The National Health and Nutrition Examination Survey Analysis

In [1]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import tree, svm
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB, MultinomialNB
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve, auc, roc_auc_score, f1_score
from sklearn.inspection import permutation_importance

import xgboost as xgb
from xgboost import XGBClassifier, plot_importance
from hyperopt import fmin, hp, tpe
```

Importing and Joining data

In [2]:
```
#loading the dataset in pandas dataset
labs = pd.read_csv('labs.csv')
exam = pd.read_csv('examination.csv')
demo = pd.read_csv('demographic.csv')
diet = pd.read_csv('diet.csv')
ques = pd.read_csv('questionnaire.csv')

exam.drop(['SEQN'], axis = 1, inplace=True)
demo.drop(['SEQN'], axis = 1, inplace=True)
diet.drop(['SEQN'], axis = 1, inplace=True)
ques.drop(['SEQN'], axis = 1, inplace=True)

df = pd.concat([labs, exam], axis=1, join='inner')
df = pd.concat([df, demo], axis=1, join='inner')
df = pd.concat([df, diet], axis=1, join='inner')
df = pd.concat([df, ques], axis=1, join='inner')
```

In [3]:
```
# SI columns are duplicate columns that give unit conversions
cols = [c for c in df.columns if c[-2:] != 'SI']
df = df[cols]
```

In [4]:
```
# Removing highly specific columns, HPV type, specific tooth missin
drop_cols = ['ORXGH', 'ORXGL', 'ORXH06', 'ORXH11', 'ORXH16', 'ORXH1
             'ORXH54', 'ORXH55', 'ORXH56', 'ORXH58', 'ORXH59', 'ORX
             'ORXH81', 'ORXH82', 'ORXH83', 'ORXH84', 'ORXHPC', 'ORX
             'OHX10TC', 'OHX11TC', 'OHX12TC', 'OHX13TC', 'OHX14TC',
             'OHX25TC', 'OHX26TC', 'OHX27TC', 'OHX28TC', 'OHX29TC',
             'OHX08CTC', 'OHX09CTC', 'OHX10CTC', 'OHX11CTC', 'OHX12
             'OHX23CTC', 'OHX24CTC', 'OHX25CTC', 'OHX26CTC', 'OHX27
             'PHAFSTMN.x','SEQN','RIDSTATR'
            ]
df.drop(drop_cols, axis=1,inplace=True)
```

Decoding Columns

In [5]:
```
col_decoder = {
'ACD011A' : 'speak_english',
'AIALANGA' : 'speak_english2',
'ALQ101' : 'drink_alcohol',
'ALQ130' : 'alcohol_per_day',
'AUQ136' : 'ear_infections',
'BMDAVSAD' : 'saggital_abdominal_avg',
'BMXARMC' : 'arm_circum',
'BMXBMI' : 'BMI',
'BMXLEG' : 'leg_length',
'BMXSAD1' : 'saggital_abdominal_1',
'BMXSAD2' : 'saggital_abdominal_2',
'BMXWAIST' : 'waist_circum',
'BMXWT' : 'weight_kg',
'BPQ020' : 'high_bp',
'BPQ056' : 'measure_bp_home',
'BPQ059' : 'measure_bp_doctor',
'BPQ060' : 'cholesterol_checked',
'BPQ070' : 'cholesterol_checked_1y',
'BPQ080' : 'high_cholesterol',
'BPQ090D' : 'cholesterol_prescription',
'BPXDI1' : 'diastolic_bp',
'BPXML1' : 'cuff_max_inflation',
'BPXSY1' : 'blood_pressure_1',
'BPXSY2' : 'blood_pressure_2',
'BPXSY3' : 'blood_pressure_3',
'CBD070' : 'grocery_budget',
'CBD090' : 'nonfood_budget',
'CBD110' : 'food_budget',
'CBD120' : 'restaurant_budget',
'CBD130' : 'food_delivery_budget',
'CBQ505' : 'fast_food',
'CBQ535' : 'saw_nutrition_fast_food',
'CBQ545' : 'use_nutrition_fast_food',
'CBQ550' : 'eat_restaurants',
'CBQ552' : 'eat_chain_restaurants',
'CBQ580' : 'saw_nutrition_restaurant',
```

```
'CBQ590' : 'use_nutrition_restaurant',
'CBQ596' : 'saw_my_plate',
'CDQ001' : 'chest_pain_ever',
'CDQ010' : 'short_breath_stairs',
'CSQ030' : 'sensative_smell',
'CSQ100' : 'loss_of_taste',
'CSQ110' : 'taste_in_mouth',
'CSQ202' : 'dry_mouth',
'CSQ204' : 'nasal_congestion',
'CSQ210' : 'wisdom_teeth_removed',
'CSQ220' : 'tonsils_removed',
'CSQ240' : 'head_injury',
'CSQ250' : 'broken_nose',
'CSQ260' : 'sinus_infections',
'DBD100' : 'salt_frequency',
'DBD895' : 'meals_not_homemade',
'DBD900' : 'meals_fast_food',
'DBD905' : 'meals_prepackaged',
'DBD910' : 'frozen_meals_per_month',
'DBQ095Z' : 'salt_type',
'DBQ197' : 'milk_product_per_month',
'DBQ229' : 'milk_drinker',
'DBQ700' : 'healthy_diet',
'DEQ034C' : 'long_sleeve_shirt',
'DEQ034D' : 'use_sunscreen',
'DEQ038G' : 'sunburn_1y',
'DIQ010' : 'diabetes',
'DIQ050' : 'taking_insulin',
'DIQ160' : 'prediabetes',
'DIQ170' : 'diabetes_risk',
'DIQ172' : 'diabetes_concern',
'DIQ180' : 'blood_test_3y',
'DLQ010' : 'deaf',
'DLQ020' : 'blind',
'DLQ040' : 'mental_issues',
'DLQ050' : 'difficulty_walking',
'DLQ060' : 'difficulty_dressing',
'DLQ080' : 'difficulty_errands',
'DMDBORN4' : 'born_in_us2',
'DMDHHSIZ' : 'people_in_house',
'DMDHHSZB' : 'children_in_house',
'DMDHHSZE' : 'people_over_60_in_house',
'DMDHRBR4' : 'born_in_us',
'DMDHRGND' : 'gender2',
'DMDMARTL' : 'Marital_Status',
'DMDYRSUS' : 'years_in_US',
'DPQ010' : 'no_interest_2w',
'DPQ020' : 'depression',
'DPQ030' : 'trouble_sleeping_2w',
'DPQ040' : 'fatigue_2w',
'DPQ050' : 'eating_problems_2w',
'DPQ060' : 'feel_bad_2w',
'DPQ070' : 'trouble_concentrating_2w',
'DPQ080' : 'speaking_problems_2w',
```

```
'DPQ090' : 'suicidal_2w',
'DPQ100' : 'depression_difficulty',
'DR1.320Z' : 'water',
'DR1_320Z' : 'plain_water_yesterday',
'DR1_330Z' : 'tap_water_yesterday',
'DR1BWATZ' : 'bottled_water_yesterday',
'DR1HELPD' : 'interview_help',
'DR1TACAR' : 'dietary_alpha_carotene',
'DR1TALCO' : 'alcohol',
'DR1TATOC' : 'dietary_vitamin_e',
'DR1TBCAR' : 'dietary_beta_carotene',
'DR1TCAFF' : 'caffeine',
'DR1TCALC' : 'dietary_calcium',
'DR1TCARB' : 'carb',
'DR1TCHL' : 'dietary_choline',
'DR1TCHOL' : 'cholesterol',
'DR1TCOPP' : 'dietary_copper',
'DR1TCRYP' : 'dietary_beta_cryptoxanthin',
'DR1TFA' : 'dietary_folic_acid',
'DR1TFF' : 'folate_food',
'DR1TFIBE' : 'fiber',
'DR1TFOLA' : 'dietary_folate',
'DR1TIRON' : 'dietary_iron',
'DR1TKCAL' : 'calories',
'DR1TLYCO' : 'dietary_lycopene',
'DR1TLZ' : 'dietary_lutein',
'DR1TM181' : 'octadecenoic_percent',
'DR1TMAGN' : 'magnesium',
'DR1TMFAT' : 'monounsaturated_fats',
'DR1TMOIS' : 'moisture',
'DR1TNIAC' : 'dietary_niacin',
'DR1TP183' : 'octadecatrienoic_percent',
'DR1TPHOS' : 'dietary_phosphorus',
'DR1TPOTA' : 'dietary_potassium',
'DR1TPROT' : 'protein',
'DR1TRET' : 'dietary_retinol',
'DR1TS140' : 'tetradeconoic_percent',
'DR1TSELE' : 'dietary_selenium',
'DR1TSODI' : 'sodium',
'DR1TSUGR' : 'sugar',
'DR1TTFAT' : 'fat',
'DR1TTHEO' : 'dietary_theobromine',
'DR1TVARA' : 'dietary_vitamin_a',
'DR1TVB1' : 'dietary_b1',
'DR1TVB12' : 'dietary_b12',
'DR1TVB2' : 'dietary_b2',
'DR1TVB6' : 'dietary_b6',
'DR1TVC' : 'dietary_vit_c',
'DR1TVD' : 'dietary_vit_d',
'DR1TVK' : 'dietary_vit_k',
'DR1TZINC' : 'dietary_zinc',
'DRABF' : 'breast_fed',
'DRD340' : 'shellfish',
'DRD350A' : 'clams',
```

```
'DRD350B' : 'crabs',
'DRD350C' : 'crayfish',
'DRD350D' : 'lobsters',
'DRD350E' : 'mussels',
'DRD350F' : 'oysters',
'DRD350G' : 'scallops',
'DRD350H' : 'shrimp',
'DRD370A' : 'breaded_fish',
'DRD370B' : 'tuna',
'DRD370C' : 'bass',
'DRD370D' : 'catfish',
'DRD370E' : 'cod',
'DRD370F' : 'flatfish',
'DRD370G' : 'haddock',
'DRD370H' : 'mackerel',
'DRD370I' : 'perch',
'DRD370J' : 'pike',
'DRD370K' : 'pollock',
'DRD370L' : 'porgy',
'DRD370M' : 'salmon',
'DRD370N' : 'sardines',
'DRD370O' : 'sea_bass',
'DRD370P' : 'shark',
'DRD370Q' : 'swordfish',
'DRD370R' : 'trout',
'DRD370S' : 'walleye',
'DRD370T' : 'other_fish',
'DRQSDIET' : 'special_diet',
'DRQSDT1' : 'low_cal_diet',
'DRQSDT10' : 'high_protein_diet',
'DRQSDT11' : 'low_gluten_diet',
'DRQSDT12' : 'kidney_diet',
'DRQSDT2' : 'low_fat_diet',
'DRQSDT3' : 'low_salt_diet',
'DRQSDT4' : 'low_sugar_diet',
'DRQSDT5' : 'low_fiber_diet',
'DRQSDT6' : 'high_fiber_diet',
'DRQSDT7' : 'diabetic_diet',
'DRQSDT8' : 'muscle_diet',
'DRQSDT9' : 'low_carb_diet',
'DRQSDT91' : 'other_diet',
'DRQSPREP' : 'salt_used',
'DUQ200' : 'marijuana',
'DUQ370' : 'needle_drugs',
'FSD032A' : 'food_insecure',
'FSD032B' : 'not_enough_food',
'FSD032C' : 'cheap_food',
'FSD032D' : 'cheap_food_children',
'FSD032E' : 'bad_food_children',
'FSD032F' : 'low_food_children',
'FSD151' : 'emergency_food_received',
'FSDAD' : 'food_secure',
'FSDCH' : 'child_food_secure',
'FSDHH' : 'household_food_secure',
```

```
'FSQ162' : 'wic_received',
'FSQ165' : 'food_stamps',
'HEQ010' : 'hepetitis_b',
'HEQ030' : 'hepetitis_c',
'HIQ011' : 'health_insurance',
'HIQ210' : 'insurance_gap',
'HIQ270' : 'prescription_insurance',
'HOD050' : 'rooms_in_home',
'HOQ065' : 'homeowner',
'HSAQUEX' : 'health_status_source_data',
'HSD010' : 'general_health',
'HSQ500' : 'ever_had_cold',
'HSQ510' : 'intestinal_illness',
'HSQ520' : 'ever_had_flu',
'HSQ571' : 'donate_blood',
'HSQ590' : 'hiv',
'HUQ010' : 'general_health2',
'HUQ020' : 'health_compared_last_year',
'HUQ030' : 'routine_healthcare',
'HUQ041' : 'healthcare_location',
'HUQ051' : 'dr_visits',
'HUQ071' : 'overnight_hospital',
'HUQ090' : 'mental_health_treatment',
'IMQ011' : 'hepatitis_a_vaccine',
'IMQ020' : 'hepatitis_b_vaccine',
'IND235' : 'monthly_income',
'INDFMMPC' : 'poverty_level_category',
'INDFMMPI' : 'poverty_level_index',
'INDFMPIR' : 'family_income',
'INQ012' : 'self_employ_income',
'INQ020' : 'income_from_wages',
'INQ030' : 'income_from_SS',
'INQ060' : 'disability_income',
'INQ080' : 'retirement_income',
'INQ090' : 'ss_income',
'INQ132' : 'state_assistance_income',
'INQ140' : 'investment_income',
'INQ150' : 'other_income',
'INQ244' : 'family_savings',
'LBDBCDLC' : 'blood_cadmium',
'LBDBGMLC' : 'methyl_mercury',
'LBDHDD' : 'HDL_mg',
'LBDIHGLC' : 'inorganic_mercury',
'LBDNENO' : 'neutrophils_percent',
'LBDTHGLC' : 'blood_mercury',
'LBDWFL' : 'floride_water',
'LBXEOPCT' : 'eosinophils_percent',
'LBXGH' : 'glyco_hemoglobin',
'LBXLYPCT' : 'lymphocite_percent',
'LBXMC' : 'hemoglobin_concentration',
'LBXSAL' : 'blood_albumin',
'LBXSCA' : 'blood_calcium',
'LBXSGL' : 'serum_glucose_mg',
'LBXSTP' : 'blood_protein',
```

```
'MCQ010' : 'asthma_ever',
'MCQ025' : 'asthma_age',
'MCQ035' : 'asthma',
'MCQ040' : 'asthma_year',
'MCQ050' : 'asthma_ER',
'MCQ053' : 'anemia',
'MCQ070' : 'psoriasis',
'MCQ080' : 'overweight',
'MCQ082' : 'celiac_disease',
'MCQ086' : 'gluten_free',
'MCQ092' : 'blood_transfusion',
'MCQ149' : 'menstruate',
'MCQ151' : 'menstruate_age',
'MCQ160A' : 'arthritis',
'MCQ160B' : 'congestive_heart_failure',
'MCQ160C' : 'coronary_heart_disease',
'MCQ160D' : 'angina',
'MCQ160E' : 'heart_attack',
'MCQ160F' : 'stroke',
'MCQ160G' : 'emphysema',
'MCQ160K' : 'bronchitis_ever',
'MCQ160L' : 'liver_condition_ever',
'MCQ160M' : 'thyroid_ever',
'MCQ160N' : 'gout',
'MCQ160O' : 'COPD',
'MCQ170K' : 'bronchitis_now',
'MCQ170L' : 'liver_condition',
'MCQ170M' : 'thyroid_now',
'MCQ180A' : 'arthritis_age',
'MCQ180B' : 'heart_failure_age',
'MCQ180C' : 'heart_disease_age',
'MCQ180D' : 'angina_age',
'MCQ180E' : 'heart_attack_age',
'MCQ180F' : 'stroke_age',
'MCQ180G' : 'emphysema_age',
'MCQ180K' : 'bronchitis_age',
'MCQ180L' : 'liver_condition_age',
'MCQ180M' : 'thyroid_age',
'MCQ180N' : 'gout_age',
'MCQ195' : 'arthritis_type',
'MCQ203' : 'jaundice',
'MCQ206' : 'jaundice_age',
'MCQ220' : 'cancer',
'MCQ230A' : 'cancer_type1',
'MCQ230B' : 'cancer_type2',
'MCQ230C' : 'cancer_type3',
'MCQ230D' : 'cancer_type4',
'MCQ240A' : 'bladder_cancer_age',
'MCQ240AA' : 'test_cancer_age',
'MCQ240B' : 'blood_cancer_age',
'MCQ240BB' : 'thyroid_cancer_age',
'MCQ240C' : 'bone_cancer_age',
'MCQ240CC' : 'uterine_cancer_age',
'MCQ240D' : 'brain_cancer_age',
```

```
'MCQ240DK' : 'cancer_age',
'MCQ240E' : 'breast_cancer_age',
'MCQ240F' : 'cervical_cancer_age',
'MCQ240G' : 'colon_cancer_age',
'MCQ240H' : 'esoph_cancer_age',
'MCQ240I' : 'gallbladder_cancer_age',
'MCQ240J' : 'kidney_cancer_age',
'MCQ240K' : 'larynx_cancer_age',
'MCQ240L' : 'leukemia_age',
'MCQ240M' : 'liver_cancer_age',
'MCQ240N' : 'lung_cancer_age',
'MCQ240O' : 'lymphoma_age',
'MCQ240P' : 'melanoma_age',
'MCQ240Q' : 'mouth_cancer_age',
'MCQ240R' : 'nervous_cancer_age',
'MCQ240S' : 'ovarian_cancer_age',
'MCQ240T' : 'pancreatic_cancer_age',
'MCQ240U' : 'prostate_cancer_age',
'MCQ240V' : 'rectal_cancer_age',
'MCQ240X' : 'skin_cancer_age',
'MCQ240Y' : 'soft_cancer_age',
'MCQ240Z' : 'stomach_cancer_age',
'MCQ300A' : 'relative_heart_attack',
'MCQ300B' : 'relative_asthma',
'MCQ300C' : 'relative_diabetes',
'MCQ365A' : 'need_weight_loss',
'MCQ365B' : 'need_exercise',
'MCQ365C' : 'need_reduce_salt',
'MCQ365D' : 'need_reduce_calories',
'MCQ370A' : 'losing_weight',
'MCQ370B' : 'excercising',
'MCQ370C' : 'reducing_salt',
'MCQ370D' : 'reducing_fat',
'MGDCGSZ' : 'grip_strength',
'OCD150' : 'work_done',
'OCD270' : 'months_of_work',
'OCD390G' : 'type_of_work',
'OCD395' : 'job_duration',
'OCQ260' : 'non_govt_employee',
'OHQ030' : 'visit_dentist',
'OHQ033' : 'dentist_reason',
'OHQ620' : 'aching_mouth',
'OHQ640' : 'mouth_problems',
'OHQ680' : 'mouth_problems2',
'OHQ770' : 'need_dental',
'OHQ835' : 'gum_disease',
'OHQ845' : 'teeth_health',
'OHQ850' : 'gum_treatment',
'OHQ855' : 'loose_teeth',
'OHQ860' : 'teeth_bone_loss',
'OHQ865' : 'weird_tooth',
'OHQ870' : 'floss',
'OHQ875' : 'use_mouthwash',
'OHQ880' : 'oral_cancer_exam',
```

```
'OHQ885' : 'oral_cancer_exam2',
'OSQ060' : 'osteoporosis',
'OSQ130' : 'take_prednisone',
'OSQ230' : 'metal_objects',
'PAAQUEX' : 'question_source',
'PAD680' : 'sedentary_time',
'PAQ605' : 'vigorous_work',
'PAQ620' : 'moderate_work',
'PAQ635' : 'walk_or_bike',
'PAQ650' : 'vigorous_recreation',
'PAQ665' : 'moderate_recreation',
'PAQ710' : 'tv_hours',
'PAQ715' : 'pc_hours',
'PEASCST1' : 'bp_status',
'PEASCTM1' : 'blood_pressure_time',
'PFQ049' : 'work_limitations',
'PFQ051' : 'work_limitations2',
'PFQ054' : 'walk_equipment_required',
'PFQ057' : 'confusion_memory_problems',
'PFQ090' : 'special_healthcare_equipment',
'PUQ100' : 'insecticide_used',
'PUQ110' : 'weedkiller_used',
'RIAGENDR' : 'gender',
'RIDAGEYR' : 'age',
'RIDRETH1' : 'hispanic',
'RXQ510' : 'take_aspirin',
'SEQN' : 'ID',
'SLD010H' : 'sleep_hours',
'SLQ050' : 'trouble_sleeping',
'SLQ060' : 'sleep_disorder',
'SMAQUEX.x' : 'question_mode',
'SMAQUEX.y' : 'question_mode2',
'SMAQUEX2' : 'question_mode3',
'SMD460' : 'smokers_in_house',
'SMDANY' : 'tobaco_1w',
'SMQ681' : 'smoked_1w',
'SMQ851' : 'tobaco2_1w',
'SMQ856' : 'smoked_at_work',
'SMQ858' : 'someone_smoked_at_job',
'SMQ860' : 'smoked_at_restaurant',
'SMQ863' : 'nicotine_1w',
'SMQ866' : 'smoked_at_bar',
'SMQ870' : 'smoked_in_car',
'SMQ872' : 'someone_smoked_in_car',
'SMQ874' : 'smoked_another_home',
'SMQ876' : 'someone_smoked_in_home',
'SMQ878' : 'smoked_other_building',
'SMQ880' : 'someone_smoked_other_building',
'SXD021' : 'sex_ever',
'URXUCR' : 'creatinine_urine',
'WHD010' : 'height_in',
'WHD020' : 'current_weight_lb',
'WHD050' : 'weight_1y',
'WHD110' : 'weight_10y'
```

```
'WHD110' : 'weight_10y',
'WHD120' : 'weight_age_25',
'WHD140' : 'greatest_weight',
'WHQ030' : 'overweight_self',
'WHQ040' : 'weightloss_desire',
'WHQ070' : 'weightloss_attempt',
'WHQ150' : 'age_when_heaviest'
}
df = df.rename(columns = col_decoder)
labs = labs.rename(columns = col_decoder)
exam = exam.rename(columns = col_decoder)
demo = demo.rename(columns = col_decoder)
diet = diet.rename(columns = col_decoder)
ques = ques.rename(columns = col_decoder)
```

In [6]:
```
cancer_df = df.dropna(subset=['cancer'])
diabetes_df = df.dropna(subset=['diabetes'])
heart_df = df.dropna(subset=['coronary_heart_disease'])
liver_df = df.dropna(subset=['liver_condition'])
```

In [7]:
```
target_dfs = [cancer_df, diabetes_df, heart_df, liver_df]
```

Handling Nulls and Category columns

Combinations of 7s and 9s are used when data is not applicable or when the patient refused to answer

In [8]:
```
for df in target_dfs:
    df.replace({7:None, 9:None, 77:None,99:None,777:None,999:None,7
            777777:None,999999:None,55:None,555:None,5555:None,8:No
```

In [9]:
```
df.replace({7:None, 9:None, 77:None,99:None,777:None,999:None,7777:
            777777:None,999999:None,55:None,555:None,5555:None,8:No
```

Remove columns and rows with excessive nulls

In [10]:
```
def filter_columns(df, cutoff=0.9):
    tot_rows = df.shape[0]
    removed_cols = []
    print("original number of columns: ", df.shape[1])
    for col in df.columns:
        num_na = df[col].isna().sum()
        if (num_na/tot_rows) > cutoff:
            #print(col, df[col].isna().sum())
            removed_cols.append(col)
    print("number of columns removed: ", len(removed_cols))
    return df.drop(removed_cols, axis=1)

def filter_rows(df, cutoff=0.9):
    tot_cols = df.shape[1]
    print("original number of rows: ", df.shape[0])
    df = df[df.isnull().sum(axis=1) < tot_cols*cutoff]
    print("remaining rows: ", df.shape[0])
    return df
```

In [11]:
```
#df = df[df.age > 18]
```

In [12]:
```
def trans_cat_cols(df, cat_cols):
    for col in cat_cols:
        df.loc[df[col] != 1, col] = 0
    return df
```

In [13]:
```
x = df.nunique()
cat_cols = x[(x<15)].index
df = trans_cat_cols(df, cat_cols)
```

In [14]:
```
#df.loc[df[col] != 1, col] = 0
```

In [15]:
```
for df in target_dfs:
    x = df.nunique()
    cat_cols = x[(x<15)].index
    df = trans_cat_cols(df, cat_cols)
```

In [16]:
```
for df in target_dfs:
    df = filter_rows(df, cutoff=0.8)
    df = filter_columns(df, cutoff=0.5)
```

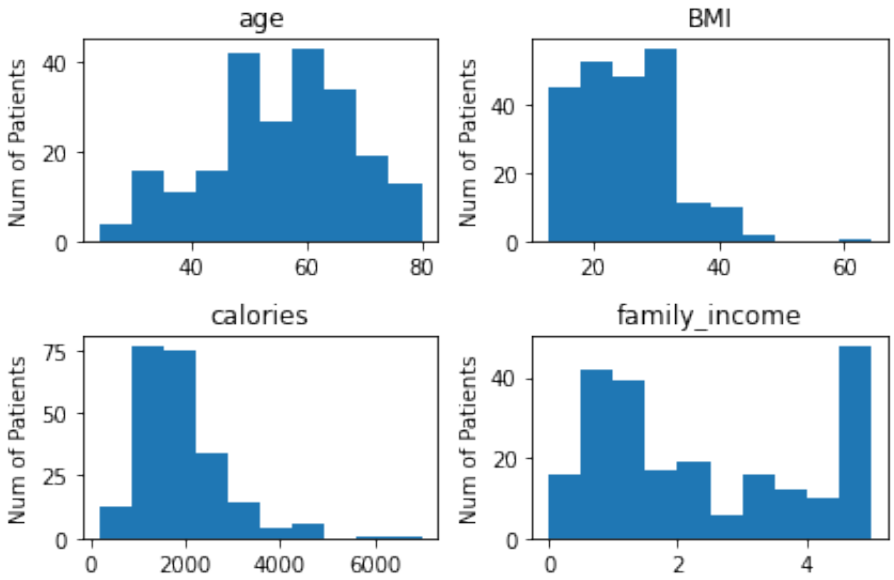
```
original number of rows:  5561
remaining rows:  5561
original number of columns:  1660
number of columns removed:  259
original number of rows:  9422
remaining rows:  9422
original number of columns:  1660
number of columns removed:  281
original number of rows:  5561
remaining rows:  5561
original number of columns:  1660
number of columns removed:  259
original number of rows:  225
remaining rows:  225
original number of columns:  1660
number of columns removed:  143
```

In [17]:
```
for df in target_dfs:
    df.fillna(df.mode().iloc[0], inplace=True)
```

Exploring Factors

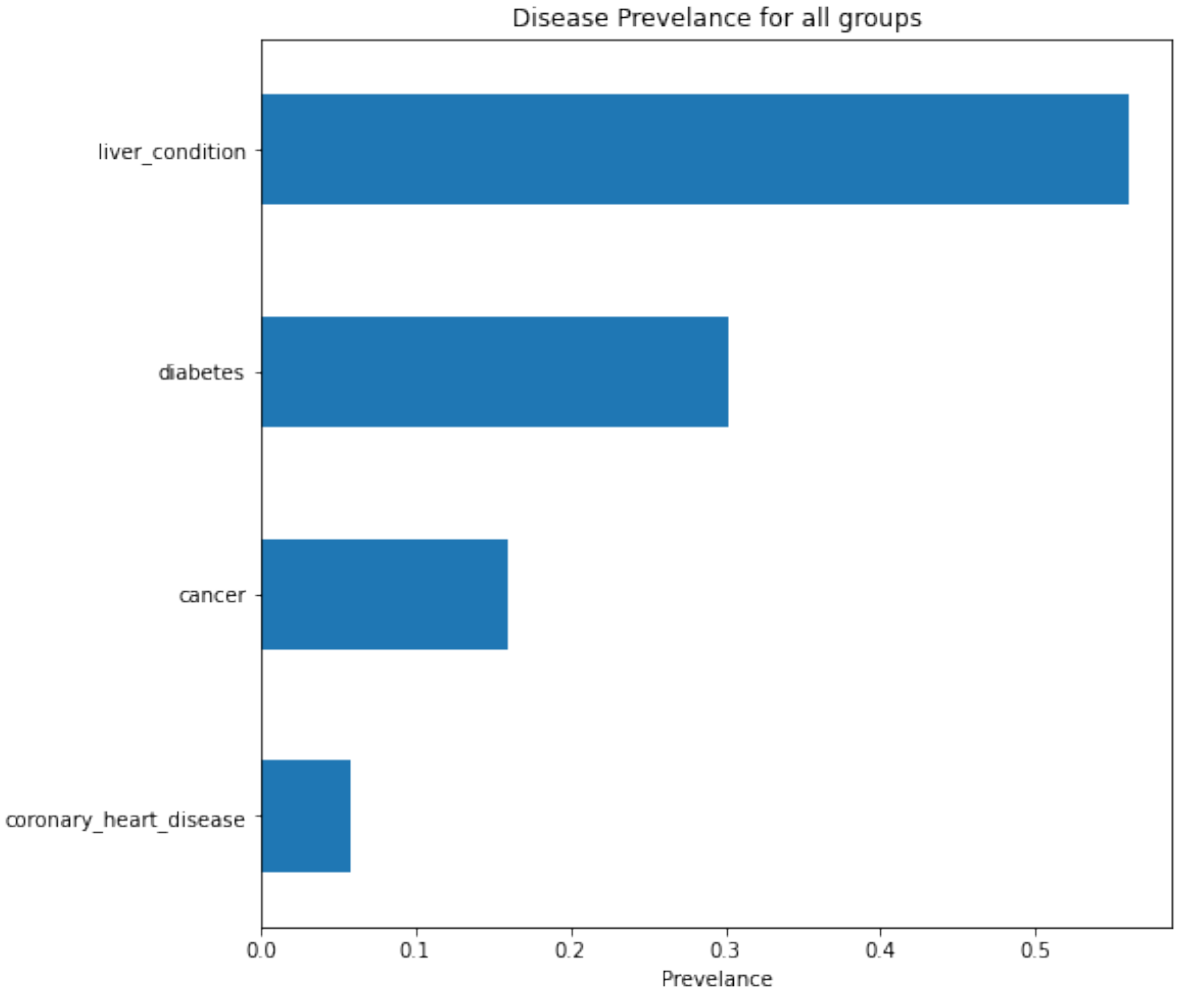
In [18]:
```
lifestyle_cols = ['drink_alcohol','alcohol_per_day','saggital_abdom
                  'saggital_abdominal_1','saggital_abdominal_2','wa
                  'grocery_budget','nonfood_budget','food_budget',
                  'restaurant_budget','food_delivery_budget','fast_
                  'use_nutrition_fast_food','eat_restaurants','eat_
                  'use_nutrition_restaurant','saw_my_plate','wisdom
                  'salt_frequency','meals_not_homemade','meals_fast
                  'frozen_meals_per_month','salt_type','milk_produc
                  'long_sleeve_shirt','use_sunscreen','people_in_ho
                  'trouble_sleeping_2w','eating_problems_2w','water
                  'bottled_water_yesterday','dietary_alpha_carotene
                  'dietary_beta_carotene','caffeine','dietary_calci
                  'dietary_copper','dietary_beta_cryptoxanthin','di
                  'dietary_folate','dietary_iron','calories','dieta
                  'octadecenoic_percent','magnesium','monounsaturat
                  'octadecatrienoic_percent','dietary_phosphorus','
                  'tetradeconoic_percent','dietary_selenium','sodiu
                  'dietary_vitamin_a','dietary_b1','dietary_b12','d
                  'dietary_vit_d','dietary_vit_k','dietary_zinc','s
                  'lobsters','mussels','oysters','scallops','shrimp
                  'cod','flatfish','haddock','mackerel','perch','pi
                  'sea_bass','shark','swordfish','trout','walleye',
                  'high_protein_diet','low_gluten_diet','kidney_die
                  'low_fiber_diet','high_fiber_diet','muscle_diet',
                  'marijuana','needle_drugs','food_insecure','not_e
                  'bad_food_children','low_food_children','emergenc
                  'household_food_secure','wic_received','food_stam
                  'prescription_insurance','donate_blood','routine_
                  'hepatitis_a_vaccine','hepatitis_b_vaccine','neut
                  'overweight','gluten_free','losing_weight','excer
                  'reducing_salt','reducing_fat','work_done','month
                  'non_govt_employee','visit_dentist','floss','use_
                  'vigorous_work','moderate_work','walk_or_bike','v
                  'tv_hours','pc_hours','bp_status','insecticide_us
                  'sleep_hours','smokers_in_house','tobaco_1w','smo
                  'someone_smoked_at_job','smoked_at_restaurant','n
                  'someone_smoked_in_car','smoked_another_home','so
                  'someone_smoked_other_building','weight_1y','weig
                  'overweight_self','weightloss_desire','weightloss
```

In [19]:
```
cols = ['age','BMI','calories','family_income']

fig, ax = plt.subplots(2, 2)
for i, col in enumerate(cols):
    ax[i//2, i%2].hist(df[col])
    ax[i//2, i%2].set_title(col)
    ax[i//2, i%2].set_ylabel("Num of Patients")
plt.tight_layout()
```



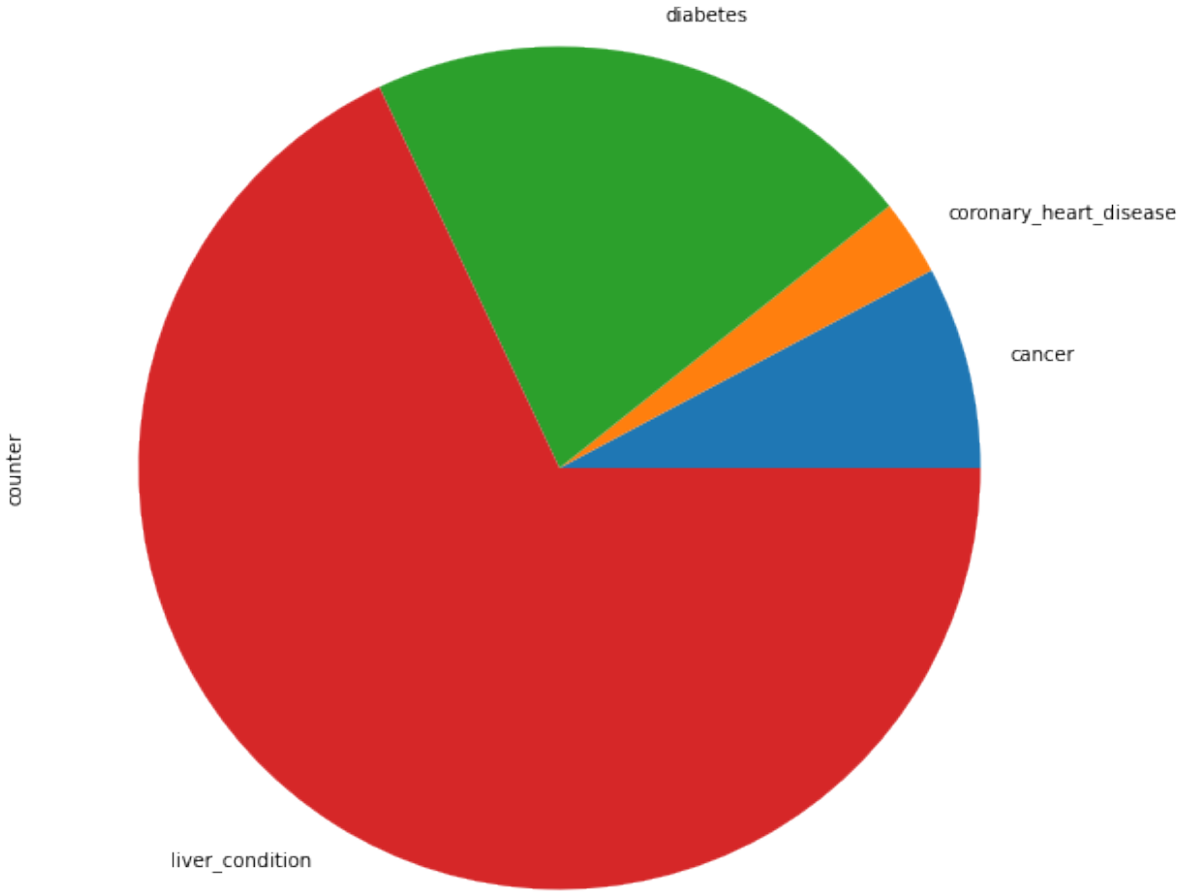
In [20]:
```
targets = ['cancer','diabetes','coronary_heart_disease','liver_cond
```

In [21]:
```
plot = df[targets].mean().sort_values().plot.barh(figsize=(8,8))
plot.set_xlabel("Prevelance")
plot.set_title("Disease Prevelance for all groups")
plt.show()
```



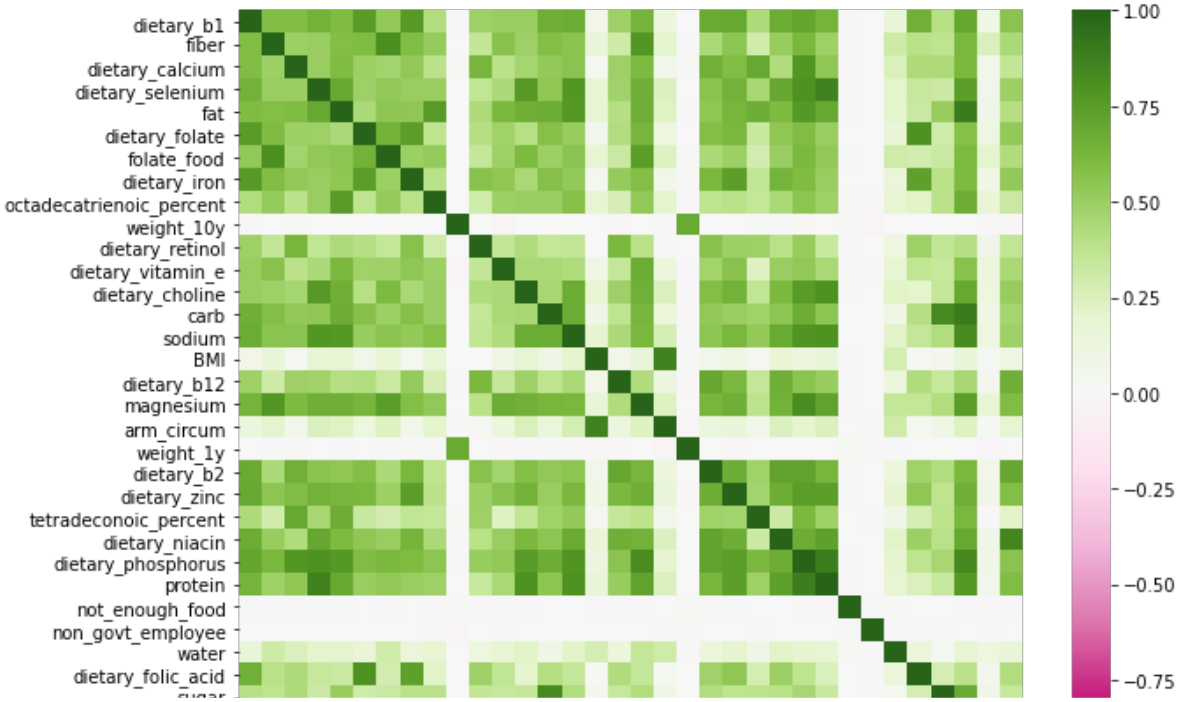
In [22]:
```
target_df = df[targets]
target_df = target_df[target_df.mean(axis=1) == 1/4]
target_df['disease'] = target_df.idxmax(1)
target_df['counter'] = 1
plt.figure(figsize = (10,10))
target_df.groupby('disease').counter.count().plot(kind='pie')
```

Out[22]: <AxesSubplot:ylabel='counter'>



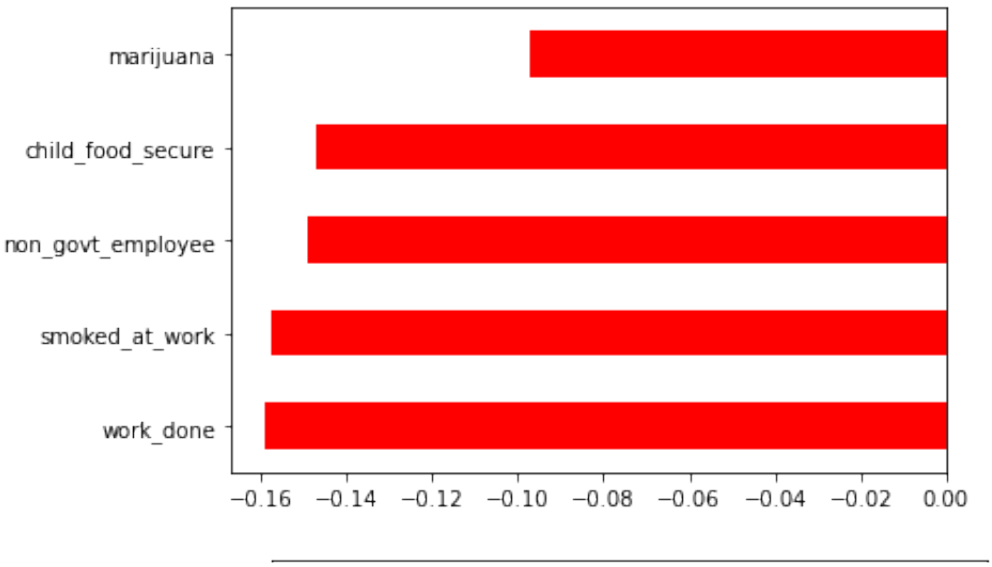
Finding Correlations with Disease

In [23]:
```
for df in target_dfs:
    ex_df = df[list(set(df.columns) & set(lifestyle_cols))]
    corr = ex_df.corr()
    corr = corr.mask(np.tril(np.ones(corr.shape)).astype(np.bool))
    redun = corr[abs(corr) >= 0.9].stack().reset_index()['level_1']
    ex_df = ex_df.drop(redun, axis=1)
    corr = ex_df.corr()
    corr = corr.mask(np.tril(np.ones(corr.shape)).astype(np.bool))
    big_corr = ex_df[corr[abs(corr).max() > 0.5].index].corr()
    big_corr = big_corr.mask(np.tril(np.ones(big_corr.shape)).astyp
    big_corr = ex_df[big_corr[abs(big_corr).max() > 0.5].index].cor

    plt.figure(figsize = (10,8))
    sns.heatmap(big_corr,
            xticklabels=big_corr.columns,
            yticklabels=big_corr.columns,
            cmap="PiYG",
            vmin=-1, vmax=1)
```



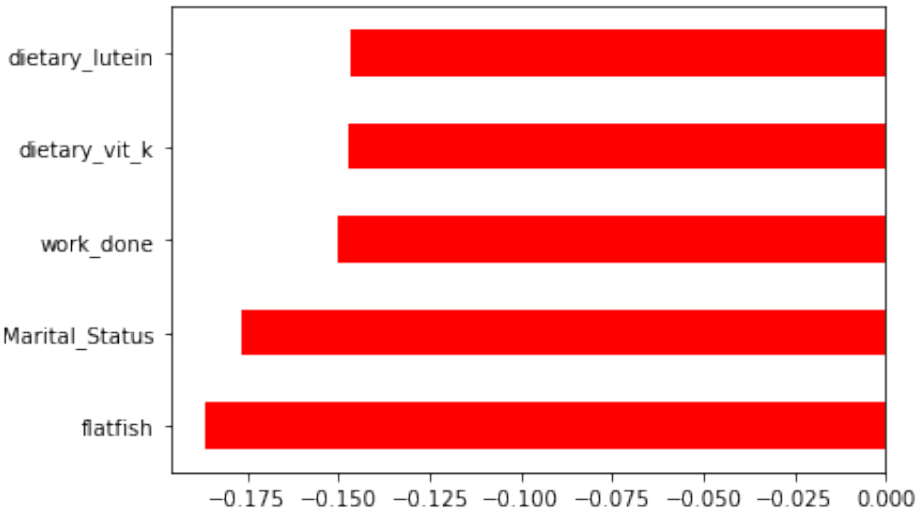
In [24]:
```
neg_factors = set()
pos_factors = set()
for col, df in zip(targets, target_dfs):
    ex_df = df[list(set(df.columns) & set(lifestyle_cols))]
    print('==========================')
    print("Correlations with ", col)
    print('==========================')
    corr = ex_df.corrwith(df[col])
    neg_factors.update(corr.sort_values().dropna().head(10).index)
    pos_factors.update(corr.sort_values().dropna().tail(10).index)
    temp_df = corr.sort_values().dropna().head(5)
    temp_df.plot(kind='barh',
                 color=(temp_df > 0).map({True: 'g',
                                          False: 'r'}))
    plt.show()
    temp_df = corr.sort_values().dropna().tail(5)
    temp_df.plot(kind='barh',
                 color=(temp_df > 0).map({True: 'g',
                                          False: 'r'}))
    plt.show()
```

```
==========================
Correlations with  cancer
==========================
```



In [25]:
```
temp_df = corr.sort_values().dropna().head(5)
temp_df.plot(kind='barh',
                     color=(temp_df > 0).map({True: 'g',
                                              False: 'r'}))
```

Out[25]: <AxesSubplot:>



In [26]:
```
temp_df > 0
```

Out[26]:
```
flatfish          False
Marital_Status    False
work_done         False
dietary_vit_k     False
dietary_lutein    False
dtype: bool
```

In [27]:
```
neg_factors
```

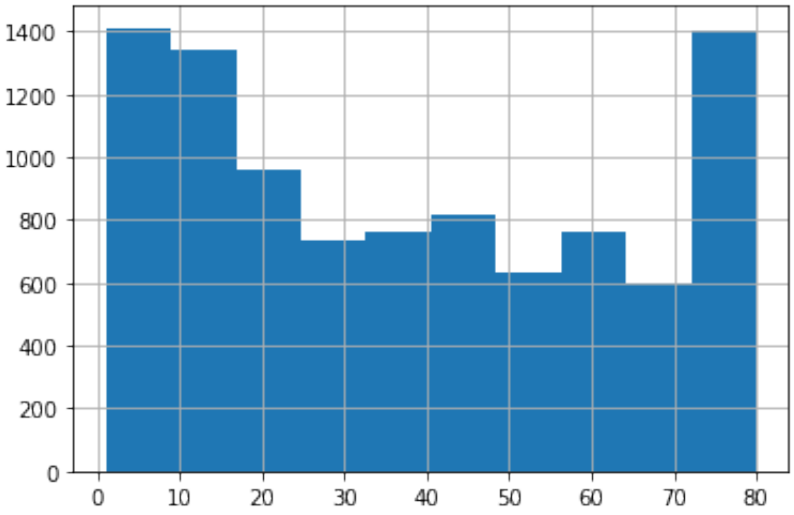
Out[27]:
```
{'Marital_Status',
 'alcohol_per_day',
 'child_food_secure',
 'children_in_house',
 'dietary_lutein',
 'dietary_vit_k',
 'dr_visits',
 'flatfish',
 'floride_water',
 'food_secure',
 'grocery_budget',
 'hepatitis_a_vaccine',
 'hepatitis_b_vaccine',
 'household_food_secure',
 'marijuana',
 'non_govt_employee',
 'smoked_another_home',
 'smoked_at_work',
 'use_sunscreen',
 'vigorous_recreation',
 'visit_dentist',
 'wic_received',
 'work_done'}
```

In [28]:
```
pos_factors
```

Out[28]:
```
{'age',
 'breaded_fish',
 'caffeine',
 'cheap_food_children',
 'emergency_food_received',
 'greatest_weight',
 'health_insurance',
 'job_duration',
 'losing_weight',
 'neutrophils_percent',
 'overweight',
 'overweight_self',
 'people_in_house',
 'prescription_insurance',
 'reducing_fat',
 'reducing_salt',
 'routine_healthcare',
 'smoked_1w',
 'smoked_other_building',
 'take_prednisone',
 'tobaco_1w',
 'tonsils_removed',
 'type_of_work',
 'vigorous_work',
 'weight_10y',
 'wisdom_teeth_removed'}
```

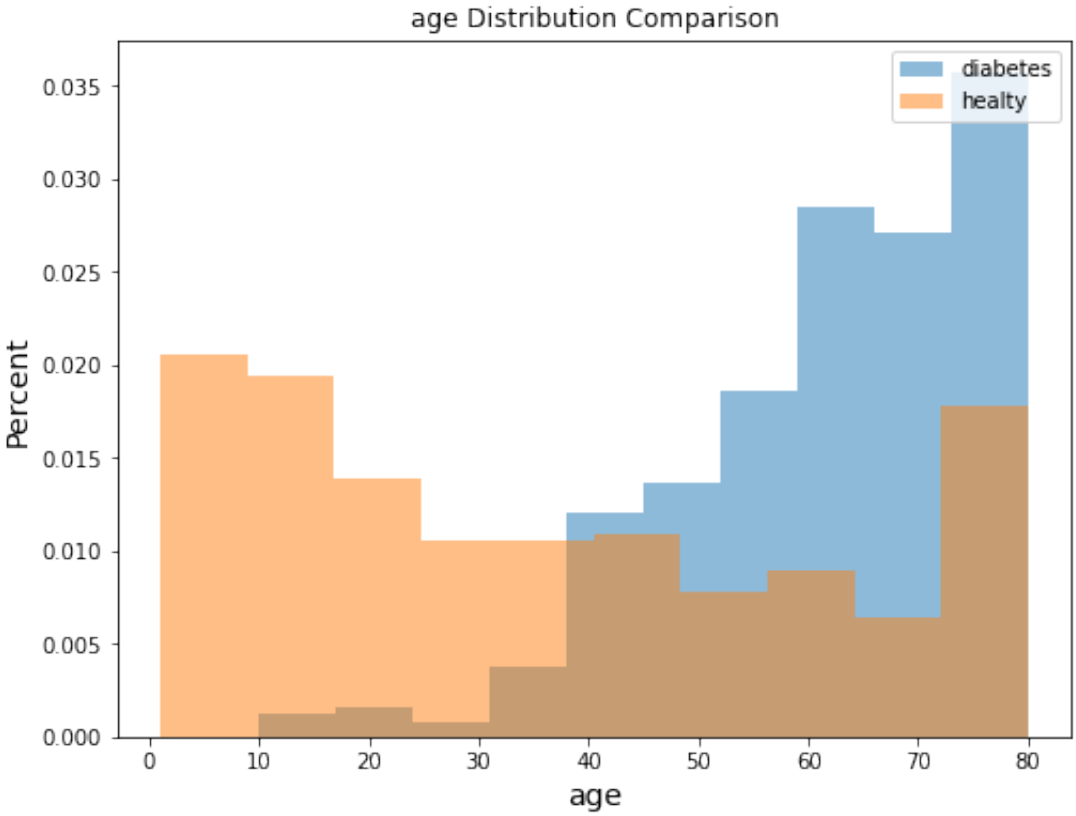
In [29]:
```
diabetes_df.age.hist()
```

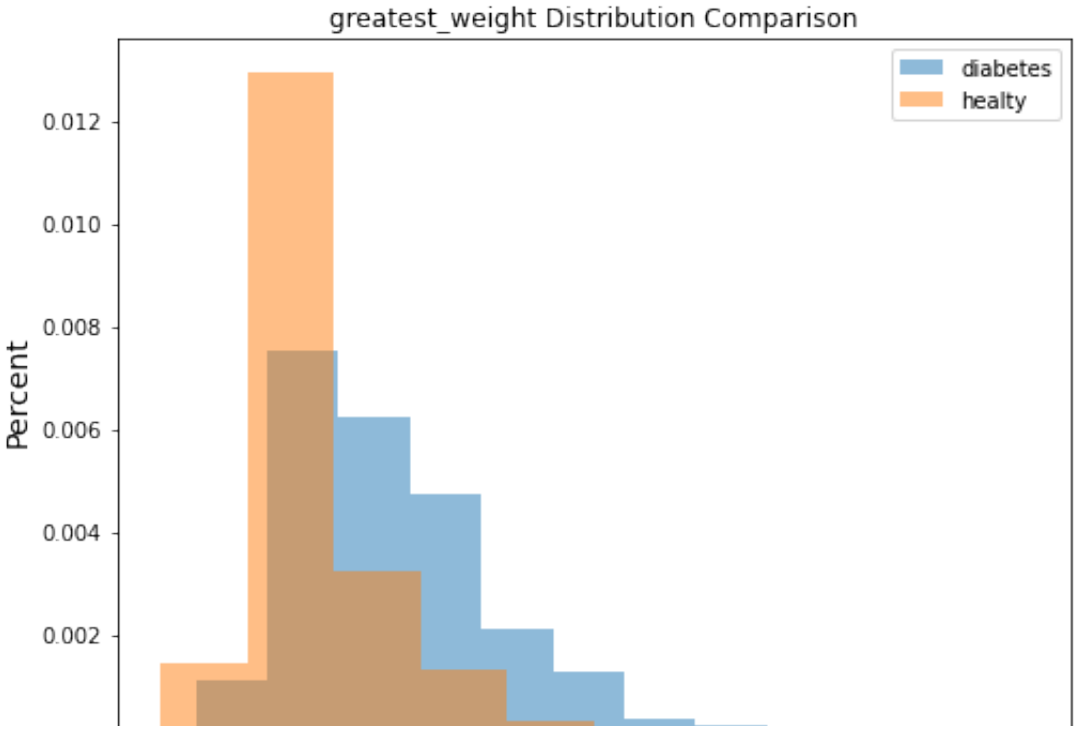
Out[29]:
```
<AxesSubplot:>
```

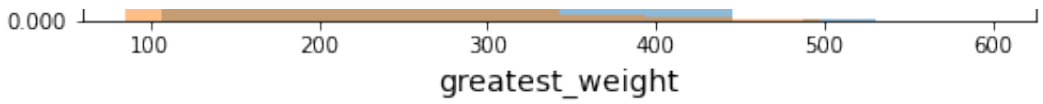


In [30]:

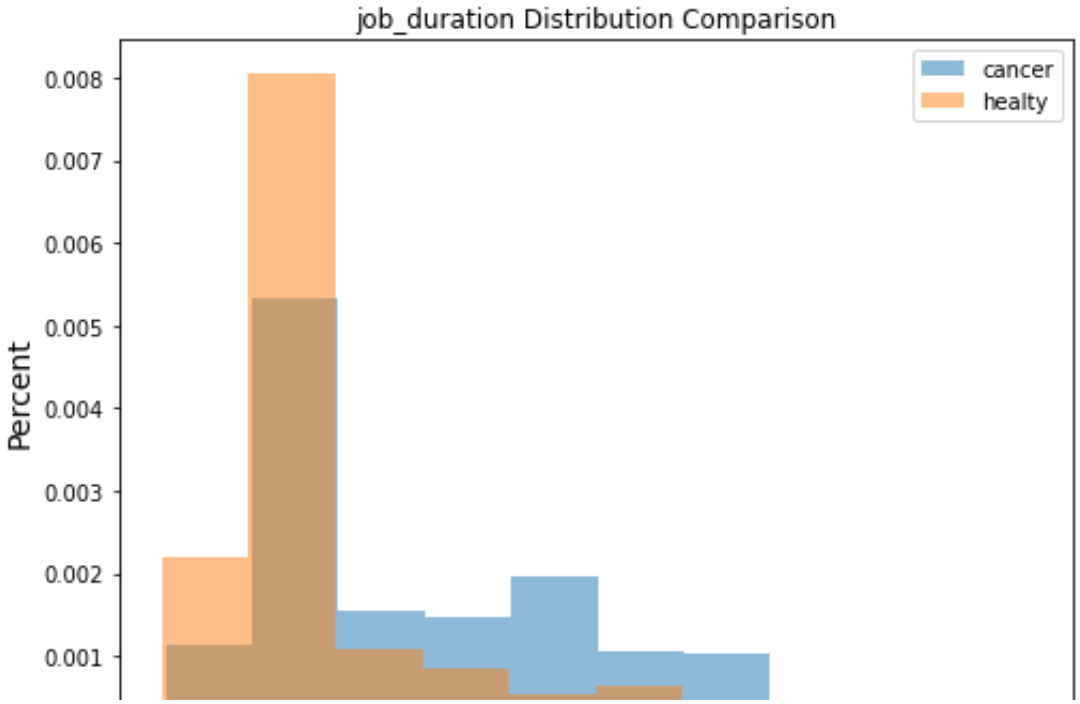
```
for col in ['age','greatest_weight']:#,'tetradeconoic_percent','BMI
    plt.figure(figsize=(8,6))
    plt.hist(diabetes_df[diabetes_df.diabetes == 1][col], bins=10,
    plt.hist(diabetes_df[diabetes_df.diabetes == 0][col], bins=10,
    plt.xlabel(col, size=14)
    plt.ylabel("Percent", size=14)
    plt.title(col + " Distribution Comparison")
    plt.legend(loc='upper right')
    plt.show()
```







In [31]:
```
for col in ['job_duration','age','child_food_secure','smoked_at_wor
    plt.figure(figsize=(8,6))
    plt.hist(cancer_df[cancer_df.cancer == 1][col], bins=10, alpha=
    plt.hist(cancer_df[cancer_df.cancer == 0][col], bins=10, alpha=
    plt.xlabel(col, size=14)
    plt.ylabel("Percent", size=14)
    plt.title(col + " Distribution Comparison")
    plt.legend(loc='upper right')
    plt.show()
```



In [32]:
```
neg_factors
```

Out[32]:
```
{'Marital_Status',
 'alcohol_per_day',
 'child_food_secure',
 'children_in_house',
 'dietary_lutein',
 'dietary_vit_k',
 'dr_visits',
 'flatfish',
 'floride_water',
 'food_secure',
 'grocery_budget',
 'hepatitis_a_vaccine',
 'hepatitis_b_vaccine',
 'household_food_secure',
 'marijuana',
 'non_govt_employee',
 'smoked_another_home',
 'smoked_at_work',
 'use_sunscreen',
 'vigorous_recreation',
 'visit_dentist',
 'wic_received',
 'work_done'}
```

In [33]:
```
pos_factors
```

Out[33]:
```
{'age',
 'breaded_fish',
 'caffeine',
 'cheap_food_children',
 'emergency_food_received',
 'greatest_weight',
 'health_insurance',
 'job_duration',
 'losing_weight',
 'neutrophils_percent',
 'overweight',
 'overweight_self',
 'people_in_house',
 'prescription_insurance',
 'reducing_fat',
 'reducing_salt',
 'routine_healthcare',
 'smoked_1w',
 'smoked_other_building',
 'take_prednisone',
 'tobaco_1w',
 'tonsils_removed',
 'type_of_work',
 'vigorous_work',
 'weight_10y',
 'wisdom_teeth_removed'}
```

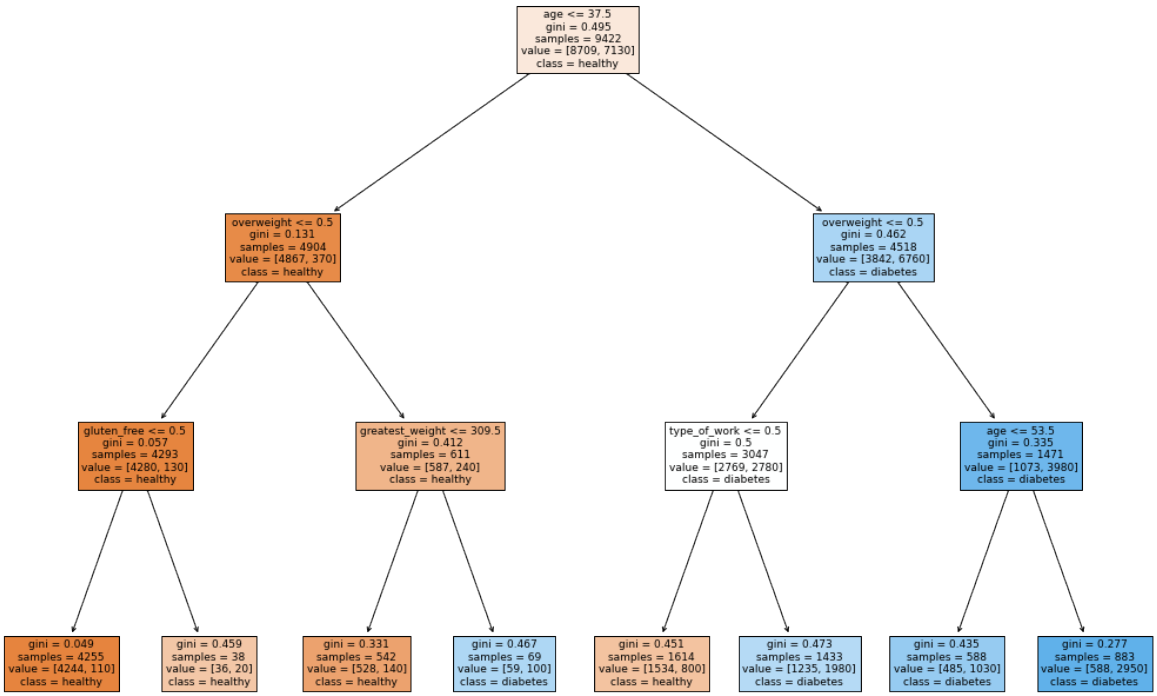
Decision Trees

In [34]:
```
def tree_diag(target, df, depth=5, ratio=3):
    ex_df = df[list(set(df.columns) & set(lifestyle_cols))]
    tree_mod = DecisionTreeClassifier(max_depth=depth, class_weight
    tree_mod.fit(ex_df, df[target])
    fig = plt.figure(figsize=(18,13))
    _ = tree.plot_tree(tree_mod,
                       feature_names = ex_df.columns,
                       class_names = ['healthy',target],
                       filled=True)
def tree_f1(target, df, ratio=3):
    ex_df = df[list(set(df.columns) & set(lifestyle_cols))]
    X_train, X_test, y_train, y_test = train_test_split(ex_df, df[[
    tree_mod = DecisionTreeClassifier(class_weight={0:1,1:ratio}, c
    tree_mod.fit(X_train, y_train)
    return(f1_score(tree_mod.predict(X_test),y_test))
```

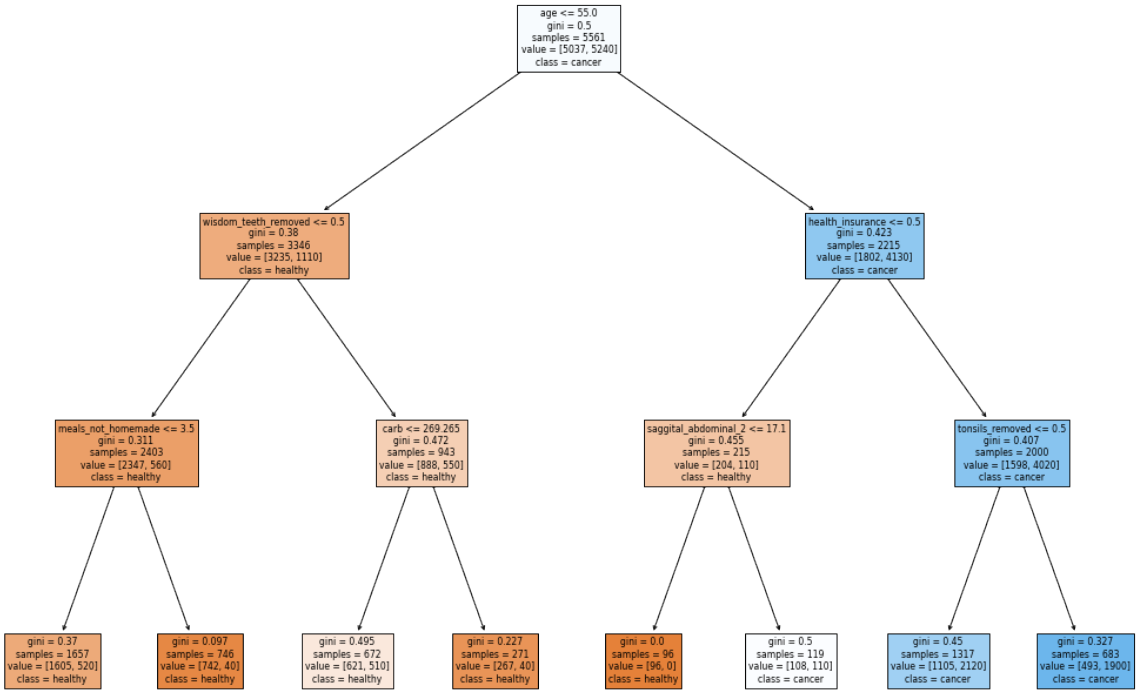
In [35]:
```
print('Diabetes F1 score: ', tree_f1('diabetes', diabetes_df, ratio
print('Cancer F1 score: ', tree_f1('cancer', cancer_df, ratio=10))
print('Heart Disease F1 score: ', tree_f1('coronary_heart_disease',
print('Liver Disease F1 score: ', tree_f1('liver_condition', liver_
```

```
Diabetes F1 score:  0.24742268041237112
Cancer F1 score:  0.1596244131455399
Heart Disease F1 score:  0.12121212121212122
Liver Disease F1 score:  0.48148148148148145
```

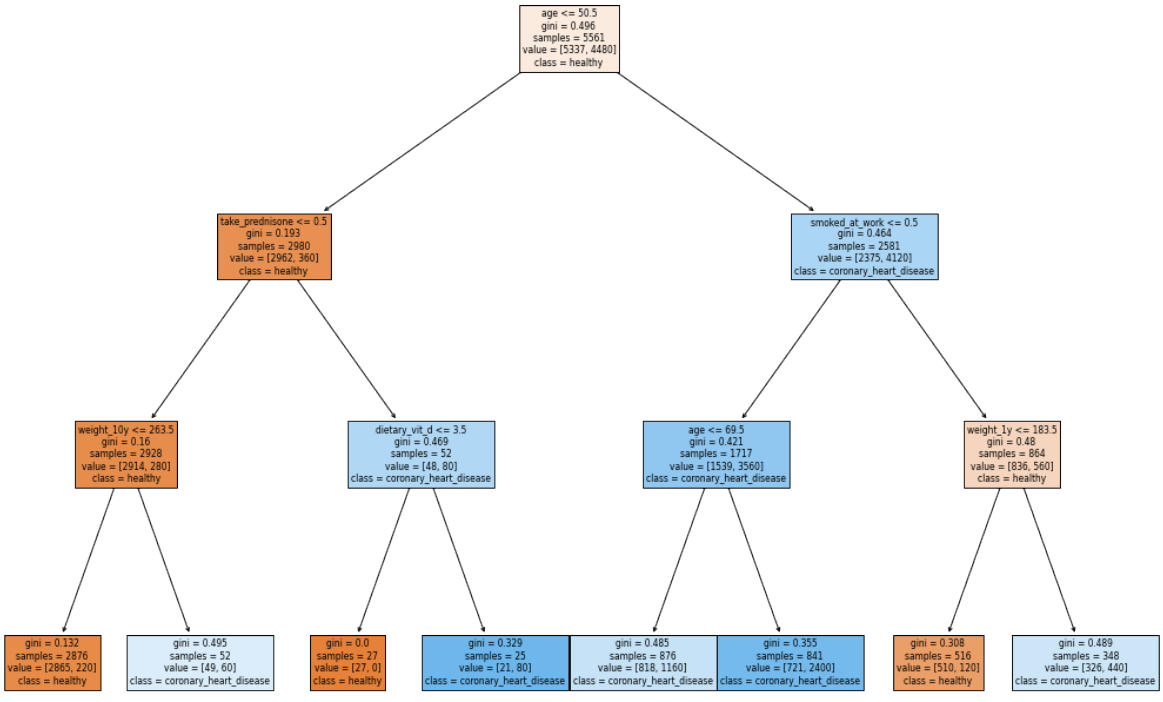
In [36]:
```
tree_diag('diabetes', diabetes_df, depth=3, ratio=10)
```



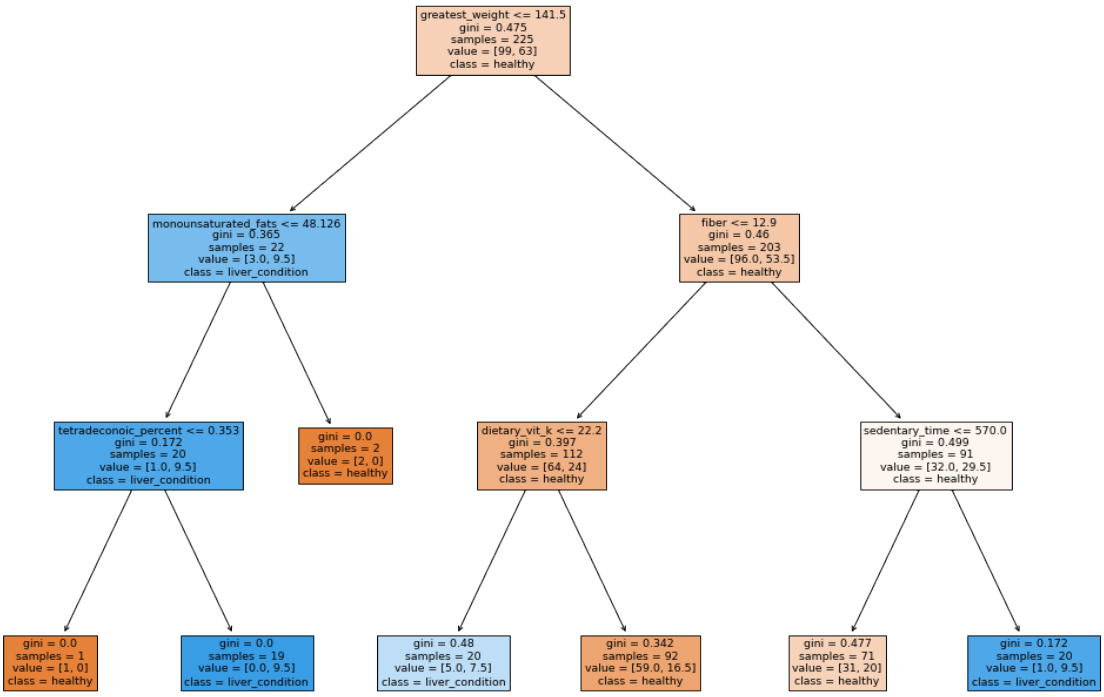
In [37]:
```
tree_diag('cancer', cancer_df, depth=3, ratio=10)
```



In [38]:
```
tree_diag('coronary_heart_disease', heart_df, 3, ratio=20)
```



In [39]:
```
tree_diag('liver_condition', liver_df, 3, ratio=0.5)
```



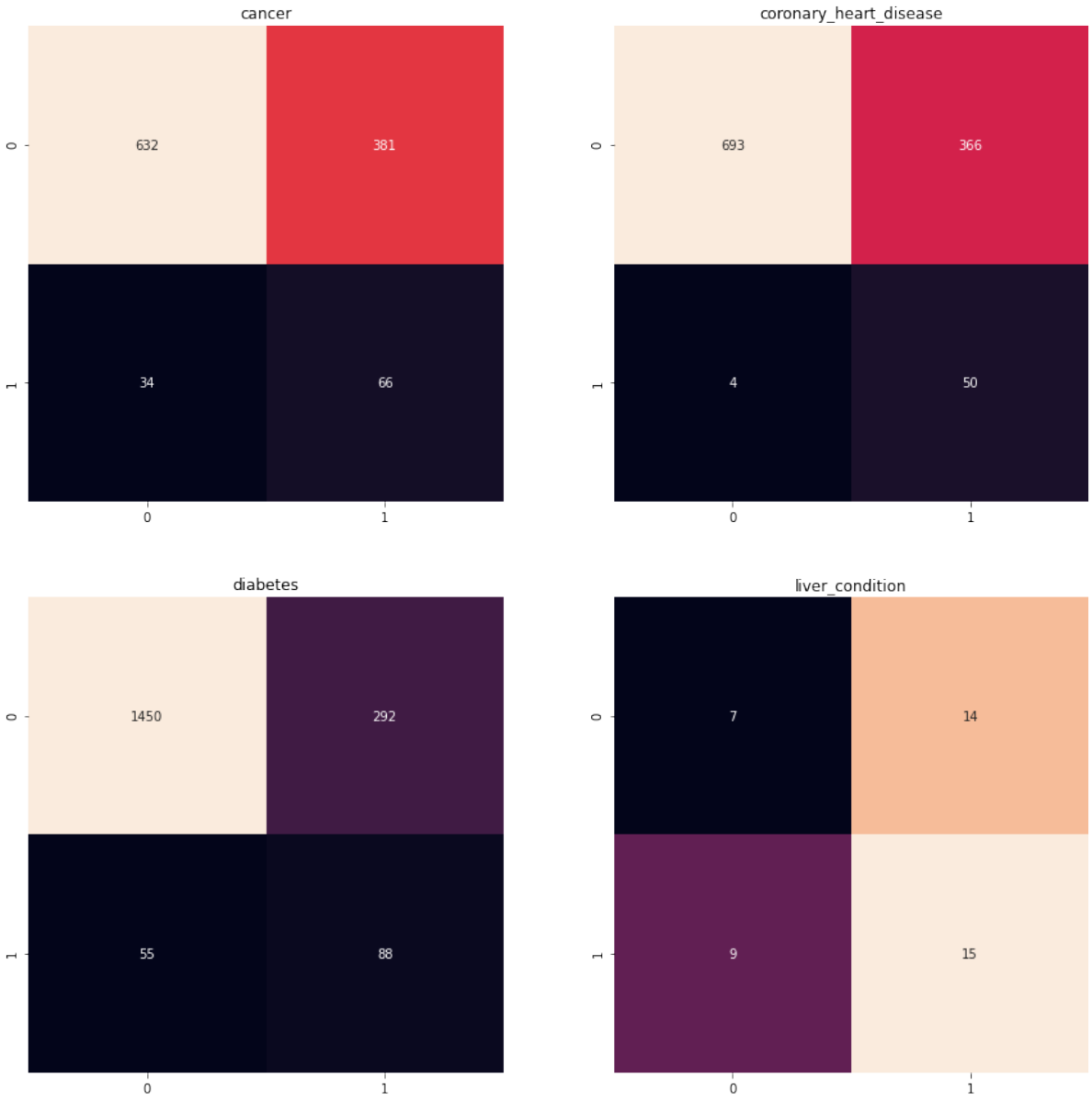
Support function

In [40]:
```
def nhanes_pred(model, target, df, proba=False):
    ex_df = df[list(set(df.columns) & set(lifestyle_cols))]
    X = ex_df
    y = df[[target]]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_
    model.fit(X_train, y_train.values.ravel())
    if proba:
        preds = model.predict_proba(X_test)[:,1]
    else:
        preds = model.predict(X_test)
    return y_test, preds

def plot_roc(y_test, probs):
    fpr, tpr, _ = roc_curve(y_test, probs)
    auc_score = auc(fpr, tpr)

    plt.plot(fpr, tpr, label='AUC = {:.2f}'.format(auc_score))
    plt.plot([0,1],[0,1],'r--')

    plt.xlim([-0.1,1.1])
    plt.ylim([-0.1,1.1])
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')

    plt.legend(loc='lower right')
    plt.show()
```

Naive Bayes

In [41]:
```
fig, ax = plt.subplots(2,2, figsize=(15,15))
for (i, target), df in zip(enumerate(targets), target_dfs):
    nb = GaussianNB()#var_smoothing=0.0000001)
    y_test, preds = nhanes_pred(nb, target, df)
    print(target, ' F1 score: ', f1_score(y_test,preds))
    mat = confusion_matrix(y_test, preds)
    sns.heatmap(mat, ax=ax[i%2, i//2], square=True, annot=True, cba

plt.show()
```

```
cancer  F1 score:  0.2413162705667276
diabetes  F1 score:  0.3365200764818356
coronary_heart_disease  F1 score:  0.2127659574468085
liver_condition  F1 score:  0.5660377358490567
```



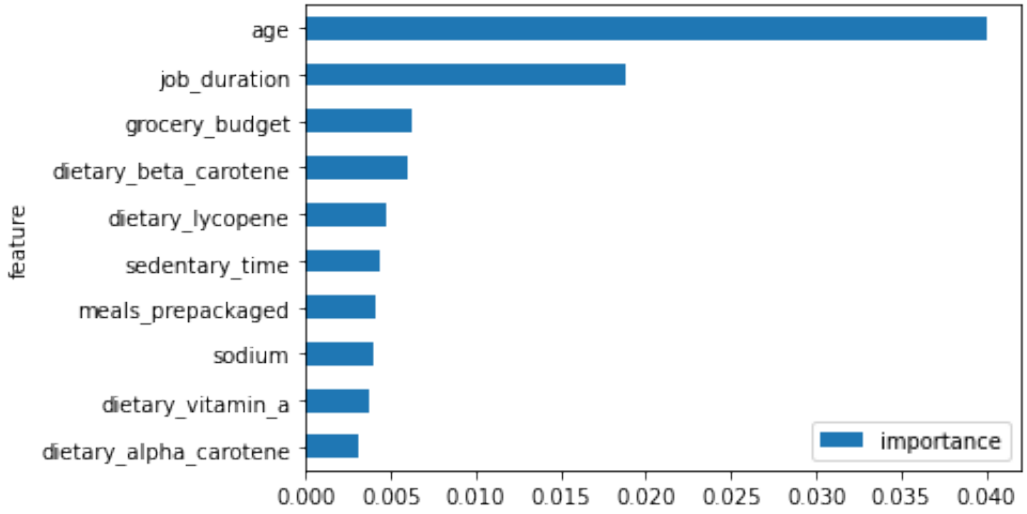
In [42]:
```
nb = GaussianNB(var_smoothing=0.0000001)
ex_df = cancer_df[list(set(cancer_df.columns) & set(lifestyle_cols)
X = ex_df
y = cancer_df[['cancer']]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
nb.fit(X_train, y_train.values.ravel())
```

Out[42]:
```
GaussianNB(var_smoothing=1e-07)
```

In [43]:
```
imps = permutation_importance(nb, X_test, y_test)
importances = imps.importances_mean
```

In [44]:
```
imp_df = pd.DataFrame(ex_df.columns, columns=['feature'])
imp_df['importance'] = importances
imp_df.set_index('feature').sort_values(by='importance').tail(10).p
```

Out[44]:
```
<AxesSubplot:ylabel='feature'>
```



SVM

In [45]:
```
space = {
        'C': hp.quniform('C', 0.005,1.0,0.01),
        'kernel': hp.choice('kernel', ['poly', 'sigmoid', 'rbf']),
        'degree': hp.choice('degree', [1,2,3,4])
        }
```

In [46]:
```
ex_df = heart_df[list(set(heart_df.columns) & set(lifestyle_cols))]
X_train, X_test, y_train, y_test = train_test_split(ex_df, heart_df
```

In [47]:
```
def svm_score(params):
    mod = svm.SVC(**params)
    mod.fit(X_train, y_train.values.ravel())
    predictions = mod.predict(X_test)
    return 1 - f1_score(y_test, predictions)
```

In [48]:
```
def get_params(score):
    best = fmin(score, space, algo=tpe.suggest, max_evals=100)
    return best
```

In [49]:
```
get_params(svm_score)
```

```
100%|██████████| 100/100 [01:23<00:00,  1.19trial/s, best loss: 1.
0]
```

Out[49]:
```
{'C': 0.07, 'degree': 1, 'kernel': 0}
```

In [50]:
```
best_params = {'C': 0.17, 'degree': 2, 'kernel': 'sigmoid'}
```

In [51]:
```
for target, df in zip(targets, target_dfs):
    mod = svm.SVC(**best_params)
    ex_df = df[list(set(df.columns) & set(lifestyle_cols))]
    X_train, X_test, y_train, y_test = train_test_split(ex_df, df[[
    mod.fit(X_train, y_train.values.ravel())
    predictions = mod.predict(X_test)
    print('F1 score for ', target, ': ', f1_score(y_test, predictio
```

```
F1 score for  cancer :  0.06134969325153374
F1 score for  diabetes :  0.00829875518672199
F1 score for  coronary_heart_disease :  0.0
F1 score for  liver_condition :  0.732394366197183
```

XGBoost

Results are poor when using default parameters.

Below is a systematic search of hyperparamters

In [52]:
```
#Dataset is imbalanced, so we will pass a balancing parameter to XG
heart_df.coronary_heart_disease.value_counts()[0]/heart_df.coronary
```

Out[52]:
```
23.825892857142858
```

In [53]:
```
targets
```

Out[53]:
```
['cancer', 'diabetes', 'coronary_heart_disease', 'liver_condition'
]
```

In [54]:
```
space = {
        'eta': hp.quniform('eta', 0.025, 0.5, 0.025),
        'max_depth':  hp.choice('max_depth', np.arange(1, 14, dtype
        'min_child_weight': hp.quniform('min_child_weight', 1, 8, 1
        'subsample': hp.quniform('subsample', 0.5, 1, 0.05),
        'gamma': hp.quniform('gamma', 0.5, 1, 0.05),
        'colsample_bytree': hp.quniform('colsample_bytree', 0.5, 1,
        'scale_pos_weight': hp.choice('scale_pos_weight', np.arange
        'eval_metric': 'auc'
    }
```

In [55]:
```
def xg_score(params):
    mod = xgb.train(params, dtrain,
                    early_stopping_rounds=100,
                    evals=[(dvalid,'valid'), (dtrain,'train')],
                    verbose_eval=False)
    predictions = mod.predict(dvalid)
    return 1 - roc_auc_score(y_test, predictions)
```

In [56]:
```
ex_df = heart_df[list(set(heart_df.columns) & set(lifestyle_cols))]
X_train, X_test, y_train, y_test = train_test_split(ex_df, heart_df
dtrain = xgb.DMatrix(data=X_train.values,
                     feature_names=X_train.columns,
                     label=y_train.values)
dvalid = xgb.DMatrix(data=X_test.values,
                     feature_names=X_test.columns,
                     label=y_test.values)
```

In [57]:
```
def get_params(score):
    best = fmin(score, space, algo=tpe.suggest, max_evals=250)
    return best
```

In [58]:
```
get_params(xg_score)
```

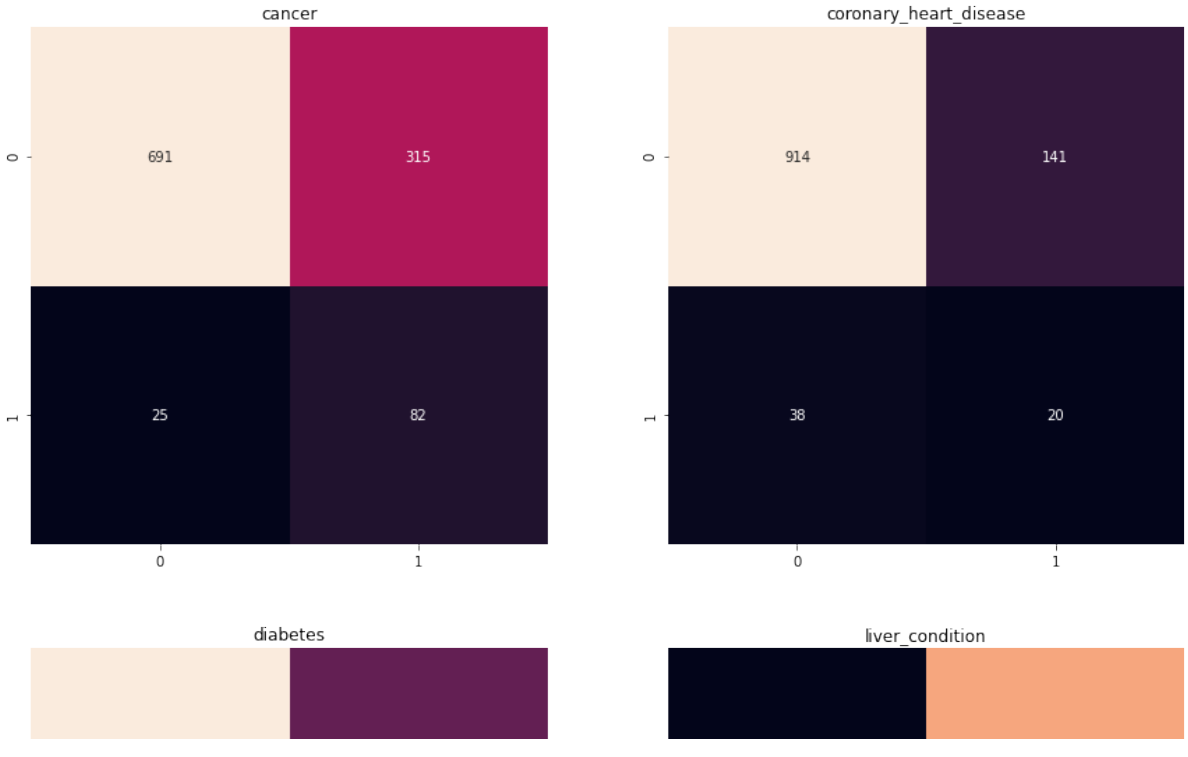
```
100%|██████████| 250/250 [01:21<00:00,  3.08trial/s, best loss: 0.
13107465208305547]
```

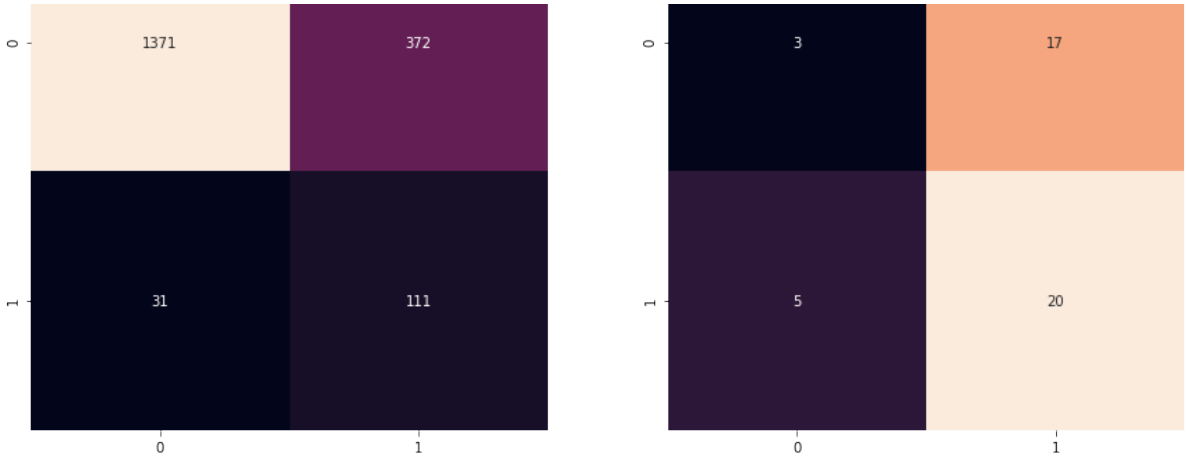
Out[58]:
```
{'colsample_bytree': 0.8,
 'eta': 0.47500000000000003,
 'gamma': 1.0,
 'max_depth': 0,
 'min_child_weight': 5.0,
 'scale_pos_weight': 1,
 'subsample': 0.6000000000000001}
```

In [59]:
```
#Note: hpchoice actually returns the index, so the value for max_de
best_params = {'colsample_bytree': 0.7,
               'eta': 0.45,
               'gamma': 0.55,
               'max_depth': 1,
               'min_child_weight': 1.0,
               'subsample': 0.85,
               'scale_pos_weight': 13,
               'eval_metric': 'auc'}
```

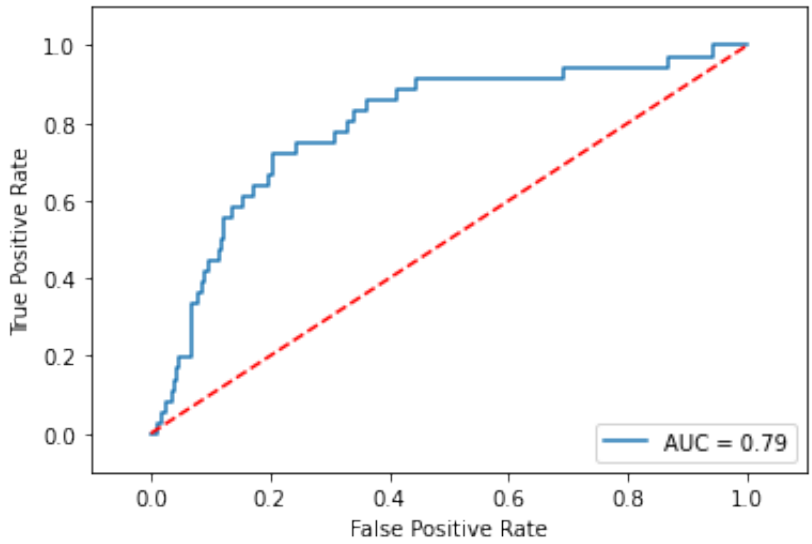
In [60]:
```
fig, ax = plt.subplots(2,2, figsize=(15,15))
for (i, target),df in zip(enumerate(targets), target_dfs):

    xgc = XGBClassifier(**best_params)
    y_test, preds = nhanes_pred(xgc, target, df)
    mat = confusion_matrix(y_test, preds)
    sns.heatmap(mat, ax=ax[i%2, i//2], square=True, annot=True, cba
    print(target, ' F1 score: ', f1_score(y_test,preds))
plt.show()
```

```
/opt/conda/lib/python3.7/site-packages/xgboost/sklearn.py:1146: Us
erWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, d
o the following: 1) Pass option use_label_encoder=False when const
ructing XGBClassifier object; and 2) Encode your labels (y) as int
egers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

cancer  F1 score:  0.32539682539682535
diabetes  F1 score:  0.35519999999999996
coronary_heart_disease  F1 score:  0.182648401826484
liver_condition  F1 score:  0.6451612903225806
```

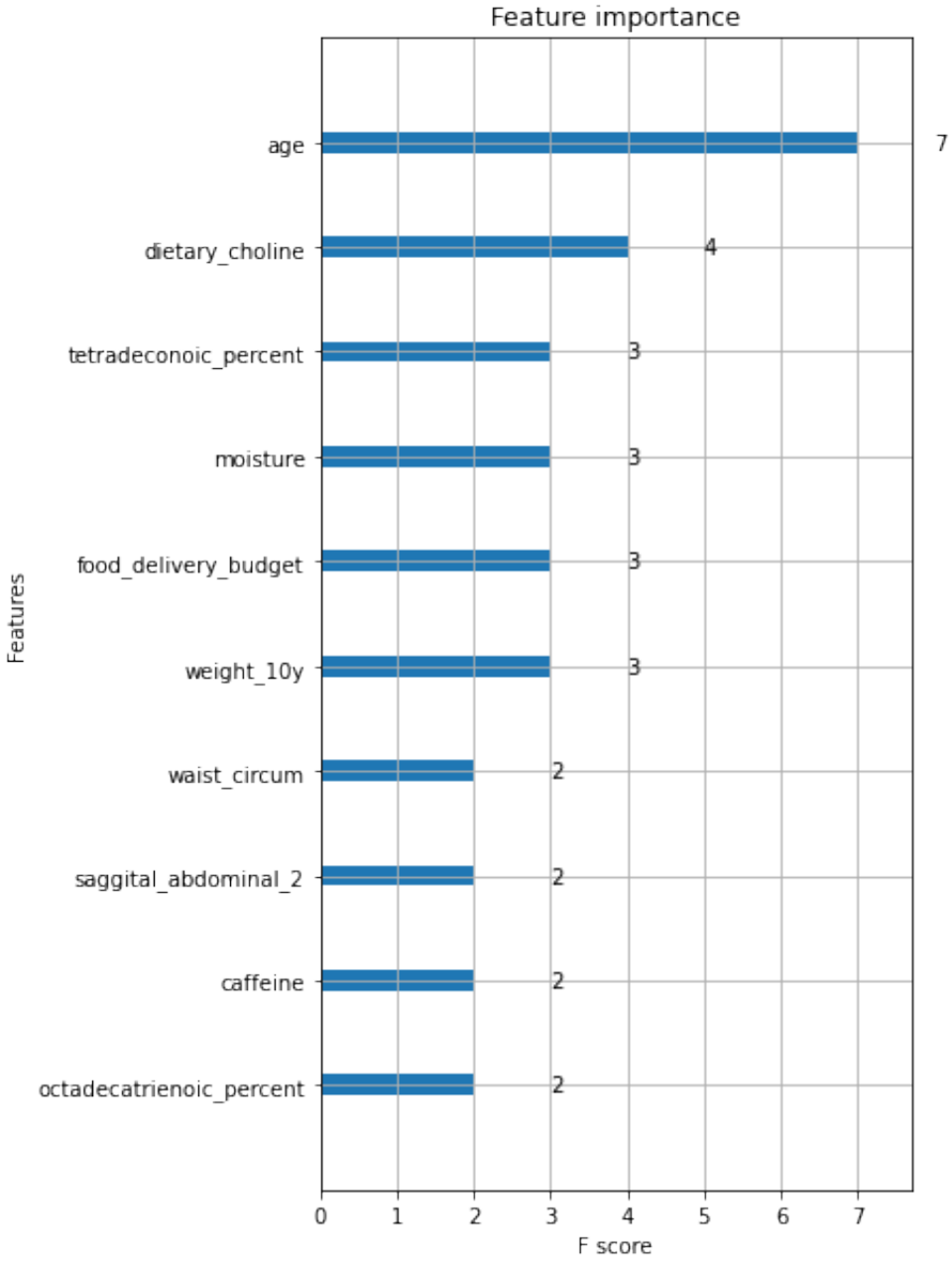




In [61]:
```
xgc = XGBClassifier(**best_params)
y_test, probs = nhanes_pred(xgc, 'coronary_heart_disease', heart_df
plot_roc(y_test, probs)
```



In [62]:
```
ax = plot_importance(xgc, max_num_features=10)
fig = ax.figure
fig.set_size_inches(5, 10)
```



In [ ]:

In [ ]:

In [ ]: