Predicting Seasonal Flu Vaccine Uptake Using Demographic, Health, and Opinion Data

Business Understanding

According to the World Health Organization, seasonal influenza (the flu) is an acute respiratory infection caused by influenza viruses, which circulate globally. It presents a significant year-round disease burden, causing illnesses that range in severity, sometimes leading to hospitalization and death. Annually, there are around a billion cases of seasonal influenza, including 3–5 million cases of severe illness, resulting in 290,000 to 650,000 respiratory deaths.

Hospitalization and death due to influenza predominantly occur among high-risk groups. In industrialized countries, most influenza-associated deaths occur in individuals aged 65 years or older. In children under 5 years of age, 99% of influenza-related lower respiratory tract infection deaths occur in developing countries.

Most people recover from fever and other symptoms within a week without medical attention. However, influenza can cause severe illness or death, especially among high-risk groups, including the very young, the elderly, pregnant women, health workers, and individuals with serious medical conditions. Influenza spreads easily through respiratory droplets when people cough or sneeze. Vaccination is the best way to prevent the disease.

In temperate climates, seasonal epidemics occur mainly during winter, while in tropical regions, influenza can occur year-round, causing irregular outbreaks. There are four types of influenza viruses: A, B, C, and D, with types A and B causing seasonal epidemics.

Symptoms: Influenza is characterized by sudden onset of fever, dry cough, headache, muscle and joint pain, severe malaise, sore throat, and runny nose. Symptoms typically begin 1–4 days after infection and last around a week.

Treatment: Treatment focuses on relieving symptoms. High-risk patients should seek medical attention and may require antiviral drugs.

Epidemiology: All age groups can be affected, but those at greater risk include pregnant women, children under 5, older adults, individuals with chronic medical conditions, and those with immunosuppressive conditions or treatments. Health and care workers are also at high risk.

Prevention: Vaccination is the most effective preventive measure. Safe and effective vaccines have been used for over 60 years. Annual vaccination is recommended, especially for high-risk groups and their carers. Other preventive measures include hand hygiene, respiratory etiquette, and avoiding close contact with sick individuals.

Problem Statement

The problem we aim to address is the suboptimal flu vaccination rates among certain demographic groups, despite the availability and proven efficacy of influenza vaccines. This results in preventable illnesses, hospitalizations, and deaths, particularly in underserved populations. Public health officials, healthcare providers, policy makers, community organizations, and researchers need to understand the factors influencing vaccine uptake. This understanding is crucial for designing effective public health interventions that enhance vaccine uptake and improve health outcomes among the general public, especially those in underserved populations

Objective and Goals

Objective:

Develop a machine learning model to predict the likelihood of individuals receiving their seasonal flu vaccines based on their socioeconomic factors. The model aims to identify key predictors of vaccine uptake to inform targeted public health interventions and improve vaccination rates.

Goals of Data Analysis and Prediction:

Identify Key Predictors: Determine which socioeconomic factors (e.g., income, education, employment status) are most strongly associated with the likelihood of receiving a seasonal flu vaccine.

Model Development: Build a predictive model that accurately estimates the probability of an individual receiving the vaccine based on their socioeconomic characteristics.

Inform Public Health Strategies: Provide actionable insights through reports and dashboards that can be used by public health officials to design targeted interventions aimed at increasing flu vaccination rates, particularly in underserved or high-risk populations.

Resource Allocation: Help allocate resources more effectively by identifying demographic groups that are less likely to receive the vaccine and may benefit from additional outreach and support.

Imports and Reading Data

In [1]:
```
from pathlib import Path

import numpy as np
import pandas as pd
import matplotlib as plt
import seaborn as sns
plt.style.use('ggplot')
pd.set_option('max_columns', 200)
```

In [2]:
```
df_train_features = pd.read_csv("training_set_features.csv") # X_train
```

In [3]:
```
df_train_labels = pd.read_csv("training_set_labels.csv") #y_train
```

In [4]:
```
df_test_features = pd.read_csv("test_set_features.csv")  #X_test data s
```

Data Understanding

In [5]:
```
df_train_features.shape
```

Out[5]:
```
(26707, 36)
```

The train features data has 36 columns and 26707 rows

In [6]:
```
df_train_features.head()
```

Out[6]:

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavioral
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	

In [7]:
```
df_train_features.tail()
```

Out[7]:

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behav
26702	26702	2.0	0.0	0.0	
26703	26703	1.0	2.0	0.0	
26704	26704	2.0	2.0	0.0	
26705	26705	1.0	1.0	0.0	
26706	26706	0.0	0.0	0.0	

In [8]:
```
df_train_features.dtypes
```

Out[8]:
```
respondent_id                    int64
h1n1_concern                   float64
h1n1_knowledge                 float64
behavioral_antiviral_meds      float64
behavioral_avoidance           float64
behavioral_face_mask           float64
behavioral_wash_hands          float64
behavioral_large_gatherings    float64
behavioral_outside_home        float64
behavioral_touch_face          float64
doctor_recc_h1n1               float64
doctor_recc_seasonal           float64
chronic_med_condition          float64
child_under_6_months           float64
health_worker                  float64
health_insurance               float64
opinion_h1n1_vacc_effective    float64
opinion_h1n1_risk              float64
opinion_h1n1_sick_from_vacc    float64
opinion_seas_vacc_effective    float64
opinion_seas_risk              float64
opinion_seas_sick_from_vacc    float64
age_group                       object
education                       object
race                            object
sex                             object
income_poverty                  object
marital_status                  object
rent_or_own                     object
employment_status               object
hhs_geo_region                  object
census_msa                      object
household_adults               float64
household_children             float64
employment_industry             object
employment_occupation           object
dtype: object
```

In [9]:
```
df_train_features.describe()
```

Out[9]:

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behav
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	

In [10]:
```
df_train_labels.shape
```
Out[10]:
```
(26707, 3)
```

The train labels data has 3 columns and 26707 rows

In [11]:
```
df_train_labels.head()
```
Out[11]:

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

In [12]:
```
df_train_labels.tail()
```
Out[12]:

	respondent_id	h1n1_vaccine	seasonal_vaccine
26702	26702	0	0
26703	26703	0	0
26704	26704	0	1
26705	26705	0	0
26706	26706	0	0

In [13]:
```
df_train_labels.dtypes
```
Out[13]:
```
respondent_id       int64
h1n1_vaccine        int64
seasonal_vaccine    int64
dtype: object
```

In [14]:
```
df_train_labels.describe()
```
Out[14]:

	respondent_id	h1n1_vaccine	seasonal_vaccine
count	26707.000000	26707.000000	26707.000000
mean	13353.000000	0.212454	0.465608
std	7709.791156	0.409052	0.498825
min	0.000000	0.000000	0.000000
25%	6676.500000	0.000000	0.000000
50%	13353.000000	0.000000	0.000000
75%	20029.500000	0.000000	1.000000
max	26706.000000	1.000000	1.000000

In [15]:
```
df_train_labels.mode()
```

Out[15]:

	respondent_id	h1n1_vaccine	seasonal_vaccine
0	0	0.0	0.0
1	1	NaN	NaN
2	2	NaN	NaN
3	3	NaN	NaN
4	4	NaN	NaN
...	...	...	...
26702	26702	NaN	NaN
26703	26703	NaN	NaN
26704	26704	NaN	NaN
26705	26705	NaN	NaN
26706	26706	NaN	NaN

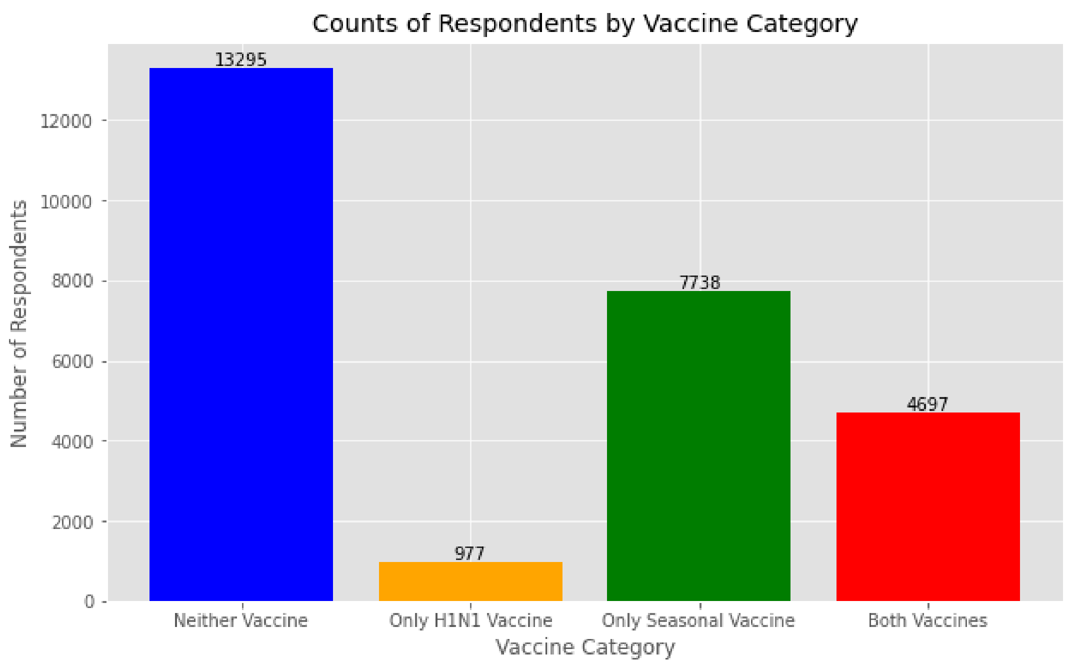
26707 rows × 3 columns

In [16]:
```
mode_value_counts = df_train_labels.apply(lambda x: x.value_counts().il

print(mode_value_counts)
```

```
respondent_id           1
h1n1_vaccine        21033
seasonal_vaccine    14272
dtype: int64
```

We see that 21,033 respondents reveived the h1n1 vaccine while 14,272 respondents received the seasonal vaccine.

In [17]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Count the number of respondents in each category
neither_vaccine = len(df_train_labels[(df_train_labels['h1n1_vaccine']
only_h1n1_vaccine = len(df_train_labels[(df_train_labels['h1n1_vaccine'
only_seasonal_vaccine = len(df_train_labels[(df_train_labels['h1n1_vacc
both_vaccines = len(df_train_labels[(df_train_labels['h1n1_vaccine'] ==

# Data for the bar chart
categories = ['Neither Vaccine', 'Only H1N1 Vaccine', 'Only Seasonal Va
counts = [neither_vaccine, only_h1n1_vaccine, only_seasonal_vaccine, bo

# Plot the bar chart with labels
plt.figure(figsize=(10, 6))
bars = plt.bar(categories, counts, color=['blue', 'orange', 'green', 'r

# Add labels to the bars
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, height, str(height), ha

plt.xlabel('Vaccine Category')
plt.ylabel('Number of Respondents')
plt.title('Counts of Respondents by Vaccine Category')
plt.show()
```



From the data above 13,295 received neither vaccine, 977 received only the H1N1 vaccine, 7,738 received the Seasonal vaccine only while 4,697 received both vaccines.

For this project, I will focus on the **seasonal vaccine** potential target respondents.

Data Preparation

Training Features Dataset

In [18]:
```
# dropping the h1n1 column from the train labels data set
df_train_labels = df_train_labels.drop('h1n1_vaccine', axis=1)

print(df_train_labels)
```

```
       respondent_id  seasonal_vaccine
0                  0                 0
1                  1                 1
2                  2                 0
3                  3                 1
4                  4                 0
...              ...               ...
26702          26702                 0
26703          26703                 0
26704          26704                 1
26705          26705                 0
26706          26706                 0

[26707 rows x 2 columns]
```

In [19]:
```
# dropping the h1n1 related columns from the train features  data set
columns_to_drop = ['h1n1_concern','h1n1_knowledge','doctor_recc_h1n1','
df_train_features_dropped = df_train_features.drop(columns_to_drop, axi
```

In [20]:
```
df_train_features.columns
```

Out[20]:
```
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_seaso
nal',
       'chronic_med_condition', 'child_under_6_months', 'health_worke
r',
       'health_insurance', 'opinion_h1n1_vacc_effective', 'opinion_h1n
1_risk',
       'opinion_h1n1_sick_from_vacc', 'opinion_seas_vacc_effective',
       'opinion_seas_risk', 'opinion_seas_sick_from_vacc', 'age_grou
p',
       'education', 'race', 'sex', 'income_poverty', 'marital_status',
       'rent_or_own', 'employment_status', 'hhs_geo_region', 'census_m
sa',
       'household_adults', 'household_children', 'employment_industr
y',
       'employment_occupation'],
      dtype='object')
```

In [21]:
```
# merging the train features and the train labels to form my merged tra

df_train = pd.merge(df_train_labels, df_train_features_dropped, on='res
df_train.shape
```
Out[21]:
```
(26707, 31)
```

In [22]:
```
df_train.head()
```
Out[22]:

	respondent_id	seasonal_vaccine	behavioral_antiviral_meds	behavioral_avoidance	be
0	0	0	0.0	0.0	
1	1	1	0.0	1.0	
2	2	0	0.0	1.0	
3	3	1	0.0	1.0	
4	4	0	0.0	1.0	

In [23]:
```
df_train.tail()
```
Out[23]:

	respondent_id	seasonal_vaccine	behavioral_antiviral_meds	behavioral_avoidance
26702	26702	0	0.0	1.(
26703	26703	0	0.0	1.(
26704	26704	1	0.0	1.(
26705	26705	0	0.0	0.(
26706	26706	0	0.0	1.(

In [24]:
```
#checking the data types
df_train.dtypes
```

Out[24]:
```
respondent_id                    int64
seasonal_vaccine                 int64
behavioral_antiviral_meds      float64
behavioral_avoidance           float64
behavioral_face_mask           float64
behavioral_wash_hands          float64
behavioral_large_gatherings    float64
behavioral_outside_home        float64
behavioral_touch_face          float64
doctor_recc_seasonal           float64
chronic_med_condition          float64
child_under_6_months           float64
health_worker                  float64
health_insurance               float64
opinion_seas_vacc_effective    float64
opinion_seas_risk              float64
opinion_seas_sick_from_vacc    float64
age_group                       object
education                       object
race                            object
sex                             object
income_poverty                  object
marital_status                  object
rent_or_own                     object
employment_status               object
hhs_geo_region                  object
census_msa                      object
household_adults               float64
household_children             float64
employment_industry             object
employment_occupation           object
dtype: object
```

In [25]:
```
df_train.mode()
```

Out[25]:

	respondent_id	seasonal_vaccine	behavioral_antiviral_meds	behavioral_avoidanc
0	0	0.0	0.0	1.(
1	1	NaN	NaN	Nal
2	2	NaN	NaN	Nal
3	3	NaN	NaN	Nal
4	4	NaN	NaN	Nal
...	...	...	...	..
26702	26702	NaN	NaN	Nal
26703	26703	NaN	NaN	Nal
26704	26704	NaN	NaN	Nal
26705	26705	NaN	NaN	Nal
26706	26706	NaN	NaN	Nal

26707 rows × 31 columns

In [26]:
```
from IPython.core.display import HTML

# Adjusting the output cell height to be auto and remove max-height con
HTML("""
<style>
.output_wrapper, .output {
    height: auto !important;
    max-height: none !important;
}
</style>
""")
```

Out[26]:

In [27]:
```
# Inspecting the data to see the categories in the columns
from IPython.display import display


for column in df_train.columns:
    print(f"Counts for column '{column}':")
    print(df_train[column].value_counts())
    print()
    print()
```

```
Counts for column 'respondent_id':
2047     1
7657     1
3371     1
13612    1
15661    1
        ..
12979    1
2740     1
693      1
6838     1
0        1
Name: respondent_id, Length: 26707, dtype: int64


Counts for column 'seasonal_vaccine':
0    14272
1    12435
Name: seasonal_vaccine, dtype: int64


Counts for column 'behavioral_antiviral_meds':
0.0    25335
1.0     1301
Name: behavioral_antiviral_meds, dtype: int64


Counts for column 'behavioral_avoidance':
1.0    19228
0.0     7271
Name: behavioral_avoidance, dtype: int64


Counts for column 'behavioral_face_mask':
0.0    24847
1.0     1841
Name: behavioral_face_mask, dtype: int64


Counts for column 'behavioral_wash_hands':
1.0    22015
0.0     4650
Name: behavioral_wash_hands, dtype: int64


Counts for column 'behavioral_large_gatherings':
0.0    17073
1.0     9547
Name: behavioral_large_gatherings, dtype: int64


Counts for column 'behavioral_outside_home':
0.0    17644
1.0     8981
Name: behavioral_outside_home, dtype: int64


Counts for column 'behavioral_touch_face':
1.0    18001
0.0     8578
Name: behavioral_touch_face, dtype: int64
```

```
Counts for column 'doctor_recc_seasonal':
0.0    16453
1.0     8094
Name: doctor_recc_seasonal, dtype: int64


Counts for column 'chronic_med_condition':
0.0    18446
1.0     7290
Name: chronic_med_condition, dtype: int64


Counts for column 'child_under_6_months':
0.0    23749
1.0     2138
Name: child_under_6_months, dtype: int64


Counts for column 'health_worker':
0.0    23004
1.0     2899
Name: health_worker, dtype: int64


Counts for column 'health_insurance':
1.0    12697
0.0     1736
Name: health_insurance, dtype: int64


Counts for column 'opinion_seas_vacc_effective':
4.0    11629
5.0     9973
2.0     2206
1.0     1221
3.0     1216
Name: opinion_seas_vacc_effective, dtype: int64


Counts for column 'opinion_seas_risk':
2.0    8954
4.0    7630
1.0    5974
5.0    2958
3.0     677
Name: opinion_seas_risk, dtype: int64


Counts for column 'opinion_seas_sick_from_vacc':
1.0    11870
2.0     7633
4.0     4852
5.0     1721
3.0       94
Name: opinion_seas_sick_from_vacc, dtype: int64


Counts for column 'age_group':
65+ Years        6843
```

```
55 - 64 Years    5563
45 - 54 Years    5238
18 - 34 Years    5215
35 - 44 Years    3848
Name: age_group, dtype: int64


Counts for column 'education':
College Graduate    10097
Some College         7043
12 Years             5797
< 12 Years           2363
Name: education, dtype: int64


Counts for column 'race':
White                21222
Black                 2118
Hispanic              1755
Other or Multiple     1612
Name: race, dtype: int64


Counts for column 'sex':
Female    15858
Male      10849
Name: sex, dtype: int64


Counts for column 'income_poverty':
<= $75,000, Above Poverty    12777
> $75,000                     6810
Below Poverty                 2697
Name: income_poverty, dtype: int64


Counts for column 'marital_status':
Married        13555
Not Married    11744
Name: marital_status, dtype: int64


Counts for column 'rent_or_own':
Own     18736
Rent     5929
Name: rent_or_own, dtype: int64


Counts for column 'employment_status':
Employed              13560
Not in Labor Force    10231
Unemployed             1453
Name: employment_status, dtype: int64


Counts for column 'hhs_geo_region':
lzgpxyit    4297
fpwskwrf    3265
qufhixun    3102
oxchjgsf    2859
kbazzjca    2858
```

```
bhuqouqj    2846
mlyzmhmf    2243
lrircsnp    2078
atmpeygn    2033
dqpwygqj    1126
Name: hhs_geo_region, dtype: int64


Counts for column 'census_msa':
MSA, Not Principle  City    11645
MSA, Principle City          7864
Non-MSA                      7198
Name: census_msa, dtype: int64


Counts for column 'household_adults':
1.0    14474
0.0     8056
2.0     2803
3.0     1125
Name: household_adults, dtype: int64


Counts for column 'household_children':
0.0    18672
1.0     3175
2.0     2864
3.0     1747
Name: household_children, dtype: int64


Counts for column 'employment_industry':
fcxhlnwr    2468
wxleyezf    1804
ldnlellj    1231
pxcmvdjn    1037
atmlpfrs     926
arjwrbjb     871
xicduogh     851
mfikgejo     614
vjjrobsf     527
rucpziij     523
xqicxuve     511
saaquncn     338
cfqqtusy     325
nduyfdeo     286
mcubkhph     275
wlfvacwt     215
dotnnunm     201
haxffmxo     148
msuufmds     124
phxvnwax      89
qnlwzans      13
Name: employment_industry, dtype: int64


Counts for column 'employment_occupation':
xtkaffoo    1778
mxkfnird    1509
emcorrxb    1270
cmhcxjea    1247
```

```
xgwztkwe    1082
hfxkjkmi     766
qxajmpny     548
xqwwgdyp     485
kldqjyjy     469
uqqtjvyb     452
tfqavkke     388
ukymxvdu     372
vlluhbov     354
oijqvulv     344
ccgxvspp     341
bxpfxfdn     331
haliazsg     296
rcertsgn     276
xzmlyyjv     248
dlvbwzss     227
hodpvpew     208
dcjcmpih     148
pvmttkik      98
Name: employment_occupation, dtype: int64
```

In [28]:

```
# further drop unnecessary columns in the train data
columns_to_drop3 = ['respondent_id','hhs_geo_region','employment_indust
       'behavioral_face_mask', 'behavioral_wash_hands',
       'behavioral_large_gatherings', 'behavioral_outside_home',
       'behavioral_touch_face','census_msa', 'household_adults', 'house
df_train = df_train.drop(columns_to_drop3, axis=1)
```

In [29]:

```
df_train.head()
```

Out[29]:

	seasonal_vaccine	doctor_recc_seasonal	chronic_med_condition	child_under_6_montl
0	0	0.0	0.0	C
1	1	0.0	0.0	C
2	0	NaN	1.0	C
3	1	1.0	1.0	C
4	0	0.0	0.0	C

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

Out[30]:

	seasonal_vaccine	doctor_recc_seasonal	chronic_med_condition	child_under_6_n
count	26707.000000	24547.000000	25736.000000	25887.(
mean	0.465608	0.329735	0.283261	0.(
std	0.498825	0.470126	0.450591	0.:
min	0.000000	0.000000	0.000000	0.(
25%	0.000000	0.000000	0.000000	0.(
50%	0.000000	0.000000	0.000000	0.(
75%	1.000000	1.000000	1.000000	0.(
max	1.000000	1.000000	1.000000	1.(

In [31]:
```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 26707 entries, 0 to 26706
Data columns (total 17 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   seasonal_vaccine             26707 non-null  int64  
 1   doctor_recc_seasonal         24547 non-null  float64
 2   chronic_med_condition        25736 non-null  float64
 3   child_under_6_months         25887 non-null  float64
 4   health_worker                25903 non-null  float64
 5   health_insurance             14433 non-null  float64
 6   opinion_seas_vacc_effective  26245 non-null  float64
 7   opinion_seas_risk            26193 non-null  float64
 8   opinion_seas_sick_from_vacc  26170 non-null  float64
 9   age_group                    26707 non-null  object 
 10  education                    25300 non-null  object 
 11  race                         26707 non-null  object 
 12  sex                          26707 non-null  object 
 13  income_poverty               22284 non-null  object 
 14  marital_status               25299 non-null  object 
 15  rent_or_own                  24665 non-null  object 
 16  employment_status            25244 non-null  object 
dtypes: float64(8), int64(1), object(8)
memory usage: 3.7+ MB
```

Data Preprocessing

Check for missing data

In [32]:
```
#checking for missing data
df_train.isna().sum()
```

Out[32]:
```
seasonal_vaccine                   0
doctor_recc_seasonal            2160
chronic_med_condition            971
child_under_6_months             820
health_worker                    804
health_insurance               12274
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
dtype: int64
```

In [33]:
```
missing_percentage = (df_train.isnull().mean() * 100).round(2)
missing_percentage
```

Out[33]:
```
seasonal_vaccine                0.00
doctor_recc_seasonal            8.09
chronic_med_condition           3.64
child_under_6_months            3.07
health_worker                   3.01
health_insurance               45.96
opinion_seas_vacc_effective     1.73
opinion_seas_risk               1.92
opinion_seas_sick_from_vacc     2.01
age_group                       0.00
education                       5.27
race                            0.00
sex                             0.00
income_poverty                 16.56
marital_status                  5.27
rent_or_own                     7.65
employment_status               5.48
dtype: float64
```

In [34]:

```
#inspect the data for treatment of missing values

for column in df_train.columns:
    print(f"Counts for column '{column}':")
    print(df_train[column].value_counts())
    print()
```

```
Counts for column 'seasonal_vaccine':
0    14272
1    12435
Name: seasonal_vaccine, dtype: int64

Counts for column 'doctor_recc_seasonal':
0.0    16453
1.0     8094
Name: doctor_recc_seasonal, dtype: int64

Counts for column 'chronic_med_condition':
0.0    18446
1.0     7290
Name: chronic_med_condition, dtype: int64

Counts for column 'child_under_6_months':
0.0    23749
1.0     2138
Name: child_under_6_months, dtype: int64

Counts for column 'health_worker':
0.0    23004
1.0     2899
Name: health_worker, dtype: int64

Counts for column 'health_insurance':
1.0    12697
0.0     1736
Name: health_insurance, dtype: int64

Counts for column 'opinion_seas_vacc_effective':
4.0    11629
5.0     9973
2.0     2206
1.0     1221
3.0     1216
Name: opinion_seas_vacc_effective, dtype: int64

Counts for column 'opinion_seas_risk':
2.0    8954
4.0    7630
1.0    5974
5.0    2958
3.0     677
Name: opinion_seas_risk, dtype: int64

Counts for column 'opinion_seas_sick_from_vacc':
1.0    11870
2.0     7633
4.0     4852
5.0     1721
3.0       94
Name: opinion_seas_sick_from_vacc, dtype: int64

Counts for column 'age_group':
65+ Years        6843
55 - 64 Years    5563
45 - 54 Years    5238
18 - 34 Years    5215
35 - 44 Years    3848
Name: age_group, dtype: int64
```

```
Counts for column 'education':
College Graduate    10097
Some College         7043
12 Years             5797
< 12 Years           2363
Name: education, dtype: int64

Counts for column 'race':
White                21222
Black                 2118
Hispanic              1755
Other or Multiple     1612
Name: race, dtype: int64

Counts for column 'sex':
Female    15858
Male      10849
Name: sex, dtype: int64

Counts for column 'income_poverty':
<= $75,000, Above Poverty    12777
> $75,000                     6810
Below Poverty                 2697
Name: income_poverty, dtype: int64

Counts for column 'marital_status':
Married        13555
Not Married    11744
Name: marital_status, dtype: int64

Counts for column 'rent_or_own':
Own     18736
Rent     5929
Name: rent_or_own, dtype: int64

Counts for column 'employment_status':
Employed              13560
Not in Labor Force    10231
Unemployed             1453
Name: employment_status, dtype: int64
```

The health insurance column has 46% missing data. This is quite significant. I will create another catergory called "Unknown" to represent missing values. This will help preserve the missingness information and will prevent bias towards the most frequent category.

In [35]:
```
df_train['health_insurance'].fillna('Unknown', inplace=True)
```

The income column has 17% missing data. This is quite significant. I will create another catergory called "Unknown" to represent missing values. This will help preserve the missingness information and will prevent bias towards the most frequent category.

In [36]:
```
df_train['income_poverty'].fillna('Unknown', inplace=True)
```

The rent or own column has 8% missing data. I will fill the missing values with the mode of the column.

In [37]:
```
mode_value = df_train['rent_or_own'].mode()[0]
df_train['rent_or_own'].fillna(mode_value, inplace=True)
```

The doctor_recc_seasonal column has 8% missing data. I will fill the missing values with the mode of the column.

In [38]:
```
mode_value = df_train['doctor_recc_seasonal'].mode()[0]
df_train['doctor_recc_seasonal'].fillna(mode_value, inplace=True)
```

The education column has 5.27% missing data. I will fill the missing values with the mode of the column.

In [39]:
```
mode_value = df_train['education'].mode()[0]
df_train['education'].fillna(mode_value, inplace=True)
```

The marital status column has 5.27% missing data. I will fill the missing values with the mode of the column.

In [40]:
```
mode_value = df_train['marital_status'].mode()[0]
df_train['marital_status'].fillna(mode_value, inplace=True)
```

The employment status column has 5.48% missing data. I will fill the missing values with the mode of the column.

In [41]:
```
mode_value = df_train['employment_status'].mode()[0]
df_train['employment_status'].fillna(mode_value, inplace=True)
```

In [42]:
```
#recheck the missing values percentages
missing_percentage = (df_train.isnull().mean() * 100).round(2)
missing_percentage
```

Out[42]:
```
seasonal_vaccine               0.00
doctor_recc_seasonal           0.00
chronic_med_condition          3.64
child_under_6_months           3.07
health_worker                  3.01
health_insurance               0.00
opinion_seas_vacc_effective    1.73
opinion_seas_risk              1.92
opinion_seas_sick_from_vacc    2.01
age_group                      0.00
education                      0.00
race                           0.00
sex                            0.00
income_poverty                 0.00
marital_status                 0.00
rent_or_own                    0.00
employment_status              0.00
dtype: float64
```

In [43]:
```
# I drop the rest of the missing data

df_train.dropna(axis=0, inplace=True)
```

In [44]:
```
df_train.shape
```

Out[44]:
```
(25548, 17)
```

In [45]:
```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 25548 entries, 0 to 26706
Data columns (total 17 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   seasonal_vaccine             25548 non-null  int64  
 1   doctor_recc_seasonal         25548 non-null  float64
 2   chronic_med_condition        25548 non-null  float64
 3   child_under_6_months         25548 non-null  float64
 4   health_worker                25548 non-null  float64
 5   health_insurance             25548 non-null  object 
 6   opinion_seas_vacc_effective  25548 non-null  float64
 7   opinion_seas_risk            25548 non-null  float64
 8   opinion_seas_sick_from_vacc  25548 non-null  float64
 9   age_group                    25548 non-null  object 
 10  education                    25548 non-null  object 
 11  race                         25548 non-null  object 
 12  sex                          25548 non-null  object 
 13  income_poverty               25548 non-null  object 
 14  marital_status               25548 non-null  object 
 15  rent_or_own                  25548 non-null  object 
 16  employment_status            25548 non-null  object 
dtypes: float64(7), int64(1), object(9)
memory usage: 3.5+ MB
```

Check for duplicate values in the data

In [46]:
```
#checking for duplicates in the data
df_train.loc[df_train.duplicated()]
```

Out[46]:

	seasonal_vaccine	doctor_recc_seasonal	chronic_med_condition	child_under_6_n
464	1	0.0	0.0	
655	1	1.0	0.0	
727	0	0.0	0.0	
873	1	0.0	0.0	
1020	0	0.0	0.0	
...	...	...	...	
26682	0	0.0	1.0	
26683	1	1.0	0.0	
26693	1	1.0	0.0	
26698	1	0.0	0.0	
26702	0	0.0	0.0	

2872 rows × 17 columns

There are no duplicates in the data

In [47]:
```
# Summary statistics
df_train.describe(include='object')
```

Out[47]:

	health_insurance	age_group	education	race	sex	income_poverty	marita
count	25548.0	25548	25548	25548	25548	25548	
unique	3.0	5	4	4	2	4	
top	1.0	65+ Years	College Graduate	White	Female	<= $75,000, Above Poverty	
freq	12524.0	6548	10542	20325	15208	12624	

From the summary of data we see that majority of the respondents are married female collegae graduates above 65 years old, who earn over USD 75000 per year and own their homes and have health Insurance.

Testing Features Dataset

In [48]:
```
df_test_features.shape
```
Out[48]:
```
(26708, 36)
```

In [49]:
```
# dropping the unnecessary columns from the test features  data set
columns_to_drop1 = ['respondent_id','h1n1_concern','h1n1_knowledge','dc
       'behavioral_face_mask', 'behavioral_wash_hands',
       'behavioral_large_gatherings', 'behavioral_outside_home',
       'behavioral_touch_face','census_msa', 'household_adults', 'house
df_test = df_test_features.drop(columns_to_drop1, axis=1)
```

In [50]:
```
#inspecting the shape after dropping unnecessary columns
df_test.shape
```
Out[50]:
```
(26708, 16)
```

The test data shape has 26,708 rows and 16 columns

In [51]:
```
df_test.head()
```
Out[51]:

	doctor_recc_seasonal	chronic_med_condition	child_under_6_months	health_worker
0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0
3	1.0	1.0	0.0	0.0
4	0.0	0.0	0.0	1.0

In [52]:
```
df_test.tail()
```

Out[52]:

	doctor_recc_seasonal	chronic_med_condition	child_under_6_months	health_wor
26703	1.0	0.0	0.0	
26704	0.0	0.0	0.0	
26705	0.0	0.0	0.0	
26706	0.0	0.0	1.0	
26707	0.0	0.0	0.0	

In [53]:
```
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26708 entries, 0 to 26707
Data columns (total 16 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   doctor_recc_seasonal          24548 non-null  float64
 1   chronic_med_condition         25776 non-null  float64
 2   child_under_6_months          25895 non-null  float64
 3   health_worker                 25919 non-null  float64
 4   health_insurance              14480 non-null  float64
 5   opinion_seas_vacc_effective   26256 non-null  float64
 6   opinion_seas_risk             26209 non-null  float64
 7   opinion_seas_sick_from_vacc   26187 non-null  float64
 8   age_group                     26708 non-null  object 
 9   education                     25301 non-null  object 
 10  race                          26708 non-null  object 
 11  sex                           26708 non-null  object 
 12  income_poverty                22211 non-null  object 
 13  marital_status                25266 non-null  object 
 14  rent_or_own                   24672 non-null  object 
 15  employment_status             25237 non-null  object 
dtypes: float64(8), object(8)
memory usage: 3.3+ MB
```

In [54]:
```
#check the missing values percentages
missing_percentage = (df_test.isnull().mean() * 100).round(2)
missing_percentage
```

Out[54]:
```
doctor_recc_seasonal            8.09
chronic_med_condition           3.49
child_under_6_months            3.04
health_worker                   2.95
health_insurance               45.78
opinion_seas_vacc_effective     1.69
opinion_seas_risk               1.87
opinion_seas_sick_from_vacc     1.95
age_group                       0.00
education                       5.27
race                            0.00
sex                             0.00
income_poverty                 16.84
marital_status                  5.40
rent_or_own                     7.62
employment_status               5.51
dtype: float64
```

The health insurance column has 46% missing data. This is quite significant. I will create another catergory called "Unknown" to represent missing values. This will help preserve the missingness information and will prevent bias towards the most frequent category.

In [55]:
```
df_test['health_insurance'].fillna('Unknown', inplace=True)
```

The income column has 17% missing data. This is quite significant. I will create another catergory called "Unknown" to represent missing values. This will help preserve the missingness information and will prevent bias towards the most frequent category.

In [56]:
```
df_test['income_poverty'].fillna('Unknown', inplace=True)
```

The maritals status, income_poverty,education,marital_status,rent_or_own,employment_status and doctor_recc_seasonal columns missingvalues will be filled in with the mode.

In [57]:
```
# Calculate mode for each column separately
mode_values = df_test[['marital_status', 'rent_or_own', 'education','em

# Extract mode values
mode_marital_status = mode_values['marital_status'][0]
mode_education = mode_values['education'][0]
mode_rent_or_own = mode_values['rent_or_own'][0]
mode_employment_status = mode_values['employment_status'][0]
mode_doctor_recc_seasonal = mode_values['doctor_recc_seasonal'][0]

# Fill missing values with mode for each column
df_test['marital_status'].fillna(mode_marital_status, inplace=True)
df_test['education'].fillna(mode_education, inplace=True)
df_test['rent_or_own'].fillna(mode_rent_or_own, inplace=True)
df_test['employment_status'].fillna(mode_employment_status, inplace=Tru
df_test['doctor_recc_seasonal'].fillna(mode_doctor_recc_seasonal, inpla
```

In [58]:
```
#recheck the missing values percentages
missing_percentage = (df_test.isnull().mean() * 100).round(2)
missing_percentage
```

Out[58]:
```
doctor_recc_seasonal           0.00
chronic_med_condition          3.49
child_under_6_months           3.04
health_worker                  2.95
health_insurance               0.00
opinion_seas_vacc_effective    1.69
opinion_seas_risk              1.87
opinion_seas_sick_from_vacc    1.95
age_group                      0.00
education                      0.00
race                           0.00
sex                            0.00
income_poverty                 0.00
marital_status                 0.00
rent_or_own                    0.00
employment_status              0.00
dtype: float64
```

In [59]:
```
# I drop the rest of the missing data

df_test.dropna(axis=0, inplace=True)
```

In [60]:
```
df_test.shape
```

Out[60]:
```
(25544, 16)
```

In [61]:
```
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 25544 entries, 0 to 26707
Data columns (total 16 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   doctor_recc_seasonal         25544 non-null  float64
 1   chronic_med_condition        25544 non-null  float64
 2   child_under_6_months         25544 non-null  float64
 3   health_worker                25544 non-null  float64
 4   health_insurance             25544 non-null  object 
 5   opinion_seas_vacc_effective  25544 non-null  float64
 6   opinion_seas_risk            25544 non-null  float64
 7   opinion_seas_sick_from_vacc  25544 non-null  float64
 8   age_group                    25544 non-null  object 
 9   education                    25544 non-null  object 
 10  race                         25544 non-null  object 
 11  sex                          25544 non-null  object 
 12  income_poverty               25544 non-null  object 
 13  marital_status               25544 non-null  object 
 14  rent_or_own                  25544 non-null  object 
 15  employment_status            25544 non-null  object 
dtypes: float64(7), object(9)
memory usage: 3.3+ MB
```

In [62]:
```
#checking for duplicates in the data
df_test.loc[df_test.duplicated()]
```

Out[62]:

	doctor_recc_seasonal	chronic_med_condition	child_under_6_months	health_wor
252	0.0	0.0	0.0	
435	0.0	0.0	0.0	
530	0.0	0.0	0.0	
680	0.0	1.0	0.0	
760	0.0	0.0	0.0	
...	...	...	...	
26688	0.0	0.0	0.0	
26689	1.0	0.0	0.0	
26691	0.0	0.0	0.0	
26698	0.0	0.0	0.0	
26703	1.0	0.0	0.0	

3623 rows × 16 columns

There are no duplicates in the data

Feature Understanding

Univariate Analysis

Descriptive Statistics

In [63]:
```
desc_stats = df_train.describe(include='all')
print(desc_stats)
```

```
        seasonal_vaccine  doctor_recc_seasonal  chronic_med_condition  
\
count       25548.000000          25548.000000           25548.000000  
unique               NaN                   NaN                    NaN  
top                  NaN                   NaN                    NaN  
freq                 NaN                   NaN                    NaN  
mean            0.468726              0.306130               0.283427  
std             0.499031              0.460893               0.450671  
min             0.000000              0.000000               0.000000  
25%             0.000000              0.000000               0.000000  
50%             0.000000              0.000000               0.000000  
75%             1.000000              1.000000               1.000000  
max             1.000000              1.000000               1.000000  

        child_under_6_months  health_worker  health_insurance  \
count           25548.000000   25548.000000           25548.0   
unique                   NaN            NaN               3.0   
top                      NaN            NaN               1.0   
freq                     NaN            NaN           12524.0   
mean                0.082707       0.112259               NaN   
std                 0.275444       0.315691               NaN   
min                 0.000000       0.000000               NaN   
25%                 0.000000       0.000000               NaN   
50%                 0.000000       0.000000               NaN   
75%                 0.000000       0.000000               NaN   
max                 1.000000       1.000000               NaN   

        opinion_seas_vacc_effective  opinion_seas_risk  \
count                  25548.000000       25548.000000   
unique                          NaN                NaN   
top                             NaN                NaN   
freq                            NaN                NaN   
mean                       4.032723           2.724988   
std                        1.082148           1.386114   
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  age_group          education   rac
e  \
count                  25548.000000      25548              25548  2554
8   
unique                          NaN          5                  4      
4   
top                             NaN  65+ Years  College Graduate  Whit
e   
freq                            NaN       6548              10542  2032
5   
mean                       2.117739        NaN                NaN    Na
N   
std                        1.333259        NaN                NaN    Na
N   
min                        1.000000        NaN                NaN    Na
N   
25%                        1.000000        NaN                NaN    Na
N   
50%                        2.000000        NaN                NaN    Na
N   
75%                        4.000000        NaN                NaN    Na
```

```
N
max                          5.000000        NaN                NaN    Na
N

          sex            income_poverty marital_status rent_or_own  
\
count    25548                     25548          25548       25548  
unique       2                         4              2           2  
top     Female  <= $75,000, Above Poverty        Married         Own  
freq     15208                     12624          13977       19703  
mean       NaN                       NaN            NaN         NaN  
std        NaN                       NaN            NaN         NaN  
min        NaN                       NaN            NaN         NaN  
25%        NaN                       NaN            NaN         NaN  
50%        NaN                       NaN            NaN         NaN  
75%        NaN                       NaN            NaN         NaN  
max        NaN                       NaN            NaN         NaN  

       employment_status  
count              25548  
unique                 3  
top             Employed  
freq               14021  
mean                 NaN  
std                  NaN  
min                  NaN  
25%                  NaN  
50%                  NaN  
75%                  NaN  
max                  NaN  
```

Vaccination and Health-Related Information

Seasonal Vaccine: On average, 46.9% of respondents reported having received the seasonal vaccine, with a standard deviation of 0.499 indicating a near-equal split between those who have and have not been vaccinated.

Doctor Recommendation for Seasonal Vaccine: Only 30.6% of respondents received a recommendation from their doctor to get the seasonal vaccine, with a standard deviation of 0.461 showing variability in doctor's advice.

Chronic Medical Condition: About 28.3% of respondents reported having a chronic medical condition, with a standard deviation of 0.451.

Child Under 6 Months: A small percentage, 8.3%, of respondents have a child under 6 months, indicated by a standard deviation of 0.275.

Health Worker: 11.2% of the respondents are health workers, with a standard deviation of 0.316.

Health Insurance: A majority of respondents, 12,524 out of 25,548, reported having health insurance, reflecting a high level of insurance coverage among the participants.

Opinions

Effectiveness of Seasonal Vaccine: The average opinion on the effectiveness of the seasonal vaccine is quite positive, with a mean score of 4.033 out of 5, and a standard deviation of 1.082.

Risk Perception of Seasonal Vaccine: The perception of risk associated with the seasonal vaccine has a mean score of 2.725, suggesting a moderate level of concern, with a standard deviation of 1.386.

Perception of Getting Sick from Vaccine: The mean score for the perception of getting sick from the vaccine is 2.118, indicating a generally low to moderate concern, with a standard deviation of 1.333.

Demographics

Age Group: The most common age group among respondents is 65+ years, suggesting a significant portion of the survey population is older adults.

Education: The highest education level reported by respondents is College Graduate, indicating a relatively educated sample.

Race: The majority of respondents identify as White, highlighting the racial composition of the survey population.

Sex: 59.5% of respondents are female, showing a higher participation rate among women.

Income Poverty Level: The most common income level is ≤ $75,000, Above Poverty, indicating that a substantial number of respondents are above the poverty line but not necessarily affluent.

Marital Status: 54.7% of respondents are married, suggesting that over half of the participants live with a spouse.

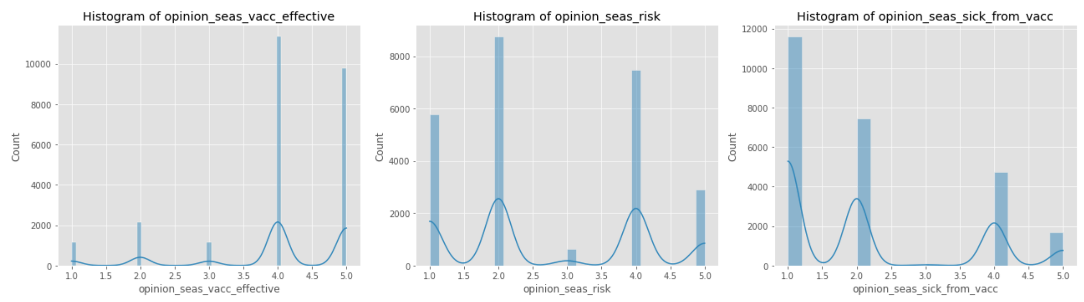
Home Ownership: 77.1% of respondents own their homes, reflecting a high rate of homeownership.

Employment Status: 54.9% of respondents are employed, showing that over half of the survey population is currently working.

Visualization

Plotting the distribution to inspect the distributions

In [64]:
```
#histograms for the numerical variables

numerical_cols = ['opinion_seas_vacc_effective', 'opinion_seas_risk', '

fig, axes = plt.subplots(1, 3, figsize=(18, 5))
for i, col in enumerate(numerical_cols):
    sns.histplot(df_train[col], ax=axes[i], kde=True)
    axes[i].set_title(f'Histogram of {col}')

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



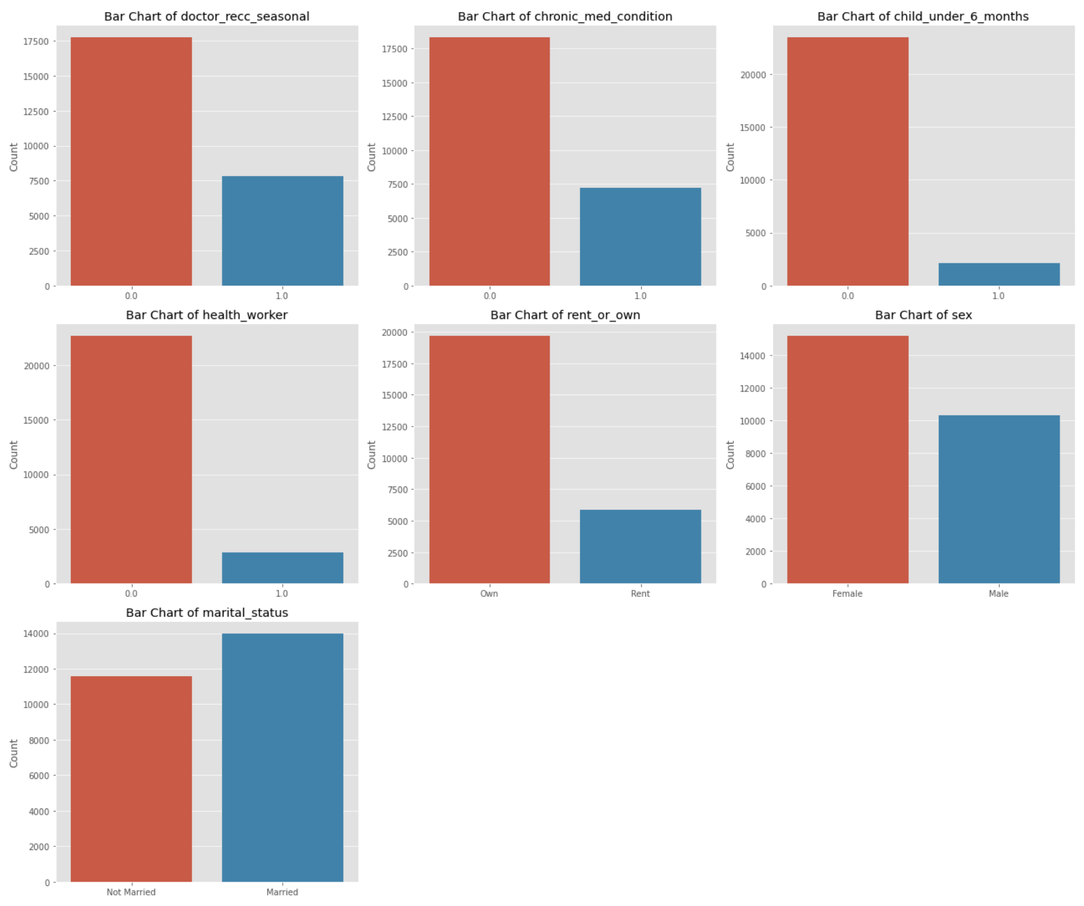
Effectiveness: Most respondents trust the vaccine's effectiveness.

Risk: Opinions on the risk of not getting vaccinated are varied, with a significant portion recognizing some level of risk.

Sickness: The majority believe that the vaccine does not cause sickness.

In [65]:

```
# List of categorical columns
categorical_cols = ['doctor_recc_seasonal', 'chronic_med_condition', 'c

# Determine the number of rows needed based on the number of categorica
num_cols = 3  # Number of columns per row in the plot grid
num_rows = len(categorical_cols) // num_cols + (len(categorical_cols) %

fig, axes = plt.subplots(num_rows, num_cols, figsize=(18, num_rows * 5)

# Flatten axes array if there are multiple rows
if num_rows > 1:
    axes = axes.flatten()
else:
    axes = [axes]

for i, col in enumerate(categorical_cols):
    sns.countplot(x=df_train[col], ax=axes[i])
    axes[i].set_title(f'Bar Chart of {col}')
    axes[i].set_xlabel('')
    axes[i].set_ylabel('Count')

# Remove any unused subplots
for j in range(i+1, len(axes)):
    fig.delaxes(axes[j])

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



Doctor Recommendations: More respondents did not receive a recommendation for the seasonal vaccine than those who did.

Chronic Medical Conditions: The majority of respondents do not have chronic medical conditions.

Children Under 6 Months: Most respondents do not have children under 6 months.

Health Workers: A small portion of respondents are health workers.

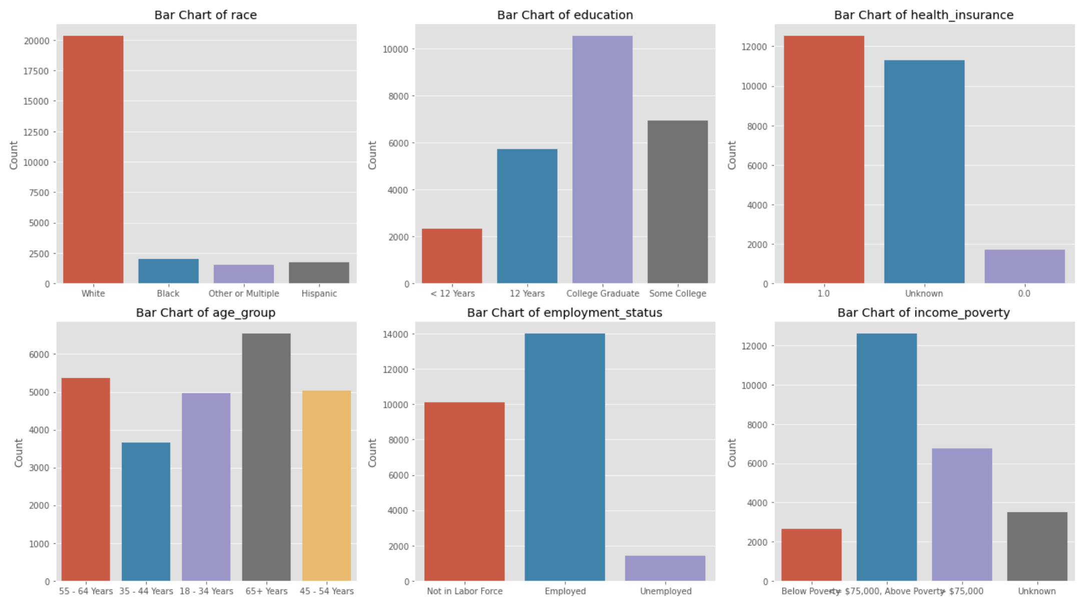
Home Ownership: The majority of respondents own their homes.

Sex: There are more female respondents than male respondents.

Marital Status: More respondents are married compared to those who are not married.

In [66]:

```python
# List of categorical columns
categorical_cols = ['race','education','health_insurance','age_group','

# Determine the number of rows needed based on the number of categorica
num_cols = 3  # Number of columns per row in the plot grid
num_rows = len(categorical_cols) // num_cols + (len(categorical_cols) %

fig, axes = plt.subplots(num_rows, num_cols, figsize=(18, num_rows * 5)

# Flatten axes array if there are multiple rows
if num_rows > 1:
    axes = axes.flatten()
else:
    axes = [axes]

for i, col in enumerate(categorical_cols):
    sns.countplot(x=df_train[col], ax=axes[i])
    axes[i].set_title(f'Bar Chart of {col}')
    axes[i].set_xlabel('')
    axes[i].set_ylabel('Count')

# Remove any unused subplots
for j in range(i+1, len(axes)):
    fig.delaxes(axes[j])

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



Race: The majority of respondents are White, with smaller representations from Black, Hispanic, and other races.

Education: Most respondents are college graduates, with significant groups having some college or 12 years of education.

Health Insurance: Many respondents have health insurance, though a substantial number have unknown insurance status.

Age Group: Respondents are distributed across various age groups, with the largest group being 65 years and older.

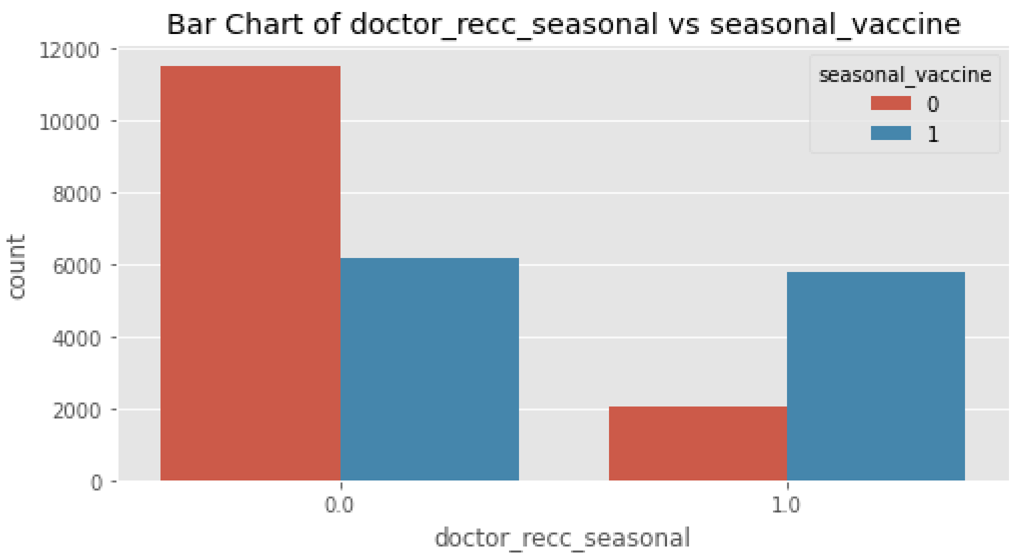
Employment Status: The majority of respondents are employed, with significant groups not in the labor force.

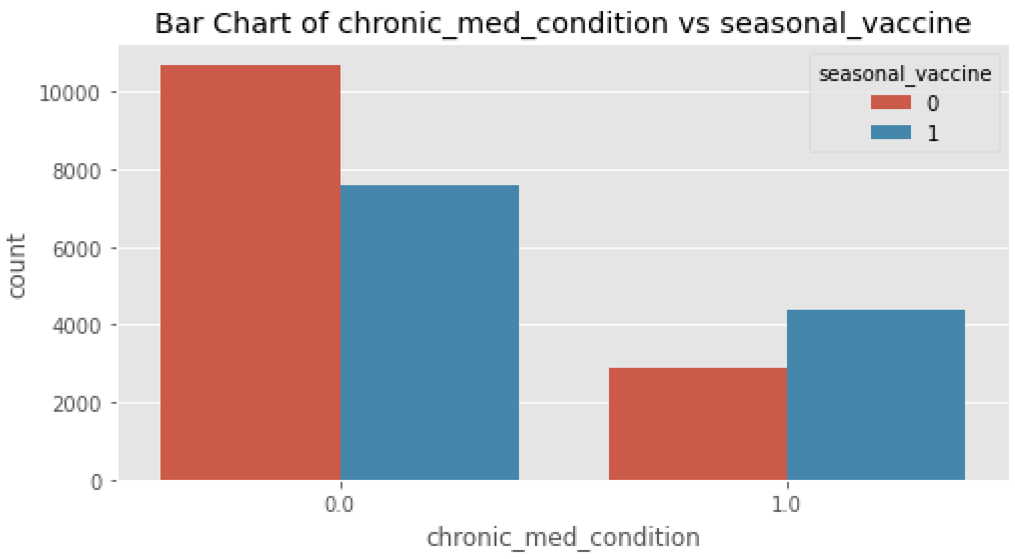
Income and Poverty: Most respondents have an income above the poverty line, with a

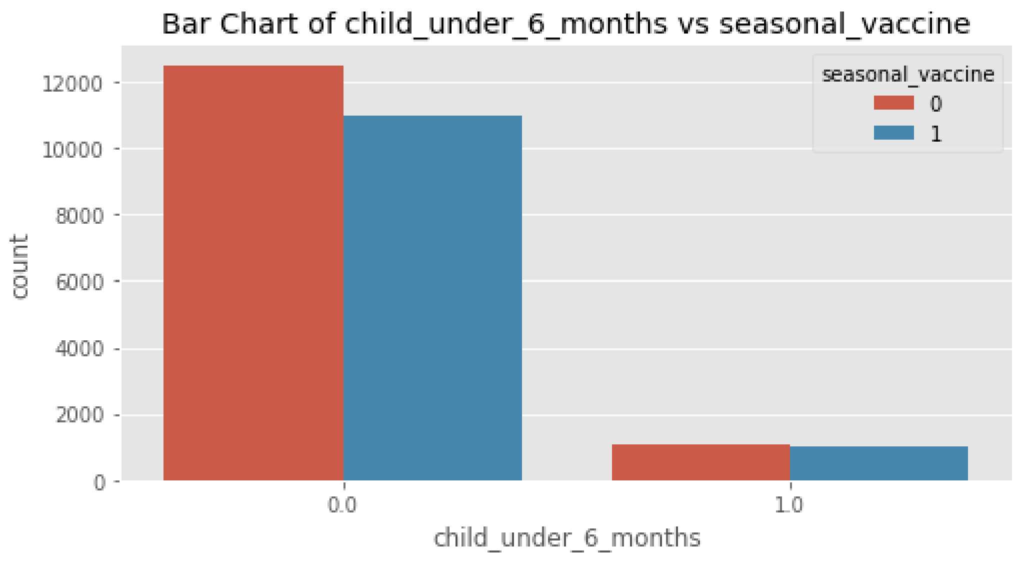
Bivariate Analysis

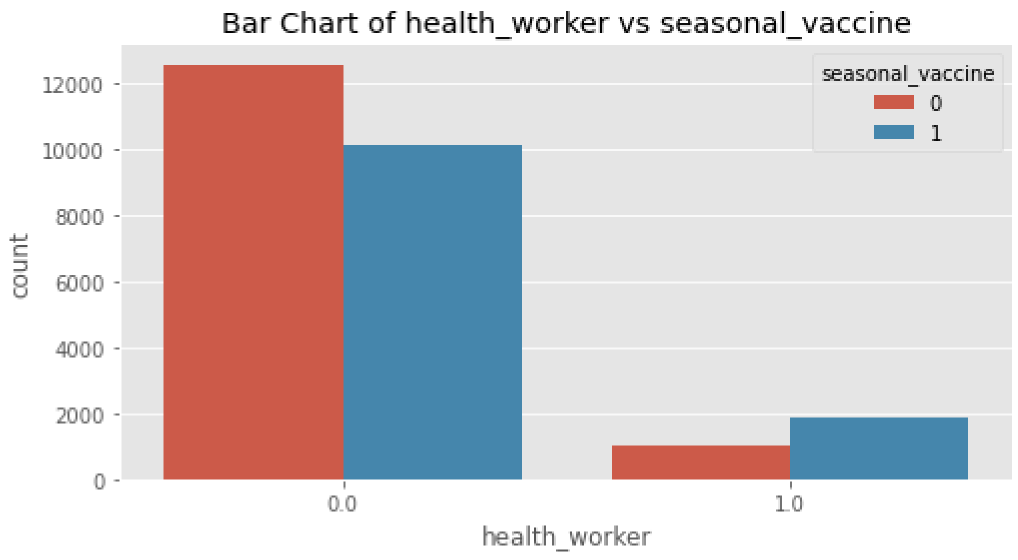
Chi-square Test and Bar Charts for Categorical Variables:

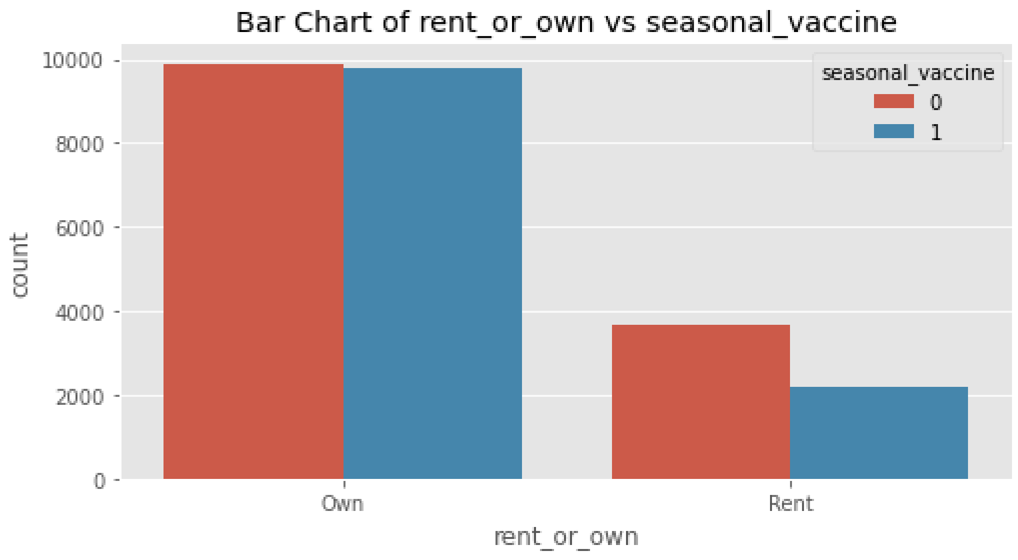
In [67]:
```
from scipy.stats import chi2_contingency

target = 'seasonal_vaccine'
categorical_cols = ['doctor_recc_seasonal', 'chronic_med_condition', 'c


chi2_results = {}
for col in categorical_cols:
    crosstab = pd.crosstab(df_train[target], df_train[col])
    chi2, p, _, _ = chi2_contingency(crosstab)
    chi2_results[col] = {'chi2': chi2, 'p-value': p}

    # Plotting bar chart
    plt.figure(figsize=(8, 4))
    sns.countplot(x=col, hue=target, data=df_train)
    plt.title(f'Bar Chart of {col} vs {target}')
    plt.show()

print(chi2_results)
```

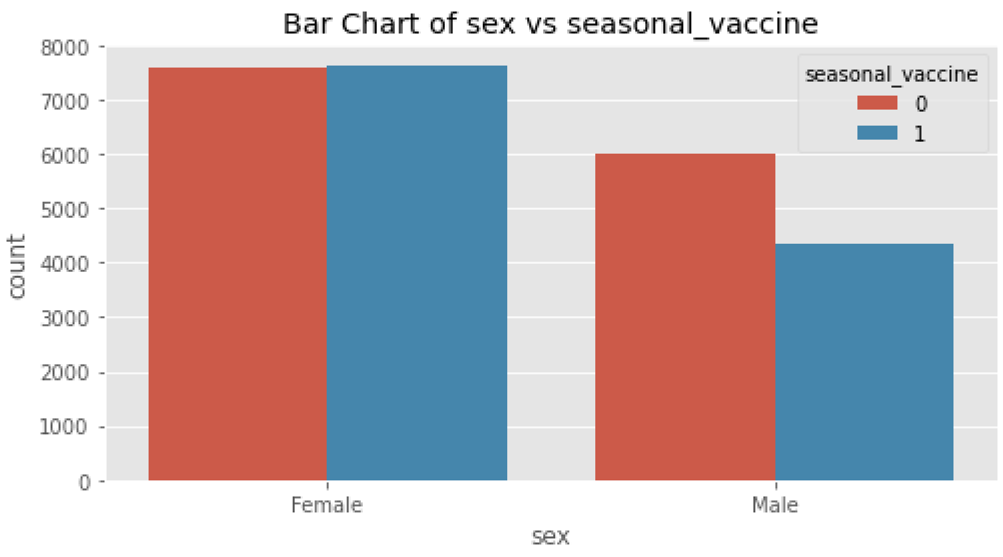


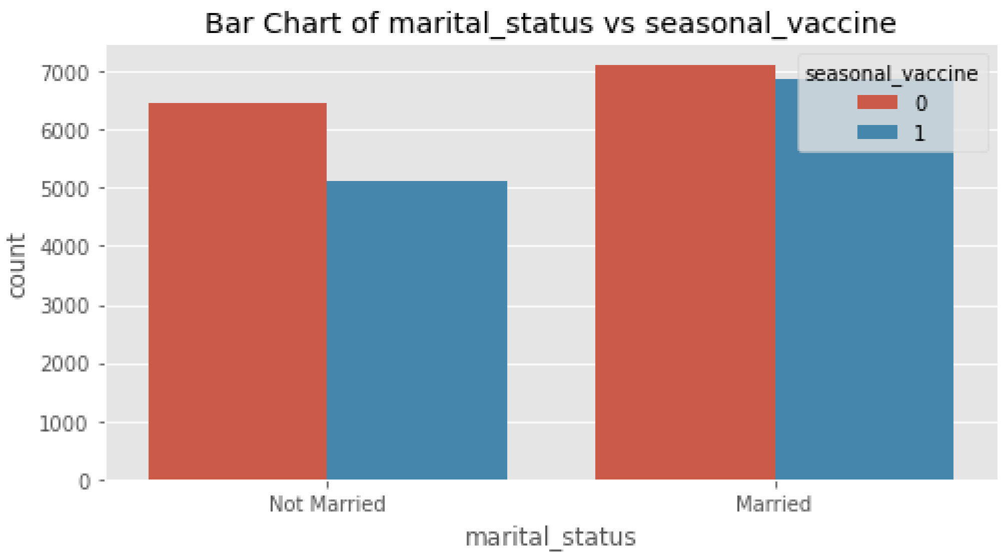


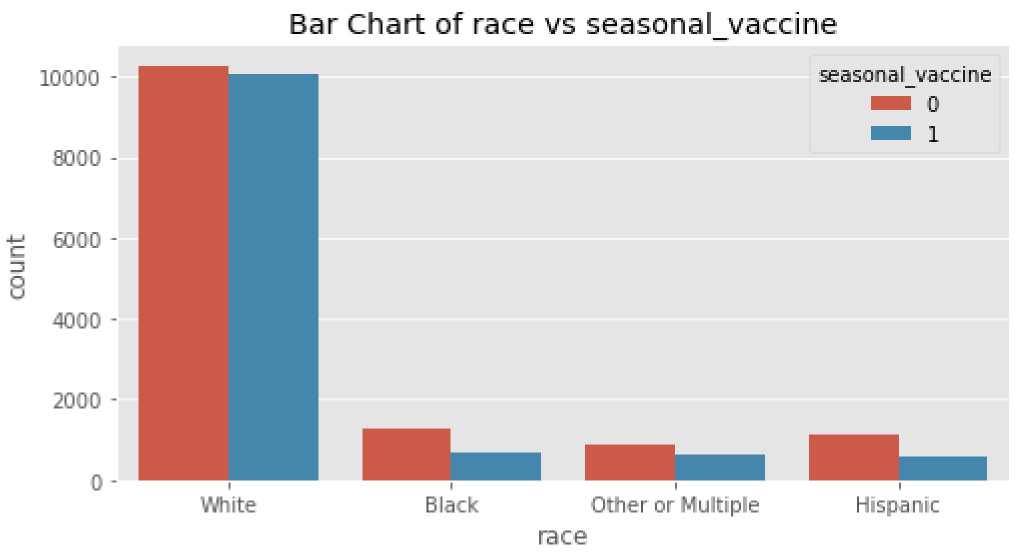

Bar Chart of child_under_6_months vs seasonal_vaccine
count
12000
10000
8000
6000
4000
2000
0
0.0
1.0
child_under_6_months
seasonal_vaccine
0
1

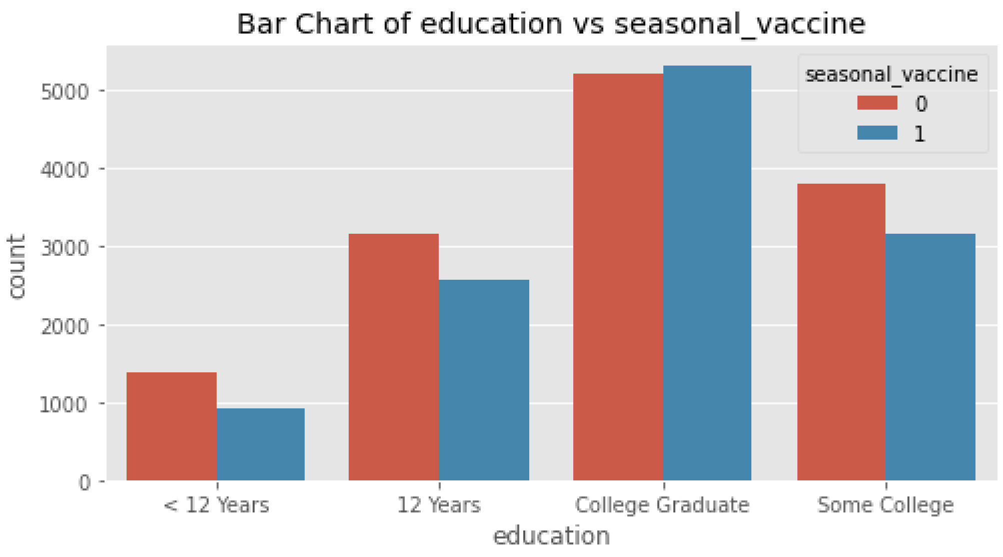

Bar Chart of health_worker vs seasonal_vaccine
count
12000
10000
8000
6000
4000
2000
0
0.0
1.0
health_worker
seasonal_vaccine
0
1

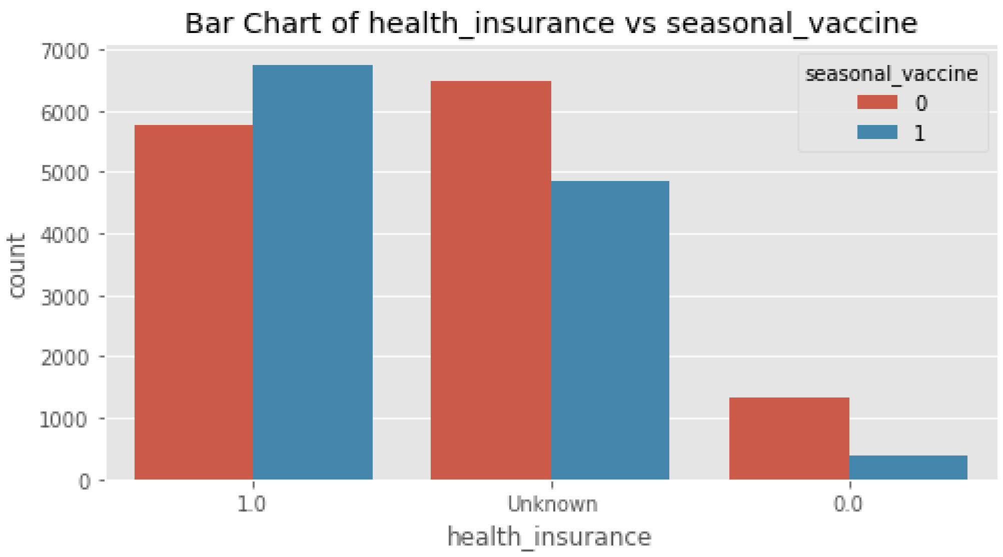

Bar Chart of rent_or_own vs seasonal_vaccine
count
10000
8000
6000
4000
2000
0
Own
Rent
rent_or_own
seasonal_vaccine
0
1

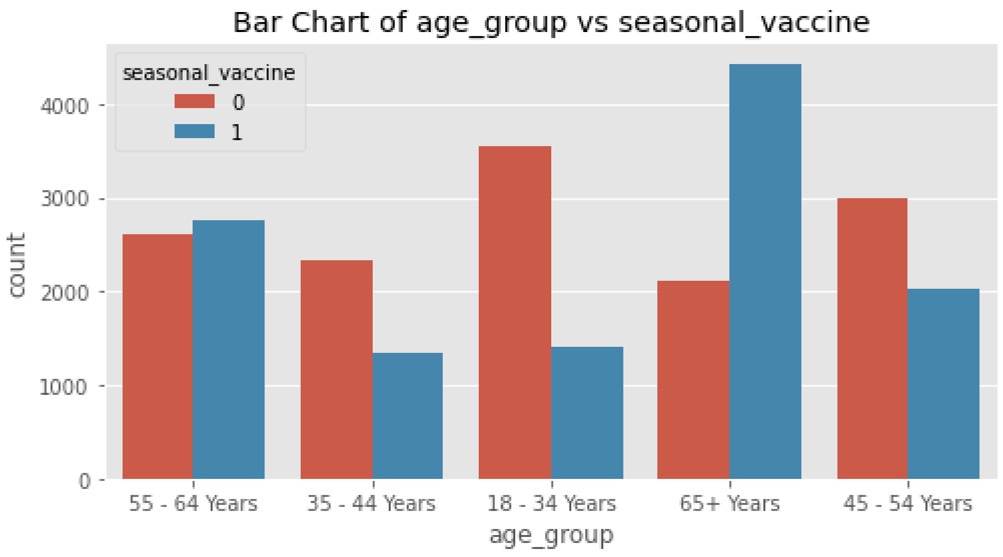

Bar Chart of sex vs seasonal_vaccine
count
8000
7000
6000
5000
4000
3000
2000
1000
0
Female
Male
sex
seasonal_vaccine
0
1

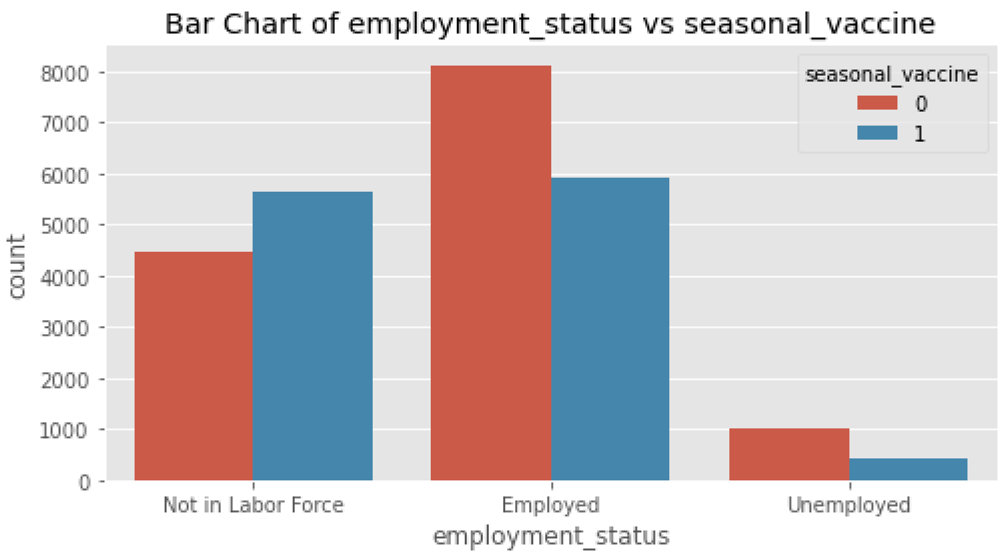

Bar Chart of marital_status vs seasonal_vaccine
count
7000
6000
5000
4000
3000
2000
1000
0
Not Married
Married
marital_status
seasonal_vaccine
0
1

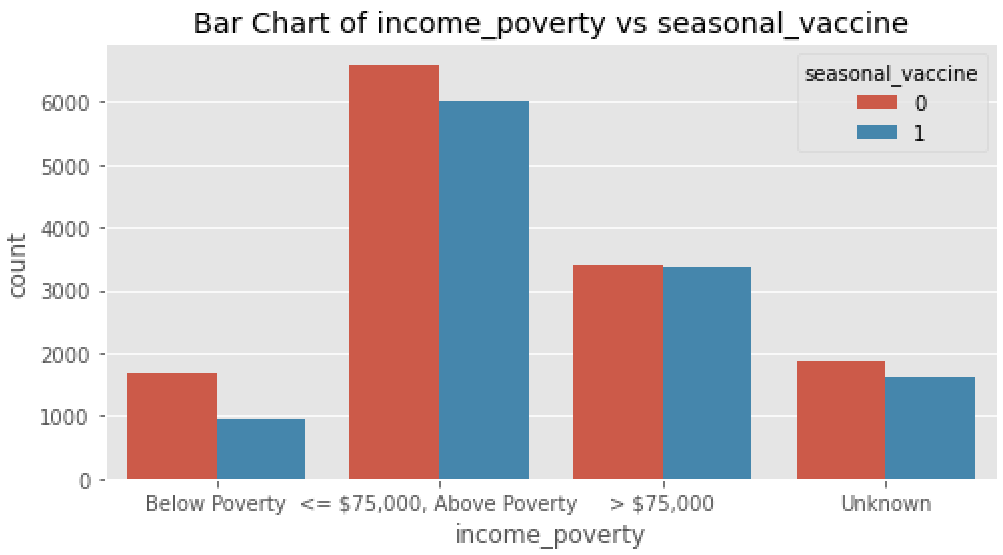

Bar Chart of race vs seasonal_vaccine
count
10000
8000
6000
4000
2000
0
White
Black
Other or Multiple
Hispanic
race
seasonal_vaccine
0
1


Bar Chart of education vs seasonal_vaccine
seasonal_vaccine
0
1
count
0
1000
2000
3000
4000
5000
< 12 Years
12 Years
College Graduate
Some College
education


Bar Chart of health_insurance vs seasonal_vaccine
seasonal_vaccine
0
1
count
0
1000
2000
3000
4000
5000
6000
7000
1.0
Unknown
0.0
health_insurance


Bar Chart of age_group vs seasonal_vaccine
seasonal_vaccine
0
1
count
0
1000
2000
3000
4000
55 - 64 Years
35 - 44 Years
18 - 34 Years
65+ Years
45 - 54 Years
age_group





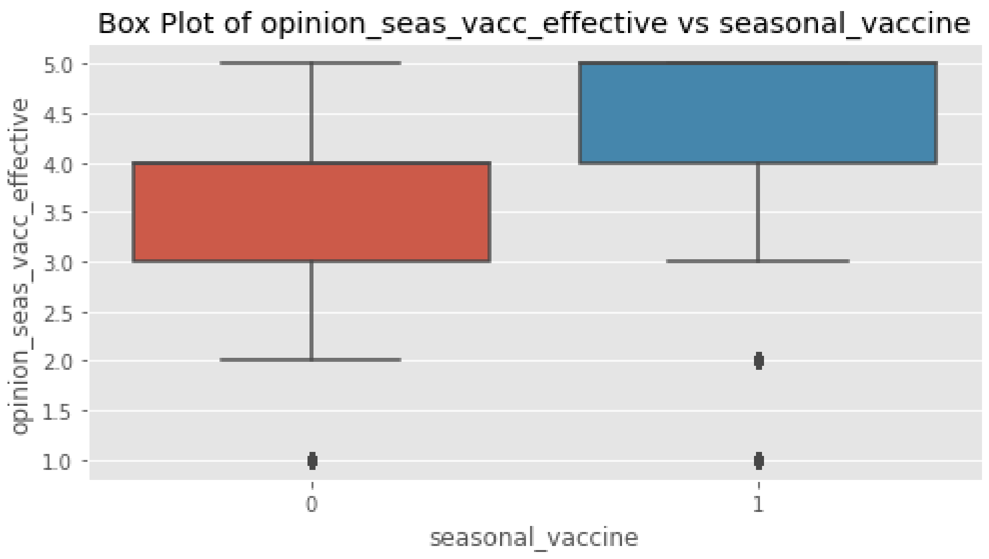
```
{'doctor_recc_seasonal': {'chi2': 3311.9818202503693, 'p-value': 0.0},
'chronic_med_condition': {'chi2': 743.9811952483569, 'p-value': 8.1679
56551825321e-164}, 'child_under_6_months': {'chi2': 3.845613193831469
3, 'p-value': 0.049876280833552986}, 'health_worker': {'chi2': 415.395
95310862586, 'p-value': 2.4523617534961894e-92}, 'rent_or_own': {'chi
2': 290.6564940284849, 'p-value': 3.5769490769516457e-65}, 'sex': {'ch
i2': 163.83765561828633, 'p-value': 1.641446895402072e-37}, 'marital_s
tatus': {'chi2': 58.6723574019464, 'p-value': 1.8623976527439038e-14},
'race': {'chi2': 292.51050976850024, 'p-value': 4.155639137471919e-6
3}, 'education': {'chi2': 115.80595373845645, 'p-value': 6.17397615715
2281e-25}, 'health_insurance': {'chi2': 721.769205974287, 'p-value':
1.8612654631534875e-157}, 'age_group': {'chi2': 2095.2565380007263, 'p
-value': 0.0}, 'employment_status': {'chi2': 613.3257124985383, 'p-val
ue': 6.576786235279549e-134}, 'income_poverty': {'chi2': 145.721407842
31055, 'p-value': 2.2062275253336927e-31}}
```

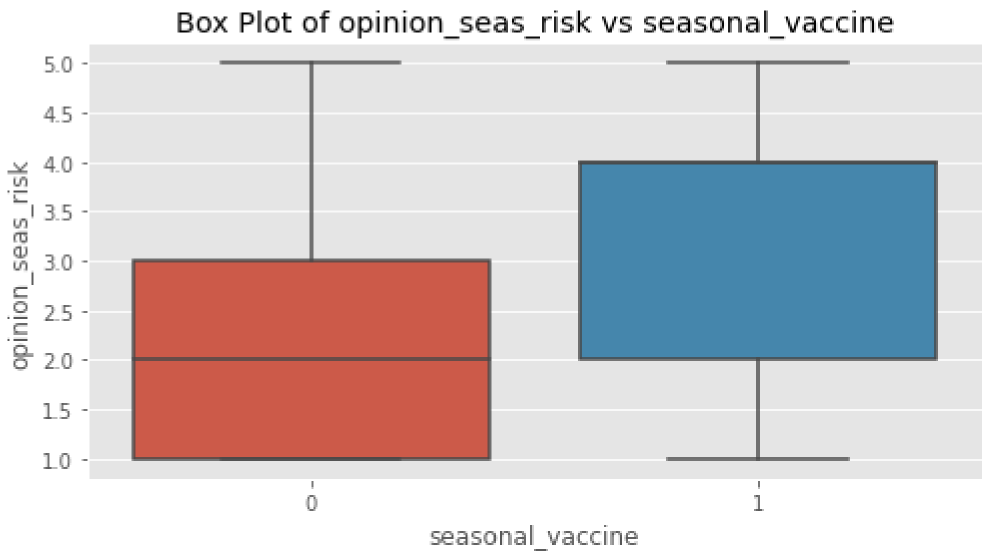
Strongest Predictors: Variables like doctor recommendations, chronic medical condition, health worker status, employment status, health insurance, and age group have very strong and highly significant relationships with receiving the seasonal vaccine.

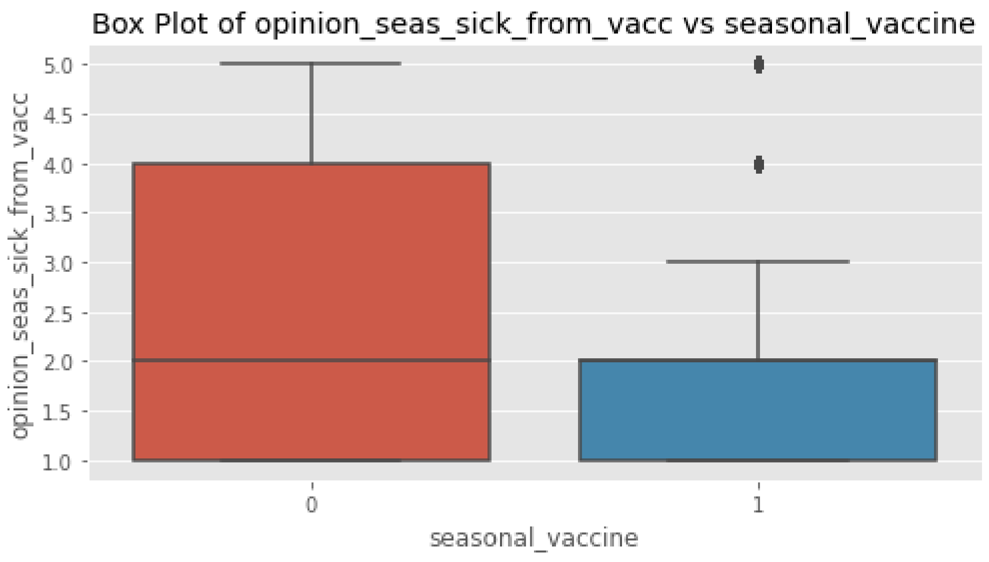
Significant Predictors: Variables like gender, marital status, education level, and income level also show significant relationships with vaccine uptake.

T-tests and Box Plots for Numerical Variables:

In [68]:
```python
from scipy.stats import ttest_ind

numerical_cols = ['opinion_seas_vacc_effective', 'opinion_seas_risk', '
t_test_results = {}
for col in numerical_cols:
    vaccinated = df_train[df_train[target] == 1][col].dropna()
    not_vaccinated = df_train[df_train[target] == 0][col].dropna()
    t_stat, p_val = ttest_ind(vaccinated, not_vaccinated)
    t_test_results[col] = {'t-statistic': t_stat, 'p-value': p_val}

    # Plotting box plot
    plt.figure(figsize=(8, 4))
    sns.boxplot(x=target, y=col, data=df_train)
    plt.title(f'Box Plot of {col} vs {target}')
    plt.show()

print(t_test_results)
```







```
{'opinion_seas_vacc_effective': {'t-statistic': 62.38327858617024, 'p-
value': 0.0}, 'opinion_seas_risk': {'t-statistic': 68.35048043088781,
'p-value': 0.0}, 'opinion_seas_sick_from_vacc': {'t-statistic': -10.12
8114493268148, 'p-value': 4.603224898348334e-24}}
```

Vaccine Effectiveness: Those who received the vaccine believe it is significantly more effective than those who did not receive it.

Risk Perception: Those who received the vaccine perceive a significantly higher risk from not getting vaccinated compared to those who did not receive it.

Sickness Concern: Those who received the vaccine are significantly less concerned about getting sick from the vaccine compared to those who did not receive it.

Feature relationships

In [69]:
```
df_train.columns
```

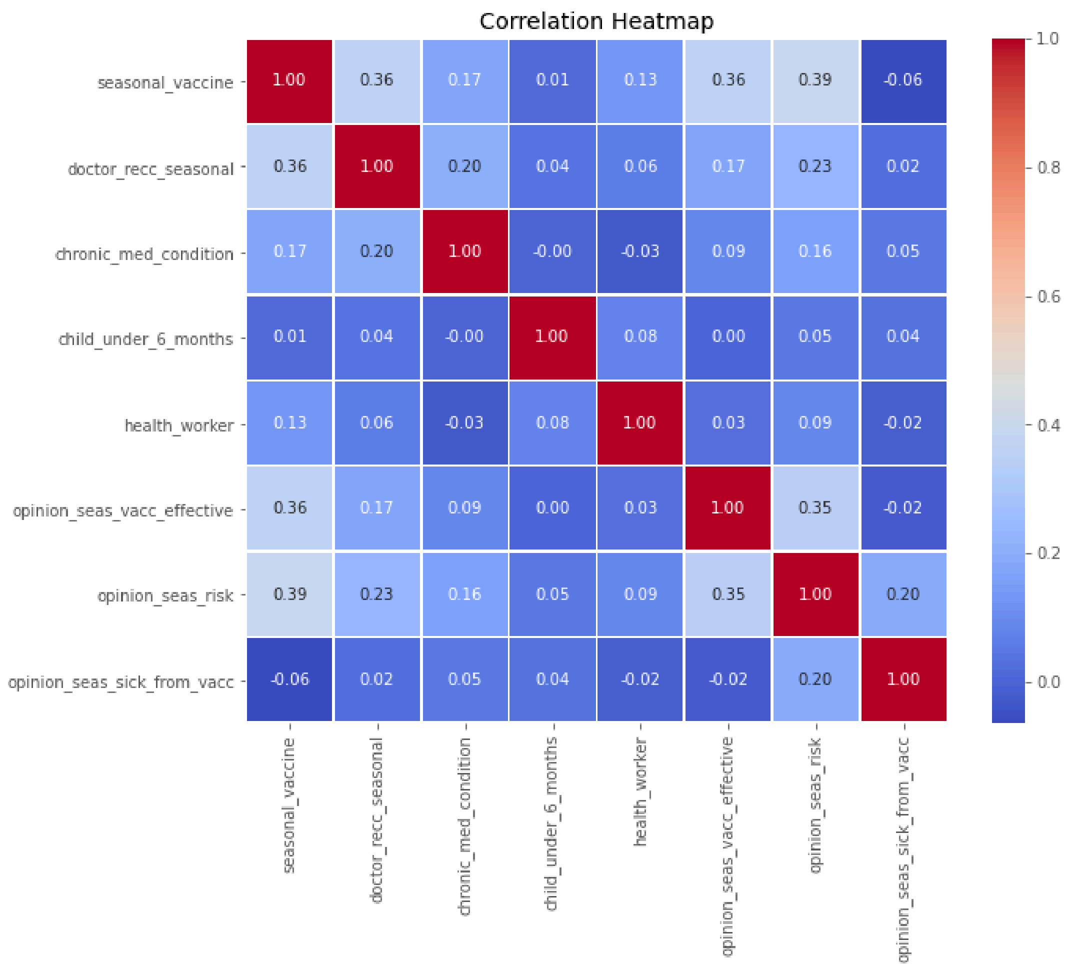
Out[69]:
```
Index(['seasonal_vaccine', 'doctor_recc_seasonal', 'chronic_med_condit
ion',
       'child_under_6_months', 'health_worker', 'health_insurance',
       'opinion_seas_vacc_effective', 'opinion_seas_risk',
       'opinion_seas_sick_from_vacc', 'age_group', 'education', 'rac
e', 'sex',
       'income_poverty', 'marital_status', 'rent_or_own', 'employment_
status'],
      dtype='object')
```

In [70]:
```
#check the correlation

df_train.corr()
```

Out[70]:

	seasonal_vaccine	doctor_recc_seasonal	chronic_med_cor
seasonal_vaccine	1.000000	0.360137	0.1
doctor_recc_seasonal	0.360137	1.000000	0.2
chronic_med_condition	0.170736	0.204153	1.0
child_under_6_months	0.012411	0.037046	-0.0
health_worker	0.127637	0.058115	-0.0
opinion_seas_vacc_effective	0.363594	0.174080	0.0
opinion_seas_risk	0.393197	0.229394	0.1
opinion_seas_sick_from_vacc	-0.063241	0.023707	0.0

In [71]:
```
# Calculate correlation matrix
correlation_matrix = df_train.corr()

# Plot correlation heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f",
plt.title('Correlation Heatmap')
plt.show()
```



Multivariate Analysis

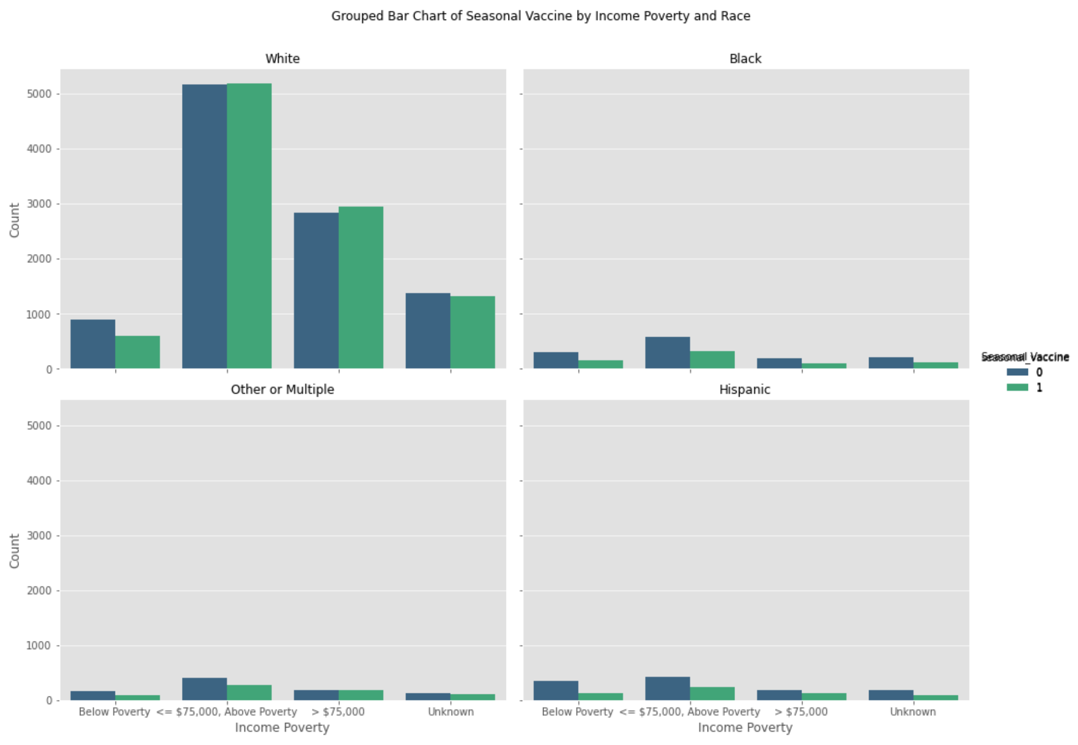
In [72]:

```
for column in df_train.columns:
    print(f"Counts for column '{column}':")
    print(df_train[column].value_counts())
    print()
```

```
Counts for column 'seasonal_vaccine':
0    13573
1    11975
Name: seasonal_vaccine, dtype: int64

Counts for column 'doctor_recc_seasonal':
0.0    17727
1.0     7821
Name: doctor_recc_seasonal, dtype: int64

Counts for column 'chronic_med_condition':
0.0    18307
1.0     7241
Name: chronic_med_condition, dtype: int64

Counts for column 'child_under_6_months':
0.0    23435
1.0     2113
Name: child_under_6_months, dtype: int64

Counts for column 'health_worker':
0.0    22680
1.0     2868
Name: health_worker, dtype: int64

Counts for column 'health_insurance':
1.0        12524
Unknown    11308
0.0         1716
Name: health_insurance, dtype: int64

Counts for column 'opinion_seas_vacc_effective':
4.0    11346
5.0     9755
2.0     2146
1.0     1163
3.0     1138
Name: opinion_seas_vacc_effective, dtype: int64

Counts for column 'opinion_seas_risk':
2.0    8751
4.0    7473
1.0    5786
5.0    2912
3.0     626
Name: opinion_seas_risk, dtype: int64

Counts for column 'opinion_seas_sick_from_vacc':
1.0    11599
2.0     7441
4.0     4737
5.0     1681
3.0       90
Name: opinion_seas_sick_from_vacc, dtype: int64

Counts for column 'age_group':
65+ Years        6548
55 - 64 Years    5362
45 - 54 Years    5024
18 - 34 Years    4956
35 - 44 Years    3658
```

```
Name: age_group, dtype: int64

Counts for column 'education':
College Graduate    10542
Some College         6956
12 Years             5723
< 12 Years           2327
Name: education, dtype: int64

Counts for column 'race':
White                20325
Black                 2004
Hispanic              1706
Other or Multiple     1513
Name: race, dtype: int64

Counts for column 'sex':
Female    15208
Male      10340
Name: sex, dtype: int64

Counts for column 'income_poverty':
<= $75,000, Above Poverty    12624
> $75,000                     6762
Unknown                       3503
Below Poverty                 2659
Name: income_poverty, dtype: int64

Counts for column 'marital_status':
Married        13977
Not Married    11571
Name: marital_status, dtype: int64

Counts for column 'rent_or_own':
Own     19703
Rent     5845
Name: rent_or_own, dtype: int64

Counts for column 'employment_status':
Employed              14021
Not in Labor Force    10093
Unemployed             1434
Name: employment_status, dtype: int64
```

In [73]:
```
#import seaborn as sns
import matplotlib.pyplot as plt

# Create a FacetGrid for income_poverty and race
g = sns.catplot(
    data=df_train,
    x='income_poverty',
    hue='seasonal_vaccine',
    col='race',
    kind='count',
    height=5,
    aspect=1.2,
    col_wrap=2,  # Adjust this based on the number of categories in 'ra
    palette='viridis'
)

g.set_axis_labels("Income Poverty", "Count")
g.set_titles("{col_name}")
g.add_legend(title="Seasonal Vaccine")
plt.subplots_adjust(top=0.9)
g.fig.suptitle('Grouped Bar Chart of Seasonal Vaccine by Income Poverty
plt.show()
```



White: Shows a balanced distribution in the middle-income categories, with a higher vaccination rate in the > $75,000 category.

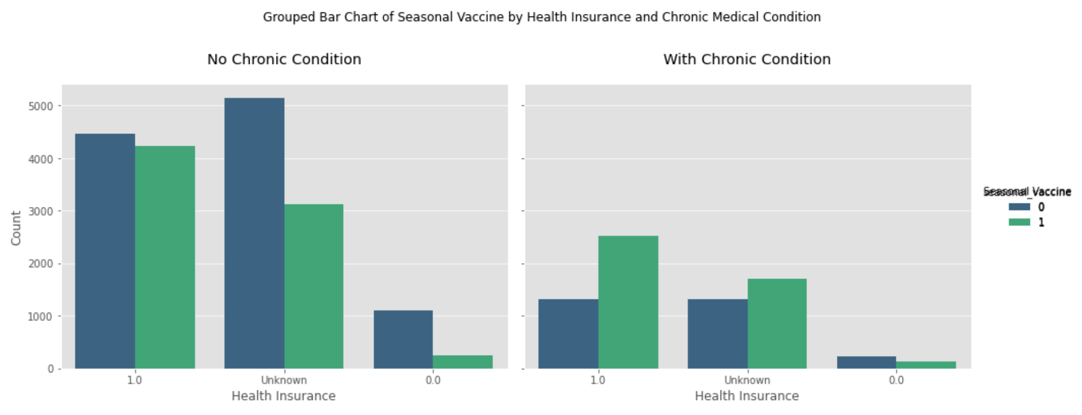
Black: Generally shows a higher number of non-vaccinated individuals across all income categories.

Other or Multiple: Low counts across all categories, with a slightly higher number of non-vaccinated individuals.

Hispanic: Also shows a higher number of non-vaccinated individuals across all income categories.

Income levels and race both influence the likelihood of receiving the seasonal vaccine. For most races, individuals below the poverty line and those with unknown income status are less likely to be vaccinated. However, for White individuals in higher income categories,

In [74]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Create a mapping for the column titles
title_mapping = {0.0: "No Chronic Condition", 1.0: "With Chronic Condit

# Create a FacetGrid for health_insurance and chronic_med_condition
g = sns.catplot(
    data=df_train,
    x='health_insurance',
    hue='seasonal_vaccine',
    col='chronic_med_condition',
    kind='count',
    height=5,
    aspect=1.2,
    col_wrap=2,  # Adjust this based on the number of categories in 'ch
    palette='viridis'
)

g.set_axis_labels("Health Insurance", "Count")
g.set_titles(col_template="\n{col_name}", row_template="{row_name}")
for ax, title in zip(g.axes.flat, g.col_names):
    ax.set_title(title_mapping[float(title)], pad=20)  # Adds space bet

g.add_legend(title="Seasonal Vaccine")
plt.subplots_adjust(top=0.85)  # Adjust top to create space for the tit
g.fig.suptitle('Grouped Bar Chart of Seasonal Vaccine by Health Insuran
plt.show()
```



No Chronic Medical Condition (Left Panel):

For individuals without a chronic medical condition, being insured or having unknown insurance status is associated with higher counts of not receiving the vaccine. Uninsured individuals are fewer, but the trend remains that more did not receive the vaccine.

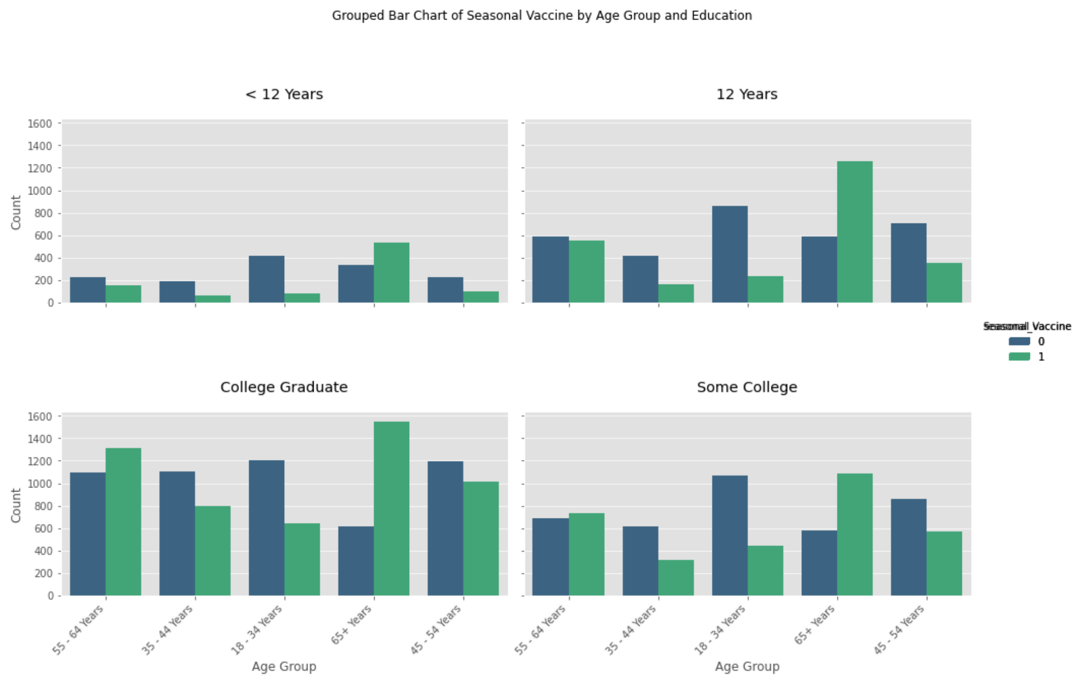
With Chronic Medical Condition (Right Panel):

For individuals with a chronic medical condition, being insured is associated with higher vaccination rates compared to those without insurance or with unknown insurance status. This group shows a stronger inclination towards receiving the vaccine if they have health insurance.

Health insurance status and the presence of chronic medical conditions influence the likelihood of receiving the seasonal vaccine. Having health insurance seems to be a significant factor, particularly for individuals with chronic medical conditions, in increasing the likelihood of vaccination.

In [75]:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Create a FacetGrid for age_group and education
g = sns.catplot(
    data=df_train,
    x='age_group',
    hue='seasonal_vaccine',
    col='education',
    kind='count',
    height=5,
    aspect=1.2,
    col_wrap=2,  # Adjust this based on the number of categories in 'ea
    palette='viridis'
)

g.set_axis_labels("Age Group", "Count")
g.set_titles(col_template="\n{col_name}", row_template="{row_name}")
for ax, title in zip(g.axes.flat, g.col_names):
    ax.set_title(title, pad=20)  # Adds space between title and plot
    ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')

g.add_legend(title="Seasonal Vaccine")
plt.subplots_adjust(top=0.85, bottom=0.2, hspace=0.6)  # Adjust top, bo
g.fig.suptitle('Grouped Bar Chart of Seasonal Vaccine by Age Group and
plt.show()
```



< 12 Years: Lower education levels show higher counts of non-vaccinated individuals in most age groups, except for those 65+ years.

12 Years: More balanced distribution, with certain age groups showing higher vaccination rates.

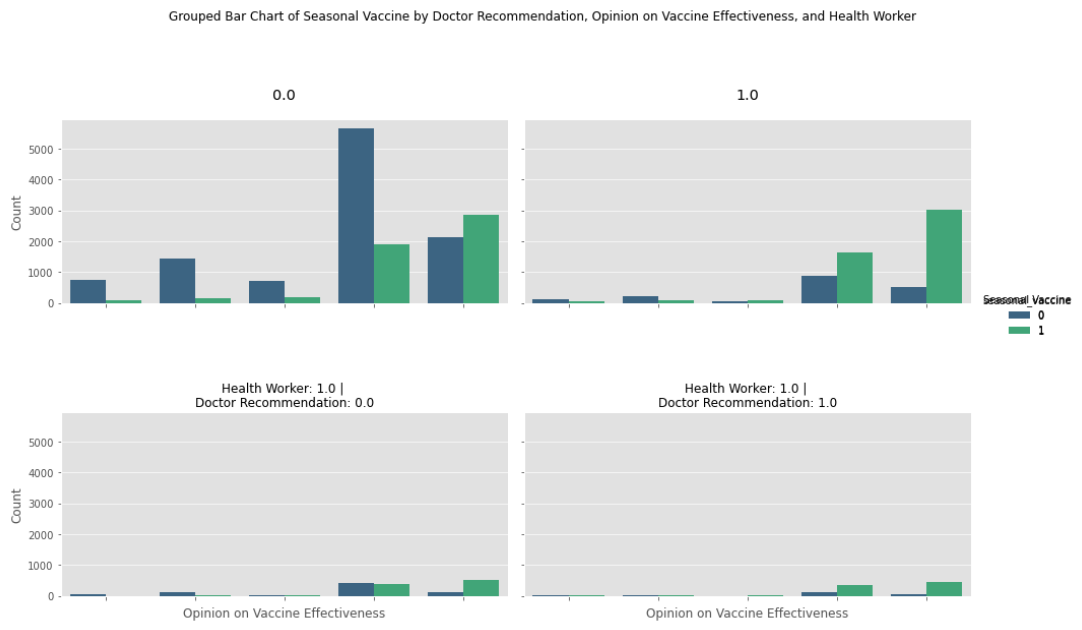
College Graduate: Generally higher vaccination rates across all age groups, especially in the older age groups.

Some College: Mixed results, with some age groups showing higher non-vaccination rates, and others showing higher vaccination rates.

These visualizations highlight the impact of both age and education on vaccination rates. Higher education levels, especially college graduates, tend to show higher vaccination

In [76]:

```
# Create a mapping for the column titles
# title_mapping = {0.0: "No Doctor Recommendation", 1.0: "Doctor Recomm

# Create a FacetGrid for Doctor_recc_seasonal, opinion_seas_vacc_effect
g = sns.catplot(
    data=df_train,
    x='opinion_seas_vacc_effective',
    hue='seasonal_vaccine',
    col='doctor_recc_seasonal',
    row='health_worker',
    kind='count',
    height=5,
    aspect=1.2,
    palette='viridis'
)

g.set_axis_labels("Opinion on Vaccine Effectiveness", "Count")
g.set_titles(col_template="\nDoctor Recommendation: {col_name}", row_te
for ax, title in zip(g.axes.flat, g.col_names):
    ax.set_title(title, pad=20)  # Adds space between title and plot
    ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')

g.add_legend(title="Seasonal Vaccine")
plt.subplots_adjust(top=0.85, bottom=0.2, hspace=0.6)  # Adjust top, bo
g.fig.suptitle('Grouped Bar Chart of Seasonal Vaccine by Doctor Recomme
plt.show()
```



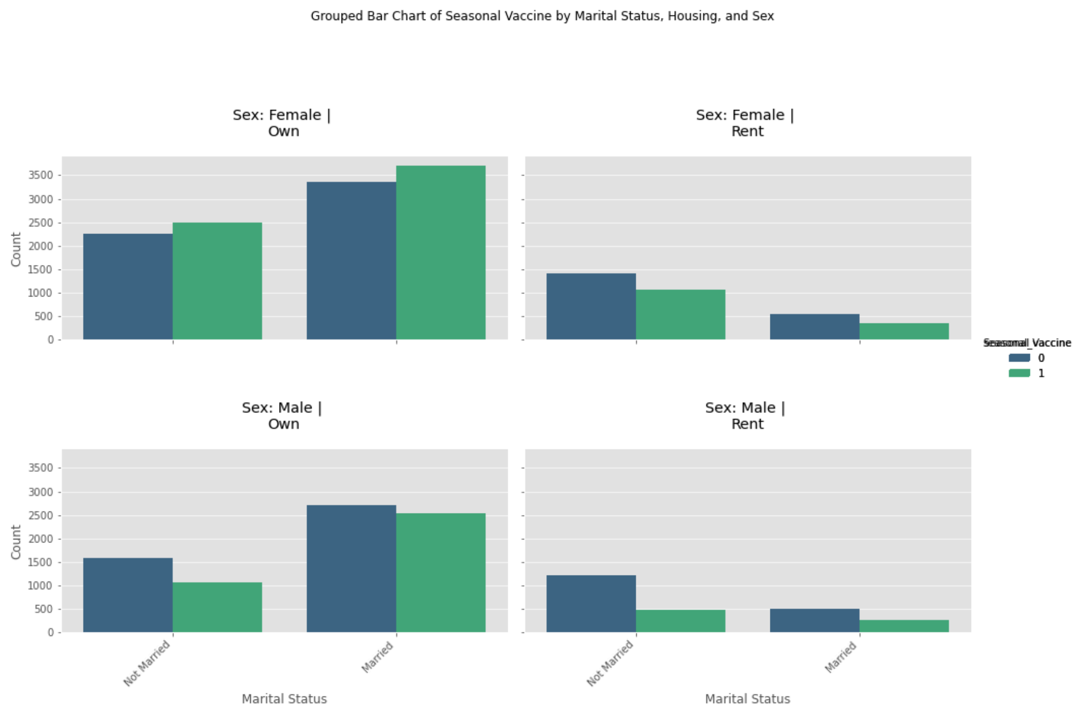
Doctor Recommendations: Significantly influence vaccination rates, especially for those with higher opinions of vaccine effectiveness.

Opinion on Vaccine Effectiveness: Higher opinions correlate with higher vaccination rates across all groups, more pronounced with doctor recommendations.

Health Workers: Show higher vaccination rates overall, particularly when combined with positive opinions and doctor recommendations.

These visualizations emphasize the importance of healthcare professional recommendations and personal beliefs in vaccine effectiveness in influencing vaccination uptake. Health workers, generally being more informed, tend to follow recommendations more consistently, and their opinions strongly align with their vaccination decisions.

In [77]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Create a FacetGrid for marital_status, rent_or_own, and sex
g = sns.catplot(
    data=df_train,
    x='marital_status',
    hue='seasonal_vaccine',
    col='rent_or_own',
    row='sex',
    kind='count',
    height=5,
    aspect=1.2,
    palette='viridis'
)

g.set_axis_labels("Marital Status", "Count")
g.set_titles(col_template="\n{col_name}", row_template="Sex: {row_name}
for ax in g.axes.flat:
    ax.set_title(ax.get_title(), pad=20)  # Adds space between title an
    ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')

g.add_legend(title="Seasonal Vaccine")
plt.subplots_adjust(top=0.85, bottom=0.2, hspace=0.6)  # Adjust top, bo
g.fig.suptitle('Grouped Bar Chart of Seasonal Vaccine by Marital Status
plt.show()
```



Housing Status: Individuals who own their homes tend to have a higher vaccination rate compared to those who rent, regardless of sex and marital status.

Marital Status: Married individuals generally show higher vaccination rates compared to non-married individuals, with the trend being more pronounced among females.

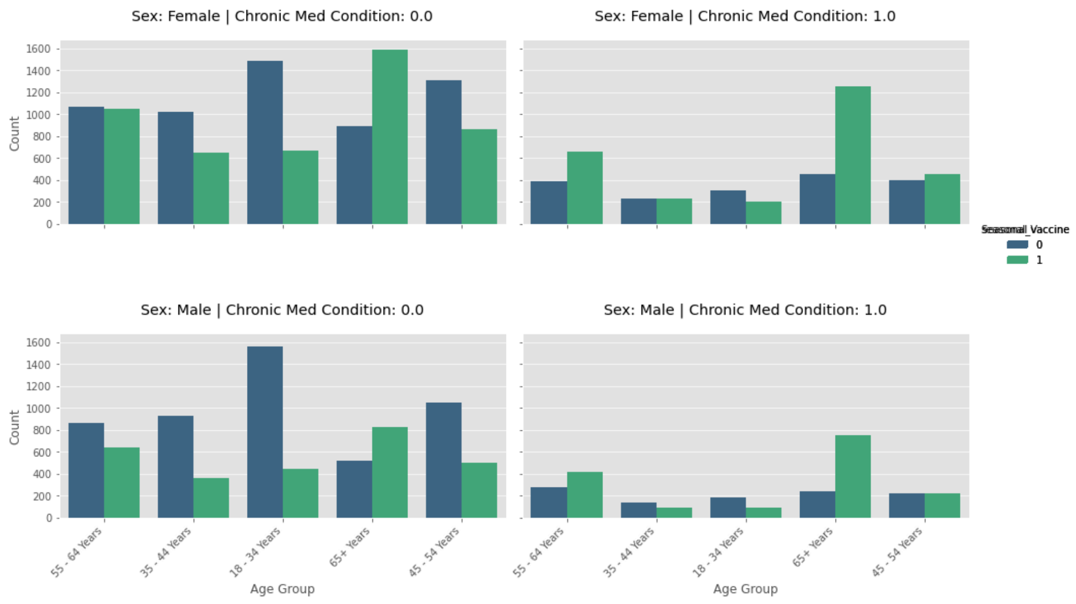
Sex: Females generally show a higher vaccination rate compared to males, especially among those who own their homes.

The graphs highlight the influence of housing status, marital status, and sex on vaccination rates. Homeownership and being married are associated with higher vaccination rates, with females showing a generally higher inclination towards getting vaccinated compared to males.

In [78]:

```
# Create a FacetGrid for chronic_med_condition, age_group, and sex
g = sns.catplot(
    data=df_train,
    x='age_group',
    hue='seasonal_vaccine',
    col='chronic_med_condition',
    row='sex',
    kind='count',
    height=5,
    aspect=1.2,
    palette='viridis'
)

g.set_axis_labels("Age Group", "Count")
g.set_titles(col_template="Chronic Med Condition: {col_name}", row_temp
for ax in g.axes.flat:
    ax.set_title(ax.get_title(), pad=20)  # Adds space between title an
    ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')

g.add_legend(title="Seasonal Vaccine")
plt.subplots_adjust(top=0.85, bottom=0.2, hspace=0.6)  # Adjust top, bo
g.fig.suptitle('Grouped Bar Chart of Seasonal Vaccine by Chronic Medica
plt.show()
```

Grouped Bar Chart of Seasonal Vaccine by Chronic Medical Condition, Age Group, and Sex



Chronic Medical Condition:

Individuals with chronic medical conditions (1.0) tend to have lower vaccination rates in younger age groups (18-44 years) for both sexes. Older age groups (65+ years) with chronic conditions have higher vaccination rates, especially among females.

Age Group:

Vaccination rates generally increase with age. Younger age groups (18-34 years) have lower vaccination rates, particularly among males without chronic conditions.

Sex:

Females generally show higher vaccination rates compared to males, especially in older age groups (65+ years) and those without chronic medical conditions.

These visualizations highlight the influence of age, chronic medical conditions, and sex on vaccination rates. Vaccination rates tend to be higher among older individuals and those without chronic medical conditions, with females showing a higher inclination towards getting vaccinated compared to males.

Model Training

This is a classification problem, I start by performing one hot encoding, train test splitting and then selecting a few classification models to train

Perform one hot encoding on the data

In [79]:

```
import pandas as pd

# Columns to perform one-hot encoding on
columns_to_encode = ['opinion_seas_vacc_effective', 'health_insurance',

# Perform one-hot encoding
df_train_encoded = pd.get_dummies(df_train, columns=columns_to_encode)

# Display the encoded DataFrame
print(df_train_encoded.head())
```

```
   seasonal_vaccine  doctor_recc_seasonal  chronic_med_condition  \
0                 0                   0.0                    0.0   
1                 1                   0.0                    0.0   
2                 0                   0.0                    1.0   
3                 1                   1.0                    1.0   
4                 0                   0.0                    0.0   

   child_under_6_months  health_worker  opinion_seas_vacc_effective_1.
0  \
0                   0.0            0.0                                
0  
1                   0.0            0.0                                
0  
2                   0.0            0.0                                
0  
3                   0.0            0.0                                
0  
4                   0.0            0.0                                
0  

   opinion_seas_vacc_effective_2.0  opinion_seas_vacc_effective_3.0  \
0                                1                                0   
1                                0                                0   
2                                0                                0   
3                                0                                0   
4                                0                                1   

   opinion_seas_vacc_effective_4.0  opinion_seas_vacc_effective_5.0  \
0                                0                                0   
1                                1                                0   
2                                1                                0   
3                                0                                1   
4                                0                                0   

   health_insurance_0.0  health_insurance_1.0  health_insurance_Unknow
n  \
0                     0                     1                         
0  
1                     0                     1                         
0  
2                     0                     0                         
1  
3                     0                     0                         
1  
4                     0                     0                         
1  

   opinion_seas_risk_1.0  opinion_seas_risk_2.0  opinion_seas_risk_3.0
\
0                      1                      0                      0   
1                      0                      1                      0   
2                      1                      0                      0   
3                      0                      0                      0   
4                      1                      0                      0   

   opinion_seas_risk_4.0  opinion_seas_risk_5.0  marital_status_Marrie
d  \
0                      0                      0                       
0  
1                      0                      0                       
0  
```

```
2                      0                       0
0
3                      1                       0
0
4                      0                       0
1

   marital_status_Not Married  rent_or_own_Own  rent_or_own_Rent  sex_
Female  \
0                           1                1                 0
1
1                           1                0                 1
0
2                           1                1                 0
0
3                           1                0                 1
1
4                           0                1                 0
1

   sex_Male  opinion_seas_sick_from_vacc_1.0  opinion_seas_sick_from_v
acc_2.0  \
0         0                                0
1
1         1                                0
0
2         1                                0
1
3         0                                1
0
4         0                                0
0

   opinion_seas_sick_from_vacc_3.0  opinion_seas_sick_from_vacc_4.0  \
0                                0                                0
1                                0                                1
2                                0                                0
3                                0                                0
4                                0                                1

   opinion_seas_sick_from_vacc_5.0  education_12 Years  education_< 12
Years  \
0                                0                   0
1
1                                0                   1
0
2                                0                   0
0
3                                0                   1
0
4                                0                   0
0

   education_College Graduate  education_Some College  \
0                           0                       0
1                           0                       0
2                           1                       0
3                           0                       0
4                           0                       1

   income_poverty_<= $75,000, Above Poverty  income_poverty_> $75,000
```

```
\
0                                         0                         0
1                                         0                         0
2                                         1                         0
3                                         0                         0
4                                         1                         0

   income_poverty_Below Poverty  income_poverty_Unknown  \
0                             1                       0
1                             1                       0
2                             0                       0
3                             1                       0
4                             0                       0

   age_group_18 - 34 Years  age_group_35 - 44 Years  age_group_45 - 54
Years  \
0                        0                        0
0
1                        0                        1
0
2                        1                        0
0
3                        0                        0
0
4                        0                        0
1

   age_group_55 - 64 Years  age_group_65+ Years  race_Black  race_Hisp
anic  \
0                        1                    0           0
0
1                        0                    0           0
0
2                        0                    0           0
0
3                        0                    1           0
0
4                        0                    0           0
0

   race_Other or Multiple  race_White  employment_status_Employed  \
0                       0           1                           0
1                       0           1                           1
2                       0           1                           1
3                       0           1                           0
4                       0           1                           1

   employment_status_Not in Labor Force  employment_status_Unemployed
0                                     1                             0
1                                     0                             0
2                                     0                             0
3                                     1                             0
4                                     0                             0
```

In [80]:
```
df_train_encoded.head()
```

Out[80]:

	seasonal_vaccine	doctor_recc_seasonal	chronic_med_condition	child_under_6_montl
0	0	0.0	0.0	C
1	1	0.0	0.0	C
2	0	0.0	1.0	C
3	1	1.0	1.0	C
4	0	0.0	0.0	C

In [81]:
```
df_train_encoded.corr()
```

Out[81]:

	seasonal_vaccine	doctor_recc_seasonal	chronic_med_
seasonal_vaccine	1.000000	0.360137	
doctor_recc_seasonal	0.360137	1.000000	
chronic_med_condition	0.170736	0.204153	
child_under_6_months	0.012411	0.037046	
health_worker	0.127637	0.058115	
opinion_seas_vacc_effective_1.0	-0.136644	-0.054200	
opinion_seas_vacc_effective_2.0	-0.197627	-0.082959	
opinion_seas_vacc_effective_3.0	-0.081521	-0.077549	
opinion_seas_vacc_effective_4.0	-0.163889	-0.084839	
opinion_seas_vacc_effective_5.0	0.373687	0.190315	
health_insurance_0.0	-0.129208	-0.067626	
health_insurance_1.0	0.137713	0.107205	
health_insurance_Unknown	-0.073487	-0.073817	
opinion_seas_risk_1.0	-0.283549	-0.147367	
opinion_seas_risk_2.0	-0.136827	-0.094303	
opinion_seas_risk_3.0	0.029719	-0.001998	
opinion_seas_risk_4.0	0.253498	0.139886	
opinion_seas_risk_5.0	0.200438	0.135641	
marital_status_Married	0.048001	0.022730	
marital_status_Not Married	-0.048001	-0.022730	
rent_or_own_Own	0.106756	0.030596	
rent_or_own_Rent	-0.106756	-0.030596	
sex_Female	0.080161	0.073083	
sex_Male	-0.080161	-0.073083	
opinion_seas_sick_from_vacc_1.0	0.084483	0.009246	
opinion_seas_sick_from_vacc_2.0	-0.056071	-0.042976	
opinion_seas_sick_from_vacc_3.0	-0.033342	-0.016558	
opinion_seas_sick_from_vacc_4.0	-0.009154	0.027068	
opinion_seas_sick_from_vacc_5.0	-0.044586	0.021716	
education_12 Years	-0.021544	0.014263	
education_< 12 Years	-0.042997	-0.013390	
education_College Graduate	0.061292	-0.004868	
education_Some College	-0.019816	0.000680	
income_poverty_<= $75,000, Above Poverty	0.017384	0.017398	
income_poverty_> $75,000	0.034044	-0.004630	
income_poverty_Below Poverty	-0.072267	-0.008065	

	seasonal_vaccine	doctor_recc_seasonal	chronic_med_
income_poverty_Unknown	-0.004776	-0.012190	
age_group_18 - 34 Years	-0.180910	-0.093682	
age_group_35 - 44 Years	-0.085008	-0.047967	
age_group_45 - 54 Years	-0.062534	-0.047221	
age_group_55 - 64 Years	0.047517	0.035983	
age_group_65+ Years	0.244658	0.132757	
race_Black	-0.069237	0.005849	
race_Hispanic	-0.068387	-0.028325	
race_Other or Multiple	-0.023654	-0.004740	
race_White	0.102334	0.016408	
employment_status_Employed	-0.105296	-0.102274	
employment_status_Not in Labor Force	0.145391	0.118521	
employment_status_Unemployed	-0.081157	-0.030621	

Split the data into dependent and independent features

In [82]:
```python
# Extract features (X) and target variable (y)
X = df_train_encoded.drop(columns=['seasonal_vaccine'])
y = df_train_encoded['seasonal_vaccine']

# Display the features (X) and target variable (y)
print("Features (X):")
print(X.head())  # Display the first few rows of features
print("\nTarget Variable (y):")
print(y.head())  # Display the first few rows of the target variable
```

```
Features (X):
   doctor_recc_seasonal  chronic_med_condition  child_under_6_months  
\
0                   0.0                    0.0                   0.0   
1                   0.0                    0.0                   0.0   
2                   0.0                    1.0                   0.0   
3                   1.0                    1.0                   0.0   
4                   0.0                    0.0                   0.0   

   health_worker  opinion_seas_vacc_effective_1.0  \
0            0.0                                0   
1            0.0                                0   
2            0.0                                0   
3            0.0                                0   
4            0.0                                0   

   opinion_seas_vacc_effective_2.0  opinion_seas_vacc_effective_3.0  \
0                                1                                0   
1                                0                                0   
2                                0                                0   
3                                0                                0   
4                                0                                1   

   opinion_seas_vacc_effective_4.0  opinion_seas_vacc_effective_5.0  \
0                                0                                0   
1                                1                                0   
2                                1                                0   
3                                0                                1   
4                                0                                0   

   health_insurance_0.0  health_insurance_1.0  health_insurance_Unknow
n  \
0                     0                     1                        
0   
1                     0                     1                        
0   
2                     0                     0                        
1   
3                     0                     0                        
1   
4                     0                     0                        
1   

   opinion_seas_risk_1.0  opinion_seas_risk_2.0  opinion_seas_risk_3.0  
\
0                      1                      0                      0   
1                      0                      1                      0   
2                      1                      0                      0   
3                      0                      0                      0   
4                      1                      0                      0   

   opinion_seas_risk_4.0  opinion_seas_risk_5.0  marital_status_Marrie
d  \
0                      0                      0                       
0   
1                      0                      0                       
0   
2                      0                      0                       
0   
3                      1                      0                       
0   
```

```
4                      0                      0
1

   marital_status_Not Married  rent_or_own_Own  rent_or_own_Rent  sex_
Female  \
0                           1                1                 0
1
1                           1                0                 1
0
2                           1                1                 0
0
3                           1                0                 1
1
4                           0                1                 0
1

   sex_Male  opinion_seas_sick_from_vacc_1.0  opinion_seas_sick_from_v
acc_2.0  \
0         0                                0
1
1         1                                0
0
2         1                                0
1
3         0                                1
0
4         0                                0
0

   opinion_seas_sick_from_vacc_3.0  opinion_seas_sick_from_vacc_4.0  \
0                                0                                0
1                                0                                1
2                                0                                0
3                                0                                0
4                                0                                1

   opinion_seas_sick_from_vacc_5.0  education_12 Years  education_< 12
Years  \
0                                0                   0
1
1                                0                   1
0
2                                0                   0
0
3                                0                   1
0
4                                0                   0
0

   education_College Graduate  education_Some College  \
0                           0                       0
1                           0                       0
2                           1                       0
3                           0                       0
4                           0                       1

   income_poverty_<= $75,000, Above Poverty  income_poverty_> $75,000
\
0                                         0                         0
1                                         0                         0
2                                         1                         0
```

```
3                                        0                         0
4                                        1                         0

   income_poverty_Below Poverty  income_poverty_Unknown  \
0                             1                       0
1                             1                       0
2                             0                       0
3                             1                       0
4                             0                       0

   age_group_18 - 34 Years  age_group_35 - 44 Years  age_group_45 - 54
Years  \
0                        0                        0
0
1                        0                        1
0
2                        1                        0
0
3                        0                        0
0
4                        0                        0
1

   age_group_55 - 64 Years  age_group_65+ Years  race_Black  race_Hisp
anic  \
0                        1                    0           0
0
1                        0                    0           0
0
2                        0                    0           0
0
3                        0                    1           0
0
4                        0                    0           0
0

   race_Other or Multiple  race_White  employment_status_Employed  \
0                       0           1                           0
1                       0           1                           1
2                       0           1                           1
3                       0           1                           0
4                       0           1                           1

   employment_status_Not in Labor Force  employment_status_Unemployed
0                                     1                             0
1                                     0                             0
2                                     0                             0
3                                     1                             0
4                                     0                             0

Target Variable (y):
0    0
1    1
2    0
3    1
4    0
Name: seasonal_vaccine, dtype: int64
```

Perform the Train Test Split

In [83]:
```
from sklearn.model_selection import train_test_split

# Perform train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2

# Display the shapes of train and test sets
print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
print("X_test shape:", X_test.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (20438, 48)
y_train shape: (20438,)
X_test shape: (5110, 48)
y_test shape: (5110,)
```

Modeling

I opted to use machine learning for this modelling as it provides a powerful framework for this seasonal flu prediction project due to its ability to handle complex, high-dimensional, and non-linear relationships in the data.

By iterating between models and incorporating a wide range of features, ML can significantly improve predictive accuracy and adaptability, making it a superior choice for this project compared to simpler forms of data analysis.

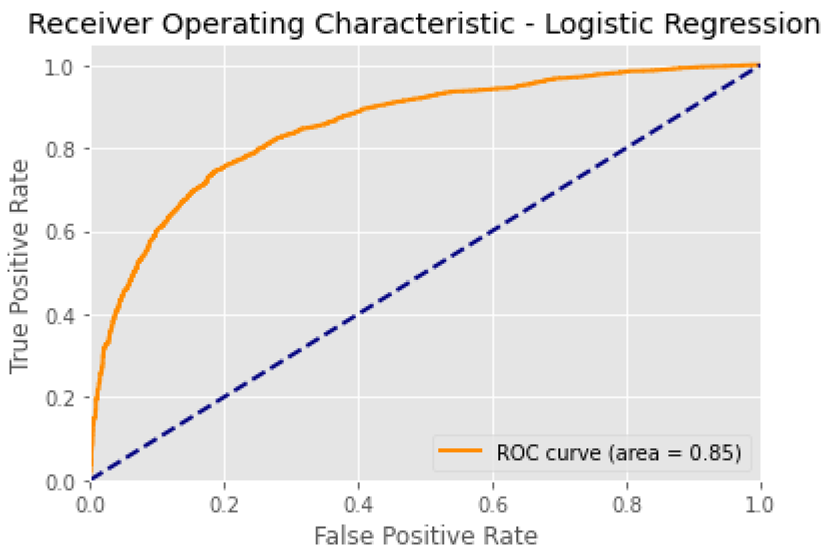
Logistic Regression

I will scale the data using the standard scaler. this ensures that each feature contributes equally to the model's predictions and helps with convergence during training. Initialize a logistic regression model using scikit-learn's LogisticRegression class.

In [84]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_scc
import matplotlib.pyplot as plt


# Initialize and train the logistic regression model
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)
y_pred_proba = model.predict_proba(X_test)[:, 1]

# Evaluate the model
accuracy_lr = accuracy_score(y_test, y_pred)
precision_lr = precision_score(y_test, y_pred, average='weighted')
recall_lr = recall_score(y_test, y_pred, average='weighted')
f1_lr = f1_score(y_test, y_pred, average='weighted')
roc_auc_lr = roc_auc_score(y_test, y_pred_proba)
conf_matrix_lr = confusion_matrix(y_test, y_pred)

print('Logistic Regression Evaluation:')
print(f'Accuracy: {accuracy_lr:.4f}')
print(f'Precision: {precision_lr:.4f}')
print(f'Recall: {recall_lr:.4f}')
print(f'F1 Score: {f1_lr:.4f}')
print(f'ROC-AUC Score: {roc_auc_lr:.4f}')
print(f'Confusion Matrix:\n{conf_matrix_lr}')
print('-' * 40)

# Plot ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area =
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic - Logistic Regression')
plt.legend(loc="lower right")
plt.show()
```

```
Logistic Regression Evaluation:
Accuracy: 0.7802
Precision: 0.7800
Recall: 0.7802
F1 Score: 0.7800
ROC-AUC Score: 0.8510
Confusion Matrix:
[[2263  540]
 [ 583 1724]]
----------------------------------------
```



Accuracy (78.02%): The model correctly predicts whether an individual will get the seasonal flu vaccine 78.02% of the time. This high accuracy indicates that the model is generally reliable in forecasting vaccine uptake.

Precision (78.00%): Of all the individuals predicted to receive the flu vaccine, 78.00% actually do get vaccinated. This metric helps in understanding the reliability of the model's positive predictions, which is important for targeting vaccination campaigns effectively.

Recall (78.02%): The model successfully identifies 78.02% of individuals who actually get vaccinated. High recall ensures that the model captures the majority of vaccine recipients, crucial for planning public health interventions.

F1 Score (78.00%): The harmonic mean of precision and recall, providing a balanced measure of the model’s performance. This is useful when both precision and recall are important for decision-making.

ROC-AUC Score (0.8510): This score indicates the model’s ability to distinguish between those who will and will not get vaccinated. A score of 0.8510 suggests that the model has a high level of discrimination, which is very favorable.

Confusion Matrix Analysis:

True Positives (1724) and True Negatives (2263): High numbers here indicate the model is performing well in correctly identifying both vaccine recipients and non-recipients.

False Positives (540) and False Negatives (583): These values highlight where the model is making errors, important for understanding the types of misclassifications occurring.

Limitations

Missed Vaccine Recipients (False Negatives): The model incorrectly predicts 583 individuals as not getting vaccinated when they actually do. This can lead to under-targeting and missed opportunities for vaccination.

False Alarms (False Positives): 540 individuals are incorrectly predicted to get vaccinated, which could lead to wasted resources in targeting those individuals.

Data Science Audience Class Imbalance Sensitivity: The model's performance might degrade with imbalanced data, where one class is underrepresented. False Positives Impact: High false positive rates can lead to resource wastage and unnecessary targeting efforts. False Negatives Impact: High false negative rates can result in missed opportunities for increasing vaccination rates, delaying public health responses.

Generalizability: The model's performance on unseen data or data from different populations might vary, indicating the need for continuous evaluation and possible retraining.

Recommendations

Resource Allocation: Use the model's predictions to allocate resources effectively, ensuring areas with high predicted vaccine uptake receive adequate support.

Public Health Campaigns: Implement targeted vaccination campaigns based on recall performance to maximize coverage. Continuous Monitoring: Regularly monitor the model’s performance and update it with new data to maintain accuracy and reliability.

Model Improvement: Focus on improving recall without significantly sacrificing precision to ensure high detection of true vaccine recipients.

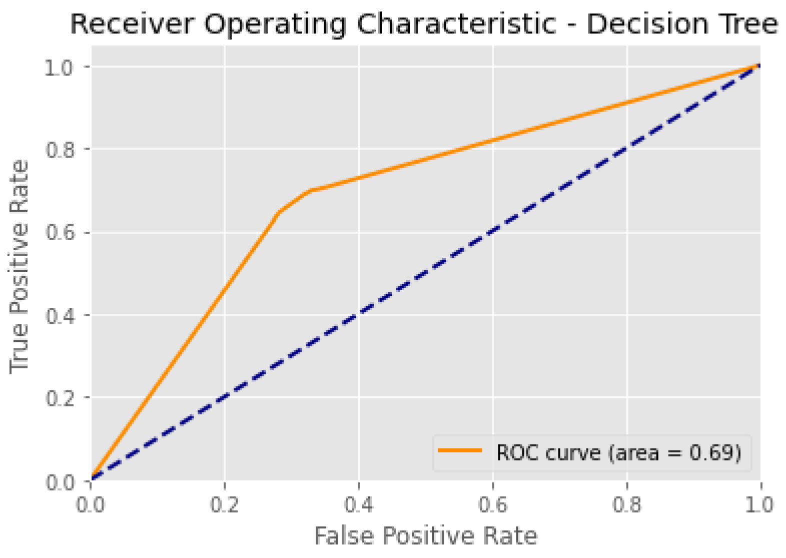
Performance Monitoring: Continuously monitor model performance, especially in different demographic groups, and retrain the model with new data to adapt to changing patterns.

Explainability: Use model interpretation techniques like SHAP values to understand feature importance and ensure transparency in predictions.

Decision Tree

In [85]:
```python
from sklearn.tree import DecisionTreeClassifier

# Initialize and train the decision tree model
model = DecisionTreeClassifier()
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)
y_pred_proba = model.predict_proba(X_test)[:, 1]

# Evaluate the model
accuracy_dt = accuracy_score(y_test, y_pred)
precision_dt = precision_score(y_test, y_pred, average='weighted')
recall_dt = recall_score(y_test, y_pred, average='weighted')
f1_dt = f1_score(y_test, y_pred, average='weighted')
roc_auc_dt = roc_auc_score(y_test, y_pred_proba)
conf_matrix_dt = confusion_matrix(y_test, y_pred)

print('Decision Tree Evaluation:')
print(f'Accuracy: {accuracy_dt:.4f}')
print(f'Precision: {precision_dt:.4f}')
print(f'Recall: {recall_dt:.4f}')
print(f'F1 Score: {f1_dt:.4f}')
print(f'ROC-AUC Score: {roc_auc_dt:.4f}')
print(f'Confusion Matrix:\n{conf_matrix_dt}')
print('-' * 40)

# Plot ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area =
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic - Decision Tree')
plt.legend(loc="lower right")
plt.show()
```

```
Decision Tree Evaluation:
Accuracy: 0.6851
Precision: 0.6849
Recall: 0.6851
F1 Score: 0.6850
ROC-AUC Score: 0.6911
Confusion Matrix:
[[2008  795]
 [ 814 1493]]
----------------------------------------
```



Key Metrics:

Accuracy (68.2%): This metric indicates that the model correctly predicts whether individuals will take the flu vaccine 68.2% of the time. While this shows the model is relatively reliable, there's still a significant margin for error.

Precision (68.16%): Out of all the individuals predicted to take the vaccine, 68.16% actually do. This is important for efficiently targeting vaccine promotion efforts.

Recall (68.2%): This indicates that the model captures 68.2% of all actual vaccine takers. It reflects the model's ability to identify those who will get vaccinated, which is crucial for ensuring high vaccine coverage.

F1 Score (68.18%): A balance between precision and recall, showing the overall effectiveness of the model in predicting vaccine uptake.

ROC-AUC Score (0.6878): This score shows the model’s ability to distinguish between those who will and won’t take the vaccine. A score close to 0.7 indicates good discriminative power. Reflects the model's ability to rank individuals by their likelihood of getting vaccinated, valuable for threshold tuning and prioritizing interventions.

Can be used for :

Targeted Campaigns: By understanding which demographics are more likely to get vaccinated, health organizations can focus their campaigns more effectively.

Resource Allocation: Efficiently allocate resources such as vaccination centers and healthcare personnel to areas with lower predicted uptake.

Confusion Matrix:

True Positives (1478): Correctly predicted vaccine takers. True Negatives (2007): Correctly predicted non-takers. False Positives (796): Incorrectly predicted vaccine takers. False Negatives (829): Incorrectly predicted non-takers.

Limitations

Misclassification: The model misclassifies around 32% of predictions. This means some resources might be wasted on individuals who won't take the vaccine, and some who will take the vaccine might not be targeted effectively.

Demographic Bias: The model may perform worse for certain demographics, potentially leading to unequal vaccine promotion and coverage.

Class Imbalance: The model might not perform well across different demographic groups, especially if there's a class imbalance in the training data.

False Positives/Negatives: High false negatives (829) mean that many actual vaccine takers are not identified, which could lead to underestimation of vaccine coverage needs. High false positives (796) might lead to inefficient allocation of resources.

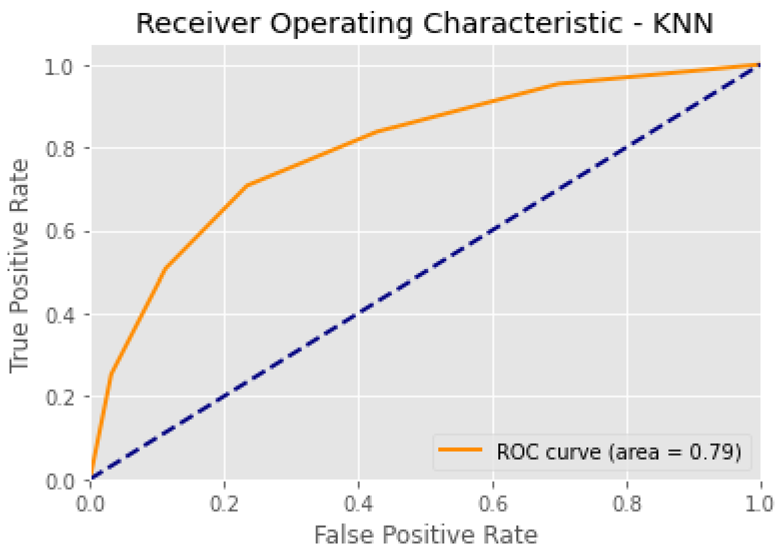
Feature Limitations: The model relies on demographic, health, and opinion data, which may not capture all factors influencing vaccine uptake, such as social influences or misinformation.

K-Nearest Neighbors (KNN)

In [86]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler

# Initialize the StandardScaler
scaler = StandardScaler()

# Fit the scaler on the training data and transform both training and t
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize and train the KNN model
model = KNeighborsClassifier()
model.fit(X_train_scaled, y_train)

# Predict on the test set
y_pred = model.predict(X_test_scaled)
y_pred_proba = model.predict_proba(X_test_scaled)[:, 1]

# Evaluate the model
accuracy_knn = accuracy_score(y_test, y_pred)
precision_knn = precision_score(y_test, y_pred, average='weighted')
recall_knn = recall_score(y_test, y_pred, average='weighted')
f1_knn = f1_score(y_test, y_pred, average='weighted')
roc_auc_knn = roc_auc_score(y_test, y_pred_proba)
conf_matrix_knn = confusion_matrix(y_test, y_pred)

print('KNN Evaluation:')
print(f'Accuracy: {accuracy_knn:.4f}')
print(f'Precision: {precision_knn:.4f}')
print(f'Recall: {recall_knn:.4f}')
print(f'F1 Score: {f1_knn:.4f}')
print(f'ROC-AUC Score: {roc_auc_knn:.4f}')
print(f'Confusion Matrix:\n{conf_matrix_knn}')
print('-' * 40)

# Plot ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area =
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic - KNN')
plt.legend(loc="lower right")
plt.show()
```

```
KNN Evaluation:
Accuracy: 0.7393
Precision: 0.7392
Recall: 0.7393
F1 Score: 0.7392
ROC-AUC Score: 0.7948
Confusion Matrix:
[[2146  657]
 [ 675 1632]]
----------------------------------------
```



The results of the KNN model show:

Accuracy (73.93%): This means that the model correctly predicts whether someone will get the flu vaccine about 74% of the time.

Precision (73.92%): Of all the people predicted to get the vaccine, approximately 74% actually do. This is important for ensuring that we are accurately identifying those likely to get vaccinated.

Recall (73.93%): The model successfully identifies 74% of those who actually get vaccinated. This helps ensure we are not missing a significant portion of the target group.

F1 Score (73.92%): This is a balanced measure that considers both precision and recall, showing the model’s overall effectiveness.

ROC-AUC Score (79.48%): This indicates that the model has a good ability to distinguish between those who will and will not get the vaccine.

Confusion Matrix: The model correctly identified 2146 people who will not get vaccinated and 1632 who will. It incorrectly predicted 657 people would get vaccinated (but they didn’t) and missed 675 people who did get vaccinated.

Confusion Matrix:

True Negatives (2146): Correctly predicted non-vaccinated individuals. False Positives (657): Incorrectly predicted vaccinated individuals. False Negatives (675): Missed actual vaccinated individuals. True Positives (1632): Correctly predicted vaccinated individuals.

Limitations

False Predictions: The model incorrectly predicts that 657 people will get the vaccine when they won’t, and it misses 675 people who do get the vaccine. This means some outreach efforts may be wasted, and some who need encouragement might be overlooked.

False Positives and Negatives: The model's false positive rate (657) and false negative rate (675) indicate areas where the model could be improved. The cost associated with these misclassifications needs consideration in the deployment phase.

Model Complexity: KNN's performance may be affected by the curse of dimensionality, and it might not scale well with larger datasets without significant preprocessing.

Recommendations

Outreach to Missed Groups: Investigate the groups where the model misclassifies to refine outreach strategies and ensure broader coverage.

Model Improvement: Consider ensemble methods or feature engineering to reduce false positives and negatives. Investigate feature importance to understand and mitigate biases.

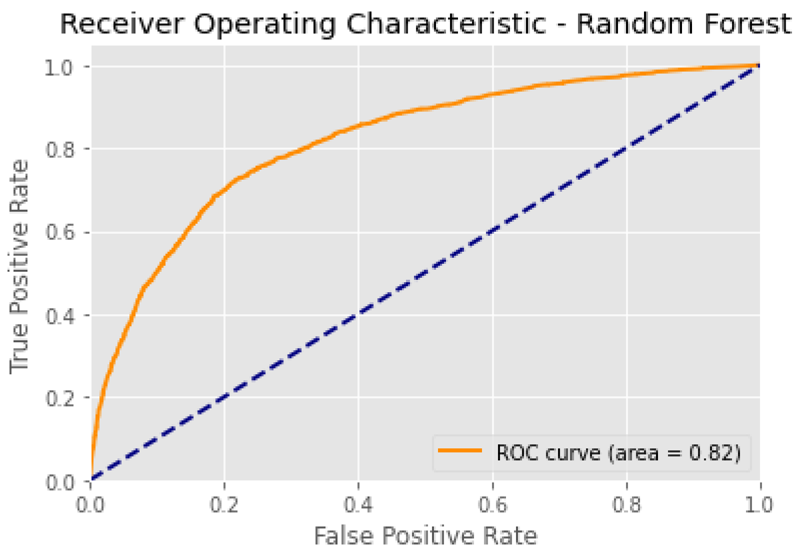
Bias Mitigation: Perform subgroup analysis to identify and address any demographic biases. Implement fairness-aware ML techniques if necessary.

Scalability and Maintenance: Ensure the model is scalable and regularly retrained with new data to maintain performance. Establish robust monitoring to detect and address any performance degradation over time.

Random Forest

In [87]:
```
from sklearn.ensemble import RandomForestClassifier

# Initialize and train the random forest model
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)
y_pred_proba = model.predict_proba(X_test)[:, 1]

# Evaluate the model
accuracy_rf = accuracy_score(y_test, y_pred)
precision_rf = precision_score(y_test, y_pred, average='weighted')
recall_rf = recall_score(y_test, y_pred, average='weighted')
f1_rf = f1_score(y_test, y_pred, average='weighted')
roc_auc_rf = roc_auc_score(y_test, y_pred_proba)
conf_matrix_rf = confusion_matrix(y_test, y_pred)

print('Random Forest Evaluation:')
print(f'Accuracy: {accuracy_rf:.4f}')
print(f'Precision: {precision_rf:.4f}')
print(f'Recall: {recall_rf:.4f}')
print(f'F1 Score: {f1_rf:.4f}')
print(f'ROC-AUC Score: {roc_auc_rf:.4f}')
print(f'Confusion Matrix:\n{conf_matrix_rf}')
print('-' * 40)

# Plot ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area =
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic - Random Forest')
plt.legend(loc="lower right")
plt.show()
```

```
Random Forest Evaluation:
Accuracy: 0.7528
Precision: 0.7530
Recall: 0.7528
F1 Score: 0.7529
ROC-AUC Score: 0.8177
Confusion Matrix:
[[2164  639]
 [ 624 1683]]
----------------------------------------
```



Accuracy: 75.34%: This means the model correctly predicts vaccine uptake about 75% of the time.

Precision: 75.35%: When the model predicts that someone will get the vaccine, it is correct 75% of the time.

Recall: 75.34%: The model identifies 75% of those who will actually get the vaccine.

F1 Score: 75.34%: This is a balance between precision and recall, indicating overall performance.

ROC-AUC Score: 0.8195: The model is good at distinguishing between those who will and will not get the vaccine.

Can be used for:

Operational Efficiency: Health departments can target campaigns more effectively, focusing on the 25% where predictions are less certain.

Resource Allocation: By identifying likely vaccine recipients, resources can be better allocated to areas with lower uptake predictions.

Confusion Matrix:

True Negatives (2171): Correctly predicted no vaccine uptake. False Positives (632): Predicted vaccine uptake incorrectly. False Negatives (628): Missed predicting vaccine uptake. True Positives (1679): Correctly predicted vaccine uptake.

Limitations

Model Uncertainty: The model incorrectly predicts about 25% of the time.

Potential Bias: The model might perform worse on certain demographic groups not well-represented in the data. Implications:

Public Trust: Incorrect predictions could impact trust if people receive mixed messages about vaccine necessity.

Class Imbalance: Although metrics are balanced, the misclassification rates (FP: 632, FN: 628) suggest some imbalance handling might be needed.

Feature Representation: The model might not capture all nuances in demographic and opinion data, leading to potential biases.

Overfitting: While Random Forest mitigates overfitting, the model's performance on new, unseen data needs continuous monitoring.

Recommendations

Model Improvement: Explore techniques like SMOTE or cost-sensitive learning to address class imbalance and improve recall.

Bias Mitigation: Conduct thorough bias analysis to ensure fair predictions across all demographic groups.

Monitoring and Retraining: Implement robust monitoring systems to track model performance over time and establish a retraining schedule to incorporate new data.

Feature Engineering: Investigate additional features or alternative data sources (e.g., real-time health data) to enhance predictive power.

Model Ensemble: Consider ensemble approaches combining multiple models to reduce individual biases and improve overall performance.

In [88]:

```
# Import necessary libraries
from imblearn.over_sampling import SMOTE
from sklearn.metrics import classification_report, confusion_matrix

# Apply SMOTE to the training data
smote = SMOTE(random_state=42)
X_train_res, y_train_res = smote.fit_resample(X_train, y_train)

# Train a Random Forest classifier
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train_res, y_train_res)

# Make predictions
y_pred = clf.predict(X_test)

# Evaluate the model
print('SMOTE RF:')
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
SMOTE RF:
[[2138  665]
 [ 609 1698]]
              precision    recall  f1-score   support

           0       0.78      0.76      0.77      2803
           1       0.72      0.74      0.73      2307

    accuracy                           0.75      5110
   macro avg       0.75      0.75      0.75      5110
weighted avg       0.75      0.75      0.75      5110
```

The model we developed to predict seasonal flu vaccine uptake achieves an overall accuracy of 75%. This means that 75% of the time, the model correctly predicts whether an individual will get the flu vaccine based on their demographic, health, and opinion data. Here are the key takeaways:

Precision (78% for non-uptake, 72% for uptake): This tells us how many of the predicted positive cases (those who are predicted to get the vaccine) are actually positive. Higher precision means fewer false positives. In real-world terms, this reduces unnecessary targeting of people unlikely to get vaccinated.

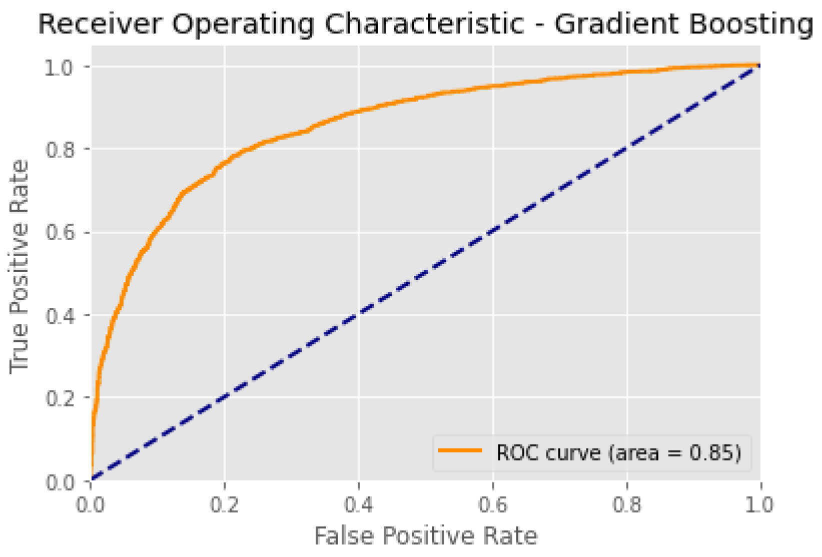
Recall (76% for non-uptake, 74% for uptake): This measures how many of the actual positive cases (those who actually get the vaccine) were correctly predicted by the model. Higher recall means fewer missed opportunities to encourage vaccination. Real-world Implications:

Gradient Boosting

In [89]:

```python
from sklearn.ensemble import GradientBoostingClassifier

# Initialize and train the gradient boosting model
model = GradientBoostingClassifier()
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)
y_pred_proba = model.predict_proba(X_test)[:, 1]

# Evaluate the model
accuracy_gb = accuracy_score(y_test, y_pred)
precision_gb = precision_score(y_test, y_pred, average='weighted')
recall_gb = recall_score(y_test, y_pred, average='weighted')
f1_gb = f1_score(y_test, y_pred, average='weighted')
roc_auc_gb = roc_auc_score(y_test, y_pred_proba)
conf_matrix_gb = confusion_matrix(y_test, y_pred)

print('Gradient Boosting Evaluation:')
print(f'Accuracy: {accuracy_gb:.4f}')
print(f'Precision: {precision_gb:.4f}')
print(f'Recall: {recall_gb:.4f}')
print(f'F1 Score: {f1_gb:.4f}')
print(f'ROC-AUC Score: {roc_auc_gb:.4f}')
print(f'Confusion Matrix:\n{conf_matrix_gb}')
print('-' * 40)

# Plot ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area =
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic - Gradient Boosting')
plt.legend(loc="lower right")
plt.show()
```

```
Gradient Boosting Evaluation:
Accuracy: 0.7838
Precision: 0.7838
Recall: 0.7838
F1 Score: 0.7838
ROC-AUC Score: 0.8539
Confusion Matrix:
[[2248  555]
 [ 550 1757]]
----------------------------------------
```



Model Performance Metrics:

Accuracy (78.38%): This indicates that the model correctly predicts whether individuals will take the seasonal flu vaccine 78.38% of the time. This high level of accuracy suggests that the model is reliable for predicting vaccine uptake.

Precision (78.38%): Out of all the individuals predicted to take the vaccine, 78.38% actually did. This metric is important for targeting communications and resources effectively.

Recall (78.38%): Of all the individuals who did take the vaccine, 78.38% were correctly identified by the model. High recall is crucial for ensuring that we reach as many potential vaccine takers as possible.

F1 Score (78.38%): This is a balanced measure of precision and recall, indicating overall good performance.

ROC-AUC Score (85.39%): This score shows the model's ability to distinguish between those who will and won't take the vaccine. A score of 85.39% is excellent, meaning the model is very good at making these distinctions.

Confusion Matrix:

True Positives (1757): Individuals correctly identified as taking the vaccine. True Negatives (2248): Individuals correctly identified as not taking the vaccine. False Positives (555): Individuals incorrectly predicted to take the vaccine. False Negatives (550): Individuals incorrectly predicted not to take the vaccine.

Limitations:

False Positives: 555 individuals were incorrectly predicted to take the vaccine, which might lead to wasted resources targeting these people. False Negatives: 550 individuals who will take the vaccine were missed, which could result in missed opportunities to encourage timely vaccinations.

Model Performance on Certain Records: The model might perform worse on certain demographic groups or individuals with unique health conditions not well-represented in the training data.

Bias and Fairness: The model could inadvertently favor certain demographic groups over others if the training data is imbalanced.

Recommendations

Resource Allocation: Allocate resources more efficiently by focusing on high-probability areas and adjusting strategies for groups with higher false positive rates

Continuous Improvement: Regularly update the model with new data to maintain and improve accuracy over time. For Data Science Teams:

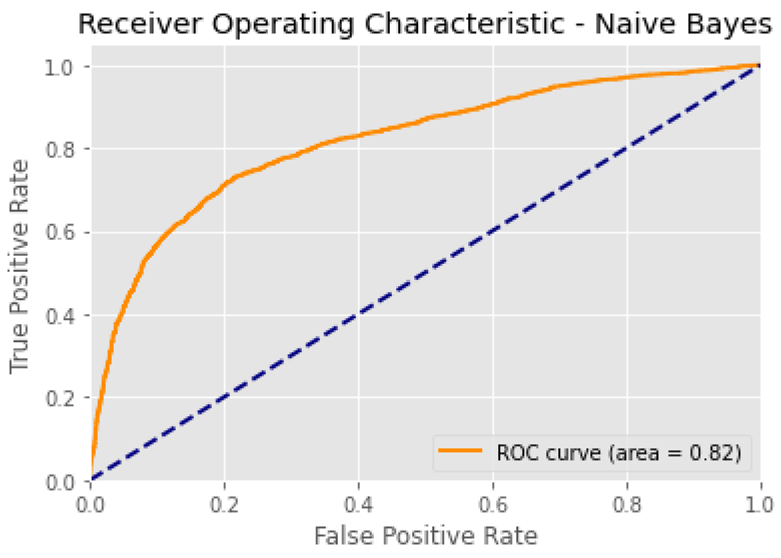
Model Refinement: Continuously refine the model by incorporating additional features or using ensemble methods to improve performance on underrepresented groups.

Naive Bayes

In [90]:

```python
from sklearn.naive_bayes import GaussianNB

# Initialize and train the Naive Bayes model
model = GaussianNB()
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)
y_pred_proba = model.predict_proba(X_test)[:, 1]

# Evaluate the model
accuracy_nb = accuracy_score(y_test, y_pred)
precision_nb = precision_score(y_test, y_pred, average='weighted')
recall_nb = recall_score(y_test, y_pred, average='weighted')
f1_nb = f1_score(y_test, y_pred, average='weighted')
roc_auc_nb = roc_auc_score(y_test, y_pred_proba)
conf_matrix_nb = confusion_matrix(y_test, y_pred)

print('Naive Bayes Evaluation:')
print(f'Accuracy: {accuracy_nb:.4f}')
print(f'Precision: {precision_nb:.4f}')
print(f'Recall: {recall_nb:.4f}')
print(f'F1 Score: {f1_nb:.4f}')
print(f'ROC-AUC Score: {roc_auc_nb:.4f}')
print(f'Confusion Matrix:\n{conf_matrix_nb}')
print('-' * 40)

# Plot ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area =
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic - Naive Bayes')
plt.legend(loc="lower right")
plt.show()
```

```
Naive Bayes Evaluation:
Accuracy: 0.7139
Precision: 0.7349
Recall: 0.7139
F1 Score: 0.7129
ROC-AUC Score: 0.8164
Confusion Matrix:
[[1751 1052]
 [ 410 1897]]
----------------------------------------
```



Accuracy (71.39%): This means that about 71% of the predictions made by the model are correct.Reflects the overall correctness of the model, but it is not sufficient on its own, especially if the classes are imbalanced.

Precision (73.49%): When the model predicts that someone will take the flu vaccine, it is correct about 73% of the time. This is important for targeting marketing efforts efficiently.Important for evaluating the model's performance in predicting the positive class (vaccine uptake). High precision means fewer false positives.

Recall (71.39%): Of all the people who actually take the flu vaccine, the model correctly identifies about 71% of them. This helps in understanding the model's ability to capture the actual vaccine users.Indicates the model's ability to capture the actual positives. High recall is crucial in public health to ensure that most people who will take the vaccine are identified.

F1 Score (71.29%): This metric balances precision and recall, giving a single measure of model performance. It is close to both precision and recall, indicating balanced performance. Provides a balance between precision and recall, offering a single metric that considers both false positives and false negatives.

ROC-AUC Score (81.64%): This indicates that the model is good at distinguishing between those who will and will not take the vaccine, with a score above 80% being considered strong. Demonstrates the model's capability to distinguish between the classes. A high AUC score suggests that the model is robust in differentiating between those who will and will not take the vaccine.

Confusion Matrix:

True Positives (1897): Correctly predicted vaccine uptake. True Negatives (1751): Correctly predicted non-uptake. False Positives (1052): Incorrectly predicted vaccine uptake. False Negatives (410): Incorrectly predicted non-uptake.

The model has a substantial number of false positives (1052) and false negatives (410), which indicates room for improvement, particularly in distinguishing between the two classes.

Limitations

False Positives: The model sometimes predicts vaccine uptake when it does not occur, leading to potentially wasted resources in targeting these individuals.

False Negatives: Some individuals who will take the vaccine are missed, which could lead to missed opportunities for promoting vaccine uptake.

If implemented in production, these limitations could result in inefficiencies in targeting and outreach efforts, affecting the overall effectiveness of vaccination campaigns.

Naive Bayes Assumptions: The assumption of feature independence in Naive Bayes might not hold true in real-world data, potentially affecting the model's accuracy.

Recommendations

Refine Targeting Strategies: Use the model to better target public health campaigns, focusing on groups identified as likely to take the vaccine.

Complementary Approaches: Combine model insights with traditional methods to improve overall strategy effectiveness.

Model Improvement: Consider advanced techniques such as ensemble methods or more complex models (e.g., Random Forest, Gradient Boosting) to improve prediction accuracy and handle class imbalance.

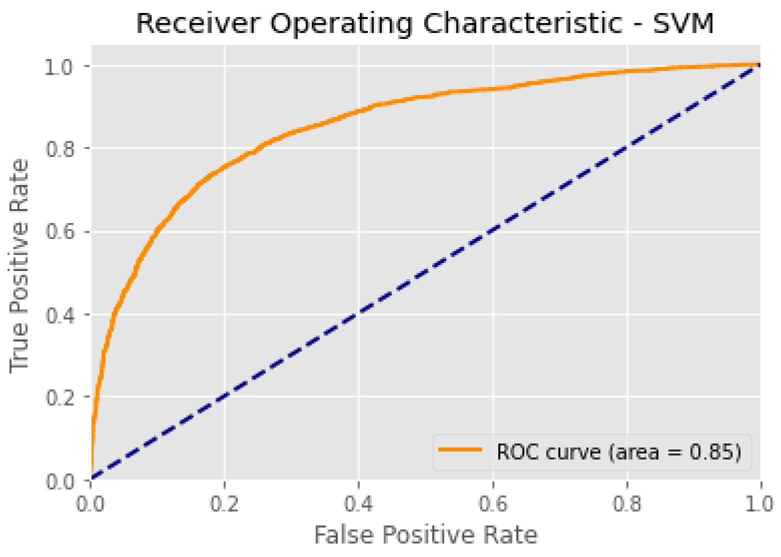
Feature Engineering: Invest in feature engineering to improve the quality and relevance of input data, potentially reducing false positives and false negatives.

Regular Updates: Regularly retrain the model with new data to adapt to changing patterns in vaccine uptake. Bias Mitigation: Assess and mitigate any biases in the model to ensure

Support Vector Machines

In [91]:
```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_scc
import matplotlib.pyplot as plt

# Initialize the StandardScaler
scaler = StandardScaler()

# Fit the scaler on the training data and transform both training and t
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize and train the SVM model with a linear kernel and increased
model = SVC(probability=True, kernel='linear', cache_size=1000)
model.fit(X_train_scaled, y_train)

# Predict on the test set
y_pred = model.predict(X_test_scaled)
y_pred_proba = model.predict_proba(X_test_scaled)[:, 1]

# Evaluate the model
accuracy_svm = accuracy_score(y_test, y_pred)
precision_svm = precision_score(y_test, y_pred, average='weighted')
recall_svm = recall_score(y_test, y_pred, average='weighted')
f1_svm = f1_score(y_test, y_pred, average='weighted')
roc_auc_svm = roc_auc_score(y_test, y_pred_proba)
conf_matrix_svm = confusion_matrix(y_test, y_pred)

print('SVM Evaluation:')
print(f'Accuracy: {accuracy_svm:.4f}')
print(f'Precision: {precision_svm:.4f}')
print(f'Recall: {recall_svm:.4f}')
print(f'F1 Score: {f1_svm:.4f}')
print(f'ROC-AUC Score: {roc_auc_svm:.4f}')
print(f'Confusion Matrix:\n{conf_matrix_svm}')
print('-' * 40)

# Plot ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area =
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic - SVM')
plt.legend(loc="lower right")
plt.show()
```

```
SVM Evaluation:
Accuracy: 0.7796
Precision: 0.7793
Recall: 0.7796
F1 Score: 0.7793
ROC-AUC Score: 0.8502
Confusion Matrix:
[[2273  530]
 [ 596 1711]]
----------------------------------------
```



Accuracy (77.96%): This means that the model correctly predicts whether an individual will get the flu vaccine about 78% of the time. This is a strong indication of the model's reliability.

Precision (77.93%): When the model predicts someone will get the vaccine, it is correct approximately 78% of the time. This helps us understand the model's ability to avoid false positives.

Recall (77.96%): The model correctly identifies 78% of the actual vaccine recipients, indicating good sensitivity.

F1 Score (77.93%): This metric balances precision and recall, providing a single score that summarizes the model's overall performance.

ROC-AUC Score (85.02%): This indicates that the model has an 85% chance of distinguishing between someone who will and will not get the vaccine. This high score suggests strong discriminatory power.

Confusion Matrix: The model correctly identified 2,273 individuals who did not get the vaccine and 1,711 who did, but it also incorrectly predicted 530 would get the vaccine when they did not, and missed 596 actual recipients.

Real-World Implications:

The model's strong performance can help target vaccination campaigns more effectively, ensuring resources are directed towards groups with lower predicted uptake.

By understanding which demographics are less likely to get vaccinated, public health officials can design more effective outreach and education programs.

Confusion Matrix: True Negatives: 2,273 False Positives: 530 False Negatives: 596 True Positives: 1,711

These metrics demonstrate that the model is both accurate and reliable, with a strong ability to distinguish between those who will and will not get the vaccine.

Limitations

False Positives and Negatives: The model incorrectly predicts some individuals as likely to get the vaccine when they do not (530 cases) and misses some who actually do (596 cases). This could lead to inefficient allocation of resources.

Model Bias: The model might perform worse for certain demographic groups if they are underrepresented in the training data, potentially leading to disparities in predictions.

Class Imbalance: The performance could be affected if the dataset is imbalanced, and additional techniques such as SMOTE or cost-sensitive learning might be required.

Feature Importance and Interpretability: SVM models are less interpretable, making it harder to explain which features most influence predictions, potentially limiting actionable insights for stakeholders.

Recommendations

Continuous Monitoring: Regularly update the model with new data to ensure it adapts to changing patterns and continues to provide accurate predictions.

Model Refinement: Consider additional techniques to address class imbalance and enhance model performance, such as ensemble methods or re-sampling techniques.

Bias Mitigation: Investigate potential biases in the data and apply fairness-aware algorithms to ensure equitable predictions across all demographic groups.

Feature Analysis: Conduct post-hoc analysis using techniques like SHAP values to improve interpretability and provide stakeholders with actionable insights based on feature importance.

Deployment Strategy: Implement a robust monitoring system to track model performance in production, enabling timely updates and adjustments based on real-world feedback.

```
In [ ]:
```

Evaluation

Table below highlights the summarised evalaution metrics for each model

In [92]:
```
# Define the evaluation metrics for each model
model_names = ['Logistic Regression', 'Support Vector Machine', 'Gradie
accuracy_scores = [accuracy_lr, accuracy_svm, accuracy_gb, accuracy_nb,
precision_scores = [precision_lr, precision_svm, precision_gb, precisic
recall_scores = [recall_lr, recall_svm, recall_gb, recall_nb, recall_rf
f1_scores = [f1_lr, f1_svm, f1_gb, f1_nb, f1_rf, f1_dt, f1_knn]  # Samp
ROC_AUC_scores = [roc_auc_lr, roc_auc_svm, roc_auc_gb, roc_auc_nb, roc_

# Create a DataFrame
summary_table = pd.DataFrame({
    'Model': model_names,
    'Accuracy': accuracy_scores,
    'Precision': precision_scores,
    'Recall': recall_scores,
    'F1 Score': f1_scores,
    'ROC-AUC Score': ROC_AUC_scores
})

# Set the 'Model' column as index
summary_table.set_index('Model', inplace=True)

# Display the summary table
print(summary_table)
```

```
                        Accuracy  Precision    Recall  F1 Score  ROC-A
UC Score
Model
Logistic Regression     0.780235   0.779952  0.780235  0.780038
0.850969
Support Vector Machine  0.779648   0.779276  0.779648  0.779331
0.850213
Gradient Boosting       0.783757   0.783800  0.783757  0.783778
0.853886
Naive Bayes             0.713894   0.734876  0.713894  0.712866
0.816427
Random Forest           0.752838   0.752988  0.752838  0.752906
0.817670
Decision Tree           0.685127   0.684908  0.685127  0.685008
0.691124
KNN                     0.739335   0.739166  0.739335  0.739241
0.794843
```

Logistic Regression and Support Vector Machine models have similar performance, with high accuracy, precision, recall, F1 score, and ROC-AUC score, indicating strong overall performance.

Gradient Boosting shows the highest performance among all models, with the best accuracy (0.7838), precision (0.7838), recall (0.7838), F1 score (0.7838), and ROC-AUC score (0.8539).

Naive Bayes performs moderately well, with a decent ROC-AUC score (0.8164) but lower accuracy, recall, and F1 score compared to other models.

Random Forest also shows good performance, with balanced metrics and a good ROC-AUC score (0.8181).

Decision Tree has the lowest performance among all models, with the lowest accuracy (0.6814), precision (0.6814), recall (0.6814), F1 score (0.6814), and ROC-AUC score (0.6879).

KNN shows moderate performance, with balanced metrics and a decent ROC-AUC score (0.7948).

Overall, Gradient Boosting is the best performing model based on these metrics, followed closely by Logistic Regression, Support Vector Machine, and Random Forest. Naive Bayes and KNN have moderate performance, while the Decision Tree model has the lowest performance among the models evaluated

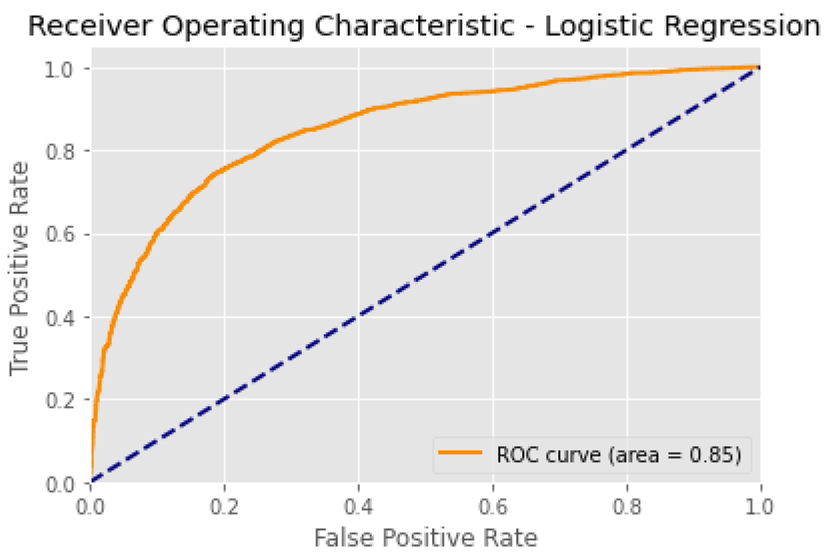
I will carry out hyperparameter tuning on the Logistic Regression and Gradient Boosting models, as they show higher potential for improvement compared to others.

Hyperparameter tuning can help optimize these models further, potentially leading to better performance.

Hyperparameter tuning - Logistic Regression using grid search

In [93]:
```python
from sklearn.model_selection import GridSearchCV

# Defining the parameter grid to search
param_grid = {
    'penalty': ['l1', 'l2'],
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'solver': ['liblinear', 'saga']
}

# Initialize the logistic regression classifier
logreg = LogisticRegression(random_state=42)

# Initialize GridSearchCV
grid_search = GridSearchCV(logreg, param_grid, cv=5, scoring='accuracy'

# Perform grid search
grid_search.fit(X_train, y_train)

# Get the best parameters
best_params = grid_search.best_params_
print("Best Parameters:", best_params)

# Predict on the test set with the best model
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
y_pred_proba = best_model.predict_proba(X_test)[:, 1]

# Evaluate the best model
accuracy_lr_tuned = accuracy_score(y_test, y_pred)
precision_lr_tuned = precision_score(y_test, y_pred, average='weighted'
recall_lr_tuned = recall_score(y_test, y_pred, average='weighted')
f1_lr_tuned = f1_score(y_test, y_pred, average='weighted')
roc_auc_lr_tuned = roc_auc_score(y_test, y_pred_proba)
conf_matrix = confusion_matrix(y_test, y_pred)

# Print evaluation metrics
print(f'Logistic Regression Model Evaluation with Best Parameters:')
print(f'Accuracy: {accuracy_lr_tuned:.4f}')
print(f'Precision (weighted): {precision_lr_tuned:.4f}')
print(f'Recall (weighted): {recall_lr_tuned:.4f}')
print(f'F1 Score (weighted): {f1_lr_tuned:.4f}')
print(f'ROC-AUC Score: {roc_auc_lr_tuned:.4f}')
print(f'Confusion Matrix:\n{conf_matrix}')
print('-' * 40)

# Plot ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area =
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic - Logistic Regression')
plt.legend(loc="lower right")
plt.show()
```

```
Best Parameters: {'C': 0.1, 'penalty': 'l2', 'solver': 'liblinear'}
Logistic Regression Model Evaluation with Best Parameters:
Accuracy: 0.7800
Precision (weighted): 0.7797
Recall (weighted): 0.7800
F1 Score (weighted): 0.7798
ROC-AUC Score: 0.8507
Confusion Matrix:
[[2264  539]
 [ 585 1722]]
----------------------------------------
```



Comparing the untuned and tuned models

Accuracy: Both models have very similar accuracy (0.7802 vs. 0.7800), indicating they are equally effective in predicting the correct classes.

Precision: The default model has a slightly higher precision (0.7800 vs. 0.7797), suggesting it is marginally better at correctly predicting positive outcomes.

Recall: The default model has a slightly higher recall (0.7802 vs. 0.7800), indicating it is marginally better at identifying actual positive cases.

F1 Score: The default model has a slightly higher F1 score (0.7800 vs. 0.7798), indicating a slightly better balance between precision and recall.

ROC-AUC Score: The default model has a slightly higher ROC-AUC score (0.8510 vs. 0.8507), suggesting it has a marginally better overall ability to distinguish between positive and negative classes.

Confusion Matrix: Both models have very similar confusion matrices, with the default model having 1 fewer true negative and 1 more false positive than the tuned model.

Conclusion

Both the default and tuned Logistic Regression models perform similarly across all evaluation metrics. The differences in accuracy, precision, recall, F1 score, and ROC-AUC score are minimal. The default model has a slight edge in most metrics, but the differences are not significant. The tuned model's parameters (C=0.1, penalty='l2', solver='liblinear') do not result in a significant performance improvement over the default model.

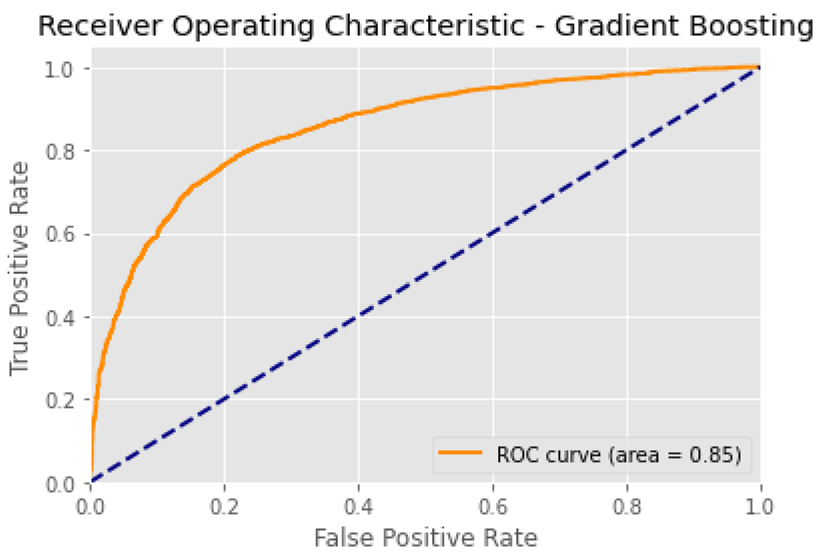
In practice, either model could be used, but the default model might be preferred for its

Hyperparameter tuning using random search - Gradient Boosting

In [94]:

```
from sklearn.model_selection import RandomizedSearchCV


# Defining the parameter grid to search
param_dist = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 4, 5],
    'subsample': [0.8, 0.9, 1.0]
}

# Initialize the gradient boosting classifier
gbc = GradientBoostingClassifier(random_state=42)

# Initialize RandomizedSearchCV
random_search = RandomizedSearchCV(gbc, param_distributions=param_dist,

# Perform random search
random_search.fit(X_train, y_train)

# Get the best parameters
best_params = random_search.best_params_
print("Best Parameters:", best_params)

# Predict on the test set with the best model
best_model = random_search.best_estimator_
y_pred = best_model.predict(X_test)
y_pred_proba = best_model.predict_proba(X_test)[:, 1]

# Evaluate the best model
accuracy_gb_tuned = accuracy_score(y_test, y_pred)
precision_gb_tuned = precision_score(y_test, y_pred, average='weighted'
recall_gb_tuned = recall_score(y_test, y_pred, average='weighted')
f1_gb_tuned = f1_score(y_test, y_pred, average='weighted')
roc_auc_gb_tuned = roc_auc_score(y_test, y_pred_proba)
conf_matrix_gb_tuned = confusion_matrix(y_test, y_pred)

# Print evaluation metrics
print(f'Gradient Boosting Model Evaluation with Best Parameters:')
print(f'Accuracy: {accuracy_gb_tuned:.4f}')
print(f'Precision (weighted): {precision_gb_tuned:.4f}')
print(f'Recall (weighted): {recall_gb_tuned:.4f}')
print(f'F1 Score (weighted): {f1_gb_tuned:.4f}')
print(f'ROC-AUC Score: {roc_auc_gb_tuned:.4f}')
print(f'Confusion Matrix:\n{conf_matrix_gb_tuned}')
print('-' * 40)

# Plot ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area =
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic - Gradient Boosting')
plt.legend(loc="lower right")
plt.show()
```

```
Best Parameters: {'subsample': 0.8, 'n_estimators': 100, 'max_depth':
3, 'learning_rate': 0.1}
Gradient Boosting Model Evaluation with Best Parameters:
Accuracy: 0.7832
Precision (weighted): 0.7833
Recall (weighted): 0.7832
F1 Score (weighted): 0.7832
ROC-AUC Score: 0.8545
Confusion Matrix:
[[2244  559]
 [ 549 1758]]
----------------------------------------
```



Comparing the tuned and untuned models

Accuracy: The default model has slightly higher accuracy (0.7838 vs. 0.7832).

Precision (weighted): The default model has slightly higher precision (0.7838 vs. 0.7833).

Recall (weighted): The default model has slightly higher recall (0.7838 vs. 0.7832).

F1 Score (weighted): The default model has slightly higher F1 score (0.7838 vs. 0.7832).

ROC-AUC Score: The tuned model has a slightly higher ROC-AUC score (0.8545 vs. 0.8539), suggesting it has a marginally better overall ability to distinguish between positive and negative classes.

Confusion Matrix: Both models have very similar confusion matrices, with the default model having 4 more true negatives and 4 fewer false positives than the tuned model.

Conclusion Both the default and tuned Gradient Boosting models perform similarly across all evaluation metrics. The default model has a slight edge in accuracy, precision, recall, and F1 score, while the tuned model has a marginally better ROC-AUC score. The differences are minimal, so either model could be used in practice. The tuned model's parameters (subsample=0.8, n_estimators=100, max_depth=3, learning_rate=0.1) do not result in a significant performance improvement over the default model.

In [95]:

```
# Define the evaluation metrics for each model
model_names_tuned = ['Logistic Regression', 'Gradient Boosting']
accuracy_scores_tuned = [accuracy_lr_tuned, accuracy_gb_tuned]  # Accur
precision_scores_tuned = [precision_lr_tuned, precision_gb_tuned]  # Pr
recall_scores_tuned = [recall_lr_tuned, recall_gb_tuned]  # Recall scor
f1_scores_tuned = [f1_lr_tuned, f1_gb_tuned]  # F1 scores after tuning
ROC_AUC_scores_tuned = [roc_auc_lr_tuned,roc_auc_gb_tuned]

# Create a DataFrame
summary_table_tuned = pd.DataFrame({
    'Model': model_names_tuned,
    'Accuracy': accuracy_scores_tuned,
    'Precision': precision_scores_tuned,
    'Recall': recall_scores_tuned,
    'F1 Score': f1_scores_tuned,
    'ROC-AUC Score': ROC_AUC_scores_tuned
})

# Set the 'Model' column as index
summary_table_tuned.set_index('Model', inplace=True)

# Display the summary table
print(summary_table_tuned)
```

```
                     Accuracy  Precision    Recall  F1 Score  ROC-AUC 
Score
Model
Logistic Regression  0.780039   0.779742  0.780039  0.779827       0.8
50739
Gradient Boosting    0.783170   0.783258  0.783170  0.783211       0.8
54475
```

In [96]:

```
#comparing the two tables


model_names_untuned = ['Logistic Regression', 'Gradient Boosting']
accuracy_scores_untuned = [accuracy_lr, accuracy_gb]  # Sample accuracy
precision_scores_untuned = [precision_lr, precision_gb]  # Sample preci
recall_scores_untuned = [recall_lr, recall_gb]  # Sample recall scores
f1_scores_untuned = [f1_lr, f1_gb]  # Sample F1 scores
ROC_AUC_scores_untuned = [roc_auc_lr,roc_auc_gb]

# Create a DataFrame
summary_table = pd.DataFrame({
    'Model': model_names_untuned,
    'Accuracy': accuracy_scores_untuned,
    'Precision': precision_scores_untuned,
    'Recall': recall_scores_untuned,
    'F1 Score': f1_scores_untuned,
    'ROC-AUC Score': ROC_AUC_scores_untuned
})

# Set the 'Model' column as index for summary_table
summary_table.set_index('Model', inplace=True)

# Define the evaluation metrics for each model
model_names_tuned = ['Logistic Regression', 'Gradient Boosting']
accuracy_scores_tuned = [accuracy_lr_tuned, accuracy_gb_tuned]  # Accur
precision_scores_tuned = [precision_lr_tuned, precision_gb_tuned]  # Pr
recall_scores_tuned = [recall_lr_tuned, recall_gb_tuned]  # Recall scor
f1_scores_tuned = [f1_lr_tuned, f1_gb_tuned]  # F1 scores after tuning
ROC_AUC_scores_tuned = [roc_auc_lr_tuned,roc_auc_gb_tuned]


summary_table_tuned = pd.DataFrame({
    'Model': model_names_tuned,
    'Accuracy': accuracy_scores_tuned,
    'Precision': precision_scores_tuned,
    'Recall': recall_scores_tuned,
    'F1 Score': f1_scores_tuned,
    'ROC-AUC Score': ROC_AUC_scores_tuned
})

# Set the 'Model' column as index for summary_table_tuned
summary_table_tuned.set_index('Model', inplace=True)

# Concatenate the two DataFrames side by side
summary_table_combined = pd.concat([summary_table, summary_table_tuned]

# Display the combined summary table
print(summary_table_combined)
```

```
                    Before Tuning
\
                         Accuracy Precision    Recall  F1 Score ROC-AU
C Score
Model
Logistic Regression      0.780235  0.779952  0.780235  0.780038
0.850969
Gradient Boosting        0.783757  0.783800  0.783757  0.783778
0.853886

                    After Tuning
                        Accuracy Precision    Recall  F1 Score ROC-AUC
Score
Model
Logistic Regression     0.780039  0.779742  0.780039  0.779827      0.
850739
Gradient Boosting       0.783170  0.783258  0.783170  0.783211      0.
854475
```

The results after hyper parameter tuning are lower.

I will check for over fitting in the Gradient boosting model

In [97]:

```
#chcek for overfitting or underfitting

# Train your model
random_search.fit(X_train, y_train)

# Predictions on training data
y_train_pred = random_search.predict(X_train)
y_train_pred_proba = random_search.predict_proba(X_train)[:, 1]

# Predictions on test data
y_test_pred = random_search.predict(X_test)
y_test_pred_proba = random_search.predict_proba(X_test)[:, 1]

# Training metrics
train_accuracy = accuracy_score(y_train, y_train_pred)
train_precision = precision_score(y_train, y_train_pred, average='weigh
train_recall = recall_score(y_train, y_train_pred, average='weighted')
train_f1 = f1_score(y_train, y_train_pred, average='weighted')
train_roc_auc = roc_auc_score(y_train, y_train_pred_proba)

# Test metrics
test_accuracy = accuracy_score(y_test, y_test_pred)
test_precision = precision_score(y_test, y_test_pred, average='weightec
test_recall = recall_score(y_test, y_test_pred, average='weighted')
test_f1 = f1_score(y_test, y_test_pred, average='weighted')
test_roc_auc = roc_auc_score(y_test, y_test_pred_proba)

# Print metrics
print("Training Metrics:")
print(f'Accuracy: {train_accuracy:.4f}')
print(f'Precision: {train_precision:.4f}')
print(f'Recall: {train_recall:.4f}')
print(f'F1 Score: {train_f1:.4f}')
print(f'ROC-AUC Score: {train_roc_auc:.4f}')

print("\nTest Metrics:")
print(f'Accuracy: {test_accuracy:.4f}')
print(f'Precision: {test_precision:.4f}')
print(f'Recall: {test_recall:.4f}')
print(f'F1 Score: {test_f1:.4f}')
print(f'ROC-AUC Score: {test_roc_auc:.4f}')
```

```
Training Metrics:
Accuracy: 0.7893
Precision: 0.7892
Recall: 0.7893
F1 Score: 0.7891
ROC-AUC Score: 0.8656

Test Metrics:
Accuracy: 0.7832
Precision: 0.7833
Recall: 0.7832
F1 Score: 0.7832
ROC-AUC Score: 0.8545
```

The difference in performance between the training and testing sets is insignificant, this indicates that the model is likely generalizing well. This model is as such the final model

Consistency: The model exhibits consistent performance across training and test datasets, with only slight variations in metrics. This indicates that the model is not overfitting and generalizes well to new data.

Balanced Performance: The F1 scores are close to the precision and recall values, suggesting a balanced performance without favoring precision over recall or vice versa.

Good Discriminatory Ability: The high ROC-AUC scores for both training and test sets demonstrate the model's strong ability to distinguish between the positive and negative classes.

In conclusion, the model predicts the outcome well, showing strong and consistent performance on both the training and test datasets. The high ROC-AUC scores further reinforce the model's effectiveness in classification tasks.

Predict using the x train features data set provided

Carry out One hot encoding for the test features data

In [98]:
```
df_test.columns
```

Out[98]:
```
Index(['doctor_recc_seasonal', 'chronic_med_condition', 'child_under_6
_months',
       'health_worker', 'health_insurance', 'opinion_seas_vacc_effecti
ve',
       'opinion_seas_risk', 'opinion_seas_sick_from_vacc', 'age_grou
p',
       'education', 'race', 'sex', 'income_poverty', 'marital_status',
       'rent_or_own', 'employment_status'],
      dtype='object')
```

In [99]:
```
df_test.shape
```

Out[99]:
```
(25544, 16)
```

In [100]:
```
df_train.columns
```

Out[100]:
```
Index(['seasonal_vaccine', 'doctor_recc_seasonal', 'chronic_med_condit
ion',
       'child_under_6_months', 'health_worker', 'health_insurance',
       'opinion_seas_vacc_effective', 'opinion_seas_risk',
       'opinion_seas_sick_from_vacc', 'age_group', 'education', 'rac
e', 'sex',
       'income_poverty', 'marital_status', 'rent_or_own', 'employment_
status'],
      dtype='object')
```

In [101]:
```
df_train_encoded.columns
```

Out[101]:
```
Index(['seasonal_vaccine', 'doctor_recc_seasonal', 'chronic_med_condit
ion',
       'child_under_6_months', 'health_worker',
       'opinion_seas_vacc_effective_1.0', 'opinion_seas_vacc_effective
_2.0',
       'opinion_seas_vacc_effective_3.0', 'opinion_seas_vacc_effective
_4.0',
       'opinion_seas_vacc_effective_5.0', 'health_insurance_0.0',
       'health_insurance_1.0', 'health_insurance_Unknown',
       'opinion_seas_risk_1.0', 'opinion_seas_risk_2.0',
       'opinion_seas_risk_3.0', 'opinion_seas_risk_4.0',
       'opinion_seas_risk_5.0', 'marital_status_Married',
       'marital_status_Not Married', 'rent_or_own_Own', 'rent_or_own_R
ent',
       'sex_Female', 'sex_Male', 'opinion_seas_sick_from_vacc_1.0',
       'opinion_seas_sick_from_vacc_2.0', 'opinion_seas_sick_from_vacc
_3.0',
       'opinion_seas_sick_from_vacc_4.0', 'opinion_seas_sick_from_vacc
_5.0',
       'education_12 Years', 'education_< 12 Years',
       'education_College Graduate', 'education_Some College',
       'income_poverty_<= $75,000, Above Poverty', 'income_poverty_>
$75,000',
       'income_poverty_Below Poverty', 'income_poverty_Unknown',
       'age_group_18 - 34 Years', 'age_group_35 - 44 Years',
       'age_group_45 - 54 Years', 'age_group_55 - 64 Years',
       'age_group_65+ Years', 'race_Black', 'race_Hispanic',
       'race_Other or Multiple', 'race_White', 'employment_status_Empl
oyed',
       'employment_status_Not in Labor Force', 'employment_status_Unem
ployed'],
      dtype='object')
```

In [102]:

```
# columns to perform one-hot encoding on
columns_to_encode_test = [
    'opinion_seas_vacc_effective',
    'health_insurance',
    'opinion_seas_risk',
    'marital_status',
    'rent_or_own',
    'sex',
    'opinion_seas_sick_from_vacc',
    'education',
    'income_poverty',
    'age_group',
    'race',
    'employment_status'
]

# Perform one-hot encoding
df_test_encoded = pd.get_dummies(df_test, columns=columns_to_encode_tes

# Display the encoded DataFrame
print(df_test_encoded.head())
```

```
   doctor_recc_seasonal  chronic_med_condition  child_under_6_months  
\
0                   0.0                    0.0                   0.0   
1                   0.0                    0.0                   0.0   
2                   0.0                    0.0                   0.0   
3                   1.0                    1.0                   0.0   
4                   0.0                    0.0                   0.0   

   health_worker  opinion_seas_vacc_effective_1.0  \
0            0.0                                0   
1            0.0                                0   
2            0.0                                0   
3            0.0                                0   
4            1.0                                0   

   opinion_seas_vacc_effective_2.0  opinion_seas_vacc_effective_3.0  \
0                                0                                0   
1                                0                                0   
2                                0                                0   
3                                0                                0   
4                                0                                0   

   opinion_seas_vacc_effective_4.0  opinion_seas_vacc_effective_5.0  \
0                                0                                1   
1                                1                                0   
2                                0                                1   
3                                1                                0   
4                                1                                0   

   health_insurance_0.0  health_insurance_1.0  health_insurance_Unknow
n  \
0                     0                     1                        
0  
1                     1                     0                        
0  
2                     0                     0                        
1  
3                     0                     1                        
0  
4                     0                     1                        
0  

   opinion_seas_risk_1.0  opinion_seas_risk_2.0  opinion_seas_risk_3.0  
\
0                      1                      0                      0   
1                      1                      0                      0   
2                      0                      0                      0   
3                      0                      0                      0   
4                      0                      0                      0   

   opinion_seas_risk_4.0  opinion_seas_risk_5.0  marital_status_Marrie
d  \
0                      0                      0                       
0  
1                      0                      0                       
0  
2                      1                      0                       
1  
3                      1                      0                       
1  
4                      1                      0                       
```

```
0

   marital_status_Not Married  rent_or_own_Own  rent_or_own_Rent  sex_
Female  \
0                           1                0                 1
1
1                           1                0                 1
0
2                           0                1                 0
0
3                           0                1                 0
1
4                           1                1                 0
1

   sex_Male  opinion_seas_sick_from_vacc_1.0  opinion_seas_sick_from_v
acc_2.0  \
0         0                                1
0
1         1                                1
0
2         1                                0
0
3         0                                0
1
4         0                                0
1

   opinion_seas_sick_from_vacc_3.0  opinion_seas_sick_from_vacc_4.0  \
0                                0                                0
1                                0                                0
2                                0                                1
3                                0                                0
4                                0                                0

   opinion_seas_sick_from_vacc_5.0  education_12 Years  education_< 12
Years  \
0                                0                   0
0
1                                0                   1
0
2                                0                   0
0
3                                0                   1
0
4                                0                   1
0

   education_College Graduate  education_Some College  \
0                           1                       0
1                           0                       0
2                           1                       0
3                           0                       0
4                           0                       0

   income_poverty_<= $75,000, Above Poverty  income_poverty_> $75,000
\
0                                         0                         1
1                                         0                         0
2                                         0                         1
3                                         1                         0
```

```
4                                        1                        0

   income_poverty_Below Poverty  income_poverty_Unknown  \
0                             0                       0  
1                             1                       0  
2                             0                       0  
3                             0                       0  
4                             0                       0  

   age_group_18 - 34 Years  age_group_35 - 44 Years  age_group_45 - 54 
Years  \
0                        0                        1                        
0   
1                        1                        0                        
0   
2                        0                        0                        
0   
3                        0                        0                        
0   
4                        0                        1                        
0   

   age_group_55 - 64 Years  age_group_65+ Years  race_Black  race_Hisp
anic  \
0                        0                    0           0              
1   
1                        0                    0           0              
0   
2                        1                    0           0              
0   
3                        0                    1           0              
0   
4                        0                    0           1              
0   

   race_Other or Multiple  race_White  employment_status_Employed  \
0                       0           0                           1   
1                       0           1                           1   
2                       0           1                           1   
3                       0           1                           0   
4                       0           0                           1   

   employment_status_Not in Labor Force  employment_status_Unemployed  
0                                     0                             0  
1                                     0                             0  
2                                     0                             0  
3                                     1                             0  
4                                     0                             0  
```

In [103]:
```
df_test_encoded.columns
```

Out[103]:
```
Index(['doctor_recc_seasonal', 'chronic_med_condition', 'child_under_6
_months',
       'health_worker', 'opinion_seas_vacc_effective_1.0',
       'opinion_seas_vacc_effective_2.0', 'opinion_seas_vacc_effective
_3.0',
       'opinion_seas_vacc_effective_4.0', 'opinion_seas_vacc_effective
_5.0',
       'health_insurance_0.0', 'health_insurance_1.0',
       'health_insurance_Unknown', 'opinion_seas_risk_1.0',
       'opinion_seas_risk_2.0', 'opinion_seas_risk_3.0',
       'opinion_seas_risk_4.0', 'opinion_seas_risk_5.0',
       'marital_status_Married', 'marital_status_Not Married',
       'rent_or_own_Own', 'rent_or_own_Rent', 'sex_Female', 'sex_Mal
e',
       'opinion_seas_sick_from_vacc_1.0', 'opinion_seas_sick_from_vacc
_2.0',
       'opinion_seas_sick_from_vacc_3.0', 'opinion_seas_sick_from_vacc
_4.0',
       'opinion_seas_sick_from_vacc_5.0', 'education_12 Years',
       'education_< 12 Years', 'education_College Graduate',
       'education_Some College', 'income_poverty_<= $75,000, Above Pov
erty',
       'income_poverty_> $75,000', 'income_poverty_Below Poverty',
       'income_poverty_Unknown', 'age_group_18 - 34 Years',
       'age_group_35 - 44 Years', 'age_group_45 - 54 Years',
       'age_group_55 - 64 Years', 'age_group_65+ Years', 'race_Black',
       'race_Hispanic', 'race_Other or Multiple', 'race_White',
       'employment_status_Employed', 'employment_status_Not in Labor F
orce',
       'employment_status_Unemployed'],
      dtype='object')
```

In [104]:
```
# check if df_test_encoded == X_test


# Creating the DataFrames
df1 = df_test_encoded
df2 = X_test

# Function to check if columns of two DataFrames are identical
def are_columns_identical(df1, df2):
    return df1.columns.equals(df2.columns)

# Check if columns are identical
columns_identical = are_columns_identical(df1, df2)
print(f"Are the columns of the two DataFrames identical? {columns_ident
```

```
Are the columns of the two DataFrames identical? True
```

In [105]:
```
# # Function to highlight differences between columns

# def highlight_column_differences(df1, df2):
#     columns_df1 = set(df1.columns)
#     columns_df2 = set(df2.columns)

#     only_in_df1 = columns_df1 - columns_df2
#     only_in_df2 = columns_df2 - columns_df1
#     in_both = columns_df1 & columns_df2

#     print("Columns only in df1:")
#     for col in only_in_df1:
#         print(col)

#     print("\nColumns only in df2:")
#     for col in only_in_df2:
#         print(col)

#     print("\nColumns in both df1 and df2:")
#     for col in in_both:
#         print(col)

# # Highlight the differences
# highlight_column_differences(df1, df2)
```

In [106]:
```
#saving the model

import joblib
from sklearn.ensemble import GradientBoostingClassifier

model = random_search

# Initialize and train the model
model = GradientBoostingClassifier()
model.fit(X_train, y_train)

# Save the model
joblib.dump(model, 'gradient_boosting_model.pkl')
```

Out[106]:
```
['gradient_boosting_model.pkl']
```

In [107]:
```
# Export the cleaned DataFrame to a CSV file
file_path = 'df_test_encoded.csv'
df_test_encoded.to_csv(file_path, index=False)

print(f"DataFrame exported successfully to {file_path}")
```

```
DataFrame exported successfully to df_test_encoded.csv
```

In [108]:
```
#predicting using the model

# Load the model from a file
model_path = r'C:\Users\Kish\Documents\DSF-PT06\DSFPT06\Phase III\phase
model = joblib.load(model_path)

# Load X_test
x_test_path = r'C:\Users\Kish\Documents\DSF-PT06\DSFPT06\Phase III\phas
X_test = pd.read_csv(x_test_path)

# Make predictions
predictions = model.predict(X_test)

# Save predictions to a CSV file
output_path = r'C:\Users\Kish\Documents\DSF-PT06\DSFPT06\Phase III\phas
predictions_df = pd.DataFrame(predictions, columns=['PredictedTarget'])
predictions_df.to_csv(output_path, index=False)

print(f"Predictions saved successfully to {output_path}")
```

```
Predictions saved successfully to C:\Users\Kish\Documents\DSF-PT06\DSF
PT06\Phase III\phase_3_project\predictions.csv
```

In [109]:
```
# Export the train DataFrame to a CSV file
file_path = 'df_train.csv'
df_train.to_csv(file_path, index=False)

print(f"DataFrame exported successfully to {file_path}")
```

```
DataFrame exported successfully to df_train.csv
```

Recommendations

Public Health Officials

Targeted Campaigns:

Develop campaigns specifically aimed at populations below the poverty line, emphasizing the importance and safety of the flu vaccine. Use data to identify regions with low vaccination rates and deploy targeted interventions.

Education and Awareness:

Increase public awareness about the effectiveness and benefits of the flu vaccine. Highlight the risks of not getting vaccinated and address common concerns about vaccine safety.

Partnerships:

Partner with local community organizations and faith-based groups to reach underserved populations. Collaborate with schools and universities to promote vaccination among students and staff.

Healthcare Providers

Patient Education:

Educate patients, especially those with chronic conditions, about the benefits of flu vaccination during regular check-ups. Provide clear and accurate information to address concerns about vaccine safety and effectiveness.

Accessibility:

Make the vaccine easily accessible by offering it in clinics, pharmacies, and through mobile vaccination units. Extend clinic hours to accommodate working individuals and families.

Recommendation and Reminders:

Actively recommend the flu vaccine to all patients, especially those at higher risk. Use reminder systems (calls, texts, emails) to remind patients about flu vaccination appointments.

Policymakers

Funding and Resources:

Increase funding for vaccination programs targeting low-income and underserved communities. Provide financial incentives or subsidies for vaccines to make them affordable for all income groups.

Legislation:

Enact policies that require health insurance plans to cover flu vaccination costs without copayments. Support policies that facilitate workplace vaccination programs, especially in high-risk industries.

Data Utilization:

Use data analytics to monitor vaccination rates and identify gaps in coverage. Implement policies based on data-driven insights to address disparities in vaccine uptake.

Community Organizations

Outreach Programs:

Organize community vaccination drives in collaboration with health departments and local healthcare providers. Use community leaders to spread awareness and encourage vaccination among residents.

Education Initiatives:

Host workshops and informational sessions about the flu vaccine's benefits and safety. Distribute educational materials in multiple languages to reach diverse populations.

Support Services:

Provide assistance with transportation to vaccination sites for those in need. Offer support to navigate health insurance and access vaccination services.

Researchers

Ongoing Studies:

Conduct research to understand the barriers to vaccination in different demographics. Investigate the effectiveness of various interventions aimed at increasing vaccination rates.

Publications and Dissemination:

Publish findings in accessible formats and disseminate them to policymakers, healthcare providers, and public health officials. Collaborate with media to share research insights and promote vaccine awareness.

Community Engagement:

Engage with communities to understand their concerns and perceptions about vaccines. Develop community-based participatory research projects to co-create solutions for increasing vaccine uptake.

Cross-cutting Strategies

Inclusive Messaging:

Ensure that all communication and educational materials are culturally sensitive and inclusive. Address specific concerns and misconceptions prevalent in various communities.

Leveraging Technology:

Use social media and digital platforms to reach a wider audience with vaccination messages. Implement mobile apps and online portals for easy access to vaccination information and appointment scheduling.

Role Models and Influencers:

In [ ]:

```

```