aerofit-project

July 30, 2023

[29]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('aerofit.csv')
```

0.1 Basic Analysis

[30]:
```
df.head(5)
```

[30]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	KP281	18	Male	14	Single	3	4	29562	112
1	KP281	19	Male	15	Single	2	3	31836	75
2	KP281	19	Female	14	Partnered	4	3	30699	66
3	KP281	19	Male	12	Single	3	3	32973	85
4	KP281	20	Male	13	Partnered	4	2	35247	47

[31]:
```
df.shape
```

[31]: (180, 9)

[32]:
```
df.columns
```

[32]:
```
Index(['Product', 'Age', 'Gender', 'Education', 'MaritalStatus', 'Usage',
       'Fitness', 'Income', 'Miles'],
      dtype='object')
```

Statistical Summary

[33]:
```
df.describe()
```

[33]:

	Age	Education	Usage	Fitness	Income	\
count	180.000000	180.000000	180.000000	180.000000	180.000000	
mean	28.788889	15.572222	3.455556	3.311111	53719.577778	
std	6.943498	1.617055	1.084797	0.958869	16506.684226	
min	18.000000	12.000000	2.000000	1.000000	29562.000000	
25%	24.000000	14.000000	3.000000	3.000000	44058.750000	

```
50%      26.000000   16.000000    3.000000    3.000000    50596.500000
75%      33.000000   16.000000    4.000000    4.000000    58668.000000
max      50.000000   21.000000    7.000000    5.000000   104581.000000

            Miles
count  180.000000
mean   103.194444
std     51.863605
min     21.000000
25%     66.000000
50%     94.000000
75%    114.750000
max    360.000000
```

[34]:
```
df.describe(include=object)
```

[34]:
```
       Product Gender MaritalStatus
count      180    180           180
unique       3      2             2
top      KP281   Male     Partnered
freq        80    104           107
```

[35]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Product        180 non-null    object
 1   Age            180 non-null    int64
 2   Gender         180 non-null    object
 3   Education      180 non-null    int64
 4   MaritalStatus  180 non-null    object
 5   Usage          180 non-null    int64
 6   Fitness        180 non-null    int64
 7   Income         180 non-null    int64
 8   Miles          180 non-null    int64
dtypes: int64(6), object(3)
memory usage: 12.8+ KB
```

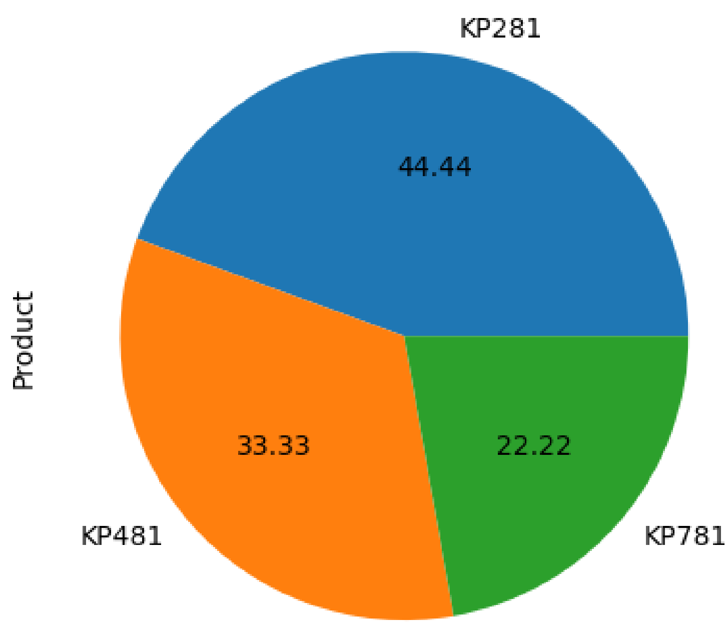
- **No null value present in given dataset**

0.2 Value counts for Categorical attribute

[36]:
```
print(df['Product'].value_counts())
df['Product'].value_counts().plot(kind='pie',autopct="%.2f")
```

```
KP281    80
KP481    60
KP781    40
Name: Product, dtype: int64
```

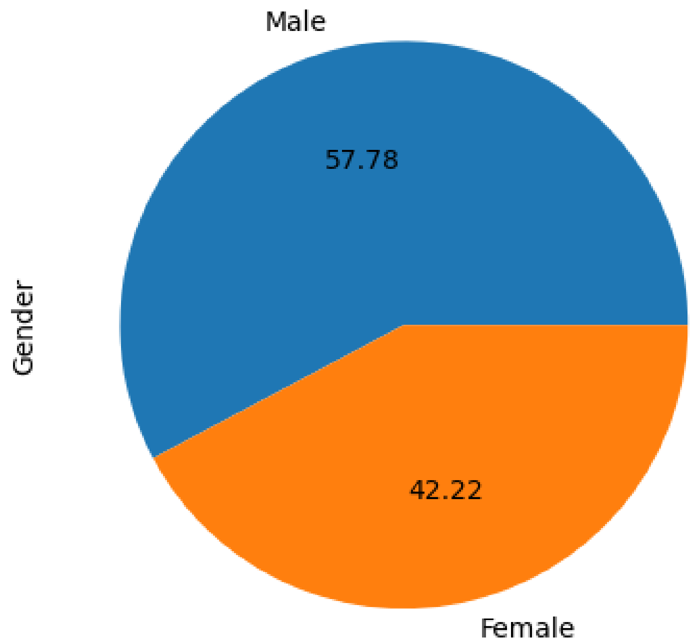
[36]: <Axes: ylabel='Product'>



[37]:
```
print(df['Gender'].value_counts())
df['Gender'].value_counts().plot(kind='pie',autopct="%.2f").plot()
```

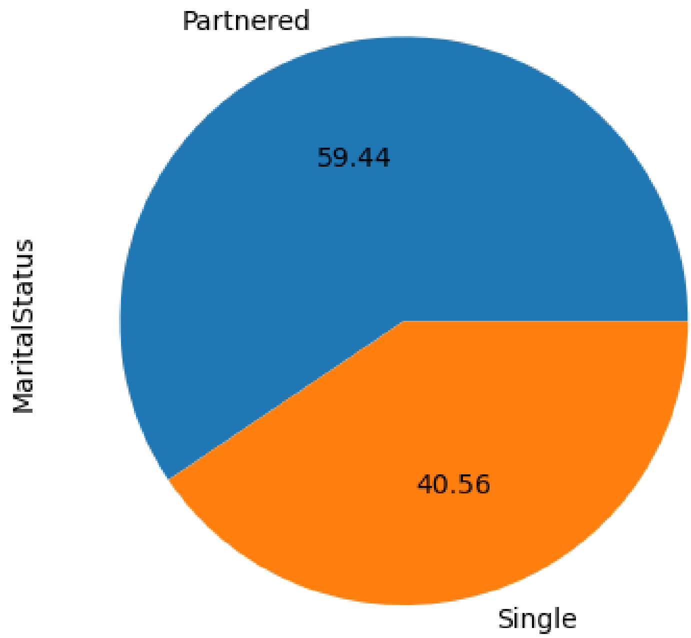
```
Male      104
Female     76
Name: Gender, dtype: int64
```

[37]: []



[38]:
```
df['MaritalStatus'].value_counts()
df['MaritalStatus'].value_counts().plot(kind='pie',autopct="%.2f")
```

[38]: <Axes: ylabel='MaritalStatus'>



[39]:
```
df['Usage'].value_counts()
```

[39]:
```
3    69
4    52
2    33
5    17
6     7
7     2
Name: Usage, dtype: int64
```

[40]:
```
fitness_counts = df['Fitness'].value_counts()
fitness_counts
```

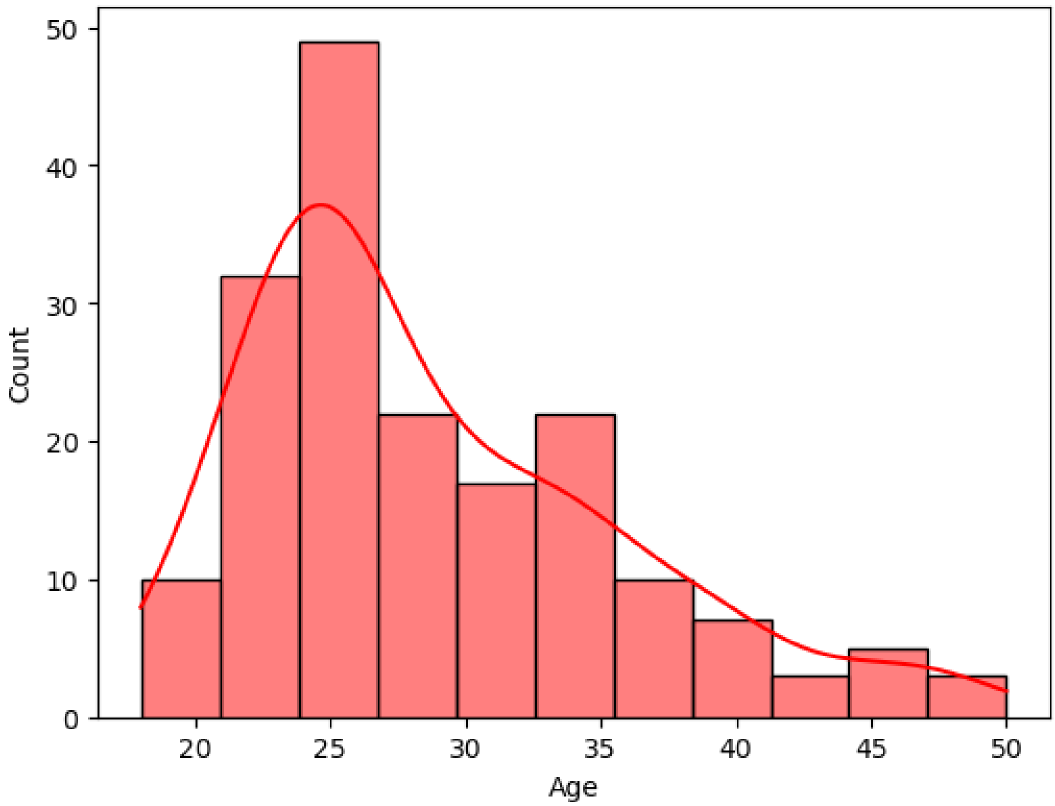
[40]:
```
3    97
5    31
2    26
4    24
1     2
Name: Fitness, dtype: int64
```

0.3 Univariate Analysis

0.3.1 Age distribution

[46]:
```
plt.figure()
sns.histplot(data = df, x = 'Age', kde = True, color = 'red')
plt.plot()
```

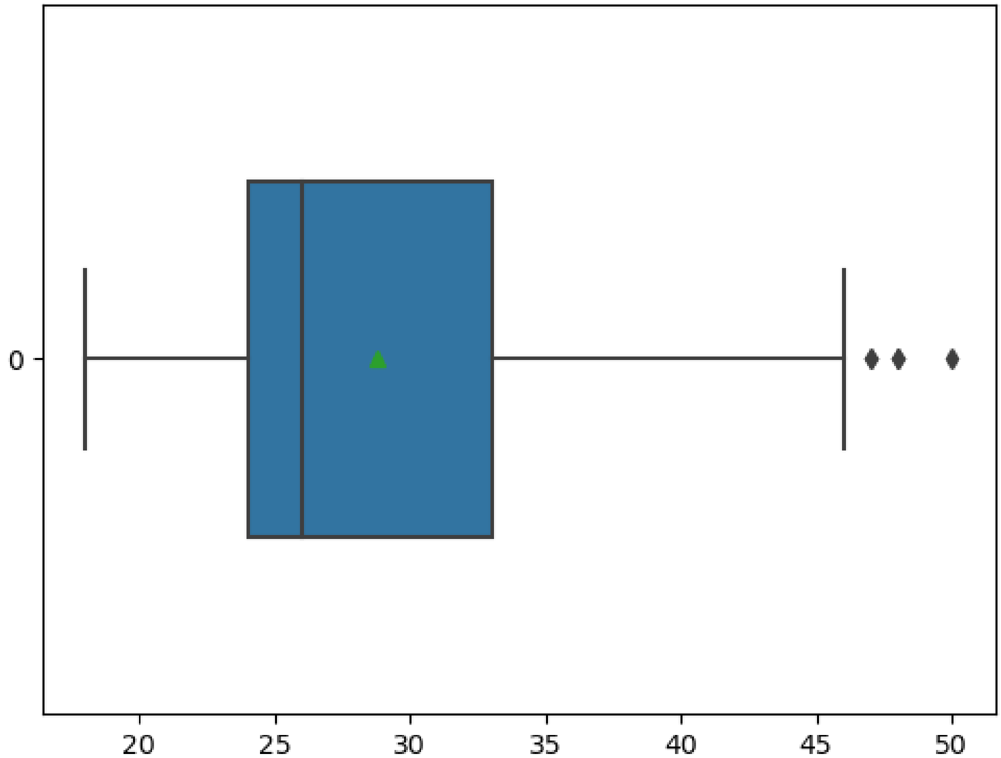
[46]: []



- The age of most of the customer(more than 75%) lies between 20 and 30.
- Less than 5% customers has age > 40

[42]:
```
sns.boxplot(data = df['Age'], width = 0.5, orient = 'h', showmeans = True)
plt.plot()
```

[42]: []



- There are some outliers (Age > 40)

[44]:
```
result = df[(df["Age"] >= 20) & (df['Age'] <= 35)]['Product'].count() / len(df)␣
↪* 100
"%% of customers whose age is between 20 and 35 is %.2f%%"%(result)
```

[44]: '% of customers whose age is between 20 and 35 is 81.67%'

[45]:
```
data = df['Age']
print('Mean : ', data.mean())
print('Median : ', data.median())
q1 = data.quantile(0.25)
q3 = data.quantile(0.75)
print("1st Quartile : ", q1)
print("3rd Quartile : ", q3)
iqr = q3 - q1
print('Innerquartile Range : ', iqr)
upper = q3 + 1.5 * iqr
lower = q1 - 1.5 * iqr
print("Upper Bound : ", upper)
print('Lower Bound : ', lower)
outliers = data[(data > upper) | (data < lower)]
```

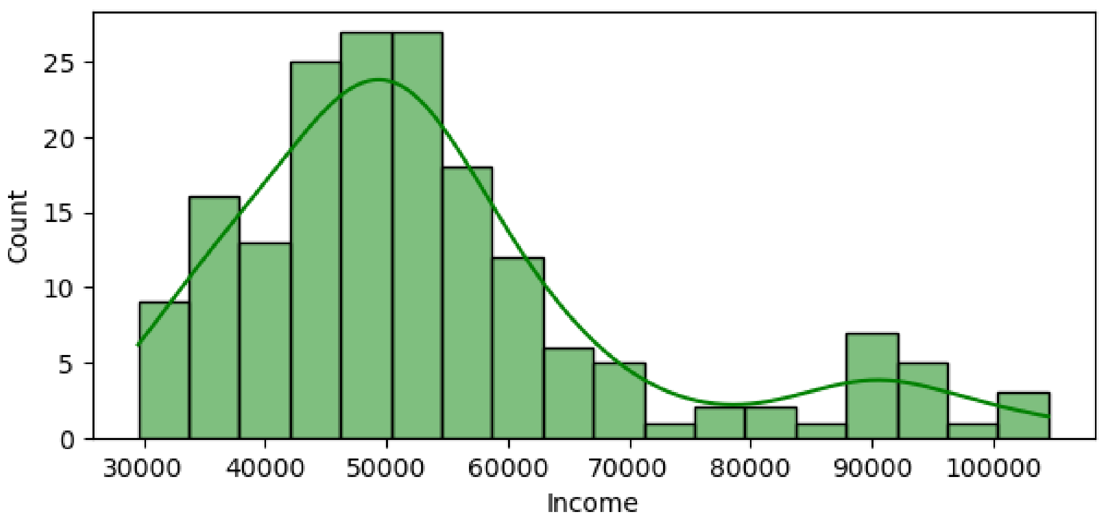
```
print("Outliers : ", sorted(outliers))
len_outliers = len((data[(data > upper) | (data < lower)]))
print('No of Outliers : ', len_outliers)
```

```
Mean :  28.788888888888888
Median :  26.0
1st Quartile :  24.0
3rd Quartile :  33.0
Innerquartile Range :  9.0
Upper Bound :  46.5
Lower Bound :  10.5
Outliers :  [47, 47, 48, 48, 50]
No of Outliers :  5
```

0.3.2 Income distribution

[52]:
```
plt.figure(figsize = (7, 3))
sns.histplot(data = df, x = 'Income', kde = True, bins = 18, color = 'green')
plt.plot()
```

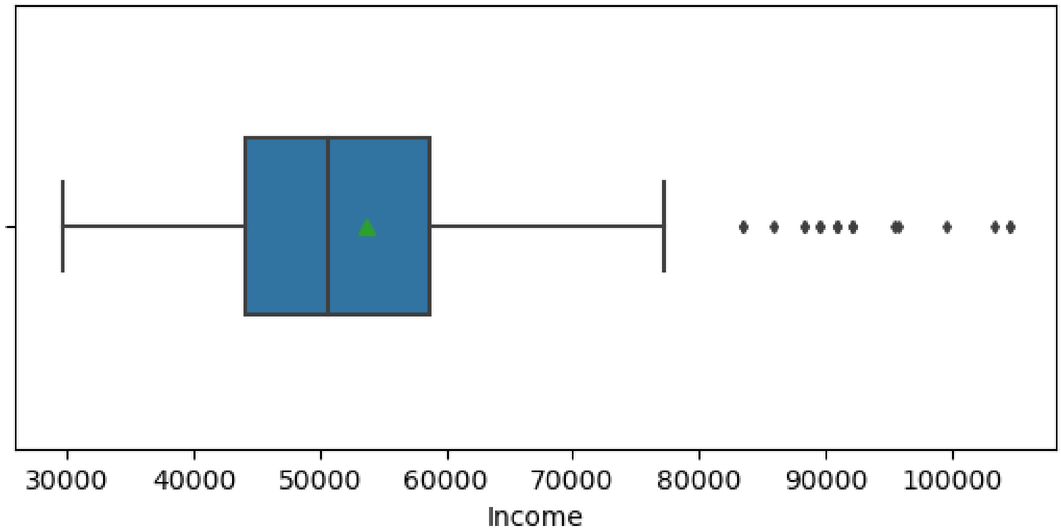
[52]: []



- 75% of customers have income < 59000
- more than 90% customer have their income between 30000 to 60000

[51]:
```
plt.figure(figsize = (7, 3))
sns.boxplot(data = df, x = 'Income', width = 0.4, orient = 'h', showmeans =␣
↪True, fliersize = 3)
plt.plot()
```

[51]: []



[103]:
```
data = df['Income']
print('Mean : ', data.mean())
print('Median : ', data.median())
q1 = data.quantile(0.25)
q3 = data.quantile(0.75)
print("1st Quartile : ", q1)
print("3rd Quartile : ", q3)
iqr = q3 - q1
print('Innerquartile Range : ', iqr)
upper = q3 + 1.5 * iqr
lower = q1 - 1.5 * iqr
print("Upper Bound : ", upper)
print('Lower Bound : ', lower)
outliers = data[(data > upper) | (data < lower)]
print("Outliers : ", sorted(outliers))
len_outliers = len((data[(data > upper) | (data < lower)]))
print('No of Outliers : ', len_outliers)
```

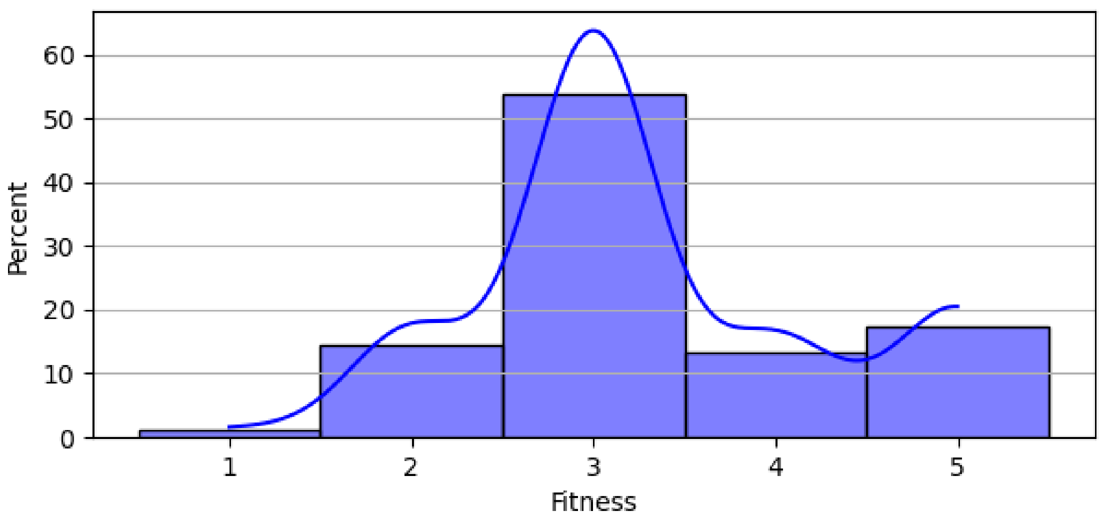
```
Mean :  53719.57777777778
Median :  50596.5
1st Quartile :  44058.75
3rd Quartile :  58668.0
Innerquartile Range :  14609.25
Upper Bound :  80581.875
Lower Bound :  22144.875
Outliers :  [83416, 83416, 85906, 88396, 88396, 89641, 89641, 90886, 90886,
90886, 92131, 92131, 92131, 95508, 95866, 99601, 103336, 104581, 104581]
```

```
No of Outliers :  19
```

0.3.3 Fitness scale distribution

[54]:
```python
plt.figure(figsize = (7, 3))
sns.histplot(data = df, x = 'Fitness', discrete = True, kde = True, stat =␣
↪'percent', color = 'blue')
plt.yticks(np.arange(0, 101, 10))
plt.grid(axis = 'y')
plt.plot()
```

[54]: []

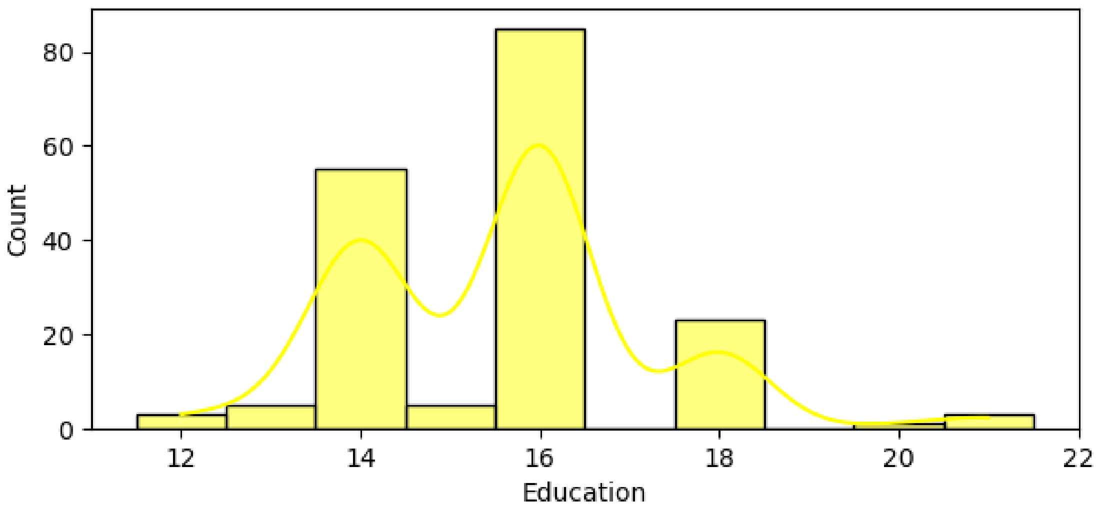


- More than 50% of customers have fitness scale 3

0.3.4 Education distribution

[66]:
```python
plt.figure(figsize = (7, 3))
sns.histplot(data = df, x = 'Education', discrete = True, kde = True, color =␣
↪'yellow')
plt.plot()
```

[66]: []

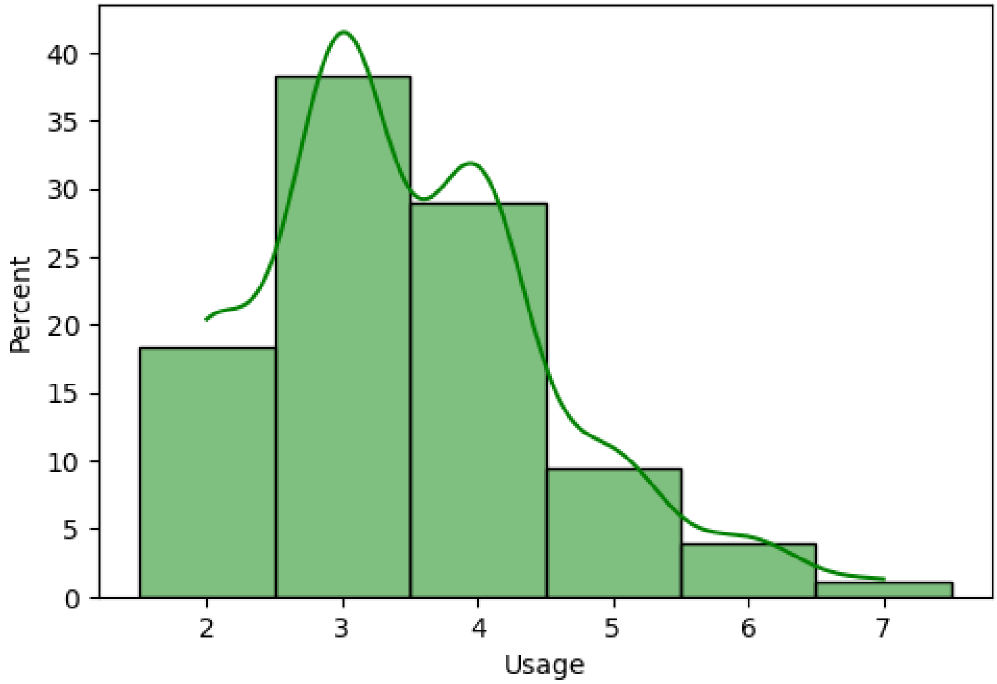


- Most of the customers have 16 years of education followed by 14 and 18 years.

0.3.5 Usage distribution

[64]:
```
plt.figure(figsize = (6, 4))
sns.histplot(data = df, x = 'Usage', kde = True, stat = 'percent', discrete =␣
↪True, color = 'green')
plt.plot()
```

[64]: []

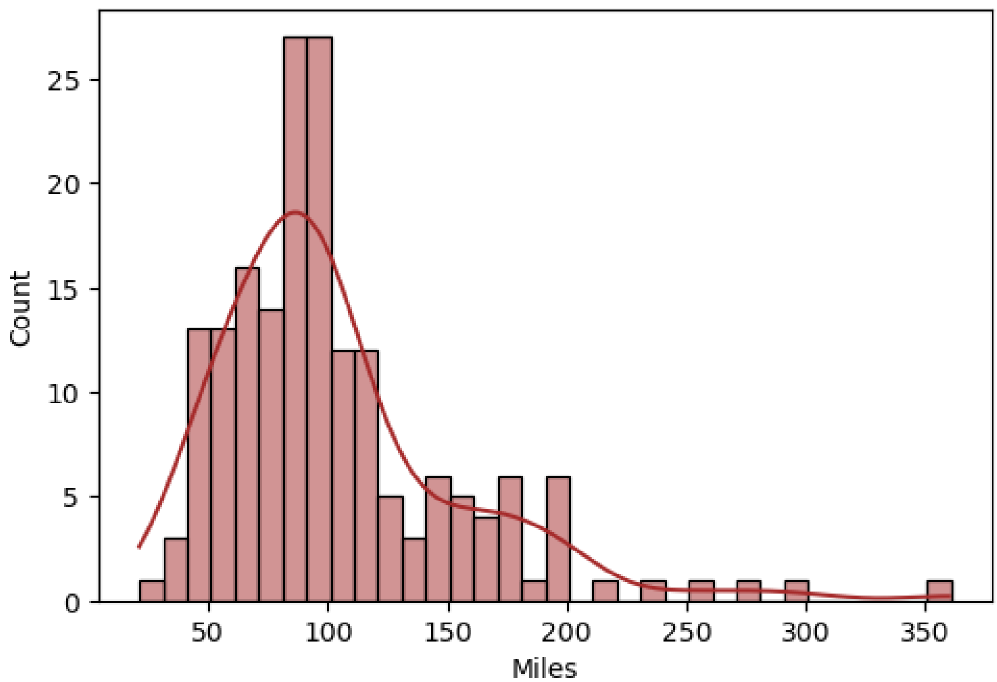


- Most of customers(almost 70%) use tredmil 3 or 4 days a week

0.3.6 Miles Distribution

[67]:
```
plt.figure(figsize = (6, 4))
sns.histplot(data = df, x = 'Miles', kde = True, binwidth = 10, color = 'brown')
plt.plot()
```

[67]: []

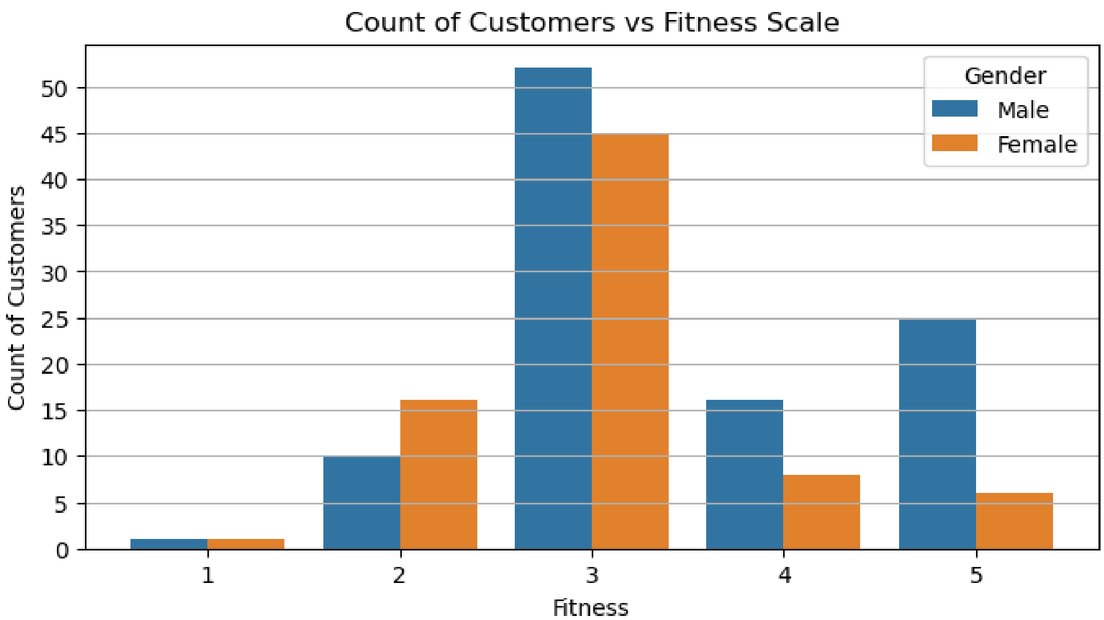


- Most of the customers walks between 50 to 120 miles per week

0.4 Bivariate Analysis

[72]:
```
plt.figure(figsize = (8, 4))
plt.title('Count of Customers vs Fitness Scale')
sns.countplot(data = df, x = 'Fitness', hue = 'Gender')
plt.grid(axis = 'y')
plt.yticks(np.arange(0, 60, 5))
plt.ylabel('Count of Customers')
plt.plot()
```

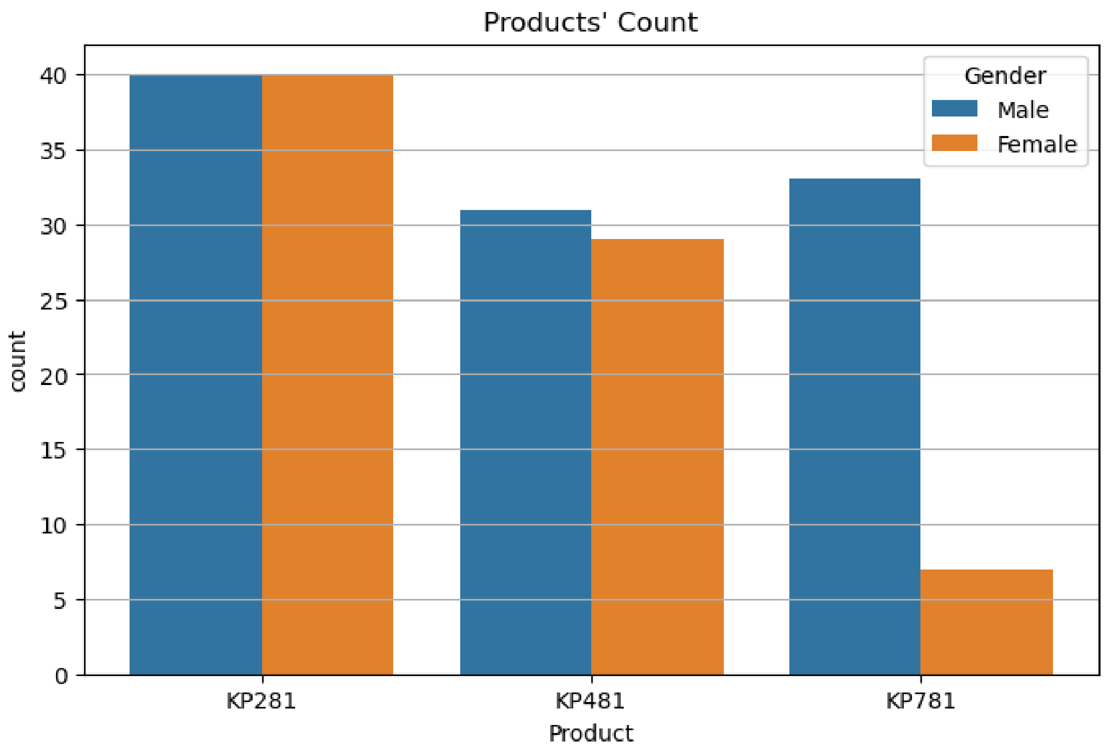
[72]: []



- More than half male and female has fitness level 3
- For fitness level 4 and 5 males are more than 3 times compasre to female.

[74]:
```
plt.figure(figsize = (8, 5))
plt.title("Products' Count")
sns.countplot(data = df, x = 'Product', hue = 'Gender')
plt.grid(axis = 'y')
plt.plot()
```

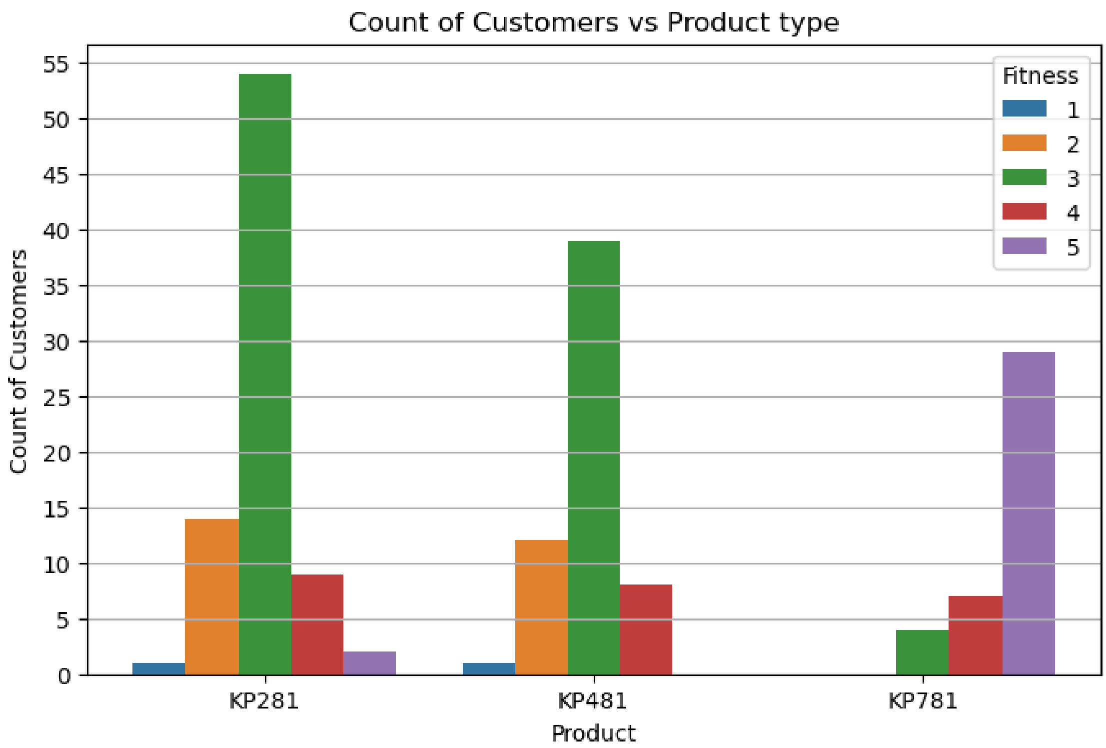
[74]: []



- For products KP281 and KP481 number of male and female customers are almost same but for KP781 number male customers are six times more than female customer.

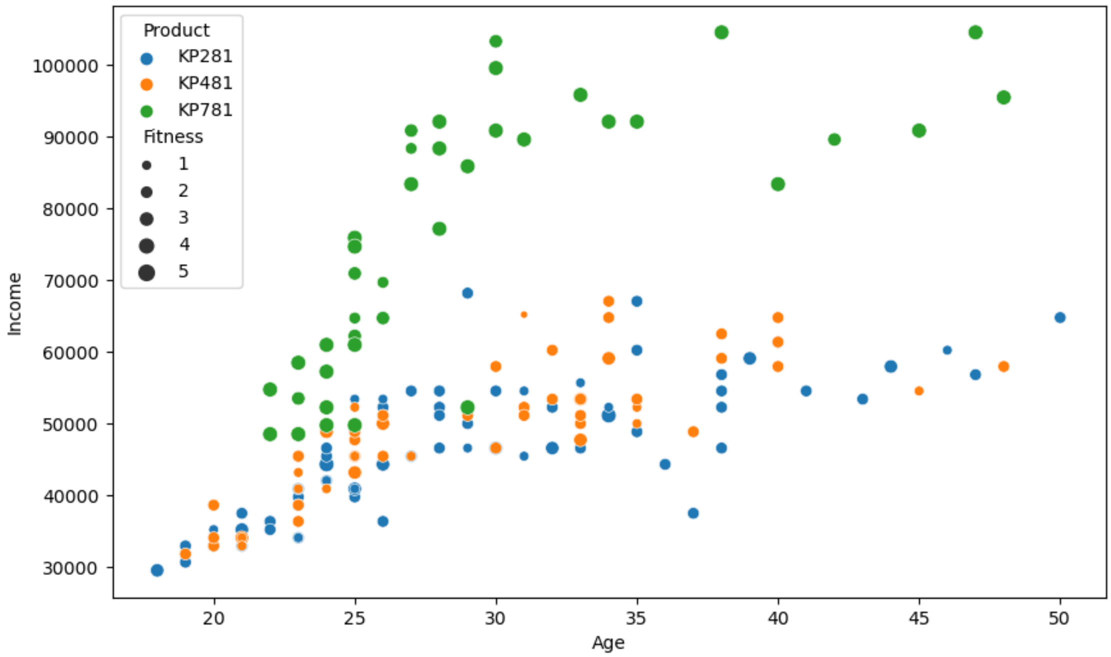
[76]:
```
# For Male, different product categories and
plt.figure(figsize = (8, 5))
plt.title("Count of Customers vs Product type")
plt.yticks(np.arange(0, 60, 5))
sns.countplot(data = df, x = 'Product', hue = 'Fitness')
plt.ylabel('Count of Customers')
plt.grid(axis = 'y')
plt.plot()
```

[76]: []



- For product KP281 and KP481 most of customer fitness level is 3 but fro KP781 it's 5.

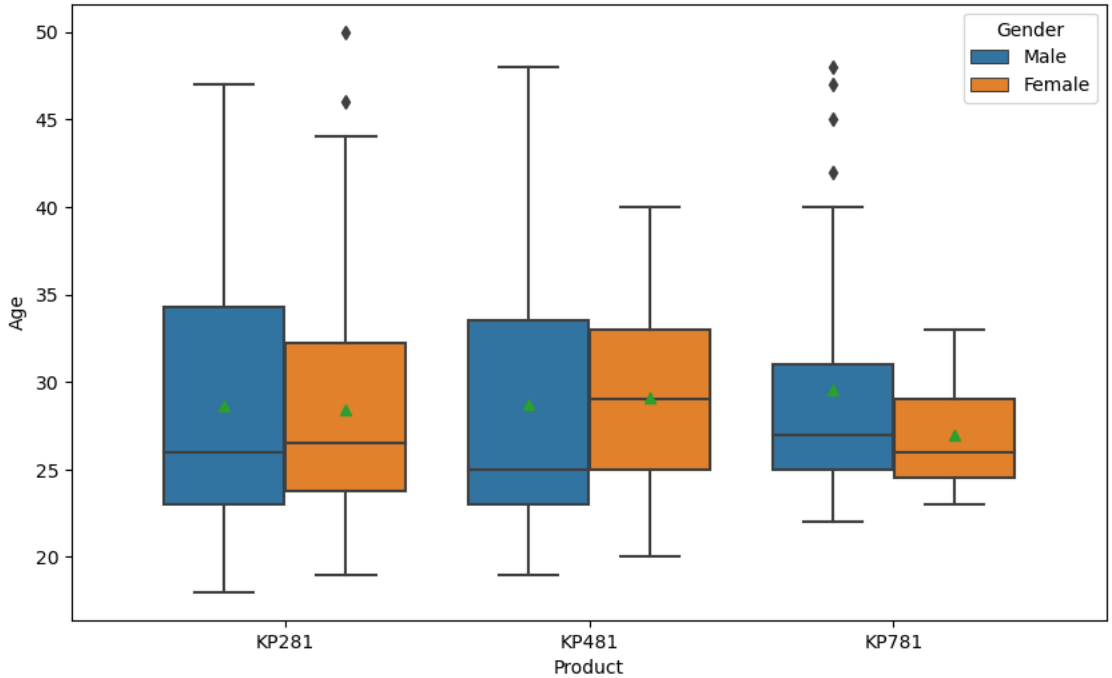
[79]:
```
plt.figure(figsize = (10, 6))
sns.scatterplot(data = df, x= 'Age', y = 'Income', hue = 'Product', size =␣
↪'Fitness')
plt.show()
```



- Customers with high income or high fitnees are more likely to buy KP781
- Customers with low fitness or income buys other two products

[81]:
```
plt.figure(figsize = (10, 6))
sns.boxplot(data = df, x = 'Product', y = 'Age', hue = 'Gender', showmeans =␣
↪True)
plt.plot()
```

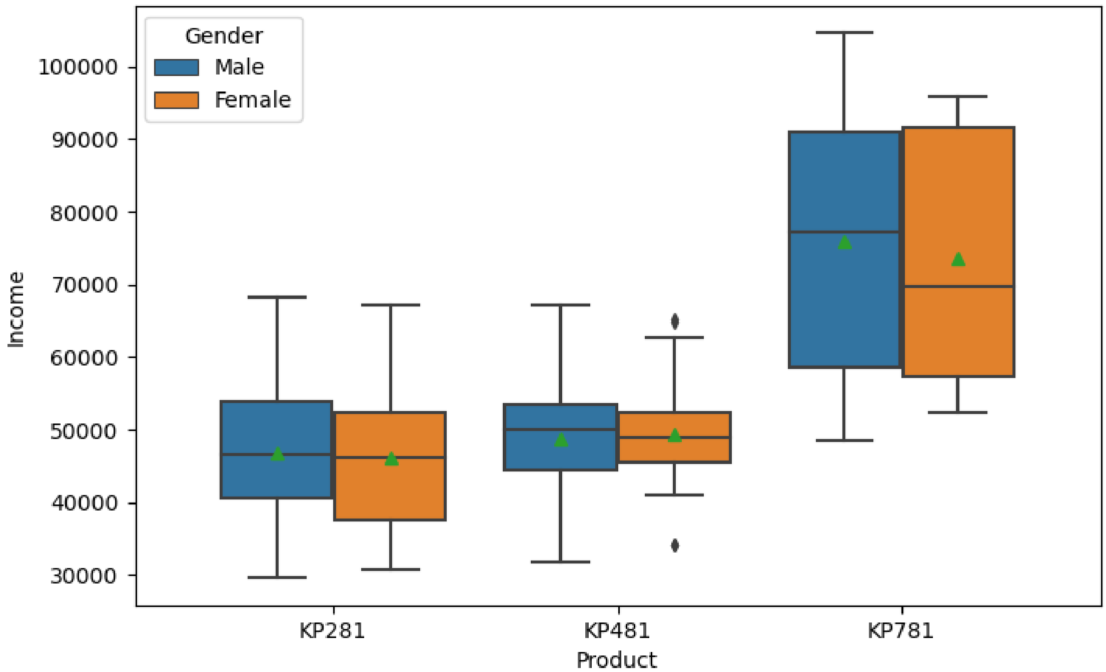
[81]: []



- Age range of male customer is more than female range for all three products.
- Age range of customers who buys KP781 is less than other two products.

[83]:
```
plt.figure(figsize = (8, 5))
sns.boxplot(data = df, x = 'Product', y = 'Income', hue = 'Gender', showmeans =␣
↪True, fliersize = 4)
plt.plot()
```

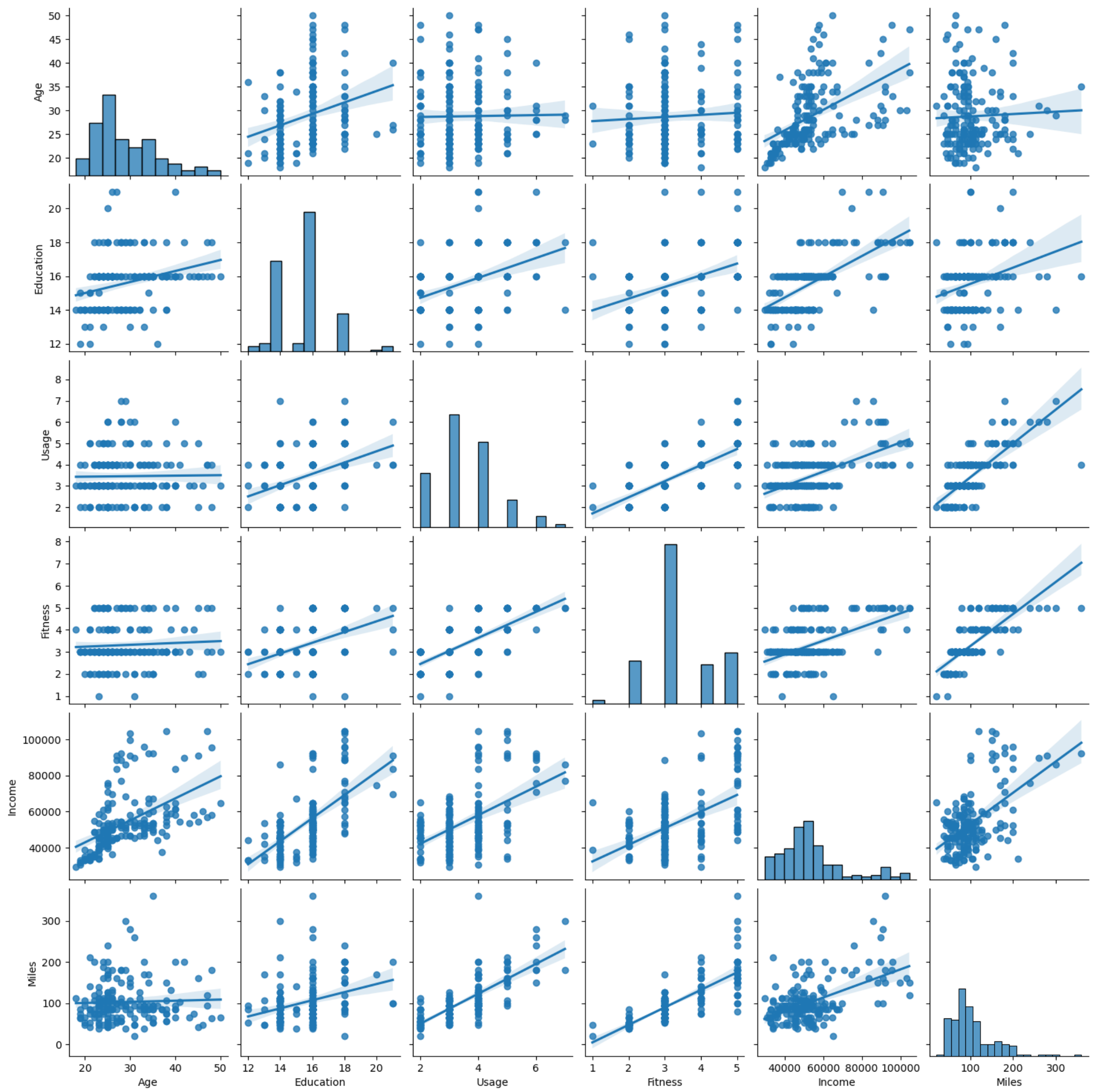
[83]: []



- Income range, mean,meadian is higher for customer who buys KP781

[84]:
```
sns.pairplot(data = df, kind = 'reg')
plt.plot()
```

[84]: []



[86]:
```
df_corr = df.corr()
df_corr
```

```
C:\Users\divya\AppData\Local\Temp\ipykernel_26612\1378791828.py:1:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only valid
columns or specify the value of numeric_only to silence this warning.
  df_corr = df.corr()
```

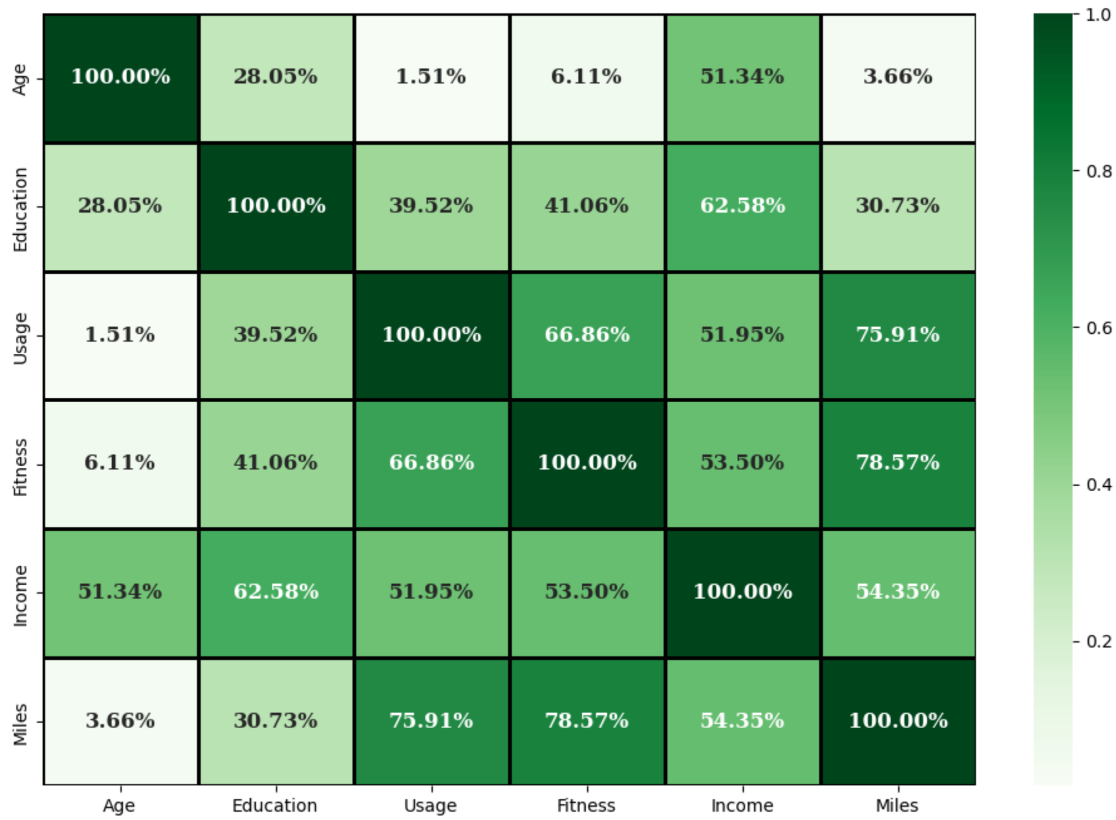
[86]:

	Age	Education	Usage	Fitness	Income	Miles
Age	1.000000	0.280496	0.015064	0.061105	0.513414	0.036618
Education	0.280496	1.000000	0.395155	0.410581	0.625827	0.307284
Usage	0.015064	0.395155	1.000000	0.668606	0.519537	0.759130

```
Fitness    0.061105   0.410581   0.668606   1.000000   0.535005   0.785702
Income     0.513414   0.625827   0.519537   0.535005   1.000000   0.543473
Miles      0.036618   0.307284   0.759130   0.785702   0.543473   1.000000
```

[87]:
```python
plt.figure(figsize = (12, 8))
sns.heatmap(data = df_corr,
            annot = True,
            fmt = '.2%',
            cmap='Greens',
            linewidth = 2,
            linecolor = 'black',
            annot_kws = {'fontsize' : 'large',
                        'fontfamily' : 'serif',
                        'fontweight': 'bold'})
plt.plot()
```

[87]: []



- Customers with high fitness are likely to walk more miles per week and use tredmil more frequently in a week.
- Customers who has more education have higher income

[88]:
```python
print(pd.crosstab(index = df['Product'], columns = df['Gender'], margins =␣
 ↪True))
print()
print('-' * 26)
print()
print("Product-wise normalization : ")
print(np.round(pd.crosstab(index = df['Product'], columns = df['Gender'],␣
 ↪normalize = 'index') * 100, 2))
print()
print('-' * 23)
print()
print("Gender-wise normalization : ")
print(np.round(pd.crosstab(index = df['Product'], columns = df['Gender'],␣
 ↪normalize = 'columns') * 100, 2))
```

```
Gender   Female  Male  All
Product
KP281        40    40   80
KP481        29    31   60
KP781         7    33   40
All          76   104  180

--------------------------

Product-wise normalization :
Gender   Female   Male
Product
KP281     50.00  50.00
KP481     48.33  51.67
KP781     17.50  82.50

-----------------------

Gender-wise normalization :
Gender   Female   Male
Product
KP281     52.63  38.46
KP481     38.16  29.81
KP781      9.21  31.73
```

- 82% of customers who baught KP781 are male.
- Females mostly buy KP281 or KP481

0.4.1 Prob of buying prodect given that customer's gender

[89]:
```
products = df['Product'].unique()
genders = df['Gender'].unique()
for i in genders:
    for j in products:
        prob = len(df[(df['Gender'] == i) & (df['Product'] == j)]) /␣
↪len(df[df['Gender'] == i])
        prob = np.round(prob * 100, 2)
        print("Probability of buying '{}' provided the customer is {} is {}% ".
↪format(j, i, prob))
        print()
```

```
Probability of buying 'KP281' provided the customer is Male is 38.46%

Probability of buying 'KP481' provided the customer is Male is 29.81%

Probability of buying 'KP781' provided the customer is Male is 31.73%

Probability of buying 'KP281' provided the customer is Female is 52.63%

Probability of buying 'KP481' provided the customer is Female is 38.16%

Probability of buying 'KP781' provided the customer is Female is 9.21%
```

0.4.2 Prob of customer belong to particular gender given that he bought some product

[90]:
```
products = df['Product'].unique()
genders = df['Gender'].unique()
for i in genders:
    for j in products:
        prob = len(df[(df['Gender'] == i) & (df['Product'] == j)]) /␣
↪len(df[df['Product'] == j])
        prob = np.round(prob * 100, 2)
        print("Probability that the customer is {} provided {} was bought is␣
↪{}% ".format(i, j, prob))
        print()
```

```
Probability that the customer is Male provided KP281 was bought is 50.0%

Probability that the customer is Male provided KP481 was bought is 51.67%

Probability that the customer is Male provided KP781 was bought is 82.5%

Probability that the customer is Female provided KP281 was bought is 50.0%
```

```
Probability that the customer is Female provided KP481 was bought is 48.33%

Probability that the customer is Female provided KP781 was bought is 17.5%
```

[91]:
```python
print(pd.crosstab(index = df['Product'], columns = df['MaritalStatus'], margins␣
 ↪= True))
print()
print('-' * 37)
print()
print("Product-wise normalization : ")
print(np.round(pd.crosstab(index = df['Product'], columns =␣
 ↪df['MaritalStatus'], normalize = 'index') * 100, 2))
print()
print('-' * 33)
print()
print("Marital Status-wise normalization : ")
print(np.round(pd.crosstab(index = df['Product'], columns =␣
 ↪df['MaritalStatus'], normalize = 'columns') * 100, 2))
```

```
MaritalStatus  Partnered  Single  All
Product
KP281                 48      32   80
KP481                 36      24   60
KP781                 23      17   40
All                  107      73  180

-------------------------------------

Product-wise normalization : 
MaritalStatus  Partnered  Single
Product
KP281               60.0    40.0
KP481               60.0    40.0
KP781               57.5    42.5

---------------------------------

Marital Status-wise normalization : 
MaritalStatus  Partnered  Single
Product
KP281              44.86   43.84
KP481              33.64   32.88
KP781              21.50   23.29
```

0.4.3 Prob of buying product provided customer's maritalStatus

[92]:
```
products = df['Product'].unique()
statuses = df['MaritalStatus'].unique()
for i in statuses:
    if i != 'Single':
        print('-' * 76)
    for j in products:
        prob = len(df[(df['MaritalStatus'] == i) & (df['Product'] == j)]) /␣
↪len(df[df['MaritalStatus'] == i])
        prob = np.round(prob * 100, 2)
        print("Probability of buying '{}' provided the customer is '{}' is {}%␣
↪".format(j, i, prob))
        print()
```

```
Probability of buying 'KP281' provided the customer is 'Single' is 43.84%

Probability of buying 'KP481' provided the customer is 'Single' is 32.88%

Probability of buying 'KP781' provided the customer is 'Single' is 23.29%

----------------------------------------------------------------------------
Probability of buying 'KP281' provided the customer is 'Partnered' is 44.86%

Probability of buying 'KP481' provided the customer is 'Partnered' is 33.64%

Probability of buying 'KP781' provided the customer is 'Partnered' is 21.5%
```

0.4.4 Prob of customer's maritalstatus provuided customer's purchase of product

[93]:
```
products = df['Product'].unique()
statuses = df['MaritalStatus'].unique()
for i in statuses:
    if i != 'Single':
        print('-' * 82)
    for j in products:
        prob = len(df[(df['MaritalStatus'] == i) & (df['Product'] == j)]) /␣
↪len(df[df['Product'] == j])
        prob = np.round(prob * 100, 2)
        print("Probability that the customer is '{}' provided '{}' was bought␣
↪is {}% ".format(i, j, prob))
        print()
```

```
Probability that the customer is 'Single' provided 'KP281' was bought is 40.0%

Probability that the customer is 'Single' provided 'KP481' was bought is 40.0%
```

```
Probability that the customer is 'Single' provided 'KP781' was bought is 42.5%

---------------------------------------------------------------------------------
--
Probability that the customer is 'Partnered' provided 'KP281' was bought is
60.0%

Probability that the customer is 'Partnered' provided 'KP481' was bought is
60.0%

Probability that the customer is 'Partnered' provided 'KP781' was bought is
57.5%
```

[94]:

```
print(pd.crosstab(index = df['Product'], columns = df['Fitness'], margins =␣
 ↪True))
print()
print('-' * 40)
print()
print("Product-wise normalization : ")
print(np.round(pd.crosstab(index = df['Product'], columns = df['Fitness'],␣
 ↪normalize = 'index') * 100, 2))
print()
print('-' * 40)
print()
print("Fitness Scale-wise normalization : ")
print(np.round(pd.crosstab(index = df['Product'], columns = df['Fitness'],␣
 ↪normalize = 'columns') * 100, 2))
```

```
Fitness  1   2   3   4   5  All
Product
KP281    1  14  54   9   2   80
KP481    1  12  39   8   0   60
KP781    0   0   4   7  29   40
All      2  26  97  24  31  180

----------------------------------------

Product-wise normalization :
Fitness     1     2     3      4     5
Product
KP281    1.25  17.5  67.5  11.25   2.5
KP481    1.67  20.0  65.0  13.33   0.0
KP781    0.00   0.0  10.0  17.50  72.5

----------------------------------------

Fitness Scale-wise normalization :
```

```
Fitness     1      2      3      4      5
Product
KP281    50.0  53.85  55.67  37.50   6.45
KP481    50.0  46.15  40.21  33.33   0.00
KP781     0.0   0.00   4.12  29.17  93.55
```

0.4.5 Prob of buying product provided customer's fitness

[95]:
```
products = df['Product'].unique()
scales = sorted(df['Fitness'].unique())
for i in scales:
    if i != 1:
        print('-' * 88)
    for j in products:
        prob = len(df[(df['Fitness'] == i) & (df['Product'] == j)]) /␣
 ↪len(df[df['Fitness'] == i])
        prob = np.round(prob * 100, 2)
        print("Probability of buying '{}' provided the customer has the fitness␣
 ↪scale '{}' is {}% ".format(j, i, prob))
        print()
```

```
Probability of buying 'KP281' provided the customer has the fitness scale '1' is
50.0%

Probability of buying 'KP481' provided the customer has the fitness scale '1' is
50.0%

Probability of buying 'KP781' provided the customer has the fitness scale '1' is
0.0%

--------------------------------------------------------------------------------
--------
Probability of buying 'KP281' provided the customer has the fitness scale '2' is
53.85%

Probability of buying 'KP481' provided the customer has the fitness scale '2' is
46.15%

Probability of buying 'KP781' provided the customer has the fitness scale '2' is
0.0%

--------------------------------------------------------------------------------
--------
Probability of buying 'KP281' provided the customer has the fitness scale '3' is
55.67%

Probability of buying 'KP481' provided the customer has the fitness scale '3' is
40.21%
```

```
Probability of buying 'KP781' provided the customer has the fitness scale '3' is
4.12%

------------------------------------------------------------------------------------------------------------------------
Probability of buying 'KP281' provided the customer has the fitness scale '4' is
37.5%

Probability of buying 'KP481' provided the customer has the fitness scale '4' is
33.33%

Probability of buying 'KP781' provided the customer has the fitness scale '4' is
29.17%

------------------------------------------------------------------------------------------------------------------------
Probability of buying 'KP281' provided the customer has the fitness scale '5' is
6.45%

Probability of buying 'KP481' provided the customer has the fitness scale '5' is
0.0%

Probability of buying 'KP781' provided the customer has the fitness scale '5' is
93.55%
```

0.4.6 Prob of customer's fitness provided product he/she bought

[96]:

```
products = df['Product'].unique()
scales = sorted(df['Fitness'].unique())
for i in scales:
    if i != 1:
        print('-' * 94)
    for j in products:
        prob = len(df[(df['Fitness'] == i) & (df['Product'] == j)]) /␣
 ↪len(df[df['Product'] == j])
        prob = np.round(prob * 100, 2)
        print("Probability that the customer has a fitness scale of '{}'␣
 ↪provided '{}' was bought is {}% ".format(i, j, prob))
        print()
```

```
Probability that the customer has a fitness scale of '1' provided 'KP281' was
bought is 1.25%

Probability that the customer has a fitness scale of '1' provided 'KP481' was
bought is 1.67%
```

```
Probability that the customer has a fitness scale of '1' provided 'KP781' was
bought is 0.0%

--------------------------------------------------------------------------------
--------------
Probability that the customer has a fitness scale of '2' provided 'KP281' was
bought is 17.5%

Probability that the customer has a fitness scale of '2' provided 'KP481' was
bought is 20.0%

Probability that the customer has a fitness scale of '2' provided 'KP781' was
bought is 0.0%

--------------------------------------------------------------------------------
--------------
Probability that the customer has a fitness scale of '3' provided 'KP281' was
bought is 67.5%

Probability that the customer has a fitness scale of '3' provided 'KP481' was
bought is 65.0%

Probability that the customer has a fitness scale of '3' provided 'KP781' was
bought is 10.0%

--------------------------------------------------------------------------------
--------------
Probability that the customer has a fitness scale of '4' provided 'KP281' was
bought is 11.25%

Probability that the customer has a fitness scale of '4' provided 'KP481' was
bought is 13.33%

Probability that the customer has a fitness scale of '4' provided 'KP781' was
bought is 17.5%

--------------------------------------------------------------------------------
--------------
Probability that the customer has a fitness scale of '5' provided 'KP281' was
bought is 2.5%

Probability that the customer has a fitness scale of '5' provided 'KP481' was
bought is 0.0%

Probability that the customer has a fitness scale of '5' provided 'KP781' was
bought is 72.5%
```

0.4.7 Relationship between Fitnees and maritalstatus of customes

[97]:
```
print(pd.crosstab(index = df['MaritalStatus'], columns = df['Fitness'], margins␣
↪= True))
print()
print('-' * 48)
print('Marital Status wise normalization : ')
print()
print(np.round(pd.crosstab(index = df['MaritalStatus'], columns =␣
↪df['Fitness'], normalize = 'index') * 100, 2))
print()
print("-" * 48)
print('Fitness levels wise normalization : ')
print()
print(np.round(pd.crosstab(index = df['MaritalStatus'], columns =␣
↪df['Fitness'], normalize = 'columns') * 100, 2))
```

```
Fitness        1   2   3   4   5  All
MaritalStatus
Partnered      1  18  57  13  18  107
Single         1   8  40  11  13   73
All            2  26  97  24  31  180

------------------------------------------------
Marital Status wise normalization :

Fitness           1      2      3      4      5
MaritalStatus
Partnered      0.93  16.82  53.27  12.15  16.82
Single         1.37  10.96  54.79  15.07  17.81

------------------------------------------------
Fitness levels wise normalization :

Fitness           1      2      3      4      5
MaritalStatus
Partnered      50.0  69.23  58.76  54.17  58.06
Single         50.0  30.77  41.24  45.83  41.94
```

[99]:
```
def income_partitions(x):
    if x < 45000:
        return '< 45k '
    elif 45000 <= x < 60000:
        return '45k - 60k'
    elif 60000 <= x < 80000:
        return '60k - 80k'
    else:
```

```
        return '> 80k'
df['income_bins'] = df['Income'].apply(income_partitions)
df['income_bins'].loc[np.random.randint(0, 180, 10)]
```

[99]:

```
156    60k - 80k
124    45k - 60k
177        > 80k
119    45k - 60k
102        < 45k
88         < 45k
65     60k - 80k
12         < 45k
5          < 45k
158    60k - 80k
Name: income_bins, dtype: object
```

[100]:

```
print(pd.crosstab(index = df['Product'], columns = df['income_bins'], margins =␣
 ↪True))
print()
print('-' * 54)
print('Product wise normalization : ')
print()
print(np.round(pd.crosstab(index = df['Product'], columns = df['income_bins'],␣
 ↪normalize = 'index') * 100, 2))
print()
print("-" * 48)
print('Income-bins wise normalization :')
print()
print(np.round(pd.crosstab(index = df['Product'], columns = df['income_bins'],␣
 ↪normalize = 'columns') * 100, 2))
```

```
income_bins  45k - 60k  60k - 80k  < 45k  > 80k  All
Product
KP281               40          6     34      0   80
KP481               38          7     15      0   60
KP781               11         10      0     19   40
All                 89         23     49     19  180

------------------------------------------------------
Product wise normalization :

income_bins  45k - 60k  60k - 80k  < 45k  > 80k
Product
KP281            50.00       7.50   42.5    0.0
KP481            63.33      11.67   25.0    0.0
KP781            27.50      25.00    0.0   47.5
```

```
---------------------------------------------------
Income-bins wise normalization :

income_bins  45k - 60k  60k - 80k  < 45k   > 80k
Product
KP281            44.94      26.09   69.39    0.0
KP481            42.70      30.43   30.61    0.0
KP781            12.36      43.48    0.00  100.0
```

0.4.8 Prob of buying a product given the income range of customer

[101]:
```
products = df['Product'].unique()
incomes = sorted(df['income_bins'].unique())
for i in incomes:
    if i != '45k - 60k':
        print('-' * 105)
    for j in products:
        prob = len(df[(df['income_bins'] == i) & (df['Product'] == j)]) /␣
 ↪len(df[df['income_bins'] == i])
        prob = np.round(prob * 100, 2)
        print("Probability of buying '{}' provided the customer has the annual␣
 ↪income in range '{}' is {}% ".format(j, i, prob))
        print()
```

```
Probability of buying 'KP281' provided the customer has the annual income in
range '45k - 60k' is 44.94%

Probability of buying 'KP481' provided the customer has the annual income in
range '45k - 60k' is 42.7%

Probability of buying 'KP781' provided the customer has the annual income in
range '45k - 60k' is 12.36%

--------------------------------------------------------------------------------
-------------------------
Probability of buying 'KP281' provided the customer has the annual income in
range '60k - 80k' is 26.09%

Probability of buying 'KP481' provided the customer has the annual income in
range '60k - 80k' is 30.43%

Probability of buying 'KP781' provided the customer has the annual income in
range '60k - 80k' is 43.48%

--------------------------------------------------------------------------------
-------------------------
Probability of buying 'KP281' provided the customer has the annual income in
range '< 45k ' is 69.39%
```

```
Probability of buying 'KP481' provided the customer has the annual income in
range '< 45k ' is 30.61%

Probability of buying 'KP781' provided the customer has the annual income in
range '< 45k ' is 0.0%

--------------------------------------------------------------------------------
-------------------------
Probability of buying 'KP281' provided the customer has the annual income in
range '> 80k' is 0.0%

Probability of buying 'KP481' provided the customer has the annual income in
range '> 80k' is 0.0%

Probability of buying 'KP781' provided the customer has the annual income in
range '> 80k' is 100.0%
```

0.4.9 Prob of income rabge provided product bought by customer

[102]:
```
products = df['Product'].unique()
incomes = sorted(df['income_bins'].unique())
for i in incomes:
    if i != '45k - 60k':
        print('-' * 105)
    for j in products:
        prob = len(df[(df['income_bins'] == i) & (df['Product'] == j)]) /␣
↪len(df[df['Product'] == j])
        prob = np.round(prob * 100, 2)
        print("Probability that the customer's annual income lies in range '{}'␣
↪provided '{}' was bought is {}% ".format(i, j, prob))
        print()
```

```
Probability that the customer's annual income lies in range '45k - 60k' provided
'KP281' was bought is 50.0%

Probability that the customer's annual income lies in range '45k - 60k' provided
'KP481' was bought is 63.33%

Probability that the customer's annual income lies in range '45k - 60k' provided
'KP781' was bought is 27.5%

--------------------------------------------------------------------------------
-------------------------
Probability that the customer's annual income lies in range '60k - 80k' provided
'KP281' was bought is 7.5%
```

```
Probability that the customer's annual income lies in range '60k - 80k' provided
'KP481' was bought is 11.67%

Probability that the customer's annual income lies in range '60k - 80k' provided
'KP781' was bought is 25.0%

--------------------------------------------------------------------------------
-------------------------
Probability that the customer's annual income lies in range '< 45k ' provided
'KP281' was bought is 42.5%

Probability that the customer's annual income lies in range '< 45k ' provided
'KP481' was bought is 25.0%

Probability that the customer's annual income lies in range '< 45k ' provided
'KP781' was bought is 0.0%

--------------------------------------------------------------------------------
-------------------------
Probability that the customer's annual income lies in range '> 80k' provided
'KP281' was bought is 0.0%

Probability that the customer's annual income lies in range '> 80k' provided
'KP481' was bought is 0.0%

Probability that the customer's annual income lies in range '> 80k' provided
'KP781' was bought is 47.5%
```

0.5 Insights

- Male customers are more compare to female. (Ratio is 60:40)
- Around 44% of customers bought KP281, 33% bought KP481 and 22% bought KP781
- Customers with high income or high finess are more likely to buy KP781
- 90% of customers who bought KP781 have fitness 4 or 5
- Prob of income > 80k is 100% given that customer bought KP781
- Five times more male customers bought KP781 compare to female
- Male customers are more likely to buy KP781
- More than 60% of customers are married
- More than 80% of customer's age is between 20 to 30
- 80% customer's income is in the range 40000 to 65000
- Most of (70%) of customers use tredmil 3 to 4 days a week
- More than 50% customers have fitness level 3
- In customers who have 4 or 5 fitness level there are more than 3 times male than female

0.6 Recomendation

- Most of the customers are in the age range of 20 to 30 so marketing strategy should be designed to attract more young people
- We can design seperate marketing strategy to sell KP781 to customers with high income or high fitness level
- We can offer discount to customer profile who are more likely to buy perticular product
- We can launch different fitness chalanges and winners can get discount

[ ]:

```

```