Name: Nguyễn Chí Trung Student ID: ITDSIU19024

REPORT
DATA ANALYSIS CHAP 4-5

Q1. Define the random variables of Gender, Type, Purchased, VehicleAge, Mileage, and MPG. **Find** their probability mass/density functions. **Program** to compute means, variances, and standard deviations of the random variables, and display the graphs of probability mass/density functions. **(70pts)**

- Firstly, I import all libraries for the exercise:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sbs
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

- Then is to import csv file into python and set their attribute, use 6 columns *Gender, Type, Purchase, VehicleAge, Mileage,* and *MPG*

```
dt = pd.read_csv('/Users/trungnguyen/Downloads/Sem1-year2/Data
Analys/AutoSurvey.csv',sep=',',header=0,usecols= [0,1,2,3,4,5],
dtype={0:str, 1:str, 2:str, 3:np.float64, 4:np.int32, 5:np.float64})
dt.columns = ['Gender','Type', 'Purchased', 'VehicleAge', 'Mileage', 'MPG']
```

- Secondly, since the attribute of *Gender, Type,* and *Purchase* is not numerical so it is impossible to calculate their mean, variance, or standard deviation. The solution here is to encoding categorical variables, for example, let "1" present for "Male" and "0" for "Female", the code as below:

```
gender_var = dt['Gender']
gender_var[gender_var == 'Male'] = 1
gender_var[gender_var == 'Female'] = 0

type_var = dt['Type']
type_var[type_var == 'Small'] = 1
type_var[type_var == 'Mid-size'] = 2
type_var[type_var == 'Minivan'] = 3
type_var[type_var == 'Small SUV'] = 4
type_var[type_var == 'Large SUV'] = 5

purchase_var = dt['Purchased']
purchase_var[purchase_var == 'Used'] =  1
```

```
purchase_var[purchase_var == 'New'] =
```

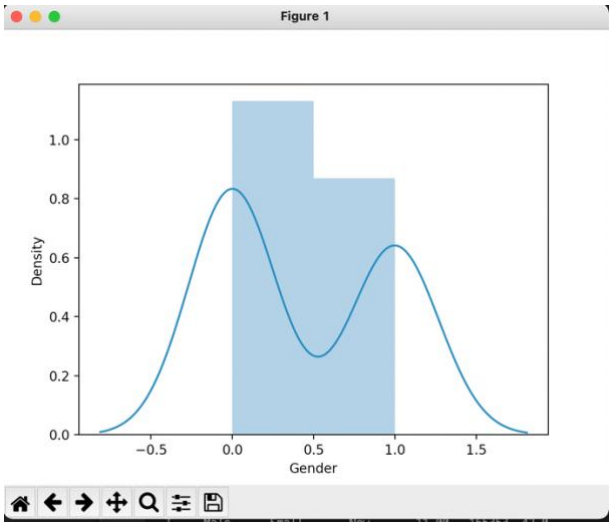
- Then is to set value of *VehicleAge, Mileage*, and *MPG* and let all of that new data frame value into new variable that called *variable*, this variable also the probability density function

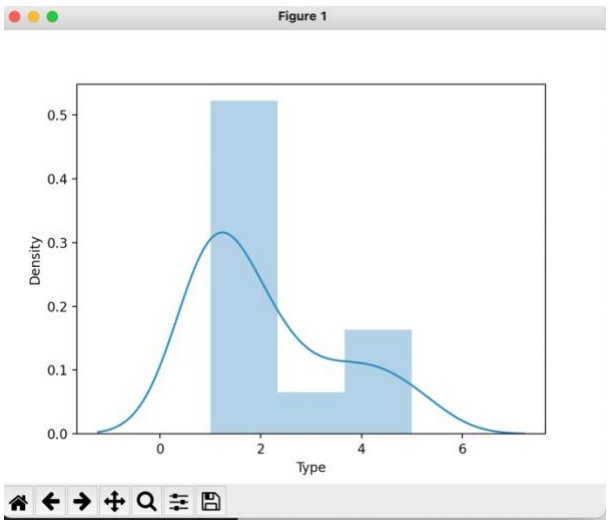
```
vehicleAge_var = dt['VehicleAge']

mileage_var = dt['Mileage']

mpg_var = dt['MPG']

variable = [gender_var, type_var, purchase_var, vehicleAge_var, mileage_var, mpg_var]
```

- Finally is calculate descriptive statistic of each function and display each chart respectively.

```
for var in variable:
    #Name of chart
    print(var.name)
    #Mean of column
    print(f'Mean = {np.mean(var):.2f}')
    #Variance of column
    print(f'Variance = {np.std(var):.2f}')
    #Standard deviation of column
    print(f'Standard deviation = {np.var(var):.2f}')
    #Show histogram and cumulative frequency line
    sbs.distplot(var)
    plt.show()
```

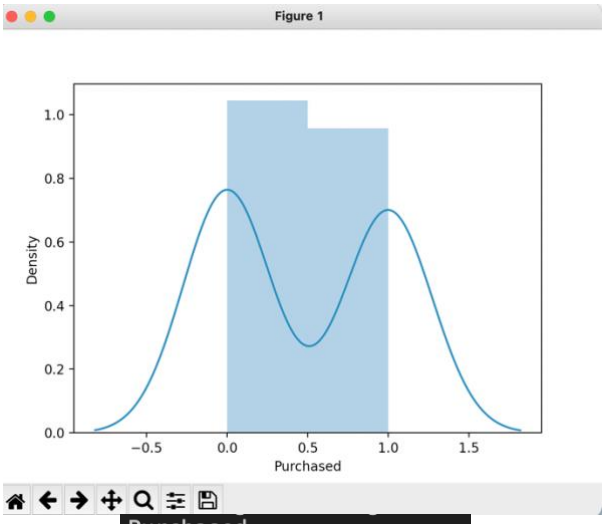
- Running this code thought visual studio code give us this output:



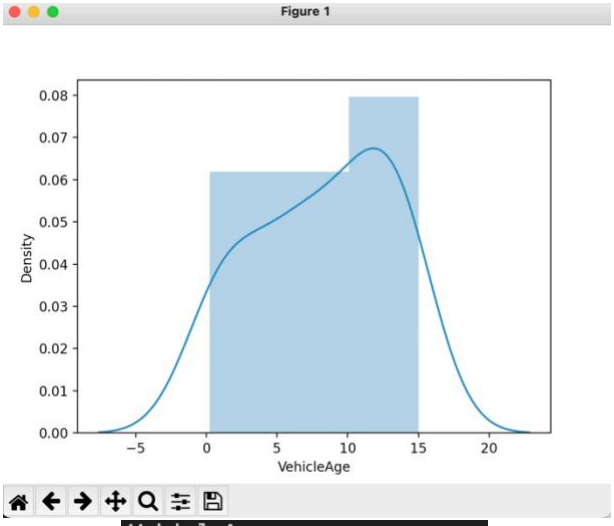


```
Gender
Mean = 0.43
Variance = 0.50
Standard deviation = 0.25
```

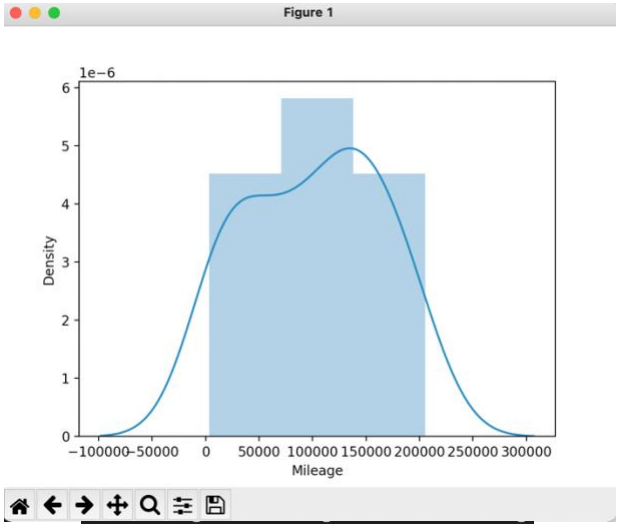
```
Type
Mean = 2.13
Variance = 1.36
Standard deviation = 1.85
```



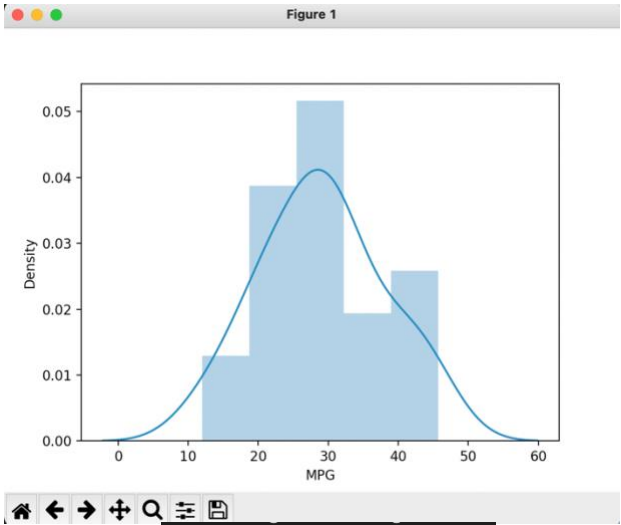

```
Purchased
Mean = 0.48
Variance = 0.50
Standard deviation = 0.25
```




```
VehicleAge
Mean = 8.21
Variance = 4.78
Standard deviation = 22.84
```




```
Mileage
Mean = 100914.52
Variance = 61969.80
Standard deviation = 3840256641.21
```




```
MPG
Mean = 29.08
Variance = 8.66
Standard deviation = 74.97
```

Q2. Assume the random variables of Gender, Type, Purchased, VehicleAge, Mileage, and MPG are jointly distributed. **Find** the marginal probability density function of MPG. **Program** to estimate the probability of MPG. **(30pts)**

First, I calculate jointly distributed of these data. Since this is a set of collective variable so we need to use multidimensional histogram (histogramdd in numpy) to record, then divided with the sum.

```
joint_prob,ranges=np.histogramdd(variable)
joint_prob /= joint_prob.sum()
ranges=ranges[-1]
```

- Second is to calculate marginal probability distribution of MPG and also estimate the probability of MPG

```
marginal_prob=[]
for i in range(10):
    marginal_prob.append(np.sum(joint_prob[:, :, :, :, :, i]))

for (x, y_x) in enumerate(marginal_prob):
    print(f'P({ranges[x]:.2f}  < MPG <  {ranges[x + 1]:.2f}) = {y_x:.2f}')
```

(Using enumerate to mark index in a loop automatically and avoid mistaking)
Result:

```
P(12.00  < MPG <  15.37) = 0.09
P(15.37  < MPG <  18.74) = 0.00
P(18.74  < MPG <  22.11) = 0.17
P(22.11  < MPG <  25.48) = 0.09
P(25.48  < MPG <  28.85) = 0.17
P(28.85  < MPG <  32.22) = 0.17
P(32.22  < MPG <  35.59) = 0.09
P(35.59  < MPG <  38.96) = 0.04
P(38.96  < MPG <  42.33) = 0.09
P(42.33  < MPG <  45.70) = 0.09
```

Q3. Predict the MPG of the last 3 records using the above program and compare the predicted results with the actual values. **(10pts)**

For this question, I will use the linear regression, in form of y = aX + b.
First, to make it clear, our original sample that will be used in this question is:

	Gender	Type	Purchased	VehicleAge	Mileage	MPG
1	Female	Mid-size	New	1.00	23970	43.4
2	Male	Small	New	7.00	77392	24.0
3	Female	Large SUV	Used	14.00	185397	15.2
4	Female	Small	New	2.00	26001	37.0
5	Female	Minivan	New	9.00	180643	20.0
6	Male	Small	Used	6.00	72083	45.7
7	Male	Small	New	11.00	165353	42.0
8	Male	Small	Used	13.00	205288	33.0
9	Female	Small	New	7.00	142897	31.0
10	Male	Minivan	Used	14.00	182584	12.0
11	Male	Small SUV	Used	13.00	140479	20.0
12	Female	Small	New	2.00	22114	28.0
13	Female	Mid-size	New	0.25	3454	28.3
14	Female	Large SUV	New	7.00	130905	21.0
15	Female	Small	Used	10.00	105628	35.0
16	Female	Small	New	5.00	48678	30.4
17	Male	Mid-size	New	0.50	6849	40.2
18	Female	Small	Used	10.00	137941	30.0
19	Female	Small SUV	New	4.00	29823	24.9
20	Male	Small SUV	Used	14.00	85763	21.0

Last 3 value of MPG is 30.0, 24.9 and 21.0
Then, building model:

```
dt1 = dt.tail()
X = dt1.iloc[:, :-1].values
y = dt1.iloc[:, -1].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 3, random_state = 0)
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

And last, predict and compare the prediction with actual value:

```
y_pred = regressor.predict(X_test)
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
print(df)
```

Output:
Since our sample was small, just 20, so this prediction is acceptable

```
   Actual  Predicted
0    21.0  26.947766
1    30.0  31.315496
2    24.9  22.265125
```