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 [3]:
```
# 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?: 24
```

Out[3]:

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	...	F13	F14	F15	F16	F17	F18	F19	F20	sympto
556	2	0	15	0	41	0	5	11	39	5	...	19	10	8	0	9	3	5	28	10
557	4	9	0	2	37	10	8	40	44	11	...	0	15	6	10	25	4	8	18	0
558	0	7	0	0	46	12	1	31	41	12	...	0	13	13	0	15	0	16	15	0
559	4	11	8	1	36	16	0	37	52	0	...	18	11	9	17	29	0	9	21	13
560	0	10	0	0	23	0	3	33	50	7	...	19	11	15	14	14	0	0	16	6
561	6	15	0	0	18	12	3	32	43	0	...	16	16	10	15	28	4	8	18	11
562	0	17	7	3	36	12	3	36	45	3	...	21	0	9	0	25	0	8	12	0
563	4	13	12	3	15	15	6	17	37	0	...	0	16	0	9	25	3	0	36	0
564	7	13	0	3	3	15	3	34	49	0	...	13	15	0	12	33	1	0	23	0
565	0	16	7	2	21	13	2	29	41	4	...	0	12	8	14	24	1	8	24	0
566	4	0	12	0	53	15	4	22	52	8	...	0	0	0	0	26	4	16	14	6
567	5	10	2	1	14	8	2	31	30	0	...	0	18	0	0	25	1	11	22	0
568	0	11	7	0	45	8	4	28	39	5	...	0	17	5	9	17	0	14	26	5
569	0	0	10	2	40	16	7	12	39	6	...	0	15	9	14	25	1	0	35	9
570	6	17	11	1	50	14	4	23	46	4	...	18	16	0	13	20	2	14	28	11
571	6	0	0	2	30	7	2	40	30	5	...	0	0	0	0	20	2	10	5	0
572	3	15	13	2	36	10	4	31	43	0	...	17	12	0	11	20	0	11	27	7
573	0	14	6	2	28	12	2	46	33	9	...	17	0	0	13	18	3	11	8	0
574	5	12	14	0	42	14	0	29	31	7	...	19	16	0	11	28	2	11	30	1
575	0	15	0	2	19	17	0	50	33	1	...	15	17	7	0	27	3	0	19	0
576	6	14	13	1	39	13	4	31	59	5	...	0	0	13	10	25	0	9	18	4
577	5	0	0	1	25	9	0	42	36	7	...	17	9	7	0	20	2	8	10	0
578	0	16	0	1	16	16	5	51	46	6	...	13	0	11	16	26	2	9	6	0
579	5	11	8	2	38	12	0	26	38	5	...	0	22	10	15	28	0	14	32	15
580	11	16	0	1	26	16	5	49	60	7	...	15	0	11	11	27	0	8	5	0
581	0	18	3	0	13	15	3	27	36	3	...	20	16	7	15	21	5	7	23	0
582	4	13	0	0	13	13	4	17	44	4	...	0	11	8	13	23	1	9	11	0
583	0	12	0	1	19	16	5	22	53	6	...	0	0	10	9	16	1	0	3	0
584	6	10	9	4	32	18	4	44	54	5	...	0	21	8	15	37	0	0	35	1
585	0	11	2	4	37	19	6	23	41	0	...	16	15	0	16	31	2	9	23	2
586	5	5	12	3	40	14	0	37	35	0	...	0	0	0	14	32	5	8	15	2
587	3	0	8	3	34	13	1	33	32	0	...	21	14	0	13	28	1	9	28	0
588	0	0	0	0	0	13	5	13	47	0	...	0	16	0	18	23	1	0	16	0
589	0	8	17	2	36	17	3	22	33	6	...	0	0	0	0	17	1	11	19	0

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	...	F13	F14	F15	F16	F17	F18	F19	F20	sympto
590	0	0	13	3	53	16	4	21	34	4	...	21	0	0	0	24	1	13	19	(
591	3	13	10	1	46	14	3	32	26	9	...	16	14	0	8	26	2	15	26	4
592	5	16	11	4	37	14	4	55	52	4	...	15	13	8	18	19	2	0	28	6
593	6	0	12	2	38	0	5	35	35	0	...	21	0	9	0	13	0	11	20	4
594	0	10	8	0	40	15	2	33	40	6	...	0	8	10	20	18	3	10	19	17
595	7	10	11	0	39	13	5	25	44	6	...	0	19	7	16	28	6	10	32	11
596	5	9	7	3	19	17	3	45	59	0	...	0	0	8	16	31	0	9	12	(

41 rows × 22 columns

In [4]:
```python
# feature seletion to determine correllation between the colummn 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: ['F3', 'F5', 'F12', 'F16', 'F20']
     F3  F5  F12  F16  F20  symptom_value  user_number
556  15  41    9    0   28      10.875690           24
557   0  37   11   10   18       0.000000           24
558   0  46   12    0   15       0.295267           24
559   8  36   10   17   21      13.479929           24
560   0  23   10   14   16       6.223515           24
561   0  18    0   15   18      11.140630           24
562   7  36    0    0   12       0.000000           24
563  12  15    0    9   36       0.000000           24
564   0   3    0   12   23       0.000000           24
565   7  21   13   14   24       0.000000           24
566  12  53    0    0   14       6.008851           24
567   2  14    0    0   22       0.000000           24
568   7  45    6    9   26       5.062169           24
569  10  40    0   14   35       9.134474           24
570  11  50    0   13   28      11.763835           24
571   0  30   17    0    5       0.000000           24
572  13  36    0   11   27       7.875999           24
573   6  28   10   13    8       0.000000           24
574  14  42   12   11   30       1.002695           24
575   0  19   20    0   19       0.000000           24
576  13  39   11   10   18       4.351895           24
577   0  25   14    0   10       0.000000           24
578   0  16   16   16    6       0.000000           24
579   8  38    0   15   32      15.507169           24
580   0  26    7   11    5       0.000000           24
581   3  13    9   15   23       0.000000           24
582   0  13    0   13   11       0.000000           24
583   0  19   10    9    3       0.000000           24
584   9  32    8   15   35       1.655116           24
585   2  37    0   16   23       2.991732           24
586  12  40   13   14   15       2.206875           24
587   8  34   13   13   28       0.000000           24
588   0   0   13   18   16       0.000000           24
589  17  36   14    0   19       0.000000           24
590  13  53    9    0   19       0.000000           24
591  10  46    0    8   26       4.089193           24
592  11  37   17   18   28       6.157548           24
593  12  38   13    0   20       4.767100           24
594   8  40    9   20   19      17.294536           24
595  11  39    8   16   32      11.946059           24
596   7  19   14   16   12       0.000000           24
```

In [5]:
```python
# split the dataset into training and test data
X = df.iloc[:,:-2]
y = df["symptom_value"]
```

In [6]:
```python
# 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 [7]:
```python
# create a DMatrix for XGBoost
dtrain = xgb.DMatrix(X_train, label=y_train)
```

In [8]:
```python
# specify XGBoost parameters
params = {'max_depth': 3, 'eta': 0.1, 'objective': 'reg:squarederror'}

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

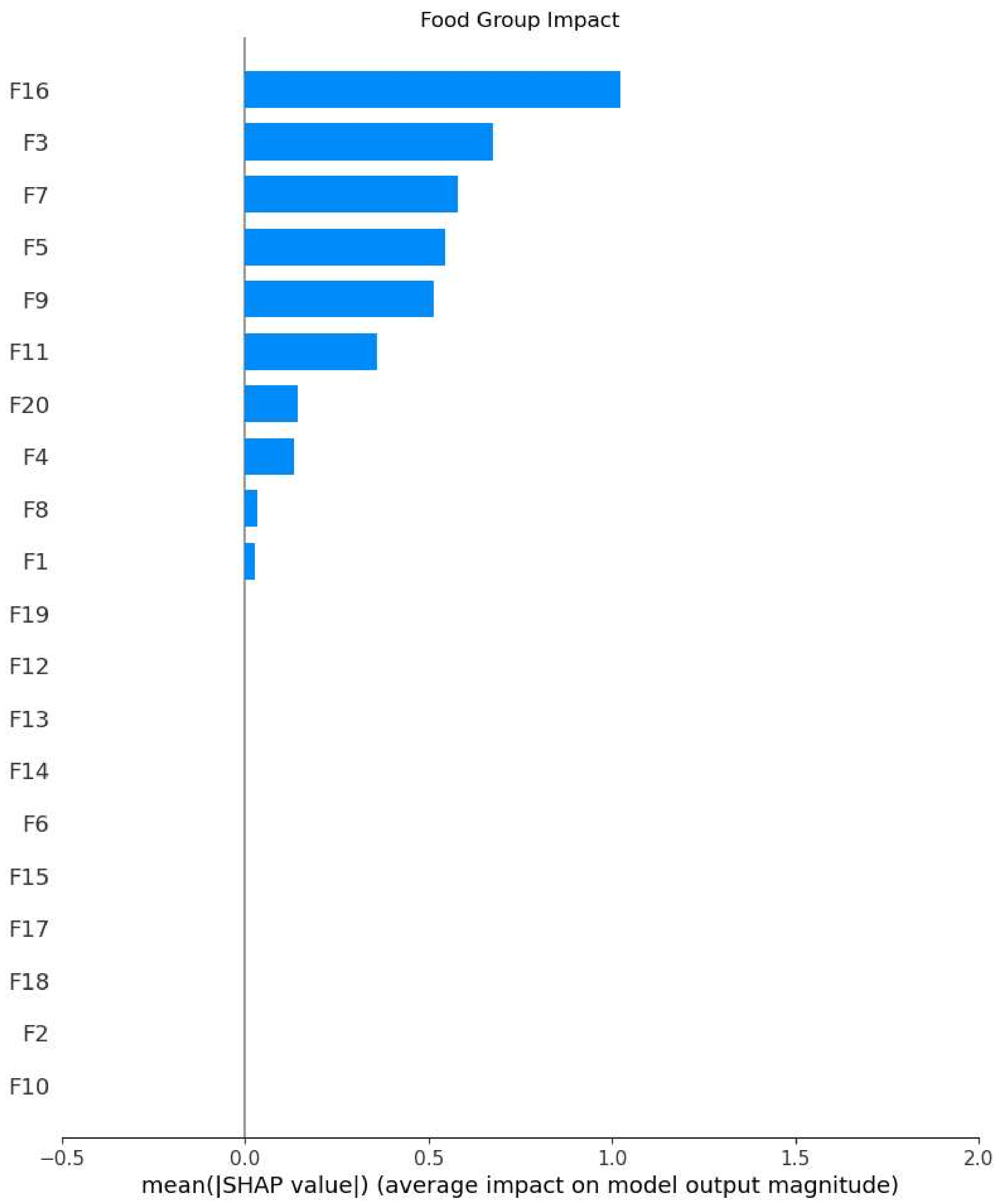
In [9]:
```
# make predictions on the test set
y_pred = model.predict(xgb.DMatrix(X_test))

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

```
R-squared score: 0.1671261311254285
```

In [10]:
```
# create an explainer object for SHAP
explainer = shap.Explainer(model, X)
```

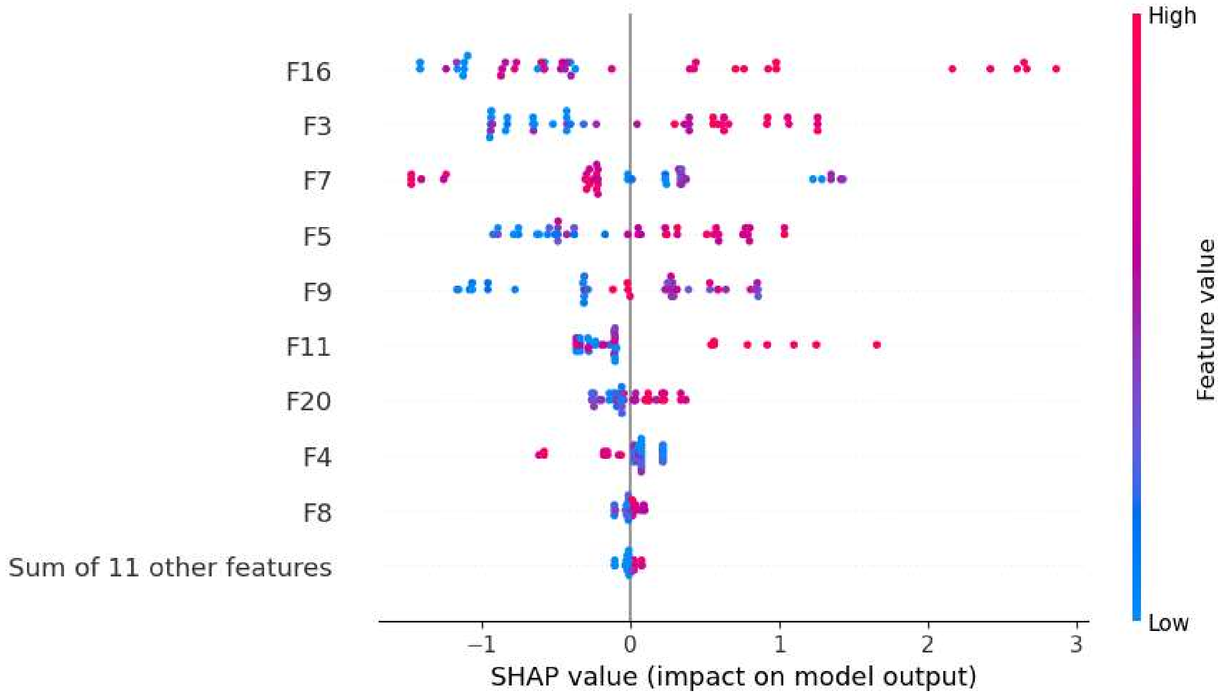
In [11]:
```
# calculate SHAP values for each feature for each instance
shap_values = explainer(X)
```

In [12]:
```
# call the bars() function
bars(shap_values)
```



In [13]:
```
# plot the heatmap
shap.plots.beeswarm(shap_values)
```

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


F16
F3
F7
F5
F9
F11
F20
F4
F8
Sum of 11 other features
−1
0
1
2
3
SHAP value (impact on model output)
High
Low
Feature value