A data science project codes in Python

January 3, 2020

```
[1]: import urllib2
```

```
[2]: url = "https://archive.ics.uci.edu/ml/machine-learning-databases/00244/
     ↪fertility_Diagnosis.txt"
```

```
[3]: sperm_p = urllib2.Request(url)
```

```
[4]: sperm = urllib2.urlopen(sperm_p)
```

```
[5]: import pandas as pd
```

```
[6]: fertility = pd.read_csv(sperm, sep=',', decimal='.', header=None ,␣
     ↪names=['Season','Age','Childish_diseases','Accident','Surgical_intervention','High_fever_la
```

```
[ ]:
```

```
[7]: fertility
```

```
[7]:    Season   Age  Childish_diseases  Accident  Surgical_intervention  \
     0    -0.33  0.69                  0         1                      1
     1    -0.33  0.94                  1         0                      1
     2    -0.33  0.50                  1         0                      0
     3    -0.33  0.75                  0         1                      1
     4    -0.33  0.67                  1         1                      0
     5    -0.33  0.67                  1         0                      1
     6    -0.33  0.67                  0         0                      0
     7    -0.33  1.00                  1         1                      1
     8     1.00  0.64                  0         0                      1
     9     1.00  0.61                  1         0                      0
     10    1.00  0.67                  1         1                      0
     11    1.00  0.78                  1         1                      1
     12    1.00  0.75                  1         1                      1
     13    1.00  0.81                  1         0                      0
     14    1.00  0.94                  1         1                      1
     15    1.00  0.81                  1         1                      0
     16    1.00  0.64                  1         0                      1
     17    1.00  0.69                  1         0                      1
```

```
18    1.00  0.75                  1         1                      1
19    1.00  0.67                  1         0                      0
20    1.00  0.67                  0         0                      1
21    1.00  0.75                  1         0                      0
22    1.00  0.67                  1         1                      0
23    1.00  0.69                  1         0                      1
24    1.00  0.56                  1         0                      1
25    1.00  0.67                  1         0                      0
26    1.00  0.67                  1         0                      1
27    1.00  0.78                  1         1                      0
28    1.00  0.58                  0         0                      1
29    1.00  0.67                  0         0                      1
..     …   …                …       …                    …
70   -0.33  0.50                  1         1                      0
71    0.33  0.69                  1         0                      0
72    1.00  0.56                  1         0                      0
73   -1.00  0.50                  1         0                      0
74   -1.00  0.53                  1         0                      0
75   -1.00  0.78                  1         0                      1
76   -1.00  0.75                  1         0                      1
77   -1.00  0.72                  1         1                      1
78   -1.00  0.53                  1         1                      0
79   -1.00  1.00                  1         0                      1
80   -0.33  0.92                  1         1                      0
81   -1.00  0.81                  1         1                      1
82   -0.33  0.92                  1         0                      0
83   -0.33  0.86                  1         1                      1
84   -0.33  0.78                  1         0                      0
85   -0.33  0.89                  1         1                      0
86   -0.33  0.75                  1         1                      1
87   -0.33  0.75                  1         1                      1
88   -0.33  0.83                  1         1                      1
89   -0.33  0.81                  1         1                      1
90   -0.33  0.81                  1         1                      1
91    0.33  0.78                  1         0                      0
92    0.33  0.75                  1         1                      0
93    0.33  0.75                  1         0                      1
94    1.00  0.58                  1         0                      0
95   -1.00  0.67                  1         0                      0
96   -1.00  0.61                  1         0                      0
97   -1.00  0.67                  1         1                      1
98   -1.00  0.64                  1         0                      1
99   -1.00  0.69                  0         1                      1

    High_fever_last_year  Alcohol_frequency  Smoking_habit  Hours_sitting  \
0                      0                0.8              0           0.88
1                      0                0.8              1           0.31
```

2	0	1.0	-1	0.50
3	0	1.0	-1	0.38
4	0	0.8	-1	0.50
5	0	0.8	0	0.50
6	-1	0.8	-1	0.44
7	0	0.6	-1	0.38
8	0	0.8	-1	0.25
9	0	1.0	-1	0.25
10	-1	0.8	0	0.31
11	0	0.6	0	0.13
12	0	0.8	1	0.25
13	0	1.0	-1	0.38
14	0	0.2	-1	0.25
15	0	1.0	1	0.50
16	0	1.0	-1	0.38
17	0	0.8	-1	0.25
18	0	1.0	1	0.25
19	0	0.8	1	0.38
20	0	0.8	-1	0.25
21	0	0.6	0	0.25
22	0	0.8	-1	0.25
23	-1	1.0	-1	0.44
24	0	1.0	-1	0.63
25	0	1.0	-1	0.25
26	0	0.6	-1	0.38
27	1	0.6	-1	0.38
28	0	1.0	-1	0.19
29	0	0.6	0	0.50
..	…	…	…	…
70	-1	0.8	0	0.88
71	1	1.0	-1	0.31
72	1	0.6	0	0.50
73	1	0.8	-1	0.44
74	1	0.8	-1	0.63
75	1	1.0	1	0.25
76	1	0.6	0	0.56
77	1	0.8	-1	0.19
78	1	0.8	-1	0.38
79	1	0.6	0	0.25
80	1	1.0	-1	0.63
81	1	0.8	0	0.19
82	1	0.6	-1	0.19
83	1	1.0	-1	0.25
84	1	1.0	1	0.06
85	0	0.6	1	0.31
86	0	0.6	1	0.25
87	1	0.8	1	0.25

```
88                     0                1.0              -1           0.31
89                     0                1.0               1           0.38
90                     1                0.8              -1           0.38
91                     0                1.0               1           0.06
92                     0                0.8              -1           0.38
93                     0                0.8              -1           0.44
94                     0                0.6               1           0.50
95                     0                1.0              -1           0.50
96                     0                0.8               0           0.50
97                     0                1.0              -1           0.31
98                     0                1.0               0           0.19
99                     0                0.6              -1           0.19

   Output
0       N
1       O
2       N
3       N
4       O
5       N
6       N
7       N
8       N
9       N
10      N
11      N
12      N
13      N
14      N
15      N
16      N
17      O
18      N
19      O
20      N
21      N
22      N
23      O
24      N
25      N
26      O
27      O
28      N
29      O
..    …
70      O
71      N
```

```
72      N
73      N
74      N
75      N
76      N
77      N
78      N
79      N
80      N
81      N
82      N
83      N
84      O
85      N
86      N
87      N
88      N
89      N
90      N
91      N
92      N
93      O
94      N
95      N
96      N
97      N
98      N
99      N

[100 rows x 10 columns]
```

[8]:

```
fertility.dtypes
```

[8]:

```
Season                    float64
Age                       float64
Childish_diseases           int64
Accident                    int64
Surgical_intervention       int64
High_fever_last_year        int64
Alcohol_frequency         float64
Smoking_habit               int64
Hours_sitting             float64
Output                     object
dtype: object
```

[9]:

```
fertility["Output"]=fertility["Output"].astype('category')
```

[10]:

```
fertility.dtypes
```

[10]:

```
Season                     float64
Age                        float64
Childish_diseases            int64
Accident                     int64
Surgical_intervention        int64
High_fever_last_year         int64
Alcohol_frequency          float64
Smoking_habit                int64
Hours_sitting              float64
Output                    category
dtype: object
```

[11]:

```
for col in fertility.columns:
    if (fertility[col].dtype.name == 'float64'):
        print fertility[col].value_counts(), '\n'
```

```
-0.33    37
 1.00    31
-1.00    28
 0.33     4
Name: Season, dtype: int64

0.67    14
0.56    12
0.75    10
0.53     9
0.78     7
0.58     7
0.50     7
0.69     7
0.64     6
0.81     5
0.61     5
0.94     2
0.72     2
0.92     2
1.00     2
0.89     1
0.83     1
0.86     1
Name: Age, dtype: int64

1.0    40
0.8    39
0.6    19
```

```
0.2     1
0.4     1
Name: Alcohol_frequency, dtype: int64

0.25    17
0.50    16
0.38    13
0.19    11
0.31    11
0.63    10
0.44     9
0.88     3
0.75     3
0.56     2
0.06     2
0.13     1
0.47     1
1.00     1
Name: Hours_sitting, dtype: int64
```

[12]:

```
for col in fertility.columns:
    if (fertility[col].dtype.name == 'int64'):
        print fertility[col].value_counts(), '\n'
```

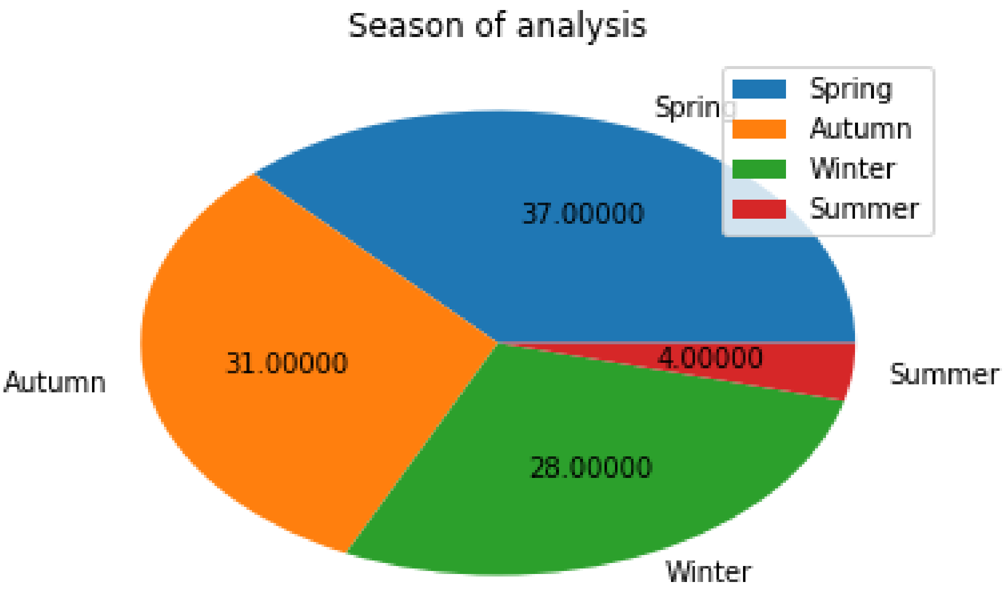
```
1    87
0    13
Name: Childish_diseases, dtype: int64

0    56
1    44
Name: Accident, dtype: int64

1    51
0    49
Name: Surgical_intervention, dtype: int64

 0    63
 1    28
-1     9
Name: High_fever_last_year, dtype: int64

-1    56
 0    23
 1    21
Name: Smoking_habit, dtype: int64
```

[13]:
```
fertility['Output'].value_counts()
```

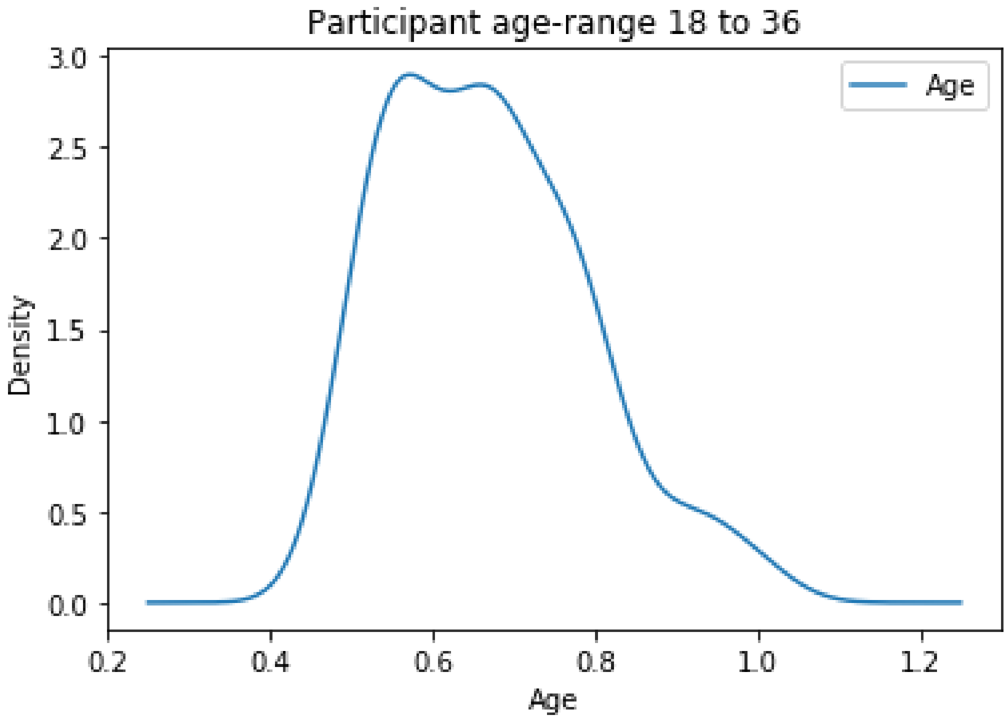
[13]:
```
N    88
O    12
Name: Output, dtype: int64
```

[14]:
```
#Task 2
import matplotlib.pyplot as plt
```

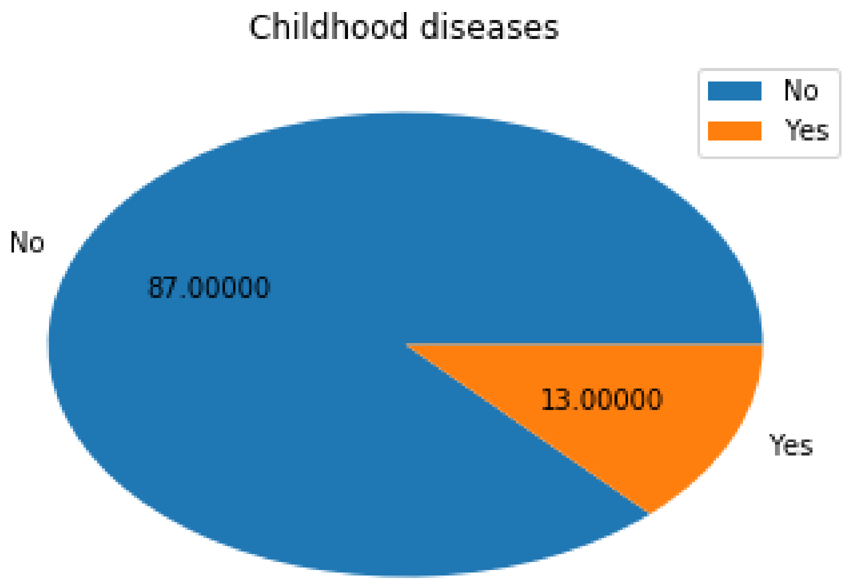
[15]:
```
fertility['Season'].value_counts().
 ↪plot(kind='pie',labels=['Spring','Autumn','Winter','Summer'], autopct='%.5f')
plt.title('Season of analysis')
plt.legend(loc='best')
plt.ylabel(' ')
plt.show()
```



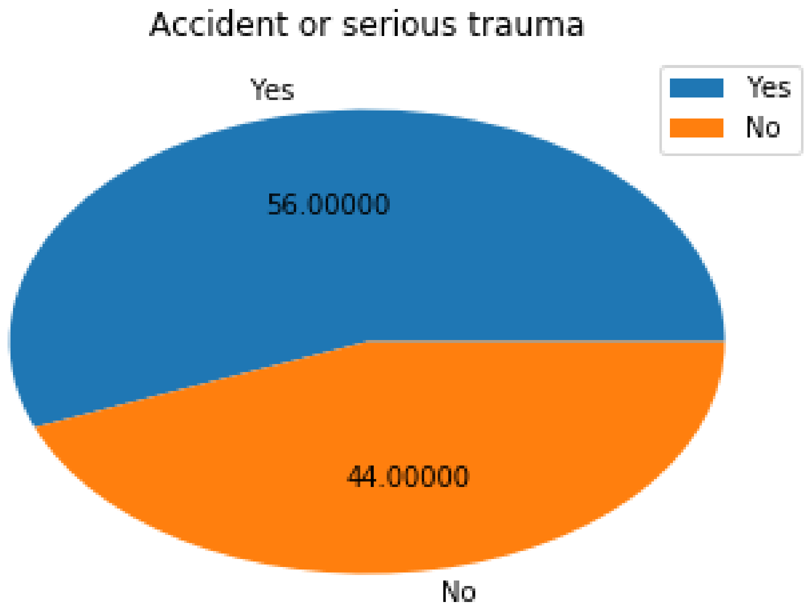
[16]:
```
fertility['Age'].plot(kind='density',x=[18,36])
plt.title('Participant age-range 18 to 36')
plt.xlabel('Age')
plt.legend()
plt.show()
```



```
[17]: fertility['Childish_diseases'].value_counts().
       ↪plot(kind='pie',labels=['No','Yes'], autopct='%.5f')
      plt.title('Childhood diseases')
      plt.legend(loc='best')
      plt.ylabel(' ')
      plt.show()
```

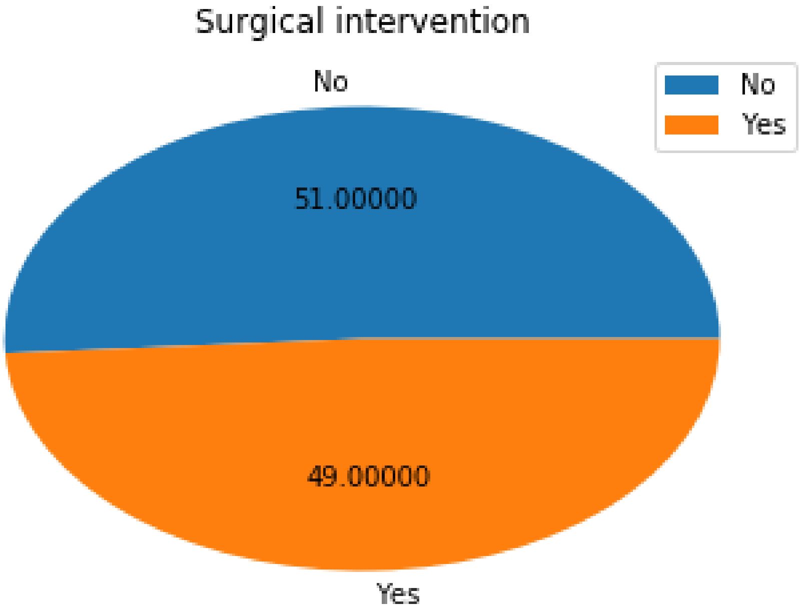


```
[18]: fertility['Accident'].value_counts().plot(kind='pie',labels=['Yes','No'], autopct='%.5f')
      plt.title('Accident or serious trauma')
      plt.legend(loc='best')
      plt.ylabel(' ')
      plt.show()
```



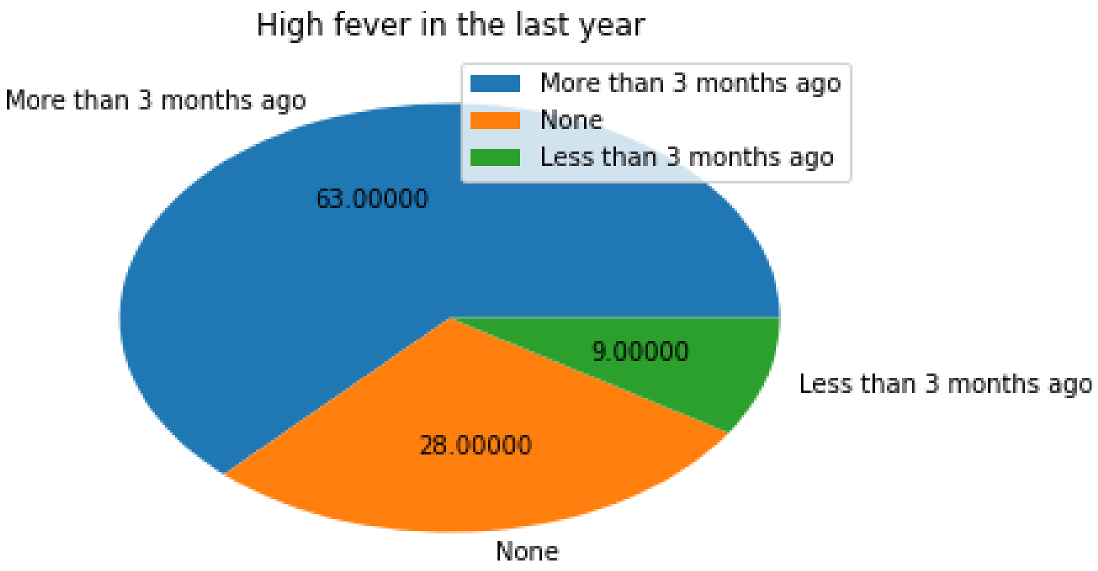
[19]:

```
fertility['Surgical_intervention'].value_counts().
 ↪plot(kind='pie',labels=['No','Yes'], autopct='%.5f')
plt.title('Surgical intervention')
plt.legend(loc='best')
plt.ylabel(' ')
plt.show()
```

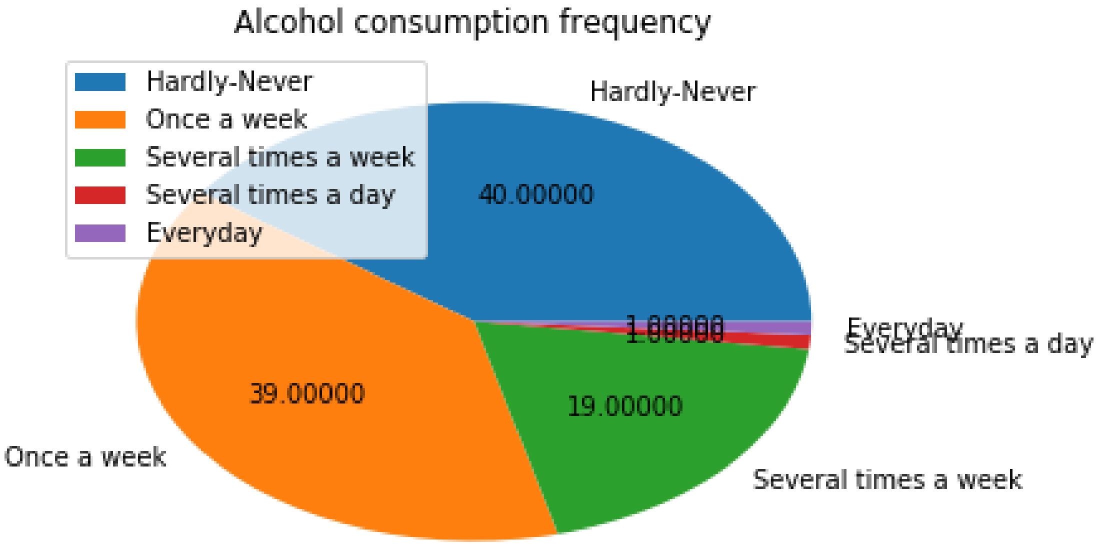


[20]:

```
fertility['High_fever_last_year'].value_counts().plot(kind='pie',labels=['More␣
 ↪than 3 months ago','None','Less than 3 months ago'], autopct='%.5f')
plt.title('High fever in the last year')
plt.legend(loc='best')
plt.ylabel(' ')
plt.show()
```

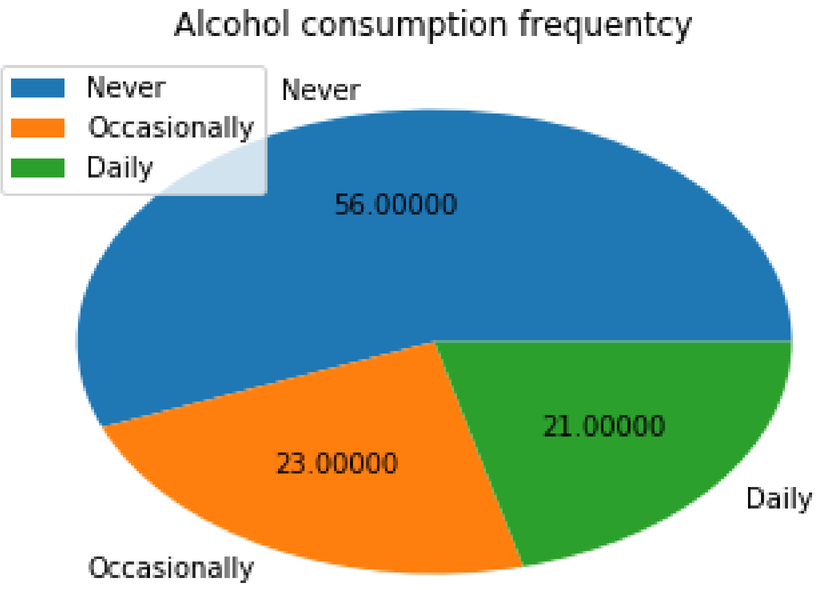


```
[21]: fertility['Alcohol_frequency'].value_counts().
       ↪plot(kind='pie',labels=['Hardly-Never','Once a week','Several times a␣
       ↪week','Several times a day','Everyday'], autopct='%.5f')
      plt.title('Alcohol consumption frequency')
      plt.legend(loc='upper left')
      plt.ylabel(' ')
      plt.show()
```



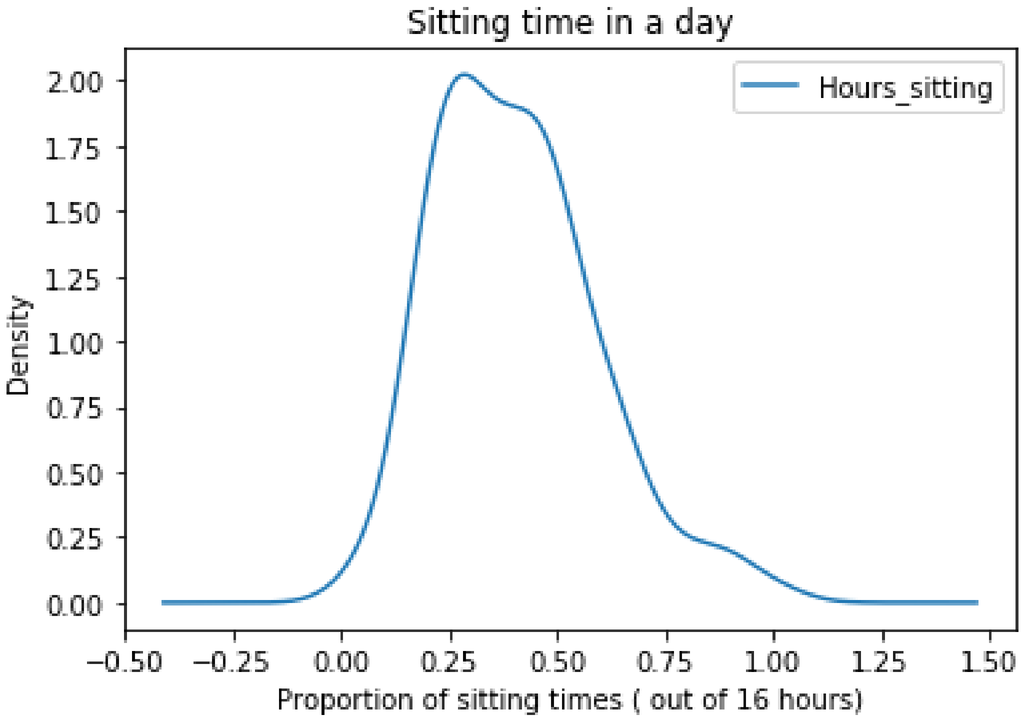
[22]:

```
fertility['Smoking_habit'].value_counts().
 ↪plot(kind='pie',labels=['Never','Occasionally','Daily'], autopct='%.5f')
plt.title('Alcohol consumption frequentcy')
plt.legend(loc='upper left')
plt.ylabel(' ')
plt.show()
```

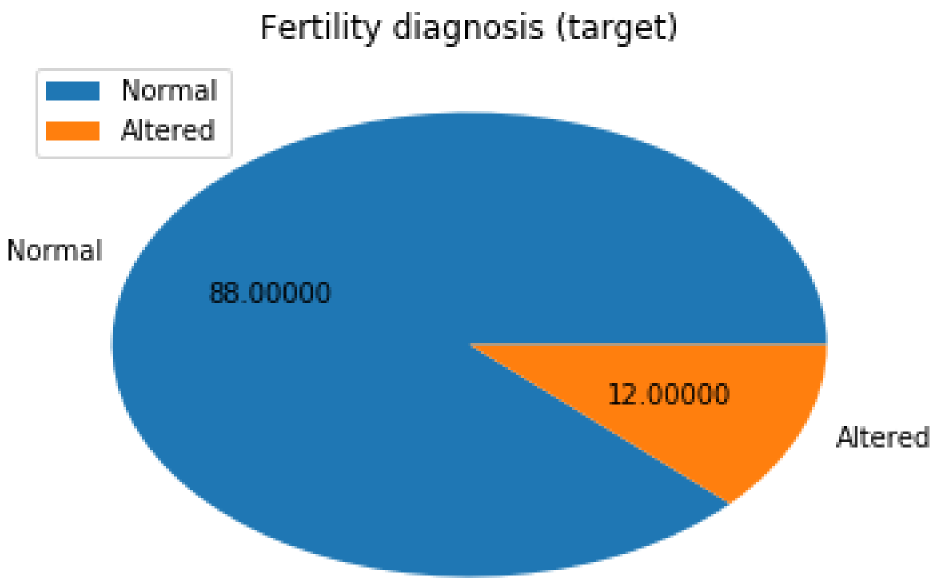


[23]:

```
fertility['Hours_sitting'].plot(kind='density')
plt.title('Sitting time in a day')
plt.xlabel('Proportion of sitting times ( out of 16 hours)')
plt.legend()
plt.show()
```

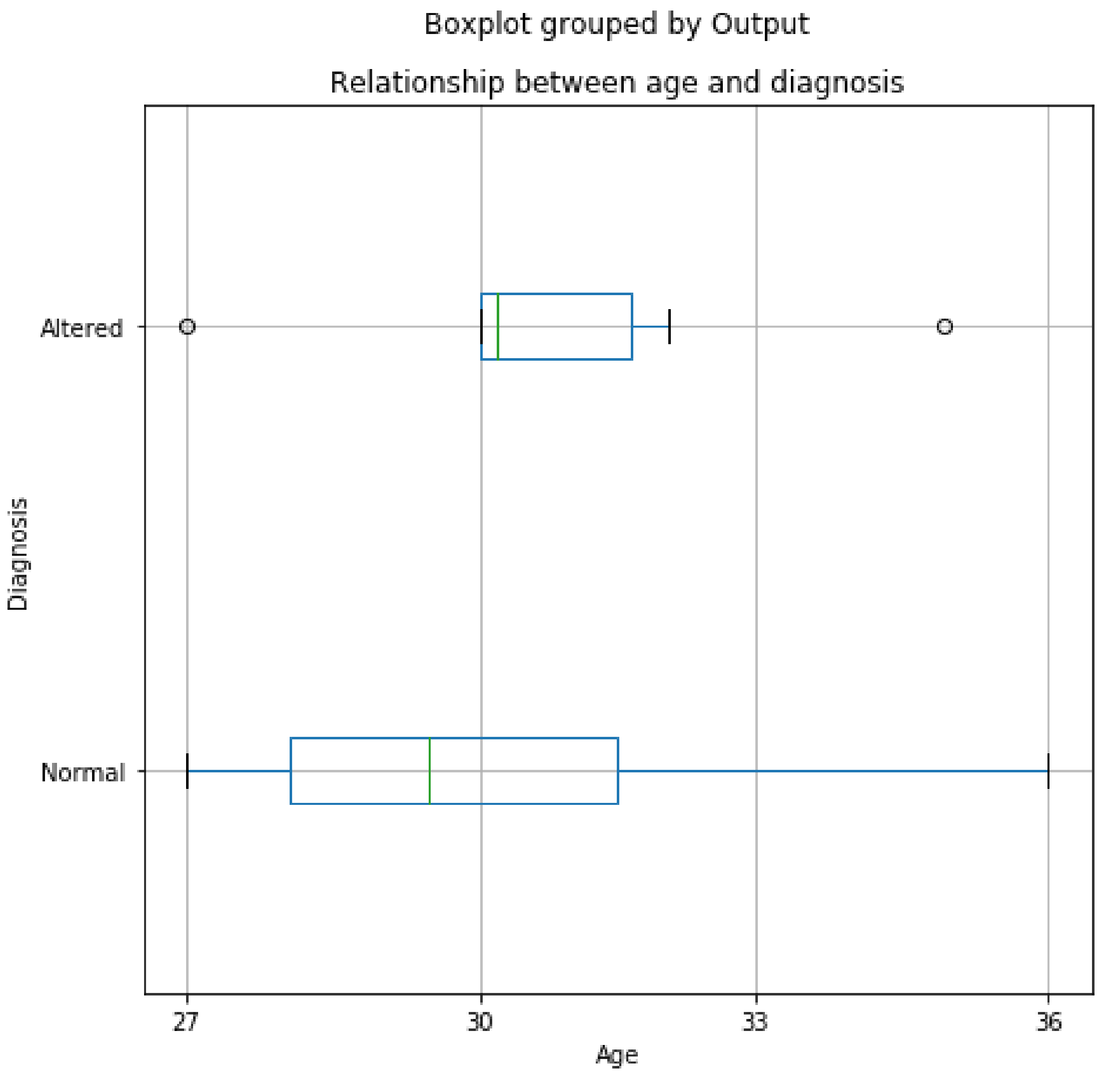


[24]:
```
fertility['Output'].value_counts().plot(kind='pie',labels=('Normal','Altered'),␣
 ↪autopct='%.5f')
plt.title('Fertility diagnosis (target)')
plt.legend(loc='upper left')
plt.ylabel(' ')
plt.show()
```



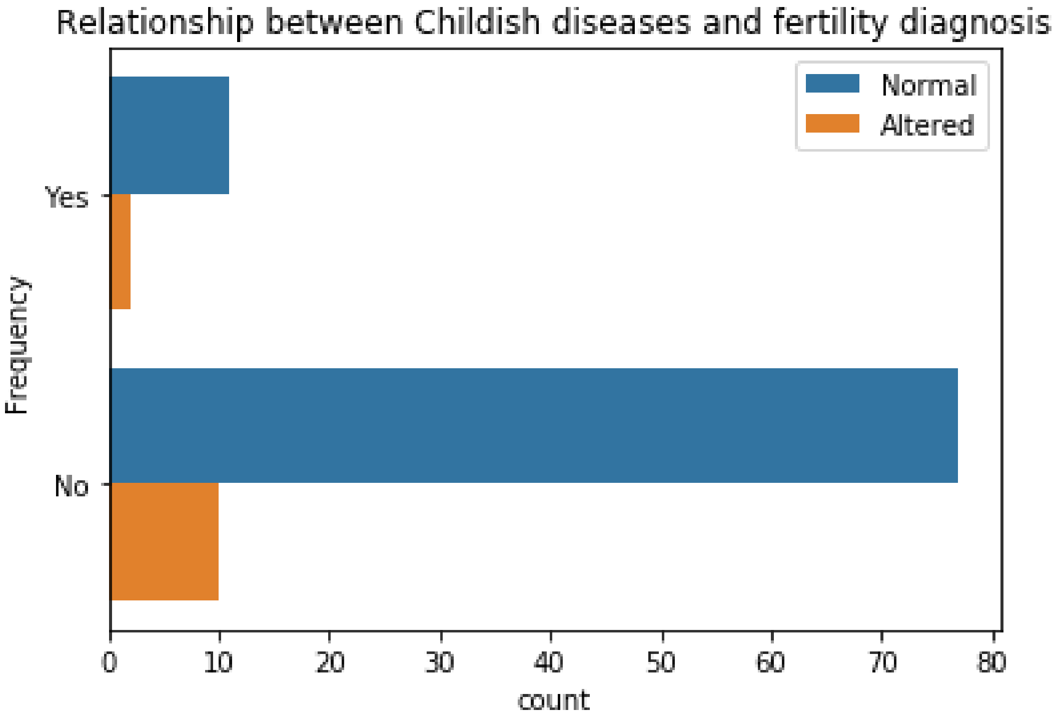
[25]:

```
ax1=fertility.dropna().boxplot(column='Age',vert= False, by='Output',figsize=(7,7),fontsize=10)
plt.title('Relationship between age and diagnosis')
plt.xlabel('Age')
ax1.set_yticklabels(['Normal','Altered'])
plt.ylabel('Diagnosis')
ax1.set_xticks([0.5,0.67,0.83,1])
ax1.set_xticklabels(['27','30','33','36'])
plt.show()
```

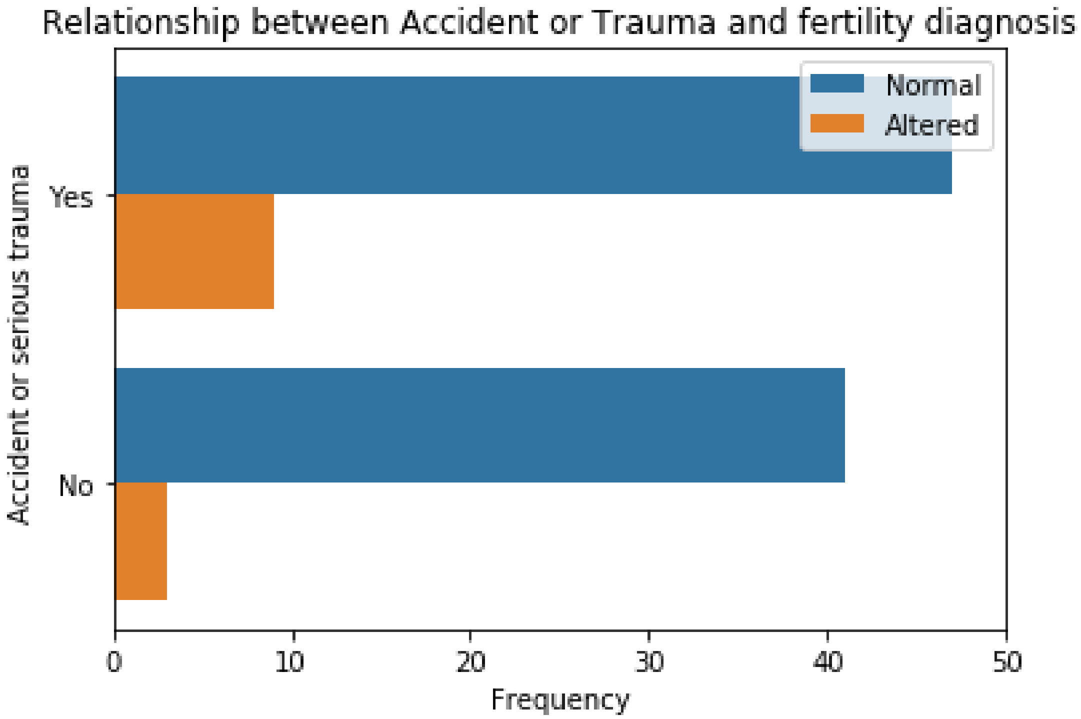


```
[26]: import seaborn as sns
```

```
[27]: ax2= sns.countplot(y='Childish_diseases', hue="Output", data=fertility)
      ax2.set_yticklabels(['Yes','No'])
      plt.ylabel('Childish diseases')
      plt.ylabel('Frequency')
      plt.legend(('Normal', 'Altered'),loc='upper right')
      plt.title('Relationship between Childish diseases and fertility diagnosis')
      plt.show()
```

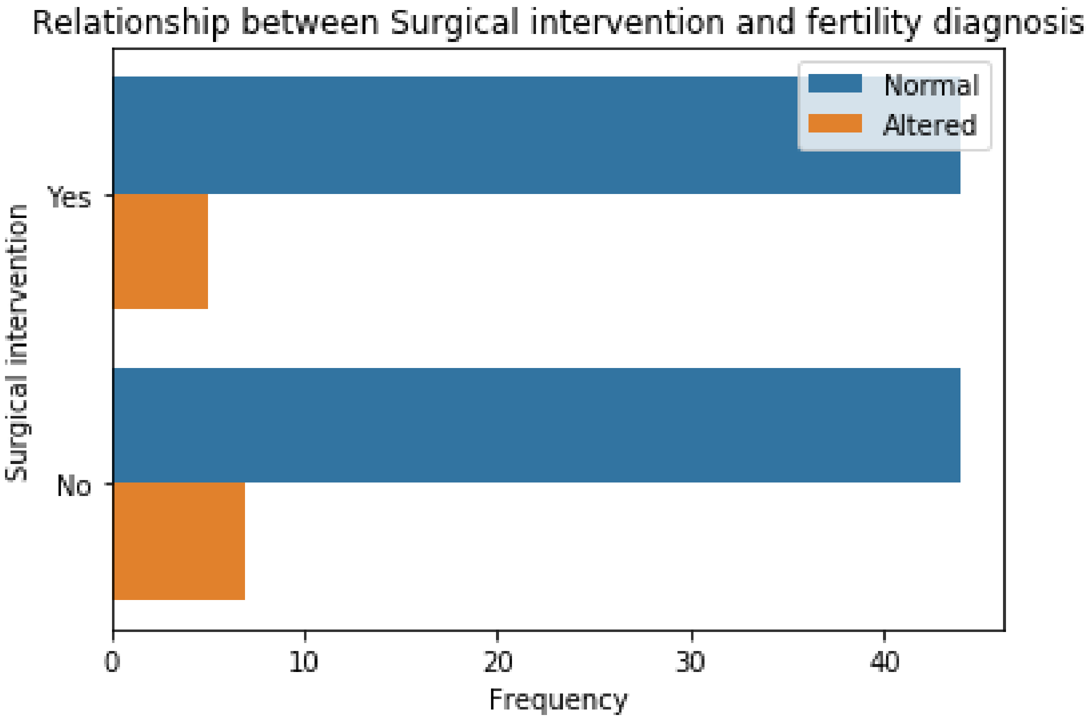


[28]:
```
ax3=sns.countplot(y='Accident', hue="Output", data=fertility)
ax3.set_yticklabels(['Yes','No'])
plt.ylabel('Accident or serious trauma')
plt.xlabel('Frequency')
ax3.set_xticks([0,10,20,30,40,50])
plt.legend(('Normal', 'Altered'),loc='upper right')
plt.title('Relationship between Accident or Trauma and fertility diagnosis')
plt.show()
```

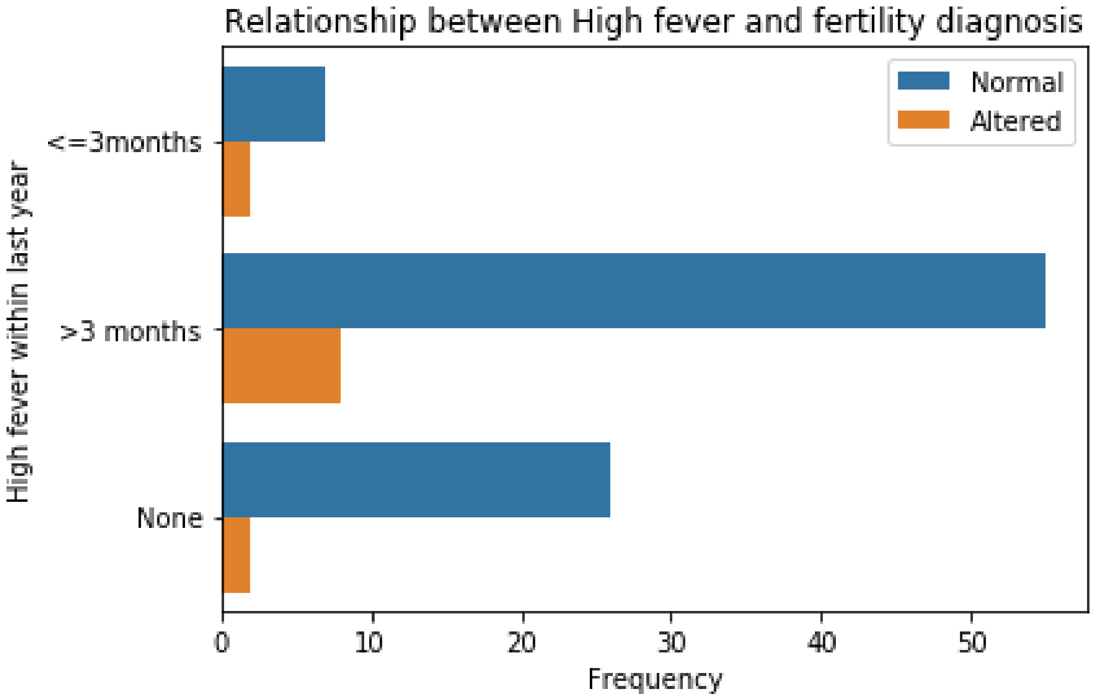


[29]:

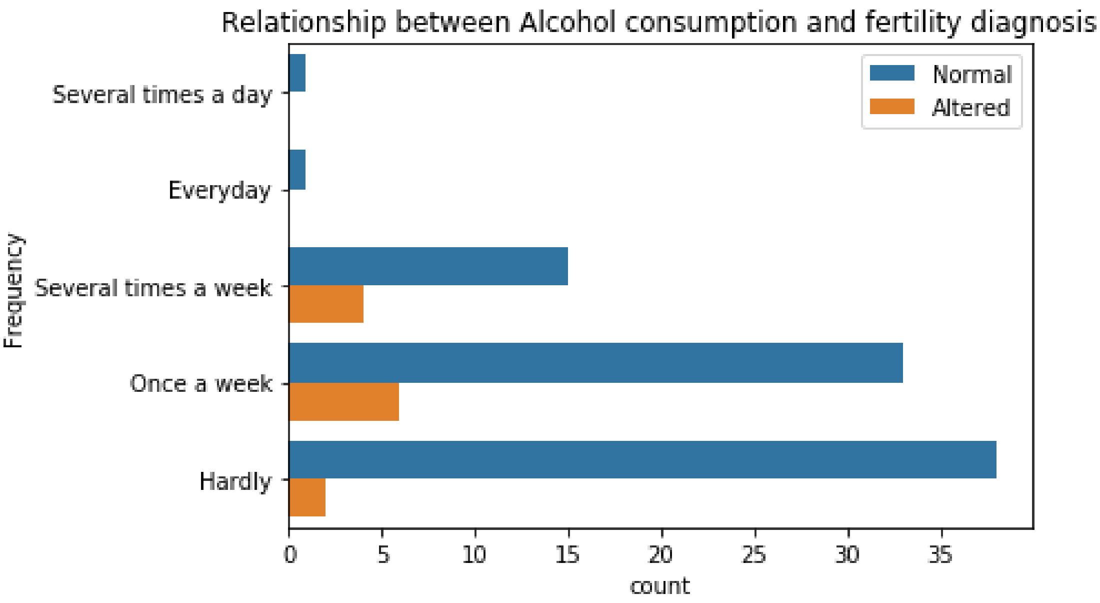
```
ax4=sns.countplot(y='Surgical_intervention', hue="Output", data=fertility)
ax4.set_yticklabels(['Yes','No'])
plt.ylabel('Surgical intervention')
plt.xlabel('Frequency')
plt.legend(('Normal', 'Altered'),loc='upper right')
plt.title('Relationship between Surgical intervention and fertility diagnosis')
plt.show()
```



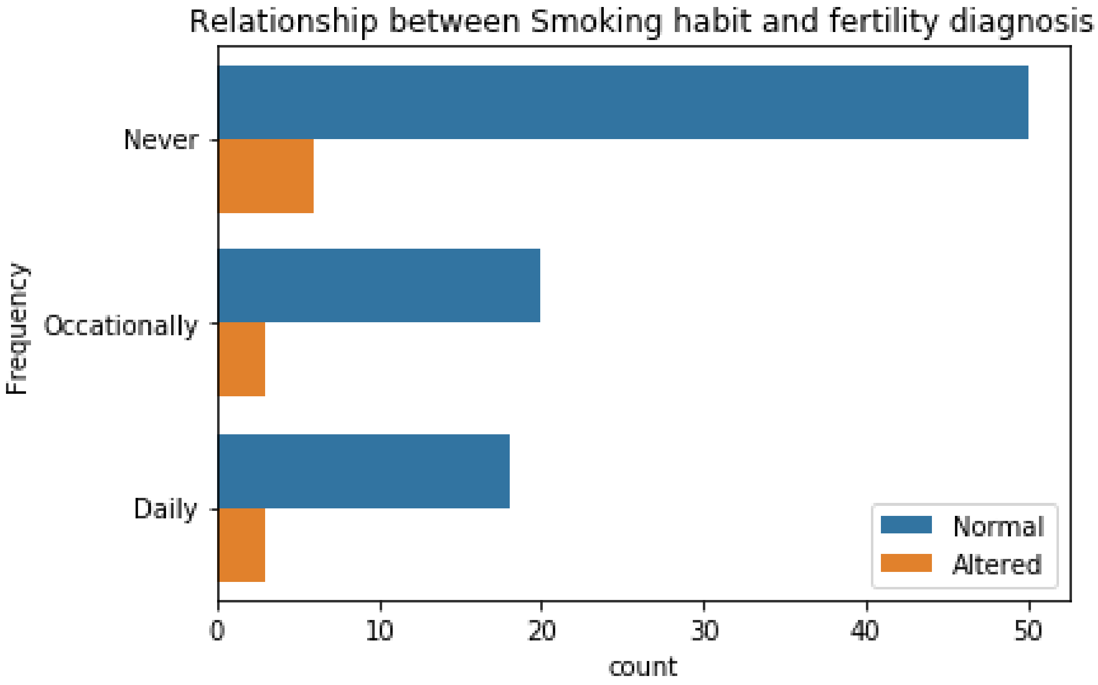
[30]:
```
ax5=sns.countplot(y='High_fever_last_year', hue="Output", data=fertility)
ax5.set_yticklabels(['<=3months','>3 months','None'])
plt.ylabel('High fever within last year')
plt.xlabel('Frequency')
plt.legend(('Normal', 'Altered'),loc='upper right')
plt.title('Relationship between High fever and fertility diagnosis')
plt.show()
```



```
[31]: ax6=sns.countplot(y='Alcohol_frequency', hue="Output", data=fertility)
      ax6.set_yticklabels(['Several times a day','Everyday','Several times a␣
       ↪week','Once a week','Hardly'])
      plt.ylabel('Surgical intervention')
      plt.ylabel('Frequency')
      plt.legend(('Normal', 'Altered'),loc='upper right')
      plt.title('Relationship between Alcohol consumption and fertility diagnosis')
      plt.show()
```

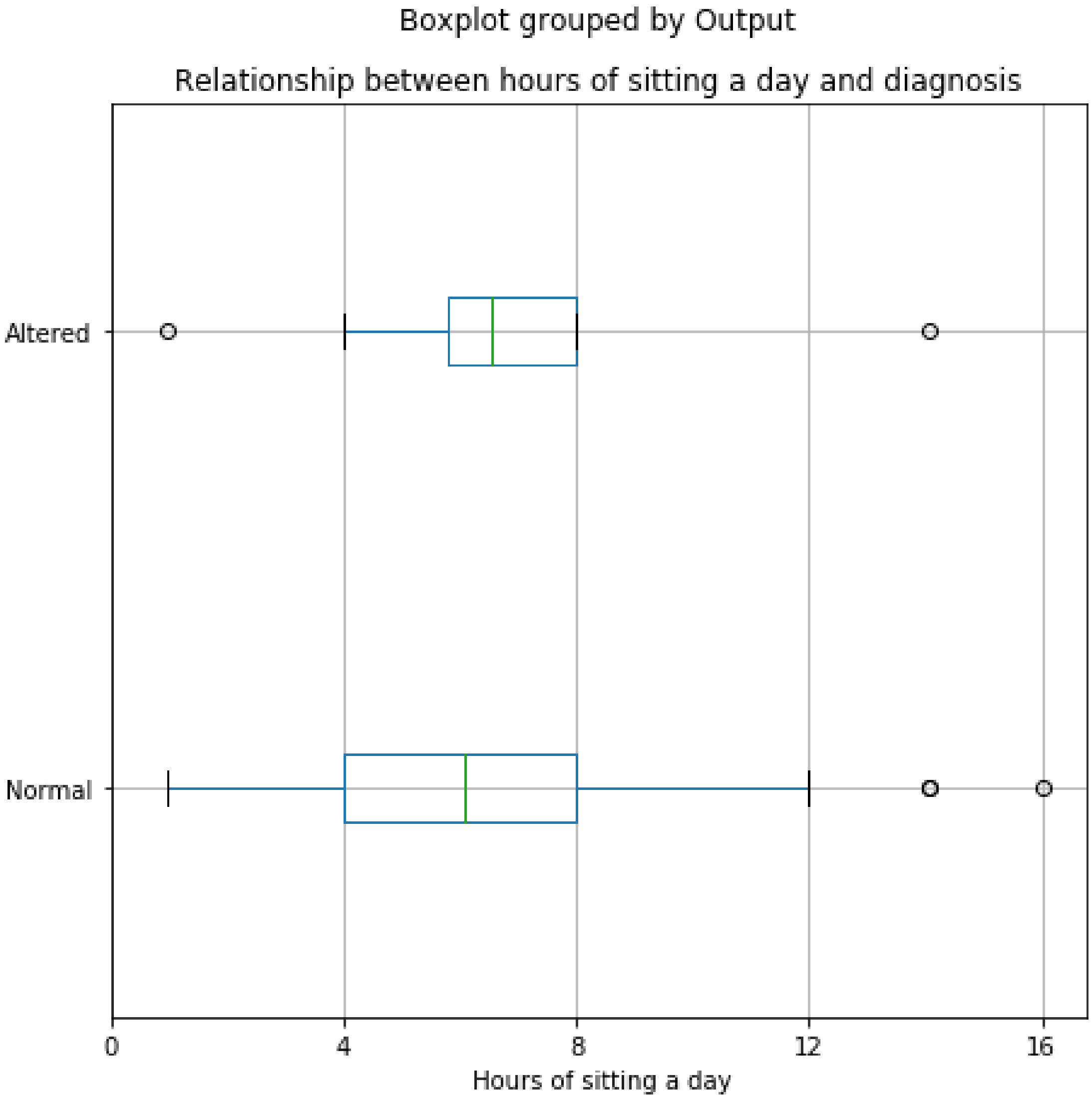


```
[32]: ax7=sns.countplot(y='Smoking_habit', hue="Output", data=fertility)
      ax7.set_yticklabels(['Never','Occationally','Daily'])
      plt.ylabel('Smoking habit')
      plt.ylabel('Frequency')
      plt.legend(('Normal', 'Altered'))
      plt.title('Relationship between Smoking habit and fertility diagnosis')
      plt.show()
```



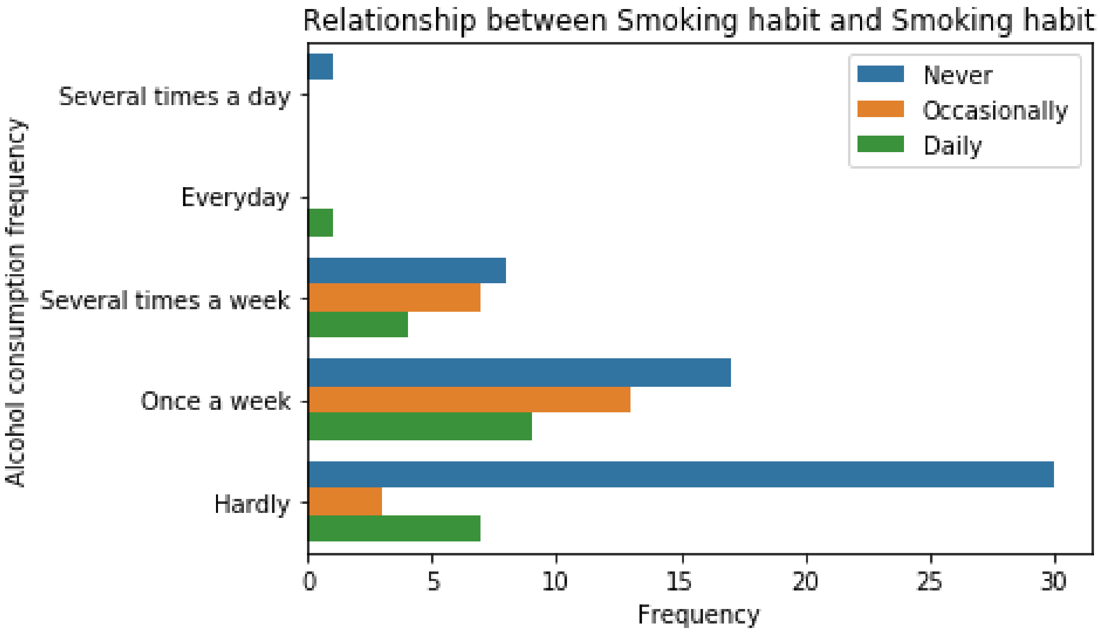
[33]:

```
ax8=fertility.dropna().boxplot(column='Hours_sitting',vert= False,␣
 ↪by='Output',figsize=(7,7),fontsize=10)
plt.title('Relationship between hours of sitting a day and diagnosis')
ax8.set_yticklabels(['Normal','Altered'])
ax8.set_xticks([0,0.25,0.5,0.75,1])
ax8.set_xticklabels(['0','4','8','12','16'])
plt.xlabel(' Hours of sitting a day')
plt.show()
```



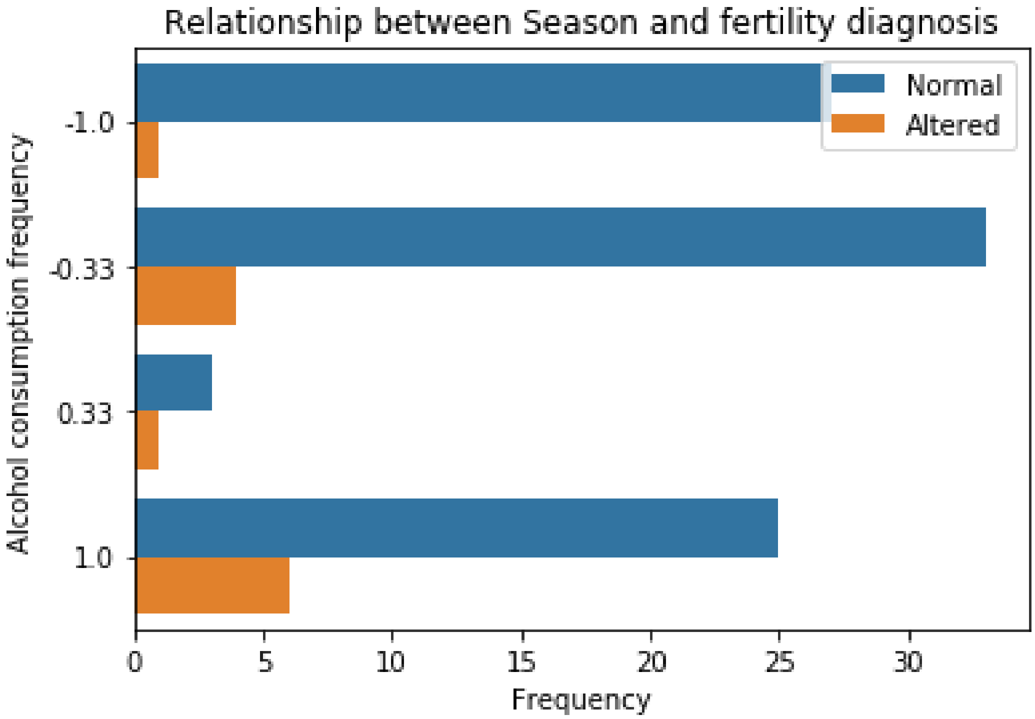
[34]:

```
ax9=sns.countplot(y='Alcohol_frequency', hue="Smoking_habit", data=fertility)
ax9.set_yticklabels(['Several times a day','Everyday','Several times a␣
 ↪week','Once a week','Hardly'])
plt.ylabel('Alcohol consumption frequency')
plt.xlabel('Frequency')
plt.legend(('Never', 'Occasionally', 'Daily'),loc='upper right')
plt.title('Relationship between Smoking habit and Smoking habit')
plt.show()
```



[35]:

```
ax10=sns.countplot(y='Season', hue="Output", data=fertility)
ax9.set_yticklabels(['Several times a day','Everyday','Several times a␣
 ↪week','Once a week'])
plt.ylabel('Alcohol consumption frequency')
plt.xlabel('Frequency')
plt.legend(('Normal', 'Altered'),loc='upper right')
plt.title('Relationship between Season and fertility diagnosis')
plt.show()
```



[36]:
```
from sklearn.model_selection import train_test_split
```

[37]:
```
sperm
```

[37]:
```
<addinfourl at 103055496L whose fp = <socket._fileobject object at
0x000000000621B6D8>>
```

[38]:
```
fertility['Output'].replace('N', '0', inplace=True)
```

[39]:
```
fertility['Output'].replace('O', '1', inplace=True)
```

[40]:
```
fertility['Output'] = fertility['Output'].astype('float64')
```

[41]:
```
fertility.dtypes
```

[41]:
```
Season                   float64
Age                      float64
Childish_diseases          int64
Accident                   int64
Surgical_intervention      int64
High_fever_last_year       int64
Alcohol_frequency        float64
Smoking_habit              int64
```

```
Hours_sitting            float64
Output                   float64
dtype: object
```

[42]:

```
for col in fertility.columns:
    if (fertility[col].dtype.name == 'int64'):
        fertility[col]=fertility[col].astype('float64')
```

[43]:

```
fertility.dtypes
```

[43]:

```
Season                   float64
Age                      float64
Childish_diseases        float64
Accident                 float64
Surgical_intervention    float64
High_fever_last_year     float64
Alcohol_frequency        float64
Smoking_habit            float64
Hours_sitting            float64
Output                   float64
dtype: object
```

[44]:

```
import numpy as np
```

[45]:

```
fertility.to_csv('fertility_ready.csv')
```

[46]:

```
fertility_data= 'fertility_ready.csv'
```

[47]:

```
dataset = np.loadtxt(fertility_data, delimiter=",", skiprows=1)
```

[48]:

```
print(dataset.shape)
```

```
(100L, 11L)
```

[49]:

```
dataset.shape
```

[49]:

```
(100L, 11L)
```

[50]:

```
X= dataset[:,1:10]
```

[51]:

```
Y= dataset[:,10]
```

[ ]:

```

```

[52]:

```
from sklearn.model_selection import train_test_split
```

[53]:

```
X
```

```
[53]: array([[-0.33,  0.69,  0.  ,  1.  ,  1.  ,  0.  ,  0.8 ,  0.  ,  0.88],
             [-0.33,  0.94,  1.  ,  0.  ,  1.  ,  0.  ,  0.8 ,  1.  ,  0.31],
             [-0.33,  0.5 ,  1.  ,  0.  ,  0.  ,  0.  ,  1.  , -1.  ,  0.5 ],
             [-0.33,  0.75,  0.  ,  1.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.38],
             [-0.33,  0.67,  1.  ,  1.  ,  0.  ,  0.  ,  0.8 , -1.  ,  0.5 ],
             [-0.33,  0.67,  1.  ,  0.  ,  1.  ,  0.  ,  0.8 ,  0.  ,  0.5 ],
             [-0.33,  0.67,  0.  ,  0.  ,  0.  , -1.  ,  0.8 , -1.  ,  0.44],
             [-0.33,  1.  ,  1.  ,  1.  ,  1.  ,  0.  ,  0.6 , -1.  ,  0.38],
             [ 1.  ,  0.64,  0.  ,  0.  ,  1.  ,  0.  ,  0.8 , -1.  ,  0.25],
             [ 1.  ,  0.61,  1.  ,  0.  ,  0.  ,  0.  ,  1.  , -1.  ,  0.25],
             [ 1.  ,  0.67,  1.  ,  1.  ,  0.  , -1.  ,  0.8 ,  0.  ,  0.31],
             [ 1.  ,  0.78,  1.  ,  1.  ,  1.  ,  0.  ,  0.6 ,  0.  ,  0.13],
             [ 1.  ,  0.75,  1.  ,  1.  ,  1.  ,  0.  ,  0.8 ,  1.  ,  0.25],
             [ 1.  ,  0.81,  1.  ,  0.  ,  0.  ,  0.  ,  1.  , -1.  ,  0.38],
             [ 1.  ,  0.94,  1.  ,  1.  ,  1.  ,  0.  ,  0.2 , -1.  ,  0.25],
             [ 1.  ,  0.81,  1.  ,  1.  ,  0.  ,  0.  ,  1.  ,  1.  ,  0.5 ],
             [ 1.  ,  0.64,  1.  ,  0.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.38],
             [ 1.  ,  0.69,  1.  ,  0.  ,  1.  ,  0.  ,  0.8 , -1.  ,  0.25],
             [ 1.  ,  0.75,  1.  ,  1.  ,  1.  ,  0.  ,  1.  ,  1.  ,  0.25],
             [ 1.  ,  0.67,  1.  ,  0.  ,  0.  ,  0.  ,  0.8 ,  1.  ,  0.38],
             [ 1.  ,  0.67,  0.  ,  0.  ,  1.  ,  0.  ,  0.8 , -1.  ,  0.25],
             [ 1.  ,  0.75,  1.  ,  0.  ,  0.  ,  0.  ,  0.6 ,  0.  ,  0.25],
             [ 1.  ,  0.67,  1.  ,  1.  ,  0.  ,  0.  ,  0.8 , -1.  ,  0.25],
             [ 1.  ,  0.69,  1.  ,  0.  ,  1.  , -1.  ,  1.  , -1.  ,  0.44],
             [ 1.  ,  0.56,  1.  ,  0.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.63],
             [ 1.  ,  0.67,  1.  ,  0.  ,  0.  ,  0.  ,  1.  , -1.  ,  0.25],
             [ 1.  ,  0.67,  1.  ,  0.  ,  1.  ,  0.  ,  0.6 , -1.  ,  0.38],
             [ 1.  ,  0.78,  1.  ,  1.  ,  0.  ,  1.  ,  0.6 , -1.  ,  0.38],
             [ 1.  ,  0.58,  0.  ,  0.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.19],
             [ 1.  ,  0.67,  0.  ,  0.  ,  1.  ,  0.  ,  0.6 ,  0.  ,  0.5 ],
             [ 1.  ,  0.61,  1.  ,  0.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.63],
             [ 1.  ,  0.56,  1.  ,  0.  ,  0.  ,  0.  ,  1.  , -1.  ,  0.44],
             [ 1.  ,  0.64,  0.  ,  0.  ,  0.  ,  0.  ,  1.  , -1.  ,  0.63],
             [ 1.  ,  0.58,  1.  ,  1.  ,  1.  ,  0.  ,  0.8 ,  0.  ,  0.44],
             [ 1.  ,  0.56,  1.  ,  1.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.63],
             [-1.  ,  0.78,  1.  ,  1.  ,  0.  ,  1.  ,  0.6 , -1.  ,  0.38],
             [-1.  ,  0.78,  1.  ,  0.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.25],
             [-1.  ,  0.56,  1.  ,  0.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.63],
             [-1.  ,  0.67,  0.  ,  0.  ,  1.  ,  0.  ,  0.6 ,  0.  ,  0.5 ],
             [-1.  ,  0.69,  1.  ,  0.  ,  0.  ,  0.  ,  1.  , -1.  ,  0.31],
             [-1.  ,  0.53,  1.  ,  1.  ,  1.  ,  0.  ,  0.8 ,  1.  ,  0.5 ],
             [-1.  ,  0.56,  1.  ,  1.  ,  0.  ,  0.  ,  0.8 ,  1.  ,  0.5 ],
             [-1.  ,  0.58,  1.  ,  0.  ,  1.  , -1.  ,  0.8 ,  1.  ,  0.5 ],
             [-1.  ,  0.56,  1.  ,  0.  ,  0.  ,  0.  ,  1.  , -1.  ,  0.44],
             [-1.  ,  0.53,  1.  ,  1.  ,  0.  ,  1.  ,  1.  ,  0.  ,  0.31],
             [-1.  ,  0.53,  1.  ,  0.  ,  0.  ,  1.  ,  1.  ,  0.  ,  0.44],
             [-0.33,  0.56,  1.  ,  0.  ,  0.  ,  0.  ,  1.  , -1.  ,  0.63],
             [-0.33,  0.72,  1.  ,  1.  ,  0.  ,  0.  ,  0.6 ,  1.  ,  0.19],
             [-0.33,  0.64,  1.  ,  1.  ,  1.  ,  0.  ,  0.8 , -1.  ,  0.31],
             [-0.33,  0.75,  1.  ,  1.  ,  1.  ,  0.  ,  0.6 , -1.  ,  0.19],
             [-0.33,  0.67,  1.  ,  0.  ,  1.  ,  0.  ,  0.8 , -1.  ,  0.19],
             [-0.33,  0.53,  1.  ,  1.  ,  0.  ,  1.  ,  1.  , -1.  ,  0.75],
             [-0.33,  0.53,  1.  ,  1.  ,  0.  ,  0.  ,  0.8 ,  0.  ,  0.5 ],
             [-0.33,  0.58,  1.  ,  1.  ,  1.  , -1.  ,  0.8 ,  0.  ,  0.19],
             [-0.33,  0.61,  1.  ,  0.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.63],
             [-0.33,  0.58,  1.  ,  0.  ,  1.  ,  0.  ,  0.8 ,  1.  ,  0.19],
             [-0.33,  0.53,  1.  ,  1.  ,  0.  ,  0.  ,  0.8 ,  0.  ,  0.75],
             [-0.33,  0.69,  1.  ,  1.  ,  1.  , -1.  ,  1.  , -1.  ,  0.75],
             [-0.33,  0.56,  1.  ,  1.  ,  0.  ,  0.  ,  0.4 ,  1.  ,  0.63],
             [ 1.  ,  0.58,  0.  ,  0.  ,  0.  ,  1.  ,  0.8 ,  1.  ,  0.44],
             [ 1.  ,  0.56,  0.  ,  0.  ,  0.  ,  1.  ,  0.8 ,  0.  ,  1.  ],
             [-1.  ,  0.64,  1.  ,  0.  ,  0.  ,  1.  ,  1.  ,  1.  ,  0.25],
             [-1.  ,  0.61,  1.  ,  1.  ,  1.  ,  0.  ,  0.6 , -1.  ,  0.38],
             [-1.  ,  0.56,  1.  ,  0.  ,  0.  ,  1.  ,  1.  , -1.  ,  0.5 ],
             [-1.  ,  0.53,  1.  ,  0.  ,  0.  ,  1.  ,  0.8 , -1.  ,  0.31],
             [-0.33,  0.56,  0.  ,  0.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.56],
             [-0.33,  0.5 ,  1.  ,  1.  ,  0.  , -1.  ,  0.8 ,  0.  ,  0.88],
             [-0.33,  0.5 ,  1.  ,  0.  ,  0.  ,  1.  ,  1.  , -1.  ,  0.47],
             [-0.33,  0.5 ,  1.  ,  0.  ,  0.  ,  1.  ,  0.8 ,  0.  ,  0.31],
             [-0.33,  0.5 ,  1.  ,  0.  ,  1.  , -1.  ,  0.8 , -1.  ,  0.5 ],
             [-0.33,  0.5 ,  1.  ,  1.  ,  0.  , -1.  ,  0.8 ,  0.  ,  0.88],
             [ 0.33,  0.69,  1.  ,  0.  ,  0.  ,  1.  ,  1.  , -1.  ,  0.31],
             [ 1.  ,  0.56,  1.  ,  0.  ,  0.  ,  1.  ,  0.6 ,  0.  ,  0.5 ],
             [-1.  ,  0.5 ,  1.  ,  0.  ,  0.  ,  1.  ,  0.8 , -1.  ,  0.44],
             [-1.  ,  0.53,  1.  ,  0.  ,  0.  ,  1.  ,  0.8 , -1.  ,  0.63],
             [-1.  ,  0.78,  1.  ,  0.  ,  1.  ,  1.  ,  1.  ,  1.  ,  0.25],
             [-1.  ,  0.75,  1.  ,  0.  ,  1.  ,  1.  ,  0.6 ,  0.  ,  0.56],
             [-1.  ,  0.72,  1.  ,  1.  ,  1.  ,  1.  ,  0.8 , -1.  ,  0.19],
             [-1.  ,  0.53,  1.  ,  1.  ,  0.  ,  1.  ,  0.8 , -1.  ,  0.38],
             [-1.  ,  1.  ,  1.  ,  0.  ,  1.  ,  1.  ,  0.6 ,  0.  ,  0.25],
             [-0.33,  0.92,  1.  ,  1.  ,  0.  ,  1.  ,  1.  , -1.  ,  0.63],
             [-1.  ,  0.81,  1.  ,  1.  ,  1.  ,  1.  ,  0.8 ,  0.  ,  0.19],
             [-0.33,  0.92,  1.  ,  0.  ,  0.  ,  1.  ,  0.6 , -1.  ,  0.19],
             [-0.33,  0.86,  1.  ,  1.  ,  1.  ,  1.  ,  1.  , -1.  ,  0.25],
             [-0.33,  0.78,  1.  ,  0.  ,  0.  ,  1.  ,  1.  ,  1.  ,  0.06],
             [-0.33,  0.89,  1.  ,  1.  ,  0.  ,  0.  ,  0.6 ,  1.  ,  0.31],
             [-0.33,  0.75,  1.  ,  1.  ,  1.  ,  0.  ,  0.6 ,  1.  ,  0.25],
             [-0.33,  0.75,  1.  ,  1.  ,  1.  ,  1.  ,  0.8 ,  1.  ,  0.25],
             [-0.33,  0.83,  1.  ,  1.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.31],
             [-0.33,  0.81,  1.  ,  1.  ,  1.  ,  0.  ,  1.  ,  1.  ,  0.38],
             [-0.33,  0.81,  1.  ,  1.  ,  1.  ,  1.  ,  0.8 , -1.  ,  0.38],
             [ 0.33,  0.78,  1.  ,  0.  ,  0.  ,  0.  ,  1.  ,  1.  ,  0.06],
             [ 0.33,  0.75,  1.  ,  1.  ,  0.  ,  0.  ,  0.8 , -1.  ,  0.38],
             [ 0.33,  0.75,  1.  ,  0.  ,  1.  ,  0.  ,  0.8 , -1.  ,  0.44],
             [ 1.  ,  0.58,  1.  ,  0.  ,  0.  ,  0.  ,  0.6 ,  1.  ,  0.5 ],
             [-1.  ,  0.67,  1.  ,  0.  ,  0.  ,  0.  ,  1.  , -1.  ,  0.5 ],
             [-1.  ,  0.61,  1.  ,  0.  ,  0.  ,  0.  ,  0.8 ,  0.  ,  0.5 ],
             [-1.  ,  0.67,  1.  ,  1.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.31],
             [-1.  ,  0.64,  1.  ,  0.  ,  1.  ,  0.  ,  1.  ,  0.  ,  0.19],
             [-1.  ,  0.69,  0.  ,  1.  ,  1.  ,  0.  ,  0.6 , -1.  ,  0.19]])
```

```
[54]: Y
```

```
[54]: array([0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
              1., 0., 1., 0., 0., 0., 1., 0., 0., 1., 1., 0., 1., 0., 0., 0., 0.,
              0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
              0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
              0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
              0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0.])
```

```
[55]: X_train,X_test,Y_train,Y_test= train_test_split(X,Y,test_size=0.
       ↪5,random_state=4)
```

```
[ ]:
```

```
[56]: X_train.shape
```

```
[56]: (50L, 9L)
```

```
[57]: from sklearn.neighbors import KNeighborsClassifier
```

```
[58]: clf = KNeighborsClassifier(5,weights='distance',p=1)
```

```
[59]: fit = clf.fit(X_train, Y_train)
```

```
[60]: y_pre = fit.predict(X_test)
```

```
[61]:  from sklearn.metrics import confusion_matrix
```

```
[62]:  cm = confusion_matrix(Y_test, y_pre)
```

```
[63]: print cm
```

```
[[43  0]
 [ 6  1]]
```

```
[64]: # Classification Error Rate
      a=50
      b=6.0
      print b / a
```

```
0.12
```

[65]:
```
from sklearn.metrics import classification_report
```

[66]:
```
print classification_report(Y_test,y_pre)
```

```
             precision    recall  f1-score   support

        0.0       0.88      1.00      0.93        43
        1.0       1.00      0.14      0.25         7

avg / total       0.89      0.88      0.84        50
```

[67]:
```
print "[Train/test split] score: {:.5f}".format(clf.score(X_test, Y_test))
```

```
[Train/test split] score: 0.88000
```

[68]:
```
#try to use the defaut setting for weights
clf = KNeighborsClassifier(5,p=1)
```

[69]:
```
fit = clf.fit(X_train, Y_train)
```

[70]:
```
clf.fit(X_train, Y_train)
```

[70]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=1,
           weights='uniform')
```

[71]:
```
y_pre = fit.predict(X_test)
```

[72]:
```
predicted = clf.predict(X_test)
```

[73]:
```
y_pre
```

[73]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

[74]:
```
 cm = confusion_matrix(Y_test, y_pre)
```

[75]:
```
print cm
```

```
[[43  0]
 [ 7  0]]
```

[76]:
```
print classification_report(Y_test,y_pre)
```

```
             precision    recall  f1-score   support

        0.0       0.86      1.00      0.92        43
        1.0       0.00      0.00      0.00         7

avg / total       0.74      0.86      0.80        50

C:\Users\s3335814\AppData\Local\Continuum\anaconda2\lib\site-
packages\sklearn\metrics\classification.py:1135: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples.
  'precision', 'predicted', average, warn_for)
```

[77]:
```
print "[Train/test split] score: {:.5f}".format(clf.score(X_test, Y_test))
```

```
[Train/test split] score: 0.86000
```

[78]:
```
#p=2
clf = KNeighborsClassifier(5,weights='distance',p=2)
```

[79]:
```
fit = clf.fit(X_train, Y_train)
```

[80]:
```
clf.fit(X_train, Y_train)
```

[80]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='distance')
```

[81]:
```
y_pre = fit.predict(X_test)
```

[82]:
```
predicted = clf.predict(X_test)
```

[83]:
```
y_pre
```

[83]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

[84]:
```
cm = confusion_matrix(Y_test, y_pre)
```

[85]:
```
print cm
```

```
[[43  0]
 [ 6  1]]
```

[86]:
```
print classification_report(Y_test,y_pre)
```

```
             precision    recall  f1-score   support

        0.0       0.88      1.00      0.93        43
        1.0       1.00      0.14      0.25         7

avg / total       0.89      0.88      0.84        50
```

[87]:
```
print "[Train/test split] score: {:.5f}".format(clf.score(X_test, Y_test))
```

```
[Train/test split] score: 0.88000
```

[88]:
```
#try smaller k value
```

[89]:
```
clf = KNeighborsClassifier(4,weights='distance',p=2)
```

[90]:
```
fit = clf.fit(X_train, Y_train)
```

[91]:
```
clf.fit(X_train, Y_train)
```

[91]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=4, p=2,
           weights='distance')
```

[92]:
```
y_pre = fit.predict(X_test)
```

[93]:
```
predicted = clf.predict(X_test)
```

[94]:
```
y_pre
```

[94]:
```
array([0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.])
```

[95]:
```
cm4 = confusion_matrix(Y_test, y_pre)
```

[96]:
```
print cm
```

```
[[43  0]
 [ 6  1]]
```

[97]:
```
print classification_report(Y_test,y_pre)
```

```
             precision    recall  f1-score   support

        0.0       0.89      0.98      0.93        43
        1.0       0.67      0.29      0.40         7
```

```
avg / total       0.86      0.88      0.86        50
```

```
[98]: print "[Train/test split] score: {:.2f}".format(clf.score(X_test, Y_test))
```

```
[Train/test split] score: 0.88
```

```
[99]: #k=3
```

```
[100]: clf = KNeighborsClassifier(3,weights='distance',p=2)
```

```
[101]: fit = clf.fit(X_train, Y_train)
```

```
[102]: clf.fit(X_train, Y_train)
```

```
[102]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                   metric_params=None, n_jobs=1, n_neighbors=3, p=2,
                   weights='distance')
```

```
[103]: y_pre = fit.predict(X_test)
```

```
[104]: predicted = clf.predict(X_test)
```

```
[105]: y_pre
```

```
[105]: array([0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.,
              0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
              0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.])
```

```
[106]: cm = confusion_matrix(Y_test, y_pre)
```

```
[107]: print cm
```

```
[[42  1]
 [ 5  2]]
```

```
[108]: print classification_report(Y_test,y_pre)
```

```
             precision    recall  f1-score   support

        0.0       0.89      0.98      0.93        43
        1.0       0.67      0.29      0.40         7

avg / total       0.86      0.88      0.86        50
```

```
[109]: print "[Train/test split] score: {:.2f}".format(clf.score(X_test, Y_test))
```

```
[Train/test split] score: 0.88
```

```
[110]: clf = KNeighborsClassifier(7,weights='distance',p=2)
```

```
[111]: fit = clf.fit(X_train, Y_train)
```

```
[112]: clf.fit(X_train, Y_train)
```

```
[112]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                  metric_params=None, n_jobs=1, n_neighbors=7, p=2,
                  weights='distance')
```

```
[113]: y_pre = fit.predict(X_test)
```

```
[114]: predicted = clf.predict(X_test)
```

```
[115]: cm = confusion_matrix(Y_test, y_pre)
```

```
[116]: print cm
```

```
[[43  0]
 [ 7  0]]
```

```
[117]: print classification_report(Y_test,y_pre)
```

```
             precision    recall  f1-score   support

        0.0       0.86      1.00      0.92        43
        1.0       0.00      0.00      0.00         7

avg / total       0.74      0.86      0.80        50
```

```
[118]: print "[Train/test split] score: {:.2f}".format(clf.score(X_test, Y_test))
```

```
[Train/test split] score: 0.86
```

```
[119]: clf = KNeighborsClassifier(2,weights='distance',p=2)
```

```
[120]: fit = clf.fit(X_train, Y_train)
```

```
[121]: clf.fit(X_train, Y_train)
```

```
[121]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                  metric_params=None, n_jobs=1, n_neighbors=2, p=2,
                  weights='distance')
```

```
[122]: y_pre = fit.predict(X_test)
```

[123]:

```
predicted = clf.predict(X_test)
```

[124]:

```
cm = confusion_matrix(Y_test, y_pre)
```

[125]:

```
print cm
```

```
[[41  2]
 [ 6  1]]
```

[126]:

```
 from sklearn.utils import shuffle # Hill climbing for KNN
```

[127]:

```
 new_Ind = []
```

[128]:

```
cur_MaxScore = 0.0
```

[129]:

```
col_num = 9
```

[130]:

```
col_Ind_Random = shuffle(range(0,col_num), random_state=12)
```

[131]:

```
for cur_f in range(0, col_num):
    new_Ind.append(col_Ind_Random[cur_f])
    newData = X[:, new_Ind]
    X_train, X_test, Y_train, Y_test = train_test_split(newData, Y, test_size=0.
 ↪50, random_state=4)
    clf = clf = KNeighborsClassifier(3,weights='distance',p=2)
    fit = clf.fit(X_train, Y_train)
    cur_Score = clf.score(X_test, Y_test)
    if cur_Score < cur_MaxScore:
                    new_Ind.remove(col_Ind_Random[cur_f])
    else:
        cur_MaxScore = cur_Score
        print "Score with " + str(len(new_Ind)) + " selected features: " +␣
 ↪str(cur_Score)
```

```
Score with 1 selected features: 0.86
Score with 2 selected features: 0.86
Score with 3 selected features: 0.88
Score with 4 selected features: 0.88
Score with 5 selected features: 0.88
Score with 6 selected features: 0.9
```

[132]:

```
print new_Ind
```

```
[8, 4, 3, 2, 1, 6]
```

[133]:

```
from sklearn.tree import DecisionTreeClassifier
```

```
[134]:  tree = DecisionTreeClassifier()
```

```
[135]: fit_tree =tree.fit(X_train, Y_train)
```

```
[136]: y_pre_tree = fit_tree.predict(X_test)
```

```
[137]: y_pre_tree
```

```
[137]: array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 0., 0.,
               1., 0., 0., 0., 0., 0., 1., 0., 0., 1., 0., 1., 1., 0., 0., 0., 0.,
               0., 0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0.])
```

```
[138]:  cm_tree = confusion_matrix(Y_test, y_pre_tree)
```

```
[139]: print cm_tree
```

```
[[36  7]
 [ 3  4]]
```

```
[140]:  print classification_report(Y_test,y_pre_tree)
```

```
             precision    recall  f1-score   support

        0.0       0.92      0.84      0.88        43
        1.0       0.36      0.57      0.44         7

avg / total       0.84      0.80      0.82        50
```

```
[141]:  #Fine tuning parameters
       cfl_tree = DecisionTreeClassifier(max_depth=4)
```

```
[142]: cfl_tree
```

```
[142]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=4,
                    max_features=None, max_leaf_nodes=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=1, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                    splitter='best')
```

```
[143]: fit=cfl_tree.fit(X_train, Y_train)
```

```
[144]: y_pre_tree = fit.predict(X_test)
```

```
[145]:  cm_tree = confusion_matrix(Y_test, y_pre_tree)
```

[146]:
```
print cm_tree
```

```
[[36  7]
 [ 3  4]]
```

[147]:
```
 print classification_report(Y_test,y_pre_tree)
```

```
             precision    recall  f1-score   support

        0.0       0.92      0.84      0.88        43
        1.0       0.36      0.57      0.44         7

avg / total       0.84      0.80      0.82        50
```

[148]:
```
from sklearn import tree
```

[149]:
```
 with open('fertility_tree.dot', 'w') as f:
    f = tree.export_graphviz(cfl_tree, out_file=f,filled=True, rounded=True,␣
 ↪special_characters=True)
```

[150]:
```
# Hill climbing for decision tree
```

[151]:
```
 new_Ind = []
```

[152]:
```
cur_MaxScore = 0.0
```

[153]:
```
col_num = 9
```

[154]:
```
col_Ind_Random = shuffle(range(0,col_num), random_state=12)
```

[155]:
```
for cur_f in range(0, col_num):
    new_Ind.append(col_Ind_Random[cur_f])
    newData = X[:, new_Ind]
    X_train, X_test, Y_train, Y_test = train_test_split(newData, Y, test_size=0.
 ↪50, random_state=4)
    clf_tree = DecisionTreeClassifier(max_depth=4)
    fit = clf_tree.fit(X_train, Y_train)
    cur_Score = clf_tree.score(X_test, Y_test)
    if cur_Score < cur_MaxScore:
                    new_Ind.remove(col_Ind_Random[cur_f])
    else:
        cur_MaxScore = cur_Score
        print "Score with " + str(len(new_Ind)) + " selected features: " +␣
 ↪str(cur_Score)
```

```
Score with 1 selected features: 0.86
Score with 2 selected features: 0.86
Score with 3 selected features: 0.86
Score with 4 selected features: 0.88
Score with 5 selected features: 0.88
Score with 6 selected features: 0.88
```

```
[156]: print new_Ind
```

```
[8, 7, 4, 3, 2, 6]
```

```
[157]:  from sklearn.ensemble import RandomForestClassifier #Random forrest
```

```
[158]: from sklearn.datasets import make_classification
```

```
[159]: cfl_forrest = RandomForestClassifier()
```

```
[160]: cfl_forrest
```

```
[160]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                   max_depth=None, max_features='auto', max_leaf_nodes=None,
                   min_impurity_decrease=0.0, min_impurity_split=None,
                   min_samples_leaf=1, min_samples_split=2,
                   min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
                   oob_score=False, random_state=None, verbose=0,
                   warm_start=False)
```

```
[161]: fit=cfl_forrest.fit(X_train, Y_train)
```

```
[162]: y_pre_forrest = fit.predict(X_test)
```

```
[163]:  cm_forrest = confusion_matrix(Y_test, y_pre_forrest)
```

```
[164]: print cm_forrest
```

```
[[43  0]
 [ 7  0]]
```

```
[165]:  print classification_report(Y_test,y_pre_forrest)
```

```
             precision    recall  f1-score   support

        0.0       0.86      1.00      0.92        43
        1.0       0.00      0.00      0.00         7

avg / total       0.74      0.86      0.80        50
```

```
[166]: #fine tuning
```

```
[167]: cfl_forrest = RandomForestClassifier(max_features=5)
```

```
[168]: fit=cfl_forrest.fit(X_train, Y_train)
```

```
[169]: y_pre_forrest = fit.predict(X_test)
```

```
[170]:  cm_forrest = confusion_matrix(Y_test, y_pre_forrest)
```

```
[171]: print cm_forrest
```

```
[[43  0]
 [ 7  0]]
```

```
[172]:  print classification_report(Y_test,y_pre_forrest)
```

```
             precision    recall  f1-score   support

        0.0       0.86      1.00      0.92        43
        1.0       0.00      0.00      0.00         7

avg / total       0.74      0.86      0.80        50
```

```
[173]:  new_Ind = []
```

```
[174]: cur_MaxScore = 0.0
```

```
[175]: col_num = 9
```

```
[176]: col_Ind_Random = shuffle(range(0,col_num), random_state=12)
```

```
[177]: for cur_f in range(0, col_num):
            new_Ind.append(col_Ind_Random[cur_f])
            newData = X[:, new_Ind]
            X_train, X_test, Y_train, Y_test = train_test_split(newData, Y, test_size=0.
         ↪50, random_state=4)
            clf_forrest = RandomForestClassifier()
            fit = clf_forrest.fit(X_train, Y_train)
            cur_Score = clf_forrest.score(X_test, Y_test)
            if cur_Score < cur_MaxScore:
                            new_Ind.remove(col_Ind_Random[cur_f])
            else:
                cur_MaxScore = cur_Score
                print "Score with " + str(len(new_Ind)) + " selected features: " +␣
         ↪str(cur_Score)
```

```
Score with 1 selected features: 0.86
Score with 2 selected features: 0.86
Score with 3 selected features: 0.86
Score with 4 selected features: 0.88
```

[178]:

```
print new_Ind
```

```
[8, 3, 2, 6]
```

[179]:

```
X
```

[179]:

```
array([[-0.33,  0.69,  0.  ,  1.  ,  1.  ,  0.  ,  0.8 ,  0.  ,  0.88],
       [-0.33,  0.94,  1.  ,  0.  ,  1.  ,  0.  ,  0.8 ,  1.  ,  0.31],
       [-0.33,  0.5 ,  1.  ,  0.  ,  0.  ,  0.  ,  1.  , -1.  ,  0.5 ],
       [-0.33,  0.75,  0.  ,  1.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.38],
       [-0.33,  0.67,  1.  ,  1.  ,  0.  ,  0.  ,  0.8 , -1.  ,  0.5 ],
       [-0.33,  0.67,  1.  ,  0.  ,  1.  ,  0.  ,  0.8 ,  0.  ,  0.5 ],
       [-0.33,  0.67,  0.  ,  0.  ,  0.  , -1.  ,  0.8 , -1.  ,  0.44],
       [-0.33,  1.  ,  1.  ,  1.  ,  1.  ,  0.  ,  0.6 , -1.  ,  0.38],
       [ 1.  ,  0.64,  0.  ,  0.  ,  1.  ,  0.  ,  0.8 , -1.  ,  0.25],
       [ 1.  ,  0.61,  1.  ,  0.  ,  0.  ,  0.  ,  1.  , -1.  ,  0.25],
       [ 1.  ,  0.67,  1.  ,  1.  ,  0.  , -1.  ,  0.8 ,  0.  ,  0.31],
       [ 1.  ,  0.78,  1.  ,  1.  ,  1.  ,  0.  ,  0.6 ,  0.  ,  0.13],
       [ 1.  ,  0.75,  1.  ,  1.  ,  1.  ,  0.  ,  0.8 ,  1.  ,  0.25],
       [ 1.  ,  0.81,  1.  ,  0.  ,  0.  ,  0.  ,  1.  , -1.  ,  0.38],
       [ 1.  ,  0.94,  1.  ,  1.  ,  1.  ,  0.  ,  0.2 , -1.  ,  0.25],
       [ 1.  ,  0.81,  1.  ,  1.  ,  0.  ,  0.  ,  1.  ,  1.  ,  0.5 ],
       [ 1.  ,  0.64,  1.  ,  0.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.38],
       [ 1.  ,  0.69,  1.  ,  0.  ,  1.  ,  0.  ,  0.8 , -1.  ,  0.25],
       [ 1.  ,  0.75,  1.  ,  1.  ,  1.  ,  0.  ,  1.  ,  1.  ,  0.25],
       [ 1.  ,  0.67,  1.  ,  0.  ,  0.  ,  0.  ,  0.8 ,  1.  ,  0.38],
       [ 1.  ,  0.67,  0.  ,  0.  ,  1.  ,  0.  ,  0.8 , -1.  ,  0.25],
       [ 1.  ,  0.75,  1.  ,  0.  ,  0.  ,  0.  ,  0.6 ,  0.  ,  0.25],
       [ 1.  ,  0.67,  1.  ,  1.  ,  0.  ,  0.  ,  0.8 , -1.  ,  0.25],
       [ 1.  ,  0.69,  1.  ,  0.  ,  1.  , -1.  ,  1.  , -1.  ,  0.44],
       [ 1.  ,  0.56,  1.  ,  0.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.63],
       [ 1.  ,  0.67,  1.  ,  0.  ,  0.  ,  0.  ,  1.  , -1.  ,  0.25],
       [ 1.  ,  0.67,  1.  ,  0.  ,  1.  ,  0.  ,  0.6 , -1.  ,  0.38],
       [ 1.  ,  0.78,  1.  ,  1.  ,  0.  ,  1.  ,  0.6 , -1.  ,  0.38],
       [ 1.  ,  0.58,  0.  ,  0.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.19],
       [ 1.  ,  0.67,  0.  ,  0.  ,  1.  ,  0.  ,  0.6 ,  0.  ,  0.5 ],
       [ 1.  ,  0.61,  1.  ,  0.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.63],
       [ 1.  ,  0.56,  1.  ,  0.  ,  0.  ,  0.  ,  1.  , -1.  ,  0.44],
       [ 1.  ,  0.64,  0.  ,  0.  ,  0.  ,  0.  ,  1.  , -1.  ,  0.63],
       [ 1.  ,  0.58,  1.  ,  1.  ,  1.  ,  0.  ,  0.8 ,  0.  ,  0.44],
       [ 1.  ,  0.56,  1.  ,  1.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.63],
       [-1.  ,  0.78,  1.  ,  1.  ,  0.  ,  1.  ,  0.6 , -1.  ,  0.38],
       [-1.  ,  0.78,  1.  ,  0.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.25],
       [-1.  ,  0.56,  1.  ,  0.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.63],
       [-1.  ,  0.67,  0.  ,  0.  ,  1.  ,  0.  ,  0.6 ,  0.  ,  0.5 ],
       [-1.  ,  0.69,  1.  ,  0.  ,  0.  ,  0.  ,  1.  , -1.  ,  0.31],
       [-1.  ,  0.53,  1.  ,  1.  ,  1.  ,  0.  ,  0.8 ,  1.  ,  0.5 ],
       [-1.  ,  0.56,  1.  ,  1.  ,  0.  ,  0.  ,  0.8 ,  1.  ,  0.5 ],
       [-1.  ,  0.58,  1.  ,  0.  ,  1.  , -1.  ,  0.8 ,  1.  ,  0.5 ],
       [-1.  ,  0.56,  1.  ,  0.  ,  0.  ,  0.  ,  1.  , -1.  ,  0.44],
       [-1.  ,  0.53,  1.  ,  1.  ,  0.  ,  1.  ,  1.  ,  0.  ,  0.31],
       [-1.  ,  0.53,  1.  ,  0.  ,  0.  ,  1.  ,  1.  ,  0.  ,  0.44],
       [-0.33,  0.56,  1.  ,  0.  ,  0.  ,  0.  ,  1.  , -1.  ,  0.63],
       [-0.33,  0.72,  1.  ,  1.  ,  0.  ,  0.  ,  0.6 ,  1.  ,  0.19],
       [-0.33,  0.64,  1.  ,  1.  ,  1.  ,  0.  ,  0.8 , -1.  ,  0.31],
       [-0.33,  0.75,  1.  ,  1.  ,  1.  ,  0.  ,  0.6 , -1.  ,  0.19],
       [-0.33,  0.67,  1.  ,  0.  ,  1.  ,  0.  ,  0.8 , -1.  ,  0.19],
       [-0.33,  0.53,  1.  ,  1.  ,  0.  ,  1.  ,  1.  , -1.  ,  0.75],
       [-0.33,  0.53,  1.  ,  1.  ,  0.  ,  0.  ,  0.8 ,  0.  ,  0.5 ],
       [-0.33,  0.58,  1.  ,  1.  ,  1.  , -1.  ,  0.8 ,  0.  ,  0.19],
       [-0.33,  0.61,  1.  ,  0.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.63],
       [-0.33,  0.58,  1.  ,  0.  ,  1.  ,  0.  ,  0.8 ,  1.  ,  0.19],
       [-0.33,  0.53,  1.  ,  1.  ,  0.  ,  0.  ,  0.8 ,  0.  ,  0.75],
       [-0.33,  0.69,  1.  ,  1.  ,  1.  , -1.  ,  1.  , -1.  ,  0.75],
       [-0.33,  0.56,  1.  ,  1.  ,  0.  ,  0.  ,  0.4 ,  1.  ,  0.63],
       [ 1.  ,  0.58,  0.  ,  0.  ,  0.  ,  1.  ,  0.8 ,  1.  ,  0.44],
       [ 1.  ,  0.56,  0.  ,  0.  ,  0.  ,  1.  ,  0.8 ,  0.  ,  1.  ],
       [-1.  ,  0.64,  1.  ,  0.  ,  0.  ,  1.  ,  1.  ,  1.  ,  0.25],
       [-1.  ,  0.61,  1.  ,  1.  ,  1.  ,  0.  ,  0.6 , -1.  ,  0.38],
       [-1.  ,  0.56,  1.  ,  0.  ,  0.  ,  1.  ,  1.  , -1.  ,  0.5 ],
       [-1.  ,  0.53,  1.  ,  0.  ,  0.  ,  1.  ,  0.8 , -1.  ,  0.31],
       [-0.33,  0.56,  0.  ,  0.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.56],
       [-0.33,  0.5 ,  1.  ,  1.  ,  0.  , -1.  ,  0.8 ,  0.  ,  0.88],
       [-0.33,  0.5 ,  1.  ,  0.  ,  0.  ,  1.  ,  1.  , -1.  ,  0.47],
       [-0.33,  0.5 ,  1.  ,  0.  ,  0.  ,  1.  ,  0.8 ,  0.  ,  0.31],
       [-0.33,  0.5 ,  1.  ,  0.  ,  1.  , -1.  ,  0.8 , -1.  ,  0.5 ],
       [-0.33,  0.5 ,  1.  ,  1.  ,  0.  , -1.  ,  0.8 ,  0.  ,  0.88],
       [ 0.33,  0.69,  1.  ,  0.  ,  0.  ,  1.  ,  1.  , -1.  ,  0.31],
       [ 1.  ,  0.56,  1.  ,  0.  ,  0.  ,  1.  ,  0.6 ,  0.  ,  0.5 ],
       [-1.  ,  0.5 ,  1.  ,  0.  ,  0.  ,  1.  ,  0.8 , -1.  ,  0.44],
       [-1.  ,  0.53,  1.  ,  0.  ,  0.  ,  1.  ,  0.8 , -1.  ,  0.63],
       [-1.  ,  0.78,  1.  ,  0.  ,  1.  ,  1.  ,  1.  ,  1.  ,  0.25],
       [-1.  ,  0.75,  1.  ,  0.  ,  1.  ,  1.  ,  0.6 ,  0.  ,  0.56],
       [-1.  ,  0.72,  1.  ,  1.  ,  1.  ,  1.  ,  0.8 , -1.  ,  0.19],
       [-1.  ,  0.53,  1.  ,  1.  ,  0.  ,  1.  ,  0.8 , -1.  ,  0.38],
       [-1.  ,  1.  ,  1.  ,  0.  ,  1.  ,  1.  ,  0.6 ,  0.  ,  0.25],
       [-0.33,  0.92,  1.  ,  1.  ,  0.  ,  1.  ,  1.  , -1.  ,  0.63],
       [-1.  ,  0.81,  1.  ,  1.  ,  1.  ,  1.  ,  0.8 ,  0.  ,  0.19],
       [-0.33,  0.92,  1.  ,  0.  ,  0.  ,  1.  ,  0.6 , -1.  ,  0.19],
       [-0.33,  0.86,  1.  ,  1.  ,  1.  ,  1.  ,  1.  , -1.  ,  0.25],
              [-0.33,  0.78,  1.  ,  0.  ,  0.  ,  1.  ,  1.  ,  1.  ,  0.06],
              [-0.33,  0.89,  1.  ,  1.  ,  0.  ,  0.  ,  0.6 ,  1.  ,  0.31],
              [-0.33,  0.75,  1.  ,  1.  ,  1.  ,  0.  ,  0.6 ,  1.  ,  0.25],
              [-0.33,  0.75,  1.  ,  1.  ,  1.  ,  1.  ,  0.8 ,  1.  ,  0.25],
              [-0.33,  0.83,  1.  ,  1.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.31],
              [-0.33,  0.81,  1.  ,  1.  ,  1.  ,  0.  ,  1.  ,  1.  ,  0.38],
              [-0.33,  0.81,  1.  ,  1.  ,  1.  ,  1.  ,  0.8 , -1.  ,  0.38],
              [ 0.33,  0.78,  1.  ,  0.  ,  0.  ,  0.  ,  1.  ,  1.  ,  0.06],
              [ 0.33,  0.75,  1.  ,  1.  ,  0.  ,  0.  ,  0.8 , -1.  ,  0.38],
              [ 0.33,  0.75,  1.  ,  0.  ,  1.  ,  0.  ,  0.8 , -1.  ,  0.44],
              [ 1.  ,  0.58,  1.  ,  0.  ,  0.  ,  0.  ,  0.6 ,  1.  ,  0.5 ],
              [-1.  ,  0.67,  1.  ,  0.  ,  0.  ,  0.  ,  1.  , -1.  ,  0.5 ],
              [-1.  ,  0.61,  1.  ,  0.  ,  0.  ,  0.  ,  0.8 ,  0.  ,  0.5 ],
              [-1.  ,  0.67,  1.  ,  1.  ,  1.  ,  0.  ,  1.  , -1.  ,  0.31],
              [-1.  ,  0.64,  1.  ,  0.  ,  1.  ,  0.  ,  1.  ,  0.  ,  0.19],
              [-1.  ,  0.69,  0.  ,  1.  ,  1.  ,  0.  ,  0.6 , -1.  ,  0.19]])
```

[180]:
```
fertility
```

[180]:
```
    Season   Age  Childish_diseases  Accident  Surgical_intervention  \
0    -0.33  0.69                0.0       1.0                    1.0   
1    -0.33  0.94                1.0       0.0                    1.0   
2    -0.33  0.50                1.0       0.0                    0.0   
3    -0.33  0.75                0.0       1.0                    1.0   
4    -0.33  0.67                1.0       1.0                    0.0   
5    -0.33  0.67                1.0       0.0                    1.0   
6    -0.33  0.67                0.0       0.0                    0.0   
7    -0.33  1.00                1.0       1.0                    1.0   
8     1.00  0.64                0.0       0.0                    1.0   
9     1.00  0.61                1.0       0.0                    0.0   
10    1.00  0.67                1.0       1.0                    0.0   
11    1.00  0.78                1.0       1.0                    1.0   
12    1.00  0.75                1.0       1.0                    1.0   
13    1.00  0.81                1.0       0.0                    0.0   
14    1.00  0.94                1.0       1.0                    1.0   
15    1.00  0.81                1.0       1.0                    0.0   
16    1.00  0.64                1.0       0.0                    1.0   
17    1.00  0.69                1.0       0.0                    1.0   
18    1.00  0.75                1.0       1.0                    1.0   
19    1.00  0.67                1.0       0.0                    0.0   
20    1.00  0.67                0.0       0.0                    1.0   
21    1.00  0.75                1.0       0.0                    0.0   
22    1.00  0.67                1.0       1.0                    0.0   
23    1.00  0.69                1.0       0.0                    1.0   
24    1.00  0.56                1.0       0.0                    1.0   
25    1.00  0.67                1.0       0.0                    0.0   
26    1.00  0.67                1.0       0.0                    1.0   
```

```
27    1.00  0.78                1.0       1.0                    0.0
28    1.00  0.58                0.0       0.0                    1.0
29    1.00  0.67                0.0       0.0                    1.0
..     …   …                …       …                    …
70   -0.33  0.50                1.0       1.0                    0.0
71    0.33  0.69                1.0       0.0                    0.0
72    1.00  0.56                1.0       0.0                    0.0
73   -1.00  0.50                1.0       0.0                    0.0
74   -1.00  0.53                1.0       0.0                    0.0
75   -1.00  0.78                1.0       0.0                    1.0
76   -1.00  0.75                1.0       0.0                    1.0
77   -1.00  0.72                1.0       1.0                    1.0
78   -1.00  0.53                1.0       1.0                    0.0
79   -1.00  1.00                1.0       0.0                    1.0
80   -0.33  0.92                1.0       1.0                    0.0
81   -1.00  0.81                1.0       1.0                    1.0
82   -0.33  0.92                1.0       0.0                    0.0
83   -0.33  0.86                1.0       1.0                    1.0
84   -0.33  0.78                1.0       0.0                    0.0
85   -0.33  0.89                1.0       1.0                    0.0
86   -0.33  0.75                1.0       1.0                    1.0
87   -0.33  0.75                1.0       1.0                    1.0
88   -0.33  0.83                1.0       1.0                    1.0
89   -0.33  0.81                1.0       1.0                    1.0
90   -0.33  0.81                1.0       1.0                    1.0
91    0.33  0.78                1.0       0.0                    0.0
92    0.33  0.75                1.0       1.0                    0.0
93    0.33  0.75                1.0       0.0                    1.0
94    1.00  0.58                1.0       0.0                    0.0
95   -1.00  0.67                1.0       0.0                    0.0
96   -1.00  0.61                1.0       0.0                    0.0
97   -1.00  0.67                1.0       1.0                    1.0
98   -1.00  0.64                1.0       0.0                    1.0
99   -1.00  0.69                0.0       1.0                    1.0

    High_fever_last_year  Alcohol_frequency  Smoking_habit  Hours_sitting  \
0                    0.0                0.8            0.0           0.88
1                    0.0                0.8            1.0           0.31
2                    0.0                1.0           -1.0           0.50
3                    0.0                1.0           -1.0           0.38
4                    0.0                0.8           -1.0           0.50
5                    0.0                0.8            0.0           0.50
6                   -1.0                0.8           -1.0           0.44
7                    0.0                0.6           -1.0           0.38
8                    0.0                0.8           -1.0           0.25
9                    0.0                1.0           -1.0           0.25
10                  -1.0                0.8            0.0           0.31
```

11	0.0	0.6	0.0	0.13
12	0.0	0.8	1.0	0.25
13	0.0	1.0	-1.0	0.38
14	0.0	0.2	-1.0	0.25
15	0.0	1.0	1.0	0.50
16	0.0	1.0	-1.0	0.38
17	0.0	0.8	-1.0	0.25
18	0.0	1.0	1.0	0.25
19	0.0	0.8	1.0	0.38
20	0.0	0.8	-1.0	0.25
21	0.0	0.6	0.0	0.25
22	0.0	0.8	-1.0	0.25
23	-1.0	1.0	-1.0	0.44
24	0.0	1.0	-1.0	0.63
25	0.0	1.0	-1.0	0.25
26	0.0	0.6	-1.0	0.38
27	1.0	0.6	-1.0	0.38
28	0.0	1.0	-1.0	0.19
29	0.0	0.6	0.0	0.50
..	…	…	…	…
70	-1.0	0.8	0.0	0.88
71	1.0	1.0	-1.0	0.31
72	1.0	0.6	0.0	0.50
73	1.0	0.8	-1.0	0.44
74	1.0	0.8	-1.0	0.63
75	1.0	1.0	1.0	0.25
76	1.0	0.6	0.0	0.56
77	1.0	0.8	-1.0	0.19
78	1.0	0.8	-1.0	0.38
79	1.0	0.6	0.0	0.25
80	1.0	1.0	-1.0	0.63
81	1.0	0.8	0.0	0.19
82	1.0	0.6	-1.0	0.19
83	1.0	1.0	-1.0	0.25
84	1.0	1.0	1.0	0.06
85	0.0	0.6	1.0	0.31
86	0.0	0.6	1.0	0.25
87	1.0	0.8	1.0	0.25
88	0.0	1.0	-1.0	0.31
89	0.0	1.0	1.0	0.38
90	1.0	0.8	-1.0	0.38
91	0.0	1.0	1.0	0.06
92	0.0	0.8	-1.0	0.38
93	0.0	0.8	-1.0	0.44
94	0.0	0.6	1.0	0.50
95	0.0	1.0	-1.0	0.50
96	0.0	0.8	0.0	0.50

```
97                   0.0                1.0            -1.0           0.31
98                   0.0                1.0             0.0           0.19
99                   0.0                0.6            -1.0           0.19

    Output
0      0.0
1      1.0
2      0.0
3      0.0
4      1.0
5      0.0
6      0.0
7      0.0
8      0.0
9      0.0
10     0.0
11     0.0
12     0.0
13     0.0
14     0.0
15     0.0
16     0.0
17     1.0
18     0.0
19     1.0
20     0.0
21     0.0
22     0.0
23     1.0
24     0.0
25     0.0
26     1.0
27     1.0
28     0.0
29     1.0
..     …
70     1.0
71     0.0
72     0.0
73     0.0
74     0.0
75     0.0
76     0.0
77     0.0
78     0.0
79     0.0
80     0.0
```

```
81     0.0
82     0.0
83     0.0
84     1.0
85     0.0
86     0.0
87     0.0
88     0.0
89     0.0
90     0.0
91     0.0
92     0.0
93     1.0
94     0.0
95     0.0
96     0.0
97     0.0
98     0.0
99     0.0

[100 rows x 10 columns]
```

[181]:
```
from sklearn.model_selection import KFold #use Kfold to cross validate
```

[182]:
```
kf = KFold(n_splits=6,random_state=0)
```

[183]:
```
for train_index, test_index in kf.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    Y_train, Y_test = Y[train_index], Y[test_index]
```

```
('TRAIN:', array([17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
       51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
       85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99],
      dtype=int64), 'TEST:', array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10,
11, 12, 13, 14, 15, 16],
      dtype=int64))
('TRAIN:', array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14,
15, 16,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
       51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
       85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99],
      dtype=int64), 'TEST:', array([17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27,
28, 29, 30, 31, 32, 33],
      dtype=int64))
('TRAIN:', array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14,
15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
       85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99],
      dtype=int64), 'TEST:', array([34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44,
45, 46, 47, 48, 49, 50],
      dtype=int64))
('TRAIN:', array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14,
15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
       85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99],
      dtype=int64), 'TEST:', array([51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61,
62, 63, 64, 65, 66, 67],
      dtype=int64))
('TRAIN:', array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14,
15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
       51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
       84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99],
      dtype=int64), 'TEST:', array([68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78,
79, 80, 81, 82, 83],
      dtype=int64))
('TRAIN:', array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14,
15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
       51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83],
      dtype=int64), 'TEST:', array([84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94,
95, 96, 97, 98, 99],
      dtype=int64))
```

[184]:

```
for k, (train_index, test_index) in enumerate(kf.split(X)):
    X_train, X_test = X[train_index], X[test_index]
    Y_train, Y_test = Y[train_index], Y[test_index]

    clf.fit(X_train, Y_train)
    print '[fold {0}] score: {1:.4f}'.format(k, clf.score(X_test, Y_test))
```

```
[fold 0] score: 0.8235
[fold 1] score: 0.6471
```

```
[fold 2] score: 0.9412
[fold 3] score: 0.7647
[fold 4] score: 0.9375
[fold 5] score: 0.8125
```