COGS9: Introduction to Data Science
Final Project
Due date: Friday 2020 December 18 23:59:59
Grading: 10% of overall course grade. 40 points total.
Completed as a group. One submission per group on Gradescope.

Group Member Information:

First Name	Last Name	PID
Yiming	Hao	A16672884
Weiyue	Li	A16226730
Yi	Li	A16316586
Xinlan	Lin	A16480021
Zhuojin	Yu	A16341528

Question

Is there an association between average wealth in the communities and their corresponding crime indexes?

Hypothesis

Our hypothesis is: The higher the average wealth in the communities, the lower their corresponding crime indexes are.

Our justification: The ArcGIS Online offers the database for the indexes of various types of crimes, such as murder, robbery, and assault, etc. And it also provides us with a profound database for household income and total home values. Based on our observations of some famous cities, we find that lower wealth areas tend to have higher crime indexes. For example, La Jolla is a well-known safe and wealthy region in the United States. In contrast, we also know that Southern Chicago is a well-known dangerous and low-income region. These facts have triggered our initial hypothesis that wealthier communities might be safer since it makes intuitive sense.

Background Information

ArcGIS is an online geographic information system that provides various data types for its users to generate maps. It includes average household income, average home value, and crime indexes in each region. According to ArcGIS, crime indexes provide a view of the relative risk of specific crime types" (Arcgis). "It is not a database of actual crimes, but rather the relative risk in an area compared to the United States in its entirety" (Arcgis). This measurement has allowed us to easily compare the safety level in each community in the United States. This is because the lower the crime index, the safer the neighborhood. The above conclusion brings up the question of what lowers the crime rates in the communities?

In the book *Social Problems*, founded on the University of Minnesota Libraries, an unnamed author has claimed that different social classes have different likelihood of committing the crime. According to this author, more impoverished people are more likely to commit street crimes than wealthier people (8.3). This conclusion brings up a discussion about what is considered street crimes. According to the book *SOU-CCJ230 Introduction to the American Criminal Justice System*, Street crimes aren't just crimes in public places; instead, it has a broader boundary. The authors have mentioned that street crimes can be as violent as homicide, rape, assault, robbery, and arson; street crimes can also be property crimes such as breaking-and-entering, burglary, and motor vehicle theft. Moreover, drug crime, hate crimes, and human trafficking are considered street crimes by the Bureau of Justice as well (Burke 1.12.). Combining these sources makes it reasonable to assume that low-income neighborhoods are more likely to have more crimes.

It is reasonable to assume that people tend to be willing to live in safer communities so that their families don't have a high chance to encounter street crimes. However, there are only limited resources according to Economics' scarcity principle. When the supply of safer communities is less than the demand for safer environments to live in, the price of houses (home values) in those safe neighborhoods will increase so that only wealthy families can afford them.

Besides these houses in safer communities, these wealthier families are more likely to offer their children higher education opportunities. According to an article on Business Insider, "From the late 1980s to 2018, the cost of an undergraduate degree has risen by 213% at public schools and 129% at private schools, adjusting for inflation" (Hillary Hoffower). This quote implies that higher education is more and more unaffordable for people, especially low-income families. This strengthens the assumption that wealthier families tend to have higher education backgrounds. In a book, *The Economics of Education*, Steve Bradley and Colin Green have said: "Economic theory implies a negative correlation between educational attainment and most crime types. Empirically, an increase in educational attainment significantly reduces subsequent violent and property crime yielding sizable social benefits" (109). This quote implies that these educated, wealthy families are less likely to commit crimes, and therefore, their communities are safer.

Data

Our goal is to find some association between the wealth of communities and their corresponding levels of safety. To better approach our goal, we have researched how to represent wealth and security levels in the communities using accessible data. There are various ways to measure wealth, and income is probably one of the most dominant measurements out there. However, when owning expensive real estate is also a sign of wealth, we also need to consider people's home values when discussing if their family is wealthy. Since we are exploring the differences between communities, we can just take their averages. These lead to our independent variables: average household-income and average home value.

In the background information, we have discussed that ArcGIS is offering crime indexes to compare the relative risk of specific crime types. We believe this is an excellent way to measure the level of safety in each community. However, since there are so many types of crimes in real life, we choose to discuss three of the most common crimes: murder, robbery, and rape. These bring out our desired dependent variables: murder index, robbery index, and rape index in each corresponding region.

Thanks to the powerful features on ArcGIS, we have found our desired average household income, average home value, murder index, robbery index, and rape index in each corresponding region. We have decided to include these data to form our perfect dataset: six different combinations of independent variables: average household income, average house value, and dependent variables: murder index, robbery index, and rape index. We have cleaned the unnecessary variables (such as the county, State, Zip, etc.) on the dataset we have obtained from ArcGIS for efficiency.

For simplicity, we chose only to obtain data of all San Diego regions to build our model. In San Diego, there are 603 observations of average home value vs. murder index, 306 observations of average home value vs. rape index, 535 observations of average home value vs. robbery index, 219 observations of average household income vs. murder index, 541 observations of average household income vs. rape index, and 159 observations of average household income vas robbery index.

Observations of Independent Variables vs. Dependent Variables in San Diego When Building the Model			
	Murder Index	Robbery Index	Rape Index
Average Household Income	219	159	541
Average Home Value	603	535	306

We have noticed that ArcGIS has given us different numbers of observations for each pair of independent and dependent variables. We will discuss this as our limitations at the end of this project.

We also chose to use the data from all Denver regions to do cross-validation for our six combinations.

Observations of Independent Variables vs. Dependent Variables in Denver when we used to do cross-validation for our six combination			
	Murder Index	Robbery Index	Rape Index
Average Household Income	143	143	143
Average Home Value	143	143	143

We also chose to use the data from all Chicago regions to do predictive analysis using our regression line of ideal combination.

Observations of Independent Variables vs. Dependent Variables in Denver when used to do predictive analysis by using our regression line of ideal combination.			
	Murder Index	Robbery Index	Rape Index
Average Household Income	N/A	1315	N/A
Average Home Value	N/A	N/A	N/A

Link to our data:
https://ucsdonline.maps.arcgis.com/home/webmap/viewer.html?webmap=17098629de7f4c148d5660b7ad0d350c

Ethical Considerations

1. Data collection
   a. Informed consent is essential in the data collection process. Participation must be voluntary because the participants have fully acknowledged the experiments' purpose/process. They have obtained the right to withdraw anytime according to their will. Because we didn't collect the data by ourselves, we have to make sure that our data sources practiced informed consent in their data collection process. For example, in our data frame about the household income and rape cases, we should ensure that informed consent is gained if personally identifiable information (PII) will be revealed.
   b. To avoid including collection bias in our data, we should make sure that the participants are chosen randomly within the proposed group. Another factor that would introduce bias to our data would be the survey design. The survey should be worded carefully and straightforwardly so that there is no misperception about the questions, and we can get more accurate answers. We also need to normalize the data collected to avoid misrepresentation. For example, we should specify the income to be yearly and in US dollars. We can also determine if the income is disposable income or before-tax income since income tax varies in each region.
   c. Limiting the exposure of personally identifiable information (PII) can help us protect the research participants' privacy and anonymity. We should collect only the data that we need to investigate our problems and avoid collecting unnecessary data that could reveal personal information. For instance, the participants should not be asked about their personal information such as names, ages, telephone numbers, or SSN during the data

collection process of this particular research. Again, since we don't collect these data on our own, we need to check the source we are obtaining from these ethical concerns.

2. Data storage
    a. To protect data security, we should have a well-rounded plan to ensure that our raw data is not exposed to the public. We will store our data and analysis on Google Drive and make sure the data can only be accessed through a specific link shared with our group members.
    b. Since we obtained our data by web scraping from ArcGIS, participants should have the full right to request their information be removed entirely from the source website, even though it has been published.
    c. Our data retention plan should include a plan to discard any data collected that reveals personal information after use—a list to delete any outdated or duplicated data if necessary.
3. Analysis
    a. We have to use whatever we have to analyze our data. It is not ethical for us to manipulate data (elicit some data and ignore others) to support our thesis statement. If our evidence doesn't support our hypothesis, we need to acknowledge it and improve our hypothesis instead of changing the data.
    b. When making graphs, we need to make sure that our visualization is presenting the data honestly. It is unethical to create charts to enlarge the difference when the difference is small or negligible. When visualizing the data, we need to express it legally without exaggerating.
    c. We should also minimize the exposure of personally identifiable information(PII) when making an analysis.
    d. To ensure our analysis's suitability and accountability, the source code we used will be shown in the analysis proposal so that people can reproduce it if the data is updated or issues are discovered.
4. Modeling
    a. We will ensure that we are using variables that are not discriminatory so that we can protect fairness among groups, such as different gender, race, and job titles.
    b. Our explanation for this experiment should be precise, proper, and easy to understand. We should use appropriate metrics to make our results more understandable and explain any limitations or shortcomings. Some possible restrictions could be the different definitions of the same crimes in other regions. For example, the definition of murder in California might be different from the definition of murder in Maine. We need to accept these limitations when testing our models and discuss them later.
5. Deployment
    a. It is essential to come up with a plan in case the participants get hurt. Therefore, since we obtain our data from sources on the Internet, we need to make sure they have this urgent plan.
    b. We will closely monitor any changes or updates of the dataset we analyzed and periodically update the static model with more recent historical data.

c. If we found out that our study's publication resulted in harming or discriminating against any groups or individuals, we will immediately withdraw our study. We would revise the fundamental research before we are ready to republish.
d. To prevent social media from misusing our findings, we should make precise statements about the purpose and the limitations of the results. We should also keep our model updated and keep track of the republication of our biased studies.

Analysis Proposal

Our goal is to determine whether there is an inverse correlation between the wealth in the communities and their corresponding crime indexes.

First of all, our analysis would start by **collecting data**. We collected data from ArcGIS by searching up the household income and the community's corresponding crime index. While doing this, we just want to extract the data we need from the vast database. For simplicity, we chose to filter the data from San Diego to build our models. For each combination, we match one independent variable and one dependent variable to form a table. We have two independent variables and three dependent variables to develop six tables on ArcGIS online. And we will download these tables as CSV files and save them to our desktop. Eventually, we will upload all six CSV files into Jupyter Notebook and rename these files accordingly.

Secondly, we will be utilizing the **data wrangling** method to clean our data. We filtered out irrelevant features such as the "aggregation method," "shape," etc., while keeping the "household income" and the "crime index." by using the python. Since we discuss several types of crimes, a CSV file will be made for each crime type. We will also keep the feature "FIPS" and set it as indexes when generating the graph. This is because the "FIPS" represents the census tract number of each census tract area, which is unique so that we can make sure our data is not misplaced. Then we will be setting up thresholds of "high murder index," "medium murder index," and "low murder index" to categorize the crime indexes. Now, we will be able to assess the value of the crime index without messing up. (For more information, please see our coding PDF).

Next, **Descriptive & Exploratory Data Analysis** will be taking place to assess if the household income analysis is appropriate for our study and if there is any correlation between wealth in the communities and the crime index. To achieve this, we will use python to produce a regression line, which could be used to fit in our dataset with the fewest residuals. We also used the A|B Testing to find the confidence interval and the p-value. Afterward, we will use python to calculate the r & r-square values to determine the correlation between two variables for each combination. (For more information, please see our coding PDF).

To make our analysis more understandable and clear, we applied **data visualization** for all six combinations, A|B testing, and the difference between actual data and predicted data. For six combinations, we use python to generate a scatter plot with a best-fit line. As a result, we will see the potential correlation between the two selected variables and our best-fit line slope. For **A|B testing**, we will use python to generate a histogram for all possible correlation coefficients we obtain from each for-loop. We will add the actual correlation coefficient to the histogram as a red dot so that we can compare the correlation coefficients between two variables after shuffling one of the columns with the actual correlation coefficient. If the red dot is far away from the histograms, it means we could not get our r-value through random chance, showing our r value is reliable. In contrast, if the red dot is inside the histograms, it means we could get our r-value through random chance, showing our r value is not reliable. We will create a group bar chart to compare the difference between actual data and predicted data during the predictive analysis. Therefore, our audiences will be able to see how accurately our selected regression line performs. (For more information, please see our coding PDF).

For our **predictive analysis**, we applied one of the techniques we learn in machine learning called **cross-validation**. As you may see, we currently have six combinations to represent the correlation between the communities' wealth and the corresponding crime index. However, we raised a question about which combination performs a more accurate analysis. To solve this problem, we will use python to perform K-Fold cross-validation. Firstly, we will split our dataset into training data and testing (validation) data with the ratio 7:3 because we need more training data to produce a better prediction. Secondly, we applied three models (LogisticRegression, SVC, and RandomForestClassifier) to calculate each combination's average accuracy score. Thirdly, we will record scores for each variable using those three models and add them to a new table. We use six combinations as our six rows and use three models as our three columns. As a result, each cell contains a score representing the average accuracy score of one specific combination using one specific model. Finally, we will select the combination that achieves a relatively high average accuracy score as our ideal combination. And, we will use this perfect combination to perform predictions. During our prediction testing, we use the ideal combination to predict the crime index level in Chicago. By doing this, we first calculate the **regression line** for our ideal combination. We will also use the variable for wealth in Chicago as x values and predict the crime index and use the variable for crime index in Chicago as y values. We will eventually assign corresponding labels based on what we got for the crime index (like we mentioned before the **classification**: low crime index, medium crime index, and high crime index) as our predicted result. We will then compare the predicted product with the actual product to check the performance of the ideal combination (For more information, please see our coding PDF).

Besides, we also applied geospatial analysis to support our hypothesis. Each choropleth map will show the correlation between the two variables.

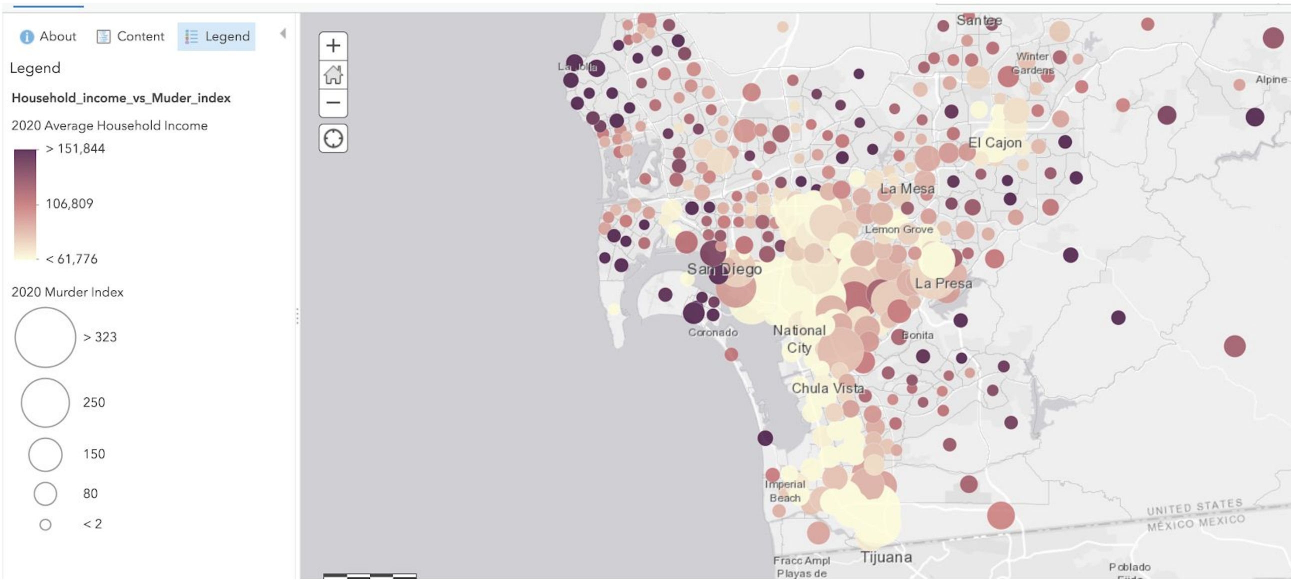


Figure 1: The larger the circles are, the higher the murder index will be in that census tract area, and the darker the color, the higher the average household income will be in that census tract areas. We can see mostly large light circles and mostly small dark circles from this cartogram, which may support our hypothesis.

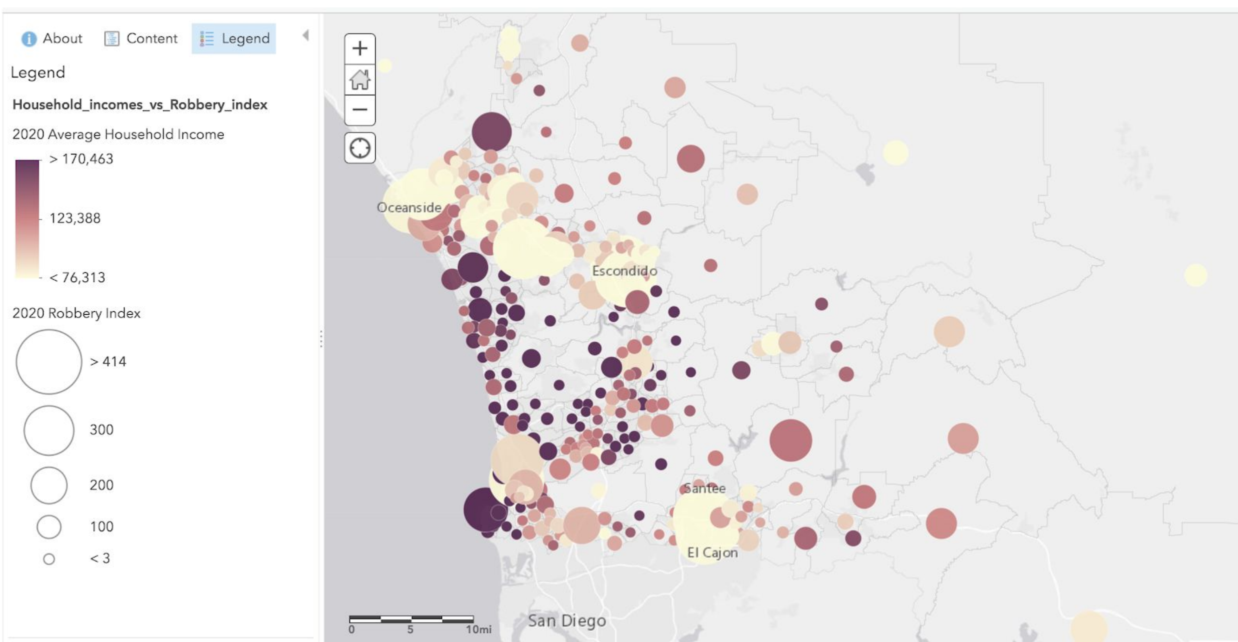


Figure 2: The larger the circles are, the higher the robbery index will be in that census tract area, and the darker the color, the higher the average household income will be in that census tract area. We can see mostly large light circles and mostly small dark circles from this cartogram, which may support our hypothesis.

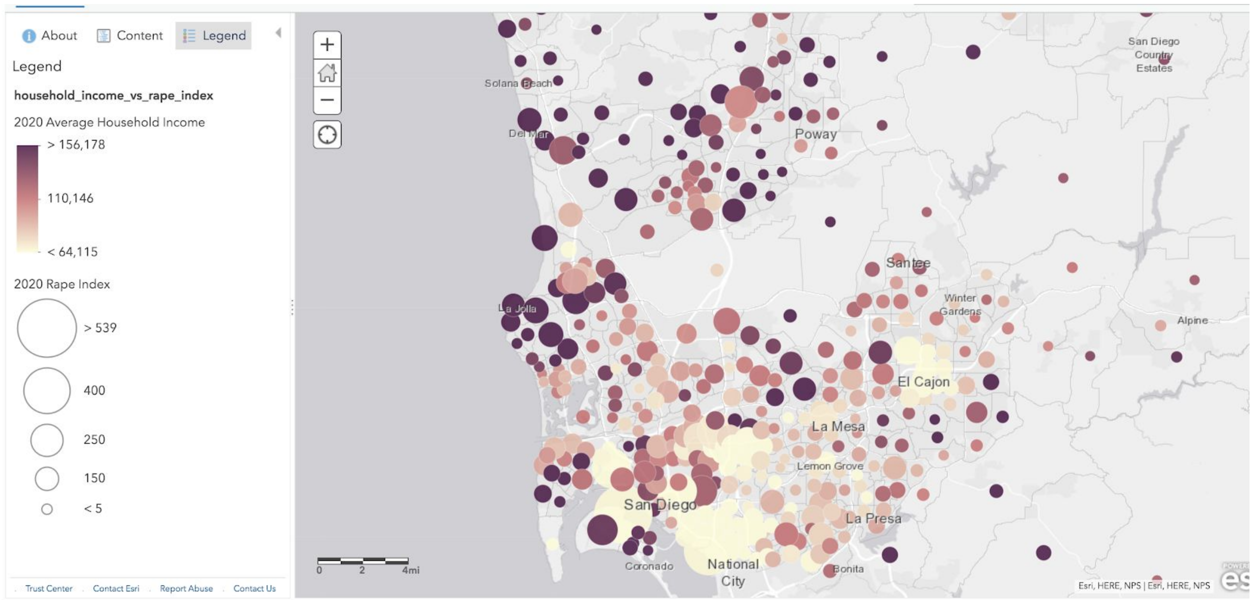


Figure 3: The larger the circles are, the higher the rape index will be in that census tract area, and the darker the color, the higher the average household income will be in that census tract areas. We can see mostly large light circles and mostly small dark circles from this cartogram, which may support our hypothesis.

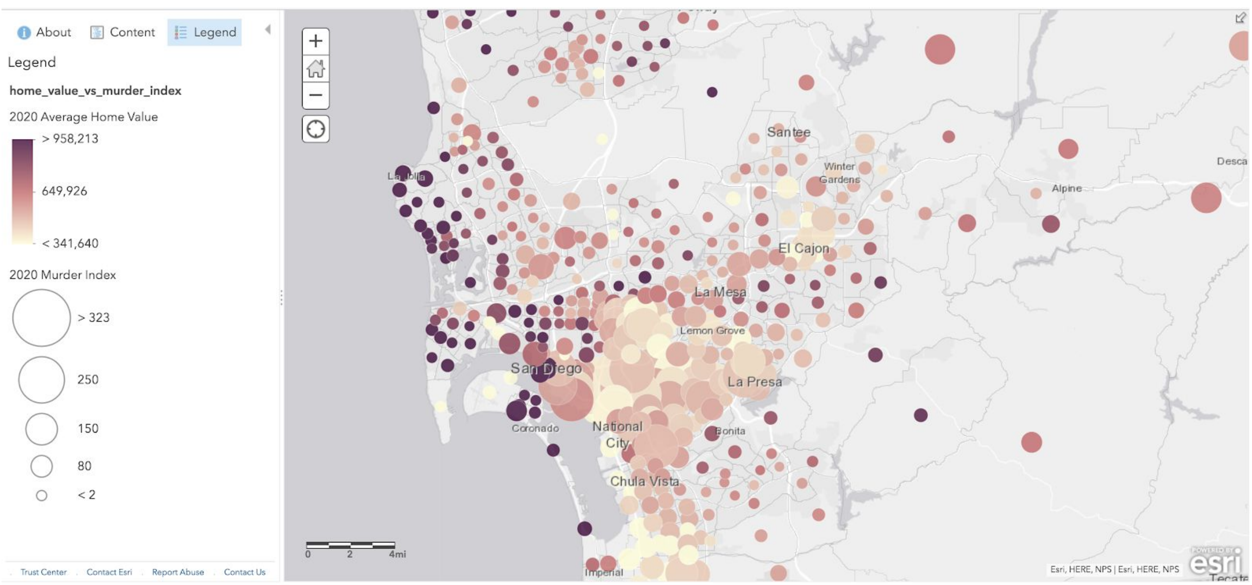


Figure 4: The larger the circles are, the higher the murder index will be in that census tract area, and the darker the color, the higher the average home value will be in that census tract areas. We can see mostly large light circles and mostly small dark circles from this cartogram, which may support our hypothesis.

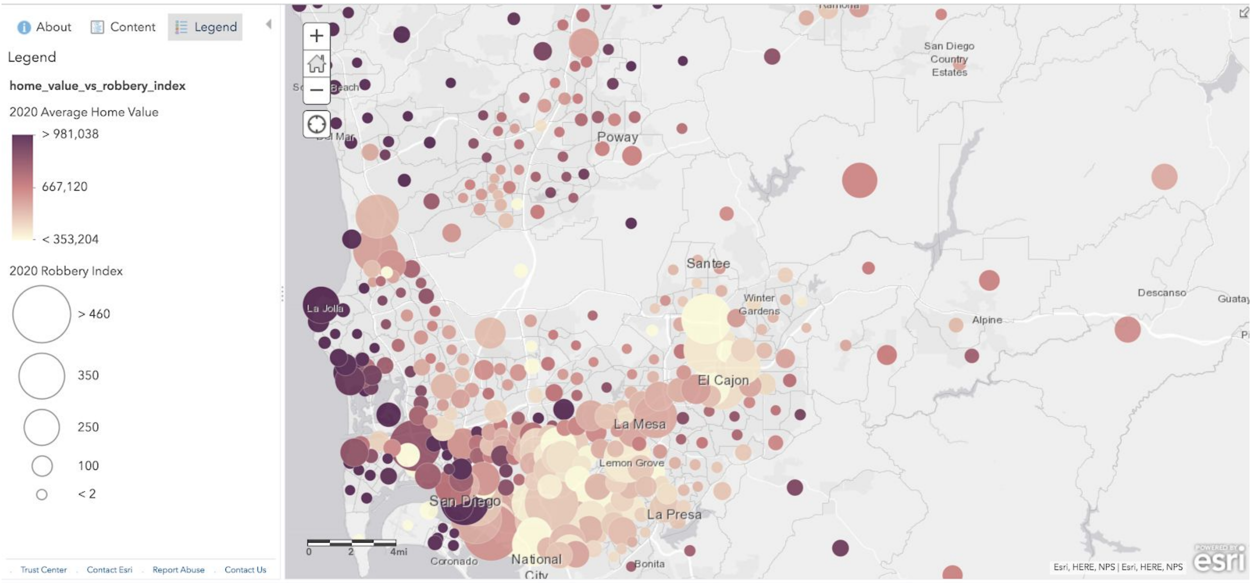


Figure 5: The larger the circles are, the higher the robbery index will be in that census tract area, and the darker the color, the higher the average home value will be in that census tract area. We can see mostly large light circles and mostly small dark circles from this cartogram, which may support our hypothesis.

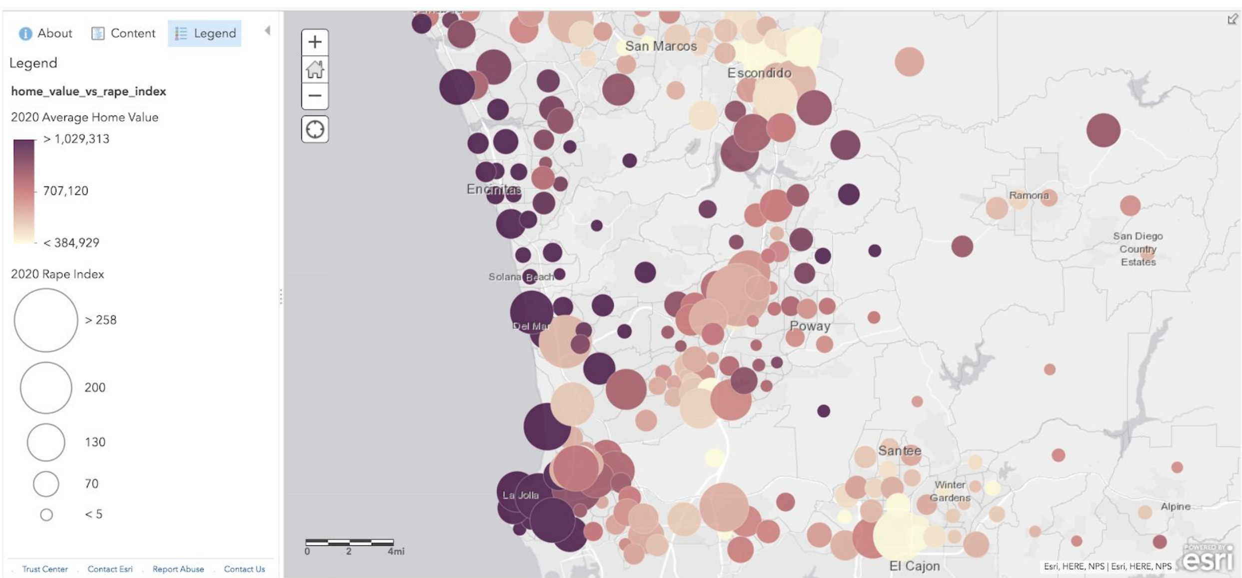


Figure 6: The larger the circles are, the higher the rape index will be in that census tract area, and the darker the color, the higher the home value will be in that census tract areas. From this cartogram, we can see some large light circles and several large dark circles, which may not support our hypothesis.

The following pages are our actual analysis and the Discussion and Participation sections are after these codes.

Cognitive Science Final

December 18, 2020

1 imports

[147]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

2 Table for average household income verses murder index in San Diego county

[148]:
```
income_murder_raw = pd.read_csv('household_income_vs_murder_index.csv')
income_murder_raw

# cleaning data
income_murder = income_murder_raw.get(['FIPS', 'AVGHINC_CY', 'CRMCYMURD'])
income_murder

# rename column name
income_murder = (income_murder.assign(average_household_income = income_murder.
 ↪get('AVGHINC_CY'),
                                      murder_index = income_murder.
 ↪get('CRMCYMURD'),
                                      San_Diego_census_tract_number =␣
 ↪income_murder.get('FIPS'))
                )
# cleaning data
income_murder = income_murder.drop(columns = ['AVGHINC_CY', 'CRMCYMURD',␣
 ↪'FIPS'])
income_murder = income_murder.set_index('San_Diego_census_tract_number')
income_murder

# delete meaningless data
income_murder = (income_murder[(income_murder.get('average_household_income') >␣
 ↪0) &
                               (income_murder.get('murder_index') > 0)]
```

```
                 )
income_murder

# classify the data based on the murder index, and labeling
def classification (values):
    if values >= 126:
        return 'High Murder Index'
    elif (values < 126) & (values >= 53):
        return 'Medium Murder Index'
    else:
        return 'Low Murder Index'

# Adding labels to a new column
income_murder = income_murder.assign(murder_index_level = income_murder.
 ↪get('murder_index'))
income_murder = (income_murder.assign(murder_index_level = income_murder.
 ↪get('murder_index')
                                                                          ␣
 ↪            .apply(classification))
                       )
income_murder
```

[148]:

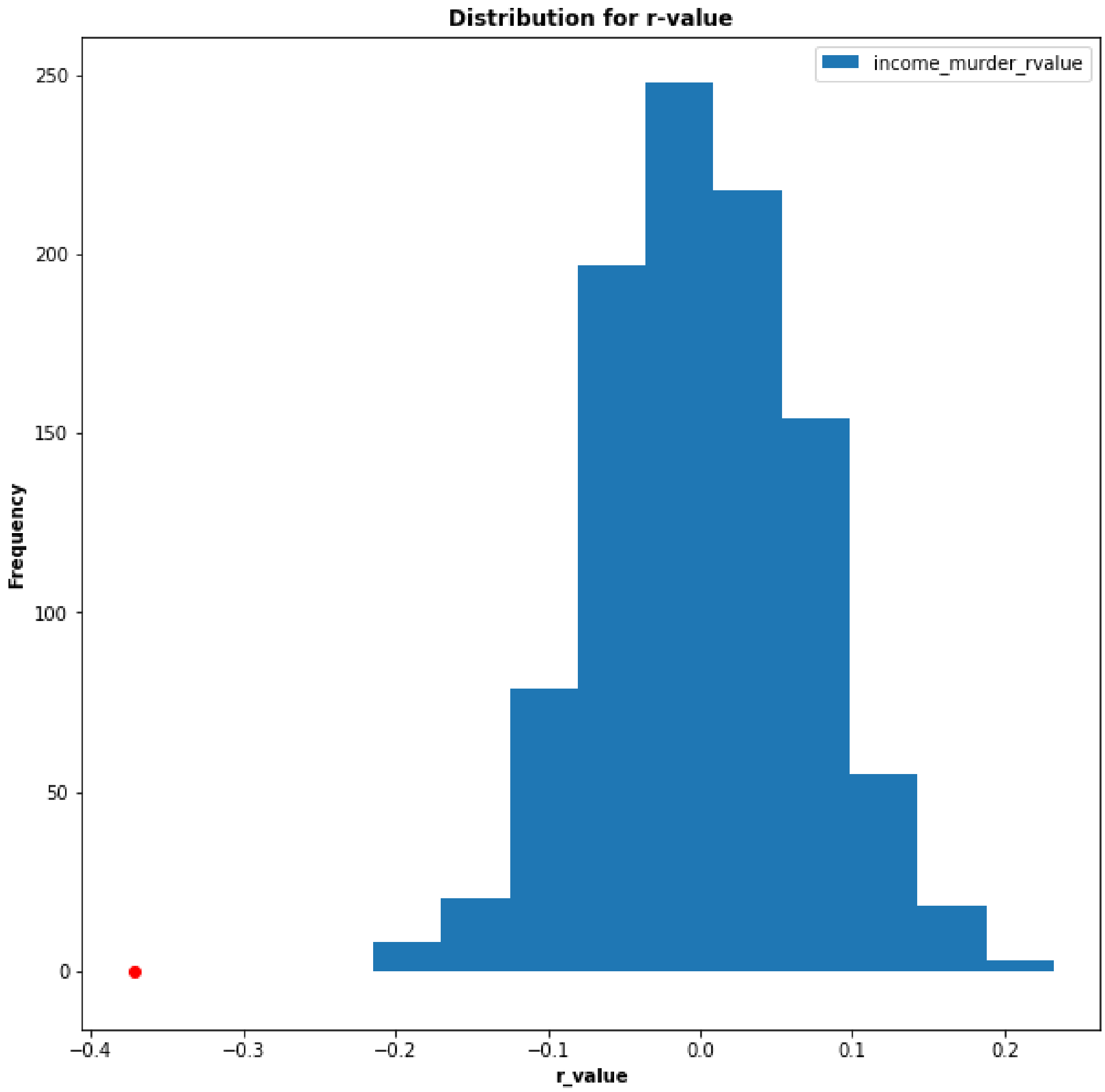
```
                                average_household_income  murder_index  \
San_Diego_census_tract_number
6073020109                                       98031.0          42.0
6073017010                                      141107.0           9.0
6073020027                                      149719.0           8.0
6073017034                                      139612.0          15.0
6073017501                                      166988.0           4.0
…                                                    …            …
6073008339                                       88171.0          41.0
6073020107                                      101953.0          19.0
6073019701                                       73599.0          18.0
6073020016                                      191105.0           4.0
6073021500                                      241878.0           4.0

                              murder_index_level
San_Diego_census_tract_number
6073020109                      Low Murder Index
6073017010                      Low Murder Index
6073020027                      Low Murder Index
6073017034                      Low Murder Index
6073017501                      Low Murder Index
…                                              …
6073008339                      Low Murder Index
6073020107                      Low Murder Index
6073019701                      Low Murder Index
```

```
6073020016                        Low Murder Index
6073021500                        Low Murder Index

[219 rows x 3 columns]
```

3 A | B testing; confidence interval

[149]:
```
# import
from scipy import stats

income_murder

# shuffling columns "murder index", and calculate the r-value for each round
possible_correlation_1 = np.array([])

for i in np.arange(1000):
    shuffling_1 = np.random.permutation(income_murder.get('murder_index'))
    income_murder_shuffle = income_murder.assign(income_murder_shuffling =␣
 ↪shuffling_1)
    rvalue_1ab = (stats.pearsonr(income_murder_shuffle.
 ↪get('average_household_income'),
                                income_murder_shuffle.
 ↪get('income_murder_shuffling'))[0]
                 )
    possible_correlation_1 = np.append(possible_correlation_1, rvalue_1ab)

# visualize the distribution for r-values of each round, and plot the red dot␣
 ↪for the real r-value
y1 = income_murder.get('murder_index')
x1 = income_murder.get('average_household_income')
pd.DataFrame().assign(income_murder_rvalue = possible_correlation_1).plot(kind␣
 ↪= 'hist')
plt.scatter(stats.pearsonr(x1, y1)[0], 0 , color = 'red')
plt.title('Distribution for r-value', fontweight='bold')
plt.gcf().set_size_inches((10, 10))
plt.xlabel('r_value', fontweight='bold')
plt.ylabel('Frequency', fontweight='bold')

# From the graph, we can see the red dot is far away from the main distribution.
 ↪
# It tell us there are correlation between two variable, and it is not due to␣
 ↪random chance.
```

[149]:
```
Text(0, 0.5, 'Frequency')
```

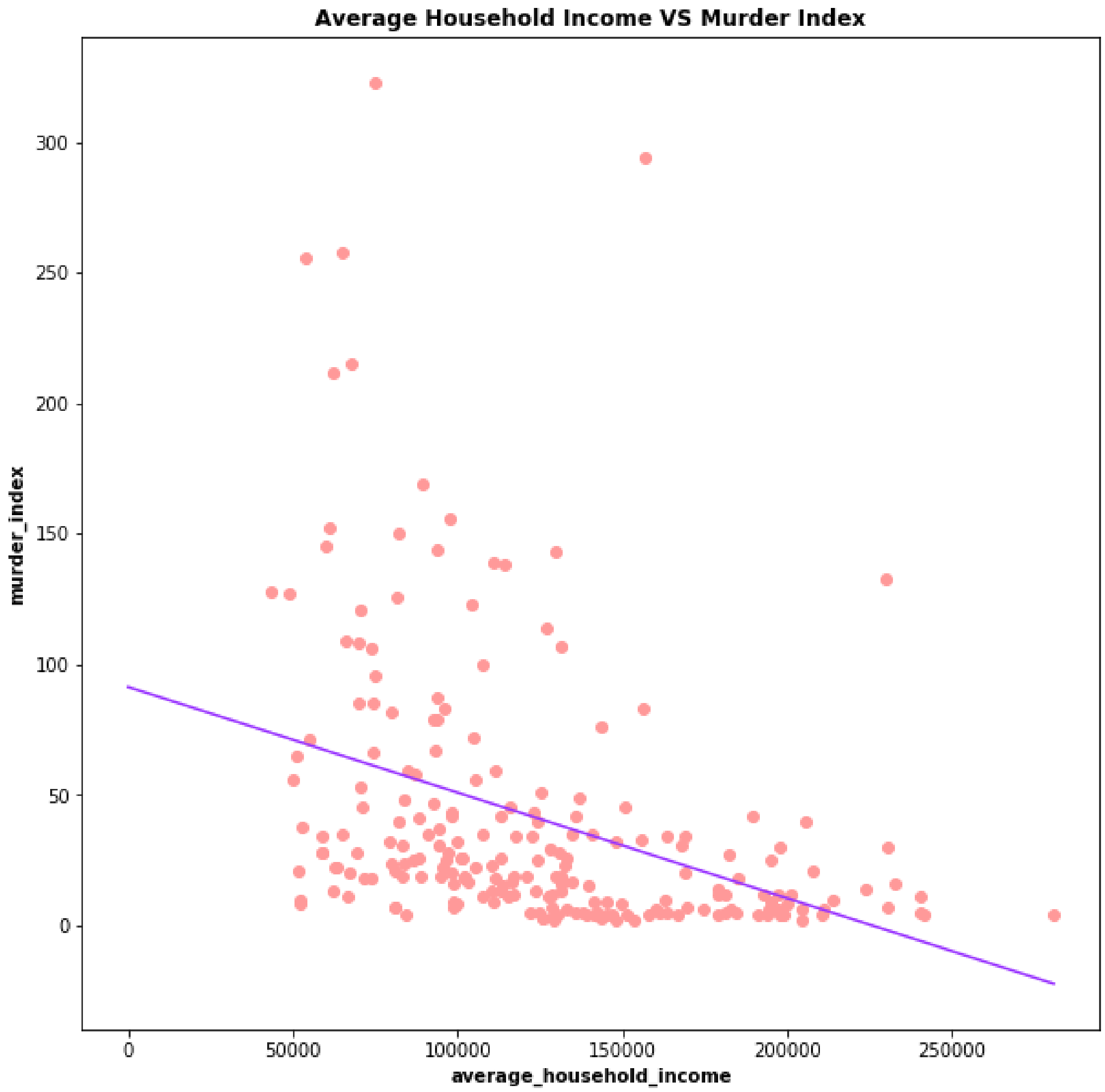


4 Visualizing the correlation between average household income and murder index, with statistic summary, such as r-square, p-value, regression line equation

[150]:
```
# making scatter plots for visualization
y1 = income_murder.get('murder_index')
x1 = income_murder.get('average_household_income')

# Give title for graph; set the color for each dots
plt.scatter(x1, y1, color = '#ff9999')
plt.title('Average Household Income VS Murder Index', fontweight='bold')
```

```python
# Add axis label
plt.xlabel('average_household_income', fontweight='bold')
plt.ylabel('murder_index', fontweight='bold')

# produce regression (best fit line) line
model_1 = np.polyfit(x1, y1, 1)
predict_1 = np.poly1d(model_1)

# calculate r-squared value
r_matrix_1 = np.corrcoef(x1, y1)
r_1 = r_matrix_1[0,1]
r2_1 = r_1 ** 2

# calculate p-value
from scipy import stats
pvalue_1 = stats.pearsonr(x1, y1)[1]

# visualize the regression line
x_lin_reg_1 = range(0, int(x1.max()))
y_lin_reg_1 = predict_1(x_lin_reg_1)
plt.plot(x_lin_reg_1, y_lin_reg_1, c = '#9933ff')
plt.gcf().set_size_inches((10, 10))

# print out the equation for regression line and corresponding r-squared values
print('Statistic Summary:')
print("y " + "= " + str(model_1[0]) + 'x ' + '+ ' + str(model_1[1]))
print('R value is ' + str(r_1))
print('R-squared value is ' + str(r2_1))
print('P value is ' + str(pvalue_1))
```

```
Statistic Summary:
y = -0.00040427945675277763x + 91.2861765029504
R value is -0.3714757697992417
R-squared value is 0.1379942475479392
P value is 1.4273055552517142e-08
```



5 Table for average household income verses robbery index in San Diego county

[151]:
```
income_robbery_raw = pd.read_csv('household_income_vs_robbery_index.csv')
income_robbery_raw

# cleaning data
income_robbery = income_robbery_raw.get(['FIPS', 'AVGHINC_CY', 'CRMCYROBB'])
income_robbery
```

```
# rename column name
income_robbery = (income_robbery.assign(average_household_income =␣
↪income_robbery.get('AVGHINC_CY'),
                                       robbery_index = income_robbery.
↪get('CRMCYROBB'),
                                       San_Diego_census_tract_number =␣
↪income_robbery.get('FIPS'))
                 )
# cleaning data
income_robbery = income_robbery.drop(columns = ['AVGHINC_CY', 'CRMCYROBB',␣
↪'FIPS'])
income_robbery = income_robbery.set_index('San_Diego_census_tract_number')
income_robbery

# delete meaningless data
income_robbery = (income_robbery[(income_robbery.
↪get('average_household_income') > 0) &
                                (income_robbery.get('robbery_index') > 0)]
                 )
income_robbery

# classify the data based on the robbery index, and labeling
def classification_robbery (values):
    if values >= 107:
        return 'High Robbery Index'
    elif (values < 107) & (values >= 52):
        return 'Medium Robbery Index'
    else:
        return 'Low Robbery Index'

# Adding labels to a new column
income_robbery = income_robbery.assign(robbery_index_level = income_robbery.
↪get('robbery_index'))
income_robbery = (income_robbery.assign(robbery_index_level = income_robbery.
↪get('robbery_index')
                                 .apply(classification_robbery))
                       )
income_robbery
```

[151]:

```
                               average_household_income  robbery_index  \
San_Diego_census_tract_number
6073017106                                     230201.0           69.0
6073017104                                     153316.0            5.0
6073017107                                     204324.0            6.0
6073017108                                     160258.0           19.0
6073017109                                     182747.0           53.0
```

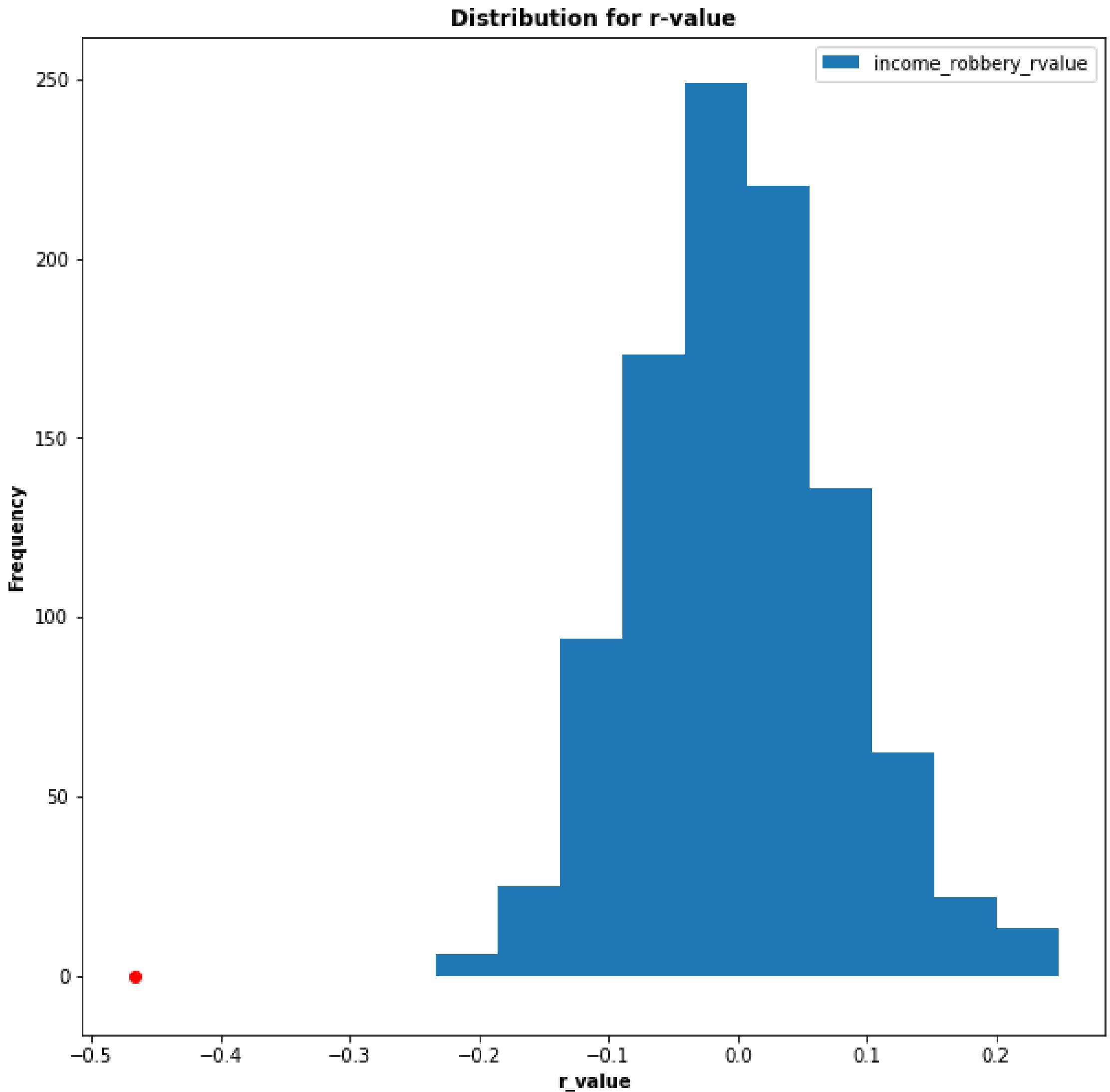
```
…                                                         …             …
6073020809                                          95722.0         101.0
6073020902                                         110753.0         162.0
6073020904                                          93463.0         164.0
6073020903                                          74925.0         114.0
6073021000                                          54999.0          97.0

                                robbery_index_level
San_Diego_census_tract_number
6073017106                      Medium Robbery Index
6073017104                         Low Robbery Index
6073017107                         Low Robbery Index
6073017108                         Low Robbery Index
6073017109                      Medium Robbery Index
…                                                …
6073020809                      Medium Robbery Index
6073020902                        High Robbery Index
6073020904                        High Robbery Index
6073020903                        High Robbery Index
6073021000                      Medium Robbery Index

[159 rows x 3 columns]
```

6 A | B testing; confidence interval

[152]:
```python
# import
from scipy import stats

income_robbery

# shuffling columns "robbery index", and calculate the r-value for each round
possible_correlation_2 = np.array([])

for i in np.arange(1000):
    shuffling_2 = np.random.permutation(income_robbery.get('robbery_index'))
    income_robbery_shuffle = income_robbery.assign(income_robbery_shuffling =␣
 ↪shuffling_2)
    rvalue_2ab = (stats.pearsonr(income_robbery_shuffle.
 ↪get('average_household_income'),
                                income_robbery_shuffle.
 ↪get('income_robbery_shuffling'))[0]
                 )
    possible_correlation_2 = np.append(possible_correlation_2, rvalue_2ab)
```

```
# visualize the distribution for r-values of each round, and plot the red dot␣
↪for the real r-value
y2 = income_robbery.get('robbery_index')
x2 = income_robbery.get('average_household_income')
pd.DataFrame().assign(income_robbery_rvalue = possible_correlation_2).plot(kind␣
↪= 'hist')
plt.scatter(stats.pearsonr(x2, y2)[0], 0 , color = 'red')
plt.title('Distribution for r-value', fontweight='bold')
plt.gcf().set_size_inches((10, 10))
plt.xlabel('r_value', fontweight='bold')
plt.ylabel('Frequency', fontweight='bold')
# From the graph, we can see the red dot is far away from the main distribution.
↪
# It tell us there are correlation between two variable, and it is not due to␣
↪random chance.
```

[152]:
```
Text(0, 0.5, 'Frequency')
```

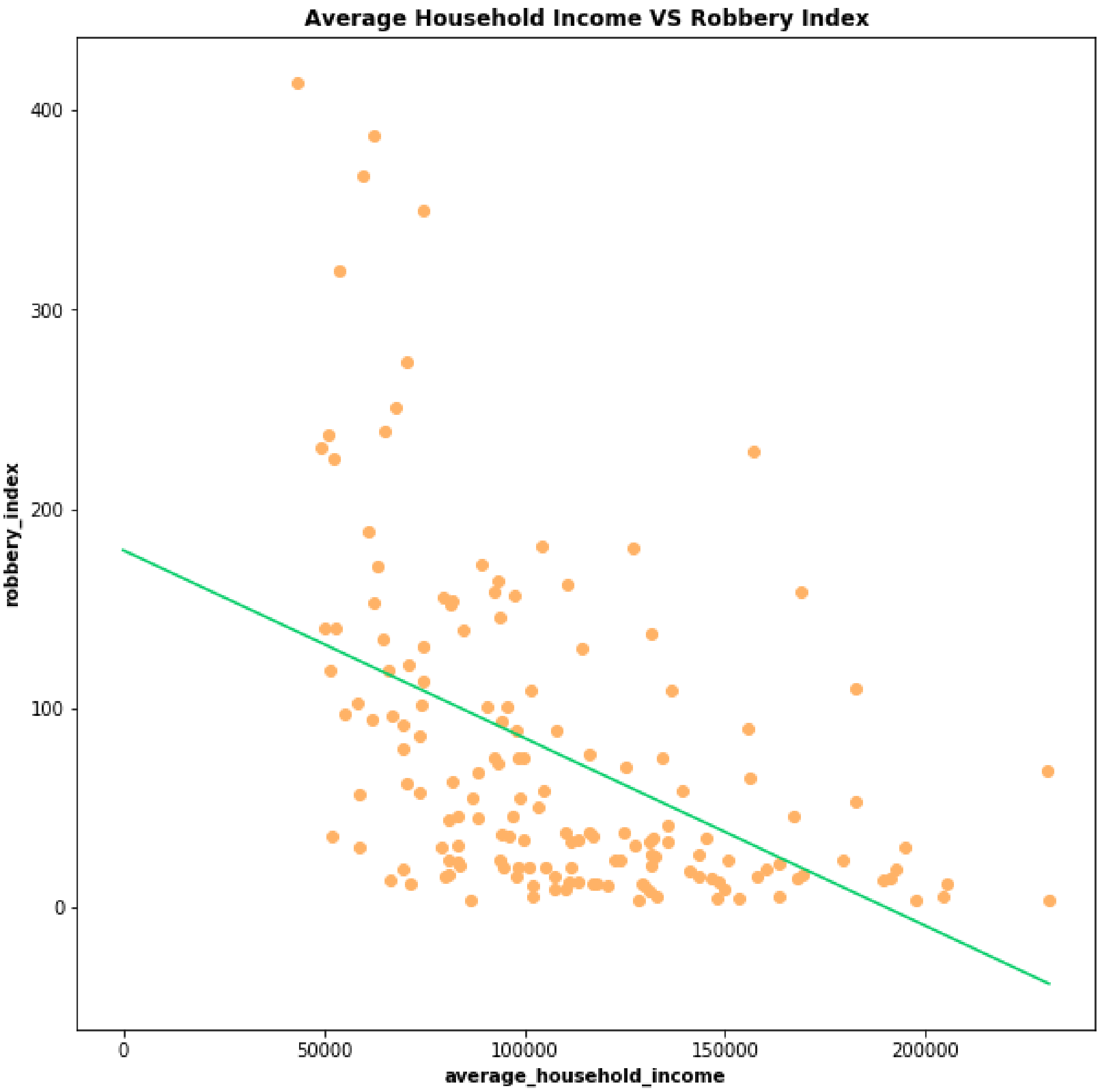


7 Visualizing the correlation between average household income and robbery index, with statistic summary, such as r-square, p-value, regression line equation

[153]:
```
# making scatter plots for visualization
y2 = income_robbery.get('robbery_index')
x2 = income_robbery.get('average_household_income')

# Give title for graph; set the color for each dots
plt.scatter(x2, y2, color = '#ffb266')
plt.title('Average Household Income VS Robbery Index', fontweight='bold')
```

```python
# Add axis label
plt.xlabel('average_household_income', fontweight='bold')
plt.ylabel('robbery_index', fontweight='bold')

# produce regression (best fit line) line
model_2 = np.polyfit(x2, y2, 1)
predict_2 = np.poly1d(model_2)

# calculate r-squared and r value
r_matrix_2 = np.corrcoef(x2, y2)
r_2 = r_matrix_2[0,1]
r2_2 = r_2 ** 2

# calculate p-value
from scipy import stats
pvalue_2 = stats.pearsonr(x2, y2)[1]

# visualize the regression line
x_lin_reg_2 = range(0, int(x2.max()))
y_lin_reg_2 = predict_2(x_lin_reg_2)
plt.plot(x_lin_reg_2, y_lin_reg_2, c = '#00cc66')
plt.gcf().set_size_inches((10, 10))

# print out the equation for regression line and corresponding r-squared values
print('Statistic Summary:')
print("y " + "= " + str(model_2[0]) + 'x ' +  '+ ' + str(model_2[1]))
print('R value is ' + str(r_2))
print('R-squared value is ' + str(r2_2))
print('P value is ' + str(pvalue_2))
```

```
Statistic Summary:
y = -0.0009421105807451566x + 179.25153282920314
R value is -0.46669005436169064
R-squared value is 0.21779960684011776
P value is 5.628856602677929e-10
```



8 Table for average household income verses rape index in San Diego county

[154]:
```
income_rape_raw = pd.read_csv('household_income_vs_rape_index.csv')
income_rape_raw

# cleaning data
income_rape = income_rape_raw.get(['FIPS', 'AVGHINC_CY', 'CRMCYRAPE'])
income_rape

# rename column name
```

```
income_rape = (income_rape.assign(average_household_income = income_rape.
 ↪get('AVGHINC_CY'),
                                      rape_index = income_rape.get('CRMCYRAPE'),
                                      San_Diego_census_tract_number =␣
 ↪income_rape.get('FIPS'))
                )
# cleaning data
income_rape = income_rape.drop(columns = ['AVGHINC_CY', 'CRMCYRAPE', 'FIPS'])
income_rape = income_rape.set_index('San_Diego_census_tract_number')
income_rape

# delete meaningless data
income_rape = (income_rape[(income_rape.get('average_household_income') > 0) &
                               (income_rape.get('rape_index') > 0)]
                )
income_rape

# classify the data based on the rape index, and labeling
def classification_rape (values):
    if values >= 131:
        return 'High Rape Index'
    elif (values < 131) & (values >= 86):
        return 'Medium Rape Index'
    else:
        return 'Low Rape Index'

# Adding labels to a new column
income_rape = income_rape.assign(rape_index_level = income_rape.
 ↪get('rape_index'))
income_rape = (income_rape.assign(rape_index_level = income_rape.
 ↪get('rape_index')
                          .apply(classification_rape))
                      )
income_rape
```

[154]:

```
                              average_household_income  rape_index  \
San_Diego_census_tract_number
6073013601                                    125163.0        47.0
6073017808                                    192601.0        56.0
6073007702                                     87375.0        51.0
6073008358                                     99626.0       100.0
6073018511                                     69526.0       123.0
…                                                    …           …
6073017104                                    153316.0        23.0
6073003303                                     47930.0        89.0
6073020710                                    189518.0        97.0
6073020809                                     95722.0        36.0
```

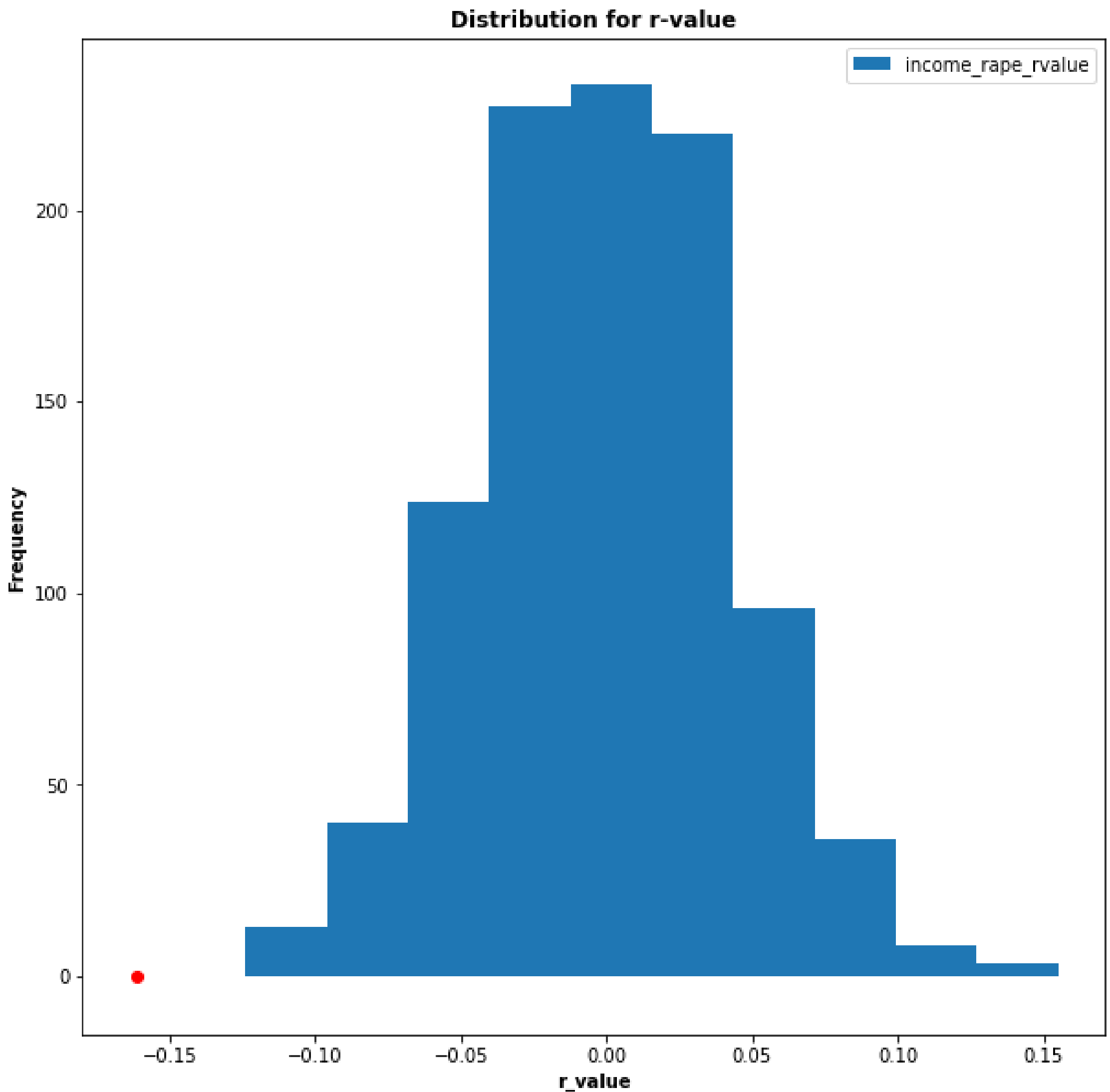
```
6073017035                                    105465.0       258.0

                                 rape_index_level
San_Diego_census_tract_number
6073013601                         Low Rape Index
6073017808                         Low Rape Index
6073007702                         Low Rape Index
6073008358                      Medium Rape Index
6073018511                      Medium Rape Index
…                                            …
6073017104                         Low Rape Index
6073003303                      Medium Rape Index
6073020710                      Medium Rape Index
6073020809                         Low Rape Index
6073017035                        High Rape Index

[541 rows x 3 columns]
```

9 A | B testing; confidence interval

[155]:
```
# import
from scipy import stats

income_rape

# shuffling columns "rape index", and calculate the r-value for each round
possible_correlation_3 = np.array([])

for i in np.arange(1000):
    shuffling_3 = np.random.permutation(income_rape.get('rape_index'))
    income_rape_shuffle = income_rape.assign(income_rape_shuffling =␣
 ↪shuffling_3)
    rvalue_3ab = (stats.pearsonr(income_rape_shuffle.
 ↪get('average_household_income'),
                                income_rape_shuffle.
 ↪get('income_rape_shuffling'))[0]
                 )
    possible_correlation_3 = np.append(possible_correlation_3, rvalue_3ab)

# visualize the distribution for r-values of each round, and plot the red dot␣
 ↪for the real r-value
y3 = income_rape.get('rape_index')
x3 = income_rape.get('average_household_income')
pd.DataFrame().assign(income_rape_rvalue = possible_correlation_3).plot(kind =␣
 ↪'hist')
```

```
plt.scatter(stats.pearsonr(x3, y3)[0], 0 , color = 'red')
plt.title('Distribution for r-value', fontweight='bold')
plt.gcf().set_size_inches((10, 10))
plt.xlabel('r_value', fontweight='bold')
plt.ylabel('Frequency', fontweight='bold')
# From the graph, we can see the red dot is far away from the main distribution.
↪
# It tell us there are correlation between two variable, and it is not due to␣
↪random chance.
```

[155]: Text(0, 0.5, 'Frequency')



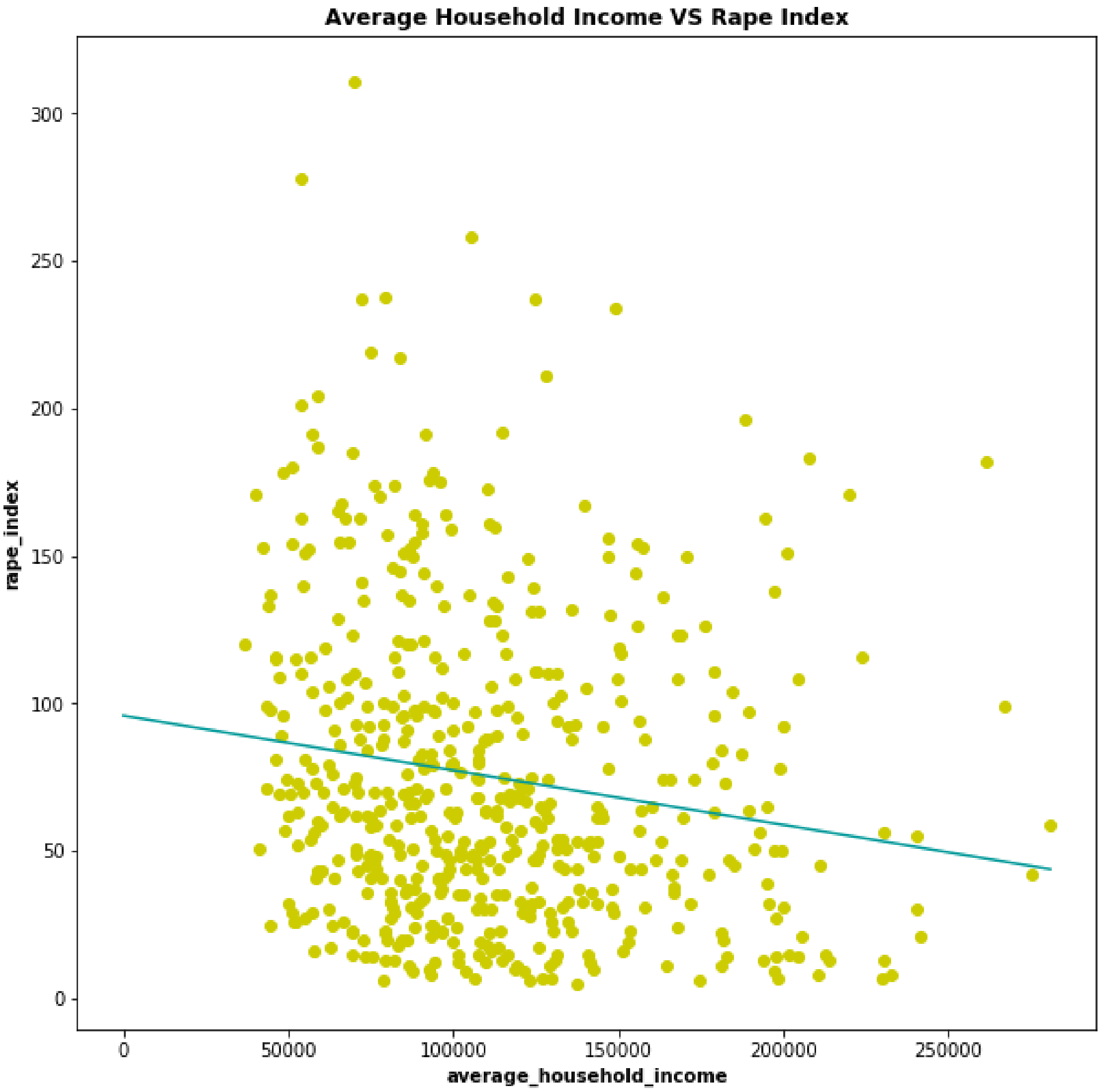
10 Visualizing the correlation between average household income and rape index, with statistic summary, such as r-square, p-value, regression line equation

[156]:

```python
# making scatter plots for visualization
y3 = income_rape.get('rape_index')
x3 = income_rape.get('average_household_income')

# Give title for graph; set the color for each dots
plt.scatter(x3, y3, color = '#cccc00')
plt.title('Average Household Income VS Rape Index', fontweight='bold')

# Add axis label
plt.xlabel('average_household_income', fontweight='bold')
plt.ylabel('rape_index', fontweight='bold')

# produce regression (best fit line) line
model_3 = np.polyfit(x3, y3, 1)
predict_3 = np.poly1d(model_3)

# calculate r-squared and r value
r_matrix_3 = np.corrcoef(x3, y3)
r_3 = r_matrix_3[0,1]
r2_3 = r_3 ** 2

# calculate p-value
from scipy import stats
pvalue_3 = stats.pearsonr(x3, y3)[1]

# visualize the regression line
x_lin_reg_3 = range(0, int(x3.max()))
y_lin_reg_3 = predict_3(x_lin_reg_3)
plt.plot(x_lin_reg_3, y_lin_reg_3, c = '#009999')
plt.gcf().set_size_inches((10, 10))

# print out the equation for regression line and corresponding r-squared values
print('Statistic Summary:')
print("y " + "= " + str(model_3[0]) + 'x ' + '+ ' + str(model_3[1]))
print('R value is ' + str(r_3))
print('R-squared value is ' + str(r2_3))
print('P value is ' + str(pvalue_3))
```

```
Statistic Summary:
y = -0.00018508552875060643x + 95.82001791250997
R value is -0.16140457582230458
R-squared value is 0.02605143709637807
```

```
P value is 0.00016309104508716005
```



11 Table for average home value verses murder index in San Diego county

[157]:
```
home_value_murder_raw = pd.read_csv('home_value_vs_murder_index.csv')
home_value_murder_raw

# cleaning data
home_value_murder = home_value_murder_raw.get(['FIPS', 'AVGVAL_CY',␣
↪'CRMCYMURD'])
```

```
home_value_murder

# rename column name
home_value_murder = (home_value_murder.assign(average_home_value =␣
↪home_value_murder.get('AVGVAL_CY'),
                                              murder_index = home_value_murder.
↪get('CRMCYMURD'),
                                              San_Diego_census_tract_number =␣
↪home_value_murder.get('FIPS'))
                )
# cleaning data
home_value_murder = home_value_murder.drop(columns = ['AVGVAL_CY', 'CRMCYMURD',␣
↪'FIPS'])
home_value_murder = home_value_murder.set_index('San_Diego_census_tract_number')
home_value_murder

# delete meaningless data
home_value_murder = (home_value_murder[(home_value_murder.
↪get('average_home_value') > 0) &
                               (home_value_murder.get('murder_index') > 0)]
                )
home_value_murder

# classify the data based on the murder index, and labeling
def classification (values):
    if values >= 126:
        return 'High Murder Index'
    elif (values < 126) & (values >= 53):
        return 'Medium Murder Index'
    else:
        return 'Low Murder Index'

# Adding labels to a new column
home_value_murder = home_value_murder.assign(murder_index_level =␣
↪home_value_murder.get('murder_index'))
home_value_murder = (home_value_murder.assign(murder_index_level =␣
↪home_value_murder.get('murder_index')
                                                                              ␣
↪          .apply(classification))
                      )
home_value_murder
```

[157]:

```
                               average_home_value  murder_index  \
San_Diego_census_tract_number
6073018906                               529195.0          25.0
6073002712                               319329.0         102.0
```

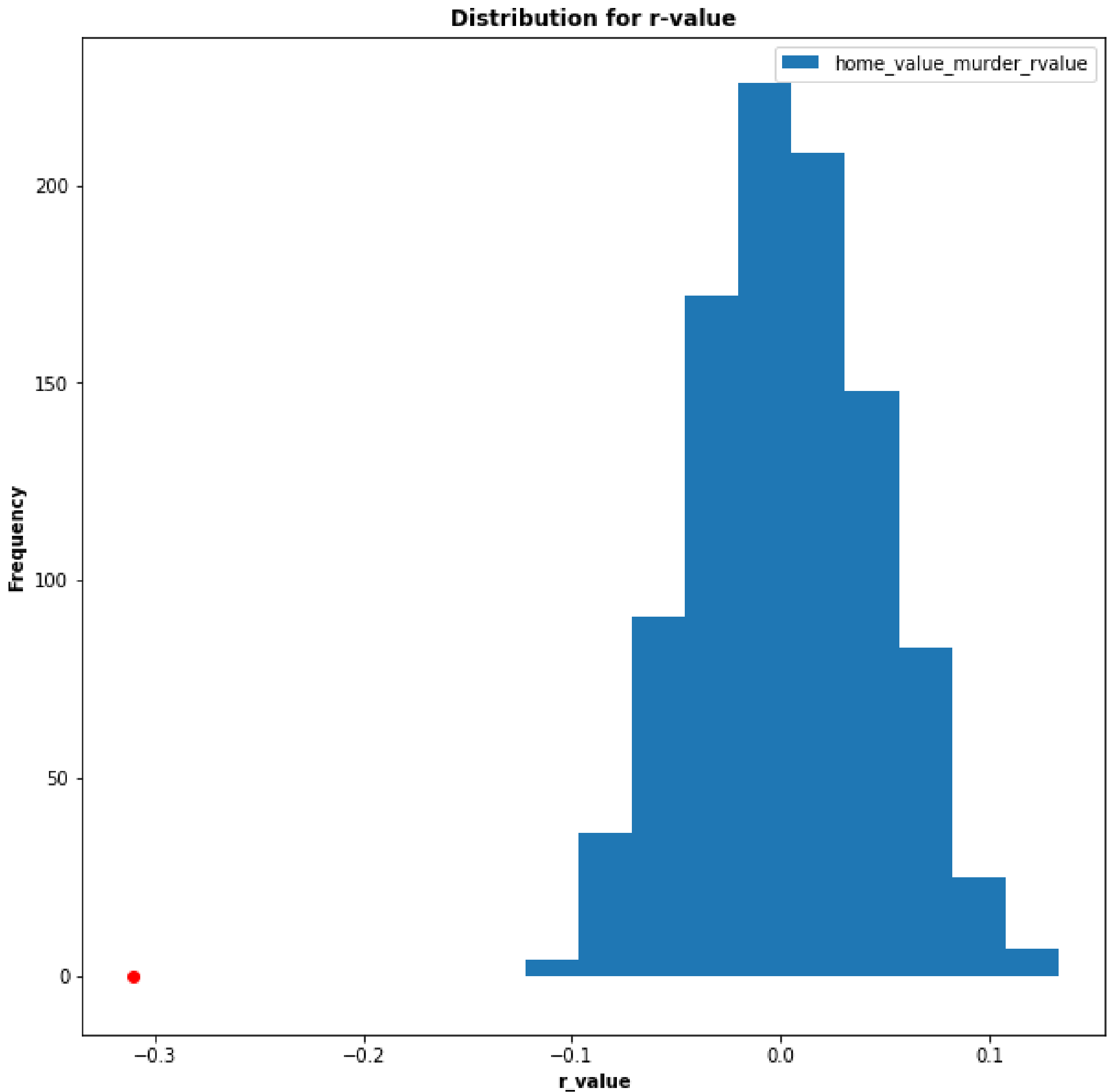
```
6073020023                               491815.0           8.0
6073007702                               908984.0          18.0
6073009301                               597041.0          21.0
…                                               …            …
6073005400                               983568.0          64.0
6073014901                               765858.0          19.0
6073018610                               570000.0          79.0
6073018504                               806116.0         114.0
6073020307                               444855.0          87.0

                              murder_index_level
San_Diego_census_tract_number
6073018906                      Low Murder Index
6073002712                   Medium Murder Index
6073020023                      Low Murder Index
6073007702                      Low Murder Index
6073009301                      Low Murder Index
…                                              …
6073005400                   Medium Murder Index
6073014901                      Low Murder Index
6073018610                   Medium Murder Index
6073018504                   Medium Murder Index
6073020307                   Medium Murder Index

[603 rows x 3 columns]
```

12 A | B testing; confidence interval

[158]:
```
# import
from scipy import stats

home_value_murder

# shuffling columns "murder index", and calculate the r-value for each round
possible_correlation_4 = np.array([])

for i in np.arange(1000):
    shuffling_4 = np.random.permutation(home_value_murder.get('murder_index'))
    home_value_murder_shuffle = home_value_murder.
 ↪assign(home_value_murder_shuffling = shuffling_4)
    rvalue_4ab = (stats.pearsonr(home_value_murder_shuffle.
 ↪get('average_home_value'),
                                home_value_murder_shuffle.
 ↪get('home_value_murder_shuffling'))[0]
                 )
```

```
    possible_correlation_4 = np.append(possible_correlation_4, rvalue_4ab)

# visualize the distribution for r-values of each round, and plot the red dot␣
↪for the real r-value
y4 = home_value_murder.get('murder_index')
x4 = home_value_murder.get('average_home_value')
pd.DataFrame().assign(home_value_murder_rvalue = possible_correlation_4).
↪plot(kind = 'hist')
plt.scatter(stats.pearsonr(x4, y4)[0], 0 , color = 'red')
plt.title('Distribution for r-value', fontweight='bold')
plt.gcf().set_size_inches((10, 10))
plt.xlabel('r_value', fontweight='bold')
plt.ylabel('Frequency', fontweight='bold')
# From the graph, we can see the red dot is far away from the main distribution.
↪
# It tell us there are correlation between two variable, and it is not due to␣
↪random chance.
```

[158]: Text(0, 0.5, 'Frequency')

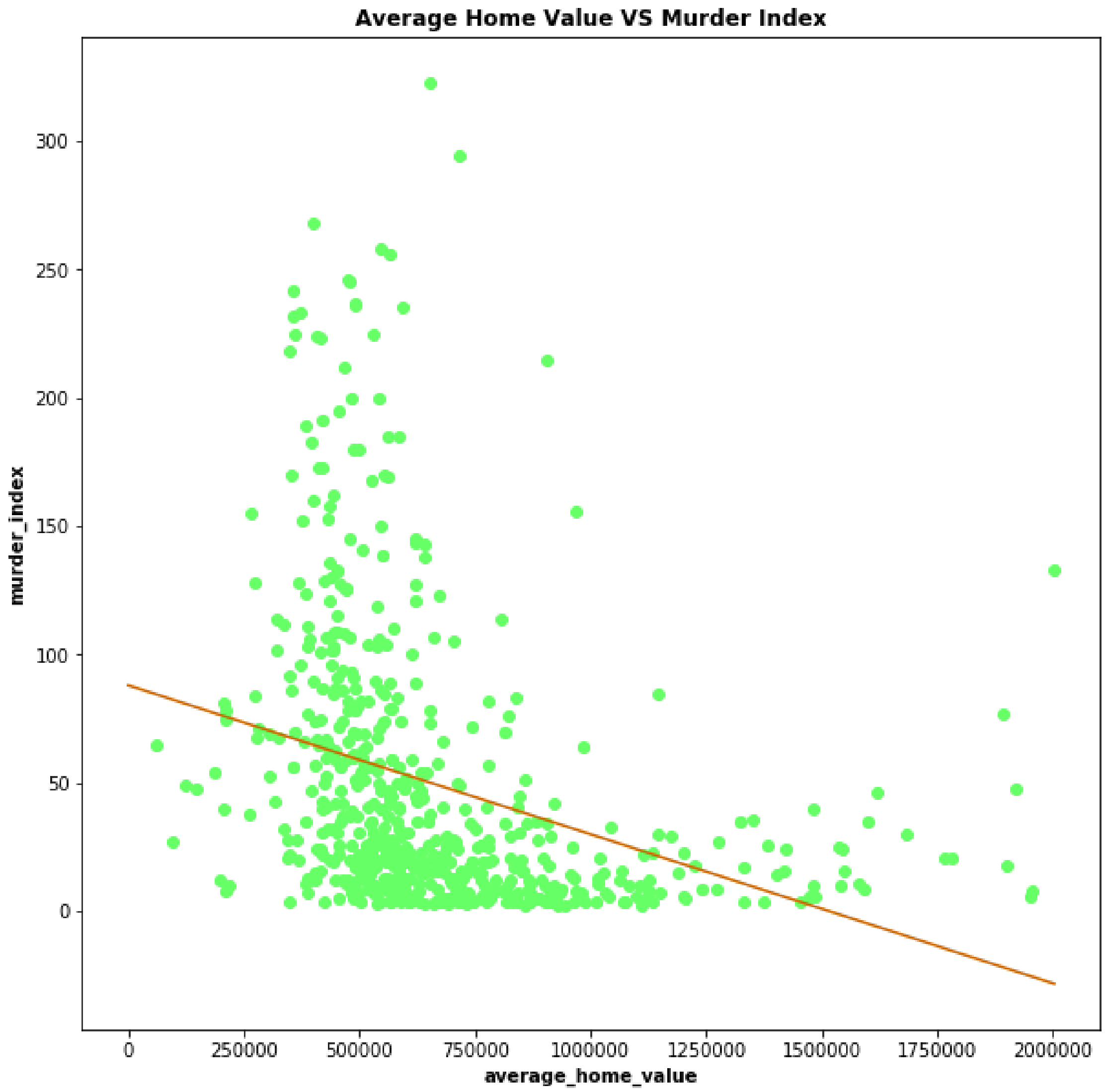


13 Visualizing the correlation between average home value and murder index, with statistic summary, such as r-square, p-value, regression line equation

[159]:
```
# making scatter plots for visualization
y4 = home_value_murder.get('murder_index')
x4 = home_value_murder.get('average_home_value')

# Give title for graph; set the color for each dots
plt.scatter(x4, y4, color = '#66ff66')
plt.title('Average Home Value VS Murder Index', fontweight='bold')
```

```
# Add axis label
plt.xlabel('average_home_value', fontweight='bold')
plt.ylabel('murder_index', fontweight='bold')

# produce regression (best fit line) line
model_4 = np.polyfit(x4, y4, 1)
predict_4 = np.poly1d(model_4)

# calculate r-squared and r value
r_matrix_4 = np.corrcoef(x4, y4)
r_4 = r_matrix_4[0,1]
r2_4 = r_4 ** 2

# calculate p-value
from scipy import stats
pvalue_4 = stats.pearsonr(x4, y4)[1]

# visualize the regression line
x_lin_reg_4 = range(0, int(x4.max()))
y_lin_reg_4 = predict_4(x_lin_reg_4)
plt.plot(x_lin_reg_4, y_lin_reg_4, c = '#cc6600')
plt.gcf().set_size_inches((10, 10))

# print out the equation for regression line and corresponding r-squared values
print('Statistic Summary:')
print("y " + "= " + str(model_4[0]) + 'x ' + '+ ' + str(model_4[1]))
print('R value is ' + str(r_4))
print('R-squared value is ' + str(r2_4))
print('P value is ' + str(pvalue_4))
```

```
Statistic Summary:
y = -5.8025420385750564e-05x + 88.03252401379632
R value is -0.3099800439867244
R-squared value is 0.0960876276700116
P value is 6.767381145239242e-15
```



14 Table for average home value verses robbery index in San Diego county

[160]:
```
home_value_robbery_raw = pd.read_csv('home_value_vs_robbery_index.csv')
home_value_robbery_raw

# cleaning data
home_value_robbery = home_value_robbery_raw.get(['FIPS', 'AVGVAL_CY',␣
↪'CRMCYROBB'])
home_value_robbery
```

```
# rename column name
home_value_robbery = (home_value_robbery.assign(average_home_value =␣
↪home_value_robbery.get('AVGVAL_CY'),
                                       robbery_index = home_value_robbery.
↪get('CRMCYROBB'),
                                       San_Diego_census_tract_number =␣
↪home_value_robbery.get('FIPS'))
                )
# cleaning data
home_value_robbery = home_value_robbery.drop(columns = ['AVGVAL_CY',␣
↪'CRMCYROBB', 'FIPS'])
home_value_robbery = home_value_robbery.
↪set_index('San_Diego_census_tract_number')
home_value_robbery

# delete meaningless data
home_value_robbery = (home_value_robbery[(home_value_robbery.
↪get('average_home_value') > 0) &
                               (home_value_robbery.get('robbery_index') > 0)]
                )
home_value_robbery

# classify the data based on the robbery index, and labeling
def classification_robbery (values):
    if values >= 107:
        return 'High Robbery Index'
    elif (values < 107) & (values >= 52):
        return 'Medium Robbery Index'
    else:
        return 'Low Robbery Index'

# Adding labels to a new column
home_value_robbery = home_value_robbery.assign(robbery_index_level =␣
↪home_value_robbery.get('robbery_index'))
home_value_robbery = (home_value_robbery.assign(robbery_index_level =␣
↪home_value_robbery.get('robbery_index')
                                                                          ␣
↪          .apply(classification_robbery))
                      )
home_value_robbery
```

[160]:

```
                               average_home_value  robbery_index  \
San_Diego_census_tract_number
6073013601                                679505.0           19.0
6073017808                               1063944.0           19.0
```

```
6073007702                                   908984.0          49.0
6073008358                                   516170.0          29.0
6073018511                                   533067.0          19.0
…                                                   …             …
6073017104                                   930238.0           5.0
6073003303                                   585629.0         203.0
6073020710                                   920481.0          14.0
6073020809                                   580524.0         101.0
6073017035                                   583570.0          40.0

                              robbery_index_level
San_Diego_census_tract_number
6073013601                      Low Robbery Index
6073017808                      Low Robbery Index
6073007702                      Low Robbery Index
6073008358                      Low Robbery Index
6073018511                      Low Robbery Index
…                                               …
6073017104                      Low Robbery Index
6073003303                     High Robbery Index
6073020710                      Low Robbery Index
6073020809                   Medium Robbery Index
6073017035                      Low Robbery Index

[535 rows x 3 columns]
```

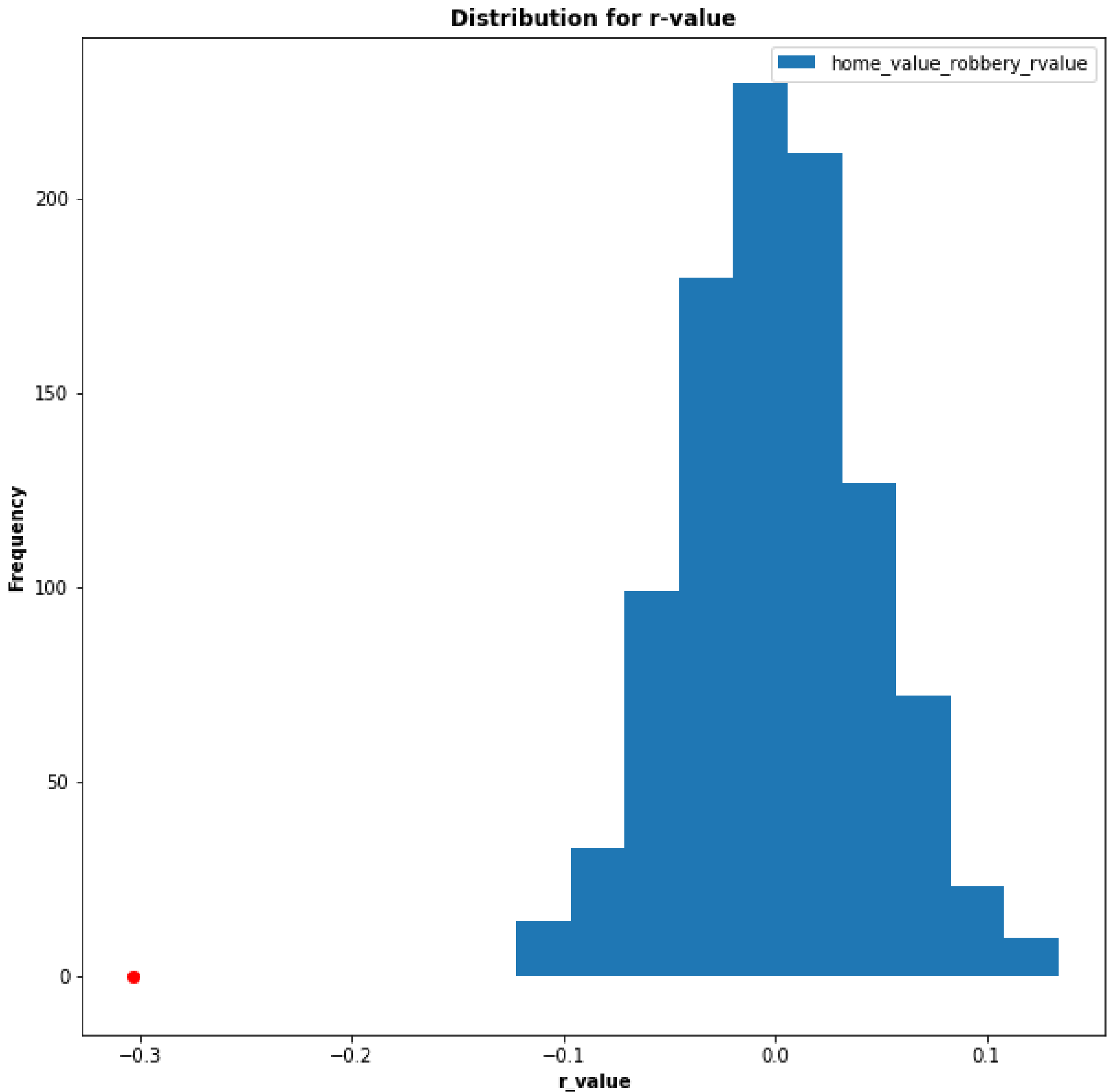
15 A | B testing; confidence interval

[161]:

```
# import
from scipy import stats

home_value_robbery

# shuffling columns "robbery index", and calculate the r-value for each round
possible_correlation_5 = np.array([])

for i in np.arange(1000):
    shuffling_5 = np.random.permutation(home_value_robbery.get('robbery_index'))
    home_value_robbery_shuffle = home_value_robbery.
 ↪assign(home_value_robbery_shuffling = shuffling_5)
    rvalue_5ab = (stats.pearsonr(home_value_robbery_shuffle.
 ↪get('average_home_value'),
                                 home_value_robbery_shuffle.
 ↪get('home_value_robbery_shuffling'))[0]
                 )
```

```
    possible_correlation_5 = np.append(possible_correlation_5, rvalue_5ab)

# visualize the distribution for r-values of each round, and plot the red dot␣
 ↪for the real r-value
y5 = home_value_robbery.get('robbery_index')
x5 = home_value_robbery.get('average_home_value')
pd.DataFrame().assign(home_value_robbery_rvalue = possible_correlation_5).
 ↪plot(kind = 'hist')
plt.scatter(stats.pearsonr(x5, y5)[0], 0 , color = 'red')
plt.title('Distribution for r-value', fontweight='bold')
plt.gcf().set_size_inches((10, 10))
plt.xlabel('r_value', fontweight='bold')
plt.ylabel('Frequency', fontweight='bold')
# From the graph, we can see the red dot is far away from the main distribution.
 ↪
# It tell us there are correlation between two variable, and it is not due to␣
 ↪random chance.
```

[161]: Text(0, 0.5, 'Frequency')

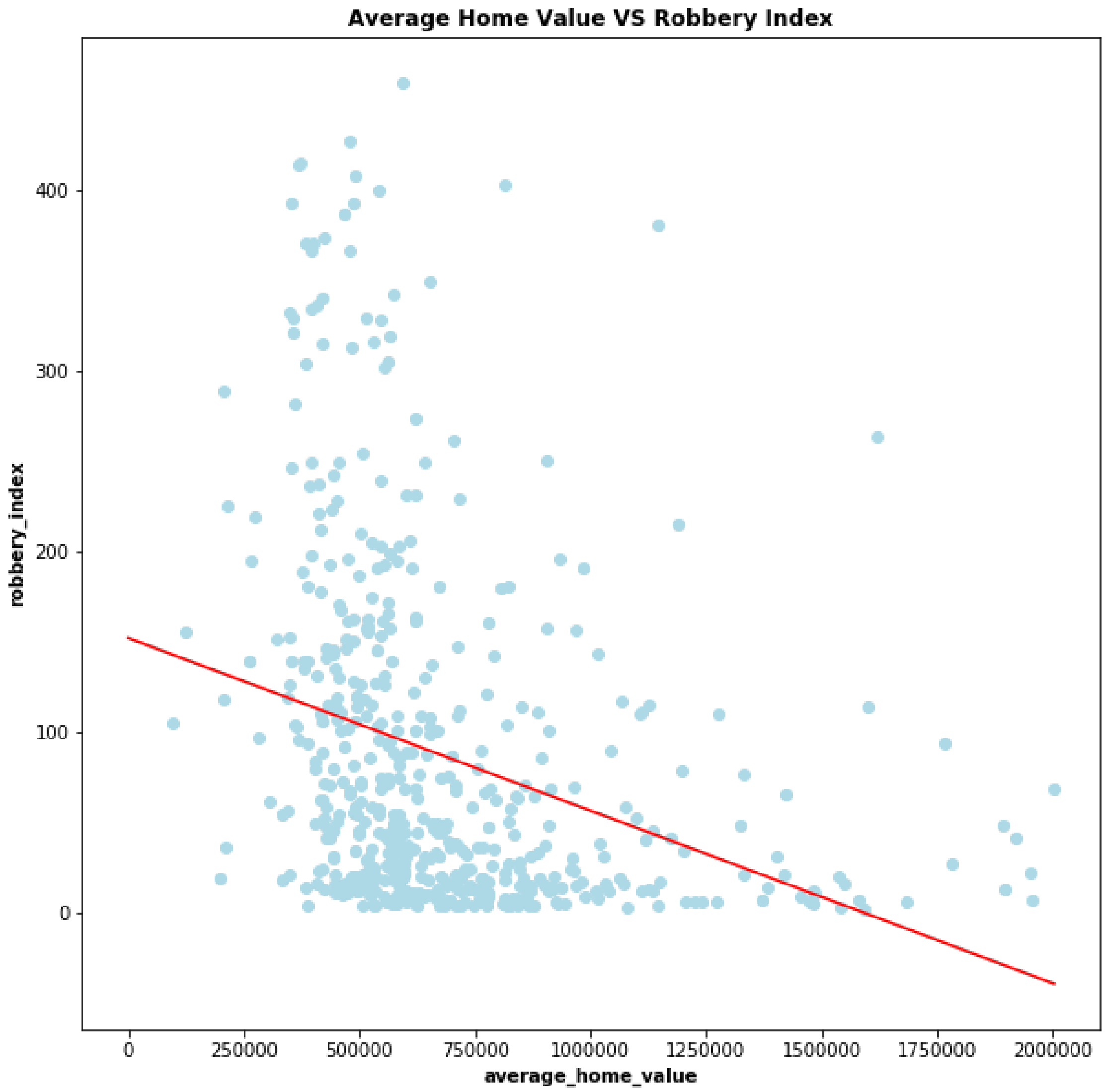


16 Visualizing the correlation between average home value and robbery index, with statistic summary, such as r-square, p-value, regression line equation

[162]:
```
# making scatter plots for visualization
y5 = home_value_robbery.get('robbery_index')
x5 = home_value_robbery.get('average_home_value')

# Give title for graph; set the color for each dots
plt.scatter(x5, y5, color = '#ADD8E6')
plt.title('Average Home Value VS Robbery Index', fontweight='bold')
```

```python
# Add axis label
plt.xlabel('average_home_value', fontweight='bold')
plt.ylabel('robbery_index', fontweight='bold')

# produce regression (best fit line) line
model_5 = np.polyfit(x5, y5, 1)
predict_5 = np.poly1d(model_5)

# calculate r-squared and r value
r_matrix_5 = np.corrcoef(x5, y5)
r_5 = r_matrix_5[0,1]
r2_5 = r_5 ** 2

# calculate p-value
from scipy import stats
pvalue_5 = stats.pearsonr(x5, y5)[1]

# visualize the regression line
x_lin_reg_5 = range(0, int(x5.max()))
y_lin_reg_5 = predict_5(x_lin_reg_5)
plt.plot(x_lin_reg_5, y_lin_reg_5, c = 'red')
plt.gcf().set_size_inches((10, 10))

# print out the equation for regression line and corresponding r-squared values
print('Statistic Summary:')
print("y " + "= " + str(model_5[0]) + 'x ' + '+ ' + str(model_5[1]))
print('R value is ' + str(r_5))
print('R-squared value is ' + str(r2_5))
print('P value is ' + str(pvalue_5))
```

```
Statistic Summary:
y = -9.553909355876393e-05x + 152.219030841975
R value is -0.3027584894515207
R-squared value is 0.09166270293496656
P value is 8.36445942891716e-13
```



17 Table for average home value verses rape index in San Diego county

[163]:
```
home_value_rape_raw = pd.read_csv('home_value_vs_rape_index.csv')
home_value_rape_raw

# cleaning data
home_value_rape = home_value_rape_raw.get(['FIPS', 'AVGVAL_CY', 'CRMCYRAPE'])
home_value_rape

# rename column name
```

```
home_value_rape = (home_value_rape.assign(average_home_value = home_value_rape.
 ↪get('AVGVAL_CY'),
                                      rape_index = home_value_rape.
 ↪get('CRMCYRAPE'),
                                      San_Diego_census_tract_number =␣
 ↪home_value_rape.get('FIPS'))
                )
# cleaning data
home_value_rape = home_value_rape.drop(columns = ['AVGVAL_CY', 'CRMCYRAPE',␣
 ↪'FIPS'])
home_value_rape = home_value_rape.set_index('San_Diego_census_tract_number')
home_value_rape

# delete meaningless data
home_value_rape = (home_value_rape[(home_value_rape.get('average_home_value') >␣
 ↪0) &
                               (home_value_rape.get('rape_index') > 0)]
                )
home_value_rape

# classify the data based on the rape index, and labeling
def classification_rape (values):
    if values >= 131:
        return 'High Rape Index'
    elif (values < 131) & (values >= 86):
        return 'Medium Rape Index'
    else:
        return 'Low Rape Index'

# Adding labels to a new column
home_value_rape = home_value_rape.assign(rape_index_level = home_value_rape.
 ↪get('rape_index'))
home_value_rape = (home_value_rape.assign(rape_index_level = home_value_rape.
 ↪get('rape_index')
                          .apply(classification_rape))
                      )
home_value_rape
```

[163]:

```
                               average_home_value  rape_index  \
San_Diego_census_tract_number
6073008102                              1781890.0        15.0
6073008301                              1241279.0       126.0
6073008311                              1955461.0        42.0
6073008503                               724589.0        44.0
6073008333                              1202596.0       108.0
…                                             …           …
```

```
6073020809                              580524.0        36.0
6073020902                              549861.0        22.0
6073020904                              619687.0        79.0
6073020903                              437608.0        61.0
6073021000                              280280.0        27.0

                                rape_index_level
San_Diego_census_tract_number
6073008102                        Low Rape Index
6073008301                     Medium Rape Index
6073008311                        Low Rape Index
6073008503                        Low Rape Index
6073008333                     Medium Rape Index
…                                            …
6073020809                        Low Rape Index
6073020902                        Low Rape Index
6073020904                        Low Rape Index
6073020903                        Low Rape Index
6073021000                        Low Rape Index

[306 rows x 3 columns]
```

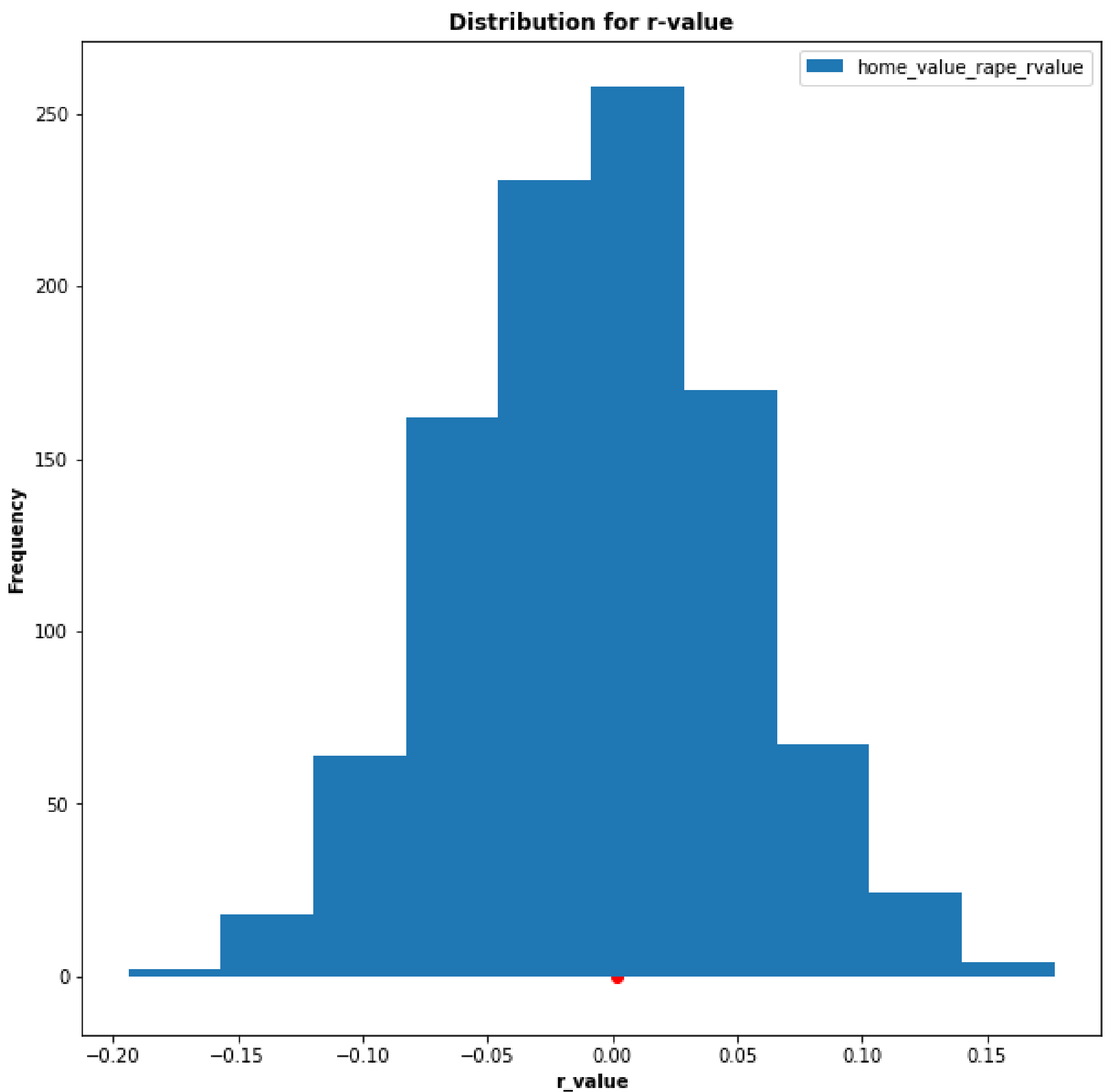
18 A | B testing; confidence interval

[164]:

```
# import
from scipy import stats

home_value_rape

# shuffling columns "rape index", and calculate the r-value for each round
possible_correlation_6 = np.array([])

for i in np.arange(1000):
    shuffling_6 = np.random.permutation(home_value_rape.get('rape_index'))
    home_value_rape_shuffle = home_value_rape.assign(home_value_rape_shuffling␣
↪= shuffling_6)
    rvalue_6ab = (stats.pearsonr(home_value_rape_shuffle.
↪get('average_home_value'),
                                home_value_rape_shuffle.
↪get('home_value_rape_shuffling'))[0]
                  )
    possible_correlation_6 = np.append(possible_correlation_6, rvalue_6ab)

# visualize the distribution for r-values of each round, and plot the red dot␣
↪for the real r-value
```

```
y6 = home_value_rape.get('rape_index')
x6 = home_value_rape.get('average_home_value')
pd.DataFrame().assign(home_value_rape_rvalue = possible_correlation_6).
 ↪plot(kind = 'hist')
plt.scatter(stats.pearsonr(x6, y6)[0], 0 , color = 'red')
plt.title('Distribution for r-value', fontweight='bold')
plt.gcf().set_size_inches((10, 10))
plt.xlabel('r_value', fontweight='bold')
plt.ylabel('Frequency', fontweight='bold')
# From the graph, we can see the red dot is inside the main distribution.
# It tell us there are no obvious correlation between two variable, or the␣
 ↪correlation may be due to random chance.
```

[164]: Text(0, 0.5, 'Frequency')



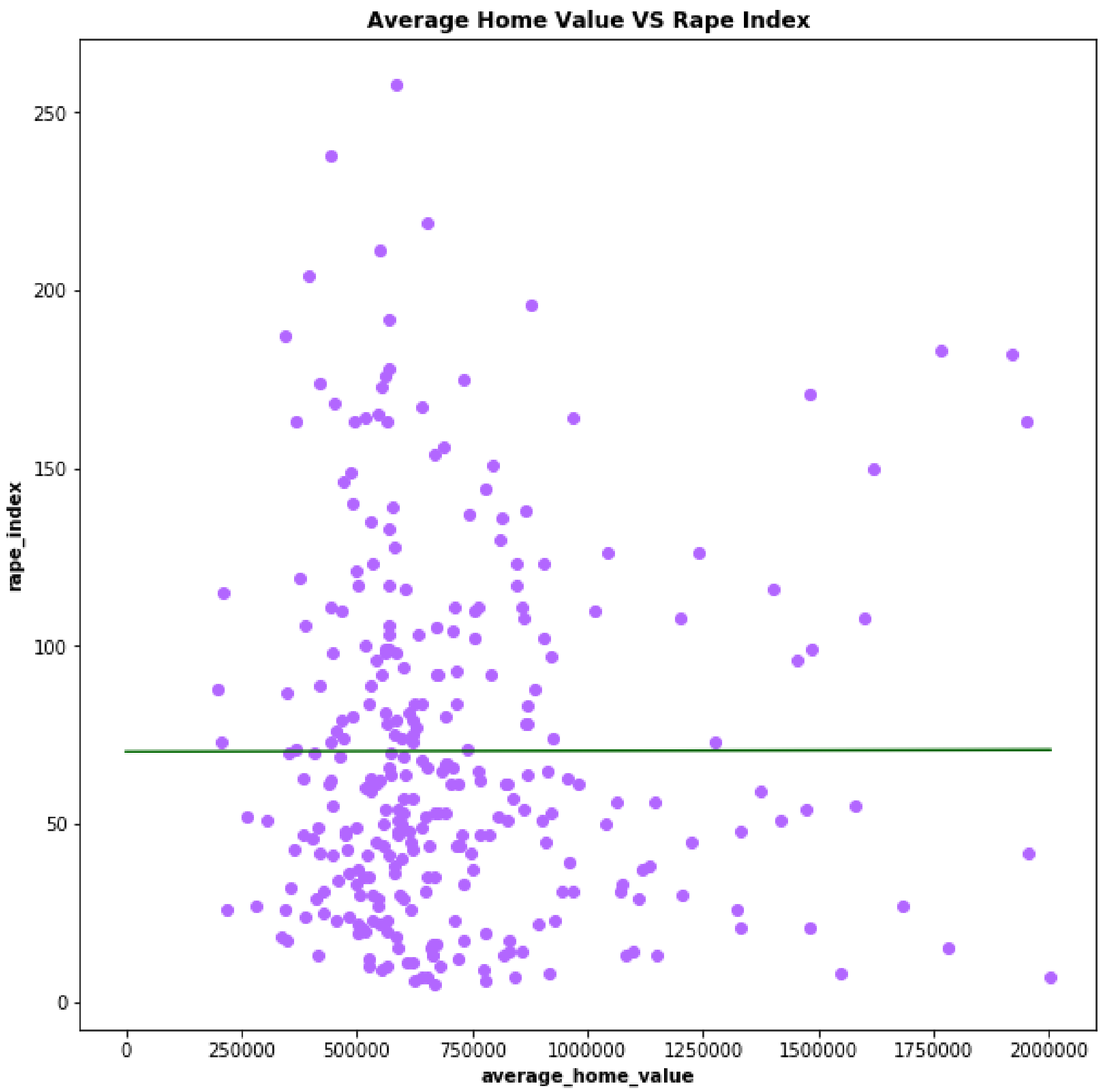
19 Visualizing the correlation between average home value and rape index, with statistic summary, such as r-square, p-value, regression line equation

[165]:

```
# making scatter plots for visualization
y6 = home_value_rape.get('rape_index')
x6 = home_value_rape.get('average_home_value')

# Give title for graph; set the color for each dots
plt.scatter(x6, y6, color = '#b266ff')
plt.title('Average Home Value VS Rape Index', fontweight='bold')

# Add axis label
plt.xlabel('average_home_value', fontweight='bold')
plt.ylabel('rape_index', fontweight='bold')

#produce regression (best fit line) line
model_6 = np.polyfit(x6, y6, 1)
predict_6 = np.poly1d(model_6)

# calculate r-squared and r value
r_matrix_6 = np.corrcoef(x6, y6)
r_6 = r_matrix_6[0,1]
r2_6 = r_6 ** 2

# calculate p-value
from scipy import stats
pvalue_6 = stats.pearsonr(x6, y6)[1]

# visualize the regression line
x_lin_reg_6 = range(0, int(x6.max()))
y_lin_reg_6 = predict_6(x_lin_reg_6)
plt.plot(x_lin_reg_6, y_lin_reg_6, c = '#006600')
plt.gcf().set_size_inches((10, 10))

# print out the equation for regression line and corresponding r-squared values
print('Statistic Summary:')
print("y " + "= " + str(model_6[0]) + 'x ' + '+ ' + str(model_6[1]))
print('R value is ' + str(r_6))
print('R-squared value is ' + str(r2_6))
print('P value is ' + str(pvalue_6))
```

```
Statistic Summary:
```

```
y = 2.6522074720393594e-07x + 70.27528337611223
R value is 0.0017062386211373747
R-squared value is 2.9112502322607695e-06
P value is 0.9762864902433023
```



20 Machine learning: Cross validation

[166]:
```python
# import
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.model_selection import cross_val_score
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# We will use three different models to test the accuracy for our six different␣
↪combinations

# 1 LogisticRegression Model
model_lr = LogisticRegression(solver='lbfgs', multi_class='auto')

# 2 Support Vector Machine Model
svm_1 = SVC(gamma='scale')

# 3 Random Forest Classifier Model
rf_1 = RandomForestClassifier(n_estimators = 40)

# KFold testing (Enable us to perform five repeated test for three models)
from sklearn.model_selection import KFold
kf = KFold(n_splits = 5)
kf

# calculate score for each model
def score_1(model, X_train, X_test, y_train, y_test):
    model.fit(X_train, y_train)
    return model.score(X_test, y_test)

# We will use models above to test six combinations
```

21 First testing combination: "independent variable: average household income; dependent variable: murder index"

[167]:
```
# table for average household income verses murder index in Denver county
denver_testing = pd.read_csv('Denver_testing.csv')
denver_testing

# cleaning data
cross_income_murder = denver_testing.get(['FIPS', 'AVGHINC_CY', 'CRMCYMURD'])
cross_income_murder


# rename column name
cross_income_murder = (cross_income_murder.assign(average_household_income =␣
↪cross_income_murder.get('AVGHINC_CY'),
```

```
                                           murder_index = cross_income_murder.
 ↪get('CRMCYMURD'),
                                           Denver_census_tract_number =␣
 ↪cross_income_murder.get('FIPS'))
                 )
# cleaning data
cross_income_murder = cross_income_murder.drop(columns = ['AVGHINC_CY',␣
 ↪'CRMCYMURD', 'FIPS'])
cross_income_murder = cross_income_murder.
 ↪set_index('Denver_census_tract_number')
cross_income_murder

# delete meaningless data
cross_income_murder = (cross_income_murder[(cross_income_murder.
 ↪get('average_household_income') > 0) &
                                (cross_income_murder.get('murder_index') > 0)]
                 )
cross_income_murder

# classify the data based on the murder index, and labeling
def classification (values):
    if values >= 126:
        return 'High Murder Index'
    elif (values < 126) & (values >= 53):
        return 'Medium Murder Index'
    else:
        return 'Low Murder Index'

# Adding labels to a new column
cross_income_murder = cross_income_murder.assign(murder_index_level =␣
 ↪cross_income_murder.get('murder_index'))
cross_income_murder = (cross_income_murder.assign(murder_index_level =␣
 ↪cross_income_murder.get('murder_index')
                                                                              ␣
 ↪          .apply(classification))
                        )
cross_income_murder
```

[167]:

```
                            average_household_income  murder_index  \
Denver_census_tract_number
8031004601                                   67921.0         146.0
8031004700                                   80855.0         120.0
8031004801                                   91124.0         129.0
8031011902                                   79204.0          32.0
8031011903                                   95060.0          10.0
…                                                 …             …
```

```
8031008389                                   105036.0           89.0
8031008390                                    74834.0           46.0
8031008391                                    91622.0           27.0
8031007037                                    61036.0          228.0
8031980000                                    72884.0           67.0

                             murder_index_level
Denver_census_tract_number
8031004601                     High Murder Index
8031004700                   Medium Murder Index
8031004801                     High Murder Index
8031011902                      Low Murder Index
8031011903                      Low Murder Index
…                                            …
8031008389                   Medium Murder Index
8031008390                      Low Murder Index
8031008391                      Low Murder Index
8031007037                     High Murder Index
8031980000                   Medium Murder Index

[143 rows x 3 columns]
```

22 Begin testing (Testing results please see the table in next couple lines)

[168]:
```python
# testing table
cross_income_murder

# assign datas from two variables to two arrays
data_1 = np.array([cross_income_murder.get('average_household_income')]).
 ↪transpose()
predicting_1 = np.array(cross_income_murder.get('murder_index_level'))

# splitting data into testing data and training data, and the ratio between␣
 ↪testing and training is 3 : 7
from sklearn.model_selection import train_test_split
X1_train, X1_test, y1_train, y1_test = (train_test_split(data_1,
                                                         predicting_1,
                                                         test_size = 0.3)
                                       )

# create arrays to collect score for 5 rounds (5 folds created by K-Fold method)
score_logistic_1 = np.array([])
score_svm_1 = np.array([])
```

```
score_rf_1 = np.array([])


# perform accuracy test by using three models for five rounds (5 folds), and␣
↪assigned score to corresponding arrays
for train_index_1, test_index_1 in kf.split(data_1):
    X1_train, X1_test, y1_train, y1_test = (data_1[train_index_1],
                                             data_1[test_index_1],
                                             predicting_1[train_index_1],
                                             predicting_1[test_index_1]
                                            )

    score_logistic_1 = np.append(score_logistic_1, score_1(model_lr, X1_train,␣
↪X1_test, y1_train, y1_test))
    score_svm_1 = np.append(score_svm_1, score_1(svm_1, X1_train, X1_test,␣
↪y1_train, y1_test))
    score_rf_1 = np.append(score_rf_1, score_1(rf_1, X1_train, X1_test,␣
↪y1_train, y1_test))
```

23 Second testing combination: "independent variable: average household income; dependent variable: robbery index"

[169]:
```
# table for average household income verses robbery index in Denver county
denver_testing = pd.read_csv('Denver_testing.csv')
denver_testing

# cleaning data
cross_income_robbery = denver_testing.get(['FIPS', 'AVGHINC_CY', 'CRMCYROBB'])
cross_income_robbery


# rename column name
cross_income_robbery = (cross_income_robbery.assign(average_household_income =␣
↪cross_income_robbery.get('AVGHINC_CY'),
                                       robbery_index = cross_income_robbery.
↪get('CRMCYROBB'),
                                       Denver_census_tract_number =␣
↪cross_income_robbery.get('FIPS'))
                )
# cleaning data
cross_income_robbery = cross_income_robbery.drop(columns = ['AVGHINC_CY',␣
↪'CRMCYROBB', 'FIPS'])
cross_income_robbery = cross_income_robbery.
↪set_index('Denver_census_tract_number')
```

```
cross_income_robbery

# delete meaningless data
cross_income_robbery = (cross_income_robbery[(cross_income_robbery.
↪get('average_household_income') > 0) &
                                             (cross_income_robbery.get('robbery_index') > 0)]
                )

# classify the data based on the robbery index, and labeling
def classification_robbery (values):
    if values >= 107:
        return 'High Robbery Index'
    elif (values < 107) & (values >= 52):
        return 'Medium Robbery Index'
    else:
        return 'Low Robbery Index'

# Adding labels to a new column
cross_income_robbery = cross_income_robbery.assign(robbery_index_level =␣
↪cross_income_robbery.get('robbery_index'))
cross_income_robbery = (cross_income_robbery.assign(robbery_index_level =␣
↪cross_income_robbery.get('robbery_index')
                                                                                        ␣
↪           .apply(classification_robbery))
                  )
cross_income_robbery
```

[169]:

```
                            average_household_income  robbery_index  \
Denver_census_tract_number
8031004601                                   67921.0          121.0
8031004700                                   80855.0          107.0
8031004801                                   91124.0           94.0
8031011902                                   79204.0           20.0
8031011903                                   95060.0           28.0
…                                                  …              …
8031008389                                  105036.0          183.0
8031008390                                   74834.0           69.0
8031008391                                   91622.0           30.0
8031007037                                   61036.0          175.0
8031980000                                   72884.0          162.0

                             robbery_index_level
Denver_census_tract_number
8031004601                    High Robbery Index
8031004700                    High Robbery Index
8031004801                  Medium Robbery Index
8031011902                     Low Robbery Index
```

```
8031011903                         Low Robbery Index
…                                                 …
8031008389                        High Robbery Index
8031008390                      Medium Robbery Index
8031008391                         Low Robbery Index
8031007037                        High Robbery Index
8031980000                        High Robbery Index

[143 rows x 3 columns]
```

24 Begin testing (Testing results please see the table in next couple lines)

[170]:
```
# testing table
cross_income_rape

# assign datas from two variables to two arrays
data_3 = np.array([cross_income_rape.get('average_household_income')]).
 ↪transpose()
predicting_3 = np.array(cross_income_rape.get('rape_index_level'))

# splitting data into testing data and training data, and the ratio between␣
 ↪testing and training is 3 : 7
from sklearn.model_selection import train_test_split
X3_train, X3_test, y3_train, y3_test = (train_test_split(data_3,
                                                         predicting_3,
                                                         test_size = 0.3)
                                        )

# create arrays to collect score for 5 rounds (5 folds created by K-Fold method)
score_logistic_3 = np.array([])
score_svm_3 = np.array([])
score_rf_3 = np.array([])


# perform accuracy test by using three models for five rounds (5 folds), and␣
 ↪assigned score to corresponding arrays
for train_index_3, test_index_3 in kf.split(data_3):
    X3_train, X3_test, y3_train, y3_test = (data_3[train_index_3],
                                            data_3[test_index_3],
                                            predicting_3[train_index_3],
                                            predicting_3[test_index_3]
                                           )
```

```
    score_logistic_3 = np.append(score_logistic_3, score_1(model_lr, X3_train,␣
↪X3_test, y3_train, y3_test))
    score_svm_3 = np.append(score_svm_3, score_1(svm_1, X3_train, X3_test,␣
↪y3_train, y3_test))
    score_rf_3 = np.append(score_rf_3, score_1(rf_1, X3_train, X3_test,␣
↪y3_train, y3_test))
```

25 Fourth testing combination: "independent variable: average home value; dependent variable: murder index"

[171]:
```
# table for average home value verses murder index in Denver county
denver_testing = pd.read_csv('Denver_testing.csv')
denver_testing

# cleaning data
cross_homevalue_murder = denver_testing.get(['FIPS', 'AVGVAL_CY', 'CRMCYMURD'])
cross_homevalue_murder



# rename column name
cross_homevalue_murder = (cross_homevalue_murder.assign(average_home_value =␣
↪cross_homevalue_murder.get('AVGVAL_CY'),
                                       murder_index = cross_homevalue_murder.
↪get('CRMCYMURD'),
                                       Denver_census_tract_number =␣
↪cross_homevalue_murder.get('FIPS'))
                )
# cleaning data
cross_homevalue_murder = cross_homevalue_murder.drop(columns = ['AVGVAL_CY',␣
↪'CRMCYMURD', 'FIPS'])
cross_homevalue_murder = cross_homevalue_murder.
↪set_index('Denver_census_tract_number')
cross_homevalue_murder

# delete meaningless data
cross_homevalue_murder = (cross_homevalue_murder[(cross_homevalue_murder.
↪get('average_home_value') > 0) &
                               (cross_homevalue_murder.get('murder_index') > 0)]
                )
cross_homevalue_murder

# classify the data based on the murder index, and labeling
def classification (values):
    if values >= 126:
```

```
        return 'High Murder Index'
    elif (values < 126) & (values >= 53):
        return 'Medium Murder Index'
    else:
        return 'Low Murder Index'

# Adding labels to a new column
cross_homevalue_murder = cross_homevalue_murder.assign(murder_index_level =␣
↪cross_homevalue_murder.get('murder_index'))
cross_homevalue_murder = (cross_homevalue_murder.assign(murder_index_level =␣
↪cross_homevalue_murder.get('murder_index')
                                                                                ␣
↪            .apply(classification))
                     )
cross_homevalue_murder
```

[171]:

```
                            average_home_value  murder_index  \
Denver_census_tract_number
8031004601                            228476.0         146.0
8031004700                            471890.0         120.0
8031004801                            430115.0         129.0
8031011902                            458520.0          32.0
8031011903                            375076.0          10.0
…                                          …            …
8031008389                            416895.0          89.0
8031008390                            332248.0          46.0
8031008391                            335379.0          27.0
8031007037                            394819.0         228.0
8031980000                            185745.0          67.0

                             murder_index_level
Denver_census_tract_number
8031004601                    High Murder Index
8031004700                  Medium Murder Index
8031004801                    High Murder Index
8031011902                     Low Murder Index
8031011903                     Low Murder Index
…                                           …
8031008389                  Medium Murder Index
8031008390                     Low Murder Index
8031008391                     Low Murder Index
8031007037                    High Murder Index
8031980000                  Medium Murder Index

[143 rows x 3 columns]
```

26 Begin testing (Testing results please see the table in next couple lines)

[172]:
```
# testing table
cross_homevalue_murder

# assign datas from two variables to two arrays
data_4 = np.array([cross_homevalue_murder.get('average_home_value')]).
 ↪transpose()
predicting_4 = np.array(cross_homevalue_murder.get('murder_index_level'))

# splitting data into testing data and training data, and the ratio between␣
 ↪testing and training is 3 : 7
from sklearn.model_selection import train_test_split
X4_train, X4_test, y4_train, y4_test = (train_test_split(data_4,
                                                         predicting_4,
                                                         test_size = 0.3)
                                       )

# create arrays to collect score for 5 rounds (5 folds created by K-Fold method)
score_logistic_4 = np.array([])
score_svm_4 = np.array([])
score_rf_4 = np.array([])


# perform accuracy test by using three models for five rounds (5 folds), and␣
 ↪assigned score to corresponding arrays
for train_index_4, test_index_4 in kf.split(data_4):
    X4_train, X4_test, y4_train, y4_test = (data_4[train_index_4],
                                            data_4[test_index_4],
                                            predicting_4[train_index_4],
                                            predicting_4[test_index_4]
                                           )

    score_logistic_4 = np.append(score_logistic_4, score_1(model_lr, X4_train,␣
 ↪X4_test, y4_train, y4_test))
    score_svm_4 = np.append(score_svm_4, score_1(svm_1, X4_train, X4_test,␣
 ↪y4_train, y4_test))
    score_rf_4 = np.append(score_rf_4, score_1(rf_1, X4_train, X4_test,␣
 ↪y4_train, y4_test))
```

27 Fifth testing combination: "independent variable: average home value; dependent variable: robbery index"

[173]:
```
# table for average home value verses robbery index in Denver county
denver_testing = pd.read_csv('Denver_testing.csv')
denver_testing

# cleaning data
cross_homevalue_robbery = denver_testing.get(['FIPS', 'AVGVAL_CY', 'CRMCYROBB'])
cross_homevalue_robbery


# rename column name
cross_homevalue_robbery = (cross_homevalue_robbery.assign(average_home_value =␣
 ↪cross_homevalue_robbery.get('AVGVAL_CY'),
                                       robbery_index = cross_homevalue_robbery.
 ↪get('CRMCYROBB'),
                                       Denver_census_tract_number =␣
 ↪cross_homevalue_robbery.get('FIPS'))
                )
# cleaning data
cross_homevalue_robbery = cross_homevalue_robbery.drop(columns = ['AVGVAL_CY',␣
 ↪'CRMCYROBB', 'FIPS'])
cross_homevalue_robbery = cross_homevalue_robbery.
 ↪set_index('Denver_census_tract_number')
cross_homevalue_robbery

# delete meaningless data
cross_homevalue_robbery = (cross_homevalue_robbery[(cross_homevalue_robbery.
 ↪get('average_home_value') > 0) &
                               (cross_homevalue_robbery.get('robbery_index') >␣
 ↪0)]
                )
cross_homevalue_robbery

# classify the data based on the robbery index, and labeling
def classification_robbery (values):
    if values >= 107:
        return 'High Robbery Index'
    elif (values < 107) & (values >= 52):
        return 'Medium Robbery Index'
    else:
        return 'Low Robbery Index'

# Adding labels to a new column
```

```
cross_homevalue_robbery = cross_homevalue_robbery.assign(robbery_index_level =␣
↪cross_homevalue_robbery.get('robbery_index'))
cross_homevalue_robbery = (cross_homevalue_robbery.assign(robbery_index_level =␣
↪cross_homevalue_robbery.get('robbery_index')
                                                                                ␣
↪          .apply(classification_robbery))
                  )
cross_homevalue_robbery
```

[173]:

```
                           average_home_value  robbery_index  \
Denver_census_tract_number
8031004601                           228476.0          121.0
8031004700                           471890.0          107.0
8031004801                           430115.0           94.0
8031011902                           458520.0           20.0
8031011903                           375076.0           28.0
…                                         …            …
8031008389                           416895.0          183.0
8031008390                           332248.0           69.0
8031008391                           335379.0           30.0
8031007037                           394819.0          175.0
8031980000                           185745.0          162.0

                              robbery_index_level
Denver_census_tract_number
8031004601                    High Robbery Index
8031004700                    High Robbery Index
8031004801                  Medium Robbery Index
8031011902                     Low Robbery Index
8031011903                     Low Robbery Index
…                                            …
8031008389                    High Robbery Index
8031008390                  Medium Robbery Index
8031008391                     Low Robbery Index
8031007037                    High Robbery Index
8031980000                    High Robbery Index

[143 rows x 3 columns]
```

28 Begin testing (Testing results please see the table in next couple lines)

[174]:
```
# testing table
cross_homevalue_robbery

# assign datas from two variables to two arrays
data_5 = np.array([cross_homevalue_robbery.get('average_home_value')]).
↪transpose()
predicting_5 = np.array(cross_homevalue_robbery.get('robbery_index_level'))

# splitting data into testing data and training data, and the ratio between␣
↪testing and training is 3 : 7
from sklearn.model_selection import train_test_split
X5_train, X5_test, y5_train, y5_test = (train_test_split(data_5,
                                                         predicting_5,
                                                         test_size = 0.3)
                                       )

# create arrays to collect score for 5 rounds (5 folds created by K-Fold method)
score_logistic_5 = np.array([])
score_svm_5 = np.array([])
score_rf_5 = np.array([])


# perform accuracy test by using three models for five rounds (5 folds), and␣
↪assigned score to corresponding arrays
for train_index_5, test_index_5 in kf.split(data_5):
    X5_train, X5_test, y5_train, y5_test = (data_5[train_index_5],
                                            data_5[test_index_5],
                                            predicting_5[train_index_5],
                                            predicting_5[test_index_5]
                                           )

    score_logistic_5 = np.append(score_logistic_5, score_1(model_lr, X5_train,␣
↪X5_test, y5_train, y5_test))
    score_svm_5 = np.append(score_svm_5, score_1(svm_1, X5_train, X5_test,␣
↪y5_train, y5_test))
    score_rf_5 = np.append(score_rf_5, score_1(rf_1, X5_train, X5_test,␣
↪y5_train, y5_test))
```

29 Sixth testing combination: "independent variable: average home value; dependent variable: rape index"

[175]:
```
# table for average home value verses rape index in Denver county
denver_testing = pd.read_csv('Denver_testing.csv')
denver_testing

# cleaning data
cross_homevalue_rape = denver_testing.get(['FIPS', 'AVGVAL_CY', 'CRMCYRAPE'])
cross_homevalue_rape



# rename column name
cross_homevalue_rape = (cross_homevalue_rape.assign(average_home_value =␣
↪cross_homevalue_rape.get('AVGVAL_CY'),
                                       rape_index = cross_homevalue_rape.
↪get('CRMCYRAPE'),
                                       Denver_census_tract_number =␣
↪cross_homevalue_rape.get('FIPS'))
                )
# cleaning data
cross_homevalue_rape = cross_homevalue_rape.drop(columns = ['AVGVAL_CY',␣
↪'CRMCYRAPE', 'FIPS'])
cross_homevalue_rape = cross_homevalue_rape.
↪set_index('Denver_census_tract_number')
cross_homevalue_rape

# delete meaningless data
cross_homevalue_rape = (cross_homevalue_rape[(cross_homevalue_rape.
↪get('average_home_value') > 0) &
                               (cross_homevalue_rape.get('rape_index') > 0)]
                )
cross_homevalue_rape

# classify the data based on the rape index, and labeling
def classification_rape (values):
    if values >= 131:
        return 'High Rape Index'
    elif (values < 131) & (values >= 86):
        return 'Medium Rape Index'
    else:
        return 'Low Rape Index'

# Adding labels to a new column
cross_homevalue_rape = cross_homevalue_rape.assign(rape_index_level =␣
↪cross_homevalue_rape.get('rape_index'))
```

```
cross_homevalue_rape = (cross_homevalue_rape.assign(rape_index_level =␣
↪cross_homevalue_rape.get('rape_index')
                                                                          ␣
↪           .apply(classification_rape))
                      )
cross_homevalue_rape
```

[175]:

```
                            average_home_value  rape_index   rape_index_level
Denver_census_tract_number
8031004601                            228476.0       246.0    High Rape Index
8031004700                            471890.0        40.0     Low Rape Index
8031004801                            430115.0       105.0  Medium Rape Index
8031011902                            458520.0       249.0    High Rape Index
8031011903                            375076.0       152.0    High Rape Index
…                                          …           …                  …
8031008389                            416895.0        63.0     Low Rape Index
8031008390                            332248.0        60.0     Low Rape Index
8031008391                            335379.0       124.0  Medium Rape Index
8031007037                            394819.0       567.0    High Rape Index
8031980000                            185745.0       721.0    High Rape Index

[143 rows x 3 columns]
```

30 Begin testing (Testing results please see the table in next couple lines)

[176]:

```
# testing table
cross_homevalue_rape

# assign datas from two variables to two arrays
data_6 = np.array([cross_homevalue_rape.get('average_home_value')]).transpose()
predicting_6 = np.array(cross_homevalue_rape.get('rape_index_level'))

# splitting data into testing data and training data, and the ratio between␣
↪testing and training is 3 : 7
from sklearn.model_selection import train_test_split
X6_train, X6_test, y6_train, y6_test = (train_test_split(data_6,
                                                         predicting_6,
                                                         test_size = 0.3)
                                       )

# create arrays to collect score for 5 rounds (5 folds created by K-Fold method)
score_logistic_6 = np.array([])
score_svm_6 = np.array([])
```

```
score_rf_6 = np.array([])


# perform accuracy test by using three models for five rounds (5 folds), and␣
↪assigned score to corresponding arrays
for train_index_6, test_index_6 in kf.split(data_6):
    X6_train, X6_test, y6_train, y6_test = (data_6[train_index_6],
                                            data_6[test_index_6],
                                            predicting_6[train_index_6],
                                            predicting_6[test_index_6]
                                           )

    score_logistic_6 = np.append(score_logistic_6, score_1(model_lr, X6_train,␣
↪X6_test, y6_train, y6_test))
    score_svm_6 = np.append(score_svm_6, score_1(svm_1, X6_train, X6_test,␣
↪y6_train, y6_test))
    score_rf_6 = np.append(score_rf_6, score_1(rf_1, X6_train, X6_test,␣
↪y6_train, y6_test))
```

31 Results for average accuracy score for six combinations under three models

[177]:
```
six_combinations = np.array(['avg_household_income vs murder_index',␣
↪'avg_household_income vs robbery_index',
                             'avg_household_income vs rape_index',␣
↪'avg_home_value vs murder_index',
                             'avg_home_value vs robbery_index', 'avg_home_value␣
↪vs rape_index'])

score_logistic = np.array([score_logistic_1.mean(), score_logistic_2.mean(),␣
↪score_logistic_3.mean(),
                           score_logistic_4.mean(), score_logistic_5.mean(),␣
↪score_logistic_6.mean()])

score_svm = np.array([score_svm_1.mean(), score_svm_2.mean(), score_svm_3.
↪mean(),
                           score_svm_4.mean(), score_svm_5.mean(), score_svm_6.
↪mean()])

score_rf = np.array([score_rf_1.mean(), score_rf_2.mean(), score_rf_3.mean(),
                           score_rf_4.mean(), score_rf_5.mean(), score_rf_6.
↪mean()])

result_table = pd.DataFrame().assign(Six_Combinations = six_combinations,
```

```
                                    Logistic_Regression_Model_Score =␣
↪score_logistic,
                                    SVM_Model_Score = score_svm,
                                    Random_Forest_Classifier_Model_Score =␣
↪score_rf)

result_table = result_table.set_index('Six_Combinations')
result_table
```

[177]:

```
                                       Logistic_Regression_Model_Score  \
Six_Combinations
avg_household_income vs murder_index                          0.330296
avg_household_income vs robbery_index                         0.685961
avg_household_income vs rape_index                            0.665025
avg_home_value vs murder_index                                0.330296
avg_home_value vs robbery_index                               0.685961
avg_home_value vs rape_index                                  0.665025

                                       SVM_Model_Score  \
Six_Combinations
avg_household_income vs murder_index          0.433744
avg_household_income vs robbery_index         0.665271
avg_household_income vs rape_index            0.650739
avg_home_value vs murder_index                0.371182
avg_home_value vs robbery_index               0.678818
avg_home_value vs rape_index                  0.672167

                                       Random_Forest_Classifier_Model_Score
Six_Combinations
avg_household_income vs murder_index                               0.426601
avg_household_income vs robbery_index                              0.588916
avg_household_income vs rape_index                                 0.461823
avg_home_value vs murder_index                                     0.392365
avg_home_value vs robbery_index                                    0.490640
avg_home_value vs rape_index                                       0.461823
```

32 Conclusion for cross validation:

33 Since the second combination 'average household income vs robbery index' receive relatively higher average accuracy scores under three testing models, we will use this combination to produce predictive analysis for average wealth in the communities vs crime index in another city: Chicago. Although, the average accuracy score for second combination is not high enough, we will still check the results of predictive analysis for Chicago, and compare our result with the real dataset.

34 Predictive analysis

35 Use regression line produced by second combination to predict robbery index level in Chicago

[178]:
```
# produce regression (best fit line) line based on the second combination from␣
↪previous testings
x7 = cross_income_robbery.get('average_household_income')
y7 = cross_income_robbery.get('robbery_index')
model_7 = np.polyfit(x7, y7, 1)
predict_7 = np.poly1d(model_7)

print('y ' + '= ' + str(model_7[0]) + 'x ' + '+ ' + str(model_7[1]))
```

```
y = -0.0009923755191330811x + 266.3858263691528
```

36 True data for average household income vs robbery index level in Chicago

[179]:
```
# Get the table for data of Chicago
chicago_income_robbery_raw = pd.read_csv('chicago_testing.csv')
chicago_income_robbery_raw

# Cleaning Data / Wranggling Data
chicago_income_robbery = chicago_income_robbery_raw.get(['FIPS', 'AVGHINC_CY',␣
↪'CRMCYROBB'])
chicago_income_robbery = (chicago_income_robbery.
↪assign(chicago_census_tract_number = chicago_income_robbery.get('FIPS'),
```

```
↪average_household_income = chicago_income_robbery.get('AVGHINC_CY'),
                                                  robbery_index =␣
↪chicago_income_robbery.get('CRMCYROBB'))
                         )
chicago_income_robbery = chicago_income_robbery.drop(columns = ['FIPS',␣
↪'AVGHINC_CY', 'CRMCYROBB'])
chicago_income_robbery = chicago_income_robbery.
↪set_index('chicago_census_tract_number')
chicago_income_robbery = (chicago_income_robbery[(chicago_income_robbery.
↪get('average_household_income') > 0) &
                                                 (chicago_income_robbery.
↪get('robbery_index') > 0)]
                         )

# classify the data based on the robbery index, and labeling
def classification_robbery (values):
    if values >= 107:
        return 'High Robbery Index'
    elif (values < 107) & (values >= 52):
        return 'Medium Robbery Index'
    else:
        return 'Low Robbery Index'

chicago_income_robbery = chicago_income_robbery.assign(robbery_index_level =␣
↪chicago_income_robbery.get('robbery_index'))
chicago_income_robbery = (chicago_income_robbery.assign(robbery_index_level =␣
↪chicago_income_robbery.get('robbery_index')
                                                                             ␣
↪          .apply(classification_robbery))
                      )
chicago_income_robbery
```

[179]:

```
                             average_household_income  robbery_index  \
chicago_census_tract_number
17031804403                                  108895.0            6.0
17031804404                                   95170.0           11.0
17031804405                                   65792.0           50.0
17031804406                                   91494.0           49.0
17031824113                                  132996.0            2.0
…                                                 …            …
17031520300                                   57693.0          200.0
17031520400                                   68327.0          237.0
17031520500                                   86616.0           87.0
17031520600                                   58505.0          148.0
17031833900                                   42602.0          693.0
```

```
                              robbery_index_level
chicago_census_tract_number
17031804403                     Low Robbery Index
17031804404                     Low Robbery Index
17031804405                     Low Robbery Index
17031804406                     Low Robbery Index
17031824113                     Low Robbery Index
…                                             …
17031520300                    High Robbery Index
17031520400                    High Robbery Index
17031520500                  Medium Robbery Index
17031520600                    High Robbery Index
17031833900                    High Robbery Index

[1315 rows x 3 columns]
```

37 Predicted data for average household income vs robbery index level in Chicago

[180]:
```
predicted_robbery_index = model_7[0] * (chicago_income_robbery.
 ↪get('average_household_income')) + model_7[1]
predicted_chicago_income_robbery = chicago_income_robbery.
 ↪assign(predicted_robber_index = predicted_robbery_index)

# classify the data based on the robbery index, and labeling
def classification_robbery (values):
    if values >= 107:
        return 'High Robbery Index'
    elif (values < 107) & (values >= 52):
        return 'Medium Robbery Index'
    else:
        return 'Low Robbery Index'

predicted_chicago_income_robbery = predicted_chicago_income_robbery.
 ↪assign(predicted_robbery_index_level = predicted_chicago_income_robbery.
 ↪get('predicted_robber_index'))
predicted_chicago_income_robbery = (predicted_chicago_income_robbery.
 ↪assign(predicted_robbery_index_level =  predicted_chicago_income_robbery.
 ↪get('predicted_robber_index')
                                                                      ␣
 ↪          .apply(classification_robbery))
                                  )
predicted_chicago_income_robbery
```

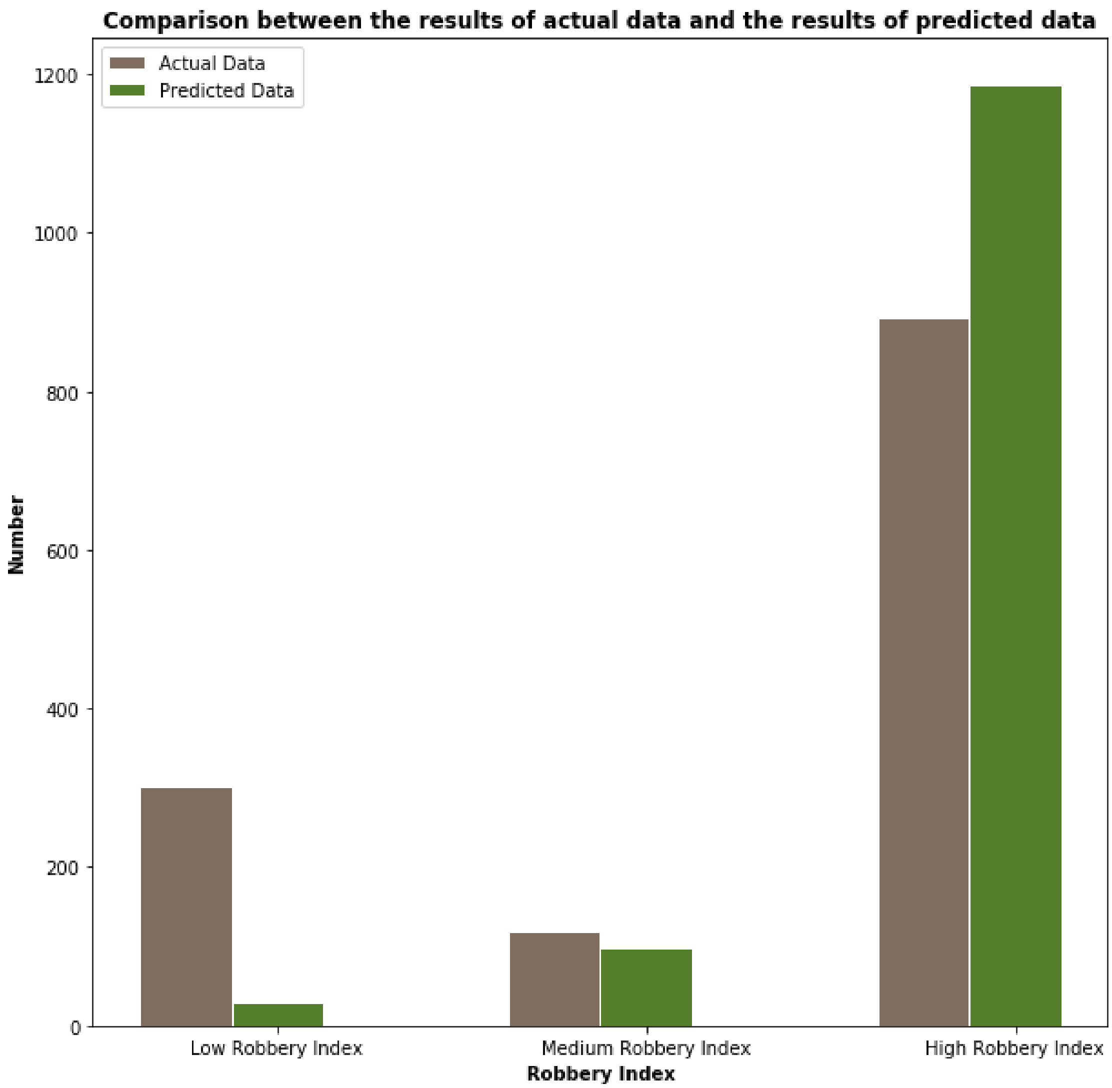
[180]:

```
                            average_household_income  robbery_index  \
chicago_census_tract_number
17031804403                                 108895.0            6.0
17031804404                                  95170.0           11.0
17031804405                                  65792.0           50.0
17031804406                                  91494.0           49.0
17031824113                                 132996.0            2.0
…                                                  …              …
17031520300                                  57693.0          200.0
17031520400                                  68327.0          237.0
17031520500                                  86616.0           87.0
17031520600                                  58505.0          148.0
17031833900                                  42602.0          693.0

                             robbery_index_level  predicted_robber_index  \
chicago_census_tract_number
17031804403                    Low Robbery Index              158.321094
17031804404                    Low Robbery Index              171.941448
17031804405                    Low Robbery Index              201.095456
17031804406                    Low Robbery Index              175.589421
17031824113                    Low Robbery Index              134.403852
…                                              …                       …
17031520300                   High Robbery Index              209.132706
17031520400                   High Robbery Index              198.579784
17031520500                 Medium Robbery Index              180.430228
17031520600                   High Robbery Index              208.326897
17031833900                   High Robbery Index              224.108645

                            predicted_robbery_index_level
chicago_census_tract_number
17031804403                            High Robbery Index
17031804404                            High Robbery Index
17031804405                            High Robbery Index
17031804406                            High Robbery Index
17031824113                            High Robbery Index
…                                                   …
17031520300                            High Robbery Index
17031520400                            High Robbery Index
17031520500                            High Robbery Index
17031520600                            High Robbery Index
17031833900                            High Robbery Index

[1315 rows x 5 columns]
```

38 visualizing the difference

[181]:
```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

x_l = (predicted_chicago_income_robbery
       [predicted_chicago_income_robbery.get('robbery_index_level') == 'Low␣
 ↪Robbery Index'].shape[0])
x_m = (predicted_chicago_income_robbery
       [predicted_chicago_income_robbery.get('robbery_index_level') == 'Medium␣
 ↪Robbery Index'].shape[0])
x_h = (predicted_chicago_income_robbery
       [predicted_chicago_income_robbery.get('robbery_index_level') == 'High␣
 ↪Robbery Index'].shape[0])

y_l = (predicted_chicago_income_robbery
       [predicted_chicago_income_robbery.get('predicted_robbery_index_level')␣
 ↪== 'Low Robbery Index'].shape[0])
y_m = (predicted_chicago_income_robbery
       [predicted_chicago_income_robbery.get('predicted_robbery_index_level')␣
 ↪== 'Medium Robbery Index'].shape[0])
y_h = (predicted_chicago_income_robbery
       [predicted_chicago_income_robbery.get('predicted_robbery_index_level')␣
 ↪== 'High Robbery Index'].shape[0])

# set width of bar
barWidth = 0.25

number_actual = [x_l, x_m, x_h]
number_predicted = [y_l, y_m, y_h]

# Set position of bar on X axis
r1 = np.arange(len(number_actual))
r2 = [x + barWidth for x in r1]

# Make the plot
plt.bar(r1, number_actual, color='#7f6d5f', width=barWidth, edgecolor='white',␣
 ↪label='Actual Data')
plt.bar(r2, number_predicted, color='#557f2d', width=barWidth,␣
 ↪edgecolor='white', label='Predicted Data')

# Add xticks on the middle of the group bars
plt.title('Comparison between the results of actual data and the results of␣
 ↪predicted data', fontweight='bold')
plt.xlabel('Robbery Index', fontweight='bold')
```

```
plt.ylabel('Number', fontweight='bold')
plt.xticks([r + barWidth for r in range(len(number_actual))], ['Low Robbery␣
↪Index', 'Medium Robbery Index', 'High Robbery Index'])

# Create legend & Show graphic
plt.legend()
plt.gcf().set_size_inches((10, 10))
plt.show()
```



39 Testing the accuracy of our regression line

[182]:
```
testing_results = np.array([])

for x in np.arange(predicted_chicago_income_robbery.shape[0]):
    if (predicted_chicago_income_robbery.get('robbery_index_level').iloc[x]
        == predicted_chicago_income_robbery.
 ↪get('predicted_robbery_index_level').iloc[x]
       ):
        testing_results = np.append(testing_results, 'True')
    else:
        testing_results = np.append(testing_results, 'False')

testing_results

accuracy_rate = np.count_nonzero(testing_results == 'True') / testing_results.
 ↪shape[0]
accuracy_rate
```

[182]: 0.6532319391634981

40 After comparing the results between robbery_index_level and predicted_robbery_index_level, we find that our regression line predicts correctly in about 65.3% of the time, which is closed to the average accuracy score for our second combination we calculated by cross validation.

41 Thank you for reading our project!!!!

42 For our Professor Fleischer and our TAs:

[183]:
```
heart = np.arange(-2, 2, 0.00001)
heart_1 = np.sqrt(1-(abs(heart)-1)**2)
heart_2 = np.arccos(1-abs(heart)) - np.pi
plt.plot(heart, heart_1, color = 'darkred')
plt.plot(heart, heart_2, color = 'darkred')
plt.fill(heart, heart_1, color = 'crimson')
plt.fill(heart, heart_2, color = 'crimson')
plt.subplots_adjust(0,0,1,1)
plt.show()
```


1
0
−1
−2
−3
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

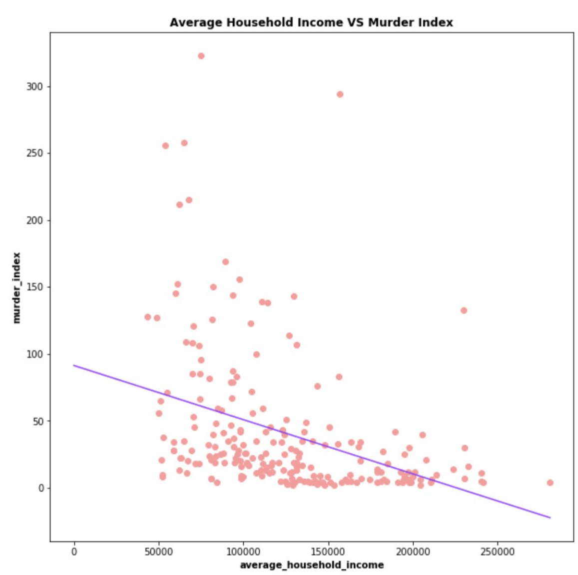
```
[ ]:
```

Discussion

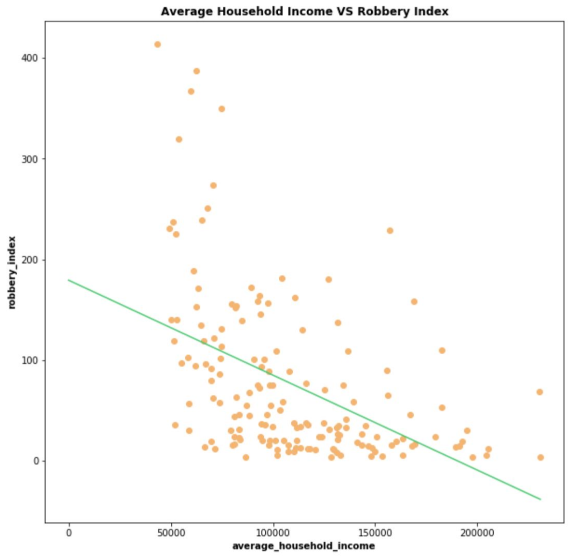
The outcome of our analysis will be multiple linear regression models, each representing correlation between one indicator of wealth and one type of crime index. We would find the Pearson Correlation factor “r,” Coefficient of determination“r squared,” and “p-value” to confirm whether our proposed analysis is accurate. We expected the “r” value to be much smaller than 0 but not equal to -1 because of the negative correlation between household income and crime index in our hypothesis. The “r squared” value shows the proportion of variants in our regression model, which need to be as small as possible. A “p-value” shorter than 5% can show the significance of our analysis. If any of those values are out of expectations, we would need to reproduce the procedure to check the data. (For more information, please see our coding PDF).

After performing the actual analysis, we have our results. The result of our first model (average household income vs. murder index) is y = -0.00404+91.286, r is -0.3715, r-square is 0.138, and the p-value is less than 0.01. The result of our second model (average household income vs. robbery index) is y = -0.00094+179.252, r is -0.467, r-square is 0.218, and the p-value is less than 0.01. The result of our third model (average household income vs. rape index) is y = -0.000185+95.82, r is -0.161, r-square is 0.0261, and the p-value is less than 0.01. The result of our fourth model (average home value vs murder index) is y = -5.803e-05x +88.033, r is -0.310, r-square is 0.096, and p-value is less than 0.01. The result of our fifth model (average home value vs robbery index) is y = -9.554e-05x + 152.219 r is -0.303, r-square is 0.0917, and p-value is less than 0.01. Above the five models, there are obvious negative correlations between the corresponding two variables because their r are all negative, and they are all statistically significant (p-value less than 0.05). (For more information, please see our coding PDF).

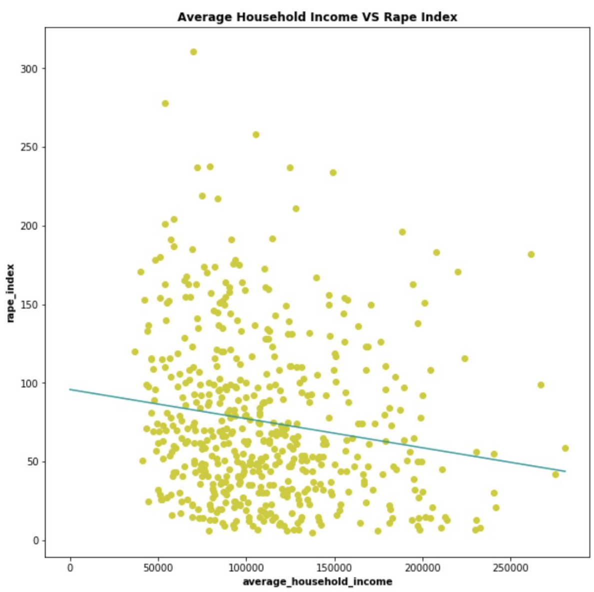
```
Statistic Summary:
y = -0.00040427945675277763x + 91.2861765029504
R value is -0.3714757697992417
R-squared value is 0.1379942475479392
P value is 1.4273055552517142e-08
```



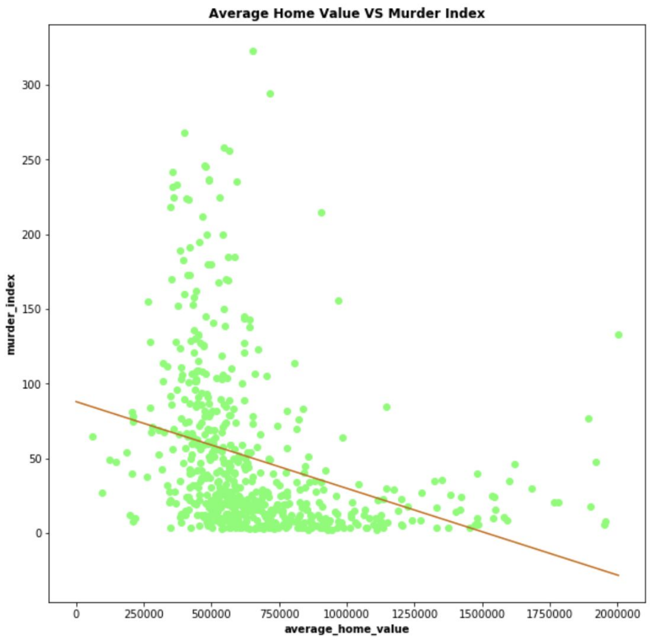
```
Statistic Summary:
y = -0.0009421105807451566x + 179.25153282920314
R value is -0.46669005436169064
R-squared value is 0.21779960684011776
P value is 5.628856602677929e-10
```



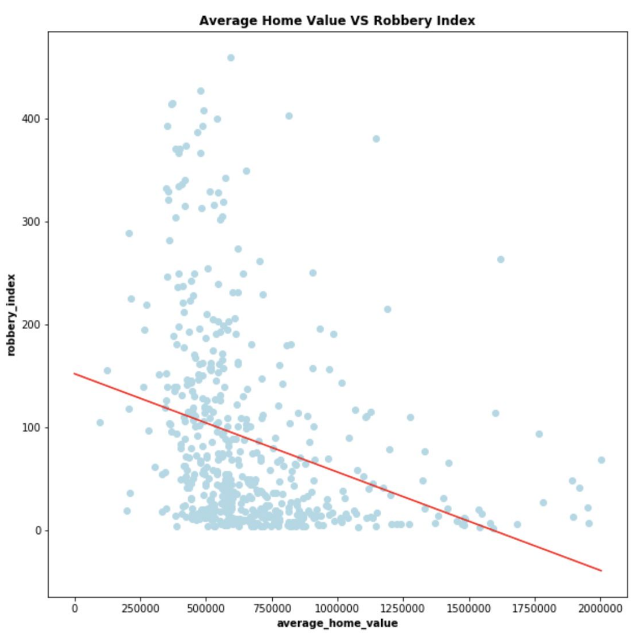

```
Statistic Summary:
y = -0.00018508552875060643x + 95.82001791250997
R value is -0.16140457582230458
R-squared value is 0.02605143709637807
P value is 0.00016309104508716005
```



```
Statistic Summary:
y = -5.8025420385750564e-05x + 88.03252401379632
R value is -0.3099800439867244
R-squared value is 0.0960876276700116
P value is 6.767381145239242e-15
```

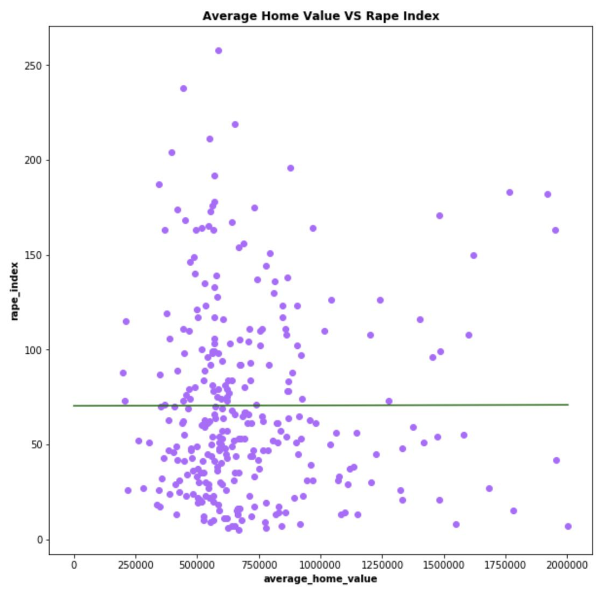



```
Statistic Summary:
y = -9.553909355876393e-05x + 152.219030841975
R value is -0.3027584894515207
R-squared value is 0.09166270293496656
P value is 8.36445942891716e-13
```

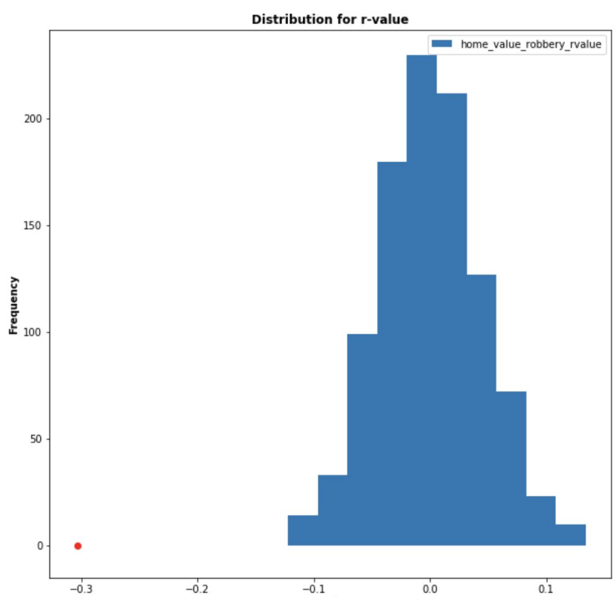


The result of our sixth model (average home value vs rape index) is y = 2.6522e-07x + 70.275, r is 0.0017, r-square is 2.911e-06, and p-value is less than 0.976. Because this model is not statistically significant (p-value greater than 0.05), and there is no negative correlation between the corresponding variables, we decided not to use it in our analysis. (For more information, please see our coding PDF).

```
Statistic Summary:
y = 2.6522074720393594e-07x + 70.27528337611223
R value is 0.0017062386211373747
R-squared value is 2.9112502322607695e-06
P value is 0.9762864902433023
```



The A|B testing in our codes has shown that the correlations in all first five models are statistically significant. (For more information, please see our coding PDF).

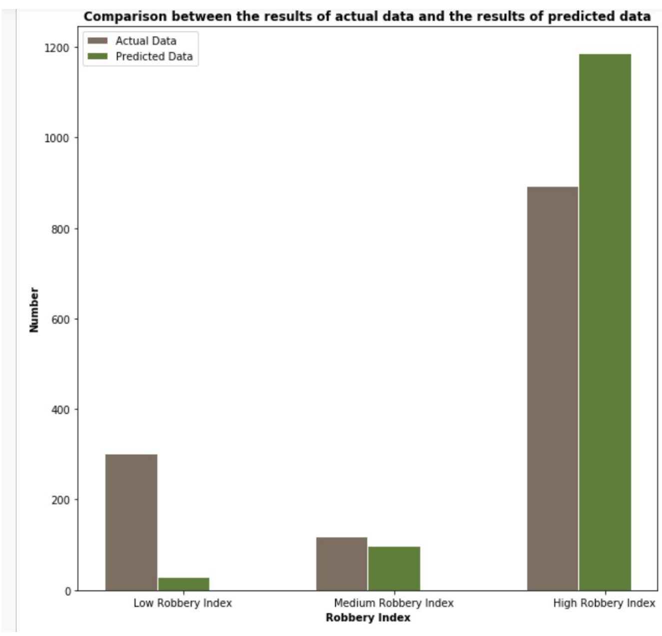


The results of Machine Learning Cross-Validation:

Six_Combinations	Logistic_Regression_Model_Score	SVM_Model_Score	Random_Forest_Classifier_Model_Score
avg_household_income vs murder_index	0.330296	0.433744	0.426601
avg_household_income vs robbery_index	0.685961	0.665271	0.588916
avg_household_income vs rape_index	0.665025	0.650739	0.461823
avg_home_value vs murder_index	0.330296	0.371182	0.392365
avg_home_value vs robbery_index	0.685961	0.678818	0.490640
avg_home_value vs rape_index	0.665025	0.672167	0.461823

From the table above, it is clear that the combination of average household income versus robbery index receives a relatively higher accuracy score for testing performed by three models. Therefore, we decide to use the second combination as our ideal combination to perform predictive analysis. (For more information, please see our coding PDF).

Predict the results of the predicted analysis:



After using our second combination (average household income versus robbery index) to perform predictive analysis on the city of Chicago, the result shows that the model has predicted **65.3%** of the communities of Chicago **correctly** in terms of three classification labels: low crime index, medium crime index, and high crime index. The group-bar chart above also supports our conclusion on the accuracy score. (For more information, please see our coding PDF).

Because our analysis is based on web-scraping from ArcGIS, there are potential biases in the data we analyzed. The potential **limitation** of our model would be a different definition of street crimes around the U.S. This is an issue when the ArcGIS database collected and recorded the data from various cities. For example, some places might consider street fight one of the crimes while some other areas might only count armed or endangered crimes. Inconsistency in counting crimes might lead to some inaccuracy in the data and our analysis. Another possible **confounder** of our analysis model would also come from the data source we used. Due to racial biases, crimes in poor neighborhoods are less likely to be recorded than those in affluent communities because of the lack of police and security equipment. If the area uses machine learning to dispatch police forces, the biases would be enlarged by the Feedback Loops Runaway simulation. This simulation takes data and assigns the police to places with more crimes recorded, which results in more crimes observed and a higher chance of dispatching the police to this area again.

To set out and address the problem with **bias**, we would have to run our simulations with different independent and dependent variables in 6 combinations and then cross-validate the results. To solve the **limitation** in this project, we need to ensure to collect all of our data from one data source with a consistent definition of “crime”. And to handle the **confounder** when possible racial biases might affect the recorded crime cases, we can manually de-bias our database as well as the algorithm we used to analyze it.

In order to better address any ethical and societal implications mentioned in the Ethical consideration section, we would want to both get and use the data right. First and foremost, the website we scrap the data from has to be official and professional. Ensuring the trustworthiness of the database can eliminate the bias from the data collection process. Secondly, we would remove any PIIs that are not related to our analysis without manipulating the data itself to protect personal information. Since we run the algorithm only once with given data, there will not be any p-packing taking place during our operation, thus decreasing the bias. Then, to verify the **accuracy** of data and its result, we applied different testing methods we learned in this course. For instance, the cross-validation and accuracy score calculation both check the validity of our models. Calculation of the p-value also adds credibility to this simulation. After all, it is impossible for us to completely eliminate the bias so far because “trying to be fair in one way necessarily means being unfair in another way” according to “The Myth of the Impartial Machine”, and what we did can minimize the bias factor as much as possible in our analysis.

Group Participation

Yiming Hao has written Ethical considerations, as well as proofread the Data. **Weiyue Li** has come with this topic, gotten approval from the Professor via office hours, written the Question, Hypothesis, Background Information, and Data, and helped proofread the Ethical Considerations. **Yi Li** has collected the data from ArcGIS, composed the Analysis Proposal, and contributed to all extra credit analysis. **Xinlan Lin** has helped with the Ethical Considerations and written the Discussion. **Zhuojin Yu** has formulated the Analysis Proposal with Yi Li and helped improve both the Extra Credit and the Ethical Considerations.