

;; Exploring car data set ;;

Guided by : Aarati Deshpande

Name; Kiran Sirsat

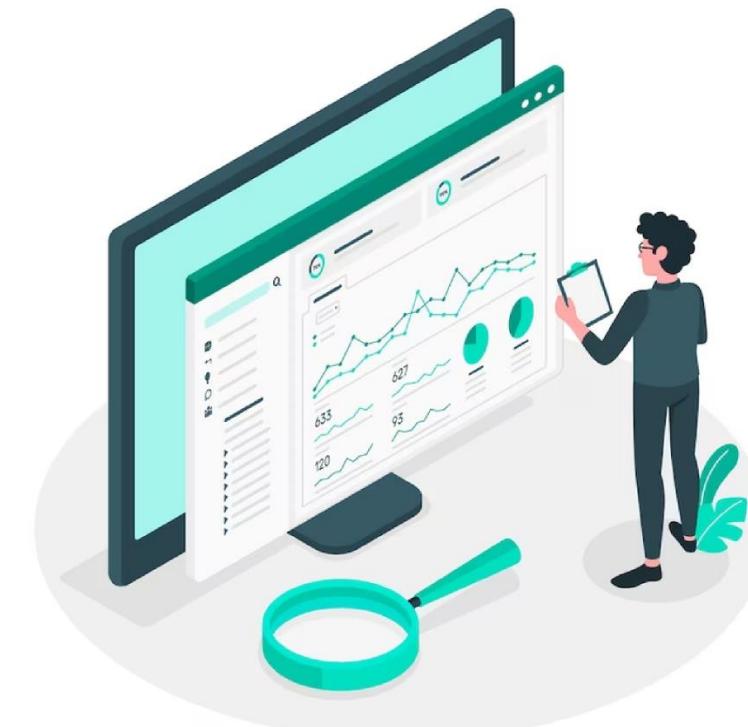
Roll no; 262

Division; B3

python

INTRODUCTION

- Data analytics involves examining and interpreting large datasets to uncover patterns and insights.
- It helps businesses make data-driven decisions and improve their operations.
- Techniques such as statistical analysis, data mining, machine learning, and visualization are used in data analytics.
- It can be applied to various fields like finance, marketing, healthcare, and manufacturing.
- Ethical considerations and data privacy are crucial in data analytics due to the handling of sensitive information.



Motivation

- Analyzing the dataset can provide valuable insights into the depreciation rate of different car models over time.
 - Understanding the relationship between the selling price and the number of kilometers driven can help buyers and sellers determine fair pricing.
 - Identifying the most popular fuel type among car owners can highlight trends and potential market opportunities.
 - Exploring the impact of car ownership (e.g., single owner vs. multiple owners) on the selling price can guide buyers and sellers in making informed decisions.
 - By analyzing the data, one can uncover patterns or correlations between the year of the car and its selling price, helping buyers assess the value retention of different car models.

Details of Dataset



- **Dataset name: Car Sales Dataset**
- **Number of features: 8 (Car name, Year, Selling price of car, Km driven, Fuel type, seller type, transmission, Owner)**
- **Number of records: 5000**

	A	B	C	D	E	F	G	H	I
1	name	year	selling_pri	km_driver	fuel	seller_type	transmissi	owner	
2	Maruti 800	2007	60000	70000	Petrol	Individual	Manual	First Owner	
3	Maruti WagonR	2007	135000	50000	Petrol	Individual	Manual	First Owner	
4	Hyundai Verna	2012	600000	100000	Diesel	Individual	Manual	First Owner	
5	Datsun Redi-GO	2017	250000	46000	Petrol	Individual	Manual	First Owner	
6	Honda Amaze	2014	450000	141000	Diesel	Individual	Manual	Second Owner	
7	Maruti Alto K10	2007	140000	125000	Petrol	Individual	Manual	First Owner	
8	Hyundai Xcent	2016	550000	25000	Petrol	Individual	Manual	First Owner	
9	Tata Indigo CS	2014	240000	60000	Petrol	Individual	Manual	Second Owner	
10	Hyundai Creta	2015	850000	25000	Petrol	Individual	Manual	First Owner	
11	Maruti Celerio	2017	365000	78000	CNG	Individual	Manual	First Owner	
12	Chevrolet Beat	2015	260000	35000	Petrol	Individual	Manual	First Owner	
13	Tata Indigo CS	2014	250000	100000	Petrol	Individual	Manual	First Owner	
14	Toyota Corolla Altis	2018	1650000	25000	Petrol	Dealer	Automatic	First Owner	
15	Maruti 800	2007	60000	70000	Petrol	Individual	Manual	First Owner	
16	Maruti WagonR	2007	135000	50000	Petrol	Individual	Manual	First Owner	
17	Hyundai Verna	2012	600000	100000	Diesel	Individual	Manual	First Owner	
18	Datsun Redi-GO	2017	250000	46000	Petrol	Individual	Manual	First Owner	
19	Honda Amaze	2014	450000	141000	Diesel	Individual	Manual	Second Owner	
20	Maruti Alto K10	2007	140000	125000	Petrol	Individual	Manual	First Owner	

Data Manipulation

MIT

Academy of
Engineering

- Data manipulation modifies, transforms, or restructures data for meaningful insights or specific requirements.
- It involves operations like filtering, sorting, merging, aggregating, and transforming data.
- Programming languages (e.g., Python, SQL) and specialized tools are used for data manipulation.
- Tasks include cleaning, preprocessing, handling missing values, outliers, and creating new variables.
- Data manipulation is vital for data analysis, preparing data, creating derived variables, and ensuring data quality.



```
print(df.describe())
print(df.mean())
print(df.max())
print(df.min())
print(df.median())
```

```
      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
year           2013.090783
selling_price  504127.311751
km_driven      66215.777419
dtype: float64
name          Volvo XC60 D5 Inscription
year            2020
selling_price   8900000
km_driven       806599
fuel            Petrol
seller_type     Trustmark Dealer
transmission    Manual
owner           Third Owner
dtype: object
name          Ambassador CLASSIC 1500 DSL AC
year            1992
selling_price   20000
km_driven        1
fuel             CNG
seller_type     Dealer
transmission    Automatic
owner           First Owner
dtype: object
year           2014.0
selling_price  350000.0
```

```
print(df.corr())
print(df.cov())
```

```
      year  selling_price    km_driven
year           1.000000  0.413922 -0.419688
selling_price  0.413922  1.000000 -0.192289
km_driven      -0.419688 -0.192289  1.000000
year           1.776912e+01  1.009465e+06 -8.251948e+04
selling_price  1.009465e+06  3.347186e+11 -5.189079e+09
km_driven      -8.251948e+04 -5.189079e+09  2.175672e+09
<ipython-input-7-f567bbe92fde>:1: FutureWarning: The default
                                print(df.corr())
<ipython-input-7-f567bbe92fde>:2: FutureWarning: The default
                                print(df.cov())
```

```
df['year'].quantile([0.25,0.50,0.75])
df[['year','selling_price']].quantile([0.25,0.50,0.75])
```

```
      year  selling_price
0.25  2011.0    208749.75
0.50  2014.0    350000.00
```

```
print(df.groupby('year').sum())
print(df.groupby('year').min())
print(df.groupby('year').count())
print(df.groupby('year').mean())
print(df.groupby('year').max())
print(df.groupby('year').get_group(2007))
```

```
      selling_price  km_driven
year
1992          50000    100000
1995          95000    100000
1996         450000     95000
1997        279000    270000
1998       2568000    775000
1999         735000   677020
2000        978000  851243
2001      2352998  1674257
2002       1905000  1786000
2003       1991000  1878441
2004      5113499  3791479
2005      9266107  6884279
2006     17357994 11286427
2007     21818999 11967363
2008     25259193 12928652
2009     44305994 17856608
2010     63104682 21466092
2011     79575988 23874094
2012    154225974 34608325
2013    187133191 28073171
2014    192025984 27776495
2015    222684984 25506372
2016    217185981 19708508
2017    358311985 18866366
2018    333341988  9967565
2019    206508994  4077933
2020    39286998  529784
```

```
      name  selling_price  km_driven  \

```

```
year
1992           Maruti 800 AC BSII      50000    100000
1995           Maruti Gypsy E MG410W ST    95000    100000
1996          Mahindra Jeep CL 500 MDI   200000     35000
1997          Mahindra Jeep CL 500 MDI    50000    70000
1998          Honda City 1.3 EXI      40000    35000
1999        Hyundai Accent GLE 1      45000     2020
2000
```

```
print(df['selling_price'].max())
print(df['selling_price'].min())
print(df['selling_price'].mean())
print(df['selling_price'].sum())
print(df['selling_price'].count())
```

```
89000000
20000
504127.3117511521
2187912533
4340
```

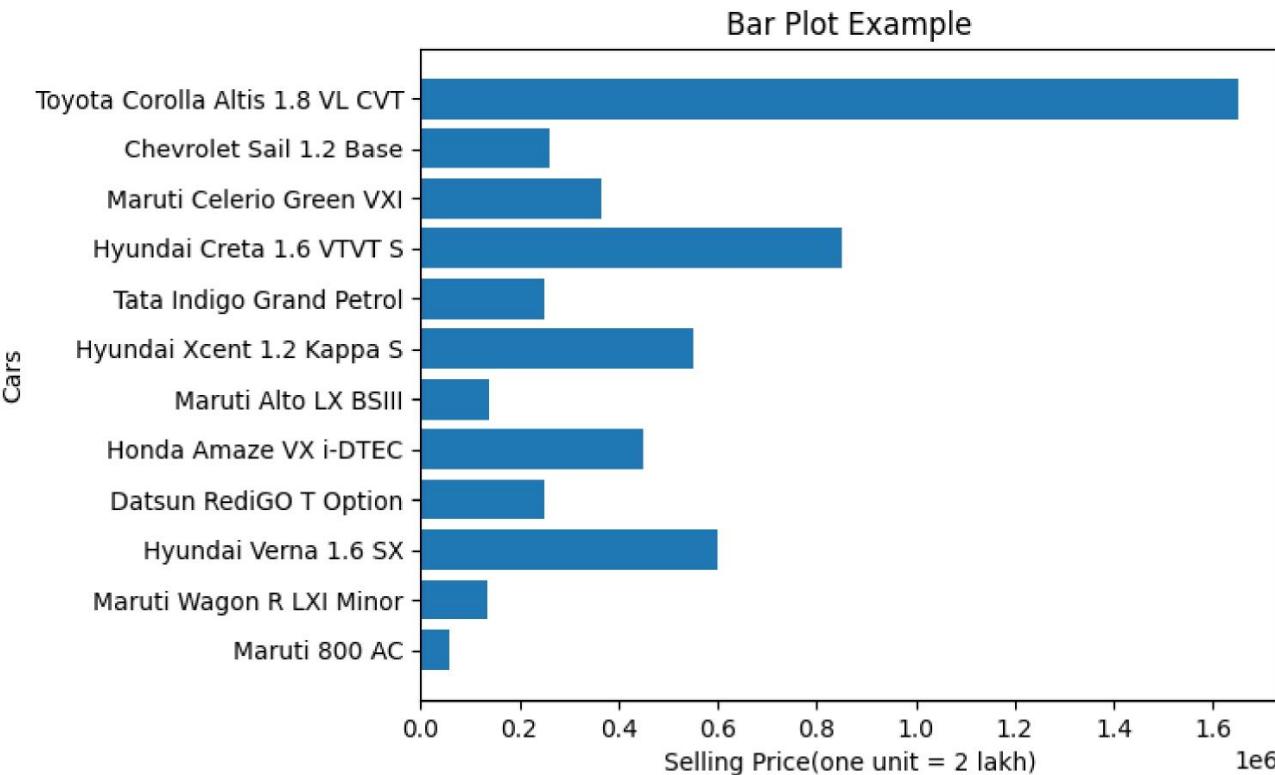
Dataset Example

```
[ ] import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv("/content/CAR DETAILS FROM CAR DEKHO.csv")
print(plt)

# Sample data

categories = df['name'].head(15)

values = df['selling_price'].head(15)
# Create a bar plot
plt.barh(categories, values)
# Customize the plot
plt.title("Bar Plot Example")
plt.xlabel("Selling Price(one unit = 2 lakh)")
plt.ylabel("Cars")
# Display the plot
plt.show()
```

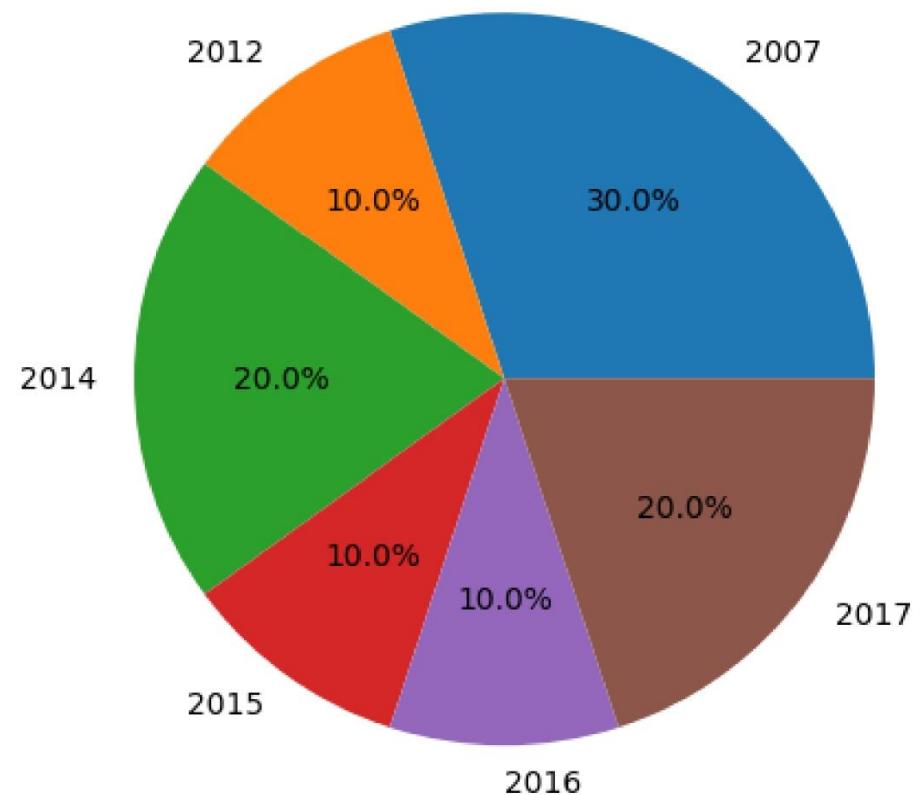


```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('/content/sample_data/car_dekkho.csv')

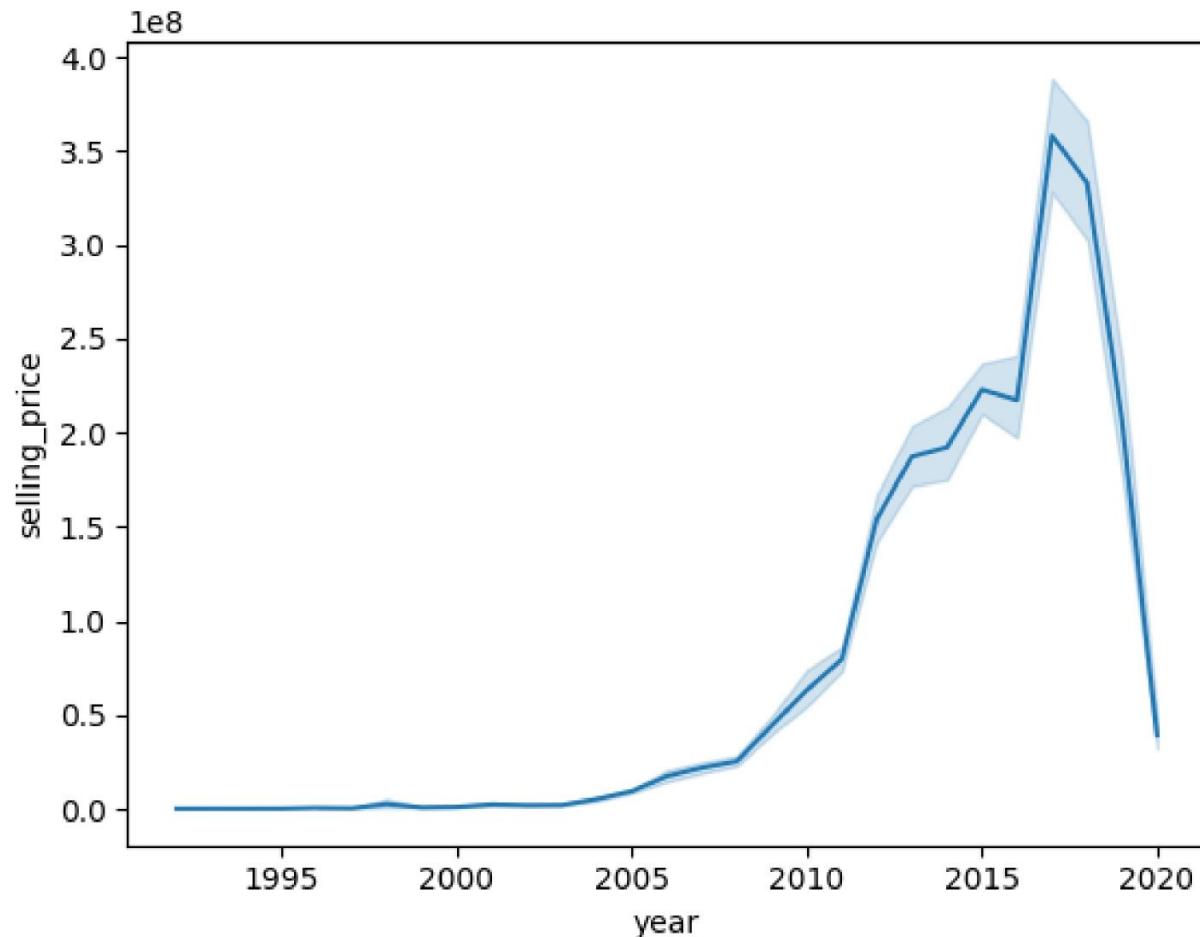
sums = df.head(10).groupby('year')['selling_price'].count()

plt.pie(sums, labels=sums.index, autopct='%1.1f%%')
plt.axis('equal')
plt.show()
```



```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Using seaborn
sns.lineplot(x='year', y='selling_price', data=df, estimator=sum)
plt.show()
```



```

import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('/content/sample_data/car_dekho.csv')

# Assuming 'df' is your DataFrame containing the relevant data

top_10 = df.groupby('fuel').head(10)

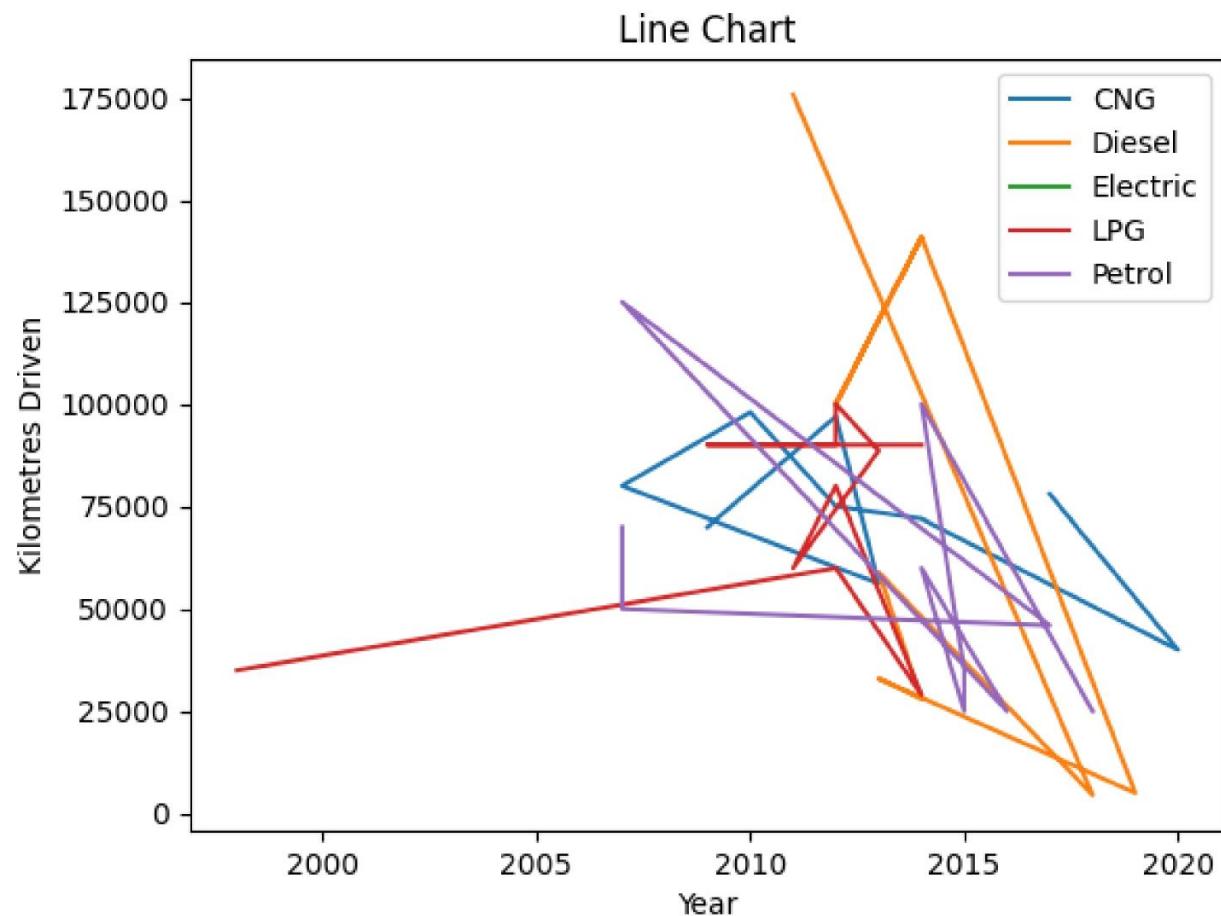
grouped_data = top_10.groupby('fuel')

plt.figure() # Create a new figure

for name, group in grouped_data:
    plt.plot(group['year'], group['km_driven'], label=name)

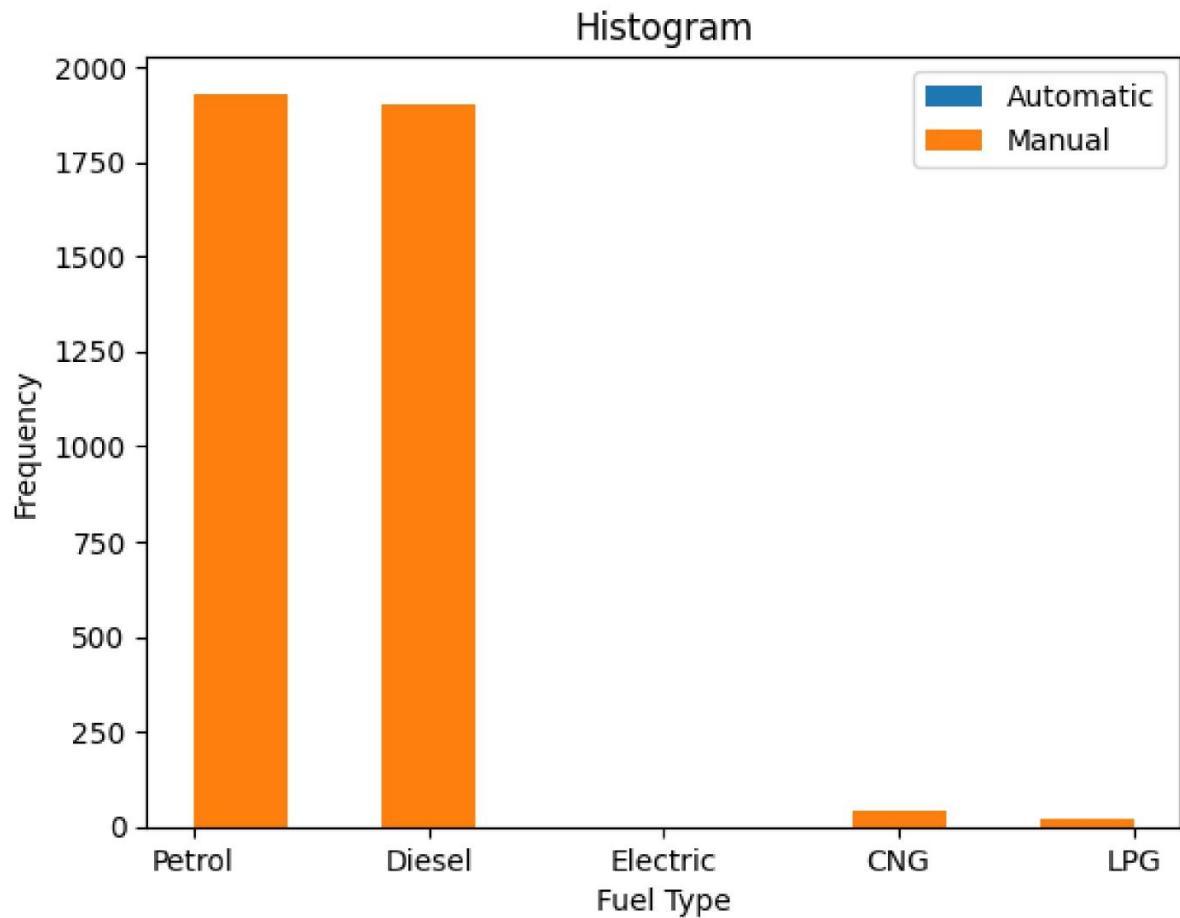
plt.xlabel('Year')
plt.ylabel('Kilometres Driven')
plt.title('Line Chart')
plt.legend()
plt.show()

```



```
grouped_data = df.groupby('transmission')
for name, group in grouped_data:
    plt.hist(group['fuel'], label=name, alpha=1)

plt.xlabel('Fuel Type')
plt.ylabel('Frequency')
plt.title('Histogram')
plt.legend()
plt.show()
```



Predictive Technique



```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Read the CSV file into a DataFrame
df = pd.read_csv("/content/sample_data/Cardekho1.csv")

# Select the features you want to use for clustering
selected_features = ['selling_price', 'km_driven']

# Extract the selected features from the DataFrame
X = df[selected_features]

# Create a KMeans object with the desired number of clusters
kmeans = KMeans(n_clusters=3)

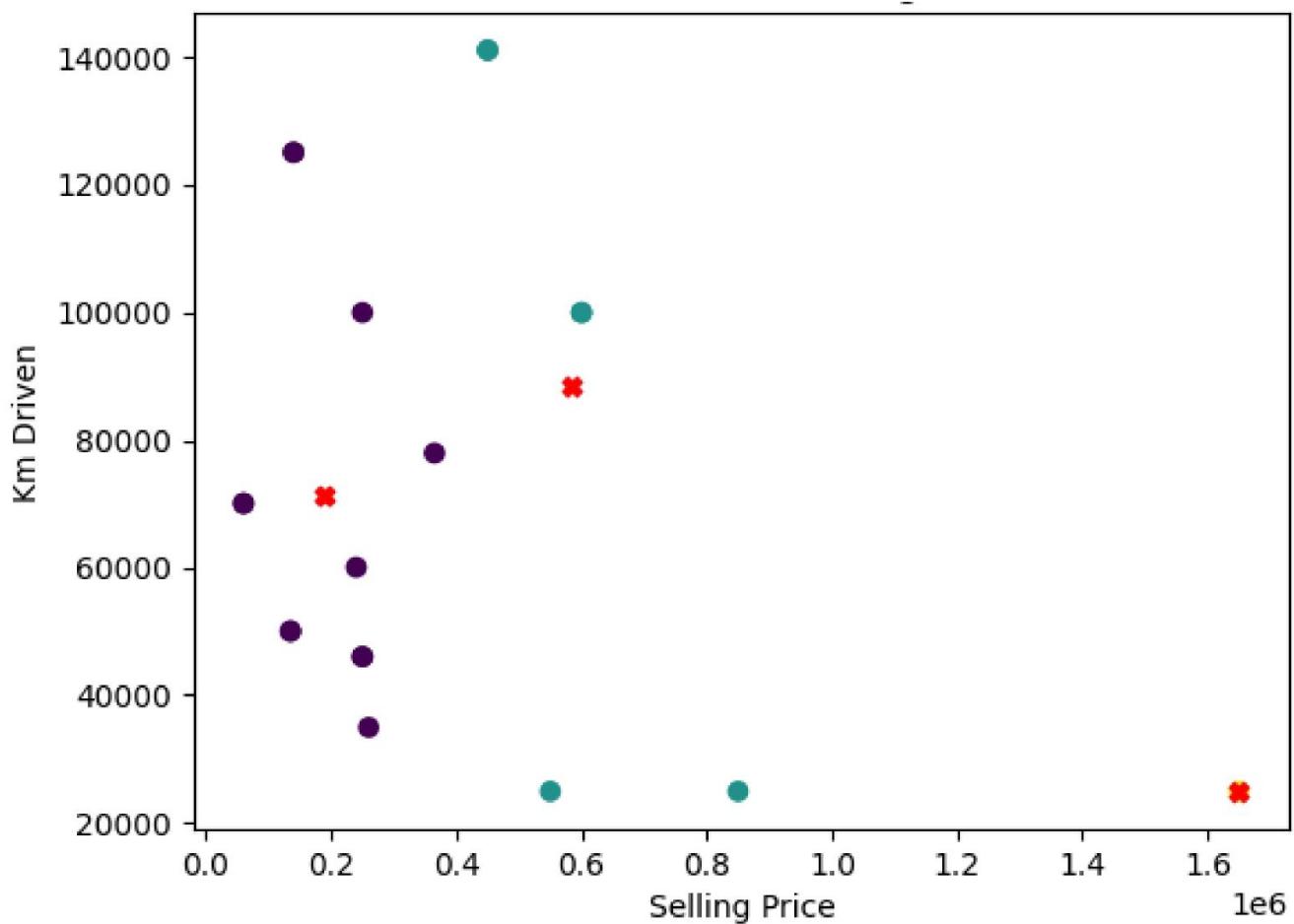
# Fit the KMeans model to the data
kmeans.fit(X)

# Get the cluster labels assigned to each data point
labels = kmeans.labels_

# Get the cluster centers
cluster_centers = kmeans.cluster_centers_

# Add the cluster labels to the DataFrame
df['cluster'] = labels

# Plot the K-means clusters
plt.scatter(X['selling_price'], X['km_driven'], c=labels)
plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1], c='red', marker='x')
plt.xlabel('Selling Price')
plt.ylabel('Km Driven')
plt.title('K-means Clustering')
plt.show()
```



```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Read the CSV file into a DataFrame
df = pd.read_csv("/content/sample_data/Cardekho1.csv")

# Select the features and target variable
selected_features = ['year', 'km_driven']
target_variable = 'selling_price'

# Extract the selected features and target variable from the DataFrame
X = df[selected_features]
y = df[target_variable]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

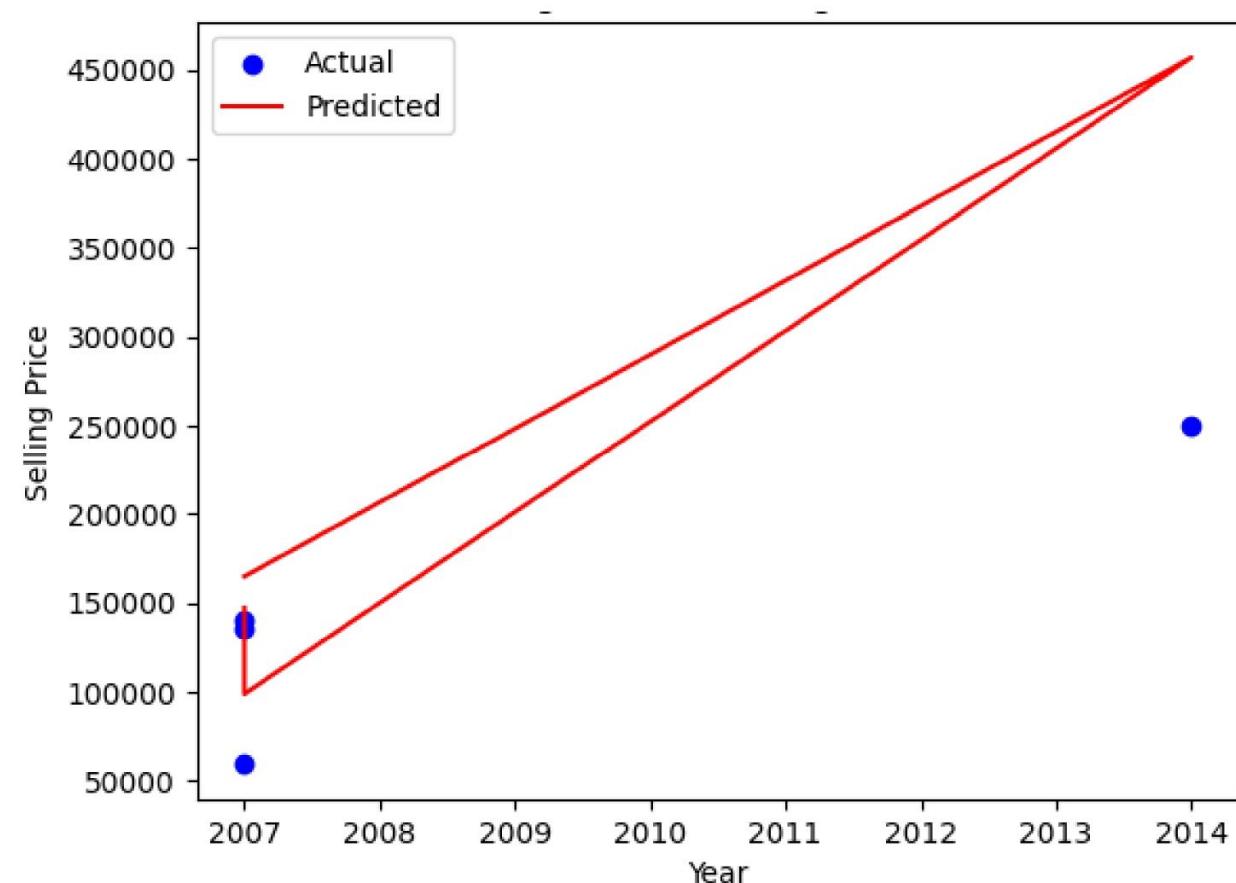
# Create a LinearRegression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Evaluate the model using mean squared error
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)

# Plot the linear regression line
plt.scatter(X_test['year'], y_test, color='blue', label='Actual')
plt.plot(X_test['year'], y_pred, color='red', label='Predicted')
plt.xlabel('Year')
plt.ylabel('Selling Price')
plt.title('Linear Regression - Selling Price vs Year')
plt.legend()
plt.show()
```



Application



- Data manipulation techniques ensure data quality and enhance usability for analysis by handling missing values, removing duplicates, and standardizing formats. Reorganizing and structuring data improves compatibility with analysis techniques, while filtering and selecting specific criteria enable focused analysis. Aggregating and summarizing data yield valuable insights like average selling prices and fuel type distribution. Applying these techniques refines the car dataset for exploration and analysis.
- Data visualization in the car dataset helps in analyzing and presenting information effectively. It includes creating charts to compare fuel type distribution, line graphs to uncover selling price trends over years, interactive dashboards for comprehensive exploration, box plots to understand transmission's impact on prices, and mapping to identify regional preferences or market penetration.
- Predictive techniques in the car dataset include regression analysis for selling price prediction, classification algorithms for fuel type prediction, clustering algorithms like K-means for identifying car segments, estimation of car ownership change likelihood, and forecasting future car demand and market trends. These techniques enhance decision-making and provide valuable insights from the dataset.

Conclusion

- The car dataset provides valuable information on car attributes, including car name, year, selling price, km driven, fuel type, transmission, and owner.
- Analysis of the dataset reveals insights into car pricing trends, fuel type preferences, transmission preferences, and ownership patterns.
- Data manipulation techniques ensure the dataset is clean, transformed, and organized for effective analysis.
- Data visualization aids in visually understanding the dataset's distribution, trends, and relationships, enabling better insights and decision-making.





Thank You
For Your Attention