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)
```

```python
# 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.

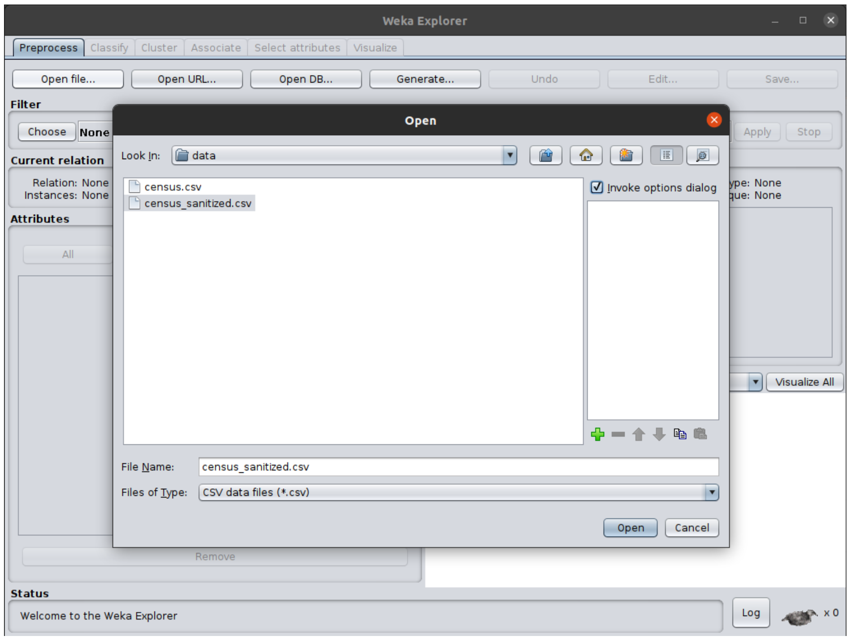
The output of the script can be seen below:

```
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
```

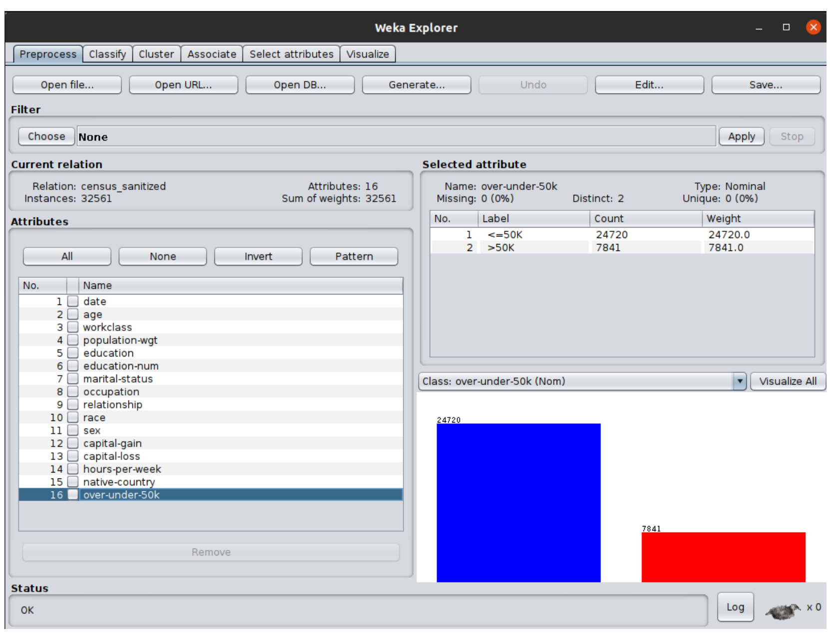
Where 'True' indicates that the pair of attributes are dependent on each other. Thus we see that **all** pairs of non-nominal attributes are independent from each other. (Or that is to say no pairs are dependent on each other.)

<u>PCA Analysis (Step 11):</u>

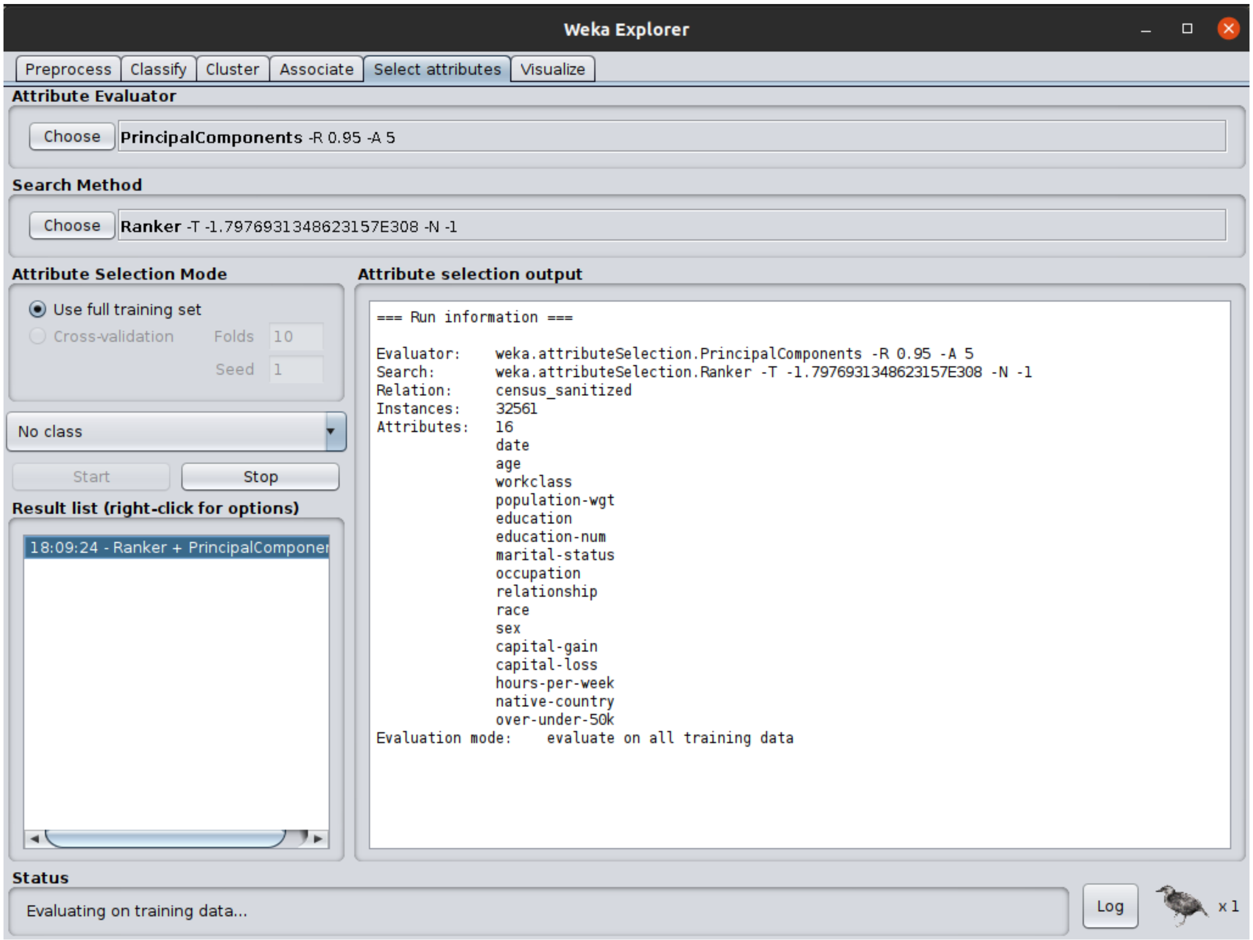
Weka was used to perform a PCA analysis on the dataset to determine the most important non-decision attributes of the dataset. Some screenshots performing the PCA analysis is shown below:



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:

```
     V97     V98
-0.0429  0.0412  0.0111  0.0024  0.049   0.0491 -0.0225 date
 0.0102  0.0069  0.0882 -0.0101 -0.1124  0.0136  0.0177 age=3
 0.04    0.024  -0.011   0.1335 -0.0559  0.0389  0.0076 age=4
 0.0008 -0.0058  0.0402 -0.0754 -0.0007 -0.0212  0.0256 age=2
-0.0234 -0.0201 -0.0415 -0.1177  0.0461 -0.0136 -0.0043 age=1
 0.0031  0.0086 -0.0143  0.062  -0.0348  0.0224 -0.0491 age=0
-0.0043 -0.0323 -0.0304  0.1041  0.0451 -0.0138  0.0186 age=5
-0.0522  0.0747 -0.0326 -0.0713  0.1036 -0.017   0.0118 age=8
-0.0292 -0.0031 -0.0793  0.0249  0.0999 -0.0422 -0.0081 age=6
 0.0338  0.0194  0.0144 -0.1125  0.0999  0.0156 -0.0092 age=7
-0.0532 -0.0493  0.0158 -0.0661  0.0973 -0.0183 -0.0495 age=9
 0.0257  0.0724  0.0485  0.1191 -0.0996 -0.0428  0.0311 workclass= State-gov
-0.1237 -0.0471 -0.2414 -0.0362 -0.0745 -0.1291 -0.0237 workclass= Self-emp-not-inc
 0.0161  0.0661  0.1003  0.0002  0.0402  0.0694 -0.0159 workclass= Private
-0.1435 -0.1203  0.0122 -0.2835 -0.2224  0.0469 -0.0978 workclass= Federal-gov
 0.064   0.0139 -0.0678  0.0194  0.0793 -0.0682  0.0561 workclass= Local-gov
 0.0134  0.0198  0.0543 -0.0113 -0.0276  0.047   0.0209 workclass= Other
 0.1527 -0.1078  0.0682  0.1809  0.2572  0.0534  0.0284 workclass= Self-emp-inc
-0.0609 -0.006  -0.053  -0.0496 -0.0664 -0.0326  0.0091 workclass= Without-pay
-0.0212  0.0155 -0.0042  0.0171  0.01    0.0002  0.0107 workclass= Never-worked
-0.0816  0.0509 -0.2376  0.0856  0.1083  0.388  -0.1483 population-wgt
 0.0667  0.0418 -0.0108  0.0833  0.0274 -0.0756  0.0856 education= Bachelors
 0.0032 -0.0026  0.0036 -0.0178  0.0022  0.0452 -0.0402 education= HS-grad
-0.0159  0.1497  0.0866  0.0108  0.011   0.0899 -0.0655 education= 11th
-0.0216  0.0972  0.0794 -0.1487  0.0814  0.0036 -0.0246 education= Masters
-0.0292  0.0747  0.0479 -0.0209 -0.0056 -0.0102 -0.0347 education= 9th
-0.0454 -0.0691  0.0026  0.0703 -0.001   0.0166  0.0197 education= Some-college
-0.0308 -0.0834 -0.0161  0.0949 -0.0156 -0.0317  0.0594 education= Assoc-acdm
-0.0678 -0.0813 -0.1672  0.0292  0.0022 -0.0655  0.0069 education= Assoc-voc
 0.1175 -0.1022  0.0292 -0.1444 -0.0359 -0.0233 -0.0467 education= 7th-8th
-0.0299  0.0009 -0.0344 -0.2818  0.0064  0.0214 -0.0553 education= Doctorate
-0.0373 -0.0917 -0.0174  0.1835 -0.0709  0.1275 -0.1138 education= Prof-school
-0.2196  0.1043 -0.0465  0.0837 -0.0017 -0.0751  0.2197 education= 5th-6th
 0.0527  0.0432  0.0376 -0.0614 -0.0258  0.0556 -0.0767 education= 10th
 0.3751 -0.106   0.0285 -0.0586 -0.0664 -0.21    0.0939 education= 1st-4th
-0.1822 -0.0172 -0.0652 -0.0258 -0.032  -0.0885 -0.0147 education= Preschool
 0.0768  0.0309 -0.0093 -0.0544 -0.0339  0.03    0.0101 education= 12th
-0.0526 -0.0227 -0.0399  0.0405  0.0614  0.0421 -0.0213 education-num
-0.0075  0.0039  0.0487 -0.0284 -0.1074 -0.0143  0.063  marital-status= Never-married
-0.0033  0.0015  0.0199  0.0095  0.0209  0.0307  0.0052 marital-status= Married-civ-spouse
-0.0016  0.0123 -0.0792 -0.0912  0.2172 -0.0453 -0.0149 marital-status= Divorced
-0.1305 -0.2408  0.1361  0.0382  0.1143 -0.0447 -0.082  marital-status= Married-spouse-absent
 0.1018  0.0312 -0.1613 -0.0191  0.1218 -0.0145 -0.0801 marital-status= Separated
 0.0342  0.0355 -0.0332  0.0236  0.0881 -0.0304  0.0167 marital-status= Married-AF-spouse
 0.01    0.0807  0.0473  0.2228 -0.413   0.0895 -0.0247 marital-status= Widowed
 0.1267  0.2289  0.0317  0.024   0.2164 -0.008   0.0775 occupation= Adm-clerical
-0.0927 -0.0734 -0.1871 -0.0642 -0.0564 -0.0308 -0.1249 occupation= Exec-managerial
 0.0727 -0.0873 -0.0628  0.1111 -0.0775 -0.0652  0.0444 occupation= Handlers-cleaners
 0.0402 -0.089   0.0288  0.0883 -0.0023  0.0161 -0.0028 occupation= Prof-specialty
-0.121   0.0198  0.1441 -0.0664 -0.0093  0.2033  0.1046 occupation= Other-service
```

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'