In [0]:

```
###############Author :Tan Haochen###############
###############Program:EIE#######################
###############Student ID: 55349692##############
```

First read the bank.csv which is download from the uci website, The download link is Bank Marketing Data Set , After successfully download the dataset, we can drop the data with the NaN values out and reset the index. The code shown below, the result is the first 5 objects' data.

In [3]:

```
import pandas as pd
import seaborn as sns
df = pd.read_csv('./bank.csv', delimiter = ';', header = 0)
df = df.dropna()
df = df.reset_index()
df.head()
```

Out[3]:

	index	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	p
0	0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1	
1	1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339	
2	2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330	
3	3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1	
4	4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1	-1	

We can see some detail infomation about all the data, like mean, min, max ,25th, 50th, 70th percentile

In [0]:

```
df.describe()
```

Out[0]:

	index	age	balance	day	duration	campaign	pdays	previous
count	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000
mean	2260.000000	41.170095	1422.657819	15.915284	263.961292	2.793630	39.766645	0.542579
std	1305.244613	10.576211	3009.638142	8.247667	259.856633	3.109807	100.121124	1.693562
min	0.000000	19.000000	-3313.000000	1.000000	4.000000	1.000000	-1.000000	0.000000
25%	1130.000000	33.000000	69.000000	9.000000	104.000000	1.000000	-1.000000	0.000000
50%	2260.000000	39.000000	444.000000	16.000000	185.000000	2.000000	-1.000000	0.000000
75%	3390.000000	49.000000	1480.000000	21.000000	329.000000	3.000000	-1.000000	0.000000
max	4520.000000	87.000000	71188.000000	31.000000	3025.000000	50.000000	871.000000	25.000000

We can also sort the data by using sort_values to sort the data by any attribute

In [0]:

```
df.sort_values('age',ascending=False).head()
```

Out[0]:

	index	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	pr
3311	3311	87	retired	married	primary	no	230	no	no	cellular	30	oct	144	1	-1	
1866	1866	86	retired	married	secondary	no	1503	no	no	telephone	18	mar	165	3	101	

	index	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	pr
4108	4108	84	retired	divorced	primary	no	639	no	no	telephone	18	may	353	3	-1	
4388	4388	83	retired	divorced	primary	no	1097	no	no	telephone	5	mar	181	1	-1	
633	633	83	retired	married	secondary	no	0	no	no	cellular	18	mar	140	10	-1	

We can collect the data which get a Y atrribute and calculate its datas' mean and median

In [0]:

```
yGroup = df.groupby('y')
yGroup.mean()
```

Out[0]:

y	index	age	balance	day	duration	campaign	pdays	previous
no	2261.324500	40.998000	1403.211750	15.948750	226.347500	2.862250	36.006000	0.471250
yes	2249.831094	42.491363	1571.955854	15.658349	552.742802	2.266795	68.639155	1.090211

In [0]:

```
yGroup.median()
```

Out[0]:

y	index	age	balance	day	duration	campaign	pdays	previous
no	2270.5	39.0	419.5	16.0	167.0	2.0	-1.0	0.0
yes	2176.0	40.0	710.0	15.0	442.0	2.0	-1.0	0.0

We can also get one person's information from the whole dataset

In [0]:

```
df.iloc[0, :]
```

Out[0]:

```
index                0
age                 30
job         unemployed
marital        married
education       primary
default              no
balance            1787
housing              no
loan                 no
contact        cellular
day                  19
month               oct
duration             79
campaign              1
pdays                -1
previous              0
poutcome        unknown
y                    no
Name: 0, dtype: object
```

Now we apply four visualization techniques from Lecture 2 (histogram, scatter plot, box plot, parallel co-ordinates)

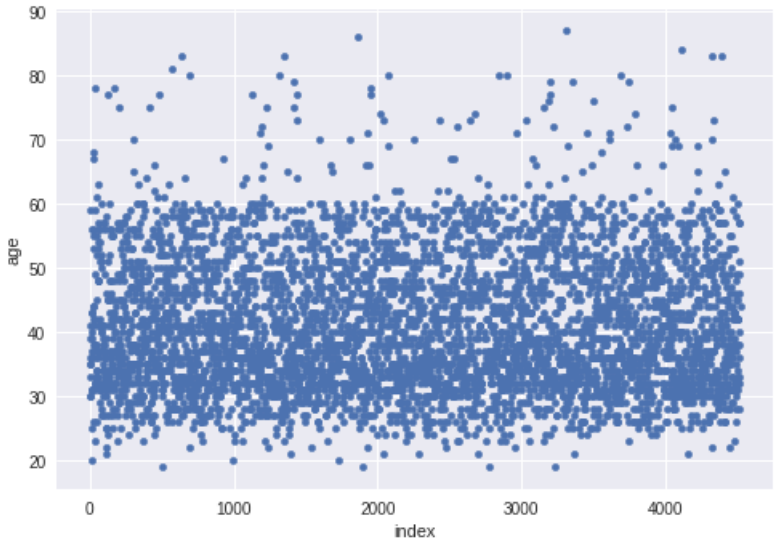
In [0]:

```
df.plot(kind = 'scatter',x ='index', y = 'age')
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-ma
pping will have precedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with
a single row if you really want to specify the same RGB or RGBA value for all points.
```

Out[0]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f865bc32588>
```

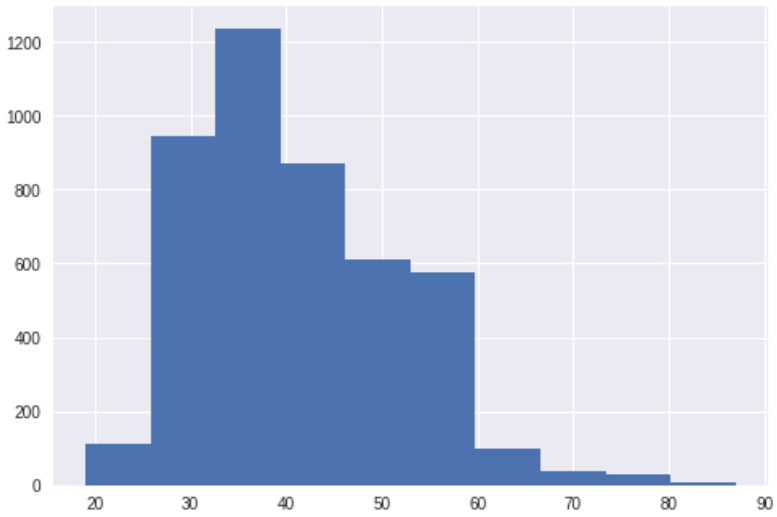


In [0]:

```
df['age'].hist()
```

Out[0]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f865dc899e8>
```

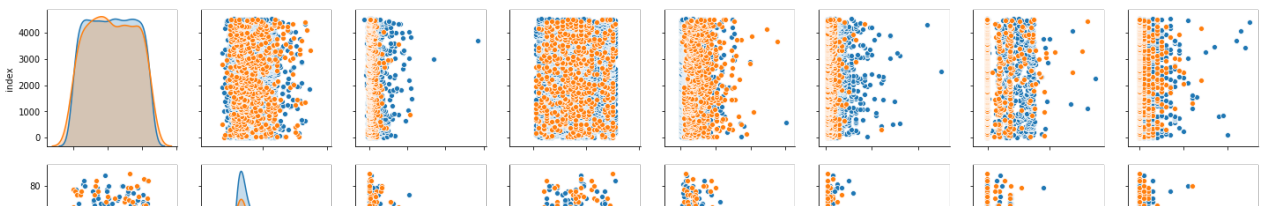


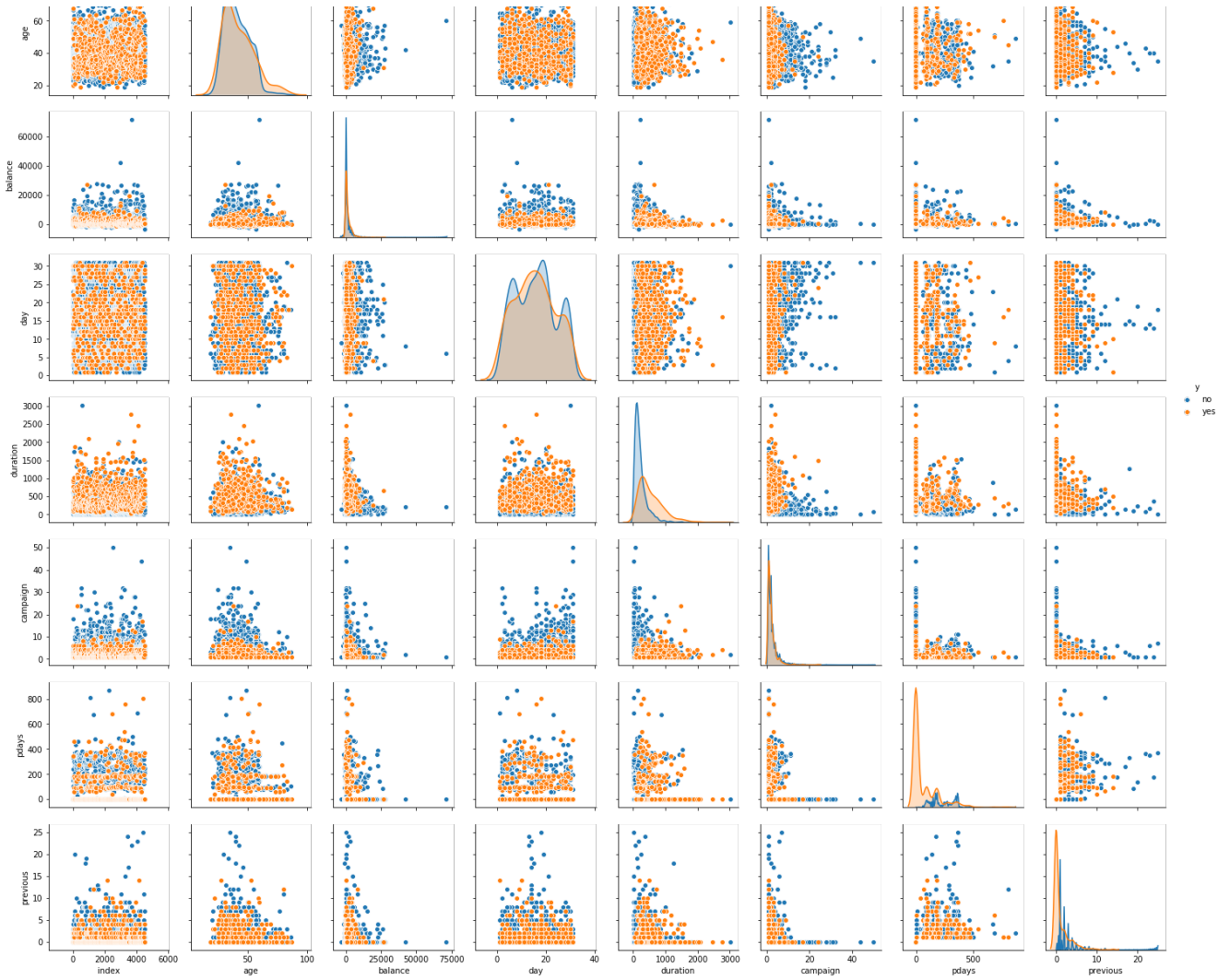
In [5]:

```
sns.pairplot(df, hue = "y")
```

Out[5]:

```
<seaborn.axisgrid.PairGrid at 0x7fc45002b7b8>
```





We can seperate the object with 50 years old age out and plot other forms of table and image

In [0]:

```
df_age50 = df[df['age']==50]
df_age50.shape
```

Out[0]:

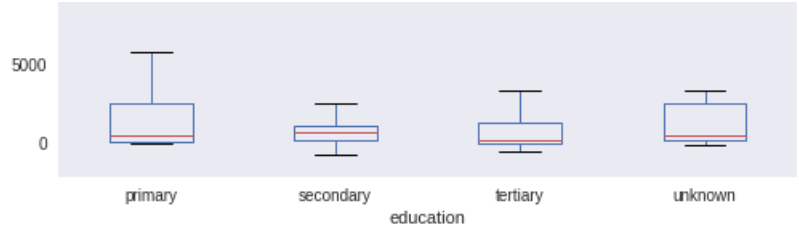
```
(101, 18)
```

In [0]:

```
df_age50.boxplot(by = 'education',
                 column = ['balance'],
                 grid = False)
```

Out[0]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f865b9bf240>
```



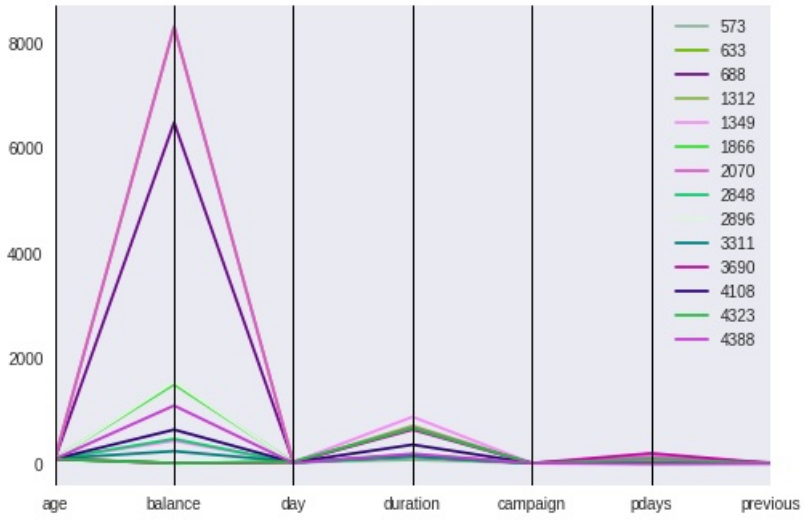
In [0]:

```
df_older_than_80 = df[df['age']>=80]
pd.tools.plotting.parallel_coordinates(df_older_than_80[['index','age', 'balance','day','duration'
,'campaign','pdays','previous']],'index')
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: FutureWarning:
'pandas.tools.plotting.parallel_coordinates' is deprecated, import
'pandas.plotting.parallel_coordinates' instead.
```

Out[0]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f865baf6b70>
```



Exercise1_6435

2019年2月7日 0:02

Problem 2:

a) $P(+) = 5/9$ $P(-) = 4/9$ Entropy is: $\frac{5}{9}\log_2\frac{9}{5} + \frac{4}{9}\log_2\frac{9}{4} = 0.9911$

b) a_1:

		+	−
4	T	1	3
5	F	4	1

$\frac{4}{9}(\frac{1}{4}\log_2 4 + \frac{3}{4}\log_2\frac{4}{3}) + \frac{5}{9}(\frac{1}{5}\log_2 5 + \frac{4}{5}\log_2\frac{5}{4})$

$= 0.7616$ $0.9911 - 0.7616 = 0.2294$

a_2:

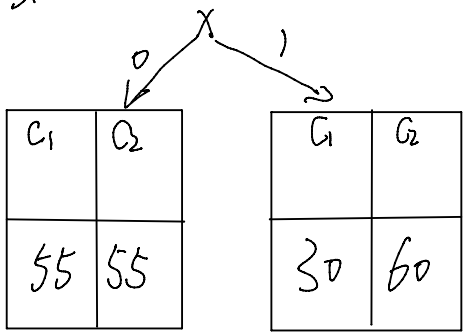
		+	−
5	T	3	2
4	F	2	2

$\frac{5}{9}(\frac{3}{5}\log_2\frac{5}{3} + \frac{2}{5}\log_2\frac{5}{2}) + \frac{4}{9}(\frac{1}{2}\log_2 2 + \frac{1}{2}\log_2 2)$

$= 0.9839$ $0.9911 - 0.9839 = 0.0072$.

other answer see the code block below

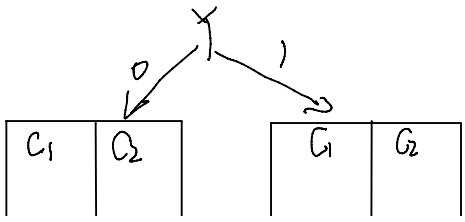
Problem 3:



Entropy: $\frac{110}{200}(\frac{1}{2}\log_2 2 + \frac{1}{2}\log_2 2) + \frac{90}{200}(\frac{3}{9}\log_2\frac{9}{3} + \frac{6}{9}\log_2\frac{9}{6})$

$= 0.55 + 0.45(0.918)$

$= 0.963$.



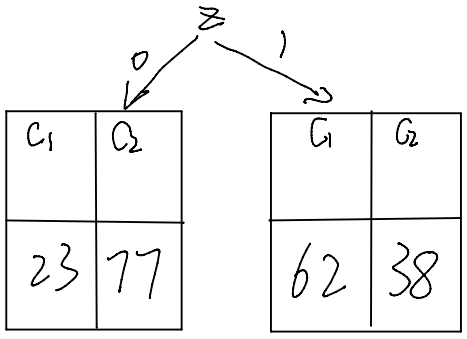
C_1	C_2
30	75

C_1	C_2
55	40

$$\text{Entropy} = \frac{105}{200}\left(\frac{30}{105}\log_2\frac{105}{30} + \frac{75}{105}\log_2\frac{105}{75}\right) + \frac{95}{200}\left(\frac{55}{95}\log_2\frac{95}{55} + \frac{40}{95}\log_2\frac{95}{40}\right)$$

$$= 0.525\,(0.863) + 0.475\,(0.981)$$

$$= 0.9195$$



C_1	C_2
23	77

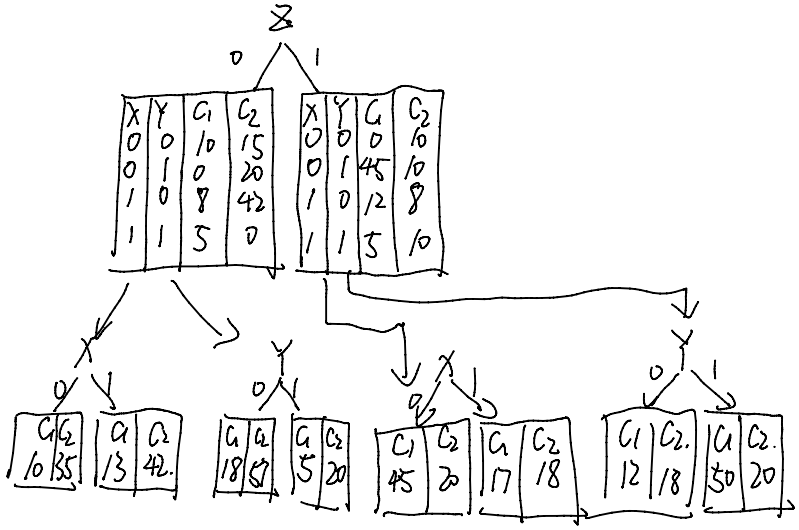
C_1	C_2
62	38

$$\text{Entropy} = 1/2\,(0.23\log_2\frac{100}{23} + 0.77\log_2\frac{100}{77}) + \frac{1}{2}\,(0.62\log_2\frac{100}{62} + 0.38\log_2\frac{100}{38})$$

$$= \frac{1}{2}\times 0.778 + \frac{1}{2}\times 0.958$$

$$= 0.868$$

⇒ we can first split by index Z



Z=0:

Entropy:

$$0.45\left(\frac{10}{45}\log_2\frac{45}{10} + \frac{35}{45}\log_2\frac{45}{35}\right) + 0.55\left(\frac{13}{55}\log_2\frac{55}{13} + \frac{42}{55}\log_2\frac{55}{42}\right)$$

$$= 0.344 + 0.434$$

$0.45\left(\frac{10}{45}\log_2\frac{45}{10}+\frac{35}{45}\log_2\frac{45}{35}\right)+0.55\left(\frac{13}{55}\log_2\frac{55}{13}+\frac{42}{55}\log_2\frac{55}{42}\right)$

$=0.344+0.434$

$=0.778$

Entropy Y_1: $0.75\left(\frac{18}{75}\log_2\frac{75}{18}+\frac{57}{75}\log_2\frac{75}{57}\right)+0.25\left(\frac{1}{5}\log_2 5+\frac{4}{5}\log_2\frac{5}{4}\right)$

$=0.846+0.180$

$=0.776$

$Z=1$:

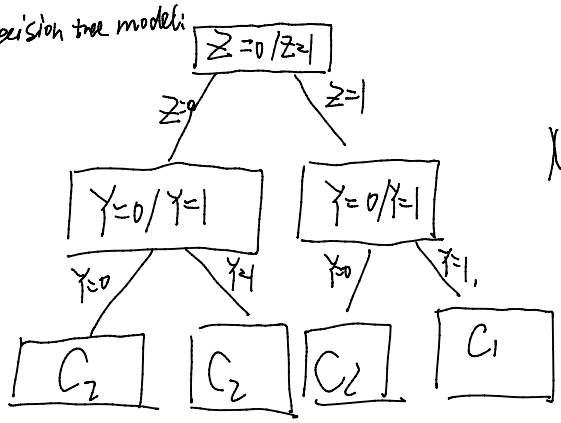
Entropy X: $0.65\left(\frac{45}{65}\log_2\frac{65}{45}+\frac{20}{65}\log_2\frac{65}{20}\right)+0.35\left(\frac{17}{35}\log_2\frac{35}{17}+\frac{18}{35}\log_2\frac{35}{18}\right)$

$=0.571+0.350=0.921$

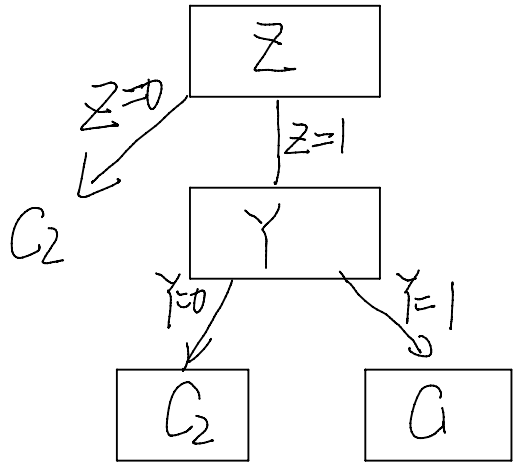
Entropy Y: $0.3\left(\frac{12}{30}\log_2\frac{30}{12}+\frac{18}{30}\log_2\frac{30}{18}\right)+0.7\left(\frac{58}{70}\log_2\frac{70}{58}+\frac{12}{70}\log_2\frac{70}{12}\right)$

$=0.291+0.604=0.895$

Decision tree model:



merge



problem 4, see the code block below

problem 5:

A	B	C	class
Y	15	3	Y
No	2	13	N
Y	6	6	Y
Y	3	14	N
Y	7	4	N

N	3	8	N
N	1	1	N
N	2	0	N
N	1	2	N
Y	10	8	Y

Code Block Answer:

In [2]:

```
import pandas as pd
import math

df = pd.read_csv("./Table1.csv", header = 0)
init_entropy = len(df[df["class"]=="n"])/9*math.log(9/len(df[df["class"]=="n"]),2)+len(df[df["class
"]=="y"])/9*math.log(9/len(df[df["class"]=="y"]),2)
df.head(10)
```

Out[2]:

	a1	a2	a3	class
0	T	T	1	n
1	T	T	4	n
2	T	F	5	y
3	F	F	4	n
4	F	T	7	y
5	F	T	6	y
6	F	F	8	y
7	T	F	7	n
8	F	T	3	y

In [ ]:

```
#(c) For a3, which is a continuous attribute, compute the information gain for every possible spli
t.
#(d) What is the best split (among a1, a2 and a3) according to the information gain?
#(e) What is the best split (between a1 and a2) according to the classification error rate?
#(f) What is the best split (between a1 and a2) according to the Gini index.
```

In [3]:

```
df[df["a1"]=="T"].head(10)
```

Out[3]:

	a1	a2	a3	class
0	T	T	1	n
1	T	T	4	n
2	T	F	5	y
7	T	F	7	n

In [42]:

```
for split in range(1,8):
  data_smaller_than_split = df[df["a3"]<=split]
  data_bigger_than_split = df[df["a3"]>split]

  num_of_small = len(data_smaller_than_split)
  num_of_big = len(data_bigger_than_split)

  num_smaller_y = len(data_smaller_than_split[data_smaller_than_split["class"]=="y"])
  if num_smaller_y == 0:
    num_smaller_y = 0.00000000000000000001
  num_smaller_n = len(data_smaller_than_split[data_smaller_than_split["class"]=="n"])

  num_bigger_y = len(data_bigger_than_split[data_bigger_than_split["class"]=="y"])
  num_bigger_n = len(data_bigger_than_split[data_bigger_than_split["class"]=="n"])
  if num_bigger_n == 0:
    num_bigger_n = 0.00000000000000000001
```

```
  entropy = num_of_small/9*(num_smaller_y/num_of_small*math.log(num_of_small/num_smaller_y, 2) + nu
m_smaller_n/num_of_small*math.log(num_of_small/num_smaller_n,2))+num_of_big/9*(num_bigger_y/num_of_
big*math.log(num_of_big/num_bigger_y,2)+num_bigger_n/num_of_big*math.log(num_of_big/num_bigger_n,2)
)
  info_gain = init_entropy - entropy
  print("Split Point:", split, "Entropy:%05f"%entropy, "Info Gain:%05f "%info_gain)
```

```
Split Point: 1 Entropy:0.848386 Info Gain:0.142690
Split Point: 2 Entropy:0.848386 Info Gain:0.142690
Split Point: 3 Entropy:0.988511 Info Gain:0.002565
Split Point: 4 Entropy:0.761639 Info Gain:0.229437
Split Point: 5 Entropy:0.899985 Info Gain:0.091091
Split Point: 6 Entropy:0.972765 Info Gain:0.018311
Split Point: 7 Entropy:0.888889 Info Gain:0.102187
```

In [ ]:

```
#So the Best Split point is to split at the point 4.5 with a info gain 0.229437.
```

In [5]:

```
df_a1_T = df[df["a1"]=="T"]
df_a1_F = df[df["a1"]=="F"]
df_a2_T = df[df["a2"]=="T"]
df_a2_F = df[df["a2"]=="F"]

num_a1_T = len(df_a1_T)
num_a1_F = len(df_a1_F)
num_a2_T = len(df_a2_T)
num_a2_F = len(df_a2_F)

num_a1_T_y = len(df_a1_T[df_a1_T["class"]=="y"])
num_a1_T_n = len(df_a1_T[df_a1_T["class"]=="n"])
num_a1_F_y = len(df_a1_F[df_a1_F['class']== 'y'])
num_a1_F_n = len(df_a1_F[df_a1_F['class']=='n'])

num_a2_T_y = len(df_a2_T[df_a2_T["class"]=="y"])
num_a2_T_n = len(df_a2_T[df_a2_T["class"]=="n"])
num_a2_F_y = len(df_a2_F[df_a2_F['class']== 'y'])
num_a2_F_n = len(df_a2_F[df_a2_F['class']=='n'])


print(num_a1_T_y)
Entropy_a1 = num_a1_T/9*(num_a1_T_y/num_a1_T*math.log(num_a1_T/num_a1_T_y,2)+num_a1_T_n/num_a1_T*ma
th.log(num_a1_T/num_a1_T_n,2))+num_a1_F/9*(num_a1_F_y/num_a1_F*math.log(num_a1_F/num_a1_F_y,2)+num_
a1_F_n/num_a1_F*math.log(num_a1_F/num_a1_F_n,2))
print("split with a1's Entropy:%05f"%Entropy_a1,"info_gain: %05f"%(init_entropy- Entropy_a1))
Entropy_a2 = num_a2_T/9*(num_a2_T_y/num_a2_T*math.log(num_a2_T/num_a2_T_y,2)+num_a2_T_n/num_a2_T*ma
th.log(num_a2_T/num_a2_T_n,2))+num_a2_F/9*(num_a2_F_y/num_a2_F*math.log(num_a2_F/num_a2_F_y,2)+num_
a2_F_n/num_a2_F*math.log(num_a2_F/num_a2_F_n,2))
print("split with a2's Entropy:%05f"%Entropy_a2, "info_gain: %05f"%(init_entropy - Entropy_a2))
```

```
1
split with a1's Entropy:0.761639 info_gain: 0.229437
split with a2's Entropy:0.983861 info_gain: 0.007215
```

In [ ]:

```
#To split the data with a1 and a3's 4.5 split point will get the same best entropy.
```

In [8]:

```
GINI_a1 = num_a1_T/len(df)*(1-(num_a1_T_y/num_a1_T)**2-(num_a1_T_n/num_a1_T)**2)+num_a1_F/len(df)*(
1-(num_a1_F_y/num_a1_F)**2-(num_a1_F_n/num_a1_F)**2)
print("Gini index of a1: %05f"%GINI_a1)
```

```
Gini index of a1: 0.344444
```

In [9]:

```
GINI_a2 = num_a2_T/len(df)*(1-(num_a2_T_y/num_a2_T)**2-(num_a2_T_n/num_a2_T)**2)+num_a2_F/len(df)*(
1-(num_a2_F_y/num_a2_F)**2-(num_a2_F_n/num_a2_F)**2)
print("Gini index of a2: %05f" %GINI_a2)
```

```
Gini index of a2: 0.488889
```

Open in Colab

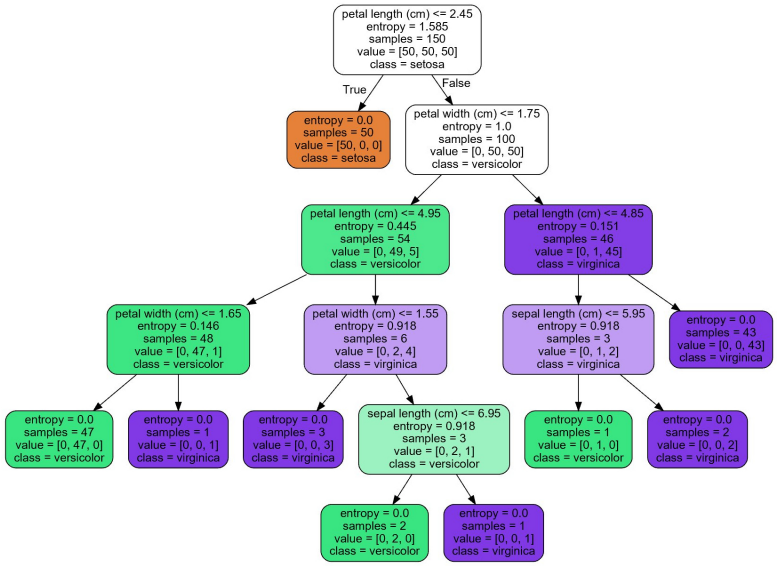
In [48]:
```
from sklearn.datasets import load_iris
from sklearn import tree

iris = load_iris()
clf = tree.DecisionTreeClassifier(criterion= "entropy",)
clf = clf.fit(iris.data, iris.target)

import graphviz
dot_data = tree.export_graphviz(clf, out_file=None)
graph = graphviz.Source(dot_data)
graph.render("iris")

dot_data = tree.export_graphviz(clf, out_file=None,
                      feature_names=iris.feature_names,
                      class_names=iris.target_names,
                      filled=True, rounded=True,)
graph = graphviz.Source(dot_data)
graph
```

Out[48]:



In [0]:
```
from google.colab import drive
drive.mount('/content/drive')
```

In [46]:
```
import os
import pandas as pd

df = pd.read_csv("./bank.csv",delimiter = ";")
df.head()
```

Out[46]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1	-1

In [0]:
```
le = preprocessing.LabelEncoder()
for column_name in df.columns:
    if df[column_name].dtype == object:
        df[column_name] = le.fit_transform(df[column_name])
    else:
        pass
```

In [43]:
```
from sklearn import tree
from sklearn.model_selection import train_test_split

input_data = df[["age","job","marital","education","default","balance","housing","loan","day","month","dur
ation","campaign","pdays","previous","poutcome"]].values
label = df["y"].values
input_data
```

Out[43]:
```
array([[ 30,  10,   1, ...,  -1,   0,   3],
       [ 33,   7,   1, ..., 339,   4,   0],
       [ 35,   4,   2, ..., 330,   1,   0],
       ...,
       [ 57,   9,   1, ...,  -1,   0,   3],
       [ 28,   1,   1, ..., 211,   3,   1],
       [ 44,   2,   2, ..., 249,   7,   1]])
```

In [49]:
```
from sklearn import preprocessing

(train_inputs, test_inputs, train_classes, test_classes) = train_test_split(input_data,label, train_size=
0.7, random_state=1)
Dtree = tree.DecisionTreeClassifier(criterion= "entropy" , max_depth= 5)
Dtree = Dtree.fit(train_inputs, train_classes)

import graphviz
bank_dot_data = tree.export_graphviz(Dtree, out_file=None)
graph_bank = graphviz.Source(bank_dot_data)
graph_bank.render("y")

bank_dot_data = tree.export_graphviz(Dtree, out_file=None,
                      filled=True, rounded=True,)
graph = graphviz.Source(bank_dot_data)
graph
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_split.py:2179: FutureWarning: From version
0.21, test_size will always complement train_size unless both are specified.
  FutureWarning)
```

Out[49]: