## Problem Statement:

Analyze Aerofit customer data to profile buyers of KP281, KP481, and KP781 treadmills and provide insights for better marketing and product recommendations.

## 1. Import the dataset and do usual data analysis steps like checking the structure & characteristics of the dataset.

### Import all files

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

### Import Data set

```
df = pd.read_csv('aerofit_treadmill.csv')
```

```
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 |

Next steps: Generate code with df | View recommended plots | New interactive sheet

### Basic Data Analysis

```
df.shape
```

```
(180, 9)
```

```
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
```

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

| | 0 |
|---|---|
| **Product** | 0 |
| **Age** | 0 |
| **Gender** | 0 |
| **Education** | 0 |
| **MaritalStatus** | 0 |
| **Usage** | 0 |
| **Fitness** | 0 |
| **Income** | 0 |
| **Miles** | 0 |

**dtype:** int64

```
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['Product'].value_counts()
```

| Product | count |
|---|---|
| **KP281** | 80 |
| **KP481** | 60 |
| **KP781** | 40 |

**dtype:** int64

## 2. Detect Outliers

Continuous Variables

1. Age
2. Usage
3. Fitness
4. Income
5. Miles

Age:

```
df['Age'].head()
```

| | Age |
|---|---|
| **0** | 18 |
| **1** | 19 |
| **2** | 19 |
| **3** | 19 |
| **4** | 20 |

**dtype:** int64

```
Q1 = np.percentile(df['Age'],25)
```

```
Q1
```

```
np.float64(24.0)
```

```
Q3 = np.percentile(df['Age'],75)
```

```
Q3
```

```
np.float64(33.0)
```

```
IQR = Q3 - Q1
```

```
IQR
```

```
np.float64(9.0)
```

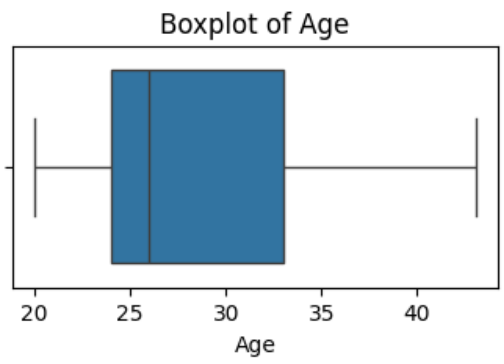
```
outliers = df[(df['Age'] < Q1 - 1.5 * IQR) | (df['Age'] > Q3 + 1.5 * IQR)]
```

```
outliers.head()
```

| | Product | Age | Gender | Education | MaritalStatus | Usage | Fitness | Income | Miles |
|---|---|---|---|---|---|---|---|---|---|
| **154** | KP781 | 25 | Male | 18 | Partnered | 6 | 4 | 70966 | 180 |
| **155** | KP781 | 25 | Male | 18 | Partnered | 6 | 5 | 75946 | 240 |
| **162** | KP781 | 28 | Female | 18 | Partnered | 6 | 5 | 92131 | 180 |
| **163** | KP781 | 28 | Male | 18 | Partnered | 7 | 5 | 77191 | 180 |
| **164** | KP781 | 28 | Male | 18 | Single | 6 | 5 | 88396 | 150 |

Next steps: Generate code with outliers | View recommended plots | New interactive sheet

```
plt.figure(figsize=(4,2))
sns.boxplot(x=df['Age'])
plt.title("Boxplot of Age")
plt.show()
```



Usage:

```
Q1 = np.percentile(df['Usage'],25)
Q3 = np.percentile(df['Usage'],75)
```

```
Q1
```

```
np.float64(3.0)
```

```
Q3
```

```
np.float64(4.0)
```

```
IQR = Q3 - Q1
IQR
```

```
np.float64(1.0)
```

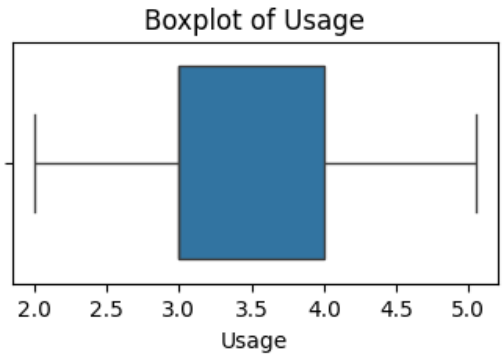
```
outliers = df[(df['Usage'] < Q1 - 1.5 * IQR) | (df['Usage'] > Q3 + 1.5 * IQR)]
```

```
outliers.head()
```

| | Product | Age | Gender | Education | MaritalStatus | Usage | Fitness | Income | Miles |
|---|---|---|---|---|---|---|---|---|---|
| **154** | KP781 | 25 | Male | 18 | Partnered | 6 | 4 | 70966 | 180 |
| **155** | KP781 | 25 | Male | 18 | Partnered | 6 | 5 | 75946 | 240 |
| **162** | KP781 | 28 | Female | 18 | Partnered | 6 | 5 | 92131 | 180 |
| **163** | KP781 | 28 | Male | 18 | Partnered | 7 | 5 | 77191 | 180 |
| **164** | KP781 | 28 | Male | 18 | Single | 6 | 5 | 88396 | 150 |

Next steps: Generate code with outliers | View recommended plots | New interactive sheet

```
plt.figure(figsize=(4,2))
sns.boxplot(x=df['Usage'])
plt.title("Boxplot of Usage")
plt.show()
```




Fitness:

```
Q1 = np.percentile(df['Fitness'],25)
Q3 = np.percentile(df['Fitness'],75)
```

```
IQR = Q3 - Q1
IQR
```

```
np.float64(1.0)
```

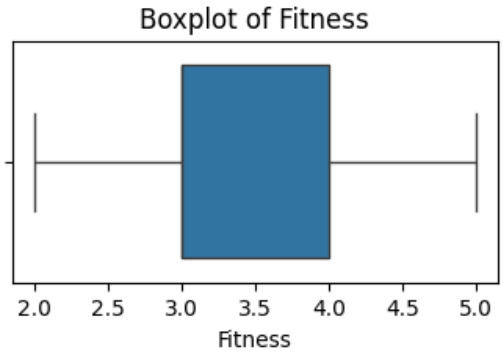
```
outliers = df[(df['Fitness'] < Q1 - 1.5 * IQR) | (df['Fitness'] > Q3 + 1.5 * IQR)]
```

```
outliers.head()
```

| | Product | Age | Gender | Education | MaritalStatus | Usage | Fitness | Income | Miles |
|---|---|---|---|---|---|---|---|---|---|
| **14** | KP281 | 23 | Male | 16 | Partnered | 3 | 1 | 38658 | 47 |
| **117** | KP481 | 31 | Female | 18 | Single | 2 | 1 | 65220 | 21 |

Next steps: Generate code with outliers | View recommended plots | New interactive sheet

```
plt.figure(figsize=(4,2))
sns.boxplot(x=df['Fitness'])
plt.title("Boxplot of Fitness")
plt.show()
```




Income:

```
Q1 = np.percentile(df['Income'],25)
Q3 = np.percentile(df['Income'],75)
```

```
Q1
```

```
np.float64(3.0)
```

```
Q3
```

```
np.float64(4.0)
```

```
IQR = Q3 - Q1
IQR
```

```
np.float64(1.0)
```

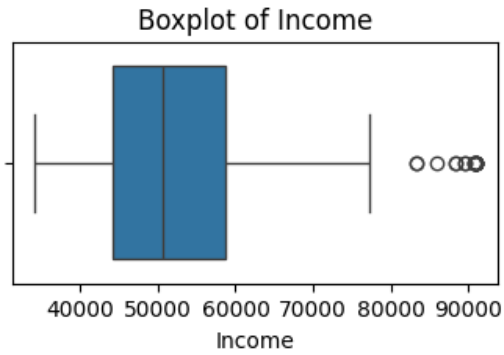
```
outliers = df[(df['Income'] < Q1 - 1.5 * IQR) | (df['Income'] > Q3 + 1.5 * IQR)]
```

```
outliers.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 |

Next steps: Generate code with outliers | View recommended plots | New interactive sheet

```
plt.figure(figsize=(4,2))
sns.boxplot(x=df['Income'])
plt.title("Boxplot of Income")
plt.show()
```




Miles:

```
Q1 = np.percentile(df['Miles'],25)
Q3 = np.percentile(df['Miles'],75)
```

```
IQR = Q3 - Q1
IQR
```

```
np.float64(48.75)
```

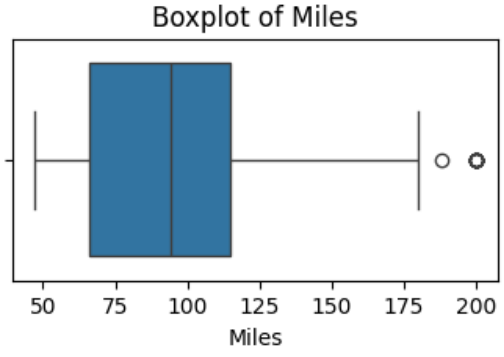
```
outliers = df[(df['Miles'] < Q1 - 1.5 * IQR) | (df['Miles'] > Q3 + 1.5 * IQR)]
```

```
outliers.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 |

Next steps: Generate code with outliers | View recommended plots | New interactive sheet

```
plt.figure(figsize=(4,2))
sns.boxplot(x=df['Miles'])
plt.title("Boxplot of Miles")
plt.show()
```




**Clipping the data between the 5 percentile and 95 percentile**

```
continuous_vars = ['Age', 'Usage', 'Fitness', 'Income', 'Miles']
for col in continuous_vars:
    lower = df[col].quantile(0.05)
    upper = df[col].quantile(0.95)
    df[col] = df[col].clip(lower, upper)
```

```
df[continuous_vars].describe()
```

| | Age | Usage | Fitness | Income | Miles |
|---|---|---|---|---|---|
| **count** | 180.000000 | 180.000000 | 180.000000 | 180.000000 | 180.000000 |
| **mean** | 28.641389 | 3.396944 | 3.322222 | 53477.070000 | 101.088889 |
| **std** | 6.446373 | 0.952682 | 0.937461 | 15463.662523 | 43.364286 |
| **min** | 20.000000 | 2.000000 | 2.000000 | 34053.150000 | 47.000000 |
| **25%** | 24.000000 | 3.000000 | 3.000000 | 44058.750000 | 66.000000 |
| **50%** | 26.000000 | 3.000000 | 3.000000 | 50596.500000 | 94.000000 |
| **75%** | 33.000000 | 4.000000 | 4.000000 | 58668.000000 | 114.750000 |
| **max** | 43.050000 | 5.050000 | 5.000000 | 90948.250000 | 200.000000 |

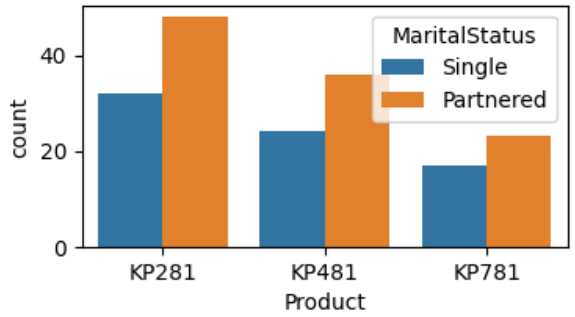
```
df[continuous_vars].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 5 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Age      180 non-null    float64
 1   Usage    180 non-null    float64
 2   Fitness  180 non-null    int64
 3   Income   180 non-null    float64
 4   Miles    180 non-null    int64
dtypes: float64(3), int64(2)
memory usage: 7.2 KB
```

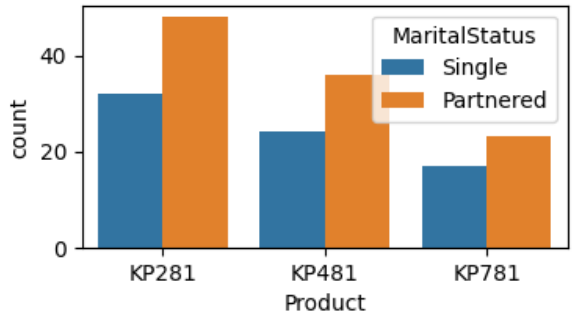
## 3. Check if features like marital status, Gender, and age have any effect on the product purchased.

Categorical Variables vs Product (Output Variable)

```
plt.figure(figsize=(4,2))
sns.countplot(x = 'Product', hue = 'MaritalStatus', data = df)
plt.show()
```



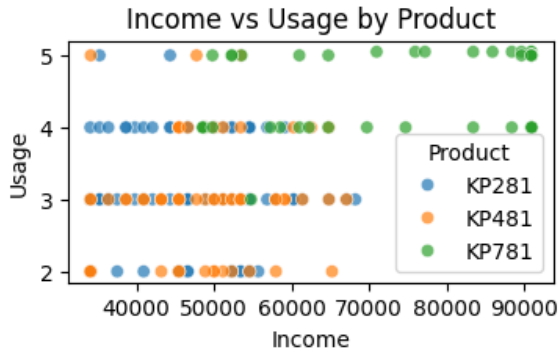

```
plt.figure(figsize=(4,2))
sns.countplot(x = 'Product', hue = 'MaritalStatus', data = df)
plt.show()
```




Continuous Variables vs Product (Output Variable)

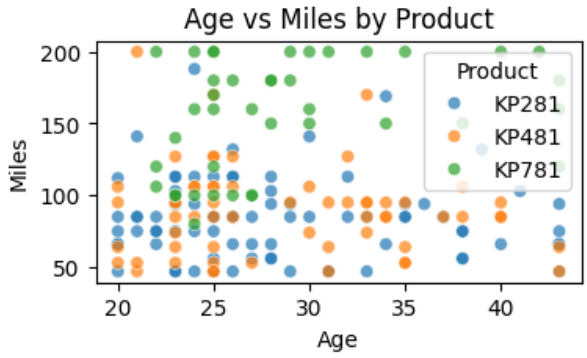
## Income vs Usage

```
plt.figure(figsize=(4,2))
sns.scatterplot(x='Income', y='Usage', hue='Product', data=df, alpha=0.7)
plt.title("Income vs Usage by Product")
plt.show()
```



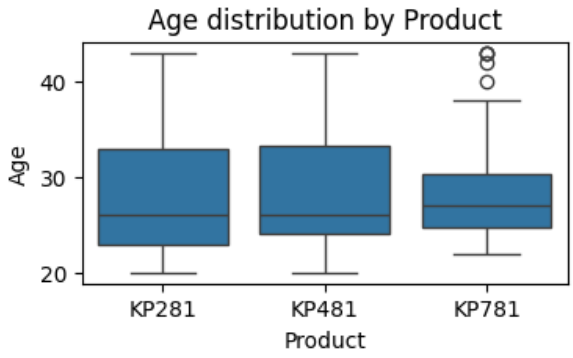

## Miles vs Age

```
plt.figure(figsize=(4,2))
sns.scatterplot(x='Age', y='Miles', hue='Product', data=df, alpha=0.7)
plt.title("Age vs Miles by Product")
plt.show()
```

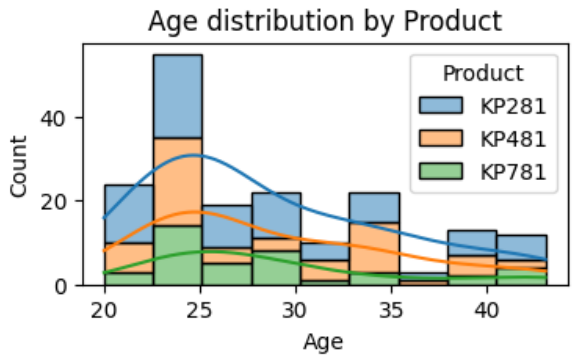



## Continuous vs categorical

```
plt.figure(figsize=(4,2))
sns.boxplot(x='Product', y='Age', data=df)
plt.title("Age distribution by Product")
plt.show()
```




```
plt.figure(figsize=(4,2))
sns.histplot(data=df, x='Age', hue='Product', kde=True, multiple="stack")
plt.title("Age distribution by Product")
plt.show()
```




**Insights:**

**Marital Status & Gender:** Partnered customers buy more treadmills across all models; product choice also varies by gender.

**Income & Usage:** Higher-income, high-usage customers prefer KP781, while lower-income, low-usage customers prefer KP281.

**Age Factor:** KP281: Popular among younger buyers (20–30 years, budget-conscious). KP481: Attracts mid-aged buyers (25–35 years, balanced choice). KP781: Preferred by older, financially stable buyers (30–40 years, advanced features).

## 4. Representing the Probability

```
# Crosstab for Product counts
product_counts = pd.crosstab(index=df['Product'], columns='count')

# Marginal probabilities
product_probs = product_counts / product_counts.sum()

# Combine into one table
marginal_table = pd.concat([product_counts, product_probs.rename(columns={'count':'marginal_prob'})], axis=1)
marginal_table
```

| col_0 | count | marginal_prob |
|---|---|---|
| Product | | |
| **KP281** | 80 | 0.444444 |
| **KP481** | 60 | 0.333333 |
| **KP781** | 40 | 0.222222 |

Next steps: Generate code with `marginal_table` | View recommended plots | New interactive sheet

Find the probability that the customer buys a product based on each column.

Product vs Gender

```
# Contingency table (counts)
ct_gender = pd.crosstab(df['Product'], df['Gender'])

# P(Product | Gender)
p_prod_given_gender = ct_gender.div(ct_gender.sum(axis=0), axis=1)
print("P(Product | Gender):\n", p_prod_given_gender)
```

```
P(Product | Gender):
 Gender     Female      Male
Product
KP281    0.526316  0.384615
KP481    0.381579  0.298077
KP781    0.092105  0.317308
```

product vs marital status

```
ct_marital = pd.crosstab(df['Product'], df['MaritalStatus'])

# P(Product | Marital Status)
p_prod_given_marital = ct_marital.div(ct_marital.sum(axis=0), axis=1)
print("P(Product | MaritalStatus):\n", p_prod_given_marital)
```

```
P(Product | MaritalStatus):
 MaritalStatus  Partnered    Single
Product
KP281           0.448598  0.438356
KP481           0.336449  0.328767
KP781           0.214953  0.232877
```

Product vs Age Group (if you bin Age)

```
# Create age bins
df['AgeGroup'] = pd.cut(df['Age'], bins=[20,25,30,35,40,45], labels=['20-25','26-30','31-35','36-40','41-45'])

ct_age = pd.crosstab(df['Product'], df['AgeGroup'])

# P(Product | AgeGroup)
p_prod_given_age = ct_age.div(ct_age.sum(axis=0), axis=1)
print("P(Product | AgeGroup):\n", p_prod_given_age)
```

```
P(Product | AgeGroup):
 AgeGroup     20-25     26-30    31-35  36-40     41-45
Product
KP281     0.405797  0.512195  0.34375  0.500  0.500000
KP481     0.347826  0.170732  0.53125  0.375  0.166667
KP781     0.246377  0.317073  0.12500  0.125  0.333333
```

Find the conditional probability that an event occurs given that another event has occurred. (Example: given that a customer is female, what is the probability she'll purchase a KP481)

```
# Contingency table Product x Gender
ct_gender = pd.crosstab(df['Product'], df['Gender'])
print(ct_gender)

# Conditional probability: P(Product | Gender)
p_prod_given_gender = ct_gender.div(ct_gender.sum(axis=0), axis=1)
print("\nConditional probabilities P(Product | Gender):\n", p_prod_given_gender)

# Example: P(KP481 | Female)
if 'Female' in p_prod_given_gender.columns and 'KP481' in p_prod_given_gender.index:
    prob = p_prod_given_gender.loc['KP481','Female']
    print("\nP(KP481 | Female) =", round(prob,3))
```

```
Gender   Female  Male
Product
KP281        40    40
```

```
KP481        29    31
KP781         7    33

Conditional probabilities P(Product | Gender):
 Gender     Female      Male
Product
KP281    0.526316  0.384615
KP481    0.381579  0.298077
KP781    0.092105  0.317308

P(KP481 | Female) = 0.382
```

**Insights:**

KP281: 53% of females, 38% of males buy it → popular with both genders.

KP481: 38% of females, 30% of males buy it → slightly female-preferred.

KP781: 9% of females, 32% of males buy it → mostly male-dominated.

Marketing Tip: Focus KP781 campaigns on males; KP281 and KP481 can target both genders.

## 5. Check the correlation among different factors

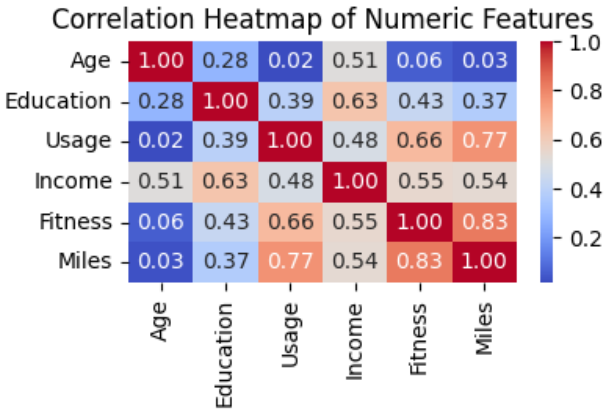
```
numeric_cols = ['Age', 'Education', 'Usage', 'Income', 'Fitness', 'Miles']
```

```
corr_matrix = df[numeric_cols].corr()
corr_matrix
```

| | Age | Education | Usage | Income | Fitness | Miles |
|---|---|---|---|---|---|---|
| **Age** | 1.000000 | 0.279533 | 0.015394 | 0.514362 | 0.057361 | 0.029636 |
| **Education** | 0.279533 | 1.000000 | 0.388802 | 0.628908 | 0.430480 | 0.367262 |
| **Usage** | 0.015394 | 0.388802 | 1.000000 | 0.481608 | 0.661978 | 0.771030 |
| **Income** | 0.514362 | 0.628908 | 0.481608 | 1.000000 | 0.546998 | 0.537297 |
| **Fitness** | 0.057361 | 0.430480 | 0.661978 | 0.546998 | 1.000000 | 0.826307 |
| **Miles** | 0.029636 | 0.367262 | 0.771030 | 0.537297 | 0.826307 | 1.000000 |

Next steps: Generate code with corr_matrix | View recommended plots | New interactive sheet

```
plt.figure(figsize=(4,2))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap of Numeric Features')
plt.show()
```




## 6. Customer Profiling and Recommendation

```
# Define income bins and labels
income_bins = [0, 30000, 60000, 90000, 120000, 150000, 200000]
```

```
income_labels = ['0-30k', '30k-60k', '60k-90k', '90k-120k', '120k-150k', '150k+']
df['IncomeGroup'] = pd.cut(df['Income'], bins=income_bins, labels=income_labels)
```

```
# Function to create profile per product
def customer_profile(product_df):
    avg_age = product_df['Age'].mean()
    common_gender = product_df['Gender'].mode()[0]  # Most frequent gender
    avg_income = product_df['Income'].mean()
    common_income_group = product_df['IncomeGroup'].mode()[0]  # Most frequent income group

    return pd.Series({
        'Average Age': round(avg_age, 1),
        'Most Common Gender': common_gender,
        'Average Income': round(avg_income, 0),
        'Most Common Income Group': common_income_group
    })
```

```
# Apply function to each product
# Group by product and create profiles
product_profiles = df.groupby('Product').apply(customer_profile).reset_index()
product_profiles
```

```
/tmp/ipython-input-869690630.py:3: DeprecationWarning: DataFrameGroupBy.apply operated on the grouping columns. This b
  product_profiles = df.groupby('Product').apply(customer_profile).reset_index()
```

| | Product | Average Age | Most Common Gender | Average Income | Most Common Income Group |
|---|---|---|---|---|---|
| 0 | KP281 | 28.4 | Female | 46584.0 | 30k-60k |
| 1 | KP481 | 28.8 | Male | 49047.0 | 30k-60k |
| 2 | KP781 | 28.8 | Male | 73908.0 | 60k-90k |

Next steps: Generate code with `product_profiles` | View recommended plots | New interactive sheet

## Customer Profile Recommendations

**KP281 (Entry-level):** Target females, mid-income (30k–60k). Highlight affordability and beginner-friendly features.

**KP481 (Mid-level):** Target males, mid-income (30k–60k). Emphasize intermediate features and performance.

**KP781 (Advanced):** Target males, higher-income (60k–90k). Focus on premium features and durability.

**Pricing & Promotions:** Offer discounts or bundles for mid-income products. Showcase premium benefits for high-income products.

**Inventory & Cross-sell:** Stock according to customer profile. Suggest complementary products like accessories or fitness trackers.