Problem Statement:

The goal is to identify the characteristics of the target audience for each AeroFit treadmill product (KP281, KP481, KP781) to better recommend treadmills to new customers. This involves understanding whether there are differences across products with respect to customer characteristics.

Dataset Overview

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

```
# Load the dataset
df = pd.read_csv('/kaggle/input/aerofit-data/aerofit_treadmill.csv')

# Display the first few rows of the dataset
df.head()
```

	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

Non-Graphical Analysis

```
# Checking the structure and summary of the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Product        180 non-null    object
 1   Age            180 non-null    int64
 2   Gender         180 non-null    object
 3   Education      180 non-null    int64
 4   MaritalStatus  180 non-null    object
 5   Usage          180 non-null    int64
 6   Fitness        180 non-null    int64
 7   Income         180 non-null    int64
 8   Miles          180 non-null    int64
dtypes: int64(6), object(3)
memory usage: 12.8+ KB
```

```
cat_cols = df.select_dtypes(include='object').columns
df[cat_cols] = df[cat_cols].astype('category')

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Product        180 non-null    category
 1   Age            180 non-null    int64
 2   Gender         180 non-null    category
 3   Education      180 non-null    int64
 4   MaritalStatus  180 non-null    category
 5   Usage          180 non-null    int64
 6   Fitness        180 non-null    int64
 7   Income         180 non-null    int64
 8   Miles          180 non-null    int64
dtypes: category(3), int64(6)
memory usage: 9.5 KB
```

```
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

```
df.isnull().sum()
```

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

```
df.nunique()
```

```
Product           3
Age              32
Gender            2
Education         8
MaritalStatus     2
Usage             6
Fitness           5
Income           62
Miles            37
dtype: int64
```

Visual Analysis

Univariate Analysis

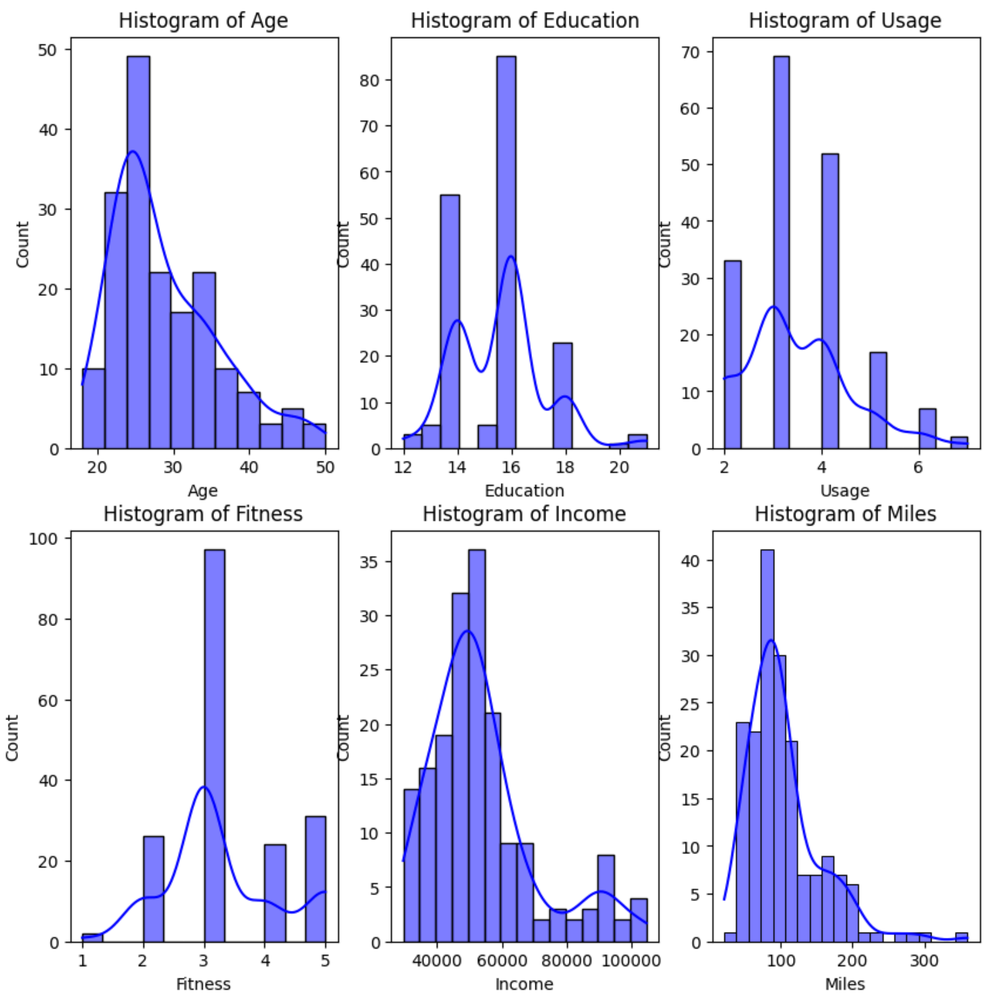
```
def plot_continuous_variable(data, variable, type='hist'):
    # Histogram
    if type=='hist':
        sns.histplot(data[variable].dropna(), color='blue', kde=True)
        plt.title(f'Histogram of {variable}')


def plot_categorical_variable(data, category, rotation=0):
    # Boxplot
    if len(data[category].value_counts()) > 5:
        top_5_categories = data[category].value_counts().nlargest(5).index
    # Filter the data to include only the top 5 categories
        filter_data = data[data[category].isin(top_5_categories)]
        sns.countplot(data=filter_data, x=category, palette='Set2', order=filter_data[category].value_counts().index)
        plt.xticks(rotation=rotation)
    else:
        sns.countplot(data=data, x=category, palette='Set2')
    plt.title(f'Barplot of {category}')
    plt.xlabel(category)
```

```
cat_cols = ['Product','Gender','MaritalStatus']
num_cols = ['Age','Education','Usage','Fitness','Income','Miles']
```

Numerical data

```
plt.figure(figsize=(10, 10))
i = 1
for col in num_cols:
    plt.subplot(2, 3, i)
    plot_continuous_variable(df, col, 'hist')
    i += 1
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na opt
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na opt
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na opt
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na opt
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na opt
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na opt
  with pd.option_context('mode.use_inf_as_na', True):
```

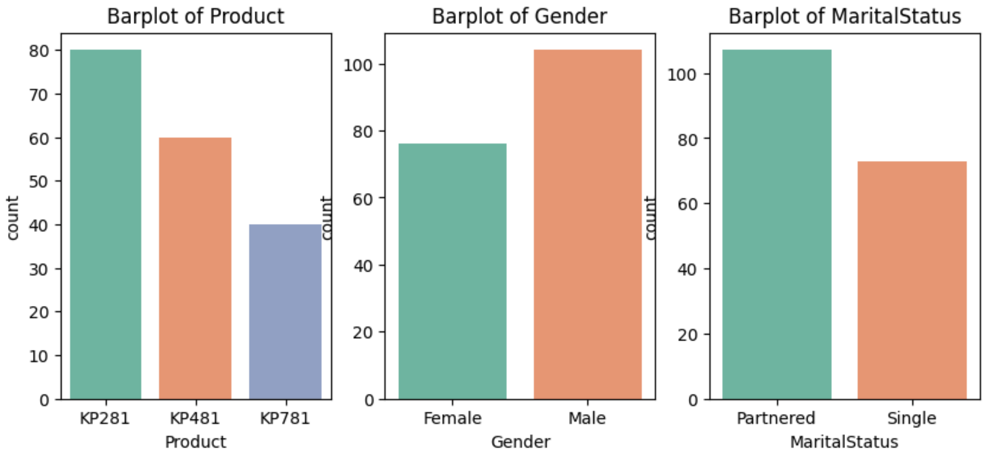


Most of the numerical data is close to or almost normally distributed data.

Categorical Data

```
plt.figure(figsize=(10, 4))
i = 1
for col in cat_cols:
    plt.subplot(1, 3, i)
    plot_categorical_variable(df, col)
    i += 1
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:641: FutureWarning: The default of
  grouped_vals = vals.groupby(grouper)
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:641: FutureWarning: The default of
  grouped_vals = vals.groupby(grouper)
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:641: FutureWarning: The default of
  grouped_vals = vals.groupby(grouper)
```



- **Mojority of data is on Product KP281.**
- **Most of the data is of Males and Partnered.**

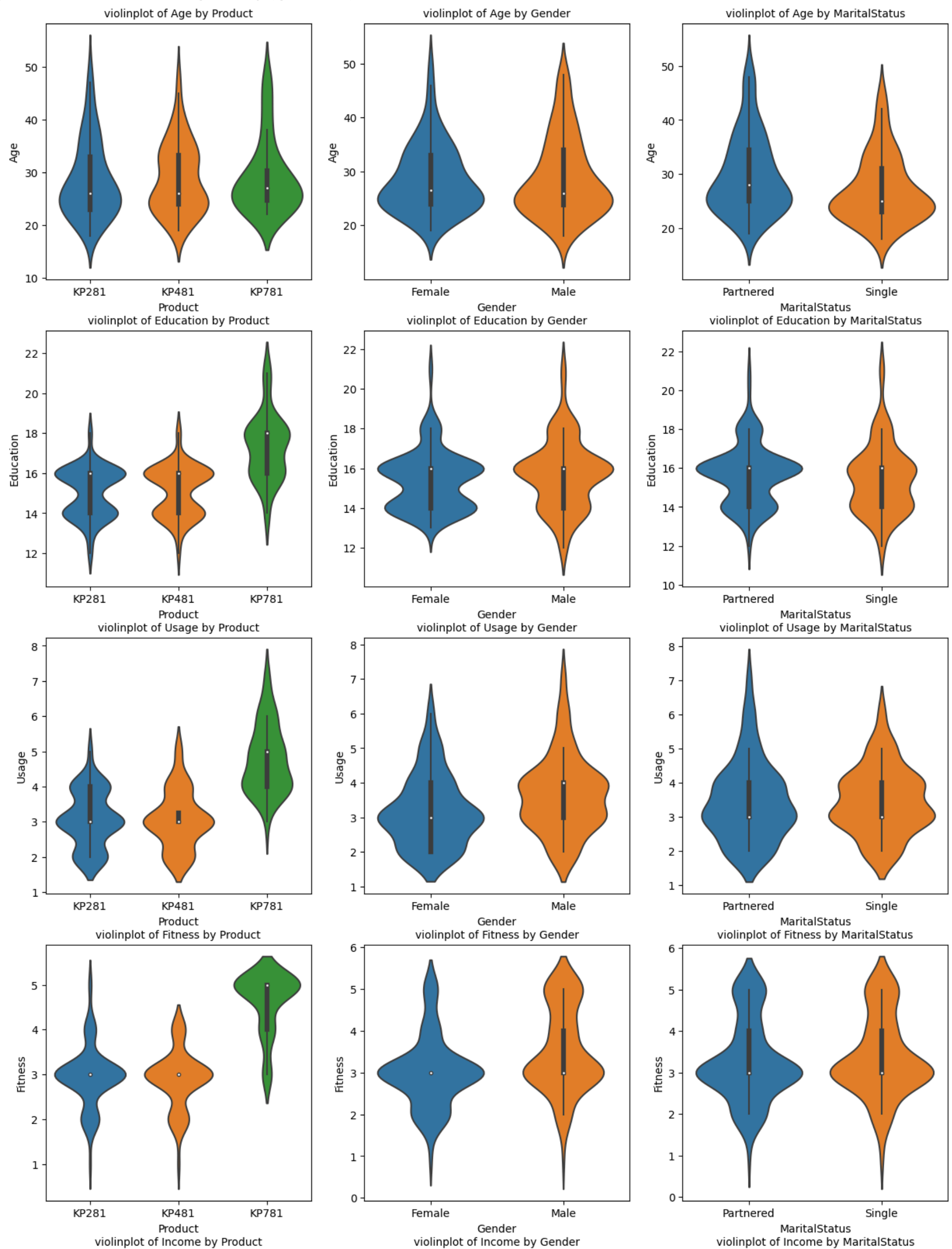
Bi-variate Analysis

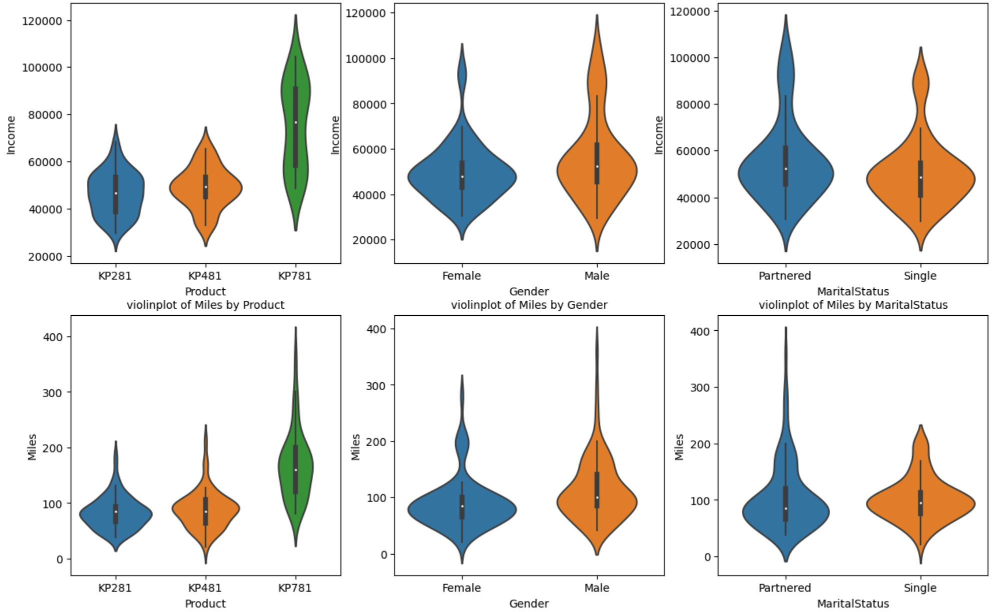
```
def plot_bivariate_plot_NC(data, category, variable):
    sns.violinplot(x=df[category].dropna(), y=df[variable].dropna())
    plt.title(f'violinplot of {variable} by {category}',fontdict={'fontsize':10})
    plt.xlabel(category)
    plt.ylabel(variable)


def plot_bivariate_plot_CC(data, category_1, category_2, rotation=0):
    sns.countplot(data=data, x=category_1, hue=category_2, palette='Set2')
    plt.xticks(rotation=rotation)
    plt.title(f'Grouped Barplot of {category_1} and {category_2}')
    plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')

# Define a function to create the required bivariate plots for continuous-continuous variables
def plot_bivariate_plot_NN(data, variable_1, variable_2):
    # Scatterplot
    sns.scatterplot(x=variable_1, y=variable_2, data=data)
    plt.title(f'scatterPlot of {variable_1} by {variable_2}')
    plt.xlabel(variable_1)
    plt.ylabel(variable_2)
```

Categorical - Continuous Data

```
plt.figure(figsize=(15,30))
i = 1
for col_n in num_cols:
    for col_c in cat_cols:
        plt.subplot(6, 3, i)
        plot_bivariate_plot_NC(df, col_c, col_n)
        i += 1
plt.show()
```

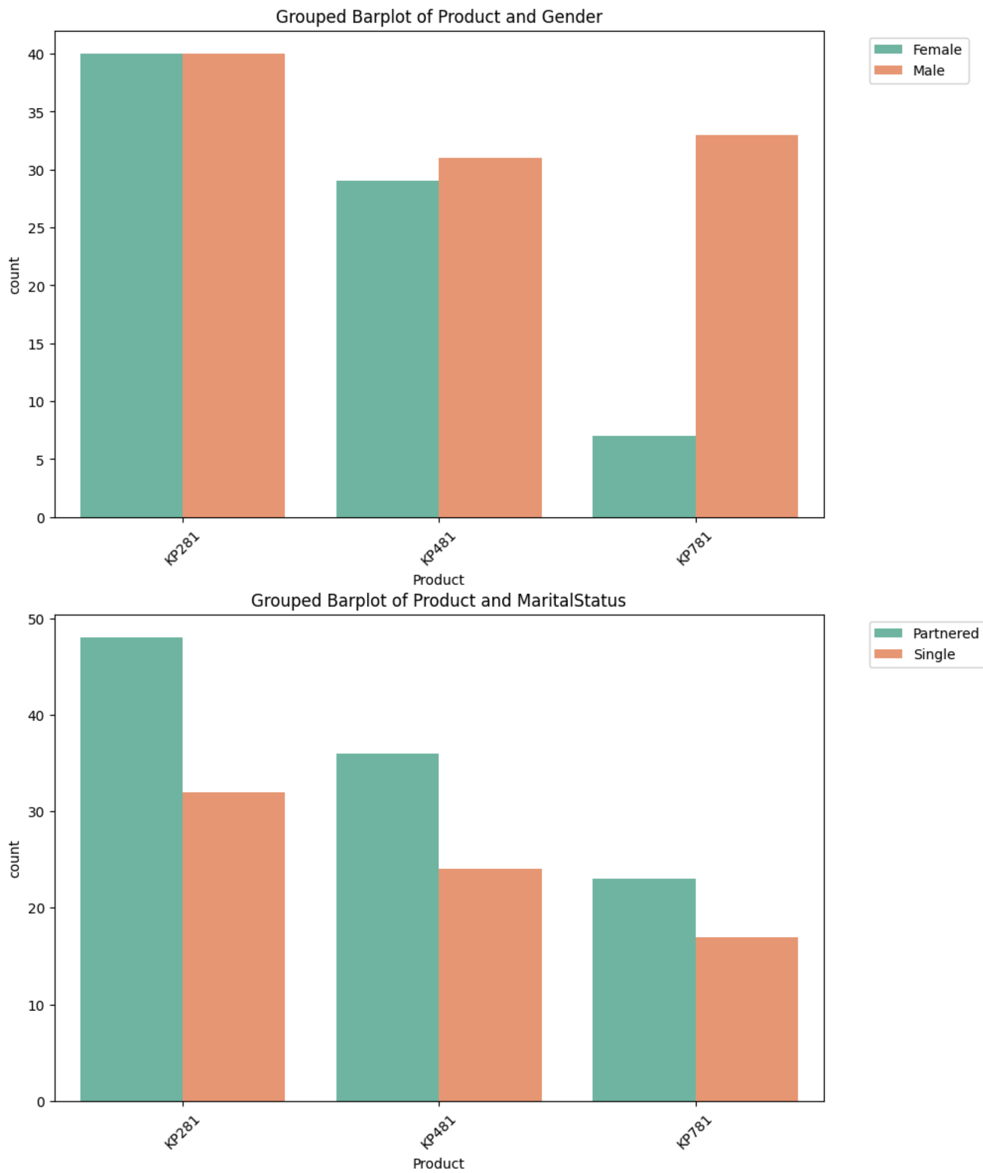
```
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:641: FutureWarning: The default of
  grouped_vals = vals.groupby(grouper)
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:641: FutureWarning: The default of
  grouped_vals = vals.groupby(grouper)
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:641: FutureWarning: The default of
  grouped_vals = vals.groupby(grouper)
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:641: FutureWarning: The default of
  grouped_vals = vals.groupby(grouper)
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:641: FutureWarning: The default of
  grouped_vals = vals.groupby(grouper)
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:641: FutureWarning: The default of
  grouped_vals = vals.groupby(grouper)
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:641: FutureWarning: The default of
  grouped_vals = vals.groupby(grouper)
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:641: FutureWarning: The default of
  grouped_vals = vals.groupby(grouper)
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:641: FutureWarning: The default of
  grouped_vals = vals.groupby(grouper)
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:641: FutureWarning: The default of
  grouped_vals = vals.groupby(grouper)
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:641: FutureWarning: The default of
  grouped_vals = vals.groupby(grouper)
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:641: FutureWarning: The default of
  grouped_vals = vals.groupby(grouper)
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:641: FutureWarning: The default of
  grouped_vals = vals.groupby(grouper)
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:641: FutureWarning: The default of
  grouped_vals = vals.groupby(grouper)
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:641: FutureWarning: The default of
  grouped_vals = vals.groupby(grouper)
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:641: FutureWarning: The default of
  grouped_vals = vals.groupby(grouper)
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:641: FutureWarning: The default of
  grouped_vals = vals.groupby(grouper)
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:641: FutureWarning: The default of
  grouped_vals = vals.groupby(grouper)
```





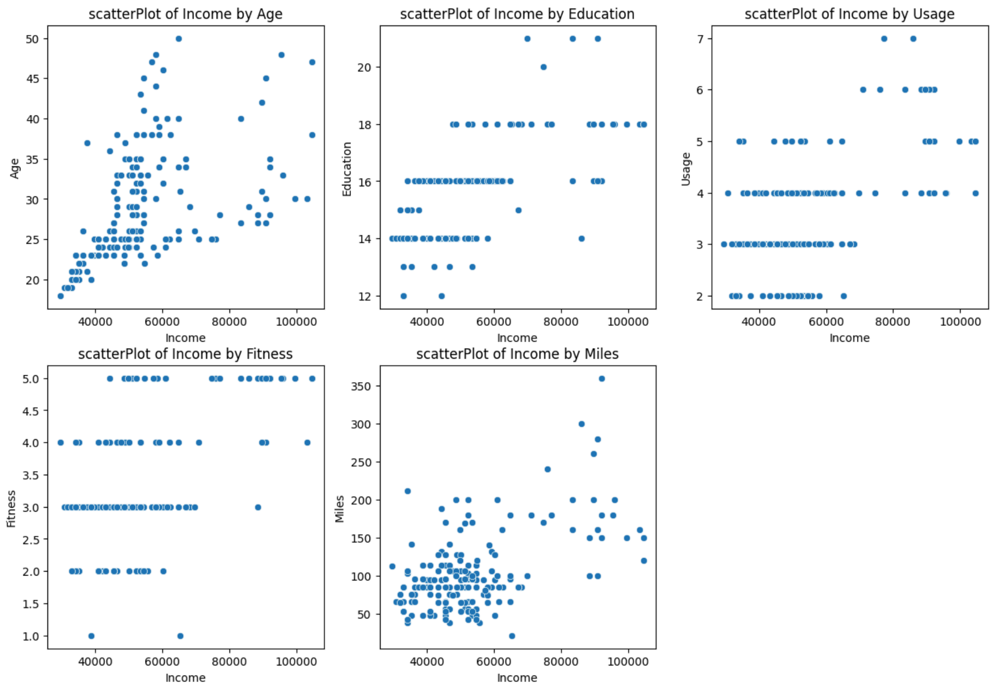
Categorical - Categorical Data

```
plt.figure(figsize=(10, 14))
plt.subplot(2, 1, 1)
plot_bivariate_plot_CC(df, cat_cols[0], cat_cols[1], 45)
plt.subplot(2, 1, 2)
plot_bivariate_plot_CC(df, cat_cols[0], cat_cols[2], 45)
plt.show()
```



Continuous - Continuous Data

```
plt.figure(figsize=(15, 10))
i = 1
for col in num_cols:
    if col!='Income':
        plt.subplot(2, 3, i)
        plot_bivariate_plot_NN(df, 'Income', col)
        i += 1
plt.show()
```

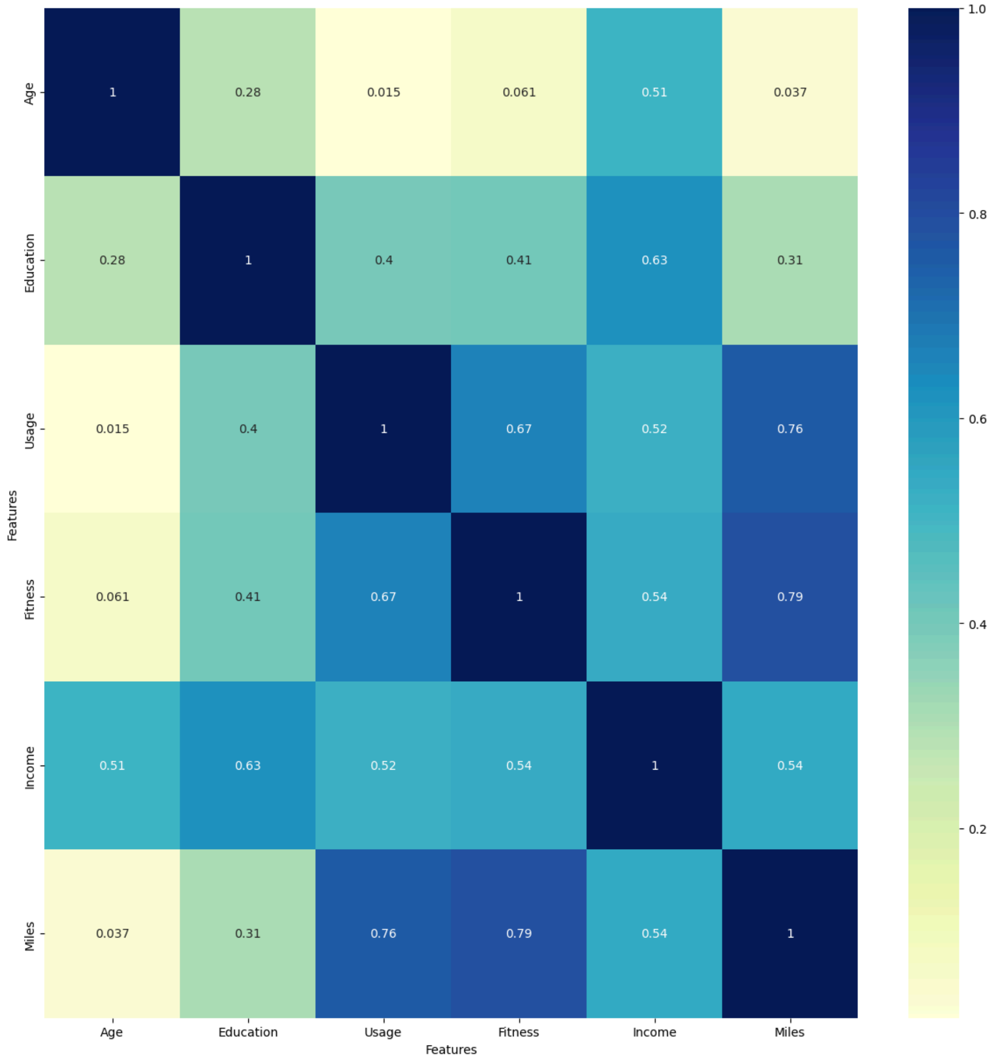


Most of the numerical data is not trending closely with the income, so income isn't dependant on most of the other numerical columns.

Correlation Analysis

Correlation Heatmap

```
correlation_matrix = df[num_cols].corr()

plt.figure(figsize=(15, 15))
sns.heatmap(correlation_matrix, annot=True, cmap="YlGnBu")
plt.xlabel('Features')
plt.ylabel('Features')
plt.show()
```



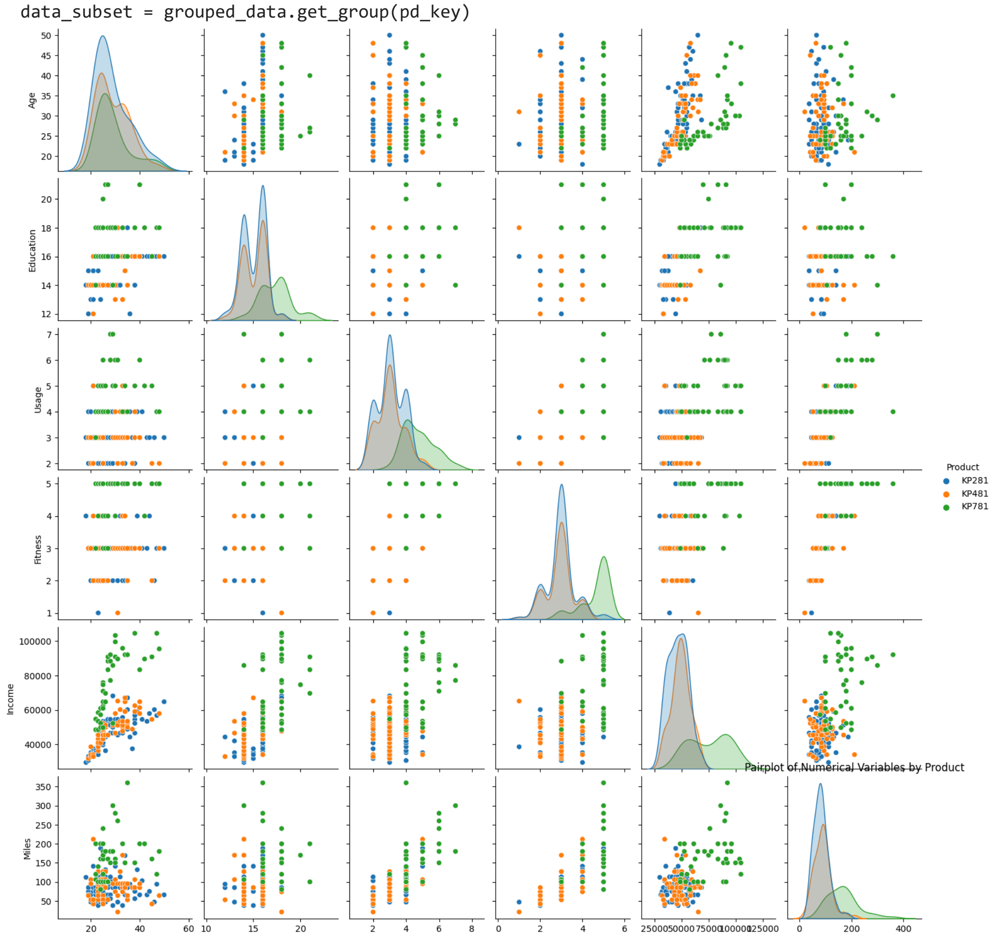
- **Age:** Positive Correlation with Income (0.513): Older customers tend to have higher incomes. This makes sense as income typically increases with age and career progression. Weak Correlations with Other Variables: Age does not strongly correlate with education, usage, fitness, or miles, suggesting that these factors are relatively independent of age.
- **Education:** Positive Correlation with Income (0.626): Higher educational levels are associated with higher incomes, which is a common socio-economic trend. Moderate Correlation with Fitness (0.411) and Usage (0.395): Better-educated customers tend to rate their fitness higher and use the treadmill more frequently. This suggests a link between education and health-conscious behavior. Positive Correlation with Miles (0.307): Higher education levels slightly correlate with the number of miles run or walked, reflecting a trend of educated individuals engaging more in physical activities.
- **Usage:** Strong Correlation with Miles (0.759): Customers who use the treadmill more frequently tend to cover more miles. This is intuitive as higher usage generally means more miles. Strong Correlation with Fitness (0.669): Higher treadmill usage is associated with better self-rated fitness levels. Regular use of the treadmill contributes to improved fitness. Moderate Correlation with Income (0.520): Higher income individuals tend to use the treadmill more frequently, possibly due to better access to fitness resources.

- **Fitness:** Strong Correlation with Miles (0.786): Customers who rate their fitness higher tend to cover more miles on the treadmill. This reflects the relationship between physical activity and fitness levels. Moderate Correlation with Income (0.535): Wealthier customers tend to rate their fitness higher, likely due to better access to fitness and healthcare facilities.
- **Income:** Moderate to Strong Correlations with Other Variables: Income shows moderate to strong positive correlations with education (0.626), usage (0.520), fitness (0.535), and miles (0.543). This indicates that higher-income individuals are generally more educated, use the treadmill more, have higher fitness levels, and cover more miles.
- **Miles:** Strong Correlation with Fitness (0.786) and Usage (0.759): More miles run or walked are associated with higher fitness levels and greater treadmill usage. Moderate Correlation with Income (0.543): Higher income is associated with more miles covered, possibly due to better access to fitness equipment and a healthier lifestyle.

Pairplot

```
sns.pairplot(df, hue='Product')
plt.title('Pairplot of Numerical Variables by Product')
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na opt
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1057: FutureWarning: The default of ol
  grouped_data = data.groupby(
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1075: FutureWarning: When grouping wi
  data_subset = grouped_data.get_group(pd_key)
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na opt
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1057: FutureWarning: The default of ol
  grouped_data = data.groupby(
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1075: FutureWarning: When grouping wi
  data_subset = grouped_data.get_group(pd_key)
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na opt
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1057: FutureWarning: The default of ol
  grouped_data = data.groupby(
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1075: FutureWarning: When grouping wi
  data_subset = grouped_data.get_group(pd_key)
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na opt
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1057: FutureWarning: The default of ol
  grouped_data = data.groupby(
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1075: FutureWarning: When grouping wi
  data_subset = grouped_data.get_group(pd_key)
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na opt
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1057: FutureWarning: The default of ol
  grouped_data = data.groupby(
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1075: FutureWarning: When grouping wi
  data_subset = grouped_data.get_group(pd_key)
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na opt
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1057: FutureWarning: The default of ol
  grouped_data = data.groupby(
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1075: FutureWarning: When grouping wi
  data_subset = grouped_data.get_group(pd_key)
```



Outlier Detection

```
# Checking for outliers using IQR method
num_data = df[num_cols]
Q1 = num_data.quantile(0.25)
Q3 = num_data.quantile(0.75)
IQR = Q3 - Q1
outliers = num_data[(num_data < (Q1 - 1.5 * IQR)) | (num_data > (Q3 + 1.5 * IQR))].dropna(how='all')
outliers
```

	Age	Education	Usage	Fitness	Income	Miles
14	NaN	NaN	NaN	1.0	NaN	NaN
23	NaN	NaN	NaN	NaN	NaN	188.0
78	47.0	NaN	NaN	NaN	NaN	NaN
79	50.0	NaN	NaN	NaN	NaN	NaN
84	NaN	NaN	NaN	NaN	NaN	212.0
117	NaN	NaN	NaN	1.0	NaN	NaN
139	48.0	NaN	NaN	NaN	NaN	NaN
142	NaN	NaN	NaN	NaN	NaN	200.0
148	NaN	NaN	NaN	NaN	NaN	200.0
152	NaN	NaN	NaN	NaN	NaN	200.0
154	NaN	NaN	6.0	NaN	NaN	NaN
155	NaN	NaN	6.0	NaN	NaN	240.0
156	NaN	20.0	NaN	NaN	NaN	NaN
157	NaN	21.0	NaN	NaN	NaN	NaN
159	NaN	NaN	NaN	NaN	83416.0	NaN
160	NaN	NaN	NaN	NaN	88396.0	NaN
161	NaN	21.0	NaN	NaN	90886.0	NaN
162	NaN	NaN	6.0	NaN	92131.0	NaN
163	NaN	NaN	7.0	NaN	NaN	NaN
164	NaN	NaN	6.0	NaN	88396.0	NaN
166	NaN	NaN	7.0	NaN	85906.0	300.0
167	NaN	NaN	6.0	NaN	90886.0	280.0
168	NaN	NaN	NaN	NaN	103336.0	NaN
169	NaN	NaN	NaN	NaN	99601.0	NaN
170	NaN	NaN	6.0	NaN	89641.0	260.0
171	NaN	NaN	NaN	NaN	95866.0	200.0
172	NaN	NaN	NaN	NaN	92131.0	NaN
173	NaN	NaN	NaN	NaN	92131.0	360.0
174	NaN	NaN	NaN	NaN	104581.0	NaN
175	NaN	21.0	6.0	NaN	83416.0	200.0
176	NaN	NaN	NaN	NaN	89641.0	200.0
177	NaN	NaN	NaN	NaN	90886.0	NaN
178	47.0	NaN	NaN	NaN	104581.0	NaN
179	48.0	NaN	NaN	NaN	95508.0	NaN

Marginal Probability

```
# Marginal probability of each product being purchased
product_counts = df['Product'].value_counts(normalize=True)
product_counts
```

```
Product
KP281    0.444444
KP481    0.333333
KP781    0.222222
Name: proportion, dtype: float64
```

Double-click (or enter) to edit

Conditional Probability/Two-way Contingency Tables

```
# Contingency table for Product and Gender
contingency_table_gender = pd.crosstab(df['Product'], df['Gender'], normalize='index')
contingency_table_gender
```

Gender	Female	Male
Product		
KP281	0.500000	0.500000
KP481	0.483333	0.516667
KP781	0.175000	0.825000

- **KP281:** Equal distribution of male and female customers (50% each). This suggests that the KP281 model appeals equally to both genders.
- **KP481:** Slightly more male customers (51.7%) than female customers (48.3%). This indicates a marginally higher preference among males for the KP481 model.
- **KP781:** A significantly higher proportion of male customers (82.5%) compared to female customers (17.5%). This suggests that the KP781 model is predominantly favored by males.

```
# Contingency table for Product and Marital Status
contingency_table_marital = pd.crosstab(df['Product'], df['MaritalStatus'], normalize='index')
contingency_table_marital
```

MaritalStatus	Partnered	Single
Product		
KP281	0.600	0.400
KP481	0.600	0.400
KP781	0.575	0.425

- **KP281:** 60% of customers are partnered, and 40% are single. This indicates a higher preference for the KP281 model among partnered individuals.
- **KP481:** Similar to KP281, 60% of customers are partnered, and 40% are single. The KP481 model also appeals more to partnered