Modeling for Predicting Presidential Elections at the County Level

In [144]:
```
#import basic tools
import pandas as pd
import numpy as np
import seaborn as sns
from tabulate import tabulate
#import pickle load and dump so pickled files can be opened and exported
from pickle import load
from pickle import dump
#set to no max columns for ease of viewing
pd.set_option("max_columns", None)
#for preprocessing
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
#pipeline stuff
from sklearn.pipeline import Pipeline
from pipelinehelper import PipelineHelper
from imblearn.pipeline import Pipeline as imbpipe
from sklearn.pipeline import make_pipeline
#cross validate and grid search
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_validate
from sklearn.model_selection import cross_val_predict
#Scalers
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import MaxAbsScaler
from sklearn.preprocessing import StandardScaler
#classifiers
from sklearn.svm import LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn import svm
#resampling
from imblearn.over_sampling import SMOTE
#metrics
from sklearn.metrics import balanced_accuracy_score
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
```

The target is the winner of the 2016 presidential election. Choices are Donald Trump and Hillary Clinton.

This is an imbalanced binary classification with the dominant class comprising over 80% of the target. The two target options are equally important so we will be looking at balanced accuracy.

From google: Balanced accuracy is calculated as the average of the proportion corrects of each class individually.

Preprocessing for this dataset can be found in the Preprocessing Final Notebook.
Many more model iterations can be found in the 'models' folder in the repo.

First Simple Model

For a first simple model I used:
Get_dummies for categorical data
Standard scaler for numeric data
svm.SVC for a classifier with 3 kernel options.

In [152]:
```
fsm.head()
```

Out[152]:

	total_pop	total_pop_one_race	pop_white	pop_african_american	pop_native	pop_asian	pop_isl
0	58805.0	55648.0	42160.0	11445.0	217.0	881.0	
1	231767.0	216743.0	189399.0	18217.0	1582.0	2067.0	
2	25223.0	24523.0	11317.0	11933.0	116.0	117.0	
3	22293.0	21534.0	16555.0	4413.0	60.0	32.0	
4	59134.0	55478.0	50663.0	845.0	337.0	178.0	

In [155]:
```
len(fsm.columns)
```

Out[155]: 71

In [2]:
```
#load in the finalized dataset
fsm = load(open('PICKLES/df_all.pkl', 'rb'))
```

In [3]:
```
#drop county columns because they were only left in for sorting verification
#drop all but one state column. We'll one hot encode the one left
#drop id because it is irrelevant
#drop 2016 total votes because it will cause data leakage
fsm = fsm.drop(['County_x', 'State_x', 'County_y', 'state', 'county', 'id'], axis
```

In [4]:
```
#get_dummies for categorical data
state_dummies = pd.get_dummies(fsm['State_y'], drop_first = True)
central_outlying = pd.get_dummies(fsm['central_outlying'], drop_first = True)
```

In [5]:
```
#drop original categorical columns
fsm = fsm.drop(['State_y', 'central_outlying'], axis = 1)
```

In [6]:
```
#join new categorical columns with dataset
fsm = pd.concat([fsm, state_dummies, central_outlying], axis = 1)
```

In [7]:
```
#map 0 and 1 to target
fsm.Target = fsm.Target.map({'Trump': 0, 'Clinton': 1})
```

In [8]:
```
#split into data and target
fsm_X = fsm.drop(['Target'], axis = 1)
fsm_y = fsm.Target
```

In [9]:
```
#train test split
fsm_X_train, fsm_X_test, fsm_y_train, fsm_y_test = train_test_split(fsm_X, fsm_y,
```

In [10]:
```
#create pipeline, parameters, and gridsearch
param = {'svc__kernel': ['rbf', 'poly', 'linear']}
pipe2 = make_pipeline(StandardScaler(), svm.SVC())
grid = GridSearchCV(pipe2, param, scoring= 'balanced_accuracy')
```

In [11]:
```
#fit the gridsearch
grid.fit(fsm_X_train, fsm_y_train)
```

Out[11]:
```
GridSearchCV(estimator=Pipeline(steps=[('standardscaler', StandardScaler()),
                                       ('svc', SVC())]),
             param_grid={'svc__kernel': ['rbf', 'poly', 'linear']},
             scoring='balanced_accuracy')
```

In [12]:
```
#best balanced accuracy score
grid.best_score_
```

Out[12]:
```
0.7682057448429769
```

In [13]:
```
#best parameters from grid search
grid.best_params_
```

Out[13]:
```
{'svc__kernel': 'linear'}
```

In [14]:
```
#take a look at predictions and confusion matrix
estimator = make_pipeline(StandardScaler(), svm.SVC(kernel = 'linear'))
```

In [15]:
```
fsm_y_hat_train = cross_val_predict(estimator, fsm_X_train, fsm_y_train)
```

In [16]:
```
confusion_matrix(fsm_y_train, fsm_y_hat_train)
```

Out[16]:
```
array([[1969,   20],
       [ 166,  200]], dtype=int64)
```

We're missing quite a few of Clinton's counties.
Best score is 76.8%.
Best kernel for svm.SVC is 'linear'.

Large Grid

Create a large grid search testing different scalers, different classifiers, and different parameters for those classifiers.

In [17]:
```
#load in the finalized dataset.
df = load(open('PICKLES/df_all.pkl', 'rb'))
```

In [18]:
```
#drop county columns because they were only left in for sorting verification
#drop all but one state column. We'll one hot encode the one left
#drop id because it is irrelevant
#drop 2016 total votes because it will cause data leakage
df = df.drop(['County_x', 'State_x', 'County_y', 'state', 'county', 'id', '2016_t
```

In [19]:
```
#map 0 and 1 to the target.
df.Target = df.Target.map({'Trump': 0, 'Clinton': 1})
```

In [20]:
```
#split into data and target
X = df.drop(['Target'], axis = 1)
y = df.Target
```

In [21]:
```
#preprocess numeric and categorical data
#build a pipeline for scaler options for numeric data
#one hot encode categorical columns.
numeric_features = list(X.select_dtypes('float64').columns)
```

In [22]:
```
numeric_transformer = Pipeline(steps = [('scaler', PipelineHelper([
        ('std', StandardScaler()),
        ('max', MaxAbsScaler()),
        ('minmax', MinMaxScaler())]))])
```

In [23]:
```
categorical_features = list(X.select_dtypes('object').columns)
```

In [24]:
```
categorical_transformer = OneHotEncoder(handle_unknown = 'ignore')
```

In [25]:
```
preprocessor = ColumnTransformer(transformers = [
    ('num', numeric_transformer, numeric_features),
    ('cat', categorical_transformer, categorical_features)
])
```

In [26]:
```
#create a pipeline with preprocessing and classifier options.
pipe = Pipeline([
    ('preprocess', preprocessor),
    ('classifier', PipelineHelper([
        ('svm', LinearSVC()),
        ('rf', RandomForestClassifier()),
        ('logreg', LogisticRegression()),
        ('dt', DecisionTreeClassifier())
    ])),
])
```

In [27]:
```
#try different paramaters for the classifiers.
params = {
    'classifier__selected_model': pipe.named_steps['classifier'].generate({
        'svm__class_weight': [None, 'balanced'],
        'rf__max_depth': [None, 5, 10, 30],
        'rf__class_weight': [None, 'balanced'],
        'rf__n_estimators': [100, 20],
        'logreg__penalty': [None, 'l1', 'l2', 'elasticnet'],
        'logreg__C': [0.1, 1.0],
        'logreg__class_weight': [None, 'balanced'],
        'logreg__solver': ['lbfgs', 'liblinear', 'sag', 'saga'],
        'dt__class_weight': [None, 'balanced']
    })
}
```

In [28]:
```
#train test split
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify = y, random_st
```

In [29]:
```
#build grid search
grid = GridSearchCV(pipe, params, scoring= 'balanced_accuracy')
```

Fit grid search.

In [30]:
```
#fit grid search
grid.fit(X_train, y_train)
```

```
C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\svm\_base.p
y:976: ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\svm\_base.p
y:976: ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\svm\_base.p
y:976: ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\svm\_base.p
y:976: ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\svm\_base.p
y:976: ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
```

In [31]:
```
#look at best scores and best parameters
grid.best_score_
```

Out[31]:
```
0.9515968900894632
```

In [32]:
```
grid.best_params_
```

Out[32]:
```
{'classifier__selected_model': ('logreg',
  {'C': 0.1, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'lbfgs'})}
```

Score: 95.2%
Best classifier: Logistic Regression
Best parameters for logreg:
C: 1.0
Class weight: 'balanced'
Penalty: 'l2'
Solver: 'lbfgs'

Next I will build a model with these best parameters but with scaler in a more visible format.

Try Best Params

Build a simple model with the best classifier. Scaling info did not come up on best params for large grid search so I moved it into a pipeline where we will see the results.

In [33]:
```
#load in prepared dataset
df = load(open('PICKLES/df_all.pkl', 'rb'))
```

In [34]:
```
#drop unnecessary columns
df = df.drop(['County_x', 'State_x', 'County_y', 'state', 'county', 'id', '2016_t
```

In [35]:
```
#map 0 and 1 to target
df.Target = df.Target.map({'Trump': 0, 'Clinton': 1})
```

In [36]:
```
#split into data and target
X = df.drop(['Target'], axis = 1)
y = df.Target
```

In [37]:
```
#preprocessing numeric and categorical data
#build a pipeline for scaler options for numeric data
#one hot encode categorical columns
numeric_features = list(X.select_dtypes('float64').columns)
```

In [38]:
```
numeric_transformer = Pipeline(steps = [('scaler', PipelineHelper([
        ('std', StandardScaler()),
        ('max', MaxAbsScaler()),
        ('minmax', MinMaxScaler())]))])
```

In [39]:
```
categorical_features = list(X.select_dtypes('object').columns)
```

In [40]:
```
categorical_transformer = OneHotEncoder(handle_unknown = 'ignore')
```

In [41]:
```
preprocessor = ColumnTransformer(transformers = [
    ('num', numeric_transformer, numeric_features),
    ('cat', categorical_transformer, categorical_features)
])
```

In [42]:
```
#create a pipeline with preprocessing, logreg, and scaling options
pipe = Pipeline([
    ('preprocess', preprocessor),
    ('scaler', PipelineHelper([
        ('std', StandardScaler()),
        ('max', MaxAbsScaler()),
        ('minmax', MinMaxScaler())
    ])),
    ('logreg', LogisticRegression()),
])
```

In [43]:
```
#try different paramaters for the classifiers and scalers
params = {
    'scaler__selected_model': pipe.named_steps['scaler'].generate({
        'std__with_mean': [True, False],
        'std__with_std': [True, False],
    }),
        'logreg__penalty': [None, 'l1', 'l2', 'elasticnet'],
        'logreg__C': [0.1, 1.0],
        'logreg__class_weight': [None, 'balanced'],
        'logreg__solver': ['lbfgs', 'liblinear', 'sag', 'saga']
}
```

In [44]:
```
#train test split
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify = y, random_st
```

In [45]:
```
#build grid search
grid = GridSearchCV(pipe, params, scoring= 'balanced_accuracy')
```

In [46]:
```
#fit grid search
grid.fit(X_train, y_train)
```

```
C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\model_selec
tion\_validation.py:548: FitFailedWarning: Estimator fit failed. The score on
this train-test partition for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\mod
el_selection\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\pip
eline.py", line 330, in fit
    Xt = self._fit(X, y, **fit_params_steps)
  File "C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\pip
eline.py", line 292, in _fit
    X, fitted_transformer = fit_transform_one_cached(
  File "C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\joblib\memo
ry.py", line 352, in __call__
    return self.func(*args, **kwargs)
  File "C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\pip
eline.py", line 740, in _fit_transform_one
    res = transformer.fit_transform(X, y, **fit_params)
```

In [47]:
```
#look at best scores and best parameters
grid.best_score_
```

Out[47]:
```
0.9515968900894632
```

In [48]:
```
grid.best_params_
```

Out[48]:
```
{'logreg__C': 0.1,
 'logreg__class_weight': 'balanced',
 'logreg__penalty': 'l2',
 'logreg__solver': 'lbfgs',
 'scaler__selected_model': ('std', {'with_mean': False, 'with_std': False})}
```

This successfully recreated the best model from the large grid search but this time we can see the scaling.

Score: 95.2%
Classifier: Logistic Regression
Parameters for logreg:
C: 1.0
Class weight: 'balanced'
Penalty: 'l2'
Solver: 'lbfgs'
Best scaler: Standard Scaler
Parameters for Standard Scaler:
with_mean = False
with_std = False

Next we'll take a look at the model without scaling.

Third attempt: No Scaling

Recreate the model above but with no scaling.

In [49]:
```
#load in the finalized dataset.
df = load(open('PICKLES/df_all.pkl', 'rb'))
```

In [50]:
```
#drop unneccesary columns
df = df.drop(['County_x', 'State_x', 'County_y', 'state', 'county', 'id', '2016_t
```

In [51]:
```
#map 0 and 1 to the target.
df.Target = df.Target.map({'Trump': 0, 'Clinton': 1})
```

In [52]:
```
#split into data and target
X = df.drop(['Target'], axis = 1)
y = df.Target
```

In [53]:
```
#preprocessing numeric and categorical data
#built a pipeline for scaler options for numeric data
#one hot encode categorical columns
categorical_features = list(X.select_dtypes('object').columns)
```

In [54]:
```
categorical_transformer = OneHotEncoder(handle_unknown = 'ignore')
```

In [55]:
```
preprocessor = ColumnTransformer(transformers = [
    ('cat', categorical_transformer, categorical_features)
])
```

In [56]:
```
#create a pipeline with preprocessing and logreg
pipe = Pipeline([
    ('preprocess', preprocessor),
    ('logreg', LogisticRegression()),
])
```

In [57]:
```
#try different paramaters for logreg
params = {
        'logreg__penalty': [None, 'l1', 'l2', 'elasticnet'],
        'logreg__C': [0.1, 1.0],
        'logreg__class_weight': [None, 'balanced'],
        'logreg__solver': ['lbfgs', 'liblinear', 'sag', 'saga']
    }
```

In [58]:
```
#train test split
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify = y, random_st
```

In [59]:
```
#build grid search
grid = GridSearchCV(pipe, params, scoring= 'balanced_accuracy')
```

In [60]:
```
#fit grid search
grid.fit(X_train, y_train)
```

```
C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\model_selec
tion\_validation.py:548: FitFailedWarning: Estimator fit failed. The score on
this train-test partition for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\mod
el_selection\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\pip
eline.py", line 335, in fit
    self._final_estimator.fit(Xt, y, **fit_params_last_step)
  File "C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\lin
ear_model\_logistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\lin
ear_model\_logistic.py", line 438, in _check_solver
    raise ValueError("Logistic Regression supports only penalties in %s,"
ValueError: Logistic Regression supports only penalties in ['l1', 'l2', 'elas
ticnet', 'none'], got None.
```

In [61]:
```
#look at best scores and best parameters
grid.best_score_
```

Out[61]:
```
0.7148898155292918
```

In [62]:
```
grid.best_params_
```

Out[62]:
```
{'logreg__C': 1.0,
 'logreg__class_weight': 'balanced',
 'logreg__penalty': 'l2',
 'logreg__solver': 'saga'}
```

This successfully recreated the best model from the large grid search but this time without scaling.

Score: 71.5%
Classifier: Logistic Regression
Parameters for logreg:
C: 1.0
Class weight: 'balanced'
Penalty: 'l2'
Solver: 'lbfgs'

This confirmed that scaling is helping the model.

SMOTE

Recreate best model with scaling and SMOTE.

In [63]:
```
#load in the finalized dataset.
df = load(open('PICKLES/df_all.pkl', 'rb'))
```

In [64]:
```
#drop county columns because they were only left in for sorting verification
#drop all but one state column. We'll one hot encode the one left
#drop id because it is irrelevant
#drop 2016 total votes because it will cause data leakage
df = df.drop(['County_x', 'State_x', 'County_y', 'state', 'county', 'id', '2016_t
```

In [65]:
```
#map 0 and 1 to the target
df.Target = df.Target.map({'Trump': 0, 'Clinton': 1})
```

In [66]:
```
#split into data and target
X = df.drop(['Target'], axis = 1)
y = df.Target
```

In [67]:
```
#preprocess numeric and categorical data
#build a pipeline for scaler options for numeric data
#one hot encode categorical columns
numeric_features = list(X.select_dtypes('float64').columns)
```

In [68]:
```
categorical_features = list(X.select_dtypes('object').columns)
```

In [69]:
```
categorical_transformer = OneHotEncoder(handle_unknown = 'ignore')
```

In [70]:
```
numeric_transformer = StandardScaler(with_mean = False, with_std = False)
```

In [71]:
```
preprocessor = ColumnTransformer(transformers = [
    ('num', numeric_transformer, numeric_features),
    ('cat', categorical_transformer, categorical_features)
])
```

In [72]:
```
#create a pipeline with preprocessing, SMOTE, and classifier
pipe = imbpipe(steps = [
    ('preprocess', preprocessor),
    ('smote', SMOTE()),
    ('logreg', LogisticRegression(C = 1.0, class_weight = 'balanced', penalty = '
])
```

In [73]:
```
#train test split
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify = y, random_st
```

In [74]:
```
#fit pipeline
pipe.fit(X_train, y_train)
```

```
C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\linear_model
\_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  n_iter_i = _check_optimize_result(
```

Out[74]:
```
Pipeline(steps=[('preprocess',
                 ColumnTransformer(transformers=[('num',
                                                  StandardScaler(with_mean=Fals
e,
                                                                 with_std=Fals
e),
                                                  ['total_pop',
                                                   'total_pop_one_race',
                                                   'pop_white',
                                                   'pop_african_american',
                                                   'pop_native', 'pop_asian',
                                                   'pop_islander', 'pop_other',
                                                   'total_pop_two_races',
                                                   'Obama', 'Romney',
                                                   '2012_total_votes',
                                                   'Density', 'poverty_total',
                                                   'poverty_under_18',
                                                   'median_household_income']),
                                                 ('cat',
                                                  OneHotEncoder(handle_unknown
='ignore'),
                                                  ['State_y',
                                                   'central_outlying'])])),
                ('smote', SMOTE()),
                ('logreg', LogisticRegression(class_weight='balanced'))])
```

In [75]:
```
#look at cross validate results
results = cross_validate(pipe, X_train, y_train, return_train_score = True, scori
```

```
C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\linear_mode
l\_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\linear_mode
l\_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\linear_mode
l\_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\linear_mode
l\_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\linear_mode
l\_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
```

```
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

In [76]:
```
#test score
results['test_score'].mean()
```

Out[76]:
```
0.9472996927091699
```

In [77]:
```
#train score
results['train_score'].mean()
```

Out[77]:
```
0.9495190736460529
```

In [78]:
```
#make cross val predictions
y_hat_train = cross_val_predict(pipe, X_train, y_train)
```

```
C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\linear_mode
l\_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\linear_mode
l\_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\linear_mode
l\_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\linear_mode
l\_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\linear_mode
l\_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
```

```
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

In [79]:
```
#confusion matrix
confusion_matrix(y_train, y_hat_train)
```

Out[79]:
```
array([[1914,   75],
       [  25,  341]], dtype=int64)
```

This successfully recreated the best model from the large grid search but this time with scaling and SMOTE.

Score: 95.1%
Classifier: Logistic Regression
Parameters for logreg:
C: 1.0
Class weight: 'balanced'
Penalty: 'l2'
Solver: 'lbfgs'
Best scaler: Standard Scaler
Parameters for Standard Scaler:
with_mean = False
with_std = False
Resampling: SMOTE

The score went down slightly. SMOTE does not seem to be helping.

Using Holdout Test Set on Final Model

Final Model:
Classifier: Logistic Regression
Parameters for logreg:
C: 1.0
Class weight: 'balanced'
Penalty: 'l2'
Solver: 'lbfgs'
Scaler: Standard Scaler
Parameters for Standard Scaler:
with_mean = False
with_std = False

In [80]:
```
#Load in prepared dataset
df = load(open('PICKLES/df_all.pkl', 'rb'))
```

In [81]:
```
#drop unnecessary columns
df = df.drop(['County_x', 'id', 'State_x', 'County_y', 'State_y', 'county', '2016
```

In [82]:
```
#get_dummies for categorical data
state_dummies = pd.get_dummies(df['state'], drop_first = True)
central_outlying = pd.get_dummies(df['central_outlying'], drop_first = True)
```

In [83]:
```
#drop original categorical columns
df = df.drop(['state', 'central_outlying'], axis = 1)
```

In [84]:
```
#join new categorical columns with dataset
df = pd.concat([df, state_dummies, central_outlying], axis = 1)
```

In [85]:
```
#map 0 and 1 to target
df.Target = df.Target.map({'Trump': 0, 'Clinton': 1})
```

In [86]:
```
#split into data and target
X = df.drop(['Target'], axis = 1)
y = df.Target
```

In [87]:
```
#train test split
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify = y, random_st
```

In [88]:
```
#instantiate standard scaler
scaler = StandardScaler(with_mean = False, with_std = False)
```

In [89]:
```
#fit and transform X_train
X_train_sc = scaler.fit_transform(X_train)
```

In [90]:
```
#transform X_test
X_test_sc = scaler.transform(X_test)
```

In [91]:
```
#instantiate the model
model = LogisticRegression(class_weight = 'balanced')
```

In [92]:
```
#fit the model
model.fit(X_train_sc, y_train)
```

```
C:\Users\angie\anaconda3\envs\learn-env\lib\site-packages\sklearn\linear_model
\_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  n_iter_i = _check_optimize_result(
```

Out[92]:
```
LogisticRegression(class_weight='balanced')
```

In [93]:
```
#make predictions on hold out test set
yhat = model.predict(X_test)
```

In [94]:
```
bal_acc = balanced_accuracy_score(y_test, yhat)
```

In [95]:
```
#score on test set
bal_acc
```

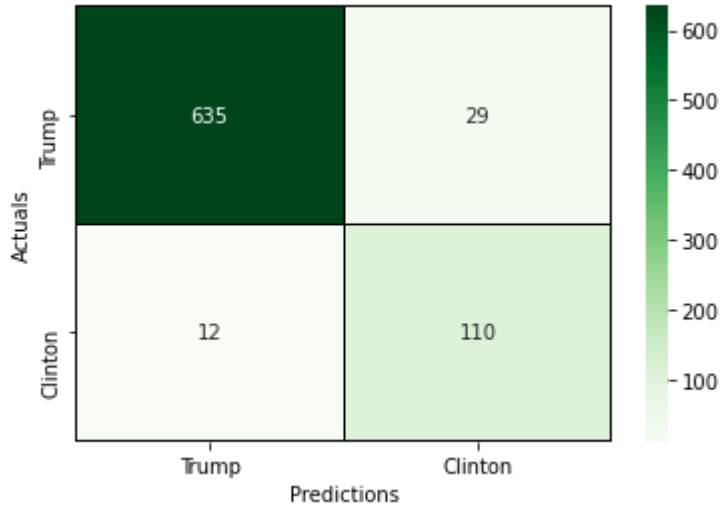
Out[95]:
```
0.9289823227335572
```

In [113]:
```
cf_matrix = confusion_matrix(y_test, yhat);
cf_matrix
```

Out[113]:
```
array([[635,  29],
       [ 12, 110]], dtype=int64)
```

In [156]:
```
ax = sns.heatmap(cf_matrix, xticklabels= ['Trump', 'Clinton'],
            yticklabels = ['Trump', 'Clinton'],
            annot = True, cmap = 'Greens', fmt="d",
           linewidths= .1, linecolor= 'black')

ax.set(xlabel="Predictions",
      ylabel="Actuals",);
```



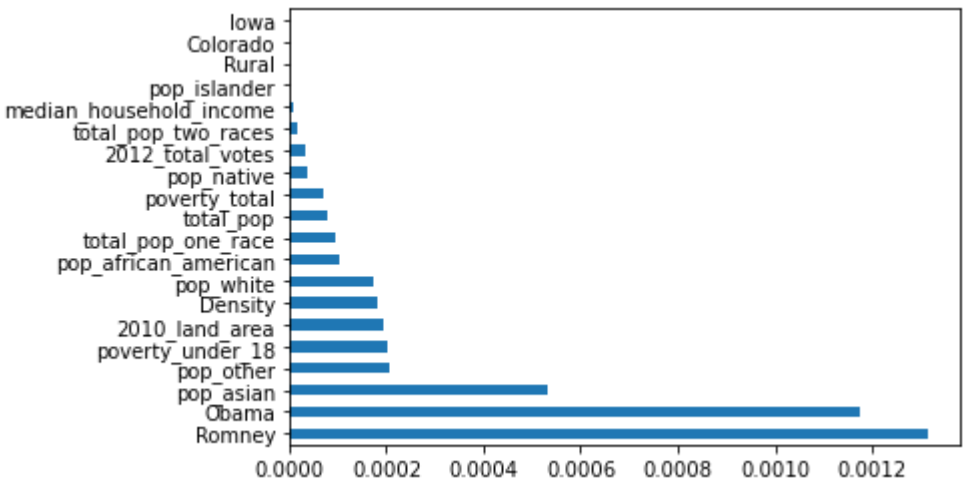
In [97]:
```
#function for looking at feature importances
def f_importances(coef, names):
    imp = coef
    imp,names = zip(*sorted(zip(imp,names)))
    plt.barh(range(len(names)), imp, align='center')
    plt.yticks(range(len(names)), names)
    plt.show()
```

In [98]:
```
#feature coefficients
model.coef_
```

Out[98]:
```
array([[ 7.81976985e-05,  9.41740323e-05, -1.72710847e-04,
        -1.02818563e-04,  3.91417282e-05,  5.33842661e-04,
         1.49908275e-06, -2.04780030e-04, -1.59763339e-05,
         1.17633402e-03, -1.31444212e-03, -3.47041176e-05,
        -1.95484960e-04,  1.80225660e-04,  7.10156260e-05,
        -2.01459514e-04, -7.90579365e-06, -3.95877182e-08,
        -1.89474614e-09,  2.94382352e-08,  8.39895389e-08,
         1.67288190e-07, -7.50822630e-09, -7.62109273e-09,
        -1.30545373e-12, -3.64309855e-08,  8.24212313e-08,
         1.28808764e-09,  2.03089699e-08, -4.55413848e-08,
        -3.03327178e-08, -1.31829068e-07, -5.92521683e-08,
        -4.40568027e-08,  3.10048443e-08, -2.10812833e-11,
        -5.98352854e-09,  1.59588246e-08, -6.44749786e-08,
        -4.17268674e-08,  1.13393813e-07, -3.81276343e-08,
        -1.75992075e-08, -9.63070735e-09, -8.92111020e-09,
        -1.11280853e-08,  6.14169891e-10,  7.87922851e-08,
        -8.53344146e-08,  2.96481862e-08, -4.41421993e-08,
        -3.90509992e-08, -3.09215795e-08, -1.12437458e-08,
         9.32384887e-09, -4.90074419e-09,  1.25489464e-08,
        -2.28991112e-08, -2.62686939e-08,  7.85235374e-10,
         2.94317351e-08,  3.68119659e-08,  9.20726660e-08,
         2.43202460e-08, -2.26155233e-08, -1.93025061e-08,
        -4.51433120e-09, -2.39054135e-08, -2.00047669e-07]])
```

In [99]:
```
#bar graph of feature importances
pd.Series(abs(model.coef_[0]), index=X.columns).nlargest(20).plot(kind='barh')
```

Out[99]:
```
<AxesSubplot:>
```



The final model used on the hold out test set has a score of 92.8%.

The most important features are:

Previous election results
Asian population within the county
Number of children in poverty
County density

In [ ]: