In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:
```
aerofit=pd.read_csv("aerofit_treadmill.csv")
```

overview

In [4]:
```
aerofit.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 [37]:
```
aerofit.shape
```

Out[37]:
```
(180, 9)
```

In [11]:
```
aerofit.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 [ ]:
```

```

Value_Counts/% values

In [40]:
```
aerofit["Product"].value_counts(normalize=True)*100
```

Out[40]:
```
KP281    44.444444
KP481    33.333333
KP781    22.222222
Name: Product, dtype: float64
```

In [42]:
```
def func(data):
    if data<=20:
        return "0-20"
    elif data>20 and data<=30:
        return "20-30"
    elif data>30 and data<=40:
        return "30-40"
    elif data >40 and data<=50:
        return "40-50"
    else:
        return "50+"
aerofit["Age"].apply(func).value_counts(normalize=True)*100
```

Out[42]:
```
20-30    61.111111
30-40    26.666667
40-50     6.666667
0-20      5.555556
Name: Age, dtype: float64
```

In [43]:
```
aerofit["Gender"].value_counts(normalize=True)*100
```

Out[43]:
```
Male      57.777778
Female    42.222222
Name: Gender, dtype: float64
```

In [44]:
```
aerofit["Education"].value_counts(normalize=True)*100
```

Out[44]:
```
16    47.222222
14    30.555556
18    12.777778
15     2.777778
13     2.777778
12     1.666667
21     1.666667
20     0.555556
Name: Education, dtype: float64
```

In [46]:
```
aerofit["MaritalStatus"].value_counts(normalize=True)*100
```

Out[46]:
```
Partnered    59.444444
Single       40.555556
Name: MaritalStatus, dtype: float64
```

In [47]:
```
aerofit["Usage"].value_counts(normalize=True)*100
```

Out[47]:
```
3    38.333333
4    28.888889
2    18.333333
5     9.444444
6     3.888889
7     1.111111
Name: Usage, dtype: float64
```

In [48]:
```
aerofit["Fitness"].value_counts(normalize=True)*100
```

Out[48]:
```
3    53.888889
5    17.222222
2    14.444444
4    13.333333
1     1.111111
Name: Fitness, dtype: float64
```

In [51]:
```
def func(data):
    if data<=25000:
        return "0-25k"
    elif data>25000 and data<=50000:
        return "25k-50k"
    elif data>50000 and data<=100000:
        return "50k-100k"
    elif data>100000 and data<=150000:
        return "100k-150k"
    else:
        return "150k+"

aerofit["Income"].apply(func).value_counts(normalize=True)*100
```

Out[51]:
```
50k-100k     52.222222
25k-50k      46.111111
100k-150k     1.666667
Name: Income, dtype: float64
```

In [50]:
```
def func(data):
    if data<=50:
        return "0-50"
    elif data>50 and data<=100:
        return "50-100"
    elif data>100 and data<=150:
        return "100-150"
    elif data>150 and data<=200:
        return "150-200"
    elif data>200 and data<=250:
        return "200-250"
    elif data>250 and data<=300:
        return "250-300"
    else:
        return "300+"

aerofit["Miles"].apply(func).value_counts(normalize=True)*100
```

Out[50]:
```
50-100     53.888889
100-150    21.111111
150-200    12.222222
0-50        9.444444
250-300     1.666667
200-250     1.111111
300+        0.555556
Name: Miles, dtype: float64
```

Unique_Values

In [7]:
```
aerofit["Product"].nunique()
```

Out[7]:
```
3
```

In [8]:
```
aerofit["Product"].unique()
```

Out[8]:
```
array(['KP281', 'KP481', 'KP781'], dtype=object)
```

In [9]:
```
aerofit["Age"].min()
```

Out[9]:
```
18
```

In [10]:
```
aerofit["Age"].max()
```

Out[10]:
```
50
```

In [14]:
```
aerofit["Age"].sort_values(ascending=True).unique()
```

Out[14]:
```
array([18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 50],
      dtype=int64)
```

In [15]:
```
aerofit["Gender"].unique()
```

Out[15]:
```
array(['Male', 'Female'], dtype=object)
```

In [17]:
```
aerofit["Education"].sort_values(ascending=True).unique()
```

Out[17]:
```
array([12, 13, 14, 15, 16, 18, 20, 21], dtype=int64)
```

In [18]:
```
aerofit["MaritalStatus"].unique()
```

Out[18]:
```
array(['Single', 'Partnered'], dtype=object)
```

In [21]:
```
aerofit["Usage"].sort_values(ascending=True).unique()
```

Out[21]:
```
array([2, 3, 4, 5, 6, 7], dtype=int64)
```

In [22]:
```
aerofit["Fitness"].sort_values(ascending=True).unique()
```

Out[22]:
```
array([1, 2, 3, 4, 5], dtype=int64)
```

In [23]:
```
aerofit["Income"].sort_values(ascending=True).unique()
```

Out[23]:
```
array([ 29562,  30699,  31836,  32973,  34110,  35247,  36384,  37521,
        38658,  39795,  40932,  42069,  43206,  44343,  45480,  46617,
        47754,  48556,  48658,  48891,  49801,  50028,  51165,  52290,
        52291,  52302,  53439,  53536,  54576,  54781,  55713,  56850,
        57271,  57987,  58516,  59124,  60261,  61006,  61398,  62251,
        62535,  64741,  64809,  65220,  67083,  68220,  69721,  70966,
        74701,  75946,  77191,  83416,  85906,  88396,  89641,  90886,
        92131,  95508,  95866,  99601, 103336, 104581], dtype=int64)
```

In [24]:
```
aerofit["Miles"].sort_values(ascending=True).unique()
```

Out[24]:
```
array([ 21,  38,  42,  47,  53,  56,  64,  66,  74,  75,  80,  85,  94,
        95, 100, 103, 106, 112, 113, 120, 127, 132, 140, 141, 150, 160,
       169, 170, 180, 188, 200, 212, 240, 260, 280, 300, 360], dtype=int64)
```

Boxplot

In [27]:
```
aerofit.describe().T
```

Out[27]:

	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

In [28]:
```
aerofit.isna().sum()
```

Out[28]:
```
Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
Usage            0
Fitness          0
Income           0
Miles            0
dtype: int64
```

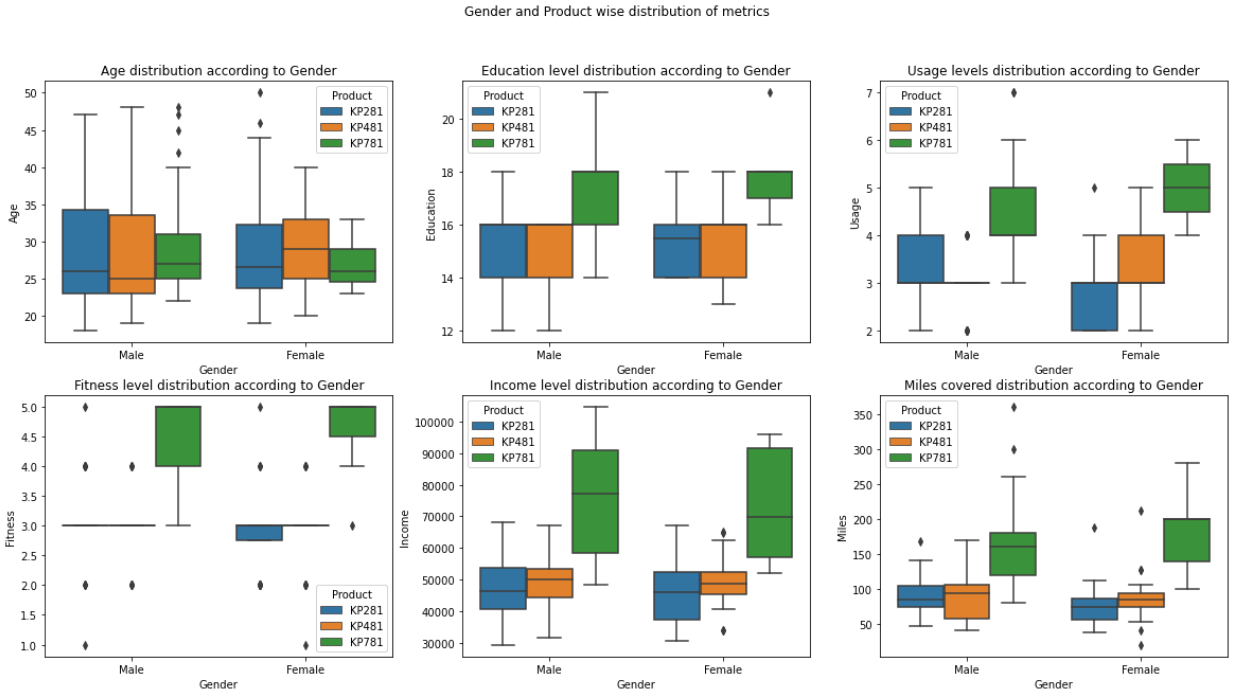
In [29]:
```
aerofit.head()
```

Out[29]:

	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 [36]:
```
plt.figure(figsize=(20,10))
plt.subplot(2,3, 1)

sns.boxplot(data=aerofit,x="Gender",y="Age",hue="Product")
plt.title("Age distribution according to Gender")

plt.subplot(2,3, 2)
sns.boxplot(data=aerofit,x="Gender",y="Education",hue="Product")
plt.title("Education level distribution according to Gender")

plt.subplot(2,3, 3)
sns.boxplot(data=aerofit,x="Gender",y="Usage",hue="Product")
plt.title("Usage levels distribution according to Gender")
```

```
plt.subplot(2,3, 4)
sns.boxplot(data=aerofit,x="Gender",y="Fitness",hue="Product")
plt.title("Fitness level distribution according to Gender")

plt.subplot(2,3, 5)
sns.boxplot(data=aerofit,x="Gender",y="Income",hue="Product")
plt.title("Income level distribution according to Gender")

plt.subplot(2,3, 6)
sns.boxplot(data=aerofit,x="Gender",y="Miles",hue="Product")
plt.title("Miles covered distribution according to Gender")

plt.suptitle("Gender and Product wise distribution of metrics")
plt.show()
```

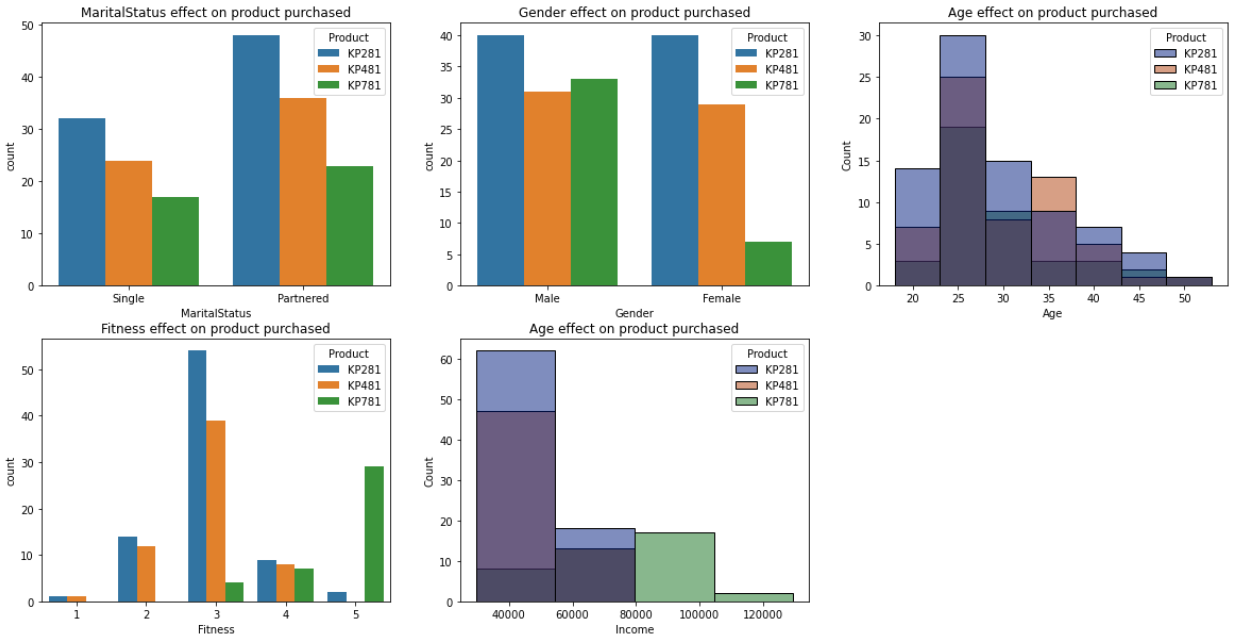


Checking on effects of metrics on product purchased

In [62]:
```
plt.figure(figsize=(20,10))
plt.subplot(2,3, 1)

sns.countplot(data=aerofit,x="MaritalStatus",hue="Product")
plt.title("MaritalStatus effect on product purchased")

plt.subplot(2,3, 2)
sns.countplot(data=aerofit,x="Gender",hue="Product")
plt.title("Gender effect on product purchased")

plt.subplot(2,3, 3)
sns.histplot(data=aerofit,x="Age",binwidth=5,hue="Product",palette="dark")
plt.title("Age effect on product purchased")

plt.subplot(2,3, 4)
```

```
sns.countplot(data=aerofit,x="Fitness",hue="Product")
plt.title("Fitness effect on product purchased")


plt.subplot(2,3, 5)
sns.histplot(data=aerofit,x="Income",binwidth=25000,hue="Product",palette="dark")
plt.title("Age effect on product purchased")
```

Out[62]:
```
Text(0.5, 1.0, 'Age effect on product purchased')
```



Crosstabs

In [83]:
```
#Total probability of products
pd.crosstab(aerofit["Product"],aerofit["Gender"],normalize=True)*100
```

Out[83]:

Gender	Female	Male
Product		
KP281	22.222222	22.222222
KP481	16.111111	17.222222
KP781	3.888889	18.333333

In [82]:
```
#total probability of product given the gender(M/F)
pd.crosstab(aerofit["Product"],aerofit["Gender"],normalize='columns')*100
```

Out[82]:

Gender	Female	Male
Product		
KP281	52.631579	38.461538
KP481	38.157895	29.807692
KP781	9.210526	31.730769

In [81]:
```
#total probability for each product for each gender
pd.crosstab(aerofit["Product"],aerofit["Gender"],normalize='index')*100
```

Out[81]:

Gender	Female	Male
Product		
KP281	50.000000	50.000000
KP481	48.333333	51.666667
KP781	17.500000	82.500000

In [80]:
```
#total probability
pd.crosstab(aerofit["Product"],aerofit["MaritalStatus"],normalize=True)*100
```

Out[80]:

MaritalStatus	Partnered	Single
Product		
KP281	26.666667	17.777778
KP481	20.000000	13.333333
KP781	12.777778	9.444444

In [79]:
```
#total probability for each product given marital status(married or single)
pd.crosstab(aerofit["Product"],aerofit["MaritalStatus"],normalize='columns')*100
```

Out[79]:

MaritalStatus	Partnered	Single
Product		
KP281	44.859813	43.835616
KP481	33.644860	32.876712
KP781	21.495327	23.287671

In [78]:
```
#total probability for each product for each marital status
pd.crosstab(aerofit["Product"],aerofit["MaritalStatus"],normalize='index')*100
```

Out[78]:

MaritalStatus	Partnered	Single
Product		
KP281	60.0	40.0
KP481	60.0	40.0
KP781	57.5	42.5

In [99]:
```
#Total probability with respect to Fitness and Product
pd.crosstab(aerofit["Product"],aerofit["Fitness"],normalize=True)*100
```

Out[99]:

Product	Fitness 1	2	3	4	5
KP281	0.555556	7.777778	30.000000	5.000000	1.111111
KP481	0.555556	6.666667	21.666667	4.444444	0.000000
KP781	0.000000	0.000000	2.222222	3.888889	16.111111

In [100…
```
#Product proability for Fitness levels
pd.crosstab(aerofit["Product"],aerofit["Fitness"],normalize='columns')*100
```

Out[100]:

Product	Fitness 1	2	3	4	5
KP281	50.0	53.846154	55.670103	37.500000	6.451613
KP481	50.0	46.153846	40.206186	33.333333	0.000000
KP781	0.0	0.000000	4.123711	29.166667	93.548387

In [101…
```
#Fitness level probability for Product
pd.crosstab(aerofit["Product"],aerofit["Fitness"],normalize='index')*100
```

Out[101]:

Product	Fitness 1	2	3	4	5
KP281	1.250000	17.5	67.5	11.250000	2.5
KP481	1.666667	20.0	65.0	13.333333	0.0
KP781	0.000000	0.0	10.0	17.500000	72.5

In [102…
```
#Total probability for Usage and Product
pd.crosstab(aerofit["Product"],aerofit["Usage"],normalize=True)*100
```

Out[102]:

Product	Usage 2	3	4	5	6	7
KP281	10.555556	20.555556	12.222222	1.111111	0.000000	0.000000
KP481	7.777778	17.222222	6.666667	1.666667	0.000000	0.000000
KP781	0.000000	0.555556	10.000000	6.666667	3.888889	1.111111

In [103…
```
#Product wise probability for Usage level
pd.crosstab(aerofit["Product"],aerofit["Usage"],normalize='columns')*100
```

Out[103]:

Usage	2	3	4	5	6	7
Product						
KP281	57.575758	53.623188	42.307692	11.764706	0.0	0.0
KP481	42.424242	44.927536	23.076923	17.647059	0.0	0.0
KP781	0.000000	1.449275	34.615385	70.588235	100.0	100.0

In [104…

```
#Usage level probability for Product
pd.crosstab(aerofit["Product"],aerofit["Usage"],normalize='index')*100
```

Out[104]:

Usage	2	3	4	5	6	7
Product						
KP281	23.750000	46.250000	27.5	2.5	0.0	0.0
KP481	23.333333	51.666667	20.0	5.0	0.0	0.0
KP781	0.000000	2.500000	45.0	30.0	17.5	5.0

In [109…

```
#Total probability with respect to Income band and Product
def func(data):
    if data<=25000:
        return "0-25k"
    elif data>25000 and data<=50000:
        return "25k-50k"
    elif data>50000 and data<=100000:
        return "50k-100k"
    else:
        return "100k+"

aerofit["Income_band"]=aerofit["Income"].apply(func)


pd.crosstab(aerofit["Product"],aerofit["Income_band"],normalize=True)*100
```

Out[109]:

Income_band	100k+	25k-50k	50k-100k
Product			
KP281	0.000000	26.666667	17.777778
KP481	0.000000	16.666667	16.666667
KP781	1.666667	2.777778	17.777778

In [110…

```
#Product wise Probability for Income Bands
pd.crosstab(aerofit["Product"],aerofit["Income_band"],normalize='columns')*100
```

Out[110]:

Product	Income_band 100k+	25k-50k	50k-100k
KP281	0.0	57.831325	34.042553
KP481	0.0	36.144578	31.914894
KP781	100.0	6.024096	34.042553

In [111…

```
#Income wise probability for Product
pd.crosstab(aerofit["Product"],aerofit["Income_band"],normalize="index")*100
```

Out[111]:

Product	Income_band 100k+	25k-50k	50k-100k
KP281	0.0	60.0	40.0
KP481	0.0	50.0	50.0
KP781	7.5	12.5	80.0

In [112…

```
def func(data):
    if data<=50:
        return "0-50"
    elif data>50 and data<=100:
        return "50-100"
    elif data>100 and data<=150:
        return "100-150"
    elif data>150 and data<=200:
        return "150-200"
    elif data>200 and data<=250:
        return "200-250"
    elif data>250 and data<=300:
        return "250-300"
    else:
        return "300+"

aerofit["Miles_band"]=aerofit["Miles"].apply(func)
```

In [113…

```
# total probability with respect to Miles band and Product
pd.crosstab(aerofit["Product"],aerofit["Miles_band"],normalize=True)*100
```

Out[113]:

Product	Miles_band 0-50	100-150	150-200	200-250	250-300	300+	50-100
KP281	6.666667	8.888889	1.111111	0.000000	0.000000	0.000000	27.777778
KP481	2.777778	7.222222	1.111111	0.555556	0.000000	0.000000	21.666667
KP781	0.000000	5.000000	10.000000	0.555556	1.666667	0.555556	4.444444

In [114…

```
#Product wise probability for Miles band
pd.crosstab(aerofit["Product"],aerofit["Miles_band"],normalize="columns")*100
```

Out[114]:

Product	0-50	100-150	150-200	200-250	250-300	300+	50-100
KP281	70.588235	42.105263	9.090909	0.0	0.0	0.0	51.546392
KP481	29.411765	34.210526	9.090909	50.0	0.0	0.0	40.206186
KP781	0.000000	23.684211	81.818182	50.0	100.0	100.0	8.247423

In [115…

```
#Miles wise probability for Product
pd.crosstab(aerofit["Product"],aerofit["Miles_band"],normalize="index")*100
```

Out[115]:

Product	0-50	100-150	150-200	200-250	250-300	300+	50-100
KP281	15.000000	20.000000	2.500000	0.000000	0.0	0.0	62.5
KP481	8.333333	21.666667	3.333333	1.666667	0.0	0.0	65.0
KP781	0.000000	22.500000	45.000000	2.500000	7.5	2.5	20.0

In [116…

```
#Total probability with respect to Education levels and Product
pd.crosstab(aerofit["Product"],aerofit["Education"],normalize=True)*100
```

Out[116]:

Product	12	13	14	15	16	18	20	21
KP281	1.111111	1.666667	16.666667	2.222222	21.666667	1.111111	0.000000	0.000000
KP481	0.555556	1.111111	12.777778	0.555556	17.222222	1.111111	0.000000	0.000000
KP781	0.000000	0.000000	1.111111	0.000000	8.333333	10.555556	0.555556	1.666667

In [117…

```
#Product wise probability for Education level
pd.crosstab(aerofit["Product"],aerofit["Education"],normalize='columns')*100
```

Out[117]:

Product	12	13	14	15	16	18	20	21
KP281	66.666667	60.0	54.545455	80.0	45.882353	8.695652	0.0	0.0
KP481	33.333333	40.0	41.818182	20.0	36.470588	8.695652	0.0	0.0
KP781	0.000000	0.0	3.636364	0.0	17.647059	82.608696	100.0	100.0

In [118…

```
#Education wise probability for Product
pd.crosstab(aerofit["Product"],aerofit["Education"],normalize='index')*100
```

Out[118]:

Education	12	13	14	15	16	18	20	21
Product								
KP281	2.500000	3.750000	37.500000	5.000000	48.750000	2.500000	0.0	0.0
KP481	1.666667	3.333333	38.333333	1.666667	51.666667	3.333333	0.0	0.0
KP781	0.000000	0.000000	5.000000	0.000000	37.500000	47.500000	2.5	7.5

In [119…

```
def func(data):
    if data<=20:
        return "0-20"
    elif data>20 and data<=30:
        return "20-30"
    elif data>30 and data<=40:
        return "30-40"
    elif data >40 and data<=50:
        return "40-50"
    else:
        return "50+"

aerofit["Age_band"]=aerofit["Age"].apply(func)
```

In [120…

```
# total probability for age band and Product
pd.crosstab(aerofit["Product"],aerofit["Age_band"],normalize=True)*100
```

Out[120]:

Age_band	0-20	20-30	30-40	40-50
Product				
KP281	3.333333	27.222222	10.555556	3.333333
KP481	2.222222	17.222222	12.777778	1.111111
KP781	0.000000	16.666667	3.333333	2.222222

In [121…

```
# Product wise probability for Age band
pd.crosstab(aerofit["Product"],aerofit["Age_band"],normalize='columns')*100
```

Out[121]:

Age_band	0-20	20-30	30-40	40-50
Product				
KP281	60.0	44.545455	39.583333	50.000000
KP481	40.0	28.181818	47.916667	16.666667
KP781	0.0	27.272727	12.500000	33.333333

In [122…

```
# Age wise probability for Products
pd.crosstab(aerofit["Product"],aerofit["Age_band"],normalize='index')*100
```

Out[122]:

Age_band	0-20	20-30	30-40	40-50
Product				
KP281	7.500000	61.250000	23.750000	7.500000
KP481	6.666667	51.666667	38.333333	3.333333
KP781	0.000000	75.000000	15.000000	10.000000

Checking correlation for metrices

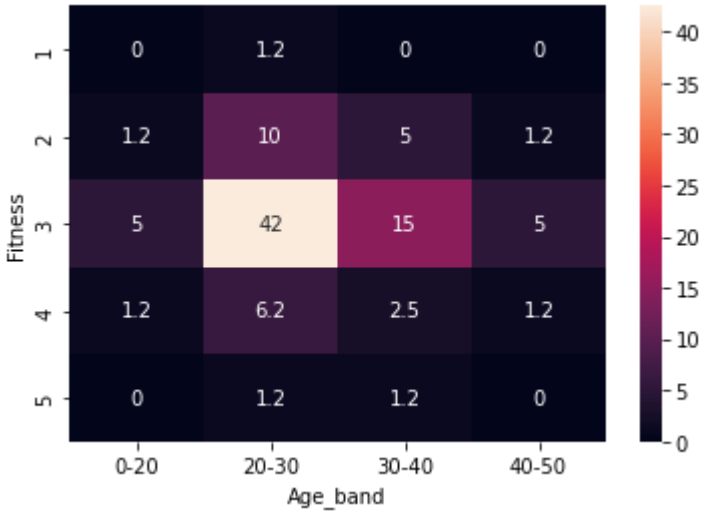
for KP281

In [124…
```
aerofit_281=aerofit.loc[aerofit["Product"]=="KP281"]
```

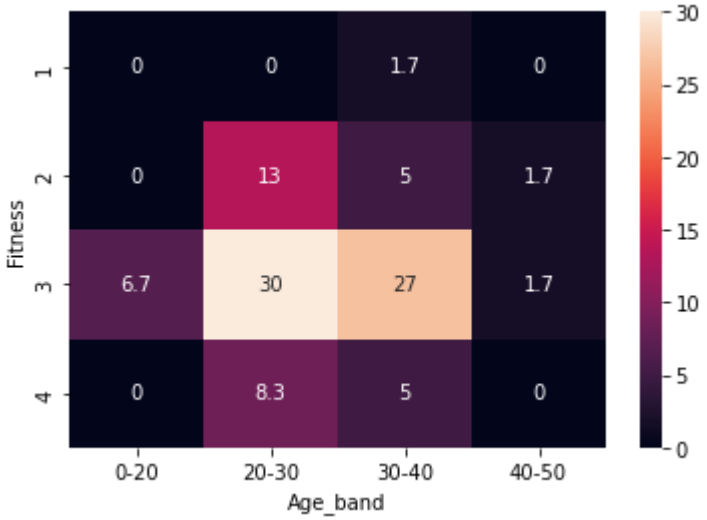
In [125…
```
aerofit_481=aerofit.loc[aerofit["Product"]=="KP481"]
```

In [126…
```
aerofit_781=aerofit.loc[aerofit["Product"]=="KP781"]
```

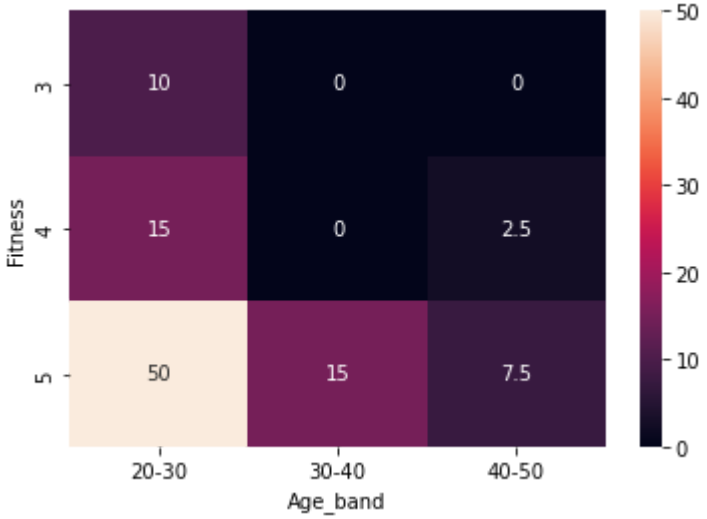
In [131…
```
#Probability distribution of product KP281 correlating Fitness and Age_band
sns.heatmap(pd.crosstab(aerofit_281["Fitness"],aerofit_281["Age_band"],normalize=True)
plt.show()
```



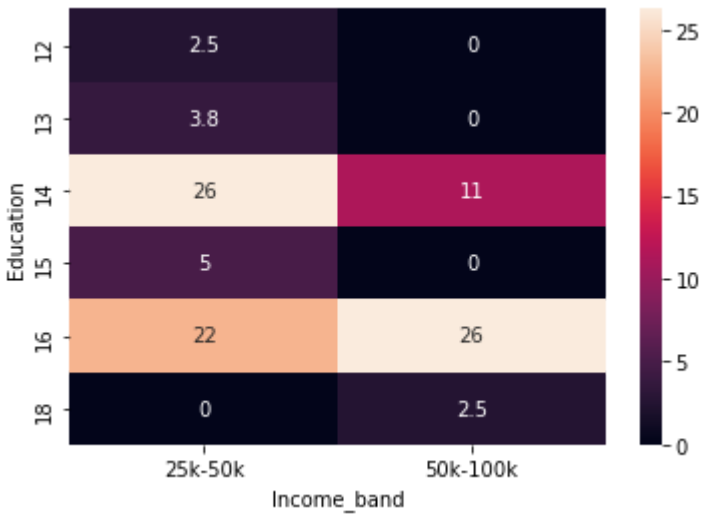
In [133…
```
#Probability distribution of product KP481 correlating Fitness and Age_band
sns.heatmap(pd.crosstab(aerofit_481["Fitness"],aerofit_481["Age_band"],normalize=True)
plt.show()
```



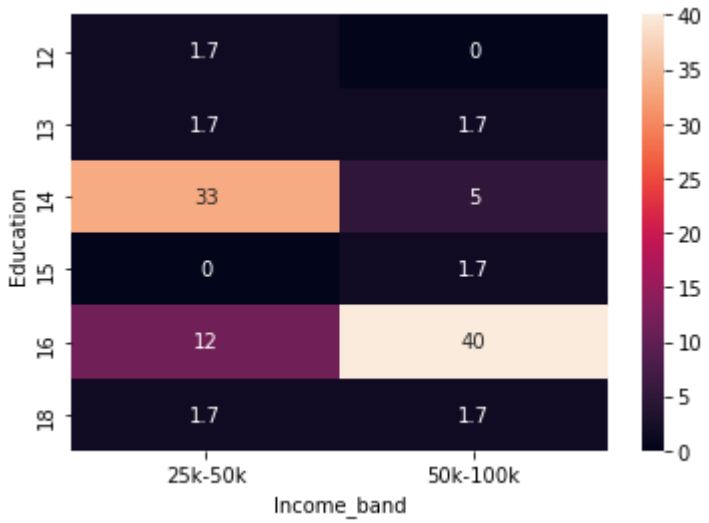
In [134…
```
#Probability distribution of product KP781 correlating Fitness and Age_band
sns.heatmap(pd.crosstab(aerofit_781["Fitness"],aerofit_781["Age_band"],normalize=True)
plt.show()
```



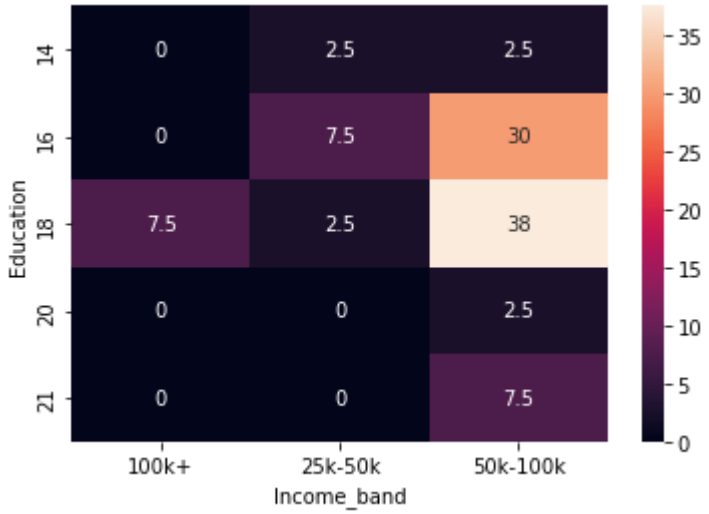
In [132…
```
#Probability distribution of product KP281 correlating Education and Income_band
sns.heatmap(pd.crosstab(aerofit_281["Education"],aerofit_281["Income_band"],normalize=
plt.show()
```



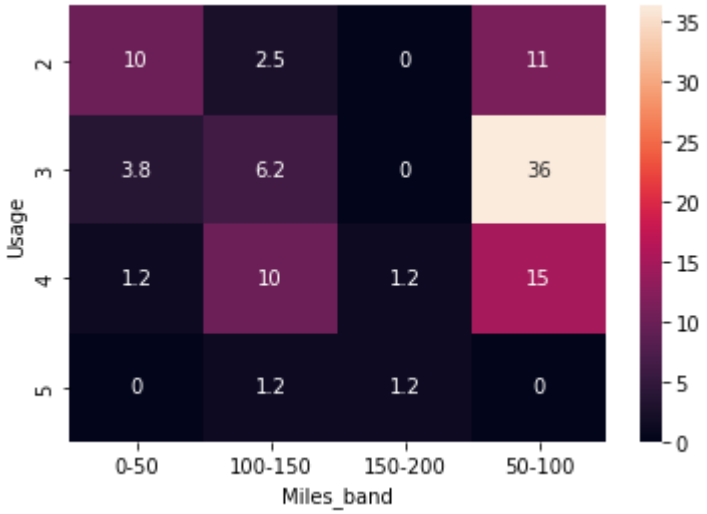
In [135…
```
#Probability distribution of product KP481 correlating Education and Income_band
sns.heatmap(pd.crosstab(aerofit_481["Education"],aerofit_481["Income_band"],normalize=
plt.show()
```



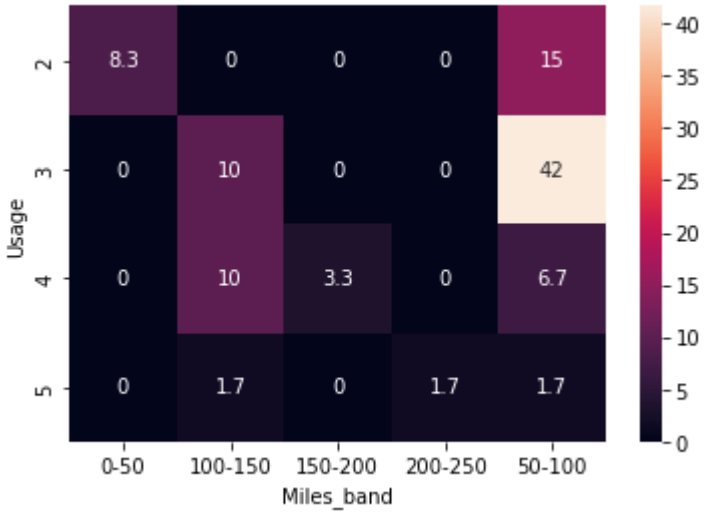
In [136…
```
#Probability distribution of product KP781 correlating Education and Income_band
sns.heatmap(pd.crosstab(aerofit_781["Education"],aerofit_781["Income_band"],normalize=
plt.show()
```



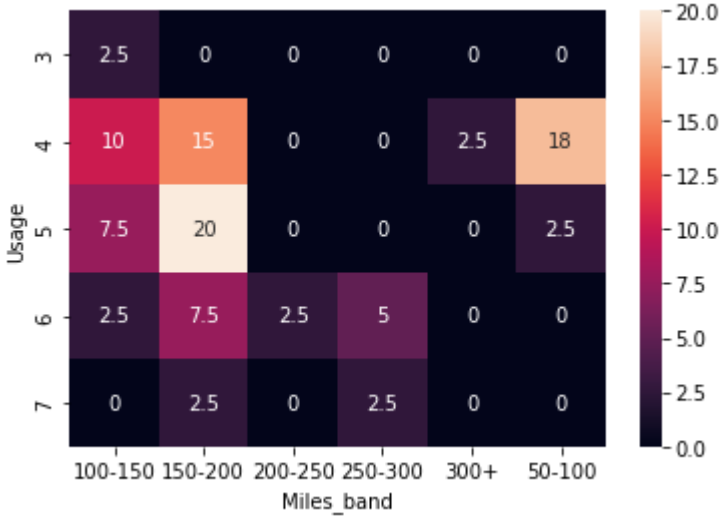
In [137…
```
#Probability distribution of product KP281 correlating Usage and Miles_band
sns.heatmap(pd.crosstab(aerofit_281["Usage"],aerofit_281["Miles_band"],normalize=True)
plt.show()
```



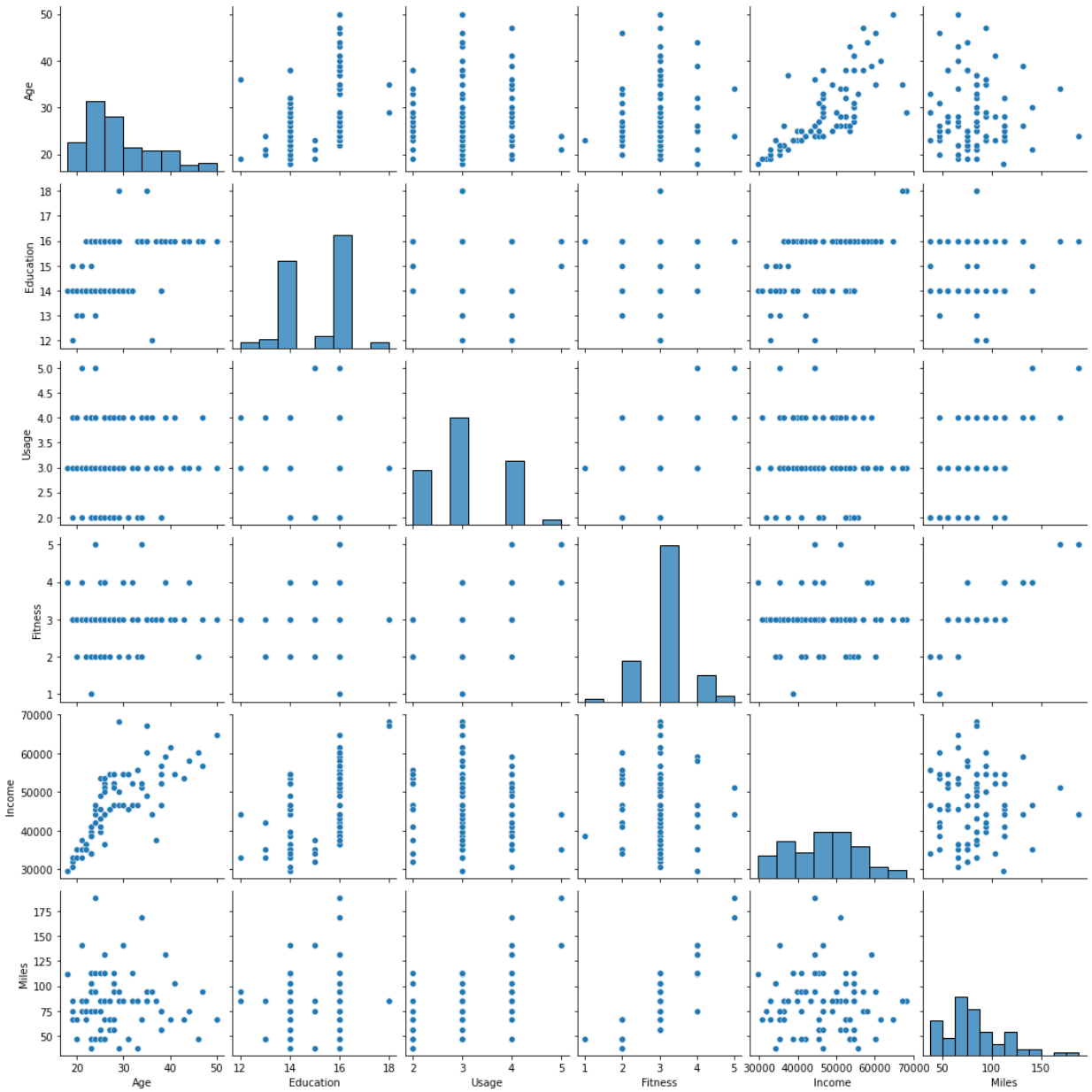
In [138…
```
#Probability distribution of product KP481 correlating Usage and Miles_band
sns.heatmap(pd.crosstab(aerofit_481["Usage"],aerofit_481["Miles_band"],normalize=True)
plt.show()
```



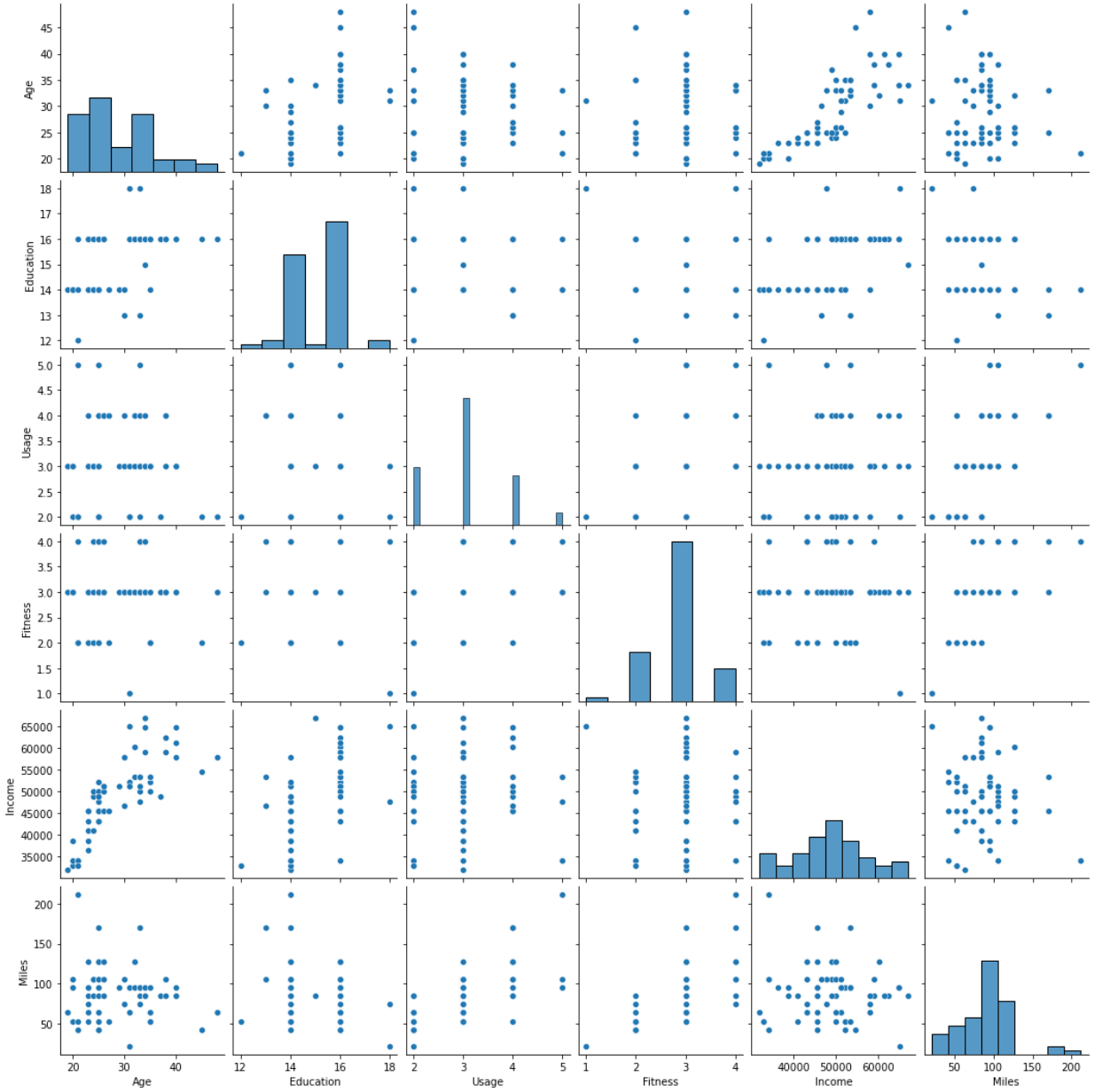
In [139…
```
#Probability distribution of product KP781 correlating Usage and Miles_band
sns.heatmap(pd.crosstab(aerofit_781["Usage"],aerofit_781["Miles_band"],normalize=True)
plt.show()
```



In [143…
```
#Pairplot for product KP281
sns.pairplot(aerofit_281)
plt.show()
```

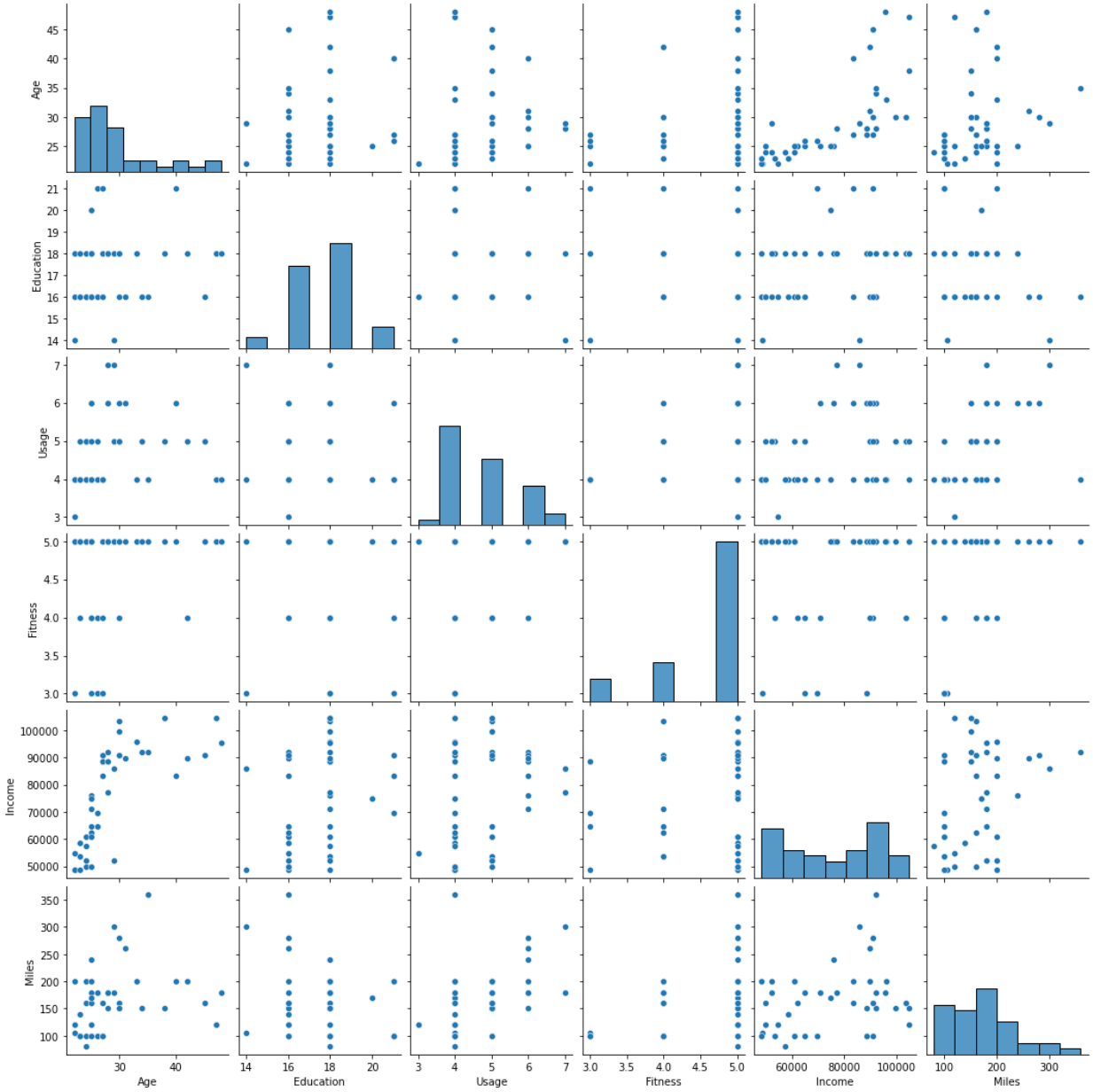


In [144…
```
#Pairplot for product KP481
sns.pairplot(aerofit_481)
plt.show()
```



In [145…

```
#pairplot for product KP781
sns.pairplot(aerofit_781)
plt.show()
```



Answering basic questions for probability

With all the above steps you can answer questions like: What is the probability of a male customer buying a KP781 treadmill?

Looking at the cross tab where product wise probability of Genders are found above, it is found that that the probability of male buying KP781 treadmill is 31.73%

Probability of person aged 20-30 years if they use KP281

Using crosstab, the answer found out is 61.25%

Probability of fitness level 5 given that product used is KP781

72.5%

Customer Profiling

In [ ]:

In [107…
```
aerofit.groupby(["Product","Gender","MaritalStatus"])["Age","Fitness","Education","Inc
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_5708\4150797457.py:1: FutureWarning: Indexin
g with multiple keys (implicitly converted to a tuple of keys) will be deprecated, us
e a list instead.
  aerofit.groupby(["Product","Gender","MaritalStatus"])["Age","Fitness","Educatio
n","Income","Miles","Usage"].aggregate({"Age":["mean",'std'],"Fitness":["mean","st
d"],"Education":["mean","std"],"Income":["mean","std"],"Miles":["mean","std"],"Usag
e":["mean","std"]})
```

Out[107]:

			Age		Fitness		Education	
			mean	std	mean	std	mean	std
Product	Gender	MaritalStatus						
KP281	Female	Partnered	28.333333	7.411011	2.851852	0.662379	14.888889	0.974022
		Single	28.692308	6.725382	2.923077	0.640513	15.538462	1.198289
	Male	Partnered	31.380952	7.857965	2.857143	0.727029	15.428571	1.247855
		Single	25.631579	5.688297	3.263158	0.561951	14.473684	1.306753
KP481	Female	Partnered	30.000000	6.380775	2.933333	0.457738	15.200000	1.014185
		Single	28.142857	5.171775	2.785714	0.892582	15.214286	1.577660
	Male	Partnered	30.380952	7.927649	2.904762	0.624881	15.285714	1.230563
		Single	25.200000	4.962078	3.000000	0.471405	14.500000	0.849837
KP781	Female	Partnered	29.000000	3.366502	5.000000	0.000000	17.500000	1.000000
		Single	24.333333	1.527525	4.000000	1.000000	18.333333	2.516611
	Male	Partnered	30.000000	7.180220	4.631579	0.597265	17.421053	1.609548
		Single	28.928571	8.061917	4.642857	0.744946	16.928571	1.685426

Recommendations

Recommendations: By observing dataset we can find out that there are three products KP281, KP481, KP781 each having their own properties and probability distribution across various parameters such as across Gender, Marital Status, Income levels , Fitness levels etc. By observing cross tabs we can find out that probability distributions of Product with respect to other parameters. It is observed that fitness levels, Income levels , Usage levels each have their effect on the purchase of Products. The product KP781 is basically recommended for the ones having high income levels , fitness levels and mostly for Males. Females also are well distributed for their probability upto Product KP481 but most females tend not to buy KP781 as the treadmill is

for high endurance training and it has something to do with high biological strength. Using the heatmaps and pairplots we can find out that correlation between various parameters such Fitness and Age. It is recommended to just have a look at the Pairplots and heatmaps for better understanding of correlation. Fitness tends to be average and higher for the age group 20-40 years mainly 20-30 years are the ones who are involved heavily with treadmill usage. The ones aged <20 and >40 are low on fitness levels and basically recommended to buy low endurance treadmill that is KP281. Also various other insights can be drawn from the report.

In [ ]: