FLIP ROBO

CAR PRICE PREDICTION PROJECT

Submitted by:

Shikha Chaudhary

ACKNOWLEDGMENT

I'd like to express my heartfelt gratitude to Shubham Yadav, my SME (Subject Matter Expert), as well as Flip Robo Technologies, for allowing me to work on this project on 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 other resources to assist me finish the job. 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 automobile industry: Before the Covid-19 crisis, the Indian automobile sector was already in trouble. It had an almost 18% decline in total growth. The commencement of the Covid-19 outbreaks, as well as continuous lockdowns across India and the rest of the world, aggravated the issue. Slow economic growth, poor consumer mood, BS-VI transitions, changes to axle load regulations, liquidity constraint, low-capacity utilisation, and probable bankruptcies are all posing challenges for the Indian automotive business in the coming two years (FY20 and FY21). The restoration to near-normalcy of everyday life and manufacturing activity in China and South Korea, as well as India's protracted shutdown, provide promise for a U-shaped economic revival. According to our prediction, 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 would suffer when India's GDP growth rate for FY21 is reduced from 5% to 0% and then to (-5%). Job creation and income levels are closely correlated with auto demand, and both have been impacted. The revenue effect on India's car sector, 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 vehicle 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 highlighted flaws in the car 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 car industry.

- ## Conceptual Background of the Domain Problem

The expanding world of e-commerce includes everything you'd expect to find at a general shop, not just gadgets and apparel. Putting away the common retail viewpoint and looking at the larger picture, the digital marketplace sees hundreds, if not millions, of transactions every day. The vehicle sector, which involves the purchasing and selling of used automobiles, is one of the most expanding markets in the digital arena. 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 automobile 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 examined before knowing the price. So, rather than performing all of these things, we may create a machine learning model that uses various attributes of used automobiles to forecast the accurate and valuable car price.

- ## Review of Literature

This project focuses on the data's exploration, feature engineering, and classification capabilities. We can conduct better data exploration and extract some interesting characteristics utilising the provided columns since we scrape a large quantity 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 characteristics. 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 separate feature helped me grasp the data based on the issue description and real-time data scraped from the OLX and Cars24 websites, as each feature delivers a distinct sort of information. Working with many forms of real-time data in a single data set and doing root cause analysis to anticipate the price of a used automobile 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 utilised by the customer to understand how the costs change with the variables. They might work on it properly and devise some ways to sell the used automobile for a profit. Furthermore, the model will assist the client in comprehending the price dynamics of a used vehicle.

Analytical Problem Framing

- **Mathematical/ Analytical Modeling of the Problem**

  In our scrapped dataset, our target variable "Used Car Price " is a continuous variable. Therefore, we will be handling this modelling problem as regression.

  This project is done in two parts:

  - Data Collection phase
  - Model Building phase

  Data Collection phase:

  You have to scrape at least 5000 used cars data. You can scrape more data as well, it’s up to you. More the data better the model. In

this section You need to scrape the data of used cars from websites (OLX, OLA, Car Dekho, Cars24 etc.) You need web scraping for this. You have to fetch data for different locations. The number of columns for data doesn't have limit, it's up to you and your creativity. Generally, these columns are Brand, model, variant, manufacturing year, driven kilometres, fuel, number of owners, location and at last target variable Price of the car. This data is to give you a hint about important variables in used car model. You can make changes to it, you can add or you can remove some columns, it completely depends on the website from which you are fetching the data. Try to include all types of cars in your data for example- SUV, Sedans, Coupe, minivan, Hatchback.

Model Building phase:

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 number 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!

```
df = pd.read_csv(r"C:\Users\swrai\Desktop\Used_Car_Data.csv")
```

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

```
df # checking the first 5 and last 5 rows
```

	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

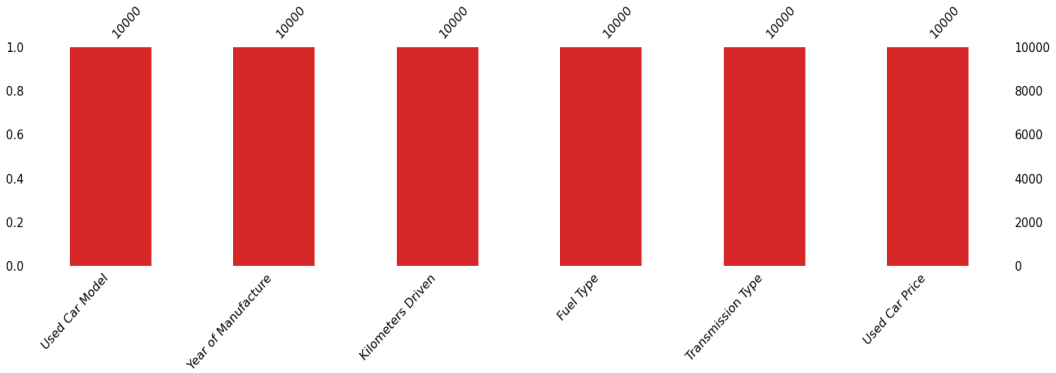
10000 rows × 6 columns

- # Data Preprocessing Done

For the data pre-processing step, I checked through the dataframe for missing values, imputed records with “- “using various imputing techniques to handle them.

```
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
```



Checked the datatype details for each column to understand the numeric ones and its further conversion process.

```
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 took a look at all the unique value present in each of the columns and then decided to dealt with the imputation part accordingly.

```
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 various data imputation performed on our data set are shown below with the code.

```
# 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
```

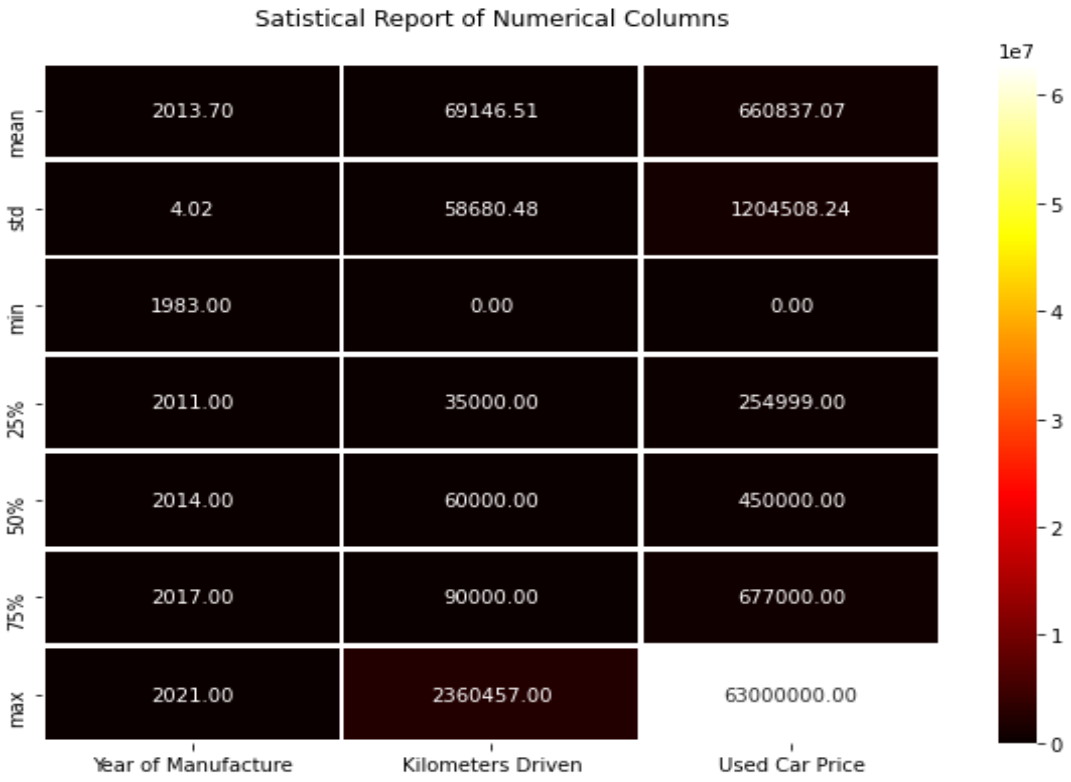
```
df["Fuel Type"]=df["Fuel Type"].apply(lambda x: x if x!='-' else 'Petrol') # replacing with common fuel type in india
df["Transmission Type"]=df["Transmission Type"].apply(lambda x: x if x!='-' else 'Manual') # common transmission is manual
df["Used Car Model"]=df["Used Car Model"].apply(lambda x: 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
```

I then used the “describe” method to check the count, mean, standard deviation, minimum, maximum, 25%, 50% and 75% quartile data.

```
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

Took a visual on just the numeric part as well and saw just the maximum value for Used Car Price column at a higher scale.



- **Data Inputs- Logic- Output Relationships**

The input data were initially all object datatype so had to clean the data by removing unwanted information like “km” from Kilometres Driven column and ensuring the numeric data are converted accordingly. I then used Ordinal Encoding method to convert all the categorical feature columns to numeric format.

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
```

To handle the skewness, I made use of Log transformation technique ensuring that at 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.

RAM : 8 GB

CPU : Intel(R) Core (TM) i3-7100U CPU @ 2.40GHz 2.40 GHz

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.

```
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**

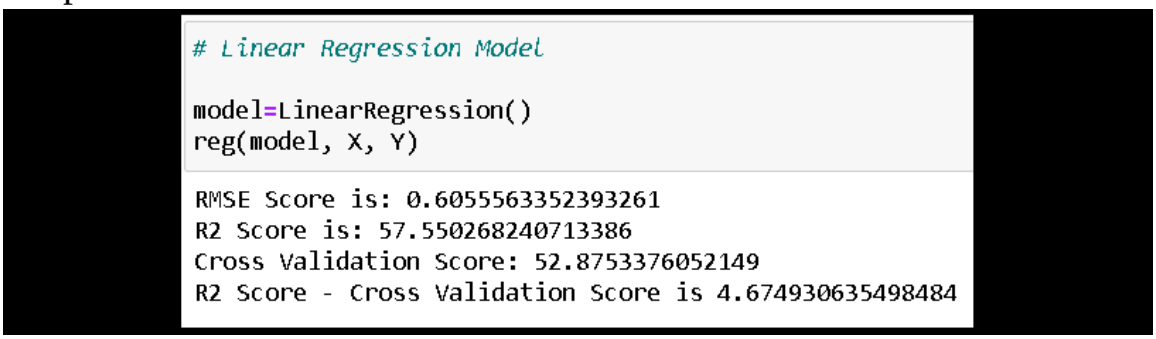
  I created a Regression Machine Leaning Model function incorporating the evaluation metrics so that we can get the required data for all the above models.

  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
```

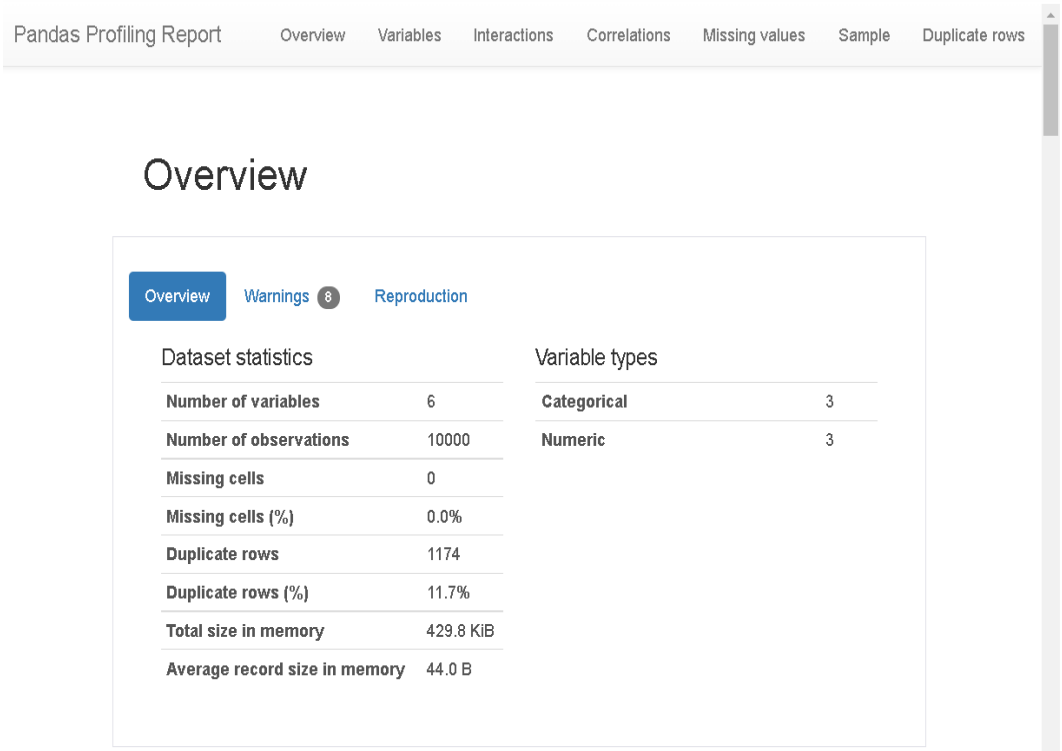
- **Visualizations**

I used the pandas profiling feature to generate an initial detailed report on my dataframe values. It gives us various information on the rendered dataset like the correlations, missing values, duplicate rows, variable types, memory size etc. This assists us in further detailed visualization separating each part one by one comparing and research for the impacts on the prediction of our target label from all the available feature columns.

Code:

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

Output:



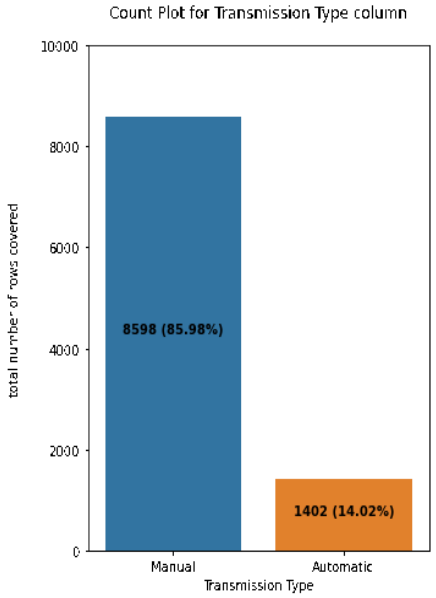
pandas-profiling is an open-source Python module with which we can quickly do an exploratory data analysis with just a few lines of code. It generates interactive reports in web format that can be presented to any person, even if they don't know programming. It also offers report generation for the dataset with lots of features and customizations for the report generated. In short, what pandas-profiling does is save us all the work of visualizing and understanding the distribution of each variable. It generates a report with all the information easily available.

I generated count plots, bar plots, pair plots, heatmap and others to visualise the datapoint present in our column records.

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
```

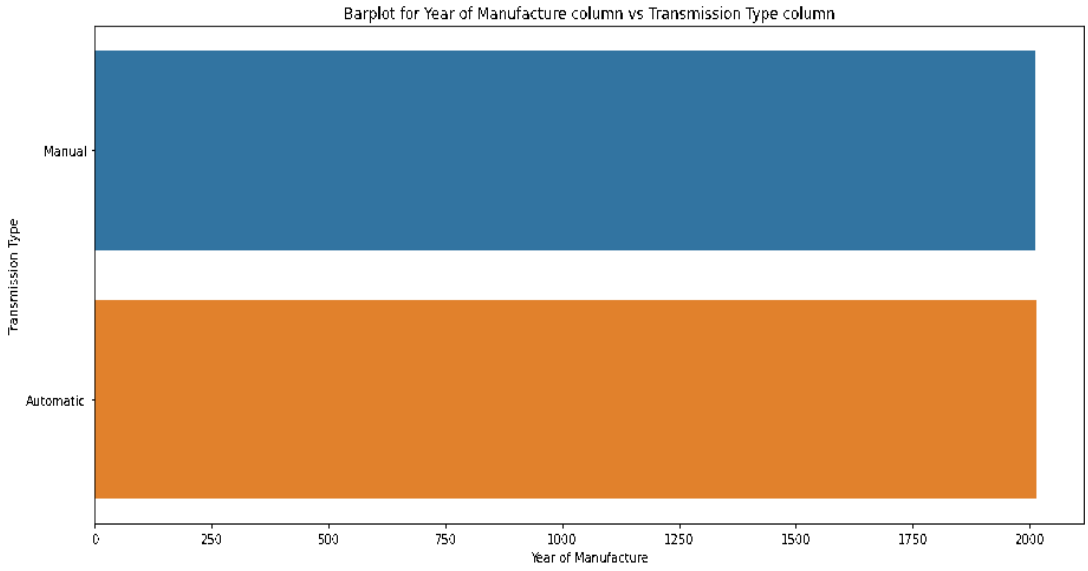
Output:



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()
```

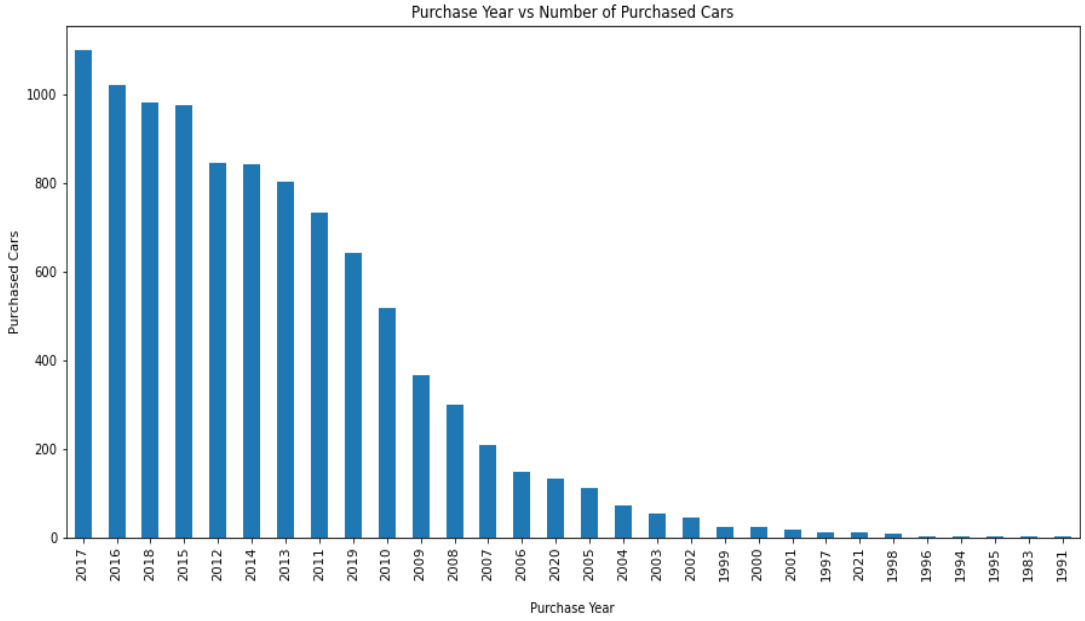
Output:



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:

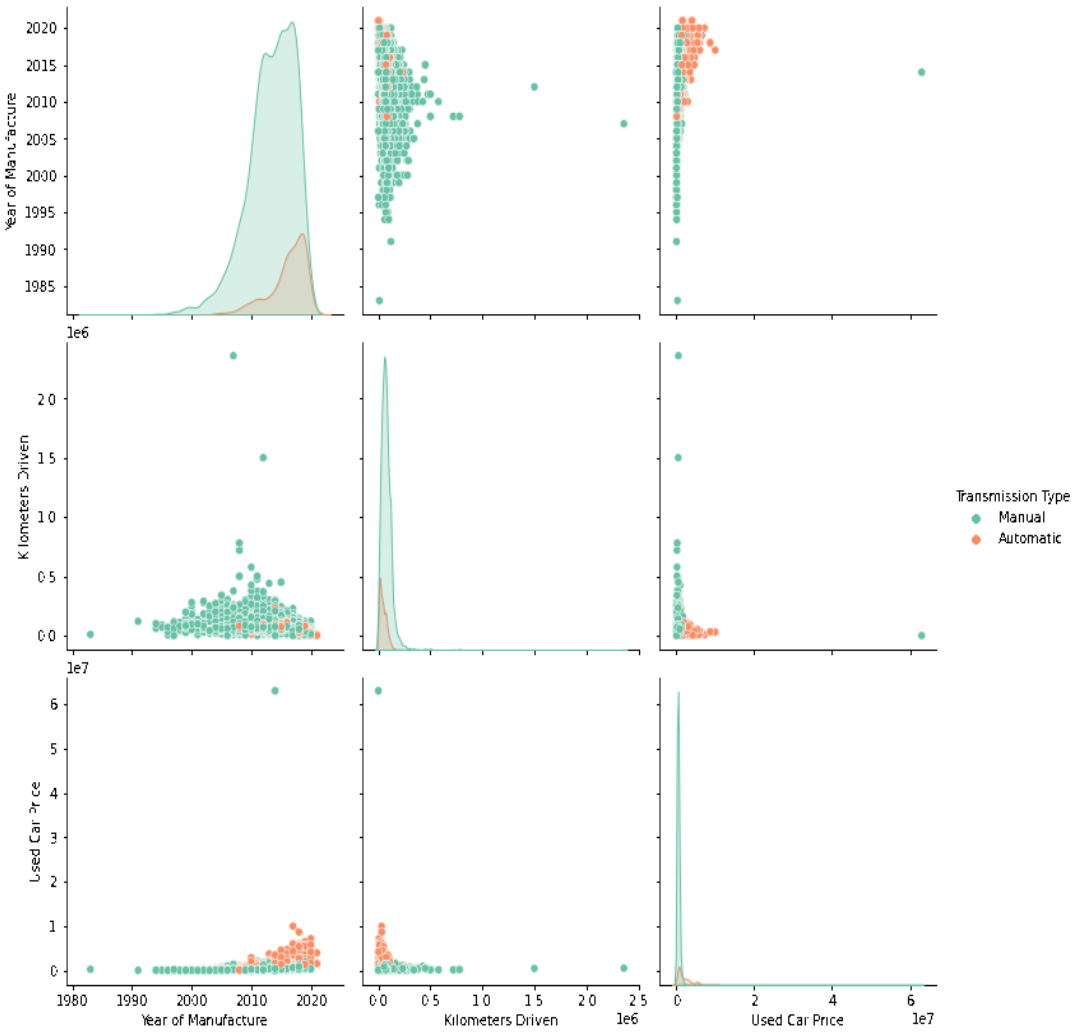


Code:

```
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()
```

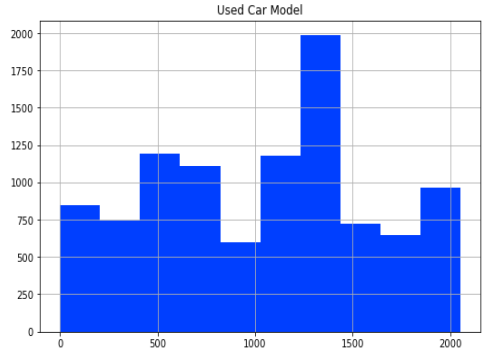
Output:

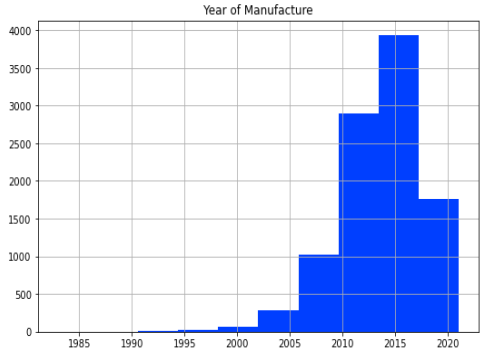


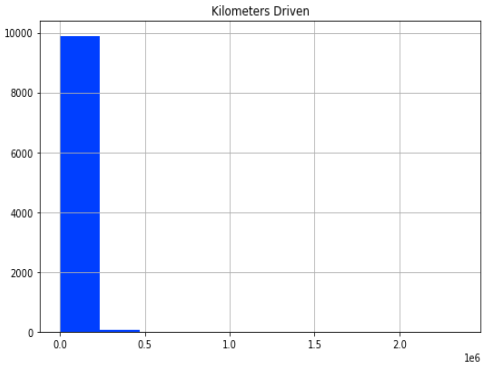
Code:

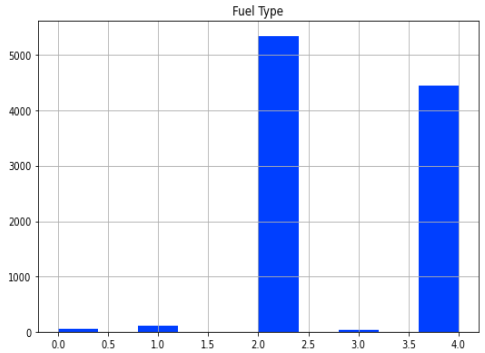
```
plt.style.use('seaborn-bright')

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

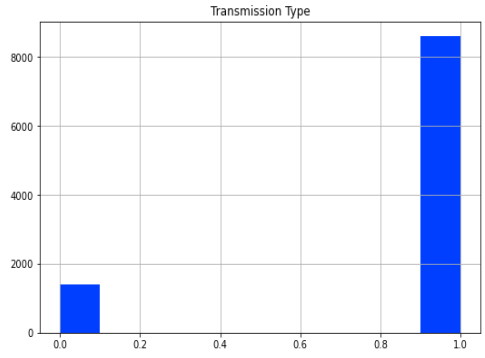
Output:

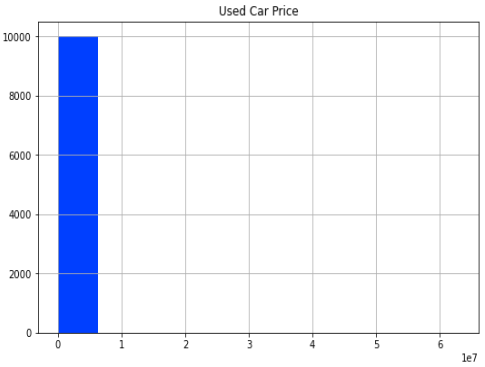








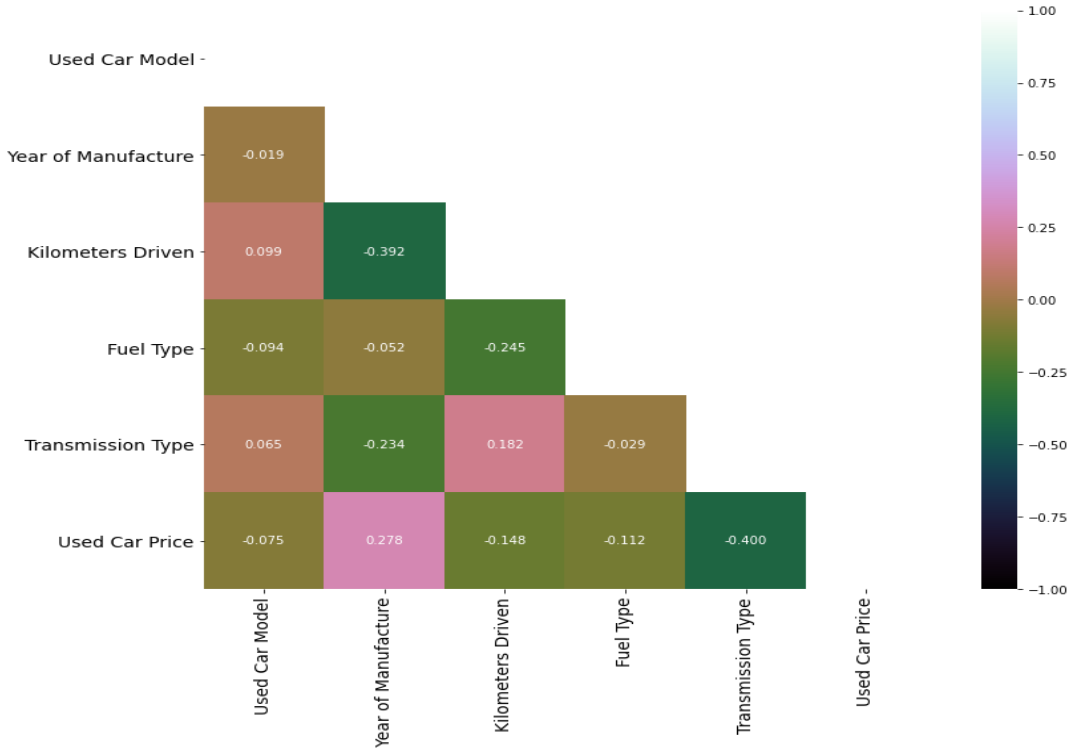




Code:

```
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()
```

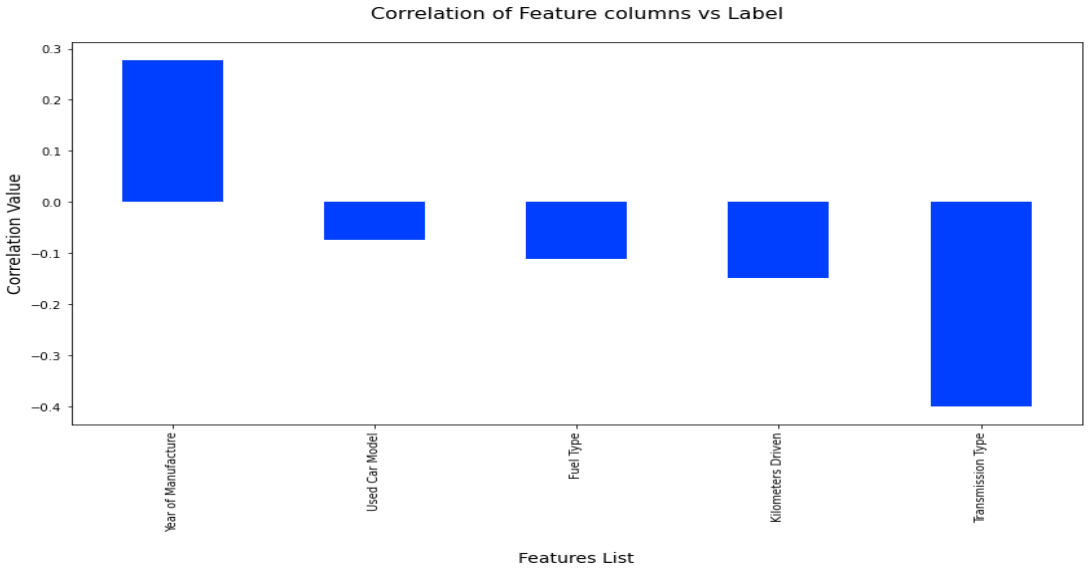
Output:



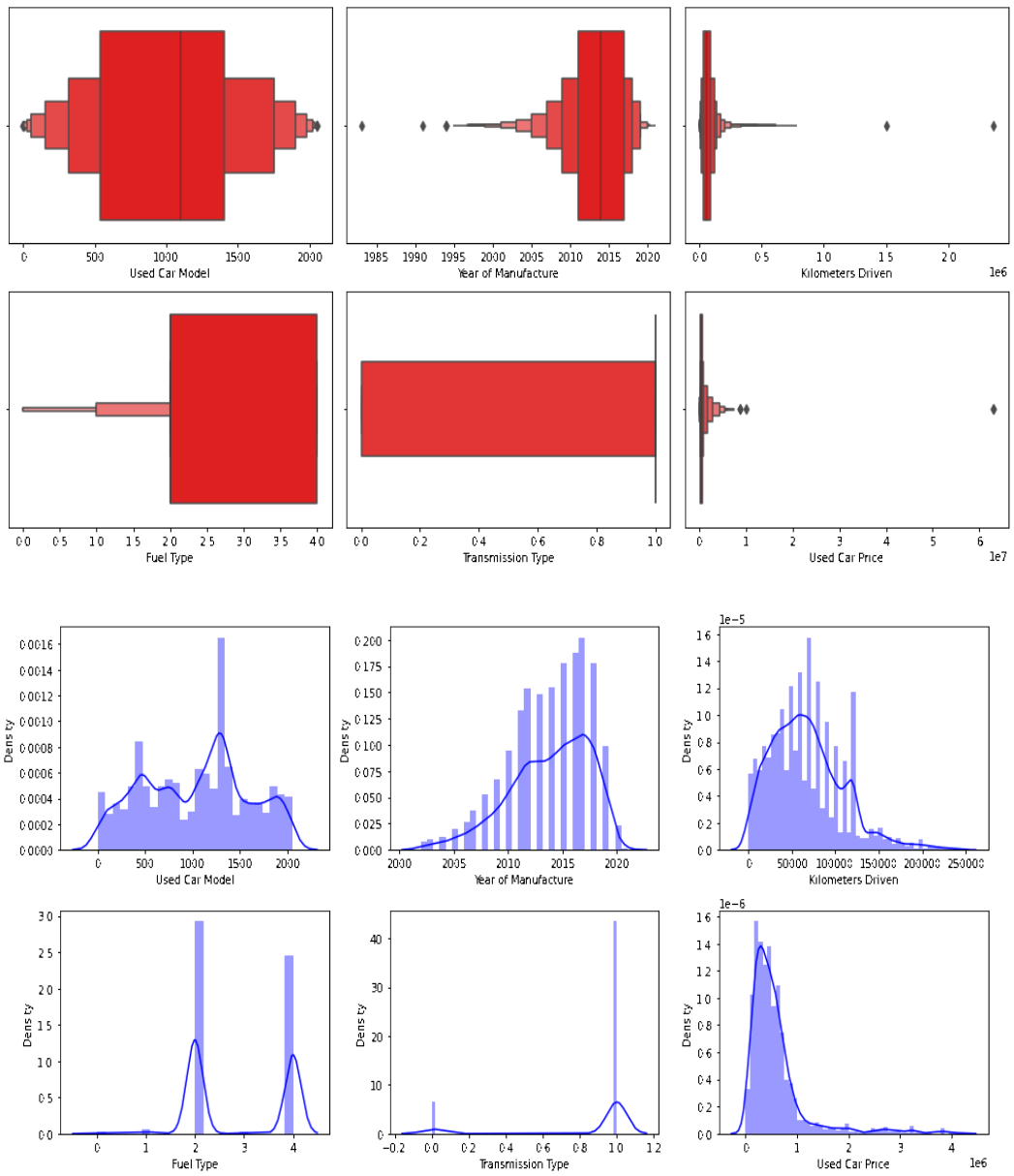
Code:

```
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:



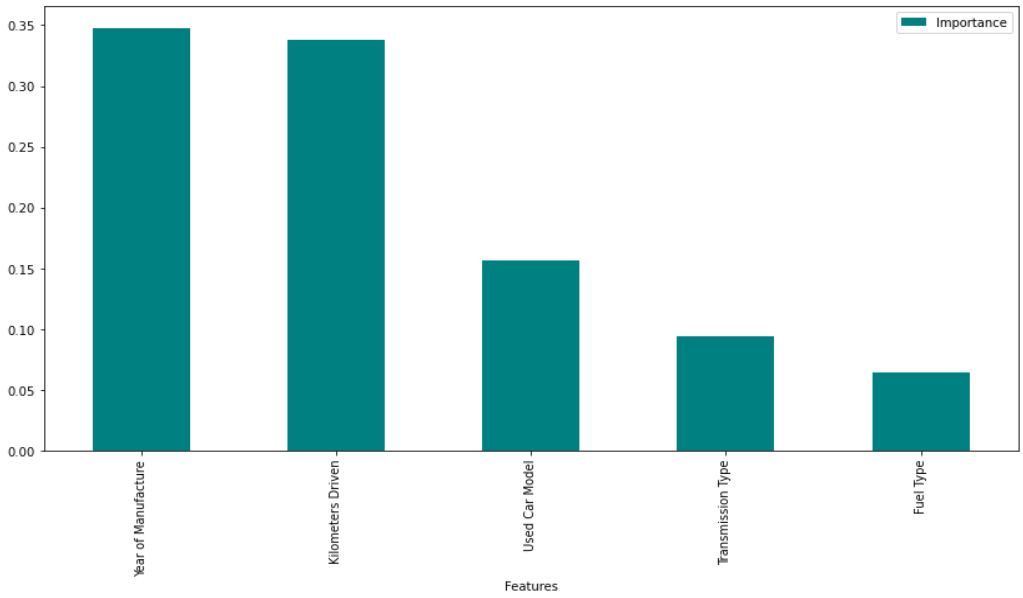
Outliers and Skewness before and after treating them:



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**

  We can see from the visuals that the features are impacting the price of used cars. There were categorical columns which I encoded using the ordinal encoder method instead of the one hot encoding to avoid the generation of large number of columns. Also, I our target label stored continuous numeric data and therefore label encoder was out of the picture to be used.

CONCLUSION

- ## Key Findings and Conclusions of the Study

  After the completion of this project, we got an insight on how to collect data, pre-processing the data, analysing the data and building a model. First, we collected the used cars data from different websites like OLX, Car Dekho, Cars 24, OLA etc and it was done by using Web Scraping. The framework used for web scraping was Beautiful Soup and Selenium, which has an advantage of automating our process of collecting data. We collected almost 10000 of data which contained the selling price and other related features of used cars. Then the scrapped data was combined in a single data frame and saved in a csv file so that we can open it and analyse the data. We did data cleaning, data pre-processing steps like finding and handling null values, removing words from numbers, converting object to int type, data visualization, handling outliers and skewness etc. After separating our train and test data, we started running different machine learning regression algorithms to find out the best performing model. We found that Extra Tree Regressor Algorithm was performing well according to their r2_score and cross validation scores. Then we performed Hyperparameter Tuning technique using Grid Search CV for getting the best parameters and improving the score. In that Extra Tree Regressor Algorithm did not perform quite well as previously on the defaults but we finalised that model for further predictions as it was still better than the rest. We saved the final model in pkl format using the joblib library after getting a dataframe of predicted and actual used car price details.

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

  Visualization part helped me to understand the data as it provides graphical representation of huge data. It assisted me to understand the feature importance, outliers/skewness detection and to compare

the independent-dependent features. Data cleaning is the most important part of model building and therefore before model building, I made sure the data is cleaned and scaled. I have generated multiple regression machine learning models to get the best model wherein I found Extra Trees Regressor Model being the best based on the metrics I have used.

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

  The limitations we faced during this project were:

  The website was poorly designed because the scrapping took a lot of time and there were many issues in accessing to next page. Also need further practise in terms of various web scraping techniques. More negative correlated data were present than the positive correlated one's. Presence of outliers and skewness were detected and while dealing with them we had to lose a bit of valuable data. No information for handling these fast-paced websites were provided so that was consuming more time in web scraping part.

  Future Work Scope:

  Current model is limited to used car data but this can further be improved for other sectors of automobiles by training the model accordingly. The overall score can also be improved further by training the model with more specific data.