CS 418: Final Project - Classification

Authors: Anusha Sagi, Fatima Kahack, Lydia Tse

Description: In this code, we will be utilizing regression to determine the poverty and child poverty of a specified county

In [14]:

```
# Load libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import linear_model
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn import metrics
from scipy.cluster.hierarchy import linkage, fcluster
from sklearn.cluster import KMeans, DBSCAN
from sklearn.metrics import mean_squared_error
import math
```

In [15]:

```
# Load Election dataset
data_census = pd.read_csv('train_dp_output.csv')
data_census = data_census.loc[:, ~data_census.columns.str.contains('^Unnamed')]
data_census.head()
```

Out[15]:

	CountyId	State	County	TotalPop	Percent_Women	Hispanic	White	Black	Income	IncomePerCap	...	WorkAtHome	MeanCommute	PrivateWork	Public
0	1001	Alabama	Autauga County	55036.0	51.124718	2.7	75.4	18.9	55317	27824	...	2.5	25.8	74.1	
1	1003	Alabama	Baldwin County	203360.0	51.058714	4.4	83.1	9.5	52562	29364	...	5.6	27.0	80.7	
2	1007	Alabama	Bibb County	22580.0	45.744021	2.4	74.6	22.0	43404	20911	...	1.5	30.0	76.0	
3	1009	Alabama	Blount County	57667.0	50.595661	9.0	87.4	1.5	47412	22021	...	2.1	35.0	83.9	
4	1011	Alabama	Bullock County	10478.0	46.401985	0.3	21.6	75.6	29655	20856	...	3.0	29.8	81.4	

5 rows × 24 columns

In [28]:

```
all_data = data_census[['CountyId', 'State', 'County', 'TotalPop', 'Percent_Women', 'Hispanic', 'White', 'Black', 'Incom
e', 'IncomePerCap', 'Professional', 'Service', 'Production', 'Carpool', 'WorkAtHome', 'PrivateWork', 'PublicWork', 'Self
Employed', 'Unemployment']]
full_data = all_data.select_dtypes(include=[np.int64,np.float64])
full_data = full_data.iloc[:,1:17]
```

In [29]:

```
x_train_full, x_validation_full, y_train, y_validation = train_test_split(data_census[['CountyId', 'State', 'County', 'T
otalPop', 'Percent_Women', 'Hispanic', 'White', 'Black', 'Income', 'IncomePerCap', 'Professional', 'Service', 'Productio
n', 'Carpool', 'WorkAtHome', 'PrivateWork', 'PublicWork', 'SelfEmployed', 'Unemployment']], data_census['Poverty Categor
y'], test_size = 0.25, random_state = 0)
# Selecting required variables for x_train
x_train = x_train_full.select_dtypes(include=[np.int64,np.float64])
x_train = x_train.iloc[:,1:17]

# Selecting required variables for x_validation
x_validation = x_validation_full.select_dtypes(include=[np.int64,np.float64])
x_validation = x_validation.iloc[:,1:17]
```

In [31]:

```
# Standardize full dataset
scaler = StandardScaler()
scaler.fit(x_train)
full_data_scaled = scaler.transform(full_data)
full_data_scaled_df = pd.DataFrame(full_data_scaled, index=full_data.index, columns=full_data.columns)

# Selecting required variables for x_validation
x_train_scaled = scaler.transform(x_train)
x_validation_scaled = scaler.transform(x_validation)
x_train_scaled_df = pd.DataFrame(x_train_scaled,index = x_train.index,columns=x_train.columns)
x_validation_scaled_df = pd.DataFrame(x_validation_scaled,index = x_validation.index,columns=x_validation.columns)
```

In [25]:

```
# Classifying using best prediction of poverty in county
classifier_poverty = SVC(kernel = 'rbf')
classifier_poverty.fit(x_train_scaled_df[['Unemployment', 'Income', 'SelfEmployed', 'Black', 'White', 'Hispanic', 'Percent_Women', 'Professional']], y_train)
y_pred = classifier_poverty.predict(x_validation_scaled_df[['Unemployment', 'Income', 'SelfEmployed', 'Black', 'White', 'Hispanic', 'Percent_Women', 'Professional']])
```

In [33]:

```
best_prediction = classifier_poverty.predict(full_data_scaled_df[['Unemployment', 'Income', 'SelfEmployed', 'Black', 'White', 'Hispanic', 'Percent_Women', 'Professional']])
```

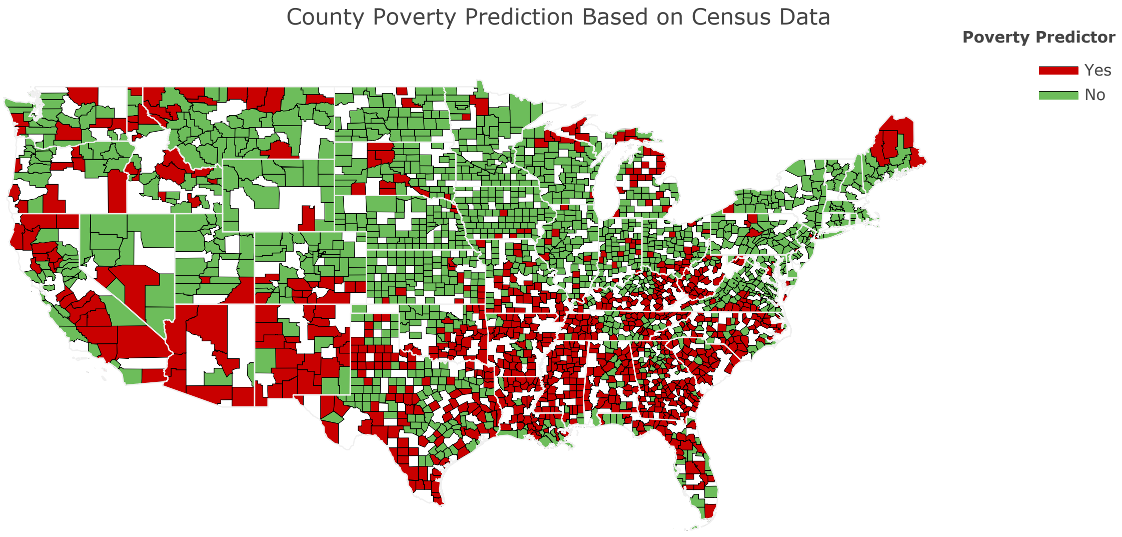
In [34]:

```
best_data = pd.DataFrame({'Poverty Prediction': best_prediction, 'County Id': all_data['CountyId']})
```

In [42]:

```
# Create a map of Democratic & Republic counties with FIPS codes based on the dataset
import plotly.figure_factory as ff
from plotly.offline import init_notebook_mode, iplot # Needed to render the figure when exporting to HTML
init_notebook_mode(connected=True)

county = best_data['County Id'].tolist()
poverty_values = best_data['Poverty Prediction'].map({0: 'No', 1: 'Yes'})
colorscale = ["#6dbd5b", "#c90000"]
figure = ff.create_choropleth(fips=county,
                              values=poverty_values,
                              colorscale=colorscale,
                              county_outline={'color': '#000000', 'width': 0.5},
                              title='County Poverty Prediction Based on Census Data',
                              legend_title='Poverty Predictor')
figure.layout.template = None
iplot(figure, validate=False) # Displaying figure even when exported to HTML
```



In [66]:

```
# Updating dataset for child poverty predictors
cx_train_full, cx_validation_full, cy_train, cy_validation = train_test_split(data_census[['CountyId', 'State', 'County', 'TotalPop', 'Percent_Women', 'Hispanic', 'White', 'Black', 'Income', 'IncomePerCap', 'Professional', 'Service', 'Production', 'Carpool', 'WorkAtHome', 'PrivateWork', 'PublicWork', 'SelfEmployed', 'Unemployment']], data_census['Child_Poverty Category'], test_size = 0.25, random_state = 0)

# Selecting required variables for x_train
cx_train = cx_train_full.select_dtypes(include=[np.int64,np.float64])
cx_train = cx_train.iloc[:,1:17]

# Selecting required variables for x_validation
cx_validation = cx_validation_full.select_dtypes(include=[np.int64,np.float64])
cx_validation = cx_validation.iloc[:,1:17]

# Standardizing the data
scaler = StandardScaler()
scaler.fit(cx_train)
cx_train_scaled = scaler.transform(cx_train)
cx_validation_scaled = scaler.transform(cx_validation)
cx_train_scaled_df = pd.DataFrame(cx_train_scaled,index=cx_train.index,columns=cx_train.columns)
cx_validation_scaled_df = pd.DataFrame(cx_validation_scaled,index=cx_validation.index,columns=cx_validation.columns)

full_data_scaled = scaler.transform(full_data)
full_data_scaled_df = pd.DataFrame(full_data_scaled, index=full_data.index, columns=full_data.columns)
```

```
In [67]: # For Predicting Child Poverty
         classifier_childPoverty = KNeighborsClassifier(n_neighbors = 13)
         classifier_childPoverty.fit(cx_train_scaled_df[['White','Hispanic','Black' ,'Unemployment', 'Income', 'WorkAtHome']], cy
         _train)
```

```
Out[67]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                              metric_params=None, n_jobs=None, n_neighbors=13, p=2,
                              weights='uniform')
```

```
In [68]: best_prediction_child = classifier_childPoverty.predict(full_data_scaled_df[['White','Hispanic','Black','Unemployment',
         'Income', 'WorkAtHome']])
```

```
In [69]: best_data_child = pd.DataFrame({'Child Poverty Prediction': best_prediction_child, 'County Id': all_data['CountyId']})
```

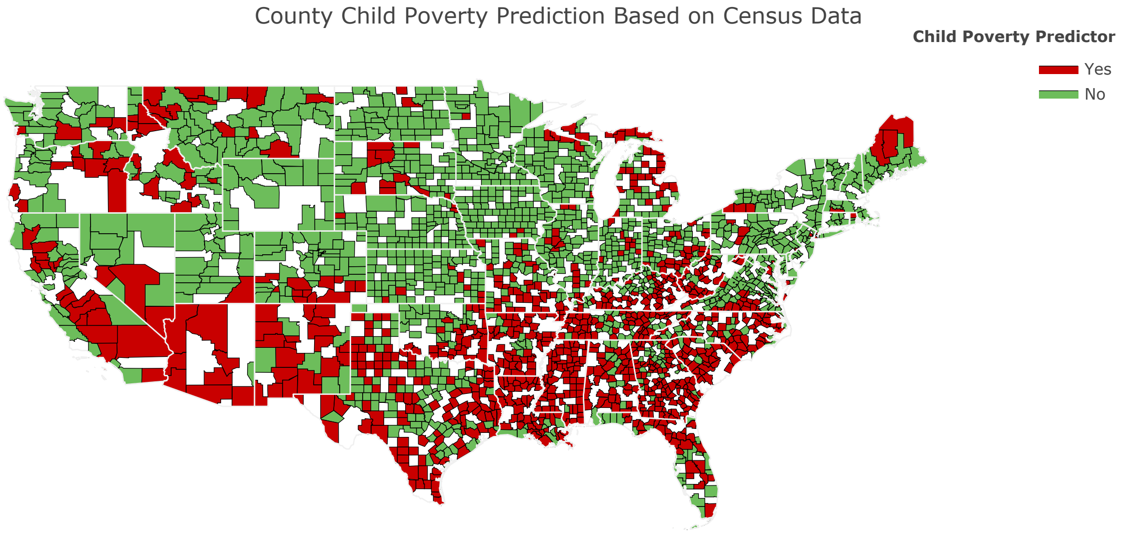
```
In [70]: child_poverty_values = best_data['Child Poverty Prediction'].map({0: 'No', 1: 'Yes'})
         colorscale = ["#6dbd5b", "#c90000"]
         figure = ff.create_choropleth(fips=county,
                                       values=child_poverty_values,
                                       colorscale=colorscale,
                                       county_outline={'color': '#000000', 'width': 0.5},
                                       title='County Child Poverty Prediction Based on Census Data',
                                       legend_title='Child Poverty Predictor')
         figure.layout.template = None
         iplot(figure, validate=False) # Displaying figure even when exported to HTML
```

```
/Users/lydia/opt/anaconda3/lib/python3.7/site-packages/pandas/core/frame.py:7123: FutureWarning:

Sorting because non-concatenation axis is not aligned. A future version
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=True'.
```



```
In [ ]:
```