In [ ]:

```python
#Standard data analytical libraries
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import datetime as dt
import os, warnings, time, dmba
import scikitplot as skplt #pip install scikit-plot

#Data Mining Book Libraries
from dmba import liftChart, gainsChart,regressionSummary, classificationSummary, exh
from dmba import backward_elimination, forward_selection, stepwise_selection, adjust
from os.path import exists
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.metrics import f1_score,accuracy_score, roc_curve, auc,roc_auc_score,pl
#Classification
from sklearn import svm
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoo
from sklearn.linear_model import Perceptron, LogisticRegression,  LinearRegression, 
from sklearn.neighbors import KNeighborsClassifier, kneighbors_graph
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer

import imblearn
from imblearn.over_sampling import SMOTE

# Used to save keystrokes when wanting to print something. Now we can just use
# p("Hello") instead of print("Hello")
p = print
# import csv
# import re

#Change this value if you are not using o_desktop
computer = 'o_desktop'
#computer = 'other'
if (computer == 'o_desktop'):
    os.environ['NUMEXPR_MAX_THREADS'] = '24'
else:
    #default is 4 or 8
    os.environ['NUMEXPR_MAX_THREADS'] = '8'

# For future use.
# import threading
# import multiprocessing
```

Data Preparation (Wrangling and Pre-Processing)

To prepare the data for modeling, a number of preprocessing steps were taken. First...

In [ ]:

In [ ]:

```python
repo_directory = r"C:/ADS_599_Final/"
data_folder_directory = r"C:/ADS_599_Final/Data_Folder/"
df_features_file = "C:/ADS_599_Final/Data_Folder/training_set_features.csv"
df_labels_file = "C:/ADS_599_Final/Data_Folder/training_set_labels.csv"
df = pd.read_csv(df_features_file)
df_labels = pd.read_csv(df_labels_file)

#This is a very important step.
#We are combining our feature and target/label variables.
df = df.join(df_labels.set_index('respondent_id'), on='respondent_id')

df.head(5)
```

Out[ ]:

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

5 rows × 38 columns

Show unique values per column in preparation of data cleaning and transformation. Only needs to be done once since the files saved in the repo.

In [ ]:

```python
uniq_values_file = "unique_values_in_column.txt"
uniq_counts_file = 'unique_counts_in_column.txt'
if not exists(data_folder_directory + uniq_values_file):
    for col in df:
        #Unique values in each column
        with open(uniq_values_file, 'a') as f:
            f.write(str(df[col].unique()))
            f.write('\n')
        p(df[col].unique())

if not exists(data_folder_directory+ uniq_counts_file):
    for col in df:
        #Occurance of each unique row value
        #must set dropna=False
        with open(uniq_counts_file, 'a') as f:
            f.write( str(df[col].value_counts(dropna=False) ))
            f.write('\n')
        p(df[col].value_counts())
```

We show the number of null values per column below.

In [ ]:

```python
p("The null counts per column are: ")
p(df.isnull().sum())
p("Health insurance stands out at missing nearly 50%")

p("\nThe shape of the dataframe is: ")
p(df.shape)
```

```
The null counts per column are:
respondent_id                      0
h1n1_concern                      92
h1n1_knowledge                   116
behavioral_antiviral_meds         71
behavioral_avoidance             208
behavioral_face_mask              19
behavioral_wash_hands             42
behavioral_large_gatherings       87
behavioral_outside_home           82
behavioral_touch_face            128
doctor_recc_h1n1                2160
doctor_recc_seasonal            2160
chronic_med_condition            971
child_under_6_months             820
health_worker                    804
health_insurance               12274
opinion_h1n1_vacc_effective      391
opinion_h1n1_risk                388
opinion_h1n1_sick_from_vacc      395
opinion_seas_vacc_effective      462
opinion_seas_risk                514
opinion_seas_sick_from_vacc      537
age_group                          0
education                       1407
race                               0
sex                                0
income_poverty                  4423
marital_status                  1408
rent_or_own                     2042
employment_status               1463
hhs_geo_region                     0
census_msa                         0
household_adults                 249
household_children               249
employment_industry            13330
employment_occupation          13470
h1n1_vaccine                       0
seasonal_vaccine                   0
dtype: int64
Health insurance stands out at missing nearly 50%

The shape of the dataframe is:
(26707, 38)
```

Drop Uneeded Column

In [ ]:

```python
# respondent_id are all unique so its irrelevant now that we merged.
df_temp = df.drop(columns=["respondent_id"],inplace=False)
# About 50 percent of health_insurance is missing.
# TODO We should try models with that drop the health_insurance column.
#df_train = df.drop(columns=["health_insurance", "respondent_id"],inplace=False)
p("We are replacing NaN with no_response for categories and -1 for numbers.")
df_train = df_temp
```

```
We are replacing NaN with no_response for categories and -1 for numbers.
```

Replace NaNs with a new value. A person refusing to answer a question could be significant.
Note: for imputations we will have to put back the NaNs

In [ ]:

```python
# Categories
df_train['h1n1_concern'] = df_train['h1n1_concern'].fillna(-1)
df_train['h1n1_knowledge'] = df_train['h1n1_knowledge'].fillna(-1)
df_train['behavioral_antiviral_meds'] = df_train['behavioral_antiviral_meds'].fillna
df_train['behavioral_avoidance'] = df_train['behavioral_avoidance'].fillna(-1)
df_train['behavioral_face_mask'] = df_train['behavioral_face_mask'].fillna(-1)
df_train['behavioral_large_gatherings'] = df_train['behavioral_large_gatherings'].fi
df_train['behavioral_outside_home'] = df_train['behavioral_outside_home'].fillna(-1)
df_train['behavioral_wash_hands'] = df_train['behavioral_wash_hands'].fillna(-1)
df_train['behavioral_touch_face'] = df_train['behavioral_touch_face'].fillna(-1)
df_train['doctor_recc_h1n1'] = df_train['doctor_recc_h1n1'].fillna(-1)
df_train['doctor_recc_seasonal'] = df_train['doctor_recc_seasonal'].fillna(-1)
df_train['chronic_med_condition'] = df_train['chronic_med_condition'].fillna(-1)
df_train['child_under_6_months'] = df_train['child_under_6_months'].fillna(-1)
df_train['health_worker'] = df_train['health_worker'].fillna(-1)
df_train['health_insurance'] = df_train['health_insurance'].fillna(-1)
df_train['opinion_h1n1_vacc_effective'] = df_train['opinion_h1n1_vacc_effective'].fi
df_train['opinion_h1n1_sick_from_vacc'] = df_train['opinion_h1n1_sick_from_vacc'].fi
df_train['opinion_h1n1_risk'] = df_train['opinion_h1n1_risk'].fillna(-1)
df_train['opinion_seas_vacc_effective'] = df_train['opinion_seas_vacc_effective'].fi
df_train['opinion_seas_risk'] = df_train['opinion_seas_risk'].fillna(-1)
df_train['opinion_seas_sick_from_vacc'] = df_train['opinion_seas_sick_from_vacc'].fi
df_train['household_adults'] = df_train['household_adults'].fillna(-1)
df_train['household_children'] = df_train['household_children'].fillna(-1)

# Numbers
df_train['age_group'] = df_train['age_group'].fillna("no_response")
df_train['education'] = df_train['education'].fillna("no_response")
df_train['race'] = df_train['race'].fillna("no_response")
df_train['income_poverty'] = df_train['income_poverty'].fillna("no_response")
df_train['marital_status'] = df_train['marital_status'].fillna("no_response")
df_train['rent_or_own'] = df_train['rent_or_own'].fillna("no_response")
df_train['employment_status'] = df_train['employment_status'].fillna("no_response")
df_train['employment_occupation'] = df_train['employment_occupation'].fillna("no_resp
df_train['employment_industry'] = df_train['employment_industry'].fillna("no_respons
```

Gotta convert those categories to numbers. Category encoding known in the R world as as.factor-ing.

In [ ]:

```python
encoding_selection = "label" # options "label" "onehot"
if encoding_selection == "label":
    df_train = df_train
    # Encode labels the below is equivalent to df_train['hhs_geo_region']= label_enc
    df_train["hhs_geo_region"] = df_train["hhs_geo_region"].astype('category')
    df_train["hhs_geo_region"] = df_train["hhs_geo_region"].cat.codes
    df_train["census_msa"] = df_train["census_msa"].astype('category')
    df_train["census_msa"] = df_train["census_msa"].cat.codes
    df_train["employment_industry"] = df_train["employment_industry"].astype('catego
    df_train["employment_industry"] = df_train["employment_industry"].cat.codes
    df_train["employment_occupation"] = df_train["employment_occupation"].astype('ca
    df_train["employment_occupation"] = df_train["employment_occupation"].cat.codes
    df_train["employment_status"] = df_train["employment_status"].astype('category')
    df_train["employment_status"] = df_train["employment_status"].cat.codes
    df_train["rent_or_own"] = df_train["rent_or_own"].astype('category')
    df_train["rent_or_own"] = df_train["rent_or_own"].cat.codes
    df_train["marital_status"] = df_train["marital_status"].astype('category')
    df_train["marital_status"] = df_train["marital_status"].cat.codes
    df_train["income_poverty"] = df_train["income_poverty"].astype('category')
    df_train["income_poverty"] = df_train["income_poverty"].cat.codes
    df_train["race"] = df_train["race"].astype('category')
    df_train["race"] = df_train["race"].cat.codes
    df_train["education"] = df_train["education"].astype('category')
    df_train["education"] = df_train["education"].cat.codes
    df_train["age_group"] = df_train["age_group"].astype('category')
    df_train["age_group"] = df_train["age_group"].cat.codes
    df_train["sex"] = df_train["sex"].astype('category')
    df_train["sex"] = df_train["sex"].cat.codes
elif encoding_selection == "onehot":
    # Right now are are using label encoding.
    # TODO We could try other encoding methods and see if the results improve
    p("We can try one hot encoder here.")

p("After encoding the null counts per column are: ")
p(df_train.isnull().sum())
```

```
After encoding the null counts per column are:
h1n1_concern                   0
h1n1_knowledge                 0
behavioral_antiviral_meds      0
behavioral_avoidance           0
behavioral_face_mask           0
behavioral_wash_hands          0
behavioral_large_gatherings    0
behavioral_outside_home        0
behavioral_touch_face          0
doctor_recc_h1n1               0
doctor_recc_seasonal           0
chronic_med_condition          0
child_under_6_months           0
health_worker                  0
health_insurance               0
opinion_h1n1_vacc_effective    0
opinion_h1n1_risk              0
opinion_h1n1_sick_from_vacc    0
opinion_seas_vacc_effective    0
opinion_seas_risk              0
opinion_seas_sick_from_vacc    0
age_group                      0
education                      0
race                           0
sex                            0
income_poverty                 0
marital_status                 0
rent_or_own                    0
employment_status              0
hhs_geo_region                 0
census_msa                     0
household_adults               0
household_children             0
employment_industry            0
employment_occupation          0
h1n1_vaccine                   0
seasonal_vaccine               0
dtype: int64
```

Handling the Nulls.

In [ ]:

```python
handling_nulls = "iterative" # options "averages" "iterative" "dropall" "noresponse"
if handling_nulls == "iterative":
    #Need to add back the NaN for the imputations.
    df_train.replace(-1, np.nan) #np.nan or pd.NA ? they behave differently
    df_train.replace("no_response", np.nan) #np.nan or pd.NA ? they behave different

    # SMOTE Sampling
    temp_columns = df_train.columns
    imp = IterativeImputer(max_iter=10, random_state=0)
    imp.fit(df_train)
    df_train = pd.DataFrame(data=imp.transform(df_train))
    df_train.columns = temp_columns
    df_train
elif handling_nulls == "averages":
    df_train = df_train
    #Need to add back the NaN for the imputations.
    df_train.replace(-1, np.nan) #np.nan or pd.NA ? they behave differently
    df_train.replace("no_response", np.nan) #np.nan or pd.NA ? they behave different

    # Technically we shouldn't use mean when there is a skew, i.e. with salaries
    df_train.fillna(df_train.mean())
    # TODO We use median or mode if the column data is skewed
    #newdf_train.fillna(df_train.median())
    #newdf_train.fillna(df_train.mode())
elif handling_nulls == "dropall":
    #Need to add back the NaN for the imputations.
    df_train.replace(-1, np.nan) #np.nan or pd.NA ? they behave differently
    df_train.replace("no_response", np.nan) #np.nan or pd.NA ? they behave different
    # See how it is if we drop the NaNs
    df_train = df_train.dropna(inplace=False) #This should be replace with imputatio
else: #elif handling_nulls == "noresponse":
    p("If we are here we keep the NaN replacements done in the previous step.")
    #This would replace all NaNs with the same value. We don't this.
    #df.fillna("no_response")

# p("There should be no nulls now: ")
# p(df_train.isnull().sum())
#df_train
```

Normalization Standardization vs Scalarize

In [ ]:

```python
# TODO Which columns to normalize, standardize, and scalarize?

#Scale the values (removes means and scale to unit variance)
scalar = MinMaxScaler()
#df_train['columnName'] = scalar.fit_transform(df_train['columnName'].values.reshape
```

Exploratory Data Analysis

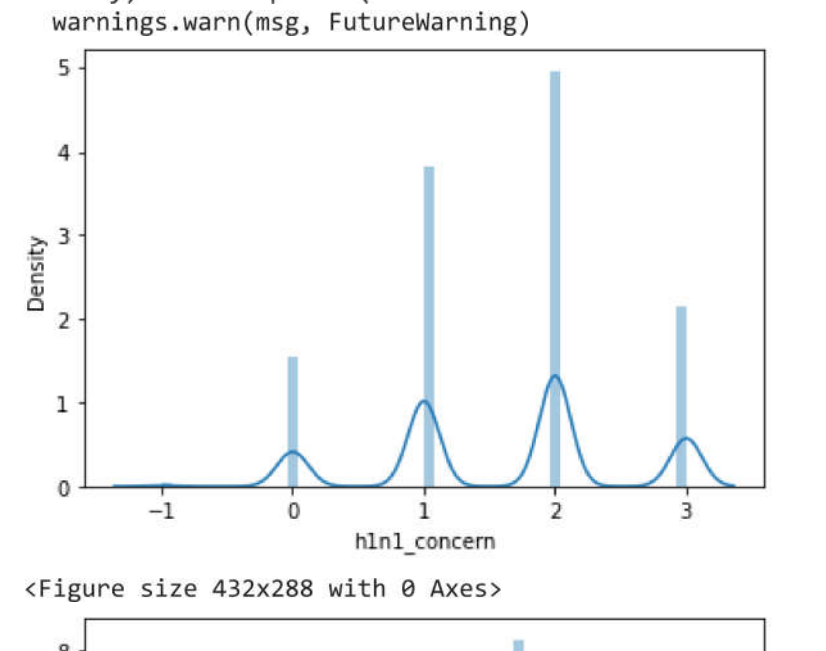
In order to better understand the variables, both numeric and categorical, the plots below were produced. Bars for the frequency of a category or numeric range are shown, along with a density curve. An interesting note is the bimodality ..... and columnX demonstrates a right-skewed distribution.

A correlation heatmap of the numeric variables is also visualized. ...strong relationship besides.... The correlation between all variables was then calculated along with the variance of each variable.

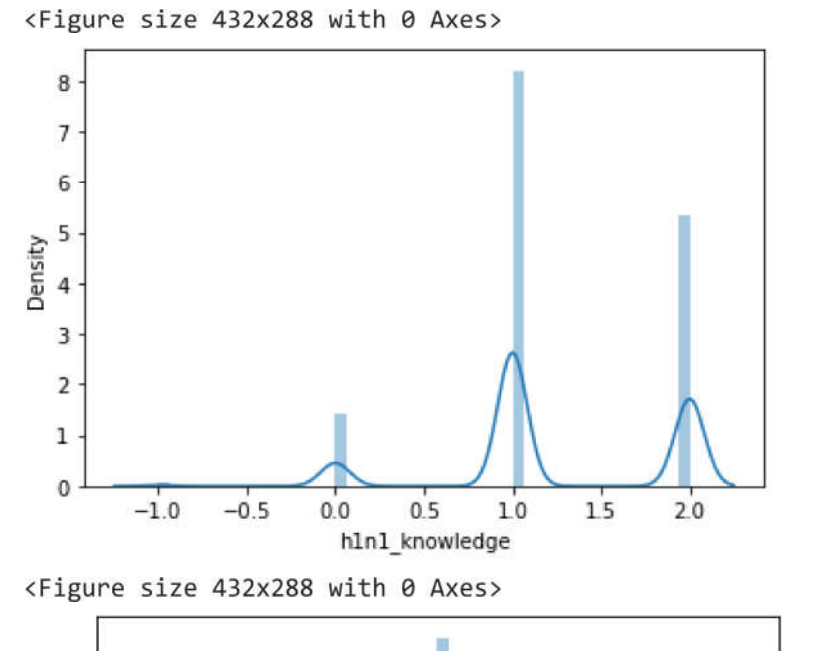
In [ ]:

```python
# create distplots
for column in df_train.columns:
    plt.figure()
    sns.distplot(df_train[column])
    plt.figure()
```

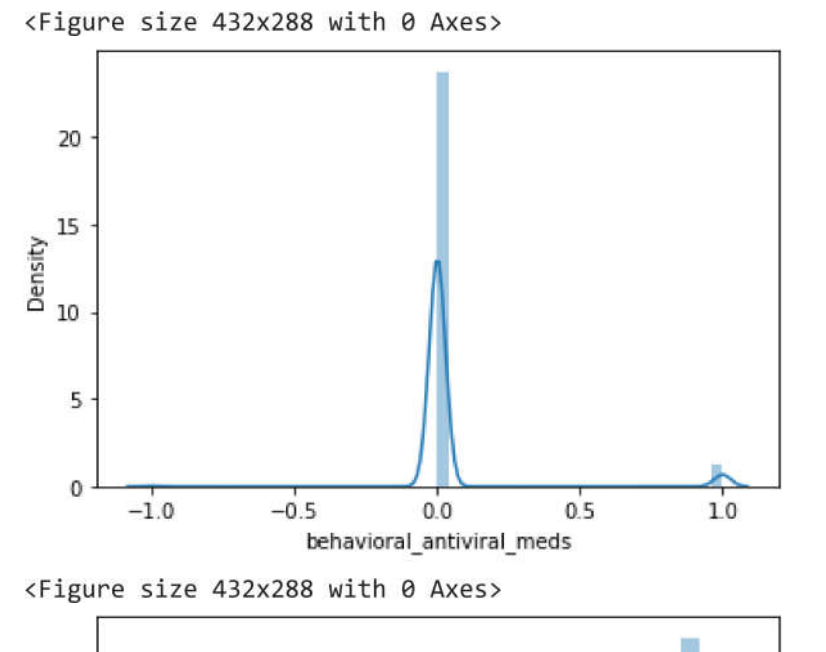
```
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\elfek\AppData\Local\Temp\ipykernel_14276\3372926779.py:3: RuntimeWarning: Mo
re than 20 figures have been opened. Figures created through the pyplot interface (`m
atplotlib.pyplot.figure`) are retained until explicitly closed and may consume too mu
ch memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  plt.figure()
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



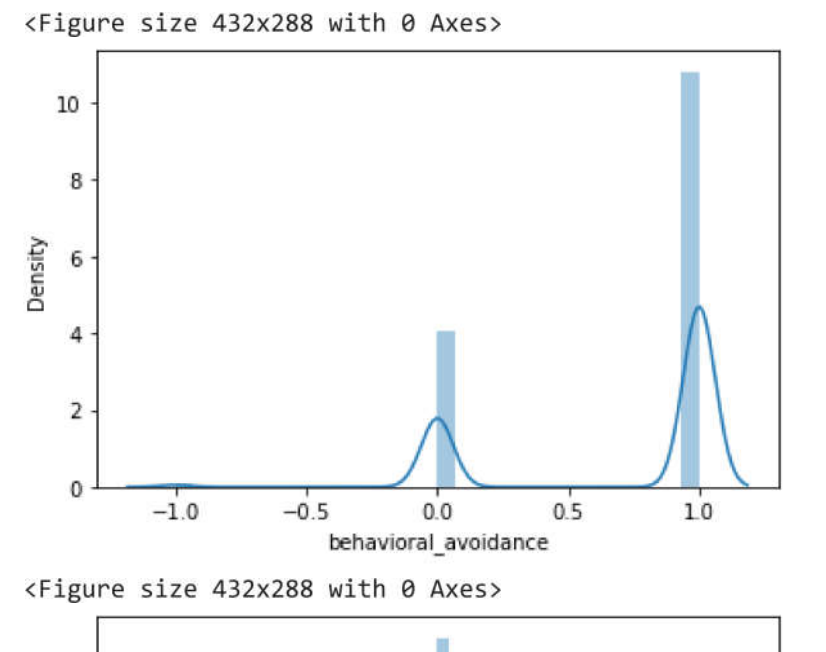
<Figure size 432x288 with 0 Axes>



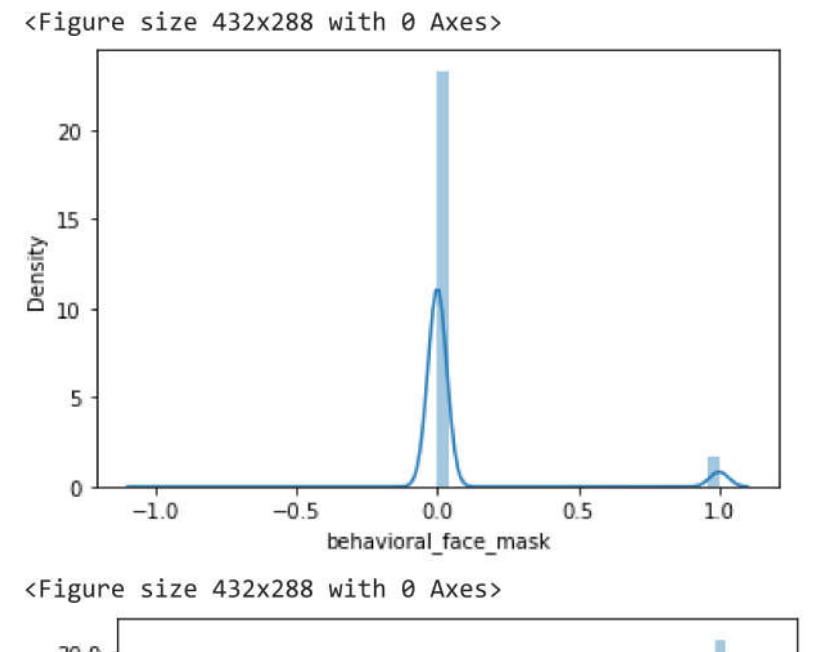
<Figure size 432x288 with 0 Axes>



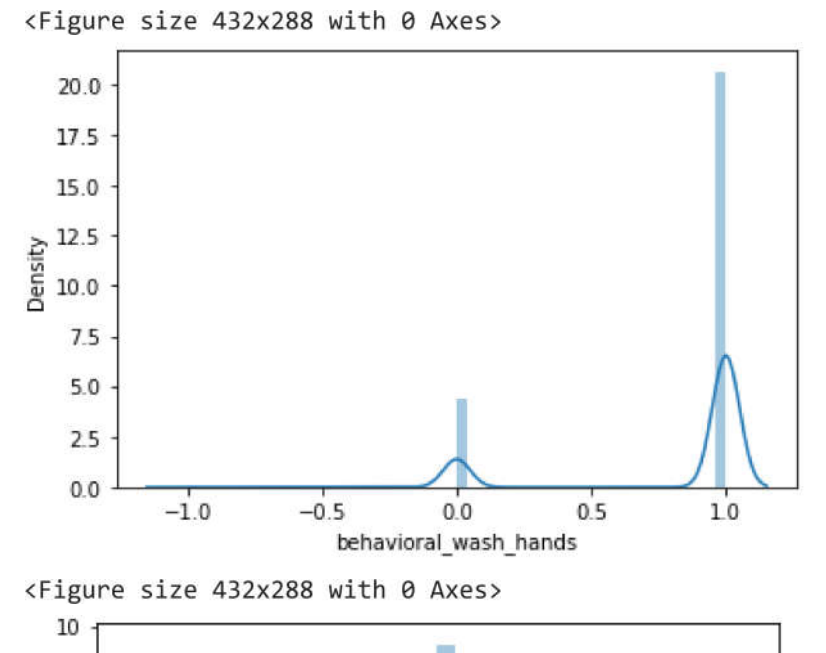
<Figure size 432x288 with 0 Axes>



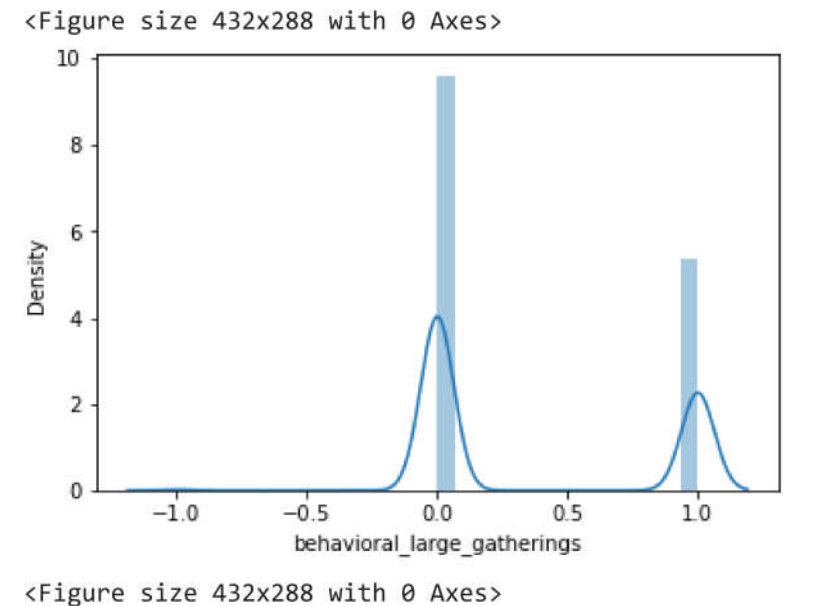
<Figure size 432x288 with 0 Axes>



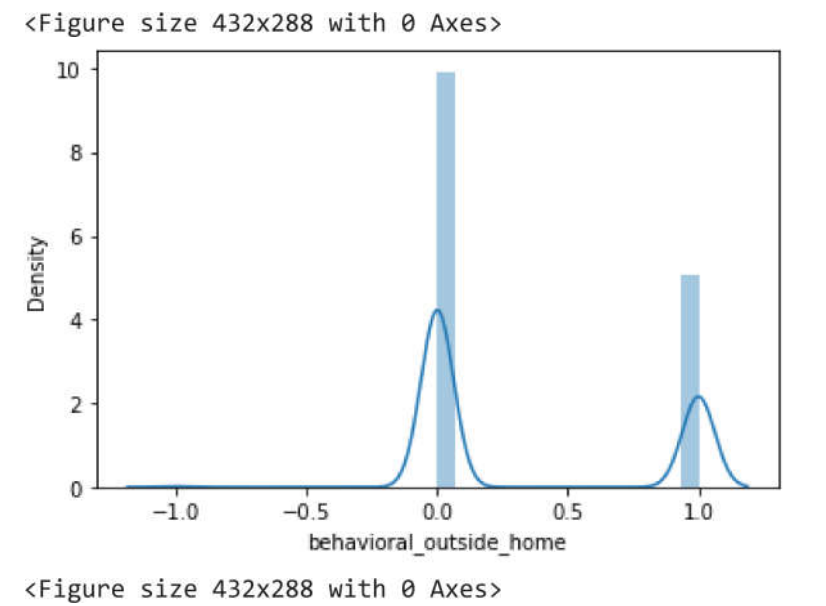
<Figure size 432x288 with 0 Axes>



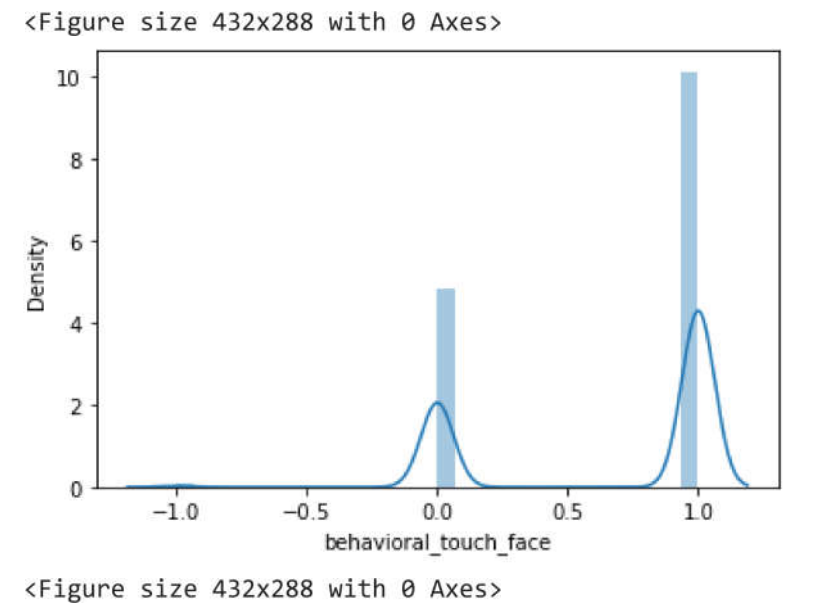
<Figure size 432x288 with 0 Axes>



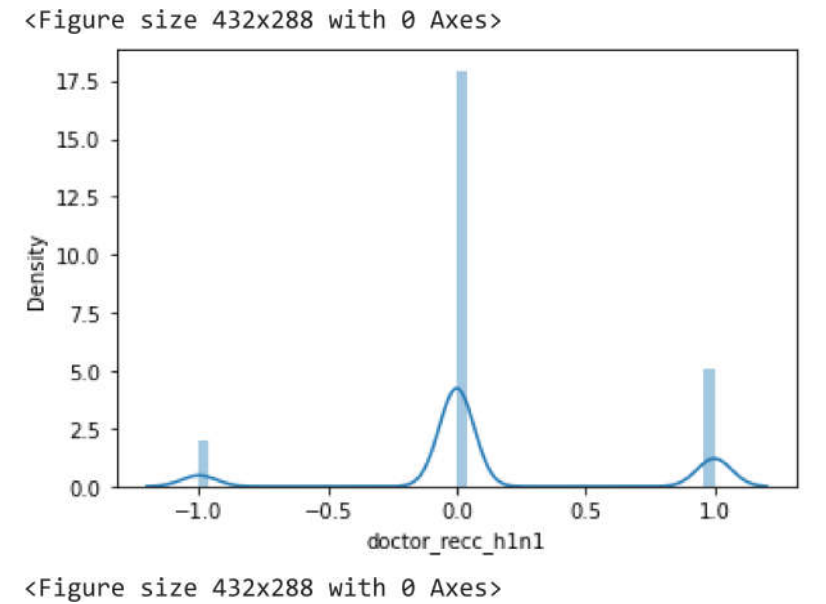
<Figure size 432x288 with 0 Axes>



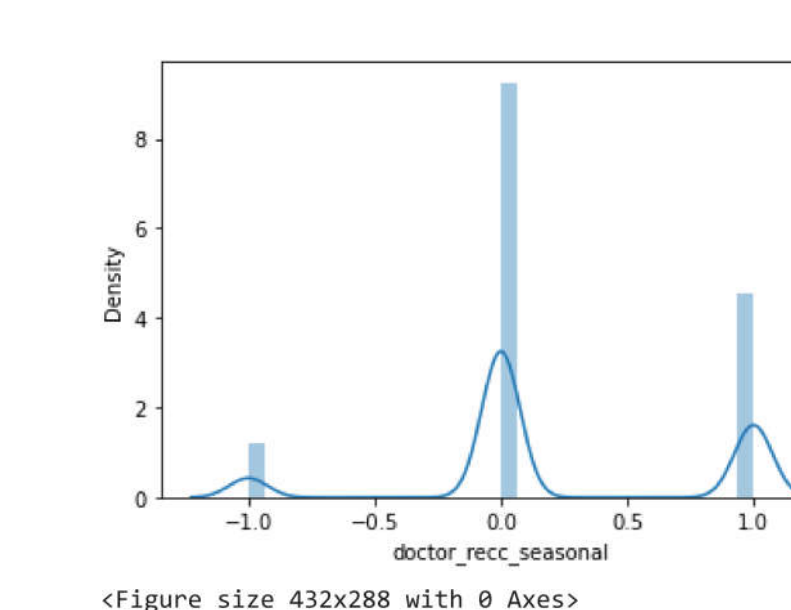
<Figure size 432x288 with 0 Axes>



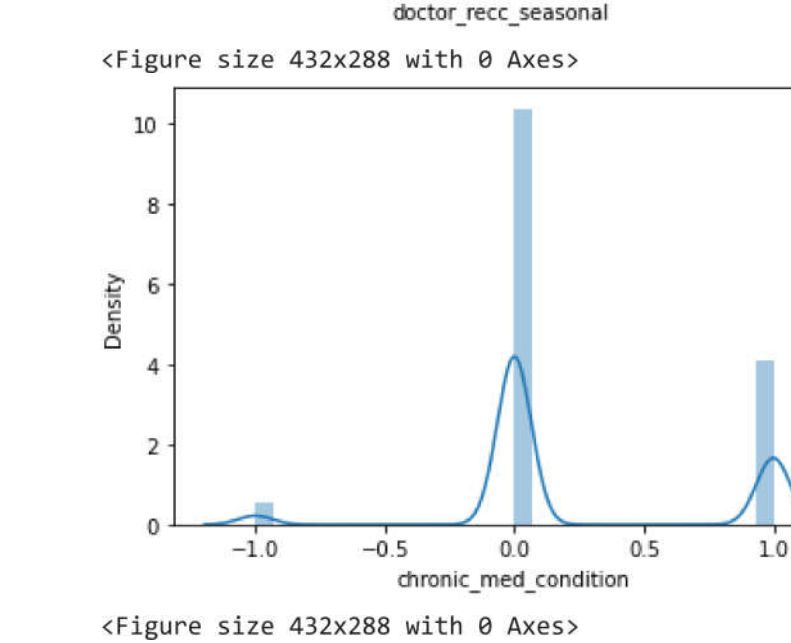
<Figure size 432x288 with 0 Axes>



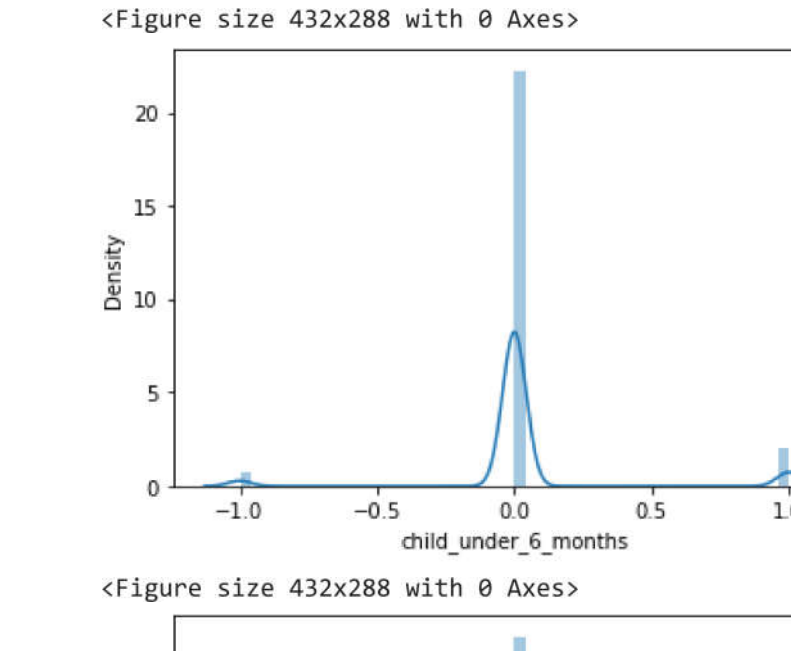
<Figure size 432x288 with 0 Axes>



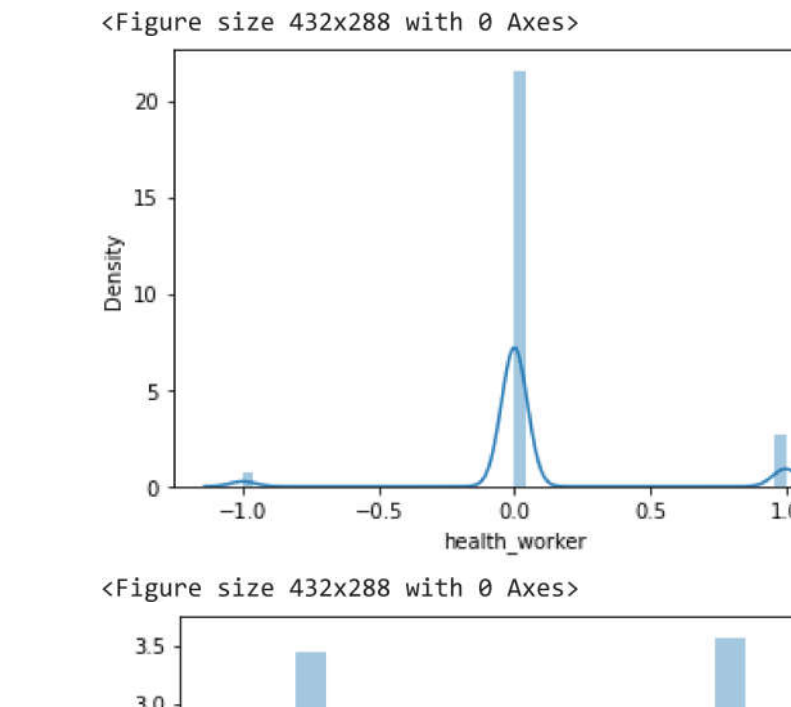
<Figure size 432x288 with 0 Axes>



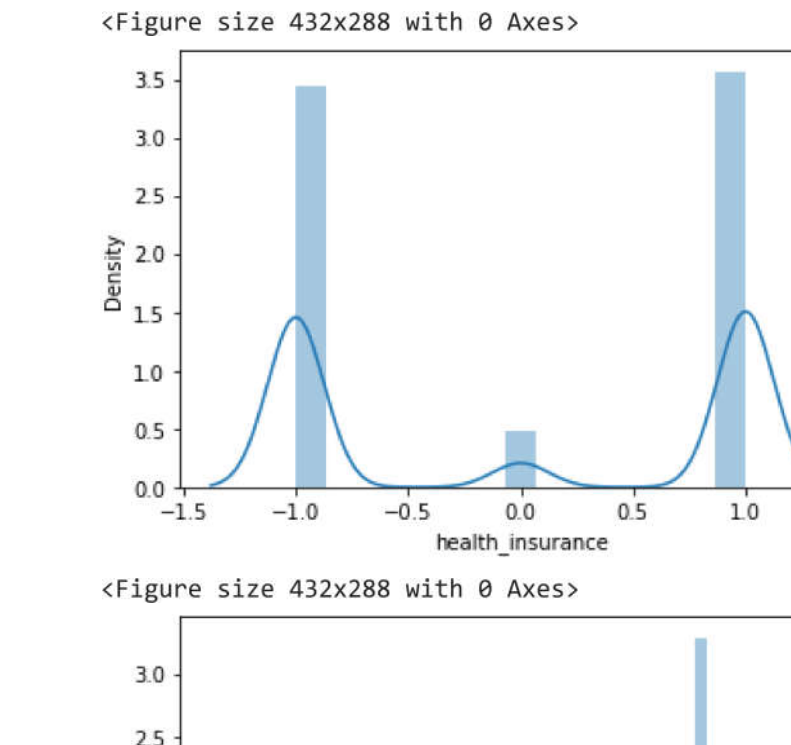
<Figure size 432x288 with 0 Axes>



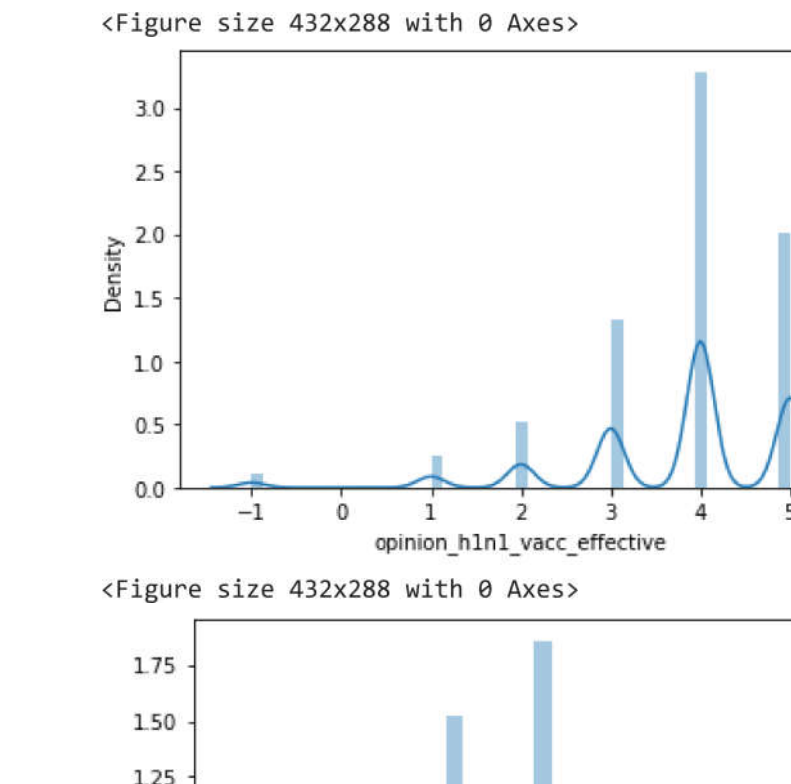
<Figure size 432x288 with 0 Axes>



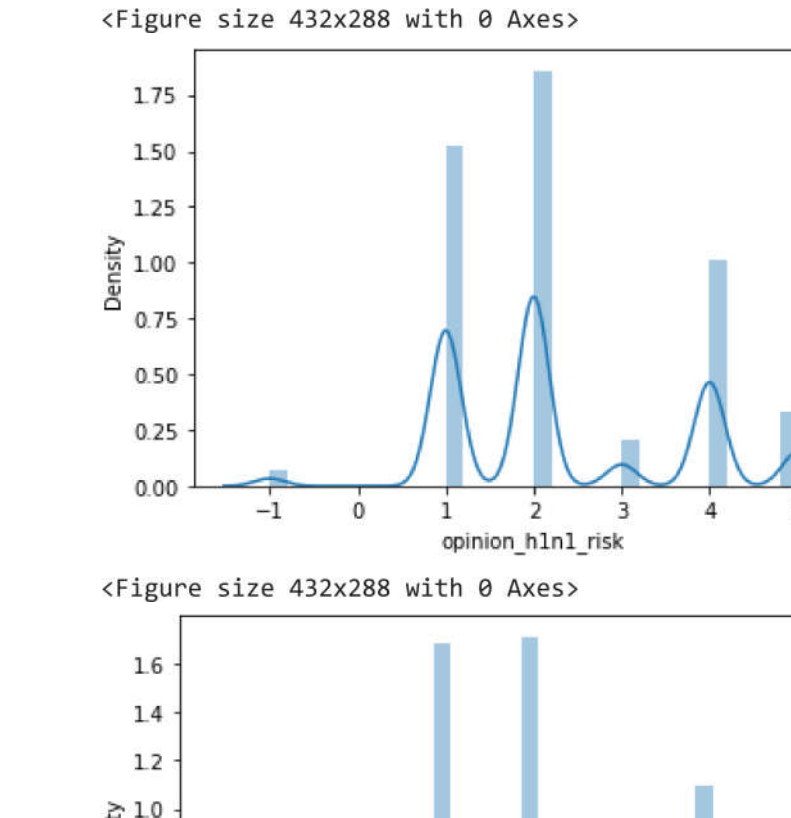
<Figure size 432x288 with 0 Axes>



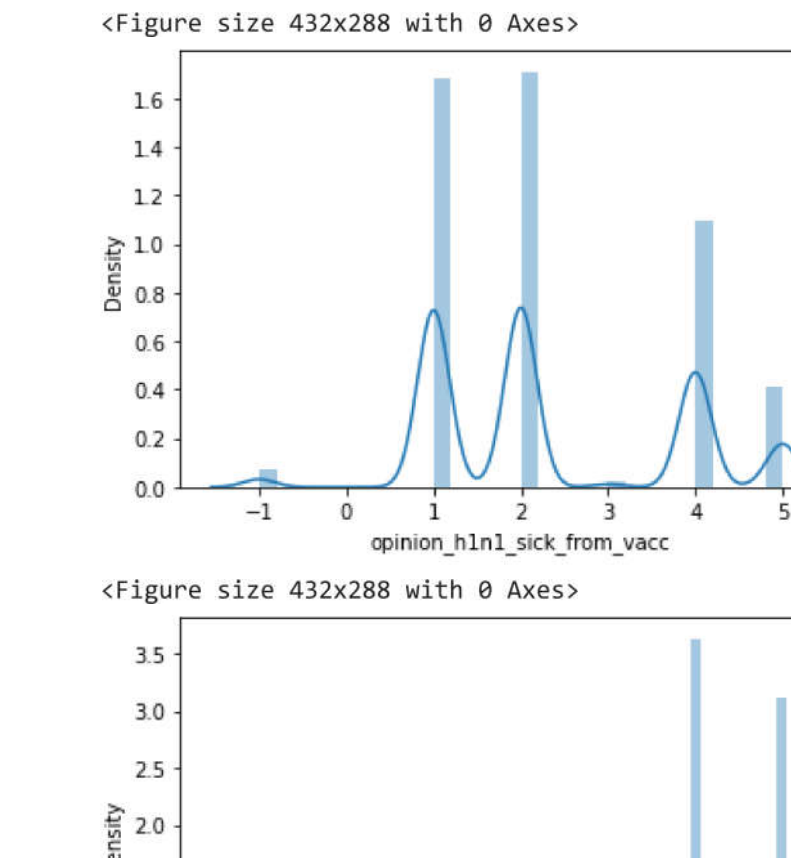
<Figure size 432x288 with 0 Axes>



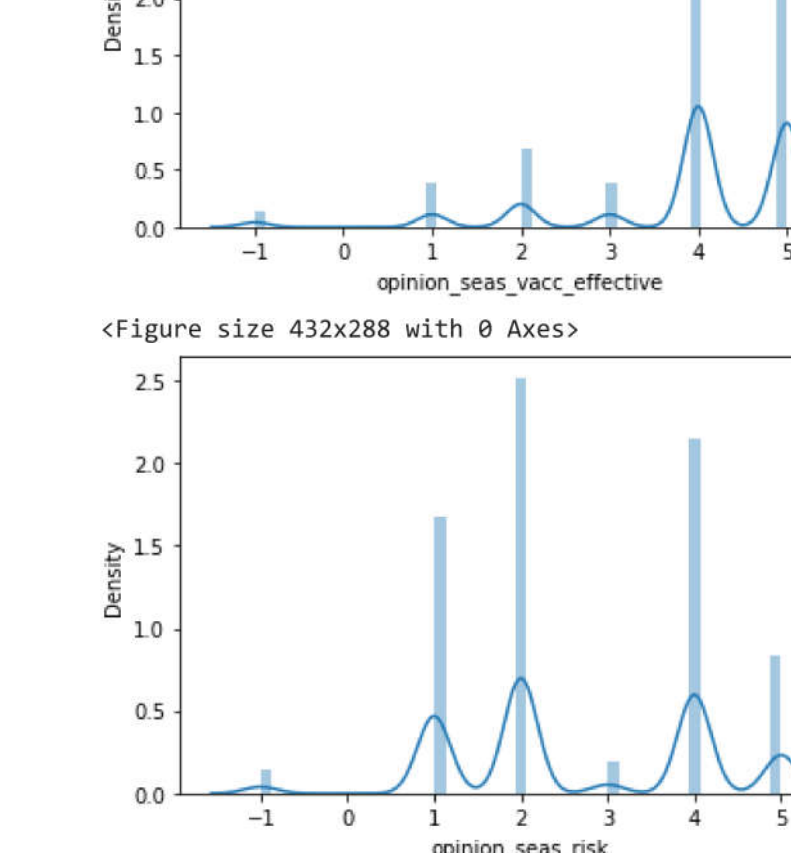
<Figure size 432x288 with 0 Axes>



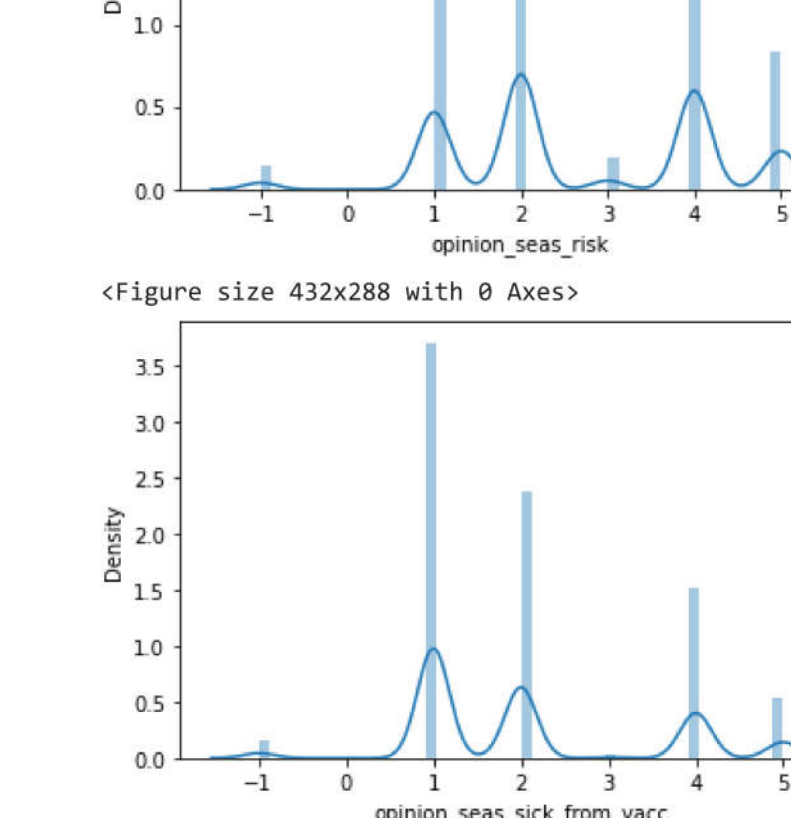
<Figure size 432x288 with 0 Axes>



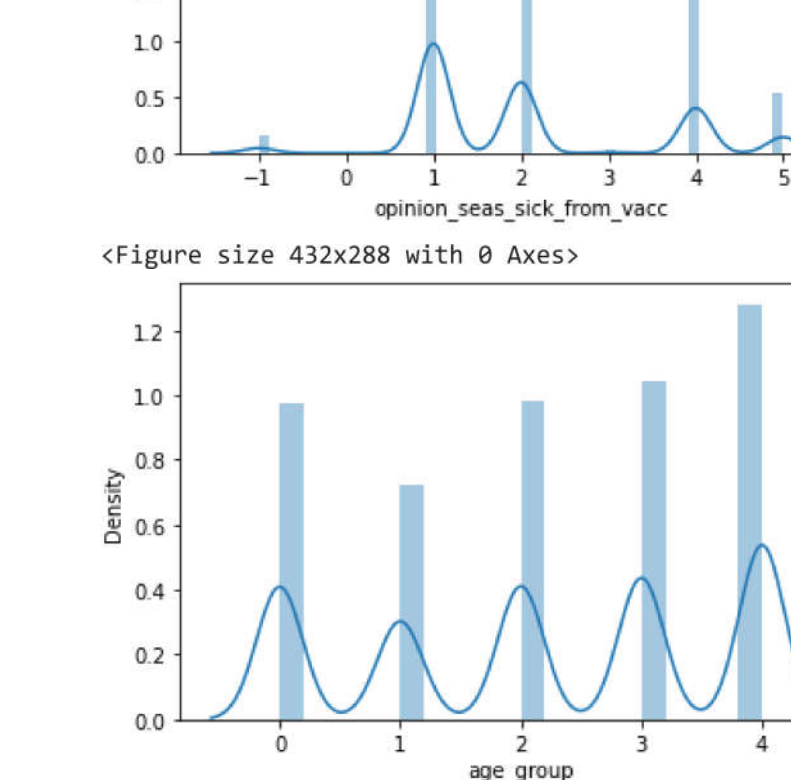
<Figure size 432x288 with 0 Axes>



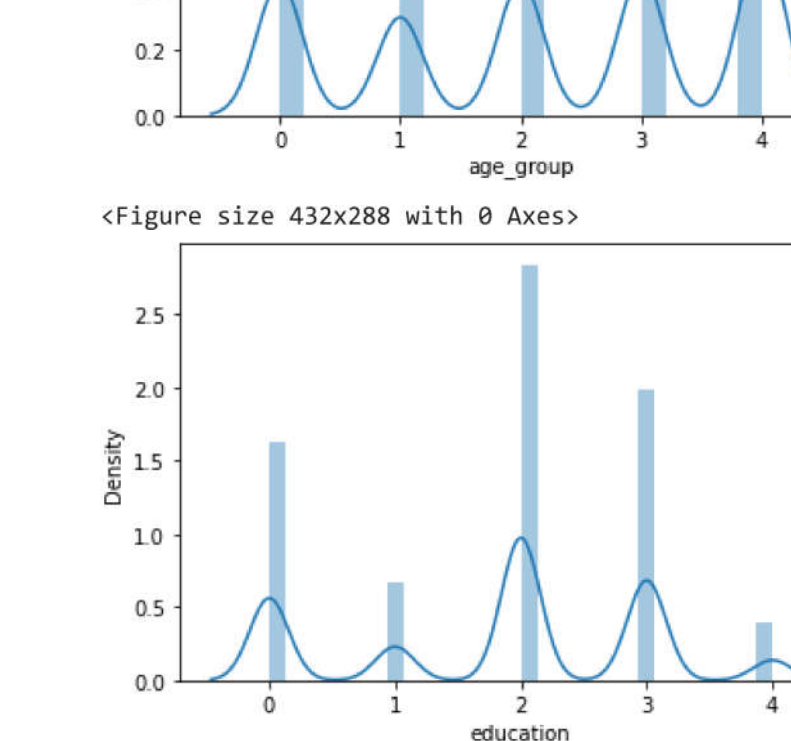
<Figure size 432x288 with 0 Axes>



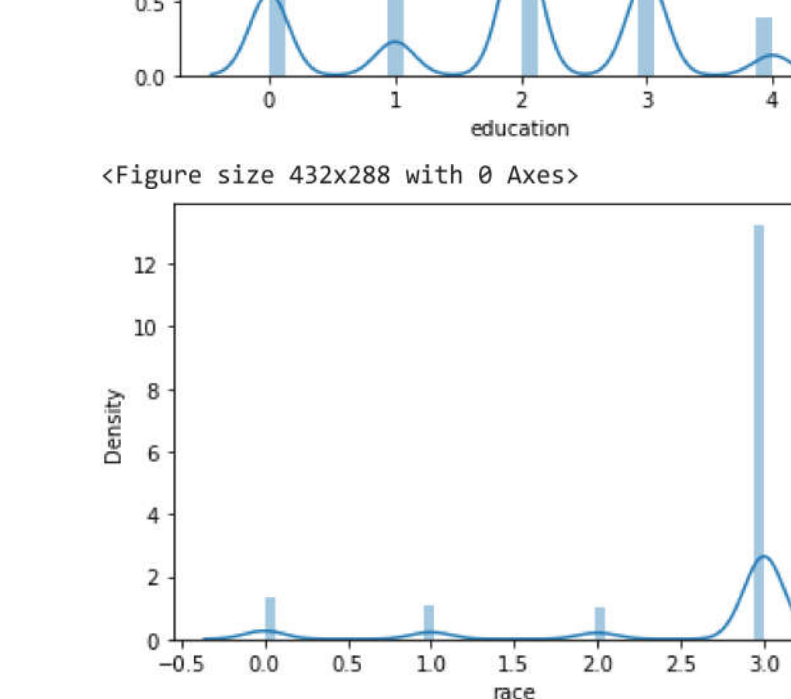
<Figure size 432x288 with 0 Axes>



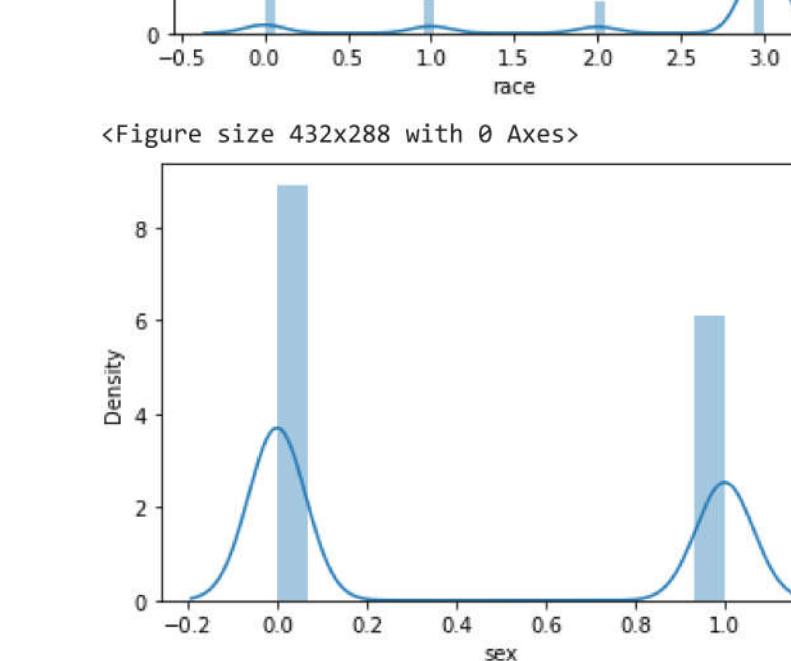
<Figure size 432x288 with 0 Axes>



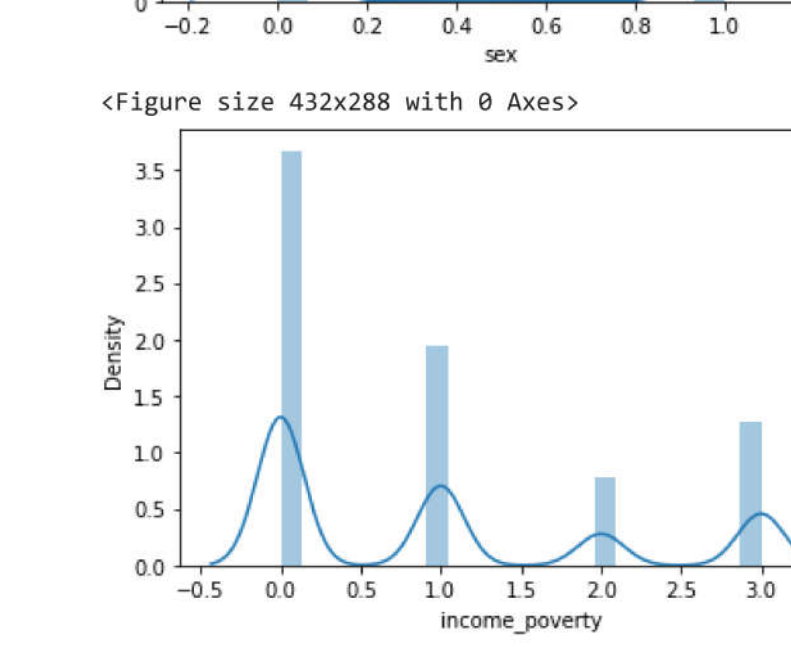
<Figure size 432x288 with 0 Axes>



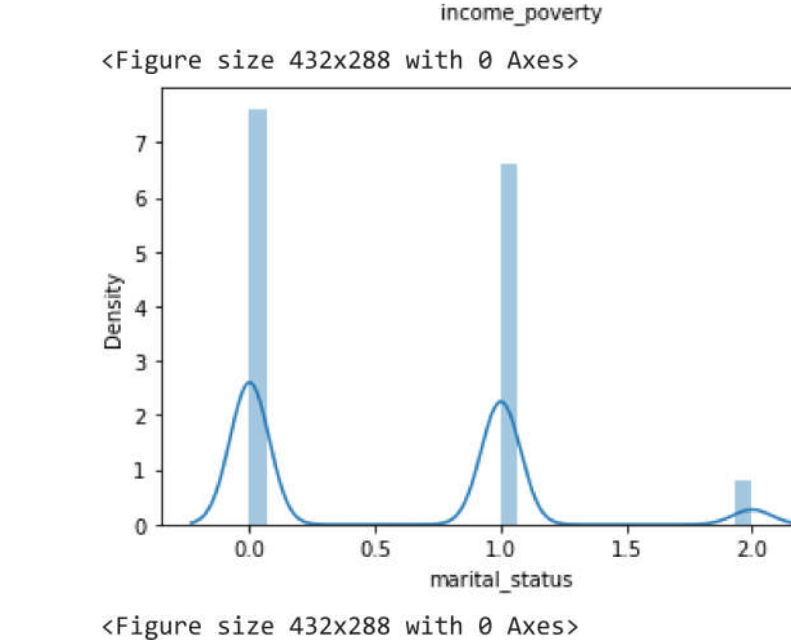
<Figure size 432x288 with 0 Axes>



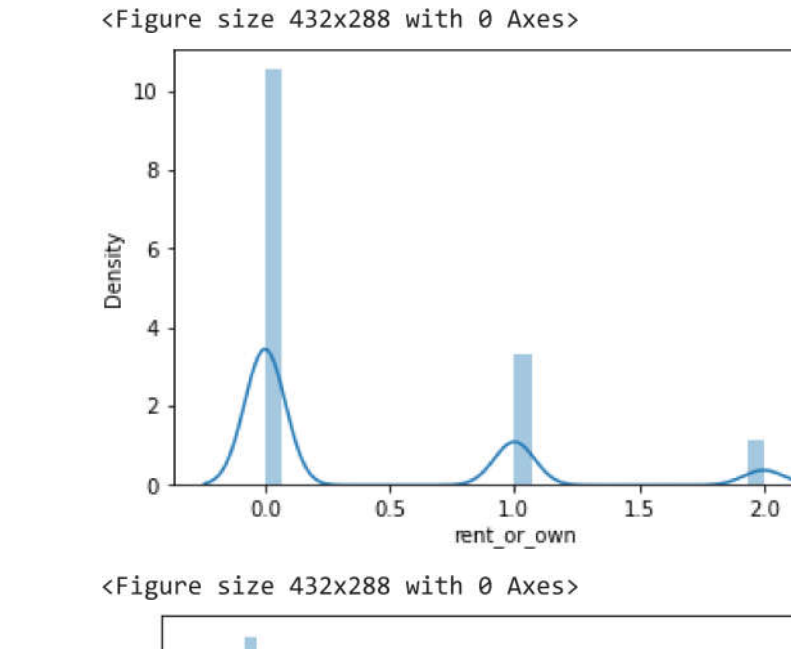
<Figure size 432x288 with 0 Axes>



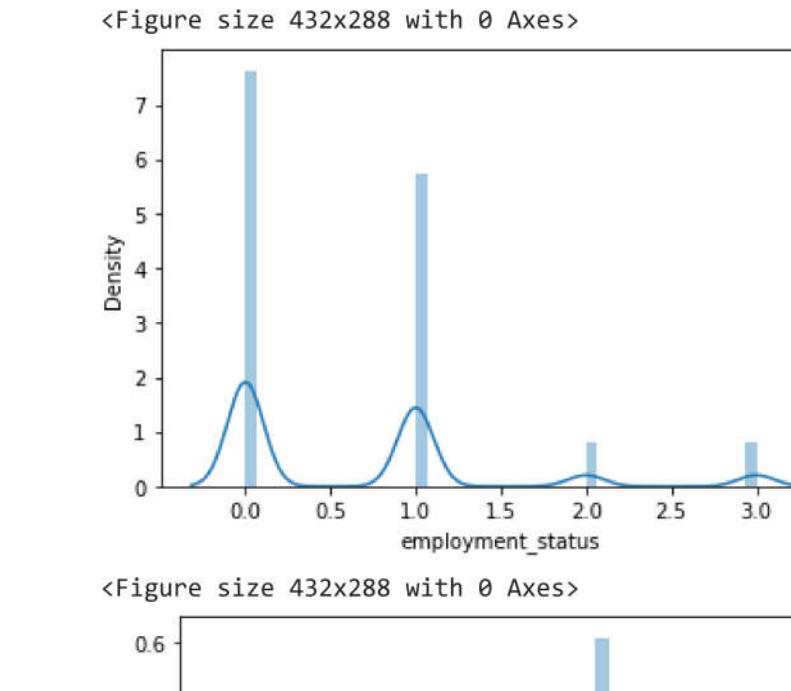
<Figure size 432x288 with 0 Axes>



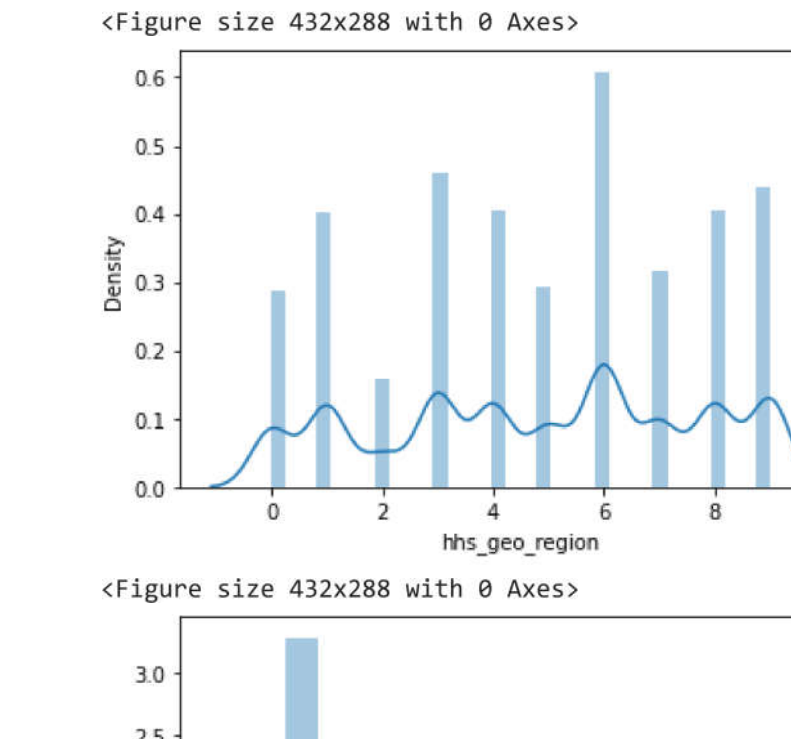
<Figure size 432x288 with 0 Axes>



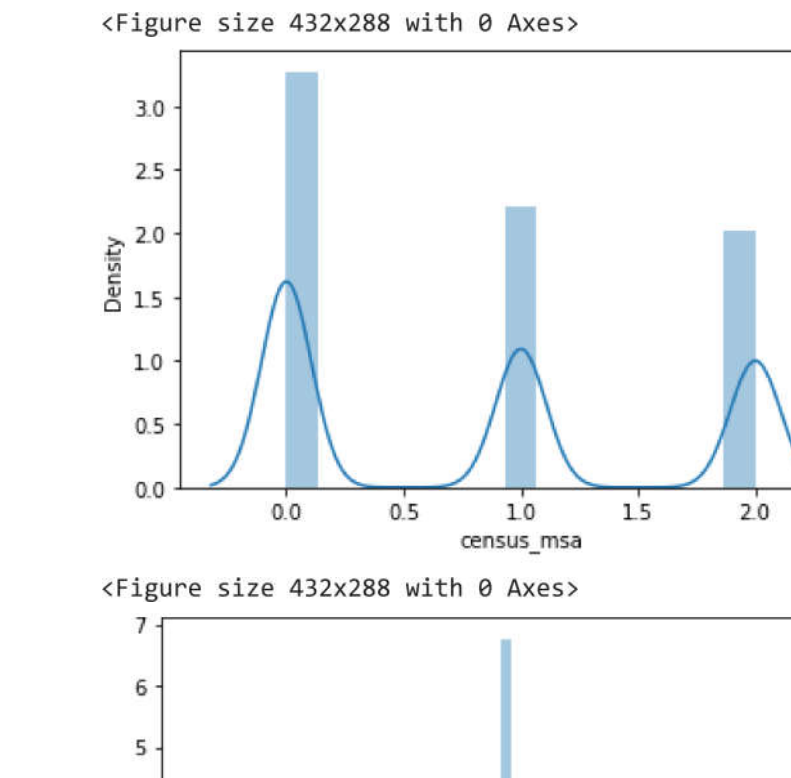
<Figure size 432x288 with 0 Axes>



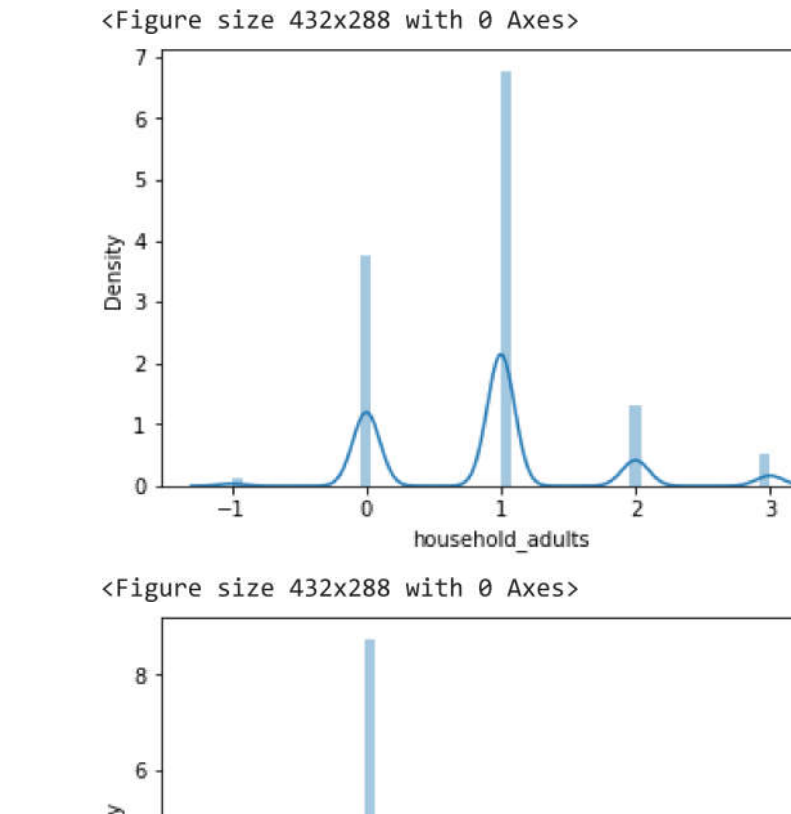
<Figure size 432x288 with 0 Axes>



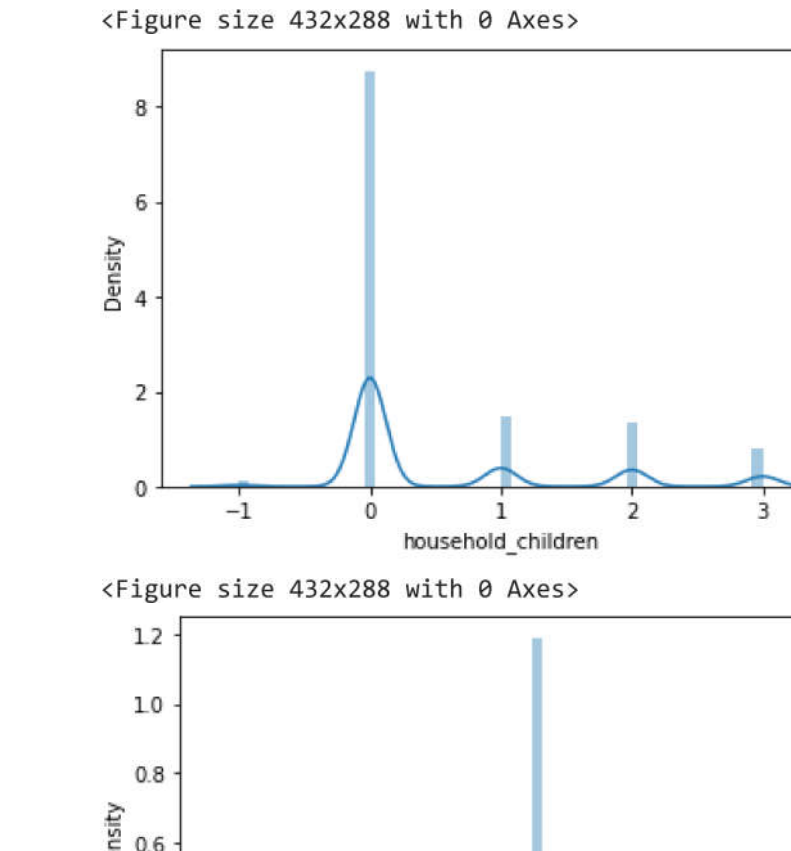
<Figure size 432x288 with 0 Axes>



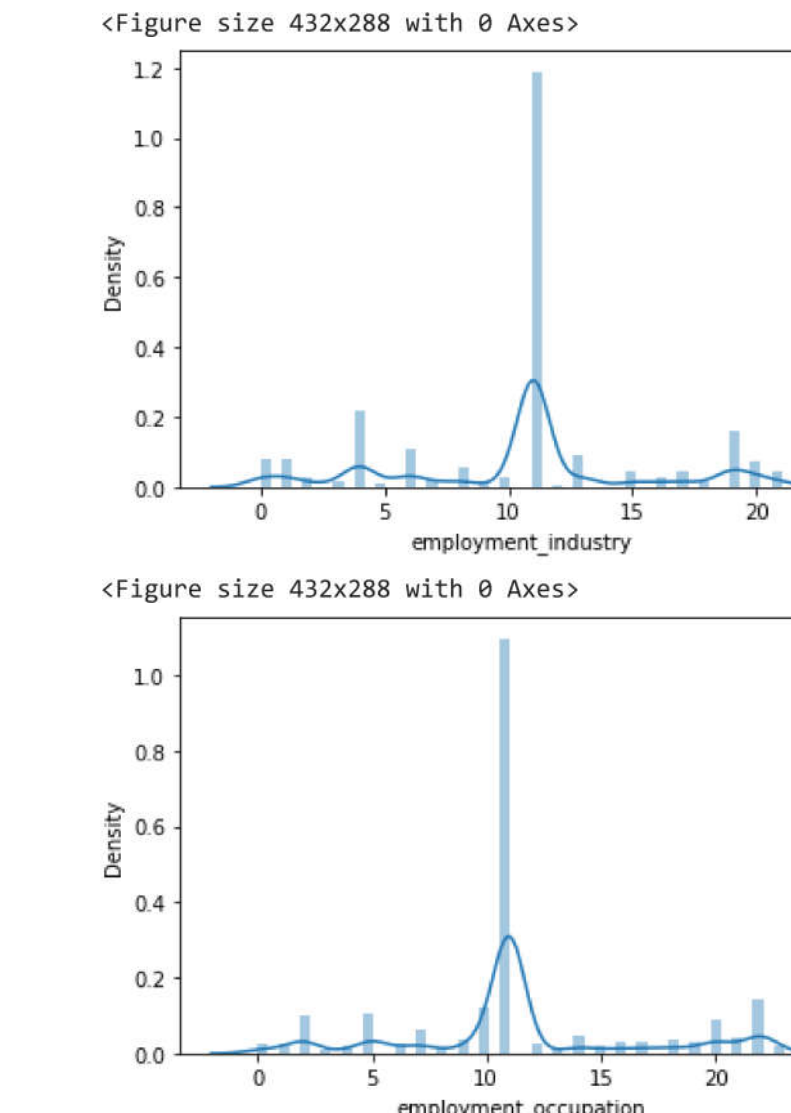
<Figure size 432x288 with 0 Axes>



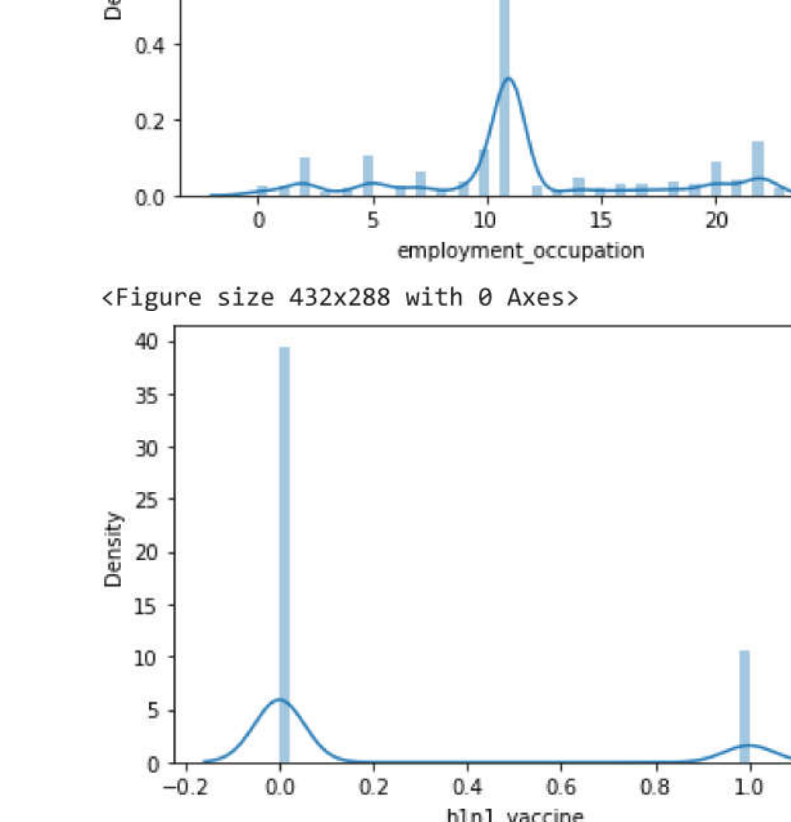
<Figure size 432x288 with 0 Axes>



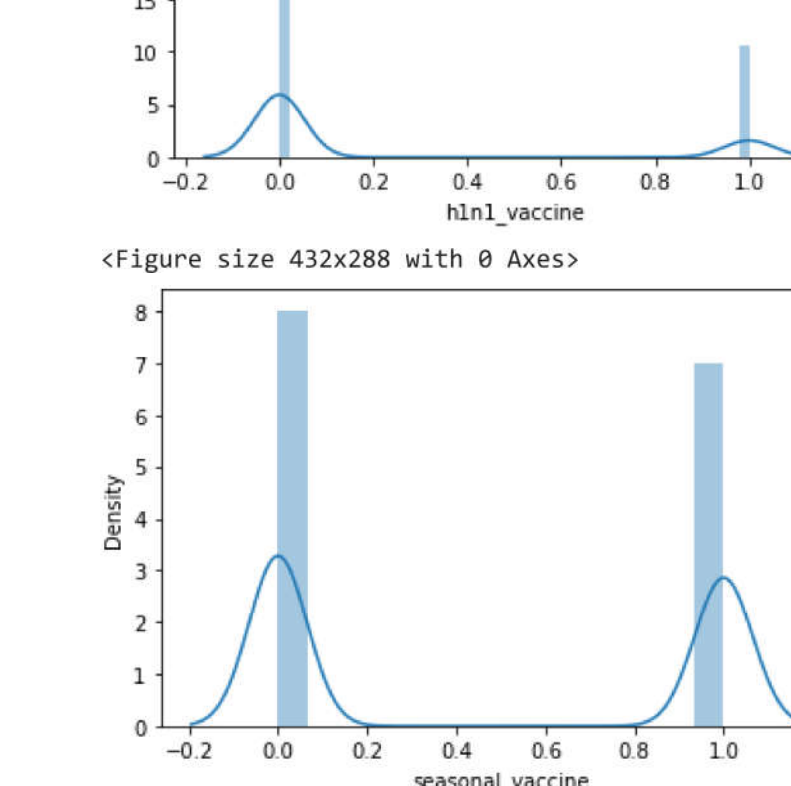
<Figure size 432x288 with 0 Axes>



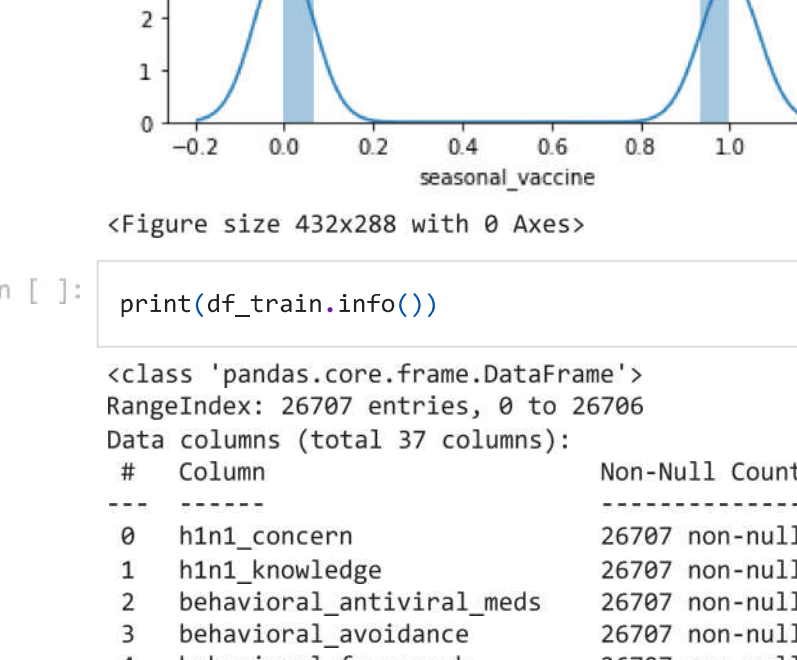
<Figure size 432x288 with 0 Axes>



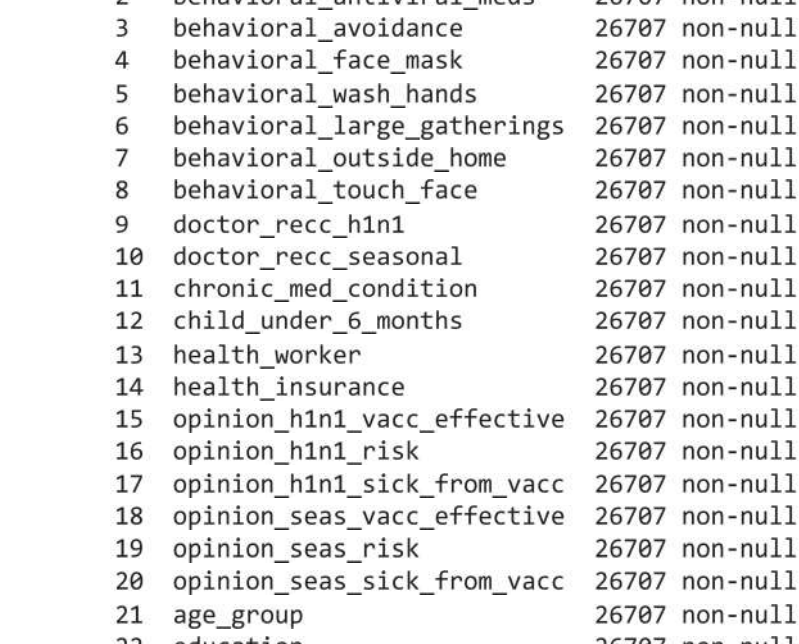
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>

In [ ]:

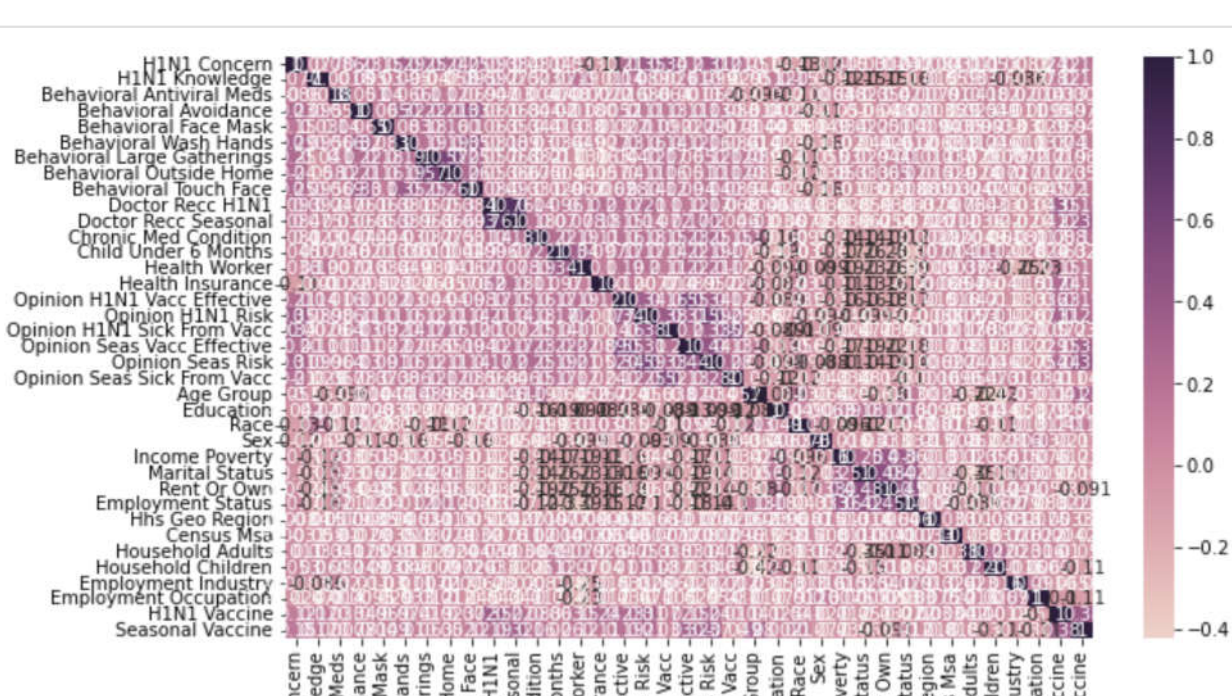
```
print(df_train.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26707 entries, 0 to 26706
Data columns (total 37 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  health_insurance             26707 non-null  float64
 15  opinion_h1n1_vacc_effective  26707 non-null  float64
 16  opinion_h1n1_risk            26707 non-null  float64
 17  opinion_h1n1_sick_from_vacc  26707 non-null  float64
 18  opinion_seas_vacc_effective  26707 non-null  float64
 19  opinion_seas_risk            26707 non-null  float64
 20  opinion_seas_sick_from_vacc  26707 non-null  float64
 21  age_group                    26707 non-null  float64
 22  education                    26707 non-null  float64
 23  race                         26707 non-null  float64
 24  sex                          26707 non-null  float64
 25  income_poverty               26707 non-null  float64
 26  marital_status               26707 non-null  float64
 27  rent_or_own                  26707 non-null  float64
 28  employment_status            26707 non-null  float64
 29  hhs_geo_region               26707 non-null  float64
 30  census_msa                   26707 non-null  float64
 31  household_adults             26707 non-null  float64
 32  household_children           26707 non-null  float64
 33  employment_industry          26707 non-null  float64
 34  employment_occupation        26707 non-null  float64
 35  h1n1_vaccine                 26707 non-null  float64
 36  seasonal_vaccine             26707 non-null  float64
dtypes: float64(37)
memory usage: 7.5 MB
None
```

Correlation Heatmap

In [ ]:

```
h_labels = [x.replace('_', ' ').title() for x in 
            list(df_train.select_dtypes(include=['number', 'bool']).columns.values)]
fig, ax = plt.subplots(figsize=(10,6))
_ = sns.heatmap(df_train.corr(), annot=True, xticklabels=h_labels, yticklabels=h_labe
```



Correlation Table

In [ ]:

```
df_train.corr()
```

Out[ ]:

	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavioral_av
h1n1_concern	1.000000	0.076568	0.085562	
h1n1_knowledge	0.076568	1.000000	-0.001816	
behavioral_antiviral_meds	0.085562	-0.001816	1.000000	
behavioral_avoidance	0.230465	0.095192	0.051383	
behavioral_face_mask	0.151493	0.029994	0.139040	
behavioral_wash_hands	0.289809	0.094993	0.065579	
behavioral_large_gatherings	0.250078	-0.039979	0.102030	
behavioral_outside_home	0.240748	-0.057778	0.121166	
behavioral_touch_face	0.246532	0.094727	0.069112	
doctor_recc_h1n1	0.092702	0.092033	0.047340	
doctor_recc_seasonal	0.083828	0.075442	0.029959	
chronic_med_condition	0.080785	0.022999	-0.004701	
child_under_6_months	0.044654	0.070276	0.004767	
health_worker	0.032723	0.187185	-0.007178	
health_insurance	-0.108791	-0.010431	0.002185	-
opinion_h1n1_vacc_effective	0.213980	0.135095	0.016339	
opinion_h1n1_risk	0.351031	0.088906	0.085836	
opinion_h1n1_sick_from_vacc	0.335772	0.002560	0.064301	
opinion_seas_vacc_effective	0.202705	0.114322	-0.001074	
opinion_seas_risk	0.307118	0.098998	0.063913	
opinion_seas_sick_from_vacc	0.208736	-0.029376	0.066733	
age_group	0.051225	-0.051356	-0.096319	-
education	-0.041615	0.116299	-0.011686	-
race	-0.126031	0.146961	-0.105900	
sex	-0.124677	-0.068803	-0.006408	-
income_poverty	0.014636	-0.122655	0.037858	-
marital_status	-0.029513	-0.152220	0.022828	-
rent_or_own	-0.002388	-0.150621	0.054453	-
employment_status	0.036798	-0.160954	0.020685	-
hhs_geo_region	-0.008409	-0.015268	0.007902	
census_msa	0.003074	-0.059151	0.010412	-
household_adults	-0.011020	0.033406	0.040948	
household_children	0.053092	0.060474	0.081529	
employment_industry	-0.008264	-0.085806	0.019574	-
employment_occupation	-0.023937	-0.078106	0.013989	-
h1n1_vaccine	0.120970	0.115295	0.033279	
seasonal_vaccine	0.152583	0.116014	0.000779	

37 rows × 37 columns

Variance

In [ ]:

```
df_train.var()
```

Out[ ]:

```
h1n1_concern                    0.849349
h1n1_knowledge                  0.402587
behavioral_antiviral_meds       0.049253
behavioral_avoidance            0.220567
behavioral_face_mask            0.064993
behavioral_wash_hands           0.148988
behavioral_large_gatherings     0.235270
behavioral_outside_home         0.228330
behavioral_touch_face           0.230957
doctor_recc_h1n1                0.268591
doctor_recc_seasonal            0.334589
chronic_med_condition           0.253347
child_under_6_months            0.108326
health_worker                   0.132504
health_insurance                0.934782
opinion_h1n1_vacc_effective     1.339503
opinion_h1n1_risk               1.788567
opinion_h1n1_sick_from_vacc     1.993946
opinion_seas_vacc_effective     1.589634
opinion_seas_risk               2.142554
opinion_seas_sick_from_vacc     1.932599
age_group                       2.123783
education                       1.407645
race                            0.852346
sex                             0.241215
income_poverty                  1.239758
marital_status                  0.353413
rent_or_own                     0.387288
employment_status               0.663105
hhs_geo_region                  7.873543
census_msa                      0.677845
household_adults                0.595224
household_children              0.875225
employment_industry            25.629023
employment_occupation          27.098985
h1n1_vaccine                    0.167323
seasonal_vaccine                0.248827
dtype: float64
```

Balancing/Oversampling minority classes using SMOTE .

In [ ]:

