When it came to machine learning, usually in biology something akin to a regression analysis is the preferred method where for a given treatment X, there exists the response variable Y and the analysis will focus on whether there correlation or a causation between X and Y (the actual order of the polynomial equation can vary). As such, in the beginning I planned on writing a machine learning algorithm akin to the one shown below:

# Import necessary modules

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

from sklearn.model\_selection import train\_test\_split

# Assign X column and y columns into their own variables

X = df['x column']

y = df['y column']

# Create training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.3, random\_state=42)

# Create the regressor: reg\_all

reg\_all = LinearRegression() #this part can also be adjusted for higher order polynomial equation, the goal is to have an equation that fits the data here

# If wanting to do a higher order polynomial instead of a straight line, do the below: (higher order can be achieved by adjusting the degree)

polynomial\_features= PolynomialFeatures(degree=2)

x\_poly = polynomial\_features.fit\_transform(x)

model = LinearRegression()

model.fit(x\_poly, y)

y\_poly\_pred = model.predict(x\_poly)

# Fit the regressor to the training data

reg\_all.fit(X\_train, y\_train)

# Predict on the test data: y\_pred

y\_pred = reg\_all.predict(X\_test)

# Compute and print R^2 and RMSE

print("R^2: {}".format(reg\_all.score(X\_test, y\_test)))

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

print("Root Mean Squared Error: {}".format(rmse))

However, the problem with this survey dataset is that there is no correlation or causation between X or Y. As such the machine learning algorithm shown above will not work, as the data here is US territories and a value associated with them (for example, the percentage of older adults who are experiencing frequent mental distress across all US territories). As such, a regression algorithm will not work here, and a clustering algorithm may be more appropriate.

For a clustering algorithm the idea is to be given a value, and then predict what state the value are most likely to come from. For this I tried a K nearest neighbors classifier, as it is the algorithm that I am the most familiar with. For each question I wanted to write an algorithm similar to the one below:

# Import KNeighborsClassifier from sklearn.neighbors

from sklearn.neighbors import KNeighborsClassifier

# Create arrays for the features and the response variable

y = df['Percentage of older adults who are experiencing frequent mental distress'].values

X = df.drop('Percentage of older adults who are experiencing frequent mental distress', axis=1).values

# Create a k-NN classifier with 6 neighbors: knn

knn = KNeighborsClassifier(n\_neighbors=6)

# Fit the classifier to the data

knn.fit(X,y)

# Predict the labels for the training data X

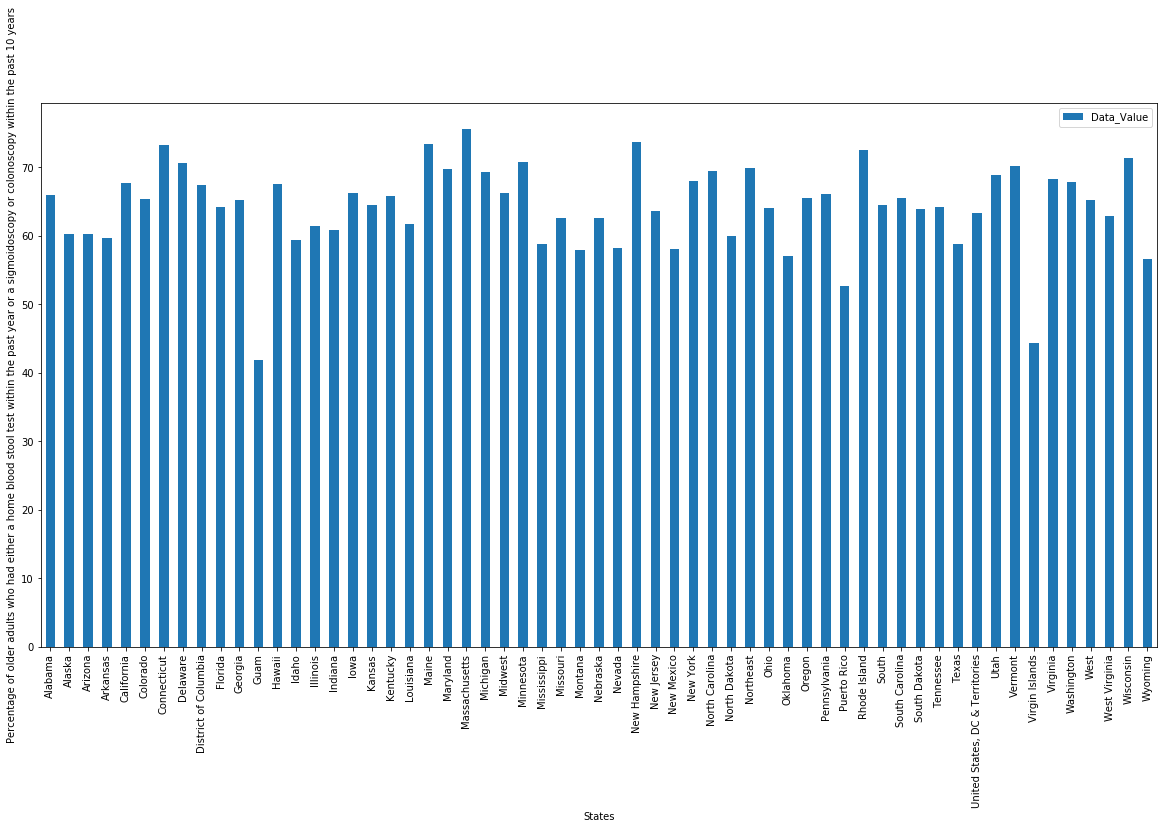
y\_pred = knn.predict(X)

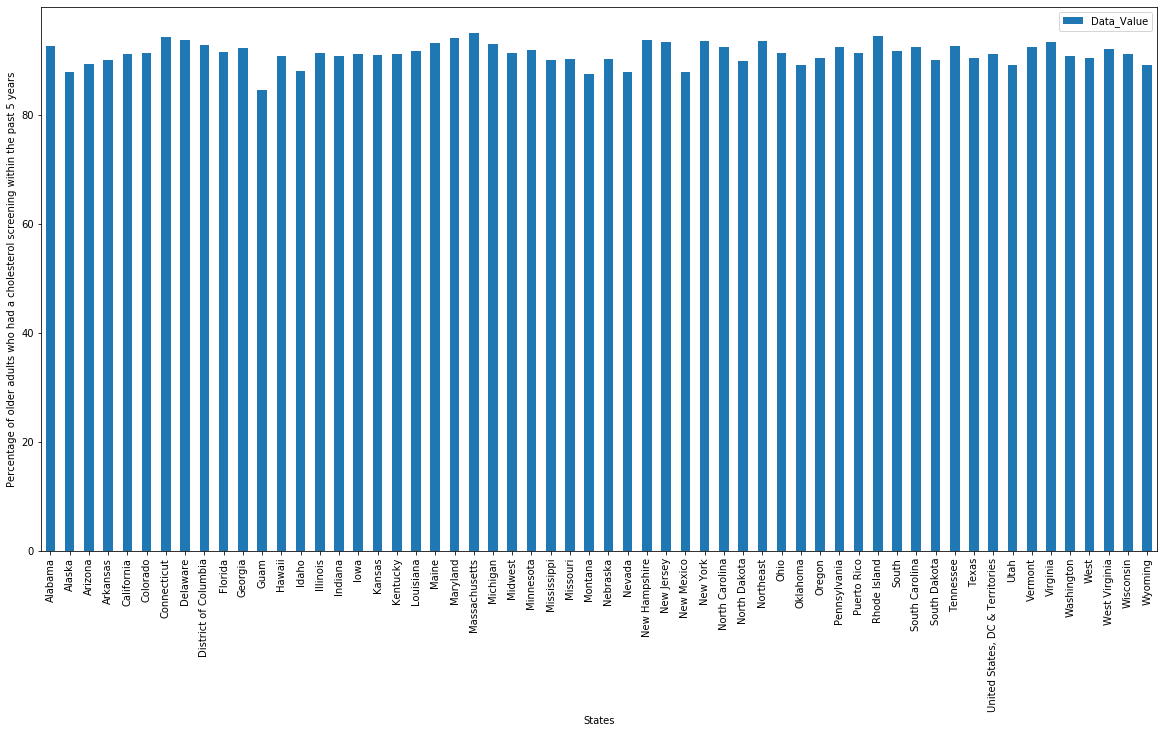
# Predict and print the label for the new data point X\_new

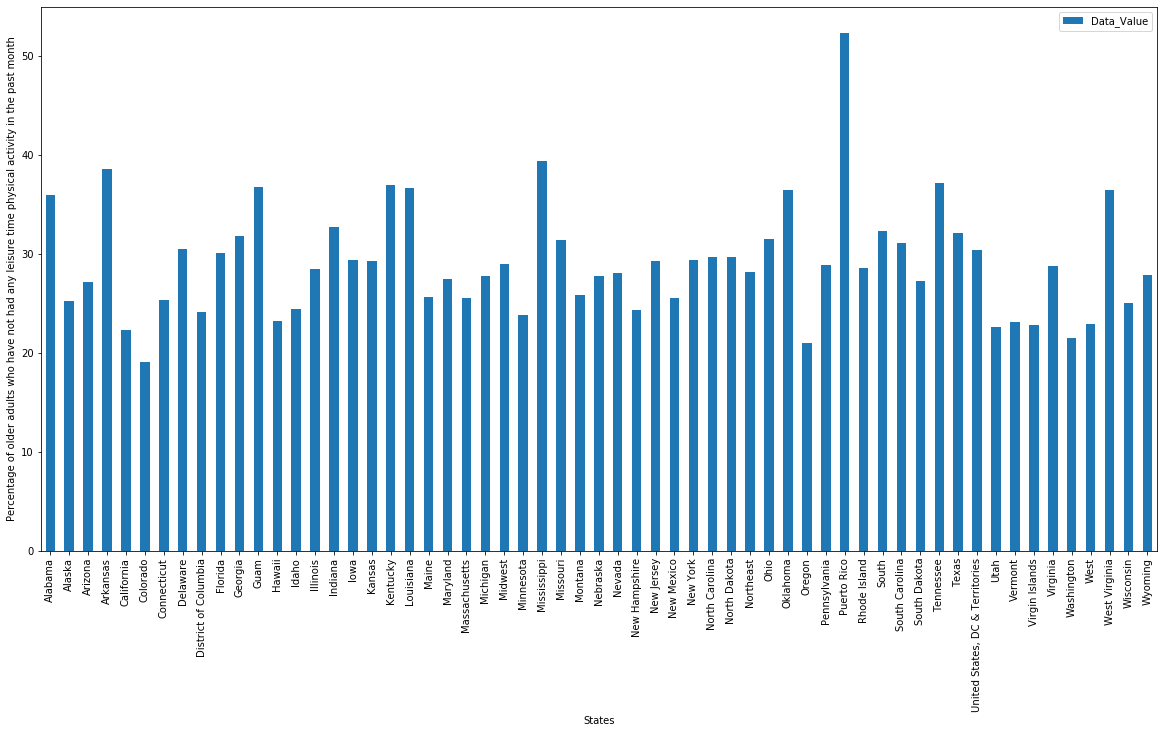
new\_prediction = knn.predict(X\_new)

print("Prediction: {}".format(new\_prediction))

However, this time another problem surfaced, in all of the questions, save for 3 US territories: Guam, US Virgin Islands and Puerto Rico, the rest of the locations are all relatively the same or irregular enough that there is no discernable pattern:







This meant that for both linear regression and classification it is basically impossible to write an algorithm that will work for any of the questions in this dataset. Proving that, once again, this survey data has extremely limited analysis value.