ALDA HOMEWORK 2

Team H29: Aakarsh Satish (Unity ID: asatish2)

Sravya Yepuri (Unity ID: syepuri)

Utkarsh Sharma (Unity ID: usharma3)

Github repository:

https://github.ncsu.edu/syepuri/engr-ALDA-Fall2023-H29

Question 1:

(a) (2 points) Load the data. Report the size of the training and testing sets. How many Class (0), and Class (1) samples are in the training set and the testing set, respectively?

Printing the Dimensions of dataset

```
print("Train dataset shape : ")
print(trainData.shape)
print("Test dataset shape : ")
print(testData.shape)
```

```
Train dataset shape :
(125, 14)
Test dataset shape :
(53, 14)
```

```
[34] print("Train dataset size : ")
     print(trainData.size)
     print("Test dataset size : ")
     print(testData.size)
```

```
Train dataset size :
1750
Test dataset size :
742
```

```
print("Class distribution for train data")
print("Class 0")
print(y_train.value_counts()[0])
print("Class 1")
print(y_train.value_counts()[1])
print("Class distribution for test data")
print("Class 0")
print(y_test.value_counts()[0])
print("Class 1")
print(y_test.value_counts()[1])
```

```
Class distribution for train data
Class 0
37
Class 1
51
Class distribution for test data
Class 0
22
Class 1
20
```

(b) (18 points) Preprocessing Data-Normalization: Please run normalization on all input features in both the training and testing datasets to obtain the normalized training and the normalized testing datasets. (Hint: you need to use the min/max of each column in the training dataset to normalize the testing dataset, and do NOT normalize the output "Class" of data.)

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
normalized_x_train = scaler.fit_transform(x_train)
normalized_x_test = scaler.transform(x_test)
```

i. **(2 points) Calculate the covariance matrix of the NEW training dataset. Please 1) specify the dimension of the resulted covariance matrix and 2) given the space limitation, please report the first 5 * 5 of the covariance matrix, that is, only reporting the first five rows and the first five columns of the entire covariance matrix.**

```
cov_normalized_x_train = np.cov(normalized_x_train, rowvar=False)
print(cov_normalized_x_train.shape)
print(cov_normalized_x_train[0:5, 0:5])
```

```
(13, 13)
[[ 0.05437915  0.00536061  0.00578453 -0.01417342  0.01611634]
 [ 0.00536061  0.05209397  0.00683929  0.01195127 -0.00075704]
 [ 0.00578453  0.00683929  0.02204741  0.01117852  0.0110121 ]
 [-0.01417342  0.01195127  0.01117852  0.02998937 -0.00311085]
 [ 0.01611634 -0.00075704  0.0110121  -0.00311085  0.03856216]]
```

ii. **(2 points) Calculate the eigenvalues and the eigenvectors based on the entire covariance matrix in (i) above. Report the size of the covariance matrix and the 5 largest eigenvalues.**

```
from numpy.linalg import eig
eigenvalues, eigenvectors = eig(cov_normalized_x_train)
print(cov_normalized_x_train.size)
print(eigenvalues[:5])
print(eigenvalues.size)
```

```
169
[0.23832336 0.12628503 0.05035845 0.04003988 0.0366412 ]
13
```

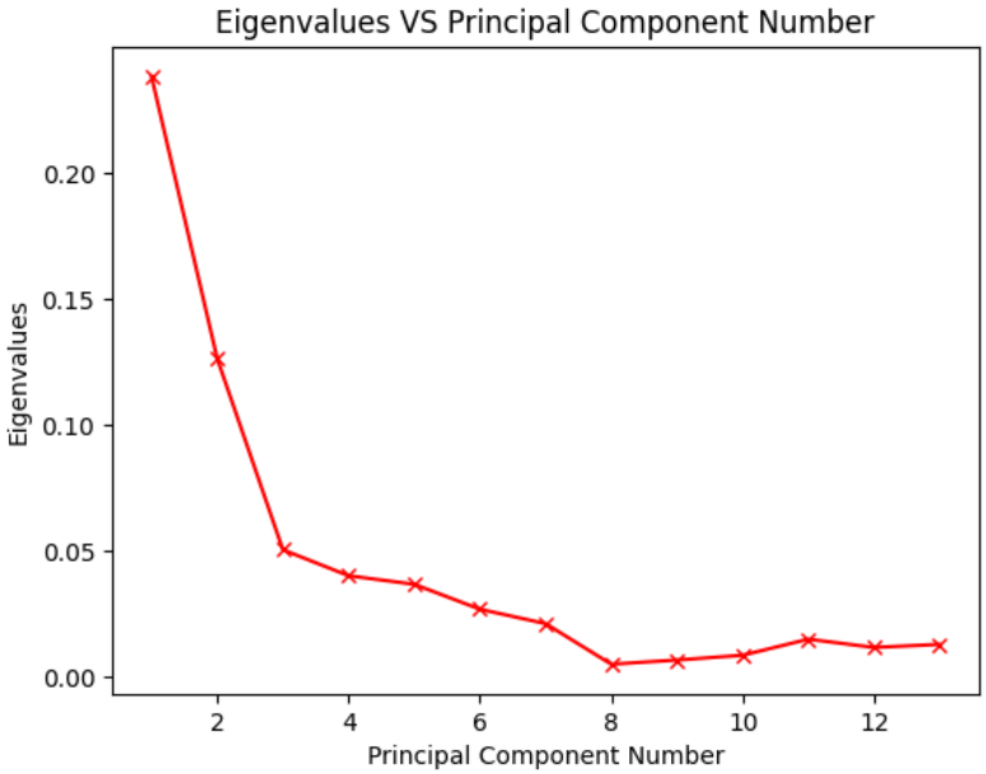
iii. **(1 point) Display the eigenvalues using a bar graph or a plot and choose a reasonable number(s) of eigenvectors. Justify your answer.**

```
import matplotlib.pyplot as plt

dimensionCount = list(range(1, len(eigenvalues)+1))

plt.plot(dimensionCount, eigenvalues, "x-r")

plt.xlabel("Principal Component Number")
plt.ylabel("Eigenvalues")
plt.title("Eigenvalues VS Principal Component Number")
plt.show()
```



The number of components to be chosen correctly would be "3" because the steepness of slope drastically reduces after crossing 3 principal components.

iv. **(13 points) Next, you will combine PCA with a K-nearest neighbor (KNN) classifier. More specifically, PCA will be applied to reduce the dimensionality of data by transforming the original data into p (p ≤ 30) principal components; and then KNN (K = 5, Euclidean distance as distance metric) will be employed to the p principal components for classification.**

a. **(5 points) Report the accuracy of the NEW testing dataset when using PCA (p = 5) with 5NN. To show your work, please submit the corresponding .csv file (including the name of .csv file in your answer below). Your .csv file should have 7 columns: columns 1-5 are the 5 principal components, column 6 is the original ground truth output "Class", and the last column is the predicted output "Class"**

```
from sklearn.decomposition import PCA

principalComponents = 5 #given in the question
kNearestNeighbors = 5 #given in the question

