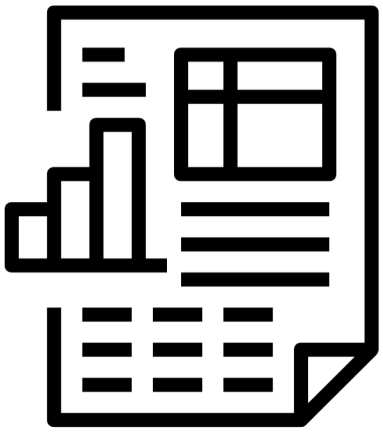
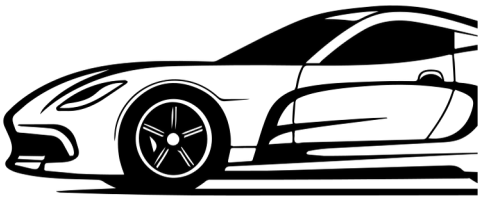




REPORT ANALYSIS ON CAR SALES

ACKNOWLEDGEMENT

We are grateful to everyone who helped us for successfully complete our training project.

We are grateful to Mr. Prateek Gupta for his valuable guidance and insights throughout the training. His expertise and patience were critical in guiding our through the complexities of advanced data science and machine learning.

We are grateful to ITS Ghaziabad for collaborating with Shape My Skills Pvt. Ltd. to offer this valuable learning opportunity. We appreciate our course coordinator, Mr. Neeraj Jain, for his dedication and support during the training. Furthermore, We are grateful to Dr. Sunil Kr. Pandey, Director of the College, for his encouragement and for providing the resources and environment that enabled this learning experience.

Finally, we thank our families and friends for their unwavering support and encouragement throughout this journey.

Thank you all for making this experience worthwhile and memorable.

Priyanshu Mitra & Mohd. Raiyyan

INDEX

TOPICS	PAGES
• **INTRODUCTION** : Overview of Dataset Objective of Analysis	1
• **DATA OVERVIEW** : Number of Rows and Columns Displaying Dataset Information Top and Bottom 10 Rows Column Summary: Mean, Median, Mode	2-5
• **DESCRIPTIVE STATISTICS** : Minimum and Maximum Values Full Dataset Description	6-8
• **DATA CLEANING** : Handling Duplicates Handling Null Values	8
• **EXPLORATORY DATA ANALYSIS (EDA)** : Car Companies in Dataset Car Sales from 2001–2022 Annual Sales Analysis	9-11

INDEX

TOPICS	PAGES
• **DATA VISUALIZATION :** Boxplots for Year, Price, Sales Pairplot Analysis of Relationships Scatterplot for Sales vs. Year Regression and Correlation Analysis Heatmap of Feature Correlations Histogram of Sales Distribution by Company	11-16
• **COMPANY-WISE SALES ANALYSIS :** Sales of Top 10 Companies (2001–2022) 2022 Sales Prediction 2001 Sales Prediction	17-20
• **MACHINE LEARNING MODELS :** Linear Regression Model for Sales Prediction Decision Tree Model for Classification Random Forest Regressor Model for Sales Prediction	21-26
• **MODEL PERFORMANCE EVALUATION :** Comparison of Machine Learning Algorithms	27
• **CONCLUSION**	28

INTRODUCTION OF OUR DATASET

Our dataset is based on “Car SALES”, Which has 7827 rows and 11 columns. We’ll perform our data analysis through python and their libraries. So, We’ll see our data analysis by python codes and their output. After that, We’ll describe also what we want to perform operations with our code.

Coding Results & Discussion

- Imports key libraries for data manipulation, visualization, and machine learning.
- Prepares the environment for subsequent data analysis and modeling tasks.

```
#Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

- Converts the cars_raw dataset from a CSV file to a Pandas Dataframe.
- Displays the dataset as a table.

```
#Loading Dataset
df=pd.read_csv("VAM Dataset.csv")
df
```

	Unique_ID	Year	Make	Model	Price	ConsumerRating	SellerRating	MinMPG	MaxMPG	Mileage	Sales
0	1001	2019	Toyota	Sienna SE	39998	4.6	3.3	19	27	29403	14
1	1002	2018	Ford	F-150 Lariat	49985	4.8	4.8	19	24	32929	20
2	1003	2017	RAM	1500 Laramie	41860	4.7	4.6	15	21	23173	39
3	1004	2021	Honda	Accord Sport SE	28500	5.0	4.6	29	35	10598	38
4	1005	2020	Lexus	RX 350	49000	4.8	4.8	20	27	28137	36
...	...	...	...	...	...	...	...	...	...	...	...
7822	8823	2019	Lexus	RX 350 350	48799	4.8	4.8	19	26	23016	36
7823	8824	2016	Porsche	Cayenne S	45388	4.8	4.8	17	24	15606	23
7824	8825	2017	Dodge	Journey GT	17997	4.7	4.5	16	24	62649	13
7825	8826	2020	Mazda	CX-9 Touring	34764	4.8	4.9	20	26	30760	26
7826	8827	2019	Nissan	Rogue SL	28799	4.7	4.8	25	32	41645	32

7827 rows × 11 columns

- It'll give you whole information about dataset

```
#Dataset Information
df.info()  #It will give the information realted to whole dataset like datatypes in there whole dataset, memory usage and more...
print()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7827 entries, 0 to 7826
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Unique_ID       7827 non-null   int64
 1   Year            7827 non-null   int64
 2   Make            7827 non-null   object
 3   Model           7827 non-null   object
 4   Price           7827 non-null   int64
 5   ConsumerRating  7827 non-null   float64
 6   SellerRating    7827 non-null   float64
 7   MinMPG          7827 non-null   int64
 8   MaxMPG          7827 non-null   int64
 9   Mileage         7827 non-null   int64
 10  Sales           7827 non-null   int64
dtypes: float64(2), int64(7), object(2)
memory usage: 672.8+ KB
```

- It'll display the number of Rows and Columns((no. of rows, no. of columns) in this format)

```
#Dataset Shape
print(df.shape) #Display the Number of Rows and Columns((no. of rows,no. of columns) in this format)
```

```
(7827, 11)
```

- It'll gives you top 10 rows of data from dataset

```
#Gathering starting rows of datas from dataset
df.head(10) #Gives you top 10 rows of datas from dataset
```

	Unique_ID	Year	Make	Model	Price	ConsumerRating	SellerRating	MinMPG	MaxMPG	Mileage	Sales
0	1001	2019	Toyota	Sienna SE	39998	4.6	3.3	19	27	29403	14
1	1002	2018	Ford	F-150 Lariat	49985	4.8	4.8	19	24	32929	20
2	1003	2017	RAM	1500 Laramie	41860	4.7	4.6	15	21	23173	39
3	1004	2021	Honda	Accord Sport SE	28500	5.0	4.6	29	35	10598	38
4	1005	2020	Lexus	RX 350	49000	4.8	4.8	20	27	28137	36
5	1006	2012	Toyota	4Runner SR5	23541	4.7	4.4	17	23	105469	39
6	1007	2017	Honda	HR-V LX	20995	4.6	4.4	28	34	10458	37
7	1008	2014	Mercedes-Benz	E-Class E 350	18985	4.8	4.4	21	30	58157	17
8	1009	2021	Honda	Pilot Touring 8-Passenger	44299	4.8	4.9	19	26	14445	24
9	1010	2020	Dodge	Charger Scat Pack	46773	4.8	4.3	15	24	25642	33

- It'll gives you last 10 rows of data from dataset

```
#Gathering Last rows of datas from dataset
df.tail(10) #Gives you last 10 rows of datas from dataset
```

	Unique_ID	Year	Make	Model	Price	ConsumerRating	SellerRating	MinMPG	MaxMPG	Mileage	Sales
7817	8818	2017	Mercedes-Benz	E-Class E 300 Sport	38391	4.8	3.7	22	30	18426	14
7818	8819	2017	Mercedes-Benz	E-Class E 300 Luxury	38391	4.8	3.7	22	29	18502	35
7819	8820	2013	Toyota	Camry SE	19998	4.4	3.2	25	35	17069	13
7820	8821	2022	Chevrolet	Tahoe Z71	75998	5.0	4.3	15	20	18025	22
7821	8822	2020	Honda	CR-V EX-L	35299	4.8	4.5	27	32	15306	21
7822	8823	2019	Lexus	RX 350 350	48799	4.8	4.8	19	26	23016	36
7823	8824	2016	Porsche	Cayenne S	45388	4.8	4.8	17	24	15606	23
7824	8825	2017	Dodge	Journey GT	17997	4.7	4.5	16	24	62649	13
7825	8826	2020	Mazda	CX-9 Touring	34764	4.8	4.9	20	26	30760	26
7826	8827	2019	Nissan	Rogue SL	28799	4.7	4.8	25	32	41645	32

- It'll display the number of elements in each column

```
#Count the values in dataset
print(df.count()) #Display the number of elements in each column
print()
```

```
Unique_ID          7827
Year               7827
Make               7827
Model              7827
Price              7827
ConsumerRating     7827
SellerRating       7827
MinMPG             7827
MaxMPG             7827
Mileage            7827
Sales              7827
dtype: int64
```

- It’s display the mean of every columns which are mentioned in the program

```
#Mean of Dataset
selected_columns = df[['Unique_ID','Year', 'Price','ConsumerRating','SellerRating','MinMPG','MaxMPG','Mileage','Sales']]
selected_column_means = selected_columns.mean()
print(selected_column_means)
```

```
Unique_ID          4914.000000
Year               2018.715344
Price             40082.255909
ConsumerRating        4.702747
SellerRating          4.407372
MinMPG               22.824709
MaxMPG               29.265875
Mileage           37952.983519
Sales                24.903028
dtype: float64
```

- It’s display the median of every columns which are mentioned in the program

```
#Median of Dataset
selected_columns = df[['Unique_ID','Year', 'Price','ConsumerRating','SellerRating','MinMPG','MaxMPG','Mileage','Sales']]
selected_column_median = selected_columns.median()
print(selected_column_median)
```

```
Unique_ID          4914.0
Year               2019.0
Price             36221.0
ConsumerRating        4.8
SellerRating          4.6
MinMPG               20.0
MaxMPG               27.0
Mileage           32886.0
Sales                25.0
dtype: float64
```

- It’s display the mode of every columns which are mentioned in the program

```
#Mode of Dataset
selected_columns = df[['Unique_ID','Year', 'Price','ConsumerRating','SellerRating','MinMPG','MaxMPG','Mileage','Sales']]
selected_column_mode = selected_columns.mode()
print(selected_column_mode)
```

```
      Unique_ID    Year    Price  ConsumerRating  SellerRating  MinMPG  \
0          1001  2019.0  29995.0             4.8           4.7    19.0
1          1002     NaN      NaN             NaN           NaN     NaN
2          1003     NaN      NaN             NaN           NaN     NaN
3          1004     NaN      NaN             NaN           NaN     NaN
4          1005     NaN      NaN             NaN           NaN     NaN
...         ...     ...      ...             ...           ...     ...
7822       8823     NaN      NaN             NaN           NaN     NaN
7823       8824     NaN      NaN             NaN           NaN     NaN
7824       8825     NaN      NaN             NaN           NaN     NaN
7825       8826     NaN      NaN             NaN           NaN     NaN
7826       8827     NaN      NaN             NaN           NaN     NaN

      MaxMPG  Mileage  Sales
0       26.0      121   22.0
1        NaN      140    NaN
2        NaN      359    NaN
3        NaN      415    NaN
4        NaN      520    NaN
...      ...      ...    ...
7822     NaN   216759    NaN
7823     NaN   226000    NaN
7824     NaN   227000    NaN
7825     NaN   228692    NaN
7826     NaN   234114    NaN
```

- It'll display all minimum values from every columns from dataset which are mentioned in this program

```
#Minimum value from each columns
print("Minimum Year: ",df.Year.min())
print()
print("Minimum Price: ",df.Price.min())
print()
print("Minimum ConsumerRating: ",df.ConsumerRating.min())
print()
print("Minimum Unique_ID: ",df.Unique_ID.min())
print()
print("Minimum SellerRating: ",df.SellerRating.min())
print()
print("Minimum MinMPG: ",df.MinMPG.min())
print()
print("Minimum MaxMPG: ",df.MaxMPG.min())
print()
print("Minimum Mileage: ",df.Mileage.min())
print()
print("Minimum Sales: ",df.Sales.min())
print()
```

```
Minimum Year:  2001

Minimum Price:  2300

Minimum ConsumerRating:  2.5

Minimum Unique_ID:  1001

Minimum SellerRating:  1.0

Minimum MinMPG:  0

Minimum MaxMPG:  0

Minimum Mileage:  121

Minimum Sales:  10
```

- It'll display all maximum values from every columns from dataset which are mentioned in this program

```
#Maximum value from each columns
print("Maximum Year: ",df.Year.max())
print()
print("Maximum Price: ",df.Price.max())
print()
print("Maximum ConsumerRating: ",df.ConsumerRating.max())
print()
print("Maximum Unique_ID: ",df.Unique_ID.max())
print()
print("Maximum SellerRating: ",df.SellerRating.max())
print()
print("Maximum MinMPG: ",df.MinMPG.max())
print()
print("Maximum MaxMPG: ",df.MaxMPG.max())
print()
print("Maximum Mileage: ",df.Mileage.max())
print()
print("Maximum Sales: ",df.Sales.max())
print()
```

```
Maximum Year:  2022

Maximum Price:  409999

Maximum ConsumerRating:  5.0

Maximum Unique_ID:  8827

Maximum SellerRating:  5.0

Maximum MinMPG:  150

Maximum MaxMPG:  133

Maximum Mileage:  234114

Maximum Sales:  40
```

- It will display the whole description of the dataset

```
#Describe the information of Dataset
print(df.describe()) # It will display the whole description of data of file
print()
```

	Unique_ID	Year	Price	ConsumerRating	SellerRating	\
count	7827.000000	7827.000000	7827.000000	7827.000000	7827.000000	
mean	4914.000000	2018.715344	40082.255909	4.702747	4.407372	
std	2259.604611	2.257672	20977.853481	0.245842	0.633776	
min	1001.000000	2001.000000	2300.000000	2.500000	1.000000	
25%	2957.500000	2018.000000	28998.000000	4.700000	4.300000	
50%	4914.000000	2019.000000	36221.000000	4.800000	4.600000	
75%	6870.500000	2020.000000	46498.000000	4.800000	4.800000	
max	8827.000000	2022.000000	409999.000000	5.000000	5.000000	

	MinMPG	MaxMPG	Mileage	Sales
count	7827.000000	7827.000000	7827.000000	7827.000000
mean	22.824709	29.265875	37952.983519	24.903028
std	15.072648	13.012918	25776.743369	8.922240
min	0.000000	0.000000	121.000000	10.000000
25%	18.000000	25.000000	18598.000000	17.000000
50%	20.000000	27.000000	32886.000000	25.000000
75%	24.000000	31.000000	48174.500000	33.000000
max	150.000000	133.000000	234114.000000	40.000000

- It will search all duplicate values from dataset then sum all duplicate values as total and then display

```
#Duplicate values in the dataset
df.duplicated().sum()
```

```
0
```

- It will search all null values in every columns from dataset then sum all null values as total and then display

```
#Check Null Values
df.isnull().sum()
```

```
Unique_ID         0
Year              0
Make              0
Model             0
Price             0
ConsumerRating    0
SellerRating      0
MinMPG            0
MaxMPG            0
Mileage           0
Sales             0
dtype: int64
```

- It'll display all Car's company which are in the dataset

```
#All cars in this dataset
import pandas as pd
# Load the dataset
file_path = "VAM Dataset.csv"
data = pd.read_csv("VAM Dataset.csv")
cars= data['Make'].unique()
print(cars)
```

```
['Toyota' 'Ford' 'RAM' 'Honda' 'Lexus' 'Mercedes-Benz' 'Dodge' 'Subaru'
 'Acura' 'BMW' 'Audi' 'Volvo' 'Lincoln' 'Land' 'Chevrolet' 'INFINITI'
 'Tesla' 'Jeep' 'Chrysler' 'Mazda' 'Kia' 'Volkswagen' 'Porsche' 'Nissan'
 'Hyundai' 'GMC' 'Buick' 'Genesis' 'Cadillac' 'Alfa' 'FIAT' 'Jaguar'
 'MINI' 'Lamborghini' 'Maserati' 'Mitsubishi' 'Bentley' 'Mercury' 'Scion'
 'Saturn' 'Ferrari']
```

- It'll show how many cars were sold by each companies from 2001-2022

```
#It'll show how many cars were sold from each company in every years (2001-2022)
import pandas as pd

df = pd.read_csv("VAM Dataset.csv")
# Filter the data for the years 2001 to 2022
filtered_df = df[(df['Year'] >= 2001) & (df['Year'] <= 2022)]

# Group by 'Make' and count the number of sales
sales_count = filtered_df.groupby('Make').size().reset_index(name='SalesCount')

# Display the result
print(sales_count)
```

	Make	SalesCount
0	Acura	191
1	Alfa	33
2	Audi	367
3	BMW	800
4	Bentley	5
5	Buick	93
6	Cadillac	257
7	Chevrolet	320
8	Chrysler	31
9	Dodge	119
10	FIAT	2
11	Ferrari	1
12	Ford	468
13	GMC	210
14	Genesis	19
15	Honda	614
16	Hyundai	172
17	INFINITI	177
18	Jaguar	28
19	Jeep	400
20	Kia	206
21	Lamborghini	12
22	Land	157
23	Lexus	413
24	Lincoln	106
25	MINI	4
26	Maserati	13
27	Mazda	177
28	Mercedes-Benz	689
29	Mercury	2
30	Mitsubishi	15
31	Nissan	199
32	Porsche	113
33	RAM	109
34	Saturn	1
35	Scion	3
36	Subaru	260
37	Tesla	164
38	Toyota	630
39	Volkswagen	113
40	Volvo	134

- It’ll display, How many cars were sold in every years from 2001-2022

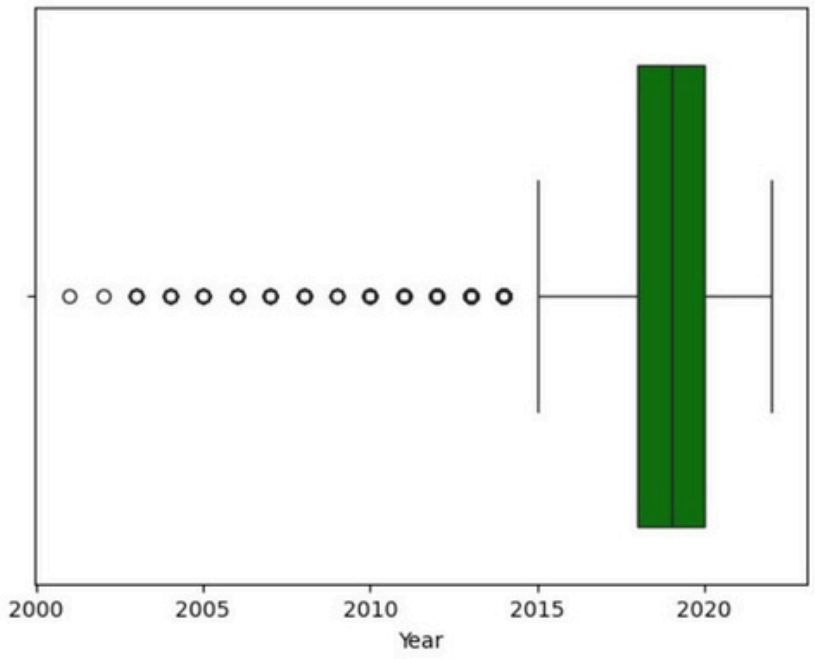
```python
#It'll show how many sales of cars in every years (2001-202
import pandas as pd
# Load the dataset
file_path = "VAM Dataset.csv"
data = pd.read_csv("VAM Dataset.csv")

# Filter for the year 2022
cars = data[data['Year'] == 2022]
num_cars = cars.shape[0] # Count the number of cars sold in
print(f"Number of cars sold in 2022: {num_cars}")
cars = data[data['Year'] == 2021]
num_cars = cars.shape[0]
print(f"Number of cars sold in 2021: {num_cars}")
cars = data[data['Year'] == 2020]
num_cars = cars.shape[0]
print(f"Number of cars sold in 2020: {num_cars}")
cars = data[data['Year'] == 2019]
num_cars = cars.shape[0]
print(f"Number of cars sold in 2019: {num_cars}")
cars = data[data['Year'] == 2018]
num_cars = cars.shape[0]
print(f"Number of cars sold in 2018: {num_cars}")
cars = data[data['Year'] == 2017]
num_cars = cars.shape[0]
print(f"Number of cars sold in 2017: {num_cars}")
cars = data[data['Year'] == 2016]
num_cars = cars.shape[0]
print(f"Number of cars sold in 2016: {num_cars}")
cars = data[data['Year'] == 2015]
num_cars = cars.shape[0]
cars = data[data['Year'] == 2014]
num_cars = cars.shape[0]
print(f"Number of cars sold in 2014: {num_cars}")
cars = data[data['Year'] == 2013]
num_cars = cars.shape[0]
print(f"Number of cars sold in 2013: {num_cars}")
cars = data[data['Year'] == 2012]
num_cars = cars.shape[0]
print(f"Number of cars sold in 2012: {num_cars}")
cars = data[data['Year'] == 2011]
num_cars = cars.shape[0]
print(f"Number of cars sold in 2011: {num_cars}")
cars = data[data['Year'] == 2010]
num_cars = cars.shape[0]
print(f"Number of cars sold in 2010: {num_cars}")
cars = data[data['Year'] == 2009]
num_cars = cars.shape[0]
print(f"Number of cars sold in 2009: {num_cars}")
cars = data[data['Year'] == 2008]
num_cars = cars.shape[0]
print(f"Number of cars sold in 2008: {num_cars}")
cars = data[data['Year'] == 2007]
num_cars = cars.shape[0]
print(f"Number of cars sold in 2007: {num_cars}")
cars = data[data['Year'] == 2006]
num_cars = cars.shape[0]
print(f"Number of cars sold in 2006: {num_cars}")
cars = data[data['Year'] == 2005]
num_cars = cars.shape[0]
print(f"Number of cars sold in 2005: {num_cars}")
cars = data[data['Year'] == 2004]
num_cars = cars.shape[0]
cars = data[data['Year'] == 2003]
num_cars = cars.shape[0]
print(f"Number of cars sold in 2003: {num_cars}")
cars = data[data['Year'] == 2002]
num_cars = cars.shape[0]
print(f"Number of cars sold in 2002: {num_cars}")
cars = data[data['Year'] == 2001]
num_cars = cars.shape[0]
print(f"Number of cars sold in 2001: {num_cars}")
```

```
Number of cars sold in 2022: 85
Number of cars sold in 2021: 1113
Number of cars sold in 2020: 1676
Number of cars sold in 2019: 2710
Number of cars sold in 2018: 911
Number of cars sold in 2017: 473
Number of cars sold in 2016: 276
Number of cars sold in 2015: 187
Number of cars sold in 2014: 119
Number of cars sold in 2013: 86
Number of cars sold in 2012: 54
Number of cars sold in 2011: 41
Number of cars sold in 2010: 28
Number of cars sold in 2009: 7
Number of cars sold in 2008: 13
Number of cars sold in 2007: 12
Number of cars sold in 2006: 8
Number of cars sold in 2005: 10
Number of cars sold in 2004: 9
Number of cars sold in 2003: 7
Number of cars sold in 2002: 1
Number of cars sold in 2001: 1
```

- It's the data visualization, Which is describe the boxplot of "Year" column

```
#Boxplot of year
sns.boxplot(x='Year',color='g', data=df)
```

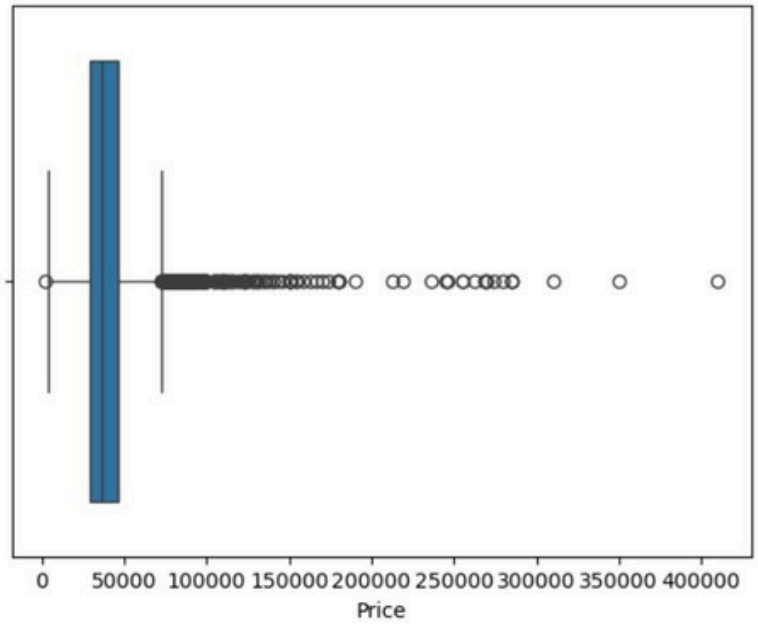
```
<Axes: xlabel='Year'>
```



- It’s the data visualization, Which is describe the boxplot of “Price” column

```
#Boxplot of Price
sns.boxplot(x='Price', data=df)
```

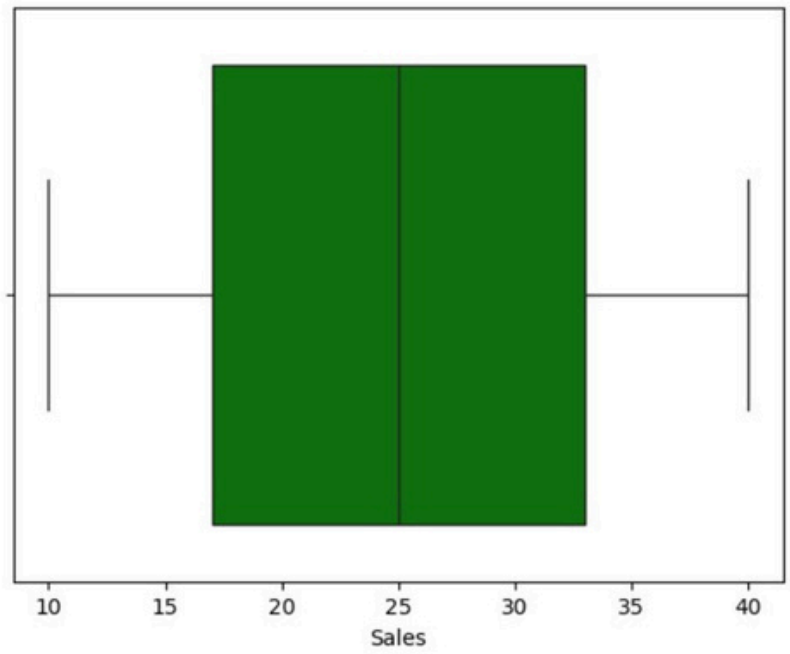
```
<Axes: xlabel='Price'>
```



- It’s the data visualization, Which is describe the boxplot of “Sales” column

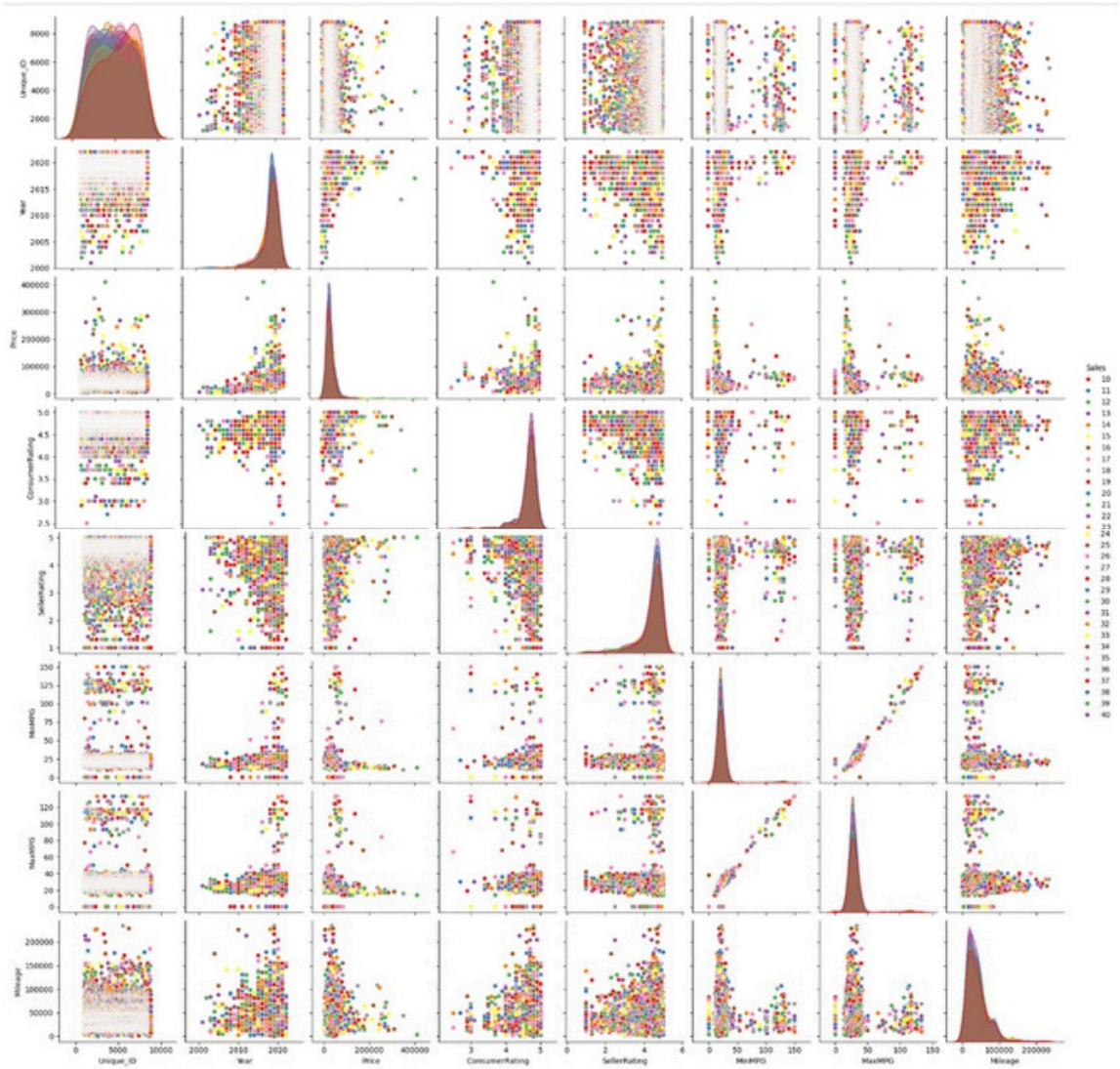
```
#Boxplot of Sales
sns.boxplot(x='Sales',color='g', data=df)
```

```
<Axes: xlabel='Sales'>
```



- It's the data visualization, Which is describe the relation between the columns in dataset through pairplot

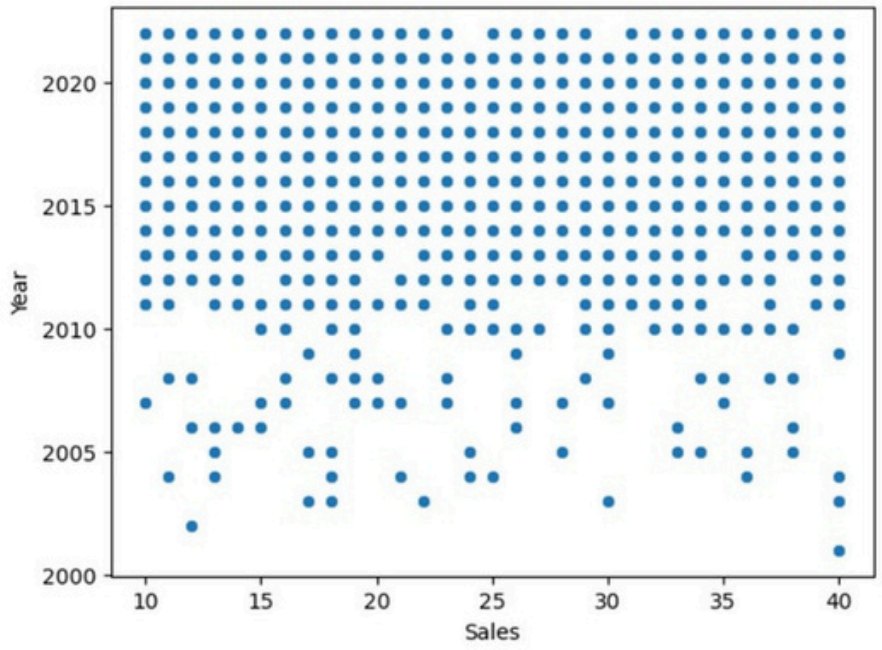
```
#pairplot (it allows you to plot pairwise relationships in a dataset)
sns.pairplot(df[['Unique_ID','Year', 'Price','ConsumerRating','SellerRating','MinMPG','MaxMPG','Mileage','Sales']],
             hue='Sales', palette='Set1')
plt.show()
```



- It's the visualization, Which is describe the relation between two continuous variables (Sales and Year) in the dataset through scatterplot

```
#scatterplot (it display relationship between two continuous variables)
sns.scatterplot(x='Sales', y= 'Year',data=df)
```

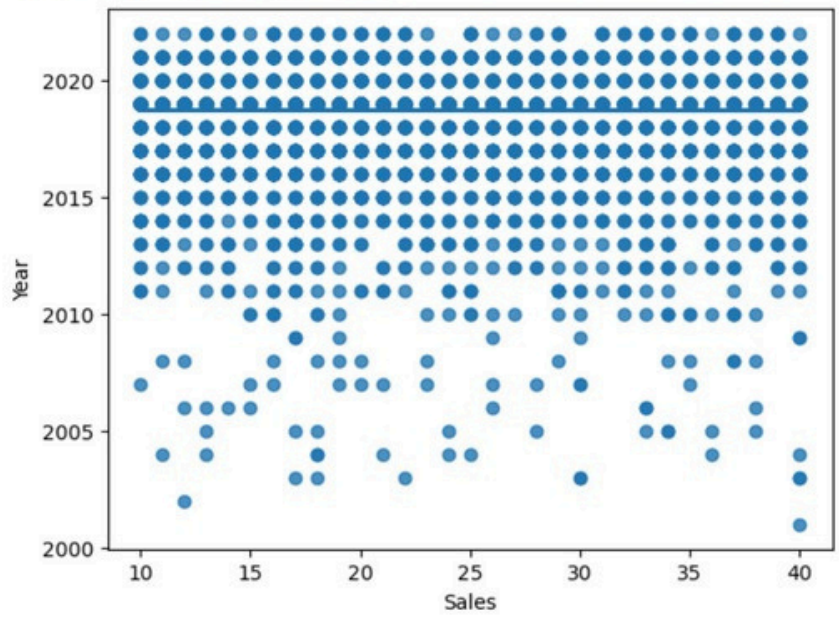
```
<Axes: xlabel='Sales', ylabel='Year'>
```



- It's also the visualization, Which is describe the relation between two continuous variables (Sales and Year) in the dataset through regplot

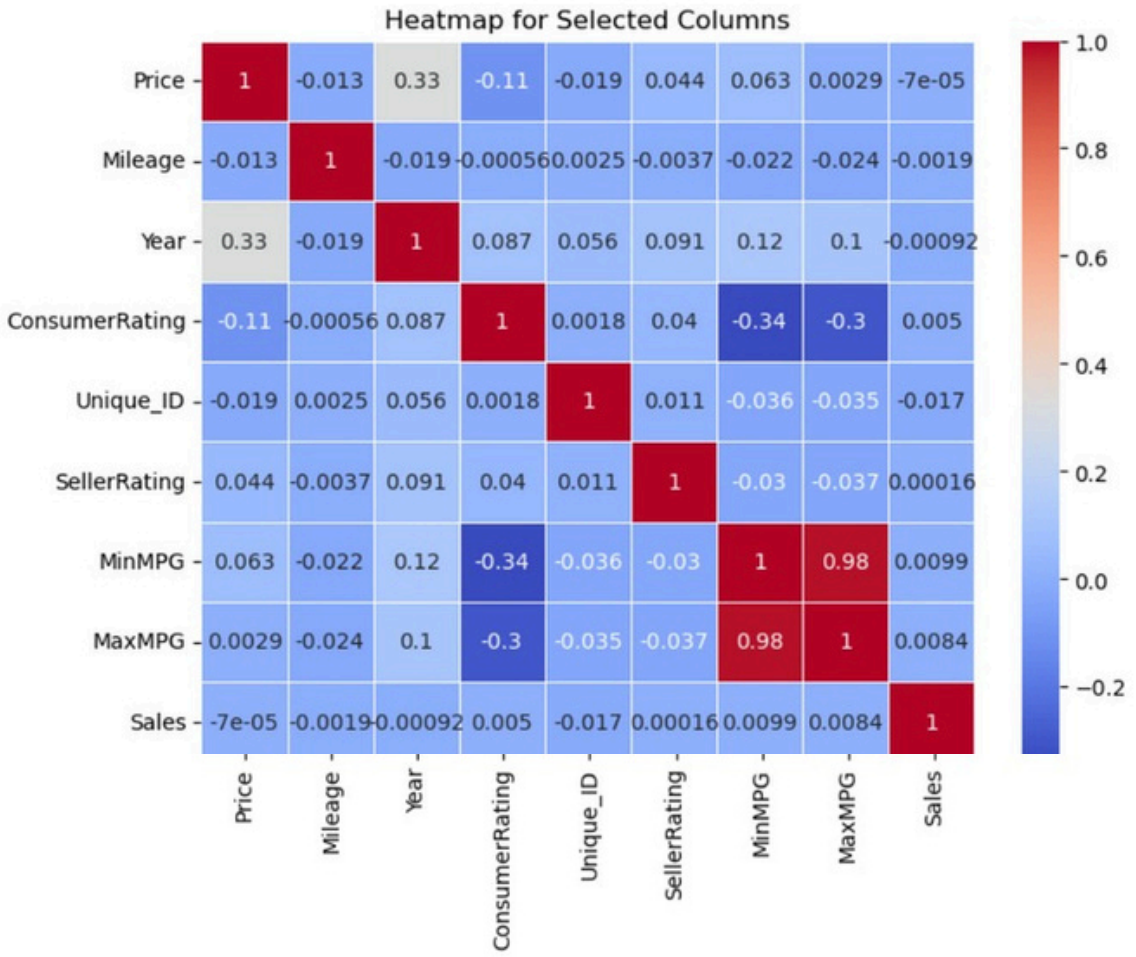
```
#regplot (it display also relationship between two continuous variables)
sns.regplot(y='Year',x= 'Sales', data=df)
```

```
<Axes: xlabel='Sales', ylabel='Year'>
```



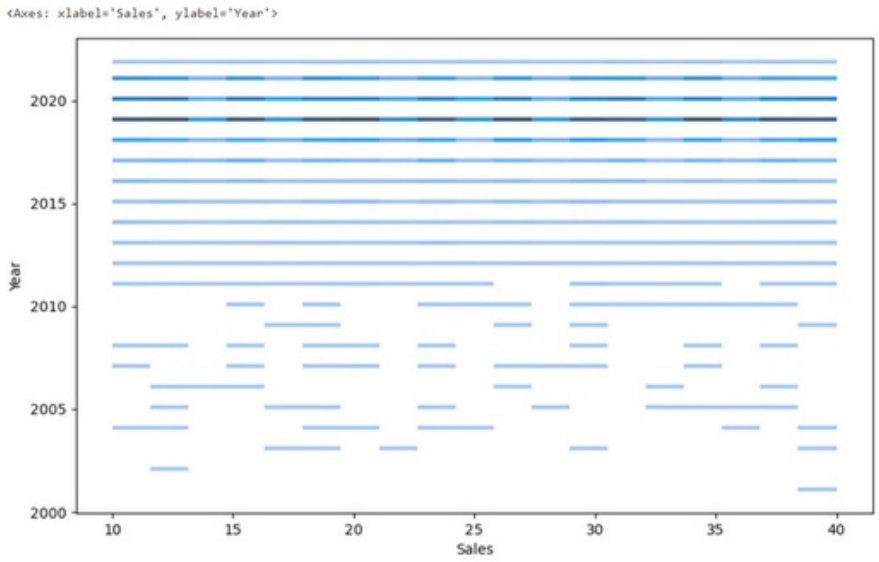
- This code selects specific columns from a dataset, calculates their correlations, and then visualizes those correlations in a heatmap

```
#sns.heatmap(df.corr(),annot=True)

# Example DataFrame (assuming df is already loaded)

# Step 1: Select the specific columns
selected_columns = ['Price', 'Mileage', 'Year','ConsumerRating','Unique_ID','SellerRating','MinMPG','MaxMPG','Sales']
subset_df = df[selected_columns]

# Step 2: Calculate the correlation matrix for the selected columns
corr_matrix = subset_df.corr()

# Step 3: Plot the heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Heatmap for Selected Columns')
plt.show()
```



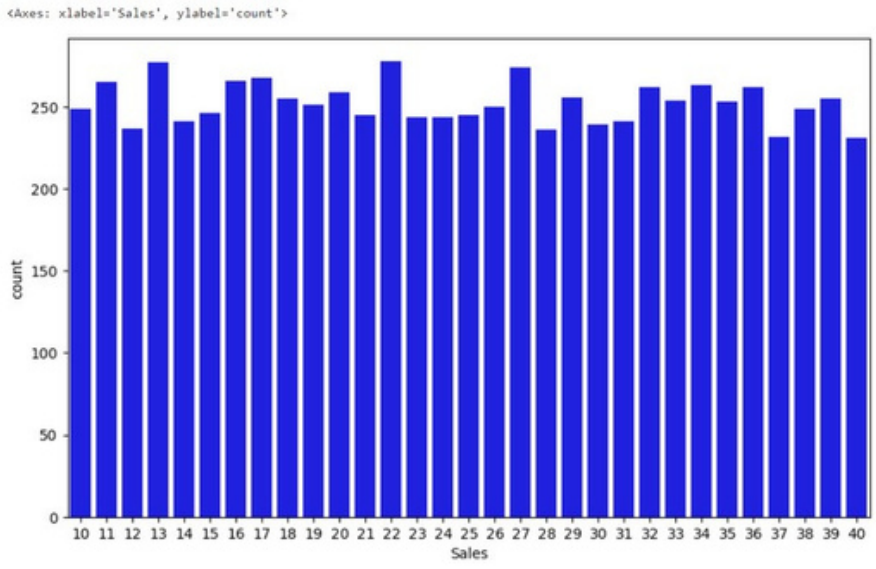
- It’s display also relationship between two continuous variables (Sales and Year) through histogram

```
#histogram (it display also relationship between two continuous variables)
plt.figure(figsize=(10, 6))
sns.histplot(x='Sales', y='Year', data=df)
```



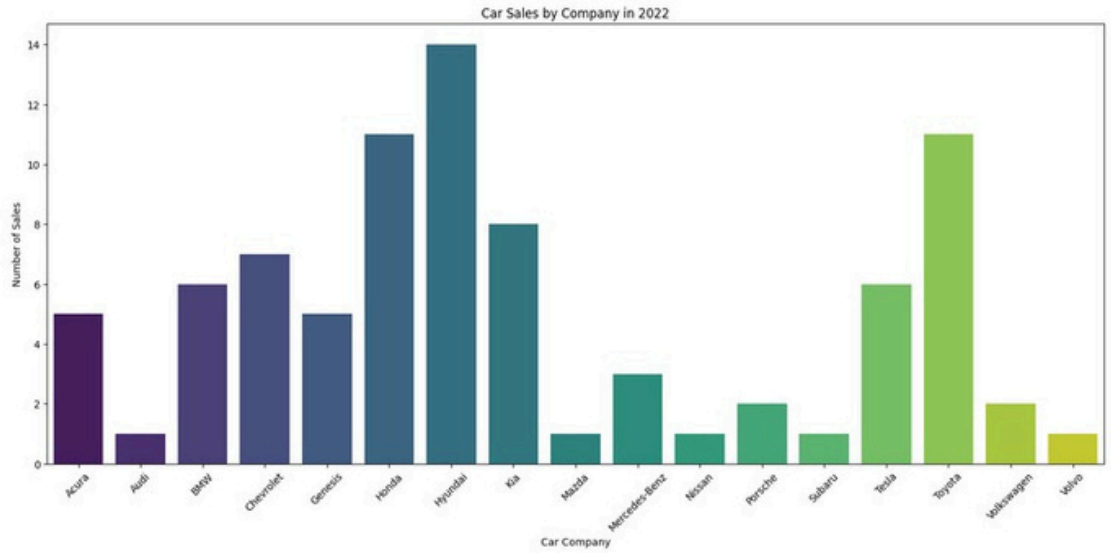
- It’s the visualization, Which describes how many sales from 10-40 are held by each and every companies cars and their models

```
#It's describes how many sales from 10-40 are held by companies
plt.figure(figsize=(10, 6))
sns.countplot(x='Sales',color='b', data=df)
```



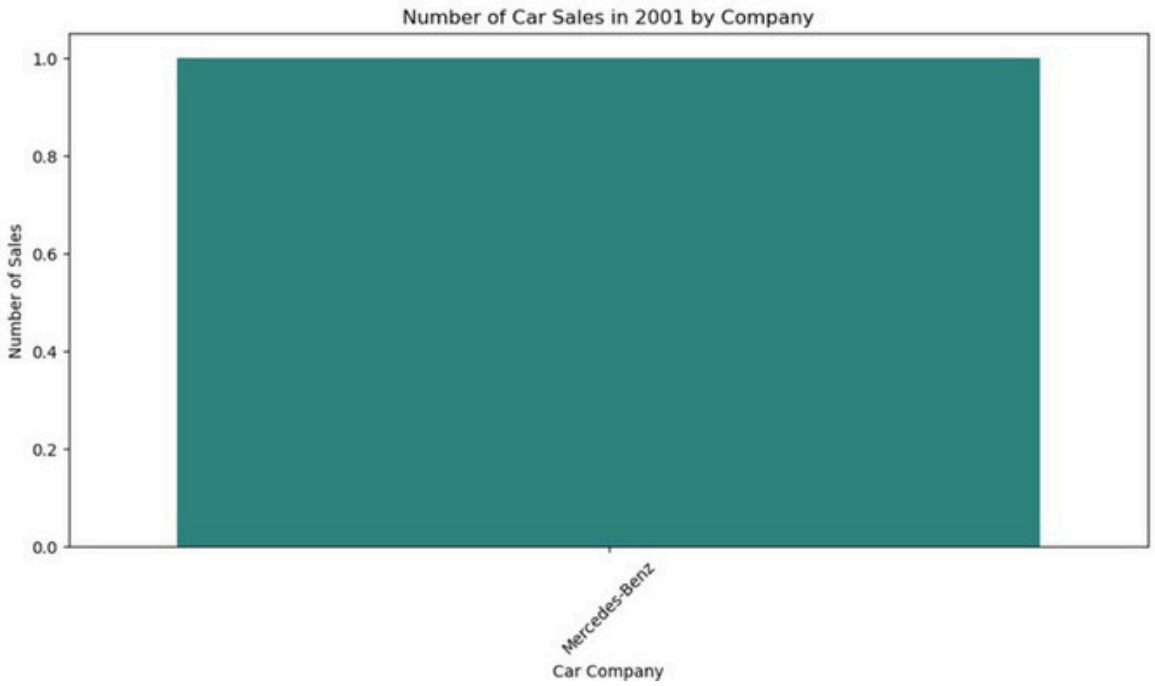
- The visualization which is describe how many cars were sold by every companies in last year (2022) and it's predict through barplot

```python
#Car Sales by Company in 2022(last year) predict through barplot
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
file_path = "VAM Dataset.csv"
df = pd.read_csv(file_path, encoding='ascii')

# Filter the data for the year 2022
cars_2022 = df[df['Year'] == 2022]

# Group by 'Make' and count the number of sales
sales_by_company_2022 = cars_2022.groupby('Make').size().reset_index(name='Sales')

# Plotting
plt.figure(figsize=(12, 8))
sns.barplot(data=sales_by_company_2022, x='Make', y='Sales', palette='viridis')
plt.title('Car Sales by Company in 2022')
plt.xlabel('Car Company')
plt.ylabel('Number of Sales')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

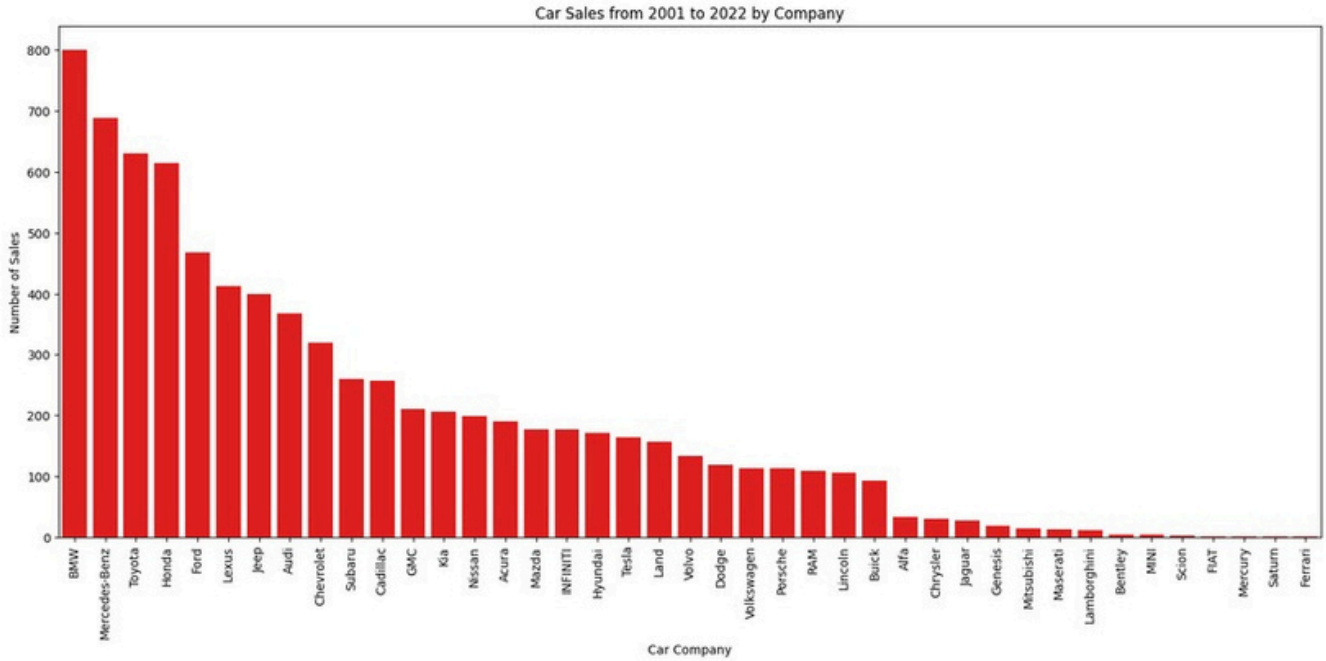


- The visualization which is describe how many cars were sold by every companies in first year (2001) and it's also predict through barplot

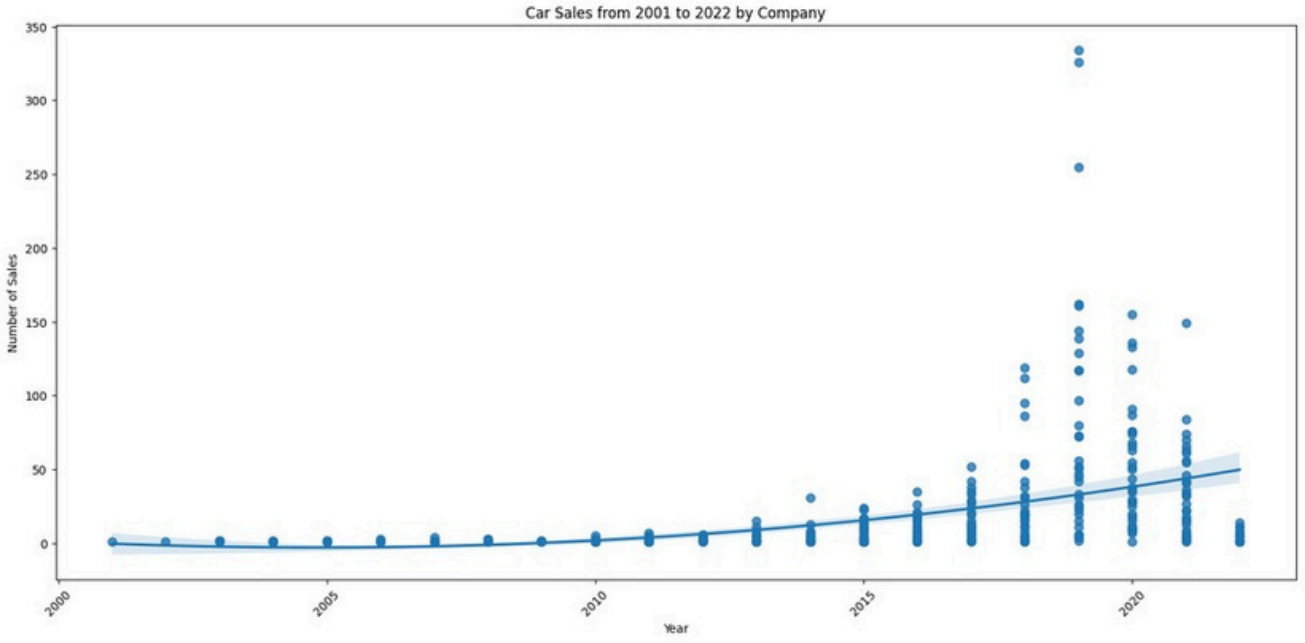
```python
#Car Sales by Company in 2001(First Year) predict through barplot
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
file_path = "VAM Dataset.csv"
df = pd.read_csv(file_path, encoding='ascii')

# Filter the data for the year 2001
cars_2001 = df[df['Year'] == 2001]

# Count the number of sales for each car company
sales_by_company = cars_2001['Make'].value_counts()

# Plot the data
plt.figure(figsize=(10, 6))
sns.barplot(x=sales_by_company.index, y=sales_by_company.values, palette='viridis')
plt.title('Number of Car Sales in 2001 by Company')
plt.xlabel('Car Company')
plt.ylabel('Number of Sales')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



- The visualization which is describe Car Sales from 2001 to 2022 by every Company through countplot

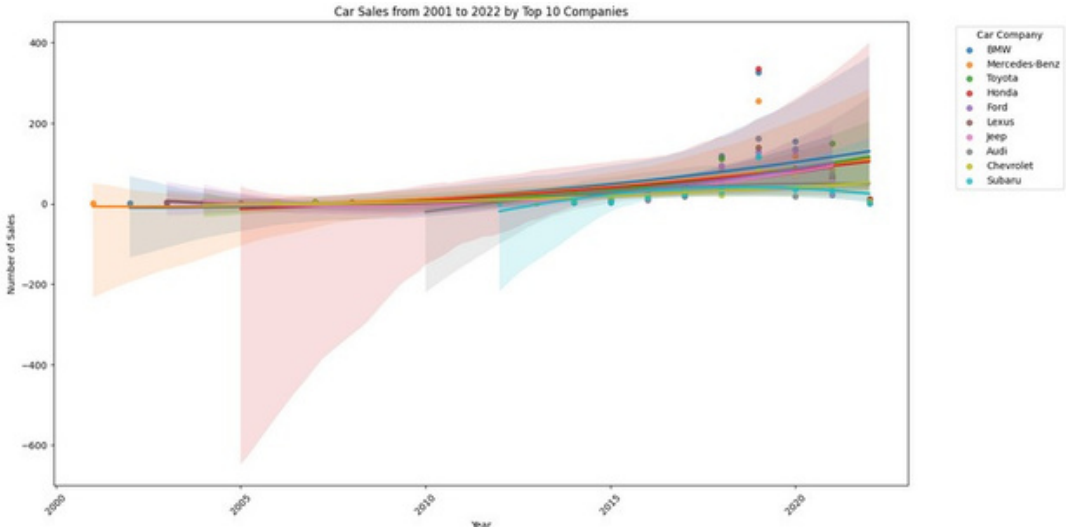


- The visualization which is describe Car Sales from 2001 to 2022 through regplot



- The visualization which is describe the Total car sales of top 10 companies from 2001 to 2022 through figure

```python
#It describe the Total sales for top 10 companies from 2001 to 2022 with figure
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
file_path = "VAM Dataset.csv"
df = pd.read_csv(file_path, encoding='ascii')

# Filter the data for years 2001 to 2022
df_filtered = df[(df['Year'] >= 2001) & (df['Year'] <= 2022)]

# Group by 'Year' and 'Make' to count the number of sales
sales_count = df_filtered.groupby(['Year', 'Make']).size().reset_index(name='Sales')

# Get the top 10 car companies by total sales
top_companies = sales_count.groupby('Make')['Sales'].sum().nlargest(10).index

# Filter the data for top 10 companies
sales_count_top10 = sales_count[sales_count['Make'].isin(top_companies)]

# Plotting
plt.figure(figsize=(16, 10))
for company in top_companies:
    company_data = sales_count_top10[sales_count_top10['Make'] == company]
    sns.regplot(data=company_data, x='Year', y='Sales',
                scatter_kws={'s': 30}, label=company, order=2)

plt.title('Car Sales from 2001 to 2022 by Top 10 Companies')
plt.xlabel('Year')
plt.ylabel('Number of Sales')
plt.legend(title='Car Company', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Print the total sales for each company
total_sales = sales_count_top10.groupby("Make")['Sales'].sum().sort_values(ascending=False)
print("Total sales for top 10 companies from 2001 to 2022:")
print(total_sales)
```




```
Total sales for top 10 companies from 2001 to 2022:
Make
BMW              800
Mercedes-Benz    689
Toyota           630
Honda            614
Ford             468
Lexus            413
Jeep             400
Audi             367
Chevrolet        320
Subaru           260
Name: Sales, dtype: int64
```

- This program is to prepare the data for machine learning modeling. Specifically, it focuses on:
- Loading and understanding the dataset by displaying its structure. Defining features and the
- target variable (predicting 'Sales' based on other variables). Splitting the data into training and testing
- sets, which is essential for training a machine learning model (using 80% of the data for training and 20% for testing). This sets the foundation for building a model to predict Sales based on the
- features in the dataset.

```python
import pandas as pd
from sklearn.model_selection import train_test_split

# Load the dataset
file_path = "VAM Dataset.csv"
df = pd.read_csv(file_path, encoding='ascii')

# Display the head of the dataframe to understand its structure
print(df.head())

# Define features and target variable
# Assuming 'Price' is the target variable and the rest are features
X = df.drop('Sales', axis=1)
y = df['Sales']

# 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)

# Display the shapes of the resulting datasets
print('X_train shape:', X_train.shape)
print('X_test shape:', X_test.shape)
print('y_train shape:', y_train.shape)
print('y_test shape:', y_test.shape)
```

```
   Unique_ID  Year    Make            Model  Price  ConsumerRating  \
0       1001  2019  Toyota        Sienna SE  39998             4.6
1       1002  2018    Ford     F-150 Lariat  49985             4.8
2       1003  2017     RAM     1500 Laramie  41860             4.7
3       1004  2021   Honda  Accord Sport SE  28500             5.0
4       1005  2020   Lexus           RX 350  49000             4.8

   SellerRating  MinMPG  MaxMPG  Mileage  Sales
0           3.3      19      27    29403     14
1           4.8      19      24    32929     20
2           4.6      15      21    23173     39
3           4.6      29      35    10598     38
4           4.8      20      27    28137     36
X_train shape: (6261, 10)
X_test shape: (1566, 10)
y_train shape: (6261,)
y_test shape: (1566,)
```

- This code is to train a linear regression model to predict a target variable based on the independent features. It's to train a linear regression model that
- can predict values (such as sales) based on given input features.

```
from sklearn.model_selection import train_test_split

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
from sklearn.linear_model import LinearRegression

# Initialize the model
model = LinearRegression()

# Train the model
model.fit(X_train, y_train)
```

```
LinearRegression
LinearRegression()
```

- This program is to train, evaluate, and save a Decision Tree model for a classification task using the VAM Dataset.csv. It's to develop a decision tree
- classifier, evaluate its performance, and persist the model for future predictions.

```
from sklearn.tree import DecisionTreeRegressor

# Initialize the model
model = DecisionTreeRegressor()

# Train and evaluate as above
```

```
import joblib

# Save the model to a file
joblib.dump(model, 'regression_model.pkl')

# Load the model from the file
loaded_model = joblib.load('regression_model.pkl')
```

```python
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report

# Load the dataset
file_path = "VAM Dataset.csv"
df = pd.read_csv(file_path)


# Display the head of the dataframe to understand its structure
print(df.head())
print()
# Prepare the data for the decision tree model
# Drop non-numeric columns and the target column 'Sales'
X = df.drop(columns=['Unique_ID', 'Make', 'Model', 'Sales'])
y = df['Sales']

# 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)

# Initialize the Decision Tree Classifier
clf = DecisionTreeClassifier(random_state=42)

# Train the model
clf.fit(X_train, y_train)

# Predict on the test set
y_pred = clf.predict(X_test)
```

```python
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)*100
report = classification_report(y_test, y_pred)

print('Accuracy:', accuracy)
print('Classification Report:\
', report)
```

```
   Unique_ID  Year    Make            Model  Price  ConsumerRating  \
0       1001  2019  Toyota        Sienna SE  39998             4.6
1       1002  2018    Ford     F-150 Lariat  49985             4.8
2       1003  2017     RAM     1500 Laramie  41860             4.7
3       1004  2021   Honda  Accord Sport SE  28500             5.0
4       1005  2020   Lexus           RX 350  49000             4.8

   SellerRating  MinMPG  MaxMPG  Mileage  Sales
0           3.3      19      27    29403     14
1           4.8      19      24    32929     20
2           4.6      15      21    23173     39
3           4.6      29      35    10598     38
4           4.8      20      27    28137     36

Accuracy: 3.3205619412515963
Classification Report:               precision    recall  f1-score   support

          10       0.05      0.04      0.05        45
          11       0.02      0.02      0.02        45
          12       0.02      0.02      0.02        52
          13       0.02      0.02      0.02        47
          14       0.00      0.00      0.00        52
          15       0.06      0.07      0.06        42
          16       0.03      0.03      0.03        63
          17       0.05      0.04      0.05        46
          18       0.04      0.04      0.04        54
          19       0.02      0.02      0.02        47
          20       0.06      0.05      0.06        58
          21       0.02      0.02      0.02        52
          22       0.04      0.03      0.03        63
          23       0.05      0.04      0.04        51
          24       0.00      0.00      0.00        46
          25       0.05      0.06      0.05        50
          26       0.00      0.00      0.00        50
          27       0.03      0.03      0.03        64
          28       0.07      0.07      0.07        43
          29       0.04      0.07      0.05        44
          30       0.04      0.05      0.04        42
          31       0.02      0.02      0.02        43
          32       0.04      0.03      0.04        58
          33       0.00      0.00      0.00        40
          34       0.10      0.09      0.10        53
          35       0.05      0.03      0.04        60
          36       0.00      0.00      0.00        59
          37       0.00      0.00      0.00        46
          38       0.03      0.04      0.03        57
          39       0.07      0.05      0.06        55
          40       0.03      0.03      0.03        39

    accuracy                           0.03      1566
   macro avg       0.03      0.03      0.03      1566
weighted avg       0.03      0.03      0.03      1566
```

- This program is to train and evaluate a Random Forest Regressor model to predict the Sales variable using other features from the dataset.
- It is to predict the Sales values using a Random Forest regression model and assess its performance based on the MSE.

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

# Load the dataset
file_path = "VAM Dataset.csv"
df = pd.read_csv(file_path)

# Display the head of the dataframe to understand its structure
print(df.head())
# Prepare the data for training
# Drop non-numeric columns and the target column 'Price'
X = df.drop(columns=['Unique_ID', 'Make', 'Model', 'Price'])
y = df['Sales']

# 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)

# Initialize the Random Forest Regressor
rf = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the model
rf.fit(X_train, y_train)

# Predict on the test set
y_pred = rf.predict(X_test)

# Calculate the mean squared error
mse = mean_squared_error(y_test, y_pred)

print('Random Forest model trained successfully.')
print('Mean Squared Error on test set:', mse)
```

```
   Unique_ID  Year    Make            Model  Price  ConsumerRating  \
0       1001  2019  Toyota        Sienna SE  39998             4.6
1       1002  2018    Ford     F-150 Lariat  49985             4.8
2       1003  2017     RAM     1500 Laramie  41860             4.7
3       1004  2021   Honda  Accord Sport SE  28500             5.0
4       1005  2020   Lexus           RX 350  49000             4.8

   SellerRating  MinMPG  MaxMPG  Mileage  Sales
0           3.3      19      27    29403     14
1           4.8      19      24    32929     20
2           4.6      15      21    23173     39
3           4.6      29      35    10598     38
4           4.8      20      27    28137     36
Random Forest model trained successfully.
Mean Squared Error on test set: 0.0
```

- This program is to build and evaluate a Linear Regression model to predict a target variable (likely ‘Sales’ or another continuous variable) based on other features. Additionally, the program makes specific predictions using the trained model. It’s to build a Linear Regression
- model, evaluate its performance, and make specific predictions based on user-defined inputs.

```
#Linear Regression
# building model
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor
regressor.fit(X_train,y_train)
y_pred2 = regressor.predict(X_test)
y_pred2
# accuracy score
from sklearn.metrics import r2_score
score= r2_score(y_pred2,y_test) * 100
print(score2)
Line1 = dt1.predict([[2019,4.6,3.3,19,27,29403, 14]])
Line2 = dt1.predict([[2018,4.8,3.3,19,24,32929, 20]])
Line3 = dt1.predict([[2017,4.7,4.6,15,24,23173, 39]])
print (Line1,Line2,Line3)
```

```
100.0
[14.] [20.] [39.]
```

- This program is to train, evaluate, and make predictions using a Random Forest Regressor model. It’s to train and evaluate a Random
- Forest model to predict Sales or another target variable and make specific predictions for given inputs.

```
from sklearn.ensemble import RandomForestRegressor
dt1 = RandomForestRegressor()
dt1
dt1.fit(X_train,y_train)
y_pred3=dt1.predict(X_test)
y_pred3
from sklearn.metrics import r2_score
score2 = r2_score(y_pred3,y_test) * 100
print(score)

Ran1 = dt1.predict([[2019,4.6,3.3,19,27,29403, 14]])
Ran2 = dt1.predict([[2018,4.8,3.3,19,24,32929, 20]])
Ran3 = dt1.predict([[2017,4.7,4.6,15,24,23173, 39]])
print (Ran1,Ran2,Ran3)
```

```
100.0
[14.] [20.] [39.]
```

- This program is to train, evaluate, and make predictions using a Decision Tree Regressor model.
- It's to build a Decision Tree Regressor model, evaluate its performance, and make predictions for specific feature sets to understand how the model predicts outcomes based on those inputs.

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score
dt1 = DecisionTreeRegressor()
dt1.fit(X_train, y_train)
y_pred3 = dt1.predict(X_test)
score3 = r2_score(y_test, y_pred3) * 100
print(score3)

Dec1 = dt1.predict([[2019,4.6,3.3,19,27,29403, 14]])
Dec2 = dt1.predict([[2018,4.8,3.3,19,24,32929, 20]])
Dec3 = dt1.predict([[2017,4.7,4.6,15,24,23173, 39]])
print (Dec1,Dec2,Dec3)
```

```
100.0
[14.] [20.] [39.]
```

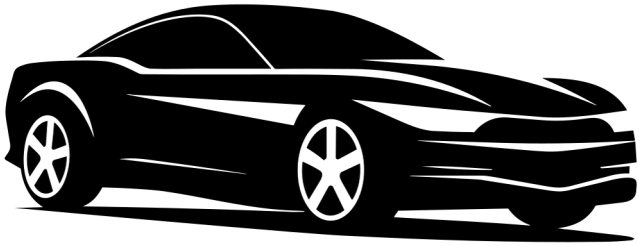
- This code is to create and display a data frame that compares the performance and predictions of different machine learning algorithms.
- This data frame provides a summary of the performance and specific predictions made by each algorithm, allowing for an easy comparison of their results.

```
import pandas as pd

# Creating the DataFrame with the given data
comparison_df = pd.DataFrame({
    "Algorithm": ["Linear Regressor", "Decision_Tree", "Random-Forest"],
    "Accuracy": [score,score2,score3],
    "Prediction-1": [Line1,Dec1,Ran1],
    "Prediction-2": [Line2,Dec2,Ran2],
    "Prediction-3": [Line3,Dec3,Ran3]})
# Display the DataFrame
comparison_df.head()
```

	Algorithm	Accuracy	Prediction-1	Prediction-2	Prediction-3
0	Linear Regressor	100.0	[14.0]	[20.0]	[39.0]
1	Decision_Tree	100.0	[14.0]	[20.0]	[39.0]
2	Random-Forest	100.0	[14.0]	[20.0]	[39.0]

CONCLUSION

The research emphasizes the necessity of knowing Car sales trends over time, as well as the efficacy of machine learning models in projecting sales based on prior data. This technique might be valuable for automobile decision-making, allowing corporations to forecast future sales and strategy accordingly.



THANK YOU