Project 2 _ Report 1

本课程相关内容已划分好文件夹全部上传到Github，地址如下：

https://github.com/foreverfruit/DataMining.git

计划任务

根据本课程项目二计划书，第一阶段任务如下（摘自proposal）：

> - 数据预处理，将文本数据整理成需要的格式。
> - 完成python语言的学习。
> - 学习一种分类算法的理论，尝试采用python实现，或者至少能应用已有开源库做一些该算法的demo。

Task1_数据预处理

1.去除缺失数据。本数据集训练集中存在属性缺失的记录数量为2399，占比0.0737，不做额外处理，直接舍弃。

2.对连续型数据做简单统计：最大值、最小值、均值、方差、与类别的相关度，统计结果如表所示：

属性名	最小值	最大值	均值	标准差	相关性
age	17	90	38.44	13.13	0.242
fnlwgt	13769	1484705	189793.83	105652.97	-0.009
edu_num	1	16	10.12	2.55	0.335
cpl_gain	0	99999	1092.01	7406.35	0.221
cpl_loss	0	4356	88.37	404.30	0.150
hours_pw	1	99	40.93	11.98	0.229

3.数据离散化：

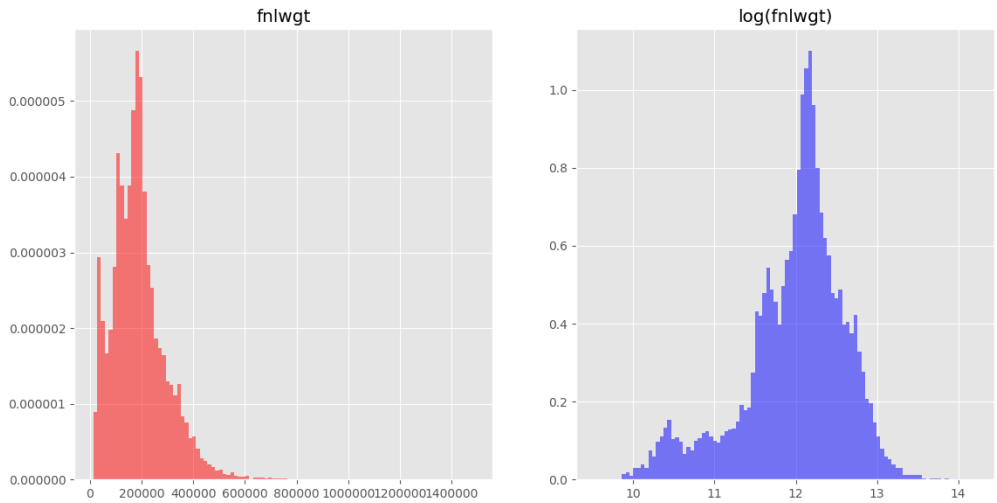
- 对连续数据进行离散化映射，区间到int值的映射，并保证区间的有序性。离散化方法：

- 先看分布情况，再根据分布做映射。
- 像age、edu_num、hours_pw这样的属性，取值空间较小，可以直接简单的固定区间长度做映射。edu_num不做处理。age、hours_pw的处理如下公式：

  $$age' = age/5, hours\_pw = hours\_pw/5$$

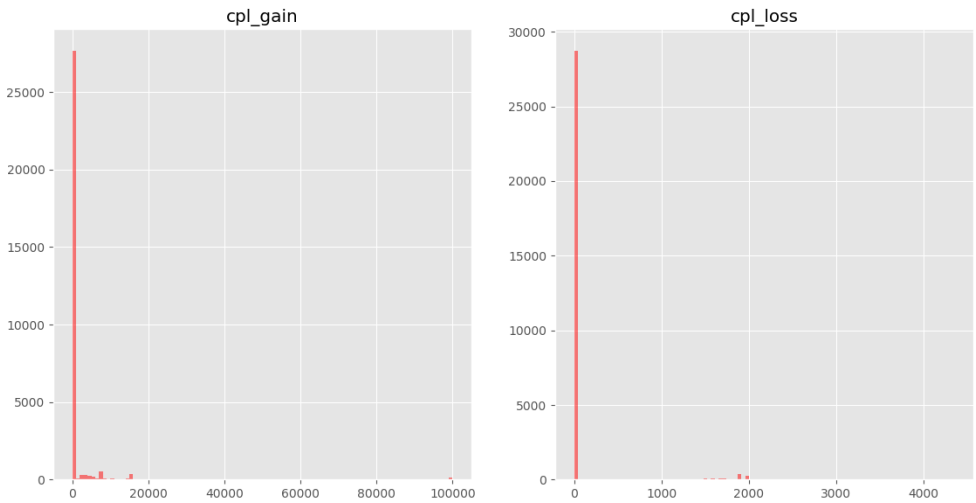
  都采用长度为5的划分区间映射到0到20的取值范围上.
- 对fnlwgt、cpl_gain、cpl_loss这样大区间的量，先做一个离群点的判断，剔除上下界上的离群点，再对剩下的数据做取对数操作之后划分区间映射。

  fnlwgt：先取log运算，再划分到（log后的取值范围[9,15]，按0.5长度做等区间长度划分）等长区间做映射。图如下

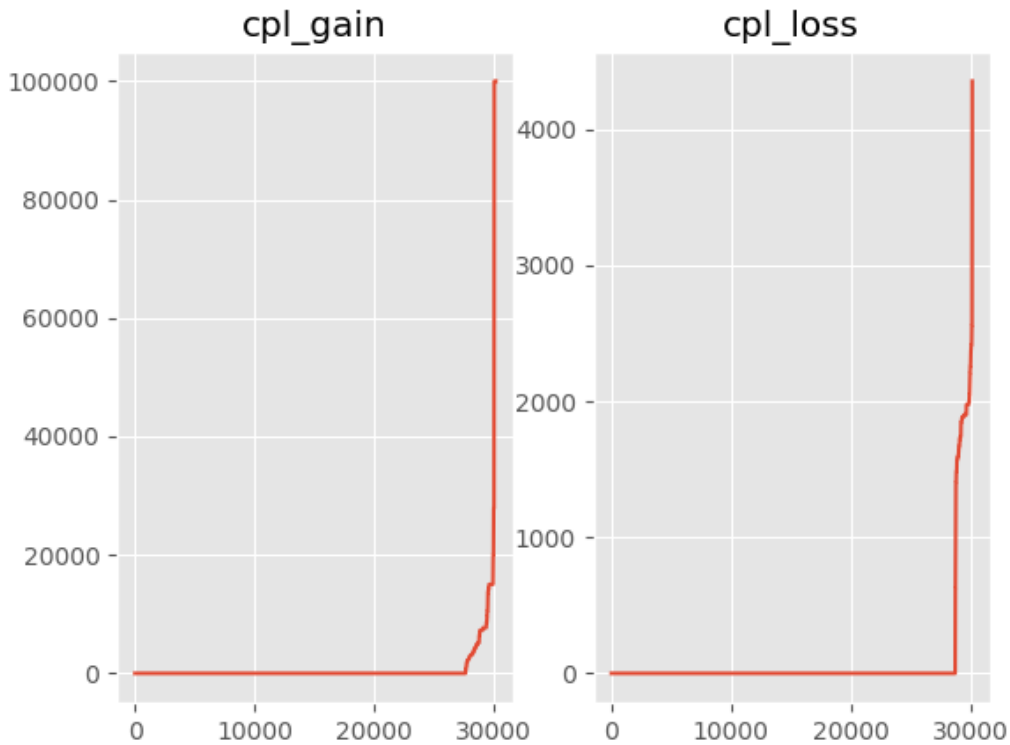



```
fnlwgt = np.loadtxt('preprocess',int,delimiter=',',usecols=2)
a = np.log(fnlwgt)

figure = plt.figure()
ax1 = figure.add_subplot(121)
ax1.hist(fnlwgt,normed=True,bins=100,color='r',alpha=0.5)
ax1.set_title('fnlwgt')

ax2 = figure.add_subplot(122)
ax2.hist(a,normed=True,bins=100,color='b',alpha=0.5)
ax2.set_title('log(fnlwgt)')
plt.show()
```

cpl_gain、cpl_loss：先看hist分布图（如下），发现值分布非常集中，但是值域很广。对这两组属性的处理先后尝试了多种办法，过程如下。



1. 采用百分位数截断。排序后先截断（上下各截断5%），留取中间的90%做映射到[1,k]，最后对下界5%的数据取0，上界5%数据取k+1，但实际效果不好，因为这个阈值不好取，取了也不能简单说明是离群的，而且很受数据分布影响，如两组数据的分布都集中在5%的的区间以内，也就说[0,95%]的数据取值都是0，故弃用这种方法。



2. 采用偏差程度（z-分数）作为偏离程度的衡量，绝对值超过3的认为是异常(对于标准差不为0或不接近0的数据，z-分数是有意义的，且通常以3作为异常值的评判阈值)，剩下的数据映射到[1,k]范围内，最后对上下界异常区间内的数据分别映射为0和k+1.但实际实现中效果不好，因为这个离群的阈值和标准std的大小很有关系，不好取，如cpl_gain取threshold为[-0.2,0]任然能满足该区间内数据占比大于90%，但没法解释，因为这样是认为超过均值的所有元素都是离群的被舍弃，这不科学，且合理数据区域的取值区间仍然很大[0,1151]。
3. 类似百分位数思想，针对数据集中的特点，采用均值的上下p%的值域内的数据做为合理数据，之外的数据认为离群点。发现90%上的数据都是0，剩下的元素取值很大。这种办法还是不能解决问

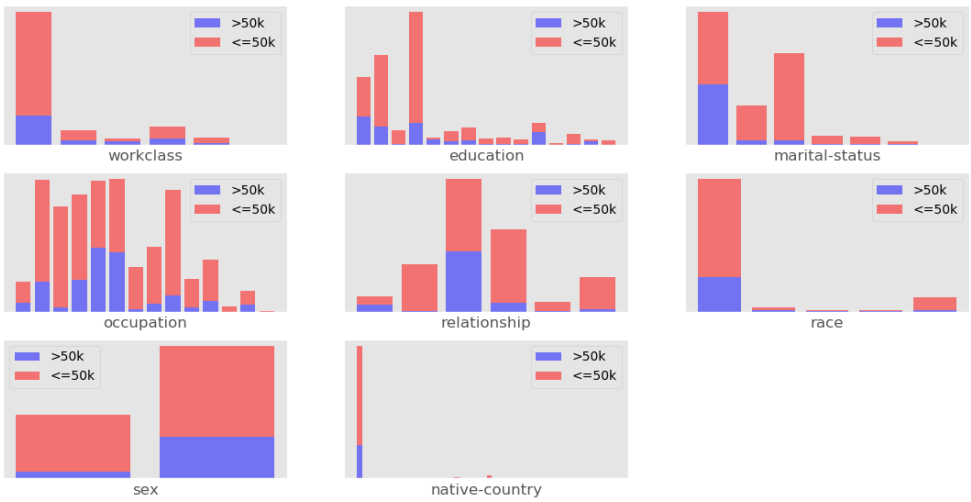
题。

4. // TODO 未解决。目前先简单做标记，对不为0的元素取1做标志，不做量的处理。

连续属性离散之后的统计数据。

属性名	最小值	最大值	均值	标准差	相关性
age	3	18	7.29	2.65	0.240
fnlwgt	1	10	5.49	1.30	-0.000
edu_num	1	16	10.12	2.55	0.335
cpl_gain	0	1	0.08	0.28	0.266
cpl_loss	0	1	0.05	0.21	0.138
hours_pw	0	19	8.11	2.41	0.230

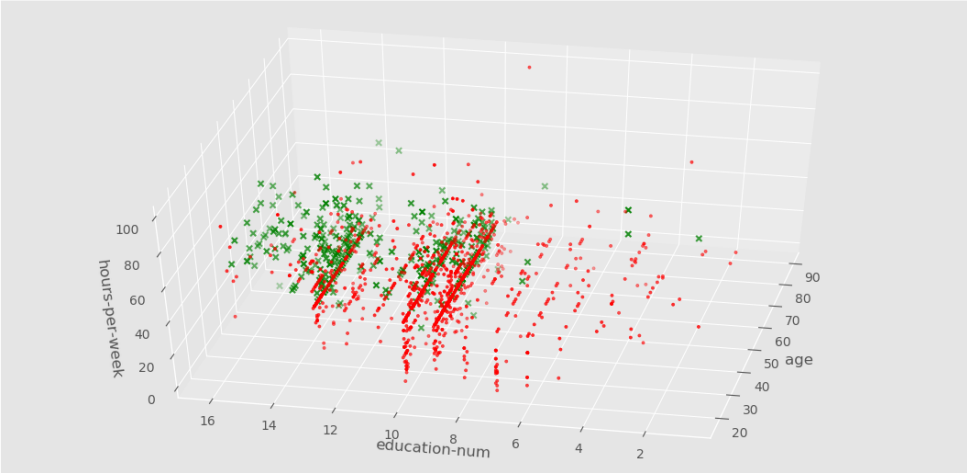
- 对字符串数据进行离散化映射，字符串到int的映射，这里没有有序性。*存在一个问题：如何衡量该属性与类别的相关性？**1**）根据经验，对字符串做排序，这样得到的映射是有序的，可以求得与类别属性的相关性。**2**）不做相关性分析，直接用图的方式表现属性取值与类别的区分关系（这里采用第二种，即可视化的方式，但感觉还是不如数学运算可靠）*



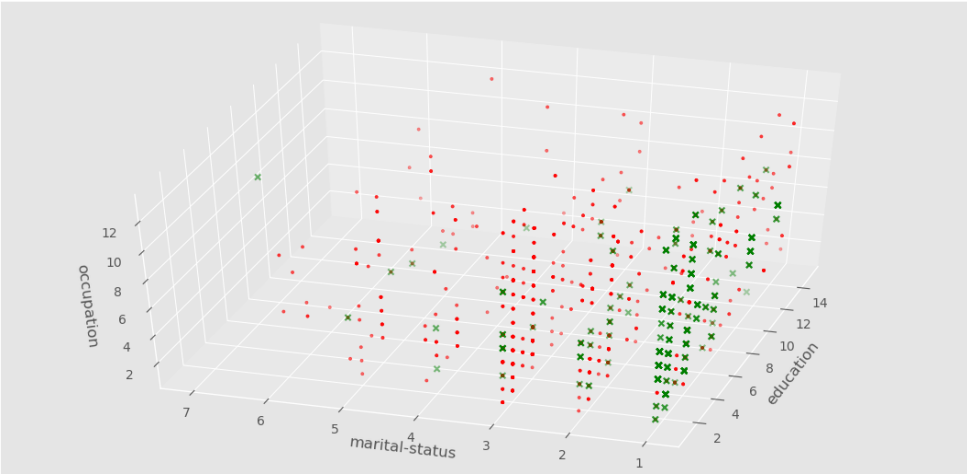
结论：解释性不强，没有明确的衡量这种区分度的方法

4.可视化（分布图、最大相关性的属性建立的3D散点图）

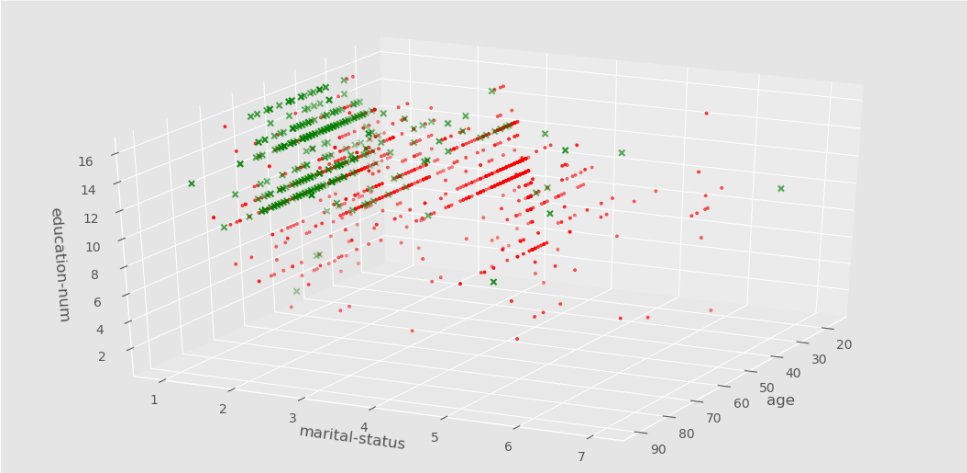
- 与类别属性具有最高相关度的3个连续属性为3个轴，绘制的类型3D散点图



- 与类别属性最相关的3个离散属性为3个轴，绘制的类型3D散点图



- 选取3个相关性较高(观测结论)的属性绘制的类型3D散点图



数据预处理重做

之前的做法存在问题，且没有选定分类方法，所以预处理不当。现选定KNN算法，则数据预处理的时候需要做归一化处理，以平衡所有属性对距离度量的影响。

处理还是分为两个部分：连续属性和标称属性

1.连续属性

```
# 属性名-列标号
age-0,fnlwgt-2,edu_num-4,cpl_gain-10,cpl_loss-11,hours_pw-12
```

以上属性根据统计的取值范围，做以下操作：

- cpl_loss和cpl_gain，采用比率的方法合并为一个属性
- 采用最大值最小值归一方法

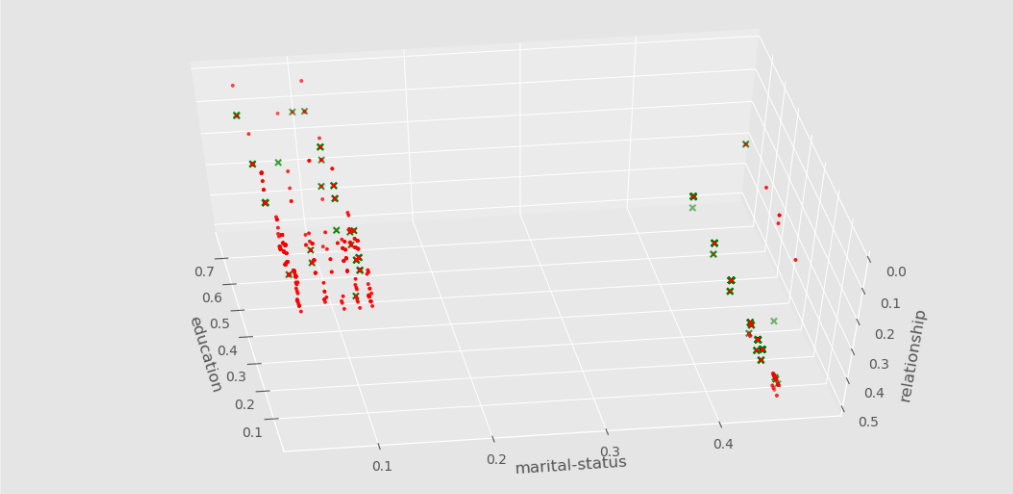
2.标称属性

为了便于统计，保留原来的整数映射，之后，根据每一个属性值的取class 1的概率做映射。这么做的效果就是，假设A属性取值为'a','b','c','d','e',若取'd'的对象0.99概率为类1，取'e'的对象0.98概率为类1，那么用概率作为属性值的映射，这两种取值看做是比较邻近的。也就是说，你和我近不近，看你和我离类1的距离（概率）近不近

3.统计信息及映射关系

```
------------相关度-----------
age: 0.242
workclass: 0.160
fnlwgt: -0.001
education: 0.367
education-num: 0.335
marital-status: 0.448
occupation: 0.350
relationship: 0.455
race: 0.100
sex: 0.217
capital-loss/gain: 0.150
hours-per-week: 0.229
native-country: 0.102
-----------------------
```

相关性最高的三个属性：marital-status、relationship、education，以这三个属性绘制的散点图如下：



映射关系：之后测试的时候，测试对象需要按照该关系做数据转换，才能应用到分类算法中

- 标称属性映射如下：

```
# step 1 数据映射字典（字符串型数据到整形映射），这里没有什么意义，仅仅为了统计计算方便
dic_workclass = {'Private':1, 'Self-emp-not-inc':2, 'Self-emp-inc':3, 'Federal-
gov':4,'Local-gov':4, 'State-gov':5, 'Without-pay':6, 'Never-worked':7}
dic_education = {'Bachelors':1, 'Some-college':2, '11th':3, 'HS-grad':4, 'Prof-
school':5,'Assoc-acdm':6, 'Assoc-voc':7, '9th':8, '7th-8th':9, '12th':10,
'Masters':11,'1st-4th':12, '10th':13, 'Doctorate':14, '5th-6th':15, 'Preschool':15}
dic_marital_status = {'Married-civ-spouse':1, 'Divorced':2, 'Never-married':3,
'Separated':4,'Widowed':5, 'Married-spouse-absent':6, 'Married-AF-spouse':7}
dic_occupation = {'Tech-support':1, 'Craft-repair':2, 'Other-service':3, 'Sales':4,
'Exec-managerial':5, 'Prof-specialty':6, 'Handlers-cleaners':7, 'Machine-op-inspct':8,
'Adm-clerical':9, 'Farming-fishing':10, 'Transport-moving':11, 'Priv-house-serv':12,
 'Protective-serv':13, 'Armed-Forces':14}
dic_relationship = {'Wife':1, 'Own-child':2, 'Husband':3, 'Not-in-family':4,'Other-
relative':5, 'Unmarried':6}
dic_race = {'White':1, 'Asian-Pac-Islander':2, 'Amer-Indian-Eskimo':3, 'Other':4,
'Black':5}
dic_sex = {'Female':1, 'Male':2}
dic_native_country = {'United-States':1, 'Cambodia':2, 'England':3, 'Puerto-
Rico':4,'Canada':5, 'Germany':6, 'Outlying-US(Guam-USVI-etc)':7, 'India':8,'Japan':9,
'Greece':10, 'South':11, 'China':12, 'Cuba':13, 'Iran':14, 'Honduras':15,
'Philippines':16, 'Italy':17, 'Poland':18, 'Jamaica':19,'Vietnam':20, 'Mexico':21,
'Portugal':22, 'Ireland':23, 'France':24,'Dominican-Republic':25, 'Laos':26,
'Ecuador':27, 'Taiwan':12, 'Haiti':29,'Columbia':30, 'Hungary':31, 'Guatemala':32,
'Nicaragua':33, 'Scotland':28,'Thailand':34, 'Yugoslavia':35, 'El-Salvador':36,
'Trinadad&Tobago':37,'Peru':38, 'Hong':39, 'Holand-Netherlands':40}
dic_class = {'<=50K':0,'>50K':1}

# 这里才是最后的映射，根据的是每个取值中class1的概率
workclass ------------------
{1.0: 0.21879206676837476, 2.0: 0.2857142857142857, 3.0: 0.55865921787709494, 4.0:
0.32358803986710966, 5.0: 0.26896012509773259, 6.0: 0.0}
education ------------------
{1.0: 0.42149088025376685, 2.0: 0.20005989817310571, 3.0: 0.056297709923664119, 4.0:
0.16432926829268293, 5.0: 0.74907749077490771, 6.0: 0.25396825396825395, 7.0:
0.26319816373374139, 8.0: 0.054945054945054944, 9.0: 0.06283662477558348, 10.0:
0.076923076923076927, 11.0: 0.56422864167178854, 12.0: 0.039735099337748346, 13.0:
0.071951219512195116, 14.0: 0.7466666666666667, 15.0: 0.036036036036036036}
marital-status ------------------
{1.0: 0.45495911837895486, 2.0: 0.10726150925486473, 3.0: 0.048324079786140242, 4.0:
0.070287539936102233, 5.0: 0.096735187424425634, 6.0: 0.083783783783783788, 7.0:
0.47619047619047616}
occupation ------------------
{1.0: 0.30482456140350878, 2.0: 0.22531017369727047, 3.0: 0.041095890410958902, 4.0:
0.27064732142857145, 5.0: 0.48522044088176353, 6.0: 0.44848935116394256, 7.0:
0.061481481481481484, 8.0: 0.12461851475076297, 9.0: 0.13383499059392637, 10.0:
0.11627906976744186, 11.0: 0.20292620865139949, 12.0: 0.006993006993006993, 13.0:
0.32608695652173914, 14.0: 0.1111111111111111}
relationship ------------------
{1.0: 0.49359886201991465, 2.0: 0.014330497089117778, 3.0: 0.45566877958757923, 4.0:
0.10652342738804038, 5.0: 0.03937007874015748, 6.0: 0.066313823163138233}
race ------------------


{1.0: 0.26371804264836307, 2.0: 0.27709497206703909, 3.0: 0.11888111888111888, 4.0:
```

```
0.090090090909090912 , 5.0: 0.12992545260915869 }
sex ------------------
{1.0: 0.11367818442036394 , 2.0: 0.3138370951913641 }
native-country ------------------
{1.0: 0.25432664339732403 , 2.0: 0.3888888888888889 , 3.0: 0.34883720930232559 , 4.0:
0.11009174311926606 , 5.0: 0.3364485981308411 , 6.0: 0.34375, 7.0: 0.0, 8.0:
0.40000000000000002 , 9.0: 0.38983050847457629 , 10.0: 0.27586206896551724 , 11.0:
0.19718309859154928 , 12.0: 0.35454545454545455 , 13.0: 0.27173913043478259 , 14.0:
0.42857142857142855 , 15.0: 0.083333333333333329 , 16.0: 0.31914893617021278 , 17.0:
0.35294117647058826 , 18.0: 0.19642857142857142 , 19.0: 0.125, 20.0: 0.078125, 21.0:
0.054098360655737705 , 22.0: 0.11764705882352941 , 23.0: 0.20833333333333334 , 24.0:
0.44444444444444442 , 25.0: 0.029850746268656716 , 26.0: 0.11764705882352941 , 27.0:
0.14814814814814814 , 28.0: 0.18181818181818182 , 29.0: 0.095238095238095233 , 30.0:
0.035714285714285712 , 31.0: 0.23076923076923078 , 32.0: 0.047619047619047616 , 33.0:
0.060606060606060608 , 34.0: 0.17647058823529413 , 35.0: 0.375, 36.0:
0.089999999999999997 , 37.0: 0.1111111111111111 , 38.0: 0.066666666666666666 , 39.0:
0.31578947368421051 , 40.0: 0.0}
```

- 连续属性映射如下：
    - age-0，hours_pw-12，取值在(0,100)以内，直接除以100,x'=x/100
    - edu_num-4，（1,16），x'=(x-1)/15
    - fnlwgt-2，取值范围很大，先取log，log后的取值为（9,15），x' = (x-9)/6
    - cpl_loss/gain,(0,4356),x'=x/4356，这里取loss/gain的比值存于原gain-10列，loss-11列舍弃
    - 由于这里采用训练集的最大值最小值，实际测试中，若越界，不做额外处理，直接取边界

Task2_Python学习

本阶段学习内容为python语言的语法学习，以及常用的科学计算库matplotlib和numpy的简单使用，能使用python语言实现数据处理、计算和可视化任务。

具体的学习笔记见本项目下文件：

- /项目二/[Task1]学习python/python.md
- /项目二/[Task1]学习python/matplotlib & numpy.md
- matplotlib练习代码：/项目二/[Task1]学习python/project_matplotlib

github地址：Task 2：python学习

Task3_决策树实现

这部分的学习内容主要是完成一个小的练习项目，对鸢尾花数据集进行挖掘，采用决策树的算法实现预测分类功能。

数据集：http://archive.ics.uci.edu/ml/datasets/Iris

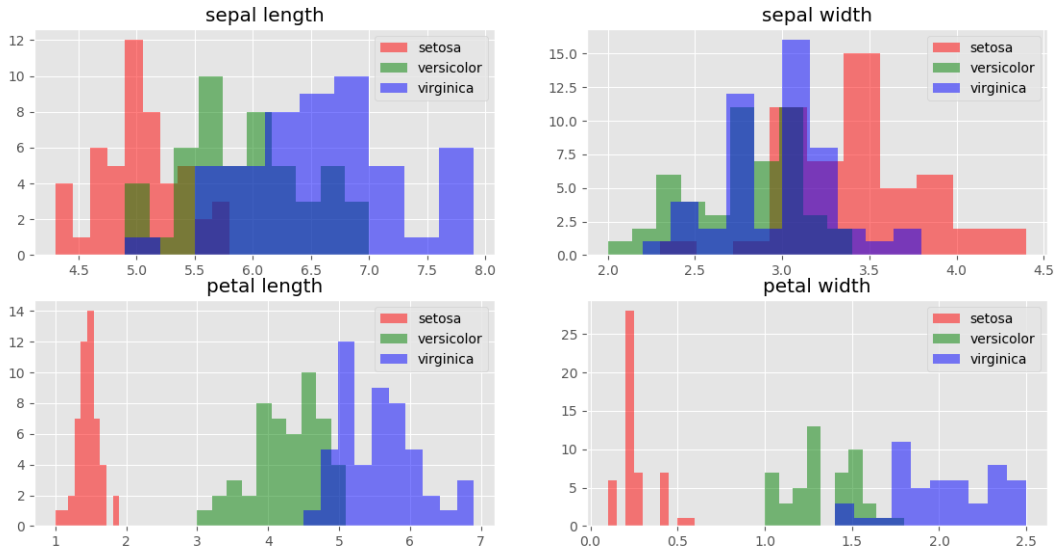
数据集信息

- 数据量小：150个记录，每个类50，共3个类标号
- 属性：4维，连续型
- 数据完整，不存在缺失项

数据可视化

这里可视化采用的是原始数据，为了大致观测到数据的一些特性

1.各属性直方图

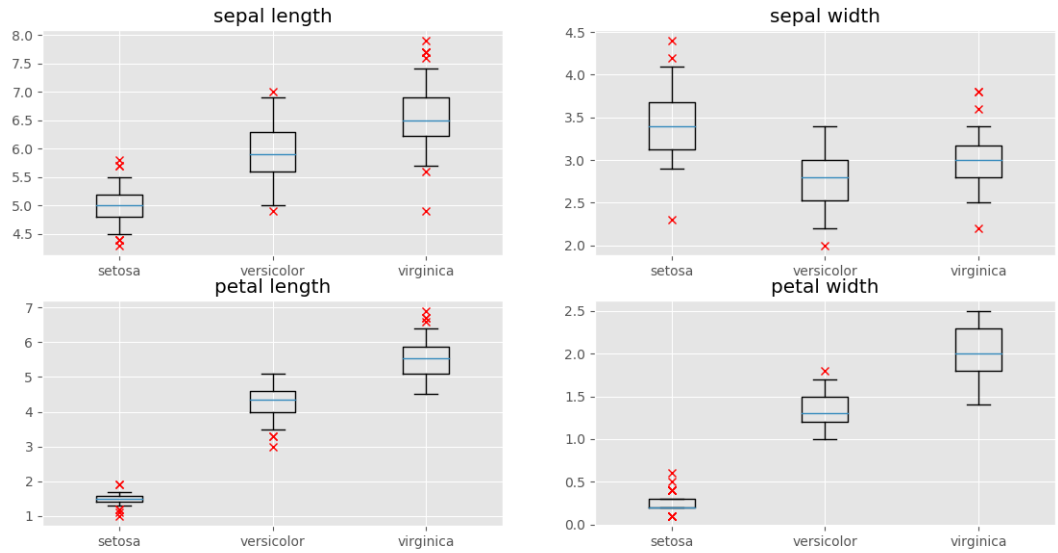


可以看到，属性petal_length和属性petal_width对类的区分度非常高，也和统计数据中的相关度相吻合。（事实上，下文的决策树在结点分裂的时候，会选用划分后Gini指数最低的属性进行划分，即子集的纯度最高，显然第一次的时候用属性petal_width会得到效果最好的划分。）

```
import  numpy as np
import  matplotlib.pyplot as plt
import  os

# load data
pro_root = os.path.abspath('..')
sl,sw,pl,pw = np.loadtxt(pro_root+'/dataset/iris.data' ,
                  delimiter=',',
                  unpack=True,dtype=float,usecols=(0,1,2,3))
datarray = np.array([sl,sw,pl,pw])
attr_names = ['sepal length','sepal width','petal length','petal width']
class_names = ['setosa','versicolor','virginica']

# plot
plt.style.use('ggplot')
figure = plt.figure()
for i,data in enumerate(datarray):
    d1,d2,d3 = data[:50],data[50:100],data[100:]
    axe = figure.add_subplot(221+i)
    axe.hist(d1,color='r',bins=10,alpha=0.5)
    axe.hist(d2,color='g',bins=10,alpha=0.5)
    axe.hist(d3,color='b',bins=10,alpha=0.5)
    axe.set_title(attr_names[i])
    axe.legend(labels = class_names)

plt.show()
```

2.各属性盒装图

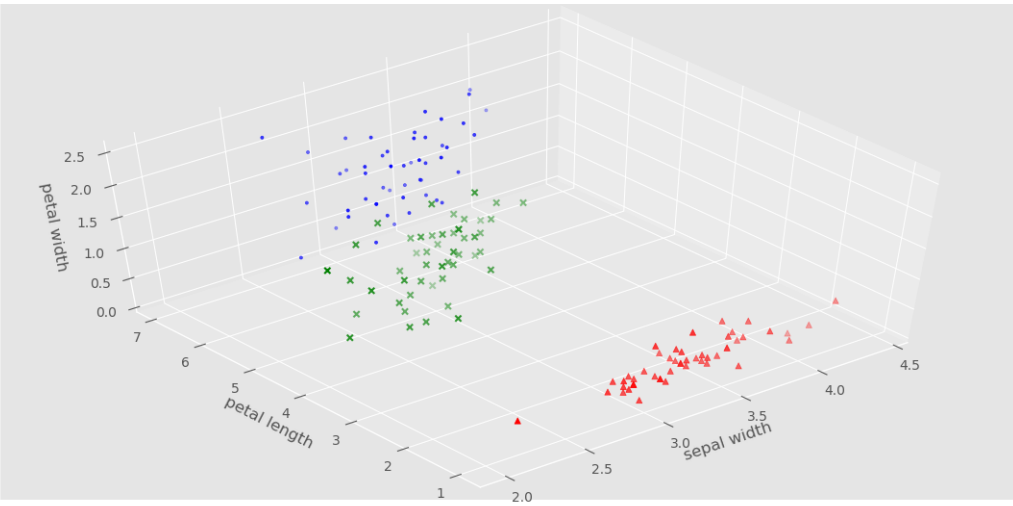
这里取whis参数为0.9，用以控制离群点的“离群”程度定义，图中可以看出类别1在4种属性上的分布都比较分散。



```
import  numpy as np
import  matplotlib.pyplot as plt
import  os

# load data
pro_root = os.path.abspath('..')
sl,sw,pl,pw = np.loadtxt(pro_root+'/dataset/iris.data' ,
                  delimiter=',',
                  unpack=True,dtype=float,usecols=(0,1,2,3))
datarray = np.array([sl,sw,pl,pw])
attr_names = ['sepal length','sepal width','petal length','petal width']
class_names = ['setosa','versicolor','virginica']

# plot
plt.style.use('ggplot')
figure = plt.figure()
for i,data in enumerate(datarray):
    d1,d2,d3 = data[:50],data[50:100],data[100:]
    axe = figure.add_subplot(221+i)
    axe.boxplot((d1,d2,d3), whis=0.9, sym='rx')
    axe.set_title(attr_names[i])
    axe.set_xticklabels(class_names)

plt.show()
```

3、三维散布图

这里选取相关度最高的三个属性做三维图，可以看到红色的类型的区别度较高，也印证直方图的属性分布图中，红色的类型setosa的4个属性与其他类型相应的属性取值范围间隔较大，几乎没有相交的，反映到三维途中会有直观的“距离”感



```
import  numpy as np
import  matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import  os

# load data
pro_root = os.path.abspath('..')
sl,sw,pl,pw = np.loadtxt(pro_root+'/dataset/iris.data',
                 delimiter=',',
                 unpack=True,dtype=float,usecols=(0,1,2,3))
datarray = np.array([sl,sw,pl,pw])
attr_names = ['sepal length','sepal width','petal length','petal width']
class_names = ['setosa','versicolor','virginica']

# plot
plt.style.use('ggplot')
figure = plt.figure()
axe = figure.add_subplot(111, projection = '3d')

axe.scatter(sw[:50], pl[:50], pw[:50], c='r', marker='^')
axe.scatter(sw[50:100], pl[50:100], pw[50:100], c='g', marker='x')
axe.scatter(sw[100:150], pl[100:150], pw[100:150], c='b', marker='.')

axe.set_xlabel(attr_names[1])
axe.set_ylabel(attr_names[2])
axe.set_zlabel(attr_names[3])

plt.show()
```

数据预处理

这里数据集较为简单，没有做过多数据预处理，仅在加载数据的同时，就地做了离散化工作，根据统计信息（如下表），做出相应的离散映射

Class	Min	Max	Mean	SD	Class Correlation
sepal length	4.3	7.9	5.84	0.83	0.78
sepal width	2.0	4.4	3.05	0.43	-0.4194
petal length	1.0	6.9	3.76	1.76	0.9490
petal width	0.1	2.5	1.2	0.76	0.9565

映射步骤：根据每个属性的最大值最小值，取得一个取值空间，然后以0.5为一个步长，将这个取值空间映射到从1开始的整数序列上，如petal width取值范围(0.1,2.5)，则取范围[0,3]根据0.5步长映射为，1 = [0,0.5),2 = [0.5,1),3 = [1,1.5), 4 = [1.5,2), 5 = [2,2.5]，实现代码如下：

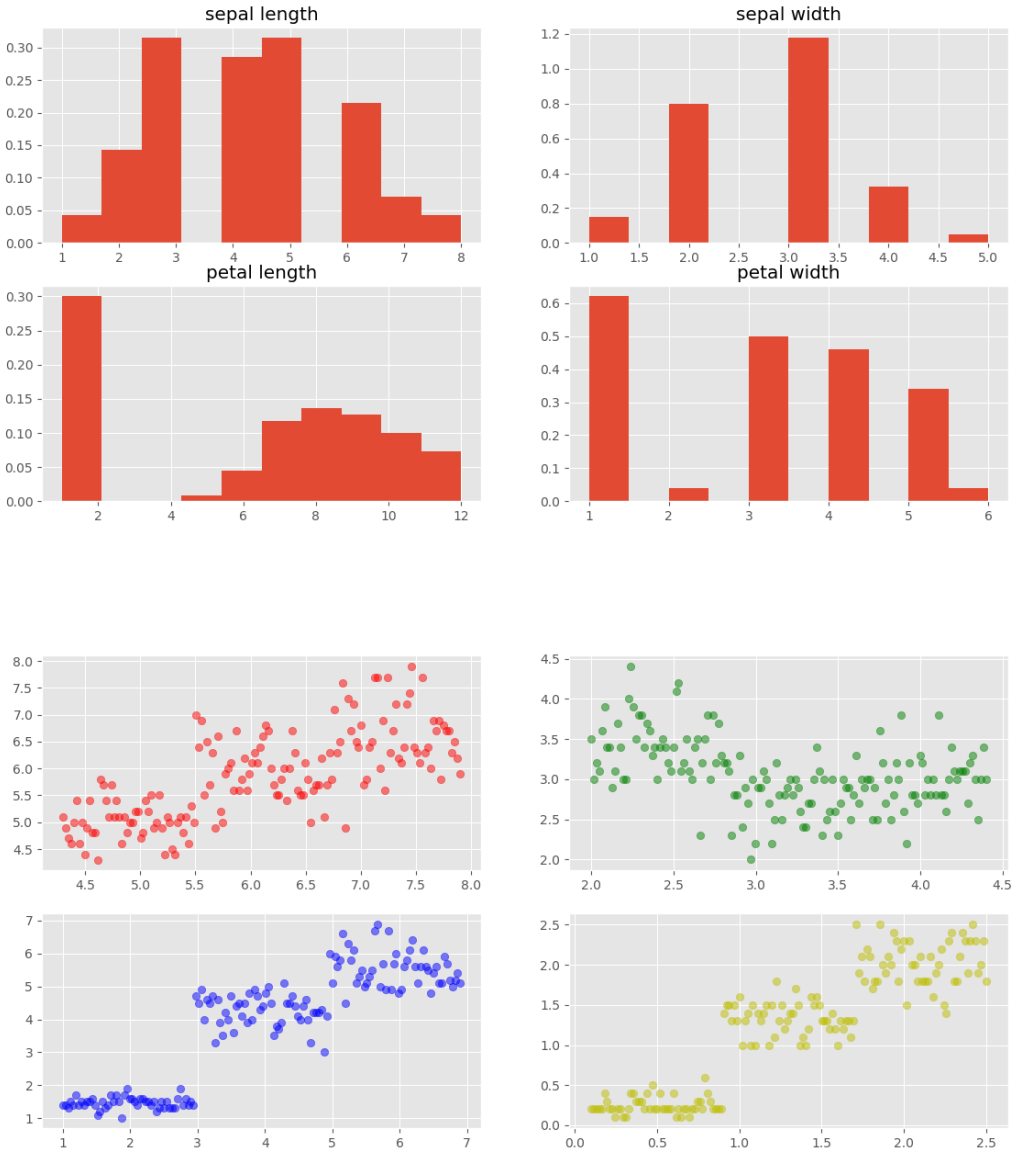
```
newset = []
# 离散值集合
valueset=range(1,13)
# 遍历处理每一条记录
for record in dataset:
    newrecord = []
    for index,value in enumerate(record):
        i = 0
        if index==0:
            # sepal length[4.3,7.9] 取[4,8]内按0.5步长做区间划分
            i = int((float(value)*10-40)/5)
        if index==1:
            i = int((float(value) * 10 - 20) / 5)
        if index==2:
            i = int((float(value) * 10 - 10) / 5)
        if index==3:
            i = int(float(value)*10/5)
        # class label 属性
        if index==4:
            newrecord.append(classlabel.get(value))
        else:
            newrecord.append(valueset[i])
    newset.append(newrecord)
```

决策树实现

这里直接采用书中决策树的描述，根据iris数据集的特点实现了决策树，实现过程中主要有一下重点问题。

- 结点分裂算法：Hunt算法，将数据当做一个数据流，边构造决策树边流动，直到所有的数据都流到了叶结点上。递归执行划分方法。
- 划分问题1：属性划分点的确定，书中着重介绍了二元划分的方案，本数据集是三元划分。三元划分可以转换为二元划分，对于某一特定的属性A，对它的取值空间做二元划分，用划分后的子集的纯度（这里我采用Gini指数度量）来衡量选取属性的效果。这种划分方法优点是实现简单，但会造成决策树深度过大。

  我采用的方法：多路划分，根据划分评价的原理：寻找子类纯度最高的划分。那么可以认为最优的划分就是经过这次划分后，子类中类别区分最明显，这里根据数据集特点做出假设：每种类别的记录在某一个属性上的取值明显呈现“聚堆”现象。那么最优的划分点就是这些“堆”的边界点。本例中划分点没有通过代码计算，采用图像观测。图像如下：



上面的离散处理之后的图像，下面的是未处理的原始数据图像。根据图像可以找到如下的划分点：（原始图上值和划分点值有差距，是因为离散化映射的问题）

```
# 因为值是int的，这里划分点取float，避免值等于划分点时的区间开闭情况
dp = {'sl':(3.5,5.5),'sw':(1.5,2.5,3.5,4.5),'pl':(3,),'pw':(2.1,3.8,4.8)}
```

这里取得上面的各属性的划分点就可以近似认为是最优的划分点，Gini最小，子类的纯度最高。

- 划分问题2：中止划分条件。1）当前结点的所有数据元素的属性值都相同，此时若类标号不同，取多数的类标号，少数的认为是异常值。2）当前结点95%的元素都是同一个类标号。
- 划分问题3：属性选取策略，一个结点具有n个可划分的属性（已经划分过的属性在该结点中元素取值肯定是相同的），问题是先选用n个中的哪一个属性作为当前的划分属性，此时已知各属性的最优划分点，所以可以用已知的划分点，对该结点的数据模拟各属性上的划分（模拟n次），计算这n次划分后的子集合的加权**Gini**系数，然后选取Gini最小的那个属性作为该结点划分属性。
- 划分问题4：可能存在一些结点，根据属性的划分点做划分后，并没有一条数据流入该子结点，此时，同样要为决策树创建这样的空叶结点，并做好标记，因为训练集没有数据流入该结点，测试集可能存在，要避免测试集数据找不到叶结点的情况，它的类别取其父节点中最多的类别。

- 算法流程：创建RootNode(这个TrainingSet流入该结点)，判断当前结点能否分裂（两个中止条件都要判定），选取最优的属性做划分（属性选取算法加权Gini），为每一个划分后的子数据集创建结点（数据分散流入子节点），递归调用子节点的划分函数divide()

算法评估

时间原因，没有做深入评估，仅做了正确率的验证：90%左右（交换不同训练集和测试集得到的近似结果，没做具体的统计）

完整项目代码：Iris_DecisionTree_Github