PCA = PCA(n_components=principalComponents)
normalized_x_train_pca = PCA.fit_transform(normalized_x_train)
normalized_x_test_pca = PCA.transform(normalized_x_test)
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
KNN = KNeighborsClassifier(n_neighbors=kNearestNeighbors, metric='euclidean')
KNN.fit(normalized_x_train_pca, y_train)
y_test_predicted = KNN.predict(normalized_x_test_pca)

modelAccuracy = accuracy_score(y_test, y_test_predicted)
print("Model accuracy for 5 principal components and then 5 nearest neighbors is: ")
print(modelAccuracy)

modelOutput = pd.DataFrame(data=normalized_x_test_pca, columns=['PC1', 'PC2', 'PC3', 'PC4', 'PC5'])
modelOutput['Ground Truth (CLASS)'] = y_test
modelOutput['Predicted Output (CLASS)'] = y_test_predicted
modelOutput.to_csv('Normalized_PCA_KNN_OUTPUT.csv', index=False)
```

```
Model accuracy for 5 principal components and then 5 nearest neighbors is:
0.9622641509433962
```

Output file: Normalized_PCA_KNN_OUTPUT.csv

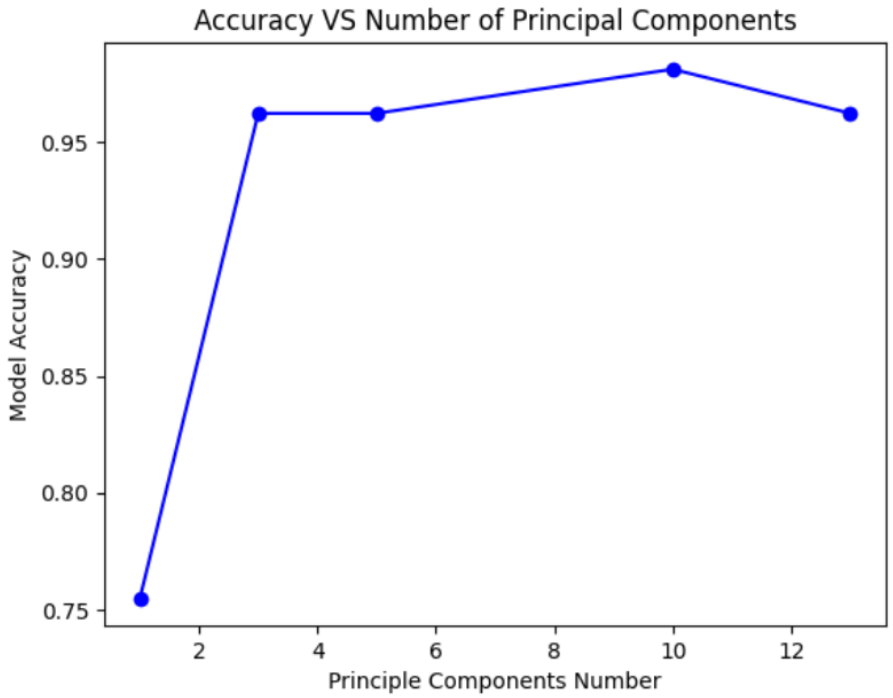
b. **6 points) Plot your results by varying p: 1, 3, 5, 10, 15, and 20 respectively. In your plot, the x-axis represents the number of principal components, and the y-axis refers to the accuracy of the NEW testing dataset using the corresponding number of principal components and 5NN.**

```python
from sklearn.decomposition import PCA
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

possiblePrincipleComponents = [1, 3, 5, 10, 13]
modelAccuracies = []

for i in range(0, len(possiblePrincipleComponents)):
  PCA_iComponents = PCA(n_components = possiblePrincipleComponents[i])
  normalized_x_train_i_pca = PCA_iComponents.fit_transform(normalized_x_train)
  normalized_x_test_i_pca = PCA_iComponents.transform(normalized_x_test)
  KNN = KNeighborsClassifier(n_neighbors=5, metric='euclidean')
  KNN.fit(normalized_x_train_i_pca, y_train)
  y_test_predicted = KNN.predict(normalized_x_test_i_pca)
  accuracy_i = accuracy_score(y_test, y_test_predicted)
  modelAccuracies.append(accuracy_i)

plt.plot(possiblePrincipleComponents, modelAccuracies, '-bo')
plt.xlabel('Principle Components Number')
plt.ylabel('Model Accuracy')
plt.title('Accuracy VS Number of Principal Components')
plt.show()
print("Model Accuracies:")
print(modelAccuracies)
```




```
Model Accuracies:
[0.7547169811320755, 0.9622641509433962, 0.9622641509433962, 0.9811320754716981, 0.9622641509433962]
```

c. 2 point) Based upon the (PCA + 5NN)'s results above, what is the most "reasonable" number of principal components among all the choices? Justify your answer.

Answer:

For 3 components (96.226% accuracy) vs 10 components (98.113% accuracy), the accuracy is a little less than 2%. That trade-off can be made with a compromise with accuracy by retaining the model simplicity that comes with 3 principal components. Hence, the reasonable number of principal components is 3. Doing a 10 dimension would be too much computationally expensive.

(c) (18 points) Preprocess Data-Standardization: Similarly, please run standardization on all input features to obtain the standardized training and the standardized testing datasets. Then repeat the four steps i-iv in (b) above on the two NEW standardized datasets.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
standardized_x_train = scaler.fit_transform(x_train) #standardized x_train
standardized_x_test = scaler.transform(x_test) #standardized y_train
```

i. (2 points) Calculate the covariance matrix of the NEW training dataset. Please 1) specify the dimension of the resulted covariance matrix and 2) given the space limitation, please report the first 5 ∗ 5 of the covariance matrix, that is, only reporting the first five rows and the first five columns of the entire covariance matrix.

```
cov_standardized_x_train = np.cov(standardized_x_train, rowvar=False)
print(cov_standardized_x_train.shape)
print(cov_standardized_x_train[0:5, 0:5])
```

```
(13, 13)
[[ 1.00806452  0.10152955  0.16840753 -0.35380446  0.35477897]
 [ 0.10152955  1.00806452  0.20343545  0.30480694 -0.01702669]
 [ 0.16840753  0.20343545  1.00806452  0.43823826  0.38071409]
 [-0.35380446  0.30480694  0.43823826  1.00806452 -0.09221525]
 [ 0.35477897 -0.01702669  0.38071409 -0.09221525  1.00806452]]
```

ii. (2 points) Calculate the eigenvalues and the eigenvectors based on the entire covariance matrix in (i) above. Report the size of the covariance matrix and the 5 largest eigenvalues.

```
from numpy.linalg import eig
eigenvalues, eigenvectors = eig(cov_standardized_x_train)
print(cov_standardized_x_train.size)
print(eigenvalues[:5])
print(eigenvalues.size)
```

```
169
[4.75913466 2.6488837  1.47962088 1.01298973 0.77218735]
13
```

