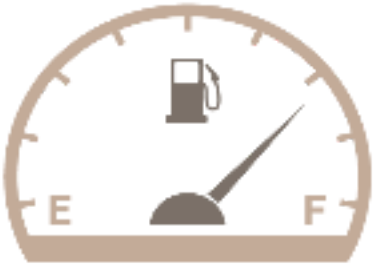
California State University, Los Angeles

CIS 5250 : Project

Date: 12/08/2019

DATA VISUALIZATION OF CAR FUEL CONSUMPTION USING ANACONDA



Instructor:

Prof. Shilpa Balan

Project by:

Sai kiran Gontyala - 400015938

Yoshitha Gattu - 400015262

DATA SET DESCRIPTION

Data set URL: https://www.kaggle.com/anderas/car-consume

This dataset helps us evaluate and predict car fuel consumption based on the gas type and the data used to build this dataset was primarily from the various values seen on the car display and it can be used to determine the effect of the weather, speed or gas type on the car consumption. This study evaluates statistical analysis for predicting fuel consumption in heavy vehicles. The idea is to use historical data describing driving situations to predict fuel consumption in liters per distance. The general problem description is to examine a large number of attributes describing a fuel consumption situation and to find a regression from such attributes. Attributes could be environmental conditions, vehicle configuration, driver behavior, and weather conditions.

The specific problem investigated in this study is how to do fuel consumption prediction for the car using the data sources available on the Car display. One goal of the study is to evaluate different approaches for regression from a set of descriptive attributes to fuel consumption in liters per distance.

The below table describes briefly about the columns in our data set and their properties:

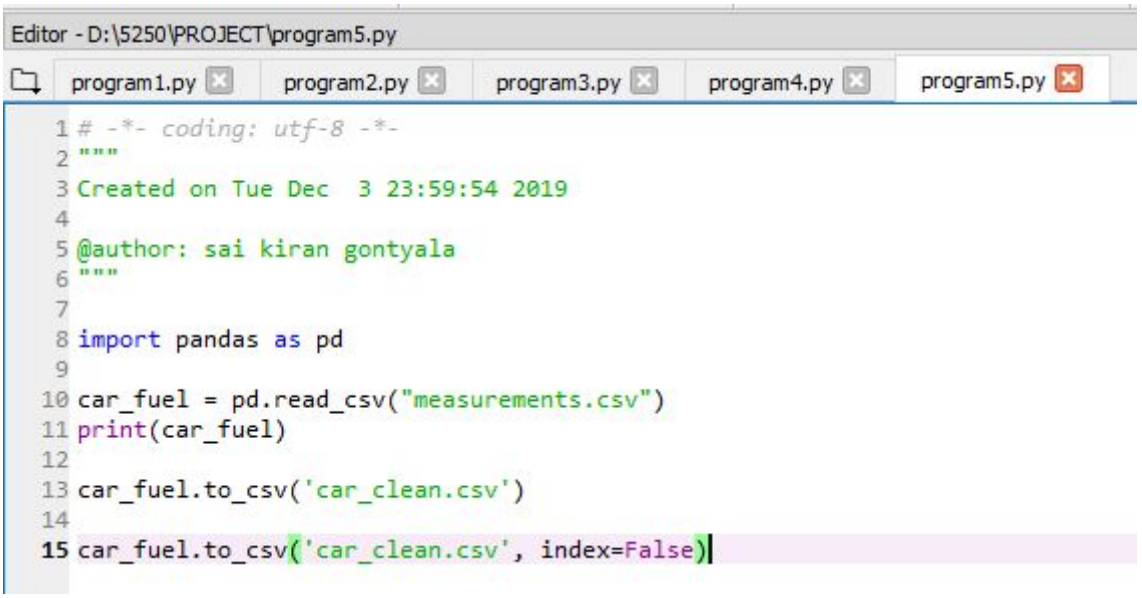
Field Name	Field Explanation	Type	Sample Value	Range Of Value	Attributes	Units
distance	Distance covered	Textual	28	2 to 216.2	Whole numbers	Kilometers (km)
consume	Fuel consumption	Textual	5	3.3 to 12.2	Whole numbers	L/100km
speed	Average speed	Textual	26	14 to 90	Whole numbers	km/h
temp_inside	Temperature inside	Ordinal	21.5	19 to 25.5	Max 15 characters	°C
temp_outside	Temperature outside	Ordinal	12	-5 to 31	Max 15 characters	°C
specials	Anything special that happened	Nominal	AC	9 categories	Max 60 characters	-
gas_type	Gas type being used	Nominal	E10	2 categories	Max 60 characters	-
AC	Air Conditioner on/off	Textual	0	0 and 1	Whole numbers	-
rain	Raining	Textual	1	0 and 1	Whole numbers	-
sun	Sunny enough	Textual	0	0 and 1	Whole numbers	-
refill_liters	Fuel refill	Nominal	45	10 to 45	Max 60 characters	Liters
refill_gas	Refill fuel type	Nominal	SP98	2 categories	Max 60 characters	-

DATA CLEANING

1. Deleting the index column:

The first thing that we did when we imported the dataset was to verify the contents using the print function. After quickly verifying the data we noticed that there is an index column associated with the data set and we wanted to get rid of the index column.

Code #1:



We imported the Pandas Data frame[1] , which is one of the python packages used to perform better operations on rows/columns like selecting, deleting, adding, and renaming. We used this data frame to read and import the file named 'measurements.csv'[2] from local folder. We transferred the data to another file named 'car_clean.csv' where we used 'index=False' as a parameter to drop the index column which was not necessary for the analysis.

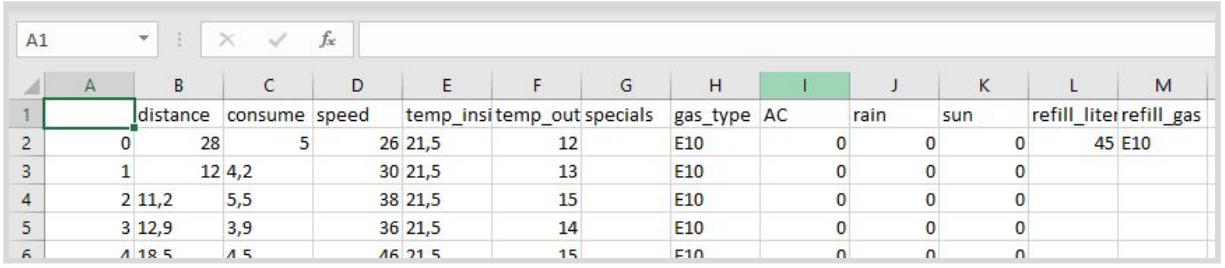
The output of 'print(car_fuel)' :

```
In [6]: runfile('D:/5250/PROJECT/program5.py', wdir='D:/5250/PROJECT')
     Unnamed: 0 distance consume  speed  ... rain  sun refill_liters refill_gas
0             0       28       5     26  ...    0    0           45         E10
1             1       12     4,2     30  ...    0    0          NaN         NaN
2             2     11,2     5,5     38  ...    0    0          NaN         NaN
3             3     12,9     3,9     36  ...    0    0          NaN         NaN
4             4     18,5     4,5     46  ...    0    0          NaN         NaN
..          ...      ...     ...    ...  ...  ...  ...          ...         ...
383         383       16     3,7     39  ...    0    0          NaN         NaN
384         384     16,1     4,3     38  ...    0    0          NaN         NaN
385         385       16     3,8     45  ...    0    0          NaN         NaN
386         386     15,4     4,6     42  ...    0    0          NaN         NaN
387         387     14,7       5     25  ...    0    0          NaN         NaN

[388 rows x 13 columns]

In [7]:
```

Here, as in the above screenshot of the output file, we can notice that there is one index column with the serial numbers and the serial numbers to the left of the index column are the default serial numbers in the python editor which denote the row numbers.

Pre-cleaning:

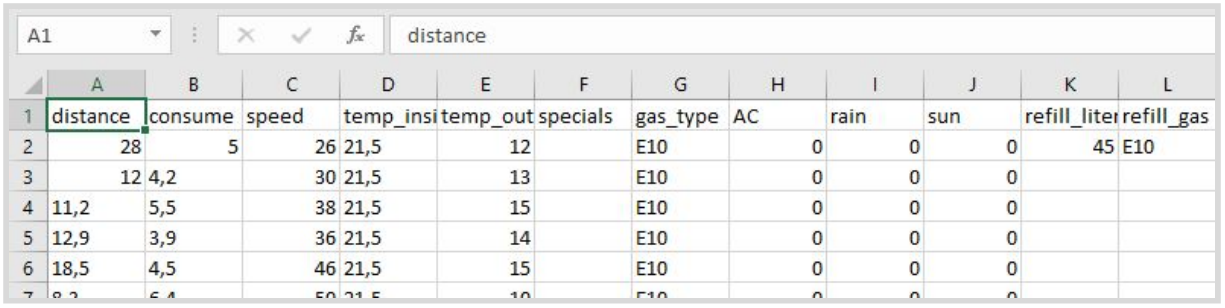
Output of 'car_fuel.csv' file



	A	B	C	D	E	F	G	H	I	J	K	L	M
1		distance	consume	speed	temp_insi	temp_out	specials	gas_type	AC	rain	sun	refill_liter	refill_gas
2	0	28	5	26	21,5	12		E10	0	0	0	45	E10
3	1	12	4,2	30	21,5	13		E10	0	0	0		
4	2	11,2	5,5	38	21,5	15		E10	0	0	0		
5	3	12,9	3,9	36	21,5	14		E10	0	0	0		
6	4	18,5	4,5	46	21,5	15		E10	0	0	0		

Post-cleaning:

Output of 'car_clean.csv' file

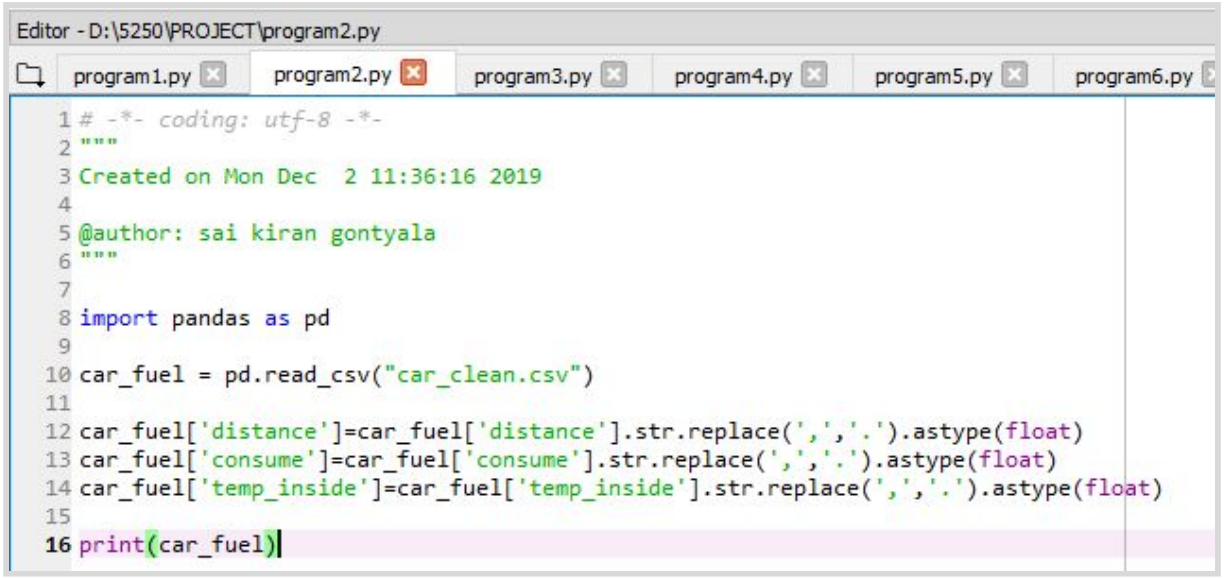


	A	B	C	D	E	F	G	H	I	J	K	L
1	distance	consume	speed	temp_insi	temp_out	specials	gas_type	AC	rain	sun	refill_liter	refill_gas
2	28	5	26	21,5	12		E10	0	0	0	45	E10
3	12	4,2	30	21,5	13		E10	0	0	0		
4	11,2	5,5	38	21,5	15		E10	0	0	0		
5	12,9	3,9	36	21,5	14		E10	0	0	0		
6	18,5	4,5	46	21,5	15		E10	0	0	0		

2. Applying delimiters - Converting ‘,’ to ‘.’:

We noticed that the decimal point notation for the numbers in the dataset was represented with ‘,’ and not ‘.’. We used a string function .str.replace[3] to swap the symbols where the replace function was employed to replace the methods in strings. We converted “, ” to “.” to represent the numbers in a correct decimal format.

Code #2:



Editor - D:\5250\PROJECT\program2.py

program1.py | program2.py | program3.py | program4.py | program5.py | program6.py

```
# -*- coding: utf-8 -*-
"""
Created on Mon Dec  2 11:36:16 2019

@author: sai kiran gontyala
"""

import pandas as pd

car_fuel = pd.read_csv("car_clean.csv")

car_fuel['distance']=car_fuel['distance'].str.replace(',','.').astype(float)
car_fuel['consume']=car_fuel['consume'].str.replace(',','.').astype(float)
car_fuel['temp_inside']=car_fuel['temp_inside'].str.replace(',','.').astype(float)

print(car_fuel)
```

Pre-cleaning:

Applying delimiters to columns : distance, consume, temp_inside

```
In [1]: runfile('D:/5250/PROJECT/program1.py', wdir='D:/5250/PROJECT')
    distance consume  speed temp_inside  ...  rain sun refill liters  refill gas
0         28       5     26        21,5  ...     0   0           45         E10
1         12     4,2     30        21,5  ...     0   0          NaN         NaN
2       11,2     5,5     38        21,5  ...     0   0          NaN         NaN
3       12,9     3,9     36        21,5  ...     0   0          NaN         NaN
4       18,5     4,5     46        21,5  ...     0   0          NaN         NaN
..       ...     ...    ...         ...  ...   ...  ..          ...         ...
383       16     3,7     39        24,5  ...     0   0          NaN         NaN
384     16,1     4,3     38          25  ...     0   0          NaN         NaN
385       16     3,8     45          25  ...     0   0          NaN         NaN
386     15,4     4,6     42          25  ...     0   0          NaN         NaN
387     14,7       5     25          25  ...     0   0          NaN         NaN

[388 rows x 12 columns]

In [2]:
```

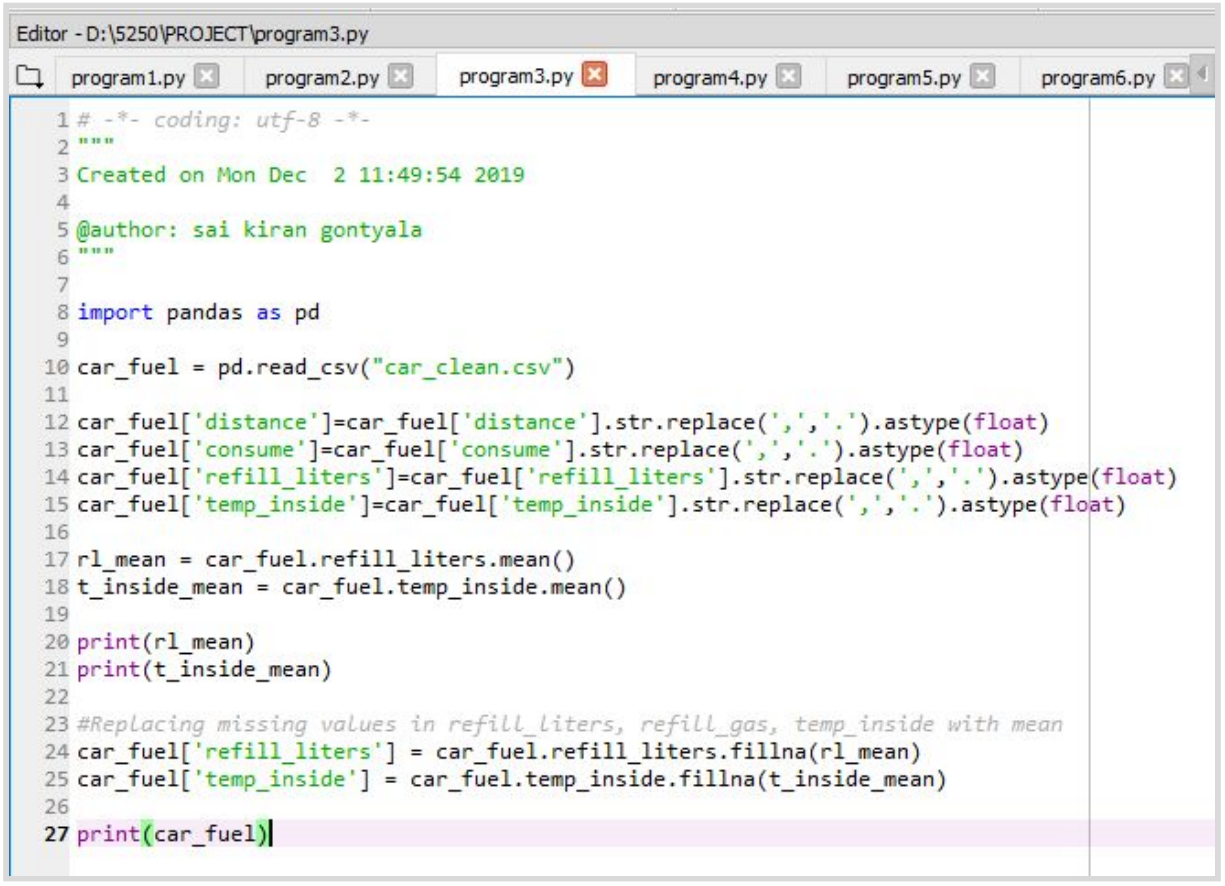
Post-cleaning:

```
In [2]: runfile('D:/5250/PROJECT/program2.py', wdir='D:/5250/PROJECT')
     distance  consume  speed  temp_inside  ...  rain sun refill liters  refill gas
0        28.0      5.0     26         21.5  ...     0   0           45         E10
1        12.0      4.2     30         21.5  ...     0   0          NaN         NaN
2        11.2      5.5     38         21.5  ...     0   0          NaN         NaN
3        12.9      3.9     36         21.5  ...     0   0          NaN         NaN
4        18.5      4.5     46         21.5  ...     0   0          NaN         NaN
..        ...      ...    ...          ...  ...   ...  ..          ...         ...
383      16.0      3.7     39         24.5  ...     0   0          NaN         NaN
384      16.1      4.3     38         25.0  ...     0   0          NaN         NaN
385      16.0      3.8     45         25.0  ...     0   0          NaN         NaN
386      15.4      4.6     42         25.0  ...     0   0          NaN         NaN
387      14.7      5.0     25         25.0  ...     0   0          NaN         NaN

[388 rows x 12 columns]

In [3]:
```

3. Replacing 'NaN' and missing values in temp_inside, refill_gas with mean:

Finally, in order to replace the 'NaN' values we found out the mean value and replaced the null values for each column necessary.

Code #3:



Pre-cleaning:

We notice 'Nan' values in the columns : temp_inside and refill_liters. Hence replacing them with the mean values of their respective columns

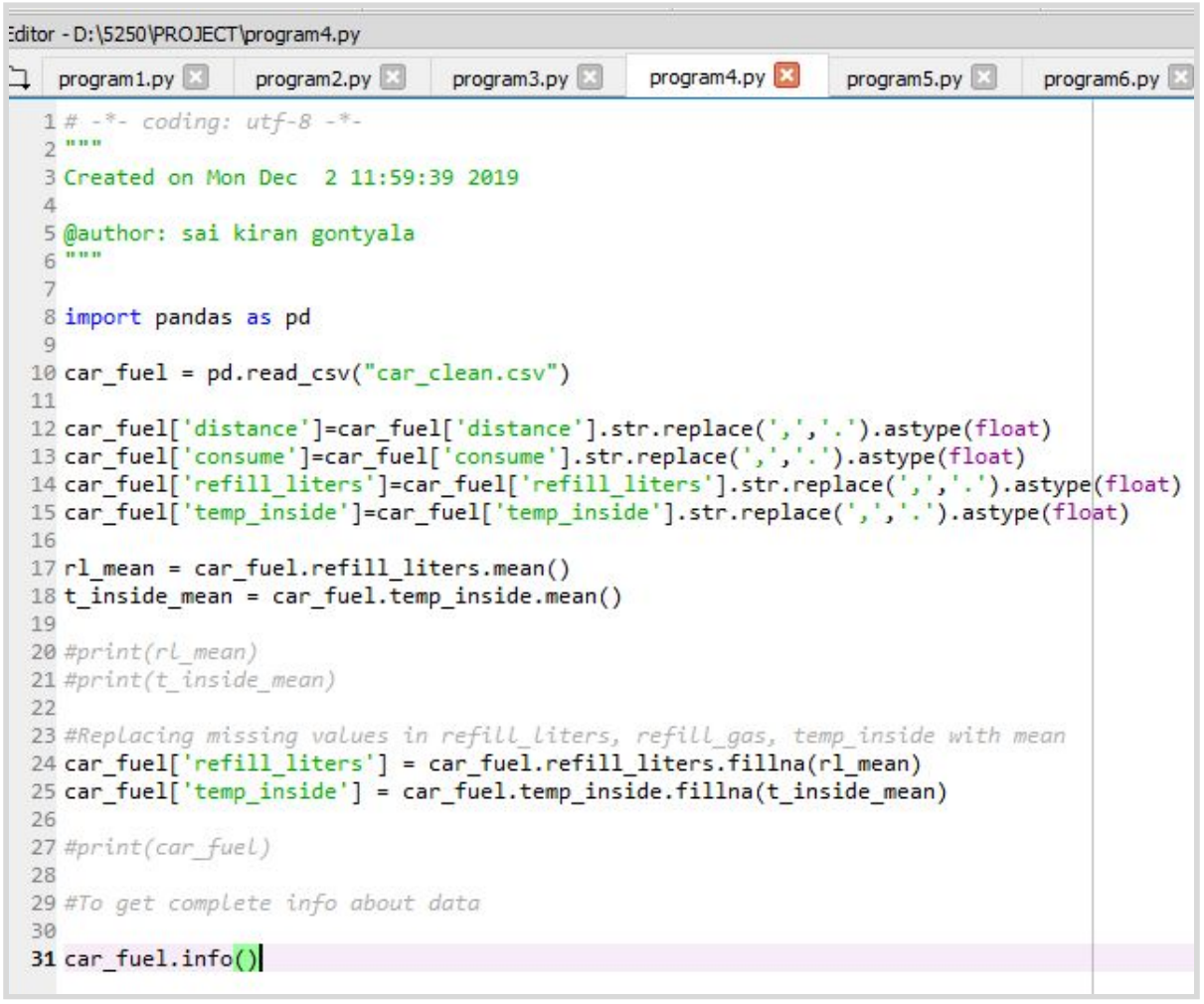
```
In [2]: runfile('D:/5250/PROJECT/program2.py', wdir='D:/5250/PROJECT')
     distance  consume  speed  temp_inside  ...  rain sun refill liters  refill gas
0        28.0      5.0     26         21.5  ...     0   0            45         E10
1        12.0      4.2     30         21.5  ...     0   0           NaN         NaN
2        11.2      5.5     38         21.5  ...     0   0           NaN         NaN
3        12.9      3.9     36         21.5  ...     0   0           NaN         NaN
4        18.5      4.5     46         21.5  ...     0   0           NaN         NaN
..        ...      ...    ...          ...  ...   ...  ..           ...         ...
383      16.0      3.7     39         24.5  ...     0   0           NaN         NaN
384      16.1      4.3     38         25.0  ...     0   0           NaN         NaN
385      16.0      3.8     45         25.0  ...     0   0           NaN         NaN
386      15.4      4.6     42         25.0  ...     0   0           NaN         NaN
387      14.7      5.0     25         25.0  ...     0   0           NaN         NaN

[388 rows x 12 columns]

In [3]:
```

Post-cleaning:

```
In [1]: runfile('D:/5250/PROJECT/program3.py', wdir='D:/5250/PROJECT')
37.11538461538461
21.929521276595743
     distance  consume  speed  temp_inside  ...  rain sun refill_liters  refill_gas
0        28.0      5.0     26         21.5  ...     0   0     45.000000         E10
1        12.0      4.2     30         21.5  ...     0   0     37.115385         NaN
2        11.2      5.5     38         21.5  ...     0   0     37.115385         NaN
3        12.9      3.9     36         21.5  ...     0   0     37.115385         NaN
4        18.5      4.5     46         21.5  ...     0   0     37.115385         NaN
..        ...      ...    ...          ...  ...   ...  ..           ...         ...
383      16.0      3.7     39         24.5  ...     0   0     37.115385         NaN
384      16.1      4.3     38         25.0  ...     0   0     37.115385         NaN
385      16.0      3.8     45         25.0  ...     0   0     37.115385         NaN
386      15.4      4.6     42         25.0  ...     0   0     37.115385         NaN
387      14.7      5.0     25         25.0  ...     0   0     37.115385         NaN

[388 rows x 12 columns]
```

4. Cleaned Data frame Information:

Code #4:



In the output we used the .info( ) function[4] to get the complete information about the cleansed data set with 12 columns and 388 rows. We can notice the data types of the columns and also that there are no numerical null values present in the data set.

```
In [8]: runfile('D:/5250/PROJECT/program4.py', wdir='D:/5250/PROJECT')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 388 entries, 0 to 387
Data columns (total 12 columns):
distance          388 non-null float64
consume           388 non-null float64
speed             388 non-null int64
temp_inside       388 non-null float64
temp_outside      388 non-null int64
specials          93 non-null object
gas_type          388 non-null object
AC                388 non-null int64
rain              388 non-null int64
sun               388 non-null int64
refill_liters     388 non-null float64
refill_gas        13 non-null object
dtypes: float64(4), int64(5), object(3)
memory usage: 36.5+ KB
```

5. Testing the data by performing basic operations:

We performed basic operations like creating a tuple $x_{[5]}$ and then converting it into a list $y_{[6]}$. Then again we changed a column name and converted x back to a tuple. We were successful in doing the same. Hence, the data set is responding to the changes we are making.

Code #5:

```
5 @author: sai kiran gontyala
6 """
7 import pandas as pd
8
9 car_fuel = pd.read_csv("car_clean.csv")
10
11 x = ("rain", "sun", "refill_liters", "refill_gas")
12 y = list(x)
13 y[1] = "hot_sun"
14 x = tuple(y)
15
16 print(x)
```

Output :

```
In [6]: runfile('D:/5250/PROJECT/program14.py', wdir='D:/5250/PROJECT')
('rain', 'hot_sun', 'refill_liters', 'refill_gas')

In [7]:
```

Therefore, the data set is now ready for further statistics and visualizations.

SUMMARY STATISTICS

1. Summary statistics of “speed” column:

To find out the basic statistics of a column we used the describe( ) function, which computes a summary of statistics pertaining to the DataFrame.

Code #6:

```
5 @author: sai kiran gontyala
6 """
7
8 import pandas as pd
9
10 car_fuel =pd.read_csv("car_clean.csv")
11
12 stat = car_fuel['speed'].describe()
13
14 print(stat)
```

Output:

```
In [9]: runfile('D:/5250/PROJECT/program6.py', wdir='D:/5250/PROJECT')
count    388.000000
mean      41.927835
std       13.598524
min       14.000000
25%       32.750000
50%       40.500000
75%       50.000000
max       90.000000
Name: speed, dtype: float64
```

2. Mean, median and standard deviation of temperature:

To find out the mean of values, median values and standard deviation of the values we used mean ( ) , median( ) and std( ) function respectively.

Code #7:

```
@author: sai kiran gontyala
"""

import pandas as pd

car_fuel = pd.read_csv("car_clean.csv")

car_fuel['temp_inside']=car_fuel['temp_inside'].str.replace(',','.').astype(float)

t_inside_mean = car_fuel.temp_inside.mean()
t_outside_mean = car_fuel.temp_outside.mean()

t_inside_median = car_fuel.temp_inside.median()
t_outside_median = car_fuel.temp_outside.median()

t_inside_std = car_fuel.temp_inside.std()
t_outside_std = car_fuel.temp_outside.std()

print(t_inside_mean)
print(t_outside_mean)

print(t_inside_median)
print(t_outside_median)

print(t_inside_std)
print(t_outside_std)
```

Output:

```
In [20]: runfile('D:/5250/PROJECT/program7.py', wdir='D:/5250/PROJECT')
21.929521276595743
11.358247422680412
22.0
10.0
1.0104550972438595
6.9915422526368785
```

3. Recoding gas_type E10 and SP98 to 0 and 1 respectively:

Here, we imported numpy package which is a general-purpose array-processing package. We created a user defined function def function_name(arguments) and used If-else to recode gas E10 to 0 and SP98 to 1.

Code #8:

```
import pandas as pd
import numpy as np

car_fuel = pd.read_csv('car_clean.csv')

car_fuel['distance']=car_fuel['distance'].str.replace(',','.').astype(float)
car_fuel['consume']=car_fuel['consume'].str.replace(',','.').astype(float)
car_fuel['refill_liters']=car_fuel['refill_liters'].str.replace(',','.').astype(float)
car_fuel['temp_inside']=car_fuel['temp_inside'].str.replace(',','.').astype(float)

rl_mean = car_fuel.refill_liters.mean()
t_inside_mean = car_fuel.temp_inside.mean()

car_fuel['refill_liters'] = car_fuel.refill_liters.fillna(rl_mean)
car_fuel['temp_inside'] = car_fuel.temp_inside.fillna(t_inside_mean)

def recode_gas_type(gas):

    if gas == 'E10':
        return 0
    elif gas == 'SP98':
        return 1
    else:
        return np.nun

car_fuel['recode'] = car_fuel.gas_type.apply(recode_gas_type)

print(car_fuel)
```

Output:

```
In [8]: runfile('D:/5250/PROJECT/program8.py', wdir='D:/5250/PROJECT')
     distance  consume  speed  ...  refill_liters  refill_gas recode
0        28.0      5.0     26  ...      45.000000         E10      0
1        12.0      4.2     30  ...      37.115385         NaN      0
2        11.2      5.5     38  ...      37.115385         NaN      0
3        12.9      3.9     36  ...      37.115385         NaN      0
4        18.5      4.5     46  ...      37.115385         NaN      0
..        ...      ...    ...  ...            ...         ...    ...
383      16.0      3.7     39  ...      37.115385         NaN      1
384      16.1      4.3     38  ...      37.115385         NaN      1
385      16.0      3.8     45  ...      37.115385         NaN      1
386      15.4      4.6     42  ...      37.115385         NaN      1
387      14.7      5.0     25  ...      37.115385         NaN      1

[388 rows x 13 columns]
```

ANALYSIS & VISUALIZATION

Question 1:

Write a code to represent frequency versus speed range data and at which speed maximum percent of vehicles are prone to travel in a graphical method?

Answer:

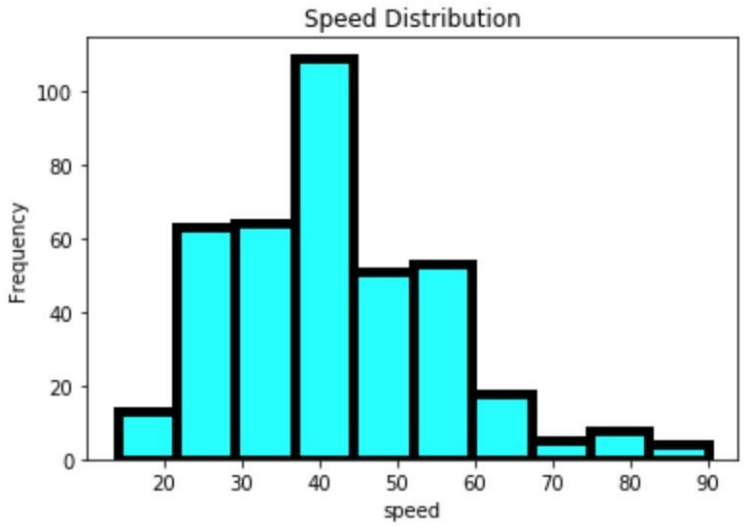
To display the speed distribution and the speed at which maximum amount of vehicles travel we used a histogram representation. After importing the file we used df [ ].plot.hist to get the desired output and passed the number of bins, line width as parameter for a basic histogram output. We also defined color, edge color as parameters to enhance the graph.

It can be understood from the graphical output that the maximum speed of a vehicle remains around 40 kmph for about 100 percent. Speed velocity below 10 kmph is very rare.

Code #9:

```
import pandas as pd
import matplotlib.pyplot as plt
car_fuel=pd.read_csv("car_clean.csv")
car_fuel['speed'].plot.hist(bins=10,linewidth=5,color='cyan',edgecolor='black')
plt.xlabel('speed')
plt.title('Speed Distribution')
plt.show()

```

Output:



Question 2:

Represent a bar chart showing what type of gas is preferred the most by the customers and show the count of each gas type?

Answer:

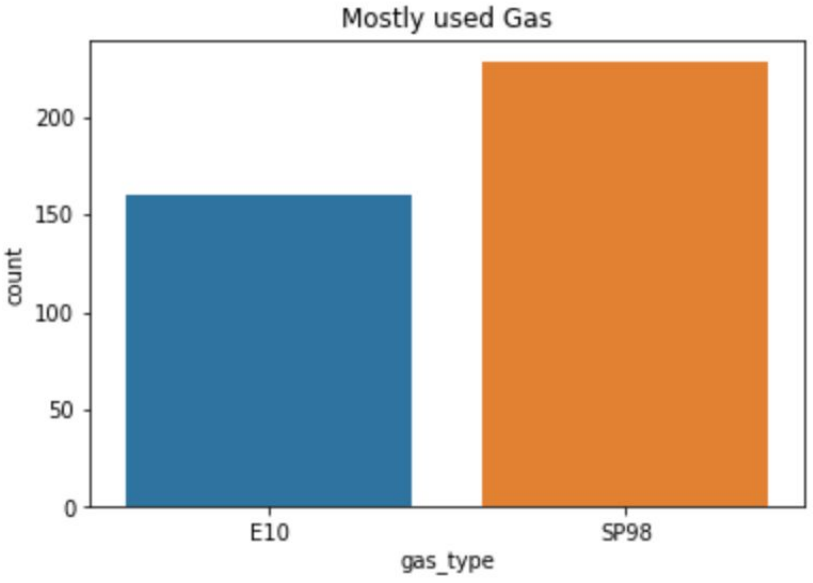
Initially while writing the code we imported all the necessary packages like pandas, matplotlib including seaborn package which provides a high-level interface for drawing attractive and informative statistical graphics like a bar chart in this case. After reading the data set we used seaborn.countplot to represent the data in a bar plot.

In the graph, we notice that there are more instances for SP98 gas with approximately more than 150 customers and more than 200 customers for E10 gas respectively.

Code #10:

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

car_fuel = pd.read_csv("car_clean.csv")
sns.countplot(car_fuel['gas_type'])
plt.title('Mostly used Gas')
plt.show()
```

Output:



Question 3:

The dataset available shows the distance traveled in ‘y’ kilometers by car after burning x liters per kilometer approximately equal to 0.42 gallons per mile of gasoline. Make a scatterplot of data to display coordinates of distance and fuel consumption. What does the correlation imply about the relationship between fuel efficiency and distance of a car?

Answer:

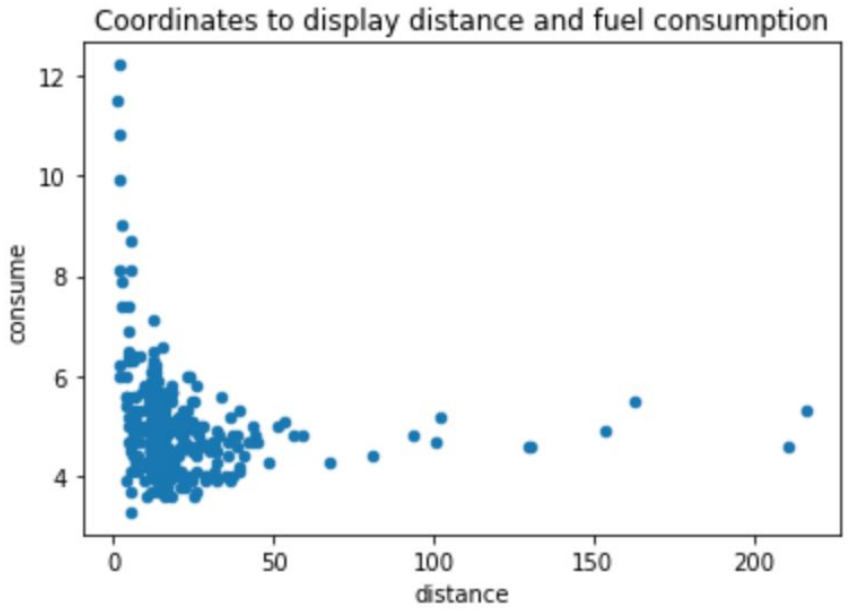
To represent the distance travelled and the amount of fuel consumed in a scatter plot, we used df.plot where we defined the plot to be scattered and passed distance and fuel consumed as parameters.

The scatter graph shows some information about cars and the amount of fuel it consumes when the car travels on gasoline. Here we do not see a generally increasing or decreasing pattern which implies that the graph is not linear; however, some points at the bottom are unusual because of which correlations can be negative, which means the two sets of data are strongly linked together having a High Correlation but one value goes down as the other value increases and so Correlation is Negative. This will give us the overall fuel efficiency of these vehicles letting us know that the fuel consumption is less as the distance travelled is more.

Code #11:

```
import pandas as pd
import matplotlib.pylab as plt
car_fuel = pd.read_csv("car_clean.csv")
car_fuel['distance']=car_fuel['distance'].str.replace(',','.').astype(float)
car_fuel['consume']=car_fuel['consume'].str.replace(',','.').astype(float)
car_fuel.plot(kind = 'scatter', x='distance', y='consume')
plt.title("Coordinates to display distance and fuel consumption")
plt.show()
```

Output:



Question 4:

Predict the best fit for understanding the trend in behavior of distance travelled and fuel consumed with and without Air conditioner while applying generalized linear model plot.

Answer:

We began with importing packages and then queried the columns of the data frame with a boolean expression by assigning 'AC ON = 0' and 'AC OFF = 1'. Every plot in Seaborn has a set of fixed parameters. For sns.lmplot(), we have three mandatory parameters and the rest are optional that we may use as per our requirements. These 3 parameters are values for X-axis, values for Y-axis and reference to dataset.

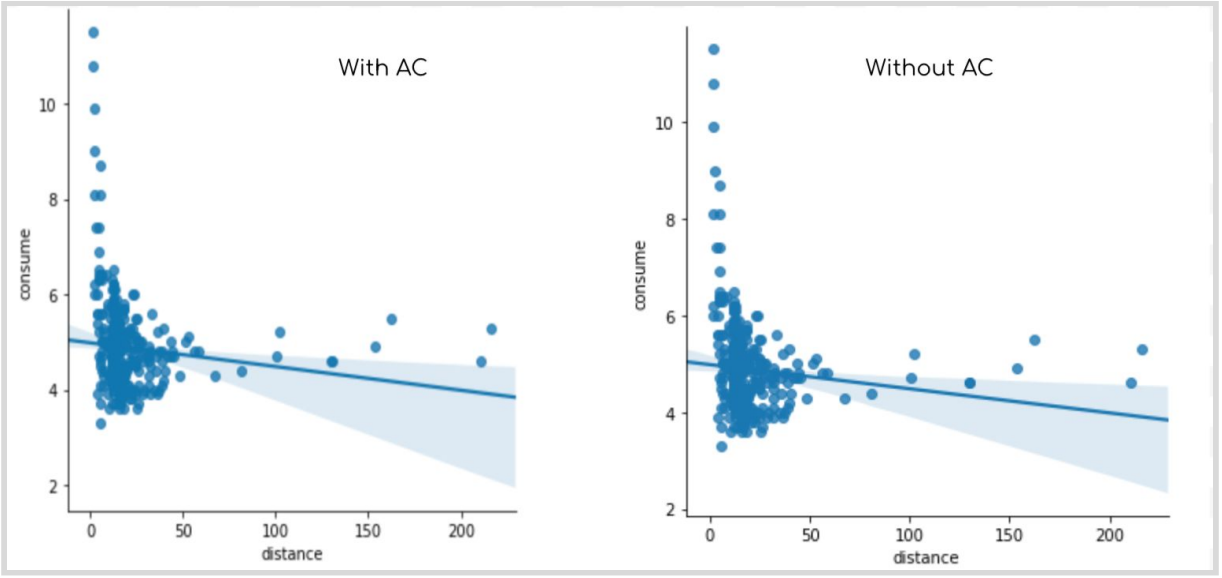
Here we see jumbled up data points on the plot with a linearly fitted line passing through, thus reflecting best fit for existing trend as per dataset. If we notice very closely, there is this shadow converging at the center where there is a chunk of our data. This convergent point is actually the statistical mean or in simpler words, the generalized prediction of fuel consumption value in a car on a daily basis.

In this case, looking at this first plot where AC = ON, we may say that if the distance is around 20 km, then the fuel consumption is approximately around 5 liters. In a similar way when we look at the second plot where AC= OFF, we notice that the distance travelled is almost upto 50 km where the fuel consumption is the same.

Code #12:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
car_fuel = pd.read_csv("car_clean.csv")
car_fuel['distance']=car_fuel['distance'].str.replace(',','.').astype(float)
car_fuel['consume']=car_fuel['consume'].str.replace(',','.').astype(float)
minac=car_fuel.query('AC==0.000000')
maxac=car_fuel.query('AC==1.000000')
sns.lmplot('distance','consume',data=minac)
sns.lmplot('distance','consume',data=maxac)
plt.title("Distance convered and Fuel consumption with and without AC")
plt.show()

```

Output With AC & Without AC:



Question 5:

Analyse the fuel consumed with respect to the distance travelled, also calculate the mean distance traveled and average distance traveled respective to the gas types? Graph a box plot accordingly and explain it.

Answer:

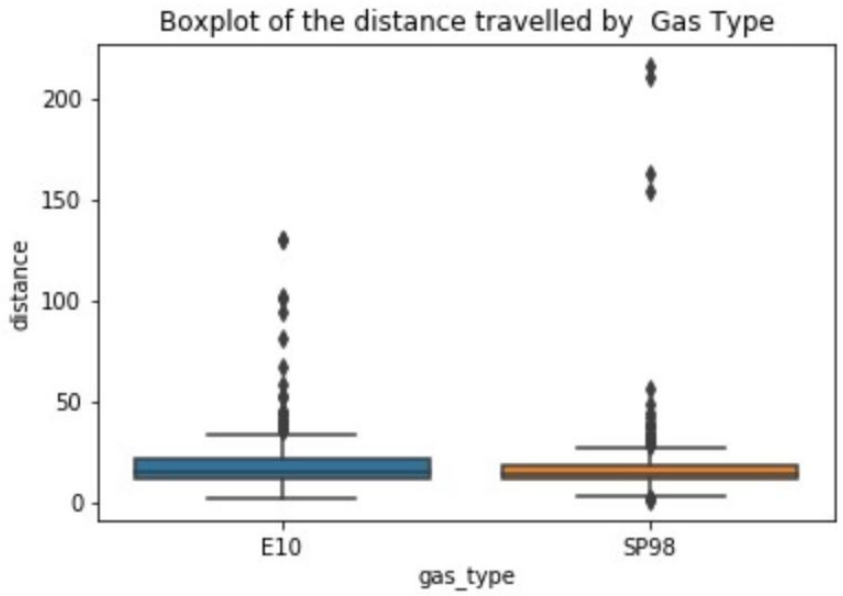
We plotted a boxplot by invoking .boxplot() on our data frame. The code below makes a boxplot of the distance traveled by gas type where we passed 'gas_type' & 'distance' as parameters to sns.boxplot.

The mean distance traveled is approximately 18 km and the average distance traveled when using E10 is approximately 15km which is higher than when using SP98 which is approximately 12 km.

Code 13:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
car_fuel = pd.read_csv("car_clean.csv")
car_fuel['distance']=car_fuel['distance'].str.replace(',','.').astype(float)
sns.boxplot(x='gas_type', y='distance', data=car_fuel)
plt.title("Boxplot of the distance travelled by  Gas Type")
plt.show()
```

Output:


Boxplot of the distance travelled by Gas Type
distance
200
150
100
50
0
E10
SP98
gas_type

APPENDIX - A

#1. Importing the dataset and removing index column

```
import pandas as pd

car_fuel = pd.read_csv("measurements.csv")

print(car_fuel)

car_fuel.to_csv('car_clean.csv')

car_fuel.to_csv('car_clean.csv', index=False)
```

#2. Applying delimiters

```
import pandas as pd

car_fuel = pd.read_csv("car_clean.csv")

car_fuel['distance']=car_fuel['distance'].str.replace(',','.').astype(float)

car_fuel['consume']=car_fuel['consume'].str.replace(',','.').astype(float)

car_fuel['temp_inside']=car_fuel['temp_inside'].str.replace(',','.').astype(float)

print(car_fuel)
```

#3. Replacing null values with mean

```
import pandas as pd

car_fuel = pd.read_csv("car_clean.csv")

car_fuel['distance']=car_fuel['distance'].str.replace(',','.').astype(float)

car_fuel['consume']=car_fuel['consume'].str.replace(',','.').astype(float)

car_fuel['refill_liters']=car_fuel['refill_liters'].str.replace(',','.').astype(float)

car_fuel['temp_inside']=car_fuel['temp_inside'].str.replace(',','.').astype(float)

rl_mean = car_fuel.refill_liters.mean()
```

```
t_inside_mean = car_fuel.temp_inside.mean()
print(rl_mean)
print(t_inside_mean)
#Replacing missing values in refill_liters, refill_gas, temp_inside with mean
car_fuel['refill_liters'] = car_fuel.refill_liters.fillna(rl_mean)
car_fuel['temp_inside'] = car_fuel.temp_inside.fillna(t_inside_mean)
print(car_fuel)
```

#4. Complete information about data

```
import pandas as pd
car_fuel = pd.read_csv("car_clean.csv")
car_fuel['distance']=car_fuel['distance'].str.replace(',','.').astype(float)
car_fuel['consume']=car_fuel['consume'].str.replace(',','.').astype(float)
car_fuel['refill_liters']=car_fuel['refill_liters'].str.replace(',','.').astype(float)
car_fuel['temp_inside']=car_fuel['temp_inside'].str.replace(',','.').astype(float)
rl_mean = car_fuel.refill_liters.mean()
t_inside_mean = car_fuel.temp_inside.mean()
#print(rl_mean)
#print(t_inside_mean)
# Replacing missing values in refill_liters, refill_gas, temp_inside with mean
car_fuel['refill_liters'] = car_fuel.refill_liters.fillna(rl_mean)
car_fuel['temp_inside'] = car_fuel.temp_inside.fillna(t_inside_mean)
print(car_fuel)
```

```
#To get complete info about data

car_fuel.info()
```

#5. Testing data

```
import pandas as pd

car_fuel = pd.read_csv("car_clean.csv")

x = ("rain", "sun", "refill_liters", "refill_gas")

y = list(x)

y[1] = "hot_sun"

x = tuple(y)

print(x)
```

#6. Statistical Summary of Speed column

```
import pandas as pd

car_fuel =pd.read_csv("car_clean.csv")

stat = car_fuel['speed'].describe()

print(stat)
```

#7. Mean, median and standard deviation

```
import pandas as pd

car_fuel = pd.read_csv("car_clean.csv")

car_fuel['temp_inside']=car_fuel['temp_inside'].str.replace(',','.').astype(float)

t_inside_mean = car_fuel.temp_inside.mean()

t_outside_mean = car_fuel.temp_outside.mean()

t_inside_median = car_fuel.temp_inside.median()
```

```
t_outside_median = car_fuel.temp_outside.median()
t_inside_std = car_fuel.temp_inside.std()
t_outside_std = car_fuel.temp_outside.std()
print(t_inside_mean)
print(t_outside_mean)
print(t_inside_median)
print(t_outside_median)
print(t_inside_std)
print(t_outside_std)
```

#8 Using numpy to recode

```
import pandas as pd
import numpy as np
car_fuel = pd.read_csv('car_clean.csv')
car_fuel['distance']=car_fuel['distance'].str.replace(',','.').astype(float)
car_fuel['consume']=car_fuel['consume'].str.replace(',','.').astype(float)
car_fuel['refill_liters']=car_fuel['refill_liters'].str.replace(',','.').astype(float)
car_fuel['temp_inside']=car_fuel['temp_inside'].str.replace(',','.').astype(float)
rl_mean = car_fuel.refill_liters.mean()
t_inside_mean = car_fuel.temp_inside.mean()
car_fuel['refill_liters'] = car_fuel.refill_liters.fillna(rl_mean)
car_fuel['temp_inside'] = car_fuel.temp_inside.fillna(t_inside_mean)
def recode_gas_type(gas):
    if gas == 'E10':
```

```
        return 0
    elif gas == 'SP98':
        return 1
    else:
        return np.nun
car_fuel['recode'] = car_fuel.gas_type.apply(recode_gas_type)
print(car_fuel)
```

#9. Speed Distribution - Histogram

```
import pandas as pd
import matplotlib.pyplot as plt
car_fuel=pd.read_csv("car_clean.csv")
car_fuel['speed'].plot.hist(bins=10,linewidth=5,color='cyan',edgecolor='black')
plt.xlabel('speed')
plt.title('Speed Distribution')
plt.show()
```

#10. Mostly used Fuel type - Bar Chart

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
car_fuel = pd.read_csv("car_clean.csv")
sns.countplot(car_fuel['gas_type'])
plt.title('Mostly used Gas')
plt.show()
```

#11. Scatter chart between distance and consumption

```
import pandas as pd
import matplotlib.pylab as plt
car_fuel = pd.read_csv("car_clean.csv")
car_fuel['distance']=car_fuel['distance'].str.replace(',','.').astype(float)
car_fuel['consume']=car_fuel['consume'].str.replace(',','.').astype(float)
car_fuel.plot(kind = 'scatter', x='distance', y='consume')
plt.show()
```

#12. Linear plot : With and without AC

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
car_fuel = pd.read_csv("car_clean.csv")
car_fuel['distance']=car_fuel['distance'].str.replace(',','.').astype(float)
car_fuel['consume']=car_fuel['consume'].str.replace(',','.').astype(float)
minac=car_fuel.query('AC==0.000000')
maxac=car_fuel.query('AC==1.000000')
sns.lmplot('distance','consume',data=minac)
sns.lmplot('distance','consume',data=maxac)
plt.show()
```

#13. Box plot for average and mean distance

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

```
import numpy as np
import seaborn as sns
car_fuel = pd.read_csv("car_clean.csv")
car_fuel['distance']=car_fuel['distance'].str.replace(',','.').astype(float)
sns.boxplot(x='gas_type', y='distance', data=car_fuel)
plt.title("Boxplot of the distance travelled by  Gas Type")
plt.show()
```

NOTE:

We used the below 6 python procedures as mentioned:

1. Lists

2. Functions

3. Tuple

4. Pandas Dataframe

5. Files

6. Strings