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

Mark Patterson, March 2021

Introduction to 2a
Following EDA, I imported the data into this notebook for additional data preparation, model preprocessing, and initial runs of classification modeling. Eventually, I
change some of the data preperation and that, along with further runs of the models are in notebook 2b. That approach, I have named Approach B.

This notebook contains runs of Approach A classification modeling - utiliing a pipeline for pre-processign and modeling.

A Note on model evaluation criteria.
To evaluate the performacne of each model I will use accuracy, to gauge the overall predictive power of the model. Another important criteria is precision of class 1.
our target variable are the respondents that DID get the H1N1 vaccination. Precision of class 1 tells us percentage of people that the model predicted to have goten t
vaccination, but in reality had not. This is important as it could mean people that need additional vaccination informaiton may get overlooked.

Import Libraries and Load Data

In [1]:
```python
# 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.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 [3]:
```python
# 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[3]:

	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 [43]:
```python
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[43]:
```
0    78.754634
1    21.245366
Name: h1n1_vaccine, dtype: float64
```

Classification Modelling
In early explorations (see other Notebook) a "dummy classifier" model was run, and with the "stratified" version, accuracy was 0.67. The plan for classification model
test 6 different models:
a) Random Forest (to btain feature importances)
b) XGBoost (for its power)
c) KNN
d) Decision Trees
e) Logistic Regression
f) SVC

Grid Search will be used to optimize the parameters of some models.
First, all models will be run with h1n1 vaccination as the target variable (Y); and then a second series of models will be run with seasonal vaccination as the target
(Y2).

A pipeline will be constructed to address multiple steps of modeling including:
a) Imputing - KNNImputer
b) Scaling - StandardScaler
c) SMOTE - for h1n1 vaccination set - as there is sizeable class imbalance
d) Model - fit and predict
e) Results - summary reports and confusion matrix

Creating Train and Test Set

In [4]:
```python
df_8 = df_7.drop(columns=['Unnamed: 0'], axis=1)
```

In [5]:
```
# Need to split data into X and y dataframes.
y = df_8['h1n1_vaccine']
X = df_8.drop(columns=['h1n1_vaccine', 'seasonal_vaccine'], axis=1)
print(y.shape)
print(X.shape)
```

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

In [6]:
```
# Create train and test sets.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=34)
print(X_train.shape)
print(X_test.shape)

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

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

Run a Dummy Classifier

In [7]:
```
from sklearn.dummy import DummyClassifier
```

In [8]:
```
dclf = DummyClassifier(strategy='stratified', random_state=15)
dclf.fit(X_train, y_train)
dclf.score(X_test, y_test)
```

Out[8]:
```
0.6680394359166355
```

Run initial classification models (defaults) without SMOTE

Initial attempts at running a pipeline that included SMOTE failed. So will first run 6 models in plain default versions using the pipeline. NOTE that this is data pre
(with 34 variables).

In [9]:
```
# Load a few more libraries
from sklearn.impute import KNNImputer
from imblearn.over_sampling import SMOTE
from sklearn.pipeline import Pipeline
from sklearn.pipeline import make_pipeline
```

In [31]:
```
# Trying a different type of pipeline (B) and see if SMOTE works. Nope... same error message as before.
# But it does work without the SMOTE.
pipe = make_pipeline(KNNImputer(), StandardScaler(), RandomForestClassifier(random_state=44))
pipe = pipe.fit(X_train, y_train)
y_pred = pipe.predict(X_test)
accuracy_1 = accuracy_score(y_test, y_pred)
```

In [32]:
```
print(accuracy_1)
```

```
0.842755522276301
```

In [24]:
```
from imblearn.pipeline import Pipeline
```

In [29]:
```
# Yet, another version... (C). It runs without an error, BUT it does not appear to actually do the SMOTE (unequal classes still)
random_state = 38
model3 = Pipeline([
        ('imp', KNNImputer()),
        ('scal', StandardScaler()),
        ('smote', SMOTE()),
        ('classification', RandomForestClassifier())
    ])


model3.fit(X_train, y_train)
training_preds = model3.predict(X_train)
test_preds = model3.predict(X_test)
accuracy_3 = accuracy_score(y_test, y_pred)
print(accuracy_3)
```

```
0.8361412704355422
```

In [30]:

```python
# Get results - one off for the imblearn pipeline.
print('--------------------------------------------------------------------------')
print('MODEL: Random Forest with SMOTE')
print('\nClassification Report - TRAIN')
print('--------------------------------------------------------------------------')
print(classification_report(y_train, training_preds))
print('--------------------------------------------------------------------------')
# Confusion Matrix
print('--------------------------------------------------------------------------')
print('Confusion Matrix - TRAIN')
print('--------------------------------------------------------------------------')
print(pd.crosstab(y_train, training_preds, rownames=['True'], colnames=['Predicted'], margins=True))
print('--------------------------------------------------------------------------')
print('--------------------------------------------------------------------------')
# Classification Report
print('--------------------------------------------------------------------------')
print('Classification Report - TEST')
print('--------------------------------------------------------------------------')
print(classification_report(y_test, test_preds))
print('--------------------------------------------------------------------------')
# Confusion Matrix
print('--------------------------------------------------------------------------')
print('Confusion Matrix - TEST')
print('--------------------------------------------------------------------------')
print(pd.crosstab(y_test, test_preds, rownames=['True'], colnames=['Predicted'], margins=True))
print('--------------------------------------------------------------------------')
```

```
--------------------------------------------------------------------------
MODEL: Random Forest with SMOTE

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

           0       1.00      1.00      1.00     14645
           1       1.00      1.00      1.00      4049

    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     1    All
True
0          14645     0  14645
1              2  4047   4049
All        14647  4047  18694
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.88      0.93      0.90      6388
           1       0.62      0.48      0.54      1625

    accuracy                           0.84      8013
   macro avg       0.75      0.70      0.72      8013
weighted avg       0.82      0.84      0.83      8013

--------------------------------------------------------------------------
--------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------
Predicted     0     1   All
True
0          5911   477  6388
1           841   784  1625
All        6752  1261  8013
--------------------------------------------------------------------------
```

In [80]:

```python
# MODEL RUN 1.
# Original Pipeline.. SMOTE step throws an error. So commented it out. Will fallback to running SMOTE as a sept. step.

def classif_report (model):
    imputer = KNNImputer()
    scaler = StandardScaler()
#     xmoter = SMOTE()
    pipeline = Pipeline(steps=[('i', imputer), ('s', scaler), ('m', model)])
    pipeline.fit(X_train, y_train)

    training_preds = pipeline.predict(X_train)
    test_preds = pipeline.predict(X_test)

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

In [81]:

```python
# Assign the model... change this for each model to run.
model = RandomForestClassifier()
```

In [82]:
```
# Run the function with the assigned model above.
classif_report (model)
```

```
--------------------------------------------------------------------------
MODEL: RandomForestClassifier()

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

           0       1.00      1.00      1.00     14645
           1       1.00      1.00      1.00      4049

    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     1    All
True
0          14645     0  14645
1              0  4049   4049
All        14645  4049  18694
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.86      0.95      0.91      6388
           1       0.68      0.41      0.51      1625

    accuracy                           0.84      8013
   macro avg       0.77      0.68      0.71      8013
weighted avg       0.83      0.84      0.82      8013

--------------------------------------------------------------------------
--------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------
Predicted     0    1   All
True
0          6079  309  6388
1           964  661  1625
All        7043  970  8013
--------------------------------------------------------------------------
```

In [83]:
```
# Take a look at feature importances (RandomForest) - from default / vanilla model (no SMOTE)

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(20)
```

Out[83]:

	features	importance
0	doctor_recc_h1n1	0.111224
1	employment_industry	0.072995
2	opinion_h1n1_risk	0.068024
3	opinion_h1n1_vacc_effective	0.062342
4	hhs_geo_region	0.054741
5	opinion_seas_risk	0.045590
6	age_group	0.037457
7	opinion_h1n1_sick_from_vacc	0.033034
8	education	0.032316
9	opinion_seas_vacc_effective	0.031408
10	h1n1_concern	0.030971
11	opinion_seas_sick_from_vacc	0.030466
12	income_poverty	0.029081
13	doctor_recc_seasonal	0.028419
14	census_msa	0.027187
15	household_adults	0.025311
16	h1n1_knowledge	0.022620
17	household_children	0.022389
18	employment_status	0.020372
19	health_insurance	0.019283

In [85]:
```
# Assign the model... change this for each model to run.
model = KNeighborsClassifier()
# Run the function with the assigned model above.
classif_report (model)
```

```
--------------------------------------------------------------------------
MODEL: KNeighborsClassifier()

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

           0       0.87      0.95      0.91     14645
           1       0.75      0.50      0.60      4049

    accuracy                           0.86     18694
   macro avg       0.81      0.73      0.76     18694
weighted avg       0.85      0.86      0.84     18694

