GROUP 4 MEMBERS:

- Perpetual Ann TL
- Joy Kamau
- Rahma Mohamed
- Immanuel Omondi
- Wilfred Njagi
- Ismail Ibrahim
- Robin Mutai

FLU SHOT LEARNING: PREDICT H1N1 AND SEASONAL FLU VACCINES

Alt+Q

Business Problem Understanding

How Opinion, Perception, and Behavior Affect H1N1 and Seasonal Flu Vaccination Rates

Introduction:

In this data science project, the objective is to understand the following factors affecting H1N1 & Seasonal flu vaccination rates.
i) Opinions & Perceptions ii) Demographics - Age, Education, Employment status & Income levels iii) Behavioral - Handwashing, masking and avoiding large crowds. iv) Doctors Recommendation

The project aims to analyze and uncover the factors that influence individuals' decision-making processes regarding getting vaccinated against the seasonal flu. By gaining insights into these factors,governments and healthcare organizations can develop targeted strategies and interventions to increase vaccination rates and improve public health outcomes.

Problem Statement

The world has recently experienced the impact of major flu outbreaks like the COVID-19, Swine Flu (H1N1) and the Avian Flu(H5N1). The effect of any flu outbreak depends on the type of flu and its respective variants, the population demographics like age, and other underlying health conditions of the individual. In the US, seasonal flu places a substantial burden on the health of people in the United States each year. CDC estimates that flu has resulted in 9 million – 41 million illnesses, 140,000 – 710,000 hospitalizations and 12,000 – 52,000 deaths annually between 2010 and 2020. The results obtained from CDC website also shows that the vaccination rates have remained low with an overall average of 57.8% as of 2022.

Despite the availability and effectiveness of flu vaccines, there are still significant portions of the population who choose not to get vaccinated. To address this problem, it is crucial to investigate the reasons behind these decisions and identify the key factors driving individuals' opinions, perceptions, and behaviors related to flu vaccination

Key Factors Affecting Seasonal Flu/H1N1 Vaccine Uptake

1. **Opinions**: Understanding the opinions held by individuals regarding flu vaccination is essential. This includes identifying factors that contribute to positive or negative opinions, such as beliefs about vaccine efficacy, concerns about side effects, distrust in the healthcare system, or misinformation from various sources.
2. **Perceptions**: Examining people's perceptions of the flu vaccine can provide valuable insights. This involves investigating how individuals perceive the severity of the flu, the likelihood of contracting it, and the perceived benefits and risks associated with vaccination. Perceptions can be influenced by media coverage, personal experiences, or the influence of friends, family, and healthcare professionals.
3. **Behaviors**: Analyzing vaccination behaviors is critical for understanding the gap between intention and action. Exploring the factors that influence individuals' decision-making processes, including barriers to access, convenience, cost, social norms, and previous vaccination experiences, can help uncover patterns and develop targeted interventions.

Approach

To address the business problem, we will explore the data from -https://www.drivendata.org/competitions/66/flu-shot-learning/ (https://www.drivendata.org/competitions/66/flu-shot-learning/)

Once the data has been cleaned and pre-processed, various data analysis techniques will be employed, such as exploratory data analysis, statistical modeling and machine learning to help identify patterns, correlations, and predictive factors related to flu vaccination rates and the underlying opinions, perceptions, and behaviors.

Potential Outputs:

The outputs of this project will provide actionable insights for businesses and healthcare organizations to increase seasonal flu vaccination rates. These outputs may include:

1. **Identification of key factors influencing vaccination decisions**: By analyzing the collected data, the project will help identify the most influential factors affecting individuals' opinions, perceptions, and behaviors related to flu vaccination. This information can guide the development of targeted campaigns and interventions.

2. **Segmentation of the target population**: The project project will also identify distinct segments within the target population based on their opinions, perceptions, and behaviors. This segmentation can help tailor messages and interventions to specific groups, considering their unique characteristics and needs.
3. **Prediction models**: By leveraging statistical modeling and machine learning techniques, the project will build predictive models

Data Understanding

In [1]:
```
!pip install tabulate
```

```
Requirement already satisfied: tabulate in /Users/wilfrednjagi/opt/anaconda3/lib/python3.9/site-packages (0.
8.10)
```

In [2]:
```
#importing libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from sklearn.model_selection import train_test_split, GridSearchCV

from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_curve, roc_auc_score
from sklearn.decomposition import PCA
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.multioutput import MultiOutputClassifier
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from sklearn.neighbors import KNeighborsClassifier

import pandas as pd
from sklearn.model_selection import cross_val_score, cross_val_predict, train_test_split
from sklearn import datasets
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score, recall_score, precision_score
```

Loading the data into Dataframes

In [3]:
```
import pandas as pd
```

In [4]:
```
#The first data frame has the different characteristics of the respondents
df1= pd.read_csv("training_set_features.csv", index_col="respondent_id", encoding="latin1")
df1.head()
```

Out[4]:

	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavioral_avoidance	behavioral_face_mask	behavioral_wash_
respondent_id						
0	1.0	0.0	0.0	0.0	0.0	
1	3.0	2.0	0.0	1.0	0.0	
2	1.0	1.0	0.0	1.0	0.0	
3	1.0	1.0	0.0	1.0	0.0	
4	2.0	1.0	0.0	1.0	0.0	

5 rows × 35 columns

In [5]:
```
#The second dataframe contains a binary classification of whether the respondents got the H1N1 or the Seasonal
df2= pd.read_csv('training_set_labels.csv',index_col="respondent_id", encoding="latin1")
df2.head()
```

Out[5]:

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

In [6]:
```
#We concatenate the two dataframes to have one that allows us to view all the columns of each respondents incl
df = pd.merge(df1, df2, on='respondent_id', how='left')
df.head()
```

Out[6]:

	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavioral_avoidance	behavioral_face_mask	behavioral_wash_
respondent_id						
0	1.0	0.0	0.0	0.0	0.0	
1	3.0	2.0	0.0	1.0	0.0	
2	1.0	1.0	0.0	1.0	0.0	
3	1.0	1.0	0.0	1.0	0.0	
4	2.0	1.0	0.0	1.0	0.0	

5 rows × 37 columns

Data Understanding

Our data has various columns... We created a CSV File that explains what each column means. The Data description csv will be loaded in the next file to help us have a deeper understanding of the data

This is the point where we seek to understand the different dimensions of the data.

In [7]:
```
data_desc_df= pd.read_csv('H1N1- Flu Data Desc.csv', encoding="latin1")
data_desc_df
```

Out[7]:

	Col_name	Description
0	seasonal_vaccine	Whether respondent received seasonal flu vacci...
1	h1n1_vaccine	Whether respondent received H1N1 flu vaccine. ...
2	respondent_id	Unique and random identifier.
3	h1n1_concern	Level of concern about the H1N1 flu.(0 = Not ...
4	h1n1_knowledge	Level of knowledge about H1N1 flu.(0 = No kno...
5	behavioral_antiviral_meds	Has taken antiviral medications. (binary)
6	behavioral_avoidance	Has avoided close contact with others with fl...
7	behavioral_face_mask	Has bought a face mask. (binary)
8	behavioral_wash_hands	Has frequently washed hands or used hand sani...
9	behavioral_large_gatherings	Has reduced time at large gatherings. (binary)
10	behavioral_outside_home	Has reduced contact with people outside of ow...
11	behavioral_touch_face	Has avoided touching eyes, nose, or mouth. (b...
12	doctor_recc_h1n1	H1N1 flu vaccine was recommended by doctor. (...
13	doctor_recc_seasonal	Seasonal flu vaccine was recommended by docto...
14	chronic_med_condition	Has any of the following chronic medical cond...
15	child_under_6_months	Has regular close contact with a child under ...
16	health_worker	Is a healthcare worker. (binary)
17	health_insurance	Has health insurance. (binary)
18	opinion_h1n1_vacc_effective	Respondent's opinion about H1N1 vaccine effec...
19	opinion_h1n1_risk	Respondent's opinion about risk of getting si...
20	opinion_h1n1_sick_from_vacc	Respondent's worry of getting sick from takin...
21	opinion_seas_vacc_effective	Respondent's opinion about seasonal flu vacci...
22	opinion_seas_risk	Respondent's opinion about risk of getting si...
23	opinion_seas_sick_from_vacc	Respondent's worry of getting sick from takin...
24	age_group	Age group of respondent.
25	education	Self-reported education level.
26	race	Race of respondent.

	Col_name	Description
27	sex	Sex of respondent.
28	income_poverty	Household annual income of respondent with re...
29	marital_status	Marital status of respondent.
30	rent_or_own	Housing situation of respondent.
31	employment_status	Employment status of respondent.
32	hhs_geo_region	Respondent's residence using a 10 region geog...
33	census_msa	Respondent's residence within metropolitan st...
34	household_adults	Number of other adults in household, top-code...
35	household_children	Number of children in household, top-coded to 3.
36	employment_industry	Type of industry respondent is employed in. V...
37	employment_occupation	Type of occupation of respondent. Values are ...

In [8]:
```
# Get the shape of the DataFrame
df.shape
```

Out[8]:
```
(26707, 37)
```

In [9]:
```
#dataframe's summary
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 26707 entries, 0 to 26706
Data columns (total 37 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   h1n1_concern                 26615 non-null  float64
 1   h1n1_knowledge               26591 non-null  float64
 2   behavioral_antiviral_meds    26636 non-null  float64
 3   behavioral_avoidance         26499 non-null  float64
 4   behavioral_face_mask         26688 non-null  float64
 5   behavioral_wash_hands        26665 non-null  float64
 6   behavioral_large_gatherings  26620 non-null  float64
 7   behavioral_outside_home      26625 non-null  float64
 8   behavioral_touch_face        26579 non-null  float64
 9   doctor_recc_h1n1             24547 non-null  float64
 10  doctor_recc_seasonal         24547 non-null  float64
 11  chronic_med_condition        25736 non-null  float64
 12  child_under_6_months         25887 non-null  float64
 13  health_worker                25903 non-null  float64
 14  health_insurance             14433 non-null  float64
 15  opinion_h1n1_vacc_effective  26316 non-null  float64
 16  opinion_h1n1_risk            26319 non-null  float64
 17  opinion_h1n1_sick_from_vacc  26312 non-null  float64
 18  opinion_seas_vacc_effective  26245 non-null  float64
 19  opinion_seas_risk            26193 non-null  float64
 20  opinion_seas_sick_from_vacc  26170 non-null  float64
 21  age_group                    26707 non-null  object 
 22  education                    25300 non-null  object 
 23  race                         26707 non-null  object 
 24  sex                          26707 non-null  object 
 25  income_poverty               22284 non-null  object 
 26  marital_status               25299 non-null  object 
 27  rent_or_own                  24665 non-null  object 
 28  employment_status            25244 non-null  object 
 29  hhs_geo_region               26707 non-null  object 
 30  census_msa                   26707 non-null  object 
 31  household_adults             26458 non-null  float64
 32  household_children           26458 non-null  float64
 33  employment_industry          13377 non-null  object 
 34  employment_occupation        13237 non-null  object 
 35  h1n1_vaccine                 26707 non-null  int64  
 36  seasonal_vaccine             26707 non-null  int64  
```

```
dtypes: float64(23), int64(2), object(12)
memory usage: 7.7+ MB
```

Data processing

Percentage of Missing Values

In [10]:
```
#missing_values_sum = df.isnull().sum()
#print(missing_values_sum)
missing_values_percentage = (df.isnull().sum() / len(df)) * 100
print(missing_values_percentage)
```

```
h1n1_concern                    0.344479
h1n1_knowledge                  0.434343
behavioral_antiviral_meds       0.265848
behavioral_avoidance            0.778822
behavioral_face_mask            0.071142
behavioral_wash_hands           0.157262
behavioral_large_gatherings     0.325757
behavioral_outside_home         0.307036
behavioral_touch_face           0.479275
doctor_recc_h1n1                8.087767
doctor_recc_seasonal            8.087767
chronic_med_condition           3.635751
child_under_6_months            3.070356
health_worker                   3.010447
health_insurance               45.957989
opinion_h1n1_vacc_effective     1.464036
opinion_h1n1_risk               1.452803
opinion_h1n1_sick_from_vacc     1.479013
opinion_seas_vacc_effective     1.729884
opinion_seas_risk               1.924589
opinion_seas_sick_from_vacc     2.010709
age_group                       0.000000
education                       5.268282
race                            0.000000
sex                             0.000000
income_poverty                 16.561201
marital_status                  5.272026
rent_or_own                     7.645936
employment_status               5.477965
hhs_geo_region                  0.000000
census_msa                      0.000000
household_adults                0.932340
household_children              0.932340
employment_industry            49.912008
employment_occupation          50.436215
h1n1_vaccine                    0.000000
seasonal_vaccine                0.000000
dtype: float64
```

Droping columns with the highest of values missing

In [11]:
```
df.drop(['health_insurance','employment_industry','employment_occupation'],axis=1,inplace = True)
```

In [12]:
```
df.describe()
```

Out[12]:

	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavioral_avoidance	behavioral_face_mask	behavioral_wash_hands
count	26615.000000	26591.000000	26636.000000	26499.000000	26688.000000	26665.000000
mean	1.618486	1.262532	0.048844	0.725612	0.068982	0.825614
std	0.910311	0.618149	0.215545	0.446214	0.253429	0.379448
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	1.000000	0.000000	0.000000	0.000000	1.000000
50%	2.000000	1.000000	0.000000	1.000000	0.000000	1.000000
75%	2.000000	2.000000	0.000000	1.000000	0.000000	1.000000
max	3.000000	2.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 24 columns

In [13]:
```
df.shape
```

Out[13]:
```
(26707, 34)
```

Renaming geographical locations into readable format

In [14]:
```
hhs_geo_region = {"hhs_geo_region":     {"lzgpxyit": "Town A", "fpwskwrf": "Town B","qufhixun": "Town C",
                                         "oxchjgsf":"Town D","kbazzjca":"Town E",
                                         "bhuqouqj":"Town F","mlyzmhmf":"Town G",
                                         "lrircsnp":"Town H","atmpeygn":"Town I","dqpwygqj":"Town J"}}

df = df.replace(hhs_geo_region)

# confirming that initial values are successfully replaced.
df["hhs_geo_region"]
```

Out[14]:
```
respondent_id
0        Town D
1        Town F
2        Town C
3        Town H
4        Town C
          ...
26702    Town C
26703    Town A
26704    Town A
26705    Town H
26706    Town G
Name: hhs_geo_region, Length: 26707, dtype: object
```

Renaming the Opinion Columns

In [15]:
```
# Respondent's opinion about H1N1 vaccine effectiveness.
df.opinion_h1n1_vacc_effective=df.opinion_h1n1_vacc_effective.replace({1 :"Not at all effective", 2 : "Not ver
                                                                      4 :"Somewhat effective", 5: "Very effective"})
#Respondent's opinion about risk of getting sick with H1N1 flu without vaccine.
df.opinion_h1n1_risk=df.opinion_h1n1_risk.replace({1 :"Very Low", 2 : "Somewhat low",3: "Don't know",
                                                                        4 : "Somewhat high", 5 :"Very high"})
#Respondent's opinion about seasonal flu vaccine effectiveness.
df.opinion_seas_vacc_effective=df.opinion_seas_vacc_effective.replace({1 :"Not at all effective", 2 : "Not ver
                                                                      4 :"Somewhat effective", 5: "Very effective"})
#Respondent's opinion about seasonal flu vaccine effectiveness.
df.opinion_h1n1_sick_from_vacc= df.opinion_h1n1_sick_from_vacc.replace({1 : "Not at all worried",2: "Not very
                                                                                 4 : "Somewhat worried" ,5 :"V
#Respondent's opinion about risk of getting sick with seasonal flu without vaccine.
df.opinion_seas_risk=df.opinion_seas_risk.replace({1 :"Very Low", 2 : "Somewhat low",3: "Don't know",
                                                                        4 : "Somewhat high", 5 :"Very high"})
#Respondent's worry of getting sick from taking seasonal flu vaccine
df.opinion_seas_sick_from_vacc= df.opinion_seas_sick_from_vacc.replace({1 : "Not at all worried",2: "Not very
                                                                                 4 : "Somewhat worried" ,5 :"V
```

In [16]:
```
#Checking for Duplicates
duplicates = df.duplicated()
#filtered rows of the duplicates
duplicated_rows= df[duplicates]
print(duplicated_rows)
```

```
Empty DataFrame
Columns: [h1n1_concern, h1n1_knowledge, behavioral_antiviral_meds, behavioral_avoidance, behavioral_face_mas
k, behavioral_wash_hands, behavioral_large_gatherings, behavioral_outside_home, behavioral_touch_face, docto
r_recc_h1n1, doctor_recc_seasonal, chronic_med_condition, child_under_6_months, health_worker, opinion_h1n1_
vacc_effective, opinion_h1n1_risk, opinion_h1n1_sick_from_vacc, opinion_seas_vacc_effective, opinion_seas_ri
sk, opinion_seas_sick_from_vacc, age_group, education, race, sex, income_poverty, marital_status, rent_or_ow
n, employment_status, hhs_geo_region, census_msa, household_adults, household_children, h1n1_vaccine, season
al_vaccine]
Index: []

[0 rows x 34 columns]
```

Our Data does not have any duplicate values

Populating the NULLs/NaNs with suitable substitutes

All Categorical features which are NULL are substituted with the mode of data, and the Numeric features with NULLs are substituted with the means

In [17]:
```
#filling missing values for numeric data types

for col in df.columns:
    if df[col].isnull().sum() and df[col].dtypes != 'object':
        df[col].loc[(df[col].isnull())] = df[col].median()
for col in df.columns:
    if df[col].isnull().sum() and df[col].dtypes == 'object':
        df[col].loc[(df[col].isnull())] = df[col].mode().max()
```

In [18]:
```
# confirming that NULLs/NaNs are successfully substituted
df.isnull().sum()
```

Out[18]:
```
h1n1_concern                   0
h1n1_knowledge                 0
behavioral_antiviral_meds      0
behavioral_avoidance           0
behavioral_face_mask           0
behavioral_wash_hands          0
behavioral_large_gatherings    0
behavioral_outside_home        0
behavioral_touch_face          0
doctor_recc_h1n1               0
doctor_recc_seasonal           0
chronic_med_condition          0
child_under_6_months           0
health_worker                  0
opinion_h1n1_vacc_effective    0
opinion_h1n1_risk              0
opinion_h1n1_sick_from_vacc    0
opinion_seas_vacc_effective    0
opinion_seas_risk              0
opinion_seas_sick_from_vacc    0
age_group                      0
education                      0
race                           0
sex                            0
income_poverty                 0
marital_status                 0
rent_or_own                    0
employment_status              0
hhs_geo_region                 0
census_msa                     0
household_adults               0
household_children             0
h1n1_vaccine                   0
seasonal_vaccine               0
dtype: int64
```

Exploratory Data Analysis

Univariate Analysis

Distribution of the different races

In [19]:
```
# Define the colors for each race category
colors = ['#0077B6', '#0096C7', '#00B4D8', '#48CAE4']

# Create the histogram
plt.figure(figsize=(8, 6))
sns.histplot(data=df, x='race', palette=colors, multiple='stack', edgecolor='black', shrink=0.8)
plt.xlabel('Race')
plt.ylabel('Count')
plt.title('Race Distribution')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Observation

- white race forms majority of the number of people captured in our dataset

Age Distribution

In [20]:
```
# Define the colors for each category
colors = ['#FF0000', '#0000FF', '#00FF00', '#FFFF00', '#800080']

# Get the count of each category
category_count = df['age_group'].value_counts()

# Create the pie chart
plt.figure(figsize=(8, 6))
plt.pie(category_count, labels=category_count.index, colors=colors, autopct='%1.1f%%')
plt.title('Pie Chart of Age Groups')
plt.axis('equal')
plt.show()
```

Observation

- As can be seen above, majority of the people in our dataset were aged 65 years and above

Vaccine Effectives

Vaccine effectives for H1N1

In [21]:
```
# Define the colors for each entry
colors = ['#7DF9FF', '#15f4EE', '#99FFFF', '#ADD8E6', '#87CEFA']

# Get the count of each entry
entry_count = df['opinion_h1n1_vacc_effective'].value_counts()

# Create the bar graph
plt.figure(figsize=(8, 6))
sns.barplot(x=entry_count.index, y=entry_count.values, palette = colors )
plt.xlabel('Opinion on H1N1 Vaccine Effectiveness')
plt.ylabel('Count')
plt.title('Bar Graph of Opinion on H1N1 Vaccine Effectiveness')
plt.xticks(rotation=45)
plt.show()
```

Observation

- majority of the people voted for the H1N1 Vaccine as being somewhat effective. This was followed by those who regard the vaccine as very effective

Seasonal Vaccine effectiveness

In [22]:

```
# Define the colors for each entry
colors = ['#7DF9FF', '#15f4EE', '#99FFFF', '#ADD8E6', '#87CEFA']

# Get the count of each entry
entry_count = df['opinion_seas_vacc_effective'].value_counts()

# Create the bar graph
plt.figure(figsize=(8, 6))
sns.barplot(x=entry_count.index, y=entry_count.values, palette = colors )
plt.xlabel('Opinion on Seasonal Vaccine Effectiveness')
plt.ylabel('Count')
plt.title('Bar Graph of Opinion Seasonal Vaccine Effectiveness')
plt.xticks(rotation=45)
plt.show()
```

Bar Graph of Opinion Seasonal Vaccine Effectiveness
Count
12000
10000
8000
6000
4000
2000
0
Somewhat effective
Very effective
Not very effective
Not at all effective
Don't know
Opinion on Seasonal Vaccine Effectiveness

Observation

- Majority of the people were of opinion that Seasonal Vaccines are somewhat effective followed by those who deem the vaccine as very effective

Handwashing Behavior

In [23]:
```
# Define the colors for each category
color_not_wash = '#FF7F50'  # Color for 0 (People who don't wash their hands)
color_wash = '#811331'  # Color for 1 (People who wash their hands)

# Get the count of each category
category_count = df['behavioral_wash_hands'].value_counts()

# Create the donut chart
plt.figure(figsize=(8, 6))
plt.pie(category_count, labels=category_count.index, autopct='%1.1f%%',
        colors=[color_not_wash, color_wash], wedgeprops=dict(width=0.4))
plt.title('Donut Chart of Handwashing Behavior')
plt.axis('equal')

# Draw a white circle at the center to create the donut shape
center_circle = plt.Circle((0, 0), 0.3, color='white')
fig = plt.gcf()
fig.gca().add_artist(center_circle)
# Create a legend
legend_labels = ['Not Wash Hands', 'Wash Hands']
legend_colors = [color_not_wash, color_wash]
plt.legend(legend_labels, loc='upper right', bbox_to_anchor=(1.1, 1), title='Handwashing Behavior',
           labels=['Do Not Wash Hands', 'Wash Hands'], handlelength=1, handleheight=1,
           edgecolor='none', facecolor='none', framealpha=0.7)
plt.show()
```

Observation

- Above analysis established that the vast majority, about 82.6% of all the the people in our dataset do not wash hands

Bivariate analysis

proportion of people who've received the Vaccine

In [24]:

```
# Step 1: Calculate the count of participants who took each combination of vaccines
vaccine_counts = df.groupby(['h1n1_vaccine', 'seasonal_vaccine']).size()

# Step 2: Create a bar graph for the vaccine combinations
plt.figure(figsize=(6, 4))
vaccine_counts.plot.bar()
plt.xlabel('Vaccine Combinations')
plt.ylabel('Number of People')
plt.title('Number of People who Received Both Vaccines')
plt.legend(['0 - Did not receive', '1 - Received'])
plt.show()

# Step 3: Create separate bar graphs for each vaccine
fig, axes = plt.subplots(nrows=2, ncols=1, figsize=(6, 8))

vaccine_a_counts = df['h1n1_vaccine'].value_counts()
axes[0].bar(vaccine_a_counts.index.astype(int), vaccine_a_counts.values)
axes[0].set_xticks([0, 1])
axes[0].set_xlabel('H1N1 Vaccine')
axes[0].set_ylabel('Number of People')
axes[0].set_title('Number of People who Received H1N1 Vaccine')
axes[0].legend(['0 - Did not receive', '1 - Received'])

vaccine_b_counts = df['seasonal_vaccine'].value_counts()
axes[1].bar(vaccine_b_counts.index.astype(int), vaccine_b_counts.values)
axes[1].set_xticks([0, 1])
axes[1].set_xlabel('Seasonal Vaccine')
axes[1].set_ylabel('Number of People')
axes[1].set_title('Number of People who Received Seasonal Vaccine')
axes[1].legend(['0 - Did not receive', '1 - Received'])

plt.tight_layout()
plt.legend(['0 - Did not receive', '1 - Received'])
plt.show()
```

Number of People who Received Both Vaccines
0 - Did not receive
Number of People
0
2000
4000
6000
8000
10000
12000
(0, 0)
(0, 1)
(1, 0)
(1, 1)
Vaccine Combinations

Number of People who Received H1N1 Vaccine
0 - Did not receive
Number of People
20000
17500
15000
12500
10000
7500
5000
2500
0
0
1
H1N1 Vaccine

Number of People who Received Seasonal Vaccine
0 - Did not receive
Number of People
14000
12000
10000
8000
6000
4000
2000
0

Observation

In [25]:
```
both_vax_ratio = df[(df['h1n1_vaccine']==1) & (df['seasonal_vaccine']==1)].shape[0] / df.shape[0]
either_vax_ratio = df[(df['h1n1_vaccine']==1) | (df['seasonal_vaccine']==1)].shape[0] / df.shape[0]
H1N1 = df[(df['h1n1_vaccine']==1)].shape[0] / df.shape[0]
Seasonal = df[(df['seasonal_vaccine']==1)].shape[0] / df.shape[0]

print("Percentage of respondents who received both vaccines: ", "{:.2%}".format(both_vax_ratio))
print("Percentage of respondents who received one of the vaccines: ", "{:.2%}".format(either_vax_ratio))
print("Percentage of respondents who received only H1N1: ", "{:.2%}".format(H1N1))
print("Percentage of respondents who received only Seasonal Vaccine: ", "{:.2%}".format(Seasonal))
```

```
Percentage of respondents who received both vaccines:  17.59%
Percentage of respondents who received one of the vaccines:  50.22%
Percentage of respondents who received only H1N1:  21.25%
Percentage of respondents who received only Seasonal Vaccine:  46.56%
```

Effect of background on vaccine uptake

In order to analyse the effect of background on vaccine uptake, the following parameters were considered;

- age group
- education
- race
- geographical location
- sex
- employment status of an individual

In [26]:
```
def vaccination_rate_plot(col, target, df, ax=None):
    """Stacked bar chart of vaccination rate for `target` against
    `col`.

    Args:
        col (string): column name of feature variable
        target (string): column name of target variable
        df (pandas DataFrame): dataframe that contains columns
            `col` and `target`
        ax (matplotlib axes object, optional): matplotlib axes
            object to attach plot to
    """
    counts = (df[[target, col]]
                  .groupby([target, col])
                  .size()
                  .unstack(target)
             )
    group_counts = counts.sum(axis='columns')
    props = counts.div(group_counts, axis='index')

    props.plot(kind="barh", stacked=True, ax=ax)
    ax.invert_yaxis()
    ax.legend().remove()

cols_to_plot = [
    'age_group', 'education', 'race',
    'sex', 'employment_status', 'hhs_geo_region',
]

fig, ax = plt.subplots(len(cols_to_plot), 2, figsize=(9, len(cols_to_plot) * 2.5))
for idx, col in enumerate(cols_to_plot):
    vaccination_rate_plot(col, 'h1n1_vaccine', df, ax=ax[idx, 0])
    vaccination_rate_plot(col, 'seasonal_vaccine', df, ax=ax[idx, 1])

ax[0, 0].legend(loc='lower center', bbox_to_anchor=(0.5, 1.05), title='h1n1_vaccine')
ax[0, 1].legend(loc='lower center', bbox_to_anchor=(0.5, 1.05), title='seasonal_vaccine')
fig.tight_layout()
```

h1n1_vaccine
0
1
seasonal_vaccine
0
1
age_group
18 - 34 Years
35 - 44 Years
45 - 54 Years
55 - 64 Years
65+ Years
education
12 Years
< 12 Years
College Graduate
Some College
race
Black
Hispanic
Other or Multiple
White
0.0
0.2
0.4
0.6
0.8
1.0

sex
Female
Male
0.0
0.2
0.4
0.6
0.8
1.0
employment_status
Employed
Not in Labor Force
Unemployed
hhs_geo_region
Town A
Town B
Town C
Town D
Town E
Town F
Town G
Town H
Town I
Town J

Observations

1. **Age group**
    - for the H1N1 Vaccine, the age group between 55-64 recorded the highest
    - for the seasonal vaccine, the highest age group was 65+
2. **Education**
    - college graduates made the majority of those who received H1N1 Vaccine
    - similarly, college graduates made the majority of those who received Seasonal vaccine
3. **race**
    - more people of hispanic origin were vaccinated against H1N1 compared to all other races
    - more white people received Seasonal vaccine than all other races
4. **sex**
    - equal proportion of men and women received H1N1 vaccine
    - slightly more women received the Seasonal Vaccine than men
5. **Employment status**
    - people not in labour force were the highest vaccinated group for both H1N1 and Seasonal
6. **geographical region**
    - people in town F made the majority of those who received H1N1 vaccine and Seasonal Vaccine

Analysing opinion about Vaccine effectiveness.

- Below is an analysis of respondents's opinion regarding the effectiveness of the vaccines

Columns Used

- opinion_h1n1_vacc_effective

- opinion_h1n1_risk
- opinion_seas_vacc_effective
- opinion_h1n1_sick_from_vacc
- opinion_seas_risk
- opinion_seas_sick_from_vacc

In [27]:
```
# Define the renamed opinion columns
opinion_columns = ['opinion_h1n1_vacc_effective', 'opinion_h1n1_risk', 'opinion_seas_vacc_effective',
                   'opinion_h1n1_sick_from_vacc', 'opinion_seas_risk', 'opinion_seas_sick_from_vacc']
renamed_columns = ['H1N1 Vaccine Effectiveness', 'H1N1 Flu Risk', 'Seasonal Vaccine Effectiveness',
                   'Worry about H1N1 Vaccine', 'Seasonal Flu Risk', 'Worry about Seasonal Vaccine']

# Create a copy of the DataFrame with the replaced category codes
df_renamed = df.copy()
replace_mapping = {
    'opinion_h1n1_vacc_effective': {1: 'Not at all effective', 2: 'Not very effective', 3: "Don't know",
                                    4: 'Somewhat effective', 5: 'Very effective'},
    'opinion_h1n1_risk': {1: 'Very Low', 2: 'Somewhat low', 3: "Don't know",
                          4: 'Somewhat high', 5: 'Very high'},
    'opinion_seas_vacc_effective': {1: 'Not at all effective', 2: 'Not very effective', 3: "Don't know",
                                    4: 'Somewhat effective', 5: 'Very effective'},
    'opinion_h1n1_sick_from_vacc': {1: 'Not at all worried', 2: 'Not very worried', 3: "Don't know",
                                    4: 'Somewhat worried', 5: 'Very worried'},
    'opinion_seas_risk': {1: 'Very Low', 2: 'Somewhat low', 3: "Don't know",
                          4: 'Somewhat high', 5: 'Very high'},
    'opinion_seas_sick_from_vacc': {1: 'Not at all worried', 2: 'Not very worried', 3: "Don't know",
                                    4: 'Somewhat worried', 5: 'Very worried'}
}

df_renamed.replace(replace_mapping, inplace=True)

# Plot stacked bar charts
fig, axes = plt.subplots(len(opinion_columns), 1, figsize=(8, 4*len(opinion_columns)))  # Adjust figsize here
fig.suptitle('Bivariate Analysis: Opinions vs. Vaccine Uptake')

for i, ax in enumerate(axes):
    data = df_renamed.groupby([opinion_columns[i], 'h1n1_vaccine', 'seasonal_vaccine']).size().unstack()
    data.plot(kind='bar', stacked=True, ax=ax)

    ax.set_xlabel('Opinion')
    ax.set_ylabel('Count')
    ax.set_title(renamed_columns[i])
    ax.legend(title='Vaccine Uptake', bbox_to_anchor=(1, 1), loc='upper left')
    ax.xaxis.label.set_rotation(45)
    ax.set_xticklabels(data.index.get_level_values(0), rotation=45)  # Set x-axis labels to the brief categori

plt.tight_layout(rect=[0, 0, 1, 0.96])
```

```
plt.show()
```

Observations

1. **opinion_h1n1_vacc_effective** : Respondent's opinion about H1N1 and Seasonal vaccine effectiveness.
    - It was rated as somewhat effective by majority of the people regardless of whether they took the H1N1/Seasonal vaccine or not
2. **opinion_h1n1_risk** :Respondent's opinion about risk of getting sick with H1N1 flu without vaccine.
    - Most respondents thought that the risk of getting sick is Somewhat low regardless of whether they took either vaccine or not
3. **opinion_seas_vacc_effective** :Respondent's opinion about seasonal flu vaccine effectiveness.
    - Most respondents who voted the Seasonal Vaccine as Somewhat effective didn't take the H1N1 vaccine
    - It was voted as Very Effective by Majority of the people who ended up taking the H1N1 vaccine too
    - Respondents who voted Don't Know or Not at All Effective barely ended up taking the Seasonal vaccine.
4. **opinion_h1n1_sick_from_vacc** : Respondent's worry of getting sick from taking H1N1 vaccine.
    - Majority of the people voted Not at all worried and Not very worried ended up having the highest H1N1 vaccine uptake
    - Respondents who voted Don't Know didn't trake either vaccine

5. **opinion_seas_risk** :Respondent's opinion about risk of getting sick with seasonal flu without vaccine.

    - Respondents who voted Somewhat Low didn't weren't keen on getting either vaccine
    - Respondents who voted Somewhat High did take the H1N1 vaccine
6. **opinion_seas_sick_from_vacc** :Respondent's worry of getting sick from taking seasonal flu vaccine.

    - Respondents who voted Not at all worried about getting sick from Seasonal Vaccine had the highest H1N1 vaccine uptake
    - Respondents who voted Don't Know had the least uptake of either vaccine

In [28]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 26707 entries, 0 to 26706
Data columns (total 34 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   h1n1_concern                 26707 non-null  float64
 1   h1n1_knowledge               26707 non-null  float64
 2   behavioral_antiviral_meds    26707 non-null  float64
 3   behavioral_avoidance         26707 non-null  float64
 4   behavioral_face_mask         26707 non-null  float64
 5   behavioral_wash_hands        26707 non-null  float64
 6   behavioral_large_gatherings  26707 non-null  float64
 7   behavioral_outside_home      26707 non-null  float64
 8   behavioral_touch_face        26707 non-null  float64
 9   doctor_recc_h1n1             26707 non-null  float64
 10  doctor_recc_seasonal         26707 non-null  float64
 11  chronic_med_condition        26707 non-null  float64
 12  child_under_6_months         26707 non-null  float64
 13  health_worker                26707 non-null  float64
 14  opinion_h1n1_vacc_effective  26707 non-null  object 
 15  opinion_h1n1_risk            26707 non-null  object 
 16  opinion_h1n1_sick_from_vacc  26707 non-null  object 
 17  opinion_seas_vacc_effective  26707 non-null  object 
 18  opinion_seas_risk            26707 non-null  object 
 19  opinion_seas_sick_from_vacc  26707 non-null  object 
 20  age_group                    26707 non-null  object 
 21  education                    26707 non-null  object 
 22  race                         26707 non-null  object 
 23  sex                          26707 non-null  object 
 24  income_poverty               26707 non-null  object 
 25  marital_status               26707 non-null  object 
 26  rent_or_own                  26707 non-null  object 
 27  employment_status            26707 non-null  object 
 28  hhs_geo_region               26707 non-null  object 
 29  census_msa                   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  
 33  seasonal_vaccine             26707 non-null  int64  
dtypes: float64(16), int64(2), object(16)
memory usage: 7.1+ MB
```

Analysing how health behaviours affect vaccine uptake

Columns Used

- behavioral_antiviral_meds
- behavioral_face_mask
- behavioral_wash_hands
- behavioral_large_gatherings
- behavioral_outside_home
- behavioral_touch_face
- doctor_recc_h1n1
- doctor_recc_seasonal
- chronic_med_condition
- child_under_6_months
- health_worker

In [29]:
```
def vaccination_rate_plot(col, target, df, ax=None):
    """Stacked bar chart of vaccination rate for `target` against
    `col`.

    Args:
        col (string): column name of feature variable
        target (string): column name of target variable
        df (pandas DataFrame): dataframe that contains columns
            `col` and `target`
        ax (matplotlib axes object, optional): matplotlib axes
            object to attach plot to
    """
    counts = (df[[target, col]]
                  .groupby([target, col])
                  .size()
                  .unstack(target)
             )
    group_counts = counts.sum(axis='columns')
    props = counts.div(group_counts, axis='index')

    props.plot(kind="barh", stacked=True, ax=ax)
    ax.invert_yaxis()
    ax.legend().remove()

cols_to_plot = [
   "behavioral_antiviral_meds" ,"behavioral_face_mask" ,"behavioral_wash_hands" ,
    "behavioral_large_gatherings" ,"behavioral_outside_home", "behavioral_touch_face" ,
    "doctor_recc_h1n1" , "doctor_recc_seasonal" , "chronic_med_condition" , "child_under_6_months" ,
    "health_worker"]

fig, ax = plt.subplots(len(cols_to_plot), 2, figsize=(9, len(cols_to_plot) * 2.5))
for idx, col in enumerate(cols_to_plot):
    vaccination_rate_plot(col, 'h1n1_vaccine', df, ax=ax[idx, 0])
    vaccination_rate_plot(col, 'seasonal_vaccine', df, ax=ax[idx, 1])

ax[0, 0].legend(loc='lower center', bbox_to_anchor=(0.5, 1.05), title='h1n1_vaccine')
ax[0, 1].legend(loc='lower center', bbox_to_anchor=(0.5, 1.05), title='seasonal_vaccine')
fig.tight_layout()
plt.show();
```

Observations

1. **behavioral_antiviral_meds** : Has taken antiviral medications.
    - majority peopele who took antiviral meds have higher vaccine uptake
2. **behavioral_face_mask** : Has bought a face mask.
    - majority of the people who bought facemask also took the vaccine
3. **behavioral_wash_hands** : Has frequently washed hands or used hand sanitizer.
    - majority of the people who their hands and used sanitizer also took the vaccine
4. **behavioral_large_gatherings** :Has reduced time at large gatherings.
    - majority of the who avoided large gatherings have higher vaccine uptake
5. **behavioral_outside_home** : Has reduced contact with people outside of own household.
    - majority of people who reduced contact with people outside their own housedholds had higher vaccine upttake
6. **behavioral_touch_face** : Has avoided touching eyes, nose, or mouth.

- majority of the people who reduced touching their face had higher vaccine uptake

7. **doctor_recc_h1n1** : H1N1 flu vaccine was recommended by doctor.

   - majority of the people who had their doctor recommends the H1N1 vaccine took it

8. **doctor_recc_seasonal**:Seasonal flu vaccine was recommended by doctor.

   - majority of the people who had their doctor recommends the Seasonal vaccine took it

9. **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.

   - majority of people with chronic illness took the vaccine

10. **child_under_6_months** :Has regular close contact with a child under the age of six months.

    - majority of the people with children under six months took the vaccine

11. **health_worker** : Is a healthcare worker.

    - majority of healthcare workers had higher vaccine uptake

Identifying numerical data which is useful in establishing correlation between features

In [30]:
```
# Filter columns with numerical data
numeric_columns = df.select_dtypes(include=['int', 'float']).columns

# Print the columns with numerical data
print("Columns with numerical data:")
print(numeric_columns)
```

```
Columns with numerical data:
Index(['h1n1_concern', 'h1n1_knowledge', 'behavioral_antiviral_meds',
       'behavioral_avoidance', 'behavioral_face_mask', 'behavioral_wash_hands',
       'behavioral_large_gatherings', 'behavioral_outside_home',
       'behavioral_touch_face', 'doctor_recc_h1n1', 'doctor_recc_seasonal',
       'chronic_med_condition', 'child_under_6_months', 'health_worker',
       'household_adults', 'household_children', 'h1n1_vaccine',
       'seasonal_vaccine'],
      dtype='object')
```

Ploting correlation map between behavioral patterns and vaccine uptake

In [31]:
```
df_corr = df[[
             'behavioral_antiviral_meds', 'behavioral_avoidance',
             'behavioral_face_mask', 'behavioral_wash_hands', 'behavioral_large_gatherings',
             'behavioral_outside_home', 'behavioral_touch_face', 'doctor_recc_h1n1', 'doctor_recc_seasonal',
             'chronic_med_condition', 'child_under_6_months', 'health_worker' ]]

corr = df_corr.corr()
g = sns.heatmap(corr,  vmax=.3, center=0,
            square=True, linewidths=1, cbar_kws={"shrink": .5}, annot=True, fmt='.2f', cmap='Spectral')
sns.despine()
g.figure.set_size_inches(30,25)
```

behavioral_antiviral_meds 1.00 0.05 0.15 0.06 0.11 0.13 0.07 0.05 0.03 0.00 0.03 0.01
behavioral_avoidance 0.05 1.00 0.06 0.33 0.23 0.22 0.33 0.07 0.07 0.04 0.00 0.00
behavioral_face_mask 0.15 0.06 1.00 0.08 0.18 0.16 0.10 0.08 0.07 0.06 0.04 0.07
behavioral_wash_hands 0.06 0.33 0.08 1.00 0.19 0.19 0.36 0.09 0.10 0.03 0.04 0.05
behavioral_large_gatherings 0.11 0.23 0.18 0.19 1.00 0.58 0.25 0.08 0.09 0.10 0.02 -0.03
behavioral_outside_home 0.13 0.22 0.16 0.19 0.58 1.00 0.27 0.07 0.08 0.09 0.02 -0.04
behavioral_touch_face 0.07 0.33 0.10 0.36 0.25 0.27 1.00 0.09 0.10 0.03 0.03 0.07
doctor_recc_h1n1 0.05 0.07 0.08 0.09 0.08 0.07 0.09 1.00 0.60 0.15 0.08 0.10
doctor_recc_seasonal 0.03 0.07 0.07 0.10 0.09 0.08 0.10 0.60 1.00 0.20 0.04 0.06
chronic_med_condition 0.00 0.04 0.06 0.03 0.10 0.09 0.03 0.15 0.20 1.00 0.00 -0.02
child_under_6_months 0.03 0.00 0.04 0.04 0.02 0.02 0.03 0.08 0.04 0.00 1.00 0.08
health_worker 0.01 0.00 0.07 0.05 -0.03 -0.04 0.07 0.10 0.06 -0.02 0.08 1.00
0.30
0.25
0.20
0.15
0.10
0.05
0.00

behav b bı beh behavio beha bc c ch cl

Observations

- high positive correlation was noted between behavioral patterns and vaccine uptake.
- High Positive Correlations between opinion of doctor recommendation of vaccines Vs whether the person really took the vaccine.
- Overall the data features seems to be positively correlating with the act of taking the vaccination, except with some rare differences

REVERTING OPINION COLUMNS TO NUMERICAL DATA

In [32]:
```
# Respondent's opinion about H1N1 vaccine effectiveness.
reverse_opinion_h1n1_vacc_effective = {
    "Not at all effective": 1,
    "Not very effective": 2,
    "Don't know": 3,
    "Somewhat effective": 4,
    "Very effective": 5
}
df.opinion_h1n1_vacc_effective = df.opinion_h1n1_vacc_effective.map(reverse_opinion_h1n1_vacc_effective)

# Respondent's opinion about risk of getting sick with H1N1 flu without vaccine.
reverse_opinion_h1n1_risk = {
    "Very Low": 1,
    "Somewhat low": 2,
    "Don't know": 3,
    "Somewhat high": 4,
    "Very high": 5
}
df.opinion_h1n1_risk = df.opinion_h1n1_risk.map(reverse_opinion_h1n1_risk)

# Respondent's opinion about seasonal flu vaccine effectiveness.
reverse_opinion_seas_vacc_effective = {
    "Not at all effective": 1,
    "Not very effective": 2,
    "Don't know": 3,
    "Somewhat effective": 4,
    "Very effective": 5
}
df.opinion_seas_vacc_effective = df.opinion_seas_vacc_effective.map(reverse_opinion_seas_vacc_effective)

# Respondent's opinion about getting sick from H1N1 vaccine.
reverse_opinion_h1n1_sick_from_vacc = {
    "Not at all worried": 1,
    "Not very worried": 2,
    "Don't know": 3,
    "Somewhat worried": 4,
    "Very worried": 5
}
df.opinion_h1n1_sick_from_vacc = df.opinion_h1n1_sick_from_vacc.map(reverse_opinion_h1n1_sick_from_vacc)

# Respondent's opinion about risk of getting sick with seasonal flu without vaccine.
reverse_opinion_seas_risk = {
    "Very Low": 1,
```

```
    "Somewhat low": 2,
    "Don't know": 3,
    "Somewhat high": 4,
    "Very high": 5
}
df.opinion_seas_risk = df.opinion_seas_risk.map(reverse_opinion_seas_risk)

# Respondent's worry of getting sick from taking seasonal flu vaccine.
reverse_opinion_seas_sick_from_vacc = {
    "Not at all worried": 1,
    "Not very worried": 2,
    "Don't know": 3,
    "Somewhat worried": 4,
    "Very worried": 5
}
df.opinion_seas_sick_from_vacc = df.opinion_seas_sick_from_vacc.map(reverse_opinion_seas_sick_from_vacc)
```

Creating a New DataFrame

Creating a new dataframe with our selected features

In [33]:
```
model_df = df[["age_group","education","race", "sex","employment_status",
               "opinion_h1n1_vacc_effective" ,"opinion_h1n1_risk" , "opinion_seas_vacc_effective",
               "opinion_h1n1_sick_from_vacc","opinion_seas_risk","opinion_seas_sick_from_vacc",
               "behavioral_antiviral_meds" ,"behavioral_face_mask" ,"behavioral_wash_hands" ,
               "behavioral_large_gatherings" ,"behavioral_outside_home", "behavioral_touch_face" ,
               "doctor_recc_h1n1" , "doctor_recc_seasonal" , "chronic_med_condition" , "child_under_6_months"
               "health_worker"]
]
model_df
```

Out[33]:

respondent_id	age_group	education	race	sex	employment_status	opinion_h1n1_vacc_effective	opinion_h1n1_risk	opinion_sea
0	55 - 64 Years	< 12 Years	White	Female	Not in Labor Force	3	1	
1	35 - 44 Years	12 Years	White	Male	Employed	5	4	
2	18 - 34 Years	College Graduate	White	Male	Employed	3	1	
3	65+ Years	12 Years	White	Female	Not in Labor Force	3	3	
4	45 - 54 Years	Some College	White	Female	Employed	3	3	
...	...	...	...	...	...	...	...	
26702	65+ Years	Some College	White	Female	Not in Labor Force	3	1	
26703	18 - 34 Years	College Graduate	White	Male	Employed	4	2	
26704	55 - 64 Years	Some College	White	Female	Employed	4	4	
26705	18 - 34 Years	Some College	Hispanic	Female	Employed	3	1	
26706	65+ Years	Some College	White	Male	Not in Labor Force	5	1	

26707 rows × 22 columns

Correlation between the selected features

In [34]:
```
corr = model_df.corr()
g = sns.heatmap(corr,  vmax=.3, center=0,
            square=True, linewidths=1, cbar_kws={"shrink": .5}, annot=True, fmt='.2f', cmap='coolwarm')
sns.despine()
g.figure.set_size_inches(30,25)

plt.show()
```

opinion_h1n1_vacc_effective 1.00 0.26 0.47 0.06 0.26 0.01 0.03 0.04 0.13 0.05 0.05 0.10 0.15 0.12 0.05 0.01 0.05
opinion_h1n1_risk 0.26 1.00 0.22 0.34 0.56 0.20 0.10 0.13 0.17 0.13 0.12 0.14 0.25 0.16 0.12 0.09 0.12
opinion_seas_vacc_effective 0.47 0.22 1.00 0.08 0.34 -0.02 0.01 0.04 0.14 0.08 0.07 0.10 0.10 0.17 0.09 0.01 0.03
opinion_h1n1_sick_from_vacc 0.06 0.34 0.08 1.00 0.27 0.49 0.08 0.10 0.15 0.18 0.17 0.13 0.11 0.06 0.08 0.04 0.01
opinion_seas_risk 0.26 0.56 0.34 0.27 1.00 0.21 0.08 0.11 0.17 0.13 0.12 0.14 0.19 0.23 0.17 0.05 0.09
opinion_seas_sick_from_vacc 0.01 0.20 -0.02 0.49 0.21 1.00 0.08 0.08 0.09 0.13 0.13 0.09 0.06 0.03 0.06 0.04 -0.01
behavioral_antiviral_meds 0.03 0.10 0.01 0.08 0.08 0.08 1.00 0.15 0.06 0.11 0.13 0.07 0.05 0.03 0.00 0.03 0.01
behavioral_face_mask 0.04 0.13 0.04 0.10 0.11 0.08 0.15 1.00 0.08 0.18 0.16 0.10 0.08 0.07 0.06 0.04 0.07
behavioral_wash_hands 0.13 0.17 0.14 0.15 0.17 0.09 0.06 0.08 1.00 0.19 0.19 0.36 0.09 0.10 0.03 0.04 0.05
behavioral_large_gatherings 0.05 0.13 0.08 0.18 0.13 0.13 0.11 0.18 0.19 1.00 0.58 0.25 0.08 0.09 0.10 0.02 -0.03
behavioral_outside_home 0.05 0.12 0.07 0.17 0.12 0.13 0.13 0.16 0.19 0.58 1.00 0.27 0.07 0.08 0.09 0.02 -0.04
behavioral_touch_face 0.10 0.14 0.10 0.13 0.14 0.09 0.07 0.10 0.36 0.25 0.27 1.00 0.09 0.10 0.03 0.03 0.07
doctor_recc_h1n1 0.15 0.25 0.10 0.11 0.19 0.06 0.05 0.08 0.09 0.08 0.07 0.09 1.00 0.60 0.15 0.08 0.10
doctor_recc_seasonal 0.12 0.16 0.17 0.06 0.23 0.03 0.03 0.07 0.10 0.09 0.08 0.10 0.60 1.00 0.20 0.04 0.06
chronic_med_condition 0.05 0.12 0.09 0.08 0.17 0.06 0.00 0.06 0.03 0.10 0.09 0.03 0.15 0.20 1.00 0.00 -0.02
child_under_6_months 0.01 0.09 0.01 0.04 0.05 0.04 0.03 0.04 0.04 0.02 0.02 0.03 0.08 0.04 0.00 1.00 0.08
health_worker 0.05 0.12 0.03 0.01 0.09 -0.01 0.01 0.07 0.05 -0.03 -0.04 0.07 0.10 0.06 -0.02 0.08 1.00
opinion_h1n1_risk
opinion_h1n1_sick_from_vacc
opinion_seas_risk
behavioral_face_mask
doctor_recc_h1n1
doctor_recc_seasonal
health_worker
0.30
0.25
0.20
0.15
0.10
0.05
0.00

In [35]:
```
model_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 26707 entries, 0 to 26706
Data columns (total 22 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   age_group                    26707 non-null  object 
 1   education                    26707 non-null  object 
 2   race                         26707 non-null  object 
 3   sex                          26707 non-null  object 
 4   employment_status            26707 non-null  object 
 5   opinion_h1n1_vacc_effective  26707 non-null  int64  
 6   opinion_h1n1_risk            26707 non-null  int64  
 7   opinion_seas_vacc_effective  26707 non-null  int64  
 8   opinion_h1n1_sick_from_vacc  26707 non-null  int64  
 9   opinion_seas_risk            26707 non-null  int64  
 10  opinion_seas_sick_from_vacc  26707 non-null  int64  
 11  behavioral_antiviral_meds    26707 non-null  float64
 12  behavioral_face_mask         26707 non-null  float64
 13  behavioral_wash_hands        26707 non-null  float64
 14  behavioral_large_gatherings  26707 non-null  float64
 15  behavioral_outside_home      26707 non-null  float64
 16  behavioral_touch_face        26707 non-null  float64
 17  doctor_recc_h1n1             26707 non-null  float64
 18  doctor_recc_seasonal         26707 non-null  float64
 19  chronic_med_condition        26707 non-null  float64
 20  child_under_6_months         26707 non-null  float64
 21  health_worker                26707 non-null  float64
dtypes: float64(11), int64(6), object(5)
memory usage: 4.7+ MB
```

Feature Selection, Extraction & Engineering

In [36]:
```
model_df['Behaviour'] = model_df[['behavioral_antiviral_meds', 'behavioral_face_mask', 'behavioral_wash_hands
                                  'behavioral_large_gatherings', 'behavioral_outside_home', 'behavioral_touch_

sns.set(style="darkgrid")

# Create the countplot
sns.countplot(x='Behaviour', data=model_df)

# Set labels and title
plt.xlabel('Behaviour')
plt.ylabel('Count')
plt.title('Count of Behaviour Ratings')

# Show the plot
plt.show()
```

Count of Behaviour Ratings
Count
8000
7000
6000
5000
4000
3000
2000
1000
0
0.0
1.0
2.0
3.0
4.0
5.0
6.0
Behaviour

In [37]:
```
model_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 26707 entries, 0 to 26706
Data columns (total 23 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   age_group                    26707 non-null  object 
 1   education                    26707 non-null  object 
 2   race                         26707 non-null  object 
 3   sex                          26707 non-null  object 
 4   employment_status            26707 non-null  object 
 5   opinion_h1n1_vacc_effective  26707 non-null  int64  
 6   opinion_h1n1_risk            26707 non-null  int64  
 7   opinion_seas_vacc_effective  26707 non-null  int64  
 8   opinion_h1n1_sick_from_vacc  26707 non-null  int64  
 9   opinion_seas_risk            26707 non-null  int64  
 10  opinion_seas_sick_from_vacc  26707 non-null  int64  
 11  behavioral_antiviral_meds    26707 non-null  float64
 12  behavioral_face_mask         26707 non-null  float64
 13  behavioral_wash_hands        26707 non-null  float64
 14  behavioral_large_gatherings  26707 non-null  float64
 15  behavioral_outside_home      26707 non-null  float64
 16  behavioral_touch_face        26707 non-null  float64
 17  doctor_recc_h1n1             26707 non-null  float64
 18  doctor_recc_seasonal         26707 non-null  float64
 19  chronic_med_condition        26707 non-null  float64
 20  child_under_6_months         26707 non-null  float64
 21  health_worker                26707 non-null  float64
 22  Behaviour                    26707 non-null  float64
dtypes: float64(12), int64(6), object(5)
memory usage: 4.9+ MB
```

Encoding Categorical datatype

In [38]:
```
temp = model_df[['age_group', 'race', 'sex', 'employment_status','education']].copy() #perhaps we can labelenc
for col in temp.columns.tolist():
    label_encoder = LabelEncoder()
    temp[col] = label_encoder.fit_transform(temp[col])

model_df.drop(columns=['age_group', 'race', 'sex', 'employment_status','education'], inplace=True)
model_df1 = pd.concat([temp,model_df], axis=1)

model_df1.head()
```

Out[38]:

	age_group	race	sex	employment_status	education	opinion_h1n1_vacc_effective	opinion_h1n1_risk	opinion_seas_vacc
respondent_id								
0	3	3	0	1	1	3	1	
1	1	3	1	0	0	5	4	
2	0	3	1	0	2	3	1	
3	4	3	0	1	0	3	3	
4	2	3	0	0	3	3	3	

5 rows × 23 columns

Modelling

In [39]:
```
X = model_df1
y = df[['h1n1_vaccine','seasonal_vaccine']]
```

Splitting the Dataset

In [40]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state =99)
```

In [41]:
```
X_train.head()
```

Out[41]:

	age_group	race	sex	employment_status	education	opinion_h1n1_vacc_effective	opinion_h1n1_risk	opinion_seas_vacc
respondent_id								
22191	4	3	1	1	2	4	2	
1351	3	3	0	0	3	5	4	
25293	4	1	0	1	2	4	2	
16249	3	3	1	1	1	5	1	
3219	2	3	0	0	1	5	2	

5 rows × 23 columns

In [42]:
```
X_test.head()
```

Out[42]:

	age_group	race	sex	employment_status	education	opinion_h1n1_vacc_effective	opinion_h1n1_risk	opinion_seas_vacc
respondent_id								
20785	0	0	0	1	3	4	2	
18587	4	1	0	1	3	4	1	
2666	4	0	0	1	3	1	1	
4574	2	3	0	0	2	5	2	
3391	2	3	0	0	2	4	2	

5 rows × 23 columns

In [43]:
```
y_train.head()
```

Out[43]:

respondent_id	h1n1_vaccine	seasonal_vaccine
22191	0	1
1351	1	1
25293	0	0
16249	0	0
3219	0	1

In [44]:
```
y_test.head()
```

Out[44]:

respondent_id	h1n1_vaccine	seasonal_vaccine
20785	1	1
18587	0	0
2666	0	1
4574	1	0
3391	0	0

In [45]:
```
print(f'y_train shape: {y_train.shape}')
print(f'y_test shape: {y_test.shape}')
print(f'X_train shape: {X_train.shape}')
print(f'X_test shape: {X_test.shape}')
```

```
y_train shape: (21365, 2)
y_test shape: (5342, 2)
X_train shape: (21365, 23)
X_test shape: (5342, 23)
```

Scaling the model

In [46]:
```
scaler = MinMaxScaler()  # Create an instance of MinMaxScaler

X_train = scaler.fit_transform(X_train)  # Scale the training set
X_test = scaler.transform(X_test)  # Scale the test set
```

Model Selection

1. KNN
2. Decision Trees
3. Logistics Regression
4. Random Forest
5. Naive Bayes

In [47]:
```
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

# Create a list of classifiers
classifiers = [
    LogisticRegression(),
    DecisionTreeClassifier(),
    KNeighborsClassifier(),
    GaussianNB(),
    RandomForestClassifier()
]

# Define the names of the columns corresponding to the target variables
column_names = ["H1N1 Vaccine", "Seasonal Vaccine"]

# Initialize lists to store results for each classifier and each column
results = []
h1n1_accuracy_scores = []  # Accuracy scores for H1N1 vaccine
seasonal_accuracy_scores = []  # Accuracy scores for seasonal vaccine

# Iterate over each classifier
for classifier in classifiers:
    # Initialize lists to store results for each column
    y_pred_list = []
    avg_recall_list = []
    avg_precision_list = []
    avg_h1n1_accuracy_list = []  # Accuracy scores for H1N1 vaccine
    avg_seasonal_accuracy_list = []  # Accuracy scores for seasonal vaccine

    # Iterate over each column of y_train
    for i in range(y_train.shape[1]):
        # Select the i-th column as the target variable
        y_train_selected = y_train.iloc[:, i]

        # Perform cross-validation prediction
        y_pred = cross_val_predict(classifier, X_train, y_train_selected, cv=5)

        # Calculate evaluation metrics
        avg_recall = recall_score(y_train_selected, y_pred, average='weighted')
        avg_precision = precision_score(y_train_selected, y_pred, average='weighted')
        avg_accuracy = accuracy_score(y_train_selected, y_pred)

        # Append results to the respective lists
```

```
        y_pred_list.append(y_pred)
        avg_recall_list.append(avg_recall)
        avg_precision_list.append(avg_precision)

        # Append accuracy scores to the respective lists based on the column index
        if i == 0:
            avg_h1n1_accuracy_list.append(avg_accuracy)
        else:
            avg_seasonal_accuracy_list.append(avg_accuracy)

    # Append results for the current classifier to the overall results list
    results.append((classifier, y_pred_list, avg_recall_list, avg_precision_list, avg_h1n1_accuracy_list, avg_

    # Compute average accuracy score for H1N1 vaccine
    avg_h1n1_accuracy_score = sum(avg_h1n1_accuracy_list) / len(avg_h1n1_accuracy_list)
    h1n1_accuracy_scores.append(avg_h1n1_accuracy_score)

    # Compute average accuracy score for seasonal vaccine
    if avg_seasonal_accuracy_list:
        avg_seasonal_accuracy_score = sum(avg_seasonal_accuracy_list) / len(avg_seasonal_accuracy_list)
        seasonal_accuracy_scores.append(avg_seasonal_accuracy_score)
    else:
        seasonal_accuracy_scores.append(None)

# Print the results for each classifier and each column
for result in results:
    classifier, y_pred_list, avg_recall_list, avg_precision_list, avg_h1n1_accuracy_list, avg_seasonal_accurac
    classifier_name = type(classifier).__name__
    print(f"Results for classifier: {classifier_name}")
    print("-----------------------------")
    for i in range(y_train.shape[1]):
        column_name = column_names[i]
        print(f"Results for {column_name}:")
        print("----------------------------")
        print("Predicted labels:", y_pred_list[i])
        print("Average Recall:", avg_recall_list[i])
        print("Average Precision:", avg_precision_list[i])
        if i == 0:
            print("Average Accuracy:", avg_h1n1_accuracy_list[i])
        else:
            print("Average Accuracy:", avg_seasonal_accuracy_list[i-1])
        print("\n")
```

```
Results for classifier: LogisticRegression
-----------------------------
Results for H1N1 Vaccine:
----------------------------
Predicted labels: [0 1 0 ... 0 0 0]
Average Recall: 0.8343552539199626
Average Precision: 0.8198556869415087
Average Accuracy: 0.8343552539199626


Results for Seasonal Vaccine:
----------------------------
Predicted labels: [1 1 0 ... 1 1 0]
Average Recall: 0.7735548794757782
Average Precision: 0.7734404702546603
Average Accuracy: 0.7735548794757782


Results for classifier: DecisionTreeClassifier
-----------------------------
Results for H1N1 Vaccine:
----------------------------
Predicted labels: [0 1 0 ... 0 0 0]
Average Recall: 0.7534285045635385
Average Precision: 0.758547286512052
Average Accuracy: 0.7534285045635385


Results for Seasonal Vaccine:
----------------------------
Predicted labels: [1 1 0 ... 1 1 0]
Average Recall: 0.676761057804821
Average Precision: 0.6765018490909228
Average Accuracy: 0.676761057804821


Results for classifier: KNeighborsClassifier
-----------------------------
Results for H1N1 Vaccine:
----------------------------
Predicted labels: [0 1 0 ... 0 0 0]
Average Recall: 0.8100163819330681
Average Precision: 0.7928360204046846
```

```
Average Accuracy: 0.8100163819330681


Results for Seasonal Vaccine:
----------------------------
Predicted labels: [1 1 1 ... 0 1 0]
Average Recall: 0.7274982447928856
Average Precision: 0.7273666433480998
Average Accuracy: 0.7274982447928856


Results for classifier: GaussianNB
-----------------------------
Results for H1N1 Vaccine:
----------------------------
Predicted labels: [0 1 0 ... 0 0 1]
Average Recall: 0.7749590451673297
Average Precision: 0.8006475744313655
Average Accuracy: 0.7749590451673297


Results for Seasonal Vaccine:
----------------------------
Predicted labels: [1 1 0 ... 1 0 1]
Average Recall: 0.7259536625321787
Average Precision: 0.7258503453187941
Average Accuracy: 0.7259536625321787


Results for classifier: RandomForestClassifier
-----------------------------
Results for H1N1 Vaccine:
----------------------------
Predicted labels: [0 1 0 ... 0 0 0]
Average Recall: 0.8293470629534285
Average Precision: 0.8139315661225037
Average Accuracy: 0.8293470629534285


Results for Seasonal Vaccine:
----------------------------
Predicted labels: [1 1 0 ... 1 1 0]
Average Recall: 0.7618534987128481
```

```
Average Precision: 0.7616001427836259
Average Accuracy: 0.7618534987128481
```

In [48]:
```
# Plot the accuracy scores for H1N1 vaccine
plt.figure(figsize=(10, 6))
plt.bar([type(classifier).__name__ for classifier in classifiers], h1n1_accuracy_scores)
plt.xlabel('Classifiers')
plt.ylabel('Accuracy Score')
plt.title('Accuracy Scores of Different Classifiers for H1N1 Vaccine')
plt.xticks(rotation=45)
plt.ylim([0, 1])
plt.show()

# Plot the accuracy scores for seasonal vaccine
plt.figure(figsize=(10, 6))
plt.bar([type(classifier).__name__ for classifier in classifiers], seasonal_accuracy_scores)
plt.xlabel('Classifiers')
plt.ylabel('Accuracy Score')
plt.title('Accuracy Scores of Different Classifiers for Seasonal Vaccine')
plt.xticks(rotation=45)
plt.ylim([0, 1])
plt.show()
```

Accuracy Scores of Different Classifiers for H1N1 Vaccine
1.0
0.8
0.6
0.4
0.2
0.0
Accuracy Score
LogisticRegression
DecisionTreeClassifier
KNeighborsClassifier
GaussianNB
RandomForestClassifier
Classifiers

Accuracy Scores of Different Classifiers for the training data

In [49]:
```
# Fit the models with the training data
for result in results:
    classifier, y_pred_list, _, _, _, _ = result
    for i in range(y_train.shape[1]):
        y_train_selected = y_train.iloc[:, i]
        classifier.fit(X_train, y_train_selected)

# Evaluate the models on the test set
for result in results:
    classifier, y_pred_list, _, _, _, _ = result
    classifier_name = type(classifier).__name__
    print(f"Evaluating classifier: {classifier_name} on the test set")
    print("-----------------------------")
    for i in range(y_test.shape[1]):
        column_name = column_names[i]
        print(f"Evaluating {column_name}:")
        print("----------------------------")
        y_pred_test = classifier.predict(X_test)
        avg_recall_test = recall_score(y_test.iloc[:, i], y_pred_test, average='weighted')
        avg_precision_test = precision_score(y_test.iloc[:, i], y_pred_test, average='weighted')
        avg_accuracy_test = accuracy_score(y_test.iloc[:, i], y_pred_test)
        print("Predicted labels:", y_pred_test)
        print("Average Recall on Test Set:", avg_recall_test)
        print("Average Precision on Test Set:", avg_precision_test)
        print("Average Accuracy on Test Set:", avg_accuracy_test)
        print("\n")
```

```
Evaluating classifier: LogisticRegression on the test set
-----------------------------
Evaluating H1N1 Vaccine:
----------------------------
Predicted labels: [1 0 1 ... 1 1 1]
Average Recall on Test Set: 0.6458255335080494
Average Precision on Test Set: 0.7699108482785715
Average Accuracy on Test Set: 0.6458255335080494


Evaluating Seasonal Vaccine:
----------------------------
Predicted labels: [1 0 1 ... 1 1 1]
Average Recall on Test Set: 0.7671284163234744
Average Precision on Test Set: 0.7672863480672123
Average Accuracy on Test Set: 0.7671284163234744


Evaluating classifier: DecisionTreeClassifier on the test set
-----------------------------
Evaluating H1N1 Vaccine:
----------------------------
Predicted labels: [1 0 0 ... 1 1 1]
Average Recall on Test Set: 0.5898539872706852
Average Precision on Test Set: 0.7327339306240381
Average Accuracy on Test Set: 0.5898539872706852


Evaluating Seasonal Vaccine:
----------------------------
Predicted labels: [1 0 0 ... 1 1 1]
Average Recall on Test Set: 0.6767128416323475
Average Precision on Test Set: 0.676500885503287
Average Accuracy on Test Set: 0.6767128416323475


Evaluating classifier: KNeighborsClassifier on the test set
-----------------------------
Evaluating H1N1 Vaccine:
----------------------------
Predicted labels: [0 0 1 ... 0 1 1]
Average Recall on Test Set: 0.6239236241108199
Average Precision on Test Set: 0.7612125583190442
```

```
Average Accuracy on Test Set: 0.6239236241108199


Evaluating Seasonal Vaccine:
----------------------------
Predicted labels: [0 0 1 ... 0 1 1]
Average Recall on Test Set: 0.729502059153875
Average Precision on Test Set: 0.7292200659415741
Average Accuracy on Test Set: 0.729502059153875


Evaluating classifier: GaussianNB on the test set
-----------------------------
Evaluating H1N1 Vaccine:
----------------------------
Predicted labels: [1 0 0 ... 0 1 1]
Average Recall on Test Set: 0.647135904155747
Average Precision on Test Set: 0.7831760276686088
Average Accuracy on Test Set: 0.647135904155747


Evaluating Seasonal Vaccine:
----------------------------
Predicted labels: [1 0 0 ... 0 1 1]
Average Recall on Test Set: 0.7302508423811307
Average Precision on Test Set: 0.729966589715835
Average Accuracy on Test Set: 0.7302508423811307


Evaluating classifier: RandomForestClassifier on the test set
-----------------------------
Evaluating H1N1 Vaccine:
----------------------------
Predicted labels: [1 0 1 ... 1 1 1]
Average Recall on Test Set: 0.6252339947585174
Average Precision on Test Set: 0.7613982968800407
Average Accuracy on Test Set: 0.6252339947585174


Evaluating Seasonal Vaccine:
----------------------------
Predicted labels: [1 0 1 ... 1 1 1]
Average Recall on Test Set: 0.7626357169599401
```

```
Average Precision on Test Set: 0.7624243243109231
Average Accuracy on Test Set: 0.7626357169599401
```

In [50]:
```
# Plot the accuracy scores
plt.figure(figsize=(10, 6))
plt.bar([type(classifier).__name__ for classifier in classifiers], h1n1_accuracy_scores)
plt.xlabel('Classifiers')
plt.ylabel('Accuracy Score')
plt.title('Accuracy Scores of Different Classifiers for the H1N1 Vaccine, TEST')
plt.xticks(rotation=45)
plt.ylim([0, 1])
plt.show()

plt.figure(figsize=(10, 6))
plt.bar([type(classifier).__name__ for classifier in classifiers], seasonal_accuracy_scores)
plt.xlabel('Classifiers')
plt.ylabel('Accuracy Score')
plt.title('Accuracy Scores of Different Classifiers for the Seasonal Vaccine, TEST')
plt.xticks(rotation=45)
plt.ylim([0, 1])
plt.show()
```

Accuracy Scores of Different Classifiers for the H1N1 Vaccine, TEST
1.0
0.8
0.6
0.4
0.2
0.0
Accuracy Score
LogisticRegression
DecisionTreeClassifier
KNeighborsClassifier
GaussianNB
RandomForestClassifier
Classifiers

Accuracy Scores of Different Classifiers for the Seasonal Vaccine, TEST
1.0
0.8
0.6
0.4
0.2
0.0
Accuracy Score
LogisticRegression
DecisionTreeClassifier
KNeighborsClassifier
GaussianNB
RandomForestClassifier
Classifiers

ROC-CURVE & AUC

In [51]:
```
# Create the figure and subplots with the desired size
fig, axs = plt.subplots(nrows=len(classifiers), ncols=2, figsize=(16, 25))

# Adjust the aspect ratio of each subplot
for ax in axs.flat:
    ax.set_aspect('auto')

# Iterate over each classifier
for idx, classifier in enumerate(classifiers):
    classifier_name = type(classifier).__name__

    # Iterate over columns 1 and 2
    for col in [0, 1]:
        # Select the column as the target variable
        y_train_selected = y_train.iloc[:, col]
        y_test_selected = y_test.iloc[:, col]

        # Fit the classifier
        classifier.fit(X_train, y_train_selected)

        # Predict the probabilities for the positive class
        y_pred_prob = classifier.predict_proba(X_test)[:, 1]

        # Compute the ROC curve
        fpr, tpr, thresholds = roc_curve(y_test_selected, y_pred_prob)

        # Compute the AUC
        auc = roc_auc_score(y_test_selected, y_pred_prob)

        # Set the subplot location
        ax = axs[idx, col]

        # Plot the ROC curve with larger line width
        ax.plot(fpr, tpr, label='ROC Curve (AUC = %0.2f)' % auc, linewidth=2, color = 'b')
        ax.plot([0, 1], [0, 1], 'k--', color = 'r')
        ax.set_xlim([0, 1])
        ax.set_ylim([0, 1.05])
        ax.set_xlabel('False Positive Rate')
        ax.set_ylabel('True Positive Rate')

        # Set the title as the classifier name
        ax.set_title(classifier_name, fontsize=12, fontweight='bold', pad=15)
```

```
        # Add the column name at the top of the subplot
        ax.text(0.5, 0.95, column_names[col], transform=ax.transAxes, ha='center', va='top')

        # Add the legend
        ax.legend(loc='lower right')

# Adjust the subplot layout
plt.tight_layout(pad = 3)

# Add a title to the entire figure
fig.suptitle('ROC Curves for H1N1 Vaccine and Seasonal Vaccine', fontsize=14, fontweight='bold')

# Display the plot
plt.show()
```

ROC Curves for H1N1 Vaccine and Seasonal Vaccine
LogisticRegression
H1N1 Vaccine
ROC Curve (AUC = 0.81)
LogisticRegression
Seasonal Vaccine
ROC Curve (AUC = 0.84)
DecisionTreeClassifier
H1N1 Vaccine
ROC Curve (AUC = 0.64)
DecisionTreeClassifier
Seasonal Vaccine
ROC Curve (AUC = 0.67)
KNeighborsClassifier
H1N1 Vaccine
KNeighborsClassifier
Seasonal Vaccine
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

Model Evaluation

ROC/AUC curves Logistic Regression and the RandomForest had the highest ROC and Accuracy scores. The Logistic regression had a score of 81% and 84% for H1N1 and Seasonal Vaccine respectively while the Random Forest Classifier had scores of 80% and 83 for H1N1 and Seasonal Vacc respectively. Since the Logistic Regression was the baseline model, we chose to go with Random Forest

Classifier which has ROC score of 83%

Hyperparameter Tuning

To improve the models perfomance we decided to do Hyper Parameter tuning using GridSearch

In [52]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
import numpy as np

# Create a Random Forest classifier
rf_classifier = RandomForestClassifier()

# Initialize a list to store the best models for each column of y_train
best_models = []

# Perform Grid Search for each column of y_train
for i in range(y_train.shape[1]):
    # Define the hyperparameter grid to search
    param_grid = {
        'n_estimators': [50, 100, 150],
        'max_depth': [None, 5, 10, 20],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4]
    }

    # Select the i-th column of y_train as the target variable
    y_train_selected = np.array(y_train)[:, i]

    # Create the Grid Search object
    grid_search = GridSearchCV(estimator=rf_classifier, param_grid=param_grid, cv=5, scoring='accuracy')

    # Perform Grid Search on the training data
    grid_search.fit(X_train, y_train_selected)

    # Get the best hyperparameters found during the search
    best_params = grid_search.best_params_
    print(f"Best Hyperparameters for column {i+1}:", best_params)

    # Get the best model with the best hyperparameters
    best_rf_model = grid_search.best_estimator_
    best_models.append(best_rf_model)

# Evaluate the best models on the test data
test_accuracies = []
for i, best_model in enumerate(best_models):
    y_test_selected = np.array(y_test)[:, i]
    test_accuracy = best_model.score(X_test, y_test_selected)
```

```
    test_accuracies.append(test_accuracy)
    print(f"Test Accuracy for column {i+1}:", test_accuracy)

# Compute the average test accuracy across all columns
average_test_accuracy = np.mean(test_accuracies)
print("Average Test Accuracy:", average_test_accuracy)
```

```
Best Hyperparameters for column 1: {'max_depth': 20, 'min_samples_leaf': 4, 'min_samples_split': 2, 'n_estim
ators': 50}
Best Hyperparameters for column 2: {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_esti
mators': 50}
Test Accuracy for column 1: 0.8313365780606514
Test Accuracy for column 2: 0.7748034444028454
Average Test Accuracy: 0.8030700112317484
```

Hyperparameters: The best hyperparameters found for both columns are similar, indicating that the same configuration performs well for both vaccines. These hyperparameters include 'max_depth' (unlimited), 'min_samples_leaf' (4), 'min_samples_split' (5 for H1N1 and 2 for Seasonal), and 'n_estimators' (100). These hyperparameters define the behavior of the machine learning model.

Test Accuracy: The test accuracy measures the performance of the model on unseen test data for each vaccine. For the H1N1 Vaccine (column 1), the model achieved a test accuracy of approximately 83.13%, while for the Seasonal Vaccine (column 2), the model achieved a test accuracy of around 77.48%. These accuracies indicate the percentage of correct predictions made by the model for each vaccine.

Average Test Accuracy: The average test accuracy, computed by taking the mean of the individual test accuracies, is approximately 80.31%. This average provides an overall evaluation of the model's performance across both vaccines, indicating that the model performs reasonably well on average for predicting the vaccine outcome.

In [55]:
```
from sklearn.ensemble import RandomForestClassifier

# Assuming you have your data and target variables: X_train, y_train
rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)


# Create a DataFrame with column names and their importances
importance_df = pd.DataFrame({'Feature': model_df1.columns, 'Importance': feature_importances})

# Sort the DataFrame by importance in descending order
importance_df = importance_df.sort_values(by='Importance', ascending=False)

# Display the feature importances table
print(importance_df)
```

```
                       Feature  Importance
9            opinion_seas_risk    0.090485
0                    age_group    0.087491
7   opinion_seas_vacc_effective    0.078868
6            opinion_h1n1_risk    0.066892
5   opinion_h1n1_vacc_effective    0.063689
4                    education    0.061875
8   opinion_h1n1_sick_from_vacc    0.060393
18        doctor_recc_seasonal    0.058911
17            doctor_recc_h1n1    0.058811
10  opinion_seas_sick_from_vacc    0.055948
22                   Behaviour    0.052487
1                         race    0.035577
3            employment_status    0.035330
2                          sex    0.033840
19       chronic_med_condition    0.027315
21               health_worker    0.022959
14  behavioral_large_gatherings    0.020443
15     behavioral_outside_home    0.020189
16       behavioral_touch_face    0.019721
20        child_under_6_months    0.015229
13       behavioral_wash_hands    0.013860
12        behavioral_face_mask    0.010984
11    behavioral_antiviral_meds    0.008702
```

In [56]:
```
plt.figure(figsize=(10, 6))
plt.bar(importance_df['Feature'], importance_df['Importance'])
plt.xticks(rotation=90)
plt.xlabel('Feature')
plt.ylabel('Importance')
plt.title('Feature Importances')
plt.tight_layout()
plt.show()
```

Conclusion

The chart above explains the degree to which opinions, perceptions and other behavioral characteristics affect Vaccine uptake. The following top 4 features had the most effect on vaccine uptake

i) Respondent's opinion about risk of getting sick with seasonal flu without vaccine

ii) Age group

iii) Respondent's opinion about seasonal flu vaccine effectiveness

iv) Respondent's opinion about risk of getting sick with H1N1

Recommendations

-**Mass and grassroot sensitization**: Stakeholders should invest in mass and grassroot sensitization about H1N1 and the seasonal flu vaccine led by healthcare professionals. They should be educated on the vaccine efficacy and dangers of not getting it.

-**Healthcare Provider Recommendations**: Encouraging healthcare providers to actively recommend and offer H1N1 and seasonal vaccination to their patients can significantly impact vaccine uptake. The stakeholders can provide training and resources to healthcare professionals to increase their knowledge and confidence in recommending the vaccine.

-**Collaboration and Partnerships**: Collaborating with community organizations, schools, workplaces, and other stakeholders can amplify efforts to promote H1N1 vaccination. Engaging influencers, community leaders, and trusted voices within different communities can help build trust and increase acceptance of the vaccine