FINAL PROJECT

Car Price Prediction

7CS078/UZ1: Data Science

Submission Date: 2 May 2023

Project Start Date and End Date: March-May

Submitted by: Mehmet Omer DEMIR

Faculty of Science and Engineering

University of Wolverhampton
7CS078/UZ1: Data Science

Instructor: Ahmed Khubaib

Email: M.O.Demir@wlv.ac.uk

Table of Contents

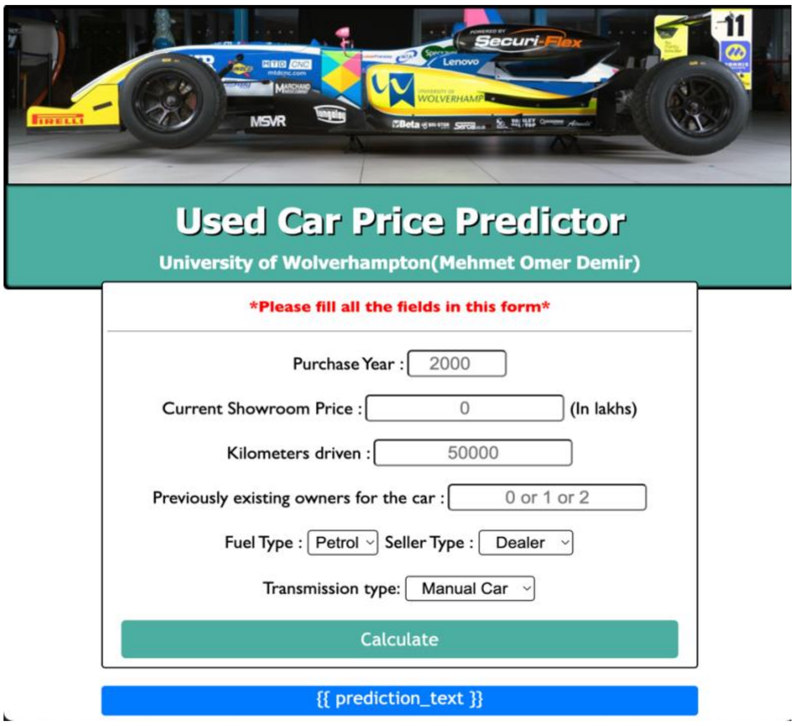
1.Abstract

In this report, we are working on a system used for price forecasting in the used car market. Used car sales are increasing in many countries due to rising new car prices and customers' limited budgets. This increase has made the correct pricing and trading of vehicles even more important, so customers want to know the market value of their vehicles.

This report describes the creation of a vehicle price estimation system using a vehicle price dataset from Kaggle. This system calculates the estimated market value of a vehicle using vehicle characteristics (mileage, year, owner, etc.) and other data and presents it to customers. Thanks to this application, people can reach the real value of their vehicles more easily and enable them to trade more consciously and quickly in the second hand market**[3].**

Taking into account the variations in the vehicle market, this method performs very effectively in generating precise price estimates. Model year, brand, mileage and other features of the vehicle are important factors in determining the price of the vehicle. In addition, this application provides information about the price-performance ratio of each car model, helping customers to choose the car that suits their needs, allowing them to make quick decisions without the need for lengthy market research.

This system for estimating vehicle costs will ensure accurate pricing in second-hand vehicle purchases and sales transactions by exceeding the market prices. This system will protect both buyers and sellers in trading, as sellers will use it to maintain the value of their vehicle, while buyers will have quick and reliable access to the market value of a used vehicle.(**Figure1**) Below is the HTML image of my system with an accuracy rate of over 92%.


Used Car Price Predictor

University of Wolverhampton(Mehmet Omer Demir)

Please fill all the fields in this form

Purchase Year : 2000

Current Showroom Price : 0 (In lakhs)

Kilometers driven : 50000

Previously existing owners for the car : 0 or 1 or 2

Fuel Type : Petrol Seller Type : Dealer

Transmission type: Manual Car

Figure 1: Screenshot of used car price estimation application.

2.Problem Statement

Building a machine learning model to accurately estimate the selling price range of used cars based on these various parameters presented in our dataset is the problem statement in this scenario.

People who want to sell their own cars at fair market value without being ripped off are one reason why accurately projecting used car prices is essential. The other factor can be that some people prefer to acquire used cars rather than brand-new ones from authorised dealers and as a result, they need accurate price information.

We can use a variety of classification algorithms to achieve this goal, including decision trees and linear regression, which, when trained on past data from comparable sales, can forecast current values within an acceptable error tolerance level.

Overall, by reducing personal bias when buying or selling a used automobile, this predictive model will ensure fair pricing, increase trust between buyers and sellers, and spare both parties the time, effort, and resources needed in negotiating deals without the support of an expert.

Challenges

- Finding the ideal regression algorithm for our situation among those like Linear Regression, Lasso Regression, Random Forest Regression, Ridge Regression, etc. was difficult.

- We had to learn because we were not familiar with random forest regression.

- Reconstructing the given dataset. Changing the specific values into numerical form. And eliminating the pointless features.

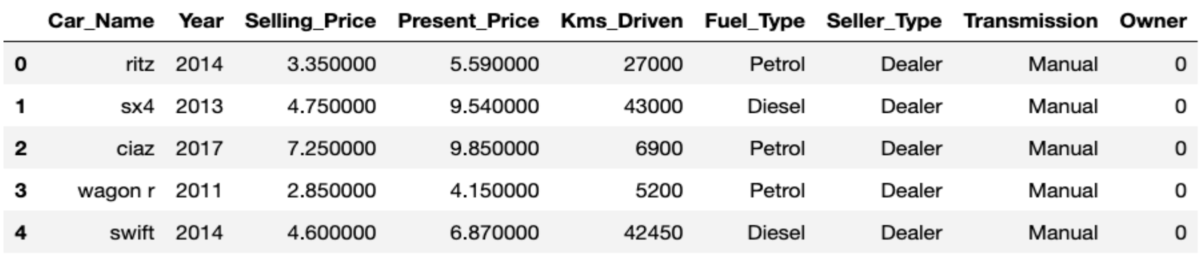
- It was difficult to decide between gridsearchCV and randomizedsearchCV while trying to determine the random forest regression's hyperparameters.

- It was difficult for us to decide on the dominating features, therefore we used heat maps with pearson coefficient correlation and an additional tree regressor to determine the value of each feature.

- We have developed a web page to show the operation of our model that we need to learn HTML and CSS. In addition to these, we have written a back-end code to show the result we have received on the web page.

3. Dataset

I used a dataset from Kaggle for this project. I mentioned some dataset-related details, such as features. The goal of the project is to develop a regression model that can precisely predict the price of a car given its features.



	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	ritz	2014	3.350000	5.590000	27000	Petrol	Dealer	Manual	0
1	sx4	2013	4.750000	9.540000	43000	Diesel	Dealer	Manual	0
2	ciaz	2017	7.250000	9.850000	6900	Petrol	Dealer	Manual	0
3	wagon r	2011	2.850000	4.150000	5200	Petrol	Dealer	Manual	0
4	swift	2014	4.600000	6.870000	42450	Diesel	Dealer	Manual	0

Figure 3.1: Car prices data overview

- **Car_Name**:Information on the used car's make and model is provided in this column. It may also provide additional information, such as features, trim level, or special editions.

- **Selling_price**:This column lists the asking price for each specific used car. As a result, we may utilise machine learning models to forecast automobile pricing based on the vehicles' atributes and traits.

- **Year**:The year column shows the year that the used car was produced. Due to the close correlation between this information and depreciation, newer cars typically have higher values than older ones. Therefore, it's essential to know the year of manufacture when determining a car's value.

- **Seller_Type**:I have two types of vendors, dealer and individual.

- **Kms_Driven**:Kilometer driven shows the total distance that has already been covered by any specific vehicle indicated in kilometers. Generally speaking, cars with lower mileage tend to be priced higher since they are considered less worn out and have less wear & tear compared to others with high odometer readings.

- **Fuel_Type**:Different fuels offer various benefits in terms of price efficiency (i.e., miles per Litre), availability at gas stations across different, but important, locations/regions.

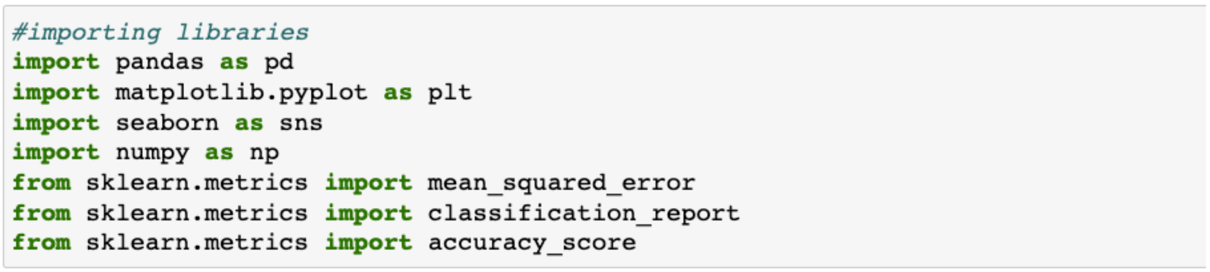
- **Transmission**:Gearbox refers to the function of a car's transmission that is responsible for shifting gears between gears based on road conditions/driving habits, user preferences, etc.

- **Owner**: If a vehicle changes hands too much, it may mean that there is a chronic problem in the mechanics of that vehicle, which is one of the factors affecting the price of the vehicle.

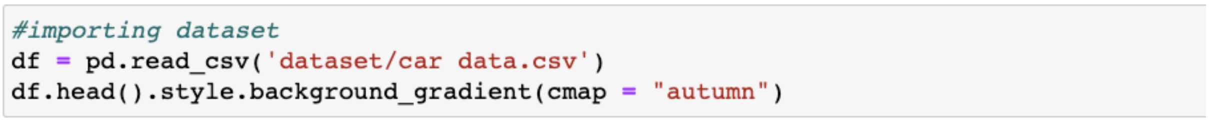
3.1.Collection of Data

The Dataset came from Kaggle. This dataset includes features like the names of the cars, selling prices,year, current prices, and fuel types, among others **(Figure 3.1)**. Now that we know how to we can read the dataset into the Jupyter Notebook, let's move on. The dataset is available for download from Kaggle as a csv file.

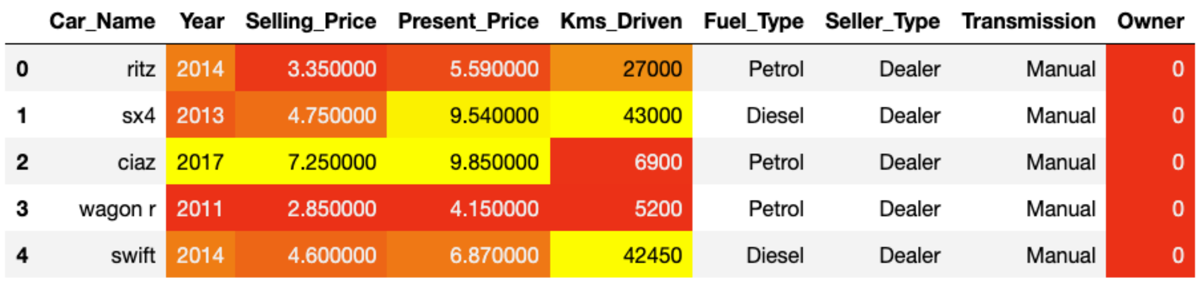
-Code for collecting data from .CSV
# Importing libraries and Dataset



```
#importing libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.metrics import mean_squared_error
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```



```
#importing dataset
df = pd.read_csv('dataset/car data.csv')
df.head().style.background_gradient(cmap = "autumn")
```



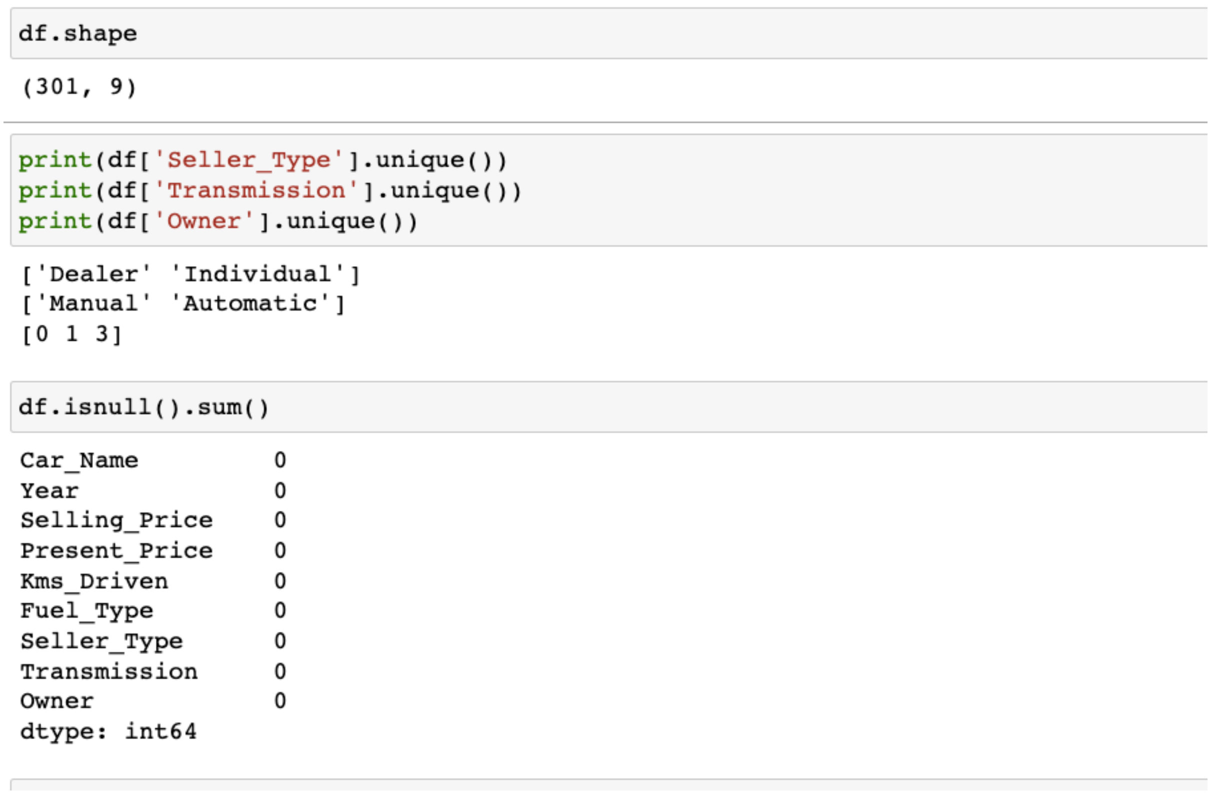
	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	ritz	2014	3.350000	5.590000	27000	Petrol	Dealer	Manual	0
1	sx4	2013	4.750000	9.540000	43000	Diesel	Dealer	Manual	0
2	ciaz	2017	7.250000	9.850000	6900	Petrol	Dealer	Manual	0
3	wagon r	2011	2.850000	4.150000	5200	Petrol	Dealer	Manual	0
4	swift	2014	4.600000	6.870000	42450	Diesel	Dealer	Manual	0

Figure 3.1: Dataset of Car Price.

3.2.Preprocessing of Dataset

We don't need to clean data in this Car Price Dataset. The dataset was already cleared when we downloaded it from Kaggle. However, to be sure, it is useful to see the missing or empty values in the dataset. We also need to know the shape of the dataset**(Figure 3.2)**.

# Shape of the Car_Price Dataset
# Checking for the Null or Missing Values



```
df.shape
```

```
(301, 9)
```

```
print(df['Seller_Type'].unique())
print(df['Transmission'].unique())
print(df['Owner'].unique())
```

```
['Dealer' 'Individual']
['Manual' 'Automatic']
[0 1 3]
```

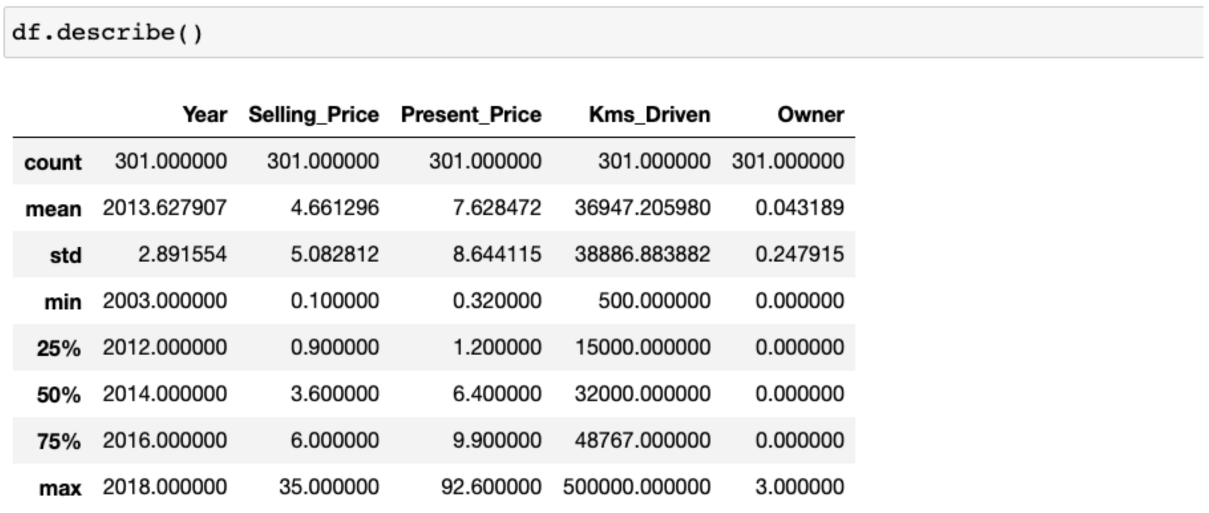
```
df.isnull().sum()
```

```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
dtype: int64
```

Figure 3.2: There is no NaN value.

3.3.Exploratory Data Analysis

Exploratory data analysis (EDA) is a method used in statistics to summarise their key features and examine data sets, frequently using visual techniqes**(Figure 3.3)**. EDA is generally used to see what the data can tell us outside of the formal hypothesis testing process or modelling, regardless of whether a statistical model is used or not.

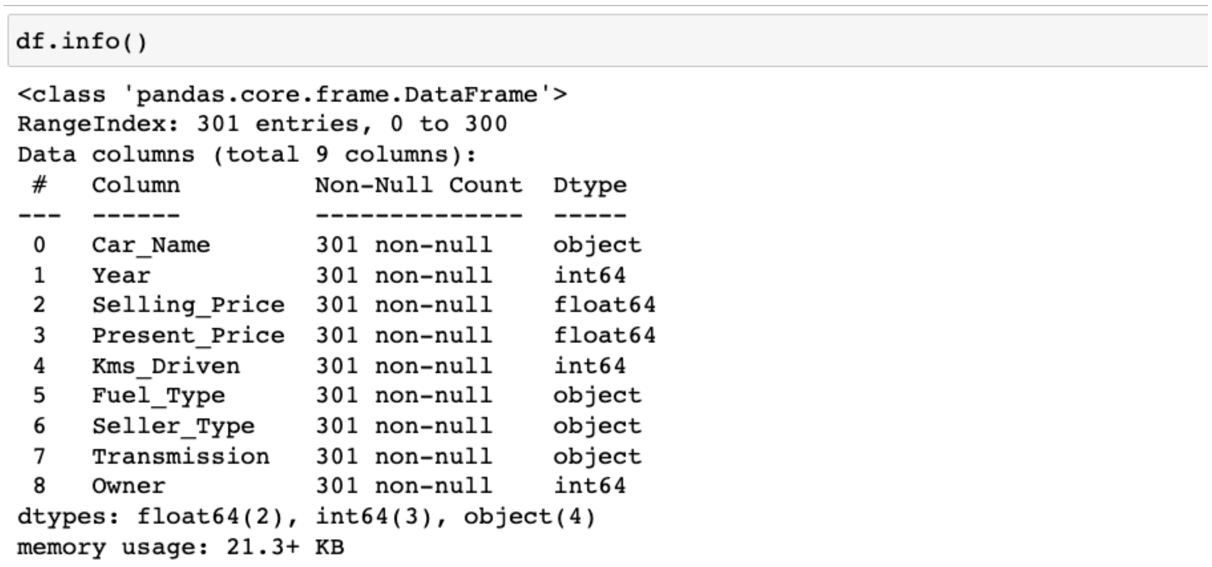


```
df.describe()
```

	Year	Selling_Price	Present_Price	Kms_Driven	Owner
count	301.000000	301.000000	301.000000	301.000000	301.000000
mean	2013.627907	4.661296	7.628472	36947.205980	0.043189
std	2.891554	5.082812	8.644115	38886.883882	0.247915
min	2003.000000	0.100000	0.320000	500.000000	0.000000
25%	2012.000000	0.900000	1.200000	15000.000000	0.000000
50%	2014.000000	3.600000	6.400000	32000.000000	0.000000
75%	2016.000000	6.000000	9.900000	48767.000000	0.000000
max	2018.000000	35.000000	92.600000	500000.000000	3.000000

Figure 3.3: Desciription of Data.

The only numerical values left for the Features are categorical values. In the following section, we'll examine it **(Figure 3.4)**.



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Car_Name       301 non-null    object 
 1   Year           301 non-null    int64  
 2   Selling_Price  301 non-null    float64
 3   Present_Price  301 non-null    float64
 4   Kms_Driven     301 non-null    int64  
 5   Fuel_Type      301 non-null    object 
 6   Seller_Type    301 non-null    object 
 7   Transmission   301 non-null    object 
 8   Owner          301 non-null    int64  
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
```

Figure 3.4 : Information of Data

3.4.Feature Engineering

Feature engineering is the process of using domain expertise and data mining techniques to extract features from unstructured data. The effectiveness of machine learning algorithms can be increased by using these features. It is possible to think of feature engineering as machine learning in application.

In this case, we're going to separate the new feature for output prediction from the existing features. That method will demonstrate your domain knowledge. We'll handle the categorical features in the dataset of car prices here as well.

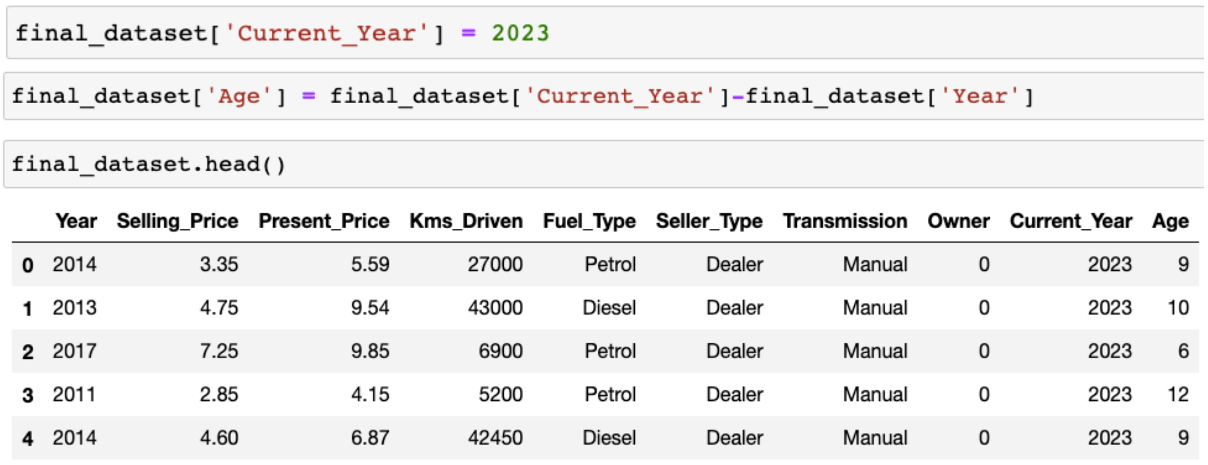
3.5.Data Cleaning

To calculate the age of the vehicle we need to use the following simple formula **(Figure 3.5)**.

Age = Current_Year —Year

For Example,

12 = 2011–2023



```
final_dataset['Current_Year'] = 2023
```

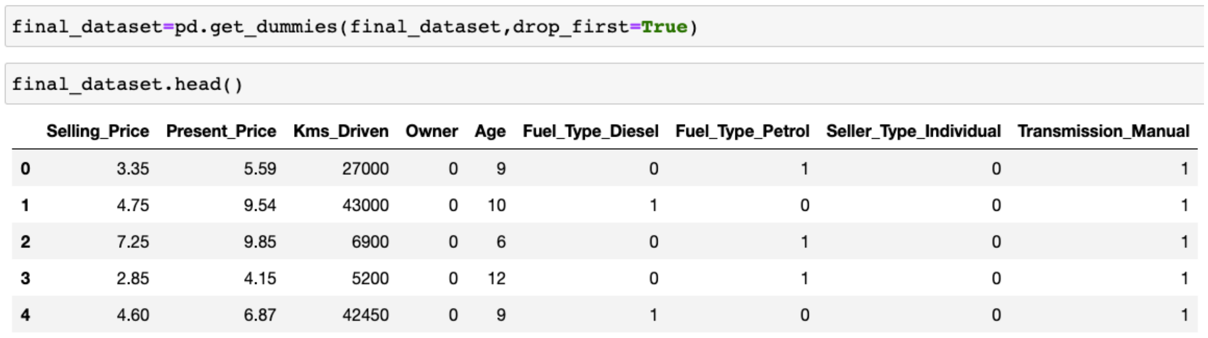
```
final_dataset['Age'] = final_dataset['Current_Year']-final_dataset['Year']
```

```
final_dataset.head()
```

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Current_Year	Age
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0	2023	9
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0	2023	10
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0	2023	6
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0	2023	12
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0	2023	9

Figure 3.5: Calculated state of the current age of the vehicles.

# The Categorical Values are Handled
The final dataset we produced after cleaning it looks like **(Figure 3.6).**



```
final_dataset=pd.get_dummies(final_dataset,drop_first=True)
```

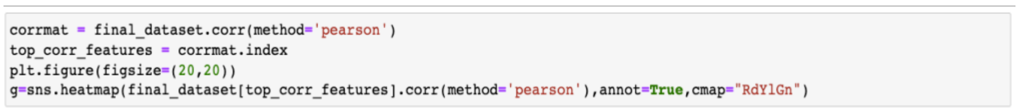
```
final_dataset.head()
```

	Selling_Price	Present_Price	Kms_Driven	Owner	Age	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
0	3.35	5.59	27000	0	9	0	1	0	1
1	4.75	9.54	43000	0	10	1	0	0	1
2	7.25	9.85	6900	0	6	0	1	0	1
3	2.85	4.15	5200	0	12	0	1	0	1
4	4.60	6.87	42450	0	9	1	0	0	1

Figure 3.6: Having obtained the feature extraction.

3.6.Feature Observation

- Plotting the heatmap of feature correlation **(Figure 3.7).**



```
corrmat = final_dataset.corr(method='pearson')
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
g=sns.heatmap(final_dataset[top_corr_features].corr(method='pearson'),annot=True,cmap="RdYlGn")
```

Figure 3.7: Implementation of heatmap code.

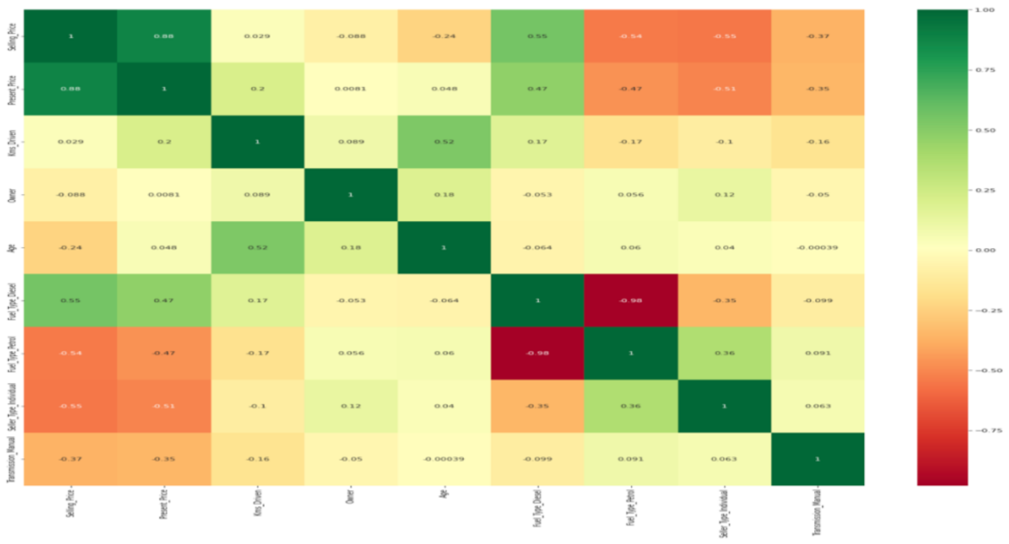


Figure 3.8: Heatmap of Dataset.

- Plotting the Pairplot

```
sns.pairplot(final_dataset)
<seaborn.axisgrid.PairGrid at 0x7fcc50191760>
```

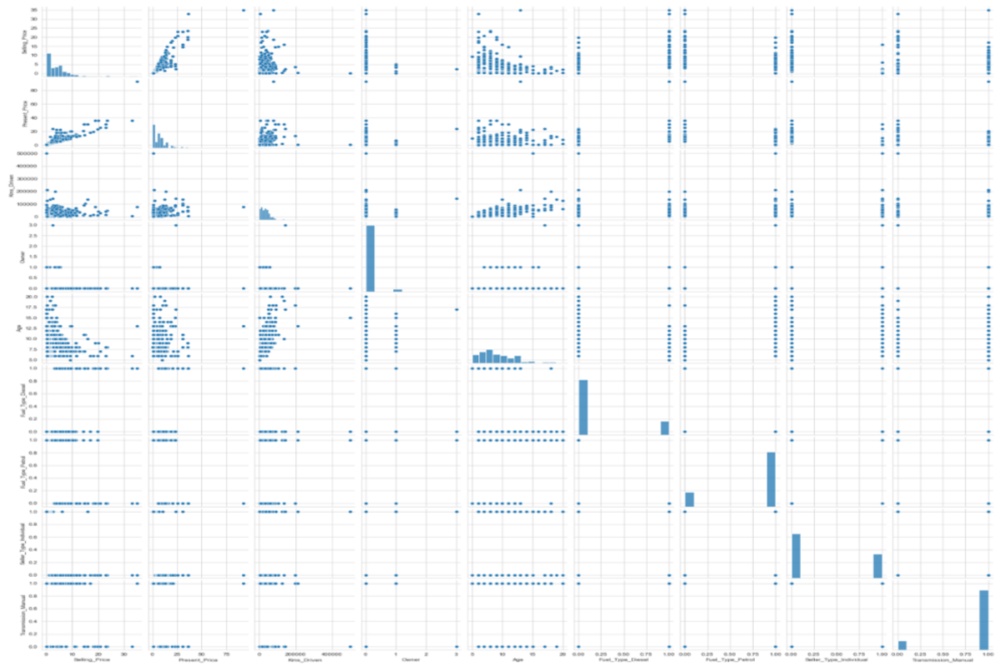


Figure 3.9: Pairplot.

```
sns.set_style('whitegrid')

sns.countplot(x='Fuel_Type', data=df)
```

```
<AxesSubplot:xlabel='Fuel_Type', ylabel='count'>
```

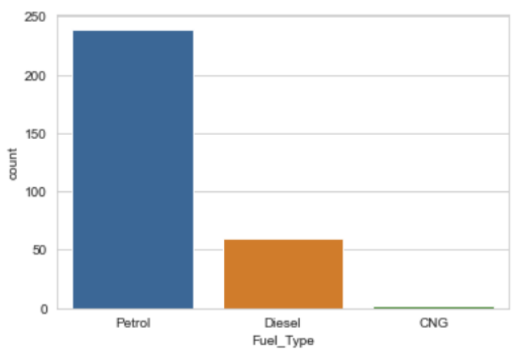


Figure 3.10 : Vehicle fuel types .

```
sns.set_style('whitegrid')

sns.countplot(x='Seller_Type',data=df)
```

```
<AxesSubplot:xlabel='Seller_Type', ylabel='count'>
```

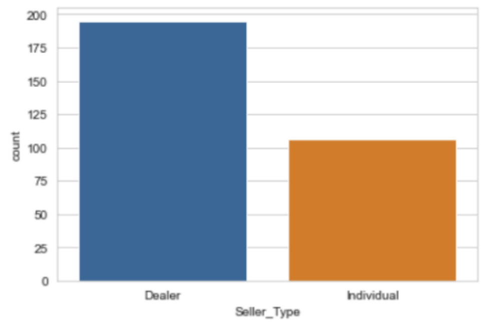


Figure 3.11 : Car dealer type chart.

```
sns.set_style('whitegrid')

sns.countplot(x='Transmission',data=df)
```

```
<AxesSubplot:xlabel='Transmission', ylabel='count'>
```

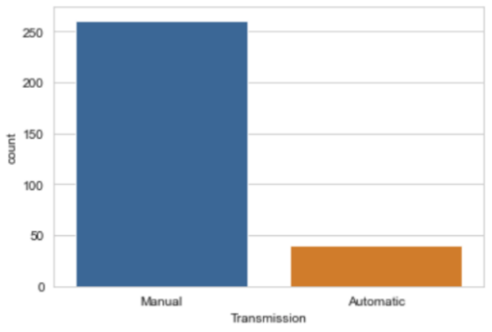


Figure 3.12 : Vehicle gear type.

```
#plot graph of  better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(5).plot(kind='barh')
plt.title('Feature Importances')
plt.show()
```

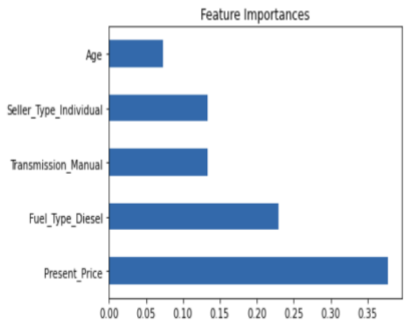


Figure 3.13 : Feature Importances.

```
sns.distplot(df['Kms_Driven'].dropna(),kde=False,color='darkred',bins=43)
```

```
<AxesSubplot:xlabel='Kms_Driven'>
```

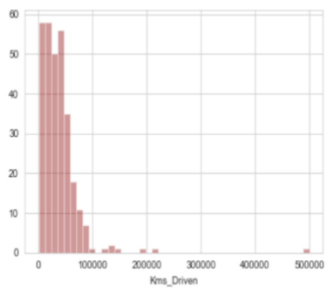


Figure 3.14: Km driven.

```
sns.distplot(df['Present_Price'].dropna(),kde=False,color='darkblue',bins=43)
```

```
<AxesSubplot:xlabel='Present_Price'>
```

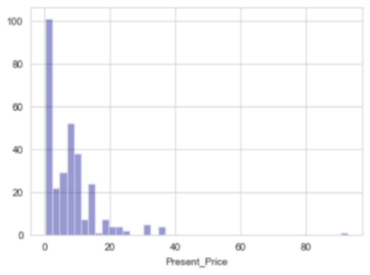


Figure 3.15: Present Price.

Summary of dataset steps;

Step 1.Collecting Data Sources: You need to decide where to get your dataset. You can pull the data from the internet using Selenium or Kaggle offers many data sets and you can easily access these data sets for free, so I used the Used Cars dataset in this project.

Step 2.Data Preprocessing and Cleaning:Preprocessing data involves editing, merging, and filtering data. For example, operations such as merging data from many data sources and deleting unnecessary columns can be performed. In this project, we got the data from Kaggle, so the data we had was very clean compared to a real project.

Step 3.Data Formatting: It is a process used to make the data suitable for analysis. In this step, it is necessary to format the data correctly and determine the data types. For example, if all the data in a column must be numeric, the data in this column must be converted to numeric.

Step 4.Data Normalization: This is a process used when data needs to be on a similar scale. In this step, it is ensured that the data is on a similar scale. For example, if the data in a columun needs to be scaled between 0 and1, the data in this column should be subjected to Decolonization.

Step 5.Data Standardization: It is a process used in case the data are at different scales. In this step, the data is prevented from being at different scales. For example, the mean and standard deviation of the data in a column are calculated, and all the data are scaled according to these calculation.

Step 6.Data Visualization: It is a process used for better understanding and discovery of data. In this step, visual tool like graphics, tables, and histograms are used to examine the data.

4.Model and Approach to Solution

This study aims at developing a robust regression model, which will allow precise estimation of car prices.We require some historical information about previously-owned vehicles, for which we used the price and a few other typical features. The car price was accepted as the dependent variable, and the other characteristics were accepted as the independent variable.

The Random Forest Method and the Extra Tree Regression algorithm, both of which are machine learning algorithms, are incorporated into the proposed model. In this model, the dataset is loaded in its earliest stage so that it can be examined further. For this particular model, I've been using a Dataset available on Kaggle. After performing data processing processes in this dataset, such as identifying missing values and hot encoding of catecile variables, we will start training a model for the distributed dataset into two First Training DataSets and Second Test DataSets**[6]**. This test data was selected at random out of the original dataset. We used two machine learning methods, Random Forest algorithm and ExtraTree Regression algorithm as well as RandomizedSearchCV to tune the parameters for optimal results prediction. I'm going to test this prediction with a test dataset built with the scikitLearn library and calculate its accuracy as soon as the model predicts a result.

A regression model based on collective learning is Random Forest. A decision tree model, specifically as the name suggests, uses several decision trees to build the ensemble model that collectively provides a prediction **(Figure 4.1).** The advantage of this model is that trees grow at the same time but with little correlation, which allows for a good outcome because no tree can be prone to individual defects in other trees. Bootstrap Aggregation, also known as bagging, provides the necessary randomness to produce robust and unrelated trees and helps partially achieve this irrelevant behavior**[2]**. Therefore, this model was chosen to compare a bagging technique with the gradient boosting techniques below and to account for the large number of features in the dataset.

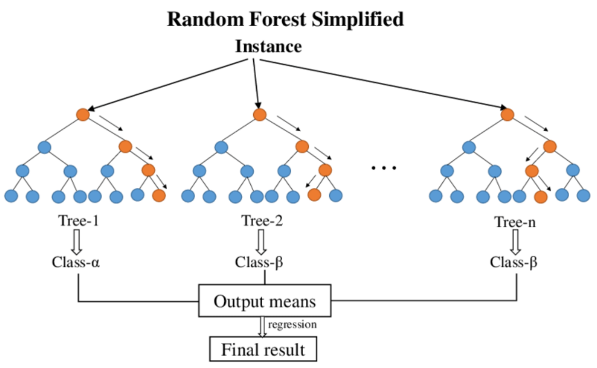
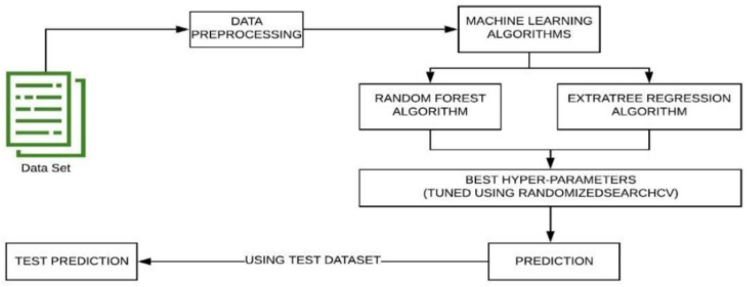


Figure 4.1: Random Forest Simplified

4.1.Model Traning


DATA PREPROCESSING
MACHINE LEARNING ALGORITHMS
RANDOM FOREST ALGORITHM
EXTRATREE REGRESSION ALGORITHM
Data Set
BEST HYPER-PARAMETERS (TUNED USING RANDOMIZEDSEARCHCV)
TEST PREDICTION
USING TEST DATASET
PREDICTION

There were splits in the train-test data. Specifically, we divide the data into training and test sets, fit potential models on the training set, assess them, and choose them on the test set. The dataset is further divided into a train set and a test set using the sklearn. model_selection library's train_test_split function.
Finding the best hyperparameters for our model prediction is done using RandomizedSearchCV tuning of this model.
Hyper parameters are those that can't be directly learned through a regular training regimen. The grid is randomly moved by RandomizedSearchCV to find the optimal set of hyperparameters.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
```

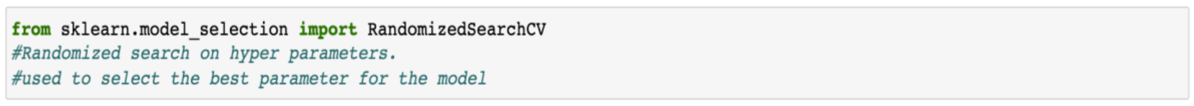
```
from sklearn.ensemble import RandomForestRegressor
regressor=RandomForestRegressor()
```

```
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
print(n_estimators)
```

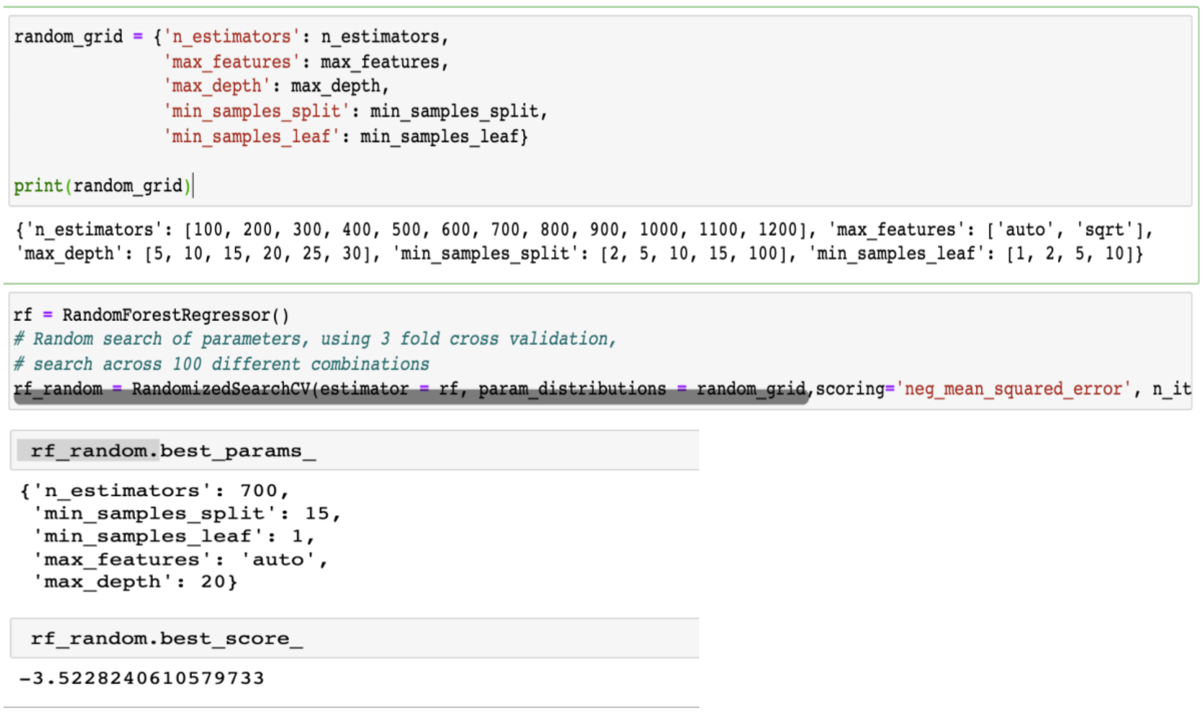
```
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200]
```

```
 #Randomized Search CV
# Number of features to consider at every split
max_features = ['auto', 'sqrt'] # we first consider all the featurees and
#then sqare root number of features to train the model

# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
#we create trees with 5 10 15 for each model...and train it

# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]
# we split as 2 nodes forst then 5 then 10 like that till 100 from the list

# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 5, 10]
```

Figure 4.2: Randomized Search CV.



```
from sklearn.model_selection import RandomizedSearchCV
#Randomized search on hyper parameters.
#used to select the best parameter for the model
```



```
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf}

print(random_grid)
```

```
{'n_estimators': [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200], 'max_features': ['auto', 'sqrt'],
'max_depth': [5, 10, 15, 20, 25, 30], 'min_samples_split': [2, 5, 10, 15, 100], 'min_samples_leaf': [1, 2, 5, 10]}
```

```
rf = RandomForestRegressor()
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,scoring='neg_mean_squared_error', n_it
```

```
rf_random.best_params_
```

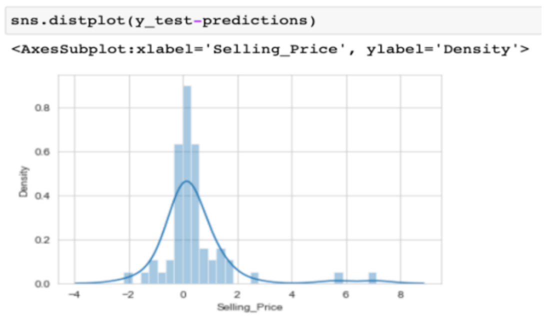
```
{'n_estimators': 700,
 'min_samples_split': 15,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': 20}
```

```
rf_random.best_score_
```

```
-3.5228240610579733
```

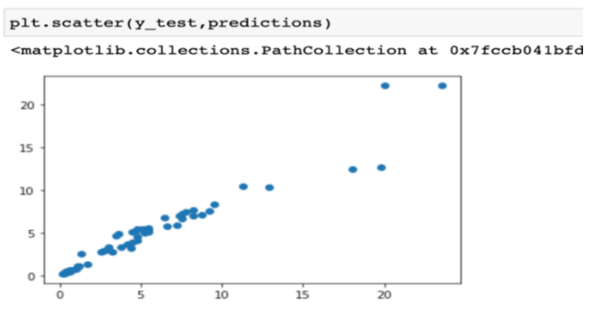
Figure 4.3: Randomazied Search.

Additionally, you can use the istplot() function to visually assess how well a parametric distribution matches the observed data by fitting it to a dataset **(Figure 4.4)**. It should to be a closed Gaussian distributed graph, and there ought to be little variation between the predictions and the y_test (actual value).



```
sns.distplot(y_test-predictions)
<AxesSubplot:xlabel='Selling_Price', ylabel='Density'>
```

Figure 4.4: Distplot.



```
plt.scatter(y_test,predictions)
<matplotlib.collections.PathCollection at 0x7fccb041bfd
```

Figure 4.5: Scatter.

Dots are used in scatter plots to depict the relationship between variables and are used to observe relationships between variables. Here, the points are almost in a straight line (**Figure 4.5**).

4.2.Checking accuracy of the model

In order to describe how well a machine learning model is performing in its predictions, the process of developing machine learning models should include evaluating the model accuracy. The RMSE, MAE, and MSE metrics are commonly used in regression analysis to evaluate model performance and prediction error rates.

RMSE: The error rate by the square root of MSE.

MAE :The absolute difference that results from averaging the original and predicted values across the data set is known as the Mean Absolute Error.

MSE: The difference between the original and predicted values is represented as the Mean Square Error, as determined by squaring the mean difference in the data set.

```
from sklearn.metrics import mean_squared_error,mean_absolute_error
mse_predict = round(mean_squared_error(y_test,predictions),4)
mae_predict = round(mean_absolute_error(y_test,predictions),4)
print ('MSE is:'+str(mse_predict))
print ('MAE is:'+str(mae_predict))
```

```
MSE is:1.9194
MAE is:0.7154
```

```
from sklearn.metrics import r2_score
r = r2_score(y_test, predictions)
print("R2 score : {}" . format(r))
```

```
R2 score : 0.9240646284310525
```

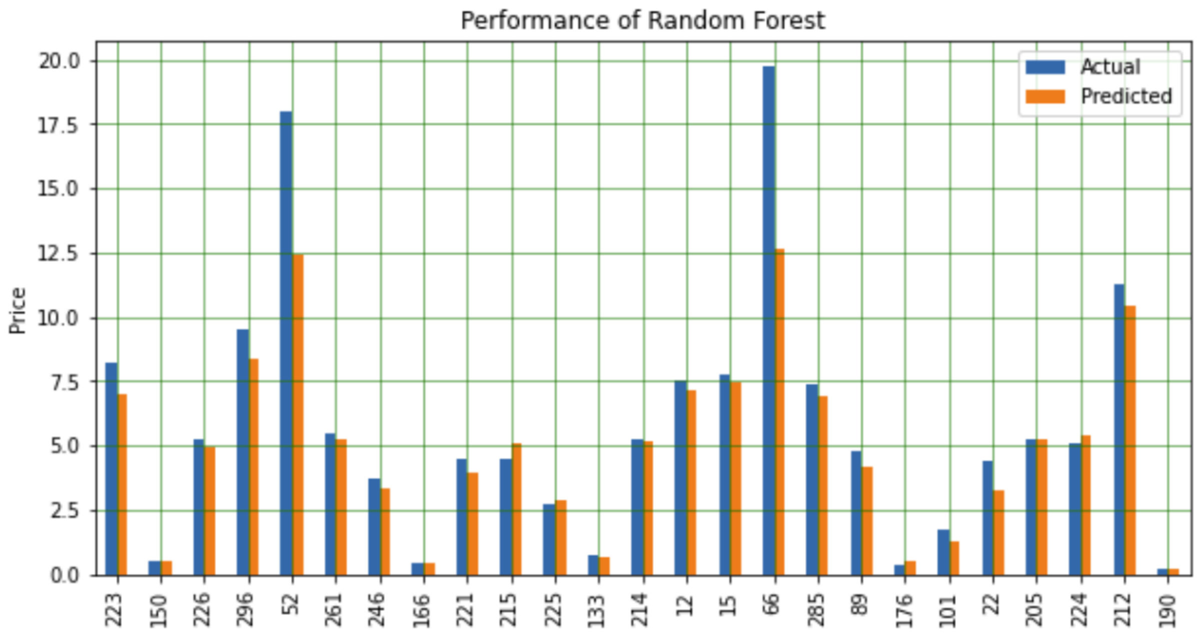
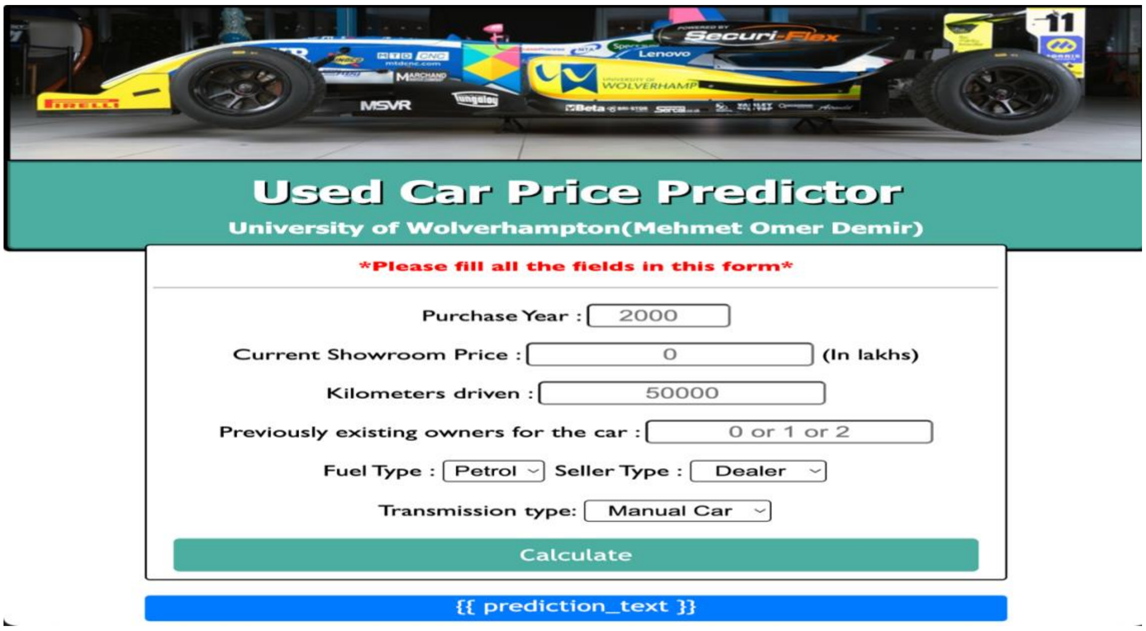


Figure 4.6: Prediction and Final Score.

4.3.Model Deployment

One of the final stages of any machine learning project is to display the model on the website. Here we will create a useful user interface. We used a flask to generate an HTML file for the car price prediction. This calculates the selling price of a used car using the input values for each feature, as shown in the following image.



Used Car Price Predictor

University of Wolverhampton(Mehmet Omer Demir)

Please fill all the fields in this form

Purchase Year : 2000

Current Showroom Price : 0 (In lakhs)

Kilometers driven : 50000

Previously existing owners for the car : 0 or 1 or 2

Fuel Type : Petrol Seller Type : Dealer

Transmission type: Manual Car

Calculate

{{ prediction_text }}

Figure 4.7:HTML page of project

5.Conclusion

- After the model was trained, we used displot to plot the differences between the test and train sets of data as well as the predictions produced. In a normal distribution with a mean of 0, this is displayed.

- As a result, the error is revealed to be small. This suggests that the predicted value may be correct.

- Additionally, when using a scatter plot, the test and predicted points fall along the line y=x, indicating that the model's predicted values are identical to the test values.

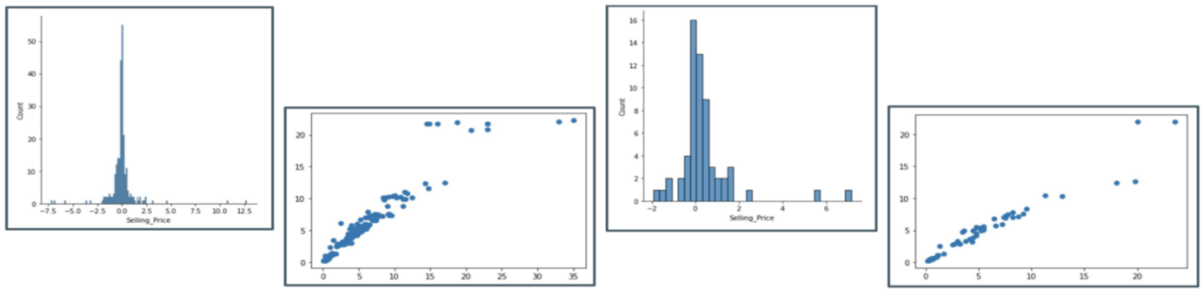
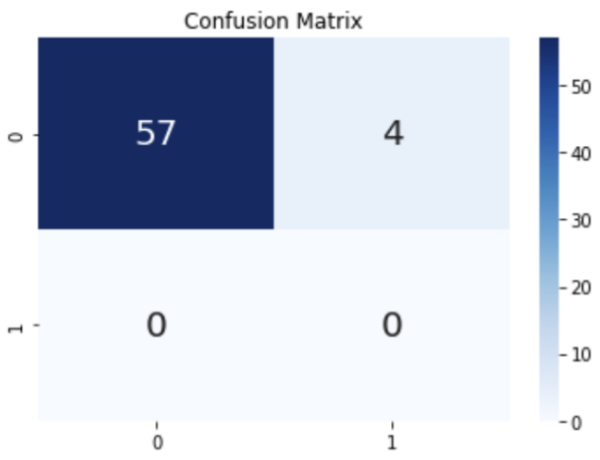


Figure 4.7: Result graphs.

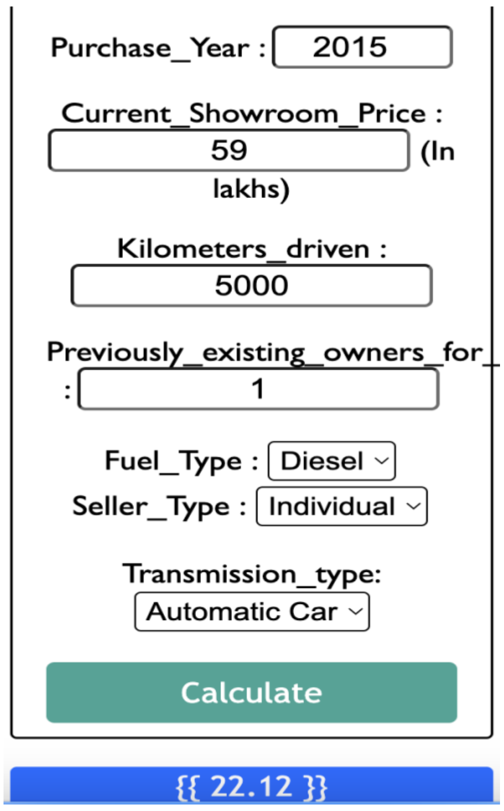
- The Root Mean Square Error, R2 Score, and Mean Square Error are used to determine the performance metrics.

Also accuracy of the model is calculated from the sklearn's built in functions. As this is only done for classification we made our predicted output using a cutoff. If the difference betwen the test and predicted value is greater than cutoff (MSE) then we classify as wrongly predicted else correct prediction.



- Additionally, the model's accuracy is determined using built-in Sklearn functions. Since this is only done for classification, we used a cutoff to create our predicted output. If the test and predicted values differ by more than the cutoff value (MSE), the prediction is deemed incorrect; otherwise, it is deemed correct.
- The Root Mean Square Error for the actual test results and predicted test results is 1.34.
- The test and predicted test values' Mean Square Error comes out to be 1.96.

6.Results



Purchase_Year : 2015

Current_Showroom_Price : 59 (In lakhs)

Kilometers_driven : 5000

Previously_existing_owners_for : 1

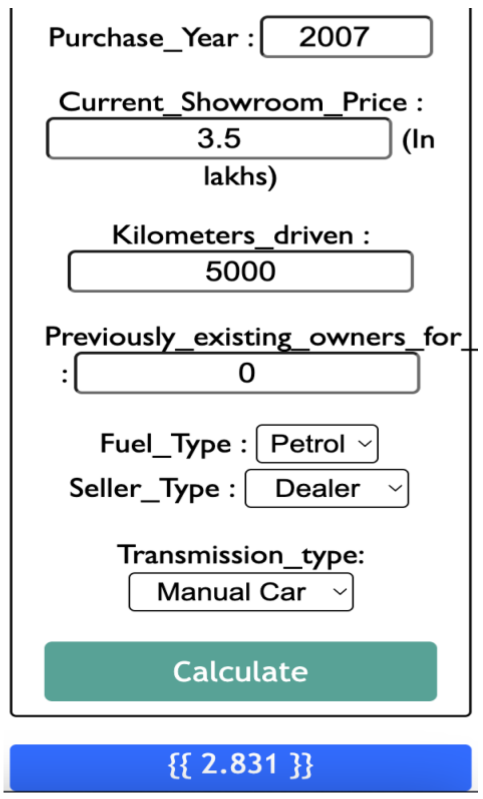
Fuel_Type : Diesel

Seller_Type : Individual

Transmission_type: Automatic Car

Calculate

{{ 22.12 }}



Purchase_Year : 2007

Current_Showroom_Price : 3.5 (In lakhs)

Kilometers_driven : 5000

Previously_existing_owners_for : 0

Fuel_Type : Petrol

Seller_Type : Dealer

Transmission_type: Manual Car

Calculate

{{ 2.831 }}

Figure 5.1: Results.

Random forest regression is exceptionally well-suited to predict car prices, thanks to the algorithm's capability to deal with complex and non-linear relationships among input and output variables. Since cars comprise intricate machines, several factors determine a car's price, including make, model, year, mileage, engine size, and transmission type, amongst others.

These features may interact in exceedingly complex and non-linear manners to influence the final price. Random forest regression can accurately capture these complex relationships and furnish accurate price estimates, even when there are interactions between the input variables. Random forest regression is further advantageous since it can manage missing data and outliers effectively. In actual scenarios, data quality might be below perfect, with some missing values or outliers that require adequate handling.

Random forest regression tackles these environmental challenges by consuming all accessible data points and employing diverse decision trees to mitigate the impact of outliers. However, it remains pertinent to recognize that the model's performance relies on various factors, such as data quality, feature selection, and model hyperparameters tuning.

In summary, the use of random forest regression in car price prediction is a powerful and effective apprach that can handle complex relationships between input variables and output variables.

References

[1] Adi Bronshtein (2017). *A Quick Introduction to the 'Pandas' Python Library*. [online] Medium. Available at: https://towardsdatascience.com/a-quick-introduction-to-the-pandas-python-library-f1b678f34673.

[2] Bruce, P., Bruce, A., & Gedeck, P. (2019). Practical Statistics for Data Scientists. O'Reilly Media, Inc.pp. 32-64.

[3] Gegic, E.; Isakovic, B.; Kec, D.; Masetic, Z.; Kevric, J. Car price prediction using machine learning techniques. TEM J. 2019, 7, 113.

[4] Geron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (2nd ed.). O'Reilly Media, Inc. pp. 45-67, 89-104.

[5] Grus, J. (2015). Data science from scratch: First principles with Python. O'Reilly Media, Inc.pp 50-55.

[6] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). An introduction to statistical learning: With applications in R. Springer.Inc.pp. 70-72.

[7] McKinney, W. (2018). Python for data analysis: Data wrangling with Pandas, NumPy, and IPython. O'Reilly Media, Inc.pp 30-35.

[8] Scikit-learn. (n.d.). *Nearest Neighbors Classification*. [online] Available at: https://scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html [Accessed 25 Apr. 2023].

[9] VanderPlas, J. (2016). Python Data Science Handbook: Essential Tools for Working with Data. O'Reilly Media, Inc. pp. 57-62, 120-125.

[10] www.kaggle.com. (n.d.). *Vehicle dataset*. [online] Available at: https://www.kaggle.com/datasets/nehalbirla/vehicle-dataset-from-cardekho.