🏃‍♂️🏃‍♀️🏃 AEROFIT - Business CaseStudy 🏃‍♂️🏃‍♀️🏃

Analysed by : **KASI**

🤔 Aerofit :-

`Aerofit` , a dynamic player in the fitness industry, traces its origins to `M/s. Sachdev Sports Co` , established in `1928` by `Ram Ratan Sachdev` . From its modest beginnings in `Hyderabad` , India, the company evolved into a leading sports equipment supplier across Andhra Pradesh and Telangana. Recognizing the growing need for fitness solutions, M/s. Sachdev Overseas emerged to import quality fitness equipment under the "Aerofit" brand, ensuring affordability and post-sales excellence.

Driven by a dedication to innovation, `Nityasach Fitness Pvt Ltd` was founded, spearheaded by director `Nityesh Sachdev` . With the brand "Aerofit" at its core, the company aimed to bridge the gap between international fitness technology and the Indian market. By importing advanced fitness equipment at accessible price points, Aerofit sought to redefine the industry landscape, prioritizing health and vitality while staying true to its legacy of passion and customer focus.

Aerofit provides a product range including machines such as treadmills, exercise bikes, gym equipment, and fitness accessories to cater to the needs of all categories of people.

🚴‍♀️🏋️ Your Fitness Journey Partner 🏋️🚴‍♂️

💪🏽Aerofit stands at the forefront of the fitness industry, delivering excellence in the world of fitness equipment With a comprehensive product range encompassing treadmills, exercise bikes, gym equipment, and a variety of fitness accessories, Aerofit is committed to meeting the diverse needs of individuals across the fitness spectrum. Whether you're a seasoned athlete or a fitness enthusiast, Aerofit is your trusted companion on the journey to a healthier, fitter lifestyle. Elevate your fitness experience with Aerofit – where innovation meets performance. 💪🏽

🏷️ Objective

- The market research team at AeroFit wants to identify the characteristics of the target audience for each type of treadmill offered by the company, to provide a better recommendation of the treadmills to the new customers. The team decides to investigate whether there are differences across the product with respect to customer characteristics.
- Perform descriptive analytics to create a customer profile for each AeroFit treadmill product by developing appropriate tables and charts. For each AeroFit treadmill product, construct two-way contingency tables and compute all conditional and marginal probabilities along with their insights/impact on the business.

👀 About Data

The company collected the data on individuals who purchased a treadmill from the AeroFit stores during `three months` .The data is available in a single csv file

Product Portfolio

- The `KP281` is an `entry-level` treadmill that sells for `USD 1,500` .
- The `KP481` is for `mid-level` runners that sell for `USD 1,750` .
- The `KP781` treadmill is having `advanced features` that sell for `USD 2,500` .

📄 Features of the dataset:

Feature	Description
Product	Product Purchased: KP281, KP481, or KP781
Age	Age of buyer in years
Gender	Gender of buyer (Male/Female)
Education	Education of buyer in years
MaritalStatus	MaritalStatus of buyer (Single or partnered)
Usage	The average number of times the buyer plans to use the treadmill each week
Income	Annual income of the buyer (in $)
Fitness	Self-rated fitness on a 1-to-5 scale, where 1 is the poor shape and 5 is the excellent shape
Miles	The average number of miles the buyer expects to walk/run each week

---

In [1]:
```
# Importing the necessary modules

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
aerofit = pd.read_csv('aerofit_treadmill.txt')
```

In [3]:
```
af = aerofit.copy()
af
```

Out[3]:

	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
...	...	...	...	...	...	...	...	...	...
175	KP781	40	Male	21	Single	6	5	83416	200
176	KP781	42	Male	18	Single	5	4	89641	200
177	KP781	45	Male	16	Single	5	5	90886	160
178	KP781	47	Male	18	Partnered	4	5	104581	120
179	KP781	48	Male	18	Partnered	4	5	95508	180

180 rows × 9 columns

In [4]:
```
af.head()
```

Out[4]:

	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

In [5]:
```
af.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
```

In [6]:
```
af.shape
```

Out[6]:
```
(180, 9)
```

🔁 Changing the Datatype of Columns

In [7]:
```
for col in af.columns:
    if af[col].dtype == 'object':
        af[col] = af[col].astype('category')
```

In [8]:
```
af.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    category
 1   Age            180 non-null    int64
 2   Gender         180 non-null    category
 3   Education      180 non-null    int64
 4   MaritalStatus  180 non-null    category
 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: category(3), int64(6)
memory usage: 9.5 KB
```

📝 Statistical Summary

In [22]:
```
af.describe().T
```

Out[22]:

	count	mean	std	min	25%	50%	75%	max
Age	180.0	28.788889	6.943498	18.0	24.00	26.0	33.00	50.0
Education	180.0	15.572222	1.617055	12.0	14.00	16.0	16.00	21.0
Usage	180.0	3.455556	1.084797	2.0	3.00	3.0	4.00	7.0
Fitness	180.0	3.311111	0.958869	1.0	3.00	3.0	4.00	5.0
Income	180.0	53719.577778	16506.684226	29562.0	44058.75	50596.5	58668.00	104581.0
Miles	180.0	103.194444	51.863605	21.0	66.00	94.0	114.75	360.0

🔍 Insights

1. Age - The age range of customers spans from `18 to 50 year`, with an average age of `29 years`.

2. Education - Customer education levels vary between `12 and 21 years`, with an average education duration of `16 years`.

3. Usage - Customers intend to utilize the product anywhere from `2 to 7 times per week`, with an average usage frequency of `3 times per week`.

4. Fitness - On average, customers have rated their fitness at `3` on a 5-point scale, reflecting a `moderate level of fitness`.

5. Income - The annual income of customers falls within the range of `USD 30,000 to USD 100,000`, with an average income of approximately `USD 54,000`.

6. Miles - Customers' weekly running goals range from `21 to 360 miles`, with an average target of `103 miles per week`.

In [23]:
```
af.describe(include='category').T
```

Out[23]:

	count	unique	top	freq
Product	180	3	KP281	80
Gender	180	2	Male	104
MaritalStatus	180	2	Partnered	107

🔍 Insights

1. Product - Over the past three months, the `KP281` product demonstrated the `highest sales performance` among the three products

2. Gender - Based on the data of last 3 months, around `58%` of the buyers were `Male` and `42%` were `female`

3. Marital Status - Based on the data of last 3 months, around `60%` of the buyers were `Married` and `40%` were `single`

👥 Duplicate Detection

In [15]:
```
af.duplicated().sum()
```

Out[15]:
```
0
```

In [18]:
```
af[af.duplicated()]
```

Out[18]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles

🔍 Insights

- There are no duplicate entries in the dataset

⁉️ Handling Missing values

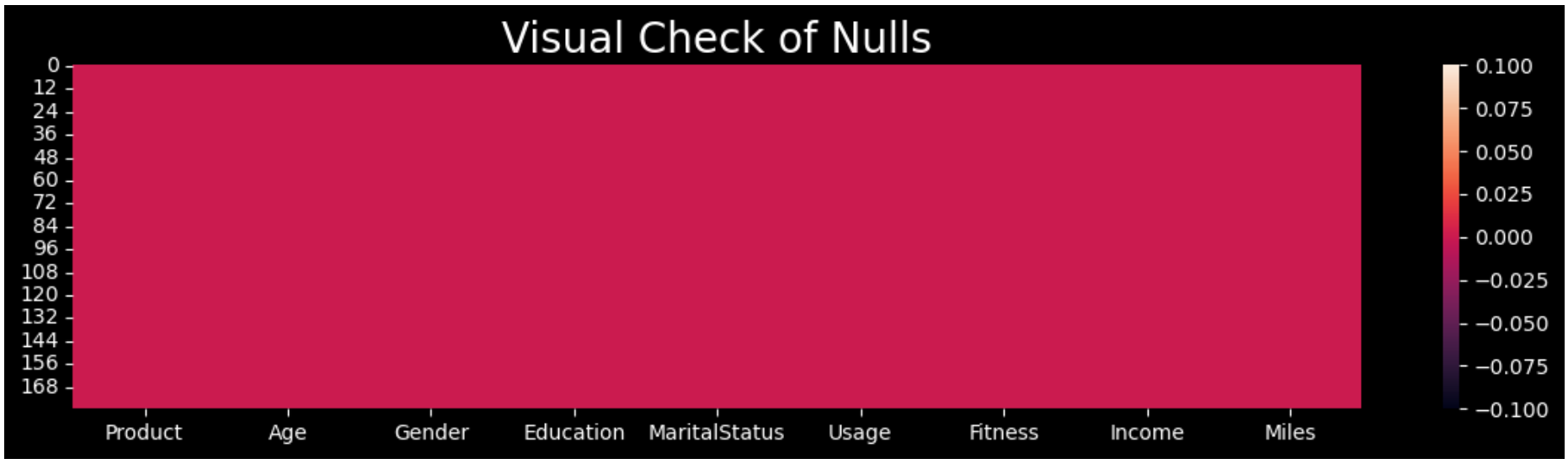
In [24]:
```
af.isna().any()
```

Out[24]:
```
Product          False
Age              False
Gender           False
Education        False
MaritalStatus    False
Usage            False
Fitness          False
Income           False
Miles            False
dtype: bool
```

In [19]:
```
miss = af.isna().sum()
miss
```

Out[19]:
```
Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
Usage            0
Fitness          0
Income           0
Miles            0
dtype: int64
```

In [21]:
```
plt.figure(figsize=(14,3))
plt.style.use('dark_background')
sns.heatmap(af.isnull())
plt.title('Visual Check of Nulls',fontsize=20)
plt.show()
```



🔍 Insights

- There are no missing entries in the dataset

In [25]:
```
print('Parameter Ranges:')
print(f"Age : {af.Age.min()}yrs - {af.Age.max()}yrs")
print(f"Education : {af.Education.min()} - {af.Education.max()} yrs")
print(f"Usage : {af.Usage.min()} - {af.Usage.max()} days per week")
print(f"Fitness : {af.Fitness.min()} - {af.Fitness.max()} in a scale")
print(f"Income : {round(af.Income.min()//1000)}k - {round(af.Income.max()//1000)}k in $")
print(f"Miles : {af.Miles.min()} - {af.Miles.max()} miles per week")
```

```
Parameter Ranges:
Age : 18yrs - 50yrs
Education : 12 - 21 yrs
Usage : 2 - 7 days per week
Fitness : 1 - 5 in a scale
Income : 29k - 104k in $
Miles : 21 - 360 miles per week
```

In [268…
```
#checking the unique values for columns
for col in af.columns:
    print()
    print('Total Unique Values in',col,'column are :-',af[col].nunique())
    print('Unique Values in',col,'column are :-\n',af[col].unique())
    print()
    print('-'*100)
```

```
Total Unique Values in Product column are :- 3
Unique Values in Product column are :-
 ['KP281', 'KP481', 'KP781']
Categories (3, object): ['KP281', 'KP481', 'KP781']

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

Total Unique Values in Age column are :- 32
Unique Values in Age column are :-
 [18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
 43 44 46 47 50 45 48 42]

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

Total Unique Values in Gender column are :- 2
Unique Values in Gender column are :-
 ['Male', 'Female']
Categories (2, object): ['Female', 'Male']

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

Total Unique Values in Education column are :- 8
Unique Values in Education column are :-
 [14 15 12 13 16 18 20 21]

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

Total Unique Values in MaritalStatus column are :- 2
Unique Values in MaritalStatus column are :-
 ['Single', 'Partnered']
Categories (2, object): ['Partnered', 'Single']

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

Total Unique Values in Usage column are :- 6
Unique Values in Usage column are :-
 [3 2 4 5 6 7]

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

Total Unique Values in Fitness column are :- 5
Unique Values in Fitness column are :-
 [4 3 2 1 5]

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

Total Unique Values in Income column are :- 62
Unique Values in Income column are :-
 [ 29562  31836  30699  32973  35247  37521  36384  38658  40932  34110
  39795  42069  44343  45480  46617  48891  53439  43206  52302  51165
  50028  54576  68220  55713  60261  67083  56850  59124  61398  57987
  64809  47754  65220  62535  48658  54781  48556  58516  53536  61006
  57271  52291  49801  62251  64741  70966  75946  74701  69721  83416
  88396  90886  92131  77191  52290  85906 103336  99601  89641  95866
 104581  95508]

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

Total Unique Values in Miles column are :- 37
Unique Values in Miles column are :-
 [112  75  66  85  47 141 103  94 113  38 188  56 132 169  64  53 106  95
 212  42 127  74 170  21 120 200 140 100  80 160 180 240 150 300 280 260
 360]

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

Total Unique Values in Fitness_comment column are :- 5
Unique Values in Fitness_comment column are :-
 ['Good Shape' 'Average Shape' 'Bad Shape' 'Poor Shape' 'Excellent Shape']

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

Total Unique Values in age_category column are :- 4
Unique Values in age_category column are :-
 ['Teenage', 'Adults', 'Middle Aged', 'Elderly']
Categories (4, object): ['Teenage' < 'Adults' < 'Middle Aged' < 'Elderly']

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

Total Unique Values in income_grp column are :- 3
Unique Values in income_grp column are :-
 ['middle_class', 'upper_middle_class', 'high_class']
Categories (3, object): ['middle_class' < 'upper_middle_class' < 'high_class']

----------------------------------------------------------------------------------------------------
```

In [46]:

```
for col in af.columns:
    if af[col].dtype != 'category':
        print(f'Value_counts of {col} are :- \n{af[col].value_counts().to_frame().reset_index()}')
        print()
        print('-'*100)
```

```
Value_counts of Age are :-
    Age  count
0    25     25
1    23     18
2    24     12
3    26     12
4    28      9
5    35      8
6    33      8
7    30      7
8    38      7
9    21      7
10   22      7
11   27      7
12   31      6
13   34      6
14   29      6
15   20      5
16   40      5
17   32      4
18   19      4
19   48      2
20   37      2
21   45      2
22   47      2
23   46      1
24   50      1
25   18      1
26   44      1
27   43      1
28   41      1
29   39      1
30   36      1
31   42      1

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

Value_counts of Education are :-
   Education  count
0         16     85
1         14     55
2         18     23
3         15      5
4         13      5
5         12      3
6         21      3
7         20      1

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

Value_counts of Usage are :-
   Usage  count
0      3     69
1      4     52
2      2     33
3      5     17
4      6      7
5      7      2

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

Value_counts of Fitness are :-
   Fitness  count
0        3     97
1        5     31
2        2     26
3        4     24
4        1      2

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

Value_counts of Income are :-
    Income  count
0    45480     14
1    52302      9
2    46617      8
3    54576      8
4    53439      8
..     ...    ...
57   65220      1
58   55713      1
59   68220      1
60   30699      1
61   95508      1

[62 rows x 2 columns]

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

Value_counts of Miles are :-
    Miles  count
0      85     27
1      95     12
```

```
2      66     10
3      75     10
4      47      9
5     106      9
6      94      8
7     113      8
8      53      7
9     100      7
10    180      6
11    200      6
12     56      6
13     64      6
14    127      5
15    160      5
16     42      4
17    150      4
18     38      3
19     74      3
20    170      3
21    120      3
22    103      3
23    132      2
24    141      2
25    280      1
26    260      1
27    300      1
28    240      1
29    112      1
30    212      1
31     80      1
32    140      1
33     21      1
34    169      1
35    188      1
36    360      1
```

👉Insights summary:

- There a 180 rows and 9 columns with most numerical data.
- There are no missing values in the data.
- There are 3 unique products ('KP281', 'KP481', 'KP781').
- KP281 is the most frequent product.
- Males are generally the frequent customers.
- The standard deviation for **Income & Miles** is very high so there might be a possibility of outliers in those attibutes.
- The Average age of the customers is 28 and they have approx 16 yrs of education.
- The Mean Usage per week is 3.4, with maximum as 7 and minimum as 2.
- The Average fitness rating is 3.3 on a scale of 1 to 5.
- The Average number of miles the customer walks is 103 with max is almost 115 and minimum of 21.
- There are 107 partnered customers and 73 single customers.
- On an Average , customers use threadmill ***THRICE*** a week.

🏃 EXPLORATORY DATA ANALYSIS 🏃

In [50]:
```
#color_palette
cp = ['#A10054','#FC993C','#FFE775','#BD4682','#8C2057']
cp1 = ['#D25380','#E7CBCB','#EE8972','#790252']
cp2 = ['#F7D695','#EE8972','#D15A7C']
cp3 = ['#4F8A8B','#FBD46D','#E7CBCB']
cp4 = ['#A10054','#FC993C','#EE8972','#D15A7C','#FFE775','#BD4682','#8C2057']
```

In [47]:
```
af.sample()
```

Out[47]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
13	KP281	22	Female	14	Single	3	3	35247	75

In [49]:
```
af.Product.value_counts()
```

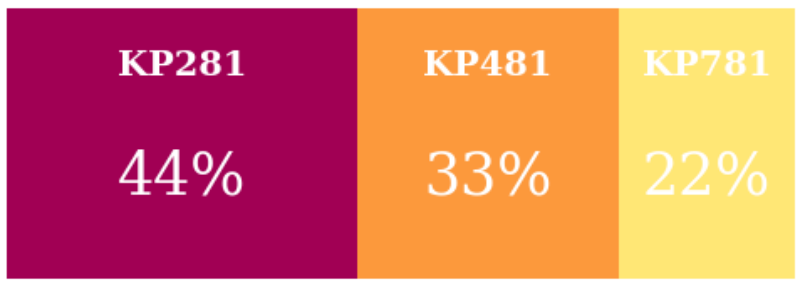
Out[49]:
```
Product
KP281    80
KP481    60
KP781    40
Name: count, dtype: int64
```

In [51]:
```
# normalize gives the frequency of occurrence of a particular in whole data.
# Customer Product statistics (listed in %)
fp = af.Product.value_counts(normalize=True).to_frame()
fp = fp.reset_index()
fp['proportion'] = round(fp.proportion*100,2)
fp
```

Out[51]:

	Product	proportion
0	KP281	44.44
1	KP481	33.33
2	KP781	22.22

In [83]:
```
plt.figure(figsize=(14,2.3))
plt.subplot(1,2,1)
product_count = af['Product'].value_counts()
product_count['percent'] = ((product_count.values/af.shape[0])* 100).round()

plt.style.use('default')
plt.style.use('seaborn-v0_8-bright')

plt.barh(product_count.index[0],product_count.loc['percent'][0],color = cp[0])
plt.barh(product_count.index[0],product_count.loc['percent'][1],left = product_count.loc['percent'][0],color = cp[1])
plt.barh(product_count.index[0],product_count.loc['percent'][2],
         left = product_count.loc['percent'][0] + product_count.loc['percent'][1], color = cp[2])
plt.xlim=(0,100)
plt.axis('off')

product_count['info_percent'] =[product_count['percent'][0]/2,product_count['percent'][0] + product_count['percent'][1]/2,
                                product_count['percent'][0] + product_count['percent'][1] + product_count['percent'][2]/2]
for i in range(3):
    plt.text(product_count['info_percent'][i],0.23,product_count.index[i],
             va = 'center', ha='center',fontsize=15, fontweight='bold', fontfamily='serif',color='white')
    plt.text(product_count['info_percent'][i],-0.1,f"{product_count['percent'][i]:.0f}%",
             va = 'center', ha='center',fontsize=25, fontweight='light', fontfamily='serif',color='white')


plt.subplot(1,2,2)
product_portfolio = [['KP281','$1500','$120k'],['KP481','$1750','$105k'],['KP781','$2500','$100k']]
color_2d = [[cp[0],'#FFFFFF','#FFFFFF'],[cp[1],'#FFFFFF','#FFFFFF'],[cp[2],'#FFFFFF','#FFFFFF']]

table = plt.table(cellText = product_portfolio, cellColours=color_2d, cellLoc='center',colLabels =['Product','Price','Sales'],
                  colLoc = 'center',bbox =[0, 0, 1, 1])
plt.axis('off')

plt.show()
```



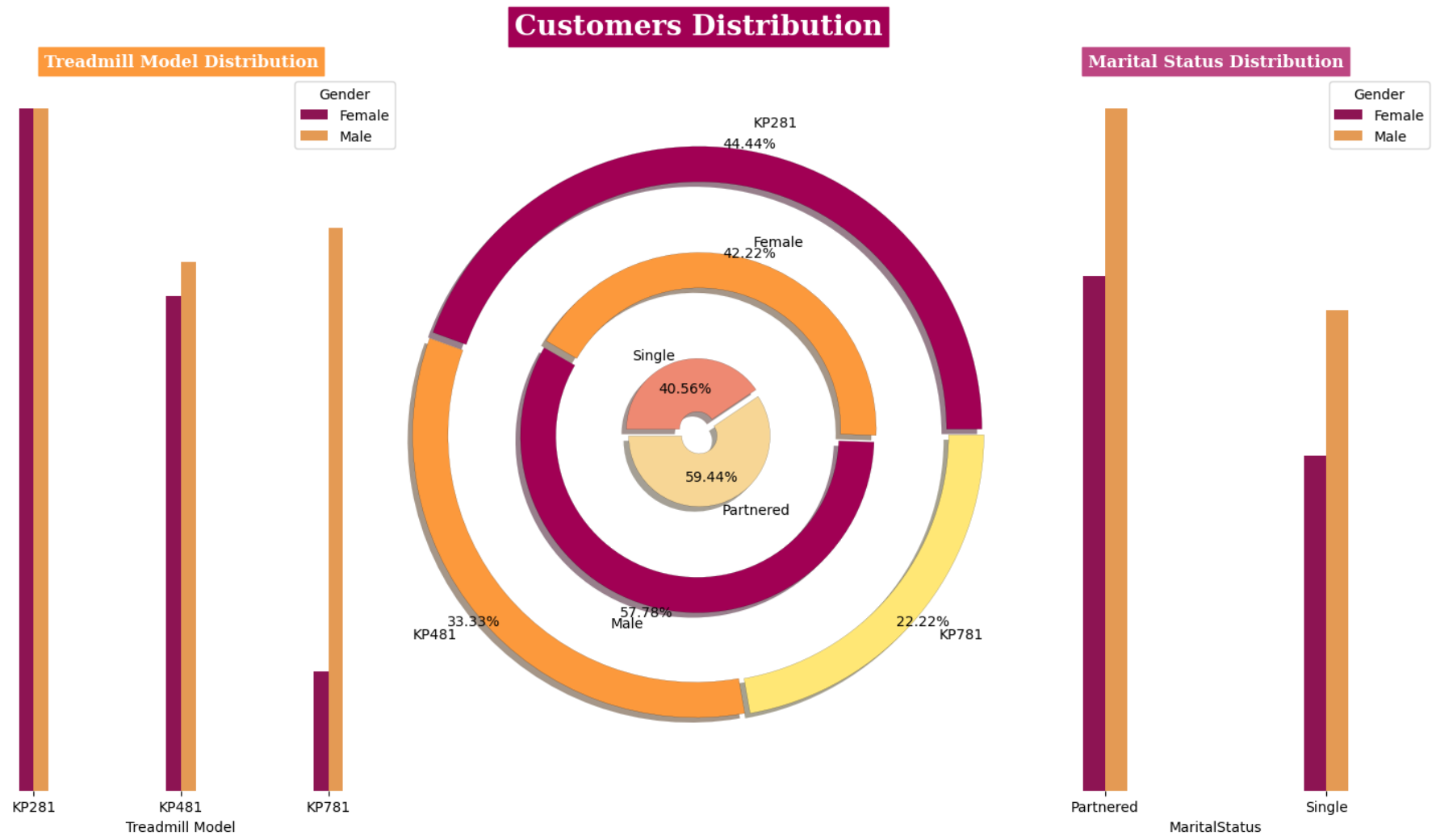
Product	Price	Sales
KP281	$1500	$120k
KP481	$1750	$105k
KP781	$2500	$100k

🔍 Insights:

- **44.44%** of customers bought **KP281** product type
- **33.33%** of customers bought **KP481** product type
- **22.22%** of customers bought **KP781** product type

In [53]:
```
plt.figure(figsize=(15,8.5))
plt.suptitle('Customers Distribution',fontfamily='serif',fontweight='bold',fontsize=20,
             backgroundcolor=cp[0],color='w')
plt.style.use('default')
plt.style.use('seaborn-v0_8-bright')

plt.subplot(1,3,1)
plt.title('Treadmill Model Distribution',fontfamily='serif',fontweight='bold',fontsize=12,
          backgroundcolor=cp[1],color='w')
a = sns.countplot(data=af,x='Product',hue='Gender',palette=cp,width=0.2)
plt.xlabel('Treadmill Model')
plt.ylabel('')
plt.yticks([])

plt.subplot(1,3,3)
a = sns.countplot(data=af,x='MaritalStatus',hue='Gender',palette=cp,width=0.2)
plt.title('Marital Status Distribution',fontfamily='serif',fontweight='bold',fontsize=12,
          backgroundcolor=cp[3],color='w')
plt.yticks([])
plt.ylabel('')
sns.despine(left=True,bottom=True,trim=True)

plt.subplot(1,3,2)
plt.pie(af.Product.value_counts(), labels=af.Product.value_counts().index,
       counterclock=True , explode=(0.02,0.02,0.02) , autopct='%.2f%%', pctdistance=1.025,
       colors=cp , textprops={'color':'k','fontsize':10} , shadow=True, radius=1.6,
       wedgeprops=dict(edgecolor='k',linewidth=0.1,width=0.2))

plt.pie(af.Gender.value_counts(), labels=af.Gender.value_counts().index,
       startangle=150 , explode=(0.02,0.02) , autopct='%.2f%%',pctdistance=1.035,
       colors=cp , textprops={'color':'k','fontsize':10} , shadow=True,radius=1,
       wedgeprops=dict(edgecolor='k',linewidth=0.1,antialiased=True,width=0.2))
```

```
plt.pie(af.MaritalStatus.value_counts(), labels=af.MaritalStatus.value_counts().index,
       startangle=180 , explode=(0.02,0.02) , autopct='%.2f%%',
       colors=cp2 , textprops={'color':'k','fontsize':10} , shadow=True, radius=0.4,
       wedgeprops=dict(edgecolor='k',linewidth=0.1,antialiased=True,width=0.3))
plt.tight_layout()
plt.show()
```

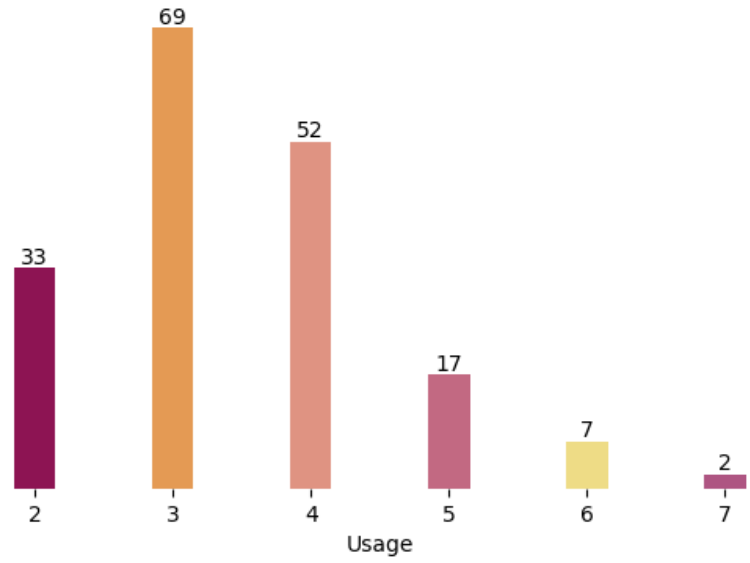


In [120…

```
plt.figure(figsize=(15,4))
plt.suptitle('Customers usage',fontfamily='serif',fontweight='bold',fontsize=20,
             backgroundcolor=cp2[1],color='w')
plt.style.use('default')
plt.style.use('seaborn-v0_8-bright')

plt.subplot(1,2,1)
usage_info = [['3','38%'],['4','29%'],['2','19%'],['5','9%'],['6','4%'],['7','1%']]
color_2d = [[cp4[0],'#FFFFFF'],[cp4[1],'#FFFFFF'],[cp4[2],'#FFFFFF'],[cp4[3],'#FFFFFF'],[cp4[5],'#FFFFFF'],
           [cp1[0],'#FFFFFF']]

plt.table(cellText = usage_info, cellColours=color_2d, cellLoc='center',colLabels =['Usage Per Week','Percent'],
                  colLoc = 'center',bbox =[0, 0, 1, 1])
plt.axis('off')

plt.subplot(1,2,2)
u = af['Usage'].value_counts()
a = sns.barplot(x=u.index,y = u.values,palette=cp4,width=0.3)
a.bar_label(a.containers[0],label_type='edge')
sns.despine(left=True,bottom=True)
#plt.xticks([])
plt.yticks([])
plt.ylabel('')

plt.figure(figsize=(15,4))
plt.suptitle('Customers Fitness',fontfamily='serif',fontweight='bold',fontsize=20,
             backgroundcolor=cp2[2],color='w')
plt.style.use('default')
plt.style.use('seaborn-v0_8-bright')

plt.subplot(1,2,1)
fitness_info = [['3','54%'],['5','17%'],['2','15%'],['4','13%'],['1','1%']]
color_2d = [[cp[0],'#FFFFFF'],[cp[1],'#FFFFFF'],[cp[2],'#FFFFFF'],[cp[3],'#FFFFFF'],[cp[4],'#FFFFFF']]

plt.table(cellText = fitness_info, cellColours=color_2d, cellLoc='center',colLabels =['Fitness','Percent'],
                  colLoc = 'center',bbox =[0, 0, 1, 1])
plt.axis('off')

plt.subplot(1,2,2)
f = af['Fitness'].value_counts()
b = sns.barplot(x=f.index,y = f.values,palette=cp4,width=0.3)
b.bar_label(b.containers[0],label_type='edge')
plt.title('Customer count based on Fitness')
sns.despine(left=True,bottom=True)
plt.yticks([])
#plt.xticks([])
plt.ylabel('')
plt.show()
```

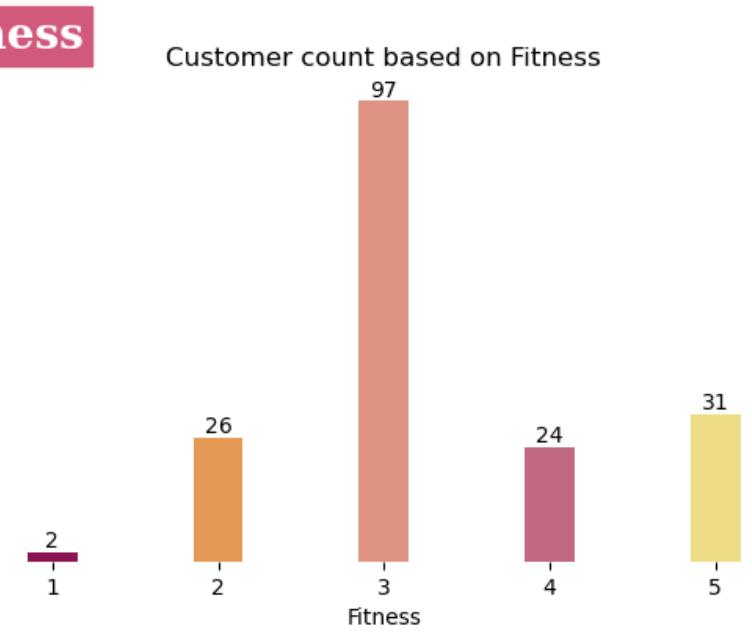
Customers usage

Usage Per Week	Percent
3	38%
4	29%
2	19%
5	9%
6	4%
7	1%



Customers Fitness

Fitness	Percent
3	54%
5	17%
2	15%
4	13%
1	1%



🔍 Insights

- Almost `85%` of the customers plan to use the treadmill for `2 to 4 times a week` and only 15% using 5 times and above each week
- 54% of the customers have self-evaluated their fitness at a level 3 on a scale of 1 to 5. Furthermore, a substantial 84% of the total customers have rated themselves at 3 or higher, indicating commendable fitness levels.

In [89]:
```
# Customer Gender statistics (listed in %)
fg = af['Gender'].value_counts(normalize=True)
fg = fg.map(lambda z: round(z*100,2))
fg
```

Out[89]:
```
Gender
Male      57.78
Female    42.22
Name: proportion, dtype: float64
```

🔍 Insights:

- **57.78%** of customers are `Male`
- **42.22%** customers are `Female`

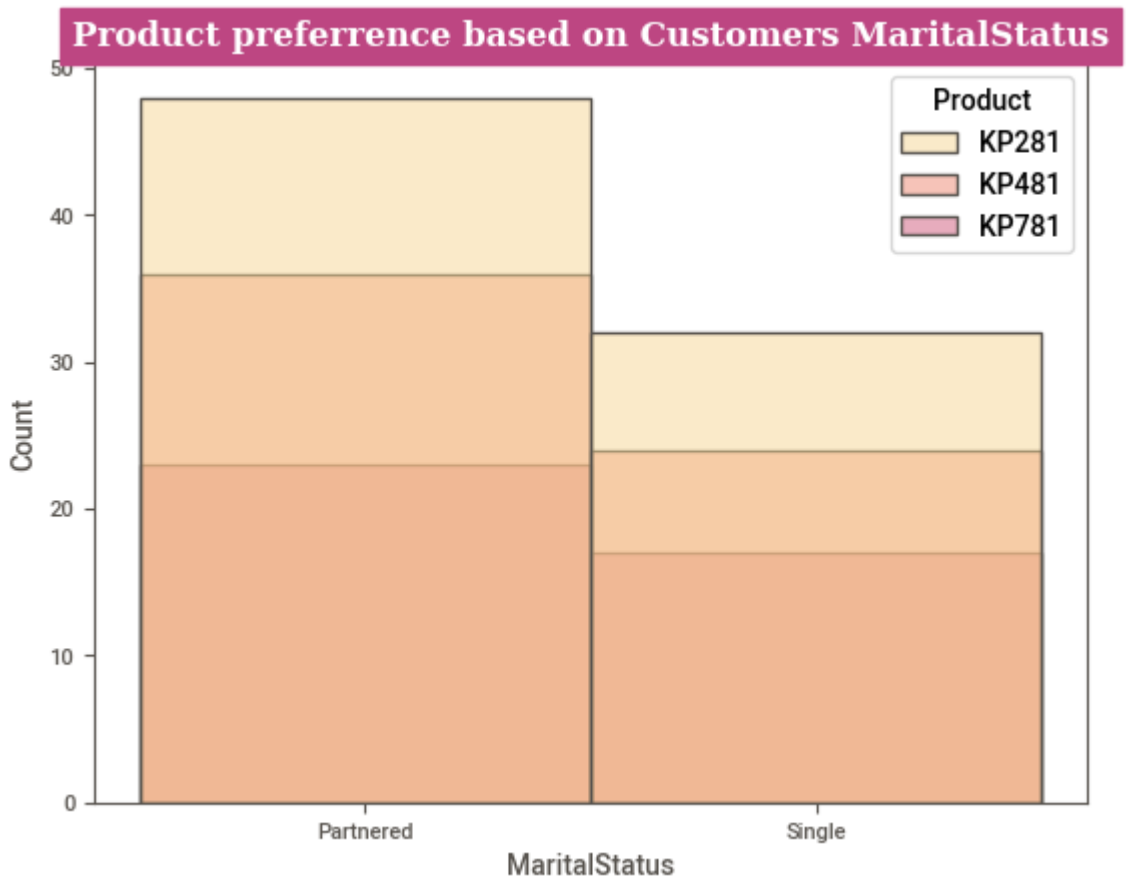
In [91]:
```
# Customer Marital Status statistics (listed in %)
mf= af['MaritalStatus'].value_counts(normalize=True)
mf = mf.map(lambda z:round(z*100,2))
mf
```

Out[91]:
```
MaritalStatus
Partnered    59.44
Single       40.56
Name: proportion, dtype: float64
```

🔍 Insights:

- **59.44%** of customers are `Married/Partnered`
- **40.56%** of customers are `Single`

In [312…
```
sns.histplot(data=af, x="MaritalStatus", hue="Product" , palette=cp2)
plt.title('Product preferrence based on Customers MaritalStatus',fontfamily='serif',fontweight='bold',fontsize=12,
             backgroundcolor=cp[3],color='w')
plt.show()
```



In [92]:
```
# Customers Usage - Number of days used per week (listed in %)
cu = af['Usage'].value_counts(normalize=True).map(lambda z:round(z*100,2)).reset_index()
cu = cu.rename({'Usage':'DaysPerWeek'},axis=1)
cu
```

Out[92]:

	DaysPerWeek	proportion
0	3	38.33
1	4	28.89
2	2	18.33
3	5	9.44
4	6	3.89
5	7	1.11

🔍 Insights:

- **Around 38%** of customers use `3 days` **per week**
- **Less than 4%** of customers use `6 days` **per week**

In [94]:
```
# Customers Fitness - Number of days used per week (listed in %)
ffc = af['Fitness_comment'].value_counts(normalize=True)
ffc = ffc.map(lambda z:round(z*100,2)).reset_index()
manual_order=['Poor Shape','Bad Shape','Average Shape','Good Shape','Excellent Shape']
ffc['Fitness_comment'] = pd.Categorical(ffc['Fitness_comment'],categories=manual_order, ordered=True)
ffc = ffc.sort_values(by='Fitness_comment').reset_index(drop=True)
ffc
```

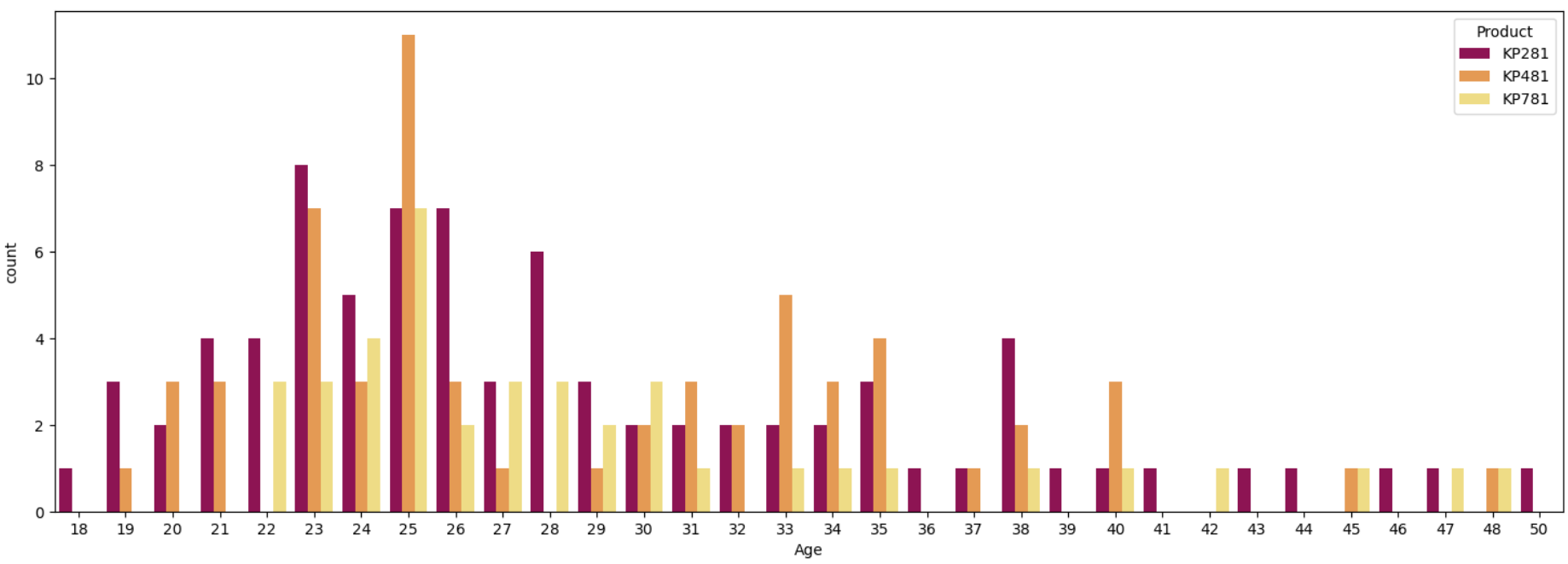
Out[94]:

	Fitness_comment	proportion
0	Poor Shape	1.11
1	Bad Shape	14.44
2	Average Shape	53.89
3	Good Shape	13.33
4	Excellent Shape	17.22

🔍 Insights:

- **Approx 54%** of customers have rated themselves as they are in `Average Shape`
- **Little close to 14%** of customers have rated their fitness less than average
- **Over 17%** of customers have `Peak fitness ratings`

In [279…
```
plt.figure(figsize = (18,6))
plt.style.use('default')
plt.style.use('seaborn-v0_8-bright')
sns.countplot(data=af, x='Age',hue="Product",palette=cp)
plt.suptitle('Products Preferences based on Customers Age',fontfamily='serif',fontweight='bold',fontsize=20,
             backgroundcolor=cp4[5],color='w')

plt.show()
```



✅ Categorizing numerical columns:

In [86]:
```
# Categorization of age:

# 0-21 -> Teenage
# 22-35 -> Adults
# 36-45 -> Middle Age
# 46-60 -> Elderly person

af['age_category'] = af.Age
af['age_category'] = pd.cut(af.age_category,bins=[0,20,35,45,60],labels=['Teenage','Adults','Middle Aged','Elderly'])
af.sample(2)
```

Out[86]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles	Fitness_comment	age_category
89	KP481	23	Female	16	Single	3	3	45480	95	Average Shape	Adults
122	KP481	33	Male	16	Partnered	3	3	51165	95	Average Shape	Adults

In [121…
```
af.age_category.value_counts()
```

Out[121]:
```
age_category
Adults         142
Middle Aged     22
Teenage         10
Elderly          6
Name: count, dtype: int64
```

In [122…
```
acc = af.groupby(['Product','age_category']).agg(cnt=('Product','count'))
acc.reset_index(inplace=True)
acc
```

Out[122]:

	Product	age_category	cnt
0	KP281	Teenage	6
1	KP281	Adults	60
2	KP281	Middle Aged	11
3	KP281	Elderly	3
4	KP481	Teenage	4
5	KP481	Adults	48
6	KP481	Middle Aged	7
7	KP481	Elderly	1
8	KP781	Teenage	0
9	KP781	Adults	34
10	KP781	Middle Aged	4
11	KP781	Elderly	2

In [87]:
```
# Categorization of Fitness Rating where 1 is the poor shape and 5 is the excellent shape.

af['Fitness_comment'] = af.Fitness
af['Fitness_comment'] = af.Fitness_comment.replace({
                            1:"Poor Shape",
                            2:"Bad Shape",
                            3:"Average Shape",
                            4:"Good Shape",
                            5:"Excellent Shape"})
af.tail()
```

Out[87]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles	Fitness_comment	age_category
175	KP781	40	Male	21	Single	6	5	83416	200	Excellent Shape	Middle Aged
176	KP781	42	Male	18	Single	5	4	89641	200	Good Shape	Middle Aged
177	KP781	45	Male	16	Single	5	5	90886	160	Excellent Shape	Middle Aged
178	KP781	47	Male	18	Partnered	4	5	104581	120	Excellent Shape	Elderly
179	KP781	48	Male	18	Partnered	4	5	95508	180	Excellent Shape	Elderly

In [125…
```
# Categorization of income:

af.Income.describe()
```

Out[125]:
```
count       180.000000
mean      53719.577778
std       16506.684226
min       29562.000000
25%       44058.750000
50%       50596.500000
75%       58668.000000
max      104581.000000
Name: Income, dtype: float64
```

In [126…
```
af['income_grp'] = pd.cut(af.Income , bins=[25000,50000,75000,150000] , labels=['middle_class','upper_middle_class','high_class']
```

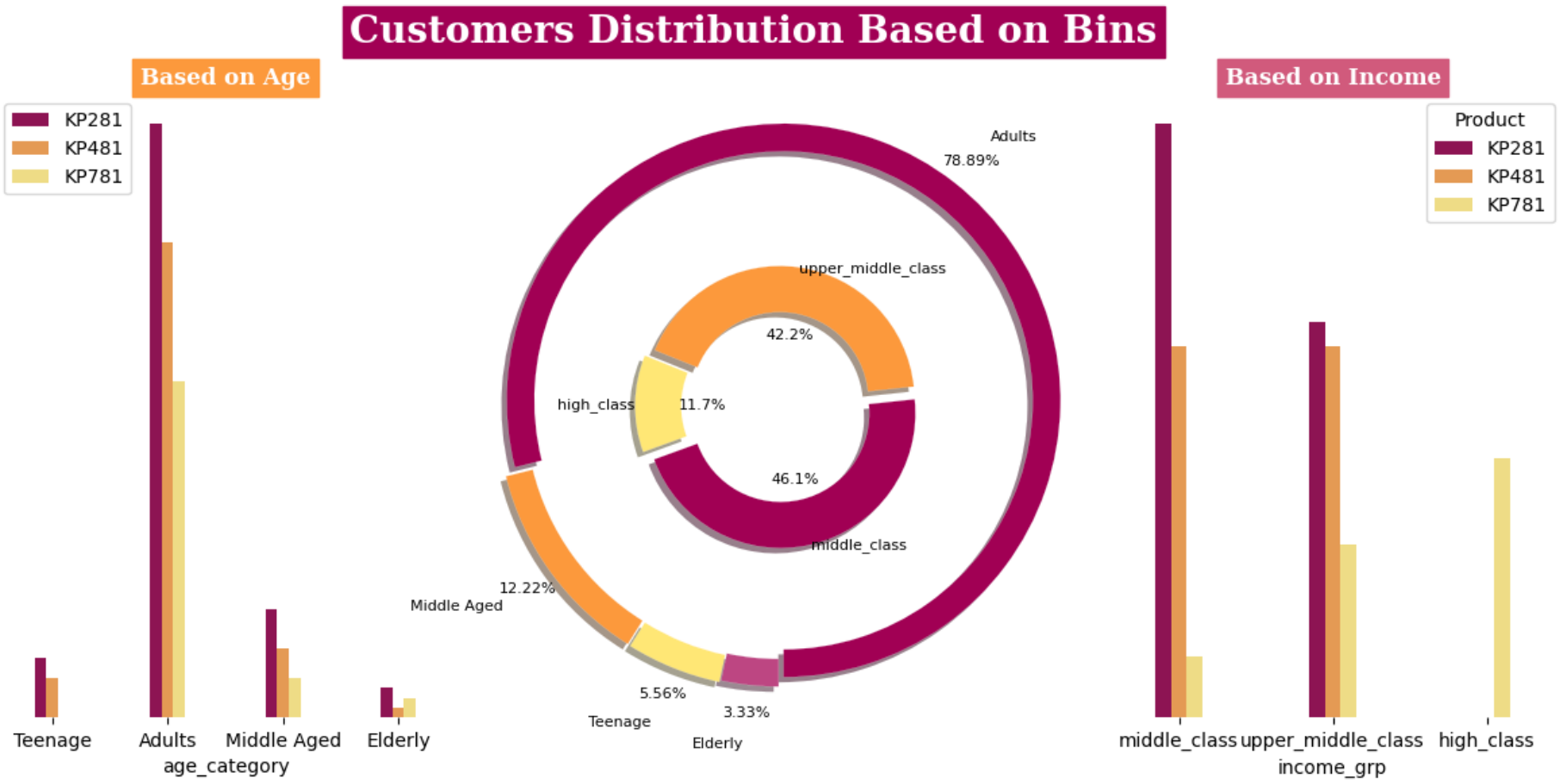
In [134…
```
af.sample(4)
```

Out[134]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles	Fitness_comment	age_category	income_grp
179	KP781	48	Male	18	Partnered	4	5	95508	180	Excellent Shape	Elderly	high_class
112	KP481	29	Female	14	Partnered	3	3	51165	95	Average Shape	Adults	upper_middle_class
114	KP481	30	Female	13	Single	4	3	46617	106	Average Shape	Adults	middle_class
43	KP281	27	Female	14	Partnered	2	3	45480	56	Average Shape	Adults	middle_class

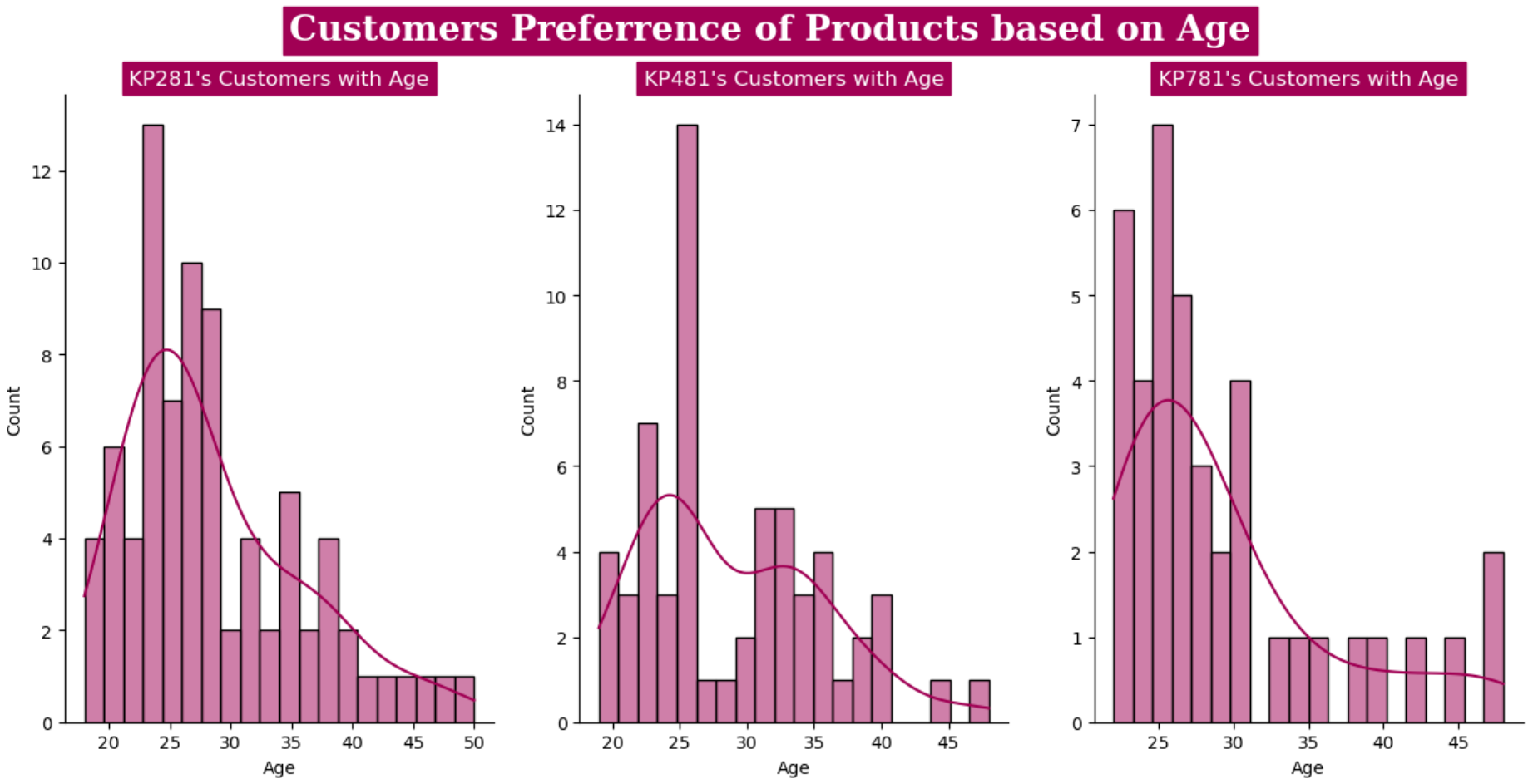
In [ ]:
```
# saving the new file of raw cleaned data:
af.to_csv('aerofit_final.csv',sep=',',index=False)
```

In [138…
```
plt.figure(figsize=(15,6))
plt.suptitle('Customers Distribution Based on Bins',fontfamily='serif',fontweight='bold',fontsize=20,
             backgroundcolor=cp4[0],color='w')
plt.style.use('default')
plt.style.use('seaborn-v0_8-bright')

plt.subplot(1,3,1)
sns.countplot(af,x='age_category',hue='Product',palette=cp,width=0.3)
plt.title('Based on Age',fontfamily='serif',fontweight='bold',fontsize=12,backgroundcolor=cp4[1],color='w')
plt.yticks([])
plt.legend(loc='upper left')
plt.ylabel('')

plt.subplot(1,3,2)
plt.pie(af.age_category.value_counts(), labels=af.age_category.value_counts().index,
        explode=(0.04,0.03,0.03,0.02) , autopct='%.2f%%', pctdistance=1.1,startangle=270,
        colors=cp , textprops={'color':'k','fontsize':8} , shadow=True, labeldistance=1.21,
        wedgeprops=dict(edgecolor='w',linewidth=0.1,width=0.15), radius=1.5)

plt.pie(af.income_grp.value_counts(), labels=af.income_grp.value_counts().index, counterclock=True,
        startangle=200 , explode=(0.04,0.03,0.05) , autopct='%.1f%%', pctdistance=0.5,
        colors=cp , textprops={'color':'k','fontsize':8} , shadow=True, labeldistance=1,
        wedgeprops=dict(edgecolor='w',linewidth=0.1,width=0.25), radius=0.73)

plt.subplot(1,3,3)
sns.countplot(af,x='income_grp',hue='Product',palette=cp,width=0.3)
plt.title('Based on Income',fontfamily='serif',fontweight='bold',fontsize=12,backgroundcolor=cp4[3],color='w')
sns.despine(left=True,bottom=True,trim=True)
plt.yticks([])
plt.ylabel('')

plt.show()
```

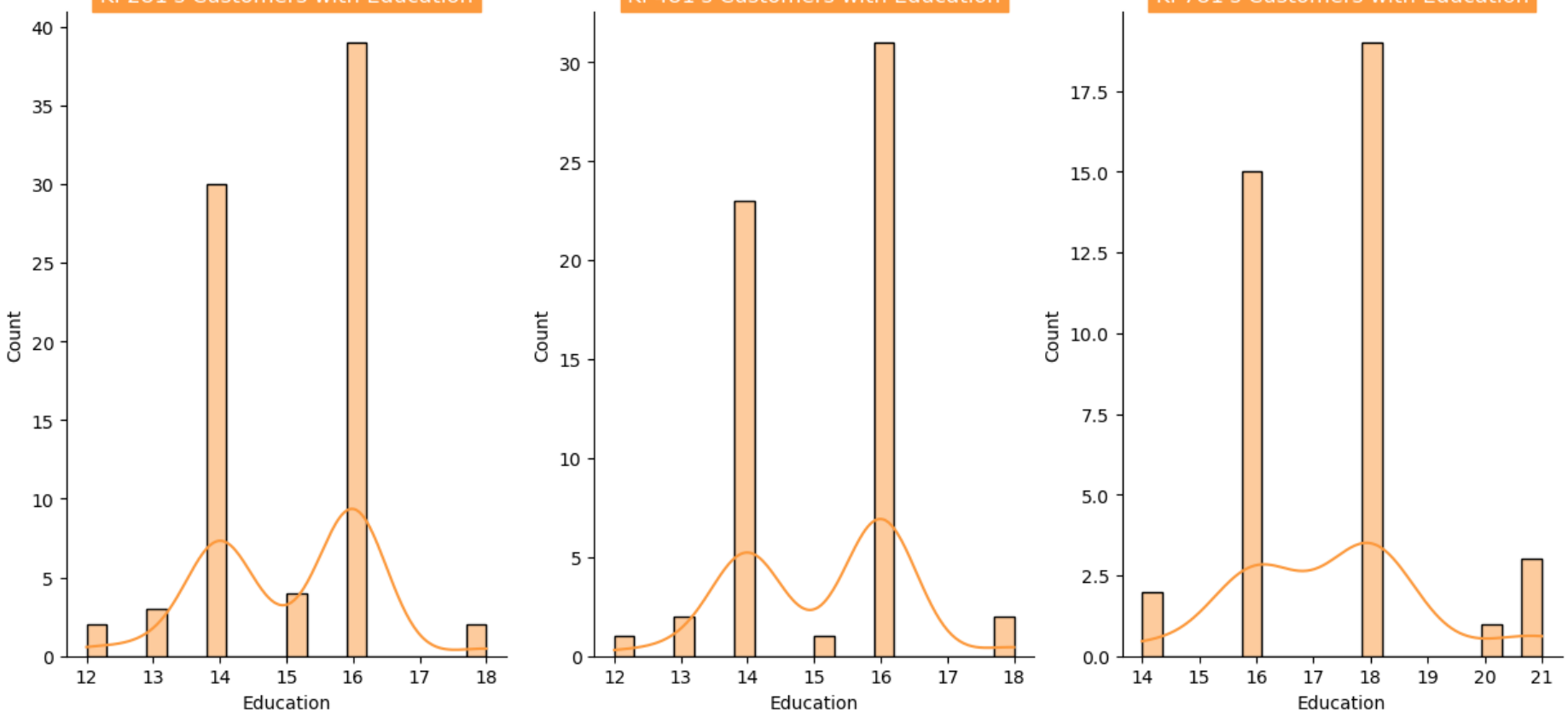


In [123…

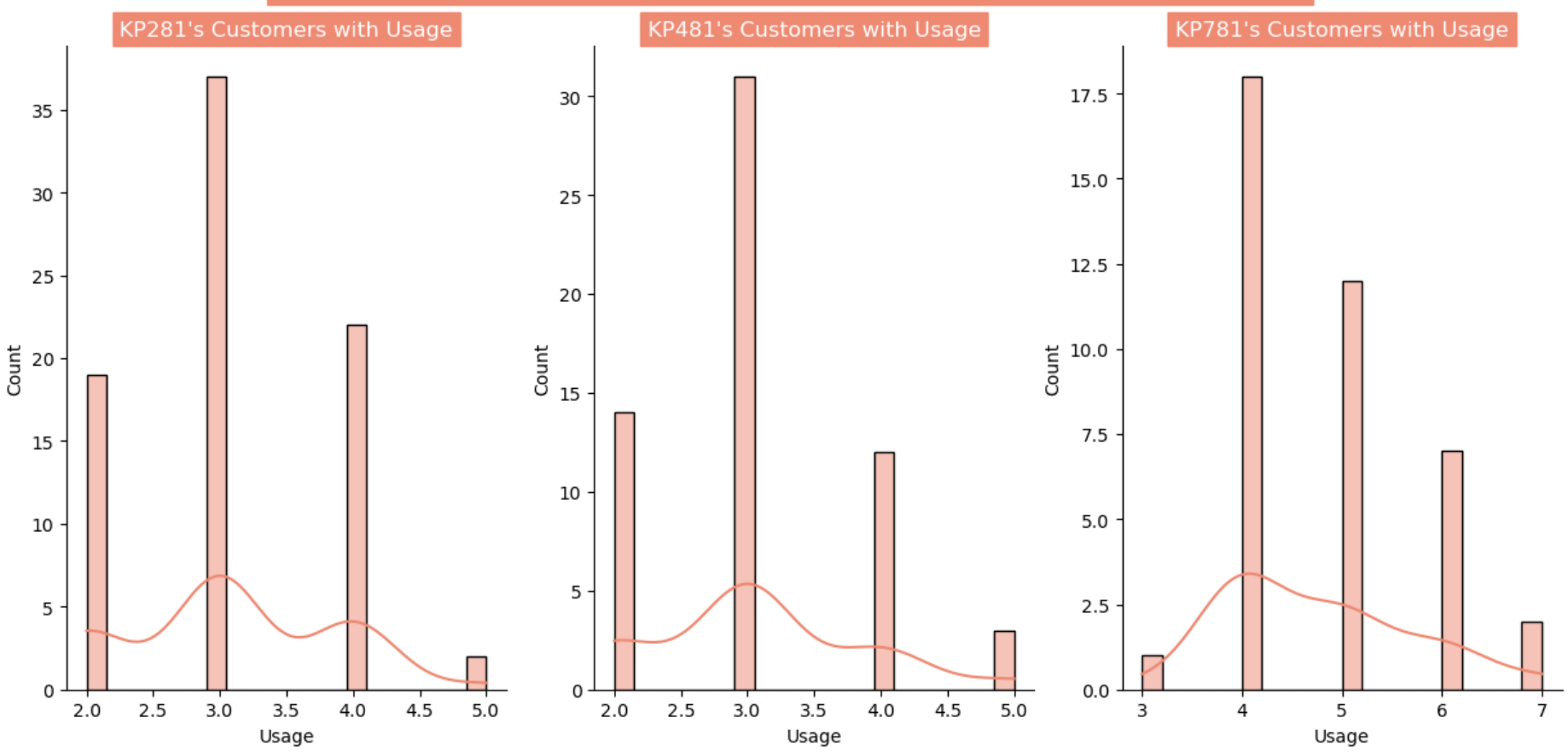
```
p = af.Product.unique()
num_cols = ['Age','Education','Usage','Fitness','Income','Miles']

for i in range(len(num_cols)):
    fig,ax = plt.subplots(nrows=1, ncols=3, figsize=(15,6.5))
    plt.suptitle(f"Customers Preferrence of Products based on {num_cols[i]}",fontsize=20,fontfamily='serif',fontweight='bold'
                 ,backgroundcolor=cp4[i],color='w')
    for j in range(len(p)):
        prd = af[af.Product==p[j]]
        sns.histplot(data=prd, x=num_cols[i], bins=20, kde=True, ax=ax[j], color=cp4[i])
        sns.despine()
        ax[j].set_title(f"{p[j]}'s Customers with {num_cols[i]}",backgroundcolor=cp4[i],color='w')
```



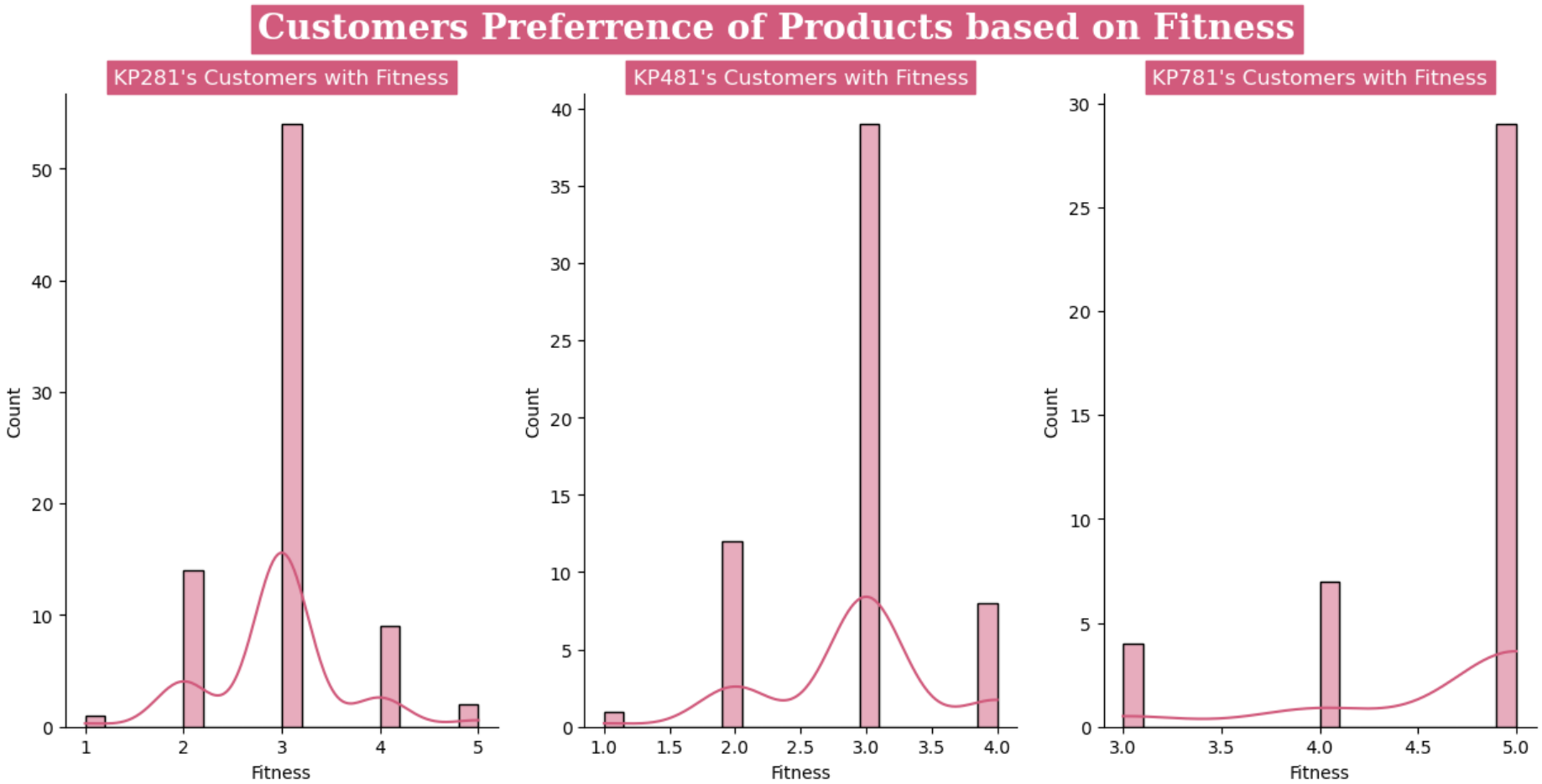
Customers Preferrence of Products based on Education

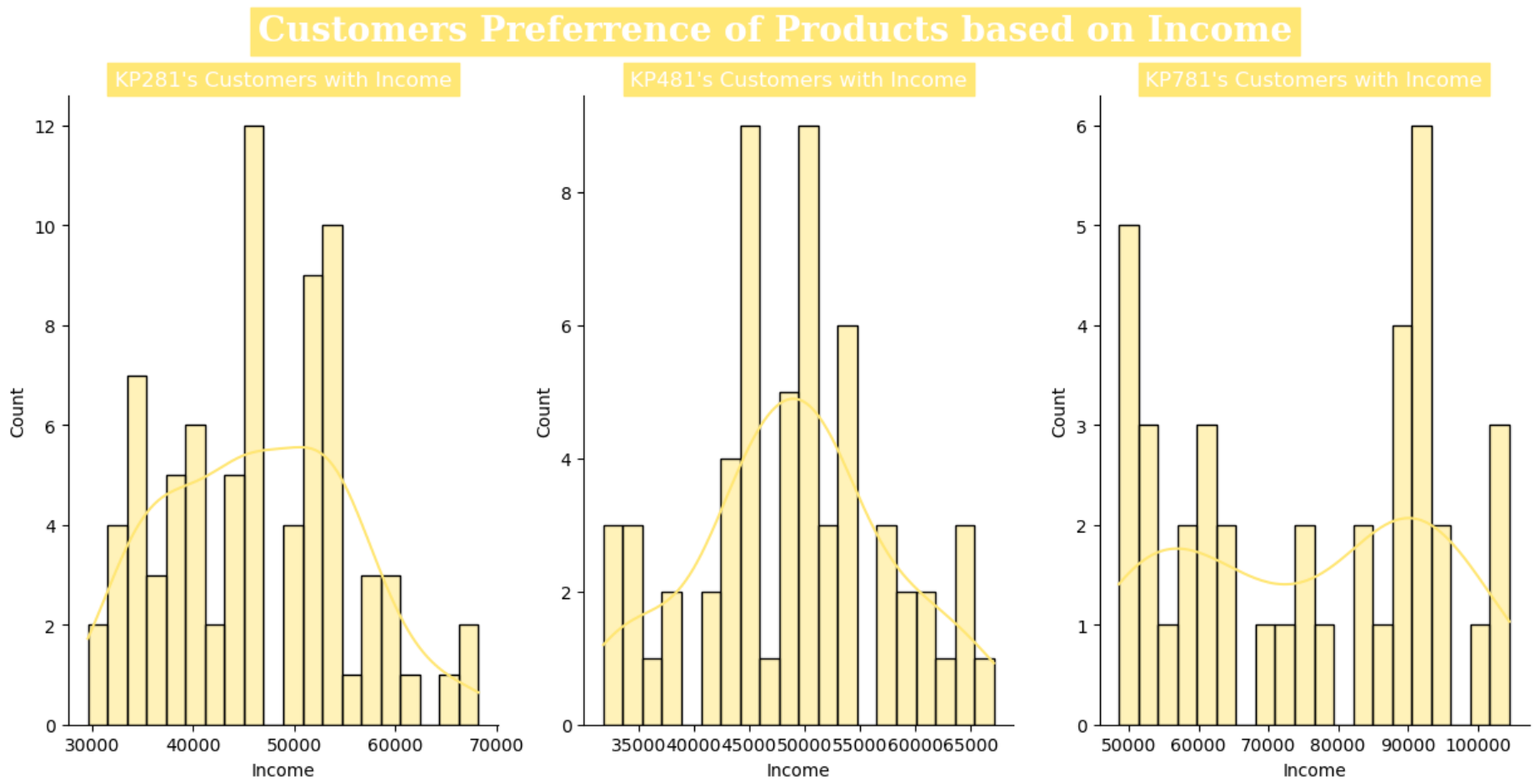


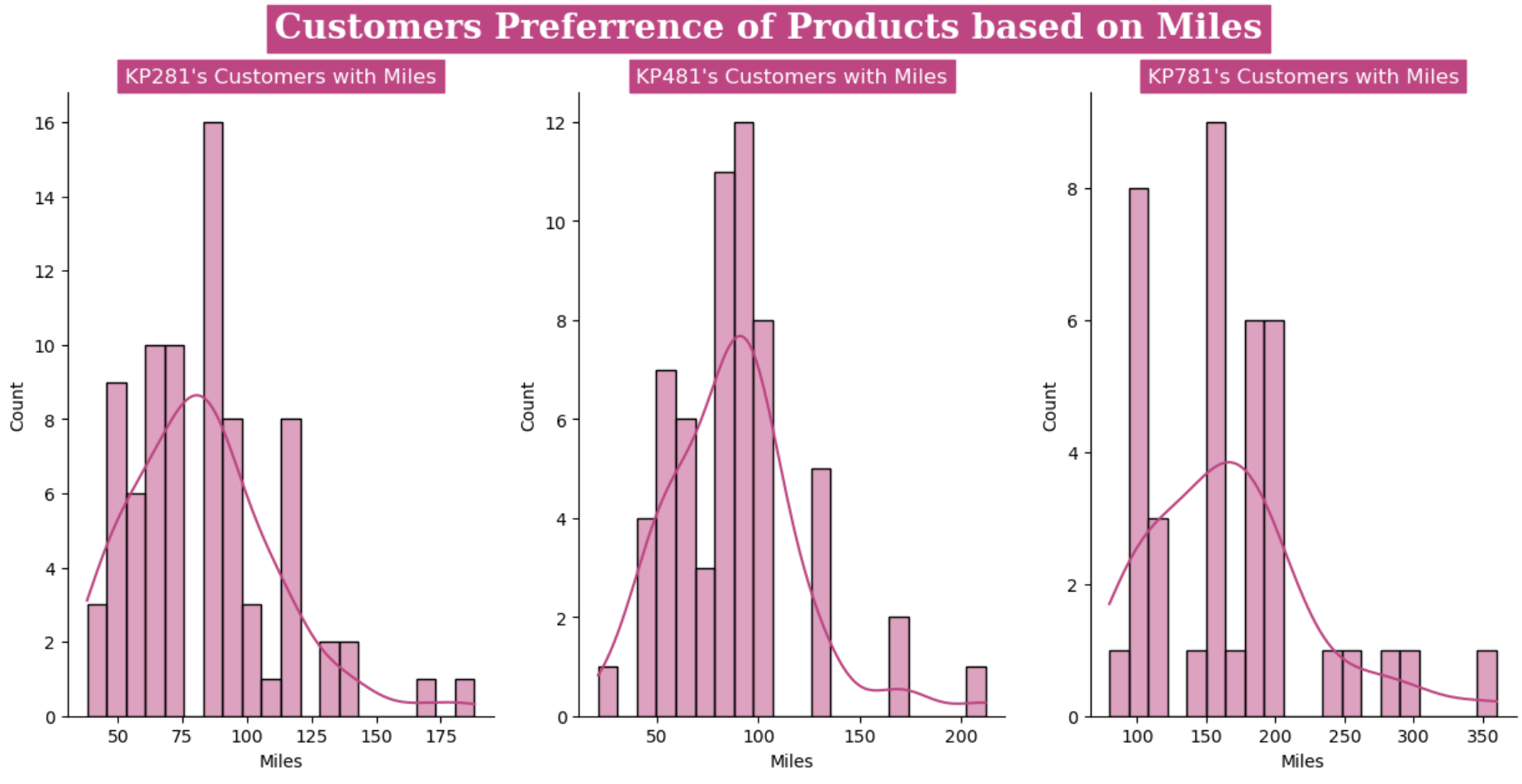
Customers Preferrence of Products based on Usage



Customers Preferrence of Products based on Fitness







Probability:

From Given Data:

Marginal probability

--> **44.44%** of customers bought **KP281**

--> **33.33%** of customers bought **KP481**

--> **22.22%** of customers bought **KP781**

🏷️ Finding ***Conditional Probabilities*** :

- **Male and Single :**

In [139…

```
pms = af[(af['Gender']=='Male') & (af['MaritalStatus']=='Single')]
v1=round(len(pms[pms['Product']=='KP281'])/(len(pms))*100,2)
v2=round(len(pms[pms['Product']=='KP481'])/(len(pms))*100,2)
v3=round(len(pms[pms['Product']=='KP781'])/(len(pms))*100,2)
print('Probability of Male and Single for buying')
print(f'- KP281 is {v1}%\n- KP481 is {v2}%\n- KP781 is {v3}%')
```

```
Probability of Male and Single for buying
- KP281 is 44.19%
- KP481 is 23.26%
- KP781 is 32.56%
```

- **Male and Partnered :**

In [143…
```
pmp = af[(af['Gender']=='Male') & (af['MaritalStatus']=='Partnered')]
v1=round(len(pmp[pmp['Product']=='KP281'])/(len(pmp))*100,2)
v2=round(len(pmp[pmp['Product']=='KP481'])/(len(pmp))*100,2)
v3=round(len(pmp[pmp['Product']=='KP781'])/(len(pmp))*100,2)
print('Probability of Male and Partnered for buying')
print(f'- KP281 is {v1}%\n- KP481 is {v2}%\n- KP781 is {v3}%')
```

```
Probability of Male and Partnered for buying
- KP281 is 34.43%
- KP481 is 34.43%
- KP781 is 31.15%
```

- **Female and Single :**

In [144…
```
pfs = af[(af['Gender']=='Female') & (af['MaritalStatus']=='Single')]
v1=round(len(pfs[pfs['Product']=='KP281'])/(len(pfs))*100,2)
v2=round(len(pfs[pfs['Product']=='KP481'])/(len(pfs))*100,2)
v3=round(len(pfs[pfs['Product']=='KP781'])/(len(pfs))*100,2)
print('Probability of Female and Single for buying')
print(f'- KP281 is {v1}%\n- KP481 is {v2}%\n- KP781 is {v3}%')
```

```
Probability of Female and Single for buying
- KP281 is 43.33%
- KP481 is 46.67%
- KP781 is 10.0%
```

- **Female and Partnered :**

In [145…
```
pfp = af[(af['Gender']=='Female') & (af['MaritalStatus']=='Partnered')]
v1=round(len(pfp[pfp['Product']=='KP281'])/(len(pfp))*100,2)
v2=round(len(pfp[pfp['Product']=='KP481'])/(len(pfp))*100,2)
v3=round(len(pfp[pfp['Product']=='KP781'])/(len(pfp))*100,2)
print('Probability of Female and Partnered for buying')
print(f'- KP281 is {v1}%\n- KP481 is {v2}%\n- KP781 is {v3}%')
```

```
Probability of Female and Partnered for buying
- KP281 is 58.7%
- KP481 is 32.61%
- KP781 is 8.7%
```

🔍 Insights:

- `Probability` of Buying *KP281* increased from 44.44% to **58.7%**, if the customer is `Female and Partnered`.
- `Probability` of Buying *KP481* increased from 33.33% to **46.67%**, if the customer is `Female and Single`.
- `Probability` of Buying *KP781* increased from 22.22% to **32.56%**, if the customer is `Male and Single`.
- `Probability` of Buying *KP481 & KP781* increased from 33.33% & 22.22% to **34.43%**, if the customer is `Male and Single`.
- `Probability` of Buying *KP781* decreased from 22.22% to **8.7%**, if the customer is `Female and Partnered`.

In [147…
```
mb = af.groupby(['MaritalStatus','Gender','Product']).agg(count=('Product','count'))
mb
```

Out[147]:

			count
MaritalStatus	**Gender**	**Product**	
Partnered	**Female**	**KP281**	27
		KP481	15
		KP781	4
	Male	**KP281**	21
		KP481	21
		KP781	19
Single	**Female**	**KP281**	13
		KP481	14
		KP781	3
	Male	**KP281**	19
		KP481	10
		KP781	14

In [148…
```
maf = af[['Product','Gender','MaritalStatus']].melt()
maf
```

Out[148]:

	variable	value
0	Product	KP281
1	Product	KP281
2	Product	KP281
3	Product	KP281
4	Product	KP281
...	...	...
535	MaritalStatus	Single
536	MaritalStatus	Single
537	MaritalStatus	Single
538	MaritalStatus	Partnered
539	MaritalStatus	Partnered

540 rows × 2 columns

In [149…
```
maff = maf.groupby(['variable','value']).agg(cust_cnt=('value','count'))
maff
```

Out[149]:

		cust_cnt
variable	**value**	
Gender	**Female**	76
	Male	104
MaritalStatus	**Partnered**	107
	Single	73
Product	**KP281**	80
	KP481	60
	KP781	40

👉Probability of Product purchase with respect to gender:

In [266…
```
gct = round(pd.crosstab(af.Product,af.Gender,margins=True,normalize=True)*100,2)
gct
```

Out[266]:

Gender	Female	Male	All
Product			
KP281	22.22	22.22	44.44
KP481	16.11	17.22	33.33
KP781	3.89	18.33	22.22
All	42.22	57.78	100.00

🔍 Insights

The **`Probability`** of a treadmill being purchased by a **`female is 42%`**.

- **`The conditional probability`** of purchasing the treadmill model given that the customer is `Female` is
  - For Treadmill model KP281 - **22%**
  - For Treadmill model KP481 - **16%**
  - For Treadmill model KP781 - **4%**

The **`Probability`** of a treadmill being purchased by a **`male is 58%`**.

- **`The conditional probability`** of purchasing the treadmill model given that the customer is **`Male`** is -
  - For Treadmill model KP281 - **22%**
  - For Treadmill model KP481 - **17%**
  - For Treadmill model KP781 - **18%**

In [283…
```
gcct = round(pd.crosstab(index=af.Product,columns=af.Gender,normalize='columns')*100,2)
gcct
```

Out[283]:

Gender	Female	Male
Product		
KP281	52.63	38.46
KP481	38.16	29.81
KP781	9.21	31.73

🔍 Insights:

- The `Probability` **of Customer purchasing the Product Genderwise (only males or only females)**
  - P(KP281 | Female) = 52.63 %
  - P(KP481 | Female) = 38.16 %
  - P(KP781 | Female) = 9.21 %
  - P(KP281 | male) = 38.46 %
  - P(KP481 | male) = 29.81 %
  - P(KP781 | male) = 31.73 %
- Probability of *Female* customer buying `KP281` (52.63%) which is more than *male*(38.46%).
- we can say that **KP281** is more preferred by ***Female*** customers.
- Probability of **Male** customer buying Product `KP781` (31.73%) is way more than female(9.21%).
- Probability of **Female** customer buying Product `KP481` (38.15%) is significantly higher than male (29.80%).

