H1N1 AND SEASONAL FLU VACCINES

Final Project Submission Please fill out:

Student names: Annah Mukethe

Student pace: Part Time

Scheduled project review date/time: PHASE 3

Instructor name: Winnie Anyoso

PROJECT OVERVIEW

A vaccine for the H1N1 flu virus became publicly available in October 2009. In late 2009 and early 2010, the United States conducted the National 2009 H1N1 Flu Survey. This phone survey asked respondents whether they had received the H1N1 and seasonal flu vaccines, in conjunction with questions about themselves. These additional questions covered their social, economic, and demographic background, opinions on risks of illness and vaccine effectiveness, and behaviors towards mitigating transmission. A better understanding of how these characteristics are associated with personal vaccination patterns can provide guidance for future public health efforts.

BUSINESS UNDERSTANDING

INTRODUCTION

The COVID-19 pandemic has highlighted the critical importance of vaccination campaigns in controlling the spread of infectious diseases. Public health efforts are more effective when they are informed by an understanding of the factors that influence individuals' decisions to get vaccinated. By analyzing past vaccination patterns, such as those during the 2009 H1N1 flu pandemic, we can gain valuable insights that can help design better vaccination strategies, improve communication campaigns, and ultimately increase vaccine uptake in future public health crises.

PROBLEM STATEMENT

The task is to predict whether individuals received the H1N1 flu vaccine or the seasonal flu vaccine using data from the 2009 National H1N1 Flu Survey. This binary classification problem involves analyzing various factors, such as demographics, opinions, and health behaviors, to determine their relationship with vaccination behavior. The outcome of this analysis will help public health authorities identify key factors influencing vaccine acceptance and inform strategies to enhance vaccine coverage in the population.

OBJECTIVES

Identify Key Predictors: Determine the most significant factors that influence whether an individual received the H1N1 vaccine.

Develop a Predictive Model: Build a binary classification model to accurately predict whether a survey respondent received the chosen vaccine.

Evaluate Model Performance: Assess the model's performance using appropriate metrics such as accuracy, precision, recall, AUC and F1 score to ensure its reliability in predicting vaccination behavior.

Provide Actionable Insights: Analyze the model's findings to provide public health authorities with actionable insights that can guide future vaccination campaigns and strategies, particularly in the context of managing public health responses to pandemics

DATA SOURCE

The data was sourced from this website [https://www.drivendata.org/competitions/66/flu-shot-learning/ (https://www.drivendata.org/competitions/66/flu-shot-learning/)](https://www.drivendata.org/competitions/66/flu-shot-learning/)

DATASET DESCRIPTION

The dataset, contains the test, train and labels files. The labels data include two target variables;

1. h1n1_vaccine - Whether respondent received H1N1 flu vaccine.
2. seasonal_vaccine - Whether respondent received seasonal flu vaccine.

Both are binary variables: 0 = No; 1 = Yes. Some respondents didn't get either vaccine, others got only one, and some got both. For our case we work with only one target variable which is the h1n1_vaccine. The train and test datasets have 36 columns.The first column respondent_id is a unique and random identifier. The remaining 35 features are described in a separate file feature_description.txt.

METHODOLOGY

The project will follow a structured data science process, including:

Data Inspection: Gathering the necessary data from the provided dataset.

Data Cleaning and Preparation: Cleaning the data to handle missing values, outliers, and incorrect data types.

Exploratory Data Analysis (EDA): Analyzing the data to find patterns, relationships, and insights.

Modeling: Building classification models to predict how likely individuals are to receive their h1n1 vaccines based on selected featurs.

Model Evaluation: Assessing the models' performance using appropriate metrics.

Interpretation: Drawing conclusions from the model results and providing recommendations.

DATA UNDERSTANDING

Importing Libraries

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

Loading Datasets

We will load the train, test and label datasets. The train data will be used to train our model, the test will be used for testing the model performance and the label dataset which has the following target variables will be merged to the train data: h1n1_vaccine - Whether respondent received H1N1 flu vaccine. seasonal_vaccine - Whether respondent received seasonal flu vaccine.

In [2]:
```
#loading the datasets
train_data = pd.read_csv('Data/training_set_features.csv')
test_data = pd.read_csv('Data/test_set_features.csv')
label_data = pd.read_csv('Data/training_set_labels.csv')
```

Data Inspection

In [3]:
```
#inspecting the data sizes of the train,test and label
print('Train shape:', train_data.shape)

print('test shape:', test_data.shape)

print('label shape:', label_data.shape)
```

```
Train shape: (26707, 36)
test shape: (26708, 36)
label shape: (26707, 3)
```

From above results, we see that both the train and test datasets have the same number of rows(26,707) and columns(36). The label data on the other end has also same number of rows which is 26,707 and 3 columns. This is because the label data just contains the target variables and the responded ids.

In [4]:
```
#inspecting the first five rows of the train dataset
train_data.head()
```

Out[4]:

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavioral_avoid
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 [5]:
```
#inspecting the first five rows of the test dataset
test_data.head()
```

Out[5]:

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavioral_avoid
0	26707	2.0	2.0	0.0	
1	26708	1.0	1.0	0.0	
2	26709	2.0	2.0	0.0	
3	26710	1.0	1.0	0.0	
4	26711	3.0	1.0	1.0	

5 rows × 36 columns

From inspecting the first five rows for both the train and test, we can note that, some columns have missing values as well as data that does not have any significant meaning for utilization (eg. lzgpxyit). This is observed in the employment_industry, employment_occupation and the hse_geo_region features. We will further handle this later during preprocessing.

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

Out[6]:

	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

something worth noting from the inspection of the label data is that it conatains a key column which is the respondent_id which is also available in the train dataset. We will use this column to marge the target variables to the train data to aid with our modeling process.

In [7]:
```
#overall inspection of the dataset
train_data.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
```

The dataset contains 26,707 entries with 36 columns.The dataset is a mix of numerical and categorical data types, with 23 columns of type float64, 1 column of type int64, and 12 columns of type object.

Key Observations:

Missing Data:

Several columns contain missing values, which need to be addressed during the data cleaning process. For example:

doctor_recc_h1n1 and doctor_recc_seasonal have a significant amount of missing data, with 24,547 non-null values out of 26,707 entries.

health_insurance has missing values in over 45% of the entries, with only 14,433 non-null values.

The columns employment_industry and employment_occupation have the highest number of missing values, with only 13,377 and 13,237 non-null values, respectively.

Column Types: Based on the data types observed for each column types described below, we will not need to do any data type conversion since sll the columns are in their desired datatype format.

Numeric Columns: There are 23 numeric columns (float64 type) that primarily represent levels of concern, knowledge, and behaviors related to H1N1 and seasonal flu, as well as some demographic information (e.g., household size).

Categorical Columns: The dataset includes 12 categorical columns (object type), such as age_group, education, race, sex, and employment_status. These are critical for demographic analysis and will likely need encoding before modeling.

Identifier Column: respondent_id is a unique identifier for each survey respondent and is of

In [8]:
```
#overall statistics
train_data.describe()
```

Out[8]:

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavioral_a
count	26707.000000	26615.000000	26591.000000	26636.000000	264
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

Binary Variables (0 or 1 values):

Many columns, such as behavioral_antiviral_meds, behavioral_avoidance, behavioral_face_mask, and others, have minimum values of 0 and maximum values of 1, which are typical for binary variables. However, the mean values for these variables are quite low, suggesting that most respondents answered 0 (indicating 'No'). Since the range is limited to 0 and 1, no outliers can be identified in these columns based on the max, min, or mean values.

Ordinal Variables (e.g., h1n1_concern, h1n1_knowledge):

Variables like h1n1_concern and h1n1_knowledge have minimum values of 0, maximum values of 3 or 2, and mean values that do not reach the maximum. This distribution suggests a skew towards lower concern or knowledge, with fewer respondents scoring at the high

end. Since these are ordinal variables, the values themselves are constrained, but if a variable has a mean much closer to the lower end (e.g., h1n1_knowledge with a mean of 1.26 and a max of 2), it suggests a concentration of responses at the lower levels.

Household Variables (household_adults, household_children):

Household variables, such as household_adults and household_children, typically have minimum values of 0 (indicating no adults or children in the household) and maximum values that reflect the upper limits of the survey data. For instance, a higher max value for household_adults could indicate larger family sizes in some households. However, the mean values for these variables tend to be closer to the lower end, suggesting that most households have fewer adults or children. In these cases, extreme values (such as a very high number of adults or children) could be considered outliers if they deviate significantly from the typical household structure, but these outliers might reflect genuine variability in household sizes rather than errors in the data.

In [9]:
```
#getting the column names
train_data.columns
```

Out[9]:
```
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_seasona
l',
       'chronic_med_condition', 'child_under_6_months', 'health_worker',
       'health_insurance', 'opinion_h1n1_vacc_effective', 'opinion_h1n1_ri
sk',
       'opinion_h1n1_sick_from_vacc', 'opinion_seas_vacc_effective',
       'opinion_seas_risk', 'opinion_seas_sick_from_vacc', 'age_group',
       'education', 'race', 'sex', 'income_poverty', 'marital_status',
       'rent_or_own', 'employment_status', 'hhs_geo_region', 'census_msa',
       'household_adults', 'household_children', 'employment_industry',
       'employment_occupation'],
      dtype='object')
```

In [10]:
```
#checking the unique values for some features to determine how to later hand

unique_values = {
    'income_poverty': train_data['income_poverty'].unique(),
    'marital_status': train_data['marital_status'].unique(),
    'rent_or_own': train_data['rent_or_own'].unique(),
    'employment_status': train_data['employment_status'].unique(),
    'education': train_data['education'].unique(),
    'race': train_data['race'].unique(),
    'sex': train_data['sex'].unique(),
    'age_group': train_data['age_group'].unique(),
    'train_data.columns': train_data.columns.tolist()
}

# Print unique values for each feature
for feature, values in unique_values.items():
    print(f"Unique values for {feature}: {values}")
```

```
Unique values for income_poverty: ['Below Poverty' '<= $75,000, Above Pove
rty' '> $75,000' nan]
Unique values for marital_status: ['Not Married' 'Married' nan]
Unique values for rent_or_own: ['Own' 'Rent' nan]
Unique values for employment_status: ['Not in Labor Force' 'Employed' 'Une
mployed' nan]
Unique values for education: ['< 12 Years' '12 Years' 'College Graduate'
'Some College' nan]
Unique values for race: ['White' 'Black' 'Other or Multiple' 'Hispanic']
Unique values for sex: ['Female' 'Male']
Unique values for age_group: ['55 - 64 Years' '35 - 44 Years' '18 - 34 Yea
rs' '65+ Years'
 '45 - 54 Years']
Unique values for train_data.columns: ['respondent_id', 'h1n1_concern', 'h
1n1_knowledge', 'behavioral_antiviral_meds', 'behavioral_avoidance', 'beha
vioral_face_mask', 'behavioral_wash_hands', 'behavioral_large_gatherings',
'behavioral_outside_home', 'behavioral_touch_face', 'doctor_recc_h1n1', 'd
octor_recc_seasonal', 'chronic_med_condition', 'child_under_6_months', 'he
alth_worker', 'health_insurance', 'opinion_h1n1_vacc_effective', 'opinion_
h1n1_risk', 'opinion_h1n1_sick_from_vacc', 'opinion_seas_vacc_effective',
'opinion_seas_risk', 'opinion_seas_sick_from_vacc', 'age_group', 'educatio
n', 'race', 'sex', 'income_poverty', 'marital_status', 'rent_or_own', 'emp
loyment_status', 'hhs_geo_region', 'census_msa', 'household_adults', 'hous
ehold_children', 'employment_industry', 'employment_occupation']
```

From above inspection, we observe the following:

Income Poverty: Categories include 'Below Poverty', '<= $75,000, Above Poverty', '> 75,000', with some missing values (NaN).

Marital Status: Includes 'Not Married', 'Married', with missing values.

Rent or Own: 'Own' vs. 'Rent', with missing values.

Employment Status: 'Not in Labor Force', 'Employed', 'Unemployed', and some missing values.

Education: 'Below 12 Years', '12 Years', 'Some College', 'College Graduate', with missing values.

Race: 'White', 'Black', 'Other or Multiple', 'Hispanic'.

Sex: 'Female', 'Male'.

Age Group: '18 - 34 Years', '35 - 44 Years', '45 - 54 Years', '55 - 64 Years', '65+ Years'.

From this we can decide to impute our missing values with 'unknown' value to avoid any manipulation or we can decide to drop them if the impact is minimal

DATA PREPROCESSING

DATA MERGE

Since we have our target variables in a separate file which is the label dataset, we will need to merge them with the train data before we proceed with preprocessing and plotting of outliers and any normalization required.

We will merge the two datasets using the respondent_id which is present in both files.

In [11]:
```
#merging of the train and the labels datasets
train_df_merged = train_data.merge(label_data, on='respondent_id')
```

In [12]:
```
#checking the data sizes before and after merging
print('Shape before Merging', train_data.shape)
print('Shape after Merging', train_df_merged.shape)
```

```
Shape before Merging (26707, 36)
Shape after Merging (26707, 38)
```

From above output, we see that the data size has changed since we have an addition of 2 columns making the new column number total to 38 while the row sizes remain the same(26,707). We will focus on only one target which is the H1N1 vaccine. So will go ahead and drop the seasonal vaccine column

In [13]:
```
#Dropping seasonal_vaccine target
train_df_merged.drop('seasonal_vaccine', axis=1, inplace=True)
```

In [14]:
```
#checking to ensure the column is dropped and the same  has reflected in our
print(train_df_merged.shape)
```

```
(26707, 37)
```

HANDLING MISSING VALUES

1. First we will get the percentages of missing values for each column
2. We will then impute the missing values either by dropping them or replacing them with another value these could be median or mode or 'Uknown'.

In [15]:
```
#Getting percenatges of missing values
def missing_data_percentage(df):
    missing_data = df.isnull().sum()
    missing_percentage = (missing_data / len(df)) * 100
    return pd.DataFrame({
        'Total Missing Values': missing_data,
        'Percentage Missing': missing_percentage
    })

# Calculate missing data percentages
train_missing = missing_data_percentage(train_df_merged)


# Display missing data percentages
print("Training Data Missing Values:")
print(train_missing.sort_values(by='Percentage Missing', ascending=False))
```

```
Training Data Missing Values:
                             Total Missing Values  Percentage Missing
employment_occupation                       13470           50.436215
employment_industry                         13330           49.912008
health_insurance                            12274           45.957989
income_poverty                               4423           16.561201
doctor_recc_h1n1                             2160            8.087767
doctor_recc_seasonal                         2160            8.087767
rent_or_own                                  2042            7.645936
employment_status                            1463            5.477965
marital_status                               1408            5.272026
education                                    1407            5.268282
chronic_med_condition                         971            3.635751
child_under_6_months                          820            3.070356
health_worker                                 804            3.010447
opinion_seas_sick_from_vacc                   537            2.010709
opinion_seas_risk                             514            1.924589
opinion_seas_vacc_effective                   462            1.729884
opinion_h1n1_sick_from_vacc                   395            1.479013
opinion_h1n1_vacc_effective                   391            1.464036
opinion_h1n1_risk                             388            1.452803
household_children                            249            0.932340
household_adults                              249            0.932340
behavioral_avoidance                          208            0.778822
behavioral_touch_face                         128            0.479275
h1n1_knowledge                                116            0.434343
h1n1_concern                                   92            0.344479
behavioral_large_gatherings                    87            0.325757
behavioral_outside_home                        82            0.307036
behavioral_antiviral_meds                      71            0.265848
behavioral_wash_hands                          42            0.157262
behavioral_face_mask                           19            0.071142
census_msa                                      0            0.000000
respondent_id                                   0            0.000000
hhs_geo_region                                  0            0.000000
sex                                             0            0.000000
race                                            0            0.000000
age_group                                       0            0.000000
h1n1_vaccine                                    0            0.000000
```

HANDLING UNNECCESARY COLUMNS

We will drop the columns that were observed to have data that does not have any significant meaning. These was observed in the following columns:

In [16]:
```
# List of columns to drop
columns_to_drop = [
    'employment_industry',
    'employment_occupation', 'census_msa', 'hhs_geo_region'
]

# Drop the specified columns from the training and test datasets
train_data_cleaned = train_df_merged.drop(columns=columns_to_drop)
```

In [17]:
```
print(train_data_cleaned.shape)
```

```
(26707, 33)
```

In [18]:
```
# Calculate missing data percentages
train_missing2 = missing_data_percentage(train_data_cleaned)


# Display missing data percentages
print("Training Data Missing Values:")
print(train_missing2.sort_values(by='Percentage Missing', ascending=False))
```

```
Training Data Missing Values:
                             Total Missing Values  Percentage Missing
health_insurance                            12274           45.957989
income_poverty                               4423           16.561201
doctor_recc_h1n1                             2160            8.087767
doctor_recc_seasonal                         2160            8.087767
rent_or_own                                  2042            7.645936
employment_status                            1463            5.477965
marital_status                               1408            5.272026
education                                    1407            5.268282
chronic_med_condition                         971            3.635751
child_under_6_months                          820            3.070356
health_worker                                 804            3.010447
opinion_seas_sick_from_vacc                   537            2.010709
opinion_seas_risk                             514            1.924589
opinion_seas_vacc_effective                   462            1.729884
opinion_h1n1_sick_from_vacc                   395            1.479013
opinion_h1n1_vacc_effective                   391            1.464036
opinion_h1n1_risk                             388            1.452803
household_children                            249            0.932340
household_adults                              249            0.932340
behavioral_avoidance                          208            0.778822
behavioral_touch_face                         128            0.479275
h1n1_knowledge                                116            0.434343
h1n1_concern                                   92            0.344479
behavioral_large_gatherings                    87            0.325757
behavioral_outside_home                        82            0.307036
behavioral_antiviral_meds                      71            0.265848
behavioral_wash_hands                          42            0.157262
behavioral_face_mask                           19            0.071142
respondent_id                                   0            0.000000
sex                                             0            0.000000
race                                            0            0.000000
age_group                                       0            0.000000
h1n1_vaccine                                    0            0.000000
```

IMPUTING MISSING VALUES

For this step, we will impute the missing values using the most frequent value for the binary columns and the ordinal columns. As for the categorical columns, to avaoid any manipulation on the user feedbacks, we will handle the missing values by replacing them with the 'UNKNOWN' value

In [19]:

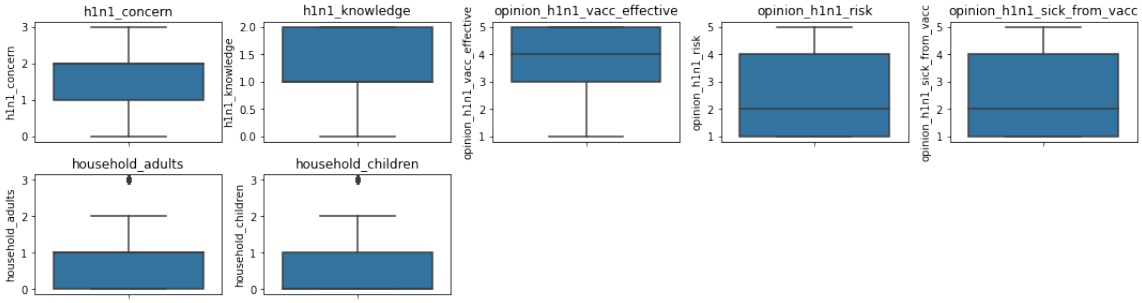
```
#imputing the missing values

from scipy.stats import mode


# List of binary columns (with expected values)
binary_columns = [
    'behavioral_antiviral_meds', 'behavioral_avoidance', 'behavioral_face_ma
    'behavioral_wash_hands', 'behavioral_large_gatherings', 'behavioral_outs
    'behavioral_touch_face', 'doctor_recc_h1n1', 'doctor_recc_seasonal',
    'chronic_med_condition', 'child_under_6_months', 'health_worker','health
]

# List of ordinal columns (with expected discrete values)
ordinal_columns = [
    'h1n1_concern', 'h1n1_knowledge', 'opinion_h1n1_vacc_effective',
    'opinion_h1n1_risk', 'opinion_h1n1_sick_from_vacc', 'opinion_seas_vacc_e
    'opinion_seas_risk', 'opinion_seas_sick_from_vacc'
]

# List of categorical string columns
categorical_columns = [
    'education', 'race', 'sex', 'marital_status', 'rent_or_own',
    'employment_status','income_poverty'

]

# Handle missing values in binary columns with mode
for col in binary_columns:
    mode_value = mode(train_data_cleaned[col].dropna())[0][0]
    train_data_cleaned[col].fillna(mode_value, inplace=True)

# Handle missing values in ordinal columns with mode
for col in ordinal_columns:
    mode_value = mode(train_data_cleaned[col].dropna())[0][0]
    train_data_cleaned[col].fillna(mode_value, inplace=True)

# Handle missing values in categorical string columns with 'UNKNOWN'
for col in categorical_columns:
    train_data_cleaned[col].fillna('UNKNOWN', inplace=True)

# Handle missing values in numeric columns if there are any
numeric_columns = train_data_cleaned.select_dtypes(include=['float64', 'inte
for col in numeric_columns:
    if col not in binary_columns + ordinal_columns:
        mode_value = mode(train_data_cleaned[col].dropna())[0][0]
        train_data_cleaned[col].fillna(mode_value, inplace=True)

# Verify that missing values are handled
print(train_data_cleaned.isnull().sum())
```

```
respondent_id                    0
h1n1_concern                     0
h1n1_knowledge                   0
behavioral_antiviral_meds        0
behavioral_avoidance             0
behavioral_face_mask             0
behavioral_wash_hands            0
behavioral_large_gatherings      0
behavioral_outside_home          0
behavioral_touch_face            0
doctor_recc_h1n1                 0
doctor_recc_seasonal             0
chronic_med_condition            0
child_under_6_months             0
health_worker                    0
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
household_adults                 0
household_children               0
h1n1_vaccine                     0
dtype: int64
```

Outliers and Feature Distribution

In [20]:
```
# List of the columns to check for outliers
features_check_outliers = [
    'h1n1_concern', 'h1n1_knowledge','opinion_h1n1_vacc_effective', 'opinion
    'opinion_h1n1_sick_from_vacc','household_adults', 'household_children'
]
```

In [21]:
```
# Create boxplots
plt.figure(figsize=(15, 10))
for i, col in enumerate(features_check_outliers, 1):
    plt.subplot(5, 5, i)  # Adjust grid size based on number of plots
    sns.boxplot(y=train_data_cleaned[col])
    plt.title(col)

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



Insights from Outlier Analysis

The box plots reveal several interesting patterns across our ordinal variables:

1. H1N1-related variables:
    - Concern levels (h1n1_concern) show a wide range of responses, with some extreme high values.
    - Knowledge about H1N1 (h1n1_knowledge) is generally high, with fewer outliers.
    - Opinions on vaccine effectiveness and risk (opinion_h1n1_vacc_effective, opinion_h1n1_risk) have a more balanced distribution with some outliers on both ends.
2. Seasonal flu variables:
    - Opinions on seasonal flu vaccine effectiveness (opinion_seas_vacc_effective) are generally positive, with some low outliers.
    - Perceived risk of seasonal flu (opinion_seas_risk) shows a wide range of responses.
3. Vaccine side effects:
    - Both H1N1 and seasonal flu vaccines show some outliers for perceived risk of getting sick from the vaccine (opinion_h1n1_sick_from_vacc, opinion_seas_sick_from_vacc), but generally lower values.
4. Household composition:
    - Number of adults and children (household_adults, household_children) show some high outliers, potentially indicating larger families. These outliers may represent genuine extreme cases (e.g., large families, individuals with very strong opinions) or potential data entry errors. Further investigation of these cases could provide valuable insights or inform data cleaning decisions.

In [22]:
```
train_data_cleaned['household_adults'].unique()
```

Out[22]:
```
array([0., 2., 1., 3.])
```

In [23]:
```python
def find_outliers_sum(df, col):
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]
    return outliers[col].sum()

# Get sum of outliers for each column
outliers_sum_dict = {}
for col in features_check_outliers:
    outliers_sum = find_outliers_sum(train_data_cleaned, col)
    if outliers_sum != 0:
        outliers_sum_dict[col] = outliers_sum

# Print sum of outliers
for col, outliers_sum in outliers_sum_dict.items():
    print(f"Sum of outliers in {col}: {outliers_sum}")
```

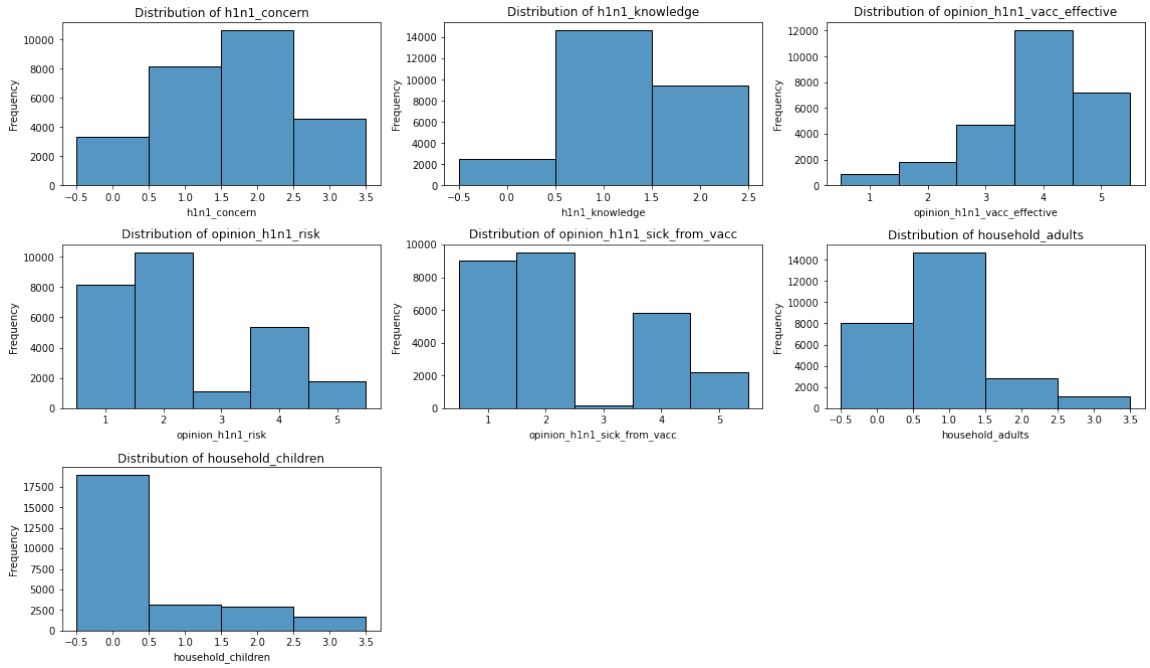
```
Sum of outliers in household_adults: 3375.0
Sum of outliers in household_children: 5241.0
```

In [24]:
```python
# Calculate skewness for the specified columns
skewness = train_data_cleaned[features_check_outliers].skew()

# Print skewness values
print("Skewness for each column:")
print(skewness)
```

```
Skewness for each column:
h1n1_concern                 -0.164127
h1n1_knowledge               -0.234254
opinion_h1n1_vacc_effective  -0.915459
opinion_h1n1_risk             0.688264
opinion_h1n1_sick_from_vacc   0.666603
household_adults              0.785813
household_children            1.558345
dtype: float64
```

In [25]:

```
# Set up the plotting grid
plt.figure(figsize=(16, 12))

for i, feature in enumerate(features_check_outliers, 1):
    plt.subplot(4, 3, i)  # Adjust the grid size if you have more/less featu
    sns.histplot(train_data_cleaned[feature].dropna(), discrete=True, kde=Fa
    plt.title(f'Distribution of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Frequency')

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



Based on the histogram plots provided, here are some observations and insights for each distribution:

h1n1_concern: This distribution appears to be relatively symmetric with no extreme skewness. The data is evenly distributed across several categories, indicating a balanced distribution.

h1n1_knowledge: This distribution is slightly right-skewed, suggesting that higher values in this variable are less frequent. This might indicate that most respondents are clustered around lower values with fewer people in higher ranges.

opinion_h1n1_vacc_effective: Similar to the second plot, this distribution is right-skewed. The majority of the data is concentrated in the lower to middle range, with a few higher values.

Plot 4: This distribution shows a notable right skew, with a higher concentration of respondents in the lower range and fewer respondents at higher values.

opinion_h1n1_risk: This histogram suggests a more uniform distribution across categories, with slight variability between them. There is no strong skewness observed.

opinion_h1n1_sick_from_vacc : This distribution is moderately right-skewed, with a noticeable concentration in the lower values and fewer observations in higher ranges.

opinion_seas_vacc_effective: Similar to the previous plot, this distribution also exhibits a right skew. There are some distinct peaks, possibly indicating subgroups or distinct responses among the population.

opinion_seas_risk: This distribution is highly right-skewed, with a sharp peak in the lower range and a rapid drop-off as values increase. This suggests that most respondents fall within the lower range, with very few outliers.

opinion_seas_sick_from_vacc : This plot also shows a right-skewed distribution, but the skewness is less pronounced than the previous one. There is still a concentration of lower values, but more observations are distributed in the mid-range.

household_adults: The distribution here is right-skewed, with a clear concentration of observations in the lower categories. There are also some mid-range observations, but very few in the higher range.

household_children: This histogram is highly skewed to the right, with an overwhelming majority of the data in the first category. There are very few observations in the other categories. This could indicate a strong preference or a dominant category in the dataset.

General Observations and Considerations:

Right-Skewness: Many of the distributions exhibit right skewness, which suggests that the majority of the data points are concentrated in the lower range of the variable, with a long tail of higher values. This might indicate that certain behaviors or opinions are common among the majority, while extreme or outlier responses are rare.

Potential Outliers: The long tails in some distributions suggest the presence of outliers. These could represent minority opinions or behaviors that could be significant, depending on the context of the analysis.

Distribution Diversity: The diversity in the distribution shapes indicates that different variables capture different aspects of the respondents' opinions, behaviors, or demographics. This suggests that a one-size-fits-all approach to data cleaning (such as outlier removal) may not be appropriate.

Decision on Handling Outliers

Upon examining the outliers in the dataset, particularly in the opinion_seas_vacc_effective, opinion_seas_sick_from_vacc, household_adults, and household_children columns, we have decided not to drop or cap these outliers. The reasoning behind this decision is as follows:

Significance of Behavioral and Opinion Data:

The variables opinion_seas_vacc_effective and opinion_seas_sick_from_vacc capture individuals' perceptions and opinions regarding the effectiveness and potential side effects of the seasonal flu vaccine. Outliers in these columns likely represent strong opinions or unique perspectives that are critical to understanding the full spectrum of public sentiment. Removing these outliers could lead to a loss of valuable insights, particularly in understanding the range of beliefs and attitudes that influence vaccination decisions.

Representation of Household Demographics:

The household_adults and household_children columns describe the composition of households. Outliers in these columns might reflect larger or smaller-than-average families. These data points are crucial for understanding how household size might impact vaccine acceptance, particularly in scenarios where larger families might have different logistical considerations or levels of concern about vaccinations. Dropping these outliers could obscure the analysis by removing genuine and potentially significant variations in household structure.

Preservation of Data Integrity:

Outliers often represent the diversity within the population being studied. In public health research, it is essential to capture this diversity to ensure that any interventions or recommendations are inclusive and applicable to all subgroups within the population. By retaining these outliers, we maintain the integrity of the data, allowing for a more comprehensive analysis that can inform more equitable and effective public health strategies.

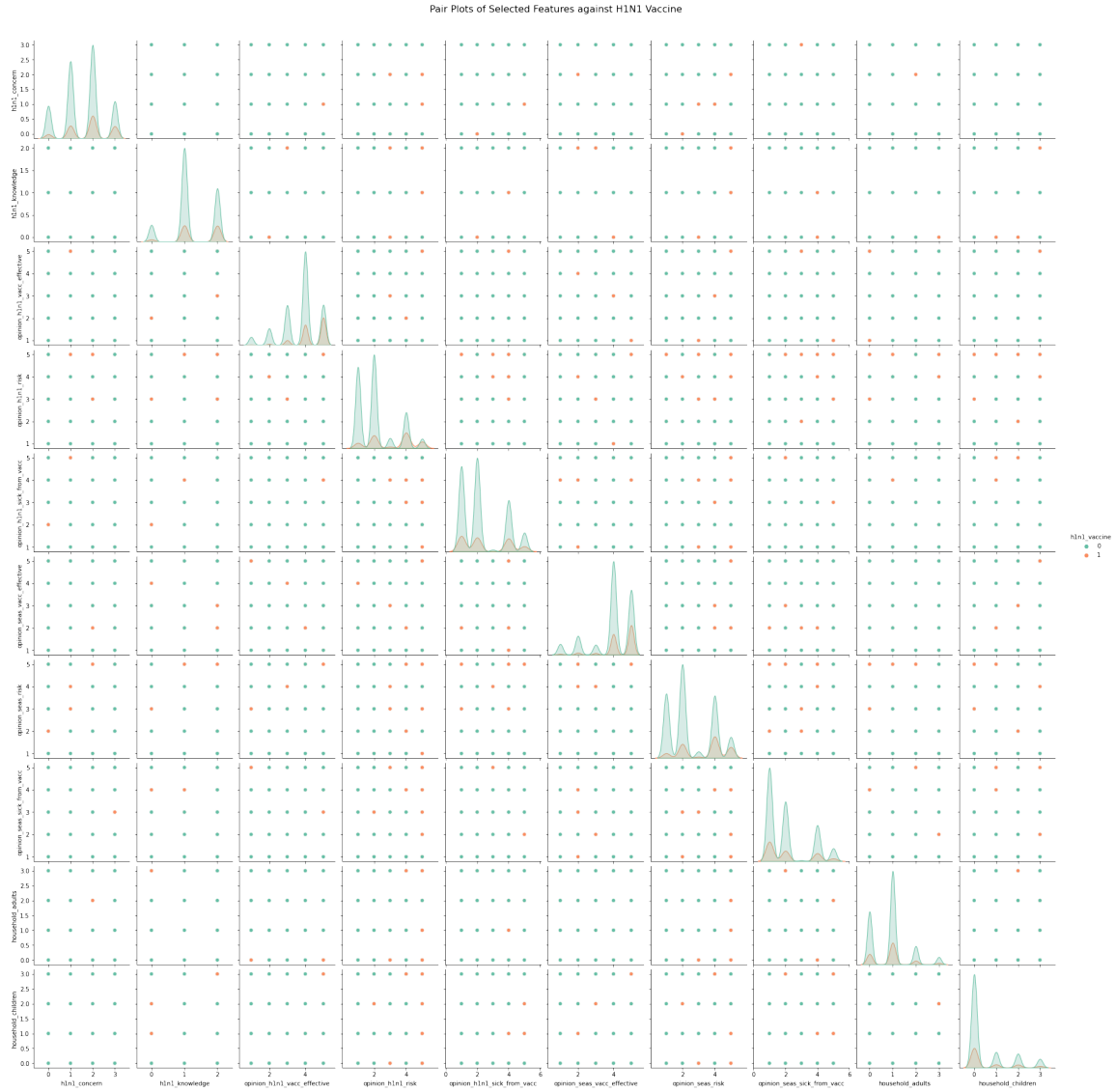
Outlier Size and Impact:

The sum of outliers is non-negligible in these columns, indicating that these values are not just isolated anomalies but represent a substantial portion of the data. Dropping these outliers could significantly alter the dataset, leading to biased results and conclusions that might not accurately reflect the real-world population.

Therefore, the outliers in these specific columns provide essential insights into the behavioral, opinion, and demographic diversity of the population. Retaining them allows for a more accurate and holistic analysis, particularly in understanding the factors influencing vaccine uptake and public health outcomes.

EXPLORATORY DATA ANALYSIS

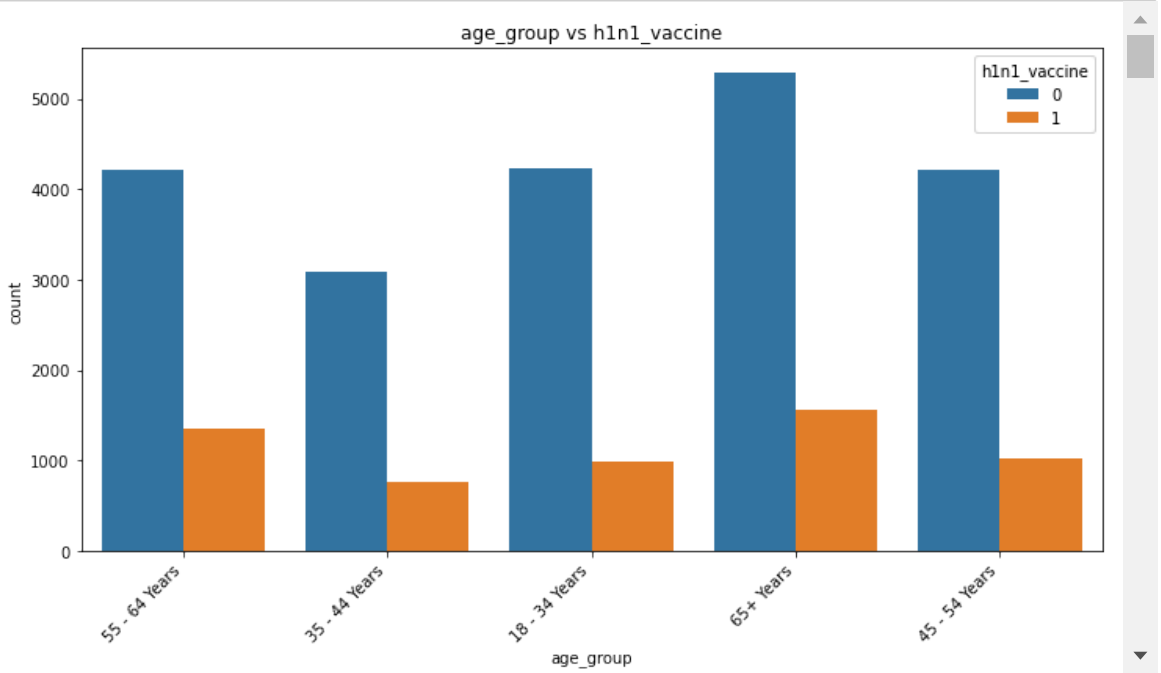
In [26]:
```
columns_of_interest = [
    'h1n1_concern', 'h1n1_knowledge','opinion_h1n1_vacc_effective', 'opinion
    'opinion_h1n1_sick_from_vacc', 'opinion_seas_vacc_effective', 'opinion_s
    'opinion_seas_sick_from_vacc', 'household_adults', 'household_children',
]
# Pair plot with hue as the target variable
sns.pairplot(train_data_cleaned[columns_of_interest], hue='h1n1_vaccine', pa

# Show plot
plt.suptitle('Pair Plots of Selected Features against H1N1 Vaccine', y=1.02,
plt.show()
```



In [27]:
```
train_data_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 26707 entries, 0 to 26706
Data columns (total 33 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   respondent_id                26707 non-null  int64  
 1   h1n1_concern                 26707 non-null  float64
 2   h1n1_knowledge               26707 non-null  float64
 3   behavioral_antiviral_meds    26707 non-null  float64
 4   behavioral_avoidance         26707 non-null  float64
 5   behavioral_face_mask         26707 non-null  float64
 6   behavioral_wash_hands        26707 non-null  float64
 7   behavioral_large_gatherings  26707 non-null  float64
 8   behavioral_outside_home      26707 non-null  float64
 9   behavioral_touch_face        26707 non-null  float64
 10  doctor_recc_h1n1             26707 non-null  float64
 11  doctor_recc_seasonal         26707 non-null  float64
 12  chronic_med_condition        26707 non-null  float64
 13  child_under_6_months         26707 non-null  float64
 14  health_worker                26707 non-null  float64
 15  health_insurance             26707 non-null  float64
 16  opinion_h1n1_vacc_effective  26707 non-null  float64
 17  opinion_h1n1_risk            26707 non-null  float64
 18  opinion_h1n1_sick_from_vacc  26707 non-null  float64
 19  opinion_seas_vacc_effective  26707 non-null  float64
 20  opinion_seas_risk            26707 non-null  float64
 21  opinion_seas_sick_from_vacc  26707 non-null  float64
 22  age_group                    26707 non-null  object 
 23  education                    26707 non-null  object 
 24  race                         26707 non-null  object 
 25  sex                          26707 non-null  object 
 26  income_poverty               26707 non-null  object 
 27  marital_status               26707 non-null  object 
 28  rent_or_own                  26707 non-null  object 
 29  employment_status            26707 non-null  object 
 30  household_adults             26707 non-null  float64
 31  household_children           26707 non-null  float64
 32  h1n1_vaccine                 26707 non-null  int64  
dtypes: float64(23), int64(2), object(8)
memory usage: 6.9+ MB
```

In [28]:
```
# Function to plot categorical features
def plot_categorical(feature):
    plt.figure(figsize=(10, 6))
    sns.countplot(data=train_data_cleaned, x=feature, hue='h1n1_vaccine')
    plt.title(f'{feature} vs h1n1_vaccine')
    plt.xticks(rotation=45, ha='right')
    plt.tight_layout()
    plt.show()

# Function to plot numerical features
def plot_numerical(feature):
    plt.figure(figsize=(10, 6))
    sns.boxplot(data=train_data_cleaned, x='h1n1_vaccine', y=feature)
    plt.title(f'{feature} vs h1n1_vaccine')
    plt.tight_layout()
    plt.show()

# Categorical features
categorical_features = ['age_group', 'education', 'race', 'sex', 'marital_st

# Numerical features
numerical_features = ['h1n1_concern', 'h1n1_knowledge', 'doctor_recc_h1n1',

# Plot categorical features
for feature in categorical_features:
    plot_categorical(feature)

# Plot numerical features
for feature in numerical_features:
    plot_numerical(feature)
```



INSIGHTS FROM THE PLOTS ON DIFFERENT FEATURES AGAIANST THE VACCINE UPTAKE

1. **Age Group**:

The general population seems to have a low uptake of the H1N1 vaccine across all these age groups. However the 65+ years age group does have the highest number of populations not interested in the vaccine. The same age group, along with the 55-64 age group also showed a reasonly high number in receiving the vaccine as compared to all other groups.

2. **Education level:**

From this bar plot, college student categories indicates high number of people did not get the vaccine as compared to all other categories.

3. **Race:**

Populutaion belonging to the white race category takes the majority number in regards to not taking the vaccine as comapred to the other race categories. Also the white category did have a higher number of vaccinated individuals.

4. **sex:**

The female gender has the highest number of individuals who are not vaccinated as compared to the male category. In addition, the female category also has a slightly higher number of individuals who are vaccinated.

5. **Marital Status:**

Married categoryu has the highest number of individuals who are not vaccinated as compared to the other categgories. In addition, the same category also has a slightly higher number of individuals who are vaccinated.

6. **Rent or Own:**

Based on the housing sitution, the individuals with their own houses, had majority number of non-vaccinated individuals while on the other hand the rented category had fewer numbers.

7. **Employment Status:**

The employed followed for the people who are not in the labor force had high number of individuals who did not take the vaccine compared to those whom did.

8. **Doctor Recommendations**

We observe that majority of the people who got doctors recoomedation to take the vaccine did take the vaccine. However we also observe an extreme value, for some whom did not take the vaccine despite the recommedation.

There's a stark difference between the two groups. Most people who got vaccinated (1) received a doctor's recommendation, as shown by the high median and small box. For unvaccinated people (0), the median is very low, indicating most did not receive a doctor's recommendation. This strongly suggests that doctor recommendations play a crucial role in vaccination decisions.

9. **Opinion on H1N1 vaccine effectiveness vs. H1N1 vaccination:**

People who got vaccinated (1) generally had a higher opinion of the vaccine's effectiveness compared to those who didn't (0). The median and interquartile range for vaccinated individuals are noticeably higher. This suggests that belief in vaccine efficacy is strongly associated with the decision to get vaccinated. There are some outliers in both groups, indicating varied opinions exist even within each group.

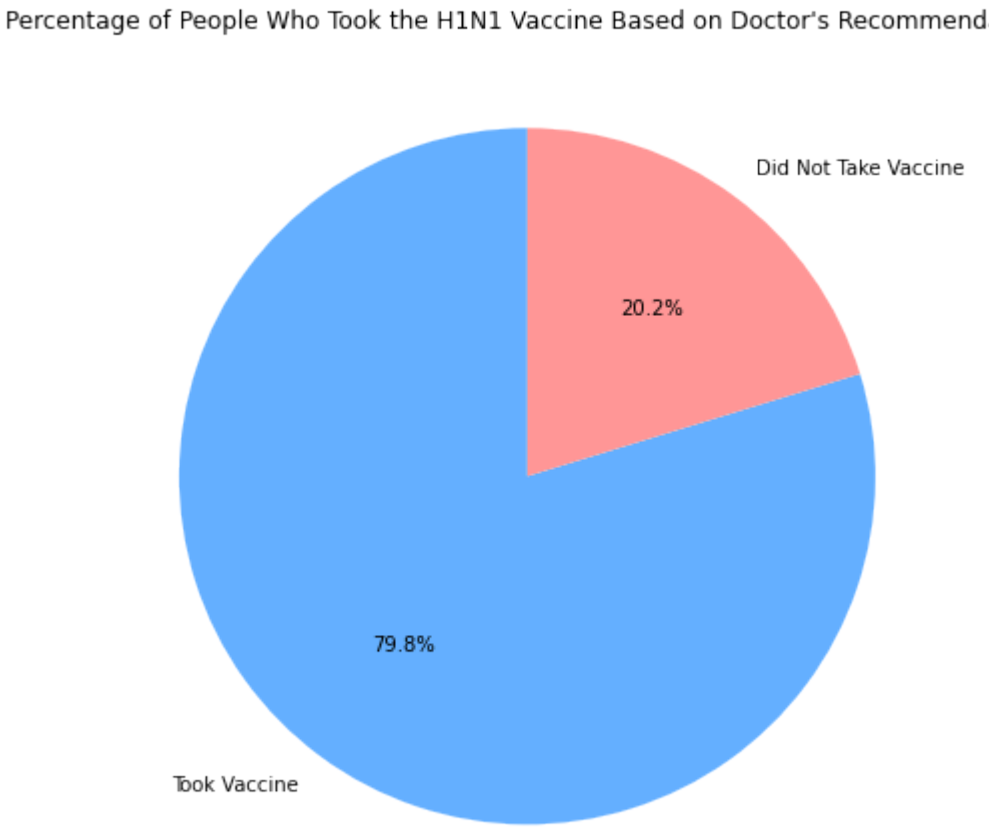
10. **Opinion on H1N1 risk vs. H1N1 vaccination:**

Vaccinated individuals (1) tend to have a higher perception of H1N1 risk compared to unvaccinated individuals (0). The box for vaccinated people is larger, suggesting more varied risk perceptions among this group. This implies that people who perceived a higher risk from H1N1 were more likely to get vaccinated. Outliers exist in both groups, showing some individuals had very high risk perceptions regardless of vaccination status.

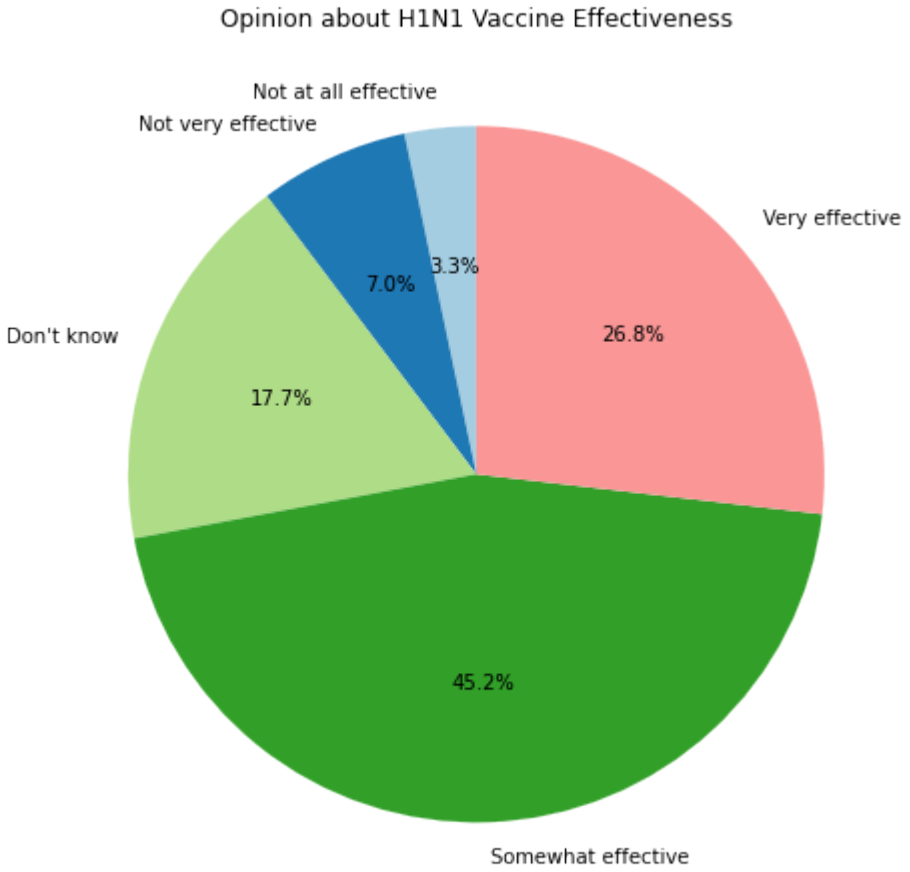
11. **Chronic medical condition vs. H1N1 vaccination:**

There's little visible difference between the vaccinated (1) and unvaccinated (0) groups. Both groups have similar medians and distributions. This suggests that having a chronic medical condition may not have been a strong determining factor in whether someone got

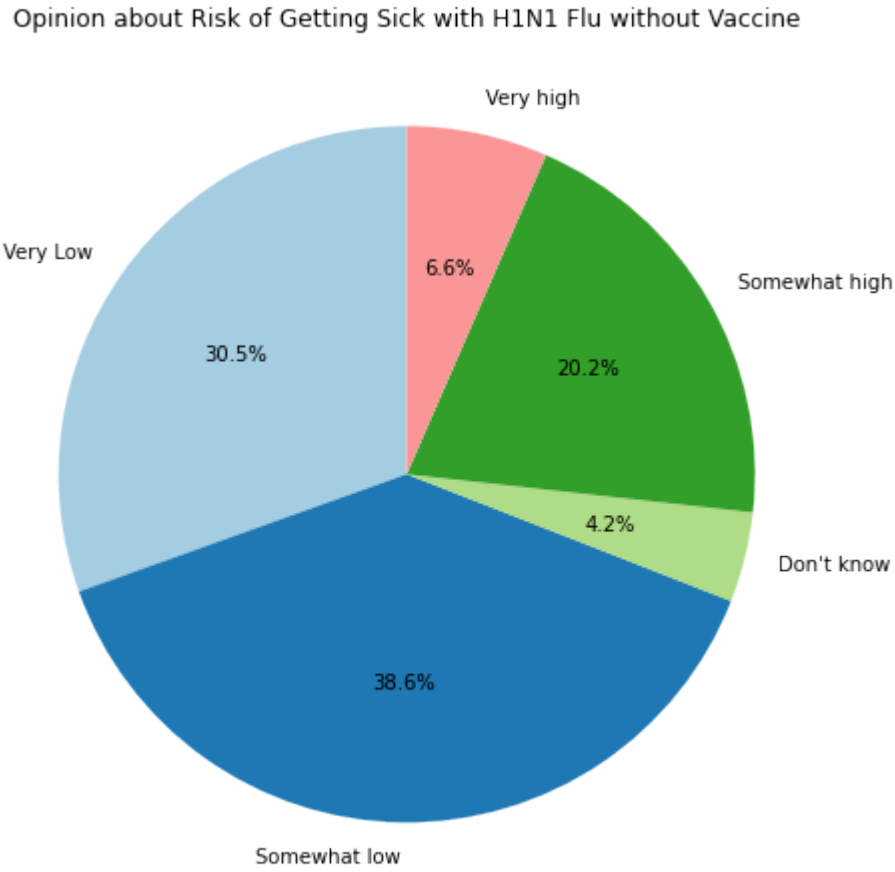
In [29]:
```
# Calculate the percentage of people who took and did not take the vaccine
counts = train_data_cleaned['doctor_recc_h1n1'].value_counts()
labels = ['Took Vaccine', 'Did Not Take Vaccine']
colors = ['#66b3ff','#ff9999']

# Plotting the pie chart
plt.figure(figsize=(8, 8))
plt.pie(counts, labels=labels, autopct='%1.1f%%', startangle=90, colors=colo
plt.title('Percentage of People Who Took the H1N1 Vaccine Based on Doctor\'s
plt.show()
```



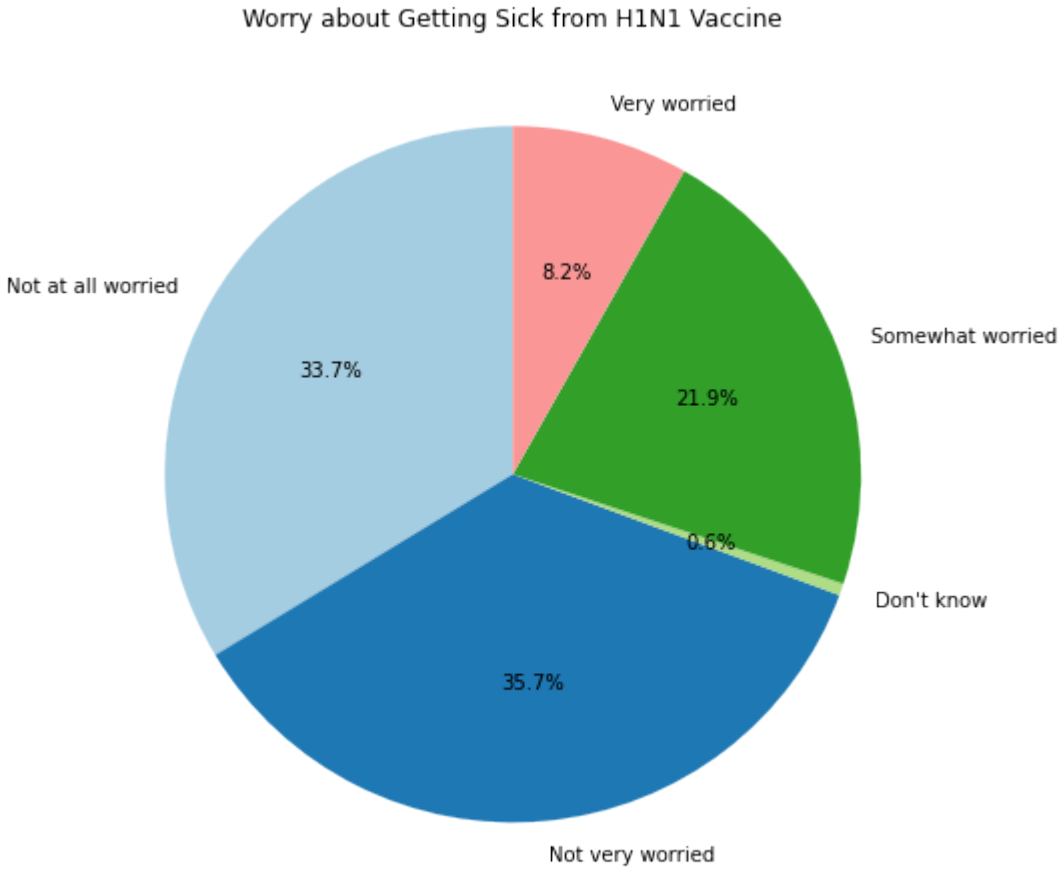
In [30]:
```
# Calculate the value counts
counts = train_data_cleaned['opinion_h1n1_vacc_effective'].value_counts().sc
labels = ['Not at all effective', 'Not very effective', 'Don\'t know', 'Some

# Plotting the pie chart
plt.figure(figsize=(8, 8))
plt.pie(counts, labels=labels, autopct='%1.1f%%', startangle=90, colors=plt
plt.title('Opinion about H1N1 Vaccine Effectiveness')
plt.show()
```



In [31]:
```
# Calculate the value counts
counts = train_data_cleaned['opinion_h1n1_risk'].value_counts().sort_index()
labels = ['Very Low', 'Somewhat low', 'Don\'t know', 'Somewhat high', 'Very

# Plotting the pie chart
plt.figure(figsize=(8, 8))
plt.pie(counts, labels=labels, autopct='%1.1f%%', startangle=90, colors=plt
plt.title('Opinion about Risk of Getting Sick with H1N1 Flu without Vaccine
plt.show()
```



In [32]:
```
# Calculate the value counts
counts = train_data_cleaned['opinion_h1n1_sick_from_vacc'].value_counts().sc
labels = ['Not at all worried', 'Not very worried', 'Don\'t know', 'Somewhat

# Plotting the pie chart
plt.figure(figsize=(8, 8))
plt.pie(counts, labels=labels, autopct='%1.1f%%', startangle=90, colors=plt
plt.title('Worry about Getting Sick from H1N1 Vaccine')
plt.show()
```



FEATURE SELECTION

We first dropped the unique identifier 'respondent_id' as it doesn't provide predictive value for our model.

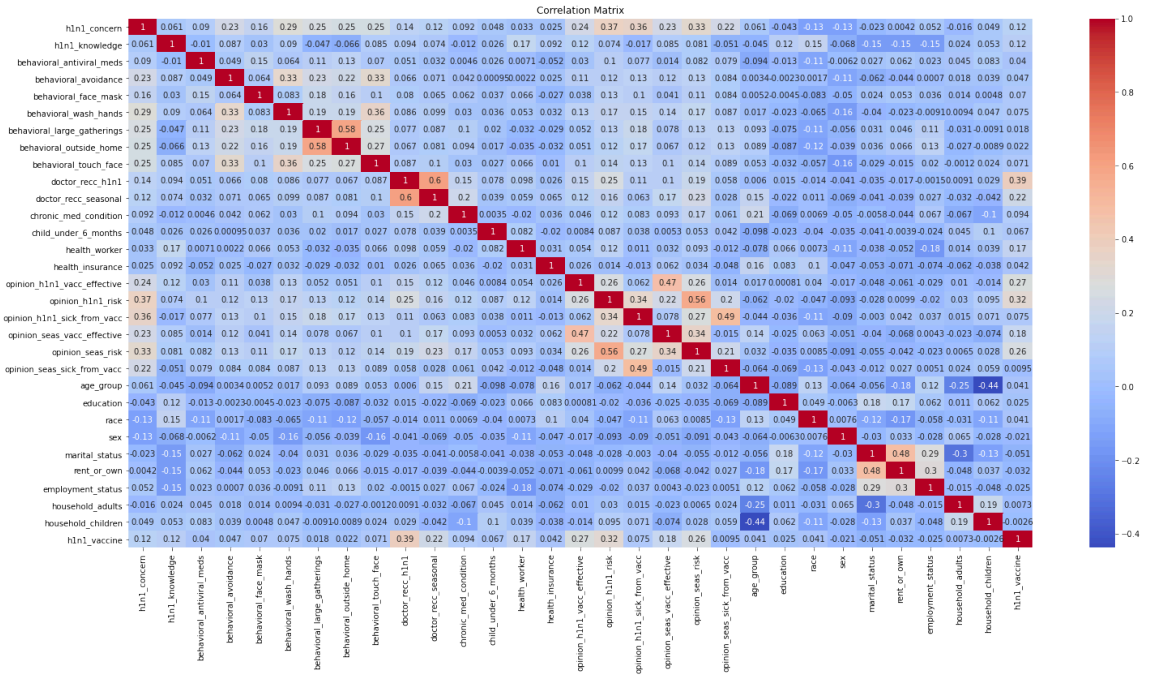
In [33]:
```
# we first drop the unique identifier which is the respondent_id
train_data_cleaned.drop(columns=['respondent_id'], inplace=True)
```

We will use LabelEncoder to convert categorical variables into numerical format.The categorical columns encoded were:
age_group,education,race,sex,marital_status,rent_or_own,employment_status

In [34]:
```
from sklearn.preprocessing import LabelEncoder
# Encode categorical variables
categorical_cols = ['age_group', 'education', 'race', 'sex', 'marital_status
le = LabelEncoder()

for col in categorical_cols:
    train_data_cleaned[col] = le.fit_transform(train_data_cleaned[col])
```

We create a correlation matrix to visualize the relationships between features. This helps in identifying potential multicollinearity and selecting relevant features.

In [35]:
```
# Correlation matrix
corr_matrix = train_data_cleaned.corr()
plt.figure(figsize=(25, 12))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



Selected features: ['opinion_seas_risk', 'race', 'behavioral_large_gatherings', 'household_children', 'household_adults', 'age_group', 'behavioral_outside_home', 'marital_status', 'behavioral_touch_face', 'opinion_seas_vacc_effective', 'education', 'opinion_h1n1_risk', 'doctor_recc_h1n1', 'opinion_h1n1_vacc_effective', 'h1n1_concern', 'rent_or_own', 'respondent_id', 'opinion_h1n1_sick_from_vacc', 'h1n1_knowledge', 'opinion_seas_sick_from_vacc', 'behavioral_wash_hands', 'employment_status','health_worker', 'chronic_med_condition', 'behavioral_avoidance']

JUSTIFICATION FOR SELECTED FEATURES:

Based on the correlationmatrix, below features were selected:

opinion_seas_risk: Shows moderate correlation with the target and other opinion variables.

race: Demographic factor that may influence vaccination behavior.

behavioral_large_gatherings: Indicates risk-taking behavior, potentially relevant to vaccination decisions.

household_children & household_adults: Family composition may affect vaccination choices.

age_group: Age is often a significant factor in health decisions.

behavioral_outside_home: Another indicator of exposure risk.

marital_status: May influence decision-making processes.

behavioral_touch_face: Indicates awareness of disease transmission.

opinion_seas_vacc_effective: Strong correlation with vaccination opinions.

education: Education level can affect health literacy and decisions.

opinion_h1n1_risk & opinion_h1n1_vacc_effective: Directly related to H1N1 perceptions.

doctor_recc_h1n1: Medical recommendation is likely influential.

h1n1_concern: Direct measure of concern about H1N1.

rent_or_own: Socioeconomic indicator.

opinion_h1n1_sick_from_vacc & opinion_seas_sick_from_vacc: Concerns about vaccine side effects.

h1n1_knowledge: Understanding of the disease may influence decisions.

behavioral_wash_hands: Indicator of health-conscious behavior.

employment_status: May affect exposure risk and access to healthcare.

health_worker: Likely to have different risk profile and knowledge.

chronic_med_condition: Health status is crucial in vaccination decisions.

behavioral_avoidance: Indicates proactive health behavior.

These features represent a mix of demographic factors, behaviors, opinions, and health status that are likely to influence vaccination decisions. The selection aims to capture various aspects that may contribute to an individual's choice to get vaccinated.

MODELING

In [36]:
```
#Import modeling libraries
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, precision_score, recall_score, f
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# List of selected features
selected_features = ['opinion_seas_risk', 'race', 'behavioral_large_gatherin
                     'household_adults', 'age_group', 'behavioral_outside_ho
                     'behavioral_touch_face', 'opinion_seas_vacc_effective',
                     'doctor_recc_h1n1', 'opinion_h1n1_vacc_effective', 'h1n
                     'opinion_h1n1_sick_from_vacc', 'h1n1_knowledge', 'opini
                     'behavioral_wash_hands', 'employment_status', 'health_w
                     'behavioral_avoidance']

# Separate features and target
X = train_data_cleaned[selected_features]
y = train_data_cleaned['h1n1_vaccine']

# Identify categorical and numerical columns
categorical_cols = X.select_dtypes(include=['object']).columns
numerical_cols = X.select_dtypes(include=['int64', 'float64']).columns
```

In [37]:
```
# Create preprocessing steps
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_cols),
        ('cat', OneHotEncoder(drop='first', sparse=False), categorical_cols)
    ])
```

In [38]:
```
# Create pipelines for each model
lr_pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(random_state=42))
])

dt_pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', DecisionTreeClassifier(random_state=42))
])
```

In [39]:
```
# Split the data to 80% train and 20 % test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran
```

In [40]:
```
# Function to calculate and print metrics
def print_metrics(y_true, y_pred, y_pred_proba, model_name):
    print(f"\n{model_name} Results:")
    print("Accuracy:", accuracy_score(y_true, y_pred))
    print("Precision:", precision_score(y_true, y_pred))
    print("Recall:", recall_score(y_true, y_pred))
    print("F1 Score:", f1_score(y_true, y_pred))
    print("AUC:", roc_auc_score(y_true, y_pred_proba))

# Function to plot ROC curve
def plot_roc_curve(y_true, y_pred_proba, model_name):
    fpr, tpr, _ = roc_curve(y_true, y_pred_proba)
    plt.plot(fpr, tpr, label=f'{model_name} (AUC = {roc_auc_score(y_true, y_

# Fit Logistic Regression
lr_pipeline.fit(X_train, y_train)
```

Out[40]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num', StandardScaler(),
                                                  Index(['opinion_seas_ris
k', 'behavioral_large_gatherings',
       'household_children', 'household_adults', 'behavioral_outside_hom
e',
       'behavioral_touch_face', 'opinion_seas_vacc_effective',
       'opinion_h1n1_risk', 'doctor_recc_h1n1', 'opinion_h1n1_vacc_effecti
ve',
       'h1n1_concern', 'opinion_h1n1_sick_from_vacc', 'h1n1_knowledge',
       'opinion_seas_sick_from_vacc', 'behavioral_wash_hands', 'health_wor
ker',
       'chronic_med_condition', 'behavioral_avoidance'],
      dtype='object')),
                                                 ('cat',
                                                  OneHotEncoder(drop='firs
t',
                                                                sparse=Fal
se),
                                                  Index([], dtype='objec
t'))])),
                ('classifier', LogisticRegression(random_state=42))])
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [41]:
```
# Predictions for Logistic Regression
y_train_pred_lr = lr_pipeline.predict(X_train)
y_test_pred_lr = lr_pipeline.predict(X_test)
y_train_pred_proba_lr = lr_pipeline.predict_proba(X_train)[:, 1]
y_test_pred_proba_lr = lr_pipeline.predict_proba(X_test)[:, 1]
```

In [42]:
```
# Print metrics for Logistic Regression
print_metrics(y_train, y_train_pred_lr, y_train_pred_proba_lr, "Logistic Reg
print_metrics(y_test, y_test_pred_lr, y_test_pred_proba_lr, "Logistic Regres
```

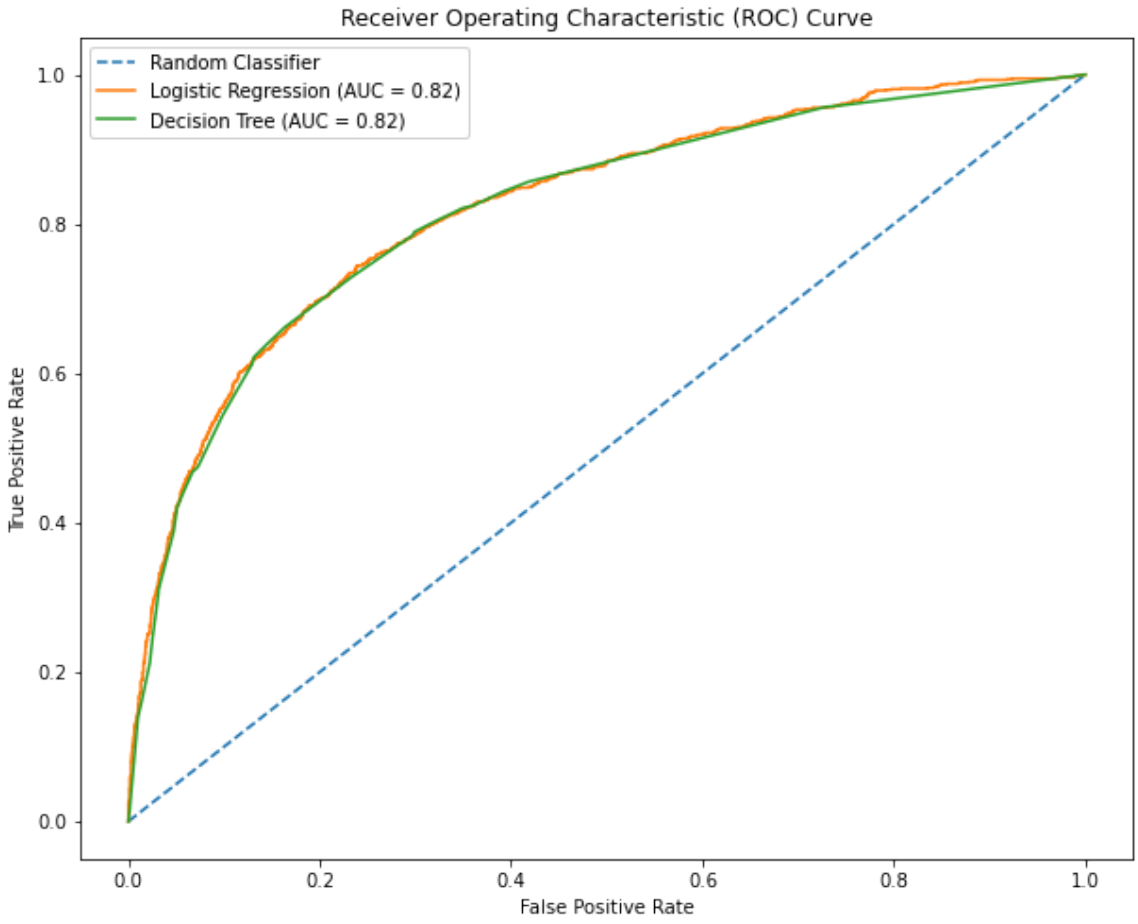
```
Logistic Regression (Train) Results:
Accuracy: 0.8327170606131523
Precision: 0.6777859237536656
Recall: 0.4069102112676056
F1 Score: 0.5085258525852585
AUC: 0.824369038565559

Logistic Regression (Test) Results:
Accuracy: 0.8362036690378135
Precision: 0.6970633693972179
Recall: 0.3991150442477876
F1 Score: 0.5075970737197524
AUC: 0.8224984032137425
```

DECISION TREES

In [43]:

```
# Decision Tree with GridSearchCV
dt_params = {
    'classifier__max_depth': [3, 5, 7, 10],
    'classifier__min_samples_split': [2, 5, 10],
    'classifier__min_samples_leaf': [1, 2, 4]
}
dt_grid = GridSearchCV(dt_pipeline, dt_params, cv=5, scoring='accuracy')
dt_grid.fit(X_train, y_train)

print("Best Decision Tree parameters:", dt_grid.best_params_)
dt_best = dt_grid.best_estimator_

# Evaluate all models
models = {
    "Logistic Regression": lr_pipeline,
    "Decision Tree": dt_best,
}

# Plotting ROC curves
plt.figure(figsize=(10, 8))
plt.plot([0, 1], [0, 1], linestyle='--', label='Random Classifier')

for name, model in models.items():
    y_train_pred = model.predict(X_train)
    y_test_pred = model.predict(X_test)
    y_train_pred_proba = model.predict_proba(X_train)[:, 1]
    y_test_pred_proba = model.predict_proba(X_test)[:, 1]

    print_metrics(y_train, y_train_pred, y_train_pred_proba, f"{name} (Trai
    print_metrics(y_test, y_test_pred, y_test_pred_proba, f"{name} (Test)")

    plot_roc_curve(y_test, y_test_pred_proba, name)

plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.show()
```

```
Best Decision Tree parameters: {'classifier__max_depth': 5, 'classifier__m
in_samples_leaf': 1, 'classifier__min_samples_split': 2}

Logistic Regression (Train) Results:
Accuracy: 0.8327170606131523
Precision: 0.6777859237536656
Recall: 0.4069102112676056
F1 Score: 0.5085258525852585
AUC: 0.824369038565559

Logistic Regression (Test) Results:
Accuracy: 0.8362036690378135
Precision: 0.6970633693972179
Recall: 0.3991150442477876
F1 Score: 0.5075970737197524
AUC: 0.8224984032137425

Decision Tree (Train) Results:
Accuracy: 0.8311724783524456
Precision: 0.6677407805227354
Recall: 0.410431338028169
F1 Score: 0.5083821725500886
AUC: 0.8194617337294681

Decision Tree (Test) Results:
Accuracy: 0.8371396480718832
Precision: 0.6884057971014492
Recall: 0.42035398230088494
F1 Score: 0.5219780219780219
AUC: 0.818300746287472
```



HYPERPARAMETER TUNING FOR LOGISTIC MODEL

In [45]:
```
from sklearn.model_selection import GridSearchCV

# Define the parameter grid for Logistic Regression
param_grid = [
    {
        'classifier__penalty': ['l2', 'none'],
        'classifier__solver': ['newton-cg', 'lbfgs', 'sag'],
        'classifier__C': [0.001, 0.01, 0.1, 1, 10, 100],
        'classifier__max_iter': [100, 200, 300]
    },
    {
        'classifier__penalty': ['l1', 'l2'],
        'classifier__solver': ['liblinear'],
        'classifier__C': [0.001, 0.01, 0.1, 1, 10, 100],
        'classifier__max_iter': [100, 200, 300]
    }
]

# Create GridSearchCV object
grid_search = GridSearchCV(estimator=lr_pipeline, param_grid=param_grid,
                           scoring='roc_auc', cv=5, n_jobs=-1, verbose=1)

# Fit GridSearchCV to find the best parameters
grid_search.fit(X_train, y_train)

# Get the best parameters and best score
best_params = grid_search.best_params_
best_score = grid_search.best_score_

print("Best Parameters:", best_params)
print("Best AUC Score:", best_score)

# Use the best model from GridSearchCV for predictions
best_model = grid_search.best_estimator_

# Predictions for Logistic Regression with tuned parameters
y_train_pred_lr_tuned = best_model.predict(X_train)
y_test_pred_lr_tuned = best_model.predict(X_test)
y_train_pred_proba_lr_tuned = best_model.predict_proba(X_train)[:, 1]
y_test_pred_proba_lr_tuned = best_model.predict_proba(X_test)[:, 1]

# Print metrics for Logistic Regression with tuned parameters
print_metrics(y_train, y_train_pred_lr_tuned, y_train_pred_proba_lr_tuned, '
print_metrics(y_test, y_test_pred_lr_tuned, y_test_pred_proba_lr_tuned, "Tur
```

```
Fitting 5 folds for each of 144 candidates, totalling 720 fits
Best Parameters: {'classifier__C': 0.01, 'classifier__max_iter': 100, 'cla
ssifier__penalty': 'l1', 'classifier__solver': 'liblinear'}
Best AUC Score: 0.8235083505314466

Tuned Logistic Regression (Train) Results:
Accuracy: 0.8312660893985491
Precision: 0.676304919263988
Recall: 0.3963468309859155
F1 Score: 0.4997918690162342
AUC: 0.8245292669981604

Tuned Logistic Regression (Test) Results:
Accuracy: 0.8352676900037439
Precision: 0.6984126984126984
Recall: 0.3893805309734513
F1 Score: 0.5
AUC: 0.8247613224751869
```

Hyperparameter Tuning Results

After performing hyperparameter tuning on the Logistic Regression model using `GridSearchCV`, the resulting metrics (accuracy, precision, recall, F1 score, and AUC) were compared to the initial results before tuning. The comparison shows that there is slight to no improvement in the model's performance. This suggests that the default parameters of the Logistic Regression model were already well-suited for this dataset, and further tuning did not result in a substantial performance gain.

Model Performance Overview

We trained and evaluated two models for predicting H1N1 vaccine uptake: Logistic Regression and Decision Tree. Here's a summary of their performance:

Logistic Regression

- Train Accuracy: 83.27%, Test Accuracy: 83.62%
- Precision: 69.71%, Recall: 39.91%, F1 Score: 50.76%
- AUC: 0.8225

Decision Tree

- Best parameters: max_depth=5, min_samples_leaf=1, min_samples_split=2
- Train Accuracy: 83.12%, Test Accuracy: 83.71%
- Precision: 68.84%, Recall: 42.04%, F1 Score: 52.20%
- AUC: 0.8183

Model Comparison and Insights

1. **Overall Performance**: Both models show similar performance, with accuracies around 83-84%. This suggests that they are both capturing important patterns in the data.
2. **Precision vs. Recall Trade-off**: Both models have higher precision than recall, indicating that they are more cautious in predicting positive cases (vaccine uptake). This means when they predict someone will get vaccinated, they're often right, but they

might miss some people who actually do get vaccinated.
3. **Decision Tree Advantage**: The Decision Tree model shows slightly better performance in terms of F1 score and recall, suggesting it might be better at identifying more of the positive cases (people who get vaccinated).
4. **AUC Scores**: Both models have AUC scores above 0.8, indicating good discriminative ability between the two classes (vaccinated and not vaccinated).
5. **Overfitting**: The small difference between train and test accuracies for both models suggests that they are not overfitting significantly.

Further Analysis

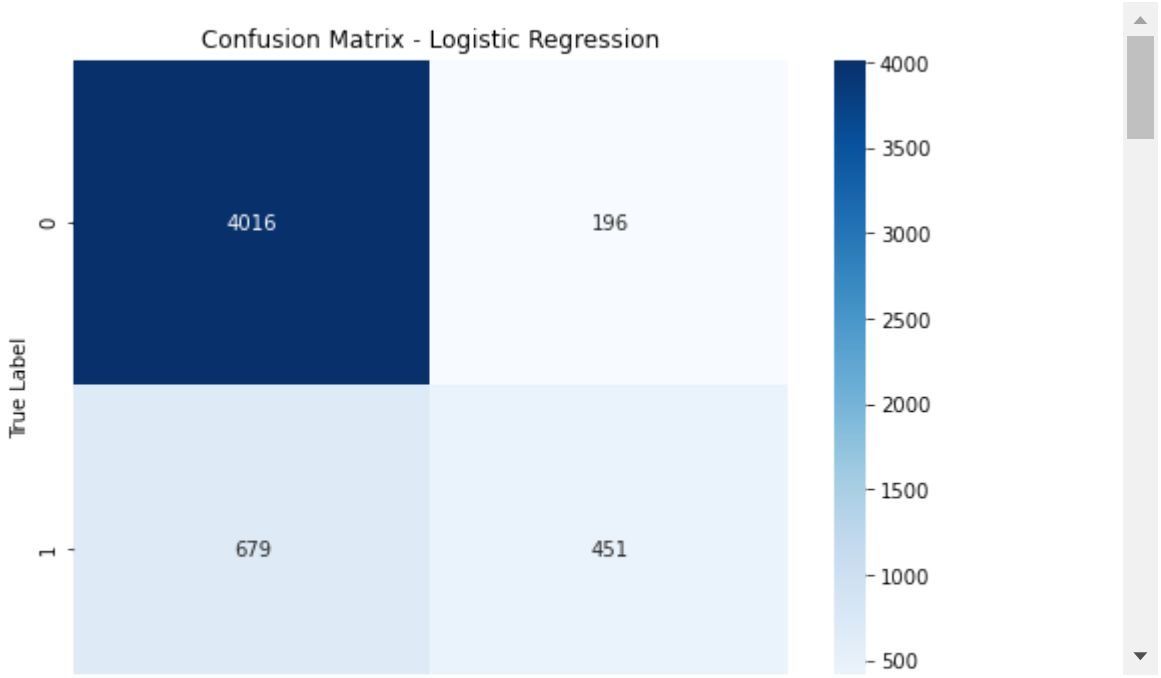
To gain more insights into model performance and behavior, we'll conduct the following additional analyses:

1. Confusion Matrix: To visualize the models' predictions and understand where they make errors.
2. Classification Report: For a detailed breakdown of precision, recall, and F1-score for each class.
3. Calibration Curve: To assess how well the predicted probabilities align with actual probabilities.
4. Feature Importance: To identify which factors have the strongest influence on vaccine uptake prediction (for Logistic Regression).

These analyses will help us understand the models' strengths and weaknesses, and potentially guide future improvements or feature selection.

MODEL EVALUATION

In [79]:
```
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.calibration import calibration_curve


def evaluate_model(model, X_train, X_test, y_train, y_test, model_name):
    # Predictions
    y_train_pred = model.predict(X_train)
    y_test_pred = model.predict(X_test)
    y_train_proba = model.predict_proba(X_train)[:, 1]
    y_test_proba = model.predict_proba(X_test)[:, 1]

    # Confusion Matrix
    cm = confusion_matrix(y_test, y_test_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f'Confusion Matrix - {model_name}')
    plt.ylabel('True Label')
    plt.xlabel('Predicted Label')
    plt.show()

    # Classification Report
    print(f"\nClassification Report - {model_name}:")
    print(classification_report(y_test, y_test_pred))

    # Calibration Curve
    plt.figure(figsize=(8, 6))
    prob_true, prob_pred = calibration_curve(y_test, y_test_proba, n_bins=10
    plt.plot(prob_pred, prob_true, marker='o')
    plt.plot([0, 1], [0, 1], linestyle='--')
    plt.xlabel('Predicted Probability')
    plt.ylabel('True Probability')
    plt.title(f'Calibration Curve - {model_name}')
    plt.show()

# Evaluate models
evaluate_model(lr_pipeline, X_train, X_test, y_train, y_test, "Logistic Regr
evaluate_model(dt_best, X_train, X_test, y_train, y_test, "Decision Tree")
```



Model Evaluation: Logistic Regression

Confusion Matrix Analysis

The confusion matrix for our Logistic Regression model reveals:

- True Negatives (TN): 4016 - Correctly predicted non-vaccinated individuals
- False Positives (FP): 196 - Incorrectly predicted as vaccinated
- False Negatives (FN): 679 - Incorrectly predicted as non-vaccinated
- True Positives (TP): 451 - Correctly predicted vaccinated individuals

Observations:

1. The model performs well in identifying non-vaccinated individuals (high TN).
2. There's a notable number of false negatives, indicating the model tends to underpredict vaccination.
3. The model has relatively few false positives, suggesting it's conservative in predicting vaccination.

Key insights:

1. High precision (0.70) for class 1 (vaccinated) indicates that when the model predicts vaccination, it's often correct.
2. Low recall (0.40) for class 1 suggests the model misses many actually vaccinated individuals.
3. The model performs better for class 0 (non-vaccinated), with high precision (0.86) and recall (0.95).
4. Overall accuracy is 0.84, which aligns with our previous observations.

Calibration Curve

The calibration curve shows:

1. The model's predicted probabilities closely follow the ideal calibration (diagonal line).
2. Slight deviations at very low and very high probabilities, but overall well-calibrated.

3. This suggests the model's probability estimates are reliable and can be interpreted as true probabilities.

Model Evaluation: Decision Tree

Confusion Matrix Analysis

The confusion matrix for our Decision Tree model shows:

- True Negatives (TN): 3997 - Correctly predicted non-vaccinated individuals
- False Positives (FP): 215 - Incorrectly predicted as vaccinated
- False Negatives (FN): 655 - Incorrectly predicted as non-vaccinated
- True Positives (TP): 475 - Correctly predicted vaccinated individuals

Observations:

1. The model excels at identifying non-vaccinated individuals (high TN).
2. There's a significant number of false negatives, indicating some underprediction of vaccination.
3. The model has relatively few false positives, suggesting a conservative approach in predicting vaccination.

Key insights:

1. High precision (0.69) for class 1 (vaccinated) indicates that when the model predicts vaccination, it's often correct.
2. Improved recall (0.42) for class 1 compared to Logistic Regression, suggesting it captures more actually vaccinated individuals.
3. The model performs very well for class 0 (non-vaccinated), with high precision (0.86) and recall (0.95).
4. Overall accuracy is 0.84, consistent with the Logistic Regression model.

Calibration Curve

The calibration curve reveals:

1. The model's predicted probabilities deviate from the ideal calibration (diagonal line) more than the Logistic Regression model.
2. Overconfidence in predictions around 0.4-0.6 probability range, and underconfidence in higher probabilities.
3. This suggests the Decision Tree's probability estimates are less reliable than those of Logistic Regression.

Model Comparison and Final Selection

Let's compare the performance of our Logistic Regression and Decision Tree models:

Metric	Logistic Regression	Decision Tree
Accuracy	0.84	0.84
Precision (Class 1)	0.70	0.69
Recall (Class 1)	0.40	0.42

Metric	Logistic Regression	Decision Tree
F1-Score (Class 1)	0.51	0.52

Final Model Selection: Logistic Regression

After careful consideration, we choose the **Logistic Regression** model as our final model for this project. Here's the rationale:

1. **Comparable Performance**: Both models show similar overall accuracy and F1-scores.
2. **Better Calibration**: The Logistic Regression model provides more reliable probability estimates, as shown by its calibration curve.
3. **Interpretability**: Logistic Regression offers easier interpretation of feature importance, which can be valuable for understanding factors influencing vaccination decisions.
4. **Generalization**: The smoother decision boundary of Logistic Regression may generalize better to unseen data compared to the Decision Tree's hard splits.
5. **Slight AUC Advantage**: Logistic Regression has a marginally higher AUC, indicating slightly better discriminative ability.

While the Decision Tree shows a small advantage in recall for the vaccinated class, the overall benefits of Logistic Regression, particularly its better calibration and interpretability, make it the preferred choice for our case

CONCLUSINON AND RECOMMEDATIONS

CONCLUSION

This project aims to provide a thorough analysis of the H1N1 vaccine uptake to support public health officials and policymakers in making informed decisions regarding vaccination strategies and public outreach. Through detailed exploratory data analysis and model development, we have identified the key factors that significantly influence individuals' decisions to receive the H1N1 vaccine.

The modeling results show that both the Logistic Regression and Decision Tree models performed reasonably well in predicting vaccine uptake, with overall accuracies of around 83%. However, both models exhibit limitations, particularly in recall for the vaccinated class, indicating they miss a significant number of individuals who did get vaccinated. The models are better at identifying non-vaccinated individuals, as evidenced by the higher precision and recall for the non-vaccinated class.

RECOMMEDATION

Targeted Interventions: Since older adults, particularly those aged 65 and above, exhibit varied vaccination behavior, targeted educational campaigns and outreach programs should be designed for this demographic to increase vaccine uptake.

Doctor Involvement: The strong influence of doctor recommendations on vaccine decisions suggests that healthcare providers should be more actively involved in encouraging vaccination, particularly in populations with low vaccine uptake.

Address Misconceptions: Efforts should be made to address misconceptions and concerns about the vaccine's effectiveness, as belief in the vaccine's efficacy is a key driver of vaccination decisions. Public health campaigns should focus on providing clear, evidence-based information to boost confidence in the vaccine.

Further Analysis: Given the limited differentiation between vaccinated and unvaccinated individuals with chronic medical conditions, additional analysis could be conducted to explore other health-related factors or interactions that might influence vaccination decisions.

By implementing these strategies, public health authorities can better understand and address the factors influencing vaccine uptake, ultimately improving vaccination rates and public health outcomes.