EDA PORTFOLIO PROJECT

Asad Raza

AeroFit Treadmills   ATOM Camp Grey Cohort

Table of Contents

EDA Portfolio Project - Treadmill Buyer Profile – Asad Raza

The analysis was performed using Python in the Google Colab Notebook tool.

Data Exploration and Processing:

The process of data exploration began after uploading the aerofit_treadmill_data.csv file. The numpy, pandas, Matplotlib, Searborn libraries were imported as a second step and the Google drive was mounted. In order to filter warnings passed the code to ignore warnings.

```
[1] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    import warnings
    warnings.filterwarnings('ignore') #Importing data
```

```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

Then proceeded with reading the data frame.

```
[3] df = pd.read_csv('//content/aerofit_treadmill_data.csv') #Reading the data
```

Just to have an idea whether the code is running ran the head and tail functions to get a birds eye view of the top 5 and bottom 5 rows.

Data Frame Head

```
df.head() #Displaying the first 5 rows of the data
```

	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

Data Frame Tail

```
df.tail() #Displaying the last 5 rows of the data
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
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

Using the shape function came to know that the data comprises 180 rows and 9 columns.

```
df.shape #Displaying the shape of the data
```

```
(180, 9)
```

In order to unearth the data type of each column ran the info code first to come up with the information about data like memory storage utilized by the data seems to be 12.8+ KB. Then the dtypes code was run and it was found that 6 out of the 9 features data type was integer namely Age, Education, Usage, Fitness, Income and Miles. Whereas 3 features were objects namely Gender, Marital Status and Product.

```python
df.info() #Displaying the information of the data
```

```
<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
```

```
[ ] df.dtypes #Displaying the datatypes of the data
```

	0
Product	object
Age	int64
Gender	object
Education	int64
MaritalStatus	object
Usage	int64
Fitness	int64
Income	int64
Miles	int64

dtype: object

For detection of Missing value the isnull and isna function was run alongside a sum code and it was found that there existed no missing values in this dataset.

```
[ ] #Missing value detection
    df.isnull().sum()
```

	0
Product	0
Age	0
Gender	0
Education	0
MaritalStatus	0
Usage	0
Fitness	0
Income	0
Miles	0

dtype: int64

Lastly to check for duplicate values in the dataset the duplicated code was run and it was seen that there were no duplicates.

```
[ ] df.duplicated().sum() #Checking for duplicate values
```

```
0
```

Statistical Summary

Provided an analysis of the statistical summary in few lines for both categorical and numerical features. Using the describe function the overall statistical summary of the data frame was shown.

```
[ ]  df.describe() #Displaying the statistical summary of the data
```

However, to get a clearer picture the statistical summary for categorical and numerical features was displayed separately.

Categorical Features

```
[ ]  #Categorical
     df.describe(include='object')
```

	Product	Gender	MaritalStatus
count	180	180	180
unique	3	2	2
top	KP281	Male	Partnered
freq	80	104	107

The statistical summary for the categorical features includes the columns Product, Gender and Marital Status. All 3 have an equal row count i.e. 180. Product column has namely 3 unique treadmills: KP281, KP481, KP781. Gender has 2 unique values Male and Female. Marital Status also has 2 unique values: Single or Partnered. The top product is KP281, top gender is Make and top Marital Status is Partnered with frequencies 80, 104 and 107 respectively.

Numeric Features

```
[ ] #Numeric
    df.describe(include='number')
```

	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

The numeric features for this data set include Age, Education, Usage, Fitness, Income and Miles. Row count is the same for all. The Statistical summary shows the mean and standard deviation for all the numeric columns. The Mean and Standard deviation for Income is the highest. Minimum value for age is 18 whereas Maximum age is 50 years. Minimum education level is 12 years and the maximum 21 years. Minimum average usage of treadmills is twice a week and maximum is 7 i.e. regularl use. Fitness level rated on a 1-5 scale depicts the minimum at 1 and maximum at 5. Minimum annual income is $29562.000000 whereas maximum is $104581.000000. Minimum miles walked each week are 21.000000 while maximum are 360.000000 miles.

Non-Graphical Analysis

Value counts function was run for all categorical features. As per the code run to find out the value counts for the product features, KP281 treadmills are on top. Next in line is KP481 and then KP781. See below:

```
[ ] #Value Counts for all categorical features
df['Product'].value_counts() #Value counts for Product
```

Product	count
KP281	80
KP481	60
KP781	40

dtype: int64

For the Gender feature, Males value count is relatively higher than females for the purpose of this analysis.

```
df['Gender'].value_counts() #Value count for Gender
```

Gender	count
Male	104
Female	76

dtype: int64

For Marital Status categorical feature, the value count for Partnered marital status is higher than single.

```
[ ] df['MaritalStatus'].value_counts() #Value counts for Marital Status
```

MaritalStatus	count
Partnered	107
Single	73

dtype: int64

Using the Unique Attributes function the unique headers for all categorical features were displayed.

```
[ ] #Unique Attributes for all categorical features
    df['Product'].unique() #Unique attributes for Product
```

```
array(['KP281', 'KP481', 'KP781'], dtype=object)
```

```
[ ] df['Gender'].unique() #Unique attributes for Gender
```

```
array(['Male', 'Female'], dtype=object)
```

```
[ ] df['MaritalStatus'].unique() #Unique attributes for
```

```
array(['Single', 'Partnered'], dtype=object)
```

Graphical Analysis

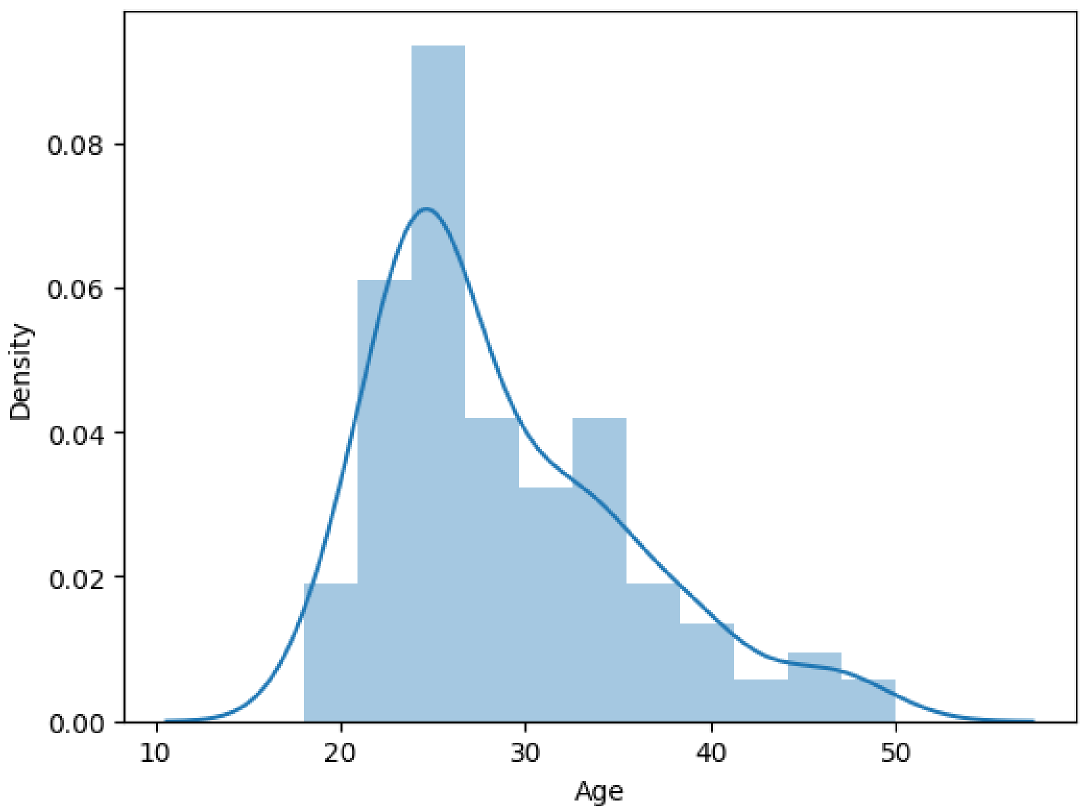
Univariate Analysis - Numerical features

Distribution Plots

Age

Using the below Sea born library code a distribution plot was created for the Age feature for the purpose of conducting univariate analysis.

```
# Univariate Analysis - Numerical features for Age,
#Distribution Plot
sns.distplot(df['Age']) #Distribution Plot for Age
```



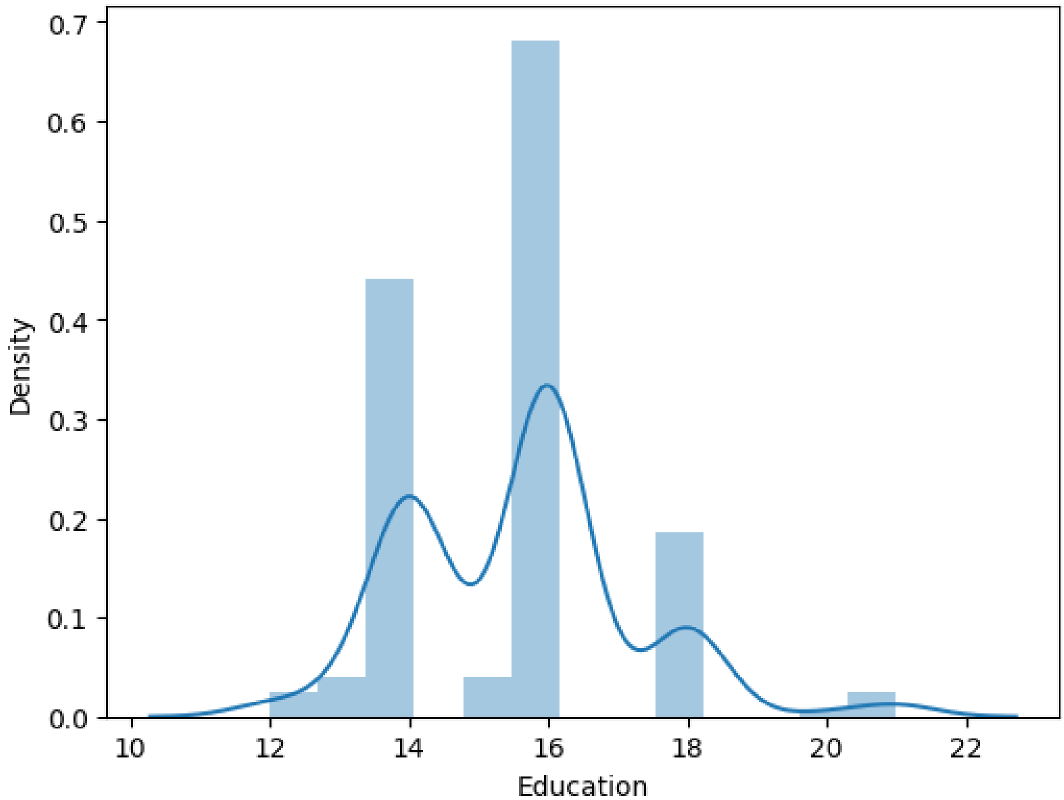
The mean age of the customers is 28.788889, with the minimum age being 18 and maximum 50.

Education

Using the below Sea born library code a distribution plot was created for the Education feature for the purpose of conducting univariate analysis.

```python
sns.distplot(df['Education']) #Distribution Plot for Education
```

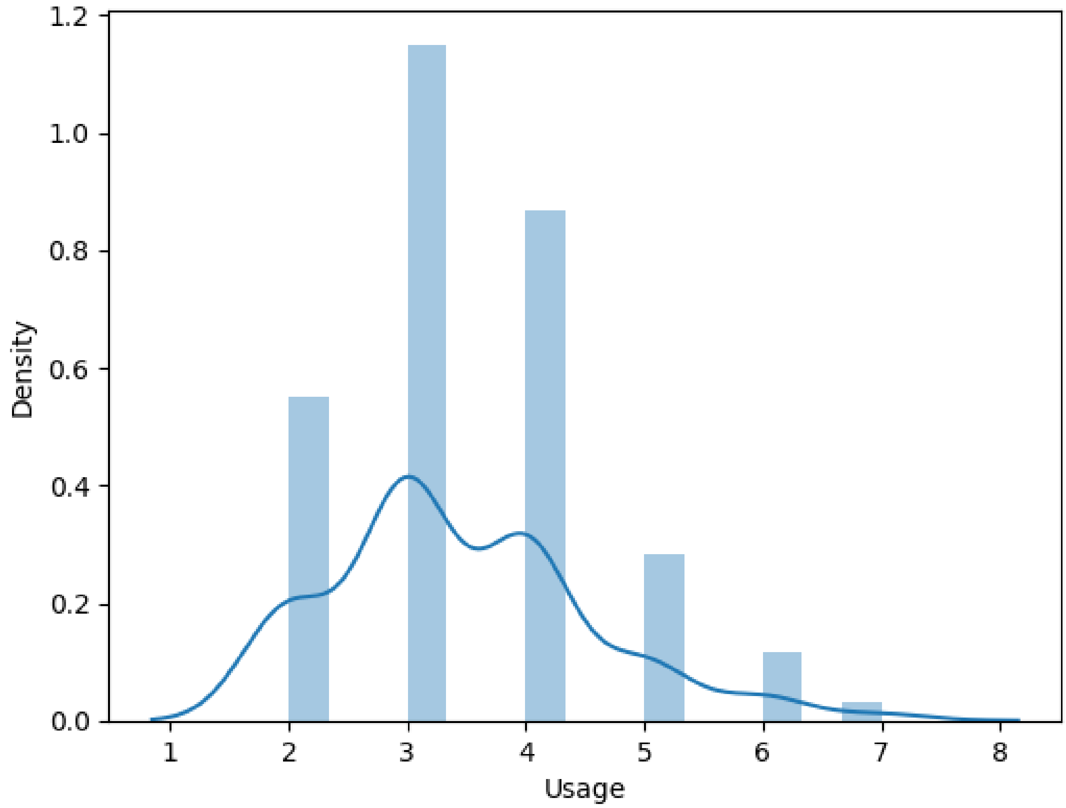
The minimum education is 12 years and maximum education is 21 years and mean level of education is 15.572222 years.



Usage

The distribution plot for usage shows that the average number of times the customer plans to use the treadmill each week is 3.455556 times with the minimum 2 and maximum 7.

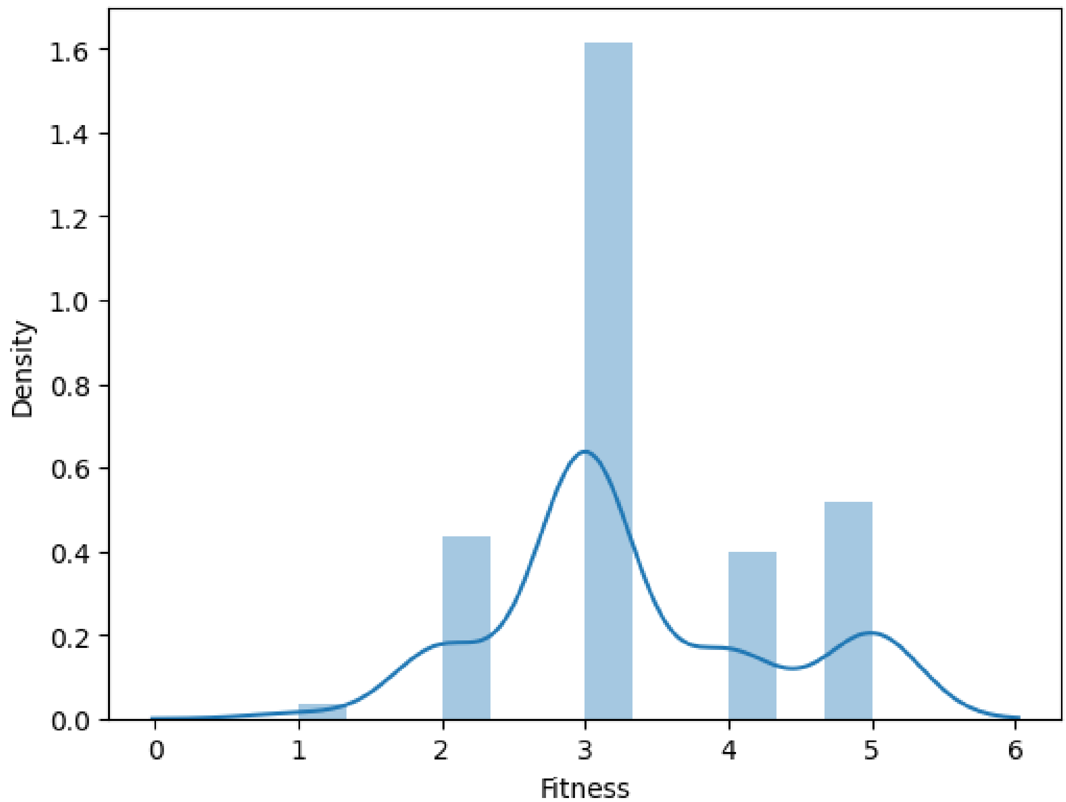
```python
sns.distplot(df['Usage']) #Distribution Plot for Usage
```



Fitness

On a self-rated fitness on a 1-5 scale, where 1 is the poor shape and 5 is the excellent shape The distribution plot for fitness shows mean fitness score is 3,31, where minimum is 1 and maximum is 5.

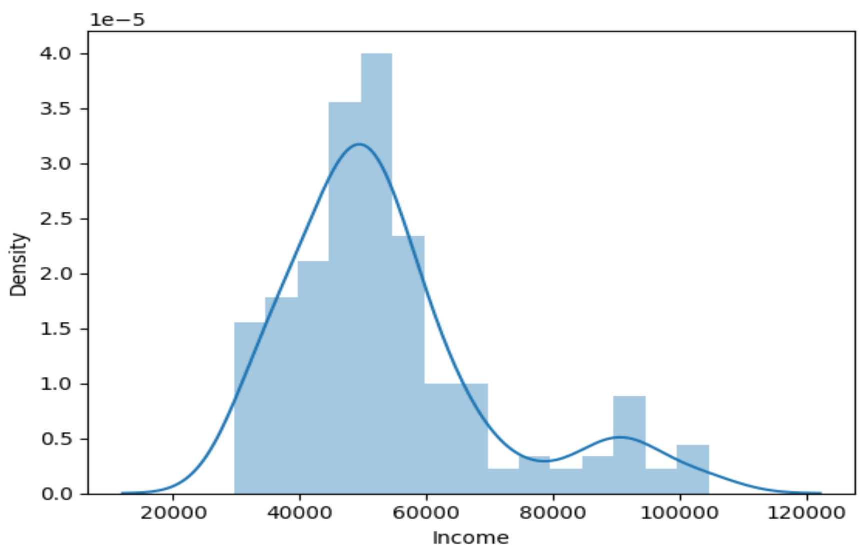
```
sns.distplot(df['Fitness']) #Distribution Plot for Fitness
```



Income

The distribution plot for income, the mean value was 53719.577778 USD, with the lowest income being 29562.000000 USD and highest income being $ 104581.000000

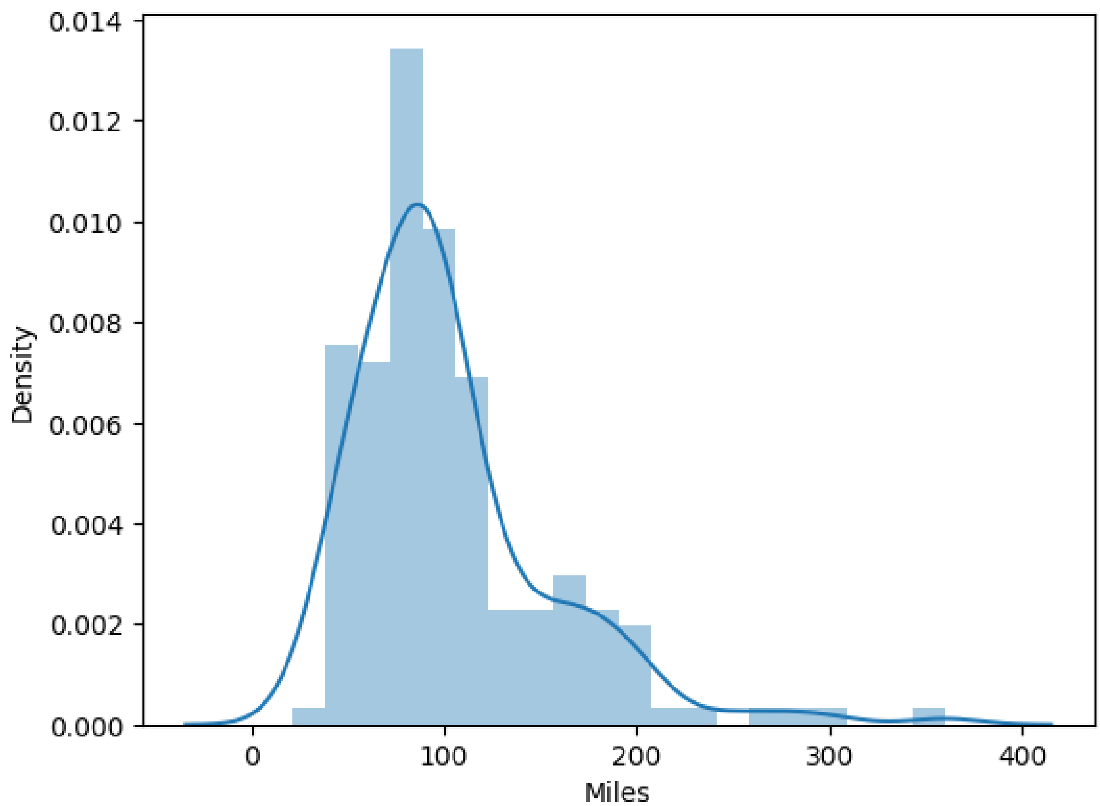
```
sns.distplot(df['Income'])  #Distribution Plot for Income
```



Miles

The distribution plot for Miles the average number of miles the customer expects to walk/run each week as per this dataset was 103.194444 miles with a minimum 21 miles and maximum 360 miles.

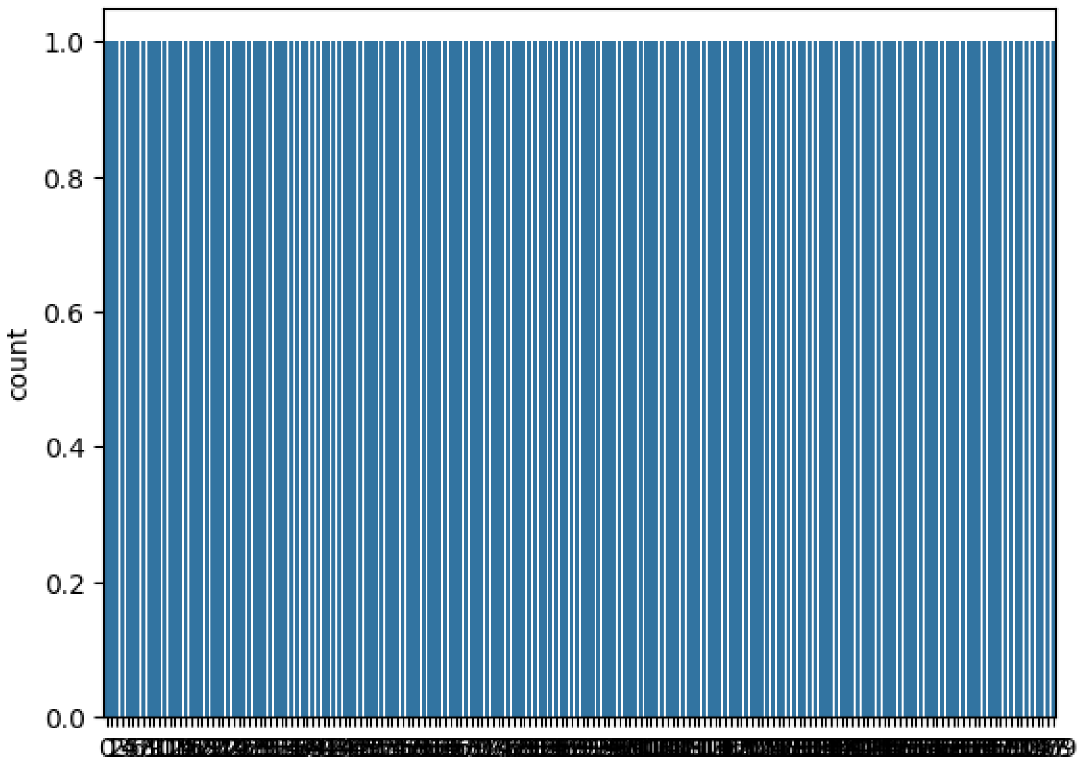
```
sns.distplot(df['Miles']) ##Distribution Plot for Miles
```



Count Plots

Using the below code, a count plot was displayed for Age feature. Avoided creating count plots for other numeric features as it is not giving any meaningful depiction of the data.

```
#Count Plot for Age
sns.countplot(df['Age'])
```

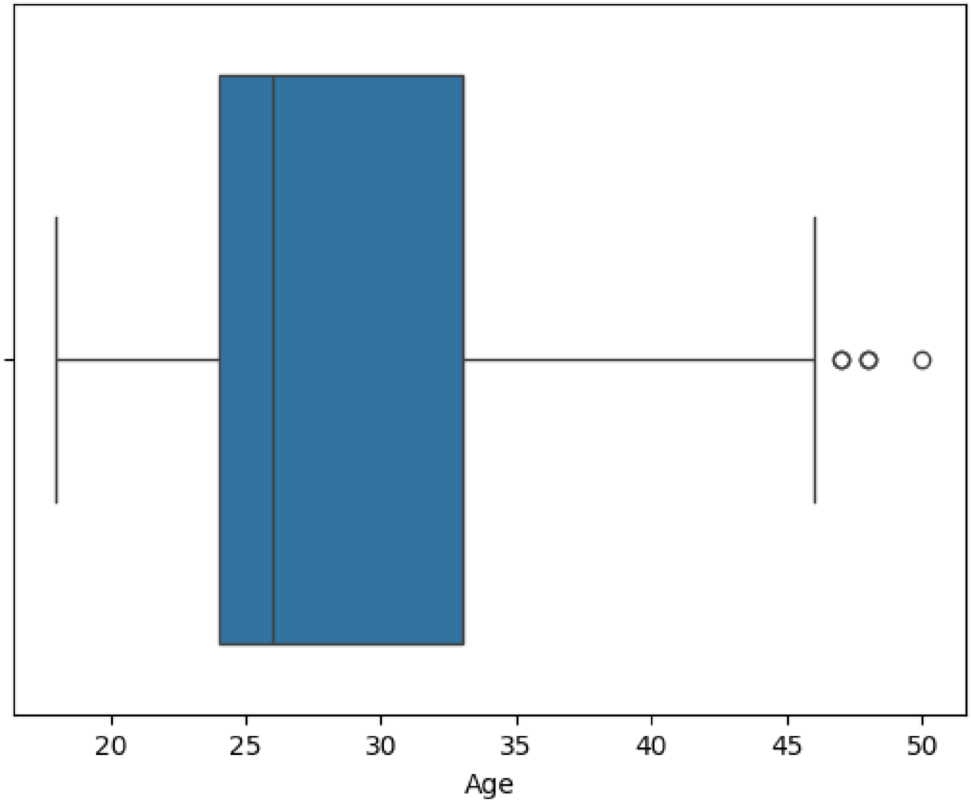


Box Plot

In descriptive statistics, a box plot or boxplot (also known as a box and whisker plot) is a type of chart often used in explanatory data analysis. Box plots visually show the distribution of numerical data and skew ness by displaying the data quartiles (or percentiles) and averages. Box plots show the five-number summary of a set of data: including the minimum score, first (lower) quartile, median, third (upper) quartile, and maximum score. Using the sea born box plot functionality created a box plot was created for the numeric features. When the median is in the middle of the box, and the whiskers are about the same on both sides of the box, then the distribution is symmetric. When the median is closer to the bottom of the box, and if the whisker is shorter on the lower end of the box, then the distribution is positively skewed (skewed right). When the median is closer to the top of the box, and if the whisker is shorter on the upper end of the box, then the distribution is negatively skewed (skewed left). An outlier is an observation that is numerically distant from the rest of the data.

Age

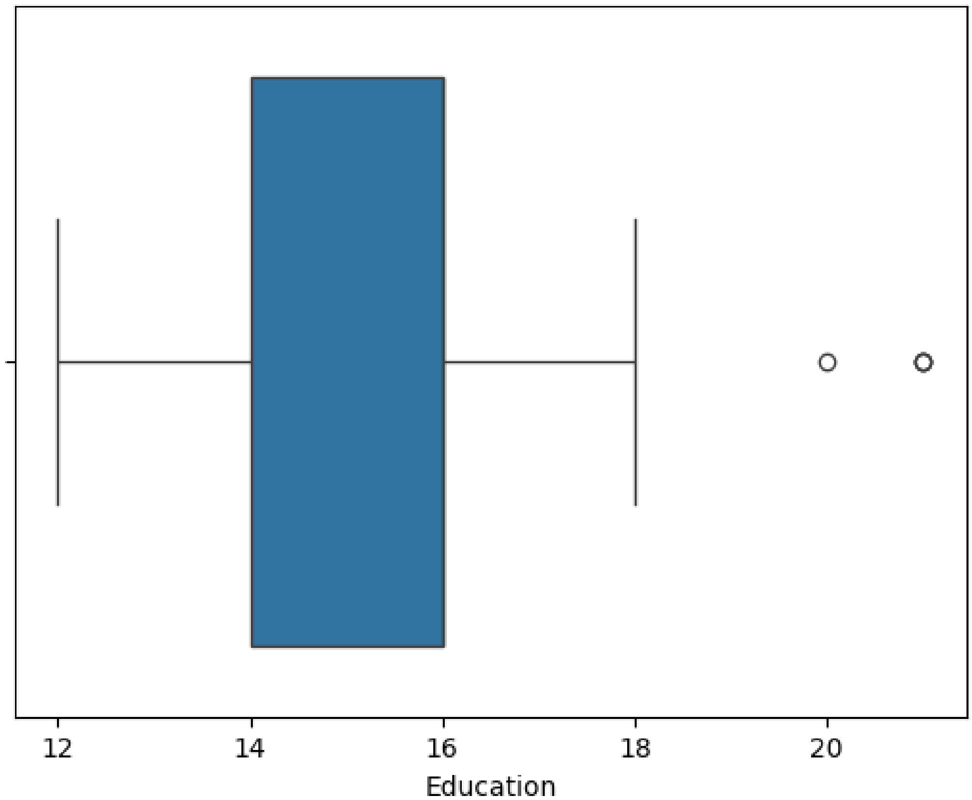
```
# Box Plot
sns.boxplot(x=df['Age']) #Box Plot for Age
```



The box plot for Age shows a positive skew and three outliers represented by the circles

Education

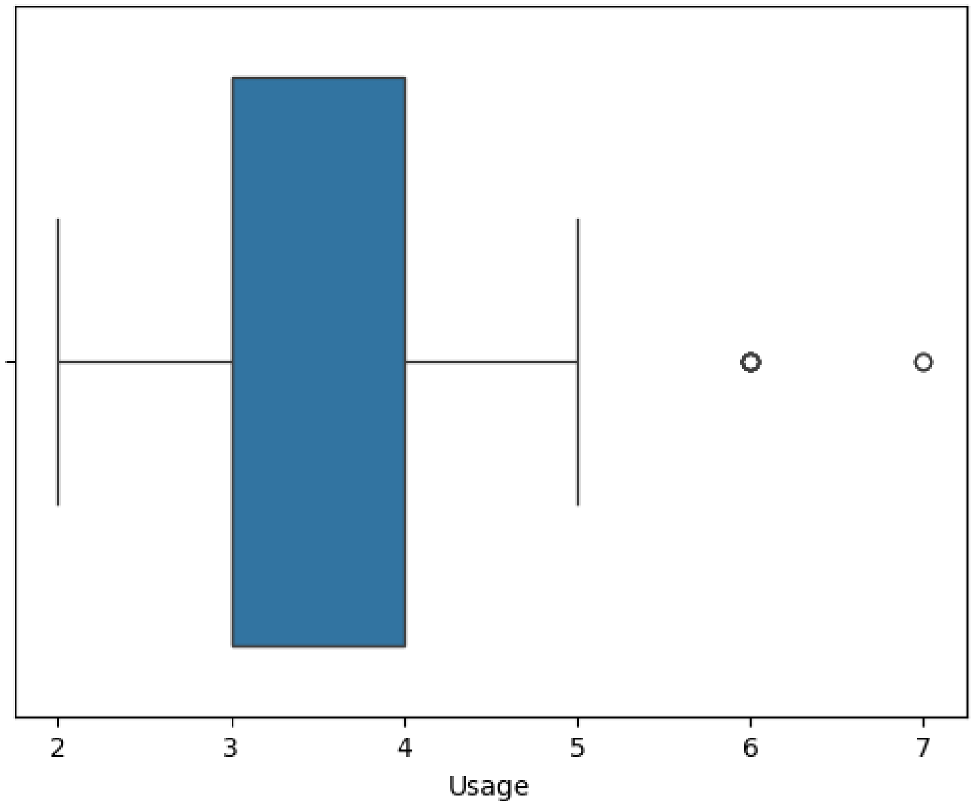
```
sns.boxplot(x=df['Education']) #Box Plot for Education
```



The data for Education is normally distributed and there seem to be only 2 outliers.

Usage

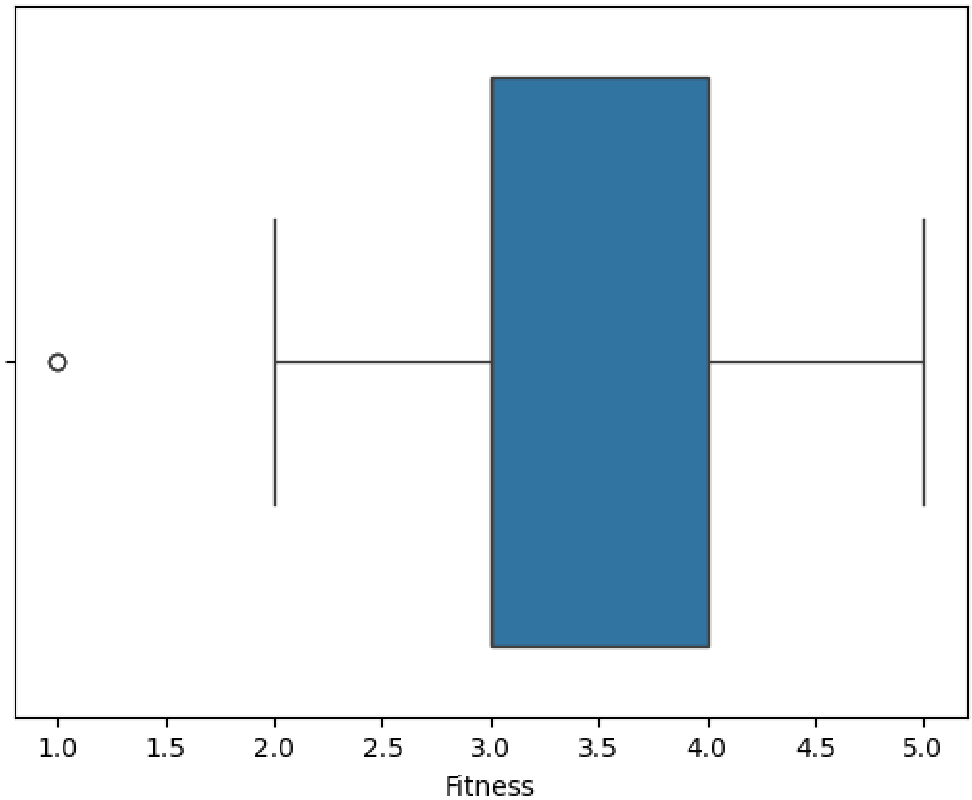
```
sns.boxplot(x=df['Usage']) #Box Plot for Usage
```



The data for Usage also seems to be normally distributed and there seem to be only 2 outliers.

Fitness Box Plot

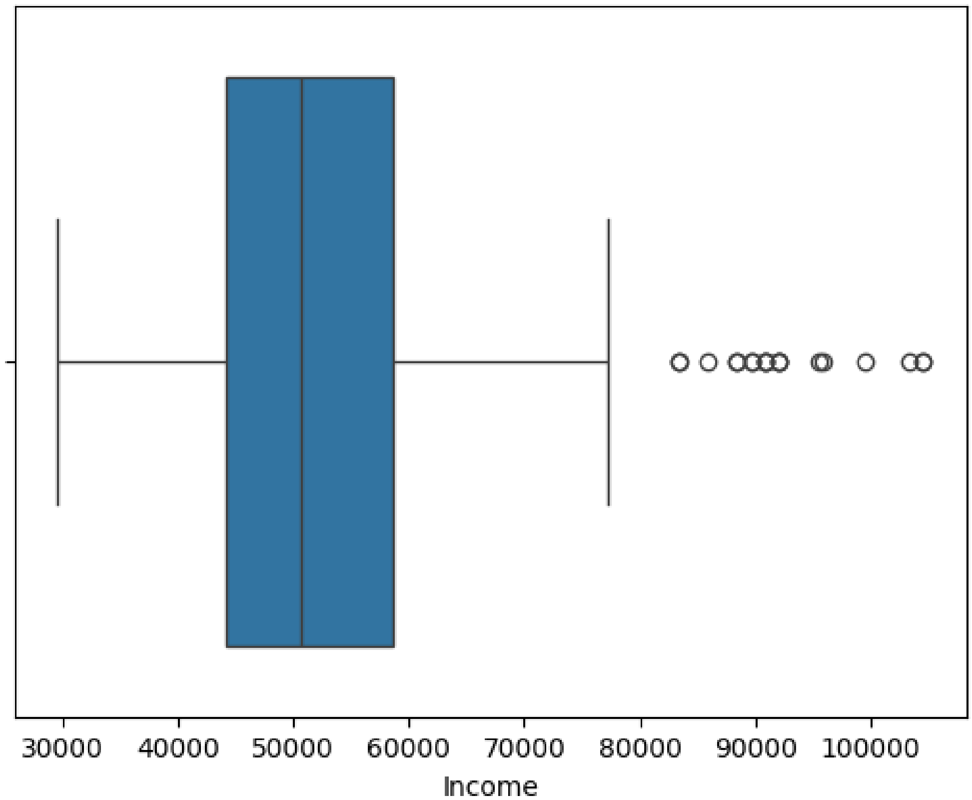
```
sns.boxplot(x=df['Fitness']) #Box Plot for Fitness
```



The data for fitness seems to be negatively skewed with one outlier.

Income Box Plot

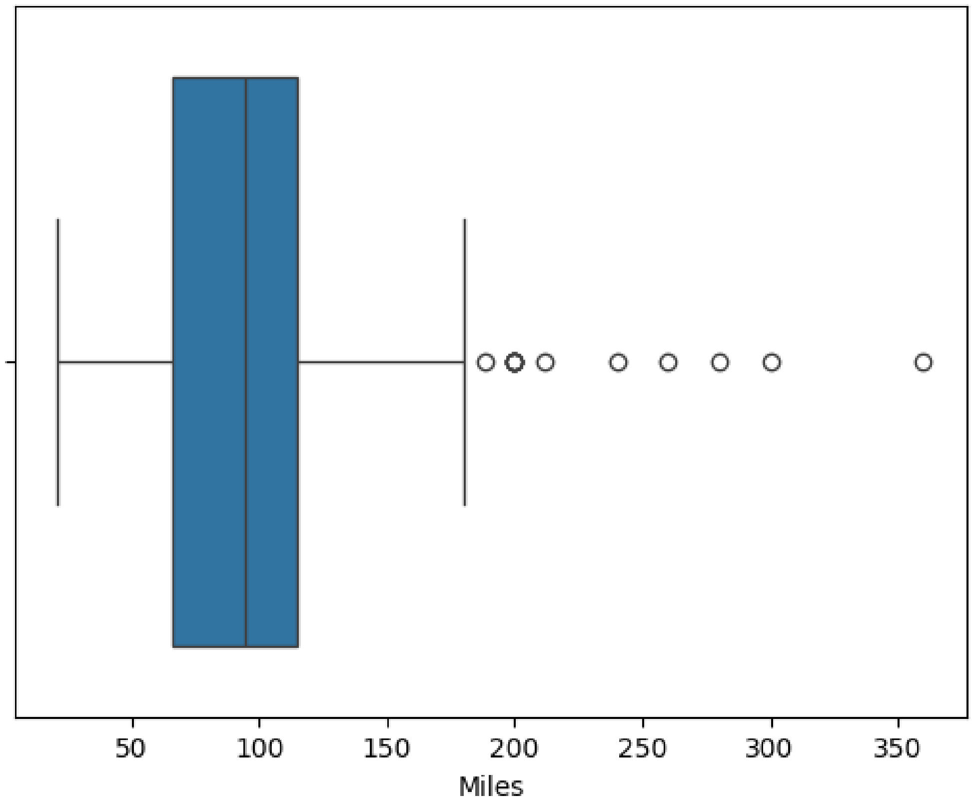
```
sns.boxplot(x=df['Income']) #Box Plot for Income
```



The data for income seems to normally distributed but with many outliers representing the high income customers purchasing power.

Miles

```
sns.boxplot(x=df['Miles']) #Box Plot for Miles
```

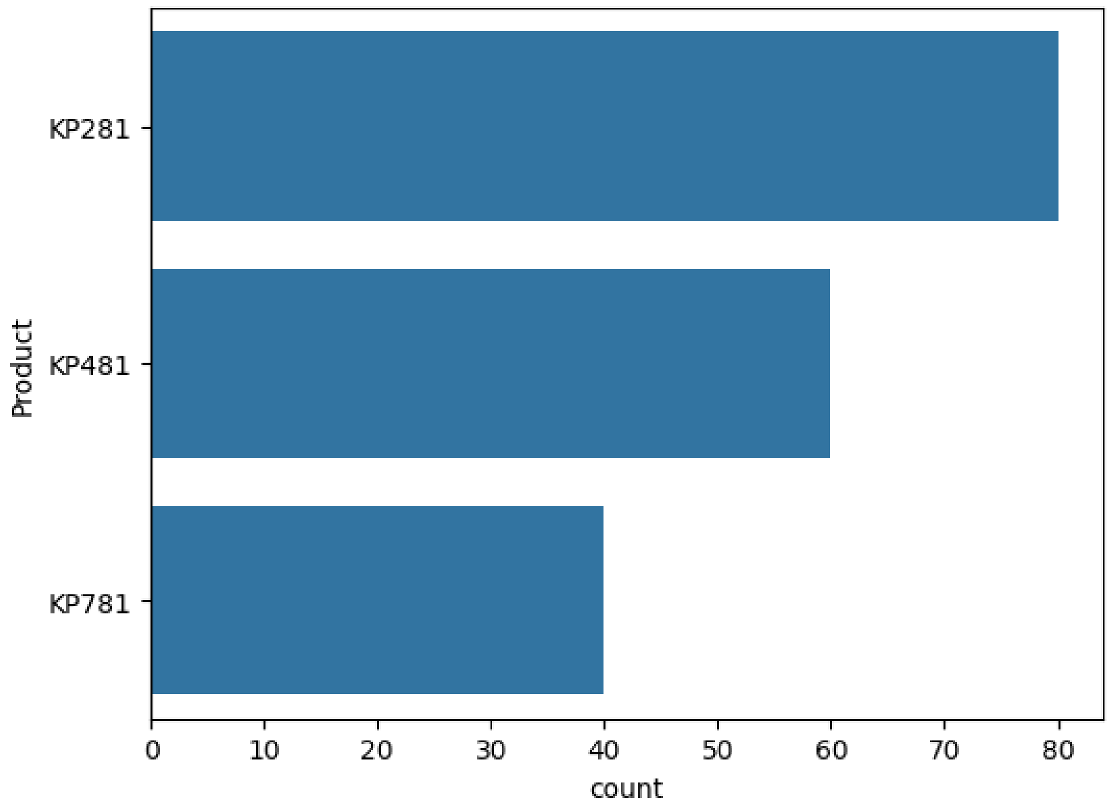


The data for miles seems to be negatively skewed with quite a few outliers

Univariate Analysis - Categorical features

Product Count Plot

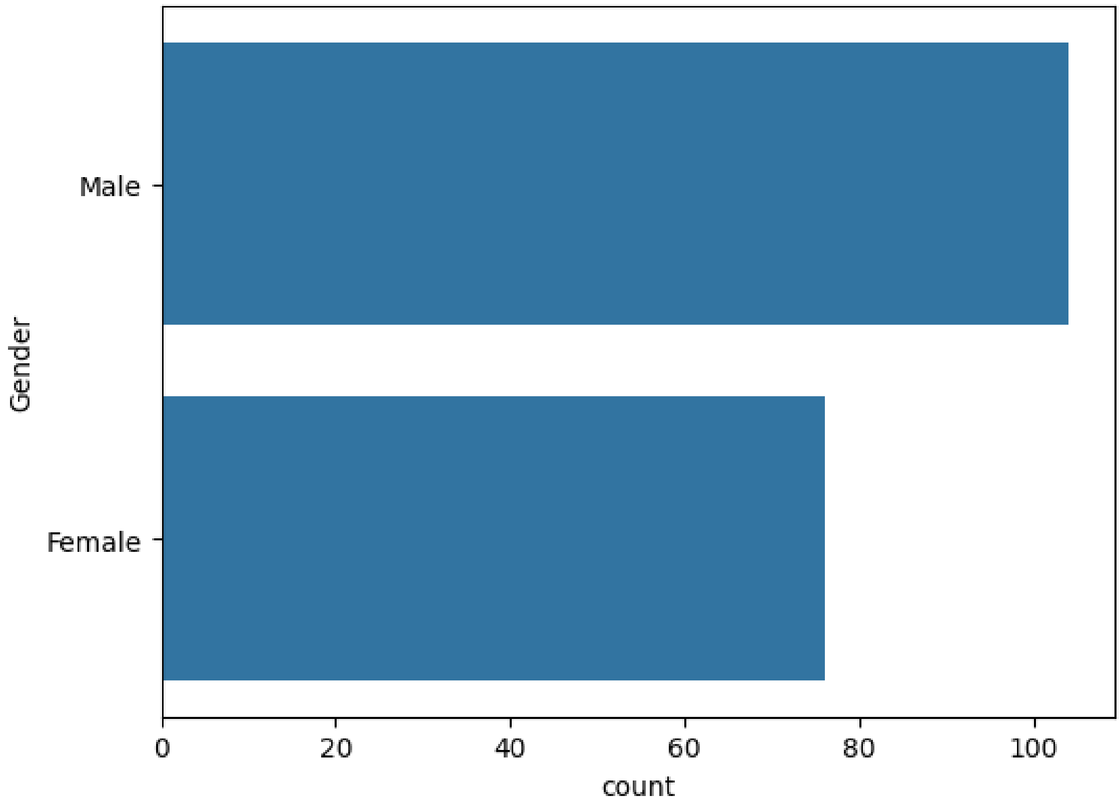
```
#Count Plot
sns.countplot(df['Product']) #Count Plot for Product
```



The count plot clearly shows that KP281 treadmills were the highest purchased treadmills around 80. Next in line were the KP481 treadmills around 60 and KP781 around 40.

Gender Count Plot

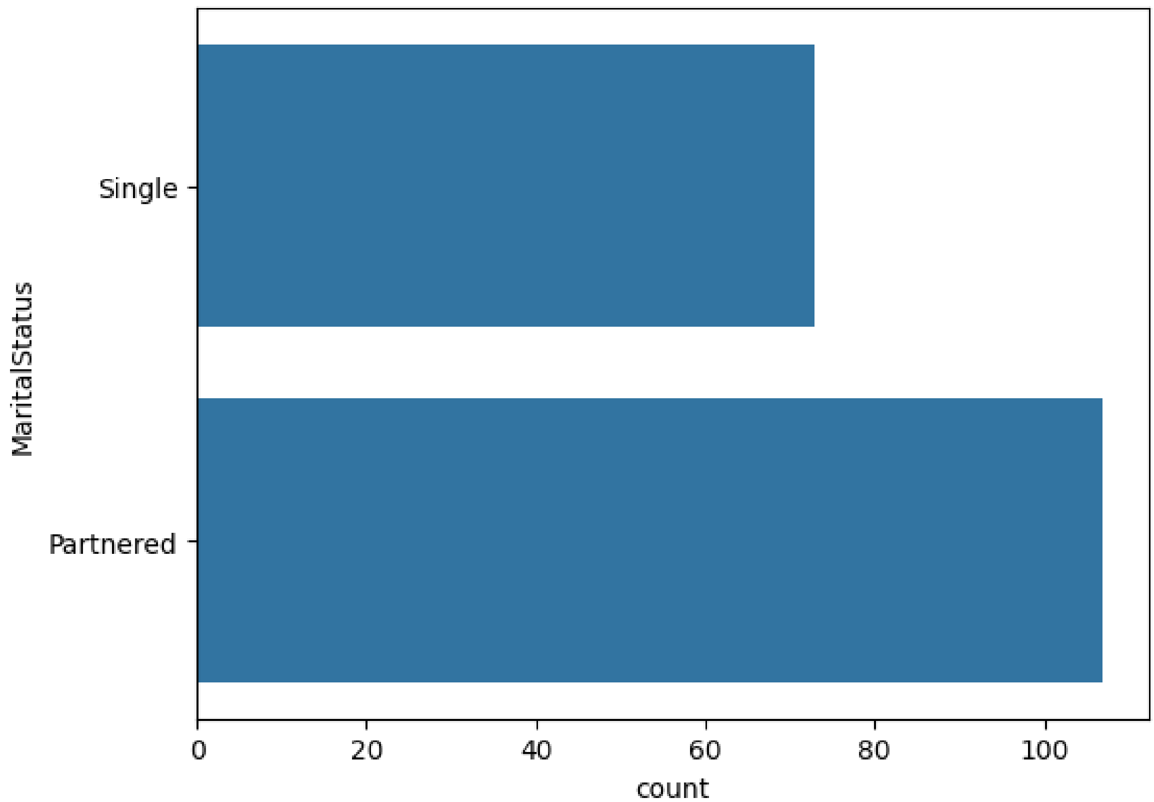
```
sns.countplot(df['Gender']) #Count Plot for Gender
```



The count plot above shows that there were 100 males and around 78 female customers purchasing the treadmills.

Marital Status Count Plot

```
sns.countplot(df['MaritalStatus']) #Count Plot for Marital Status
```



The count plot shows that amongst customers who purchased the treadmills around 73 were single and 107 were partnered.

Bivariate Analysis

In order to check features effect on the product purchased e.g.

Product vs Gender

The following code was run to come up with a kind of matrix as shown below

```
#Check features effect on the product purchased e.g.  Product vs Gender
pd.crosstab(df['Product'],df['Gender']) #Check
```

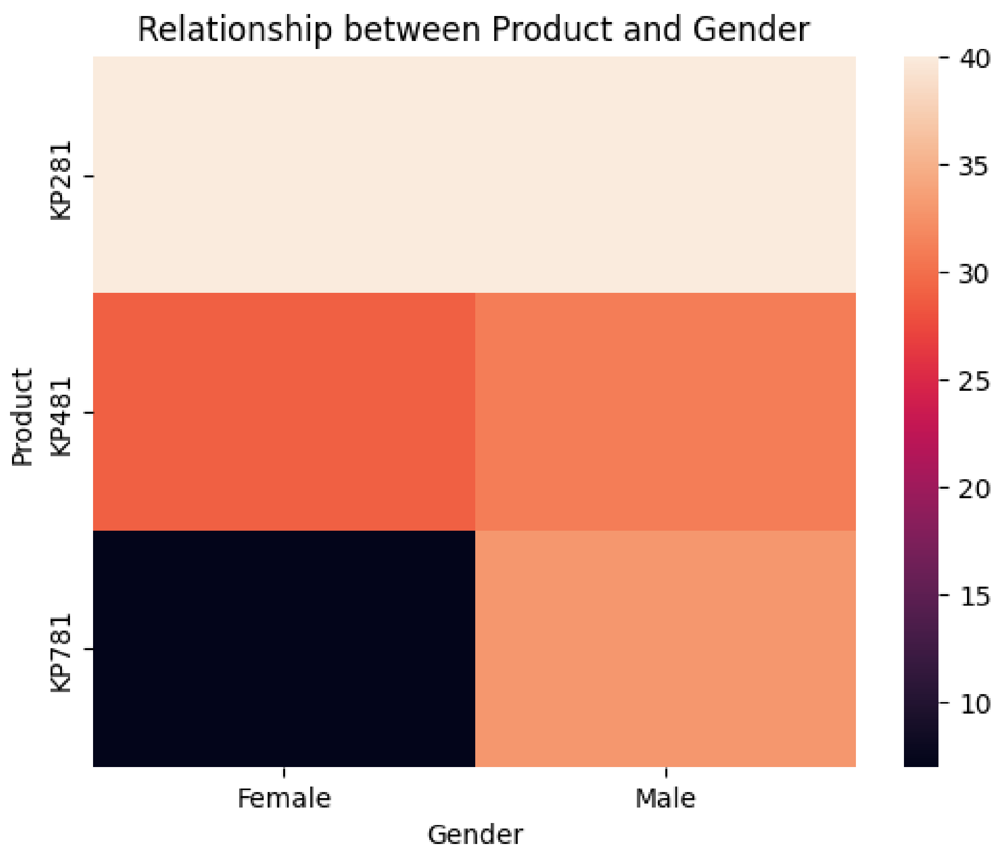
Gender	Female	Male
Product		
KP281	40	40
KP481	29	31
KP781	7	33

The above matrix shows that for KP281 treadmills 40 customers were female and 40 were Males for the time period considered for the purpose of this analysis. Similarly for KP481 treadmills 29 were females and 31 were males. Lastly, for KP781 treadmills 7 were females and 33 were males.

A heat map was generated to second the findings above using the below code.

```
#Heat Map for the above Product Vs. Gender
sns.heatmap(pd.crosstab(df['Product'],df['Gender']))

plt.title("Relationship between Product and Gender")
#
```



The light color gradient clearly shows that the gender distribution was equal for KP281. Similarly, a darker shade for females seems to depict lesser females than males for KP481. Lastly, black color gradient for females shows a huge contrast between the number of males purchasing KP781 and number of females where the dark shade denotes females.

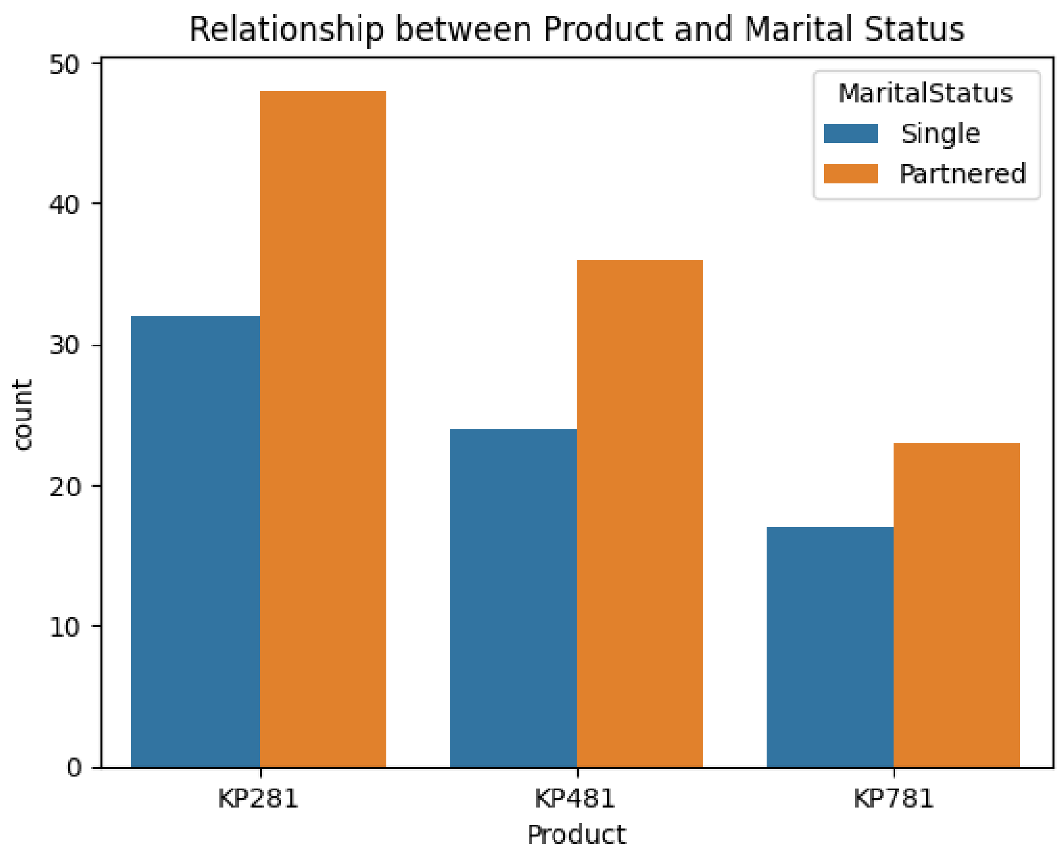
Product vs. Marital Status

A similar code was run to unearth Product vs. Marital Status effect on the product purchased. A similar matrix was created.

```
#Checking features effect on the product purchased e.g. Product vs MaritalStatus
pd.crosstab(df['Product'],df['MaritalStatus'])
```

MaritalStatus	Partnered	Single
Product		
KP281	48	32
KP481	36	24
KP781	23	17

The above matrix shoes a bifurcation for product by marital status. For KP281 48 customers were partnered whereas 32 were single. For KP481 36 were Partnered whereas 24 were single. Lastly, for KP781 23 were partnered and 17 were single. The below graph provides a visual depiction of the same.



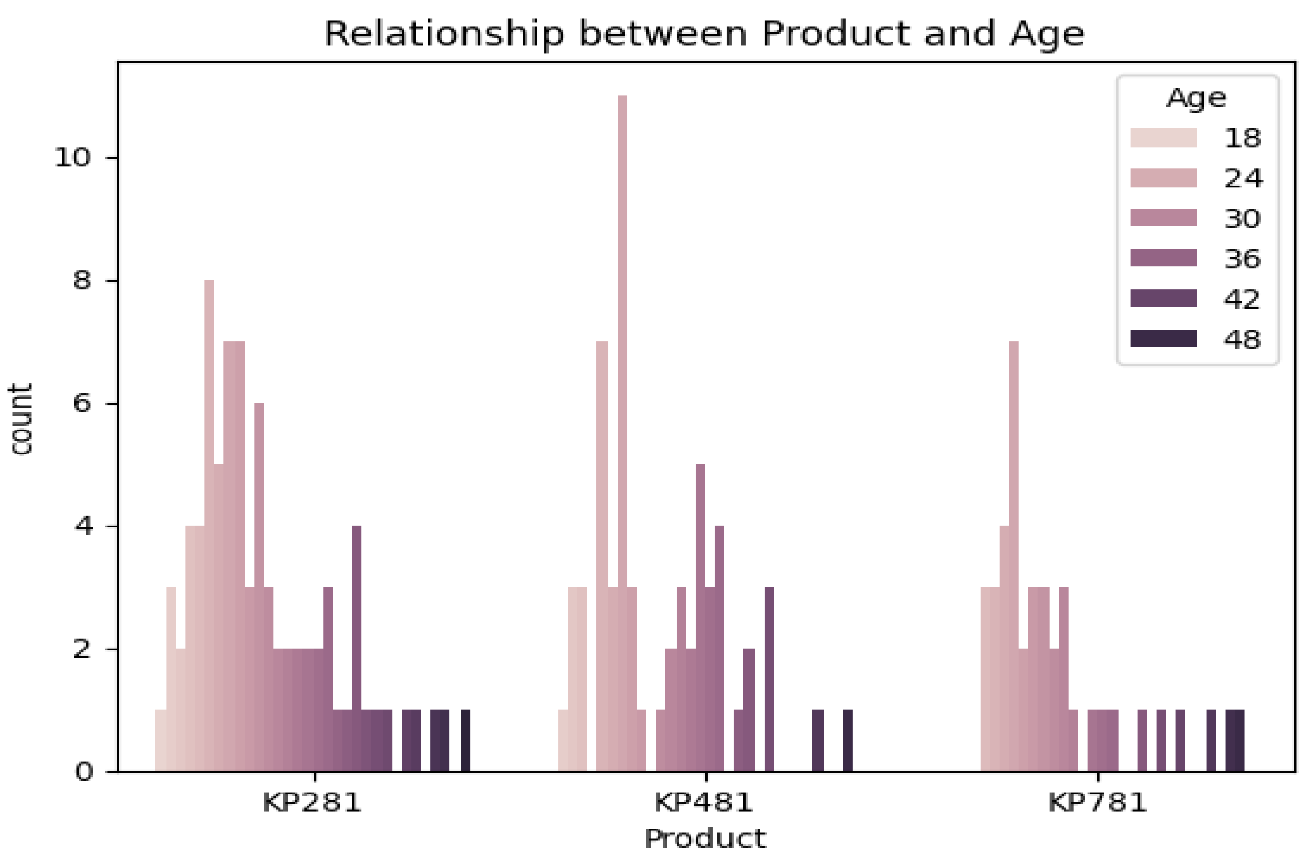
So, overall partnered individuals seem to have a higher tendency to purchase treadmills.

Product vs. Age

The below code was run to create a matrix for checking effect of Product and Age on purchasing patterns of the treadmills.

```
#Checking features affect on products purchase for e.g. Product vs Age
pd.crosstab(df['Product'],df['Age'])
```

Age Product	18	19	20	21	22	23	24	25	26	27	...	40	41	42	43	44	45	46	47	48	50
KP281	1	3	2	4	4	8	5	7	7	3	...	1	1	0	1	1	0	1	1	0	1
KP481	0	1	3	3	0	7	3	11	3	1	...	3	0	0	0	0	1	0	0	1	0
KP781	0	0	0	0	3	3	4	7	2	3	...	1	0	1	0	0	1	0	1	1	0

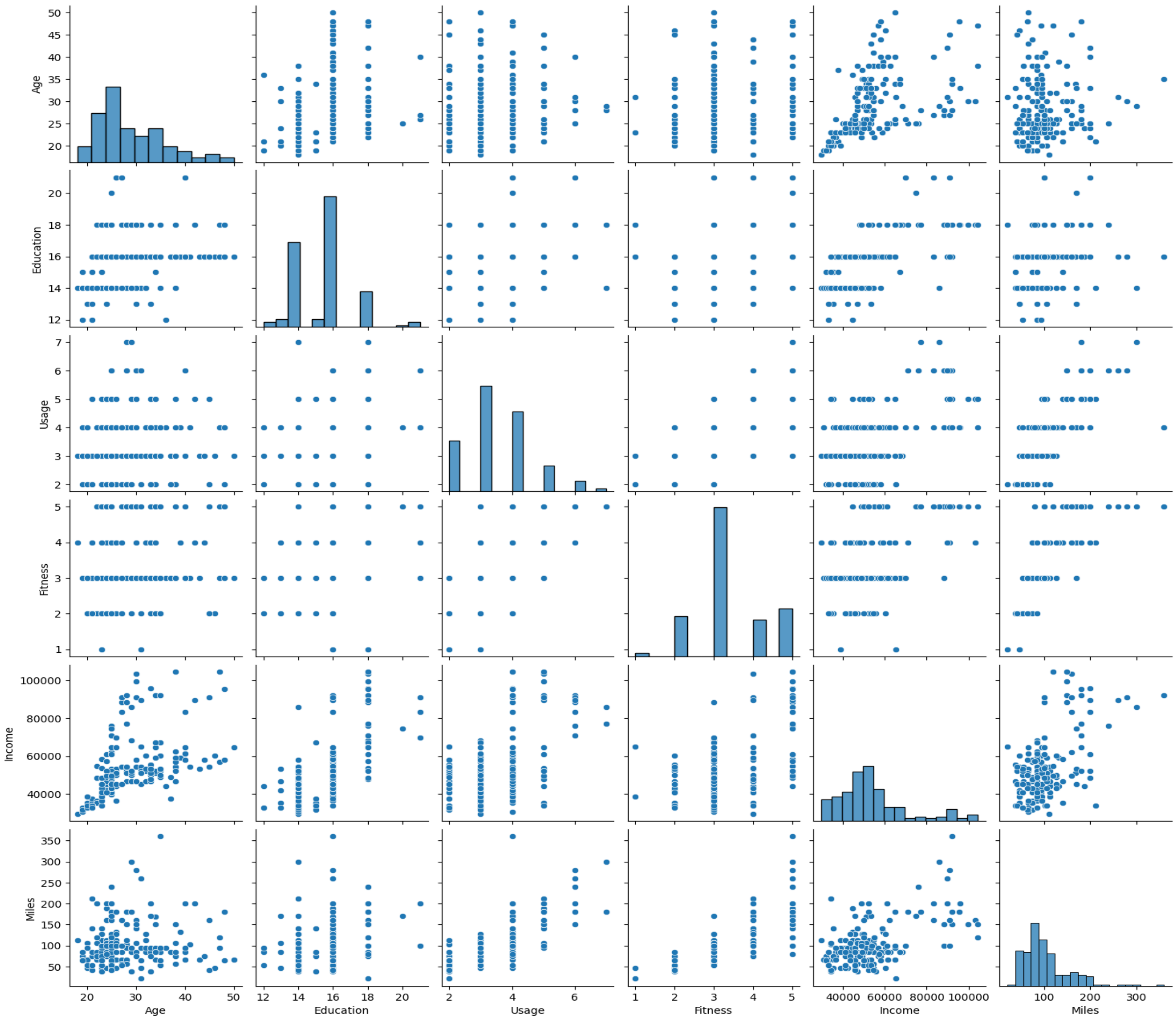


It can be seen that KP281 has the highest concentration of all age groups represented by the closed cluster and different shades of the purple color gradient representing dark shades for bigger ages and lighters shades for younger customers. KP481 has a few gaps having relatively more customers belonging to specific age groups. Similar is the case with KP781 with gaps representing relative lower number of customers due to the price of the treadmill and one thick cluster representing concentration of younger and richer individuals more conscious about their fitness.

Multivariate Analysis

Using the below code, pair plots were created to show comprehensive view of relationship between features. See below.

```
#Pair Plot
sns.pairplot(df)
```



All the numeric functions were placed on both x and y axis and different visuals provided a depiction of the different relationships that exist within this dataset.

Correlation Analysis

Using the below code, a correlation matrix was created to show the correlation on a heat map. The gradient scale for green was used to denote the intensity of relationship represented by the correlation coefficient where lighter green denoted lower and darker shade of green represented more intensity.

Since correlation is a statistical measure that shows the direction of relationship between variables, it was observed that all the features seem to be positively correlated. There is a weak correlation b/w Education and Age features. There is a moderate correlation b/w Usage and Education. There is a strong correlation b/w Fitness and Usage. There seems to be a strong correlation b/w income and fitness. Similarly for Miles and Income. There is a weak correlation b/w usage and age & b/w Fitness and Age. Lastly, there is a very strong relationship in b/w miles and usage & mules and fitness. Benchmarked against the following interpretations: https://www.scribbr.com/statistics/correlation-coefficient/

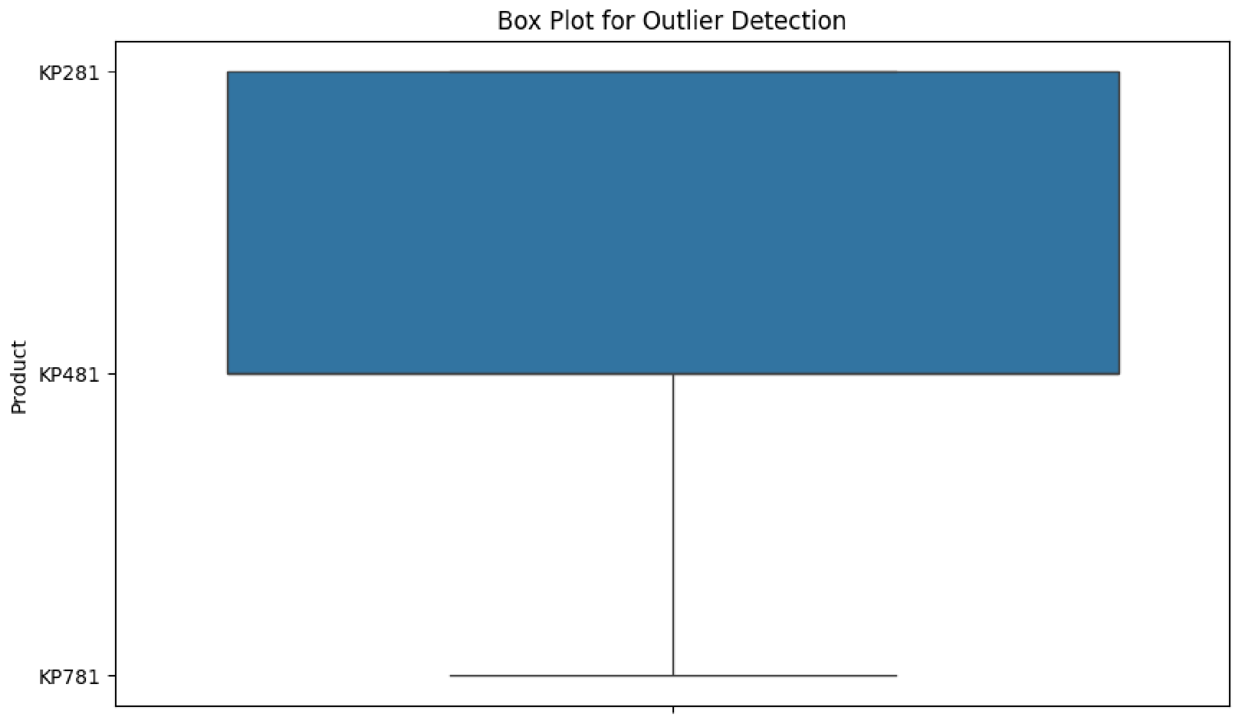
```
#Show the correlation matrix on heatmap
sns.heatmap(df.corr(numeric_only=True),annot=True,cmap="crest")
```

Outlier Detection

The below code was run to check for the outliers by using the IQR method.

Outlier box plot for Products

```
#Check for the outliers by using the IQR method.
# Box plot for outlier detection
plt.figure(figsize=(10, 6))
sns.boxplot(data=df['Product'])  # Replace 'column_name' with your specific column
plt.title('Box Plot for Outlier Detection')
plt.show()
```

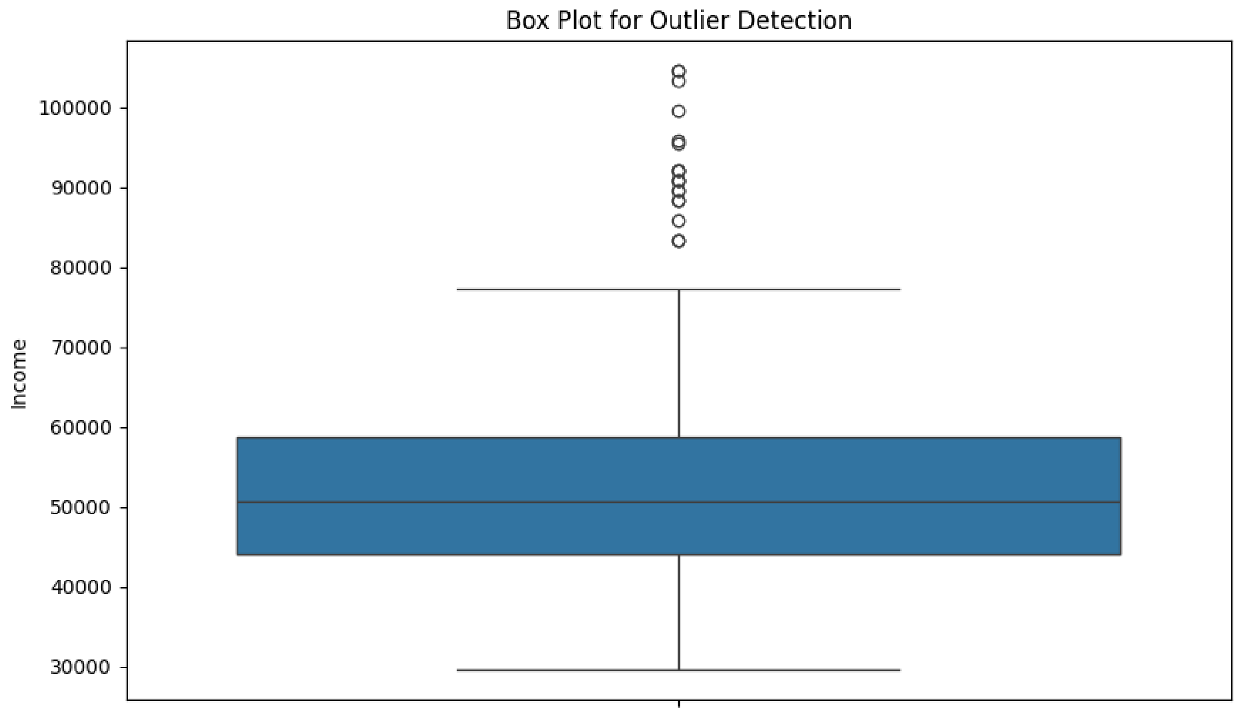


It can clearly be seen from the above box plot depiction that KP281 turned out to be the outlier in terms of the best performing treadmill product in terms of sales.

Outlier Analysis for Income

```
#Outlier for Income
Q1 = df['Income'].quantile(0.25)
Q3 = df['Income'].quantile(0.75)
IQR = Q3 - Q1
IQR
```

```
14609.25
```

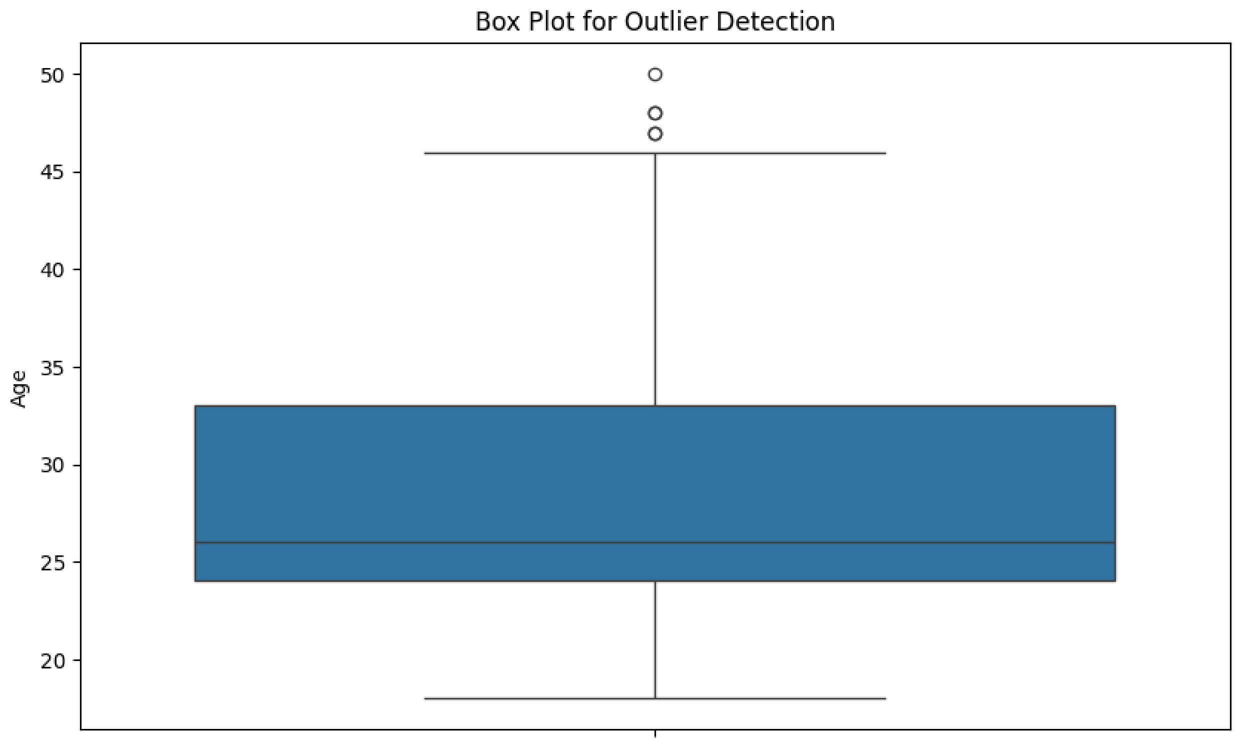


For the income feature, $14609.25 income was the outlier.

Outlier analysis for Age

```
#Outlier for Age
Q1 = df['Age'].quantile(0.25)
Q3 = df['Age'].quantile(0.75)
IQR = Q3 - Q1
IQR
```

```
9.0
```

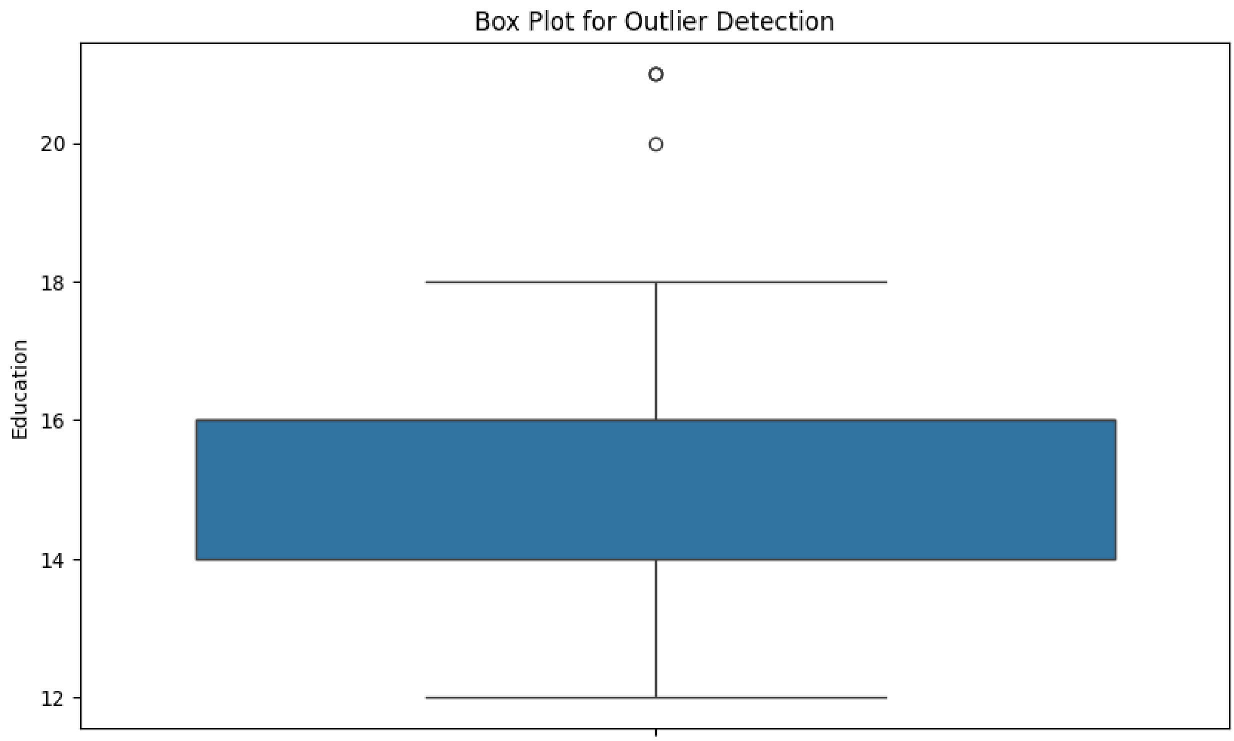


For Age feature, 9 years customer was the outlier.

Outlier for Education

```
#Outlier for Education
Q1 = df['Education'].quantile(0.25)
Q3 = df['Education'].quantile(0.75)
IQR = Q3 - Q1
IQR
```

```
2.0
```

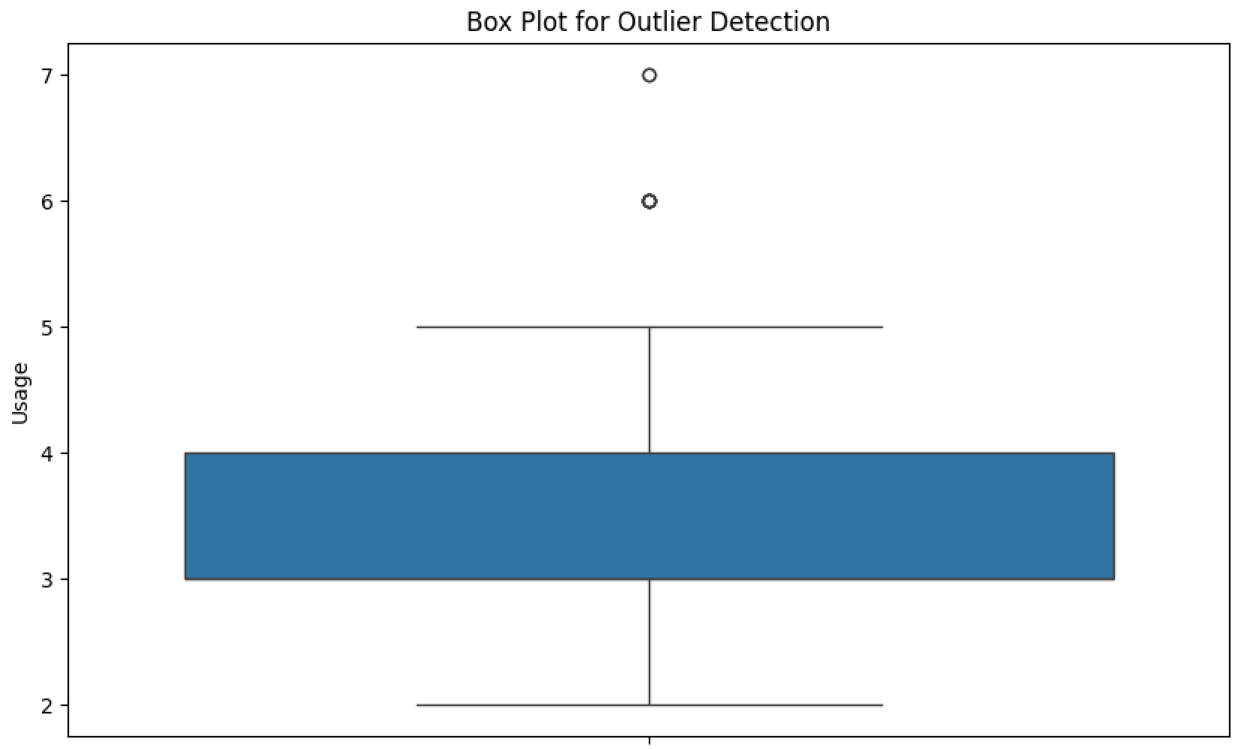


A customer with 2 years of education was the exception or outlier in terms of the education feature.

Outlier for Usage

```
#Outlier for Usage
Q1 = df['Usage'].quantile(0.25)
Q3 = df['Usage'].quantile(0.75)
IQR = Q3 - Q1
IQR
```

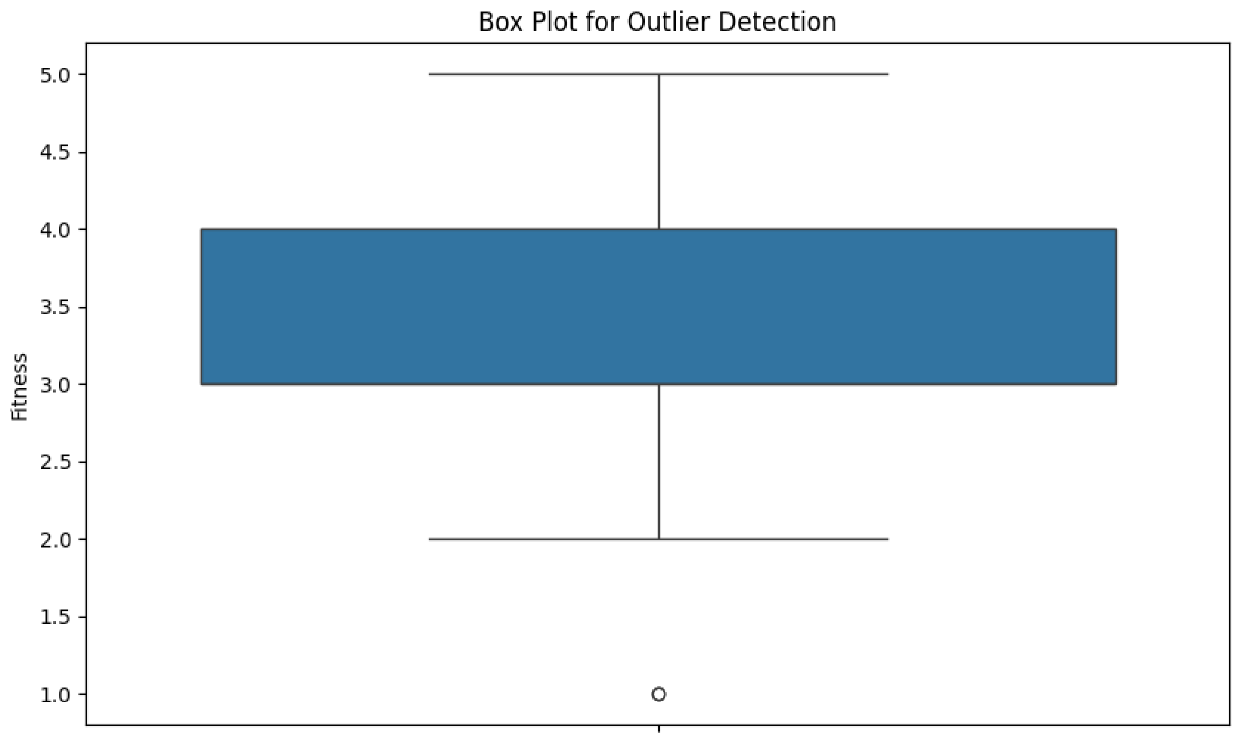
```
1.0
```



The customer who used the treadmill only once a week was the outlier in terms of usage.

Outlier for Fitness

```
#Outlier for Fitness
Q1 = df['Fitness'].quantile(0.25)
Q3 = df['Fitness'].quantile(0.75)
IQR = Q3 - Q1
IQR
#
```

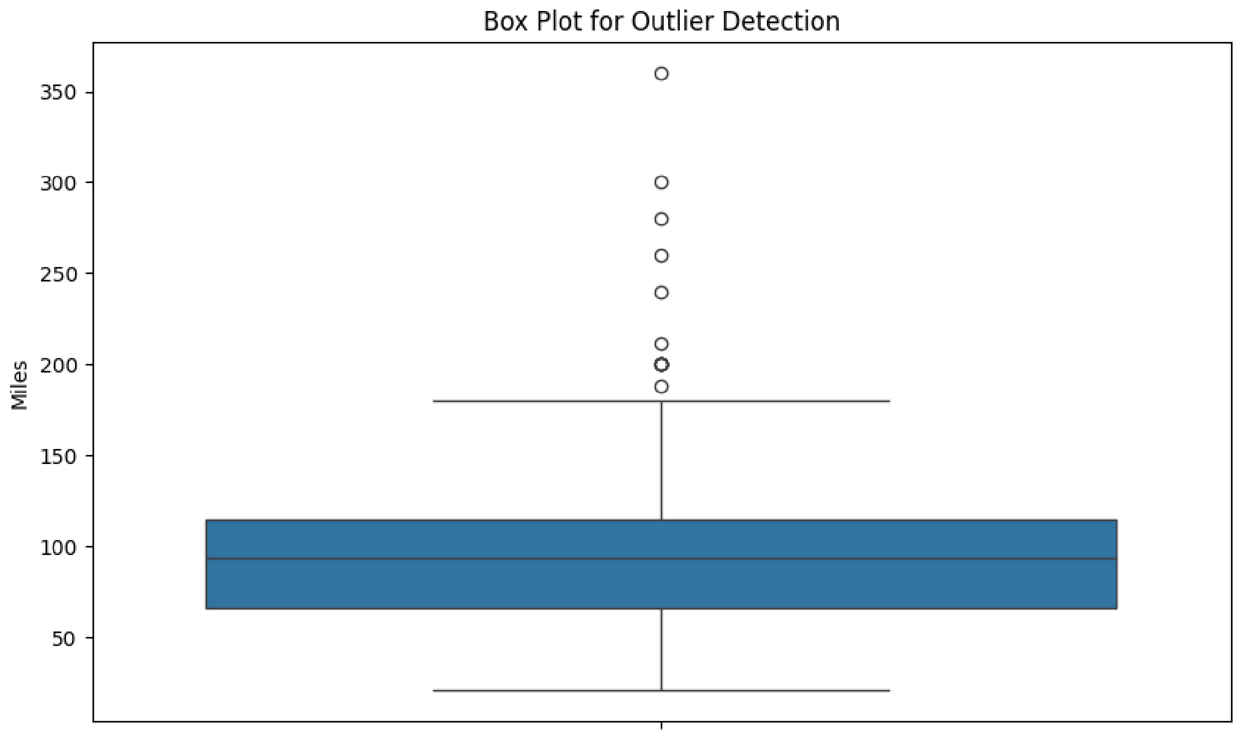


A customer who self-rated their fitness level to be in poor shape even after using the treadmill was only 1.

Outlier for Miles

```
#Outlier for Miles
Q1 = df['Miles'].quantile(0.25)
Q3 = df['Miles'].quantile(0.75)
IQR = Q3 - Q1
IQR
```

```
48.75
```



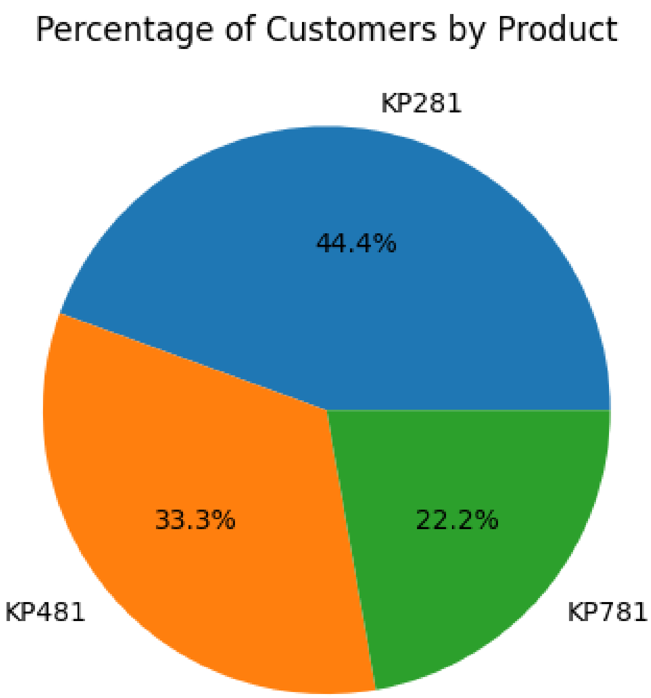
The customer who suggested that they walk/ran only 48.75 miles was the outlier in terms of the miles ran/walked on the treadmills.

Conditional Probabilities

Using the code below, it was found that 44% of the customers have purchased KP281, 33% have purchased KP481 and 22% have purchased KP781. Same is depicted by the pie chart.

```
percentage_counts = df['Product'].value_counts(normalize=True) * 100
print(percentage_counts)
```

```
Product
KP281    44.444444
KP481    33.333333
KP781    22.222222
Name: proportion, dtype: float64
```



Product – Gender

Using the below code a frequency table was created for Product vs. Gender.

```
[15] # Create frequency tables and calculate the percentage as follows Product - Gender
     #Percentage of a Male customer purchasing a treadmill
     frequency_table = pd.crosstab(df['Product'], df['Gender'])
     print(frequency_table)
```

```
Gender   Female  Male
Product
KP281        40    40
KP481        29    31
KP781         7    33
```

By running the code below it was found that the percentage of a male customer purchasing a KP281 treadmill was 50%, KP481 treadmill was 51.67% and KP781 treadmill was 82.50%.

Percentage of a Male customer purchasing a treadmill

```
percentage_male = (frequency_table['Male'] / frequency_table.sum(axis=1)) * 100
print(percentage_male)
```

```
Product
KP281     50.000000
KP481     51.666667
KP781     82.500000
dtype: float64
```

By running the below code it was found that the percentage of a Female customer purchasing KP781 treadmill was 100%

```
 # Percentage of a Female customer purchasing KP781 treadmill
frequency_table.loc['KP781', 'Female'] / frequency_table.loc['KP781'] * 100
```

Gender	KP781
Female	100.000000
Male	21.212121

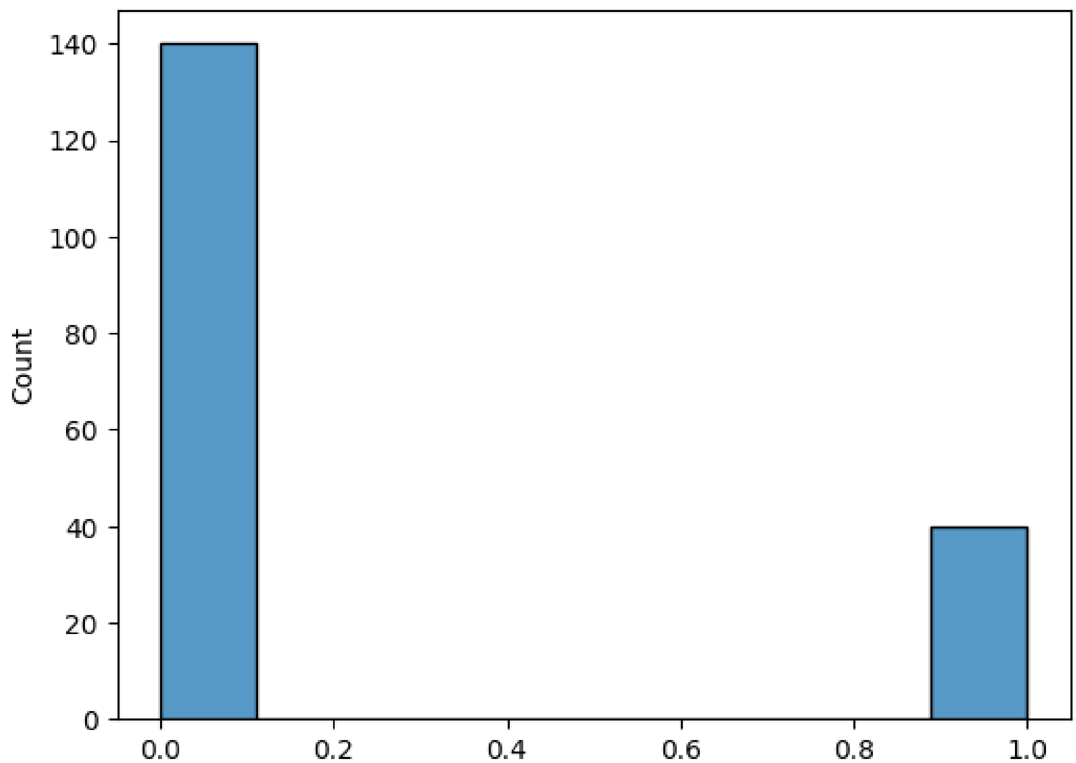
dtype: float64

It was also found that the probability of a customer being a Female given that Product is KP281 was 1. Same is depicted by the histo plot.

```
#Probability of a customer being a Female given that Product is KP281
frequency_table.loc['KP281', 'Female'] / frequency_table.loc['KP281']
```

Gender	KP281
Female	1.0
Male	1.0

dtype: float64



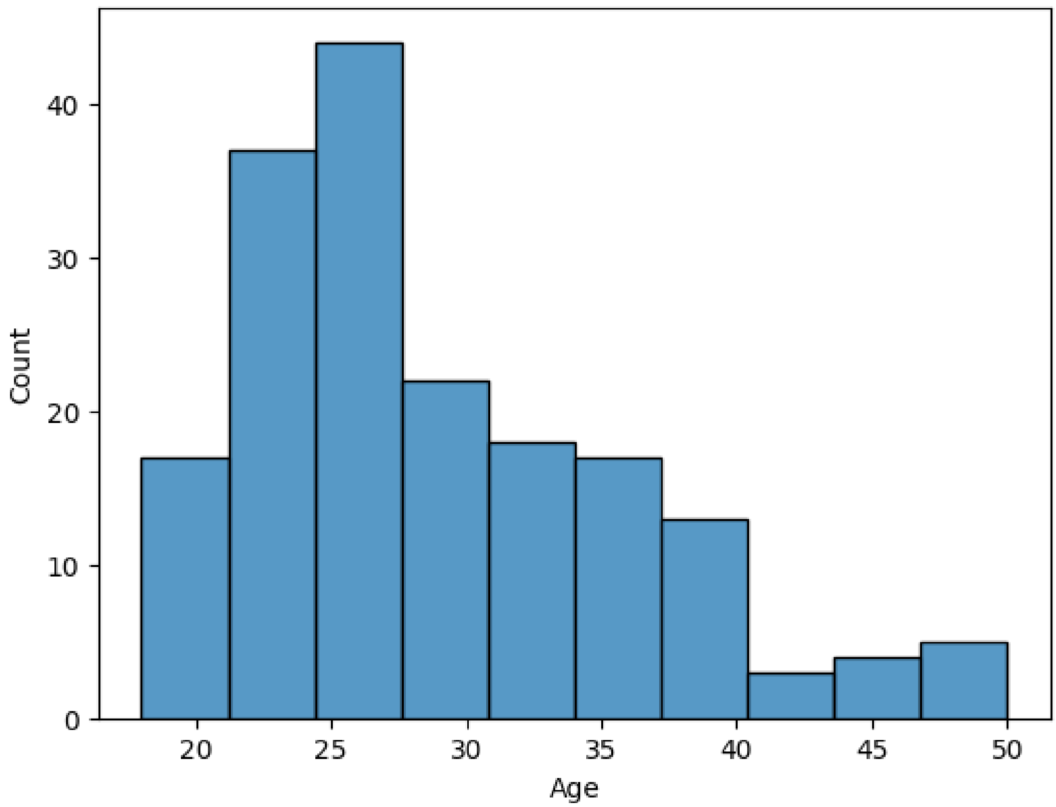
Product – Age

To find out the Percentage of customers with Age between 20s and 30s among all customers the following code was run.

```
#Percentage of customers with Age between 20s and 30s among all customers
age_counts = df['Age'].value_counts(normalize=True) *100
age_counts
```

	proportion
Age	
25	13.888889
23	10.000000
24	6.666667
26	6.666667
28	5.000000
35	4.444444
33	4.444444
30	3.888889

These seem to be around 51% of customers between ages 20’s to 30’s.



Product – Income

In order to unearth the percentage of a low-income customer purchasing a treadmill first of all a describe code (df['Income'].describe()) was run on the income feature and based on the income patterns results, 3 categories of High, Medium and Low Income were created for purpose of analysis. Each category having equal gap of $40000 since minimum salary as per the table below was $29562 mean/average salary was $53719.57 and max salary was 104581.000000.

Low income was defined as: Low_Income = df[df['Income'] < 40000]

Medium income was defined as: _Income = df[(df['Income'] >= 40000) & (df['Income'] <= 80000)]

High income was defined as: High_Income = df[df['Income'] > 80000]

	Income
count	180.000000
mean	53719.577778
std	16506.684226
min	29562.000000
25%	44058.750000
50%	50596.500000
75%	58668.000000
max	104581.000000

dtype: float64

Having defined the categories, there were a series of code run. The first step was to create a frequency table showing the bifurcation of product by income running the below code. Before calculating the percentage of a low-income customer purchasing a treadmill next step was to calculate the total purchases of each treadmill and then defining after calculating low income purchases of each product to arrive at a formula to calculate the %age of low income customers purchasing each treadmill.

STEP 1

```
#Percentage of a low-income customer purchasing a treadmill

frequency_table = pd.crosstab(df['Product'], df['Income'])
# Create a frequency table for Product and Income
print(frequency_table)
```

STEP 2

```
Income   29562   30699   31836   32973   34110   35247   36384   37521   \
Product
KP281        1       1       1       3       2       5       3       2
KP481        0       0       1       2       3       0       1       0
KP781        0       0       0       0       0       0       0       0

Income   38658   39795   ...  85906   88396   89641   90886   92131   95508   \
Product                  ...
KP281        3       2   ...      0       0       0       0       0       0
KP481        2       0   ...      0       0       0       0       0       0
KP781        0       0   ...      1       2       2       3       3       1

Income   95866   99601   103336  104581
Product
KP281        0       0       0       0
KP481        0       0       0       0
KP781        1       1       1       2

[3 rows x 62 columns]
```

STEP 3

In step 3 a list of all customers was passed for the <40000 incomes i.e. the low income customers and a loc function was called to locate the KP281 product from the above frequency table.

```
# Use a list to select multiple values for the 'Income' index
Low_Income_purchases = frequency_table.loc['KP281', [29562, 306
Low_Income_Purchases_KP_281 = Low_Income_purchases.sum()
```

Step 4

The total low income purchases for KP 281 were than divided by the total purchases of KP 281 i.e. 80.

```
#Percentage of Low income purchases for KP_281
percentage_low_income = (Low_Income_Purchases_KP_281 / 80) * 100
percentage_low_income
```

```
28.749999999999996
```

Step 3 and 4 were repeated for each product similarly and then a dictionary was defined to call the percentage of low income purchases.

For KP481

```
Low_Income_purchases = frequency_table.loc['KP481', [29562, 30699, 31836,
Low_Income_Purchases_KP_481 = Low_Income_purchases.sum()
```

```
#Percentage of Low income purchases for KP_481
percentage_low_income = (Low_Income_Purchases_KP_481 / 60) * 100
percentage_low_income
```

```
15.0
```

For KP781

```
Low_Income_purchases = frequency_table.loc['KP781', [29562, 30699, 31836,
Low_Income_Purchases_KP_781 = Low_Income_purchases.sum()
```

```
#Percentage of Low income purchases for KP_481
percentage_low_income = (Low_Income_Purchases_KP_781 / 40) * 100
percentage_low_income
```

```
0.0
```

Dictionary for Product by %age of Low income customers

```
# Calculate the percentage of low-income customers purchasing each treadmill model
percentage_low_income = {'KP281':'28.75','KP481':15.0,'KP781':0}
percentage_low_income
```

```
{'KP281': '28.75', 'KP481': 15.0, 'KP781': 0}
```

```
Frequency Table:
Income   29562   30699   31836   32973   34110   35247   36384   37521   \
Product
KP281        1       1       1       3       2       5       3       2
KP481        0       0       1       2       3       0       1       0
KP781        0       0       0       0       0       0       0       0

Income   38658   39795  ...  85906   88396   89641   90886   92131   95508   \
Product                 ...
KP281        3       2  ...      0       0       0       0       0       0
KP481        2       0  ...      0       0       0       0       0       0
KP781        0       0  ...      1       2       2       3       3       1

Income   95866   99601   103336  104581
Product
KP281        0       0       0       0
KP481        0       0       0       0
KP781        1       1       1       2

[3 rows x 62 columns]

Percentage of Low-income customers purchasing each Treadmill model:
{'KP281': '28.75', 'KP481': 15.0, 'KP781': 0}
```


Percentage of a high-income customer purchasing KP781 treadmill

Same steps were applied for deducing the percentage of a high income customer purchasing KP 781 except that the frequency table was located for KP 781 using the loc function and a list was passed for the above 80000 incomes. Once the high income purchase were determined its value 17 was divided by total purchase of KP781 which were 40 and multiplied by 100 to give 42.5%.

```
# Percentage of a high-income customer purchasing KP781 treadmill
High_Income_purchases = frequency_table.loc['KP781', [85906, 88396, 89641, 90886, 92131, 95508, 95866, 99601,
High_Income_purchases_KP_781 = High_Income_purchases.sum()
High_Income_purchases_KP_781
```

```
17
```

```
#Percentage of High income purchases for KP_481
percentage_high_income = (High_Income_purchases_KP_781 / 40) * 100
percentage_high_income
```

```
42.5
```

```
# Calculate the percentage of high-income customers purchasing each KP781
percentage_high_income = {'KP781':42.5}
percentage_high_income
```

```
{'KP781': 42.5}
```

Percentage of customer with high-income salary buying treadmill given that Product is KP781

The below code was run and in front of each high income salary customer the respective percentage purchased of KP781 treadmill was shown

```
#Percentage of customer with high-income salary buying treadmill given that Product is KP781
frequency_table.loc['KP781', [85906, 88396, 89641, 90886, 92131, 95508, 95866, 99601, 103336, 104581]] / 40 *
```

	KP781
Income	
85906	2.5
88396	5.0
89641	5.0
90886	7.5
92131	7.5
95508	2.5
95866	2.5
99601	2.5
103336	2.5
104581	5.0

dtype: float64

Product – Fitness

Percentage of customers that have fitness level 5

Running the below code, it was found that 17.22% of customers have fitness level 5

```
#Percentage of customers that have fitness level 5
df['Fitness'].value_counts(normalize=True) * 100
#Filter results above for fitness level 5 = 17.22
```

Fitness	proportion
3	53.888889
5	17.222222
2	14.444444
4	13.333333
1	1.111111

dtype: float64

Percentage of a customer with Fitness Level 5 purchasing KP781 treadmill

Running the below code created a matrix showing a bifurcation for each fitness level alongside the frequency of customers purchasing each treadmill.

```
#Percentage of a customer with Fitness Level 5 purchasing KP781 treadmill
frequency_table = pd.crosstab(df['Product'], df['Fitness'])
frequency_table
```

Fitness	1	2	3	4	5
Product					
KP281	1	14	54	9	2
KP481	1	12	39	8	0
KP781	0	0	4	7	29

Then executing the 2 codes below helped find out that the percentage of customers with fitness level 5 who purchased KP781 was 93%.

```
Fitness_Level_5 = frequency_table.loc['KP781', 5]
Fitness_Level_5
```

```
29
```

```
Percentage_Fitness_Level_5_KP_781 = 29/31 *100
Percentage_Fitness_Level_5_KP_781
```

```
93.54838709677419
```

Percentage of customer with fitness level 5 buying KP281 treadmill

Similarly running the below code helped find out that the percentage of customers with fitness level 5 purchasing KP281 treadmills was around 6%.

```
#Percentage of customer with fitness level 5 buying KP281 treadmill
Percentage_Fitness_Level_5_KP_281 = 2/31 * 100
Percentage_Fitness_Level_5_KP_281
```

```
6.451612903225806
```

Product - Marital Status

Percentage of a customers who are partnered using treadmills

Bifurcation for Product by Marital Status

```
#Percentage of a customers who are partnered using treadmills
frequency_table = pd.crosstab(df['Product'], df['MaritalStatus'])
frequency_table
```

MaritalStatus	Partnered	Single
Product		
KP281	48	32
KP481	36	24
KP781	23	17

Firstly, a matrix was shown to show the purchase distribution by marital status of each of the 3 treadmills. Then, the total sum of the purchases by customers whose marital status was partnered was calculated

```
#Sum of Partnered using all 3 products
Partnered_KP_281 = frequency_table.loc['KP281', 'Partnered']
Partnered_KP_281

Partnered_KP_481 = frequency_table.loc['KP481', 'Partnered']
Partnered_KP_481

Partnered_KP_781 = frequency_table.loc['KP781', 'Partnered']
Partnered_KP_781

Total_Partnered = Partnered_KP_281 + Partnered_KP_481 + Partnered_KP_781
Total_Partnered
```

```
107
```

```
#Total Marital Staus using treadmills
Total_Marital_Status = frequency_table.sum(axis=0)
Total_Marital_Status
```

	0
MaritalStatus	
Partnered	107
Single	73

dtype: int64

Once it was found that 107 customers were partnered and purchased all treadmills, that was divided by 180 which is the total customers figure whether single or partnered to arrive at the percentage of partnered customers that purchased treadmills i.e. 59.44%.

```
Total_Marital_Status = 180
#Percentage of Partnered using treadmills
Percentage_Partnered = (Total_Partnered / Total_Marital_Status) * 100
Percentage_Partnered
```

```
59.44444444444444
```

Recommendations for the Market Research Team

Based on the observations provided, here are some actionable recommendations for the AeroFit market research team:

Segmentation Analysis

Action Incorporate fitness levels, income categories, and age groups into the segmentation analysis.

Purpose: Understanding how fitness levels correlate with product preferences can enhance targeting strategies.

Targeted Marketing Campaigns

Action: Develop tailored marketing campaigns for specific income categories (low, medium, high) and fitness levels (e.g., targeting fitness enthusiasts for KP781).

Purpose: Customized messages will resonate more with each income segment, improving engagement and conversions.

Product Feature Highlighting

Action: Emphasize specific features of each treadmill that align with customer characteristics, such as durability for high-income customers and affordability for low-income buyers.

Purpose: Aligning product features with customer needs will enhance perceived value and satisfaction.

Customer Feedback Loop

Action: Gather feedback on how fitness levels influence product selection, especially for higher-end models like KP781.

Purpose: Understanding motivations can refine marketing and product development.

Educational Content Development

Action: Create fitness-oriented content tailored to different fitness levels, addressing how each treadmill can support specific fitness goals.

Purpose: Educational resources can help customers see the value of investing in a treadmill aligned with their fitness ambitions.

Sales Training for Staff

Action: Train staff to recognize and respond to various income and fitness levels, helping them tailor their recommendations effectively.

Purpose: Personalized sales approaches can enhance customer experiences and increase upselling opportunities.

Incorporate Fitness Metrics in Marketing

Action: Highlight the average fitness levels of customers purchasing each treadmill in marketing materials, particularly focusing on KP781 for high-fitness customers.

Purpose: This can create a community feel and attract customers who aspire to reach similar fitness levels.

Analysis of Purchase Patterns by Gender

Action: Create targeted campaigns that address the specific interests of male and female customers, particularly noting the higher percentage of male customers purchasing KP781.

Purpose: Tailored messaging can improve engagement and increase sales among specific gender demographics.

Promote Community Engagement Initiatives

Action: Consider organizing fitness challenges or community events, particularly for high-fitness customers, encouraging the use of KP781.

Purpose: Building a community around fitness can enhance brand loyalty and word-of-mouth marketing.

Use of Conditional Probability Insights

Action: Leverage insights from conditional probabilities to create marketing strategies that address customer likelihood of purchase based on their profiles (e.g., high-income customers for KP781).

Purpose: This data-driven approach can maximize marketing effectiveness and sales strategies.

Visualize Data for Ongoing Analysis

Action: Regularly update visualizations of customer data to track changes in demographics, income categories, and fitness levels.

Purpose: Continuous monitoring will allow AeroFit to adapt its strategies in real time to evolving customer needs.

Retention Strategies for High-Fitness Customers

Action: Develop loyalty programs or discounts for repeat customers who purchase high-end products like KP781, particularly those with high fitness levels.

Purpose: Retaining high-value customers can lead to increased lifetime value and referrals.

By integrating these updated and new recommendations, AeroFit can better position itself in the market, improve customer engagement, and drive sales across its treadmill product lines.