FLIP ROBO

USED CAR PRICE PREDICTION PROJECT

Submitted by: Guduri Premsagar

ACKNOWLEDGMENT

I'd like to express my heartfelt gratitude to my SME (Subject Matter Expert) Khushboo Garg, as well as Flip Robo Technologies, for allowing me to work on this project on Used Car Price Prediction and for assisting me in conducting extensive research, which allowed me to learn a lot of new things, particularly in terms of data collection.

In addition, I used a few outside resources to help me finish the project. I made sure to learn from the samples and adjust things to fit my project's needs. All the external resources that were used in creating this project are listed below:

1) https://www.youtube.com/

2) https://www.google.com/

3) https://github.com/

4) https://scikit-learn.org/stable/user_guide.html

5) https://www.kaggle.com/

6) https://www.analyticsvidhya.com/

7) https://medium.com/

8) https://towardsdatascience.com/

INTRODUCTION

- ## Business Problem Framing

COVID-19's impact on the Indian automotive industry: Before the Covid-19 crisis, the Indian automotive sector was already in trouble. It had an almost 18% decline in overall growth. The commencement of the Covid-19 outbreak, as well as ongoing lockdowns across India and the rest of the world, aggravated the issue. Slow economic growth, negative consumer mood, BS-VI transition, changes to axle load norms, liquidity constraint, low capacity utilisation, and potential bankruptcies are all posing challenges for the Indian automotive business in the coming two years (FY20 and FY21). The restoration to near-normalcy of daily life and manufacturing activity in China and South Korea, as well as India's extended shutdown, provide promise for a U-shaped economic revival. According to our analysis, the Indian automobile sector will begin to revive in the third quarter of FY21. In FY21, we anticipate a 15% to 25% decrease in industry demand. OEMs, dealers, and suppliers with large cash reserves and easy access to finance will be better positioned to weather the storm. The auto industry has been hampered by a combination of demand and supply issues. There are, however, certain good results that we will examine.

The car sector will suffer as India's GDP growth rate for FY21 is reduced from 5% to 0% and then to (-5%). Job creation and income levels are highly correlated with auto demand, and both have been impacted. The revenue impact on India's auto industry, according to the CII, is expected to be $2 billion each month.

The supply chain could be the hardest hit. Supply chain difficulties are expected to continue even after China recovers. Problems on the Indo-China border in Ladakh are making matters worse. Domestic providers are pitching in, but demand remains sluggish, resulting in an inventory surplus.

The release of the Unlock 1.0 will correspond with the application of the BS-VI standards, which will result in greater reductions for both dealers and customers. Even if automakers manage their expenses, discounts will have a significant influence on profits.

In the post-COVID-19 scenario, the true pain may be felt by dealers, who are likely to be dealing with surplus inventory and a lack of finance options. The BS-VI price hikes are also expected to have an impact on auto demand. COVID-19 has brought about two beneficial developments. The "Make in India" movement is being forced to invest heavily due to the China supply chain shock. The COVID-19 dilemma has exposed flaws in the automobile business paradigm, and it may serve as a catalyst for a major shift toward electric automobiles (EVs). That might be a huge plus for the auto industry.

- ## Conceptual Background of the Domain Problem

The expanding world of e-commerce includes everything you'd expect to find at a general store, not just electronics and clothing. Putting aside the common retail perspective and looking at the big picture, the digital marketplace sees thousands, if not millions, of transactions every day. The vehicle industry, which involves the purchasing and selling of used cars, is one of the most expanding markets in the digital space. To receive a used automobile price quote, we sometimes have to walk up to the dealer or private sellers. However, when it comes to used car appraisal, or second-hand car valuation, buyers and sellers confront a huge stumbling block. In the past, you would visit a dealership and have your automobile inspected before learning the price. So, rather than doing all of these things, we may create a machine learning model that uses various attributes of used automobiles to forecast the exact and valuable car price.

- ## Review of Literature

  This project focuses on the data's exploration, feature engineering, and classification capabilities. We can do better data exploration and extract some interesting characteristics utilising the provided columns because we scrape a large amount of data that includes more automobile related variables.

  The purpose of this project is to create an application that can forecast automobile prices using various features. In the long run, this would allow consumers in an increasingly digital society to better discuss and review their purchases with one another.

- ## Motivation for the Problem Undertaken

  I realised how each independent feature helped me grasp the data based on the issue statement and real-time data scraped from the OLX and Cars24 websites, as each feature delivers a distinct kind of information. Working with many forms of real-time data in a single data set and performing root cause analysis to anticipate the price of a used car is quite exciting. Based on the car's model, miles travelled, transmission type, fuel type, and other factors. I would be able to estimate the price of a used car because the client will use this model to understand how costs fluctuate depending on the variables. They might work on it properly and devise some ways to sell the used car for a profit. Furthermore, the model will be a good way for the client to understand the pricing dynamics of a used car.

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

In our scrapped dataset, our target variable "Used Car Cost " may be a persistent variable. Hence, we are going be dealing with this displaying issue as regression.

This project is done in two parts:

- Data Collection phase
- Model Building phase

Data Collection phase:

You must scrape data on at least 5000 used autos. You can also scrape more data if you want to. The more data there is, the better the model. This section requires web scraping to get data on used automobiles from websites (OLX, OLA, Car Dekho, Cars24, and so on). You must collect data from various sites. The amount of data columns is limitless; it's entirely up to you and your imagination. Brand, model, variant, manufacture year, driven km, fuel, number of owners, location, and the objective variable Price of the car are the most common columns. This information is provided to give you an idea of crucial variables in a used car model. You can make adjustments to it, such as adding or removing columns, depending on the website from which you are obtaining the information. Include all sorts of vehicles in your data, such as SUVs, sedans, coupes, minivans, and hatchbacks.

Model Building phase:

After you've gathered your data, you'll need to create a machine learning model. Complete all data pre-processing processes prior to developing the model. Select the optimal model by experimenting

with various hyper parameters. Follow the entire data science life cycle. Include all of the following steps:

1. Data Cleaning
2. Exploratory Data Analysis (EDA)
3. Data Pre-processing and Visualisation
4. Model Building
5. Model Evaluation
6. Selecting the best model

- ## Data Sources and their formats

The dataset is in CSV (Comma Separated Value) format and contains 6 columns (5 features and 1 label) with a total of 10000 records, as shown below:

- Used Car Model - This shows the car model names
- Year of Manufacture - Gives us the year in which the car was made
- Kilometres Driven - Number of kilometres the car the driven reflecting on the Odometer
- Fuel Type - Shows the fuel type used by the vehicle
- Transmission Type - Gives us the manual or automatic gear shifting mechanism
- Used Car Price - Lists the selling price of the used cars

Our dataset comprises a column with the target label "Utilized Car Price," and the remaining feature columns can be used to determine or assist in forecasting the price of used cars. Due to the fact that price is a continuous variable, this is a regression problem!

```
#importing dataset

df = pd.read_csv(r"../Car Price Prediction/used_cars.csv", )
```

```
#checking df data
df
```

	LOCATION	MNF_YEAR	BRAND	MODEL	VARIANT	DRIVEN_KM	FUELTYPE	NOOF_OWNERS	PRICE
0	New Delhi	2018	Hyundai	Creta 1.6 E + VTVT	Manual	39,294 km	Petrol	2nd Owner	₹8,98,799
1	New Delhi	2016	Maruti	Vitara Brezza ZDI	Manual	92,569 km	Diesel	1st Owner	₹6,69,099
2	New Delhi	2018	Maruti	Vitara Brezza VDI OPT	Manual	60,395 km	Diesel	1st Owner	₹6,83,399
3	New Delhi	2020	Hyundai	VENUE S MT 1.2 KAPPA	Manual	3,393 km	Petrol	1st Owner	₹6,78,199
4	New Delhi	2017	Toyota	Innova Crysta Touring Sport Diesel AT	Automatic	85,295 km	Diesel	1st Owner	₹15,00,699
...	...	...	...	...	...	...	...	...	...
7681	Rohini Sector 14, Delhi	2010	Hyundai	Executive CNG	MANUAL	80000.0 KM	CNG & HYBRIDS	1st	2,10,000
7682	Rohini Sector 14, Delhi	2014	Hyundai	Magna	MANUAL	34000.0 KM	PETROL	1st	3,10,000
7683	Thrikkakara, Kochi	2019	Mercedes Benz Fortuner	3.0 4x4 Automatic	AUTOMATIC	70000.0 KM	DIESEL	1st	36,50,000
7684	Nana Varachha, Surat	2013	Hyundai	Diesel Sportz	MANUAL	58250.0 KM	DIESEL	2nd	3,51,000
7685	Nana Varachha, Surat	2012	Maruti	DDiS VDI	MANUAL	58250.0 KM	DIESEL	2nd	3,25,000

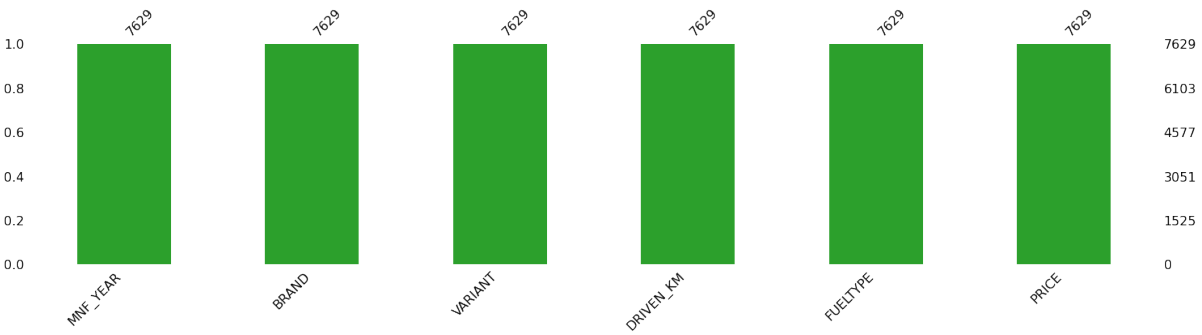
7686 rows × 9 columns

- ## Data Preprocessing Done

For the data pre-processing stage, I checked the dataframe for missing values and handled them by imputing records with "-" and other imputing strategies.

```
# checking again null values

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

```
MNF_YEAR      0
BRAND         0
VARIANT       0
DRIVEN_KM     0
FUELTYPE      0
PRICE         0
dtype: int64
```



I looked at the datatype specifications for each column to figure out which ones were numeric and how to convert them.

```
# checking info to know non-null, count & datatype information

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7629 entries, 0 to 7685
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   MNF_YEAR    7629 non-null   int64 
 1   BRAND       7629 non-null   object
 2   VARIANT     7629 non-null   object
 3   DRIVEN_KM   7629 non-null   object
 4   FUELTYPE    7629 non-null   object
 5   PRICE       7629 non-null   object
dtypes: int64(1), object(5)
memory usage: 417.2+ KB
```

I also looked at all of the unique values in each of the columns before deciding how to deal with the imputation part.

```
# checking unique values
df.nunique().sort_values().to_frame("Unique Values")
```

	Unique Values
VARIANT	5
FUELTYPE	11
MNF_YEAR	29
BRAND	102
PRICE	1582
DRIVEN_KM	1790

The code for the various data imputations done on our data set is presented below.

```
#Data pre-processing

df["PRICE"]=df["PRICE"].apply(lambda x: x.replace('₹','') if x!='' else '')

df["DRIVEN_KM"]=df["DRIVEN_KM"].apply(lambda x: x.replace('.0','') if x!='' else '')
df["DRIVEN_KM"]=df["DRIVEN_KM"].apply(lambda x: x.replace('KM','') if x!='' else '')
df["DRIVEN_KM"]=df["DRIVEN_KM"].apply(lambda x: x.replace('km','') if x!='' else '')
df["DRIVEN_KM"]=df["DRIVEN_KM"].apply(lambda x: x.replace(',', ''))

df
```

```
# converting PRICE column - object datatype into float datatype
try:
    df["PRICE"]=df["PRICE"].apply(lambda x: x.split(' ')[1] if x!='-' else '0,0')
except IndexError:
    pass

try:
    df["PRICE"]=df["PRICE"].apply(lambda x: str(x.replace(',', ''))) #replacing ',' with '' no space in price column
except ValueError:
    pass

df["PRICE"]=df["PRICE"].str.strip() # removing extra white space from the column records
df["PRICE"]=pd.to_numeric(df["PRICE"].str.replace('-','0'), errors='coerce')
df["PRICE"]=df["PRICE"].astype(float) # converting object to float data type
df
```

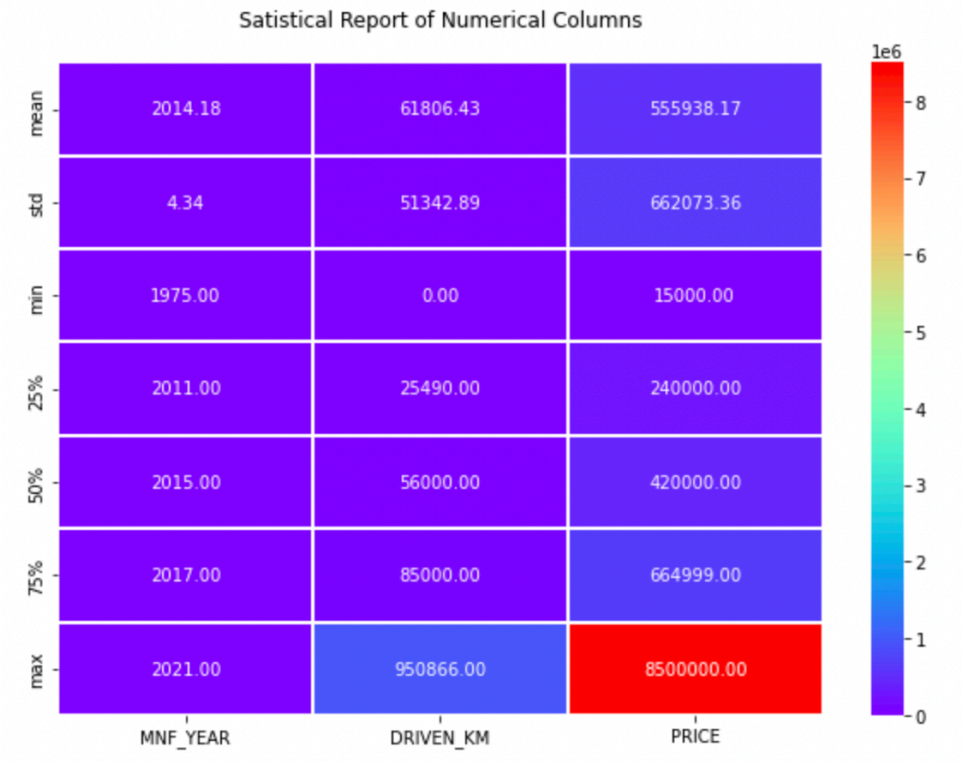
```
# replacing values

df["FUELTYPE"]=df["FUELTYPE"].apply(lambda x: x if x!='--' else 'Petrol') # replacing with common fuel type in india
df["FUELTYPE"]=df["FUELTYPE"].apply(lambda x: x if x!='PETROL' else 'Petrol') # replacing PETROL to Petrol.
df["FUELTYPE"]=df["FUELTYPE"].apply(lambda x: x if x!='DIESEL' else 'Diesel') # replacing DIESEL to Diesel.


df["VARIANT"]=df["VARIANT"].apply(lambda x: x if x!='--' else 'Manual') # common Variant is manual
df["VARIANT"]=df["VARIANT"].apply(lambda x: x if x!='MANUAL' else 'Manual')
df["VARIANT"]=df["VARIANT"].apply(lambda x: x if x!='PETROL' else 'Petrol')
df["VARIANT"]=df["VARIANT"].apply(lambda x: x if x!='AUTOMATIC' else 'Automatic')
df["BRAND"]=df["BRAND"].apply(lambda x: x if x!='--' else 'Hyundai') # common used car model

df["DRIVEN_KM"]=df["DRIVEN_KM"].apply(lambda x: x if x!='--' else 'None')

avg_usedcar_price=df["PRICE"].mean()
df["PRICE"]=df["PRICE"].apply(lambda x: x if x!='--' else avg_usedcar_price) # average used car prices

df
```

The count, mean, standard deviation, minimum, maximum, 25%, 50%, and 75% quartile data were then checked using the "describe" technique.

```
#checking describe
df.describe(include="all")
```

	MNF_YEAR	BRAND	VARIANT	DRIVEN_KM	FUELTYPE	PRICE
count	7629.000000	7629	7629	7629.000000	7629	7.629000e+03
unique	NaN	102	2	NaN	8	NaN
top	NaN	Maruti	Manual	NaN	Petrol	NaN
freq	NaN	1567	6440	NaN	4076	NaN
mean	2014.184690	NaN	NaN	61806.425088	NaN	5.559382e+05
std	4.338351	NaN	NaN	51342.886853	NaN	6.620734e+05
min	1975.000000	NaN	NaN	0.000000	NaN	1.500000e+04
25%	2011.000000	NaN	NaN	25490.000000	NaN	2.400000e+05
50%	2015.000000	NaN	NaN	56000.000000	NaN	4.200000e+05
75%	2017.000000	NaN	NaN	85000.000000	NaN	6.649990e+05
max	2021.000000	NaN	NaN	950866.000000	NaN	8.500000e+06

I also took a look at the numeric component and noted that the maximum value for the Used Car Price column was at a higher scale.



- ## Data Inputs- Logic- Output Relationships

Because the incoming data was all object datatype, it was necessary to clean it up by deleting unnecessary information such as "km" from the Kilometres Driven column and ensuring that the numeric data was translated appropriately. I then converted all of the categorical feature columns to numeric representation using the Ordinal Encoding technique.

Code:

```
# Ordinal Encoder

oe = OrdinalEncoder()
def ordinal_encode(df, column):
    df[column] = oe.fit_transform(df[column])
    return df

column=["VARIANT", "FUELTYPE", "BRAND"]
df=ordinal_encode(df, column)
df
```

Made use of Z score method to remove outliers that were present on our dataset.

```
# Using Z Score to remove outliers

z = np.abs(zscore(df))
threshold = 3
df1 = df[(z<3).all(axis = 1)]

print ("Shape of the dataframe before removing outliers: ", df.shape)
print ("Shape of the dataframe after removing outliers: ", df1.shape)
print ("Percentage of data loss post outlier removal: ", (df.shape[0]-df1.shape[0])/df.shape[0]*100)

df=df1.copy() # reassigning the changed dataframe name to our original dataframe name
```

```
Shape of the dataframe before removing outliers:  (7629, 6)
Shape of the dataframe after removing outliers:  (7338, 6)
Percentage of data loss post outlier removal:  3.814392449862367
```

I used the Log transformation technique to deal with the skewness, ensuring that at the very least a bell shape curve closer to normal distribution is achieved.

```
# Using Log Transform to fix skewness

df_log=df.copy()
for col in df_log.columns:
    if df_log.skew().loc[col]>0.55:
        df_log[col]=np.log1p(df_log[col])
```

- ## Hardware and Software Requirements and Tools Used

Hardware technology being used:-

CPU: MacBook Pro

Chip: Apple M1 - 8 (4 performance and 4 efficiency)

GPU: 8 GB

RAM: 8 GB

Software technology being used:-

Programming language: Python

Distribution: Anaconda Navigator

Browser based language shell: Jupyter Notebook

Libraries/Packages specifically being used:-

Pandas, NumPy, matplotlib, seaborn, scikit-learn, pandas-profiling, missingno.

Model/s Development and Evaluation

- **Identification of possible problem-solving approaches (methods)**
    1. Clean the dataset from unwanted scraped details.
    2. Impute missing values with meaningful information.
    3. Encoding the categorical data to get numerical input data.
    4. Compare different models and identify the suitable model.
    5. R2 score is used as the primary evaluation metric.
    6. MSE and RMSE are used as secondary metrics.
    7. Cross Validation Score was used to ensure there are no overfitting our underfitting models.

- **Testing of Identified Approaches (Algorithms)**

Libraries and Machine Learning Regression models that were used in this project are shown below.

```
# importing required libraries

import pandas as pd    # Data Manipulation and exploring
import numpy as np     # Data Statistical Analysis
import seaborn as sns  # Statistical Data Visualization
import matplotlib.pyplot as plt  # Data Visualization
%matplotlib inline

import warnings  # to ignore warnings
warnings.simplefilter("ignore")
warnings.filterwarnings("ignore")

import missingno
import pandas_profiling
from sklearn import metrics
from scipy.stats import zscore
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingRegressor

from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV

#to save model
import joblib
```

All the regression machine learning algorithms used are:

- Linear Regression Model
- Ridge Regularization Model
- Lasso Regularization Model
- Support Vector Regression Model
- Decision Tree Regression Model
- Random Forest Regression Model
- K Neighbours Regression Model
- Gradient Boosting Regression Model
- Ada Boost Regression Model
- Extra Trees Regression Model

• Run and Evaluate selected models

I constructed a Regression Machine Leaning Model function that includes the evaluation metrics so that we can acquire the data we need for all of the models listed above.

Code:

Machine Learning Model for Regression with Evaluation Metrics

```
# Regression Model Function

def reg(model, X, Y):
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=251)

    # Training the model
    model.fit(X_train, Y_train)

    # Predicting Y_test
    pred = model.predict(X_test)

    # RMSE - a lower RMSE score is better than a higher one
    rmse = mean_squared_error(Y_test, pred, squared=False)
    print("RMSE Score is:", rmse)

    # R2 score
    r2 = r2_score(Y_test, pred, multioutput='variance_weighted')*100
    print("R2 Score is:", r2)

    # Cross Validation Score
    cv_score = (cross_val_score(model, X, Y, cv=5).mean())*100
    print("Cross Validation Score:", cv_score)

    # Result of r2 score minus cv score
    result = r2 - cv_score
    print("R2 Score - Cross Validation Score is", result)
```

Output:

```
# Linear Regression Model

model=LinearRegression()
reg(model, X, Y)
```

```
RMSE Score is: 0.5240734208725968
R2 Score is: 55.09064471975369
Cross Validation Score: 36.87677102301901
R2 Score - Cross Validation Score is 18.21387369673468
```

- ## Key Metrics for success in solving problem under consideration

RMSE Score:

The root mean square error (RMSE) is the residuals' standard deviation (prediction errors). Residuals are a measure of how far the data points are from the regression line; RMSE is a measure of how spread out these residuals are. In other words, it indicates how tightly the data is clustered around the line of best fit.

R2 Score:

The R2 score is a critical indicator for assessing the effectiveness of a regression-based machine learning model. It's also known as the coefficient of determination and is pronounced R squared. It operates by calculating the amount of variation in the dataset-explained predictions.

Cross Validation Score:

The statistical method of cross-validation is used to measure the skill of machine learning models. It is frequently used in applied machine learning to compare and select a model for a specific predictive modelling problem since it is simple to grasp, simple to implement, and produces skill estimates with lower bias than other methods. The k-fold cross validation process is used to determine the model's skill on new data. You can use a variety of methods to choose the value of k for your dataset (I have used 5-fold validation in this project). In scikit-learn, there are several regularly used cross-validation variations, such as stratified and repeated.

Hyper Parameter Tuning:

The task of selecting a collection of ideal hyperparameters for a learning algorithm is known as hyperparameter optimization or tuning in machine learning. A hyperparameter is a value for a parameter that is used to influence the learning process. Other factors, such as node weights, are, on the other hand, learned.

Code:

```
# Choosing Extra Trees Regressor

fmod_param = {'n_estimators' : [100, 200, 300],
              'criterion' : ['squared_error', 'mse', 'absolute_error', 'mae'],
              'n_jobs' : [-2, -1, 1],
              'random_state' : [42, 251, 340]
             }

GSCV = GridSearchCV(ExtraTreesRegressor(), fmod_param, cv=5)
GSCV.fit(X_train,Y_train)
```

```
GridSearchCV(cv=5, estimator=ExtraTreesRegressor(),
             param_grid={'criterion': ['squared_error', 'mse', 'absolute_error',
                                       'mae'],
                         'n_estimators': [100, 200, 300], 'n_jobs': [-2, -1, 1],
                         'random_state': [42, 251, 340]})
```

Final model score after plugging in the best parameter values:

```
Final_Model = ExtraTreesRegressor(criterion='mse', n_estimators=300, n_jobs=-1, random_state=42)
Model_Training = Final_Model.fit(X_train, Y_train)
fmod_pred = Final_Model.predict(X_test)
fmod_r2 = r2_score(Y_test, fmod_pred, multioutput='variance_weighted')*100
print("R2 score for the Best Model is:", fmod_r2)
```

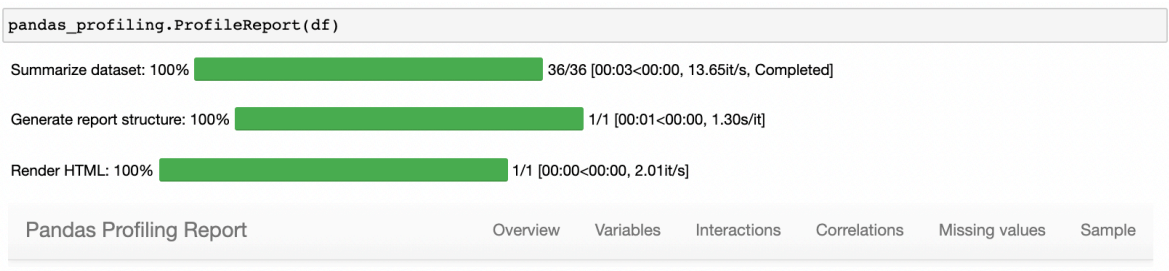
```
R2 score for the Best Model is: 73.45479082360302
```

- ## Visualizations

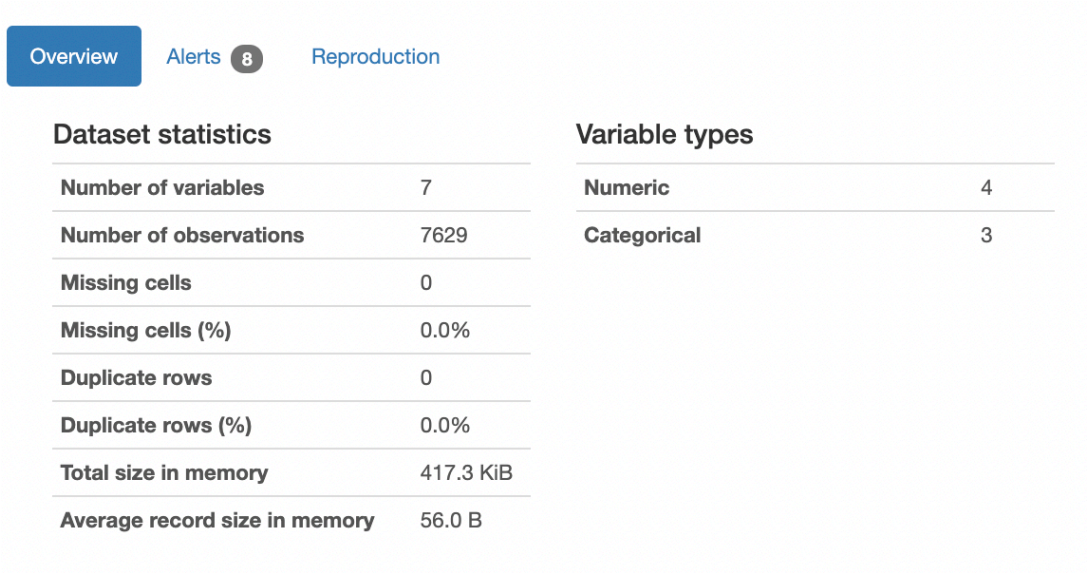
I generated an initial thorough report on my dataframe values using the pandas profiling tool. It provides us with information like as correlations, missing values, duplicate rows, variable types, memory capacity, and so on for the produced dataset. This aids us in a more detailed visualisation by separating each portion one by one and comparing and researching the effects of all the accessible feature columns on the prediction of our target label.

Code:

```
pandas_profiling.ProfileReport(df)
```



Output:



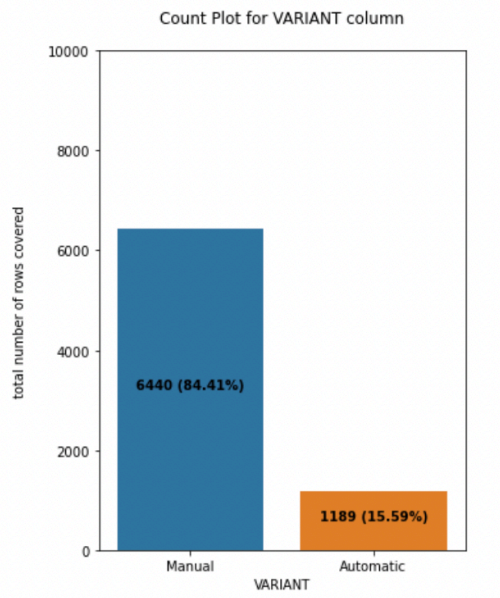
pandas-profiling is a free Python module that allows us to perform exploratory data analysis with just a few lines of code. It creates interactive web reports that may be delivered to anyone, even if they have no programming experience. It also provides report production for the dataset, with a variety of features and customizations. In other words, pandas-profiling saves us the time and effort of seeing and comprehending each variable's distribution. It generates a report with all of the data in one place.

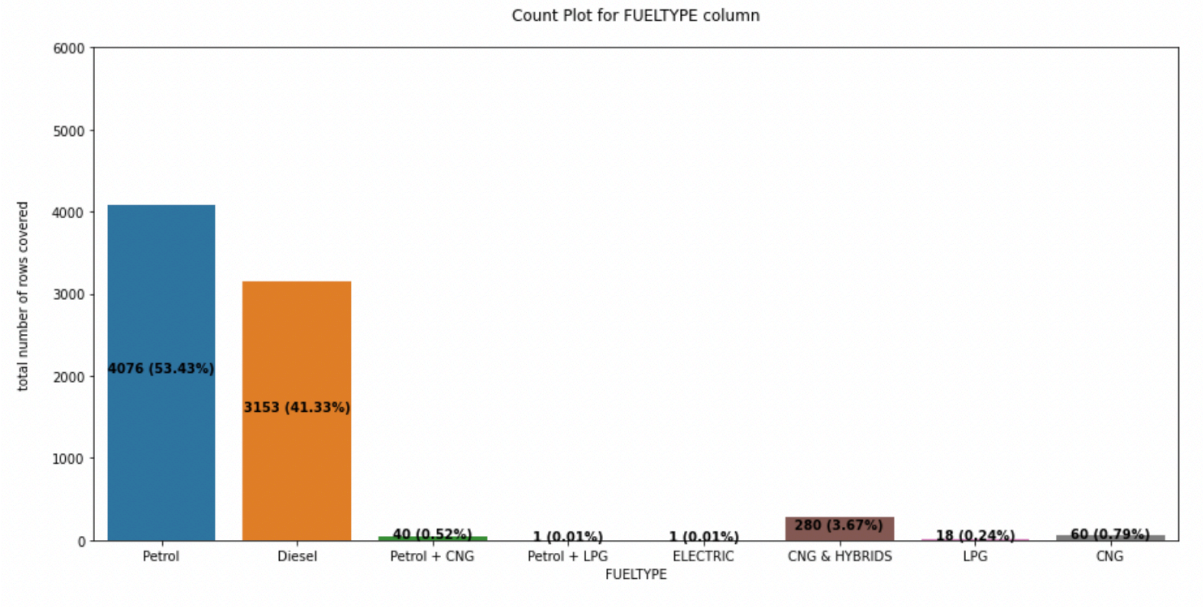
To visualise the datapoints in our column records, I created count plots, bar plots, pair plots, heatmaps, and other visualisations.

Code:

```
try:
    x = 'VARIANT'
    k=0
    plt.figure(figsize=[5,7])
    axes = sns.countplot(df[x])
    for i in axes.patches:
        ht = i.get_height()
        mr = len(df[x])
        st = f"{ht} ({round(ht*100/mr,2)}%)"
        plt.text(k, ht/2, st, ha='center', fontweight='bold')
        k += 1
    plt.ylim(0,10000)
    plt.title(f'Count Plot for {x} column\n')
    plt.ylabel(f'total number of rows covered\n')
    plt.show()

except Exception as e:
    print("Error:", e)
    pass
```

Output:

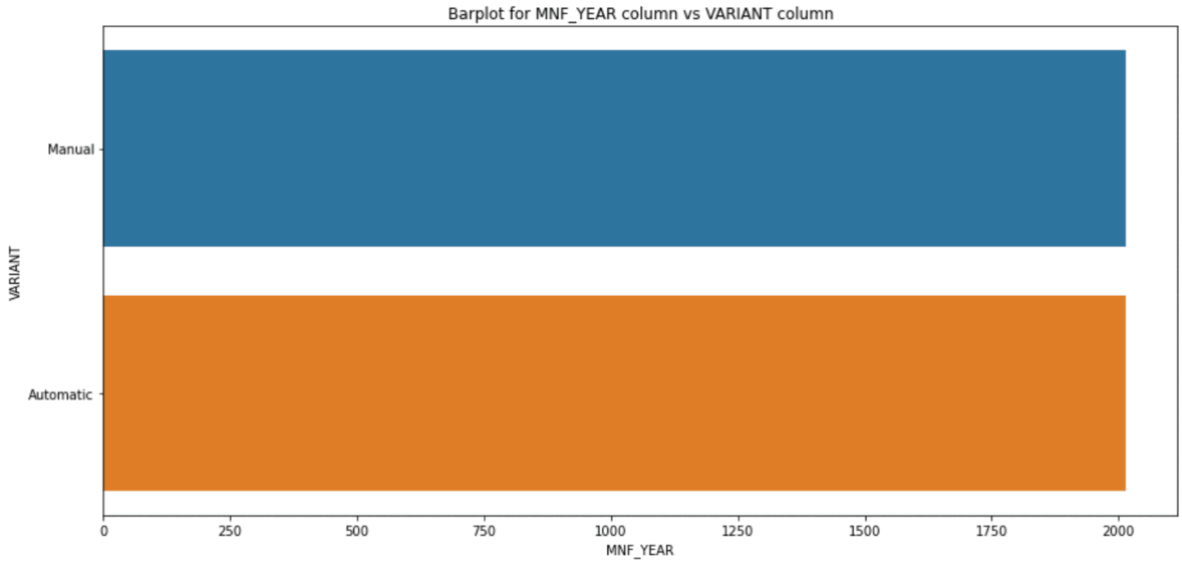




Code:

```
y = 'VARIANT'

x = 'MNF_YEAR'
plt.figure(figsize=[15,7])
sns.barplot(x,y,data=df,orient='h')
plt.title(f"Barplot for {x} column vs {y} column")
plt.show()

x = 'DRIVEN_KM'
plt.figure(figsize=[15,7])
sns.barplot(x,y,data=df,orient='h')
plt.title(f"Barplot for {x} column vs {y} column")
plt.show()

x = 'PRICE'
plt.figure(figsize=[15,7])
sns.barplot(x,y,data=df,orient='h')
plt.title(f"Barplot for {x} column vs {y} column")
plt.show()
```

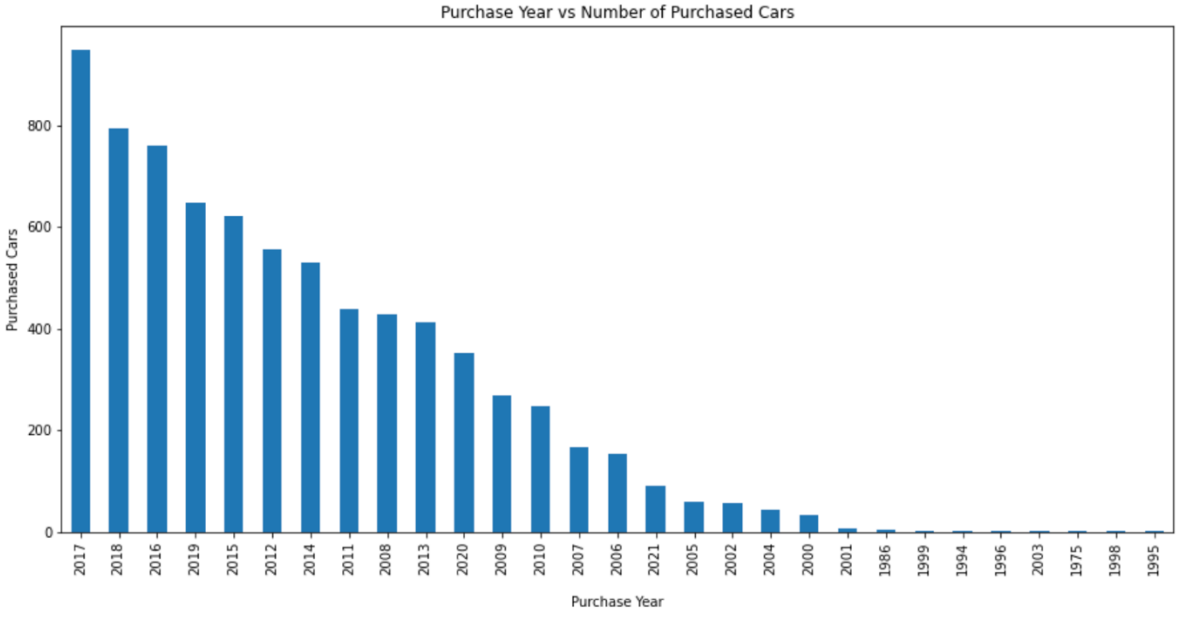
Output:



Code:

```
plt.figure(figsize=[15,7])
purchased_car_per_year = df['MNF_YEAR'].value_counts()
purchased_car_per_year.plot(kind='bar')
plt.xlabel("\nPurchase Year")
plt.ylabel("Purchased Cars")
plt.title("Purchase Year vs Number of Purchased Cars")
plt.show()
```

Output:

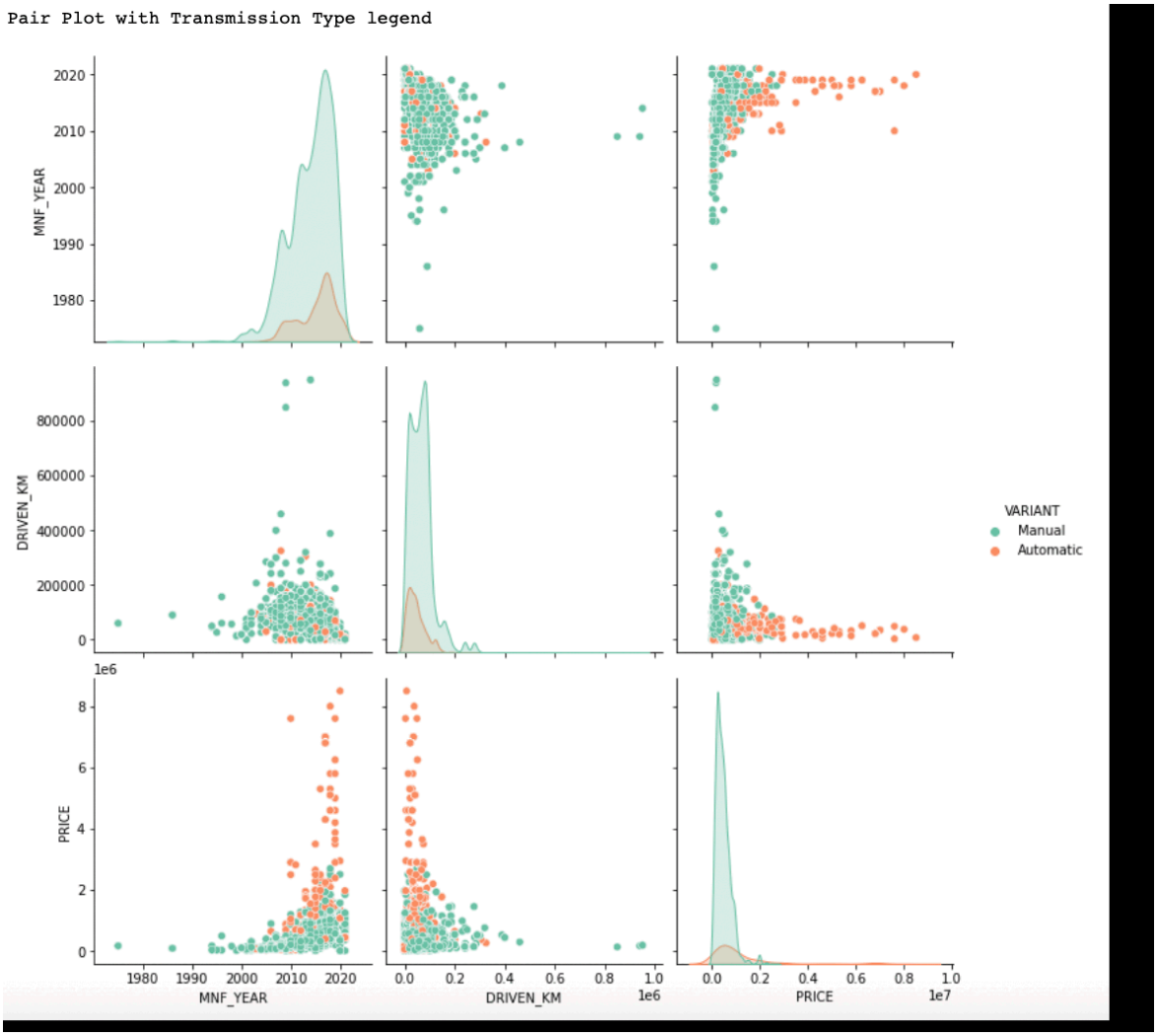


Code:

```
print("Pair Plot with Transmission Type legend")
sns.pairplot(df, hue='VARIANT', diag_kind="kde", kind="scatter", palette="Set2", height=3.5)
plt.show()
print("Pair Plot with Fuel Type legend")
sns.pairplot(df, hue='FUELTYPE', diag_kind="kde", kind="scatter", palette="tab10", height=3.5)
plt.show()
```

```
Manual = df[df['VARIANT']=='Manual']
Automatic = df[df['VARIANT']=='Automatic']

print('Manual VARIANT used car fuel details')
sns.pairplot(Manual, hue='FUELTYPE', diag_kind="kde", kind="scatter", palette="tab10", height=3.5)
plt.show()

print('Automatic VARIANT used car fuel details')
sns.pairplot(Automatic, hue='FUELTYPE', diag_kind="kde", kind="scatter", palette="hls", height=3.5)
plt.show()
```

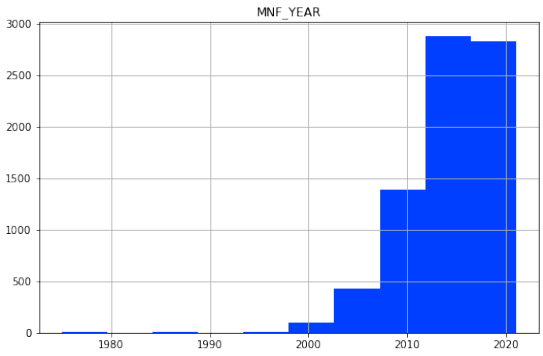
Output:

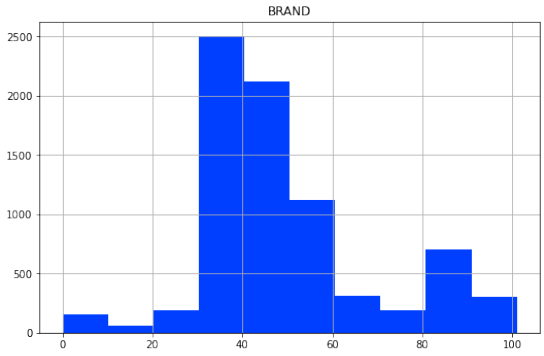


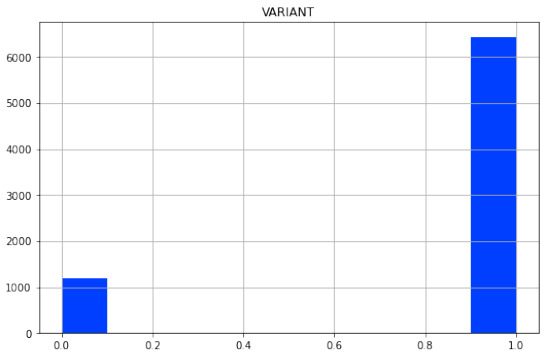
Code:

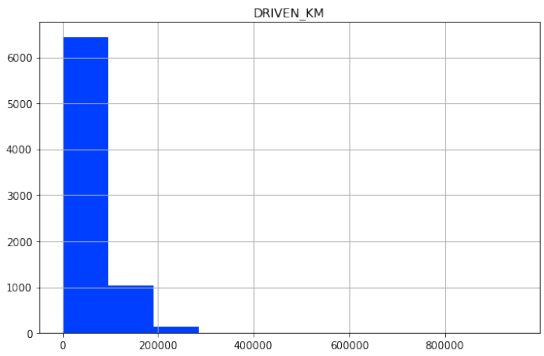
```
plt.style.use('seaborn-bright')

df.hist(figsize=(20,20))
plt.show()
```

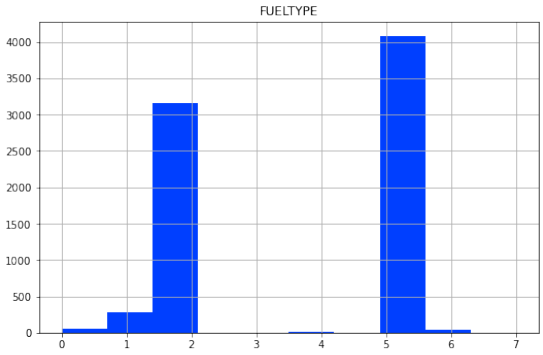
Output:

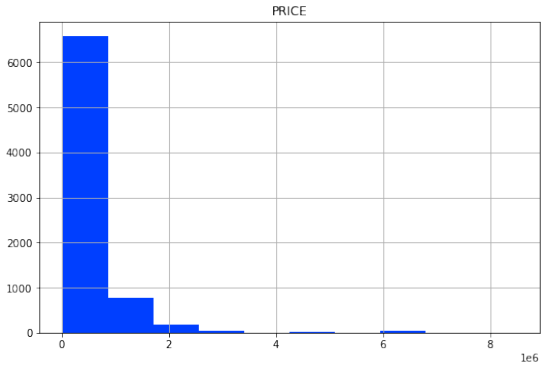








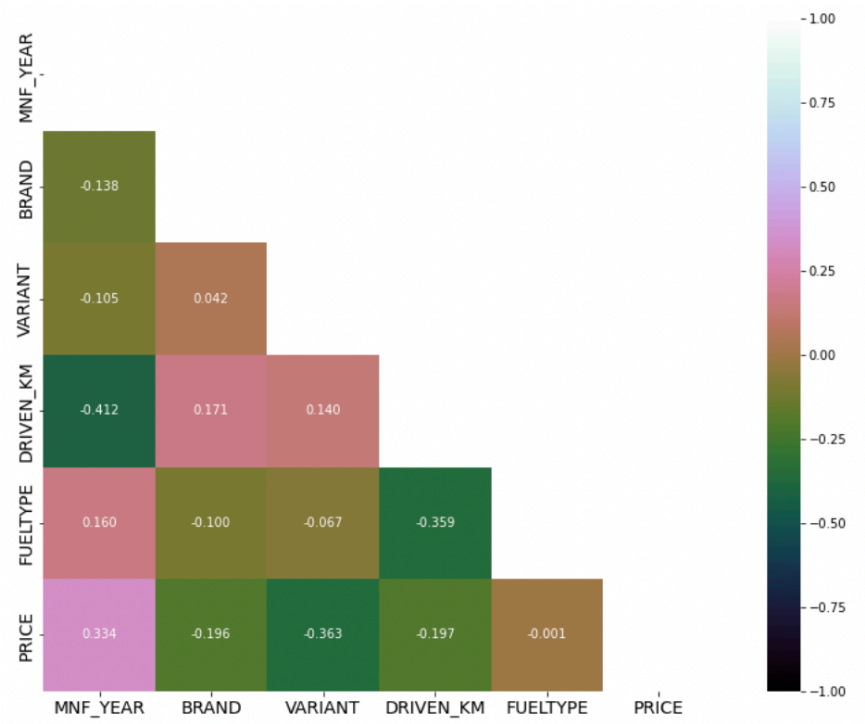




Code:

```
upper_triangle = np.triu(df.corr())
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(), vmin=-1, vmax=1, annot=True, square=True, fmt='0.3f',
            annot_kws={'size':10}, cmap="rainbow", mask=upper_triangle)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```

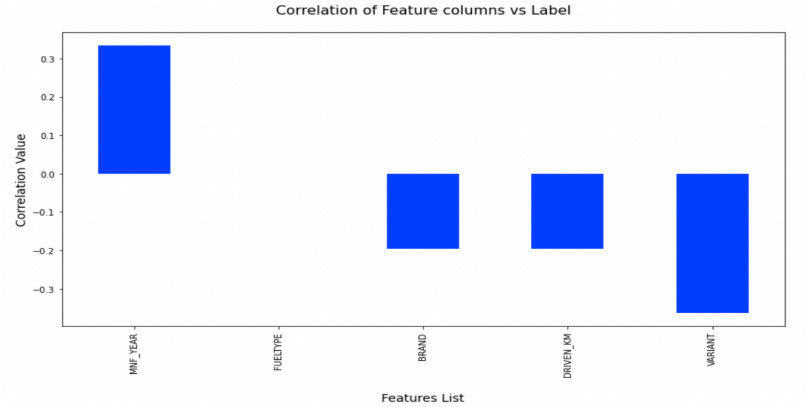
Output:



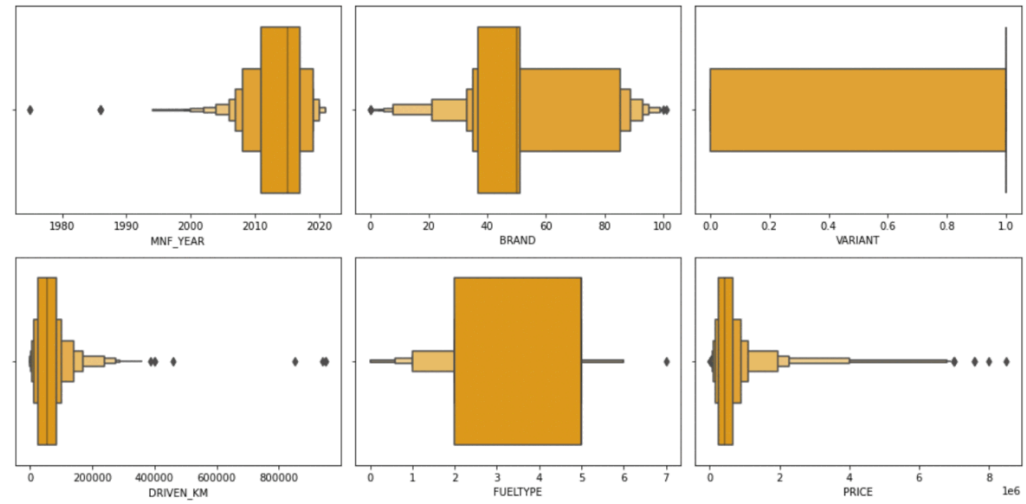
Code:

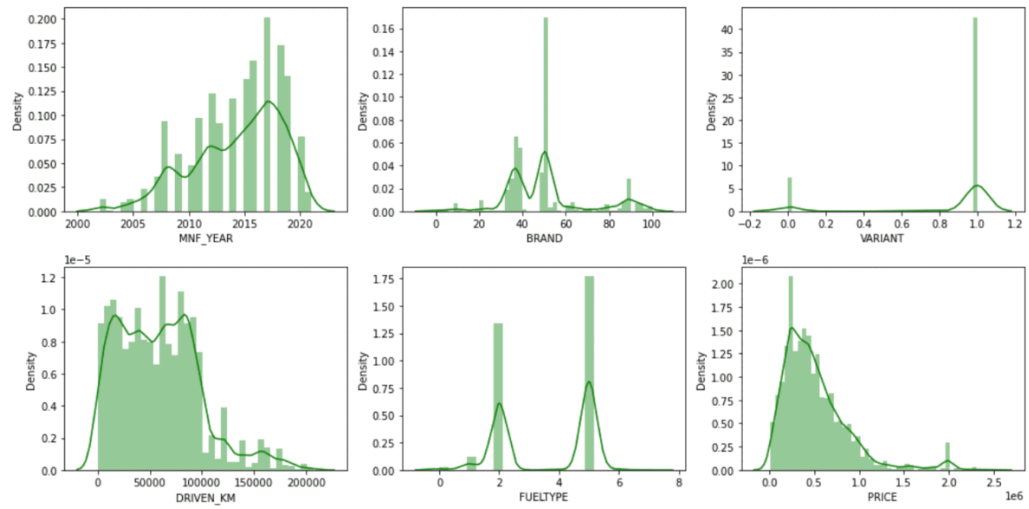
```
df_corr = df.corr()
plt.figure(figsize=(14,7))
df_corr['PRICE'].sort_values(ascending=False).drop('PRICE').plot.bar()
plt.title("Correlation of Feature columns vs Label\n", fontsize=16)
plt.xlabel("\nFeatures List", fontsize=14)
plt.ylabel("Correlation Value", fontsize=14)
plt.show()
```

Output:



Outliers and Skewness before and after treating them:

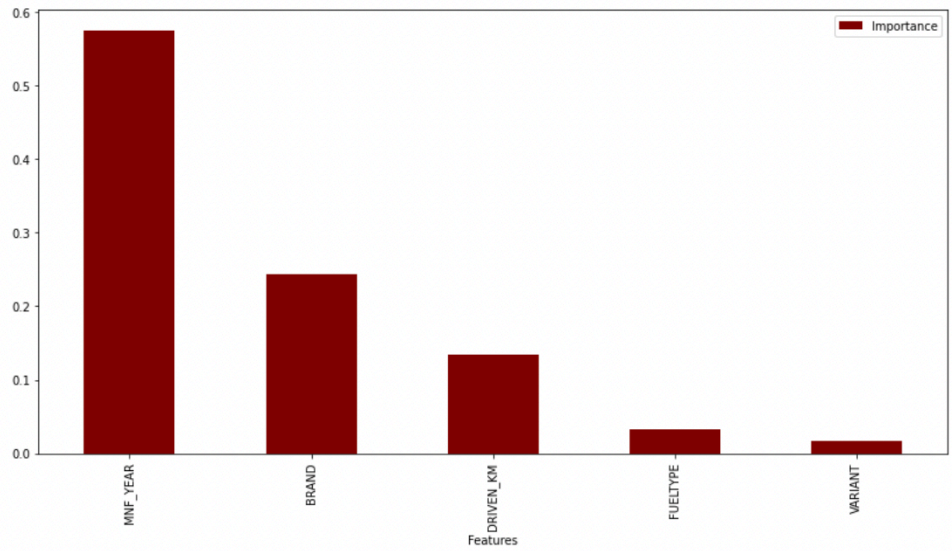




Code:

```
rf=RandomForestRegressor()
rf.fit(X_train, Y_train)
importances = pd.DataFrame({'Features':X.columns, 'Importance':np.round(rf.feature_importances_,3)})
importances = importances.sort_values('Importance', ascending=False).set_index('Features')
plt.rcParams["figure.figsize"] = (14,7)
importances.plot.bar(color='maroon')
importances
```

Output:



- ## Interpretation of the Results

The pictures show that features have an impact on the pricing of used cars. To avoid the production of a large number of columns, I encoded categorical columns using the ordinal encoder method rather than the single hot encoding approach. Furthermore, because our desired label contained continuous numeric data, a label encoder was not an option.

CONCLUSION

- ## Key Findings and Conclusions of the Study

We gained knowledge into how to collect data, pre-process the data, analyse the data, and construct a model after completing this project. First, we used Web Scraping to acquire data on used automobiles from several websites such as OLX, Car Dekho, Cars 24, OLA, and others. Beautiful Soup and Selenium were utilised for web scraping, which provides the advantage of automating our data collection process. We gathered about 10,000 pieces of information, including the selling price of used autos and other relevant details. The scraped data was then integrated into a single data frame and stored as a csv file, which we could read and analyse. We cleaned the data and performed data pre-processing tasks such as discovering and handling null values, removing words from numbers, converting objects to int types, data visualisation, and handling outliers and skewness, among other things. We started testing multiple machine learning regression techniques to discover the best performing model after separating our train and test data. According to their r2 score and cross validation scores, the Extra Tree Regressor Algorithm performed well. Then, using Grid Search CV, we used the Hyperparameter Tuning approach to find the optimal parameters and improve the score. Although the Extra Tree Regressor Algorithm did not do as well as the defaults, we decided to keep it for future forecasts because it was still better than the rest. After receiving a dataframe containing anticipated and real used car price details, we stored the final model in pkl format using the joblib package.

- ## Learning Outcomes of the Study in respect of Data Science

The data visualisation section assisted me in understanding the data by providing a graphical depiction of large amounts of data. It helped me comprehend the value of characteristics, discover

outliers/skewness, and compare independent and dependent features. Because data cleaning is the most critical element of model construction, I made sure the data was cleaned and scaled before starting. I created several regression machine learning models in order to find the best one, and Extra Trees Regressor Model was the best based on the metrics I utilised.

- ## Limitations of this work and Scope for Future Work

The limitations we faced during this project were:

The website was badly constructed because scraping took a long time and getting the next page was difficult. In addition, I need more practise with various web scraping approaches. There were more negative connected data than positive correlated data. Outliers and skewness were found, and we had to lose some valuable data while dealing with them. Because no instructions for dealing with these fast-paced websites were supplied, the web scraping process took longer.

Future Work Scope:

The current model is confined to used car data, but by training the model appropriately, it can be enhanced for various types of automobiles. By training the model with more precise data, the overall score can be increased even more.