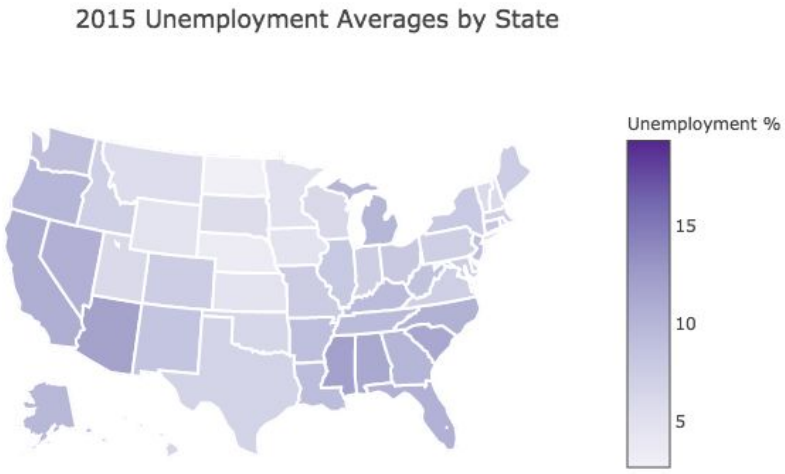
Ryan Schaub
Joseph Kraut

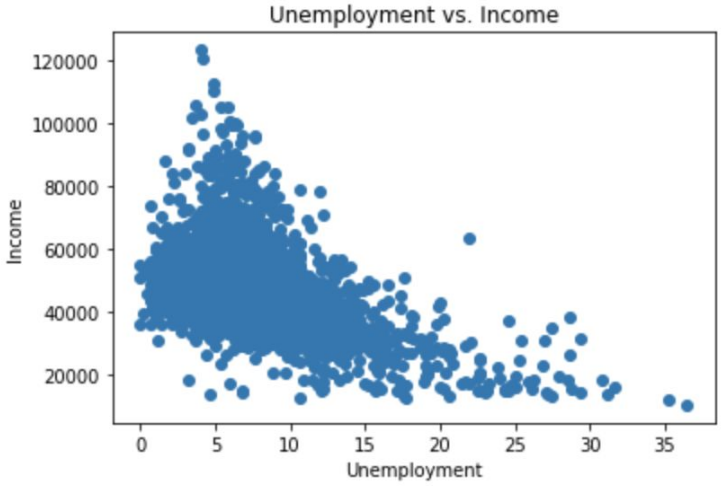
Project 1 Write Up

In trying to regress upon financial indicators the first step we took was to clean and explore our dataset. That meant dropping all n/a values first and then calling the Pandas functions “describe” and “columns” to see what types of data we might be working with and their distributions.

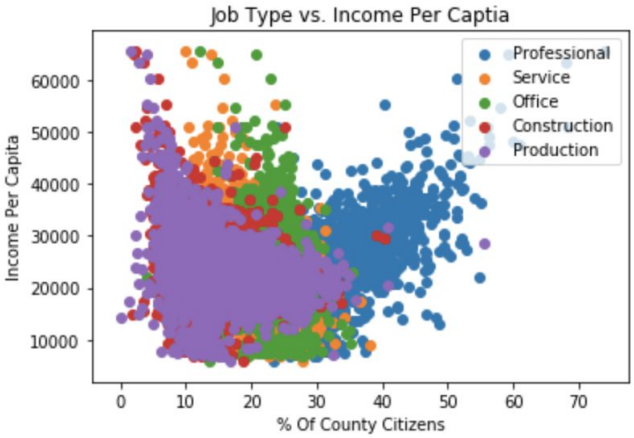
Our next step was to visualize the data and get an understanding of what financial indicators are strongly dependent on other features. We began with a choropleth plot to visualize the unemployment percentage averaged across the states. This choropleth seemed to

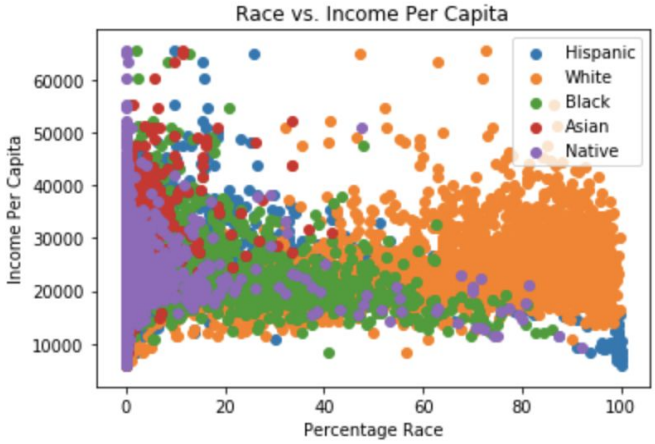


demonstrate that higher unemployment is primarily concentrated in the west coast and southeast of America- with the northeast contributing median levels of unemployment and the midwest seeing very little unemployment. Intuitively, the causal factor *could* be that states with higher unemployment also have a larger population and are forced to disperse limited work across a larger population. This choropleth plot also demonstrated a pattern in the unemployment data that we could likely regress upon; so we decided to explore more about unemployment. We then plotted a scatter diagram of unemployment and income and found

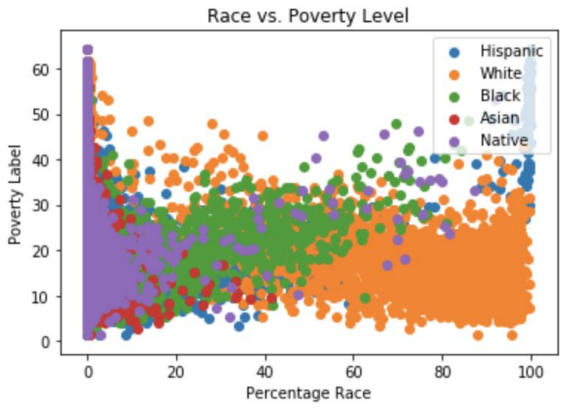


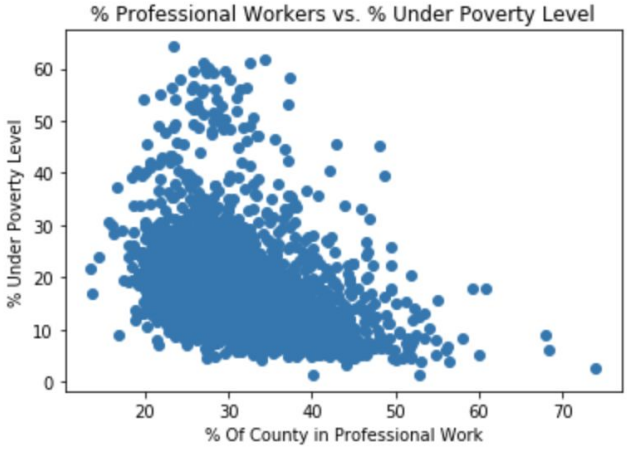
an approximately linearly - possibly logarithmic- trend. This implored us to first of all linearize the Income data when regressing on unemployment, by taking the negative log of the data, and second to explore income per capita as a possible financial predictor to regress upon. In exploring income and income per capita we realized both had somewhat strong linear relationships with certain job types and ethnicities. This helped us decide that regression upon





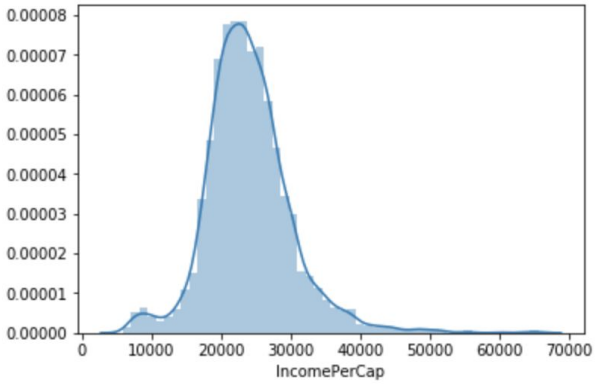
income per capita would likely yield reasonable results. Finally, we decided on poverty as our third financial indicator after plotting poverty vs professional workers and poverty vs race. We

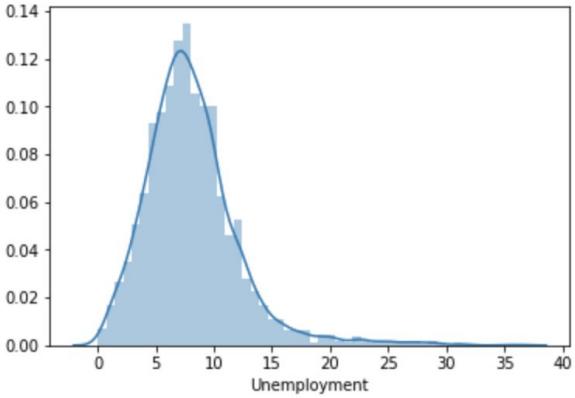


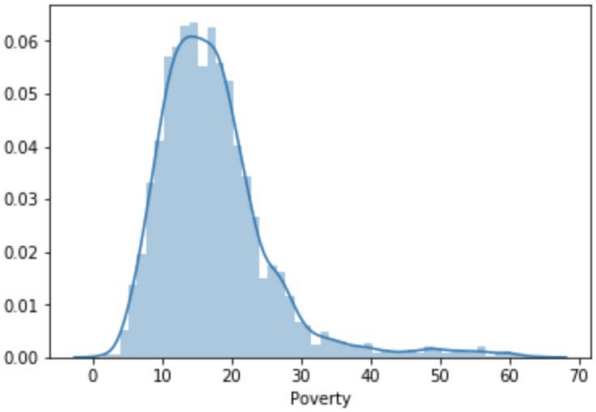


saw once again somewhat strong relationships between races (specifically black, white, and hispanic) and poverty levels, as well as a relationship between the percentage of professional workers in a county and poverty levels. We deemed these relationships as well as the relationships not shown to be sufficient to regress upon, concluding our three financial predictors to be poverty, unemployment, and income per capita.

After visualizing and exploring our data in depth we began regression to predict our three financial indicators. The first step was to visualize the distributions of our financial indicators so we could assess reasonable mean squared error values, so we plotted seaborn histograms of each of our financial indicators.







These histograms helped give us an idea of the distributions our linear regression should ideally be fitting. We then ran linear regression upon each of the three financial indicators and got a validation MSE of 3.835 for poverty, 5.162 for unemployment, and 5711083.42888 for income per capita. We tried polynomial feature regression which made almost no significant effect on our data- overfitting very quickly- and only increased our runtime. As well, we tried LASSO regression in an attempt to bring down the MSE for our income per capita regression in particular. However, none of these techniques seemed to work very well, so we settled for the high MSE on income per capita. Finally we outputted the summary of our models and noted the R^2 values of 0.940 for poverty level, 0.636 for unemployment level, and 0.877 for income per capita. This perhaps explains at least some of the high MSE, however, the linear model for unemployment seemed to fit better than the model for income per capita which contradicts what the R^2 values would suggest. As the summaries for our models showed, we saw high coefficients for job type and somewhat high coefficients for ethnicity in our models, signalling that those variables were stronger predictors than others. As well, transportation and working from home seemed to be two variables whose weights were much higher than we anticipated. Overall, linear regression may not have been the best model for our financial predictors; we achieved acceptable degrees of accuracy, but no linear model truly seemed to capture the underlying probability distributions.

Project 1 Skeleton Code

March 1, 2018

1 Project 1: Data Cleaning, Visualization, and Mining

In [2]:
```
# Add all of your import statements here
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
import plotly as py
import plotly.offline as py
import plotly.graph_objs as go
import plotly.tools as tls
py.init_notebook_mode(connected=True)
```

2 Setup, Cleaning, Organizing, and Exploring the Data

Let's now try to read in the data, clean up the data by getting rid of NAs, and explore the data. Feel free to use any commands to do these parts. As always, report your steps in the writeup.

In [3]:
```
#read in the data
df = pd.read_csv('acs2015_county_data.csv')
#drop the NAs
df = df.dropna()
#show the head
df.head()
```

Out[3]:

	CensusId	State	County	TotalPop	Men	Women	Hispanic	White	Black	\
0	1001	Alabama	Autauga	55221	26745	28476	2.6	75.8	18.5	
1	1003	Alabama	Baldwin	195121	95314	99807	4.5	83.1	9.5	
2	1005	Alabama	Barbour	26932	14497	12435	4.6	46.2	46.7	
3	1007	Alabama	Bibb	22604	12073	10531	2.2	74.5	21.4	
4	1009	Alabama	Blount	57710	28512	29198	8.6	87.9	1.5	

	Native	...	Walk	OtherTransp	WorkAtHome	MeanCommute	Employed	\
0	0.4	...	0.5	1.3	1.8	26.5	23986	
1	0.6	...	1.0	1.4	3.9	26.4	85953	

```
2     0.2      ...        1.8          1.5         1.6          24.1      8597
3     0.4      ...        0.6          1.5         0.7          28.8      8294
4     0.3      ...        0.9          0.4         2.3          34.9     22189

   PrivateWork  PublicWork  SelfEmployed  FamilyWork  Unemployment
0         73.6        20.9           5.5         0.0           7.6
1         81.5        12.3           5.8         0.4           7.5
2         71.8        20.8           7.3         0.1          17.6
3         76.8        16.1           6.7         0.4           8.3
4         82.0        13.5           4.2         0.4           7.7

[5 rows x 37 columns]
```

In [4]:
```
#get an idea of the shape
df.shape
```

Out[4]:
```
(3218, 37)
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['CensusId', 'State', 'County', 'TotalPop', 'Men', 'Women', 'Hispanic',
       'White', 'Black', 'Native', 'Asian', 'Pacific', 'Citizen', 'Income',
       'IncomeErr', 'IncomePerCap', 'IncomePerCapErr', 'Poverty',
       'ChildPoverty', 'Professional', 'Service', 'Office', 'Construction',
       'Production', 'Drive', 'Carpool', 'Transit', 'Walk', 'OtherTransp',
       'WorkAtHome', 'MeanCommute', 'Employed', 'PrivateWork', 'PublicWork',
       'SelfEmployed', 'FamilyWork', 'Unemployment'],
      dtype='object')
```

In [6]:
```
df.describe()
```

Out[6]:
```
           CensusId      TotalPop           Men         Women     Hispanic  \
count   3218.000000  3.218000e+03  3.218000e+03  3.218000e+03  3218.000000
mean   31393.444065  9.947107e+04  4.892729e+04  5.054378e+04    11.006029
std    16291.853976  3.193951e+05  1.567252e+05  1.627076e+05    19.242390
min     1001.000000  2.670000e+02  1.360000e+02  1.310000e+02     0.000000
25%    19033.500000  1.122525e+04  5.657750e+03  5.574250e+03     1.900000
50%    30024.000000  2.607950e+04  1.294700e+04  1.306300e+04     3.900000
75%    46104.500000  6.645750e+04  3.299825e+04  3.352250e+04     9.800000
max    72153.000000  1.003839e+07  4.945351e+06  5.093037e+06    99.900000

              White        Black       Native        Asian      Pacific  \
count  3218.000000  3218.000000  3218.000000  3218.000000  3218.000000
mean     75.451243     8.670883     1.720603     1.223244     0.071815
std      22.922274    14.281924     7.252676     2.610159     0.393455
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%      64.100000     0.500000     0.100000     0.200000     0.000000
50%      84.100000     1.900000     0.300000     0.500000     0.000000
75%      93.200000     9.600000     0.600000     1.200000     0.000000
```

```
max      99.800000    85.900000    92.100000    41.600000    11.100000

              ...          Walk  OtherTransp  WorkAtHome  MeanCommute  \
count         ...   3218.000000  3218.000000  3218.000000  3218.000000
mean          ...      3.310534     1.609105     4.628713    23.282474
std           ...      3.699291     1.654761     3.173193     5.596578
min           ...      0.000000     0.000000     0.000000     4.900000
25%           ...      1.400000     0.900000     2.700000    19.500000
50%           ...      2.400000     1.300000     3.900000    23.000000
75%           ...      4.000000     1.900000     5.600000    26.800000
max           ...     71.200000    39.100000    37.200000    44.000000

             Employed  PrivateWork   PublicWork  SelfEmployed   FamilyWork  \
count    3.218000e+03  3218.000000  3218.000000   3218.000000  3218.000000
mean     4.562182e+04    74.235643    17.543350      7.932846     0.288285
std      1.497417e+05     7.817393     6.458126      3.914249     0.455222
min      1.660000e+02    29.500000     5.800000      0.000000     0.000000
25%      4.562000e+03    70.500000    13.100000      5.400000     0.100000
50%      1.052200e+04    75.700000    16.200000      6.900000     0.200000
75%      2.864425e+04    79.700000    20.500000      9.400000     0.300000
max      4.635465e+06    88.300000    66.200000     36.600000     9.800000

         Unemployment
count     3218.000000
mean         8.094779
std          4.093038
min          0.000000
25%          5.500000
50%          7.600000
75%          9.900000
max         36.500000

[8 rows x 35 columns]
```

3 Visualizing the Data

This is the most fun part of the project! Now you can go ahead and create your graphs (appropriately labeled of course), models, and other visualizations. Make sure you demonstrate your ability to create rich graphs by creating various different types of graphs, both for numerical as well as categorical data.

Here are some types of graphs that you will want to create for this project. Remember to browse the online documentation for many Python graphing and visualization packages such as plotly, seaborn, and matplotlib if you need to borrow any code or get some help! 1. Creating multiple graphs that plot important financial indicators (y-axis) like the median annual income, per capita income, poverty rate, childhood poverty rate, and/or unemployment rate present in a particular county AGAINST several features in the dataset (x-axis) that might affect these factors, such as the total population of the county; the racial demographics of the county; or the

occupations of different workers (professionals, service workers, construction, manufacturing).

2. Using plotly as a graphing library to visualize these financial indicators in the 2015 American Community Survey data on a geographical map of the United States on a state-wide or county-based level (similar to what we did in class when we visualized global average temperatures for different countries on a global map).

```
In [7]: #state codes dictionary to convert to a state code for plotting
        us_state_codes = {
            'Alabama': 'AL',
            'Alaska': 'AK',
            'Arizona': 'AZ',
            'Arkansas': 'AR',
            'California': 'CA',
            'Colorado': 'CO',
            'Connecticut': 'CT',
            'Delaware': 'DE',
            'Florida': 'FL',
            'Georgia': 'GA',
            'Hawaii': 'HI',
            'Idaho': 'ID',
            'Illinois': 'IL',
            'Indiana': 'IN',
            'Iowa': 'IA',
            'Kansas': 'KS',
            'Kentucky': 'KY',
            'Louisiana': 'LA',
            'Maine': 'ME',
            'Maryland': 'MD',
            'Massachusetts': 'MA',
            'Michigan': 'MI',
            'Minnesota': 'MN',
            'Mississippi': 'MS',
            'Missouri': 'MO',
            'Montana': 'MT',
            'Nebraska': 'NE',
            'Nevada': 'NV',
            'New Hampshire': 'NH',
            'New Jersey': 'NJ',
            'New Mexico': 'NM',
            'New York': 'NY',
            'North Carolina': 'NC',
            'North Dakota': 'ND',
            'Ohio': 'OH',
            'Oklahoma': 'OK',
            'Oregon': 'OR',
            'Pennsylvania': 'PA',
            'Rhode Island': 'RI',
```

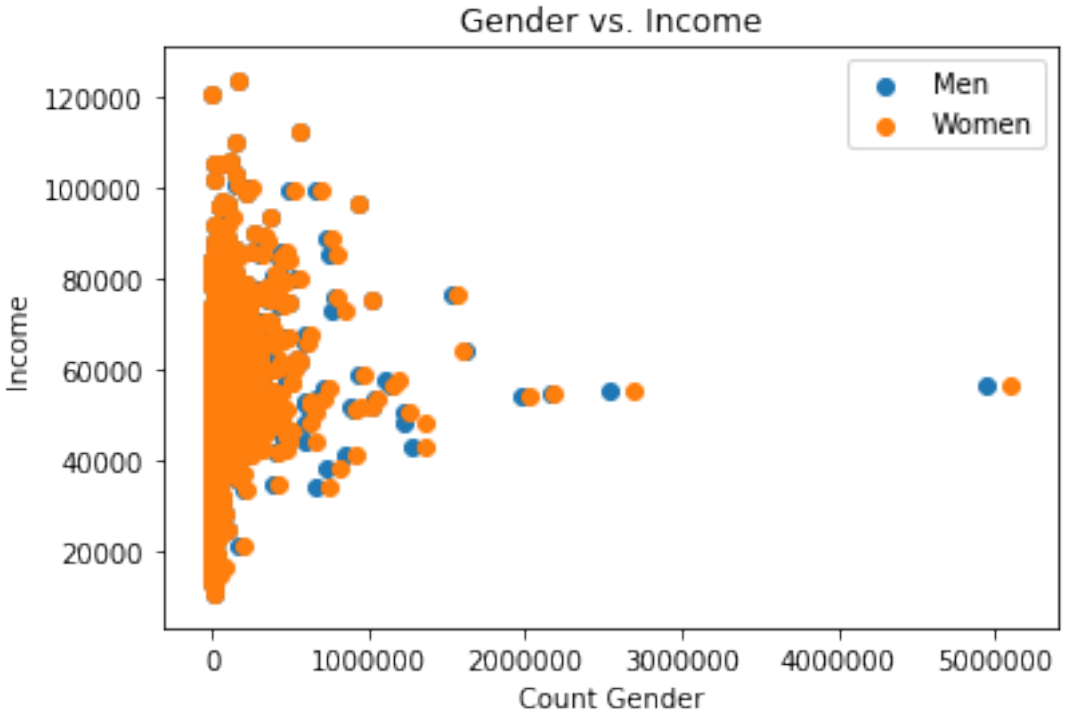
```
        'South Carolina': 'SC',
        'South Dakota': 'SD',
        'Tennessee': 'TN',
        'Texas': 'TX',
        'Utah': 'UT',
        'Vermont': 'VT',
        'Virginia': 'VA',
        'Washington': 'WA',
        'West Virginia': 'WV',
        'Wisconsin': 'WI',
        'Wyoming': 'WY',
        'District of Columbia': 'DC',
        'Puerto Rico': 'PR'
    }
```

In [8]:
```
#choropleth plot of United States
#make a copy of the dataframe
data_copy = df.copy()
#convert all to string data
states = np.unique(df['State'])
state_codes = []
unemployment = []
#append all mean unemployments to unemployment by state
for state in states:
    #append unemployment
    unemployment.append(df[df['State'] == state]['Unemployment'].mean())
    state_codes += [us_state_codes[state]]



#out of plotly documentation
scl = [[0.0, 'rgb(242,240,247)'],[0.2, 'rgb(218,218,235)'],[0.4, 'rgb(188,189,220)'],\
            [0.6, 'rgb(158,154,200)'],[0.8, 'rgb(117,107,177)'],[1.0, 'rgb(84,39,143)']

#begin the plotting
data = [ dict(
        type='choropleth',
        colorscale=scl,
        autocolorscale = False,
        locations = state_codes,
        z = unemployment,
        locationmode = 'USA-states',
        text = states,
        marker = dict(
            line = dict (
                color = 'rgb(255,255,255)',
                width = 2
            ) ),
```

```
            colorbar = dict(
                title = "Unemployment %")
            ) ]

        layout = dict(
            title = '2015 Unemployment Averages by State',
            geo = dict(
                scope='usa',
                projection=dict( type='albers usa' ),
                showlakes = True,
                lakecolor = 'rgb(255, 255, 255)'),
                 )

        fig = dict(data=data, layout=layout)
        py.iplot(fig, validate=False, filename='statesmap')
```

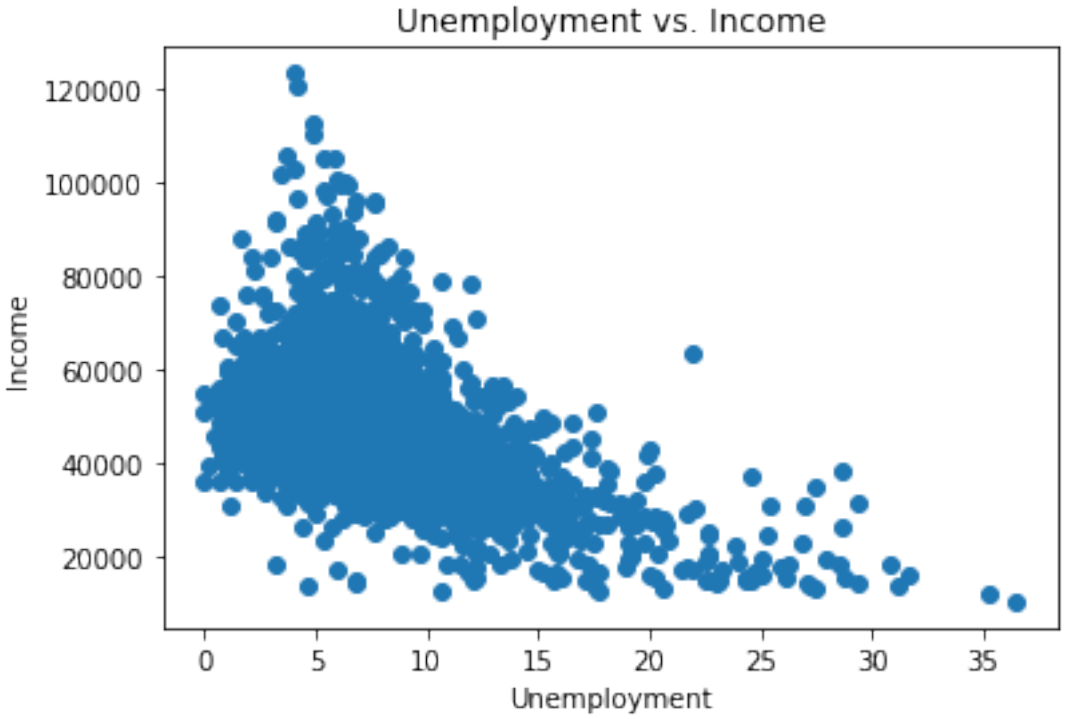
In [9]:
```
#plotting gender vs income
plt.title("Gender vs. Income")
plt.xlabel("Count Gender")
plt.ylabel("Income")
plt.scatter(df['Men'], df['Income'], label="Men")
plt.scatter(df['Women'], df['Income'], label="Women")
plt.legend(loc='upper right')
```

Out[9]: <matplotlib.legend.Legend at 0x102c1ccf8>



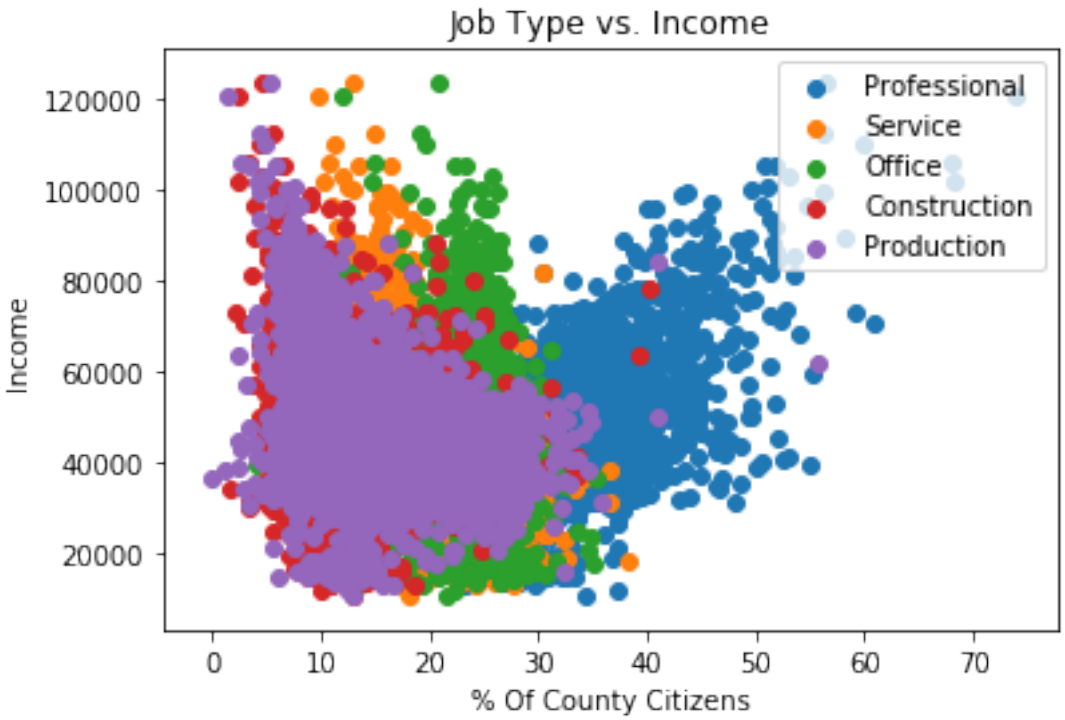
In [10]:
```
#plotting unemployment vs. income
plt.figure()
plt.title("Unemployment vs. Income")
plt.xlabel("Unemployment")
plt.ylabel("Income")
plt.scatter(df['Unemployment'], df['Income'])
```

Out[10]: <matplotlib.collections.PathCollection at 0x1a13caf6d8>

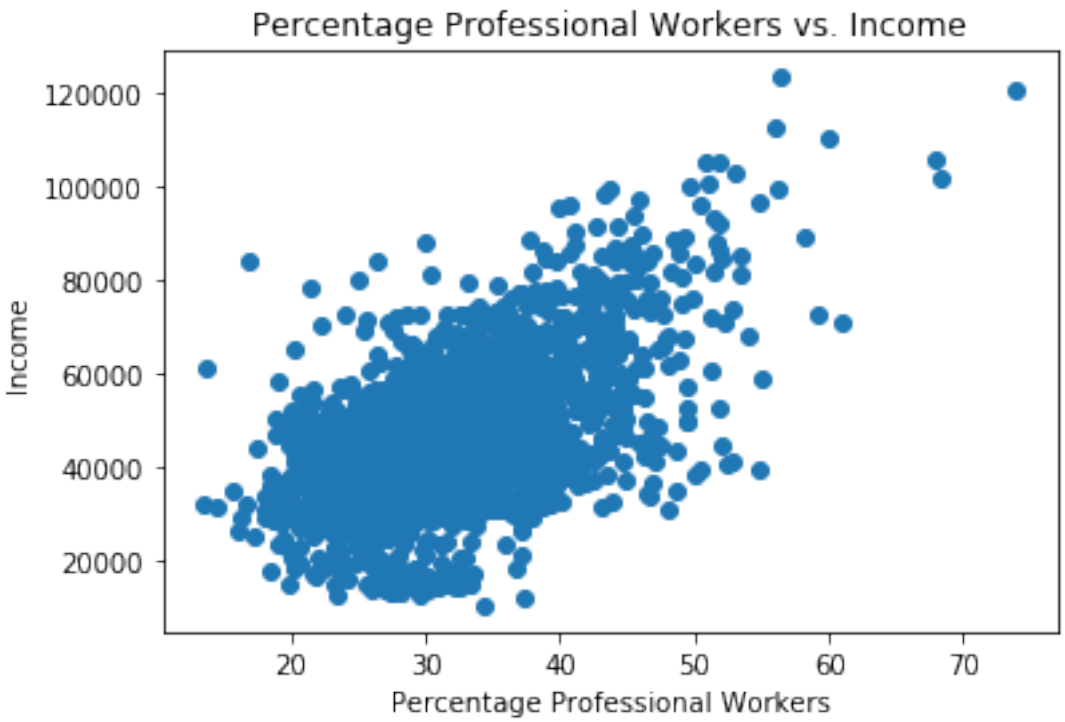


In [11]:
```
plt.figure()
plt.title("Job Type vs. Income")
plt.xlabel("% Of County Citizens")
plt.ylabel("Income")
plt.scatter(df['Professional'], df['Income'], label="Professional")
plt.scatter(df['Service'], df['Income'], label="Service")
plt.scatter(df['Office'], df['Income'], label="Office")
plt.scatter(df['Construction'], df['Income'], label="Construction")
plt.scatter(df['Production'], df['Income'], label="Production")
plt.legend(loc='upper right')
```

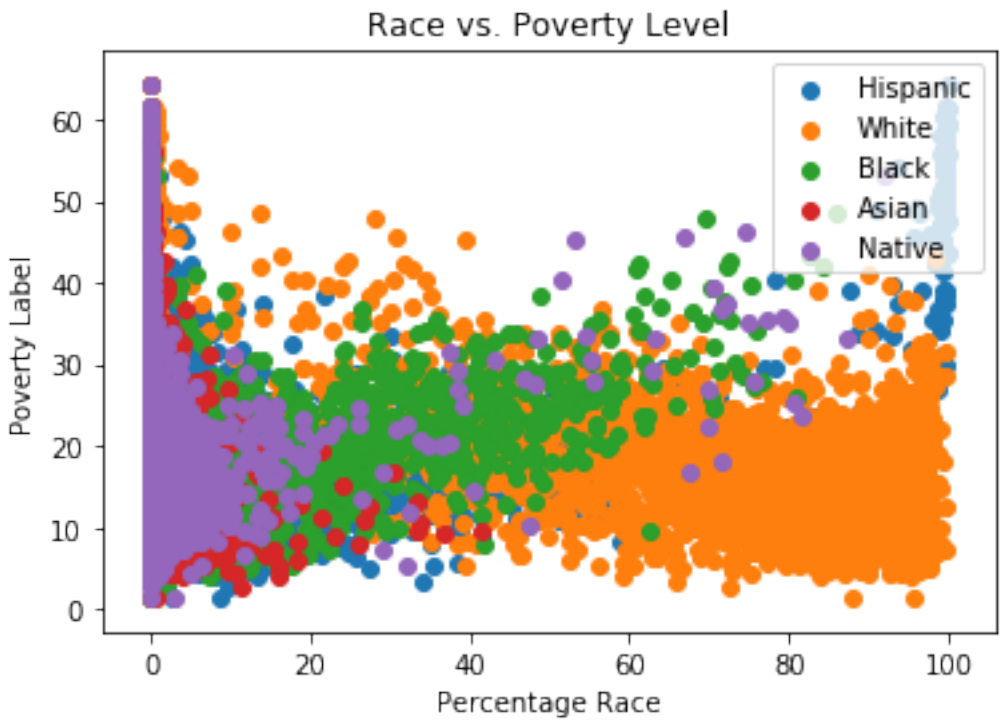
Out[11]: <matplotlib.legend.Legend at 0x1a13daecf8>



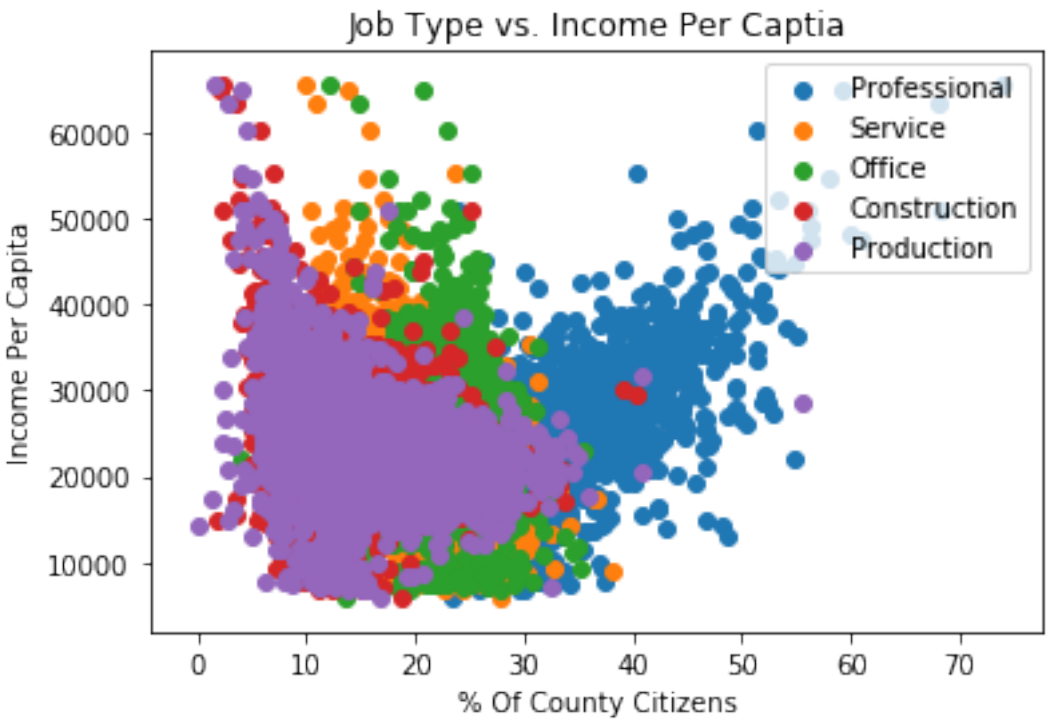

```
In [12]: plt.figure()
         plt.title("Percentage Professional Workers vs. Income")
         plt.xlabel("Percentage Professional Workers")
         plt.ylabel("Income")
         plt.scatter(df['Professional'], df['Income'])

Out[12]: <matplotlib.collections.PathCollection at 0x1a13e864a8>
```



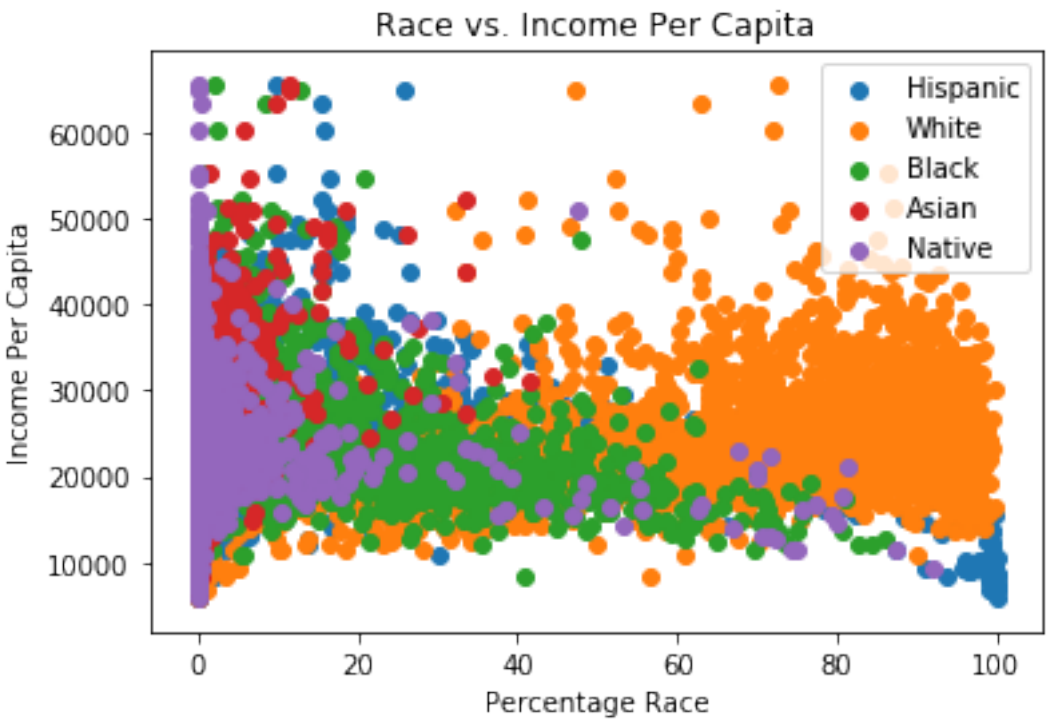

```
In [13]: plt.figure()
         plt.title("Race vs. Poverty Level")
         plt.xlabel("Percentage Race")
         plt.ylabel("Poverty Label")
         plt.scatter(df['Hispanic'], df['Poverty'], label='Hispanic')
         plt.scatter(df['White'], df['Poverty'], label='White')
         plt.scatter(df['Black'], df['Poverty'], label='Black')
         plt.scatter(df['Asian'], df['Poverty'], label='Asian')
         plt.scatter(df['Native'], df['Poverty'], label='Native')
         plt.legend(loc='upper right')

Out[13]: <matplotlib.legend.Legend at 0x1a1407cda0>
```



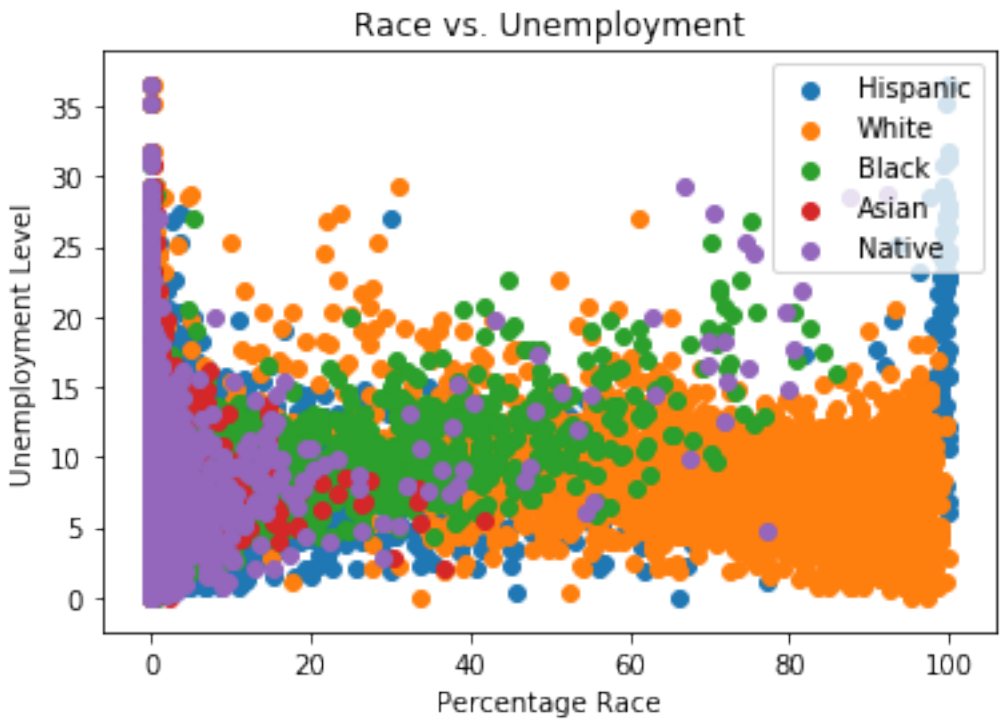

```
In [14]: plt.figure()
         plt.title("Job Type vs. Income Per Captia")
         plt.xlabel("% Of County Citizens")
         plt.ylabel("Income Per Capita")
         plt.scatter(df['Professional'], df['IncomePerCap'], label="Professional")
         plt.scatter(df['Service'], df['IncomePerCap'], label="Service")
         plt.scatter(df['Office'], df['IncomePerCap'], label="Office")
         plt.scatter(df['Construction'], df['IncomePerCap'], label="Construction")
         plt.scatter(df['Production'], df['IncomePerCap'], label="Production")
         plt.legend(loc='upper right')

Out[14]: <matplotlib.legend.Legend at 0x1a140b5a90>
```




```
In [15]: plt.figure()
         plt.title("Race vs. Income Per Capita")
         plt.xlabel("Percentage Race")
         plt.ylabel("Income Per Capita")
         plt.scatter(df['Hispanic'], df['IncomePerCap'], label='Hispanic')
         plt.scatter(df['White'], df['IncomePerCap'], label='White')
         plt.scatter(df['Black'], df['IncomePerCap'], label='Black')
         plt.scatter(df['Asian'], df['IncomePerCap'], label='Asian')
         plt.scatter(df['Native'], df['IncomePerCap'], label='Native')
         plt.legend(loc='upper right')

Out[15]: <matplotlib.legend.Legend at 0x1a141cfef0>
```




```
In [16]: plt.figure()
         plt.title("Race vs. Unemployment")
         plt.xlabel("Percentage Race")
         plt.ylabel("Unemployment Level")
         plt.scatter(df['Hispanic'], df['Unemployment'], label='Hispanic')
         plt.scatter(df['White'], df['Unemployment'], label='White')
         plt.scatter(df['Black'], df['Unemployment'], label='Black')
         plt.scatter(df['Asian'], df['Unemployment'], label='Asian')
         plt.scatter(df['Native'], df['Unemployment'], label='Native')
         plt.legend(loc='upper right')

Out[16]: <matplotlib.legend.Legend at 0x1a142ae710>
```




```
In [17]: plt.figure()
         plt.title("% Professional Workers vs. % Under Poverty Level")
         plt.xlabel("% Of County in Professional Work")
         plt.ylabel("% Under Poverty Level")
         plt.scatter(df['Professional'], df['Poverty'], label="Professional")

Out[17]: <matplotlib.collections.PathCollection at 0x1a14479e80>
```

4 Linear Regression

```
In [18]: from sklearn.feature_selection import RFECV
         from sklearn.linear_model import LinearRegression
         from sklearn import datasets
         from sklearn.preprocessing import PolynomialFeatures
         from sklearn.metrics import mean_squared_error
         import statsmodels.api as sm
         from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LassoCV
```

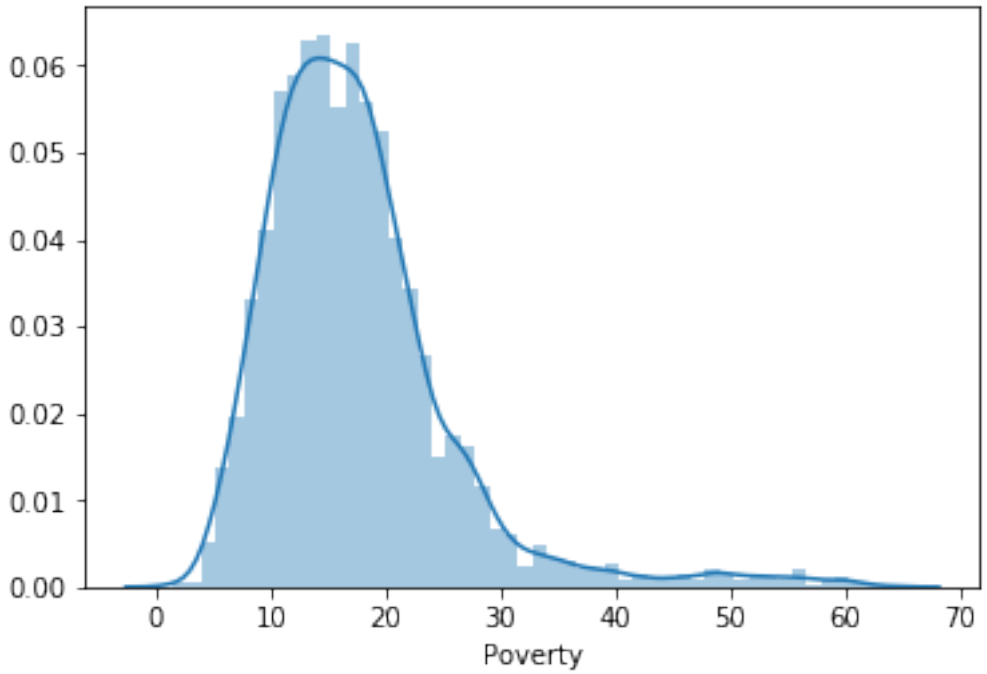
```
/Users/joey/anaconda3/lib/python3.6/site-packages/statsmodels/compat/pandas.py:56: FutureWarni

The pandas.core.datetools module is deprecated and will be removed in a future version. Please
```

Make sure you also demonstrate your ability to create regressions on your data. In this part of the project, you will pick 3 financial indicators (like the median annual income, per capita income, poverty rate, childhood poverty rate, and/or unemployment rate) in the dataset. You will then train 3 linear regression models in scikit-learn that attempt to predict these 3 financial indicators of a county from the rest of the data collected in the dataset. You should analyze the accuracy of your linear regression model and report important predictors or features in your dataset that had the highest effect size on these 3 financial indicators (as discovered by your regression model).

Optional: You can try adding regularization to see if you can get better results.

```
In [75]: sns.distplot(df['Poverty'])
```

```
Out[75]: <matplotlib.axes._subplots.AxesSubplot at 0x1c23afa400>
```




```
In [76]: #regression for poverty level
         df['Income'] = -np.log(df["Income"])
         features = df.drop(['Poverty', 'State', 'CensusId', 'County'], axis=1)
         labels = df['Poverty']
         features = sm.add_constant(features)
         #split into train and test
         train_x, test_x, train_y, test_y = train_test_split(features, labels, test_size=0.2)
         ols = sm.OLS(train_y, train_x).fit()

         #predict the values
         validation_predictions = ols.predict(test_x)
         train_predictions = ols.predict(train_x)

         #compute MSE
         val_mse = mean_squared_error(test_y, validation_predictions)
         train_mse = mean_squared_error(train_y, train_predictions)
         print("Training set MSE: " + str(train_mse) + "\nTesting set MSE: " + str(val_mse))
```

```
Training set MSE: 4.06374270114
Testing set MSE: 3.68106477534
```

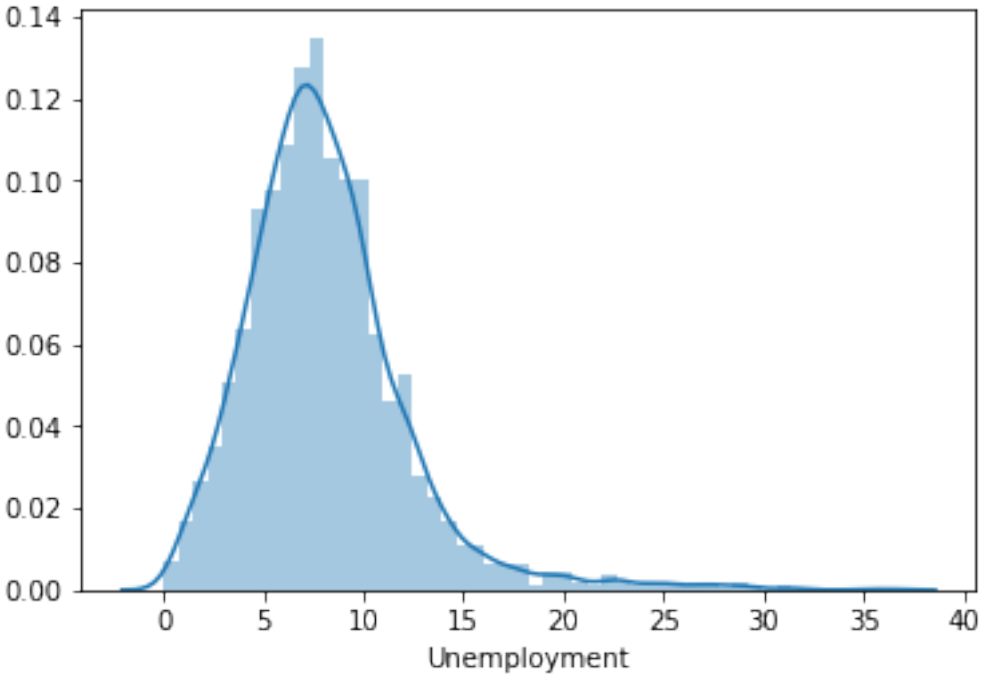
```
In [77]: ols.summary()
```

```
Out[77]: <class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                Poverty   R-squared:                       0.940
Model:                            OLS   Adj. R-squared:                  0.940
Method:                 Least Squares   F-statistic:                     1250.
Date:                Thu, 01 Mar 2018   Prob (F-statistic):               0.00
Time:                        09:26:48   Log-Likelihood:                -5456.9
No. Observations:                2574   AIC:                         1.098e+04
Df Residuals:                    2541   BIC:                         1.117e+04
Df Model:                          32
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        122.9761    110.373      1.114      0.265     -93.454     339.406
TotalPop    2.465e-07   1.78e-06      0.139      0.890   -3.24e-06    3.73e-06
Men          4.342e-05   1.02e-05      4.257      0.000    2.34e-05    6.34e-05
Women       -4.317e-05   1.03e-05     -4.204      0.000   -6.33e-05    -2.3e-05
Hispanic        0.0409      0.028      1.487      0.137      -0.013       0.095
White          -0.0068      0.028     -0.242      0.808      -0.062       0.048
Black          -0.0007      0.028     -0.024      0.981      -0.055       0.054
Native          0.0511      0.030      1.694      0.090      -0.008       0.110
Asian           0.1166      0.039      3.008      0.003       0.041       0.193
Pacific        -0.0477      0.145     -0.330      0.742      -0.332       0.236
Citizen     -4.805e-06   2.75e-06     -1.747      0.081   -1.02e-05    5.87e-07
Income         11.4546      0.442     25.939      0.000      10.589      12.320
IncomeErr   7.168e-05    3.2e-05      2.240      0.025    8.94e-06       0.000
IncomePerCap -8.439e-05  1.85e-05     -4.559      0.000      -0.000   -4.81e-05
IncomePerCapErr -0.0002  6.11e-05     -2.588      0.010      -0.000   -3.83e-05
ChildPoverty    0.3774      0.008     48.312      0.000       0.362       0.393
Professional    0.7944      0.633      1.255      0.210      -0.447       2.036
Service         0.6061      0.633      0.957      0.338      -0.635       1.848
Office          0.6349      0.633      1.003      0.316      -0.607       1.876
Construction    0.6090      0.633      0.962      0.336      -0.632       1.850
Production      0.6057      0.633      0.957      0.339      -0.636       1.847
Drive          -0.1856      0.579     -0.321      0.748      -1.320       0.949
Carpool        -0.1961      0.579     -0.339      0.735      -1.331       0.939
Transit        -0.1209      0.578     -0.209      0.835      -1.255       1.013
Walk           -0.0876      0.579     -0.151      0.880      -1.222       1.047
OtherTransp    -0.1894      0.578     -0.327      0.743      -1.324       0.945
WorkAtHome     -0.1833      0.579     -0.317      0.752      -1.319       0.952
MeanCommute     0.0337      0.009      3.751      0.000       0.016       0.051
Employed     6.007e-06    4.6e-06      1.305      0.192   -3.02e-06     1.5e-05
PrivateWork    -0.3932      0.708     -0.555      0.579      -1.782       0.995
```

```
PublicWork          -0.4017      0.708     -0.567      0.571      -1.790       0.987
SelfEmployed        -0.5469      0.708     -0.773      0.440      -1.934       0.841
FamilyWork          -0.3831      0.710     -0.540      0.590      -1.775       1.009
Unemployment         0.1239      0.016      7.717      0.000       0.092       0.155
==============================================================================
Omnibus:                      577.794   Durbin-Watson:                   2.038
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4413.403
Skew:                           0.850   Prob(JB):                         0.00
Kurtosis:                       9.185   Cond. No.                     8.86e+15
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
[2] The smallest eigenvalue is 8.2e-18. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
"""
```

In [78]:
```
sns.distplot(df['Unemployment'])
```

Out[78]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c274357b8>
```



In [79]:
```
#code for the unemployment regression
features_unemployment = df.drop(['Unemployment', 'State', 'CensusId', 'County'], axis=
labels_unemployment = df['Unemployment']
#add constants
```

```
features_unemployment = sm.add_constant(features_unemployment)

#split into train and test
train_x, test_x, train_y, test_y = train_test_split(features_unemployment, labels_une
#train the model
ols_unemployment = sm.OLS(train_y, train_x).fit()

#predict the values
validation_predictions_unemployment = ols_unemployment.predict(test_x)
train_predictions_unemployment = ols_unemployment.predict(train_x)

#compute MSE
val_mse = mean_squared_error(test_y, validation_predictions_unemployment)
train_mse = mean_squared_error(train_y, train_predictions_unemployment)
print("Training set MSE: " + str(train_mse) + "\nTesting set MSE: " + str(val_mse))
```

```
Training set MSE: 5.87221442132
Testing set MSE: 6.61068727403
```

```
In [80]: ols_unemployment.summary()
```

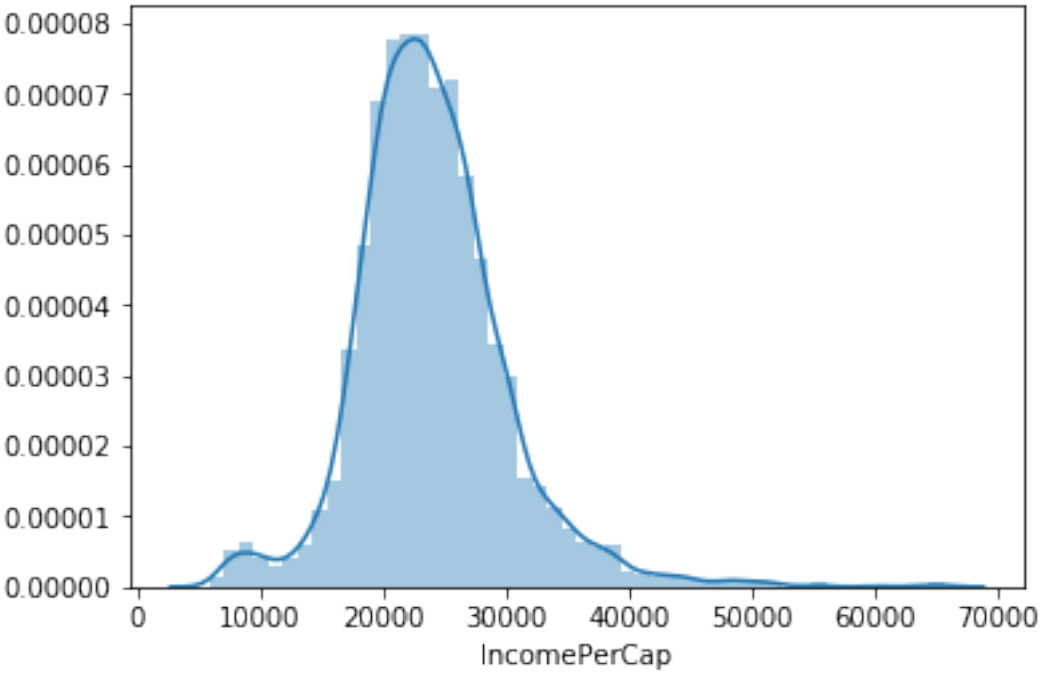
Out[80]:

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:           Unemployment   R-squared:                       0.636
Model:                            OLS   Adj. R-squared:                  0.632
Method:                 Least Squares   F-statistic:                     139.0
Date:                Thu, 01 Mar 2018   Prob (F-statistic):               0.00
Time:                        09:26:49   Log-Likelihood:                -5930.6
No. Observations:                2574   AIC:                         1.193e+04
Df Residuals:                    2541   BIC:                         1.212e+04
Df Model:                          32
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          8.7933    134.368      0.065      0.948    -254.689     272.276
TotalPop     2.761e-06   2.07e-06      1.337      0.181   -1.29e-06    6.81e-06
Men           3.57e-05   1.26e-05      2.836      0.005     1.1e-05    6.04e-05
Women        -3.294e-05   1.26e-05     -2.623      0.009   -5.76e-05   -8.32e-06
Hispanic         0.0049      0.034      0.145      0.884      -0.061       0.071
White           -0.0121      0.034     -0.352      0.725      -0.080       0.055
Black            0.0239      0.034      0.700      0.484      -0.043       0.091
Native           0.0593      0.037      1.590      0.112      -0.014       0.132
Asian            0.0127      0.048      0.266      0.790      -0.081       0.106
Pacific          -0.1781      0.172     -1.034      0.301      -0.516       0.160
```

```
Citizen          1.681e-05   3.35e-06      5.023      0.000    1.02e-05    2.34e-05
Income              4.1059      0.587      6.991      0.000       2.954       5.258
IncomeErr         8.71e-06   3.78e-05      0.230      0.818   -6.54e-05    8.29e-05
IncomePerCap      8.868e-05   2.18e-05      4.076      0.000     4.6e-05       0.000
IncomePerCapErr    -0.0003   7.34e-05     -3.936      0.000      -0.000      -0.000
Poverty             0.1777      0.024      7.454      0.000       0.131       0.224
ChildPoverty       -0.0173      0.013     -1.326      0.185      -0.043       0.008
Professional        0.6117      0.766      0.799      0.425      -0.890       2.114
Service             0.7301      0.766      0.953      0.341      -0.772       2.232
Office              0.7287      0.766      0.951      0.342      -0.773       2.231
Construction        0.6202      0.766      0.810      0.418      -0.881       2.122
Production          0.6871      0.766      0.897      0.370      -0.815       2.189
Drive              -0.3064      0.702     -0.437      0.662      -1.682       1.069
Carpool            -0.2737      0.702     -0.390      0.697      -1.650       1.103
Transit            -0.3167      0.701     -0.452      0.652      -1.692       1.059
Walk               -0.2941      0.702     -0.419      0.675      -1.670       1.082
OtherTransp        -0.1549      0.702     -0.221      0.825      -1.531       1.221
WorkAtHome         -0.2930      0.702     -0.417      0.676      -1.669       1.083
MeanCommute         0.1651      0.010     16.010      0.000       0.145       0.185
Employed        -2.977e-05   6.03e-06     -4.934      0.000   -4.16e-05   -1.79e-05
PrivateWork        -0.0225      0.857     -0.026      0.979      -1.704       1.659
PublicWork          0.0347      0.857      0.041      0.968      -1.646       1.715
SelfEmployed       -0.1157      0.857     -0.135      0.893      -1.795       1.564
FamilyWork          0.0741      0.859      0.086      0.931      -1.611       1.759
==============================================================================
Omnibus:                      300.915   Durbin-Watson:                   2.005
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1941.949
Skew:                           0.347   Prob(JB):                         0.00
Kurtosis:                       7.198   Cond. No.                     1.06e+16
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
[2] The smallest eigenvalue is 5.73e-18. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
"""
```

In [81]:
```
sns.distplot(df['IncomePerCap'])
```

Out[81]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c21b2bd68>
```



In [82]:
```
#regression for IncomePerCap
features = df.drop(['IncomePerCap', 'State', 'CensusId', 'County'], axis=1)
labels = df['IncomePerCap']

features = sm.add_constant(features)

#split into train and test
train_x, test_x, train_y, test_y = train_test_split(features, labels, test_size=0.2)
#train the model
ols = sm.OLS(train_y, train_x).fit()

#predict the values
validation_predictions = ols.predict(test_x)
train_predictions = ols.predict(train_x)

#compute MSE
val_mse = mean_squared_error(test_y, validation_predictions)
train_mse = mean_squared_error(train_y, train_predictions)
print("Training set MSE: " + str(train_mse) + "\nTesting set MSE: " + str(val_mse))
```

```
Training set MSE: 4612296.19563
Testing set MSE: 4810137.50296
```

In [83]:
```
ols.summary()
```

Out[83]: <class 'statsmodels.iolib.summary.Summary'>

"""

OLS Regression Results

Dep. Variable:	IncomePerCap	R-squared:	0.877
Model:	OLS	Adj. R-squared:	0.876
Method:	Least Squares	F-statistic:	567.6
Date:	Thu, 01 Mar 2018	Prob (F-statistic):	0.00
Time:	09:26:49	Log-Likelihood:	-23400.
No. Observations:	2574	AIC:	4.687e+04
Df Residuals:	2541	BIC:	4.706e+04
Df Model:	32		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.625e+04	1.18e+05	0.476	0.634	-1.76e+05	2.88e+05
TotalPop	-0.0215	0.002	-12.097	0.000	-0.025	-0.018
Men	-0.0195	0.011	-1.769	0.077	-0.041	0.002
Women	-0.0020	0.011	-0.177	0.860	-0.024	0.020
Hispanic	6.3272	29.374	0.215	0.829	-51.273	63.927
White	15.4052	29.780	0.517	0.605	-42.990	73.800
Black	4.2570	29.651	0.144	0.886	-53.886	62.400
Native	-28.5579	32.180	-0.887	0.375	-91.660	34.544
Asian	51.7758	41.350	1.252	0.211	-29.307	132.858
Pacific	140.9987	143.457	0.983	0.326	-140.306	422.303
Citizen	0.0175	0.003	5.811	0.000	0.012	0.023
Income	-1.419e+04	449.003	-31.608	0.000	-1.51e+04	-1.33e+04
IncomeErr	-0.1247	0.035	-3.597	0.000	-0.193	-0.057
IncomePerCapErr	1.1494	0.060	19.132	0.000	1.032	1.267
Poverty	-88.1188	21.024	-4.191	0.000	-129.345	-46.893
ChildPoverty	58.7174	11.253	5.218	0.000	36.652	80.783
Professional	922.2149	670.218	1.376	0.169	-392.015	2236.444
Service	757.5177	670.195	1.130	0.258	-556.665	2071.701
Office	745.6728	670.425	1.112	0.266	-568.962	2060.308
Construction	691.5057	670.091	1.032	0.302	-622.474	2005.485
Production	682.2672	670.323	1.018	0.309	-632.168	1996.703
Drive	-1090.0056	615.833	-1.770	0.077	-2297.591	117.580
Carpool	-1161.7230	615.935	-1.886	0.059	-2369.509	46.063
Transit	-915.3854	615.331	-1.488	0.137	-2121.987	291.216
Walk	-1035.2466	615.803	-1.681	0.093	-2242.773	172.280
OtherTransp	-1049.6966	615.790	-1.705	0.088	-2257.197	157.804
WorkAtHome	-1097.6117	616.440	-1.781	0.075	-2306.389	111.165
MeanCommute	-6.9009	9.596	-0.719	0.472	-25.717	11.915
Employed	0.0459	0.005	9.489	0.000	0.036	0.055
PrivateWork	-1545.9904	752.535	-2.054	0.040	-3021.635	-70.346
PublicWork	-1658.2977	752.492	-2.204	0.028	-3133.858	-182.737
SelfEmployed	-1497.2056	752.130	-1.991	0.047	-2972.055	-22.356

```
FamilyWork      -1714.1271    755.035     -2.270      0.023   -3194.674    -233.580
Unemployment       64.5259     17.520      3.683      0.000      30.170      98.881
==============================================================================
Omnibus:                      433.600   Durbin-Watson:                   2.036
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             8258.803
Skew:                           0.138   Prob(JB):                         0.00
Kurtosis:                      11.771   Cond. No.                     1.05e+16
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly speci
[2] The smallest eigenvalue is 5.54e-18. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
"""
```