Flu Shot Learning: Predict Seasonal Flu Vaccines

Main objective of the analysis that specifies whether your model will be focused on prediction or interpretation

The project is focus solely on prediction of flu vaccines.

Brief description of the data set and a summary of its attributes

The data for this competition comes from the National 2009 H1N1 Flu Survey (NHFS).

The National 2009 H1N1 Flu Survey (NHFS) was sponsored by the National Center for Immunization and Respiratory Diseases (NCIRD) and conducted jointly by NCIRD and the National Center for Health Statistics (NCHS), Centers for Disease Control and Prevention (CDC). The NHFS was a list-assisted random-digit-dialing telephone survey of households, designed to monitor influenza immunization coverage in the 2009-10 season.

The target population for the NHFS was all persons 6 months or older living in the United States at the time of the interview. Data from the NHFS were used to produce timely estimates of vaccination coverage rates for both the monovalent pH1N1 and trivalent seasonal influenza vaccines.

The NHFS was conducted between October 2009 and June 2010. It was one-time survey designed specifically to monitor vaccination during the 2009-2010 flu season in response to the 2009 H1N1 pandemic. The CDC has other ongoing programs for annual phone surveys that continue to monitor seasonal flu vaccination.

Problem description

Can you predict whether people got seasonal flu vaccines using information they shared about their backgrounds, opinions, and health behaviors?

In this challenge, we will take a look at vaccination, a key public health measure used to fight infectious diseases. Vaccines provide immunization for individuals, and enough immunization in a community can further reduce the spread of diseases through "herd immunity."

The features in this dataset

Your goal is to predict how likely individuals are to receive their seasonal flu vaccines.

Field	Description
seasonal_vaccine	Whether respondent received seasonal flu vaccine
respondent_id	a unique and random identifier
behavioral_antiviral_meds	Has taken antiviral medications. (binary)
behavioral_avoidance	Has avoided close contact with others with flu-like symptoms. (binary)
behavioral_face_mask	Has bought a face mask. (binary)
behavioral_wash_hands	Has frequently washed hands or used hand sanitizer. (binary)
behavioral_large_gatherings	Has reduced time at large gatherings. (binary)
behavioral_outside_home	Has reduced contact with people outside of own household. (binary)
behavioral_touch_face	Has avoided touching eyes, nose, or mouth. (binary)
doctor_recc_seasonal	Seasonal flu vaccine was recommended by doctor. (binary)
chronic_med_condition	Has any of the following chronic medical conditions: asthma or an other lung condition, diabetes, a heart condition, a kidney condition, sickle cell anemia or other anemia, a neurological or neuromuscular condition, a liver condition, or a weakened immune system caused by a chronic illness or by medicines taken for a chronic illness. (binary)
child_under_6_months	Has regular close contact with a child under the age of six months. (binary)
health_worker	Is a healthcare worker. (binary)
health_insurance	Has health insurance. (binary)
opinion_seas_vacc_effective	Respondent's opinion about seasonal flu vaccine effectiveness. 1 = Not at all effective; 2 = Not very effective; 3 = Don't know; 4 = Somewhat effective; 5 = Very effective.
opinion_seas_risk	Respondent's opinion about risk of getting sick with seasonal flu without vaccine. 1 = Very Low; 2 = Somewhat low; 3 = Don't know; 4 = Somewhat high; 5 = Very high.
opinion_seas_sick_from_vacc	Respondent's worry of getting sick from taking seasonal flu vaccine. 1 = Not at all worried; 2 = Not very worried; 3 = Don't know; 4 = Somewhat worried; 5 = Very worried.
age_group	Age group of respondent.
education	Self-reported education level.
race	Race of respondent.
sex	Sex of respondent.
income_poverty	Household annual income of respondent with respect to 2008 Census poverty thresholds.
marital_status	Marital status of respondent.
employment_status	Employment status of respondent.
hhs_geo_region	Respondent's residence using a 10-region geographic classification defined by the U.S. Dept. of Health and Human Services. Values are represented as short random character strings.
census_msa	Respondent's residence within metropolitan statistical areas (MSA) as defined by the U.S. Census.
household_adults	Number of other adults in household, top-coded to 3.
household_children	Number of children in household, top-coded to 3.
employment_industry	Type of industry respondent is employed in. Values are represented as short random character strings.
employment_occupation	Type of occupation of respondent. Values are represented as short random character strings.

Brief summary of data exploration and actions taken for data cleaning and feature engineering

Data Exploration includes data summary, statistics, relevant graphs to find any relationships within.

As for data cleaning, we will check for missing values and decide what imputation method. We also check for data duplicates and outliers. Finally perform binary encoding before model training.

Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.model_selection import cross_val_score, train_test_split, GridSearchC
V, RandomizedSearchCV
from sklearn.preprocessing import LabelEncoder, StandardScaler, MinMaxScaler, OneHo
tEncoder, PolynomialFeatures
from sklearn.metrics import confusion_matrix, classification_report, mean_absolute_
error, mean_squared_error,r2_score
from sklearn.metrics import plot_confusion_matrix, plot_precision_recall_curve, plo
t_roc_curve, accuracy_score
from sklearn.metrics import auc, f1_score, precision_score, recall_score, roc_auc_s
core


%matplotlib inline
sns.set_style('dark')
sns.set(font_scale=1.2)

import warnings
warnings.filterwarnings('ignore')
import pandas.util.testing as tm
from pycaret.classification import *

np.random.seed(123)

pd.options.display.max_columns= None
#pd.options.display.max_rows = None
```

```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tools\_testing.py:19: Fut
ureWarning: pandas.util.testing is deprecated. Use the functions in the public A
PI at pandas.testing instead.
  import pandas.util.testing as tm
```

In [2]:
```
df = pd.read_csv("training_set_features.csv")
```

In [3]:
```
df
```

Out[3]:

	respondent_id	behavioral_antiviral_meds	behavioral_avoidance	behavioral_face_mask	behavioral_wash_
0	0	0.0	0.0	0.0	
1	1	0.0	1.0	0.0	
2	2	0.0	1.0	0.0	
3	3	0.0	1.0	0.0	
4	4	0.0	1.0	0.0	
...	...	...	...	...	
26702	26702	0.0	1.0	0.0	
26703	26703	0.0	1.0	0.0	
26704	26704	0.0	1.0	1.0	
26705	26705	0.0	0.0	0.0	
26706	26706	0.0	1.0	0.0	

26707 rows × 31 columns

Dataset has 31 categorical features.

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26707 entries, 0 to 26706
Data columns (total 31 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   respondent_id                26707 non-null  int64  
 1   behavioral_antiviral_meds    26636 non-null  float64
 2   behavioral_avoidance         26499 non-null  float64
 3   behavioral_face_mask         26688 non-null  float64
 4   behavioral_wash_hands        26665 non-null  float64
 5   behavioral_large_gatherings  26620 non-null  float64
 6   behavioral_outside_home      26625 non-null  float64
 7   behavioral_touch_face        26579 non-null  float64
 8   doctor_recc_seasonal         24547 non-null  float64
 9   chronic_med_condition        25736 non-null  float64
 10  child_under_6_months         25887 non-null  float64
 11  health_worker                25903 non-null  float64
 12  health_insurance             14433 non-null  float64
 13  opinion_seas_vacc_effective  26245 non-null  float64
 14  opinion_seas_risk            26193 non-null  float64
 15  opinion_seas_sick_from_vacc  26170 non-null  float64
 16  age_group                    26707 non-null  object 
 17  education                    25300 non-null  object 
 18  race                         26707 non-null  object 
 19  sex                          26707 non-null  object 
 20  income_poverty               22284 non-null  object 
 21  marital_status               25299 non-null  object 
 22  rent_or_own                  24665 non-null  object 
 23  employment_status            25244 non-null  object 
 24  hhs_geo_region               26707 non-null  object 
 25  census_msa                   26707 non-null  object 
 26  household_adults             26458 non-null  float64
 27  household_children           26458 non-null  float64
 28  employment_industry          13377 non-null  object 
 29  employment_occupation        13237 non-null  object 
 30  seasonal_vaccine             26707 non-null  int64  
dtypes: float64(17), int64(2), object(12)
memory usage: 6.3+ MB
```

Summary of statistics below:

In [5]:
```
df.describe(include='all').T
```

Out[5]:

	count	unique	top	freq	mean	std	min	25%	50%	7
respondent_id	26707	NaN	NaN	NaN	13353	7709.79	0	6676.5	13353	2002
behavioral_antiviral_meds	26636	NaN	NaN	NaN	0.0488437	0.215545	0	0	0	
behavioral_avoidance	26499	NaN	NaN	NaN	0.725612	0.446214	0	0	1	
behavioral_face_mask	26688	NaN	NaN	NaN	0.0689823	0.253429	0	0	0	
behavioral_wash_hands	26665	NaN	NaN	NaN	0.825614	0.379448	0	1	1	
behavioral_large_gatherings	26620	NaN	NaN	NaN	0.35864	0.47961	0	0	0	
behavioral_outside_home	26625	NaN	NaN	NaN	0.337315	0.472802	0	0	0	
behavioral_touch_face	26579	NaN	NaN	NaN	0.677264	0.467531	0	0	1	
doctor_recc_seasonal	24547	NaN	NaN	NaN	0.329735	0.470126	0	0	0	
chronic_med_condition	25736	NaN	NaN	NaN	0.283261	0.450591	0	0	0	
child_under_6_months	25887	NaN	NaN	NaN	0.0825897	0.275266	0	0	0	
health_worker	25903	NaN	NaN	NaN	0.111918	0.315271	0	0	0	
health_insurance	14433	NaN	NaN	NaN	0.87972	0.3253	0	1	1	
opinion_seas_vacc_effective	26245	NaN	NaN	NaN	4.02599	1.08656	1	4	4	
opinion_seas_risk	26193	NaN	NaN	NaN	2.71916	1.38506	1	2	2	
opinion_seas_sick_from_vacc	26170	NaN	NaN	NaN	2.11811	1.33295	1	1	2	
age_group	26707	5	65+ Years	6843	NaN	NaN	NaN	NaN	NaN	N
education	25300	4	College Graduate	10097	NaN	NaN	NaN	NaN	NaN	N
race	26707	4	White	21222	NaN	NaN	NaN	NaN	NaN	N
sex	26707	2	Female	15858	NaN	NaN	NaN	NaN	NaN	N
income_poverty	22284	3	<= $75,000, Above Poverty	12777	NaN	NaN	NaN	NaN	NaN	N
marital_status	25299	2	Married	13555	NaN	NaN	NaN	NaN	NaN	N
rent_or_own	24665	2	Own	18736	NaN	NaN	NaN	NaN	NaN	N
employment_status	25244	3	Employed	13560	NaN	NaN	NaN	NaN	NaN	N
hhs_geo_region	26707	10	lzgpxyit	4297	NaN	NaN	NaN	NaN	NaN	N
census_msa	26707	3	MSA, Not Principle City	11645	NaN	NaN	NaN	NaN	NaN	N
household_adults	26458	NaN	NaN	NaN	0.886499	0.753422	0	0	1	
household_children	26458	NaN	NaN	NaN	0.534583	0.928173	0	0	0	
employment_industry	13377	21	fcxhlnwr	2468	NaN	NaN	NaN	NaN	NaN	N
employment_occupation	13237	23	xtkaffoo	1778	NaN	NaN	NaN	NaN	NaN	N
seasonal_vaccine	26707	NaN	NaN	NaN	0.465608	0.498825	0	0	0	

Shape of dataset:

In [6]:
```
df.shape
```

Out[6]: (26707, 31)

In [7]:
```
df.columns
```

Out[7]:
```
Index(['respondent_id', 'behavioral_antiviral_meds', 'behavioral_avoidance',
       'behavioral_face_mask', 'behavioral_wash_hands',
       'behavioral_large_gatherings', 'behavioral_outside_home',
       'behavioral_touch_face', 'doctor_recc_seasonal',
       'chronic_med_condition', 'child_under_6_months', 'health_worker',
       'health_insurance', 'opinion_seas_vacc_effective', 'opinion_seas_risk',
       'opinion_seas_sick_from_vacc', 'age_group', 'education', 'race', 'sex',
       'income_poverty', 'marital_status', 'rent_or_own', 'employment_status',
       'hhs_geo_region', 'census_msa', 'household_adults',
       'household_children', 'employment_industry', 'employment_occupation',
       'seasonal_vaccine'],
      dtype='object')
```

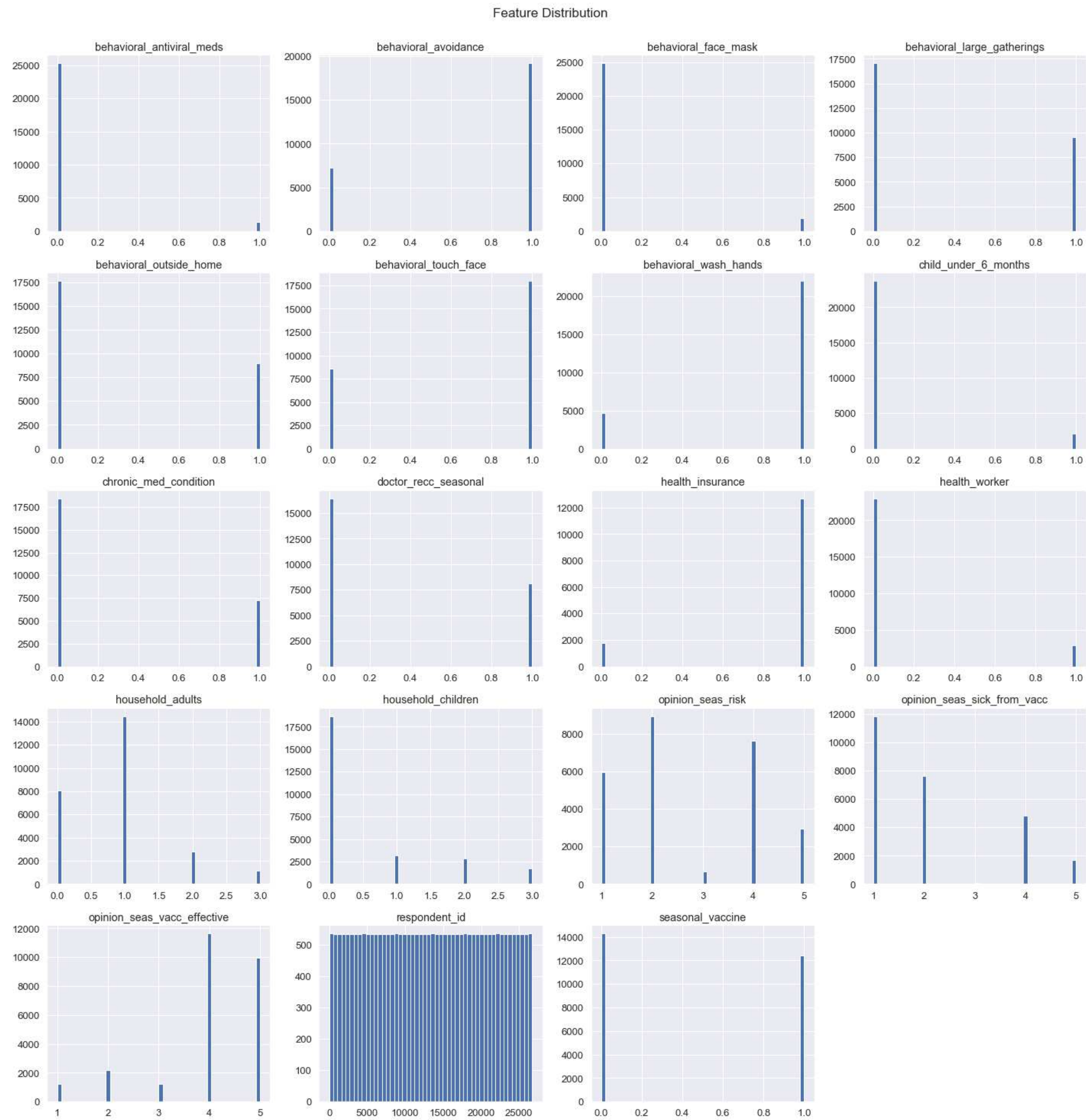
Data Exploration

The best way is to create graphs!

Data Visualization

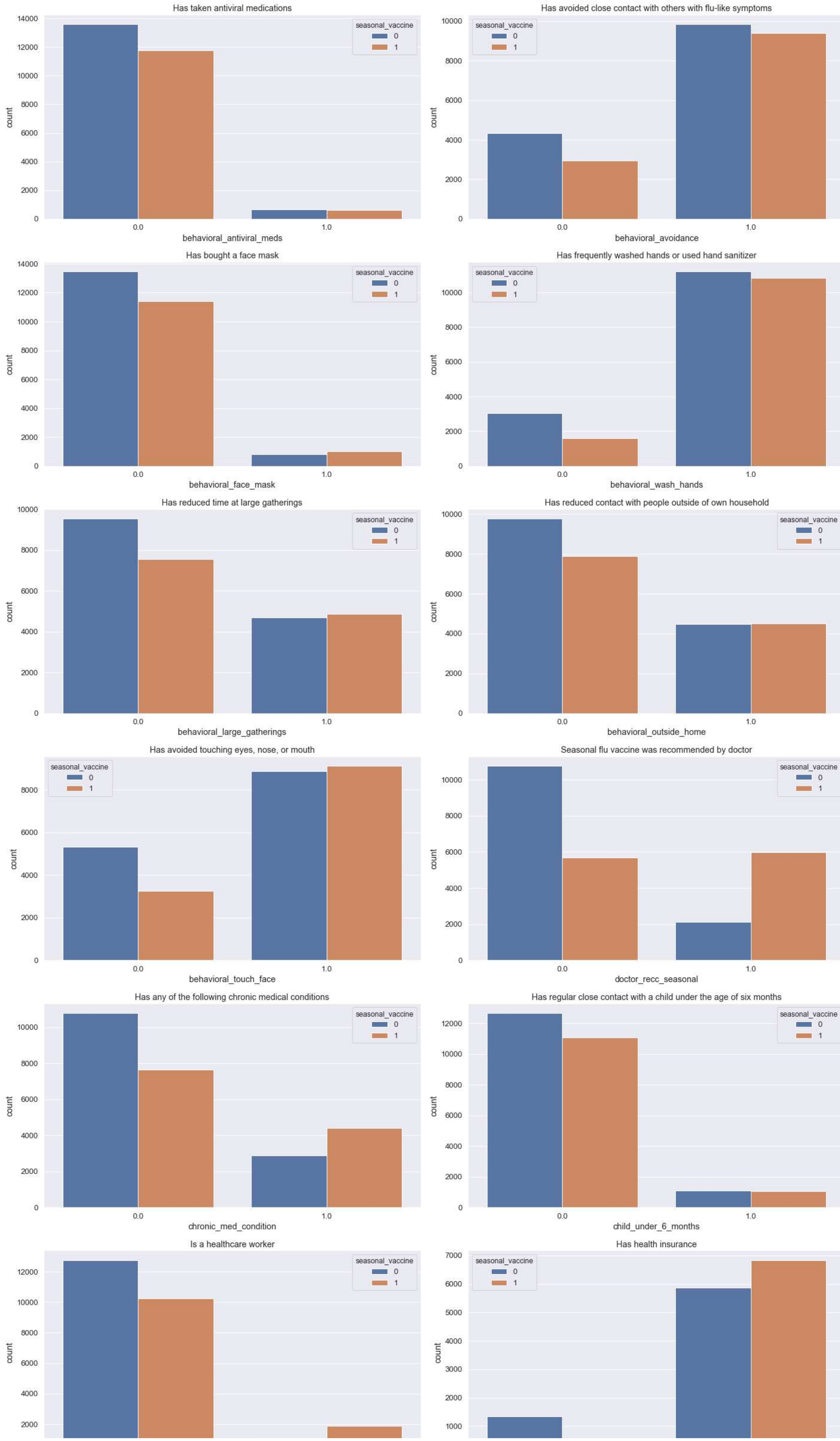
The dataset is mainly discrete/categorical types.

In [8]:
```
df.hist(bins=50, figsize=(20,20))

plt.suptitle('Feature Distribution', x=0.5, y=1.02, ha='center', fontsize='large')

plt.tight_layout()

plt.show();
```



Below are each visuals of the data:

In [10]:
```
fig = plt.figure(figsize=(20,40))

plt.subplot(7,2,1)
plt.title("Has taken antiviral medications")
sns.countplot(df.behavioral_antiviral_meds, hue=df.seasonal_vaccine)

plt.subplot(7,2,2)
plt.title("Has avoided close contact with others with flu-like symptoms")
sns.countplot(df.behavioral_avoidance, hue=df.seasonal_vaccine)

plt.subplot(7,2,3)
plt.title("Has bought a face mask")
sns.countplot(df.behavioral_face_mask, hue=df.seasonal_vaccine)

plt.subplot(7,2,4)
plt.title("Has frequently washed hands or used hand sanitizer")
sns.countplot(df.behavioral_wash_hands, hue=df.seasonal_vaccine)

plt.subplot(7,2,5)
plt.title("Has reduced time at large gatherings")
sns.countplot(df.behavioral_large_gatherings, hue=df.seasonal_vaccine)

plt.subplot(7,2,6)
plt.title("Has reduced contact with people outside of own household")
sns.countplot(df.behavioral_outside_home, hue=df.seasonal_vaccine)

plt.subplot(7,2,7)
plt.title("Has avoided touching eyes, nose, or mouth")
sns.countplot(df.behavioral_touch_face, hue=df.seasonal_vaccine)

plt.subplot(7,2,8)
plt.title("Seasonal flu vaccine was recommended by doctor")
sns.countplot(df.doctor_recc_seasonal, hue=df.seasonal_vaccine)

plt.subplot(7,2,9)
plt.title("Has any of the following chronic medical conditions")
sns.countplot(df.chronic_med_condition, hue=df.seasonal_vaccine)

plt.subplot(7,2,10)
plt.title("Has regular close contact with a child under the age of six months")
sns.countplot(df.child_under_6_months, hue=df.seasonal_vaccine)

plt.subplot(7,2,11)
plt.title("Is a healthcare worker")
sns.countplot(df.health_worker, hue=df.seasonal_vaccine)

plt.subplot(7,2,12)
plt.title("Has health insurance")
sns.countplot(df.health_insurance, hue=df.seasonal_vaccine)


plt.tight_layout()
plt.show()
```



Has taken antiviral medications
behavioral_antiviral_meds
Has avoided close contact with others with flu-like symptoms
behavioral_avoidance
Has bought a face mask
behavioral_face_mask
Has frequently washed hands or used hand sanitizer
behavioral_wash_hands
Has reduced time at large gatherings
behavioral_large_gatherings
Has reduced contact with people outside of own household
behavioral_outside_home
Has avoided touching eyes, nose, or mouth
behavioral_touch_face
Seasonal flu vaccine was recommended by doctor
doctor_recc_seasonal
Has any of the following chronic medical conditions
chronic_med_condition
Has regular close contact with a child under the age of six months
child_under_6_months
Is a healthcare worker
Has health insurance
count
seasonal_vaccine
0
1
0.0
1.0

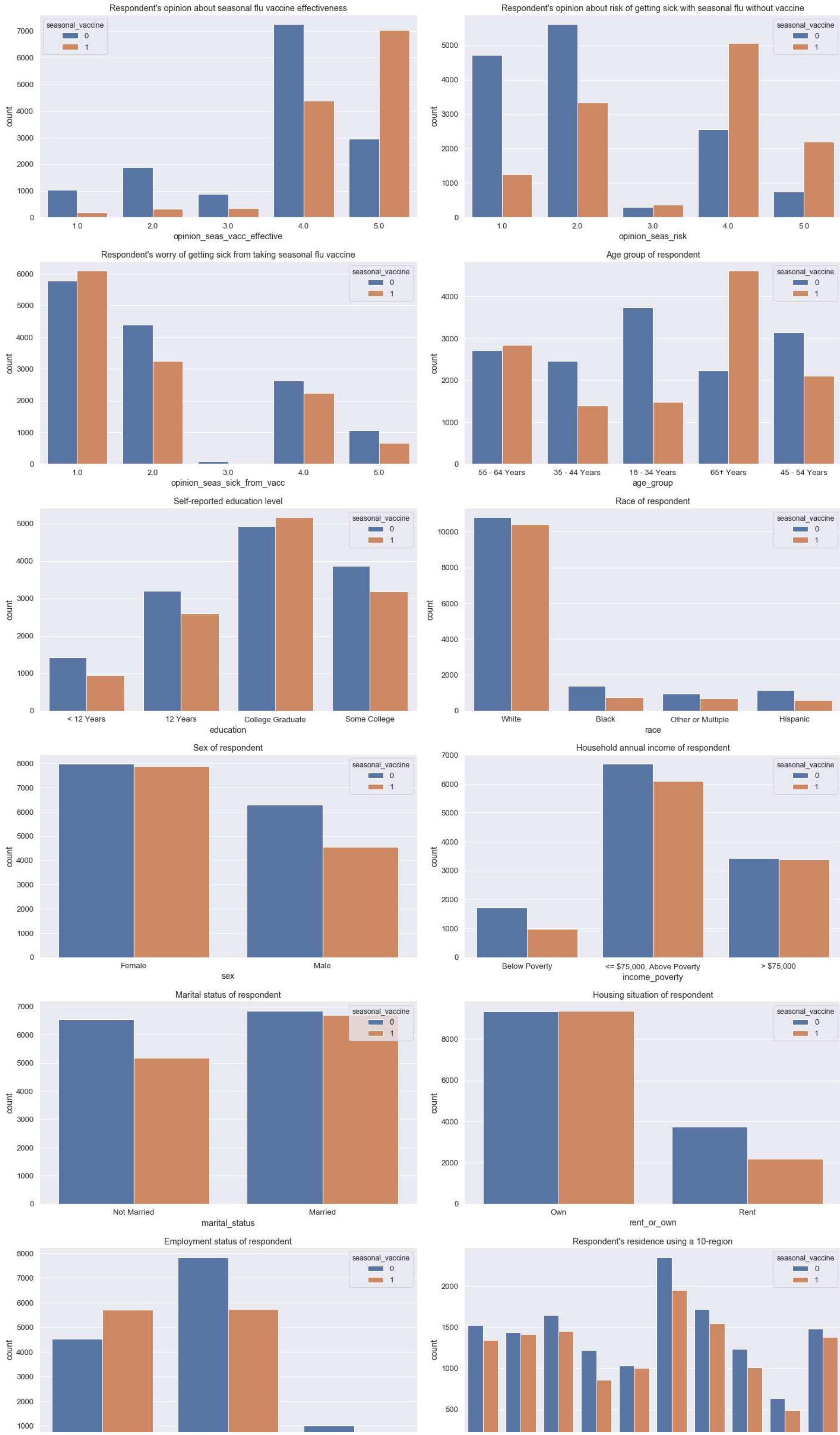
Part 1 of Data Analysis:

Those who had vaccine avoided close contacts which is surprising since vaccines are supposed to protect them. But they didn't avoid large gatherings.

As for flu vaccine which doctor recommended, there is such acceptance among people.

Health Care workers are most vulnerable but majority of them do vaccinate.

In [11]:
```
fig = plt.figure(figsize=(20,40))

plt.subplot(7,2,1)
plt.title("Respondent's opinion about seasonal flu vaccine effectiveness")
sns.countplot(df.opinion_seas_vacc_effective, hue=df.seasonal_vaccine)

plt.subplot(7,2,2)
plt.title("Respondent's opinion about risk of getting sick with seasonal flu withou
t vaccine")
sns.countplot(df.opinion_seas_risk, hue=df.seasonal_vaccine)

plt.subplot(7,2,3)
plt.title("Respondent's worry of getting sick from taking seasonal flu vaccine")
sns.countplot(df.opinion_seas_sick_from_vacc, hue=df.seasonal_vaccine)

plt.subplot(7,2,4)
plt.title("Age group of respondent")
sns.countplot(df.age_group, hue=df.seasonal_vaccine)

plt.subplot(7,2,5)
plt.title("Self-reported education level")
sns.countplot(df.education, hue=df.seasonal_vaccine)

plt.subplot(7,2,6)
plt.title("Race of respondent")
sns.countplot(df.race, hue=df.seasonal_vaccine)

plt.subplot(7,2,7)
plt.title("Sex of respondent")
sns.countplot(df.sex, hue=df.seasonal_vaccine)

plt.subplot(7,2,8)
plt.title("Household annual income of respondent")
sns.countplot(df.income_poverty, hue=df.seasonal_vaccine)

plt.subplot(7,2,9)
plt.title("Marital status of respondent")
sns.countplot(df.marital_status, hue=df.seasonal_vaccine)

plt.subplot(7,2,10)
plt.title("Housing situation of respondent")
sns.countplot(df.rent_or_own, hue=df.seasonal_vaccine)

plt.subplot(7,2,11)
plt.title("Employment status of respondent")
sns.countplot(df.employment_status, hue=df.seasonal_vaccine)

plt.subplot(7,2,12)
plt.title("Respondent's residence using a 10-region")
sns.countplot(df.hhs_geo_region, hue=df.seasonal_vaccine)



plt.tight_layout()
plt.show()
```


Respondent's opinion about seasonal flu vaccine effectiveness
seasonal_vaccine
0
1
count
opinion_seas_vacc_effective
Respondent's opinion about risk of getting sick with seasonal flu without vaccine
opinion_seas_risk
Respondent's worry of getting sick from taking seasonal flu vaccine
opinion_seas_sick_from_vacc
Age group of respondent
55 - 64 Years
35 - 44 Years
18 - 34 Years
65+ Years
45 - 54 Years
age_group
Self-reported education level
< 12 Years
12 Years
College Graduate
Some College
education
Race of respondent
White
Black
Other or Multiple
Hispanic
race
Sex of respondent
Female
Male
sex
Household annual income of respondent
Below Poverty
<= $75,000, Above Poverty
> $75,000
income_poverty
Marital status of respondent
Not Married
Married
marital_status
Housing situation of respondent
Own
Rent
rent_or_own
Employment status of respondent
Respondent's residence using a 10-region

Part 2 of Data Analysis:

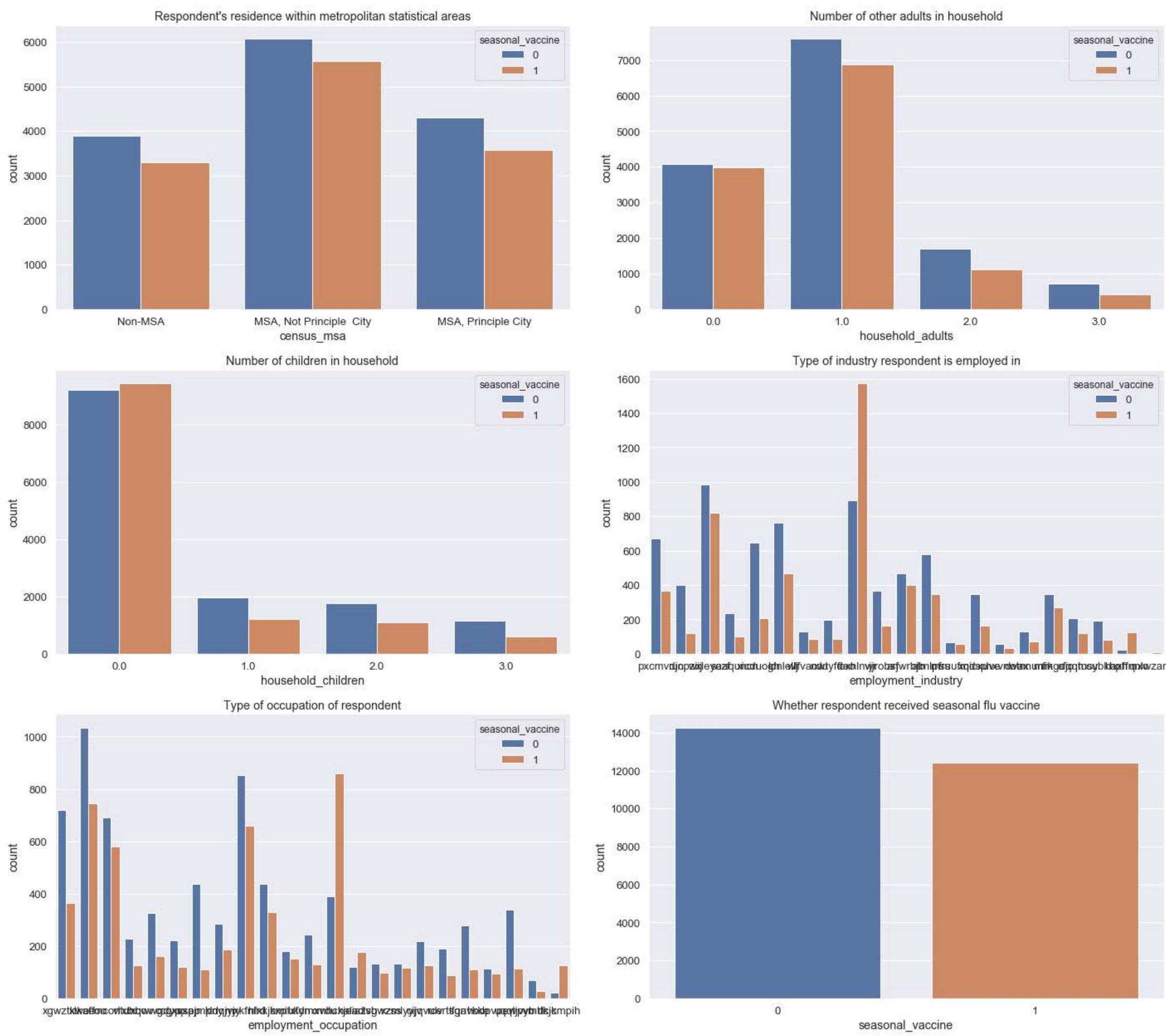
As for respondents opinion, risks and worry, there are no surprises for those who trust flu vaccines.

Respondents more than age 65 and College Educated are vaccinated.

Mainly whites, female, more than 75k income, married, own a house and employed can afford flu vaccines.

In [12]:

```python
fig = plt.figure(figsize=(20,40))

plt.subplot(7,2,1)
plt.title("Respondent's residence within metropolitan statistical areas")
sns.countplot(df.census_msa, hue=df.seasonal_vaccine)

plt.subplot(7,2,2)
plt.title("Number of other adults in household")
sns.countplot(df.household_adults, hue=df.seasonal_vaccine)

plt.subplot(7,2,3)
plt.title("Number of children in household")
sns.countplot(df.household_children, hue=df.seasonal_vaccine)

plt.subplot(7,2,4)
plt.title("Type of industry respondent is employed in")
sns.countplot(df.employment_industry, hue=df.seasonal_vaccine)

plt.subplot(7,2,5)
plt.title("Type of occupation of respondent")
sns.countplot(df.employment_occupation, hue=df.seasonal_vaccine)

plt.subplot(7,2,6)
plt.title("Whether respondent received seasonal flu vaccine")
sns.countplot(df.seasonal_vaccine)


plt.tight_layout()
plt.show()
```



Part 3 of Data Analysis:

City dwellers, one household adults and no children mainly are vaccinated.

Unknown employment industry and occupation type is masked/not revealed to us.

As for seasonal vaccine, both are more or less equal quantity.

In [13]:
```
df['seasonal_vaccine'].value_counts()
```

Out[13]:
```
0    14272
1    12435
Name: seasonal_vaccine, dtype: int64
```

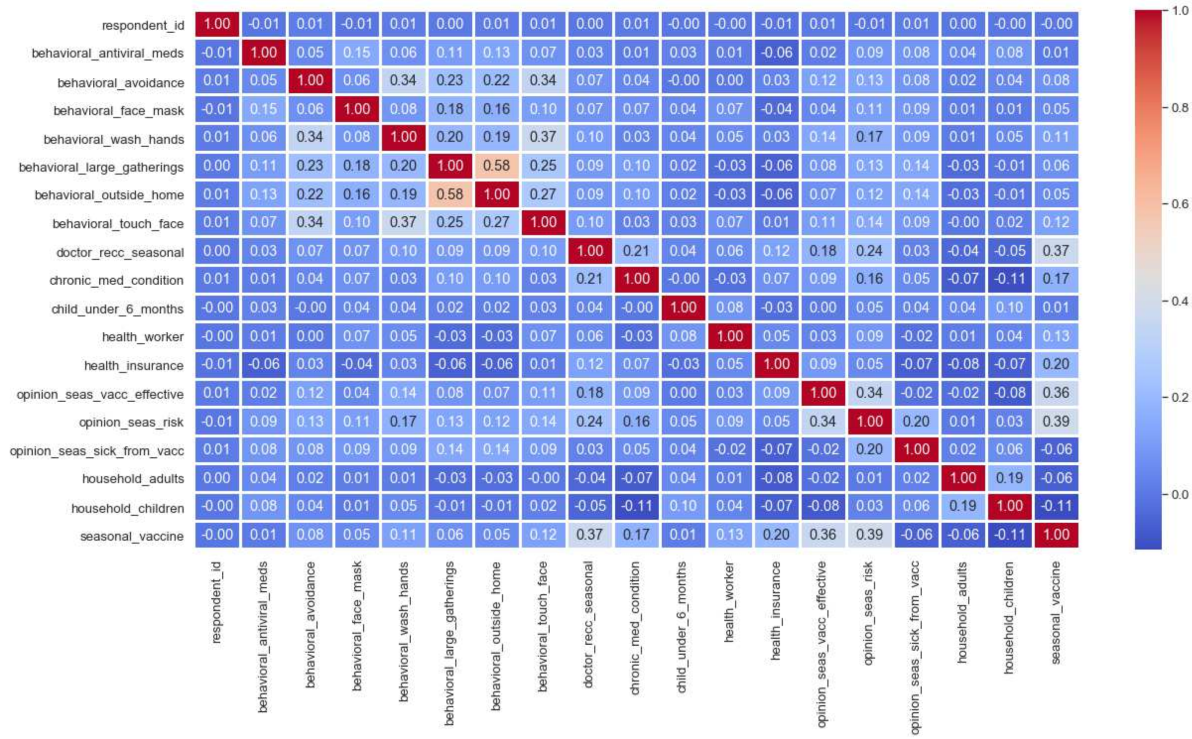
Now we check any correlation between features:

In [14]:
```
df.corr()
```

Out[14]:

	respondent_id	**behavioral_antiviral_meds**	**behavioral_avoidance**	**behavioral_face_ı**
respondent_id	1.000000	-0.008475	0.009638	-0.00
behavioral_antiviral_meds	-0.008475	1.000000	0.049247	0.14
behavioral_avoidance	0.009638	0.049247	1.000000	0.06
behavioral_face_mask	-0.006644	0.146261	0.064946	1.00
behavioral_wash_hands	0.011105	0.064119	0.338130	0.08
behavioral_large_gatherings	0.004539	0.106287	0.227675	0.18
behavioral_outside_home	0.009011	0.127679	0.220348	0.16
behavioral_touch_face	0.007575	0.070868	0.335335	0.10
doctor_recc_seasonal	0.001500	0.030909	0.074088	0.06
chronic_med_condition	0.005797	0.008465	0.039435	0.06
child_under_6_months	-0.004839	0.028788	-0.000414	0.03
health_worker	-0.003149	0.009465	0.001180	0.06
health_insurance	-0.012603	-0.063988	0.032662	-0.04
opinion_seas_vacc_effective	0.005935	0.015003	0.119554	0.04
opinion_seas_risk	-0.005291	0.085315	0.129504	0.11
opinion_seas_sick_from_vacc	0.009563	0.084305	0.082942	0.09
household_adults	0.000187	0.044900	0.019122	0.01
household_children	-0.003726	0.084822	0.040328	0.00
seasonal_vaccine	-0.004652	0.006277	0.076395	0.05

In [15]:
```python
plt.figure(figsize=(20,10))
sns.heatmap(df.corr(),cmap="coolwarm",annot=True,fmt='.2f',linewidths=2)
plt.show()
```



Quote:

"Factors that may bias the results of observational studies can be broadly categorized as: selection bias resulting from the way study subjects are recruited or from differing rates of study participation depending on the subjects' cultural background, age, or socioeconomic status, information bias, measurement error, confounders, and further factors."

We will drop a number of features which we think that will make the model biased to a certain group/gender/income/social.

Drop unwanted features

In [16]:
```python
df.columns
```

Out[16]:
```
Index(['respondent_id', 'behavioral_antiviral_meds', 'behavioral_avoidance',
       'behavioral_face_mask', 'behavioral_wash_hands',
       'behavioral_large_gatherings', 'behavioral_outside_home',
       'behavioral_touch_face', 'doctor_recc_seasonal',
       'chronic_med_condition', 'child_under_6_months', 'health_worker',
       'health_insurance', 'opinion_seas_vacc_effective', 'opinion_seas_risk',
       'opinion_seas_sick_from_vacc', 'age_group', 'education', 'race', 'sex',
       'income_poverty', 'marital_status', 'rent_or_own', 'employment_status',
       'hhs_geo_region', 'census_msa', 'household_adults',
       'household_children', 'employment_industry', 'employment_occupation',
       'seasonal_vaccine'],
      dtype='object')
```

In [17]:
```python
df.drop(['respondent_id','health_insurance','age_group', 'education', 'race', 'sex
', 'income_poverty', 'marital_status', 'rent_or_own',
          'employment_status','hhs_geo_region', 'census_msa', 'household_adults', 'h
ousehold_children', 'employment_industry',
           'employment_occupation'],axis=1,inplace=True)
```

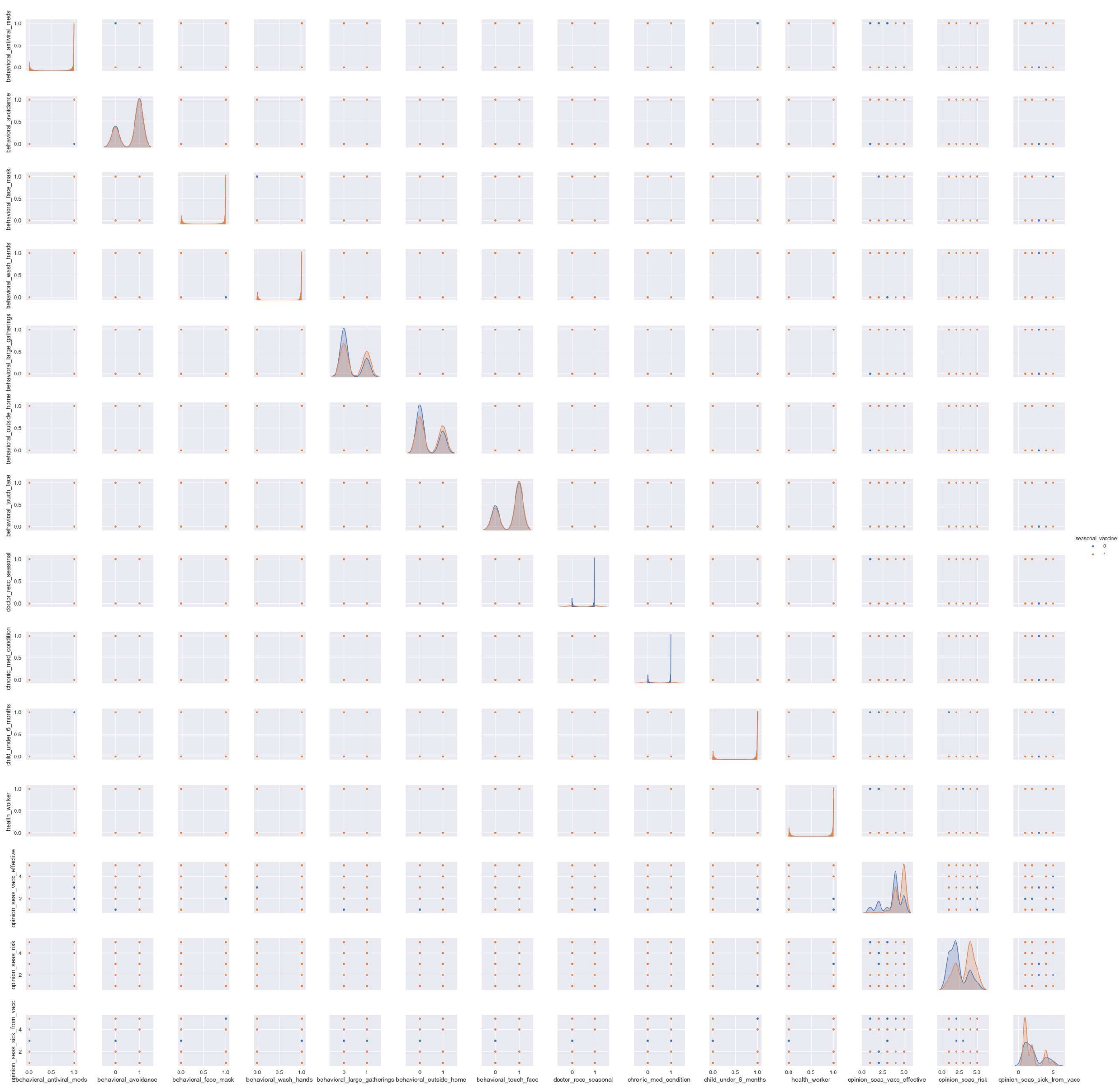
In [18]:
```
df
```

Out[18]:

	behavioral_antiviral_meds	behavioral_avoidance	behavioral_face_mask	behavioral_wash_hands	behavio
0	0.0	0.0	0.0	0.0	
1	0.0	1.0	0.0	1.0	
2	0.0	1.0	0.0	0.0	
3	0.0	1.0	0.0	1.0	
4	0.0	1.0	0.0	1.0	
...	...	...	...	...	
26702	0.0	1.0	0.0	0.0	
26703	0.0	1.0	0.0	1.0	
26704	0.0	1.0	1.0	1.0	
26705	0.0	0.0	0.0	0.0	
26706	0.0	1.0	0.0	0.0	

26707 rows × 15 columns

In [19]:
```
sns.pairplot(df.sample(500), hue='seasonal_vaccine')
plt.show()
```



Treat Missing Values

In [20]:
```
df.isnull().sum()
```

Out[20]:
```
behavioral_antiviral_meds        71
behavioral_avoidance            208
behavioral_face_mask             19
behavioral_wash_hands            42
behavioral_large_gatherings      87
behavioral_outside_home          82
behavioral_touch_face           128
doctor_recc_seasonal           2160
chronic_med_condition           971
child_under_6_months            820
health_worker                   804
opinion_seas_vacc_effective     462
opinion_seas_risk               514
opinion_seas_sick_from_vacc     537
seasonal_vaccine                  0
dtype: int64
```

In [21]:
```
df.dropna(inplace=True)
```

In [22]:
```
df.isnull().sum()
```

Out[22]:
```
behavioral_antiviral_meds      0
behavioral_avoidance           0
behavioral_face_mask           0
behavioral_wash_hands          0
behavioral_large_gatherings    0
behavioral_outside_home        0
behavioral_touch_face          0
doctor_recc_seasonal           0
chronic_med_condition          0
child_under_6_months           0
health_worker                  0
opinion_seas_vacc_effective    0
opinion_seas_risk              0
opinion_seas_sick_from_vacc    0
seasonal_vaccine               0
dtype: int64
```

In [23]:
```
df.reset_index(drop=True,inplace=True)
```

In [24]:
```
df
```

Out[24]:

	behavioral_antiviral_meds	behavioral_avoidance	behavioral_face_mask	behavioral_wash_hands	behavio
0	0.0	0.0	0.0	0.0	
1	0.0	1.0	0.0	1.0	
2	0.0	1.0	0.0	1.0	
3	0.0	1.0	0.0	1.0	
4	0.0	1.0	0.0	1.0	
...	...	...	...	...	
23183	0.0	0.0	0.0	1.0	
23184	0.0	1.0	0.0	0.0	
23185	0.0	1.0	0.0	1.0	
23186	0.0	1.0	1.0	1.0	
23187	0.0	1.0	0.0	0.0	

23188 rows × 15 columns

In [25]:
```
df['seasonal_vaccine'].value_counts()
```

Out[25]:
```
0    12111
1    11077
Name: seasonal_vaccine, dtype: int64
```

In [26]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23188 entries, 0 to 23187
Data columns (total 15 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   behavioral_antiviral_meds    23188 non-null  float64
 1   behavioral_avoidance         23188 non-null  float64
 2   behavioral_face_mask         23188 non-null  float64
 3   behavioral_wash_hands        23188 non-null  float64
 4   behavioral_large_gatherings  23188 non-null  float64
 5   behavioral_outside_home      23188 non-null  float64
 6   behavioral_touch_face        23188 non-null  float64
 7   doctor_recc_seasonal         23188 non-null  float64
 8   chronic_med_condition        23188 non-null  float64
 9   child_under_6_months         23188 non-null  float64
 10  health_worker                23188 non-null  float64
 11  opinion_seas_vacc_effective  23188 non-null  float64
 12  opinion_seas_risk            23188 non-null  float64
 13  opinion_seas_sick_from_vacc  23188 non-null  float64
 14  seasonal_vaccine             23188 non-null  int64  
dtypes: float64(14), int64(1)
memory usage: 2.7 MB
```

In [27]:
```
df = df.astype('int8')  #Change to integer type
```

In [28]:
```
df.dtypes
```

Out[28]:
```
behavioral_antiviral_meds      int8
behavioral_avoidance           int8
behavioral_face_mask           int8
behavioral_wash_hands          int8
behavioral_large_gatherings    int8
behavioral_outside_home        int8
behavioral_touch_face          int8
doctor_recc_seasonal           int8
chronic_med_condition          int8
child_under_6_months           int8
health_worker                  int8
opinion_seas_vacc_effective    int8
opinion_seas_risk              int8
opinion_seas_sick_from_vacc    int8
seasonal_vaccine               int8
dtype: object
```

In [29]:
```
df
```

Out[29]:

	behavioral_antiviral_meds	behavioral_avoidance	behavioral_face_mask	behavioral_wash_hands	behavio
0	0	0	0	0	
1	0	1	0	1	
2	0	1	0	1	
3	0	1	0	1	
4	0	1	0	1	
...	...	...	...	...	
23183	0	0	0	1	
23184	0	1	0	0	
23185	0	1	0	1	
23186	0	1	1	1	
23187	0	1	0	0	

23188 rows × 15 columns

In [30]:
```
df.describe()
```

Out[30]:

	behavioral_antiviral_meds	behavioral_avoidance	behavioral_face_mask	behavioral_wash_hands	behavio
count	23188.000000	23188.000000	23188.000000	23188.000000	
mean	0.049336	0.731197	0.068139	0.829481	
std	0.216573	0.443347	0.251989	0.376096	
min	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	1.000000	
50%	0.000000	1.000000	0.000000	1.000000	
75%	0.000000	1.000000	0.000000	1.000000	
max	1.000000	1.000000	1.000000	1.000000	

In [31]:
```
df['opinion_seas_vacc_effective'] = df['opinion_seas_vacc_effective'].astype('objec
t')
```

In [32]:
```
df['opinion_seas_risk'] = df['opinion_seas_risk'].astype('object')
```

In [33]:
```
df['opinion_seas_sick_from_vacc'] = df['opinion_seas_sick_from_vacc'].astype('objec
t')
```

In [34]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23188 entries, 0 to 23187
Data columns (total 15 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   behavioral_antiviral_meds    23188 non-null  int8
 1   behavioral_avoidance         23188 non-null  int8
 2   behavioral_face_mask         23188 non-null  int8
 3   behavioral_wash_hands        23188 non-null  int8
 4   behavioral_large_gatherings  23188 non-null  int8
 5   behavioral_outside_home      23188 non-null  int8
 6   behavioral_touch_face        23188 non-null  int8
 7   doctor_recc_seasonal         23188 non-null  int8
 8   chronic_med_condition        23188 non-null  int8
 9   child_under_6_months         23188 non-null  int8
 10  health_worker                23188 non-null  int8
 11  opinion_seas_vacc_effective  23188 non-null  object
 12  opinion_seas_risk            23188 non-null  object
 13  opinion_seas_sick_from_vacc  23188 non-null  object
 14  seasonal_vaccine             23188 non-null  int8
dtypes: int8(12), object(3)
memory usage: 815.3+ KB
```

Create dummy variables

In [35]:
```
df2 = pd.get_dummies(data=df, drop_first=True)
```

In [36]:
```
df2
```

Out[36]:

	behavioral_antiviral_meds	behavioral_avoidance	behavioral_face_mask	behavioral_wash_hands	behavio
0	0	0	0	0	
1	0	1	0	1	
2	0	1	0	1	
3	0	1	0	1	
4	0	1	0	1	
...	...	...	...	...	
23183	0	0	0	1	
23184	0	1	0	0	
23185	0	1	0	1	
23186	0	1	1	1	
23187	0	1	0	0	

23188 rows × 24 columns

In [37]:
```
df2.columns
```

Out[37]:
```
Index(['behavioral_antiviral_meds', 'behavioral_avoidance',
       'behavioral_face_mask', 'behavioral_wash_hands',
       'behavioral_large_gatherings', 'behavioral_outside_home',
       'behavioral_touch_face', 'doctor_recc_seasonal',
       'chronic_med_condition', 'child_under_6_months', 'health_worker',
       'seasonal_vaccine', 'opinion_seas_vacc_effective_2',
       'opinion_seas_vacc_effective_3', 'opinion_seas_vacc_effective_4',
       'opinion_seas_vacc_effective_5', 'opinion_seas_risk_2',
       'opinion_seas_risk_3', 'opinion_seas_risk_4', 'opinion_seas_risk_5',
       'opinion_seas_sick_from_vacc_2', 'opinion_seas_sick_from_vacc_3',
       'opinion_seas_sick_from_vacc_4', 'opinion_seas_sick_from_vacc_5'],
      dtype='object')
```

In [38]:
```
df2 = df2[['behavioral_antiviral_meds', 'behavioral_avoidance',
       'behavioral_face_mask', 'behavioral_wash_hands',
       'behavioral_large_gatherings', 'behavioral_outside_home',
       'behavioral_touch_face', 'doctor_recc_seasonal',
       'chronic_med_condition', 'child_under_6_months', 'health_worker',
       'opinion_seas_vacc_effective_2',
       'opinion_seas_vacc_effective_3', 'opinion_seas_vacc_effective_4',
       'opinion_seas_vacc_effective_5', 'opinion_seas_risk_2',
       'opinion_seas_risk_3', 'opinion_seas_risk_4', 'opinion_seas_risk_5',
       'opinion_seas_sick_from_vacc_2', 'opinion_seas_sick_from_vacc_3',
       'opinion_seas_sick_from_vacc_4', 'opinion_seas_sick_from_vacc_5','seasonal_v
accine' ]]
```

In [39]:
```
df2
```

Out[39]:

	behavioral_antiviral_meds	behavioral_avoidance	behavioral_face_mask	behavioral_wash_hands	behavio
0	0	0	0	0	
1	0	1	0	1	
2	0	1	0	1	
3	0	1	0	1	
4	0	1	0	1	
...	...	...	...	...	
23183	0	0	0	1	
23184	0	1	0	0	
23185	0	1	0	1	
23186	0	1	1	1	
23187	0	1	0	0	

23188 rows × 24 columns

Create and save processed dataset

In [40]:
```
df2.to_csv("train.csv",index=False)
```

In [ ]:

In [41]:
```
df = pd.read_csv("train.csv")
```

In [42]:
```
df
```

Out[42]:

	behavioral_antiviral_meds	behavioral_avoidance	behavioral_face_mask	behavioral_wash_hands	behavio
0	0	0	0	0	
1	0	1	0	1	
2	0	1	0	1	
3	0	1	0	1	
4	0	1	0	1	
...	...	...	...	...	
23183	0	0	0	1	
23184	0	1	0	0	
23185	0	1	0	1	
23186	0	1	1	1	
23187	0	1	0	0	

23188 rows × 24 columns

In [43]:
```
df.shape
```

Out[43]:
```
(23188, 24)
```

Summary of training at least three different classifier models, preferably of different nature in explainability and predictability. For example, you can start with a simple logistic regression as a baseline, adding other models or ensemble models. Preferably, all your models use the same training and test splits, or the same cross-validation method.

Train Test Split

In [44]:
```
X = df.iloc[:,0:23]
y = df.iloc[:,23]
```

In [45]:
```
X.values
```

Out[45]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 1, 0, ..., 0, 1, 0],
       [0, 1, 0, ..., 0, 0, 0],
       ...,
       [0, 1, 0, ..., 0, 0, 0],
       [0, 1, 1, ..., 0, 0, 0],
       [0, 1, 0, ..., 0, 0, 0]], dtype=int64)
```

In [46]:
```
y.values
```

Out[46]:
```
array([0, 1, 1, ..., 0, 1, 0], dtype=int64)
```

In [47]:
```
X_train, X_test, y_train, y_test = train_test_split(X.values, y.values, test_size=
0.2, random_state=123, stratify=y)
```

In [48]:
```
X_train
```

Out[48]:
```
array([[0, 1, 0, ..., 0, 0, 0],
       [0, 1, 0, ..., 0, 0, 0],
       [0, 1, 0, ..., 0, 0, 0],
       ...,
       [0, 1, 1, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 1, 0, ..., 0, 0, 0]], dtype=int64)
```

In [49]:
```
X_test
```

Out[49]:
```
array([[0, 1, 0, ..., 0, 0, 0],
       [0, 1, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0],
       [0, 1, 0, ..., 0, 0, 0]], dtype=int64)
```

In [50]:
```
y_train
```

Out[50]:
```
array([1, 0, 1, ..., 1, 1, 1], dtype=int64)
```

Logistic Regression

In [51]:
```
lr = LogisticRegression(random_state=123)
```

In [52]:
```
lr.fit(X_train, y_train)
```

Out[52]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=123, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [53]:
```
lr.coef_
```

Out[53]:
```
array([[-0.23062336, -0.07794305, -0.00527337,  0.09815467,  0.00333909,
        -0.07933634,  0.27948713,  1.36430408,  0.35155097, -0.16748045,
         0.79348647, -0.29698112,  0.6244859 ,  0.75495696,  1.77721418,
         0.80614637,  1.6792209 ,  1.70246871,  2.0063813 , -0.44950573,
        -1.62206222, -0.67756796, -1.24302248]])
```

In [54]:
```
lr.intercept_
```

Out[54]:
```
array([-2.63615376])
```

In [55]:
```
ypred_lr = lr.predict(X_test)
```

In [56]:
```
y_test[:10]
```

Out[56]:
```
array([1, 0, 1, 0, 1, 1, 1, 0, 1, 1], dtype=int64)
```

In [57]:
```
ypred_lr[:10]
```

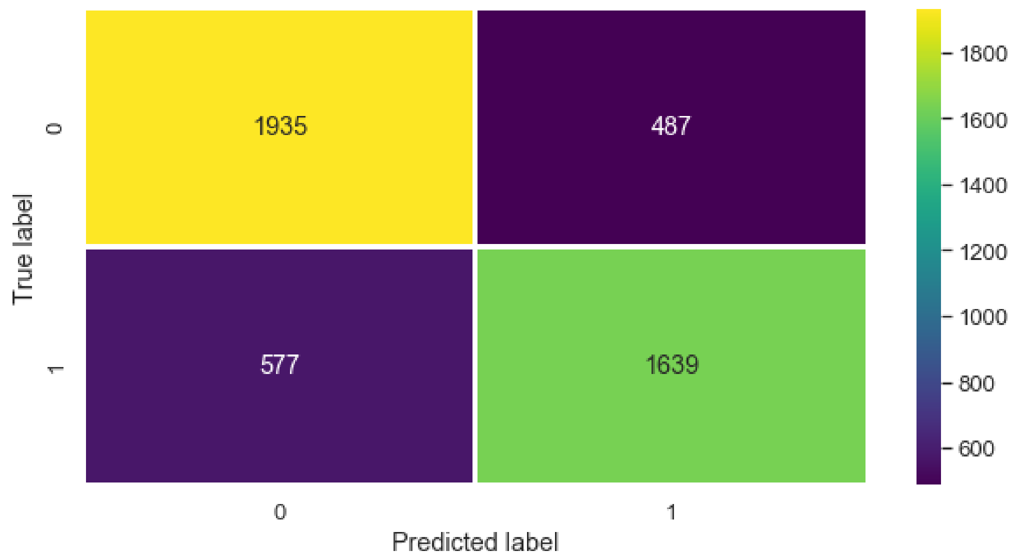
Out[57]:
```
array([1, 0, 0, 0, 1, 1, 1, 0, 0, 1], dtype=int64)
```

Logistic Regression Model Evaluation

In [58]:
```
cm = confusion_matrix(y_test,ypred_lr)
cm
```

Out[58]:
```
array([[1935,  487],
       [ 577, 1639]], dtype=int64)
```

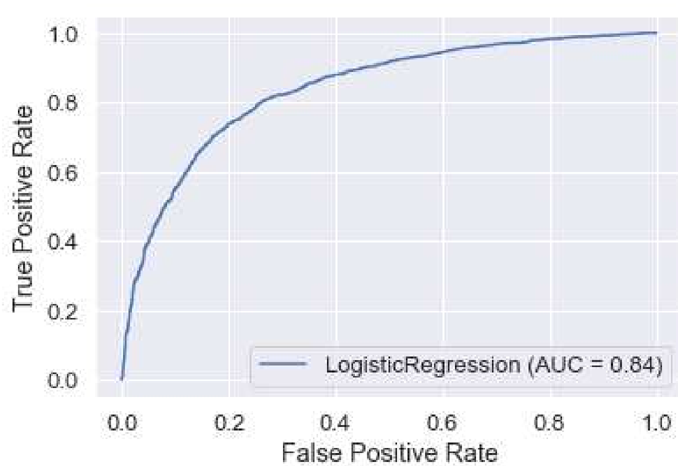
In [59]:
```
fig , ax = plt.subplots(figsize=(10,5))
sns.heatmap(cm, annot=True,fmt='.4g',linewidths=2, cmap='viridis')
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```



In [60]:
```
print(classification_report(y_test,ypred_lr))
```

```
              precision    recall  f1-score   support

           0       0.77      0.80      0.78      2422
           1       0.77      0.74      0.75      2216

    accuracy                           0.77      4638
   macro avg       0.77      0.77      0.77      4638
weighted avg       0.77      0.77      0.77      4638
```

In [61]:
```
plot_roc_curve(lr,X_test,y_test)
plt.show()
```



In [62]:
```
accuracy_score(y_test,ypred_lr)
```
Out[62]:
```
0.7705907718844329
```

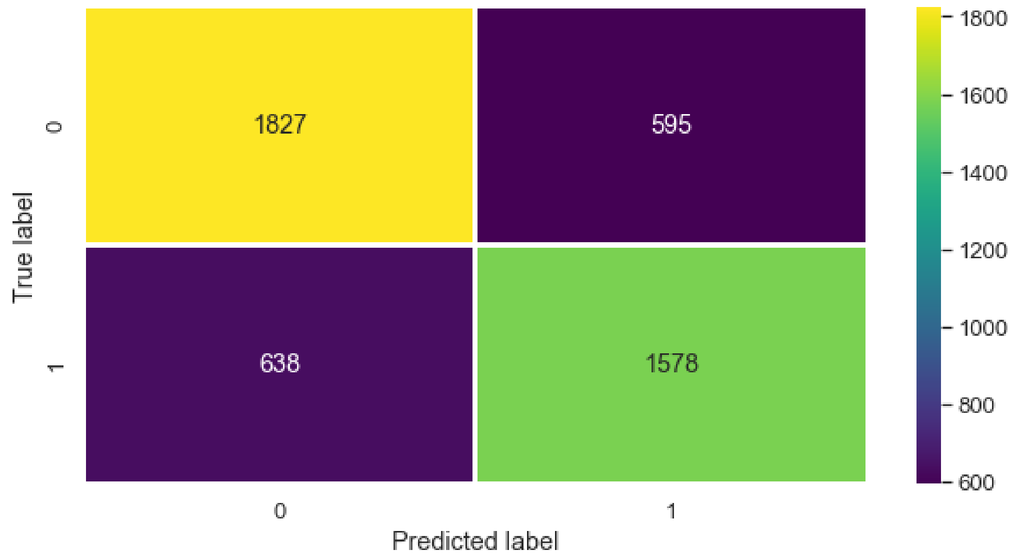
Random Forest Classifier

In [63]:
```
rf = RandomForestClassifier(random_state=123)
```
In [64]:
```
rf.fit(X_train, y_train)
```
Out[64]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=123,
                       verbose=0, warm_start=False)
```
In [65]:
```
ypred_rf = rf.predict(X_test)
```
In [66]:
```
y_test[:10]
```
Out[66]:
```
array([1, 0, 1, 0, 1, 1, 1, 0, 1, 1], dtype=int64)
```
In [67]:
```
ypred_rf[:10]
```
Out[67]:
```
array([1, 0, 0, 0, 1, 1, 1, 1, 0, 1], dtype=int64)
```

Random Forest Model Evaluation

In [68]:
```
cm = confusion_matrix(y_test,ypred_rf)
cm
```
Out[68]:
```
array([[1827,  595],
       [ 638, 1578]], dtype=int64)
```

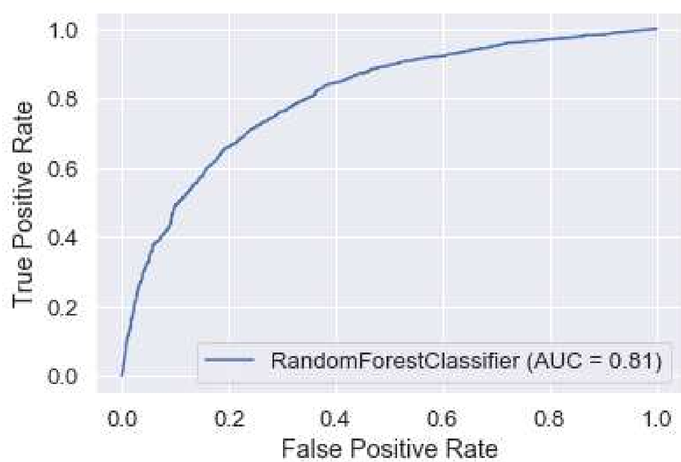
In [69]:
```
fig , ax = plt.subplots(figsize=(10,5))
sns.heatmap(cm, annot=True,fmt='.4g',linewidths=2, cmap='viridis')
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```



In [70]:
```
print(classification_report(y_test,ypred_rf))
```

```
              precision    recall  f1-score   support

           0       0.74      0.75      0.75      2422
           1       0.73      0.71      0.72      2216

    accuracy                           0.73      4638
   macro avg       0.73      0.73      0.73      4638
weighted avg       0.73      0.73      0.73      4638
```

In [71]:
```
plot_roc_curve(rf,X_test,y_test)
plt.show()
```



In [72]:
```
accuracy_score(y_test,ypred_rf)
```

Out[72]: 0.7341526520051747

Gradient Boosting Classifer

In [73]:
```
gbc = GradientBoostingClassifier(random_state=123)
```

In [74]:
```
gbc.fit(X_train,y_train)
```

Out[74]:
```
GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='deviance', max_depth=3,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=100,
                           n_iter_no_change=None, presort='deprecated',
                           random_state=123, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)
```

In [75]:
```
ypredgbc = gbc.predict(X_test)
```

In [76]:
```
y_test[:10]
```

Out[76]:
```
array([1, 0, 1, 0, 1, 1, 1, 0, 1, 1], dtype=int64)
```

In [77]:
```
ypredgbc[:10]
```

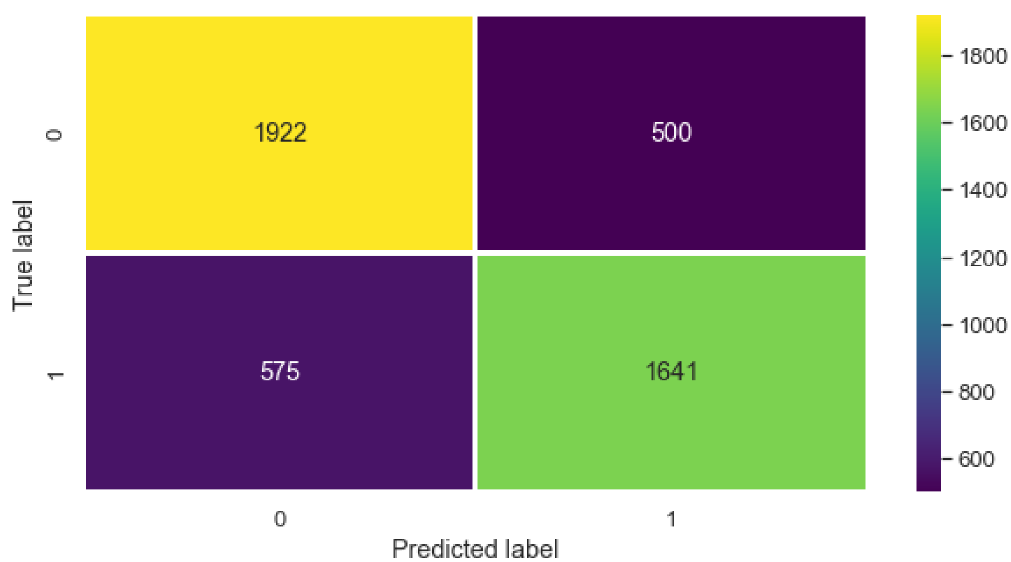
Out[77]:
```
array([1, 0, 0, 0, 1, 1, 1, 1, 0, 1], dtype=int64)
```

Gradient Boosting Model Evaluation

In [78]:
```
cm = confusion_matrix(y_test,ypredgbc)
cm
```

Out[78]:
```
array([[1922,  500],
       [ 575, 1641]], dtype=int64)
```

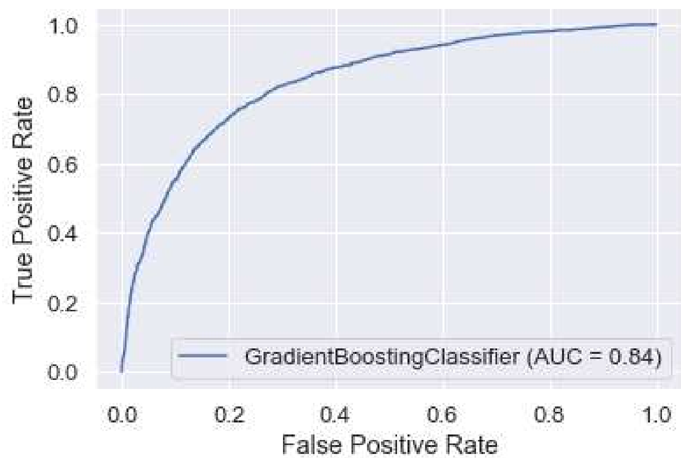
In [79]:
```
fig , ax = plt.subplots(figsize=(10,5))
sns.heatmap(cm, annot=True,fmt='.4g',linewidths=2, cmap='viridis')
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```



In [80]:
```
print(classification_report(y_test,ypredgbc))
```

```
              precision    recall  f1-score   support

           0       0.77      0.79      0.78      2422
           1       0.77      0.74      0.75      2216

    accuracy                           0.77      4638
   macro avg       0.77      0.77      0.77      4638
weighted avg       0.77      0.77      0.77      4638
```

In [81]:
```
plot_roc_curve(gbc,X_test,y_test)
plt.show()
```



In [82]:
```
accuracy_score(y_test,ypredgbc)
```

Out[82]: 0.7682190599396291

A paragraph explaining which of your classifier models you recommend as a final model that best fits your needs in terms of accuracy and explainability.

Logistic Regression gives us the best accuracy and F1 score. Therefore it is recommended.

Summary Key Findings and Insights, which walks your reader through the main drivers of your model and insights from your data derived from your classifier model.

The features we selected gave us a decent accuracy and good result. The result differences are small and we select Logistic Regression because it's a simple model.

Suggestions for next steps in analyzing this data, which may include suggesting revisiting this model adding specific data features to achieve a better explanation or a better prediction.

For features that are biased, we need to gather more data and made equal values for race, sex, income etc. We have to ensure the model we developed stays bias free.

We can also explore other models like decision tree, support vector machine, KNN classifiers model to see if they can able to analyse the data patterns to give better predictions. We also can adjust hyperparameters for each model to get better results.