Final Project Submission

Please fill out:

- Student name: Amase Oyakapeli
- Student pace: full time Hybrid
- Scheduled project review date/time:
- Instructor name: William Okomba
- Blog post URL:

BUSINESS PROBLEM. During flu outbreaks, low vaccination rates contribute to increased hospitalizations, economic burdens, and public health crises. Understanding the factors that influence an individual's willingness to get vaccinated is critical for government agencies, healthcare providers, and pharmaceutical companies. Traditional vaccination campaigns rely on broad, one-size-fits-all messaging, which may not effectively target hesitant populations. Come up with a model that predicts how likely individuals are to receive their H1N1 vaccine.

OBJECTIVES

1. Identify the relationship between risk perception and vaccine uptake.
2. Determine if higher knowledge about H1N1 increases the likelihood of getting vaccinated.
3. Analyze if concerns about vaccine side effects reduce uptake.
4. Segment population groups based on trust in vaccine effectiveness.

DATA UNDERSTANDING You are provided a dataset with 36 columns. The first column respondent_id is a unique and random identifier. The remaining 35 features are described below.

For all binary variables: 0 = No; 1 = Yes. About Their H1N1 Concerns.

1. h1n1_concern - Level of concern about the H1N1 flu. 0 = Not at all concerned; 1 = Not very concerned; 2 = Somewhat concerned; 3 = Very concerned.
2. h1n1_knowledge - Level of knowledge about H1N1 flu. 0 = No knowledge; 1 = A little knowledge; 2 = A lot of knowledge.

Their Actions (Behaviors) 3. behavioral_antiviral_meds - Has taken antiviral medications. (binary) 4. behavioral_avoidance - Has avoided close contact with others with flu-like symptoms. (binary) 5. behavioral_face_mask - Has bought a face mask. (binary) 6. behavioral_wash_hands - Has frequently washed hands or used hand sanitizer. (binary) 7. behavioral_large_gatherings - Has reduced time at large gatherings. (binary) 8. behavioral_outside_home - Has reduced contact with people outside of own household. (binary) 9. behavioral_touch_face - Has avoided touching eyes, nose, or mouth. (binary)

Doctor's Recommendations 10. doctor_recc_h1n1 - H1N1 flu vaccine was recommended by doctor. (binary) 11. doctor_recc_seasonal - Seasonal flu vaccine was recommended by doctor. (binary)

Health Factors 12. chronic_med_condition - Has any of the following chronic medical conditions: asthma or an other lung condition, diabetes, a heart condition, a kidney condition, sickle cell anemia or other anemia, a neurological or neuromuscular condition, a liver condition, or a weakened immune system caused by a chronic illness or by medicines taken for a chronic illness. (binary) 13. child_under_6_months - Has regular close contact with a child under the age of six months. (binary) 14. health_worker - Is a healthcare worker. (binary) 15. health_insurance - Has health insurance. (binary)

Opinions about Vaccines 16. opinion_h1n1_vacc_effective - Respondent's opinion about H1N1 vaccine effectiveness. 1 = Not at all effective; 2 = Not very effective; 3 = Don't know; 4 = Somewhat effective; 5 = Very effective. 17. opinion_h1n1_risk - Respondent's opinion about risk of getting sick with H1N1 flu without vaccine. 1 = Very Low; 2 = Somewhat low; 3 = Don't know; 4 = Somewhat high; 5 = Very high. 18. opinion_h1n1_sick_from_vacc - Respondent's worry of getting sick from taking H1N1 vaccine. 1 = Not at all worried; 2 = Not very worried; 3 = Don't know; 4 = Somewhat worried; 5 = Very worried. 19. opinion_seas_vacc_effective - Respondent's opinion about seasonal flu vaccine effectiveness. 1 = Not at all effective; 2 = Not very effective; 3 = Don't know; 4 = Somewhat effective; 5 = Very effective. 20. opinion_seas_risk - Respondent's opinion about risk of getting sick with seasonal flu without vaccine. 1 = Very Low; 2 = Somewhat low; 3 = Don't know; 4 = Somewhat high; 5 = Very high. 21. opinion_seas_sick_from_vacc - Respondent's worry of getting sick from taking seasonal flu vaccine. 1 = Not at all worried; 2 = Not very worried; 3 = Don't know; 4 = Somewhat worried; 5 = Very worried.

About the Person (Demographics) 22. age_group - Age group of respondent. 23. education - Self-reported education level. 24. race - Race of respondent. 25. sex - Sex of respondent. 26. income_poverty - Household annual income of respondent with respect to 2008 Census poverty thresholds. 27. marital_status - Marital status of respondent. 28. rent_or_own - Housing situation of respondent. 29. employment_status - Employment status of respondent. 30. hhs_geo_region - Respondent's residence using a 10-region geographic classification defined by the U.S. Dept. of Health and Human Services. Values are represented as short random character strings. 31. census_msa - Respondent's residence within metropolitan statistical areas (MSA) as defined by the U.S. Census. 32. household_adults - Number of other adults in household, top-coded to 3. 33. household_children - Number of children in household, top-coded to 3. 34. employment_industry - Type of industry respondent is employed in. Values are represented as short random character strings. 35. employment_occupation - Type of occupation of respondent. Values are represented as short

In [2]:
```
#Importing necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [4]:
```
# Loading the dataset
df = pd.read_csv('training_set_features.csv')
# checking the first 5 rows
df.head()
```

Out[4]:

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavio
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	

5 rows × 36 columns

In [51]:
```
# checking the last 5 rows
df.tail()
```

Out[51]:

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	beha
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	

5 rows × 36 columns

In [52]:
```
df.describe()
```

Out[52]:

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	beha
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	

8 rows × 24 columns

In [53]:
```
# Checking rows and columns in the dataset
df.shape
```

Out[53]:
```
(26707, 36)
```

In [54]:
```
df.columns
```

Out[54]:
```
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_seas
onal',
       'chronic_med_condition', 'child_under_6_months', 'health_worke
r',
       'health_insurance', 'opinion_h1n1_vacc_effective', 'opinion_h1
n1_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_statu
s',
       'rent_or_own', 'employment_status', 'hhs_geo_region', 'census_
msa',
       'household_adults', 'household_children', 'employment_industr
y',
       'employment_occupation'],
      dtype='object')
```

In [55]:
```
# checking the datatypes
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26707 entries, 0 to 26706
Data columns (total 36 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   respondent_id                26707 non-null  int64  
 1   h1n1_concern                 26615 non-null  float64
 2   h1n1_knowledge               26591 non-null  float64
 3   behavioral_antiviral_meds    26636 non-null  float64
 4   behavioral_avoidance         26499 non-null  float64
 5   behavioral_face_mask         26688 non-null  float64
 6   behavioral_wash_hands        26665 non-null  float64
 7   behavioral_large_gatherings  26620 non-null  float64
 8   behavioral_outside_home      26625 non-null  float64
 9   behavioral_touch_face        26579 non-null  float64
 10  doctor_recc_h1n1             24547 non-null  float64
 11  doctor_recc_seasonal         24547 non-null  float64
 12  chronic_med_condition        25736 non-null  float64
 13  child_under_6_months         25887 non-null  float64
 14  health_worker                25903 non-null  float64
 15  health_insurance             14433 non-null  float64
 16  opinion_h1n1_vacc_effective  26316 non-null  float64
 17  opinion_h1n1_risk            26319 non-null  float64
 18  opinion_h1n1_sick_from_vacc  26312 non-null  float64
 19  opinion_seas_vacc_effective  26245 non-null  float64
 20  opinion_seas_risk            26193 non-null  float64
 21  opinion_seas_sick_from_vacc  26170 non-null  float64
 22  age_group                    26707 non-null  object 
 23  education                    25300 non-null  object 
 24  race                         26707 non-null  object 
 25  sex                          26707 non-null  object 
 26  income_poverty               22284 non-null  object 
 27  marital_status               25299 non-null  object 
 28  rent_or_own                  24665 non-null  object 
 29  employment_status            25244 non-null  object 
 30  hhs_geo_region               26707 non-null  object 
 31  census_msa                   26707 non-null  object 
 32  household_adults             26458 non-null  float64
 33  household_children           26458 non-null  float64
 34  employment_industry          13377 non-null  object 
 35  employment_occupation        13237 non-null  object 
dtypes: float64(23), int64(1), object(12)
memory usage: 7.3+ MB
```

In [56]:
```
# DATA PREPARATION
## Data cleaning
# checking missing values
df.isna().sum()
```

Out[56]:
```
respondent_id                      0
h1n1_concern                      92
h1n1_knowledge                   116
behavioral_antiviral_meds         71
behavioral_avoidance             208
behavioral_face_mask              19
behavioral_wash_hands             42
behavioral_large_gatherings       87
behavioral_outside_home           82
behavioral_touch_face            128
doctor_recc_h1n1                2160
doctor_recc_seasonal            2160
chronic_med_condition            971
child_under_6_months             820
health_worker                    804
health_insurance               12274
opinion_h1n1_vacc_effective      391
opinion_h1n1_risk                388
opinion_h1n1_sick_from_vacc      395
opinion_seas_vacc_effective      462
opinion_seas_risk                514
opinion_seas_sick_from_vacc      537
age_group                          0
education                       1407
race                               0
sex                                0
income_poverty                  4423
marital_status                  1408
rent_or_own                     2042
employment_status               1463
hhs_geo_region                     0
census_msa                         0
household_adults                 249
household_children               249
employment_industry            13330
employment_occupation          13470
dtype: int64
```

In [8]:
```
# dropping missing values
df = df.dropna()
```

In [58]:
```
# Confirming there are no missing values
df.isna().sum()
```

Out[58]:
```
respondent_id                  0
h1n1_concern                   0
h1n1_knowledge                 0
behavioral_antiviral_meds      0
behavioral_avoidance           0
behavioral_face_mask           0
behavioral_wash_hands          0
behavioral_large_gatherings    0
behavioral_outside_home        0
behavioral_touch_face          0
doctor_recc_h1n1               0
doctor_recc_seasonal           0
chronic_med_condition          0
child_under_6_months           0
health_worker                  0
health_insurance               0
opinion_h1n1_vacc_effective    0
opinion_h1n1_risk              0
opinion_h1n1_sick_from_vacc    0
opinion_seas_vacc_effective    0
opinion_seas_risk              0
opinion_seas_sick_from_vacc    0
age_group                      0
education                      0
race                           0
sex                            0
income_poverty                 0
marital_status                 0
rent_or_own                    0
employment_status              0
hhs_geo_region                 0
census_msa                     0
household_adults               0
household_children             0
employment_industry            0
employment_occupation          0
dtype: int64
```

In [59]:
```
# checking duplicates
df.duplicated().sum()
```

Out[59]:
```
0
```

```
#there are zero duplicates in this data set
```

FEATURE ENGINEERING

In [60]:
```
#  Behavioral Compliance Score
df['safety_behavior_score'] = df[['behavioral_antiviral_meds', 'behavi
                                  'behavioral_avoidance', 'behavioral_
                                  'behavioral_touch_face']].sum(axis=1
df.head()
```

Out[60]:

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavio
1	1	3.0	2.0	0.0	
7	7	1.0	0.0	0.0	
10	10	2.0	1.0	0.0	
11	11	1.0	2.0	0.0	
15	15	1.0	1.0	0.0	

5 rows × 37 columns

In [61]:
```
#  Doctor Recommendation Influence
df['doctor_recc_total'] = df['doctor_recc_h1n1'] + df['doctor_recc_sea
df.head()
```

Out[61]:

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavio
1	1	3.0	2.0	0.0	
7	7	1.0	0.0	0.0	
10	10	2.0	1.0	0.0	
11	11	1.0	2.0	0.0	
15	15	1.0	1.0	0.0	

5 rows × 38 columns

In [62]:
```
#  Health Risk Factor Score
df['health_risk_score'] = df[['chronic_med_condition', 'health_worker'
df.head()
```

Out[62]:

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavio
1	1	3.0	2.0	0.0	
7	7	1.0	0.0	0.0	
10	10	2.0	1.0	0.0	
11	11	1.0	2.0	0.0	
15	15	1.0	1.0	0.0	

5 rows × 39 columns

In [63]:
```
#  Household Vulnerability Score
df['household_vulnerability'] = df[['household_adults', 'household_chi
df.head()
```

Out[63]:

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavio
1	1	3.0	2.0	0.0	
7	7	1.0	0.0	0.0	
10	10	2.0	1.0	0.0	
11	11	1.0	2.0	0.0	
15	15	1.0	1.0	0.0	

5 rows × 40 columns

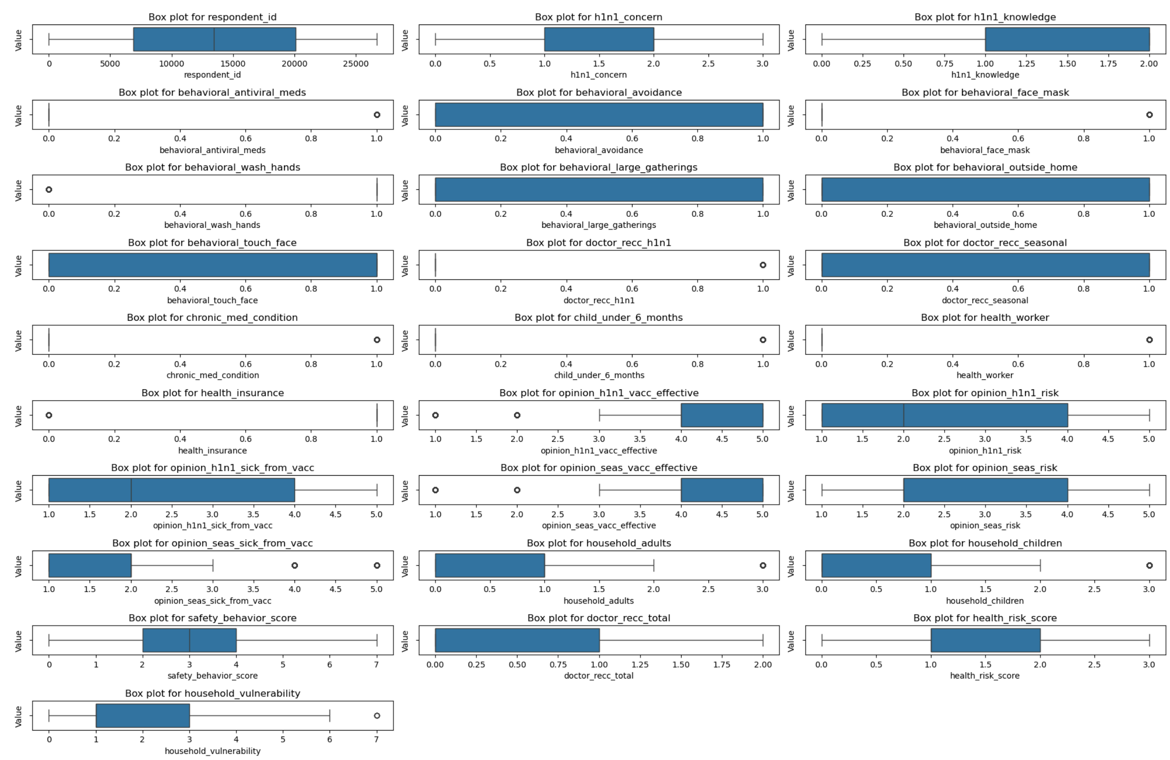
In [64]:
```
#  Socioeconomic Status Indicator
def categorize_socioeconomic(row):
    if row['income_poverty'] == 'above_poverty' and row['employment_st
        return 'High'
    elif row['employment_status'] == 'Employed' and (row['income_pover
        return 'Medium'
    else:
        return 'Low'

df['socioeconomic_status'] = df.apply(categorize_socioeconomic, axis=1
df.head()
```

Out[64]:

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavio
1	1	3.0	2.0	0.0	
7	7	1.0	0.0	0.0	
10	10	2.0	1.0	0.0	
11	11	1.0	2.0	0.0	
15	15	1.0	1.0	0.0	

5 rows × 41 columns

In [65]:
```
# Checking for outliers
#Only select numeric columns
numeric_columns = df.select_dtypes(include=['float','integer']).columr

# Grid layout
rows, cols = 10, 3
fig, axes = plt.subplots(rows, cols, figsize=(20, 13))



# Flatten
axes = axes.flatten()

for i, column in enumerate(numeric_columns):
    sns.boxplot(x=df[column], ax = axes[i])
    axes[i].set_title(f"Box plot for {column}")
    axes[i].set_xlabel(column)
    axes[i].set_ylabel('Value')

# Hide empty subplots
for j in range(i + 1, rows * cols):
    axes[j].axis('off')

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

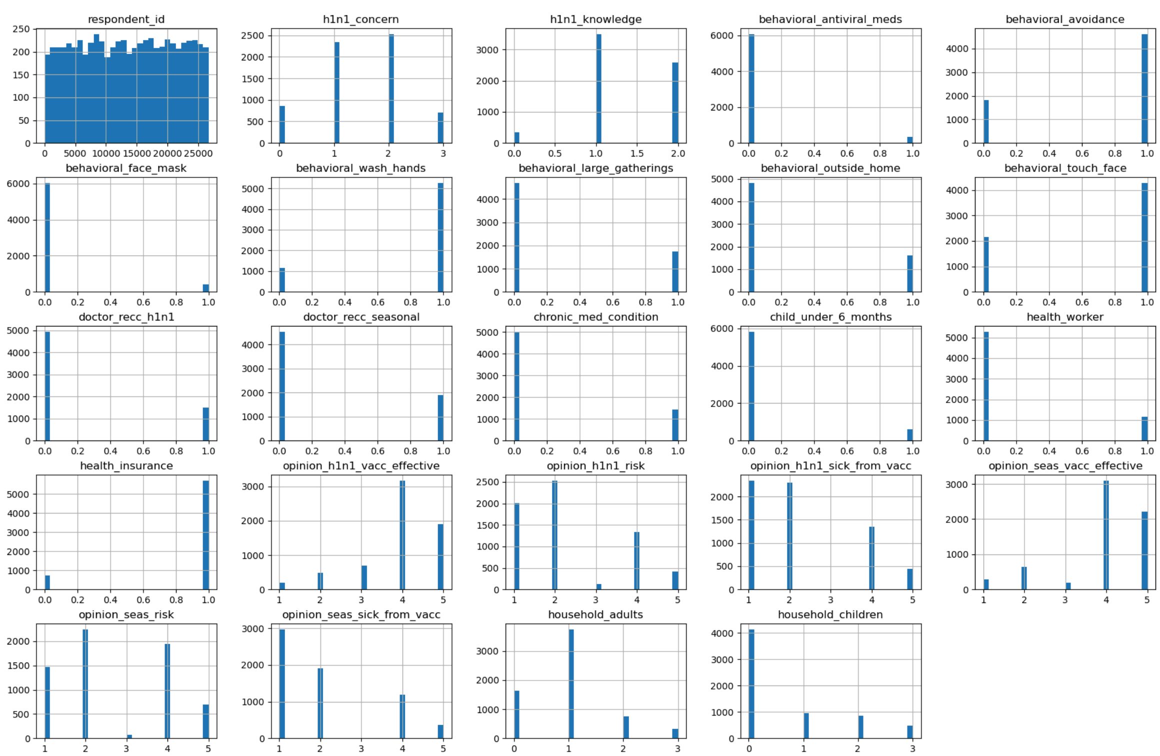


EDA

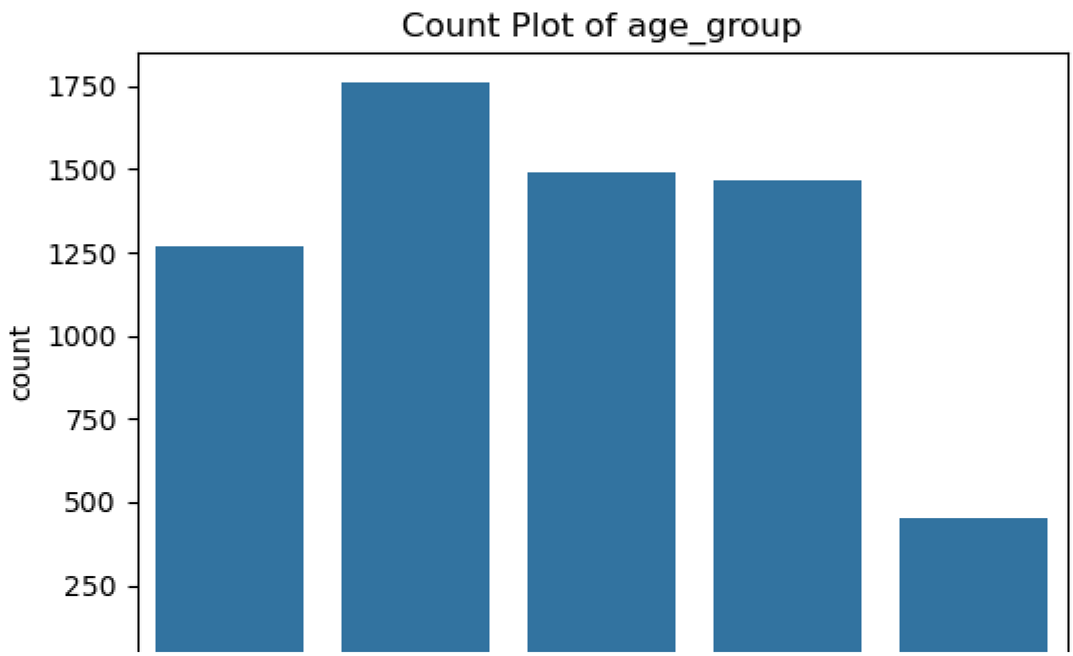
In [66]:
```
df.columns
```

Out[66]:
```
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_seas
onal',
       'chronic_med_condition', 'child_under_6_months', 'health_worke
r',
       'health_insurance', 'opinion_h1n1_vacc_effective', 'opinion_h1
n1_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_statu
s',
       'rent_or_own', 'employment_status', 'hhs_geo_region', 'census_
msa',
       'household_adults', 'household_children', 'employment_industr
y',
       'employment_occupation', 'safety_behavior_score', 'doctor_recc
_total',
       'health_risk_score', 'household_vulnerability', 'socioeconomic
_status'],
      dtype='object')
```

In [52]:
```
# Histograms for numerical variables
df.hist(figsize=(22, 14), bins=30)
plt.show()
```



In [69]:
```
# Count plots for categorical variables
categorical_cols = ['age_group', 'education', 'race', 'sex', 'income_p
                    'marital_status', 'rent_or_own', 'employment_statu
                    'hhs_geo_region', 'census_msa']

for col in categorical_cols:
    plt.figure(figsize=(6, 4))
    sns.countplot(x=df[col])
    plt.title(f"Count Plot of {col}")
    plt.xticks(rotation=45)
    plt.show()
```

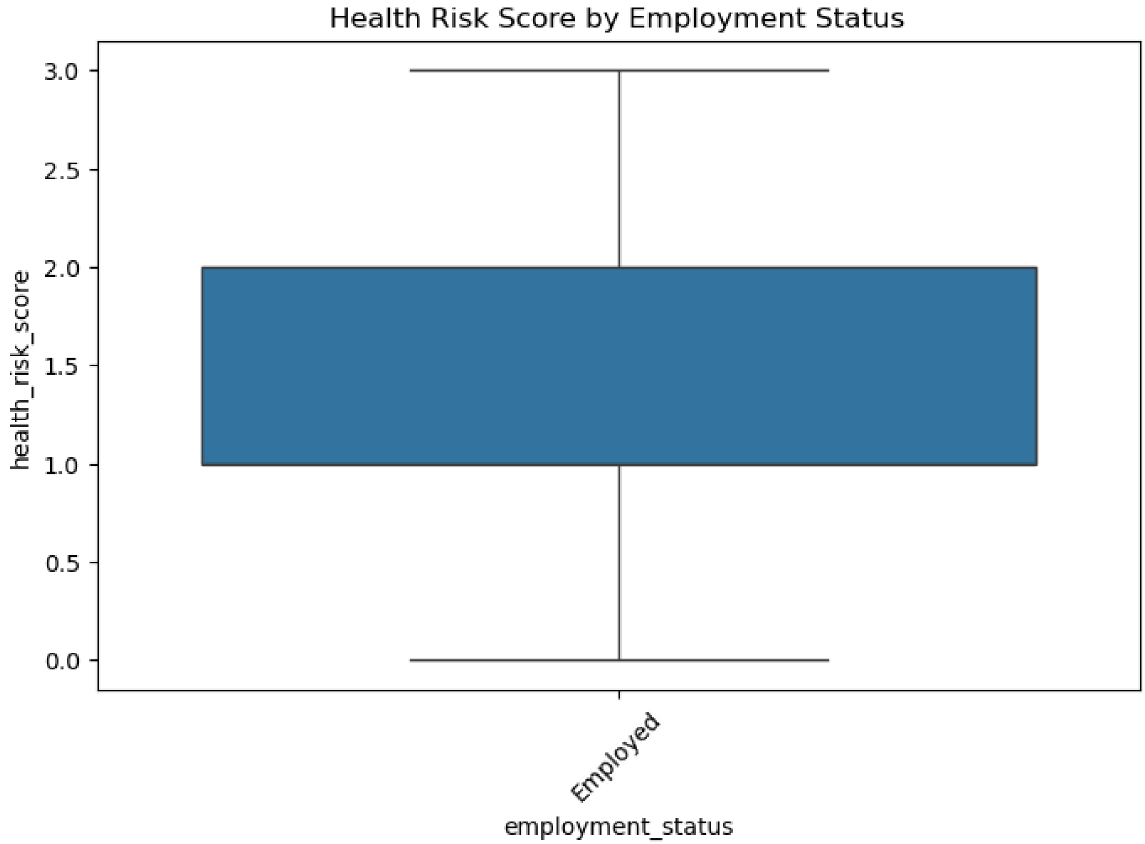



```
Observations
1. Most of the data comprises of the age group 45-54years.
2. college graduates are the majority in the sample.
3. the white race are the majority.
4. females are the major class.
5. majority income is less or equal to 75000$ .
6. most are married.
7. majority own their homes.
```

In [71]:
```
# Scatter plot: Safety Behavior Score vs Health Risk Score
sns.scatterplot(x=df['safety_behavior_score'], y=df['health_risk_score
plt.title('Safety Behavior Score vs Health Risk Score')
plt.show()
```


```
The scatter plot reveals a complex relationship (or lack thereof)
between Safety Behavior Score and Health Risk Score.
The lack of a clear linear trend implies that the two variables are
not strongly related in a direct, linear way.
```

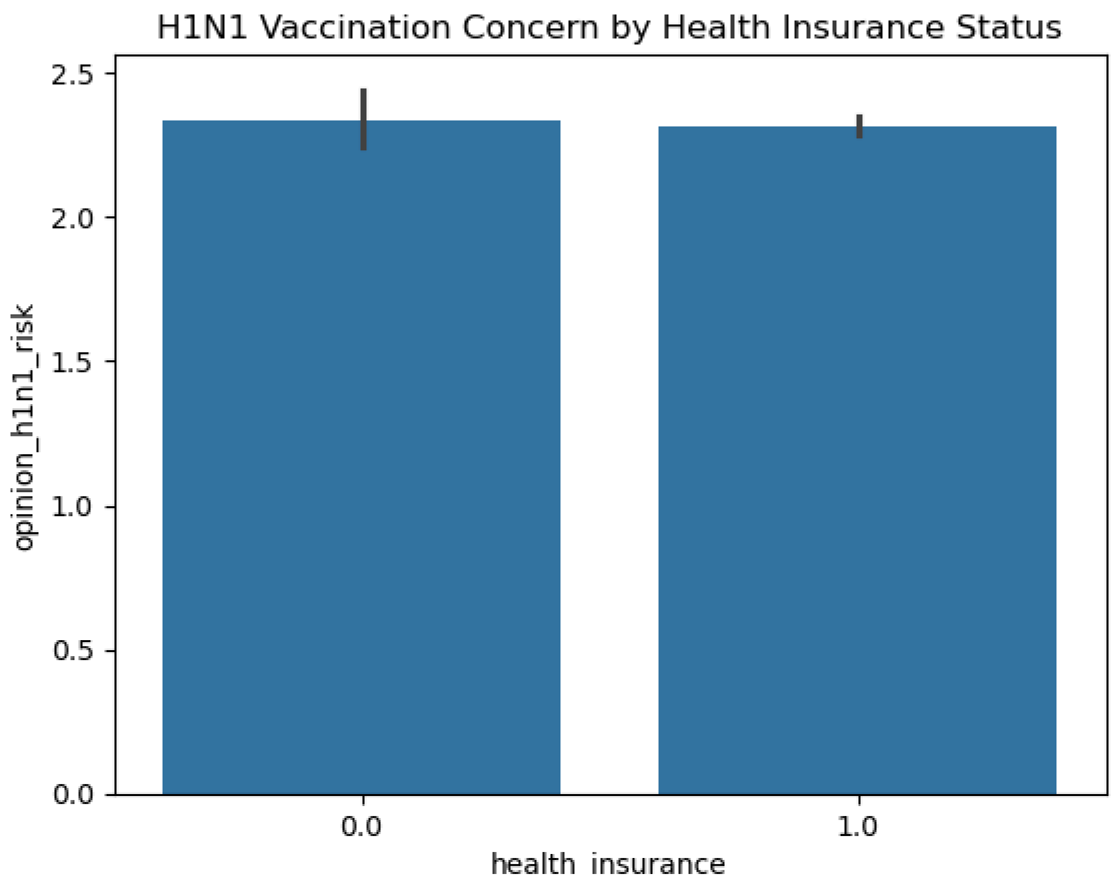
In [72]:
```
# Boxplot: Health Risk Score vs Employment Status
plt.figure(figsize=(8, 5))
sns.boxplot(x=df['employment_status'], y=df['health_risk_score'])
plt.title('Health Risk Score by Employment Status')
plt.xticks(rotation=45)
plt.show()
```



From the above boxplot we can note most employed people are in the average risk score to h1n1 and seasonal flu.

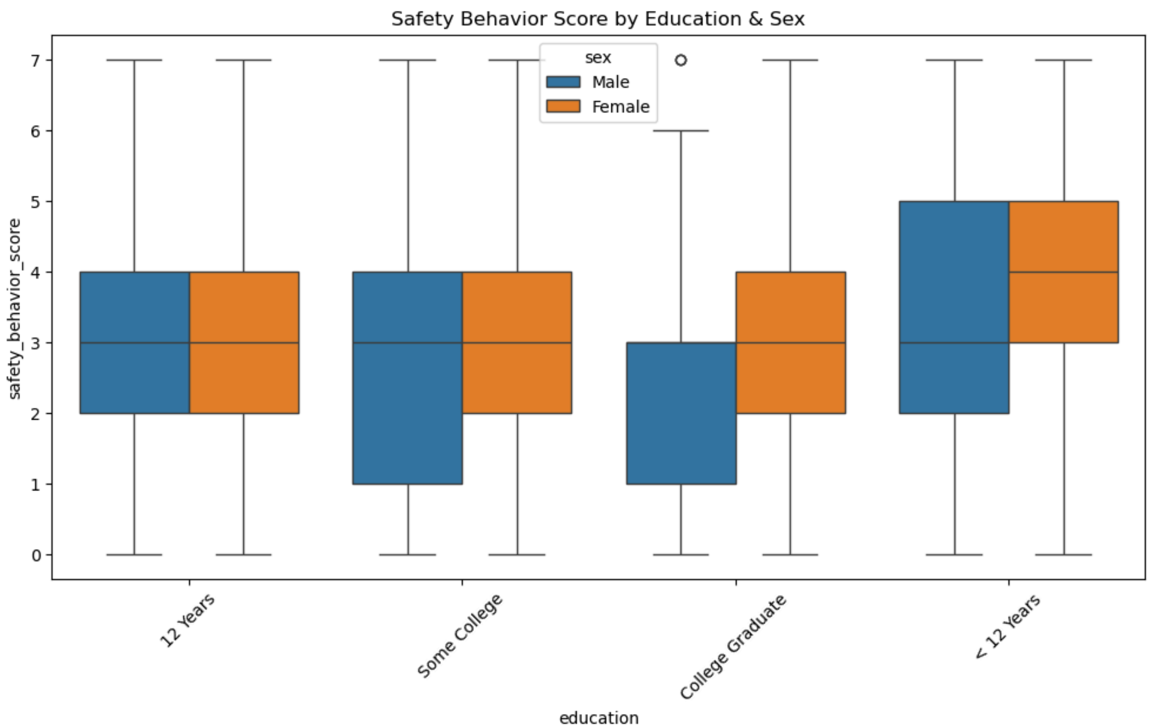
In [73]:
```
# Cross-tabulation of Doctor Recommendation & Vaccination Concern
print(pd.crosstab(df['doctor_recc_h1n1'], df['opinion_h1n1_risk'], nor

# Bar plot: Health Insurance vs H1N1 Vaccination Concern
sns.barplot(x=df['health_insurance'], y=df['opinion_h1n1_risk'])
plt.title("H1N1 Vaccination Concern by Health Insurance Status")
plt.show()
```

```
opinion_h1n1_risk       1.0       2.0       3.0       4.0       5.0
doctor_recc_h1n1
0.0                0.354917  0.411777  0.020842  0.170376  0.042088
1.0                0.174582  0.333779  0.018060  0.330435  0.143144
```



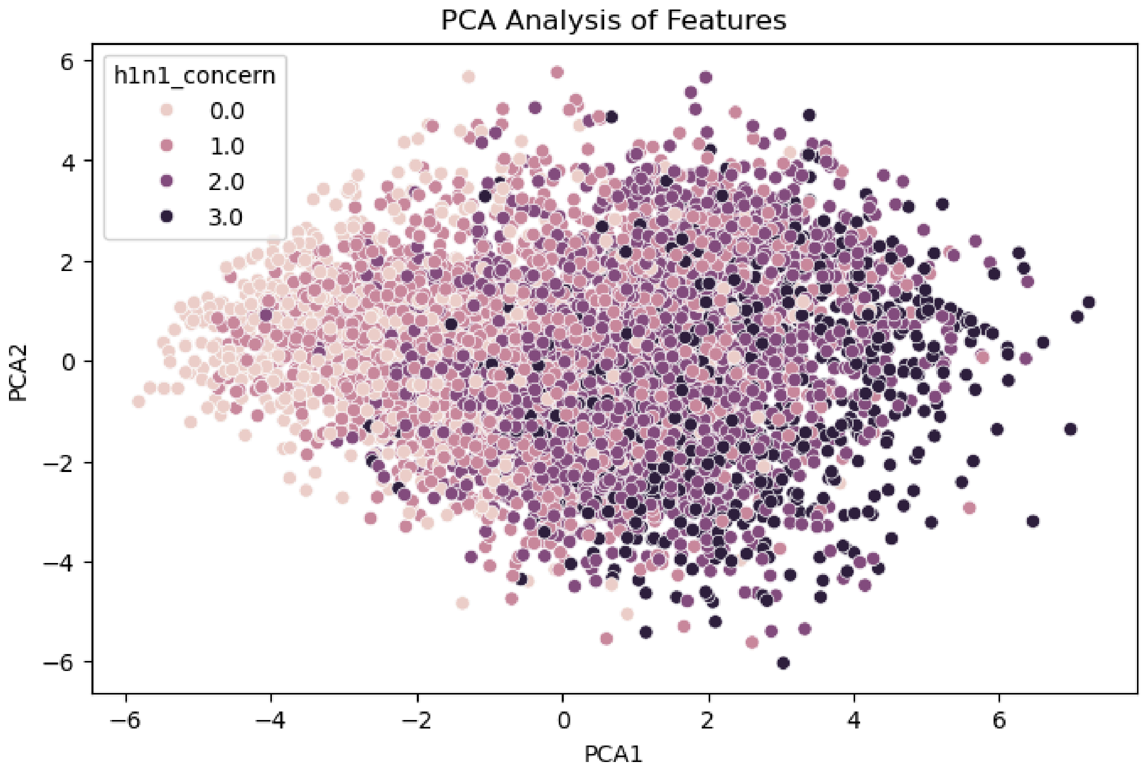

```
# Safety behavior vs education vs sex
```

In [75]:
```
plt.figure(figsize=(12, 6))
sns.boxplot(x='education', y='safety_behavior_score', hue='sex', data=
plt.title("Safety Behavior Score by Education & Sex")
plt.xticks(rotation=45)
plt.show()
```




```
The grouped boxplot reveals a strong positive association between
education level and safety behavior scores, with higher education
generally linked to better safety practices. While sex might play a
minor role, its influence is less pronounced compared to education.
from the above it eveident that in the age of below 12 are the ones
practising safety behavior patterns with females in the lead .
```

In [77]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Standardizing numerical data
numerical_cols = df.select_dtypes(include=['number']).columns.tolist()
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df[numerical_cols])

# Apply PCA
pca = PCA(n_components=2)
pca_result = pca.fit_transform(scaled_data)

# Add PCA results to the DataFrame
df['PCA1'] = pca_result[:, 0]
df['PCA2'] = pca_result[:, 1]

# Scatter plot of PCA results
plt.figure(figsize=(8, 5))
sns.scatterplot(x=df['PCA1'], y=df['PCA2'], hue=df['h1n1_concern'])
plt.title('PCA Analysis of Features')
plt.show()
```



In [ ]:

In [5]:
```
# Modeling
df1 = pd.read_csv('training_set_labels.csv')
df1.head()
```

Out[5]:

	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 [10]:
```
# Merging the data set
df2 = pd.merge(df, df1, on = ['respondent_id'], how = 'left')
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6437 entries, 0 to 6436
Data columns (total 38 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   respondent_id                6437 non-null   int64  
 1   h1n1_concern                 6437 non-null   float64
 2   h1n1_knowledge               6437 non-null   float64
 3   behavioral_antiviral_meds    6437 non-null   float64
 4   behavioral_avoidance         6437 non-null   float64
 5   behavioral_face_mask         6437 non-null   float64
 6   behavioral_wash_hands        6437 non-null   float64
 7   behavioral_large_gatherings  6437 non-null   float64
 8   behavioral_outside_home      6437 non-null   float64
 9   behavioral_touch_face        6437 non-null   float64
 10  doctor_recc_h1n1             6437 non-null   float64
 11  doctor_recc_seasonal         6437 non-null   float64
 12  chronic_med_condition        6437 non-null   float64
 13  child_under_6_months         6437 non-null   float64
 14  health_worker                6437 non-null   float64
 15  health_insurance             6437 non-null   float64
 16  opinion_h1n1_vacc_effective  6437 non-null   float64
 17  opinion_h1n1_risk            6437 non-null   float64
 18  opinion_h1n1_sick_from_vacc  6437 non-null   float64
 19  opinion_seas_vacc_effective  6437 non-null   float64
 20  opinion_seas_risk            6437 non-null   float64
 21  opinion_seas_sick_from_vacc  6437 non-null   float64
 22  age_group                    6437 non-null   object 
 23  education                    6437 non-null   object 
 24  race                         6437 non-null   object 
 25  sex                          6437 non-null   object 
 26  income_poverty               6437 non-null   object 
 27  marital_status               6437 non-null   object 
 28  rent_or_own                  6437 non-null   object 
 29  employment_status            6437 non-null   object 
 30  hhs_geo_region               6437 non-null   object 
 31  census_msa                   6437 non-null   object 
 32  household_adults             6437 non-null   float64
 33  household_children           6437 non-null   float64
 34  employment_industry          6437 non-null   object 
 35  employment_occupation        6437 non-null   object 
 36  h1n1_vaccine                 6437 non-null   int64  
 37  seasonal_vaccine             6437 non-null   int64  
dtypes: float64(23), int64(3), object(12)
memory usage: 1.9+ MB
```

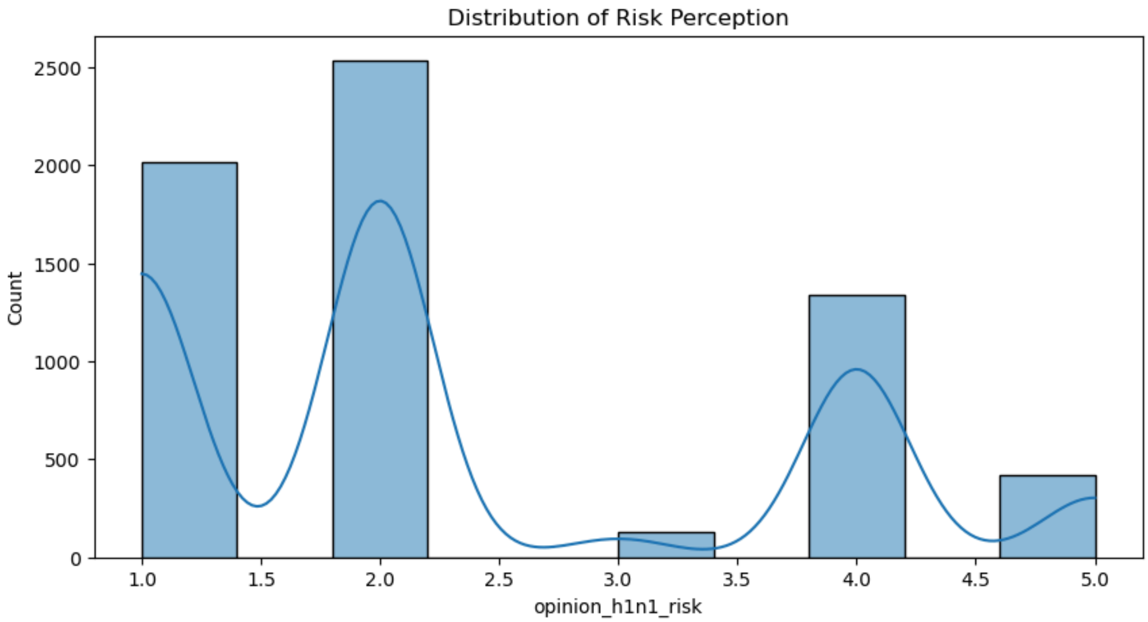
In [67]:
```
df2.columns
```

Out[67]:
```
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_seas
onal',
       'chronic_med_condition', 'child_under_6_months', 'health_worke
r',
       'health_insurance', 'opinion_h1n1_vacc_effective', 'opinion_h1
n1_risk',
       'opinion_h1n1_sick_from_vacc', 'opinion_seas_vacc_effective',
       'opinion_seas_risk', 'opinion_seas_sick_from_vacc', 'household
_adults',
       'household_children', 'h1n1_vaccine', 'seasonal_vaccine',
       'age_group_35 - 44 Years', 'age_group_45 - 54 Years',
       'age_group_55 - 64 Years', 'age_group_65+ Years',
       'education_< 12 Years', 'education_College Graduate',
       'education_Some College', 'race_Hispanic', 'race_Other or Mult
iple',
       'race_White', 'sex_Male', 'income_poverty_> $75,000',
       'income_poverty_Below Poverty', 'marital_status_Not Married',
       'rent_or_own_Rent', 'hhs_geo_region_bhuqouqj',
       'hhs_geo_region_dqpwygqj', 'hhs_geo_region_fpwskwrf',
       'hhs_geo_region_kbazzjca', 'hhs_geo_region_lrircsnp',
       'hhs_geo_region_lzgpxyit', 'hhs_geo_region_mlyzmhmf',
       'hhs_geo_region_oxchjgsf', 'hhs_geo_region_qufhixun',
       'census_msa_MSA, Principle City', 'census_msa_Non-MSA',
       'employment_industry_atmlpfrs', 'employment_industry_cfqqtus
y',
       'employment_industry_dotnnunm', 'employment_industry_fcxhlnw
r',
       'employment_industry_haxffmxo', 'employment_industry_ldnlell
j',
       'employment_industry_mcubkhph', 'employment_industry_mfikgej
o',
       'employment_industry_msuufmds', 'employment_industry_nduyfde
o',
       'employment_industry_phxvnwax', 'employment_industry_pxcmvdj
n',
       'employment_industry_qnlwzans', 'employment_industry_rucpzii
j',
       'employment_industry_saaquncn', 'employment_industry_vjjrobs
f',
       'employment_industry_wlfvacwt', 'employment_industry_wxleyez
f',
       'employment_industry_xicduogh', 'employment_industry_xqicxuv
e',
       'employment_occupation_ccgxvspp', 'employment_occupation_cmhcx
jea',
       'employment_occupation_dcjcmpih', 'employment_occupation_dlvbw
zss',
       'employment_occupation_emcorrxb', 'employment_occupation_halia
zsg',
       'employment_occupation_hfxkjkmi', 'employment_occupation_hodpv
pew',
       'employment_occupation_kldqjyjy', 'employment_occupation_mxkfn
ird',
```

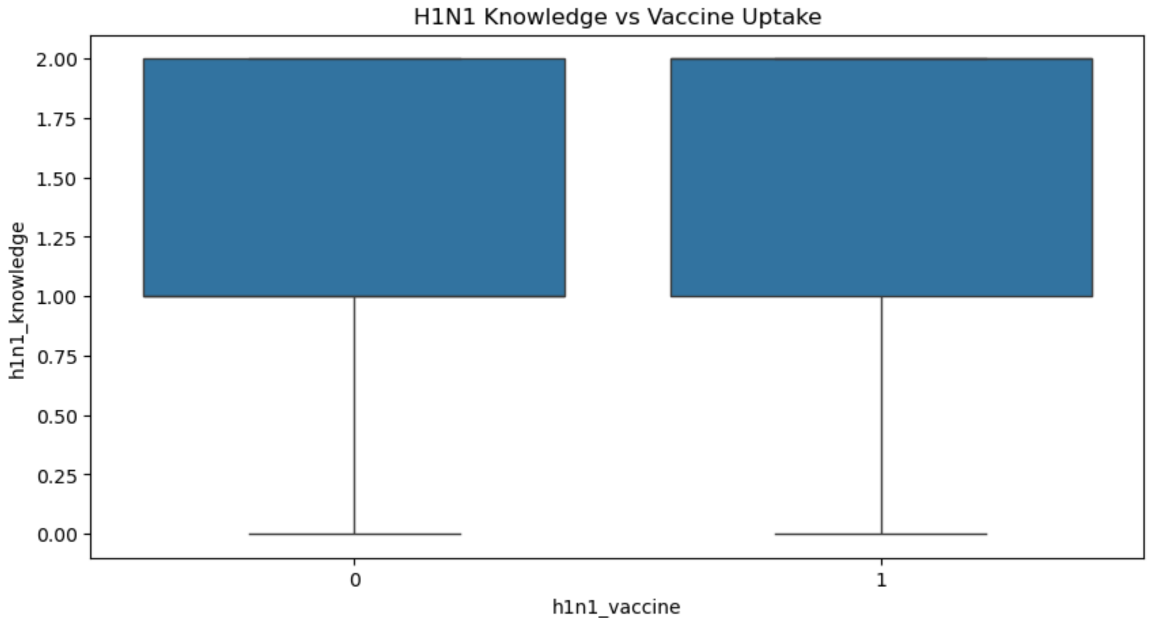
```
       'employment_occupation_oijqvulv', 'employment_occupation_pvmtt
kik',
       'employment_occupation_qxajmpny', 'employment_occupation_rcert
sgn',
       'employment_occupation_tfqavkke', 'employment_occupation_ukymx
vdu',
       'employment_occupation_uqqtjvyb', 'employment_occupation_vlluh
bov',
       'employment_occupation_xgwztkwe', 'employment_occupation_xqwwg
dyp',
       'employment_occupation_xtkaffoo', 'employment_occupation_xzmly
yjv'],
      dtype='object')
```

In [62]:

```
# Risk Perception distribution
plt.figure(figsize=(10, 5))
sns.histplot(df2['opinion_h1n1_risk'], bins=10, kde=True)
plt.title("Distribution of Risk Perception")
plt.show()
```



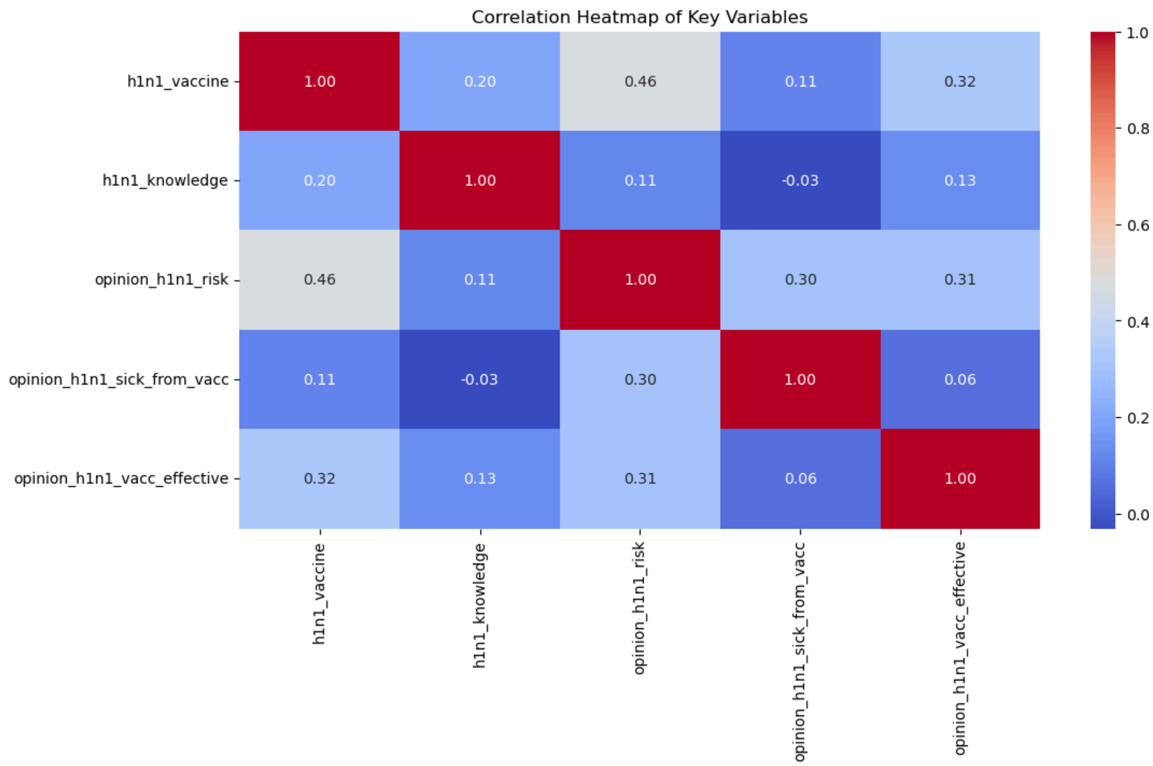
In [63]:
```
# H1N1 knowledge vs Vaccine uptake
plt.figure(figsize=(10, 5))
sns.boxplot(x=df2['h1n1_vaccine'], y=df2['h1n1_knowledge'])
plt.title("H1N1 Knowledge vs Vaccine Uptake")
plt.show()
```




```
from the above we can h1n1 knowledge has not significant impact on
h1n1 vaccine uptake
```

Correlation heatmap of key variables

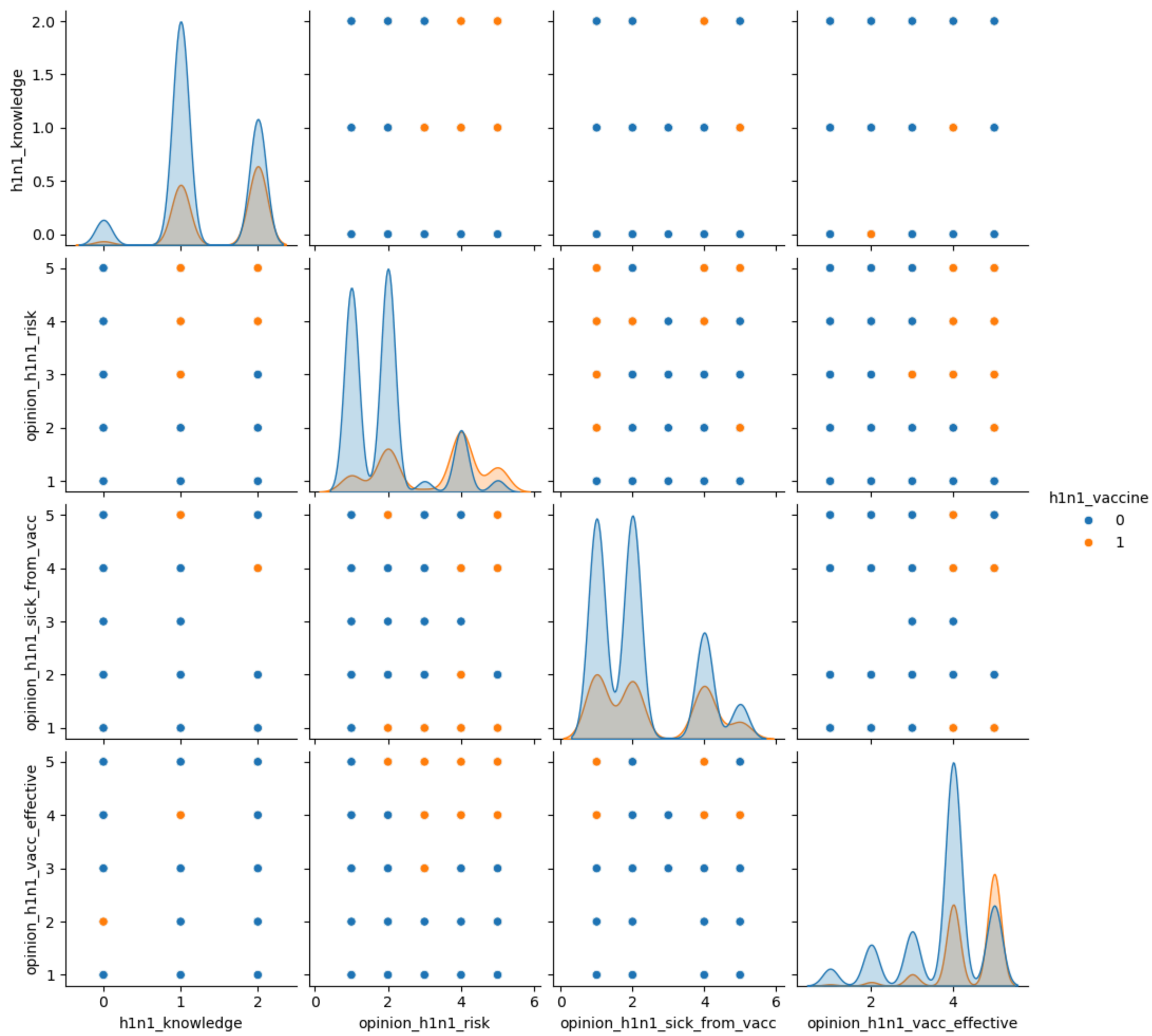
In [64]:
```
plt.figure(figsize=(12, 6))
sns.heatmap(df2[['h1n1_vaccine', 'h1n1_knowledge', 'opinion_h1n1_risk'
                'opinion_h1n1_vacc_effective']].corr(), annot=True, cm
plt.title("Correlation Heatmap of Key Variables")
plt.show()
```




```
Key Observations and Interpretations:

h1n1_vaccine Correlations:
Strong Positive with opinion_h1n1_vacc_effective (0.32): People who
believed the vaccine was effective were more likely to get
vaccinated. This is a logical and expected relationship.
Moderate Positive with opinion_h1n1_risk (0.46): Individuals who
perceived a higher risk of getting H1N1 were more likely to be
vaccinated. This also makes intuitive sense.
Weak Positive with h1n1_knowledge (0.20): Those with more knowledge
about H1N1 were slightly more likely to get vaccinated, but the
relationship is not as strong.
Weak Positive with opinion_h1n1_sick_from_vacc (0.11): Interestingly,
there's a slight tendency for people concerned about getting sick
from the vaccine to still get vaccinated. This might seem
counterintuitive but could suggest that even with concerns,
individuals might have perceived the risk of H1N1 as greater or
trusted official recommendations.

Positive Correlation between opinion_h1n1_risk and both
opinion_h1n1_sick_from_vacc (0.30) and opinion_h1n1_vacc_effective
(0.31): People who perceived a higher risk also tended to be more
concerned about getting sick from the vaccine and had more positive
views on its effectiveness.
```

```
Weak Negative Correlation between h1n1_knowledge and
opinion_h1n1_sick_from_vacc (-0.03): Slightly, those with more
knowledge were less concerned about getting sick from the vaccine.
```

In [65]:
```python
sns.pairplot(df2[['h1n1_knowledge', 'opinion_h1n1_risk', 'opinion_h1n1
                  'opinion_h1n1_vacc_effective', 'h1n1_vaccine']], hue=
plt.show()
```




```
Observations:

Diagonal Plots :

h1n1_vaccine: Shows the overall proportion of vaccinated vs.
unvaccinated individuals.
h1n1_knowledge: Suggests a somewhat normal distribution, possibly
with a slight positive skew.
opinion_h1n1_risk: Appears to be skewed, with most people rating risk
relatively low.
opinion_h1n1_sick_from_vacc: Also skewed, with a concentration of
responses indicating low concern about getting sick from the vaccine.
opinion_h1n1_vacc_effective: Shows a distribution concentrated
towards higher effectiveness ratings.
Off-Diagonal Plots (Relationships):
```

```
h1n1_vaccine vs. Other Variables:
h1n1_knowledge: A weak positive relationship might exist, with
vaccinated individuals tending to have slightly higher knowledge
scores on average.
opinion_h1n1_risk: A more noticeable positive relationship. People
who perceived a higher risk were more likely to be vaccinated.
opinion_h1n1_sick_from_vacc: Possibly a very weak positive
relationship, if any.
opinion_h1n1_vacc_effective: A moderate positive relationship. People
with more positive opinions about vaccine effectiveness were more
likely to be vaccinated.
Relationships Among Opinions and Knowledge:
opinion_h1n1_risk vs. opinion_h1n1_sick_from_vacc and
opinion_h1n1_vacc_effective: Positive relationships appear to exist.
People who perceived higher risk were also more likely to be
concerned about getting sick from the vaccine and to believe in its
effectiveness.
h1n1_knowledge vs. opinion_h1n1_risk: Possibly a weak positive
relationship.

Key Insights and Observations:
Vaccine Hesitancy Insights: The plot provides visual evidence of
factors associated with vaccine uptake. The strongest relationships
appear to be between vaccination status and perceived risk and belief
in vaccine effectiveness.
```

In [76]:

```python
# Define the save path
save_path = "C:/Users/ADMIN/OneDrive/Desktop/machine learning datascie


# Save the DataFrame to CSV
df2.to_csv(save_path, index=False)

print(f"DataFrame has been saved to {save_path}")
```

```
DataFrame has been saved to C:/Users/ADMIN/OneDrive/Desktop/machine l
earning datascience/phase3_project/H1N1_analysis/cleaned_data.csv
```

In [12]:

```python
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix , classif
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier ,plot_tree
```

Preprocessing

In [16]:
```
cat_cols = df2.select_dtypes(include=['object']).columns
cat_cols
```

Out[16]:
```
Index(['age_group', 'education', 'race', 'sex', 'income_poverty',
       'marital_status', 'rent_or_own', 'employment_status', 'hhs_geo
_region',
       'census_msa', 'employment_industry', 'employment_occupation'],
      dtype='object')
```

In [20]:
```
from sklearn.preprocessing import OneHotEncoder

# Create a list of columns to encode
categorical_columns = ['age_group', 'education', 'race', 'sex', 'incom
       'marital_status', 'rent_or_own', 'employment_status', 'hhs_geo_
       'census_msa', 'employment_industry', 'employment_occupation']

# Create a copy of the DataFrame with the selected columns
encoded_df2 = df2.copy()

# Create an instance of OneHotEncoder
# sparse=False to produce a dense array and drop='first' to drop the f
encoder = OneHotEncoder(sparse_output=False, drop='first')

# Iterate through each categorical column

for column in categorical_columns:
    # Fit and transform the selected column
    one_hot_encoded = encoder.fit_transform(encoded_df2[[column]])

    # Create a DataFrame with one-hot encoded columns
    one_hot_df = pd.DataFrame(one_hot_encoded, columns=encoder.get_fea

    # Concatenate the one-hot encoded DataFrame with the original Data
    encoded_df2 = pd.concat([encoded_df2, one_hot_df], axis=1)


    # Drop the original categorical column
    encoded_df2 = encoded_df2.drop([column], axis=1)

# Display the resulting DataFrame
df2= encoded_df2.copy()

df2.head()
```

Out[20]:

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavio
0	1	3.0	2.0	0.0	
1	7	1.0	0.0	0.0	
2	10	2.0	1.0	0.0	
3	11	1.0	2.0	0.0	
4	15	1.0	1.0	0.0	

5 rows × 94 columns

In [24]:
```
# Split data into training and test sets
x = df2.drop(columns=['h1n1_vaccine', 'seasonal_vaccine', 'respondent_
y = df2['h1n1_vaccine']
```

In [25]:
```
# split dataset into train and test split
x_train,x_test,y_train,y_test = train_test_split(x,y , test_size= 0.2,
```

In [26]:
```
print(x_train.shape,x_test.shape,y_train.shape,y_test.shape)
```

```
(5149, 91) (1288, 91) (5149,) (1288,)
```

In [27]:
```
# Initialize the scaler
scaler = StandardScaler()

# Fit and transform training data
x_train_scaled = scaler.fit_transform(x_train)

# Transform  test sets
x_test_scaled = scaler.transform(x_test)
```

In [35]:
```
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassif
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_sc
import time
import numpy as np
import pandas as pd

# Define the models
models = {
    'Decision Tree': DecisionTreeClassifier(),
    'Extra Trees': ExtraTreesClassifier(),
    'Random Forest': RandomForestClassifier(),
    'Gradient Boosting': GradientBoostingClassifier(),
    'Logistic Regression': LogisticRegression()
}

# Initialize a dictionary to store results
results = {
    'Model': [], 'Accuracy': [], 'Precision': [], 'Recall': [], 'F1-sc
    'Training Time (s)': [], 'Prediction Time (s)': []
}

# Create a loop to iterate over the models
for model_name, model in models.items():
    # Measure the training time
    start_time = time.time()
    model.fit(x_train_scaled, y_train)
    training_time = time.time() - start_time

    # Measure the prediction time
    start_time = time.time()
    y_pred = model.predict(x_test_scaled)
    prediction_time = time.time() - start_time

    # Evaluating the model
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='weighted')
    recall = recall_score(y_test, y_pred, average='weighted')
    f1 = f1_score(y_test, y_pred, average='weighted')

    # ROC AUC only works for binary classification, so we check before
    if len(np.unique(y_test)) == 2:
        roc_auc = roc_auc_score(y_test, model.predict_proba(x_test_sca
    else:
        roc_auc = np.nan  # Not applicable for multi-class

    # Store results in the dictionary
    results['Model'].append(model_name)
    results['Accuracy'].append(accuracy)
    results['Precision'].append(precision)
    results['Recall'].append(recall)
    results['F1-score'].append(f1)
    results['ROC AUC'].append(roc_auc)
    results['Training Time (s)'].append(training_time)
    results['Prediction Time (s)'].append(prediction_time)
```

```
# Create a DataFrame for results
results_df2 = pd.DataFrame(results)

# Display the results
print(results_df2)
```

```
                 Model  Accuracy  Precision    Recall  F1-score   ROC
AUC  \
0        Decision Tree  0.733696   0.736882  0.733696  0.735188  0.68
4250
1          Extra Trees  0.829193   0.824122  0.829193  0.821843  0.86
1398
2        Random Forest  0.828416   0.823146  0.828416  0.821542  0.86
2137
3    Gradient Boosting  0.830745   0.826010  0.830745  0.826653  0.87
3427
4  Logistic Regression  0.830745   0.827035  0.830745  0.828142  0.87
6953

   Training Time (s)  Prediction Time (s)
0           0.158936             0.002064
1           2.286785             0.060160
2           1.999319             0.052783
3           3.233816             0.006033
4           0.087466             0.001162
```

In [36]:
```
import plotly.express as px

# Plot grouped bar chart for classification metrics
fig = px.bar(results_df2, x='Model',
             y=['Accuracy', 'Precision', 'Recall', 'F1-score', 'ROC AU
             labels={'value': 'Metric Value'},
             title='Comparison of Classification Models',
             barmode='group')

# Update layout
fig.update_layout(
    width=1000,  # Set the width of the entire plot
    height=700,  # Set the height of the entire plot
    bargap=0.2,  # Set the gap between bars
)

fig.show()
```

Logistic Regression

In [37]:
```
from sklearn.model_selection import StratifiedKFold, cross_val_score
from sklearn.linear_model import LogisticRegression
import numpy as np

# Define Stratified K-Fold cross-validation
kf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# Define the Logistic Regression model
lg_model = LogisticRegression(random_state=42)

# Perform cross-validation and compute the ROC AUC score
cv_scores = cross_val_score(lg_model, x_train_scaled, y_train, cv=kf,

# Print cross-validation results
print("Cross-validation ROC AUC Scores:", cv_scores)
print("Mean ROC AUC Score:", np.mean(cv_scores))
```

```
Cross-validation ROC AUC Scores: [0.87179211 0.89598957 0.87916935 0.
88129419 0.86297407 0.87536605
 0.87884794 0.8810085  0.82867295 0.87891095]
Mean ROC AUC Score: 0.8734025687873294
```

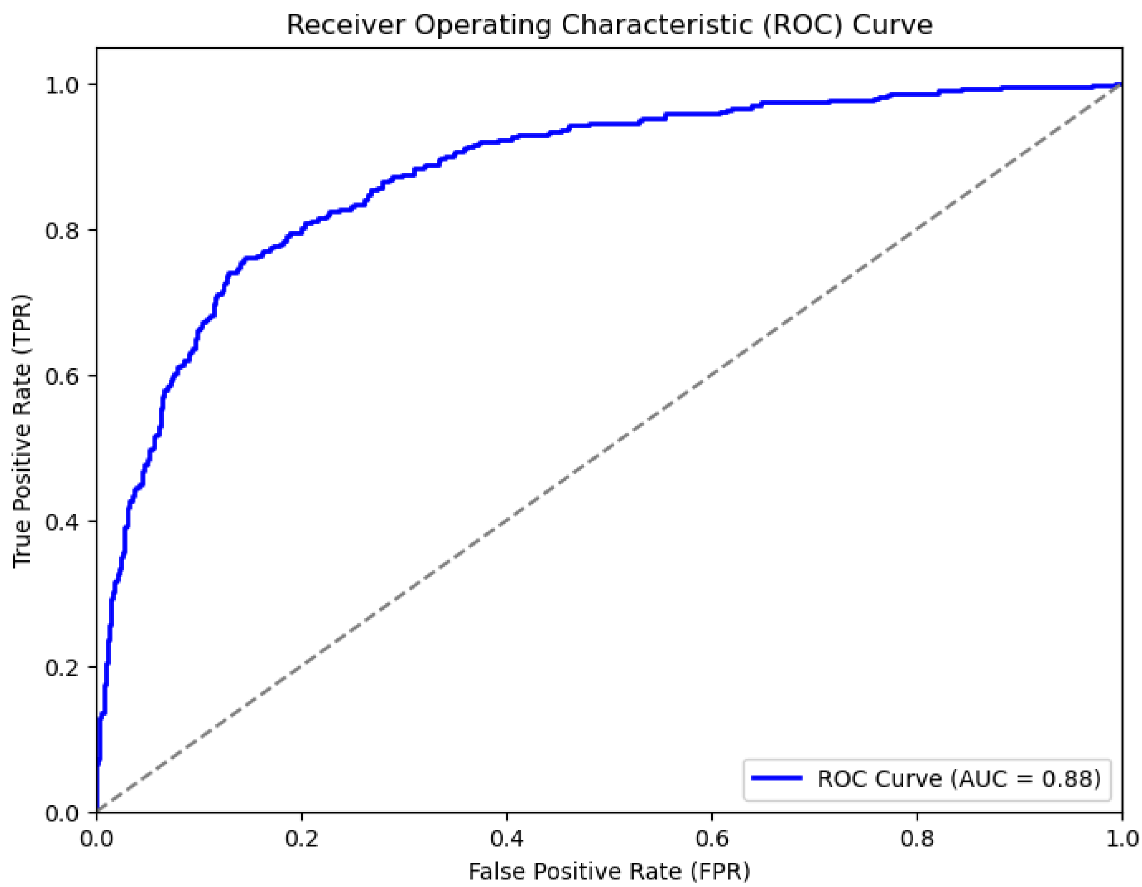
```
Mean roc of  0.8734025687873294 means the model is able to
distinguish between people who actually got the h1n1 vaccine and who
didn't
A 87% score suggests the model is highly confident in its
classifications.
Closer to 1.0 is better, while 0.5 would mean the model is performing
no better than random guessing.
```

In [38]:
```
# Train on training data

lg_model.fit(x_train_scaled, y_train)

# Predict on the validation set
y_pred = lg_model.predict(x_test_scaled)

# Print the classification report
print('Classification Report on Validation Set:')
print(classification_report(y_test, y_pred))

# Print the confusion matrix
print('Confusion Matrix on Validation Set:')
print(confusion_matrix(y_test, y_pred))
```

```
Classification Report on Validation Set:
              precision    recall  f1-score   support

           0       0.87      0.90      0.88       911
           1       0.73      0.66      0.70       377

    accuracy                           0.83      1288
   macro avg       0.80      0.78      0.79      1288
weighted avg       0.83      0.83      0.83      1288

Confusion Matrix on Validation Set:
[[820  91]
 [127 250]]
```

```
Observations
precision 0f 73% means that when the model predicts h1n1 vaccination,
it's correct about 73% of the time.
Higher precision is desirable when false positives (predicting
vaccination when the person actually didn’t get vaccinated) are
costly.
Recall (66%)
this means the model captures about 66% of all who were likely to get
vaccinated.
If recall is low, it means the model is missing many  (false
negatives).
f1-score (70%)
a higher f1 score suggest that the model is doing well.
accuracy (83%)
83% our model has been correct
```

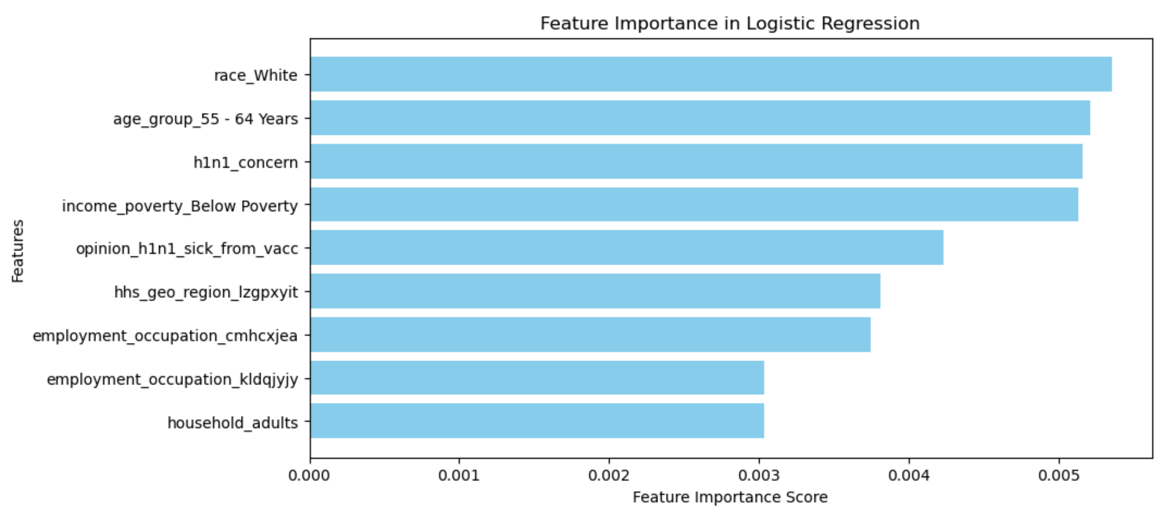
In [59]:
```
from sklearn.linear_model import LogisticRegression

# Step 1: Train the Logistic Regression model
lg_model.fit(x_train_scaled, y_train)

# Step 2: Get probability predictions
y_pred = lg_model.predict_proba(x_test_scaled)[:, 1]  # Get probabilit

# Step 3: Compute and Plot ROC Curve (same as above)
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC Curve (AUC = {roc_a
plt.plot([0, 1], [0, 1], color='grey', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



```
ROC curve indicates the model is performing well as it is hugging the
upper left corner of the graph.
```

In [68]:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression

# Train a Logistic Regression Model
logreg = LogisticRegression(max_iter=500)
logreg.fit(x_train_scaled, y_train)  # Assuming x_train_scaled is star

# Get Absolute Coefficients as Feature Importance
feature_importances = np.abs(logreg.coef_[0])  # Absolute value of coe

# Convert to DataFrame for Visualization
feature_names = x_train.columns  # Use original DataFrame column names
feature_importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': feature_importances
}).sort_values(by='Importance', ascending=False)[16:25]  # Selecting 1

# Plot Feature Importances
plt.figure(figsize=(10, 5))
plt.barh(feature_importance_df2['Feature'], feature_importance_df2['Im
plt.xlabel("Feature Importance Score")
plt.ylabel("Features")
plt.title("Feature Importance in Logistic Regression")
plt.gca().invert_yaxis()  # Invert y-axis for better readability
plt.show()
```



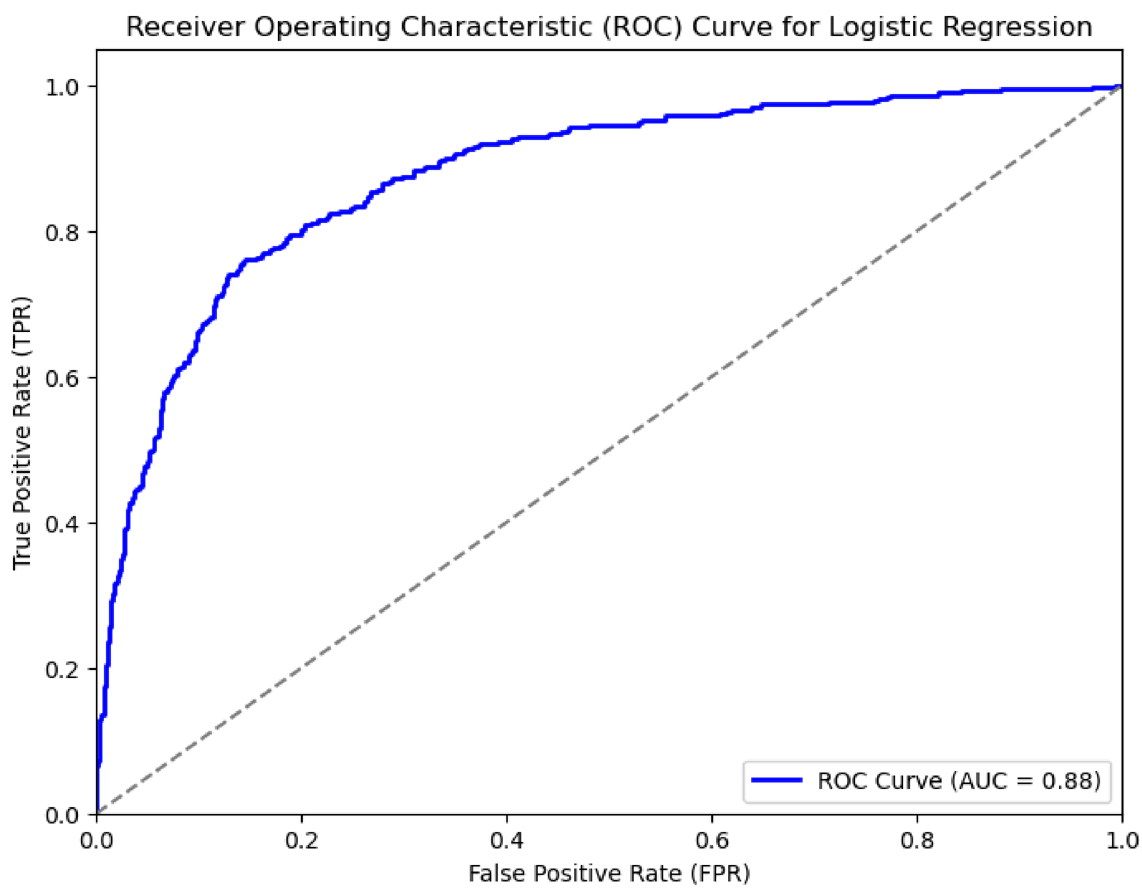

```
Observations:
Top Important Features:
race_White, age_group_55-64_Years, h1n1_concern, and
income_poverty_Below Poverty emerge as the most important features.
They have the longest bars, indicating they contribute most
significantly to the model's predictions.

Moderate Importance Features:
```

```
opinion_h1n1_sick_from_vacc and hhs_geo_region_Izgpxyit have moderate
importance.

Lower Importance Features:
employment_occupation_cmhcxjea, employment_occupation_kldqjyjy, and
household_adults have relatively lower importance scores, suggesting
they play a smaller role in the model's predictive power.

Key Insights:
Demographics and Attitudes: Race, age group, concern about H1N1, and
income level are highly influential in the model. This suggests that
these socio-demographic factors and individual attitudes are strong
predictors of the target variable.
Geographic Influence: Geographic region (hhs_geo_region_Izgpxyit)
also plays a noticeable role.
Occupation and Household: Occupation and household size have a
comparatively smaller impact based on this visualization.
```

In [69]:
```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

# Define the Logistic Regression classifier
logreg = LogisticRegression(random_state=42, solver='saga', max_iter=1

# Define the hyperparameter grid for random search
param_dist = {
    'C': np.logspace(-4, 4, 20),  # Inverse of regularization strength
    'penalty': ['l1', 'l2', 'elasticnet', None],  # Regularization typ
    'l1_ratio': np.linspace(0, 1, 10)  # Only used for elasticnet
}

# Create a RandomizedSearchCV object
random_search = RandomizedSearchCV(
    logreg, param_distributions=param_dist, scoring='f1_macro',
    cv=3, n_iter=20, n_jobs=-1, random_state=42, error_score='raise'
)

# Fit the random search to the data
random_search.fit(x_train_scaled, y_train)

# Get the best model from the random search
best_logreg_model = random_search.best_estimator_

# Make predictions on the testing data
y_pred_logreg = best_logreg_model.predict(x_test_scaled)

# Print the classification report for the best model
print("Classification Report for Best Logistic Regression Model:")
print(classification_report(y_test, y_pred_logreg))

# Print the best hyperparameters found during the search
print("Best Hyperparameters:", random_search.best_params_)
```

```
C:\Users\ADMIN\anaconda3\Lib\site-packages\sklearn\linear_model\_logi
stic.py:1197: UserWarning:

l1_ratio parameter is only used when penalty is 'elasticnet'. Got (pe
nalty=l1)


Classification Report for Best Logistic Regression Model:
              precision    recall  f1-score   support

           0       0.87      0.90      0.88       911
           1       0.73      0.66      0.70       377

    accuracy                           0.83      1288
   macro avg       0.80      0.78      0.79      1288
weighted avg       0.83      0.83      0.83      1288

Best Hyperparameters: {'penalty': 'l1', 'l1_ratio': 0.444444444444444
4, 'C': 4.281332398719396}
```

In [70]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Step 1: Initialize and train the Logistic Regression model
logreg = LogisticRegression(max_iter=1000, random_state=42)
logreg.fit(x_train_scaled, y_train)

# Step 2: Get probability predictions
y_pred_proba = logreg.predict_proba(x_test_scaled)[:, 1]  # Get probab

# Step 3: Compute ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
roc_auc = auc(fpr, tpr)  # Compute AUC score

# Step 4: Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC Curve (AUC = {roc_a
plt.plot([0, 1], [0, 1], color='grey', linestyle='--')  # Dashed diago
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('Receiver Operating Characteristic (ROC) Curve for Logistic
plt.legend(loc="lower right")
plt.show()
```



In [73]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix

# Step 1: Initialize and train the Logistic Regression model
logreg = LogisticRegression(max_iter=1000, random_state=42)
logreg.fit(x_train_scaled, y_train)

# Step 2: Make predictions
y_pred = logreg.predict(x_test_scaled)

# Step 3: Compute confusion matrix
confm = confusion_matrix(y_test, y_pred)

# Display the confusion matrix
print("Confusion Matrix:")
print(confm)
```

```
Confusion Matrix:
[[820  91]
 [127 250]]
```

Gradient Boosting Model

In [40]:
```
from sklearn.model_selection import StratifiedKFold, cross_val_score
from sklearn.ensemble import GradientBoostingClassifier
import numpy as np

# Define Stratified K-Fold cross-validation
kf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# Define the Gradient Boosting model
gb_model = GradientBoostingClassifier(random_state=42)

# Perform cross-validation and compute the ROC AUC score
cv_scores = cross_val_score(gb_model, x_train_scaled, y_train, cv=kf,

# Print cross-validation results
print("Cross-validation ROC AUC Scores:", cv_scores)
print("Mean ROC AUC Score:", np.mean(cv_scores))
```

```
Cross-validation ROC AUC Scores: [0.88100358 0.89315049 0.8695361  0.
8861867  0.88418684 0.8732412
 0.87133062 0.88132991 0.83219056 0.89021475]
Mean ROC AUC Score: 0.8762370756159505
```

```
Mean roc of  0.8762370756159505 means the model is able to
distinguish between people who actually got the h1n1 vaccine and who
didn't
A 87% score suggests the model is highly confident in its
classifications.
```

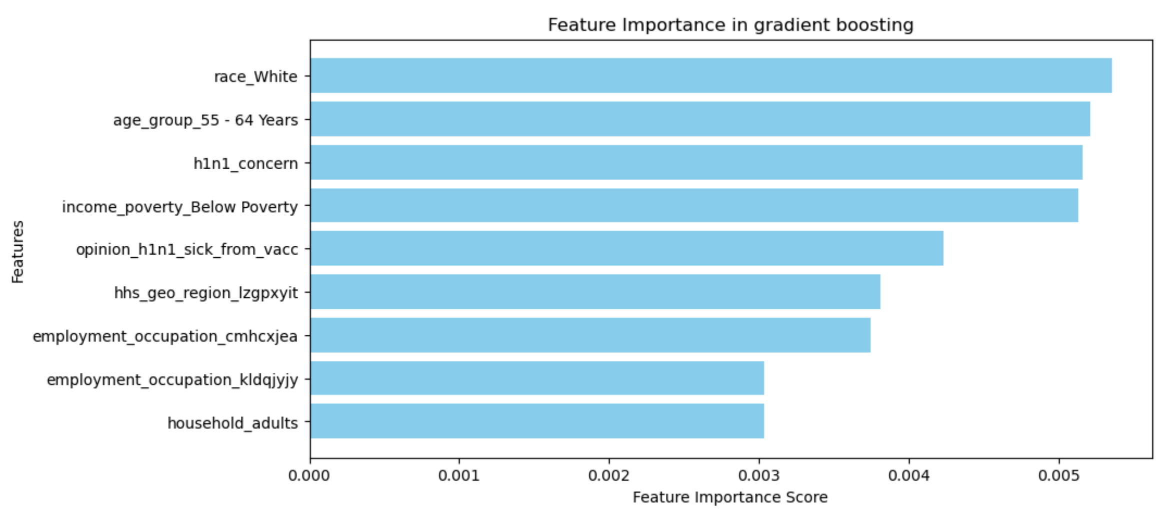
```
Closer to 1.0 is better, while 0.5 would mean the model is performing
no better than random guessing.
```

In [43]:
```python
# Train on training data

gb_model.fit(x_train_scaled, y_train)

# Predict on the validation set
y_pred = gb_model.predict(x_test_scaled)

# Print the classification report
print('Classification Report on Validation Set:')
print(classification_report(y_test, y_pred))

# Print the confusion matrix
print('Confusion Matrix on Validation Set:')
print(confusion_matrix(y_test, y_pred))
```

```
Classification Report on Validation Set:
              precision    recall  f1-score   support

           0       0.86      0.91      0.88       911
           1       0.75      0.64      0.69       377

    accuracy                           0.83      1288
   macro avg       0.80      0.77      0.79      1288
weighted avg       0.83      0.83      0.83      1288

Confusion Matrix on Validation Set:
[[829  82]
 [136 241]]
```

```
Observations
precision 0f 75% means that when the model predicts h1n1 vaccination,
it's correct about 75% of the time.
Higher precision is desirable when false positives (predicting
vaccination when the person actually didn’t get vaccinated) are
costly.
Recall (64%)
this means the model captures about 64% of all who were likely to get
vaccinated.
If recall is low, it means the model is missing many  (false
negatives).
f1-score (69%)
A score of 69% suggests a moderate balance but room for improvement.
accuracy (83%)
83% our model has been correct
```

In [46]:
```
import matplotlib.pyplot as plt
import pandas as pd

# Get Feature Importances
feature_importances = gb_model.feature_importances_

# Convert X_train_scaled to a DataFrame if needed
feature_names = x_train.columns  # Use original DataFrame column names

# Create a DataFrame for better visualization
feature_importance_df2 = pd.DataFrame({
    'Feature': feature_names,
    'Importance': feature_importances
}).sort_values(by='Importance', ascending=False)[16:25]

# Plot Feature Importances
plt.figure(figsize=(10, 5))
plt.barh(feature_importance_df2['Feature'], feature_importance_df2['Im
plt.xlabel("Feature Importance Score")
plt.ylabel("Features")
plt.title("Feature Importance in gradient boosting")
plt.gca().invert_yaxis()  # Invert y-axis for better readability
plt.show()
```




```
Observations:

Top Important Features:
race_White, age_group_55-64_Years, h1n1_concern, and
income_poverty_Below Poverty emerge as the most important features.
They have the longest bars, indicating they contribute most
significantly to the model's predictions.

Moderate Importance Features:
opinion_h1n1_sick_from_vacc and hhs_geo_region_Izgpxyit have moderate
importance.

Lower Importance Features:
employment_occupation_cmhcxjea, employment_occupation_kldqjyjy, and
household_adults have relatively lower importance scores, suggesting
they play a smaller role in the model's predictive power.
```

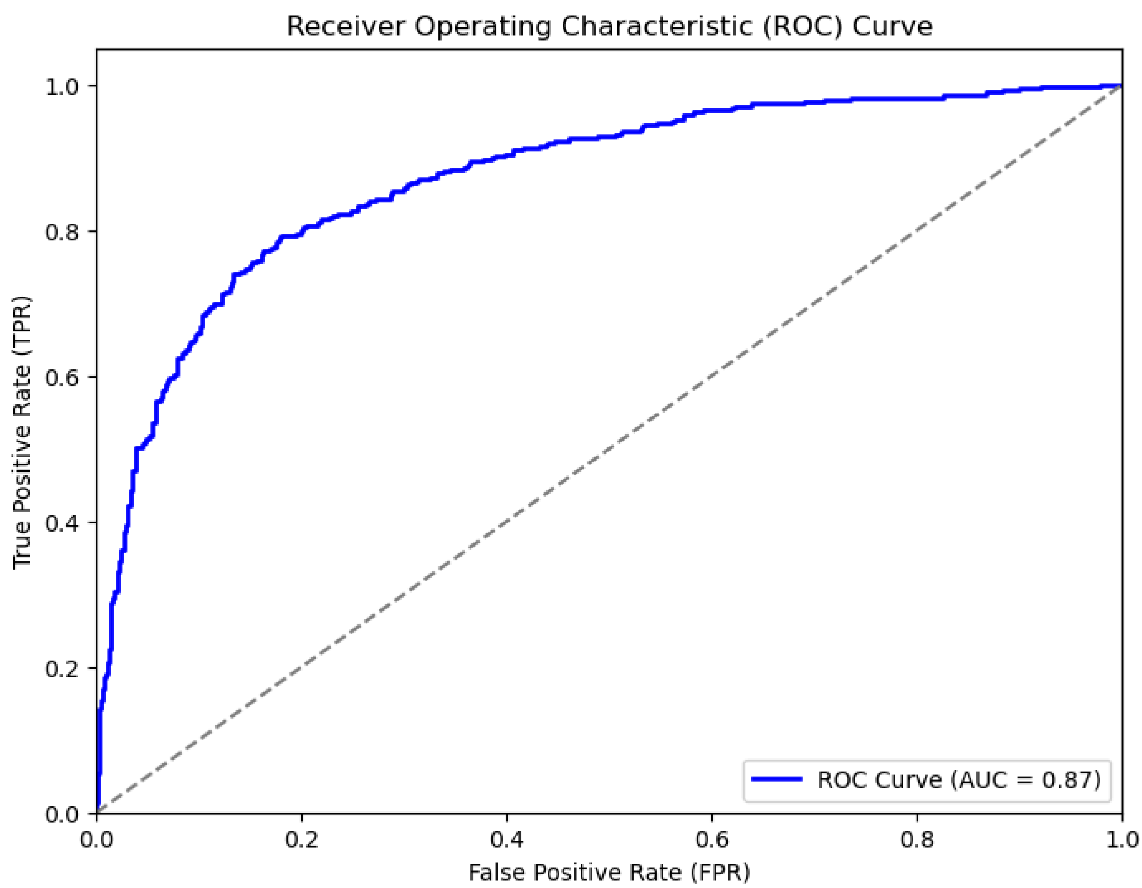
```
Key Insights:

Demographics and Attitudes: Race, age group, concern about H1N1, and
income level are highly influential in the model. This suggests that
these socio-demographic factors and individual attitudes are strong
predictors of the target variable.
Geographic Influence: Geographic region (hhs_geo_region_Izgpxyit)
also plays a noticeable role.
Occupation and Household: Occupation and household size have a
comparatively smaller impact based on this visualization.

In summary: This chart provides valuable insights into which features
drive the gradient boosting model's predictions.  It suggests that
socio-demographic factors and individual attitudes are key
predictors.
```

Hyperparameter search using Randomized search

In [48]:
```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import classification_report
import numpy as np

# Define the Gradient Boosting classifier
gb_classifier = GradientBoostingClassifier(random_state=42)

# Define the hyperparameter grid for random search
param_dist = {
    'n_estimators': [50, 100, 200, 300],
    'learning_rate': [0.01, 0.05, 0.1, 0.2],
    'max_depth': [3, 5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'subsample': [0.7, 0.8, 0.9, 1.0],
    'max_features': ['sqrt', 'log2', None]
}

# Create a RandomizedSearchCV object
random_search = RandomizedSearchCV(
    gb_classifier, param_distributions=param_dist, scoring='f1_macro',
    cv=3, n_iter=20, n_jobs=-1, random_state=42, error_score='raise'
)

# Fit the random search to the data
random_search.fit(x_train_scaled, y_train)

# Get the best model from the random search
best_gb_model = random_search.best_estimator_

# Make predictions on the testing data
y_pred_gb = best_gb_model.predict(x_test_scaled)

# Print the classification report for the best model
print("Classification Report for Best Gradient Boosting Classifier:")
print(classification_report(y_test, y_pred_gb))

# Print the best hyperparameters found during the search
print("Best Hyperparameters:", random_search.best_params_)
```

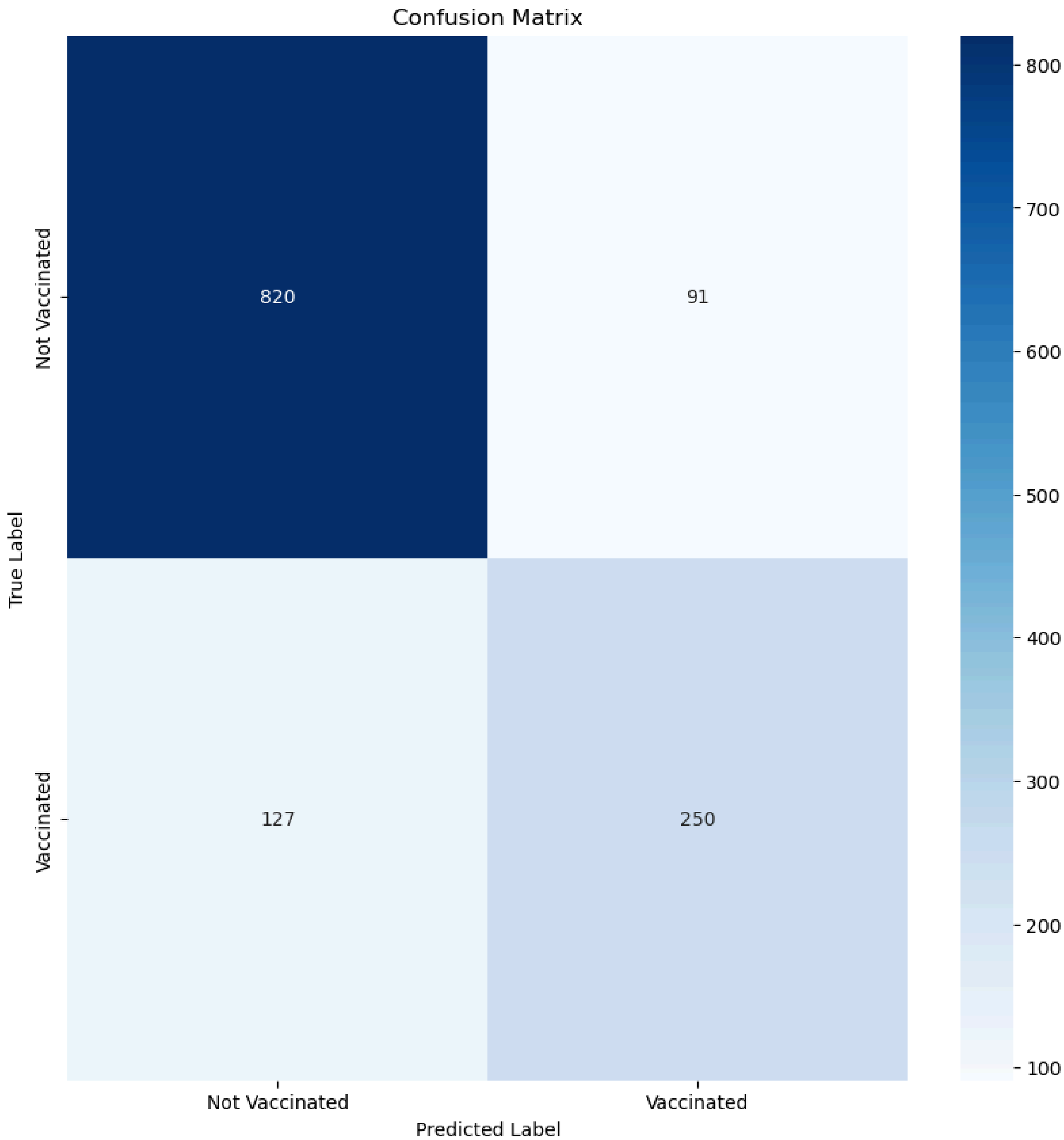
```
Classification Report for Best Gradient Boosting Classifier:
              precision    recall  f1-score   support

           0       0.87      0.90      0.88       911
           1       0.74      0.66      0.70       377

    accuracy                           0.83      1288
   macro avg       0.80      0.78      0.79      1288
weighted avg       0.83      0.83      0.83      1288

Best Hyperparameters: {'subsample': 0.7, 'n_estimators': 300, 'min_sa
mples_split': 5, 'min_samples_leaf': 1, 'max_features': None, 'max_de
pth': 3, 'learning_rate': 0.1}
```

In [55]:
```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Step 1: Initialize and train the Gradient Boosting Classifier
gb_classifier.fit(x_train_scaled, y_train)

# Step 2: Get probability predictions
y_pred = gb_classifier.predict_proba(x_test_scaled)[:, 1]  # Get proba

# Step 3: Compute ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)  # Compute AUC score

# Step 4: Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC Curve (AUC = {roc_a
plt.plot([0, 1], [0, 1], color='grey', linestyle='--')  # Dashed diago
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



In [60]:
```
# Compute confusion matrix
y_pred = model.predict(x_test_scaled)

confm=confusion_matrix(y_test, y_pred)
```

In [66]:
```
# Plot confusion matrix as a heatmap
plt.figure(figsize=(10,10))
sns.heatmap(confm, annot=True, fmt="d", cmap="Blues", xticklabels=["No
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```




```
820 (True Negative): This is the number of individuals who actually
did not get vaccinated, and the model correctly predicted they
wouldn't.
```

91 (False Positive): This is the number of individuals who actually did not get vaccinated, but the model incorrectly predicted they would be vaccinated. (Also known as a Type I error)
127 (False Negative): This is the number of individuals who actually did get vaccinated, but the model incorrectly predicted they would not be vaccinated. (Also known as a Type II error)
250 (True Positive): This is the number of individuals who actually did get vaccinated, and the model correctly predicted they did.

Key Findings in the modeling:
both the logistic regression and gradient boosting performed well using the metrics of roc/auc as our evaluating metrics.
where logistic regression had an roc of 0.876953 and gradient boosting 0.873427 roc.
later performed a cross validation on both models and got an average mean of 0.8734025687873294 in logistic regression and
0.8762370756159505
in gradient boosting model.

Logistic Regression Preferred (Slightly): While both models performed very similarly (ROC AUC around 0.87), logistic regression has a slight edge in cross-validated performance (0.8734 vs 0.8762 for gradient boosting). More importantly, logistic regression offers greater interpretability. In public health contexts, understanding why a model makes a prediction is often as important as the prediction itself. Therefore, unless there's a strong reason to prioritize the slightly higher (but likely not statistically significant) performance of gradient boosting, logistic regression is the recommended choice.

Recommendations
1. Target Educational Interventions:
Tailored Messaging: Develop safety behavior campaigns that are specifically tailored to different education levels. Use language, channels, and examples that resonate with each group. For lower education levels, focus on clear, simple messages and practical demonstrations.
Accessible Information: Ensure that information about vaccines and safety behaviors is easily accessible and available in multiple formats (e.g., videos, infographics, community workshops). Consider language barriers and digital literacy levels.
Community Outreach: Partner with community organizations, schools, and local leaders to disseminate information and build trust, especially in communities with lower education levels.
Interactive Education: Implement interactive educational programs that engage individuals and allow them to practice safety behaviors in a safe environment.

2. Address Vaccine Hesitancy by Emphasizing Effectiveness:
Highlight Success Stories: Share real-life stories of how vaccines have prevented serious illnesses and protected communities.
Trusted Messengers: Utilize trusted figures (doctors, nurses, community leaders) to communicate vaccine effectiveness and safety information.
Address Misinformation: Actively combat misinformation and debunk common myths about vaccines using scientific evidence.

```
Transparency and Openness: Be transparent about the vaccine
development process and any potential side effects. Openly address
concerns and questions.

3. Communicate Risk Clearly and Empathetically:
Personalized Risk Communication: Help individuals understand their
personal risk level based on factors like age, health conditions, and
lifestyle.
Emphasize the Benefits of Vaccination: Clearly communicate how
vaccination reduces the risk of contracting the disease, experiencing
severe symptoms, and spreading it to others.
Address Fears and Concerns: Acknowledge and address common fears and
concerns about vaccines, such as side effects and long-term health
risks.
Use Emotional Appeals Sparingly but Effectively: Use emotional
appeals (e.g., protecting loved ones) judiciously to connect with
individuals on a personal level.

4. Reinforce Belief in Vaccine Effectiveness:
Data-Driven Communication: Share data and statistics on vaccine
effectiveness in a clear and understandable way.
Expert Endorsements: Highlight endorsements from reputable scientific
and medical organizations.
Address Concerns about Safety: Provide information on the rigorous
safety testing that vaccines undergo.
Counter Misinformation: Proactively address and correct
misinformation about vaccine effectiveness.

5. Empower Individuals to Make Informed Decisions:
Provide Balanced Information: Present information about both the
risks and benefits of vaccination, allowing individuals to make
informed decisions.
Encourage Dialogue: Create opportunities for open dialogue and
discussion about vaccines and safety behaviors.
Respect Individual Choices: While strongly encouraging vaccination
and safe behaviors, respect individual choices and avoid coercive
tactics.
```