Data Science with Python Career Program - Capstone Project

- By **Shubham Sarraf**

Agenda

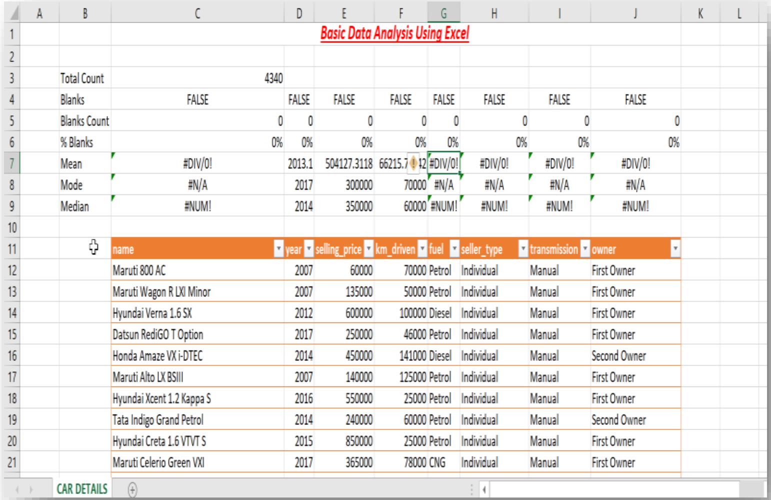
skill academy

- Data Exploration (Using Excel and Python)
- Data insights (Using Excel and 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

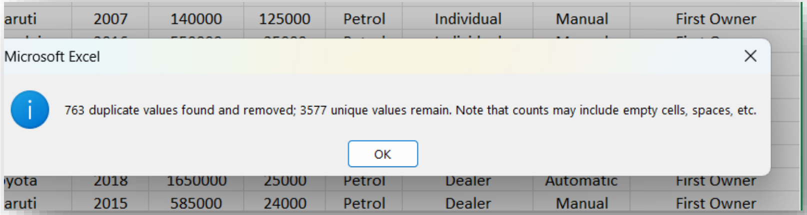
Data Exploration (Using Excel)

skill academy

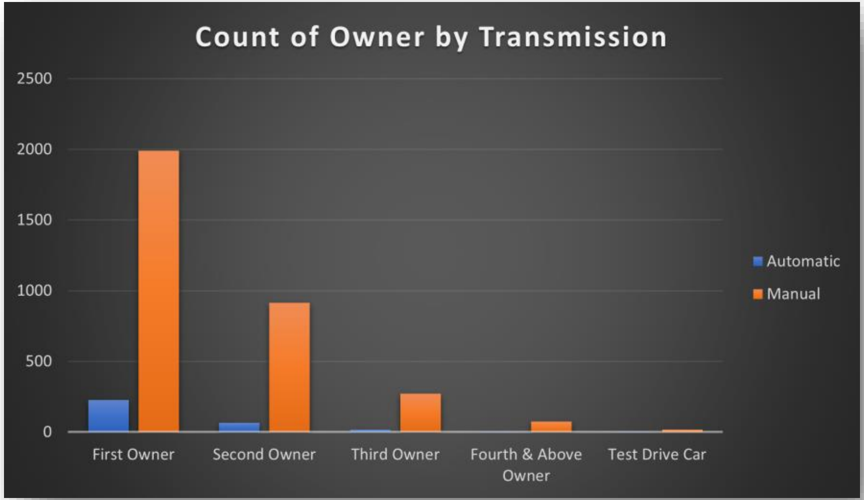
Basic Analysis Using Excel Sheet



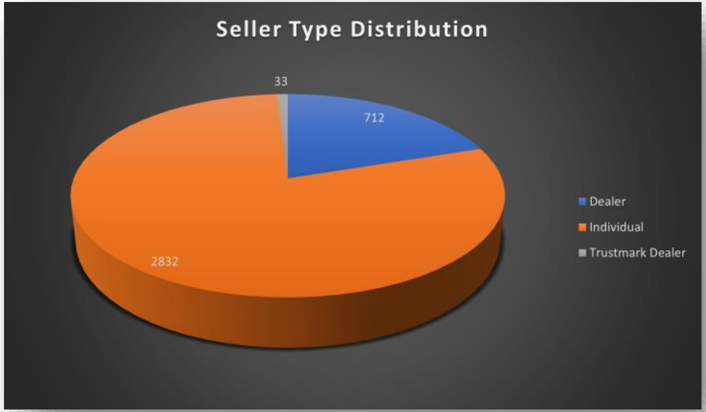
	B	C	D	E	F	G	H	I	J
1	Basic Data Analysis Using Excel								
2									
3	Total Count	4340							
4	Blanks	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
5	Blanks Count	0	0	0	0	0	0	0	0
6	% Blanks	0%	0%	0%	0%	0%	0%	0%	0%
7	Mean	#DIV/0!	2013.1	504127.3118	66215.7[illegible]42	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!
8	Mode	#N/A	2017	300000	70000	#N/A	#N/A	#N/A	#N/A
9	Median	#NUM!	2014	350000	60000	#NUM!	#NUM!	#NUM!	#NUM!
10									
11		name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
12		Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner
13		Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner
14		Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner
15		Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner
16		Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner
17		Maruti Alto LX BSIII	2007	140000	125000	Petrol	Individual	Manual	First Owner
18		Hyundai Xcent 1.2 Kappa S	2016	550000	25000	Petrol	Individual	Manual	First Owner
19		Tata Indigo Grand Petrol	2014	240000	60000	Petrol	Individual	Manual	Second Owner
20		Hyundai Creta 1.6 VTVT S	2015	850000	25000	Petrol	Individual	Manual	First Owner
21		Maruti Celerio Green VXI	2017	365000	78000	CNG	Individual	Manual	First Owner

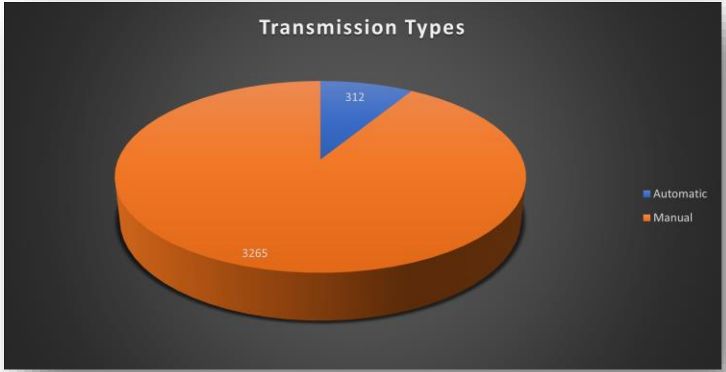


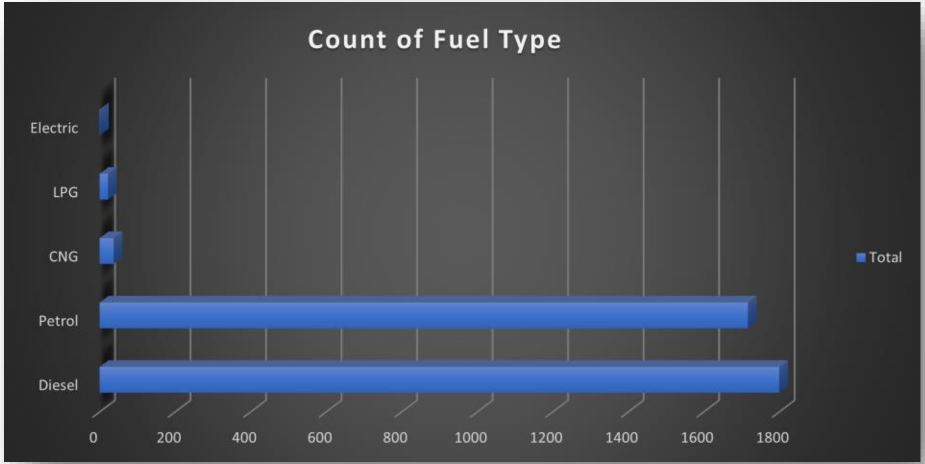
- The Dataset contains **8 rows & 4240** columns showing the information of used cars dataset.
- The Basic analysis shows that there *is no null value* present.
- The Analysis further shows that there are *763 duplicate* values.
- Analise the count of owners by transmission type, focusing on the first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



Data Exploration (Using Excel)

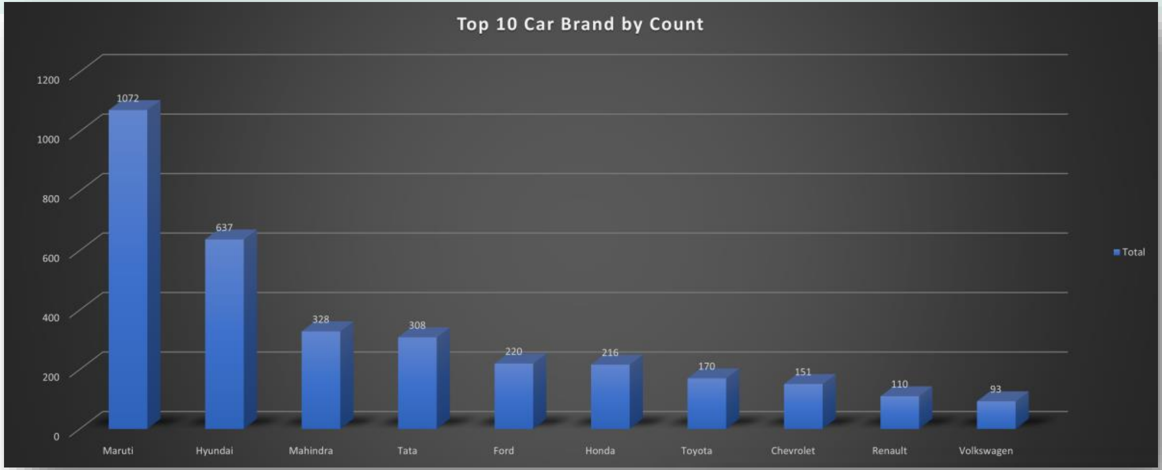






- These analysis shows distribution of different type of sellers, distribution of transmission and count of fuel types.

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 Exploration (Using Python)

Data Cleaning & Analysis

Import necessary library

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
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')
```
✓ 0.0s

- Importing the necessary library.

Data Exploration (Using Python)

skill academy

Loading and Understanding the dataset

```
df = pd.read_csv(r"C:\Users\shubh\Downloads\2\CAR DETAILS.csv")
df.head()
```

✓ 0.0s

	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)
```

✓ 0.0s

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
3385	Hyundai i20 Magna Optional 1.4 CRDi	2012	350000	80000	Diesel	Individual	Manual	First Owner
3422	Maruti Alto 800 VXI	2020	210000	40000	Petrol	Individual	Manual	First Owner
328	Hyundai i20 1.2 Spotz	2017	575000	20000	Petrol	Individual	Manual	Second Owner
1514	Skoda Rapid 1.6 TDI Elegance	2012	275000	120000	Diesel	Individual	Manual	Second Owner
4108	Mahindra Bolero Power Plus Plus AC BSIV PS	2015	295000	90000	Diesel	Individual	Manual	Third Owner
3692	Toyota Fortuner 3.0 Diesel	2012	1680000	129627	Diesel	Dealer	Manual	First Owner
4181	Maruti Swift VDI	2007	225000	50000	Diesel	Dealer	Manual	First Owner
367	Mahindra XUV500 W6 2WD	2012	550000	80000	Diesel	Individual	Manual	First Owner
3492	Skoda Laura Elegance 2.0 TDI CR AT	2019	475000	105000	Diesel	Dealer	Automatic	First Owner
1724	Toyota Innova 2.5 V Diesel 8-seater	2008	500000	154000	Diesel	Individual	Manual	Third Owner

- Loading the dataset.
- Getting the sample 10 data from the dataset(df).

Data Exploration (Using Python)

skill academy

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

✓ 0.0s

```
(4340, 8)
```

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

✓ 0.0s

```
<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()
```

✓ 0.0s

```
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()
```

✓ 0.0s

	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

- Getting the concise summary of the dataset.
- Checking the null values.
- Generating descriptive statistics of the df(dataset).

Data Exploration (Using Python)

```
# To find wethere there are any duplicate values in the df
df.duplicated().sum()
```
✓ 0.0s

```
np.int64(763)
```

```
#To drop duplicate values
df.drop_duplicates(inplace = True)
```
✓ 0.0s

```
df.shape
```
✓ 0.0s

```
(3577, 8)
```

```
#To get information about the columns
df.columns
```
✓ 0.0s

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

- Checking duplicate values and dropping it.
- Checking the shape of dataset.
- Getting the name of all the columns(features).

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()
```

✓ 0.0s

```
(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)
```

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

```
#creating two new column in the dataframe
df[['brand', 'model']] = df['name'].str.split(n=1, expand=True)
```

✓ 0.0s

```
#removing name column
df.drop('name',axis=1,inplace=True)
```

✓ 0.0s

```
df.head()
```

✓ 0.0s

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

+ Code + Markdown

```
new_order=['brand','model','year','km_driven','fuel','seller_type','transmission','owner','selling_price']
df = df.reindex(columns=new_order)
```

✓ 0.0s

- Checking unique values of 'name' column.
- Extracting two new columns, 'brand' and 'model' form the 'name' column.
- Dropping the 'name' column.

Data Exploration (Using Python)

skill academy

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

✓ 0.0s

```
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
Mitsubishi          5
Jaguar              5
Land                5
Volvo               4
Jeep                3
Ambassador          3
OpelCorsa           2
...
Force               1
```

```
# 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()
```

✓ 0.0s

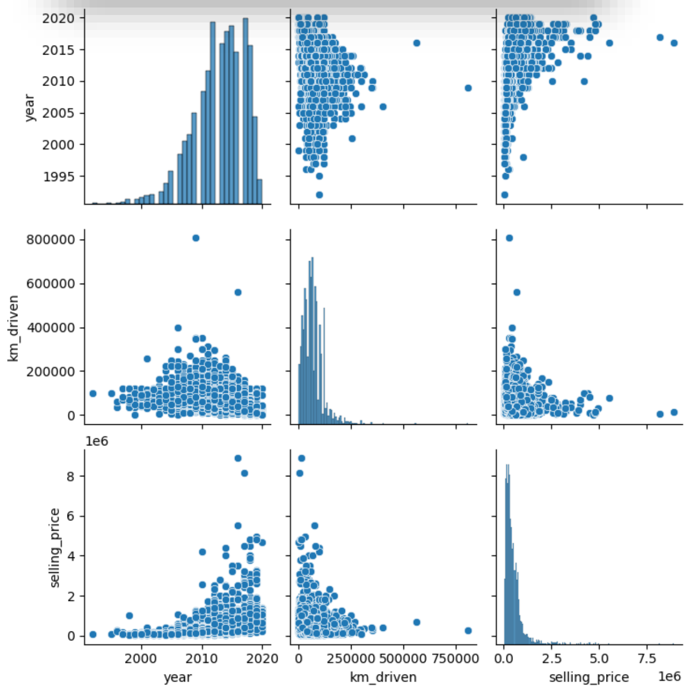
```
brand
Maruti        1072
Hyundai        637
Mahindra       328
Tata           308
Ford           220
Honda          216
Other          171
Toyota         170
Chevrolet      151
Renault        110
Volkswagen      93
Nissan          52
Skoda           49
Name: count, dtype: int64
```

- Applying value_counts to check the how many brand are present in the dataset.
- Applying more than 40 to check which brand which has more than 40 cars.

Data Exploration (Using Python)

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

✓ 1.8s



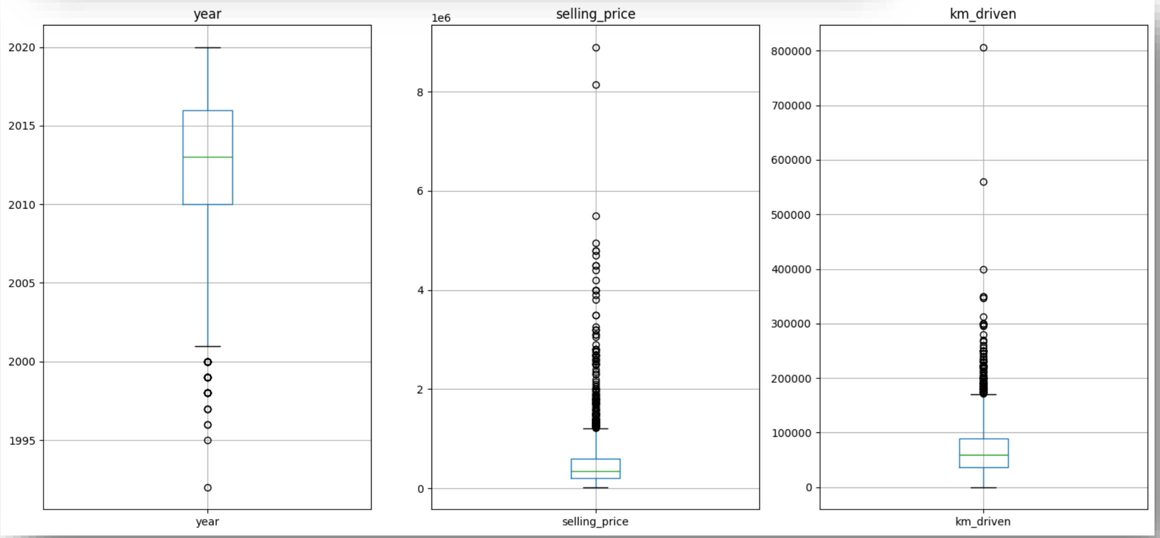
- Pairplot to visualize the dataset.

Data Exploration (Using Python)

skill academy

```
# 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()
```

✓ 0.3s

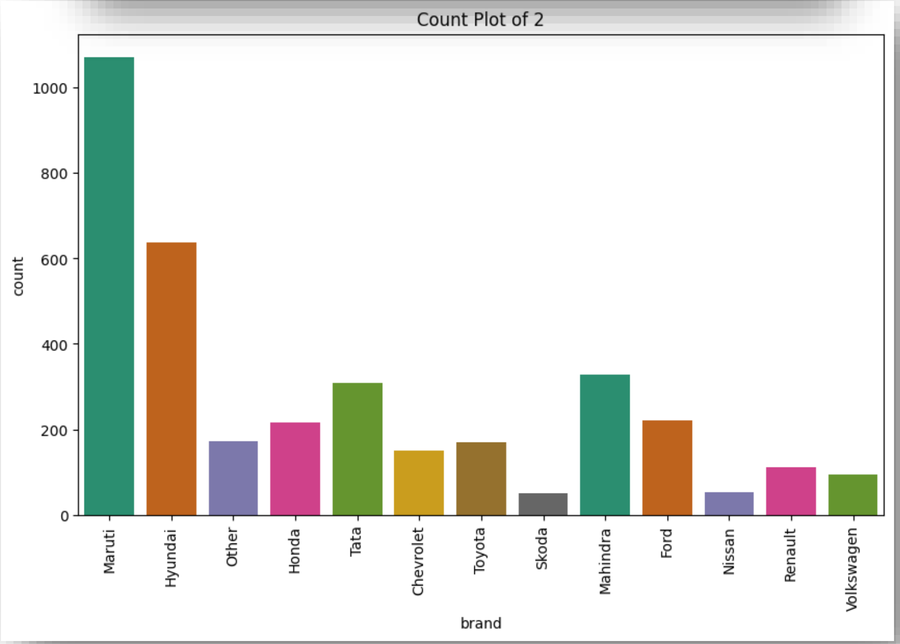


- Boxplot shows that there are outliers in numerical columns. We will handle it later.

Data Exploration (Using Python)

```python
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='brand', palette='Dark2')
plt.title(f'Count Plot of {i}')
plt.xticks(rotation=90)
plt.show()
```

✓ 0.2s



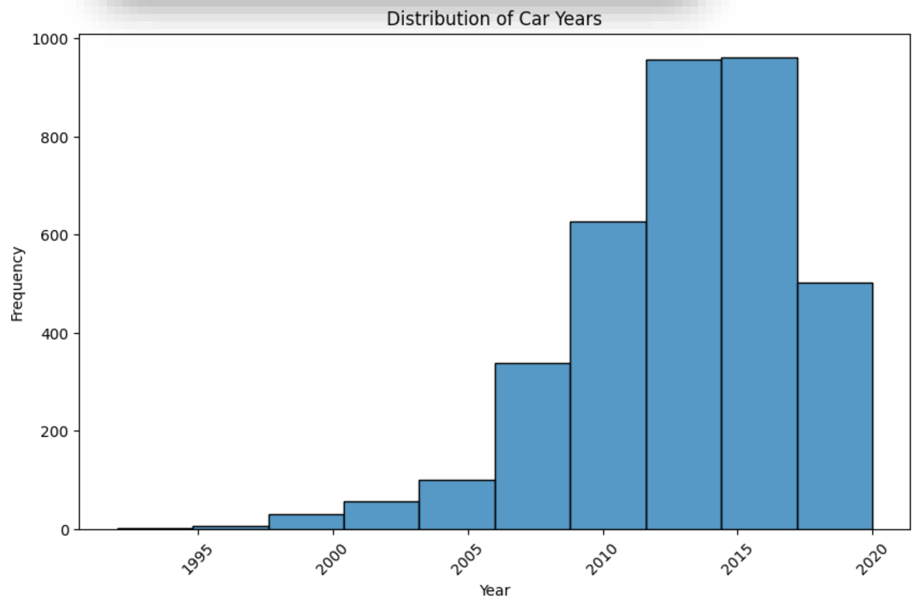
- **Maruti Suzuki is the most popular car brand**, with almost twice the number of sales compared to the second-place brand, Hyundai.
- There is a significant drop in sales between Maruti Suzuki and Hyundai, with the following brands having considerably fewer sales.

Data Exploration (Using Python)

skill academy

```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()
```

✓ 0.1s



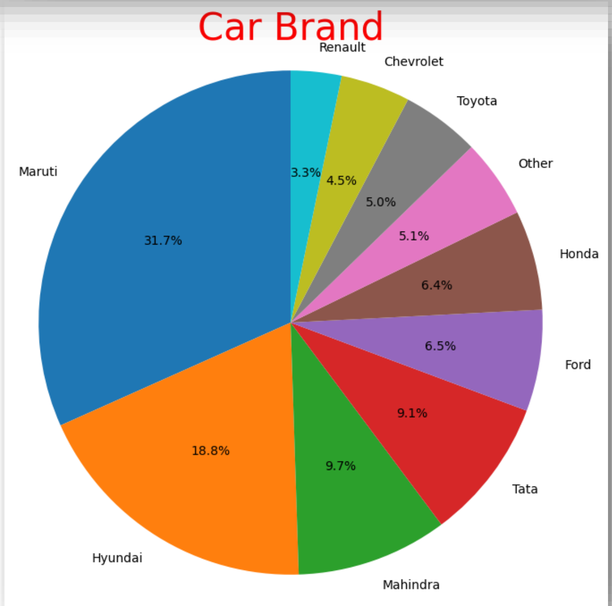
- The **most** frequent car model year is **2015**. There are more cars from 2015 than any other year shown in the data set.
- The distribution of car model years is **skewed to the right**. This means that there are more recent model year cars than older model year cars.
- There are a **few** cars from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

Data Exploration (Using Python)

skill academy

```
top_cars= df['brand'].value_counts().nlargest(10)
plt.figure(figsize=(8, 8))
plt.pie(top_cars, labels=top_cars.index, autopct='%1.1f%%', startangle=90, colors=plt.cm.tab10.colors)
plt.title('Car Brand', fontsize=30, color='red')
plt.axis('equal')
plt.show()
```

✓ 0.1s



- 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)

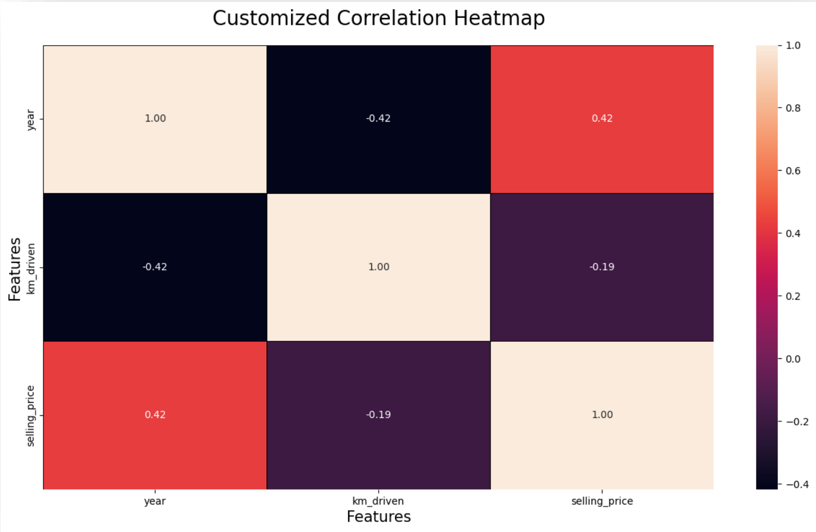
```python
numerical_columns = df.select_dtypes(include='number')
corr = numerical_columns.corr()
corr
```
✓ 0.0s

	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='rocket', 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()
```
✓ 0.1s

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

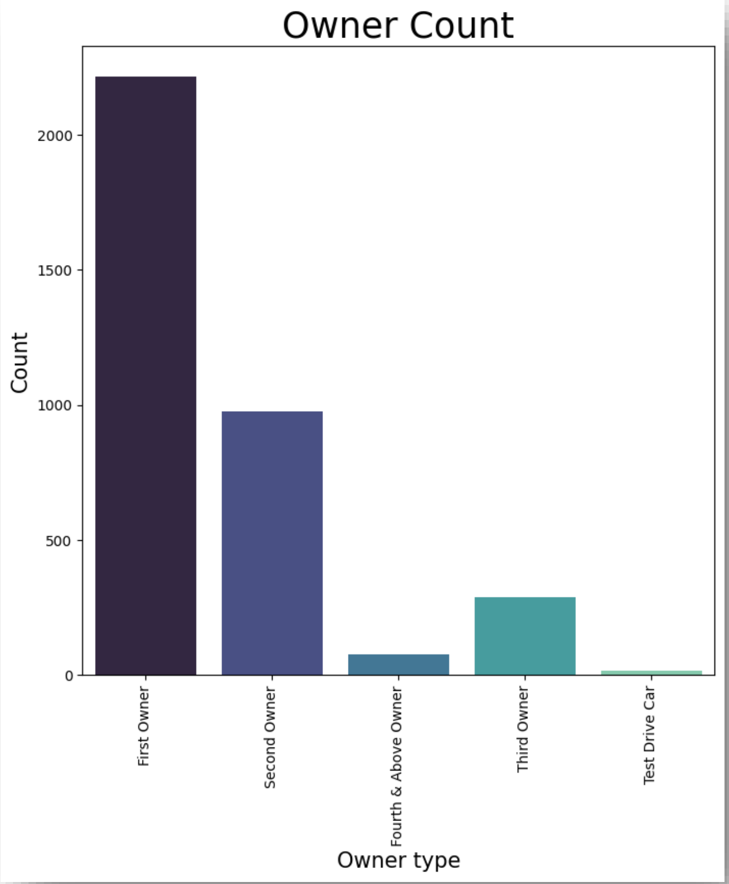
```
df.owner.value_counts()
```

✓ 0.0s

```
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="mako")
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()
```

✓ 0.1s



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

Data Exploration (Using Python)

skill academy

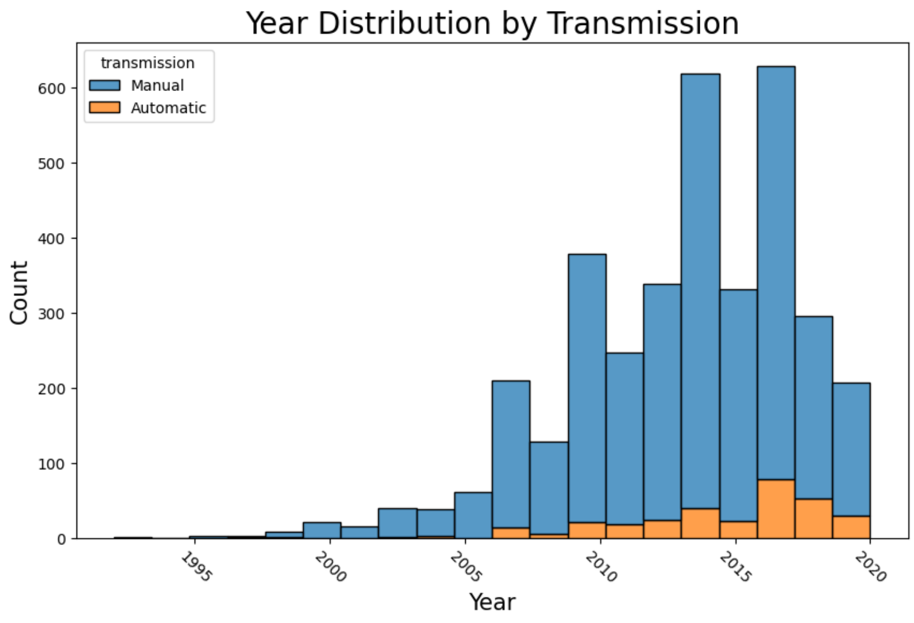
```
df.columns
✓ 0.0s
```

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

```
df['transmission'].value_counts()
✓ 0.0s
```

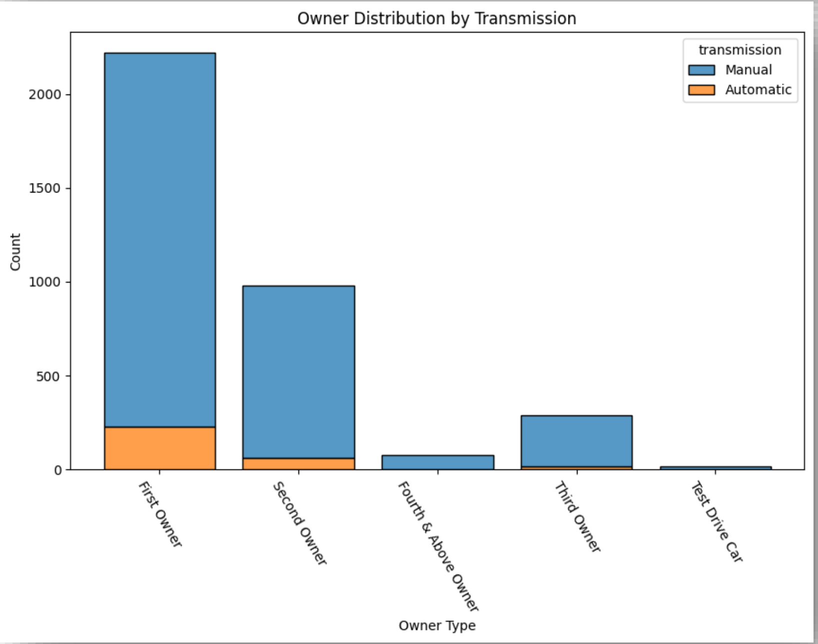
```
transmission
Manual       3265
Automatic     312
Name: count, dtype: int64
```

```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='year', hue='transmission', multiple='stack', bins=20, kde=False)

# Set titles and labels
plt.title('Year Distribution by Transmission', fontsize=20)
plt.xlabel('Year', fontsize=15)
plt.ylabel('Count', fontsize=15)
plt.xticks(rotation=-45)

# Show the plot
plt.show()
```



Data Exploration (Using Python)

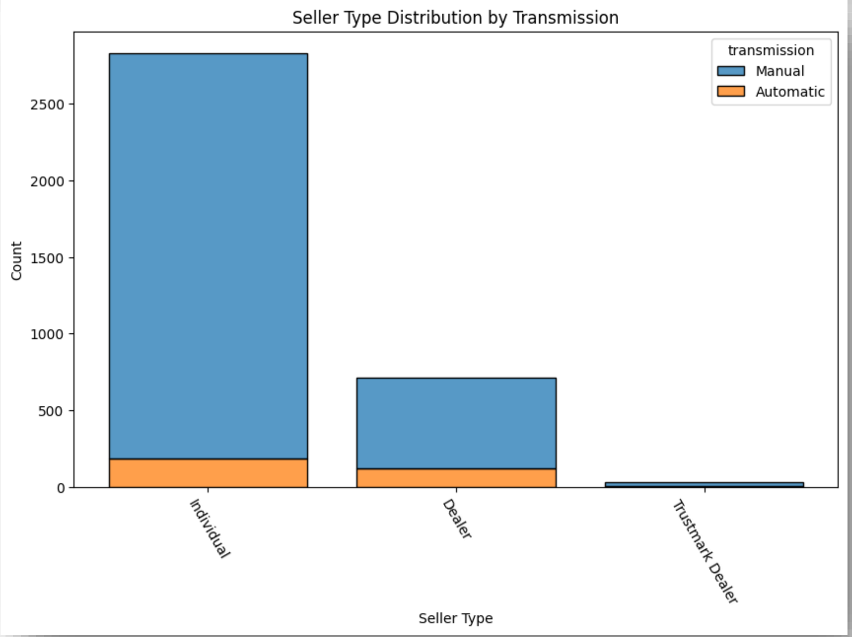
```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='owner', hue='transmission', multiple='stack', shrink=0.8)
plt.title('Owner Distribution by Transmission')
plt.xlabel('Owner Type')
plt.ylabel('Count')
plt.xticks(rotation=-60)
plt.show()
```
✓ 0.2s



- 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)

```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='seller_type', hue='transmission', multiple='stack', shrink=0.8)
plt.title('Seller Type Distribution by Transmission')
plt.xlabel('Seller Type')
plt.ylabel('Count')
plt.xticks(rotation=-60)
plt.show()
```
✓ 0.1s

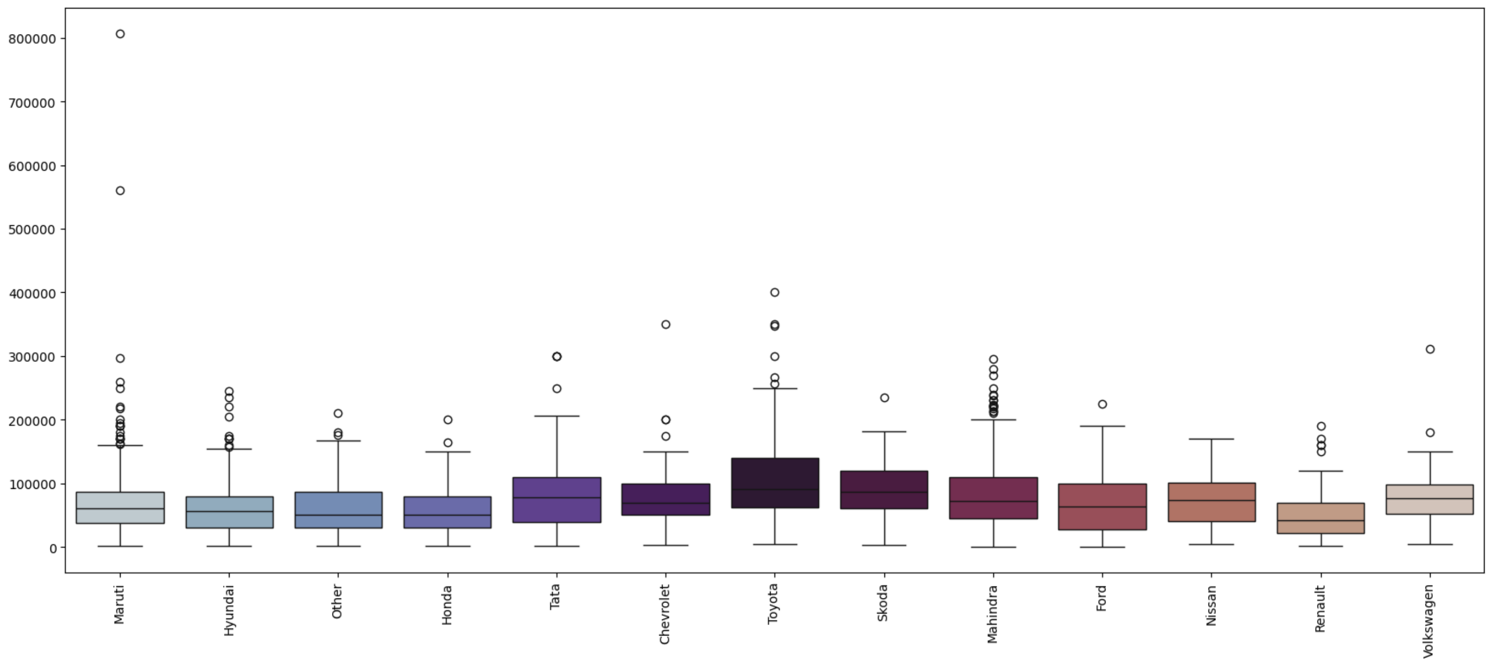


- A total of 2,832 cars were purchased directly by individuals, comprising 2,646 manual and 186 automatic vehicles.
- A total of 712 cars were purchased by individuals through dealers, comprising 593 manual and 119 automatic vehicles.
- A total of 33 cars were purchased by individuals through trustmark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

Data Exploration (Using Python)

```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()
```

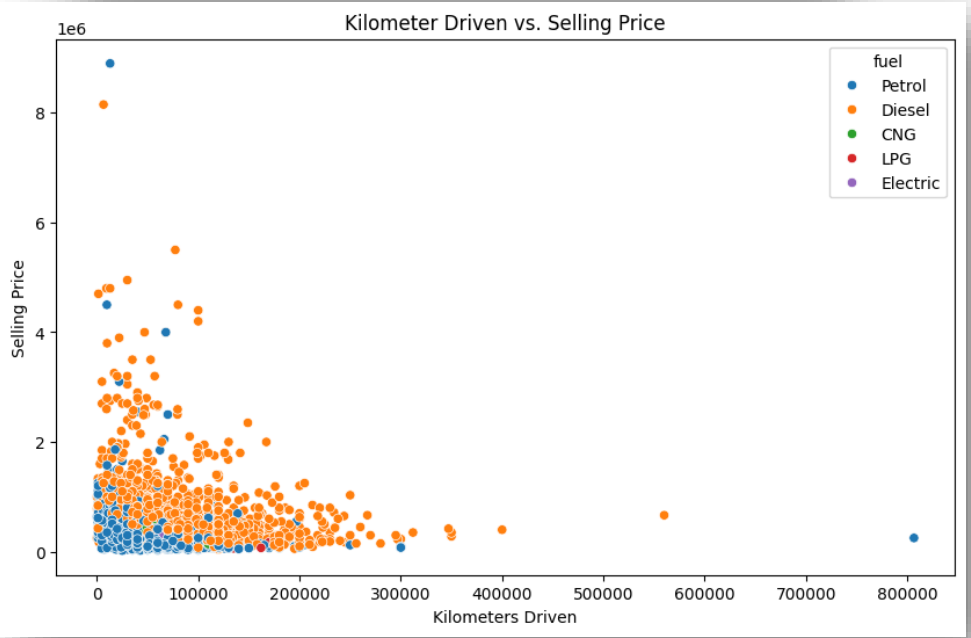
0.2s



Data Exploration (Using Python)

skill academy

```
# 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)

skill academy

Outliers Detections

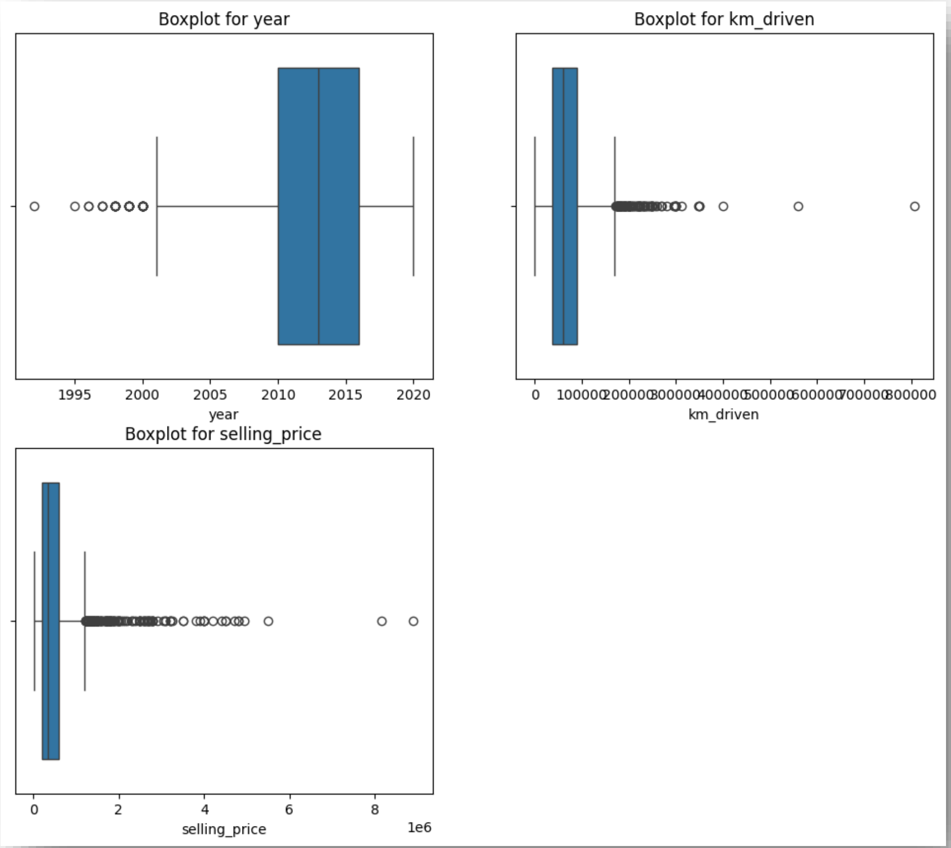
```python
num_cols = df.dtypes[df.dtypes!='object'].index
num_cols
```

✓ 0.0s

```
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()
```

✓ 0.2s



- Outliers Detected in Year, Km_driven and selling price columns.

Data Exploration (Using Python)

skill academy

Outlier Treatment - Cap

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

✓ 0.0s Python

	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
km_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
elling_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

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

✓ 0.0s Python

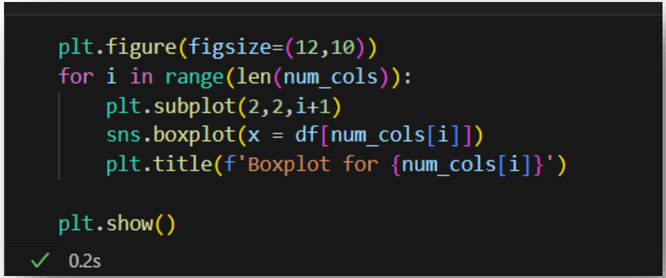
```
(37, 9)
(170, 9)
(179, 9)
```

```python
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'])
```

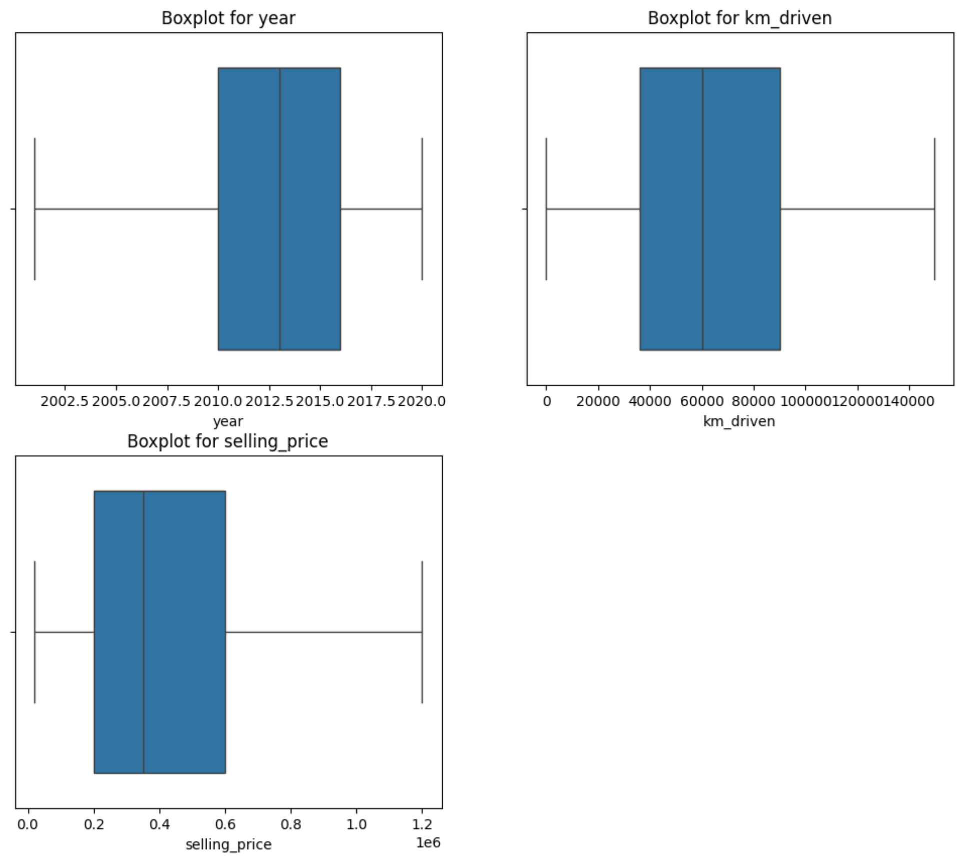
✓ 0.0s Python

- 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)



```
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)

skill academy

```
df1.head()
```
✓ 0.0s

	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)
```
✓ 0.0s

```
df1.to_csv("cleaned data.csv",index=False)
```
✓ 0.0s

```
pd.read_csv('cleaned data.csv')
```
✓ 0.0s

	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

skill academy

Model Building,Training & Testing

Import necessary library

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
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')
```

✓ 1.8s

Loading and Understanding the dataset

```python
df = pd.read_csv(r"C:\Users\shubh\Downloads\2\cleaned data.csv")
df.head()
```

✓ 0.0s

	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

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

✓ 0.0s

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

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

✓ 0.0s

```
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
```

Machine Learning Model Buliding

skill academy

```
cat_cols = df.dtypes[df.dtypes=='object'].index
print(cat_cols)
```
✓ 0.0s

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

```
df.dtypes
```
✓ 0.0s

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

```
df.head()
```
✓ 0.0s

	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()
```
✓ 0.0s

```
for col in cat_cols:
    df[col] = lb.fit_transform(df[col])
```
✓ 0.0s

```
df.head()
```
✓ 0.0s

	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)
```
✓ 0.0s

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)
```
✓ 0.0s

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

```
x_train.head()
```
✓ 0.0s

	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()
```
✓ 0.0s

- Encoding the categorical features.
- Splitting the data into independent(x) and dependent(y) variables.
- Splitting the data into train and test.

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
```

✓ 0.0s

Build ML models

1) Linear Regression

```python
lr1 = LinearRegression()

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

✓ 0.0s

	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
```

✓ 0.0s

	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
```

✓ 0.0s

	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)  # random_state

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

✓ 0.0s

	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
```

✓ 0.2s

	Train_R2	Test_R2	Test_MAE	Test_MSE	Test_RMSE
RandomForestRegressor	0.788152	0.734938	112866.709593	2.488828e+10	157760.202851

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
```

✓ 0.1s

	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
```

✓ 0.0s

	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
```

✓ 0.3s

	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
```

✓ 0.6s

	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

skill academy

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

✓ 0.0s

	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	0.788152	0.734938	112866.709593	2.488828e+10	157760.202851
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
```

✓ 0.0s

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

✓ 0.0s

- The best model is **Random Forest Regression.**
- After this we save the model and load it in the next slide.

Machine Learning Model Buliding

skill academy

Loading the saved model

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

✓ 0.0s Pytho

```
Name of loaded Model : RandomForest.pkl
```

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

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

✓ 0.0s Pytho

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.

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

Machine Learning Model Buliding

skill academy

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

```
random_datasample_df.reset_index()
```

✓ 0.0s

	index	brand	year	km_driven	fuel	seller_type	transmission	owner
0	2684	5	2017.0	39000.0	4	0	1	0
1	1583	5	2008.0	35008.0	4	0	1	0
2	2669	5	2002.0	60000.0	4	1	1	4
3	2284	5	2012.0	149534.8	1	1	1	0
4	2416	4	2005.0	149534.8	1	1	1	4
5	3292	5	2005.0	70000.0	4	1	1	2
6	2598	0	2012.0	120000.0	1	1	0	4
7	2450	0	2014.0	52000.0	1	0	1	0
8	1769	5	2016.0	40000.0	1	1	1	2
9	2754	3	2015.0	70000.0	1	1	1	0
10	1911	5	2015.0	50000.0	4	1	1	2
11	1821	5	2013.0	25000.0	4	1	1	2
12	723	4	2013.0	41988.0	1	0	1	0
13	91	5	2008.0	29173.0	4	0	1	0
14	502	10	2014.0	120000.0	1	1	1	0
15	2465	7	2009.0	45000.0	1	1	0	4
16	2624	10	2012.0	35000.0	1	1	1	0
17	986	4	2011.0	70000.0	1	1	0	0
18	2111	7	2014.0	120000.0	1	1	1	2
19	3176	5	2019.0	5000.0	4	1	1	0

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
✓ 0.0s
```

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

Sample Dataframe contents
```

	brand	year	km_driven	fuel	seller_type	transmission	owner
0	5	2017.0	39000.0	4	0	1	0
1	5	2008.0	35008.0	4	0	1	0
2	5	2002.0	60000.0	4	1	1	4
3	5	2012.0	149534.8	1	1	1	0
4	4	2005.0	149534.8	1	1	1	4
5	5	2005.0	70000.0	4	1	1	2
6	0	2012.0	120000.0	1	1	0	4
7	0	2014.0	52000.0	1	0	1	0
8	5	2016.0	40000.0	1	1	1	2
9	3	2015.0	70000.0	1	1	1	0
10	5	2015.0	50000.0	4	1	1	2
11	5	2013.0	25000.0	4	1	1	2
12	4	2013.0	41988.0	1	0	1	0
13	5	2008.0	29173.0	4	0	1	0
14	10	2014.0	120000.0	1	1	1	0
15	7	2009.0	45000.0	1	1	0	4
16	10	2012.0	35000.0	1	1	1	0
17	4	2011.0	70000.0	1	1	0	0
18	7	2014.0	120000.0	1	1	1	2
19	5	2019.0	5000.0	4	1	1	0

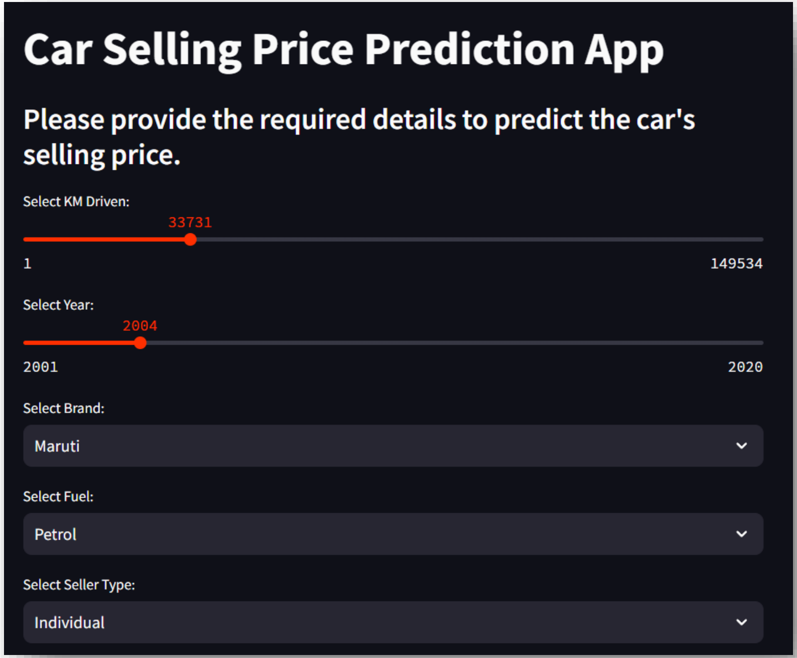
- Loading the sample data and making predictions on it.

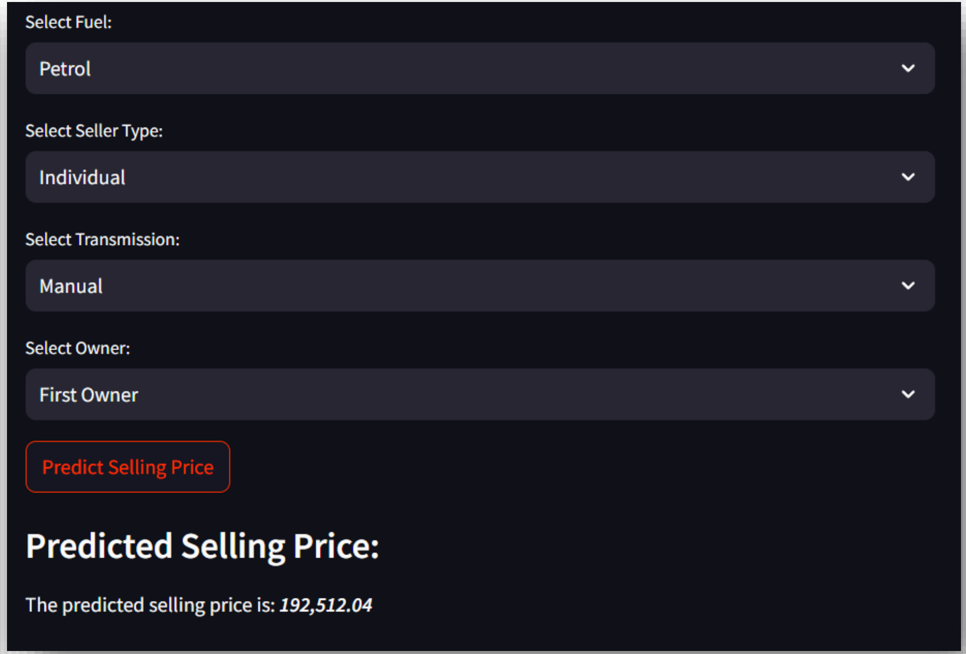
Making Predictions on sample dataset against the trained model

```
# making prediction on random data
predicted_data = load_model.predict(testsample_df)
print(f"The predicted data from RandomForest model:\n", predicted_data)
✓ 0.0s
```

```
The predicted data from RandomForest model:
 [ 399172.39560742  126646.40715668   84215.64066698  406224.98349041
  203691.54928784   99076.8162395   542356.42035961  410605.03177213
  625454.48858924  513792.46656592  299867.19702465  259507.98039449
  461813.86052521  126646.40715668  309616.51446951 1076302.5457455
  229562.55361735  477055.83737642  447721.57131376  478754.4430406 ]
```

Predication of Selling price

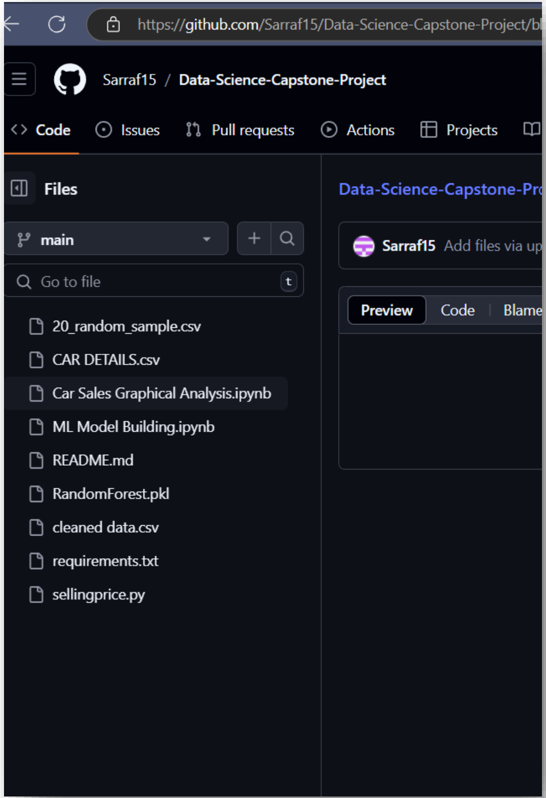




Deployment of ML models using Streamlit

skill academy

Github Code structure



Endnotes

skill academy

About Me :

- Name – Shubham Sarraf
- Course – Data Science With Python career Program (ChatGPT Included)
- LinkedIn - shubhamsarraf15
- GitHub –Sarraf15
- Email –shubhamsarraf98@gmail.com

Reference Links:-

- GitHub Repo Link
- Streamlit App Weblink

END