aerofit-project

February 25, 2024

##Business Problem

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.

#Importing Libraries

[6]:
```
#Import the needed libraries

import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
import seaborn as sns
import warnings

warnings.simplefilter(action='ignore', category=FutureWarning)
```

#Loading Data into Colab

[4]:
```
#For Import the file
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving AeroFit_Project.csv to AeroFit_Project.csv
```

#Loading the dataset

[7]:
```
# Loading the dataset
df = pd.read_csv(r'AeroFit_Project.csv')
```

[8]:
```
#To check the head values of the dataset
df.head()
```

[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

#Basic Dataset Checking

#Defining Problem Statement and Analysing basic metrics (10 Points) Observations on shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), statistical summary

[9]:
```
#Check the size of the Dataset
df.size
```

[9]:
```
1620
```

[10]:
```
#Check the shape of the dataset
df.shape
```

[10]:
```
(180, 9)
```

[11]:
```
#Check the columns of the dataset
df.columns
```

[11]:
```
Index(['Product', 'Age', 'Gender', 'Education', 'MaritalStatus', 'Usage',
       'Fitness', 'Income', 'Miles'],
      dtype='object')
```

[12]:
```
#Check the data types of the dataset
df.dtypes
```

[12]:
```
Product          object
Age               int64
Gender           object
Education         int64
MaritalStatus    object
Usage             int64
Fitness           int64
Income            int64
Miles             int64
dtype: object
```

[13]:
```
#Check the info of the dataset
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
```

#Non-Graphical Analysis: Value counts and unique attributes (10 Points)

[14]:
```
# Number of unique values in each column
for i in df.columns:
  print(i, ':', df[i].nunique())
```

```
Product : 3
Age : 32
Gender : 2
Education : 8
MaritalStatus : 2
Usage : 6
Fitness : 5
Income : 62
Miles : 37
```

From the above observation, we can conclude that only Income, Miles and Age can be considered as Continuous, the rest of the columns though integers/floats should be considered as categories.

Checking null values

[15]:
```
# Checking for null values -
df.isnull().sum()
```

[15]:
```
Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
```

```
Usage           0
Fitness         0
Income          0
Miles           0
dtype: int64
```

There are no null values in the provided dataset

#Checking the value counts for categorical columns

[16]:
```
df['Product'].value_counts()
```

[16]:
```
KP281    80
KP481    60
KP781    40
Name: Product, dtype: int64
```

[17]:
```
df['Gender'].value_counts()
```

[17]:
```
Male      104
Female     76
Name: Gender, dtype: int64
```

[18]:
```
df['Education'].value_counts()
```

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

[19]:
```
df['MaritalStatus'].value_counts()
```

[19]:
```
Partnered    107
Single        73
Name: MaritalStatus, dtype: int64
```

[20]:
```
df['Usage'].value_counts()
```

[20]:
```
3    69
4    52
2    33
5    17
6     7
```

```
7     2
Name: Usage, dtype: int64
```

[21]:
```
df['Fitness'].value_counts()
```

[21]:
```
3    97
5    31
2    26
4    24
1     2
Name: Fitness, dtype: int64
```

Visual Analysis - Univariate & Bivariate (30 Points) For continuous variable(s): Distplot, countplot, histogram for univariate analysis (10 Points) For categorical variable(s): Boxplot (10 Points) For correlation: Heatmaps, Pairplots(10 Points)

[22]:
```
# A broader look at correlation between the columns of dataframe

# Creating a copy of the dataframe -
df_copy=df.copy()

df_copy['Gender'].replace(['Male', 'Female'], [1, 0], inplace=True)

df_copy['MaritalStatus'].replace(['Single', 'Partnered'], [0, 1], inplace=True)

df_copy['Product'].replace(['KP281', 'KP481', 'KP781'], [0, 1, 2], inplace=True)

df_copy.corr()
```

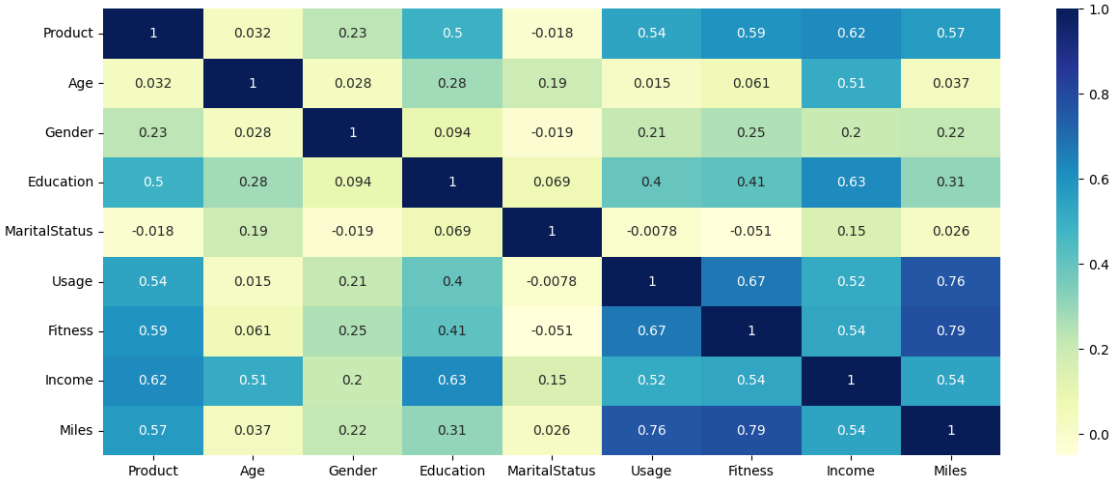
[22]:

	Product	Age	Gender	Education	MaritalStatus	\
Product	1.000000	0.032225	0.230653	0.495018	-0.017602	
Age	0.032225	1.000000	0.027544	0.280496	0.192152	
Gender	0.230653	0.027544	1.000000	0.094089	-0.018836	
Education	0.495018	0.280496	0.094089	1.000000	0.068569	
MaritalStatus	-0.017602	0.192152	-0.018836	0.068569	1.000000	
Usage	0.537447	0.015064	0.214424	0.395155	-0.007786	
Fitness	0.594883	0.061105	0.254609	0.410581	-0.050751	
Income	0.624168	0.513414	0.202053	0.625827	0.150293	
Miles	0.571596	0.036618	0.217869	0.307284	0.025639	

	Usage	Fitness	Income	Miles
Product	0.537447	0.594883	0.624168	0.571596
Age	0.015064	0.061105	0.513414	0.036618
Gender	0.214424	0.254609	0.202053	0.217869
Education	0.395155	0.410581	0.625827	0.307284
MaritalStatus	-0.007786	-0.050751	0.150293	0.025639
Usage	1.000000	0.668606	0.519537	0.759130

Fitness	0.668606	1.000000	0.535005	0.785702
Income	0.519537	0.535005	1.000000	0.543473
Miles	0.759130	0.785702	0.543473	1.000000

[23]:
```
# Correlation Plot above as a Heatmap -

plt.figure(figsize=(15,6))
sns.heatmap(df_copy.corr(), cmap="YlGnBu", annot=True)
plt.show()
```



Important Points

1) The product/treadmill purchased highly correlates with Education, Income, Usage, Fitness and Miles
2) Age is highly correlated to Income (0.51) which definitely seems reasonable. It's also correlated with Education and Marital Status which stands completely alright.
3) Gender certainly has some correlation to Usage, Fitness, Income and Miles.
4) Education is correlated to Age and Miles. It's highly correlated to Income (as expected). It's sufficiently correlated to Usage and Fitness too.
5) Marital Status has some correlation to Income and Age (as expected).
6) Usage is extremely correlated to Fitness and Miles and has a higher correlation with Income as well.
7) Fitness has a great correlation with Income.

More Observations

1) Product, Fitness, Usage and Miles depict a ridiculously higher correlation among themselves which looks as expected since more the usage implying more miles run and certainly more fitness.

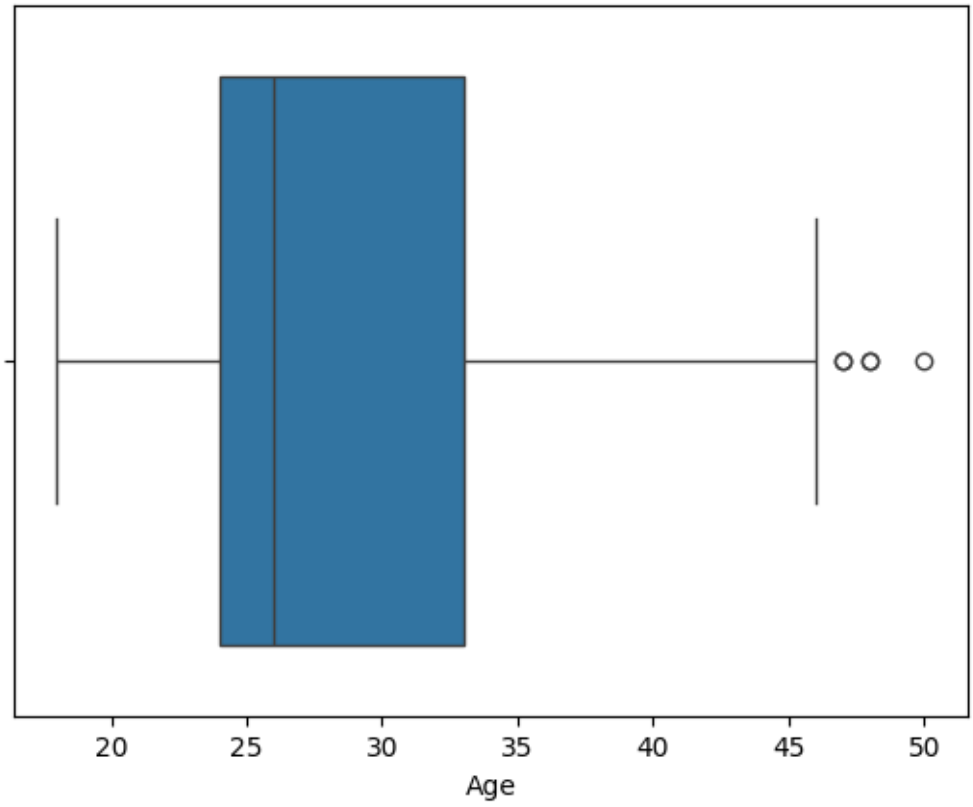
2) Also a story which seems reasonable is that Age and Education (predominately) are indicators of Income which affects the products bought. The more advanced the product is, the more its usage and hence more the miles run which in turns improves the fitness.

Note:- Above point 2 is just something which may or mayn't be true as **Correlation doesn't imply Causation**.

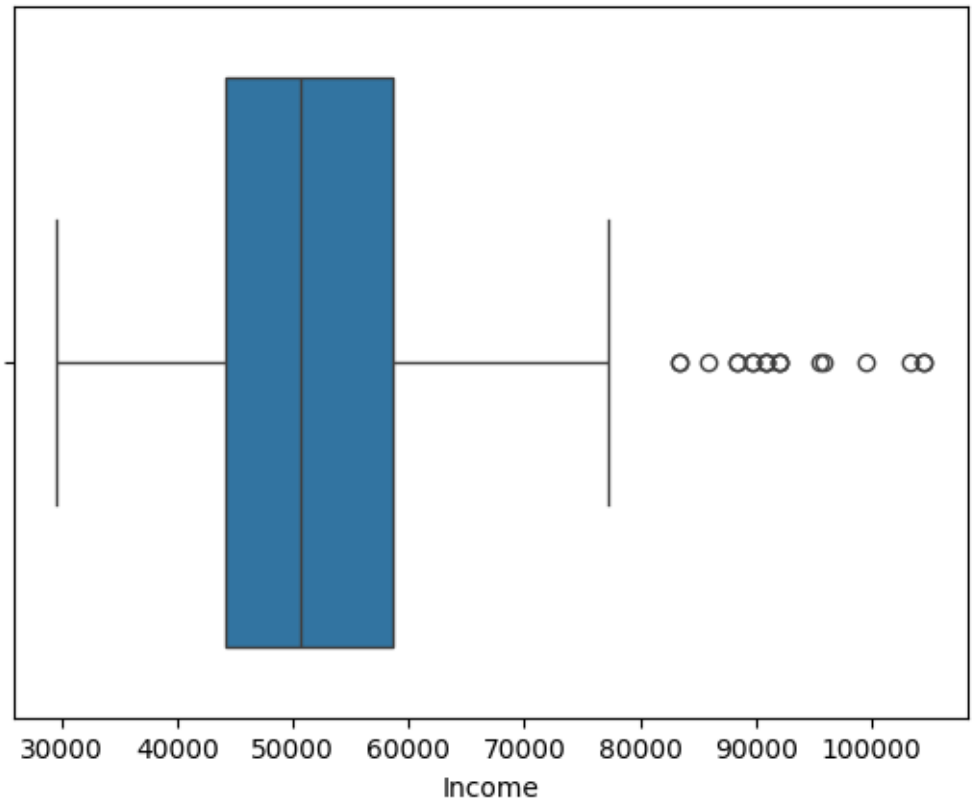
##Missing Value & Outlier Detection (10 Points)

Observing the Outliers of Age, Income and Miles -

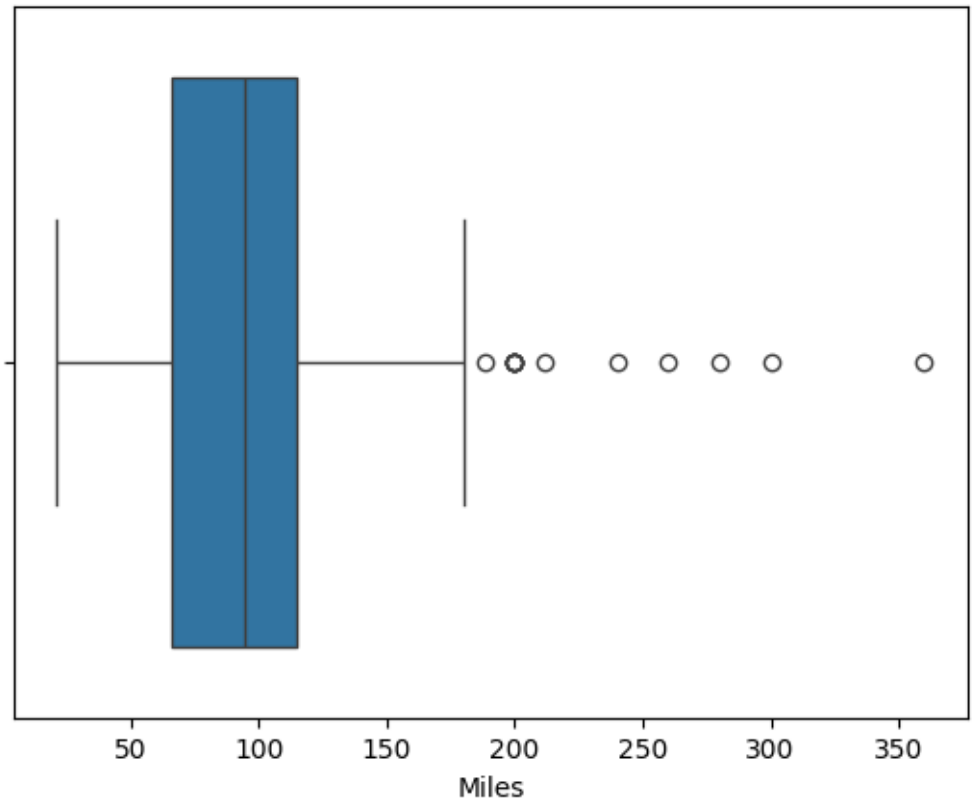
[24]:
```
ax = sns.boxplot(x=df["Age"])
plt.show()
```



[25]:
```
ax = sns.boxplot(x=df["Income"])
plt.show()
```



[26]:
```
ax = sns.boxplot(x=df["Miles"])
plt.show()
```



Certainly there are outliers in our data. When we remove them, this leads to loss of information and moreover we have a very small dataset of 180 rows. So instead of removing, it's going to be clipped (i.e. - ranges between 5 percentile and 95 percentile).

The outlier treatment is demonstrated in the below piece of code for the copy of dataframe but it is not incorporated in the below analysis since these three numerical columns are binned and then analyzed.

[27]:
```
num_feat=['Age', 'Income', 'Miles']
for col in num_feat:
    percentiles = df[col].quantile([0.05,0.95]).values
    df_copy[col] = np.clip(df_copy[col], percentiles[0], percentiles[1])
```

[28]:
```
df.columns
```

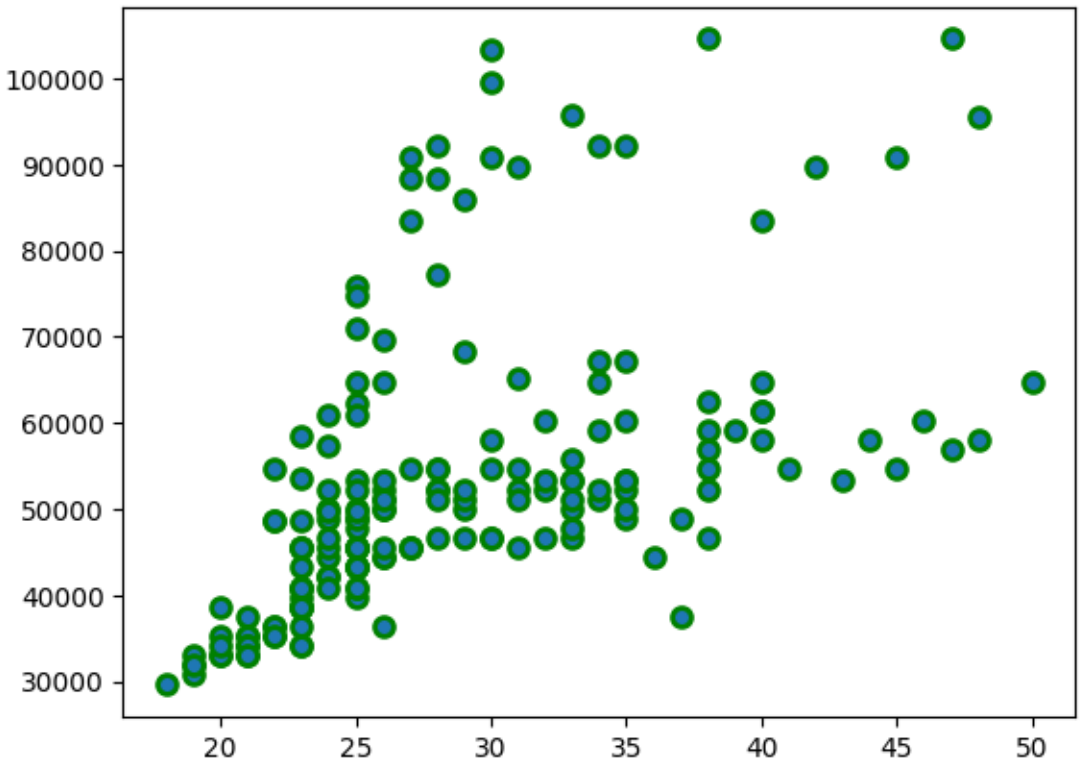
[28]:
```
Index(['Product', 'Age', 'Gender', 'Education', 'MaritalStatus', 'Usage',
       'Fitness', 'Income', 'Miles'],
      dtype='object')
```

##Business Insights based on Non-Graphical and Visual Analysis (10 Points) Comments on the range of attributes Comments on the distribution of the variables and relationship between them Comments for each univariate and bivariate plot

Scatterplots for Analysis of Continuous Variables -

[29]:
```
# Observing the association between Age and Income -

plt.scatter(df['Age'], df['Income'],
            linewidths = 2,
            marker ="o",
            edgecolor ="green",
            s = 50)
```

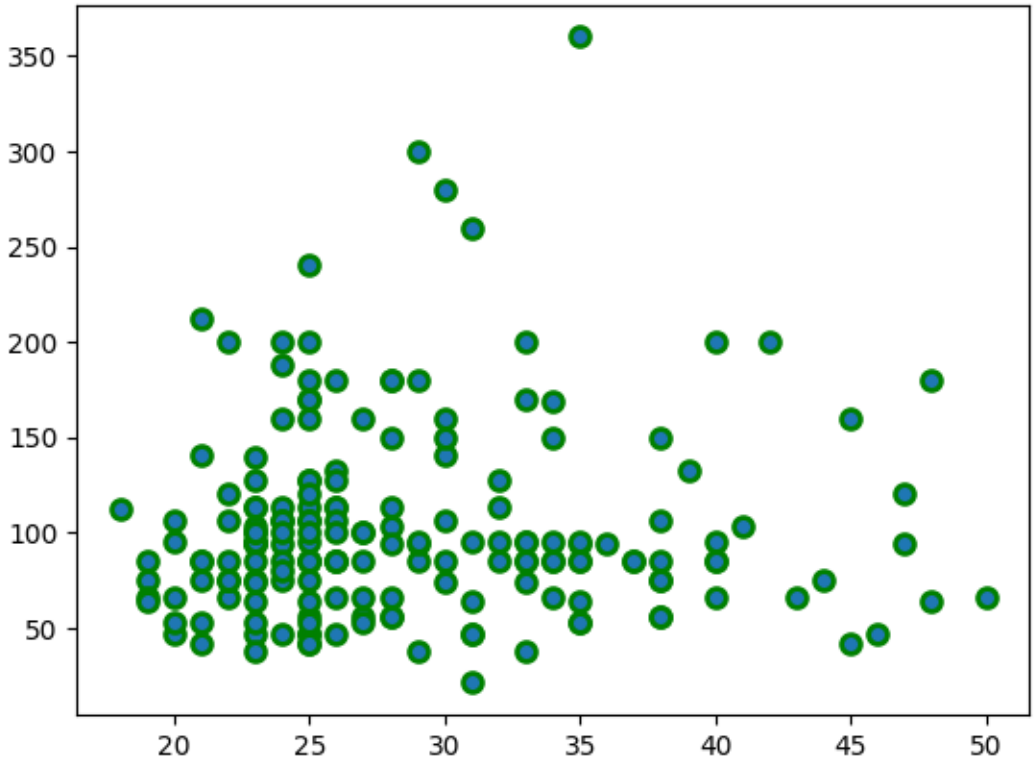
[29]: <matplotlib.collections.PathCollection at 0x7afc5376e6b0>



The variance of income in lower ages is smaller as compared to the variance in higher ages, probably something called as **Heteroscadasticity**.

[30]:
```
# Observing the association between Age and Miles -

plt.scatter(df['Age'], df['Miles'],
            linewidths = 2,
            marker ="o",
            edgecolor ="green",
            s = 50)
```

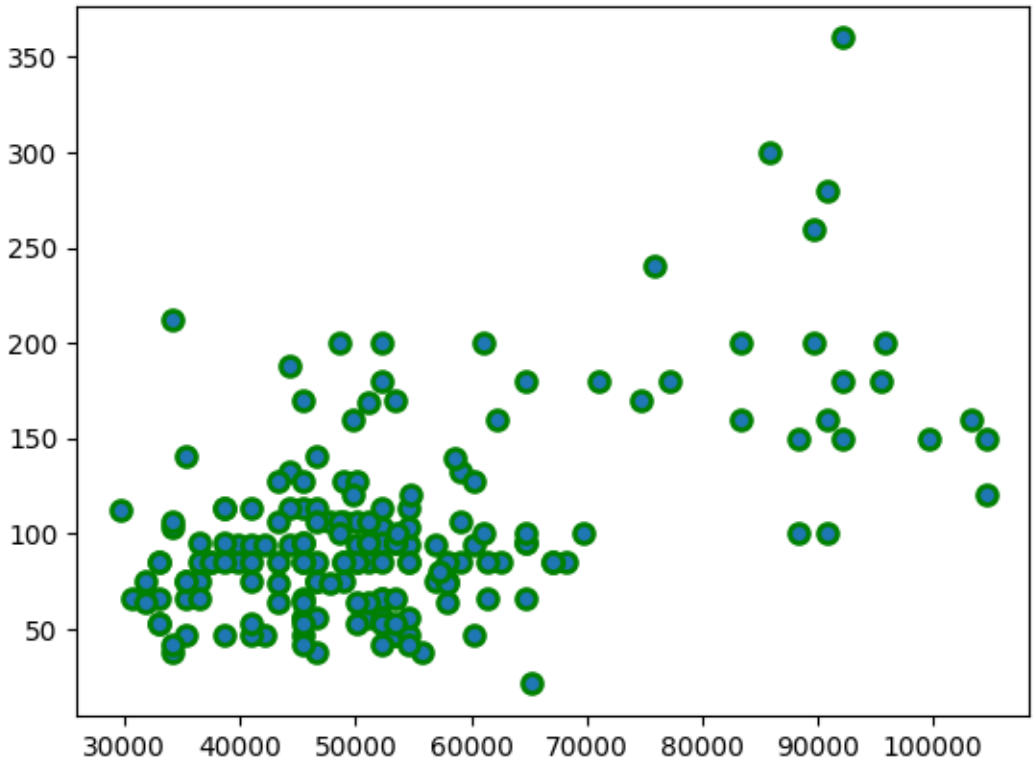
[30]: <matplotlib.collections.PathCollection at 0x7afc53805c30>



No significant pattern or observation between Age and Miles run on a treadmill, also implied by heatmap with a correlation of 0.037.

[31]:
```
# Observing the association between Miles and Income -

plt.scatter(df['Income'], df['Miles'],
            linewidths = 2,
            marker ="o",
            edgecolor ="green",
            s = 50)
```

[31]: <matplotlib.collections.PathCollection at 0x7afc5376ed10>



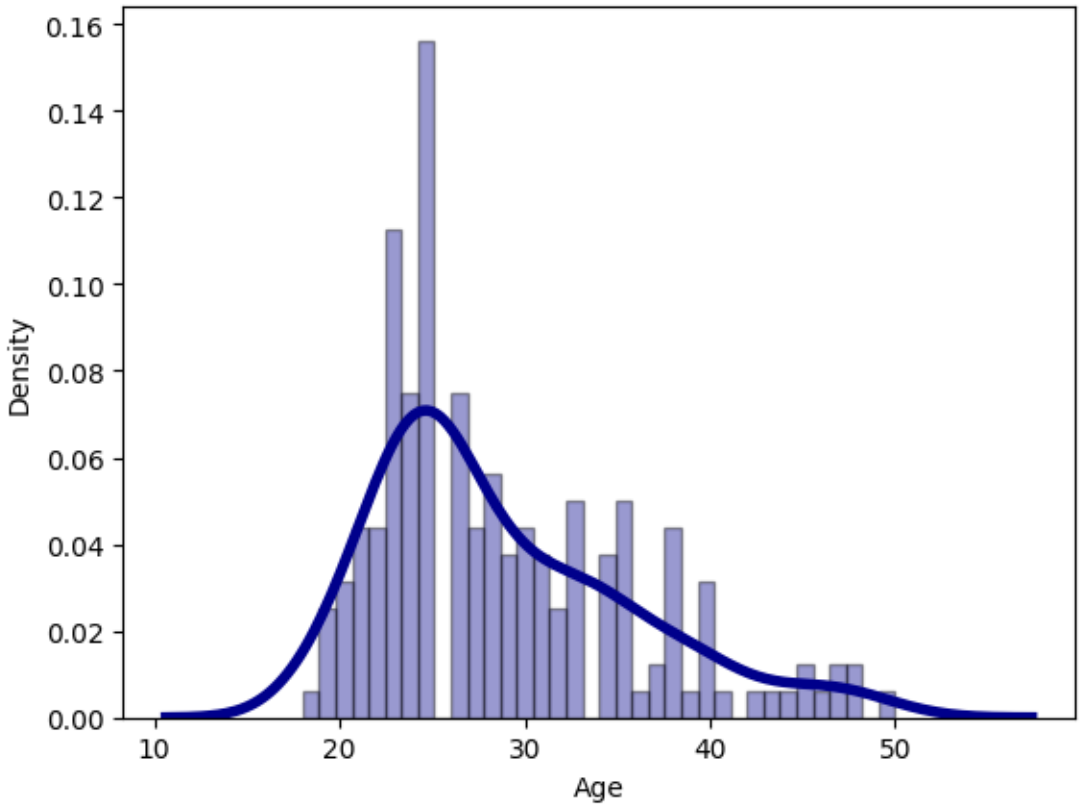
The miles run on treadmill increase significantly with income, infact only people having incomes above 70000 have run over 220 miles. Moreover there's also a **Heteroscadastic** effect.

Analysis of Categorical Columns with the Product - For this section, We'll be converting the Ages, Incomes and Miles to bins for better analysis.

[32]:
```
# Observing the ages to create bins -

sns.distplot(df['Age'], hist=True, kde=True,
bins=int(36), color = 'darkblue',
hist_kws={'edgecolor':'black'},
kde_kws={'linewidth': 4})
plt.show()
```

```
<ipython-input-32-edc3aa6432b2>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

```
sns.distplot(df['Age'], hist=True, kde=True,
```



[33]:
```
# Creating bins on intervals of 5 as age gaps and below 20, above 40 -

bins = [-1,20,25,30,35,40,55]
labels = ['<20','20-25','25-30','30-35','35-40','40+']
df['Age_bins'] = pd.cut(df['Age'], bins=bins, labels=labels)
df.head()
```

[33]:

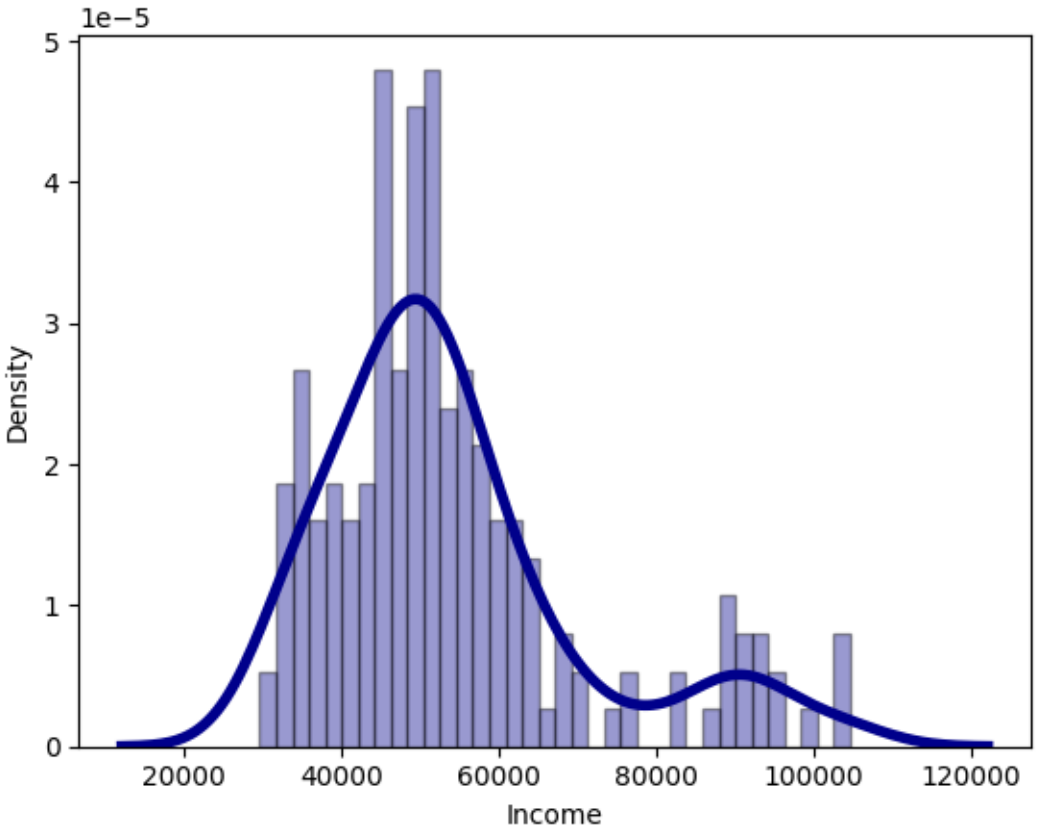
	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	\
0	KP281	18	Male	14	Single	3	4	29562	
1	KP281	19	Male	15	Single	2	3	31836	
2	KP281	19	Female	14	Partnered	4	3	30699	
3	KP281	19	Male	12	Single	3	3	32973	
4	KP281	20	Male	13	Partnered	4	2	35247	

	Miles	Age_bins
0	112	<20
1	75	<20
2	66	<20

```
3      85       <20
4      47       <20
```

[34]:
```
# Observing the incomes to create bins -

sns.distplot(df['Income'], hist=True, kde=True,
bins=int(36), color = 'darkblue',
hist_kws={'edgecolor':'black'},
kde_kws={'linewidth': 4})
plt.show()
```

```
<ipython-input-34-6807b30b7300>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Income'], hist=True, kde=True,
```



[35]:
```
df['Income'].describe()
```

[35]:
```
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
```

[36]:
```
# Creating bins for income -

bins = [-1,35000,45000,50000,60000,70000,90000,120000]
labels =␣
 ↪['<35000','35000-45000','45000-50000','50000-60000','60000-70000','70000-90000','90000+']
df['Income_bins'] = pd.cut(df['Income'], bins=bins, labels=labels)
df.head()
```

[36]:

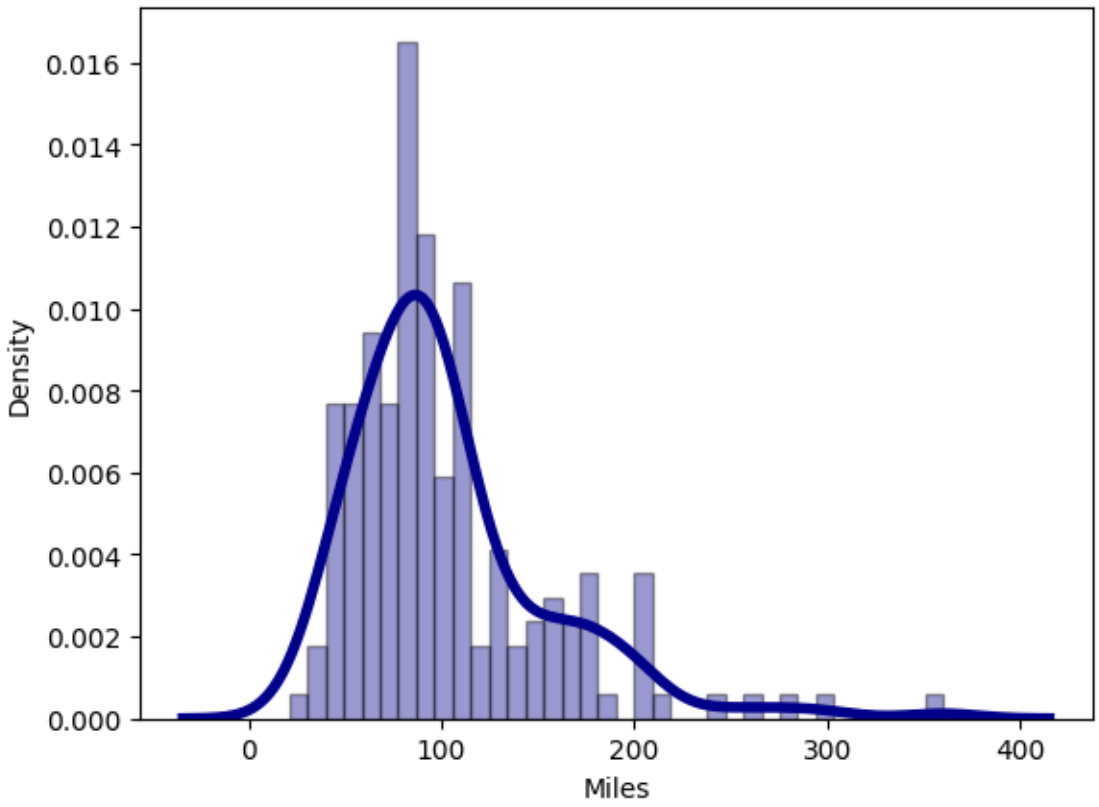
	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	\
0	KP281	18	Male	14	Single	3	4	29562	
1	KP281	19	Male	15	Single	2	3	31836	
2	KP281	19	Female	14	Partnered	4	3	30699	
3	KP281	19	Male	12	Single	3	3	32973	
4	KP281	20	Male	13	Partnered	4	2	35247	

	Miles	Age_bins	Income_bins
0	112	<20	<35000
1	75	<20	<35000
2	66	<20	<35000
3	85	<20	<35000
4	47	<20	35000-45000

[37]:
```
# Observing the miles to create bins -

sns.distplot(df['Miles'], hist=True, kde=True,
bins=int(36), color = 'darkblue',
hist_kws={'edgecolor':'black'},
kde_kws={'linewidth': 4})
plt.show()
```

```
<ipython-input-37-27d391b58bc0>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.
```

```
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Miles'], hist=True, kde=True,
```



[38]:
```
df['Miles'].describe()
```

[38]:
```
count    180.000000
mean     103.194444
std       51.863605
min       21.000000
25%       66.000000
50%       94.000000
75%      114.750000
max      360.000000
Name: Miles, dtype: float64
```

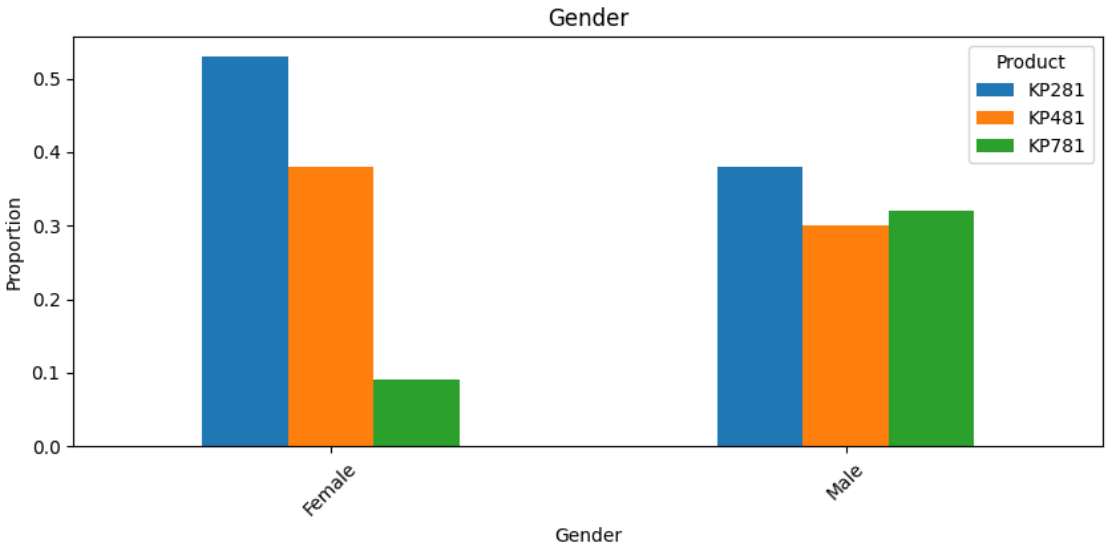
[39]:
```
# Creating bins for miles -

bins = [-1,50,100,150,400]
labels = ['<50','50-100','100-150','150+']
df['Mile_bins'] = pd.cut(df['Miles'], bins=bins, labels=labels)
df.head()
```

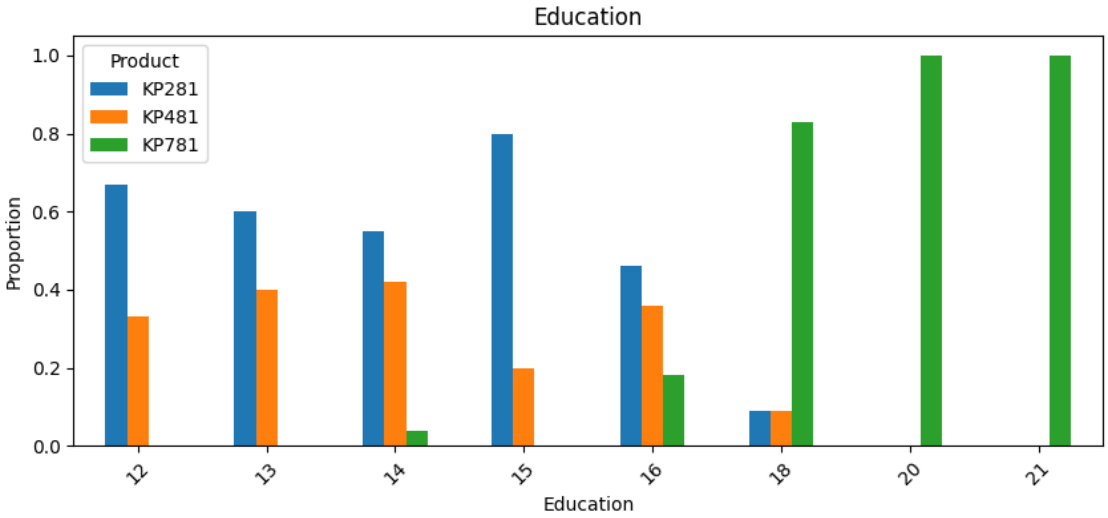
[39]:

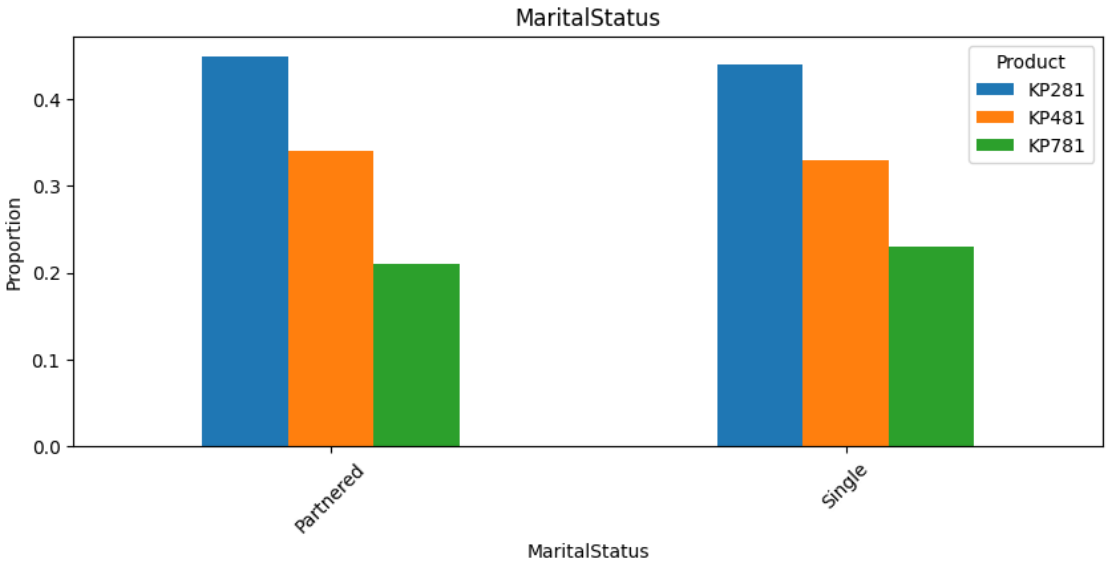
	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	\
0	KP281	18	Male	14	Single	3	4	29562	
1	KP281	19	Male	15	Single	2	3	31836	
2	KP281	19	Female	14	Partnered	4	3	30699	
3	KP281	19	Male	12	Single	3	3	32973	
4	KP281	20	Male	13	Partnered	4	2	35247	

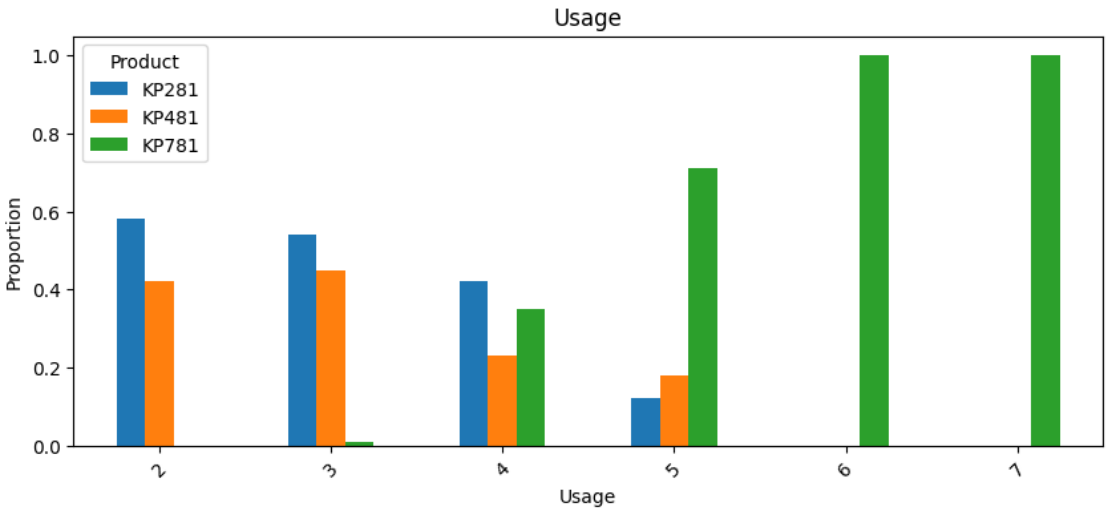
	Miles	Age_bins	Income_bins	Mile_bins
0	112	<20	<35000	100-150
1	75	<20	<35000	50-100
2	66	<20	<35000	50-100
3	85	<20	<35000	50-100
4	47	<20	35000-45000	<50

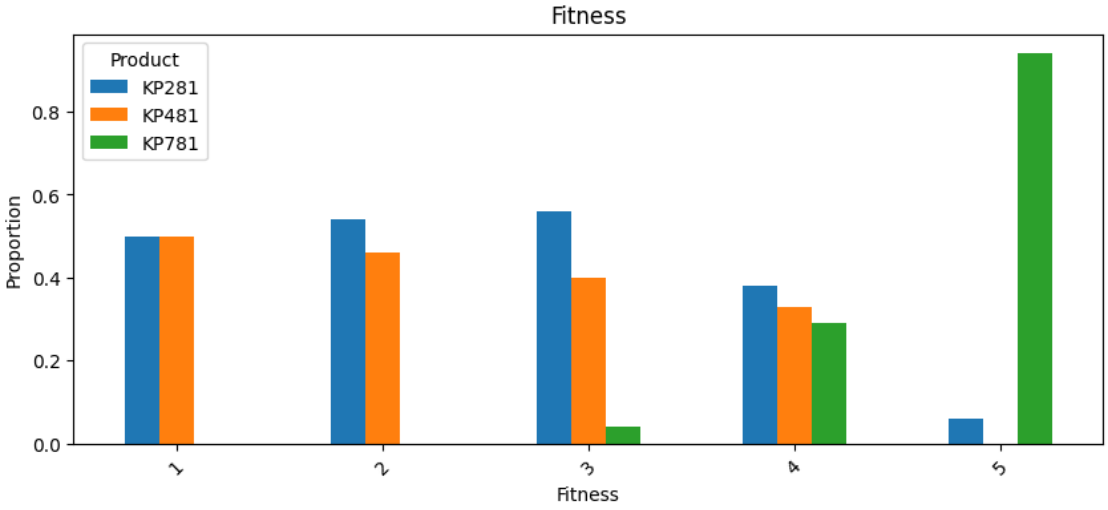
[40]:
```
# Crosstabs -

cat_cols=['Gender','Education', 'MaritalStatus',␣
 ↪'Usage','Fitness','Age_bins','Income_bins','Mile_bins']
for i in cat_cols:
    other= round(pd.crosstab(df[df[i].notnull()][i], df['Product']).\
                 div(pd.crosstab(df[df[i].notnull()][i],df['Product']).
 ↪apply(sum,1),0),2)
    ax = other.plot(kind ='bar', title = i, figsize = (10,4))
    ax.set_xlabel(i)
    ax.set_ylabel('Proportion')
    plt.xticks(rotation=45)
    plt.show()
```

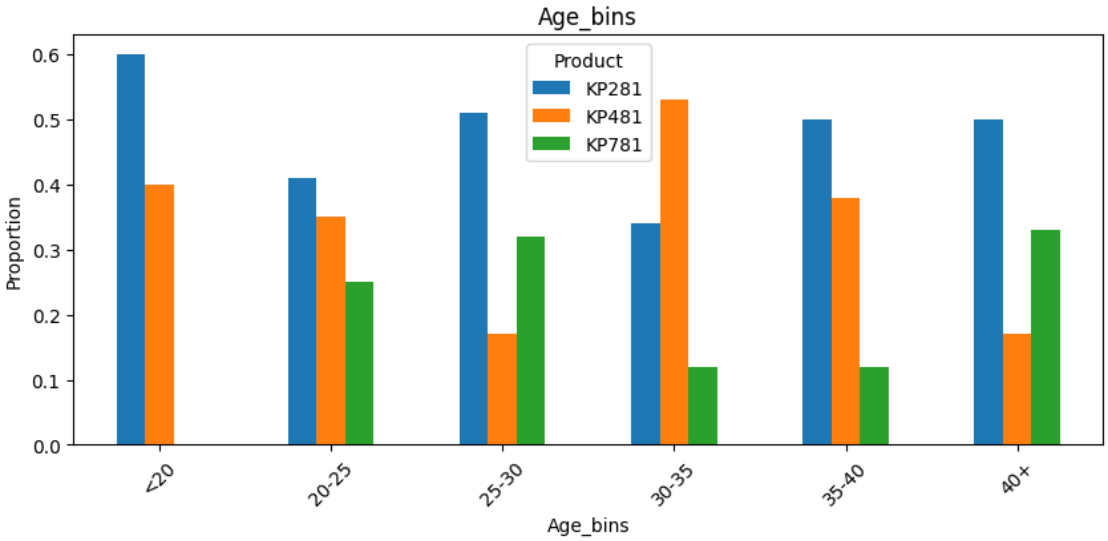

Gender
Product
KP281
KP481
KP781
Proportion
0.0
0.1
0.2
0.3
0.4
0.5
Female
Male
Gender

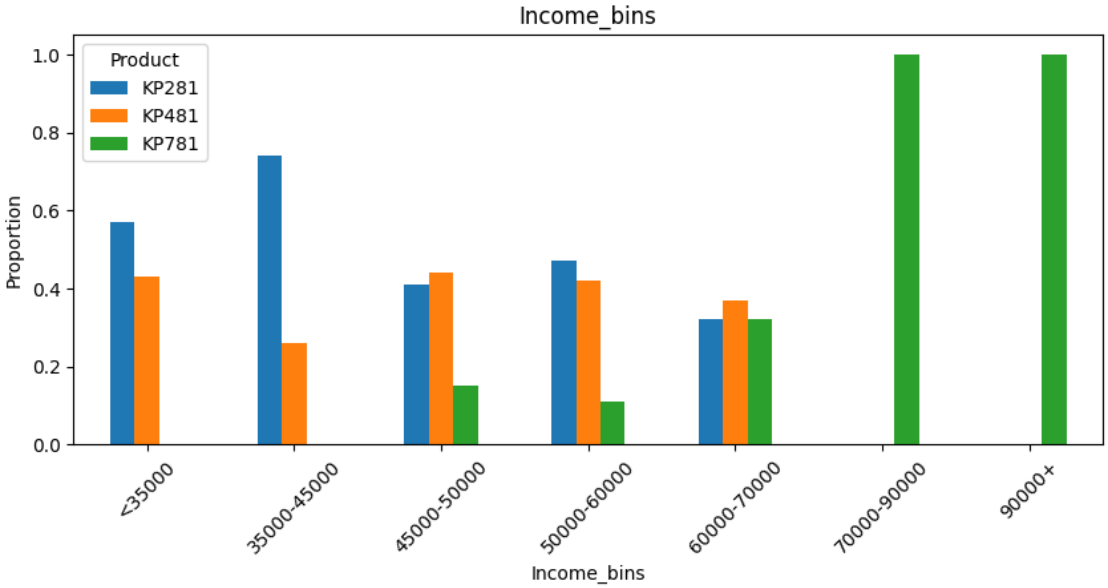

Education
Product
KP281
KP481
KP781
Proportion
0.0
0.2
0.4
0.6
0.8
1.0
12
13
14
15
16
18
20
21
Education

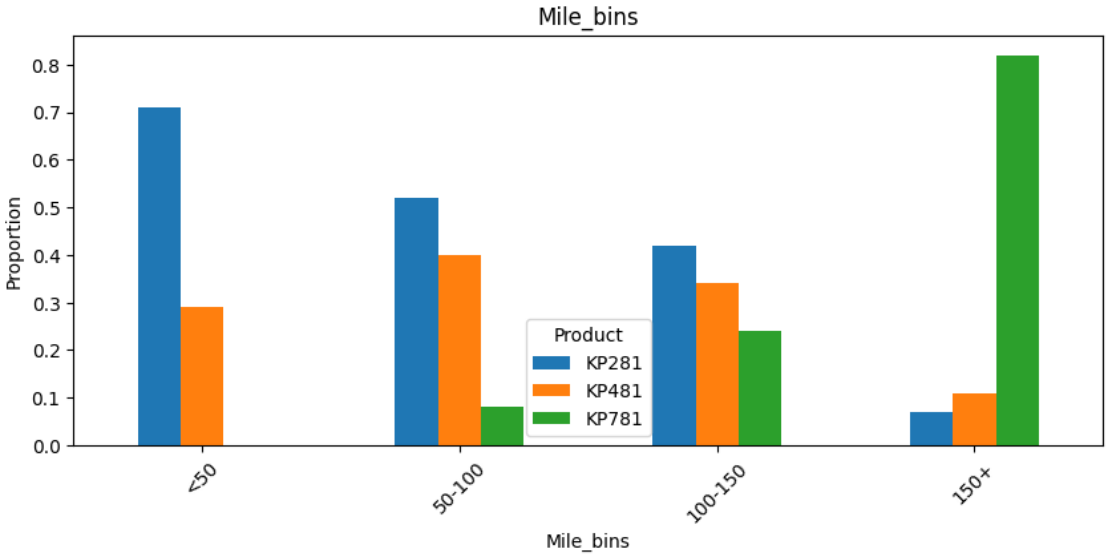

MaritalStatus
Product
KP281
KP481
KP781
Proportion
0.0
0.1
0.2
0.3
0.4
Partnered
Single
MaritalStatus


Usage
Product
KP281
KP481
KP781
Proportion
0.0
0.2
0.4
0.6
0.8
1.0
2
3
4
5
6
7
Usage


Fitness
Product
KP281
KP481
KP781
Proportion
0.0
0.2
0.4
0.6
0.8
1
2
3
4
5
Fitness


Age_bins
Product
KP281
KP481
KP781
Proportion
0.0
0.1
0.2
0.3
0.4
0.5
0.6
<20
20-25
25-30
30-35
35-40
40+
Age_bins





Observations on the basis of above Categorical Plots

1) Around 55% of women prefer KP281 and only 10% prefer KP781. While around 35% of men prefer KP781.

2) 80% in Eduation level of 18 and everyone in Education levels of 20 or 21 use KP781 while below 14 level, no one uses KP781.

3) Marital Status implies no significant information on the usages of different treadmills.

4) Those who workout 6 or 7 days a week use KP781 while 60% of those who workout 5 days a

week use KP781.

5) 95% of customers having fitness level of 5 use KP781 and none of those having fitness level below 3 use KP781.

6) No one below 20 years of age use KP781.

7) Above 70000 units of Income, people only use KP781 while in Incomes below 45000, no one uses KP781.

8) Almost 80% of people who run over 200 miles and those who run above 150 miles use KP781 and no one who runs below 50 miles use KP781. The usage of KP281 decreases with the increase in miles while that of KP781 increases with the increase in miles.

Conditional probabilities - are the probability that an event occurs given that another event has occurred. For example, given that a customer is female, what is the probability she'll purchase a Mac?

Marginal Probablities - Divide the row or column total by the total sample size. Marginal probabilities are the probabilities that a single event occurs with no regard to other events in the table. These probabilities do not depend on the condition of another outcome.

Probability(Mac) ; Probability(Female)

Normalize: Default value is False Normalize by dividing all values by the sum of values:

If passed 'all' or True, will normalize over all values.

If passed 'index' will normalize over each row.

If passed 'columns' will normalize over each column.

If margins is True, will also normalize margin values.

[41]:
```
pd.crosstab(index=df['Gender'],columns=df['Product'])
```

[41]:
```
Product  KP281  KP481  KP781
Gender
Female      40     29      7
Male        40     31     33
```

[42]:
```
pd.crosstab(index=df['Gender'],columns=df['Product'],margins=True)
```

[42]:
```
Product  KP281  KP481  KP781  All
Gender
Female      40     29      7   76
Male        40     31     33  104
All         80     60     40  180
```

[43]:
```
pd.
  ↪crosstab(index=df['Gender'],columns=df['Product'],margins=True,normalize=True)*100
#each element divided by 180
```

[43]:
Product	KP281	KP481	KP781	All
Gender				
Female	22.222222	16.111111	3.888889	42.222222
Male	22.222222	17.222222	18.333333	57.777778
All	44.444444	33.333333	22.222222	100.000000

[44]:
```
pd.
  ↪crosstab(index=df['Gender'],columns=df['Product'],margins=True,normalize='index')
#each element divided by summation of individual rows
```

[44]:
Product	KP281	KP481	KP781
Gender			
Female	0.526316	0.381579	0.092105
Male	0.384615	0.298077	0.317308
All	0.444444	0.333333	0.222222

Optimised code

[45]:
```
from IPython.display import display
for i in cat_cols:
  print('Table for',str(i),'vs Treadmill Product')
  display(pd.crosstab(df[i], df['Product'], margins=True, normalize='index'))
  print("\n")
```

Table for Gender vs Treadmill Product

Product	KP281	KP481	KP781
Gender			
Female	0.526316	0.381579	0.092105
Male	0.384615	0.298077	0.317308
All	0.444444	0.333333	0.222222

Table for Education vs Treadmill Product

Product	KP281	KP481	KP781
Education			
12	0.666667	0.333333	0.000000
13	0.600000	0.400000	0.000000
14	0.545455	0.418182	0.036364
15	0.800000	0.200000	0.000000
16	0.458824	0.364706	0.176471
18	0.086957	0.086957	0.826087
20	0.000000	0.000000	1.000000
21	0.000000	0.000000	1.000000
All	0.444444	0.333333	0.222222

Table for MaritalStatus vs Treadmill Product

Product	KP281	KP481	KP781
MaritalStatus			
Partnered	0.448598	0.336449	0.214953
Single	0.438356	0.328767	0.232877
All	0.444444	0.333333	0.222222

Table for Usage vs Treadmill Product

Product	KP281	KP481	KP781
Usage			
2	0.575758	0.424242	0.000000
3	0.536232	0.449275	0.014493
4	0.423077	0.230769	0.346154
5	0.117647	0.176471	0.705882
6	0.000000	0.000000	1.000000
7	0.000000	0.000000	1.000000
All	0.444444	0.333333	0.222222

Table for Fitness vs Treadmill Product

Product	KP281	KP481	KP781
Fitness			
1	0.500000	0.500000	0.000000
2	0.538462	0.461538	0.000000
3	0.556701	0.402062	0.041237
4	0.375000	0.333333	0.291667
5	0.064516	0.000000	0.935484
All	0.444444	0.333333	0.222222

Table for Age_bins vs Treadmill Product

Product	KP281	KP481	KP781
Age_bins			
<20	0.600000	0.400000	0.000000
20-25	0.405797	0.347826	0.246377
25-30	0.512195	0.170732	0.317073
30-35	0.343750	0.531250	0.125000
35-40	0.500000	0.375000	0.125000
40+	0.500000	0.166667	0.333333
All	0.444444	0.333333	0.222222

Table for Income_bins vs Treadmill Product

```
Product          KP281     KP481     KP781
Income_bins
<35000        0.571429  0.428571  0.000000
35000-45000   0.742857  0.257143  0.000000
45000-50000   0.411765  0.441176  0.147059
50000-60000   0.472727  0.418182  0.109091
60000-70000   0.315789  0.368421  0.315789
70000-90000   0.000000  0.000000  1.000000
90000+        0.000000  0.000000  1.000000
All           0.444444  0.333333  0.222222
```

```
Table for Mile_bins vs Treadmill Product
```

```
Product       KP281     KP481     KP781
Mile_bins
<50         0.705882  0.294118  0.000000
50-100      0.515464  0.402062  0.082474
100-150     0.421053  0.342105  0.236842
150+        0.071429  0.107143  0.821429
All         0.444444  0.333333  0.222222
```

Normalize parameter accept boolean values. It depicts the percentage of time each combination occurs (i.e. the marginal probability).

They normalize by dividing all values by the sum of values.

- If passed 'all' or True, will normalize over all values.
- If passed 'index' will normalize over each row.
- If passed 'columns' will normalize over each column.
- In case if margins is True, will also normalize margin values.

Brief depiction of Probabilities Inferred from the above tables

1) In all the tables, one can see the last row named All, it consists of the overall probabilities of purchases of those 3 treadmills, i.e. - Probability of purchase of KP281= 44.44%, KP481= 33.33% and KP781=22.22%
2) P(KP281|Education=12) = 66.66% and P(KP781|Education=18) = 82.6% P(KP781|Education=20) = P(KP781|Education=21) = 100%
3) P(KP281|Usage=2) = 57.57%, P(KP781|Usage=6)=P(KP781|Usage=7) = 100%
4) P(KP481|Fitness=2) = 46.15%
5) P(KP481|Age_bins=30-35) = 53.12%
6) P(KP781|Income>70000) = 100% and P(KP481|Income_bins=45000-50000) = 44.11%

7) P(KP281|Mile_bins<50) = 70.5% and P(KP781|Mile_bins>150)=82.1%

Till now we've got great insights for the customers peratining to KP781 while clarity regarding KP281 and KP481 seems to be missing

[46]:
```
df.head()
```

[46]:

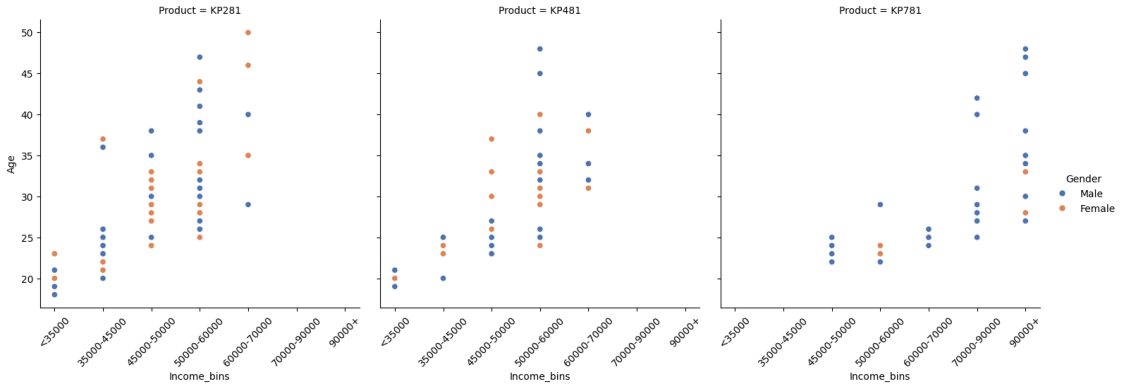
	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	\
0	KP281	18	Male	14	Single	3	4	29562	
1	KP281	19	Male	15	Single	2	3	31836	
2	KP281	19	Female	14	Partnered	4	3	30699	
3	KP281	19	Male	12	Single	3	3	32973	
4	KP281	20	Male	13	Partnered	4	2	35247	

	Miles	Age_bins	Income_bins	Mile_bins
0	112	<20	<35000	100-150
1	75	<20	<35000	50-100
2	66	<20	<35000	50-100
3	85	<20	<35000	50-100
4	47	<20	35000-45000	<50

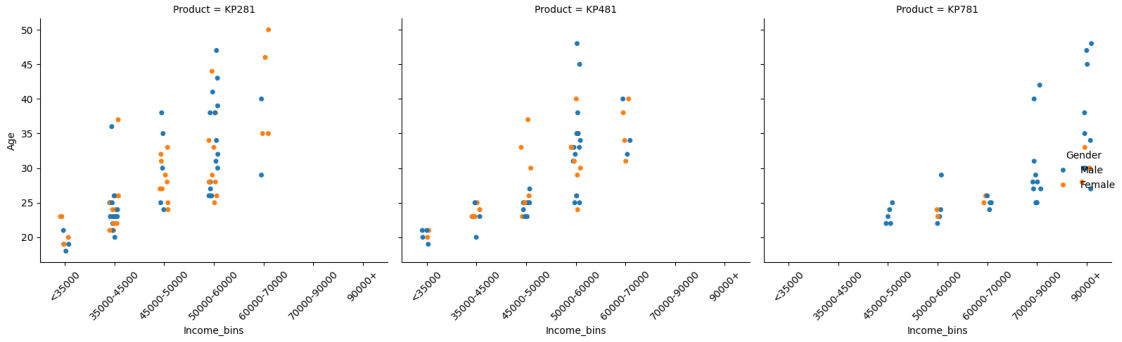
Multivariate Analysis using Scatterplots and Factorplots for different Products (predominately to understand target audience for KP281 and KP481 which appear quite similar).

[49]:
```
plot=sns.relplot(data=df, x='Income_bins', y='Age', col='Product',␣
 ↪hue='Gender', palette="deep", kind="scatter")
plt.figure(figsize=(10,6))
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
plt.tight_layout()
plot=sns.catplot(x ='Income_bins', y ='Age',
                 hue='Gender',col = 'Product', data = df)
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
plt.tight_layout()
plt.show()
```

```
<ipython-input-49-fdac430c3544>:4: UserWarning: FixedFormatter should only be
used together with FixedLocator
  _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
<ipython-input-49-fdac430c3544>:9: UserWarning: FixedFormatter should only be
used together with FixedLocator
  _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
```




```
<Figure size 1000x600 with 0 Axes>
```



For women having incomes below 70k, the average age of those who use KP281 is 40 while it's 35 for those who use KP481.

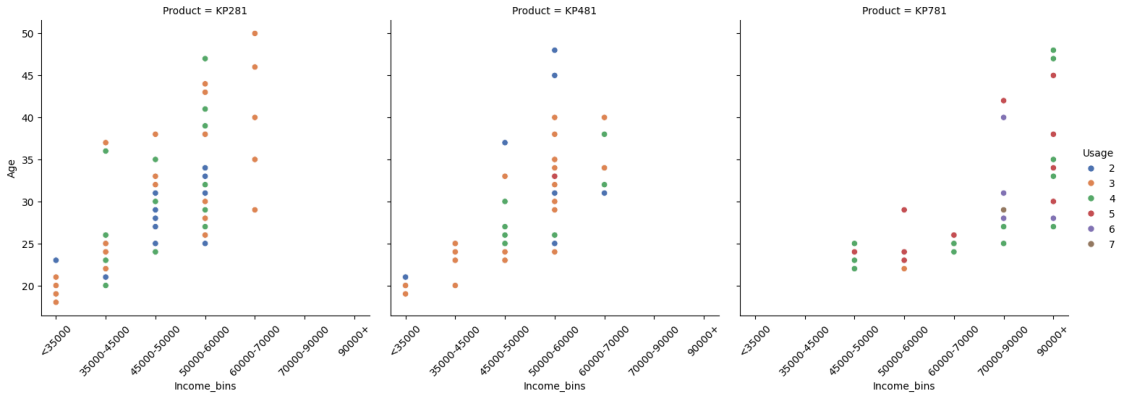
Only 2 women have incomes over 70k which is certainly the reason for a large proportion of them not buying KP781 (affordability).

Moreover the variances are higher in case of KP281. Though there are less data points for this observation, so it requires more data to be verified.

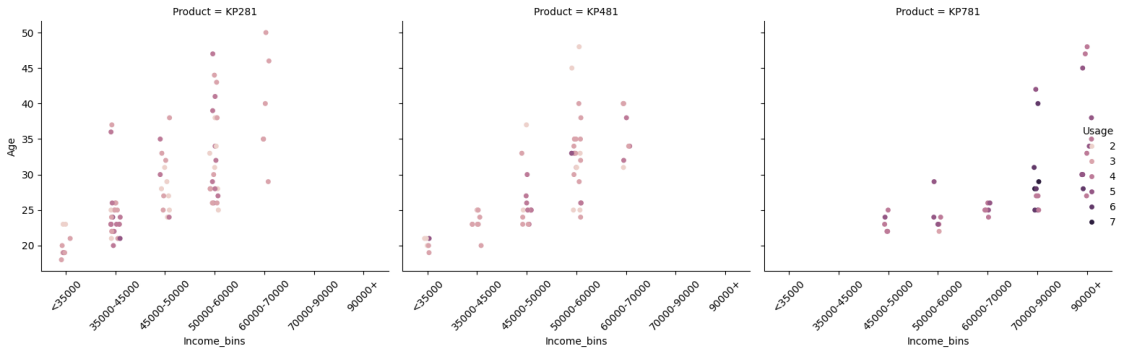
[51]:
```
plot=sns.relplot(data=df, x='Income_bins', y='Age', col='Product', hue='Usage',␣
↪palette="deep", kind="scatter")
plt.figure(figsize=(10,6))
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
plt.tight_layout()
plot=sns.catplot(x ='Income_bins', y ='Age',
                 hue='Usage',col = 'Product', data = df)
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
plt.tight_layout()
```

```
plt.show()
```

```
<ipython-input-51-2014331e5053>:4: UserWarning: FixedFormatter should only be
used together with FixedLocator
  _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
<ipython-input-51-2014331e5053>:9: UserWarning: FixedFormatter should only be
used together with FixedLocator
  _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
```



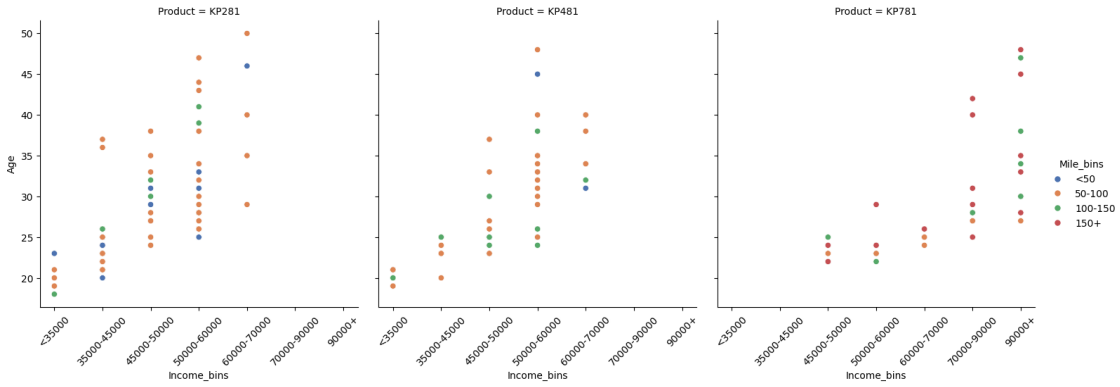

```
<Figure size 1000x600 with 0 Axes>
```



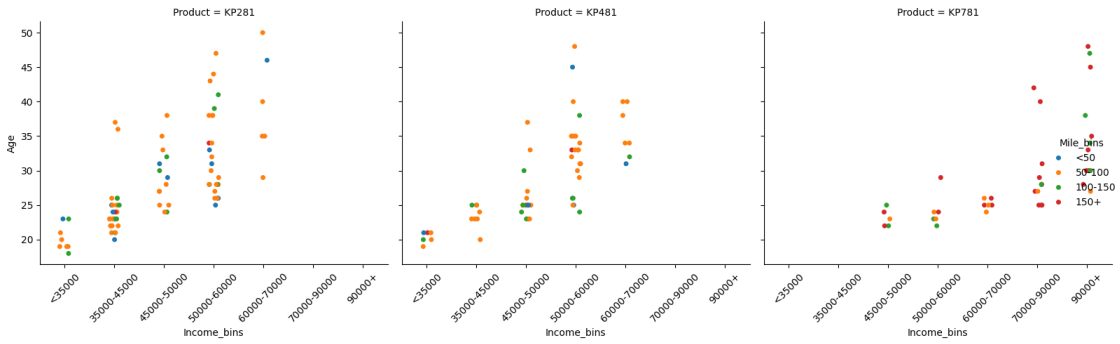
1) For Usage=3 and Income in the range 60k-70k, we are very much certain of the user to be buying the KP281 Treadmill.
2) For Usage=2 and Income in the range 45k-50k, we are very much certain of the user to be buying the KP281 Treadmill.
3) For income range 45k-50k and Usage=4, we are very much certain of the user to be buying KP281 Treadmill.
4) For income range 50k-60k and Usage=4, we are very much certain of the user to be buying KP281 Treadmill.

[52]:
```
plot=sns.relplot(data=df, x='Income_bins', y='Age', col='Product',␣
 ↪hue='Mile_bins', palette="deep", kind="scatter")
plt.figure(figsize=(10,6))
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
plt.tight_layout()
plot=sns.catplot(x ='Income_bins', y ='Age',
                 hue='Mile_bins',col = 'Product', data = df)
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
plt.tight_layout()
plt.show()
```

```
<ipython-input-52-33009d92c6aa>:4: UserWarning: FixedFormatter should only be
used together with FixedLocator
  _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
<ipython-input-52-33009d92c6aa>:9: UserWarning: FixedFormatter should only be
used together with FixedLocator
  _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
```



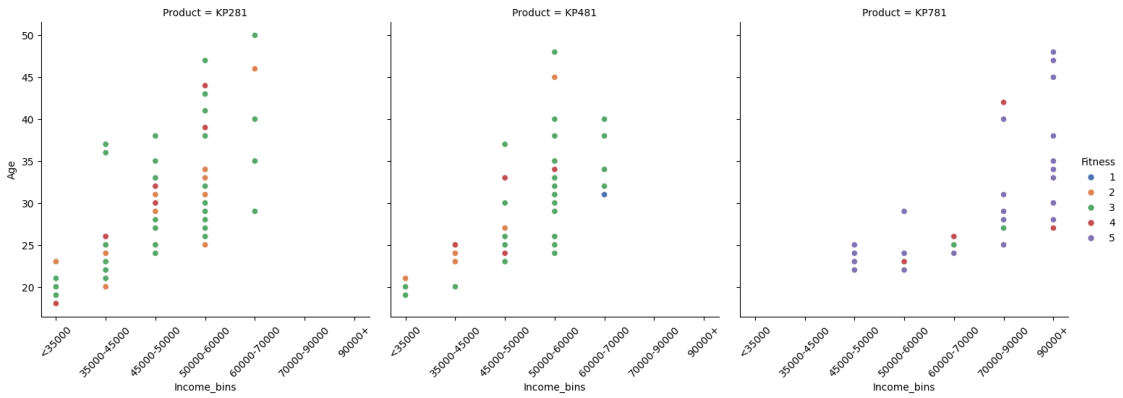

```
<Figure size 1000x600 with 0 Axes>
```



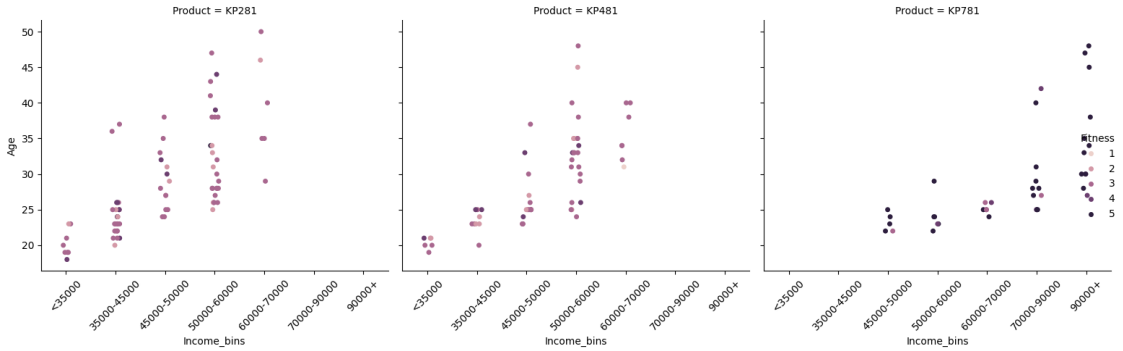
For no. of miles in range 100-150, those customers whose incomes are in range of 50k-60k and age between 25 to 30 tend to use KP481.

[54]:
```
plot=sns.relplot(data=df, x='Income_bins', y='Age', col='Product',␣
 ↪hue='Fitness', palette="deep", kind="scatter")
plt.figure(figsize=(10,6))
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
plt.tight_layout()
plot=sns.catplot(x ='Income_bins', y ='Age',
                 hue='Fitness',col = 'Product', data = df)
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
plt.tight_layout()
plt.show()
```

```
<ipython-input-54-885da73d0585>:4: UserWarning: FixedFormatter should only be
used together with FixedLocator
  _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
<ipython-input-54-885da73d0585>:9: UserWarning: FixedFormatter should only be
used together with FixedLocator
  _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
```




```
<Figure size 1000x600 with 0 Axes>
```

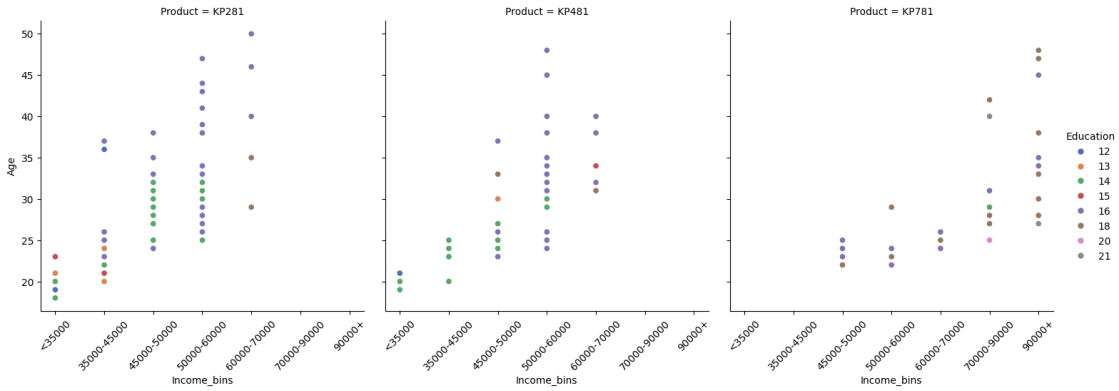


Customers with around 40 years of age, having fitness level of 4 and incomes in the range 50k-60k tend to use KP281.

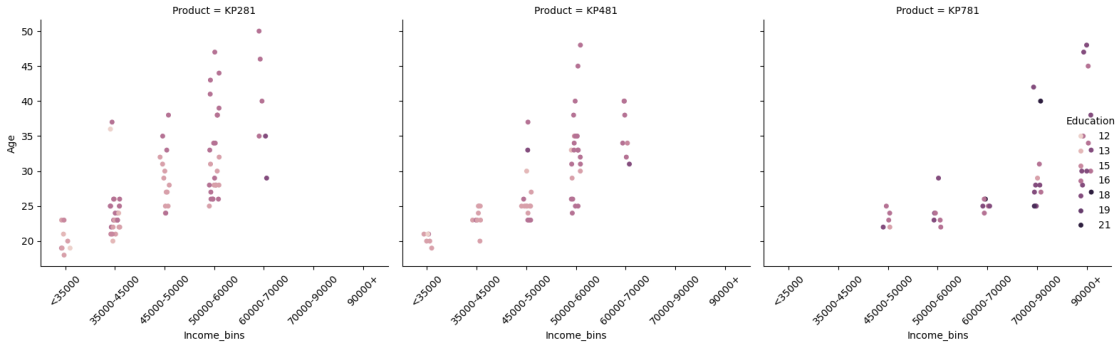
Customers with around 25-32 years of age and having fitness level of 4 with incomes in the range 50k-60k tend to use KP481.

[56]:
```
plot=sns.relplot(data=df, x='Income_bins', y='Age', col='Product',␣
 ↪hue='Education', palette="deep", kind="scatter")
plt.figure(figsize=(10,6))
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
plt.tight_layout()
plot=sns.catplot(x ='Income_bins', y ='Age',
                 hue='Education',col = 'Product', data = df)
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
plt.tight_layout()
plt.show()
```

```
<ipython-input-56-c3995913cff6>:4: UserWarning: FixedFormatter should only be
used together with FixedLocator
  _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
<ipython-input-56-c3995913cff6>:9: UserWarning: FixedFormatter should only be
used together with FixedLocator
  _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
```




```
<Figure size 1000x600 with 0 Axes>
```



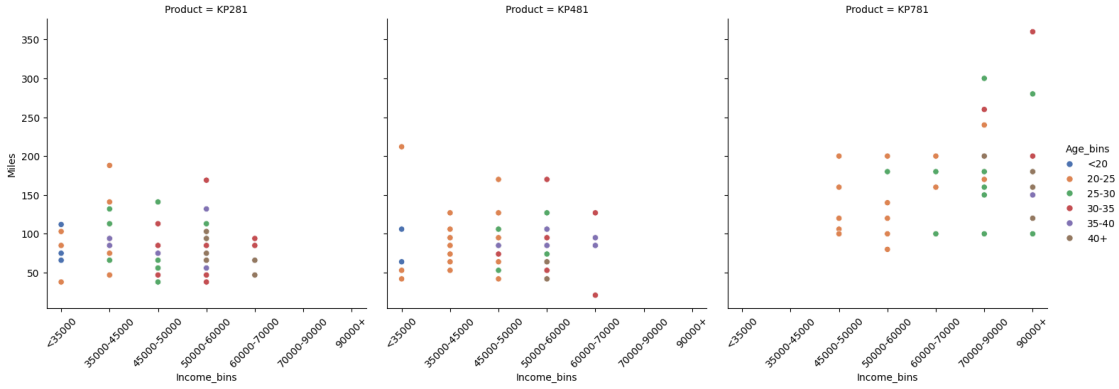
For Education level of 16, above 32 years of age with Income between 45k-50k will tend to use KP281 and below 22 will tend to use KP481.

Also for the same Education level customers but in Income range 60k-70k, above 45 years of age will tend to use KP281 while customers below 35 years of age will tend to use KP481.

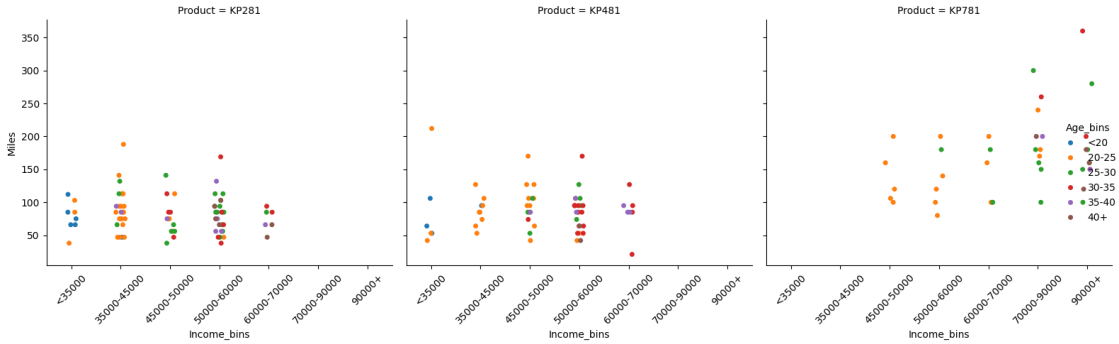
[57]:
```
plot=sns.relplot(data=df, x='Income_bins', y='Miles', col='Product',␣
↪hue='Age_bins', palette="deep", kind="scatter")
plt.figure(figsize=(10,6))
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
plt.tight_layout()
plot=sns.catplot(x ='Income_bins', y ='Miles',
                 hue='Age_bins',col = 'Product', data = df)
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
plt.tight_layout()
plt.show()
```

```
<ipython-input-57-a3482cf33653>:4: UserWarning: FixedFormatter should only be
```

```
used together with FixedLocator
  _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
<ipython-input-57-a3482cf33653>:9: UserWarning: FixedFormatter should only be
used together with FixedLocator
  _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
```




```
<Figure size 1000x600 with 0 Axes>
```



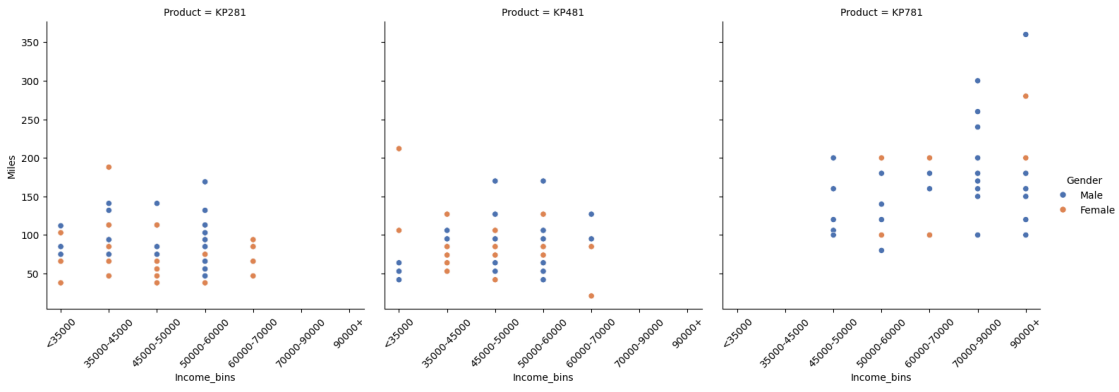
1) All 25-30 and 35-40 years old individuals having incomes in the range of 35k-45k tend to buy KP281.

2) Individuals having 40+ years of age and income ranges in 60k-70k are more likely to use KP281 while the individuals of same income range but 35-40 years of age tend to use KP481.

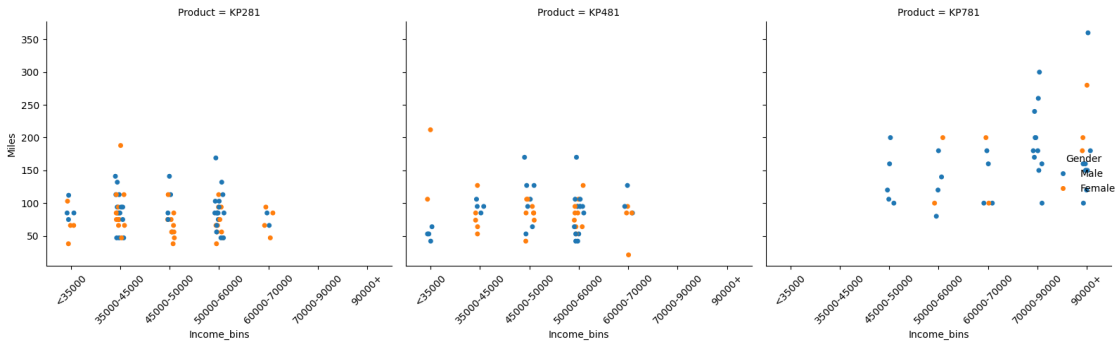
[58]:
```
plot=sns.relplot(data=df, x='Income_bins', y='Miles', col='Product',␣
 ↪hue='Gender', palette="deep", kind="scatter")
plt.figure(figsize=(10,6))
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
plt.tight_layout()
plot=sns.catplot(x ='Income_bins', y ='Miles',
                 hue='Gender',col = 'Product', data = df)
```

```
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
plt.tight_layout()
plt.show()
```

```
<ipython-input-58-47b23ec3b7d0>:4: UserWarning: FixedFormatter should only be
used together with FixedLocator
  _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
<ipython-input-58-47b23ec3b7d0>:9: UserWarning: FixedFormatter should only be
used together with FixedLocator
  _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
```




```
<Figure size 1000x600 with 0 Axes>
```



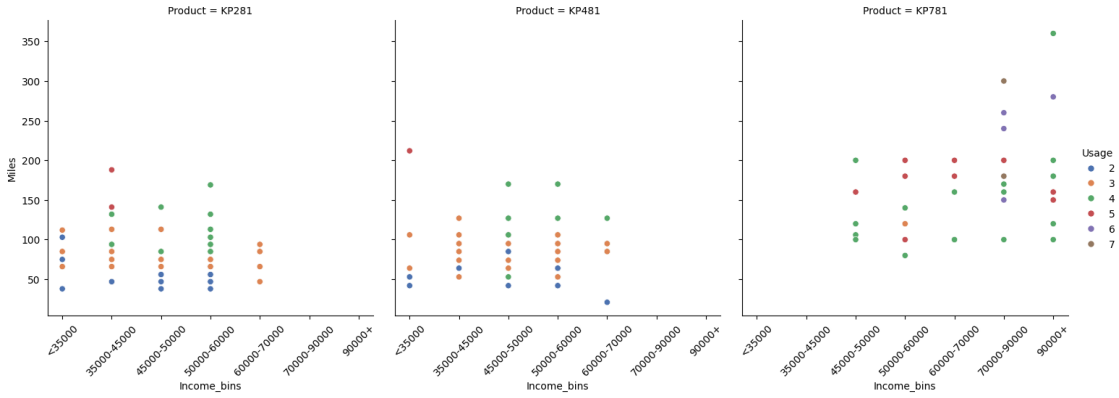
1) For income level below 35k, women who tread over 105 miles tend to use KP481 while those who tread below 105 tend to use KP281.

2) Men with income level in 60k-70k, those who run in the range of 100-150 miles tend to use KP481.

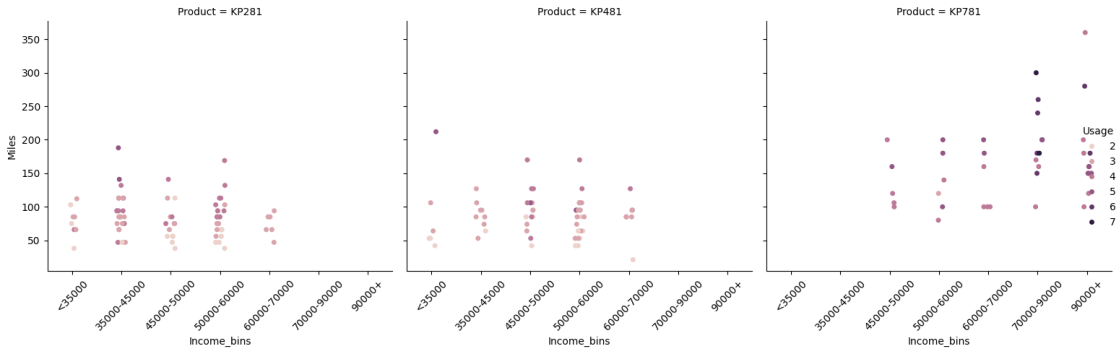
[59]:
```
plot=sns.relplot(data=df, x='Income_bins', y='Miles', col='Product',␣
↪hue='Usage', palette="deep", kind="scatter")
```

```
plt.figure(figsize=(10,6))
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
plt.tight_layout()
plot=sns.catplot(x ='Income_bins', y ='Miles',
                 hue='Usage',col = 'Product', data = df)
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
plt.tight_layout()
plt.show()
```

```
<ipython-input-59-1d19dd8a6a37>:4: UserWarning: FixedFormatter should only be
used together with FixedLocator
  _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
<ipython-input-59-1d19dd8a6a37>:9: UserWarning: FixedFormatter should only be
used together with FixedLocator
  _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
```




```
<Figure size 1000x600 with 0 Axes>
```

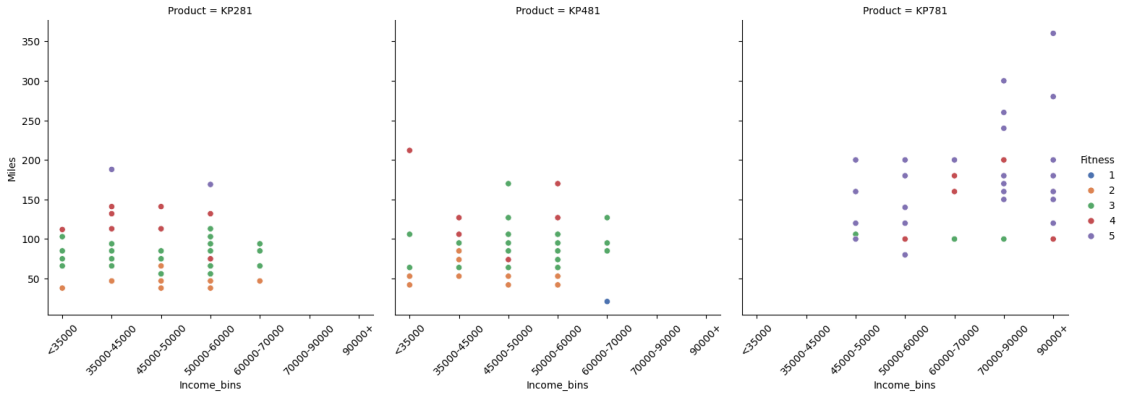


For customers with 5 days usage in a week and incomes in range 35k-45k, if they run more than

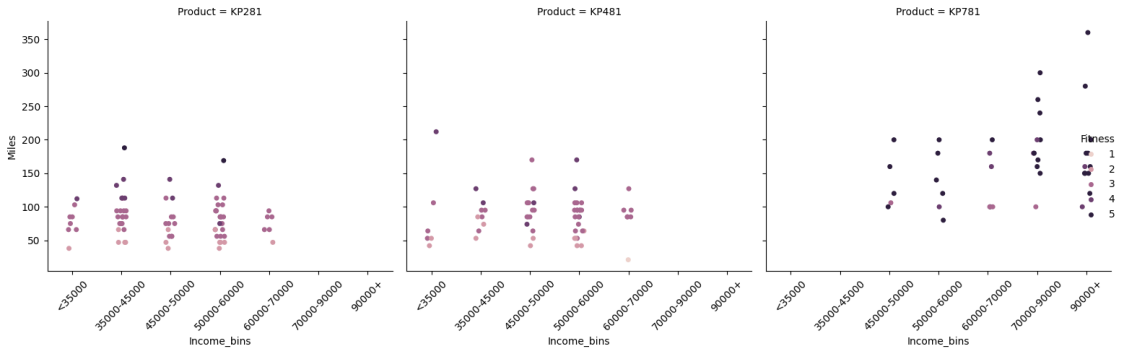
140 miles, they tend to use KP281.

[60]:
```
plot=sns.relplot(data=df, x='Income_bins', y='Miles', col='Product',␣
 ↪hue='Fitness', palette="deep", kind="scatter")
plt.figure(figsize=(10,6))
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
plt.tight_layout()
plot=sns.catplot(x ='Income_bins', y ='Miles',
                 hue='Fitness',col = 'Product', data = df)
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
plt.tight_layout()
plt.show()
```

```
<ipython-input-60-5ad39fef4f26>:4: UserWarning: FixedFormatter should only be
used together with FixedLocator
  _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
<ipython-input-60-5ad39fef4f26>:9: UserWarning: FixedFormatter should only be
used together with FixedLocator
  _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
```




```
<Figure size 1000x600 with 0 Axes>
```

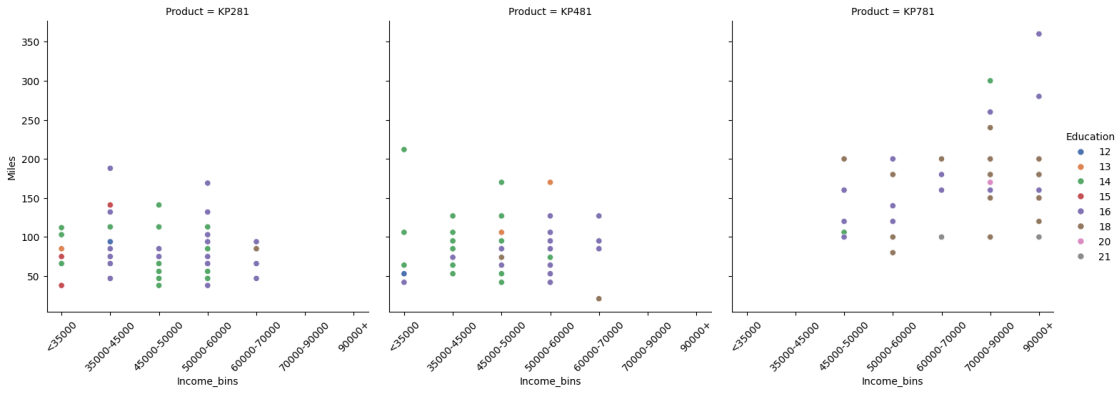


1) If Fitness level=4 and incomes between 50k-60k, then these customers will tend to use KP281

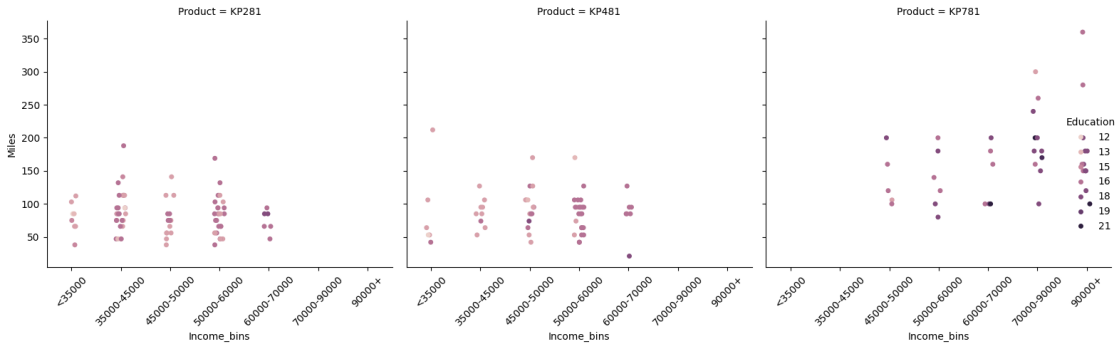
2) If Fitness level=4 in Income level of 50k-60k, if the person runs more than 100 miles, they tend to use KP481.

[61]:
```
plot=sns.relplot(data=df, x='Income_bins', y='Miles', col='Product',␣
 ↪hue='Education', palette="deep", kind="scatter")
plt.figure(figsize=(10,6))
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
plt.tight_layout()
plot=sns.catplot(x ='Income_bins', y ='Miles',
                 hue='Education',col = 'Product', data = df)
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
plt.tight_layout()
plt.show()
```

```
<ipython-input-61-492ebe79b208>:4: UserWarning: FixedFormatter should only be
used together with FixedLocator
  _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
<ipython-input-61-492ebe79b208>:9: UserWarning: FixedFormatter should only be
used together with FixedLocator
  _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
```



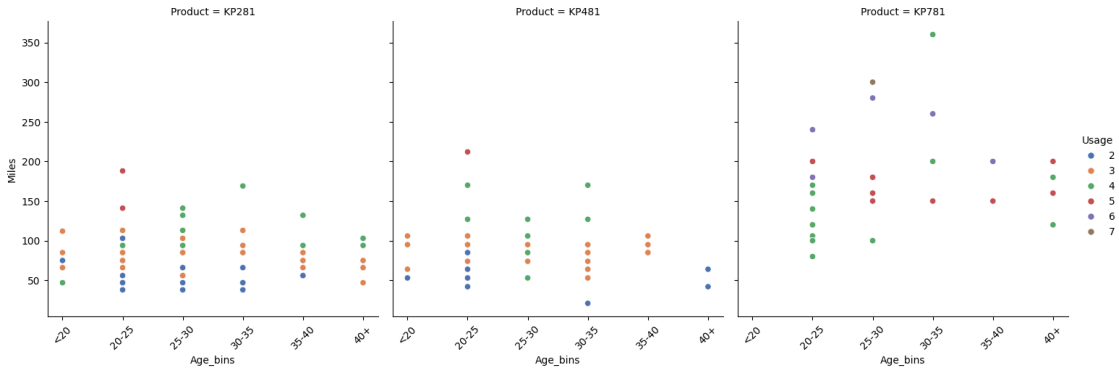

```
<Figure size 1000x600 with 0 Axes>
```



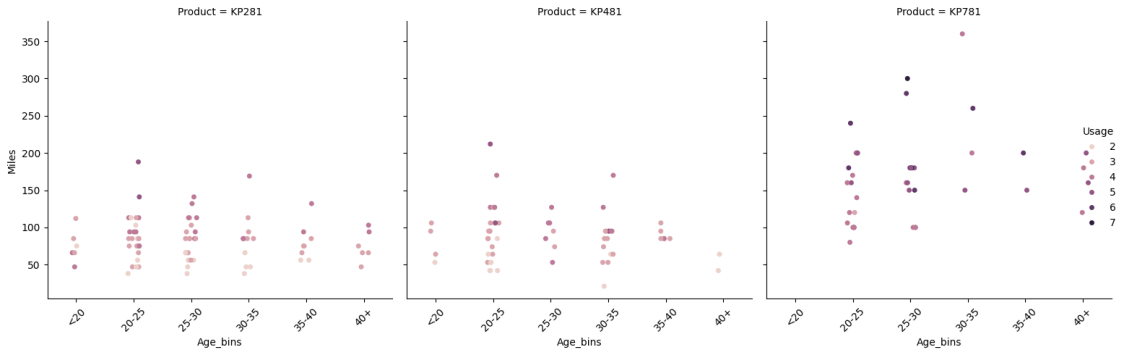
Customers with Education level of 13 and in Income range of 45k-60k will tend to use KP481 while those with Education level of 15 and below 35k income will tend to use KP281.

[63]:
```
plot=sns.relplot(data=df, x='Age_bins', y='Miles', col='Product', hue='Usage',␣
 ↪palette="deep", kind="scatter")
plt.figure(figsize=(10,6))
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
plt.tight_layout()
plot=sns.catplot(x ='Age_bins', y ='Miles',
                 hue='Usage',col = 'Product', data = df)
for axes in plot.axes.flat:
    _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
plt.tight_layout()
plt.show()
```

```
<ipython-input-63-1de63bfe100d>:4: UserWarning: FixedFormatter should only be
used together with FixedLocator
  _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
<ipython-input-63-1de63bfe100d>:9: UserWarning: FixedFormatter should only be
used together with FixedLocator
  _ = axes.set_xticklabels(axes.get_xticklabels(), rotation=45)
```



<Figure size 1000x600 with 0 Axes>



1) 40+ age users with 2 usages per week are using KP481 and 40+ age users with 3 usages per week who tread below 70 miles use KP281.

2) Users with ages 25-30 and 2 usages per week use KP281.

##Recommendations (10 Points) - Actionable items for business. No technical jargon. No complications. Simple action items that everyone can understand

##Business Insights

Customer Profiles for KP781

1) Only people having incomes greater than 70k have run over 220 miles and all of then use KP781.

2) Recommend KP781 if one or more conditions are satisfied along with a necessary condition of Income > 70000:-

    a) Education Level >= 18
    b) Usage days > = 5
    c) Fitness Levels = 5
    d) The person runs more than 150 miles(80% of them use KP781)

3) Never Recommend KP781 if one or more of these conditions are satisfied:-

    a) Education Levels < 14
    b) Fitness < 3
    c) Age < 20 d)Income < 45000
    d) Miles run < 50

Why very few women have bought the luxurious KP781 treadmill? Only 2 women have incomes over 70k which is certainly the reason for a large proportion of them not buying KP781(affordability).

Note for below mentioned points KP281 and KP481 don't have much differences in their costs and the characteristics of customers who use them . Still a few of them have been identified but they need to be validated with an incremental data.

Customer Profiles for KP281:

1) Women having incomes below 70k and age > 40
2) Customers having income in range 60k-70k and usage days=3
3) Customers having income in range 45k-50k and usage days=2
4) Customers having income in range 35k-45k and usage days=4
5) Customers having income in range 50k-60k and usage days=4
6) Customers with Fitness=4, age closer to 40 and income 50k-60k
7) Customers with Education Level=16, Age>32 and income 45k-50k
8) Customers with Education Level=16, Age>45 and income 60k-70k
9) Customers with Age in 25-30 and 35-40 having incomes in range 35k-45k
10) Customers with 40+ Age and 60k-70k income
11) Women with incomes < 35k and whose miles run < 105
12) Customers with usages=5, incomes in range 35k-45k and who run more than 140 miles
13) Customers with Fitness=5, incomes < 70k and Incomes in 45k-50k
14) Customers with Education level=15 having incomes less than 35k
15) Customers with Usages=3, miles run < 70 and Age>40
16) Customers with Usages=2 and Age between 25-30

Customer Profiles for KP481:

1) Women having incomes below 70k and age between 32-37
2) Customers with age < 25, incomes in range 50-60k and the miles run is in the range 100-150
3) Customers with Fitness=4, age in range 25-32 and income 50k-60k
4) Customers with Education Level=16, Age< 22 and income 45k-50k
5) Customers with Education Level=16, Age< 35 and income 60k-70k
6) Customers with 35-40 Age and 60k-70k income
7) Women with incomes < 35k and whose miles run >105
8) Men with incomes 60k-70k and who tread in range 100-150 miles
9) Customers with Fitness=4, incomes < 45k-50k and who run more than 100 miles
10) Customers with Education level=13 having incomes in ranges 45-60k
11) Customers with Usages=2 and Age>40