Aerofit_CaseStudy : Probability analysis

by Pavan Kumaar

Customer Profiling for Aerofit Threadmill products KP281, KP481, KP781

Importing required libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm import tqdm
import plotly.express as px
import plotly.graph_objects as go
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)

import warnings
warnings.filterwarnings("ignore")
```

Reading data file in CSV

In [2]:
```
data=pd.read_csv('aerofit_treadmill.csv')
data.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 [3]:
```
#Describing the dataframe
data.describe()
```

Out[3]:

	Age	Education	Usage	Fitness	Income	Miles
count	180.000000	180.000000	180.000000	180.000000	180.000000	180.000000
mean	28.788889	15.572222	3.455556	3.311111	53719.577778	103.194444
std	6.943498	1.617055	1.084797	0.958869	16506.684226	51.863605
min	18.000000	12.000000	2.000000	1.000000	29562.000000	21.000000
25%	24.000000	14.000000	3.000000	3.000000	44058.750000	66.000000
50%	26.000000	16.000000	3.000000	3.000000	50596.500000	94.000000
75%	33.000000	16.000000	4.000000	4.000000	58668.000000	114.750000
max	50.000000	21.000000	7.000000	5.000000	104581.000000	360.000000

In [4]:

```
# DAtaframe has 9 columns and 180 r0ws
data.shape
```

Out[4]:

```
(180, 9)
```

In [5]:

```
data.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]:

```
# Checking Outliers
data.isna().sum()
```

Out[6]:

```
Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
Usage            0
Fitness          0
Income           0
Miles            0
dtype: int64
```

In [7]:

```
#Checking Duplicates
data.duplicated().sum()
```

Out[7]:

```
0
```

Seeing gist of data and its properties

In [8]:
```
data.head(10)
```

Out[8]:

	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	KP281	20	Female	14	Partnered	3	3	32973	66
6	KP281	21	Female	14	Partnered	3	3	35247	75
7	KP281	21	Male	13	Single	3	3	32973	85
8	KP281	21	Male	15	Single	5	4	35247	141
9	KP281	21	Female	15	Partnered	2	3	37521	85

In [9]:
```
data['Gender'].value_counts()
```

Out[9]:
```
Male      104
Female     76
Name: Gender, dtype: int64
```

In [10]:
```
data['Product'].value_counts()
```

Out[10]:
```
KP281    80
KP481    60
KP781    40
Name: Product, dtype: int64
```

In [11]:
```
data['Age'].unique()
```

Out[11]:
```
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, 43, 44, 46, 47, 50, 45, 48, 42],
      dtype=int64)
```

In [12]:
```
data['MaritalStatus'].value_counts()
```

Out[12]:
```
Partnered    107
Single        73
Name: MaritalStatus, dtype: int64
```

In [13]:
```
data['Fitness'].value_counts()
```

Out[13]:
```
3    97
5    31
2    26
4    24
1     2
Name: Fitness, dtype: int64
```

In [14]:
```
data['Usage'].value_counts().sort_index()
```

Out[14]:
```
2    33
3    69
4    52
5    17
6     7
7     2
Name: Usage, dtype: int64
```

In [15]:
```
data_cat=data
data_cat['Fitness_level'] = data.Fitness
data_cat["Fitness_level"].replace({1:"Poor",
                                   2:"Below_Avg",
                                   3:"Avg",
                                   4:"Good",
                                   5:"Excellent"},inplace=True)
data_cat.head()
```

Out[15]:

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

Distribution of Data based on Gender, Marital Status, Product Type

In [16]:
```
# Product percentage
prodser = data['Product'].value_counts(normalize=True)
prodstat = prodser.map(lambda calc: round(100*calc,2))
prodstat
```

Out[16]:
```
KP281    44.44
KP481    33.33
KP781    22.22
Name: Product, dtype: float64
```

In [17]:
```
# Gender statistics
gender = data['Gender'].value_counts(normalize=True)
gender_ser = gender.map(lambda calc: round(100*calc,2))
gender_ser
```

Out[17]:
```
Male      57.78
Female    42.22
Name: Gender, dtype: float64
```

In [18]:
```
#Marital Status
marital_status = data['MaritalStatus'].value_counts(normalize=True)
marital_status_ser = marital_status.map(lambda calc:round(100*calc,2))
marital_status_ser
```

Out[18]:
```
Partnered    59.44
Single       40.56
Name: MaritalStatus, dtype: float64
```

In [19]:
```
# Usage: Number of days used in week
usage = data['Usage'].value_counts(normalize=True).map(lambda calc:round(100*calc,
usage.rename(columns={'index':'DaysinWeek'},inplace=True)
usage
```

Out[19]:

	DaysinWeek	Usage
0	3	38.33
1	4	28.89
2	2	18.33
3	5	9.44
4	6	3.89
5	7	1.11

In [20]:

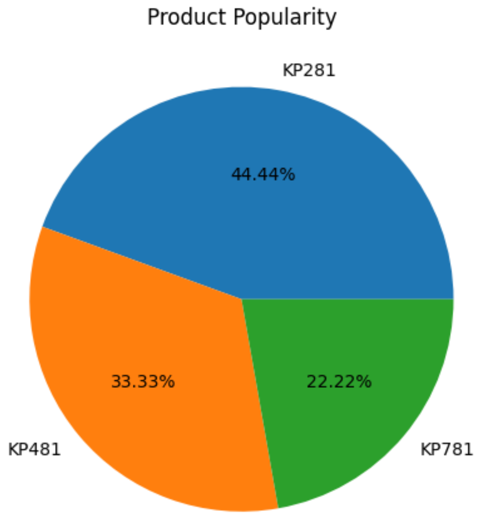
```
# Customer rating percentage
rating = data['Fitness'].value_counts(normalize=True).map(lambda calc:round(100*ca
rating.rename(columns={'index':'Rating'},inplace=True)
rating
```

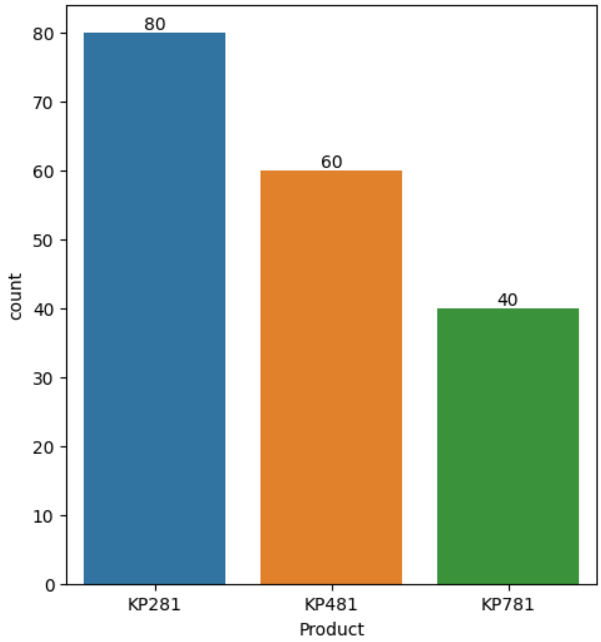
Out[20]:

	Rating	Fitness
0	3	53.89
1	5	17.22
2	2	14.44
3	4	13.33
4	1	1.11

In [21]:

```
# Comparison of Product
x = data.groupby(['Product'])['Product'].count()
y = len(data)
r = ((x/y) * 100).round(2)
mf_ratio = pd.DataFrame(r)
mf_ratio.rename({'Product': '%'}, axis=1, inplace=True)

plt.figure(figsize=(12, 6))
plt.subplot(1,2,1)
plt.pie(mf_ratio['%'], labels=mf_ratio.index,autopct='%1.2f%%')
plt.title('Product Popularity')

# Product count plot
plt.subplot(1,2,2)
ax = sns.countplot(data=data,x='Product')

ax.bar_label(ax.containers[0], label_type='edge')
plt.show


plt.show()
```





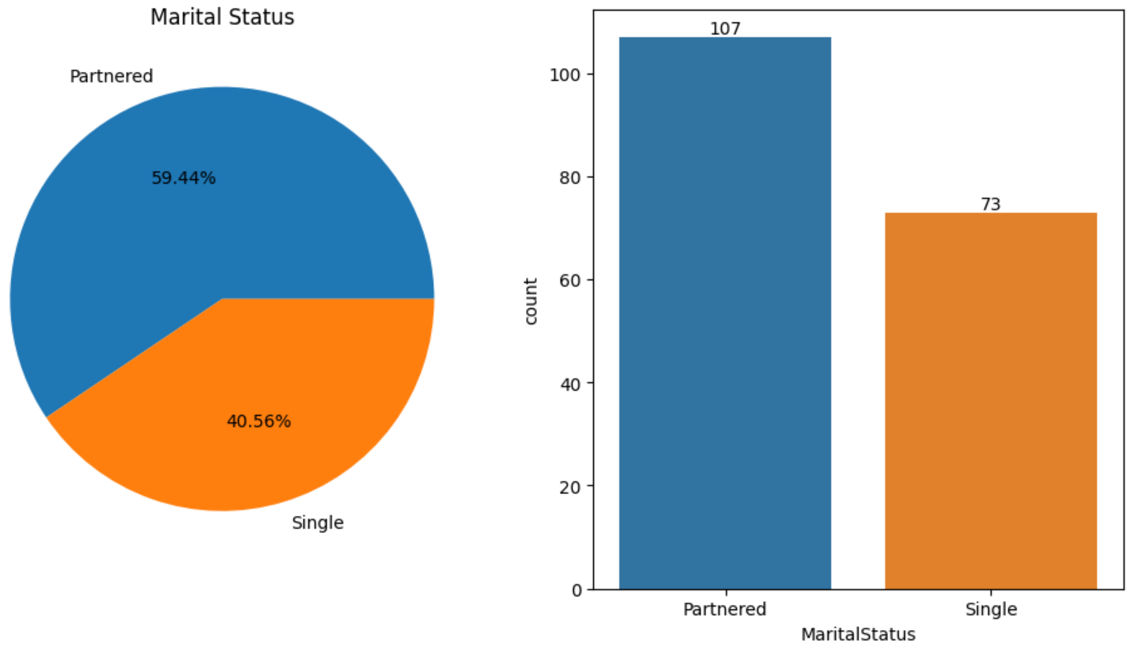
KP281 is the low end model which is popular

KP481 is the mid end model which is 33.33% of sold product in threadmill category.

KP781 is the high end model which is least sold but premium product in this category.

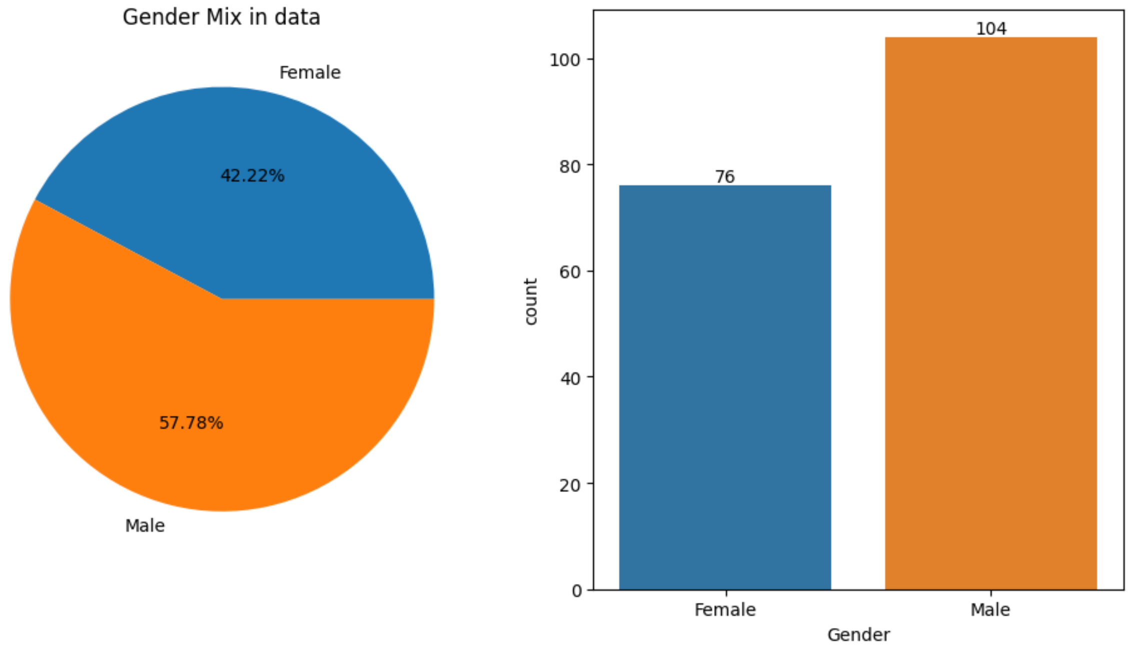
In [22]:

```
# Comparison of Marital Status
x = data.groupby(['MaritalStatus'])['MaritalStatus'].count()
y = len(data)
r = ((x/y) * 100).round(2)
mf_ratio = pd.DataFrame(r)
mf_ratio.rename({'MaritalStatus': '%'}, axis=1, inplace=True)

plt.figure(figsize=(12, 6))
plt.subplot(1,2,1)
plt.pie(mf_ratio['%'], labels=mf_ratio.index,autopct='%1.2f%%')
plt.title('Marital Status')

# Product count plot
plt.subplot(1,2,2)

ax = sns.countplot(data=data,x='MaritalStatus',order=['Partnered','Single'])

ax.bar_label(ax.containers[0], label_type='edge')
plt.show


plt.show()
```



Almost 60% of this product are preferred by Partnered customer over Single customer.

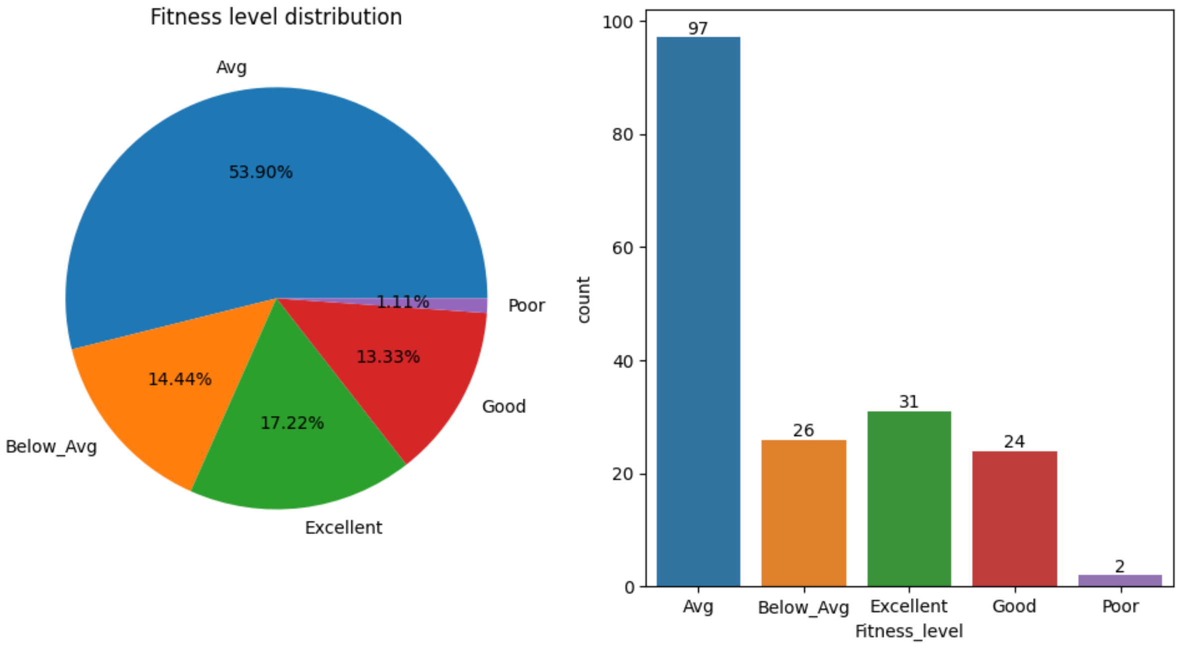
In [23]:
```
# Comparison of Gender
x = data.groupby(['Gender'])['Gender'].count()
y = len(data)
r = ((x/y) * 100).round(2)
mf_ratio = pd.DataFrame(r)
mf_ratio.rename({'Gender': '%'}, axis=1, inplace=True)

plt.figure(figsize=(12, 6))
plt.subplot(1,2,1)
plt.pie(mf_ratio['%'], labels=mf_ratio.index,autopct='%1.2f%%')
plt.title('Gender Mix in data')

# Product count plot
plt.subplot(1,2,2)
ax = sns.countplot(data=data,x='Gender',order=['Female','Male'])

ax.bar_label(ax.containers[0], label_type='edge')
plt.show


plt.show()
```



Almost 60% of this product are preferred by Male customer over Female customer.

In [24]:

```
# Comparison of Fitness Categroy
x = data_cat.groupby(['Fitness_level'])['Fitness_level'].count()
y = len(data_cat)
r = ((x/y) * 100).round(2)
mf_ratio = pd.DataFrame(r)
mf_ratio.rename({'Fitness_level': '%'}, axis=1, inplace=True)

plt.figure(figsize=(12, 6))
plt.subplot(1,2,1)
plt.pie(mf_ratio['%'], labels=mf_ratio.index,autopct='%1.2f%%')
plt.title('Fitness level distribution')

# Product count plot
plt.subplot(1,2,2)
ax = sns.countplot(data=data_cat,x='Fitness_level',order=['Avg','Below_Avg','Excel

ax.bar_label(ax.containers[0], label_type='edge')

plt.show


plt.show()
```



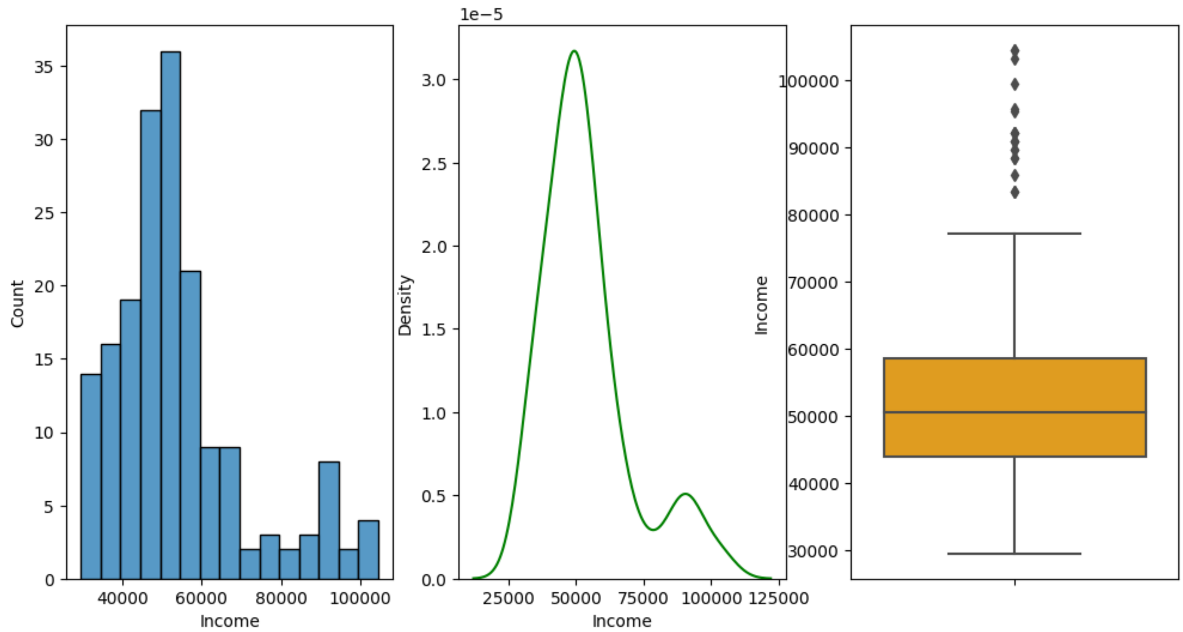
Almost 50% of the sales are by Avg fitness customers with fitness score 3

In [25]:

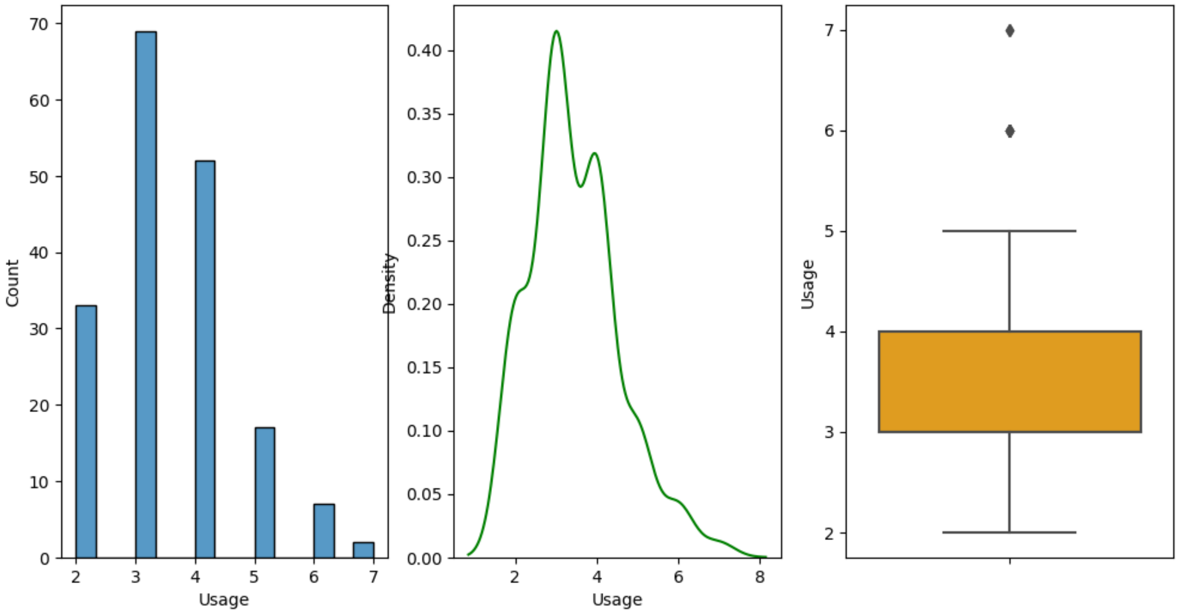
```
# Income Analysis

plt.figure(figsize=(12, 6))

plt.subplot(1,3,1)
sns.histplot(data.Income)
plt.subplot(1,3,2)
sns.kdeplot(data.Income,color='green')
plt.subplot(1,3,3)
sns.boxplot(data=data,y='Income', color='orange')


plt.show
```

Out[25]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



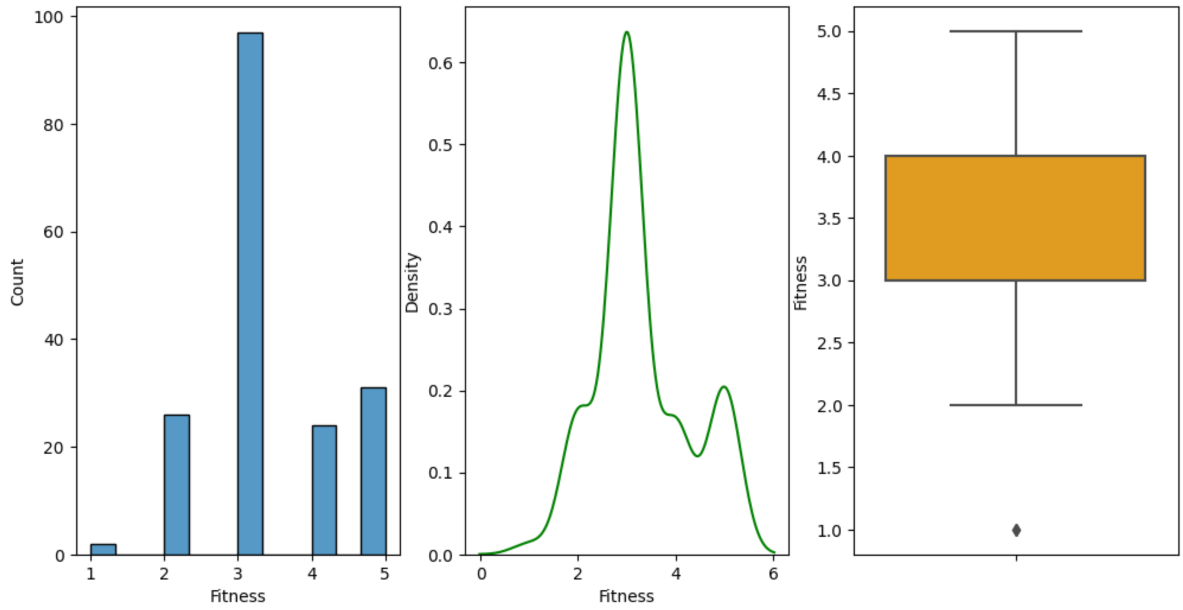
In [26]:
```python
# Usage analysis

plt.figure(figsize=(12, 6))

plt.subplot(1,3,1)
sns.histplot(data.Usage)
plt.subplot(1,3,2)
sns.kdeplot(data.Usage,color='green')
plt.subplot(1,3,3)
sns.boxplot(data=data,y='Usage', color='orange')
plt.show()
```



In [ ]:

In [27]:
```python
# Fitness Analysis

plt.figure(figsize=(12, 6))

plt.subplot(1,3,1)
sns.histplot(data.Fitness)
plt.subplot(1,3,2)
sns.kdeplot(data.Fitness,color='green')
plt.subplot(1,3,3)
sns.boxplot(data=data,y='Fitness', color='orange')


plt.show
```

Out[27]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```

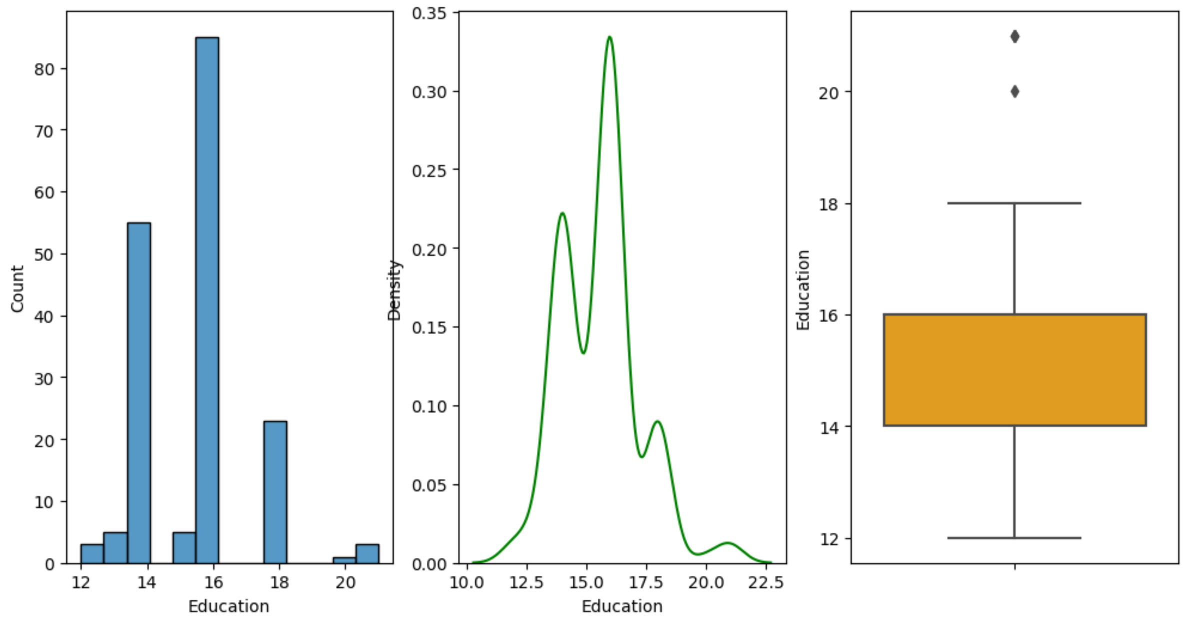


In [28]:

```
# Education Analysis

plt.figure(figsize=(12, 6))

plt.subplot(1,3,1)
sns.histplot(data.Education)
plt.subplot(1,3,2)
sns.kdeplot(data.Education,color='green')
plt.subplot(1,3,3)
sns.boxplot(data=data,y='Education', color='orange')


plt.show
```

Out[28]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```

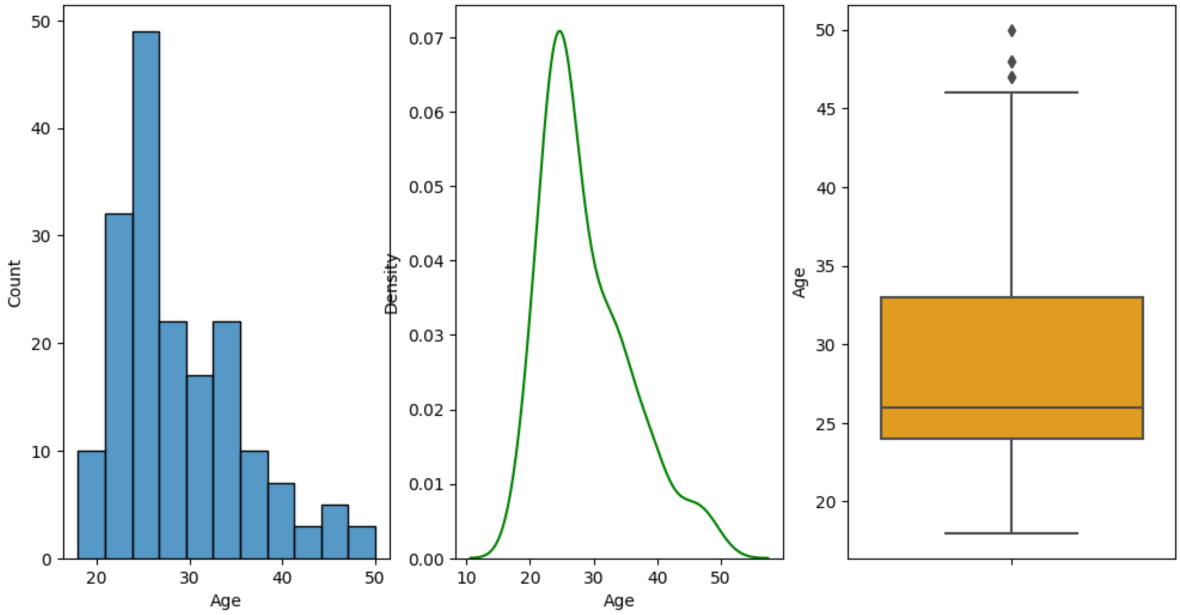


In [29]:

```
# Age Analysis

plt.figure(figsize=(12, 6))

plt.subplot(1,3,1)
sns.histplot(data.Age)
plt.subplot(1,3,2)
```

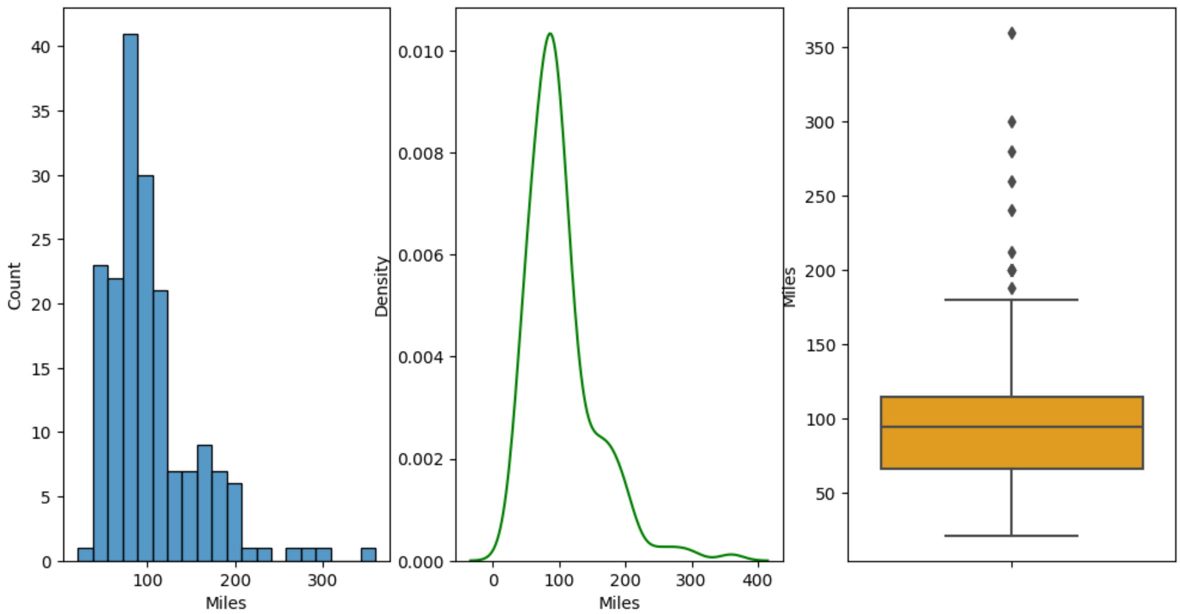
```
sns.kdeplot(data.Age,color='green')
plt.subplot(1,3,3)
sns.boxplot(data=data,y='Age', color='orange')

plt.show
```

Out[29]: <function matplotlib.pyplot.show(close=None, block=None)>

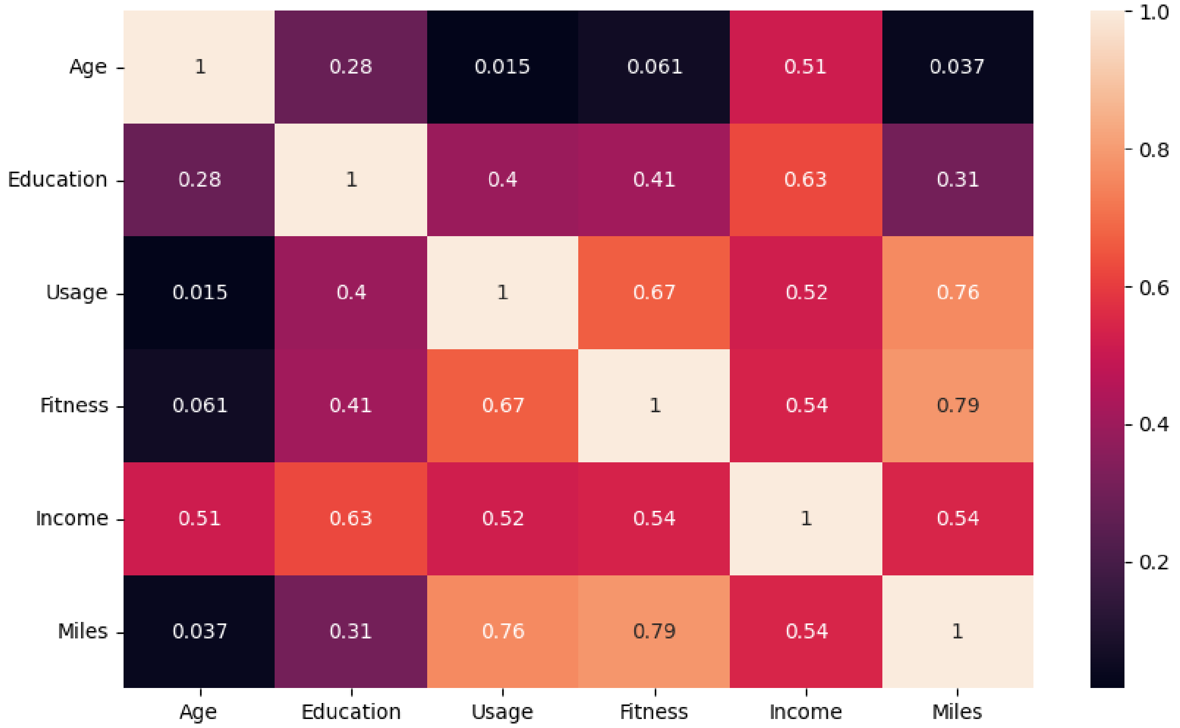


In [30]:
```
# Miles Analysis

plt.figure(figsize=(12, 6))

plt.subplot(1,3,1)
sns.histplot(data.Miles)
plt.subplot(1,3,2)
sns.kdeplot(data.Miles,color='green')
plt.subplot(1,3,3)
sns.boxplot(data=data,y='Miles', color='orange')

plt.show
```

Out[30]: <function matplotlib.pyplot.show(close=None, block=None)>



In [31]:
```
#Correlation HeatMap
plt.figure(figsize=(10,6))
ax = sns.heatmap(data.corr(),annot=True)
plt.yticks(rotation=0)
plt.show()
```



Usage-Miles, Fitness-Miles have more than 75% correlation followed by Fitness-Usage, Income-Education

In [32]:
```
data.groupby('Product')['Usage'].mean()
```

Out[32]:
```
Product
KP281    3.087500
KP481    3.066667
KP781    4.775000
Name: Usage, dtype: float64
```

In [33]:
```
data.groupby('Product')['Usage'].median()
```

Out[33]:
```
Product
KP281    3.0
KP481    3.0
KP781    5.0
Name: Usage, dtype: float64
```

In [34]:
```
data.groupby('Product')['Age'].mean()
```

Out[34]:
```
Product
KP281    28.55
KP481    28.90
KP781    29.10
Name: Age, dtype: float64
```

In [35]:
```
data.groupby('Product')['Age'].median()
```

Out[35]:
```
Product
KP281    26.0
KP481    26.0
KP781    27.0
Name: Age, dtype: float64
```

In [36]:
```
data.groupby('Product')['Education'].mean()
```

Out[36]:
```
Product
KP281    15.037500
KP481    15.116667
KP781    17.325000
Name: Education, dtype: float64
```

In [37]:
```
data.groupby('Product')['Education'].median()
```

Out[37]:
```
Product
KP281    16.0
KP481    16.0
KP781    18.0
Name: Education, dtype: float64
```

In [38]:
```
data.groupby('Product')['Fitness'].mean()
```

Out[38]:
```
Product
KP281    2.9625
KP481    2.9000
KP781    4.6250
Name: Fitness, dtype: float64
```

In [39]:
```
data.groupby('Product')['Fitness'].median()
```

Out[39]:
```
Product
KP281    3.0
KP481    3.0
KP781    5.0
Name: Fitness, dtype: float64
```

In [40]:
```
data.groupby('Product')['Income'].mean()
```

Out[40]:
```
Product
KP281    46418.025
KP481    48973.650
KP781    75441.575
Name: Income, dtype: float64
```

In [41]:
```
data.groupby('Product')['Income'].median()
```

Out[41]:
```
Product
KP281    46617.0
KP481    49459.5
KP781    76568.5
Name: Income, dtype: float64
```

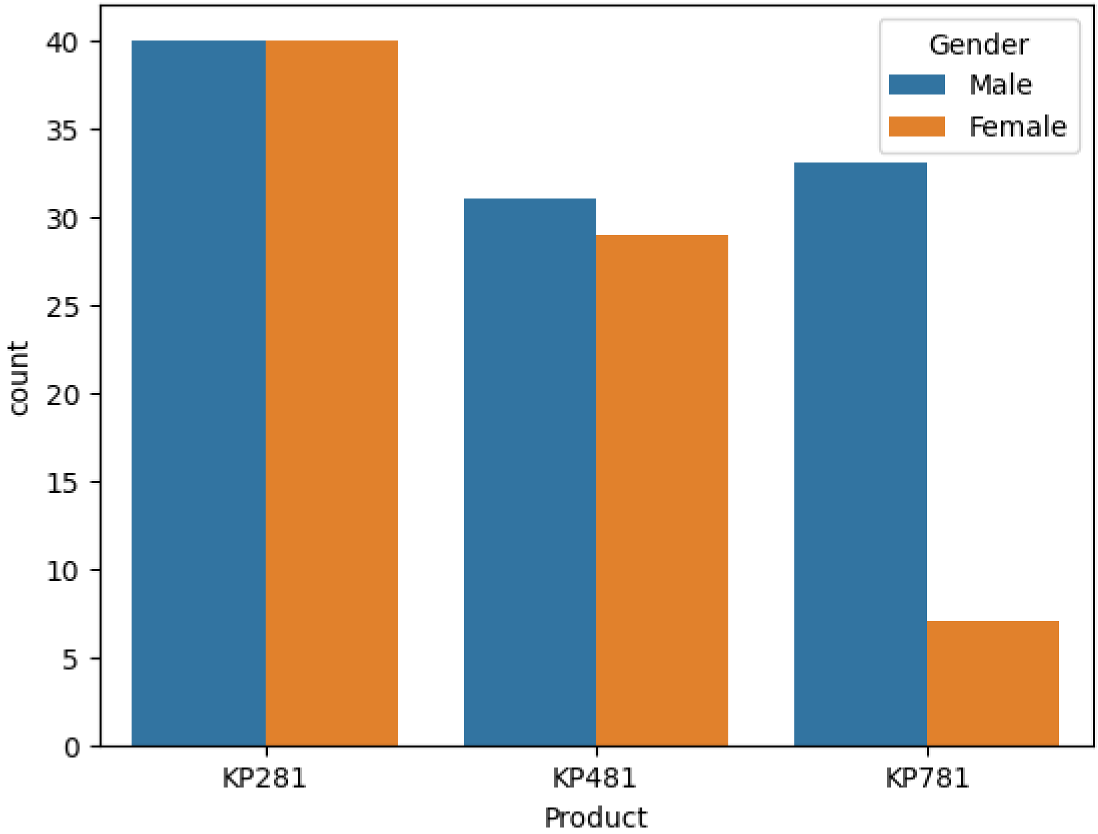
Mean and Median of the data are almost same, hence, it fits into Gaussian Distribution.

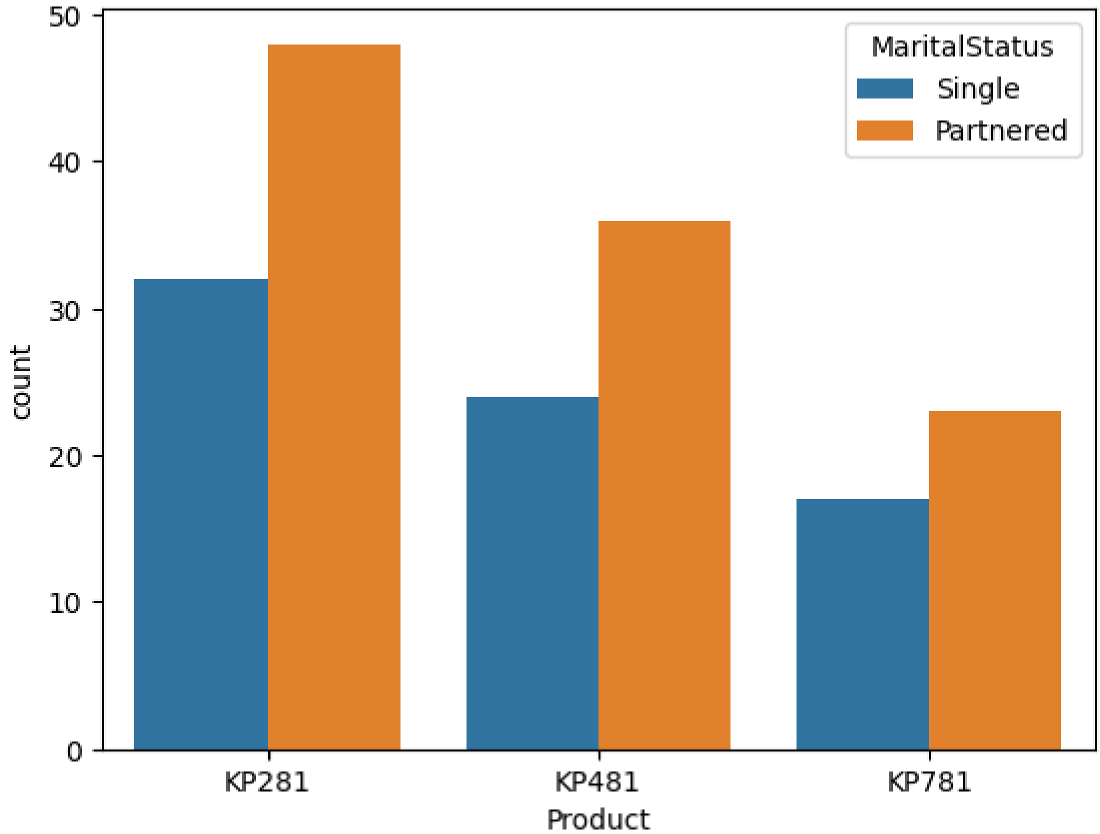
--Univariate Analysis--

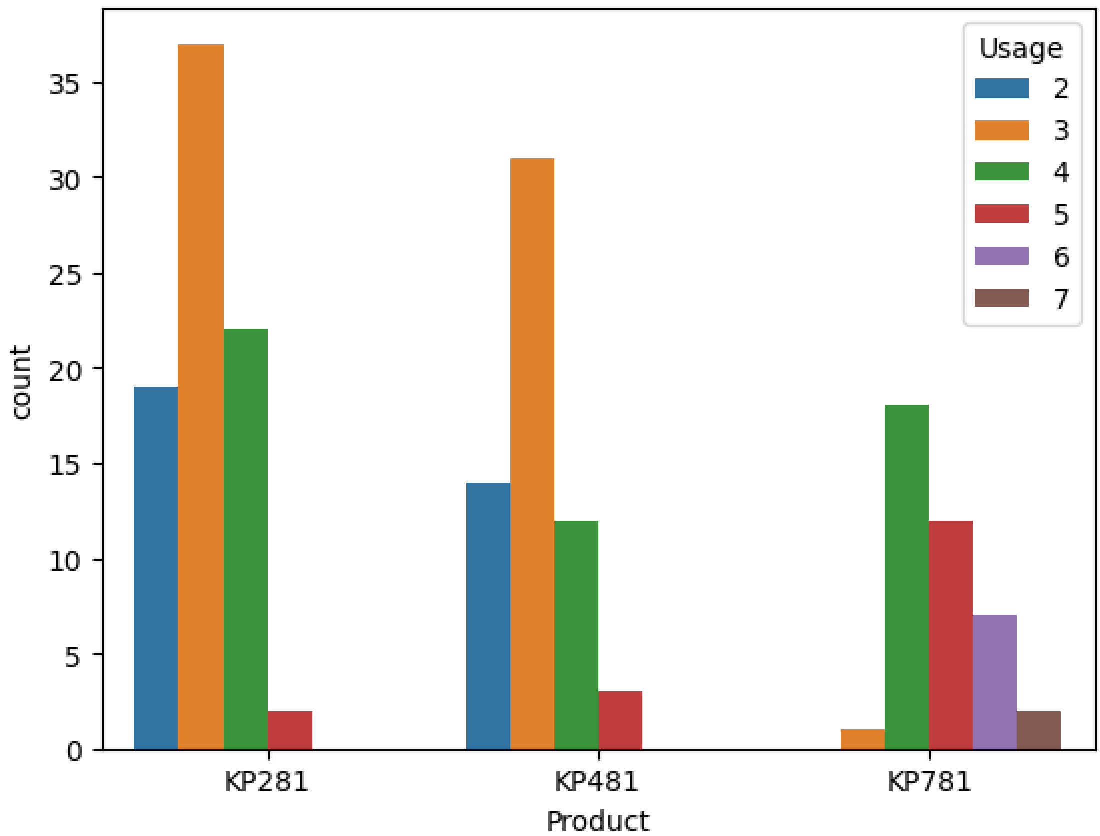
In [42]:
```
data.columns.tolist()
```

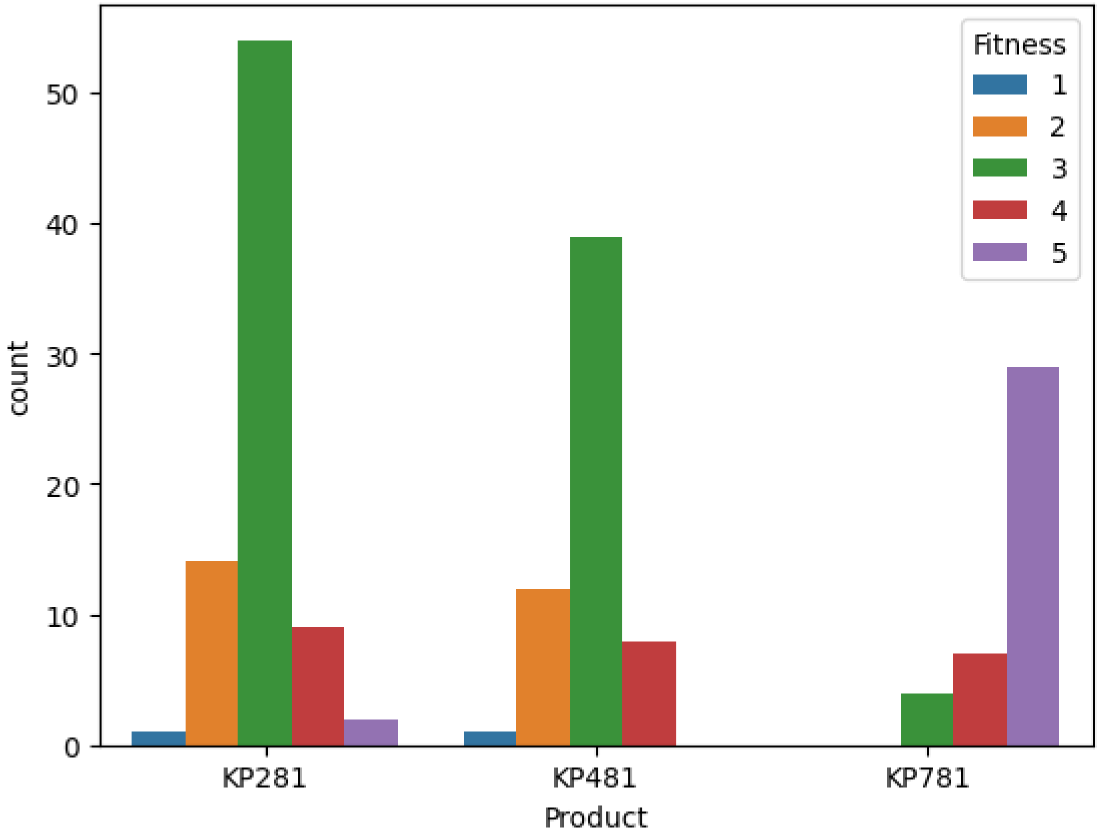
Out[42]:
```
['Product',
 'Age',
 'Gender',
 'Education',
 'MaritalStatus',
 'Usage',
 'Fitness',
 'Income',
 'Miles',
 'Fitness_level']
```

In [43]:
```
for col in ['Gender','MaritalStatus','Usage','Fitness']:
    sns.countplot(data=data,x='Product',hue=col)
    plt.show()
```




MaritalStatus
Single
Partnered
count
50
40
30
20
10
0
KP281
KP481
KP781
Product


Usage
2
3
4
5
6
7
count
35
30
25
20
15
10
5
0
KP281
KP481
KP781
Product



Low and Mid end product is preferred by both genders almost equally but high end is usually preferred by Males

Partnered customers are more like to purchase products at all product types over Single customers

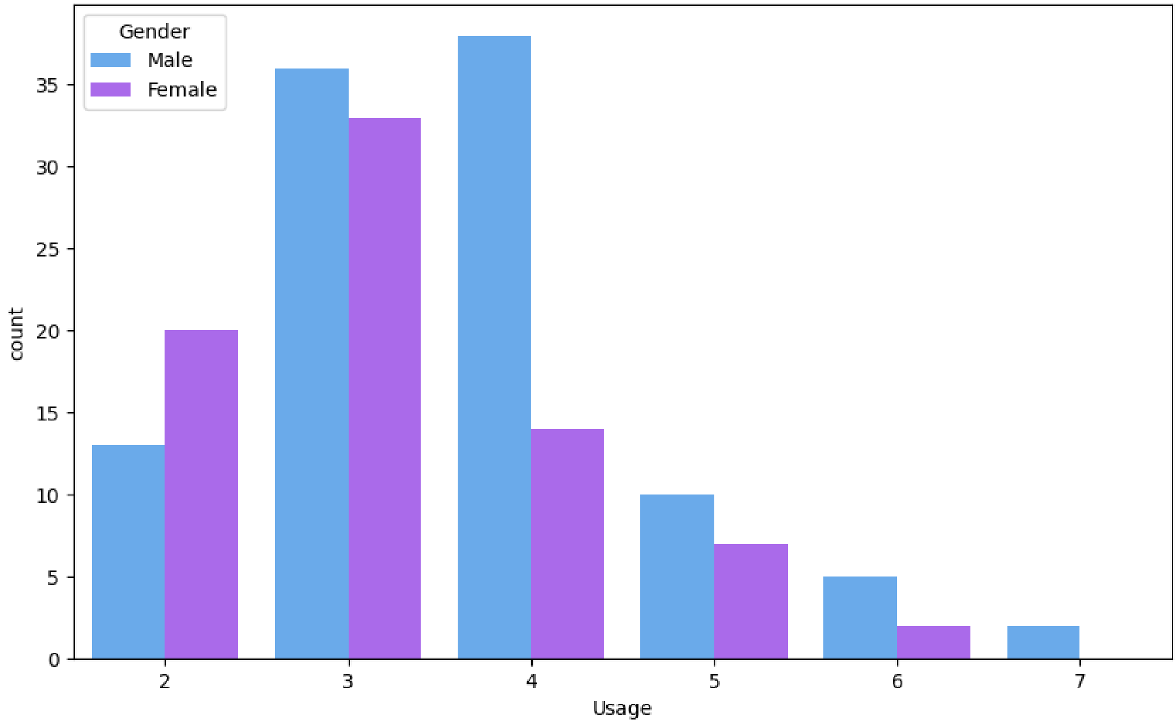
Low and Mid end product is preferred by customers whose usage is 2-4 days but those having usage above 4 days a week prefer high end model.

Low and Mid end product is preferred by customers whose fitness score is 3 but High end models are takem by customers who have more than 4 fitness score.

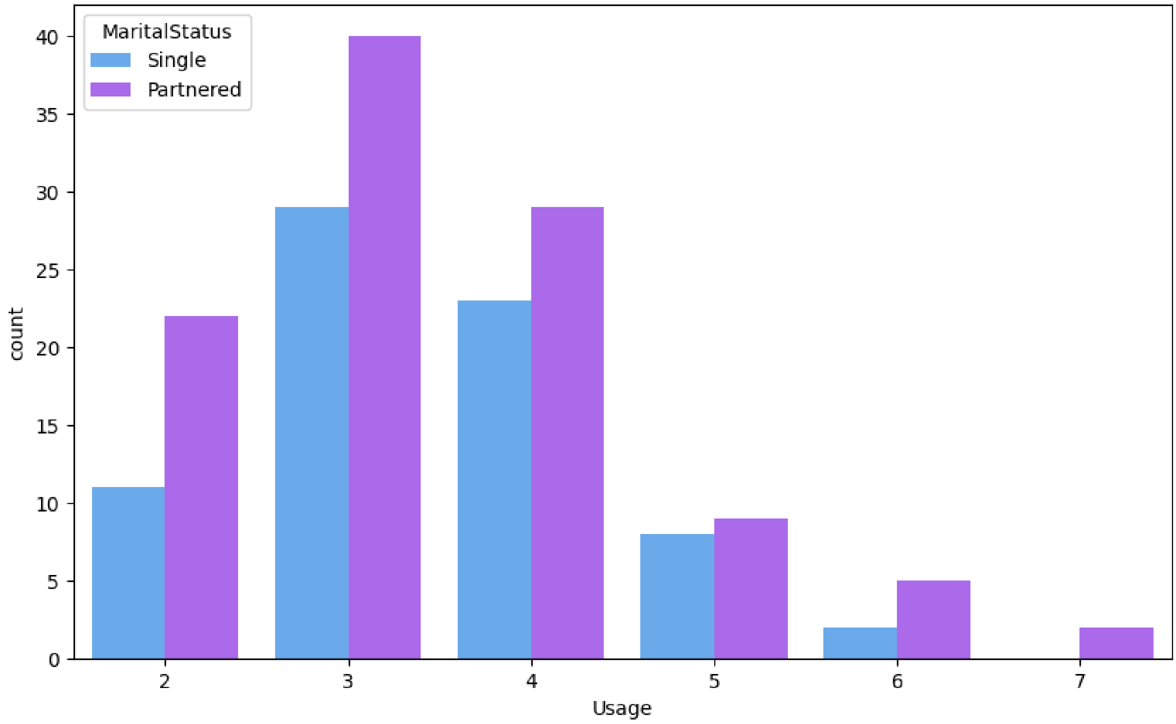
Bi-variate Analysis

In [44]:

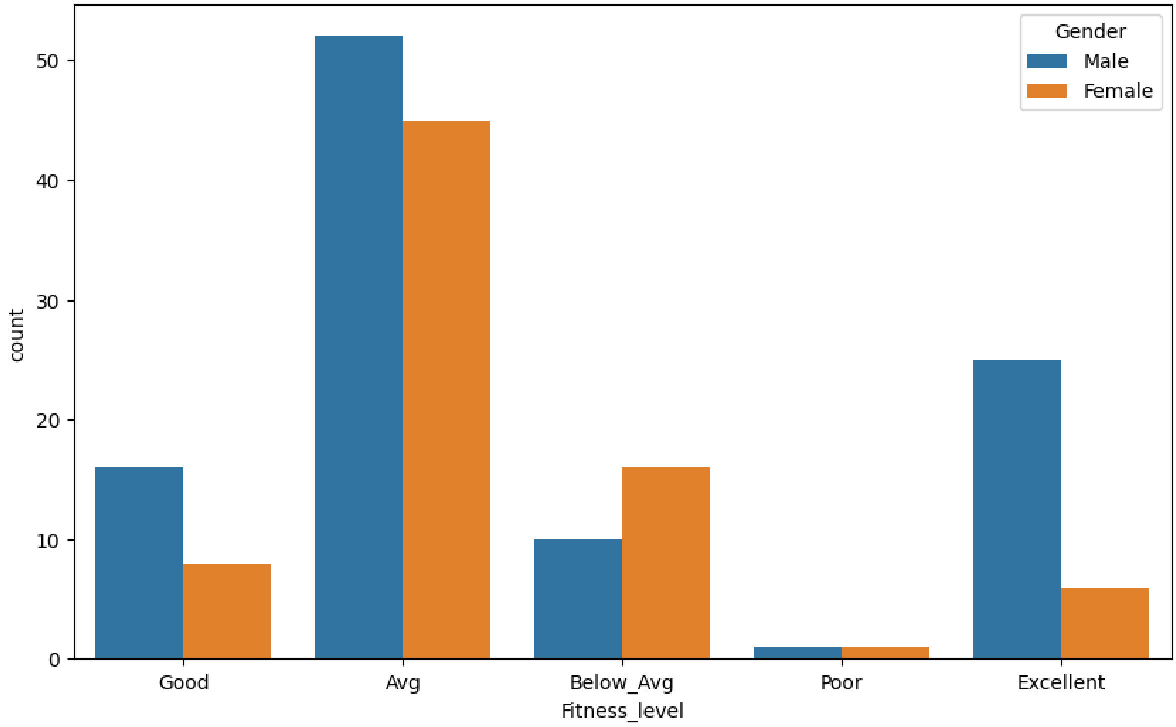
```
# Purchased product usage among Gender : both the Genders tend to use 3 days a weel
plt.figure(figsize=(10,6))
sns.countplot(data=data,x='Usage',hue='Gender',palette='cool')
plt.show()
```



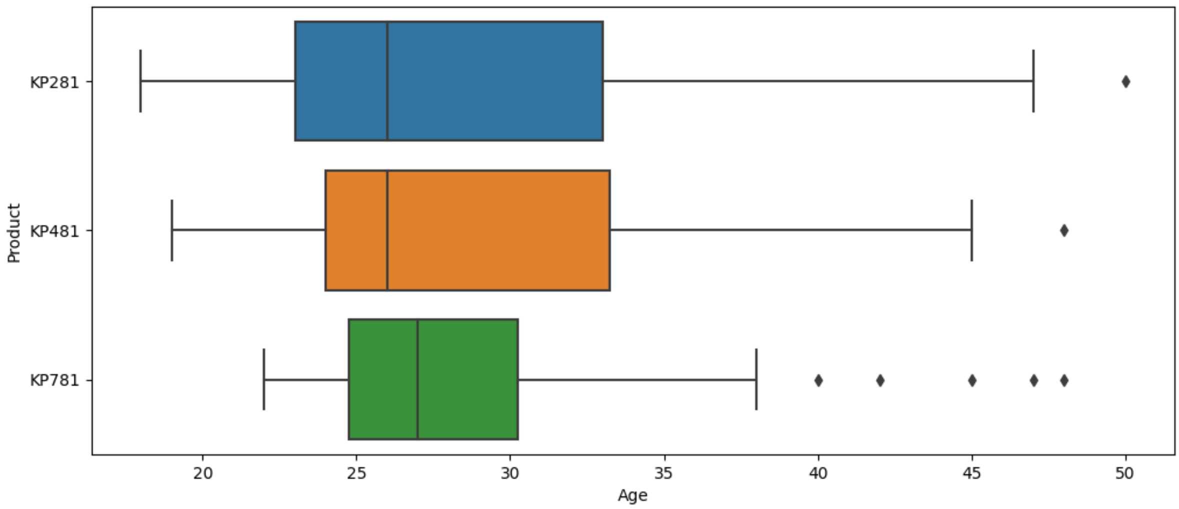
In [45]:
```
# Partners customers tend to have better usage over Single customers.
plt.figure(figsize=(10,6))
sns.countplot(data=data,x='Usage',hue='MaritalStatus',palette='cool')
plt.show()
```



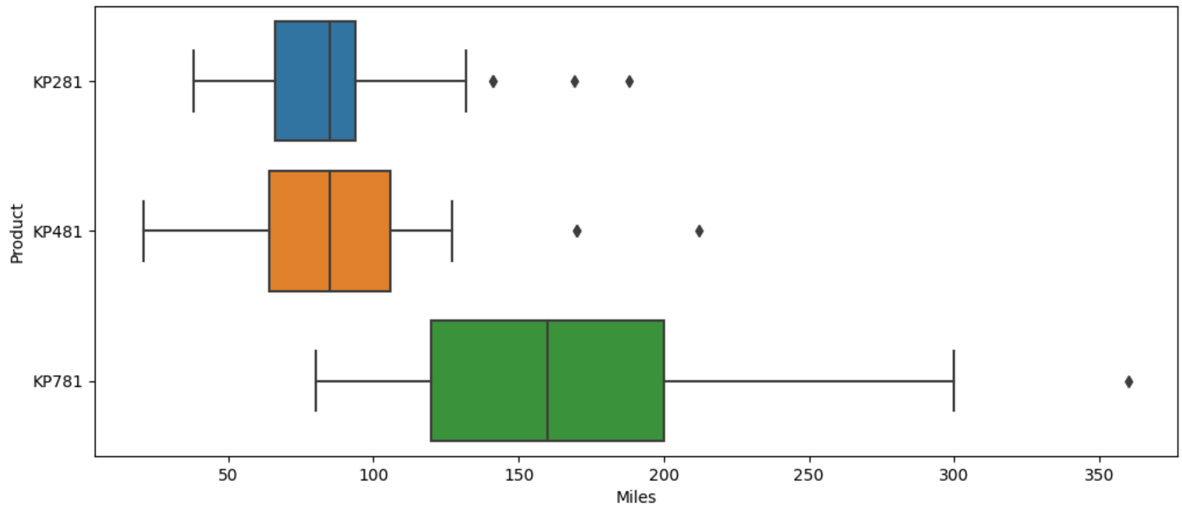
In [46]:
```
# Males are found to be fitter than females based on fitness score
plt.figure(figsize=(10,6))
sns.countplot(data=data,x='Fitness_level',hue='Gender')
plt.show()
```



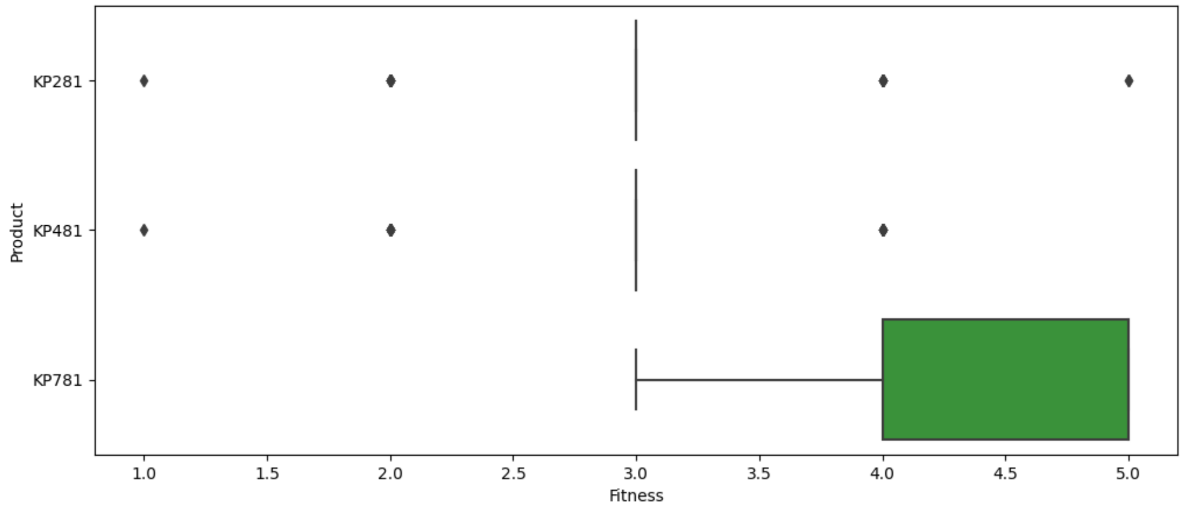
In [47]:
```
# Average age of customers across products are between 25-30 years
plt.figure(figsize=(12,5))
sns.boxplot(x='Age',y='Product',data=data)
plt.show()
```



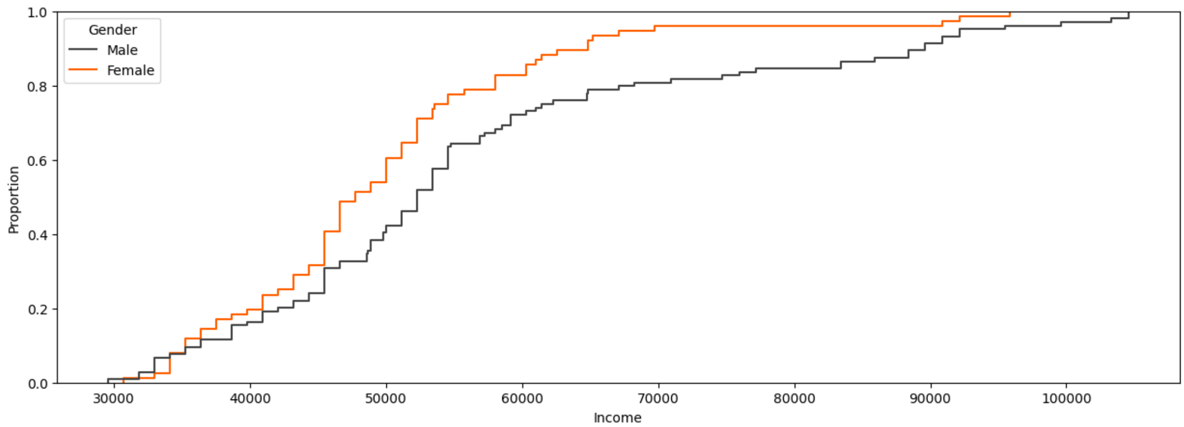
In [48]:
```
# Miles with each product : The customers using High end product run twice than tho
plt.figure(figsize=(12,5))
sns.boxplot(x='Miles',y='Product',data=data)
plt.show()
```



In [49]:
```
# Fitness of customer with each product
plt.figure(figsize=(12,5))
sns.boxplot(x='Fitness',y='Product',data=data)
plt.show()
```



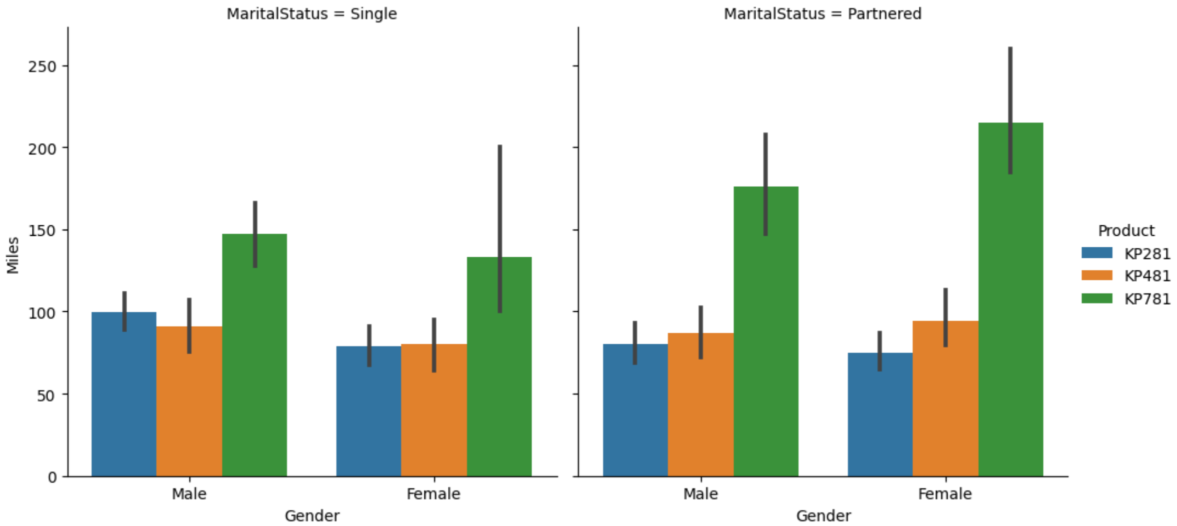
In [50]:
```
# Min Income of customer who can afford any of the three products is 30000
plt.figure(figsize=(15,5))
sns.ecdfplot(data=data,x='Income',hue='Gender',complementary=False,palette=['#4545
plt.show()
```



-- Mulit-variate Analysis

In [51]:
```
# Miles covered in each product by gender and their marital status
sns.catplot(x='Gender',y='Miles',hue='Product',col='MaritalStatus',data=data,kind=
```

```
plt.show()
'''
Among Partnered feamles have run better than males but in Singles the Males have be
However, the high end product KP781 has outperfomed other products in encouraging m
'''
```



Out[51]:

```
'\nAmong Partnered feamles have run better than males but in Singles the Males hav
e better than females. \nHowever, the high end product KP781 has outperfomed other
products in encouraging more miles ot run among customers.\n'
```

Probability Analysis

In [52]:

```
def gender_Probability(gender,data):
    print(f"Prob P(KP781) for {gender}: {round(data['KP781'][gender]/data.loc[gende
    print(f"Prob P(KP481) for {gender}: {round(data['KP481'][gender]/data.loc[gende
    print(f"Prob P(KP281) for {gender}: {round(data['KP281'][gender]/data.loc[gende

df_temp = pd.crosstab(index=data['Gender'],columns=[data['Product']])
print("Prob of Male: ",round(df_temp.loc['Male'].sum()/len(data),3))
print("Prob of Female: ",round(df_temp.loc['Female'].sum()/len(data),3))
print()
gender_Probability('Male',df_temp)
print()
gender_Probability('Female',df_temp)
```

```
Prob of Male:  0.578
Prob of Female:  0.422

Prob P(KP781) for Male: 0.317
Prob P(KP481) for Male: 0.298
Prob P(KP281) for Male: 0.385

Prob P(KP781) for Female: 0.092
Prob P(KP481) for Female: 0.382
Prob P(KP281) for Female: 0.526
```

In [53]:

```
def MS_Probability(ms_status,data):
    print(f"Prob P(KP781) for {ms_status}: {round(data['KP781'][ms_status]/data.lo
    print(f"Prob P(KP481) for {ms_status}: {round(data['KP481'][ms_status]/data.lo
    print(f"Prob P(KP281) for {ms_status}: {round(data['KP281'][ms_status]/data.lo

df_temp = pd.crosstab(index=data['MaritalStatus'],columns=[data['Product']])
print("Prob of P(Single): ",round(df_temp.loc['Single'].sum()/len(data),3))
print("Prob of P(Married/Partnered): ",round(df_temp.loc['Partnered'].sum()/len(da
print()
MS_Probability('Single',df_temp)
```

```
print()
MS_Probability('Partnered',df_temp)
```

```
Prob of P(Single):  0.406
Prob of P(Married/Partnered):  0.594

Prob P(KP781) for Single: 0.233
Prob P(KP481) for Single: 0.329
Prob P(KP281) for Single: 0.438

Prob P(KP781) for Partnered: 0.215
Prob P(KP481) for Partnered: 0.336
Prob P(KP281) for Partnered: 0.449
```

In [54]:
```
data['Miles'].min(), data['Miles'].max()
```

Out[54]:
```
(21, 360)
```

In [55]:
```
data_cat['mile_group'] = data_cat.Miles
data_cat.mile_group = pd.cut(data.mile_group,bins=[0,100,200,300,400],labels=['Und
data_cat.head()
```

Out[55]:

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

In [56]:
```
data_cat['age_group'] = data_cat.Age
data_cat.head()
```

Out[56]:

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

In [57]:
```
data_cat.age_group = pd.cut(data.age_group,bins=[0,21,35,45,60],labels=['Teen','Adu
```

In [58]:
```
data_cat.head()
```

Out[58]:

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

In [59]:

```
pd.crosstab(index=data_cat.Product,columns=data_cat.age_group,margins=True)
```

Out[59]:

age_group	Teen	Adult	Middle Aged	Elder	All
Product					
KP281	10	56	11	3	80
KP481	7	45	7	1	60
KP781	0	34	4	2	40
All	17	135	22	6	180

In [60]:

```
# Conditional and Marginal Probabilities with product type and age group
np.round(pd.crosstab(index=data_cat.Product,columns=data_cat.age_group,normalize=T
```

Out[60]:

age_group	Teen	Adult	Middle Aged	Elder	All
Product					
KP281	0.06	0.31	0.06	0.02	0.44
KP481	0.04	0.25	0.04	0.01	0.33
KP781	0.00	0.19	0.02	0.01	0.22
All	0.09	0.75	0.12	0.03	1.00

In [61]:

```
# Conditional and Marginal Probabilities with product type and fitness level
round(pd.crosstab(index=data_cat["Product"],columns=data_cat["Fitness_level"],norma
```

Out[61]:

Fitness_level	Avg	Below_Avg	Excellent	Good	Poor
Product					
KP281	0.56	0.54	0.06	0.38	0.5
KP481	0.40	0.46	0.00	0.33	0.5
KP781	0.04	0.00	0.94	0.29	0.0

In [62]:

```
round(pd.crosstab(index=[data_cat.Product,data_cat.Fitness_level],columns=data_cat
```

Out[62]:

Product	Fitness_level	Female	Male
KP281	Avg	0.14	0.16
	Below_Avg	0.06	0.02
	Excellent	0.01	0.01
	Good	0.02	0.03
	Poor	0.00	0.01
KP481	Avg	0.10	0.12
	Below_Avg	0.03	0.03
	Good	0.02	0.02
	Poor	0.01	0.00
KP781	Avg	0.01	0.02
	Excellent	0.03	0.13
	Good	0.01	0.03

In [63]:

```
round(pd.crosstab(index=[data_cat.Product,data_cat.MaritalStatus],columns=data_cat
```

Out[63]:

Product	MaritalStatus	Female	Male
KP281	Partnered	0.15	0.12
	Single	0.07	0.11
KP481	Partnered	0.08	0.12
	Single	0.08	0.06
KP781	Partnered	0.02	0.11
	Single	0.02	0.08

In [64]:

```
np.round(((pd.crosstab(data.Product,data.Gender,margins=True))/len(data)),2)
```

Out[64]:

Product	Female	Male	All
KP281	0.22	0.22	0.44
KP481	0.16	0.17	0.33
KP781	0.04	0.18	0.22
All	0.42	0.58	1.00

In [65]:

```
np.round((pd.crosstab([data.Product],data.Gender,margins=True,normalize="columns")
```

Out[65]:

Gender	Female	Male	All
Product			
KP281	0.53	0.38	0.44
KP481	0.38	0.30	0.33
KP781	0.09	0.32	0.22

In [66]:

```
np.round((pd.crosstab([data.Product,data.Gender],data.mile_group,margins=True,norma
```

Out[66]:

	mile_group	Under_runner	Good_Runner	Acheiver	Extra_Miler	All
Product	**Gender**					
KP281	**Female**	0.30	0.10	0.0	0.0	0.22
	Male	0.25	0.20	0.0	0.0	0.22
KP481	**Female**	0.19	0.10	0.2	0.0	0.16
	Male	0.19	0.15	0.0	0.0	0.17
KP781	**Female**	0.02	0.07	0.2	0.0	0.04
	Male	0.05	0.38	0.6	1.0	0.18

In [ ]:

In [ ]: