business-case-aerofit

October 5, 2024

1 Business Case: Aerofit - Descriptive Statistics & Probability

2 1. Introduction

2.1 What is Aerofit?

`Aerofit`, a dynamic player in the fitness industry, traces its origins to `M/s. Sachdev Sports Co`, established in `1928` by `Ram Ratan Sachdev`. From its modest beginnings in `Hyderabad`, India, the company evolved into a leading sports equipment supplier across Andhra Pradesh and Telangana. Recognizing the growing need for fitness solutions, M/s. Sachdev Overseas emerged to import quality fitness equipment under the "Aerofit" brand, ensuring affordability and post-sales excellence.

Driven by a dedication to innovation, `Nityasach Fitness Pvt Ltd` was founded, spearheaded by director `Nityesh Sachdev`. With the brand "Aerofit" at its core, the company aimed to bridge the gap between international fitness technology and the Indian market. By importing advanced fitness equipment at accessible price points, Aerofit sought to redefine the industry landscape, prioritizing health and vitality while staying true to its legacy of passion and customer focus.

Aerofit provides a product range including machines such as treadmills, exercise bikes, gym equipment, and fitness accessories to cater to the needs of all categories of people.

Objective

Create comprehensive customer profiles for each AeroFit treadmill product through descriptive analytics. Develop two-way contingency tables and analyze conditional and marginal probabilities to discern customer characteristics, facilitating improved product recommendations and informed business decisions.

About Data

The company collected the data on individuals who purchased a treadmill from the AeroFit stores during `three months`.The data is available in a single csv file

Product Portfolio

- The `KP281` is an `entry-level` treadmill that sells for `USD 1,500`.
- The `KP481` is for `mid-level` runners that sell for `USD 1,750`.
- The `KP781` treadmill is having `advanced features` that sell for `USD 2,500`.

Features of the dataset:

Feature	Description
Product	Product Purchased: KP281, KP481, or KP781
Age	Age of buyer in years
Gender	Gender of buyer (Male/Female)
Education	Education of buyer in years
MaritalStatus	MaritalStatus of buyer (Single or partnered)
Usage	The average number of times the buyer plans to use the treadmill each week
Income	Annual income of the buyer (in $)
Fitness	Self-rated fitness on a 1-to-5 scale, where 1 is the poor shape and 5 is the excellent shape
Miles	The average number of miles the buyer expects to walk/run each week

3 2. Exploratory Data Analysis

[3]:
```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
import copy
```

[9]:
```
# loading the dataset
df = pd.read_csv('/content/sample_data/aerofit_treadmill.csv')
```

[10]:
```
df.head()
```

[10]:

	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

[11]:
```
df.tail()
```

[11]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	\
175	KP781	40	Male	21	Single	6	5	83416	
176	KP781	42	Male	18	Single	5	4	89641	
177	KP781	45	Male	16	Single	5	5	90886	
178	KP781	47	Male	18	Partnered	4	5	104581	

```
179   KP781   48   Male          18      Partnered       4         5    95508

     Miles
175    200
176    200
177    160
178    120
179    180
```

[12]:
```
df.shape
```

[12]:
```
(180, 9)
```

[13]:
```
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
```

3.0.1 Insights:

- From the above exploratory analysis, the dataset consists of 180 records with no missing values across its 9 features.
- It contains 6 numerical features (Age, Education, Usage, Fitness, Income, Miles) and 3 categorical features (Product, Gender, MaritalStatus).
- The data types of each column are appropriate based on their respective content

3.1 Changing the Datatype of Columns

- To enhance interpretability, convert the data types of the Usage and Fitness columns from int64 to object, as these represent categorical values rather than continuous numerical data.

[14]:
```
df['Usage'] = df['Usage'].astype('str')
df['Fitness'] = df['Fitness'].astype('str')
```

```
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    object
 6   Fitness        180 non-null    object
 7   Income         180 non-null    int64 
 8   Miles          180 non-null    int64 
dtypes: int64(4), object(5)
memory usage: 12.8+ KB
```

3.1.1 Statistical Summary

[16]:
```
# statisctical summary of object type columns
df.describe()
df.describe(include = 'object')
```

[16]:

	Product	Gender	MaritalStatus	Usage	Fitness
count	180	180	180	180	180
unique	3	2	2	6	5
top	KP281	Male	Partnered	3	3
freq	80	104	107	69	97

3.1.2 Insights

1. **Product Sales Performance:** Over the past three months, the KP281 treadmill has emerged as the best-selling product, representing approximately 44% of total sales, outperforming the other two models.

2.Gender Distribution: The data from the last three months indicates that 58% of the buyers were male, while 42% were female, highlighting a slightly higher participation from male customers.

3.Marital Status: Analysis of the last three months shows that 60% of the buyers were married, compared to 40% who were single, suggesting a stronger inclination towards treadmill purchases among married individuals.

[ ]:
```
# statisctical summary of numerical data type columns

df.describe()
```

[ ]:

	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

3.1.3 Insights

1. Age: The average age of buyers is approximately 28.8 years, with ages spanning from 18 to 50 years.

2. Education: On average, buyers have completed 15.6 years of education, with educational backgrounds ranging from 12 to 21 years.

3. Usage: Buyers plan to use the treadmill 3.46 times per week on average, with usage frequency ranging from 2 to 7 times per week.∴

4. Fitness: Self-assessed fitness levels, rated on a scale from 1 (poor shape) to 5 (excellent shape), show an average score of 3.31, reflecting a generally moderate level of fitness among buyers.

5.Income: The annual income of buyers ranges from $29,562 to $104,581, with an average income of $53,078, indicating a diverse income range within the customer base.

6. Miles: Buyers expect to walk or run an average of 82.8 miles per week, with a minimum of 21 miles and a maximum of 156 miles, showcasing a wide variation in fitness goals.

3.2 Duplicate Detection

[17]:
```
df.duplicated().value_counts()
```

[17]:
```
False    180
Name: count, dtype: int64
```

3.2.1 Insights

- All 180 rows are unique, there are no duplicate entries in the dataset

3.3 Sanity Check for columns

[ ]:
```
# checking the unique values for columns
for i in df.columns:
    print('Unique Values in',i,'column are :-')
    print(df[i].unique())
    print('-'*70)A
```

```
Unique Values in Product column are :-
['KP281' 'KP481' 'KP781']
----------------------------------------------------------------------
Unique Values in Age column are :-
[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]
----------------------------------------------------------------------
Unique Values in Gender column are :-
['Male' 'Female']
----------------------------------------------------------------------
Unique Values in Education column are :-
[14 15 12 13 16 18 20 21]
----------------------------------------------------------------------
Unique Values in MaritalStatus column are :-
['Single' 'Partnered']
----------------------------------------------------------------------
Unique Values in Usage column are :-
[3 2 4 5 6 7]
----------------------------------------------------------------------
Unique Values in Fitness column are :-
[4 3 2 1 5]
----------------------------------------------------------------------
Unique Values in Income column are :-
[ 29562  31836  30699  32973  35247  37521  36384  38658  40932  34110
  39795  42069  44343  45480  46617  48891  53439  43206  52302  51165
  50028  54576  68220  55713  60261  67083  56850  59124  61398  57987
  64809  47754  65220  62535  48658  54781  48556  58516  53536  61006
  57271  52291  49801  62251  64741  70966  75946  74701  69721  83416
  88396  90886  92131  77191  52290  85906 103336  99601  89641  95866
 104581  95508]
----------------------------------------------------------------------
Unique Values in Miles column are :-
[112  75  66  85  47 141 103  94 113  38 188  56 132 169  64  53 106  95
 212  42 127  74 170  21 120 200 140 100  80 160 180 240 150 300 280 260
 360]
----------------------------------------------------------------------
```

3.3.1 Insights

- The dataset does not contain any abnormal values.

Adding new columns for better analysis

- Creating New Column and Categorizing values in `Age,Education,Income and Miles` to different classes for better visualization

Age Column

- Categorizing the values in age column in 4 different buckets:

1. Young Adult: from 18 - 25
2. Adults: from 26 - 35
3. Middle Aged Adults: 36-45
4. Elder :46 and above

Education Column

- Categorizing the values in education column in 3 different buckets:

1. Primary Education: upto 12
2. Secondary Education: 13 to 15
3. Higher Education: 16 and above

Income Column

- Categorizing the values in Income column in 4 different buckets:

1. Low Income - Upto 40,000
2. Moderate Income - 40,000 to 60,000
3. High Income - 60,000 to 80,000
4. Very High Income - Above 80,000

Miles column

- Categorizing the values in miles column in 4 different buckets:

1. Light Activity - Upto 50 miles
2. Moderate Activity - 51 to 100 miles
3. Active Lifestyle - 101 to 200 miles
4. Fitness Enthusiast - Above 200 miles

[19]:
```
# Binning the 'Age' column into meaningful age groups
age_bins = [17, 25, 35, 45, float('inf')]
age_labels = ['Young Adults', 'Adults', 'Middle-Aged Adults', 'Elderly']

# Creating a new column 'age_group' based on age binning
df['age_group'] = pd.cut(df['Age'], bins=age_bins, labels=age_labels)

# Binning the 'Education' column into education levels
education_bins = [0, 12, 15, float('inf')]
education_labels = ['Primary Education', 'Secondary Education', 'Higher␣
 ↪Education']
```

```
# Creating a new column 'edu_group' based on education binning
df['edu_group'] = pd.cut(df['Education'], bins=education_bins,␣
 ↪labels=education_labels)

# Binning the 'Income' column into income categories
income_bins = [0, 40000, 60000, 80000, float('inf')]
income_labels = ['Low Income', 'Moderate Income', 'High Income', 'Very High␣
 ↪Income']

# Creating a new column 'income_group' based on income binning
df['income_group'] = pd.cut(df['Income'], bins=income_bins,␣
 ↪labels=income_labels)

# Binning the 'Miles' column into activity levels based on miles run
miles_bins = [0, 50, 100, 200, float('inf')]
miles_labels = ['Light Activity', 'Moderate Activity', 'Active Lifestyle',␣
 ↪'Fitness Enthusiast']

# Creating a new column 'miles_group' based on miles binning
df['miles_group'] = pd.cut(df['Miles'], bins=miles_bins, labels=miles_labels)

# Displaying the first few rows to verify the new bins
df[['Age', 'age_group', 'Education', 'edu_group', 'Income', 'income_group',␣
 ↪'Miles', 'miles_group']].head()
```

[19]:

	Age	age_group	Education	edu_group	Income	income_group	Miles	miles_group
0	18	Young Adults	14	Secondary Education	29562	Low Income	112	Active Lifestyle
1	19	Young Adults	15	Secondary Education	31836	Low Income	75	Moderate Activity
2	19	Young Adults	14	Secondary Education	30699	Low Income	66	Moderate Activity
3	19	Young Adults	12	Primary Education	32973	Low Income	85	Moderate Activity
4	20	Young Adults	13	Secondary Education	35247	Low Income	47	Light Activity

[20]:
```
df.head()
```

[20]:

	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

```
      age_group            edu_group income_group        miles_group
0  Young Adults  Secondary Education   Low Income   Active Lifestyle
1  Young Adults  Secondary Education   Low Income  Moderate Activity
2  Young Adults  Secondary Education   Low Income  Moderate Activity
3  Young Adults    Primary Education   Low Income  Moderate Activity
4  Young Adults  Secondary Education   Low Income     Light Activity
```

4 3.Univariate Analysis

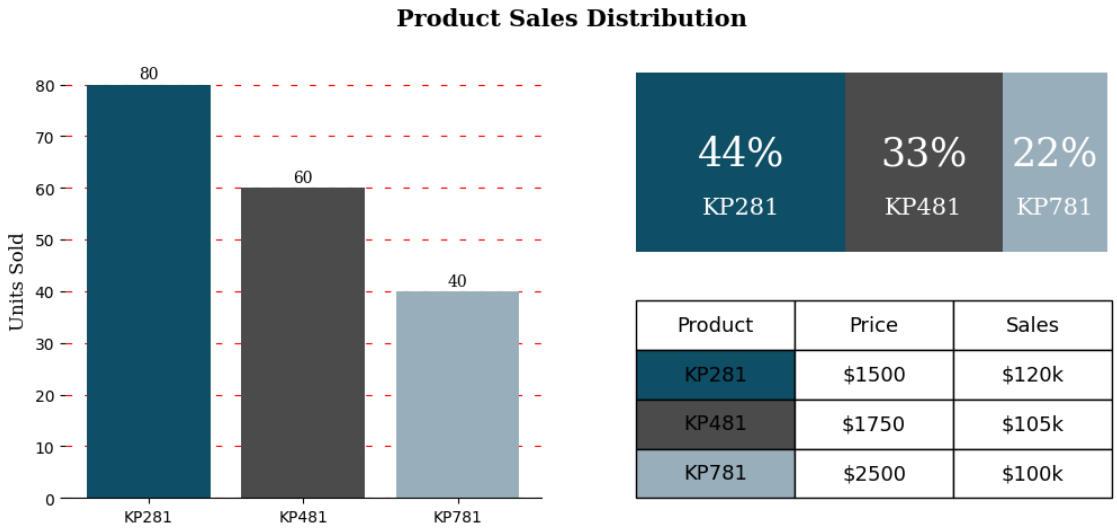
4.1 3.1 Categorical Variables

4.1.1 3.1.1 Product Sales Distribution

[23]:
```
#setting the plot style

fig = plt.figure(figsize = (12,5))
gs = fig.add_gridspec(2,2)

                                        #creating plot for product column

ax0 = fig.add_subplot(gs[:,0])

product_count = df['Product'].value_counts()

color_map = ["#0e4f66", "#4b4b4c",'#99AEBB']

ax0.bar(product_count.index,product_count.values,color = color_map,zorder = 2)

#adding the value_counts
for i in product_count.index:
    ax0.text(i,product_count[i]+2,product_count[i],{'font':'serif','size' :␣
 ↪10},ha = 'center',va = 'center')

#adding grid lines
ax0.grid(color = 'red',linestyle = '--', axis = 'y', zorder = 0, dashes =␣
 ↪(5,10))

#removing the axis lines
for s in ['top','left','right']:
    ax0.spines[s].set_visible(False)

#adding axis label
ax0.set_ylabel('Units Sold',fontfamily='serif',fontsize = 12)

                                        #creating a plot for product % sale
```

```
ax1 = fig.add_subplot(gs[0,1])

product_count['percent'] = ((product_count.values/df.shape[0])* 100).round()

ax1.barh(product_count.index[0],product_count.loc['percent'][0],color =␣
 ↪"#0e4f66")
ax1.barh(product_count.index[0],product_count.loc['percent'][1],left =␣
 ↪product_count.loc['percent'][0],color = '#4b4b4c')
ax1.barh(product_count.index[0],product_count.loc['percent'][2],
         left = product_count.loc['percent'][0] + product_count.
 ↪loc['percent'][1], color = '#99AEBB')
ax1.set(xlim=(0,100))


# adding info to the each bar
product_count['info_percent'] =[product_count['percent'][0]/
 ↪2,product_count['percent'][0] + product_count['percent'][1]/2,
                                product_count['percent'][0] +␣
 ↪product_count['percent'][1] + product_count['percent'][2]/2]
for i in range(3):
    ax1.text(product_count['info_percent'][i],0.
 ↪04,f"{product_count['percent'][i]:.0f}%",
             va = 'center', ha='center',fontsize=25, fontweight='light',␣
 ↪fontfamily='serif',color='white')

    ax1.text(product_count['info_percent'][i],-0.2,product_count.index[i],
             va = 'center', ha='center',fontsize=15, fontweight='light',␣
 ↪fontfamily='serif',color='white')

#removing the axis lines
ax1.axis('off')

                                        #creating a plot for product portfolio


ax2 = fig.add_subplot(gs[1,1])

product_portfolio =␣
 ↪[['KP281','$1500','$120k'],['KP481','$1750','$105k'],['KP781','$2500','$100k']]
color_2d =␣
 ↪[['#0e4f66','#FFFFFF','#FFFFFF'],['#4b4b4c','#FFFFFF','#FFFFFF'],['#99AEBB','#FFFFFF','#FFF
table = ax2.table(cellText = product_portfolio, cellColours=color_2d,␣
 ↪cellLoc='center',colLabels =['Product','Price','Sales'],
                  colLoc = 'center',bbox =[0, 0, 1, 1])

table.set_fontsize(13)
```

```
#removing axis
ax2.axis('off')

#adding title to the visual
fig.suptitle('Product Sales Distribution',fontproperties = {'family':'serif',␣
 ↪'size':15,'weight':'bold'})

plt.show()
```

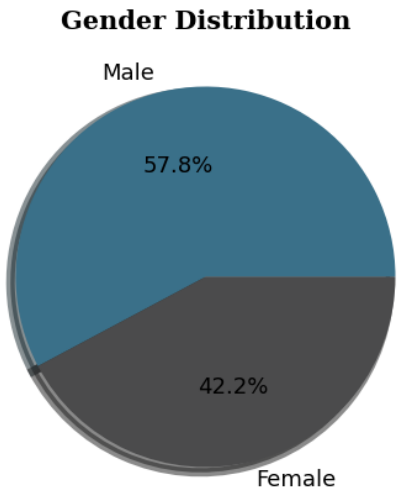


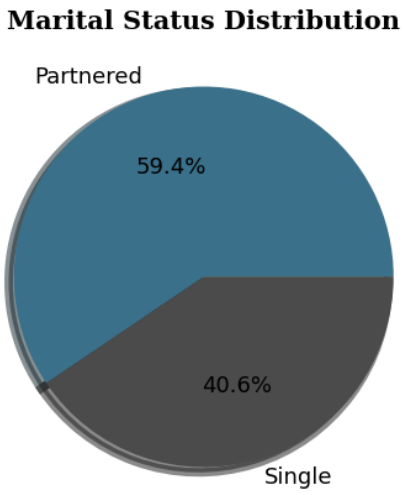
4.1.2 Insights

- KP281 Treadmill: The KP281 model, positioned as an entry-level treadmill, leads in terms of the number of units sold, making it the most popular choice among customers. This suggests that affordability and simplicity are likely key drivers for its success in the market.
- KP481 and KP781 Treadmills: While the KP481 (mid-range) and KP781 (premium) models have lower unit sales compared to KP281, they cater to more specialized or advanced customer segments. The sales of these models reflect a demand for higher-end features and a more premium workout experience.
- Revenue Distribution: Despite the higher sales volume of the KP281, all three treadmill models contribute nearly equally to the overall revenue. This can be attributed to the higher pricing of the KP481 and KP781 models, which balances the revenue contribution despite fewer units sold. This highlights the effectiveness of a diversified product strategy in maintaining balanced financial returns across different customer segments.

4.1.3 3.1.2 Gender and Marital Status Disribution

[24]:
```
#setting the plot style
fig = plt.figure(figsize = (12,5))
gs = fig.add_gridspec(1,2)

                                        # creating pie chart for gender␣
 ↪disribution
ax0 = fig.add_subplot(gs[0,0])

color_map = ["#3A7089", "#4b4b4c"]
ax0.pie(df['Gender'].value_counts().values,labels = df['Gender'].value_counts().
 ↪index,autopct = '%.1f%%',
        shadow = True,colors = color_map,wedgeprops = {'linewidth':␣
 ↪5},textprops={'fontsize': 13, 'color': 'black'})

#setting title for visual
ax0.set_title('Gender Distribution',{'font':'serif', 'size':15,'weight':'bold'})

                                        # creating pie chart for marital status
ax1 = fig.add_subplot(gs[0,1])

color_map = ["#3A7089", "#4b4b4c"]
ax1.pie(df['MaritalStatus'].value_counts().values,labels = df['MaritalStatus'].
 ↪value_counts().index,autopct = '%.1f%%',
        shadow = True,colors = color_map,wedgeprops = {'linewidth':␣
 ↪5},textprops={'fontsize': 13, 'color': 'black'})

#setting title for visual
ax1.set_title('Marital Status Distribution',{'font':'serif', 'size':15,'weight':
 ↪'bold'})

plt.show()
```



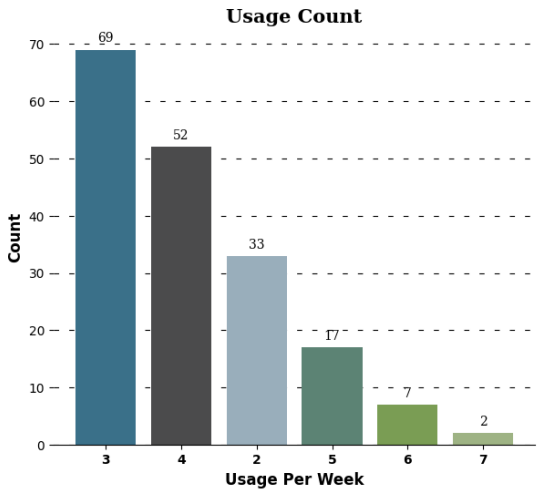


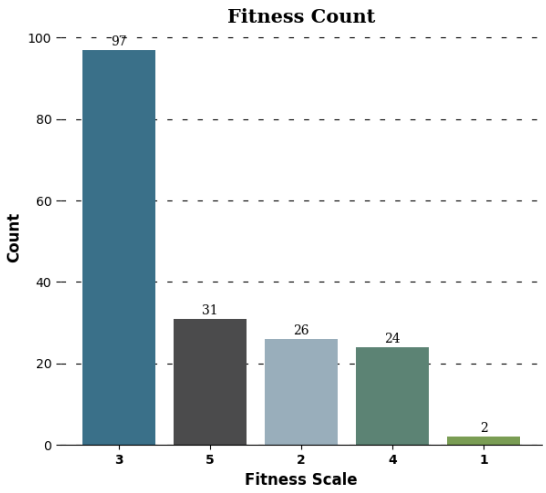
4.1.4 3.1.3 Buyer Fitness and Treadmill Usage

[25]:
```
#setting the plot style
fig = plt.figure(figsize = (15,10))
gs = fig.add_gridspec(2,2,height_ratios=[0.65, 0.35])

                                    # creating bar chart for usage␣
 ↪disribution

ax0 = fig.add_subplot(gs[0,0])
temp = df['Usage'].value_counts()
color_map = ["#3A7089", "#4b4b4c",'#99AEBB','#5C8374','#7A9D54','#9EB384']
ax0.bar(x=temp.index,height = temp.values,color = color_map,zorder = 2)

#adding the value_counts
for i in temp.index:
    ax0.text(i,temp[i]+2,temp[i],{'font':'serif','size' : 10},ha = 'center',va␣
 ↪= 'center')

#adding grid lines
ax0.grid(color = 'black',linestyle = '--', axis = 'y', zorder = 0, dashes =␣
 ↪(5,10))

#removing the axis lines
for s in ['top','left','right']:
    ax0.spines[s].set_visible(False)

#adding axis label
ax0.set_ylabel('Count',fontweight = 'bold',fontsize = 12)
```

```
ax0.set_xlabel('Usage Per Week',fontweight = 'bold',fontsize = 12)
ax0.set_xticklabels(temp.index,fontweight = 'bold')

#setting title for visual
ax0.set_title('Usage Count',{'font':'serif', 'size':15,'weight':'bold'})

                                        #creating a info table for usage

ax1 = fig.add_subplot(gs[1,0])
usage_info =␣
 ↪[['3','38%'],['4','29%'],['2','19%'],['5','9%'],['6','4%'],['7','1%']]
color_2d =␣
 ↪[["#3A7089",'#FFFFFF'],["#4b4b4c",'#FFFFFF'],['#99AEBB','#FFFFFF'],['#5C8374','#FFFFFF'],['
           ['#9EB384','#FFFFFF']]

table = ax1.table(cellText = usage_info, cellColours=color_2d,␣
 ↪cellLoc='center',colLabels =['Usage Per Week','Percent'],
                  colLoc = 'center',bbox =[0, 0, 1, 1])

table.set_fontsize(13)

#removing axis
ax1.axis('off')

                                        # creating bar chart for fitness scale

ax2 = fig.add_subplot(gs[0,1])
temp = df['Fitness'].value_counts()
color_map = ["#3A7089", "#4b4b4c",'#99AEBB','#5C8374','#7A9D54','#9EB384']
ax2.bar(x=temp.index,height = temp.values,color = color_map,zorder = 2)

#adding the value_counts
for i in temp.index:
    ax2.text(i,temp[i]+2,temp[i],{'font':'serif','size' : 10},ha = 'center',va␣
 ↪= 'center')

#adding grid lines
ax2.grid(color = 'black',linestyle = '--', axis = 'y', zorder = 0, dashes =␣
 ↪(5,10))

#removing the axis lines
for s in ['top','left','right']:
    ax2.spines[s].set_visible(False)

#adding axis label
ax2.set_ylabel('Count',fontweight = 'bold',fontsize = 12)
ax2.set_xlabel('Fitness Scale',fontweight = 'bold',fontsize = 12)
```

```
ax2.set_xticklabels(temp.index,fontweight = 'bold')

#setting title for visual
ax2.set_title('Fitness Count',{'font':'serif', 'size':15,'weight':'bold'})

                                        #creating a info table for usage

ax1 = fig.add_subplot(gs[1,1])
fitness_info = [['3','54%'],['5','17%'],['2','15%'],['4','13%'],['1','1%']]
color_2d =␣
 ↪[["#3A7089",'#FFFFFF'],["#4b4b4c",'#FFFFFF'],['#99AEBB','#FFFFFF'],['#5C8374','#FFFFFF'],['
table = ax1.table(cellText = fitness_info, cellColours=color_2d,␣
 ↪cellLoc='center',colLabels =['Fitness','Percent'],
                  colLoc = 'center',bbox =[0, 0, 1, 1])

table.set_fontsize(13)

#removing axis
ax1.axis('off')


plt.show()
```





Usage Per Week	Percent
3	38%
4	29%
2	19%
5	9%
6	4%
7	1%

Fitness	Percent
3	54%
5	17%
2	15%
4	13%
1	1%

4.1.5 Insights

- **Usage Frequency:** Approximately 85% of customers plan to use the treadmill 2 to 4 times per week, indicating that most users engage in moderate fitness routines. Only 15% of customers report using the treadmill 5 or more times per week, reflecting a smaller segment of highly active users.
- **Self-Evaluated Fitness Levels:** Over 54% of customers rate their fitness level as a 3 on a scale of 1 to 5. Additionally, 84% of customers have rated themselves 3 or higher, suggesting that the majority of customers perceive themselves to be at or above an average fitness level, indicating a generally health-conscious user base.

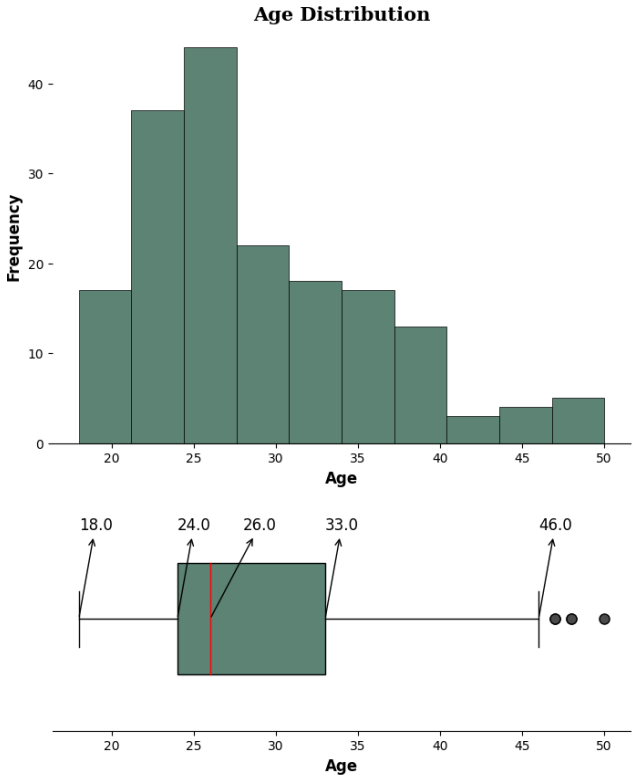
4.2 3.2 Numerical Variables

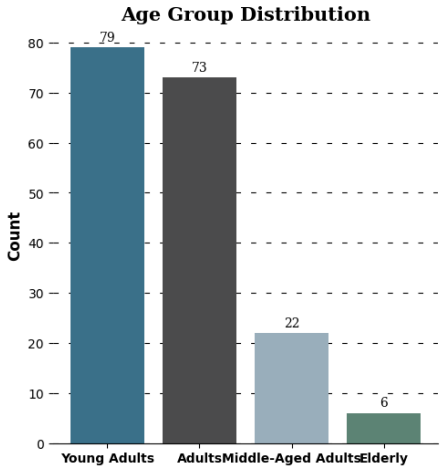
4.2.1 3.2.1 Customer Age Distribution

[26]:
```
#setting the plot style

fig = plt.figure(figsize = (15,10))
gs = fig.add_gridspec(2,2,height_ratios=[0.65, 0.35],width_ratios = [0.6,0.4])

                                    #creating age histogram

ax0 = fig.add_subplot(gs[0,0])

ax0.hist(df['Age'],color= '#5C8374',linewidth=0.5,edgecolor='black')
ax0.set_xlabel('Age',fontsize = 12,fontweight = 'bold')
ax0.set_ylabel('Frequency',fontsize = 12,fontweight = 'bold')

#removing the axis lines
for s in ['top','left','right']:
    ax0.spines[s].set_visible(False)

#setting title for visual
ax0.set_title('Age Distribution',{'font':'serif', 'size':15,'weight':'bold'})


                                    #creating box plot for age

ax1 = fig.add_subplot(gs[1,0])
boxplot = ax1.boxplot(x = df['Age'],vert = False,patch_artist = True,widths = 0.
 ↪5)

# Customize box and whisker colors
boxplot['boxes'][0].set(facecolor='#5C8374')

# Customize median line
boxplot['medians'][0].set(color='red')
```

```
# Customize outlier markers
for flier in boxplot['fliers']:
    flier.set(marker='o', markersize=8, markerfacecolor= "#4b4b4c")

#removing the axis lines
for s in ['top','left','right']:
    ax1.spines[s].set_visible(False)

#adding 5 point summary annotations
info = [i.get_xdata() for i in boxplot['whiskers']] #getting the␣
 ↪upperlimit,Q1,Q3 and lowerlimit

median = df['Age'].quantile(0.5) #getting Q2

for i,j in info: #using i,j here because of the output type of info list␣
 ↪comprehension

    ax1.annotate(text = f"{i:.1f}", xy = (i,1), xytext = (i,1.4),fontsize = 12,
                 arrowprops= dict(arrowstyle="<-", lw=1,␣
 ↪connectionstyle="arc,rad=0"))

    ax1.annotate(text = f"{j:.1f}", xy = (j,1), xytext = (j,1.4),fontsize = 12,
                 arrowprops= dict(arrowstyle="<-", lw=1,␣
 ↪connectionstyle="arc,rad=0"))

#adding the median separately because it was included in info list
ax1.annotate(text = f"{median:.1f}",xy = (median,1),xytext = (median + 2,1.
 ↪4),fontsize = 12,
            arrowprops= dict(arrowstyle="<-", lw=1,␣
 ↪connectionstyle="arc,rad=0"))

#removing y-axis ticks
ax1.set_yticks([])

#adding axis label
ax1.set_xlabel('Age',fontweight = 'bold',fontsize = 12)

                                    #creating age group bar chart

ax2 = fig.add_subplot(gs[0,1])
temp = df['age_group'].value_counts()
color_map = ["#3A7089", "#4b4b4c",'#99AEBB','#5C8374']
ax2.bar(x=temp.index,height = temp.values,color = color_map,zorder = 2)

#adding the value_counts
for i in temp.index:
```

```
    ax2.text(i,temp[i]+2,temp[i],{'font':'serif','size' : 10},ha = 'center',va␣
 ↪= 'center')

#adding grid lines
ax2.grid(color = 'black',linestyle = '--', axis = 'y', zorder = 0, dashes =␣
 ↪(5,10))

#removing the axis lines
for s in ['top','left','right']:
    ax2.spines[s].set_visible(False)

#adding axis label
ax2.set_ylabel('Count',fontweight = 'bold',fontsize = 12)
ax2.set_xticklabels(temp.index,fontweight = 'bold')

#setting title for visual
ax2.set_title('Age Group Distribution',{'font':'serif', 'size':15,'weight':
 ↪'bold'})

                                    #creating a table for group info

ax3 = fig.add_subplot(gs[1,1])
age_info = [['Young Adults','44%','18 to 25'],['Adults','41%','26 to␣
 ↪35'],['Middle Aged','12%','36 to 45'],
            ['Elder','3%','Above 45']]
color_2d =␣
 ↪[["#3A7089",'#FFFFFF','#FFFFFF'],["#4b4b4c",'#FFFFFF','#FFFFFF'],['#99AEBB','#FFFFFF','#FFF
            ['#5C8374','#FFFFFF','#FFFFFF']]

table = ax3.table(cellText = age_info, cellColours=color_2d,␣
 ↪cellLoc='center',colLabels =['Age','Probability','Group'],
                  colLoc = 'center',bbox =[0, 0, 1, 1])

table.set_fontsize(13)

#removing axis
ax3.axis('off')


plt.show()
```





Age	Probability	Group
Young Adults	44%	18 to 25
Adults	41%	26 to 35
Middle Aged	12%	36 to 45
Elder	3%	Above 45

4.2.2 Insights

- **Demographic Concentration:** 85% of customers are between the ages of 18 and 35, with a median age of 26. This indicates that the company's products are particularly appealing to a younger demographic, highlighting a strong interest among younger adults.
- **Outlier Detection: The age distribution contains 3 outliers, as identified in the box plot. These outliers may represent older customers who are exceptions to the company's typical customer profile.**

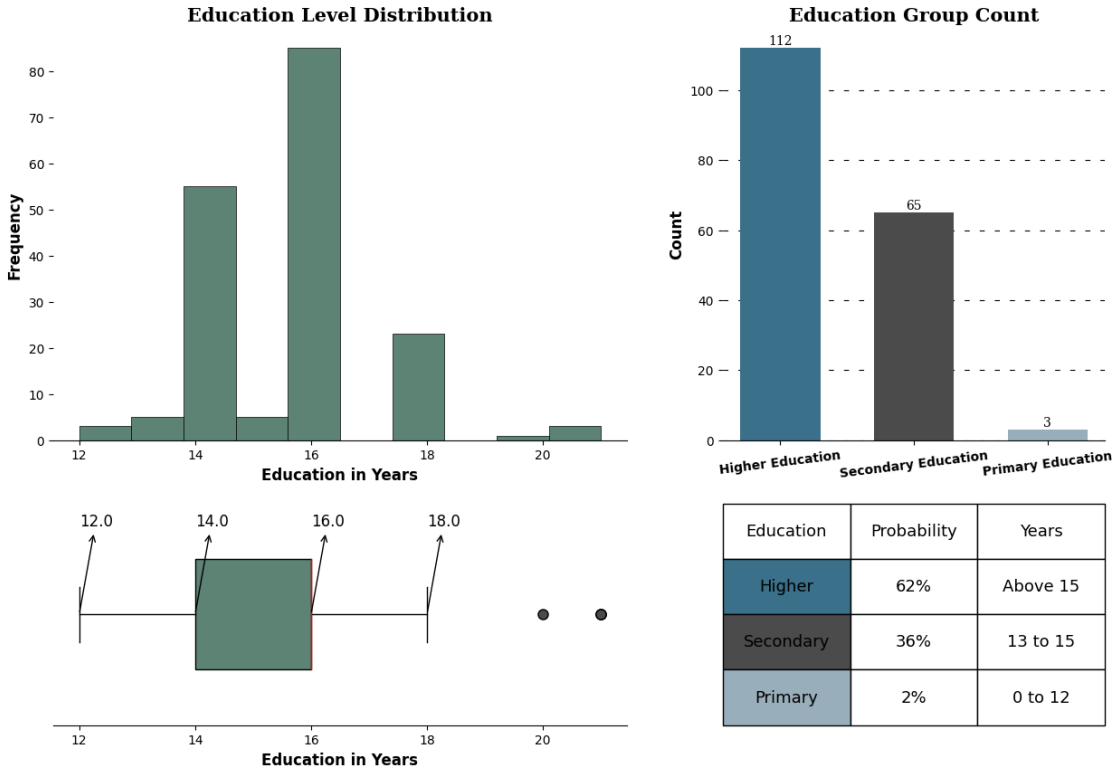
4.2.3 3.2.2 Customer Education Distribution

[61]:
```
#setting the plot style

fig = plt.figure(figsize = (15,10))
gs = fig.add_gridspec(2,2,height_ratios=[0.65, 0.35],width_ratios = [0.6,0.4])

                                    #creating education histogram

ax0 = fig.add_subplot(gs[0,0])

ax0.hist(df['Education'],color= '#5C8374',linewidth=0.5,edgecolor='black')
ax0.set_xlabel('Education in Years',fontsize = 12,fontweight = 'bold')
```

```
ax0.set_ylabel('Frequency',fontsize = 12,fontweight = 'bold')

#removing the axis lines
for s in ['top','left','right']:
    ax0.spines[s].set_visible(False)

#setting title for visual
ax0.set_title('Education Level Distribution',{'font':'serif', 'size':
 ↪15,'weight':'bold'})



                                    #creating box plot for education

ax1 = fig.add_subplot(gs[1,0])
boxplot = ax1.boxplot(x = df['Education'],vert = False,patch_artist =␣
 ↪True,widths = 0.5)

# Customize box and whisker colors
boxplot['boxes'][0].set(facecolor='#5C8374')

# Customize median line
boxplot['medians'][0].set(color='red')

# Customize outlier markers
for flier in boxplot['fliers']:
    flier.set(marker='o', markersize=8, markerfacecolor= "#4b4b4c")

#removing the axis lines
for s in ['top','left','right']:
    ax1.spines[s].set_visible(False)

#adding 5 point summary annotations
info = [i.get_xdata() for i in boxplot['whiskers']] #getting the␣
 ↪upperlimit,Q1,Q3 and lowerlimit

median = df['Education'].quantile(0.5) #getting Q2

for i,j in info: #using i,j here because of the output type of info list␣
 ↪comprehension

    ax1.annotate(text = f"{i:.1f}", xy = (i,1), xytext = (i,1.4),fontsize = 12,
                 arrowprops= dict(arrowstyle="<-", lw=1,␣
 ↪connectionstyle="arc,rad=0"))

    ax1.annotate(text = f"{j:.1f}", xy = (j,1), xytext = (j,1.4),fontsize = 12,
                 arrowprops= dict(arrowstyle="<-", lw=1,␣
 ↪connectionstyle="arc,rad=0"))
```

```
#removing y-axis ticks
ax1.set_yticks([])

#adding axis label
ax1.set_xlabel('Education in Years',fontweight = 'bold',fontsize = 12)

                                    #creating education group bar chart

ax2 = fig.add_subplot(gs[0,1])
temp = df['edu_group'].value_counts()
color_map = ["#3A7089", "#4b4b4c",'#99AEBB']
ax2.bar(x=temp.index,height = temp.values,color = color_map,zorder = 2,width =␣
 ↪0.6)

#adding the value_counts
for i in temp.index:
    ax2.text(i,temp[i]+2,temp[i],{'font':'serif','size' : 10},ha = 'center',va␣
 ↪= 'center')

#adding grid lines
ax2.grid(color = 'black',linestyle = '--', axis = 'y', zorder = 0, dashes =␣
 ↪(5,10))

#removing the axis lines
for s in ['top','left','right']:
    ax2.spines[s].set_visible(False)

#adding axis label
ax2.set_ylabel('Count',fontweight = 'bold',fontsize = 12)
ax2.set_xticklabels(temp.index,fontweight = 'bold',rotation = 7)

#setting title for visual
ax2.set_title('Education Group Count',{'font':'serif', 'size':15,'weight':
 ↪'bold'})


                                    #creating a table for group info

ax3 = fig.add_subplot(gs[1,1])
edu_info = [['Higher','62%','Above 15'],['Secondary','36%','13 to␣
 ↪15'],['Primary','2%','0 to 12']]
color_2d =␣
 ↪[["#3A7089",'#FFFFFF','#FFFFFF'],["#4b4b4c",'#FFFFFF','#FFFFFF'],['#99AEBB','#FFFFFF','#FFF
```

```
table = ax3.table(cellText = edu_info, cellColours=color_2d,␣
 ↪cellLoc='center',colLabels =['Education','Probability','Years'],
                  colLoc = 'center',bbox =[0, 0, 1, 1])

table.set_fontsize(13)

#removing axis
ax3.axis('off')


plt.show()
```



4.2.4 Insights :

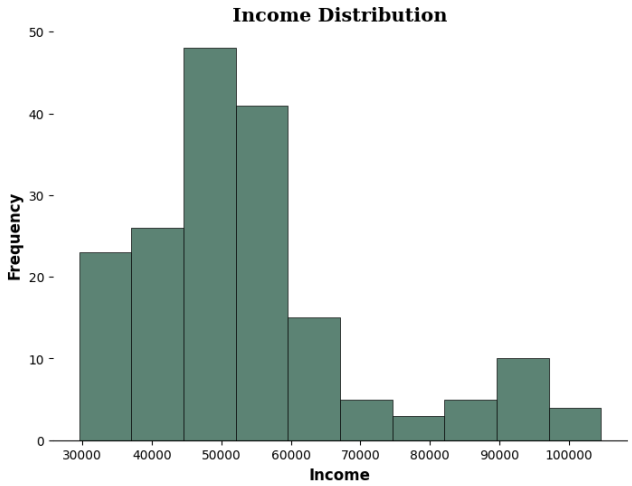
- **Education Level:** 98% of customers have completed more than 13 years of education, indicating that the company's products are highly popular among well-educated individuals. This trend suggests that health awareness—which is often associated with higher education levels—could be a significant factor driving these purchases.
- **Outlier Detection: The box plot reveals 2 outliers in the education data. These may represent customers with significantly different educational backgrounds, offering potential insights into smaller customer segments.**

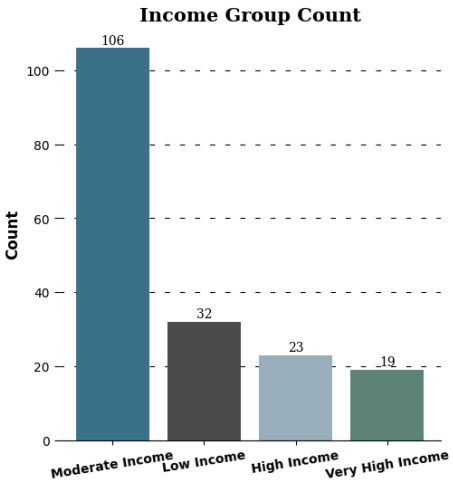
4.2.5 3.2.3 Customer Income Distribution

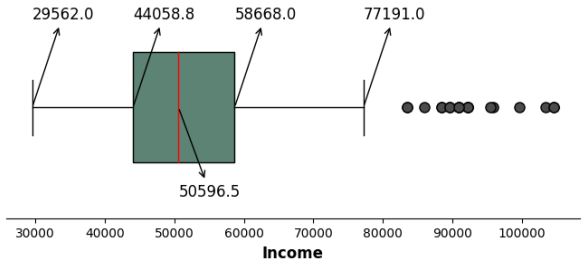
[28]:
```
#setting the plot style

fig = plt.figure(figsize = (15,10))
gs = fig.add_gridspec(2,2,height_ratios=[0.65, 0.35],width_ratios = [0.6,0.4])

                                    #creating Income histogram

ax0 = fig.add_subplot(gs[0,0])

ax0.hist(df['Income'],color= '#5C8374',linewidth=0.5,edgecolor='black')
ax0.set_xlabel('Income',fontsize = 12,fontweight = 'bold')
ax0.set_ylabel('Frequency',fontsize = 12,fontweight = 'bold')

#removing the axis lines
for s in ['top','left','right']:
    ax0.spines[s].set_visible(False)

#setting title for visual
ax0.set_title('Income Distribution',{'font':'serif', 'size':15,'weight':'bold'})


                                    #creating box plot for Income

ax1 = fig.add_subplot(gs[1,0])
boxplot = ax1.boxplot(x = df['Income'],vert = False,patch_artist = True,widths␣
 ↪= 0.5)

# Customize box and whisker colors
boxplot['boxes'][0].set(facecolor='#5C8374')

# Customize median line
boxplot['medians'][0].set(color='red')

# Customize outlier markers
for flier in boxplot['fliers']:
    flier.set(marker='o', markersize=8, markerfacecolor= "#4b4b4c")

#removing the axis lines
for s in ['top','left','right']:
    ax1.spines[s].set_visible(False)

#adding 5 point summary annotations
info = [i.get_xdata() for i in boxplot['whiskers']] #getting the␣
 ↪upperlimit,Q1,Q3 and lowerlimit
```

```
median = df['Income'].quantile(0.5) #getting Q2

for i,j in info: #using i,j here because of the output type of info list␣
 ↪comprehension

    ax1.annotate(text = f"{i:.1f}", xy = (i,1), xytext = (i,1.4),fontsize = 12,
                 arrowprops= dict(arrowstyle="<-", lw=1,␣
 ↪connectionstyle="arc,rad=0"))

    ax1.annotate(text = f"{j:.1f}", xy = (j,1), xytext = (j,1.4),fontsize = 12,
                 arrowprops= dict(arrowstyle="<-", lw=1,␣
 ↪connectionstyle="arc,rad=0"))

#adding the median separately because it was included in info list
ax1.annotate(text = f"{median:.1f}",xy = (median,1),xytext = (median,0.
 ↪6),fontsize = 12,
            arrowprops= dict(arrowstyle="<-", lw=1,␣
 ↪connectionstyle="arc,rad=0"))

#removing y-axis ticks
ax1.set_yticks([])

#adding axis label
ax1.set_xlabel('Income',fontweight = 'bold',fontsize = 12)

                                    #creating Income group bar chart

ax2 = fig.add_subplot(gs[0,1])
temp = df['income_group'].value_counts()
color_map = ["#3A7089", "#4b4b4c",'#99AEBB','#5C8374']
ax2.bar(x=temp.index,height = temp.values,color = color_map,zorder = 2)

#adding the value_counts
for i in temp.index:
    ax2.text(i,temp[i]+2,temp[i],{'font':'serif','size' : 10},ha = 'center',va␣
 ↪= 'center')

#adding grid lines
ax2.grid(color = 'black',linestyle = '--', axis = 'y', zorder = 0, dashes =␣
 ↪(5,10))

#removing the axis lines
for s in ['top','left','right']:
    ax2.spines[s].set_visible(False)

#adding axis label
```

```
ax2.set_ylabel('Count',fontweight = 'bold',fontsize = 12)
ax2.set_xticklabels(temp.index,fontweight = 'bold',rotation = 9)

#setting title for visual
ax2.set_title('Income Group Count',{'font':'serif', 'size':15,'weight':'bold'})

                                        #creating a table group info

ax3 = fig.add_subplot(gs[1,1])
inc_info = [['Low','18%','Below 40k'],['Moderate','59%','40k to␣
 ↪60k'],['High','13%','60k to 80k'],
            ['Vey High','10%','Above 80k']]
color_2d =␣
 ↪[["#4b4b4c",'#FFFFFF','#FFFFFF'],["#3A7089",'#FFFFFF','#FFFFFF'],['#99AEBB','#FFFFFF','#FFF
            ['#5C8374','#FFFFFF','#FFFFFF']]

table = ax3.table(cellText = inc_info, cellColours=color_2d, cellLoc='center',
                  colLabels =['Income Grp','Probability','Income($)'],
                  colLoc = 'center',bbox =[0, 0, 1, 1])

table.set_fontsize(13)

#removing axis
ax3.axis('off')
bin_range3 = [0,40000,60000,80000,float('inf')]
bin_labels3 = ['Low Income','Moderate Income','High Income','Very High Income']

plt.show()
```







Income Grp	Probability	Income($)
Low	18%	Below 40k
Moderate	59%	40k to 60k
High	13%	60k to 80k
Vey High	10%	Above 80k

4.2.6 Insights

- Income Distribution: Nearly 60% of customers belong to the income group of `$40,000` to `$60,000`, indicating a strong preference for the company's products among middle-income individuals.
- **Lower Income Segment**: Interestingly, 18% of customers fall into the under `$40,000` income group. This means that 77% of the total customer base earns below `$60,000`, while only 23% belong to the `$60,000` and above income group. This suggests that the company's products appeal more to middle and lower-income segments.
- **Outliers:** The box plot reveals the presence of several outliers in the income data, which may represent higher-income customers purchasing the products outside the primary income range.

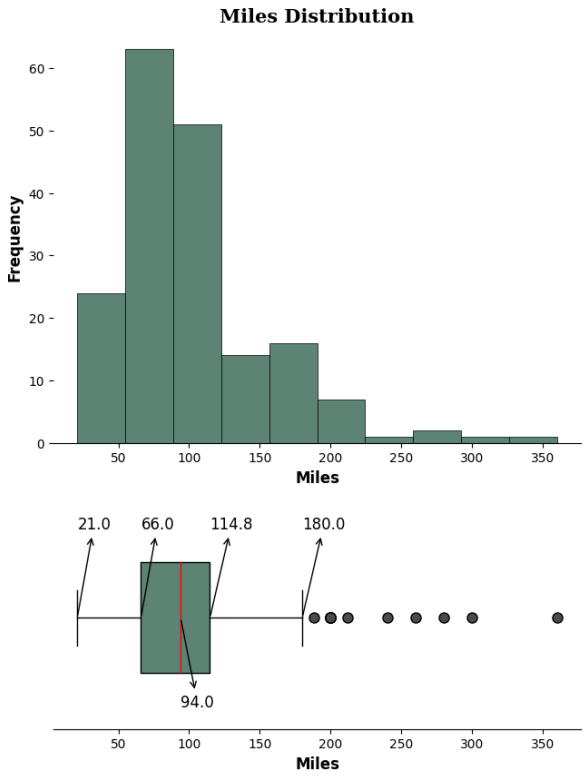
4.2.7 3.2.4 Customers Expected Weekly Mileage

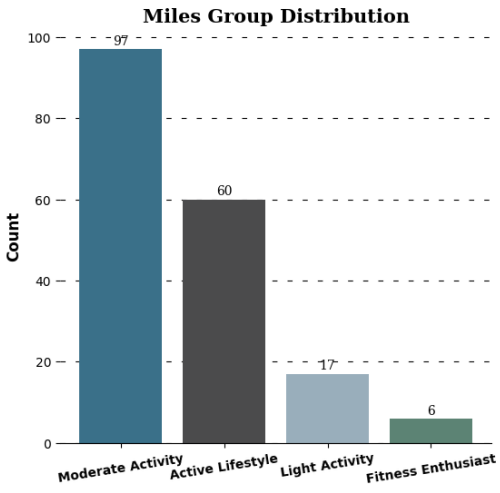
[29]:
```
#setting the plot style

fig = plt.figure(figsize = (15,10))
gs = fig.add_gridspec(2,2,height_ratios=[0.65, 0.35],width_ratios = [0.55,0.45])

                                    #creating miles histogram
```

```
ax0 = fig.add_subplot(gs[0,0])

ax0.hist(df['Miles'],color= '#5C8374',linewidth=0.5,edgecolor='black')
ax0.set_xlabel('Miles',fontsize = 12,fontweight = 'bold')
ax0.set_ylabel('Frequency',fontsize = 12,fontweight = 'bold')

#removing the axis lines
for s in ['top','left','right']:
    ax0.spines[s].set_visible(False)

#setting title for visual
ax0.set_title('Miles Distribution',{'font':'serif', 'size':15,'weight':'bold'})



                                    #creating box plot for miles

ax1 = fig.add_subplot(gs[1,0])
boxplot = ax1.boxplot(x = df['Miles'],vert = False,patch_artist = True,widths =␣
 ↪0.5)

# Customize box and whisker colors
boxplot['boxes'][0].set(facecolor='#5C8374')

# Customize median line
boxplot['medians'][0].set(color='red')

# Customize outlier markers
for flier in boxplot['fliers']:
    flier.set(marker='o', markersize=8, markerfacecolor= "#4b4b4c")

#removing the axis lines
for s in ['top','left','right']:
    ax1.spines[s].set_visible(False)

#adding 5 point summary annotations
info = [i.get_xdata() for i in boxplot['whiskers']] #getting the␣
 ↪upperlimit,Q1,Q3 and lowerlimit

median = df['Miles'].quantile(0.5) #getting Q2

for i,j in info: #using i,j here because of the output type of info list␣
 ↪comprehension

    ax1.annotate(text = f"{i:.1f}", xy = (i,1), xytext = (i,1.4),fontsize = 12,
                 arrowprops= dict(arrowstyle="<-", lw=1,␣
 ↪connectionstyle="arc,rad=0"))
```

```
    ax1.annotate(text = f"{j:.1f}", xy = (j,1), xytext = (j,1.4),fontsize = 12,
                 arrowprops= dict(arrowstyle="<-", lw=1,␣
 ↪connectionstyle="arc,rad=0"))

#adding the median separately because it was included in info list
ax1.annotate(text = f"{median:.1f}",xy = (median,1),xytext = (median,0.
 ↪6),fontsize = 12,
             arrowprops= dict(arrowstyle="<-", lw=1,␣
 ↪connectionstyle="arc,rad=0"))

#removing y-axis ticks
ax1.set_yticks([])

#adding axis label
ax1.set_xlabel('Miles',fontweight = 'bold',fontsize = 12)



                                    #creating Miles group bar chart

ax2 = fig.add_subplot(gs[0,1])
temp = df['miles_group'].value_counts()
color_map = ["#3A7089", "#4b4b4c",'#99AEBB','#5C8374']
ax2.bar(x=temp.index,height = temp.values,color = color_map,zorder = 2)

#adding the value_counts
for i in temp.index:
    ax2.text(i,temp[i]+2,temp[i],{'font':'serif','size' : 10},ha = 'center',va␣
 ↪= 'center')

#adding grid lines
ax2.grid(color = 'black',linestyle = '--', axis = 'y', zorder = 0, dashes =␣
 ↪(5,10))

#removing the axis lines
for s in ['top','left','right']:
    ax2.spines[s].set_visible(False)

#adding axis label
ax2.set_ylabel('Count',fontweight = 'bold',fontsize = 12)
ax2.set_xticklabels(temp.index,fontweight = 'bold',rotation = 9)

#setting title for visual
ax2.set_title('Miles Group Distribution',{'font':'serif', 'size':15,'weight':
 ↪'bold'})
```

```
                                        #creating a table for group info

ax3 = fig.add_subplot(gs[1,1])
miles_info = [['Light Activity','9%','0 to 50'],['Moderate Activity','54%','51␣
 ↪to 100'],['Active Lifestyle','34%','101 to 200'],
           ['Fitness Enthusiast','3%','Above 200']]
color_2d =␣
 ↪[['#99AEBB','#FFFFFF','#FFFFFF'],["#3A7089",'#FFFFFF','#FFFFFF'],["#4b4b4c",'#FFFFFF','#FFF
           ['#5C8374','#FFFFFF','#FFFFFF']]

table = ax3.table(cellText = miles_info, cellColours=color_2d,␣
 ↪cellLoc='center',colLabels =['Activity','Probability','Miles'],
                 colLoc = 'center',bbox =[0, 0, 1, 1])

table.set_fontsize(11)

#removing axis
ax3.axis('off')


plt.show()
```





Activity	Probability	Miles
Light Activity	9%	0 to 50
Moderate Activity	54%	51 to 100
Active Lifestyle	34%	101 to 200
Fitness Enthusiast	3%	Above 200

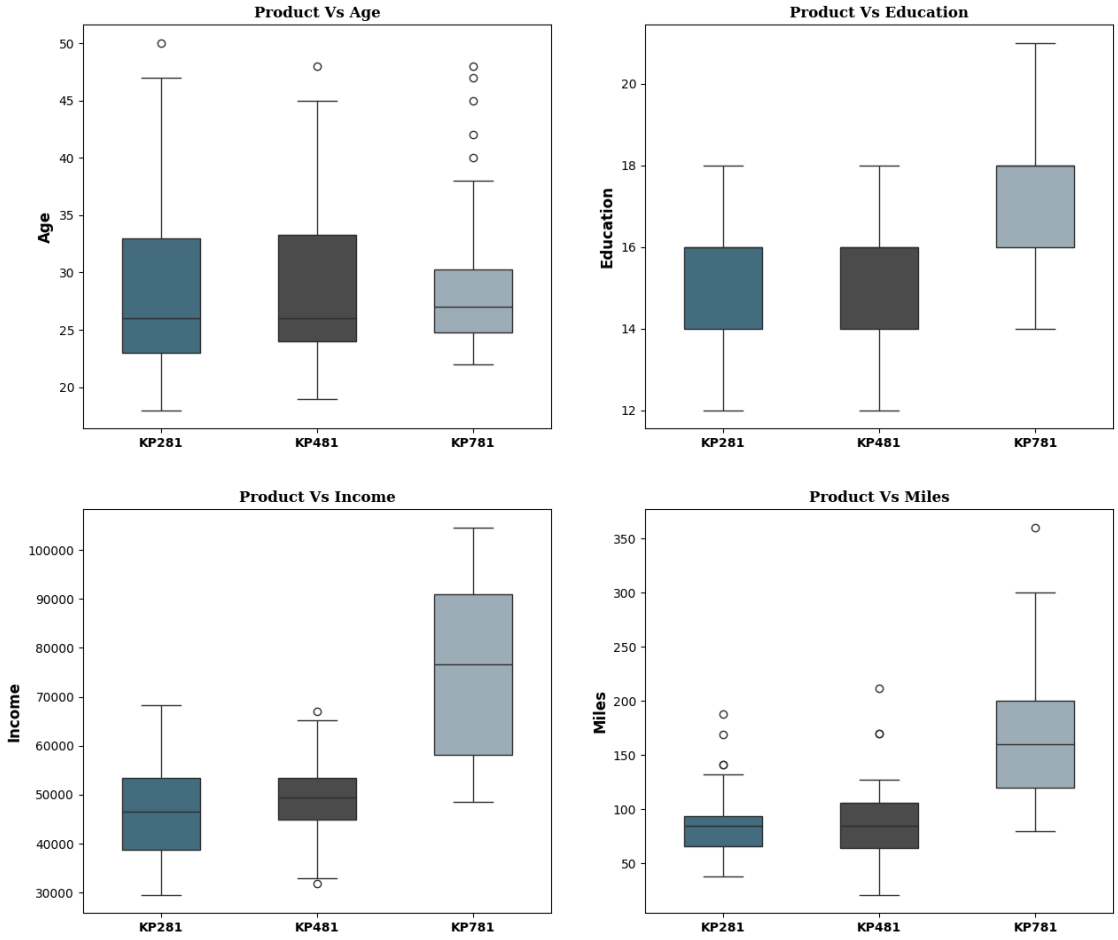
4.2.8 Insights:

- **Treadmill Usage:** Approximately 88% of customers plan to use the treadmill for 50 to 200 miles per week, with a median usage of 94 miles per week. This indicates that most customers are engaging in regular, moderate to high levels of activity.
- **Outliers:** The box plot reveals 8 outliers in the miles data, which may represent customers with exceptionally high or low treadmill usage patterns.

5 4. Bivariate Analysis

5.1 4.1 Analysis of Product Type

[32]:
```
#setting the plot style
fig = plt.figure(figsize = (15,13))
gs = fig.add_gridspec(2,2)

for i,j,k in [(0,0,'Age'),(0,1,'Education'),(1,0,'Income'),(1,1,'Miles')]:

    #plot position
    ax0 = fig.add_subplot(gs[i,j])

    #plot
    sns.boxplot(data = df, x = 'Product', y  = k ,ax = ax0,width = 0.5, palette␣
 ↪=["#3A7089", "#4b4b4c",'#99AEBB'])

    #plot title
    ax0.set_title(f'Product Vs {k}',{'font':'serif', 'size':12,'weight':'bold'})

    #customizing axis
    ax0.set_xticklabels(df['Product'].unique(),fontweight = 'bold')
    ax0.set_ylabel(f'{k}',fontweight = 'bold',fontsize = 12)
    ax0.set_xlabel('')

plt.show()
```



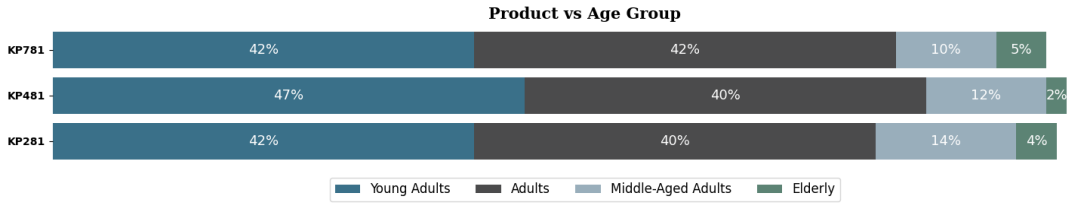
5.1.1 Insights:

- The analysis highlights a clear preference for the KP781 treadmill model among customers who demonstrate higher education levels, elevated income brackets, and a commitment to more intensive running routines, typically exceeding `150 miles per week`. This suggests that the `KP781` resonates particularly well with well-educated, high-income individuals who prioritize advanced fitness features and are likely seeking premium equipment for rigorous use.

5.2 4.2 Product Preferences Across Age

[36]:
```
# Setting the plot style and figure size
fig, ax0 = plt.subplots(figsize=(15, 3))

# Analyzing the relationship between product and age group
val = 'age_group'
```

```
# Creating a DataFrame that groups by Product and normalizes age group␣
 ↪distribution
df_grp = df.groupby('Product')[val].value_counts(normalize=True).round(2).
 ↪reset_index(name='proportion')
df_grp = df_grp.pivot(columns=val, index='Product', values='proportion')

# Setting up the base for stacked horizontal bars
temp = np.zeros(len(df_grp), dtype=float)
color_map = ["#3A7089", "#4b4b4c", '#99AEBB', '#5C8374']

# Plotting the stacked horizontal bar chart
for i, j in zip(df_grp.columns, color_map):
    ax0.barh(df_grp.index, width=df_grp[i], left=temp, label=i, color=j)
    temp += df_grp[i].values

# Inserting percentage labels inside the bars
temp = np.zeros(len(df_grp), dtype=float)
for i in df_grp.columns:
    for j, k in enumerate(df_grp[i]):
        if k == 0:
            continue
        ax0.text(k / 2 + temp[j], df_grp.index[j], f"{k:.0%}", va='center',␣
 ↪ha='center', fontsize=13, color='white')
    temp += df_grp[i].values

# Removing unnecessary axis lines
for s in ['top', 'left', 'right', 'bottom']:
    ax0.spines[s].set_visible(False)

# Customizing ticks and labels
ax0.set_xticks([])
ax0.set_yticklabels(df_grp.index, fontweight='bold')

# Adding the title to the plot
ax0.set_title('Product vs Age Group', fontdict={'family': 'serif', 'size': 15,␣
 ↪'weight': 'bold'})

# Adding a legend below the plot
ax0.legend(loc='lower center', bbox_to_anchor=(0.5, -0.3), ncol=4, fontsize=12)

# Displaying the plot
plt.tight_layout()
plt.show()
```



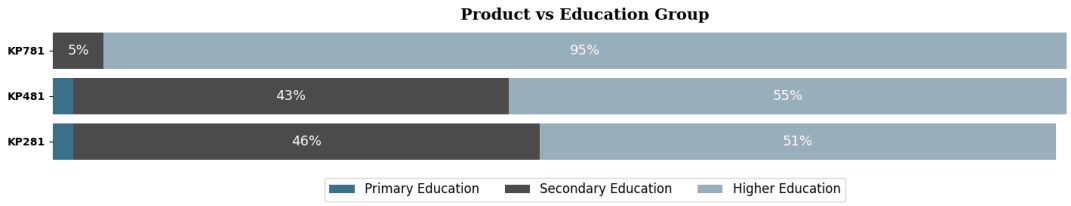
5.2.1 Insights:

- The analysis provided above distinctly demonstrates that there exists no strong correlation between age groups and product preferences. This is evident from the **nearly uniform distribution of age groups across all the products.**

5.3 4.3 Product Preferences Across Education Levels

[38]:
```
# Setting the plot style and figure size
fig, ax0 = plt.subplots(figsize=(15, 3))

# Analyzing the relationship between product and education group
val = 'edu_group'

# Creating a DataFrame that groups by Product and normalizes education group␣
 ↪distribution
df_grp = df.groupby('Product')[val].value_counts(normalize=True).round(2).
 ↪reset_index(name='proportion')
df_grp = df_grp.pivot(columns=val, index='Product', values='proportion')

# Setting up the base for stacked horizontal bars
temp = np.zeros(len(df_grp), dtype=float)
color_map = ["#3A7089", "#4b4b4c", '#99AEBB']

# Plotting the stacked horizontal bar chart
for i, j in zip(df_grp.columns, color_map):
    ax0.barh(df_grp.index, width=df_grp[i], left=temp, label=i, color=j)
    temp += df_grp[i].values

# Inserting percentage labels inside the bars
temp = np.zeros(len(df_grp), dtype=float)
for i in df_grp.columns:
    for j, k in enumerate(df_grp[i]):
        if k < 0.05:
            continue  # Skipping small percentages to avoid overcrowding
        ax0.text(k / 2 + temp[j], df_grp.index[j], f"{k:.0%}", va='center',␣
 ↪ha='center', fontsize=13, color='white')
    temp += df_grp[i].values
```

```
# Removing unnecessary axis lines
for s in ['top', 'left', 'right', 'bottom']:
    ax0.spines[s].set_visible(False)

# Customizing ticks and labels
ax0.set_xticks([])
ax0.set_yticklabels(df_grp.index, fontweight='bold')

# Adding the title to the plot
ax0.set_title('Product vs Education Group', fontdict={'family': 'serif', 'size':
 ↪ 15, 'weight': 'bold'})

# Adding a legend below the plot
ax0.legend(loc='lower center', bbox_to_anchor=(0.5, -0.3), ncol=3, fontsize=12)

# Displaying the plot
plt.tight_layout()
plt.show()
```



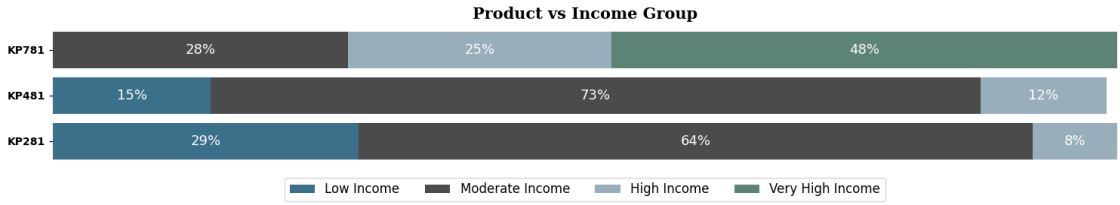
5.3.1 Insights:

- The analysis clearly shows a strong preference for the 'KP781 treadmill model among customers with higher education levels, indicating that more educated individuals are inclined towards premium products with advanced features.
- For the KP481 and KP281 treadmill models, the distribution of customers with secondary education and higher education is almost evenly split. This suggests that these models cater to a broader demographic, appealing to both moderately and highly educated individuals.

5.4 4.4 Product Preference Across Income Group

[40]:
```
# Setting the plot style and figure size
fig, ax0 = plt.subplots(figsize=(15, 3))

# Analyzing the relationship between product and income group
val = 'income_group'
```

```
# Creating a DataFrame that groups by Product and normalizes income group␣
 ↪distribution
df_grp = df.groupby('Product')[val].value_counts(normalize=True).round(2).
 ↪reset_index(name='proportion')
df_grp = df_grp.pivot(columns=val, index='Product', values='proportion')

# Setting up the base for stacked horizontal bars
temp = np.zeros(len(df_grp), dtype=float)
color_map =  ["#3A7089", "#4b4b4c", '#99AEBB', '#5C8374']

# Plotting the stacked horizontal bar chart
for i, j in zip(df_grp.columns, color_map):
    ax0.barh(df_grp.index, width=df_grp[i], left=temp, label=i, color=j)
    temp += df_grp[i].values

# Inserting percentage labels inside the bars
temp = np.zeros(len(df_grp), dtype=float)
for i in df_grp.columns:
    for j, k in enumerate(df_grp[i]):
        if k < 0.05:
            continue  # Skipping small percentages to avoid overcrowding
        ax0.text(k / 2 + temp[j], df_grp.index[j], f"{k:.0%}", va='center',␣
 ↪ha='center', fontsize=13, color='white')
    temp += df_grp[i].values

# Removing unnecessary axis lines
for s in ['top', 'left', 'right', 'bottom']:
    ax0.spines[s].set_visible(False)

# Customizing ticks and labels
ax0.set_xticks([])
ax0.set_yticklabels(df_grp.index, fontweight='bold')

# Adding the title to the plot
ax0.set_title('Product vs Income Group', fontdict={'family': 'serif', 'size':␣
 ↪15, 'weight': 'bold'})

# Adding a legend below the plot
ax0.legend(loc='lower center', bbox_to_anchor=(0.5, -0.3), ncol=4, fontsize=12)

# Displaying the plot
plt.tight_layout()
plt.show()
```



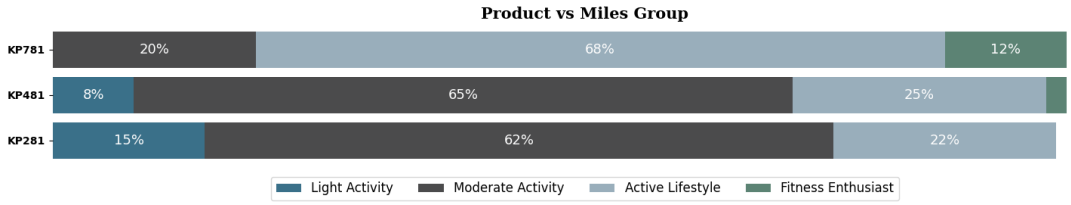
6 Insights:

- The KP781 treadmill model is predominantly preferred by customers in the Very High Income group, indicating that this premium product appeals most to affluent individuals seeking advanced features.
- For the KP481 and KP281 treadmill models, there is a clear preference among customers in the Moderate Income group, suggesting that these models align well with the needs of middle-income consumers looking for value without compromising on quality.

6.1 4.5 Product preference across customer weekly mileage

[42]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Setting the plot style and figure size
fig, ax0 = plt.subplots(figsize=(15, 3))

# Analyzing the relationship between product and miles group
val = 'miles_group'

# Creating a DataFrame that groups by Product and normalizes miles group␣
 ↪distribution
df_grp = df.groupby('Product')[val].value_counts(normalize=True).round(2).
 ↪reset_index(name='proportion')
df_grp = df_grp.pivot(columns=val, index='Product', values='proportion')

# Setting up the base for stacked horizontal bars
temp = np.zeros(len(df_grp), dtype=float)
color_map = ["#3A7089", "#4b4b4c", '#99AEBB', '#5C8374']

# Plotting the stacked horizontal bar chart
for i, j in zip(df_grp.columns, color_map):
    ax0.barh(df_grp.index, width=df_grp[i], left=temp, label=i, color=j)
    temp += df_grp[i].values

# Inserting percentage labels inside the bars
```

```
temp = np.zeros(len(df_grp), dtype=float)
for i in df_grp.columns:
    for j, k in enumerate(df_grp[i]):
        if k < 0.05:
            continue  # Skipping small percentages to avoid clutter
        ax0.text(k / 2 + temp[j], df_grp.index[j], f"{k:.0%}", va='center',␣
 ↪ha='center', fontsize=13, color='white')
    temp += df_grp[i].values

# Removing unnecessary axis lines
for s in ['top', 'left', 'right', 'bottom']:
    ax0.spines[s].set_visible(False)

# Customizing ticks and labels
ax0.set_xticks([])
ax0.set_yticklabels(df_grp.index, fontweight='bold')

# Adding the title to the plot
ax0.set_title('Product vs Miles Group', fontdict={'family': 'serif', 'size':␣
 ↪15, 'weight': 'bold'})

# Adding a legend below the plot
ax0.legend(loc='lower center', bbox_to_anchor=(0.5, -0.3), ncol=4, fontsize=12)

# Displaying the plot
plt.tight_layout()
plt.show()
```



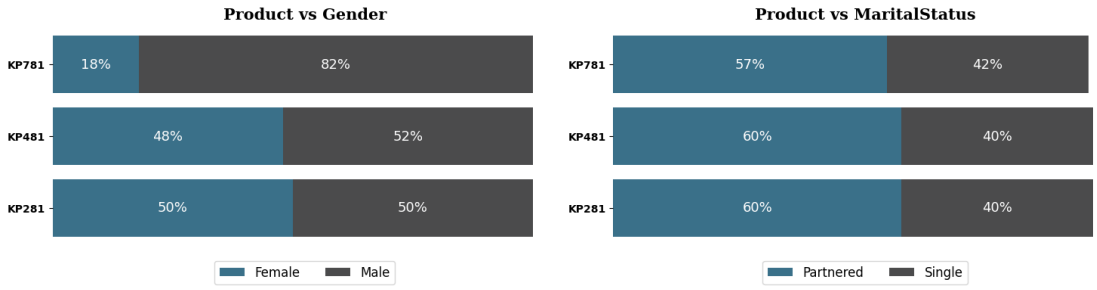
6.1.1 Insights:

- The KP781 treadmill model is predominantly favored by customers who plan to run 100 to 200 miles per week, indicating that it appeals to individuals with more intensive workout routines.
- Both the KP481 and KP281 treadmill models are more popular among customers planning to run 50 to 100 miles per week, suggesting that these models cater to those with moderate running habits.

6.2 4.6 Product Preference across Gender and Marital Status

[44]:
```
# Setting the plot style and figure size
fig = plt.figure(figsize=(15, 4))
gs = fig.add_gridspec(1, 2)

# Loop for creating the visualizations for 'Gender' and 'MaritalStatus'
for r, c, val in [(0, 0, 'Gender'), (0, 1, 'MaritalStatus')]:

    ax0 = fig.add_subplot(gs[r, c])

    # Creating required DataFrame
    df_grp = df.groupby('Product')[val].value_counts(normalize=True).round(2)
    df_grp.name = 'count'
    df_grp = df_grp.reset_index()
    df_grp = df_grp.pivot(columns=val, index='Product', values='count')

    # Initialize the base for the stacked horizontal bar chart
    temp = np.zeros(len(df_grp), dtype=float)
    color_map = ["#3A7089", "#4b4b4c"]  # Custom color scheme for the bars

    # Plotting the visual
    for i, j in zip(df_grp.columns, color_map):
        ax0.barh(df_grp.index, width=df_grp[i], left=temp, label=i, color=j)
        temp += df_grp[i].values

    # Inserting text for percentage labels
    temp = np.zeros(len(df_grp), dtype=float)

    for i in df_grp.columns:
        for j, k in enumerate(df_grp[i]):
            if k < 0.05:  # Skipping very small percentages
                continue
            ax0.text(k / 2 + temp[j], df_grp.index[j], f"{k:.0%}", va='center',␣
↪ha='center', fontsize=13, color='white')
        temp += df_grp[i].values

    # Removing unnecessary axis lines
    for s in ['top', 'left', 'right', 'bottom']:
        ax0.spines[s].set_visible(False)

    # Customizing ticks
    ax0.set_xticks([])
    ax0.set_yticklabels(df_grp.index, fontweight='bold')

    # Plot title
```

```
    ax0.set_title(f'Product vs {val}', fontdict={'family': 'serif', 'size': 15,␣
 ↪'weight': 'bold'})

    # Adding legend below the plot
    ax0.legend(loc='lower center', bbox_to_anchor=(0.5, -0.2), ncol=2,␣
 ↪fontsize=12)

# Displaying the plot
plt.tight_layout()
plt.show()
```



6.3 Insights:

1.Gender :

The KP781 treadmill model is predominantly preferred by male customers, indicating a potential alignment between the features of this premium model and male preferences, such as performance or advanced capabilities.

For the KP481 and KP281 models, the distribution of both genders is almost equal, suggesting these models appeal equally to both male and female customers, likely due to their affordability and balance of features suitable for general use.

2. Marital Status

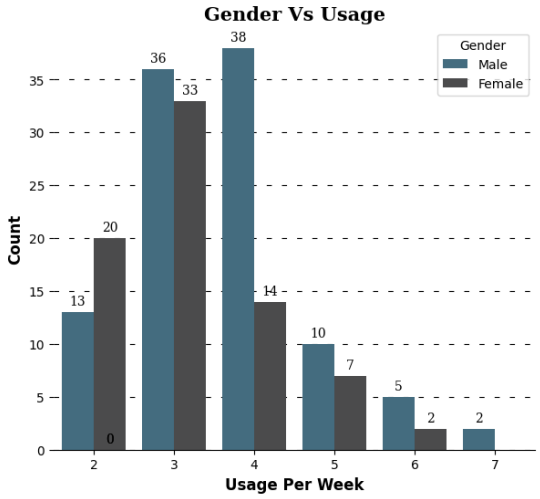
Across all three treadmill models, there is a uniform distribution of married and single customers. However, there is a slight preference among married customers, indicating that couples or families might lean towards purchasing treadmills for shared use, particularly in home gym setups.

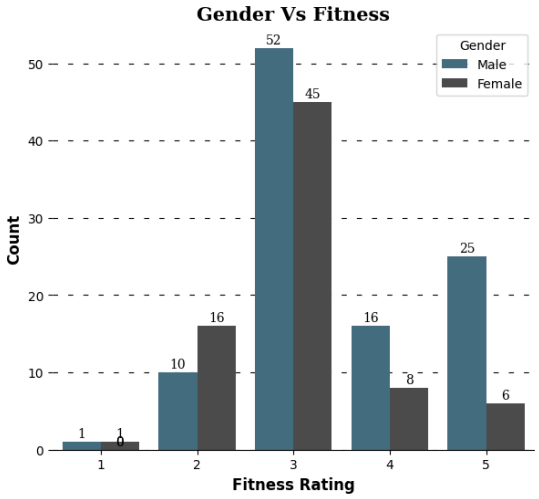
6.4 4.7 Gender vs Product Usage And Gender Vs Fitness

[45]:
```
#setting the plot style
fig = plt.figure(figsize = (15,6))
gs = fig.add_gridspec(1,2)

                                    # Usage Vs Gender

#creating bar plot
```

```
ax1 = fig.add_subplot(gs[0,0])

plot = sns.countplot(data = df, x = 'Usage', hue = 'Gender',order =␣
 ↪sorted(df['Usage'].unique()),
              ax = ax1,palette = ["#3A7089","#4b4b4c"],zorder = 2)

#adding the value_counts
for i in plot.patches:
    ax1.text(i.get_x()+0.2,i.get_height()+1,f'{i.get_height():.0f}',{'font':
 ↪'serif','size' : 10},ha = 'center',va = 'center')

#adding grid lines
ax1.grid(color = 'black',linestyle = '--', axis = 'y', zorder = 0, dashes =␣
 ↪(5,10))

#removing the axis lines
for s in ['top','left','right']:
    ax1.spines[s].set_visible(False)

#adding axis label
ax1.set_xlabel('Usage Per Week',fontweight = 'bold',fontsize = 12)
ax1.set_ylabel('Count',fontweight = 'bold',fontsize = 12)

#setting title for visual
ax1.set_title('Gender Vs Usage',{'font':'serif', 'size':15,'weight':'bold'})


                                    # Fitness Vs Gender

#creating bar plot
ax2 = fig.add_subplot(gs[0,1])

plot = sns.countplot(data = df, x = 'Fitness', hue = 'Gender',order =␣
 ↪sorted(df['Fitness'].unique()),
              ax = ax2,palette = ["#3A7089","#4b4b4c"],zorder = 2)

#adding the value_counts
for i in plot.patches:
    ax2.text(i.get_x()+0.2,i.get_height()+1,f'{i.get_height():.0f}',{'font':
 ↪'serif','size' : 10},ha = 'center',va = 'center')

#adding grid lines
ax2.grid(color = 'black',linestyle = '--', axis = 'y', zorder = 0, dashes =␣
 ↪(5,10))

#removing the axis lines
```

```
for s in ['top','left','right']:
    ax2.spines[s].set_visible(False)

#customizing axis labels
ax2.set_xlabel('Fitness Rating',fontweight = 'bold',fontsize = 12)
ax2.set_ylabel('Count',fontweight = 'bold',fontsize = 12)

#setting title for visual
ax2.set_title('Gender Vs Fitness',{'font':'serif', 'size':15,'weight':'bold'})

plt.show()
```





6.4.1 Insights:

1. Gender vs. Usage

70% of female customers plan to use the treadmill 2 to 3 times a week, indicating a more moderate approach to fitness routines.

In contrast, 70% of male customers plan to use the treadmill 3 to 4 times a week, suggesting that male customers tend to engage in slightly more frequent workout sessions.

2. Gender vs. Fitness

80% of female customers rated their fitness level between 2 and 3 on a scale of 1 to 5, reflecting a generally moderate self-assessment of fitness.

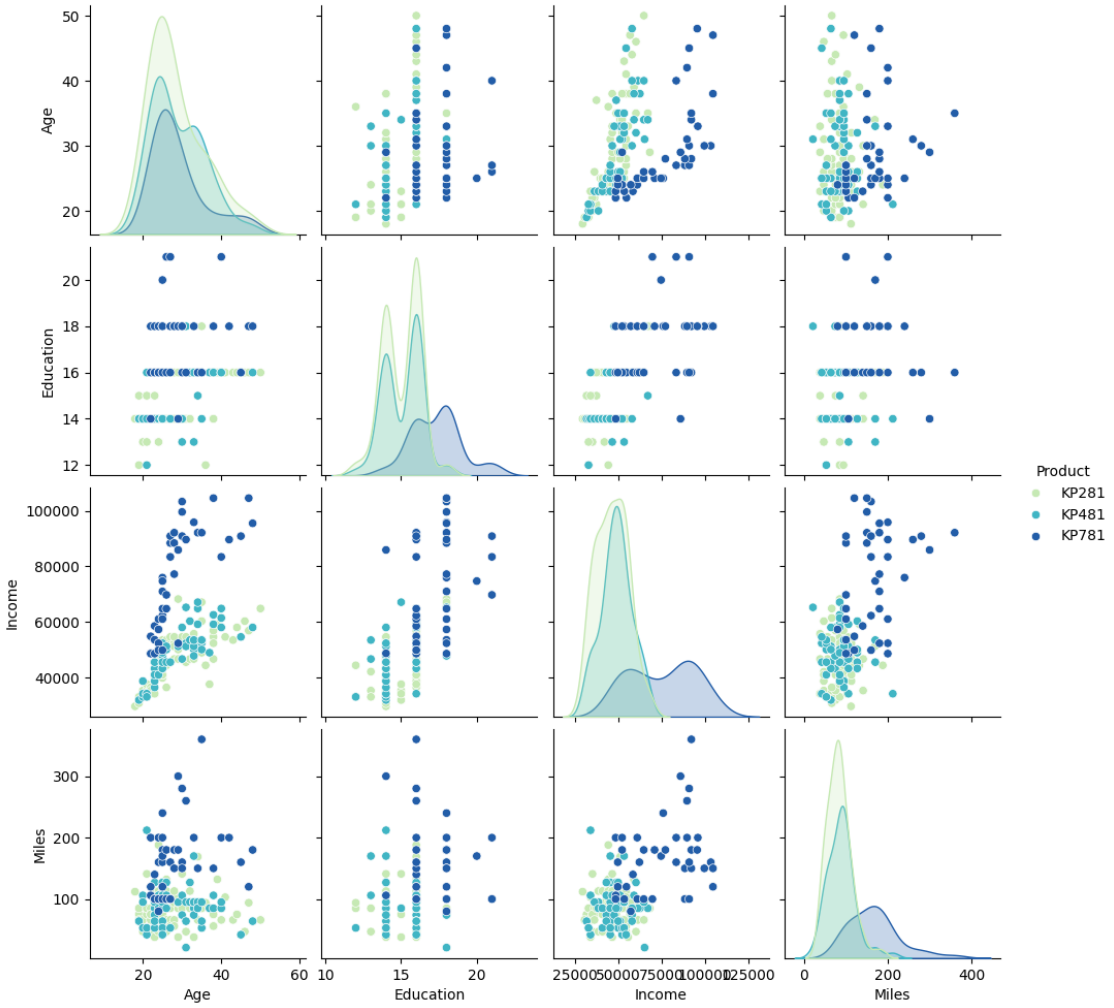
On the other hand, 90% of male customers rated their fitness between 3 and 5, indicating that male customers perceive themselves to have higher fitness levels and likely seek more performance-oriented fitness routines.

7 5. Correlation between Variables

7.1 5.1 Pairplot

[46]:
```
df_copy = copy.deepcopy(df)
```

[47]:
```
sns.pairplot(df_copy, hue ='Product', palette= 'YlGnBu')
plt.show()
```



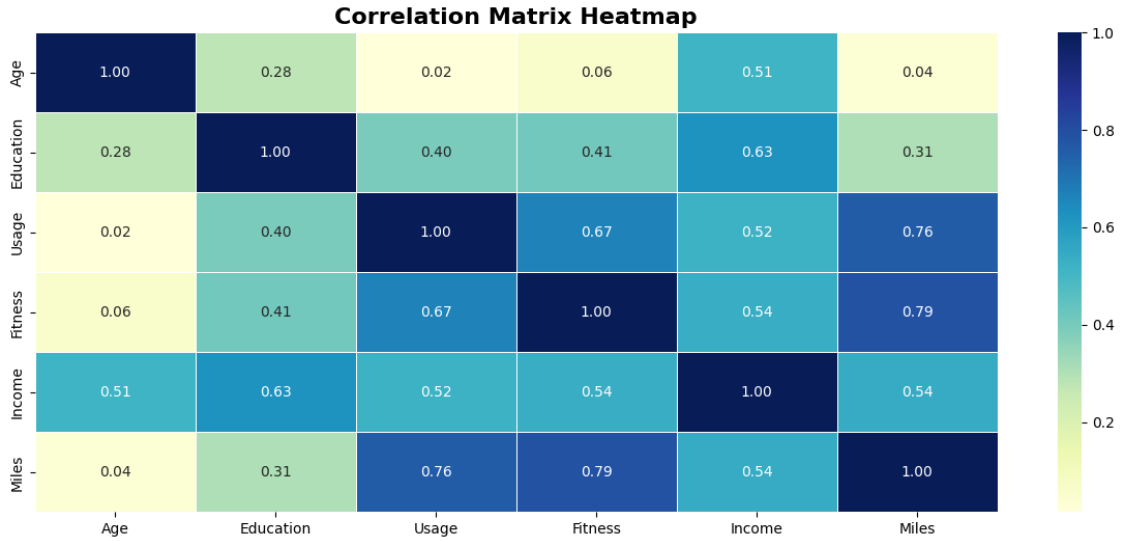
7.2 5.2 Heatmap

[48]:
```
# First we need to convert object into int datatype for usage and fitness␣
↪columns

df_copy['Usage'] = df_copy['Usage'].astype('int')
df_copy['Fitness'] = df_copy['Fitness'].astype('int')
```

```
df_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 13 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   
 9   age_group      180 non-null    category
 10  edu_group      180 non-null    category
 11  income_group   180 non-null    category
 12  miles_group    180 non-null    category
dtypes: category(4), int64(6), object(3)
memory usage: 14.2+ KB
```

[51]:
```
# Selecting only numerical columns from the DataFrame
numerical_df = df_copy.select_dtypes(include=['float64', 'int64'])

# Calculating the correlation matrix
corr_mat = numerical_df.corr()

# Setting the figure size
plt.figure(figsize=(15, 6))

# Creating the heatmap with enhanced formatting
sns.heatmap(corr_mat, annot=True, cmap="YlGnBu", fmt='.2f', linewidths=0.5,␣
 ↪annot_kws={"size": 10})

# Adding a title to the heatmap
plt.title('Correlation Matrix Heatmap', fontsize=16, fontweight='bold')

# Displaying the heatmap
plt.show()
```



7.2.1 Insights:

- **1. Age vs. Income Correlation**

From both the pair plot and heatmap, we observe a positive correlation between Age and Income, indicating that older customers tend to have higher income levels, a pattern often linked to career progression and earnings growth over time.

- **2. Education vs. Income and Fitness**

Education and Income are highly correlated, as expected, reflecting the well-established link between higher education and higher earning potential. Additionally, Education shows a significant correlation with Fitness ratings and Treadmill Usage, suggesting that well-educated individuals tend to prioritize their fitness and are more likely to use the treadmill regularly.

- **3. Usage vs. Fitness and Miles**

Treadmill Usage is strongly correlated with both Fitness and Miles, affirming the intuitive relationship that the more frequently customers use the treadmill, the fitter they become, and the more miles they log.

8 6. Computing Probability - Marginal, Conditional Probability

8.1 6.1 Probability of Product Purchase Based on Gender

[52]:
```
pd.crosstab(index =df['Product'],columns = df['Gender'],margins =␣
↪True,normalize = True ).round(2)
```

[52]:
```
Gender   Female  Male   All
Product
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
```

8.1.1 Insights:

1. The overall probability of a treadmill being purchased by a female is 42%.

- The conditional probability of a female customer purchasing a specific treadmill model is as follows:

  KP281: 22%

  KP481: 16%

  KP781: 4%

  This shows that while female customers are more inclined towards the KP281 and KP481 models, only a small percentage opt for the premium KP781.

2. The overall probability of a treadmill being purchased by a male is 58%. The conditional probability of a male customer purchasing a specific treadmill model is:

KP281: 22%

KP481: 17%

KP781: 18%

Male customers demonstrate a more balanced distribution across all three treadmill models, with a particularly stronger preference for the high-end KP781 compared to females.

8.2 6.2 Probability of product purchase based on Age

[53]:
```
pd.crosstab(index =df['Product'],columns = df['age_group'],margins =␣
↪True,normalize = True ).round(2)
```

[53]:
```
age_group  Young Adults  Adults  Middle-Aged Adults  Elderly   All
Product
KP281              0.19    0.18                0.06     0.02  0.44
KP481              0.16    0.13                0.04     0.01  0.33
KP781              0.09    0.09                0.02     0.01  0.22
All                0.44    0.41                0.12     0.03  1.00
```

8.2.1 Insights:

1. The probability of a treadmill being purchased by a Young Adult (18-25) is 44%.The conditional probability of a Young Adult purchasing a specific treadmill model is:

- KP281: 19%
- KP481: 16%
- KP781: 9%

This indicates that Young Adults have a notable preference for the KP281 and KP481 models, with fewer opting for the high-end KP781.

2. The probability of a treadmill being purchased by an Adult (26-35) is 41%. The conditional probability of an Adult purchasing a specific treadmill model is:
   - KP281: 18%
   - KP481: 13%
   - KP781: 9%

Adults exhibit similar preferences to Young Adults, with a slightly lower inclination towards the KP481 and KP781 models.

3. The probability of a treadmill being purchased by a Middle Aged individual (36-45) is 12%. This suggests that Middle Aged customers represent a smaller portion of the overall customer base, with no significant preference for any specific model.

4. The probability of a treadmill being purchased by an Elder (Above 45) is only 3%.Elders make up a very small share of the customer base, indicating that the company's products are less popular among older customers.

8.3 6.3 Probability of product purchase based on Education level

[54]:
```
pd.crosstab(index =df['Product'],columns = df['edu_group'],margins =␣
↪True,normalize = True ).round(2)
```

[54]:
```
edu_group  Primary Education  Secondary Education  Higher Education   All
Product
KP281                   0.01                 0.21              0.23  0.44
KP481                   0.01                 0.14              0.18  0.33
KP781                   0.00                 0.01              0.21  0.22
All                     0.02                 0.36              0.62  1.00
```

8.3.1 Insights:

1. The probability of a treadmill being purchased by a customer with Higher Education (above 15 years) is 62%.
   - The conditional probability of purchasing a specific treadmill model for customers with Higher Education is:
     - For Treadmill model KP281 - **23%**
     - For Treadmill model KP481 - **18%**
     - For Treadmill model KP781 - **21%**

     This indicates a strong preference among highly educated customers for the KP781, a premium model, along with balanced interest in the KP281 and KP481 models.

2. The probability of a treadmill being purchased by a customer with Secondary Education (13-15 years) is 36%..

- The conditional probability of purchasing a specific treadmill model for customers with Secondary Education is-
  - For Treadmill model KP281 - **21%**
  - For Treadmill model KP481 - **14%**
  - For Treadmill model KP781 - **1%**

  Customers with secondary education tend to prefer the KP281, with significantly lower interest in the KP781, showing a stronger inclination towards more affordable options.

3. The probability of a treadmill being purchased by a customer with Primary Education (0 to 12 years) is only 2%. This reflects that customers with primary education make up a very small portion of the overall customer base, showing minimal interest in purchasing treadmills, possibly due to affordability or lifestyle differences.

8.4 6.4 Probability of product purchase based on Income

[55]:
```
pd.crosstab(index =df['Product'],columns = df['income_group'],margins =␣
↪True,normalize = True ).round(2)
```

[55]:

income_group	Low Income	Moderate Income	High Income	Very High Income	All
Product					
KP281	0.13	0.28	0.03	0.00	0.44
KP481	0.05	0.24	0.04	0.00	0.33
KP781	0.00	0.06	0.06	0.11	0.22
All	0.18	0.59	0.13	0.11	1.00

9 Insights :

- The probability of a treadmill being purchased by a customer with Low Income (<40k) is 18%. The conditional probability of purchasing a specific treadmill model for customers with Low Income is:
- KP281: 13%
- KP481: 5%
- KP781: 0%

Customers with low income predominantly prefer the KP281 model, with almost no interest in the high-end KP781, likely due to affordability constraints. * The probability of a treadmill being purchased by a customer with Moderate Income (40k - 60k) is 59%.

`The conditional probability of purchasing a specific treadmill model for customers with Moderat`

- KP281: 28%
- KP481: 24%
- KP781: 6%

Moderate-income customers show a balanced interest in both the KP281 and KP481 models, reflecting their preference for affordable, yet feature-rich products. Only a small portion opts for the premium KP781.

- The probability of a treadmill being purchased by a customer with High Income (60k - 80k) is 13%. The conditional probability of purchasing a specific treadmill model for customers with High Income is:
- KP281: 3%
- KP481: 4%
- KP781: 6%

High-income customers exhibit a greater preference for the KP781 compared to lower-income groups, though a significant portion still chooses the KP281 and KP481 models.

- The probability of a treadmill being purchased by a customer with Very High Income (>80k) is 11%.

The conditional probability of purchasing a specific treadmill model for customers with Very High Income is:

- KP281: 0%
- KP481: 0%
- KP781: 11%

Very high-income customers overwhelmingly prefer the premium KP781, with no interest in the lower-tier KP281 and KP481 models, reflecting a clear preference for advanced, high-end features.

9.1 6.5 Probability of product purchase w.r.t. Marital Status

[58]:
```
pd.crosstab(index =df['Product'],columns = df['MaritalStatus'],margins =␣
↪True,normalize = True ).round(2)
```

[58]:
```
MaritalStatus  Partnered  Single   All
Product
KP281               0.27    0.18  0.44
KP481               0.20    0.13  0.33
KP781               0.13    0.09  0.22
All                 0.59    0.41  1.00
```

9.1.1 Insights:

- The probability of a treadmill being purchased by a married customer is 59%.

The conditional probability of purchasing a specific treadmill model given that the customer is married is:

- KP281: 27%
- KP481: 20%

- KP781: 13%

Married customers show a strong preference for the KP281 and KP481 models, indicating that affordability and practicality may be important factors for couples or families. A notable portion also prefers the premium KP781 model, reflecting interest in high-end features.

- The probability of a treadmill being purchased by an unmarried customer is 41%.

The conditional probability of purchasing a specific treadmill model given that the customer is unmarried is:

- KP281: 18%
- KP481: 13%
- KP781: 9%

Unmarried customers also lean towards the KP281 and KP481 models, but to a lesser extent compared to married customers. Their preference for the premium KP781 is lower, suggesting they may prioritize cost-effectiveness over advanced features.

9.2 6.6 Probability of product purchase based on Weekly Usage

[59]:
```
pd.crosstab(index =df['Product'],columns = df['Usage'],margins = True,normalize
↪= True ).round(2)
```

[59]:

Usage	2	3	4	5	6	7	All
Product							
KP281	0.11	0.21	0.12	0.01	0.00	0.00	0.44
KP481	0.08	0.17	0.07	0.02	0.00	0.00	0.33
KP781	0.00	0.01	0.10	0.07	0.04	0.01	0.22
All	0.18	0.38	0.29	0.09	0.04	0.01	1.00

9.2.1 Insights:

- The probability of a treadmill being purchased by a customer who uses it 3 times per week is 38%.

The conditional probability of purchasing a specific treadmill model for customers with 3 times per week usage is:

* KP281: 21%

* KP481: 17%

* KP781: 1%

This suggests that customers with moderate usage (3 times per week) prefer the more affordable KP281 and KP481 models, with very few opting for the premium KP781.

- The probability of a treadmill being purchased by a customer who uses it 4 times per week is 29%.

The conditional probability of purchasing a specific treadmill model for customers with 4 times per week usage is:

* KP281: 12%

* KP481: 7%

* KP781: 10%

Customers with more frequent usage (4 times per week) demonstrate a higher preference for the premium KP781, indicating that they may seek advanced features to support their more active fitness routines.

- The probability of a treadmill being purchased by a customer who uses it 2 times per week is 18%.

The conditional probability of purchasing a specific treadmill model for customers with 2 times per week usage is:

* KP281: 11%

* KP481: 8%

* KP781: 0%

For light users (2 times per week), the KP281 and KP481 models remain the popular choices, with no interest in the premium KP781 model, likely due to its advanced features not being necessary for this usage level.

9.3 6.7 Probability of product purchase w.r.t. Customer Fitness

[ ]:
```
pd.crosstab(index =df['Product'],columns = df['Fitness'],margins =␣
↪True,normalize = True ).round(2)
```

[ ]:
```
Fitness     1     2     3     4     5   All
Product
KP281    0.01  0.08  0.30  0.05  0.01  0.44
KP481    0.01  0.07  0.22  0.04  0.00  0.33
KP781    0.00  0.00  0.02  0.04  0.16  0.22
All      0.01  0.14  0.54  0.13  0.17  1.00
```

9.3.1 Insights:

1. The probability of a treadmill being purchased by a customer with average fitness (3) is 54%.

The conditional probability of purchasing a specific treadmill model for customers with average fitness is:

- KP281: 30%
- KP481: 22%
- KP781: 2%

This suggests that customers with average fitness levels predominantly prefer the KP281 and KP481 models, while very few opt for the premium KP781.

2. The probability of a treadmill being purchased by a customer with a fitness level of 2, 4, or 5 is approximately 15%.

- This reflects a more balanced distribution across various fitness levels, with these customers making up a smaller portion of the total treadmill purchases.

3. The probability of a treadmill being purchased by a customer with very low fitness (1) is only 1%.

- This indicates that customers with very low fitness levels are the least likely to purchase a treadmill, possibly due to a lack of engagement in fitness activities or unfamiliarity with treadmill use.

9.4 6.8 Probability of product purchase based on the weekly mileage

[60]:
```
pd.crosstab(index =df['Product'],columns = df['miles_group'],margins =␣
↪True,normalize = True ).round(2)
```

[60]:
```
miles_group  Light Activity  Moderate Activity  Active Lifestyle  \
Product
KP281                  0.07               0.28              0.10
KP481                  0.03               0.22              0.08
KP781                  0.00               0.04              0.15
All                    0.09               0.54              0.33

miles_group  Fitness Enthusiast   All
Product
KP281                      0.00  0.44
KP481                      0.01  0.33
KP781                      0.03  0.22
All                        0.03  1.00
```

9.4.1 Insights:

- The probability of a treadmill being purchased by a customer with a Light Activity lifestyle (0 to 50 miles/week) is 9%.

The conditional probability of purchasing a specific treadmill model for customers with a light activity lifestyle is:

* KP281: 7%

* KP481: 3%

* KP781: 0%

Customers with light activity prefer the entry-level KP281, with very little interest in the mid-range KP481 or the premium KP781 models, reflecting lower engagement in high-performance features.

- The probability of a treadmill being purchased by a customer with a Moderate Activity lifestyle (51 to 100 miles/week) is 54%.

The conditional probability of purchasing a specific treadmill model for customers with a moderate activity lifestyle is:

- KP281: 28%
- KP481: 22%
- KP781: 4%

Customers with moderate activity show a balanced preference for the KP281 and KP481 models, with a small percentage opting for the premium KP781, indicating that regular users value both affordability and features

- The probability of a treadmill being purchased by a customer with an Active Lifestyle (100 to 200 miles/week) is 33%.

The conditional probability of purchasing a specific treadmill model for customers with an active lifestyle is:

* `KP281: 10%`

* `KP481: 8%`

* `KP781: 15%`

Active lifestyle customers show a stronger inclination towards the premium KP781 model, as their usage likely demands more advanced features and durability.

- The probability of a treadmill being purchased by a customer who is a Fitness Enthusiast (more than 200 miles/week) is only 3%.
  - This indicates that fitness enthusiasts represent a small segment of the overall market, possibly because they seek specialized equipment beyond standard consumer models.

10 7. Customer Profiling

Based on above analysis

- Probability of purchase of KP281 = 44%
- Probability of purchase of KP481 = 33%
- Probability of purchase of KP781 = 22%
- `Customer Profile` for `KP281` Treadmill:
  - Age of customer mainly between 18 to 35 years with few between 35 to 50 years
  - Education level of customer 13 years and above
  - Annual Income of customer below USD 60,000
  - Weekly Usage - 2 to 4 times
  - Fitness Scale - 2 to 4
  - Weekly Running Mileage - 50 to 100 miles

- `Customer Profile` for `KP481` Treadmill:
  - Age of customer mainly between 18 to 35 years with few between 35 to 50 years
  - Education level of customer 13 years and above
  - Annual Income of customer between USD 40,000 to USD 80,000
  - Weekly Usage - 2 to 4 times
  - Fitness Scale - 2 to 4
  - Weekly Running Mileage - 50 to 200 miles
- `Customer Profile` for `KP781` Treadmill:
  - Gender - Male
  - Age of customer between 18 to 35 years
  - Education level of customer 15 years and above
  - Annual Income of customer USD 80,000 and above
  - Weekly Usage - 4 to 7 times
  - Fitness Scale - 3 to 5
  - Weekly Running Mileage - 100 miles and above

11 8. Recommendations

1. **Targeted Marketing for KP781**

The KP781 model shows a gender gap, with only 18% of sales from female customers. To increase female engagement:

- Exclusive Promotions: Offer promotions and discounts tailored to female buyers, potentially bundling fitness gear or accessories.
- Trial Programs: Introduce trial programs where women can test the treadmill before purchase, building product confidence.
- Targeted Messaging: Ensure marketing campaigns highlight features that appeal to women, such as custom workout programs and design aesthetics.

2. **Affordable Pricing and Payment Plans for KP281 and KP481**

`For the price-sensitive demographic:`

- Competitive Pricing: Keep KP281 and KP481 at an affordable price point that aligns with younger, moderate-income customers.
- Flexible Payment Plans: Offer interest-free payment options to make the products more accessible.
- Subscription Bundles: Provide value-added bundles, pairing the treadmill with workout plans or fitness app subscriptions.

3. **User-Friendly App Integration**

   Enhance engagement by developing a companion app for treadmill users:

   - Mileage Tracking & Feedback: Allow users to track mileage, get real-time performance feedback, and see progress.

- Personalized Workouts: Provide custom workout recommendations based on user fitness goals and treadmill usage.
- Goal Setting & Motivation: Implement goal-setting features with rewards or badges to keep users motivated.

These strategies will drive growth, boost engagement, and ensure the treadmills meet the needs of a diverse customer base.

[ ]: