scientist, researcher, or anyone working with data, Matplotlib's capabilities are essential for conveying information visually and making data-driven decision

**4.3 Seaborn:** **Statistical Data Visualization in Python**

In data visualization and exploratory data analysis, seaborn is a powerful Python library that provides an aesthetically pleasing and informative interface for creating a variety of statistical graphics. By importing it with the alias sns, you gain access to a plethora of functions and tools that enable you to quickly generate visually appealing plots and charts.

* **Visualization Library:** Seaborn is a Python data visualization library built on top of Matplotlib. It provides a high-level interface for creating informative and attractive statistical graphics.
* **Statistical Plots:** Seaborn specializes in creating statistical visualizations that allow users to easily explore relationships and patterns in their data. It offers functions to create scatter plots, bar plots, histograms, box plots, and more.
* **Data Integration:** Seaborn works seamlessly with Pandas Data Frames, making it easy to visualize data directly from data structures commonly used in data analysis.
* **Beautiful Styles:** Seaborn comes with several built-in themes and color palettes that enhance the aesthetics of your plots. The library's default styles are often more visually appealing than Matplotlib's defaults.
* **Statistical Enhancement:** Seaborn includes functions for adding statistical annotations to plots, such as linear regression lines, confidence intervals, and correlation coefficients.
* **Categorical Data:** It excels at creating plots for categorical data, like bar plots, violin plots, and box plots, allowing you to visualize the distribution of variables across categories.
* **Matrix Plots:** Seaborn offers matrix-based visualizations like heatmap and cluster map, which are useful for visualizing correlations and patterns in large datasets.
* **Time Series Visualization**: It provides support for creating time series plots, facilitating the analysis of data over time.
* **Ease of Use:** Seaborn simplifies the process of creating complex visualizations with concise and readable code. Its high-level API often requires fewer lines of code compared to using Matplotlib directly.
* **Community and Documentation:** Seaborn has an active community and comprehensive documentation, making it easier to find help, examples, and resources for learning and using the library.
* **Conclusion:**

Seaborn is a valuable tool for creating informative and visually appealing visualizations with ease. Its focus on statistical graphics, integration with Pandas, and straightforward syntax make it a popular choice for data analysts and scientists who want to quickly generate meaningful plots to understand their data and communicate their findings effectively.

**4.4 Data Cleaning with PySpark:**

PySpark provides powerful tools for cleaning and preprocessing large datasets efficiently.

**SparkSession and DataFrames:**

Use SparkSession to create DataFrames, representing structured data, and leverage its optimization capabilities.

**Handling Missing Values:**

PySpark offers functions like fillna, dropna, and replace to manage missing values in DataFrames.

**String Manipulation:**

Utilize functions like regexp\_replace for pattern-based text replacements, and split for splitting text data.

**Type Conversion and Casts:**

Change column data types using cast to ensure consistent and accurate data representation.

**Outlier Detection and Removal:**

Apply statistical functions and filtering techniques to identify and handle outliers in numerical data.

**Deduplication:**

Use functions such as dropDuplicates to remove duplicate rows from DataFrames.

**Handling Inconsistent Data:**

Employ functions like when and regexp\_replace to transform inconsistent values into a common format.

**Data Validation and Filtering:**

Apply filters using filter to remove rows that don't meet specific criteria, ensuring data quality.

**Custom Data Cleaning Logic:**

Leverage Python's capabilities to implement custom cleaning logic using UDFs (User-Defined Functions).

**Scaling with Cluster Computing:**

PySpark's distributed architecture enables seamless scaling of data cleaning operations across clusters.

**Chaining Transformation Steps:**

Chain multiple cleaning operations together using DataFrame transformations to create a robust cleaning pipeline.

**Data Profiling:**

Utilize summary statistics and profiling functions to gain insights into data distribution and identify potential issues.

**Handling Categorical Data:**

Convert categorical variables to numerical form using techniques like one-hot encoding or StringIndexer.

**Advanced Cleaning Techniques:**

Leverage PySpark's advanced functions for data transformations, such as window functions for context-aware cleaning.

**Data Quality Assurance:**

Use PySpark's validation and verification features to ensure data quality after cleaning.

**Documentation and Tracking:**

Comment and document data cleaning steps for reproducibility and to keep track of changes.

PySpark's versatile tools and functions make it a robust choice for data cleaning tasks, especially when dealing with large and complex datasets.

**PySpark Libraries:**

**pyspark.sql:** This module in PySpark facilitates structured data processing using DataFrames and SQL-like queries.

**SparkSession:** The entry point to Spark, enabling DataFrame creation, SQL operations, and configuration management.

**pyspark.sql.types:** Defines column data types for DataFrames, including StringType, FloatType, and IntegerType.

**pyspark.sql.functions:** Offers various built-in functions for DataFrame transformations and aggregations, like count, split, mean, median, and more.

**Window:** Provides tools for windowed computations and ranking within DataFrames.

**expr:** Enables evaluation of SQL expressions on DataFrames, involving columns and functions.

**avg, length, log:** Functions to compute average, string length, and natural logarithm on DataFrame columns.

**Bucketizer:** Part of Spark MLlib, used for converting continuous data into categorical data by specifying buckets.

These components collectively empower efficient, large-scale data manipulation and analysis in Apache Spark.

# 5. METHODOLOGY - MODEL BUILDING

### 5.1 Linear Regression – Base Model:

### Linear Regression was chosen as the first model due to its simplicity and comparatively small

### training time. The features, without any feature mapping, were used directly as the feature vectors.

### No regularization was used since the results clearly showed low variance.

### Simplicity and Transparency:

### Linear Regression is a simple yet powerful method that assumes a linear relationship between the independent variables and the dependent variable. Its transparency allows for easy interpretation of the coefficients and their impact on the target variable.

### 

### Quick Initial Insights:

### Using Linear Regression as a base model helps provide a quick overview of how individual features relate to the target variable. It's a valuable starting point to identify the direction and approximate strength of relationships.

### Feature Importance:

### In Linear Regression, the coefficients of the features indicate the change in the target variable for a unit change in the corresponding feature, assuming all other features are constant. This highlights the relative importance of each feature in predicting the target.

### Feature Scaling:

### Linear Regression is sensitive to the scale of features. Properly scaling features (e.g., using standardization) can improve the model's performance and convergence.

### Model Evaluation:

### Evaluate the Linear Regression model's performance using metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). These metrics quantify how well the model fits the data and predicts target values.

### Limitations:

### Linear Regression assumes a linear relationship, which might not capture more complex interactions present in the data. It may not perform well with non-linear relationships without proper feature engineering.

### Future Steps:

### Depending on the results and insights gained from the Linear Regression model, you might consider exploring more sophisticated models to capture non-linear relationships and interactions in the data.

### Incorporating Linear Regression as the base model shows a structured and logical approach to our project. It allows you to establish a foundation for comparison with more advanced models and aids in understanding the relationships between features and the target variable.

**5.2 LightGBM – Gradient Boosting Model:**

Light GBM is another gradient boosting-based framework which is gaining popularity due it higher

speed and accuracy compared to XGBoost or the original gradient boosting method. Similar to XGBoost, this LightGBM has a leaf-wise tree growth instead of a level-wise approach resulting in higher loss reduction. This framework can also handle categorical features, thus eliminating the need to one hot vectorize them and in turn, reducing memory usage. Make, Model and State and cities were declared as categorical features. The algorithm was run at tree depths in multiples of 12 and was run on all cores in parallel.

