MATH2319 Machine Learning Project Phase 1
Predicting Individual Income using 1994 US Census Data

Names: John Doe & Jane Doe
Student ID: s1111111 & s2222222

March 13, 2019

Contents

Chapter 1

Introduction

1.1 Objective

The objective of this project is to predict whether an individual earns more than USD 50,000 or less in a year using the 1994 US Census Data. The data sets were sourced from the UCI Machine Learning Repository at http://archive.ics.uci.edu/ml/datasets/Census+Income [1]. This project has two phases. Phase I focuses on data preprocessing and exploration, as covered in this report. We shall present model building in Phase II. The rest of this report is organised as follows. Section 2 describes the data sets and their attributes. Section 3 covers data pre-processing. In Section 4, we explore each attribute and their inter-relationships. The last section presents a brief summary. Compiled from Jupyter Notebook, this report contains both narratives and the `Python` codes used for data pre-processing and exploration.

1.2 Data Sets

The UCI Machine Learning Repository provides five data sets, but only `adult.data`, `adult.test`, and `adult.names` were useful in this project. adult.data and adult.test are the training and test data sets respectively. adult.names contains the details of attributes or variables. The training data set has 32,561 training observations. Meanwhile, the test data set has 16,281 test observations. Both data sets consist of 14 descriptives features and one target feature. In this project, we combined both training and test data into one. In Phase II, we would build the classifiers from the combined the data set and evaluate their performance using cross-validation.

1.2.1 Target Feature

The response feature is income which is given as:

$$\text{income} = \begin{cases} > 50K & \text{if the income exceeds USD 50,000} \\ \leq 50K & \text{otherwise} \end{cases}$$

The target feature has two classes and hence it is a binary classification problem. To reiterate, The goal is to predict **whether a person makes over USD 50,000 a year**.

1.2.2 Descriptive Features

The variable description is produced here from `adult.names` file:

- `age`: continuous.
- `workclass`: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
- `fnlwgt`: continuous.

- `education`: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
- `education-num`: continuous.
- `marital-status`: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
- `occupation`: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-*inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
- `relationship`: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
- `race`: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
- `sex`: Female, Male.
- `capital-gain`: continuous.
- `capital-loss`: continuous.
- `hours-per-week`: continuous.
- `native-country`: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

Most of the descriptive features are self-explanatory, except `fnlwgt` which stands for "Final Weight" defined by the US Census. The weight is an "estimate of the number of units in the target population that the responding unit represents" [1]. This feature aims to allocate similar weights to people with similar demographic characteristics.

Chapter 2

Data Pro-processsing

2.1 Preliminaries

We read the training and test datasets directly from the data URL's. Also, since the data sets do not contain the attribute names, they were explicitly specified during loading the data sets. `adultData` was read and then it would be concatenated with `adultTest`.

```
In [1]: attributeNames = [
            'age',
            'workclass',
            'fnlwgt',
            'education',
            'education-num',
            'marital-status',
            'occupation',
            'relationship',
            'race',
            'sex',
            'capital-gain',
            'capital-loss',
            'hours-per-week',
            'native-country',
            'income',
        ]

        url = (
            "http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data",
            "http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.test",
        )
```

```
In [2]: import pandas as pd

        adultData = pd.read_csv(url[0], sep = ',', names = attributeNames, header = None)
        adultTest = pd.read_csv(url[1] , sep = ',', names = attributeNames, skiprows = 1)
```

Note: alternatively, we could read the datasets from `txt` files downloaded from the UCI repositort as following:

```
adultData = pd.read_csv('adult.data.txt' sep = ',', names = attributeNames, header = None)
adultTest = pd.read_csv('adult.test.txt' , sep = ',', names = attributeNames, skiprows = 1)
```

```
In [3]: adultData = pd.concat([adultData,adultTest])
```

2.2 Data Cleaning and Transformation

First, we confirmed that the feature types matched the description as outlined in the documentation.

```
In [4]: print(f"Dimension of the data set is {adultData.shape} \n")
        print(f"Data Types are: ")
        print(adultData.dtypes)
```

```
Dimension of the data set is (48842, 15)

Data Types are:
age                int64
workclass         object
fnlwgt             int64
education         object
education-num      int64
marital-status    object
occupation        object
relationship      object
race              object
sex               object
capital-gain       int64
capital-loss       int64
hours-per-week     int64
native-country    object
income            object
dtype: object
```

On surface, no attributes contain `NaN` values (though the missing values might be coded with different labels) as shown in the code chunk.

```
In [5]: print(f"\nNumber of missing value for each feature:")
        print(adultData.isnull().sum())
```

```
Number of missing value for each feature:
age               0
workclass         0
fnlwgt            0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
income            0
dtype: int64
```

Table 2 shows the target feature, `income` had four categories (or a cardinality of 4). It was supposed to be 2 since income must be binary. We shall present further investigation and how we handled this issue later. Table 1 and Table 2 reveals that the education_num ranged from 1 to 16 which coincided with the cardinality of education. They might convey the same information. In Table 1, the max value of `capital_gain` was 99999, potentially a value to represent missing value. The max value of `hours_per_week` was 99. It could be a valid or missing value.

```
In [6]: from IPython.display import display, HTML
        display(HTML('<b>Table 1: Summary of continuous features</b>'))
        display(adultData.describe(include = 'int64'))

        display(HTML('<b>Table 2: Summary of categorical (object) features</b>'))
        display(adultData.describe(include = 'object'))
```

```
<IPython.core.display.HTML object>
```

	age	fnlwgt	education-num	capital-gain	capital-loss	\
count	48842.000000	4.884200e+04	48842.000000	48842.000000	48842.000000	
mean	38.643585	1.896641e+05	10.078089	1079.067626	87.502314	
std	13.710510	1.056040e+05	2.570973	7452.019058	403.004552	
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	
25%	28.000000	1.175505e+05	9.000000	0.000000	0.000000	
50%	37.000000	1.781445e+05	10.000000	0.000000	0.000000	
75%	48.000000	2.376420e+05	12.000000	0.000000	0.000000	
max	90.000000	1.490400e+06	16.000000	99999.000000	4356.000000	

	hours-per-week
count	48842.000000
mean	40.422382
std	12.391444
min	1.000000
25%	40.000000
50%	40.000000
75%	45.000000
max	99.000000

```
<IPython.core.display.HTML object>
```

	workclass	education	marital-status	occupation	relationship	\
count	48842	48842	48842	48842	48842	
unique	9	16	7	15	6	
top	Private	HS-grad	Married-civ-spouse	Prof-specialty	Husband	
freq	33906	15784	22379	6172	19716	

	race	sex	native-country	income
count	48842	48842	48842	48842
unique	5	2	42	4
top	White	Male	United-States	<=50K
freq	41762	32650	43832	24720

Some character or categorical attributes contain excessive white spaces and irregular cases. Since each column is not a string but a series, applying the `strip()` function would result in errors. Therefore, we

wrote function named `noWhiteSpace(x)` to remove extra white spaces in a series object x (see the chunk code below). This function is applied on each categorical attribute using `apply(noWhiteSpace)`. Similarly, a function named lower(x) was written to convert string values to lower cases.

```
In [7]: def noWhiteSpace(x):
            return x.strip()

In [8]: def lower(x):
            return x.lower()
```

2.2.1 Continuous Features

As discussed in previous section, `fnlwght` has no predictive power and hence it was removed. The range of `age` appears reasonable as the minimum and maximum ages are 17 and 90 respectively.

```
In [9]: adultData = adultData.drop(['fnlwgt'],1)
```

We observed that 244 rows have capital gain values of 99,999. These observations report to have earned more than USD 50,000, as indicated by `income > 50K` or `income > 50k.` in the codes below. We also found inconsistent labelling of `income`; some were recorded as `>50K.` whereas some were labelled as `>50k`. This explained why Table 2 shows `income` has 4 categories. We will discuss how we fixed the incorrect labels later.

```
In [10]: adultData.loc[adultData['capital-gain'] == 99999.000000, 'income'].value_counts()

Out[10]:  >50K     159
          >50K.     85
          Name: income, dtype: int64
```

By excluding the observations with capital gain of USD 99,999, despite persistently strong negative skewness, the mean and maximum values drop drastically to around 582 and 41310 respectively. Since a capital gain value of 99,999 always returns a higher income earner (at least in the training data), we conjectured it might be a useful predictive value and hence we did not remove the observations with this value.

```
In [11]: adultData.loc[adultData['capital-gain'] != 99999.000000, 'capital-gain'].describe()

Out[11]: count    48598.000000
         mean       582.412136
         std       2536.651465
         min          0.000000
         25%          0.000000
         50%          0.000000
         75%          0.000000
         max      41310.000000
         Name: capital-gain, dtype: float64
```

We suspected that `capital-loss` = $-$`capital-gain` because an individual can either pay any capital gain or claim for capital loss to reduce capital gain in future [2]. Another possibility is to pay neither gain nor loss. Hence, it is more reasonable to record gain or loss as a single variable, rather than having them separate. Before defining such new variable, we used the mask to verify that no observations recorded positive `capital-gain` and `capital-loss` values.

```
In [12]: mask_both = (adultData['capital-gain'] > 0) & (adultData['capital-loss'] > 0)
         mask_both.value_counts()
```

```
Out[12]: False    48842
         dtype: int64
```

Next, we defined `capital = capital-gain - capital-loss` and then removed the latter. The summary statistic for `capital` is displayed below.

```
In [13]: adultData['capital'] = adultData['capital-gain'] - adultData['capital-loss']
         adultData = adultData.drop(['capital-gain', 'capital-loss' ], 1)
         adultData['capital'].describe()
```

```
Out[13]: count    48842.000000
         mean       991.565313
         std       7475.549906
         min      -4356.000000
         25%          0.000000
         50%          0.000000
         75%          0.000000
         max      99999.000000
         Name: capital, dtype: float64
```

On the other hand, 137 observations report 99 working hours per week with 60 of them earn below USD 50,000 (see the chunk codes below). As shown previously, the `income` variable contains inconsistent labels.

```
In [14]: adultData.loc[ adultData['hours-per-week'] == 99, 'income'].value_counts()
```

```
Out[14]:  <=50K     60
          <=50K.    36
          >50K      25
          >50K.     16
         Name: income, dtype: int64
```

By excluding observations with 99 hours per week, we found that the maximum become 98 - refuting our initial suspicion. Perhaps, it was plausible to work more than 90 hours per week. Therefore, we concluded the range of `hours-per-week` is reasonable at this point and hence did not pre-process it further.

```
In [15]: adultData.loc[adultData['hours-per-week'] != 99, 'hours-per-week'].describe()
```

```
Out[15]: count    48705.000000
         mean        40.257612
         std         12.012519
         min          1.000000
         25%         40.000000
         50%         40.000000
         75%         45.000000
         max         98.000000
         Name: hours-per-week, dtype: float64
```

2.2.2 Categorical Features

All categorical features, including the target feature, contain excessive white space in front of their characters (see the chunk codes below). For example, `workclass` has category of "State-gov". All observations with "State-gov" are recorded as " State - Gov ". There is no other instances such as "State-gov" or "state-gov". the cases are regular and consistent and hence there was no need to convert all strings into lower cases.

```
In [16]: categoricalColumn = ['workclass', 'education', 'marital-status', 'relationship',
                              'occupation', 'race', 'sex','native-country','income']
         for col in categoricalColumn:
             print('Unique values for ' + col)
             print(adultData[col].unique())
             print('')
```

```
Unique values for workclass
[' State-gov' ' Self-emp-not-inc' ' Private' ' Federal-gov' ' Local-gov'
 ' ?' ' Self-emp-inc' ' Without-pay' ' Never-worked']

Unique values for education
[' Bachelors' ' HS-grad' ' 11th' ' Masters' ' 9th' ' Some-college'
 ' Assoc-acdm' ' Assoc-voc' ' 7th-8th' ' Doctorate' ' Prof-school'
 ' 5th-6th' ' 10th' ' 1st-4th' ' Preschool' ' 12th']

Unique values for marital-status
[' Never-married' ' Married-civ-spouse' ' Divorced'
 ' Married-spouse-absent' ' Separated' ' Married-AF-spouse' ' Widowed']

Unique values for relationship
[' Not-in-family' ' Husband' ' Wife' ' Own-child' ' Unmarried'
 ' Other-relative']

Unique values for occupation
[' Adm-clerical' ' Exec-managerial' ' Handlers-cleaners' ' Prof-specialty'
 ' Other-service' ' Sales' ' Craft-repair' ' Transport-moving'
 ' Farming-fishing' ' Machine-op-inspct' ' Tech-support' ' ?'
 ' Protective-serv' ' Armed-Forces' ' Priv-house-serv']

Unique values for race
[' White' ' Black' ' Asian-Pac-Islander' ' Amer-Indian-Eskimo' ' Other']

Unique values for sex
[' Male' ' Female']

Unique values for native-country
[' United-States' ' Cuba' ' Jamaica' ' India' ' ?' ' Mexico' ' South'
 ' Puerto-Rico' ' Honduras' ' England' ' Canada' ' Germany' ' Iran'
 ' Philippines' ' Italy' ' Poland' ' Columbia' ' Cambodia' ' Thailand'
 ' Ecuador' ' Laos' ' Taiwan' ' Haiti' ' Portugal' ' Dominican-Republic'
 ' El-Salvador' ' France' ' Guatemala' ' China' ' Japan' ' Yugoslavia'
 ' Peru' ' Outlying-US(Guam-USVI-etc)' ' Scotland' ' Trinadad&Tobago'
 ' Greece' ' Nicaragua' ' Vietnam' ' Hong' ' Ireland' ' Hungary'
 ' Holand-Netherlands']

Unique values for income
[' <=50K' ' >50K' ' <=50K.' ' >50K.']
```

```
In [17]: for col in categoricalColumn:
             adultData[col] = adultData[col].apply(noWhiteSpace)
```

The `workclass`, `occupation`, and `native-country` contain some missing values encoded as "?". We noticed in most of the missing observations, the `workclass` and `occupation` variables are missing data

together, accounting for 5.7% of the data sets. The nativecountry contains less than 2 % of missing values. Note that the missing data (around more than 90 %) are predominantly to the <=50K income whereas ~76 % of observations pertain to the <=50K at an aggregate level. Therefore, we decided not to impute these missing values but removed them instead as there would be very minimal information loss.

```
In [18]: mask = (adultData['workclass'] == '?') & (adultData['occupation'] == '?')
         mask.value_counts(normalize = True)*100
```

```
Out[18]: False    94.269276
         True      5.730724
         dtype: float64
```

We removed the rows with missing occupations, workclass, and native-country.

```
In [19]: adultData = adultData[adultData['occupation'] != "?"]
         adultData = adultData[adultData['native-country'] != "?"]
```

Lastly we corrected incorrect labels of income to make sure it is binary.

```
In [20]: print('Before correction, the numbers ofs unique income labels are: ')
         print(adultData['income'].value_counts())
         print("")

         def noDot(x):
             return x.rstrip(".")

         adultData['income'] = adultData['income'].apply(noDot)

         print('After correction, the numbers of unique income labels are: ')
         print(adultData['income'].value_counts())
         print("")
```

```
Before correction, the numbers ofs unique income labels are:
<=50K     22654
<=50K.    11360
>50K       7508
>50K.      3700
Name: income, dtype: int64

After correction, the numbers of unique income labels are:
<=50K    34014
>50K     11208
Name: income, dtype: int64
```

Chapter 3

Data Exploration

3.1 Univariate Visualisation

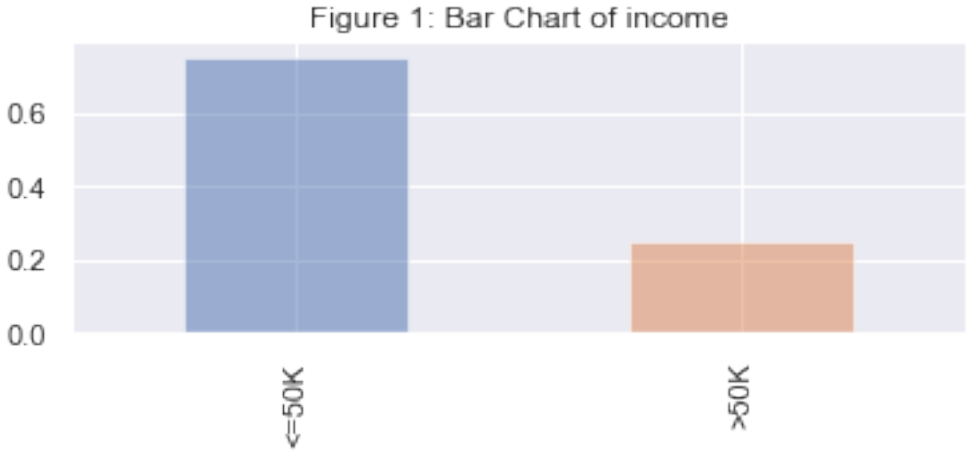
For convenience, we defined two functions named `BarPlot(x)` and `BoxHistogramPlot(x)` for categorical and numerical features respectively. For given an input categorical column `x`, `BarPlot(x)` returns a bar chart with percentage on top of each bar. A bar chart is useful to present the proportions by categories. For given an input numerical column `x`, `BoxHistogramPlot(x)` plots a histogram and a box plot. A histogram is useful to visualize the shape of the underlying distribution whereas A box plot tells the range of the attribute and helps detect any outliers. The following chunk codes show how these function were defined using the numpy library and the matplotlib library.

```
In [21]: import matplotlib.pyplot as plt
         import seaborn as sns

         sns.set(color_codes=True)

         def BarPlot(x):
             total = float(len(adultData))
             ax = adultData[x].value_counts(normalize = True).plot(
                 kind = "bar", alpha = 0.5)

         def BoxHistogramPlot(x):
             f, (ax_box, ax_hist) = plt.subplots(2, sharex=True,
                                                 gridspec_kw={"height_ratios": (.15, .85)})
             sns.boxplot(x, ax=ax_box)
             sns.distplot(x, ax=ax_hist)

             ax_box.set(yticks=[])
             sns.despine(ax=ax_hist)
             sns.despine(ax=ax_box, left=True)
             plt.show()
```

Except `native-country` (which has too many categories), we applied the `BarPlot` for all categorical columns. In Figure 1, around 75 % of individuals earned less than USD 50,000 in 1994. Figure 2 reveals most of them (~74 %) worked for private companies. Collectively, close to 14 % of individuals work for the US Governments at different levels. It would be intriguing if there is any income difference working for Federal, State, and Local governments. Figure 3 shows more than 60 % (including those who obtained higher levels of education) of working individuals at least graduated from high schools. It is plausible that a higher level of education should result in a higher income. In Figure 4, close to 50 % of individuals are married followed by ~32 % of unmarried. The divorced accounted for ~14% as well. In Figure 5, the

proportion of white collars (i.e. Administrative-Clerical, Executive-Managerial, Professional-Specialty and Sales) account approximately 50% which is slightly higher than blue-collar jobs. Unsurprisingly, the White Americans dominated the datasets as shown in Figure 6. Figure 7 shows that majority of males were the income earners. This is consistent with Figure 8 where the husbands were the bread-winners.

```
In [22]: i = 1   # initialize figure labelling4)
         for col in ['income', 'workclass', 'education', 'marital-status',
                     'occupation', 'race', 'sex', 'relationship']:
             plt.figure(figsize=(6,2))
             plt.title("Figure " + str(i) + ": Bar Chart of " + col, fontsize = 12)
             BarPlot(col)
             plt.show()
             i = i + 1
```



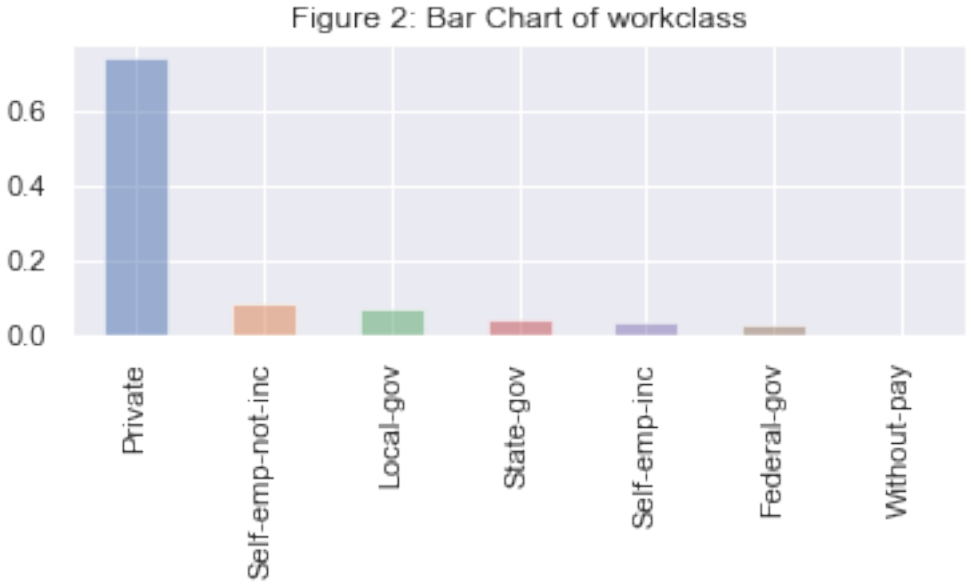


Figure 3: Bar Chart of education

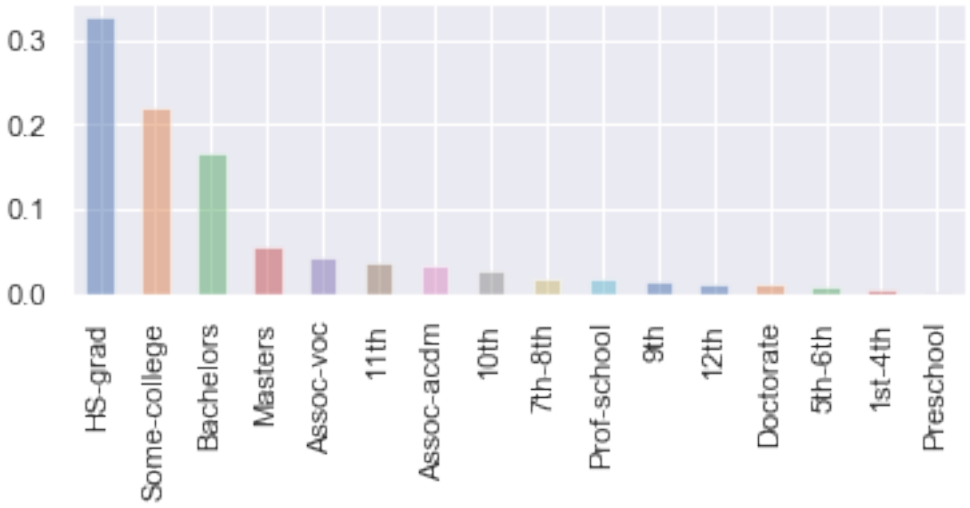


Figure 4: Bar Chart of marital-status

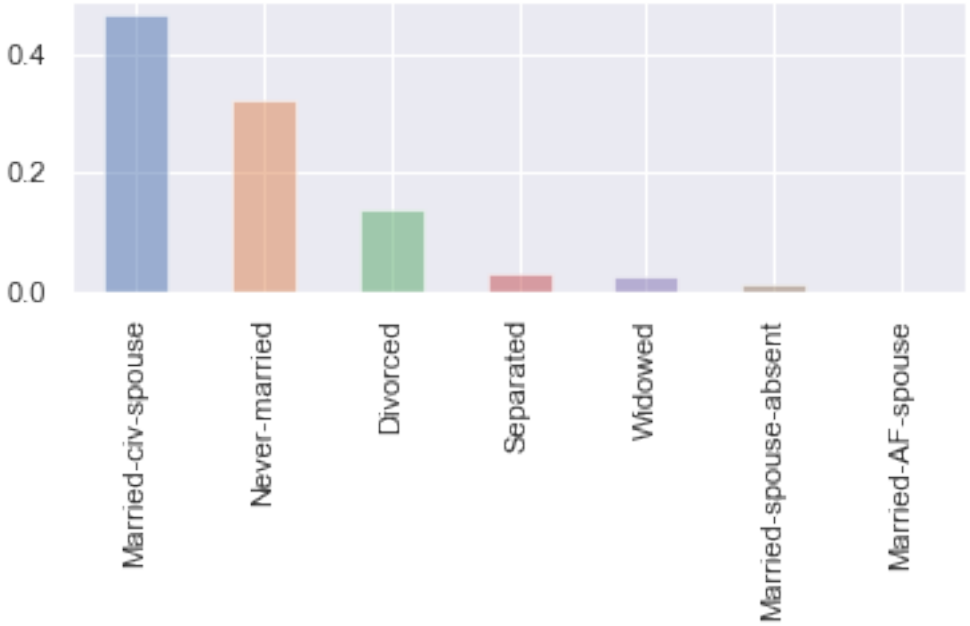


Figure 5: Bar Chart of occupation

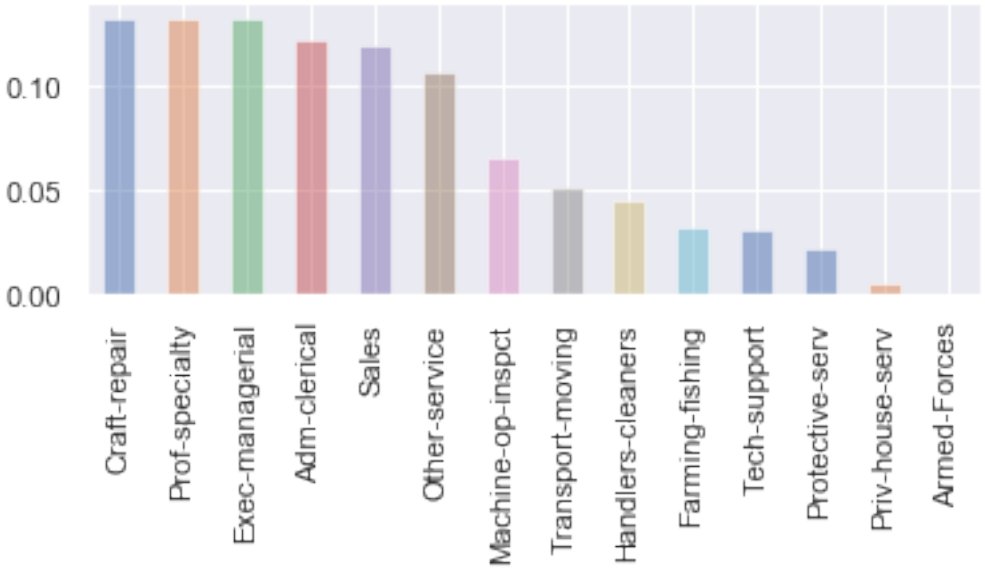
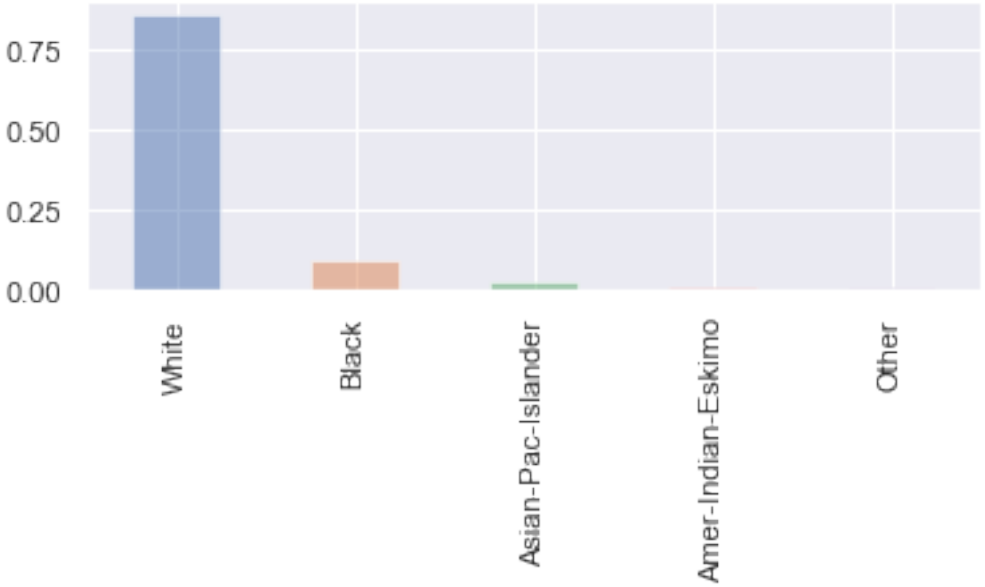
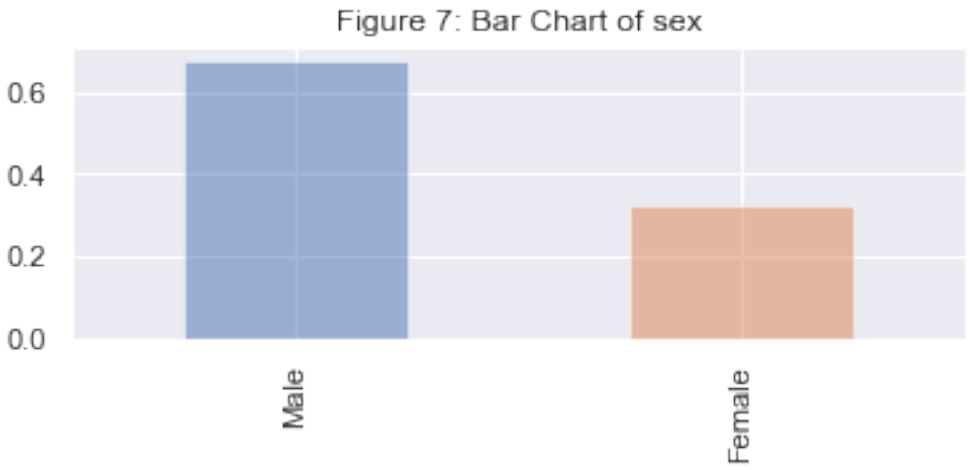
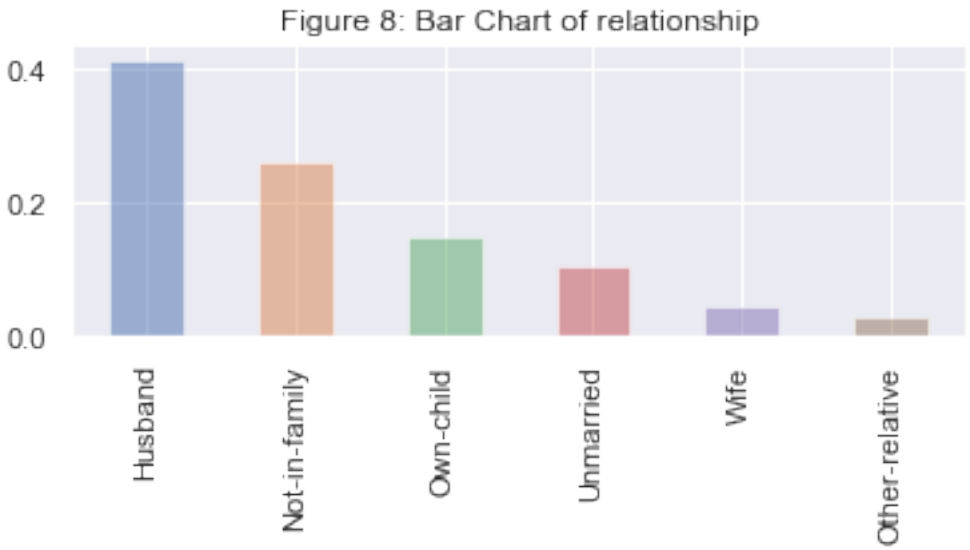


Figure 6: Bar Chart of race



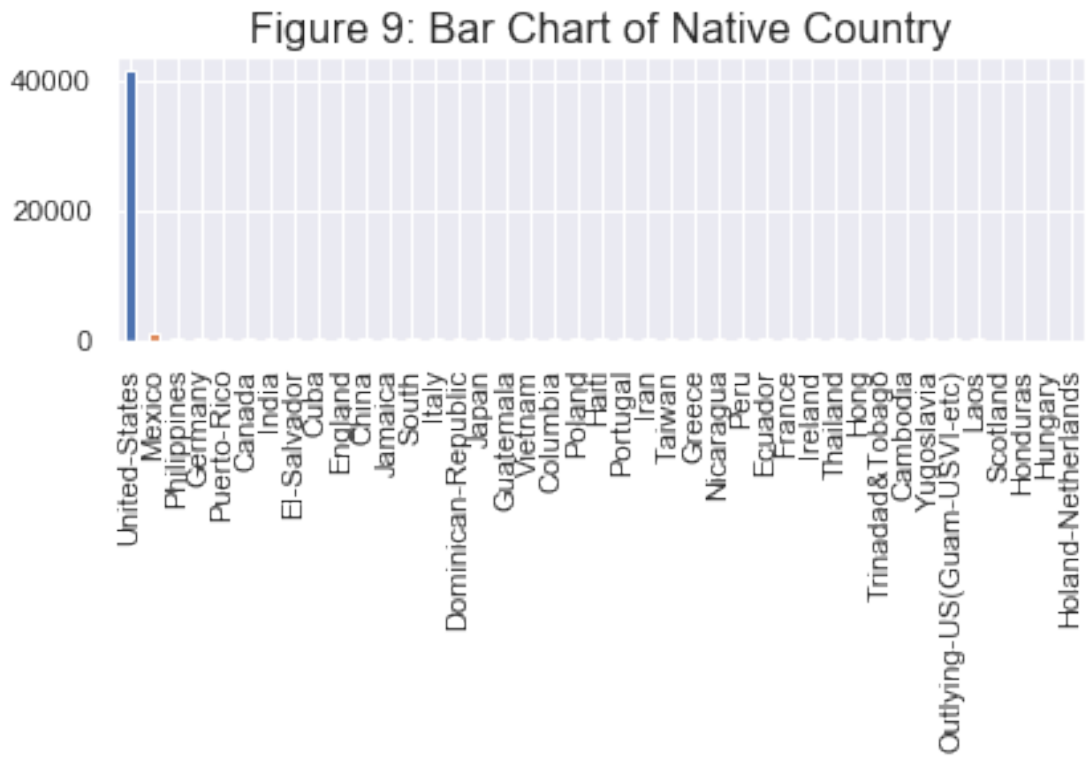




Observations from the exploration of individual attributes; There are many more cases of earning below 50 thousand dollars than above. Private is the most popular work class by far, accounting for 73.6 percent of work class types. 'High school graduate' is the most popular education level, followed by 'some college' and then 'bachelors' (having completed college). The largest group in marital status is married, followed by never married and then divorced. Regarding occupation, the least popular jobs are private house services and the armed forces. The individuals in the dataset are predominantly white and male. Because of this, there are more husbands than wives.

Since `native-country` has too many categories, it might be too granular for predictive modelling (see Figure 9). Also, more 90 % of individuals were born in the US.

```
In [23]: ax = adultData['native-country'].value_counts().plot(kind= "bar")
         ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
         plt.tight_layout()
         plt.title("Figure " + str(i) + ": Bar Chart of Native Country", fontsize = 16)
         plt.show()
         i = i + 1
```



Since `native-country` is too granular and unbalanced, we decided to group other countries as "Other" except "United-States". Likewise, we also categorized other faces as "Other" except "White".

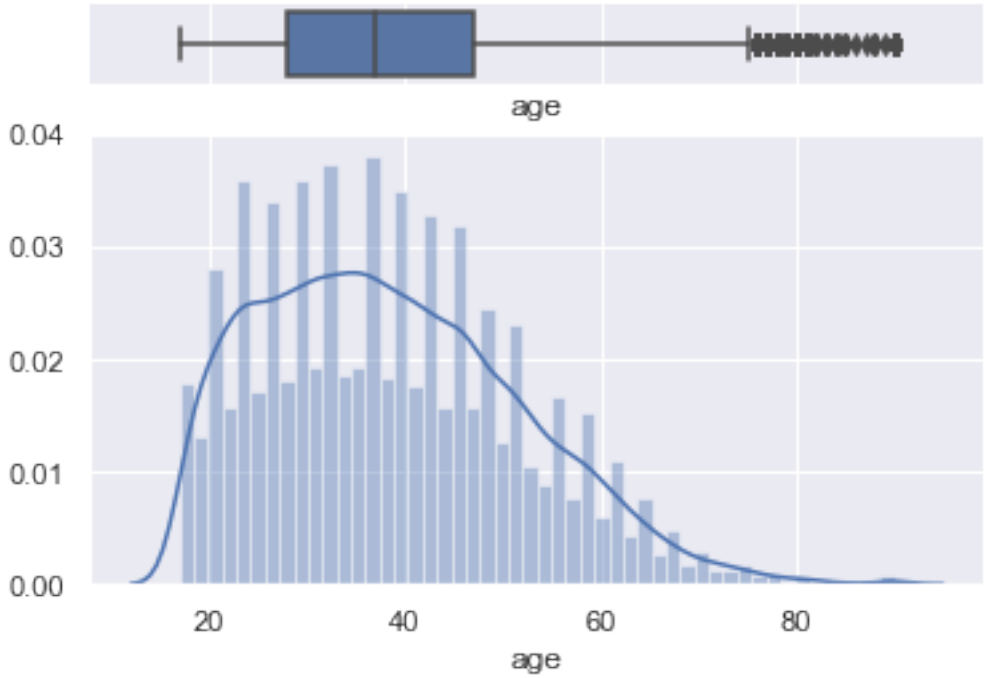
```
In [24]: adultData.loc[adultData['native-country'] != "United-States", 'native-country'] = "Other"
         adultData.loc[adultData['race'] != "White", 'race'] = "Other"
```

The `BoxHistogramPlot` function was applied to all numeric variables, except `capital` whose missing values are recored as `99,999`. Figure 9 depicts that most of individuals aged between 20 to 50 years. The histogram (Figure 10) reveals a bi-modal (two modes) distribution in the `education-num`. The first mode suggests a higher proportion of individuals spent between 8 to 10 years, mostly up to high schools. The second mode, which is lower, represents proportion of individuals who spent extra years to attend some colleges or complete Bachelor degrees. This histogram is consistent with Figure 3 (Bar Chart of `education`). Therefore, `education` and `education-num` might be "highly collinear" as they represent the same information. This might cause some estimation issues in running Logistic Regression (LR). We shall discuss how the multicollinearity issue was mitigated. In Figure 11, most of individuals work between 35 to 40 hours. In these figures, there might be very few outliers (values beyond the IQR in the boxplot) for `education-num` and `age`. On contrary, many observations lie outside the IQR in boxplot of `hours-per-week`. It is expected an individual, who worked for fewer hours, would likely earn less than AUD 50,000.

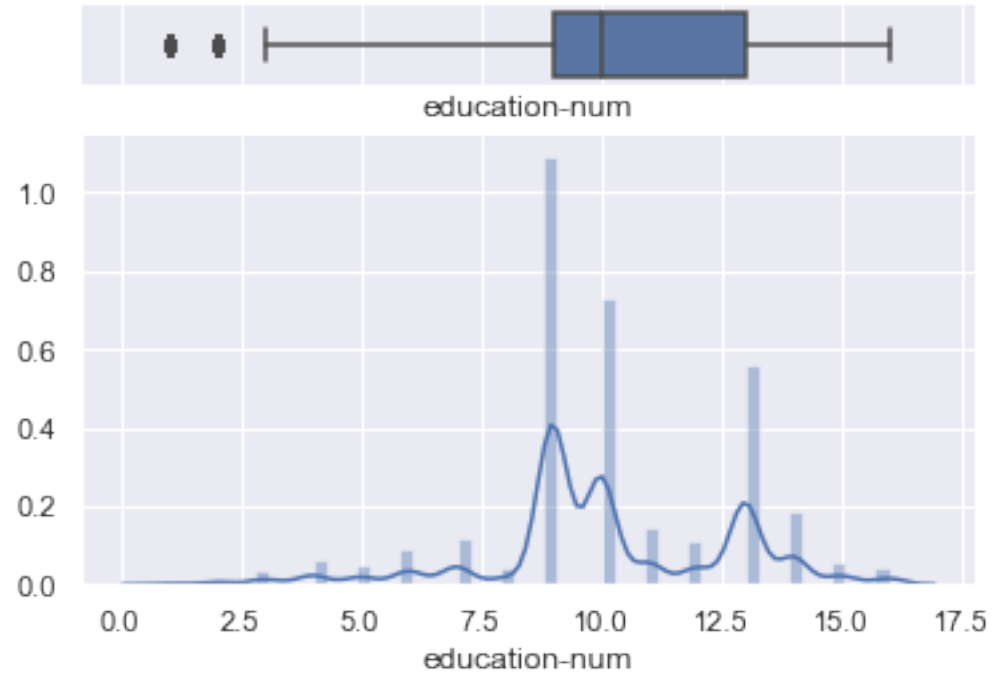
```
In [25]: for col in ['age','education-num', 'hours-per-week']:
             plt.suptitle("Figure " + str(i) + ": Histogram and Box Plot of " + col)
             BoxHistogramPlot(adultData[col])
             plt.show()
             i = 1 + i
```

```
/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using a non-tuple seque
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

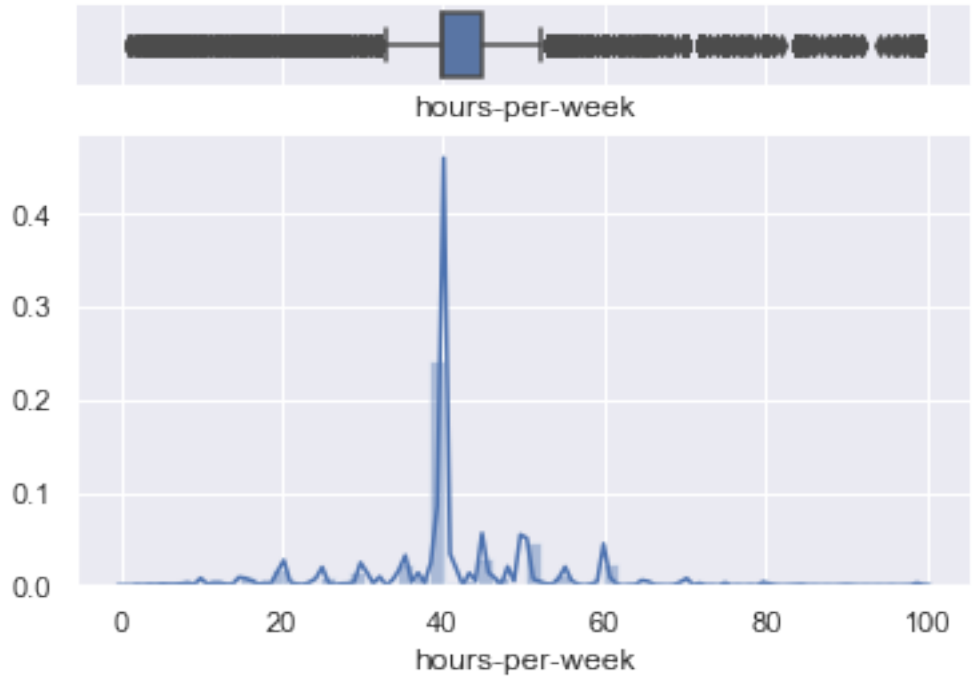
```
<Figure size 432x288 with 0 Axes>
```




```
<Figure size 432x288 with 0 Axes>
```



```
<Figure size 432x288 with 0 Axes>
```



In the previous section, we have justified why we kept the missing values recorded as 99,999 for the `capital` variable. As an unintended consequence, its histogram and boxplot would be heavily negative skewed. Therefore, we excluded these "missing values" in plotting the following diagrams.

These plots reveal that the bulk of the individuals in the data set are under 40 years old. The histogram showing the number of years in education has an interesting bimodel property, the first mode being for individuals having completed high school and maybe some college, the second being for those that completed college. Regarding working hours, the majority of people work between 30-40 hours a week, which aligns with what we would consider a normal work week.

```
In [26]: x = adultData['capital']
         x = x[x!=99999]
         plt.suptitle("Figure " + str(i) + ": Histogram and Box Plot of Capital Excluding values of 99,9
         BoxHistogramPlot(x)
         plt.show()
         i = i + 1

/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using a non-tuple sequ
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval


<Figure size 432x288 with 0 Axes>
```

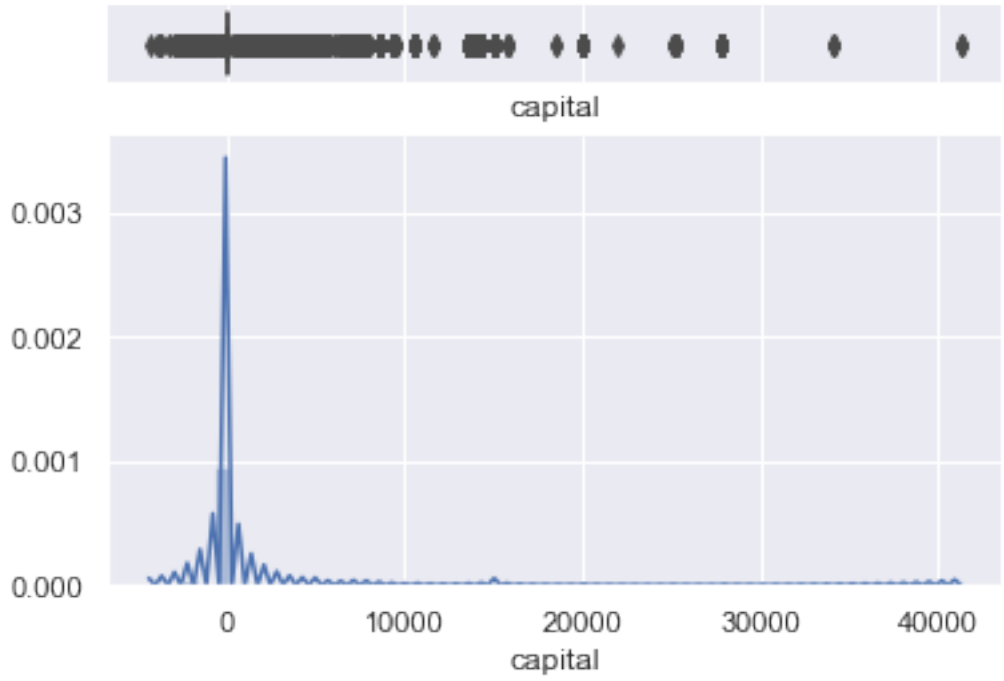


Figure 13 reveals close to 40,000 individuals neither paid nor claim for capital gain/loss. As most of individuals earn less than USD 50,000, it might be possible that they did not pay or claim as they might have not reached the third tax bracket threshold at USD 49,925 to USD 91,850 depending if taxes were filed as a single or a household. For more information, see US Tax Bracket 1994.

3.2 Multivariate Visualisation

3.2.1 Histogram of Numeric Features Segregated by Income Level

The following are histograms for each numeric attributes segregated by income level. Figure 10 shows age is negatively skewed for earners below USD 50,000 compared to high income earners. This suggests younger individuals were likely to earn less than USD 50,000 in 1995. Also, education can increase expected earnings. There is no discernible difference in hours per week (Figure 12) between two income groups. Although most individuals did not pay capital or incurred capital loss, some higher income earners paid substantial amount of capital gain.

```
In [27]: for col in ['age', 'education-num', 'hours-per-week', 'capital']:
             data1 = adultData.loc[adultData['income']=="<=50K", col]
             data2 = adultData.loc[adultData['income']==">50K", col]
             plt.hist(data1, alpha = 0.3, bins = 30)
             plt.hist(data2, alpha = 0.3, bins = 30)
             plt.title("Figure " + str(i) + ": Histogram of " + col + " segregated by income level")
             i = i + 1
             plt.legend(adultData['income'].unique(), bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.
             plt.show()
```

Figure 14: Histogram of age segregated by income level

Figure 15: Histogram of education-num segregated by income level

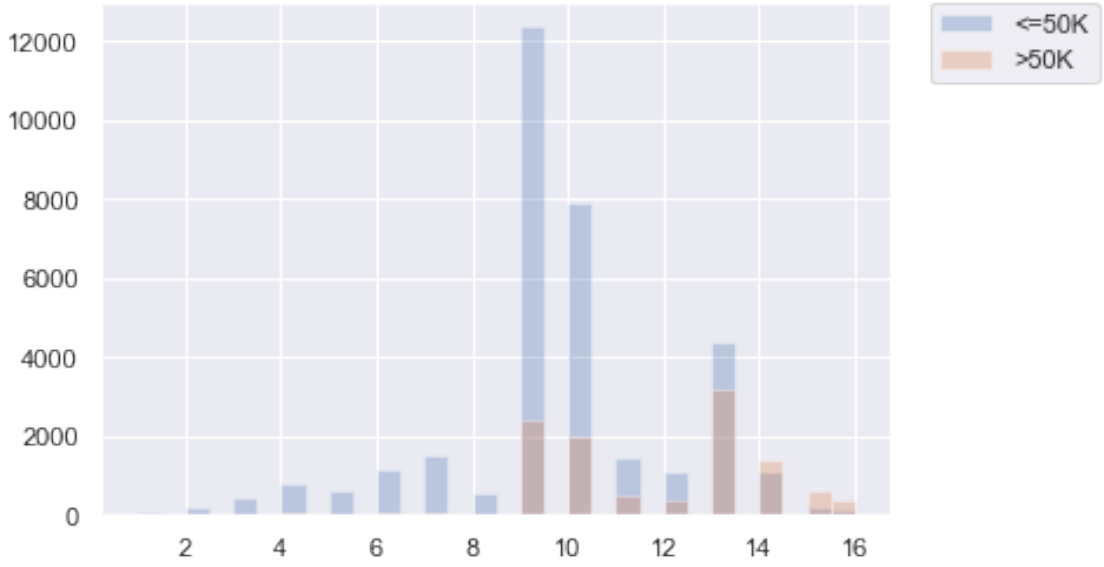


Figure 16: Histogram of hours-per-week segregated by income level

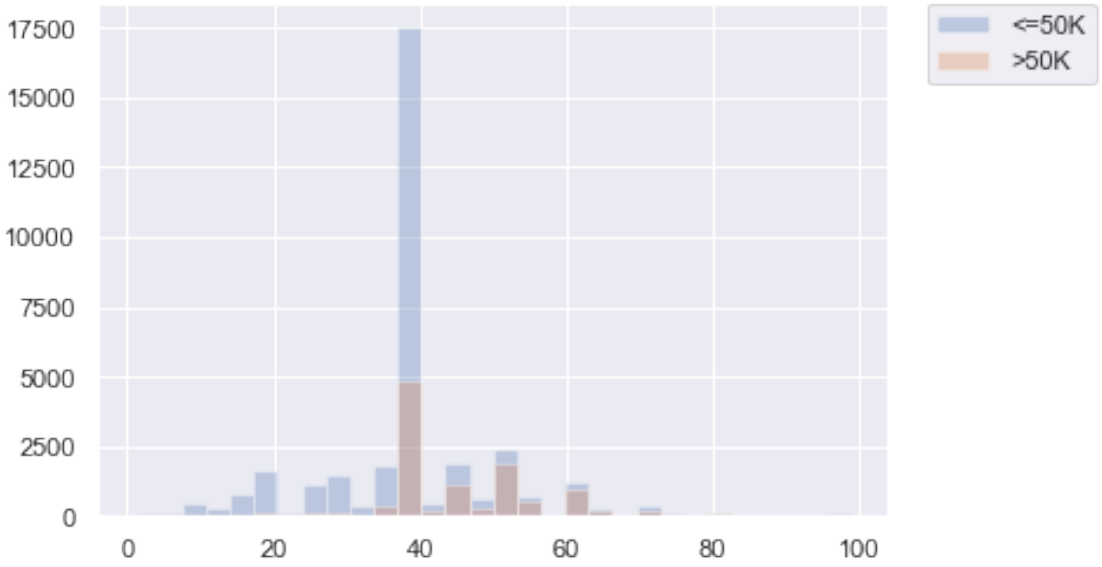
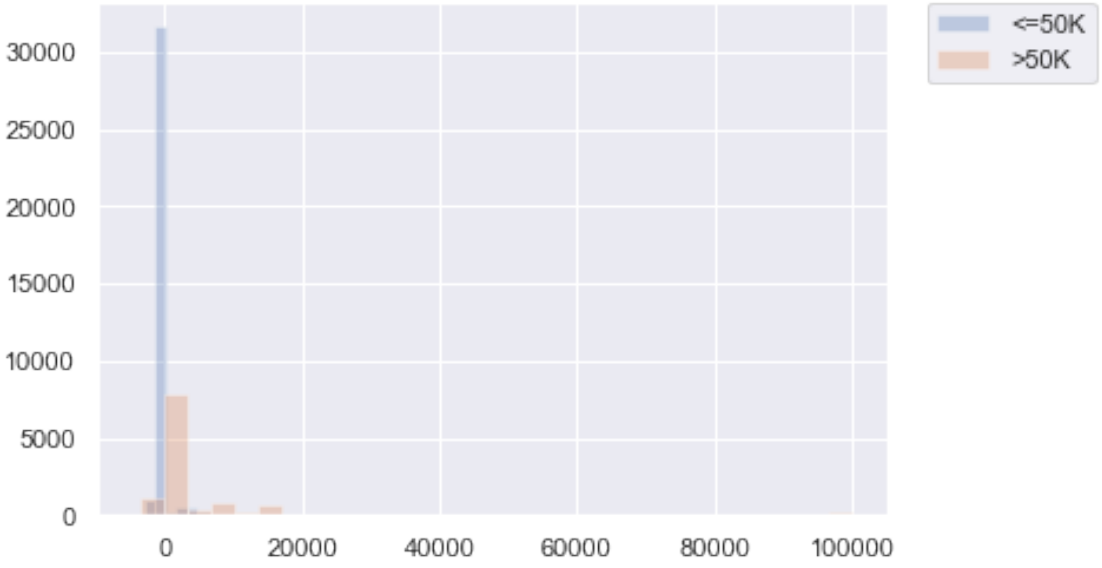


Figure 17: Histogram of capital segregated by income level

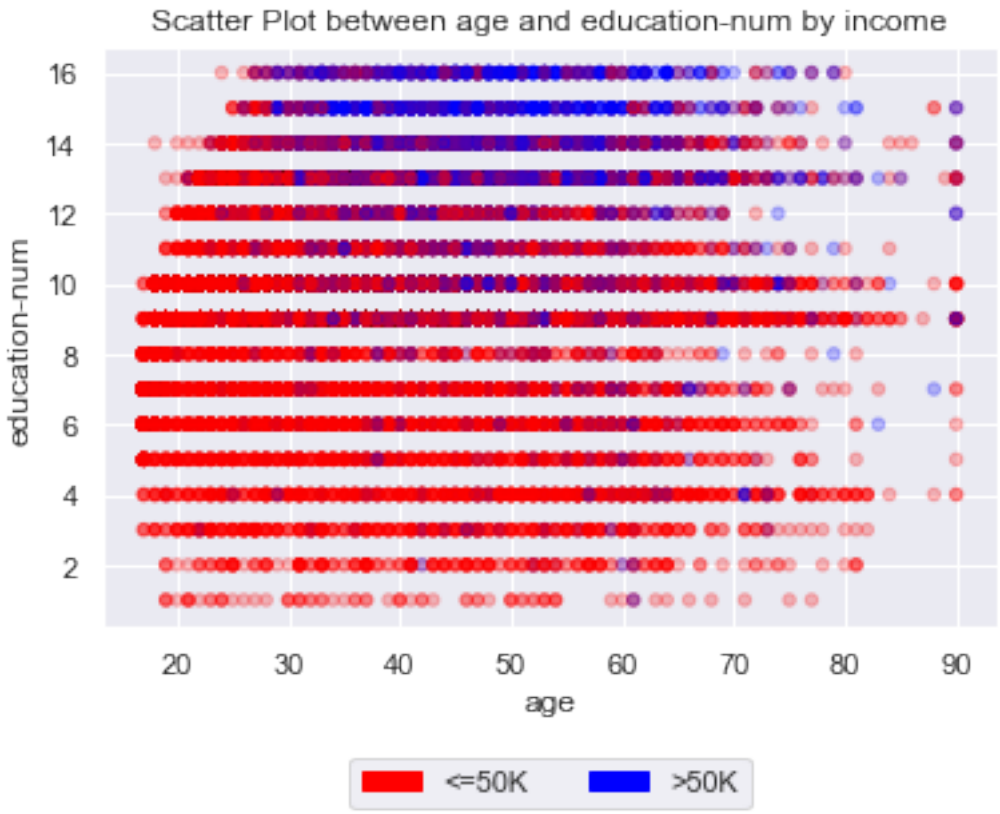


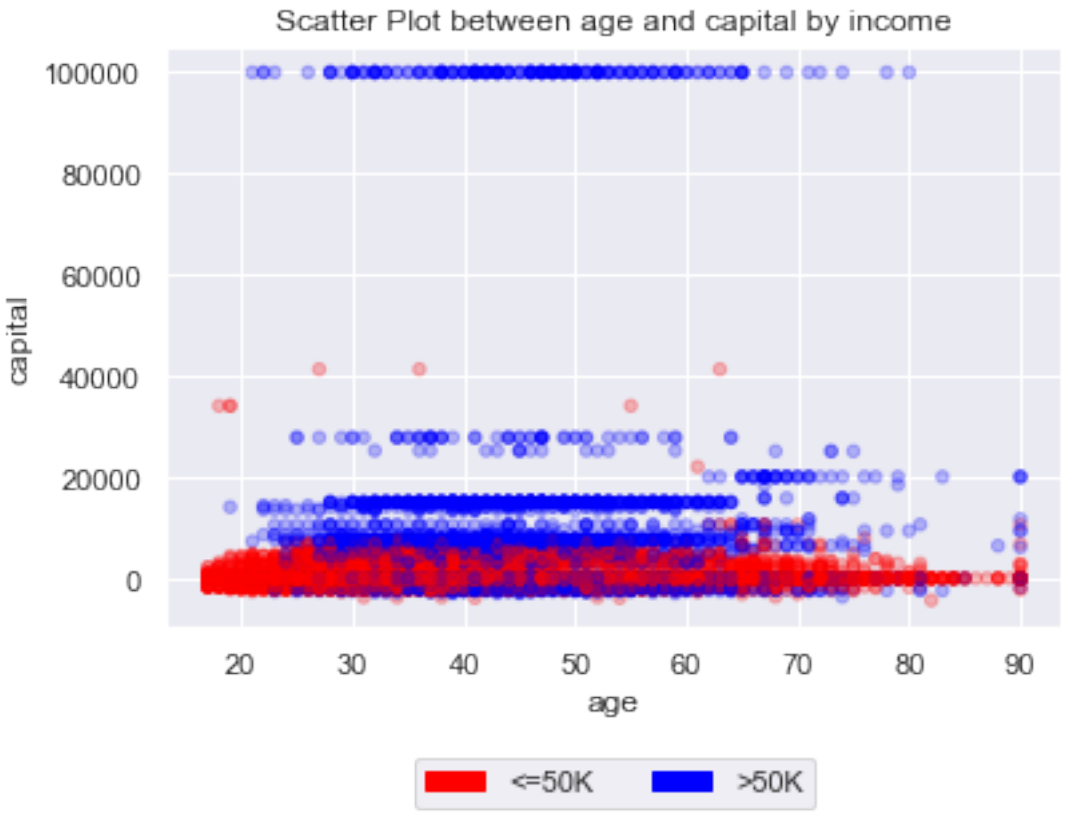
3.2.2 Pairwise Scatter Plots between Two Numeric Features by Income Level

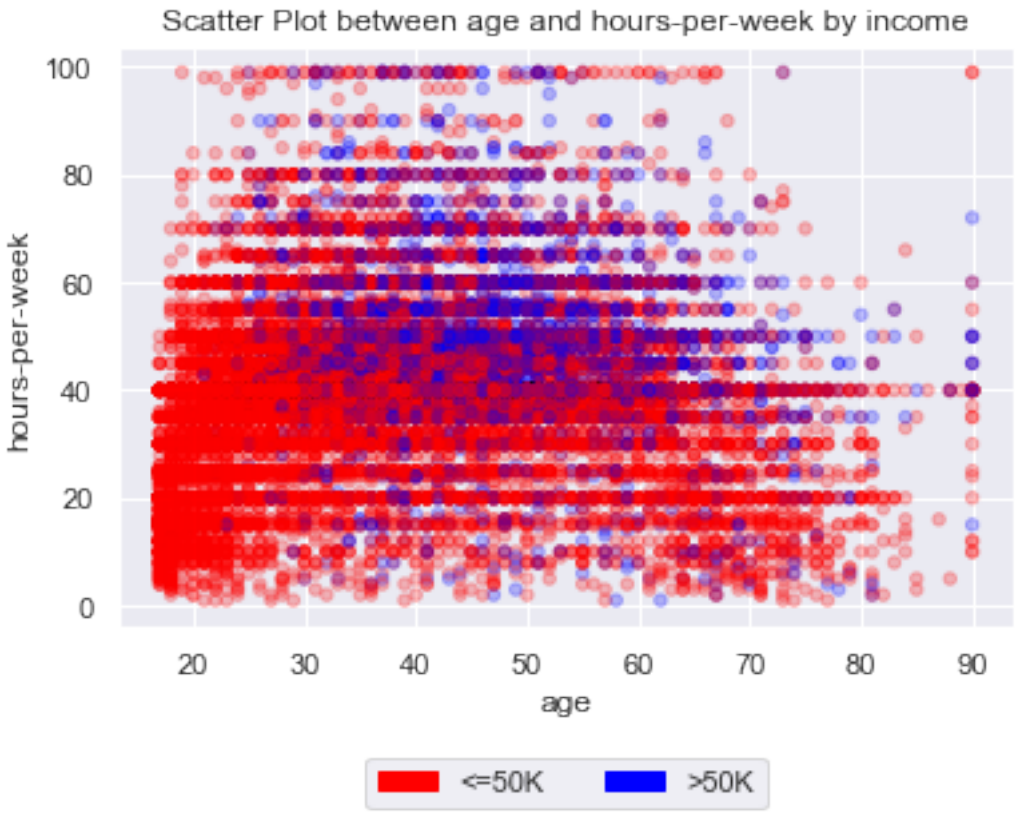
A function named `scatterplotByCategory(c, x, y, D)`is designed to draw a scatterplot between two numeric attributes `y` and `x` labelled by a categorical attribute `c` given an input data `D`. In the case, `D` is `adultData` and `c` is the income level . The codes below show the details of this function. Remark: an alternative would be a scatter matrix. However, we found that the figure size would be too small to fit in this report and hence we decided to "break" up it into following pairwise scatterplots.

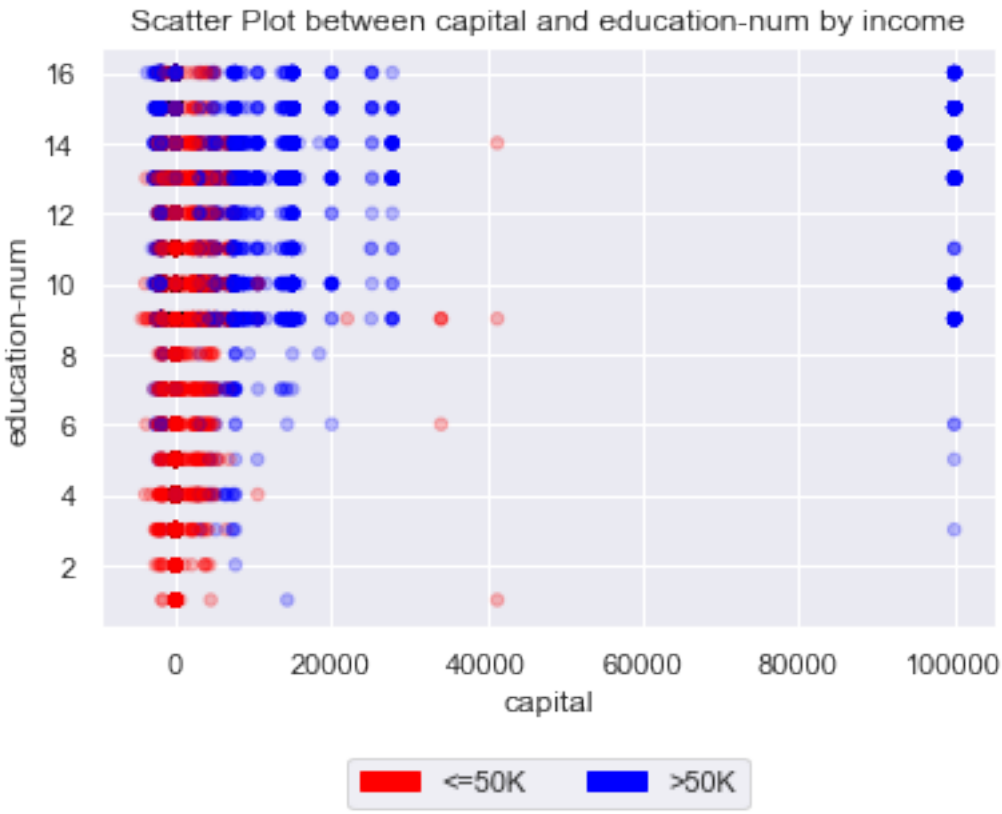
In [28]:
```
import matplotlib.patches as mpatches
def scatterplotByCategory(c, x, y, D):
    data = D
    v = data[c].unique()
    m_1 = data[c] == v[0]
    m_2 = data[c] == v[1]
    data.loc[m_1, c]= 0
    data.loc[m_2, c]= 1
    data[c].value_counts()
    colors_palette = {0: 'red', 1:'blue'}
    colors = [colors_palette[i] for i in data[c]]
    red_patch = mpatches.Patch(color='red', label=v[0])
    blue_patch = mpatches.Patch(color='blue', label=v[1])
    data[[x, y]].plot(kind = 'scatter', x = 0, y = 1, c = colors, alpha = 0.25)
    plt.title('Scatter Plot between ' + x + ' and ' + y + ' by ' + c)
    plt.legend(loc = 9, handles=[red_patch, blue_patch],
    bbox_to_anchor = (0.5, -0.2), ncol = 2)
    plt.show()

    # return the original string values
    n_1 = data[c] == 0
    n_2 = data[c] == 1
    data.loc[n_1, c]= v[0]
    data.loc[n_2, c]= v[1]
```

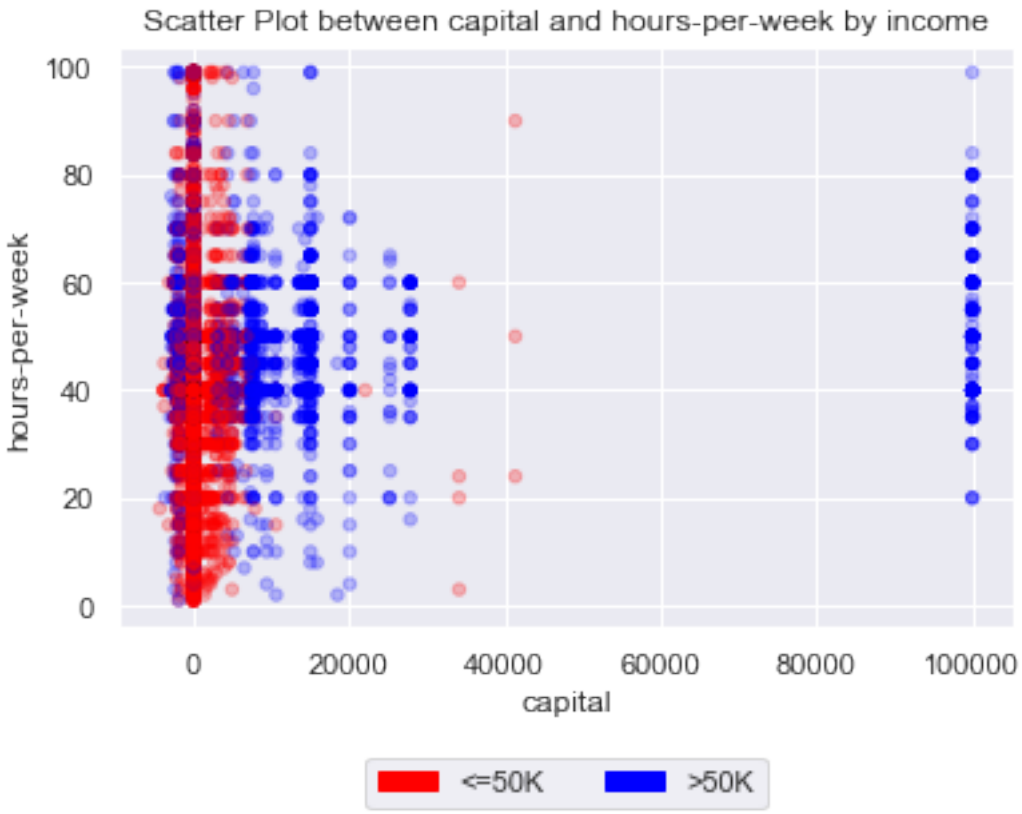
In [29]:
```
for col in ['education-num','capital','hours-per-week']:
    scatterplotByCategory('income', 'age', col , adultData)

for col in ['education-num','hours-per-week']:
    scatterplotByCategory('income', 'capital', col , adultData)

scatterplotByCategory('income', 'hours-per-week', 'education-num' , adultData)
```

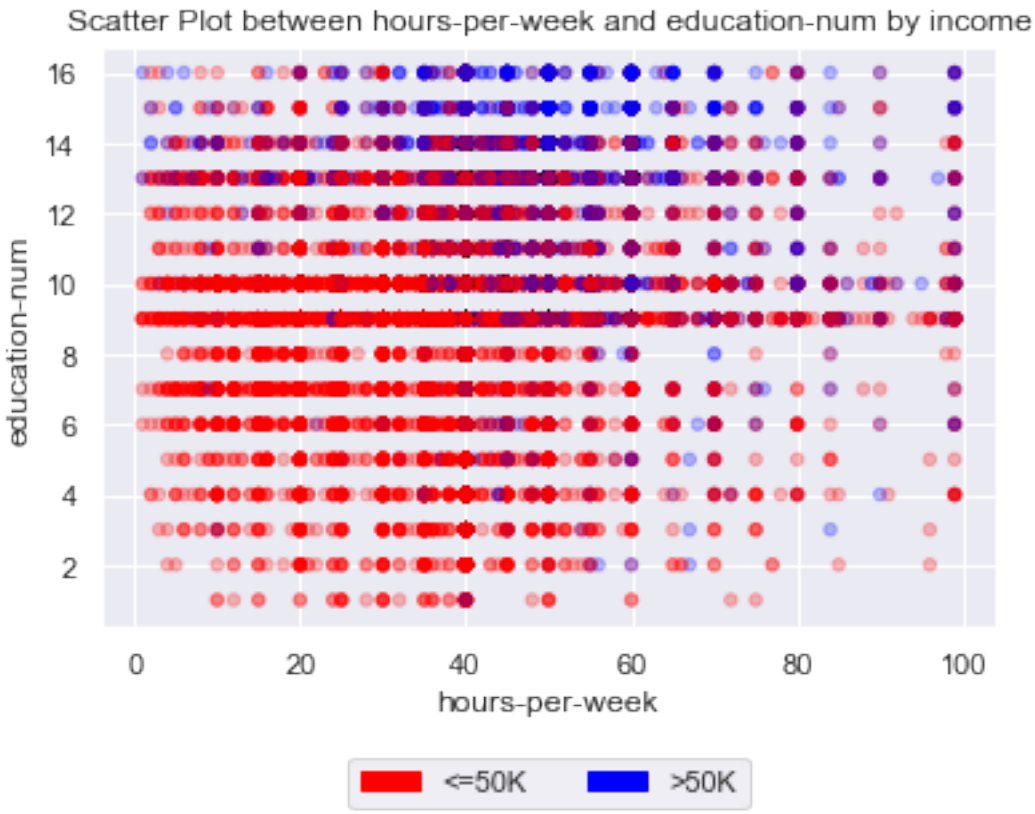

Scatter Plot between age and education-num by income
education-num
16
14
12
10
8
6
4
2
20
30
40
50
60
70
80
90
age
<=50K
>50K


Scatter Plot between age and capital by income
100000
80000
60000
40000
20000
0
capital
20
30
40
50
60
70
80
90
age
<=50K
>50K


Scatter Plot between age and hours-per-week by income
hours-per-week
100
80
60
40
20
0
20
30
40
50
60
70
80
90
age
<=50K
>50K


Scatter Plot between capital and education-num by income
education-num
16
14
12
10
8
6
4
2
0
20000
40000
60000
80000
100000
capital
<=50K
>50K


Scatter Plot between capital and hours-per-week by income
hours-per-week
100
80
60
40
20
0
0
20000
40000
60000
80000
100000
capital
<=50K
>50K

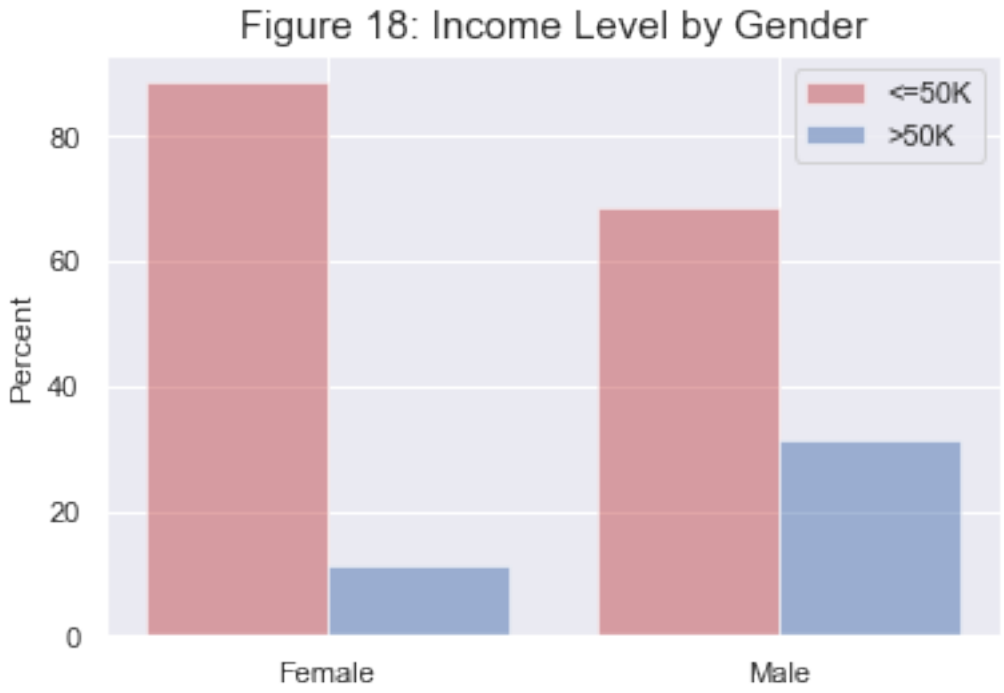


The scatterplots above show no clear correlation between any two numeric variables. Therefore, numeric features are likely to be independent which reduce multicollinearity issues during modelling stage.

3.2.3 Categorical Attributes Segregated by Income Level

```
In [30]: import numpy as np
         yCounts = (adultData.groupby(['sex'])['income']
                             .value_counts(normalize=True)
                             .rename('Percentage')
                             .mul(100)
                             .reset_index())

         data1 = yCounts.loc[yCounts['income']=='<=50K','Percentage']
         data2 = yCounts.loc[yCounts['income']=='>50K','Percentage']

         N = 2
         ind = np.arange(N)  # the x locations for the groups
         width = 0.40        # the width of the bars

         fig, ax = plt.subplots()
         rects1 = ax.bar(ind, data1, width, color='r', alpha = 0.5)
         rects2 = ax.bar(ind + width, data2, width, color='b', alpha = 0.5)

         ax.set_ylabel('Percent')
         ax.set_title("Figure " + str(i) +': Income Level by Gender', fontsize = 15)
```

```
ax.set_xticks(ind + width / 2)
ax.set_xticklabels(('Female','Male'))

ax.legend((rects1[0], rects2[0]), ('<=50K', '>50K'))
plt.show()
i = i + 1
```

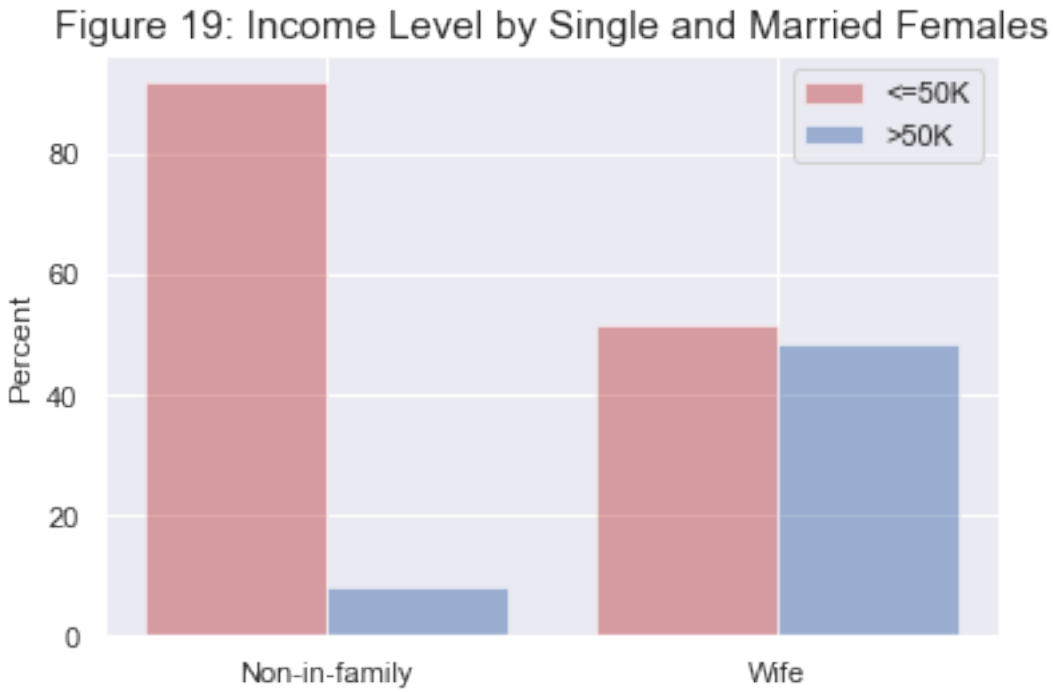


In this dataset, we can see that women are more likely to earn less than 50 thousand dollars a year when compared with men. We can also see that for both men and women, those earning less than 50 thousand dollars out number those earning more.

```
In [31]: family_female_mask = (adultData.relationship.isin(['Not-in-family','Wife'])) & (adultData.sex.
```

```
yCounts = (adultData[family_female_mask].groupby(['relationship'])['income']
                    .value_counts(normalize=True)
                    .rename('Percentage')
                    .mul(100)
                    .reset_index())

data1 = yCounts.loc[yCounts['income']=='<=50K','Percentage']
data2 = yCounts.loc[yCounts['income']=='>50K','Percentage']

N = 2
ind = np.arange(N)  # the x locations for the groups
width = 0.40        # the width of the bars

fig, ax = plt.subplots()
rects1 = ax.bar(ind, data1, width, color='r', alpha = 0.5)
rects2 = ax.bar(ind + width, data2, width, color='b', alpha = 0.5)
```

```
ax.set_ylabel('Percent')
ax.set_title("Figure " + str(i) +': Income Level by Single and Married Females', fontsize = 15)
ax.set_xticks(ind + width / 2)
ax.set_xticklabels(('Non-in-family','Wife'))

ax.legend((rects1[0], rects2[0]), ('<=50K', '>50K'))
plt.show()
i = i +1
```



Single women are much more likely to earn less than 50 thousand dollars a year compared to married women. This does not take age into account though. It may be that younger women make less and are also less likely to be married.

```
In [32]: family_male_mask = (adultData.relationship.isin(['Not-in-family','Husband'])) & (adultData.sex
         yCounts = (adultData[family_male_mask].groupby(['relationship'])['income']
                             .value_counts(normalize=True)
                             .rename('Percentage')
                             .mul(100)
                             .reset_index())

         data1 = yCounts.loc[yCounts['income']=='<=50K','Percentage']
         data2 = yCounts.loc[yCounts['income']=='>50K','Percentage']

         N = 2
         ind = np.arange(N)  # the x locations for the groups
         width = 0.40        # the width of the bars

         fig, ax = plt.subplots()
```

```
rects1 = ax.bar(ind, data1, width, color='r', alpha = 0.5)
rects2 = ax.bar(ind + width, data2, width, color='b', alpha = 0.5)

ax.set_ylabel('Percent')
ax.set_title("Figure " + str(i) +': Income Level by Single and Married Males', fontsize = 12)
ax.set_xticks(ind + width / 2)
ax.set_xticklabels(('Husband','Not-in-family'))

ax.legend((rects1[0], rects2[0]), ('<=50K', '>50K'))
plt.show()
i = i + 1
```

Single and married men have a similar earning distribution to single and married women respectively. Less likely to earn more than 50 thousand dollars for single men compared to married men.

```
In [33]: edu_mask = adultData.education.isin(['Bachelors','HS-grad'])
         yCounts = (adultData[edu_mask].groupby(['education'])['income']
                             .value_counts(normalize=True)
                             .rename('Percentage')
                             .mul(100)
                             .reset_index())

         data1 = yCounts.loc[yCounts['income']=='<=50K','Percentage']
         data2 = yCounts.loc[yCounts['income']=='>50K','Percentage']

         N = 2
         ind = np.arange(N)  # the x locations for the groups
         width = 0.40        # the width of the bars

         fig, ax = plt.subplots()
```

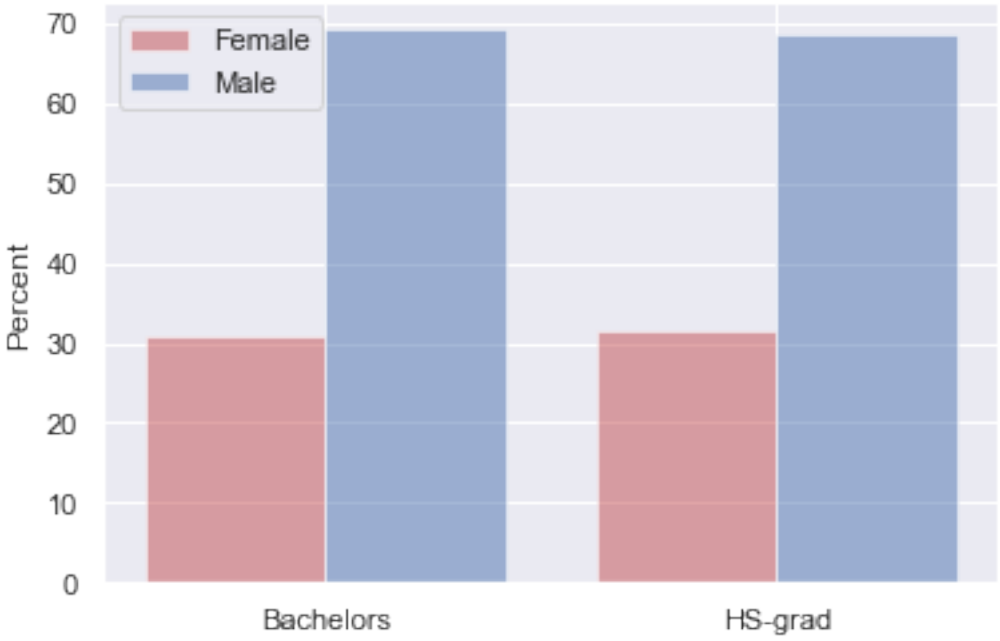
```
rects1 = ax.bar(ind, data1, width, color='r', alpha = 0.5)
rects2 = ax.bar(ind + width, data2, width, color='b', alpha = 0.5)

ax.set_ylabel('Percent')
ax.set_title("Figure " + str(i) + ": Income between Bachelors vs High School Graduation", fonts
ax.set_xticks(ind + width / 2)
ax.set_xticklabels(('Bachelors','HS-grad'))

ax.legend((rects1[0], rects2[0]), ('<=50K', '>50K'))
plt.show()
i = i + 1
```

Both bachelor and high school graduates are more likely than not to earn less than 50 thousand dollars, but bachelor graduates are still more likely to earn more than 50 thousand dollars compared to those who have only completed high school.

```
In [34]: yCounts = (adultData[edu_mask].groupby(['education'])['sex']
                           .value_counts(normalize=True)
                           .rename('Percentage')
                           .mul(100)
                           .reset_index())

         data1 = yCounts.loc[yCounts['sex']=='Female','Percentage']
         data2 = yCounts.loc[yCounts['sex']=='Male','Percentage']

         N = 2
         ind = np.arange(N)  # the x locations for the groups
         width = 0.40        # the width of the bars

         fig, ax = plt.subplots()
```

```
rects1 = ax.bar(ind, data1, width, color='r', alpha = 0.5)
rects2 = ax.bar(ind + width, data2, width, color='b', alpha = 0.5)

ax.set_ylabel('Percent')
ax.set_title("Figure " + str(i) + ": Gender Proportion between Bachelors vs High School Graduat
ax.set_xticks(ind + width / 2)
ax.set_xticklabels(('Bachelors','HS-grad'))

ax.legend((rects1[0], rects2[0]), ('Female', 'Male'))
plt.show()
i = i + 1
```

Figure 22: Gender Proportion between Bachelors vs High School Graduation

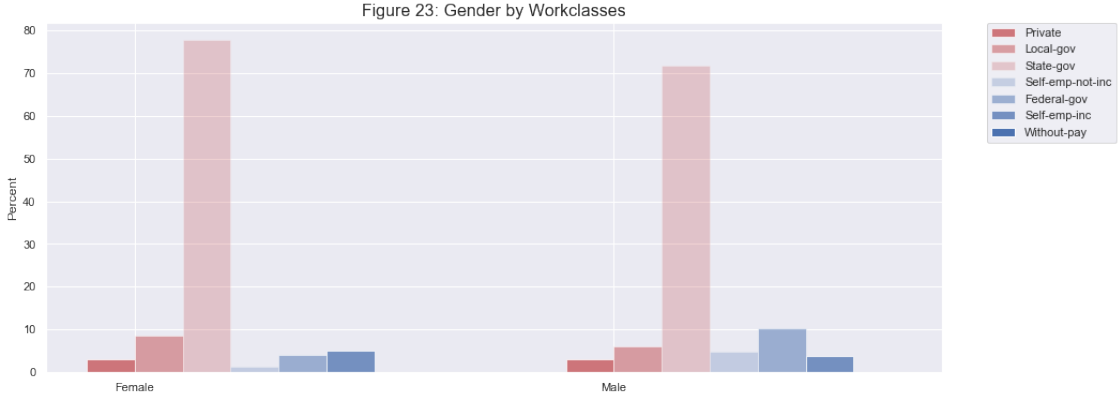


The rates of high school and bachelor completion for men and women are very similar for this dataset.

```
In [35]: yCounts = (adultData.groupby(['sex'])['workclass']
                            .value_counts(normalize=True)
                            .rename('Percentage')
                            .mul(100)
                            .reset_index())

N = 2
ind = np.arange(N)  # the x locations for the groups
width = 0.10       # the width of the bars

data1 = yCounts.loc[yCounts['workclass']=='Federal-gov','Percentage']
data2 = yCounts.loc[yCounts['workclass']=='Local-gov','Percentage']
data3 = yCounts.loc[yCounts['workclass']=='Private','Percentage']
data4 = yCounts.loc[yCounts['workclass']=='Self-emp-inc','Percentage']
data5 = yCounts.loc[yCounts['workclass']=='Self-emp-not-inc','Percentage']
data6 = yCounts.loc[yCounts['workclass']=='State-gov','Percentage']
```

```
data7 = yCounts.loc[yCounts['workclass']=='Without-pay','Percentage']

fig, ax = plt.subplots(figsize=(15, 6))
rects1 = ax.bar(ind, data1, width, color='r', alpha = 0.75)
rects2 = ax.bar(ind + width, data2, width, color='r', alpha = 0.50)
rects3 = ax.bar(ind + width*2, data3, width, color='r', alpha = 0.25)
rects4 = ax.bar(ind + width*3, data4, width, color='b', alpha = 0.25)
rects5 = ax.bar(ind + width*4, data5, width, color='b', alpha = 0.50)
rects6 = ax.bar(ind + width*5, data6, width, color='b', alpha = 0.75)
rects7 = ax.bar(ind + width*6, data7, width, color='b', alpha = 1.0)

ax.set_ylabel('Percent')

ax.set_title("Figure " + str(i) + ": Gender by Workclasses", fontsize = 16)
ax.set_xticks(ind + width / 2)
ax.set_xticklabels(('Female', 'Male'))
plt.legend(yCounts['workclass'].unique(), bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.show()
i = i + 1
```

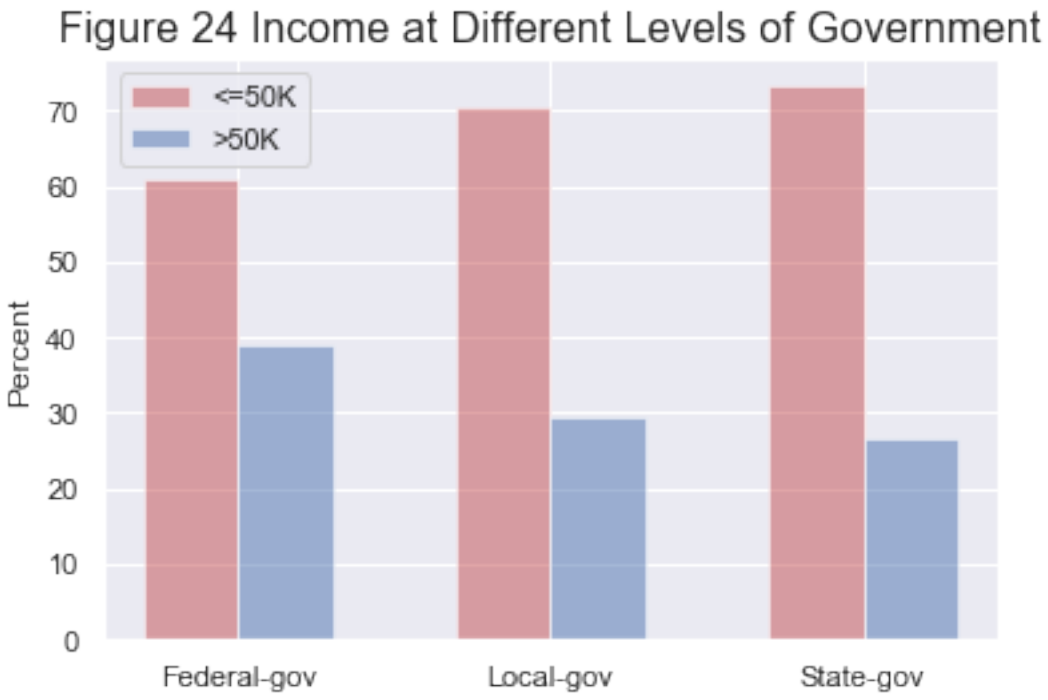
Figure 23: Gender by Workclasses



The private sector is the most popular work category for both males and females, with females being slightly more likely to be in the private sector and less likely to be self employed.

In [36]:
```
gov_mask = adultData['workclass'].isin(['Federal-gov','Local-gov','State-gov'])
yCounts = (adultData[gov_mask].groupby(['workclass'])['income']
                        .value_counts(normalize=True)
                        .rename('Percentage')
                        .mul(100)
                        .reset_index())

N = 3
ind = np.arange(N)  # the x locations for the groups
width = 0.30        # the width of the bars

data1 = yCounts.loc[yCounts['income']=='<=50K','Percentage']
data2 = yCounts.loc[yCounts['income']=='>50K','Percentage']

fig, ax = plt.subplots()
rects1 = ax.bar(ind, data1, width, color='r', alpha = 0.5)
```

```
rects2 = ax.bar(ind + width, data2, width, color='b', alpha = 0.5)

ax.set_ylabel('Percent')
ax.set_title("Figure " + str(i) + " Income at Different Levels of Government", fontsize = 16)
ax.set_xticks(ind + width / 2)
ax.set_xticklabels(('Federal-gov','Local-gov', 'State-gov'))

ax.legend((rects1[0], rects2[0]), ('<=50K', '>50K'))
plt.show()
i = i + 1
```



Focusing just on governments, it appears that the federal government has the highest proportion of people earning more than 50 thousand dollars, followed by the local government and then the state government.

3.2.4 Interaction between Categorical and Numeric Features

In the grouped boxplots segregated by income levels below, those earning more than 50 thousand dollars are older than those earning less and that this pattern is consistent across work class, education level, marital status, occupation, race, sex and relationship type. Those having earned more than 50 thousand dollars a year are also consistently more educated, again taking into account work class, marital status, occupation, race, sex and relationship type. It is unsurprising to see that earning less than 50 thousand dollars a year is correlated with working less hours a week compared those earning more than 50 thousand dollars a year. The boxplot "disappears" between `education-num` and `education` vindicating our initial suspicion that these variables carry the same information. So, we decided to remove `education`.

```
In [37]: plt.rcParams["font.family"] = "DejaVu Sans"
         for col in ['age','education-num', 'hours-per-week', 'capital']:
             for k in ['workclass', 'education', 'marital-status', 'occupation',
```

```
        'race', 'sex', 'relationship']:
    ax = adultData.groupby('income').boxplot(column = col, by = k,
                                             vert = False,
                                             fontsize= 8,
                                             figsize=(15,4.5))
    plt.suptitle("Figure " + str(i) + ": Box Plot of " + col + " grouped by " + k +
                 " and segregated by income levels",
                 fontsize = 12)
    plt.yticks()
    plt.show()
    i = 1 + i
```

Figure 25: Box Plot of age grouped by workclass and segregated by income levels

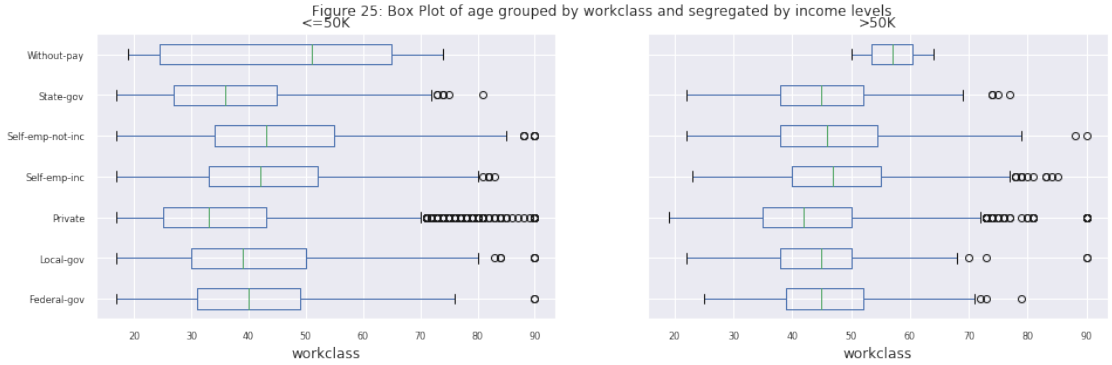


Figure 26: Box Plot of age grouped by education and segregated by income levels

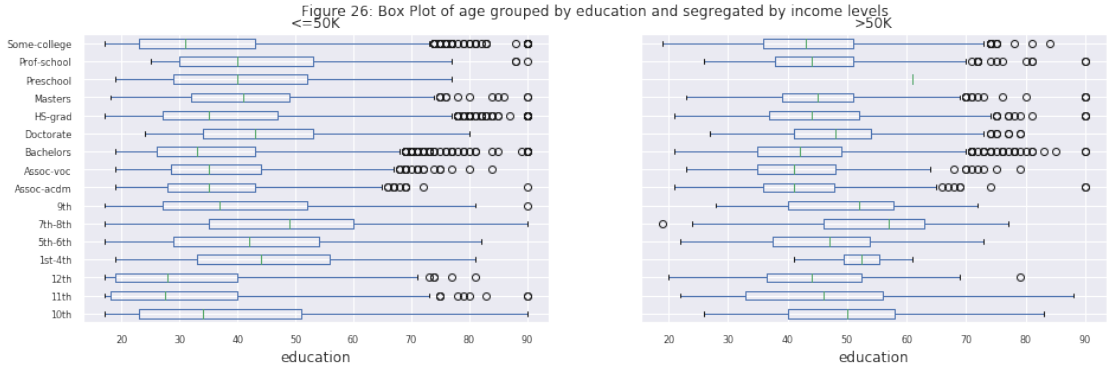


Figure 27: Box Plot of age grouped by marital-status and segregated by income levels

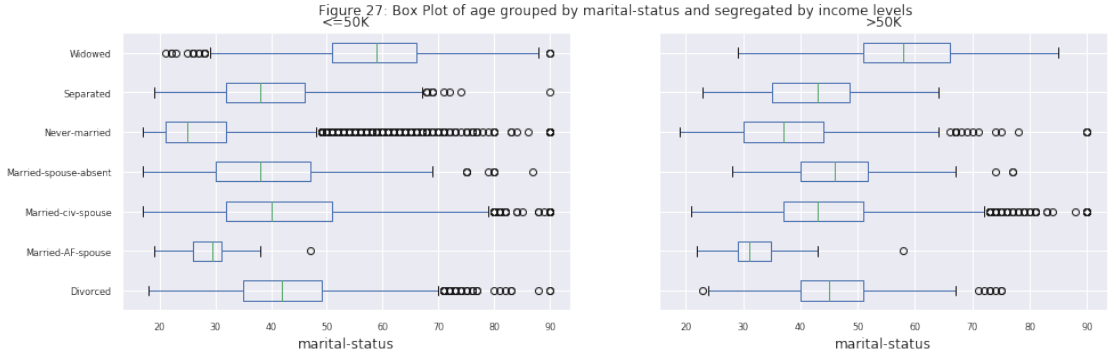


Figure 28: Box Plot of age grouped by occupation and segregated by income levels

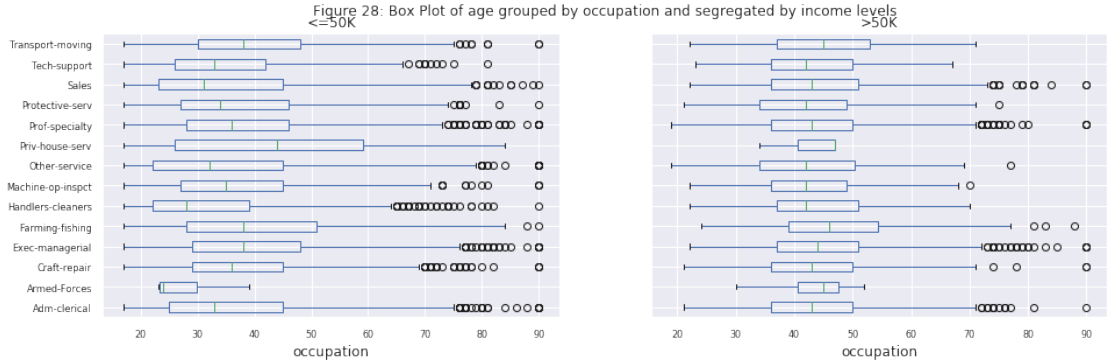


Figure 29: Box Plot of age grouped by race and segregated by income levels

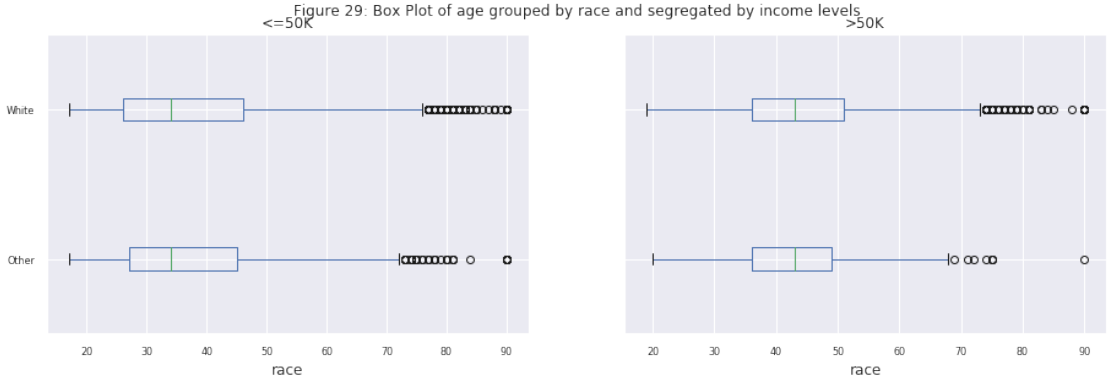


Figure 30: Box Plot of age grouped by sex and segregated by income levels

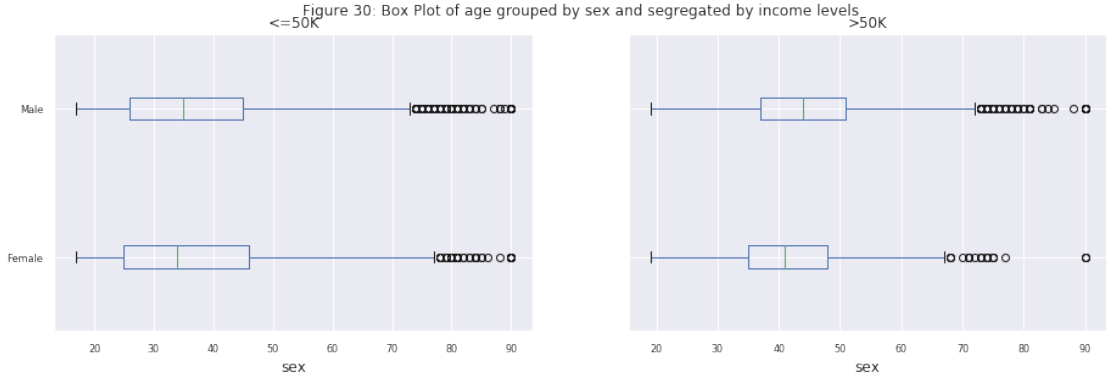


Figure 31: Box Plot of age grouped by relationship and segregated by income levels

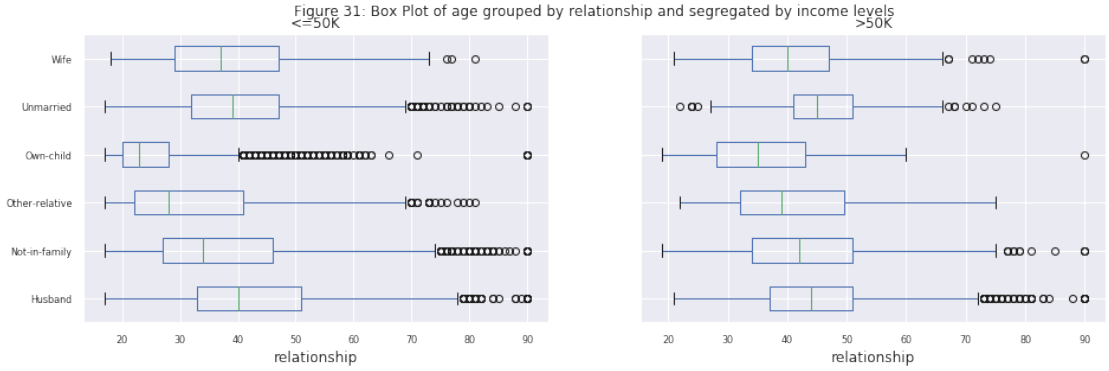


Figure 32: Box Plot of education-num grouped by workclass and segregated by income levels

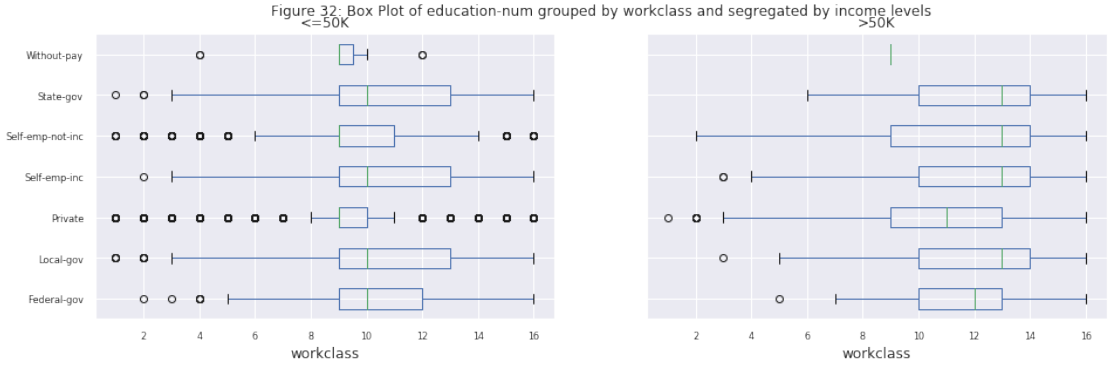


Figure 33: Box Plot of education-num grouped by education and segregated by income levels

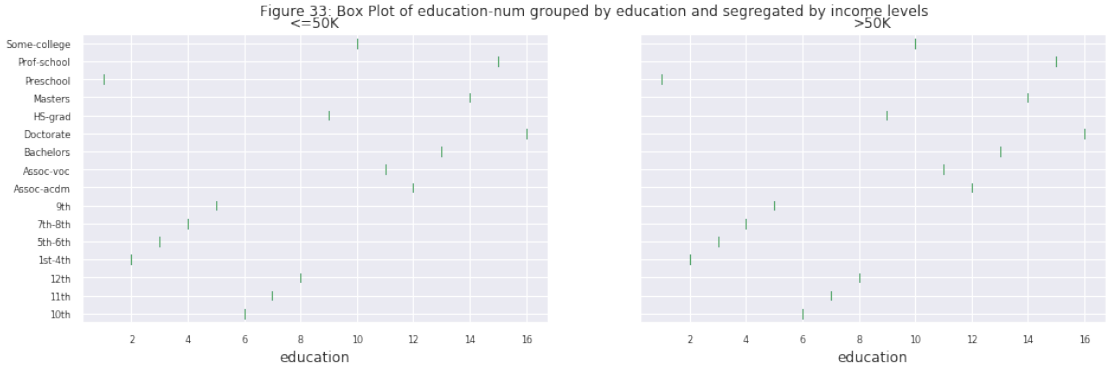


Figure 34: Box Plot of education-num grouped by marital-status and segregated by income levels

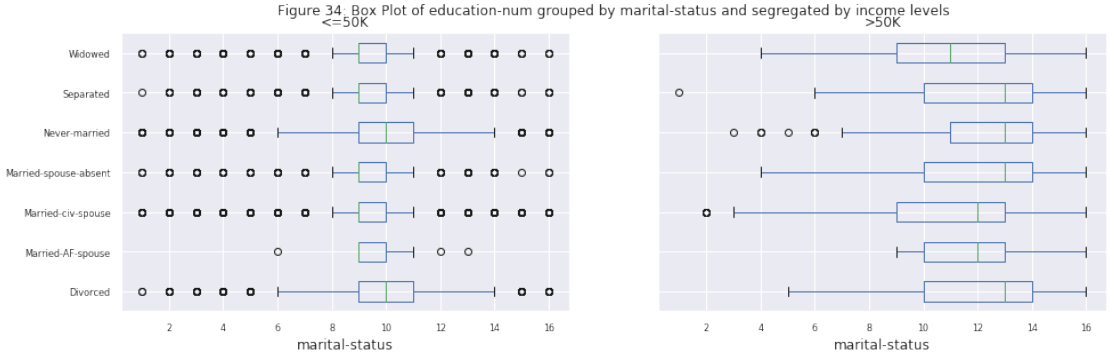


Figure 35: Box Plot of education-num grouped by occupation and segregated by income levels

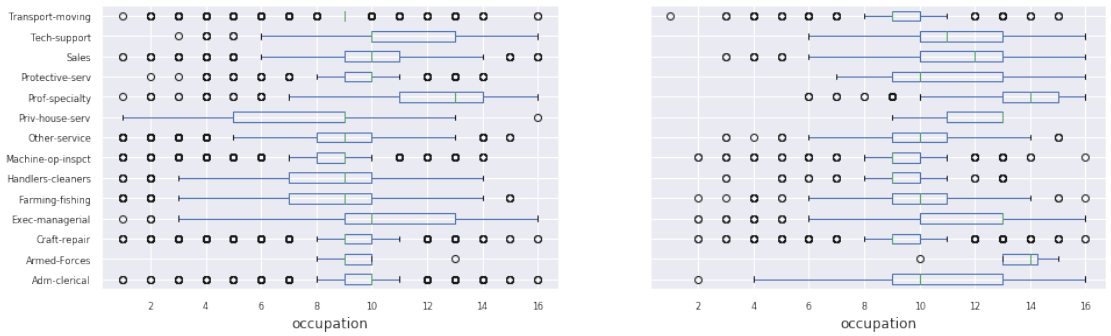


Figure 36: Box Plot of education-num grouped by race and segregated by income levels

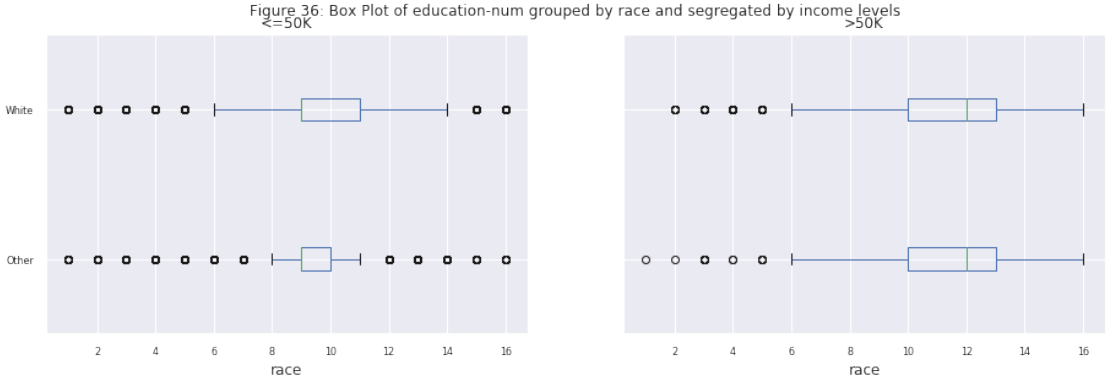


Figure 37: Box Plot of education-num grouped by sex and segregated by income levels

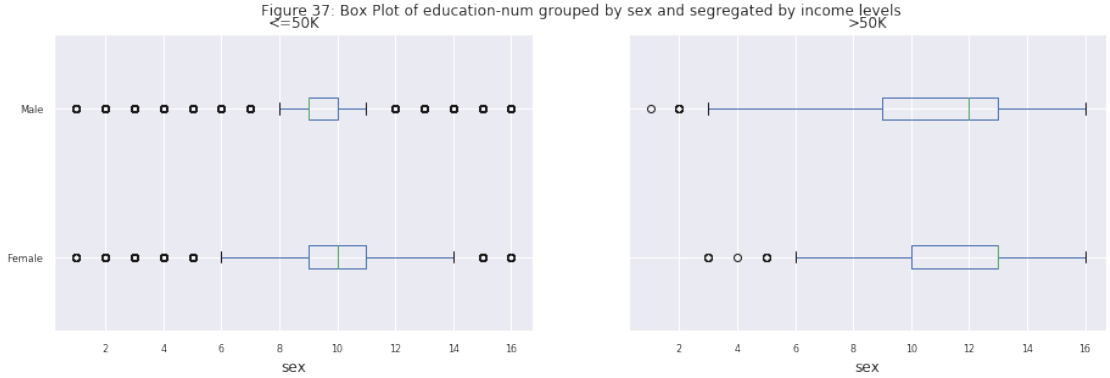


Figure 38: Box Plot of education-num grouped by relationship and segregated by income levels

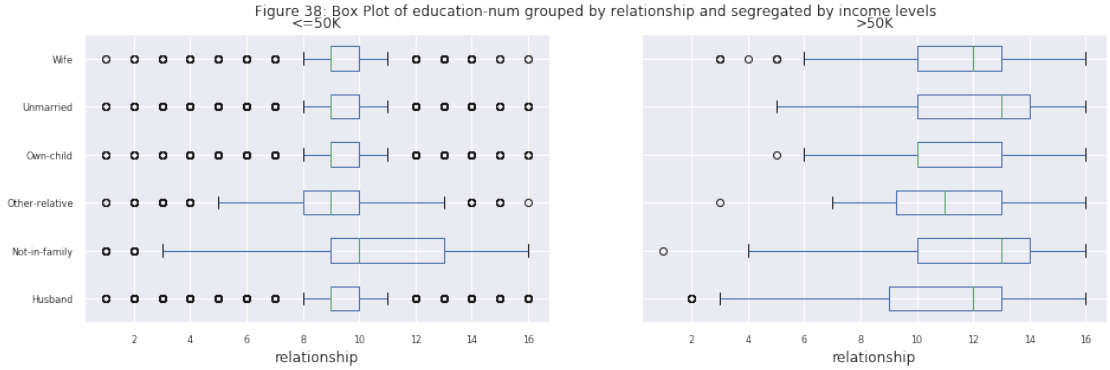


Figure 39: Box Plot of hours-per-week grouped by workclass and segregated by income levels

Figure 40: Box Plot of hours-per-week grouped by education and segregated by income levels

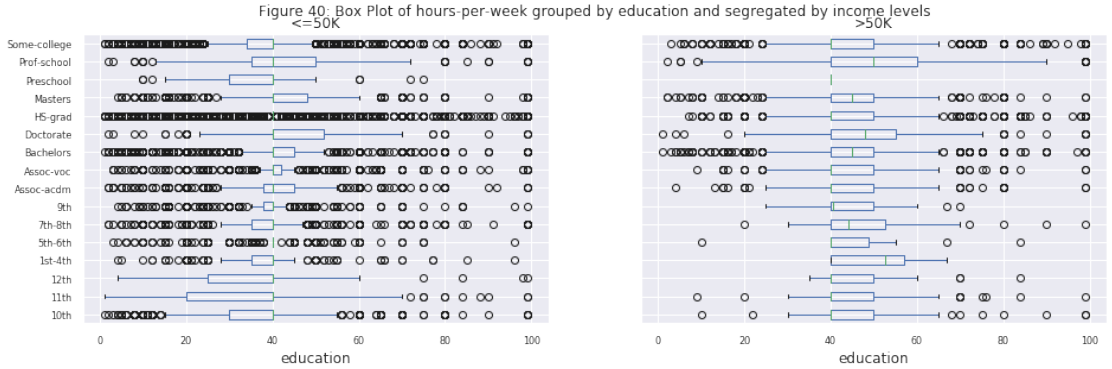


Figure 41: Box Plot of hours-per-week grouped by marital-status and segregated by income levels

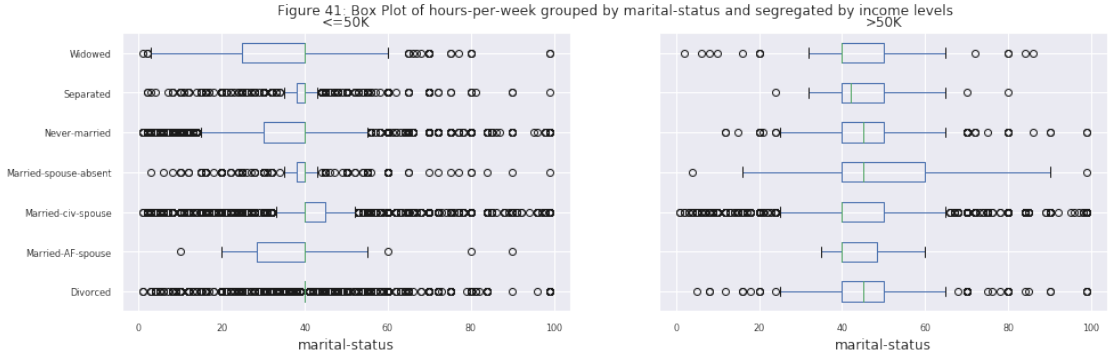


Figure 42: Box Plot of hours-per-week grouped by occupation and segregated by income levels

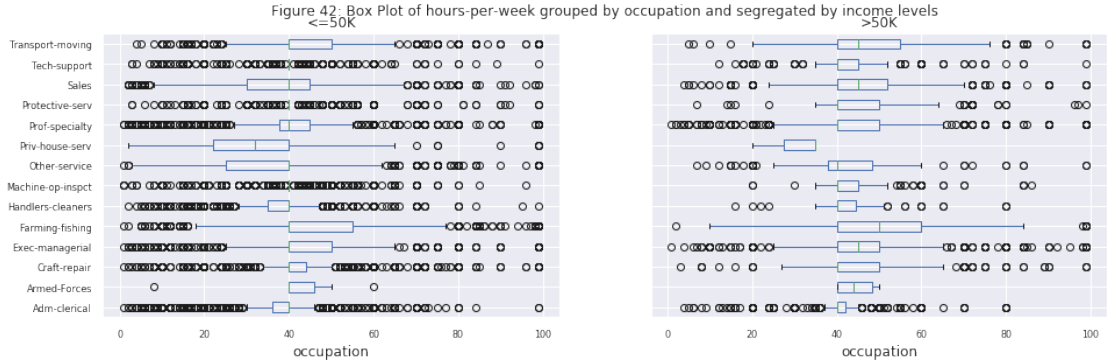


Figure 43: Box Plot of hours-per-week grouped by race and segregated by income levels

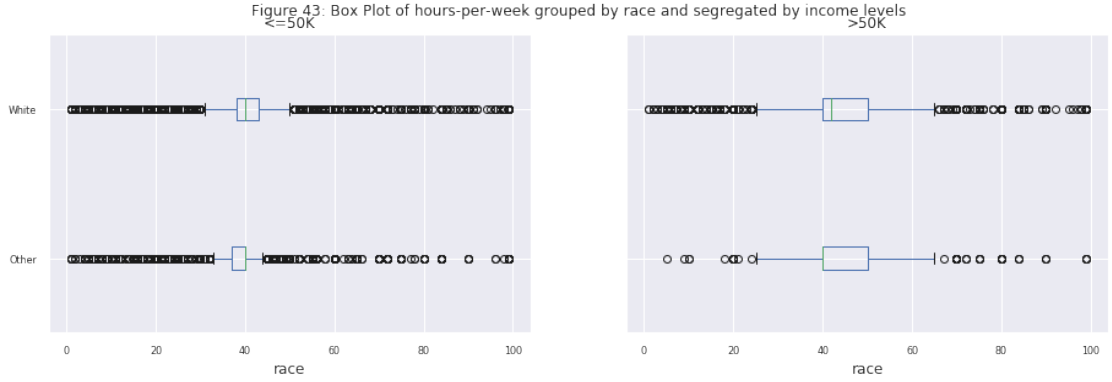


Figure 44: Box Plot of hours-per-week grouped by sex and segregated by income levels

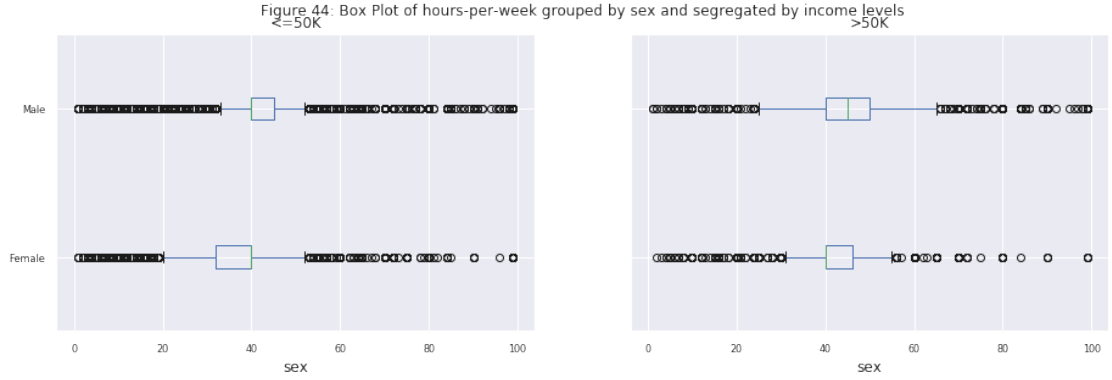


Figure 45: Box Plot of hours-per-week grouped by relationship and segregated by income levels

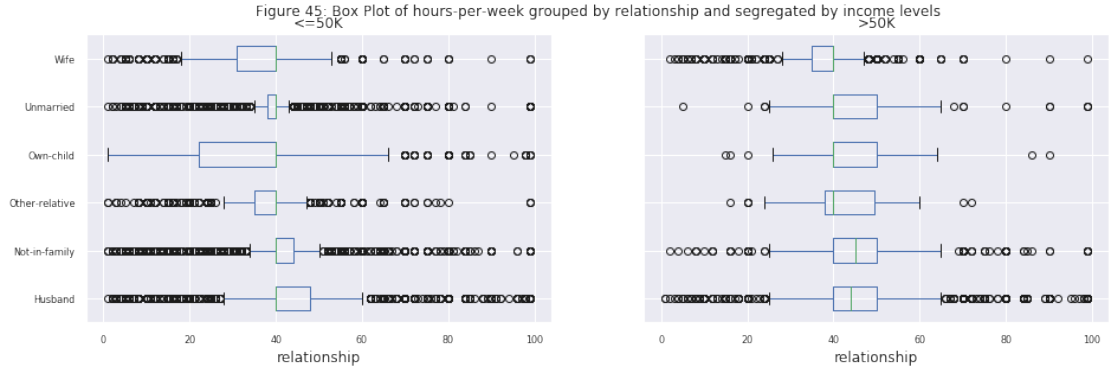


Figure 46: Box Plot of capital grouped by workclass and segregated by income levels

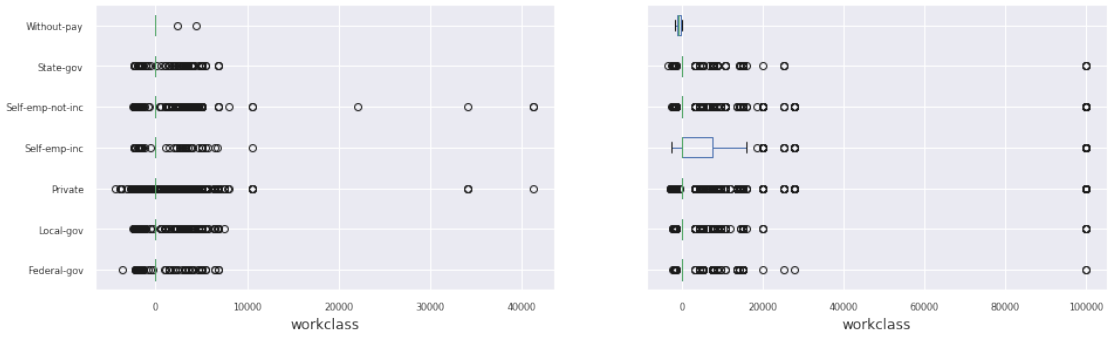


Figure 47: Box Plot of capital grouped by education and segregated by income levels

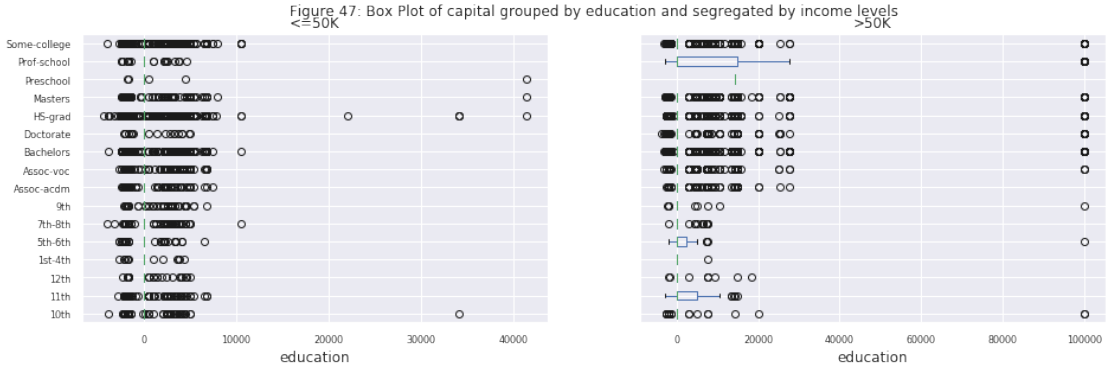


Figure 48: Box Plot of capital grouped by marital-status and segregated by income levels

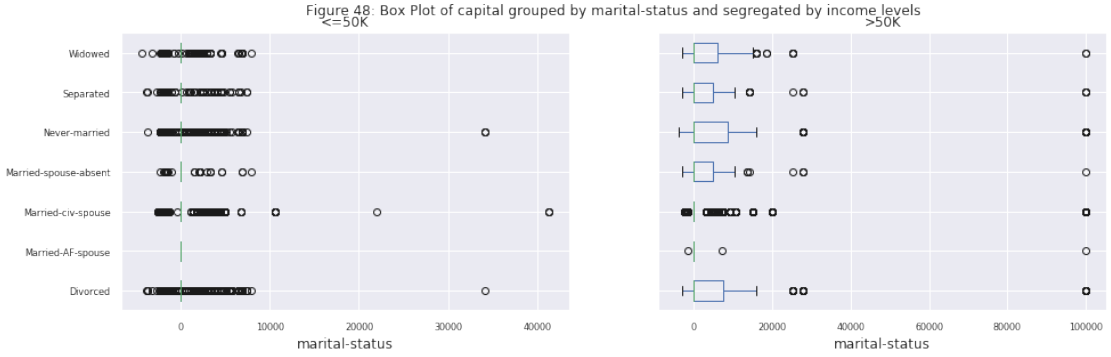


Figure 49: Box Plot of capital grouped by occupation and segregated by income levels

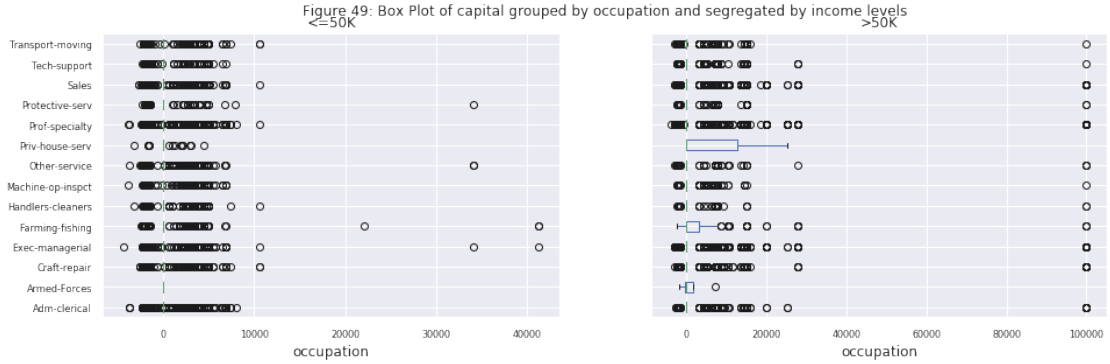


Figure 50: Box Plot of capital grouped by race and segregated by income levels

Figure 51: Box Plot of capital grouped by sex and segregated by income levels

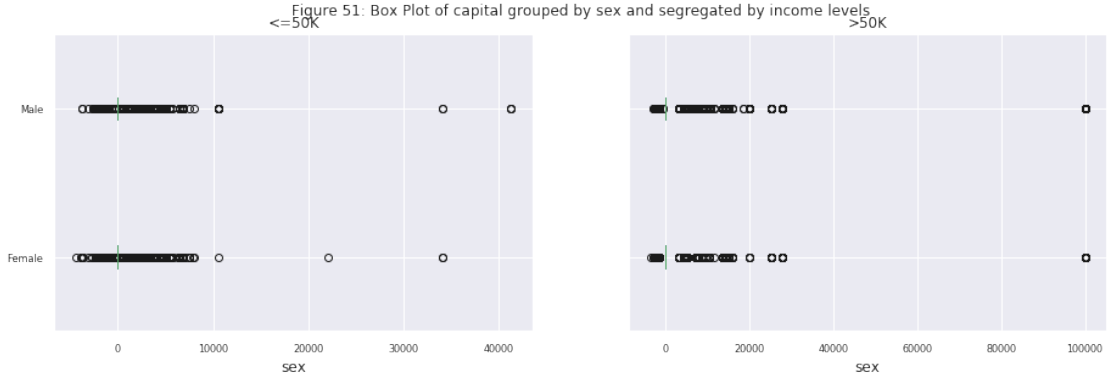
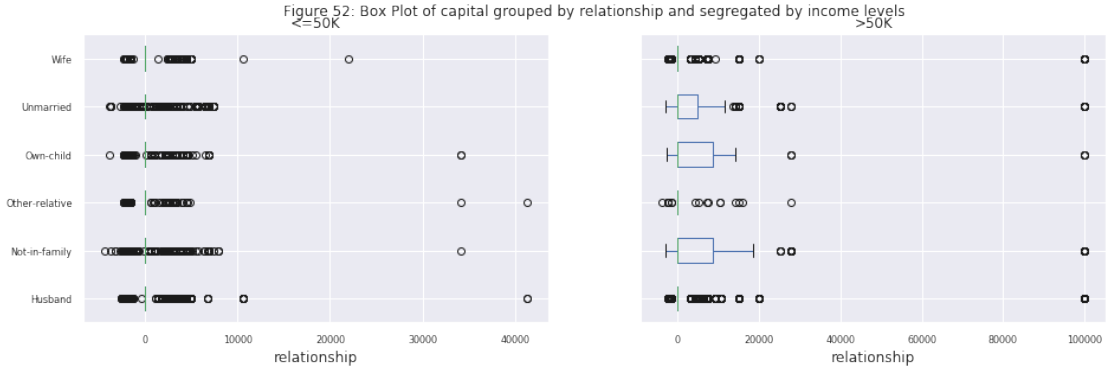


Figure 52: Box Plot of capital grouped by relationship and segregated by income levels




```
In [38]: adultData = adultData.drop(['education'], 1)
```

Chapter 4

Summary

In Phase 1, we removed any observation encoded as `?`. We defined `capital` from `capital_loss` and `capital_gain`. We removed `education` since it carried the same information with `education_num`. We also dropped `fnlwght` - an estimate of the number of units in the target population that the responding unit represents. From the data exploration, we found that education levels, workclasses, gender, ages, and marital statuses were potentially useful features in predicting the income classes.

Bibliography

[1] M. Lichman. UCI Machine Learning Repository: Adult Data Set.

[2] Tax bracket.org.