303 rows x 14 columns

About this file

Data Set Information:

This database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland database is the only one that has been used by ML researchers to this date. The "goal" field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4. Experiments with the Cleveland database have concentrated on simply attempting to distinguish presence (values 1,2,3,4) from absence (value 0).

The names and social security numbers of the patients were recently removed from the database, replaced with dummy values.

One file has been "processed", that one containing the Cleveland database. All four unprocessed files also exist in this directory.

To see Test Costs (donated by Peter Turney), please see the folder "Costs"

Attribute Information:

Only 14 attributes used: 1. #3 (age) 2. #4 (sex) 3. #9 (cp) 4. #10 (trestbps) 5. #12 (chol) 6. #16 (fbs) 7. #19 (restecg) 8. #32 (thalach) 9. #38 (exang) 10. #40 (oldpeak) 11. #41 (slope) 12. #44 (ca) 13. #51 (thal) 14. #58 (num) (the predicted attribute)

Complete attribute documentation:

1 id: patient identification number

2 ccf: social security number (I replaced this with a dummy value of 0)

3 age: age in years

4 sex: sex (1 = male; 0 = female)

5 painloc: chest pain location (1 = substernal; 0 = otherwise)

6 painexer (1 = provoked by exertion; 0 = otherwise)

7 relrest (1 = relieved after rest; 0 = otherwise)

8 pncaden (sum of 5, 6, and 7)

9 cp: chest pain type -- Value 1: typical angina -- Value 2: atypical angina -- Value 3: non-anginal pain -- Value 4: asymptomatic

10 trestbps: resting blood pressure (in mm Hg on admission to the hospital)

11 htn

12 chol: serum cholestoral in mg/dl

13 smoke: I believe this is 1 = yes; 0 = no (is or is not a smoker)

14 cigs (cigarettes per day)

15 years (number of years as a smoker)

16 fbs: (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)

17 dm (1 = history of diabetes; 0 = no such history)

18 famhist: family history of coronary artery disease (1 = yes; 0 = no)

19 restecg: resting electrocardiographic results -- Value 0: normal -- Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV) -- Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria

20 ekgmo (month of exercise ECG reading)

21 ekgday(day of exercise ECG reading)

22 ekgyr (year of exercise ECG reading)

23 dig (digitalis used furing exercise ECG: 1 = yes; 0 = no)

24 prop (Beta blocker used during exercise ECG: 1 = yes; 0 = no)

25 nitr (nitrates used during exercise ECG: 1 = yes; 0 = no)

26 pro (calcium channel blocker used during exercise ECG: 1 = yes; 0 = no)

27 diuretic (diuretic used used during exercise ECG: 1 = yes; 0 = no)

28 proto: exercise protocol 1 = Bruce 2 = Kottus 3 = McHenry 4 = fast Balke 5 = Balke 6 = Noughton 7 = bike 150 kpa min/min (Not sure if "kpa min/min" is what was written!) 8 = bike 125 kpa min/min 9 = bike 100 kpa min/min 10 = bike 75 kpa min/min 11 = bike 50 kpa min/min 12 = arm ergometer

29 thaldur: duration of exercise test in minutes

30 thaltime: time when ST measure depression was noted

31 met: mets achieved

32 thalach: maximum heart rate achieved

33 thalrest: resting heart rate

34 tpeakbps: peak exercise blood pressure (first of 2 parts)

35 tpeakbpd: peak exercise blood pressure (second of 2 parts)

36 dummy

37 trestbpd: resting blood pressure

38 exang: exercise induced angina (1 = yes; 0 = no)

39 xhypo: (1 = yes; 0 = no)

40 oldpeak = ST depression induced by exercise relative to rest

41 slope: the slope of the peak exercise ST segment -- Value 1: upsloping -- Value 2: flat -- Value 3: downsloping

42 rldv5: height at rest

43 rldv5e: height at peak exercise

44 ca: number of major vessels (0-3) colored by flourosopy

45 restckm: irrelevant

46 exerckm: irrelevant

47 restef: rest raidonuclid (sp?) ejection fraction

48 restwm: rest wall (sp?) motion abnormality 0 = none 1 = mild or moderate 2 = moderate or severe 3 = akinesis or dyskmem (sp?)

49 exeref: exercise radinalid (sp?) ejection fraction

50 exerwm: exercise wall (sp?) motion

51 thal: 3 = normal; 6 = fixed defect; 7 = reversable defect

52 thalsev: not used

53 thalpul: not used

54 earlobe: not used

55 cmo: month of cardiac cath (sp?) (perhaps "call")

56 cday: day of cardiac cath (sp?)

57 cyr: year of cardiac cath (sp?)

58 num: diagnosis of heart disease (angiographic disease status) -- Value 0: < 50% diameter narrowing -- Value 1: > 50% diameter narrowing (in any major vessel: attributes

59 through

68 are vessels)

59 lmt

60 ladprox

61 laddist

62 diag

63 cxmain

64 ramus

65 om1

66 om2

67 rcaprox

68 rcadist

69 lvx1: not used

70 lvx2: not used

71 lvx3: not used

72 lvx4: not used

73 lvf: not used

74 cathef: not used

75 junk: not used

76 name: last name of patient (I replaced this with the dummy string "name")

This project is to explore the heart disease data. As we can observe, the values of some features are discrete integer numbers. Here, we can consider the statistic analysis and building a probabilistic graph model. Because, we assume that the random variables are mutually independent. There will be some unexpected results.

There are five .py files.

<main>

```
from HeartDiseaseAnalysis import analysis
from Data_Preprocessing import preprocessing
from Load_Data import load_data


def run(df):
    analysis(df)


def app():

    print('Loading Data ......')
    df = load_data()

    print('Preprocessing data ......')
    data = preprocessing(df)

    print('Analysing ......')
    run(data)


if __name__ == '__main__':
    app()
```

<Libs>

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

```
import seaborn as sns
from pandas import DataFrame
```

<Load_Data>

```
from Libs import *

def load_data():

    filename = 'DataSet/heart.csv'
    df = pd.read_csv(filename)

    return df
```

<Data_Preprocessing>

```
from Libs import *


def preprocessing(df):

    columns = list(df.columns)

    names = []
    for column in columns:

        if len(np.unique(df[column])) > 10:
            continue
        else:
            names.append(column)

    return df[names]
```

<HeartDiseaseAnalysis>

```
from Libs import *


def correlation_plot(df):

    cols = df.columns

    cm = np.corrcoef(df.values.T)
    sns.set(font_scale=0.8)
    _ = sns.heatmap(cm, cbar=True,
                    annot=True,
                    square=True,
```

```
                fmt='.2f',
                annot_kws={'size': 7},
                yticklabels=cols,
                xticklabels=cols)
    plt.show()


def hist_plot(df):

    names = df.columns

    length = len(names)
    cols = 4
    rows = np.ceil(length / cols)
    fig = plt.figure()

    for ith_fig, name in enumerate(names):
        ax = fig.add_subplot(rows, cols, ith_fig + 1)
        num_bins = len(np.unique(df[name]))
        df[name].hist(bins=num_bins, ax=ax)

        plt.ylabel('Amount')
        plt.title('Amount(' + name + ')')

    plt.subplots_adjust(wspace=0.5, hspace=0.5)
    plt.show()


def hist_plot_given_target(df):

    names = list(df.columns)
    names.remove('target')

    length = len(names)
    cols = 4
    rows = np.ceil(length / cols)
    fig = plt.figure()

    for ith_fig, column in enumerate(names):
        counts = pd.crosstab(df['target'], df[column])

        for i in range(len(np.unique(df['target']))):
            counts.ix[i] = counts.ix[i] / counts.ix[i].sum()
```

```
        ax = fig.add_subplot(rows, cols, ith_fig + 1)
        counts.plot(kind='bar', stacked=True, rot=0, ax=ax, sharey=True)
        plt.ylabel('Portion')
        plt.title('p(' + column + '|target)')

    plt.subplots_adjust(wspace=0.5, hspace=0.5)
    plt.show()


def hist_plot_given_sex_n_target(df):

    names = list(df.columns)
    names.remove('sex')
    names.remove('target')

    length = len(names)
    cols = 4
    rows = np.ceil(length / cols)
    fig = plt.figure()

    for i, item in enumerate(names):
        column = ['target', 'sex', item]
        data = df.groupby(column).size().unstack().fillna(0) / 1.0

        for m in range(2):
            for n in range(2):
                data.ix[m].ix[n] = data.ix[m].ix[n] / data.ix[m].ix[n].sum()

        ax = fig.add_subplot(rows, cols, i + 1)
        data.plot(kind='bar', stacked=True, rot=0, ax=ax, sharey=True)
        plt.xlabel(('target', 'sex'))
        plt.ylabel('Portion')
        plt.title('p(' + item + '|target, sex)')

    plt.subplots_adjust(wspace=0.5, hspace=0.5)
    plt.show()


def calculate_prob(prob, names, variable):

    p1 = 1
    p2 = 1

    if variable == [] or len(variable) < 9:
```

```
        sex = np.random.randint(2)
        cp = np.random.randint(4)
        fbs = np.random.randint(2)
        restecg = np.random.randint(3)
        exang = np.random.randint(2)
        slope = np.random.randint(3)
        ca = np.random.randint(5)
        thal = np.random.randint(4)
        target = np.random.randint(2)
        variable = [sex, cp, fbs, restecg, exang, slope, ca, thal, target]
    else:
        [sex, cp, fbs, restecg, exang, slope, ca, thal, target] = variable

    for v, name in zip(variable, names):
        p1 *= prob[name + '|target'][target][v]
        p2 *= prob[name][v]

    p_hd = p1 * prob['target'][target] / p2

    print('p(target=%d|sex=%d, cp=%d, fbs=%d, restecg=%d, exang=%d, slope=%d,
ca=%d, thal=%d) = %f'
          % (target, sex, cp, fbs, restecg, exang, slope, ca, thal, p_hd))


def train(df, names):

    prob = {}
    tot_num = df.shape[0]

    for name in names:
        # calculate p(variable)
        prob[name] = list(df.groupby(name).size() / tot_num)

        if name == 'target':
            continue
        else:
            # calculate p(variable|target)
            p = pd.crosstab(df['target'], df[name])
            for i in range(len(np.unique(df['target']))):
                p.ix[i] = p.ix[i] / p.ix[i].sum()
            prob[name+'|target'] = np.array(p)

    return prob
```

```
def probabilistic_graph_model(df):

    names = list(df.columns)


print('----------------------------------------------------------------------
--------------------')
    print('Graphical Model Node:')
    print(names)
    print('Assuming that each variable is independent. '
          'For each variable in each node, the possible values of each variable
and its probability:')
    for name in names:
        print(name, '=', np.unique(df[name]))

    print('''Naive Bayes:
                                p(sex,...|target)p(target)
          p(target|sex, ...)  =  ----------------------------
                                       p(sex,...)''')

print('----------------------------------------------------------------------
--------------------')

    prob = train(df, names)
    names.remove('target')

    for i in range(10):
        calculate_prob(prob, names, list(df.ix[i]))


def analysis(csv_data):

    df = csv_data

    print('Plotting Correlation Matrix ......')
    correlation_plot(df)

    print('Plotting Histgram ......')
    hist_plot(df)

    print('Plotting Histgram given target ......')
    print('Target = {0, 1}, 0 = have not disease, 1 = have disease')
    hist_plot_given_target(df)
```

```
hist_plot_given_sex_n_target(df)

print('Starting Graphical Model ......')
probabilistic_graph_model(df)
```

The Results

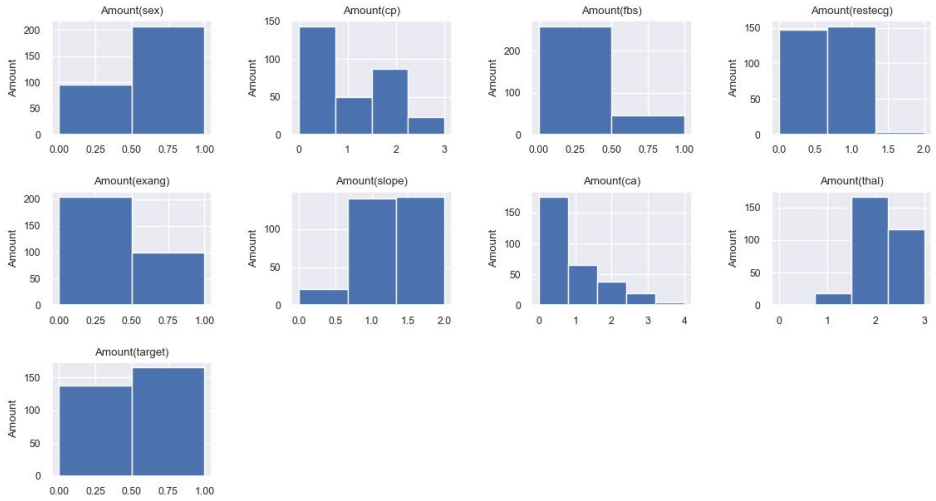
Loading Data .....

Preprocessing data .....

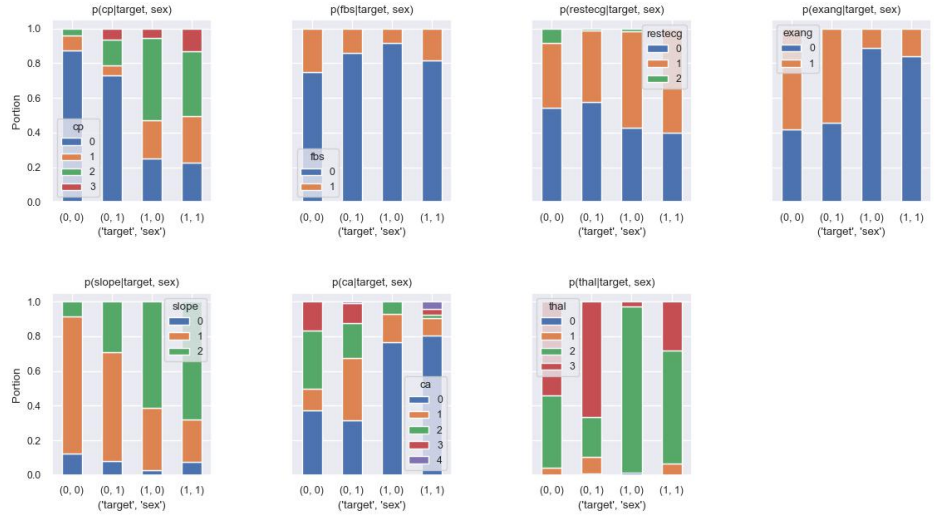
Analysing .....

Plotting Correlation Matrix .....

Plotting Histgram .....

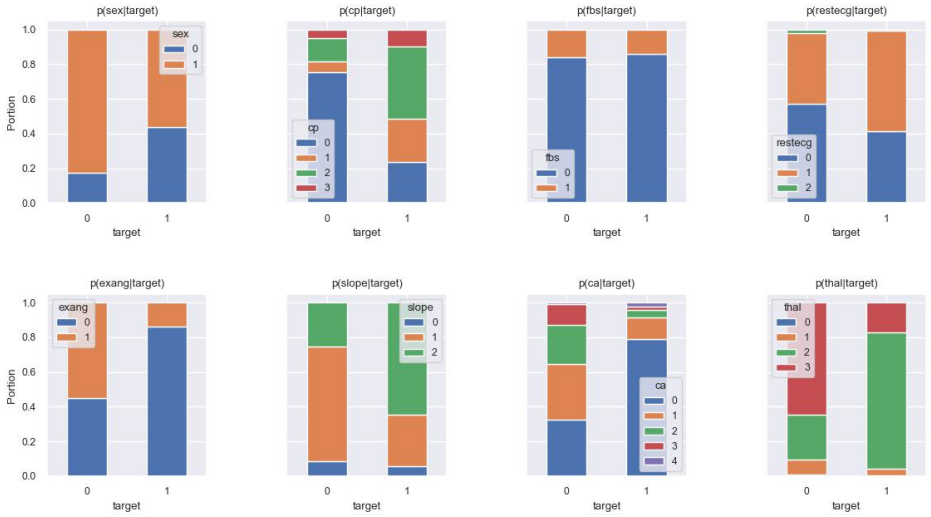


Plotting Probability given target .....



Target = {0, 1}, 0 = have not disease, 1 = have disease

Plotting Probability given target and sex



Starting Graphical Model .....

Graphical Model Node:

['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']

Assuming that each variable is independent. For each variable in each node, the possible values of each variable and its probability:

sex = [0 1]

cp = [0 1 2 3]

fbs = [0 1]

restecg = [0 1 2]

exang = [0 1]

slope = [0 1 2]

ca = [0 1 2 3 4]

thal = [0 1 2 3]

target = [0 1]

Naive Bayes:

$$p(target|sex, ...) = \frac{p(sex,...|target)p(target)}{p(sex,...)}$$

---

p(target=1|sex=1, cp=3, fbs=1, restecg=0, exang=0, slope=0, ca=0, thal=1) = 0.384404

p(target=1|sex=1, cp=2, fbs=0, restecg=1, exang=0, slope=0, ca=0, thal=2) = 1.513810

p(target=1|sex=0, cp=1, fbs=0, restecg=0, exang=0, slope=2, ca=0, thal=2) = 3.364653

p(target=1|sex=1, cp=1, fbs=0, restecg=1, exang=0, slope=2, ca=0, thal=2) = 2.751869

p(target=1|sex=0, cp=0, fbs=0, restecg=1, exang=1, slope=2, ca=0, thal=2) = 0.509948

p(target=1|sex=1, cp=0, fbs=0, restecg=1, exang=0, slope=1, ca=0, thal=1) = 0.180950

p(target=1|sex=0, cp=1, fbs=0, restecg=0, exang=0, slope=1, ca=0, thal=2) = 1.562834

p(target=1|sex=1, cp=1, fbs=0, restecg=1, exang=0, slope=2, ca=0, thal=3) = 0.840939

p(target=1|sex=1, cp=2, fbs=1, restecg=1, exang=0, slope=2, ca=0, thal=3) = 0.755314

p(target=1|sex=1, cp=2, fbs=0, restecg=1, exang=0, slope=2, ca=0, thal=2) = 2.661606

The graphical model is not good. The probability will be over 1 due to the assumption of independence. We also do not normalize the probability. Maybe, we can reduce the dependencies between variables such that we can get good results.