```
#df_labels
df_train
```

Out[ ]:

	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavioral_avoidance	behavioral_f
0	1.0	0.0	0.0	0.0	
1	3.0	2.0	0.0	1.0	
2	1.0	1.0	0.0	1.0	
3	1.0	1.0	0.0	1.0	
4	2.0	1.0	0.0	1.0	
...	...	...	...	...	
26702	2.0	0.0	0.0	1.0	
26703	1.0	2.0	0.0	1.0	
26704	2.0	2.0	0.0	1.0	
26705	1.0	1.0	0.0	0.0	
26706	0.0	0.0	0.0	1.0	

26707 rows × 37 columns

In [ ]:

```
# transform the dataset
X = df_train.drop(columns = ['h1n1_vaccine','seasonal_vaccine'])
y_swine = df_train['h1n1_vaccine']
y_seasonal = df_train['seasonal_vaccine']
oversample = SMOTE()
X, y_swine = oversample.fit_resample(X, y_swine)
#TODO the y_seasonal data is missing for some records. Need to investigate and corre
#X, y_seasonal = oversample.fit_resample(X, y_seasonal)

#Split the data TODO Split into 70 30 10 (train test and validation)
X_train, X_test, y_train_swine, y_test_swine = train_test_split(X,y_swine, train_siz
#X_train, X_test, y_train_swine, y_test_swine = train_test_split(X,y_seasonal, train_
```

Model Strategies

For the problem at hand, the most appropriate strategy was to fit a number of classification models to classify observations into vaccinated or un-vaccinated.

The following models were fit, tested, and tuned to produce the most accurate model possible based on F-1 score: neural network model, logistic regression, SVM, gradient boosting classifier, Gaussian Naive Bayes, KNN, Random Forest, and a boosted classifier.

In [ ]:

```
#Neural Network Model
NN = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5, 2), max_iter =
NN.fit(X_train, y_train_swine)
y_pred = NN.predict(X_test)
f1 = round(f1_score(y_test_swine, y_pred, average='macro'), 3)
print("\nNeural Network (using scaler inputs) f1 score: ", f1)
classificationSummary(y_test_swine, y_pred)

#Logistic Regression Model
logistic = LogisticRegressionCV(cv=5, penalty = 'l2', solver = 'liblinear',tol=1e-5,
logistic.fit(X_train, y_train_swine)
y_pred = logistic.predict(X_test)
f1 = round(f1_score(y_test_swine, y_pred, average='macro'), 3)
print("\nLogistic Regression (using scaler inputs) f1 score: ", f1)
classificationSummary(y_test_swine, y_pred)

#Support Vector Machine model
SVM = svm.LinearSVC(max_iter = 5000, penalty = 'l2', loss = 'hinge', random_state = 
SVM.fit(X_train, y_train_swine)
y_pred = SVM.predict(X_test)
f1 = round(f1_score(y_test_swine, y_pred, average='macro'), 3)
print("\nSVM (using scaler inputs) f1 score: ", f1)
classificationSummary(y_test_swine, y_pred)
```

```python
#Gradient Boosting Classifier
gb_classif = GradientBoostingClassifier()
gb_classif.fit(X_train, y_train_swine)
y_pred = gb_classif.predict(X_test)
f1 = round(f1_score(y_test_swine, y_pred, average='macro'), 3)
print("\nGradientBoosting (using scaler inputs) f1 score: ", f1)
classificationSummary(y_test_swine, y_pred)

#Gaussian Naive Bayes model
gnb = GaussianNB()
gnb.fit(X_train, y_train_swine)
y_pred = gnb.predict(X_test)
f1 = round(f1_score(y_test_swine, y_pred, average='macro'), 3)
print("\nNaive Bayes (using scaler inputs) f1 score: ", f1)
classificationSummary(y_test_swine, y_pred)

#K-Nearest Neighbors
knn = KNeighborsClassifier(n_neighbors=3, weights = 'distance')
knn.fit(X_train, y_train_swine)
y_pred = knn.predict(X_test)
f1 = round(f1_score(y_test_swine, y_pred, average='macro'), 3)
print("\nKNN f1 score: ", f1)
classificationSummary(y_test_swine, y_pred)

#Random Forest model
rf = RandomForestClassifier(max_depth=2, random_state = 12345)
rf.fit(X_train, y_train_swine)
y_pred = rf.predict(X_test)
f1 = round(f1_score(y_test_swine, y_pred, average='macro'), 3)
print("\nRandom Forest f1 score: ", f1)
classificationSummary(y_test_swine, y_pred)
```

```
Neural Network (using scaler inputs) f1 score:  0.331
Confusion Matrix (Accuracy 0.4945)

       Prediction
Actual    0    1
     0    0 6380
     1    0 6240

Logistic Regression (using scaler inputs) f1 score:  0.785
Confusion Matrix (Accuracy 0.7848)

       Prediction
Actual    0    1
     0 5046 1334
     1 1382 4858
C:\Users\elfek\AppData\Roaming\Python\Python39\site-packages\sklearn\svm\_base.py:122
5: ConvergenceWarning: Liblinear failed to converge, increase the number of iteration
s.
  warnings.warn(
SVM (using scaler inputs) f1 score:  0.79
Confusion Matrix (Accuracy 0.7899)

       Prediction
Actual    0    1
     0 5137 1243
     1 1409 4831

GradientBoosting (using scaler inputs) f1 score:  0.907
Confusion Matrix (Accuracy 0.9076)

       Prediction
Actual    0    1
     0 6033  347
     1  819 5421

Naive Bayes (using scaler inputs) f1 score:  0.752
Confusion Matrix (Accuracy 0.7517)

       Prediction
Actual    0    1
     0 4584 1796
     1 1338 4902

KNN f1 score:  0.805
Confusion Matrix (Accuracy 0.8101)

       Prediction
Actual    0    1
     0 4115 2265
     1  131 6109

Random Forest f1 score:  0.827
Confusion Matrix (Accuracy 0.8269)

       Prediction
Actual    0    1
     0 5439  941
     1 1243 4997
```

In [ ]:

```python
#Find best paramters of Adaboost TODO using random search instead of grid search
boost_grid = {
    'n_estimators': [50, 100, 200, 400],
    'learning_rate': [0.01, 0.1, 0.5, 1.0, 1.5, 2.0]
}

boost_gridSearch = GridSearchCV(AdaBoostClassifier(), boost_grid, cv = 3)
boost_gridSearch.fit(X_train, y_train_swine)

print('Initial Adaboost Parameters:', boost_gridSearch.best_params_)
```

```
Initial Adaboost Parameters: {'learning_rate': 1.0, 'n_estimators': 400}
```

Confusion Matrix of Best Performing Model (Adaboost or Gradiantboost I think but we need to tinker with the hyperparameters).

In [ ]:

```python
#Boosted Classifier
adaboost = AdaBoostClassifier(DecisionTreeClassifier(max_depth = 2), learning_rate =
adaboost.fit(X_train, y_train_swine)
y_pred = adaboost.predict(X_test)
f1 = round(f1_score(y_test_swine, y_pred, average='macro'), 3)
print("\nAdaboost f1 score: ", f1)
#classificationSummary(y_test, y_pred)
```

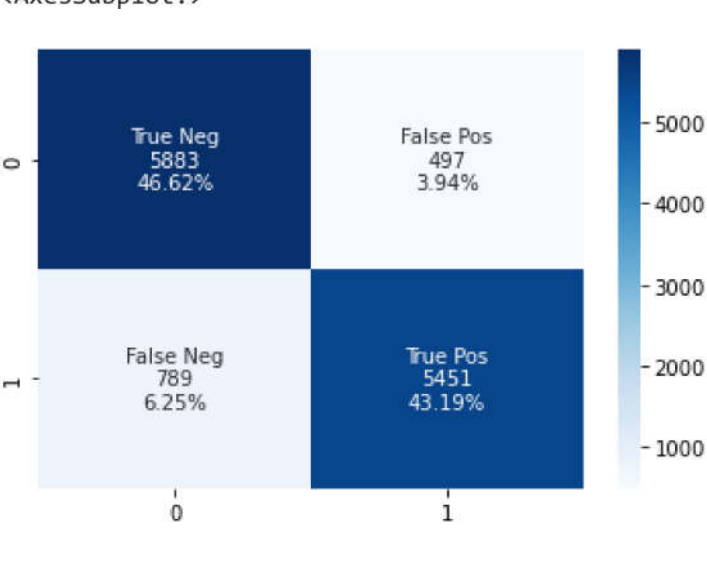
```
Adaboost f1 score:  0.898
```

In [ ]:

```python
#Create plot of the confusion matrix, showing performance of the model
cm = confusion_matrix(y_test_swine, y_pred)
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ["{0:0.0f}".format(value) for value in
                cm.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                     cm.flatten()/np.sum(cm)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(cm, annot=labels, fmt='', cmap='Blues')
```

Out[ ]:

```
<AxesSubplot:>
```



Results and Conclusions

The final results indicate that the best model for predicting...