Prediction of United States' Counties Poverty Rates[¶](#Prediction-of-United-States'-Counties-Poverty-Rates)

Mark Vervuurt  
25-01-2018

## Executive Summary[¶](#Executive-Summary)

This document presents the results of the regression analysis to predict poverty rates of United States' Counties. The result of this regression analysis is the creation of an AdaBoostRegressor. This AdaBoostRegressor is able to predict United States' Counties poverty rates with an RMSE of 2.7853.

In the data understanding phase was discovered that the following features play a significant role in predicting United States' counties poverty rates. They have moderate positive and negative Pearson correlation coefficients. Furthermore the Boxplots show enough variation and separation of the data with respect to the target variable poverty\_rate. However recursive function elimination was ultimately used to select the optimum number of features for the lowest RMSE score.

|  |  |
| --- | --- |
| Significant Features | Short Description |
| area\_\_rucc | Rural urban continuum code of county |
| econ\_\_economic\_typology | economic dependence type of county |
| aui\_pct65y\_cat | created categorical feature combining area\_\_urban\_influence and categorical percentage of 65 years old per county |
| demo\_\_pct\_adults\_less\_than\_a\_high\_school\_diploma | percentage of adults with less than high school diploma per county |
| health\_\_homicides\_per\_100k | homicides per 100k inhabitants per county |
| econ\_\_pct\_unemployment | percentage of unemployment per county |
| health\_\_pct\_low\_birthweight | percentage of low birth weight per county |
| econ\_\_pct\_uninsured\_adults | percentage of uninsured adults per county |
| health\_\_pct\_diabetes | percentage of diabetes per county |
| demo\_\_pct\_non\_hispanic\_african\_american | percentage of African Americans per county |
| econ\_\_pct\_civilian\_labor | percentage of civilian labor per county |

The CRISP-DM Methodology was used in order to create an accurate regression model:

* **Business Understanding**: read through the '[Rural Poverty & Well-being](https://www.ers.usda.gov/topics/rural-economy-population/rural-poverty-well-being/poverty-overview.aspx)' report to better understand the circumstances of poverty.
* **Data Understanding**: explore the quantitative and categorical variables that play a key role in predicting poverty rates. Create new, better and informative features.
* **Data Preparation**: drop redundant and uninformative features, fill missing values, etc.
* **Modeling**: create and select the best regression model.
* **Evaluation**: evaluate the regression models using nested cross validation.
* **Deployment**: the deployment of the regression model is not strictly applicable here. However presenting the results of the regression analysis with this report can be considered as the deployment step.

## Business Understanding[¶](#Business-Understanding)

As described in the online report the '[Rural Poverty & Well-being](https://www.ers.usda.gov/topics/rural-economy-population/rural-poverty-well-being/poverty-overview.aspx)': "Concentrated poverty contributes to poor housing and health conditions, higher crime and school dropout rates, as well as employment dislocations". With this information the data will be explored to see how health, crime,education and employment related factors contribute to poverty.

Another important feature of poverty is time. An area that doesn't have a high level of poverty in two following years is likely better off than an area that has a high level of poverty in both years. It will not be possible to construct a feature with this information because we cannot compare the state's poverty rate over year 'a' and 'b' within this data set. We don't have a unique key to identify counties.

Counties are generally compared by their Non-Metro and Metro status. There is more poverty in Non-Metro areas than Metro areas. Poverty is also higher under certain ages and ethnicities. Here also the data will be explored on the basis of this information.

In [72]:

import re  
import bs4  
import time  
import plyfile  
import html5lib  
import multiprocessing  
import itertools  
  
import numpy as np  
import pandas as pd  
  
import seaborn as sns  
from scipy import misc  
import scipy.io.wavfile as wavfile  
  
import scipy  
from math import sqrt  
from scipy import stats  
from pprint import pprint  
from sklearn import tree  
from sklearn.svm import SVC  
from sklearn import manifold  
from tempfile import mkdtemp  
from textwrap import wrap  
from matplotlib import cm as cm  
  
import sklearn.metrics as metrics  
from pandas.plotting import scatter\_matrix  
from scipy.stats import randint as sp\_randint  
from sklearn.pipeline import TransformerMixin  
from sklearn.metrics.scorer import make\_scorer  
from sklearn.pipeline import Pipeline  
from sklearn.decomposition import PCA  
from sklearn.datasets import load\_iris  
from sklearn.pipeline import make\_pipeline  
from sklearn.preprocessing import Binarizer  
from sklearn.naive\_bayes import MultinomialNB  
from sklearn.tree import DecisionTreeRegressor, DecisionTreeClassifier  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.feature\_selection import RFECV, SelectFromModel, f\_regression, SelectKBest  
from sklearn.ensemble import RandomForestClassifier, AdaBoostRegressor, AdaBoostClassifier  
from sklearn.dummy import DummyClassifier, DummyRegressor  
from sklearn.cluster import AgglomerativeClustering, KMeans  
from sklearn.feature\_extraction.text import CountVectorizer  
from sklearn.linear\_model import LinearRegression, LassoCV, RidgeCV, Lasso, Ridge  
from sklearn.preprocessing import MaxAbsScaler, MinMaxScaler, Normalizer, RobustScaler, StandardScaler  
from sklearn.metrics import recall\_score, accuracy\_score, confusion\_matrix, roc\_curve, roc\_auc\_score, mean\_squared\_error, accuracy\_score  
from sklearn.model\_selection import train\_test\_split, GridSearchCV, RandomizedSearchCV, cross\_val\_score, cross\_validate, cross\_val\_predict, KFold, ShuffleSplit, StratifiedShuffleSplit  
  
import matplotlib  
import matplotlib.pyplot as plt  
from matplotlib import cm as cm  
from mpl\_toolkits.mplot3d import Axes3D  
from pandas.plotting import parallel\_coordinates, andrews\_curves  
  
%matplotlib inline  
matplotlib.style.use('ggplot')  
  
pd.set\_option('display.max\_columns', None)

## Data Understanding[¶](#Data-Understanding)

In order to build this regression model and determine its most significant features a thorough data exploration was done to understand the relationship between poverty rates and other features.

### Initial Data Exploration[¶](#Initial-Data-Exploration)

The dataset consists of 3198 records about United States' counties. Each record contains socioeconomic indicators about a United States' county for a given year. Besides the 'row\_id', 'yr' and the target value 'poverty\_rate', the dataset contains 32 features about socioeconomic indicators.

In [2]:

poverty\_train = pd.read\_csv('./Microsoft\_-\_DAT102x\_Predicting\_Poverty\_in\_the\_United\_States\_-\_Training\_values.csv')

In [3]:

train\_shape\_tmp = poverty\_train.shape

In [4]:

train\_dytpes\_tmp = poverty\_train.dtypes

#### Individual Feature Statistics[¶](#Individual-Feature-Statistics)

Here are the summary statistics for all the socioeconomic features:

* summary statistics of categorical variables: the total count (count), number of unique elements (unique), most frequent element (top) and the frequency of the most frequent element (frequent)
* summary statistics of quantitative variables: the mean, the standard deviation (std), the minimum value (min), 25% percentile, 50% percentile (median), 75% percentile and the maximum value (max).

In [73]:

poverty\_train.drop(columns=['row\_id'], axis=1).describe(include='all').T

Out[73]:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | count | unique | top | freq | mean | std | min | 25% | 50% | 75% | max |
| area\_\_rucc | 3198 | 9 | Nonmetro - Urban population of 2,500 to 19,999... | 608 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| area\_\_urban\_influence | 3198 | 12 | Small-in a metro area with fewer than 1 millio... | 692 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| econ\_\_economic\_typology | 3198 | 6 | Nonspecialized | 1266 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| econ\_\_pct\_civilian\_labor | 3198 | NaN | NaN | NaN | 0.467071 | 0.074541 | 0.217 | 0.42 | 0.467 | 0.514 | 1 |
| econ\_\_pct\_unemployment | 3198 | NaN | NaN | NaN | 0.0596104 | 0.0228497 | 0.008 | 0.044 | 0.057 | 0.071 | 0.24 |
| econ\_\_pct\_uninsured\_adults | 3196 | NaN | NaN | NaN | 0.217534 | 0.0673718 | 0.046 | 0.166 | 0.216 | 0.262 | 0.495 |
| econ\_\_pct\_uninsured\_children | 3196 | NaN | NaN | NaN | 0.0859202 | 0.0400046 | 0.009 | 0.057 | 0.077 | 0.105 | 0.285 |
| demo\_\_pct\_female | 3196 | NaN | NaN | NaN | 0.498781 | 0.0242508 | 0.294 | 0.493 | 0.503 | 0.512 | 0.576 |
| demo\_\_pct\_below\_18\_years\_of\_age | 3196 | NaN | NaN | NaN | 0.227763 | 0.0342909 | 0.098 | 0.207 | 0.226 | 0.24525 | 0.417 |
| demo\_\_pct\_aged\_65\_years\_and\_older | 3196 | NaN | NaN | NaN | 0.170137 | 0.0435937 | 0.043 | 0.142 | 0.167 | 0.194 | 0.355 |
| demo\_\_pct\_hispanic | 3196 | NaN | NaN | NaN | 0.0902334 | 0.142707 | 0 | 0.019 | 0.035 | 0.088 | 0.945 |
| demo\_\_pct\_non\_hispanic\_african\_american | 3196 | NaN | NaN | NaN | 0.0911167 | 0.147104 | 0 | 0.006 | 0.022 | 0.09625 | 0.855 |
| demo\_\_pct\_non\_hispanic\_white | 3196 | NaN | NaN | NaN | 0.770207 | 0.207903 | 0.06 | 0.648 | 0.854 | 0.936 | 0.998 |
| demo\_\_pct\_american\_indian\_or\_alaskan\_native | 3196 | NaN | NaN | NaN | 0.0246586 | 0.0846341 | 0 | 0.002 | 0.007 | 0.014 | 0.852 |
| demo\_\_pct\_asian | 3196 | NaN | NaN | NaN | 0.0133035 | 0.0253656 | 0 | 0.003 | 0.007 | 0.013 | 0.346 |
| demo\_\_pct\_adults\_less\_than\_a\_high\_school\_diploma | 3198 | NaN | NaN | NaN | 0.148794 | 0.0682547 | 0.016129 | 0.0974683 | 0.133501 | 0.195171 | 0.466867 |
| demo\_\_pct\_adults\_with\_high\_school\_diploma | 3198 | NaN | NaN | NaN | 0.3503 | 0.0705342 | 0.0728205 | 0.305915 | 0.355701 | 0.399197 | 0.551689 |
| demo\_\_pct\_adults\_with\_some\_college | 3198 | NaN | NaN | NaN | 0.301366 | 0.0524976 | 0.112821 | 0.265362 | 0.301595 | 0.335972 | 0.474216 |
| demo\_\_pct\_adults\_bachelors\_or\_higher | 3198 | NaN | NaN | NaN | 0.19954 | 0.0891577 | 0.013986 | 0.13884 | 0.177247 | 0.233258 | 0.794872 |
| demo\_\_birth\_rate\_per\_1k | 3198 | NaN | NaN | NaN | 11.677 | 2.73952 | 4 | 10 | 11 | 13 | 29 |
| demo\_\_death\_rate\_per\_1k | 3198 | NaN | NaN | NaN | 10.3011 | 2.78614 | 0 | 8 | 10 | 12 | 27 |
| health\_\_pct\_adult\_obesity | 3196 | NaN | NaN | NaN | 0.307599 | 0.043404 | 0.14 | 0.284 | 0.309 | 0.334 | 0.484 |
| health\_\_pct\_adult\_smoking | 2734 | NaN | NaN | NaN | 0.213519 | 0.0630903 | 0.05 | 0.171 | 0.211 | 0.24975 | 0.526 |
| health\_\_pct\_diabetes | 3196 | NaN | NaN | NaN | 0.109287 | 0.0231967 | 0.033 | 0.094 | 0.109 | 0.124 | 0.197 |
| health\_\_pct\_low\_birthweight | 3016 | NaN | NaN | NaN | 0.0835345 | 0.0223822 | 0.025 | 0.068 | 0.08 | 0.095 | 0.232 |
| health\_\_pct\_excessive\_drinking | 2220 | NaN | NaN | NaN | 0.164832 | 0.0502321 | 0.038 | 0.129 | 0.164 | 0.196 | 0.358 |
| health\_\_pct\_physical\_inacticity | 3196 | NaN | NaN | NaN | 0.277309 | 0.0529475 | 0.097 | 0.243 | 0.28 | 0.313 | 0.443 |
| health\_\_air\_pollution\_particulate\_matter | 3170 | NaN | NaN | NaN | 11.6265 | 1.54493 | 7 | 10 | 12 | 13 | 15 |
| health\_\_homicides\_per\_100k | 1231 | NaN | NaN | NaN | 5.95075 | 5.06337 | -0.39 | 2.66 | 4.84 | 7.825 | 51.49 |
| health\_\_motor\_vehicle\_crash\_deaths\_per\_100k | 2781 | NaN | NaN | NaN | 21.1161 | 10.517 | 3.09 | 13.46 | 19.63 | 26.47 | 110.45 |
| health\_\_pop\_per\_dentist | 2954 | NaN | NaN | NaN | 3431.44 | 2569.44 | 339 | 1812.25 | 2690 | 4089.75 | 28129 |
| health\_\_pop\_per\_primary\_care\_physician | 2968 | NaN | NaN | NaN | 2551.35 | 2100.48 | 189 | 1419 | 1999 | 2859 | 23400 |
| yr | 3198 | 2 | b | 1599 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

In [6]:

poverty\_labels = pd.read\_csv('Microsoft\_-\_DAT102x\_Predicting\_Poverty\_in\_the\_United\_States\_-\_Training\_labels.csv')

In [7]:

lbl\_shape\_tmp = poverty\_labels.shape

In [8]:

lbl\_dtype\_tmp = poverty\_labels.dtypes

Here are the summary statistics for the target variable which is quantitative:

In [74]:

poverty\_labels.drop(columns=['row\_id'], axis=1).describe().T

Out[74]:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | count | mean | std | min | 25% | 50% | 75% | max |
| poverty\_rate | 3198.0 | 16.817136 | 6.697969 | 2.5 | 12.0 | 15.8 | 20.3 | 47.4 |

Poverty rates are right or positively skewed with a skew value of 1.048357. We can recognize a slight bell curve in the data. The mean and median are relatively close to each other and the standard deviation is relatively low which indicates low variability in the poverty rates. Most United States' counties have a poverty\_rate between 10% and 20% poverty.

In [150]:

ht\_pov = sns.distplot(poverty\_labels.drop(columns=['row\_id'], axis=1), kde=False, norm\_hist=False, color='#1f77b4', axlabel='poverty\_rate').set\_title('Histogram poverty\_rate')

![](data:image/png;base64;base64,)

In [11]:

poverty = pd.merge(poverty\_train, poverty\_labels, on='row\_id')  
pov\_shape\_tmp = poverty.shape

From the summary statistics above, should be clear that there are three categorical variables included in the dataset:

* area\_\_rucc with 9 values:
  + 'Nonmetro - Urban population of 2,500 to 19,999, adjacent to a metro area' counties are most frequent with 608 counties.
  + 'Nonmetro - Urban population of 20,000 or more, not adjacent to a metro area' counties are most infrequent with 100 counties.
* area\_\_urban\_influence with 12 values:
  + 'Small-in a metro area with fewer than 1 million residents' counties are most frequent with 692 counties.
  + 'Noncore not adjacent to a metro/micro area and contains a town of 2,500 or more residents' counties are most infrequent with 122.
* econ\_\_economic\_typology with 6 values:
  + 'Non specialized' economic typology counties are most frequent with 1266 counties.
  + 'Mining-dependent' economic typology counties are most infrequent with 254 counties.

In [130]:

bh\_ar = sns.countplot(y='area\_\_rucc', data=poverty, color='#1f77b4').set\_title("Barplot area\_rucc")

![](data:image/png;base64;base64,)

In [129]:

bh\_aui = sns.countplot(y='area\_\_urban\_influence', data=poverty, color='#1f77b4').set\_title("Barplot area\_\_urban\_influence")

![](data:image/png;base64;base64,)

In [153]:

bh\_eet = sns.countplot(y='econ\_\_economic\_typology', data=poverty, color='#1f77b4').set\_title("Barplot econ\_\_economic\_typology")

![](data:image/png;base64;base64,)

### Data Exploration and Visualization of Categorical Variables[¶](#Data-Exploration-and-Visualization-of-Categorical-Variables)

Here the predictive value of the categorical variables 'econ\_\_economic\_typology', 'area\_\_urban\_influence', 'area\_\_rucc' and 'yr' is explored. Box plots are used to explore these categorical variables.

The boxplots of the categorical variables "econ**economic\_typology", "area**urban\_influence" and "area\_\_rucc" show interesting variation:

* 'Farm-dependent' counties have the lowest poverty rates and 'Federal/State government-dependent' counties have the highest poverty rates.
* 'Large-in a metro area with at least 1 million residents or more' counties have the lowest poverty rates.
* 'Metro - Counties in metro areas with 1 million population or more' counties have the lowest poverty rates.

Furthermore by combining features more interesting categorical variables can be created explaining much more of the variance in poverty rates.

* "demo**pct\_aged\_65\_years\_and\_older" and "area**urban\_influence". The general trend is that counties with a low percentage population of "65 years or older" have a higher poverty rate.

The difference in poverty over year 'a' and 'b' is really minimal. Furthermore it doesn't make sense to use this feature to predict poverty rates. This feature will be dropped at the cleaning stage.

In [144]:

bpd\_ar = sns.boxplot(orient="h", x='poverty\_rate', y='area\_\_rucc', data=poverty, color='#1f77b4', saturation=1.0).set\_title('Boxplot poverty\_rate by area\_rucc')

![](data:image/png;base64;base64,)

In [143]:

bpd\_eet = sns.boxplot(orient="h", x='poverty\_rate', y='econ\_\_economic\_typology', data=poverty, color='#1f77b4', saturation=1.0).set\_title('Boxplot poverty\_rate by econ\_\_economic\_typology')

![](data:image/png;base64;base64,)

In [142]:

plt.figure(figsize=(6,6))  
bpd\_aui = sns.boxplot(orient="h", x='poverty\_rate', y='area\_\_urban\_influence', data=poverty, color='#1f77b4', saturation=1.0).set\_title('Boxplot poverty\_rate by area\_\_urban\_influence')

![](data:image/png;base64;base64,)

In [17]:

def create\_old\_age\_cat(input\_df):  
 low\_pct\_olds = poverty.demo\_\_pct\_aged\_65\_years\_and\_older < 0.167000  
 high\_pct\_olds = poverty.demo\_\_pct\_aged\_65\_years\_and\_older >= 0.167000  
 input\_df.loc[low\_pct\_olds,'pct\_65years\_cat'] = 'low\_pct\_65years'  
 input\_df.loc[high\_pct\_olds,'pct\_65years\_cat'] = 'high\_pct\_65years'  
   
 age\_old\_cats = ['low\_pct\_65years','high\_pct\_65years']  
 input\_df.loc[:,'pct\_65years\_cat'] = input\_df.pct\_65years\_cat.astype('category')  
 input\_df.loc[:,'pct\_65years\_cat'] = input\_df.pct\_65years\_cat.cat.set\_categories(age\_old\_cats, ordered=True)  
 return input\_df

In [18]:

poverty = create\_old\_age\_cat(poverty)

In [19]:

def create\_aui\_pct65y\_cat(input\_df):  
 aui\_cats = input\_df.area\_\_urban\_influence.unique()  
 pct65y\_cats = input\_df.pct\_65years\_cat.cat.categories  
   
 aui\_pct65y\_masks = [ ((input\_df.area\_\_urban\_influence == aui) & (input\_df.pct\_65years\_cat == pct65y)  
 , aui + ', ' + pct65y)   
 for (aui, pct65y) in list(itertools.product(aui\_cats, pct65y\_cats))]  
  
 aui\_pct65y\_lbls = [aui + ', ' + pct65y for (aui, pct65y)   
 in list(itertools.product(aui\_cats, pct65y\_cats))]  
   
 for mask, aui\_pct65y\_lb in aui\_pct65y\_masks:  
 input\_df.loc[mask, 'aui\_pct65y\_cat'] = aui\_pct65y\_lb  
   
 input\_df.loc[:,'aui\_pct65y\_cat'] = input\_df.aui\_pct65y\_cat.astype('category')  
 input\_df.loc[:,'aui\_pct65y\_cat'] = input\_df.aui\_pct65y\_cat.cat.set\_categories(aui\_pct65y\_lbls)  
 return input\_df

In [79]:

poverty = create\_aui\_pct65y\_cat(poverty)

### Data Exploration and Visualization of Quantitative Variables[¶](#Data-Exploration-and-Visualization-of-Quantitative-Variables)

For the quantitative variables the correlation matrix is computed first followed by the visual display of the scatter plot matrices.

#### Correlation Matrix[¶](#Correlation-Matrix)

The strongest correlations observed are moderate positive and negative for the following features:

* demo\_\_pct\_adults\_less\_than\_a\_high\_school\_diploma
* health\_\_homicides\_per\_100k
* econ\_\_pct\_unemployment
* health\_\_pct\_low\_birthweight
* econ\_\_pct\_uninsured\_adults
* health\_\_pct\_diabetes
* demo\_\_pct\_non\_hispanic\_african\_american
* econ\_\_pct\_civilian\_labor
* demo\_\_pct\_non\_hispanic\_white
* demo\_\_pct\_adults\_bachelors\_or\_higher

The whole correlation matrix of interest is shown here under.

|  |  |
| --- | --- |
| Features | Pearson Correlation Coefficient |
| econ\_\_pct\_civilian\_labor | -0.670417 |
| demo\_\_pct\_non\_hispanic\_white | -0.499974 |
| demo\_\_pct\_adults\_bachelors\_or\_higher | -0.467134 |
| demo\_\_pct\_adults\_with\_some\_college | -0.363875 |
| health\_\_pct\_excessive\_drinking | -0.353254 |
| demo\_\_pct\_asian | -0.163033 |
| demo\_\_pct\_aged\_65\_years\_and\_older | -0.088123 |
| demo\_\_pct\_female | -0.068065 |
| demo\_\_pct\_below\_18\_years\_of\_age | 0.039237 |
| health\_\_air\_pollution\_particulate\_matter | 0.058582 |
| econ\_\_pct\_uninsured\_children | 0.098882 |
| demo\_\_pct\_hispanic | 0.105574 |
| demo\_\_birth\_rate\_per\_1k | 0.127506 |
| health\_\_pop\_per\_primary\_care\_physician | 0.156942 |
| demo\_\_pct\_adults\_with\_high\_school\_diploma | 0.202928 |
| demo\_\_pct\_american\_indian\_or\_alaskan\_native | 0.236508 |
| demo\_\_death\_rate\_per\_1k | 0.244093 |
| health\_\_pop\_per\_dentist | 0.268996 |
| health\_\_pct\_adult\_smoking | 0.395457 |
| health\_\_motor\_vehicle\_crash\_deaths\_per\_100k | 0.420348 |
| health\_\_pct\_physical\_inacticity | 0.437680 |
| health\_\_pct\_adult\_obesity | 0.444293 |
| demo\_\_pct\_non\_hispanic\_african\_american | 0.507048 |
| health\_\_pct\_diabetes | 0.537038 |
| econ\_\_pct\_uninsured\_adults | 0.541712 |
| health\_\_pct\_low\_birthweight | 0.565456 |
| econ\_\_pct\_unemployment | 0.592022 |
| health\_\_homicides\_per\_100k | 0.621399 |
| demo\_\_pct\_adults\_less\_than\_a\_high\_school\_diploma | 0.680360 |

#### Scatter Plot Matrices[¶](#Scatter-Plot-Matrices)

After reading the '[Rural Poverty & Well-being](https://www.ers.usda.gov/topics/rural-economy-population/rural-poverty-well-being/poverty-overview.aspx)' report it is clear that education, ethnicity and health related issues play an important role in predicting poverty. In this dataset are also added economic indicators of United States' counties. The scatter plot matrices of these four groups of socioeconomic indicators are shown here under. The scatter plot matrices visually confirms the finding of the correlation matrix.

NB: linear statistical transformations (sqrt, square, exponential, etc) were also applied to the target variable 'poverty\_rate' but they did not improve substantially the correlation coefficients and the scatter plot matrices.

The correlation matrices and scatter plot matrices visually confirm that the variables correlate moderately strong with the target variable 'poverty\_rate' seem to have a linear relationship.

--> limit to most important or special scatterplot matrices

##### Moderately Strong Correlating Features Scatter Plot Matrices[¶](#Moderately-Strong-Correlating-Features-Scatter-Plot-Matrices)

In [171]:

sns.set(style="ticks")  
plt.rcParams["axes.labelsize"] = 15  
scatter\_top = sns.pairplot(poverty.loc[:, ['demo\_\_pct\_adults\_less\_than\_a\_high\_school\_diploma'  
 ,'health\_\_homicides\_per\_100k','econ\_\_pct\_unemployment'  
 ,'health\_\_pct\_low\_birthweight','econ\_\_pct\_uninsured\_adults'  
 ,'poverty\_rate']].dropna(), size=5  
 , plot\_kws={'color': '#1f77b4'}, diag\_kws={'color': '#1f77b4'})

![](data:image/png;base64;base64,)

In [174]:

plt.rcParams["axes.labelsize"] = 15  
scatter\_top = sns.pairplot(poverty.loc[:, [ 'health\_\_pct\_diabetes', 'demo\_\_pct\_non\_hispanic\_african\_american'  
 ,'econ\_\_pct\_civilian\_labor','demo\_\_pct\_non\_hispanic\_white'  
 ,'demo\_\_pct\_adults\_bachelors\_or\_higher'  
 ,'poverty\_rate']].dropna(), size=5  
 , plot\_kws={'color': '#1f77b4'}, diag\_kws={'color': '#1f77b4'})

![](data:image/png;base64;base64,)

##### Categorical Features Scatter Plot Matrices[¶](#Categorical-Features-Scatter-Plot-Matrices)

These quantitative features behave like categorical features.

In [173]:

scatter\_non\_ln = sns.pairplot(poverty.loc[:, ['demo\_\_death\_rate\_per\_1k'  
 ,'demo\_\_birth\_rate\_per\_1k'  
 ,'health\_\_air\_pollution\_particulate\_matter'  
 ,'poverty\_rate']].dropna(), size=5  
 , plot\_kws={'color': '#1f77b4'}, diag\_kws={'color': '#1f77b4'})

![](data:image/png;base64;base64,)

#### Creation and Visualization of New Categorical Variables[¶](#Creation-and-Visualization-of-New-Categorical-Variables)

From observing the scatter plot matrices of the features 'demo\_\_death\_rate\_per\_1k', 'demo\_\_birth\_rate\_per\_1k' and 'health\_\_air\_pollution\_particulate\_matter' is clear that these quantitative variables behave like categorical variables. They will be transformed into categorical features by binning them.

In [27]:

def create\_birthrate\_cat(input\_df):  
 cats = [ "birthrate {0} - {1}".format(i, i + 5) for i in range(0, 40, 5) ]  
 input\_df.loc[:,'birth\_rate\_cat'] = pd.cut(input\_df.demo\_\_birth\_rate\_per\_1k, range(0, 45, 5), right=False, include\_lowest=True, labels=cats)  
 input\_df.loc[:,'birth\_rate\_cat'] = input\_df.birth\_rate\_cat.astype('category')  
 input\_df.loc[:,'birth\_rate\_cat'] = input\_df.birth\_rate\_cat.cat.set\_categories(cats, ordered=True)  
 return input\_df

In [28]:

def create\_deathrate\_cat(input\_df):  
 cats = [ "deathrate {0} - {1}".format(i, i + 5) for i in range(0, 40, 5) ]  
 input\_df.loc[:,'death\_rate\_cat'] = pd.cut(input\_df.demo\_\_death\_rate\_per\_1k, range(0, 45, 5), right=False, include\_lowest=True, labels=cats)  
 input\_df.loc[:,'death\_rate\_cat'] = input\_df.death\_rate\_cat.astype('category')  
 input\_df.loc[:,'death\_rate\_cat'] = input\_df.death\_rate\_cat.cat.set\_categories(cats, ordered=True)  
 return input\_df

In [29]:

def create\_air\_poll\_cat(input\_df):  
 cats = [ "airpoll {0} - {1}".format(i, i + 5) for i in range(0, 35, 5) ]  
 input\_df.loc[:,'air\_poll\_cat'] = pd.cut(input\_df.health\_\_air\_pollution\_particulate\_matter, range(0, 40, 5), right=False, include\_lowest=True, labels=cats)  
 input\_df.loc[:,'air\_poll\_cat'] = input\_df.air\_poll\_cat.astype('category')  
 input\_df.loc[:,'air\_poll\_cat'] = input\_df.air\_poll\_cat.cat.set\_categories(cats, ordered=True)  
 return input\_df

In [30]:

def create\_features(input\_df):  
 input\_df = create\_birthrate\_cat(input\_df)  
 input\_df = create\_deathrate\_cat(input\_df)  
 input\_df = create\_air\_poll\_cat(input\_df)  
 return input\_df

In [31]:

poverty = create\_features(poverty)

## Data Preparation[¶](#Data-Preparation)

This phase involves mostly the cleaning, scaling and one hot encoding of features:

* dropping redundant features
* converting features to the right type
* missing values are replaced by the respective median value of the feature. The median is preferred over the mean because it is less sensible to skewed data and gives a better measure of centrality.
* features are scaled to have the same scale. The MinMaxScaler is applied to the features "health**homicides\_per\_100k' and 'health**motor\_vehicle\_crash\_deaths\_per\_100k' to scale them the same way as other quantitative variables that are in percentages between 0 and 1.
* One hot encoding of the categorical variables is performed

In [32]:

def drop\_features(input\_df):  
 result\_df = input\_df.drop(columns=['health\_\_air\_pollution\_particulate\_matter'  
 ,'demo\_\_death\_rate\_per\_1k', 'demo\_\_birth\_rate\_per\_1k'  
 ,'pct\_65years\_cat','aui\_pct65y\_cat','yr'], axis=1)  
 return result\_df

In [33]:

poverty\_clean = drop\_features(poverty)

In [34]:

def convert\_to\_cat(input\_df):  
 input\_df.loc[:,'area\_\_rucc'] = input\_df.area\_\_rucc.astype("category")  
 input\_df.loc[:,'area\_\_urban\_influence'] = input\_df.area\_\_urban\_influence.astype("category")  
 input\_df.loc[:,'econ\_\_economic\_typology'] = input\_df.econ\_\_economic\_typology.astype("category")  
 return input\_df

In [35]:

poverty\_clean = convert\_to\_cat(poverty\_clean)

In [36]:

dtypes\_tmp = poverty\_clean.dtypes

In [37]:

def clean\_nans(input\_df):  
 result\_df = input\_df.fillna(poverty\_clean.median())  
 return result\_df

In [38]:

poverty\_clean = clean\_nans(poverty\_clean)

In [39]:

clean\_tmp = poverty\_clean.isnull().sum()

In [40]:

def scale\_features(input\_df):  
 input\_scale = input\_df.loc[:,['health\_\_homicides\_per\_100k'  
 ,'health\_\_motor\_vehicle\_crash\_deaths\_per\_100k'  
 ,'health\_\_pop\_per\_dentist'  
 ,'health\_\_pop\_per\_primary\_care\_physician']]  
   
 input\_scaled = pd.DataFrame(MinMaxScaler().fit\_transform(input\_scale), columns=input\_scale.columns)  
   
 input\_df.loc[:,'health\_\_homicides\_per\_100k'] = input\_scaled.loc[:,'health\_\_homicides\_per\_100k']  
 input\_df.loc[:,'health\_\_motor\_vehicle\_crash\_deaths\_per\_100k'] = input\_scaled.loc[:,'health\_\_motor\_vehicle\_crash\_deaths\_per\_100k']  
 input\_df.loc[:,'health\_\_pop\_per\_dentist'] = input\_scaled.loc[:,'health\_\_pop\_per\_dentist']  
 input\_df.loc[:,'health\_\_pop\_per\_primary\_care\_physician'] = input\_scaled.loc[:,'health\_\_pop\_per\_primary\_care\_physician']  
 return input\_df

In [41]:

poverty\_clean = scale\_features(poverty\_clean)

In [42]:

def cat\_to\_dummies(input\_df):  
 result\_df = pd.get\_dummies(input\_df, dummy\_na=True, columns=['area\_\_rucc','econ\_\_economic\_typology'  
 ,'birth\_rate\_cat','death\_rate\_cat','air\_poll\_cat'  
 ,'area\_\_urban\_influence'])  
 return result\_df

In [43]:

poverty\_clean = cat\_to\_dummies(poverty\_clean)

In [44]:

shape\_tmp = poverty\_clean.shape

## Modeling and Evaluation[¶](#Modeling-and-Evaluation)

In this phase two models are compared with each other using the RMSE evaluation metric:

* Least Square Linear Model after applying recursive feature selection to create a linear model with the most important features
* An AdaBoostRegressor which is an ensemble learning model of decision trees.

The best RMSE scores obtained by these two models are:

|  |  |
| --- | --- |
| Model | RMSE |
| AdaBoostRegressor | 2.7847 |
| Linear Regression | 2.9297 |

The AdaBoostRegressor happens to be more precise than the Least Squares Linear Model because it can handle non linear relationships. The AdaBoostRegressor is chosen as the regression model to predict poverty rates for United States Counties.

In [45]:

rng = np.random.RandomState(0)

In [46]:

poverty\_X = poverty\_clean.drop(columns=['row\_id','poverty\_rate'], axis=1)  
poverty\_y = poverty\_clean.poverty\_rate

In [47]:

# use r2 adjusted in the future  
scoring = {'r2':'r2','mse': make\_scorer(mean\_squared\_error, greater\_is\_better=False)}

In [48]:

inner\_cv = ShuffleSplit(n\_splits=5, test\_size=0.3, random\_state=rng)  
outer\_cv = ShuffleSplit(n\_splits=5, test\_size=0.3, random\_state=rng)

### Recursive Feature Selection Linear Regression[¶](#Recursive-Feature-Selection-Linear-Regression)

In [49]:

caching = mkdtemp()  
  
mse = make\_scorer(mean\_squared\_error, greater\_is\_better=False)  
  
rfecv = RFECV(estimator=LinearRegression(), step=1, cv=inner\_cv, scoring=mse)  
  
rfecv.fit(poverty\_X, poverty\_y)  
  
print("Optimal number of features : %d" % rfecv.n\_features\_)

Optimal number of features : 73

In [50]:

pprint('RMSE score: %f' % np.sqrt(np.abs(rfecv.grid\_scores\_[rfecv.n\_features\_])))

'RMSE score: 3.121806'

In [158]:

plt.figure()  
plt.title('Recursive Function Elimination Linear Regression')  
plt.xlabel("Number of features selected")  
plt.ylabel("Cross validation score (rmse)")  
plt.plot(range(1, len(rfecv.grid\_scores\_) + 1), np.sqrt(np.abs(rfecv.grid\_scores\_)))  
plt.show()

![](data:image/png;base64;base64,)

### Nested Cross Validation AdaBoostRegressor Vs Linear Regression[¶](#Nested-Cross-Validation-AdaBoostRegressor-Vs-Linear-Regression)

Using nested cross validation the AdaBoostRegressor is compared with the linear regression model. The AdaBoostRegressor wins the lowest RMSE score.

In [52]:

cachedir = mkdtemp()  
estimators = [('reg\_model', LinearRegression())]  
regr\_pipe = Pipeline(estimators, memory=cachedir)

In [53]:

adaReg = AdaBoostRegressor(base\_estimator=DecisionTreeRegressor(max\_depth=13, splitter='random', criterion='mse'  
 , random\_state=rng)  
 , n\_estimators=600, loss='linear', learning\_rate=1, random\_state=rng)

In [54]:

param\_grid = dict(reg\_model=[rfecv, adaReg])

In [55]:

reg\_grid = GridSearchCV(estimator=regr\_pipe, param\_grid=param\_grid, scoring=scoring, cv=inner\_cv  
 , error\_score=0, refit='mse')  
reg\_pred = reg\_grid.fit(poverty\_X, poverty\_y)

Out[55]:

Pipeline(memory='/var/folders/\_j/vyb4dyfx2wq850vj9vh25wy40000gn/T/tmpe8qvy037',  
 steps=[('reg\_model', AdaBoostRegressor(base\_estimator=DecisionTreeRegressor(criterion='mse', max\_depth=13, max\_features=None,  
 max\_leaf\_nodes=None, min\_impurity\_decrease=0.0,  
 min\_impurity\_split=None, min\_samples\_leaf=1,  
 min\_samples\_split=2, min\_weight\_fraction\_leaf=0....oss='linear', n\_estimators=600,  
 random\_state=<mtrand.RandomState object at 0x1a2f38ecf0>))])

In [56]:

train\_scores = cross\_validate(reg\_grid, poverty\_X, poverty\_y, cv=outer\_cv, scoring=scoring, return\_train\_score=True)  
pprint('RMSE score: %f' % np.average(np.sqrt(np.abs(train\_scores['test\_mse']))))

'RMSE score: 2.428183'

### Recursive Feature Selection AdaBoostRegressor[¶](#Recursive-Feature-Selection-AdaBoostRegressor)

To improve the AdaBoostRegressor even more, the best features are selected using recursive feature elimination or backwards elimination.

In [57]:

mse = make\_scorer(mean\_squared\_error, greater\_is\_better=False)  
  
rfecv = RFECV(estimator=adaReg, step=1, cv=inner\_cv, scoring=mse)  
  
rfecv.fit(poverty\_X, poverty\_y)  
  
print("Optimal number of features : %d" % rfecv.n\_features\_)

Optimal number of features : 62

In [58]:

pprint('RMSE score: %f' % np.sqrt(np.abs(rfecv.grid\_scores\_[rfecv.n\_features\_])))

'RMSE score: 2.406639'

In [59]:

train\_scores = cross\_validate(rfecv, poverty\_X, poverty\_y, cv=outer\_cv, scoring=scoring, return\_train\_score=True)  
pprint('RMSE score: %f' % np.average(np.sqrt(np.abs(train\_scores['test\_mse']))))

'RMSE score: 2.366428'

In [60]:

plt.figure()  
plt.title('Recursive Function Elimination AdaBoostRegressor')  
plt.xlabel("Number of features selected")  
plt.ylabel("Cross validation score (rmse)")  
plt.plot(range(1, len(rfecv.grid\_scores\_) + 1), np.sqrt(np.abs(rfecv.grid\_scores\_)))  
plt.show()

![](data:image/png;base64;base64,)

### Analysis of Predictions and Residuals[¶](#Analysis-of-Predictions-and-Residuals)

The Analysis of the quality of the predictions and residuals shows that the accuracy of the AdaBoostRegressor is high. However the AdaBoostRegressor probably slightly overfits the data.

In [159]:

sc1\_tmp = sns.regplot(x=poverty\_y, y=rfecv.predict(poverty\_X), fit\_reg=False, color='#1f77b4', scatter\_kws={'alpha':0.3})  
tmp = sc1\_tmp.set\_ylabel('predictions')  
tmp = sc1\_tmp.set\_title('Scatter Plot Actual poverty\_rate Vs Predictions')

![](data:image/png;base64;base64,)

In [160]:

sc2\_tmp = sns.regplot(x=rfecv.predict(poverty\_X), y=rfecv.predict(poverty\_X) - poverty\_y  
 , fit\_reg=False, color='#1f77b4', scatter\_kws={'alpha':0.3})  
tmp = sc2\_tmp.set\_xlabel('predictions')  
tmp = sc2\_tmp.set\_ylabel('residuals')  
tmp = sc2\_tmp.set\_title('Scatter Plot Predictions Vs Residuals')

![](data:image/png;base64;base64,)

In [161]:

residuals = rfecv.predict(poverty\_X) - poverty\_y  
residuals = residuals.rename('residuals')  
ht\_pov = sns.distplot(residuals, color='#1f77b4', kde=False).set\_title('Histogram Residuals')

![](data:image/png;base64;base64,)

In [67]:

poverty\_test = pd.read\_csv('./Microsoft\_-\_DAT102x\_Predicting\_Poverty\_in\_the\_United\_States\_-\_Test\_values.csv')

In [68]:

#Create Features  
poverty\_test = create\_old\_age\_cat(poverty\_test)  
poverty\_test = create\_aui\_pct65y\_cat(poverty\_test)  
poverty\_test = create\_features(poverty\_test)  
  
#Convert to correct type  
poverty\_test = convert\_to\_cat(poverty\_test)  
  
#Drop Features  
poverty\_row\_id = poverty\_test.row\_id  
poverty\_test = drop\_features(poverty\_test)  
poverty\_test = poverty\_test.drop(columns=['row\_id'], axis=1)  
  
#Replace NANs  
poverty\_test\_clean = poverty\_test.fillna(poverty\_test.median())  
poverty\_test\_clean = cat\_to\_dummies(poverty\_test\_clean)  
  
#Scale Features  
poverty\_test\_clean = scale\_features(poverty\_test\_clean)

In [69]:

#Create Prediction  
submission = pd.DataFrame(rfecv.predict(poverty\_test\_clean))  
submission = np.clip(submission,2.50, 48.00)

In [70]:

poverty\_submission = pd.concat([poverty\_row\_id, submission], axis=1)  
poverty\_submission = poverty\_submission.rename(index=str, columns={0: 'poverty\_rate'})  
poverty\_submission = poverty\_submission.round({'poverty\_rate':2})

In [71]:

poverty\_submission.to\_csv(path\_or\_buf='./MV\_Poverty\_Submission\_AdaReg\_tmp.csv', index=False)

## Conclusion[¶](#Conclusion)

The Regression analysis shows that is possible to build an accurate regression model to predict poverty rates of United States' counties using an AdaBoostRegressor. From the data exploration phase it is clear that economical, educational, ethnical and health related factors play an important role in predicting poverty. However is recursive function elemination is used to determine the optimal number of features that leads to the best prediction.