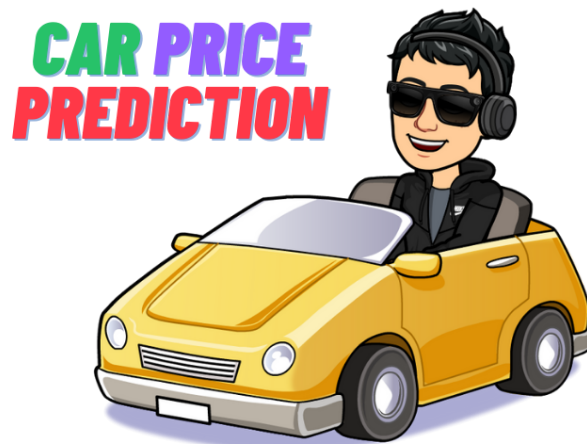**FLIP ROBO**

CAR PRICE PREDICTION PROJECT

CAR PRICE
PREDICTION

Submitted by: Anurag Krishnan

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my SME (Subject Matter Expert) Khushboo Garg as well as Flip Robo Technologies who gave me the opportunity to do this project on Car Price Prediction, which also helped me in doing lots of research wherein I came to know about so many new things especially the data collection part.

Also, I have utilized a few external resources that helped me to complete the project. I ensured that I learn from the samples and modify things according to my project requirement.

INTRODUCTION

- ## Business Problem Framing

   With the covid 19 impact in the market, we have seen a lot of changes in the car market. Now some cars are in demand hence making them costly and some are not in demand hence cheaper. One of our clients works with small traders, who sell used cars. With the change in market due to covid 19 impact, our client is facing problems with their previous car price valuation machine learning models. So, they are looking for new machine learning models from new data. We have to make a car price valuation model. This project contains two phase- OEMs, dealers and suppliers with strong cash reserves and better access to capital will be better positioned to sail through. Auto sector has been under pressure due to a mix of demand and supply factors. However, there are also some positive outcomes, which we shall look at.

   • With India's GDP growth rate for FY21 being downgraded from 5% to 0% and later to (-5%), the auto sector will take a hit. Auto

demand is highly sensitive to job creation and income levels and both have been impacted. CII has estimated the revenue impact at $2 billion on a monthly basis across the auto industry in India.

• Supply chain could be the worst affected. Even as China recovers, supply chain disruptions are likely to last for some more time. The problems on the Indo-China border at Ladakh are not helping matters. Domestic suppliers are chipping in but they will face an inventory surplus as demand remains tepid.

• The Unlock 1.0 will coincide with the implementation of the BS-VI norms and that would mean heavier discounts to dealers and also to customers. Even as auto companies are managing costs, the impact of discounts on profitability is going to be fairly steep.

• The real pain could be on the dealer end with most of them struggling with excess inventory and lack of funding options in the post COVID-19 scenario. The BS-VI price increases are also likely to hit auto demand. There are two positive developments emanating from COVID-19. The China supply chain shock is forcing major investments in the "Make in India" initiative. The COVID-19 crisis has exposed chinks in the automobile business model and it could catalyse a big move towards electric vehicles (EVs). That could be a big positive for the auto sector.
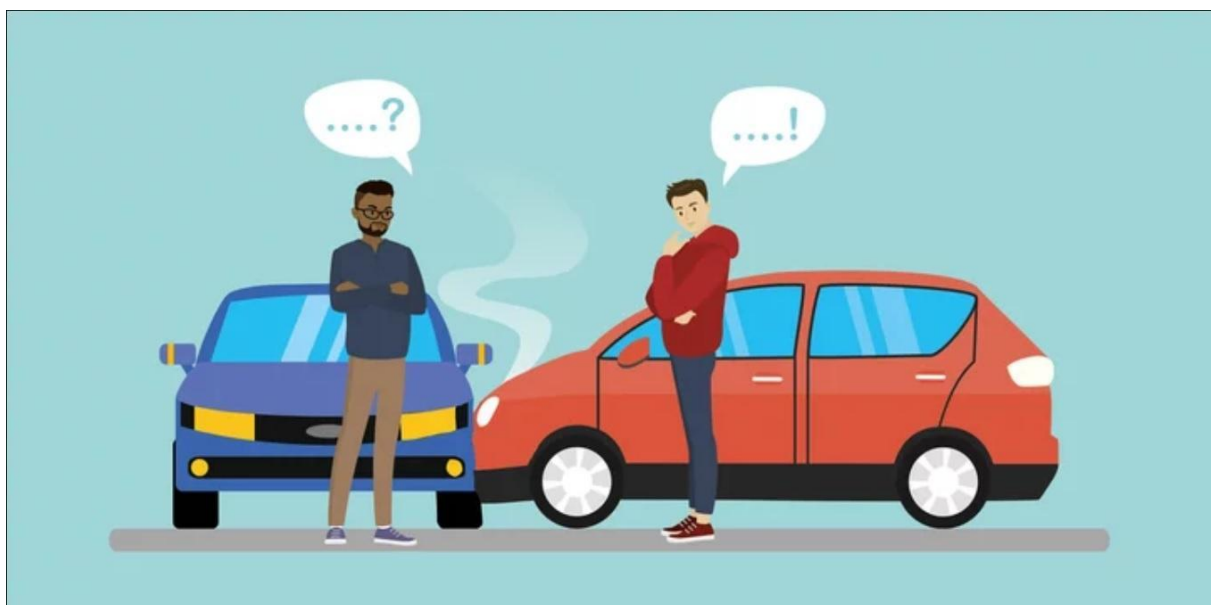
- ## Conceptual Background of the Domain Problem

The growing world of e-commerce is not just restricted to buying electronics and clothing but everything that you expect in a general store. Keeping the general store perspective aside and looking at the bigger picture, every day there are thousands or perhaps millions of deals happening in the digital marketplace. One of the most booming markets in the digital space is that of the automobile industry wherein the buying and selling of used cars take place. Sometimes we need to walk up to the dealer or individual sellers to get a used car price quote. However, buyers and sellers face a major stumbling block when it comes to their used car valuation or say their second-hand car valuation. Traditionally, you would go to a showroom and get your vehicle inspected before learning about the price. So instead of doing all this stuff we can build a machine learning model using different features of the used cars to predict the exact and valuable car price.

- ## Review of Literature

This project is more about exploration, feature engineering and classification that can be done on this data. Since we scrape huge amounts of data that includes more car related features, we can do better data exploration and derive some interesting features using the available columns.

The goal of this project is to build an application which can predict the car prices with the help of other features. In the long term, this would allow people to better explain and review their purchase with each other in this increasingly digital world.

- ## Motivation for the Problem Undertaken
Based on the problem statement and the real time data scraped from the CarDekho.com and Cars24 websites, I have understood how each independent feature helped me to understand the data as each feature provides a different kind of information. It is so interesting to work with different types of real time data in a single

data set and perform root cause analysis to predict the price of the used car. Based on the analysis of the model of the car, kilometres driven, transmission type, fuel type etc. I would be able to model the price of a used car as this model will then be used by the client to understand how exactly the prices vary with the variables. They can accordingly work on it and make some strategies to sell the used car and get some high returns. 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 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 the 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 a 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 models. 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:

- 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
- Car Price - Lists the selling price of the 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 a Regression problem!

```
1  df = pd.read_csv('file1.csv')
2
3  df.head()
```

	Unnamed: 0	Car Model	Year	Kilometers Driven	Fuel Type	Transmission Type	Price
0	0	Maruti Swift	2015	34588.0	Petrol	Manual	410499.0
1	1	Hyundai Grand i10	2014	43841.0	Petrol	Manual	354499.0
2	2	Maruti Ritz	2013	14047.0	Petrol	Manual	383999.0
3	3	Honda Accord	2012	50161.0	Petrol	Manual	520199.0
4	4	Maruti Swift	2019	15445.0	Petrol	Automatic	673399.0

- ● Data Preprocessing Done

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

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

```
Car Model            0
Year                 0
Kilometers Driven    0
Fuel Type            0
Transmission Type    48
Price                0
dtype: int64
```

The missing values have been found in column 'Transmission Type' and based on several factors I used fillna method to fill the missing values.

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

```
Car Model             0
Year                  0
Kilometers Driven     0
Fuel Type             0
Transmission Type     0
Price                 0
dtype: int64
```

```
1  missingno.bar(df,color='blue')
```

<AxesSubplot:>
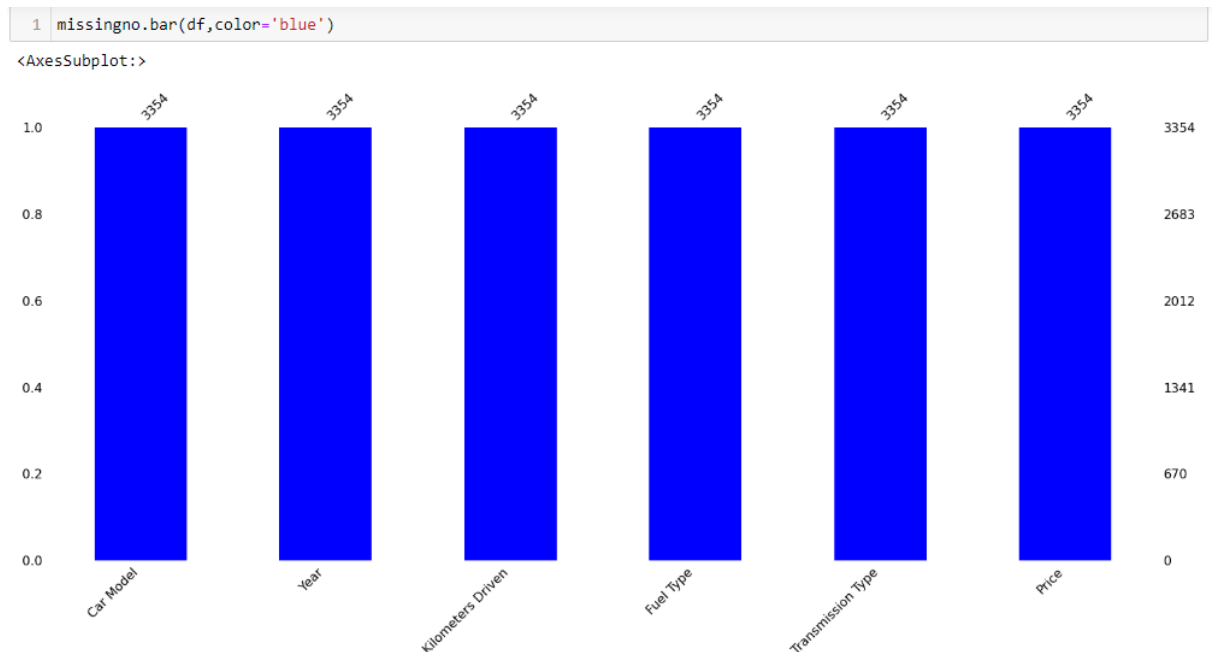
Checked the data type details for each column to understand the numeric ones and its further conversion process.

```
1  #Information about the dataset
2  df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3354 entries, 0 to 3353
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Car Model          3354 non-null   object
 1   Year               3354 non-null   int64
 2   Kilometers Driven  3354 non-null   float64
 3   Fuel Type          3354 non-null   object
 4   Transmission Type  3306 non-null   object
 5   Price              3354 non-null   float64
dtypes: float64(2), int64(1), object(3)
memory usage: 157.3+ KB
```

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

```
1  df.describe()
```

	Year	Kilometers Driven	Price
count	3354.000000	3354.000000	3.354000e+03
mean	2016.773703	45547.321705	5.103039e+05
std	2.971745	30472.565539	4.540507e+05
min	1999.000000	141.000000	1.000000e+00
25%	2015.000000	23092.250000	1.750000e+01
50%	2017.000000	40288.500000	4.743490e+05
75%	2019.000000	63000.000000	7.284740e+05
max	2022.000000	390000.000000	2.194499e+06

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.

```
1  plt.figure(figsize=(10,10))
2  sns.heatmap(df.describe(),annot=True,linecolor='yellow',linewidths=3)
```

<AxesSubplot:>
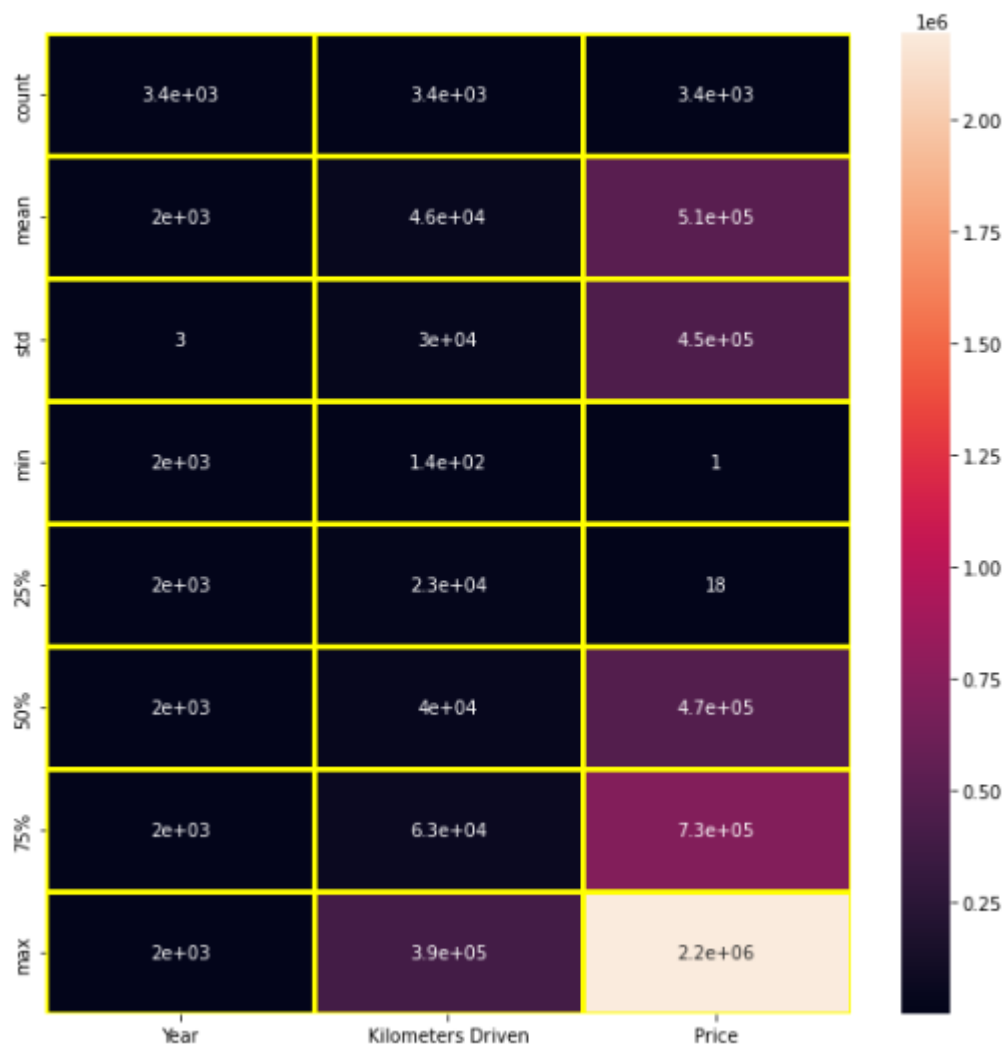
Data Inputs- Logic- Output Relationships

The input data were initially all object data type 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   : AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx 2.10 GHz

GPU : AMD Radeon ™ Vega 8 Graphics and NVIDIA GeForce GTX 1650 Ti

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
- Decision Tree Regression Model
- Random Forest Regression Model
- K Neighbours Regression Model
- Gradient Boosting Regression Model
- XGBoost Model

● Run and Evaluate selected models

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

Code:

```
1  x_train, x_test, y_train, y_test = train_test_split(X_scaled, Y, test_size=0.30, random_state=61)
2
3  #training our model
4  xgbt=xgb.XGBRegressor()
5  xgbt.fit(x_train, y_train)
6
7  #Predicting y_test
8  y_xgbt = xgbt.predict(x_test)
9  xgbt_r2 = r2_score(y_test, y_xgbt)
10
11 #R2 Score
12 print("R2 score : ", xgbt_r2)
13
14 #Cross Validation Score
15 cross_score_xgbt = cross_val_score(xgbt,X_scaled,Y,cv=5)
16 print(cross_score_xgbt)
17 print("Cross Validation Score : ",cross_score_xgbt.mean())
18
19 #Error Estimation Methods
20 print("Mean Absolute Error :",mean_absolute_error(y_test,y_xgbt))
21 print("Mean Square Error : ",mean_squared_error(y_test,y_xgbt))
22 print("Root Mean Square Error :",np.sqrt(mean_squared_error(y_test,y_xgbt)))
```

Output:

```
R2 score :   0.5415197783546648
[ 5.19193089e-02 -2.77596994e-01  3.99492164e-02  1.08452458e-01
 -2.48342694e+03]
Cross Validation Score :  -496.70084257046994
Mean Absolute Error : 176268.6771682403
Mean Square Error :  62236339718.163635
Root Mean Square Error : 249472.12212622803
```

- 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:

```
1  final_model = xgb.XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
2              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=0.3,
3              early_stopping_rounds=None, enable_categorical=False,
4              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
5              importance_type=None, interaction_constraints='',
6              learning_rate=0.1, max_bin=256, max_cat_to_onehot=4,
7              max_delta_step=0, max_depth=10, max_leaves=0, min_child_weight=1, monotone_constraints='()', n_estimators=200,
8              num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
9              reg_lambda=1)
10 print(final_model)
11 print("-----------------------------------------------------------------")
12 final_model.fit(x_train,y_train)
13 final_model_pred = final_model.predict(x_test)
14 print(final_model_pred)
15 print("R2 Score :",r2_score(y_test,final_model_pred))
16 print("-----------------------------------------------------------------")
17 print("Mean Absolute Error :",mean_absolute_error(y_test,final_model_pred))
18 print("Mean Square Error : ",mean_squared_error(y_test,final_model_pred))
19 print("Root Mean Square Error :",np.sqrt(mean_squared_error(y_test,final_model_pred)))
20 print("-----------------------------------------------------------------")
```

Final model score after plugging in the best parameter values:

```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=0.3,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.1, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=10, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=200, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, ...)
-------------------------------------------------------------------
[462088.34 952450.75 300632.66 ... 368083.12 980642.56 424966.84]
R2 Score : 0.5586933139664205
-------------------------------------------------------------------
Mean Absolute Error : 183130.16664674188
Mean Square Error :  59905120297.92436
Root Mean Square Error : 244755.2252719528
```

● 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)
```

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:

Overview

Overview   Warnings **8**   Reproduction

Dataset statistics

Number of variables	6
Number of observations	10000
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	1174
Duplicate rows (%)	11.7%
Total size in memory	429.8 KiB
Average record size in memory	44.0 B

Variable types

Categorical	3
Numeric	3

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.
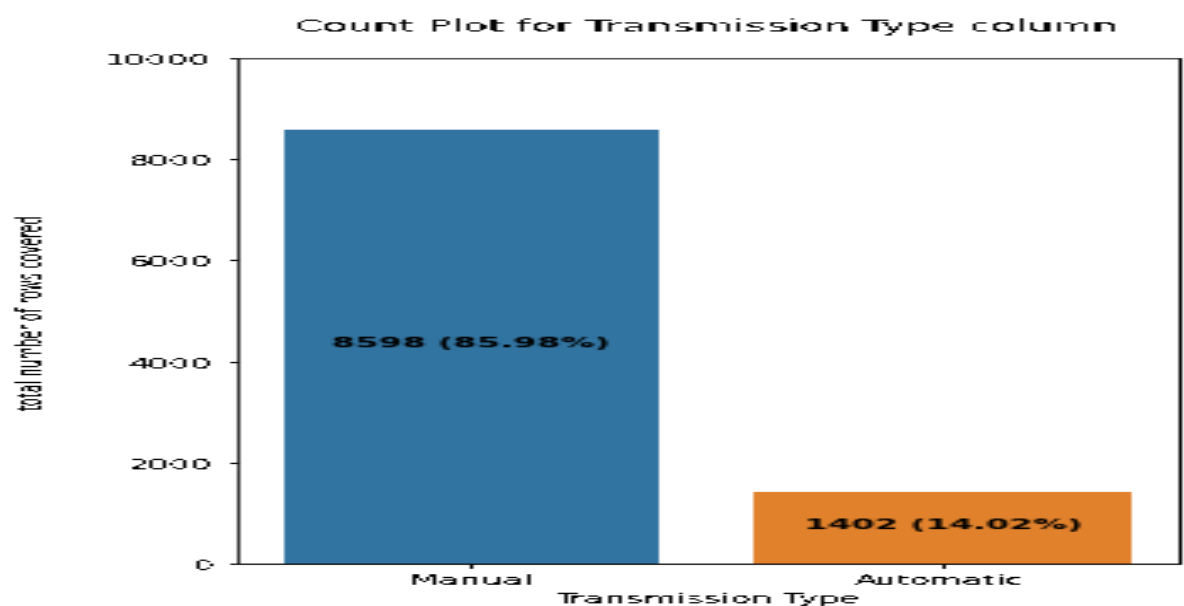
Code:

```python
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:

```python
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()
```
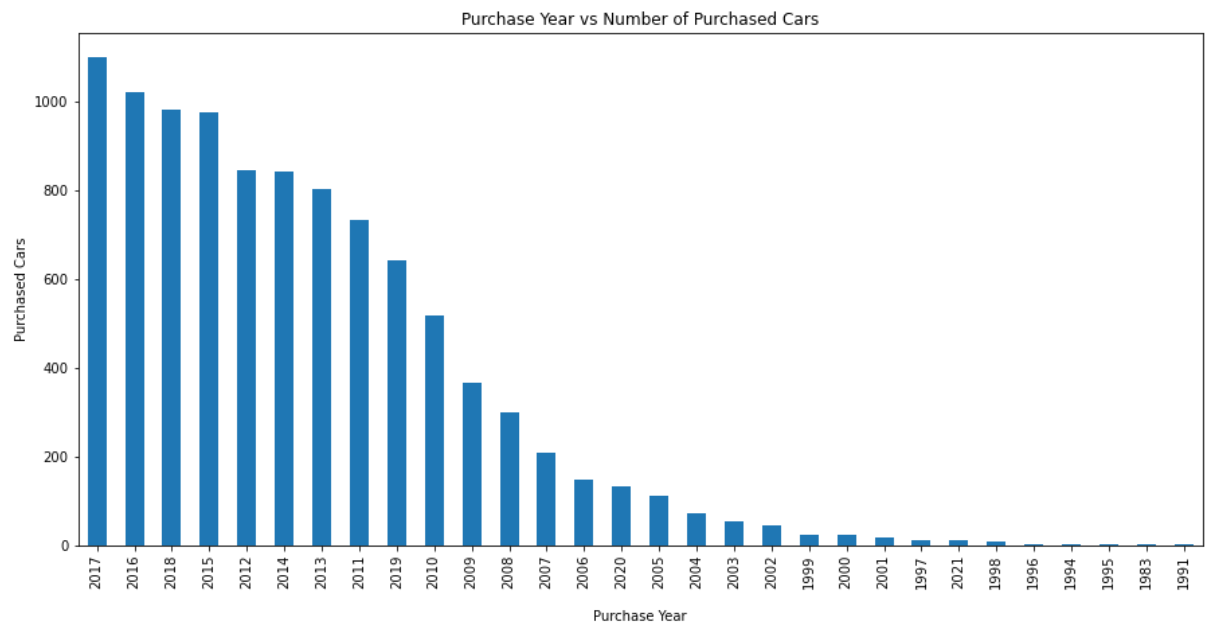
Output:

Barplot for Year of Manufacture column vs Transmission Type column
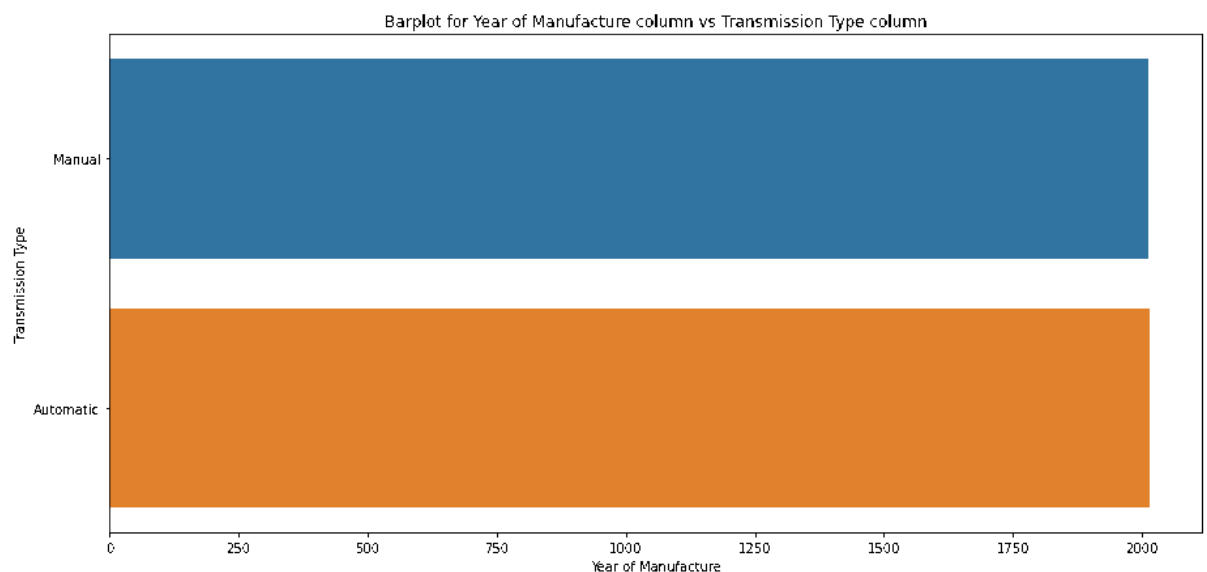
Code:

```python
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:

Purchase Year vs Number of Purchased Cars

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()
```
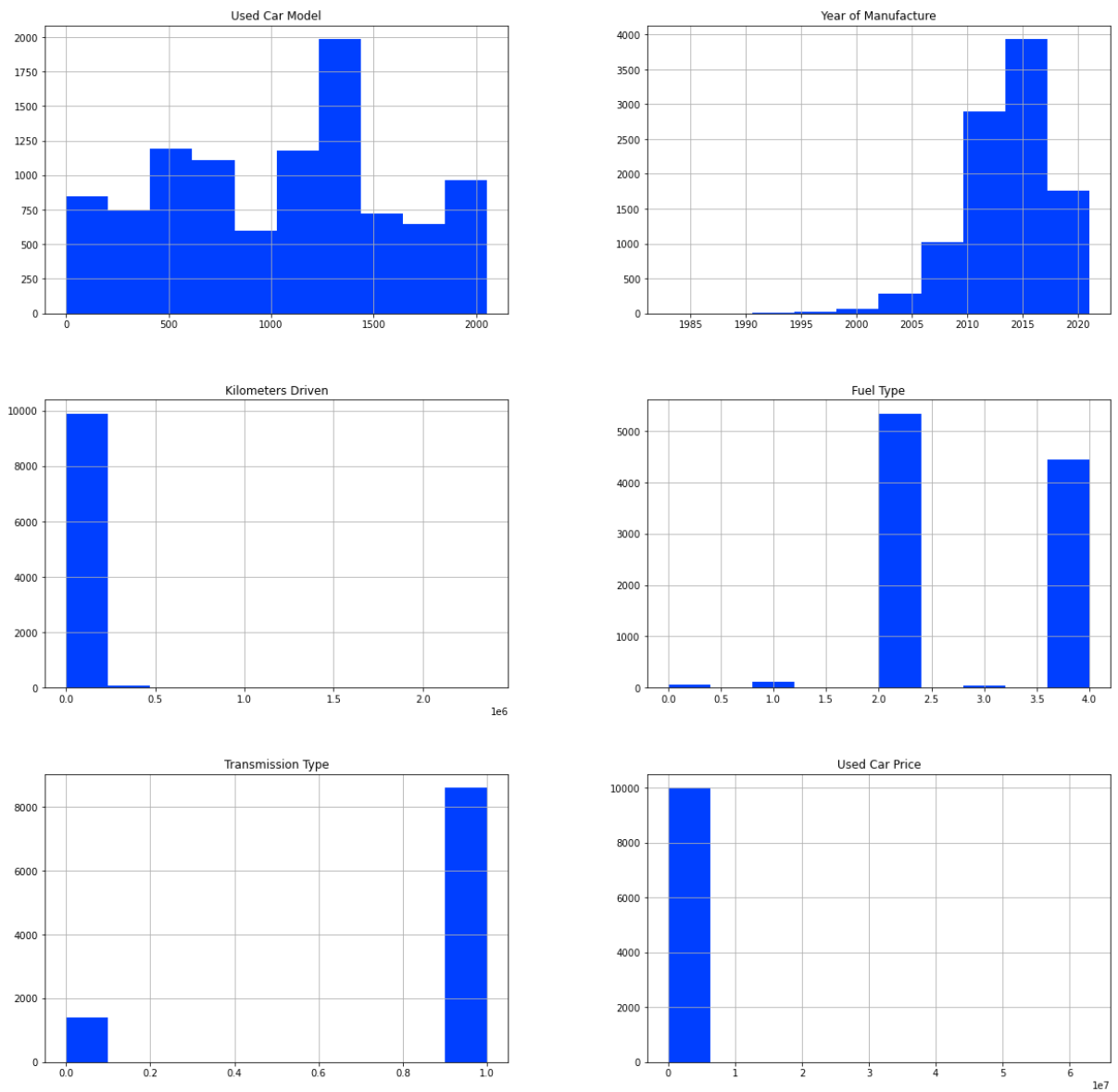
Output:
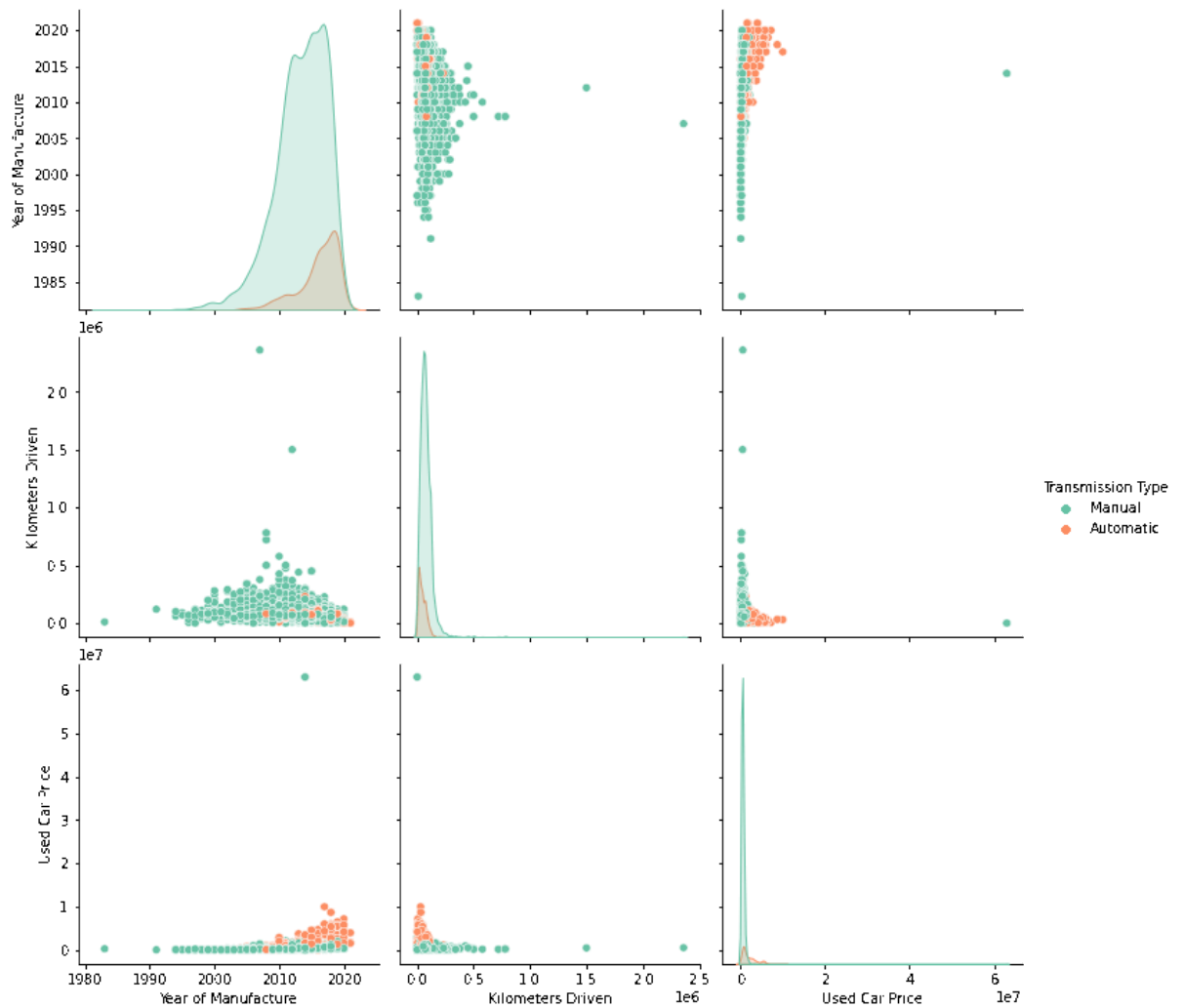
Code:

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

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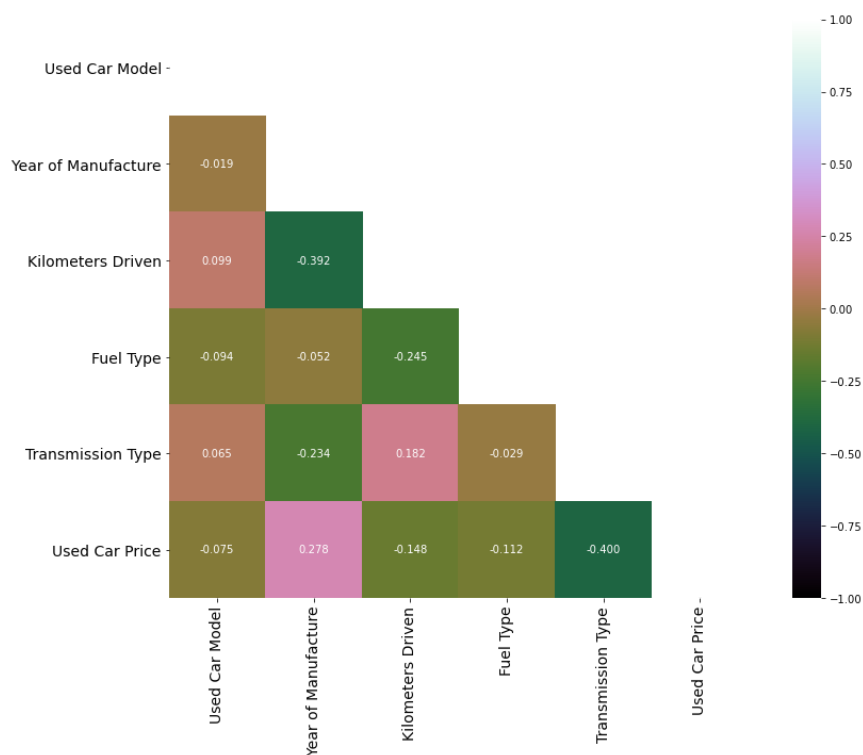
Output:

Code:

```python
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()
```
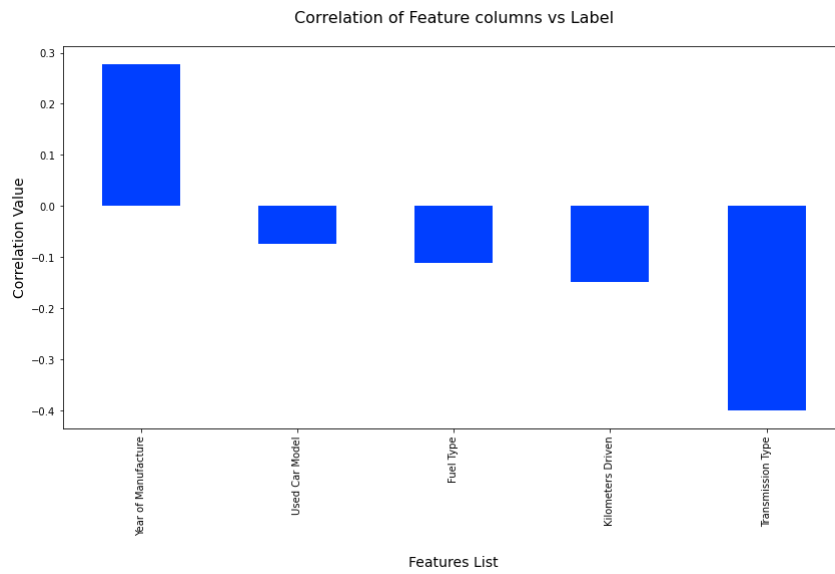
Output:

Code:

```python
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:

Correlation of Feature columns vs Label

Outliers and Skewness before and after treating them:
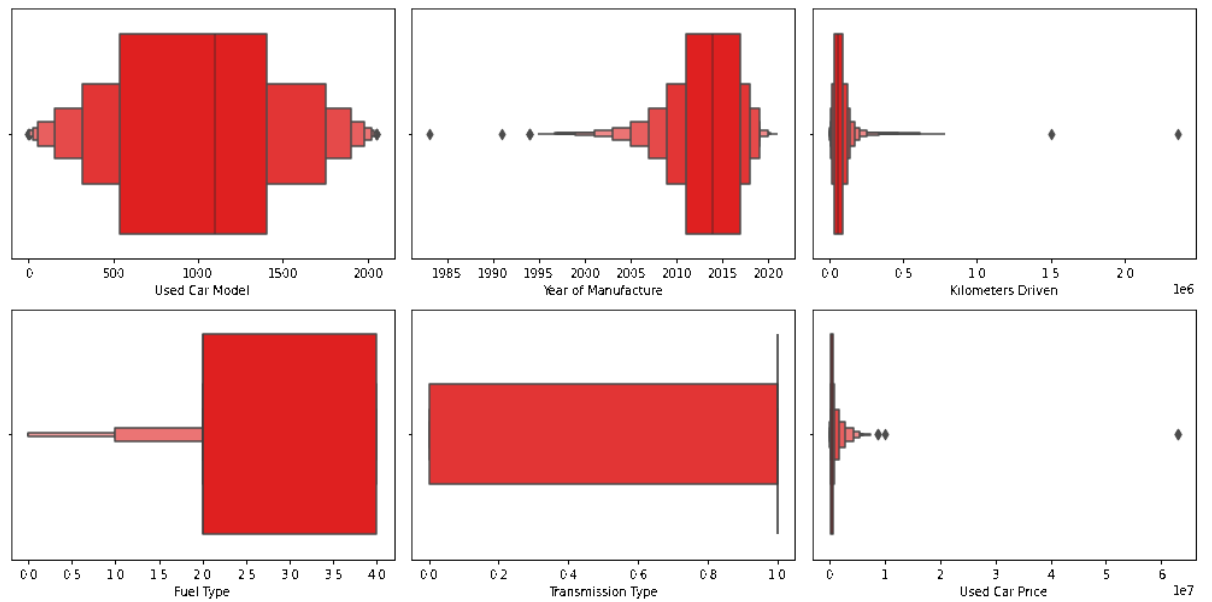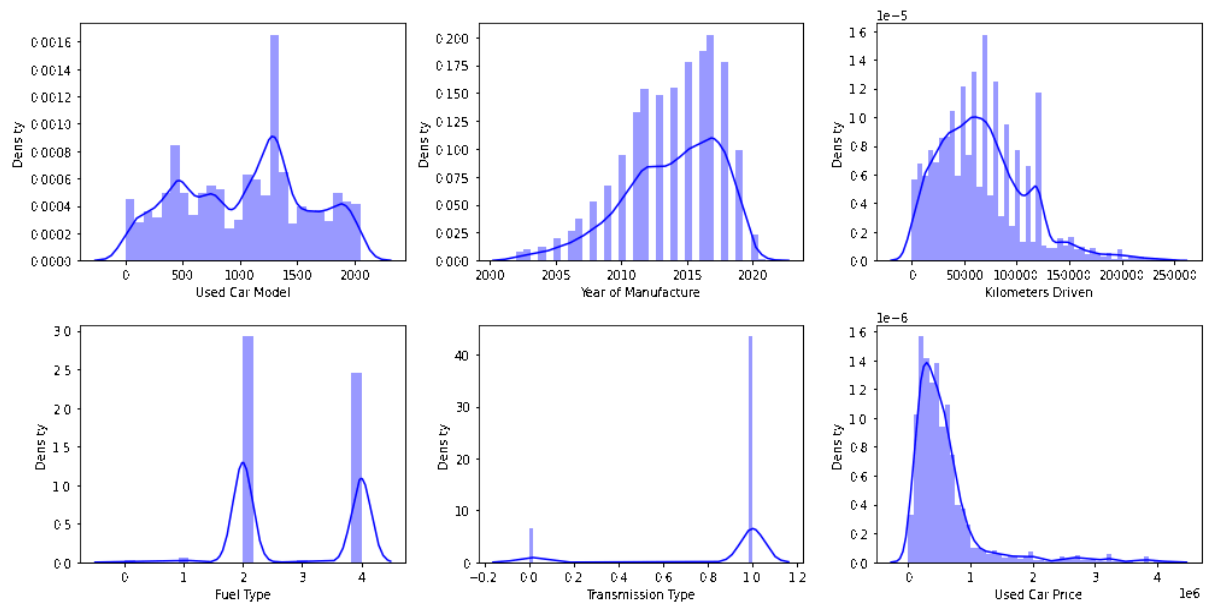
Code:

```python
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
```
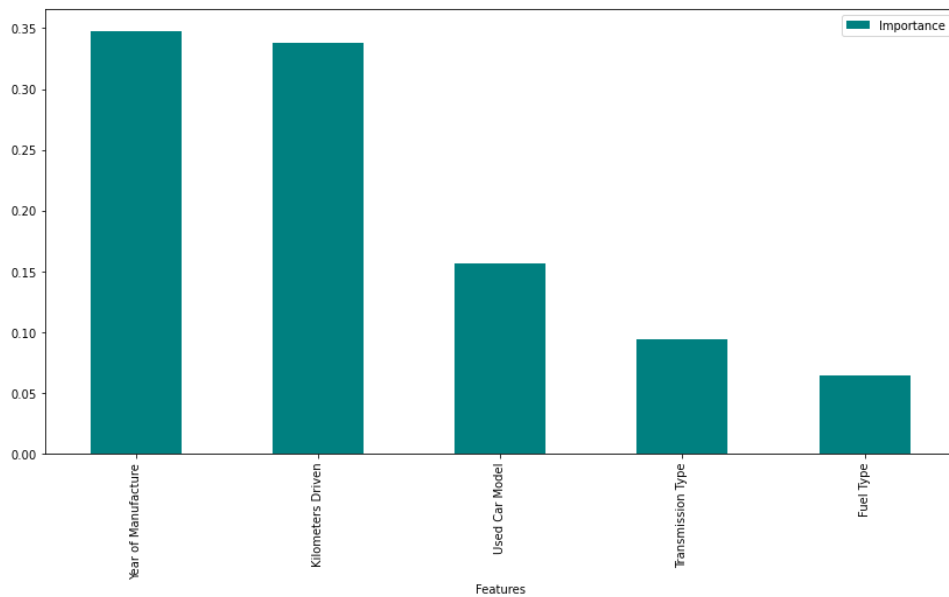
Output:

● 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 numbers of columns. Also, 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 Car Dekho, Cars 24, 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 scraped 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 the XGBOOST 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 as 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

  The 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 was 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 the next page. Also need further practice 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 the 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.