Q5 - Naive Bayes and Decision Trees

Import the libraries

```
from sklearn.naive_bayes import MultinomialNB
from IPython.display import display
from IPython.display import SVG
from id3 import export_graphviz
from id3 import Id3Estimator
from sklearn import tree

import matplotlib.pyplot as plt
import numpy as np
import graphviz
import os
%matplotlib inline
```

Load the data

```
def data_and_headers(filename):
    data = None
    with open(filename) as fp:
        data = [x.strip().split(',') for x in fp.readlines()]
    headers = data[0]
    headers = np.asarray(headers)
    class_field = len(headers) - 1
    data_x = [[x[i] for i in range(class_field)] for x in data[1:]]
    data_x = np.asarray(data_x)
    data_y = [[x[i] for i in range(class_field, class_field + 1)] for x in data[1:]]
    data_y = np.asarray(data_y)
    return headers, data_x, data_y
```

```
headers, X, Y = data_and_headers('Data' + os.sep + 'hw2q5.csv')
indexes=[int(x) for x in list(X[:,0])]
X=X[:,1:]
```

(A) K-Fold Splits

```
def createfolds(indexes):
    folds = {i:{'train':[], 'test':[]} for i in range(1,6)}
    for i in range(len(indexes)):
        for j in range(1,6):
            if indexes[i] % 5 == j-1:
                folds[j]['test'].append(i)
            else:
                folds[j]['train'].append(i)
    return folds
```

```
folds = createfolds(indexes)
```

Naive Bayes

```
X = X.tolist()
Y = np.ravel(Y).tolist()
d1={'presbyopic':2, 'pre-presbyopic':1, 'young':0, 'myope':0,
'hypermetrope':1, 'no':0, 'yes':1, 'normal':1, 'reduced':0}
d2={'Yes':1, 'No':0}
X = [[d1[X[i][j]] for j in range(len(X[0]))] for i in range(len(X))]
X = np.asarray(X)
Y = [d2[Y[i]] for i in range(len(Y))]
Y = np.asarray(Y)
```

```
cnt = 0
subheaders = headers[1:-1]
for i in sorted(folds.keys()):
    print('Fold '+str(i))
    nb = MultinomialNB(alpha=1)
    nb=nb.fit(X[folds[i]['train'],:],Y[folds[i]['train']])
    ypred=nb.predict(X[folds[i]['test'],:])
    print('\tTest IID -\t' + ', '.join([str(x) for x in
np.asarray(indexes)[folds[i]['test']]]))
    print('\tActual -\t'+', '.join(['Yes' if x==1 else 'No' for x in
Y[folds[i]['test']]]))
    print('\tPredict -\t' + ', '.join(['Yes' if x==1 else 'No' for x in
ypred]))
    for j in range(len(ypred)):
        if ypred[j]!=Y[folds[i]['test']][j]:
            cnt+=1
    print('\tProbabilities - ')
    dt={0:'Yes', 1:'No'}
    for i in range(len(nb.feature_log_prob_)):
```

```
        for j in range(len(subheaders)):
            print('\t\tP({}|Class={}) = {:.3f}'.format(subheaders[j],
dt[i], np.exp(nb.feature_log_prob_)[i][j]))
print('\nNaive-Bayes 5-fold CV accuracy - '+ str((24-cnt)*100/24) + '%')
```

```
Fold 1
    Test IID -  5, 10, 15, 20
    Actual -    No, Yes, No, Yes
    Predict -   No, No, No, No
    Probabilities -
        P(patient age|Class=Yes) = 0.472
        P(spectacle prescription|Class=Yes) = 0.194
        P(astigmatic|Class=Yes) = 0.222
        P(tear production rate|Class=Yes) = 0.111
        P(patient age|Class=No) = 0.227
        P(spectacle prescription|Class=No) = 0.227
        P(astigmatic|Class=No) = 0.182
        P(tear production rate|Class=No) = 0.364
Fold 2
    Test IID -  1, 6, 11, 16, 21
    Actual -    No, Yes, No, No, No
    Predict -   No, Yes, No, Yes, No
    Probabilities -
        P(patient age|Class=Yes) = 0.452
        P(spectacle prescription|Class=Yes) = 0.226
        P(astigmatic|Class=Yes) = 0.226
        P(tear production rate|Class=Yes) = 0.097
        P(patient age|Class=No) = 0.308
        P(spectacle prescription|Class=No) = 0.154
        P(astigmatic|Class=No) = 0.192
        P(tear production rate|Class=No) = 0.346
Fold 3
    Test IID -  2, 7, 12, 17, 22
    Actual -    Yes, No, Yes, No, Yes
    Predict -   Yes, No, No, No, No
    Probabilities -
        P(patient age|Class=Yes) = 0.444
        P(spectacle prescription|Class=Yes) = 0.222
        P(astigmatic|Class=Yes) = 0.222
        P(tear production rate|Class=Yes) = 0.111
        P(patient age|Class=No) = 0.250
        P(spectacle prescription|Class=No) = 0.200
        P(astigmatic|Class=No) = 0.200
        P(tear production rate|Class=No) = 0.350
Fold 4
```

```
    Test IID -  3, 8, 13, 18, 23
    Actual -    No, Yes, No, No, No
    Predict -   No, Yes, No, Yes, No
    Probabilities -
        P(patient age|Class=Yes) = 0.433
        P(spectacle prescription|Class=Yes) = 0.233
        P(astigmatic|Class=Yes) = 0.233
        P(tear production rate|Class=Yes) = 0.100
        P(patient age|Class=No) = 0.320
        P(spectacle prescription|Class=No) = 0.160
        P(astigmatic|Class=No) = 0.160
        P(tear production rate|Class=No) = 0.360
Fold 5
    Test IID -  4, 9, 14, 19, 24
    Actual -    Yes, No, Yes, No, No
    Predict -   Yes, No, Yes, No, No
    Probabilities -
        P(patient age|Class=Yes) = 0.419
        P(spectacle prescription|Class=Yes) = 0.258
        P(astigmatic|Class=Yes) = 0.226
        P(tear production rate|Class=Yes) = 0.097
        P(patient age|Class=No) = 0.304
        P(spectacle prescription|Class=No) = 0.174
        P(astigmatic|Class=No) = 0.174
        P(tear production rate|Class=No) = 0.348

Naive-Bayes 5-fold CV accuracy - 75.0%
```

Decision Tree

```
headers, X, Y = data_and_headers('Data' + os.sep + 'hw2q5.csv')
indexes=[int(x) for x in list(X[:,0])]
X=X[:,1:]
```

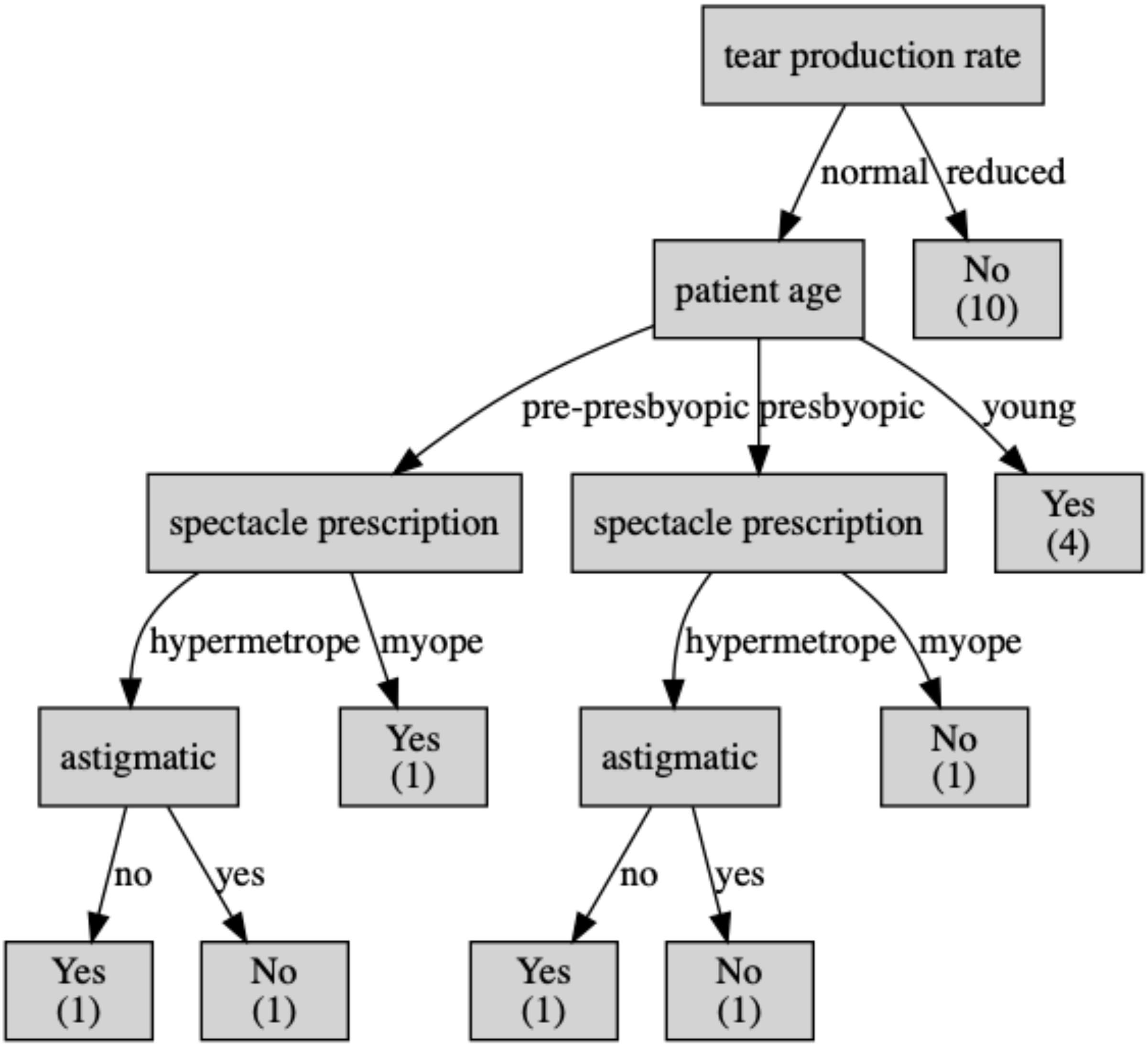
```
cnt = 0
subheaders = headers[1:-1]
for i in sorted(folds.keys()):
    print('Fold '+str(i))
    #dt = tree.DecisionTreeClassifier(criterion='entropy', splitter='best')
    dt = Id3Estimator(gain_ratio=True)
    dt = dt.fit(X[folds[i]['train'],:],Y[folds[i]['train']])
    ypred=dt.predict(X[folds[i]['test'],:])
```

```
    print('\tTest IID -\t' + ', '.join([str(x) for x in np.asarray(indexes)[folds[i]['test']]]))
    print('\tActual -\t'+', '.join(np.ravel(Y[folds[i]['test']])))
    print('\tPredict -\t' + ', '.join(ypred))
#     print('\tActual -\t'+', '.join(['Yes' if x==1 else 'No' for x in Y[folds[i]['test']]]))
#     print('\tPredict -\t' + ', '.join(['Yes' if x==1 else 'No' for x in ypred]))
    for j in range(len(ypred)):
        if ypred[j]!=Y[folds[i]['test']][j]:
            cnt+=1
    dot_data = export_graphviz(dt.tree_, 'fold'+str(i)+'.dot',feature_names = subheaders)
print('\nDecision Tree 5-fold CV accuracy - '+ str((24-cnt)*100/24) + '%')
```

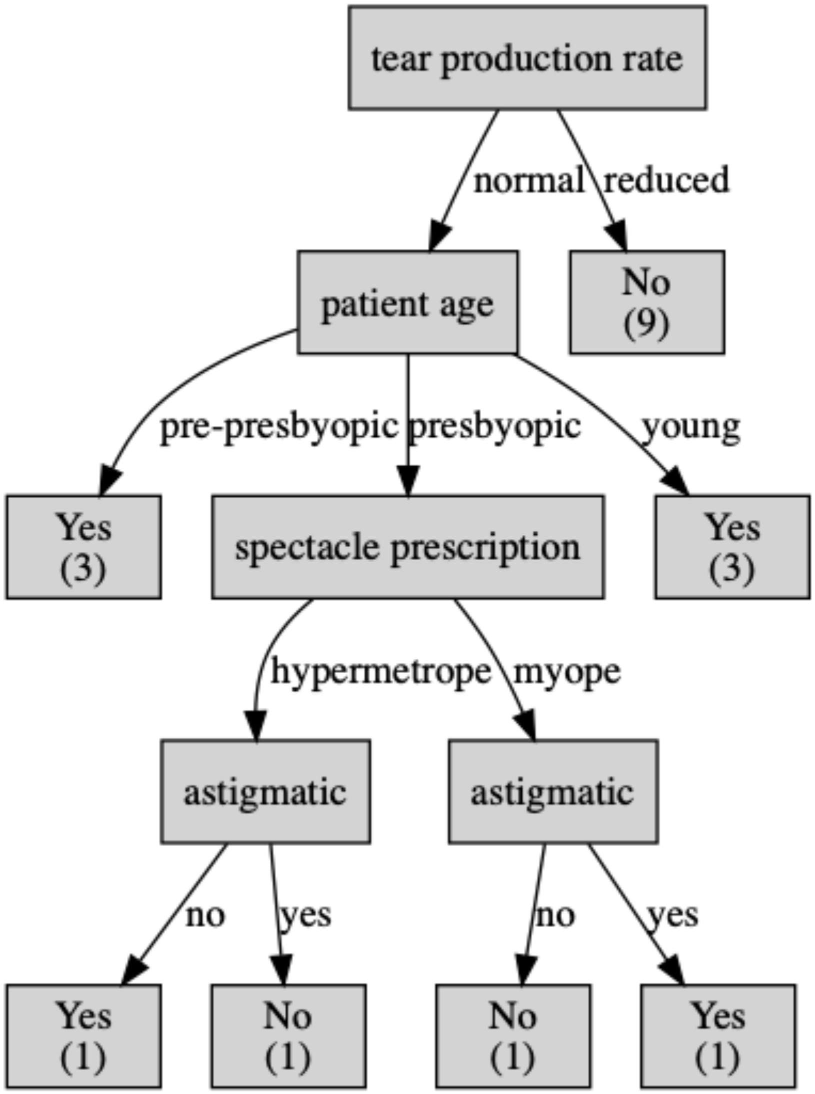
```
Fold 1
    Test IID -  5, 10, 15, 20
    Actual -    No, Yes, No, Yes
    Predict -   No, Yes, No, No
Fold 2
    Test IID -  1, 6, 11, 16, 21
    Actual -    No, Yes, No, No, No
    Predict -   No, Yes, No, Yes, No
Fold 3
    Test IID -  2, 7, 12, 17, 22
    Actual -    Yes, No, Yes, No, Yes
    Predict -   Yes, No, No, No, No
Fold 4
    Test IID -  3, 8, 13, 18, 23
    Actual -    No, Yes, No, No, No
    Predict -   No, No, No, Yes, No
Fold 5
    Test IID -  4, 9, 14, 19, 24
    Actual -    Yes, No, Yes, No, No
    Predict -   Yes, No, No, No, Yes

Decision Tree 5-fold CV accuracy - 66.66666666666667%
```

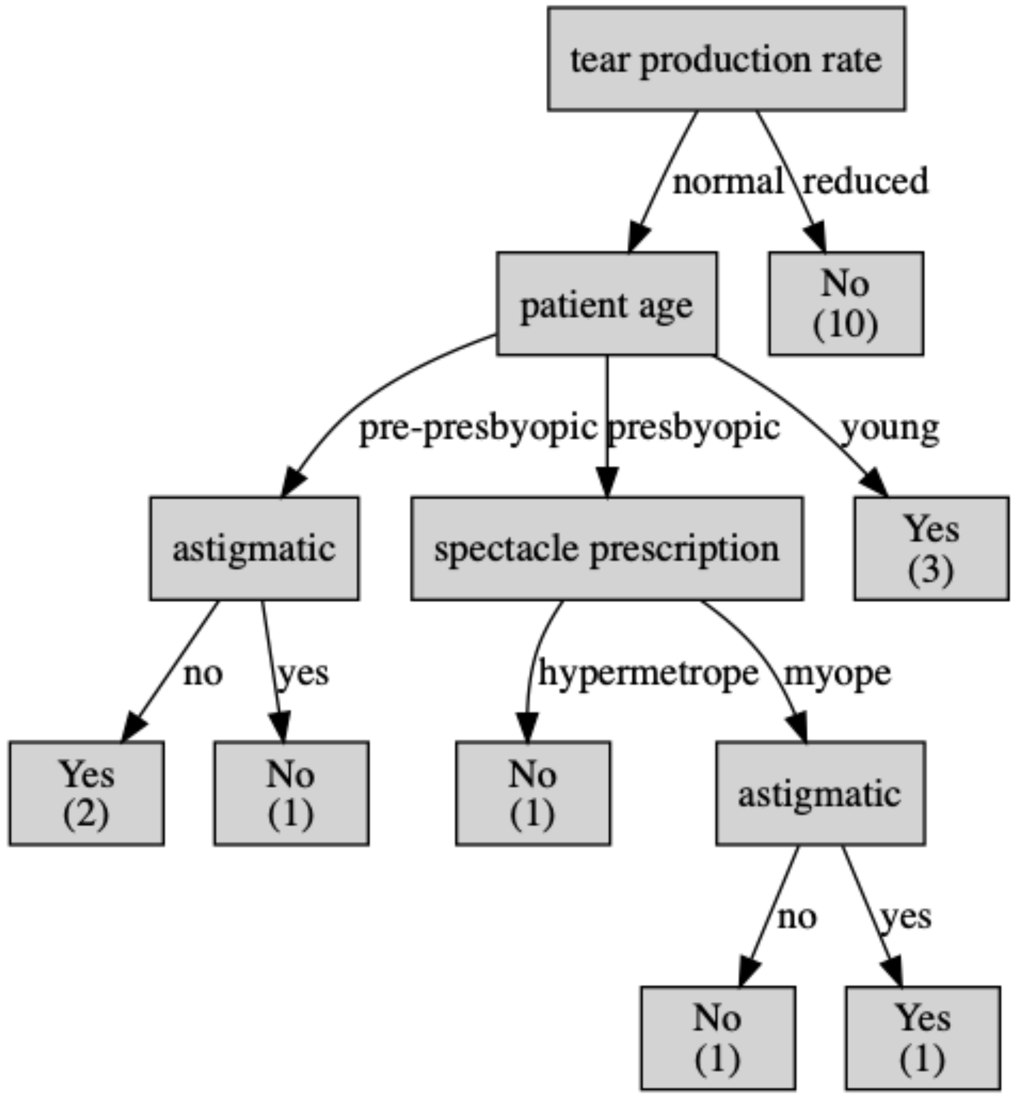
Fold 1


tear production rate
normal
reduced
patient age
No
(10)
pre-presbyopic
presbyopic
young
spectacle prescription
spectacle prescription
Yes
(4)
hypermetrope
myope
hypermetrope
myope
astigmatic
Yes
(1)
astigmatic
No
(1)
no
yes
no
yes
Yes
(1)
No
(1)
Yes
(1)
No
(1)

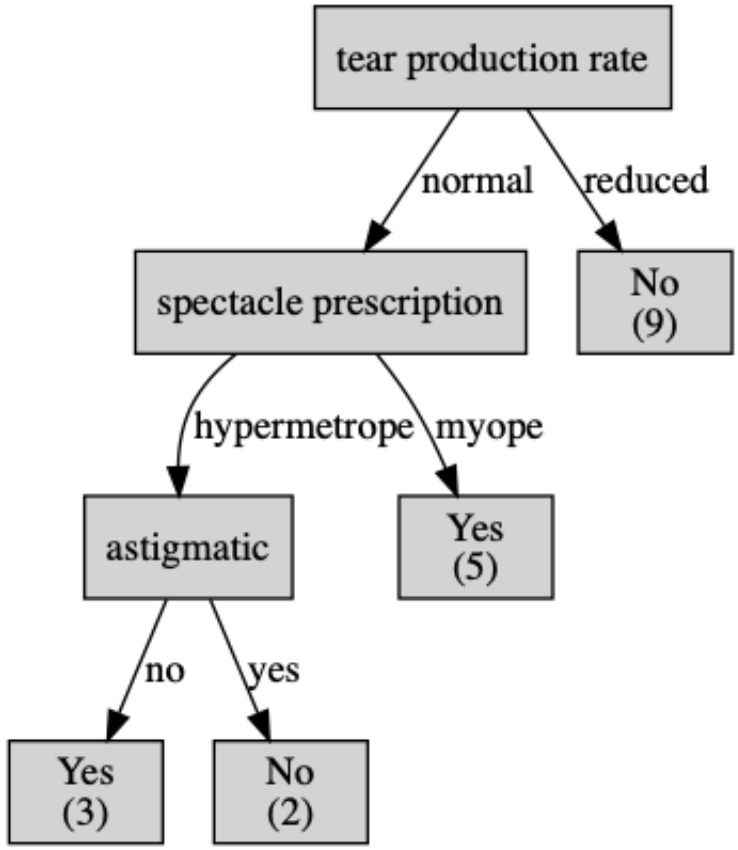
Fold 2


tear production rate
normal
reduced
patient age
No
(9)
pre-presbyopic
presbyopic
young
Yes
(3)
spectacle prescription
Yes
(3)
hypermetrope
myope
astigmatic
astigmatic
no
yes
no
yes
Yes
(1)
No
(1)
No
(1)
Yes
(1)

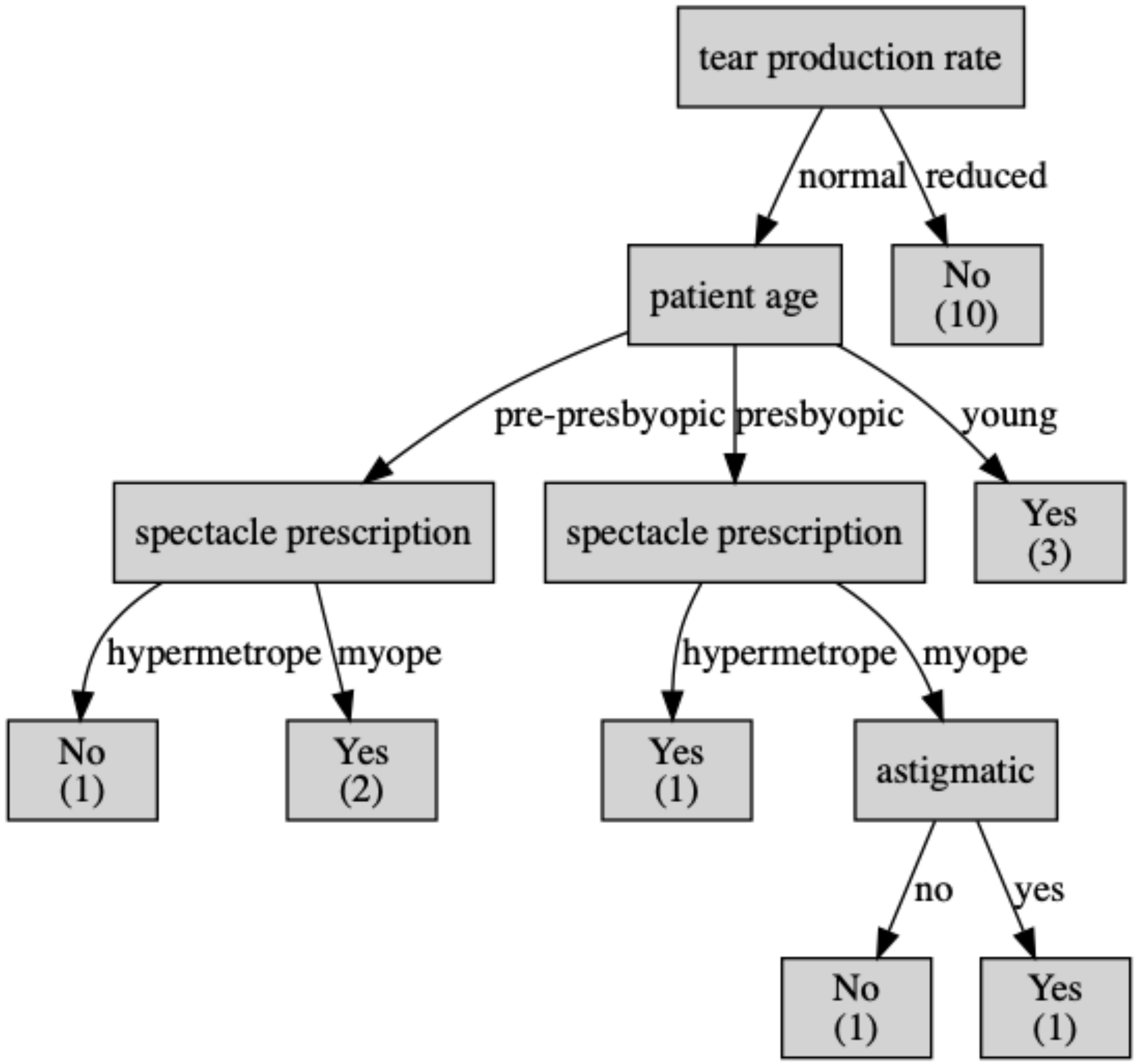
Fold 3


tear production rate
normal
reduced
patient age
No
(10)
pre-presbyopic
presbyopic
young
astigmatic
spectacle prescription
Yes
(3)
no
yes
hypermetrope
myope
Yes
(2)
No
(1)
No
(1)
astigmatic
no
yes
No
(1)
Yes
(1)

Fold 4


tear production rate
normal
reduced
spectacle prescription
No
(9)
hypermetrope
myope
astigmatic
Yes
(5)
no
yes
Yes
(3)
No
(2)

Fold 5



(B) Choosing Model

Based on the above 5-fold CV accuracy, it seems that Naive Bayes is the better model for this dataset.

Naive Bayes for Full Data

```
X = X.tolist()
Y = np.ravel(Y).tolist()
d1={'presbyopic':2, 'pre-presbyopic':1, 'young':0, 'myope':0,
'hypermetrope':1, 'no':0, 'yes':1, 'normal':1, 'reduced':0}
d2={'Yes':1, 'No':0}
X = [[d1[X[i][j]] for j in range(len(X[0]))] for i in range(len(X))]
X = np.asarray(X)
Y = [d2[Y[i]] for i in range(len(Y))]
Y = np.asarray(Y)
```

```python
nb = MultinomialNB(alpha=1)
nb=nb.fit(X,Y)
print('Final Model for Naive Bayes')
print('Probabilities - ')
dt={0:'Yes', 1:'No'}
for i in range(len(nb.feature_log_prob_)):
    for j in range(len(subheaders)):
        print('\tP({}|Class={}) = {:.3f}'.format(subheaders[j], dt[i], np.exp(nb.feature_log_prob_)[i][j]))
```

Model Details -

```
Final Model for Naive Bayes
Probabilities -
    P(patient age|Class=Yes) = 0.450
    P(spectacle prescription|Class=Yes) = 0.225
    P(astigmatic|Class=Yes) = 0.225
    P(tear production rate|Class=Yes) = 0.100
    P(patient age|Class=No) = 0.286
    P(spectacle prescription|Class=No) = 0.179
    P(astigmatic|Class=No) = 0.179
    P(tear production rate|Class=No) = 0.357
```