Using Survey Data as a Predictor of Pandemic Vaccination
2b - Classification Modeling

Mark Patterson, March 2021

Introduction to 2b: Classification Modeling

This is part 2 of classification modeling. After figuring things out and running some base models in the other notebook, this notebook contains the remainder of the m
including a summary of model stats at the end of this notebook. All told about 34 models were run - 25 on the target variable of h1n1 vaccination and 9 for the season
target variable.

In [1]:
```
# Import the relevant libraries
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns

from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.impute import KNNImputer
from imblearn.over_sampling import SMOTE
from sklearn.pipeline import Pipeline
from sklearn.pipeline import make_pipeline
from imblearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
from xgboost import XGBClassifier
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, explained_variance_score, confusion_matrix, accuracy_score, classification_report, log_loss
from math import sqrt
from sklearn.metrics import accuracy_score, roc_curve, auc
from sklearn.preprocessing import OneHotEncoder
from sklearn import tree
from sklearn.model_selection import cross_val_score

%matplotlib inline

# Increase column width to display df
pd.set_option('display.max_columns', None)
```

In [2]:
```
# Load the data
df_7 = pd.read_csv('data/df_5.csv')

# print the shape
print(df_7.shape)
df_7.head()
```

```
(26707, 37)
```

Out[2]:

	Unnamed: 0	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
0	0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	
1	1	3.0	2.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	
2	2	1.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	
3	3	1.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	
4	4	2.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	

In [3]:
```
print(df_7['h1n1_vaccine'].value_counts())
df_7['h1n1_vaccine'].value_counts(normalize=True)*100
```

```
0    21033
1     5674
Name: h1n1_vaccine, dtype: int64
```

Out[3]:
```
0    78.754634
1    21.245366
Name: h1n1_vaccine, dtype: float64
```

In [4]:
```
df_11 = df_7.drop(columns=['Unnamed: 0'], axis=1)
```

In [5]:
```
# KNNImputer
imputer = KNNImputer(n_neighbors=5)
df_11 = pd.DataFrame(imputer.fit_transform(df_11),columns = df_11.columns)
df_11.head()
```

Out[5]:

	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
0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0
1	3.0	2.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0
2	1.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0
4	2.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0

In [ ]:
```
```

In [7]:
```
# Scaler - oops, forgot to remove Y so it got jumbled too. Should do on just X?
scale = StandardScaler()
df_12 = pd.DataFrame(scale.fit_transform(df_11),columns = df_11.columns)
df_12.head()
```

Out[7]:

	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
0	-0.679706	-2.043198	-0.226969	-1.628644	-0.272519	-2.177411	-0.748768	1.402893	0.692148	-0.540137
1	1.519686	1.196125	-0.226969	0.618083	-0.272519	0.459682	-0.748768	1.402893	0.692148	-0.540137
2	-0.679706	-0.423537	-0.226969	0.618083	-0.272519	-2.177411	-0.748768	-0.714440	-1.450219	-0.540137
3	-0.679706	-0.423537	-0.226969	0.618083	-0.272519	0.459682	1.338595	-0.714440	-1.450219	-0.540137
4	0.419990	-0.423537	-0.226969	0.618083	-0.272519	0.459682	1.338595	-0.714440	0.692148	-0.540137

In [8]:
```
df_11.head()
```

Out[8]:

ı_seas_sick_from_vacc	age_group	education	race	sex	income_poverty	marital_status	rent_or_own	employment_status	hhs_geo_region	census_msa	household_adults	household_children	employment_industry	h1n1_vaccine
2.0	3.0	0.0	3.0	0.0	0.0	0.0	1.0	1.0	8.0	2.0	0.0	0.0	4.8	0.0
4.0	1.0	1.0	3.0	1.0	0.0	0.0	0.0	2.0	1.0	0.0	0.0	0.0	3.0	0.0
2.0	0.0	3.0	3.0	1.0	1.0	0.0	1.0	2.0	9.0	0.0	2.0	0.0	9.0	0.0
1.0	4.0	1.0	3.0	0.0	0.0	0.0	0.0	1.0	5.0	1.0	0.0	0.0	5.0	0.0
4.0	2.0	2.0	3.0	0.0	1.0	1.0	1.0	2.0	9.0	0.0	1.0	0.0	1.0	0.0

In [34]:
```
df_11.tail()
```

Out[34]:

	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_h
26702	2.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	
26703	1.0	2.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	
26704	2.0	2.0	0.0	1.0	1.0	1.0	1.0	0.0	1.0	
26705	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.6	
26706	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	

In [35]:
```
# Export the imputed step for use in other notebooks
df_11.to_csv(r'df1imputed.csv')
```

In [12]:
```
# Cut the y out of the df.
y = df_11['h1n1_vaccine']
y.tail()
```

Out[12]:
```
26702    0.0
26703    0.0
26704    0.0
26705    0.0
26706    0.0
Name: h1n1_vaccine, dtype: float64
```

In [38]:
```
y.head()
```

Out[38]:
```
0    0.0
1    0.0
2    0.0
3    0.0
4    0.0
Name: h1n1_vaccine, dtype: float64
```

In [39]:
```
# Export the y for use in other notebooks
y.to_csv(r'df1y.csv')
```

```
/Users/markp/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/ipykernel_launcher.py:2: FutureWarning: The signature of `Series.to_csv` was aligned to that of
_csv`, and argument 'header' will change its default value from False to True: please pass an explicit value to suppress this warning.
```

In [13]:
```
df_13 = df_11.drop(columns=['h1n1_vaccine', 'seasonal_vaccine'], axis=1)
df_13.head()
```

Out[13]:

	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
0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0
1	3.0	2.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0
2	1.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0
4	2.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0

In [14]:
```
# Scaler - take 2
scale = StandardScaler()
df_14 = pd.DataFrame(scale.fit_transform(df_13),columns = df_13.columns)
df_14.head()
```

Out[14]:

	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
0	-0.679706	-2.043198	-0.226969	-1.628644	-0.272519	-2.177411	-0.748768	1.402893	0.692148	-0.540137
1	1.519686	1.196125	-0.226969	0.618083	-0.272519	0.459682	-0.748768	1.402893	0.692148	-0.540137
2	-0.679706	-0.423537	-0.226969	0.618083	-0.272519	-2.177411	-0.748768	-0.714440	-1.450219	-0.540137
3	-0.679706	-0.423537	-0.226969	0.618083	-0.272519	0.459682	1.338595	-0.714440	-1.450219	-0.540137
4	0.419990	-0.423537	-0.226969	0.618083	-0.272519	0.459682	1.338595	-0.714440	0.692148	-0.540137

In [36]:
```
df_14.tail()
```

Out[36]:

	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_h
26702	0.419990	-2.043198	-0.226969	0.618083	-0.272519	-2.177411	-0.748768	1.402893	-1.450219	-0.540
26703	-0.679706	1.196125	-0.226969	0.618083	-0.272519	0.459682	-0.748768	-0.714440	-1.450219	1.951
26704	0.419990	1.196125	-0.226969	0.618083	3.673573	0.459682	1.338595	-0.714440	0.692148	-0.540
26705	-0.679706	-0.423537	-0.226969	-1.628644	-0.272519	-2.177411	-0.748768	-0.714440	-0.164799	-0.540
26706	-1.779402	-2.043198	-0.226969	0.618083	-0.272519	-2.177411	-0.748768	-0.714440	-1.450219	-0.540

In [37]:
```
# Export the scaled step (post imputed) for use in other notebooks
df_14.to_csv(r'df1impscaled.csv')
```

In [15]:
```
# Need to split data into X and y dataframes.
X = df_14
print(y.shape)
print(X.shape)
```

```
(26707,)
(26707, 34)
```

In [16]:
```
# Create train and test sets.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=35)
print(X_train.shape)
print(X_test.shape)

print(y_train.shape)
print(y_test.shape)
```

```
(18694, 34)
(8013, 34)
(18694,)
(8013,)
```

In [17]:
```
# Address the target class imbalance with SMOTE - as a seperate step. Not sure how to get the output of pre-pipe?

print("Before OverSampling, counts of label '0': {}".format(sum(y_train == 0)))
print("Before OverSampling, counts of label '1': {}".format(sum(y_train == 1)))

sm = SMOTE(random_state = 4)
X_train_s, y_train_s = sm.fit_sample(X_train, y_train)

print('After OverSampling, the shape of train_X: {}'.format(X_train_s.shape))
print('After OverSampling, the shape of train_y: {} \n'.format(y_train_s.shape))

print("After OverSampling, counts of label '0': {}".format(sum(y_train_s == 0)))
print("After OverSampling, counts of label '1': {}".format(sum(y_train_s == 1)))
```

```
Before OverSampling, counts of label '0': 14713
Before OverSampling, counts of label '1': 3981
After OverSampling, the shape of train_X: (29426, 34)
After OverSampling, the shape of train_y: (29426,)

After OverSampling, counts of label '0': 14713
After OverSampling, counts of label '1': 14713
```

In [ ]:

Run the models - the SMOTE versions

In [18]:
```
def run_smodel (model):
    # Instantiate classification model
    smodel = model
    # Fit the classifier
    smodel.fit(X_train_s, y_train_s)

    # Predict on training and test sets
    training_preds = smodel.predict(X_train_s)
    test_preds = smodel.predict(X_test)

    # Get detailed results (Train and Test)
    print('-------------------------------------------------------------------------------')
    print(f'MODEL: {model}')
    # Classification Report
    print('-------------------------------------------------------------------------------')
    print('Classification Report - TRAIN')
    print('-------------------------------------------------------------------------------')
    print(classification_report(y_train_s, training_preds))
    # Confusion Matrix
    print('-------------------------------------------------------------------------------')
    print('Confusion Matrix - TRAIN')
    print('-------------------------------------------------------------------------------')
    print(pd.crosstab(y_train_s, training_preds, rownames=['True'], colnames=['Predicted'], margins=True))
    print('\n-------------------------------------------------------------------------------')
    # Classification Report
    print('-------------------------------------------------------------------------------')
    print('Classification Report - TEST')
    print('-------------------------------------------------------------------------------')
    print(classification_report(y_test, test_preds))
    # Confusion Matrix
    print('-------------------------------------------------------------------------------')
    print('Confusion Matrix - TEST')
    print('-------------------------------------------------------------------------------')
    print(pd.crosstab(y_test, test_preds, rownames=['True'], colnames=['Predicted'], margins=True))
    print('-------------------------------------------------------------------------------')
```

In [21]:
```
model = RandomForestClassifier()
run_smodel (model)
```

```
------------------------------------------------------------------------------
MODEL: RandomForestClassifier()
------------------------------------------------------------------------------
Classification Report - TRAIN
------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     14713
         1.0       1.00      1.00      1.00     14713

    accuracy                           1.00     29426
   macro avg       1.00      1.00      1.00     29426
weighted avg       1.00      1.00      1.00     29426

------------------------------------------------------------------------------
Confusion Matrix - TRAIN
------------------------------------------------------------------------------
Predicted    0.0    1.0    All
True
0.0        14713      0  14713
1.0            0  14713  14713
All        14713  14713  29426

------------------------------------------------------------------------------
------------------------------------------------------------------------------
Classification Report - TEST
------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.87      0.93      0.90      6320
         1.0       0.64      0.48      0.55      1693

    accuracy                           0.83      8013
   macro avg       0.76      0.70      0.72      8013
weighted avg       0.82      0.83      0.82      8013

------------------------------------------------------------------------------
Confusion Matrix - TEST
------------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        5866   454  6320
1.0         878   815  1693
All        6744  1269  8013
------------------------------------------------------------------------------
```

In [22]:
```
# Take a look at feature importances (RandomForest) - from SMOTE model

importance = pd.DataFrame(data={'features': X_train_s.columns, 'importance': model.feature_importances_})
importance = importance.sort_values('importance', ascending=False)
importance = importance.reset_index()
importance.drop('index', axis=1, inplace=True)
importance.head(20)
```

Out[22]:

	features	importance
0	opinion_h1n1_vacc_effective	0.131235
1	doctor_recc_h1n1	0.096571
2	opinion_h1n1_risk	0.092553
3	opinion_seas_vacc_effective	0.066621
4	opinion_seas_risk	0.059219
5	employment_industry	0.043681
6	h1n1_knowledge	0.036167
7	h1n1_concern	0.035785
8	hhs_geo_region	0.035431
9	age_group	0.030194
10	opinion_h1n1_sick_from_vacc	0.029448
11	health_insurance	0.027820
12	education	0.026750
13	opinion_seas_sick_from_vacc	0.026499
14	doctor_recc_seasonal	0.025959
15	income_poverty	0.022320
16	census_msa	0.022155
17	household_adults	0.019086
18	employment_status	0.016225
19	household_children	0.015846

In [23]:
```
model = KNeighborsClassifier()
run_smodel (model)
```

```
------------------------------------------------------------------------
MODEL: KNeighborsClassifier()
------------------------------------------------------------------------
Classification Report - TRAIN
------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.99      0.75      0.85     14713
         1.0       0.80      0.99      0.88     14713

    accuracy                           0.87     29426
   macro avg       0.89      0.87      0.87     29426
weighted avg       0.89      0.87      0.87     29426

------------------------------------------------------------------------
Confusion Matrix - TRAIN
------------------------------------------------------------------------
Predicted    0.0    1.0    All
True
0.0        11032   3681  14713
1.0          142  14571  14713
All        11174  18252  29426

------------------------------------------------------------------------
------------------------------------------------------------------------
Classification Report - TEST
------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.89      0.66      0.76      6320
         1.0       0.35      0.70      0.47      1693

    accuracy                           0.67      8013
   macro avg       0.62      0.68      0.61      8013
weighted avg       0.78      0.67      0.70      8013

------------------------------------------------------------------------
Confusion Matrix - TEST
------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        4141  2179  6320
1.0         500  1193  1693
All        4641  3372  8013
------------------------------------------------------------------------
```

In [24]:
```
model = XGBClassifier()
run_smodel (model)
```

```
------------------------------------------------------------------------
MODEL: XGBClassifier()
------------------------------------------------------------------------
Classification Report - TRAIN
------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.87      0.92      0.89     14713
         1.0       0.91      0.87      0.89     14713

    accuracy                           0.89     29426
   macro avg       0.89      0.89      0.89     29426
weighted avg       0.89      0.89      0.89     29426

------------------------------------------------------------------------
Confusion Matrix - TRAIN
------------------------------------------------------------------------
Predicted    0.0    1.0    All
True
0.0        13519   1194  14713
1.0         1979  12734  14713
All        15498  13928  29426

------------------------------------------------------------------------
------------------------------------------------------------------------
Classification Report - TEST
------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.88      0.91      0.89      6320
         1.0       0.61      0.52      0.56      1693

    accuracy                           0.83      8013
   macro avg       0.74      0.71      0.73      8013
weighted avg       0.82      0.83      0.82      8013

------------------------------------------------------------------------
Confusion Matrix - TEST
------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        5755   565  6320
1.0         814   879  1693
All        6569  1444  8013
------------------------------------------------------------------------
```

In [25]:
```
model = DecisionTreeClassifier()
run_smodel (model)
```

```
--------------------------------------------------------------------------------
MODEL: DecisionTreeClassifier()
--------------------------------------------------------------------------------
Classification Report - TRAIN
--------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     14713
         1.0       1.00      1.00      1.00     14713

    accuracy                           1.00     29426
   macro avg       1.00      1.00      1.00     29426
weighted avg       1.00      1.00      1.00     29426

--------------------------------------------------------------------------------
Confusion Matrix - TRAIN
--------------------------------------------------------------------------------
Predicted    0.0    1.0    All
True
0.0        14713      0  14713
1.0            0  14713  14713
All        14713  14713  29426

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.85      0.82      0.83      6320
         1.0       0.41      0.46      0.43      1693

    accuracy                           0.74      8013
   macro avg       0.63      0.64      0.63      8013
weighted avg       0.76      0.74      0.75      8013

--------------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        5176  1144  6320
1.0         909   784  1693
All        6085  1928  8013
--------------------------------------------------------------------------------
```

In [26]:
```
model = LogisticRegression()
run_smodel (model)
```

```
--------------------------------------------------------------------------------
MODEL: LogisticRegression()
--------------------------------------------------------------------------------
Classification Report - TRAIN
--------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.76      0.79      0.78     14713
         1.0       0.78      0.75      0.77     14713

    accuracy                           0.77     29426
   macro avg       0.77      0.77      0.77     29426
weighted avg       0.77      0.77      0.77     29426

--------------------------------------------------------------------------------
Confusion Matrix - TRAIN
--------------------------------------------------------------------------------
Predicted    0.0    1.0    All
True
0.0        11641   3072  14713
1.0         3609  11104  14713
All        15250  14176  29426

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.92      0.79      0.85      6320
         1.0       0.48      0.73      0.58      1693

    accuracy                           0.77      8013
   macro avg       0.70      0.76      0.71      8013
weighted avg       0.82      0.77      0.79      8013

--------------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        4965  1355  6320
1.0         461  1232  1693
All        5426  2587  8013
--------------------------------------------------------------------------------
```

In [27]:
```
from sklearn.svm import SVC
```

In [28]:
```
model = SVC()
run_smodel (model)
```

```
------------------------------------------------------------------------
MODEL: SVC()
------------------------------------------------------------------------
Classification Report - TRAIN
------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.89      0.89      0.89     14713
         1.0       0.89      0.89      0.89     14713

    accuracy                           0.89     29426
   macro avg       0.89      0.89      0.89     29426
weighted avg       0.89      0.89      0.89     29426

------------------------------------------------------------------------
Confusion Matrix - TRAIN
------------------------------------------------------------------------
Predicted    0.0    1.0    All
True
0.0        13079   1634  14713
1.0         1548  13165  14713
All        14627  14799  29426

------------------------------------------------------------------------
------------------------------------------------------------------------
Classification Report - TEST
------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.89      0.85      0.87      6320
         1.0       0.52      0.63      0.57      1693

    accuracy                           0.80      8013
   macro avg       0.71      0.74      0.72      8013
weighted avg       0.82      0.80      0.81      8013

------------------------------------------------------------------------
Confusion Matrix - TEST
------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        5353   967  6320
1.0         634  1059  1693
All        5987  2026  8013
------------------------------------------------------------------------
```

Try and optimize XGBoost model with GridSearchCV

In [29]:
```
model = XGBClassifier()
run_smodel (model)
```

```
------------------------------------------------------------------------
MODEL: XGBClassifier()
------------------------------------------------------------------------
Classification Report - TRAIN
------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.87      0.92      0.89     14713
         1.0       0.91      0.87      0.89     14713

    accuracy                           0.89     29426
   macro avg       0.89      0.89      0.89     29426
weighted avg       0.89      0.89      0.89     29426

------------------------------------------------------------------------
Confusion Matrix - TRAIN
------------------------------------------------------------------------
Predicted    0.0    1.0    All
True
0.0        13519   1194  14713
1.0         1979  12734  14713
All        15498  13928  29426

------------------------------------------------------------------------
------------------------------------------------------------------------
Classification Report - TEST
------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.88      0.91      0.89      6320
         1.0       0.61      0.52      0.56      1693

    accuracy                           0.83      8013
   macro avg       0.74      0.71      0.73      8013
weighted avg       0.82      0.83      0.82      8013

------------------------------------------------------------------------
Confusion Matrix - TEST
------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        5755   565  6320
1.0         814   879  1693
All        6569  1444  8013
------------------------------------------------------------------------
```

In [30]:
```
# Set-up the parameter grid
param_grid = {
    'learning_rate': [0.1, 0.4],
    'max_depth': [5, 6],
    'min_child_weight': [2, 4],
    'subsample': [0.4, 0.7],
    'n_estimators': [100],
}
```

In [31]:
```
# Code to run it

grid_clf = GridSearchCV(model, param_grid, scoring='accuracy', cv=5, n_jobs=1)
grid_clf.fit(X_train_s, y_train_s)

best_parameters = grid_clf.best_params_

print('Grid Search found the following optimal parameters: ')
for param_name in sorted(best_parameters.keys()):
    print('%s: %r' % (param_name, best_parameters[param_name]))

training_preds = grid_clf.predict(X_train_s)
test_preds = grid_clf.predict(X_test)
training_accuracy = accuracy_score(y_train_s, training_preds)
test_accuracy = accuracy_score(y_test, test_preds)

print('')
print('Training Accuracy: {:.4}%'.format(training_accuracy * 100))
print('Validation accuracy: {:.4}%'.format(test_accuracy * 100))
```

```
Grid Search found the following optimal parameters:
learning_rate: 0.1
max_depth: 6
min_child_weight: 4
n_estimators: 100
subsample: 0.7

Training Accuracy: 91.57%
Validation accuracy: 83.79%
```

In [32]:
```
# Set-up the parameter grid
param_grid = {
    'learning_rate': [0.1, 0.2],
    'max_depth': [6, 7],
    'min_child_weight': [4, 6],
    'subsample': [0.7, 0.8],
    'n_estimators': [100],
}
```

In [33]:
```
# Code to run it

grid_clf = GridSearchCV(model, param_grid, scoring='accuracy', cv=5, n_jobs=1)
grid_clf.fit(X_train_s, y_train_s)

best_parameters = grid_clf.best_params_

print('Grid Search found the following optimal parameters: ')
for param_name in sorted(best_parameters.keys()):
    print('%s: %r' % (param_name, best_parameters[param_name]))

print(grid_clf.best_estimator_)

training_preds = grid_clf.predict(X_train_s)
test_preds = grid_clf.predict(X_test)
training_accuracy = accuracy_score(y_train_s, training_preds)
test_accuracy = accuracy_score(y_test, test_preds)

print('')
print('Training Accuracy: {:.4}%'.format(training_accuracy * 100))
print('Validation accuracy: {:.4}%'.format(test_accuracy * 100))
```

```
Grid Search found the following optimal parameters:
learning_rate: 0.1
max_depth: 7
min_child_weight: 4
n_estimators: 100
subsample: 0.7
XGBClassifier(max_depth=7, min_child_weight=4, subsample=0.7)

Training Accuracy: 92.47%
Validation accuracy: 83.46%
```

In [ ]:
```

```

Approach B to Data Prep: better data prep with one-hot encoding

In [74]:
```
# Reload the data and start from scratch
# Load the data
raw_data_x = pd.read_csv('data/training_set_features.csv')
raw_data_y = pd.read_csv('data/training_set_labels.csv')

# print the shape
print("Raw_data_x:", raw_data_x.shape)
print("Raw_data_y:", raw_data_y.shape)
```

```
Raw_data_x: (26707, 36)
Raw_data_y: (26707, 3)
```

In [75]:
```
# Combine 2 original dataframes into one
rawc_all = pd.merge(raw_data_x, raw_data_y, on="respondent_id", how="inner")
print(rawc_all.shape)
rawc_all.head()
```

```
(26707, 38)
```

Out[75]:

ın_seas_vacc_effective	opinion_seas_risk	opinion_seas_sick_from_vacc	age_group	education	race	sex	income_poverty	marital_status	rent_or_own	employment_status	hhs_geo_region	census_msa	household_adults	hou
2.0	1.0	2.0	55 - 64 Years	< 12 Years	White	Female	Below Poverty	Not Married	Own	Not in Labor Force	oxchjgsf	Non-MSA	0.0	
4.0	2.0	4.0	35 - 44 Years	12 Years	White	Male	Below Poverty	Not Married	Rent	Employed	bhuqouqj	MSA, Not Principle City	0.0	
4.0	1.0	2.0	18 - 34 Years	College Graduate	White	Male	<= $75,000, Above Poverty	Not Married	Own	Employed	qufhixun	MSA, Not Principle City	2.0	
5.0	4.0	1.0	65+ Years	12 Years	White	Female	Below Poverty	Not Married	Rent	Not in Labor Force	lrircsnp	MSA, Principle City	0.0	
3.0	1.0	4.0	45 - 54 Years	Some College	White	Female	<= $75,000, Above Poverty	Married	Own	Employed	qufhixun	MSA, Not Principle City	1.0	

In [76]:
```
# save rawb_all full df as csv file for later use - modeling
rawc_all.to_csv(r'df_all.csv')
```

Transform categorical text to numerical

This time, I will take a different approach to transforming the remaining text values into numerical values.
a) I will recode the 6 opinion questions, so the scale starts at 0 rather than 1.
b) I will use OrdinalEncoder ONLY where the features truly have are ordinal.
c) I will use One-hot-encoder for other variables. In the previous approach some features may have had an unfair advantage in the models as they had values from 0 to
(and they were NOT ordinal in nature).

In [78]:
```
# Recode the 6 opinion questions
opinions = {1 : 0, 2 : 1, 3 : 2, 4 : 3, 5 : 4}
rawc_all['opinion_h1n1_vacc_effective'] = rawc_all['opinion_h1n1_vacc_effective'].map(opinions)
rawc_all.head()
```

Out[78]:

ffective	opinion_seas_risk	opinion_seas_sick_from_vacc	age_group	education	race	sex	income_poverty	marital_status	rent_or_own	employment_status	hhs_geo_region	census_msa	household_adults	household_childre
2.0	1.0	2.0	55 - 64 Years	< 12 Years	White	Female	Below Poverty	Not Married	Own	Not in Labor Force	oxchjgsf	Non-MSA	0.0	0
4.0	2.0	4.0	35 - 44 Years	12 Years	White	Male	Below Poverty	Not Married	Rent	Employed	bhuqouqj	MSA, Not Principle City	0.0	0
4.0	1.0	2.0	18 - 34 Years	College Graduate	White	Male	<= $75,000, Above Poverty	Not Married	Own	Employed	qufhixun	MSA, Not Principle City	2.0	0
5.0	4.0	1.0	65+ Years	12 Years	White	Female	Below Poverty	Not Married	Rent	Not in Labor Force	lrircsnp	MSA, Principle City	0.0	0
3.0	1.0	4.0	45 - 54 Years	Some College	White	Female	<= $75,000, Above Poverty	Married	Own	Employed	qufhixun	MSA, Not Principle City	1.0	0

In [79]:
```
opinions = {1 : 0, 2 : 1, 3 : 2, 4 : 3, 5 : 4}
rawc_all['opinion_h1n1_risk'] = rawc_all['opinion_h1n1_risk'].map(opinions)
```

In [80]:
```
opinions = {1 : 0, 2 : 1, 3 : 2, 4 : 3, 5 : 4}
rawc_all['opinion_h1n1_sick_from_vacc'] = rawc_all['opinion_h1n1_sick_from_vacc'].map(opinions)
rawc_all.head()
```

Out[80]:

onths	health_worker	health_insurance	opinion_h1n1_vacc_effective	opinion_h1n1_risk	opinion_h1n1_sick_from_vacc	opinion_seas_vacc_effective	opinion_seas_risk	opinion_seas_sick_from_vacc	age_group	education	race	
0.0	0.0	1.0	2.0	0.0	1.0	2.0	1.0	2.0	55 - 64 Years	< 12 Years	White	Fem
0.0	0.0	1.0	4.0	3.0	3.0	4.0	2.0	4.0	35 - 44 Years	12 Years	White	M
0.0	0.0	NaN	2.0	0.0	0.0	4.0	1.0	2.0	18 - 34 Years	College Graduate	White	M
0.0	0.0	NaN	2.0	2.0	4.0	5.0	4.0	1.0	65+ Years	12 Years	White	Fem
0.0	0.0	NaN	2.0	2.0	1.0	3.0	1.0	4.0	45 - 54 Years	Some College	White	Fem

In [81]:
```
opinions = {1 : 0, 2 : 1, 3 : 2, 4 : 3, 5 : 4}
rawc_all['opinion_seas_sick_from_vacc'] = rawc_all['opinion_seas_sick_from_vacc'].map(opinions)
```

In [82]:
```
opinions = {1 : 0, 2 : 1, 3 : 2, 4 : 3, 5 : 4}
rawc_all['opinion_seas_risk'] = rawc_all['opinion_seas_risk'].map(opinions)
```

In [83]:
```
opinions = {1 : 0, 2 : 1, 3 : 2, 4 : 3, 5 : 4}
rawc_all['opinion_seas_vacc_effective'] = rawc_all['opinion_seas_vacc_effective'].map(opinions)
rawc_all.head()
```

Out[83]:

k	opinion_h1n1_sick_from_vacc	opinion_seas_vacc_effective	opinion_seas_risk	opinion_seas_sick_from_vacc	age_group	education	race	sex	income_poverty	marital_status	rent_or_own	employment_status	hhs_geo_regio
0	1.0	1.0	0.0	1.0	55 - 64 Years	< 12 Years	White	Female	Below Poverty	Not Married	Own	Not in Labor Force	oxchjg
0	3.0	3.0	1.0	3.0	35 - 44 Years	12 Years	White	Male	Below Poverty	Not Married	Rent	Employed	bhuqou
0	0.0	3.0	0.0	1.0	18 - 34 Years	College Graduate	White	Male	<= $75,000, Above Poverty	Not Married	Own	Employed	qufhixu
0	4.0	4.0	3.0	0.0	65+ Years	12 Years	White	Female	Below Poverty	Not Married	Rent	Not in Labor Force	lrircsn
0	1.0	2.0	0.0	3.0	45 - 54 Years	Some College	White	Female	<= $75,000, Above Poverty	Married	Own	Employed	qufhixu

In [84]:
```
# 3 features are binary so just map to 0 and 1.
sex = {'Female': 0,'Male': 1,}
rawc_all['sex'] = rawc_all['sex'].map(sex)
```

In [85]:
```
marital = {'Not Married': 0,'Married': 1,}
rawc_all['marital_status'] = rawc_all['marital_status'].map(marital)
```

In [86]:
```
home = {'Rent': 0,'Own': 1}
rawc_all['rent_or_own'] = rawc_all['rent_or_own'].map(home)
rawc_all.head()
```

Out[86]:

pinion_seas_vacc_effective	opinion_seas_risk	opinion_seas_sick_from_vacc	age_group	education	race	sex	income_poverty	marital_status	rent_or_own	employment_status	hhs_geo_region	census_msa	household_adults	ho
1.0	0.0	1.0	55 - 64 Years	< 12 Years	White	0	Below Poverty	0.0	1.0	Not in Labor Force	oxchjgsf	Non-MSA	0.0	
3.0	1.0	3.0	35 - 44 Years	12 Years	White	1	Below Poverty	0.0	0.0	Employed	bhuqouqj	MSA, Not Principle City	0.0	
3.0	0.0	1.0	18 - 34 Years	College Graduate	White	1	<= $75,000, Above Poverty	0.0	1.0	Employed	qufhixun	MSA, Not Principle City	2.0	
4.0	3.0	0.0	65+ Years	12 Years	White	0	Below Poverty	0.0	0.0	Not in Labor Force	lrircsnp	MSA, Principle City	0.0	
2.0	0.0	3.0	45 - 54 Years	Some College	White	0	<= $75,000, Above Poverty	1.0	1.0	Employed	qufhixun	MSA, Not Principle City	1.0	

In [103]:
```
rawc_all.head()
```

Out[103]:

pinion_seas_vacc_effective	opinion_seas_risk	opinion_seas_sick_from_vacc	age_group	education	race	sex	income_poverty	marital_status	rent_or_own	employment_status	hhs_geo_region	census_msa	household_adults	ho
1.0	0.0	1.0	55 - 64 Years	< 12 Years	White	0	Below Poverty	0.0	1.0	Not in Labor Force	oxchjgsf	Non-MSA	0.0	
3.0	1.0	3.0	35 - 44 Years	12 Years	White	1	Below Poverty	0.0	0.0	Employed	bhuqouqj	MSA, Not Principle City	0.0	
3.0	0.0	1.0	18 - 34 Years	College Graduate	White	1	<= $75,000, Above Poverty	0.0	1.0	Employed	qufhixun	MSA, Not Principle City	2.0	
4.0	3.0	0.0	65+ Years	12 Years	White	0	Below Poverty	0.0	0.0	Not in Labor Force	lrircsnp	MSA, Principle City	0.0	
2.0	0.0	3.0	45 - 54 Years	Some College	White	0	<= $75,000, Above Poverty	1.0	1.0	Employed	qufhixun	MSA, Not Principle City	1.0	

In [104]:
```
# drop the feature of employment_occupation as it is repetitive.
rawf_all = rawc_all.drop(columns=['employment_occupation'], axis=1)
rawf_all.shape
```

Out[104]: (26707, 37)

In [88]:
```
# save rawb_all full df as csv file for later use - modeling
rawd_all.to_csv(r'df_all_columns.csv')
```

In [105]:
```
rawf_all['education'].value_counts(normalize=True)*100
```

Out[105]:
```
College Graduate    39.909091
Some College        27.837945
12 Years            22.913043
< 12 Years           9.339921
Name: education, dtype: float64
```

In [106]:
```
rawf_all['age_group'].value_counts(normalize=True)*100
```

Out[106]:
```
65+ Years        25.622496
55 - 64 Years    20.829745
45 - 54 Years    19.612836
18 - 34 Years    19.526716
35 - 44 Years    14.408208
Name: age_group, dtype: float64
```

In [92]:
```
# Actually, I am going to do Ordinal Encoder for education and age... seems to me we should consider the inherent order.
# BUT... it doesn's like the NANs so back to direct mapping.
# oe = OrdinalEncoder(categories=[['< 12 Years ', '12 Years', 'Some College', 'College Graduate'], ['18 - 34 Years', '35 - 44 Years', '45 - 54 Years', '55 - 64 Years'
# oe.fit_transform(rawd_all[['education', 'age_group']])
# rawd_all.head()
```

In [107]:
```
edu = {'< 12 Years': 0,'12 Years': 1,'Some College': 2,'College Graduate': 3}
rawf_all['education'] = rawf_all['education'].map(edu)
rawf_all.head()
```

Out[107]:

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavioral_avoidance	behavioral_face_mask	behavioral_wash_hands	behavioral_large_gatherings	behavioral_outside_home	behavioral_touch_face	dc
0	0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	
1	1	3.0	2.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	
2	2	1.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	
3	3	1.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	
4	4	2.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	

In [108]:
```
age = {'18 - 34 Years': 0,'35 - 44 Years': 1,'45 - 54 Years': 2,'55 - 64 Years': 3, '65+ Years': 4}
rawf_all['age_group'] = rawf_all['age_group'].map(age)
rawf_all.head()
```

Out[108]:

	opinion_seas_sick_from_vacc	age_group	education	race	sex	income_poverty	marital_status	rent_or_own	employment_status	hhs_geo_region	census_msa	household_adults	household_children	employment_industry	h1n
	1.0	3	0.0	White	0	Below Poverty	0.0	1.0	Not in Labor Force	oxchjgsf	Non-MSA	0.0	0.0	NaN	
	3.0	1	1.0	White	1	Below Poverty	0.0	0.0	Employed	bhuqouqj	MSA, Not Principle City	0.0	0.0	pxcmvdjn	
	1.0	0	3.0	White	1	<= $75,000, Above Poverty	0.0	1.0	Employed	qufhixun	MSA, Not Principle City	2.0	0.0	rucpziij	
	0.0	4	1.0	White	0	Below Poverty	0.0	0.0	Not in Labor Force	lrircsnp	MSA, Principle City	0.0	0.0	NaN	
	3.0	2	2.0	White	0	<= $75,000, Above Poverty	1.0	1.0	Employed	qufhixun	MSA, Not Principle City	1.0	0.0	wxleyezf	

In [111]:
```
# Time to one-hot encode (get_dummies) the remaining text columns (6); try one column first.
add_race = pd.get_dummies(rawf_all['race'], prefix='race_')
add_race
```

Out[111]:

	race__Black	race__Hispanic	race__Other or Multiple	race__White
0	0	0	0	1
1	0	0	0	1
2	0	0	0	1
3	0	0	0	1
4	0	0	0	1
...	...	...	...	...
26702	0	0	0	1
26703	0	0	0	1
26704	0	0	0	1
26705	0	1	0	0
26706	0	0	0	1

26707 rows × 4 columns

In [112]:
```
rawg_all = pd.concat([rawf_all, add_race], axis=1)
rawg_all.head()
```

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

       race__White
0                1
1                1
2                1
3                1
4                1
...            ...
26702            1
26703            1
26704            1
```

In [113]:
```
rawg_all.head()
```

Out[113]:

from_vacc	opinion_seas_vacc_effective	opinion_seas_risk	opinion_seas_sick_from_vacc	age_group	education	race	sex	income_poverty	marital_status	rent_or_own	employment_status	hhs_geo_region	census_msa	household
1.0	1.0	0.0	1.0	3	0.0	White	0	Below Poverty	0.0	1.0	Not in Labor Force	oxchjgsf	Non-MSA	
3.0	3.0	1.0	3.0	1	1.0	White	1	Below Poverty	0.0	0.0	Employed	bhuqouqj	MSA, Not Principle City	
0.0	3.0	0.0	1.0	0	3.0	White	1	<= $75,000, Above Poverty	0.0	1.0	Employed	qufhixun	MSA, Not Principle City	
4.0	4.0	3.0	0.0	4	1.0	White	0	Below Poverty	0.0	0.0	Not in Labor Force	lrircsnp	MSA, Principle City	
1.0	2.0	0.0	3.0	2	2.0	White	0	<= $75,000, Above Poverty	1.0	1.0	Employed	qufhixun	MSA, Not Principle City	

In [114]:
```
add_INC = pd.get_dummies(rawf_all['income_poverty'], prefix='INC_')
rawg_all = pd.concat([rawf_all, add_INC], axis=1)
rawg_all.head()
```

Out[114]:

nion_seas_sick_from_vacc	age_group	education	race	sex	income_poverty	marital_status	rent_or_own	employment_status	hhs_geo_region	census_msa	household_adults	household_children	employment_industry	h1n1_vac
1.0	3	0.0	White	0	Below Poverty	0.0	1.0	Not in Labor Force	oxchjgsf	Non-MSA	0.0	0.0	NaN	
3.0	1	1.0	White	1	Below Poverty	0.0	0.0	Employed	bhuqouqj	MSA, Not Principle City	0.0	0.0	pxcmvdjn	
1.0	0	3.0	White	1	<= $75,000, Above Poverty	0.0	1.0	Employed	qufhixun	MSA, Not Principle City	2.0	0.0	rucpziij	
0.0	4	1.0	White	0	Below Poverty	0.0	0.0	Not in Labor Force	lrircsnp	MSA, Principle City	0.0	0.0	NaN	
3.0	2	2.0	White	0	<= $75,000, Above Poverty	1.0	1.0	Employed	qufhixun	MSA, Not Principle City	1.0	0.0	wxleyezf	

In [115]:
```
add_EMP = pd.get_dummies(rawf_all['employment_status'], prefix='EMP_')
add_GEO = pd.get_dummies(rawf_all['hhs_geo_region'], prefix='GEO_')
add_MSA = pd.get_dummies(rawf_all['census_msa'], prefix='MSA_')
add_INDUST = pd.get_dummies(rawf_all['employment_industry'], prefix='INDUST_')
```

In [116]:
```
rawh_all = pd.concat([rawf_all, add_race, add_INC, add_EMP, add_GEO, add_MSA, add_INDUST], axis=1)
print(rawh_all.shape)
rawh_all.head()
```

```
(26707, 81)
```

Out[116]:

sa	household_adults	household_children	employment_industry	h1n1_vaccine	seasonal_vaccine	race__Black	race__Hispanic	race__Other or Multiple	race__White	INC__<= $75,000, Above Poverty	INC__> $75,000	INC__Below Poverty	EMP__Employed	EMP__Not in Labor Force	EMP__Unemp
SA	0.0	0.0	NaN	0	0	0	0	0	1	0	0	1	0	1	
lot ity	0.0	0.0	pxcmvdjn	0	1	0	0	0	1	0	0	1	1	0	
lot ity	2.0	0.0	rucpziij	0	0	0	0	0	1	1	0	0	1	0	
A, ity	0.0	0.0	NaN	0	1	0	0	0	1	0	0	1	0	1	
lot ity	1.0	0.0	wxleyezf	0	0	0	0	0	1	1	0	0	1	0	

In [117]:
```
# drop the original columns we just created dummies from (6)
rawi_all = rawh_all.drop(columns=['respondent_id', 'race', 'income_poverty', 'employment_status', 'hhs_geo_region', 'census_msa', 'employment_industry'], axis=1)
rawi_all.shape
```

Out[117]: (26707, 74)

In [118]:
```
rawi_all.head()
```

Out[118]:

nal_vaccine	race__Black	race__Hispanic	race__Other or Multiple	race__White	INC__<= $75,000, Above Poverty	INC__> $75,000	INC__Below Poverty	EMP__Employed	EMP__Not in Labor Force	EMP__Unemployed	GEO__atmpeygn	GEO__bhuqouqj	GEO__dqpwygqj	GEO__fpwskwrf	GEO__kl
0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	
1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	
0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	
1	0	0	0	1	0	0	1	0	1	0	0	0	0	0	
0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	

In [119]:
```
# Need to rename sets of 3 dummy columns created
rawi_all.rename(columns = {'INC__<= $75,000, Above Poverty':'INC__75K_to_Poverty', 'INC__> $75,000':'INC__over_75K',
                           'INC__Below Poverty':'INC__below_Poverty'}, inplace = True)
rawi_all.head()
```

Out[119]:

to_Poverty	INC__over_75K	INC__below_Poverty	EMP__Employed	EMP__Not in Labor Force	EMP__Unemployed	GEO__atmpeygn	GEO__bhuqouqj	GEO__dqpwygqj	GEO__fpwskwrf	GEO__kbazzjca	GEO__lrircsnp	GEO__lzgpxyit	GEO__mlyzmh
0	0	1	0	1	0	0	0	0	0	0	0	0	
0	0	1	1	0	0	0	1	0	0	0	0	0	
1	0	0	1	0	0	0	0	0	0	0	0	0	
0	0	1	0	1	0	0	0	0	0	0	1	0	
1	0	0	1	0	0	0	0	0	0	0	0	0	

In [120]:
```
rawi_all.rename(columns = {'EMP__Not in Labor Force':'EMP__Not_Labor'}, inplace = True)
rawi_all.head()
```

Out[120]:

GEO__kbazzjca	GEO__lrircsnp	GEO__lzgpxyit	GEO__mlyzmhmf	GEO__oxchjgsf	GEO__qufhixun	MSA__MSA, Not Principle City	MSA__MSA, Principle City	MSA__Non-MSA	INDUST__arjwrbjb	INDUST__atmlpfrs	INDUST__cfqqtusy	INDUST__dotnnunm	INDUST__fcx
0	0	0	0	1	0	0	0	1	0	0	0	0	
0	0	0	0	0	0	1	0	0	0	0	0	0	
0	0	0	0	0	1	1	0	0	0	0	0	0	
0	1	0	0	0	0	0	1	0	0	0	0	0	
0	0	0	0	0	1	1	0	0	0	0	0	0	

In [126]:
```
rawi_all.rename(columns = {'MSA__MSA, Not Principle  City':'MSA_Not_Principle_City'}, inplace = True)
rawi_all.head()
```

Out[126]:

_Poverty	EMP__Employed	EMP__Not_Labor	EMP__Unemployed	GEO__atmpeygn	GEO__bhuqouqj	GEO__dqpwygqj	GEO__fpwskwrf	GEO__kbazzjca	GEO__lrircsnp	GEO__lzgpxyit	GEO__mlyzmhmf	GEO__oxchjgsf	GEO__qufhix
1	0	1	0	0	0	0	0	0	0	0	0	1	
1	1	0	0	0	1	0	0	0	0	0	0	0	
0	1	0	0	0	0	0	0	0	0	0	0	0	
1	0	1	0	0	0	0	0	0	1	0	0	0	
0	1	0	0	0	0	0	0	0	0	0	0	0	

In [127]:
```
# save rawi_all which is all numeric but not pre-processed df.
rawi_all.to_csv(r'df2_all_numerical.csv')
```

Preprocessing: Impute (to remove NaNs) and Scale (to normailze values)

In [128]:
```
# KNNImputer
imputer = KNNImputer(n_neighbors=5)
rawj = pd.DataFrame(imputer.fit_transform(rawi_all),columns = rawi_all.columns)
rawj.head()
```

Out[128]:

e_group	education	sex	marital_status	rent_or_own	household_adults	household_children	h1n1_vaccine	seasonal_vaccine	race__Black	race__Hispanic	race__Other or Multiple	race__White	INC__75K_to_Poverty	INC__over_75K	INC__b
3.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	
1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	
0.0	3.0	1.0	0.0	1.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	
4.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	
2.0	2.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	

In [129]:
```
# Cut the y out of the df.
y1 = rawj['h1n1_vaccine']
y2 = rawj['seasonal_vaccine']
print(y1.tail())
y2.head()
```

```
26702    0.0
26703    0.0
26704    0.0
26705    0.0
26706    0.0
Name: h1n1_vaccine, dtype: float64
```

Out[129]:
```
0    0.0
1    1.0
2    0.0
3    1.0
4    0.0
Name: seasonal_vaccine, dtype: float64
```

In [130]:
```
rawk = rawj.drop(columns=['h1n1_vaccine', 'seasonal_vaccine'], axis=1)
rawk.head()
```

Out[130]:

	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
0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0
1	3.0	2.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0
2	1.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0
4	2.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0

In [131]:
```
# Scaler - for this phase
scale = StandardScaler()
rawm = pd.DataFrame(scale.fit_transform(rawk),columns = rawk.columns)
rawm.head()
```

Out[131]:

	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
0	-0.679477	-2.043782	-0.22717	-1.627773	-0.272472	-2.177275	-0.748761	1.402512	0.692333	-0.538812
1	1.519841	1.196063	-0.22717	0.618403	-0.272472	0.459797	-0.748761	1.402512	0.692333	-0.538812
2	-0.679477	-0.423859	-0.22717	0.618403	-0.272472	-2.177275	-0.748761	-0.714605	-1.450210	-0.538812
3	-0.679477	-0.423859	-0.22717	0.618403	-0.272472	0.459797	1.338625	-0.714605	-1.450210	-0.538812
4	0.420182	-0.423859	-0.22717	0.618403	-0.272472	0.459797	1.338625	-0.714605	0.692333	-0.538812

In [132]:
```
rawm.shape
```

Out[132]:
```
(26707, 72)
```

In [133]:
```
# save df as csv file for later use - modeling - imputed and scaled
rawm.to_csv(r'df2_imputed_scaled.csv')
```

Create Train-Test Split

In [134]:
```
# Need to split data into X and y dataframes - for h1n1
X = rawm
print(y1.shape)
print(X.shape)
```

```
(26707,)
(26707, 72)
```

In [135]:
```
# Create train and test sets.
X_train, X_test, y_train, y_test = train_test_split(X, y1, test_size=0.30, random_state=36)
print(X_train.shape)
print(X_test.shape)

print(y_train.shape)
print(y_test.shape)
```

```
(18694, 72)
(8013, 72)
(18694,)
(8013,)
```

Run the models - the new B-data prep versions

In [ ]:

In [136]:
```
def run_model (model):
    # Instantiate classification model
    cfmodel = model
    # Fit the classifier
    cfmodel.fit(X_train, y_train)

    # Predict on training and test sets
    training_preds = cfmodel.predict(X_train)
    test_preds = cfmodel.predict(X_test)

    # Get detailed results (Train and Test)
    print('------------------------------------------------------------------------------')
    print(f'MODEL: {model}')
    # Classification Report
    print('------------------------------------------------------------------------------')
    print('Classification Report - TRAIN')
    print('------------------------------------------------------------------------------')
    print(classification_report(y_train, training_preds))
    # Confusion Matrix
    print('------------------------------------------------------------------------------')
    print('Confusion Matrix - TRAIN')
    print('------------------------------------------------------------------------------')
    print(pd.crosstab(y_train, training_preds, rownames=['True'], colnames=['Predicted'], margins=True))
    print('\n------------------------------------------------------------------------------')
    # Classification Report
    print('------------------------------------------------------------------------------')
    print('Classification Report - TEST')
    print('------------------------------------------------------------------------------')
    print(classification_report(y_test, test_preds))
    # Confusion Matrix
    print('------------------------------------------------------------------------------')
    print('Confusion Matrix - TEST')
    print('------------------------------------------------------------------------------')
    print(pd.crosstab(y_test, test_preds, rownames=['True'], colnames=['Predicted'], margins=True))
    print('------------------------------------------------------------------------------')
```

In [137]:
```
# Assign the model... change this for each model to run.
model = DecisionTreeClassifier()
run_model(model)
```

```
------------------------------------------------------------------------------
MODEL: DecisionTreeClassifier()
------------------------------------------------------------------------------
Classification Report - TRAIN
------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     14754
         1.0       1.00      1.00      1.00      3940

    accuracy                           1.00     18694
   macro avg       1.00      1.00      1.00     18694
weighted avg       1.00      1.00      1.00     18694

------------------------------------------------------------------------------
Confusion Matrix - TRAIN
------------------------------------------------------------------------------
Predicted    0.0   1.0    All
True
0.0        14754     0  14754
1.0            0  3940   3940
All        14754  3940  18694

------------------------------------------------------------------------------
------------------------------------------------------------------------------
Classification Report - TEST
------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.85      0.84      0.85      6279
         1.0       0.45      0.47      0.46      1734

    accuracy                           0.76      8013
   macro avg       0.65      0.66      0.65      8013
weighted avg       0.77      0.76      0.76      8013

------------------------------------------------------------------------------
Confusion Matrix - TEST
------------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        5289   990  6279
1.0         920   814  1734
All        6209  1804  8013
------------------------------------------------------------------------------
```

In [143]:
```
# Assign the model... change this for each model to run.
model = RandomForestClassifier()
run_model(model)
```

```
------------------------------------------------------------------------
MODEL: RandomForestClassifier()
------------------------------------------------------------------------
Classification Report - TRAIN
------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     14754
         1.0       1.00      1.00      1.00      3940

    accuracy                           1.00     18694
   macro avg       1.00      1.00      1.00     18694
weighted avg       1.00      1.00      1.00     18694

------------------------------------------------------------------------
Confusion Matrix - TRAIN
------------------------------------------------------------------------
Predicted    0.0   1.0    All
True
0.0        14754     0  14754
1.0            0  3940   3940
All        14754  3940  18694

------------------------------------------------------------------------
------------------------------------------------------------------------
Classification Report - TEST
------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.85      0.96      0.90      6279
         1.0       0.73      0.39      0.51      1734

    accuracy                           0.84      8013
   macro avg       0.79      0.67      0.70      8013
weighted avg       0.82      0.84      0.82      8013

------------------------------------------------------------------------
Confusion Matrix - TEST
------------------------------------------------------------------------
Predicted   0.0  1.0   All
True
0.0        6028  251  6279
1.0        1061  673  1734
All        7089  924  8013
------------------------------------------------------------------------
```

In [145]:
```
# Take a look at feature importances (RandomForest) - from data B model

importance = pd.DataFrame(data={'features': X_train.columns, 'importance': model.feature_importances_})
importance = importance.sort_values('importance', ascending=False)
importance = importance.reset_index()
importance.drop('index', axis=1, inplace=True)
importance.head(40)
```

Out[145]:

	features	importance
0	doctor_recc_h1n1	0.096910
1	opinion_h1n1_risk	0.068652
2	opinion_h1n1_vacc_effective	0.058505
3	opinion_seas_risk	0.042475
4	age_group	0.034154
5	opinion_h1n1_sick_from_vacc	0.030539
6	opinion_seas_vacc_effective	0.030010
7	education	0.030009
8	h1n1_concern	0.028942
9	doctor_recc_seasonal	0.028576
10	opinion_seas_sick_from_vacc	0.028254
11	household_adults	0.023907
12	h1n1_knowledge	0.022021
13	health_insurance	0.020749
14	household_children	0.020612
15	health_worker	0.017638
16	marital_status	0.016116
17	sex	0.015494
18	chronic_med_condition	0.015339
19	rent_or_own	0.014601
20	behavioral_large_gatherings	0.014242
21	MSA_Not_Principle_City	0.013936
22	INC__75K_to_Poverty	0.013912
23	behavioral_outside_home	0.013744
24	behavioral_touch_face	0.013462
25	MSA_Principle_City	0.012943
26	behavioral_avoidance	0.012774
27	MSA_Not	0.012473
28	INC__over_75K	0.011456
29	INDUST__fcxhlnwr	0.010671
30	GEO__lzgpxyit	0.010458
31	GEO__bhuqouqj	0.010173
32	EMP__Not_Labor	0.010062
33	EMP__Employed	0.009984
34	GEO__qufhixun	0.009970
35	child_under_6_months	0.009704
36	GEO__fpwskwrf	0.009587
37	race__White	0.009504
38	GEO__oxchjgsf	0.009459
39	GEO__kbazzjca	0.009311

In [139]:
```
# Assign the model... change this for each model to run.
model = KNeighborsClassifier()
run_model(model)
```

```
------------------------------------------------------------------------
MODEL: KNeighborsClassifier()
------------------------------------------------------------------------
Classification Report - TRAIN
------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.86      0.97      0.91     14754
         1.0       0.77      0.43      0.55      3940

    accuracy                           0.85     18694
   macro avg       0.81      0.70      0.73     18694
weighted avg       0.84      0.85      0.84     18694

------------------------------------------------------------------------
Confusion Matrix - TRAIN
------------------------------------------------------------------------
Predicted    0.0   1.0    All
True
0.0        14239   515  14754
1.0         2252  1688   3940
All        16491  2203  18694

------------------------------------------------------------------------
------------------------------------------------------------------------
Classification Report - TEST
------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.82      0.95      0.88      6279
         1.0       0.59      0.27      0.37      1734

    accuracy                           0.80      8013
   macro avg       0.71      0.61      0.63      8013
weighted avg       0.77      0.80      0.77      8013

------------------------------------------------------------------------
Confusion Matrix - TEST
------------------------------------------------------------------------
Predicted   0.0  1.0   All
True
0.0        5959  320  6279
1.0        1267  467  1734
All        7226  787  8013
------------------------------------------------------------------------
```

In [140]:
```
# Assign the model... change this for each model to run.
model = LogisticRegression()
run_model(model)
```

```
------------------------------------------------------------------------
MODEL: LogisticRegression()
------------------------------------------------------------------------
Classification Report - TRAIN
------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.86      0.95      0.90     14754
         1.0       0.69      0.43      0.53      3940

    accuracy                           0.84     18694
   macro avg       0.77      0.69      0.72     18694
weighted avg       0.82      0.84      0.82     18694

------------------------------------------------------------------------
Confusion Matrix - TRAIN
------------------------------------------------------------------------
Predicted    0.0   1.0    All
True
0.0        13975   779  14754
1.0         2241  1699   3940
All        16216  2478  18694

------------------------------------------------------------------------
------------------------------------------------------------------------
Classification Report - TEST
------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.86      0.95      0.90      6279
         1.0       0.69      0.42      0.52      1734

    accuracy                           0.83      8013
   macro avg       0.77      0.68      0.71      8013
weighted avg       0.82      0.83      0.82      8013

------------------------------------------------------------------------
Confusion Matrix - TEST
------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        5947   332  6279
1.0        1007   727  1734
All        6954  1059  8013
------------------------------------------------------------------------
```

In [141]:
```
# Assign the model... change this for each model to run.
model = XGBClassifier()
run_model(model)
```

```
--------------------------------------------------------------------------
MODEL: XGBClassifier()
--------------------------------------------------------------------------
Classification Report - TRAIN
--------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.87      0.95      0.91     14754
         1.0       0.72      0.46      0.56      3940

    accuracy                           0.85     18694
   macro avg       0.80      0.71      0.74     18694
weighted avg       0.84      0.85      0.84     18694

--------------------------------------------------------------------------
Confusion Matrix - TRAIN
--------------------------------------------------------------------------
Predicted    0.0   1.0    All
True
0.0        14050   704  14754
1.0         2114  1826   3940
All        16164  2530  18694

--------------------------------------------------------------------------
--------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.86      0.95      0.90      6279
         1.0       0.71      0.44      0.54      1734

    accuracy                           0.84      8013
   macro avg       0.79      0.70      0.72      8013
weighted avg       0.83      0.84      0.83      8013

--------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        5969   310  6279
1.0         970   764  1734
All        6939  1074  8013
--------------------------------------------------------------------------
```

In [142]:
```
# Assign the model... change this for each model to run.
model = SVC()
run_model(model)
```

```
--------------------------------------------------------------------------
MODEL: SVC()
--------------------------------------------------------------------------
Classification Report - TRAIN
--------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.88      0.97      0.92     14754
         1.0       0.81      0.51      0.62      3940

    accuracy                           0.87     18694
   macro avg       0.85      0.74      0.77     18694
weighted avg       0.87      0.87      0.86     18694

--------------------------------------------------------------------------
Confusion Matrix - TRAIN
--------------------------------------------------------------------------
Predicted    0.0   1.0    All
True
0.0        14289   465  14754
1.0         1942  1998   3940
All        16231  2463  18694

--------------------------------------------------------------------------
--------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.85      0.95      0.90      6279
         1.0       0.70      0.40      0.51      1734

    accuracy                           0.83      8013
   macro avg       0.78      0.68      0.71      8013
weighted avg       0.82      0.83      0.82      8013

--------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------
Predicted   0.0  1.0   All
True
0.0        5984  295  6279
1.0        1036  698  1734
All        7020  993  8013
--------------------------------------------------------------------------
```

Observations on the 6 models via data prep B

The results were not that different from the models run with approach A to data prep. XGBoost and Random Forest models had the best accuracy at 0.84, Random Forest ha
better precission for class 1 with 0.72 (versus 0.71 for XGBoost). Accuracy was about the same, but the precision improved slightly with data prep B - for Random Fore
0.68 to 0.72. See summary table of results.

In [ ]:

Run subset of models - the new B-data prep versions - for SEASONAL vacc

In [146]:
```
# Create train and test sets.
X_train, X_test, y_train, y_test = train_test_split(X, y2, test_size=0.30, random_state=36)
print(X_train.shape)
print(X_test.shape)

print(y_train.shape)
print(y_test.shape)
```

```
(18694, 72)
(8013, 72)
(18694,)
(8013,)
```

In [147]:
```
def run_model (model):
    # Instantiate classification model
    cfmodel = model
    # Fit the classifier
    cfmodel.fit(X_train, y_train)

    # Predict on training and test sets
    training_preds = cfmodel.predict(X_train)
    test_preds = cfmodel.predict(X_test)

    # Get detailed results (Train and Test)
    print('------------------------------------------------------------------------')
    print(f'MODEL: {model}')
    # Classification Report
    print('------------------------------------------------------------------------')
    print('Classification Report - TRAIN')
    print('------------------------------------------------------------------------')
    print(classification_report(y_train, training_preds))
    # Confusion Matrix
    print('------------------------------------------------------------------------')
    print('Confusion Matrix - TRAIN')
    print('------------------------------------------------------------------------')
    print(pd.crosstab(y_train, training_preds, rownames=['True'], colnames=['Predicted'], margins=True))
    print('\n------------------------------------------------------------------------')
    # Classification Report
    print('------------------------------------------------------------------------')
    print('Classification Report - TEST')
    print('------------------------------------------------------------------------')
    print(classification_report(y_test, test_preds))
    # Confusion Matrix
    print('------------------------------------------------------------------------')
    print('Confusion Matrix - TEST')
    print('------------------------------------------------------------------------')
    print(pd.crosstab(y_test, test_preds, rownames=['True'], colnames=['Predicted'], margins=True))
    print('------------------------------------------------------------------------')
```

In [148]:
```
# Assign the model... change this for each model to run.
model = XGBClassifier()
run_model(model)
```

```
------------------------------------------------------------------------
MODEL: XGBClassifier()
------------------------------------------------------------------------
Classification Report - TRAIN
------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.80      0.82      0.81     10004
         1.0       0.79      0.76      0.78      8690

    accuracy                           0.80     18694
   macro avg       0.80      0.79      0.79     18694
weighted avg       0.80      0.80      0.80     18694

------------------------------------------------------------------------
Confusion Matrix - TRAIN
------------------------------------------------------------------------
Predicted    0.0   1.0    All
True
0.0         8246  1758  10004
1.0         2047  6643   8690
All        10293  8401  18694

------------------------------------------------------------------------
------------------------------------------------------------------------
Classification Report - TEST
------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.79      0.82      0.80      4268
         1.0       0.78      0.75      0.77      3745

    accuracy                           0.79      8013
   macro avg       0.79      0.79      0.79      8013
weighted avg       0.79      0.79      0.79      8013

------------------------------------------------------------------------
Confusion Matrix - TEST
------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        3491   777  4268
1.0         923  2822  3745
All        4414  3599  8013
------------------------------------------------------------------------
```

In [149]:
```
model = SVC()
run_model(model)
```

```
------------------------------------------------------------------------
MODEL: SVC()
------------------------------------------------------------------------
Classification Report - TRAIN
------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.85      0.87      0.86     10004
         1.0       0.85      0.82      0.83      8690

    accuracy                           0.85     18694
   macro avg       0.85      0.84      0.84     18694
weighted avg       0.85      0.85      0.85     18694

------------------------------------------------------------------------
Confusion Matrix - TRAIN
------------------------------------------------------------------------
Predicted    0.0   1.0    All
True
0.0         8700  1304  10004
1.0         1576  7114   8690
All        10276  8418  18694

------------------------------------------------------------------------
------------------------------------------------------------------------
Classification Report - TEST
------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.78      0.81      0.79      4268
         1.0       0.77      0.74      0.76      3745

    accuracy                           0.78      8013
   macro avg       0.78      0.77      0.77      8013
weighted avg       0.78      0.78      0.78      8013

------------------------------------------------------------------------
Confusion Matrix - TEST
------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        3451   817  4268
1.0         977  2768  3745
All        4428  3585  8013
------------------------------------------------------------------------
```

In [150]:
```
# Assign the model... change this for each model to run.
model = RandomForestClassifier()
run_model(model)
```

```
------------------------------------------------------------------------
MODEL: RandomForestClassifier()
------------------------------------------------------------------------
Classification Report - TRAIN
------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     10004
         1.0       1.00      1.00      1.00      8690

    accuracy                           1.00     18694
   macro avg       1.00      1.00      1.00     18694
weighted avg       1.00      1.00      1.00     18694

------------------------------------------------------------------------
Confusion Matrix - TRAIN
------------------------------------------------------------------------
Predicted    0.0   1.0    All
True
0.0        10004     0  10004
1.0            0  8690   8690
All        10004  8690  18694

------------------------------------------------------------------------
------------------------------------------------------------------------
Classification Report - TEST
------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.78      0.81      0.80      4268
         1.0       0.78      0.74      0.76      3745

    accuracy                           0.78      8013
   macro avg       0.78      0.78      0.78      8013
weighted avg       0.78      0.78      0.78      8013

------------------------------------------------------------------------
Confusion Matrix - TEST
------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        3476   792  4268
1.0         955  2790  3745
All        4431  3582  8013
------------------------------------------------------------------------
```

In [151]:
```
# Take a look at feature importances (RandomForest) - from data B model - SEASONAL

importance = pd.DataFrame(data={'features': X_train.columns, 'importance': model.feature_importances_})
importance = importance.sort_values('importance', ascending=False)
importance = importance.reset_index()
importance.drop('index', axis=1, inplace=True)
importance.head(40)
```

Out[151]:

	features	importance
0	opinion_seas_risk	0.088941
1	opinion_seas_vacc_effective	0.080817
2	doctor_recc_seasonal	0.077480
3	age_group	0.057590
4	opinion_h1n1_risk	0.034100
5	opinion_h1n1_vacc_effective	0.033194
6	health_insurance	0.031792
7	opinion_seas_sick_from_vacc	0.029273
8	education	0.028802
9	h1n1_concern	0.027067
10	opinion_h1n1_sick_from_vacc	0.026804
11	household_adults	0.021780
12	doctor_recc_h1n1	0.021421
13	h1n1_knowledge	0.020267
14	household_children	0.019361
15	chronic_med_condition	0.018175
16	rent_or_own	0.014809
17	marital_status	0.014665
18	sex	0.014240
19	behavioral_touch_face	0.013454
20	behavioral_large_gatherings	0.012610
21	MSA_Not_Principle_City	0.012596
22	INC__75K_to_Poverty	0.012416
23	behavioral_outside_home	0.012311
24	behavioral_avoidance	0.011956
25	health_worker	0.011783
26	MSA_Principle_City	0.011652
27	EMP__Not_Labor	0.011152
28	MSA_Not	0.010979
29	INC__over_75K	0.010138
30	race__White	0.009888
31	GEO__lzgpxyit	0.009695
32	behavioral_wash_hands	0.009397
33	GEO__fpwskwrf	0.008947
34	GEO__bhuqouqj	0.008850
35	EMP__Employed	0.008847
36	INDUST__fcxhlnwr	0.008612
37	GEO__qufhixun	0.008492
38	GEO__oxchjgsf	0.008207
39	GEO__kbazzjca	0.008148

Observations on models run on the seasonal_vaccine target variable (data-prep B)

Only a small improvement over data-prep A. Best performance was with XGBoost at accuracy of 0.79 and precission of 0.78 (class 1).

In [ ]:

Run some models for H1N1 with SMOTE and B Data Prep Set

In [152]:
```
# Create train and test sets.
X_train, X_test, y_train, y_test = train_test_split(X, y1, test_size=0.30, random_state=36)
print(X_train.shape)
print(X_test.shape)

print(y_train.shape)
print(y_test.shape)
```

```
(18694, 72)
(8013, 72)
(18694,)
(8013,)
```

In [153]:
```
# Address the target class imbalance with SMOTE

print("Before OverSampling, counts of label '0': {}".format(sum(y_train == 0)))
print("Before OverSampling, counts of label '1': {}".format(sum(y_train == 1)))

sm = SMOTE(random_state = 4)
X_train_s, y_train_s = sm.fit_sample(X_train, y_train)

print('After OverSampling, the shape of train_X: {}'.format(X_train_s.shape))
print('After OverSampling, the shape of train_y: {} \n'.format(y_train_s.shape))

print("After OverSampling, counts of label '0': {}".format(sum(y_train_s == 0)))
print("After OverSampling, counts of label '1': {}".format(sum(y_train_s == 1)))
```

```
Before OverSampling, counts of label '0': 14754
Before OverSampling, counts of label '1': 3940
After OverSampling, the shape of train_X: (29508, 72)
After OverSampling, the shape of train_y: (29508,)

After OverSampling, counts of label '0': 14754
After OverSampling, counts of label '1': 14754
```

In [154]:
```
def run_smodel (model):
    # Instantiate classification model
    smodel = model
    # Fit the classifier
    smodel.fit(X_train_s, y_train_s)

    # Predict on training and test sets
    training_preds = smodel.predict(X_train_s)
    test_preds = smodel.predict(X_test)

    # Get detailed results (Train and Test)
    print('------------------------------------------------------------------------------')
    print(f'MODEL: {model}')
    # Classification Report
    print('------------------------------------------------------------------------------')
    print('Classification Report - TRAIN')
    print('------------------------------------------------------------------------------')
    print(classification_report(y_train_s, training_preds))
    # Confusion Matrix
    print('------------------------------------------------------------------------------')
    print('Confusion Matrix - TRAIN')
    print('------------------------------------------------------------------------------')
    print(pd.crosstab(y_train_s, training_preds, rownames=['True'], colnames=['Predicted'], margins=True))
    print('\n------------------------------------------------------------------------------')
    # Classification Report
    print('------------------------------------------------------------------------------')
    print('Classification Report - TEST')
    print('------------------------------------------------------------------------------')
    print(classification_report(y_test, test_preds))
    # Confusion Matrix
    print('------------------------------------------------------------------------------')
    print('Confusion Matrix - TEST')
    print('------------------------------------------------------------------------------')
    print(pd.crosstab(y_test, test_preds, rownames=['True'], colnames=['Predicted'], margins=True))
    print('------------------------------------------------------------------------------')
```

In [155]:
```
# Assign the model... change this for each model to run.
model = SVC()
run_smodel(model)
```

```
------------------------------------------------------------------------------
MODEL: SVC()
------------------------------------------------------------------------------
Classification Report - TRAIN
------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.91      0.91      0.91     14754
         1.0       0.91      0.91      0.91     14754

    accuracy                           0.91     29508
   macro avg       0.91      0.91      0.91     29508
weighted avg       0.91      0.91      0.91     29508

------------------------------------------------------------------------------
Confusion Matrix - TRAIN
------------------------------------------------------------------------------
Predicted    0.0    1.0    All
True
0.0        13368   1386  14754
1.0         1259  13495  14754
All        14627  14881  29508

------------------------------------------------------------------------------
------------------------------------------------------------------------------
Classification Report - TEST
------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.88      0.88      0.88      6279
         1.0       0.57      0.58      0.58      1734

    accuracy                           0.81      8013
   macro avg       0.73      0.73      0.73      8013
weighted avg       0.82      0.81      0.82      8013

------------------------------------------------------------------------------
Confusion Matrix - TEST
------------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        5514   765  6279
1.0         723  1011  1734
All        6237  1776  8013
------------------------------------------------------------------------------
```

In [156]:
```
model = XGBClassifier()
run_smodel(model)
```

```
--------------------------------------------------------------------------------
MODEL: XGBClassifier()
--------------------------------------------------------------------------------
Classification Report - TRAIN
--------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.88      0.92      0.90     14754
         1.0       0.92      0.87      0.89     14754

    accuracy                           0.90     29508
   macro avg       0.90      0.90      0.90     29508
weighted avg       0.90      0.90      0.90     29508

--------------------------------------------------------------------------------
Confusion Matrix - TRAIN
--------------------------------------------------------------------------------
Predicted    0.0    1.0    All
True
0.0        13632   1122  14754
1.0         1925  12829  14754
All        15557  13951  29508

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.87      0.92      0.90      6279
         1.0       0.65      0.51      0.57      1734

    accuracy                           0.83      8013
   macro avg       0.76      0.71      0.73      8013
weighted avg       0.82      0.83      0.83      8013

--------------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        5799   480  6279
1.0         856   878  1734
All        6655  1358  8013
--------------------------------------------------------------------------------
```

In [157]:
```
model = RandomForestClassifier()
run_smodel (model)
```

```
--------------------------------------------------------------------------------
MODEL: RandomForestClassifier()
--------------------------------------------------------------------------------
Classification Report - TRAIN
--------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     14754
         1.0       1.00      1.00      1.00     14754

    accuracy                           1.00     29508
   macro avg       1.00      1.00      1.00     29508
weighted avg       1.00      1.00      1.00     29508

--------------------------------------------------------------------------------
Confusion Matrix - TRAIN
--------------------------------------------------------------------------------
Predicted    0.0    1.0    All
True
0.0        14754      0  14754
1.0            0  14754  14754
All        14754  14754  29508

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.86      0.94      0.90      6279
         1.0       0.67      0.45      0.53      1734

    accuracy                           0.83      8013
   macro avg       0.76      0.69      0.72      8013
weighted avg       0.82      0.83      0.82      8013

--------------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        5895   384  6279
1.0         961   773  1734
All        6856  1157  8013
--------------------------------------------------------------------------------
```

In [158]:
```
# Take a look at feature importances (RandomForest) - from data B model - SEASONAL

importance = pd.DataFrame(data={'features': X_train_s.columns, 'importance': model.feature_importances_})
importance = importance.sort_values('importance', ascending=False)
importance = importance.reset_index()
importance.drop('index', axis=1, inplace=True)
importance.head(20)
```

Out[158]:

	features	importance
0	doctor_recc_h1n1	0.106653
1	opinion_h1n1_vacc_effective	0.093445
2	opinion_h1n1_risk	0.085158
3	opinion_seas_vacc_effective	0.066370
4	opinion_seas_risk	0.062130
5	h1n1_knowledge	0.039483
6	doctor_recc_seasonal	0.036241
7	h1n1_concern	0.035996
8	age_group	0.027195
9	opinion_h1n1_sick_from_vacc	0.026153
10	opinion_seas_sick_from_vacc	0.025804
11	education	0.024914
12	health_insurance	0.023276
13	household_adults	0.018192
14	sex	0.015575
15	marital_status	0.015235
16	chronic_med_condition	0.014388
17	behavioral_touch_face	0.014125
18	household_children	0.013996
19	INC__75K_to_Poverty	0.013153

Observations on the SMOTED version of data prep B.

I ran 3 models and wsa seeing that the results were not any better than without SMOTE so decided not to persue this further. Although the accuracy was only slightly w
0.1 or so, the precission was worse, off by 0.5 or so. This indicates that perhaps the non-SMOTED versions of the models were overtraining a bit on the majority class

In [ ]:

Sidebar

Trying some Feature Engineering: try and create 3 new columns

a) Contact
b) Equity
c) Concerned

NOTE: Initially I tried creating variables that were too specific... and very few respondents fit all of the criteria, so did not use these variables. Ultimatly I ran
to persue this further.

In [159]:
```
df_7.head()
```

Out[159]:

	Unnamed: 0	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavioral_avoidance	behavioral_face_mask	behavioral_wash_hands	behavioral_large_gatherings	behavioral_outside_home	behavioral_touch_face	docto
0	0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	
1	1	3.0	2.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	
2	2	1.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	
3	3	1.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	
4	4	2.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	

In [161]:
```
df_8 = df_7
```

In [162]:
```
# Attenpt one
conditions = [(df_7['household_adults'] == 0) & (df_7['household_children'] == 0) & (df_7['child_under_6_months'] == 0) & (df_7['marital_status'] == 0)]
values = ['low']
df_8['contact_people'] = np.select(conditions, values)
df_8.head()
```

Out[162]:

_vacc	age_group	education	race	sex	income_poverty	marital_status	rent_or_own	employment_status	hhs_geo_region	census_msa	household_adults	household_children	employment_industry	h1n1_vaccine	seasonal_vaccii
2.0	3.0	0.0	3.0	0.0	0.0	0.0	1.0	1.0	8.0	2.0	0.0	0.0	NaN	0	
4.0	1.0	1.0	3.0	1.0	0.0	0.0	0.0	2.0	1.0	0.0	0.0	0.0	3.0	0	
2.0	0.0	3.0	3.0	1.0	1.0	0.0	1.0	2.0	9.0	0.0	2.0	0.0	9.0	0	
1.0	4.0	1.0	3.0	0.0	0.0	0.0	0.0	1.0	5.0	1.0	0.0	0.0	NaN	0	
4.0	2.0	2.0	3.0	0.0	1.0	1.0	1.0	2.0	9.0	0.0	1.0	0.0	1.0	0	

In [163]:
```
df_8.tail()
```

Out[163]:

_vacc	age_group	education	race	sex	income_poverty	marital_status	rent_or_own	employment_status	hhs_geo_region	census_msa	household_adults	household_children	employment_industry	h1n1_vaccine	seasonal_vaccii
2.0	4.0	2.0	3.0	0.0	1.0	0.0	1.0	1.0	9.0	2.0	0.0	0.0	NaN	0	
1.0	0.0	3.0	3.0	1.0	1.0	0.0	0.0	2.0	6.0	1.0	1.0	0.0	0.0	0	
2.0	3.0	2.0	3.0	0.0	NaN	0.0	1.0	NaN	6.0	0.0	0.0	0.0	NaN	0	
2.0	0.0	2.0	1.0	0.0	1.0	1.0	0.0	2.0	5.0	2.0	1.0	0.0	0.0	0	
1.0	4.0	2.0	3.0	1.0	1.0	1.0	1.0	1.0	7.0	1.0	1.0	0.0	NaN	0	

In [164]:
```
# Attenpt one-b
conditions = [(df_7['health_worker'] == 0) & (df_7['behavioral_avoidance'] == 1) & (df_7['behavioral_large_gatherings'] == 1) & (df_7['behavioral_outside_home'] == 1)
values = ['low']
df_8['contact_behavior'] = np.select(conditions, values)
df_8.head()
```

Out[164]:

education	race	sex	income_poverty	marital_status	rent_or_own	employment_status	hhs_geo_region	census_msa	household_adults	household_children	employment_industry	h1n1_vaccine	seasonal_vaccine	contact_people
0.0	3.0	0.0	0.0	0.0	1.0	1.0	8.0	2.0	0.0	0.0	NaN	0	0	low
1.0	3.0	1.0	0.0	0.0	0.0	2.0	1.0	0.0	0.0	0.0	3.0	0	1	low
3.0	3.0	1.0	1.0	0.0	1.0	2.0	9.0	0.0	2.0	0.0	9.0	0	0	0
1.0	3.0	0.0	0.0	0.0	0.0	1.0	5.0	1.0	0.0	0.0	NaN	0	1	low
2.0	3.0	0.0	1.0	1.0	1.0	2.0	9.0	0.0	1.0	0.0	1.0	0	0	0

In [165]:
```
df_8.tail()
```

Out[165]:

education	race	sex	income_poverty	marital_status	rent_or_own	employment_status	hhs_geo_region	census_msa	household_adults	household_children	employment_industry	h1n1_vaccine	seasonal_vaccine	contact_people
2.0	3.0	0.0	1.0	0.0	1.0	1.0	9.0	2.0	0.0	0.0	NaN	0	0	low
3.0	3.0	1.0	1.0	0.0	0.0	2.0	6.0	1.0	1.0	0.0	0.0	0	0	0
2.0	3.0	0.0	NaN	0.0	1.0	NaN	6.0	0.0	0.0	0.0	NaN	0	1	low
2.0	1.0	0.0	1.0	1.0	0.0	2.0	5.0	2.0	1.0	0.0	0.0	0	0	0
2.0	3.0	1.0	1.0	1.0	1.0	1.0	7.0	1.0	1.0	0.0	NaN	0	0	0

In [166]:
```
# Attenpt one-c
conditions = [(df_8['contact_people'] == 'low') & (df_8['contact_behavior'] == 'low')]
values = [1]
df_8['contact_total'] = np.select(conditions, values)
df_8.head()
```

Out[166]:

inion_h1n1_vacc_effective	opinion_h1n1_risk	opinion_h1n1_sick_from_vacc	opinion_seas_vacc_effective	opinion_seas_risk	opinion_seas_sick_from_vacc	age_group	education	race	sex	income_poverty	marital_status	rent_or_ow
3.0	1.0	2.0	2.0	1.0	2.0	3.0	0.0	3.0	0.0	0.0	0.0	1
5.0	4.0	4.0	4.0	2.0	4.0	1.0	1.0	3.0	1.0	0.0	0.0	0
3.0	1.0	1.0	4.0	1.0	2.0	0.0	3.0	3.0	1.0	1.0	0.0	1
3.0	3.0	5.0	5.0	4.0	1.0	4.0	1.0	3.0	0.0	0.0	0.0	0
3.0	3.0	2.0	3.0	1.0	4.0	2.0	2.0	3.0	0.0	1.0	1.0	1

In [167]:
```
df_8['contact_total'].value_counts()
```

Out[167]:
```
0    25381
1     1326
Name: contact_total, dtype: int64
```

In [168]:
```
df_8['contact_behavior'].value_counts()
```

Out[168]:
```
0      21515
low     5192
Name: contact_behavior, dtype: int64
```

In [169]:
```
df_8['contact_people'].value_counts()
```

Out[169]:
```
0      20682
low     6025
Name: contact_people, dtype: int64
```

In [ ]:
```
# Need to convert some values to 1 instead of low
```

In [170]:
```
df_9 = df_8
```

In [171]:
```
# Attenpt two-a
conditions = [(df_8['h1n1_concern'] == 0) & (df_8['h1n1_knowledge'] == 0) & (df_8['opinion_h1n1_risk'] == 0) & (df_8['opinion_h1n1_vacc_effective'] == 0) & (df_8['opi
values = ['low']
df_9['Concern_level'] = np.select(conditions, values)
df_9.head()
```

Out[171]:

_poverty	marital_status	rent_or_own	employment_status	hhs_geo_region	census_msa	household_adults	household_children	employment_industry	h1n1_vaccine	seasonal_vaccine	contact_people	contact_behavior	contact_to
0.0	0.0	1.0	1.0	8.0	2.0	0.0	0.0	NaN	0	0	low	0	
0.0	0.0	0.0	2.0	1.0	0.0	0.0	0.0	3.0	0	1	low	0	
1.0	0.0	1.0	2.0	9.0	0.0	2.0	0.0	9.0	0	0	0	0	
0.0	0.0	0.0	1.0	5.0	1.0	0.0	0.0	NaN	0	1	low	0	
1.0	1.0	1.0	2.0	9.0	0.0	1.0	0.0	1.0	0	0	0	0	

In [172]:
```
df_9['Concern_level'].value_counts()
```

Out[172]:
```
0    26707
Name: Concern_level, dtype: int64
```

In [173]:
```
# Attenpt two-b
conditions = [(df_8['h1n1_concern'] == 3) & (df_8['h1n1_knowledge'] == 2) & (df_8['opinion_h1n1_risk'] == 4) & (df_8['opinion_h1n1_vacc_effective'] == 4) & (df_8['opi
values = ['high']
df_9['Concern_level'] = np.select(conditions, values)
df_9.head()
```

Out[173]:

pinion_seas_sick_from_vacc	age_group	education	race	sex	income_poverty	marital_status	rent_or_own	employment_status	hhs_geo_region	census_msa	household_adults	household_children	employment_industry	h1n1_va
2.0	3.0	0.0	3.0	0.0	0.0	0.0	1.0	1.0	8.0	2.0	0.0	0.0	NaN	
4.0	1.0	1.0	3.0	1.0	0.0	0.0	0.0	2.0	1.0	0.0	0.0	0.0	3.0	
2.0	0.0	3.0	3.0	1.0	1.0	0.0	1.0	2.0	9.0	0.0	2.0	0.0	9.0	
1.0	4.0	1.0	3.0	0.0	0.0	0.0	0.0	1.0	5.0	1.0	0.0	0.0	NaN	
4.0	2.0	2.0	3.0	0.0	1.0	1.0	1.0	2.0	9.0	0.0	1.0	0.0	1.0	

In [174]:
```
df_9['Concern_level'].value_counts()
```

Out[174]:
```
0       26630
high       77
Name: Concern_level, dtype: int64
```

In [178]:
```
df_9['race'].value_counts()
```

Out[178]:
```
3.0    21222
0.0     2118
1.0     1755
2.0     1612
Name: race, dtype: int64
```

In [179]:
```
# Attenpt three
conditions = [(df_8['census_msa'] == 1) & (df_8['health_insurance'] == 0) & (df_8['income_poverty'] == 0) & (df_8['employment_status'] == 0) & (df_8['rent_or_own'] ==
values = ['low']
df_9['Equity_level'] = np.select(conditions, values)
df_9.head()
```

Out[179]:

ırital_status	rent_or_own	employment_status	hhs_geo_region	census_msa	household_adults	household_children	employment_industry	h1n1_vaccine	seasonal_vaccine	contact_people	contact_behavior	contact_total	Concern
0.0	1.0	1.0	8.0	2.0	0.0	0.0	NaN	0	0	low	0	0	
0.0	0.0	2.0	1.0	0.0	0.0	0.0	3.0	0	1	low	0	0	
0.0	1.0	2.0	9.0	0.0	2.0	0.0	9.0	0	0	0	0	0	
0.0	0.0	1.0	5.0	1.0	0.0	0.0	NaN	0	1	low	0	0	
1.0	1.0	2.0	9.0	0.0	1.0	0.0	1.0	0	0	0	0	0	

In [180]:
```
df_9['Equity_level'].value_counts()
```

Out[180]:
```
0      26687
low       20
Name: Equity_level, dtype: int64
```

In [ ]:

Attempt C at Feature Eng.
Going to go back to basics and do by row and utilize dot apply

In [218]:
```
df_7.head()
```

Out[218]:

ırital_status	rent_or_own	employment_status	hhs_geo_region	census_msa	household_adults	household_children	employment_industry	h1n1_vaccine	seasonal_vaccine	contact_people	contact_behavior	contact_total	Concern
0.0	1.0	1.0	8.0	2.0	0.0	0.0	NaN	0	0	low	0	0	
0.0	0.0	2.0	1.0	0.0	0.0	0.0	3.0	0	1	low	0	0	
0.0	1.0	2.0	9.0	0.0	2.0	0.0	9.0	0	0	0	0	0	
0.0	0.0	1.0	5.0	1.0	0.0	0.0	NaN	0	1	low	0	0	
1.0	1.0	2.0	9.0	0.0	1.0	0.0	1.0	0	0	0	0	0	

In [ ]:
```
# Want to create a new variable in DF based off of values in 6 existing columns
# Goal: new column has values from 0 to 6
def function to do a row:
Create new column z
For each row,
if col x = 1, then add 1 to col z
if col y = 1, then add 1 to col z
if col a = 1, then add 1 to col z

pandas.apply (for blah in blah)
```

In [ ]:
```
# Attenpt three
conditions = [(df_8['census_msa'] == 1) & (df_8['health_insurance'] == 0) & (df_8['income_poverty'] == 0) & (df_8['employment_status'] == 0) & (df_8['rent_or_own'] ==
values = ['low']
df_9['Equity_level'] = np.select(conditions, values)
df_9.head()
```

In [ ]:

Try a revised, cut down DF based on Feature Importance from RandomForest model -B-prep, non-SMOTE
The top 19 features only. Starting with df=rawm

In [182]:
```
rawm.shape
```

Out[182]:
```
(26707, 72)
```

In [183]:
```
rawm.columns
```

Out[183]:
```
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',
       'health_insurance', 'opinion_h1n1_vacc_effective', 'opinion_h1n1_risk',
       'opinion_h1n1_sick_from_vacc', 'opinion_seas_vacc_effective',
       'opinion_seas_risk', 'opinion_seas_sick_from_vacc', 'age_group',
       'education', 'sex', 'marital_status', 'rent_or_own', 'household_adults',
       'household_children', 'race__Black', 'race__Hispanic',
       'race__Other or Multiple', 'race__White', 'INC__75K_to_Poverty',
       'INC__over_75K', 'INC__below_Poverty', 'EMP__Employed',
       'EMP__Not_Labor', 'EMP__Unemployed', 'GEO__atmpeygn', 'GEO__bhuqouqj',
       'GEO__dqpwyqqj', 'GEO__fpwskwrf', 'GEO__kbazzjca', 'GEO__lrircsnp',
       'GEO__lzgpxyit', 'GEO__mlyzmhmf', 'GEO__oxchjgsf', 'GEO__qufhixun',
       'MSA_Not_Principle_City', 'MSA_Principle_City', 'MSA_Not',
       'INDUST__arjwrbjb', 'INDUST__atmlpfrs', 'INDUST__cfqqtusy',
       'INDUST__dotnnunm', 'INDUST__fcxhlnwr', 'INDUST__haxffmxo',
       'INDUST__ldnlellj', 'INDUST__mcubkhph', 'INDUST__mfikgejo',
       'INDUST__msuufmds', 'INDUST__nduyfdeo', 'INDUST__phxvnwax',
       'INDUST__pxcmvdjn', 'INDUST__qnlwzans', 'INDUST__rucpziij',
       'INDUST__saaquncn', 'INDUST__vjjrobsf', 'INDUST__wlfvacwt',
       'INDUST__wxleyezf', 'INDUST__xicduogh', 'INDUST__xqicxuve'],
      dtype='object')
```

In [184]:
```
# Cut out the less important features
rawp = rawm.drop(columns=['sex', 'marital_status', 'rent_or_own', 'race__Black', 'race__Hispanic',
       'race__Other or Multiple', 'race__White', 'behavioral_antiviral_meds',
       'behavioral_avoidance', 'behavioral_face_mask', 'behavioral_wash_hands',
       'behavioral_large_gatherings', 'behavioral_outside_home',
       'behavioral_touch_face', 'chronic_med_condition', 'child_under_6_months', 'GEO__atmpeygn', 'GEO__bhuqouqj',
       'GEO__dqpwygqj', 'GEO__fpwskwrf', 'GEO__kbazzjca', 'GEO__lrircsnp',
       'GEO__lzgpxyit', 'GEO__mlyzmhmf', 'GEO__oxchjgsf', 'GEO__qufhixun',
       'MSA_Not_Principle_City', 'MSA_Principle_City', 'MSA_Not',
       'INDUST__arjwrbjb', 'INDUST__atmlpfrs', 'INDUST__cfqqtusy',
       'INDUST__dotnnunm', 'INDUST__fcxhlnwr', 'INDUST__haxffmxo',
       'INDUST__ldnlellj', 'INDUST__mcubkhph', 'INDUST__mfikgejo',
       'INDUST__msuufmds', 'INDUST__nduyfdeo', 'INDUST__phxvnwax',
       'INDUST__pxcmvdjn', 'INDUST__qnlwzans', 'INDUST__rucpziij',
       'INDUST__saaquncn', 'INDUST__vjjrobsf', 'INDUST__wlfvacwt',
       'INDUST__wxleyezf', 'INDUST__xicduogh', 'INDUST__xqicxuve', 'EMP__Employed',
       'EMP__Not_Labor', 'EMP__Unemployed'], axis=1)
rawp.head()
```

Out[184]:

	h1n1_concern	h1n1_knowledge	doctor_recc_h1n1	doctor_recc_seasonal	health_worker	health_insurance	opinion_h1n1_vacc_effective	opinion_h1n1_risk	opinion_h1n1_sick_from_vacc	opinion_seas_vacc_effective	opinion_se
0	-0.679477	-2.043782	-0.538812	-0.714456	-0.354921	0.429421	-0.844586	-1.048700	-0.264324	-1.869153	-1
1	1.519841	1.196063	-0.538812	-0.714456	-0.354921	0.429421	1.150052	1.296788	1.210360	-0.019914	-0
2	-0.679477	-0.423859	-0.538812	-0.714456	-0.354921	-1.074749	-0.844586	-1.048700	-1.001667	-0.019914	-1
3	-0.679477	-0.423859	-0.538812	1.477700	-0.354921	0.429421	-0.844586	0.514959	1.947702	0.904706	0
4	0.420182	-0.423859	-0.538812	-0.714456	-0.354921	0.429421	-0.844586	0.514959	-0.264324	-0.944533	-1

In [185]:
```
rawp.shape
```

Out[185]:
```
(26707, 19)
```

In [210]:
```
# save df as csv file for later use - modeling - imputed and scaled - and cut to 19
rawp.to_csv(r'df2_im_scal_19.csv')
```

In [186]:
```
# Need to split data into X and y dataframes - for h1n1
X = rawp
print(y1.shape)
print(X.shape)
```

```
(26707,)
(26707, 19)
```

In [187]:
```
# Create train and test sets.
X_train, X_test, y_train, y_test = train_test_split(X, y1, test_size=0.30, random_state=36)
print(X_train.shape)
print(X_test.shape)

print(y_train.shape)
print(y_test.shape)
```

```
(18694, 19)
(8013, 19)
(18694,)
(8013,)
```

In [188]:
```
def run_model (model):
    # Instantiate classification model
    cfmodel = model
    # Fit the classifier
    cfmodel.fit(X_train, y_train)

    # Predict on training and test sets
    training_preds = cfmodel.predict(X_train)
    test_preds = cfmodel.predict(X_test)

    # Get detailed results (Train and Test)
    print('-------------------------------------------------------------------------')
    print(f'MODEL: {model}')
    # Classification Report
    print('-------------------------------------------------------------------------')
    print('Classification Report - TRAIN')
    print('-------------------------------------------------------------------------')
    print(classification_report(y_train, training_preds))
    # Confusion Matrix
    print('-------------------------------------------------------------------------')
    print('Confusion Matrix - TRAIN')
    print('-------------------------------------------------------------------------')
    print(pd.crosstab(y_train, training_preds, rownames=['True'], colnames=['Predicted'], margins=True))
    print('\n-------------------------------------------------------------------------')
    # Classification Report
    print('-------------------------------------------------------------------------')
    print('Classification Report - TEST')
    print('-------------------------------------------------------------------------')
    print(classification_report(y_test, test_preds))
    # Confusion Matrix
    print('-------------------------------------------------------------------------')
    print('Confusion Matrix - TEST')
    print('-------------------------------------------------------------------------')
    print(pd.crosstab(y_test, test_preds, rownames=['True'], colnames=['Predicted'], margins=True))
    print('-------------------------------------------------------------------------')
```

In [189]:
```
# Assign the model... change this for each model to run.
model = DecisionTreeClassifier()
run_model(model)
```

```
--------------------------------------------------------------------------------
MODEL: DecisionTreeClassifier()
--------------------------------------------------------------------------------
Classification Report - TRAIN
--------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     14754
         1.0       1.00      0.99      0.99      3940

    accuracy                           1.00     18694
   macro avg       1.00      0.99      1.00     18694
weighted avg       1.00      1.00      1.00     18694

--------------------------------------------------------------------------------
Confusion Matrix - TRAIN
--------------------------------------------------------------------------------
Predicted    0.0   1.0    All
True
0.0        14752     2  14754
1.0           51  3889   3940
All        14803  3891  18694

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.85      0.84      0.84      6279
         1.0       0.44      0.45      0.44      1734

    accuracy                           0.76      8013
   macro avg       0.64      0.64      0.64      8013
weighted avg       0.76      0.76      0.76      8013

--------------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        5279  1000  6279
1.0         960   774  1734
All        6239  1774  8013
--------------------------------------------------------------------------------
```

In [190]:
```
# Assign the model... change this for each model to run.
model = LogisticRegression()
run_model(model)
```

```
--------------------------------------------------------------------------------
MODEL: LogisticRegression()
--------------------------------------------------------------------------------
Classification Report - TRAIN
--------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.86      0.95      0.90     14754
         1.0       0.68      0.42      0.52      3940

    accuracy                           0.84     18694
   macro avg       0.77      0.68      0.71     18694
weighted avg       0.82      0.84      0.82     18694

--------------------------------------------------------------------------------
Confusion Matrix - TRAIN
--------------------------------------------------------------------------------
Predicted    0.0   1.0    All
True
0.0        13970   784  14754
1.0         2299  1641   3940
All        16269  2425  18694

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.85      0.95      0.90      6279
         1.0       0.68      0.41      0.51      1734

    accuracy                           0.83      8013
   macro avg       0.77      0.68      0.71      8013
weighted avg       0.82      0.83      0.82      8013

--------------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        5950   329  6279
1.0        1019   715  1734
All        6969  1044  8013
--------------------------------------------------------------------------------
```

In [195]:
```
model = KNeighborsClassifier()
run_model(model)
```

```
--------------------------------------------------------------------------------
MODEL: KNeighborsClassifier()
--------------------------------------------------------------------------------
Classification Report - TRAIN
--------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.88      0.95      0.91     14754
         1.0       0.73      0.53      0.62      3940

    accuracy                           0.86     18694
   macro avg       0.81      0.74      0.77     18694
weighted avg       0.85      0.86      0.85     18694

--------------------------------------------------------------------------------
Confusion Matrix - TRAIN
--------------------------------------------------------------------------------
Predicted    0.0   1.0    All
True
0.0        13987   767  14754
1.0         1837  2103   3940
All        15824  2870  18694

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.86      0.92      0.89      6279
         1.0       0.60      0.44      0.50      1734

    accuracy                           0.81      8013
   macro avg       0.73      0.68      0.70      8013
weighted avg       0.80      0.81      0.80      8013

--------------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        5764   515  6279
1.0         975   759  1734
All        6739  1274  8013
--------------------------------------------------------------------------------
```

In [191]:
```
# Assign the model... change this for each model to run.
model = XGBClassifier()
run_model(model)
```

```
--------------------------------------------------------------------------------
MODEL: XGBClassifier()
--------------------------------------------------------------------------------
Classification Report - TRAIN
--------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.87      0.95      0.91     14754
         1.0       0.71      0.47      0.57      3940

    accuracy                           0.85     18694
   macro avg       0.79      0.71      0.74     18694
weighted avg       0.84      0.85      0.84     18694

--------------------------------------------------------------------------------
Confusion Matrix - TRAIN
--------------------------------------------------------------------------------
Predicted    0.0   1.0    All
True
0.0        13983   771  14754
1.0         2072  1868   3940
All        16055  2639  18694

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.86      0.95      0.90      6279
         1.0       0.70      0.45      0.55      1734

    accuracy                           0.84      8013
   macro avg       0.78      0.70      0.72      8013
weighted avg       0.83      0.84      0.83      8013

--------------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        5942   337  6279
1.0         953   781  1734
All        6895  1118  8013
--------------------------------------------------------------------------------
```

In [192]:
```
model = SVC()
run_model(model)
```

```
------------------------------------------------------------------------------
MODEL: SVC()
------------------------------------------------------------------------------
Classification Report - TRAIN
------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.87      0.96      0.91     14754
         1.0       0.75      0.46      0.57      3940

    accuracy                           0.85     18694
   macro avg       0.81      0.71      0.74     18694
weighted avg       0.84      0.85      0.84     18694

------------------------------------------------------------------------------
Confusion Matrix - TRAIN
------------------------------------------------------------------------------
Predicted    0.0   1.0    All
True
0.0        14142   612  14754
1.0         2146  1794   3940
All        16288  2406  18694

------------------------------------------------------------------------------
------------------------------------------------------------------------------
Classification Report - TEST
------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.86      0.95      0.90      6279
         1.0       0.71      0.42      0.53      1734

    accuracy                           0.84      8013
   macro avg       0.78      0.68      0.71      8013
weighted avg       0.82      0.84      0.82      8013

------------------------------------------------------------------------------
Confusion Matrix - TEST
------------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        5985   294  6279
1.0        1012   722  1734
All        6997  1016  8013
------------------------------------------------------------------------------
```

In [193]:
```
# Assign the model... change this for each model to run.
model = RandomForestClassifier()
run_model(model)
```

```
------------------------------------------------------------------------------
MODEL: RandomForestClassifier()
------------------------------------------------------------------------------
Classification Report - TRAIN
------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     14754
         1.0       1.00      0.99      0.99      3940

    accuracy                           1.00     18694
   macro avg       1.00      0.99      1.00     18694
weighted avg       1.00      1.00      1.00     18694

------------------------------------------------------------------------------
Confusion Matrix - TRAIN
------------------------------------------------------------------------------
Predicted    0.0   1.0    All
True
0.0        14738    16  14754
1.0           37  3903   3940
All        14775  3919  18694

------------------------------------------------------------------------------
------------------------------------------------------------------------------
Classification Report - TEST
------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.86      0.94      0.90      6279
         1.0       0.68      0.42      0.52      1734

    accuracy                           0.83      8013
   macro avg       0.77      0.68      0.71      8013
weighted avg       0.82      0.83      0.82      8013

------------------------------------------------------------------------------
Confusion Matrix - TEST
------------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        5926   353  6279
1.0         999   735  1734
All        6925  1088  8013
------------------------------------------------------------------------------
```

In [194]:
```
# Take a look at feature importances (RandomForest) - from data B model - cut-down-df

importance = pd.DataFrame(data={'features': X_train.columns, 'importance': model.feature_importances_})
importance = importance.sort_values('importance', ascending=False)
importance = importance.reset_index()
importance.drop('index', axis=1, inplace=True)
importance.head(16)
```

Out[194]:

	features	importance
0	doctor_recc_h1n1	0.119126
1	opinion_h1n1_risk	0.088653
2	opinion_h1n1_vacc_effective	0.077146
3	age_group	0.072514
4	opinion_seas_risk	0.064126
5	education	0.063053
6	opinion_h1n1_sick_from_vacc	0.059550
7	h1n1_concern	0.059083
8	opinion_seas_sick_from_vacc	0.057919
9	household_adults	0.053944
10	opinion_seas_vacc_effective	0.048267
11	household_children	0.042494
12	h1n1_knowledge	0.040488
13	doctor_recc_seasonal	0.037387
14	health_insurance	0.030810
15	health_worker	0.027041

Observations on the models run on the data set of 19 features

This run seemed to perform at about the same level as the previous run of models with the full set of 72 variables. Here XGBoost and SV performed best at accuracy of precission (class 1) of 0.71.

Trying one more round of DF simplification and run a few models

Will simplify the data from 19 columns, down to 10 to see if performance still holds.

In [196]:
```
print(rawp.shape)
rawp.head()
```

```
(26707, 19)
```

Out[196]:

	h1n1_concern	h1n1_knowledge	doctor_recc_h1n1	doctor_recc_seasonal	health_worker	health_insurance	opinion_h1n1_vacc_effective	opinion_h1n1_risk	opinion_h1n1_sick_from_vacc	opinion_seas_vacc_effective	opinion_se
0	-0.679477	-2.043782	-0.538812	-0.714456	-0.354921	0.429421	-0.844586	-1.048700	-0.264324	-1.869153	-1
1	1.519841	1.196063	-0.538812	-0.714456	-0.354921	0.429421	1.150052	1.296788	1.210360	-0.019914	-0
2	-0.679477	-0.423859	-0.538812	-0.714456	-0.354921	-1.074749	-0.844586	-1.048700	-1.001667	-0.019914	-1
3	-0.679477	-0.423859	-0.538812	1.477700	-0.354921	0.429421	-0.844586	0.514959	1.947702	0.904706	0
4	0.420182	-0.423859	-0.538812	-0.714456	-0.354921	0.429421	-0.844586	0.514959	-0.264324	-0.944533	-1

In [197]:
```
# Cut out the less important features
rawq = rawp.drop(columns=['h1n1_knowledge', 'doctor_recc_seasonal', 'health_worker', 'health_insurance', 'household_adults', 'household_children', 'INC__75K_to_Povert
print(rawq.shape)
rawq.head()
```

```
(26707, 10)
```

Out[197]:

	h1n1_concern	doctor_recc_h1n1	opinion_h1n1_vacc_effective	opinion_h1n1_risk	opinion_h1n1_sick_from_vacc	opinion_seas_vacc_effective	opinion_seas_risk	opinion_seas_sick_from_vacc	age_group	education
0	-0.679477	-0.538812	-0.844586	-1.048700	-0.264324	-1.869153	-1.245424	-0.090271	0.558480	-2.006398
1	1.519841	-0.538812	1.150052	1.296788	1.210360	-0.019914	-0.519056	1.420485	-0.813928	-0.990286
2	-0.679477	-0.538812	-0.844586	-1.048700	-1.001667	-0.019914	-1.245424	-0.090271	-1.500131	1.041938
3	-0.679477	-0.538812	-0.844586	0.514959	1.947702	0.904706	0.933679	-0.845649	1.244684	-0.990286
4	0.420182	-0.538812	-0.844586	0.514959	-0.264324	-0.944533	-1.245424	1.420485	-0.127724	0.025826

In [198]:
```
# Need to split data into X and y dataframes - for h1n1
X = rawq
print(y1.shape)
print(X.shape)
```

```
(26707,)
(26707, 10)
```

In [199]:
```
# Create train and test sets.
X_train, X_test, y_train, y_test = train_test_split(X, y1, test_size=0.30, random_state=36)
print(X_train.shape)
print(X_test.shape)

print(y_train.shape)
print(y_test.shape)
```

```
(18694, 10)
(8013, 10)
(18694,)
(8013,)
```

In [200]:
```
def run_model (model):
    # Instantiate classification model
    cfmodel = model
    # Fit the classifier
    cfmodel.fit(X_train, y_train)

    # Predict on training and test sets
    training_preds = cfmodel.predict(X_train)
    test_preds = cfmodel.predict(X_test)

    # Get detailed results (Train and Test)
    print('------------------------------------------------------------------------')
    print(f'MODEL: {model}')
    # Classification Report
    print('------------------------------------------------------------------------')
    print('Classification Report - TRAIN')
    print('------------------------------------------------------------------------')
    print(classification_report(y_train, training_preds))
    # Confusion Matrix
    print('------------------------------------------------------------------------')
    print('Confusion Matrix - TRAIN')
    print('------------------------------------------------------------------------')
    print(pd.crosstab(y_train, training_preds, rownames=['True'], colnames=['Predicted'], margins=True))
    print('\n------------------------------------------------------------------------')
    # Classification Report
    print('------------------------------------------------------------------------')
    print('Classification Report - TEST')
    print('------------------------------------------------------------------------')
    print(classification_report(y_test, test_preds))
    # Confusion Matrix
    print('------------------------------------------------------------------------')
    print('Confusion Matrix - TEST')
    print('------------------------------------------------------------------------')
    print(pd.crosstab(y_test, test_preds, rownames=['True'], colnames=['Predicted'], margins=True))
    print('------------------------------------------------------------------------')
```

In [201]:
```
# Assign the model... change this for each model to run.
model = XGBClassifier()
run_model(model)
```

```
------------------------------------------------------------------------
MODEL: XGBClassifier()
------------------------------------------------------------------------
Classification Report - TRAIN
------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.86      0.95      0.90     14754
         1.0       0.69      0.44      0.54      3940

    accuracy                           0.84     18694
   macro avg       0.78      0.69      0.72     18694
weighted avg       0.83      0.84      0.83     18694

------------------------------------------------------------------------
Confusion Matrix - TRAIN
------------------------------------------------------------------------
Predicted    0.0   1.0    All
True
0.0        13965   789  14754
1.0         2198  1742   3940
All        16163  2531  18694

------------------------------------------------------------------------
------------------------------------------------------------------------
Classification Report - TEST
------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.86      0.95      0.90      6279
         1.0       0.68      0.42      0.52      1734

    accuracy                           0.83      8013
   macro avg       0.77      0.69      0.71      8013
weighted avg       0.82      0.83      0.82      8013

------------------------------------------------------------------------
Confusion Matrix - TEST
------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        5940   339  6279
1.0         998   736  1734
All        6938  1075  8013
------------------------------------------------------------------------
```

In [202]:
```
model = SVC()
run_model(model)
```

```
------------------------------------------------------------------------------
MODEL: SVC()
------------------------------------------------------------------------------
Classification Report - TRAIN
------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.86      0.95      0.90     14754
         1.0       0.70      0.42      0.52      3940

    accuracy                           0.84     18694
   macro avg       0.78      0.69      0.71     18694
weighted avg       0.83      0.84      0.82     18694

------------------------------------------------------------------------------
Confusion Matrix - TRAIN
------------------------------------------------------------------------------
Predicted    0.0   1.0    All
True
0.0        14037   717  14754
1.0         2284  1656   3940
All        16321  2373  18694

------------------------------------------------------------------------------
------------------------------------------------------------------------------
Classification Report - TEST
------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.85      0.95      0.90      6279
         1.0       0.68      0.39      0.50      1734

    accuracy                           0.83      8013
   macro avg       0.76      0.67      0.70      8013
weighted avg       0.81      0.83      0.81      8013

------------------------------------------------------------------------------
Confusion Matrix - TEST
------------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        5956   323  6279
1.0        1051   683  1734
All        7007  1006  8013
------------------------------------------------------------------------------
```

In [203]:
```
model = RandomForestClassifier()
run_model(model)
```

```
------------------------------------------------------------------------------
MODEL: RandomForestClassifier()
------------------------------------------------------------------------------
Classification Report - TRAIN
------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.97      0.98      0.97     14754
         1.0       0.93      0.87      0.90      3940

    accuracy                           0.96     18694
   macro avg       0.95      0.93      0.94     18694
weighted avg       0.96      0.96      0.96     18694

------------------------------------------------------------------------------
Confusion Matrix - TRAIN
------------------------------------------------------------------------------
Predicted    0.0   1.0    All
True
0.0        14513   241  14754
1.0          506  3434   3940
All        15019  3675  18694

------------------------------------------------------------------------------
------------------------------------------------------------------------------
Classification Report - TEST
------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.85      0.92      0.88      6279
         1.0       0.59      0.43      0.50      1734

    accuracy                           0.81      8013
   macro avg       0.72      0.67      0.69      8013
weighted avg       0.80      0.81      0.80      8013

------------------------------------------------------------------------------
Confusion Matrix - TEST
------------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        5758   521  6279
1.0         987   747  1734
All        6745  1268  8013
------------------------------------------------------------------------------
```

In [204]:
```
# Take a look at feature importances (RandomForest) - from data B model - cut-down-df

importance = pd.DataFrame(data={'features': X_train.columns, 'importance': model.feature_importances_})
importance = importance.sort_values('importance', ascending=False)
importance = importance.reset_index()
importance.drop('index', axis=1, inplace=True)
importance.head(16)
```

Out[204]:

	features	importance
0	doctor_recc_h1n1	0.156979
1	age_group	0.123590
2	opinion_h1n1_risk	0.110586
3	education	0.105288
4	opinion_h1n1_vacc_effective	0.093408
5	h1n1_concern	0.088006
6	opinion_seas_risk	0.086017
7	opinion_seas_sick_from_vacc	0.084452
8	opinion_h1n1_sick_from_vacc	0.084041
9	opinion_seas_vacc_effective	0.067634

Try a run of GridSearch CV - on the XGBoost model with 10 feature DF

In [205]:
```
model = XGBClassifier()
run_model(model)
```

```
--------------------------------------------------------------------------------
MODEL: XGBClassifier()
--------------------------------------------------------------------------------
Classification Report - TRAIN
--------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.86      0.95      0.90     14754
         1.0       0.69      0.44      0.54      3940

    accuracy                           0.84     18694
   macro avg       0.78      0.69      0.72     18694
weighted avg       0.83      0.84      0.83     18694

--------------------------------------------------------------------------------
Confusion Matrix - TRAIN
--------------------------------------------------------------------------------
Predicted    0.0   1.0    All
True
0.0        13965   789  14754
1.0         2198  1742   3940
All        16163  2531  18694

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------------
              precision    recall  f1-score   support

         0.0       0.86      0.95      0.90      6279
         1.0       0.68      0.42      0.52      1734

    accuracy                           0.83      8013
   macro avg       0.77      0.69      0.71      8013
weighted avg       0.82      0.83      0.82      8013

--------------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------------
Predicted   0.0   1.0   All
True
0.0        5940   339  6279
1.0         998   736  1734
All        6938  1075  8013
--------------------------------------------------------------------------------
```

In [206]:
```
# Set-up the parameter grid
param_grid = {
    'learning_rate': [0.1, 0.2],
    'max_depth': [4, 5, 6],
    'min_child_weight': [3, 4],
    'subsample': [0.5, 0.7],
    'n_estimators': [100],
}
```

In [207]:
```
# Code to run it

grid_clf = GridSearchCV(model, param_grid, scoring='accuracy', cv=5, n_jobs=1)
grid_clf.fit(X_train, y_train)

best_parameters = grid_clf.best_params_

print('Grid Search found the following optimal parameters: ')
for param_name in sorted(best_parameters.keys()):
    print('%s: %r' % (param_name, best_parameters[param_name]))

training_preds = grid_clf.predict(X_train)
test_preds = grid_clf.predict(X_test)
training_accuracy = accuracy_score(y_train, training_preds)
test_accuracy = accuracy_score(y_test, test_preds)

print('')
print('Training Accuracy: {:.4}%'.format(training_accuracy * 100))
print('Validation accuracy: {:.4}%'.format(test_accuracy * 100))
```

```
Grid Search found the following optimal parameters:
learning_rate: 0.1
max_depth: 4
min_child_weight: 3
n_estimators: 100
subsample: 0.5

Training Accuracy: 84.43%
Validation accuracy: 83.28%
```

In [208]:
```
# Set-up the parameter grid
param_grid = {
    'learning_rate': [0.05, 0.1],
    'max_depth': [2, 3, 4],
    'min_child_weight': [2, 3],
    'subsample': [0.4, 0.5],
    'n_estimators': [100],
}
```

In [209]:
```
# Code to run it

grid_clf = GridSearchCV(model, param_grid, scoring='accuracy', cv=5, n_jobs=1)
grid_clf.fit(X_train, y_train)

best_parameters = grid_clf.best_params_

print('Grid Search found the following optimal parameters: ')
for param_name in sorted(best_parameters.keys()):
    print('%s: %r' % (param_name, best_parameters[param_name]))

training_preds = grid_clf.predict(X_train)
test_preds = grid_clf.predict(X_test)
training_accuracy = accuracy_score(y_train, training_preds)
test_accuracy = accuracy_score(y_test, test_preds)

print('')
print('Training Accuracy: {:.4}%'.format(training_accuracy * 100))
print('Validation accuracy: {:.4}%'.format(test_accuracy * 100))
```

```
Grid Search found the following optimal parameters:
learning_rate: 0.1
max_depth: 4
min_child_weight: 2
n_estimators: 100
subsample: 0.5

Training Accuracy: 84.41%
Validation accuracy: 83.26%
```

Observations on the models run on the data set of 10 features

Intersting, models still performing at about the same level as with 19 features - at least accuracy wise (0.83). The precssion for class 1 is dropping a bit - down fr
SVC down to 0.68. Two rounds of GridSearch CV were also tried on this set of data, but very modest improvement was obtained - accuracy only improved to 0.8326.

In [ ]:

Compiling the Results of Classifiacation Modeling

Putting a summary of the modeling results into dataframes. These summary tables include all modeling runs - with both data preperations - approach A and approach B.

Approach A. Keeping all features as ordinal (39 features)

This approach used KNN Imputing to address missing values, and StandardScaler.

In [211]:
```
# Base modeling
data1 = [['XGBoost', 0.85, 0.68], ['Random Forest', 0.84, 0.68], ['SVC', 0.84, 0.67], ['Logistic Regression', 0.84, 0.66], ['KNN', 0.81, 0.54], ['Decision Trees', 0.7
dfA_results = pd.DataFrame(data1, columns = ['Classification model', 'Accuracy', 'Precision-1'])
dfA_results
```

Out[211]:

	Classification model	Accuracy	Precision-1
0	XGBoost	0.85	0.68
1	Random Forest	0.84	0.68
2	SVC	0.84	0.67
3	Logistic Regression	0.84	0.66
4	KNN	0.81	0.54
5	Decision Trees	0.75	0.40

In [212]:
```
# Base modeling with SMOTE - same as above but with SMOTE added.
data2 = [['XGBoost', 0.83, 0.61], ['Random Forest', 0.83, 0.64], ['SVC', 0.80, 0.52], ['Logistic Regression', 0.77, 0.48], ['KNN', 0.67, 0.35], ['Decision Trees', 0.7
dfB_results = pd.DataFrame(data2, columns = ['Classification model', 'Accuracy', 'Precision-1'])
dfB_results
```

Out[212]:

	Classification model	Accuracy	Precision-1
0	XGBoost	0.83	0.61
1	Random Forest	0.83	0.64
2	SVC	0.80	0.52
3	Logistic Regression	0.77	0.48
4	KNN	0.67	0.35
5	Decision Trees	0.74	0.41

Approach B. Onehot encoding some variables and leave some as ordinal (72 features)

This approach used KNN Imputing to address missing values, and StandardScaler.

In [213]:
```
# Approach B - Base modeling
data3 = [['XGBoost', 0.84, 0.71], ['Random Forest', 0.84, 0.72], ['SVC', 0.83, 0.70], ['Logistic Regression', 0.83, 0.69], ['KNN', 0.80, 0.59], ['Decision Trees', 0.7
dfC_results = pd.DataFrame(data3, columns = ['Classification model', 'Accuracy', 'Precision-1'])
dfC_results
```

Out[213]:

	Classification model	Accuracy	Precision-1
0	XGBoost	0.84	0.71
1	Random Forest	0.84	0.72
2	SVC	0.83	0.70
3	Logistic Regression	0.83	0.69
4	KNN	0.80	0.59
5	Decision Trees	0.76	0.45

In [214]:
```
# Approach B - Base modeling with SMOTE
data4 = [['XGBoost', 0.83, 0.65], ['Random Forest', 0.83, 0.67], ['SVC', 0.81, 0.57]]
dfD_results = pd.DataFrame(data4, columns = ['Classification model', 'Accuracy', 'Precision-1'])
dfD_results
```

Out[214]:

	Classification model	Accuracy	Precision-1
0	XGBoost	0.83	0.65
1	Random Forest	0.83	0.67
2	SVC	0.81	0.57

Approach C. Same as B, but cut down to most impt features (19 features; 10 features)

This approach used KNN Imputing to address missing values, and StandardScaler.

In [215]:
```
# Approach C - with 19 features (based on feature importances from previous models) and also cut to 10.
data5 = [['XGBoost w/ 19', 0.84, 0.70], ['SVC w/ 19', 0.84, 0.71], ['XGBoost w/ 10', 0.83, 0.68], ['SVC w/ 10', 0.83, 0.68]]
dfE_results = pd.DataFrame(data5, columns = ['Classification model', 'Accuracy', 'Precision-1'])
dfE_results
```

Out[215]:

	Classification model	Accuracy	Precision-1
0	XGBoost w/ 19	0.84	0.70
1	SVC w/ 19	0.84	0.71
2	XGBoost w/ 10	0.83	0.68
3	SVC w/ 10	0.83	0.68

Approach D. SEASONAL Vaccine - Same as approach A... Keeping features all as ordinal (39 features)

This approach used KNN Imputing to address missing values, and StandardScaler.
Note that this dataset has balanced target classes. No SMOTING was needed.

In [216]:
```
# Base modeling
data6 = [['XGBoost', 0.78, 0.77], ['Random Forest', 0.77, 0.76], ['SVC', 0.77, 0.76], ['Logistic Regression', 0.77, 0.76], ['KNN', 0.71, 0.69], ['Decision Trees', 0.6
dfF_seas_results = pd.DataFrame(data6, columns = ['Classification model', 'Accuracy', 'Precision-1'])
dfF_seas_results
```

Out[216]:

	Classification model	Accuracy	Precision-1
0	XGBoost	0.78	0.77
1	Random Forest	0.77	0.76
2	SVC	0.77	0.76
3	Logistic Regression	0.77	0.76
4	KNN	0.71	0.69
5	Decision Trees	0.67	~

In [217]:
```
# Using Data Prep Approach B (72 features)
data7 = [['XGBoost', 0.79, 0.78], ['Random Forest', 0.78, 0.78], ['SVC', 0.78, 0.77]]
dfG_seas_results = pd.DataFrame(data7, columns = ['Classification model', 'Accuracy', 'Precision-1'])
dfG_seas_results
```

Out[217]:

	Classification model	Accuracy	Precision-1
0	XGBoost	0.79	0.78
1	Random Forest	0.78	0.78
2	SVC	0.78	0.77

In [ ]: