Data_Pre_Processing

April 16, 2021

Objectives

Provide a description of the data pre processing steps. The final dataframe product is the input to the model evaluation steps.

Section 1: Variable Selection

This section selects the variables that are relevant to the analysis. It uses a look up table of the NHANES variables from the input file.

[59]:
```
import pandas as pd
import regex as re
import warnings
from pandas.core.common import SettingWithCopyWarning

warnings.simplefilter(action="ignore", category=SettingWithCopyWarning)


#Read the NHANES full dataset
nhanes_full = pd.read_csv('../../Data/nhanes_full.csv')

'''
Step #1: Remove columns that are not required, using the variable analysis␣
 ↪lookup
Original data set has 823012x86
Transformed data set has 823012x50
'''
#Use the variable lookup to filter out variables for reducing the dataset
variable_lookup_full = pd.read_csv('../../Analysis/
 ↪Variable_Analysis_Lookup_NHANES_full.csv')
#Pull out the priority 0 variables, which are dropped from the start
var_full_pri_0 = variable_lookup_full[variable_lookup_full['Priority'] == 0]
#Pull out the priority 2 variables, which are dropped from the start
var_full_pri_2 = variable_lookup_full[variable_lookup_full['Priority'] == 2]
#Pull out the priority 3 variables, which are dropped from the start
var_full_pri_3 = variable_lookup_full[variable_lookup_full['Priority'] == 3]

#The following variables have been deemed irrelevant for this analysis, so they␣
 ↪are dropped.
```

```
nhanes_full = nhanes_full.drop(var_full_pri_0['Variable'], axis = 1)
nhanes_full = nhanes_full.drop(var_full_pri_2['Variable'], axis = 1)
nhanes_full = nhanes_full.drop(var_full_pri_3['Variable'], axis = 1)
```

Section 2: Extract Non-Aggregated Variables

This section extracts variables that are used later in the analysis, but are not aggregated at the meal level. They are the 'eathome', participant age, and seafood species variables.

[60]:
```
'''
Step: Create age table and eathome table for merging in after aggregation
Creates a table of unique values with each participant and their age
'''
#Extract age for each participant
age_table = nhanes_full[['SEQN','age']]
age_table = age_table.drop_duplicates()


#Define the dataframe join key for pulling other food items consumed with␣
 ↪seafood
meal_key = ['SEQN', 'DR1.030Z', 'DR1.020']

#Create dataframe that contains the eathome variable
df_eathome_key = ['SEQN', 'DR1.030Z', 'DR1.020', 'eathome']
df_eathome = nhanes_full[df_eathome_key]
#Group by unique meal and aggregate the eathome column
df_eathome = df_eathome.groupby(meal_key, as_index=False)['eathome'].sum()
#If eathome was 0, remains 0. Otherwise, convert it to 1
df_eathome['eathome'] = (df_eathome['eathome'] >= 1).astype(int)

'''
Step: Create a table with seafood species per meal.
If a meal has more than one species, keep the one with higher PF_SEAFD_TOT␣
 ↪value and drop others
Each seafood meal then has one type of species for a target
'''


#Extract observations that have a seafood species
df_sf_species = nhanes_full[nhanes_full['species'].notna()]
#Obtain the unique meals + species
df_sf_species= df_sf_species[['SEQN', 'DR1.030Z', 'DR1.020',␣
 ↪'DR1I_PF_SEAFD_TOT', 'species']]
#Find the species with higher PF content within a meal, if there are multiple␣
 ↪species per meal
df_sf_species = df_sf_species.groupby(['SEQN', 'DR1.030Z', 'DR1.020',␣
 ↪'species'], sort=False)['DR1I_PF_SEAFD_TOT'].max()
df_sf_species = df_sf_species.reset_index()
```

```
df_sf_species = df_sf_species.drop(['DR1I_PF_SEAFD_TOT'], axis=1)
#Drop duplicates
df_sf_species = df_sf_species.drop_duplicates(['SEQN', 'DR1.030Z', 'DR1.020',␣
↪'species'])
```

Section 3: Meal Level Aggregation

This section aggregates the FPED variables to the meal level, merging in the 'eathome', 'age', and species variables at the end.

[61]:
```
'''
Step: Perform aggregation.
Create alist of variables that is required.
Filter the dataframe on that list and perform aggregation
'''

#Obtain list of all columns
nhanes_full_cols = nhanes_full.columns.values.tolist()

#Create a list of all the FPED columns, removing the "DR1I_" prefix.
#Create a list of all columns that will be used for meal level aggregation
fped_cols = []
df_agg_columns = []
for i in range(len(nhanes_full_cols)):
    if (re.match(r"DR1I_", nhanes_full_cols[i])):
        fped_cols.append(re.findall(r"DR1I_(.*)", nhanes_full_cols[i])[0])
        df_agg_columns.append(re.findall(r"DR1I_(.*)", nhanes_full_cols[i])[0])



#Add meal key to aggregation cols
df_agg_columns.extend(meal_key)
#Add calories to aggregation cols
df_agg_columns.extend(['DR1IKCAL'])


#Remove the "DR1I_" prefix for simplification
nhanes_full.columns = nhanes_full.columns.str.replace('^DR1I_', '')

#Keep FPED cols,calories, and meal key
nhanes_full= nhanes_full[df_agg_columns]

#Group by meal key and aggregate on FPED columns
nhanes_full = nhanes_full.groupby(meal_key).sum()

'''
Step: Merge in the participant age, eathome, species tables
'''
```

```
#Merge in the eathome variable
nhanes_full = nhanes_full.merge(df_eathome, how='left', on=meal_key)
#Merge in the participant's age
nhanes_full = nhanes_full.merge(age_table, how='left', on=['SEQN'])
#Merge in the seafood species
nhanes_full = nhanes_full.merge(df_sf_species, how='left', on=meal_key)
```

[62]:
```
#Compute dimensions of first step in meal aggregation:
df_length = len(nhanes_full)
df_sfd_length = len(nhanes_full[nhanes_full['PF_SEAFD_TOT'] > 0])
df_non_sfd_length = len(nhanes_full[nhanes_full['PF_SEAFD_TOT'] == 0])
print("The number of total observations at this step is: "+str(df_length))
print("The number of seafood meal observations at this step is:␣
↪"+str(df_sfd_length))
print("The number of non seafood meal observations at this step is:␣
↪"+str(df_non_sfd_length))
```

```
The number of total observations at this step is: 311002
The number of seafood meal observations at this step is: 9505
The number of non seafood meal observations at this step is: 301497
```

Section 4: Obtain Meals of Interest

This section selects the meals of interest, in this case they are lunch and dinner. Some meal names are in Spanish, and they are included based on a translation.

[63]:
```
'''
Step: Keep only meals that are in the lunch or dinner category
'''

#Create meal name variable based on lookup, mapping from CDC source
meal_name_lookup = {1: 'Breakfast', 2: 'Lunch', 3: 'Dinner', 4: 'Supper', 5:
↪'Brunch', 6:'Snack',
                    7: 'Drink', 8: 'Infant Feeding', 9: 'Extended consumption',␣
↪10: 'Desayano',
                    11: 'Almuerzo', 12: 'Comida', 13: 'Merienda', 14: 'Cena',15:
↪ 'Enter comida',
                    16: 'Botana', 17: 'Bocadillo', 18: 'Tentempie', 19:
↪'Bebida', 91: 'Other'}

#Add meal name to dataframe
nhanes_full['Meal_Name'] = nhanes_full['DR1.030Z'].map(meal_name_lookup)

#Obtain the meals of interest
meal_name_filter = ['Lunch', 'Dinner', 'Supper', 'Brunch', 'Almuerzo',␣
↪'Comida', 'Cena', 'Enter comida']
```

```
#Keep only the meals of interest
nhanes_full = nhanes_full[nhanes_full['Meal_Name'].isin(meal_name_filter)]
```

[64]:
```
#Compute dimensions of first step in meal aggregation:
df_length = len(nhanes_full)
df_sfd_length = len(nhanes_full[nhanes_full['PF_SEAFD_TOT'] > 0])
df_non_sfd_length = len(nhanes_full[nhanes_full['PF_SEAFD_TOT'] == 0])
print("The number of total observations at this step is: "+str(df_length))
print("The number of seafood meal observations at this step is:␣
 ↪"+str(df_sfd_length))
print("The number of non seafood meal observations at this step is:␣
 ↪"+str(df_non_sfd_length))
```

```
The number of total observations at this step is: 103592
The number of seafood meal observations at this step is: 8588
The number of non seafood meal observations at this step is: 95004
```

Section 5: Create New Variables

This section creates new variables of interest, based on the existing data. The seafood meal class is created here, along with a category for the meal size. Meals that contain both seafood and meat are re-classified based on a seafood to meat quantity ratio of 1.5 to 1. If the ratio is between 1.5 and the inverse (1/1.5), those meals are dropped because they cannot be classified as either seafood or non-seafood.

[65]:
```
'''
Step: Create seafood meal variable, to categorize if meal has seafood
'''

#Determine if the meal has seafood in it. If yes, variable = 1, 0 otherwise
nhanes_full['seafood_meal'] = (nhanes_full['PF_SEAFD_TOT'] > 0).astype(int)

'''
Step: Meals that contain both meat and seafood
If meals contain both meat and sefood, then compute the ratio of seafood␣
 ↪quantity to meat.
If the ratio is higher than a threshold, classify as seafood meal
If the ratio is lower than a threshold, classify as a non-seafood meal
If the ratio is in a grey area, or between above thresholds, drop that meals␣
 ↪altogether.
Threshold is 1.5 seafood quantity to 1 meat for seafood class and inverse for␣
 ↪non-seafood class
'''

#Create a separate dataframe equal to nhanes full
df_meat_sfd = nhanes_full
#Create meat PF variable, by subtracting the seafood total from the MPS Total
```

```
df_meat_sfd['PF_MPS_TOTAL_NSFD'] = df_meat_sfd['PF_MPS_TOTAL'] -␣
 ↪df_meat_sfd['PF_SEAFD_TOT']
#Find meals that contain both meat and seafood
df_meat_sfd =␣
 ↪df_meat_sfd[(df_meat_sfd['PF_MPS_TOTAL_NSFD']>0)&(df_meat_sfd['PF_SEAFD_TOT']>0)]
#Compute the seafood to meat ratio
df_meat_sfd['seafood_meat_ratio'] = df_meat_sfd['PF_MPS_TOTAL_NSFD']/
 ↪df_meat_sfd['PF_SEAFD_TOT']
#Create a temporary class to classify based on the ratio
df_meat_sfd.loc[df_meat_sfd['seafood_meat_ratio']>1.5, 'seafood_class'] = "Yes"
df_meat_sfd.loc[df_meat_sfd['seafood_meat_ratio']<(1/1.5), 'seafood_class'] =␣
 ↪"No"
df_meat_sfd.loc[df_meat_sfd['seafood_class'].isna(), 'seafood_class'] = "Maybe"
#Only keep the meal key and the new class, drop duplicates
df_meat_sfd = df_meat_sfd[['SEQN', 'DR1.030Z', 'DR1.020', 'seafood_class']]
df_meat_sfd = df_meat_sfd.drop_duplicates(['SEQN', 'DR1.030Z', 'DR1.020',␣
 ↪'seafood_class'])
#Merge the new class into the dataframe
nhanes_full = nhanes_full.merge(df_meat_sfd, how='left', on=meal_key)

#Remove meals that are in the grey area between thresholds
nhanes_full = nhanes_full[nhanes_full['seafood_class'] != 'Maybe']
#Recode the seafood class based on the results from these observations
nhanes_full.loc[nhanes_full.seafood_class == 'No', ['seafood_meal']] = 0
#Drop the temporary class
nhanes_full = nhanes_full.drop(['seafood_class'], axis=1)

'''
Step: Create a meal energy variable
'''

#Create meal energy category based on quantiles from KCAL
nhanes_full.loc[nhanes_full['DR1IKCAL'] < nhanes_full['DR1IKCAL'].
 ↪describe()['25%'], 'meal_energy'] = "Low"
nhanes_full.loc[(nhanes_full['DR1IKCAL'] > nhanes_full['DR1IKCAL'].
 ↪describe()['25%'])
      & (nhanes_full['DR1IKCAL'] < nhanes_full['DR1IKCAL'].describe()['50%']),␣
 ↪'meal_energy'] = "Medium-Low"
nhanes_full.loc[(nhanes_full['DR1IKCAL'] > nhanes_full['DR1IKCAL'].
 ↪describe()['50%'])
      & (nhanes_full['DR1IKCAL'] < nhanes_full['DR1IKCAL'].describe()['75%']),␣
 ↪'meal_energy'] = "Medium-High"
nhanes_full.loc[nhanes_full['DR1IKCAL'] > nhanes_full['DR1IKCAL'].
 ↪describe()['75%'], 'meal_energy'] = "High"
```

[66]:
```
#Compute dimensions of first step in meal aggregation:
df_length = len(nhanes_full)
df_sfd_length = len(nhanes_full[nhanes_full['PF_SEAFD_TOT'] > 0])
df_non_sfd_length = len(nhanes_full[nhanes_full['PF_SEAFD_TOT'] == 0])
print("The number of total observations at this step is: "+str(df_length))
print("The number of seafood meal observations at this step is:␣
 ↪"+str(df_sfd_length))
print("The number of non seafood meal observations at this step is:␣
 ↪"+str(df_non_sfd_length))
```

```
The number of total observations at this step is: 102944
The number of seafood meal observations at this step is: 7940
The number of non seafood meal observations at this step is: 95004
```

Section 6: Additional Observation Filters

This section filters out additional observations:

1. Drop meals that are 0 calories. These meals are assumed to be water only, and therefore are not observations of interest.

[67]:
```
'''
Step: Drop meals that are 0 calories
'''
nhanes_full = nhanes_full[nhanes_full['DR1IKCAL']>0]
```

[68]:
```
#Compute dimensions of first step in meal aggregation:
df_length = len(nhanes_full)
df_sfd_length = len(nhanes_full[nhanes_full['PF_SEAFD_TOT'] > 0])
df_non_sfd_length = len(nhanes_full[nhanes_full['PF_SEAFD_TOT'] == 0])
print("The number of total observations at this step is: "+str(df_length))
print("The number of seafood meal observations at this step is:␣
 ↪"+str(df_sfd_length))
print("The number of non seafood meal observations at this step is:␣
 ↪"+str(df_non_sfd_length))
```

```
The number of total observations at this step is: 101963
The number of seafood meal observations at this step is: 7940
The number of non seafood meal observations at this step is: 94023
```

2. Drop participants that are below the age of 18. Adult participants are more likely to make their own choices when it comes to food alternatives. Non-adults are not observations of interest.

[69]:
```
'''
Step: Drop meals of participants below the age of 18
'''
nhanes_full = nhanes_full[nhanes_full['age'] >= 18]
```

[70]:
```
#Compute dimensions of first step in meal aggregation:
df_length = len(nhanes_full)
df_sfd_length = len(nhanes_full[nhanes_full['PF_SEAFD_TOT'] > 0])
df_non_sfd_length = len(nhanes_full[nhanes_full['PF_SEAFD_TOT'] == 0])
print("The number of total observations at this step is: "+str(df_length))
print("The number of seafood meal observations at this step is:␣
 ↪"+str(df_sfd_length))
print("The number of non seafood meal observations at this step is:␣
 ↪"+str(df_non_sfd_length))
```

```
The number of total observations at this step is: 61144
The number of seafood meal observations at this step is: 6197
The number of non seafood meal observations at this step is: 54947
```

3. Drop meals that are not made at home. It is more likely that participants who are eating outside, have less alternatives on what to eat with seafood as opposed to non-seafood. This because these choices are mostly restricted by a restaurant menu.

[71]:
```
'''
Step: Drop meals that are not home made
'''
nhanes_full = nhanes_full[nhanes_full['eathome']==1]
```

[72]:
```
#Compute dimensions of first step in meal aggregation:
df_length = len(nhanes_full)
df_sfd_length = len(nhanes_full[nhanes_full['PF_SEAFD_TOT'] > 0])
df_non_sfd_length = len(nhanes_full[nhanes_full['PF_SEAFD_TOT'] == 0])
print("The number of total observations at this step is: "+str(df_length))
print("The number of seafood meal observations at this step is:␣
 ↪"+str(df_sfd_length))
print("The number of non seafood meal observations at this step is:␣
 ↪"+str(df_non_sfd_length))
```

```
The number of total observations at this step is: 41349
The number of seafood meal observations at this step is: 3634
The number of non seafood meal observations at this step is: 37715
```

4. Drop meals that are vegeterian, ir order to compare seafood meals with meals that contain other meats.

[73]:
```
'''
Step: Drop meals that are vegeterian
'''
#Drop meals that do not contain any type of meat or seafood
nhanes_full = nhanes_full[nhanes_full['PF_MPS_TOTAL'] > 0]
```

[74]:
```
#Compute dimensions of first step in meal aggregation:
df_length = len(nhanes_full)
df_sfd_length = len(nhanes_full[nhanes_full['PF_SEAFD_TOT'] > 0])
```

```
df_non_sfd_length = len(nhanes_full[nhanes_full['PF_SEAFD_TOT'] == 0])
print("The number of total observations at this step is: "+str(df_length))
print("The number of seafood meal observations at this step is:␣
 ↪"+str(df_sfd_length))
print("The number of non seafood meal observations at this step is:␣
 ↪"+str(df_non_sfd_length))
```

```
The number of total observations at this step is: 29243
The number of seafood meal observations at this step is: 3634
The number of non seafood meal observations at this step is: 25609
```

Section 7: Outstanding Modifications

Convert units of FPED components to a more common unit of measurement.

[ ]: