In [1]:
```
#imports
import shap
import pandas as pd
import xgboost as xgb
import statsmodels.api as sm
import matplotlib.pyplot as plt
from sklearn.inspection import permutation_importance
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
```

In [2]:
```
CONSTRAINT = 0.3

def bars(shap_values):
    # summarize the SHAP values for each feature
    shap.summary_plot(shap_values, X, plot_type='bar', show=False)

    # plot the SHAP values for each feature
    fig, ax = plt.gcf(), plt.gca()
    ax.set_xlim(-0.5, 2)
    ax.set_title("Food Group Impact")
    plt.show()
```

In [66]:
```
# select user input
user_input = int(input("Which user's result do you want to look at?: "))

# read the dataframe
data = pd.read_csv('data.csv')
df = data[data['user_number'] == user_input]

# drop all Null data (filtering null values)
df.dropna()
```

```
Which user's result do you want to look at?: 9
```

Out[66]:

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	...	F13	F14	F15	F16	F17	F18	F19	F20	sympto
230	9	16	0	0	27	13	4	40	38	5	...	26	16	0	0	29	2	12	16	(
231	8	10	4	0	34	18	0	40	48	10	...	24	0	8	18	30	0	11	5	1
232	0	16	0	0	23	16	2	42	40	9	...	11	0	0	9	29	5	0	6	(
233	0	14	3	0	42	9	4	31	28	8	...	19	0	6	15	23	2	7	3	(
234	8	0	9	2	42	14	5	8	52	7	...	17	16	11	11	27	0	8	27	2
235	5	0	0	1	28	0	4	28	42	0	...	0	15	11	10	14	4	12	20	(
236	0	12	7	3	38	0	2	38	29	0	...	11	13	0	22	9	2	8	29	(
237	11	0	13	4	29	21	3	21	58	4	...	0	17	14	13	43	1	8	34	9
238	0	13	10	0	40	13	5	30	51	6	...	16	20	9	21	13	1	8	34	3
239	0	16	16	3	51	11	9	34	26	6	...	15	17	6	19	19	1	15	43	9
240	0	17	13	4	38	20	8	32	56	5	...	15	14	6	10	25	2	7	32	8
241	0	15	3	0	26	9	8	39	56	6	...	0	0	11	0	17	2	0	6	(
242	10	15	13	2	48	12	3	40	51	5	...	15	20	7	0	28	3	10	35	6
243	7	19	13	3	44	20	0	30	59	7	...	17	0	14	0	37	2	10	17	5
244	5	11	0	3	24	12	9	30	42	0	...	19	11	5	0	27	1	0	14	(
245	0	9	8	2	48	0	0	38	17	8	...	17	19	0	0	15	2	13	32	(
246	0	16	10	1	49	15	7	46	49	11	...	21	12	10	17	23	1	10	23	9
247	0	16	8	2	37	16	0	48	41	5	...	0	0	0	0	16	0	8	10	(
248	0	14	0	1	13	0	5	29	41	0	...	0	12	10	10	0	2	0	13	(
249	4	13	0	0	39	18	7	40	37	8	...	20	14	0	12	29	3	9	14	3
250	4	7	7	2	38	18	4	37	50	7	...	20	0	11	0	33	3	8	10	2
251	5	11	13	0	41	11	5	40	45	11	...	0	0	6	11	25	0	0	13	4
252	8	0	2	0	23	11	0	31	29	6	...	15	13	0	0	34	3	0	15	(
253	10	23	11	2	50	0	5	57	41	7	...	0	17	11	10	18	1	14	32	7
254	7	0	0	0	24	9	0	33	36	3	...	10	21	6	11	24	1	0	21	(
255	0	11	6	0	46	10	3	42	42	5	...	14	0	0	6	18	2	12	8	(
256	5	11	8	2	42	16	3	27	48	7	...	16	0	0	11	29	2	11	15	1
257	0	0	12	0	32	23	4	22	44	0	...	10	15	0	15	31	2	0	27	1
258	5	0	11	3	27	0	3	24	27	6	...	15	0	0	0	15	0	0	14	(
259	8	0	5	1	36	15	3	24	36	3	...	0	14	0	14	32	3	10	21	(
260	2	7	8	1	23	18	7	19	45	4	...	21	17	7	11	23	2	10	29	2
261	5	14	12	5	49	19	1	49	50	4	...	13	0	10	0	34	5	13	23	7

32 rows × 22 columns

In [67]:
```
# feature seletion to determine correllation between the columnn and symtpom val
# using corrwidth to compare all values of columns F1-F20 to the symptom_value
# for each data point. Using method - "Spearman Correlation"
correlation = df.iloc[:,:-2].corrwith(df['symptom_value'],method='spearman').abs()
# filtering all food categories with vorrelation value >0.3 and indexing to get names
high_corr_categories = correlation[correlation>CONSTRAINT].index.tolist()

# print the dataframe
df2 = df[high_corr_categories]
df2 = df2.join(df.iloc[:,-2:])
print("Highest affecting food group categories:", high_corr_categories)
print(df2)
```

```
Highest affecting food group categories: ['F2', 'F3', 'F4', 'F5', 'F6', 'F7', 'F9',
'F11', 'F12', 'F15', 'F17', 'F19', 'F20']
     F2  F3  F4  F5  F6  F7  F9  F11  F12  F15  F17  F19  F20  symptom_value  \
230  16   0   0  27  13   4  38    0    0    0   29   12   16       0.000000
231  10   4   0  34  18   0  48    7    8    8   30   11    5      15.907488
232  16   0   0  23  16   2  40   11   16    0   29    0    6       0.000000
233  14   3   0  42   9   4  28    0   10    6   23    7    3       0.000000
234   0   9   2  42  14   5  52    7    0   11   27    8   27      28.719602
235   0   0   1  28   0   4  42   13   16   11   14   12   20       0.000000
236  12   7   3  38   0   2  29   11    8    0    9    8   29       0.000000
237   0  13   4  29  21   3  58    0   10   14   43    8   34      99.907087
238  13  10   0  40  13   5  51    7    0    9   13    8   34      35.636293
239  16  16   3  51  11   9  26    0    0    6   19   15   43      99.918674
240  17  13   4  38  20   8  56    9    0    6   25    7   32      80.515864
241  15   3   0  26   9   8  56   16    8   11   17    0    6       0.000000
242  15  13   2  48  12   3  51   10    0    7   28   10   35      65.025245
243  19  13   3  44  20   0  59    5    4   14   37   10   17      51.994957
244  11   0   3  24  12   9  42    0   14    5   27    0   14       0.000000
245   9   8   2  48   0   0  17    9   14    0   15   13   32       0.000000
246  16  10   1  49  15   7  49    2   15   10   23   10   23      99.995380
247  16   8   2  37  16   0  41   14   14    0   16    8   10       0.000000
248  14   0   1  13   0   5  41   11   15   10    0    0   13       0.000000
249  13   0   0  39  18   7  37    0    0    0   29    9   14      32.107742
250   7   7   2  38  18   4  50    0    9   11   33    8   10      23.532020
251  11  13   0  41  11   5  45    7   12    6   25    0   13       4.464153
252   0   2   0  23  11   0  29    5   11    0   34    0   15       0.000000
253  23  11   2  50   0   5  41    2    0   11   18   14   32      71.994961
254   0   0   0  24   9   0  36    0    8    6   24    0   21       0.000000
255  11   6   0  46  10   3  42    9   11    0   18   12    8       0.000000
256  11   8   2  42  16   3  48    9    0    0   29   11   15      10.627128
257   0  12   0  32  23   4  44    6    0    0   31    0   27      19.491361
258   0  11   3  27   0   3  27    0    0    0   15    0   14       0.000000
259   0   5   1  36  15   3  36   10   16    0   32   10   21       0.000000
260   7   8   1  23  18   7  45   11   10    7   23   10   29      22.537727
261  14  12   5  49  19   1  50    0   14   10   34   13   23      74.824253

     user_number
230            9
231            9
232            9
233            9
234            9
235            9
236            9
237            9
238            9
239            9
240            9
241            9
242            9
243            9
244            9
245            9
246            9
247            9
248            9
249            9
250            9
251            9
252            9
```

```
253           9
254           9
255           9
256           9
257           9
258           9
259           9
260           9
261           9
```

In [68]:
```
# split the dataset into training and test data
X = df.iloc[:,:-2]
y = df["symptom_value"]
```

In [69]:
```
# split the dataset into training and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

In [70]:
```
# create a DMatrix for XGBoost
dtrain = xgb.DMatrix(X_train, label=y_train)
```

In [71]:
```
# specify XGBoost parameters
params = {'max_depth': 3, 'eta': 0.1, 'objective': 'reg:squarederror'}

# train the model
model = xgb.train(params, dtrain)
```

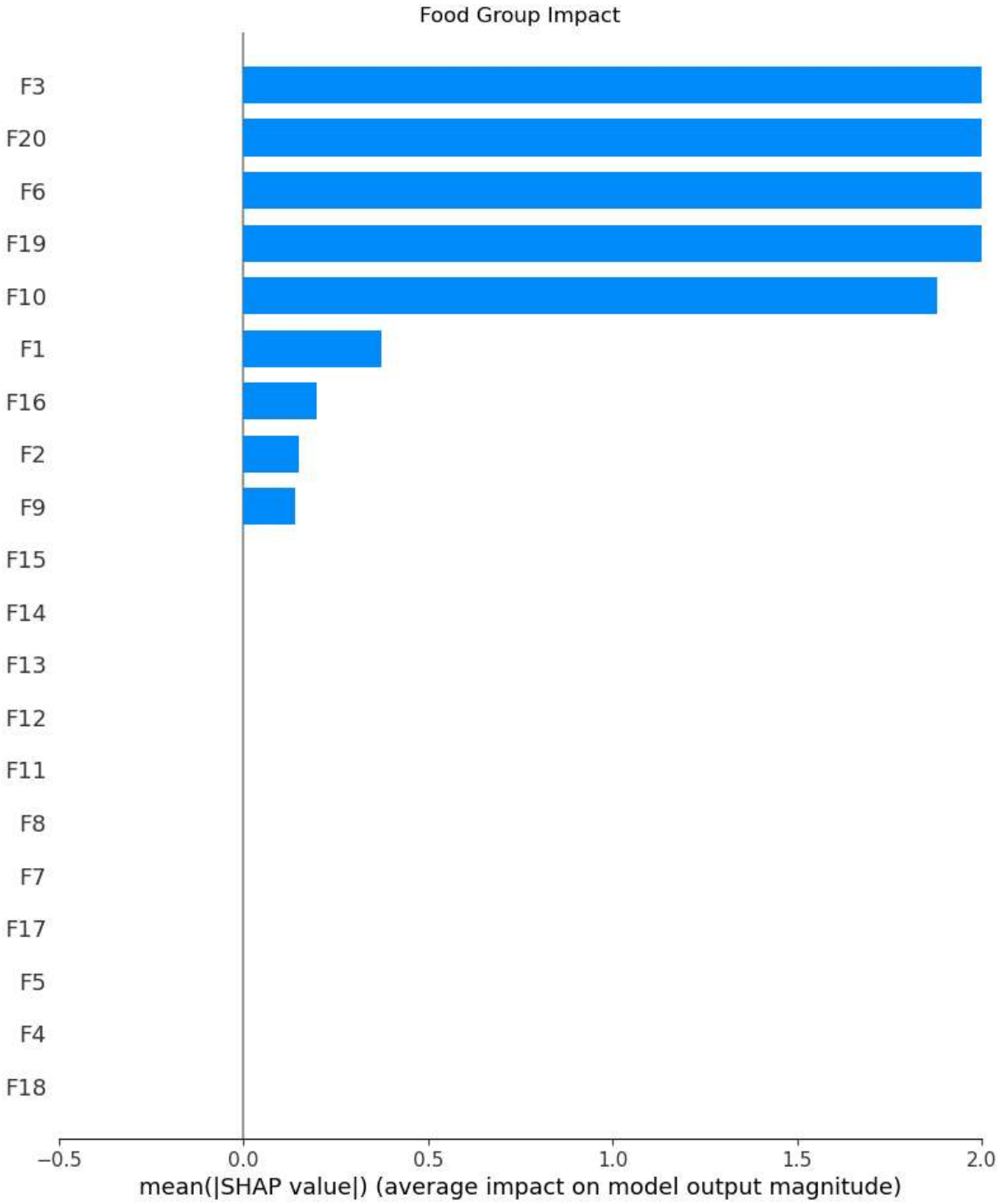
In [72]:
```
# make predictions on the test set
y_pred = model.predict(xgb.DMatrix(X_test))

# calculate R-squared score
r2 = r2_score(y_test, y_pred)
print("R-squared score:", r2)
```

```
R-squared score: 0.6302067752099298
```

In [73]:
```
# create an explainer object for SHAP
explainer = shap.Explainer(model, X)
```

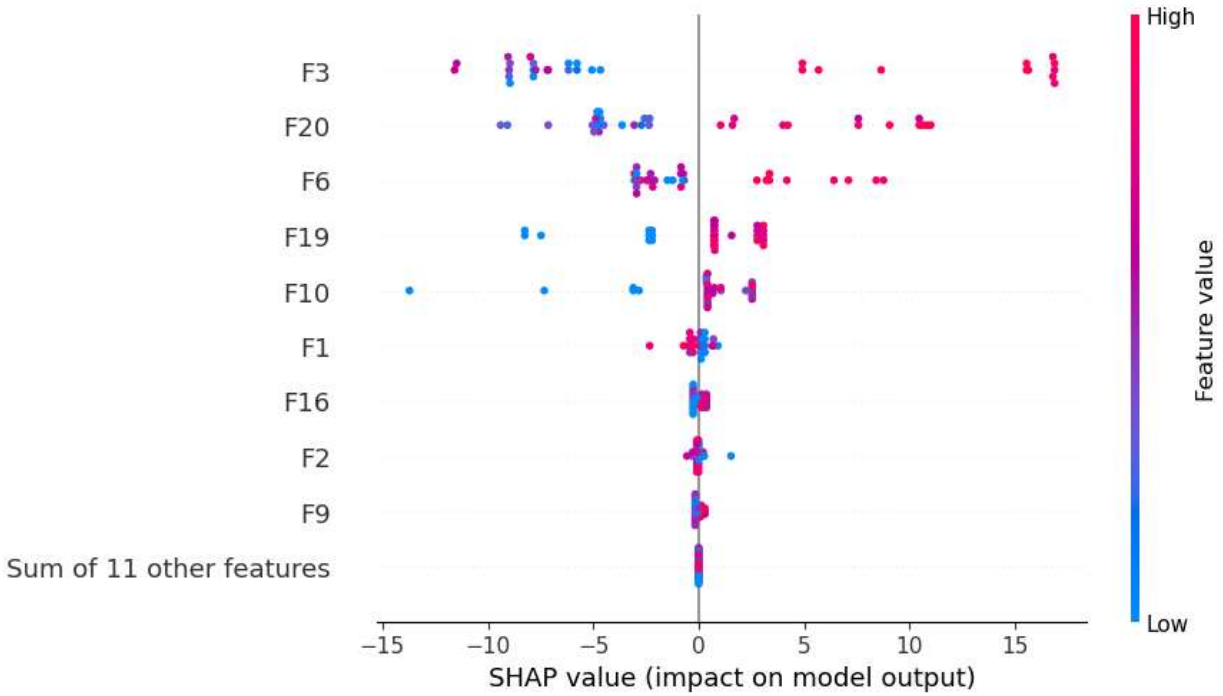
In [74]:
```
# calculate SHAP values for each feature for each instance
shap_values = explainer(X)
```

In [75]:
```
# call the bars() function
bars(shap_values)
```



In [76]:
```
# plot the heatmap
shap.plots.beeswarm(shap_values)
```

```
No data for colormapping provided via 'c'. Parameters 'vmin', 'vmax' will be ignored
```


F3
F20
F6
F19
F10
F1
F16
F2
F9
Sum of 11 other features
−15
−10
−5
0
5
10
15
SHAP value (impact on model output)
High
Feature value
Low

In [ ]: