Data Science with Python Career Program - Capstone Project

- By Ravi Pandey

Agenda

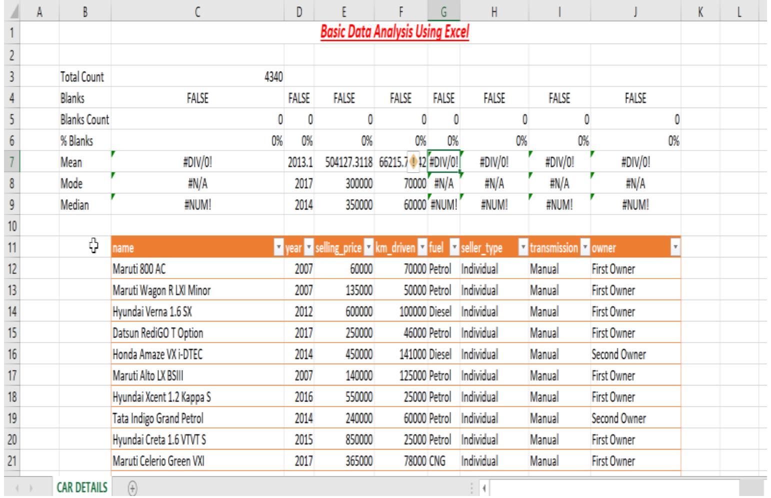
skill academy

- Data Exploration (Using Excel & Python)
- Data insights (Excel & Python)
- EDA Graphs
- Graphical Analysis and conclusion on Data
- Data Cleaning & Pre-Processing Steps
- ML Modeling
- Model Test Evaulation & Prediction Analysis
- Deployment of ML Models using Streamlit

Data Exploration (Using Excel)

skill academy

Basic Analysis Using Excel Sheet

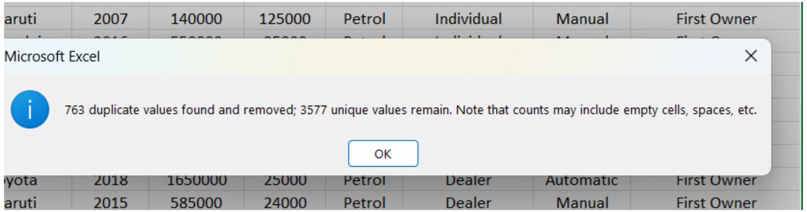


Basic Data Analysis Using Excel

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
Total Count	4340							
Blanks	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Blanks Count	0	0	0	0	0	0	0	0
% Blanks	0%	0%	0%	0%	0%	0%	0%	0%
Mean	#DIV/0!	2013.1	504127.3118	66215.7	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!
Mode	#N/A	2017	300000	70000	#N/A	#N/A	#N/A	#N/A
Median	#NUM!	2014	350000	60000	#NUM!	#NUM!	#NUM!	#NUM!

name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner
Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner
Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner
Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner
Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner
Maruti Alto LX BSIII	2007	140000	125000	Petrol	Individual	Manual	First Owner
Hyundai Xcent 1.2 Kappa S	2016	550000	25000	Petrol	Individual	Manual	First Owner
Tata Indigo Grand Petrol	2014	240000	60000	Petrol	Individual	Manual	Second Owner
Hyundai Creta 1.6 VTVT S	2015	850000	25000	Petrol	Individual	Manual	First Owner
Maruti Celerio Green VXI	2017	365000	78000	CNG	Individual	Manual	First Owner

CAR DETAILS



aruti	2007	140000	125000	Petrol	Individual	Manual	First Owner

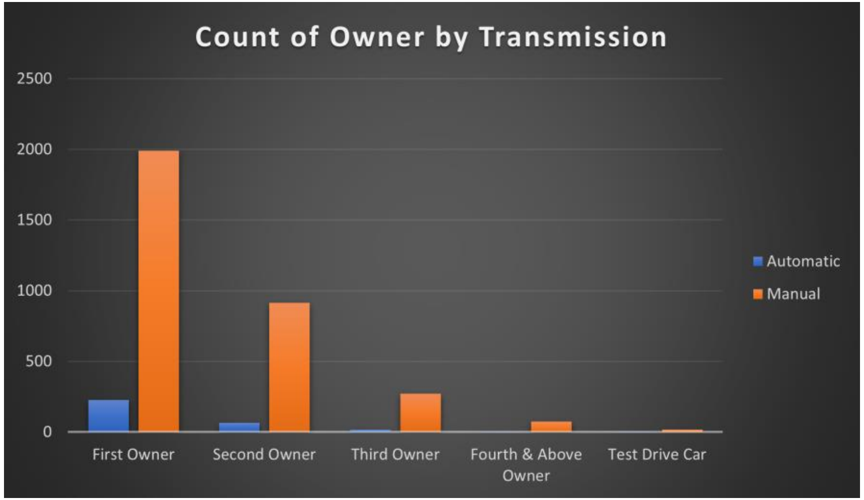
Microsoft Excel

763 duplicate values found and removed; 3577 unique values remain. Note that counts may include empty cells, spaces, etc.

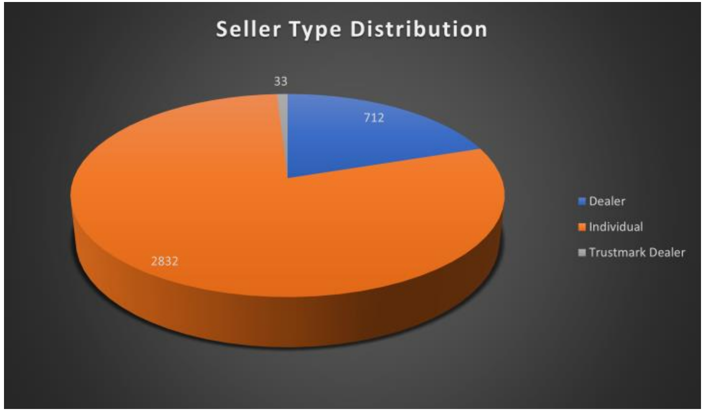
OK

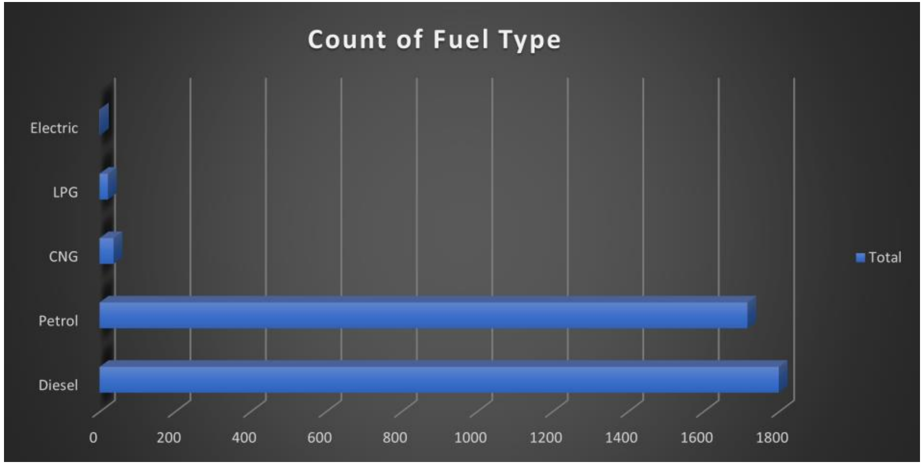
yota	2018	1650000	25000	Petrol	Dealer	Automatic	First Owner
aruti	2015	585000	24000	Petrol	Dealer	Manual	First Owner

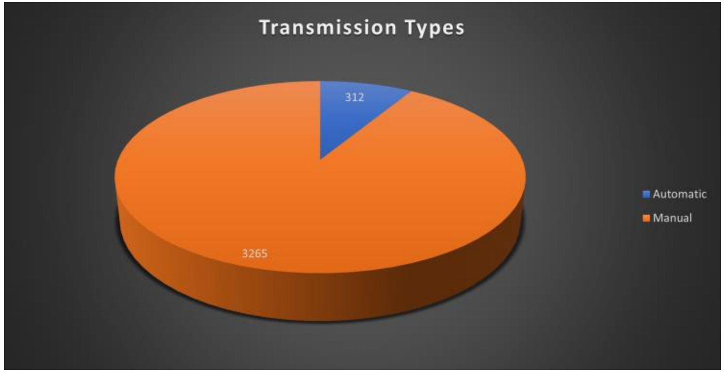
- The Dataset contains **8 rows & 4240** columns showing the information of used cars dataset.
- The Basic analysis shows that there *is no null value* present.
- The Analysis further shows that there are *763 duplicate* values.
- Analyze the count of owners by transmission type, focusing on the first owner who has maximum number of manual cars.



Data Exploration (Using Excel)

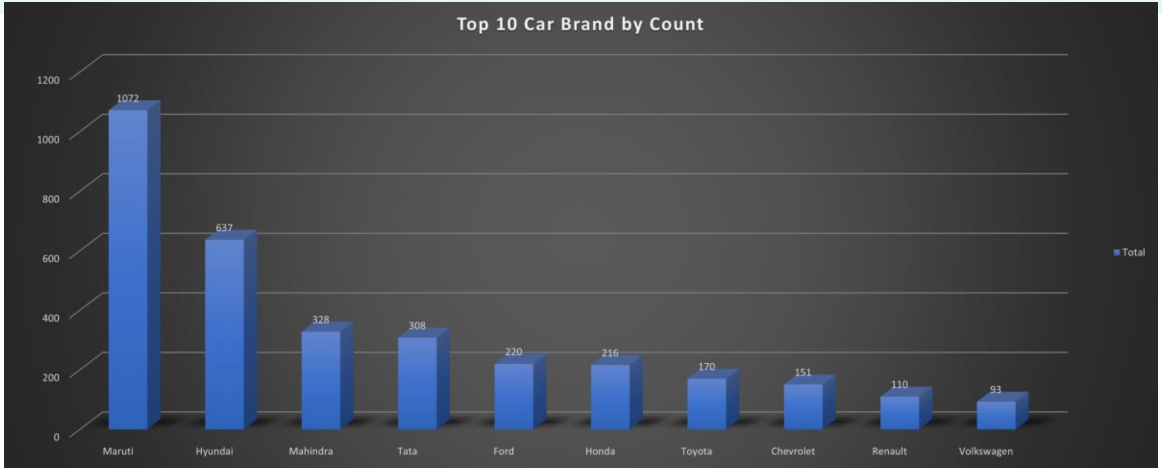






- These analysis shows distribution of different type of sellers, distribution of transmission and count of fuel types.

Data Exploration (Using Excel)



- Top 10 Car Brands by count.
- Maruti and Hyundai are the top most selling cars.

- Average selling price distribution by year.
- The graph show that the latest model car having higher price.

Data Cleaning & Analysis

Import necessary library

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import BayesianRidge
from sklearn.svm import SVR
from sklearn.metrics import *

import warnings
warnings.filterwarnings('ignore')
```

Data Exploration (Using Python)

skill academy

Loading and Understanding the dataset

```
df = pd.read_csv('CAR DETAILS.csv')
df.head()
```

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
0	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner
2	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner
3	Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner

```
# To print the sample rows of the df
df.sample(10)
```

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
4065	Hyundai i10 Magna	2011	250000	28000	Petrol	Dealer	Manual	First Owner
2131	Mahindra XUV500 W6 2WD	2014	625000	68000	Diesel	Dealer	Manual	First Owner
3696	Maruti Alto K10 VXI	2018	231999	20000	Petrol	Individual	Manual	First Owner
1138	Maruti Wagon R LXI CNG	2013	250000	71000	CNG	Individual	Manual	Second Owner
2401	Toyota Innova 2.5 E Diesel MS 7-seater	2011	665000	267000	Diesel	Individual	Manual	Second Owner
4202	Ford Figo 1.5 Sports Edition MT	2017	600000	43235	Diesel	Dealer	Manual	First Owner
1288	Mercedes-Benz E-Class E250 CDI Blue Efficiency	2012	2500000	35000	Diesel	Individual	Automatic	First Owner
3183	Maruti Alto 800 LXI	2015	265000	32933	Petrol	Dealer	Manual	First Owner
2632	Ford Fiesta Diesel Trend	2012	200000	91245	Diesel	Dealer	Manual	First Owner
1423	Hyundai i10 Magna 1.2 iTech SE	2011	235000	74500	Petrol	Individual	Manual	Second Owner

Data Exploration (Using Python)

skill academy

```
# Checking the shape of data
df.shape
```

```
(4340, 8)
```

```
# To print the concise summary of data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4340 entries, 0 to 4339
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   name           4340 non-null   object
 1   year           4340 non-null   int64
 2   selling_price  4340 non-null   int64
 3   km_driven      4340 non-null   int64
 4   fuel           4340 non-null   object
 5   seller_type    4340 non-null   object
 6   transmission   4340 non-null   object
 7   owner          4340 non-null   object
dtypes: int64(3), object(5)
memory usage: 271.4+ KB
```

Data Cleaning and Manipulation

```
# Checking for the null values
df.isnull().sum()
```

```
name             0
year             0
selling_price    0
km_driven        0
fuel             0
seller_type      0
transmission     0
owner            0
dtype: int64
```

```
# To generate descriptive statistics of the df
df.describe()
```

	year	selling_price	km_driven
count	4340.000000	4.340000e+03	4340.000000
mean	2013.090783	5.041273e+05	66215.777419
std	4.215344	5.785487e+05	46644.102194
min	1992.000000	2.000000e+04	1.000000
25%	2011.000000	2.087498e+05	35000.000000
50%	2014.000000	3.500000e+05	60000.000000
75%	2016.000000	6.000000e+05	90000.000000
max	2020.000000	8.900000e+06	806599.000000

Data Exploration (Using Python)

skill academy

```
# To find wethere there are any duplicate values in the df
df.duplicated().sum()
```

```
763
```

```
#To drop duplicate values
df.drop_duplicates(inplace = True)
```

```
df.shape
```

```
(3577, 8)
```

```
#To get information about the columns
df.columns
```

```
Index(['name', 'year', 'selling_price', 'km_driven', 'fuel', 'seller_type',
       'transmission', 'owner'],
      dtype='object')
```

```
numerical_columns = df.select_dtypes(exclude=['object']).columns
categorical_columns = df.select_dtypes(include=['object']).columns

print("Categorical Columns:")
print(categorical_columns)

print("\nNumerical Columns:")
print(numerical_columns)
```

```
Categorical Columns:
Index(['name', 'fuel', 'seller_type', 'transmission', 'owner'], dtype='object')

Numerical Columns:
Index(['year', 'selling_price', 'km_driven'], dtype='object')
```

Data Exploration (Using Python)

skill academy

Adjusting Column Names

```
#To print all the unique values in the "name" columns of the df
df["name"].unique(),df["name"].nunique()
```

```
(array(['Maruti 800 AC', 'Maruti Wagon R LXI Minor',
        'Hyundai Verna 1.6 SX', ..., 'Mahindra Verito 1.5 D6 BSIII',
        'Toyota Innova 2.5 VX (Diesel) 8 Seater BS IV',
        'Hyundai i20 Magna 1.4 CRDi'], dtype=object),
 1491)
```

```
#creating a new column in the DataFrame df named "name_2" by extracting the first word from each entry in the
df['brand'] = df['name'].apply(lambda x: pd.Series(x.split(" ")))[0]
```

```
df['model'] = df['name'].apply(lambda x: " ".join(x.split(' ')[1:]))
```

```
df.head()
```

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	brand	model
0	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner	Maruti	800 AC
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner	Maruti	Wagon R LXI Minor
2	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner	Hyundai	Verna 1.6 SX
3	Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner	Datsun	RediGO T Option
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner	Honda	Amaze VX i-DTEC

```
brand_name = df["brand"]
model_name = df["model"]
selling_price = df["selling_price"]

df.drop(
    ["name","brand", "model", "selling_price"], axis=1, inplace=True
)

df.insert(0, "brand", brand_name)
df.insert(1, "model", model_name)
df.insert(len(df.columns),
                    "selling_price", selling_price)
```

Extracting two new columns, 'brand' and 'model', from the 'name' column.

```
df.head()
```

	brand	model	year	km_driven	fuel	seller_type	transmission	owner	selling_price
0	Maruti	800 AC	2007	70000	Petrol	Individual	Manual	First Owner	60000
1	Maruti	Wagon R LXI Minor	2007	50000	Petrol	Individual	Manual	First Owner	135000
2	Hyundai	Verna 1.6 SX	2012	100000	Diesel	Individual	Manual	First Owner	600000
3	Datsun	RediGO T Option	2017	46000	Petrol	Individual	Manual	First Owner	250000
4	Honda	Amaze VX i-DTEC	2014	141000	Diesel	Individual	Manual	Second Owner	450000

Data Exploration (Using Python)

skill academy

```
df['brand'].value_counts()
```

```
brand
Maruti           1072
Hyundai           637
Mahindra          328
Tata              308
Ford              220
Honda             216
Toyota            170
Chevrolet         151
Renault           110
Volkswagen         93
Nissan             52
Skoda              49
Fiat               32
Audi               31
Datsun             29
BMW                25
Mercedes-Benz      21
Jaguar              5
Mitsubishi          5
Land                5
Volvo               4
Jeep                3
Ambassador          3
MG                  2
OpelCorsa           2
Daewoo              1
Force               1
Isuzu               1
Kia                 1
Name: count, dtype: int64
```

```
# Calculate value counts of 'brand'
brand_counts = df['brand'].value_counts()

# Create a new column for grouping brands with less than 50 counts as 'Other'
df['brand'] = df['brand'].apply(lambda x: x if brand_counts[x] >= 40 else 'Other')

df['brand'].value_counts()
```

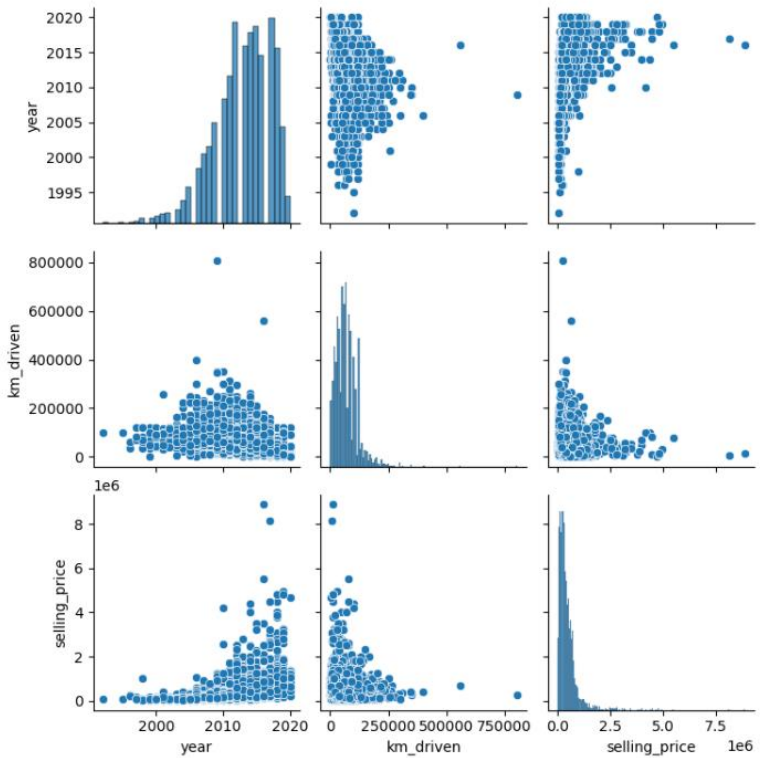
```
brand
Maruti        1072
Hyundai        637
Mahindra       328
Tata           308
Ford           220
Honda          216
Other          171
Toyota         170
Chevrolet      151
Renault        110
Volkswagen      93
Nissan          52
Skoda           49
Name: count, dtype: int64
```

In the 'brand' column, combine brands with fewer than 40 occurrences into a new category called 'Others'.

Data Exploration (Using Python)

Data Visualization

```
# To visualize the pairplot of the df
sns.pairplot(df)
plt.show()
```

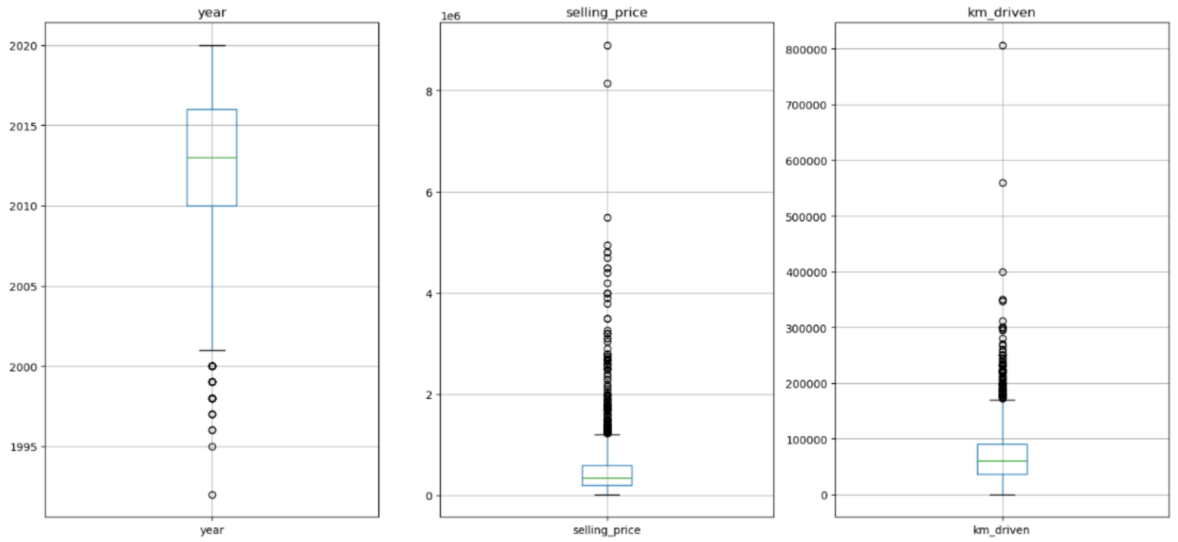


Data Exploration (Using Python)


```python
# Create subplots
fig, axes = plt.subplots(nrows=1, ncols=len(numerical_columns), figsize=(15, 7))

# Create a boxplot for each numerical column
for i, column in enumerate(numerical_columns):
    df.boxplot(column=column, ax=axes[i])
    axes[i].set_title(column)

# Adjust layout
plt.tight_layout()
plt.show()
```



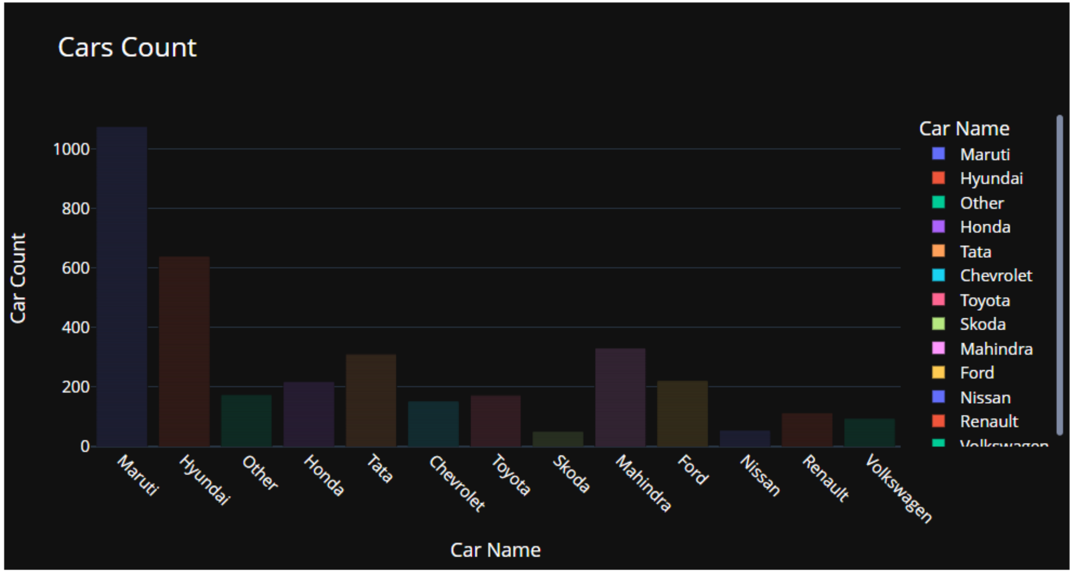
The 'year', 'selling price', and 'km driven' columns contain outliers.

- Boxplot shows that there are outliers in numerical columns. Will handle later

Data Exploration (Using Python)

skill academy

```
fig = px.bar(df,x='brand', title='Car Count', labels={"brand": "Car Name", "count": "Car Count"},
             template="plotly_dark", color="brand")
fig.update_layout(
    xaxis_title="Car Name",
    yaxis_title="Car Count",
    xaxis=dict(tickangle=45),
    font=dict(color="white", size=15),
    title=dict(text="Cars Count", font_size=25),
)
fig.show()
```

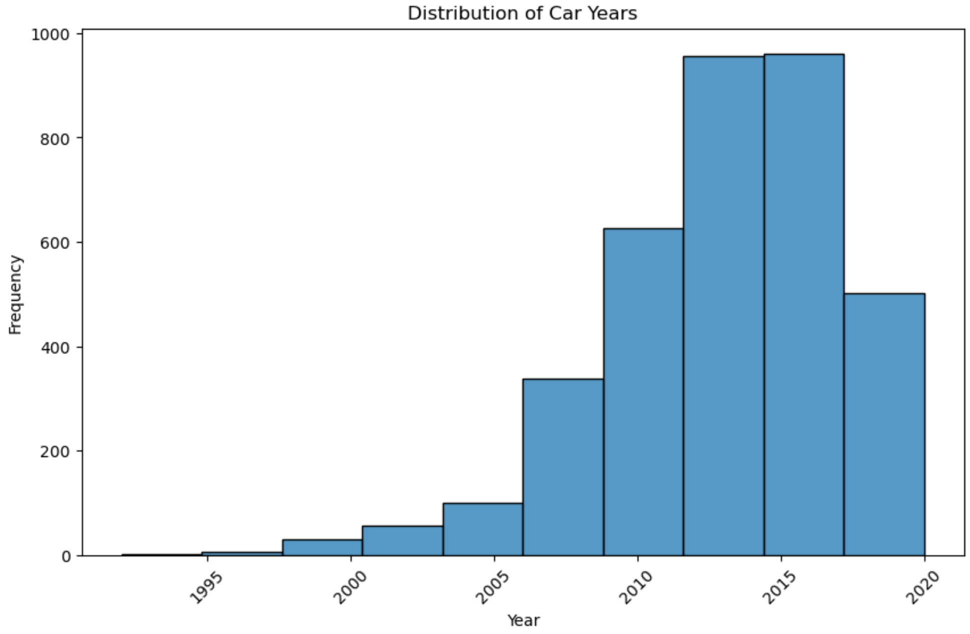


Maruti and Hyundai are the top 2 selling cars

- **Maruti Suzuki is the most popular car brand**, with almost twice the number of sales compared to the second-place brand, Hyundai.
- There is a significant drop in sales between Maruti Suzuki and Hyundai, with the following brands having considerably fewer sales.

Data Exploration (Using Python)


```
plt.figure(figsize=(10, 6))
sns.histplot(df['year'], bins=10, kde=False)
plt.title('Distribution of Car Years')
plt.xlabel('Year')
plt.ylabel('Frequency')
plt.xticks(rotation=45)
plt.show()
```



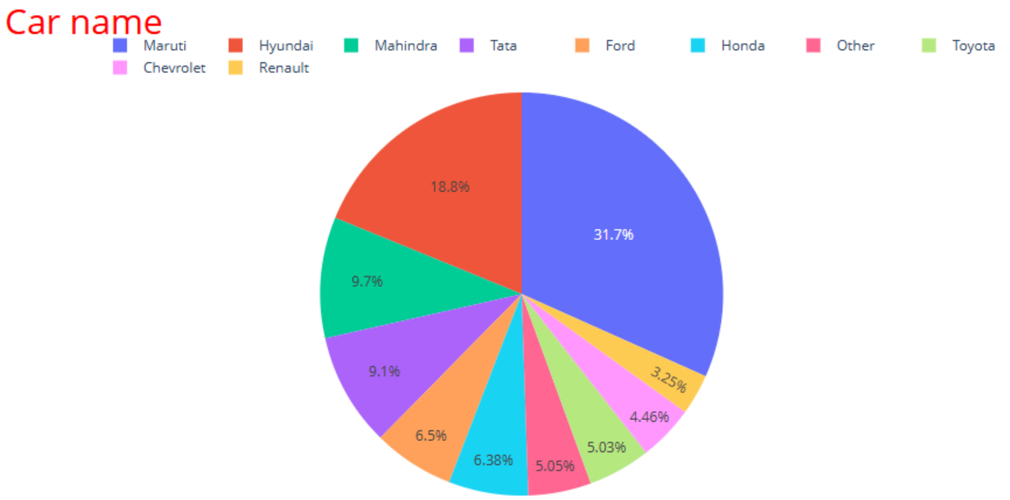
2013-2017 are the years in which highest cars were sold

- The most frequent car model year is 2015. There are more cars from 2015 than any other year shown in the data set.
- The distribution of car model years is skewed to the right. This means that there are more recent model year cars than older model year cars.
- There are a few cars from before 2000. However, the number of cars steadily increases from 2000 to 2015.

Data Exploration (Using Python)

skill academy

```python
top_cars = df["brand"].value_counts().nlargest(10)
fig = px.pie(top_cars, labels=top_cars.index, values=top_cars,
             title='Car name', names=top_cars.index,
             template="plotly_white",
             )
fig.update_layout(
    title=dict(text='Car name', font_size=30, font_color='red'),  # Set title font color to red
    legend=dict(orientation='h', yanchor='bottom', y=1.02, xanchor='right', x=1),  # Position legend
)
fig.show()
```



31.7% of cars are sold under the Maruti brand, 18.8% under the Hyundai brand, and other brands have lower percentages.

- The pie chart shows the distribution of car sales for different car brands. The largest slice of the pie chart is Maruti, at 31.7%. This suggests that Maruti is the most popular car brand out of the ones listed. Other large slices of the pie chart include Hyundai (19.3%) and Mahindra (10.3%). Brands such as Renault and Chevrolet have a much smaller slice of the pie chart (1.2% and 0.9% respectively).
- Here are some other insights you can draw from the pie chart:
  - The top 5 car brands (Maruti, Hyundai, Mahindra, Tata, and Toyota) account for over 63% of the car sales.
  - There are a significant number of other car brands that are not listed in the chart but that collectively account for 13.8% of the sales.
- Overall, the pie chart suggests that the car sales market is dominated by a few major brands.

Data Exploration (Using Python)

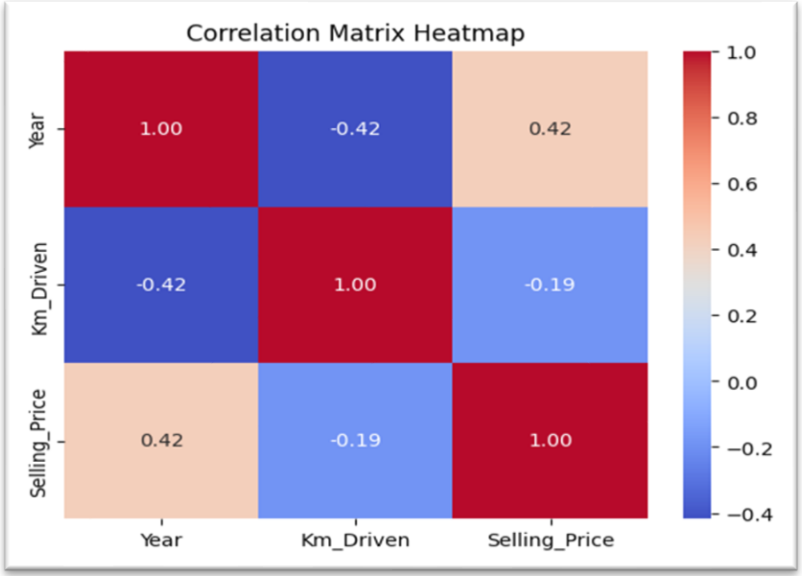
```python
numerical_columns = df.select_dtypes(include='number')
corr = numerical_columns.corr()
corr
```

	year	km_driven	selling_price
year	1.00000	-0.417490	0.424260
km_driven	-0.41749	1.000000	-0.187359
selling_price	0.42426	-0.187359	1.000000

```python
plt.figure(figsize=(15,8))
sns.heatmap(corr, annot = True, fmt='.2f', cmap='coolwarm', linewidths=0.5, linecolor='black')
plt.title('Customized Correlation Heatmap', fontsize=20, pad=20)
plt.xlabel('Features', fontsize=15)
plt.ylabel('Features', fontsize=15)
plt.show()
```

The correlation matrix heatmap you sent shows that there is a negative correlation between the selling price of a car and the number of kilometers driven (Km_Driven). A negative correlation means that two variables move in opposite directions. In this case, as the number of kilometers driven increases, the selling price of the car decreases. This makes sense because cars with higher mileage are typically less valuable than cars with lower mileage.

- Here are some other factors that may influence the selling price of a car:
  - **Make and model:** Different makes and models of cars depreciate at different rates.
  - **Age of the car:** As a car gets older, it is typically worth less.
  - **Overall condition of the car:** Cars that are in good condition will typically sell for more than cars that are in poor condition
  - **Features:** Cars with more features will typically sell for more than cars with fewer features.



or zero owners.

Data Exploration (Using Python)

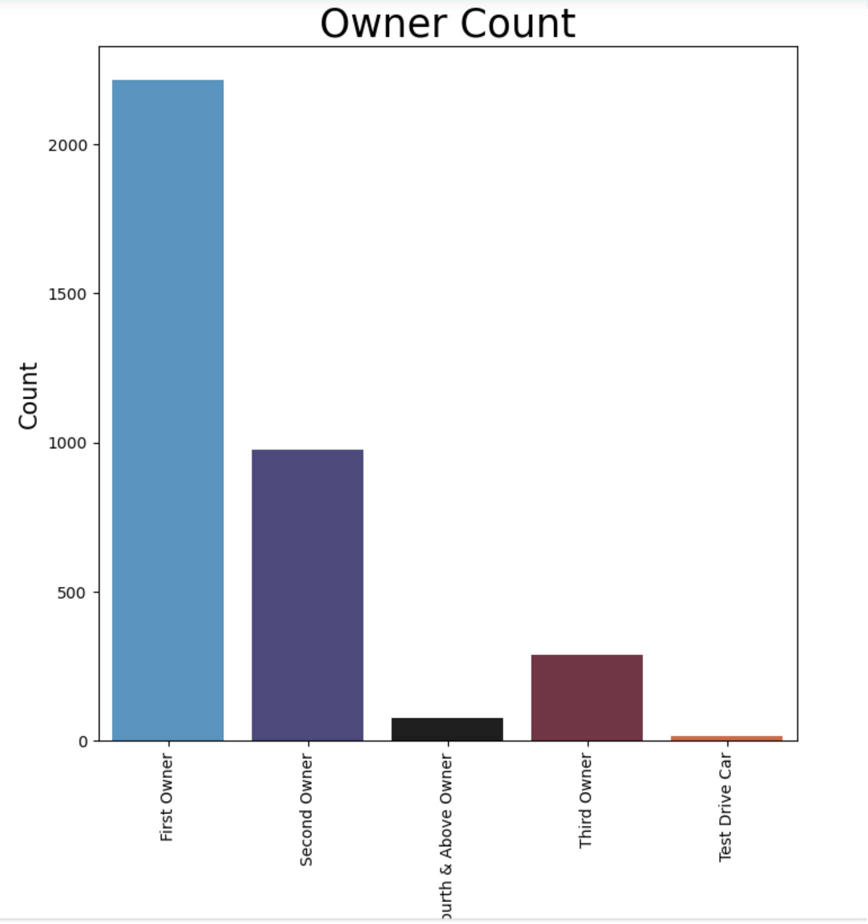
skill academy

```
df.owner.value_counts()
```

```
owner
First Owner             2218
Second Owner             978
Third Owner              289
Fourth & Above Owner      75
Test Drive Car            17
Name: count, dtype: int64
```

```
plt.figure(figsize=(8,8))
sns.countplot(data=df,x="owner",palette="icefire")
plt.xticks(rotation=90)
plt.xlabel("Owner type",fontsize=15,color="black")
plt.ylabel("Count",fontsize=15,color="black")
plt.title("Owner Count",fontsize=25,color="black")
plt.show()
```

- Overall, the countplot suggests that most of the cars in the dataset have had only one or zero owners.pen_spark



Data Exploration (Using Python)

```
df.columns
```

```
Index(['brand', 'model', 'year', 'km_driven', 'fuel', 'seller_type',
       'transmission', 'owner', 'selling_price'],
      dtype='object')
```

```
df['transmission'].value_counts()
```

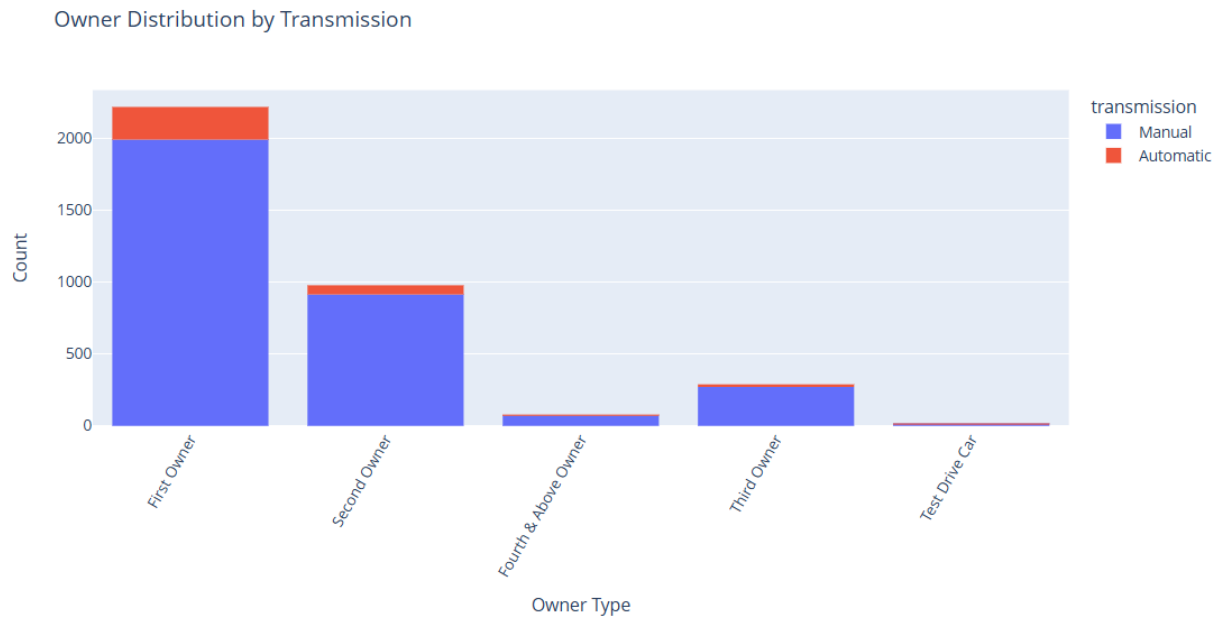
```
transmission
Manual       3265
Automatic     312
Name: count, dtype: int64
```

```
fig = px.histogram(df, x="year", color="transmission", title="Year Distribution by Transmission",
                   labels={"year": "Year", "count": "Count"}, template="plotly")
fig.update_layout(
    xaxis_title="Year",
    yaxis_title="Count",
    title=dict(text="Year Distribution by Transmission"),
    xaxis=dict(tickangle=-45),
)
fig.show()
```

Data Exploration (Using Python)

skill academy

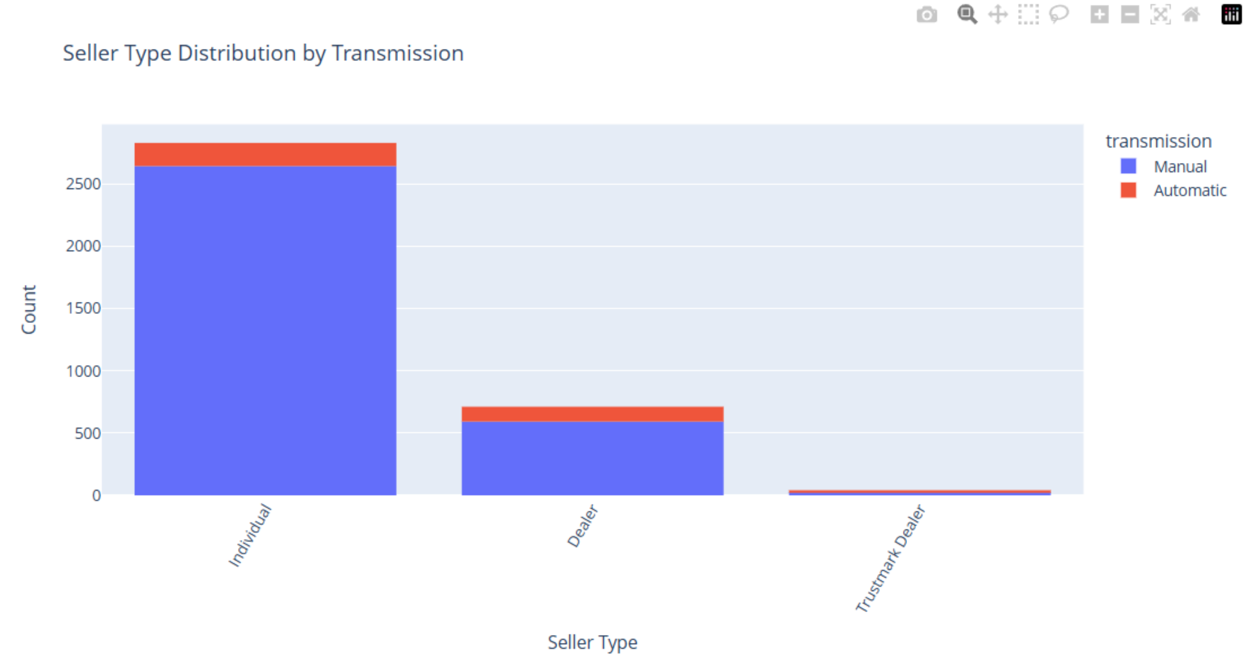
```python
fig = px.histogram(df, x="owner", color="transmission", title="Year Distribution by Transmission",
                   labels={"owner": "Owner Type", "count": "Count"}, template="plotly")
fig.update_layout(
    xaxis_title="Owner Type",
    yaxis_title="Count",
    title=dict(text="Owner Distribution by Transmission"),
    xaxis=dict(tickangle=-60),
)
fig.show()
```



- There are 227 first-owner vehicles with automatic transmissions.
- There are 1,991 first-owner vehicles with manual transmissions.
- Among second-owner vehicles:
    - 64 have automatic transmissions.
    - 914 have manual transmissions.
- Among third-owner vehicles:
    - 2 have automatic transmissions.
    - 271 have manual transmissions.
- Test drive cars are negligible in number.

Data Exploration (Using Python)

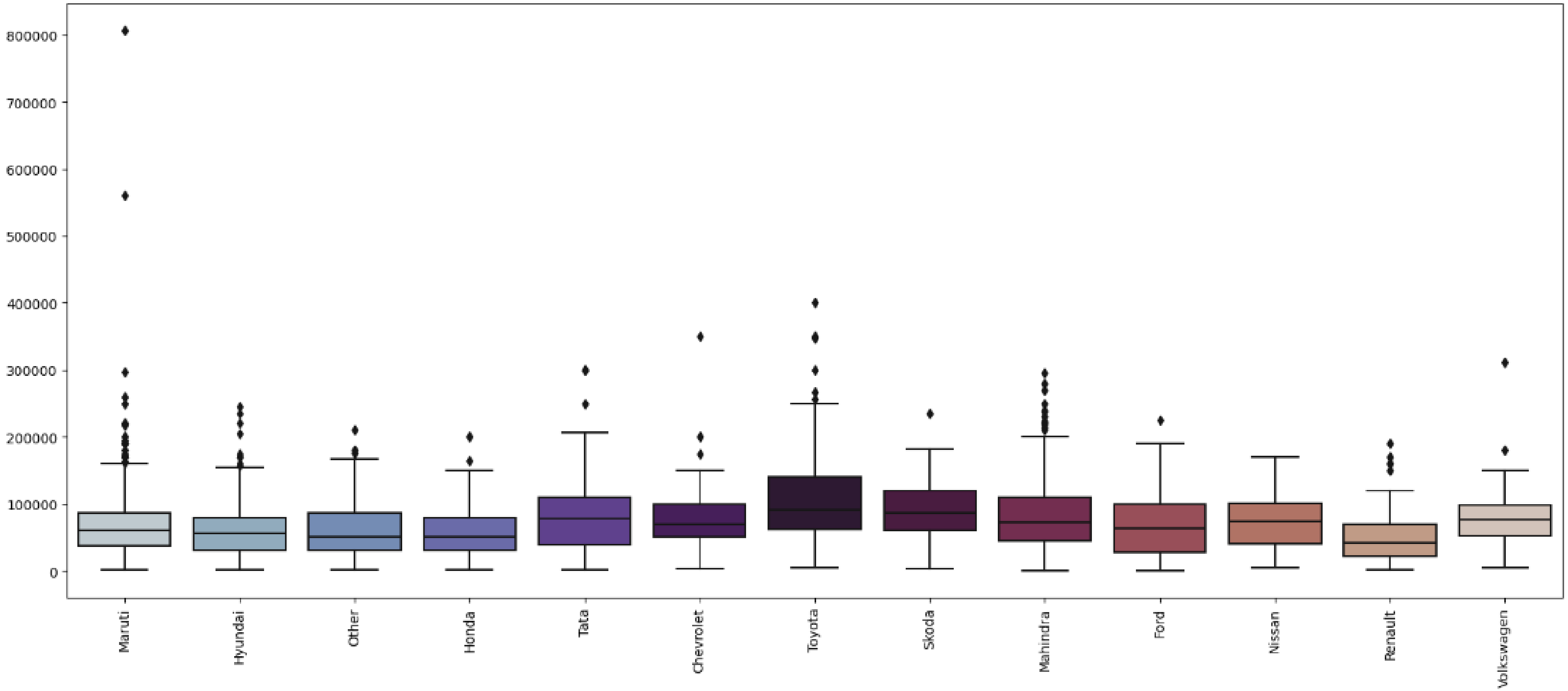
```
fig = px.histogram(df, x="seller_type", color="transmission", title="Seller Type Distribution by Transmission",
                   labels={"seller_type": "Seller Type", "count": "Count"}, template="plotly")
fig.update_layout(
    xaxis_title="Seller Type",
    yaxis_title="Count",
    title=dict(text="Seller Type Distribution by Transmission"),
    xaxis=dict(tickangle=-60),
)
fig.show()
```



- A total of 2,832 cars were purchased directly by individuals, comprising 2,646 manual and 186 automatic vehicles.
- A total of 712 cars were purchased by individuals through dealers, comprising 593 manual and 119 automatic vehicles.
- A total of 33 cars were purchased by individuals through trustmark dealers, comprising 26 manual and 7 automatic vehicles.

Data Exploration (Using Python)

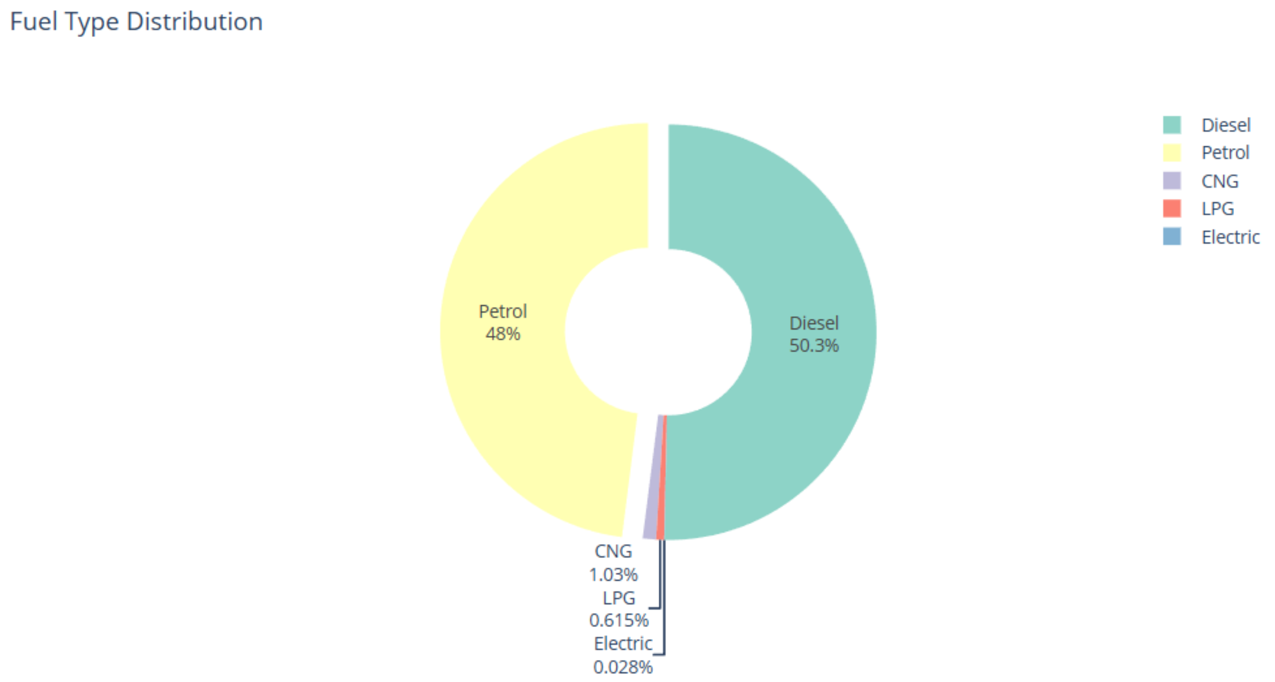
```
f, ax = plt.subplots(figsize=(20,8))
sns.boxplot(x=df["brand"].values, y = df["km_driven"].values,palette="twilight",ax=ax)
plt.xticks(rotation=90)
plt.show()
```



Data Exploration (Using Python)

skill academy

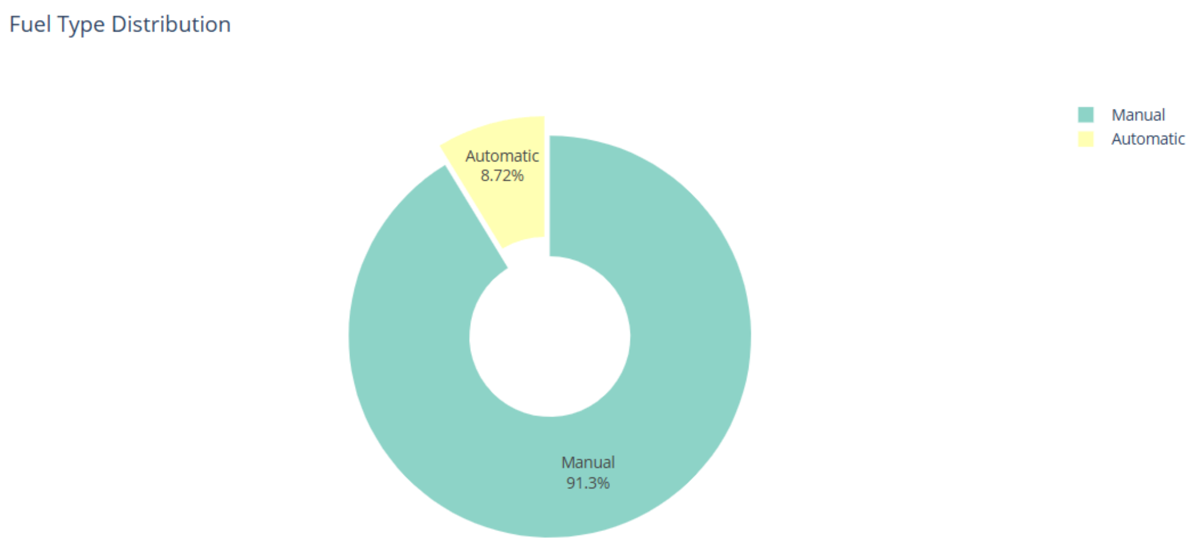
```
fig = px.pie(df, names='fuel', hole=0.4, title='Fuel Type Distribution',
             color_discrete_sequence=px.colors.qualitative.Set3)
fig.update_traces(textinfo='percent+label', pull=[0.1, 0])
fig.show()
```



- 50.3% of vehicles are diesel vehicles.
- 48% of vehicles are petrol vehicles.
- The remaining vehicles are classified under other categories.

Data Exploration (Using Python)

```python
fig = px.pie(df, names='transmission', hole=0.4, title='Fuel Type Distribution',
             color_discrete_sequence=px.colors.qualitative.Set3)
fig.update_traces(textinfo='percent+label', pull=[0.1, 0])
fig.show()
```



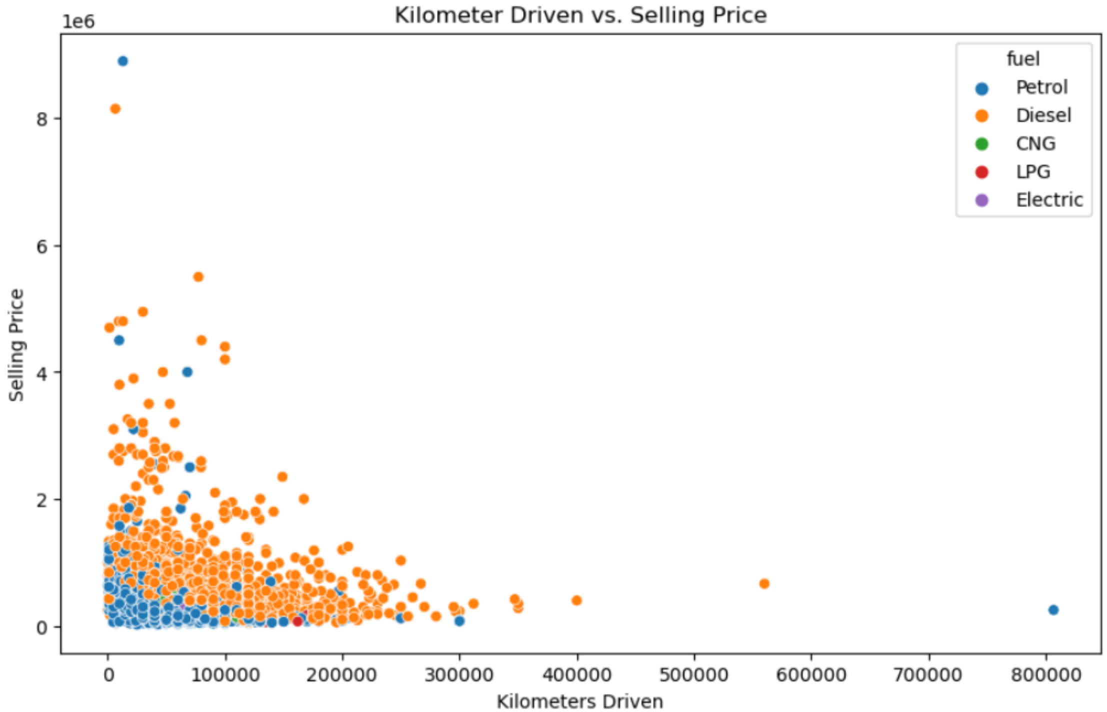
91.3% of cars have manual transmission.

Only 8.72% of cars have automatic transmission

Data Exploration (Using Python)

```python
# Scatter Plot of km_driven vs. Selling Price
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='km_driven', y='selling_price', hue='fuel')
plt.title('Kilometer Driven vs. Selling Price')
plt.xlabel('Kilometers Driven')
plt.ylabel('Selling Price')
plt.show()

```



- **There is a negative correlation between Kilometer Driven and selling price:** As the mileage driven increases, the selling price tends to decrease. This suggests that cars with higher mileage tend to sell for lower prices.

Data Exploration (Using Python)

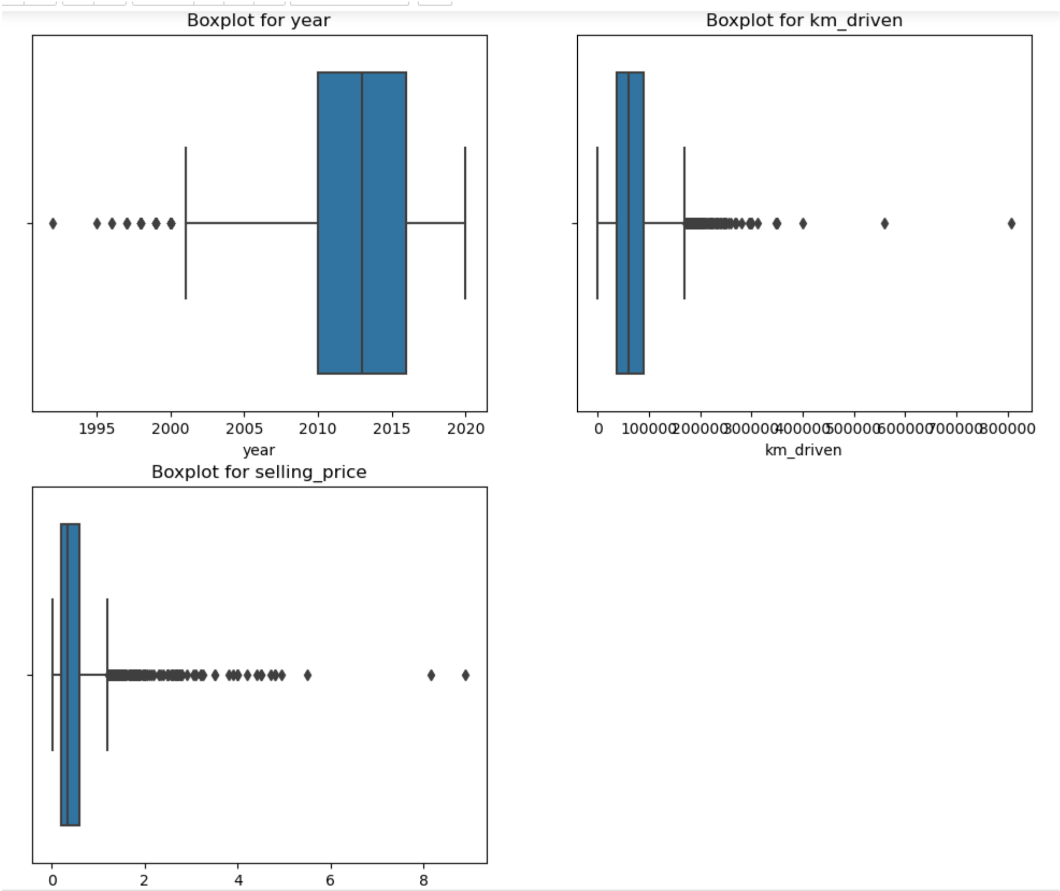
Outliers Detections

```python
num_cols = df.dtypes[df.dtypes!='object'].index
num_cols
```

```
Index(['year', 'km_driven', 'selling_price'], dtype='object')
```

```python
plt.figure(figsize=(12,10))
for i in range(len(num_cols)):
    plt.subplot(2,2,i+1)
    sns.boxplot(x = df[num_cols[i]])
    plt.title(f'Boxplot for {num_cols[i]}')

plt.show()
```



- Outliers Detected in Year, Kilometer driven and selling price column

Data Exploration (Using Python)

skill academy

Outlier Treatment - Cap

```
df1[num_cols].describe(percentiles=[0.01,0.05,0.25,0.75,0.95,0.97,0.98,0.99]).T
```

	count	mean	std	min	1%	5%	25%	50%	75%	95%	97%	98%	99%	max
year	3577.0	2012.962538	4.251759	1992.0	2000.00	2005.0	2010.0	2013.0	2016.0	2019.0	2019.0	2019.0	2020.0	2020.0
1_driven	3577.0	69250.545709	47579.940016	1.0	1744.08	10000.0	36000.0	60000.0	90000.0	149534.8	170000.0	193440.0	223158.4	806599.0
ıg_price	3577.0	473912.542074	509301.809816	20000.0	51786.64	80000.0	200000.0	350000.0	600000.0	1200000.0	1497200.0	1800000.0	2675000.0	8900000.0

```
print(df1[df1['year']<2001.00].shape)
print(df1[df1['selling_price']>1200000.0].shape)
print(df1[df1['km_driven']>149534.8].shape)
```

```
(37, 9)
(170, 9)
(179, 9)
```

```
df1['year'] = np.where(df1['year']<2001.00 , 2001.00,df1['year'])
df1['selling_price'] = np.where(df1['selling_price']>1200000.0 , 1200000.0,df1['selling_price'])
df1['km_driven'] = np.where(df1['km_driven']>149534.8 , 149534.8,df1['km_driven'])
```

To handle outliers in the 'year' column, we will cap the values, setting a lower bound of 2001.

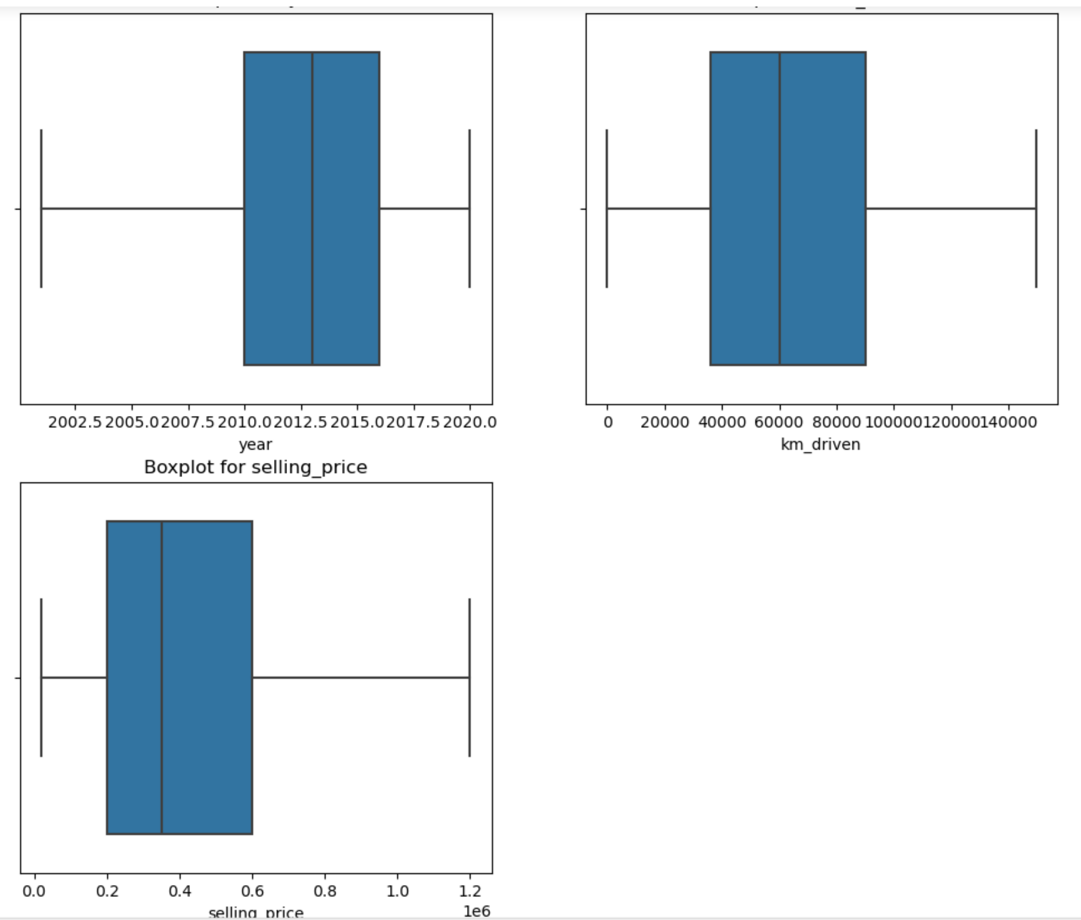
For the 'Selling Price' column, we will cap outliers at the 95th percentile.

For the 'KM Driven' column, we will cap outliers at the 95th percentile.

Data Exploration (Using Python)


```python
plt.figure(figsize=(12,10))
for i in range(len(num_cols)):
    plt.subplot(2,2,i+1)
    sns.boxplot(x = df[num_cols[i]])
    plt.title(f'Boxplot for {num_cols[i]}')

plt.show()
```



- BoxPlot after handling outliers.

Data Exploration (Using Python)

```
df1.head()
```

	brand	model	year	km_driven	fuel	seller_type	transmission	owner	selling_price
0	Maruti	800 AC	2007.0	70000.0	Petrol	Individual	Manual	First Owner	60000.0
1	Maruti	Wagon R LXI Minor	2007.0	50000.0	Petrol	Individual	Manual	First Owner	135000.0
2	Hyundai	Verna 1.6 SX	2012.0	100000.0	Diesel	Individual	Manual	First Owner	600000.0
3	Other	RediGO T Option	2017.0	46000.0	Petrol	Individual	Manual	First Owner	250000.0
4	Honda	Amaze VX i-DTEC	2014.0	141000.0	Diesel	Individual	Manual	Second Owner	450000.0

```
df1.drop('model', axis=1, inplace=True)
```

```
df1.to_csv("cleaned data.csv",index=False)
```

```
pd.read_csv('cleaned data.csv')
```

	brand	year	km_driven	fuel	seller_type	transmission	owner	selling_price
0	Maruti	2007.0	70000.0	Petrol	Individual	Manual	First Owner	60000.0
1	Maruti	2007.0	50000.0	Petrol	Individual	Manual	First Owner	135000.0
2	Hyundai	2012.0	100000.0	Diesel	Individual	Manual	First Owner	600000.0
3	Other	2017.0	46000.0	Petrol	Individual	Manual	First Owner	250000.0
4	Honda	2014.0	141000.0	Diesel	Individual	Manual	Second Owner	450000.0
...	...	...	...	...	...	...	...	...
3572	Hyundai	2014.0	80000.0	Diesel	Individual	Manual	Second Owner	409999.0
3573	Hyundai	2014.0	80000.0	Diesel	Individual	Manual	Second Owner	409999.0
3574	Maruti	2009.0	83000.0	Petrol	Individual	Manual	Second Owner	110000.0
3575	Hyundai	2016.0	90000.0	Diesel	Individual	Manual	First Owner	865000.0
3576	Renault	2016.0	40000.0	Petrol	Individual	Manual	First Owner	225000.0

3577 rows × 8 columns

Data Cleaning part is done and we saved our cleaned data. Now next we will move towards model building part.

Machine Learning Model Buliding

Model Building,Training & Testing

Import necessary library

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import BayesianRidge
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.svm import SVR
from sklearn.metrics import *

import warnings
warnings.filterwarnings('ignore')
```

Loading and Understanding the dataset

```
df = pd.read_csv('cleaned data.csv')
df.head()
```

	brand	year	km_driven	fuel	seller_type	transmission	owner	selling_price
0	Maruti	2007.0	70000.0	Petrol	Individual	Manual	First Owner	60000.0
1	Maruti	2007.0	50000.0	Petrol	Individual	Manual	First Owner	135000.0
2	Hyundai	2012.0	100000.0	Diesel	Individual	Manual	First Owner	600000.0
3	Other	2017.0	46000.0	Petrol	Individual	Manual	First Owner	250000.0
4	Honda	2014.0	141000.0	Diesel	Individual	Manual	Second Owner	450000.0

Encode the Catgeorical Features

```
cat_cols = df.dtypes[df.dtypes=='object'].index
print(cat_cols)
```

```
Index(['brand', 'fuel', 'seller_type', 'transmission', 'owner'], dtype='object')
```

```
for i in cat_cols:
    print(i,df[i].unique(),df[i].nunique())
    print()
```

```
brand ['Maruti' 'Hyundai' 'Other' 'Honda' 'Tata' 'Chevrolet' 'Toyota' 'Skoda'
 'Mahindra' 'Ford' 'Nissan' 'Renault' 'Volkswagen'] 13

fuel ['Petrol' 'Diesel' 'CNG' 'LPG' 'Electric'] 5

seller_type ['Individual' 'Dealer' 'Trustmark Dealer'] 3

transmission ['Manual' 'Automatic'] 2

owner ['First Owner' 'Second Owner' 'Fourth & Above Owner' 'Third Owner'
 'Test Drive Car'] 5
```


```
cat_cols = df.dtypes[df.dtypes=='object'].index
print(cat_cols)
```

```
Index(['brand', 'fuel', 'seller_type', 'transmission', 'owner'], dtype='object')
```


```
df.dtypes
```

```
brand             object
year             float64
km_driven        float64
fuel              object
seller_type       object
transmission      object
owner             object
selling_price    float64
dtype: object
```

```
df.head()
```

	brand	year	km_driven	fuel	seller_type	transmission	owner	selling_price
0	Maruti	2007.0	70000.0	Petrol	Individual	Manual	First Owner	60000.0

Machine Learning Model Buliding

```
df.head()
```

	brand	year	km_driven	fuel	seller_type	transmission	owner	selling_price
0	Maruti	2007.0	70000.0	Petrol	Individual	Manual	First Owner	60000.0
1	Maruti	2007.0	50000.0	Petrol	Individual	Manual	First Owner	135000.0
2	Hyundai	2012.0	100000.0	Diesel	Individual	Manual	First Owner	600000.0
3	Other	2017.0	46000.0	Petrol	Individual	Manual	First Owner	250000.0
4	Honda	2014.0	141000.0	Diesel	Individual	Manual	Second Owner	450000.0

```
lb = LabelEncoder()
```

```
for col in cat_cols:
    df[col] = lb.fit_transform(df[col])
```

```
df.head()
```

	brand	year	km_driven	fuel	seller_type	transmission	owner	selling_price
0	5	2007.0	70000.0	4	1	1	0	60000.0
1	5	2007.0	50000.0	4	1	1	0	135000.0
2	3	2012.0	100000.0	1	1	1	0	600000.0
3	7	2017.0	46000.0	4	1	1	0	250000.0
4	2	2014.0	141000.0	1	1	1	2	450000.0

Select x and y

```
x = df.drop('selling_price',axis=1)
y = df['selling_price']
print(x.shape)
print(y.shape)
```

```
(3577, 7)
(3577,)
```

Split the data into train and test

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.30,
                                                 random_state=42)

print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(2503, 7)
(1074, 7)
(2503,)
(1074,)
```

```
x_train.head()
```

	brand	year	km_driven	fuel	seller_type	transmission	owner
1078	5	2012.0	90000.0	4	1	1	4
844	5	2019.0	5000.0	4	1	1	0
1656	11	2013.0	80000.0	1	1	1	4
2887	4	2013.0	149534.8	1	1	1	0
1731	5	2003.0	35000.0	4	1	1	0

```
x_test.head()
```

	brand	year	km_driven	fuel	seller_type	transmission	owner
907	5	2018.0	20000.0	4	1	1	0
2684	5	2017.0	39000.0	4	0	1	0
1373	11	2016.0	146000.0	1	0	1	0
538	7	2018.0	10000.0	1	0	0	0
1454	2	2007.0	70000.0	4	1	1	2

Machine Learning Model Buliding

skill academy

Create Function to Evaluate the Model

```python
def eval_model(model, mname):
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)
    train_r2 = model.score(x_train, y_train)
    test_r2 = model.score(x_test, y_test)
    test_mae = mean_absolute_error(y_test, y_pred)
    test_mse = mean_squared_error(y_test, y_pred)
    test_rmse = np.sqrt(test_mse)
    res_df = pd.DataFrame({
        'Train_R2': train_r2,
        'Test_R2': test_r2,
        'Test_MAE': test_mae,
        'Test_MSE': test_mse,
        'Test_RMSE': test_rmse
    }, index=[mname])
    return res_df
```

Build ML models

1) Linear Regression

```python
lr1 = LinearRegression()

lr1_res = eval_model(lr1,'LinearRegressor')
lr1_res
```

	Train_R2	Test_R2	Test_MAE	Test_MSE	Test_RMSE
LinearRegressor	0.555788	0.579887	153230.769931	3.944687e+10	198612.357479

2) Ridge Reg

```python
ridge = Ridge()

ridge_res = eval_model(ridge,'ridge')
ridge_res
```

	Train_R2	Test_R2	Test_MAE	Test_MSE	Test_RMSE
ridge	0.555785	0.579828	153256.885429	3.945243e+10	198626.367221

3) Lasso Reg

```python
lasso = Lasso()

lasso_res = eval_model(lasso,'lasso')
lasso_res
```

	Train_R2	Test_R2	Test_MAE	Test_MSE	Test_RMSE
lasso	0.555788	0.579887	153230.914584	3.944690e+10	198612.429698

4) Decision Tree Reg

```python
dt1 = DecisionTreeRegressor(max_depth=8,min_samples_split=12)  # rand

dt1_res = eval_model(dt1,'DecisionTreeRegressor')
dt1_res
```

	Train_R2	Test_R2	Test_MAE	Test_MSE	Test_RMSE
DecisionTreeRegressor	0.77143	0.678067	120035.656589	3.022823e+10	173862.676653

Machine Learning Model Buliding

skill academy

5) Random Forest Regression

```
rf1 = RandomForestRegressor(n_estimators=80,max_depth=8,
                            min_samples_split=12)

rf1_res = eval_model(rf1,'RandomForestRegressor')
rf1_res
```

	Train_R2	Test_R2	Test_MAE	Test_MSE	Test_RMSE
RandomForestRegressor	0.786994	0.73476	113046.034295	2.490499e+10	157813.155149

6) Gradient Boosting Regressor

```
gbr = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)

gbr_res = eval_model(gbr,'GradientBoostingRegressor')
gbr_res
```

	Train_R2	Test_R2	Test_MAE	Test_MSE	Test_RMSE
GradientBoostingRegressor	0.756626	0.73666	115429.761771	2.472651e+10	157246.649185

7) KNeighborsRegressor

```
knn = KNeighborsRegressor(n_neighbors=5)
knn_res = eval_model(knn,'KNeighborsRegressor')
knn_res
```

	Train_R2	Test_R2	Test_MAE	Test_MSE	Test_RMSE
KNeighborsRegressor	0.534877	0.327522	179414.437803	6.314297e+10	251282.652291

8) AdaBoostRegressor

```
base_regressor = DecisionTreeRegressor(max_depth=4)

# Initialize the AdaBoostRegressor
ada_regressor = AdaBoostRegressor(estimator=base_regressor, n_estimators=100, random_state=42)
ada_res = eval_model(ada_regressor,'AdaBoostRegressor')
ada_res
```

	Train_R2	Test_R2	Test_MAE	Test_MSE	Test_RMSE
AdaBoostRegressor	0.546912	0.553408	172073.81943	4.193312e+10	204775.79026

9) BaggingRegressor

```
base_regressor = DecisionTreeRegressor()

bagging_regressor = BaggingRegressor(estimator=base_regressor, n_estimators=100, random_state=42)
bagging_res = eval_model(bagging_regressor,'BaggingRegressor')
bagging_res
```

	Train_R2	Test_R2	Test_MAE	Test_MSE	Test_RMSE
BaggingRegressor	0.939412	0.714573	116540.092679	2.680044e+10	163708.404279

Machine Learning Model Buliding

```
all_res = pd.concat([lr1_res, ridge_res, lasso_res, dt1_res, rf1_res, gbr_res, knn_res, ada_res, bagging_res])
all_res
```

	Train_R2	Test_R2	Test_MAE	Test_MSE	Test_RMSE
LinearRegressor	0.555788	0.579887	153230.769931	3.944687e+10	198612.357479
ridge	0.555785	0.579828	153256.885429	3.945243e+10	198626.367221
lasso	0.555788	0.579887	153230.914584	3.944690e+10	198612.429698
DecisionTreeRegressor	0.771430	0.678067	120035.656589	3.022823e+10	173862.676653
RandomForestRegressor	0.786994	0.734760	113046.034295	2.490499e+10	157813.155149
GradientBoostingRegressor	0.756626	0.736660	115429.761771	2.472651e+10	157246.649185
KNeighborsRegressor	0.534877	0.327522	179414.437803	6.314297e+10	251282.652291
AdaBoostRegressor	0.546912	0.553408	172073.819430	4.193312e+10	204775.790260
BaggingRegressor	0.939412	0.714573	116540.092679	2.680044e+10	163708.404279

The best performing model is RandomForestRegressor

Saving the Model

```
import pickle
```

```
# pickle.dump(gbr,open('GradientBoosting.pkl','wb'))
pickle.dump(rf1,open('RandomForest.pkl','wb'))
```

skill academy

Loading the saved model

```
load_model = pickle.load(
    open(f"../models/{f_modelname}.pkl", "rb"))  # rb = read binary
print(f"Name of loaded Model : {f_modelname}")
load_model
```

```
with open('RandomForest.pkl', 'rb') as file:
    load_model = pickle.load(file)
```

Take the original data set and make another dataset by randomly picking 20 data points from the CAR DETAILS dataset and apply the saved model on the same Dataset and test the model.

Generating sample data from cleaned df to test on the trained model.

```
random_datasample = df.sample(20)
random_datasample_df = random_datasample.drop("selling_price", axis=1)
print(random_datasample_df.shape)
random_datasample_df.head()
```

```
(20, 7)
```

	brand	year	km_driven	fuel	seller_type	transmission	owner
1685	5	2006.0	40000.0	4	1	1	2
2362	2	2015.0	50000.0	4	1	1	0
1251	10	2014.0	90000.0	1	1	1	2
2909	5	2013.0	63000.0	1	1	1	0
2019	4	2013.0	110000.0	1	1	1	2

Machine Learning Model Buliding

Resetting the index as the randomly generated data has no continuos index (wil delete later,just for understanding)

```
random_datasample_df.reset_index()
```

	index	brand	year	km_driven	fuel	seller_type	transmission	owner
0	1685	5	2006.0	40000.0	4	1	1	2
1	2362	2	2015.0	50000.0	4	1	1	0
2	1251	10	2014.0	90000.0	1	1	1	2
3	2909	5	2013.0	63000.0	1	1	1	0
4	2019	4	2013.0	110000.0	1	1	1	2
5	406	3	2017.0	80577.0	4	1	1	0
6	1271	9	2011.0	149534.8	1	1	0	0
7	291	3	2017.0	50000.0	4	1	1	0
8	2047	5	2017.0	93000.0	4	0	1	0
9	1739	5	2014.0	15000.0	4	1	1	2
10	2308	4	2013.0	110000.0	1	1	1	0
11	2612	0	2014.0	60000.0	1	1	1	0
12	950	8	2018.0	10000.0	4	0	1	0
13	1596	3	2013.0	59213.0	4	1	1	0
14	2453	2	2012.0	47000.0	4	0	1	2
15	850	5	2019.0	5000.0	4	1	1	0
16	473	2	2014.0	44000.0	1	1	1	0
17	2190	5	2017.0	70000.0	4	1	1	2
18	2736	3	2018.0	40000.0	1	1	0	0
19	1778	5	2007.0	100000.0	4	1	1	2

```
random_datasample_df.to_csv("20_random_sample.csv", index=False)
```

Machine Learning Model Buliding

Loading the sample data and checking basics

```
testsample_df = pd.read_csv("20_random_sample.csv")
print(
    "Shape of loaded sample dataframe:",
    testsample_df.shape,
    "\n\nSample Dataframe contents",
)
testsample_df
```

```
Shape of loaded sample dataframe: (20, 7)

Sample Dataframe contents
```

	brand	year	km_driven	fuel	seller_type	transmission	owner
0	5	2006.0	40000.0	4	1	1	2
1	2	2015.0	50000.0	4	1	1	0
2	10	2014.0	90000.0	1	1	1	2
3	5	2013.0	63000.0	1	1	1	0
4	4	2013.0	110000.0	1	1	1	2
5	3	2017.0	80577.0	4	1	1	0
6	9	2011.0	149534.8	1	1	0	0
7	3	2017.0	50000.0	4	1	1	0
8	5	2017.0	93000.0	4	0	1	0
9	5	2014.0	15000.0	4	1	1	2
10	4	2013.0	110000.0	1	1	1	0
11	0	2014.0	60000.0	1	1	1	0
12	8	2018.0	10000.0	4	0	1	0
13	3	2013.0	59213.0	4	1	1	0
14	2	2012.0	47000.0	4	0	1	2
15	5	2019.0	5000.0	4	1	1	0
16	2	2014.0	44000.0	1	1	1	0
17	5	2017.0	70000.0	4	1	1	2
18	3	2018.0	40000.0	1	1	0	0
19	5	2007.0	100000.0	4	1	1	2

Making Predictions on sample dataset against the trained model ¶

```
# making prediction on random data
predicted_data = load_model.predict(testsample_df)
print(f"The predicted data from RandomForest model:\n", predicted_data)
```

```
The predicted data from RandomForest model:
 [ 107554.07843006  403374.15036932  312467.40654653  417839.41918645
  418826.68947375  485403.80112932  479534.26129732  487344.9011473
  389905.1285884   344193.27797721  418826.68947375  394500.04992677
  457963.56529579  312416.41447158  276848.52307676  481591.32336767
  497399.78796213  369684.64276376 1145450.99419     120371.34124774]
```

Endnotes

skill academy

About Me :

- Name – Ravi Pandey
- Course – Data Science With Python career Program (ChatGPT Included)
- LinkedIn - Ravi Pandey
- GitHub - Ravi
- Email - ravipandey4568@gmail.com

Reference Links:-

- GitHub Repo Link
- Streamlit App Weblink

END