In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
from sklearn.model_selection import *
from sklearn.multioutput import *
from sklearn.linear_model import *
from sklearn.metrics import *
from sklearn.preprocessing import *
```

In [2]:
```
train_input = pd.read_csv("training_set_features.csv")
train_output = pd.read_csv("training_set_labels.csv")
test_input = pd.read_csv("test_set_features.csv")
test_output = pd.read_csv("submission_format.csv")
```

In [3]:
```
print("Percentage of Null Values\n")
for i in train_input:
    p = train_input[i].isna().sum()*100/len(train_input[i])
    print(i,":",round(p,2))
```

```
Percentage of Null Values

respondent_id : 0.0
h1n1_concern : 0.34
h1n1_knowledge : 0.43
behavioral_antiviral_meds : 0.27
behavioral_avoidance : 0.78
behavioral_face_mask : 0.07
behavioral_wash_hands : 0.16
behavioral_large_gatherings : 0.33
behavioral_outside_home : 0.31
behavioral_touch_face : 0.48
doctor_recc_h1n1 : 8.09
doctor_recc_seasonal : 8.09
chronic_med_condition : 3.64
child_under_6_months : 3.07
health_worker : 3.01
health_insurance : 45.96
opinion_h1n1_vacc_effective : 1.46
opinion_h1n1_risk : 1.45
opinion_h1n1_sick_from_vacc : 1.48
opinion_seas_vacc_effective : 1.73
opinion_seas_risk : 1.92
opinion_seas_sick_from_vacc : 2.01
age_group : 0.0
education : 5.27
race : 0.0
sex : 0.0
income_poverty : 16.56
marital_status : 5.27
rent_or_own : 7.65
employment_status : 5.48
hhs_geo_region : 0.0
census_msa : 0.0
household_adults : 0.93
household_children : 0.93
employment_industry : 49.91
employment_occupation : 50.44
```

In [4]:
```
replace_with_one_value_one_condition = np.array(["behavioral_avoidance","beh

replace_with_one_value_two_conditions = np.array(["behavioral_antiviral_meds

replace_with_one_value_three_conditions = ["behavioral_wash_hands"] #{"h1n1_
#-------------------------------------------------------------------------
replace_with_value_0 = ["doctor_recc_h1n1","doctor_recc_seasonal","health_wo

replace_with_value_1 = ["h1n1_knowledge","household_adults","income_poverty"

replace_with_value_2 = ["h1n1_concern"]

replace_with_value_3 = ["opinion_h1n1_vacc_effective","opinion_h1n1_sick_fro

replace_with_median = ["chronic_med_condition","education"]
#-------------------------------------------------------------------------
sort_and_numerical_encoding = ["age_group","race","sex","employment_industry
```

```
#--------------------------------------------------------------------------
values_replace_dict = {
    "education" : {"< 12 Years":0,"12 Years":1,"College Graduate":2,"Some Co
    "income_poverty" : {"Below Poverty":0,"<= $75,000, Above Poverty":1,"> $
    "marital_status" : {"Not Married":0,"Married":1},
    "rent_or_own" : {"Rent":0,"Own":1},
    "employment_status" : {"Unemployed":0,"Not in Labor Force":1,"Employed":
    "census_msa" : {"Non-MSA":0,"MSA, Not Principle City":1,"MSA, Principle
}
```

In [5]:

```
def sort_using_numerical_encoding(df,sort_and_numerical_encoding):
    for i in sort_and_numerical_encoding:
        req_val = {}
        ctr = 0
        vals = list(df[i].unique())
        while np.nan in vals:
            vals.remove(np.nan)
        vals = np.sort(np.array(vals))
        for j in vals:
            req_val[j] = ctr
            ctr += 1
        df[i] = df[i].map(req_val)
    return df

def replace_na_with_values(df,replace_with_value_0,replace_with_value_1,repl
    for i in replace_with_value_0:
        df[i] = df[i].fillna(0.0)
    for i in replace_with_value_1:
        df[i] = df[i].fillna(1.0)
    for i in replace_with_value_2:
        df[i] = df[i].fillna(2.0)
    for i in replace_with_value_3:
        df[i] = df[i].fillna(3.0)
    return df

def replace_na_with_conditional_values(df,replace_with_one_value_one_conditi

    for i in replace_with_one_value_one_condition:
        df[i][df["h1n1_concern"]==3.0] = df[i][df["h1n1_concern"]==3.0].fill
        df[i] = df[i].fillna(0.0)

    for i in replace_with_one_value_two_conditions:
        df[i][(df["h1n1_concern"]==3) & (df["h1n1_knowledge"]==2)] = df[i][(
        df[i] = df[i].fillna(0.0)

    for i in replace_with_one_value_three_conditions:
        df[i][(df["h1n1_concern"]==3) & (df["h1n1_knowledge"]==2)] = df[i][(
        df[i][(df["h1n1_concern"]==3) & (df["h1n1_knowledge"]==1)] = df[i][(
        df[i] = df[i].fillna(0.0)

    return df

def replace_with_dict_values(df,values_replace_dict):
    ctr = 0
    for i in values_replace_dict:
        df[i] = df[i].map(values_replace_dict[i])
```

```
            ctr += 1
        return df

def replace_null_with_relevant_values(df):
    df["chronic_med_condition"] = df["chronic_med_condition"].fillna(df["chr
    df["education"] = df["education"].fillna(df["education"].median())
    df["income_poverty"] = df["income_poverty"].fillna(df["income_poverty"].
    df["marital_status"] = df["marital_status"].fillna(0.0)
    df["rent_or_own"] = df["rent_or_own"].fillna(0.0)
    df["employment_status"] = df["employment_status"].fillna(df["employment_
    df["census_msa"] = df["census_msa"].fillna(1.0)
    df["employment_industry"] = df["employment_industry"].fillna(df_train_ir
    df["employment_occupation"] = df["employment_occupation"].fillna(df_trai
    return df
```

In [6]:
```
df_train_input = train_input.copy()

df_train_input = replace_na_with_values(df_train_input,replace_with_value_0,

df_train_input = replace_na_with_conditional_values(df_train_input,replace_w

df_train_input = replace_with_dict_values(df_train_input,values_replace_dict

df_train_input = sort_using_numerical_encoding(df_train_input,sort_and_numer

df_train_input = replace_null_with_relevant_values(df_train_input)

df_train_input = df_train_input.astype("int64")
```

In [7]:
```
df_test_input = test_input.copy()

df_test_input = replace_na_with_values(df_test_input,replace_with_value_0,re

df_test_input = replace_na_with_conditional_values(df_test_input,replace_wit

df_test_input = replace_with_dict_values(df_test_input,values_replace_dict)

df_test_input = sort_using_numerical_encoding(df_test_input,sort_and_numeric

df_test_input = replace_null_with_relevant_values(df_test_input)

df_test_input = df_test_input.astype("int64")
```

In [8]:
```
ss = MinMaxScaler()
X_train = pd.DataFrame((ss.fit_transform(X=df_train_input.drop(columns="resp
X_test = pd.DataFrame((ss.fit_transform(X=df_test_input.drop(columns="respor
y_train = train_output.drop(columns="respondent_id")
```

In [9]:
```
lgmodel = LogisticRegression(solver='lbfgs')
mlc = MultiOutputClassifier(estimator=lgmodel)
mlc.fit(X_train,y_train)

y_prob = mlc.predict_proba(X_test)
y_test_prob = pd.DataFrame(columns=["h1n1_vaccine","seasonal_vaccine"])
y_test_prob["h1n1_vaccine"] = y_prob[0][:,1]
```

```
y_test_prob["seasonal_vaccine"] = y_prob[1][:,1]

y = mlc.predict(X_test)
y_test = pd.DataFrame(columns=["h1n1_vaccine","seasonal_vaccine"])
y_test["h1n1_vaccine"] = y[:,0]
y_test["seasonal_vaccine"] = y[:,1]
```

In [10]:
```
roc_auc_score(y_test,y_test_prob)
```

Out[10]: 1.0

In [11]:
```
fin_sol = pd.DataFrame(columns=["respondent_id","h1n1_vaccine","seasonal_vac
fin_sol["respondent_id"] = df_test_input["respondent_id"].copy()
fin_sol["h1n1_vaccine"] = y_test_prob["h1n1_vaccine"].copy()
fin_sol["seasonal_vaccine"] = y_test_prob["seasonal_vaccine"].copy()
```

In [12]:
```
fin_sol.to_csv("submission.csv",index=False,header=True)
```

In [ ]: