In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use("ggplot")
import warnings
warnings.filterwarnings("ignore")
pd.set_option('display.float_format', lambda x: '%.3f' % x)
```

In [2]:
```
cardiofit = pd.read_csv("CardioGoodFitness.csv")
```

In [3]:
```
cardiofit.head(10)
```

Out[3]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	TM195	18	Male	14	Single	3	4	29562	112
1	TM195	19	Male	15	Single	2	3	31836	75
2	TM195	19	Female	14	Partnered	4	3	30699	66
3	TM195	19	Male	12	Single	3	3	32973	85
4	TM195	20	Male	13	Partnered	4	2	35247	47
5	TM195	20	Female	14	Partnered	3	3	32973	66
6	TM195	21	Female	14	Partnered	3	3	35247	75
7	TM195	21	Male	13	Single	3	3	32973	85
8	TM195	21	Male	15	Single	5	4	35247	141
9	TM195	21	Female	15	Partnered	2	3	37521	85

Looking at the first 10 rows, I noticed that single males expect to put in the most miles followed by partnered females, then partnered males. Let's see if this trend holds up.

In [4]:
```
cardiofit.shape
```

Out[4]:
```
(180, 9)
```

- 180 rows and 9 columns

In [5]:
```
cardiofit.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
```

Age and Fitness are technically categories, so I will change these columns' Dtype.

In [6]:
```
cardiofit.describe()
```

Out[6]:

	Age	Education	Usage	Fitness	Income	Miles
count	180.000	180.000	180.000	180.000	180.000	180.000
mean	28.789	15.572	3.456	3.311	53719.578	103.194
std	6.943	1.617	1.085	0.959	16506.684	51.864
min	18.000	12.000	2.000	1.000	29562.000	21.000
25%	24.000	14.000	3.000	3.000	44058.750	66.000
50%	26.000	16.000	3.000	3.000	50596.500	94.000
75%	33.000	16.000	4.000	4.000	58668.000	114.750
max	50.000	21.000	7.000	5.000	104581.000	360.000

- Looking at the Age and Income ranges, I can see that there is a good amount of diversity in the customers.
- The average age is 28-29.
- Average income is 53,719. That seems about right given the average age.
- Average expected miles is 103. That seems reasonable given average age and fitness level.

In [7]:
```
cardiofit.Age = cardiofit.Age.astype('category')
```

In [8]:
```
cardiofit.Fitness = cardiofit.Fitness.astype('category')
```

In [9]:
```
cardiofit.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    category
 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    category
 7   Income         180 non-null    int64   
 8   Miles          180 non-null    int64   
dtypes: category(2), int64(4), object(3)
memory usage: 12.0+ KB
```

Age and Fitness are now categories.

In [10]:
```
cardiofit.columns
```

Out[10]:
```
Index(['Product', 'Age', 'Gender', 'Education', 'MaritalStatus', 'Usage',
       'Fitness', 'Income', 'Miles'],
      dtype='object')
```

Just printing out the columns above.

In [11]:
```
cardiofit.skew()
```

Out[11]:
```
Education   0.622
Usage       0.739
Income      1.292
Miles       1.724
dtype: float64
```

Checked the skews to gain insight on what I can expect when visualizing distributions.

-

Going to check for missing values now.

In [12]:
```
def missing_check(df):
    total = df.isnull().sum().sort_values(ascending=False)
    percent = (df.isnull().sum()/df.isnull().count()).sort_values(ascending=Fals
    missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent']
    return missing_data
missing_check(cardiofit)
```

Out[12]:

	Total	Percent
Miles	0	0.000
Income	0	0.000
Fitness	0	0.000
Usage	0	0.000
MaritalStatus	0	0.000
Education	0	0.000
Gender	0	0.000
Age	0	0.000
Product	0	0.000

There appears to be no missing values!

-

Going to make frequency tables now for Age, Gender, Product, MaritalStatus, Usage, Fitness, and Education.

In [13]:
```
my_tab = pd.crosstab(index=cardiofit["Age"],
                     columns="count")
my_tab
```

Out[13]:

col_0	count
Age	
18	1
19	4
20	5
21	7
22	7
23	18
24	12
25	25
26	12
27	7
28	9
29	6
30	7
31	6
32	4
33	8
34	6
35	8
36	1
37	2
38	7
39	1
40	5
41	1
42	1
43	1
44	1
45	2
46	1
47	2
48	2
50	1

- A large portion of customers are of ages 23-26.

In [14]:
```
my_tab = pd.crosstab(index=cardiofit["Gender"],
                     columns="count")
my_tab
```

Out[14]:

Gender	col_0 count
Female	76
Male	104

- The customer base is mostly male.

In [15]:
```
my_tab = pd.crosstab(index=cardiofit["Product"],
                     columns="count")
my_tab
```

Out[15]:

Product	col_0 count
TM195	80
TM498	60
TM798	40

- The best selling model is the TM195, followed by the 498 and lastly the 798.

In [16]:
```
my_tab = pd.crosstab(index=cardiofit["MaritalStatus"],
                     columns="count")
my_tab
```

Out[16]:

MaritalStatus	col_0 count
Partnered	107
Single	73

- Most customers are partnered.

In [17]:
```
my_tab = pd.crosstab(index=cardiofit["Usage"],
                     columns="count")
my_tab
```

Out[17]:

Usage	col_0 count
2	33
3	69
4	52

Usage	col_0 count
5	17
6	7
7	2

- It seems most customers want to use their treadmill at least 3-4 times a week.

In [18]:
```
my_tab = pd.crosstab(index=cardiofit["Fitness"],
                     columns="count")
my_tab
```

Out[18]:

Fitness	col_0 count
1	2
2	26
3	97
4	24
5	31

- Again, most customers consider themselves about average fitness(3).

In [19]:
```
my_tab = pd.crosstab(index=cardiofit["Education"],
                     columns="count")
my_tab
```

Out[19]:

Education	col_0 count
12	3
13	5
14	55
15	5
16	85
18	23
20	1
21	3

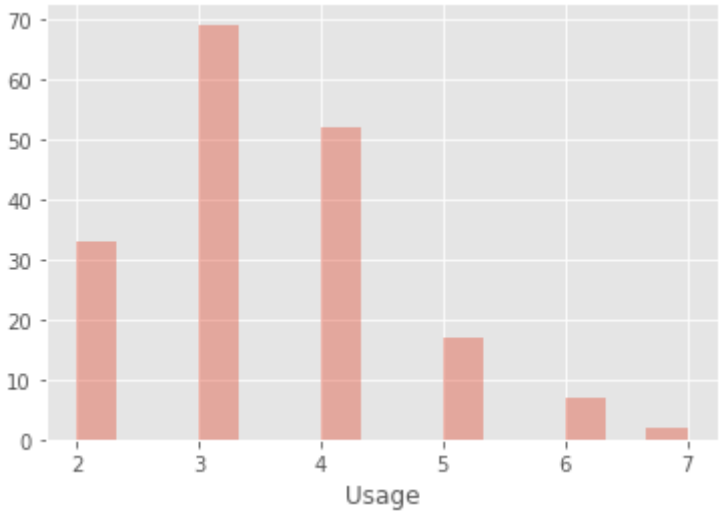
- It looks like most customers have a bachelor's degree followed by an associate degree.

-

In [20]:
```
sns.distplot(cardiofit['Age']);
```

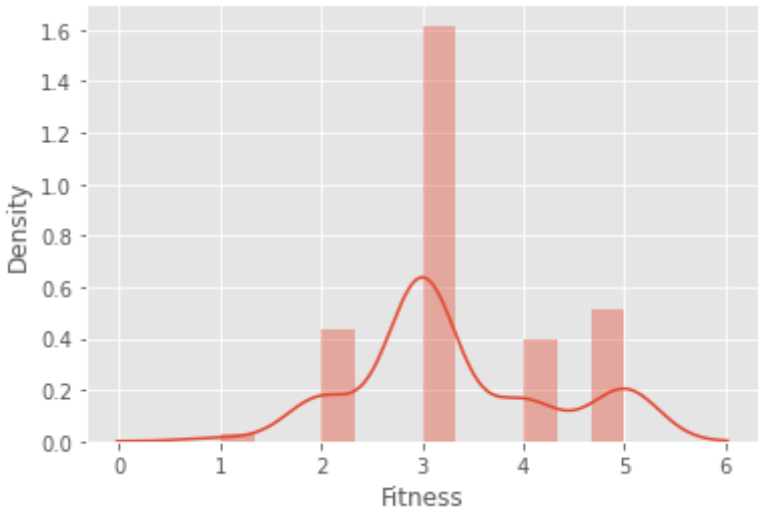
Positive skewed distribution for Age.

In [21]:
```
sns.distplot(cardiofit['Usage'], kde = False);
```



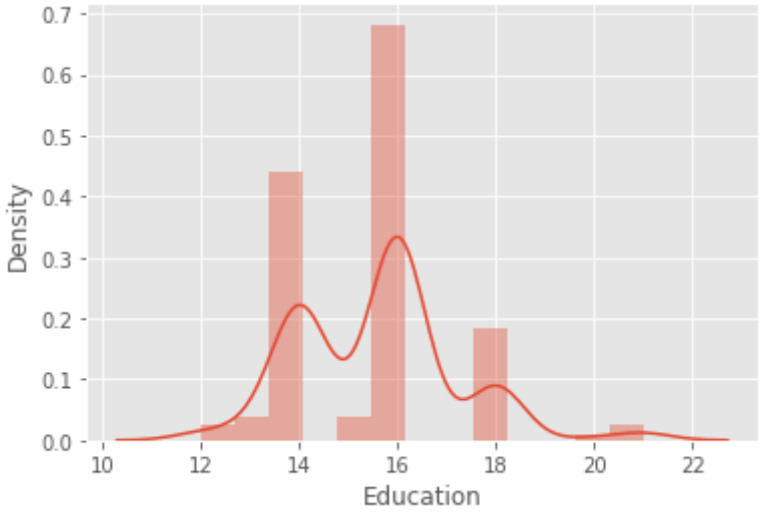
Positive skewed distribution for Usage.

In [22]:
```
sns.distplot(cardiofit['Fitness']);
```



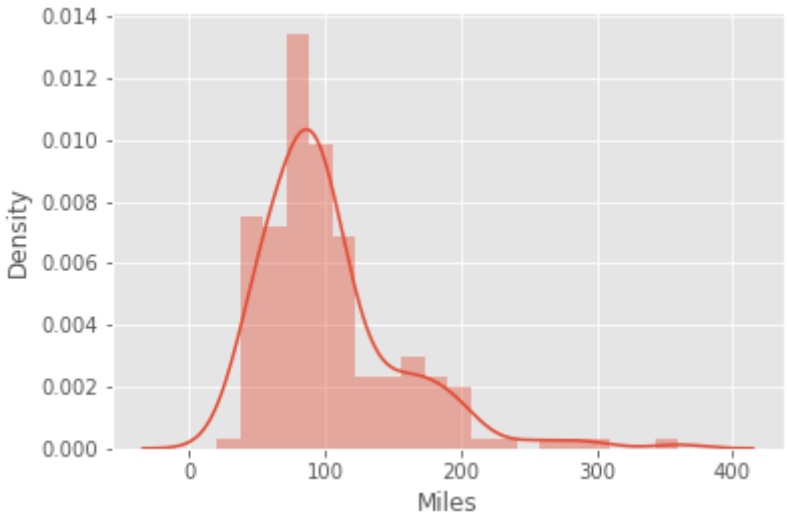
A slight symmetrical skew for Fitness.

In [23]:
```
sns.distplot(cardiofit['Education']);
```



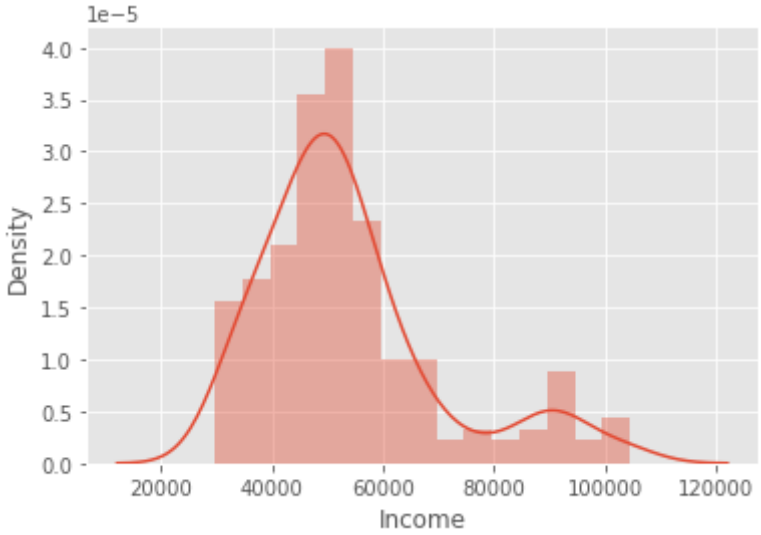
Slight positive skew for Education.

In [24]:
```
sns.distplot(cardiofit['Miles']);
```



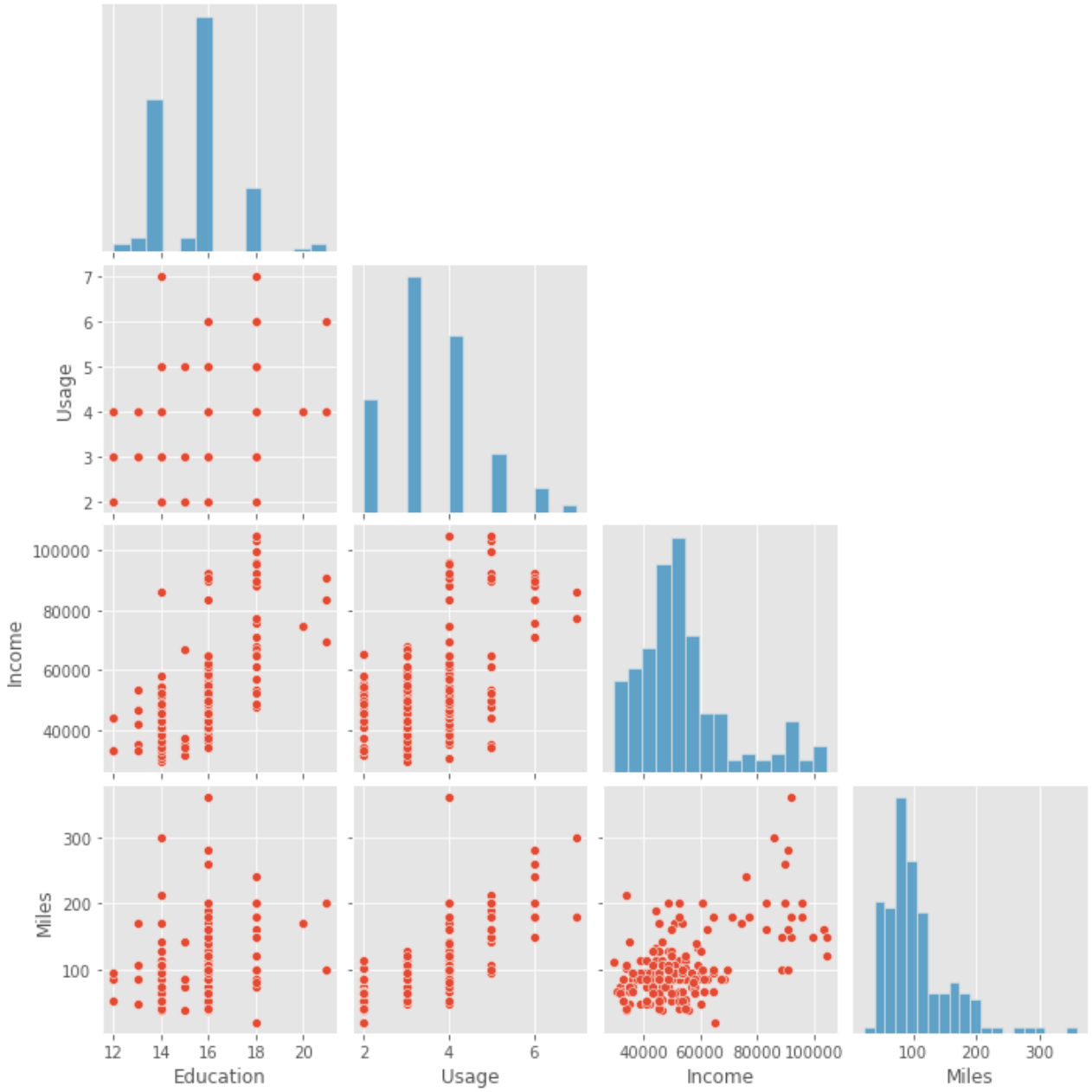
A clear positive skewed distribution for Miles.

In [25]:
```
sns.distplot(cardiofit['Income']);
```



A positive skewed distribution for Income.

In [26]:
```
sns.pairplot(cardiofit, corner = True);
```



In [27]:
```
correlation = cardiofit.corr()
correlation
```

Out[27]:

	Education	Usage	Income	Miles
Education	1.000	0.395	0.626	0.307
Usage	0.395	1.000	0.520	0.759
Income	0.626	0.520	1.000	0.543
Miles	0.307	0.759	0.543	1.000

In [28]:
```
sns.heatmap(correlation, annot=True, cmap='YlGnBu', vmin=-1, vmax=1);
```

In [29]:
```
cardiofit.Age = cardiofit.Age.astype('int64')
```

In [30]:
```
cardiofit.Fitness = cardiofit.Fitness.astype('int64')
```

In [31]:
```
correlation = cardiofit.corr()
correlation
```

Out[31]:

	Age	Education	Usage	Fitness	Income	Miles
Age	1.000	0.280	0.015	0.061	0.513	0.037
Education	0.280	1.000	0.395	0.411	0.626	0.307
Usage	0.015	0.395	1.000	0.669	0.520	0.759
Fitness	0.061	0.411	0.669	1.000	0.535	0.786
Income	0.513	0.626	0.520	0.535	1.000	0.543
Miles	0.037	0.307	0.759	0.786	0.543	1.000

In [32]:
```
sns.heatmap(correlation, annot=True, cmap='YlGnBu', vmin=-1, vmax=1);
```

- Fitness v Miles has highest correlation.
- Usage v Miles has second highest correlation.

- Usage v Fitness has third highest correlation.

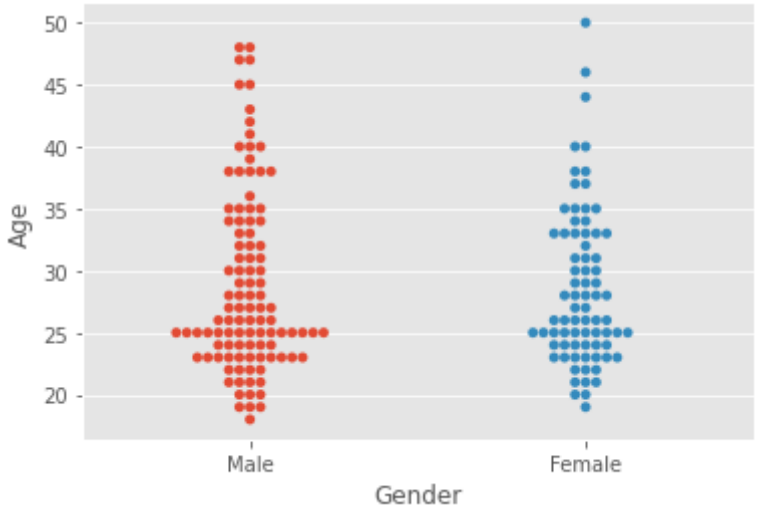
In [33]:
```
sns.pairplot(cardiofit, corner = True);
```

In [34]:
```
cardiofit.Age = cardiofit.Age.astype('category')
```

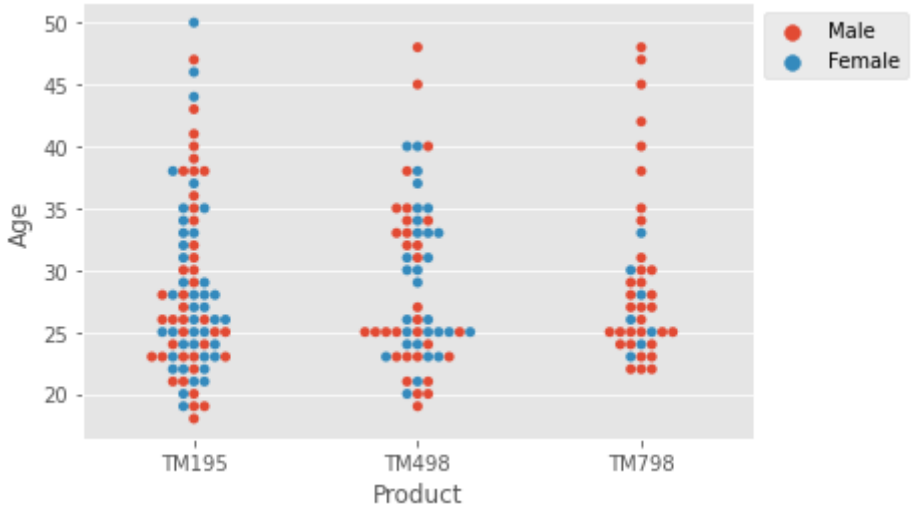
In [35]:
```
cardiofit.Fitness = cardiofit.Fitness.astype('category')
```

Briefly converted Age and Fitness to int64 to see more correlations in the pairplot and heatmap.

In [36]:
```
sns.swarmplot(cardiofit['Gender'], cardiofit['Age']);
```

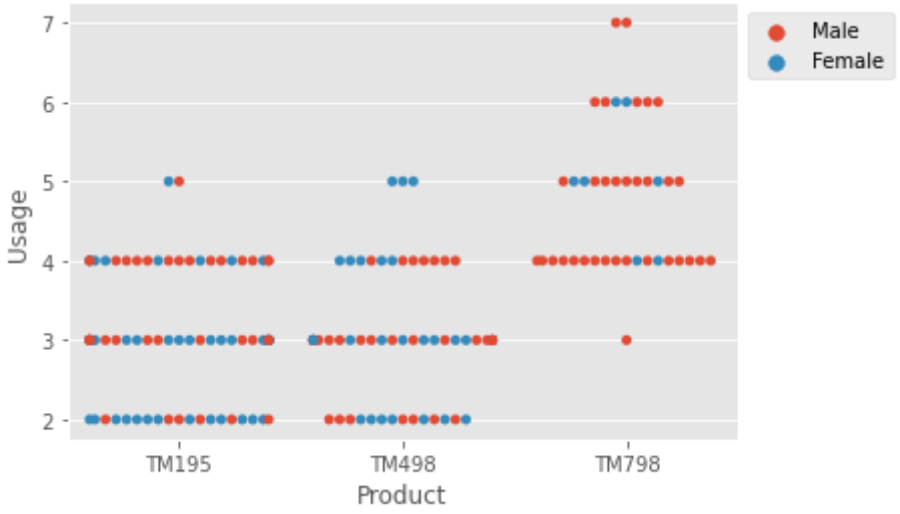


In [37]:
```
sns.swarmplot(cardiofit['Product'], cardiofit['Age'], hue=cardiofit['Gender'])
plt.legend(bbox_to_anchor=(1, 1));
```



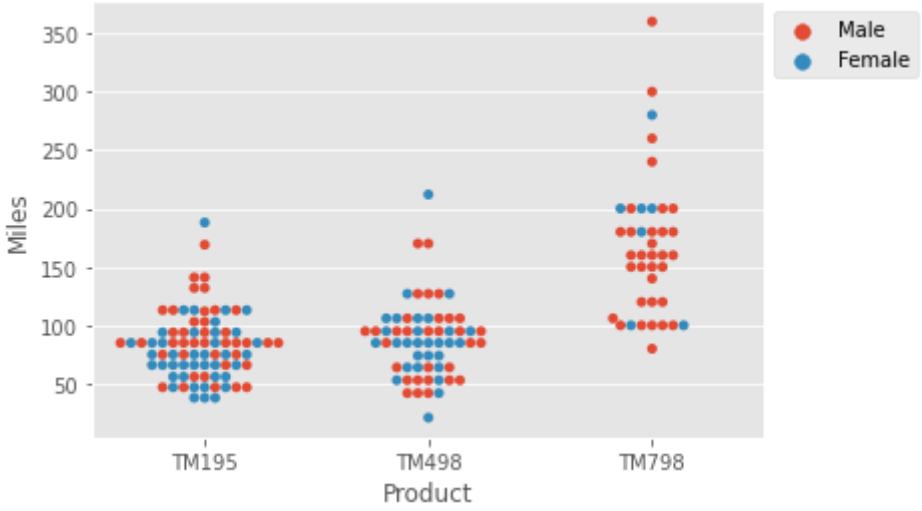
- TM195: 80 users, 40 females, 40 males
- TM498: 60 users, 29 females, 31 males
- TM798: 40 users, 7 females, 33 males
- The TM195 and TM498 both seem equally popular between genders, except the TM798, which is heavily favored by males.

In [38]:
```
sns.swarmplot(cardiofit['Product'], cardiofit['Usage'], hue=cardiofit['Gender'])
plt.legend(bbox_to_anchor=(1, 1));
```

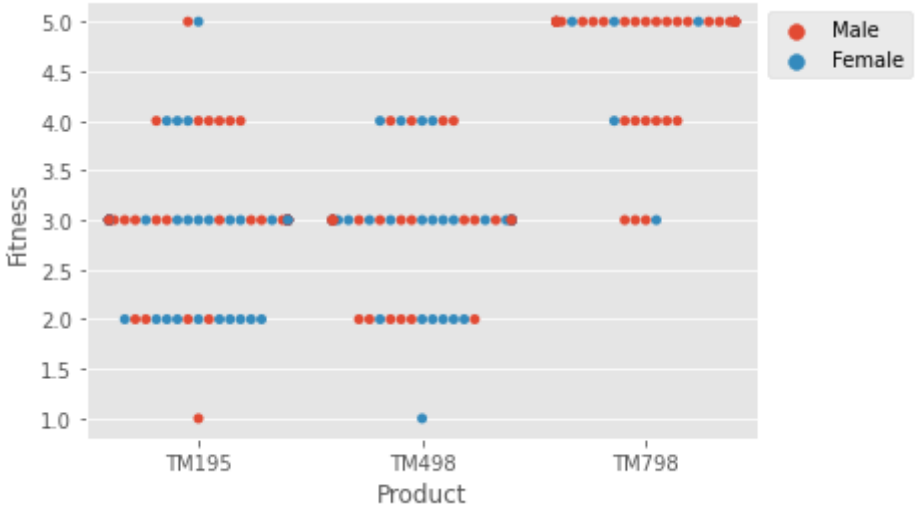


- The expected use of the TM798 looks to be higher than the other two models.

In [39]:
```
sns.swarmplot(cardiofit['Product'], cardiofit['Miles'], hue=cardiofit['Gender'])
plt.legend(bbox_to_anchor=(1, 1));
```

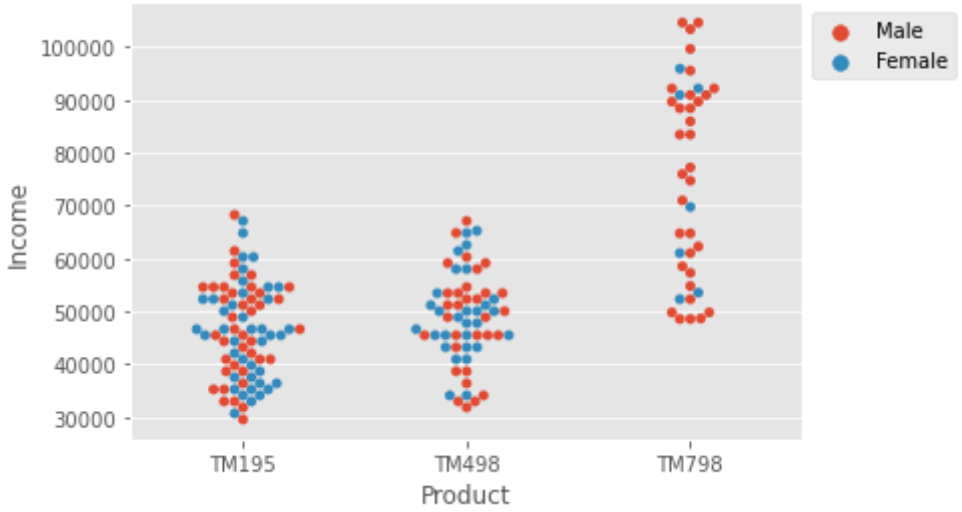


In [40]:
```
sns.swarmplot(cardiofit['Product'], cardiofit['Fitness'], hue=cardiofit['Gender'
plt.legend(bbox_to_anchor=(1, 1));
```

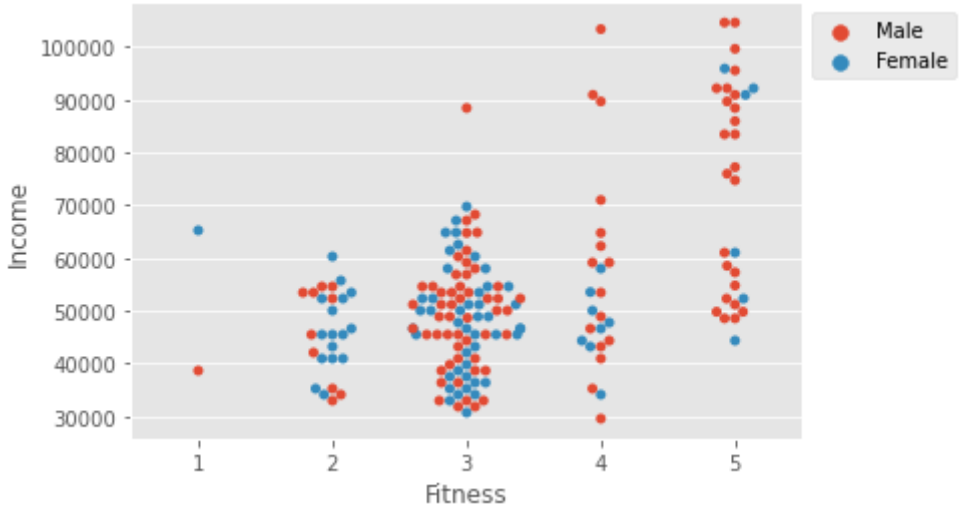


- The TM798 seems to be the prefered model for the fitness enthusiast.

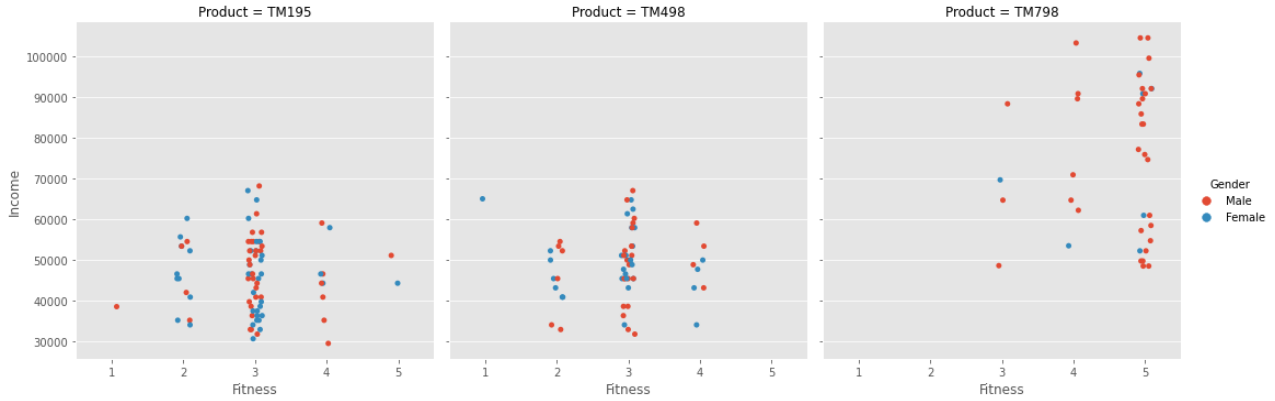
In [41]:
```
sns.swarmplot(cardiofit['Product'], cardiofit['Income'], hue=cardiofit['Gender'])
plt.legend(bbox_to_anchor=(1, 1));
```



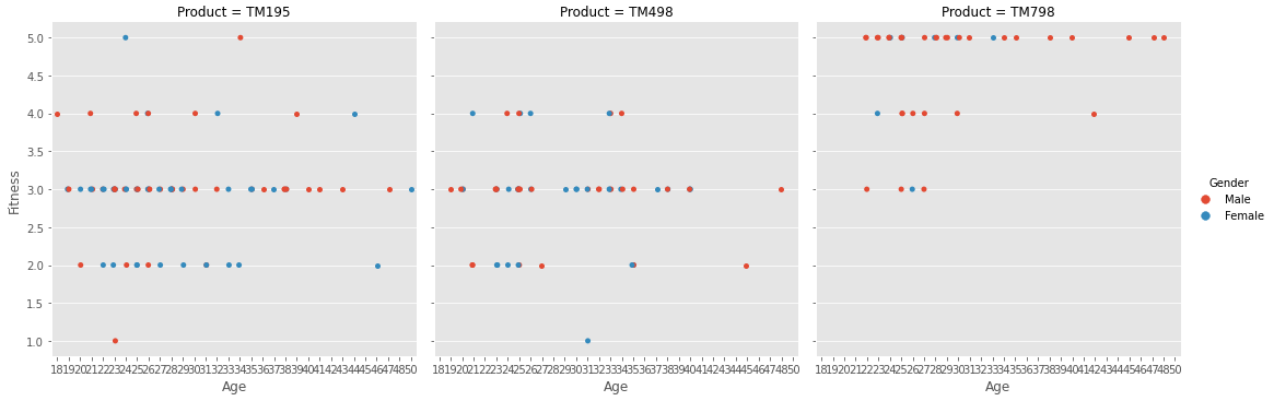
In [42]:
```
sns.swarmplot(cardiofit['Fitness'], cardiofit['Income'], hue=cardiofit['Gender'])
plt.legend(bbox_to_anchor=(1, 1));
```



In [43]:
```
sns.catplot(x="Fitness",
            y="Income",
            hue="Gender",
            col="Product",
            data=cardiofit,
            kind="strip");
```

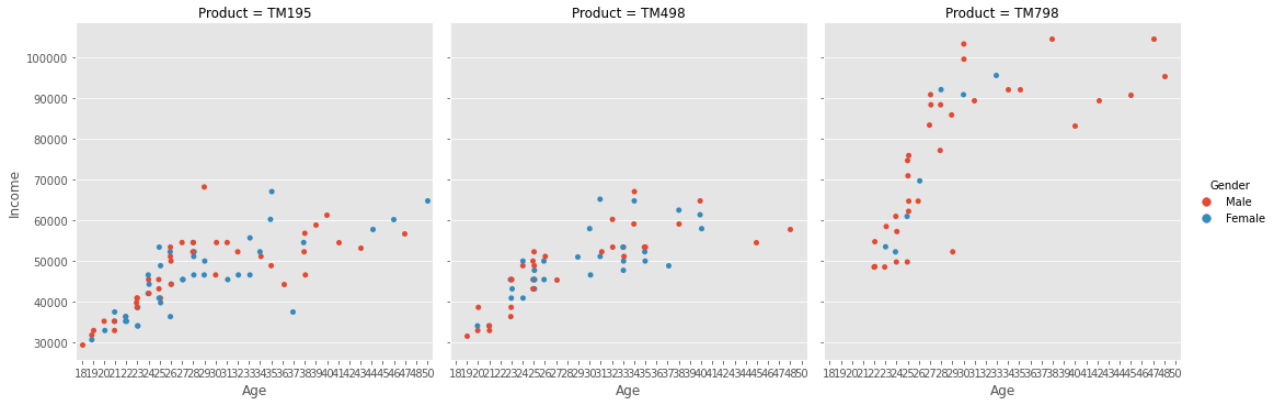


In [44]:
```
sns.catplot(x="Age",
            y="Fitness",
            hue="Gender",
            col="Product",
            data=cardiofit,
            kind="strip");
```



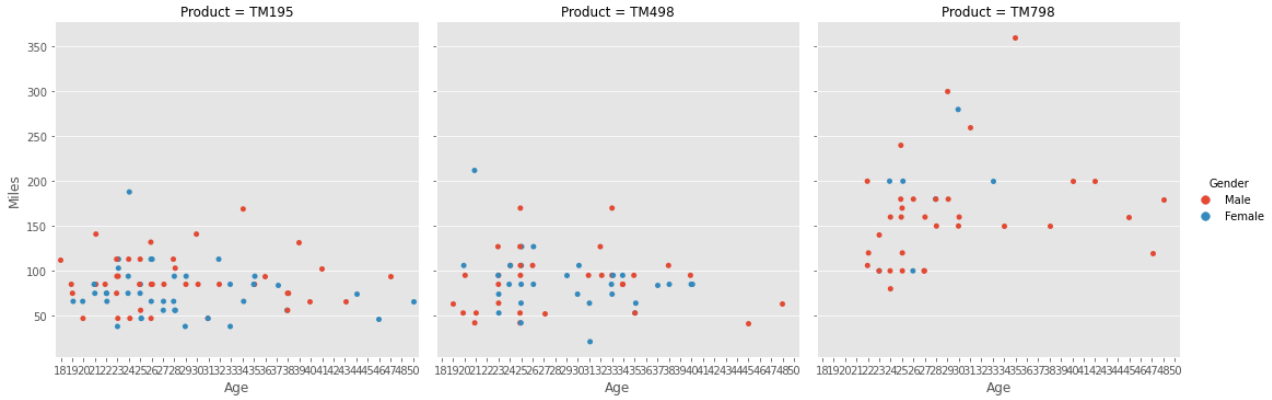
- More guys consider themselves to be very fit.

In [45]:
```
sns.catplot(x="Age",
            y="Income",
            hue="Gender",
            col="Product",
            data=cardiofit,
            kind="strip");
```



In [46]:
```
sns.catplot(x="Age",
            y="Miles",
            hue="Gender",
```

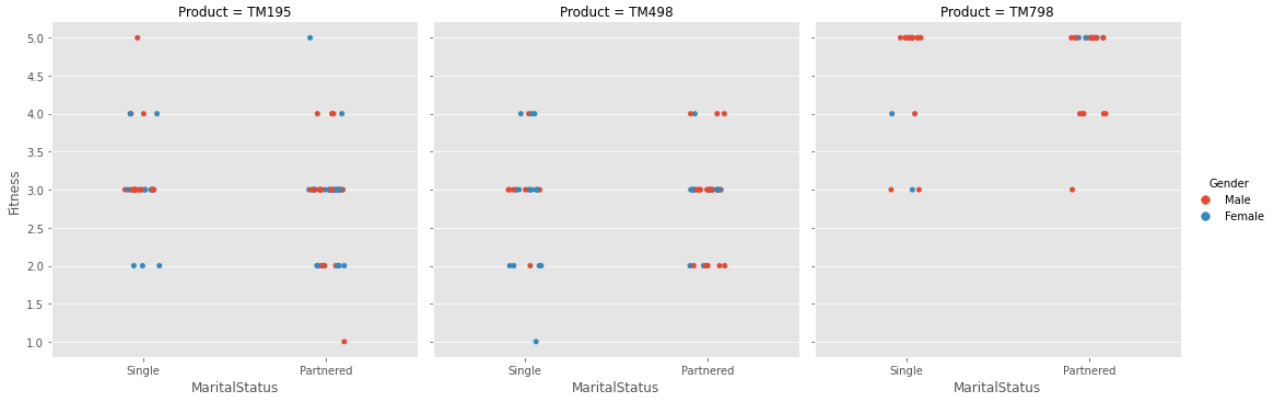
```
            col="Product",
            data=cardiofit,
            kind="strip");
```



- The older males who have higher incomes seems to have higher percieved fitness levels.
- The higher income males tend to buy the TM798.
- Having a higher income seems to correlate with having a higer fitness level. Other factors like quality of food may play a part in percieved fitness as well.
- It does look like those who have a high percieved fitness do plan to put in more miles.
- From what I gather, I am assuming the TM798 is the more expensive model, followed by the TM498 and lastly the TM195.

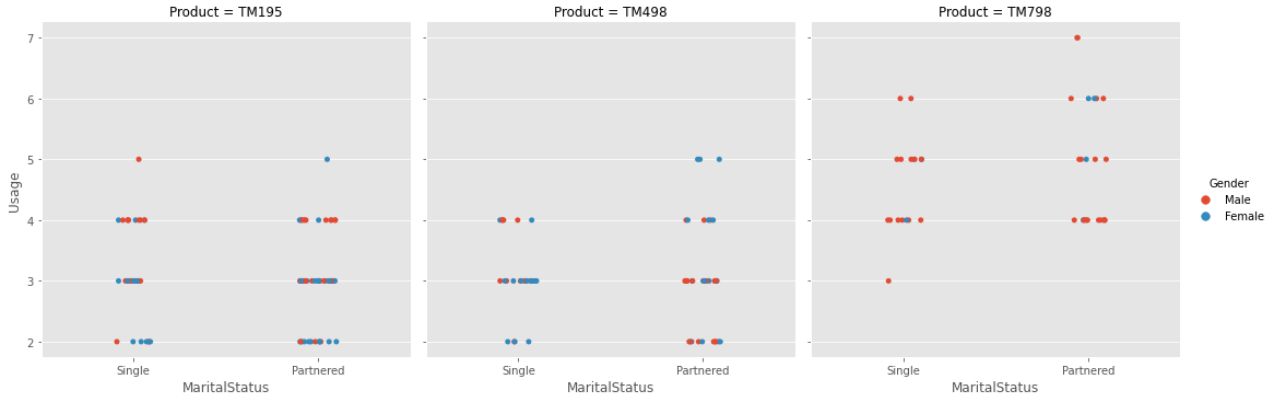
In [47]:
```
cardiofit.Fitness = cardiofit.Fitness.astype('int64')
```

In [48]:
```
sns.catplot(x="MaritalStatus",
            y="Fitness",
            hue="Gender",
            col="Product",
            data=cardiofit,
            kind="strip");
```

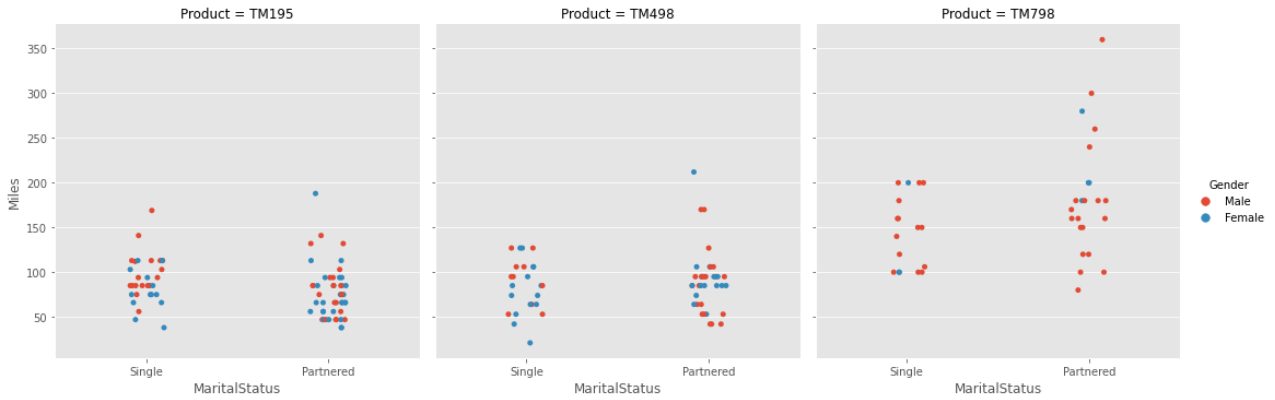


In [49]:
```
sns.catplot(x="MaritalStatus",
            y="Usage",
            hue="Gender",
            col="Product",
```

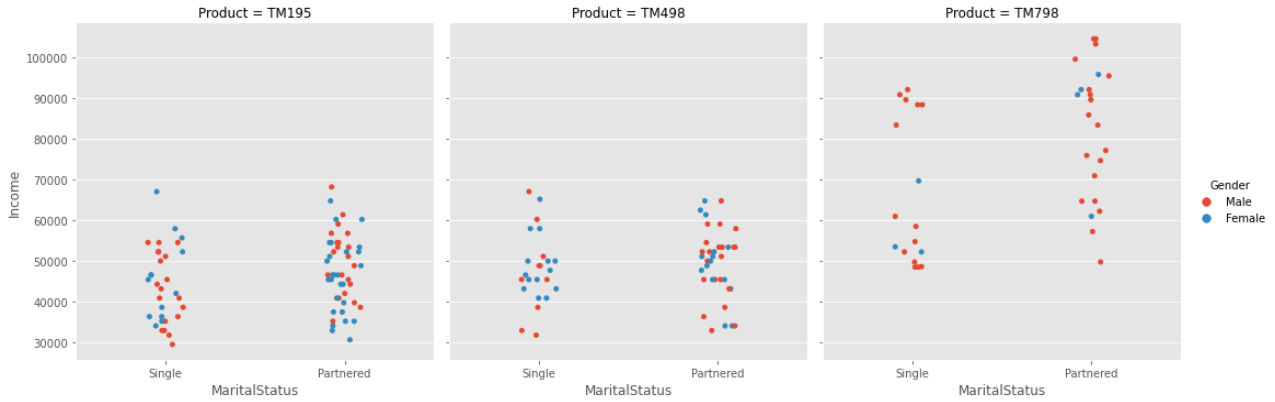
```
            data=cardiofit,
            kind="strip");
```



In [50]:
```
sns.catplot(x="MaritalStatus",
            y="Miles",
            hue="Gender",
            col="Product",
            data=cardiofit,
            kind="strip");
```

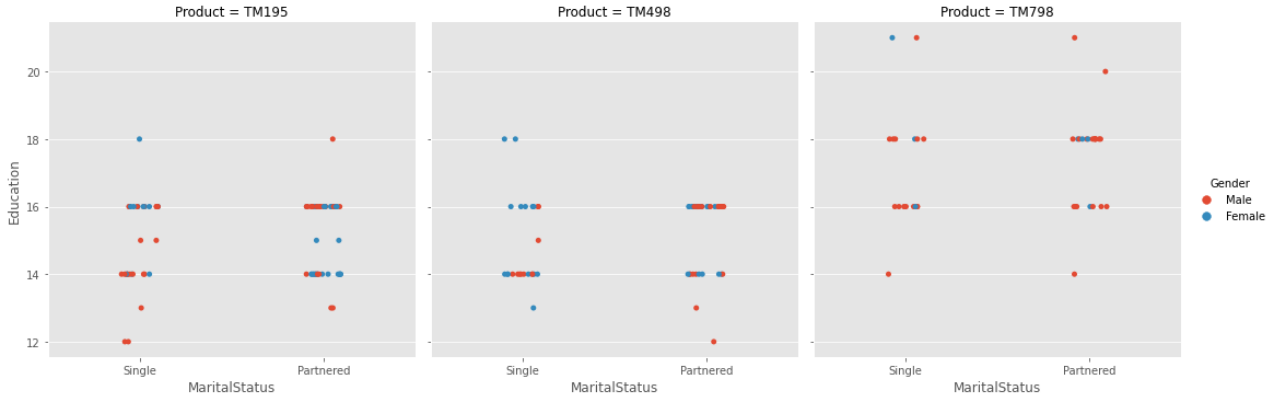


In [51]:
```
sns.catplot(x="MaritalStatus",
            y="Income",
            hue="Gender",
            col="Product",
            data=cardiofit,
            kind="strip");
```

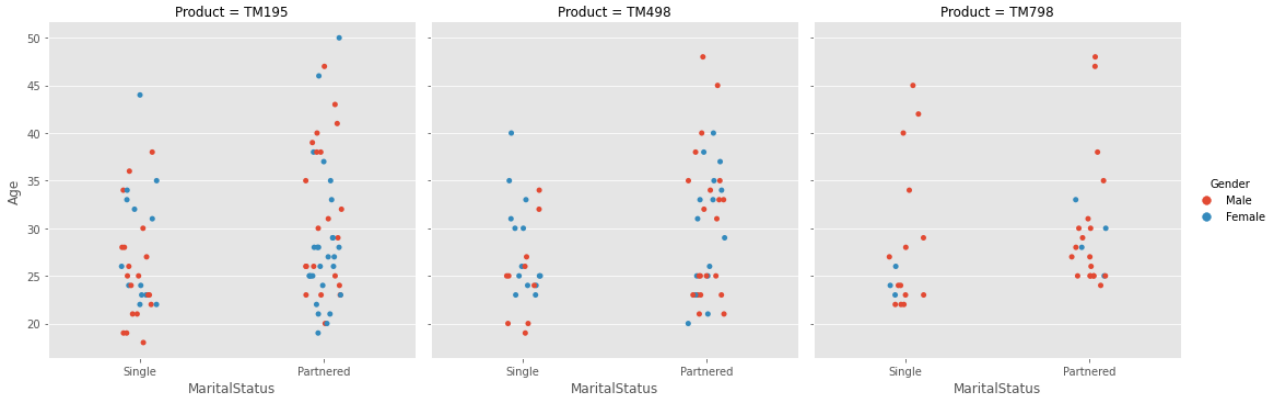


In [52]:
```
sns.catplot(x="MaritalStatus",
            y="Education",
            hue="Gender",
```

```
            col="Product",
            data=cardiofit,
            kind="strip");
```



In [53]:
```
sns.catplot(x="MaritalStatus",
            y="Age",
            hue="Gender",
            col="Product",
            data=cardiofit,
            kind="strip");
```



- It looks like marital status doesn't play much of a factor.

In [54]:
```
import pandas_profiling

pandas_profiling.ProfileReport(cardiofit, title='Pandas Profiling Report')
```

Overview

Dataset statistics

Number of variables	9
Number of observations	180
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	13.1 KiB
Average record size in memory	74.2 B

Variable types

NUM	5
CAT	4

Reproduction

Analysis started	2021-06-19 05:11:15.103377
Analysis finished	2021-06-19 05:11:23.303671
Duration	8.2 seconds

Out[54]:

Conclusion/Recommendation

- Those with incomes over 70,000 buy the TM798.
- Most people who expect to put in 100 miles or more get the TM798.

- People who consider themselves very fit (5) go with the TM798.
- The average person generally has the TM196 or TM498.
- Marital status seems to not be a differentiating factor.
- There is a lack of customers over the age of 33 and who make 70,000 or more.
- Maybe producing a new treadmill model that fits between the TM498 and TM798. Maybe it will be of high quality material/structure and performance, but without the features aimed at fitness enthusiasts.
- Maybe a new tradmill model that is easily movable, compact, and affordable. I think something like this will bring in people who want to start or are interested in exercising. Probably people that would be considered in the 1-2 fitness level.