Experiment no.:9 *Date: 18-04-2024*

Experiment 9: Dataset Analysis and Quality Assurance Tool.

Aim: To provide essential insights into the dataset, including descriptive statistics, missing value detection, and unique value counting in the 'FuelType' column, facilitating informed analysis and decision-making.

Dataset : 'Toyota.xlsx'

Software Required: Google Colab.

Description:

1. **Descriptive Statistics Calculation**: The program calculates various descriptive statistics for numeric columns in the dataset. This includes statistics such as count, mean, standard deviation, minimum, maximum, and quartiles (25th, 50th, and 75th percentiles).
2. **Missing Values Check**: The program checks for missing values in each column of the dataset. It iterates through each column and counts the number of missing values, if any.
3. **Unique Values Count**: It counts the number of unique values in the 'FuelType' column. This provides insight into the distribution of different types of fuel in the dataset.
4. **Output Formatting**: The program formats the results neatly, likely for display or further analysis.
5. **Theoretical Purpose**: This program could be part of data preprocessing or exploratory data analysis (EDA) tasks. Descriptive statistics help in understanding the distribution and characteristics of the dataset, while checking for missing values ensures data completeness. Counting unique values provides insight into categorical variables.

Program:

```
cars_data=pd.read_excel('Toyota.xlsx', index_col=0)
cars_data.replace(["??", "###", "????"], np.nan, inplace=True)
# cleaning Doors column # checking the unique values of variable Doors
cars_data1=cars_data.copy (deep=True)
print(pd.unique(cars_data1 ['Doors']))
# ['2' '3' '4' '5' 'five' 'four' 'three'] there are strings along with
integer values
# use replace() to replace a value with desired value
cars_data1['Doors'].replace('three', 3, inplace=True)
cars_data1['Doors'].replace('four', 4, inplace=True)
cars_data1['Doors'].replace('five', 5, inplace=True)
# converting Doors to int64
cars_data1['Doors'] = cars_data1['Doors'].astype('int64')
# rechecking the Doors column for datatype and unique values
print(np.unique(cars_data1 ['Doors']))
```

```
print(cars_data1.info())
# removing row/columns with nan values.
#removing all rows with missing values
cars1=cars_data1.dropna (axis=0, how='any')
print(cars1.info())
#removing rows with all nan values
cars1=cars_data1.dropna (axis=0, how='all')
print(cars1.info())
#removing all columns with missing values
cars2=cars_data1.dropna (axis=1, how='any')
print(cars2.info())
#removing columns with all nan values
cars2=cars_data1.dropna (axis=1, how='all')
print(cars2.info())
#Handling the missing data/ nan values
cars_data2=cars_data1.copy()
cars_data3=cars_data1.copy()
#To check the count of missing values present in each column
Dataframe.isnull.sum () is used
print('count of missing values in each column:\n',
cars_data2.isnull().sum())
print('count of missing values in each column:\n',
cars_data2.isna().sum())
# Subsetting the rows that have one or more missing values
missing=cars_data2 [cars_data2.isnull().any(axis=1)]
# imputing the missing values
# Two ways of approach to fill the missing data
''' 1. Fill the missing values by mean / median, in case of numerical
variable
2. Fill the missing values with the class which has maximum count
in case of categorical variable '''
'''Using DataFrame.describe() Generate descriptive statistics like
count, mean, std, min,
25%, 50%, 75%, and max that summarize the central tendency,
dispersion and shape of a dataset's distribution, excluding NaN
values'''
# pd.set_option('max_columns', 50)
c_d=cars_data2.describe()
print(c_d)
# imputing the missing values with mean/median for numerical datatype
'''The mean and median of Age variable is almost same. hence replace
the missing values with mean'''
Age_mean=cars_data2['Age'].mean()
cars_data2['Age'].fillna (cars_data2 ['Age'].mean(), inplace=True)
print('count of missing values in Age column: \n',
cars_data2['Age'].isnull().sum())
''' The mean and median of KM variable are very far from each other.
This is because of high extreme values present.
```

```
Hence use median value to replace the missing values'''
KM_median=cars_data2['KM'].median()
cars_data2['KM'].fillna (cars_data2['KM'].median(), inplace=True)
print('count of missing values in KM column: \n', cars_data2
['KM'].isnull().sum())
'''The mean and median of HP variable is almost same.
hence replace the missing values with mean'''
HP_mean=cars_data2['HP'].mean()
cars_data2['HP'].fillna (cars_data2['HP'].mean(), inplace=True)
print('count of missing values in HP column: \n', cars_data2
['HP'].isnull().sum())
# Imputing missing values of 'FuelType'
'''Series.value_counts () Returns a Series containing counts of unique
values.
The values will be in descending order so that the first element is the
most frequently occurring element
• Excludes NA values by default'''
print('count of unique values from FuelType column: \n',
cars_data2['FuelType'].value_counts())
'''As frequency of petrol category is more when compared to others,
replace the missing values with Petrol category'''
cars_data2['FuelType'].fillna (cars_data2
['FuelType'].value_counts().index [0], inplace=True)
print('count of missing values in Fueltype column:\n', cars_data2
['FuelType'].isnull().sum())
# Imputing missing values of 'MetColor' using mode
Met_mode=cars_data2 ['MetColor'].mode()
cars_data2['MetColor'].fillna(cars_data2 ['MetColor'].mode() [0],
inplace=True)
print('count of missing values in MetColor column:\n', cars_data2
['MetColor'].isnull().sum())
# rechecking the null values in the data frame after filling the data
print('count of missing values in each column: \n',
cars_data2.isnull().sum())
# Imputing missing values using lambda function
cars_data3=cars_data3.apply(lambda x:x.fillna(x.mean()) if
x.dtype=='float'
else x.fillna(x.value_counts().index[0]))
print('count of missing values in each column:\n',
cars_data3.isnull().sum())
```

Output:

```
count   1436.000000  1336.000000    1421.000000  1430.000000
1286.000000
mean   10730.824513    55.672156   68647.239972   101.478322
0.674961
std     3626.964585    18.589804   37333.023589    14.768255
0.468572
min     4350.000000     1.000000       1.000000    69.000000
0.000000
25%     8450.000000    43.000000   43210.000000    90.000000
0.000000
50%     9900.000000    60.000000   63634.000000   110.000000
1.000000
75%    11950.000000    70.000000   87000.000000   110.000000
1.000000
max    32500.000000    80.000000  243000.000000   192.000000
1.000000

         Automatic           CC        Doors      Weight
count  1436.000000  1436.000000  1436.000000  1436.00000
mean      0.055710  1566.827994     4.033426  1072.45961
std       0.229441   187.182436     0.952677    52.64112
min       0.000000  1300.000000     2.000000  1000.00000
25%       0.000000  1400.000000     3.000000  1040.00000
50%       0.000000  1600.000000     4.000000  1070.00000
75%       0.000000  1600.000000     5.000000  1085.00000
max       1.000000  2000.000000     5.000000  1615.00000
count of missing values in Age column:
 0
count of missing values in KM column:
 0
count of missing values in HP column:
 0
count of unique values from FuelType column:
 FuelType
Petrol    1177
Diesel     144
CNG         15
Name: count, dtype: int64
count of missing values in Fueltype column:
 0
count of missing values in MetColor column:
 0
count of missing values in each column:
 Price        0
Age          0
KM           0
FuelType     0
HP           0
MetColor     0
Automatic    0
CC           0
Doors        0
Weight       0
dtype: int64
count of missing values in each column:
 Price        0
```

```
Age           0
KM            0
FuelType      0
HP            0
MetColor      0
Automatic     0
CC            0
Doors         0
Weight        0
dtype: int64
```

Result: A Python program was written to perform the given operations.

Experiment no.:10 *Date: 18-04-2024*

Experiment 10: Pandas Dataframes: Exploratory Data Analysis

Aim: Write a program to perform Exploratory data analysis using.
- Frequency tables
- Two-way tables
- Two-way tables – joint probability
- Two-way tables – marginal probability
- Two-way tables – conditional probability
- Correlation matrix

Dataset : 'Iris_data_sample.csv', 'Iris_data_sample.xlsx', 'Iris_data_sample.txt', 'Toyota.csv'

Software Required: Google Colab.

Description:

The program conducts thorough exploratory data analysis (EDA), summarizing data distribution with frequency tables, exploring relationships between variables with two-way tables and their associated probabilities, and identifying correlations using a correlation matrix. This helps in gaining insights into the dataset's characteristics and relationships, guiding subsequent analysis or modeling decisions effectively.

A)

Program:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import files
# Upload a file from your local machine
uploaded = files.upload()


#importing data from csv file
cars_data=pd.read_csv('cars_cleaned_app10.csv',index_col=0)
#checking for missing values in the dataframe
print('count of missing values in each column: \n',
cars_data.isnull().sum())
```

Output:

```
count of missing values in each column:
 Price         0
Age          0
KM           0
FuelType     0
HP           0
MetColor     0
```

```
Automatic     0
CC            0
Doors         0
Weight        0
dtype: int64
```

Result: A Python program was written to perform the given operations.

B)

Program:

```
#creating copy of original data
cars_data2=cars_data.copy()
# Frequency Tables
'''pandas.crosstab() --->To compute a simple cross tabulation of one,
two (or more)
factors --->By default computes a frequency table of the factors'''
cross = pd.crosstab(index=cars_data2['FuelType'], columns='count')
print("Frequency of categorial data in FuelType column is:", cross)
```

Output:

```
Frequency of categorial data in FuelType column is: col_0     count
FuelType
CNG            15
Diesel        144
Petrol       1277
```

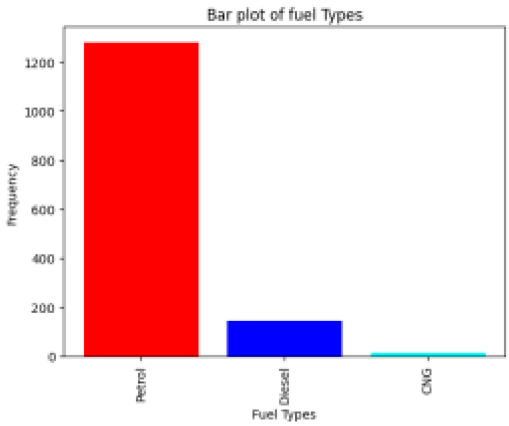
Result: A Python program was written to perform the given operations.

C)

Program:

```
# It is observed most of the cars have petrol as fuel type
# This is similar to barplot
counts=np.array([cars_data2 ['FuelType'].value_counts()])
counts=counts.flatten()
fuelType=('Petrol', 'Diesel', 'CNG')
index=np.arange(len(fuelType))
plt.bar(index, counts, color=['red', 'blue', 'cyan'])
plt.title('Bar plot of fuel Types')
plt.xlabel('Fuel Types')
plt.ylabel('Frequency')
plt.xticks (index, fuelType, rotation=90)
plt.show()
```

Output:



Result: A Python program was written to perform the given operations.

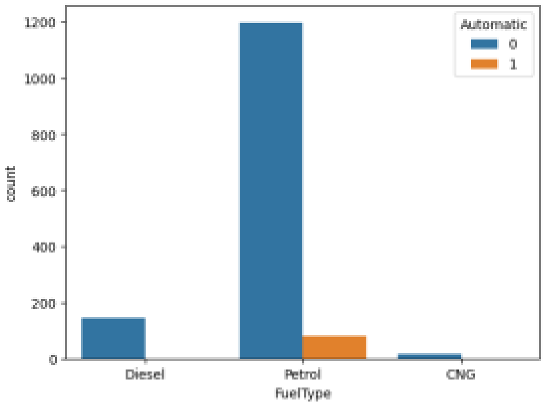
D)

Program:

```
#Two-way tables
'''--->To look at the frequency distribution of gearbox types with
respect to different fuel types of the cars'''
cross2=pd.crosstab(index=cars_data2['Automatic'], columns=cars_data2
['FuelType'])
print("Frequency relationship between FuelType and Automatic variable:
\n", cross2)
# It is observed that cars with Automatic gearbox system are of Petrol
fuel type
# This can be visualized by countplot function in seaborn
sns.countplot(x='FuelType', data=cars_data2, hue='Automatic')
# Two-way table: joint probability
'''Joint probability is the likelihood of two independent events
happening at the same time'''
cross3=pd.crosstab(index=cars_data2['Automatic'], columns=cars_data2
['FuelType'], normalize=True)
print("joint probability between FuelType and Automatic variable: \n",
cross3)
```

Output:

```
Frequency relationship between FuelType and Automatic variable:
 FuelType    CNG  Diesel  Petrol
Automatic
0            15     144    1197
1             0       0      80
joint probability between FuelType and Automatic variable:
 FuelType         CNG     Diesel     Petrol
Automatic
0           0.010446   0.100279   0.833565
   1           0.000000   0.000000   0.055710
```



Result: A Python program was written to perform the given operations.

E)

Program:

```
# Two-way table: marginal probability
cross4=pd.crosstab(index=cars_data2 ['Automatic'],
columns=cars_data2['FuelType'], margins=True, normalize=True)
print("marginal probability of FuelType and Automatic variables within
joint probability: \n", cross4)
# probability of cars having manual gear box when the fuel type are CNG
or Diesel or Petrol is 0.95
```

Output:

```
marginal probability of FuelType and Automatic variables within joint
probability:
 FuelType        CNG     Diesel     Petrol       All
Automatic
0          0.010446  0.100279  0.833565  0.94429
1          0.000000  0.000000  0.055710  0.05571
All        0.010446  0.100279  0.889276  1.00000
```

Result: A Python program was written to perform the given operations.

F)

Program:

```
# Two-way table: conditional probability
'''---> Conditional probability is the probability of an event (A),
given that another event (B) has already occurred ---> Given the type
of
gear box, probability of different fuel type'''
cross5=pd.crosstab(index=cars_data2['Automatic'],
columns=cars_data2['FuelType'], margins=True, normalize='index')
```

```
print("marginal probability of FuelType when Automatic variable has
already occured: \n", cross5)
cross6=pd.crosstab(index=cars_data2['Automatic'],
columns=cars_data2['FuelType'], margins=True, normalize='columns')
print("marginal probability of Automatic when FuelType variable has
already occured: \n", cross6)
```

Output:

```
marginal probability of FuelType when Automatic variable has already
occured:
 FuelType        CNG    Diesel    Petrol
Automatic
0          0.011062  0.106195  0.882743
1          0.000000  0.000000  1.000000
All        0.010446  0.100279  0.889276
marginal probability of Automatic when FuelType variable has already
occured:
 FuelType   CNG  Diesel    Petrol      All
Automatic
0          1.0     1.0  0.937353  0.94429
1          0.0     0.0  0.062647  0.05571
```

Result: A Python program was written to perform the given operations.

G)

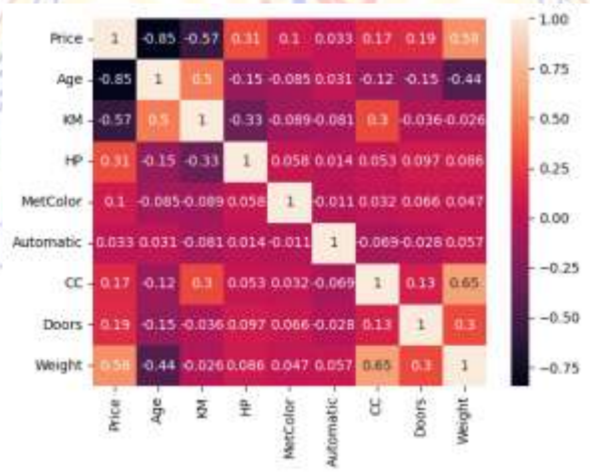
Program:

```
# correlation
'''... DataFrame.corr(self, 'pearson') --->Correlation between
numerical variables --->To compute pairwise
correlation of columns excluding NA/null values --->Excluding the
categorical variables to find the
Pearson's correlation'''
numerical_data=cars_data2.select_dtypes (exclude=[object])
print(numerical_data.shape)
corr_matrix=numerical_data.corr()
print('correlation between numerical variables is', corr_matrix)
# heat maps for visualizing correlation
sns.heatmap (corr_matrix, annot=True)
# pairwise plots
'''--->It is used to plot pairwise relationships in a dataset ---
>Creates
scatterplots for joint relationships and histograms for univariate
distributions'''
# pairwise plots with hue as fueltype
sns.pairplot (cars_data2, kind="scatter", hue="FuelType")
plt.show()
```

Output:

```
(1436, 9)
correlation between numerical variables is             Price
Age         KM         HP  MetColor  Automatic  \
Price      1.000000 -0.845111 -0.565926  0.308414  0.100920   0.033081
Age       -0.845111  1.000000  0.495609 -0.152946 -0.084719   0.030931
KM        -0.565926  0.495609  1.000000 -0.332297 -0.088657  -0.080803
HP         0.308414 -0.152946 -0.332297  1.000000  0.058166   0.013753
MetColor   0.100920 -0.084719 -0.088657  0.058166  1.000000  -0.011450
Automatic  0.033081  0.030931 -0.080803  0.013753 -0.011450   1.000000
CC         0.165067 -0.116255  0.296305  0.053466  0.032108  -0.069321
Doors      0.185326 -0.151785 -0.035771  0.096938  0.065953  -0.027654
Weight     0.581198 -0.442055 -0.026476  0.086214  0.046614   0.057249

                 CC     Doors    Weight
Price      0.165067  0.185326  0.581198
Age       -0.116255 -0.151785 -0.442055
KM         0.296305 -0.035771 -0.026476
HP         0.053466  0.096938  0.086214
MetColor   0.032108  0.065953  0.046614
Automatic -0.069321 -0.027654  0.057249
CC         1.000000  0.126768  0.651450
Doors      0.126768  1.000000  0.302618
Weight     0.651450  0.302618  1.000000
```



Result: A Python program was written to perform the given operations.