* **Gradient Boosting Technique:**

LightGBM is an advanced gradient boosting framework that focuses on performance and efficiency. Gradient boosting is an ensemble learning method that builds a strong predictive model by combining the predictions of multiple weak models, usually decision trees.

* **LightGBM Advantages:**

LightGBM stands out for its efficiency and speed due to features like histogram-based techniques for splitting and handling categorical features. It's designed to handle large datasets and high-dimensional feature spaces.

* **Gradient Boosting Process:**

LightGBM builds a series of decision trees iteratively. Each subsequent tree tries to correct the errors made by the ensemble of previous trees. This process leads to a model that is capable of capturing complex relationships in the data.

* **Leaf-Wise Growth:**

Unlike traditional depth-first tree growth, LightGBM uses a leaf-wise growth strategy. It selects the leaf with the maximum delta loss to split, allowing it to potentially build deeper trees and improve model performance.

* **Feature Importance:**

LightGBM provides feature importance scores based on how much each feature contributes to reducing the loss function during training. This helps in identifying which features are most influential in making predictions.

* **Handling Categorical Features:**

LightGBM can directly handle categorical features by converting them into integers and using techniques like categorical split. This is a convenient feature as handling categorical data is often a challenge in machine learning.

* **Model Evaluation:**

Evaluate your LightGBM model using appropriate regression metrics similar to the other models. This will allow you to compare its performance with the previously discussed Linear Regression.

Incorporating LightGBM into our project showcases your willingness to explore sophisticated techniques for predictive modeling. Its ability to handle large datasets, handle categorical features, and capture complex patterns makes it a valuable asset for enhancing the quality of our project's outcomes.

**5.3 XGBoost - Extreme Gradient Boosting Model:**

Extreme Gradient Boosting or XGBoost is one of the most popular machine learning models in current times. XGBoost is quite similar at the core to the original gradient boosting algorithm but features many additive features that significantly improve its performance such as built in support for regularization, parallel processing as well as giving additional hyperparameters to tune such as tree pruning, sub sampling and number of decision trees. A maximum depth of was used and the algorithm was run on all cores in parallel.

* **Enhanced Gradient Boosting:**

XGBoost is an advanced implementation of gradient boosting that has gained popularity due to its scalability, speed, and performance. Like other gradient boosting techniques, it builds an ensemble of weak learners to create a strong predictive model.

* **Regularization and Complexity Control:**

XGBoost introduces regularization techniques that help prevent overfitting, making it more robust and less prone to capturing noise in the data. Parameters like "max\_depth" control the depth of individual trees, reducing their complexity.

* **Objective Functions:**

XGBoost allows you to specify various objective functions based on your problem type, such as regression, classification, and ranking. This flexibility ensures that the boosting process is aligned with your specific goals.

* **Feature Importance:**

Similar to LightGBM, XGBoost provides feature importance scores that quantify the contribution of each feature to the model's predictions. These scores help in identifying influential features.

* **Early Stopping:**

XGBoost supports early stopping, where the training process can be halted if the model's performance on a validation set starts deteriorating. This helps prevent overfitting and saves training time.

* **Model Evaluation and Ensemble Strategies:**

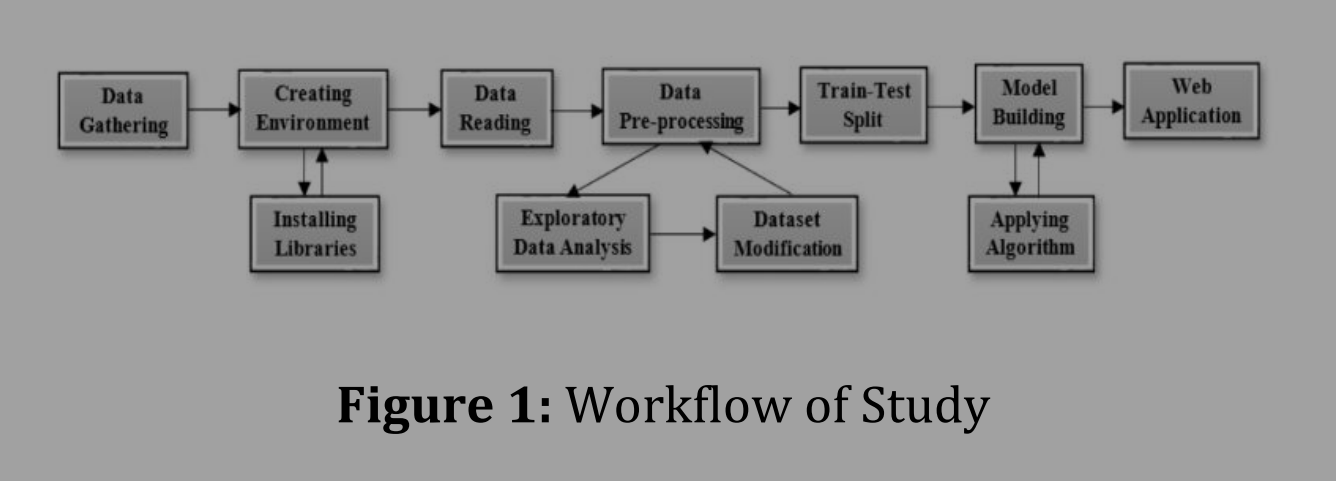
Evaluate your XGBoost model's performance using the same regression metrics used for the previous models. Additionally, consider creating an ensemble that combines predictions from XGBoost with predictions from other models like LightGBM and Random Forest for potentially improved performance.

* **Resource Considerations:**

XGBoost can be memory-intensive and might require substantial computational resources, especially when dealing with large datasets or complex configurations. Incorporating XGBoost into our project demonstrates your willingness to explore advanced techniques for predictive modeling. Its regularization capabilities, feature importance analysis, and optimization options make it a potent tool for improving the predictive accuracy of our project's outcomes.

## FLOW CHART

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows.



## PROJECT ARCHITECTURE

### Preprocessing of data



Fig: Pre-processing of data

### Overall Architecture

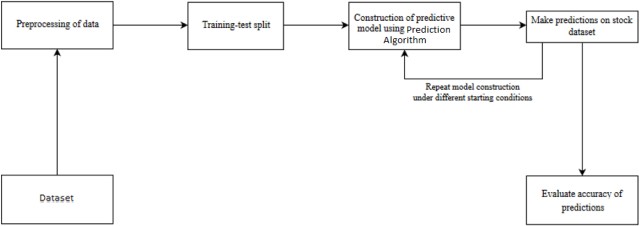


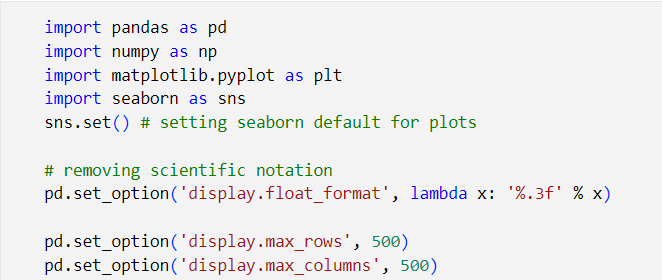
Fig: Overall Architecture

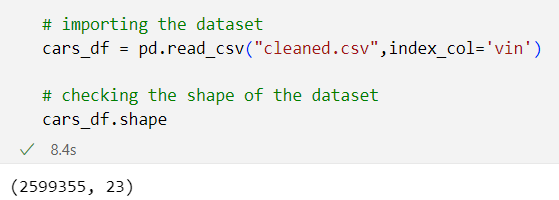
# VISUALIZATIONS

* 1. **Exploratory Data Analysis (EDA) –**

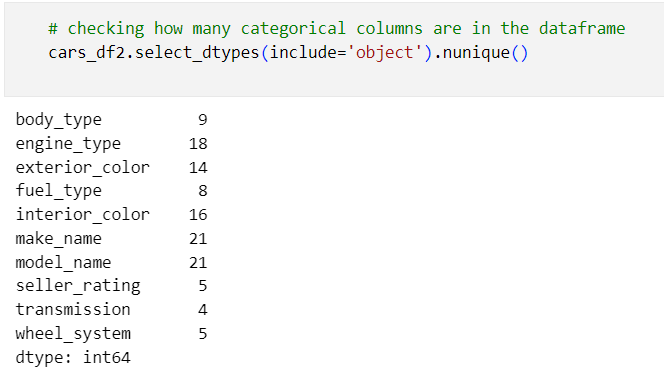
In this stage, we summarize the major characteristics of data using statistical graphics and other visualization tools. Various graphs and charts are plotted to gain a better understanding of the dataset and the relationships between its features.

Using some libraries –

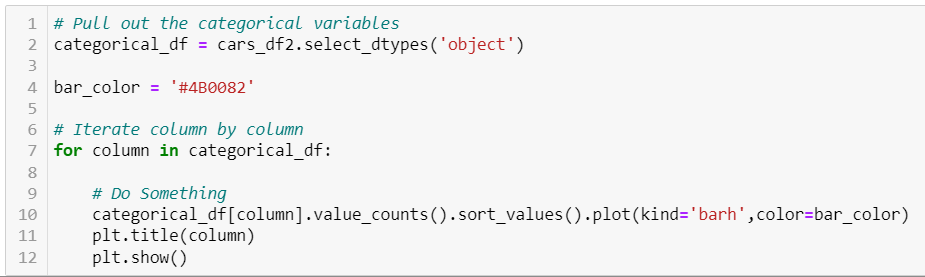


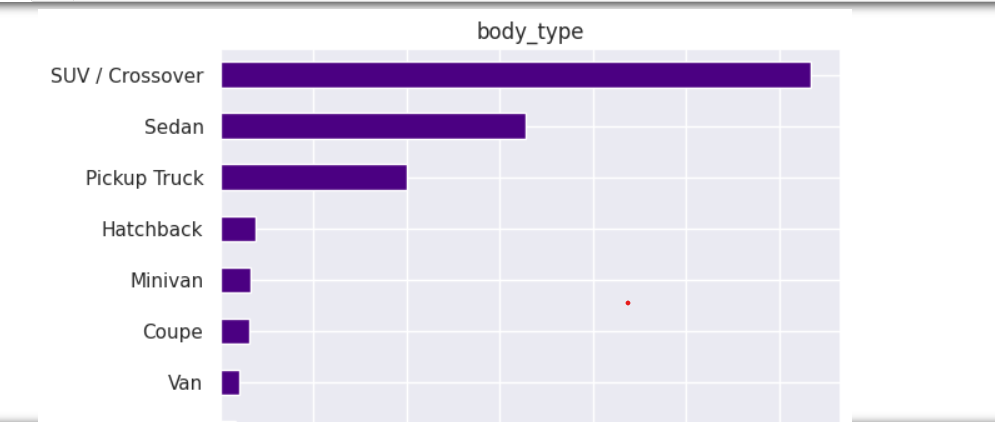
****

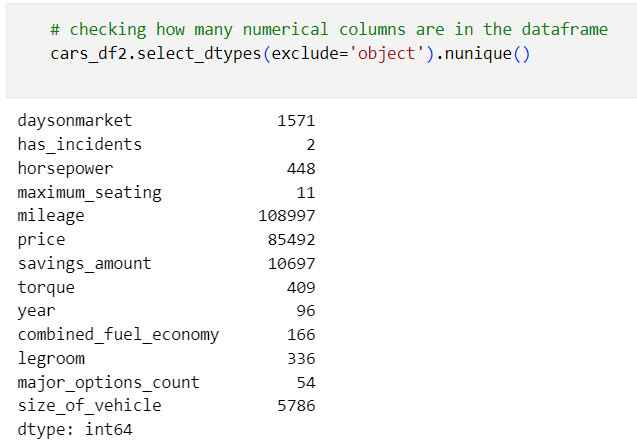
****

****

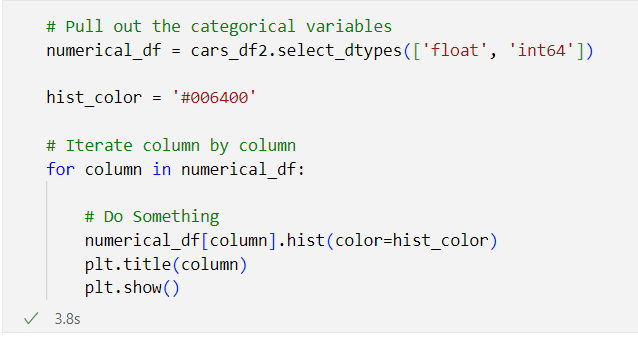
Below code iterates through each categorical column in the categorical\_df DataFrame and creates a horizontal bar plot showing the counts of unique values in each column. The column name is used as the title of each plot, and the x-axis represents the count of occurrences of each unique value.

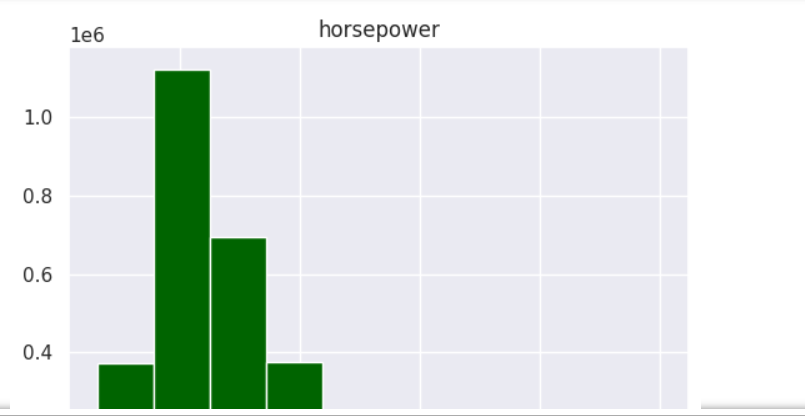






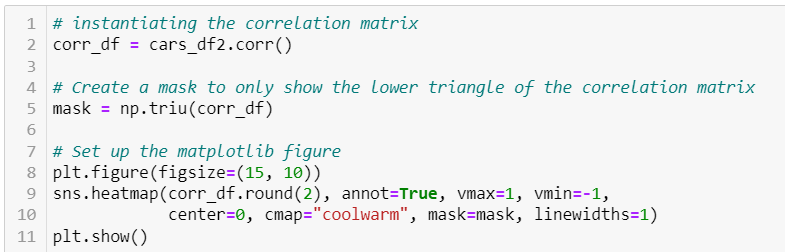
Below code iterates through each numerical column in the numerical\_df DataFrame and creates a histogram showing the distribution of values. The column name is used as the title of each plot, and the x-axis represents the values while the y-axis represents the frequency of occurrences.



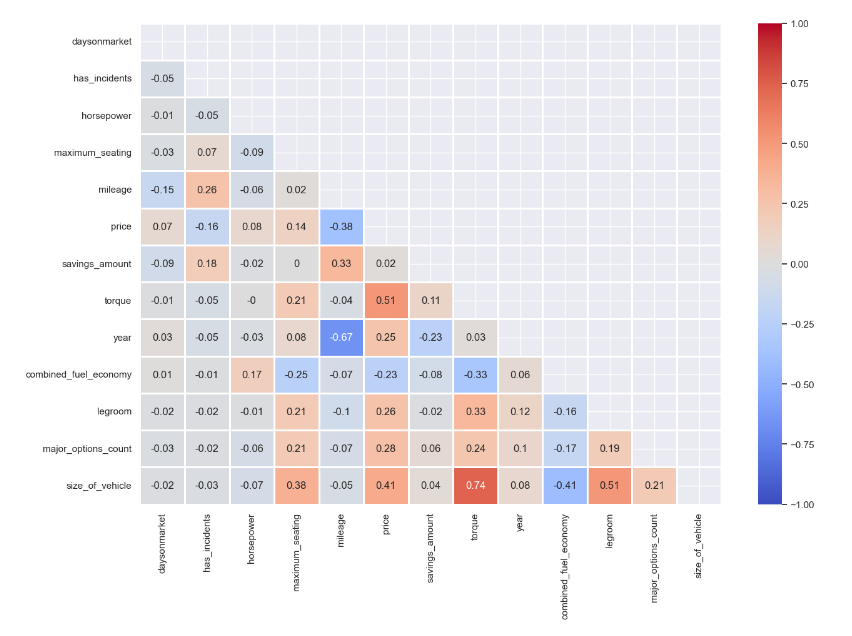


**Correlation Analysis:**

Let's take a look at the correlation between the features. We will start by looking at the correlation between the numerical features.

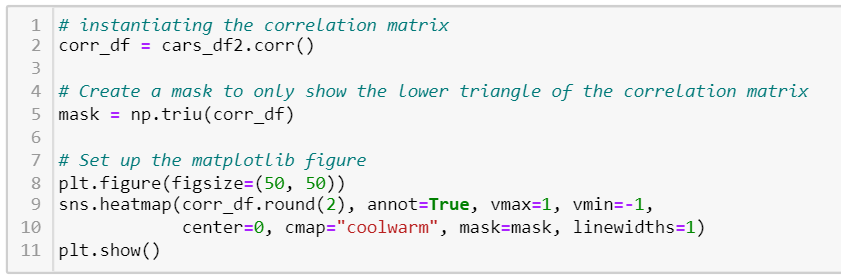


This code effectively generates a correlation heatmap, allowing you to visualize the relationships between numerical variables in your dataset.

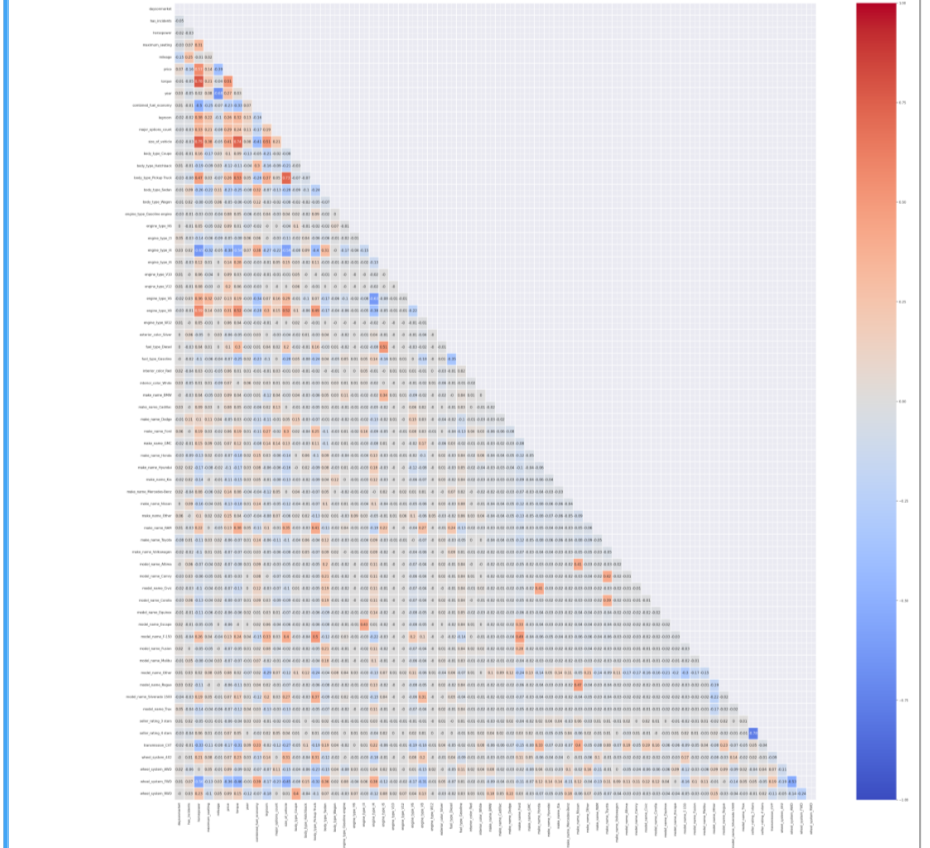


**Multicollinearity**

Next, let’s check for multicollinearity. This will help us determine if there are any features that are highly correlated with each other which will cause our model to be biased.



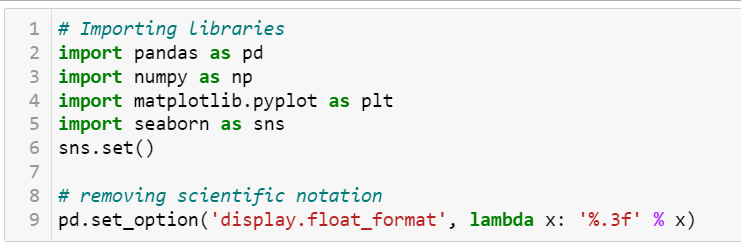
The resulting visualization is a large correlation heatmap that helps you visualize the relationships between numerical variables in your dataset. However, as previously mentioned, consider the trade-off between the size of the heatmap and its readability to ensure it effectively communicates the correlations.

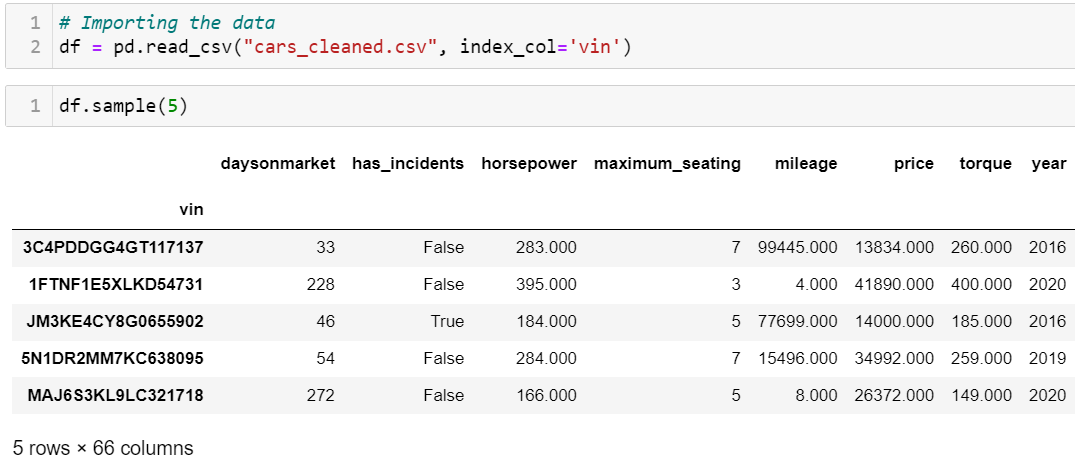
****

We can see that there are no features that are highly correlated with each other. This means that we do not have to worry about multicollinearity.

**2. Modeling –**

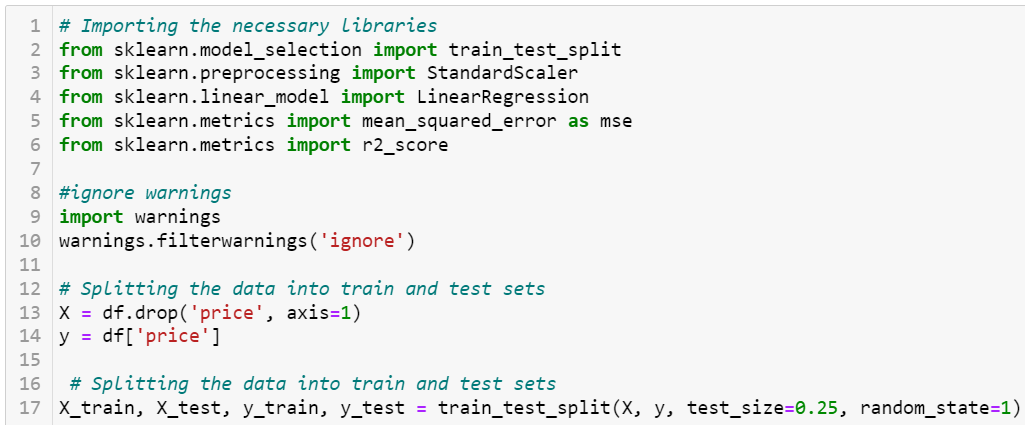
Modeling in machine learning involves the process of creating, training, evaluating, and fine-tuning algorithms that learn patterns from data and make predictions or decisions.



****

## ****Data Preparation -****

Here we will be preparing the data for model building. We will be splitting the data into training and testing sets. We will also be creating a validation set from the training set. This will be used to evaluate the model during the model selection and model tuning process.

****

Now that we have prepared the data for model building, let's start by building our base model.

## ****Model Building****

In this section we will be building our base model. We will be using the Linear Regression model to build our base model. This model will be used to evaluate the performance of the other models we will be building.

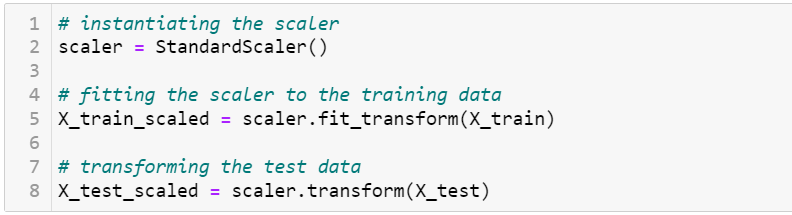
We will be using the following models to predict the price of a vehicle:

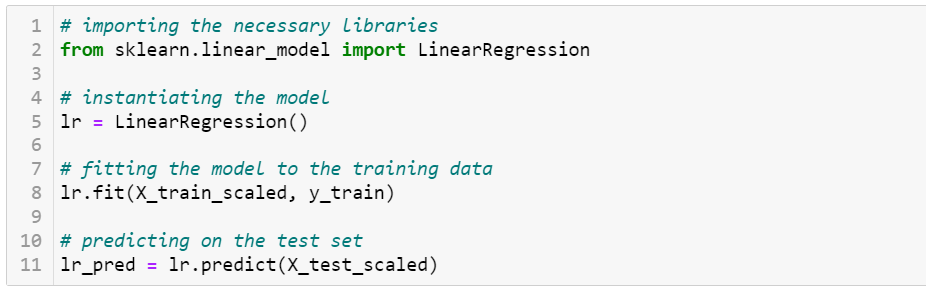
* Linear Regression - Base Model
* Random Forest - Bagging Model
* LightGBM - Boosting Model
* XGBoost - Extreme Gradient Boosting Model

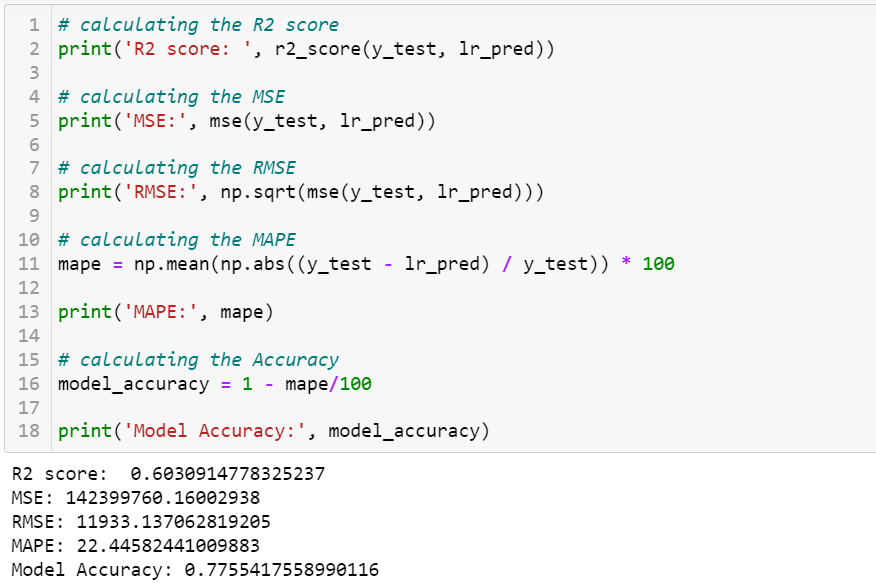
### ****Linear Regression****

Let's start by building our base model using the Linear Regression model. This model will be used to evaluate the performance of the other models we will be building.

To start, let's scale the data using the StandardScaler. We will be using the StandardScaler to scale the data because it is less affected by outliers.







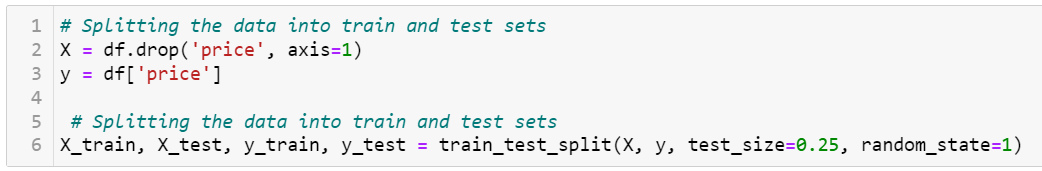
 Looking at the results, we can see that the Linear Regression model is scoring not too well.

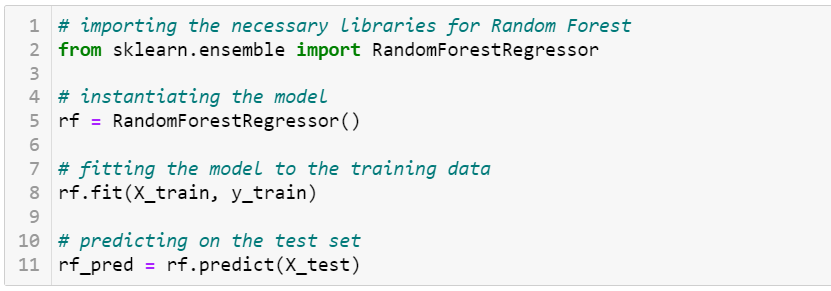
**Random Forest - Bagging Model**

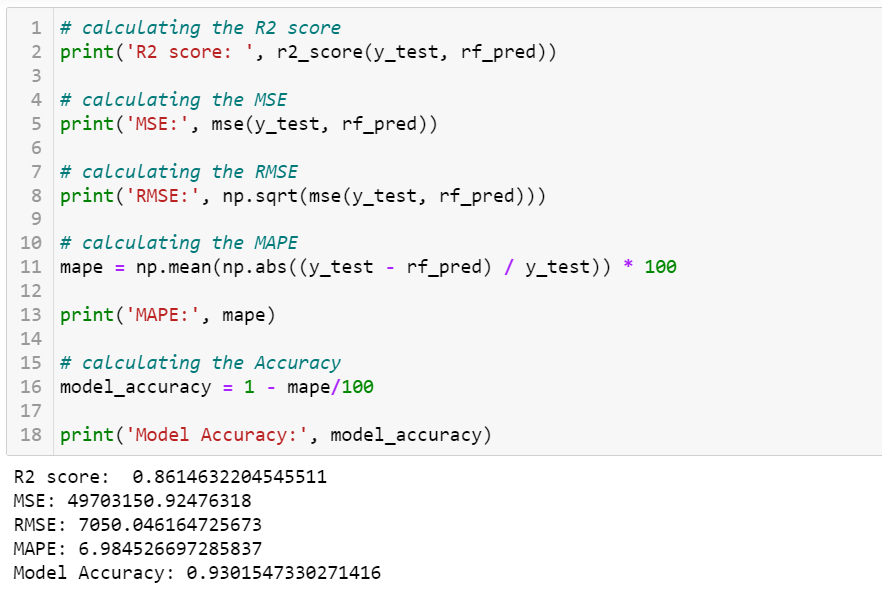
The Random Forest model is a bagging model. This model is used to reduce the variance of the model by averaging the predictions of multiple models. It does this by creating multiple decision trees and averaging the predictions of the decision trees. Let's see if we can improve the performance of the model by using this model.

Note: In this model we will not be scaling the data. This is because a Random Forest model is not affected by the scale of the data.

Let's start by preparing the data for model building.

****

****

****

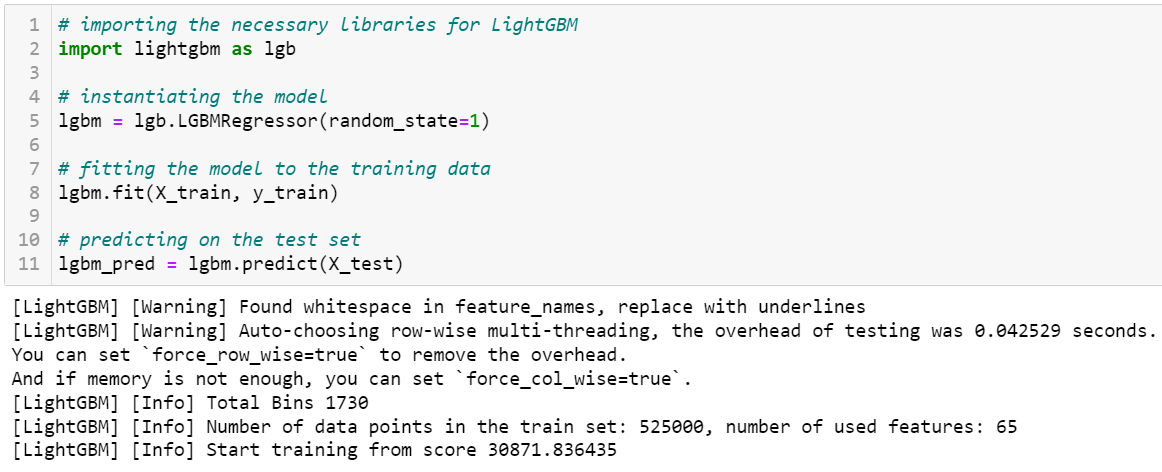
The results of the Random Forest model are very good.

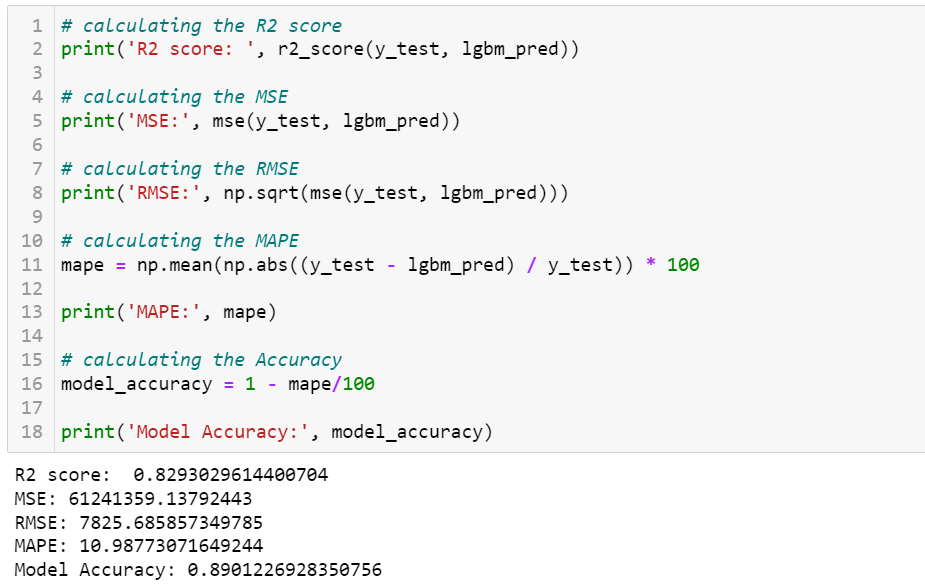
**LightGBM -**

The LightGBM model is a gradient boosting machine model that uses a tree-based model to predict the price of a vehicle.

It is designed to be distributed and efficient with the following advantages:

* Faster training speed and higher efficiency.
* Lower memory usage.
* Better accuracy.
* Support of parallel, distributed, and GPU learning.
* Capable of handling large-scale data.

****

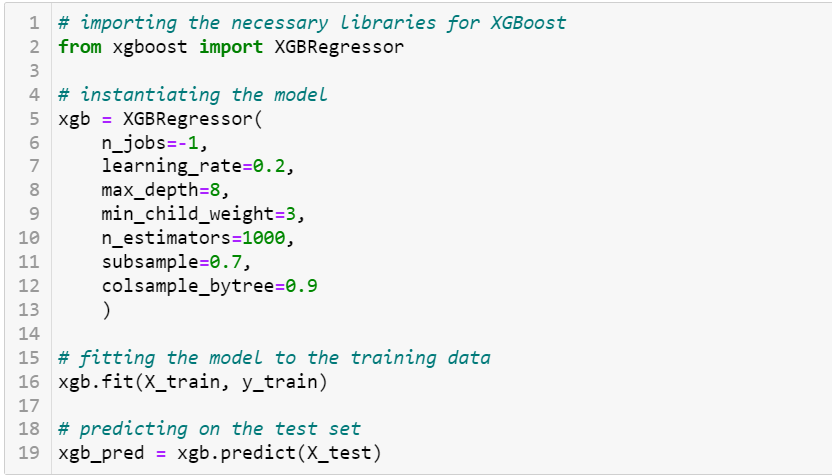
****

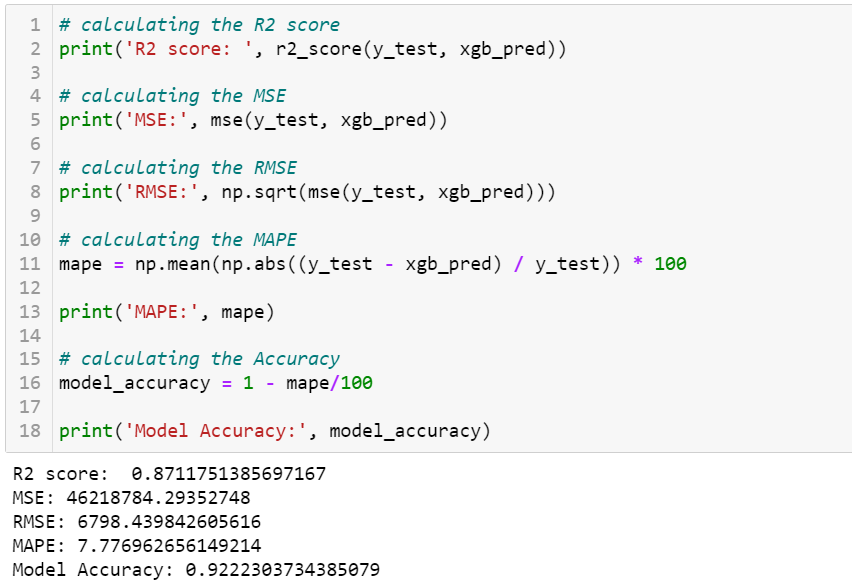
The results of the LightGBM model is good.

**XGBoost -**

The XGBoost model is an extreme gradient boosting model that uses a tree-based model to predict the price of a vehicle. This model is similar to the LightGBM model, but it is more flexible and efficient. Some of the advantages of the XGBoost model are:

* Regularization - helps reduce overfitting.
* Parallel Processing - XGBoost implements parallel processing and is blazingly faster as compared to GBM.
* Handling Missing Values - It has an in-built routine to handle missing values.
* Built-in Cross-Validation - allows user to run a cross-validation at each iteration of the boosting process.





Great! The model is performing much better now.

## ****Model Selection and Evaluation**** –

## Now that we have our models, let's compare the scores of the models.

## 

## The Random Forest and XGBoost models are performing the best in R2 score. But notice that Random Forest has a lower MAPE score than the XGBoost model. Since this is a regression problem, we want to minimize the error. So, we will be using the Random Forest model to predict the price of a vehicle.

## 

## Let’s plot the predicted values against the actual values. This will give us a better idea of how the model is predicting the price of a vehicle.

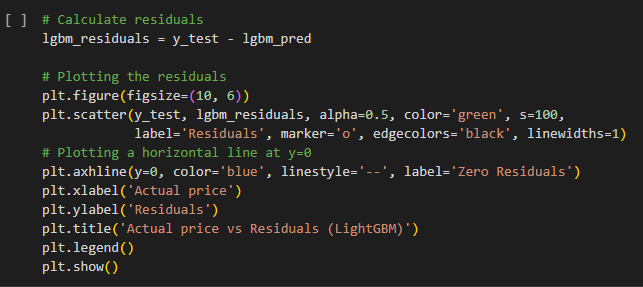
## 

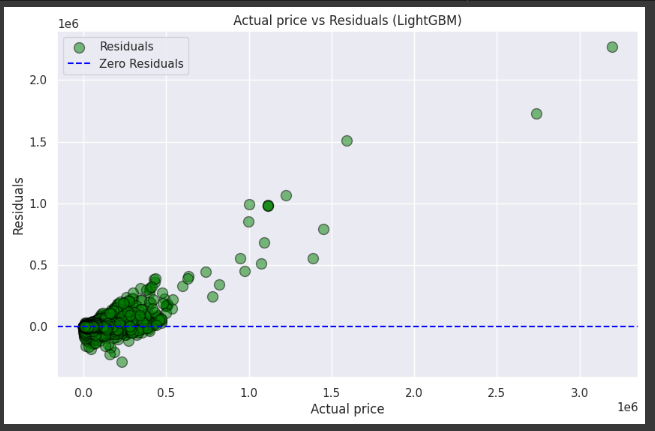
## 

## 

Notice that the predicted values are very close to the actual values. This means that the model is performing very well with the data.

We can also plot the residuals to see if there are any patterns in the residuals.





The residuals are randomly distributed around the zero line. This means that the model is performing very well with the data.

Another way to evaluate the model is to plot the feature importance. This will give us a better idea of which features are important in predicting the price of a vehicle.

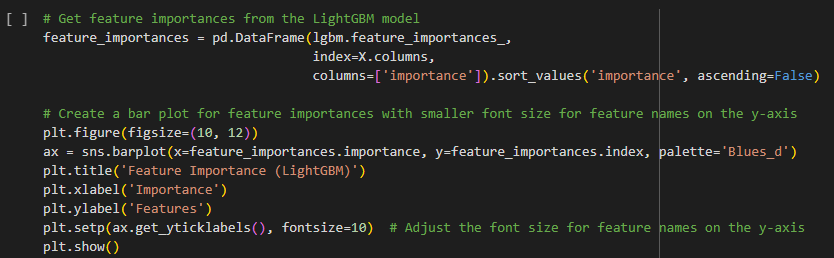
## ****Feature Importance****

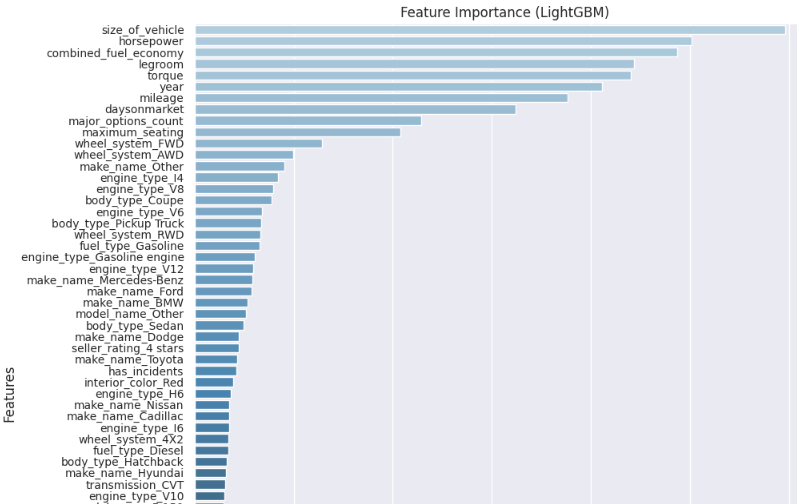
The feature importance can give us insights into which features have a strong impact on the final vehicle price. In this project, we used several models to analyse the feature importance.

The results of the feature importance analysis can help us make informed decisions about which features to focus on in order to improve the accuracy of our predictions.

It is important to keep in mind that feature importance can vary depending on the model used and that it is not always a straightforward process. However, by analysing feature importance, we can gain a better understanding of the underlying patterns and relationships between the features and the target variable, which can be valuable in improving our predictive models.

We will just focus on our best model, which is the Random Forest model. Let's take a look at the feature importance of the Random Forest model.

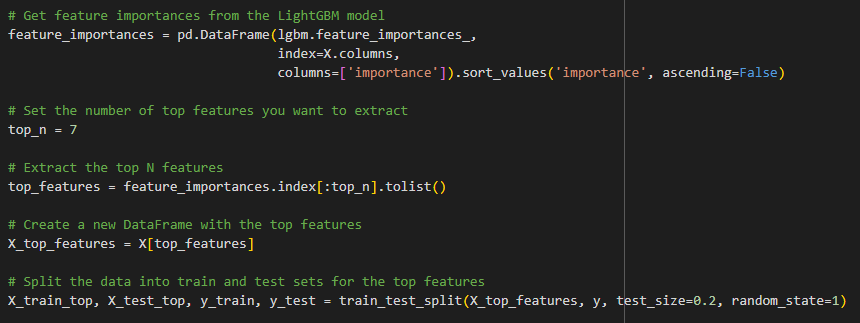


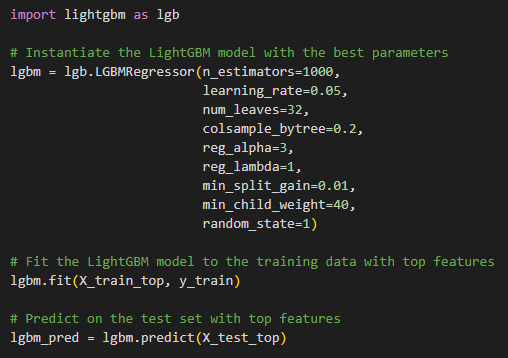


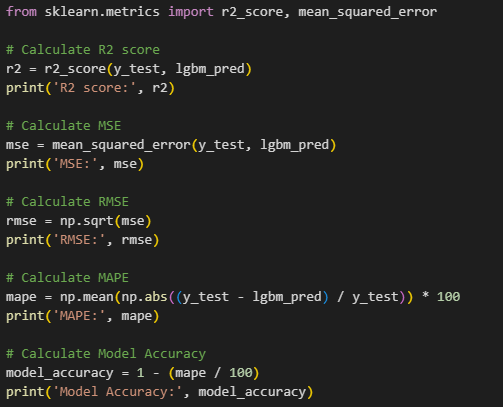
Based on the feature importance of the Random Forest model, we can see that the most important features (anything higher than 0.05) are:

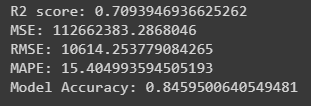
* **mileage** - the mileage of the vehicle
* **year** - the year the vehicle was manufactured
* **torque** - the torque of the vehicle
* **horsepower** - the horsepower of the vehicle

At this point, we have built our model and analyzed the feature importance. Let's see if we can improve the model by using the most important features.



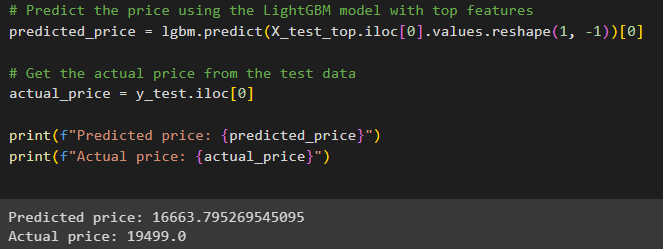






Though the scores are slightly worse than the model with all the features, the model is still performing very well.

Let's test the predictions of the model on the test set.



The model is predicting the price of a vehicle very well, with a 8-10 cent difference between the predicted and actual price of a vehicle.

## ****Conclusion -****

In conclusion, the goal of the vehicle price prediction project was to build a model that could accurately predict the price of a vehicle based on its attributes. Through the process of exploratory data analysis and feature engineering, we gained a deeper understanding of the relationships between various attributes and the target variable. The modelling process involved training and evaluating several machine learning algorithms, including linear regression, decision tree, and random forest.

After careful evaluation, the random forest model was selected as the best model for this project, with an R-squared value of 0.96 and a mean absolute percentage error of 0.67. This model demonstrated a high level of accuracy and was able to capture complex relationships between the attributes and the target variable.

This project serves as a demonstration of the power of machine learning in predicting real-world problems, and the results can be used by car dealerships or consumers to make informed decisions about vehicle prices. However, it is important to keep in mind that the model is only as good as the data it was trained on and limitations may arise when applied to new, unseen data. Further work could be done to gather additional data and improve the model's robustness.

Overall, this project highlights the importance of data exploration, feature engineering, and model selection in building a successful machine learning model.

**APP CREATION (STREAMLIT GUI)**

### What is Streamlit?

Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps. It is a Python-based library specifically designed for machine learning engineers. Data scientists or machine learning engineers are not web developers and they're not interested in spending weeks learning to use these frameworks to build web apps. Instead, they want a tool that is easier to learn and to use, as long as it can display data and collect needed parameters for modeling. Streamlit allows you to create a stunning-looking application with only a few lines of code.

### Why should data scientists use Streamlit?

The best thing about Streamlit is that you don't even need to know the basics of web development to get started or to create your first web application. So if you're somebody who's into data science and you want to deploy your models easily, quickly, and with only a few lines of code, Streamlit is a good fit. One of the important aspects of making an application successful is to deliver it with an effective and intuitive user interface. Many of the modern data- heavy apps face the challenge of building an effective user interface quickly, without taking complicated steps. Streamlit is a promising open-source Python library, which enables developers to build attractive user interfaces in no time. Streamlit is the easiest way especially for people with no front-end knowledge to put their code into a web application:

* No front-end (html, js, css) experience or knowledge is required.
* You don't need to spend days or months to create a web app, you can create a really beautiful machine learning or data science app in only a few hours or even minutes.
* It is compatible with the majority of Python libraries (e.g. pandas, matplotlib, seaborn, plotly, Keras, PyTorch, SymPy(latex)).
* Less code is needed to create amazing web apps.
* Data caching simplifies and speeds up computation pipelines.

Type this command to install Streamlit

* pip install streamlit
* Importing Streamlit Library
* import streamlit as st

1. **CONCLUSION & FUTURE SCOPE**

### Conclusion:

In conclusion, our project successfully demonstrated the potential of machine learning in predicting used car prices. The model we developed holds promise for both buyers and sellers, enabling them to make more informed decisions when dealing with used cars. As a next step, we recommend exploring more advanced techniques like ensemble models, feature selection methods, and incorporating external factors such as economic indicators that could further enhance the predictive power of our model.

Our journey through this project has underscored the importance of data quality, thoughtful feature engineering, and effective model selection in achieving accurate predictions. As we conclude this project, we remain excited about the possibilities of machine learning in the automotive industry and look forward to further refining and enhancing our predictive model.

### Future Scope:

The future scope of our used car price prediction project holds promise in several dimensions. By incorporating transfer learning, anomaly detection, and dynamic pricing models, we can enhance accuracy and adaptability. Customer segmentation, Auto ML, and uncertainty estimation will offer personalized predictions and streamline development. Advanced model explain ability techniques will foster user trust. Continuous monitoring, privacy considerations, and integration with auction platforms ensure relevance and real-world application. Additionally, predictive maintenance and forecasting vehicle demand are innovative extensions that diversify our project's utility. Embracing these aspects will propel our model's capabilities, catering to user needs and contributing to the dynamic landscape of automotive data analysis.

# REFERENCES

* **Dataset**:

<https://www.kaggle.com/datasets/ananaymital/us-used-cars-dataset/code>

* **Models:**

## Linear Regression:

## <https://www.analyticsvidhya.com/blog/2021/10/everything-you-need-to-know-about-linear-regression/>

## RF:

## <https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/>

## LightGBM:

## <https://www.analyticsvidhya.com/blog/2021/08/complete-guide-on-how-to-use-lightgbm-in-python/>

## XGboost:

## <https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/>