DATA CLEANING USING PYTHON

Presented By:
Bhawna Gupta

Loading CSV File

```
import pandas as pd;
df=pd.read_csv("C:/Users/Folder/Desktop/Projects/Python/Cars Data/Cars Data From Kaggle.csv")
```

Viewing First 10 Rows

```
df.head(10)
```

	S.No.	car_name	car_prices_in_rupee	kms_driven	fuel_type	transmission	ownership	manufacture	engine	Seats	Product Number
0	0	Jeep Compass 2.0 Longitude Option BSIV	10.03 Lakh	86,226 kms	Diesel	Manual	1st Owner	1/1/2017	1956 cc	5.0	62837.0
1	1	Renault Duster RXZ Turbo CVT	12.83 Lakh	13,248 kms	Petrol	Automatic	1st Owner	1/1/2021	1330 cc	5.0	34629.0
2	2	Toyota Camry 2.5 G	16.40 Lakh	60,343 kms	Petrol	Automatic	1st Owner	1/1/2016	2494 cc	5.0	64168.0
3	3	Honda Jazz VX CVT	7.77 Lakh	26,696 kms	Petrol	Automatic	1st Owner	1/1/2018	1199 cc	5.0	52540.0
4	4	Volkswagen Polo 1.2 MPI Highline	5.15 Lakh	69,414 kms	Petrol	Manual	1st Owner	1/1/2016	1199 cc	5.0	63117.0
5	5	Volkswagen Vento 1.2 TSI Highline AT	7.66 Lakh	49,719 kms	Petrol	Automatic	1st Owner	1/1/2017	1197 cc	5.0	18798.0
6	6	Volkswagen Vento 1.2 TSI Highline Plus AT	7.58 Lakh	43,688 kms	Petrol	Automatic	1st Owner	1/1/2017	1197 cc	5.0	76782.0
7	7	Honda WR-V VX Diesel	11.60 Lakh	14,470 kms	Diesel	Manual	1st Owner	1/1/2021	1498 cc	5.0	48059.0
8	8	Honda City i VTEC CVT SV	6.99 Lakh	21,429 kms	Petrol	Automatic	1st Owner	1/1/2015	1497 cc	5.0	18113.0
9	9	Renault Duster Petrol RXS CVT	7.53 Lakh	31,750 kms	Petrol	Automatic	1st Owner	1/1/2017	1498 cc	5.0	40454.0

Viewing Non-Null Count And Datatypes

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5512 entries, 0 to 5511
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   S.No.                 5512 non-null   int64  
 1   car_name              5512 non-null   object 
 2   car_prices_in_rupee   5512 non-null   object 
 3   kms_driven            5505 non-null   object 
 4   fuel_type             5501 non-null   object 
 5   transmission          5501 non-null   object 
 6   ownership             5508 non-null   object 
 7   manufacture           5512 non-null   object 
 8   engine                5507 non-null   object 
 9   Seats                 5485 non-null   float64
 10  Product Number        5497 non-null   float64
dtypes: float64(2), int64(1), object(8)
memory usage: 473.8+ KB
```

Viewing Null Values Percentage

```
df.isnull().sum()/df.shape[0]*100
```

```
S.No.                0.000000
car_name             0.000000
car_prices_in_rupee  0.000000
kms_driven           0.126996
fuel_type            0.199565
transmission         0.199565
ownership            0.072569
manufacture          0.000000
engine               0.090711
Seats                0.489840
Product Number       0.272134
dtype: float64
```

Filling The NA Values

```
df['kms_driven'].fillna(method='ffill', inplace=True, limit=None)
df.fillna({'fuel_type':'Diesel'}, inplace=True)
df.fillna({'transmission':'Automatic'}, inplace=True)
df.fillna({'ownership':'2nd Owner'}, inplace=True)
df['engine'].fillna(method='bfill',inplace=True)
df['Seats'].fillna(df['Seats'].mean(), inplace=True)
```

```
#KNNImputer
from sklearn.impute import KNNImputer
impute=KNNImputer()
for i in df.select_dtypes(include='number').columns:
    df[i]=impute.fit_transform(df[[i]])
```

Viewing Null Values Percentage Again

```
df.isnull().sum()/df.shape[0]*100
```

```
S.No.                0.0
car_name             0.0
car_prices_in_rupee  0.0
kms_driven           0.0
fuel_type            0.0
transmission         0.0
ownership            0.0
manufacture          0.0
engine               0.0
Seats                0.0
Product Number       0.0
dtype: float64
```

Separating Company Name And Product Name In Car_Name

```
def com_name(x):
    return x[:x.index(" ")]

df['com_name']=df['car_name'].apply(com_name)
df['com_name']
```

```
def prod_name(x):
    return x[x.index(" ")+1:]

df['prod_name']=df['car_name'].apply(prod_name)
df['prod_name']
```

Replacing Commas And Converting Data In Same Units (Numbers)

```
df['car_prices_in_rupee']=df['car_prices_in_rupee'].str.replace(',','')
def unit_change(x):
    p=x.split(' ')

    try:
        if p[1]=='Lakh':
            return str(float(p[0])*100000)
        elif p[1]=='Crore':
            return str(float(p[0])*10000000)
    except:
        return x

df['car_prices_in_rupee']=df['car_prices_in_rupee'].apply(unit_change)

df['car_prices_in_rupee']=df['car_prices_in_rupee'].astype('float64')
```

Replacing Commas And Units Of Other Columns

```
df['kms_driven']=df['kms_driven'].str.replace(',','')
df['kms_driven']=df['kms_driven'].str.replace(' kms','')
df['kms_driven']=df['kms_driven'].astype('int64')

df['engine']=df['engine'].str.replace(' cc','')
df['engine']=df['engine'].astype('int64')

df['engine']=df['engine'].astype('int64')
```

Converting Date Format And Extracting Year

```
df['manufacture']=pd.to_datetime(df['manufacture'])

df['Year']=df['manufacture'].dt.year
df.head()
```

Dropping unnecessary/duplicate columns.

```
df.drop(['car_name','manufacture'], axis=1,inplace=True)
```

Viewing The DataFrame

```
df.head(10)
```

	S.No.	car_prices_in_rupee	kms_driven	fuel_type	transmission	ownership	engine	Seats	Product Number	com_name	prod_name	Year
0	0.0	1003000.0	86226	Diesel	Manual	1st Owner	1956	5.0	62837.0	Jeep	Compass 2.0 Longitude Option BSIV	2017
1	1.0	1283000.0	13248	Petrol	Automatic	1st Owner	1330	5.0	34629.0	Renault	Duster RXZ Turbo CVT	2021
2	2.0	1640000.0	60343	Petrol	Automatic	1st Owner	2494	5.0	64168.0	Toyota	Camry 2.5 G	2016
3	3.0	777000.0	26696	Petrol	Automatic	1st Owner	1199	5.0	52540.0	Honda	Jazz VX CVT	2018
4	4.0	515000.0	69414	Petrol	Manual	1st Owner	1199	5.0	63117.0	Volkswagen	Polo 1.2 MPI Highline	2016
5	5.0	766000.0	49719	Petrol	Automatic	1st Owner	1197	5.0	18798.0	Volkswagen	Vento 1.2 TSI Highline AT	2017
6	6.0	758000.0	43688	Petrol	Automatic	1st Owner	1197	5.0	76782.0	Volkswagen	Vento 1.2 TSI Highline Plus AT	2017
7	7.0	1160000.0	14470	Diesel	Manual	1st Owner	1498	5.0	48059.0	Honda	WR-V VX Diesel	2021
8	8.0	699000.0	21429	Petrol	Automatic	1st Owner	1497	5.0	18113.0	Honda	City i VTEC CVT SV	2015
9	9.0	753000.0	31750	Petrol	Automatic	1st Owner	1498	5.0	40454.0	Renault	Duster Petrol RXS CVT	2017

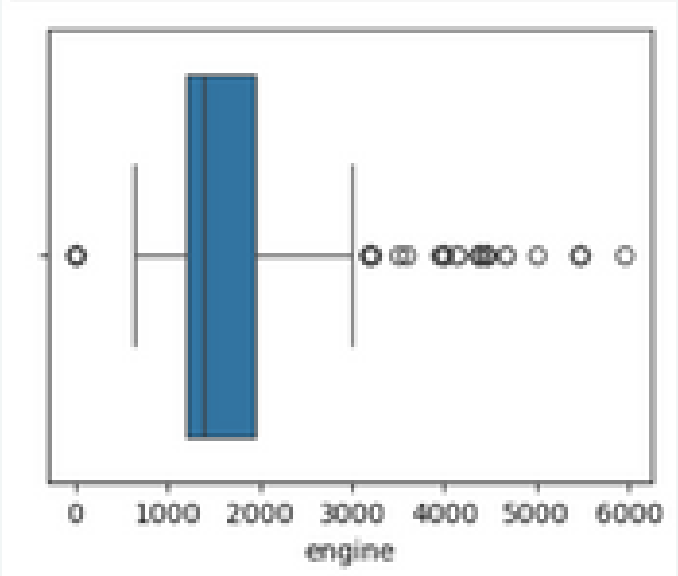
Exploratory Data Analysis (EDA)

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
S.No.	5512.0	2.755500e+03	1.591322e+03	0.0	1377.75	2755.500000	4133.25	5511.0
car_prices_in_rupee	5512.0	1.162510e+06	1.969919e+06	35000.0	315000.00	550000.000000	1025750.00	19200000.0
kms_driven	5512.0	6.314705e+04	4.183638e+04	250.0	33000.00	59000.000000	84226.00	560000.0
engine	5512.0	1.531607e+03	5.791745e+02	0.0	1197.00	1396.000000	1950.00	5950.0
Seats	5512.0	5.247949e+00	7.156885e-01	2.0	5.00	5.000000	5.00	8.0
Product Number	5512.0	4.615872e+04	1.967625e+04	12396.0	29047.75	46158.724759	63200.25	79994.0
Year	5512.0	2.015456e+03	3.927974e+00	1996.0	2013.00	2016.000000	2018.00	2022.0

Boxplot to understand outliers

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

plt.figure(figsize=(4,3))
sns.boxplot(x='engine', data=df)
plt.show()
```

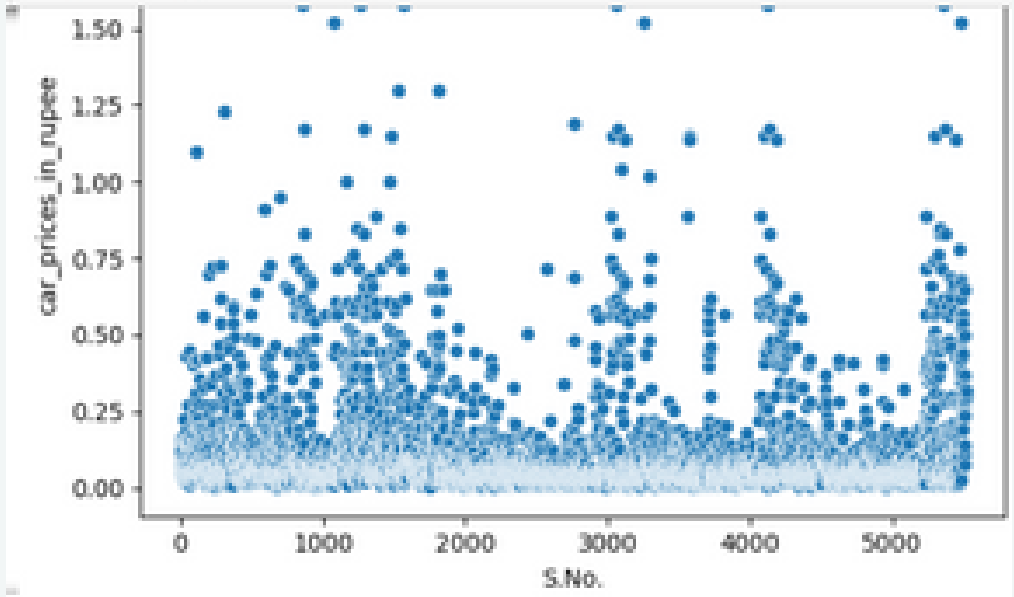


Removing Outlier

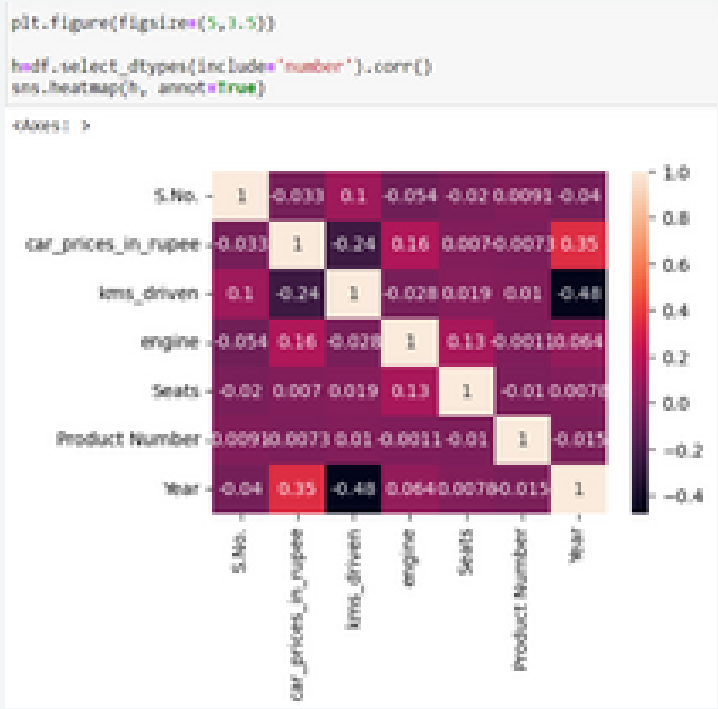
```
import numpy as np
q1=np.quantile(df['engine'],0.25)
q3=np.quantile(df['engine'],0.75)

iqr=q3-q1

min=q1-(1.5*iqr)
max=q3+(1.5*iqr)

df=df[df['engine']>min]
df=df[df['engine']<max]
```

Scatterplot For Relationship Between Two Continuous Variables

```
df.select_dtypes(include='number').columns

for i in ['S.No.', 'kms_driven', 'engine', 'Seats',
          'Product Number', 'Year']:
    sns.scatterplot(x=i,y='car_prices_in_rupee', data=df)
    plt.show()
```



Heatmap For Correlation Between Multiple Variables As A Color-Coded Matrix



Encoding Of Data Using Pandas get_dummies()

```
df_e=pd.get_dummies(data=df,columns=['fuel_type','transmission'], drop_first=True)
print(df_e)
```

```
      S.No.  car_prices_in_rupee  kms_driven  ownership  engine  Seats  \
0       0.0            1003000.0       86226  1st Owner    1956    5.0
1       1.0            1283000.0       13248  1st Owner    1330    5.0
2       2.0            1648000.0       60343  1st Owner    2494    5.0
3       3.0             777000.0       26696  1st Owner    1199    5.0
4       4.0             515000.0       69414  1st Owner    1199    5.0
...     ...                  ...         ...        ...     ...    ...
5507  5507.0            2890000.0       45000  1st Owner    2995    7.0
5508  5508.0            6490000.0       29000  2nd Owner    1968    5.0
5509  5509.0            1375000.0       90000  2nd Owner    2755    5.0
5510  5510.0            2990000.0       79000  3rd Owner    2967    6.0
5511  5511.0            3190000.0       42000  2nd Owner    1991    5.0

      Product Number    com_name                        prod_name  Year  \
0            62837.0        Jeep  Compass 2.0 Longitude Option BSIV  2017
1            34629.0     Renault              Duster RXZ Turbo CVT  2021
2            64168.0      Toyota                        Camry 2.5 G  2016
3            52540.0       Honda                       Jazz VX CVT  2018
4            63117.0  Volkswagen             Polo 1.2 MPI Highline  2016
```

Encoding Of Data Using Sklearn LabelEncoder

```
from sklearn.preprocessing import LabelEncoder
df_e=LabelEncoder()
df['fuel_type']=df_e.fit_transform(df['fuel_type'])
df
```

	S.No.	car_prices_in_rupee	kms_driven	fuel_type	transmission	ownership	engine	Seats	Product Number	com_name	prod_name	Year
0	0.0	1003000.0	86226	1	Manual	1st Owner	1956	5.0	62837.0	Jeep	Compass 2.0 Longitude Option BSIV	2017
1	1.0	1283000.0	13248	4	Automatic	1st Owner	1330	5.0	34629.0	Renault	Duster RXZ Turbo CVT	2021
2	2.0	1640000.0	60343	4	Automatic	1st Owner	2494	5.0	64168.0	Toyota	Camry 2.5 G	2016
3	3.0	777000.0	26696	4	Automatic	1st Owner	1199	5.0	52540.0	Honda	Jazz VX CVT	2018
4	4.0	515000.0	69414	4	Manual	1st Owner	1199	5.0	63117.0	Volkswagen	Polo 1.2 MPI Highline	2016

Key Results

- **Data Completeness:** Understanding the extent of missing values in the dataset and how they were handled. Ensuring a complete dataset for analysis without significant gaps.
- **Data Quality:** Addressing outliers, and inconsistencies, to ensure that the data is of higher quality and more reliable for analysis. Reducing the likelihood of errors and misleading insights.
- **Data Consistency:** Resolving inconsistencies or discrepancies in the dataset, such as conflicting values or formats to ensure that the data is consistent and can be interpreted correctly.
- **Improved Analysis**: Conducting more accurate and meaningful analyses with a clean dataset. Trusting the results and insights obtained from the cleaned data to make informed decisions.
- **Visualization**: Creating clear and informative plots and charts to explore patterns, trends, and relationships in the data. Cleaned data is more suitable for high quality visualizations.