USED CAR PRICE PREDICTION

Submitted By: Girija Chandra Mohan

ACKNOWLEDGMENT

I would like to express my gratitude to my guide Khusboo Garg (SME, Flip Robo) for her constant guidance, encouragement and unconditional help towards the development of this Used Car Price Prediction project. She helped me whenever I got stuck somewhere in between. The project would have not been completed without her support and confidence.

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. All the external resources that were used in creating this project are listed below:

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/

9) https://www.olx.com/

Business Problem Framing

Impact of COVID-19 on Indian automotive sector

The Indian automotive sector was already struggling in FY20. before the Covid-19 crisis. It saw an overall degrowth of nearly 18 per cent. This situation was worsened by the onset of the Covid-19 pandemic and the ongoing lockdowns across India and the rest of the world. These two years (FY20 and FY21) are challenging times for the Indian automotive sector on account of slow economic growth, negative consumer sentiment, BS-VI transition, changes to the axle load norms, liquidity crunch, low-capacity utilisation and potential bankruptcies.

The return of daily life and manufacturing activity to near normalcy in China and South Korea, along with extended lockdown in India, gives hope for a U-shaped economic recovery. Our analysis indicates that the Indian automotive sector will start to see recovery in the third quarter of FY21. We expect the industry demand to be down 15-25 per cent in FY21. With such degrowth, 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 the big positive for auto sector.

Conceptual Background of the Domain Problem

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 these stuffs 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

As per the requirement of our client, I have scrubbed data from OLX used cars selling merchants websites, and so based on the data collected I have tried analyzing based on what factors the used car price is decided? What is the relationship between cost of the used cars and other factors like Fuel type, Brand and year the car is purchased And so based on all the above consideration I have developed a model that will predict the price of the used cars.

Motivation for the Problem Undertaken

I have taken this problem based on the requirement of the client and also, with a curiosity to know how the used cars markets are after the time of pandemic.

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 modeling 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, we need to build a machine learning model. Before model building we shall 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 Visualization
4. Model Building
5. Model Evaluation
6. Selecting the best model

Data Sources and their formats

The Data is scrubbed on ecommerce website that sells used cars in India, OLX . These data are scrubbed and stored in a CSV format. Data contains following columns.

1. 'LOCATION' – It will tell which location the car is sold.
2. 'MNF_YEAR' – At what year the car is manufactured
3. 'BRAND' – Brand is manufacturer or which company made
4. 'DRIVEN_KM' – no of Kms driven before selling
5. 'FUELTYPE' – Petrol, diesel, CNG, LPG, Electric
6. TRANSMISSION TYPE' – Mannual or Automatic
7. 'PRICE' – our target variable that tells what is the price of the used car.

```
df=pd.read_csv("used_cars.csv")
df.head()
```

	Used Car Model	Year of Manufacture	Kilometers Driven	Fuel Type	Transmission Type	Used Car Price	Location
0	Hyundai	2017	2,200 km	Petrol	Manual	5,25,000	Delhi
1	Hyundai	2013	91,500 km	Diesel	Manual	5,95,000	Delhi
2	Ford	2017	36,000 km	Diesel	Manual	7,75,000	Delhi
3	Honda	2015	90,000 km	Diesel	Manual	4,00,000	Delhi
4	Maruti Suzuki	2010	40,000 km	Petrol	Manual	2,30,000	Delhi

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.isnull().sum()
```

```
Used Car Model          0
Year of Manufacture     0
Kilometers Driven       0
Fuel Type               0
Transmission Type       0
Used Car Price          0
Location                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: 8348 entries, 0 to 8347
Data columns (total 7 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   Used Car Model        8348 non-null   object
 1   Year of Manufacture   8348 non-null   object
 2   Kilometers Driven     8348 non-null   object
 3   Fuel Type             8348 non-null   object
 4   Transmission Type     8348 non-null   object
 5   Used Car Price        8348 non-null   object
 6   Location              8348 non-null   object
dtypes: object(7)
memory usage: 456.7+ KB
```

I also took a look at all the unique value present in each of the columns and then decided to deal it with the imputation part accordingly.

```
df.nunique().sort_values()
```

```
Transmission Type         3
Fuel Type                 6
Location                  6
Year of Manufacture      31
Used Car Price          909
Kilometers Driven      1038
Used Car Model         1996
dtype: int64
```

The various data processing performed on the data set is shown below with the code.

```
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.head()
```

	Used Car Model	Year of Manufacture	Kilometers Driven	Fuel Type	Transmission Type	Used Car Price	Location
0	Hyundai	2017	2200	Petrol	Manual	5,25,000	Delhi
1	Hyundai	2013	91500	Diesel	Manual	5,95,000	Delhi
2	Ford	2017	36000	Diesel	Manual	7,75,000	Delhi
3	Honda	2015	90000	Diesel	Manual	4,00,000	Delhi
4	Maruti Suzuki	2010	40000	Petrol	Manual	2,30,000	Delhi

```
try:
    df["Used Car Price"]=df["Used Car Price"].apply(lambda x: x.split(' ')[1])
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"]=pd.to_numeric(df["Used Car Price"].str.replace('-','0'))
```

```
df["Used Car Price"]=df["Used Car Price"].astype(float)
```

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

To change year of Manufacture to numeric data, we have converted '-' to median values.

```
df["Fuel Type"]=df["Fuel Type"].apply(lambda x: x if x!='-' else df["Fuel Type"].mode()[0]) # replacing
df["Transmission Type"]=df["Transmission Type"].apply(lambda x: x if x!='-' else df["Transmission Type"
df["Used Car Model"]=df["Used Car Model"].apply(lambda x: x if x!='-' else df["Used Car Model"].mode()[
df["Kilometers Driven"]=df["Kilometers Driven"].apply(lambda x: x if x!='-' else 'None')
df["Used Car Price"]=df["Used Car Price"].apply(lambda x: x if x!='-' else df["Used Car Price"].mean()
```

Then I separated the categorical columns to see the unique values each columns represent to get to proper visualization.

```
#cat_col
cat_columns= [x for x in df.dtypes.index if df.dtypes[x] =='object']
cat_columns
```

```
['Used Car Model', 'Fuel Type', 'Transmission Type', 'Location']
```

```
#To get the unique value info of necessary columns
nes_col = ["Transmission Type", "Fuel Type", "Year of Manufacture"]
for col in nes_col:
    print(col)
    print(df[col].value_counts())
    print("="*120)
```

```
Transmission Type
Manual       7216
Automatic    1132
Name: Transmission Type, dtype: int64
========================================================================
====================
Fuel Type
Diesel           4501
Petrol           3697
CNG & Hybrids      54
```

Data Inputs- Logic- Output Relationships

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

```
LE= LabelEncoder()
```

```
for i in cat_columns:
    df[i]= LE.fit_transform(df[i])
```

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

```
df.describe()
```

	Used Car Model	Year of Manufacture	Kilometers Driven	Fuel Type	Transmission Type	Used Car Price	Location
count	8348.000000	8348.000000	8.348000e+03	8348.000000	8348.000000	8.348000e+03	8348.000000
mean	1014.285458	2013.725084	6.966386e+04	2.871346	0.864399	6.496622e+05	2.353019
std	536.701323	4.032961	5.849813e+04	1.024548	0.342385	1.063936e+06	1.697031
min	0.000000	1983.000000	0.000000e+00	0.000000	0.000000	0.000000e+00	0.000000
25%	560.000000	2011.000000	3.500000e+04	2.000000	1.000000	2.500000e+05	1.000000
50%	1071.000000	2014.000000	6.000000e+04	2.000000	1.000000	4.500000e+05	2.000000
75%	1370.000000	2017.000000	9.000000e+04	4.000000	1.000000	6.700000e+05	4.000000
max	1994.000000	2021.000000	2.360457e+06	4.000000	1.000000	6.300000e+07	5.000000

Then I used correlation method to find correlation between columns and with the target variable.

```
df.corr()
```

	Used Car Model	Year of Manufacture	Kilometers Driven	Fuel Type	Transmission Type	Used Car Price	Location
Used Car Model	1.000000	-0.023692	0.091993	-0.092895	0.050159	-0.066101	-0.004205
Year of Manufacture	-0.023692	1.000000	-0.395833	-0.053734	-0.239915	0.315987	-0.013616
Kilometers Driven	0.091993	-0.395833	1.000000	-0.244240	0.191012	-0.169436	0.013132
Fuel Type	-0.092895	-0.053734	-0.244240	1.000000	-0.033687	-0.145213	-0.002057
Transmission Type	0.050159	-0.239915	0.191012	-0.033687	1.000000	-0.452446	0.011880
Used Car Price	-0.066101	0.315987	-0.169436	-0.145213	-0.452446	1.000000	-0.004531
Location	-0.004205	-0.013616	0.013132	-0.002057	0.011880	-0.004531	1.000000

```
plt.figure(figsize=(15,7))
sns.heatmap(df.corr(), annot=True)
```

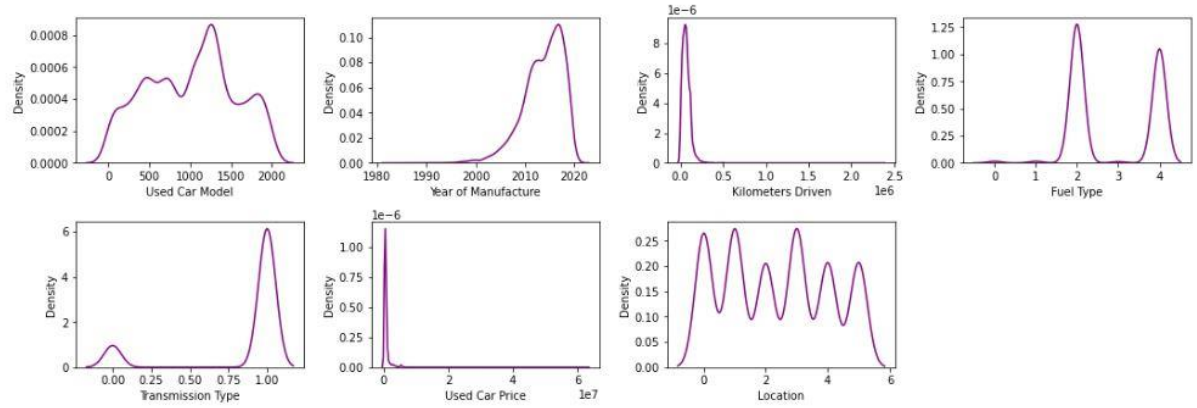
```
<AxesSubplot:>
```

Then I used Power Transform to remove Skewness from the dataset.

```
df.skew().sort_values(ascending=True)
```

```
Transmission Type      -2.129099
Year of Manufacture    -1.011707
Used Car Model         -0.064684
Fuel Type               0.078609
Location                0.100319
Kilometers Driven      10.864394
Used Car Price         25.960253
dtype: float64
```

```
plt.figure(figsize=(15,15))
for i in range (0, len(df.columns)):
    plt.subplot(6,4,i+1)
    sns.kdeplot(df[df.columns[i]], color = "purple")
    plt.tight_layout()
```




```
from sklearn.preprocessing import power_transform
df1_new= power_transform(df1)
```

Then I used zscore technique to remove outliers.

```
from scipy.stats import zscore
z=np.abs(zscore(df1))
z
```

```
# threshold for zscore is 3........., zscore greater than 3 is outliers
threshold=3
print(np.where(z>3))
```

```
df_new= df1[(z<3).all(axis=1)]
```

Hardware and Software Requirements and Tools Used

1. Python 3.8.
2. NumPy.
3. Pandas.
4. Matplotlib.
5. Seaborn.
6. Data science.
7. SciPy
8. Sklearn.
9. Anaconda Environment & Jupyter Notebook

Hardware and Software Requirements and Tools Used
Open source web-application used for programming:

1. Jupyter Notebook

Python Libraries / Packages used were:

1. **Pandas**: pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

We have used pandas to import the csv file using pd.read_csv all data analysis have been done using the pandas and numpy libraries. The

data characteristics have been studied using pandas functions like df.shape(), df .dtypes, df.columns etc.

2. **NumPy:** NumPy is an open-source numerical Python library. NumPy contains a multi-dimensional array and matrix data structures. It can be utilised to perform a number of mathematical operations on arrays such as trigonometric, statistical, and algebraic.

We have used the np.where function many times while dealing with the z-scores.np.abs () function has also been used to find the zscore and some mathematical operations like square root.

3. **Matplotlib**: library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

The Matplotlib libraries pyplot function is used for making plots ,plt.show() ,plt.figure(figsize) that has been used is a part of matplotlib library.

4. **Seaborn**: Seaborn is a Python data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

All the visualizations made are built using the seaborn library. Alias used for seaborn is sns.
Sns.boxplot(), sns.heatmap(), sns.distplot(), sns.scatterplot() ,sns.stripplot,sns.swarmplot ,heatmap are few of the libraries used

5. **SciPy**: SciPy, a scientific library for Python is an open source library for mathematics, science and engineering. The SciPy library depends on NumPy, which provides convenient and fast N-dimensional array

manipulation. The main reason for building the SciPy library is that, it should work with NumPy arrays.
In this particular project scipy functions such as scipy.stats is used .Zscores are also obtained via scipy.stats library

6. **Sklearn:** Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, and clustering and dimensionality reduction via a consistence interface in Python.

All the Machine Learning regression algorithms have been imported from the sklearn package. Simple imputer used is also a part of sklearn The evaluation metrics, RMSE,MSE,MAE functions are also imported from same.

7. **Python:** is a set of Data analysis tools in python which gives us measures of association for categorical features, Plots features correlation and association for mixed data-sets (categorical and continuous features) in an easy and simple way.

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods).

Considering the business requirement provided, I have collected the data to predict the used car price but there where multiple data of the car that are available. Data like colour of the car, sun roof attached, music system brand, electronics in the car, tyre brands, seat colours and much more. But after analysing all these data I have selected the data that have more correlation with the price of the car. Data like

manufacturing year, fuel variant, gear shift variant, Brand of the car, kilometers driven and Location of the car. I studied and visualized how these variables contribute to the deciding factor of the car price. Based on such visualization I have built the models.

For this project we need to predict the sale price of cars, means our target column is continuous so this is a regression problem. I have used various regression algorithms and tested for the prediction. By doing various evaluations I have selected RandomForest Regressor as best suitable algorithm for our final model as it is giving good r2-score and least difference in r2-score and CV-score among all the algorithms used. Other regression algorithms are also giving good accuracy but some are over-fitting and some are with under-fitting.

In order to get good performance as well as accuracy and to check my model from over-fitting and under-fitting I made hyper parameter tuning for the final model.

Once I was able to get the desired final model I ensured to save that model to obtain the predicted sale price out of the Regression Machine Learning Model.

The algorithms used on data are as follows:

- Linear Regression Model
- Ridge Regularization Regression Model
- Lasso Regularization Regression Model
- Support Vector Regression Model
- Decision Tree Regression Model
- Random Forest Regression Model
- K Nearest Neighbours Regression Model

Run and Evaluate selected models

We have used a total of 7 Regression Models after choosing the random state amongst 1-100 number. Then it is checked for best fold for CV using the following codes.

```
maxr2=0
maxRS=0

for i in range(1,200):
    xtrain,xtest,ytrain,ytest= train_test_split(x,y,test_size=20,random_state=i)
    lr=LinearRegression()
    lr.fit(xtrain,ytrain)
    predlr= lr.predict(xtest)
    R2=r2_score(ytest,predlr)
    if R2>maxr2:
        maxr2=R2
        maxRS=i
print("Best R2_score is", maxr2, 'on Random_state',maxRS)
```

```
Best R2_score is 0.8519386589017288 on Random_state 49
```

```
pred_tr= lr.predict(xtrain)
pred_ts= lr.predict(xtest)
```

```
train_accuracy= r2_score(ytrain,pred_tr)
test_accuracy= r2_score(ytest,pred_ts)
```

```
from sklearn.model_selection import cross_val_score
for j in range(2,10):
    cv_score= cross_val_score(lr,x,y,cv=j)
    cv_mean=cv_score.mean()
    print(f"At cross fold {j} the cv score is {cv_mean} and accuracy score for training is {train_accuracy} and the accuracy for
    print("\n")
```

```
At cross fold 2 the cv score is 0.4974768879238155 and accuracy score for training is 0.49933326363470587 and the accuracy for
testing is 0.6352316187683589


At cross fold 3 the cv score is 0.49763786019362016 and accuracy score for training is 0.49933326363470587 and the accuracy for
testing is 0.6352316187683589


At cross fold 4 the cv score is 0.4947333451353525 and accuracy score for training is 0.49933326363470587 and the accuracy for
testing is 0.6352316187683589


At cross fold 5 the cv score is 0.4944676008996935 and accuracy score for training is 0.49933326363470587 and the accuracy for
testing is 0.6352316187683589
```

Regression Model Function

```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=20, random_state=49)
```

Linear Regression

```
#Linear Regression

lr=LinearRegression()
lr.fit(xtrain,ytrain)
predlr= lr.predict(xtest)
print('R2=', r2_score(ytest,predlr))
print('RMSE=', np.sqrt(mean_squared_error(ytest,predlr)))
print('MAE=', mean_absolute_error(ytest,predlr))
```

```
R2= 0.8519386589017288
RMSE= 151252.54390632082
MAE= 119645.55393430719
```

```
cv_score=cross_val_score(lr,x,y, cv=3)
print('Cross Validation Score of LinearRegression is:', cv_score.mean())
```

```
Cross Validation Score of LinearRegression is: 0.49763786019362016
```

```
diff= 0.8519- 0.4976
diff
```

```
0.3543
```

Ridge Regression

```
# Ridge Regression

rd=Ridge(alpha=1e-2, normalize=True)
rd.fit(xtrain,ytrain)
predrd= rd.predict(xtest)
print('R2=', r2_score(ytest,predrd))
print('RMSE=', np.sqrt(mean_squared_error(ytest,predrd)))
print('MBE=', mean_absolute_error(ytest,predrd))
```

```
R2= 0.8515510878532258
RMSE= 151450.3767589356
MBE= 119017.0626608385
```

```
cv_score=cross_val_score(rd,x,y, cv=3)
print('Cross Validation Score of RidgeRegression is:', cv_score.mean())
```

```
Cross Validation Score of RidgeRegression is: 0.49764996821710544
```

```
diff= 0.8515- 0.4976
diff
```

```
0.35390000000000005
```

Lasso Regression

```
# Lasso Regression

ls=Lasso(alpha=1e-2, normalize=True)
ls.fit(xtrain,ytrain)
predls= ls.predict(xtest)
print('R2=', r2_score(ytest,predls))
print('RMSE=', np.sqrt(mean_squared_error(ytest,predls)))
print('MBE=', mean_absolute_error(ytest,predls))
```

```
R2= 0.8519402068834074
RMSE= 151251.75323139835
MBE= 119644.59781313362
```

```
cv_score=cross_val_score(ls,x,y, cv=9)
print('Cross Validation Score of LassoRegression is:', cv_score.mean())
```

```
Cross Validation Score of LassoRegression is: 0.4972563151000655
```

```
diff=0.8519-0.4976
diff
```

```
0.3543
```

DecisionTree Regression

```
# Decision Tree Regressor

dtc=DecisionTreeRegressor()
dtc.fit(xtrain,ytrain)
preddtc= dtc.predict(xtest)
print('Score:',dtc.score(xtrain,ytrain))
print('root_mean_squared_error', np.sqrt(mean_squared_error(ytest,preddtc)))
print('mean_absolute_error:', mean_absolute_error(preddtc,ytest))
print('r2_score:', r2_score(preddtc,ytest))
```

```
Score: 0.9948789483167475
root_mean_squared_error 109000.63989896268
mean_absolute_error: 73670.88333333333
r2_score: 0.930146730691516
```

```
cv_score=cross_val_score(dtc,x,y, cv=9)
print('Cross Validation Score of DecisionTreeRegressor is:', cv_score.mean())
```

```
Cross Validation Score of DecisionTreeRegressor is: 0.7785162070515667
```

```
diff= 0.9301-0.7785
diff
```

```
0.15160000000000007
```

Random Forest Regression

```
# Random Forest Regressor

rf=RandomForestRegressor()
rf.fit(xtrain,ytrain)
predrf= rf.predict(xtest)
print('root mean_squared_error', np.sqrt(mean_squared_error(ytest,predrf)))
print('mean_absolute_error:', mean_absolute_error(predrf,ytest))
print('Rf Score', rf.score(xtrain,ytrain))
print('r2_score:', r2_score(predrf,ytest))
```

```
root mean_squared_error 102860.09269043796
mean_absolute_error: 63769.345373015865
Rf Score 0.9831105004575759
r2_score: 0.9412285363230315
```

```
cv_score=cross_val_score(rf,x,y, cv=9)
print('Cross Validation Score of RandomForestRegressor is:', cv_score.mean())
```

```
Cross Validation Score of RandomForestRegressor is: 0.862004404437632
```

```
diff= 0.9412- 0.8620
diff
```

```
0.07920000000000005
```

Support Vector Regression

```
#SVR

svc=SVR()
svc.fit(xtrain,ytrain)
predsvc= svc.predict(xtest)
print('root mean_squared_error', np.sqrt(mean_squared_error(ytest,predsvc)))
print('mean_absolute_error:', mean_absolute_error(predsvc,ytest))
print('Score:', svc.score(xtrain,ytrain))
print('r2_score:', r2_score(predsvc,ytest))
```

```
root mean_squared_error 405223.68438796565
mean_absolute_error: 255201.56550282557
Score: -0.0482977267298883
r2_score: -1431979.7321459975
```

KNN

```
# K Neighbors Regressor

knn=KNeighborsRegressor()
knn.fit(xtrain,ytrain)
predknn= knn.predict(xtest)
print('root mean_squared_error', np.sqrt(mean_squared_error(ytest,predknn)))
print('mean_absolute_error:', mean_absolute_error(predknn,ytest))
print('Rf Score', knn.score(xtrain,ytrain))
print('r2_score:', r2_score(predknn,ytest))
```

```
root mean_squared_error 129396.7797219699
mean_absolute_error: 99130.04000000001
Rf Score 0.8294617219343752
r2_score: 0.9011673587839933
```

```
cv_score=cross_val_score(knn,x,y, cv=9)
print('Cross Validation Score of KNN is:', cv_score.mean())
```

```
Cross Validation Score of KNN is: 0.6775129191136643
```

```
diff= 0.9011-0.6775
diff
```

```
0.22360000000000002
```

Key Metrics for success in solving problem under consideration

The key metrics used here were r2_score, cross_val_score, MAE, MSE and RMSE. We tried to find out the best parameters and also to increase our scores by using Hyperparameter Tuning and we will be using GridSearchCV method.

1. Cross Validation:

Cross-validation helps to find out the over fitting and under fitting of the model. In the cross validation the model is made to run on different subsets of the dataset which will get multiple measures of the model. If we take 7 folds, the data will be divided into 7 pieces, where each part being 20% of full dataset. While running the Cross-validation the 1st part (17%) of the 7parts will be kept out as a holdout set for validation

and everything else is used for training data. This way we will get the first estimate of the model quality of the dataset.

In the similar way further iterations are made for the second 17% of the dataset is held as a holdout set and remaining 4 parts are used for training data during process. This way we will get the second estimate of the model quality of the dataset. These steps are repeated during the cross-validation process to get the remaining estimate of the model quality.

2. R2 Score:

It is a statistical measure that represents the goodness of fit of a regression model. The ideal value for r-square is 1. The closer the value of r-square to 1, the better is the model fitted.

3. Mean Squared Error (MSE):

MSE of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors — that is, the average squared difference between the estimated values and what is estimated. MSE is a risk function, corresponding to the expected value of the squared error loss. RMSE is the Root Mean Squared Error.

4. Mean Absolute Error (MAE):

MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

5. Hyperparameter Tuning:

There is a list of different machine learning models. They all are different in some way or the other, but what makes them different is nothing but input parameters for the model. These input parameters are named as Hyperparameters. These hyperparameters will define the architecture of the model, and the best part about these is that you get a choice to select these for your model. You must select from a specific list of hyperparameters for a given model as it varies from model to model.

We are not aware of optimal values for hyperparameters which would generate the best model output. So, what we tell the model is to explore and select the optimal model architecture automatically. This selection procedure for hyperparameter is known as Hyperparameter Tuning. We can do tuning by using GridSearchCV.

GridSearchCV is a function that comes in SK-learn model selection package. This function helps to loop through predefined hyperparameters and fit our estimator (model) on our training set. So, we can select the best parameters from the listed hyperparameters.

HyperParameter Tuning With GridSearchCV

```
from sklearn.model_selection import GridSearchCV
```

```
RandomForestRegressor()
```

```
RandomForestRegressor()
```

```
parameters={'max_features': ["auto", "sqrt", "log2"],
            'min_samples_leaf': [1,2,3],
            'criterion':["squared_error","absolute_error", "poisson"],
            'max_depth':[3,4,5,6],
            'min_samples_split': [2,3,4,5]}
```

```
GCV= GridSearchCV(RandomForestRegressor(), parameters, cv=9, scoring='r2')
GCV.fit(xtrain,ytrain)
GCV.best_params_
```

```
{'criterion': 'squared_error',
 'max_depth': 6,
 'max_features': 'auto',
 'min_samples_leaf': 1,
 'min_samples_split': 2}
```

```
GCV.best_estimator_
```

```
RandomForestRegressor(max_depth=6)
```

```
GCV_pred=GCV.best_estimator_.predict(xtest)
r2_score(ytest,GCV_pred)
```

```
0.59627699866985
```

```
#Final Model
flrf=RandomForestRegressor(max_depth=4,
                           min_samples_split=3,
                           min_samples_leaf=3,
                           max_features= 'log2',
                           criterion = 'squared_error')
flrf.fit(xtrain,ytrain)
predrf= flrf.predict(xtest)
print('root mean_squared_error', np.sqrt(mean_squared_error(ytest,predrf)))
print('mean_absolute_error:', mean_absolute_error(predrf,ytest))
print('Rf Score', rf.score(xtrain,ytrain))
print('r2_score:', r2_score(predrf,ytest))
```

```
root mean_squared_error 140578.05874328382
mean_absolute_error: 113477.58211537167
Rf Score 0.9831105004575759
r2_score: 0.8884742368094669
```

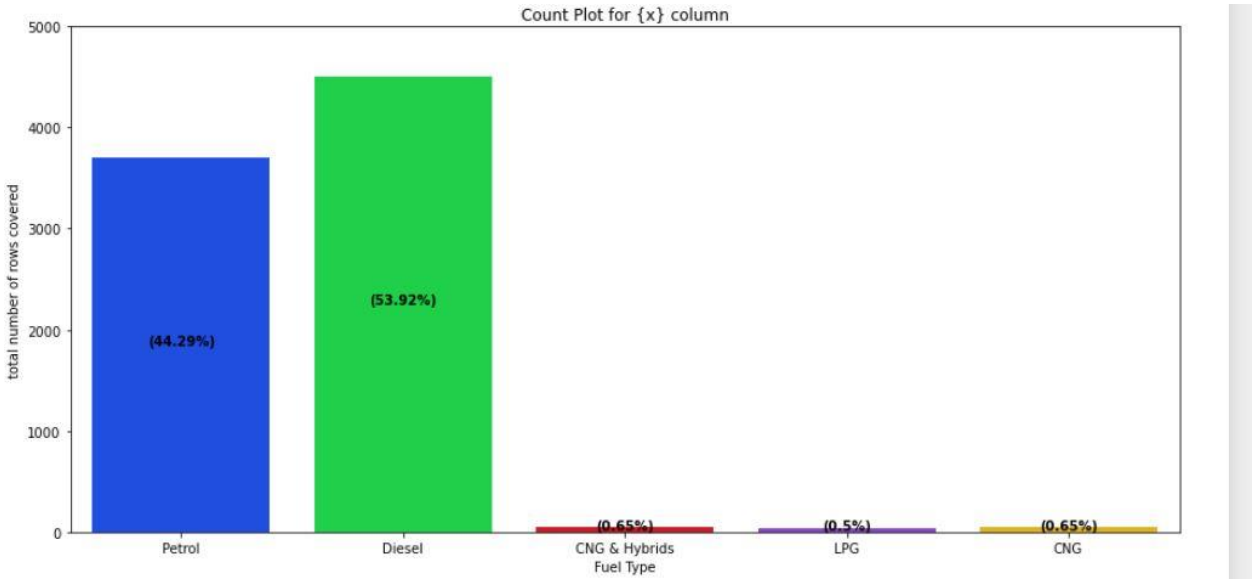
It is possible that there are times when the default parameters perform better than the parameters list obtained from the tuning and it only indicates that there are more permutations and combinations that one needs to go through for obtaining better results.

Visualizations

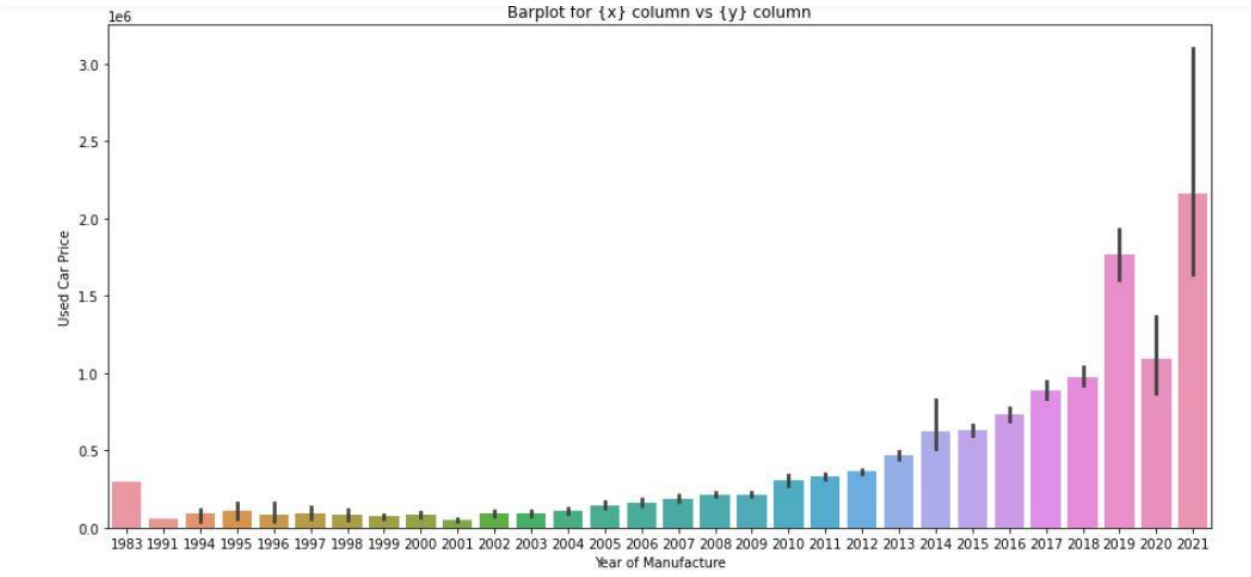
I have done some visualizations to understand the input output logic of the data collected.

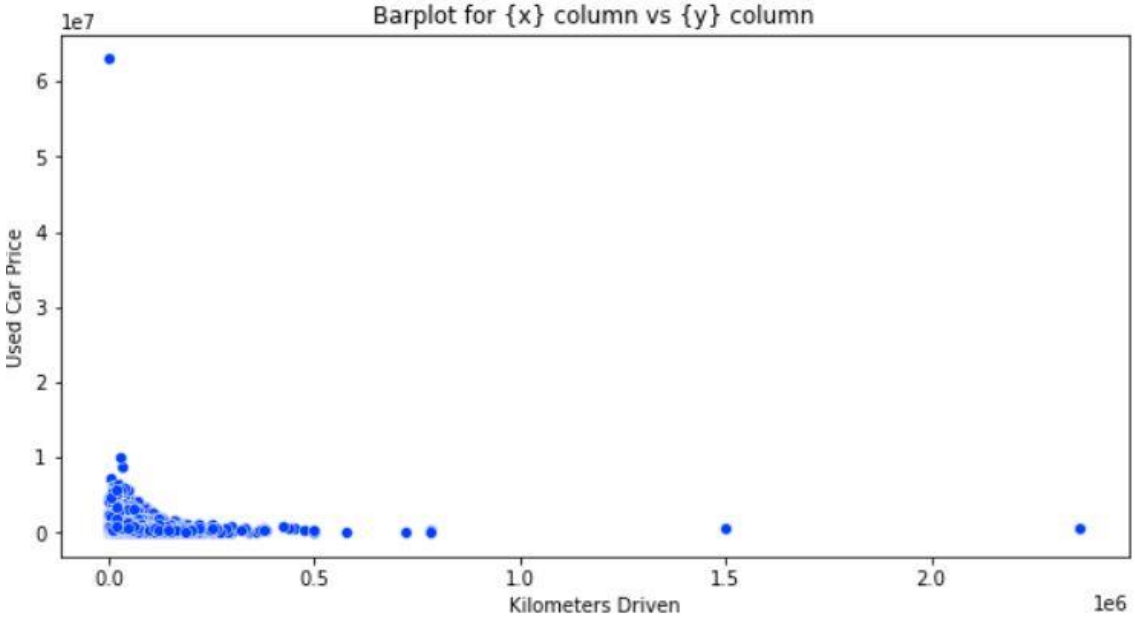
Univariate Analysis

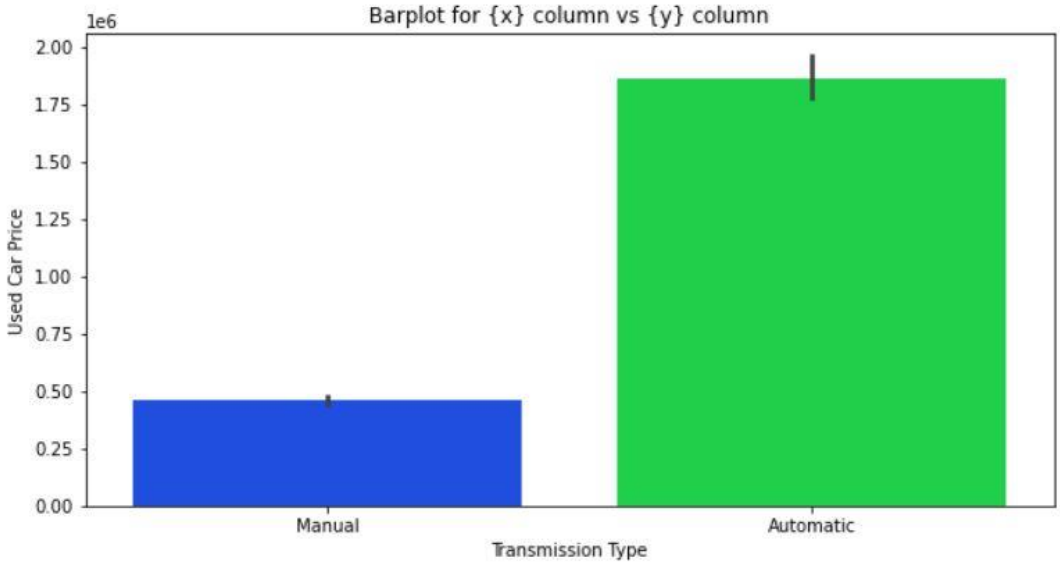
```
x = 'Fuel Type'
k=0
plt.figure(figsize=[15,7])
axes = sns.countplot(df[x])
for i in axes.patches:
    ht = i.get_height()
    ln = len(df[x])
    st = f"({round(ht*100/ln,2)}%)"
    plt.text(k, ht/2, st, ha='center', fontweight='bold')
    k += 1
plt.ylim(0,5000)
plt.title('Count Plot for {x} column')
plt.ylabel('total number of rows covered')
plt.show()
```



```
y = 'Used Car Price'

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

x = 'Kilometers Driven'
plt.figure(figsize=[10,5])
sns.scatterplot(x,y,data=df)
plt.title("Barplot for {x} column vs {y} column")
plt.show()

x = 'Transmission Type'
plt.figure(figsize=[10,5])
sns.barplot(x,y,data=df)
plt.title("Barplot for {x} column vs {y} column")
plt.show()
```

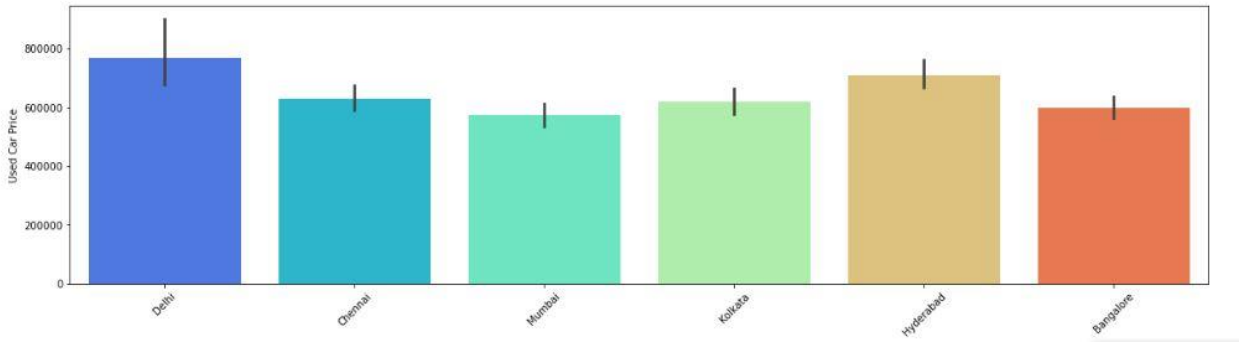



Barplot for {x} column vs {y} column
1e7
Used Car Price
Kilometers Driven
1e6

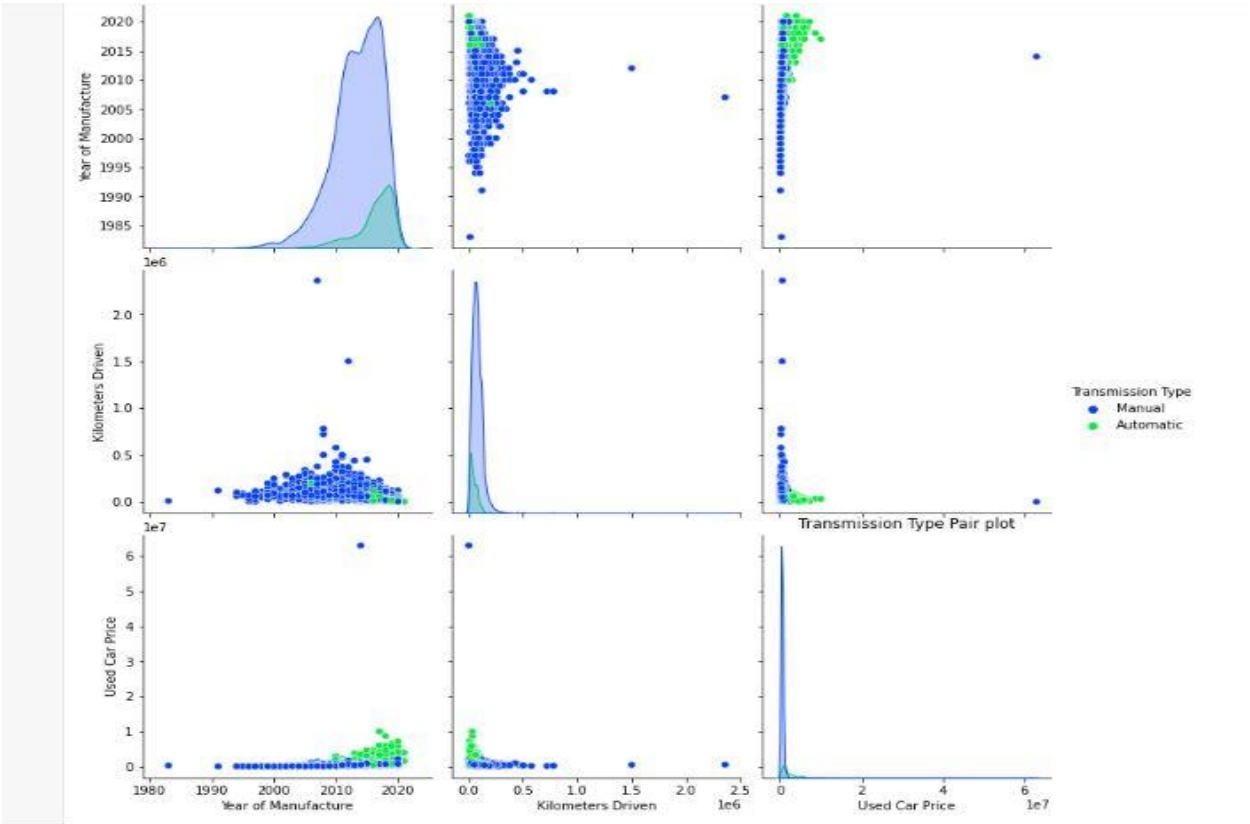

Barplot for {x} column vs {y} column
1e6
Used Car Price
Manual
Automatic
Transmission Type

Bivariate analysis

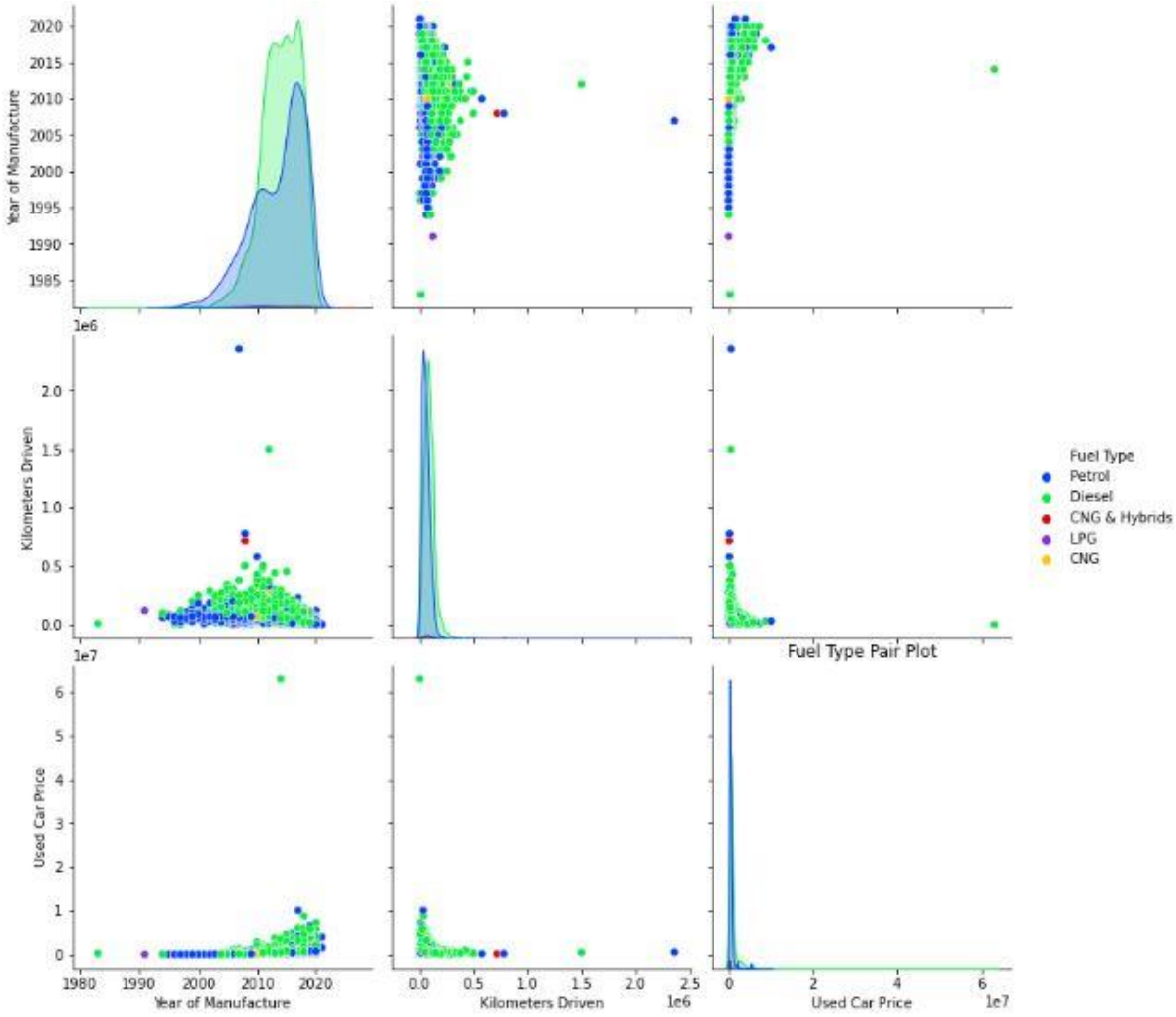
```
plt.figure(figsize=(20,5))
sns.barplot(x = "Location",y = "Used Car Price", data = df, palette = "rainbow")
plt.xticks(rotation=45)
plt.show()
```



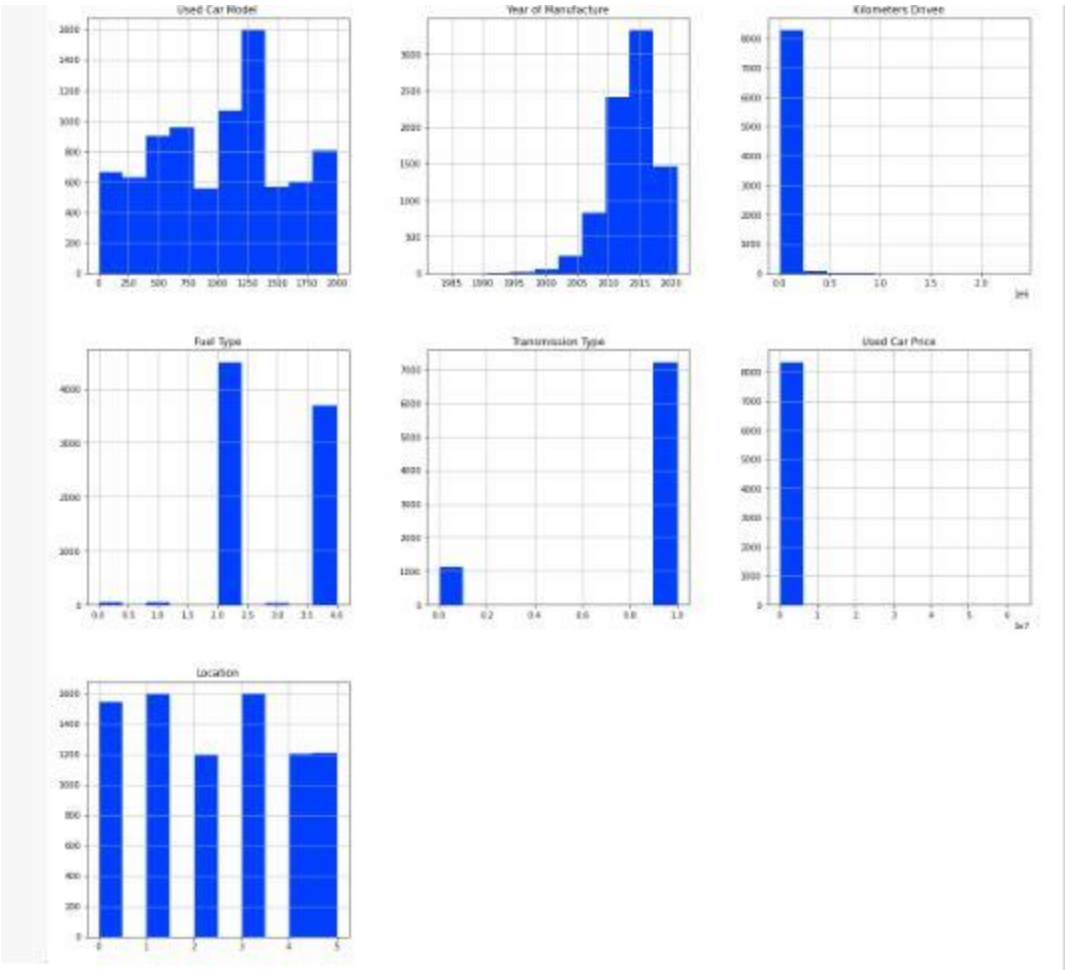

```
sns.pairplot(df, hue='Transmission Type', diag_kind="kde", kind="scatter", height=3.5)
plt.title('Transmission Type Pair plot')
plt.show()
```



```
sns.pairplot(df, hue='Fuel Type', diag_kind="kde", kind="scatter", height=3.5)
plt.title('Fuel Type Pair Plot')
plt.show()
```




```
plt.style.use('seaborn-bright')
df.hist(figsize=(20,20))
plt.show()
```



Interpretation of the Results:

From the visualization above we can clearly understand that the used car price factors are decided by the factors such as brand, location, year made, fuel type of the car.

From that it is clear that the used car price depends on the Brand that is the manufacturer and model it varies. The manufacturer like Land Rover, Benz, BMW cars are costliest used car in the market comparatively to other cars, the low kilometers driven and also if the manufacturing year is lesser on these brands those cars sells in much higher rates or closest to new car rates. The Diesel variant and automatic shift variants are also costliest car variants in the used car market.

Conclusions of the Study

After the completion of this project, I got an insight on how to collect data, pre-processing the data, analyzing the data and building a model. First, I collected the used cars data from different websites like OLX and it was done by using Web Scraping. The framework used for web scraping was Selenium, which has an advantage of automating the process of collecting data. I have collected almost 8400 data which contained the selling price and other related features of used cars. Then the scrapped data was converted to data frame and saved in a csv file so that we can open it and analyse the data. I performed 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, I was running different machine learning regression algorithms to find out the best performing model. I found that Random Forest Regressor Algorithm was performing well according to their r2_score and cross validation scores. Then I performed Hyperparameter Tuning technique using GridSearchCV for getting the best parameters and improving the score. In that Random Forest Regressor Algorithm did not perform quite well as previously on the defaults but I finalised that model for further predictions as it was still better than the rest. I saved the final model in pkl format using the pickle 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 Random Foprest 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 limitation of the study is that in the volatile changing market I have taken the data, to be more precise that the data is taken after the time of pandemic, so the market correction is happening slowly and stabilization phase is on. So based on that again the deciding factors of the used car prize might change and I have shortlisted and taken these data from the important cities across India, if the seller is from the different city our model might fail to predict the accurate prize of that used car.

Future Work Scope:

The overall score can also be improved further by training the model with more specific data from different cities and different model cars.