Curtis Brinker

CS5402

September 11, 2020

HW1

Pre-processing (Steps 1-8):

The pre-processing for steps 1-8 were done in python, primarily with the help of the pandas library. The source code used for the pre-processing is given below, followed by a brief explanation of the code.

# data\_preprocessing

#

# Prepare data from census.csv for analysis

# Imports

**import** pandas **as** pd

**from** datetime **import** datetime

**from** feature\_engine**.**discretisers **import** EqualWidthDiscretiser**,** EqualFrequencyDiscretiser

# Load data

DATA\_SOURCE **=** r'../data/census.csv'

df **=** pd**.**read\_csv**(**DATA\_SOURCE**)**

# Sanitize date format (Step 1)

df**[**'date'**]** **=** pd**.**to\_datetime**(**df**[**'date'**])**

df**[**'date'**]** **=** df**[**'date'**].**apply**(**

**lambda** x**:** x**.**replace**(**year**=**1994**).**date**().**strftime**(**'%m/%d/%Y'**)**

**)**

# Discretisation (Steps 2, 7)

age\_discretiser **=** EqualWidthDiscretiser**(**bins**=**10**,** variables**=[**'age'**])**

hpw\_discretiser **=** EqualFrequencyDiscretiser**(**q**=**5**,** variables**=[**'hours-per-week'**])**

df **=** age\_discretiser**.**fit\_transform**(**df**)**

df **=** hpw\_discretiser**.**fit\_transform**(**df**)**

# Value replacement (Steps 3, 5, 6, 8)

# Workclass: ? -> Other

# Occupation: ? -> Other

# Native-country: ? -> Unspecified

# Sex: Starts with f -> female

# Starts with m -> male

df**[**'workclass'**].**replace**(**to\_replace**=**'\?'**,** value**=**'Other'**,** inplace**=True,** regex**=True)**

df**[**'occupation'**].**replace**(**to\_replace**=**'\?'**,** value**=**'Other'**,** inplace**=True,** regex**=True)**

df**[**'native-country'**].**replace**(**to\_replace**=**'\?'**,** value**=**'Unspecified'**,** inplace**=True,** regex**=True)**

df**[**'sex'**].**replace**([**'^(\W)\*[Ff].\*'**,** '^(\W)\*[Mm].\*'**],** **[**'Female'**,** 'Male'**],** inplace**=True,** regex**=True)**

# Value normalization (Step 4)

# Normalize population-wgt

wgt **=** df**[**'population-wgt'**]**

df**[**'population-wgt'**]** **=** **(**wgt**-**wgt**.**min**())/(**wgt**.**max**()** **-** wgt**.**min**())**

# Write sanitized data to file

SAVE\_LOCATION **=** r'../data/census\_sanitized.csv'

df**.**to\_csv**(**SAVE\_LOCATION**,** index**=False)**

This script groups the pre-processing actions required in steps 1-8 into logical groups and completes each group sequentially. Comments are placed to indicate which steps are being completed in each logical group. When pre-processing is complete, the dataframe is saved to a new file called census\_sanitized.csv.

Chi-squared test (Step 9):

A Chi-squared test was performed on each unique pair of nominal values. This was done in a python script, using both the pandas and scipy library. The source code for the script is given below:

# Imports

**from** scipy**.**stats **import** chi2\_contingency

**from** scipy**.**stats **import** chi2

**from** itertools **import** combinations

**import** pandas **as** pd

**def** is\_dependent**(**df**,** attr1**,** attr2**,** significance**=**0.05**):**

# Returns True if attr1 and attr2 in a specificied

# dataframe are considered dependent using the Chi^2 test

observation **=** create\_observation\_table**(**df**,** attr1**,** attr2**)**

chi**,** pval**,** dof**,** exp **=** chi2\_contingency**(**observation**)**

p **=** 1 **-** significance

critical\_value **=** chi2**.**ppf**(**p**,** dof**)**

**return** **(**chi **>** critical\_value**)**

**def** create\_observation\_table**(**df**,** attr1**,** attr2**):**

# Creates the observation table for two attributes

# in a specified dataframe

# Get unique values for attributes

index **=** df**[**attr1**].**unique**()**

cols **=** df**[**attr2**].**unique**()**

# Sort elements in cols/index

**[**arr**.**sort**()** **for** arr **in** **[**index**,** cols**]]**

# Create empty table

observation **=** pd**.**DataFrame**([],** index**=**index**,** columns**=**cols**)**

# Insert data

**for** idx**,** val **in** df**.**groupby**([**attr1**,** attr2**]).**size**().**items**():**

row**,** col **=** idx

observation**[**col**].**loc**[**row**]** **=** val

observation**.**fillna**(**0**,** inplace**=True)**

**return** observation

**if** \_\_name\_\_ **==** '\_\_main\_\_'**:**

# Read from data source

DATA\_SOURCE **=** r'../data/census\_sanitized.csv'

df **=** pd**.**read\_csv**(**DATA\_SOURCE**)**

# List of all nominal attributes

nominal\_attributes **=** **[**'age'**,**

'workclass'**,**

'education'**,**

'marital-status'**,**

'occupation'**,**

'relationship'**,**

'race'**,**

'sex'**,**

'hours-per-week'**,**

'native-country' **]**

# Iterate through combinations, determine dependence

**for** c **in** combinations**(**nominal\_attributes**,** 2**):**

**print(**f'{str(c[0]) + " & " + str(c[1]):<35}: {is\_dependent(df, \*c)}'**)**

The basic flow of the script is as follows:

1. Generate a unique combination of nominal attributes.

2. Generate the observation table for that pair of attributes

3. Use the observation table to perform a chi-squared analysis

4. Use the results of the analysis to determine if the variables are (or are not) dependent.

The output of this script can be seen below:

age & workclass : True

age & education : True

age & marital-status : True

age & occupation : True

age & relationship : True

age & race : True

age & sex : True

age & hours-per-week : True

age & native-country : True

workclass & education : True

workclass & marital-status : True

workclass & occupation : True

workclass & relationship : True

workclass & race : True

workclass & sex : True

workclass & hours-per-week : True

workclass & native-country : True

education & marital-status : True

education & occupation : True

education & relationship : True

education & race : True

education & sex : True

education & hours-per-week : True

education & native-country : True

marital-status & occupation : True

marital-status & relationship : True

marital-status & race : True

marital-status & sex : True

marital-status & hours-per-week : True

marital-status & native-country : True

occupation & relationship : True

occupation & race : True

occupation & sex : True

occupation & hours-per-week : True

occupation & native-country : True

relationship & race : True

relationship & sex : True

relationship & hours-per-week : True

relationship & native-country : True

race & sex : True

race & hours-per-week : True

race & native-country : True

sex & hours-per-week : True

sex & native-country : True

hours-per-week & native-country : True

Where 'True' indicates that the pair of attributes are dependent on each other. Thus we see that **no** pair of nominal attributes are independent from each other. (Or that is to say they are all dependent on each other.)

Spearman test (Step 10):

A Spearman test was performed on each unique pair of non-nominal values. This was done in a python script, using both the pandas and scipy library. The source code for the script is given below:

# Imports

**from** scipy**.**stats **import** spearmanr

**from** itertools **import** combinations

**from** datetime **import** datetime

**import** pandas **as** pd

**def** is\_dependent**(**df**,** attr1**,** attr2**,** threshold**=**0.8**):**

# Uses spearman test to check if two attributes in the

# specified dataframe are dependent.

X**,** Y **=** df**[**attr1**],** df**[**attr2**]**

corr**,** pvalue **=** spearmanr**(**X**,** Y**)**

# Attributes are likely dependent if >= threshold

**return** abs**(**corr**)** **>=** threshold

**if** \_\_name\_\_ **==** '\_\_main\_\_'**:**

# Read from data source

DATA\_SOURCE **=** r'../data/census\_sanitized.csv'

df **=** pd**.**read\_csv**(**DATA\_SOURCE**)**

# Read date as datetime object using MM/DD/YYYY format, convert to timestamp

df**[**'date-timestamp'**]** **=** df**[**'date'**].**apply**(**

**lambda** x**:** datetime**.**strptime**(**x**,** '%m/%d/%Y'**).**timestamp**()**

**)**

# We change 'date' to 'date-timestamp' so that the date can be

# considered a continious number

nonnominal\_attributes **=** **[**'date-timestamp'**,** 'population-wgt'**,**

'education-num'**,** 'capital-gain'**,**

'capital-loss'**]**

# Iterate through combinations, determine dependence

**for** c **in** combinations**(**nonnominal\_attributes**,** 2**):**

X**,** Y **=** df**[**c**[**0**]],** df**[**c**[**1**]]**

**print(**f'{str(c[0]) + " & " + str(c[1]):<35}: {is\_dependent(df, \*c)}'**)**

The basic flow of the script is as follows:

1. Temporarily convert the 'date' values into timestamps, so they can be used as a continuous value

2. Generate a unique combination of non-nominal attributes.

3. Get the values in the columns for both attributes, and store to X and Y

4. Use these values to perform a Spearman test

4. Use the results of the test to determine if the variables are (or are not) dependent.

The output of the script can be seen below:

date-timestamp & population-wgt : False

date-timestamp & education-num : False

date-timestamp & capital-gain : False

date-timestamp & capital-loss : False

population-wgt & education-num : False

population-wgt & capital-gain : False

population-wgt & capital-loss : False

education-num & capital-gain : False

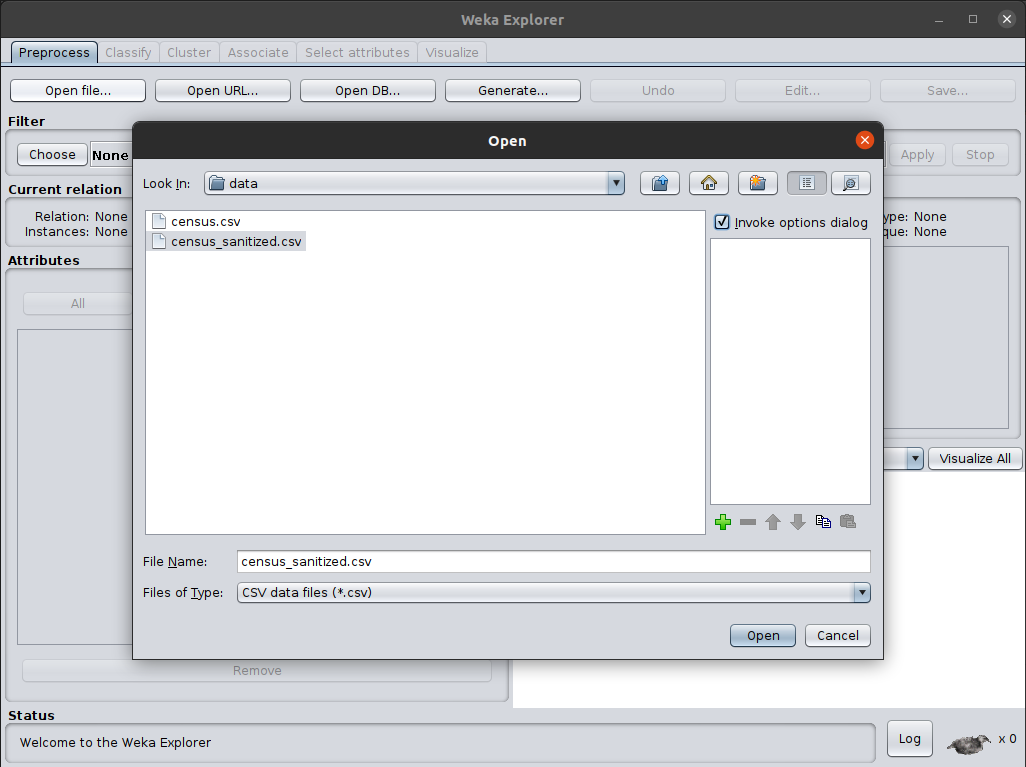
education-num & capital-loss : False

capital-gain & capital-loss : False

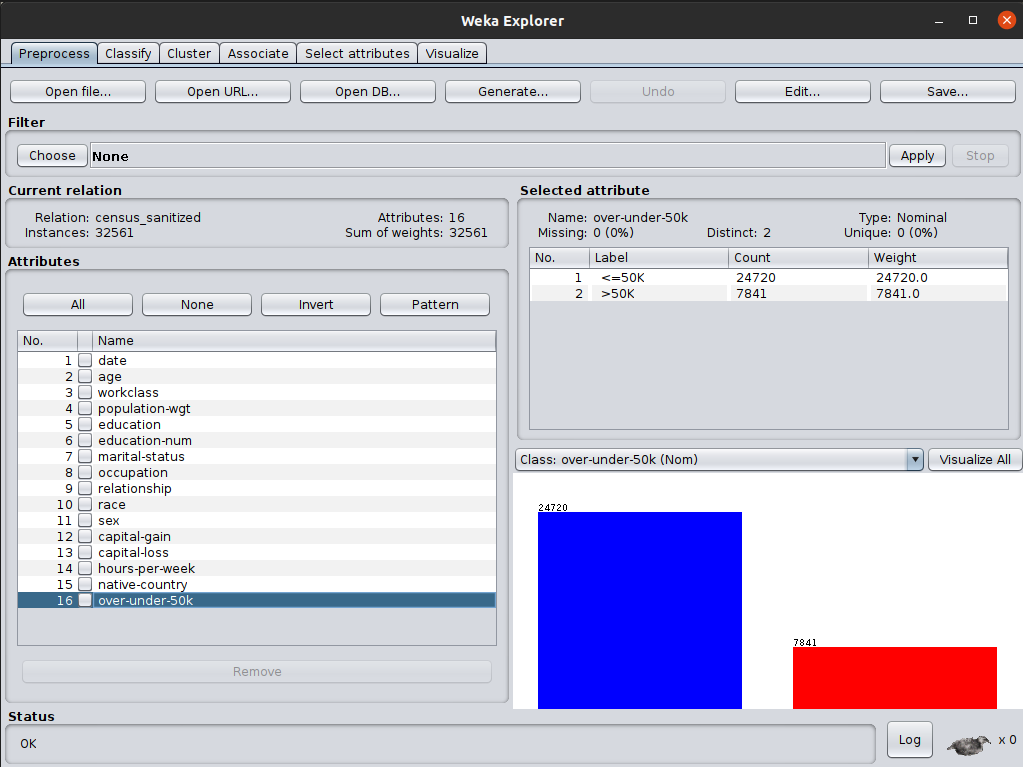
Where 'True' indicates that the pair of attributes are dependent on each other. Thus we see that **all** pairs of non-nominal attributes are independent from each other. (Or that is to say no pairs are dependent on each other.)

PCA Analysis (Step 11):

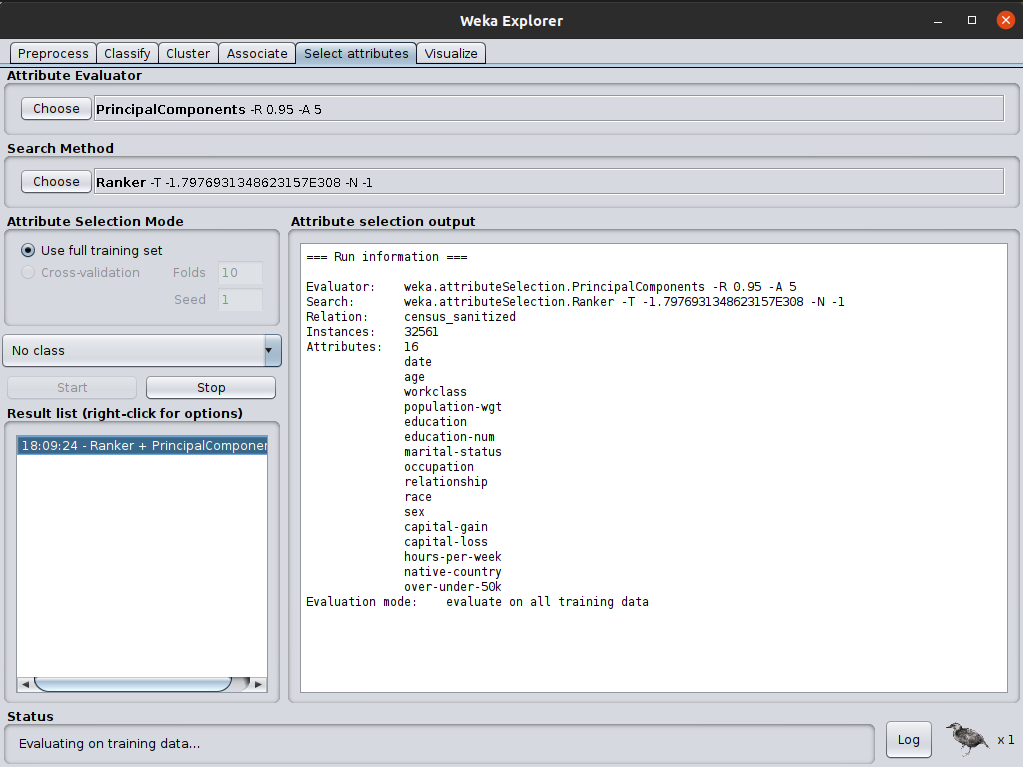
Weka was used to perform a PCA analysis on the dataset to determine the most important non-decision attributes of the dataset. Some screenshots performing the PCA analysis is shown below:



*Loading the Dataset into Weka*

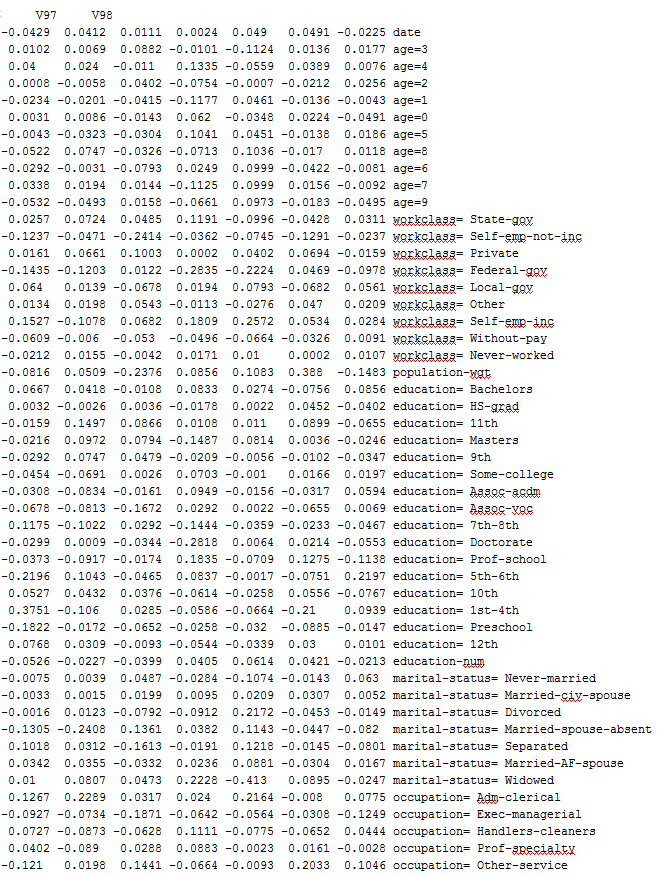


*Dataset Loaded into Weka*



*Running the PCA Analysis Using Weka*

The output of the PCA analysis is much too large to be properly displayed in this document. A small portion of the output is shown below:



A dump of the full output can be accessed at <https://controlc.com/3daef850>, using the password 'CS5402' (without quotes.)

After analyzing the eigenvectors from Weka's PCA analysis, we determine the following 9 attribute are the 'most important'

1. 'date'
2. 'age'
3. 'workclass'
4. 'population-wgt'
5. 'education'
6. 'education-num'
7. 'marital-status'
8. 'occupation'
9. 'relationship'