healthy fast foods analysis and Classification

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

Data collection

In [2]:
```
#loading the dataset in pandas
data = pd.read_csv('fastfood.csv')
```

In [3]:
```
#check first five rows of the dataset
data.head()
```

Out[3]:

	restaurant	item	calories	cal_fat	total_fat	sat_fat	trans_fat	cholesterol	sodiu
0	Mcdonalds	Artisan Grilled Chicken Sandwich	380	60	7	2.0	0.0	95	111
1	Mcdonalds	Single Bacon Smokehouse Burger	840	410	45	17.0	1.5	130	158
2	Mcdonalds	Double Bacon Smokehouse Burger	1130	600	67	27.0	3.0	220	192
3	Mcdonalds	Grilled Bacon Smokehouse Chicken Sandwich	750	280	31	10.0	0.5	155	194
4	Mcdonalds	Crispy Bacon Smokehouse Chicken Sandwich	920	410	45	12.0	0.5	120	198

In [4]:
```
#check last five rows of the dataset
data.tail()
```

Out[4]:

	restaurant	item	calories	cal_fat	total_fat	sat_fat	trans_fat	cholesterol	sodi
510	Taco Bell	Spicy Triple Double Crunchwrap	780	340	38	10.0	0.5	50	1
511	Taco Bell	Express Taco Salad w/ Chips	580	260	29	9.0	1.0	60	1
512	Taco Bell	Fiesta Taco Salad-Beef	780	380	42	10.0	1.0	60	1
513	Taco Bell	Fiesta Taco Salad-Chicken	720	320	35	7.0	0.0	70	1
514	Taco Bell	Fiesta Taco Salad-Steak	720	320	36	8.0	1.0	55	1

In [5]:
```
#check shape of the dataset
data.shape
```

Out[5]:
```
(515, 17)
```

In [6]:
```
#check columns
data.columns
```

Out[6]:
```
Index(['restaurant', 'item', 'calories', 'cal_fat', 'total_fat', '
sat_fat',
       'trans_fat', 'cholesterol', 'sodium', 'total_carb', 'fiber'
, 'sugar',
       'protein', 'vit_a', 'vit_c', 'calcium', 'salad'],
      dtype='object')
```

In [7]:
```
#check basic infomation
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 515 entries, 0 to 514
Data columns (total 17 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   restaurant   515 non-null    object 
 1   item         515 non-null    object 
 2   calories     515 non-null    int64  
 3   cal_fat      515 non-null    int64  
 4   total_fat    515 non-null    int64  
 5   sat_fat      515 non-null    float64
 6   trans_fat    515 non-null    float64
 7   cholesterol  515 non-null    int64  
 8   sodium       515 non-null    int64  
 9   total_carb   515 non-null    int64  
 10  fiber        503 non-null    float64
 11  sugar        515 non-null    int64  
 12  protein      514 non-null    float64
 13  vit_a        301 non-null    float64
 14  vit_c        305 non-null    float64
 15  calcium      305 non-null    float64
 16  salad        515 non-null    object 
dtypes: float64(7), int64(7), object(3)
memory usage: 68.5+ KB
```

In [8]:
```
#check mathamtic realtionship
data.describe()
```

Out[8]:

	calories	cal_fat	total_fat	sat_fat	trans_fat	cholesterol	sod
count	515.000000	515.000000	515.000000	515.000000	515.000000	515.000000	515.000
mean	530.912621	238.813592	26.590291	8.153398	0.465049	72.456311	1246.737
std	282.436147	166.407510	18.411876	6.418811	0.839644	63.160406	689.954
min	20.000000	0.000000	0.000000	0.000000	0.000000	0.000000	15.000
25%	330.000000	120.000000	14.000000	4.000000	0.000000	35.000000	800.000
50%	490.000000	210.000000	23.000000	7.000000	0.000000	60.000000	1110.000
75%	690.000000	310.000000	35.000000	11.000000	1.000000	95.000000	1550.000
max	2430.000000	1270.000000	141.000000	47.000000	8.000000	805.000000	6080.000

In [9]:
```
#check corr realtion of the dataset
data.corr()
```

```
/var/folders/qn/zzvcdwg51dzdyg67zv6vp9lw0000gn/T/ipykernel_1054/3539882025.py:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  data.corr()
```

Out[9]:

	calories	cal_fat	total_fat	sat_fat	trans_fat	cholesterol	sodium	tc
calories	1.000000	0.901661	0.900494	0.739664	0.530354	0.762460	0.817855	
cal_fat	0.901661	1.000000	0.995311	0.852150	0.648422	0.803997	0.670458	
total_fat	0.900494	0.995311	1.000000	0.846716	0.648821	0.801352	0.669182	
sat_fat	0.739664	0.852150	0.846716	1.000000	0.812126	0.764030	0.487456	
trans_fat	0.530354	0.648422	0.648821	0.812126	1.000000	0.680858	0.261466	
cholesterol	0.762460	0.803997	0.801352	0.764030	0.680858	1.000000	0.596164	
sodium	0.817855	0.670458	0.669182	0.487456	0.261466	0.596164	1.000000	
total_carb	0.712701	0.419373	0.422543	0.276534	0.100284	0.238728	0.671976	
fiber	0.287031	0.032643	0.036386	-0.041220	-0.121890	-0.061503	0.301023	
sugar	0.437711	0.255485	0.259370	0.234218	0.112651	0.298259	0.422993	
protein	0.831957	0.720379	0.719518	0.603645	0.478960	0.880960	0.766942	
vit_a	-0.153963	-0.121748	-0.122280	-0.054292	-0.086352	-0.057142	-0.134031	-
vit_c	0.007387	-0.115456	-0.112868	-0.088834	-0.141754	-0.015610	0.062994	
calcium	0.351207	0.166801	0.168817	0.304948	0.114094	0.162438	0.284882	

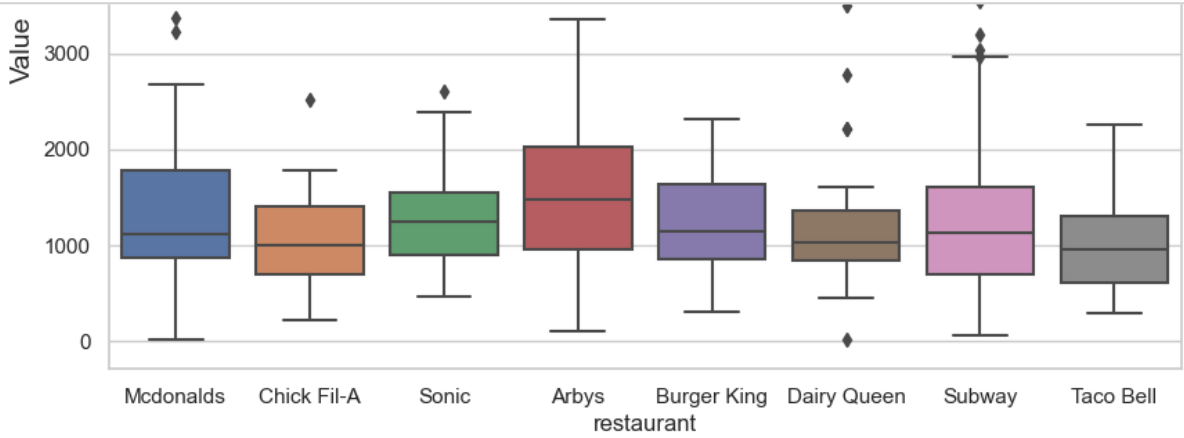
In [10]:
```
#check missing value of the dataset
data.isnull().sum()
```

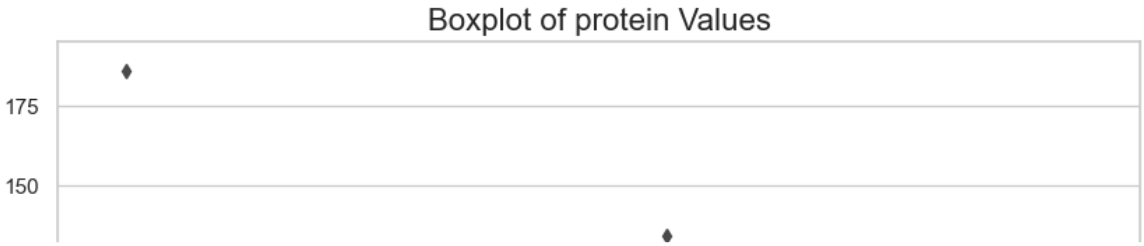
Out[10]:
```
restaurant       0
item             0
calories         0
cal_fat          0
total_fat        0
sat_fat          0
trans_fat        0
cholesterol      0
sodium           0
total_carb       0
fiber           12
sugar            0
protein          1
vit_a          214
vit_c          210
calcium        210
salad            0
dtype: int64
```

