USED CAR PRICE PREDICTION PROJECT

Submitted by:

Milind Mishra

ACKNOWLEDGMENT

I'd like to express my heartfelt gratitude to my SME (Subject Matter Expert) ‘Swati Mahaseth’, 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. The following are all of the external resources that were utilised to create this project:

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

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

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

4) https://github.com/

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

6) https://medium.com/

7) https://towardsdatascience.com/

8) https://www.analyticsvidhya.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 discounts 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. I would be able to model the price of a

used automobile based on a study of the car model, kilometres travelled, transmission type, fuel type, and other characteristics, and this model would then be used by the client to understand how the costs change with the variables.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

Our goal variable "Used Car Price" is a continuous variable in our discarded dataset. As a result, we'll approach this modelling challenge as a regression problem.

This project is divided into two sections:

- Phase of Data Collection

- Phase of Model Construction

i. Phase of data collection:

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. You must scrape the data of used automobiles from websites in this area (OLX, OLA, Car Dekho, Cars24 etc.) For this, you'll need to use web scraping. 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.

ii. Phase of Model Construction:

After collecting the data, you need to build a machine learning model. Before model building do all data pre-processing steps.

Try different models with different hyper parameters and select the best model. Follow the complete life cycle of data science. Include all the below steps mentioned:

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 the form of CSV (Comma Separated Value) format and consists of 6 columns (5 features and 1 label) with 10000 numbers of records as explained 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

We can see our dataset includes a target label "Used Car Price" column and the remaining feature columns can be used to determine or help in predicting the price of the used cars. Since price is a continuous value it makes this to be a Regression problem!

Importing all the necessary libraries/dependencies here.

In [4]:
```
df = pd.read_csv("Used_Car_Data.csv")
```

I am importing the collected dataset comma separated values file and storing it into our dataframe for further usage.

In [5]:
```
df # checking the first 5 and last 5 rows
```

Out[5]:

	Used Car Model	Year of Manufacture	Kilometers Driven	Fuel Type	Transmission Type	Used Car Price
0	Hyundai	2017	2,200 km	Petrol	Manual	5,25,000
1	Hyundai	2013	91,500 km	Diesel	Manual	5,95,000
2	Ford	2017	36,000 km	Diesel	Manual	7,75,000
3	Honda	2015	90,000 km	Diesel	Manual	4,00,000
4	Maruti Suzuki	2010	40,000 km	Petrol	Manual	2,30,000
...	...	...	...	...	...	...
9995	Hyundai	2012	65,000 km	Petrol	Manual	3,25,000
9996	Maruti Suzuki	2018	85,000 km	CNG & Hybrids	Manual	2,90,000
9997	Maruti Suzuki	2010	72,000 km	Petrol	Manual	3,20,000
9998	Tata	2012	70,000 km	Diesel	Manual	1,85,000
9999	Ford	2018	53,764 km	Diesel	Manual	8,75,000

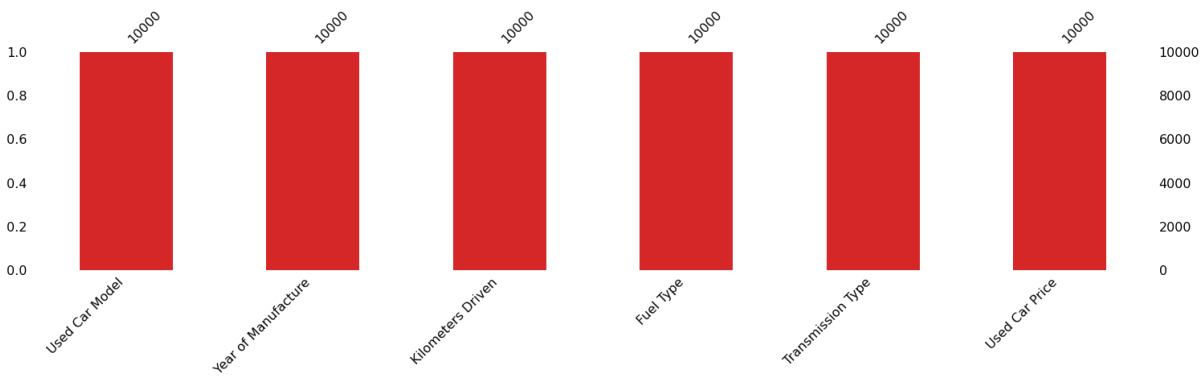
10000 rows × 6 columns

- Data Pre-processing 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.

```
df.isna().sum() # checking for missing values
```

```
Used Car Model          0
Year of Manufacture     0
Kilometers Driven       0
Fuel Type               0
Transmission Type       0
Used Car 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.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   Used Car Model       10000 non-null  object
 1   Year of Manufacture  10000 non-null  object
 2   Kilometers Driven    10000 non-null  object
 3   Fuel Type            10000 non-null  object
 4   Transmission Type    10000 non-null  object
 5   Used Car Price       10000 non-null  object
dtypes: object(6)
memory usage: 468.9+ KB
```

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

```
df.nunique().sort_values().to_frame("Unique Values")
```

	Unique Values
Transmission Type	3
Fuel Type	6
Year of Manufacture	31
Used Car Price	940
Kilometers Driven	1094
Used Car Model	2056

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

```
# Data pre processing

df["Kilometers Driven"]=df["Kilometers Driven"].apply(lambda x: x.replace(',','') if x!='-' else '-')
df["Kilometers Driven"]=df["Kilometers Driven"].apply(lambda x: int(x.split(' ')[0]) if x!='-' else 0)
df
```

```
df["Year of Manufacture"]=df["Year of Manufacture"].apply(lambda x: int(x.strip()[0:4]) if x!='-' else 0)
median_val_year=df["Year of Manufacture"].median()
df["Year of Manufacture"]=df["Year of Manufacture"].apply(lambda x: x if x!=0 else median_val_year)
df["Year of Manufacture"]=df["Year of Manufacture"].astype(int)
df
```

```
try:
    df["Used Car Price"]=df["Used Car Price"].apply(lambda x: x.split(' ')[1] if x!='-' else '0,0')
except IndexError:
    pass

try:
    df["Used Car Price"]=df["Used Car Price"].apply(lambda x: str(x.replace(',', '')))
except ValueError:
    pass

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

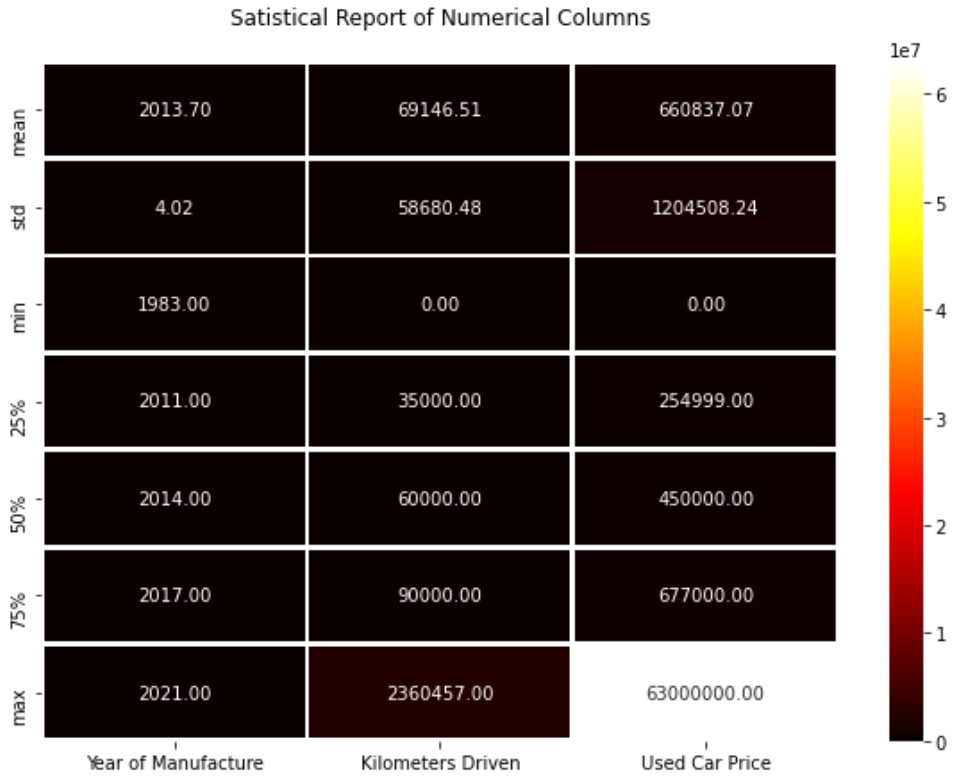
```
df["Fuel Type"]=df["Fuel Type"].apply(lambda x: x if x!='-' else 'Petrol') # replacing with common fuel type in india
df["Transmission Type"]=df["Transmission Type"].apply(lambda x: x if x!='-' else 'Manual') # common transmission is manual
df["Used Car Model"]=df["Used Car Model"].apply(lambda x: x if x!='-' else 'Hyundai') # common used car model
df["Kilometers Driven"]=df["Kilometers Driven"].apply(lambda x: x if x!='-' else 'None')
avg_usedcar_price=df["Used Car Price"].mean()
df["Used Car Price"]=df["Used Car 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.

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

	Used Car Model	Year of Manufacture	Kilometers Driven	Fuel Type	Transmission Type	Used Car Price
count	10000	10000.00000	1.000000e+04	10000	10000	1.000000e+04
unique	2055	NaN	NaN	5	2	NaN
top	Maruti Suzuki	NaN	NaN	Diesel	Manual	NaN
freq	602	NaN	NaN	5345	8598	NaN
mean	NaN	2013.69860	6.914651e+04	NaN	NaN	6.608371e+05
std	NaN	4.02124	5.868048e+04	NaN	NaN	1.204508e+06
min	NaN	1983.00000	0.000000e+00	NaN	NaN	0.000000e+00
25%	NaN	2011.00000	3.500000e+04	NaN	NaN	2.549990e+05
50%	NaN	2014.00000	6.000000e+04	NaN	NaN	4.500000e+05
75%	NaN	2017.00000	9.000000e+04	NaN	NaN	6.770000e+05
max	NaN	2021.00000	2.360457e+06	NaN	NaN	6.300000e+07

I also took a look at only the numeric component and observed only the maximum number for the Used Car Price column on a larger 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=["Transmission Type", "Fuel Type", "Used Car Model"]
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:  (10000, 6)
Shape of the dataframe after removing outliers:  (9660, 6)
Percentage of data loss post outlier removal:  3.4000000000000004
```

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])
```

Describe the set of assumptions (if any) that you have about the problem you're working on.

Here you can write about any assumptions you've made.

- Hardware and Software Requirements and Tools Used
    - Processor Intel(R) Core(TM) i3-6100T CPU @ 3.20GHz 3.19 GHz
    - Installed RAM 8.00 GB
    - System type 64-bit operating system, x64-based processor
    - 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. Remove any scraped information from the dataset.
  2. Use meaningful information to fill in missing values.
  3. Generating numerical input data from categorical data.
  4. Evaluate many models and select the most appropriate one.
  5. The major evaluation metric is the R2 score.
  6. Secondary metrics MSE and RMSE are used.
  7. The Cross Validation Score was utilised to guarantee that our underfitting models were not overfit.

- Testing of Identified Approaches (Algorithms)

  Machine Learning, Libraries, and Here are the regression models that were employed in this study.

```
import warnings
warnings.simplefilter("ignore")
warnings.filterwarnings("ignore")
import joblib

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

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
```

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

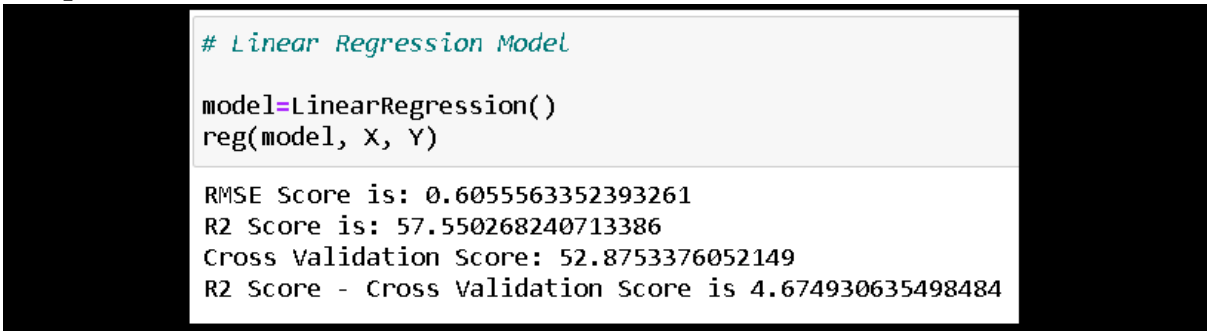
  I constructed a Regression Machine Leaning Model function that includes the evaluation metrics so that we can acquire the data we need for all of the models listed above.

  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:



- Key Metrics for success in solving problem under consideration

RMSE Score:
Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit.

R2 Score:
The R2 score is a very important metric that is used to evaluate the performance of a regression-based machine learning model. It is pronounced as R squared and is also known as the coefficient of determination. It works by measuring the amount of variance in the predictions explained by the dataset.

Cross Validation Score:
Cross-validation is a statistical method used to estimate the skill of machine learning models. It is commonly used in applied machine learning to compare and select a model for a given predictive modelling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods. The k-fold cross validation is a procedure used to estimate the skill of the model on new data. There are common tactics that you can use to select the value of k for your dataset (I have used 5-fold validation in this project). There are

commonly used variations on cross-validation such as stratified and repeated that are available in scikit-learn.

Hyper Parameter Tuning:
In machine learning, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are 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 part one by one and comparing and researching the effects of all the available feature columns on the prediction of our target label.

  Code:

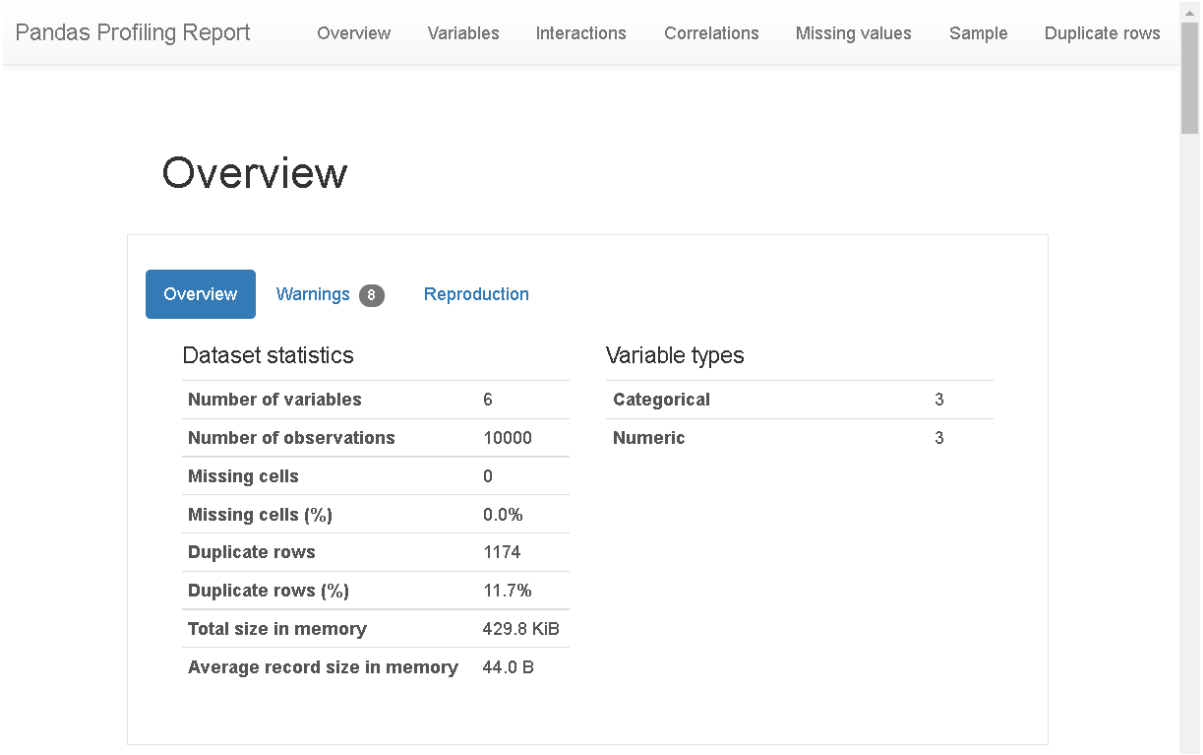
```
pandas_profiling.ProfileReport(df)
```

Summarize dataset: 100% 19/19 [00:08<00:00, 1.59it/s, Completed]

Generate report structure: 100% 1/1 [00:03<00:00, 3.51s/it]

Render HTML: 100% 1/1 [00:00<00:00, 1.12it/s]

Output:



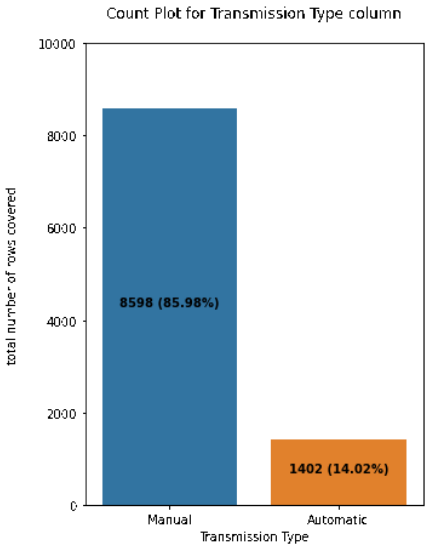
pandas-profiling is a free Python module that allows us to perform exploratory data analysis with just a few lines of code. It creates interactive web reports that may be delivered to anyone, even if they have no programming experience. It also provides report generation for the dataset, with a variety 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 = 'Transmission Type'
    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
```

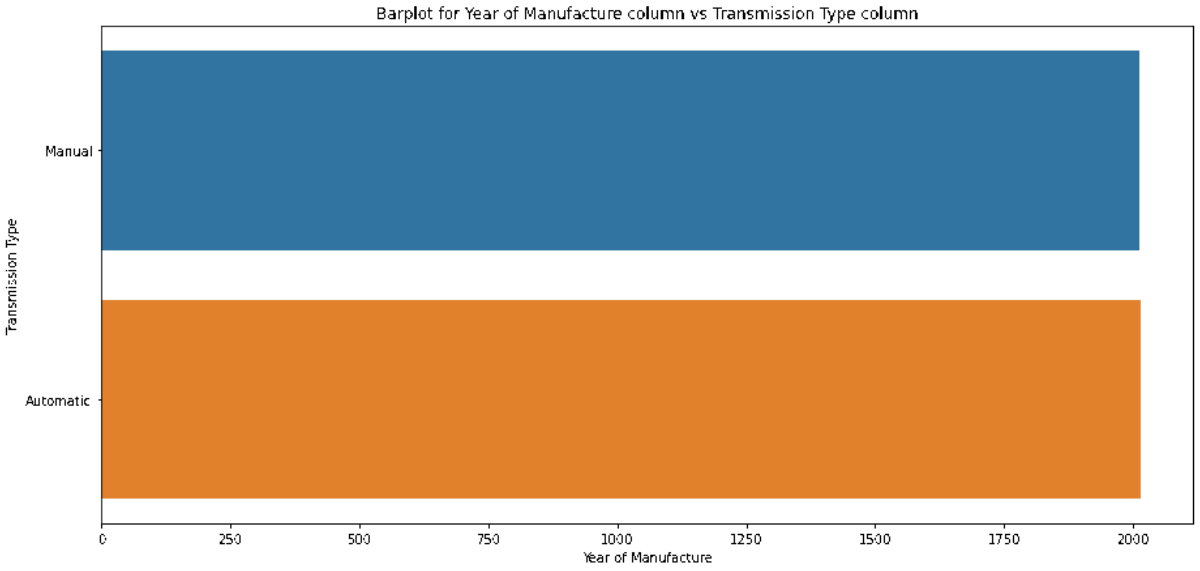
Output:



Code:

```
y = 'Transmission Type'

x = 'Year of Manufacture'
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 = 'Kilometers Driven'
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 = 'Used Car 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()
```

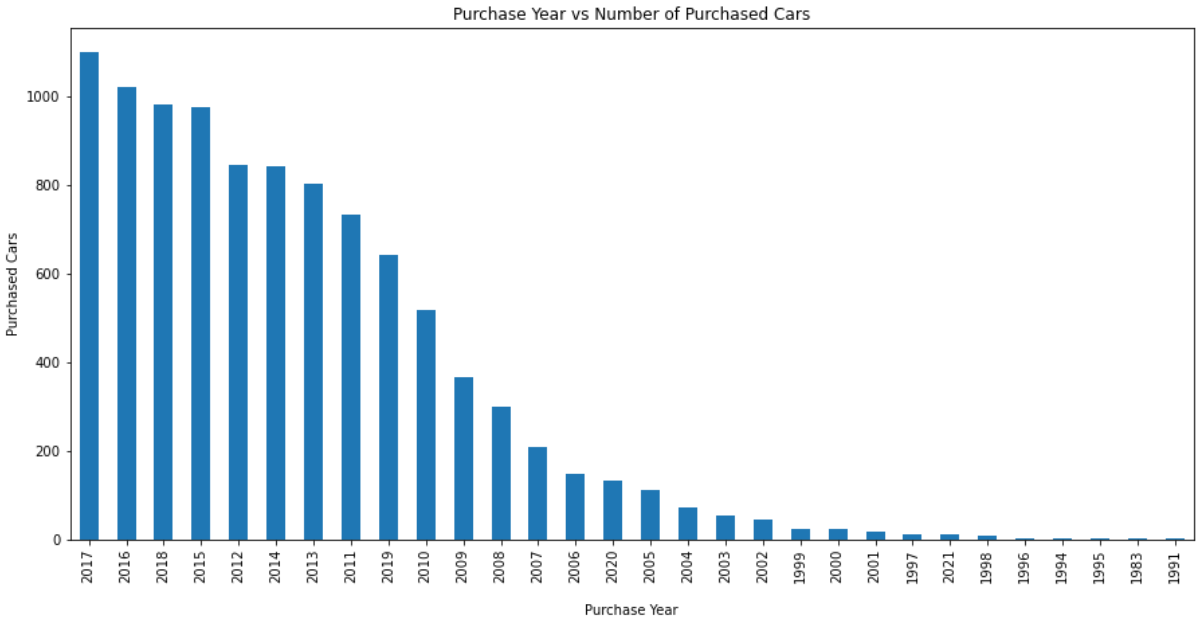
Output:



Code:

```
plt.figure(figsize=[15,7])
purchased_car_per_year = df['Year of Manufacture'].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:

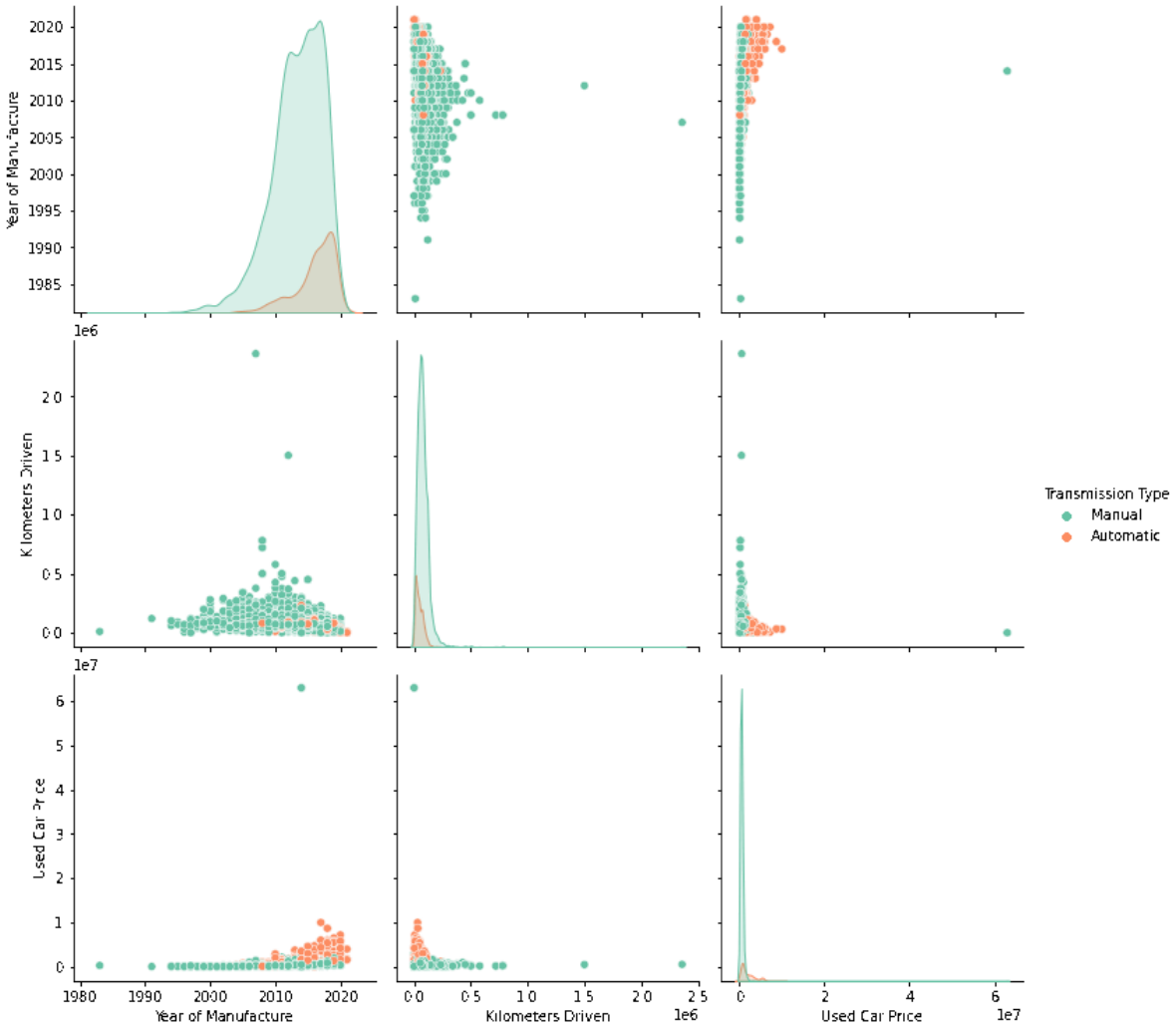


Code:

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

```
Manual = df[df['Transmission Type']=='Manual']
Automatic = df[df['Transmission Type']=='Automatic']

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

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

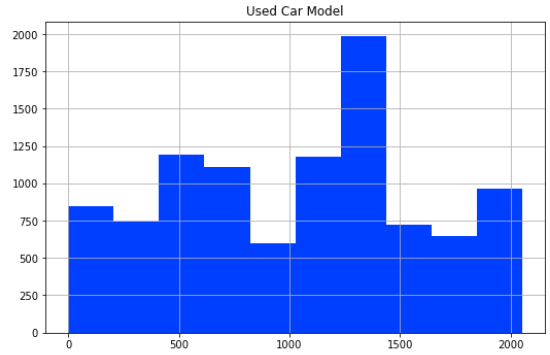
Output:

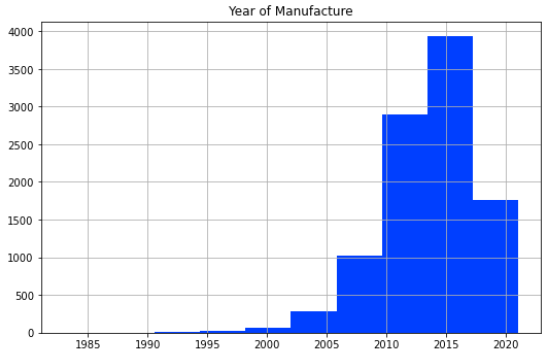


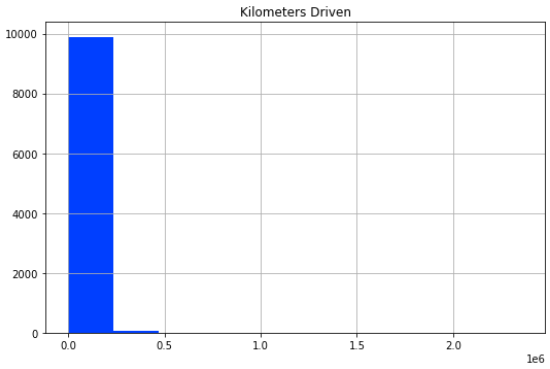
Code:

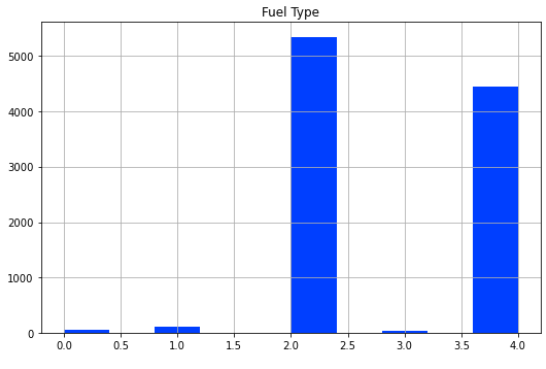
```
plt.style.use('seaborn-bright')

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

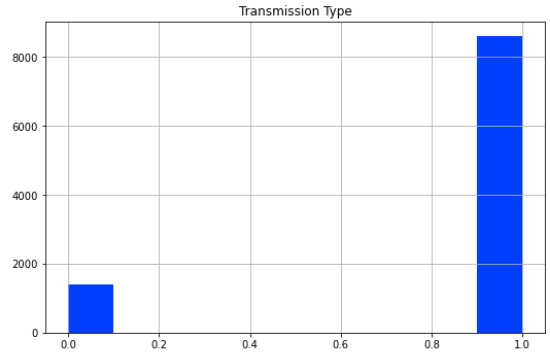
Output:

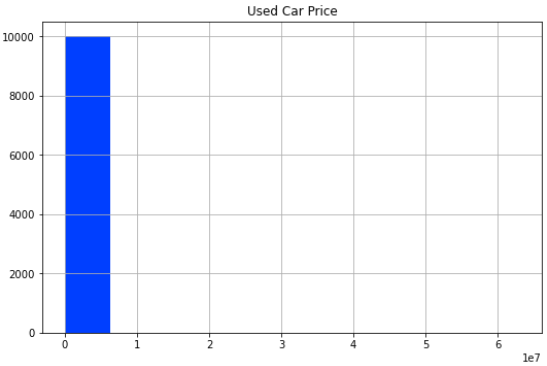








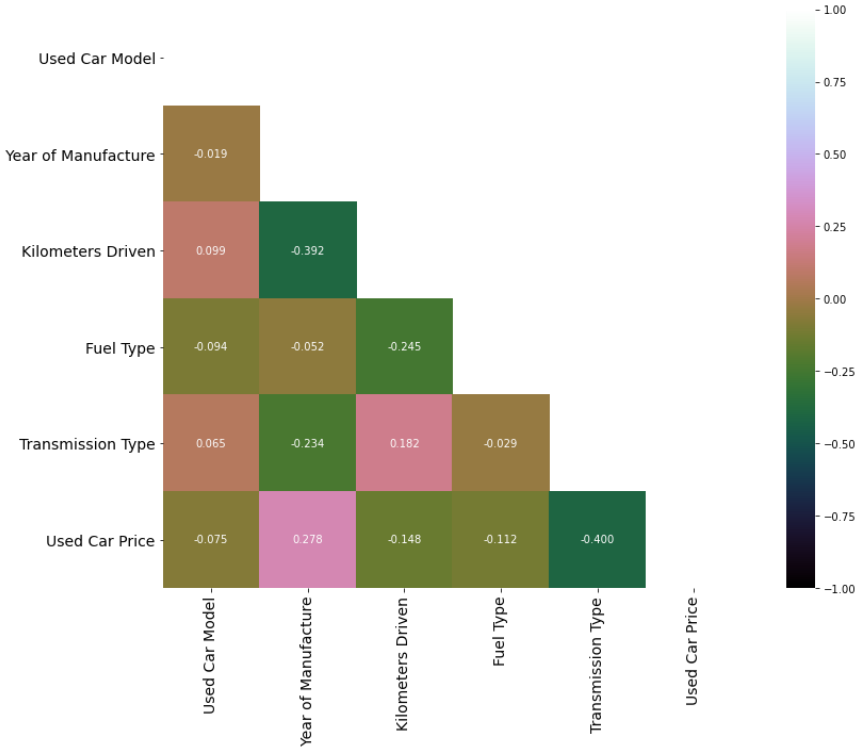




Code:

```
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="cubehelix", mask=upper_triangle)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```

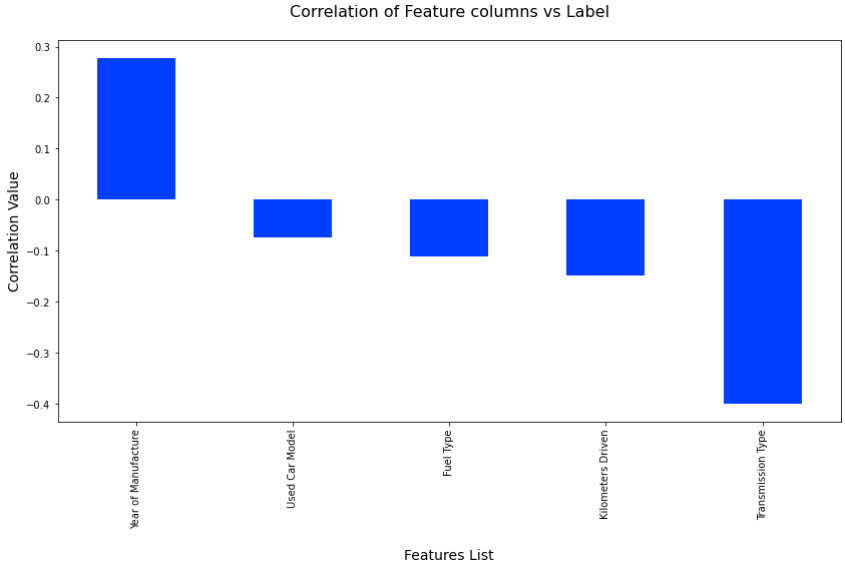
Output:



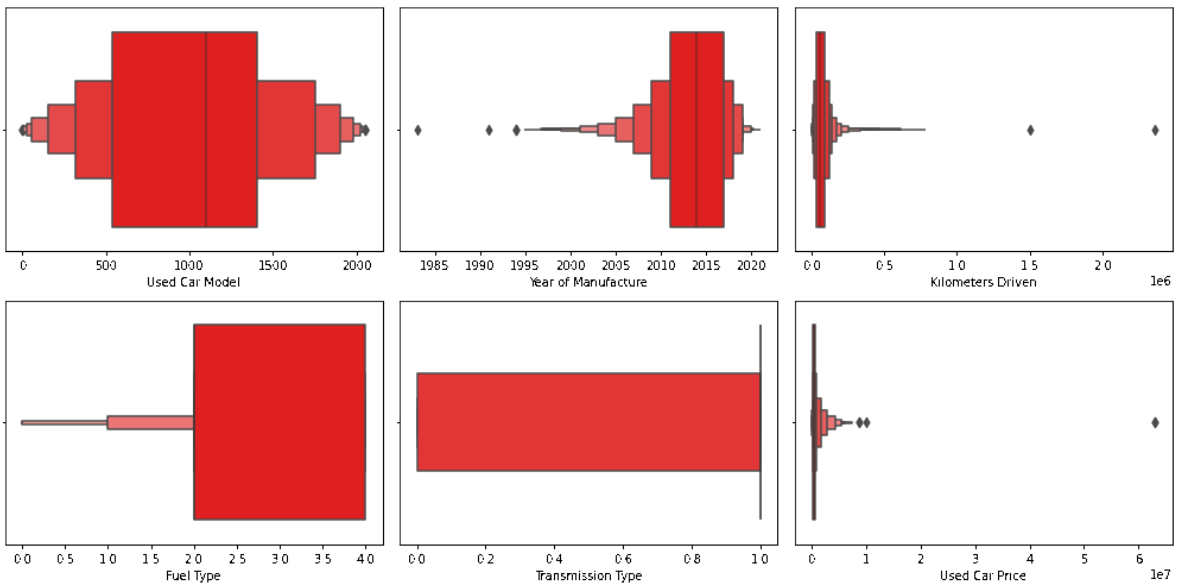
Code:

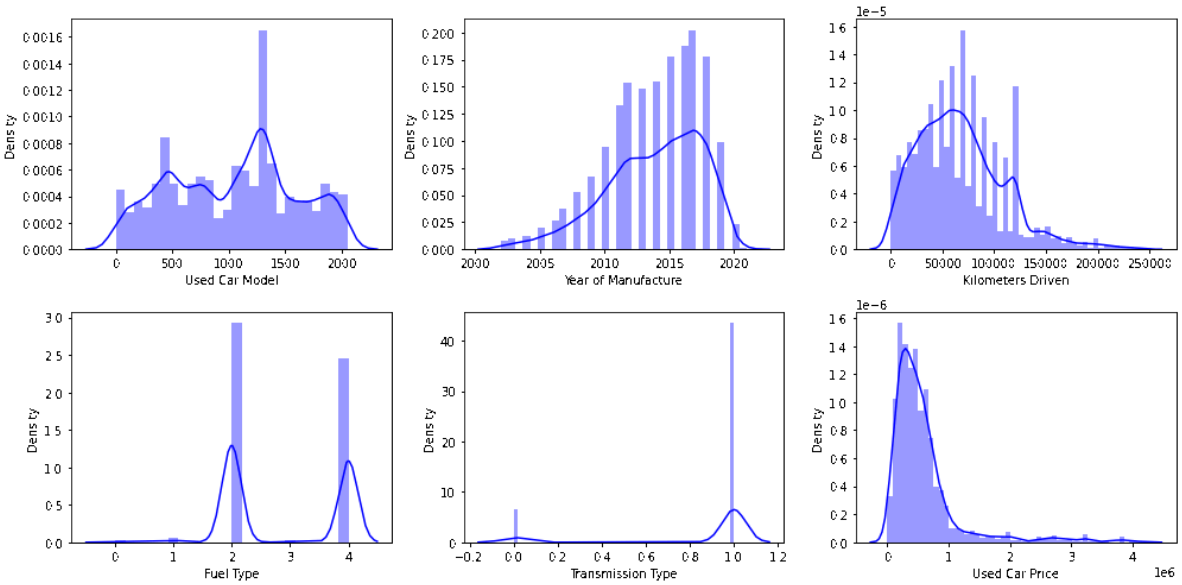
```
df_corr = df.corr()
plt.figure(figsize=(14,7))
df_corr['Used Car Price'].sort_values(ascending=False).drop('Used Car Price').plot.bar()
plt.title("Correlation of Feature columns vs Label\n", fontsize=16)
plt.xlabel("\nFeatures List", fontsize=14)
plt.ylabel("Correlation Value", fontsize=14)
plt.show()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

Output:



Outliers and Skewness before and after treating them:

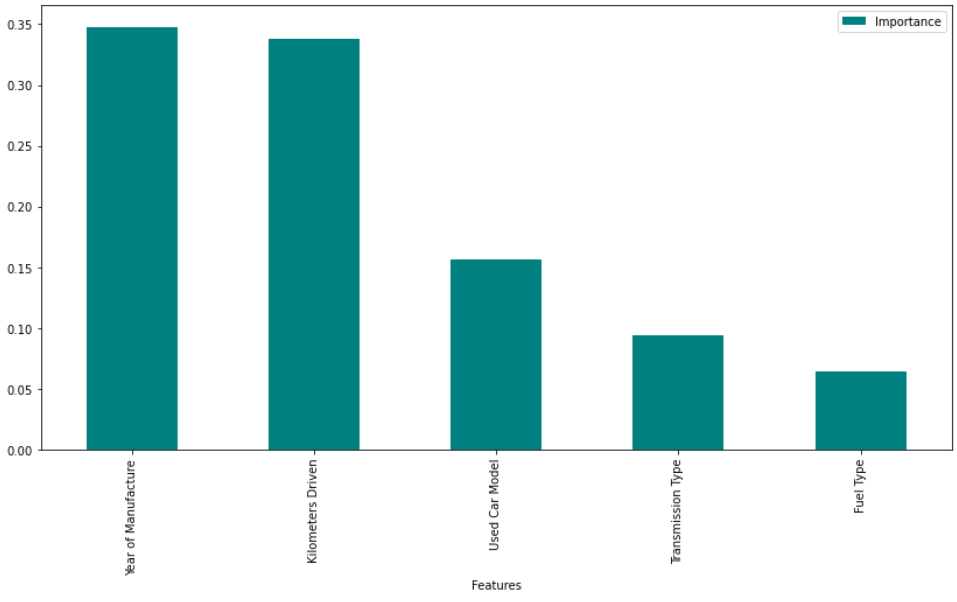




Code:

```
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='teal')
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.