👉🏻Probability of Product purchase with respect to MaritalStatus:

In [287…

```
(pd.crosstab(index =af['Product'],columns = af['MaritalStatus'],margins = True,normalize = True)*100).round(2)
# round(pd.crosstab(af.Product,af.MaritalStatus,margins=True,normalize=True)*100,2)
```

Out[287]:

MaritalStatus	Partnered	Single	All
Product			
KP281	26.67	17.78	44.44
KP481	20.00	13.33	33.33
KP781	12.78	9.44	22.22
All	59.44	40.56	100.00

🔍 Insights

1. The `Probability` of a treadmill being purchased by a `Married/partnered Customer is 59%`.
   - `The conditional probability` of purchasing the treadmill model given that the customer is `Married/partnered` is
     - For Treadmill model KP281 - **27%**
     - For Treadmill model KP481 - **20%**
     - For Treadmill model KP781 - **13%**
1. The `Probability` of a treadmill being purchased by a `Single Customer is 41%`.
   - `The conditional probability` of purchasing the treadmill model given that the customer is `Single` is -
     - For Treadmill model KP281 - **18%**
     - For Treadmill model KP481 - **13%**
     - For Treadmill model KP781 - **9%**

In [289…

```
(pd.crosstab(index =af['Product'],columns = af['MaritalStatus'],normalize = 'columns')*100).round(2)
```

Out[289]:

Product	MaritalStatus	Partnered	Single
KP281		44.86	43.84
KP481		33.64	32.88
KP781		21.50	23.29

🔍 Insights

1. The `Probability` of a treadmill being purchased
   - `The conditional probability` of purchasing the treadmill model given that the customer is only `Married/partnered` is
     - For Treadmill model KP281 - **45%**
     - For Treadmill model KP481 - **34%**
     - For Treadmill model KP781 - **21%**

```
- **`The conditional probability`** of purchasing the treadmill model given that the customer is only
**`Single`** is -

    - For Treadmill model KP281 - **43%**

    - For Treadmill model KP481 - **32%**

    - For Treadmill model KP781 - **23%**
```

👉Probability of Product purchase with respect to Usage:

In [151…

```
uct = round(pd.crosstab(af.Product,af.Usage,margins=True,normalize=True)*100,2)
uct
```

Out[151]:

Product	Usage	2	3	4	5	6	7	All
KP281		10.56	20.56	12.22	1.11	0.00	0.00	44.44
KP481		7.78	17.22	6.67	1.67	0.00	0.00	33.33
KP781		0.00	0.56	10.00	6.67	3.89	1.11	22.22
All		18.33	38.33	28.89	9.44	3.89	1.11	100.00

🔍 Insights

1. The `Probability` of a treadmill being purchased by a customer with `Usage 3 per week is 38%`.
   - `The conditional probability` of purchasing the treadmill model given that the customer has `Usage 3 per week` is -
     - For Treadmill model KP281 - **21%**
     - For Treadmill model KP481 - **17%**
     - For Treadmill model KP781 - **1%**
1. The `Probability` of a treadmill being purchased by a customer with `Usage 4 per week is 29%`.
   - `The conditional probability` of purchasing the treadmill model given that the customer has `Usage 4 per week` is -
     - For Treadmill model KP281 - **12%**
     - For Treadmill model KP481 - **7%**
     - For Treadmill model KP781 - **10%**
2. The `Probability` of a treadmill being purchased by a customer with `Usage 2 per week is 18%`
   - `The conditional probability` of purchasing the treadmill model given that the customer has `Usage 2 per week` is -
     - For Treadmill model KP281 - **11%**
     - For Treadmill model KP481 - **8%**
     - For Treadmill model KP781 - **0%**

👉Probability of Product purchase with respect to Fitness:

In [154…
```
fcct = pd.crosstab(af.Product,af.Fitness_comment,margins=True).T
fcct
```

Out[154]:

Product	KP281	KP481	KP781	All
Fitness_comment				
Average Shape	54	39	4	97
Bad Shape	14	12	0	26
Excellent Shape	2	0	29	31
Good Shape	9	8	7	24
Poor Shape	1	1	0	2
All	80	60	40	180

In [153…
```
fcct = np.round(pd.crosstab(af.Product,af.Fitness_comment,normalize='columns')*100,2)
fcct.T
```

Out[153]:

Product	KP281	KP481	KP781
Fitness_comment			
Average Shape	55.67	40.21	4.12
Bad Shape	53.85	46.15	0.00
Excellent Shape	6.45	0.00	93.55
Good Shape	37.50	33.33	29.17
Poor Shape	50.00	50.00	0.00

In [155…
```
fcct = np.round(pd.crosstab(af.Product,af.Fitness_comment,margins=True,normalize=True)*100,2)
fcct.T
```

Out[155]:

Product	KP281	KP481	KP781	All
Fitness_comment				
Average Shape	30.00	21.67	2.22	53.89
Bad Shape	7.78	6.67	0.00	14.44
Excellent Shape	1.11	0.00	16.11	17.22
Good Shape	5.00	4.44	3.89	13.33
Poor Shape	0.56	0.56	0.00	1.11
All	44.44	33.33	22.22	100.00

🔍 Insights

- The `Probability` of a treadmill being purchased by a customer with `Average(3) Fitness is 54%`.
  - `The conditional probability` of purchasing the treadmill model given that the customer has `Average Fitness` is -
    - For Treadmill model KP281 - **30%**
    - For Treadmill model KP481 - **22%**
    - For Treadmill model KP781 - **2%**
- The `Probability` of a treadmill being purchased by a customer with `Fitness of 2,4,5 is almost 15%`.
- The `Probability` of a treadmill being purchased by a customer with `very low(1) Fitness is only 1%`.

👉🏻Probability of Product purchase with respect to Fitness Genderwise:

In [162…
```
pd.crosstab(index=[af.Product,af.Fitness_comment],columns=af.Gender).T
```

Out[162]:

Product	KP281					KP481				KP781		
Fitness_comment	**Average Shape**	**Bad Shape**	**Excellent Shape**	**Good Shape**	**Poor Shape**	**Average Shape**	**Bad Shape**	**Good Shape**	**Poor Shape**	**Average Shape**	**Excellent Shape**	**Good Shape**
Gender												
Female	26	10	1	3	0	18	6	4	1	1	5	1
Male	28	4	1	6	1	21	6	4	0	3	24	6

In [161…
```
np.round(pd.crosstab(index=[af.Product,af.Fitness_comment],columns=af.Gender,margins=True,normalize=True)*100,2).T
```

Out[161]:

Product	KP281					KP481						KP781	All
Fitness_comment	Average Shape	Bad Shape	Excellent Shape	Good Shape	Poor Shape	Average Shape	Bad Shape	Good Shape	Poor Shape	Average Shape	Excellent Shape	Good Shape	
Gender													
Female	14.44	5.56	0.56	1.67	0.00	10.00	3.33	2.22	0.56	0.56	2.78	0.56	42.22
Male	15.56	2.22	0.56	3.33	0.56	11.67	3.33	2.22	0.00	1.67	13.33	3.33	57.78
All	30.00	7.78	1.11	5.00	0.56	21.67	6.67	4.44	0.56	2.22	16.11	3.89	100.00

In [163…
```
np.round(pd.crosstab(index=[af.Product,af.Fitness_comment],columns=af.Gender,normalize='columns')*100,2).T
```

Out[163]:

Product	KP281					KP481						KP781
Fitness_comment	Average Shape	Bad Shape	Excellent Shape	Good Shape	Poor Shape	Average Shape	Bad Shape	Good Shape	Poor Shape	Average Shape	Excellent Shape	Good Shape
Gender												
Female	34.21	13.16	1.32	3.95	0.00	23.68	7.89	5.26	1.32	1.32	6.58	1.32
Male	26.92	3.85	0.96	5.77	0.96	20.19	5.77	3.85	0.00	2.88	23.08	5.77

- Probability of **Average shape** Customers who are `Females` Purchasing any product is significantly higher than others.

👉Probability of Product purchase with respect to Age:

In [164…
```
pd.crosstab(af.Product,af.age_category,margins=True).T
```

Out[164]:

Product	KP281	KP481	KP781	All
age_category				
Teenage	6	4	0	10
Adults	60	48	34	142
Middle Aged	11	7	4	22
Elderly	3	1	2	6
All	80	60	40	180

In [165…
```
round(pd.crosstab(af.Product,af.age_category,normalize='columns')*100,2).T
```

Out[165]:

Product	KP281	KP481	KP781
age_category			
Teenage	60.00	40.00	0.00
Adults	42.25	33.80	23.94
Middle Aged	50.00	31.82	18.18
Elderly	50.00	16.67	33.33

In [166…
```
round(pd.crosstab(af.Product,af.age_category,margins=True,normalize=True)*100,2).T
```

Out[166]:

Product	KP281	KP481	KP781	All
age_category				
Teenage	3.33	2.22	0.00	5.56
Adults	33.33	26.67	18.89	78.89
Middle Aged	6.11	3.89	2.22	12.22
Elderly	1.67	0.56	1.11	3.33
All	44.44	33.33	22.22	100.00

🔍 Insights

The `Probability` of a treadmill being purchased by a `Teens(0-20) is 6%`.

- `The conditional probability` of purchasing the treadmill model given that the customer is `Teens` is
  - For Treadmill model KP281 - **3%**
  - For Treadmill model KP481 - **2%**
  - For Treadmill model KP781 - **0%**

The `Probability` of a treadmill being purchased by a `Adults(20-35) is 79%`.

- `The conditional probability` of purchasing the treadmill model given that the customer is `Adult` is -
  - For Treadmill model KP281 - **33%**
  - For Treadmill model KP481 - **26%**
  - For Treadmill model KP781 - **19%**
- The `Probability` of a treadmill being purchased by a `Middle Aged(36-45) is 12%`.
- The `Probability` of a treadmill being purchased by a `Elder(Above 45) is only 3%`.

👉Probability of Product purchase with respect to Income:

In [168…
```
pd.crosstab(index=af.Product , columns=af.income_grp).T
```

Out[168]:

Product	KP281	KP481	KP781
income_grp			
middle_class	48	30	5
upper_middle_class	32	30	14
high_class	0	0	21

In [170…
```
np.round(pd.crosstab(index=af.Product , columns=af.income_grp , normalize='columns')*100,2).T
```

Out[170]:

Product	KP281	KP481	KP781
income_grp			
middle_class	57.83	36.14	6.02
upper_middle_class	42.11	39.47	18.42
high_class	0.00	0.00	100.00

In [169…
```
np.round(pd.crosstab(index=af.Product , columns=af.income_grp , margins=True , normalize=True)*100,2).T
```

Out[169]:

Product	KP281	KP481	KP781	All
income_grp				
middle_class	26.67	16.67	2.78	46.11
upper_middle_class	17.78	16.67	7.78	42.22
high_class	0.00	0.00	11.67	11.67
All	44.44	33.33	22.22	100.00

🔍 Insights

- The `Probability` of a treadmill being purchased by a customer with `Middle Class is 46%`.
  - `The conditional probability` of purchasing the treadmill model given that the customer has `Middle Class` is -
    - For Treadmill model KP281 - **26%**
    - For Treadmill model KP481 - **17%**
    - For Treadmill model KP781 - **3%**
- The `Probability` of a treadmill being purchased by a customer with `Upper Middle cLass is 42%`.
  - `The conditional probability` of purchasing the treadmill model given that the customer has `Upper Middle cLass` is -
    - For Treadmill model KP281 - **18%**
    - For Treadmill model KP481 - **17%**
    - For Treadmill model KP781 - **8%**
- The `Probability` of a treadmill being purchased by a customer with `High Class is 12%`
  - `The conditional probability` of purchasing the treadmill model given that the customer has `High Class` is -
    - For Treadmill model KP281 - **0%**
    - For Treadmill model KP481 - **0%**
    - For Treadmill model KP781 - **12%**

👉Probability of Product purchase with respect to Miles ran:

In [295…
```
(pd.crosstab(index =af['Product'],columns = af['Miles'],margins = True,normalize = True)*100).round(2).T
```

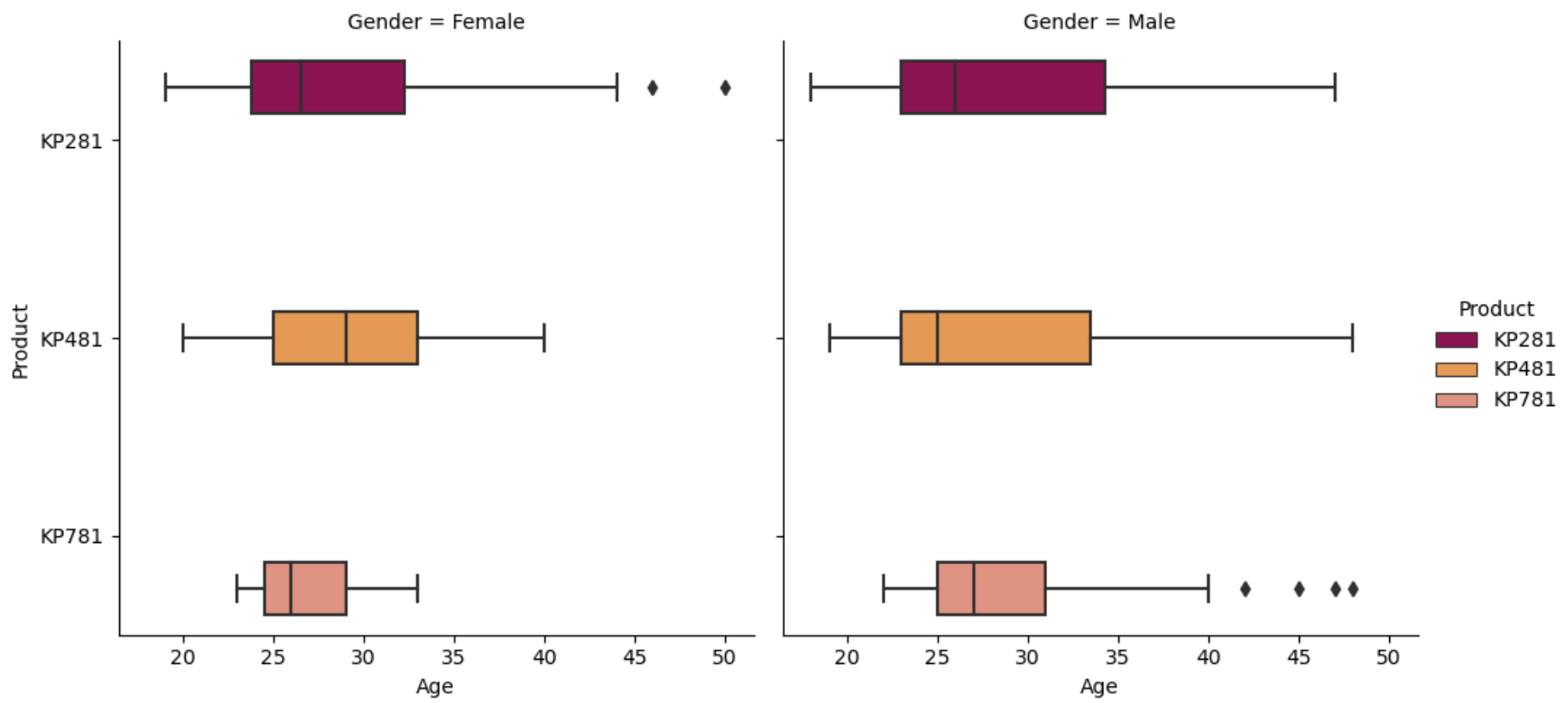
Out[295]:

Product Miles	KP281	KP481	KP781	All
21	0.00	0.56	0.00	0.56
38	1.67	0.00	0.00	1.67
42	0.00	2.22	0.00	2.22
47	5.00	0.00	0.00	5.00
53	0.00	3.89	0.00	3.89
56	3.33	0.00	0.00	3.33
64	0.00	3.33	0.00	3.33
66	5.56	0.00	0.00	5.56
74	0.00	1.67	0.00	1.67
75	5.56	0.00	0.00	5.56
80	0.00	0.00	0.56	0.56
85	8.89	6.11	0.00	15.00
94	4.44	0.00	0.00	4.44
95	0.00	6.67	0.00	6.67
100	0.00	0.00	3.89	3.89
103	1.67	0.00	0.00	1.67
106	0.00	4.44	0.56	5.00
112	0.56	0.00	0.00	0.56
113	4.44	0.00	0.00	4.44
120	0.00	0.00	1.67	1.67
127	0.00	2.78	0.00	2.78
132	1.11	0.00	0.00	1.11
140	0.00	0.00	0.56	0.56
141	1.11	0.00	0.00	1.11
150	0.00	0.00	2.22	2.22
160	0.00	0.00	2.78	2.78
169	0.56	0.00	0.00	0.56
170	0.00	1.11	0.56	1.67
180	0.00	0.00	3.33	3.33
188	0.56	0.00	0.00	0.56
200	0.00	0.00	3.33	3.33
212	0.00	0.56	0.00	0.56
240	0.00	0.00	0.56	0.56
260	0.00	0.00	0.56	0.56
280	0.00	0.00	0.56	0.56
300	0.00	0.00	0.56	0.56
360	0.00	0.00	0.56	0.56
All	44.44	33.33	22.22	100.00

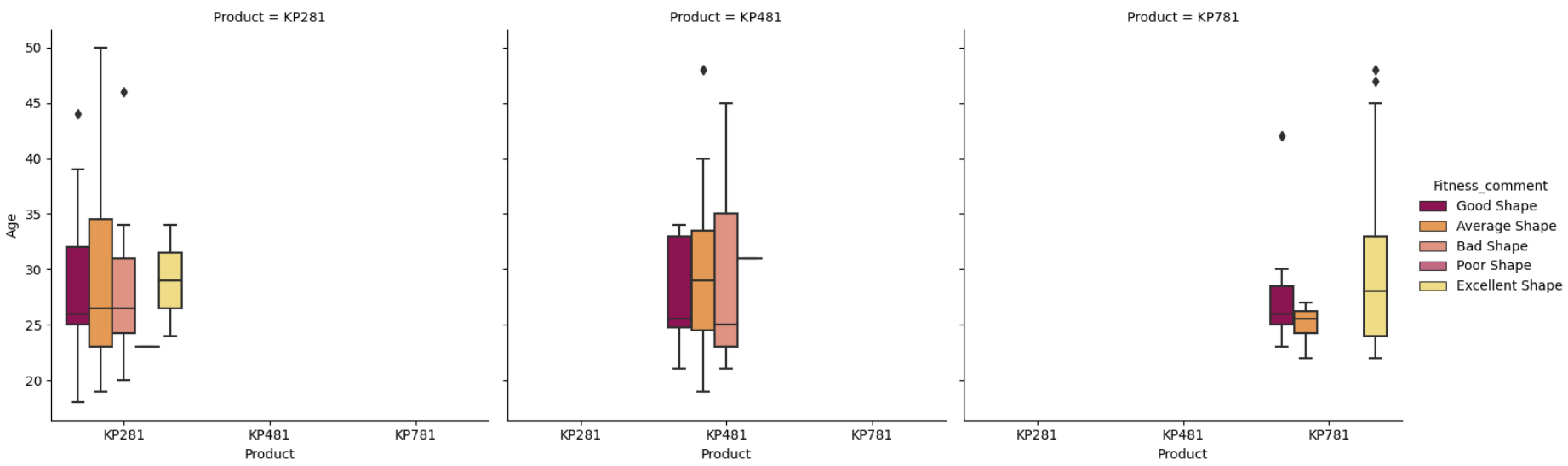
🔍 Insights

- The `Probability` of a treadmill being purchased by Customer who ran `85 Miles` is
  - For Treadmill model KP281 - **9%**
  - For Treadmill model KP281 - **6%**
- The `Probability` of a treadmill being purchased by Customer who ran `100 Miles` is
  - For Treadmill model KP781 - **4%**
- The `Probability` of a treadmill being purchased by Customer who ran `Above 150 Miles` is
  - For Treadmill model KP781 - **18%**

In [171…
```
# Product used among age group seggregated by Gender
sns.catplot(x='Age',y='Product',hue='Product',col='Gender',data=af,kind='box',palette=cp4)
plt.show()
```



In [172…
```
# Product used among age group seggregated by Gender
sns.catplot(y='Age',x='Product',hue='Fitness_comment',col='Product',data=af,kind='box',palette=cp4)
plt.show()
```



In [173…
```
# Miles covered in each product by gender and their marital status

# sns.stripplot(x='Gender',y='Miles',hue='Product',data=af,palette=cp,marker='*')

sns.catplot(x='Gender',y='Miles',hue='Product',col='MaritalStatus',data=af,kind='strip',palette=cp,marker='*')
plt.show()
```

■ 🔍 Theoretical outlier detection: ❕

In [315…
```
num_cols = ['Age','Education','Usage','Fitness','Income','Miles']
```

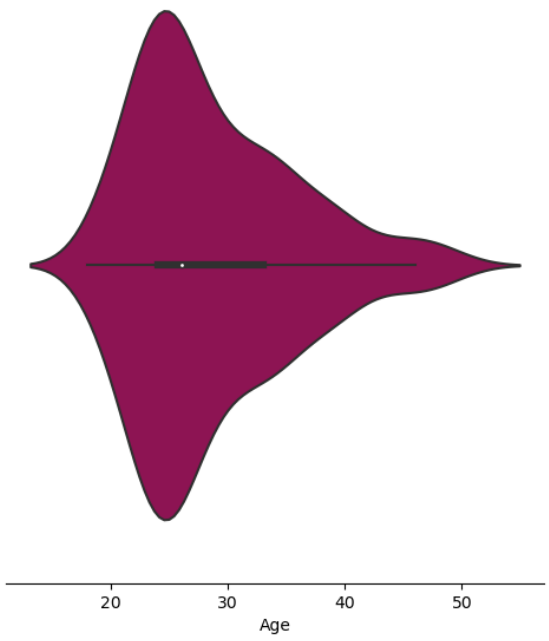
```
for i in range(len(num_cols)):
    data = af[num_cols[i]].tolist()
    mini = np.min(data)
    Q1 = np.percentile(data, 25)
    Q2 = np.median(data)
    Q3 = np.percentile(data, 75)
    maxi = np.max(data)
    IQR = Q3 - Q1

    lo = Q1 - (1.5 * IQR)
    ho = Q3 + (1.5 * IQR)

    lower_outliers=[]
    upper_outliers=[]
    for k in data:
        if k < lo:
            lower_outliers.append(k)

        elif k > ho:
            upper_outliers.append(k)

    uo_pct = round((len(upper_outliers)*100/af.shape[0]),2)
    lo_pct = round((len(lower_outliers)*100/af.shape[0]),2)

    print()
    print(f"Outlier detection of {num_cols[i]}")
    print('.'*30)
    print("Minimum:", mini)
    print("Maximum:", maxi)
    print(f'Initial Range (with outlier) : {(maxi-mini)}')
    print("Q1:", Q1)
    print("Q2:", Q2)
    print("Q3:", Q3)
    print("IQR:", IQR)
    print(f'Final Range (without outlier) : {(ho-lo)}')
    print("Lower outliers are:", lower_outliers)
    print("Upper outliers are:", upper_outliers)
    print(f'Lower Outlier Percentage is {lo_pct}%')
    print(f'Upper Outlier Percentage is {uo_pct}%')
    print(f'Overall Outlier Percentage is {(lo_pct+uo_pct)}%')

    if len(set(lower_outliers)):
        print(f'Outlier points towards left of boxplot : {len(set(lower_outliers))} and they are {(set(lower_outliers))}')
    else:
        print(f'Outlier points towards left of boxplot : {len(set(lower_outliers))}')
    if len(set(upper_outliers)):
        print(f'Outlier points towards right of boxplot : {len(set(upper_outliers))} and they are {(set(upper_outliers))}')
    else:
        print(f'Outlier points towards right of boxplot : {len(set(upper_outliers))}')
    print()

    plt.figure(figsize=(20,7))
    plt.style.use('default')
    plt.style.use('seaborn-v0_8-bright')
    plt.suptitle(f'Customers classification Based on {num_cols[i]}',fontfamily='serif',fontweight='bold',fontsize=20,
             backgroundcolor=cp4[i],color='w')


    plt.subplot(1,3,1)
    sns.violinplot(af,x=num_cols[i],color=cp4[i])
    plt.title(f'Violinplot of {num_cols[i]}',fontfamily='serif',fontweight='bold',fontsize=12,
             loc='center',backgroundcolor=cp4[i],color='w')
    plt.yticks([])


    plt.subplot(1,3,2)
    bxp = sns.boxplot(af,x=num_cols[i],color=cp4[i],width=0.5,saturation=97,flierprops={"marker":"s"},
                      medianprops={"color": "k", "linewidth": 6})#,boxprops={"facecolor": (.3, .5, .7, .5)})
    plt.title(f'Box & Whisker plot of {num_cols[i]}',fontfamily='serif',fontweight='bold',fontsize=12,
              loc='center',backgroundcolor=cp4[i],color='w')
    sns.despine(left=True)
    plt.yticks([])

    plt.subplot(1,3,3)
    sns.swarmplot(af,x=num_cols[i],color=cp4[i])
    plt.title(f'Swarmplot of {num_cols[i]}',fontfamily='serif',fontweight='bold',fontsize=12,
              loc='center',backgroundcolor=cp4[i],color='w')
    sns.despine(left=True)
    plt.yticks([])
    print('-'*127)
```

```
Outlier detection of Age
.............................
Minimum: 18
Maximum: 50
Initial Range (with outlier) : 32
Q1: 24.0
Q2: 26.0
Q3: 33.0
IQR: 9.0
Final Range (without outlier) : 36.0
Lower outliers are: []
Upper outliers are: [47, 50, 48, 47, 48]
Lower Outlier Percentage is 0.0%
Upper Outlier Percentage is 2.78%
Overall Outlier Percentage is 2.78%
Outlier points towards left of boxplot : 0
Outlier points towards right of boxplot : 3 and they are {48, 50, 47}

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

Outlier detection of Education
.............................
Minimum: 12
Maximum: 21
Initial Range (with outlier) : 9
Q1: 14.0
Q2: 16.0
Q3: 16.0
IQR: 2.0
Final Range (without outlier) : 8.0
Lower outliers are: []
Upper outliers are: [20, 21, 21, 21]
Lower Outlier Percentage is 0.0%
Upper Outlier Percentage is 2.22%
Overall Outlier Percentage is 2.22%
Outlier points towards left of boxplot : 0
Outlier points towards right of boxplot : 2 and they are {20, 21}

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

Outlier detection of Usage
.............................
Minimum: 2
Maximum: 7
Initial Range (with outlier) : 5
Q1: 3.0
Q2: 3.0
Q3: 4.0
IQR: 1.0
Final Range (without outlier) : 4.0
Lower outliers are: []
Upper outliers are: [6, 6, 6, 7, 6, 7, 6, 6, 6]
Lower Outlier Percentage is 0.0%
Upper Outlier Percentage is 5.0%
Overall Outlier Percentage is 5.0%
Outlier points towards left of boxplot : 0
Outlier points towards right of boxplot : 2 and they are {6, 7}

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

Outlier detection of Fitness
.............................
Minimum: 1
Maximum: 5
Initial Range (with outlier) : 4
Q1: 3.0
Q2: 3.0
Q3: 4.0
IQR: 1.0
Final Range (without outlier) : 4.0
Lower outliers are: [1, 1]
Upper outliers are: []
Lower Outlier Percentage is 1.11%
Upper Outlier Percentage is 0.0%
Overall Outlier Percentage is 1.11%
Outlier points towards left of boxplot : 1 and they are {1}
Outlier points towards right of boxplot : 0

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

Outlier detection of Income
.............................
Minimum: 29562
Maximum: 104581
Initial Range (with outlier) : 75019
Q1: 44058.75
Q2: 50596.5
Q3: 58668.0
IQR: 14609.25
Final Range (without outlier) : 58437.0
Lower outliers are: []
Upper outliers are: [83416, 88396, 90886, 92131, 88396, 85906, 90886, 103336, 99601, 89641, 95866, 92131, 92131, 104581, 83416,
89641, 90886, 104581, 95508]
```

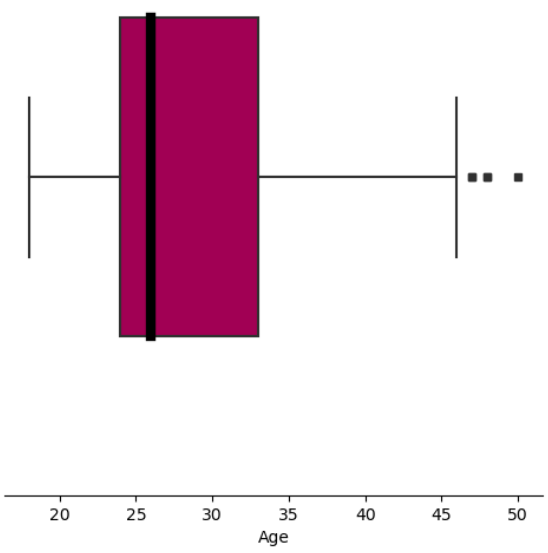
```
Lower Outlier Percentage is 0.0%
Upper Outlier Percentage is 10.56%
Overall Outlier Percentage is 10.56%
Outlier points towards left of boxplot : 0
Outlier points towards right of boxplot : 11 and they are {92131, 104581, 90886, 103336, 89641, 88396, 99601, 85906, 95508, 8341
6, 95866}

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

Outlier detection of Miles
..........................
Minimum: 21
Maximum: 360
Initial Range (with outlier) : 339
Q1: 66.0
Q2: 94.0
Q3: 114.75
IQR: 48.75
Final Range (without outlier) : 195.0
Lower outliers are: []
Upper outliers are: [188, 212, 200, 200, 200, 240, 300, 280, 260, 200, 360, 200, 200]
Lower Outlier Percentage is 0.0%
Upper Outlier Percentage is 7.22%
Overall Outlier Percentage is 7.22%
Outlier points towards left of boxplot : 0
Outlier points towards right of boxplot : 8 and they are {260, 200, 360, 300, 240, 212, 280, 188}

-----------------------------------------------------------------------------------------------------------------------
```

Customers classification Based on Age

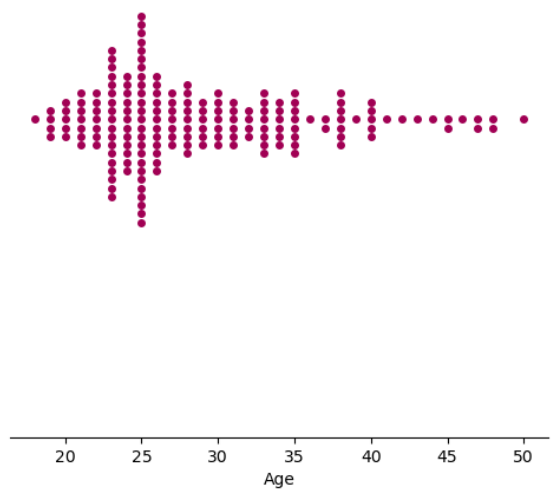
Violinplot of Age



Box & Whisker plot of Age

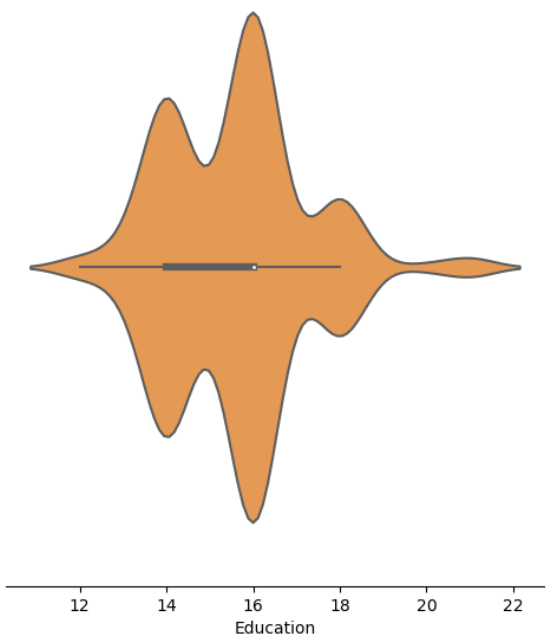


Swarmplot of Age

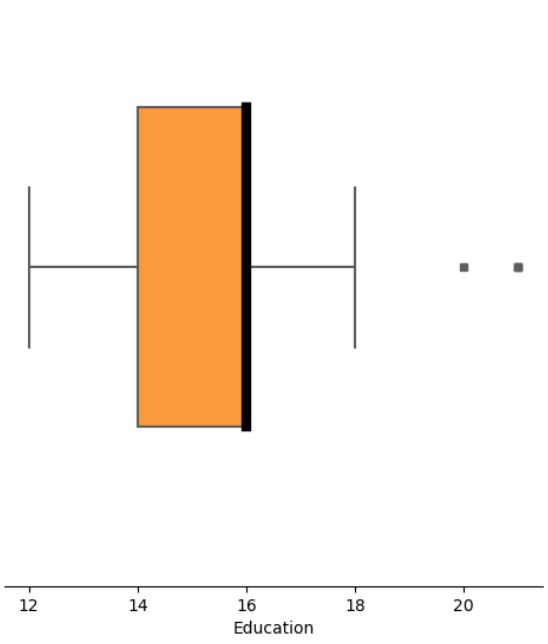


Customers classification Based on Education

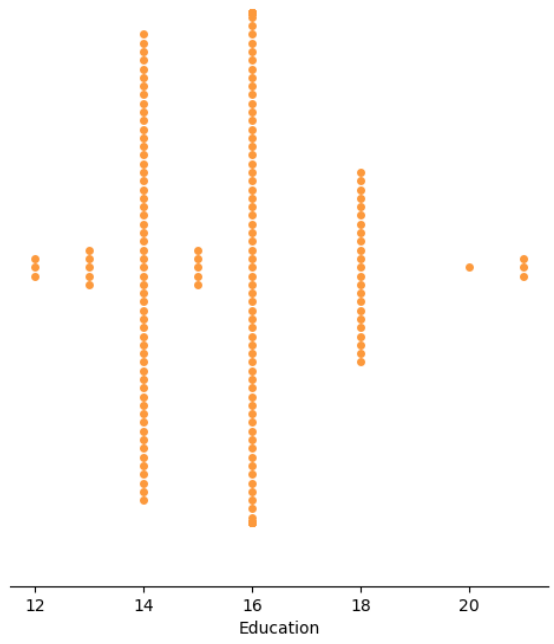
Violinplot of Education



Box & Whisker plot of Education

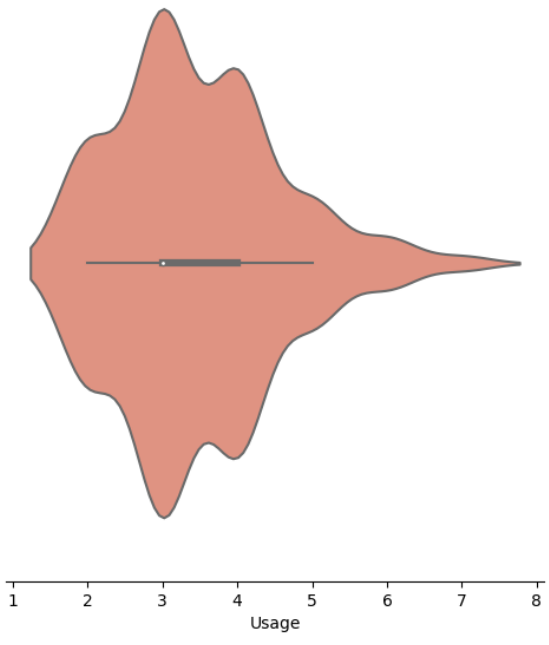


Swarmplot of Education

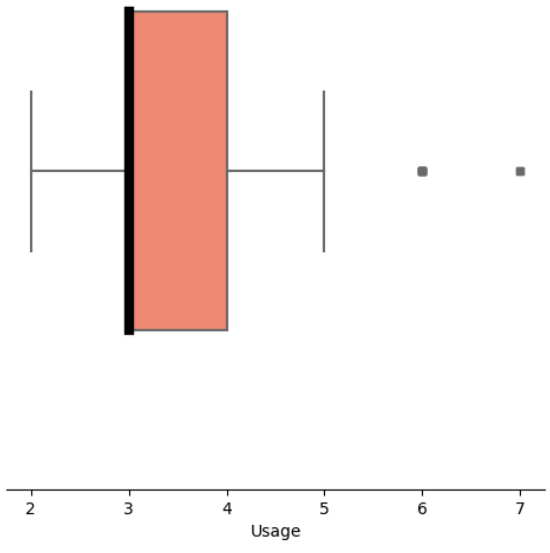


Customers classification Based on Usage

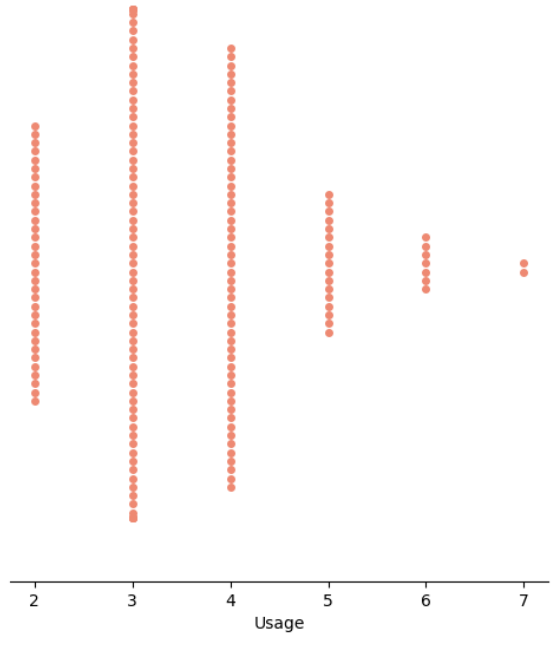
Violinplot of Usage



Box & Whisker plot of Usage

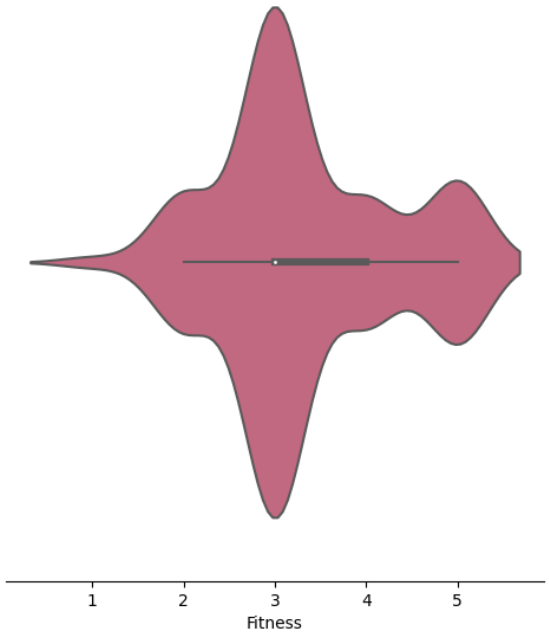


Swarmplot of Usage

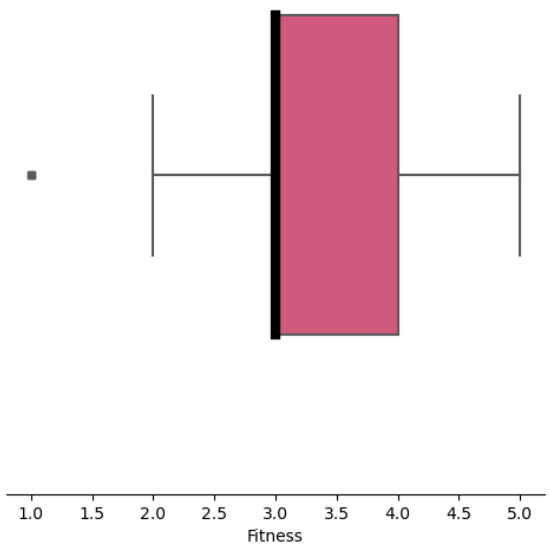


Customers classification Based on Fitness

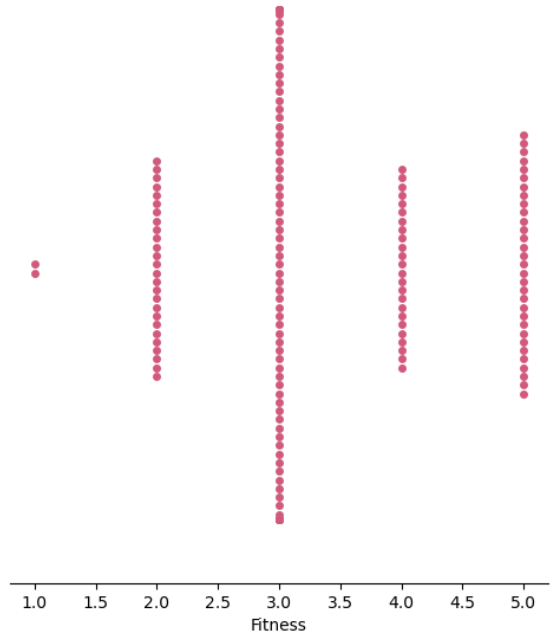
Violinplot of Fitness



Box & Whisker plot of Fitness

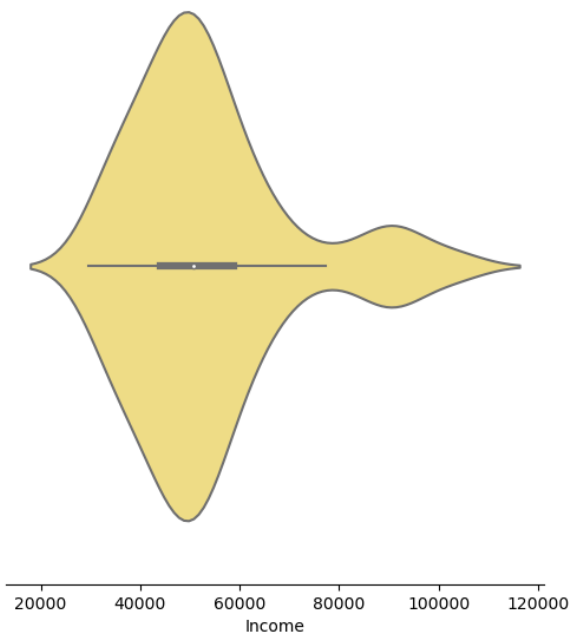


Swarmplot of Fitness

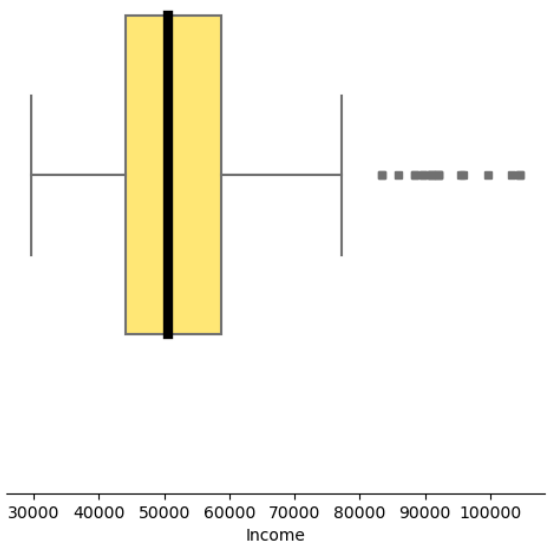


Customers classification Based on Income

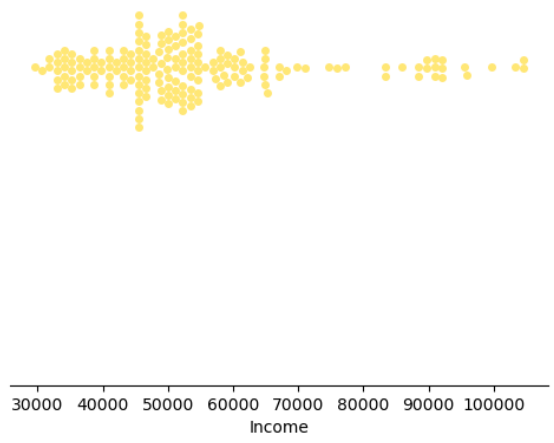
Violinplot of Income



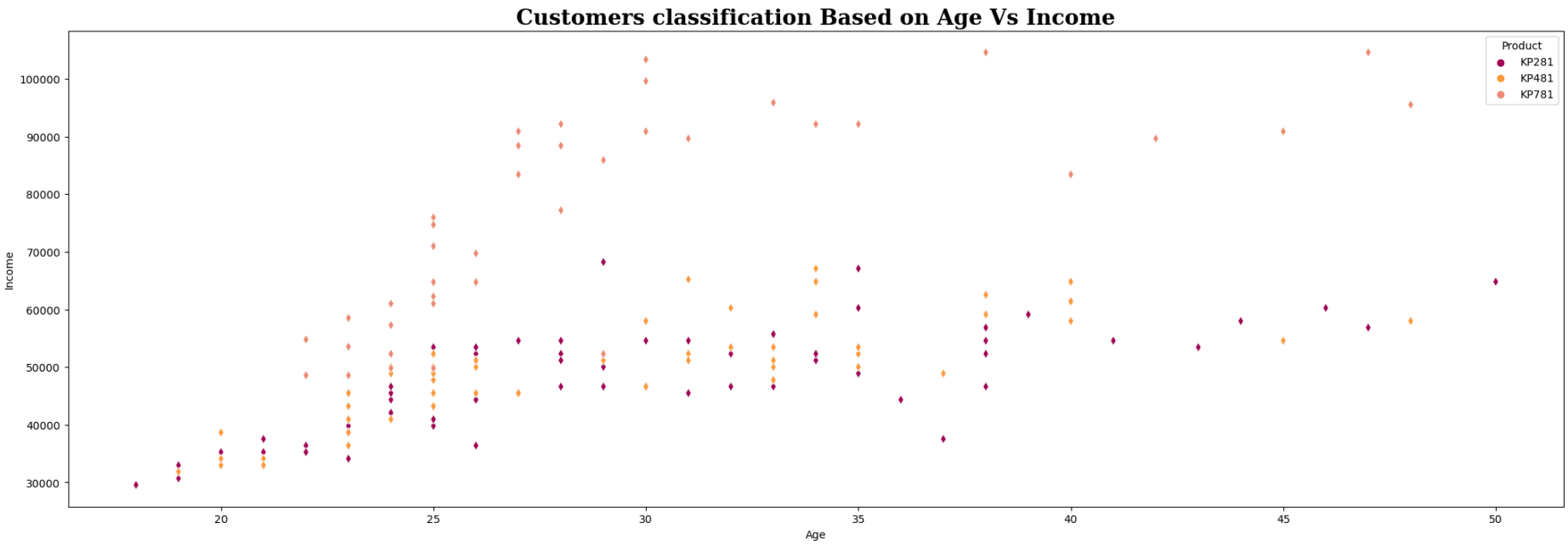
Box & Whisker plot of Income



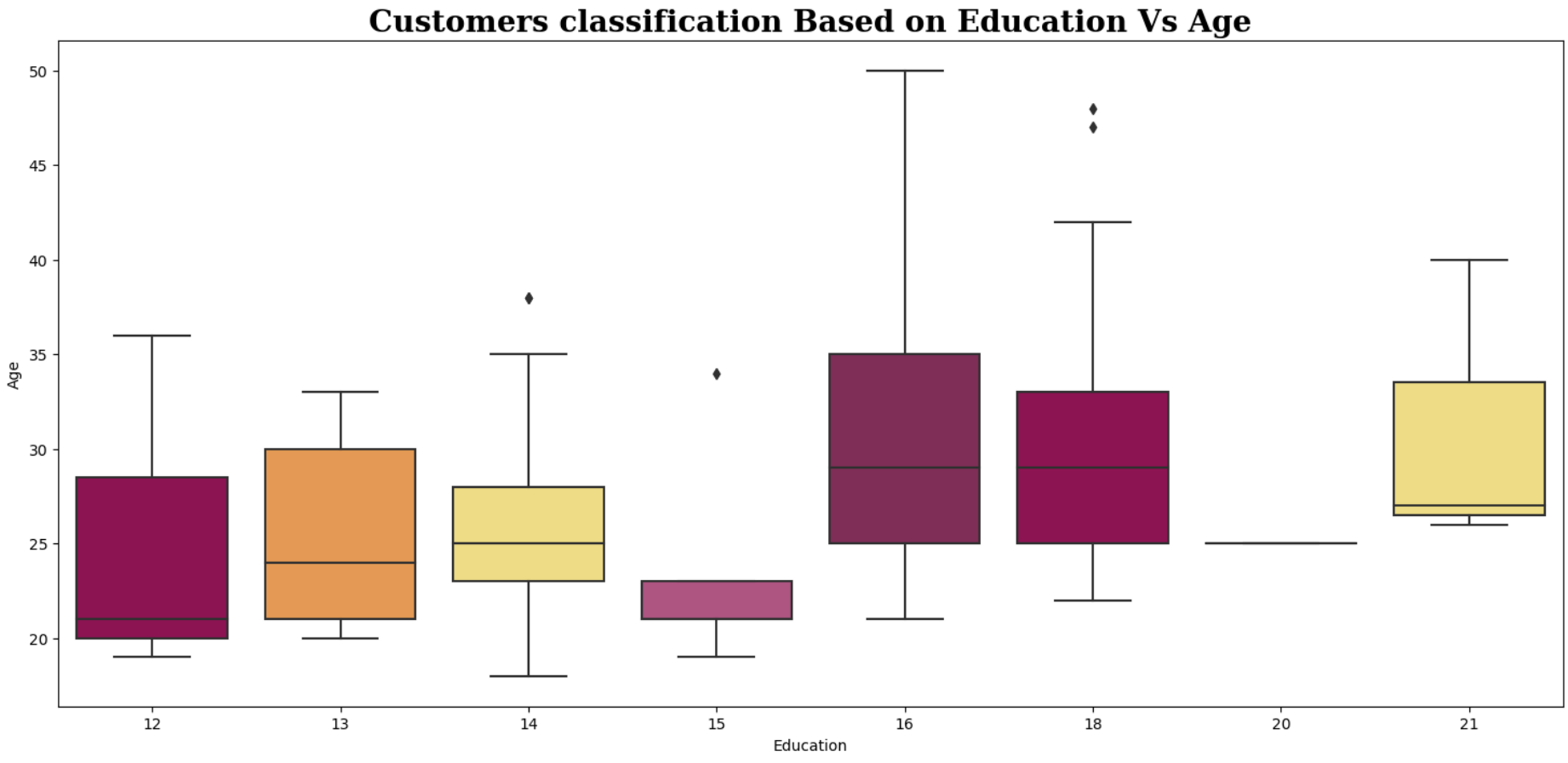
Swarmplot of Income



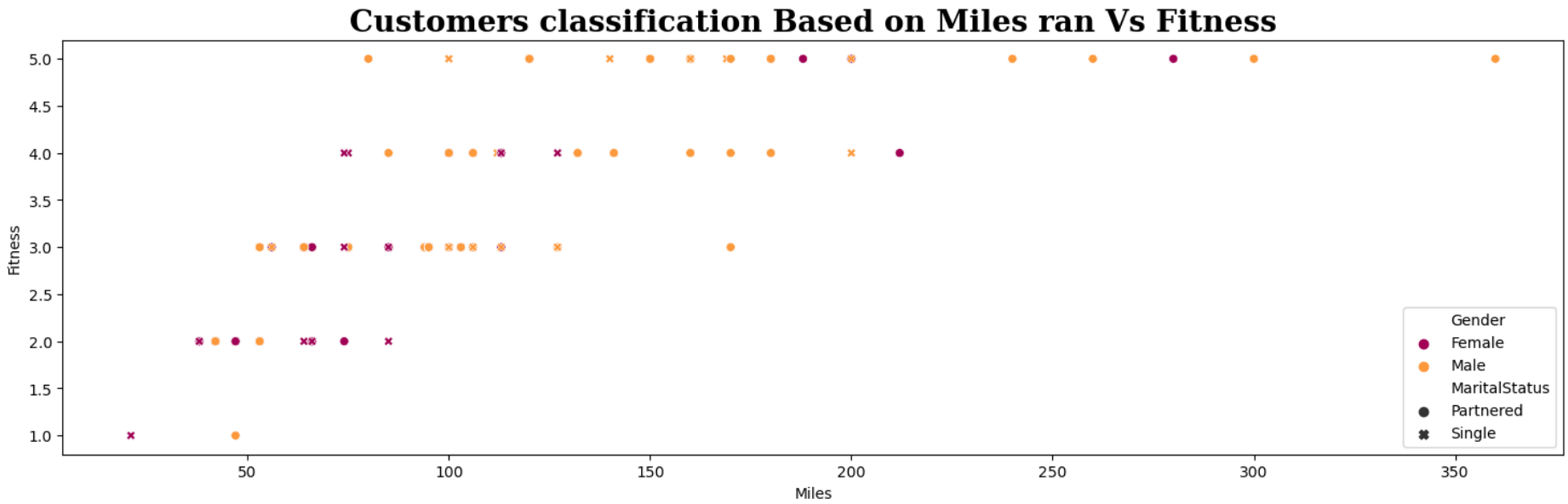
In [243…
```
plt.figure(figsize=(25,8))
sns.scatterplot(data=af, x='Age',y='Income', hue='Product',palette=cp4,marker='d')
plt.title('Customers classification Based on Age Vs Income',fontfamily='serif',fontweight='bold',fontsize=20)
plt.show()
```



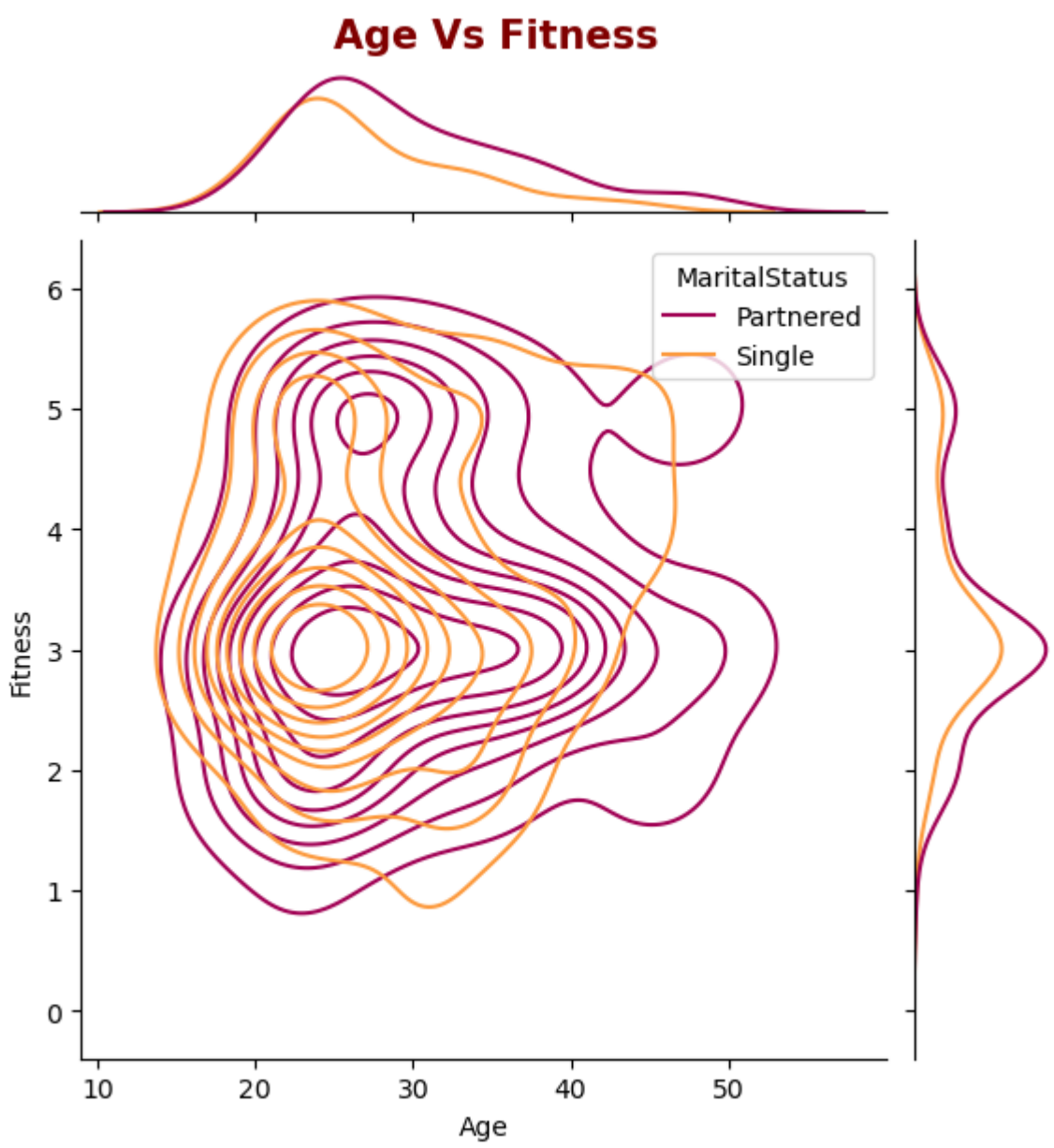
In [302…
```
plt.figure(figsize=(18,8))
sns.boxplot(af,x='Education',y='Age',palette=cp)
plt.title('Customers classification Based on Education Vs Age',fontfamily='serif',fontweight='bold',fontsize=20)
plt.show()
```



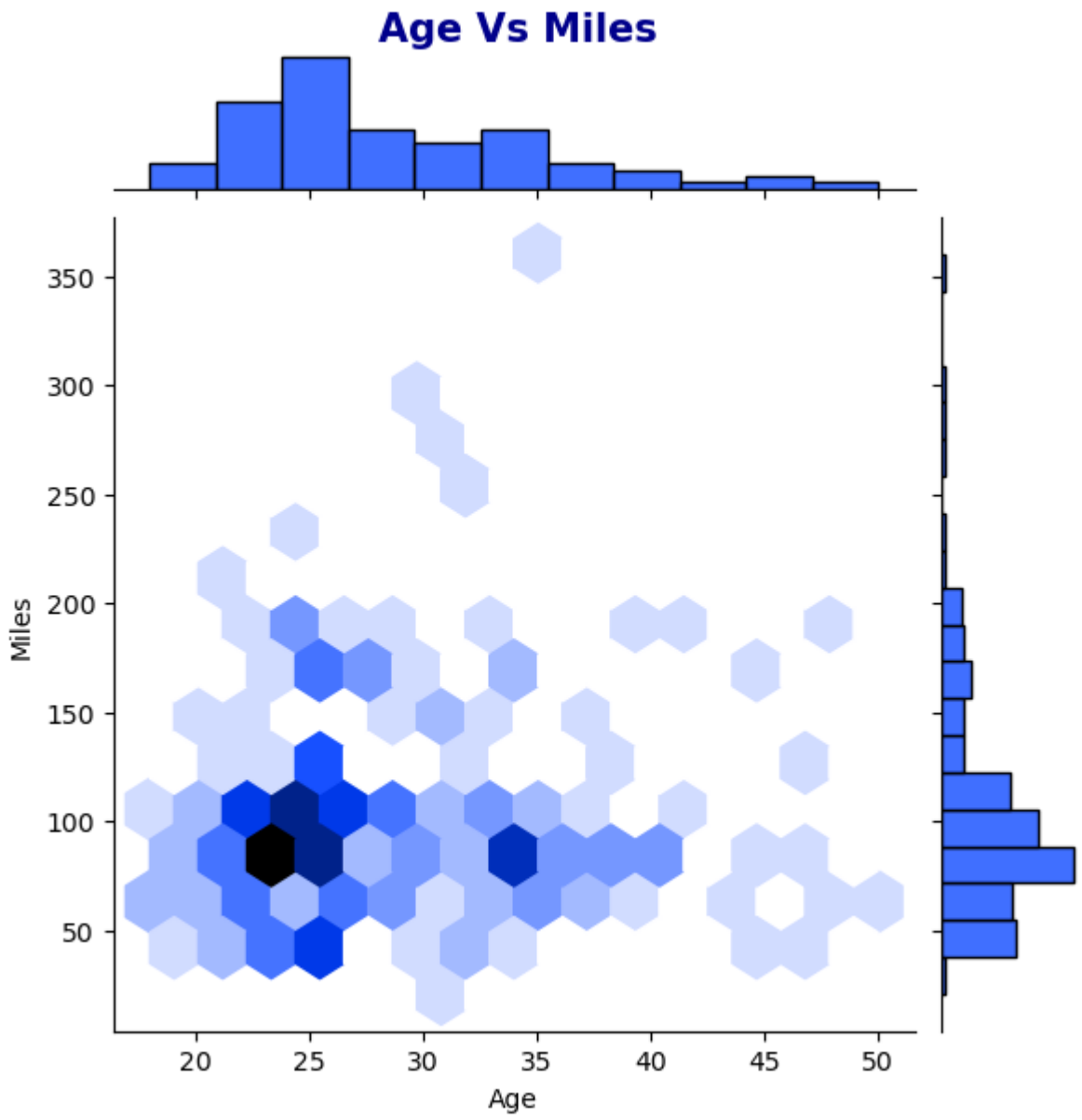
In [297…
```
plt.figure(figsize=(18,5))
sns.scatterplot(x='Miles',y='Fitness',data=af,hue='Gender',style='MaritalStatus',palette=cp4)
plt.title('Customers classification Based on Miles ran Vs Fitness',fontfamily='serif',fontweight='bold',fontsize=20)
plt.legend(loc='lower right')
plt.show()
```



In [245…
```
sns.jointplot(x="Age", y="Fitness",hue='MaritalStatus',width=12,data=af,kind="kde",palette=cp)
plt.text(25,8,"Age Vs Fitness",color='maroon',fontsize=15,fontweight='bold')
plt.show()
```

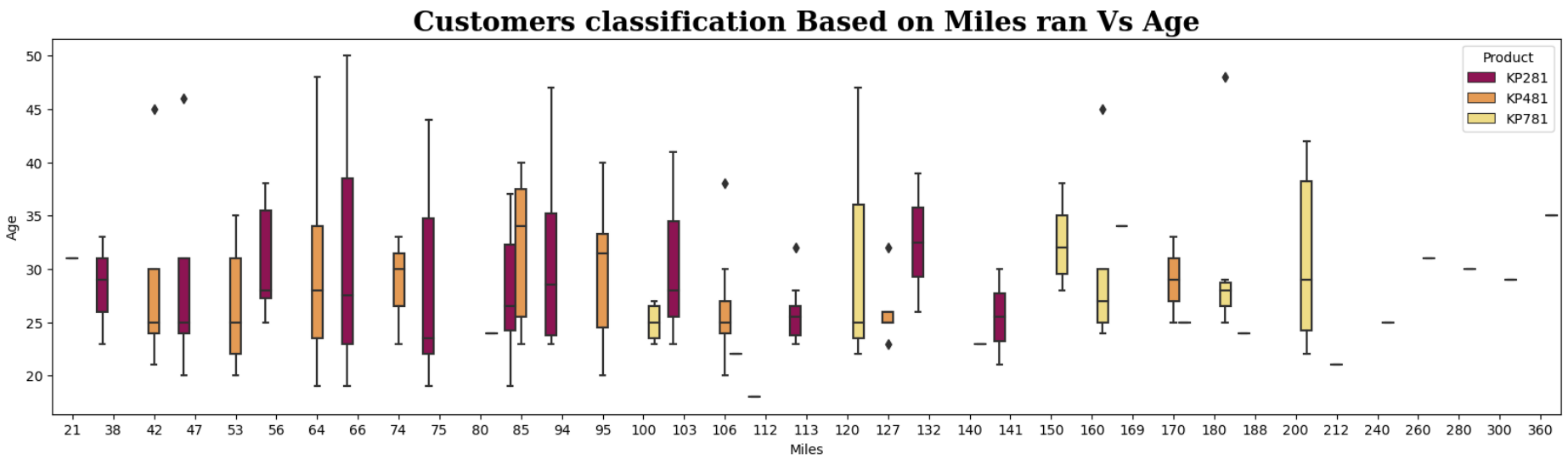


In [321…
```
sns.jointplot(x="Age", y="Miles",data=af,kind="hex",palette=cp[0])
plt.text(28,458,"Age Vs Miles",color='darkblue',fontsize=15,fontweight='bold')
plt.show()
```



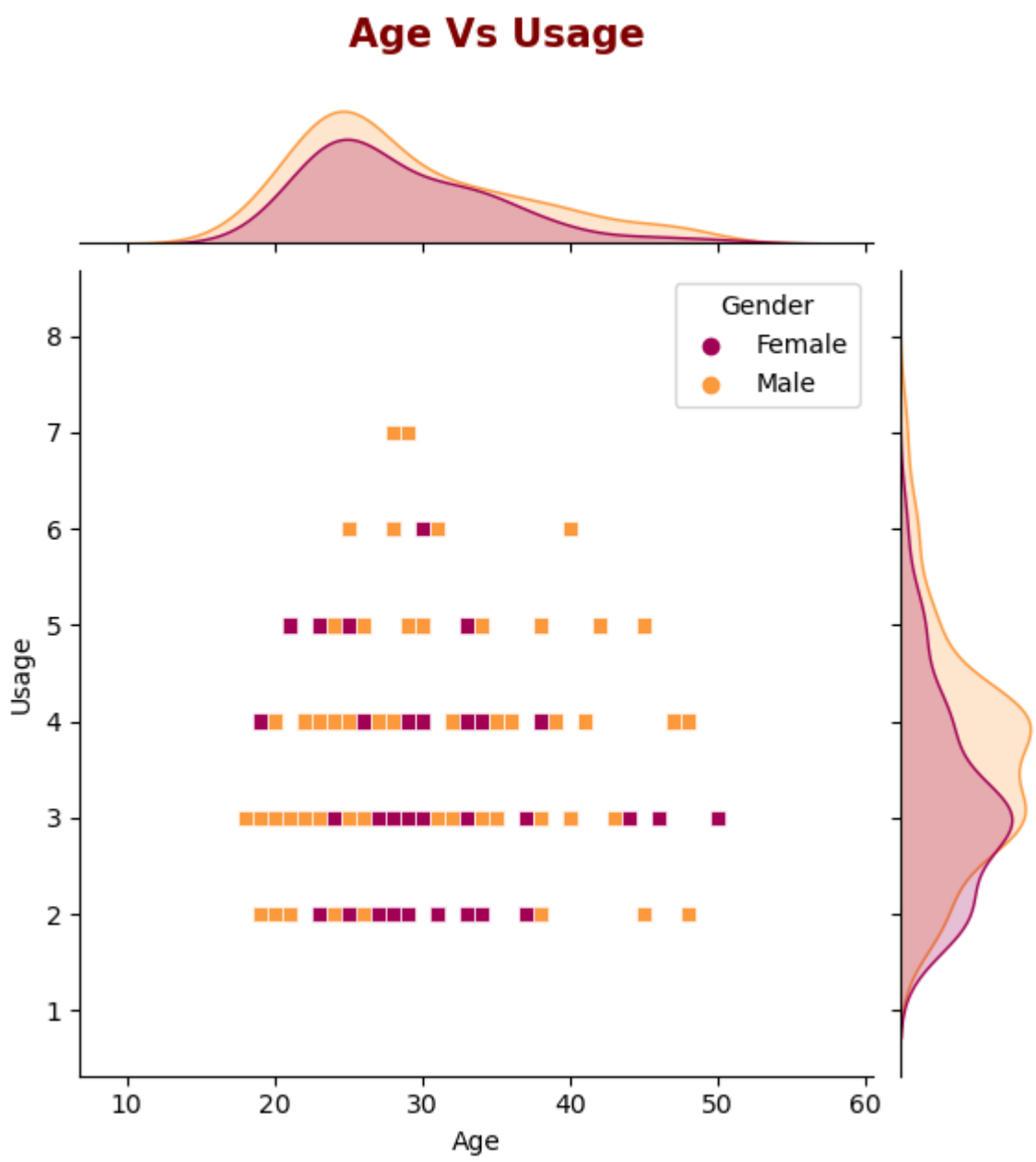
In [316…

```
# Miles with each product
plt.figure(figsize=(20,5))
sns.boxplot(af,x='Miles',y='Age',hue='Product',palette=cp)
plt.title('Customers classification Based on Miles ran Vs Age',fontfamily='serif',fontweight='bold',fontsize=20)
plt.show()
print()
```



In [317…

```
sns.jointplot(x="Age", y="Usage",hue='Gender', data=af,kind="scatter",palette=cp,marker='s')
plt.text(25,11,"Age Vs Usage",color='maroon',fontsize=15,fontweight='bold')
plt.show()
print()
```



In [318…

```
sns.jointplot(x="Age", y="Miles",hue='Fitness_comment', data=af,palette=cp,marker='d')
plt.text(10,451,"Age Vs Miles ran",color='maroon',fontsize=15,fontweight='bold')
plt.show()
print()
```

In [249…

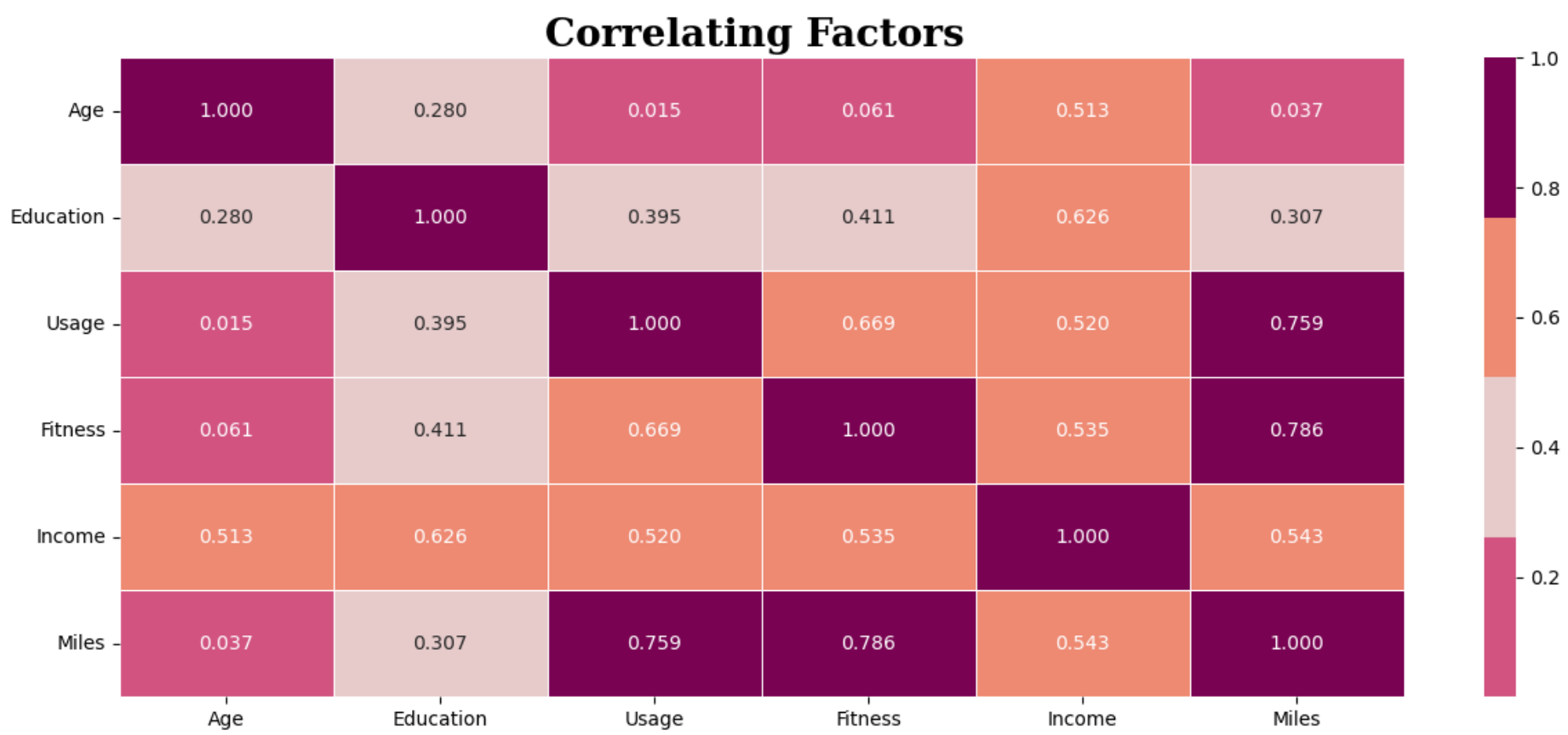
```
afcorr = af[['Age','Education','Usage','Fitness','Income','Miles']].corr()
afcorr
```

Out[249]:

	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

In [250…

```
#Correlation HeatMap
plt.figure(figsize=(15,6))
ax = sns.heatmap(afcorr,annot=True,fmt='.3f',linewidths=.5,cmap=cp1)
plt.title('Correlating Factors ',fontfamily='serif',fontweight='bold',fontsize=20)
plt.yticks(rotation=0)
plt.show()
```

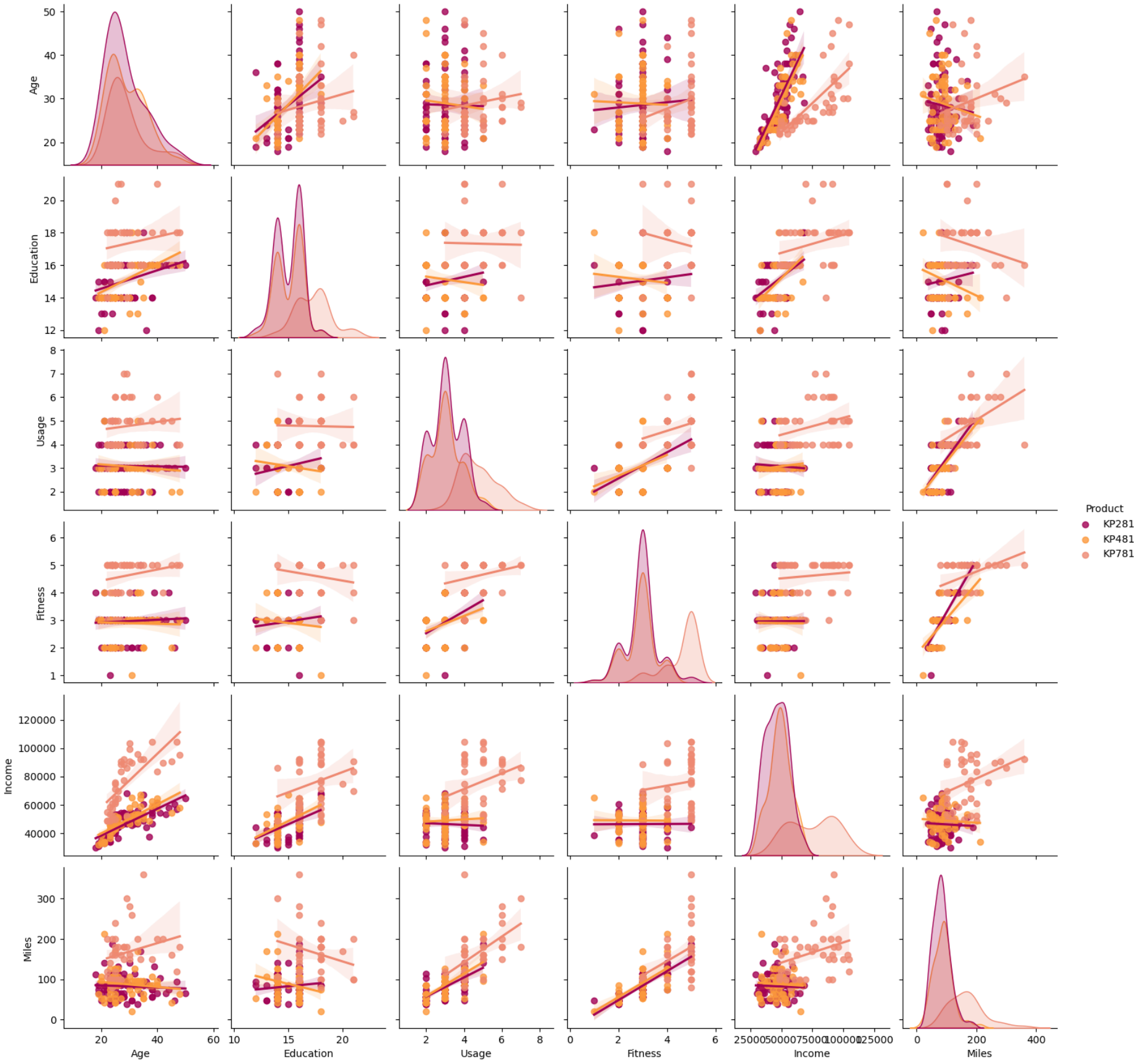


Insights

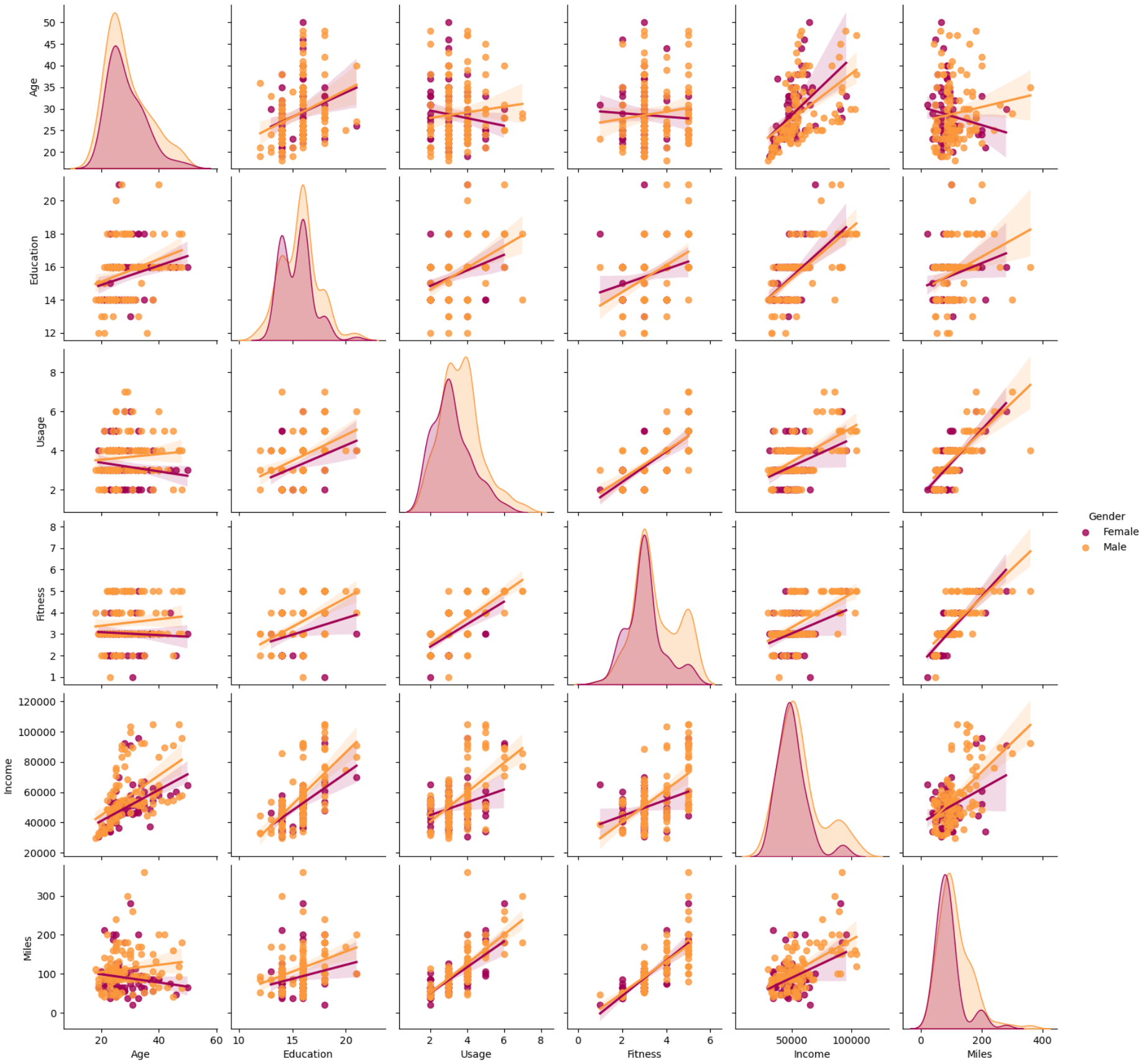
- From the pair plot we can see `Age and Income` are **positively correlated** and heatmap also suggests a **strong correlation** betwwen them
- `Eductaion` and `Income` are highly correlated as its obvious. Eductation also has significatnt correlation between `Fitness rating and Usage of the treadmill.`
- `Usage` is highly correlated with `Fitness and Miles` as more the usage more the fitness and mileage.

In [319…

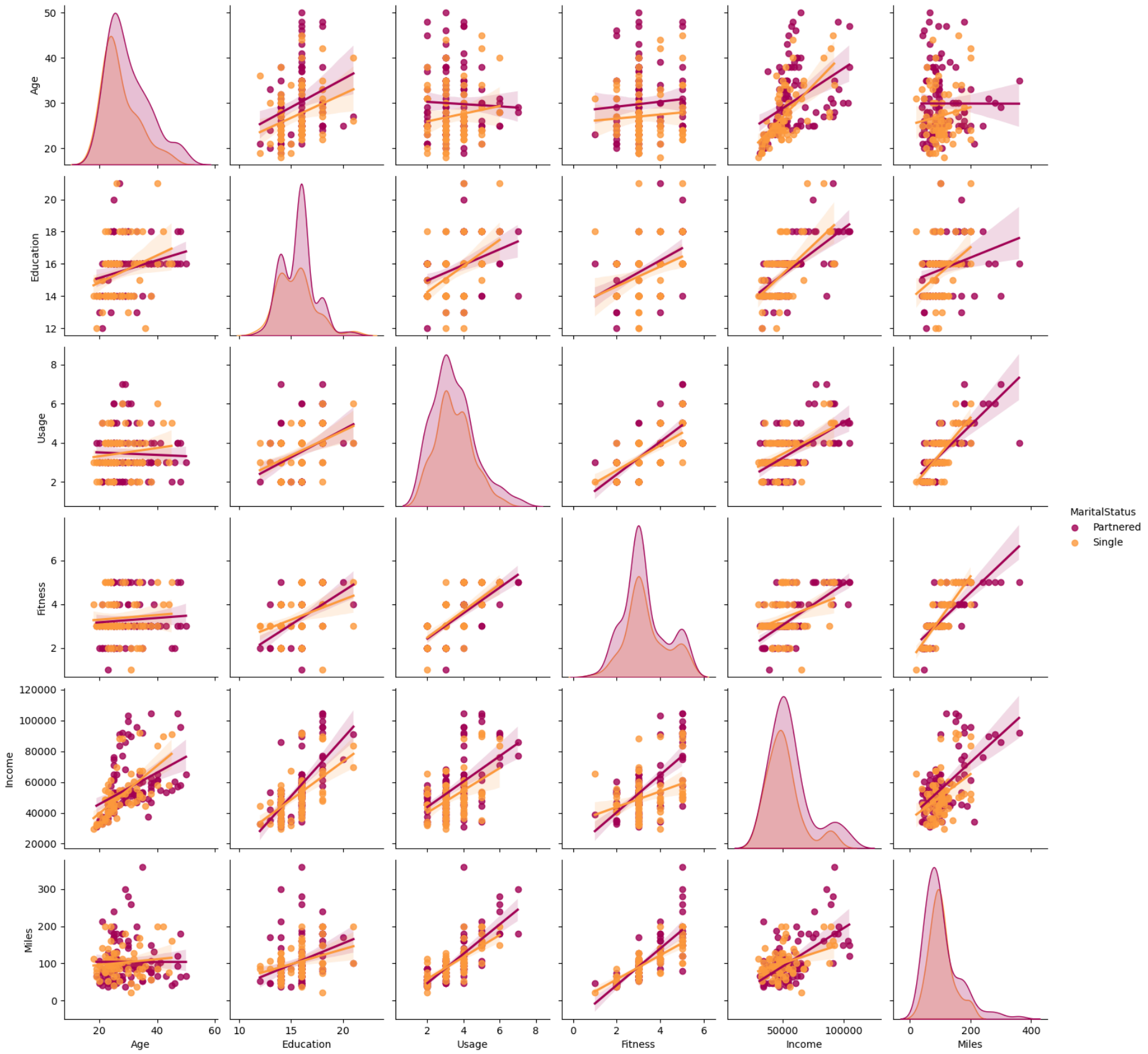
```
# Pairplot Analysis
cat_cols = ['Product','Gender','MaritalStatus']
for i in range(len(cat_cols)):
    plt.figure(figsize=(14,0.05))
    plt.axis('off')
    plt.title(f' Pairplot based on {cat_cols[i]} ',fontfamily='serif',fontweight='bold',fontsize=20,backgroundcolor='maroon',colo
    sns.pairplot(af,hue=cat_cols[i],kind='reg',palette=cp4)
    plt.show()
    print()
```



Pairplot based on Gender



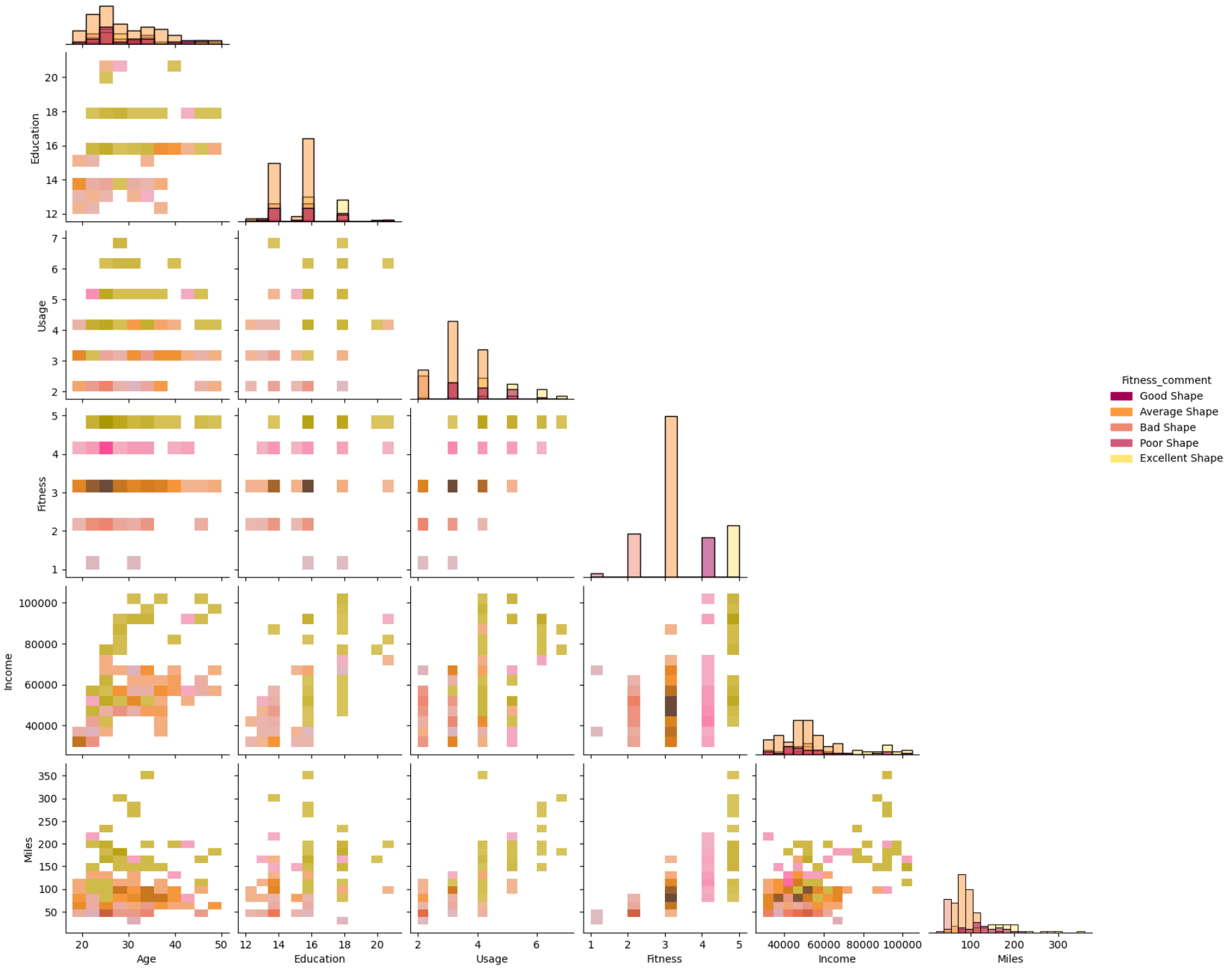
Pairplot based on MaritalStatus



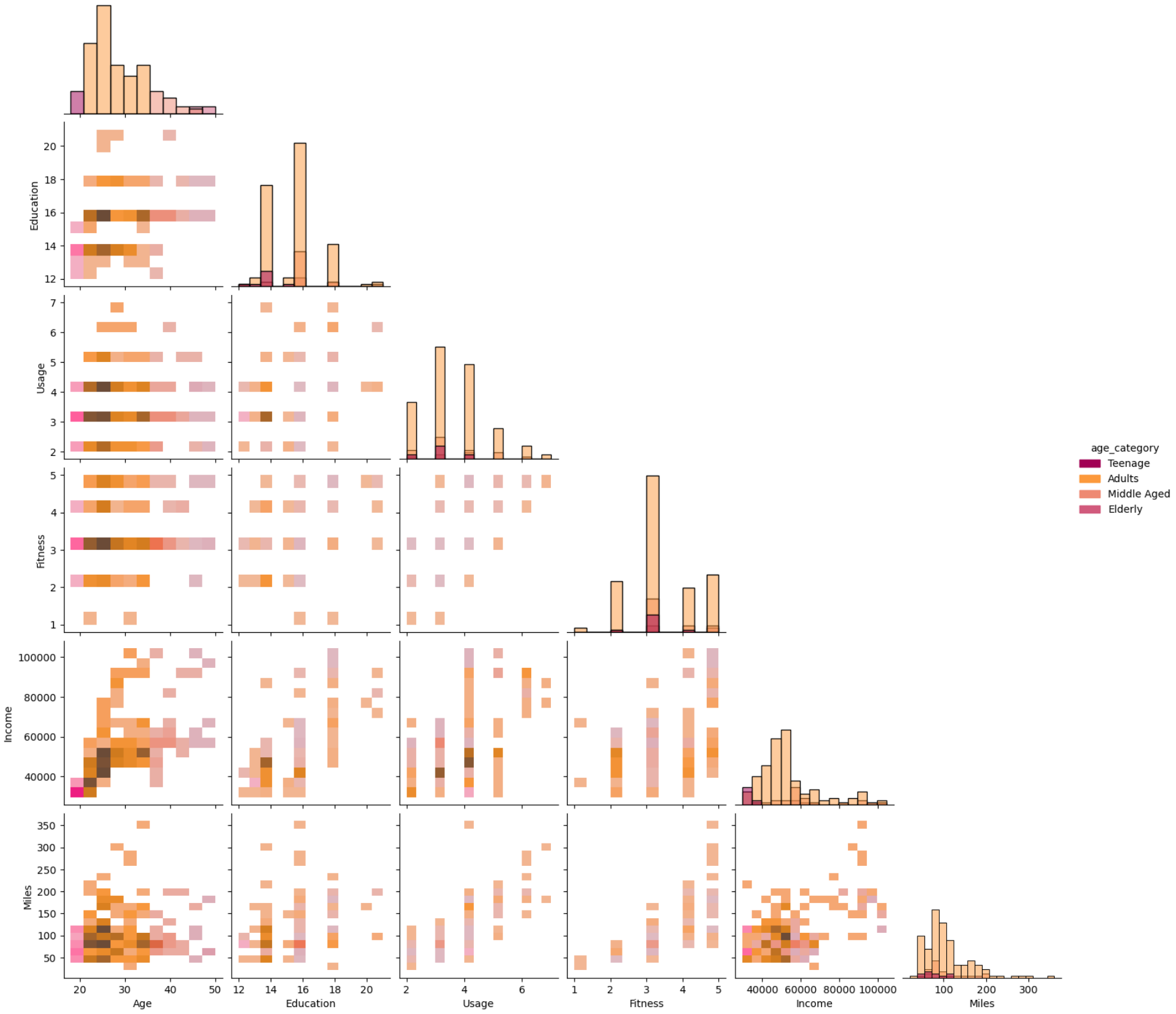
In [320…

```
# Pairplot Analysis
cat_cols = ['Fitness_comment','age_category','income_grp']
for i in range(len(cat_cols)):
    plt.figure(figsize=(14,0.05))
    plt.axis('off')
    plt.title(f' Pairplot based on {cat_cols[i]} ',fontfamily='serif',fontweight='bold',
              fontsize=20,backgroundcolor='maroon',color='w')
    sns.pairplot(af,hue=cat_cols[i],kind='hist',corner=True,palette=cp4)
    plt.show()
    print()
    print()
```

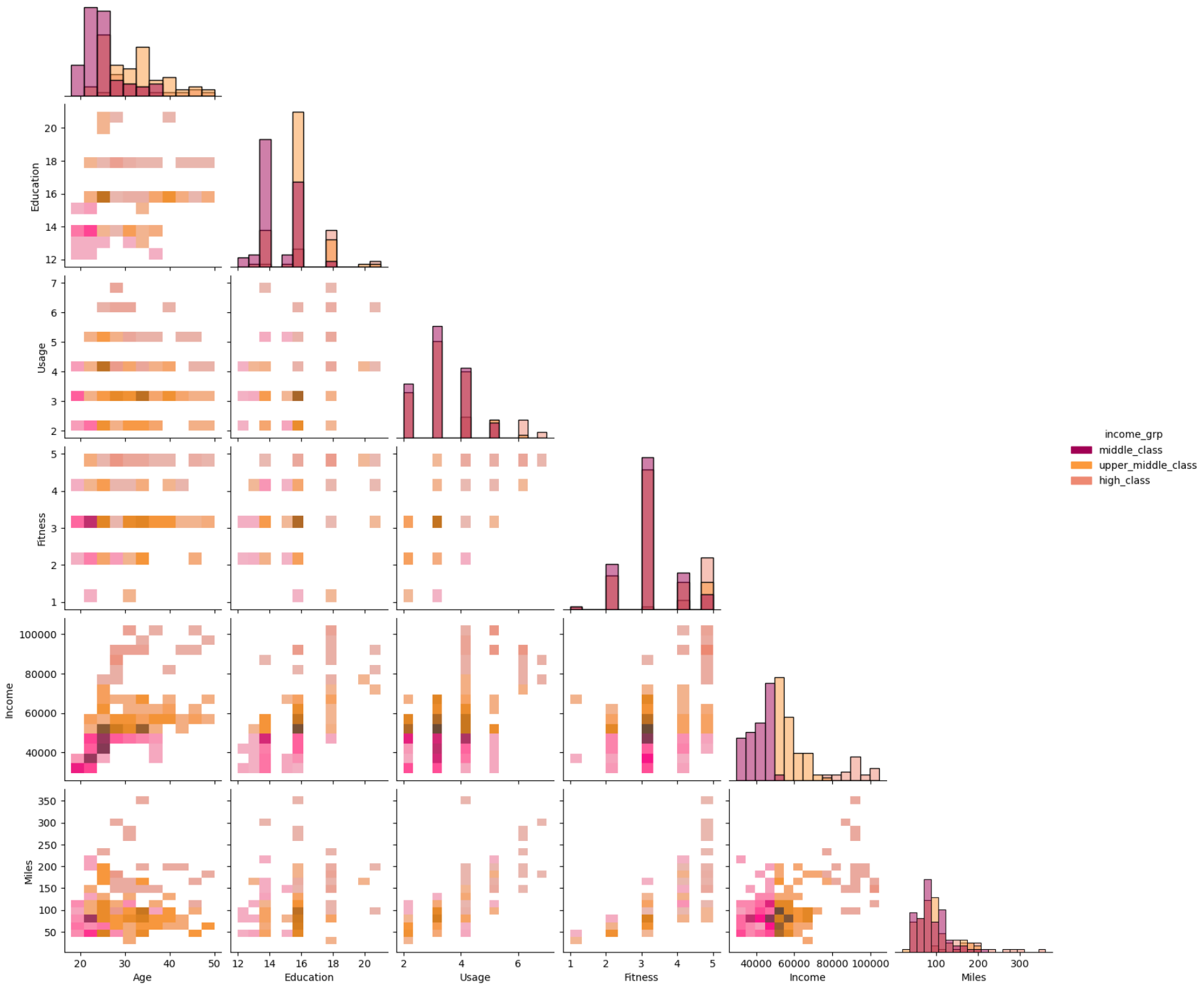
Pairplot based on Fitness_comment



Pairplot based on age_category



Pairplot based on income_grp

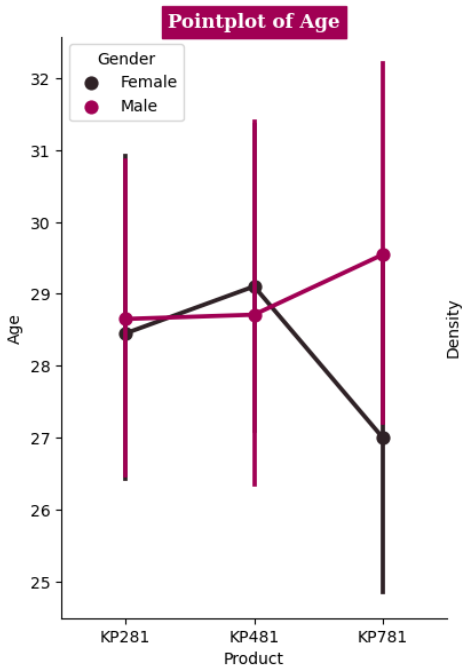


In [308…

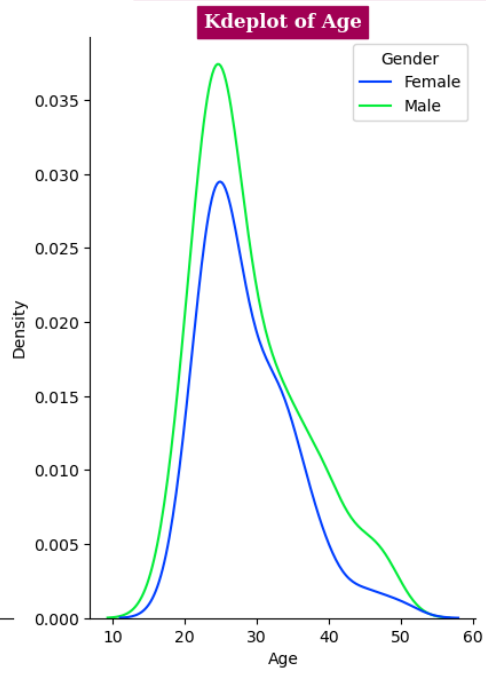
```
num_cols = ['Age','Education','Usage','Fitness','Income','Miles']


for i in range(len(num_cols)):
    plt.figure(figsize=(20,6.6))
    plt.style.use('default')
    plt.style.use('seaborn-v0_8-bright')
    plt.suptitle(f'Customers classification Based on {num_cols[i]}',fontfamily='serif',fontweight='bold',fontsize=20,
                 backgroundcolor=cp4[i],color='w')

    plt.subplot(1,4,1)
    plt.title(f'Pointplot of {num_cols[i]}',fontfamily='serif',fontweight='bold',fontsize=12,
                 loc='center',backgroundcolor=cp4[i],color='w')
    sns.pointplot(af,y=num_cols[i],x='Product',color=cp4[i],hue='Gender')

    plt.subplot(1,4,2)
    sns.kdeplot(af,x=num_cols[i],hue='Gender')
    plt.title(f'Kdeplot of {num_cols[i]}',fontfamily='serif',fontweight='bold',fontsize=12,
                 loc='center',backgroundcolor=cp4[i],color='w')
    plt.subplot(1,4,4)
    sns.kdeplot(af,x=num_cols[i],hue='MaritalStatus')
    plt.title(f'Kdeplot of {num_cols[i]}',fontfamily='serif',fontweight='bold',fontsize=12,
                 loc='center',backgroundcolor=cp4[i],color='w')

    plt.subplot(1,4,3)
    sns.stripplot(af,y=num_cols[i],x='Product',color=cp4[i],hue='MaritalStatus')
    plt.title(f'Stripplot of {num_cols[i]}',fontfamily='serif',fontweight='bold',fontsize=12,
                 loc='center',backgroundcolor=cp4[i],color='w')
    sns.despine()
plt.show()
```

Customers classification Based on Age

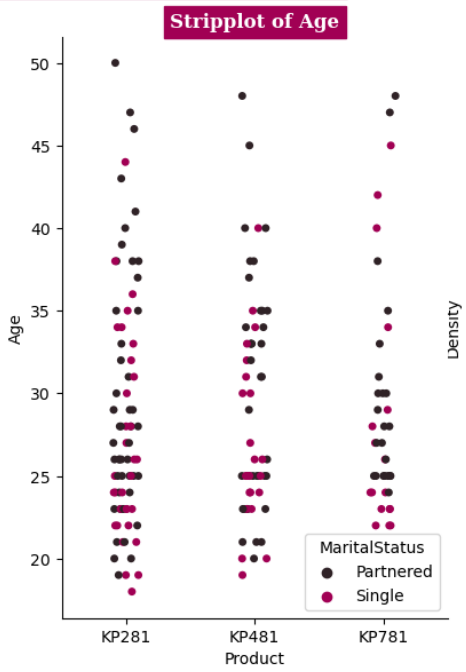
Pointplot of Age



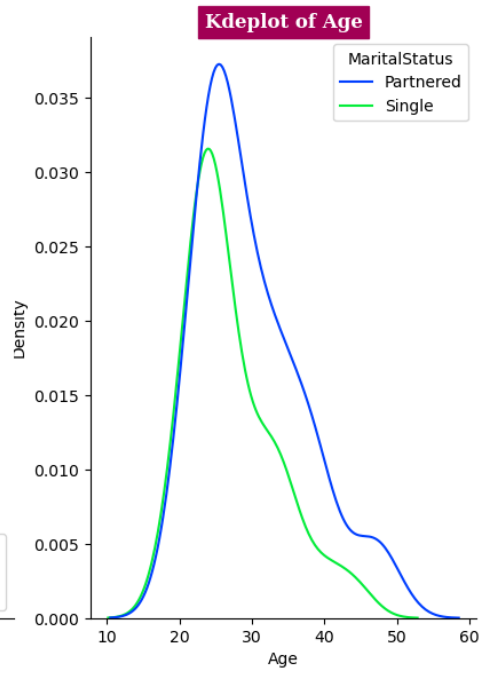
Kdeplot of Age



Stripplot of Age

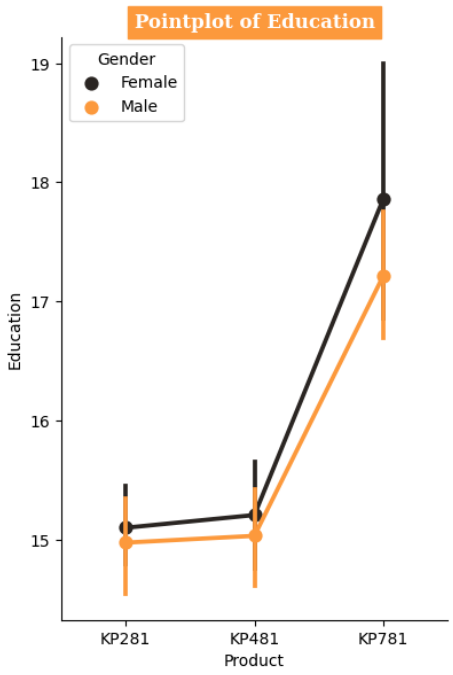


Kdeplot of Age

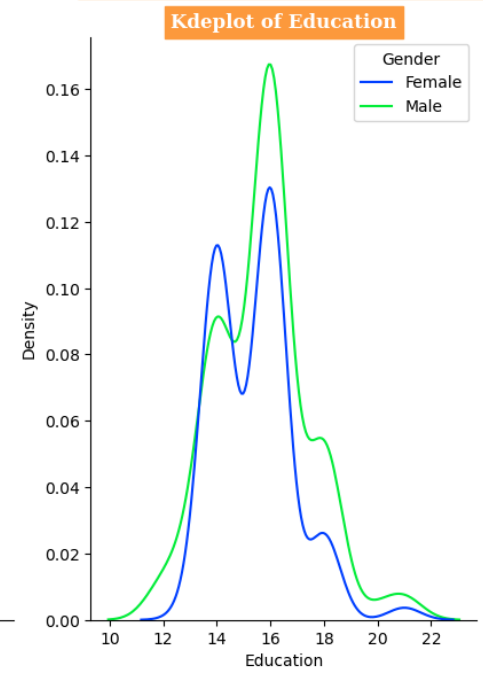


Customers classification Based on Education

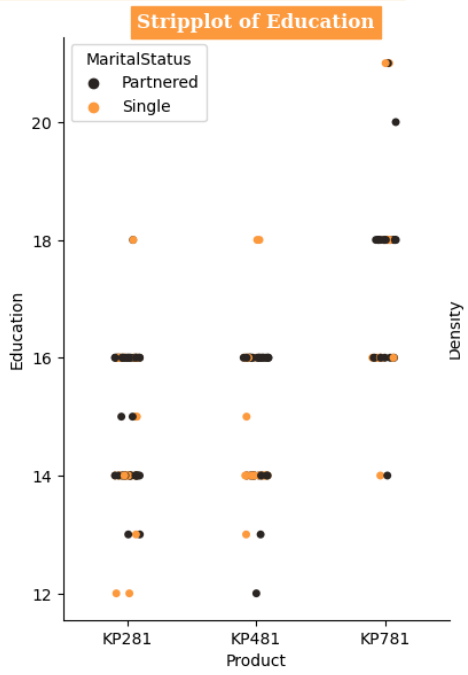
Pointplot of Education



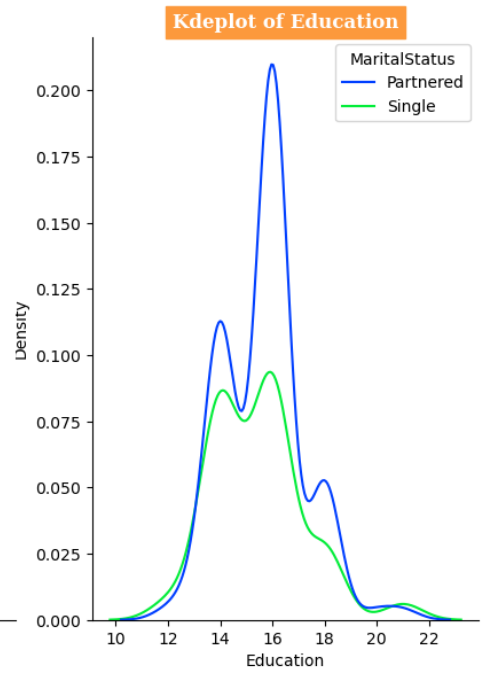
Kdeplot of Education



Stripplot of Education

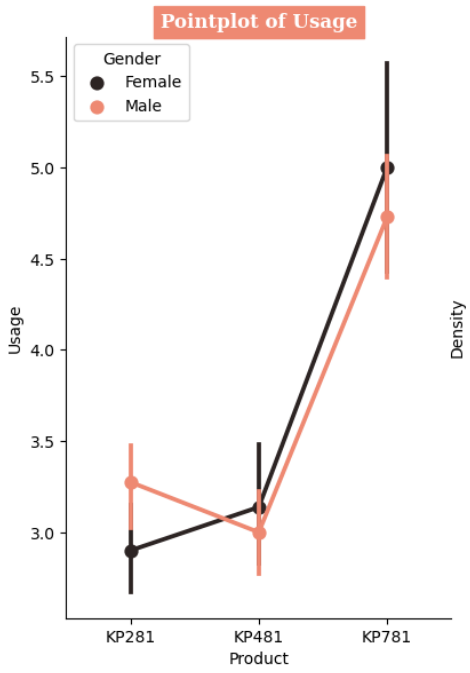


Kdeplot of Education

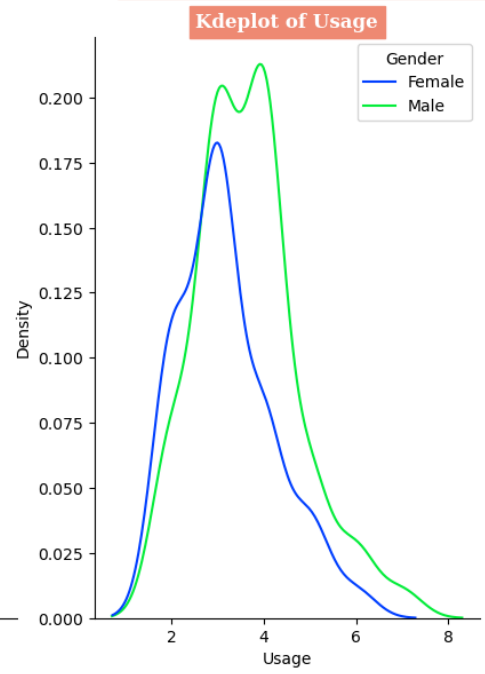


Customers classification Based on Usage

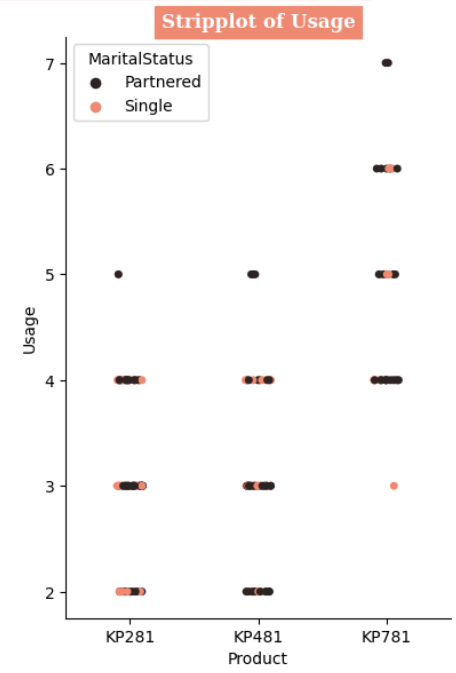
Pointplot of Usage



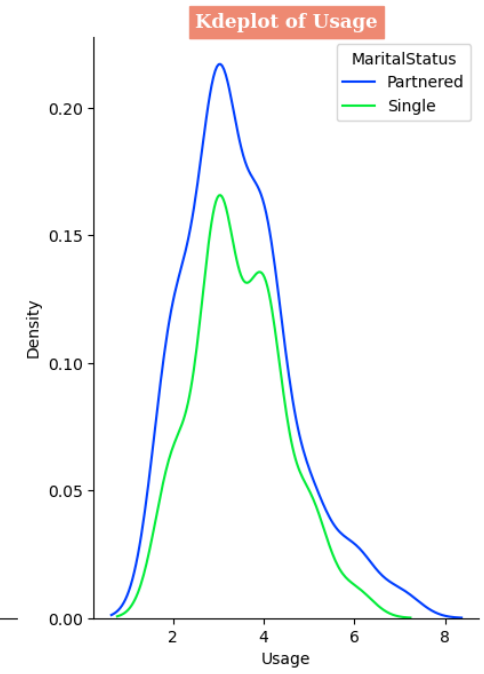
Kdeplot of Usage



Stripplot of Usage

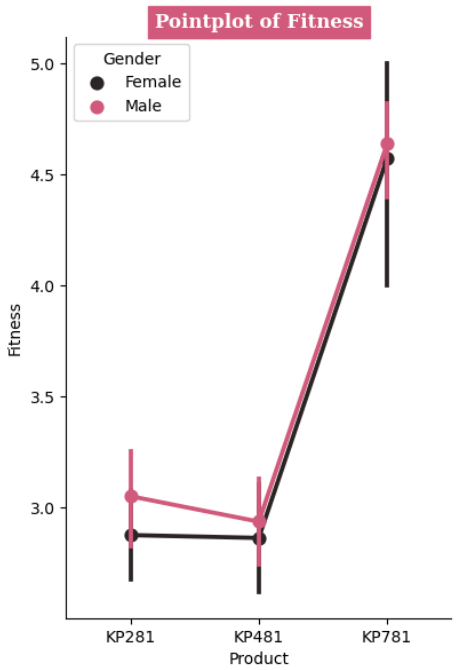


Kdeplot of Usage

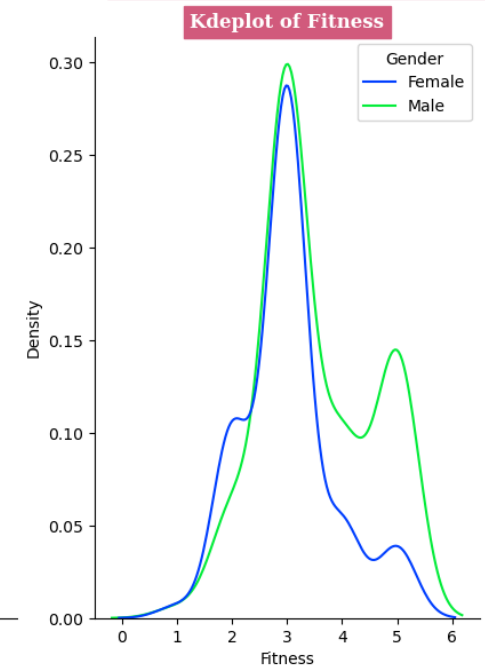


Customers classification Based on Fitness

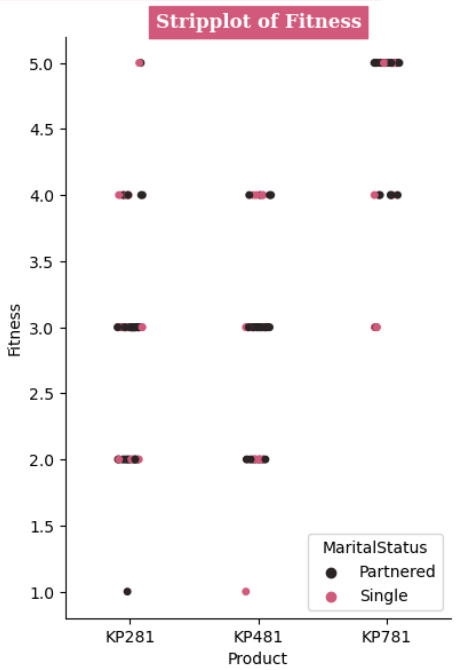
Pointplot of Fitness



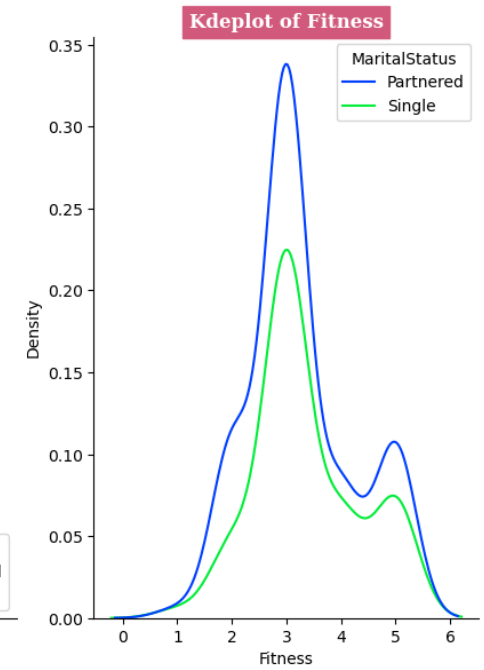
Kdeplot of Fitness

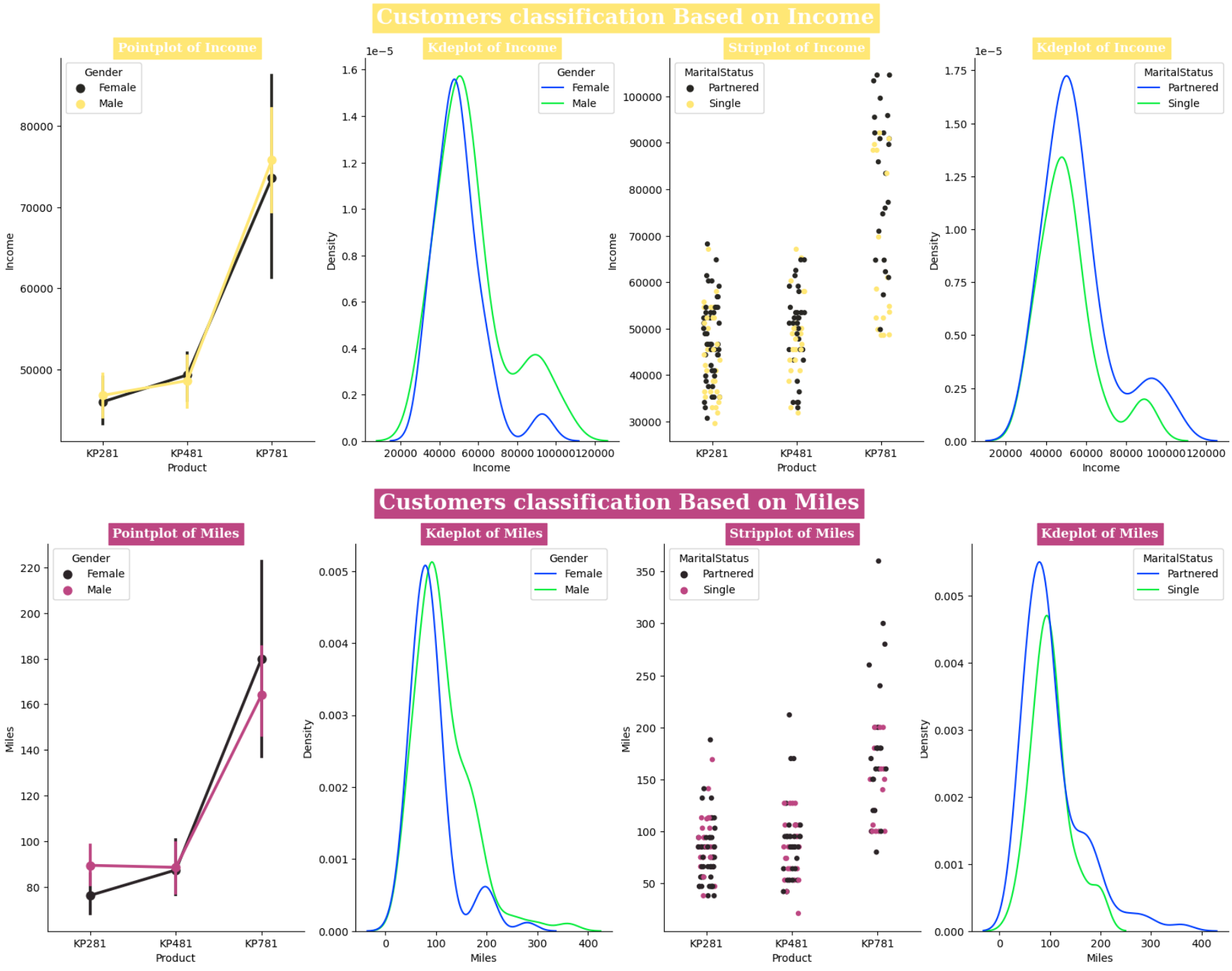


Stripplot of Fitness



Kdeplot of Fitness





In [309…
```
import sweetviz as sv
report = sv.analyze(af)
report.show_html(filepath='SWEETVIZ_REPORT.html',
            open_browser=True,
            layout='widescreen',
            scale=None)
```

```
                                             |          | [  0%]   00:00 -> (? left)
Report SWEETVIZ_REPORT.html was generated! NOTEBOOK/COLAB USERS: the web browser MAY not pop up, regardless, the report IS saved
in your notebook/colab files.
```

In [310…
```
report = sv.analyze(af)
report.show_notebook(filepath='SWEETVIZ_REPORT.html',layout='vertical', h='full',scale=0.9)
```

```
                                             |          | [  0%]   00:00 -> (? left)
```

SweetViz 2.2.1

Get updates, docs & report issues here

Created & maintained by Francois Bertrand
Graphic design by Jean-Francois Hains

DataFrame

NO COMPARISON TARGET

180	ROWS
0	DUPLICATES
23.7 kb	RAM
12	FEATURES
9	CATEGORICAL
3	NUMERICAL
0	TEXT

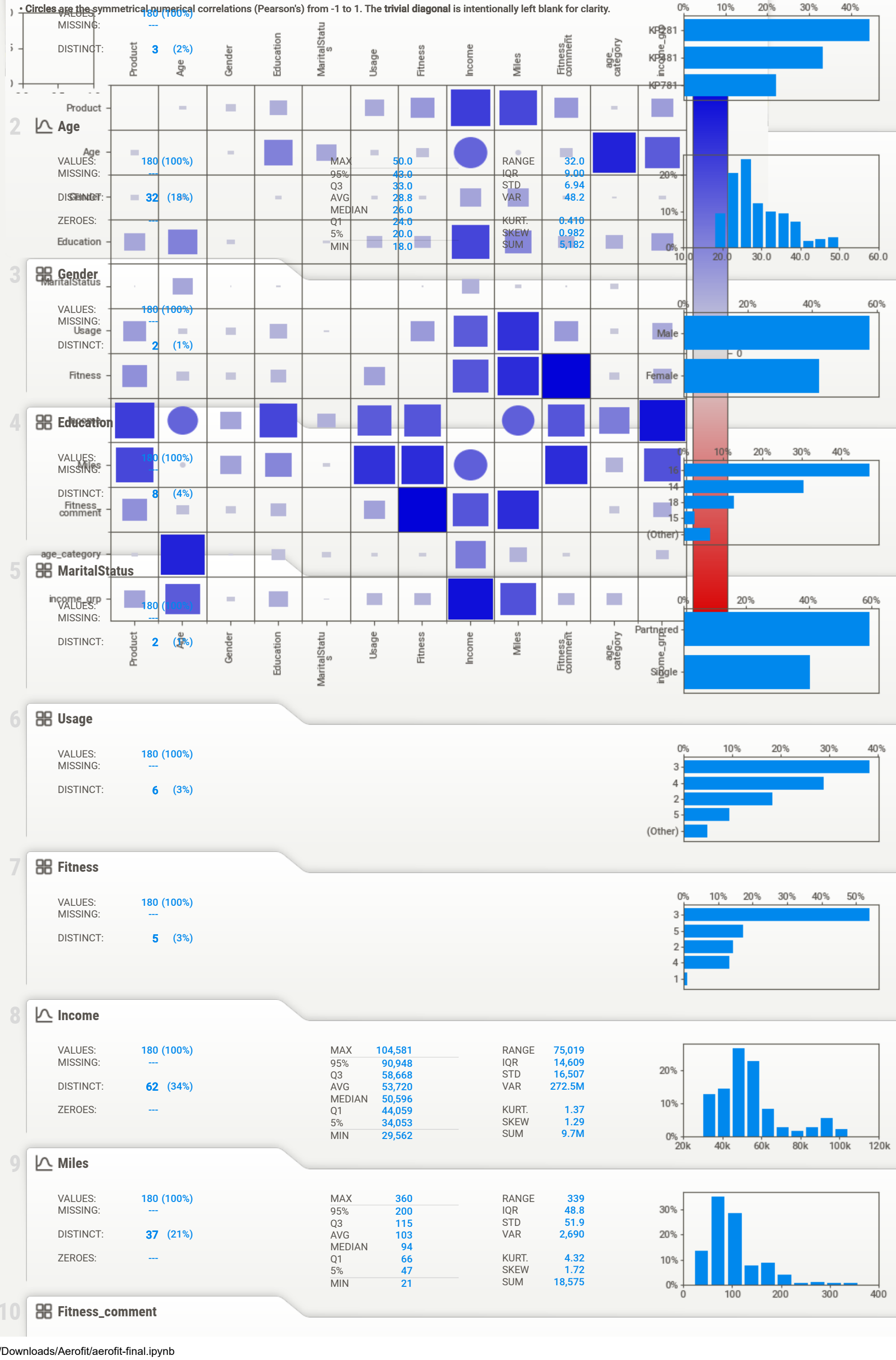
ASSOCIATIONS

Associations

[Only including dataset "DataFrame"]

■ Square categorical associations (uncertainty coefficient & correlation ratio) from 0 to 1. The uncertainty coefficient is assymmetrical, (i.e. ROW LABEL values indicate how much they PROVIDE INFORMATION to each LABEL at the TOP).

• Circles are the symmetrical numerical correlations (Pearson's) from -1 to 1. The trivial diagonal is intentionally left blank for clarity.



1 Product

VALUES:	180	(100%)
MISSING:	---	
DISTINCT:	3	(2%)

2 Age

VALUES:	180	(100%)
MISSING:	---	
DISTINCT:	32	(18%)
ZEROES:	---	

MAX	50.0	RANGE	32.0
95%	43.0	IQR	9.00
Q3	33.0	STD	6.94
AVG	28.8	VAR	48.2
MEDIAN	26.0		
Q1	24.0	KURT.	0.410
5%	20.0	SKEW	0.982
MIN	18.0	SUM	5,182

3 Gender

VALUES:	180	(100%)
MISSING:	---	
DISTINCT:	2	(1%)

4 Education

VALUES:	180	(100%)
MISSING:	---	
DISTINCT:	8	(4%)

5 MaritalStatus

VALUES:	180	(100%)
MISSING:	---	
DISTINCT:	2	(1%)

6 Usage

VALUES:	180	(100%)
MISSING:	---	
DISTINCT:	6	(3%)

7 Fitness

VALUES:	180	(100%)
MISSING:	---	
DISTINCT:	5	(3%)

8 Income

VALUES:	180	(100%)
MISSING:	---	
DISTINCT:	62	(34%)
ZEROES:	---	

MAX	104,581	RANGE	75,019
95%	90,948	IQR	14,609
Q3	58,668	STD	16,507
AVG	53,720	VAR	272.5M
MEDIAN	50,596		
Q1	44,059	KURT.	1.37
5%	34,053	SKEW	1.29
MIN	29,562	SUM	9.7M

9 Miles

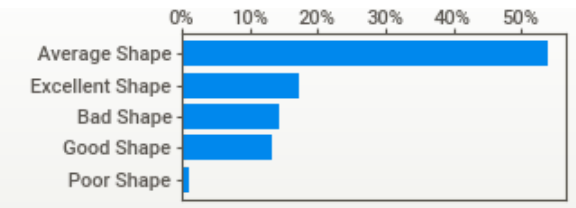
VALUES:	180	(100%)
MISSING:	---	
DISTINCT:	37	(21%)
ZEROES:	---	

MAX	360	RANGE	339
95%	200	IQR	48.8
Q3	115	STD	51.9
AVG	103	VAR	2,690
MEDIAN	94		
Q1	66	KURT.	4.32
5%	47	SKEW	1.72
MIN	21	SUM	18,575

10 Fitness_comment

VALUES: 180 (100%)
MISSING: ---

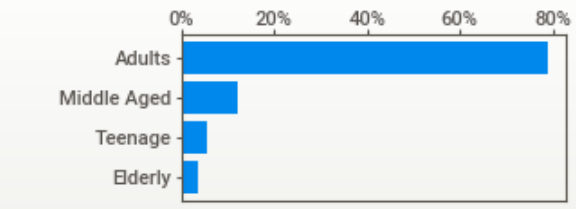
DISTINCT: 5 (3%)



11 **age_category**

VALUES: 180 (100%)
MISSING: ---

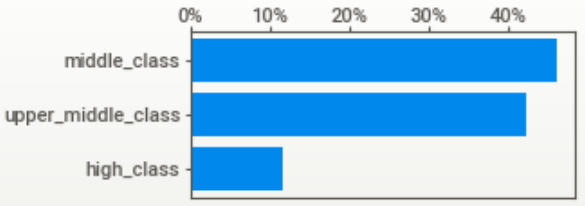
DISTINCT: 4 (2%)



12 **income_grp**

VALUES: 180 (100%)
MISSING: ---

DISTINCT: 3 (2%)



Report 'SWEETVIZ_REPORT.html' was saved to storage.

🚶‍♂️ 🚶‍♀️ 🧍‍♂️ CUSTOMER PROFILING :

`Based on all the above analysis`

```
-> Probability of purchase of KP281 = 44%
-> Probability of purchase of KP481 = 33%
-> Probability of purchase of KP781 = 22%
```

- `Customer Profile` for **`KP281`** Treadmill:
  - Most preferred by entry level
  - Age of customer mainly between 18 to 35 years with few between 35 to 50 years
  - Education level of customer 13 years and above
  - Annual Income of customer ranges from 35k USD 55k USD
  - Weekly Usage - 3 to 4 times
  - Fitness Scale - 2 to 4
  - Weekly Running Mileage - 50 to 100 miles
  - Mostly Single Female and Partnered Male prefer this product

---

- `Customer Profile` for **`KP481`** Treadmill:
  - This is an Intermediate level Product.
  - Age of customer mainly between 18 to 35 years with few between 35 to 50 years
  - Education level of customer 13 years and above

- Annual Income of customer between 40k-80k USD
- Weekly Usage - 2 to 4 times
- Fitness Scale - 2 to 4
- Weekly Running Mileage - 50 to 200 miles
- Probability of Female customer buying KP481 is significantly higher than male.

- `Customer Profile` for **`KP781`** Treadmill:
  - Due to the High Price & being the advanced type, customer prefers less of this product.
  - Age of customer between 18 to 35 years
  - Education level of customer 15 years and above
  - Annual Income of customer 80k USD and above
  - Weekly Usage - 4 to 7 times
  - Fitness Scale - 3 to 5
  - Weekly Running Mileage is >100 miles and above
  - Partnered Female bought KP781 treadmill compared to Partnered Male.
  - This product is preferred by the customer where the correlation between Education and Income is High. </kbd>

Recommendations :

- **Gender-Targeted Marketing for KP784:**
  - **Recommendation:** Implement targeted strategies to address the significant sales disparity in KP781 between genders.
  - **Actionable Insight:** Launch exclusive promotions and trials designed specifically for female customers to increase their engagement and contribution to sales.

- **Affordable Pricing and Flexible Payment Plans for KP281 and KP481:**
  - **Recommendation:** Make the Prices of KP281 and KP481 more affordably by providing flexible payment plans.
  - **Actionable Insight:** Conduct a pricing analysis considering the target customers' age, education level, and income to determine an optimal and attractive price point.

- **User-Friendly App Integration for Enhanced Engagement:**
  - **Recommendation:** Develop a user-friendly app to enhance the overall treadmill experience and keep users engaged.
  - **Actionable Insight:** Include features such as mileage tracking, real-time feedback, and personalized workout recommendations in the app for added value.

- **Strategic Product Promotion:**
  - **Recommendation:** Leverage the characteristics of KP781 customers for targeted promotions.
  - **Actionable Insight:** Analyze the preferences of KP781 customers (males, high income, premium preference) and tailor marketing messages to highlight features that align with these preferences.

- **Focus on KP481 with No-Cost EMI Support:**
  - **Recommendation:** Promote KP481 with additional incentives such as no-cost EMI support.
  - **Actionable Insight:** Position KP481 as an attractive option, especially for customers looking for flexible payment options, and communicate the benefits of the no-cost EMI plan.

- **Targeted Marketing through E-commerce and Social Media:**
  - **Recommendation:** Implement personalized ads on E-commerce and Social Media platforms.
  - **Actionable Insight:** Utilize customer data to create targeted advertisements that resonate with specific characteristics and preferences, maximizing the impact of marketing efforts.

- **Encouraging Female Fitness:**
  - **Recommendation:** Launch a marketing campaign to encourage women to embrace fitness with Aerofit treadmills.
  - **Actionable Insight:** Create empowering and inclusive marketing messages that highlight the benefits of using Aerofit treadmills for female customers.

- **Budget Treadmill Image for KP281 & KP481:**
  - **Recommendation:** Position KP281 and KP481 as budget-friendly options.
  - **Actionable Insight:** Use pricing and payment plan flexibility to create a perception of affordability, appealing to customers with varying budgets.

- **Professional and Athlete Endorsement for KP781:**
  - **Recommendation:** Market KP781 as a premium product suitable for professionals and athletes.
  - **Actionable Insight:** Collaborate with influencers and international athletes for endorsements to enhance credibility and reach a broader audience.

- **Market Expansion and Health Considerations:**
  - **Recommendation:** Conduct research on expanding the market beyond 50 years of age.
  - **Actionable Insight:** Collect data on health considerations and preferences of the older demographic to tailor marketing strategies and product features accordingly.

- **Customer Support and Upgrade Recommendations:**
  - **Recommendation:** Provide robust customer support services.
  - **Actionable Insight:** Implement a customer support system that includes guidance on upgrading to higher-level treadmill models after consistent usage, enhancing customer satisfaction and brand loyalty.

- **Tailored Recommendations for KP781 to Females:**
  - **Recommendation:** Highlight KP781's advanced features for female customers engaging in extensive exercise routines.
  - **Actionable Insight:** Develop marketing materials that specifically showcase the benefits and user-friendly aspects of KP781 for female users.

- **Target Age Group Above 40 for KP781:**
  - **Recommendation:** Specifically target and recommend KP781 to the age group above 40 years.
  - **Actionable Insight:** Create targeted advertising campaigns focusing on health benefits and features that align with the preferences of customers aged 40 and above.
