Aerofit_treadmill

March 12, 2023

[2]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats.contingency import margins
from scipy.stats import norm
import numpy as np
```

[3]:
```
df = pd.read_csv("Aerofit_treadmill.csv")
```

[4]:
```
df.head()
```

[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

[5]:
```
df.dtypes
```

[5]:
```
Product          object
Age               int64
Gender           object
Education         int64
MaritalStatus    object
Usage             int64
Fitness           int64
Income            int64
Miles             int64
dtype: object
```

[6]:
```
df.describe()
```

[6]:

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

```
25%     24.000000   14.000000    3.000000    3.000000   44058.750000
50%     26.000000   16.000000    3.000000    3.000000   50596.500000
75%     33.000000   16.000000    4.000000    4.000000   58668.000000
max     50.000000   21.000000    7.000000    5.000000  104581.000000
```

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

[7]:
```
df.info()
```

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

[8]:
```
df.shape
```

[8]:
```
(180, 9)
```

[9]:
```
df.head(3)
```

[9]:

	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

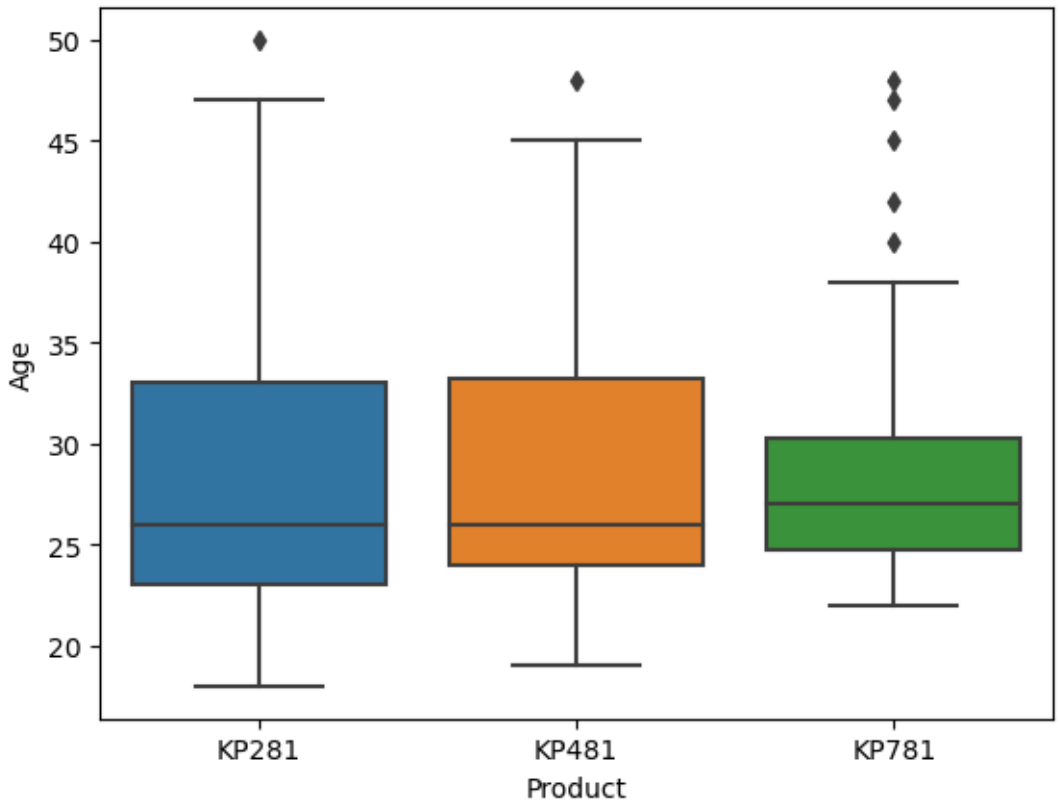
[10]:
```
df.Age.dtype
```

[10]:
```
dtype('int64')
```

1 Detecting Outliers

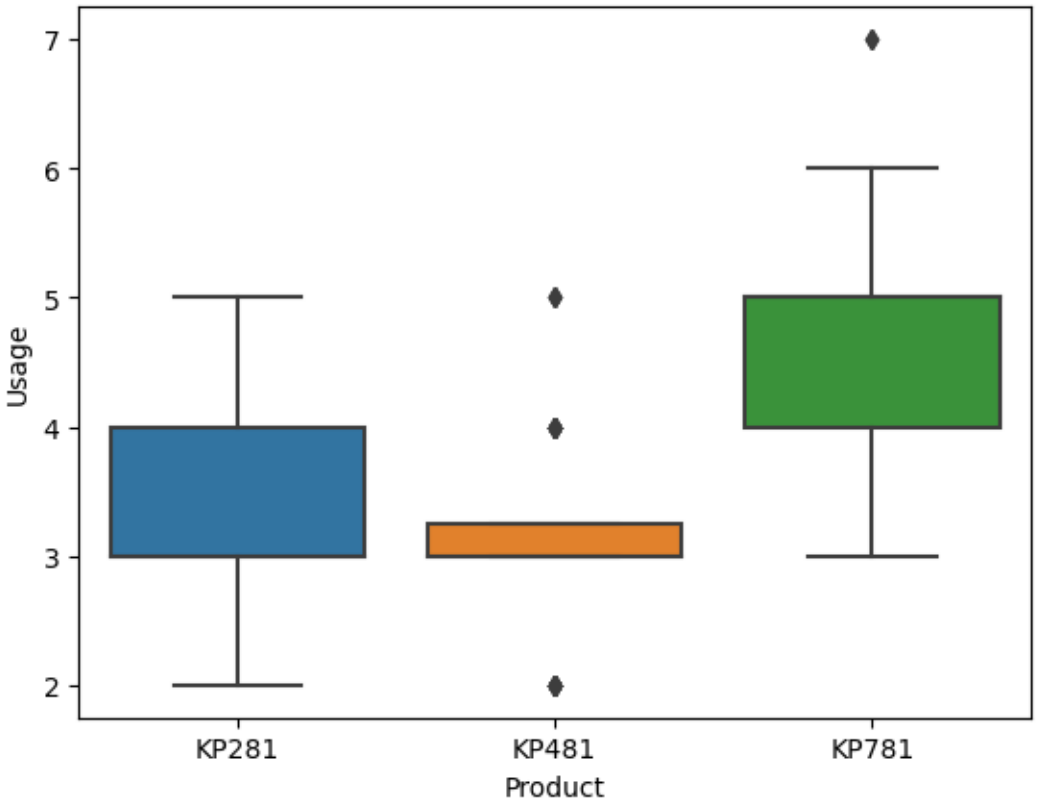
[11]:
```
sns.boxplot(data=df,x="Product",y="Age")
```

[11]: <AxesSubplot:xlabel='Product', ylabel='Age'>



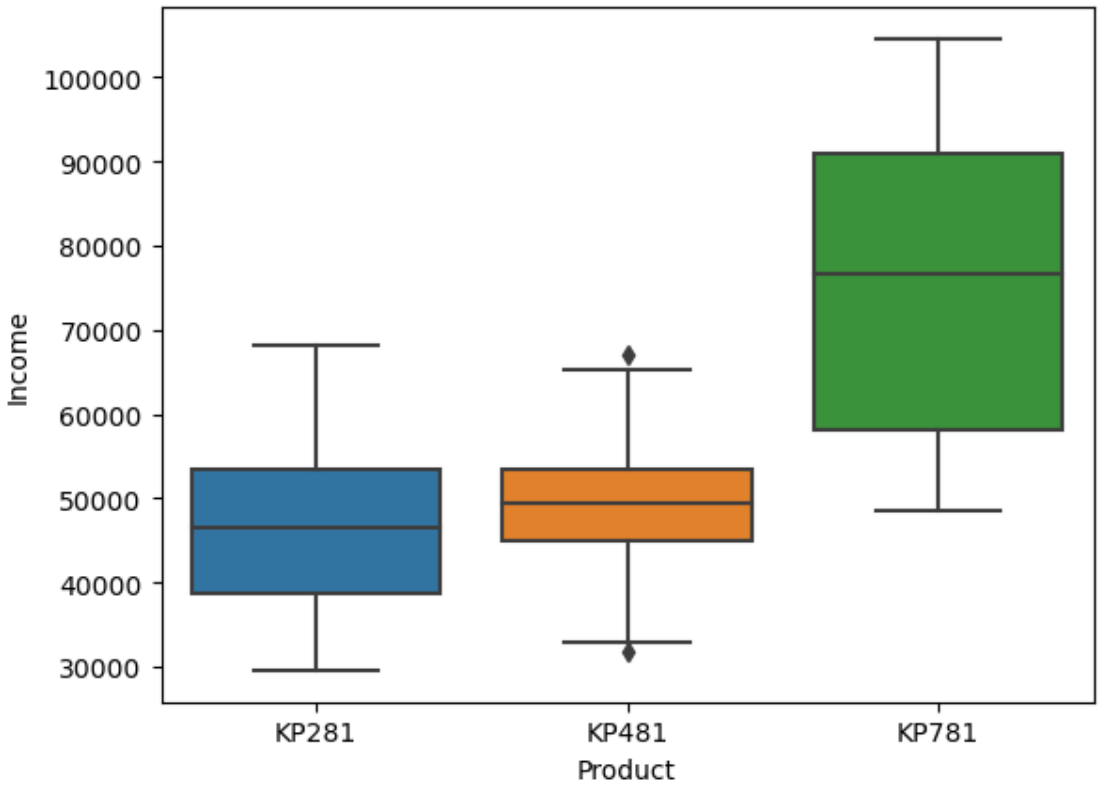
[12]:
```
sns.boxplot(data=df,x="Product",y="Usage")
```

[12]: <AxesSubplot:xlabel='Product', ylabel='Usage'>



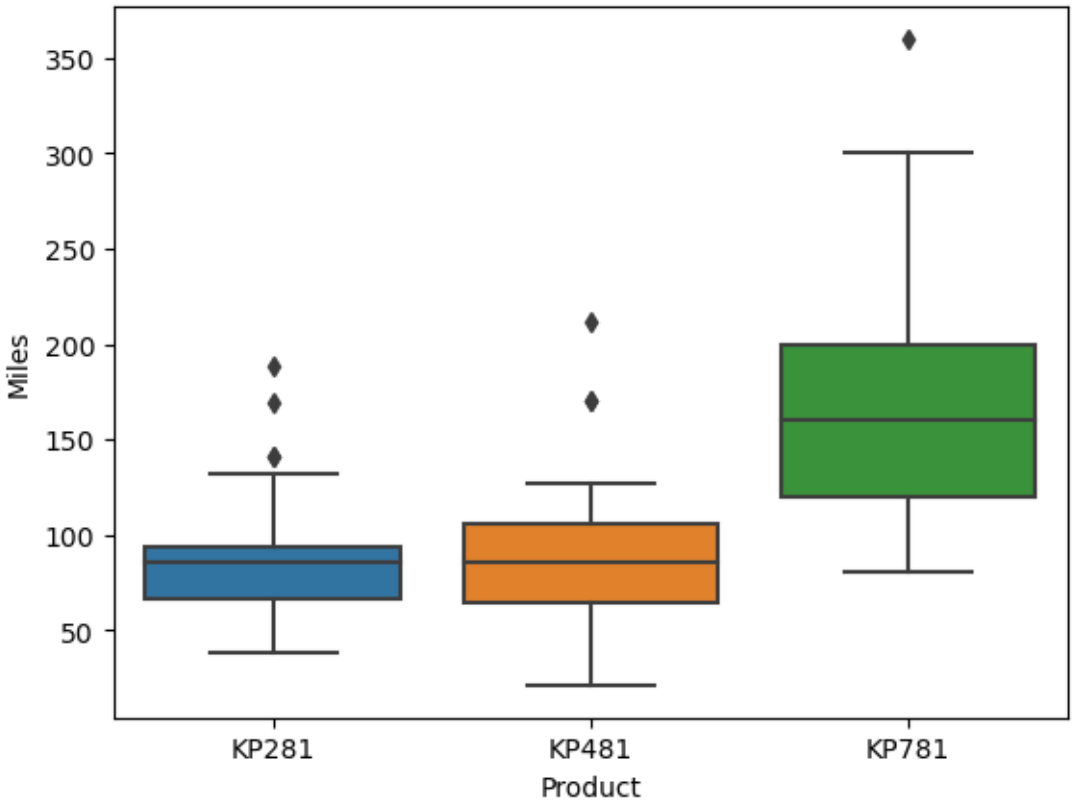
[13]:
```
sns.boxplot(data=df,x="Product",y="Income")
```

[13]:
```
<AxesSubplot:xlabel='Product', ylabel='Income'>
```



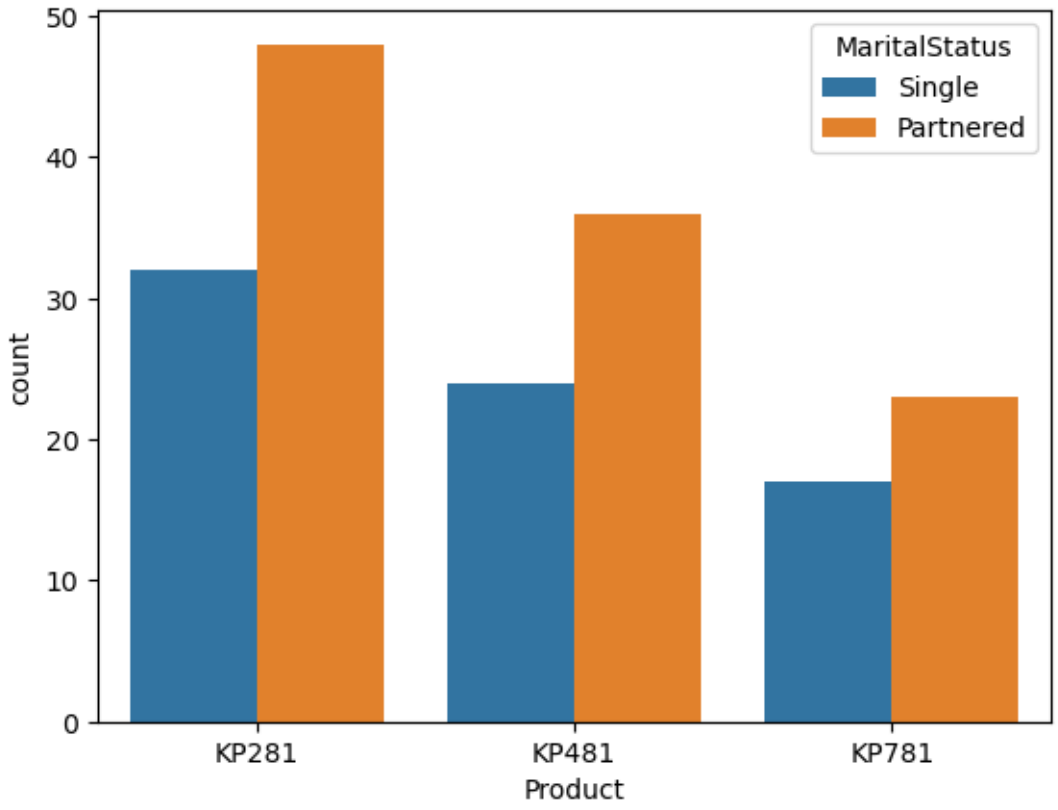
[14]:
```
sns.boxplot(data=df,x="Product",y="Miles")
```

[14]: <AxesSubplot:xlabel='Product', ylabel='Miles'>



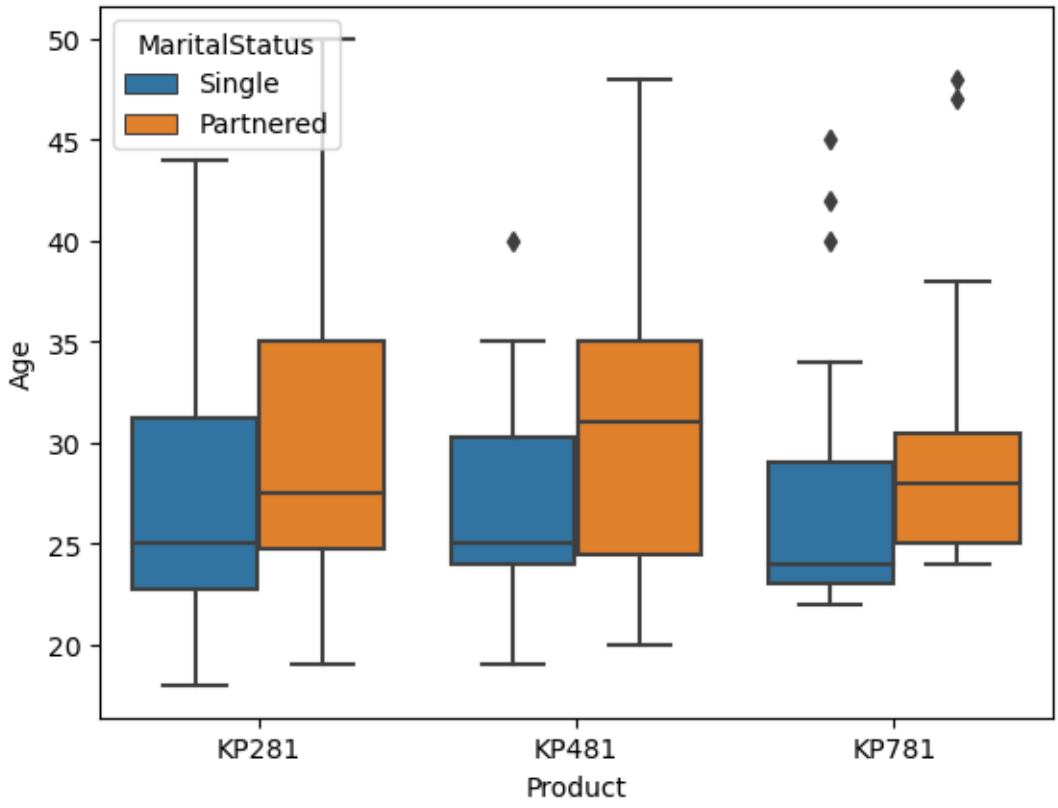
[15]:
```
sns.countplot(data=df,x="Product",hue="MaritalStatus")
```

[15]: <AxesSubplot:xlabel='Product', ylabel='count'>



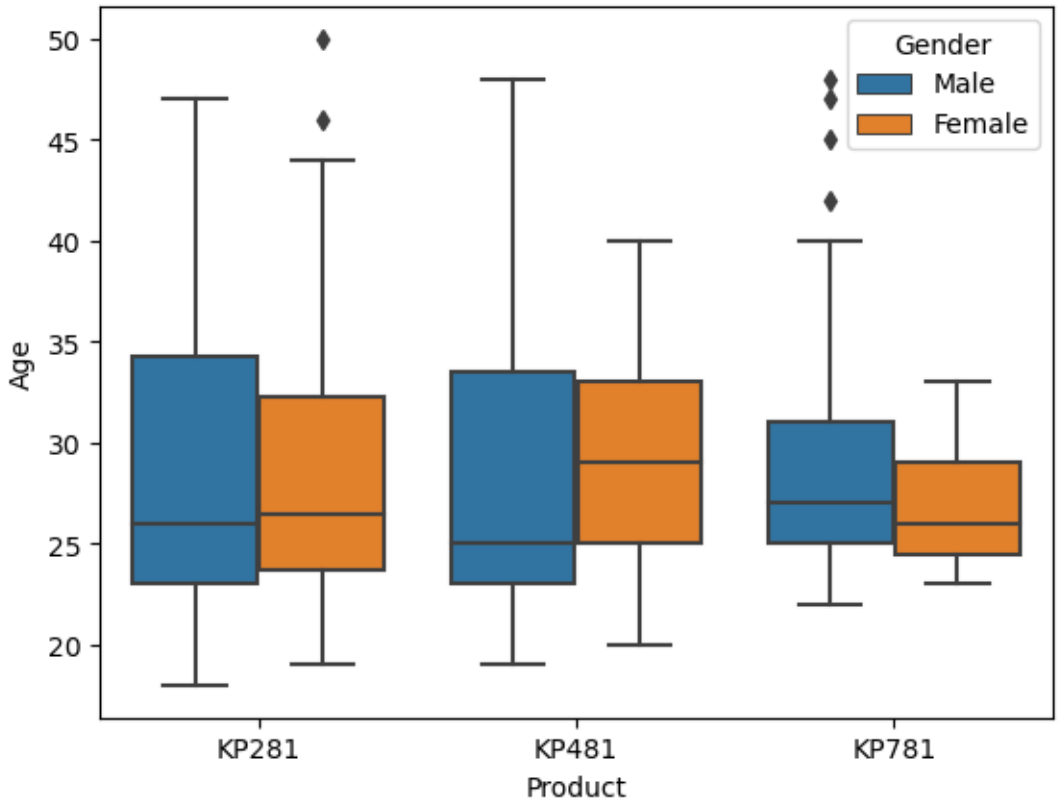
[16]:
```
sns.boxplot(data=df,x="Product",y= "Age", hue="MaritalStatus")
```

[16]: <AxesSubplot:xlabel='Product', ylabel='Age'>



[17]:
```
sns.boxplot(data=df,x="Product",y= "Age", hue="Gender")
```

[17]: <AxesSubplot:xlabel='Product', ylabel='Age'>



2 Contigency Table

[19]:
```
pd.crosstab(df["Product"], df["Gender"],margins=True)
```

[19]:
```
Gender   Female  Male  All
Product
KP281        40    40   80
KP481        29    31   60
KP781         7    33   40
All          76   104  180
```

[20]:
```
#Probablility of product KP781 given Female
7/76
```

[20]:
```
0.09210526315789473
```

[21]:
```
#Probablility of product KP781 given Male
33/104
```

[21]:
```
0.3173076923076923
```

[22]:
```
#Probablility of product KP481 given Female
29/76
```

[22]: 0.3815789473684211

[23]:
```
#Probablility of product KP481 given Male
31/104
```

[23]: 0.2980769230769231

[24]:
```
#Probablility of product KP281 given Female
40/76
```

[24]: 0.5263157894736842

[25]:
```
#Probablility of product KP281 given Male
40/104
```

[25]: 0.38461538461538464

[26]:
```
pd.crosstab(df["Product"], df["MaritalStatus"],margins=True)
```

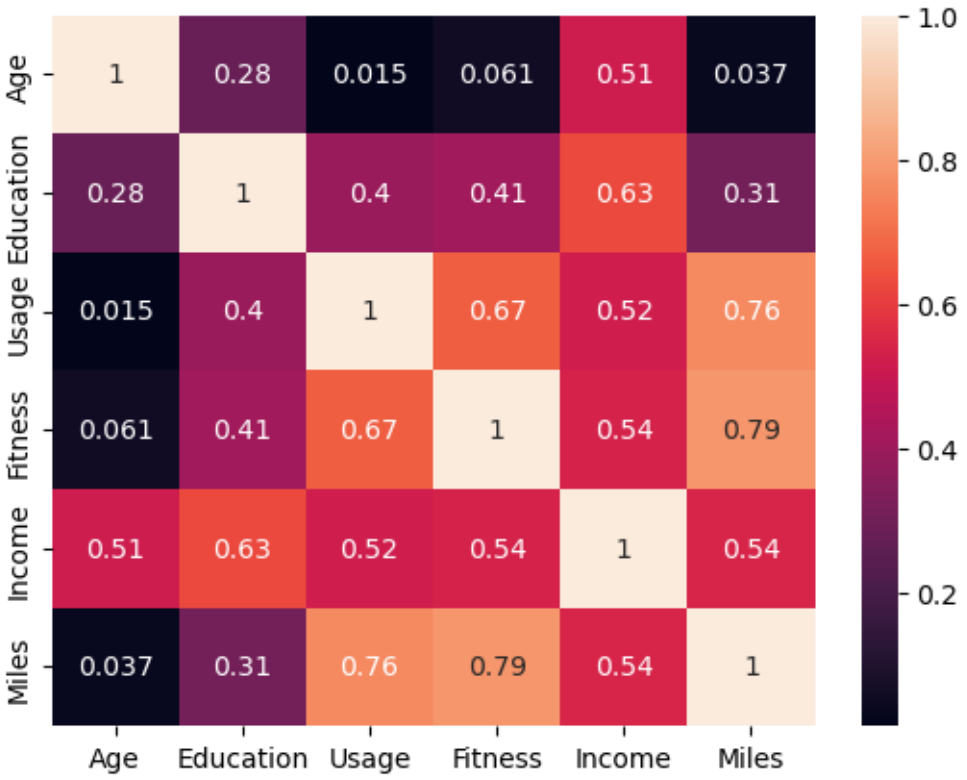
[26]:

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

#From the above table, 1. 44% of people bought KP281 2. 33% of people bought KP481 3. 22% of people bought KP781

[27]:
```
sns.heatmap(df.corr(),annot=True)
```

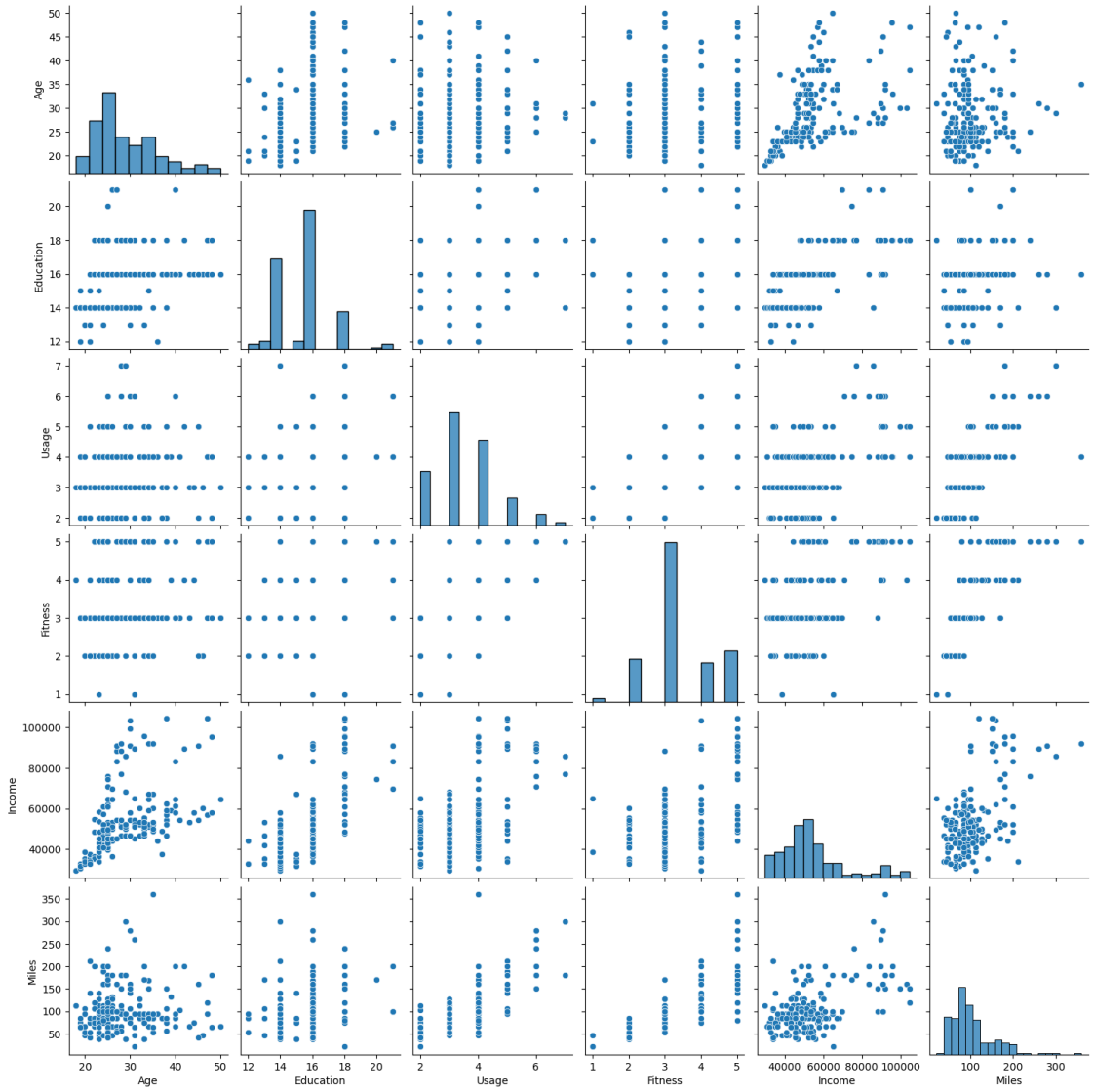
[27]: <AxesSubplot:>



1. Fitness, Usage are highly positively correlated with Miles
2. Income is positively correlated with other variables, which means if age is increasing the income also increases.
3. If the usage of threadmill is high their fitness level also high.

[28]:
```
sns.pairplot(data=df)
```

[28]:
```
<seaborn.axisgrid.PairGrid at 0x1fa21bf2460>
```



[29]:
```
pd.crosstab(df["Product"], df["Gender"],margins=True,normalize=True)*100
```

[29]:

Gender	Female	Male	All
Product			
KP281	22.222222	22.222222	44.444444
KP481	16.111111	17.222222	33.333333
KP781	3.888889	18.333333	22.222222
All	42.222222	57.777778	100.000000

1. What is the probability of a male customer buying a KP781 treadmill?

   Ans:18.33% of Male customers KP781

2. What is the probability of a female customer buying a KP781 treadmill?

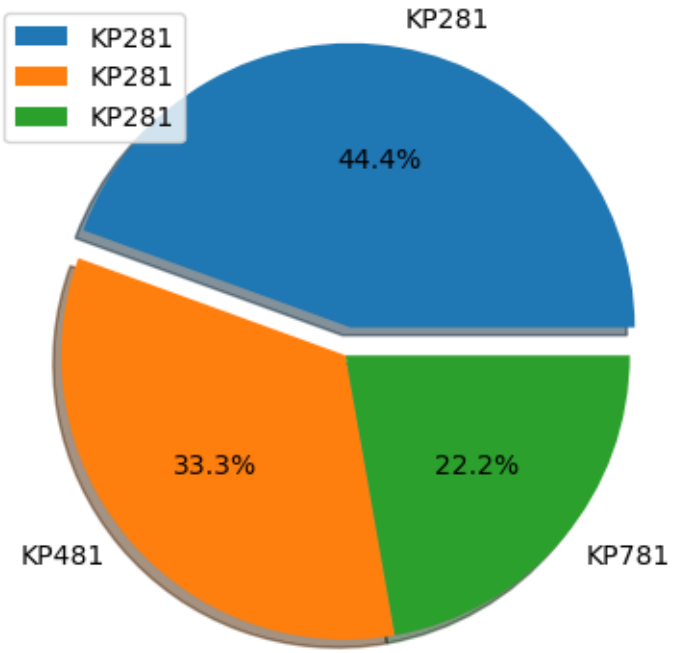
Ans: 3.88% of Female customers bought KP781

[30]:
```
pd.crosstab(df["Product"], df["MaritalStatus"],margins=True,normalize=True)*100
```

[30]:
```
MaritalStatus  Partnered     Single         All
Product
KP281          26.666667  17.777778   44.444444
KP481          20.000000  13.333333   33.333333
KP781          12.777778   9.444444   22.222222
All            59.444444  40.555556  100.000000
```

[31]:
```
plt.pie(x=df["Product"].
  ↪value_counts(),labels=["KP281","KP481","KP781"],autopct='%1.1f%%',explode=[0.
  ↪1,0,0],shadow=True)
plt.legend(df["Product"],loc="upper left")
```

[31]:
```
<matplotlib.legend.Legend at 0x1fa245a9940>
```



From the above table, we can see that,

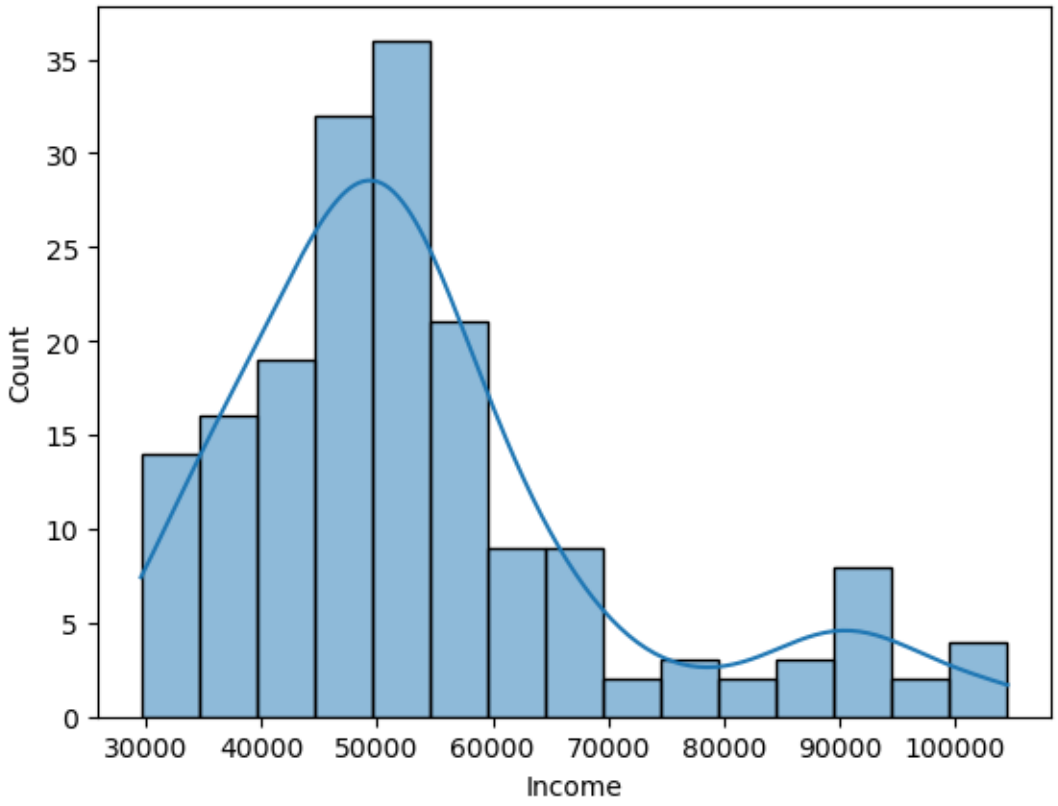
1. Most of the married people are buying treadmill compared to singles,
2. 26% of Married people buying KP281, 20% of Married people buying KP481,12% of married people are buying KP781

[32]:
```
df["Income"].max(), df["Income"].min()
```

[32]: (104581, 29562)

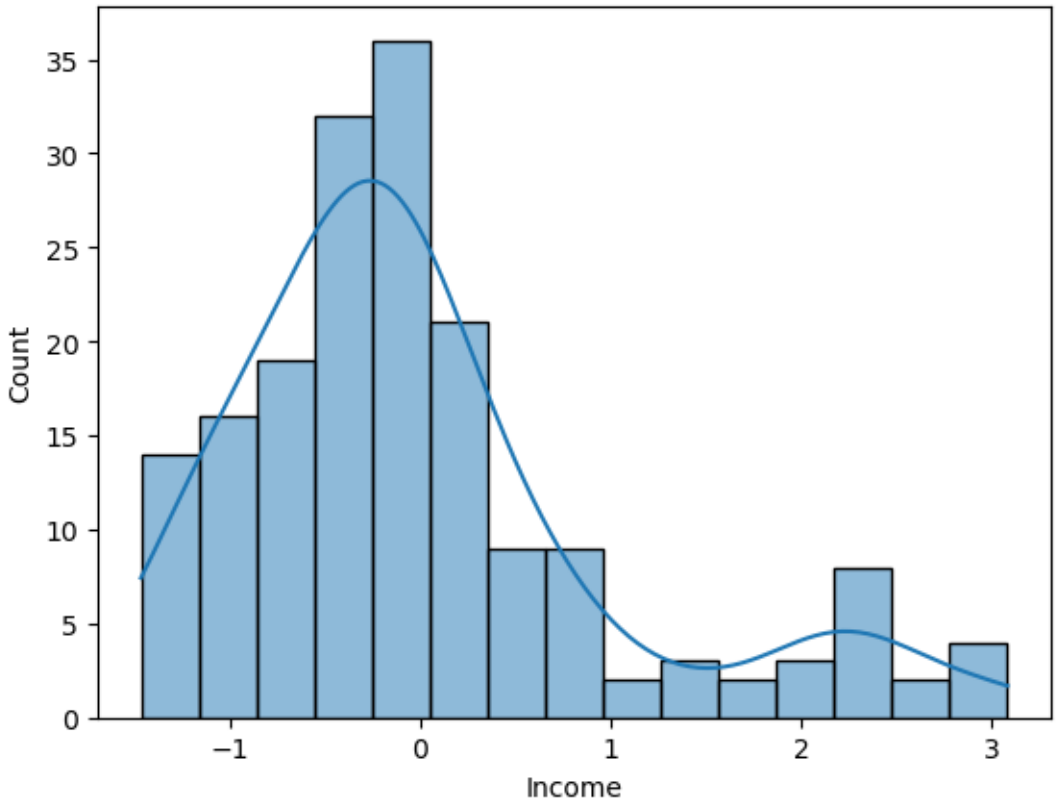
[33]:
```
sns.histplot(data=df,x="Income",kde=True)
```

[33]: <AxesSubplot:xlabel='Income', ylabel='Count'>



[34]:
```
sns.histplot(data=df, x=(df["Income"] - df["Income"].mean())/df["Income"].
↪std(),kde=True)
```

[34]: <AxesSubplot:xlabel='Income', ylabel='Count'>



[35]:
```
norm.cdf((60000 - df["Income"].mean())/df["Income"].std())
```

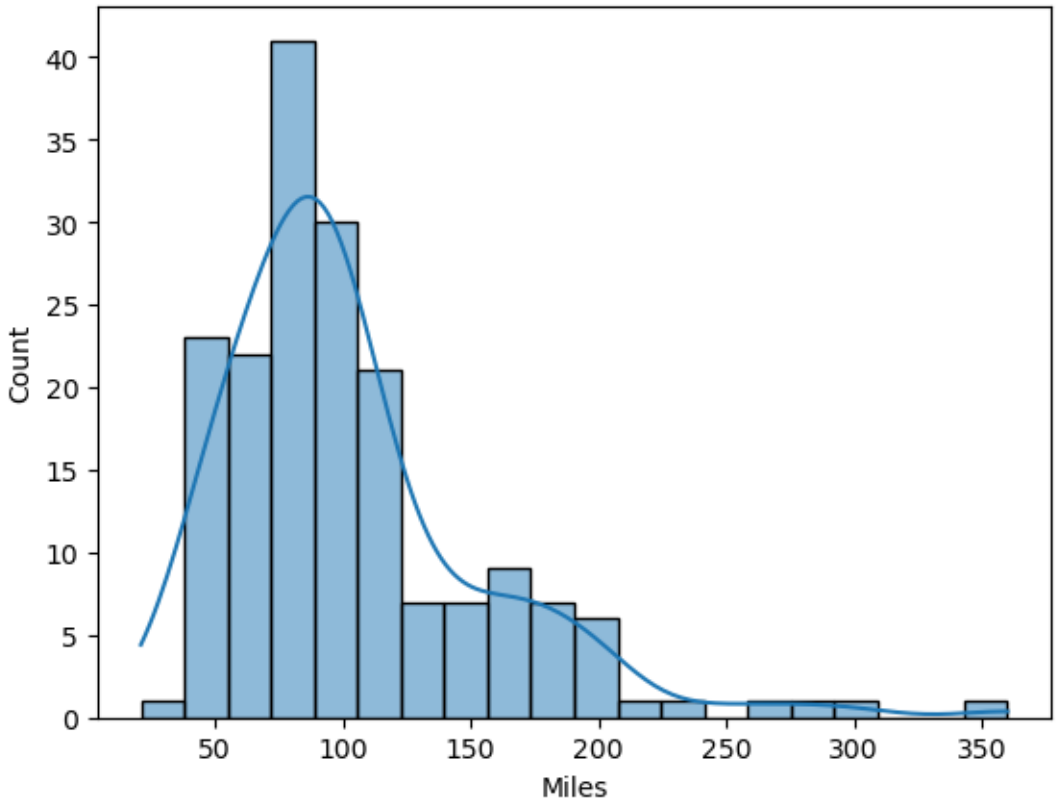
[35]: 0.6482045084887522

[36]:
```
1 - norm.cdf((60000 - df["Income"].mean())/df["Income"].std())
```

[36]: 0.3517954915112478

[37]:
```
sns.histplot(data=df,x="Miles",kde=True)
```

[37]: <AxesSubplot:xlabel='Miles', ylabel='Count'>



[39]:
```
df[df["Usage"] >=6]["Income"]
```

[39]:
```
154    70966
155    75946
162    92131
163    77191
164    88396
166    85906
167    90886
170    89641
175    83416
Name: Income, dtype: int64
```

[40]:
```
df[df["Fitness"] ==5].describe()
```

[40]:

	Age	Education	Usage	Fitness	Income	Miles
count	31.000000	31.000000	31.000000	31.0	31.000000	31.000000
mean	29.806452	17.064516	4.838710	5.0	74396.709677	178.935484
std	7.295749	1.436094	1.003221	0.0	19866.024082	60.290926
min	22.000000	14.000000	3.000000	5.0	44343.000000	80.000000
25%	24.000000	16.000000	4.000000	5.0	53536.000000	150.000000

```
50%    28.000000  16.000000   5.000000      5.0   77191.000000  170.000000
75%    33.500000  18.000000   5.500000      5.0   91508.500000  200.000000
max    48.000000  21.000000   7.000000      5.0  104581.000000  360.000000
```

[41]:
```
Probability_of_kp781_given_incomegreater_than_70000 = df[(df["Product"]
  ↪=="KP781")&(df["Income"]>70000)].shape[0]/df[(df["Income"]>50000)].shape[0]
Probability_of_kp781_given_incomegreater_than_70000
```

[41]:
```
0.23711340206185566
```

[42]:
```
df[df["Fitness"] ==3]["Product"].value_counts()
```

[42]:
```
KP281    54
KP481    39
KP781     4
Name: Product, dtype: int64
```

[43]:
```
df[df["Fitness"] ==5]["MaritalStatus"].value_counts()
```

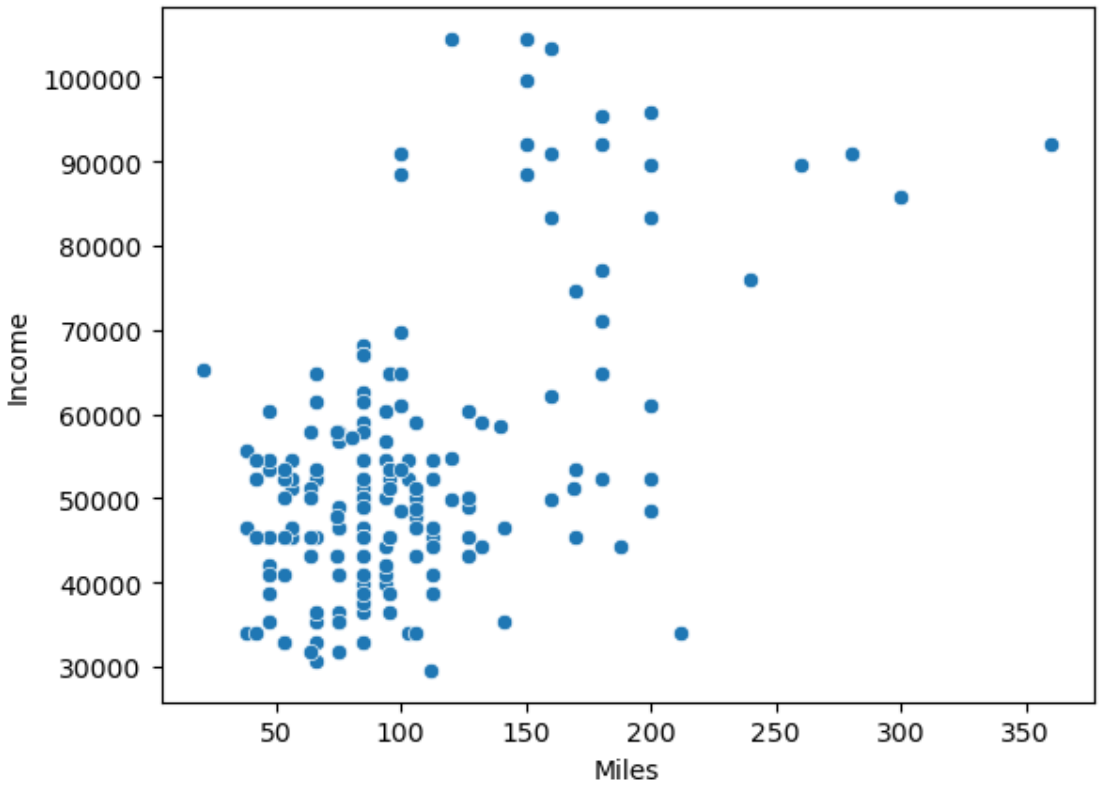
[43]:
```
Partnered    18
Single       13
Name: MaritalStatus, dtype: int64
```

[44]:
```
df[df["Fitness"] ==5]["Gender"].value_counts()
```

[44]:
```
Male      25
Female     6
Name: Gender, dtype: int64
```

[45]:
```
sns.scatterplot(data=df,x="Miles",y="Income")
```

[45]:
```
<AxesSubplot:xlabel='Miles', ylabel='Income'>
```



3 Conditional Probabilities of products given Income range

[108]:
```
#Probability of KP781 given that income greater than 70000
df[(df["Product"] =="KP781")&(df["Income"]>70000)].shape[0]/
  ↪df[(df["Income"]>70000)].shape[0]
```

[108]: 1.0

[47]:
```
#Probability of KP481 given that income greater than 50000
df[(df["Product"] =="KP481")&(df["Income"]>20000)].shape[0]/
  ↪df[(df["Income"]>20000)].shape[0]
```

[47]: 0.3333333333333333

[48]:
```
#Probability of KP481 given that income lesser than 70000 and greater than 50000
df[(df["Product"] =="KP481")&((df["Income"]>=50000) & (df["Income"]<=70000))].
  ↪shape[0]/df[(df["Income"]>=50000) & (df["Income"]<=70000)].shape[0]
```

[48]: 0.40540540540540543

[49]:
```
#Probability of KP281 given that income lesser than 50000 and greater than 20000
df[(df["Product"] =="KP281")&((df["Income"]>=20000) & (df["Income"]<=50000))].
↪shape[0]/df[(df["Income"]>=20000) & (df["Income"]<=70000)].shape[0]
```

[49]: 0.3057324840764331

[50]:
```
df[(df["Income"]>60000)].shape
```

[50]: (42, 9)

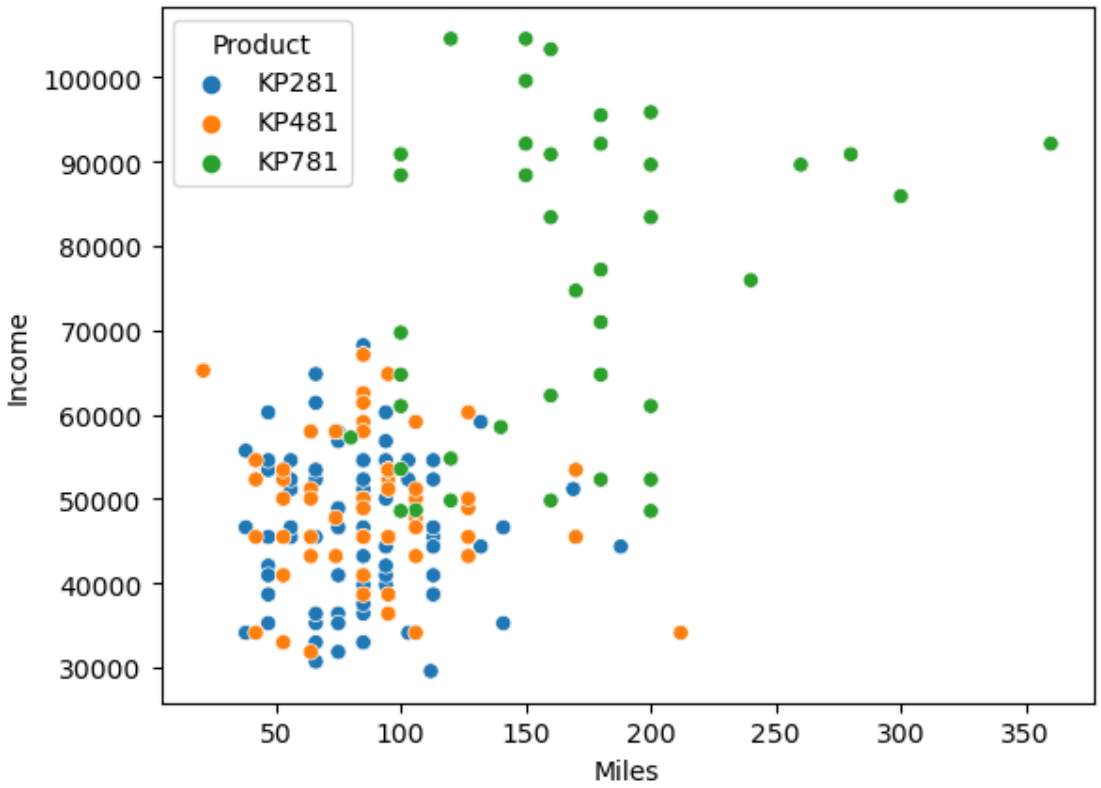
[51]:
```
df.head()
```

[51]:

	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

[52]:
```
sns.scatterplot(data=df,x="Miles",y="Income",hue="Product")
```

[52]: <AxesSubplot:xlabel='Miles', ylabel='Income'>



4 Conditional Probabilities of product given Miles

[53]:
```
#Probability of Miles > 200 given that the product is KP781
df[(df["Product"] =="KP781")&(df["Miles"]>200)].shape[0]/df[(df["Miles"]>200)].
 ↪shape[0]
```

[53]: 0.8333333333333334

[54]:
```
#Probability of Miles > 200 given that the product is KP481
df[(df["Product"] =="KP481")&(df["Miles"]>200)].shape[0]/df[(df["Miles"]>200)].
 ↪shape[0]
```

[54]: 0.16666666666666666

[55]:
```
#Probability of Miles > 200 given that the product is KP281
df[(df["Product"] =="KP281")&(df["Miles"]>200)].shape[0]/df[(df["Miles"]>200)].
 ↪shape[0]
```

[55]: 0.0

[56]:
```
#Probability of Miles > 100 and Miles<200 given that the product is KP481
df[(df["Product"] =="KP481")&((df["Miles"]<=200) & (df["Miles"]>=100))].
 ↪shape[0]/df[((df["Miles"]<=200) & (df["Miles"]>=100))].shape[0]
```

[56]: 0.22388059701492538

[57]:
```
#Probability of Miles <100  given that the product is KP481
df[(df["Product"] =="KP481")&(df["Miles"]<=100)].shape[0]/
 ↪df[(df["Miles"]<=200)].shape[0]
```

[57]: 0.25287356321839083

[58]:
```
#Probability of Miles > 100 and Miles<200 given that the product is KP281
df[(df["Product"] =="KP281")&((df["Miles"]<=200) & (df["Miles"]>=100))].
 ↪shape[0]/df[((df["Miles"]<=200) & (df["Miles"]>=100))].shape[0]
```

[58]: 0.26865671641791045

[59]:
```
#Probability of Miles < 100  given that the product is KP281
df[(df["Product"] =="KP281")&(df["Miles"]<=100)].shape[0]/
 ↪df[(df["Miles"]<=200)].shape[0]
```

[59]: 0.3563218390804598

[60]:
```
df.head(5)
```

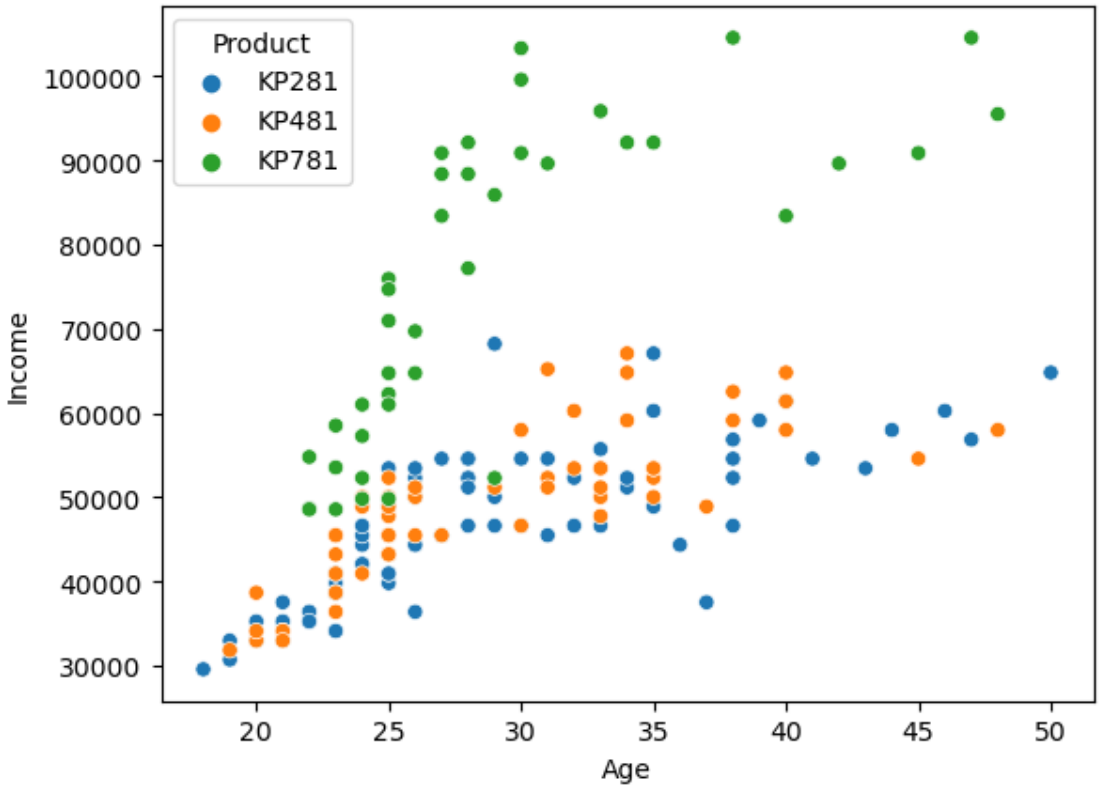
[60]:

	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

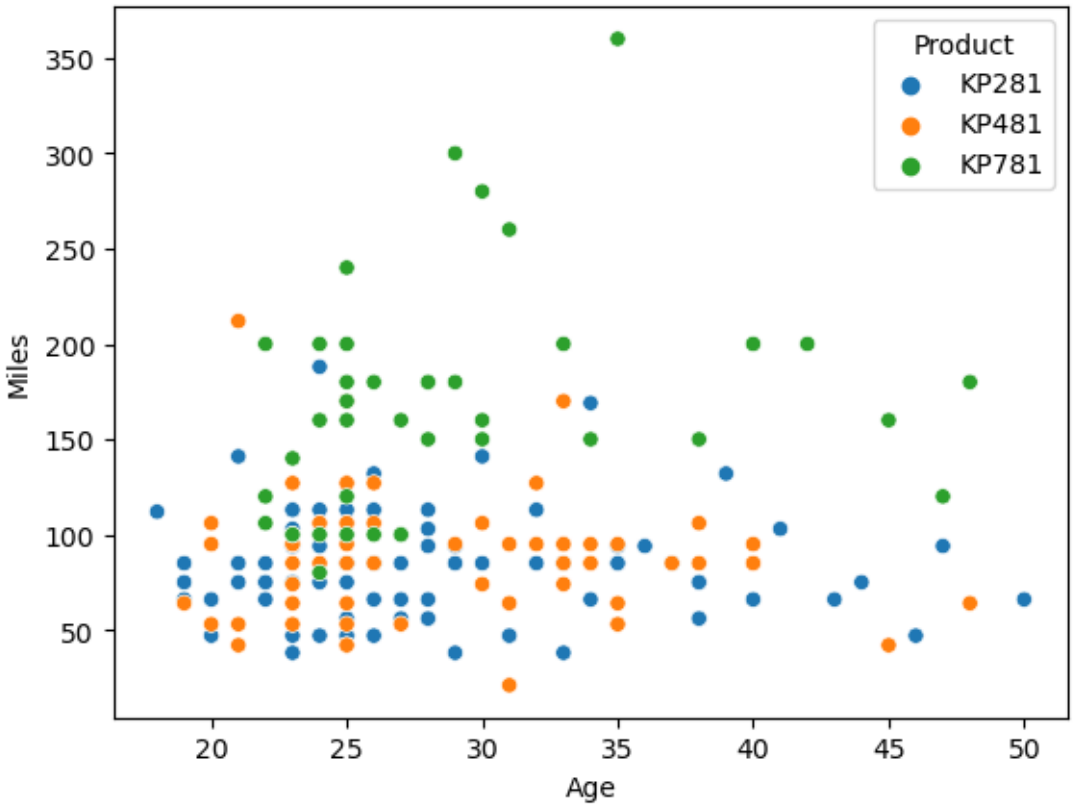
[61]:
```
sns.scatterplot(data=df, x="Age",y="Income",hue="Product")
```

[61]: <AxesSubplot:xlabel='Age', ylabel='Income'>



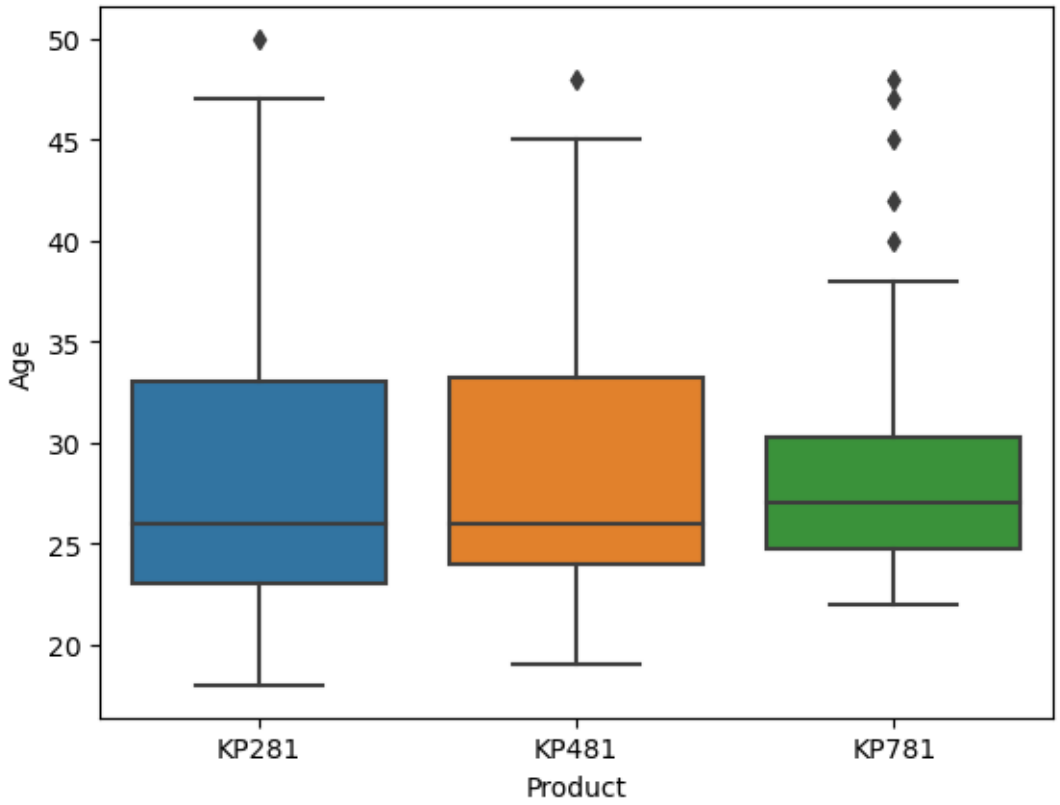
[62]:
```
sns.scatterplot(data=df, x="Age",y="Miles",hue="Product")
```

[62]: <AxesSubplot:xlabel='Age', ylabel='Miles'>



[63]:
```
sns.boxplot(data=df, x="Product",y="Age")
```

[63]: <AxesSubplot:xlabel='Product', ylabel='Age'>



5 Conditional Probabilites of product given Age

[64]:
```
# Probabililty of buying product KP781 given age greater than 30
df[(df["Product"] =="KP781")&(df["Age"]>=23)].shape[0]/df[(df["Age"]>=23)].
  ↪shape[0]
```

[64]: 0.23717948717948717

[65]:
```
# Probabililty of buying product KP781 given age lesser than 30
df[(df["Product"] =="KP781")&(df["Age"]<30)].shape[0]/df[(df["Age"]<30)].
  ↪shape[0]
```

[65]: 0.23893805309734514

[66]:
```
# Probabililty of buying product KP481 given age lesser than 30
df[(df["Product"] =="KP481")&(df["Age"]<30)].shape[0]/df[(df["Age"]<30)].
  ↪shape[0]
```

[66]: 0.2920353982300885

[67]:
```
# Probabililty of buying product KP281 given age lesser than 30
df[(df["Product"] =="KP281")&(df["Age"]>30)].shape[0]/df[(df["Age"]>30)].
 ↪shape[0]
```

[67]:
```
0.4166666666666667
```

[68]:
```
df.head(4)
```

[68]:

	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

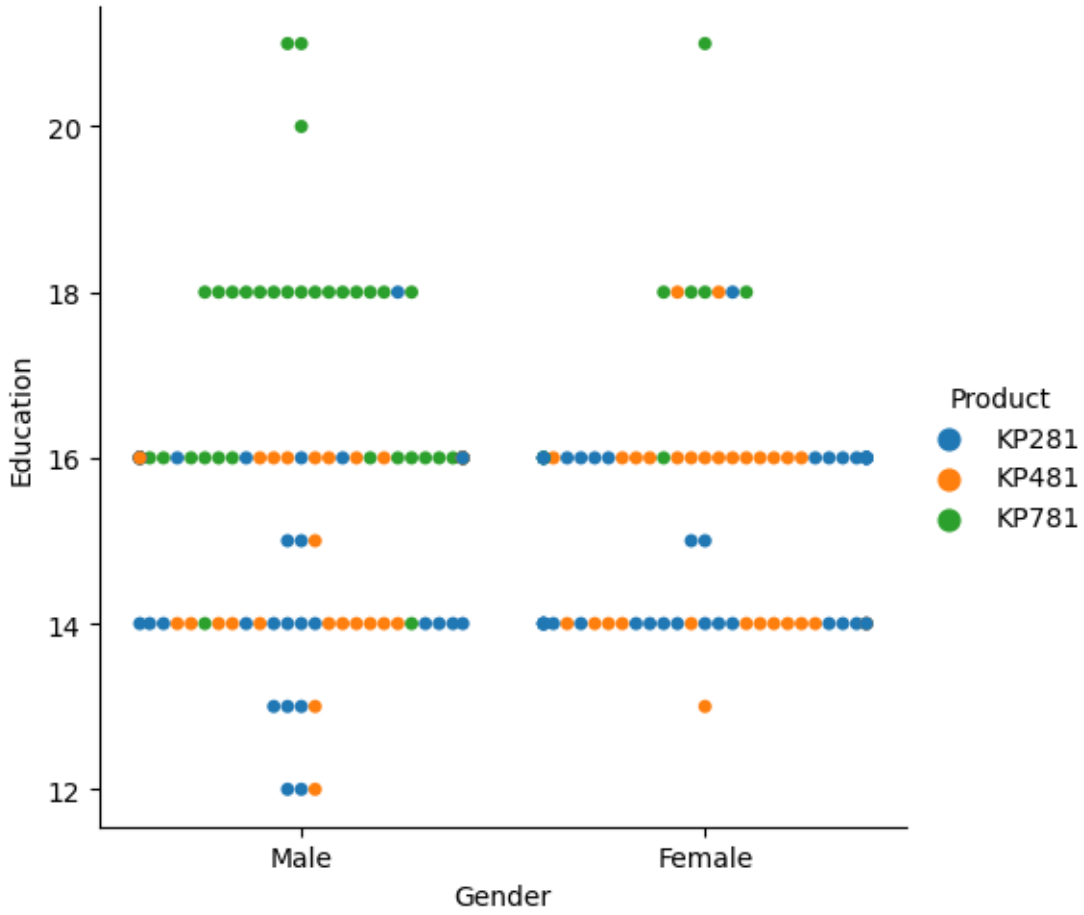
[69]:
```
df["Education"].value_counts()
```

[69]:
```
16    85
14    55
18    23
15     5
13     5
12     3
21     3
20     1
Name: Education, dtype: int64
```

[70]:
```
sns.catplot(data=df,x="Gender",y="Education",hue = "Product",kind="swarm")
```

```
c:\Users\revan\anaconda3\lib\site-packages\seaborn\categorical.py:1296:
UserWarning: 27.9% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\revan\anaconda3\lib\site-packages\seaborn\categorical.py:1296:
UserWarning: 25.0% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

[70]:
```
<seaborn.axisgrid.FacetGrid at 0x1fa26233e20>
```



6 Confitional Probability of products based of Education

[71]:
```
# Probability of product KP281 Given Education greater than 18
df[(df["Product"] == "KP281") & (df["Education"]>=18)].shape[0] /␣
 ↪df[df["Education"]>=18].shape[0]
```

[71]: 0.07407407407407407

[72]:
```
# Probability of product KP481 Given Education greater than 18
df[(df["Product"] == "KP481") & (df["Education"]>=18)].shape[0] /␣
 ↪df[df["Education"]>=18].shape[0]
```

[72]: 0.07407407407407407

[73]:
```
# Probability of product KP781 Given Education greater than 18
df[(df["Product"] == "KP781") & (df["Education"]>=16)].shape[0] /␣
 ↪df[df["Education"]>=16].shape[0]
```

[73]: 0.3392857142857143

[74]:
```
# Probability of product KP281 Given Education lesser than 18
df[(df["Product"] == "KP281") & (df["Education"]<18)].shape[0] /␣
 ↪df[df["Education"]<18].shape[0]
```

[74]: 0.5098039215686274

[75]:
```
# Probability of product KP281 Given Education between 12 to 16
df[(df["Product"] == "KP281") & (df["Education"]<16) & (df["Education"]>12)].
 ↪shape[0] / df[(df["Education"]<16) & (df["Education"]>12)].shape[0]
```

[75]: 0.5692307692307692

[76]:
```
# Probability of product KP481 Given Education between 14 to 18
df[(df["Product"] == "KP281") & (df["Education"]<18) & (df["Education"]>14)].
 ↪shape[0] / df[(df["Education"]<18) & (df["Education"]>14)].shape[0]
```

[76]: 0.4777777777777778

[77]:
```
df.head(4)
```

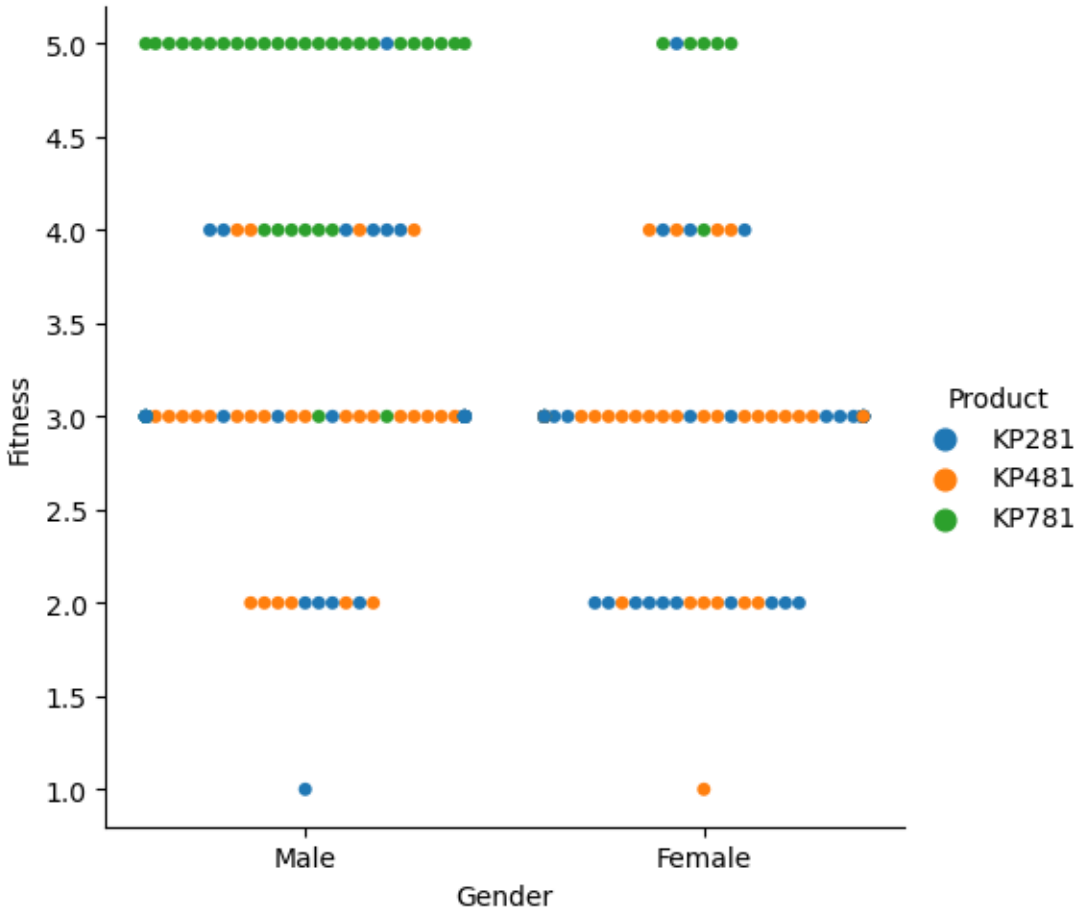
[77]:

	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

[78]:
```
sns.catplot(data=df,x="Gender",y="Fitness",hue = "Product",kind="swarm")
```

```
c:\Users\revan\anaconda3\lib\site-packages\seaborn\categorical.py:1296:
UserWarning: 29.8% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\revan\anaconda3\lib\site-packages\seaborn\categorical.py:1296:
UserWarning: 28.9% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

[78]: <seaborn.axisgrid.FacetGrid at 0x1fa24f4c730>



7 Coniditonal Probability of product based on fitness level

[79]:
```
#Probability of product KP781 given fitness level greater than 4
df[(df["Product"] == "KP781") & (df["Fitness"] >=4)].shape[0]/df[(df["Fitness"]␣
↪>=4)].shape[0]
```

[79]: 0.6545454545454545

[80]:
```
#Probability of product KP781 given fitness level == 5
df[(df["Product"] == "KP781") & (df["Fitness"] ==5)].shape[0]/df[(df["Fitness"]␣
↪==5)].shape[0]
```

[80]: 0.9354838709677419

[81]:
```
#Probability of product KP481 given fitness level == 3
df[(df["Product"] == "KP481") & (df["Fitness"] ==3)].shape[0]/df[(df["Fitness"]␣
↪==3)].shape[0]
```

[81]: 0.4020618556701031

[82]:
```
#Probability of product KP481 given fitness level <= 3
df[(df["Product"] == "KP481") & (df["Fitness"] <=3)].shape[0]/df[(df["Fitness"]␣
↪<=3)].shape[0]
```

[82]: 0.416

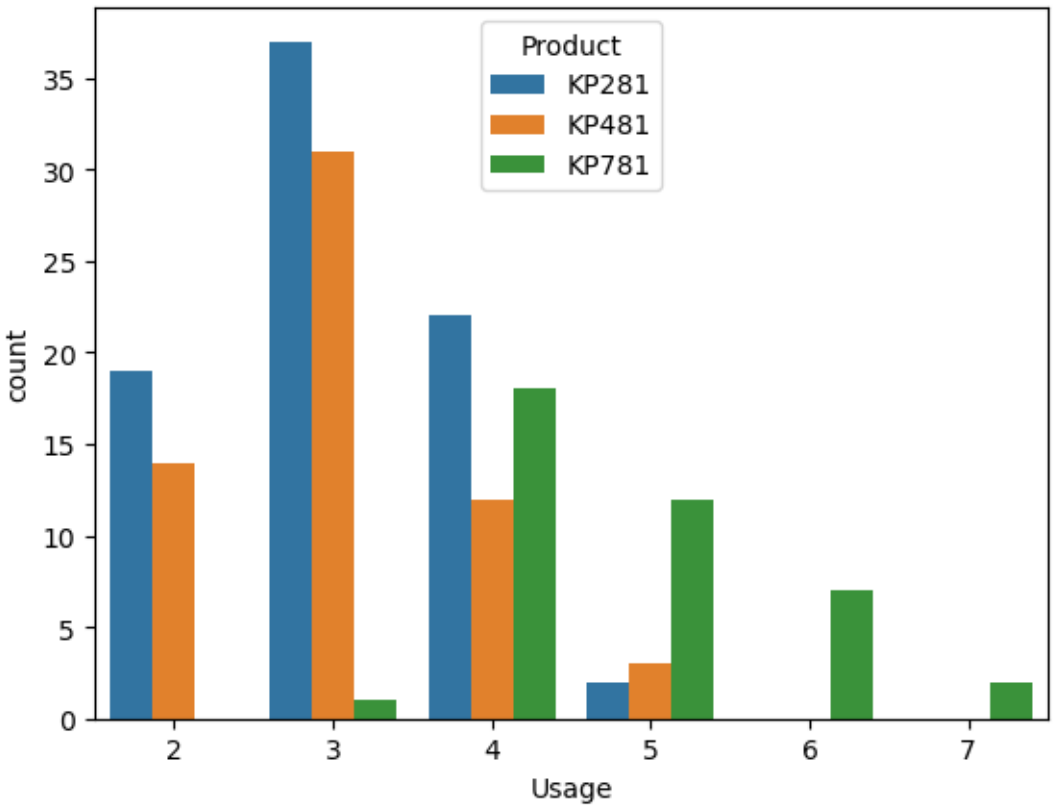
[83]:
```
#Probability of product KP281 given fitness level <= 3
df[(df["Product"] == "KP281") & (df["Fitness"] <=3)].shape[0]/df[(df["Fitness"]␣
↪<=3)].shape[0]
```

[83]: 0.552

8 Conditonal Probability based on Usage

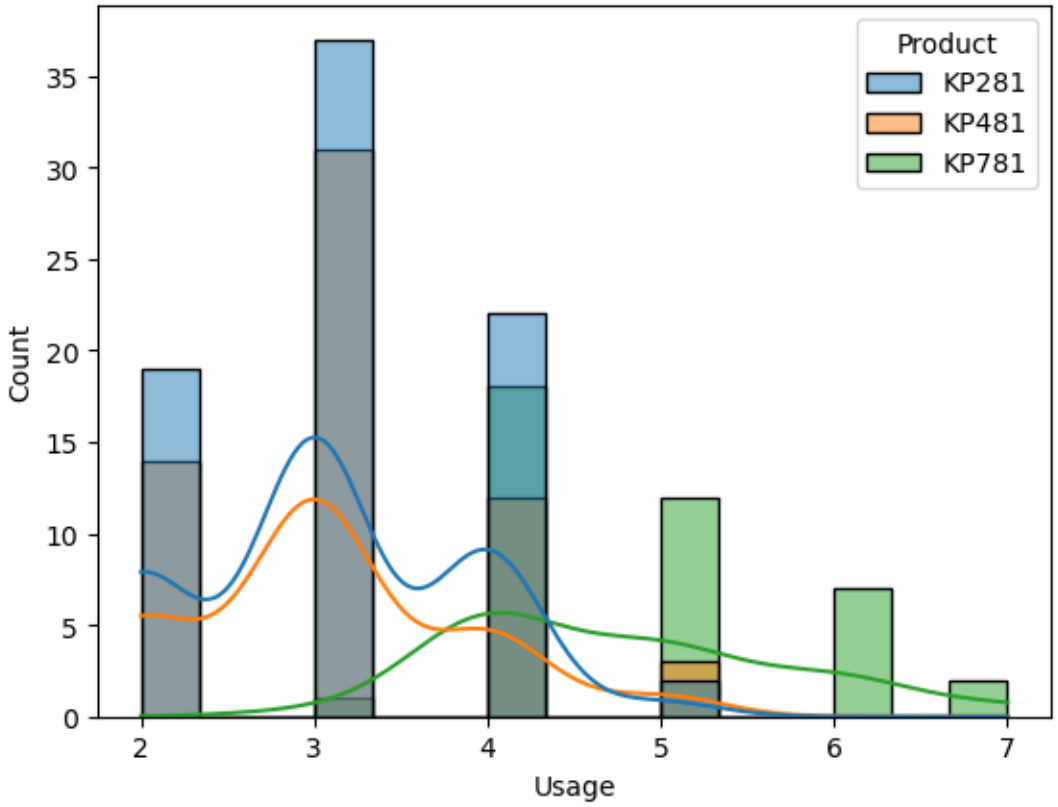
[85]:
```
sns.countplot(data=df,x="Usage",hue="Product")
```

[85]: <AxesSubplot:xlabel='Usage', ylabel='count'>



[86]:
```
sns.histplot(data=df,x="Usage",hue = "Product", kde=True,color="cyan")
```

[86]: <AxesSubplot:xlabel='Usage', ylabel='Count'>



[87]:
```
#Probability of usage greater than 4 times a week given product KP781
df[(df["Usage"] >= 4) & (df["Product"] == "KP781")].shape[0] /df[df["Product"]␣
↪== "KP781"].shape[0]
```

[87]: 0.975

[88]:
```
#Probability of usage lesser than 4 times a week given product KP281
df[(df["Usage"] <= 4) & (df["Product"] == "KP281")].shape[0] /df[df["Product"]␣
↪== "KP281"].shape[0]
```

[88]: 0.975

[89]:
```
#Probability of usage lesser than 3 times a week given product KP281
df[(df["Usage"] <= 3) & (df["Product"] == "KP281")].shape[0] /df[df["Product"]␣
↪== "KP281"].shape[0]
```

[89]: 0.7

[90]:
```
#Probability of usage lesser than 4 times a week given product KP481
df[(df["Usage"] <= 4) & (df["Product"] == "KP481")].shape[0] /df[df["Product"]
 ↪== "KP481"].shape[0]
```

[90]:
```
0.95
```

[91]:
```
df.head()
```

[91]:

	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

[92]:
```
df.columns
```

[92]:
```
Index(['Product', 'Age', 'Gender', 'Education', 'MaritalStatus', 'Usage',
       'Fitness', 'Income', 'Miles'],
      dtype='object')
```

9 Categorization

[93]:
```
def category(fitness):
    if fitness>=4:
        return "Cardio_Uses"
    elif fitness>2 and fitness<4:
        return "Weight_Loss"
    else:
        return "General"
```

[101]:
```
df["Category"] = df["Fitness"].apply(category)
```

[103]:
```
df.head()
```

[103]:

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

```
4     47      General
```

[94]:
```
#Cardio_Uses
df[df["Fitness"] >= 4].groupby(["Product"])["Usage","Miles","Age"].mean()
```

```
C:\Users\revan\AppData\Local\Temp\ipykernel_12560\3585738980.py:2:
FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of
keys) will be deprecated, use a list instead.
  df[df["Fitness"] >= 4].groupby(["Product"])["Usage","Miles","Age"].mean()
```

[94]:

Product	Usage	Miles	Age
KP281	3.727273	129.909091	29.000000
KP481	3.500000	125.875000	27.625000
KP781	4.861111	174.166667	29.555556

[95]:
```
df[df["Fitness"] >= 4].
↪groupby(["Product","Gender","MaritalStatus"])["Usage","Miles","Age"].mean()
```

```
C:\Users\revan\AppData\Local\Temp\ipykernel_12560\2600877475.py:1:
FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of
keys) will be deprecated, use a list instead.
  df[df["Fitness"] >=
4].groupby(["Product","Gender","MaritalStatus"])["Usage","Miles","Age"].mean()
```

[95]:

Product	Gender	MaritalStatus	Usage	Miles	Age
KP281	Female	Partnered	4.000000	150.500000	25.000000
		Single	3.000000	94.000000	38.000000
	Male	Partnered	4.000000	135.000000	31.666667
		Single	3.750000	133.750000	24.500000
KP481	Female	Partnered	5.000000	212.000000	21.000000
		Single	3.333333	109.333333	28.000000
	Male	Partnered	3.333333	120.333333	30.666667
		Single	3.000000	106.000000	24.000000
KP781	Female	Partnered	5.250000	215.000000	29.000000
		Single	5.000000	150.000000	23.500000
	Male	Partnered	4.888889	180.555556	30.277778
		Single	4.666667	155.000000	29.666667

[96]:
```
df[df["Fitness"] >= 4].
↪groupby(["Product","Gender","MaritalStatus"])["Usage","Miles","Age"].median()
```

```
C:\Users\revan\AppData\Local\Temp\ipykernel_12560\997112317.py:1: FutureWarning:
Indexing with multiple keys (implicitly converted to a tuple of keys) will be
deprecated, use a list instead.
  df[df["Fitness"] >=
4].groupby(["Product","Gender","MaritalStatus"])["Usage","Miles","Age"].median()
```

[96]:
```
                                Usage  Miles   Age
Product Gender MaritalStatus
KP281   Female Partnered          4.0  150.5  25.0
               Single            3.0   94.0  38.0
        Male   Partnered          4.0  132.0  30.0
               Single            3.5  127.0  23.0
KP481   Female Partnered          5.0  212.0  21.0
               Single            3.0  127.0  26.0
        Male   Partnered          3.0  106.0  33.0
               Single            3.0  106.0  24.0
KP781   Female Partnered          5.5  200.0  29.0
               Single            5.0  150.0  23.5
        Male   Partnered          4.5  165.0  27.5
               Single            5.0  155.0  26.0
```

[107]:
```
df.groupby("Category")["Product"].value_counts()
```

[107]:
```
Category     Product
Cardio_Uses  KP781      36
             KP281      11
             KP481       8
General      KP281      15
             KP481      13
Weight_Loss  KP281      54
             KP481      39
             KP781       4
Name: Product, dtype: int64
```

[97]:
```
#Weight_loss
df[(df["Fitness"] >= 2)&(df["Fitness"]<4)].
 ↪groupby(["Product","Gender","MaritalStatus"])["Usage","Miles","Age"].mean()
```

```
C:\Users\revan\AppData\Local\Temp\ipykernel_12560\3326204159.py:2:
FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of
keys) will be deprecated, use a list instead.
  df[(df["Fitness"] >= 2)&(df["Fitness"]<4)].groupby(["Product","Gender","Marita
lStatus"])["Usage","Miles","Age"].mean()
```

[97]:
```
                                Usage       Miles        Age
Product Gender MaritalStatus
KP281   Female Partnered     2.760000   68.880000  28.600000
               Single       3.000000   76.090909  27.000000
        Male   Partnered     3.176471   72.470588  31.823529
               Single       3.133333   90.400000  25.933333
KP481   Female Partnered     3.214286   85.571429  30.642857
               Single       2.900000   77.400000  27.900000
        Male   Partnered     2.777778   81.722222  30.333333
               Single       3.333333   89.444444  25.333333
```

```
KP781   Female Single          4.000000  100.000000  26.000000
        Male   Partnered       4.000000  100.000000  25.000000
               Single          4.000000  103.000000  24.500000
```

[98]:
```
df[(df["Fitness"] >= 2)&(df["Fitness"]<4)].
 ↪groupby(["Product","Gender","MaritalStatus"])["Usage","Miles","Age"].median()
```

```
C:\Users\revan\AppData\Local\Temp\ipykernel_12560\1730690076.py:1:
FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of
keys) will be deprecated, use a list instead.
  df[(df["Fitness"] >= 2)&(df["Fitness"]<4)].groupby(["Product","Gender","Marita
lStatus"])["Usage","Miles","Age"].median()
```

[98]:

Product	Gender	MaritalStatus	Usage	Miles	Age
KP281	Female	Partnered	3.0	66.0	27.0
		Single	3.0	75.0	24.0
	Male	Partnered	3.0	75.0	31.0
		Single	3.0	85.0	25.0
KP481	Female	Partnered	3.0	85.0	32.0
		Single	3.0	79.5	25.0
	Male	Partnered	3.0	90.0	28.0
		Single	3.0	95.0	25.0
KP781	Female	Single	4.0	100.0	26.0
	Male	Partnered	4.0	100.0	25.0
		Single	4.0	103.0	24.5

[99]:
```
#General Purpose
df[df["Fitness"] <= 2].
 ↪groupby(["Product","Gender","MaritalStatus"])["Usage","Miles","Age","Fitness"].
 ↪mean()
```

```
C:\Users\revan\AppData\Local\Temp\ipykernel_12560\319462389.py:2: FutureWarning:
Indexing with multiple keys (implicitly converted to a tuple of keys) will be
deprecated, use a list instead.
  df[df["Fitness"] <= 2].groupby(["Product","Gender","MaritalStatus"])["Usage","
Miles","Age","Fitness"].mean()
```

[99]:

Product	Gender	MaritalStatus	Usage	Miles	Age	Fitness
KP281	Female	Partnered	2.428571	49.857143	28.142857	2.0
		Single	2.000000	50.333333	32.666667	2.0
	Male	Partnered	2.800000	47.000000	24.800000	1.8
KP481	Female	Partnered	3.000000	63.500000	29.000000	2.0
		Single	2.600000	53.000000	27.600000	1.8
	Male	Partnered	2.200000	46.400000	29.400000	2.0
		Single	4.000000	53.000000	27.000000	2.0

10 Actionable Insights

[100]:
```
# KP781 was bought by customers who was earning above 65k annual, KP481 was␣
 ↪bought by customers earning in range of 40 to 65k
# KP281 was bought by below 40k income users
# Those who are using treadmill above 4 times per week are having great fitness␣
 ↪in the scale of 4 - 5, we can consider them as cardio_uses, fitness people
# Some people using treadmill in the range of 2-4 and they are having fitness␣
 ↪scale in the average of 3-4, we can consider them as weight loss people
# Some People are not that much interested in treadmill, they also have very␣
 ↪low fitness scale, they all under general purpose
# Majority of people or buying KP281 and KP241
# Those who bought KP781 are seriously using their treadmill and getting fitted␣
 ↪by using treadmill 5 times per week and they used to walk above 150 miles␣
 ↪per week
# Customers earning above 65k was highly possible of buying KP781
# We can split customers in 3 buckets based on their Use cases.
# Cardio_uses,                  Weight_loss,                       General purpose.
#  Income>60000               Income between 40k to 60k          Income < 30k
#  Usage >= 4/week               3 > Usage>4                         Usage<=3
# The products is highly correlated with many variables like usage, income,␣
 ↪fitness, education
```

11 Business Recommendations

[ ]:
```
# Aerofit can sell KP781 whose are earning above 60k Annual income
# Based on their customer profile we can sell products easily, for general␣
 ↪purpose we can promote KP281, For weight_loss we can promote KP481,
# Fitness freak people always used to buy KP781, we can promote based on their␣
 ↪fitness, usage.
# If the customers are Married there is a high chance of selling KP481 and KP781
# In an online portal we can create filters for usage and fitness, Based on the␣
 ↪filters selected by customers we can promote them any of the particular␣
 ↪product.
```