--------------------------------------------------------------------------
--------------------------------------------------------------------------
Confusion Matrix - TRAIN
--------------------------------------------------------------------------
Predicted      0     1    All
True
0          13949   696  14645
1           2005  2044   4049
All        15954  2740  18694
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.85      0.92      0.88      6388
           1       0.54      0.38      0.44      1625

    accuracy                           0.81      8013
   macro avg       0.70      0.65      0.66      8013
weighted avg       0.79      0.81      0.80      8013

--------------------------------------------------------------------------
--------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------
Predicted     0     1   All
True
0          5874   514  6388
1          1013   612  1625
All        6887  1126  8013
--------------------------------------------------------------------------
```

In [86]:
```
# Assign the model... change this for each model to run.
model = XGBClassifier()
# Run the function with the assigned model above.
classif_report (model)
```

```
--------------------------------------------------------------------------
MODEL: XGBClassifier()

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

           0       0.87      0.95      0.91     14645
           1       0.72      0.49      0.58      4049

    accuracy                           0.85     18694
   macro avg       0.79      0.72      0.74     18694
weighted avg       0.84      0.85      0.84     18694

--------------------------------------------------------------------------
--------------------------------------------------------------------------
Confusion Matrix - TRAIN
--------------------------------------------------------------------------
Predicted      0     1    All
True
0          13865   780  14645
1           2070  1979   4049
All        15935  2759  18694
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.87      0.94      0.91      6388
           1       0.68      0.47      0.55      1625

    accuracy                           0.85      8013
   macro avg       0.78      0.71      0.73      8013
weighted avg       0.83      0.85      0.84      8013

--------------------------------------------------------------------------
--------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------
Predicted     0     1   All
True
0          6027   361  6388
1           866   759  1625
All        6893  1120  8013
--------------------------------------------------------------------------
```

In [87]:
```python
# Assign the model... change this for each model to run.
model = DecisionTreeClassifier()
# Run the function with the assigned model above.
classif_report (model)
```

```
--------------------------------------------------------------------------
MODEL: DecisionTreeClassifier()

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

           0       1.00      1.00      1.00     14645
           1       1.00      1.00      1.00      4049

    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     1    All
True
0          14645     0  14645
1              0  4049   4049
All        14645  4049  18694
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.86      0.83      0.84      6388
           1       0.40      0.46      0.43      1625

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

--------------------------------------------------------------------------
--------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------
Predicted     0     1   All
True
0          5281  1107  6388
1           885   740  1625
All        6166  1847  8013
--------------------------------------------------------------------------
```

In [88]:
```python
# Assign the model... change this for each model to run.
model = LogisticRegression()
# Run the function with the assigned model above.
classif_report (model)
```

```
--------------------------------------------------------------------------
MODEL: LogisticRegression()

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

           0       0.86      0.94      0.90     14645
           1       0.67      0.43      0.52      4049

    accuracy                           0.83     18694
   macro avg       0.77      0.68      0.71     18694
weighted avg       0.82      0.83      0.82     18694

--------------------------------------------------------------------------
--------------------------------------------------------------------------
Confusion Matrix - TRAIN
--------------------------------------------------------------------------
Predicted      0     1    All
True
0          13812   833  14645
1           2324  1725   4049
All        16136  2558  18694
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.86      0.95      0.90      6388
           1       0.66      0.42      0.51      1625

    accuracy                           0.84      8013
   macro avg       0.76      0.68      0.71      8013
weighted avg       0.82      0.84      0.82      8013

--------------------------------------------------------------------------
--------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------
Predicted     0     1   All
True
0          6037   351  6388
1           943   682  1625
All        6980  1033  8013
--------------------------------------------------------------------------
```

In [89]:
```
# Assign the model... change this for each model to run.
model = SVC()
# Run the function with the assigned model above.
classif_report (model)
```

```
--------------------------------------------------------------------------
MODEL: SVC()

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

           0       0.88      0.97      0.92     14645
           1       0.80      0.51      0.62      4049

    accuracy                           0.87     18694
   macro avg       0.84      0.74      0.77     18694
weighted avg       0.86      0.87      0.85     18694

--------------------------------------------------------------------------
--------------------------------------------------------------------------
Confusion Matrix - TRAIN
--------------------------------------------------------------------------
Predicted      0     1    All
True
0          14136   509  14645
1           1986  2063   4049
All        16122  2572  18694
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.87      0.95      0.91      6388
           1       0.67      0.42      0.52      1625

    accuracy                           0.84      8013
   macro avg       0.77      0.69      0.71      8013
weighted avg       0.83      0.84      0.83      8013

--------------------------------------------------------------------------
--------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------
Predicted     0     1   All
True
0          6056   332  6388
1           936   689  1625
All        6992  1021  8013
--------------------------------------------------------------------------
```

Observations on the first 6 models - dat prep A with no SMOTE (34 variables)

Accuracy was about equal across most models at 0.84. XGBoost had the best precision (class 1) score at 0.68. Decision Trees performed worst at accuracy of 0.0.75 and
0.40.

In [ ]:

Reconfigure things for SMOTE

Tried using SMOTE as part of the pipeline. Had some issues getting it to run. Tried doing a seperate preprocessing pipeline, but then had issues figuring out how to f
a seperate SMOTE step.

In [97]:
```
# Preprocessing pipeline
imputer = KNNImputer()
scaler = StandardScaler()
pipe_pre = make_pipeline(imputer, scaler)
# x_train = pipe_pre.fit_transform(X_train, y_train)
```

Out[97]:
```
array([[ 0.42475416,  1.19594035, -0.22565807, ..., -1.18156013,
         0.49172474, -1.55273258],
       [ 1.52458714, -0.42438383, -0.22565807, ..., -1.18156013,
        -0.58271974,  0.56911297],
       [-0.67507883, -0.42438383, -0.22565807, ...,  2.80800331,
         0.49172474, -1.23124083],
       ...,
       [ 1.52458714, -0.42438383, -0.22565807, ...,  0.14829435,
        -0.58271974, -1.23124083],
       [ 0.42475416,  1.19594035, -0.22565807, ...,  1.47814883,
        -0.58271974,  0.44051627],
       [ 0.42475416, -0.42438383, -0.22565807, ...,  0.14829435,
         0.49172474, -1.55273258]])
```

In [98]:
```python
# 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 = 3)
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': 14645
Before OverSampling, counts of label '1': 4049

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-98-3af90d0179de> in <module>()
      5 
      6 sm = SMOTE(random_state = 3)
----> 7 X_train_s, y_train_s = sm.fit_sample(X_train, y_train)
      8 
      9 print('After OverSampling, the shape of train_X: {}'.format(X_train_s.shape))

/Users/markp/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/imblearn/base.py in fit_resample(self, X, y)
     75         check_classification_targets(y)
     76         arrays_transformer = ArraysTransformer(X, y)
---> 77         X, y, binarize_y = self._check_X_y(X, y)
     78 
     79         self.sampling_strategy_ = check_sampling_strategy(

/Users/markp/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/imblearn/base.py in _check_X_y(self, X, y, accept_sparse)
    133         y, binarize_y = check_target_type(y, indicate_one_vs_all=True)
    134         X, y = self._validate_data(
--> 135             X, y, reset=True, accept_sparse=accept_sparse
    136         )
    137         return X, y, binarize_y

/Users/markp/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/sklearn/base.py in _validate_data(self, X, y, reset, validate_separately, **check_params)
    430                 y = check_array(y, **check_y_params)
    431             else:
--> 432                 X, y = check_X_y(X, y, **check_params)
    433             out = X, y
    434 

/Users/markp/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/sklearn/utils/validation.py in inner_f(*args, **kwargs)
     71                           FutureWarning)
     72         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 73         return f(**kwargs)
     74     return inner_f
     75 

/Users/markp/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/sklearn/utils/validation.py in check_X_y(X, y, accept_sparse, accept_large_sparse, dtype, order,
all_finite, ensure_2d, allow_nd, multi_output, ensure_min_samples, ensure_min_features, y_numeric, estimator)
    801                     ensure_min_samples=ensure_min_samples,
    802                     ensure_min_features=ensure_min_features,
--> 803                     estimator=estimator)
    804     if multi_output:
    805         y = check_array(y, accept_sparse='csr', force_all_finite=True,

/Users/markp/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/sklearn/utils/validation.py in inner_f(*args, **kwargs)
     71                           FutureWarning)
     72         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 73         return f(**kwargs)
     74     return inner_f
     75 

/Users/markp/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/sklearn/utils/validation.py in check_array(array, accept_sparse, accept_large_sparse, dtype, ord
ce_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator)
    644         if force_all_finite:
    645             _assert_all_finite(array,
--> 646                                allow_nan=force_all_finite == 'allow-nan')
    647 
    648     if ensure_min_samples > 0:

/Users/markp/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/sklearn/utils/validation.py in _assert_all_finite(X, allow_nan, msg_dtype)
     98                     msg_err.format
     99                     (type_err,
--> 100                      msg_dtype if msg_dtype is not None else X.dtype)
    101             )
    102     # for object dtype data, we only check for NaNs (GH-13254)

ValueError: Input contains NaN, infinity or a value too large for dtype('float64').
```

In [ ]:

Another attempt at a pipeline with SMOTE

In [69]:
```
# Assemble the Pipeline (no error messages from SMOTE, BUT not applying it either?

def classif_report1 (model):
    imputer = KNNImputer()
    scaler = StandardScaler()
    xmoter = SMOTE()
    pipeline = Pipeline(steps=[('i', imputer), ('s', scaler), ('xm', xmoter),('m', model)])
    pipeline.fit(X_train, y_train)

    training_preds = pipeline.predict(X_train)
    test_preds = pipeline.predict(X_test)

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

In [70]:
```
# Assign the model... change this for each model to run.
model = KNeighborsClassifier()
```

In [71]:
```
# Run the function with the assigned model above.
classif_report1 (model)
```

```
--------------------------------------------------------------------------
MODEL: KNeighborsClassifier()

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

           0       0.99      0.74      0.84     14645
           1       0.50      0.96      0.66      4049

    accuracy                           0.79     18694
   macro avg       0.74      0.85      0.75     18694
weighted avg       0.88      0.79      0.80     18694

--------------------------------------------------------------------------
--------------------------------------------------------------------------
Confusion Matrix - TRAIN
--------------------------------------------------------------------------
Predicted      0     1    All
True
0          10805  3840  14645
1            151  3898   4049
All        10956  7738  18694
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.90      0.65      0.75      6388
           1       0.34      0.71      0.46      1625

    accuracy                           0.66      8013
   macro avg       0.62      0.68      0.61      8013
weighted avg       0.78      0.66      0.69      8013

--------------------------------------------------------------------------
--------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------
Predicted     0     1   All
True
0          4132  2256  6388
1           471  1154  1625
All        4603  3410  8013
--------------------------------------------------------------------------
```

In [74]:
```
# Assign the model... change this for each model to run.
model = XGBClassifier()
```

In [75]:
```
# Run the function with the assigned model above.
classif_report1 (model)
```

```
------------------------------------------------------------------------
MODEL: XGBClassifier()

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

           0       0.88      0.91      0.90     14645
           1       0.64      0.56      0.60      4049

    accuracy                           0.84     18694
   macro avg       0.76      0.74      0.75     18694
weighted avg       0.83      0.84      0.83     18694

------------------------------------------------------------------------
------------------------------------------------------------------------
Confusion Matrix - TRAIN
------------------------------------------------------------------------
Predicted      0     1    All
True
0          13368  1277  14645
1           1773  2276   4049
All        15141  3553  18694
------------------------------------------------------------------------
------------------------------------------------------------------------
------------------------------------------------------------------------
Classification Report - TEST
------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.88      0.91      0.90      6388
           1       0.60      0.53      0.56      1625

    accuracy                           0.83      8013
   macro avg       0.74      0.72      0.73      8013
weighted avg       0.83      0.83      0.83      8013

------------------------------------------------------------------------
------------------------------------------------------------------------
Confusion Matrix - TEST
------------------------------------------------------------------------
Predicted     0     1   All
True
0          5801   587  6388
1           762   863  1625
All        6563  1450  8013
------------------------------------------------------------------------
```

In [ ]:

Run Models for Seasonal Vaccination Target

Will use the seasonal vaccination as the target, so need to reassign Y, and then run the 3 models.. classes are fairly balanced, so for this data, SMOTE is not necess

In [34]:
```
print(df_7.shape)
df_7.head()
```

```
(26707, 37)
```

Out[34]:

	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 [35]:
```
df_9 = df_7.drop(columns=['Unnamed: 0'], axis=1)
df_9.shape
```

Out[35]: `(26707, 36)`

In [44]:
```
print(df_9['seasonal_vaccine'].value_counts())
df_9['seasonal_vaccine'].value_counts(normalize=True)*100
```

```
0    14272
1    12435
Name: seasonal_vaccine, dtype: int64
```

Out[44]:
```
0    53.439173
1    46.560827
Name: seasonal_vaccine, dtype: float64
```

In [36]:
```
# Need to split data into X and y dataframes.
y1 = df_9['seasonal_vaccine']
X1 = df_9.drop(columns=['h1n1_vaccine', 'seasonal_vaccine'], axis=1)
print(y1.shape)
print(X1.shape)
```

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

In [45]:
```
# Create train and test sets.
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.30, random_state=35)
print(X1_train.shape)
print(X1_test.shape)

print(y1_train.shape)
print(y1_test.shape)
```

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

In [46]:
```
# RUN a DUMMY Classifier as a baseline
dclf = DummyClassifier(strategy='stratified', random_state=16)
dclf.fit(X1_train, y1_train)
dclf.score(X1_test, y1_test)
```

Out[46]: `0.49831523773867464`

Run classification models (target = seasonal vacc) - data prep A.

```
This run of models uses the "classif_report" function to run the pipeline.
A) KNN
B) XGBoost
C) Random Forest (plus feature importance)
D) Decion Trees
E) Logistic Regression
F) SVC
```

In [102]:
```python
# Assemble the Pipeline

def classif_report (model):
    imputer = KNNImputer()
    scaler = StandardScaler()
    pipeline = Pipeline(steps=[('i', imputer), ('s', scaler), ('m', model)])
    pipeline.fit(X1_train, y1_train)

    training_preds = pipeline.predict(X1_train)
    test_preds = pipeline.predict(X1_test)

    # Get results
    print('---------------------------------------------------------------------------')
    print(f'MODEL: {model}')
    print('\nClassification Report - TRAIN')
    print('---------------------------------------------------------------------------')
    print(classification_report(y1_train, training_preds))
    print('---------------------------------------------------------------------------')
    # Confusion Matrix
    print('---------------------------------------------------------------------------')
    print('Confusion Matrix - TRAIN')
    print('---------------------------------------------------------------------------')
    print(pd.crosstab(y1_train, training_preds, rownames=['True'], colnames=['Predicted'], margins=True))
    print('---------------------------------------------------------------------------')
    print('---------------------------------------------------------------------------')
    # Classification Report
    print('---------------------------------------------------------------------------')
    print('Classification Report - TEST')
    print('---------------------------------------------------------------------------')
    print(classification_report(y1_test, test_preds))
    print('---------------------------------------------------------------------------')
    # Confusion Matrix
    print('---------------------------------------------------------------------------')
    print('Confusion Matrix - TEST')
    print('---------------------------------------------------------------------------')
    print(pd.crosstab(y1_test, test_preds, rownames=['True'], colnames=['Predicted'], margins=True))
    print('---------------------------------------------------------------------------')
```

In [48]:
```python
# Assign the model... change this for each model to run.
model = KNeighborsClassifier()
```

In [49]:
```python
# Run the function with the assigned model above.
classif_report (model)
```

```
---------------------------------------------------------------------------
MODEL: KNeighborsClassifier()

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

           0       0.82      0.83      0.83      9946
           1       0.81      0.79      0.80      8748

    accuracy                           0.81     18694
   macro avg       0.81      0.81      0.81     18694
weighted avg       0.81      0.81      0.81     18694

---------------------------------------------------------------------------
---------------------------------------------------------------------------
Confusion Matrix - TRAIN
---------------------------------------------------------------------------
Predicted      0     1    All
True
0           8278  1668   9946
1           1837  6911   8748
All        10115  8579  18694
---------------------------------------------------------------------------
---------------------------------------------------------------------------
---------------------------------------------------------------------------
Classification Report - TEST
---------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.73      0.74      0.74      4326
           1       0.69      0.68      0.68      3687

    accuracy                           0.71      8013
   macro avg       0.71      0.71      0.71      8013
weighted avg       0.71      0.71      0.71      8013

---------------------------------------------------------------------------
---------------------------------------------------------------------------
Confusion Matrix - TEST
---------------------------------------------------------------------------
Predicted     0     1   All
True
0          3204  1122  4326
1          1183  2504  3687
All        4387  3626  8013
---------------------------------------------------------------------------
```

In [103]:
```python
# Assign the model... change this for each model to run.
model = XGBClassifier()
```

In [104]:
```python
# Run the function with the assigned model above.
classif_report (model)
```

```
--------------------------------------------------------------------------
MODEL: XGBClassifier()

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

           0       0.80      0.82      0.81      9946
           1       0.79      0.76      0.77      8748

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

--------------------------------------------------------------------------
--------------------------------------------------------------------------
Confusion Matrix - TRAIN
--------------------------------------------------------------------------
Predicted      0     1    All
True
0           8154  1792   9946
1           2080  6668   8748
All        10234  8460  18694
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.79      0.81      0.80      4326
           1       0.77      0.75      0.76      3687

    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     1   All
True
0          3499   827  4326
1           925  2762  3687
All        4424  3589  8013
--------------------------------------------------------------------------
```

In [ ]:
```python
# Try and so some grid search on XGBoost
```

In [105]:
```python
# Set-up the parameter grid
param_grid = {
    'learning_rate': [0.1, 0.3],
    'max_depth': [5, 8],
    'min_child_weight': [3, 5],
    'subsample': [0.5, 0.8],
    'n_estimators': [50],
}
```

In [106]:
```python
grid_clf = GridSearchCV(model, param_grid, scoring='accuracy', cv=5, n_jobs=1)
grid_clf.fit(X1_train, y1_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]))

print('---------------------------------------------------')
print(grid_clf.best_estimator_)

training_preds = grid_clf.predict(X1_train)
test_preds = grid_clf.predict(X1_test)
training_accuracy = accuracy_score(y1_train, training_preds)
test_accuracy = accuracy_score(y1_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: 5
min_child_weight: 5
n_estimators: 50
subsample: 0.8
---------------------------------------------
XGBClassifier(max_depth=5, min_child_weight=5, n_estimators=50, subsample=0.8)

Training Accuracy: 80.05%
Validation accuracy: 78.52%
```

In [107]:
```python
# Attempt 2 at GridSearchCV on XGBoost
# Set-up the parameter grid
param_grid = {
    'learning_rate': [0.05, 0.1],
    'max_depth': [4, 7],
    'min_child_weight': [4, 5],
    'subsample': [0.6, 0.8],
    'n_estimators': [100],
}
```

In [108]:
```
grid_clf = GridSearchCV(model, param_grid, scoring='accuracy', cv=5, n_jobs=1)
grid_clf.fit(X1_train, y1_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]))

print('---------------------------------------------')
print(grid_clf.best_estimator_)

training_preds = grid_clf.predict(X1_train)
test_preds = grid_clf.predict(X1_test)
training_accuracy = accuracy_score(y1_train, training_preds)
test_accuracy = accuracy_score(y1_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: 4
n_estimators: 100
subsample: 0.6
---------------------------------------------
XGBClassifier(max_depth=4, min_child_weight=4, subsample=0.6)

Training Accuracy: 80.15%
Validation accuracy: 78.57%
```

In [52]:
```
# Assign the model... change this for each model to run.
model = RandomForestClassifier()
```

In [53]:
```
# Run the function with the assigned model above.
classif_report (model)
```

```
--------------------------------------------------------------------------
MODEL: RandomForestClassifier()

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

           0       1.00      1.00      1.00      9946
           1       1.00      1.00      1.00      8748

    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     1    All
True
0          9946     0   9946
1             0  8748   8748
All        9946  8748  18694
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.78      0.80      0.79      4326
           1       0.76      0.74      0.75      3687

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

--------------------------------------------------------------------------
--------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------
Predicted     0     1   All
True
0          3473   853  4326
1           960  2727  3687
All        4433  3580  8013
--------------------------------------------------------------------------
```

In [55]:
```python
# Take a look at feature importances - from default / vanilla model

importance = pd.DataFrame(data={'features': X1_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(20)
```

Out[55]:

	features	importance
0	opinion_seas_risk	0.098468
1	opinion_seas_vacc_effective	0.088989
2	doctor_recc_seasonal	0.085048
3	employment_industry	0.065018
4	age_group	0.060375
5	hhs_geo_region	0.049494
6	opinion_h1n1_vacc_effective	0.033921
7	opinion_h1n1_risk	0.033715
8	education	0.031279
9	opinion_seas_sick_from_vacc	0.030675
10	income_poverty	0.029536
11	opinion_h1n1_sick_from_vacc	0.028192
12	h1n1_concern	0.028158
13	health_insurance	0.025172
14	census_msa	0.024687
15	household_adults	0.023794
16	h1n1_knowledge	0.022410
17	household_children	0.021476
18	doctor_recc_h1n1	0.021177
19	employment_status	0.020544

In [56]:
```python
# Assign the model... change this for each model to run.
model = DecisionTreeClassifier()
```

In [57]:
```python
# Run the function with the assigned model above.
classif_report (model)
```

```
--------------------------------------------------------------------------
MODEL: DecisionTreeClassifier()

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

           0       1.00      1.00      1.00      9946
           1       1.00      1.00      1.00      8748

    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     1    All
True
0          9946     0   9946
1             0  8748   8748
All        9946  8748  18694
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.70      0.69      0.69      4326
           1       0.64      0.65      0.64      3687

    accuracy                           0.67      8013
   macro avg       0.67      0.67      0.67      8013
weighted avg       0.67      0.67      0.67      8013

--------------------------------------------------------------------------
--------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------
Predicted     0     1   All
True
0          2978  1348  4326
1          1295  2392  3687
All        4273  3740  8013
--------------------------------------------------------------------------
```

In [58]:
```python
# Assign the model... change this for each model to run.
model = LogisticRegression()
```

In [59]:
```
# Run the function with the assigned model above.
classif_report (model)
```

```
--------------------------------------------------------------------------
MODEL: LogisticRegression()

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

           0       0.78      0.81      0.79      9946
           1       0.77      0.74      0.76      8748

    accuracy                           0.78     18694
   macro avg       0.78      0.77      0.77     18694
weighted avg       0.78      0.78      0.78     18694

--------------------------------------------------------------------------
--------------------------------------------------------------------------
Confusion Matrix - TRAIN
--------------------------------------------------------------------------
Predicted      0     1    All
True
0           8022  1924   9946
1           2272  6476   8748
All        10294  8400  18694
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.78      0.81      0.79      4326
           1       0.76      0.73      0.75      3687

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

--------------------------------------------------------------------------
--------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------
Predicted     0     1   All
True
0          3496   830  4326
1           990  2697  3687
All        4486  3527  8013
--------------------------------------------------------------------------
```

In [60]:
```
from sklearn.svm import SVC
```

In [61]:
```
# Assign the model... change this for each model to run.
model = SVC()
```

In [62]:
```
# Run the function with the assigned model above.
classif_report (model)
```

```
--------------------------------------------------------------------------
MODEL: SVC()

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

           0       0.84      0.85      0.85      9946
           1       0.83      0.81      0.82      8748

    accuracy                           0.83     18694
   macro avg       0.83      0.83      0.83     18694
weighted avg       0.83      0.83      0.83     18694

--------------------------------------------------------------------------
--------------------------------------------------------------------------
Confusion Matrix - TRAIN
--------------------------------------------------------------------------
Predicted      0     1    All
True
0           8495  1451   9946
1           1646  7102   8748
All        10141  8553  18694
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
Classification Report - TEST
--------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.78      0.80      0.79      4326
           1       0.76      0.74      0.75      3687

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

--------------------------------------------------------------------------
--------------------------------------------------------------------------
Confusion Matrix - TEST
--------------------------------------------------------------------------
Predicted     0     1   All
True
0          3463   863  4326
1           954  2733  3687
All        4417  3596  8013
--------------------------------------------------------------------------
```

Observations on the models run on target - seasonal vaccination

About 4 of these models had similar accuracy rates, but the best was XGBoost at 0.78, with precision for class 1 at 0.77. The accuracy is lower than was seen for the variable, and my guess is that this is due to the fact that the seasonal target classes are balanced and perhaps not as prone to overfitting.

In [ ]:

APPENDIX:

In [ ]:
```
# Melody grid search example:
grid_xgb2 = GridSearchCV(xgb_clf, xgb_param_grid2, scoring='accuracy', cv=None, n_jobs=1)
grid_xgb2.fit(X_train_encoded_cleaned_SMOTE, y_train_SMOTE)
```

In [ ]:
```
# Individual pieces of pre-processing for reference:

# KNNImputer
imputer = KNNImputer(n_neighbors=5)
raw5 = pd.DataFrame(imputer.fit_transform(raw5),columns = raw5.columns)

# Scaler
scale = StandardScaler().fit(X_train)

# SMOTE
sm = SMOTE(random_state = 2)
X_train, y_train = sm.fit_sample(X_train, y_train)
```

In [ ]:
```
# Another example of pipeline with SMOTE as part... different type of pipeline.
from imblearn.pipeline import Pipeline
model = Pipeline([
        ('sampling', SMOTE()),
        ('classification', LogisticRegression())
    ])

grid = GridSearchCV(model, params, ...)
grid.fit(X, y)
```

In [ ]:
```
# Yet, another version...
random_state = 38
model = Pipeline([
        ('posFeat1', featureVECTOR()),
        ('scal', StandardScaler()),

        # Original SMOTE class
        ('smote', SMOTE(random_state=random_state)),
        ('classification', SGDClassifier(loss='hinge', max_iter=1, random_state=random_state, tol=None))
    ])

# Not sure about the bottom part here...
model.fit(train_df, train_df['label'].values.tolist())
predicted = model.predict(test_df)
```

In [ ]:
```
# And one more example

pipe = make_pipeline(SMOTE(random_state=42), StandardScaler(), LinearSVC(dual=False, random_state=13))
pipe = pipe.fit(X_train, np.array(y_train))
y_pred = pipe.predict(X_test)
accuracy_1 = accuracy_score(y_test, y_pred)
```

In [ ]:
```
# Example pipeline with some grid search

sel = SelectFromModel(ExtraTreesClassifier(n_estimators=10, random_state=444),
                      threshold='mean')
clf = RandomForestClassifier(n_estimators=5000, random_state=444)

model = Pipeline([('sel', sel), ('clf', clf)])
params = {'clf__max_features': ['auto', 'sqrt', 'log2']}

gs = GridSearchCV(model, params)
gs.fit(X_train, y_train)

# How well do your hyperparameter optimizations generalize
# to unseen test data?
gs.score(X_test, y_test)
```

In [ ]:
```
```