Project

- This dataset is part of an active competition until March 31, 2022!
- As the world struggles to vaccinate the global population against COVID-19, an understanding of how people’s backgrounds, opinions, and health behaviors are related to their personal vaccination patterns can provide guidance for future public health efforts. Your audience could be someone guiding those public health efforts.
- This challenge: can you predict whether people got H1N1 and seasonal flu vaccines using data collected in the National 2009 H1N1 Flu Survey? This is a binary classification problem, but there are two potential targets: whether the survey respondent received the seasonal flu vaccine, or whether the respondent received the H1N1 flu vaccine. Please choose just one of these potential targets for your minimum viable project.

Business Problem

- Stakeholder:
  - Public health organizations or healthcare providers aiming to increase vaccine uptake for H1N1 and seasonal flu to reduce the spread and impact of these illnesses.
- Business Problem
  - Predict whether individuals are likely to get vaccinated for H1N1 and seasonal flu based on demographic, social, and behavioral factors. Insights from the model can guide targeted vaccination campaigns and policy decisions.

Data understanding

- Dataset overview:
  - Two target variables: H1N1_vaccine and seasonal_vaccine (binary: 1 for vaccinated, 0 otherwise).
  - Predictors include demographic (age, gender), social (education, marital status), and behavioral (health conditions, vaccine awareness) features.
- Objective:
  - This is a classification problem, where the task is to predict binary outcomes (vaccinated or not).

```python
# importing libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

```
df = pd.read_csv("C:/Users/DELL/Desktop/Projects/FLU
VACCINES/Data/H1N1_Flu_Vaccines.csv")

df.head()

   respondent_id  h1n1_concern  h1n1_knowledge
behavioral_antiviral_meds  \
0              0           1.0             0.0
0.0
1              1           3.0             2.0
0.0
2              2           1.0             1.0
0.0
3              3           1.0             1.0
0.0
4              4           2.0             1.0
0.0

   behavioral_avoidance  behavioral_face_mask
behavioral_wash_hands  \
0                   0.0                   0.0                    0.0

1                   1.0                   0.0                    1.0

2                   1.0                   0.0                    0.0

3                   1.0                   0.0                    1.0

4                   1.0                   0.0                    1.0


   behavioral_large_gatherings  behavioral_outside_home  \
0                          0.0                      1.0
1                          0.0                      1.0
2                          0.0                      0.0
3                          1.0                      0.0
4                          1.0                      0.0

   behavioral_touch_face  ...  rent_or_own   employment_status  \
0                    1.0  ...          Own  Not in Labor Force
1                    1.0  ...         Rent            Employed
2                    0.0  ...          Own            Employed
3                    0.0  ...         Rent  Not in Labor Force
4                    1.0  ...          Own            Employed

   hhs_geo_region                census_msa  household_adults  \
0        oxchjgsf                   Non-MSA               0.0
1        bhuqouqj  MSA, Not Principle  City               0.0
2        qufhixun  MSA, Not Principle  City               2.0
3        lrircsnp       MSA, Principle City               0.0
4        qufhixun  MSA, Not Principle  City               1.0
```

```
   household_children  employment_industry  employment_occupation  \
0                 0.0                  NaN                    NaN   
1                 0.0             pxcmvdjn               xgwztkwe   
2                 0.0             rucpziij               xtkaffoo   
3                 0.0                  NaN                    NaN   
4                 0.0             wxleyezf               emcorrxb   

   h1n1_vaccine  seasonal_vaccine  
0             0                 0  
1             0                 1  
2             0                 0  
3             0                 1  
4             0                 0  

[5 rows x 38 columns]
```

Descriptive Statistics

```
df.shape
```

```
(26707, 38)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26707 entries, 0 to 26706
Data columns (total 38 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   respondent_id                26707 non-null  int64  
 1   h1n1_concern                 26615 non-null  float64
 2   h1n1_knowledge               26591 non-null  float64
 3   behavioral_antiviral_meds    26636 non-null  float64
 4   behavioral_avoidance         26499 non-null  float64
 5   behavioral_face_mask         26688 non-null  float64
 6   behavioral_wash_hands        26665 non-null  float64
 7   behavioral_large_gatherings  26620 non-null  float64
 8   behavioral_outside_home      26625 non-null  float64
 9   behavioral_touch_face        26579 non-null  float64
 10  doctor_recc_h1n1             24547 non-null  float64
 11  doctor_recc_seasonal         24547 non-null  float64
 12  chronic_med_condition        25736 non-null  float64
 13  child_under_6_months         25887 non-null  float64
 14  health_worker                25903 non-null  float64
 15  health_insurance             14433 non-null  float64
 16  opinion_h1n1_vacc_effective  26316 non-null  float64
 17  opinion_h1n1_risk            26319 non-null  float64
```

```
 18  opinion_h1n1_sick_from_vacc  26312 non-null  float64
 19  opinion_seas_vacc_effective  26245 non-null  float64
 20  opinion_seas_risk            26193 non-null  float64
 21  opinion_seas_sick_from_vacc  26170 non-null  float64
 22  age_group                    26707 non-null  object
 23  education                    25300 non-null  object
 24  race                         26707 non-null  object
 25  sex                          26707 non-null  object
 26  income_poverty               22284 non-null  object
 27  marital_status               25299 non-null  object
 28  rent_or_own                  24665 non-null  object
 29  employment_status            25244 non-null  object
 30  hhs_geo_region               26707 non-null  object
 31  census_msa                   26707 non-null  object
 32  household_adults             26458 non-null  float64
 33  household_children           26458 non-null  float64
 34  employment_industry          13377 non-null  object
 35  employment_occupation        13237 non-null  object
 36  h1n1_vaccine                 26707 non-null  int64
 37  seasonal_vaccine             26707 non-null  int64
dtypes: float64(23), int64(3), object(12)
memory usage: 7.7+ MB
```

```
df.describe()
```

```
       respondent_id  h1n1_concern  h1n1_knowledge  
behavioral_antiviral_meds  \
count   26707.000000  26615.000000    26591.000000  
26636.000000   
mean    13353.000000      1.618486        1.262532  
0.048844   
std      7709.791156      0.910311        0.618149  
0.215545   
min         0.000000      0.000000        0.000000  
0.000000   
25%      6676.500000      1.000000        1.000000  
0.000000   
50%     13353.000000      2.000000        1.000000  
0.000000   
75%     20029.500000      2.000000        2.000000  
0.000000   
max     26706.000000      3.000000        2.000000  
1.000000   

       behavioral_avoidance  behavioral_face_mask  
behavioral_wash_hands  \
count          26499.000000          26688.000000  
26665.000000   
mean               0.725612              0.068982  
0.825614   
```

```
std                   0.446214               0.253429
0.379448
min                   0.000000               0.000000
0.000000
25%                   0.000000               0.000000
1.000000
50%                   1.000000               0.000000
1.000000
75%                   1.000000               0.000000
1.000000
max                   1.000000               1.000000
1.000000

       behavioral_large_gatherings  behavioral_outside_home  \
count                  26620.00000             26625.000000
mean                       0.35864                 0.337315
std                        0.47961                 0.472802
min                        0.00000                 0.000000
25%                        0.00000                 0.000000
50%                        0.00000                 0.000000
75%                        1.00000                 1.000000
max                        1.00000                 1.000000

       behavioral_touch_face  ...  opinion_h1n1_vacc_effective  \
count           26579.000000  ...                 26316.000000
mean                0.677264  ...                     3.850623
std                 0.467531  ...                     1.007436
min                 0.000000  ...                     1.000000
25%                 0.000000  ...                     3.000000
50%                 1.000000  ...                     4.000000
75%                 1.000000  ...                     5.000000
max                 1.000000  ...                     5.000000

       opinion_h1n1_risk  opinion_h1n1_sick_from_vacc  \
count       26319.000000                 26312.000000
mean            2.342566                     2.357670
std             1.285539                     1.362766
min             1.000000                     1.000000
25%             1.000000                     1.000000
50%             2.000000                     2.000000
75%             4.000000                     4.000000
max             5.000000                     5.000000

       opinion_seas_vacc_effective  opinion_seas_risk  \
count                 26245.000000       26193.000000
mean                      4.025986           2.719162
std                       1.086565           1.385055
min                       1.000000           1.000000
25%                       4.000000           2.000000
50%                       4.000000           2.000000
```

```
75%                             5.000000          4.000000
max                             5.000000          5.000000

       opinion_seas_sick_from_vacc  household_adults  
household_children  \
count                   26170.000000      26458.000000   
26458.000000   
mean                        2.118112          0.886499   
0.534583   
std                         1.332950          0.753422   
0.928173   
min                         1.000000          0.000000   
0.000000   
25%                         1.000000          0.000000   
0.000000   
50%                         2.000000          1.000000   
0.000000   
75%                         4.000000          1.000000   
1.000000   
max                         5.000000          3.000000   
3.000000   

       h1n1_vaccine  seasonal_vaccine  
count  26707.000000      26707.000000  
mean       0.212454          0.465608  
std        0.409052          0.498825  
min        0.000000          0.000000  
25%        0.000000          0.000000  
50%        0.000000          0.000000  
75%        0.000000          1.000000  
max        1.000000          1.000000  

[8 rows x 26 columns]
```

Exploratory Data Analysis

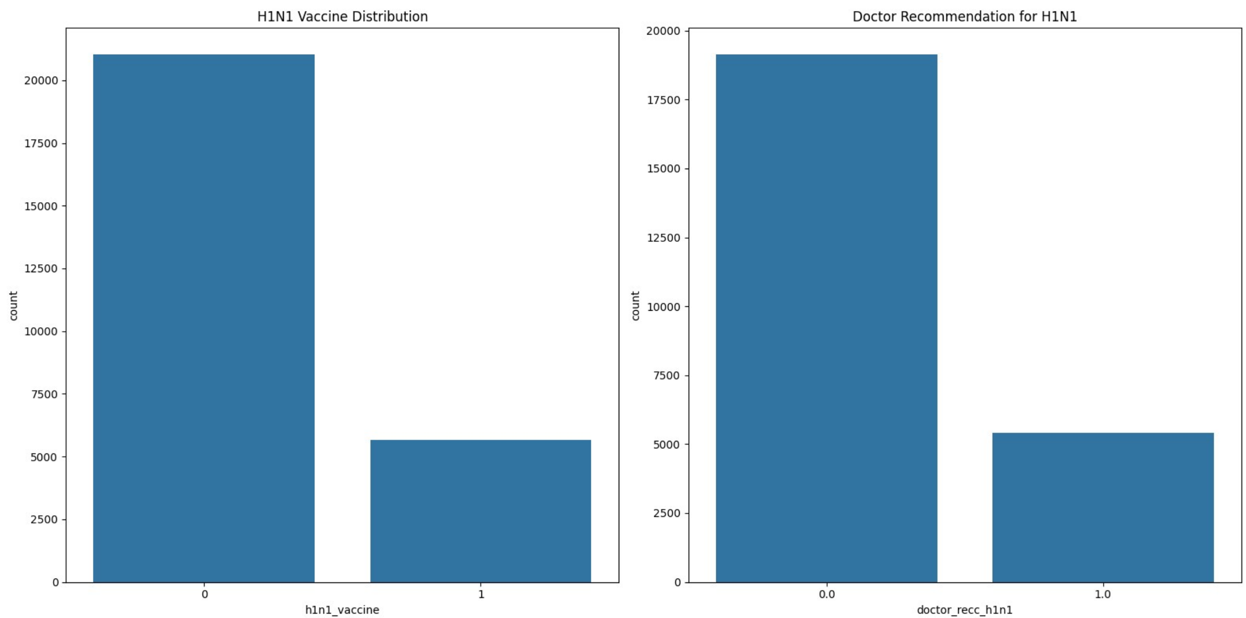
- Steps:
  - Load and inspect data:
    - check for missing values
    - understand the distribution of target variables and predictors.
  - Visualize relationships
  - Handle missing values
  - Feature engineering:
    - Encode categorical variables
    - Scale numeric features
  - check class imbalance

```
df.isnull().sum()
```

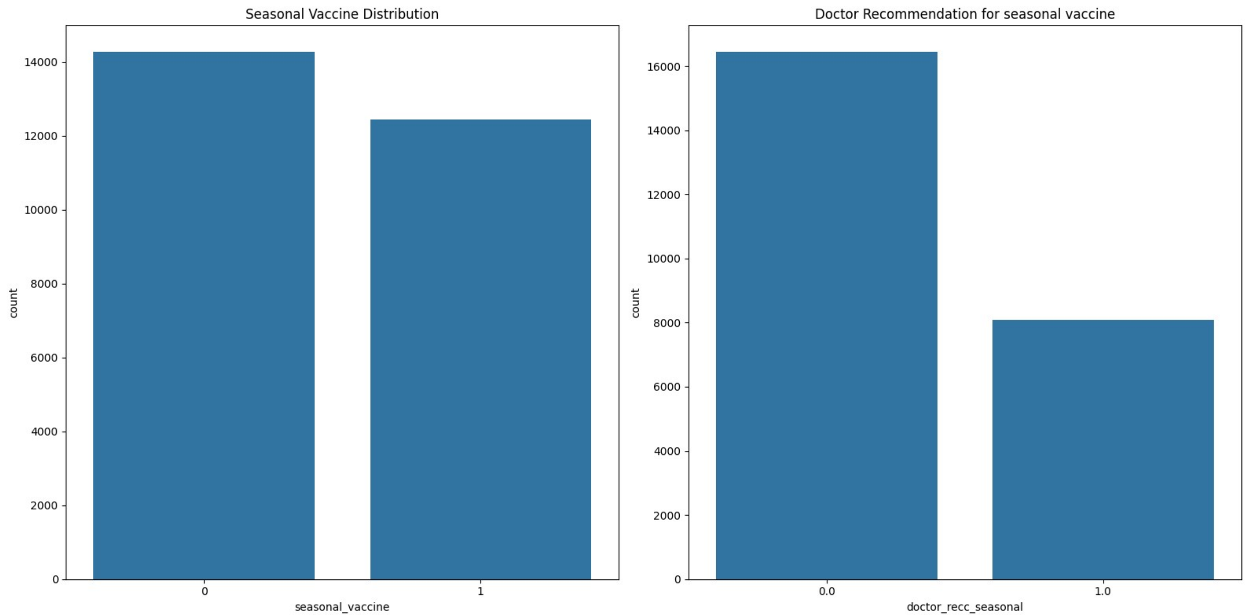
```
respondent_id                      0
h1n1_concern                      92
h1n1_knowledge                   116
behavioral_antiviral_meds         71
behavioral_avoidance             208
behavioral_face_mask              19
behavioral_wash_hands             42
behavioral_large_gatherings       87
behavioral_outside_home           82
behavioral_touch_face            128
doctor_recc_h1n1                2160
doctor_recc_seasonal            2160
chronic_med_condition            971
child_under_6_months             820
health_worker                    804
health_insurance               12274
opinion_h1n1_vacc_effective      391
opinion_h1n1_risk                388
opinion_h1n1_sick_from_vacc      395
opinion_seas_vacc_effective      462
opinion_seas_risk                514
opinion_seas_sick_from_vacc      537
age_group                          0
education                       1407
race                               0
sex                                0
income_poverty                  4423
marital_status                  1408
rent_or_own                     2042
employment_status               1463
hhs_geo_region                     0
census_msa                         0
household_adults                 249
household_children               249
employment_industry            13330
employment_occupation          13470
h1n1_vaccine                       0
seasonal_vaccine                   0
dtype: int64
```

```
# Create a figure with two subplots side by side
fig, axes = plt.subplots(1, 2, figsize=(16, 8))

# First plot: h1n1_vaccine
sns.countplot(x="h1n1_vaccine", data=df, ax=axes[0])
axes[0].set_title("H1N1 Vaccine Distribution")

# Second plot: doctor_recc_h1n1
```

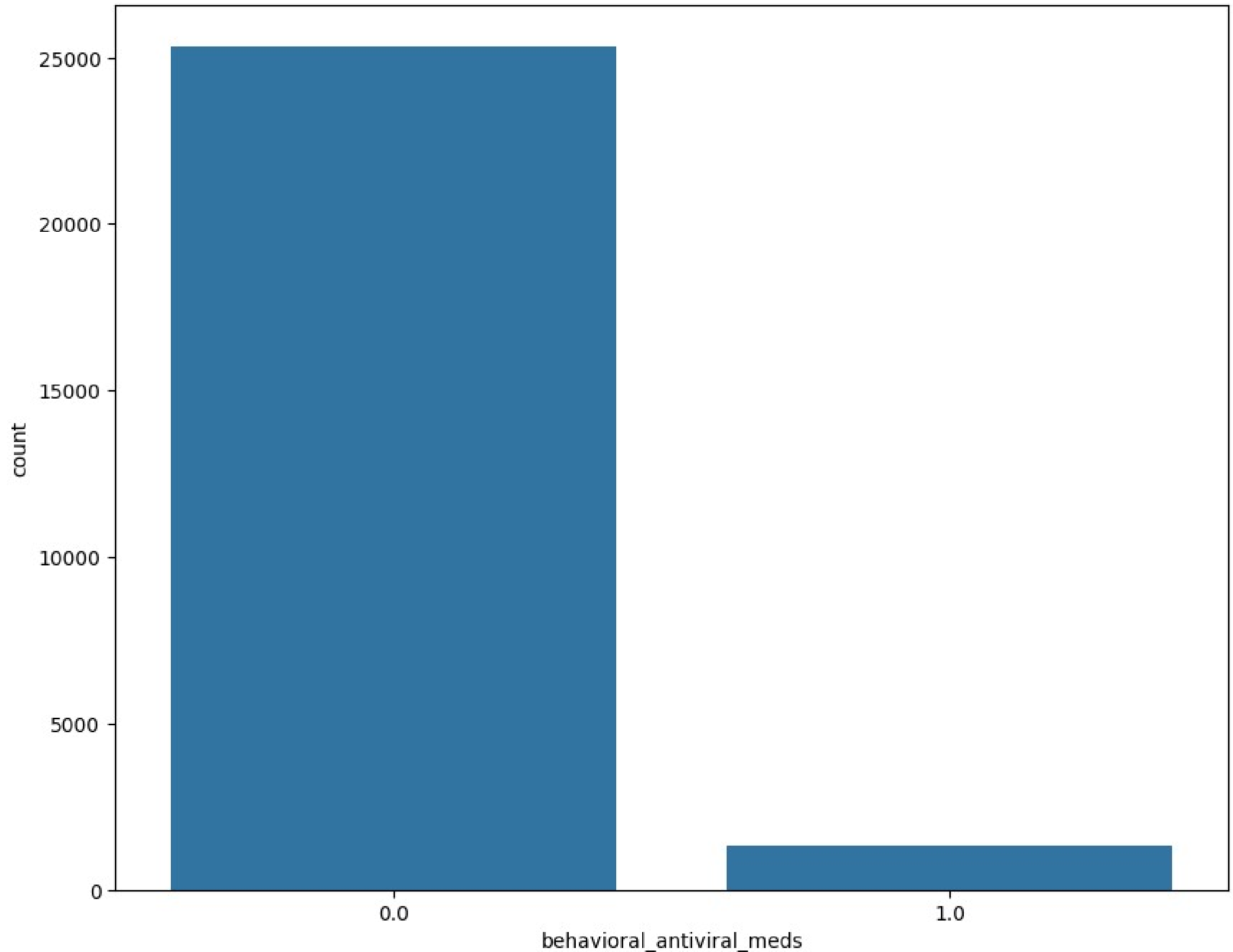
```
sns.countplot(x="doctor_recc_h1n1", data=df, ax=axes[1])
axes[1].set_title("Doctor Recommendation for H1N1")

# Display the plots
plt.tight_layout()
plt.show()
```




```
# Create a figure with two subplots side by side
fig, axes = plt.subplots(1, 2, figsize=(16, 8))

# First plot: seasonal_vaccine
sns.countplot(x="seasonal_vaccine", data=df, ax=axes[0])
axes[0].set_title("Seasonal Vaccine Distribution")

# Second plot: doctor_recc_seasonal
sns.countplot(x="doctor_recc_seasonal", data=df, ax=axes[1])
axes[1].set_title("Doctor Recommendation for seasonal vaccine")

# Display the plots
plt.tight_layout()
plt.show()
```




```
plt.figure(figsize=(10,8))
sns.countplot(x="behavioral_antiviral_meds",data=df)
plt.show()
```



Dropping data with more than 30% missing data

```
# Calculate the percentage of null values per column
null_percentage = (df.isnull().sum() / len(df)) * 100

# Identify columns with more than 50% null values
columns_to_drop = null_percentage[null_percentage > 30].index

# Drop those columns from the DataFrame
df = df.drop(columns=columns_to_drop)

(df.isnull().sum() / len(df) * 100).sort_values(ascending=False)
```

income_poverty	16.561201
doctor_recc_h1n1	8.087767
doctor_recc_seasonal	8.087767
rent_or_own	7.645936

```
employment_status              5.477965
marital_status                 5.272026
education                      5.268282
chronic_med_condition          3.635751
child_under_6_months           3.070356
health_worker                  3.010447
opinion_seas_sick_from_vacc    2.010709
opinion_seas_risk              1.924589
opinion_seas_vacc_effective    1.729884
opinion_h1n1_sick_from_vacc    1.479013
opinion_h1n1_vacc_effective    1.464036
opinion_h1n1_risk              1.452803
household_adults               0.932340
household_children             0.932340
behavioral_avoidance           0.778822
behavioral_touch_face          0.479275
h1n1_knowledge                 0.434343
h1n1_concern                   0.344479
behavioral_large_gatherings    0.325757
behavioral_outside_home        0.307036
behavioral_antiviral_meds      0.265848
behavioral_wash_hands          0.157262
behavioral_face_mask           0.071142
respondent_id                  0.000000
race                           0.000000
age_group                      0.000000
sex                            0.000000
census_msa                     0.000000
hhs_geo_region                 0.000000
h1n1_vaccine                   0.000000
seasonal_vaccine               0.000000
dtype: float64
```

Mode Imputation for categorical variables

```python
from sklearn.impute import SimpleImputer

# Separate categorical and numerical columns
categorical_cols = df.select_dtypes(include=['object']).columns

# Imputer for categorical columns (mode)
mode_imputer = SimpleImputer(strategy='most_frequent')
df[categorical_cols] =
mode_imputer.fit_transform(df[categorical_cols])
```

Median imputer for numerical data

```
from sklearn.impute import SimpleImputer

# Separate categorical and numerical columns
numerical_cols = df.select_dtypes(include=['int64',
'float64']).columns

# Imputer for numerical columns (median)
median_imputer = SimpleImputer(strategy='median')
df[numerical_cols] = median_imputer.fit_transform(df[numerical_cols])

# Check if there are any remaining missing values
remaining_missing = df.isnull().sum()
remaining_missing[remaining_missing > 0]
```

```
Series([], dtype: int64)
```

```
df.isna().sum()
```

```
respondent_id                  0
h1n1_concern                   0
h1n1_knowledge                 0
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_h1n1               0
doctor_recc_seasonal           0
chronic_med_condition          0
child_under_6_months           0
health_worker                  0
opinion_h1n1_vacc_effective    0
opinion_h1n1_risk              0
opinion_h1n1_sick_from_vacc    0
opinion_seas_vacc_effective    0
opinion_seas_risk              0
opinion_seas_sick_from_vacc    0
age_group                      0
education                      0
race                           0
sex                            0
income_poverty                 0
marital_status                 0
rent_or_own                    0
employment_status              0
hhs_geo_region                 0
```

```
census_msa                              0
household_adults                        0
household_children                      0
h1n1_vaccine                            0
seasonal_vaccine                        0
dtype: int64
```

EDA after cleaning data

```
df.shape
```

```
(26707, 35)
```

```
df.describe()
```

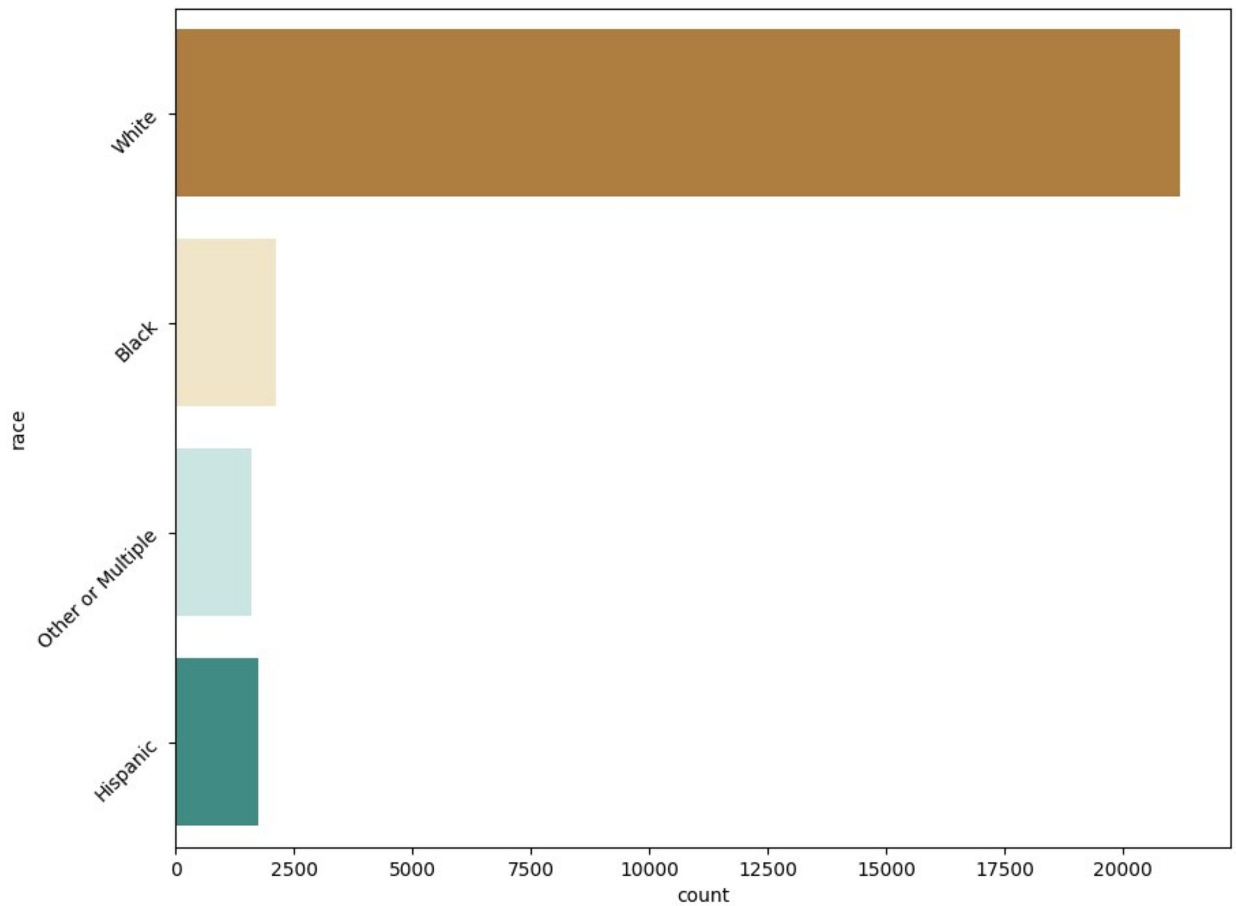
```
       respondent_id  h1n1_concern  h1n1_knowledge  
behavioral_antiviral_meds  \
count   26707.000000  26707.000000    26707.000000  
26707.000000   
mean    13353.000000      1.619800        1.261392  
0.048714   
std      7709.791156      0.909016        0.617047  
0.215273   
min         0.000000      0.000000        0.000000  
0.000000   
25%      6676.500000      1.000000        1.000000  
0.000000   
50%     13353.000000      2.000000        1.000000  
0.000000   
75%     20029.500000      2.000000        2.000000  
0.000000   
max     26706.000000      3.000000        2.000000  
1.000000   

       behavioral_avoidance  behavioral_face_mask  
behavioral_wash_hands  \
count          26707.000000          26707.000000  
26707.000000   
mean               0.727749              0.068933  
0.825888   
std                0.445127              0.253345  
0.379213   
min                0.000000              0.000000  
0.000000   
25%                0.000000              0.000000  
1.000000   
50%                1.000000              0.000000  
1.000000   
```

```
75%                  1.000000              0.000000
1.000000
max                  1.000000              1.000000
1.000000

       behavioral_large_gatherings  behavioral_outside_home  \
count                 26707.000000             26707.000000
mean                      0.357472                 0.336279
std                       0.479264                 0.472444
min                       0.000000                 0.000000
25%                       0.000000                 0.000000
50%                       0.000000                 0.000000
75%                       1.000000                 1.000000
max                       1.000000                 1.000000

       behavioral_touch_face  ...  opinion_h1n1_vacc_effective  \
count           26707.000000  ...                 26707.000000
mean                0.678811  ...                     3.852810
std                 0.466942  ...                     1.000195
min                 0.000000  ...                     1.000000
25%                 0.000000  ...                     3.000000
50%                 1.000000  ...                     4.000000
75%                 1.000000  ...                     5.000000
max                 1.000000  ...                     5.000000

       opinion_h1n1_risk  opinion_h1n1_sick_from_vacc  \
count       26707.000000                 26707.000000
mean            2.337589                     2.352380
std             1.276825                     1.353339
min             1.000000                     1.000000
25%             1.000000                     1.000000
50%             2.000000                     2.000000
75%             4.000000                     4.000000
max             5.000000                     5.000000

       opinion_seas_vacc_effective  opinion_seas_risk  \
count                 26707.000000       26707.000000
mean                      4.025536           2.705321
std                       1.077131           1.375216
min                       1.000000           1.000000
25%                       4.000000           2.000000
50%                       4.000000           2.000000
75%                       5.000000           4.000000
max                       5.000000           5.000000

       opinion_seas_sick_from_vacc  household_adults
household_children  \
count                 26707.000000      26707.000000
26707.000000
mean                      2.115737          0.887558
```

```
0.529599
std                          1.319585          0.749980
0.925264
min                          1.000000          0.000000
0.000000
25%                          1.000000          0.000000
0.000000
50%                          2.000000          1.000000
0.000000
75%                          2.000000          1.000000
1.000000
max                          5.000000          3.000000
3.000000

       h1n1_vaccine  seasonal_vaccine
count  26707.000000      26707.000000
mean       0.212454          0.465608
std        0.409052          0.498825
min        0.000000          0.000000
25%        0.000000          0.000000
50%        0.000000          0.000000
75%        0.000000          1.000000
max        1.000000          1.000000

[8 rows x 25 columns]
```

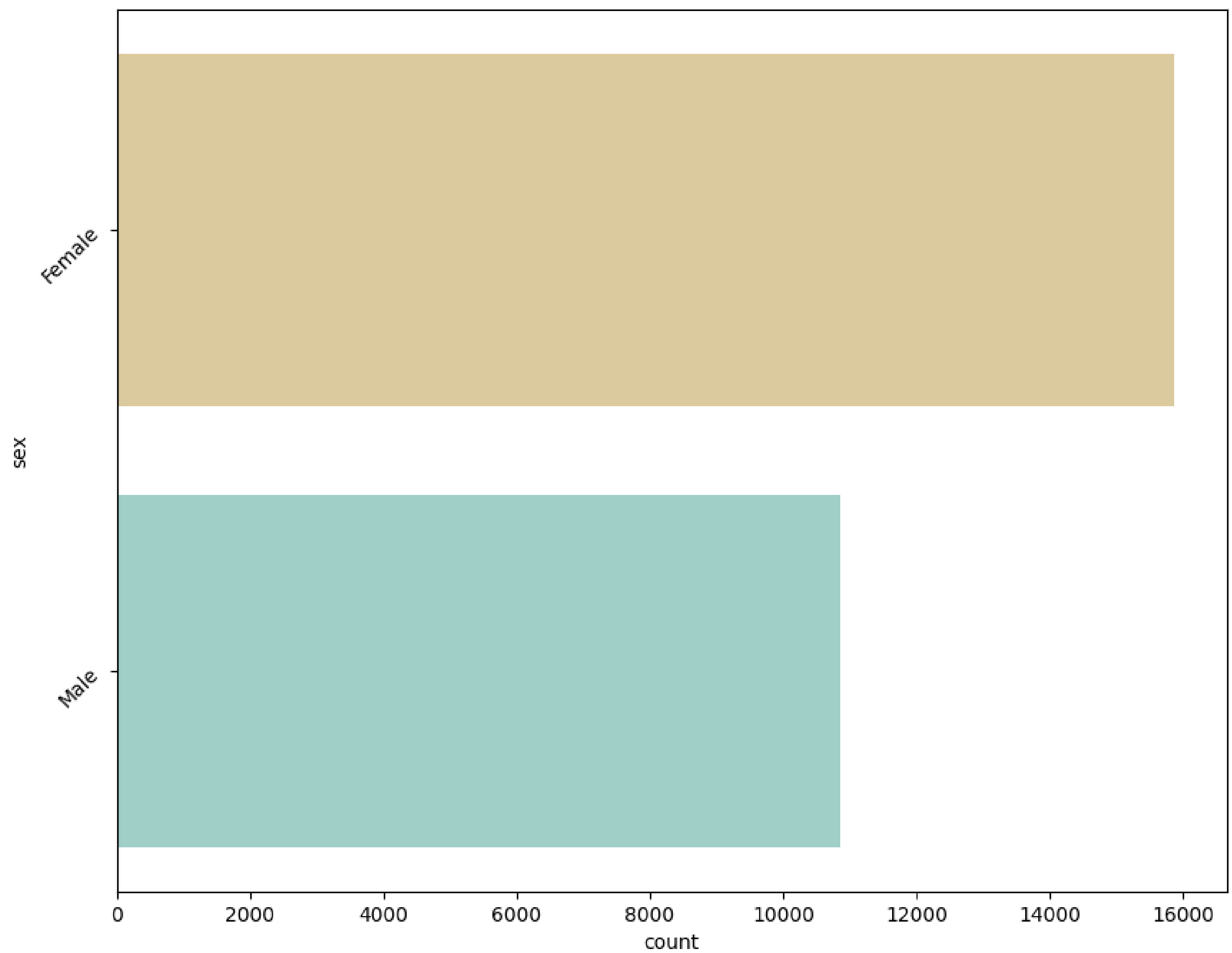
```
plt.figure(figsize=(10,8))
sns.countplot(y="race",data=df,palette="BrBG")
plt.yticks(rotation=45)
plt.show()
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_22004\584491901.py:2:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.countplot(y="race",data=df,palette="BrBG")
```




```
plt.figure(figsize=(10,8))
sns.countplot(y="sex",data=df,palette="BrBG")
plt.yticks(rotation=45)
plt.show()
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_22004\2993517352.py:2:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.countplot(y="sex",data=df,palette="BrBG")
```




```
df["income_poverty"].value_counts()
```

```
income_poverty
<= $75,000, Above Poverty    17200
> $75,000                     6810
Below Poverty                 2697
Name: count, dtype: int64
```

```
df["marital_status"].value_counts()
```

```
marital_status
Married        14963
Not Married    11744
Name: count, dtype: int64
```

```
df["rent_or_own"].value_counts()
```

```
rent_or_own
Own     20778
Rent     5929
Name: count, dtype: int64
```

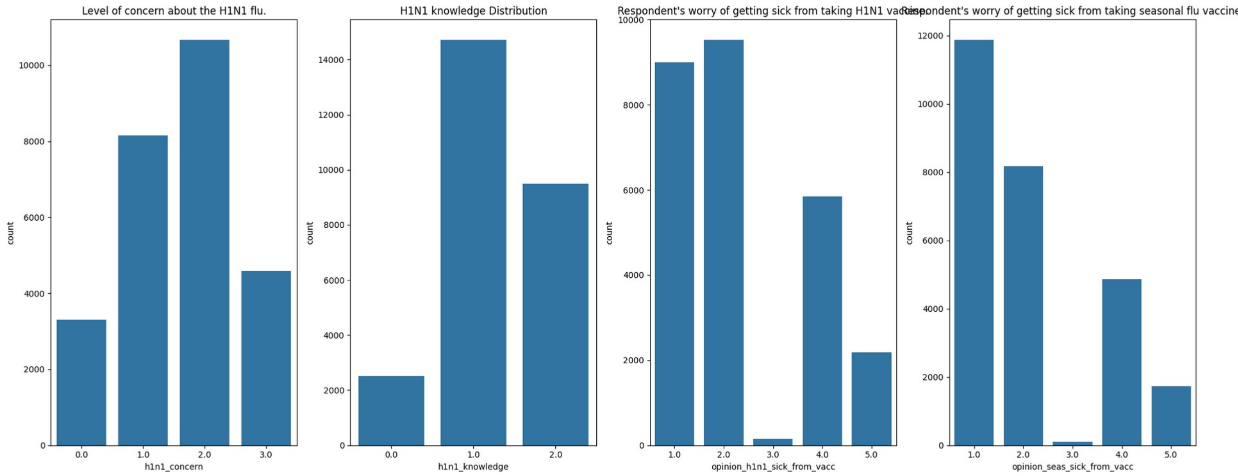
```
df["employment_status"].value_counts()
```

```
employment_status
Employed              15023
Not in Labor Force    10231
Unemployed             1453
Name: count, dtype: int64
```

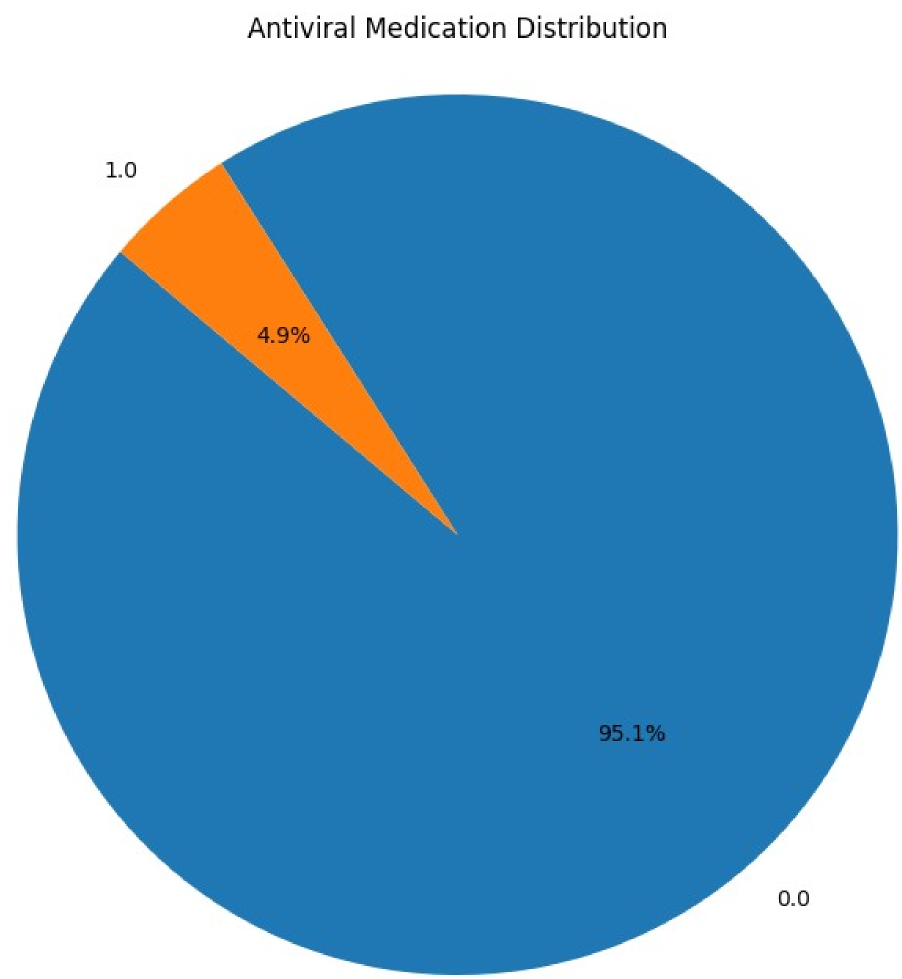
```
df["census_msa"].value_counts()
```

```
census_msa
MSA, Not Principle  City    11645
MSA, Principle City          7864
Non-MSA                      7198
Name: count, dtype: int64
```

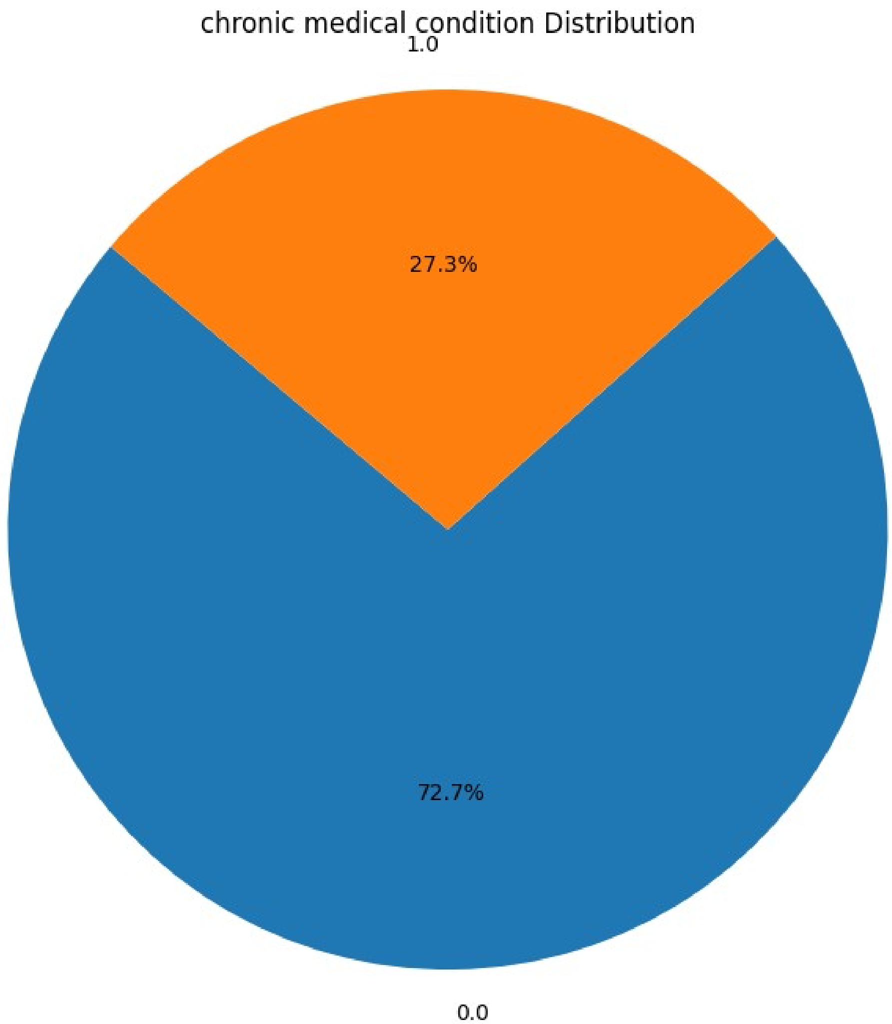
```
# Create a figure with two subplots side by side
fig, axes = plt.subplots(1, 4, figsize=(20, 8))

# First plot: h1n1_concern
sns.countplot(x="h1n1_concern", data=df, ax=axes[0])
axes[0].set_title("Level of concern about the H1N1 flu.")

# Second plot: h1n1_vaccine
sns.countplot(x="h1n1_knowledge", data=df, ax=axes[1])
axes[1].set_title("H1N1 knowledge Distribution")

# Third plot: doctor_recc_h1n1
sns.countplot(x="opinion_h1n1_sick_from_vacc", data=df, ax=axes[2])
axes[2].set_title("Respondent's worry of getting sick from taking H1N1
vaccine.")

# Third plot: doctor_recc_h1n1
sns.countplot(x="opinion_seas_sick_from_vacc", data=df, ax=axes[3])
axes[3].set_title("Respondent's worry of getting sick from taking
seasonal flu vaccine.")

# Display the plots
plt.tight_layout()
plt.show()
```



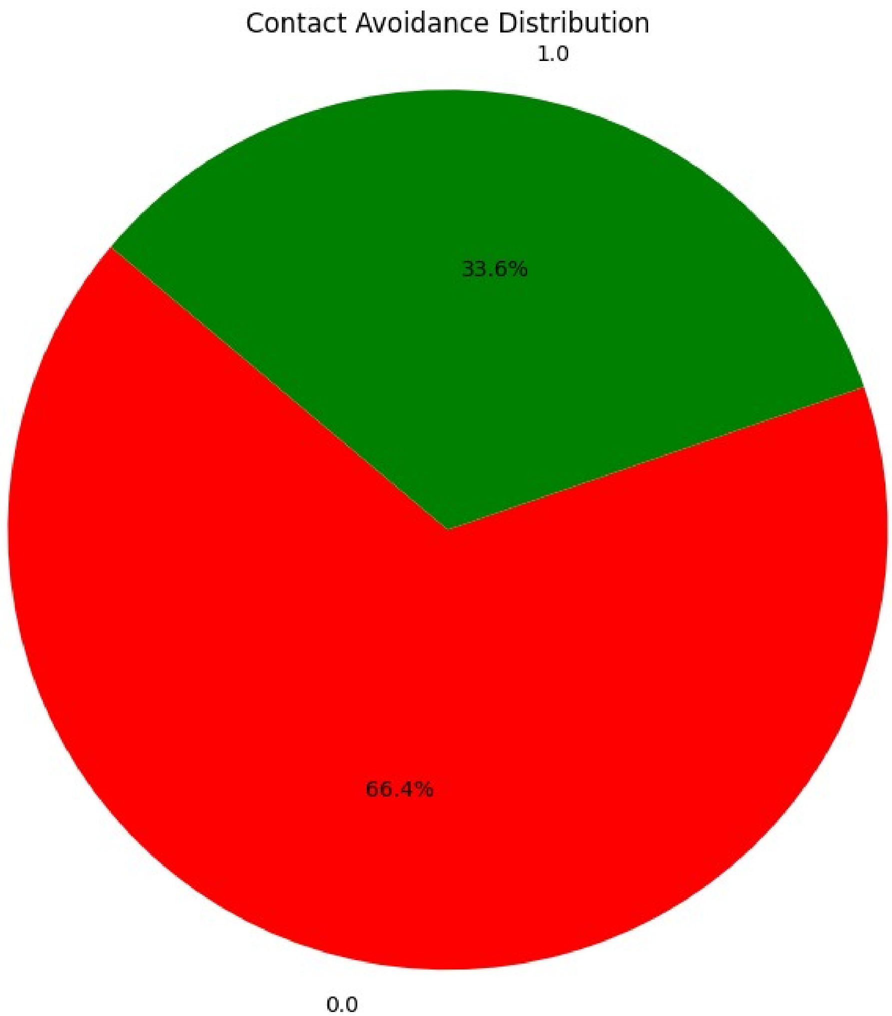

```
value_counts = df["behavioral_antiviral_meds"].value_counts()

plt.figure(figsize=(10,8))
plt.pie(value_counts, labels=value_counts.index, autopct='%1.1f%%',
startangle=140)
plt.title('Antiviral Medication Distribution')
plt.axis('equal')
plt.show()
```



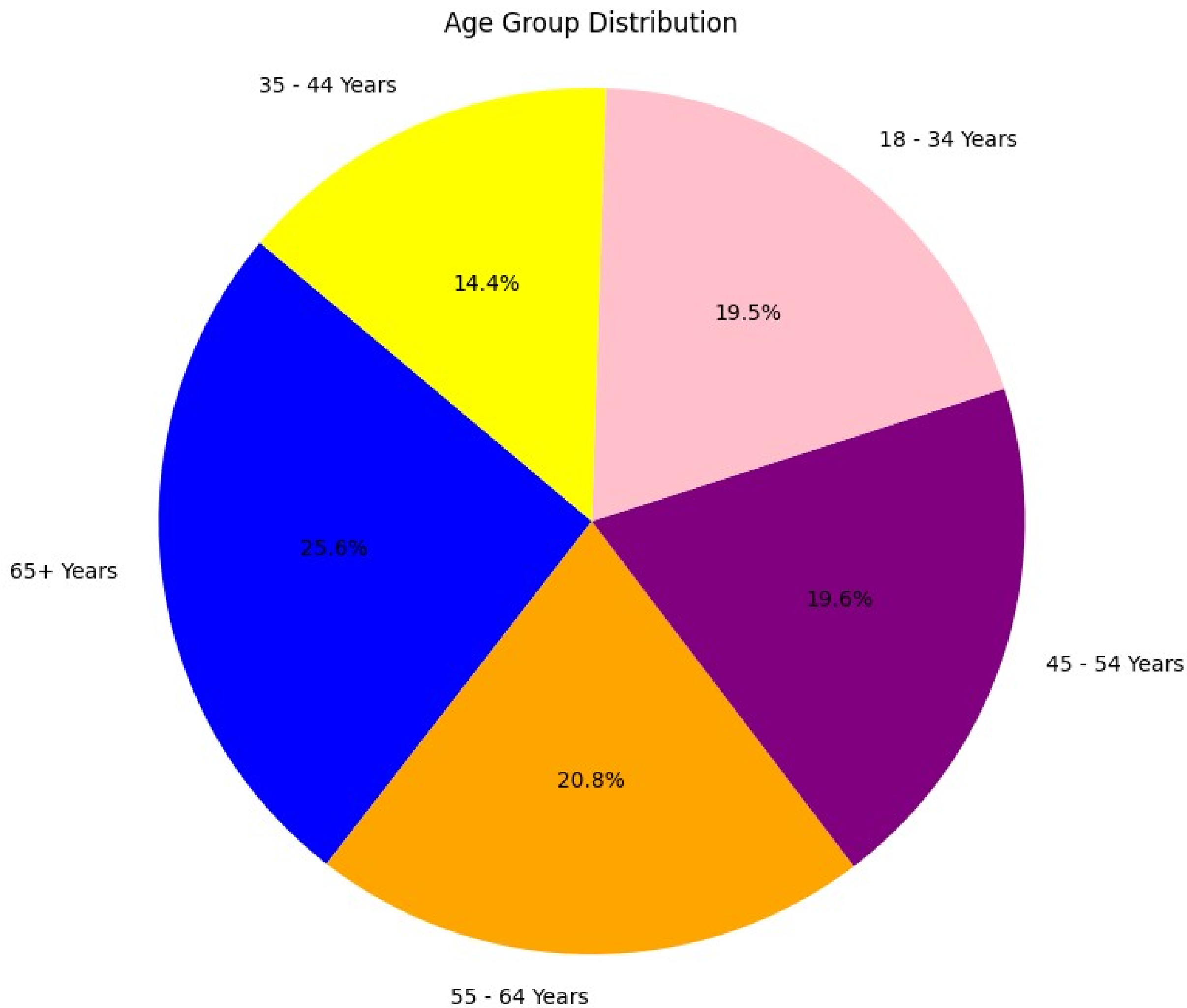

```
value_counts = df["chronic_med_condition"].value_counts()

plt.figure(figsize=(10,8))
plt.pie(value_counts, labels=value_counts.index, autopct='%1.1f%%',
startangle=140)
plt.title('chronic medical condition Distribution')
plt.axis('equal')
plt.show()
```



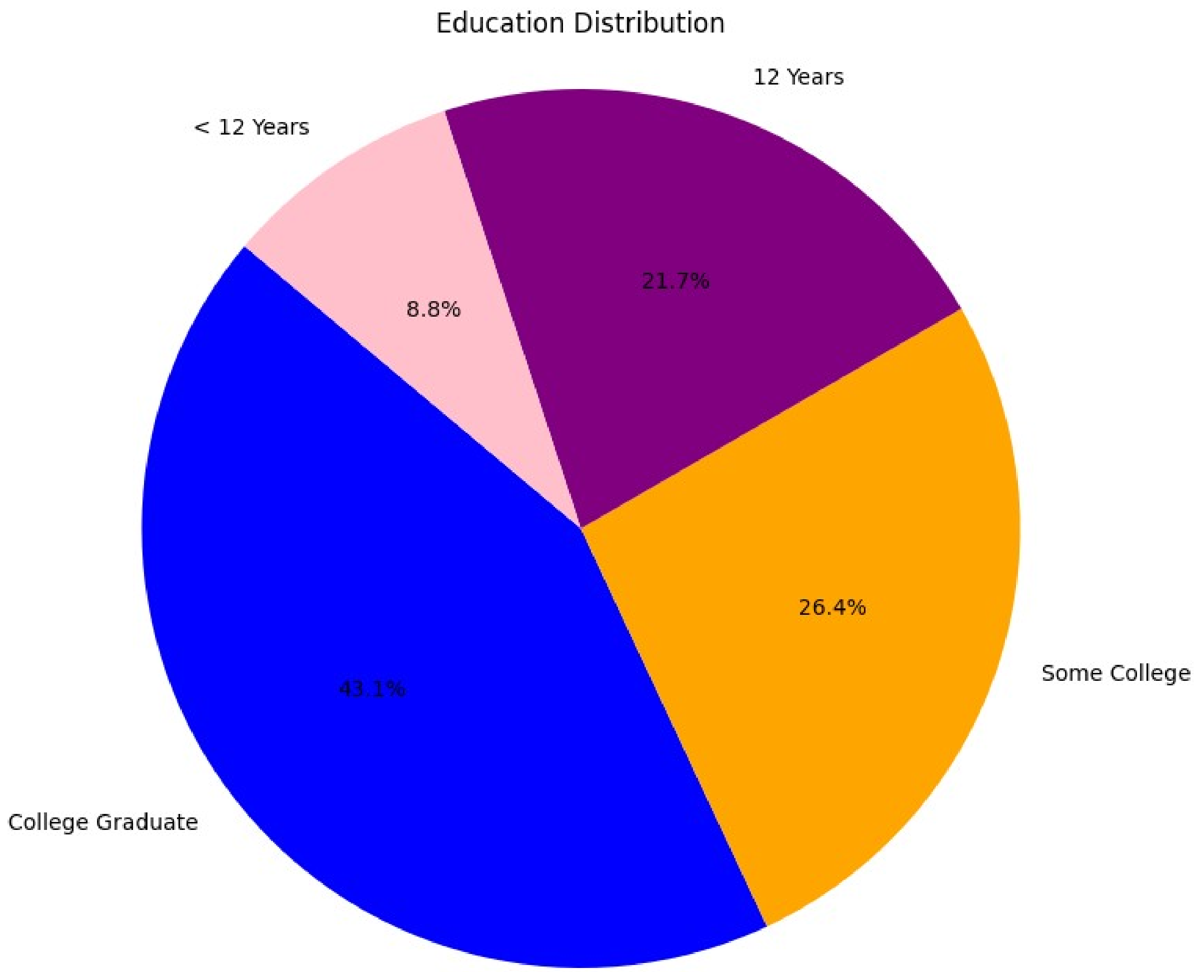

```
contact_avoidance_counts =
df["behavioral_outside_home"].value_counts()

plt.figure(figsize=(10,8))
color=["red","green"]
plt.pie(contact_avoidance_counts,
labels=contact_avoidance_counts.index, autopct='%1.1f%%',
startangle=140,colors=color)
plt.title('Contact Avoidance Distribution')
plt.axis('equal')
plt.show()
```




```
age_group_counts = df["age_group"].value_counts()

plt.figure(figsize=(10,8))
color=["blue","orange", 'purple', 'pink', 'yellow']
plt.pie(age_group_counts, labels=age_group_counts.index,
autopct='%1.1f%%', startangle=140,colors=color)
plt.title('Age Group Distribution')
plt.axis('equal')
plt.show()
```




```
education = df["education"].value_counts()

plt.figure(figsize=(10,8))
color=["blue","orange", 'purple', 'pink', 'yellow']
plt.pie(education, labels=education.index, autopct='%1.1f%%',
startangle=140,colors=color)
plt.title('Education Distribution')
plt.axis('equal')
plt.show()
```




```
numerical_cols

Index(['respondent_id', 'h1n1_concern', 'h1n1_knowledge',
       'behavioral_antiviral_meds', 'behavioral_avoidance',
       'behavioral_face_mask', 'behavioral_wash_hands',
       'behavioral_large_gatherings', 'behavioral_outside_home',
       'behavioral_touch_face', 'doctor_recc_h1n1',
'doctor_recc_seasonal',
       'chronic_med_condition', 'child_under_6_months',
'health_worker',
       'opinion_h1n1_vacc_effective', 'opinion_h1n1_risk',
       'opinion_h1n1_sick_from_vacc', 'opinion_seas_vacc_effective',
       'opinion_seas_risk', 'opinion_seas_sick_from_vacc',
'household_adults',
       'household_children', 'h1n1_vaccine', 'seasonal_vaccine'],
      dtype='object')

categorical_cols
```

```
Index(['age_group', 'education', 'race', 'sex', 'income_poverty',
       'marital_status', 'rent_or_own', 'employment_status',
'hhs_geo_region',
       'census_msa'],
      dtype='object')
```

Data preprocessing

- Feature Engineering

```
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
encoder = LabelEncoder()
df_categorical = df.select_dtypes(include=['object'])
df_categorical_preprocessed =
pd.DataFrame(df_categorical.apply(encoder.fit_transform))
df_categorical_preprocessed
```

```
       age_group  education  race  sex  income_poverty  marital_status
\
0              3          1     3    0               2               1

1              1          0     3    1               2               1

2              0          2     3    1               0               1

3              4          0     3    0               2               1

4              2          3     3    0               0               0

...          ...        ...   ...  ...             ...             ...

26702          4          3     3    0               0               1

26703          0          2     3    1               0               1

26704          3          3     3    0               0               1

26705          0          3     1    0               0               0

26706          4          3     3    1               0               0


       rent_or_own  employment_status  hhs_geo_region  census_msa
0                0                  1               8           2
1                1                  0               1           0
2                0                  0               9           0
3                1                  1               5           1
4                0                  0               9           0
...            ...                ...             ...         ...
26702            0                  1               9           2
```

```
26703              1                  0               6           1
26704              0                  0               6           0
26705              1                  0               5           2
26706              0                  1               7           1

[26707 rows x 10 columns]

df_numerical = pd.DataFrame(df.select_dtypes(exclude=['object']))
df_numerical

       respondent_id  h1n1_concern  h1n1_knowledge
behavioral_antiviral_meds  \
0                0.0           1.0             0.0
0.0
1                1.0           3.0             2.0
0.0
2                2.0           1.0             1.0
0.0
3                3.0           1.0             1.0
0.0
4                4.0           2.0             1.0
0.0
...              ...           ...             ...
...
26702        26702.0           2.0             0.0
0.0
26703        26703.0           1.0             2.0
0.0
26704        26704.0           2.0             2.0
0.0
26705        26705.0           1.0             1.0
0.0
26706        26706.0           0.0             0.0
0.0

       behavioral_avoidance  behavioral_face_mask
behavioral_wash_hands  \
0                       0.0                   0.0
0.0
1                       1.0                   0.0
1.0
2                       1.0                   0.0
0.0
3                       1.0                   0.0
1.0
4                       1.0                   0.0
1.0
...                     ...                   ...                      .
..
26702                   1.0                   0.0
```

```
0.0
26703                 1.0                    0.0
1.0
26704                 1.0                    1.0
1.0
26705                 0.0                    0.0
0.0
26706                 1.0                    0.0
0.0

       behavioral_large_gatherings  behavioral_outside_home  \
0                              0.0                      1.0
1                              0.0                      1.0
2                              0.0                      0.0
3                              1.0                      0.0
4                              1.0                      0.0
...                            ...                      ...
26702                          0.0                      1.0
26703                          0.0                      0.0
26704                          1.0                      0.0
26705                          0.0                      0.0
26706                          0.0                      0.0

       behavioral_touch_face  ...  opinion_h1n1_vacc_effective  \
0                        1.0  ...                          3.0
1                        1.0  ...                          5.0
2                        0.0  ...                          3.0
3                        0.0  ...                          3.0
4                        1.0  ...                          3.0
...                      ...  ...                          ...
26702                    0.0  ...                          3.0
26703                    0.0  ...                          4.0
26704                    1.0  ...                          4.0
26705                    1.0  ...                          3.0
26706                    0.0  ...                          5.0

       opinion_h1n1_risk  opinion_h1n1_sick_from_vacc  \
0                    1.0                          2.0
1                    4.0                          4.0
2                    1.0                          1.0
3                    3.0                          5.0
4                    3.0                          2.0
...                  ...                          ...
26702                1.0                          1.0
26703                2.0                          2.0
26704                4.0                          2.0
26705                1.0                          2.0
26706                1.0                          1.0

       opinion_seas_vacc_effective  opinion_seas_risk  \
```

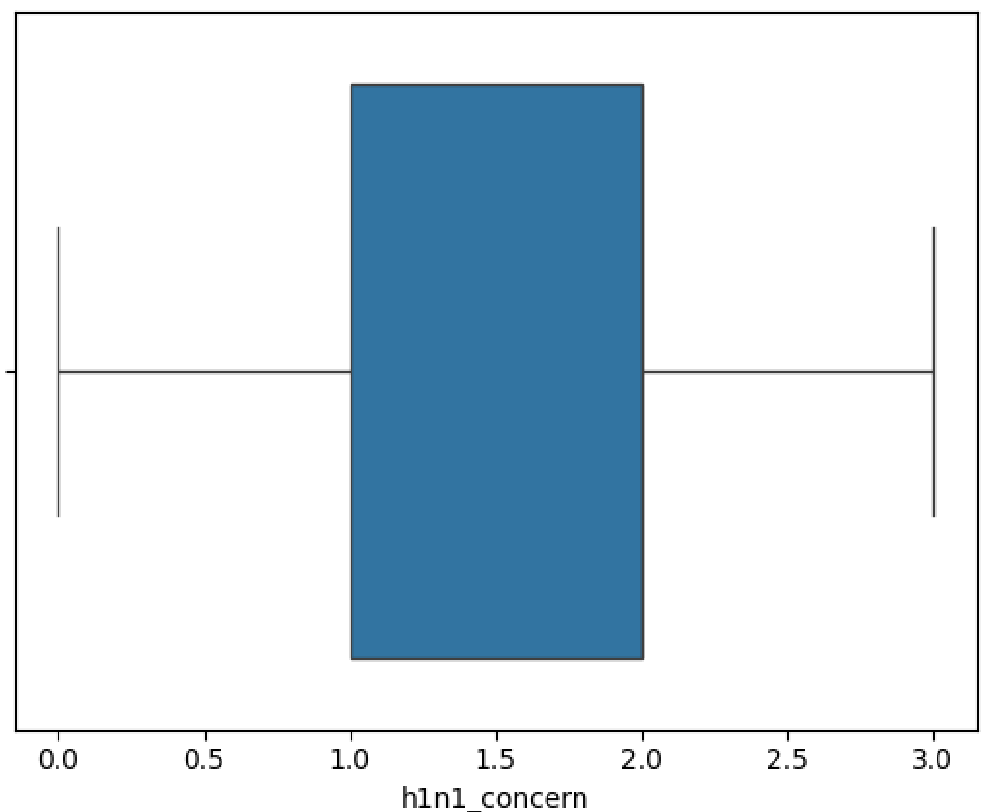
```
0                              2.0               1.0
1                              4.0               2.0
2                              4.0               1.0
3                              5.0               4.0
4                              3.0               1.0
...                            ...               ...
26702                          5.0               2.0
26703                          5.0               1.0
26704                          5.0               4.0
26705                          2.0               1.0
26706                          5.0               1.0

       opinion_seas_sick_from_vacc  household_adults
household_children  \
0                              2.0              0.0
0.0
1                              4.0              0.0
0.0
2                              2.0              2.0
0.0
3                              1.0              0.0
0.0
4                              4.0              1.0
0.0
...                            ...              ...                 .
..
26702                          2.0              0.0
0.0
26703                          1.0              1.0
0.0
26704                          2.0              0.0
0.0
26705                          2.0              1.0
0.0
26706                          1.0              1.0
0.0

       h1n1_vaccine  seasonal_vaccine
0               0.0               0.0
1               0.0               1.0
2               0.0               0.0
3               0.0               1.0
4               0.0               0.0
...             ...               ...
26702           0.0               0.0
26703           0.0               0.0
26704           0.0               1.0
26705           0.0               0.0
26706           0.0               0.0
```

```
[26707 rows x 25 columns]

df_new = pd.concat(
    [pd.DataFrame(df_categorical_preprocessed),
pd.DataFrame(df_numerical)], axis=1
)
```

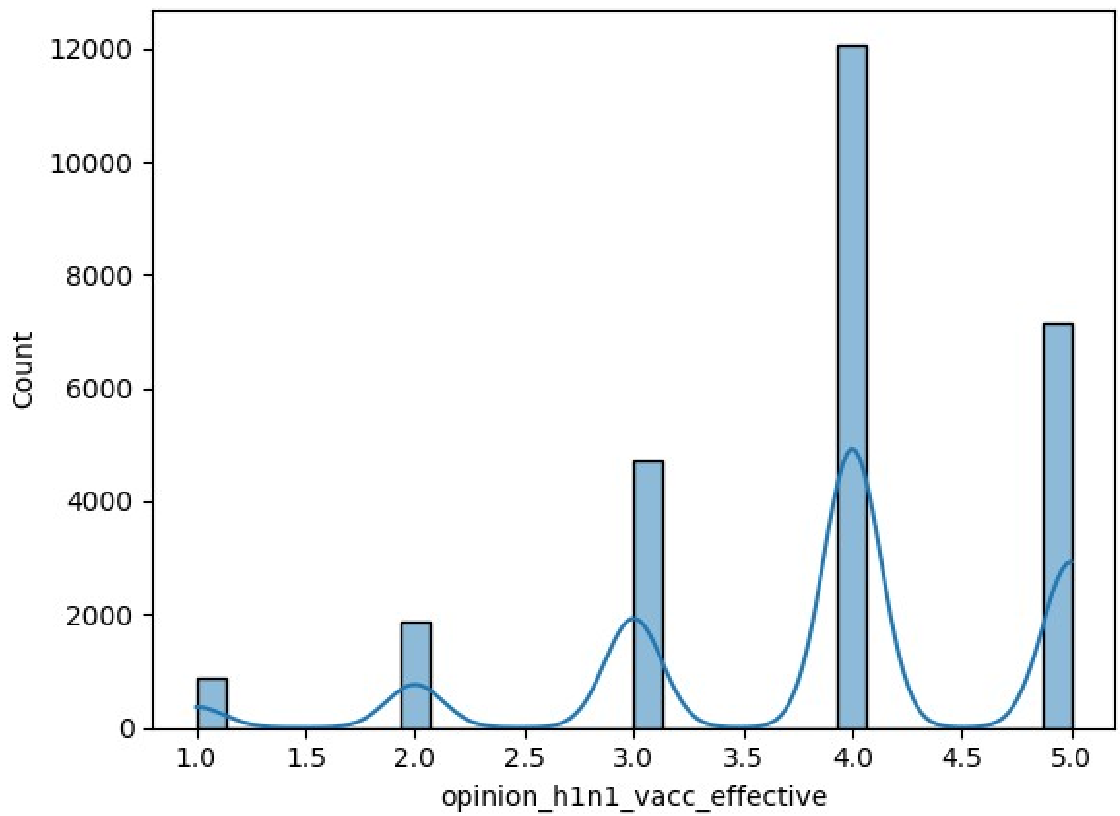
Statistical Analysis

Univariate analysis

```
sns.boxplot(data=df, x='h1n1_concern')

<Axes: xlabel='h1n1_concern'>
```

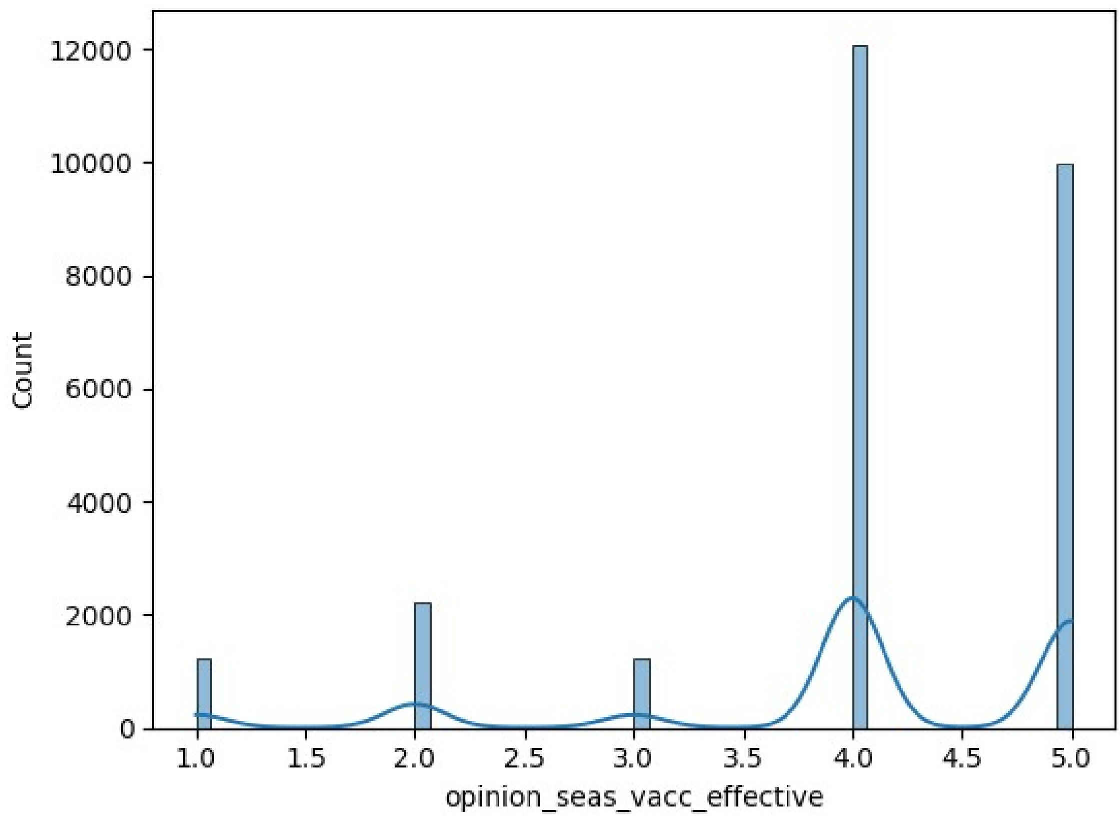



```
sns.histplot(data=df, x='opinion_h1n1_vacc_effective', kde=True)

<Axes: xlabel='opinion_h1n1_vacc_effective', ylabel='Count'>
```




```
sns.histplot(data=df, x='opinion_seas_vacc_effective', kde=True)
```

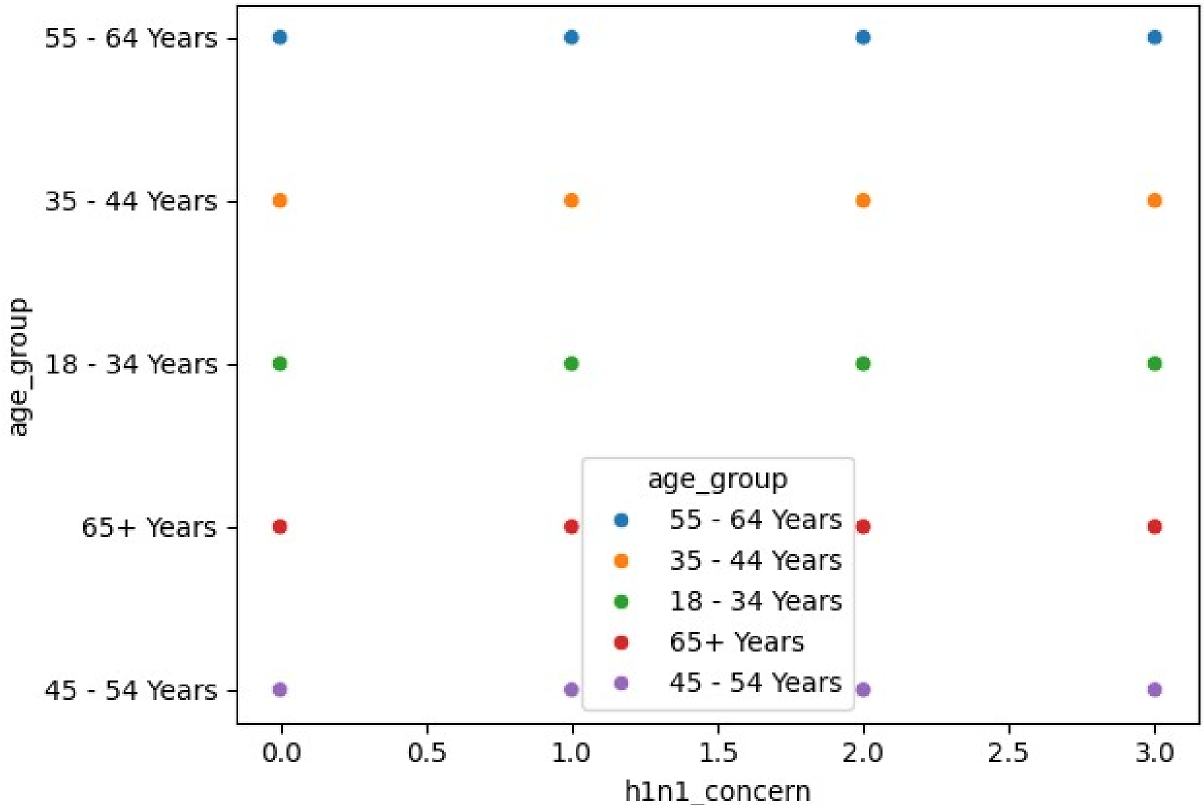
```
<Axes: xlabel='opinion_seas_vacc_effective', ylabel='Count'>
```



Bivariate analysis

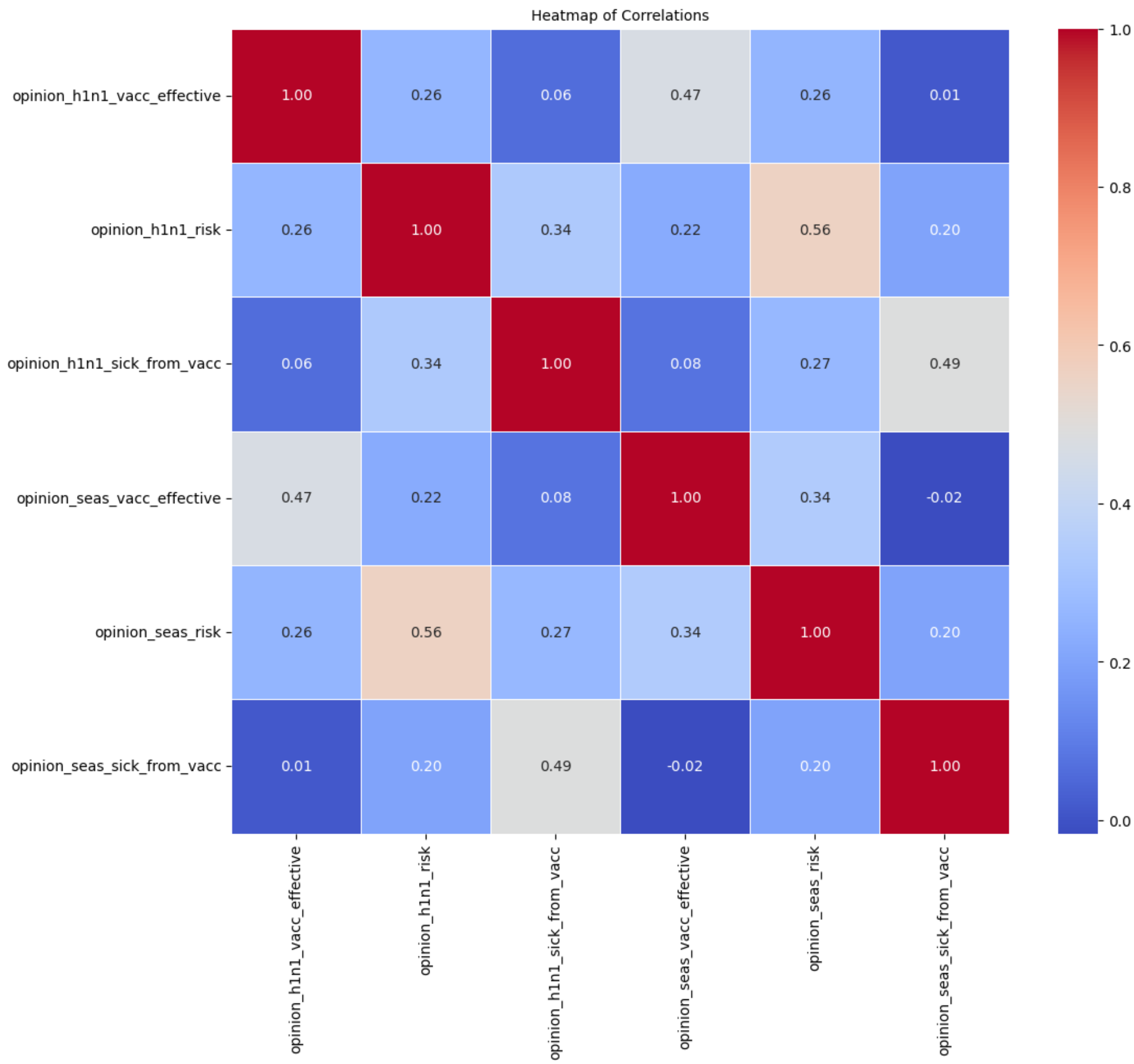
```
sns.scatterplot(data=df, x= 'h1n1_concern',y='age_group',
hue='age_group')
```

```
<Axes: xlabel='h1n1_concern', ylabel='age_group'>
```



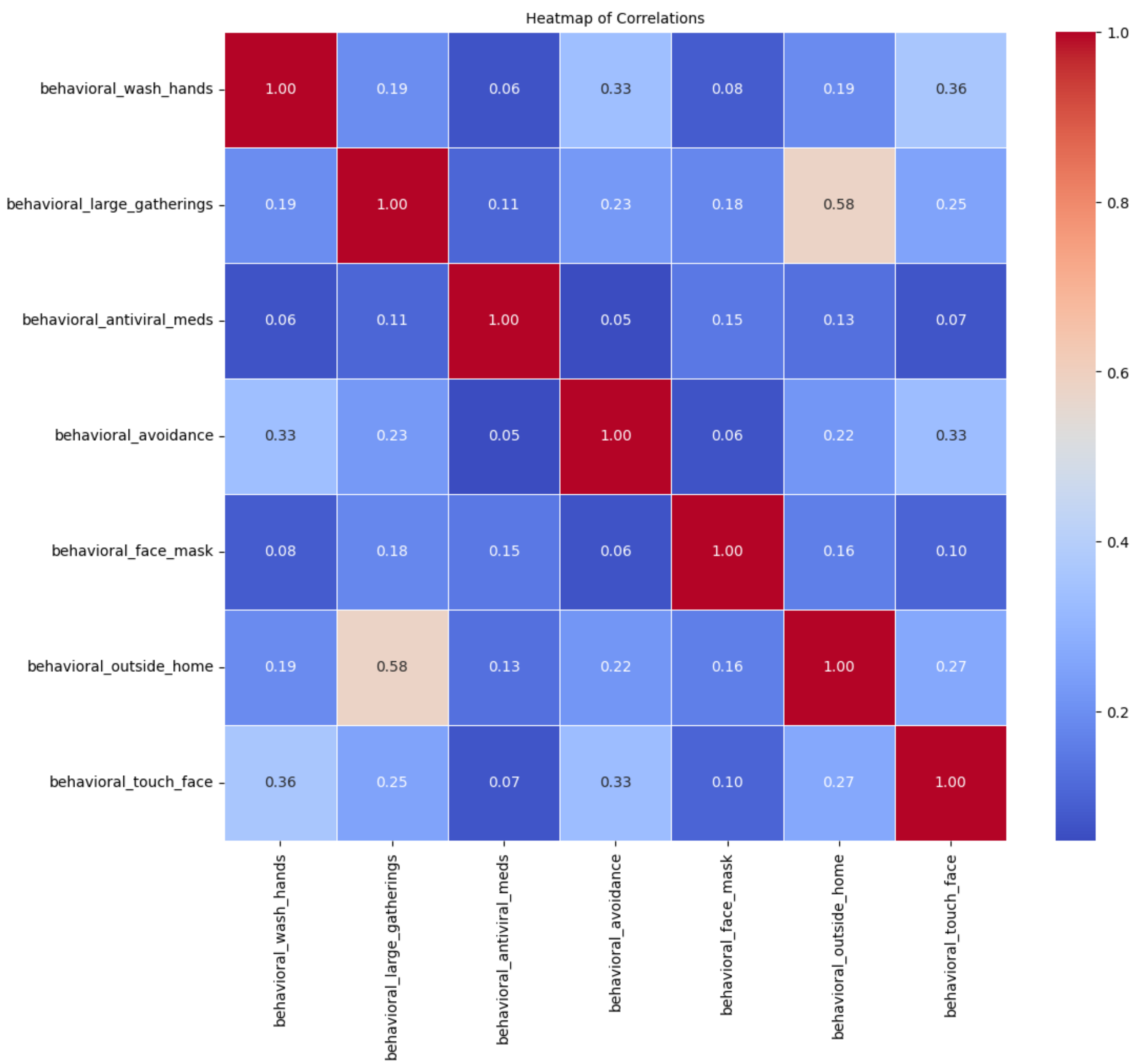

```
# data
opinion_response = df[
                    ['opinion_h1n1_vacc_effective',
                      'opinion_h1n1_risk',
                      'opinion_h1n1_sick_from_vacc',
                      'opinion_seas_vacc_effective',
                      'opinion_seas_risk',
                      'opinion_seas_sick_from_vacc']
]

# Create a DataFrame
opinion_response = pd.DataFrame(opinion_response)

# Compute the correlation matrix
correlation_matrix = opinion_response.corr()

# Create a heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm",
fmt=".2f", linewidths=0.5)

# Add titles and labels
# Add titles and labels
```

```
plt.title('Heatmap of Correlations', fontsize=10)
plt.show()
```




```
# data
behavior_response = df[
                    ['behavioral_wash_hands',
                      'behavioral_large_gatherings',
                      'behavioral_antiviral_meds',
                      'behavioral_avoidance',
                      'behavioral_face_mask',
                      'behavioral_outside_home',
                      'behavioral_touch_face']
]
```

```
# Create a DataFrame
behavior_response = pd.DataFrame(behavior_response)

# Compute the correlation matrix
correlation_matrix = behavior_response.corr()

# Create a heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm",
fmt=".2f", linewidths=0.5)

# Add titles and labels
# Add titles and labels
plt.title('Heatmap of Correlations', fontsize=10)
plt.show()
```



Predictive Statistics

Modeling

- Basleine model
- Refined model
- Evaluation
- Adress class imbalance

```
# Model creation
X=df_new.drop(['h1n1_vaccine','seasonal_vaccine'], axis=1)
y=df_new[['h1n1_vaccine','seasonal_vaccine']]

y.shape

(26707, 2)

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

Scaling

```
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

X_train_scaled.shape

(21365, 33)

y_train.shape

(21365, 2)
```

Multi-label Classification problem

- Binary Relevance
- Classifier chain
- Label Powerset
- Gradient Boosting
- SGD classifier

Baseline Model

- Classifier chain Logistic regression

```
from skmultilearn.problem_transform import ClassifierChain
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

chain1 = ClassifierChain(classifier=LogisticRegression())
chain1.fit(X_train_scaled,y_train)
predict_lr = chain1.predict(X_test_scaled)

lr_cc= accuracy_score(y_test,predict_lr)
lr_cc

0.6761512542119057
```

- Binary Relevance + Gradient Boosting

```
from sklearn.ensemble import GradientBoostingClassifier
from skmultilearn.problem_transform import BinaryRelevance

classifier5 = BinaryRelevance(GradientBoostingClassifier())
classifier5.fit(X_train_scaled,y_train)
predictions_gb = classifier5.predict(X_test_scaled)

gb_br= accuracy_score(y_test,predictions_gb)
gb_br

0.6834518906776488
```

- Classifier chain + Gradient boosting

```
classifier2 = ClassifierChain(GradientBoostingClassifier())
classifier2.fit(X_train_scaled,y_train)
predictions_CC = classifier2.predict(X_test_scaled)

gb_CC= accuracy_score(y_test,predictions_CC)
gb_CC

0.6840134780980907
```

- Label powerset + Gradient Boosting

```
from skmultilearn.problem_transform import LabelPowerset

model = LabelPowerset(GradientBoostingClassifier())
model_1 = model.fit(X_train_scaled, y_train)
predictions_nb_ps1 = model.predict(X_test_scaled)

nb_ps= accuracy_score(y_test,predictions_nb_ps1)
nb_ps
```

```
0.690565331336578
```

Othermodels

- Binary Relevance + KNN

```
from sklearn.neighbors import KNeighborsClassifier

classifier3 = BinaryRelevance(KNeighborsClassifier())
classifier3.fit(X_train_scaled,y_train)
predictions_rf = classifier3.predict(X_test_scaled)

rf_br= accuracy_score(y_test,predictions_rf)
rf_br

0.5990265818045676
```

- Binary Relevance + Logistic Regression

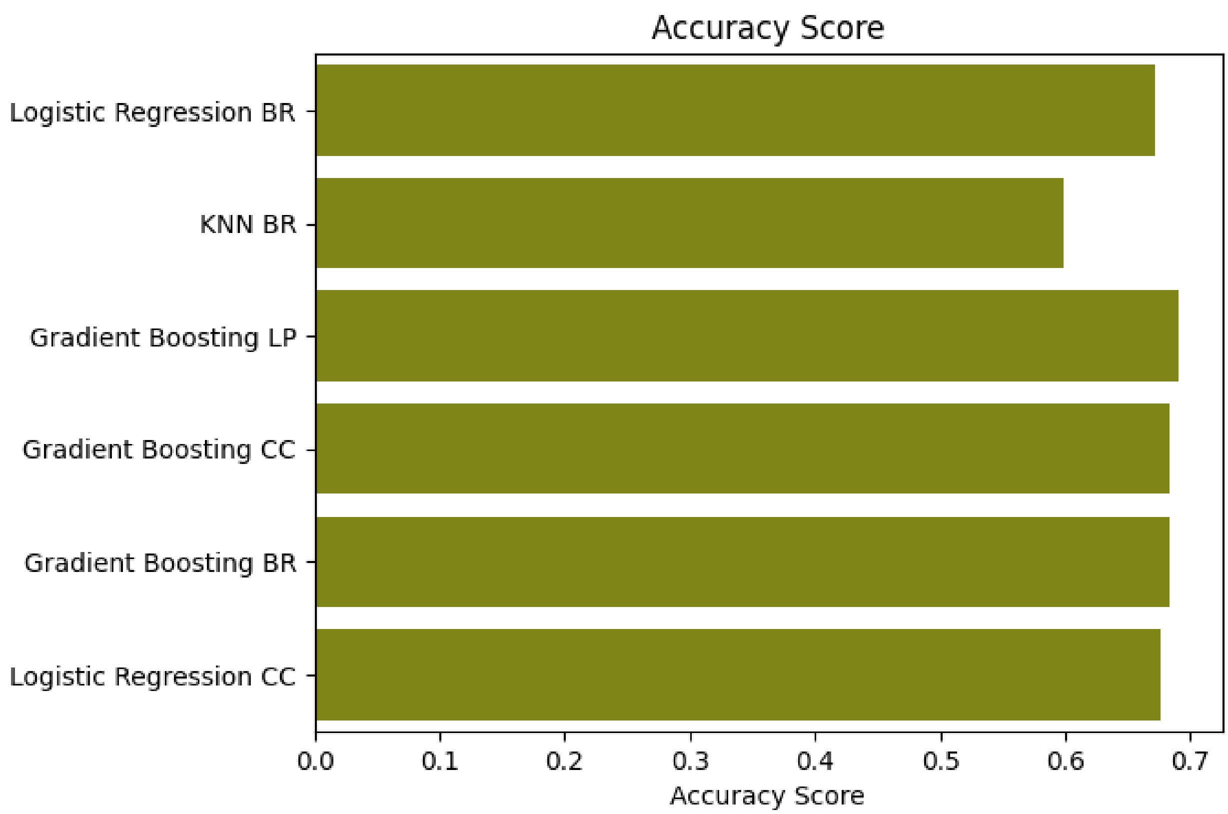
```
classifier1 = BinaryRelevance(LogisticRegression())
classifier1.fit(X_train_scaled,y_train)
predictions_lr = classifier1.predict(X_test_scaled)

lr_br=accuracy_score(y_test,predictions_lr)
lr_br

0.6718457506551854
```

Evaluation and insights

- Report metrics
- Analyze feature importance
- Interpretation

Model Performance

```
Accuracy_Score = [lr_br,rf_br,nb_ps,gb_CC,gb_br,lr_cc]
Models = ['Logistic Regression BR', 'KNN BR' , 'Gradient Boosting LP',
'Gradient Boosting CC',
          'Gradient Boosting BR', 'Logistic Regression CC']

sns.barplot(x=Accuracy_Score, y=Models, color="xkcd:baby poop green")
plt.xlabel('Accuracy Score')
plt.title('Accuracy Score')
plt.show()
```



Hypertuning for the best model

- Label powerset + Gradient boosting

```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import GradientBoostingClassifier
from skmultilearn.problem_transform import LabelPowerset
from sklearn.metrics import make_scorer, accuracy_score
from scipy.stats import uniform, randint

# Define the parameter distribution
param_distributions = {
    'classifier__learning_rate': uniform(0.01, 0.3),  # Learning rate
    'classifier__n_estimators': randint(100, 500),    # Number of 
trees
    'classifier__max_depth': randint(3, 10),          # Maximum tree 
depth
    'classifier__min_samples_split': randint(2, 20),  # Minimum 
samples to split
    'classifier__min_samples_leaf': randint(1, 10),   # Minimum 
samples per leaf
}

# Wrap GradientBoostingClassifier with LabelPowerset
```

```
base_classifier = GradientBoostingClassifier()
model = LabelPowerset(classifier=base_classifier)

# RandomizedSearchCV setup
random_search = RandomizedSearchCV(
    estimator=model,
    param_distributions=param_distributions,
    scoring=make_scorer(accuracy_score),
    n_iter=30,  # Number of parameter settings sampled
    cv=3,       # Cross-validation folds
    verbose=2,
    random_state=42,
    n_jobs=-1
)

# Fit the search
random_search.fit(X_train_scaled, y_train)

# Best model and parameters
best_model = random_search.best_estimator_
best_params = random_search.best_params_
print("Best Parameters:", best_params)

# Evaluate performance on test data
predictions_nb_ps = best_model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, predictions_nb_ps)
print("Accuracy on Test Set:", accuracy)
```

```
Fitting 3 folds for each of 30 candidates, totalling 90 fits
```

```
C:\Users\DELL\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-
packages\Python312\site-packages\sklearn\model_selection\
_search.py:1102: UserWarning: One or more of the test scores are non-
finite: [nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan]
  warnings.warn(
```

```
Best Parameters: {'classifier__learning_rate':
np.float64(0.12236203565420874), 'classifier__max_depth': 7,
'classifier__min_samples_leaf': 8, 'classifier__min_samples_split': 8,
'classifier__n_estimators': 221}
Accuracy on Test Set: 0.6722201422688132
```

```
from sklearn.metrics import accuracy_score, precision_score,
recall_score,f1_score, roc_auc_score, classification_report,roc_curve

# Evaluate on test data
predictions_nb_ps = model_1.predict(X_test_scaled)
```

```python
# Calculate metrics
nb_ps_accuracy = accuracy_score(y_test, predictions_nb_ps)
nb_ps_precision = precision_score(y_test, predictions_nb_ps,
average='weighted')  # Use 'weighted' for multilabel
nb_ps_recall = recall_score(y_test, predictions_nb_ps,
average='weighted')
nb_ps_f1 = f1_score(y_test, predictions_nb_ps, average='weighted')

# Print results
print("Accuracy:", nb_ps_accuracy)
print("Precision:", nb_ps_precision)
print("Recall:", nb_ps_recall)
print("f1_score:", nb_ps_f1)
```

```
Accuracy: 0.690565331336578
Precision: 0.7503850457154486
Recall: 0.6467467187936331
f1_score: 0.6940706211013609
```

```python
print(classification_report(y_test,predictions_nb_ps))
```

```
              precision    recall  f1-score   support

           0       0.65      0.51      0.57      1130
           1       0.80      0.71      0.75      2451

   micro avg       0.75      0.65      0.70      3581
   macro avg       0.72      0.61      0.66      3581
weighted avg       0.75      0.65      0.69      3581
 samples avg       0.32      0.32      0.31      3581


C:\Users\DELL\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-
packages\Python312\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
C:\Users\DELL\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-
packages\Python312\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Recall is ill-defined
and being set to 0.0 in samples with no true labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
C:\Users\DELL\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-
packages\Python312\site-packages\sklearn\metrics\
```

```
_classification.py:1531: UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 in samples with no true nor predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

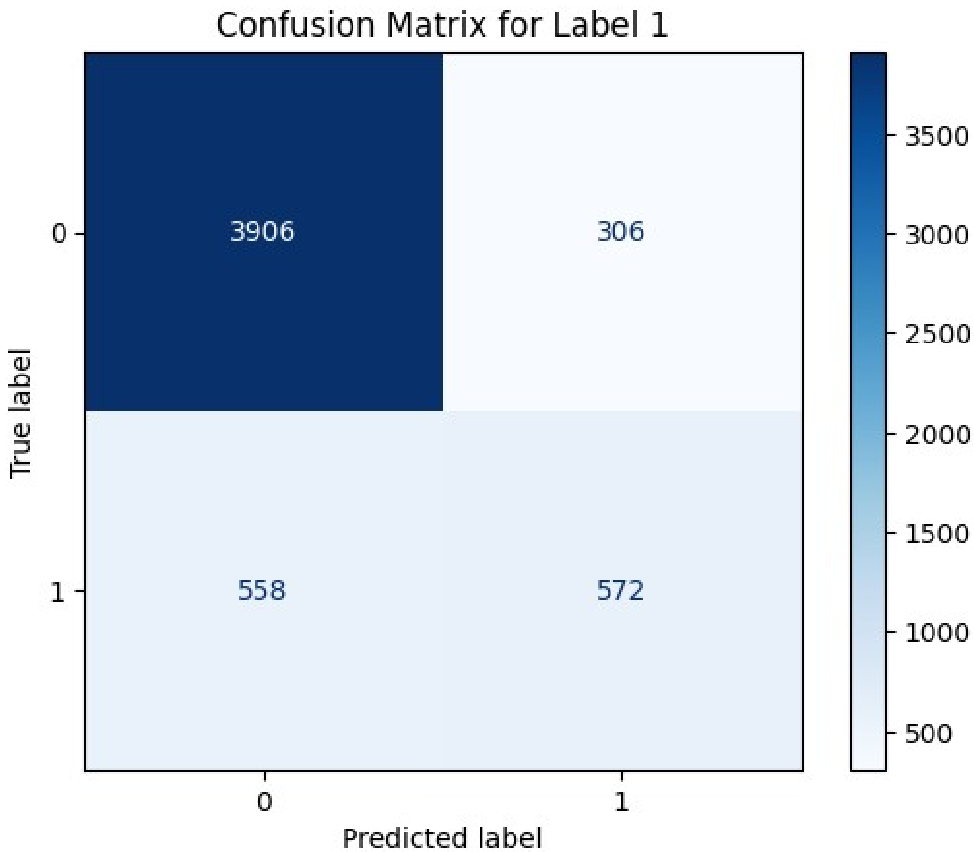
```
# Ensure y_test and predictions_nb_ps are numpy arrays
y_test = np.array(y_test.toarray()) if not isinstance(y_test, np.ndarray) else y_test
predictions_nb_ps = np.array(predictions_nb_ps.toarray()) if not isinstance(predictions_nb_ps, np.ndarray) else predictions_nb_ps

roc_auc = roc_auc_score(y_test,predictions_nb_ps)

roc_auc
```

```
np.float64(0.7476184114825843)
```

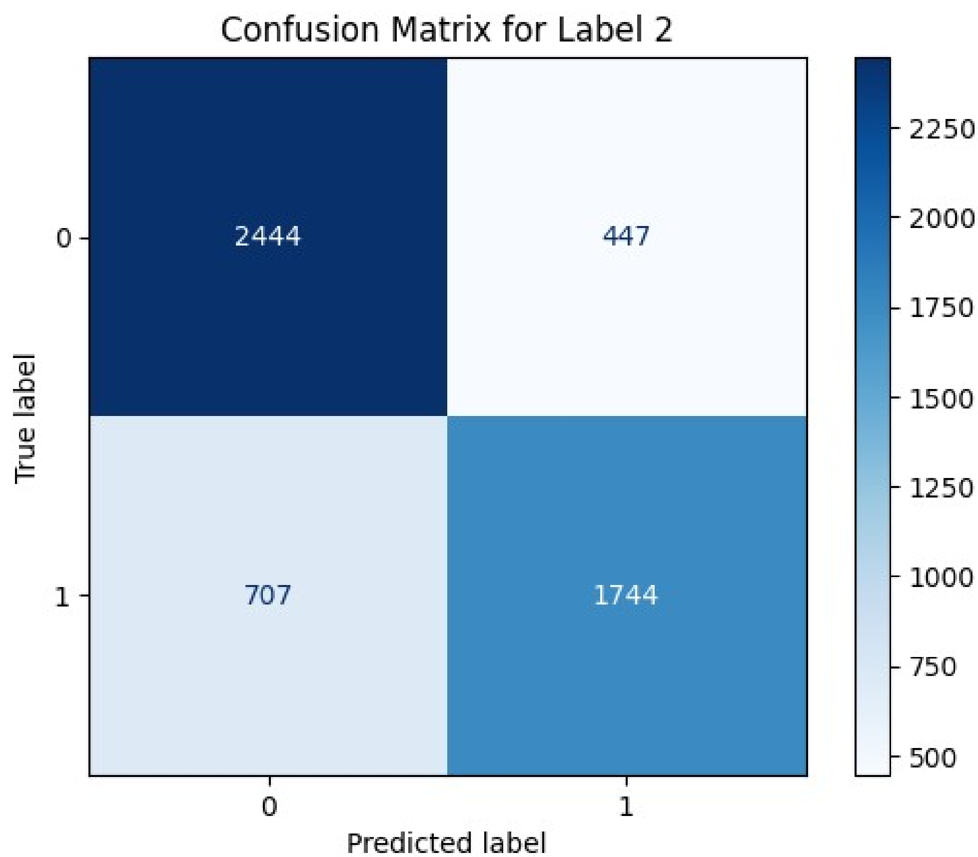
- The curve can not be plotted for a multiclass label

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, accuracy_score
import matplotlib.pyplot as plt
import numpy as np

# Ensure y_test and predictions_nb_ps are numpy arrays
y_test = np.array(y_test.toarray()) if not isinstance(y_test, np.ndarray) else y_test
predictions_nb_ps = np.array(predictions_nb_ps.toarray()) if not isinstance(predictions_nb_ps, np.ndarray) else predictions_nb_ps

# Create confusion matrices for each label
for i in range(y_test.shape[1]):  # Iterate over each label (column)
    cm = confusion_matrix(y_test[:, i], predictions_nb_ps[:, i])  # Confusion matrix for label i
    disp = ConfusionMatrixDisplay(confusion_matrix=cm)  # Display confusion matrix

    # Plot
    print(f"Confusion Matrix for Label {i + 1}:")
    disp.plot(cmap="Blues", colorbar=True)
    plt.title(f"Confusion Matrix for Label {i + 1}")
    plt.show()
```

```
Confusion Matrix for Label 1:
```




```
Confusion Matrix for Label 2:
```



Project Report

1. Modeling Section

Baseline Model

We started with a baseline Logistic Regression model to establish a benchmark. The model used the default hyperparameters and was trained on preprocessed data, including scaled numeric features and labeled categorical variables. The dataset was split into an 80-20 train-test split to evaluate performance.

Feature Engineering

Before modeling, the following preprocessing steps were applied:

- **Missing Value Imputation**: Median imputation for numeric features and mode for categorical features.
- **Scaling**: StandardScaler was used for continuous variables.
- **Labeling**: LabelEncoder was applied to categorical variables.

Model Refinement

- To improve performance, hyperparameter tuning was performed using GridSearchCV for the best performing model which is the LabelPowerset with the GradientBoostingClassifier.

2. Evaluation Section

Metrics Chosen

Given the dataset's multiclass classification problem and potential class imbalance, the following metrics were used:

- **Accuracy**: To measure overall correctness.
- **Precision**: To evaluate the proportion of true positive predictions among all positive predictions.
- **Recall**: To assess the model's ability to detect true positives.
- **F1-Score**: As a balance between precision and recall.
- **ROC-AUC**: To measure the overall ability of the model to distinguish between classes.

Results

- The baseline Logistic Regression achieved an accuracy of 67% and an F1-score of 0.71 on the test set.
- The tuned Gradient Boosting gave an accuracy score of 67%

---

3. Findings Section

- **Feature Importance**: Vaccine awareness, health conditions, and demographics (age, education level) were the most influential features across models.
- **Model Performance**: While both the Logistic Regression and Gradient Boosting performed comparably, the Gradient Boosting showed greater robustness across different subsets of the data.
- **Limitations**: The models struggled with certain subpopulations where data was sparse or highly imbalanced, such as older age groups with limited survey responses.

---

4. Recommendations Section

- **Targeted Campaigns**: Public health organizations should focus on populations identified as less likely to vaccinate, such as individuals with lower education levels or limited vaccine awareness.
- **Data-Driven Strategies**: Modifying awareness campaigns to increase education around the benefits of H1N1 and seasonal flu vaccines could improve uptake.

- **Future Improvements**: Collecting more balanced data across diverse demographics will enhance model accuracy and applicability.
- **Model Application**: Use predictions to identify high-risk areas for non-vaccination and allocate resources to these regions.