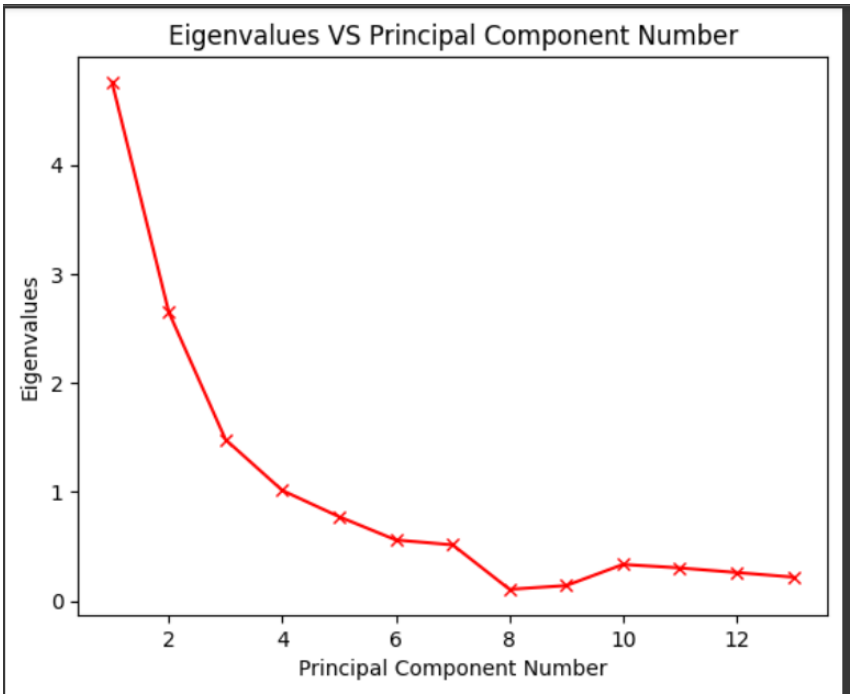
iii. **(1 point) Display the eigenvalues using a bar graph or a plot and choose a reasonable number(s) of eigenvectors. Justify your answer**

```
import matplotlib.pyplot as plt

dimensionCount = list(range(1, len(eigenvalues)+1))

plt.plot(dimensionCount, eigenvalues, "x-r")

plt.xlabel("Principal Component Number")
plt.ylabel("Eigenvalues")
plt.title("Eigenvalues VS Principal Component Number")
plt.show()
```



The number of components to be chosen correctly would be "3" because the steepness of slope drastically reduces after crossing 3 principal components.

iv. **(13 points) Next, you will combine PCA with a K-nearest neighbor (KNN) classifier. More specifically, PCA will be applied to reduce the dimensionality of data by transforming the original data into p (p ≤ 30) principal components; and then KNN (K = 5, Euclidean distance as distance metric) will be employed to the p principal components for classification.**

a. **(5 points) Report the accuracy of the NEW testing dataset when using PCA (p = 5) with 5NN. To show your work, please submit the corresponding .csv file (including the name of .csv file in your answer below). Your .csv file should have 7 columns: columns 1-5 are the 5 principal components, column 6 is the original ground truth output "Class", and the last column is the predicted output "Class".**

```
from sklearn.decomposition import PCA

principalComponents = 5 #given in the question
kNearestNeighbors = 5 #given in the question

PCA = PCA(n_components=principalComponents)
standardized_x_train_pca = PCA.fit_transform(standardized_x_train)
standardized_x_test_pca = PCA.transform(standardized_x_test)
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
KNN = KNeighborsClassifier(n_neighbors=kNearestNeighbors, metric='euclidean')
KNN.fit(standardized_x_train_pca, y_train)
y_test_predicted = KNN.predict(standardized_x_test_pca)

modelAccuracy = accuracy_score(y_test, y_test_predicted)
print("Model accuracy for 5 principal components and then 5 nearest neighbors is: ")
print(modelAccuracy)

modelOutput = pd.DataFrame(data=standardized_x_test_pca, columns=['PC1', 'PC2', 'PC3', 'PC4', 'PC5'])
modelOutput['Ground Truth (CLASS)'] = y_test
modelOutput['Predicted Output (CLASS)'] = y_test_predicted
modelOutput.to_csv('Standardized_PCA_KNN_OUTPUT.csv', index=False)
```

```
Model accuracy for 5 principal components and then 5 nearest neighbors is:
0.9245283018867925
```

Output file: Standardized_PCA_KNN_OUTPUT.csv

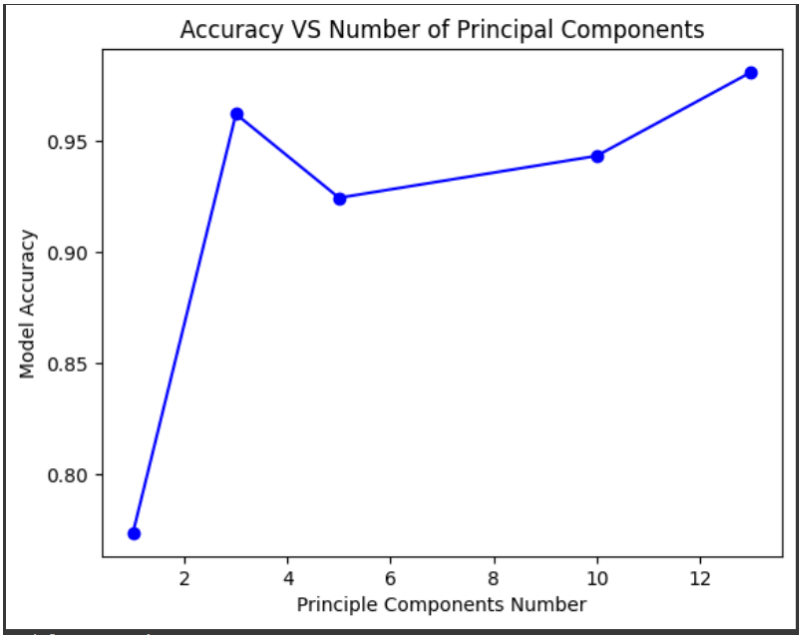
b. **6 points) Plot your results by varying p: 1, 3, 5, 10, 15, and 20 respectively. In your plot, the x-axis represents the number of principal components and the y-axis refers to the accuracy of the NEW testing dataset using the corresponding number of principal components and 5NN.**

```
from sklearn.decomposition import PCA
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

possiblePrincipleComponents = [1, 3, 5, 10, 13]
modelAccuracies = []

for i in range(0, len(possiblePrincipleComponents)):
  PCA_iComponents = PCA(n_components = possiblePrincipleComponents[i])
  standardized_x_train_i_pca = PCA_iComponents.fit_transform(standardized_x_train)
  standardized_x_test_i_pca = PCA_iComponents.transform(standardized_x_test)
  KNN = KNeighborsClassifier(n_neighbors=5, metric='euclidean')
  KNN.fit(standardized_x_train_i_pca, y_train)
  y_test_predicted = KNN.predict(standardized_x_test_i_pca)
  accuracy_i = accuracy_score(y_test, y_test_predicted)
  modelAccuracies.append(accuracy_i)

plt.plot(possiblePrincipleComponents, modelAccuracies, '-bo')
plt.xlabel('Principle Components Number')
plt.ylabel('Model Accuracy')
plt.title('Accuracy VS Number of Principal Components')
plt.show()
print("Model Accuracies:")
print(modelAccuracies)
```




```
Model Accuracies:
[0.7735849056603774, 0.9622641509433962, 0.9245283018867925, 0.9433962264150944, 0.9811320754716981]
```

c. **(2 point) Based upon the (PCA + 5NN)'s results above, what is the most "reasonable" number of principal components among all the choices? Justify your answer.**

Answer:

For 3 components (96.226% accuracy) vs 13 components (98.113% accuracy), the accuracy is a little less than 2%. But again, the trade-off can be made with a compromise with accuracy by retaining the model simplicity that comes with 3 principal components. Hence, the reasonable number of principal components is 3.

(d) (2 points) Comparing the results from (b) and (c), which of the two data-processing procedures, normalization or standardization, would you prefer for the given datasets? And why? (Answer without any justification will get zero points.)

Answer:

I would say that doing normalization helped achieve the top accuracy on 10 principal components as compared to 13 in the standardization.

Also, the accuracies were more consistently better with normalization.

Therefore, we would prefer normalization for the given datasets.

2) (a)

CLASS	CONSTANCY	ERR	SIGN	DIRECTION
Auto	xConst	XL	nn	H
Manual	xConst	XL	nn	T
Manual	Const	XL	pp	T
Manual	Const	XL	pp	T
Manual	Const	MM	nn	T
Auto	Const	SS	pp	H
Auto	Const	SS	pp	T
Auto	Const	SS	pp	H
Auto	Const	SS	nn	H
Auto	Const	MM	pp	H
Auto	Const	MM	pp	H
Auto	Const	MM	pp	T
Auto	Const	MM	pp	T

STEP 1:

$$H(\text{Class}) = -P(\text{Class}=\text{Auto}) \cdot \log_2(P(\text{Class}=\text{Auto})) - P(\text{Class}=\text{Manual}) \cdot \log_2(P(\text{Class}=\text{Manual}))$$

$$= -\frac{9}{13} \cdot \log_2\left(\frac{9}{13}\right) - \frac{4}{13} \cdot \log_2\left(\frac{4}{13}\right)$$

$$= 0.89049$$

~~$H(\text{Class} \mid \text{Constancy}) = P(\text{Constancy}) \cdot H(\text{Class})$~~

$$H(\text{Class} \mid \text{Constancy}) = P(\text{Constancy}=\text{xConst}) \cdot H(\text{Class} \mid \text{Constancy}=\text{xConst}) + P(\text{Constancy}=\text{Const}) \cdot H(\text{Class} \mid \text{Constancy}=\text{Const})$$

$$= \frac{2}{13} \cdot \left[\frac{-1}{2}\lg\left(\frac{1}{2}\right) - \frac{1}{2}\lg\left(\frac{1}{2}\right)\right] + \frac{11}{13} \cdot \left[\frac{-3}{11}\lg\left(\frac{3}{11}\right) - \frac{8}{11}\lg\left(\frac{8}{11}\right)\right]$$

$$= \frac{2}{13} \times [1] + \frac{11}{13} \times [0.84535]$$

$$= 0.86914$$

$$H(\text{Class}|\text{Err}) = P(\text{Err}=SS)\cdot H(\text{Class}|\text{Err}=SS) + P(\text{Err}=MM)\cdot H(\text{Class}|\text{Err}=MM) + P(\text{Err}=XL)\cdot H(\text{Class}|\text{Err}=XL)$$

$$= \frac{4}{13}\cdot\left[\frac{-4}{4}\cdot\lg\left(\frac{4}{4}\right) - 0\right] + \frac{5}{13}\cdot\left[\frac{-1}{5}\lg\left(\frac{1}{5}\right) - \frac{4}{5}\lg\left(\frac{4}{5}\right)\right] + \frac{4}{13}\cdot\left[\frac{-1}{4}\cdot\lg\left(\frac{1}{4}\right) - \frac{3}{4}\cdot\lg\left(\frac{3}{4}\right)\right]$$

$$= 0 + 0.27766 + 0.24962 = 0.52728$$

$$H(\text{Class}|\text{Sign}) = P(\text{Sign}=nn)\cdot H(\text{Class}|\text{Sign}=nn) + P(\text{Sign}=pp)\cdot H(\text{Class}|\text{Sign}=pp)$$

$$= \frac{4}{13}\cdot\left[\frac{-2}{4}\cdot\lg\left(\frac{2}{4}\right) - \frac{2}{4}\cdot\lg\left(\frac{2}{4}\right)\right] + \frac{9}{13}\cdot\left[\frac{-2}{9}\cdot\lg\left(\frac{2}{9}\right) - \frac{7}{9}\lg\left(\frac{7}{9}\right)\right]$$

$$= 0.30769 + 0.52906 = 0.83675$$

$$H(\text{Class}|\text{Direction}) = P(\text{Direction}=H)\cdot H(\text{Class}|\text{Direction}=H) + P(\text{Direction}=T)\cdot H(\text{Class}|\text{Direction}=T)$$

$$= \frac{6}{13}\cdot\left[\frac{-6}{6}\cdot\lg\left(\frac{6}{6}\right) + 0\right] + \frac{7}{13}\cdot\left[\frac{-4}{7}\cdot\lg\left(\frac{4}{7}\right) - \frac{3}{7}\cdot\lg\left(\frac{3}{7}\right)\right]$$

$$= 0 + 0.530507 = 0.530507$$

~~$IG(\text{Class}|\text{Constancy}) = 0.89049 - 0.$~~

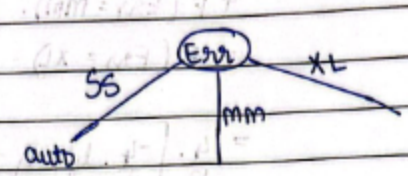
$$IG(\text{Class}|\text{Constancy}) = H(\text{Class}) - H(\text{Class}|\text{Constancy})$$
$$= 0.89049 - 0.86914 = 0.02135$$

$$IG(\text{Class}|\text{Err}) = H(\text{Class}) - H(\text{Class}|\text{Err})$$
$$= 0.89049 - 0.52728 = 0.36321$$

$$IG(\text{Class}|\text{Direction}) = H(\text{Class}) - H(\text{Class}|\text{direction})$$
$$= 0.89049 - 0.53050 = 0.35999$$

IG (Class | Err) has the greatest value.

Decision tree:



STEP 2:

① Err = MM:

CLASS	CONSTANCY	SIGN	DIRECTION
Manual	Const	mm	T
Auto	Const	pp	H
Auto	Const	pp	H
Auto	Const	pp	T
Auto	Const	pp	T

$$H(Class) = -P(Class = Auto) \cdot \log_2(P(Class = Auto)) - P(Class = Manual) \cdot \log_2(P(Class = Manual))$$

$$= -\frac{4}{5} \cdot \log_2\left(\frac{4}{5}\right) - \frac{1}{5} \cdot \log_2\left(\frac{1}{5}\right) = 0.721928$$

$$H(Class \mid Constancy) = P(Constancy = xConst) \cdot H(Class \mid Constancy = xConst) + P(Constancy = Const) \cdot H(Class \mid Constancy = Const)$$

$$= 0 + \frac{5}{5} \cdot \left[-\frac{1}{5} \cdot \log_2\left(\frac{1}{5}\right) - \frac{4}{5} \log_2\left(\frac{4}{5}\right)\right] = 0.721928$$

$$H(Class \mid Sign) = P(Sign = mm) \cdot H(Class \mid Sign = mm) + P(Sign = pp) \cdot H(Class \mid Sign = pp)$$

$$= \frac{1}{5} \cdot \left[\frac{1}{1} \cdot \log(1) + 0\right] + \frac{4}{5}\left[\frac{4}{4} \cdot \log(1)\right] = 0$$

$$H(Class \mid Direction) = P(Direction = T) \cdot H(Class \mid Direction = T) + P(Direction = H) \cdot H(Class \mid Direction = H)$$

$$= \frac{3}{5} \cdot \left[-\frac{1}{3} \cdot \log\left(\frac{1}{3}\right) - \frac{2}{3} \cdot \log\left(\frac{2}{3}\right)\right] + \frac{2}{5}[0] = 0.550977$$

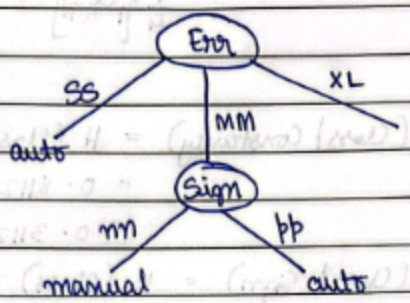
$IG(Class|Constancy) = H(Class) - H(Class|Constancy)$
$= 0.721928 - 0.721928 = 0$

$IG(Class|Sign) = H(Class) - H(Class|Sign)$
$= 0.721928 - 0 = 0.721928$

$IG(Class|Direction) = H(Class) - H(Class|Direction)$
$= 0.721928 - 0.550977 = 0.170951$

$IG(Class|Sign)$ has the greatest value.

Decision tree:



② Error = XL:

CLASS	CONSTANCY	SIGN	DIRECTION
Auto	xConst	nn	H
Manual	xConst	nn	T
Manual	Const	pp	T
Manual	Const	pp	T

$H(Class) = -P(Class = Auto) \cdot \log_2(P(Class = Auto)) - P(Class = Manual) \cdot \log_2(P(Class = Manual))$
$= -\frac{1}{4} \cdot \lg\left(\frac{1}{4}\right) - \frac{3}{4} \cdot \lg\left(\frac{3}{4}\right) = 0.811278$

$H(Class|Constancy) = P(Constancy = xConst) \cdot H(Class|Constancy = xConst) + P(Constancy = Const) \cdot H(Class|Constancy = Const)$
$= \frac{2}{4} \cdot \left[-\frac{1}{2}\lg\left(\frac{1}{2}\right) - \frac{1}{2}\lg\left(\frac{1}{2}\right)\right] + \frac{2}{4} \cdot \left[\frac{2}{2}\lg\left(\frac{2}{2}\right) + 0\right]$
$= 0.5$

$$H(\text{Class}|\text{Sign}) = P(\text{Sign}=mm)\cdot H(\text{Class}|\text{Sign}=mm) + P(\text{Sign}=pp)\cdot H(\text{Class}|\text{Sign}=pp)$$

$$= \frac{2}{4}\cdot\left[\frac{-1}{2}\lg\left(\frac{1}{2}\right)-\frac{1}{2}\lg\left(\frac{1}{2}\right)\right] + \frac{2}{4}\left[\frac{2}{2}\lg\left(\frac{2}{2}\right)\right]$$

$$= 0.5$$

$$H(\text{Class}|\text{Direction}) = P(\text{Direction}=H)\cdot H(\text{Class}|\text{Direction}=H) + P(\text{Direction}=T)\cdot H(\text{Class}|\text{Direction}=T)$$

$$= \frac{1}{4}\left[\frac{1}{1}\lg(1)\right] + \frac{3}{4}\cdot\left[\frac{3}{3}\lg\left(\frac{3}{3}\right)\right] = 0$$

$$IG(\text{Class}|\text{Constancy}) = H(\text{Class}) - H(\text{Class}|\text{Constancy})$$

$$= 0.811278 - 0.5$$

$$= 0.311278$$

$$IG(\text{Class}|\text{Sign}) = H(\text{Class}) - H(\text{Class}|\text{Sign})$$

$$= 0.811278 - 0.5$$

$$= 0.311278$$

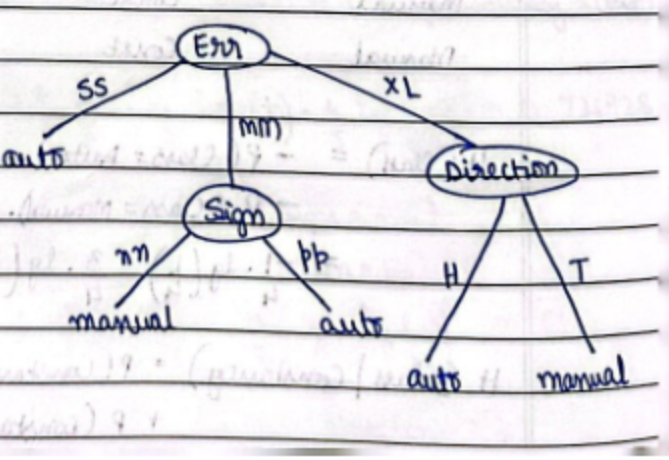
$$IG(\text{Class}|\text{Direction}) = H(\text{Class}) - H(\text{Class}|\text{Direction})$$

$$= 0.811278 - 0 = 0.811278$$

IG (Class | Direction) has the highest value.

Decision Tree :

The final decision tree is shown on the right side..



2) (b)

By taking Constancy as root, we start with the decision tree below:

Considering Constancy = x Const :

CLASS	ERR	SIGN	DIRECTION
Auto	XL	nn	H
Manual	XL	nn	T

	Auto	Manual
ERR = XL	1	1
SIGN = nn	1	1
DIRECTION = H	1	0
T	0	1

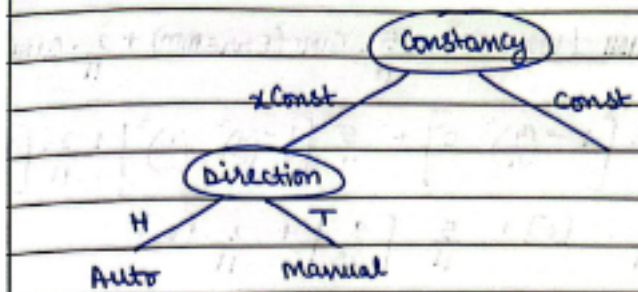
$GINI(Err) = GINI(Err = XL) = 1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2 = \frac{1}{2}$

$GINI(Sign) = GINI(Sign = nn) = 1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2 = \frac{1}{2}$

$GINI(Direction) = \frac{1}{2} \cdot GINI(Direction = H) + \frac{1}{2} \cdot GINI(Direction = T)$

$= \frac{1}{2} \cdot \left[1 - (1)^2 - 0^2\right] + \frac{1}{2} \cdot \left[1 - 0^2 - 1^2\right] = 0$

GINI (Direction) has the lowest value.



Constancy = Const:

CLASS	ERR	SIGN	DIRECTION
Manual	XL	pp	T
Manual	XL	pp	T
Manual	MM	nn	T
Auto	SS	pp	H
Auto	SS	pp	T
Auto	SS	pp	H
Auto	SS	nn	H
Auto	MM	pp	H
Auto	MM	pp	H
Auto	MM	pp	T
Auto	MM	pp	T

	Auto	Manual
ERR = XL	0	2
MM	4	1
SS	4	0
Sign = nn	1	1
pp	7	2
Direction = H	5	0
T	3	3

$$GINI(Err) = \frac{4}{11} \cdot GINI(Err=SS) + \frac{5}{11} \cdot GINI(Err=mm) + \frac{2}{11} \cdot GINI(Err=XL)$$

$$= \frac{4}{11} \cdot \left[1 - \left(\frac{4}{4}\right)^2 - 0^2\right] + \frac{5}{11} \cdot \left[1 - \left(\frac{4}{5}\right)^2 - \left(\frac{1}{5}\right)^2\right] + \frac{2}{11} \left[1 - 0^2 - \left(\frac{2}{2}\right)^2\right]$$

$$= \frac{4}{11} \cdot [0] + \frac{5}{11} \cdot \left[\frac{8}{25}\right] + \frac{2}{11} \cdot [0]$$

$$= 0.14545$$

$$GINI(Sign) = \frac{2}{11} \cdot GINI(Sign=nn) + \frac{9}{11} \cdot GINI(Sign=hp)$$

$$= \frac{2}{11} \cdot \left[1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2\right] + \frac{9}{11} \cdot \left[1 - \left(\frac{7}{9}\right)^2 - \left(\frac{2}{9}\right)^2\right]$$

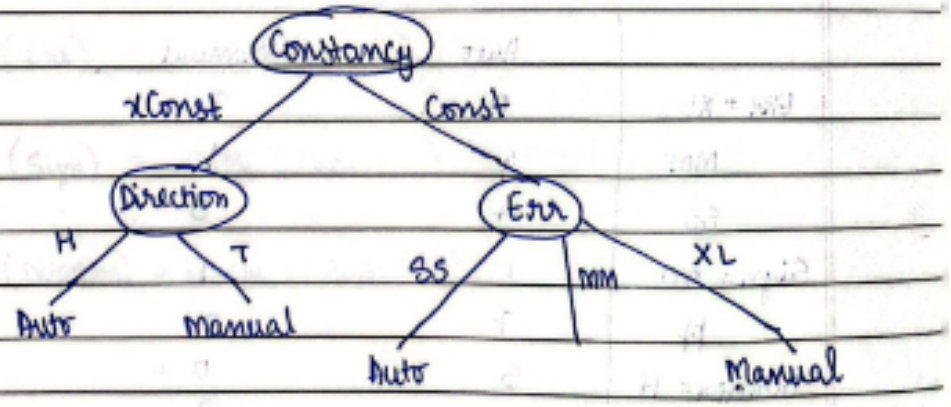
$$= \frac{2}{11} \cdot \left[\frac{1}{2}\right] + \frac{9}{11} \cdot \left[\frac{28}{81}\right] = 0.373737$$

$$GINI(Direction) = \frac{5}{11} \cdot GINI(Direction=H) + \frac{6}{11} \cdot GINI(Direction=T)$$

$$= \frac{5}{11} \cdot \left[1 - \left(\frac{5}{5}\right)^2 - 0^2\right] + \frac{6}{11} \cdot \left[1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2\right]$$

$$= \frac{5}{11} \cdot [0] + \frac{6}{11} \cdot \left[\frac{1}{2}\right] = 0.272727$$

GINI (Err) has the lowest value.



Constancy = Const and Err = MM:

CLASS	SIGN	DIRECTION
Manual	mm	T
Auto	pp	H
Auto	pp	H
Auto	pp	T
Auto	pp	T

	Auto	Manual
Sign = mm	0	1
pp	4	0
Direction = T	2	1
H	2	0

$$GINI(Sign) = \frac{1}{5} \cdot GINI(Sign = mm) + \frac{4}{5} \cdot GINI(Sign = pp)$$

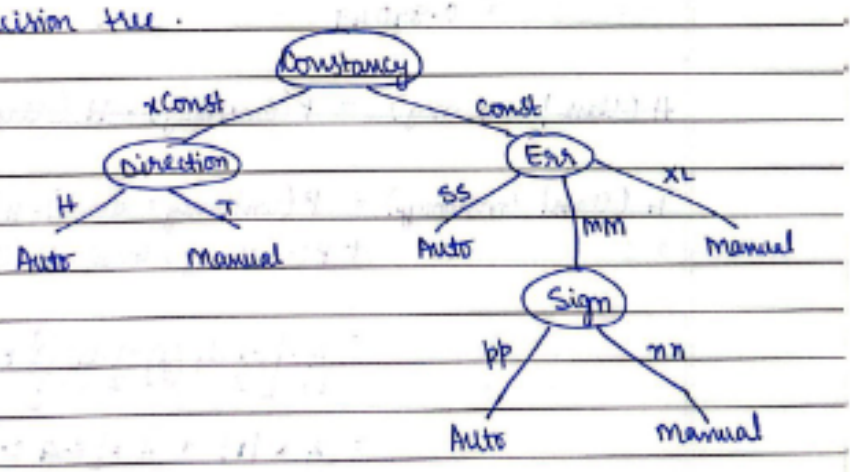
$$= \frac{1}{5} \cdot \left[1 - 1^2 - 0^2\right] + \frac{4}{5} \cdot \left[1 - \left(\frac{4}{4}\right)^2 - 0^2\right] = 0$$

$$GINI(Direction) = \frac{3}{5} \cdot GINI(Direction = T) + \frac{2}{5} \cdot GINI(Direction = H)$$

$$= \frac{3}{5} \cdot \left[1 - \left(\frac{2}{3}\right)^2 - \left(\frac{1}{3}\right)^2\right] + \frac{2}{5} \cdot \left[1 - \left(\frac{2}{2}\right)^2 - 0^2\right]$$

$$= \frac{3}{5} \cdot \left[\frac{4}{9}\right] + \frac{2}{5} \cdot [0] = 0.26667$$

Since, GINI (Sign) has the lowest value, below is the final decision tree.



3) An eye doctor decides whether the patient should be fitted with contact lenses or not depending on various factors like patient's spectacle prescription, astigmatism, and tear production rate. We assume that in this simplified task, it is more harmful for a person who needs fitted contact lenses but does not get them rather than a person who doesn't need them but getting them.
Figure 1 below is the decision tree created by a medical student who is trying to build a classifier that can do the job of the eye doctor. Your task is to determine whether to prune the given decision tree. Additionally, you will use the provided test dataset in "lenses.csv" to determine the effectiveness of the resulting decision trees. Note, that the tree may not be constructed to maximize accuracy.

(a) (10 points) Post-pruning based on optimistic errors.

i. (3 points) Calculate the optimistic errors before splitting and after splitting using the attribute Spectacle respectively.

Sol : The optimistic errors for the given tree :

After Splitting the tree at the attribute "Spectacles":
Here we shall find two leaf nodes as mentioned in the question.
For "Hypermetrope":
No = 2
Yes = 5

Here the majority is Yes. So we take error value as "no" which is equal to 2.

For Myope :
no = 3
yes = 2

Here the majority is no. So we take the error value to be "Yes" which is equal to 2.
So error value is 2+2.
So e(t) = 2+2 / (no+yes)
= 4/12

Where 4 is the errors found and 12 is the total number of cases.

Before Splitting the subtree :

This suggests that we need to find the majority class attribute for the node "Spectacles".
Here as there is no splitting of the tree at the attribute "Spectacles" , we need to combine and add both the cases of "Hypermetrope" and "Myope" classes.

No = 2+3 = 5
Yes = 5+2 = 7

Here we can see that the "yes" value is greater than "no" value. So we consider the error value to be "no"

e(t) = no / (no+yes)
= 5/12

Where 5 is the errors found and 12 is the total number of cases.

ii) (2 points) Based upon the optimistic errors, would the subtree be pruned or retained? If it is pruned, draw the resulting decision tree and use it for the next question; otherwise, use the original decision tree shown in Figure 1 for the next question.

Sol : It wont be pruned as error e(t) is less in the subtree generated. Hence we shall keep the original decision tree as given.

iii. (5 points) Use the decision tree from (a).ii above to classify the test dataset ("lenses.csv"). Report its performance on the following five evaluation metrics: Accuracy, Recall (Sensitivity), Precision, Specificity, and F1 Measure.

Sol : Here in the decision tree which we shall use , for the attribute that we consider “Spectacles” , we consier “yes” as the class for the “Hypermetrope” and “no” for the class of “Myope” as they have the majority values as mentioned above.
So the classifier will be shown as below table:

Spectacle	Astigmatic	TearProductionRate	LenseType	Classification of Decision tree
myope	no	normal	yes	Yes
myope	yes	normal	yes	No
hypermetrope	no	normal	yes	Yes
hypermetrope	yes	normal	yes	Yes
myope	no	normal	yes	Yes
myope	yes	normal	yes	No
hypermetrope	no	normal	yes	Yes
hypermetrope	yes	normal	no	Yes
myope	no	normal	no	Yes
myope	yes	normal	yes	No
hypermetrope	no	normal	yes	Yes
hypermetrope	yes	normal	no	Yes
myope	no	reduced	no	no
myope	yes	reduced	no	no
hypermetrope	no	reduced	no	no
hypermetrope	yes	reduced	no	no
myope	no	reduced	no	no
myope	yes	reduced	no	no
hypermetrope	no	reduced	no	no

hypermetrope	yes	reduced	no	no
myope	no	reduced	no	no
myope	yes	reduced	no	no
hypermetrope	no	reduced	no	no
hypermetrope	yes	reduced	no	No

By following the decision tree as mentioned, we get the above given classifier decision.

No we shall build the confusion matrix:

PREDICTED VS ACTUAL

YY = 6 (True Positive)	YN = 3 (False Negative)
NY= 3 (False Positive)	NN = 12 (True Negative)

Let us calculate

1) Accuracy: (TP + TN)/(TP+TN+FP+FN)
   = 6+12/6+3+3+12
   = 18/24
   = 3/4

2) Sensitivity : TP/(TP+FN)
   = 6/6+3
   = 6/9
   = 2/3

3) Specificity : TN/(TN+FP)
   = 12/12+3
   = 12/15
   = 4/5

4) Precision : TP/(TP+FP)
= 6/6+3
= 6/9
= 2/3

5) F1 Measure : 2*(Precision*Sensitivity)/(Precision+Sensitivity)
= 2*(2/3 * 2/3)/(2/3 + 2/3)
= 2/3

b) (10 points) Post-pruning based on pessimistic errors. When calculating pessimistic errors, each leaf node will add a factor of 2 to the error.
i. (3 points) Calculate the pessimistic errors before splitting and after splitting using the attribute Spectacle respectively

Sol :
i) After Splitting: :

Here there are two leaf nodes for the 'spectacle' attribute:
One is for Hypermetrope which holds the values:
No = 2
yes = 5
In this case the majority value is 'yes' and hence we consider the error value as 'no'.
For Myope :
No = 3
yes = 2
Here the majority is the "no" value. Therefore the error is the "yes" = 2

So the pessimistic error
$= e'(t) = (\text{'}no\text{'} + xFactor * number\ of\ leaves)/total$
= (4 + (2*2)) / 12
= 8/12

Before Splitting : In this case we see that the:
'no' case for the attribute 'spectacles' combined is:

2 + 3 = 5.

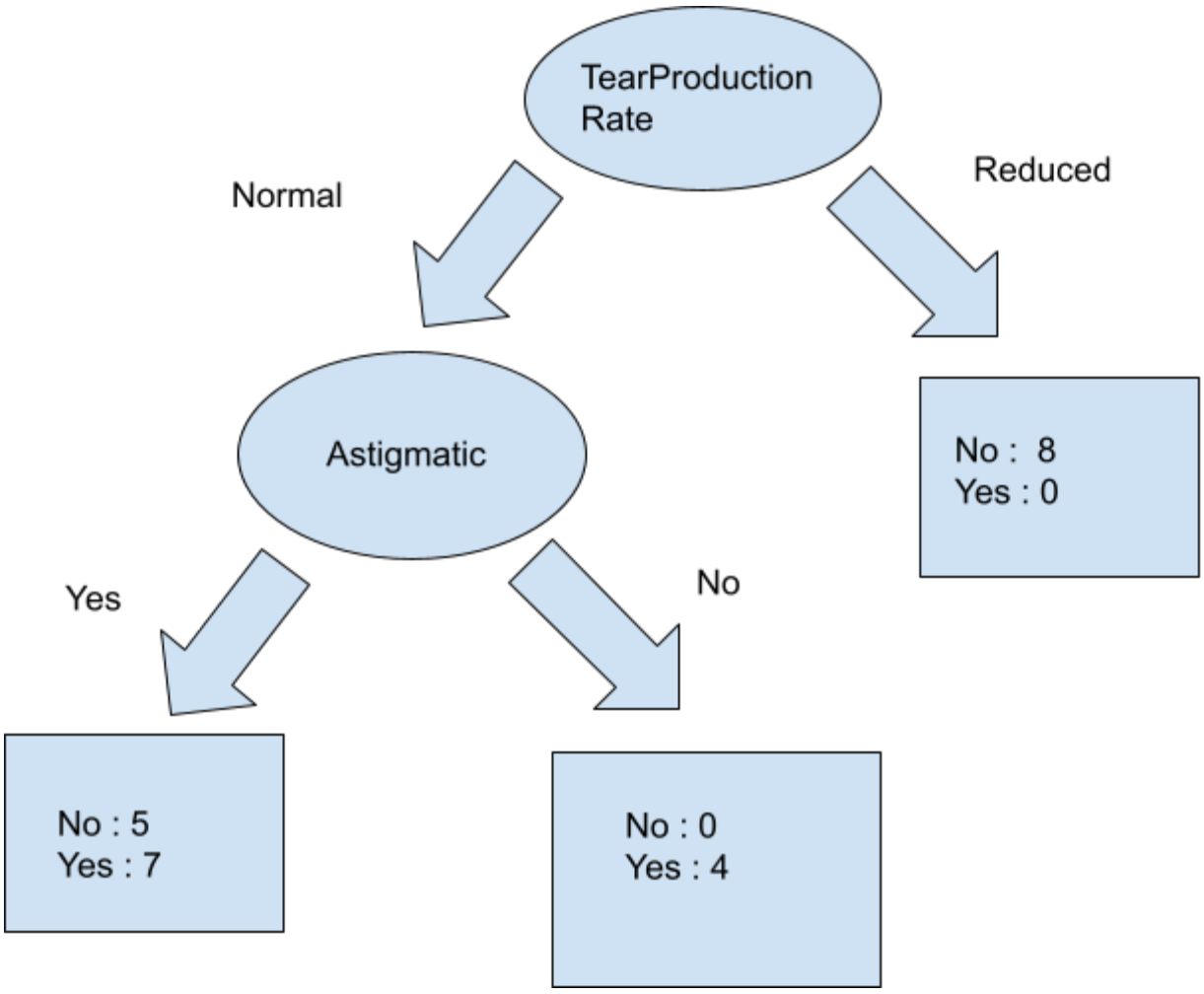
And for ‘yes’ it is :
5 + 2 = 7

So for this we can consider ‘yes’ as the majority class and consider error value as ‘no’
So the pessimistic error

= $e'(t)$ = (‘*no*’ + *xFactor* * *number of leaves*)/*total*
= (5 + (1*2)) / 12
= 7/12

ii. (2 points) Based on the pessimistic errors, would the subtree be pruned or retained? If it is pruned, draw the resulting decision tree and use it for the next question; otherwise, use the original decision tree shown in Figure 1 for the next question.

Sol: It will be pruned as the error is less in the decision tree which has not been expanded. So the Decision tree will look like this :



In this:
i) case Yes for Astigmatism leads us to predict 'Yes' as the decision and
ii) case 'No' will also lead to 'yes' as a decision in this decision tree as in both cases 'Yes' is the majority.

iii) (5 points) Use the decision tree from (a).ii above to classify the test dataset ("lenses.csv"). Report its performance on the following five evaluation metrics: Accuracy, Recall (Sensitivity), Precision, Specificity, and F1 Measure.

Sol: Here in the decision tree which we shall consier “yes” for both the cases of Astigmatism as ‘yes’ dominates in both the cases as shown in the decision tree. So the classifier will be shown as below table:

Spectacle	Astigmatic	TearProductionRate	LenseType	Classification of Decision tree
myope	no	normal	yes	Yes
myope	yes	normal	yes	Yes
hypermetrope	no	normal	yes	Yes
hypermetrope	yes	normal	yes	Yes
myope	no	normal	yes	Yes
myope	yes	normal	yes	Yes
hypermetrope	no	normal	yes	Yes
hypermetrope	yes	normal	no	Yes
myope	no	normal	no	Yes
myope	yes	normal	yes	Yes
hypermetrope	no	normal	yes	Yes
hypermetrope	yes	normal	no	Yes
myope	no	reduced	no	no
myope	yes	reduced	no	no
hypermetrope	no	reduced	no	no
hypermetrope	yes	reduced	no	no
myope	no	reduced	no	no
myope	yes	reduced	no	no
hypermetrope	no	reduced	no	no
hypermetrope	yes	reduced	no	no

myope	no	reduced	no	no
myope	yes	reduced	no	no
hypermetrope	no	reduced	no	no
hypermetrope	yes	reduced	no	no

The confusion matrix for this will be :

PREDICTED VS ACTUAL

YY = 9 (True Positive)	YN = 0 (False Negative)
NY= 3 (False Positive)	NN = 12 (True Negative)

i) Accuracy : (TP + TN)/(TP+TN+FP+FN)
= 9+12/9+0+3+12
= 21/24
= 7/8

ii) Sensitivity: = TP/(TP+FN)
= 9/(9+0)
= 1

iii) Specificity: = TN/(TN+FP)
= 12/12+3
= 12/15
= 4/5

iv) Precision = TP/(TP+FP)
= 9/9+3
= 9/12

= 3/4

v) F1 Measure = 2 * (Precision*Sensitivity)/(Precision+Sensitivity)
= (2*(3/4*1))/(3/4+1)
= 6/7

c) 10 points) We will compare the performance of the decision trees from (a).ii and from (b).ii using the test dataset ("lenses.csv"). For the task of the eye doctor, which of the five evaluation metrics: Accuracy, Recall(Sensitivity), Precision, Specificity, and F1 Measure, are the most important? Based on your selected evaluation metrics, which decision tree, (a).ii or (b).ii, is better for this task? Justify your answers.

Sol: In this solution we compare the values of Accuracy,Specificity, Sensitivity, Precision and F1 Measure for which we got for the respective decision trees without pruning which we got for a (ii) and after we pruned which we did it in the case of b (ii).

For this problem let us consider Tree1 to be the decision tree of a (ii) and Tree2 to be the decision tree of b (ii).
Now we shall compare the values and performances which we obtained for the 2 decision trees Tree1 and Tree2.

i) Accuracy :
Tree1 vs Tree2
3/4 vs 7/8
Here when we compute we shall see that 0.75 < 0.875.
Which means Tree1 < Tree2 in terms of accuracy.
Here we can conclude that Tree2 performs better than Tree1 in terms of accuracy.

ii) Sensitivity
Tree1 vs Tree2

2/3 vs 1
When we compute we clearly see that 2/3 < 1.
Tree2 performs better than Tree1 in terms of Sensitivity.

iii) Specificity
Tree1 vs Tree2
4/5 vs 4/5
0.8 vs 0.8
Here we can see that 0.8 < 0.8
Again we see that Tree2 value for specificity is better than Tree1.

iv) Precision
Tree1 vs Tree2
2/3 vs 3/4
2/3 < 3/4
Again we see that Tree2 is more precise than Tree1.

v) F1 Measure
Tree1 vs Tree2
2/3 vs 6/7
Tree2 > Tree1 again.

Out of these 5 values we calculated we must decide based on the condition/sentence which is given in the question : " It is more harmful for a person who needs fitted contact lenses but does not get them rather than a person who doesn't need them but getting them."
Here not detetcting if a person needs fitted contact lenses when they need them is more dangerous than a person getting fitted lenses when they dont need them. So hence recall / Sensitivity must be maximised in our case. When we compare the two sensitivity we see that Tree2 has a higher recall than Tree1. Hence we can conclude that based out of sensitivity, the tree which is pruned (Tree2) is better to choose than choosing Tree1 as a decision tree.
From these we can conclude that selecting Tree2 is better than Tree1 as it performs better.