EDA of the Data

In [11]:
```
restaurants = data['restaurant'].unique()
nutritions = ['calories', 'sat_fat', 'trans_fat', 'sodium', 'protei
```

In [12]:
```
sns.set(style="whitegrid")


for nutrition in nutritions:
    fig, ax = plt.subplots(figsize=(10, 6))
    sns.boxplot(x='restaurant', y=nutrition, data=data, ax=ax)
    ax.set_ylabel('Value', fontsize=14)
    ax.set_title(f"Boxplot of {nutrition} Values", fontsize=16)

    plt.show()
```





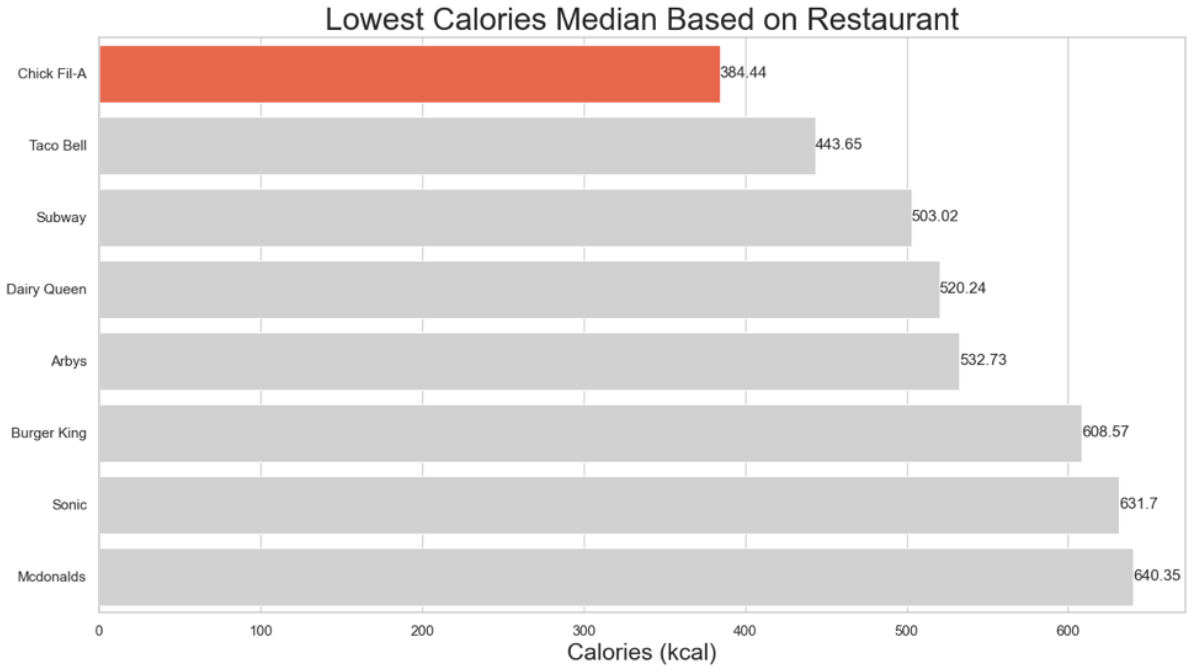
In [13]:
```
nutritions_per_restaurant = data.groupby('restaurant').mean().round
nutritions_per_restaurant
```

```
/var/folders/qn/zzvcdwg51dzdyg67zv6vp9lw0000gn/T/ipykernel_1054/35
68129121.py:1: FutureWarning: The default value of numeric_only in
DataFrameGroupBy.mean is deprecated. In a future version, numeric_
only will default to False. Either specify numeric_only or select
only columns which should be valid for the function.
  nutritions_per_restaurant = data.groupby('restaurant').mean().ro
und(2).sort_values(by='calories').reset_index()
```

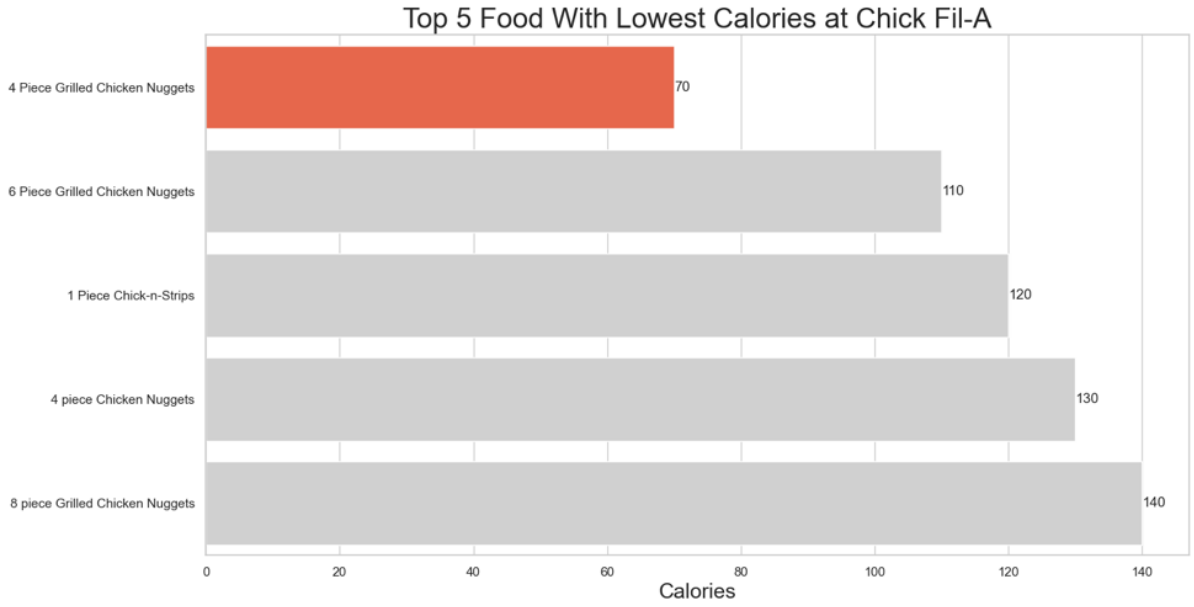
Out[13]:

	restaurant	calories	cal_fat	total_fat	sat_fat	trans_fat	cholesterol	sodium	total_carb
0	Chick Fil-A	384.44	145.37	16.15	4.11	0.04	79.07	1151.48	28.63
1	Taco Bell	443.65	188.00	20.90	6.59	0.26	39.04	1013.91	46.63
2	Subway	503.02	165.10	18.48	6.20	0.22	61.30	1272.97	54.72
3	Dairy Queen	520.24	260.48	28.86	10.44	0.68	71.55	1181.79	38.69
4	Arbys	532.73	237.84	26.98	7.97	0.42	70.45	1515.27	44.87
5	Burger King	608.57	333.76	36.81	11.15	0.86	100.86	1223.57	39.31
6	Sonic	631.70	338.30	37.64	11.42	0.93	86.98	1350.75	47.21
7	Mcdonalds	640.35	285.61	31.81	8.29	0.46	109.74	1437.89	48.79

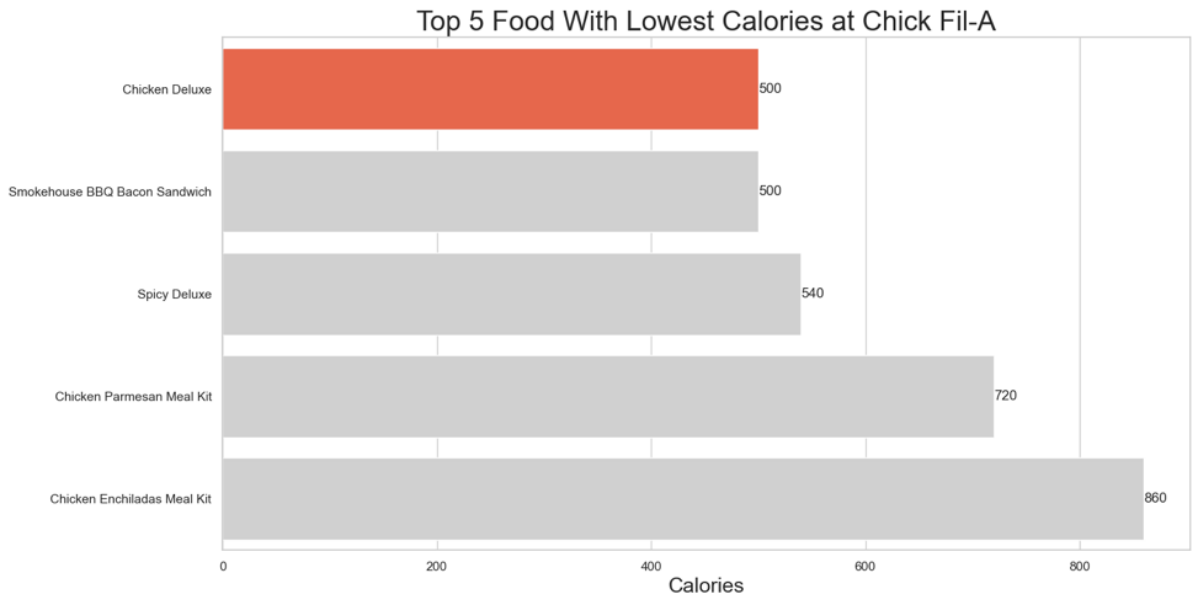
In [14]:
```
fig, ax = plt.subplots(figsize=(15, 8))

custom_palette = ['#FF5733'] + ['#D0D0D0']*(len(nutritions_per_rest

sns.barplot(x = 'calories', y='restaurant', data=nutritions_per_res
            , palette=custom_palette)
plt.title('Lowest Calories Median Based on Restaurant', fontsize=24
plt.xlabel('Calories (kcal)', fontsize=18)
plt.ylabel('')

for i in ax.containers:
    plt.bar_label(i, label_type='edge')


plt.show()
```



In [15]:
```
top_5_lowest_chick = data[data['restaurant'] == 'Chick Fil-A'].sort

custom_palette = ['#FF5733'] + ['#D0D0D0']*(len(top_5_lowest_chick[


fig, ax = plt.subplots(figsize=(15, 8))
sns.barplot(x='calories', y='item', palette=custom_palette,
            data=top_5_lowest_chick, dodge=False, ax=ax)

for i in ax.containers:
    ax.bar_label(i, label_type='edge')

plt.title('Top 5 Food With Lowest Calories at Chick Fil-A', fontsiz
plt.xlabel('Calories', fontsize=18)
plt.ylabel('')
plt.show()
```

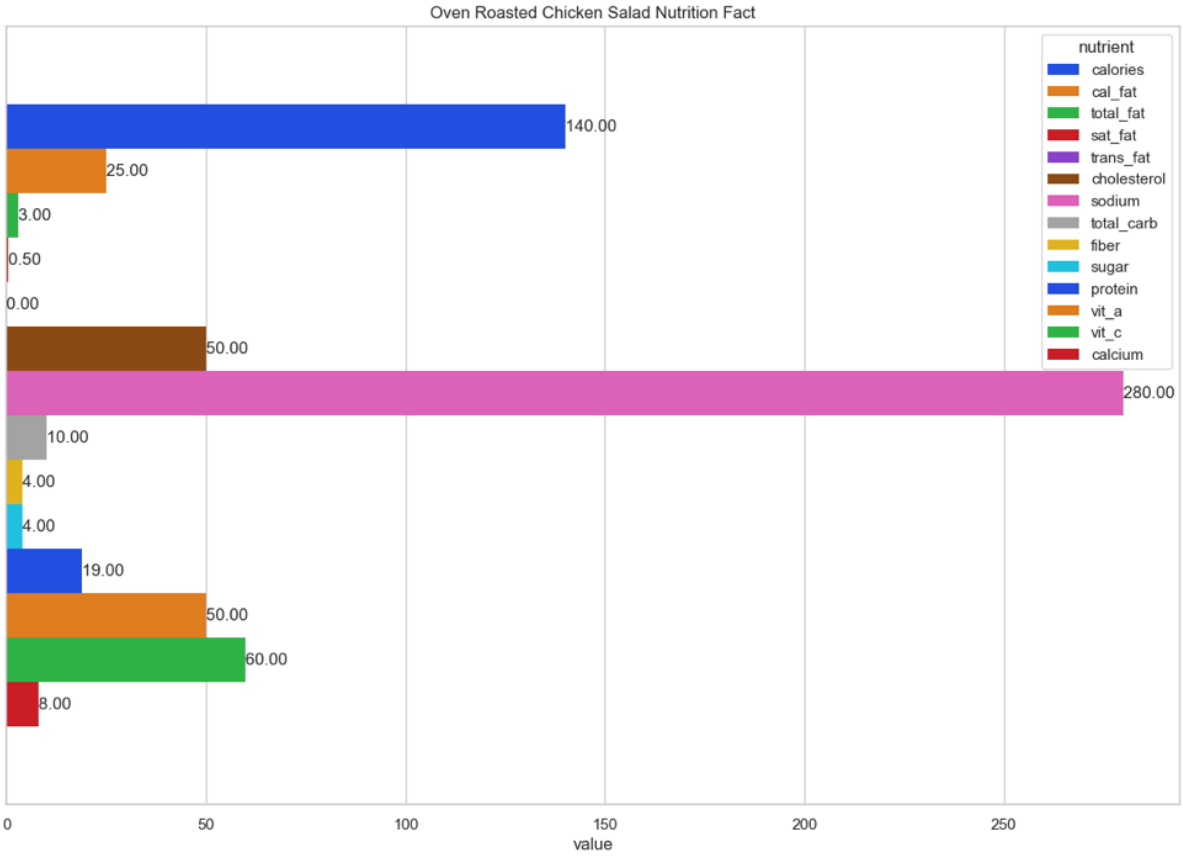


In [16]:
```
op_5_chick = data[(data['restaurant'] == 'Chick Fil-A') & (data['cal

ustom_palette = ['#FF5733'] + ['#D0D0D0']*(len(top_5_chick['item'].u


ig, ax = plt.subplots(figsize=(15, 8))
ns.barplot(x='calories', y='item', palette=custom_palette,
           data=top_5_chick, dodge=False, ax=ax)

or i in ax.containers:
   ax.bar_label(i, label_type='edge')

lt.title('Top 5 Food With Lowest Calories at Chick Fil-A', fontsize=
lt.xlabel('Calories', fontsize=18)
lt.ylabel('')
lt.show()
```

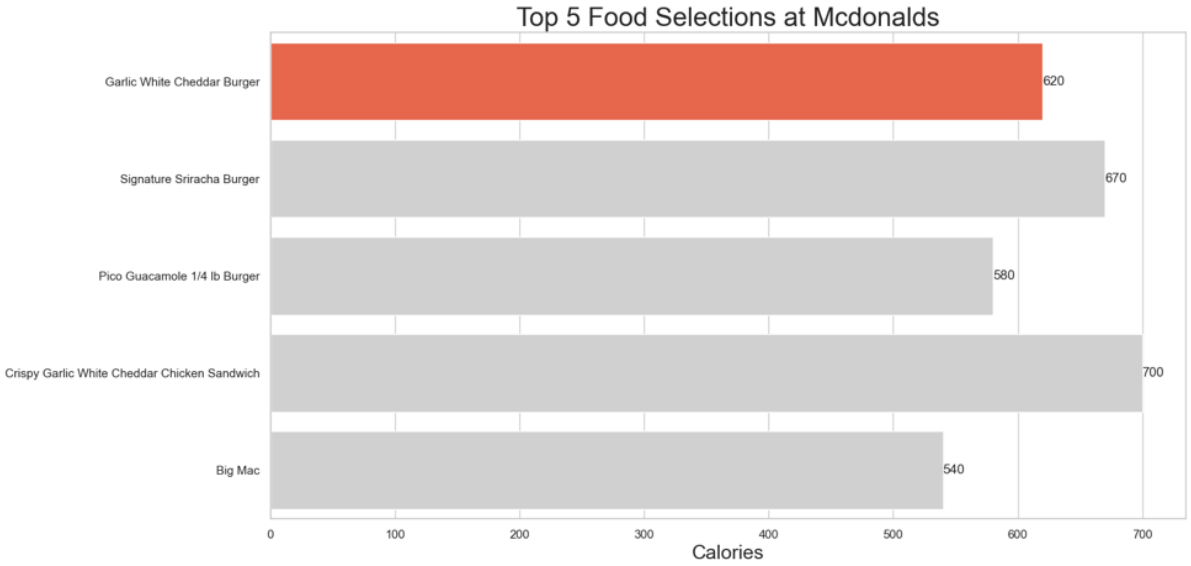


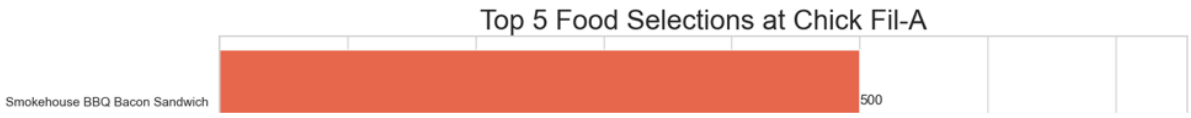
In [17]:
```
#Protein to calories ratio
data['protein_to_cal_ratio'] = data['protein'] / data['calories']
data['sodium_to_cal_ratio'] = data['sodium'] / data['calories']
top_10_cal_ratio = data.sort_values(by=['sodium_to_cal_ratio', 'pro
top_10_cal_ratio = top_10_cal_ratio[top_10_cal_ratio['protein_to_ca
top_10_cal_ratio.drop('salad', inplace=True, axis=1)
```

In [18]:
```
top_1_protein_cal_ratio = pd.melt(
    top_10_cal_ratio.head(1).drop(['restaurant', 'protein_to_cal_ra
    id_vars=['item'], var_name='nutrient', value_name='value')
top_1_protein_cal_ratio['value'] = top_1_protein_cal_ratio['value']

fig, ax = plt.subplots(figsize=(15, 10))
sns.barplot(x='value', y='item', hue='nutrient', data=top_1_protein
plt.title(f"{top_1_protein_cal_ratio['item'][0]} Nutrition Fact")
ax.get_yaxis().set_ticks([])
ax.set_ylabel('')

for container in ax.containers:
    ax.bar_label(container,labels=[f"{val:.2f} " for val in contain

plt.show()
```



In [19]:
```
for restaurant in restaurants:
    top_5 = data[(data['restaurant'] == restaurant) & (data['calori
    top_5 = top_5.sort_values(by=['sodium_to_cal_ratio', 'protein_t

    custom_palette = ['#FF5733'] + ['#D0D0D0']*(len(top_5['item'].u


    fig, ax = plt.subplots(figsize=(15, 8))
    sns.barplot(x='calories', y='item', palette=custom_palette,
                data=top_5, dodge=False, ax=ax)

    for i in ax.containers:
        ax.bar_label(i, label_type='edge')

    plt.title(f"Top 5 Food Selections at {restaurant}", fontsize=24
    plt.xlabel('Calories', fontsize=18)
    plt.ylabel('')
    plt.show()
```





In [20]:
```
null_vals = dict(data.isnull().sum())
null_vals
```

Out[20]:
```
{'restaurant': 0,
 'item': 0,
 'calories': 0,
 'cal_fat': 0,
 'total_fat': 0,
 'sat_fat': 0,
 'trans_fat': 0,
 'cholesterol': 0,
 'sodium': 0,
 'total_carb': 0,
 'fiber': 12,
 'sugar': 0,
 'protein': 1,
 'vit_a': 214,
 'vit_c': 210,
 'calcium': 210,
 'salad': 0,
 'protein_to_cal_ratio': 1,
 'sodium_to_cal_ratio': 0}
```

In [21]:
```
# % null values
for key,val in null_vals.items():
    print(f"null values for {key} =======> {(int(val)/data.shape[0]
```

```
null values for restaurant =======> 0.0
null values for item =======> 0.0
null values for calories =======> 0.0
null values for cal_fat =======> 0.0
null values for total_fat =======> 0.0
null values for sat_fat =======> 0.0
null values for trans_fat =======> 0.0
null values for cholesterol =======> 0.0
null values for sodium =======> 0.0
null values for total_carb =======> 0.0
null values for fiber =======> 2.3300970873786406
null values for sugar =======> 0.0
null values for protein =======> 0.1941747572815534
null values for vit_a =======> 41.55339805825243
null values for vit_c =======> 40.77669902912621
null values for calcium =======> 40.77669902912621
null values for salad =======> 0.0
null values for protein_to_cal_ratio =======> 0.1941747572815534
null values for sodium_to_cal_ratio =======> 0.0
```

In [22]:
```
# replace vitamin a ,c and calcium with mean value, for this collec
null_cols = ['fiber','protein','vit_a','vit_c','calcium']
null_cols_avg = {}
for col in null_cols:
    null_cols_avg[col] = data[col].describe().mean()
null_cols_avg
```

Out[22]:
```
{'fiber': 67.14682965168379,
 'protein': 102.8843714070368,
 'vit_a': 70.6551841488183,
 'vit_c': 99.97034180718309,
 'calcium': 87.92181644364442}
```

In [23]:
```
data.fillna(value=null_cols_avg,inplace=True)
data.isnull().sum()
```

Out[23]:
```
restaurant              0
item                    0
calories                0
cal_fat                 0
total_fat               0
sat_fat                 0
trans_fat               0
cholesterol             0
sodium                  0
total_carb              0
fiber                   0
sugar                   0
protein                 0
vit_a                   0
vit_c                   0
calcium                 0
salad                   0
protein_to_cal_ratio    1
sodium_to_cal_ratio     0
dtype: int64
```

In [24]:
```
# Oh cleared it, now ready to go
# here we will consider only total fat, the reason is I have not go
# about trans_fat,sat_fat and cal_fat, I appologise for this, lets
data.drop(['salad','cal_fat','sat_fat','trans_fat'],axis=1,inplace=
```

1) Collecting labels `unsuperwised learning`

Steps followed:

- Collecting libraries
- Dropping unrequired columns
- Finding best cluster
- Visualizing elbow method
- Training model with best cluster
- Getting labels

In [25]:
```
df_seg = data.drop(['restaurant','item'],axis='columns')
df_seg.sample(3)
```

Out[25]:

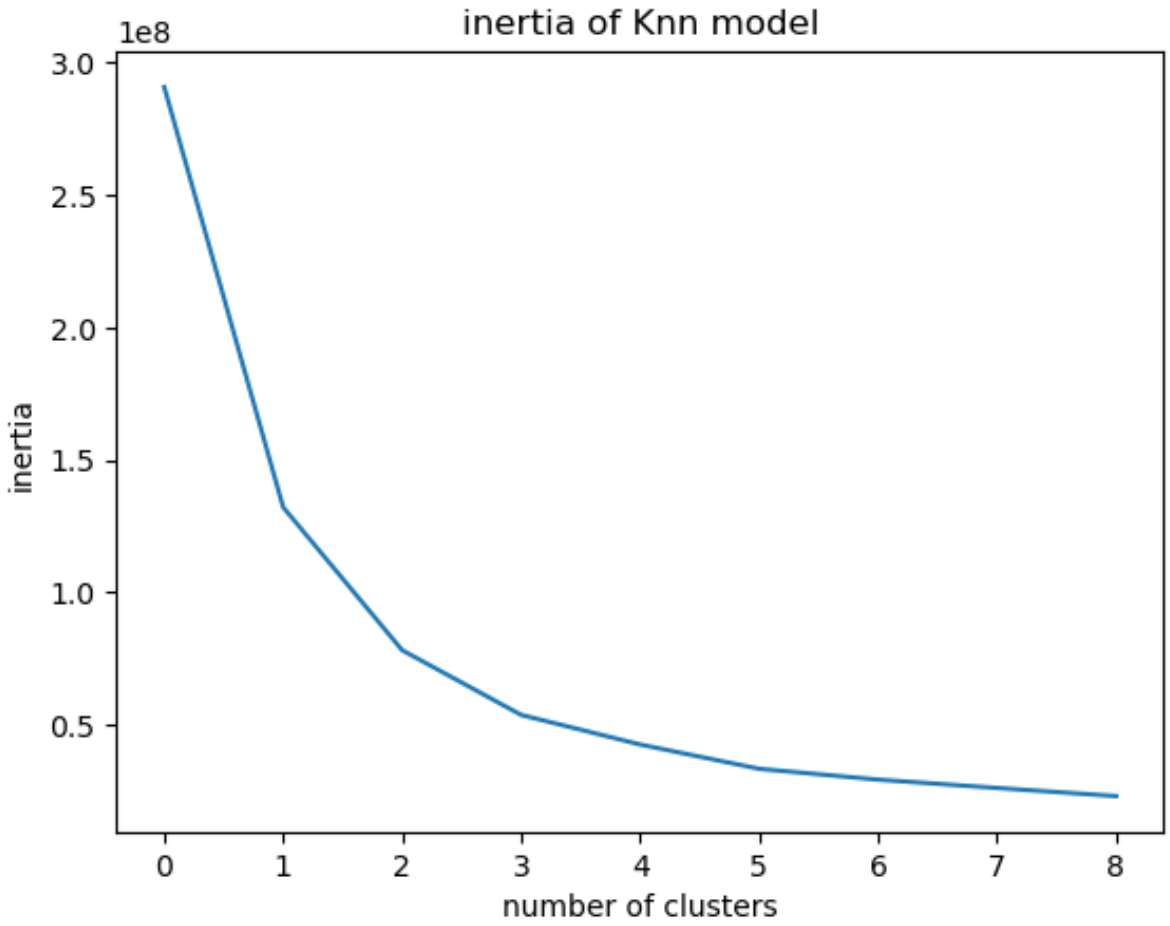
	calories	total_fat	cholesterol	sodium	total_carb	fiber	sugar	protein	vit_a	vit_c
87	410	24	55	730	32	1.0	4	20.0	7.0	1.0
458	420	28	65	1070	23	4.0	2	19.0	6.0	4.0
366	390	7	10	800	56	8.0	8	23.0	15.0	20.0

Finding best cluster size for segmentation

In [ ]:
```
inertias = []
for i in range(1,10):
    model = KMeans(n_clusters=i,init="k-means++",random_state=42)
    model.fit(df_seg)
    inertias.append(model.inertia_)
print(inertias)
```

Elbow method

In [13]:
```
plt.plot(inertias)
plt.title("inertia of Knn model")
plt.xlabel("number of clusters")
plt.ylabel("inertia")
plt.show()
```



As per elbow method 3 types of clusters are best for our model, that means we can classify foods into 3 types.

In [14]:
```
model = KMeans(n_clusters=3,init='k-means++',random_state=42)
model.fit(df_seg)
```

Out[14]:
```
KMeans(n_clusters=3, random_state=42)
```

In [15]:
```
cluster_centers = model.cluster_centers_
cluster_centers
```

Out[15]:
```
array([[1134.16666667,   57.25      ,  172.08333333, 3282.91666667
,
          87.29166667,    9.2561179 ,   15.        ,   67.
,
          25.5804627 ,   43.41048788,   47.73371176],
       [ 362.79863481,   17.54266212,   47.11604096,  806.55290102
,
          32.54266212,    3.570467  ,    5.25938567,   19.07167235
,
          43.55714278,   52.93645632,   46.15857203],
       [ 706.56565657,   36.26262626,   97.87878788, 1651.31313131
,
          60.03535354,    8.17408231,    9.28787879,   36.5802241
,
          37.47492661,   53.50271904,   57.44197356]])
```

In [16]:
```
labels = model.labels_
labels[:10]
```

Out[16]:
```
array([1, 2, 2, 2, 2, 1, 1, 1, 1, 2], dtype=int32)
```

In [17]:
```
df['labels'] = labels
df.sample(3)
```

Out[17]:

	restaurant	item	calories	total_fat	cholesterol	sodium	total_carb	fiber	sugar
252	Burger King	Jalapeno Chicken Fries	300	18	40	950	19	1.0	1
420	Taco Bell	Steak Quesarito	630	31	65	1410	64	3.0	4
297	Dairy Queen	Crispy Chicken Wrap	350	21	35	820	30	2.0	1

Done first part 😌

2) Segmenting data into `healthy food` and `unhealthy food`

In [19]:
```
label_0 = df[df['labels']==0]
label_1 = df[df['labels']==1]
label_2 = df[df['labels']==2]
print("label 0",len(label_0))
print("label 1",len(label_1))
print("label 2",len(label_2))
```

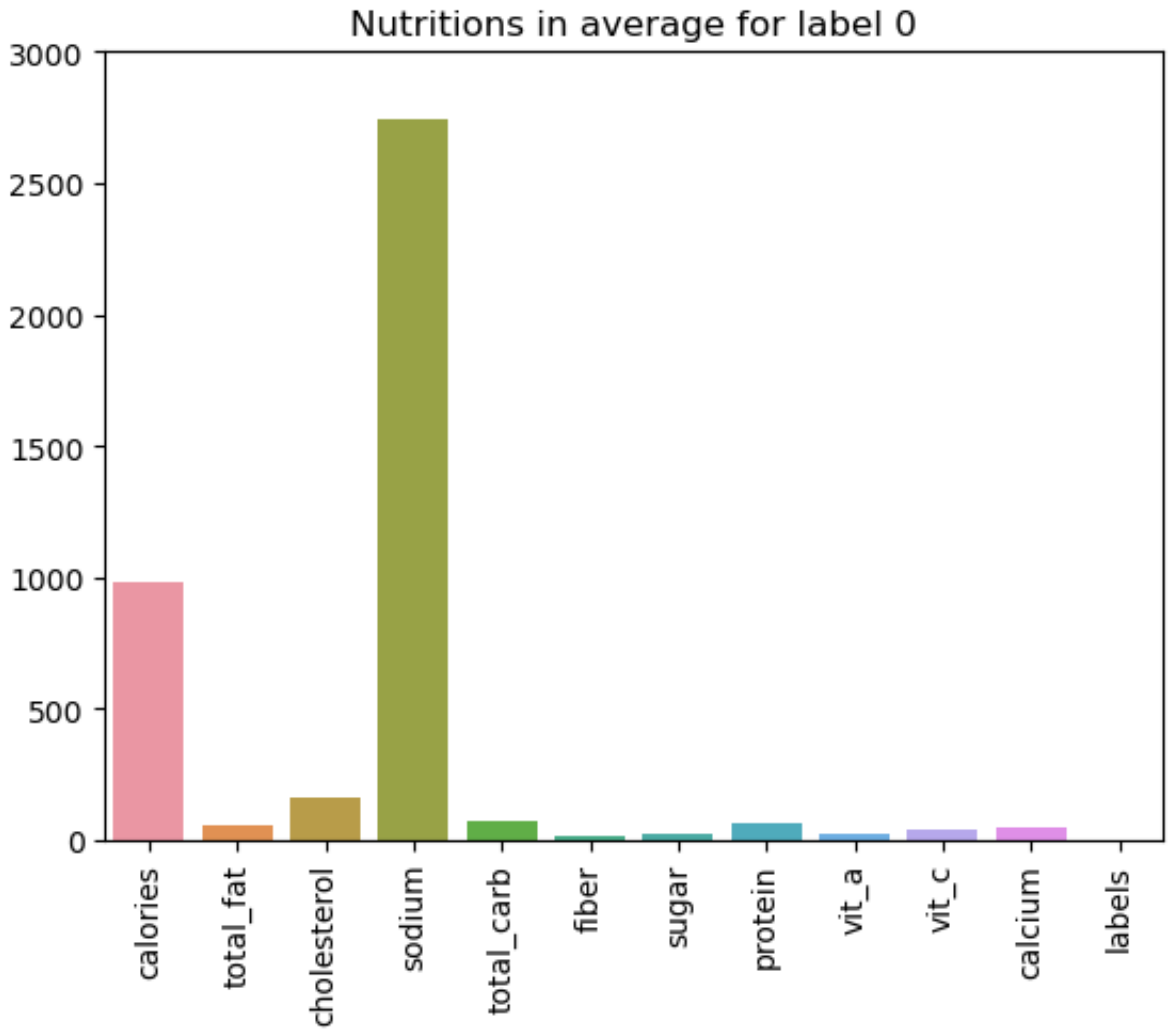
```
label 0 24
label 1 293
label 2 198
```

Collecting average of Nutrition

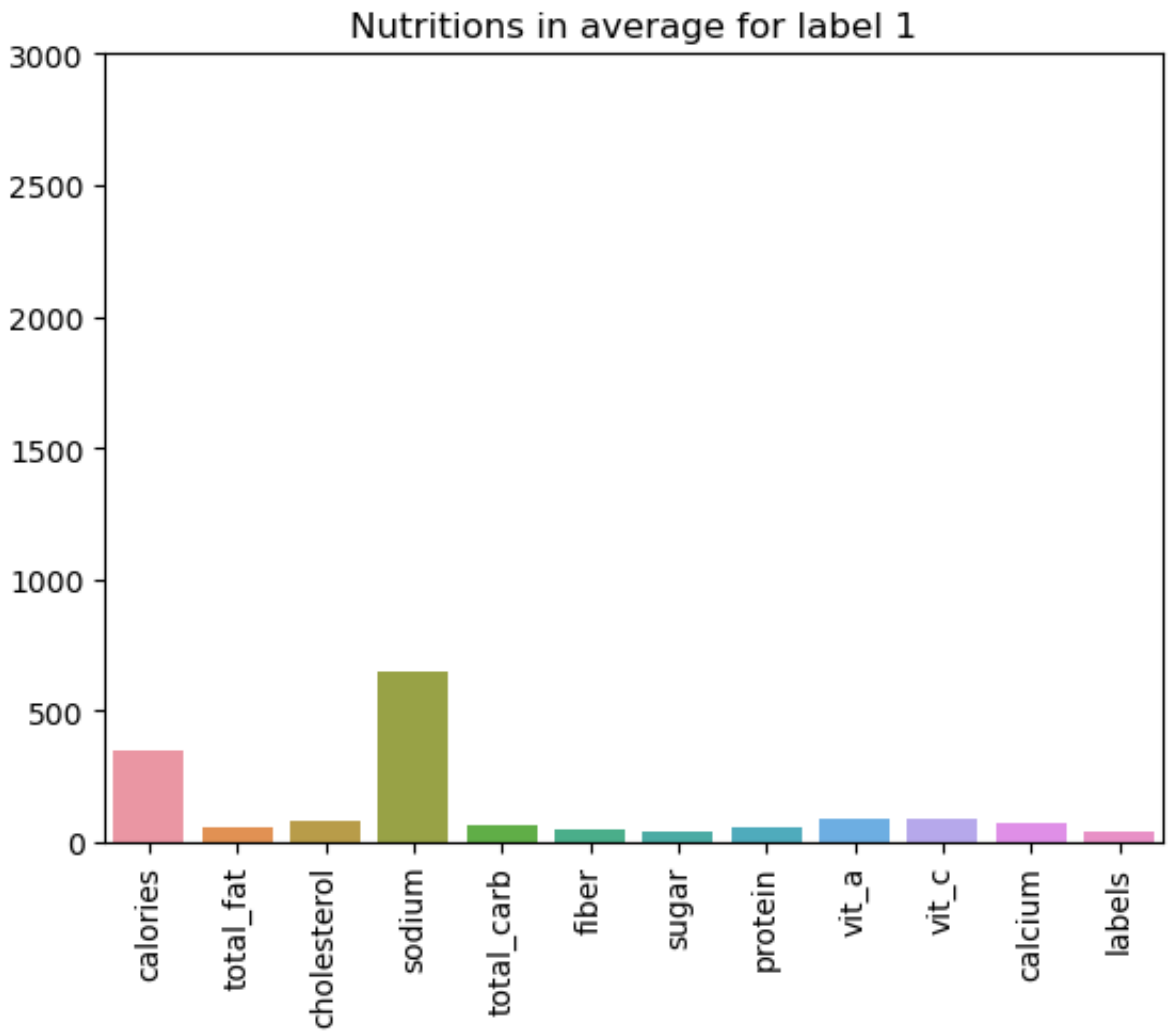
In [20]:
```
nutritions = list(label_0.describe().columns)
label_0_nutri_avg = label_0.describe().mean().values
label_1_nutri_avg = label_1.describe().mean().values
label_2_nutri_avg = label_2.describe().mean().values
print("label_0_nutri_avg",label_0_nutri_avg)
print("label_1_nutri_avg",label_1_nutri_avg)
print("label_2_nutri_avg",label_2_nutri_avg)
```

```
label_0_nutri_avg [ 983.68437654   54.17094372  159.72297763 2747.
68371272   74.06228987
   16.85584648   22.03154889   64.67569437   26.03244575   41.5551
4837
   45.10334094    3.        ]
label_1_nutri_avg [348.17926188  53.34113266  81.06354433 651.4759
0738  66.8572915
  47.13631206  44.21520818  54.22214806  85.46994363  93.22141349
  75.50071478  37.375     ]
label_2_nutri_avg [ 597.76546522   60.88604693  179.10830333 1338.
3392055    72.52380115
   37.79632294   34.45431951   62.62686176   54.51430051  106.1125
0799
   94.82042994   26.25      ]
```

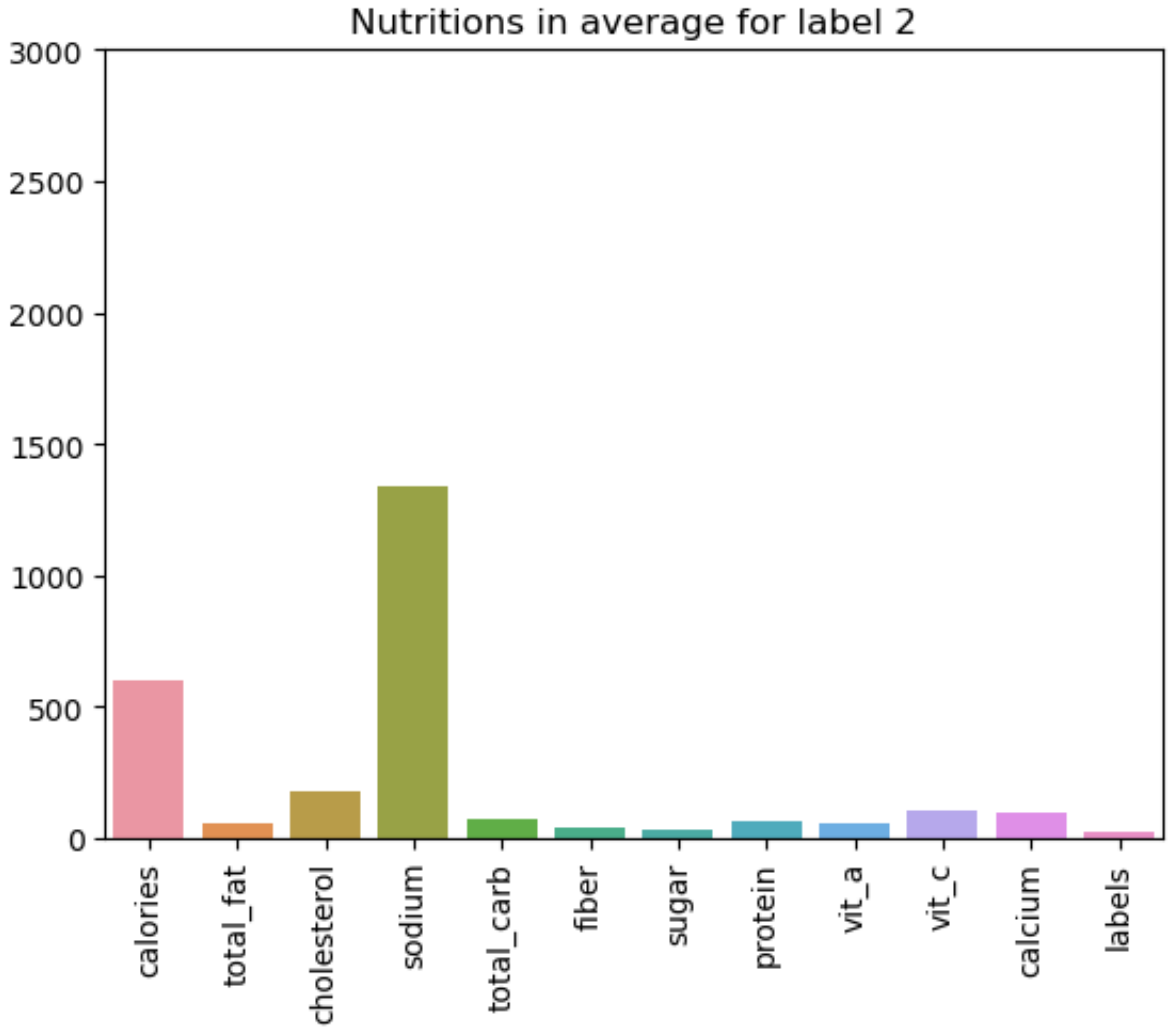
In [21]:
```
# label 0
sn.barplot(x=nutritions,y=label_0_nutri_avg)
plt.title("Nutritions in average for label 0")
plt.xticks(rotation=90)
plt.ylim([0,3000])
plt.show()
```



In [22]:
```
# label 1
sn.barplot(x=nutritions,y=label_1_nutri_avg)
plt.title("Nutritions in average for label 1")
plt.xticks(rotation=90)
plt.ylim([0,3000])
plt.show()
```



In [23]:
```
# label 2
sn.barplot(x=nutritions,y=label_2_nutri_avg)
plt.title("Nutritions in average for label 2")
plt.xticks(rotation=90)
plt.ylim([0,3000])
plt.show()
```



In [24]:
```
positive_ntr = ['protein','calcium','total_carb','vit_a','vit_c','f
negative_ntr = ['sugar','calories','total_fat','sodium','cholestero
```

zipping average values of nutritions and labels (nutritions) in dictionary format

In [25]:
```
def zipper(lis1,lis2):
    temp = {}
    for v1,v2 in zip(lis1,lis2):
        temp[v1] = v2
    return temp

label_0_dic = zipper(nutritions,label_0_nutri_avg)
label_1_dic = zipper(nutritions,label_1_nutri_avg)
label_2_dic = zipper(nutritions,label_2_nutri_avg)
print("label_0_zipped",label_0_dic)
print("\n\nlabel_1_zipped",label_1_dic)
print("\n\nlabel_2_zipped",label_2_dic)
```

```
label_0_zipped {'calories': 983.6843765410205, 'total_fat': 54.170
943716023075, 'cholesterol': 159.7229776299231, 'sodium': 2747.683
7127196836, 'total_carb': 74.06228986650855, 'fiber': 16.855846476
00119, 'sugar': 22.031548893984787, 'protein': 64.67569436810251,
'vit_a': 26.03244574783713, 'vit_c': 41.55514836677963, 'calcium':
45.10334093742827, 'labels': 3.0}


label_1_zipped {'calories': 348.1792618797805, 'total_fat': 53.341
132663545174, 'cholesterol': 81.06354432632749, 'sodium': 651.4759
07381187, 'total_carb': 66.8572915034758, 'fiber': 47.136312059280
49, 'sugar': 44.215208179041106, 'protein': 54.22214806008845, 'vi
t_a': 85.46994362676779, 'vit_c': 93.2214134895857, 'calcium': 75.
50071477749262, 'labels': 37.375}


label_2_zipped {'calories': 597.7654652211517, 'total_fat': 60.886
046928150904, 'cholesterol': 179.10830332930465, 'sodium': 1338.33
92054964756, 'total_carb': 72.52380115219286, 'fiber': 37.79632293
7000525, 'sugar': 34.45431950755102, 'protein': 62.626861761152654
, 'vit_a': 54.51430051314611, 'vit_c': 106.11250798555193, 'calciu
m': 94.82042994096822, 'labels': 26.25}
```

creating dictionary for each label (cluster) with positive nutrition and negative nutrition average values

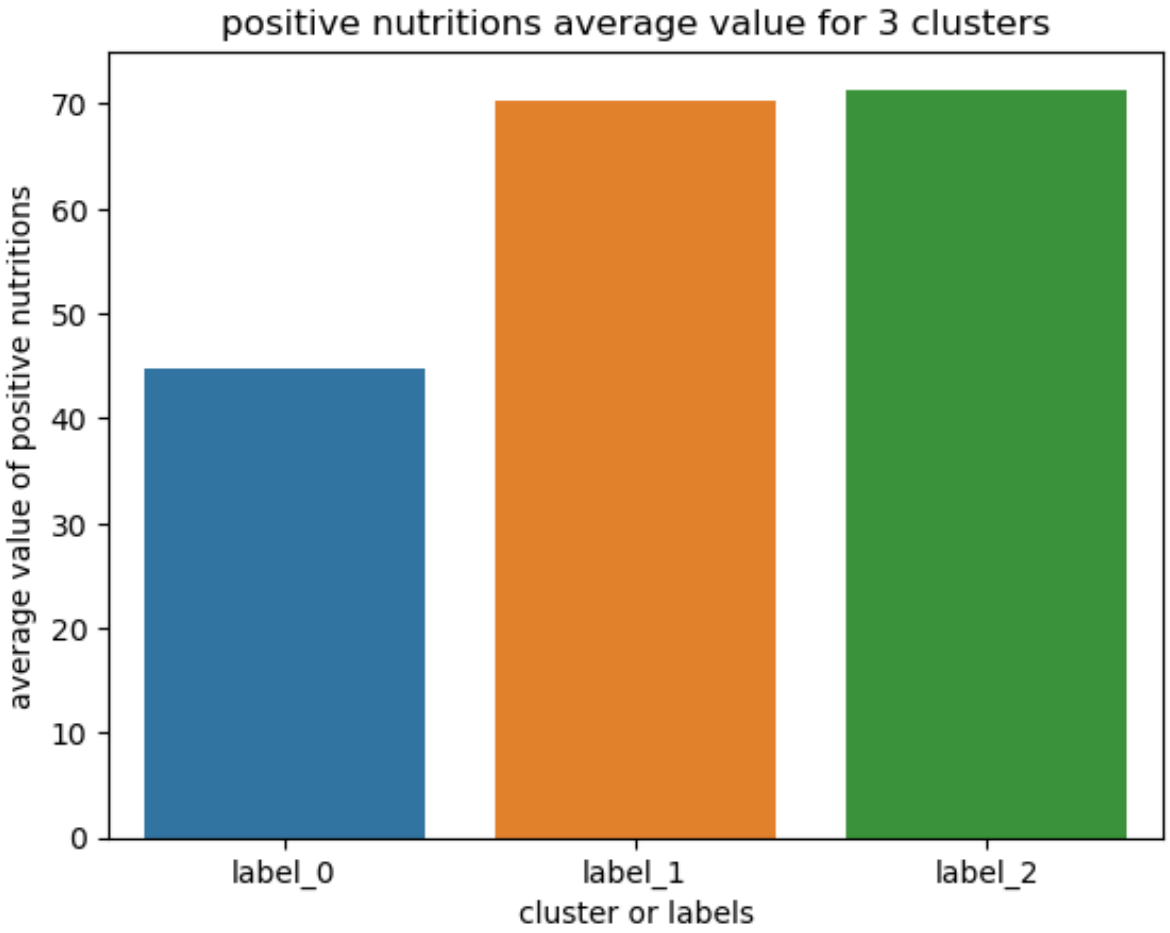
In [26]:
```
pn_avg_label_0 = {}
pn_avg_label_1 = {}
pn_avg_label_2 = {}
for i in range(3):
    lis1 = []
    lis2 = []
    lis3 = []
    for pos_val in positive_ntr:
        lis1.append(label_0_dic[pos_val])
        lis2.append(label_1_dic[pos_val])
        lis3.append(label_2_dic[pos_val])
    pn_avg_label_0['pos'] = sum(lis1)/len(lis1)
    pn_avg_label_1['pos'] = sum(lis2)/len(lis2)
    pn_avg_label_2['pos'] = sum(lis3)/len(lis3)


for i in range(3):
    lis1 = []
    lis2 = []
    lis3 = []
    for pos_val in negative_ntr:
        lis1.append(label_0_dic[pos_val])
        lis2.append(label_1_dic[pos_val])
        lis3.append(label_2_dic[pos_val])
    pn_avg_label_0['neg'] = sum(lis1)/len(lis1)
    pn_avg_label_1['neg'] = sum(lis2)/len(lis2)
    pn_avg_label_2['neg'] = sum(lis3)/len(lis3)
```

In [27]:
```
pos = [pn_avg_label_0['pos'],pn_avg_label_1['pos'],pn_avg_label_2['
neg = [pn_avg_label_0['neg'],pn_avg_label_1['neg'],pn_avg_label_2['
```

positive nutrition average (should be large)

picking high value cluster as winner

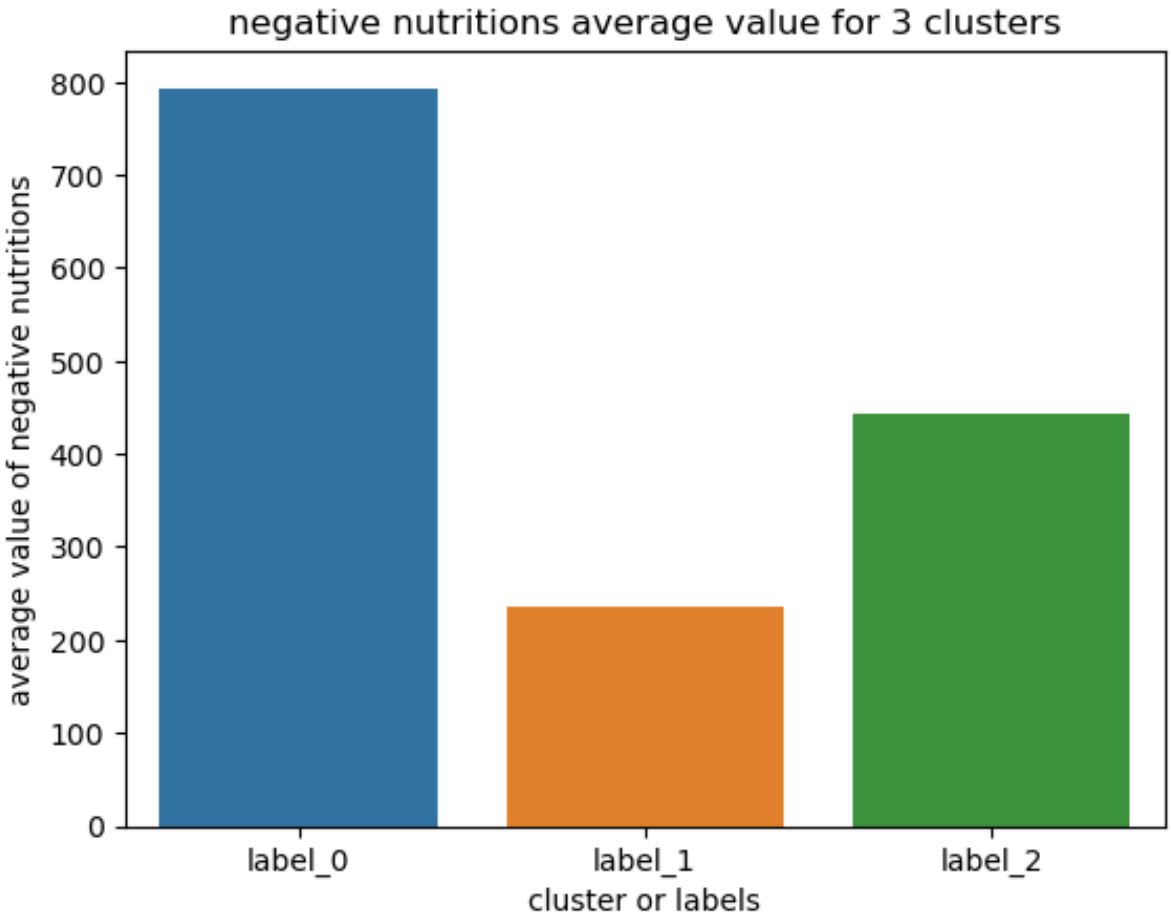
In [28]:
```
sn.barplot(x=["label_0","label_1","label_2"],y=pos)
plt.title("positive nutritions average value for 3 clusters")
plt.xlabel("cluster or labels")
plt.ylabel("average value of positive nutritions")
plt.show()
```



negative nutrition average (should be low)

picking low valued cluster as winner

In [29]:
```
sn.barplot(x=["label_0","label_1","label_2"],y=neg)
plt.title("negative nutritions average value for 3 clusters")
plt.xlabel("cluster or labels")
plt.ylabel("average value of negative nutritions")
plt.show()
```



Now the exciting part (which food should I consume)

> by looking above grphs label 1 having more positive average and less negative average so it get first rank, then label 2 second rank and label 3 will be at last

Healthiest foods

prefer to eat this fast foods rather than others🧐🍔

In [30]:
```
label_1['item'].unique()
```

```
       'Premium Southwest Salad w/ Grilled Chicken',
       'Premium Southwest Salad w/ Crispy Chicken',
       'Chargrilled Chicken Club Sandwich',
       'Chargrilled Chicken Sandwich', 'Chick-n-Slider',
       '1 Piece Chick-n-Strips', '2 Piece Chick-n-Strips',
       '3 Piece Chick-n-Strips', '4 piece Chick-n-Strips',
       '4 piece Chicken Nuggets', '6 piece Chicken Nuggets',
       '8 piece Chicken Nuggets', 'Chicken Salad Sandwich',
       '4 Piece Grilled Chicken Nuggets',
       '6 Piece Grilled Chicken Nuggets',
       '8 piece Grilled Chicken Nuggets',
       '12 Piece Grilled Chicken Nuggets',
       'Regular Grilled Chicken Sub Sandwich',
       'Smokehouse BBQ Bacon Sandwich', 'Chargrilled Chicken Cool
Wrap',
       'Hatch Green Chile Cheeseburger', 'Jalapeno Burger', 'Jr. B
urger',
       'Jr. Chili Cheeseburger', 'Jr. Deluxe Burger',
       'Jr. Deluxe Cheeseburger', 'Sonic Burger W/ Mustard',
       'Sonic Burger W/ Ketchup', 'Sonic Burger W/ Mayonnaise',
```

Medium healthy food

In [31]:
```
label_2['item'].unique()
```

```
       '4 Piece Super Crunch Chicken Strip Dinner',
       'Traditional Ultimate Chicken Sandwich', 'Ultimate Chicken
Club',
       'All Beef Chicago Dog - 6"', 'Cheesy Bacon Pretzel Dog - 6
In.',
       'Footlong Quarter Pound Coney', "Beef 'n Cheddar Classic",
       "Beef 'n Cheddar Mid", 'Bourbon BBQ Brisket Sandwich',
       'Bourbon BBQ Chicken Sandwich', 'Bourbon BBQ Steak Sandwich
',
       'Buttermilk Buffalo Chicken Sandwich',
       'Buttermilk Chicken Bacon & Swiss',
       'Buttermilk Chicken Cordon Bleu Sandwich',
       'Buttermilk Crispy Chicken Sandwich',
       'Classic French Dip & Swiss/Au Jus', 'Double Roast Beef',
       'Fire-Roasted Philly Steak', 'Grand Turkey Club', 'Greek Gy
ro',
       'Half Pound Roast Beef Sandwich', 'Loaded Italian Sandwich'
,
       'Pecan Chicken Salad Sandwich',
```

Less healthy foods

In [32]:
```
label_0['item'].unique()
```

Out[32]:
```
array(['10 piece Buttermilk Crispy Chicken Tenders',
       '12 piece Buttermilk Crispy Chicken Tenders',
       '20 piece Buttermilk Crispy Chicken Tenders',
       '40 piece Chicken McNuggets',
       "6 piece Sweet N' Spicy Honey BBQ Glazed Tenders",
       "10 piece Sweet N' Spicy Honey BBQ Glazed Tenders",
       '30 piece Chicken Nuggets', 'Chicken Enchiladas Meal Kit',
       'Buffalo Dunked Ultimate Chicken Sandwich',
       '5 Piece Super Crunch Chicken Strip Dinner',
       "Half Pound Beef 'n Cheddar Sandwich",
       'Half Pound French Dip & Swiss', 'Triple Decker Sandwich',
       '4 Piece Chicken Strip Basket w/ Country Gravy',
       '6 Piece Chicken Strip Basket w/ Country Gravy',
       'Footlong Big Hot Pastrami', 'Footlong Big Philly Cheeseste
ak',
       'Footlong Carved Turkey & Bacon w/ Cheese',
       'Footlong Corned Beef Reuben', 'Footlong Italian B.M.T.',
       'Footlong Italian Hero', 'Footlong Spicy Italian',
       'Footlong Turkey Italiano Melt (with Provolone)',
       'Turkey, Bacon & Guacamole Wrap'], dtype=object)
```

In [ ]:
```

```