Aero t : Descriptive Statistics & Probability

About AeroFit

About Aero t

Aero t is a leading brand in the eld of tness equipment. Aero t provides a product range including machines such as treadmills, exercise bikes, gym equipment, and tness accessories to cater to the needs of all categories of people.

Problem Statement :

The market research team at AeroFit wants to identify the characteristics of the target audience for each type of treadmill o ered by the company, to provide a better recommendation of the treadmills to the new customers. The team decides to investigate whether there are di erences across the product with respect to customer characteristics.

Goal

Perform descriptive analytics to create a customer pro le for each AeroFit treadmill product by developing appropriate tables and charts.

Importing the required libraries for the analysis

In [1]:
```
# Importing all the necessary libraries needed for computing, visualizing, model building.

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing the Dataset

In [3]:
```
!gdown 15lzphg3S4M5URVxqgtP_VST7Y0y-rMGW
```

```
Downloading...
From: https://drive.google.com/uc?id=15lzphg3S4M5URVxqgtP_VST7Y0y-rMGW
To: /content/aerofit_treadmill.csv
100% 7.28k/7.28k [00:00<00:00, 18.4MB/s ]
```

Basic Analysis : Getting to know the dataset

In [5]:
```
df = pd.read_csv('aerofit_treadmill.csv')
print(f"The dataset is of the shape {df.shape}.")
df
```

```
The dataset is of the shape (180, 9).
```

Out[5]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	KP281	18	Male	14	Single	3	4	29562	112
1	KP281	19	Male	15	Single	2	3	31836	75
2	KP281	19	Female	14	Partnered	4	3	30699	66
3	KP281	19	Male	12	Single	3	3	32973	85
4	KP281	20	Male	13	Partnered	4	2	35247	47
...	...	...	...	...	...	...	...	...	...
175	KP781	40	Male	21	Single	6	5	83416	200
176	KP781	42	Male	18	Single	5	4	89641	200
177	KP781	45	Male	16	Single	5	5	90886	160
178	KP781	47	Male	18	Partnered	4	5	104581	120
179	KP781	48	Male	18	Partnered	4	5	95508	180

180 rows × 9 columns

In [6]:
```
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
```

Here we can con rm that the dataset consist of no null values since all the columns have 180 non-null values.

Also, the 'Product' is going to be our target/dependent variable for the analysis.

In [7]:
```
# To check presence of any duplicate records.
np.any(df.duplicated())
```

Out[7]: False

In [8]:
```
# Checking the amount of null values in all the columns.
df.isnull().sum()
```

Out[8]:
```
Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
Usage            0
Fitness          0
Income           0
Miles            0
dtype: int64
```

The dataset thus consists of no null values.

In [9]:
```
# Filtering the numerical columns
num_cols = df.select_dtypes(np.number).columns
num_cols
```

Out[9]: Index(['Age', 'Education', 'Usage', 'Fitness', 'Income', 'Miles'], dtype='object')

In [10]:
```
# Finding number of unique values from the numerical columns
for col in num_cols:
    print(f"Number of unique values in {col} = {df[col].nunique()}\n")
```

```
Number of unique values in Age = 32

Number of unique values in Education = 8

Number of unique values in Usage = 6

Number of unique values in Fitness = 5

Number of unique values in Income = 62

Number of unique values in Miles = 37
```

In [11]:
```
# Changing the data types of few columns to category which will save us some memory and reduce the execution time
(cat_cols := df.columns[~df.columns.isin(num_cols)])
df[cat_cols] = df[cat_cols].astype('category')
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    category
 1   Age            180 non-null    int64
 2   Gender         180 non-null    category
 3   Education      180 non-null    int64
 4   MaritalStatus  180 non-null    category
 5   Usage          180 non-null    int64
 6   Fitness        180 non-null    int64
 7   Income         180 non-null    int64
 8   Miles          180 non-null    int64
dtypes: category(3), int64(6)
memory usage: 9.5 KB
```

We can see that Product, Gender and MaritalStatus columns in the following info has been changed into category type.

In [12]:
```
# Get unique values and value counts of non-numerical columns
for col in cat_cols:
    print(f"Unique values of {col}:- {df[col].unique()}\n\nValue_counts of {col}:-\n{df[col].value_counts()}\n",
end = '----------------------------------------------------------------------------------\n')
```

```
Unique values of Product:- ['KP281', 'KP481', 'KP781']
Categories (3, object): ['KP281', 'KP481', 'KP781']

Value_counts of Product:-
Product
KP281    80
KP481    60
KP781    40
Name: count, dtype: int64
----------------------------------------------------------------------------------
Unique values of Gender:- ['Male', 'Female']
Categories (2, object): ['Female', 'Male']

Value_counts of Gender:-
Gender
Male      104
Female     76
Name: count, dtype: int64
----------------------------------------------------------------------------------
Unique values of MaritalStatus:- ['Single', 'Partnered']
Categories (2, object): ['Partnered', 'Single']

Value_counts of MaritalStatus:-
MaritalStatus
Partnered    107
Single        73
Name: count, dtype: int64
----------------------------------------------------------------------------------
```

In [13]:
```
pd.set_option('display.float_format', lambda x: '%.2f' % x)
(df_numerical_description := df.describe())
```

Out[13]:

	Age	Education	Usage	Fitness	Income	Miles
count	180.00	180.00	180.00	180.00	180.00	180.00
mean	28.79	15.57	3.46	3.31	53719.58	103.19
std	6.94	1.62	1.08	0.96	16506.68	51.86
min	18.00	12.00	2.00	1.00	29562.00	21.00
25%	24.00	14.00	3.00	3.00	44058.75	66.00
50%	26.00	16.00	3.00	3.00	50596.50	94.00
75%	33.00	16.00	4.00	4.00	58668.00	114.75
max	50.00	21.00	7.00	5.00	104581.00	360.00

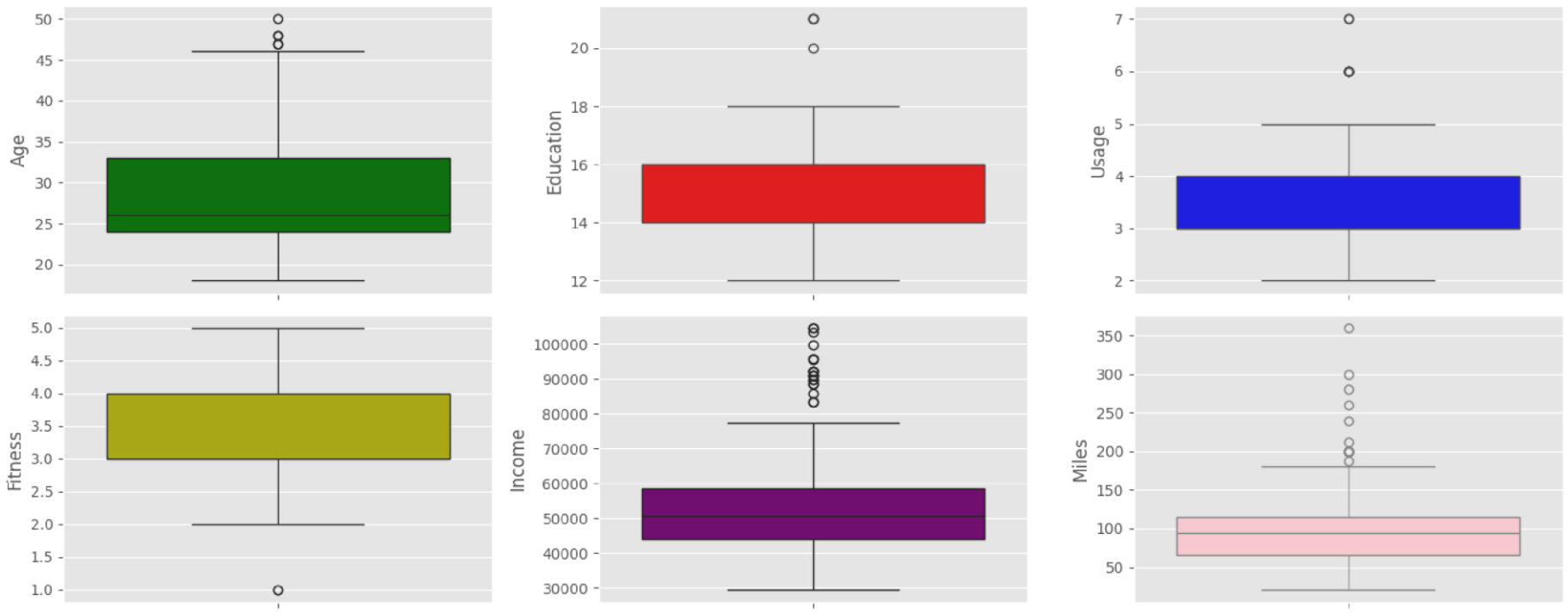
Outlier Detection and Processing

In [15]:
```
# Substracting mean from median to find the outliers.
# Greater the number deviates from 0, the more outliers present in the columns,
# which we can easily verify using the boxplot

df_numerical_description.loc['50%'] - df_numerical_description.loc['mean']
```

Out[15]:
```
Age            -2.79
Education       0.43
Usage          -0.46
Fitness        -0.31
Income      -3123.08
Miles          -9.19
dtype: float64
```

We can see that the 'Income' and the 'Miles' columns have high mean-median deviation, thereby we can say that they have higher number of outliers.

In [16]:
```
# Figuring out the outliers present in the numerical columns
with plt.style.context('ggplot'):
    fig = plt.figure(figsize=(15,6))

    plt.subplot(2,3,1)
    sns.boxplot(data=df,y='Age',color = 'g')

    plt.subplot(2,3,2)
    sns.boxplot(data=df,y='Education', color = 'r')

    plt.subplot(2,3,3)
    sns.boxplot(data=df,y='Usage', color = 'b')

    plt.subplot(2,3,4)
    sns.boxplot(data=df,y='Fitness', color = 'y')

    plt.subplot(2,3,5)
    sns.boxplot(data=df,y='Income',color='purple')

    plt.subplot(2,3,6)
    sns.boxplot(data=df,y='Miles',color='pink')

    plt.tight_layout()
```



Observing the plot further cements our prior nding about the number of outliers in the 'Income' and the 'Miles' columns.

Let us create a dataframe comprising of all the relevant statistical metrics for each column.

In [17]:
```
def find_outliers(df):
    q1=df.quantile(0.25)
    q3=df.quantile(0.75)
    iqr = q3-q1
    median = df.median()
    lower_fence = round(q1-1.5*iqr, 2)
    upper_fence = round(q1+1.5*iqr, 2)
    min_ = df.min()
    max_ = df.max()
    outliers = df[((df<(q1-1.5*iqr)) | (df>(q3+1.5*iqr)))].to_list()

    return q1, q3, iqr, median, lower_fence, upper_fence, min_, max_, len(outliers), outliers
```

In [18]:
```
# Create a new dataframe for our stats data of columns.
stats_df = pd.DataFrame(data = None, columns = num_cols)

# Creating a new column for index. Will later set it as the index column.
stats_df['stats_index'] = ['q1', 'q3', 'iqr', 'median', 'lower_fence', 'upper_fence', 'min_', 'max_',
'outlier_count']
stats_df.set_index('stats_index', inplace = True)

# Capturing the outliers of all the columns in a dictionary.
outliers_dict = dict()

for col in stats_df.columns:
  col_result = find_outliers(df[col])
  stats_df[col] = col_result[:9]
  outliers_dict[col] = col_result[-1]

stats_df
```

Out[18]:

stats_index	Age	Education	Usage	Fitness	Income	Miles

q1	24.00	14.00	3.00	3.00	44058.75	66.00
q3	33.00	16.00	4.00	4.00	58668.00	114.75
iqr	9.00	2.00	1.00	1.00	14609.25	48.75
median	26.00	16.00	3.00	3.00	50596.50	94.00
lower_fence	10.50	11.00	1.50	1.50	22144.88	-7.12
upper_fence	37.50	17.00	4.50	4.50	65972.62	139.12
min_	18.00	12.00	2.00	1.00	29562.00	21.00
max_	50.00	21.00	7.00	5.00	104581.00	360.00
outlier_count	5.00	4.00	9.00	2.00	19.00	13.00

In [19]:
```
# Printing the Outlier details.

for col in outliers_dict: print(f'The {len(outliers_dict[col])} outliers in the \'{col}\' column are :
{outliers_dict[col]}')
```

```
The 5 outliers in the 'Age' column are : [47, 50, 48, 47, 48]
The 4 outliers in the 'Education' column are : [20, 21, 21, 21]
The 9 outliers in the 'Usage' column are : [6, 6, 6, 7, 6, 7, 6, 6, 6]
The 2 outliers in the 'Fitness' column are : [1, 1]
The 19 outliers in the 'Income' column are : [83416, 88396, 90886, 92131, 88396, 85906, 90886, 103336, 99601, 89641,
95866, 92131, 92131, 104581, 83416, 89641, 90886, 104581, 95508] The 13 outliers in the 'Miles' column are : [188, 212,
200, 200, 200, 240, 300, 280, 260, 200, 360, 200, 200]
```

We now have a dataframe that consists of all the essential parameters of each numerical column.

However, if we look carefully at the 'lower_fence' of the 'Miles' column, we see that it is negative. But a person cannot plan to walk negative miles. So clearly that needs to be handled.

We'll replace it with the minimum value of the miles column.

In [20]:
```
# Replacing the lower_fence for the 'Miles' column with the minimum value in the column.

stats_df.loc['lower_fence', 'Miles'] = df['Miles'].min()
```

In [21]:
```
stats_df
```

Out[21]:

stats_index	Age	Education	Usage	Fitness	Income	Miles
q1	24.00	14.00	3.00	3.00	44058.75	66.00
q3	33.00	16.00	4.00	4.00	58668.00	114.75
iqr	9.00	2.00	1.00	1.00	14609.25	48.75
median	26.00	16.00	3.00	3.00	50596.50	94.00
lower_fence	10.50	11.00	1.50	1.50	22144.88	21.00
upper_fence	37.50	17.00	4.50	4.50	65972.62	139.12
min_	18.00	12.00	2.00	1.00	29562.00	21.00
max_	50.00	21.00	7.00	5.00	104581.00	360.00
outlier_count	5.00	4.00	9.00	2.00	19.00	13.00

In order to reduce the outliers impact on the rest of the data, let us clip the data between the 5th percentile and the 95th percentile.

Dataset clipping

In [22]:
```
clipped_df = df
```

Out[23]:

In [23]:
```
for col in stats_df.columns:
  clipped_df[col] = np.clip(clipped_df[col], clipped_df[col].quantile(0.05), clipped_df[col].quantile(0.95))

clipped_df
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	KP281	20.00	Male	14	Single	3.00	4	34053.15	112
1	KP281	20.00	Male	15	Single	2.00	3	34053.15	75
2	KP281	20.00	Female	14	Partnered	4.00	3	34053.15	66
3	KP281	20.00	Male	14	Single	3.00	3	34053.15	85
4	KP281	20.00	Male	14	Partnered	4.00	2	35247.00	47
...	...	...	...	...	...	...	...	...	...
175	KP781	40.00	Male	18	Single	5.05	5	83416.00	200
176	KP781	42.00	Male	18	Single	5.00	4	89641.00	200
177	KP781	43.05	Male	16	Single	5.00	5	90886.00	160
178	KP781	43.05	Male	18	Partnered	4.00	5	90948.25	120
179	KP781	43.05	Male	18	Partnered	4.00	5	90948.25	180

180 rows × 9 columns

In the resulting clipped_df, we can observe that post the calculations, the columns 'Age' and 'Usage' have oating point values which do not make sense for the type of data that they represent.

We will have to rectify that data to only contain `int` values for these columns.

In [24]:
```
# Converting the 'Age' & 'Usage' to 'int'
clipped_df[['Age', 'Usage']] = clipped_df[['Age', 'Usage']].astype(int)
clipped_df
```

Out[24]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	KP281	20	Male	14	Single	3	4	34053.15	112
1	KP281	20	Male	15	Single	2	3	34053.15	75
2	KP281	20	Female	14	Partnered	4	3	34053.15	66
3	KP281	20	Male	14	Single	3	3	34053.15	85
4	KP281	20	Male	14	Partnered	4	2	35247.00	47
...	...	...	...	...	...	...	...	...	...
175	KP781	40	Male	18	Single	5	5	83416.00	200
176	KP781	42	Male	18	Single	5	4	89641.00	200
177	KP781	43	Male	16	Single	5	5	90886.00	160
178	KP781	43	Male	18	Partnered	4	5	90948.25	120
179	KP781	43	Male	18	Partnered	4	5	90948.25	180

180 rows × 9 columns

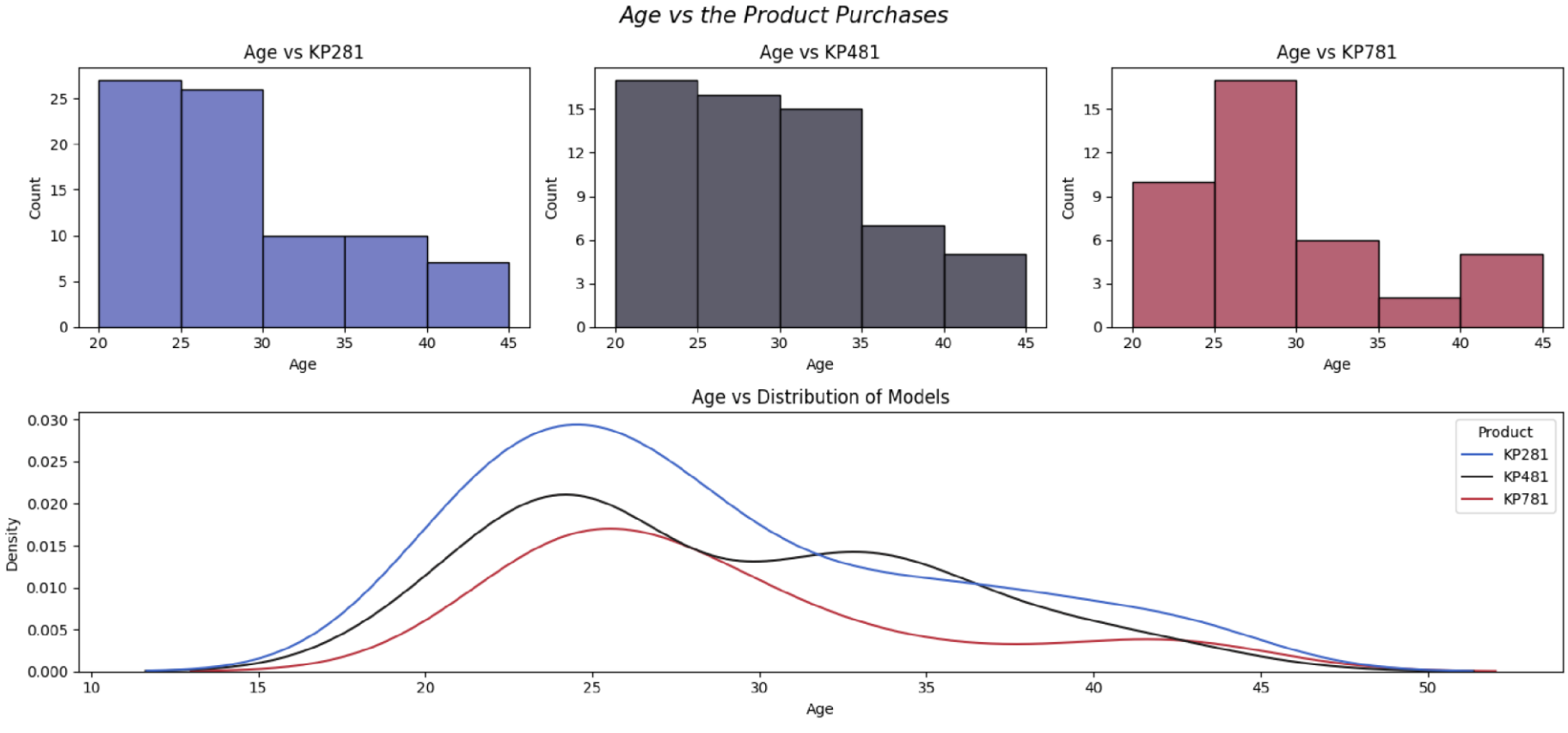
We will now be using the clipped_df as our nal dataset for our analysis.

Basic visualizations : Relationships between variables

Relationships between Categorical variables and the output variable *Age vs the*

Product Purchases

In [26]:
```
plt.figure(figsize = (15,7))
plt.suptitle('Age vs the Product Purchases', style = 'italic', size = 15)

ax1 = plt.subplot(231)
sns.histplot(clipped_df[clipped_df['Product'] == 'KP281'], x = 'Age', bins = [20,25,30,35,40,45], color =
'#4954b0')
plt.title('Age vs KP281')

ax2 = plt.subplot(232)
sns.histplot(clipped_df[clipped_df['Product'] == 'KP481'], x = 'Age', bins = [20,25,30,35,40,45], color =
'#282739')
ax2.set_yticks(np.arange(0,17,3))
plt.title('Age vs KP481')

ax3 = plt.subplot(233)
sns.histplot(clipped_df[clipped_df['Product'] == 'KP781'], x = 'Age', bins = [20,25,30,35,40,45], color =
'#9c2f45')
ax3.set_yticks(np.arange(0,17,3))
plt.title('Age vs KP781')

plt.subplot(212)
sns.kdeplot(clipped_df, x = 'Age', hue = 'Product', palette = 'icefire')
plt.title('Age vs Distribution of Models')

plt.tight_layout()
```



Observations :

- People aged around 20 to 30 years have the most number of purchases.
- The general trend of the customer's ages are that they're peaked between 20 to 30 and then slowly declining, indicating lesser customers of old age.
- *Interesting insight here is that there's a sudden rise in the number of customers in their early 30's when it comes to the KP481 model, i.e. it is the go to model for the people in their early 30's.*

In [27]:

Gender vs the Product purchase

```
plt.figure(figsize = (15,7))
sns.countplot(clipped_df, x = 'Gender', hue = 'Product', palette = 'viridis')
plt.ylabel('Number of Purchases')
plt.xlabel('Gender of the Customer')
plt.title('Relationship between the Gender of the customers and their Product choices')
plt.show()
```

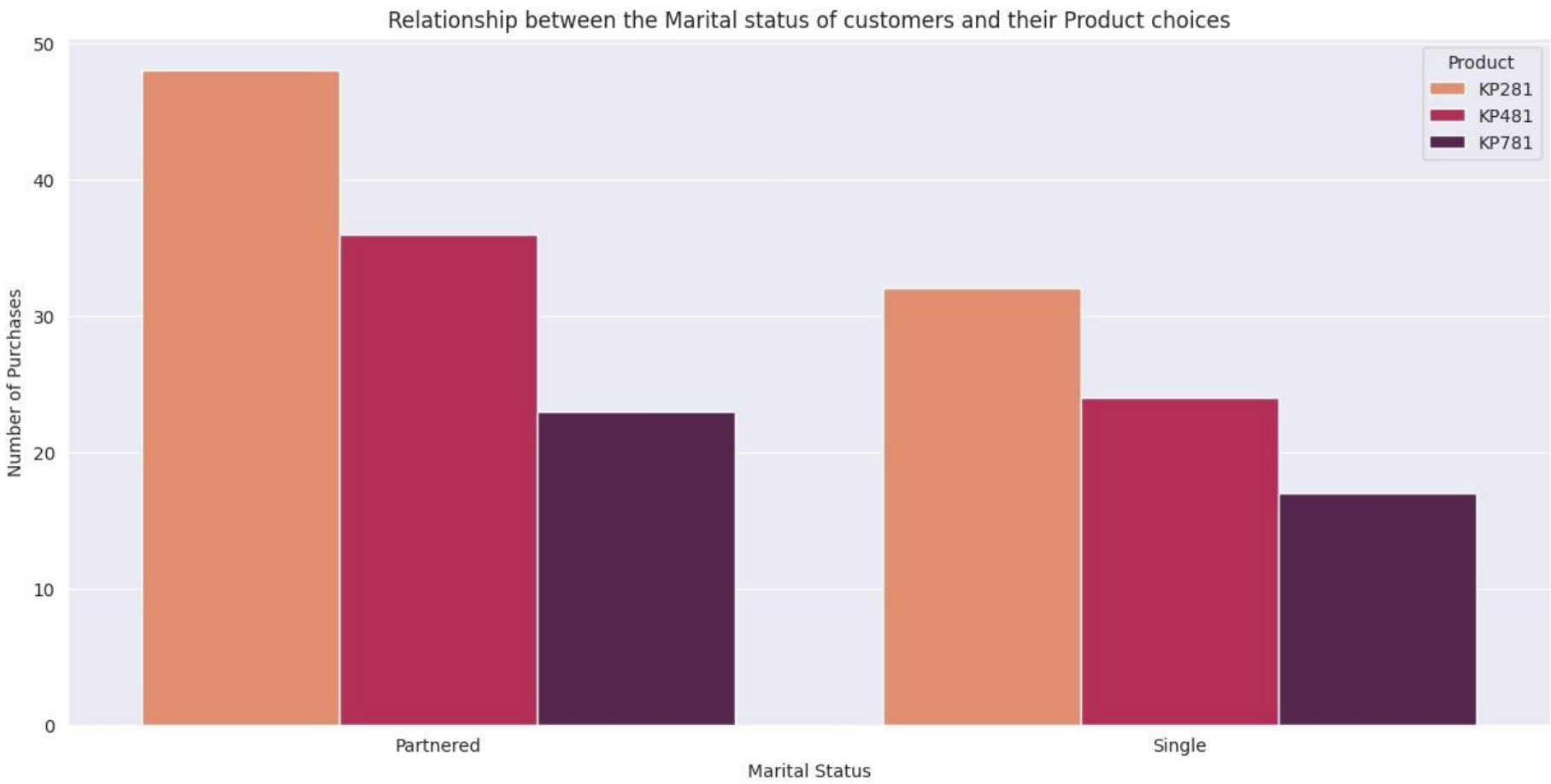
Observations :

- From the above plot, it is clear that the KP281 and the KP481 models are more or less equally preferred by customers of both the Genders.
- *Although when it comes to the KP781 model, it is seen that despite being the expensive one of the two, the model is preferred over the KP481 by customers of the 'Male' Gender, which can also be interpreted as the Males also tend to prefer more advanced features in their treadmills.*

In [28]:

Marital status vs the Product purchase

```
sns.set_style('darkgrid')

plt.figure(figsize = (15,7))
sns.countplot(clipped_df, x = 'MaritalStatus', hue = 'Product', palette = 'rocket_r')
plt.ylabel('Number of Purchases')
plt.xlabel('Marital Status')
plt.title('Relationship between the Marital status of customers and their Product choices')
plt.show()
```



Observations :

- From the above plot, we can infer that there is no speci c relationship between the marital status of the customer and the choice of product.
- We can see that the KP281, being the aordable model, is the User's choice of trademill followed by the KP481 and the KP781 respectively, all of this, regardless of the Marital status of the customers.
- *However, the interesting thing here is that the people who have Partners or Spouses with them, tend to purchase the products more than the single people.*

In [29]:

Customer's Income vs the Product Purchase

```
plt.figure(figsize = (15,7))
plt.suptitle('Income vs the Product Purchases', style = 'italic', size = 15)

ax1 = plt.subplot(231)
sns.histplot(clipped_df[clipped_df['Product'] == 'KP281'], x = 'Income', bins = np.arange(30000, 71000, 5000),
color = '#ffa300')
plt.xticks(rotation  = 75)
plt.title('Income vs KP281')

ax2 = plt.subplot(232)
sns.histplot(clipped_df[clipped_df['Product'] == 'KP481'], x = 'Income', bins = np.arange(30000, 71000, 5000),
color = '#e60049')
plt.xticks(rotation  = 75)
ax2.set_yticks(np.arange(0,17,3))
plt.title('Income vs KP481')

ax3 = plt.subplot(233)
sns.histplot(clipped_df[clipped_df['Product'] == 'KP781'], x = 'Income', bins = np.arange(45000, 96000,
5000),color = '#9b19f5')
plt.xticks(rotation  = 75)
ax3.set_yticks(np.arange(0,17,3))
plt.title('Income vs KP781')

plt.subplot(212)
cpal = ["#ffa300", "#e60049", "#9b19f5"]
sns.kdeplot(clipped_df, x = 'Income', hue = 'Product', palette = cpal)
plt.title('Income vs Distribution of Models')

plt.tight_layout()
```

Observations :

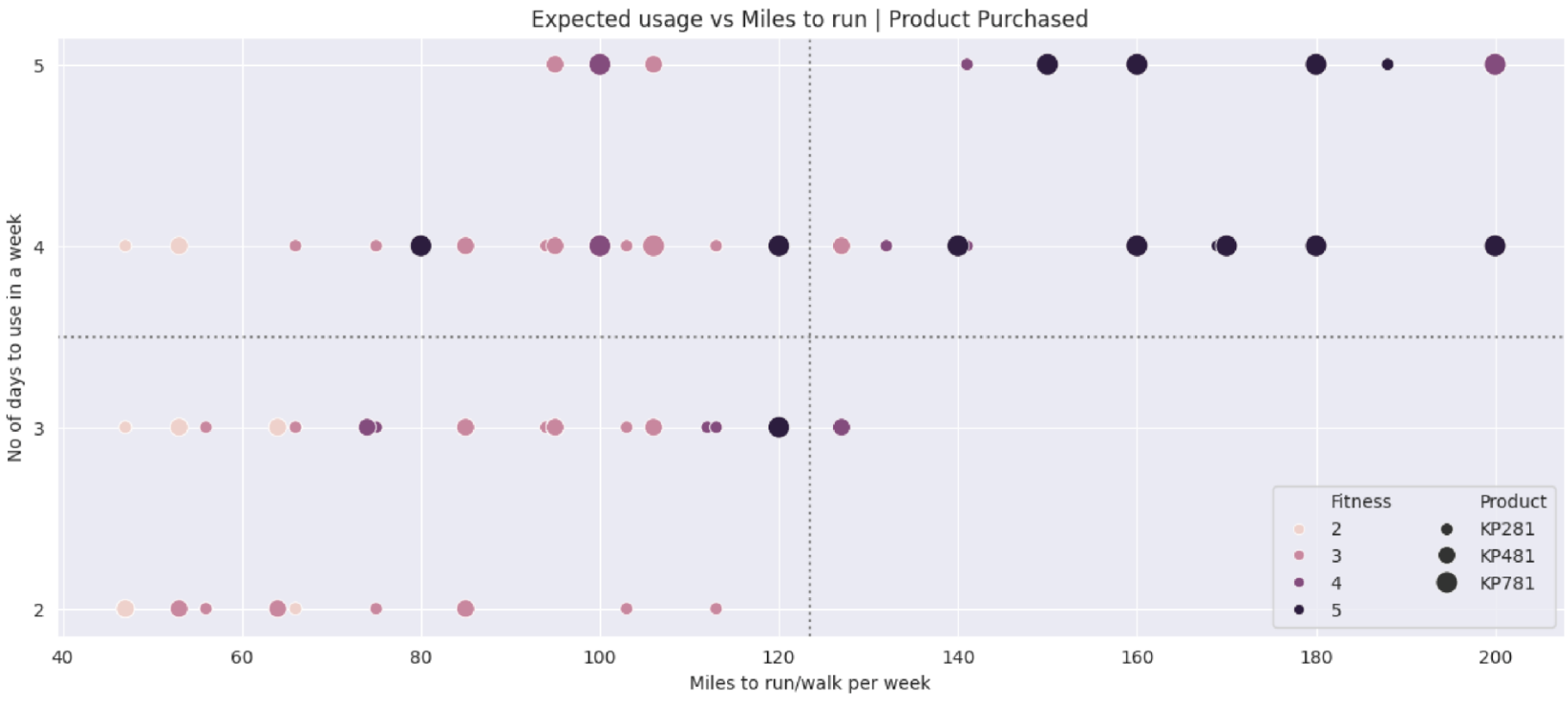
In [30]:

- KP281 is the most delivered product by the company. People having low to moderate income favor the KP281.
- KP481 is most likely purchased by people with moderate income like from 45000-55000.
- People earning over 80000 mostly buy the KP781 model of the treadmills.

Relationships between the Continous variables and the output variable

Expected Usage vs Miles to run of the customer

In [31]:
```
plt.figure(figsize = (15,6))
sns.scatterplot(data = clipped_df, x = 'Miles', y = 'Usage', hue = 'Fitness', size = 'Product', sizes = (150, 50))
plt.yticks(np.arange(2,6,1))
plt.ylabel('No of days to use in a week')
plt.xlabel('Miles to run/walk per week')
plt.legend(ncol = 2, loc = 'lower right')

# Calculate the center positions
x_center = (clipped_df['Miles'].max() + clipped_df['Miles'].min()) / 2
y_center = (clipped_df['Usage'].max() + clipped_df['Usage'].min()) / 2
# Add vertical and horizontal lines to create quadrants
plt.axvline(x=x_center, color='grey', linestyle=':')
plt.axhline(y=y_center, color='grey', linestyle=':')

plt.title('Expected usage vs Miles to run | Product Purchased')
plt.show()
```



Observations:

- Moderately t customers usually think of running less than 120. People running for more than 120 are usually on the tter end of the scale.
- Generally, people who plan to use the treadmill for more than 3 days a week and plan to run at least 120 miles a week, seem to rate themselves on the tter end of the scale.
- People who rate themselves the Fittest tend to run nothing less than 120 miles.
- Another interesting thing to note from the upper half of the diagram is that people using the treadmills for more than 3 days, most likely buy the KP781.

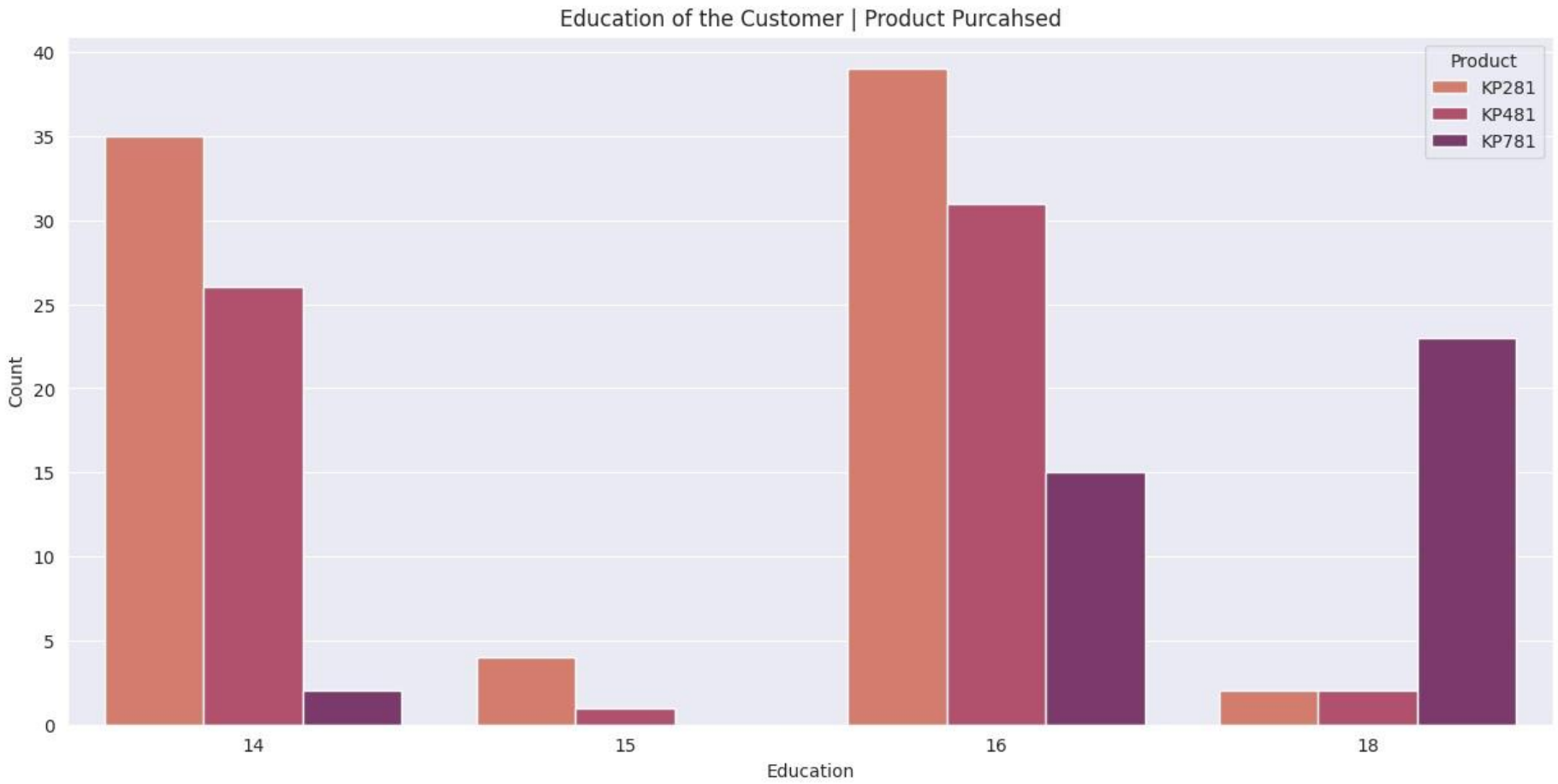
In [32]:
```
clipped_df[clipped_df['Usage']>3]['Product'].value_counts()
```

Out[32]:
```
Product
KP781    39
KP281    24
KP481    15
Name: count, dtype: int64
```

Thus we can con rm the last observation to be true from the above output.

Education vs the Product Purchased

In [33]:
```
# Relationship between Education and Product purchased

plt.figure(figsize = (15,7))
sns.countplot(data = clipped_df, x = 'Education', hue = 'Product', palette = 'flare')
plt.ylabel('Count')
plt.title('Education of the Customer | Product Purcahsed')
plt.show()
```

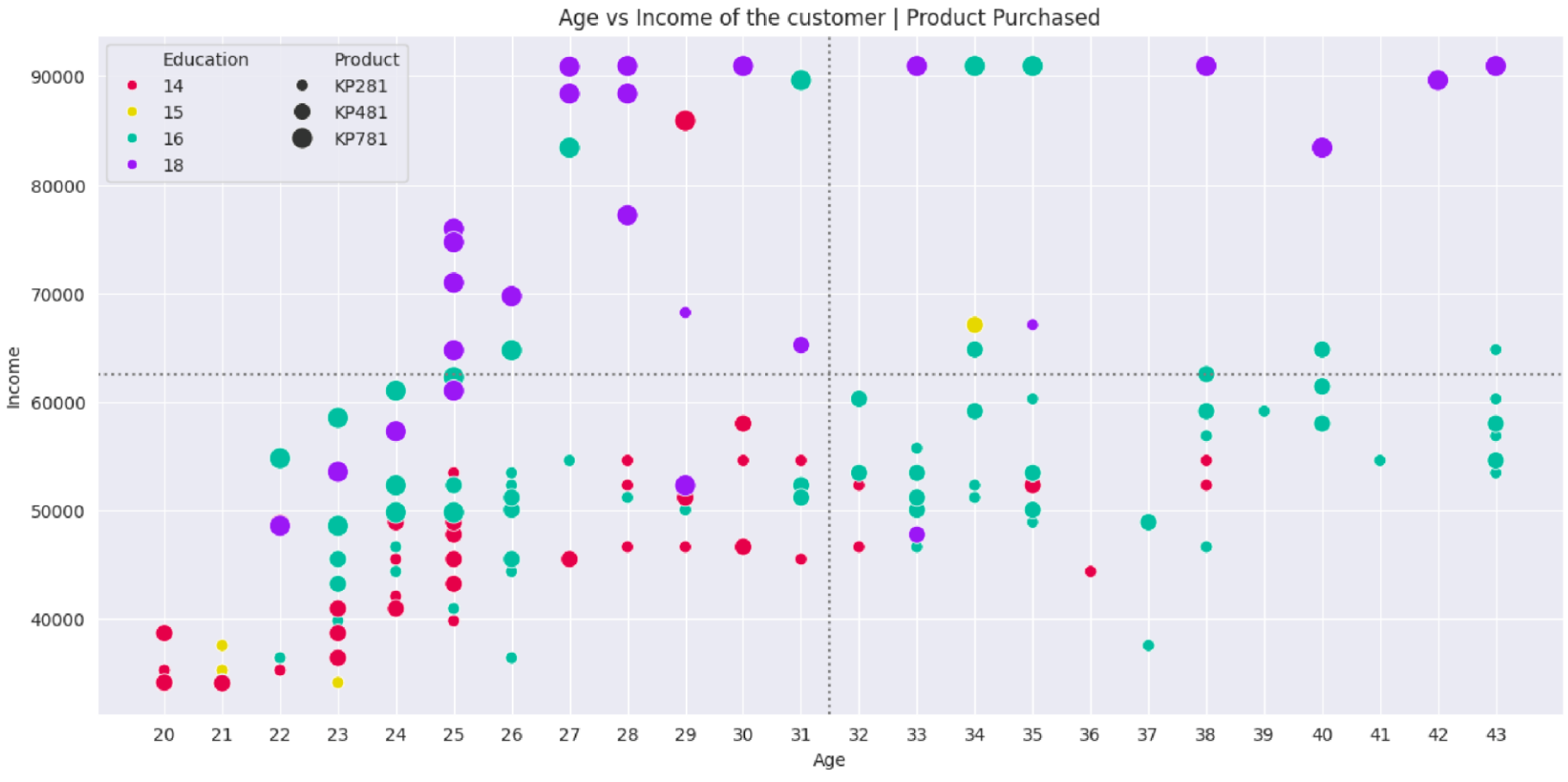


Observations :

- More or less educated, the KP281 is the Customer's choice.
- Very highly quali ed people are likely to go for the KP781, which can be correlated with their want to use the Advanced features o erred by the KP781.

Age vs Income of the customer | Product Purchased

In [34]:
```
plt.figure(figsize = (15,7))
sns.scatterplot(
    data = clipped_df, x = 'Age', y = 'Income', hue = 'Education', size = 'Product',
    palette = ['#e60049', '#e6d800', '#00bfa0', '#9b19f5'], sizes = (150, 50)
    )

# Calculate the center positions
x_center = (clipped_df['Age'].max() + clipped_df['Age'].min()) / 2
y_center = (clipped_df['Income'].max() + clipped_df['Income'].min()) / 2
# Add vertical and horizontal lines to create quadrants
plt.axvline(x=x_center, color='grey', linestyle=':')
plt.axhline(y=y_center, color='grey', linestyle=':')

plt.xticks(clipped_df['Age'].unique())
plt.legend(ncol = 2)
plt.title('Age vs Income of the customer | Product Purchased')
plt.show()
```

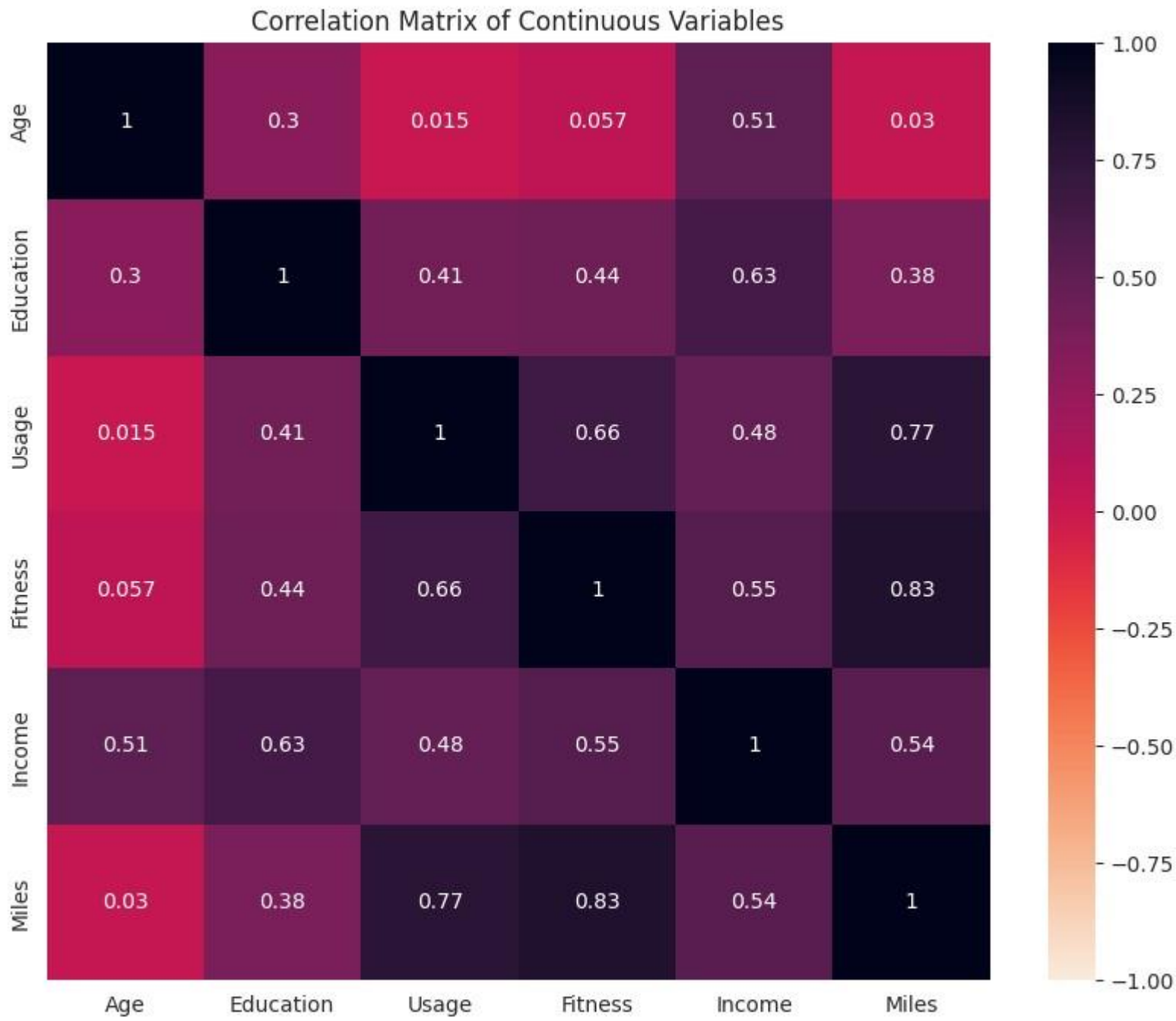


Observations:

- People with higher number of educational years are likely to make a higher salary.
- Since most tiny circles are either red or green, we can infer that people with lesser education tend to avoid the KP781 model.
- The KP781 model is popular only among the 20-30 age group, very few beyond this group prefer the model, indicating most old people are no fans of advanced features in their treadmills.

Correlation among di erent factors

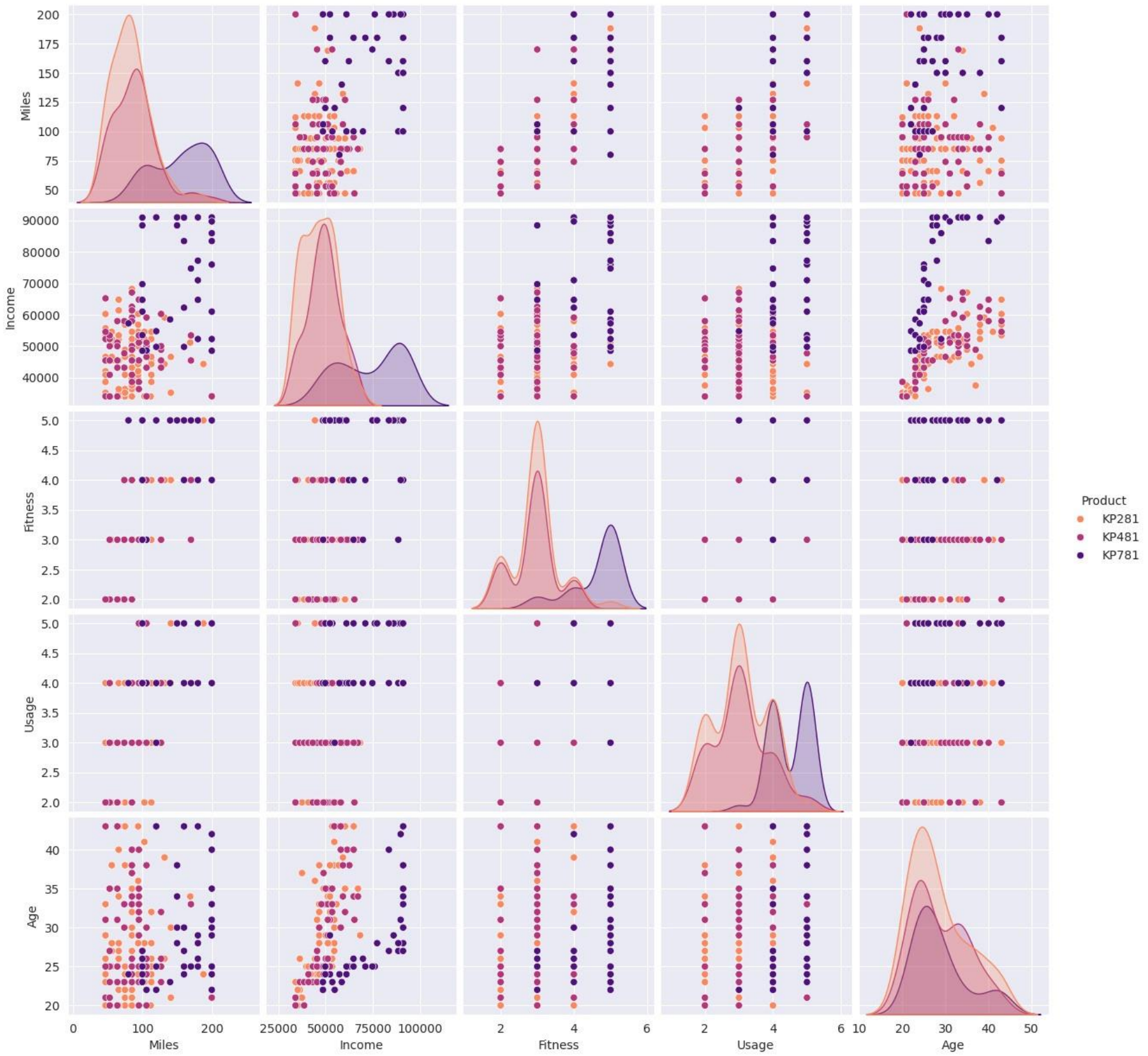
In [36]:
```
corr_matrix = clipped_df[num_cols].corr()

# Plotting the correlation matrix
plt.figure(figsize = (10, 8))
sns.heatmap(corr_matrix, annot = True, cmap = 'rocket_r', vmin = -1, vmax = 1)
plt.title('Correlation Matrix of Continuous Variables')
plt.show()
```



Observations:

- All the columns are positively correlated with each other.
- Age is highly correlated with Income and Education, which seems to be okay since a person earns and learns more with age.
- Usage is highly correlated with Fitness & Miles. Exercise and tness do walk hand in hand.
- Fitness & Income are also highly correlated, which tells us that Fitter people are likely to earn more.

In [37]:
```
# Extracting few columns from original dataframe and using pairplot on this newly created data frame

df_pairplot = clipped_df[['Miles','Income','Fitness','Usage','Age','Product']]
sns.pairplot(data = df_pairplot, hue = 'Product', palette = 'magma_r')
plt.show()
```



There are many insights we can get from this plot, like

- Customers who are more t and have more number of miles run, they purchased product with high rating i.e. KP781, followed by KP481 and KP281
- Customers with high level of tness and having higher incomes, purchased KP781 product followed by KP481 and KP281
- Customers who purchased product KP781, have higher incomes and they run more number of miles comparing to other type of customers.

Probability Representation

Percentage of purchases

In [39]:
```
(purchases_gender := pd.crosstab(clipped_df['Product'], clipped_df['Gender'], margins = True, margins_name =
'Marginal Probability', normalize = True))
```

Out[39]:

Gender	Female	Male	Marginal Probability
Product			

KP281	0.22	0.22	0.44
KP481	0.16	0.17	0.33
KP781	0.04	0.18	0.22
Marginal Probability	0.42	0.58	1.00

Observations :

- Thus, we have the marginal probabilities for the sale of each products and we see that it is highest for the KP281 model.

In [40]:
```
# Marital Status

(purchases_marital_status := pd.crosstab(clipped_df['Product'], clipped_df['MaritalStatus'], margins = True,
margins_name = 'Marginal Probability', normalize = True))
```

Out[40]:

MaritalStatus	Partnered	Single	Marginal Probability
Product			
KP281	0.27	0.18	0.44
KP481	0.20	0.13	0.33
KP781	0.13	0.09	0.22
Marginal Probability	0.59	0.41	1.00

In [41]:
```
# Age

(purchases_age := pd.crosstab(clipped_df['Product'], clipped_df['Age'], margins = True, margins_name = 'Marginal
Probability', normalize = True))
```

Out[41]:

Age	20	21	22	23	24	25	26	27	28	29	...	35	36	37	38	39	40	41	42	43	Marginal Probability
Product																					
KP281	0.03	0.02	0.02	0.04	0.03	0.04	0.04	0.02	0.03	0.02	...	0.02	0.01	0.01	0.02	0.01	0.01	0.01	0.00	0.03	0.44
KP481	0.02	0.02	0.00	0.04	0.02	0.06	0.02	0.01	0.00	0.01	...	0.02	0.00	0.01	0.01	0.00	0.02	0.00	0.00	0.01	0.33
KP781	0.00	0.00	0.02	0.02	0.02	0.04	0.01	0.02	0.02	0.01	...	0.01	0.00	0.00	0.01	0.00	0.01	0.00	0.01	0.02	0.22
Marginal Probability	0.06	0.04	0.04	0.10	0.07	0.14	0.07	0.04	0.05	0.03	...	0.04	0.01	0.01	0.04	0.01	0.03	0.01	0.01	0.06	1.00

4 rows × 25 columns

In [42]:
```
# Usage

(purchases_usage := pd.crosstab(clipped_df['Product'], clipped_df['Usage'], margins = True, margins_name =
'Marginal Probability', normalize = True))
```

Out[42]:

Usage	2	3	4	5	Marginal Probability
Product					
KP481	0.08	0.17	0.07	0.02	0.33
KP781	0.00	0.01	0.10	0.12	0.22
Marginal Probability	0.18	0.38	0.29	0.14	1.00

Out[43]:

Fitness	2	3	4	5	Marginal Probability
Product					
KP281	0.08	0.30	0.05	0.01	0.44
KP481	0.07	0.22	0.04	0.00	0.33
KP781	0.00	0.02	0.04	0.16	0.22
Marginal Probability	0.16	0.54	0.13	0.17	1.00

In [43]:
```
# Fitness

(purchases_fitness := pd.crosstab(clipped_df['Product'], clipped_df['Fitness'], margins = True, margins_name =
'Marginal Probability', normalize = True))
```

In [44]:
```
# Education

(purchases_education := pd.crosstab(clipped_df['Product'], clipped_df['Education'], margins = True, margins_name =
'Marginal Probability', normalize = True))
```

Out[44]:

Education	14	15	16	18	Marginal Probability
Product					
KP281	0.19	0.02	0.22	0.01	0.44
KP481	0.14	0.01	0.17	0.01	0.33
KP781	0.01	0.00	0.08	0.13	0.22
Marginal Probability	0.35	0.03	0.47	0.15	1.00

From the above crosstab results, we can de nitely gure out the probabilities of a customer purchasing a certain product based on their characteristics (or column).

For e.g., It can be identi ed,

- *from the 'purchases_gender' dataframe, that the probability of a 'Male' person purchasing the KP281 is 0.22. from the 'purchases_*
- *tness' dataframe, that the probability of a 'Fitness 5' rated person purcahsing the KP781 model is 0.16. from the*
- *'purchases_maritalstatus', that the probability of a 'Partnered' person purchasing the KP481 model is 0.20.*

Conditional Probabilities

Conditional probability is known as the possibility of an event or outcome happening, based on the existence of a previous event or outcome. It is calculated by multiplying the probability of the preceding event by the renewed probability of the succeeding, or conditional, event.

Formula : `P(A|B) = N(A∩B)/N(B)`

What is the probability of selling a KP781 given that the customer is a Male?

In [46]:
```
(conProb_male781 := round(purchases_gender.loc['KP781', 'Male']/purchases_gender.loc['KP781', 'Marginal Probability'], 3))
```

Out[46]: 0.825

What is the probability of selling a KP781 given that the customer is a Female?

In [47]:
```
(conProb_male781 := round(purchases_gender.loc['KP781', 'Female']/purchases_gender.loc['KP781', 'Marginal Probability'], 3))
```

Out[47]: 0.175

Similarly, we can create an entire dataframe comprising of the conditional probabilities that the customer buys a particular model given that the gender of the customer is known.

In [48]:
```
# Conditional Probabilities : by Gender
(purchases_gender_cp := purchases_gender[['Male', 'Female']].loc['KP281':'KP781'].div(purchases_gender['Marginal Probability'].loc['KP281':'KP781'], axis = 0))
```

Out[48]:

Gender Product	Male	Female
KP281	0.50	0.50
KP481	0.52	0.48
KP781	0.82	0.18

In [49]:
```
# Conditional Probabilities : by Marital Status
(purchases_marital_status_cp := purchases_marital_status[['Partnered', 'Single']].loc['KP281':'KP781'].div(purchases_marital_status['Marginal Probability'].loc['KP281':'KP781'], axis = 0))
```

Out[49]:

MaritalStatus Product	Partnered	Single
KP281	0.60	0.40
KP481	0.60	0.40
KP781	0.57	0.42

Customer Pro ling

Treadmills : KP281, KP481 & KP781

KP281: *Entry-Level Treadmill - $1,500*

The KP281 is predominantly purchased by individuals aged between 20 and 30 years, who have completed approximately 14 to 16 years of education. This product appeals equally to both genders, with a signi cant proportion being partnered individuals. These customers typically have an annual income ranging from 35,000*to*55,000. They are low to moderately t and generally plan to run up to 120 miles per week, making the KP281 an excellent choice *for those starting their tness journey.* Ideal Customer:

This model is perfect for young adults, both men and women, who are in the early stages of their careers, moderately t, and looking for an a ordable treadmill to support their tness goals within a budget.

KP481: *Mid-Level Treadmill - $1,750*

The KP481 is favored by partnered individuals aged between 20 and 35 years, with a notable preference among those in the 30-35 age group. These customers usually earn between 45,000*and*55,000 annually. The product appeals equally to both genders and is popular among those who have completed 14 to 16 years of education. Moderately t runners who plan to run around 120 miles per week nd the KP481 to be an ideal choice, providing the *perfect balance of features and performance.* Ideal Customer:

This model suits partnered individuals in their late twenties to mid-thirties, moderately t, and looking for a reliable treadmill to support their regular running routines. It is especially attractive to those with a slightly higher budget seeking a balance between a ordability and advanced features.

KP781: *Advanced Treadmill - $2,500*

The KP781 is the preferred choice for highly educated, partnered males aged between 25 and 30 years, with an annual income exceeding $85,000. These customers typically rank high on the tness scale and plan to run approximately upto 200 miles per week. The advanced features of the KP781 cater to the needs of serious runners and tness enthusiasts with athletic tness, making it the top choice *for those seeking high performance and advanced functionality.* Ideal Customer:

This model is ideal for high-income, highly educated young men who are serious about their tness and running routines. It is perfect for those seeking an advanced treadmill with superior features to support an intensive training schedule.

Recommendations

- Focus marketing e orts on individuals aged 20 to 30, as they represent the largest customer base.
- Highlight the KP481 model speci cally to customers in their early 30s. Consider targeted promotions and advertisements that promote KP481 for this age group.
- Since males prefer the more advanced KP781 model, create marketing campaigns that emphasize the advanced features and higher performance of the KP781.
- Since customers with partners or spouses tend to purchase more, develop marketing campaigns that target couples and families. Oer discounts or incentives for joint purchases or family bundles.
- Emphasize the aordability and value of the KP281, especially for customers with low to moderate income. Highlight its features that provide good value for money.
- Continue to innovate and add advanced features to the KP781 to maintain its appeal to high-income, highly educated customers, and those who prioritize tness. Collect feedback from this segment to understand their needs and preferences.
- Based on customer feedback, consider incremental enhancements to the KP481 and KP281 models that could make them more attractive without signi cantly increasing costs.
- Collaborate with tness in uencers and trainers(possibly female to attract the Female customers towards tness) who can endorse the treadmills and provide authentic reviews and usage tips.