In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from io import StringIO
from sklearn.preprocessing import LabelEncoder
import numpy as np
import seaborn as sns
```

In [2]:
```
audio = pd.read_csv('audiology2.csv',header=0,sep=',')
audio.columns
df=audio
```

In [3]:
```
df=df.drop("number_row",axis = 1)
var_mod = ['age_gt_60','airBoneGap','boneAbnormal','history_buzzing','history_
dizziness',
       'history_fluctuating', 'history_fullness', 'history_heredity',
       'history_nausea', 'history_noise', 'history_recruitment',
       'history_ringing', 'history_roaring', 'history_vomiting',
       'late_wave_poor', 'm_at_2k', 'm_cond_lt_1k', 'm_gt_1k', 'm_m_gt_2k',
       'm_m_sn', 'm_m_sn_gt_1k', 'm_m_sn_gt_2k', 'm_m_sn_gt_500',
       'm_p_sn_gt_2k', 'm_s_gt_500', 'm_s_sn', 'm_s_sn_gt_1k', 'm_s_sn_gt_2k',
       'm_s_sn_gt_3k', 'm_s_sn_gt_4k', 'm_sn_2_3k', 'm_sn_gt_1k', 'm_sn_gt_2k'
,
       'm_sn_gt_3k', 'm_sn_gt_4k', 'm_sn_gt_500', 'm_sn_gt_6k', 'm_sn_lt_1k',
       'm_sn_lt_2k', 'm_sn_lt_3k', 'middle_wave_poor', 'mod_gt_4k',
       'mod_mixed', 'mod_s_mixed', 'mod_s_sn_gt_500', 'mod_sn', 'mod_sn_gt_1k'
,
       'mod_sn_gt_2k', 'mod_sn_gt_3k', 'mod_sn_gt_4k', 'mod_sn_gt_500',
       'notch_4k', 'notch_at_4k', 's_sn_gt_1k',
       's_sn_gt_2k', 's_sn_gt_4k', 'static_normal','viith_nerve_signs',
      'wave_V_delayed', 'waveform_ItoV_prolonged']

le = LabelEncoder()

for i in var_mod:

    df[i] = le.fit_transform(df[i])

df.head(10)
```

Out[3]:

	age_gt_60	air()	airBoneGap	ar_c()	ar_u()	bone()	boneAbnormal	bser()	history_buz
0	0	moderate	0	normal	normal	?	1	?	
1	1	mild	1	?	absent	mild	1	?	
2	1	mild	1	?	absent	mild	0	?	
3	1	mild	0	normal	normal	mild	1	?	
4	1	mild	0	normal	normal	mild	1	?	
5	0	mild	0	normal	normal	mild	1	?	
6	0	mild	0	normal	normal	mild	1	?	
7	0	severe	0	?	?	?	1	?	
8	1	mild	0	elevated	absent	mild	1	?	
9	1	mild	0	normal	absent	mild	1	?	

10 rows × 70 columns

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 70 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   age_gt_60                 200 non-null    int32
 1   air()                     200 non-null    object
 2   airBoneGap                200 non-null    int32
 3   ar_c()                    200 non-null    object
 4   ar_u()                    200 non-null    object
 5   bone()                    200 non-null    object
 6   boneAbnormal              200 non-null    int32
 7   bser()                    200 non-null    object
 8   history_buzzing           200 non-null    int32
 9   history_dizziness         200 non-null    int32
 10  history_fluctuating       200 non-null    int32
 11  history_fullness          200 non-null    int32
 12  history_heredity          200 non-null    int32
 13  history_nausea            200 non-null    int32
 14  history_noise             200 non-null    int32
 15  history_recruitment       200 non-null    int32
 16  history_ringing           200 non-null    int32
 17  history_roaring           200 non-null    int32
 18  history_vomiting          200 non-null    int32
 19  late_wave_poor            200 non-null    int32
 20  m_at_2k                   200 non-null    int32
 21  m_cond_lt_1k              200 non-null    int32
 22  m_gt_1k                   200 non-null    int32
 23  m_m_gt_2k                 200 non-null    int32
 24  m_m_sn                    200 non-null    int32
 25  m_m_sn_gt_1k              200 non-null    int32
 26  m_m_sn_gt_2k              200 non-null    int32
 27  m_m_sn_gt_500             200 non-null    int32
 28  m_p_sn_gt_2k              200 non-null    int32
 29  m_s_gt_500                200 non-null    int32
 30  m_s_sn                    200 non-null    int32
 31  m_s_sn_gt_1k              200 non-null    int32
 32  m_s_sn_gt_2k              200 non-null    int32
 33  m_s_sn_gt_3k              200 non-null    int32
 34  m_s_sn_gt_4k              200 non-null    int32
 35  m_sn_2_3k                 200 non-null    int32
 36  m_sn_gt_1k                200 non-null    int32
 37  m_sn_gt_2k                200 non-null    int32
 38  m_sn_gt_3k                200 non-null    int32
 39  m_sn_gt_4k                200 non-null    int32
 40  m_sn_gt_500               200 non-null    int32
 41  m_sn_gt_6k                200 non-null    int32
 42  m_sn_lt_1k                200 non-null    int32
 43  m_sn_lt_2k                200 non-null    int32
 44  m_sn_lt_3k                200 non-null    int32
 45  middle_wave_poor          200 non-null    int32
 46  mod_gt_4k                 200 non-null    int32
 47  mod_mixed                 200 non-null    int32
 48  mod_s_mixed               200 non-null    int32
 49  mod_s_sn_gt_500           200 non-null    int32
 50  mod_sn                    200 non-null    int32
 51  mod_sn_gt_1k              200 non-null    int32
```

```
 52  mod_sn_gt_2k              200 non-null    int32
 53  mod_sn_gt_3k              200 non-null    int32
 54  mod_sn_gt_4k              200 non-null    int32
 55  mod_sn_gt_500             200 non-null    int32
 56  notch_4k                  200 non-null    int32
 57  notch_at_4k               200 non-null    int32
 58  o_ar_c()                  200 non-null    object
 59  o_ar_u()                  200 non-null    object
 60  s_sn_gt_1k                200 non-null    int32
 61  s_sn_gt_2k                200 non-null    int32
 62  s_sn_gt_4k                200 non-null    int32
 63  speech()                  200 non-null    object
 64  static_normal             200 non-null    int32
 65  tymp()                    200 non-null    object
 66  viith_nerve_signs         200 non-null    int32
 67  wave_V_delayed            200 non-null    int32
 68  waveform_ItoV_prolonged   200 non-null    int32
 69  classification            200 non-null    object
dtypes: int32(60), object(10)
memory usage: 62.6+ KB
```

In [5]:
```
df['speech()'].value_counts()
```

Out[5]:
```
normal        83
good          36
very_good     35
poor          19
very_poor     18
?              6
unmeasured     3
Name: speech(), dtype: int64
```

In [6]:
```
le = LabelEncoder()
data_cat=df['speech()']
data_cat_encoded= le.fit_transform(data_cat)
data_cat_encoded= pd.DataFrame(data_cat_encoded,columns=["speech()"])
df['speech()']=data_cat_encoded
df['speech()'].value_counts()
```

Out[6]:
```
2    83
1    36
5    35
3    19
6    18
0     6
4     3
Name: speech(), dtype: int64
```

In [7]:
```
df['air()'].value_counts()
```

Out[7]:
```
mild        101
normal       73
moderate     17
severe        8
profound      1
Name: air(), dtype: int64
```

In [8]:
```
le = LabelEncoder()
data_cat=df['air()']
data_cat_encoded= le.fit_transform(data_cat)
data_cat_encoded= pd.DataFrame(data_cat_encoded,columns=['air()'])
df['air()']=data_cat_encoded
df['air()'].value_counts()
```

Out[8]:
```
0    101
2     73
1     17
4      8
3      1
Name: air(), dtype: int64
```

In [9]:
```
df['ar_c()'].value_counts()
```

Out[9]:
```
normal      117
absent       50
elevated     29
?             4
Name: ar_c(), dtype: int64
```

In [10]:
```
le = LabelEncoder()
data_cat=df['ar_c()']
data_cat_encoded= le.fit_transform(data_cat)
data_cat_encoded= pd.DataFrame(data_cat_encoded,columns=['ar_c()'])
df['ar_c()']=data_cat_encoded
df['ar_c()'].value_counts()
```

Out[10]:
```
3    117
1     50
2     29
0      4
Name: ar_c(), dtype: int64
```

In [11]:
```
le = LabelEncoder()
data_cat=df['ar_u()']
data_cat_encoded= le.fit_transform(data_cat)
data_cat_encoded= pd.DataFrame(data_cat_encoded,columns=['ar_u()'])
df['ar_u()']=data_cat_encoded
df['ar_u()'].value_counts()
```

Out[11]:
```
3    121
1     41
2     35
0      3
Name: ar_u(), dtype: int64
```

In [12]:
```
df['bone()'].value_counts()
```

Out[12]:
```
?             75
mild          55
normal        37
unmeasured    30
moderate       3
Name: bone(), dtype: int64
```

In [13]:
```
le = LabelEncoder()
data_cat=df['bone()']
data_cat_encoded= le.fit_transform(data_cat)
data_cat_encoded= pd.DataFrame(data_cat_encoded,columns=['bone()'])
df['bone()']=data_cat_encoded
df['bone()'].value_counts()
```

Out[13]:
```
0    75
1    55
3    37
4    30
2     3
Name: bone(), dtype: int64
```

In [14]:
```
df['bser()'].value_counts()
```

Out[14]:
```
?           196
degraded      2
normal        2
Name: bser(), dtype: int64
```

In [15]:
```
le = LabelEncoder()
data_cat=df['bser()']
data_cat_encoded= le.fit_transform(data_cat)
data_cat_encoded= pd.DataFrame(data_cat_encoded,columns=['bser()'])
df['bser()']=data_cat_encoded
df['bser()'].value_counts()
```

Out[15]:
```
0    196
2      2
1      2
Name: bser(), dtype: int64
```

In [16]:
```
df['o_ar_c()'].value_counts()
le = LabelEncoder()
data_cat=df['o_ar_c()']
data_cat_encoded= le.fit_transform(data_cat)
data_cat_encoded= pd.DataFrame(data_cat_encoded,columns=['o_ar_c()'])
df['o_ar_c()']=data_cat_encoded
df['o_ar_c()'].value_counts()
```

Out[16]:
```
3    124
1     47
2     24
0      5
Name: o_ar_c(), dtype: int64
```

In [17]:
```
df['o_ar_u()'].value_counts()
le = LabelEncoder()
data_cat=df['o_ar_u()']
data_cat_encoded= le.fit_transform(data_cat)
data_cat_encoded= pd.DataFrame(data_cat_encoded,columns=['o_ar_u()'])
df['o_ar_u()']=data_cat_encoded
df['o_ar_u()'].value_counts()
```

Out[17]:
```
3    114
1     47
2     37
0      2
Name: o_ar_u(), dtype: int64
```

In [18]:
```
df['tymp()'].value_counts()
le = LabelEncoder()
data_cat=df['tymp()']
data_cat_encoded= le.fit_transform(data_cat)
data_cat_encoded= pd.DataFrame(data_cat_encoded,columns=['tymp()'])
df['tymp()']=data_cat_encoded
df['tymp()'].value_counts()
```

Out[18]:
```
0    169
2     12
3      8
4      7
1      4
Name: tymp(), dtype: int64
```

In [19]:
```
df['tymp()'].value_counts()
```

Out[19]:
```
0    169
2     12
3      8
4      7
1      4
Name: tymp(), dtype: int64
```

In [20]:
```
le = LabelEncoder()
data_cat=df['tymp()']
data_cat_encoded= le.fit_transform(data_cat)
data_cat_encoded= pd.DataFrame(data_cat_encoded,columns=['tymp()'])
df['tymp()']=data_cat_encoded
df['tymp()'].value_counts()
```

Out[20]:
```
0    169
2     12
3      8
4      7
1      4
Name: tymp(), dtype: int64
```

In [21]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 70 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   age_gt_60                200 non-null    int32
 1   air()                    200 non-null    int32
 2   airBoneGap               200 non-null    int32
 3   ar_c()                   200 non-null    int32
 4   ar_u()                   200 non-null    int32
 5   bone()                   200 non-null    int32
 6   boneAbnormal             200 non-null    int32
 7   bser()                   200 non-null    int32
 8   history_buzzing          200 non-null    int32
 9   history_dizziness        200 non-null    int32
 10  history_fluctuating      200 non-null    int32
 11  history_fullness         200 non-null    int32
 12  history_heredity         200 non-null    int32
 13  history_nausea           200 non-null    int32
 14  history_noise            200 non-null    int32
 15  history_recruitment      200 non-null    int32
 16  history_ringing          200 non-null    int32
 17  history_roaring          200 non-null    int32
 18  history_vomiting         200 non-null    int32
 19  late_wave_poor           200 non-null    int32
 20  m_at_2k                  200 non-null    int32
 21  m_cond_lt_1k             200 non-null    int32
 22  m_gt_1k                  200 non-null    int32
 23  m_m_gt_2k                200 non-null    int32
 24  m_m_sn                   200 non-null    int32
 25  m_m_sn_gt_1k             200 non-null    int32
 26  m_m_sn_gt_2k             200 non-null    int32
 27  m_m_sn_gt_500            200 non-null    int32
 28  m_p_sn_gt_2k             200 non-null    int32
 29  m_s_gt_500               200 non-null    int32
 30  m_s_sn                   200 non-null    int32
 31  m_s_sn_gt_1k             200 non-null    int32
 32  m_s_sn_gt_2k             200 non-null    int32
 33  m_s_sn_gt_3k             200 non-null    int32
 34  m_s_sn_gt_4k             200 non-null    int32
 35  m_sn_2_3k                200 non-null    int32
 36  m_sn_gt_1k               200 non-null    int32
 37  m_sn_gt_2k               200 non-null    int32
 38  m_sn_gt_3k               200 non-null    int32
 39  m_sn_gt_4k               200 non-null    int32
 40  m_sn_gt_500              200 non-null    int32
 41  m_sn_gt_6k               200 non-null    int32
 42  m_sn_lt_1k               200 non-null    int32
 43  m_sn_lt_2k               200 non-null    int32
 44  m_sn_lt_3k               200 non-null    int32
 45  middle_wave_poor         200 non-null    int32
 46  mod_gt_4k                200 non-null    int32
 47  mod_mixed                200 non-null    int32
 48  mod_s_mixed              200 non-null    int32
 49  mod_s_sn_gt_500          200 non-null    int32
 50  mod_sn                   200 non-null    int32
 51  mod_sn_gt_1k             200 non-null    int32
```

```
 52  mod_sn_gt_2k              200 non-null    int32
 53  mod_sn_gt_3k              200 non-null    int32
 54  mod_sn_gt_4k              200 non-null    int32
 55  mod_sn_gt_500             200 non-null    int32
 56  notch_4k                  200 non-null    int32
 57  notch_at_4k               200 non-null    int32
 58  o_ar_c()                  200 non-null    int32
 59  o_ar_u()                  200 non-null    int32
 60  s_sn_gt_1k                200 non-null    int32
 61  s_sn_gt_2k                200 non-null    int32
 62  s_sn_gt_4k                200 non-null    int32
 63  speech()                  200 non-null    int32
 64  static_normal             200 non-null    int32
 65  tymp()                    200 non-null    int64
 66  viith_nerve_signs         200 non-null    int32
 67  wave_V_delayed            200 non-null    int32
 68  waveform_ItoV_prolonged   200 non-null    int32
 69  classification            200 non-null    object
dtypes: int32(68), int64(1), object(1)
memory usage: 56.4+ KB
```

In [22]:
```
le = LabelEncoder()
data_cat=df['classification']
data_cat_encoded= le.fit_transform(data_cat)
data_cat_encoded= pd.DataFrame(data_cat_encoded,columns=['classification'])
df['classification']=data_cat_encoded
df['classification'].value_counts()
```

Out[22]:
```
7     48
2     46
18    20
3     18
6     16
22     8
9      6
14     5
11     4
21     4
19     4
15     3
5      2
8      2
23     2
12     2
13     2
17     2
10     1
16     1
4      1
20     1
1      1
0      1
Name: classification, dtype: int64
```

In [23]:
```
df_label=df["classification"].copy()
```

In [24]:
```
df=df.drop("bser()",axis = 1)
```

In [25]:
```
from sklearn.preprocessing import StandardScaler

#feature_scal = StandardScaler()
#df = pd.DataFrame(feature_scal.fit_transform(df), columns=df.columns)
#df.head()
y=df.classification
x = df.drop(columns=['classification'])
```

In [26]:

```
df=df.drop("classification",axis = 1)
median = df['mod_sn'].median()
df['mod_sn'].fillna(median)
df.head(50)
```

Out[26]:

	age_gt_60	air()	airBoneGap	ar_c()	ar_u()	bone()	boneAbnormal	history_buzzing	history_
0	0	1	0	3	3	0	1	0	
1	1	0	1	0	1	1	1	0	
2	1	0	1	0	1	1	0	0	
3	1	0	0	3	3	1	1	0	
4	1	0	0	3	3	1	1	0	
5	0	0	0	3	3	1	1	0	
6	0	0	0	3	3	1	1	0	
7	0	4	0	0	0	0	1	0	
8	1	0	0	2	1	1	1	0	
9	1	0	0	3	1	1	1	0	
10	1	0	0	3	3	0	1	0	
11	0	4	0	3	3	2	1	0	
12	0	2	0	2	3	1	0	0	
13	1	0	0	3	3	1	1	0	
14	1	0	0	3	2	1	1	0	
15	0	0	0	3	3	1	1	0	
16	0	0	0	3	2	1	1	0	
17	0	2	0	3	3	0	0	0	
18	0	2	0	3	2	0	0	0	
19	1	0	0	3	3	0	1	0	
20	1	0	0	3	3	0	1	0	
21	0	2	0	3	3	0	1	0	
22	0	2	0	3	3	3	0	0	
23	0	1	0	2	2	0	1	0	
24	0	2	0	3	3	3	1	0	
25	0	2	0	3	2	3	1	0	
26	0	0	0	3	3	0	1	0	
27	0	0	0	1	1	1	1	0	
28	0	0	0	2	3	1	1	0	
29	0	0	0	3	3	0	0	0	
30	0	2	0	2	3	0	0	0	
31	0	2	0	3	2	0	0	0	
32	0	2	0	2	2	0	0	0	
33	0	1	1	1	1	1	1	0	
34	0	0	0	1	3	1	1	0	

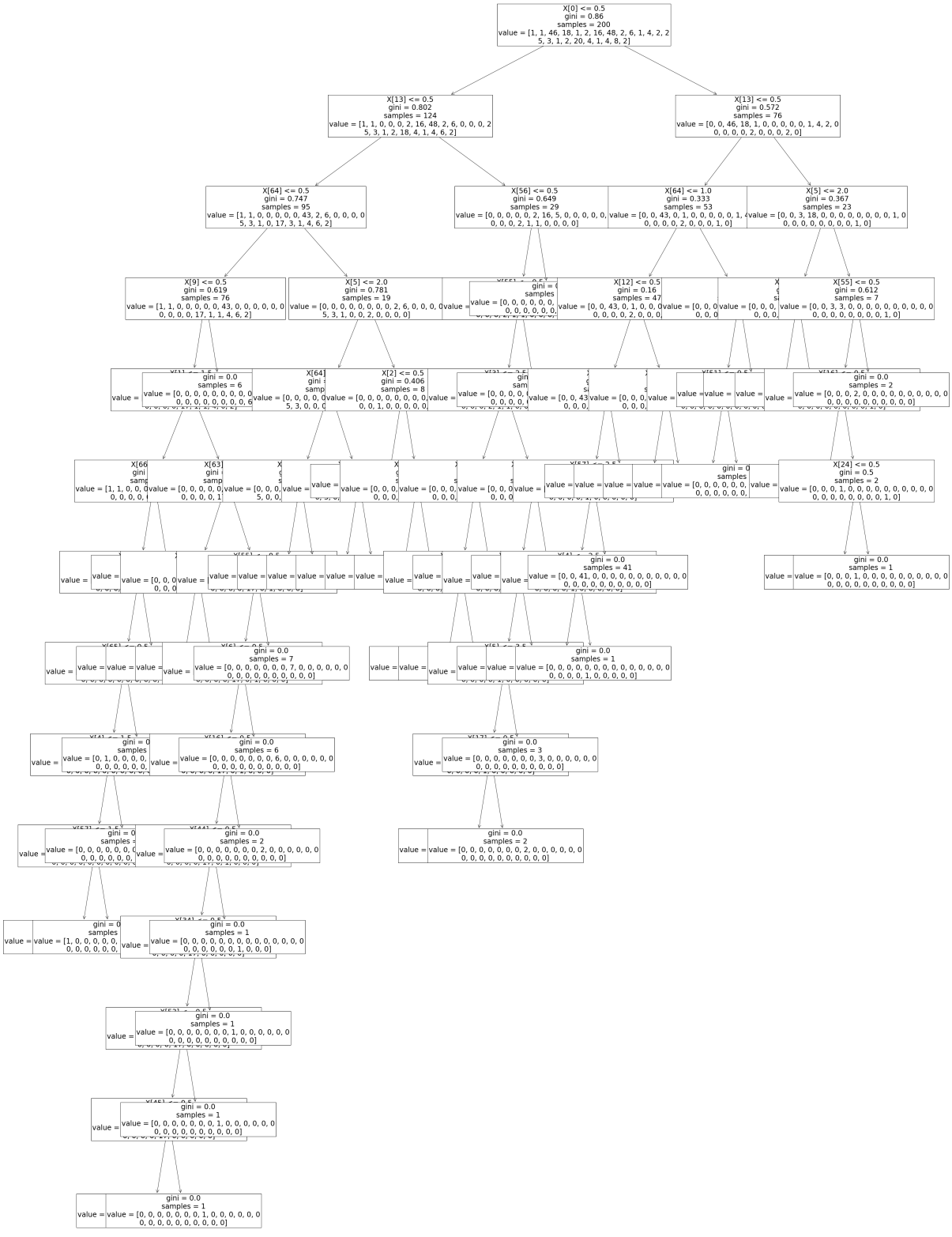
	age_gt_60	air()	airBoneGap	ar_c()	ar_u()	bone()	boneAbnormal	history_buzzing	history_
35	1	0	0	1	1	1	1	0	
36	0	2	0	1	2	3	0	0	
37	0	4	1	1	1	1	0	0	
38	1	2	0	3	3	0	0	0	
39	1	2	0	2	3	0	0	0	
40	0	2	0	3	2	3	0	0	
41	1	0	0	3	3	0	0	0	
42	0	0	0	2	3	0	0	0	
43	0	0	0	2	2	0	0	0	
44	0	1	0	1	1	2	1	1	
45	0	0	1	1	1	3	0	0	
46	0	0	0	1	1	1	1	0	
47	0	2	0	1	2	3	0	0	
48	1	1	0	3	3	0	0	0	
49	1	1	0	3	2	0	0	0	

50 rows × 68 columns

In [27]:
```
x_train,x_test,y_train,y_test =  train_test_split(x,y,test_size=0.4,random_sta
te=400)
```

In [28]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn import preprocessing
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
clft=DecisionTreeClassifier()
clft=clft.fit(x_train,y_train)
y_predt = clft.predict(x_test)


from sklearn import tree
plt.figure(figsize=(50,80))
temp = tree.plot_tree(clft.fit(x,y),fontsize=24)
plt.show()
```


X[0] <= 0.5
gini = 0.86
samples = 200
value = [1, 1, 46, 18, 1, 2, 16, 48, 2, 6, 1, 4, 2, 2
5, 3, 1, 2, 20, 4, 1, 4, 8, 2]
X[13] <= 0.5
gini = 0.802
samples = 124
value = [1, 1, 0, 0, 0, 2, 16, 48, 2, 6, 0, 0, 0, 2
5, 3, 1, 2, 18, 4, 1, 4, 6, 2]
X[13] <= 0.5
gini = 0.572
samples = 76
value = [0, 0, 46, 18, 1, 0, 0, 0, 0, 0, 1, 4, 2, 0
0, 0, 0, 0, 2, 0, 0, 0, 2, 0]
X[64] <= 0.5
gini = 0.747
samples = 95
value = [1, 1, 0, 0, 0, 0, 0, 43, 2, 6, 0, 0, 0, 0
5, 3, 1, 0, 17, 3, 1, 4, 6, 2]
X[56] <= 0.5
gini = 0.649
samples = 29
value = [0, 0, 0, 0, 0, 2, 16, 5, 0, 0, 0, 0, 0, 0
0, 0, 0, 2, 1, 1, 0, 0, 0, 0]
X[64] <= 1.0
gini = 0.333
samples = 53
value = [0, 0, 43, 0, 1, 0, 0, 0, 0, 0, 1, 4
0, 0, 0, 0, 2, 0, 0, 0, 1, 0]
X[5] <= 2.0
gini = 0.367
samples = 23
value = [0, 0, 3, 18, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0
0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
X[9] <= 0.5
gini = 0.619
samples = 76
value = [1, 1, 0, 0, 0, 0, 0, 43, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 17, 1, 1, 4, 6, 2]
X[5] <= 2.0
gini = 0.781
samples = 19
value = [0, 0, 0, 0, 0, 0, 0, 0, 2, 6, 0, 0, 0, 0
5, 3, 1, 0, 0, 2, 0, 0, 0, 0]
X[12] <= 0.5
gini = 0.16
samples = 47
value = [0, 0, 43, 0, 1, 0, 0, 0, 0, 0, 1, 0
0, 0, 0, 0, 2, 0, 0, 0, 1, 0]
X[55] <= 0.5
gini = 0.612
samples = 7
value = [0, 0, 3, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
gini = 0.0
samples = 6
value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
X[64]
gini
samples
value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0
5, 3, 0, 0, 0
X[2] <= 0.5
gini = 0.406
samples = 8
value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 1, 0, 0, 0, 0, 0]
gini = 0.0
samples = 2
value = [0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
X[66
gini
samp
value = [1, 1, 0, 0, 0
0, 0, 0, 0, 0
X[63
gini
samp
value = [0, 0, 0, 0, 0
0, 0, 0, 0, 1
X[24] <= 0.5
gini = 0.5
samples = 2
value = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
gini = 0.0
samples = 41
value = [0, 0, 41, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
gini = 0.0
samples = 1
value = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
gini = 0.0
samples = 7
value = [0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
gini = 0.0
samples = 1
value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
gini = 0.0
samples = 6
value = [0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
gini = 0.0
samples = 3
value = [0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
gini = 0.0
samples = 2
value = [0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
gini = 0.0
samples = 2
value = [0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
gini = 0.0
samples = 1
value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
gini = 0.0
samples = 1
value = [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
gini = 0.0
samples = 1
value = [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
gini = 0.0
samples = 1
value = [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

In [29]:
```
from sklearn.naive_bayes import GaussianNB
clfb = GaussianNB()
clfb.fit(x_train,y_train.ravel())
y_predb = clfb.predict(x_test)
print(classification_report(y_test,clfb.predict(x_test)))
```

```
              precision    recall  f1-score   support

           2       0.95      0.91      0.93        22
           3       0.80      0.80      0.80        10
           6       1.00      1.00      1.00         4
           7       0.86      0.75      0.80        16
           9       1.00      1.00      1.00         4
          11       0.25      1.00      0.40         1
          14       1.00      0.67      0.80         3
          15       1.00      1.00      1.00         1
          17       1.00      1.00      1.00         1
          18       0.80      0.80      0.80        10
          19       0.00      0.00      0.00         3
          21       0.00      0.00      0.00         2
          22       0.33      1.00      0.50         3

    accuracy                           0.80        80
   macro avg       0.69      0.76      0.69        80
weighted avg       0.81      0.80      0.79        80


C:\Users\Amirshnll\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1221: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples. Use `zero_division` par
ameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [30]:
```
from sklearn.neighbors import KNeighborsClassifier
k=1
clfk= KNeighborsClassifier(n_neighbors=k)
clfk.fit(x_train,y_train.ravel())
y_predk=clfk.predict(x_test)
print("when k = {} neighbors , knn test acuracy : {}" .format(k,clfk.score(x_t
est,y_test)))
print("when k = {} neighbors , knn test acuracy : {}" .format(k,clfk.score(x_t
rain,y_train)))
print(classification_report(y_test,clfk.predict(x_test)))
ran = np.arange(1,30)
train_list = []
test_list = []
for i,each in enumerate(ran):
    clfk= KNeighborsClassifier(n_neighbors=each)
    clfk.fit(x_train,y_train.ravel())
```

```
when k = 1 neighbors , knn test acuracy : 0.6125
when k = 1 neighbors , knn test acuracy : 1.0
              precision    recall  f1-score   support

           2       0.71      0.91      0.80        22
           3       1.00      0.50      0.67        10
           6       0.33      0.50      0.40         4
           7       0.44      0.50      0.47        16
           8       0.00      0.00      0.00         0
           9       1.00      0.75      0.86         4
          11       0.50      1.00      0.67         1
          14       0.60      1.00      0.75         3
          15       0.00      0.00      0.00         1
          17       1.00      1.00      1.00         1
          18       0.50      0.50      0.50        10
          19       0.00      0.00      0.00         3
          21       0.00      0.00      0.00         2
          22       1.00      0.33      0.50         3

    accuracy                           0.61        80
   macro avg       0.51      0.50      0.47        80
weighted avg       0.62      0.61      0.59        80


C:\Users\Amirshnll\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1221: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples. Use `zero_division` par
ameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Amirshnll\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1221: UndefinedMetricWarning: Recall and F-score are ill-defined and bei
ng set to 0.0 in labels with no true samples. Use `zero_division` parameter t
o control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [31]:
```
from sklearn.neural_network import MLPClassifier
clfm = MLPClassifier(hidden_layer_sizes=(5,),max_iter=1500)
clfm.fit(x_train,y_train.ravel())
y_predm = clfm.predict(x_test)
print ("acuracy:", metrics.accuracy_score (y_test,y_predm))
```

```
acuracy: 0.7625

C:\Users\Amirshnll\anaconda3\lib\site-packages\sklearn\neural_network\_multil
ayer_perceptron.py:582: ConvergenceWarning: Stochastic Optimizer: Maximum ite
rations (1500) reached and the optimization hasn't converged yet.
  warnings.warn(
```

In [ ]: